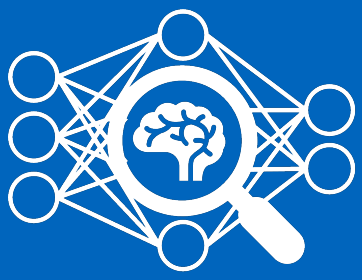


Dissertation

Rethinking Feature Attribution for Neural Network Explanation

Ashkan Khakzar

School of Computation
Information and Technology
Technische Universität München

Technische Universität München
TUM School of Computation, Information and Technology

Rethinking Feature Attribution for Neural Network Explanation

Ashkan Khakzar

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Daniel Rückert

Prüfer der Dissertation: Prof. Dr. Nassir Navab

Prof. Dr. Bernt Schiele

Prof. Gustavo Carneiro, Ph.D.

Die Dissertation wurde am 10.01.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 13.07.2023 angenommen.

Ashkan Khakzar
Rethinking Feature Attribution for Neural Network Explanation
Technische Universität München
TUM School of Computation, Information and Technology

Boltzmannstraße 3
85748 and Garching bei München

Abstract

Feature attribution is arguably the predominant approach for illuminating black-box neural networks. The objective of attribution is to identify the salient input features for the network's output. Despite its dominion in neural network explainability research in the past years and the multitude of proposed feature attribution approaches, the problem remains open. We observe a significant discrepancy among the results of different attribution methods. The phenomenon raises ambiguity regarding the trustworthiness of these methods. Moreover, the evaluation methodologies which test the sanity of feature attributions are also conflicting, further complicating the puzzle. The dissertation addresses these issues by rethinking feature attribution and looking into the problem through different lenses:

Through the lens of neural pathways: Do sparse neural pathways encode critical features of a given input? Contrary to previous belief, the pruning objective does not identify these critical pathways. We discuss finding critical pathways and leveraging them for input feature attribution.

Through the lens of Information: Feature information can be used as a proxy for the relevance of features. How can we identify input features with predictive information? We propose an approach for identifying input features with predictive information by leveraging informative deep features. Thus presenting a fine-grained input feature attribution.

Through the lens of the model: How do we know if the attribution is telling the truth? The model knows best. We leverage the model itself to generate features that impose a desired behavior on the output of the neural network. Hence, we devise a controlled experimental setup to evaluate whether attributions conform to axioms empirically.

The dissertation also rethinks feature attribution for the explanation of medical imaging models. We propose alternative ways to perform the attribution that benefits medical image analysis. Furthermore, we investigate approaches beyond attribution and identify the concepts learned by the network to discover if neural networks learn pathology-related features without any explicit cues.

Zusammenfassung

Die Feature-Attribution ist zweifellos der dominierende Ansatz, um Black-Box Neuronale Netze zu beleuchten. Das Ziel der Attribution ist es, die markanten Input-Merkmale für den Output des Netzes zu identifizieren. Trotz seiner Dominanz in der Forschung zur Erklärbarkeit neuronaler Netze in den letzten Jahren und der Vielzahl der vorgeschlagenen Ansätze zur Feature-Attribution, bleibt das Problem offen. Wir beobachten eine signifikante Diskrepanz zwischen den Ergebnissen der verschiedenen Attributionsmethoden. Dieses Phänomen wirft Zweifel an der Vertrauenswürdigkeit dieser Methoden auf. Darüber hinaus sind die Evaluierungsmethoden, mit denen die Richtigkeit der Feature-Attributionen geprüft wird, ebenfalls widersprüchlich, was das Puzzle weiter verkompliziert. Die Dissertation befasst sich mit diesen Fragen, indem sie die Feature-Attribution neu überdenkt und das Problem aus verschiedenen Perspektiven betrachtet:

Aus der Perspektive der neuronalen Verbindungen: Kodieren spärliche neuronale Verbindungen kritische Merkmale eines gegebenen Inputs? Entgegen früherer Meinungen werden diese kritischen Verbindungen durch das Pruning-Ziel nicht identifiziert. Wir diskutieren die Suche nach kritischen Verbindungen und deren Nutzung für die Feature-Attribution.

Aus der Perspektive der Information: Feature-Informationen können als Proxy für die Relevanz von Merkmalen verwendet werden. Wie können wir Input-Features mit prädiktiven Informationen identifizieren? Wir schlagen einen Ansatz zur Identifizierung von Eingangsmerkmalen mit prädiktiven Informationen vor, indem wir informative tiefe Merkmale nutzen. Auf diese Weise wird eine detaillierte Feature-Attribution ermöglicht.

Aus der Perspektive des Modells: Woher wissen wir, ob die Attribution der Wahrheit entspricht? Das Modell weiß es am besten. Wir nutzen das Modell selbst, um Merkmale zu erzeugen, die der Ausgabe des neuronalen Netzes ein gewünschtes Verhalten aufzwingen. Daher entwickeln wir einen kontrollierten experimentellen Rahmen, um empirisch zu bewerten, ob die Attributionen den Axiomen folgen.

In der Dissertation wird auch die Feature-Attribution für die Erklärung von medizinischen Bildgebungsmodellen neu überdacht. Wir schlagen alternative Wege zur Attribution vor, die der medizinischen Bildanalyse zugute kommen. Außerdem untersuchen wir Ansätze, die über die Attribution hinausgehen, und identifizieren die vom Netzwerk erlernten Konzepte, um herauszufinden, ob neuronale Netzwerke pathologiebezogene Merkmale ohne explizite Anzeichen erlernen.

Contents

1 List of Publications

Included in this Dissertation for Grading

[103] **Khakzar, Ashkan**, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. "Neural response interpretation through the lens of critical pathways." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13528-13538. 2021.

[114] **Khakzar, Ashkan** *, Pedram Khorsandi *, Rozhin Nobahari *, and Nassir Navab. "Do explanations explain? model knows best." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10244-10253. 2022.

[109] Zhang, Yang *, **Khakzar Ashkan** * , Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. "Fine-grained neural network explanation by identifying input features with predictive information." Advances in Neural Information Processing Systems 34 (2021): 20040-20051.
(* denotes equal contribution)

[104] **Khakzar, Ashkan**, Sabrina Musatian, Jonas Buchberger, Icxel Valeriano Quiroz, Nikolaus Pinger, Soroosh Baselizadeh, Seong Tae Kim, and Nassir Navab. "Towards semantic interpretation of thoracic disease and covid-19 diagnosis models." International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 499-508. Springer, Cham, 2021.

[105] **Khakzar, Ashkan**, Yang Zhang, Wejdene Mansour, Yuezhi Cai, Yawei Li, Yucheng Zhang, Seong Tae Kim, and Nassir Navab. "Explaining covid-19 and thoracic pathology model predictions by identifying informative input features." International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 391-401. Springer, Cham, 2021.

Other Articles

[111] Atad, Matan, Vitalii Dmytrenko, Yitong Li, Xinyue Zhang, Matthias Keicher, Jan Kirschke, Bene Wiestler, **Ashkan Khakzar**, and Nassir Navab. "CheXplaining in Style: Counterfactual Explanations for Chest X-rays using StyleGAN." 2nd Workshop on Interpretable Machine Learning in Healthcare at ICML 2022.

[115] **Khakzar, Ashkan**, Yawei Li, Yang Zhang, Mirac Sanisoglu, Seong Tae Kim, Mina Rezaei, Bernd Bischl, and Nassir Navab. "Analyzing the Effects of Handling Data Imbalance on Learned Features from Medical Images by Looking Into the Models." 2nd Workshop on Interpretable Machine Learning in Healthcare at ICML 2022.

[112] Engstler, Paul, Matthias Keicher, David Schinz, Kristina Mach, ..., Jan S. Kirschke, **Ashkan Khakzar**, Nassir Navab. "Interpretable Vertebral Fracture Diagnosis." Interpretability of Machine Intelligence in Medical Image Computing at MICCAI 2022.

[106] Kim, Seong Tae, Leili Goli, Magdalini Paschali, **Ashkan Khakzar**, Matthias Keicher, Tobias Czempiel, Egon Burian, Rickmer Braren, Nassir Navab, and Thomas Wendler. "Longitudinal quantitative assessment of COVID-19 infection progression from chest CTs." International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 273-282. Springer, Cham, 2021.

[93] **Khakzar, Ashkan**, Soroosh Baselizadeh, and Nassir Navab. "Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods." 5th ICML Workshop on Human Interpretability in Machine Learning (WHI), arXiv preprint arXiv:2012.00362 (2020).

[89] Denner, Stefan, **Ashkan Khakzar**, Moiz Sajid, Mahdi Saleh, Ziga Spiclin, Seong Tae Kim, and Nassir Navab. "Spatio-temporal learning from longitudinal data for multiple sclerosis lesion segmentation." International MICCAI Brainlesion Workshop, pp. 111-121. Springer, Cham, 2020.

[77] **Khakzar, Ashkan**, Shadi Albarqouni, and Nassir Navab. "Learning interpretable features via adversarially robust optimization." International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 793-800. Springer, Cham, 2019.

Part I

Introduction

2 How Do Machines Think?

"Can machines think?" is the question that has had the stage since the dawn of modern computing. With the staggering progress in artificial intelligence research, the time has come to direct the spotlight on a new question, *How do machines think?*.

Since the rise of deep learning in the past decade, the artificial intelligence community has designed learning systems that are capable of tasks hitherto unimaginable. We see learning algorithms [36], [55] emerging that beat the GO champions and are slithering into real-world problems such as protein folding. AlphaFold [102] can accurately predict 3D models of protein structures and is expected to revolutionize drug discovery. The same algorithm is also recently shown to break a 50-year-old record in matrix computation efficiency. The AlphaTensor algorithm [113] can discover the fastest existing algorithm for matrix multiplication (Strassen's algorithm) by itself and even further continues to find faster algorithms. Another trend of progress is in neural architecture research. Nowadays, we see gigantic neural transformer architectures that have revolutionized natural language processing research and show eerie performance in natural language processing tasks [88]. In computer vision, the architectural progress trend started with deep convolutional neural architectures [18], [25], [32] breaking records in image classification and making quantum leaps in many other computer vision tasks. And recently, transformer architectures [90] made their way into computer vision, and we observe gigantic vision-language transformer architectures that show surprising cognitive capability in understanding the visual world [107], [110], can generate descriptions, and can generate artistic level images from text [108], [116].

In all these examples, the complexity of learning algorithms and neural architectures has reached a level that their working mechanism is not readily comprehensible to us, the humans who have designed these systems. We have reached the point where we need to analyze the learning systems *after* the systems are designed and trained to understand how they function. To understand whether meaningful representations of the world are forming in these models, whether there is causal understanding shaping up in them, and to find out whether "thinking" is emerging within these artifacts. We are already reaching the point where we want to know how machines think. Before we analyze the "thinking" of machines, we need to understand better what "thinking"

is. Let us start with an excerpt from Turing's seminal paper, Computing Machinery, and Intelligence [1]. Turing writes:

> *I propose to consider the question, 'Can machines think?' This should begin with definitions of the meaning of the terms 'machine' and 'think'. The definitions might be framed so as to reflect so far as possible the normal use of the words, but this attitude is dangerous. If the meaning of the words 'machine' and 'think' are to be found by examining how they are commonly used, it is difficult to escape the conclusion that the meaning and the answer to the question, 'Can machines think?' is to be sought in a statistical survey such as a Gallup poll. But this is absurd. Instead of attempting such a definition I shall replace the question by another, which is closely related to it and is expressed in relatively unambiguous words.*
>
> **— Alan Turing**
> Computing Machinery and Intelligence [1]

Turing points out that the question, 'Can Machines Think?' is ill-posed as one needs to define the terms' machine' and 'think'. Turing instead proceeds by replacing this question with a behavioral test, the imitation game, aka the Turing test. Within the imitation game, an interrogator asks several questions from the machine to judge whether the responses are from a human or a machine and judge whether the machine is "thinking." It better be pointed out that here I am not referring to the imitation game as was exactly proposed in the original paper. I want you to focus on the philosophical notion that Turing raises. The idea of a pragmatic and operational approach for defining 'thinking.' That is, a machine is thinking if it appears to think humanly as evaluated by a human. This notion of subjectivity is particularly of interest to us in this dissertation, as interpreting how other intelligent systems work is subjective by nature. As we humans, the biological computing machines (if you subscribe to the monism mind-body view) are the ones forming an understanding of another intelligent system. One famous critique of the notion that machines can think, especially with regard to how the imitation game defines thinking, is reflected in the following sentence [9], [15]. "Not until a machine could write a sonnet or compose a concerto because of thoughts and emotions felt, and not by the chance fall of symbols, could we agree that machine equals brain—that is, not only write it but know that it had written it." The criticism upholds that the machines must be conscious ("not only write it but know that it had written") and have direct experience ("...because of thoughts and emotions felt"). In response, in the article "Intelligent Machinery, A Heretical Theory" [9] Turing argues that we humans unequivocally accept that other humans "think"

without any knowledge about the subjective experience of others. He continues that "Instead of arguing continually over this point, it is usual to have the polite convention that everyone thinks."

Turing infuses the notion of subjectivity for the definition of thinking; that is, if a human (or, in a more general sense, an intelligent system) perceives that a system is thinking, then that system is thinking. From this view, the recent algorithms and neural networks mentioned earlier [36], [55], [107], [116] are already demonstrating some "thinking" capabilities. There are, however, other perspectives on artificial thinking. The computer scientist Dijkstra states [15], "The question of whether Machines Can Think . . . is about as relevant as the question of whether Submarines Can Swim." On a similar note, Richard Feynman, the renowned noble laureate physicist and one of the most curious and exotic thinkers, answers the question, 'Can Machines Think?' as such:

> *With regard to the question of whether we can make it to think like [human beings], my opinion is based on the following idea: We try to make these things work as efficiently as we can with the materials that we have. Materials are different than nerves, and so on. If we would like to make something that runs rapidly over the ground, then we could watch a cheetah running, and we could try to make a machine that runs like a cheetah. But it's easier to make a machine with wheels. With fast wheels or something that flies just above the ground in the air. When we make a bird, the airplanes don't fly like a bird, they fly, but they don't fly like a bird, okay? They don't flap their wings exactly.*
> ...
> *So, there's no question that the later machines are not going to think like people think, in that sense.*
>
> — **Richard Feynman**
> Idiosyncratic Thinking workshop, Computers from the Inside Out

The "thinking" that Dijkstra and Feynman refer to here is intertwined with accomplishing tasks and achieving outcomes. Feynman and Dijkstra use analogies of performing flying or moving through the water, where we don't need to have machines that move like birds or swim like humans to accomplish flight and moving through water. Through this perspective, an agent is intelligent if it achieves a specified performance or outcome on a task that requires mental faculties. From this point of view, it's not important how the agent achieves the task and whether it's solving a task similar to

a human. This line of thought is more aligned with artificial intelligence research in the past decades, where researchers pursue intelligent agents that act rationally. That is, the agents use the toolset of rational thinking to reason and solve specific tasks [15]. It is noteworthy that within this line of thinking, it only matters that the machine accomplishes the tasks. We don't care how and we don't care if it's purely rational or whether it performs like a human. Within this perspective, the machine can accomplish tasks with a combination of rationality and randomness. For instance, a robotic vacuum cleaner could accomplish the task of cleaning a room with a combination of random moves and saving the traversed trajectory or move in a structured way of moving around the edges of the room and each time moving one step towards the inner circumference until the room is clean. Within the task-based perspective of intelligence, both machines are intelligent. From this perspective, we can declare that machines can already "think".

Feynman and Dijkstra (and a good proportion of artificial intelligence researchers) stress this "practical" rationality for formalizing intelligence and thinking. However, "pure" rationality is another perspective and attempt to formalize thinking dating back to Aristotle's work on logic. This perspective grounds intelligence based on rationality. It gained a lot of attention in the artificial intelligence community in the second half of the 20th century, when programs were made for solving problems based on logic [15]. Contrary to practical rationality and human thinking perspectives on thinking, pure rationality is well-defined and formalized. And from this perspective, we can also say that a logical program "thinks". However, rationality is not equivalent to thinking per se, but merely a type of thinking. That is, as long as we accept humans are thinking, we cannot say rationality is equivalent, as we know humans don't think and act rationally [43].

So far, to better understand what "thinking" is, we went through the human thinking perspective, the pragmatic task-solving perspective, and pure rationality. The latter two perspectives are more specific as they are defined by tasks and the laws of rationality. However, "human" thinking remains ambiguous. Turing puts forward the simple idea of mimicking human thought and the subjective evaluation of another human. Philosophy and science have attempted to shed light on human thinking for millennia. Though interesting, the discussion of the philosophy of mind is out of the scope of "how machines think." But the scientific findings regarding the human mind are insightful regarding our question of interest, how machines think. Specifically, psychology, cognitive sciences, and (computational) neuroscience try to answer the same question in the domain of the human mind.

One of the starting points of the *scientific* study of the mind can be traced to the birth of the field of psychology, where one of the main questions is how humans think. This question is analogous to our research question, though instead of machines, it studies

humans. One interesting revelation within psychology was the emergence of cognitive psychology. The cognitive perspective considers the brain as an information-processing machine [15]. One of the early works establishing such a concept is “The Nature of Explanation” [4], [15], which models cognition with the following steps: An internal representation of the input is formed, then the representation is transformed by some cognitive processes, and then the final output (action) is returned. It is interesting to think about how this modeling is akin to the computational process of current neural network-based learning systems. Moreover, this modeling is relevant to the notion of explanation, that an explanation is associated with how another cognitive system (e.g., a brain) represents another system. The follow-up works in cognitive psychology started to view psychological processes as information processing models [15], [20]. Such computational modeling eventually led to the emergence of the field of cognitive sciences, where methods from psychology and computational and information sciences are interleaved to construct theories of the human mind.

Almost simultaneous to developments in psychology and cognitive sciences to understand the human mind, neuroscience studied the connection between the biological nervous system and cognitive abilities. It is shown that localized regions within the brain are associated with different cognitive capabilities. Finding the correlation between a damaged (perturbed) region of the brain and a deficit (change or ablation) in a specific cognitive function is a powerful way of studying the brain (or an intelligent system). In the following chapters, we will see analogous methods for understanding the behavior of artificial neural networks. For instance, many explanation methods identify the relevance of features for the neural network’s output by removing features. Some works also ablate neurons [69] in artificial neural networks to identify their role. Another method to decode the concepts encoded by neurons is to study the features that fire up the neurons. Some methods in artificial neural networks follow the same idea [40] to decode neurons in artificial neural networks. While neuroscience was revealing different associations between neural tissues and cognitive abilities, another branch of research popped out by mathematically modeling the behavior of the nervous system. The thread of computational neuroscience [10], [15], [19] attempts to unveil the underlying processes of cognitive capabilities via mathematical and computational models of the nervous system. If we consider the brain also as an information-processing machine, then computational neuroscience is a mathematical method of answering the question, "how do machines think?"

Computational neuroscience attempts to understand how the information-processing machine in our heads, the brain, gives rise to cognitive capabilities and thinking. In other words, using computational models, it tries to answer how biological neural systems think. And as these neural systems are computational “machines,” then computational neuroscience also studies the question of how machines think. However, one could argue that the questions of how brains think, and how machines think, are

not as simply aligned as stated. Biological neural systems, especially the human brain, are incredibly complex and black boxes. But machines such as artificial neural systems are not as complex and are designed by us, hence, we know their processes as we design every connection ourselves. Therefore, the question of how artificial neural networks think is mute. In response, first, we need to consider that simply knowing the structure of neural connections and the learning algorithm does not mean we know how they give rise to specific cognitive capabilities. Similarly, even for simple biological neural systems knowing the full map of connections (such as knowing an entire map of fruit fly connectome [96]) is not enough to understand how it gives rise to thinking. So in this regard, both questions are aligned and necessary. And though computational neuroscience is limited to studying existing biological neural systems, the question of how machines think is more inclusive and involves itself with any possible neural system and connectivity pattern imaginable. Thus it tries to understand a broader range of machines. Moreover, recent trends in artificial neural network system design show that the creators are merely assembling layers of neurons on top of each other, sometimes with a certain level of “intuition,” and come up with networks that accomplish challenging tasks and show bizarre cognitive capabilities. This trend of stacking neurons is visible in engineering small-scale computer vision tasks such as Pneumonia classification to larger-scale vision and NLP tasks with gigantic models [88], [108], [116] with billions of neurons. Therefore, even though we are designing the machines, we are not necessarily aware of how they would lead to specific "thinking" capabilities.

3 Neural Network Explanation

What do we mean by neural network explanation? In the literature, we see the terms "explanation" and "interpretation" are almost used interchangeably, and some discussions have tried to delineate between them. In many works, the term explanation is directly used to refer to feature attribution, which is only one way, among many, to explain. However, regardless of the exact definition, one notion that readily comes up from both interpretation and explanation is that they are both subjective. They involve an observer (e.g., a human), which forms an interpretation of the neural network, and provides an explanation. The interpretation/explanation is not necessarily the objective truth representing the other system. It's a description or a quasi-representation of the system. I used the terms description, and quasi-representation, as we are not talking about another representation of the system. You can represent a system differently, but all representations directly correspond with the system. On the other hand, a quasi-representation can be a simplified or even more detailed description of the other system. For instance, we humans sometimes have overly complex interpretations of the actions of people around us, and we also tend to have overly simplified representations of people around us. In the realm of neural networks, an example of a simplified representation of the neural network function at a given input point is an importance ranking of features within the input, i.e., feature attribution, which is the focus of this dissertation.

3.1 The Motivation for Explanation

Answering "How do machines think?" is not just an explanation for quenching intellectual curiosity. We just went through significant aspects of studying this research question that could be of utility by some imagination and drawing similarities between cognitive science and computational neuroscience in studying artificial intelligence systems. However, I want to motivate the study of this question even further. Often, the question arises in different discussions and scenarios, why do we need to explain/interpret intelligent systems? Or in other words, why do we need to know how machines think? As long as the system is accomplishing the tasks, everything is fine, right? I want to answer this question by pointing out its benefits by considering the current state of neural network research and the future of artificially intelligent systems.

First, a dose of reality. Let us consider a real-world medical problem, Pneumonia classification. The community has introduced machine learning models with radiologist-level classification performance [51]. We have re-implemented the model with similar classification performance in [77], [115], and we observe that features used by these models for classification are not aligned with pathologies in many cases. So, for the time being, explaining these classification systems raises the following points: Do these features generalize to new datasets? Can we rely on these features to be relevant in every patient, case, and dataset? What are these spurious features? are they predictive markers *caused* by the disease? Are they patterns inherent in the data collection (e.g., scanning device) or data storage (e.g., intensity variations) and have nothing to do with the pathology? Are they biological markers correlated with the pathology (in this case, the spurious features are informative) that the clinicians did not see before? In any case, even for diagnoses carried out by clinicians, if the diagnosis is wrong, we blame the clinician if their diagnosis is not aligned with the accepted modern medicine. We don't trust witch doctors or folk medicine, although their diagnosis and treatment might work. Why should it be any different for intelligent machines?

Trust The previous questions bear several motivations for understanding how an intelligent system thinks. The questions of whether the model would generalize, whether the features are causal, or whether the model is aligned with medical knowledge are all associated with *reliability and trustworthiness*, which is one of the primary motivations for understanding the thinking of another intelligent system. Trustworthiness is particularly relevant in mission-critical domains such as medical diagnosis. In the current state of machine learning, with example models in [51], [77], [115], where the machine learning model is performing as well as a radiologist in classification metrics but not relying on relevant features, it would not be reliable in the clinical routine. Though the system could still act as an assistant to the clinician, in which case understanding the thinking of the system could help the clinician even further in noticing features within the data that might have been missed. But do the thinking of the machines need to be aligned with humans for them to be trustworthy? No. The argument of alignment for trust loses its color for many tasks. For instance, take the traffic sign classification task in autonomous driving. The system could also be using predictive features that have nothing to do with how we read the signs. For instance, the yield sign is a unique inverted triangle. It does not matter whether the model uses the red color inside as long as it can classify the sign with its shape. In such non-critical applications, the performance itself builds trust.

Debug Even in cases where understanding the system is not needed for trust; understanding can help debug the system. Consider the autonomous driving example, where the improved performance consistently increases trust in the systems. In this domain, in unfortunate failure cases, such as accidents due to misclassification or

detection of an object, we can understand the decision-making process of the system and debug the system for the future. Consider the Pneumania classification case again. Assume the intelligent system achieves high accuracy on a specific dataset by relying on dataset-specific spurious predictive features that are not associated with pneumonia. This performance will not generalize to another Pneumonia dataset that does not contain those spurious features. By understanding how the system works and the fact that the model focuses on spurious features instead of Pneumonia, we can guide the system toward learning pneumonia-related features.

Knowledge Another question within the Pneumonia classification example is whether the features used by the models are indeed informative and predictive features that the clinicians do not use. In this case, the features can indeed be informative and predictive regarding the dataset. They might even be generalizable features. With further scientific experiments, it could also be found that they are causal features. In this case, the model has *discovered knowledge*. Knowledge discovery is another motivation behind understanding how a machine thinks. In my opinion, it could be the main motivation for interpretability in the future. When machines are reliably and consistently outperforming humans in every task by interpreting the machines, we are indeed the ones learning. Back to the current state of machine learning, knowledge discovery seems to be already emerging as a motivation for interpreting neural networks. In a new study, [101] novel mathematical intuitions for developing new theories in knot theory and representation theory are extracted by interpreting machine learning models trained on mathematical data.

3.2 The Means for Explanation

Earlier, we drew parallels between brains and intelligent machines and between computational neuroscience and the study of how machines think. Computational neuroscience [7], [10], [19] has already defined three types of analysis which I think are interesting to be applied to the study of intelligent machines. Specifically, the machines we discuss in this thesis are artificial neural networks which support this analogy even further. To understand the brain, computational neuroscience researchers have developed mechanistic, interpretive, and descriptive paradigms. We can follow the same paradigms for the study of artificial neural networks. The majority of the works, including the works on feature attribution, are within the category of the descriptive paradigm. However, the following categorization can be helpful for future explanation approaches as we move toward more complex neural networks.

Mechanistic In the mechanistic paradigm, we try to understand "*how*" the neurons behave the way they do. For instance, we try to understand how V1 cells in the

visual cortex get fired by simple patterns, such as edges in a specific orientation. We can assign a convolution model to this behavior as they behave like a convolution operation. The analogy in neural networks would be analyzing the trained filters of the early layers of a CNN as in [26] and finding out that these filters have patterns like edge detector filters and these neurons behave as edge detectors. Another analogy is studying how different detectors combine in different layers to form detectors of more advanced concepts [94]. For instance, the neural circuit analysis [94] shows how a car detector is assembled from window, wheel and car body detectors.

Interpretive In the interpretive paradigm, the question is "*why*" the brains work the way they do. For instance, why are some V1 cells in the visual cortex structured like edge detectors? The researchers have adopted the efficient coding hypothesis [3] for visual data and discovered that an efficient (with minimum filters) representation of visual data would result in filters with primary patterns such as edge detectors [5]. I.e., the sparse code for natural images resembles simple-cell feature detectors. This type of analysis in the artificial neural networks domain could answer questions on architectural design. For instance, understanding why convolutional neural networks are designed the way they are. This type of understanding demands that the new architectures and design choices be explained. Furthermore, in the future, with neural architecture search algorithms, the question tries to understand why these architectures are formed.

Descriptive In the descriptive analysis, the goal is to study to "*what*" stimuli the brain and the neurons respond. I.e., what information and features do they encode, and what information can be decoded from them. Adapting this view to artificial neural networks, the goal is to understand the features that activate neurons, which features they encode, and how we can decode the information encoded in a trained neural network. This view is most relevant to how we try to understand and explain neural networks in this thesis. And this view is aligned with most of the neural network interpretability research. We can categorize the research in this domain into methods that identify what features within an *input* are relevant for a neuron output and methods that try to decode neurons. Examples of the decoding methods can be the feature visualization approaches [13], [30], [33], [48], [65], [94] or approaches that look for features in the dataset that activate neurons [27], [40], [71], [104]. Although we use these approaches for network understanding in our thesis, the main focus is the first category. We are interested in understanding the input features relevant to the neuron's output. We do so by attributing the output of a neuron (or the network) to input features and identifying the input features relevant to the neuron output. This idea is known as feature attribution.

4 Explanation via Feature Attribution

The dominion of neural networks in Artificial Intelligence has rendered the field of eXplainable Artificial Intelligence (XAI) equivalent to explainable neural networks. And within explainable neural network research, feature attribution is arguably the dominant approach. It is so commonly used in the literature that the word is almost unanimous with explanation, and it is used interchangeably. However, feature attribution is only one way to explain/interpret neural networks. Furthermore, after all these years of insightful research and contrary to popular belief, the simple problem is far from solved.

4.1 Feature Attribution is an Open Problem

The Discrepancy Problem We observe in chapters 6 and 7 that different attribution methods are identifying different features as relevant for the network's prediction. The attributions under study cover multiple classes of attribution methods. From pioneering works to axiomatically and mathematically grounded methods. The discrepancy issue exists, and meanwhile, the machine learning community, unaware of this issue, uses these attributions to explain different models. The community's oblivion regarding this issue results in reaching wrong conclusions from problematic explanations. It raises doubts about any conclusion made from explanations so far. Many research works use explanations to support the soundness of their methods. Moreover, this oblivion can have a catastrophic and far-reaching impact in domains such as medical image analysis. Wrong explanations can give a false sense of trust in automatic medical image diagnosis. The models could be using spurious and clinically irrelevant features, but the false explanation might be aligned with the clinician's insight. For instance, in a thoracic pathology classification problem, the explanation method might highlight an area overlapping with the lungs. However, this might be due to the explanation recovering salient image features and not what the network is using for the prediction. We see this discrepancy in our work in [77]. The gradient of the network highlights regions that disagree with class activation maps. The gradient merely reflects the local sensitivity of the neural network with respect to the image. A user (clinician) might consider the gradient as an explanation without knowing what the gradient reflects. In this case, the visually pleasing result of the gradient might give a false impression that the network is using the highlighted features. We see

in our experiments in chapters 5, 6, and 7 through different evaluation perspectives repeatedly that the gradient does not reveal contributing features. By acknowledging the existence of the discrepancy problem, it's time to attend to the question that arises from this phenomenon: Which attribution method is correct?

4.2 Feature Attribution Evaluation

Methods for evaluating explanations were born simultaneously with the advent of approaches for explaining and understanding neural networks. In the beginning, evaluations were implicitly introduced alongside the explanations. In computer vision, the early feature attribution methods (aka saliency methods) were validated by their visual interpretability. For instance, the saliency maps were shown to highlight relevant objects for a classification problem. Or they were shown to be more precise and visually appealing in highlighting the objects [37], [44], [53], [68], [73]. Within these early works, if the explanation made sense visually, then it was accepted as a good saliency method. However, there is a significant problem lurking under the hoods of these approaches. There is no guarantee that the model is using the visual features we are using. Under this assumption and its resulting faulty evaluation, if a saliency method recovers salient image features irrespective of the network, it would get the stamp of approval from the visual evaluation. Indeed there are examples of such methods [63], [93], which are described in the next section. The following is a summary of different lenses through which we can evaluate feature attribution methods.

Alignment

Alignment with human intuition: One way to know whether an attribution method is identifying features relevant to the neural network's output is to see whether it is aligned with what we consider as relevant features. As naive as it might sound, this approach was among the first methods for evaluation attributions in the beginning, and it still is commonly used. Early feature attribution methods in computer vision classification models were endorsed by their visual quality and their alignment with salient objects in the image relevant to the classification output. The evaluation via alignment was at first a qualitative approach [21], [26], [31], and later it formed into a quantitative one and became an essential metric for several years [37], [44], [53], [68], [73]. Comparing the saliency maps against ground truth bounding boxes in terms of intersection over union is one example of quantitative visual evaluation [37]. Another renowned visual evaluation approach is the pointing game [68], which again compares the saliency maps with bounding boxes but instead checks whether the highest saliency value is within the bounding box and the comparison is made for different output classes to check how attribution changes focus. Comparison to a

ground truth of what humans think is important has one underlying shaky assumption. The assumption is that the model uses the same features as humans. For instance, the model could be using spurious features to classify an image (a model can classify ImageNet [12] classes by solely relying on non-robust features which do not resemble the features humans use for classifying those classes [76]). In this case, if we expect a feature attribution method to highlight the objects relevant to the class, we risk approving a method that wrongly attributes as such while discarding a saliency method that is rightfully attributing to the spurious features.

But considering the issue within the evaluation via alignment, should we avoid evaluating this way? Alignment with intuition is by no means a tool for reliably assessing an attribution method. However, the evaluation is insightful if we mind the caveats. The correct way to adopt this approach is only to get an insight to form a hypothesis regarding how the model is behaving. Moreover, the alignment evaluation can be applied to evaluate other aspects of the attribution methods. For instance, the pointing game evaluation can show that some attributions attribute to the same features when explaining different classes [93]. That is, the attribution method generates the same saliency map for different outputs. Such an observation reveals the class-insensitivity issue within several attribution methods. We reveal the class-insensitivity issue in several attribution methods in our experiments in [93]. In our work in chapter 6 we leverage visual evaluation towards identifying how fine-grained the visualization of a saliency method is. Note that we do not use visual evaluation to check whether the attribution is reliable. However, within reliable methods, a method that is more fine-grained is more desirable. For instance, if a method identifies the features that are indeed contributing to the output but does so in a visually blurry fashion, then it might not be very informative for the user. That is, the attribution might show the regions associated with pneumonia correctly but also include the surrounding area. Therefore it won't show the regions relevant for pneumonia classification precisely. In our work in chapter 6, we propose a metric to better measure this property of attribution methods. The visual evaluation is useful when the model passes other objective evaluations.

Alignment with synthetic ground truth: A fairly recent idea to evaluate explanations is devising experimental setups where we know the predictive features via the means of generating data. For instance, [87] leveraged a shape dataset generator [45] for this purpose. We have control over the shapes we generate and their color. Therefore for each classification label, we know the predictive cues. For instance, we can assign a specific label, "red rectangle," to the image that contains a red rectangle. Then we retrain the network on the synthetically generated dataset. Now an attribution method is expected to identify the red rectangle when classifying the corresponding label. A beautiful idea. However, there are some issues within this experimental setup. The issue is again that there is no guarantee that the model is using the same features

as us to classify the image as a red rectangle. Even if we manage to control the setup such that there is no other predictive feature, we don't know if the model would use the edges and the color of the object close to the edges, the entire object, or just part of the shape to do the classification. Therefore we cannot know if an attribution is rightfully highlighting a region. But the evaluation can still be very insightful as it can show the wrong behavior of many attribution methods. For instance, if attribution is always highlighting all the objects, it raises suspicion about its sanity. Or when an attribution always highlights specific features (such as edges), then the attribution might simply be performing edge detection. Thus alignment with synthetic ground truth is a promising and powerful tool in our arsenal.

In our work in chapter 7, we take the idea one step further. Generating a synthetic dataset without considering the model results in uncertainty about the features that the model would learn from the synthetic data. We propose to use the model itself to generate the features. We devise an experimental setup in which we know exactly how specific features contribute to the output prediction. Using optimization on the input space, we generate features that impose a certain behavior on the output of the network. We then test attributions in this experimental setup and test if they behave properly.

Removal

What does "importance" mean? Intuitively when we want to see the importance of evidence in an outcome, we remove the outcome and see (or ask) what happens to the outcome with the evidence removed. The removal is one way to think about importance. From this point of view, if we want to see whether a feature contributes to the output of a neural network, we can remove that feature and observe its effect on the output of the network. The more the removal of a feature disrupts the output of the network, the more important the feature is. Therefore given a relevance ranking (saliency map) for features of an input, we can sequentially remove features from the input based on their ranking and observe the output change. The experiment results in a chart (change of output vs. the number of features removed). Through this approach, we can compare various attribution methods against each other. Moreover, we can measure if an attribution is indeed identifying important features by comparing it with a random-ordered removal of features. If the method behaves akin to random removal, then we need to be doubtful about its reliability. The notion of removing features based on the relevance ranking and directly observing output change is first proposed in [52]. Later on, another variant of the idea, Sensitivity-N [39] was proposed. In this variant of feature removal analysis, a certain portion of features are first removed randomly, and the output change is measured. Subsequently, the correlation between the relevance score of these features and the output change of these features is computed. The operation is done for various numbers of features in

increasing order. The plot, in essence, reflects how the saliency values correlate with output change for various attribution methods.

[39], [52] are grounded in the removal of features. But one question remains; what do we mean by removal? I.e., what value should we replace the features with? Intuitively we replace features with values that represent absence or missingness. The representation of absence is heavily dependent on the data distribution. For instance, in a data distribution of flying birds, the absence can be represented by the sky. In this case, if we have a dataset of flying birds, a reasonable choice is to remove pixels by values representing the color of the sky. Within computer vision applications, the pixel value of zero is considered a good representative [57]. But this is questionable, as replacing pixels with black values can introduce new features and also can add out-of-distribution effects. Therefore a better choice is to consider the input data itself when removing values. For instance, for an image, we can replace the pixel values based on the surrounding pixels and the image semantics. A prototype of this idea is presented in [59]. An inpainter is one way to deal with the choice of baseline and alleviating out-of-distribution effects. The removed feature values introducing out-of-distribution effects is a significant problem. I.e., it is not clear whether the output change resulting from removing a feature is due to it being out of a distribution or it being important. For instance, an adversarially selected perturbation can have a tremendous effect on the output [23].

To address the out-of-distribution issue, [75] propose an interesting idea: Retraining. The core idea is simple and elegant. Retrain the model on the data with the features removed and analyze how the model generalizes. The idea has two variants: 1) retrain the network on the input dataset with its important features removed, and 2) retrain the network on the input dataset with its important features remaining. In both cases, we chose a certain percentage of features to remove or keep. The removal (or keeping) is done based on the relevance score of the attribution method under evaluation. The two variations convey different messages and reveal complementary insights. In the variation in which the important features are removed, if the accuracy of the model does not drop, it means the attribution missed features that were predictive by themselves. If the accuracy drops, by assuming that the accuracy drop is not due to the network and training (a controversial assumption), we can infer the removed features were indeed leveraged by the model. In the variation in which important features are kept, if the accuracy of the model is equivalent to the original or swiftly rises to the original, then the attribution method has identified features that are sufficient for the model to get to the original accuracy. The retraining approach is computationally expensive for large datasets, therefore in research papers [78], [84], [97], [103], [109] it is usually implemented on smaller datasets [17], [22], [117]. The retraining-based evaluation is a promising approach for objectively measuring the importance of features identified by attributions.

Axiomatic

Defining importance (contribution) through removal is intuitive but leads to ambiguous cases. Assume we have an image of a Persian cat as input to the network. Also, assume that the network can use either "ear" or "whiskers" to classify with maximum output score that the image is a Persian cat. If we remove any of these features from the image (e.g., cover the cat ears), we still get the same output score from the network. In this scenario, the definition of contribution based on removal assigns zero contribution to each feature. However, we are confident that the network is using at least one of these features to the output, Persian cat. What do we expect the contribution assignment approach to do in this case? Sketching up axioms can help. We can define contribution by any score assignment that adheres to several axioms. For instance, using the axioms [100] of efficiency (completeness [57]), symmetry, null-player (Dummy), and linearity we can formalize the contribution definition grounded on the notion of removal[100]. The new axiomatic formulation is designed such that it has desirable properties that we want the contribution assignment method to have. For instance, for the Persian cat scenario, adherence to the null feature axiom prohibits the assignment of zero contribution to either of the predictive features. The aforementioned axioms have an interesting property that justifies the axiomatic formulation even further. The property is that only one unique solution satisfies all the axioms. The unique solution is the Shapley value [2], [46] which was developed and formalized in the context of cooperative game theory.

Moreover, we can sketch up other axioms to impose other desirable behaviors on the attributions. For instance, a desirable property for attribution is that if we randomly change the parameters of a model and therefore change the prediction, the attribution may as well change. The attribution methods can either be shown theoretically to conform to a certain property or can be experimentally shown that they do not. For instance, [58] proposes a set of experiments (sanity checks) that evaluate whether attributions are sensitive to model parameter randomization. Surprisingly, many methods are shown to be not sensitive. Another property that is desirable for an attribution method is to satisfy class sensitivity [93], [95], [98]. The property is specific to networks with multiple output classes. We expect an attribution method to distinguish between the features contributing to two different outputs. In the early days of attribution, research [63] proved theoretically that a method [26], [31] reconstructs the input image rather than explaining the input. And the rather visually appealing results of this attribution method are due to the image reconstruction phenomenon. The work also proposed experiments to demonstrate this phenomenon in practice. We continued to show that even more methods [60], [79], [84] suffer from this issue. In [93] we hypothesize that several methods might suffer from this issue due to their formulation and perform rigorous experiments to validate the hypothesis.

4.3 The Evaluation Conundrum

We just went over several approaches for feature attribution evaluation. The question pops up, which evaluation method do we use? The short answer is that each attribution is revealing insights from a different perspective and is informative in its own regard. For instance, the feature removal evaluation [39], [52] considers the immediate effect of removing a feature on the neural network output. Indeed the output change might be due to that feature being out-of-distribution or adversarial, nonetheless, through this perspective the feature is relevant. The retraining evaluation considers features from another perspective: generalizability. If by keeping only certain features within the dataset, the network still generalizes, then these features are relevant from this perspective. The visual analysis (such as pointing game [68] and EHR [109]) can test the visual usability of the explanations. The rest of the discussed empirical evaluations test properties such as class-sensitivity [95], [98] and sensitivity to parameter randomization [58] and conformity with different axioms [114].

Is Shapley Value the Holy Grail?

With regard to axioms, one interesting question arises. If we already know the unique solution, the Shapley value [2], which satisfies [100] the axioms of efficiency, dummy, linearity, and symmetry, why do we even need evaluations? We can strive to compute the Shapley value or evaluate methods on a simple dataset where the Shapley value is computable. Indeed if we take the Shapley value as the only way to define contribution, we can consider attribution solved and close the book. However, that is simply not the case. Even if we can compute the Shapley value (it is computationally expensive [46], [70], [81], [103], as for N features, 2^N output computations are required), in my view, we still have not solved the attribution problem. First, one notion that I see usually ignored in Shapley value literature is that the works seem to miss that the Shapley value of a feature is a different notion from the Shapley value of a group of features (by definition) [11]. For example, if we compute the Shapley value of all pixels in an image containing a cat, the result will be different than computing the Shapley value of various superpixels (e.g., the parts that constitute the cat). Removing a pixel from an object can be effective in all possible coalitions with other pixels (Shapley value of zero). However, removing an entire object (superpixel) is a different story. The Shapley value of a superpixel is not equivalent to the sum of Shapley values of all the pixels within it. Then in this context, it is ambiguous how we should compute the Shapley value. If we chose superpixels, which superpixels should we select? This discussion over this phenomenon is planned for my future work. Another issue regarding the Shapley value as the ultimate feature attribution solution is that it is only *one* among several approaches for defining relevance. The Shapley value defines relevance through some desirable axioms. However, such an axiomatic definition is only one way; there are other views through which we can

identify relevance. For instance, through information [97], [109] or the activation [37], [53] of neurons.

Another issue is the sampling (choice) of missing values [91], [99], i.e. the baseline reference values. Remember that for computing the Shapley value (and other removal methods), we have to "remove" features. But what does removing mean? Though in discrete games, it is evident how to remove a player, in value-based inputs, it may not be evidently clear what value we should replace the features with in order to consider it removed. For instance, in an image, if an object is black itself, replacing several pixels instead with black values is not a good choice for removing these values. We need to consider the correlations between features in the dataset to find the right baseline for removal. The correlation is relevant to the data distribution not the feature interactions described in [92], which computes the second-order interaction between features considering the model's Hessian. An interesting [59] proposal to consider these correlations is the use of inpainters.

Nevertheless, we still need evaluations even if we accept the Shapley value as the holy grail. Since the value is not computable, approximate methods [46], [70], [100] are proposed. However, these approximations can break the methods and cause them to not conform with axioms anymore [100]. Moreover, theoretical analysis is cumbersome and impossible due to the complexity of these proposed methods. Therefore empirical evaluations of axioms and desirable properties can serve as sanity checks on the final solutions. Moreover, other evaluations (removal and retraining) reveal other issues and properties associated with these approximate methods.

Robustness of Explanations

Several works [72], [74] discuss the phenomenon that adding adversarial perturbation to inputs can "fool" the attributions, and we should strive to make attributions robust. By adding infinitesimal adversarial noise to an image, we can guide the saliency method towards identifying some new features as important while the image looks similar to the original (from our perspective) and the output is the same. For instance, we can add noise to an image such that the gradient of the output with respect to the input forms into a desired shape. [74] discuss this phenomenon and [72] provides an interesting discussion about the reason. However, stating that attribution methods are "fooled" is questionable. We have generated a new input and have introduced new features to it. Why should we expect that the network would use the original features? Indeed the feature attribution method is pointing to this change (perturbation). Thus a correct attribution indeed has to change if the features relevant to the prediction have changed.

4.4 A Tale of Feature Attribution

In this section, we go through a series of seminal contributions to the quest for solving feature attribution.

It all Started with the Gradient

The pioneering works on attribution can be attributed to [14], [21]. Both works propose a linear assumption of the models and use the model's gradient as attribution. The term saliency map was coined in [21], and since the work uses network gradient as attribution, many works refer to this specific method as saliency maps in the literature. However, the term, in general, can be used to refer to any attribution (especially in the visual domain). The gradients of the neural networks are "noisy." Therefore several methods proposed new variations based on gradients. [56] proposed smoothGrad, which smoothes the gradient by computing the network gradient for multiple versions of the input image, where each sample is generated by adding Gaussian noise. Then the resulting gradients are averaged, and the final result looks less noisy. Other works show that less noisy saliency maps are achieved by only propagating positive gradients in Deconvolution[26] and Guided Backpropagation, [31], and Excitation Backprop [68]. However, the evaluations in [58], [63], [93], [98] show these methods that propagate positive gradients to be problematic. It seems that the methods only reveal salient image features (such as edges) instead of identifying the relevant features for prediction. Therefore the results are less "noisy" and look visually appealing. The gradient-based methods do not perform well in removal-based evaluations [75], [97], [103], [109]. However, they still reflect the model behavior locally. They show the local sensitivity of the model with respect to input, which is informative.

Backpropagating the Relevance

The aforementioned methods all leverage a version of network gradients. There is another series of works that leverage backpropagation but are unfairly categorized as gradient methods. Though these methods use backpropagation, their formulation is entirely different. These methods aim to backpropagate the contribution layer by layer. For instance, DeepLift [54] backpropagates the output difference layer by layer towards the input. The difference is with respect to having set the previous layer to a baseline value. Therefore this method is based on the notion of removing a feature (setting a feature to a baseline). Another pioneering work that stays relevant to this date is Layerwise Relevance Propagation (LRP) [28], [47]. This method provides a generalized framework for backpropagating the contribution to an output layerwise through the proposal of various backpropagation rules. One specific method within the LRP formulation is the DeepTaylor [47] method that approximates the contribution to the previous layer by Taylor decomposition.

Feature Removal and the Gradient Join Forces

Another method that is unjustly regarded as a gradient-based method is the Integrated Gradients [57]. The core idea behind this method is also based on removing evidence. The method attributes the output difference to input features. The output difference is computed with respect to a baseline input (for instance, an input of zero values). The method satisfies the efficiency (completeness) axiom, meaning that the sum of the attribution scores equals the output difference (with respect to the baseline input). Integrated Gradients leverages the gradients of the network on samples ranging from the baseline input to the original input. The path integral of gradients moving from the baseline to the input equals the difference between the outputs of the baseline and the output. It is later reported that integrated gradients approximates the Shapley value in the continuous domain [100].

Why Not Just Remove Features?

There are, however, methods that are explicitly built upon the notion of feature removal. The most basic formulation will be to remove a feature and observe the resulting output difference. We can try the removal for all features within the input and use the output difference values as relevance scores. For instance, in computer vision, we can occlude the image one pixel at a time and see the effect of the removal of each pixel from the image. The method is referred to as occlusion-1 in [39]. Naturally, we can regard patches of pixels as features and carry out the same methodology. Interestingly the resulting saliency maps from removing pixels and patches differ significantly. The same observation is also reported in [86]. The issue with removing features one by one is that it's computationally expensive. We have to compute the output of a neural network as many times as the number of pixels within the image. Moreover, it's not clear how many pixels we should occlude. Should we continue pixel-wise or patch-wise? The search for masks can be an answer.

Searching for Masks

Since it's unclear which group size to occlude, [42], [44], [73], [82] propose optimizing for an occlusion mask within the image. The objective of the optimization is to find the smallest mask that keeps the output equal to the original output (naturally, the largest mask is the input image itself). Since this optimization could result in finding trivial masks (such as adversarial examples), several regularization terms, such as mask smoothing, are used [44], [73] to find meaningful perturbations. In essence, the regularization terms enforce a prior for considering groups of features. [85] introduces an interesting regularization term. During optimization, the gradients are not allowed to pass through neurons that were not originally active. In essence, they are restricting the network to the originally active subnetwork. It is observed that such a regularization alone is quite effective in sidestepping trivial solutions.

In our work [78] we observe something similar. When we restrict the network to critical pathways while generating an adversarial perturbation, the relevant pixels are perturbed (relevant as evaluated by feature removal methods [75] and visually).

In Pursuit of the Shapley Value

The Shapley value [2] is unique between methods that are grounded upon removing of features. Unique as it is the unique solution that satisfies several axioms [46], [100]. It was first developed in cooperative game theory in the 50s, and recently it was adapted to neural networks [46]. Shapley value accounts for removing a feature in all possible coalitions with other features. Due to the exponential number of possible coalitions, the Shapley value is computationally expensive, specifically in the domain of neural networks, as each inference is expensive by itself. Therefore research is proposing approximate and fast methods for computing the Shapley value [46], [70], [100]. For a discussion on Shapley value, please refer to section 4.3.

Neuron Activations

Convolutional neural networks have a special property. They keep the spatial correspondences. Only parts of the input that are within a neuron's receptive field, activate the neuron. If a neuron gets activated, we know something within the receptive field of that neuron has caused the activation, and the other regions within the input do not have any effect. Class Activation Maps (CAM) [37] uses this property to identify what input regions are relevant for a neural network's output. It is originally proposed for a convolutional neural network that has only one linear layer after the last convolutional feature map. In this architecture, we know the weight connecting each feature map (activation map) to each neuron in the last layer (the network's outputs). Therefore we know the relevance of each activation map for each output. We can thus sum the activation maps in the last layer weighted by their weights, and we would know the relevance of different activation regions for the network's output. The idea is simple but powerful. Neuron activations in the final layer represent the activation of high-level concepts, and since there is only one linear layer after them, we know their relevance (as it is a linear layer). An extension of this method, GradCAM [53] implements the idea of convolutional neural networks with more layers after the final convolutional one. The final layers are approximated with a linear one through the use of the network's gradient (hence the word Grad in GradCAM). The approximation would be detrimental as the number of layers after the final convolutional layer increases, as the layers would not act linearly anymore. Furthermore, if we try to use earlier convolutional layers, again, we have to approximate their following layers as linear. Thus the network is the most sound in the original CAM scenario.

Surrogate Linear Models

One of the most influential works in feature attribution research is [34], which not only introduces the LIME (local interpretable model-agnostic explanations) method but also motivates the need for explanations. The core idea behind the method is simple, sample several data points in the vicinity of the input, compute the network's output for each of these inputs, and fit a linear model to these input/output pairs. The linear model depends on the choice of the neighboring samples. In [34] the neighboring samples are generated by masking different regions of the input. The weights of the linear model signify the relevance scores. In essence, LIME is looking for a local linear surrogate model that implicitly captures features the original network uses at that point. Our work in chapter 5 can also be interpreted as a surrogate model. The critical pathway is a subnetwork that locally (around the input) resembles the original model and contains the critical features the original network uses at that input point.

Part II

Rethinking Feature Attribution

5 Through the Lens of Neural Pathways

The questions in this chapter stem from my curiosity regarding what happens to the internal representations of networks in the presence of adversarial examples [23]. Exploring this question led to more fundamental questions regarding neural networks and understanding them, and it branched off in this direction (explaining neural networks) ever since. A few works [35], [66], [83] discuss this "significant" change of values in the hidden representations when the input is slightly perturbed. They do so in an intuitive manner and with exciting experiments. The intuitive discussion in [66] that looks at this phenomenon through an abstract trick, critical paths, ignited my interest in the topic; of how neural networks encode (represent) input information within a sparse set of active neurons. What could this type of looking at a network response tell us about the network? Moreover, in computational neuroscience, the research in sparsity is already well-established, and it is pretty much illuminating regarding how the brain encodes input information. Let us delve into a pathway-based view of neural network internal representations without further ado.

It is observed that information is encoded by a sparse set of active neurons in the brain. Several notable works in computational neuroscience discuss this phenomenon and have compiled a collection of justifications for this behavior [6], [8]. In these works, it is also postulated that the rectifier activation functions of neurons cause sparsity. Such findings in computational neuroscience research inspired the introduction of rectifier activation functions (such as ReLU) to the artificial neural network research [16]. Using rectifier activation functions, the response of a neural network also becomes sparse, i.e., merely a subset of neurons is activated. For instance, for a VGG [25] neural network, roughly 60% of neurons are activated on average given an input image. An exciting line of research discusses the reason behind the sparsification of response using rectifiers [49], [64]. These works show using rectifiers is akin to running a simple pursuit algorithm for sparse coding. I am just scratching the surface of the fascinating realm of sparse coding of information in neural networks. The primary motivation behind this chapter is what insights can be learned about the neural network's behavior by analyzing the pathways of neurons.

Before delving into the topic, it is necessary to know what is meant by a *pathway*. This work refers to a sub-network or, equivalently, a union of paths connecting an input/out

pair within the network as a pathway. And the definition of a path is borrowed from Graph theory. Considering the neural network as a graph where neurons are nodes and their connecting weights are edges, the path is a sequence of edges connecting two neurons (or input and output). Having cleared the definition of the pathway, let us head back to the topic of acquiring insights from studying the pathways. Since the dawn of deep learning, several works have revealed that neurons within a trained artificial neural network encode human-interpretable concepts. E.g., some neurons activate by curves and others by the presence of circles within the input image [26], [48]. These analyses are similar to the neural decoding method in computational neuroscience [7], [19], where the goal is to identify to what patterns each neuron responds. It can be seen that more high-level concepts are constructed from low-level concepts as we move deeper into the neural network. For instance, circles are assembled from curves. More recently, a study on pathways [94] reveals more human interpretable *connections* on neural pathways. For instance, a combination of window and tire detectors leads to a car detector.

Discovering such interpretable concepts along pathways is already enough reason to study them in more depth. But there are findings from different perspectives that make studying neural networks through the lens of pathways more interesting. For instance, we observe that for inputs of the same/similar classes, the pathways of active neurons in artificial neural networks overlap considerably [67]. It is also shown that we can identify "*critical*" pathways associated with a given input, and we observe that the pathways associated with inputs of different classes and adversarial inputs differ significantly. But what are "*critical*" pathways? This question is the focus of the first part of this chapter. [66] use the pruning objective, and knowledge distillation [29] to extract highly sparse subsets ($\sim$87% for VGG-16 [25] on ImageNet [12]) that produce the same output for a given input. So [66] is implicitly defining the critical pathway as a sparse pathway (a subnetwork) that produces the same output for a given input. I.e., satisfying the pruning objective on a given input would lead to the critical pathway.

In the first part of this chapter, we reveal that subsets that produce the same response as the original network do not necessarily encode the critical input features. It is intuitive that given the many numbers of neurons, it is possible to select multiple subsets that produce the same output for the given input. Note that we are discussing only one input sample (the pruning literature works on another question, that is, pruning the network for the entire dataset). To show the multitude of solutions, we propose a pathological greedy pruning algorithm that, by design, searches for irrelevant pathways that satisfy the pruning objective. By "irrelevant" pathways, we refer to pathways that are not encoding the input information. We demonstrate that our greedy pruning algorithm can select pathways that were not originally active but satisfy the pruning objective. We show the same phenomenon happening for

the knowledge distillation [66] approach that solves the pruning objective. We also investigate the concepts encoded by the neurons of various pathways to check whether they encode concepts relevant to the input by using feature visualization methods [30], [48]. Decoding what specific neurons correspond to is known as neural decoding in computational neuroscience [7], [19]. Therefore we refer to it as such.

The question is if the pruning objective does not lead to critical pathways, how do we identify these pathways? We hypothesize that these pathways are comprised of critical neurons. The intuition comes from studying numerous works that have studied the contribution and the role of individual neurons for the neural network outputs. These works show how an individual neuron can be essential for the prediction capability of the network for one or a subset of classes [40], [48], [62], [69]. I.e., these neurons have high importance for the prediction of the network. This gives us the idea to investigate a pathway of individually important neurons for each input response. Therefore we can first identify critical/important neurons for a specific output and extract pathways based on the neuron contributions. For instance, we can extract the top 5% of important neurons and analyze the behavior of the selected subnetwork (We do this by replacing the other neurons with constant values equal to their original activation value.). But how do we identify important neurons? For this purpose, we rely on the axiomatic definition of importance, which is the Shapley value [2], [46], [70], [86], [100], which itself is grounded upon another intuition. A feature is important if its removal has a high effect on the output, i.e., it has a high marginal contribution to the output. We first analyze how the selected pathways using neuron contributions overlap with originally active pathways. We proceed to evaluate to what features these pathways correspond by feature visualization.

We also propose identifying corresponding input features using the critical pathways. I.e., we propose a feature attribution approach based on critical pathways. We show we can directly identify features within the image that correspond to the pathway. We achieve this by discovering an exciting phenomenon. That is, the critical pathways are locally linear subnetworks. We prove this phenomenon mathematically. The proof is in the appendix and stands on the shoulders of works on neural network linearity and activation patterns [24], [50], [80]. By leveraging the local linearity property, we can use the network's gradient with respect to the input to identify input features that correspond to the pathway. We call this method of feature attribution Pathway Gradient. We analyze this method rigorously via various attribution evaluation methods. We observe that this method indeed identifies important input features. Through these feature attribution evaluation experiments, we can further validate that the pathways of critical neurons indeed correspond to critical input features and that they are encoding critical input features.

Neural Response Interpretation through the Lens of Critical Pathways

Ashkan Khakzar[1], Soroosh Baselizadeh[1], Saurabh Khanduja[1], Christian Rupprecht[2], Seong Tae Kim [1], Nassir Navab[1]

[1] Technical University of Munich
[2] University of Oxford

It is the accepted but not the published version of the paper due to copyright restrictions.

Published in 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Published version: `https://ieeexplore.ieee.org/document/9577965`

Neural Response Interpretation through the Lens of Critical Pathways

Ashkan Khakzar[1], Soroosh Baselizadeh[1], Saurabh Khanduja[1]
Christian Rupprecht[2], Seong Tae Kim[1], Nassir Navab[1]
ashkan.khakzar@tum.de

[1]CAMP @ Technical University of Munich
[2]VGG @ University of Oxford

Abstract

Is critical input information encoded in specific sparse pathways within the neural network? In this work, we discuss the problem of identifying these critical pathways and subsequently leverage them for interpreting the network's response to an input. The pruning objective — selecting the smallest group of neurons for which the response remains equivalent to the original network — has been previously proposed for identifying critical pathways. We demonstrate that sparse pathways derived from pruning do not necessarily encode critical input information. To ensure sparse pathways include critical fragments of the encoded input information, we propose pathway selection via neurons' contribution to the response. We proceed to explain how critical pathways can reveal critical input features. We prove that pathways selected via neuron contribution are locally linear (in an ℓ_2-ball), a property that we use for proposing a feature attribution method: "pathway gradient". We validate our interpretation method using mainstream evaluation experiments. The validation of pathway gradient interpretation method further confirms that selected pathways using neuron contributions correspond to critical input features. The code[1] [2] is publicly available.

1. Introduction

Understanding the rationale behind the response of a neural network is of considerable significance. Such transparency is required for adoption and safe deployment in mission-critical domains. Interpreting the response also helps in debugging and designing neural networks, and quenches the intellectual curiosity over how neural networks function [15, 47, 24, 63, 10, 50].

What insights can we acquire about the underpinnings of a neural network's response by putting the networks under the microscope and analyzing neurons and pathways? By "*pathway*", we refer to a union of paths (equivalently a sub-network) that connect the input to the output. Discovering to what patterns neurons correspond — also known as neural decoding in computational neuroscience [6] — has revealed human interpretable concepts encoded in neurons of artificial neural networks, *e.g.* curve and circle detectors [41, 65]. Analyzing the neural pathways has recently revealed human interpretable *connections* between concepts encoded within each neuron on the pathway, *e.g.* circles being assembled from curves [40]. In this work we discuss pathways responsible for the network's response given a *specific input*, but how can we identify these pathways?

Deep rectified neural networks encode the input information using a sparse set of active neurons [12], and their inference can be deemed as a pursuit algorithm for sparse coding [43, 59]. Such sparse coding of information is akin to how biological neurons encode information in the brain [42, 16]. Yu *et al*. [64] reported that the pathways of active neurons in artificial neural networks overlap significantly for inputs of a given class. Recently, [63] proposed using the pruning objective and knowledge distillation [19] to show that significantly higher levels of sparsity (∼87% for VGG-16 [54] on ImageNet [7]) can be achieved while keeping the prediction intact. These highly sparse pathways are reported as the critical paths and are shown to be different for inputs of different classes and adversarial inputs [45, 63, 64].

We first investigate, whether these highly sparse pathways derived from the pruning objective indeed encode critical input features. *We show that the pruning objective has solutions that are not critical pathways*, even though they have the same response as the original network. To illustrate *how* the pruning objective can result in such pathways, we construct a pathological greedy pruning algorithm that by design searches for irrelevant pathways while satisfying the pruning objective. Furthermore, we analyze the pathways selected by distillation guided routing [63] and ob-

[1] https://github.com/CAMP-eXplain-AI/PathwayGrad
[2] https://github.com/CAMP-eXplain-AI/RoarTorch

serve a similar phenomenon. We also use feature visualization [41, 30] to *decode* and semantically analyze the pathways. If these pathways do not encode critical input features, how can we find such pathways? Numerous works have studied the importance of individual neurons for the neural response, and how each neuron encodes information specific to one or a subset of classes [69, 4, 36, 41]. It is therefore intuitive that selected sparse pathways should encompass *important/critical* neurons for the corresponding response. We thus investigate selecting pathways based on neuron contributions as opposed to the pruning objective. In order to compute the importance of neurons, we use notions of marginal contribution and the *Shapley* value [51, 29, 2, 60, 70]. The first section of the work is devoted to the discussion of critical pathways.

We proceed to answer how critical pathways can help us interpret the response of the network. We prove that in rectified neural networks, pathways selected by neuron contributions are locally linear. We leverage this property and propose an input feature attribution methodology which we refer to as "pathway gradient". We evaluate our attribution methodology with input degradation [48], sanity checks [1], and Remove-and-Retrain (ROAR) [20] on Cifar10 [25], Bridsnap [5], and ImageNet [7] datasets. By validating our attribution methodology, we also validate that selected pathways using neuron contributions indeed correspond to critical input features. In summary, the main contributions of the paper are:

- We show that the pruning objective does not necessarily extract critical pathways. We illustrate how the pruning can fail by proposing a pathological greedy algorithm that by design searches for irrelevant pathways. Subsequently, we propose selecting pathways based on neuron contributions instead.
- We prove that critical pathways selected by neuron contributions are locally linear (ℓ_2-ball) in rectified networks. Using local linearity, we propose a feature attribution approach, "pathway gradient", that reveals input features associated with features encoded in the critical pathways.
- We empirically show that computing contribution (approximated Shapley value) of *neurons* rather than input pixels, improves input feature attribution.

2. Background and related work

Feature visualization / Neural decoding: This task identifies which input patterns activate a neuron. One family of solutions searches for image patches within the dataset that maximize the activations of neurons [67, 65, 4]. Another series of works generates images that maximize certain neuron activations [38, 41, 8, 53, 30].

Feature attribution: Here, the problem is to find what features in the input are important for the response of a neuron. The notion of *importance/contribution* is grounded in the effect of removal of a feature on the response. The amount of output change after removing the feature is *marginal contribution*, and the average of marginal contributions of a feature in all possible coalitions with other features in the input is the *Shapley value* [51]. Due to computational complexity, several works such as integrated gradients (IntGrad) [61] and DeepSHAP [29] approximate the Shapley value. Recently it has been shown that many approximations break the axioms [60], leaving integrated gradients as a promising candidate.

A principal class of feature attribution methods use network gradients. [53, 3] propose the input gradient itself as attribution. Guided backpropagation (GBP) [57], LRP [34], and DeepLIFT [52] modify gradients during back-propagation. Class Activation Maps (CAM) [68] and GradCAM [50] perform a weighted sum of the last convolutional feature maps. Grounded in marginal contribution, other approaches (*perturbation methods*) mask the input [10, 9, 44, 62] or neurons [9, 49] and observe the output (or information flow [49]).

Evaluation of feature attribution methods: Early evaluation of interpretability relied on human perception, *e.g.* evaluation by localization accuracy [68] or pointing game [66]. However, the model could be using features outside the human annotation or even non-robust features as in [21], and such localization-based evaluations penalize the correct attribution method. Moreover, [39, 23, 55] show/prove that several methods with human interpretable attributions generate the same attribution even after the network's weights are randomized. These methods are GBP [57], Deconvolution [65], DeepTaylor(=LRP-$\alpha 1\beta 0$)[34] and Excitation BackProp [66]. To evaluate such sensitivity to model parameter randomization, sanity checks [1] have been proposed. Recently, input degradation [48] and ROAR [20] experiments have been introduced for evaluating feature importance. Each of these evaluations measure a different perspective which we explain in section 4.2.

3. Selection of critical pathways

3.1. Setup and notation

Consider a neural network $\Phi_\Theta(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}$ with ReLU activation functions, parameters $\Theta = \{\theta^1, ..., \theta^L\}$, and L hidden layers with N_i neurons in layer $i \in \{1, ..., L\}$. The total number of neurons is $N = \sum_{i=1}^{L} N_i$. We use $\mathbf{z}^i \in \mathbb{R}^{N_i}$ to represent pre-activation vector at layer i, and $\mathbf{a}^i \in \mathbb{R}^{N_i}$ for representing the corresponding activation vector, where $\mathbf{a}^i = \text{ReLU}(\mathbf{z}^i)$, $\mathbf{z}^i = \theta^i \mathbf{a}^{i-1} + \mathbf{b}^i$, and $\mathbf{a}^0 = \mathbf{x}$. Note, our definition of $\Phi_\theta(\mathbf{x})$ has a single real valued output. The reason is that we are considering the neural pathway to one neuron, and this neuron could in fact be a hidden neuron in a larger network. Thus the response is defined by

$\Phi_\theta(\mathbf{x}) = \theta^{L+1}\mathbf{a}^L + \mathbf{b}^{L+1}$. Each individual neuron in layer i is specified by index $j \in \{1, ..., N_i\}$, thus denoted by $\mathbf{z}^i_j$ and $\mathbf{a}^i_j$. The vectors $\mathbf{z}^i$ and $\mathbf{a}^i$ are specifically associated with input $\mathbf{x}$. The vector containing activations of all N neurons is denoted by $\mathbf{a} = [\mathbf{a}^i_j]^N$ (same notation for vectors of other entities related to neurons). $\{\mathbf{a}^i_j\}^{N_i}_{j=1}$ denotes the set of neurons in layer N_i, and $\{0, 1\}^N$ denotes a set of size N containing 0s and 1s.

3.2. Selection by pruning objective

We first describe the pruning objective [28] and discuss how *a solution satisfying this objective does not necessitate it being critical*. Let $\mathbf{m} = \{0, 1\}^N$ be a mask that represents the neurons to be kept and pruned. The pruning objective given an input $\mathbf{x}$ is then defined as:

$$\arg\min_{\mathbf{m}} \mathcal{L}\left(\Phi_\theta(\mathbf{x}), \Phi_\theta(\mathbf{x}; \mathbf{m} \odot \mathbf{a})\right) \quad \text{s.t. } \|\mathbf{m}\|_0 \leq \kappa, \quad (1)$$

where $\odot$ and $\mathcal{L}$ denote the Hadamard product and the loss respectively. κ controls the sparsity. Equation (1) is a combinatorial optimization problem and a plethora of solutions exist [26, 18, 17, 33, 28].

Does the pruning objective result in sparse pathways that encode the input? Rectified neural networks have sparse positive activations [12, 42, 43, 59], and encode the input in a sparse set of *active* neurons [42, 12]. Thus, the network represents discriminative and infrequent features (i.e. features with high information) by sparse/infrequent activation values. In this regime, positive activations are detectors of features [4, 8, 67, 41, 65] and encode features that *exist* in the input. Zero activations represent missingness of features. Therefore, if a neuron is dead (zero), its corresponding features do not exist in the input. We also posit that a dead neuron does not contribute to the output (considering zero activation as a baseline for missingness):

Lemma 1 (Dead Neurons) *Considering $\mathbf{a}^i$ as the input at layer i to the following layers of the network defined by function $\Phi^{>i}_\theta(.) : \mathbb{R}^{N_i} \to \mathbb{R}$, the Shapley value of a neuron $\mathbf{a}^i_j$ defined by $\sum_{C \subseteq \{\mathbf{a}^i_j\}^{N_i}_{j=1} \setminus \mathbf{a}^i_j} \frac{|C|!(N_i - |C| - 1)!}{N_i} (\Phi^{>i}_\theta(C \cup \mathbf{a}^i_j) - \Phi^{>i}_\theta(C))$ is zero if the neuron is dead ($\mathbf{a}^i_j = 0$).*

Lemma 1 (proofs for lemmas and propositions are provided in the appendix) shows that dead neurons have a Shapley value of zero, thus do not contribute to the response. We proceed to explain how the pruning objective can select originally dead neurons as critical neurons. Removing an active neuron results in a change in inputs to next layer's neurons and thus *may* change their activation value, and this *can* result in activating an originally dead neuron. We consider a sparse selected pathway *undesirable* if it contains neurons that were originally dead, but have become active due to the pruning of other neurons. Such a selected pathway is an artificial construct, which does not reflect the original sparse encoding of the input.

Algorithm 1: Pathological greedy pruning

initialize $\mathbf{m}^i_j = 1 \quad \forall\, i, j$
while $\|\mathbf{m}\|_0 \geq \kappa$ **do**
  $s^i_j \leftarrow |\mathbf{a}^i_j \nabla_{\mathbf{a}^i_j} \Phi_\theta(\mathbf{x})|$
  if $s^i_j \leq s_\kappa \wedge s^i_j \neq 0$ **then** $\mathbf{m}^i_j \leftarrow 0$;

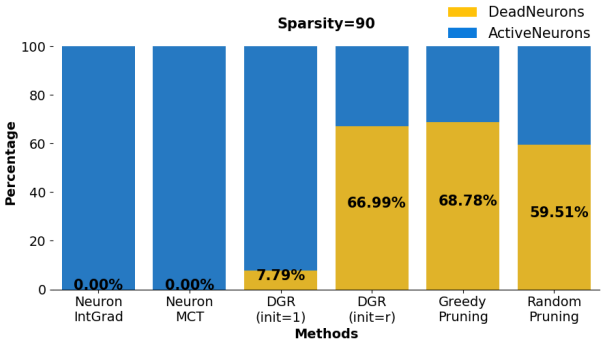


Figure 1. **Dead Neuron Selection of Pruning Objective.** The percentage of originally dead neurons in the selected pathways of different methods reported for sparsity of 90% (see appendix for more sparsity values). Evaluation on pathways extracted from VGG-16 on 1k ImageNet images. All pathways selected by pruning objective contain originally dead (now active) neurons. The observation that when selecting the top 10% of critical neurons, pruning methods select neurons from the dead regions of the network (which after pruning become active) points to the fact that they are selecting pathways unrelated to the input. Our proposal is to use neuron contributions (our NeuronIntGrad, NeuronMCT).

Pathological greedy pruning: We construct a greedy algorithm that by design searches for irrelevant pathways while solving the pruning objective (Eq. (1)). The algorithm illustrates how originally dead neurons turn active and become part of the highly sparse selected pathway that produces the same network response. Our greedy approach (Alg. 1) first ranks all neurons based on their relevance score for the response. The relevance is determined by the effect of removing a neuron, approximated by Taylor expansion similar to [33, 37]. The relevance score s^i_j is then:

$$s^i_j = |\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}^i_j \leftarrow 0)| = |\mathbf{a}^i_j \nabla_{\mathbf{a}^i_j} \Phi_\theta(\mathbf{x})| \,. \quad (2)$$

Next we remove the neuron(s) with the lowest rank, and alternate between rank computation and removal. However, *we tweak the algorithm to find pathways that contain originally dead neurons*. At each removal step, we remove the lowest contributing neuron that is not dead (without this *crucial step*, dead neurons will be pruned before others as their relevance score is zero). By removing a non-zero neuron, the activation pattern can change and some originally

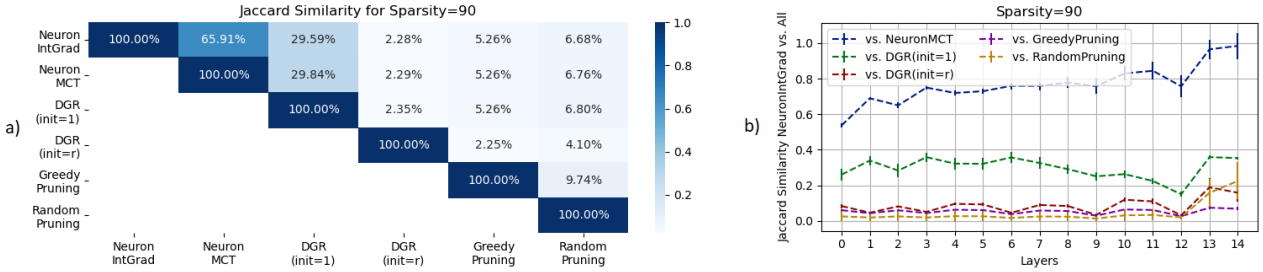


Figure 2. **Pathway Analysis.** Overlap between pathways of methods: a) in entire network. b) layer-wise overlap between pathways of all methods and NeuronIntGrad. Among the pruning-based methods, only DGR(init=1) does not diverge far from originally active pathways.

dead neurons can activate and become included in the pathway. Our algorithm illustrates that solving for the pruning objective can result in undesirable pathways.

Distillation Guided Routing: DGR [63] relaxes the pruning objective (Eq. (1)) by replacing $\mathbf{m} = \{0,1\}^N$ with continuous valued gates $0 \leq \lambda_j^i \in \mathbb{R}$. To induce sparsity, the objective is regularized with an ℓ_1 norm, *i.e.* $\left\|\lambda_j^i\right\|_1$:

$$\min_{\Lambda} \mathcal{L}(\Phi_\theta(\mathbf{x}), \Phi_\theta(\mathbf{x}; \Lambda \odot \mathbf{a})) + \gamma \sum_{k=1}^{N} \left\|\lambda_j^i\right\|_1 \quad \text{s.t. } \lambda_j^i \geq 0 \,, \tag{3}$$

where $\Lambda = [\lambda_j^i]^N$ is the vector of all λ_j^i. In our experiments we find that the initial value of Λ plays a significant role. Wang *et al.* [63] use $\lambda_j^i = 1 \; \forall i, j$ without discussing its role. We denote different initilizations with DGR(init=*value*).

3.3. Selection by neuron contribution

Individual neuron ablation [69] and network dissection [4] reveal that specific neurons are critical for certain classes. It is therefore intuitive that critical pathways contain important neurons. The effect of removing a unit, $|\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}_j^i \leftarrow 0)|$, is called the marginal contribution. Computing the exact value for the marginal contribution of all neurons is computationally expensive. As we have to ablate each neuron (total N) in the network and observe its effect after inference. Therefore we use a Taylor approximation similar to Eq. (2), $\mathbf{c}_j^i = |\Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{x}; \mathbf{a}_j^i \leftarrow 0)| = |\mathbf{a}_j^i \nabla_{\mathbf{a}_j^i} \Phi_\theta(\mathbf{x})|$, where $\mathbf{c}_j^i$ denotes the contribution of neuron $\mathbf{a}_j^i$. Pathways selected by this method are hereon referred to as NeuronMCT, where MCT stands for Marginal Contribution Taylor.

The Shapley value is the unique definition that satisfies desirable axioms of feature attribution [29]. It is defined as the average of marginal contributions of a feature in all possible coalitions with other features in the input. For each neuron, its coalitions with neurons of the same layer are considered. This results in $2^{N_i - 1}$ possible coalitions. Considering all layers, the total required inference steps becomes $\sum_{i=1}^{L} 2^{N_i - 1}$, which is computationally expensive. Thus we use an approximation method. The IntGrad [61] method with baseline 0 is equivalent to the Aumann-Shapley value, which is an extension of the Shapley value to continuous setting [60].

The contribution $\mathbf{c}_j^i$ using IntGrad with baseline 0 is:

$$\mathbf{c}_j^i = \mathbf{a}_j^i \int_{\alpha=0}^{1} \frac{\partial \Phi_\theta(\alpha \mathbf{a}_j^i; \mathbf{x})}{\partial \mathbf{a}_j^i} \mathrm{d}\alpha \tag{4}$$

Henceforth, the contributions assigned as such are referred to as NeuronIntGrad.

Remark 2 *NeuronMCT and NeuronIntGrad assign* $\mathbf{c}_j^i = 0$ *to a neuron with* $\mathbf{a}_j^i = 0$ *(dead neuron).*

We denote a pathway by $\mathbf{e} = [\mathbf{e}_j^i]^N$, where $\mathbf{e}_j^i \in \{0,1\}$ are indicator variables for each neuron indicating whether neuron belongs to pathway $\mathbf{e}$. Having computed the contributions $\mathbf{c}_j^i$, in order to select a pathway $\mathbf{e} = [\mathbf{e}_j^i]^N$, with sparsity value κ, we select neurons with $\mathbf{c}_j^i \geq \mathbf{c}_\kappa$ where $\mathbf{c}_\kappa$ is the contribution value of the corresponding sparsity κ in a sorted list of contributions, *i.e.* if $\mathbf{c}_j^i \geq \mathbf{c}_\kappa$ then $\mathbf{e}_j^i = 1$, else $\mathbf{e}_j^i = 0$.

Selecting the values higher than $\mathbf{c}_\kappa$ is average-case $\mathcal{O}(n)$. Thus the computational burden of selecting a pathway depends on the contribution assignment procedure. NeuronMCT requires one inference, and for NeuronIntGrad we use 50 inference steps in the experiments. For both methods, the ranking of the $\mathbf{a}_j^i$ are performed network-wise, and not layer-wise. We directly compare the contribution of neurons from different layers. This is possible because integrated gradients satisfies completeness, and for each layer i, $\sum_{j=1}^{N_i} \mathbf{c}_j^i = \Phi_\theta(\mathbf{x}) - \Phi_\theta(\mathbf{a}^i \leftarrow 0)$, making the scores directly comparable. For marginal contribution, by definition the contribution is the change in the output, so the contributions are inherently comparable.

3.4. Pathway selection experiments

Pathway analysis To corroborate the claim that the pruning objective results in undesirable pathways, we evaluate the pathways extracted by the discussed methods from a

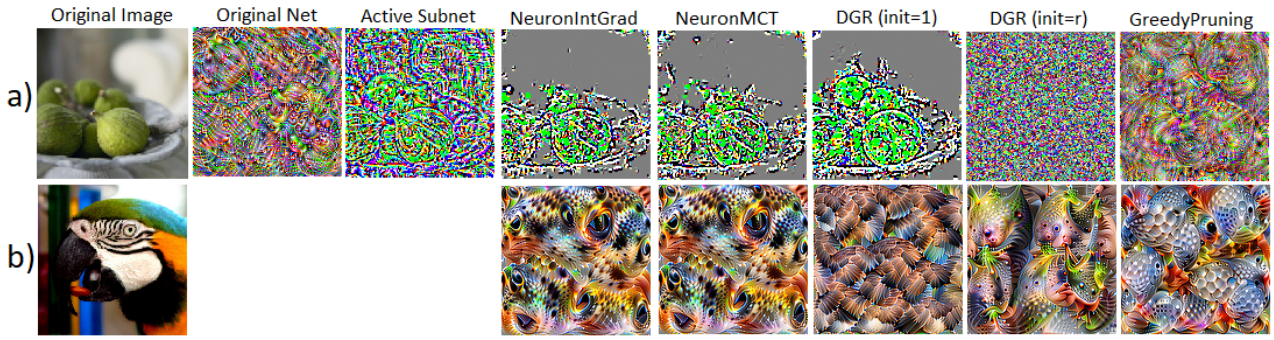


Figure 3. **Pathway Decoding.** **a)** Generating an input that maximizes the network response while restricting the network to a specific pathway selected by different methods. For the original network, the generated input contains class-specific (related to "fig" class) features. When restricted to active neurons (Active Subnet), an input similar to the original input is reconstructed. The reconstructed input for our contribution-based methods (NeuronIntGrad, NeuronMCT) contains only the critical input features (the figs) of the original image. The reconstructed images for pruning-based pathways (except DGR(init=1)) do not show any input related information. **b)** Feature visualization of the top selected neuron of the final convolutional layer in each pathway (this experiment is only relevant for pathway selection methods). The top neuron in NeuronIntGrad and NeuronMCT pathways encodes features related to the bird's eye, which is highly relevant to the input image. The top neurons in pruning-based pathways are associated with features not related to the input.

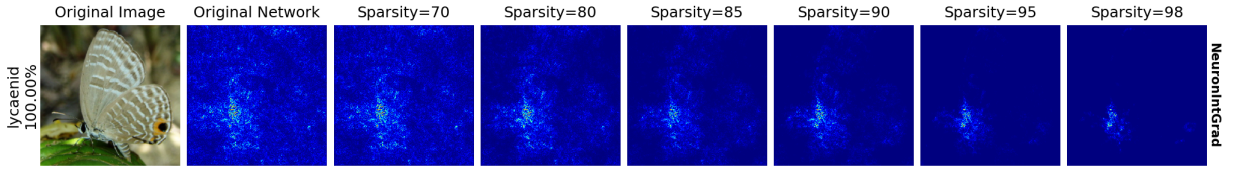


Figure 4. **Feature Attribution via Pathway Gradient.** The gradients of the locally linear critical pathways at different sparsity levels. The pathway is selected using NeuronIntGrad. The pathways are locally linear and their gradient reflects the critical input features. As we select sparser critical pathways, feature attribution reveals input features that are more critical. More examples in the supplementary.

VGG-16 [54] network for 1k ImageNet [7] images. We report results for the pathways of 90% sparsity (more values in appendix). Fig. 1 shows the percentage of previously dead neurons in the selected pathways of all methods. We observe that all pruning based methods converge to selecting undesirable pathways. $\sim$69% of neurons in the top 10% selected neurons of GreedyPruning are originally dead neurons. Another noteworthy observation is the effect of the initial value of gates in the DGR method. When gates are initialized to 1, the pathways do not drift away from the original active pathway as much as they do with random (uniform $[0, 1]$) initialization (DGR(init=r)). Nevertheless, still $\sim$8% of neurons in the top 10% of DGR(init=1) are originally dead neurons. We analyze the overlap of the selected pathways using the Jaccard similarity between pathway indicators $\mathbf{e}$ in Fig. 2a. Note the similarity between NeuronIntGrad and DGR(init=1) compared to DGR(init=r). This suggests that when initializing DGR with 1, the selected pathways do not drift significantly to undesired pathways, and they still roughly contain the critical neurons, explaining why [63] observed meaningful pathways. We also perform a layer-wise similarity analysis between pathways in Fig. 2b. We observe that the overlap between pathways of NeuronIntGrad and NeuronMCT increases as we move towards final layers, implying that the overall difference between their pathways is due to differences in earlier layers.

Pathway decoding Feature visualization estimates the input that maximizes a neuron's response. In order to generate an image $\mathbf{x}_G$ that maximizes the response, the network's weights are frozen and optimization by gradient descent is done on the input, *i.e.* $\arg\max_{\mathbf{x}_G} \Phi_\theta(\mathbf{x}_G)$. Such optimization without any regularization or priors is prone to generating adversarial artifacts [14, 41]. Hence, we optimize with preconditioning and transformation robustness techniques as in [41] to generated natural looking images. The question we are interested in here is what the pathway corresponding to the input can tell us about that input. This allows us to semantically evaluate the pathways derived from different pathway selection methods. In Fig. 3a, we generate inputs that maximize the network response while the network is restricted to different pathways. When considering the original network, features related to the predicted ("Fig") class are visualized. When we restrict the network to the active pathway (Active Subnet), optimization attempts to reconstruct the image. At 98% sparsity we observe that, critical features relevant to the predicted class are reconstructed for contribution-based methods. This signifies that the selected

sparse pathway has indeed encoded features relevant to the prediction. However, for pathways selected by the pruning objective (except DGR(init=1)), the reconstructions resemble noise. In Fig. 3b, we perform feature visualization (using entire network) of the selected *top neuron* in the final convolutional layer of each pathway. We observe that the selected top neuron by NeuronMCT and NeuronIntGrad is semantically highly relevant to the input, as the neuron is responsible for the bird's eye. The top neuron of DGR(init=1) is also relevant as it relates to feathers. While for the other methods, the top selected neuron is semantically irrelevant, further confirming that the selected pathways are not encoding the input.

4. Interpreting response via critical pathways

In Section 4.1 we show that sparse pathways selected by NeuronIntGrad and NeuronMCT are locally linear. The local linearity is later used in Section 4.2 for input feature attribution via "pathway gradients" and understanding to which features in the input the pathways correspond.

4.1. Local linearity of pathways of critical neurons

Networks with piecewise linear activation functions are piecewise linear in their output domain [35], and thus are linear at a specific point $\mathbf{x}$, and $\forall i, j$:

$$\Phi_\theta(\mathbf{x}) = (\nabla_\mathbf{x}\Phi_\theta(\mathbf{x}))^\top \mathbf{x} + \mathbf{b}^{L+1} \; ; \; \mathbf{z}^i_j = (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \mathbf{x} + \mathbf{b}^i_j \tag{5}$$

Although the network degenrates into a linear function at a given point, *it does not mean it is locally linear*. Indeed both the value [13] and the gradient [11] are unstable around a point. To discuss the local linearity of the rectified network, we need to define activation pattern [46, 27]:

Definition 3 (Activation Pattern($\mathcal{AP}$)) *$\mathcal{AP}$ is a set of indicators for neurons denoted by $\mathcal{AP} = \{\mathbb{1}(\mathbf{a}^i_j)\}^N$ where $\mathbb{1}(\mathbf{a}^i_j) = 1$ if $\mathbf{a}^i_j > 0$ and $\mathbb{1}(\mathbf{a}^i_j) = 0$ if $\mathbf{a}^i_j \leq 0$.*

The feasible set $S(\mathbf{x})$ of an $\mathcal{AP}$ is the input regions where the $\mathcal{AP}$ is constant and thus the function is linear. Let $\mathcal{B}(\mathbf{x})_{\epsilon,2} = \{\bar{\mathbf{x}} \in \mathbb{R}^D : ||\bar{\mathbf{x}} - \mathbf{x}||_2 \leq \epsilon\}$ denote the ℓ_2-ball around $\mathbf{x}$ with radius ϵ, and let $\hat{\epsilon}_{\mathbf{x},2}$ denote the largest ℓ_2-ball around $\mathbf{x}$ where the $\mathcal{AP}$ is fixed, *i.e.*

$$\hat{\epsilon}_{\mathbf{x},2} \doteq \max_{\epsilon \geq 0 : \mathcal{B}_{\epsilon,2}(\mathbf{x}) \subseteq S(\mathbf{x})} \epsilon \tag{6}$$

$\hat{\epsilon}_{\mathbf{x},2}$ is the minimum ℓ_2 distance between $\mathbf{x}$ and the corresponding hyperplanes of all neurons $\mathbf{z}^i_j$ [27]. The hyperplane defined by neuron $\mathbf{z}^i_j$ at point $\mathbf{x}$ is $\{\bar{\mathbf{x}} \in \mathbb{R}^D : (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \bar{\mathbf{x}} + \mathbf{b} = 0\}$ or $\{\bar{\mathbf{x}} \in \mathbb{R}^D : (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \bar{\mathbf{x}} + (\mathbf{z}^i_j - (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \mathbf{x}) = 0\}$. If $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$ then the distance between $\mathbf{x}$ and $\mathbf{z}^i_j$ is

$$|(\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \mathbf{x} + (\mathbf{z}^i_j - (\nabla_\mathbf{x}\mathbf{z}^i_j)^\top \mathbf{x})| / ||\nabla_\mathbf{x}\mathbf{z}^i_j||_2 = |\mathbf{z}^i_j| / ||\nabla_\mathbf{x}\mathbf{z}^i_j||_2 \tag{7}$$

For a neuron $\mathbf{z}^i_j$, if $\nabla_\mathbf{x}\mathbf{z}^i_j = 0$, then $\mathbf{z}^i_j = \mathbf{b}^i_j$. In order for the activation of this neuron to change, the $\nabla_\mathbf{x}\mathbf{z}^i_j$ and consequently the $\mathcal{AP}$ has to change. Therefore the distance is goverened by neurons for which $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$. [27] prove that $\hat{\epsilon}_{\mathbf{x},2} = \min_{i,j} |\mathbf{z}^i_j| / ||\nabla_\mathbf{x}\mathbf{z}^i_j||_2$. Since $\nabla_\mathbf{x}\mathbf{z}^i_j \neq 0$, the existence of a linear region $\hat{\epsilon}_{\mathbf{x},2}$ depends on $|\mathbf{z}^i_j|$ not being zero.

Locally linear network approximation: In order to approximate the original model $\Phi_\theta(\mathbf{x})$ with a selected pathway $\mathbf{e}$, we replace each neuron $\mathbf{a}^i_j$ which is not in the pathway, *i.e.* $\mathbf{e}^i_j = 0$ with a constant value equal to the initial value ($\mathbf{a}^i_j$) of that neuron. Note the new constant is not a neuron anymore and thus does not propagate gradient. Replacing the neuron with its initial value keeps $\mathcal{AP}$, and neurons $\mathbf{z}^i_j$ unchanged. We denote such an approximate model by $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$. Proofs are provided in the appendix.

Proposition 4 *In a* ReLU *neural network $\Phi_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$, for a pathway defined by $[\mathbf{e}^i_j]^N$, if $\mathbf{a}^i_j > 0 \; \forall \; \mathbf{e}^i_j = 1$, then there exists a linear region $\hat{\epsilon}_{\mathbf{x},2} > 0$ for $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$.*

Proposition 5 *Using NeuronIntGrad and NeuronMCT, if $\mathbf{c}_\kappa > 0$, then $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$ is locally linear.*

4.2. Input feature attribution via critical pathways

The gradient of a linear model represents the contributions of each corresponding input feature[32, 3, 53]. Therefore several works perform a first-order Taylor approximation of the network [3, 53]. However, the gradients of networks are unstable and drastically change around an input. To capture the true direction of change, SmoothGrad [56] averages gradients and LIME [32] fits a linear model to the input neighborhood. However, for a locally linear network, the gradients are already stable (constant within a region) in the linear region neighborhood. The gradient reflects the contributions of features in that neighborhood. Based on Proposition 5, the approximate model $\hat{\Phi}_\theta(\mathbf{x}; \mathbf{e})$ is locally linear for NeuronMCT and NeuronIntGrad. Thus, we can derive linear approximations for the model using the critical pathways of the model and use their gradient as attribution maps. We refer to this method as "pathway gradient". We use different levels of sparsity and observe the most critical input features for the response (Fig. 4). The attribution methodology is visually compared with other attribution methods in Fig. 5. For pathways selected via pruning, as their pathways do not contain critical input information, it would be senseless to use them for feature attribution. There is no guarantee for their linearity as Proposition 4 requires all neurons within the pathway to be active.

Computing contribution of neurons vs. input pixels: Computing the Shapley value [51] for *pixels* does not account for correlations between pixels[3]. Ideally, one should

[3]Correlation and interaction are different. The latter is related to effect of features in different coalitions which is accounted for by Shapley value

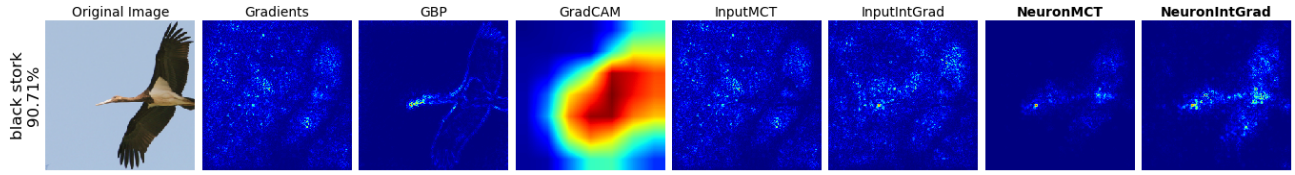


Figure 5. **Comparison with Attribution Methods.** Results for our method "pathway gradient" are shown on the right (for pathways selected by NeuronMCT and NeuronIntGrad). Our method provides pixel-level explanations as opposed to GradCAM [50]. GBP [57] is visually pleasing, but it is merely reconstructing image (Sec. 2). Note the improvement of IntGrad (integrated gradients [61]) on the neurons (NeuronIntGrad) over IntGrad on input (InputIntGrad). Also note the improvement of marginal contribution on neurons (NeuronMCT) over direct implementation on input (InputMCT = input×gradient [52]). More examples for VGG-16 / ResNet-50 in the appendix.

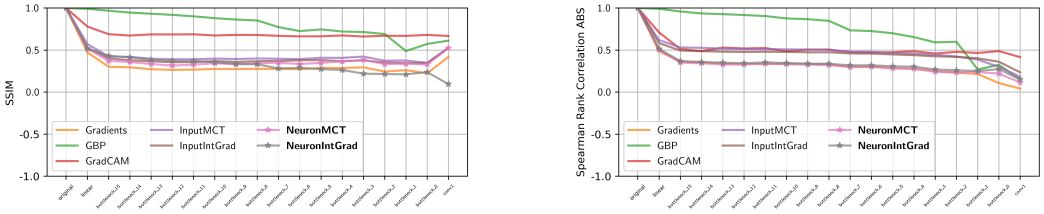


Figure 6. **Randomization-Sensitivity Sanity Check. [1]** Similarity of attributions before and after network (ResNet-50, ImageNet) parameter randomization. High similarity after randomization suggests that the attribution method is not explaining the network.

know which pixels are correlated (e.g. belong to the same object), and compute a single Shapley value for this group. The Shapley value for a group is known as the Generalized Shapley value [31]. There is an exponential number of groups of pixels. We thus aim to compute the Shapley value only for groups of *correlated* pixels *e.g.* an object (more in the appendix). Within the pathway gradient framework, we compute the Shapley value for *neurons* instead. Neurons inherently correspond to correlated groups of pixels. Thus we are indirectly computing the Shapley value (contribution) of those correlated groups of pixels. In our experiments, using MCT and IntGrad on neurons (denoted by NeuronMCT and NeuronIntGrad) results in considerably better attributions compared to applying them only to input pixels (denoted by InputMCT and InputIntGrad).

Baseline choice: The baseline in feature attribution represents the absence of a feature. In the image domain, [65, 61] consider the zero (black image) as baseline. However, zero pixel values do not necessarily reflect the absence of a feature [58, 22]. As explained in Sec. 3.2, zero neurons represent missingness in sparse rectifier networks. The zero baseline is therefore, more justified for neurons than for the input space, and has been also used in [2, 52].

4.3. Feature attribution evaluation experiments

Grounding attribution in theoretical notions such as Shapley value is desired [29, 60] (GradCAM, GBP, and Gradients are not based on this notion). However, experiments can point to specific shortcomings in methods, e.g. the approximate Shapley value (IntGrad[61]) for pixels does not perform well in experiments, which may be due to disregarding correlations between input pixels. Each of our experiments examines the methods from a different perspective. As explained in Sec. 2, visual evaluation can be unreliable. Network parameter randomization sanity checks [1] evaluate whether the method is explaining model behavior. Input degradation [48] and Remove-and-Retrain (ROAR) [20] evaluate whether the attribution maps are showing important features in the input (refer to appendix for details). We use TorchRay [9] for implementation of other attribution methods.

Network parameter randomization sanity checks [1]: Several attribution methods, such as LRP$-\alpha 1\beta 0$ [34], Excitation Backprop [66], and (GBP)[57] generate the same result after the network is randomly initialized, thus they are not explaining the network [1, 55]. In this experiment, parameters of the network are successively replaced by random weights, from the last layer to the first layer. At each step, the similarity between the attributions from the original and randomized network are reported (Fig. 6) for 1k ImageNet images (ResNet-50). It is noteworthy that GradCAM seems sensitive here, but the experiment is unfair to it due to the low dimensionality of its maps [1]. Our attributions confidently pass this sanity check.

Input degradation - LeRF [48]: Pixels are removed based on their attribution score and the output change is measured. We remove least relevant features first (LeRF). LeRF evaluates methods based on sufficiency of the features for classification and how well methods avoid assigning scores to unimportant features. Results are reported for ResNet-50 on ImageNet, Birdsnap, and Cifar. We observe (Fig. 7(a,b,c)) considerable improvement of NeuronIntGrad

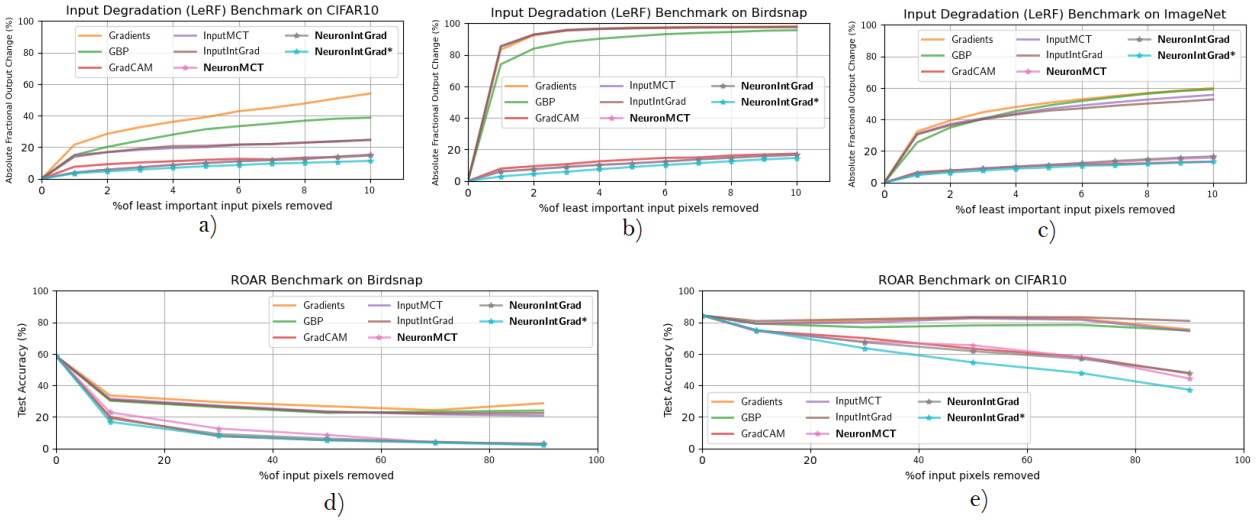


Figure 7. **Feature Importance.** a,b,c) **Input degradation-LeRF**: Removes input pixels based on their importance (least relevant features first) and measures output change. Represents how well the method avoids attributing the response to unimportant features (Bridsnap, and ImageNet). d,e) **Remove and retrain (ROAR)**: Removes top 10/30/50/70/90 percent of important pixels and retrains on the modified inputs (Birdsnap, Cifar10). If the accuracy does not drop, the attribution method is not highlighting important features.

and NeuronMCT over InputIntGrad and InputMCT.

Remove and retrain (ROAR) [20]: In input degradation experiments, the change in output might be a result of the network not having seen such artifacts during training. Therefore the ROAR benchmark retrains the network on the modified images. If the accuracy does not drop, the attribution method is not highlighting important features. The experiment is performed for different percentiles of removed pixels. Due to the large number (105) of retraining sessions required, we exclude ImageNet and use Cifar10 (ResNet-8) and Bridsnap (ResNet-50) datasets. Gradients, GBP, InputMCT and InputIntGrad are not revealing features that the model learns to use during training. We observe that NeuronIntGrad and NeuronMCT immensely improve their input counterparts (Fig. 7d,e). NeuronIntGrad and NeuronMCT are performing equally to GradCAM. GradCAM benefits from interpolation and has smooth heatmaps, which seem to help with ROAR performance. We see that smoothing (by morphological opening) on our methods (referred to by * in Fig. 7), performs best.

In summary, the attribution experiments show that attribution via critical pathways is a valid methodology with fine-grained attributions. Fine-grained (pixel-level) attributions convey more accurately which features are important to users (Fig. 5). The results support that computing marginal contribution and the Shapley value for neurons improves attribution over directly computing them for input pixels. We posit that (Sec. 4.2) this can be due to the Shapley value for pixels not accounting for correlations between them. Whereas, computing the Shapley value of neurons implicitly considers correlations between pixels. The feature attribution experiments also validate that selected pathways using neuron contributions indeed correspond to critical input features.

5. Conclusion

We demonstrate that solving the pruning objective does not necessarily yield pathways that encode critical input features. We propose finding critical pathways based on the neurons contributions to the response, and show how these sparse pathways can be leveraged for interpreting the neural response by proposing the "pathway gradient" feature attribution method. Our findings on critical pathways and pruning imply that we may need to revisit the reliability of pruned networks, and point to the direction of pruning via attribution. Moreover, considering critical pathways can be of value to other interpretation approaches, *e.g.* restricting the network to critical pathways can serve as a possible remedy to the vulnerability of perturbation-based attribution methods to adversarial solutions.

Acknowledgments

The authors acknowledge the support of the Munich Center for Machine Learning (MCML) and partial support of Siemens Healthineers. C. Rupprecht is supported by Innovate UK (project 71653) on behalf of UK Research and Innovation (UKRI) and by the European Research Council (ERC) IDIU-638009. A. Khakzar and S.T. Kim are corresponding authors.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018. 2, 7

[2] Marco Ancona, Cengiz Öztireli, and Markus Gross. Explaining deep neural networks with a polynomial time algorithm for Shapley values approximation. In *36th International Conference on Machine Learning, ICML 2019*, 2019. 2, 7

[3] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010. 2, 6

[4] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017. 2, 3, 4

[5] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, June 2014. 2

[6] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001. 1

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1, 2, 5

[8] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2, 3

[9] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2950–2958, 2019. 2, 7

[10] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017. 1, 2

[11] Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3681–3688, 2019. 6

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Journal of Machine Learning Research*, 2011. 1, 3

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016. 6

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 5

[15] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a right to explanation. *AI magazine*, 38(3):50–57, 2017. 1

[16] Richard LT Hahnloser. On the piecewise analysis of networks of linear threshold neurons. *Neural Networks*, 11(4):691–697, 1998. 1

[17] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 3

[18] Babak Hassibi and David G Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171, 1993. 3

[19] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006. 1

[20] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9737–9748, 2019. 2, 7, 8

[21] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019. 2

[22] Cosimo Izzo, Aldo Lipani, Ramin Okhrati, and Francesca Medda. A baseline for shapely values in mlps: from missingness to neutrality. *arXiv preprint arXiv:2006.04896*, 2020. 7

[23] Ashkan Khakzar, Soroosh Baselizadeh, and Nassir Navab. Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods. *arXiv preprint arXiv:2012.00362*, 2020. 2

[24] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018. 1

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2

[26] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 3

[27] Guang He Lee, David Alvarez-Melis, and Tommi S. Jaakkola. Towards robust, locally linear deep networks. In *7th International Conference on Learning Representations, ICLR 2019*, 2019. 6

[28] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. Snip: Single-shot network pruning based on connection sensitivity. *arXiv preprint arXiv:1810.02340*, 2018. 3

[29] Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017. 2, 4, 7

[30] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 2

[31] Jean-Luc Marichal, Ivan Kojadinovic, and Katsushige Fujimoto. Axiomatic characterizations of generalized values. *Discrete Applied Mathematics*, 155(1):26–43, 2007. 7

[32] Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pages 537–543, 2017. 6

[33] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 3

[34] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 2017. 2, 7

[35] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, 2014. 6

[36] Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the imporance of single directions for generalization. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018. 2

[37] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Advances in neural information processing systems*, pages 107–115, 1989. 3

[38] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016. 2

[39] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *arXiv preprint arXiv:1805.07039*, 2018. 2

[40] Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. Zoom in: An introduction to circuits. *Distill*, 2020. https://distill.pub/2020/circuits/zoom-in. 1

[41] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. 1, 2, 3, 5

[42] Bruno A Olshausen and David J Field. Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487, 2004. 1, 3

[43] Vardan Papyan, Yaniv Romano, and Michael Elad. Convolutional neural networks analyzed via convolutional sparse coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017. 1, 3

[44] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. *CVPR Workshop*, 2019. 2

[45] Yuxian Qiu, Jingwen Leng, Cong Guo, Quan Chen, Chao Li, Minyi Guo, and Yuhao Zhu. Adversarial defense through network profiling based path extraction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019. 1

[46] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. 6

[47] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 13-17-August-2016, pages 1135–1144, New York, New York, USA, aug 2016. Association for Computing Machinery. 1

[48] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. 2, 7

[49] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. 2

[50] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 1, 2, 7

[51] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. 2

[52] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning, ICML 2017*, 2017. 2, 7

[53] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2, 6

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 5

[55] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. *arXiv*, pages arXiv–1912, 2019. 2, 7

[56] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017. 6

[57] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015. 2, 7

[58] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020. 7

[59] Jeremias Sulam, Vardan Papyan, Yaniv Romano, and Michael Elad. Multilayer convolutional sparse modeling: Pursuit and dictionary learning. *IEEE Transactions on Signal Processing*, 66(15):4090–4104, 2018. 1, 3

[60] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *37th International Conference on Machine Learning, ICML 2020*, 2020. 2, 4, 7

[61] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. 2, 4, 7

[62] Jorg Wagner, Jan Mathias Kohler, Tobias Gindele, Leon Hetzel, Jakob Thaddaus Wiedemer, and Sven Behnke. Interpretable and fine-grained visual explanations for convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9097–9107, 2019. 2

[63] Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. Interpret neural networks by identifying critical data routing paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8906–8914, 2018. 1, 4, 5

[64] Fuxun Yu, Zhuwei Qin, and Xiang Chen. Distilling critical paths in convolutional neural networks. *arXiv preprint arXiv:1811.02643*, 2018. 1

[65] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 1, 2, 3, 7

[66] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018. 2, 7

[67] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014. 2, 3

[68] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 2

[69] Bolei Zhou, Yiyou Sun, David Bau, and Antonio Torralba. Revisiting the Importance of Individual Units in CNNs via Ablation. jun 2018. 2, 4

[70] Luisa M. Zintgraf, Taco S. Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. 2

6 Through the Lens of Information

In this chapter, we look into feature attribution from the information perspective. The objective of feature attribution is to identify the relevance of features (for the output). What do we mean by relevance? In the previous chapter, we formulated the relevance based on the Shapley value [2], [46] concept. The Shapley value is borrowed from cooperative game theory and identifies the contribution of players to a game. The Shapley value is formulated based on the effect of removing a player (feature). Since Shapley value is the unique solution that satisfies several axioms [100], it is becoming the holy grail of feature attribution and being considered by many the ultimate way to assign relevance. Thus many works are following up on solving the issues surrounding Shapley value, such as the choice of baseline [91], [99] and computational complexity [46], [70]. And many classical works (see section 4.4) can be associated with Shapley value as they define contribution based on the removal of features.

But we should not forget that the removal of features and Shapley value are just one way to assign relevance. One ready example is assigning relevance based on attention. Class Activation Maps [37] assign relevance based on the network's activation values (i.e. attention). It does not have the properties of methods inspired by Shapley value and, in many cases, does not conform to certain axioms. But it does not mean it is the wrong way to assign relevance. It is merely a different perspective of showing how the network works and what is relevant for the prediction. It better be reiterated, as it is very important to beware of the perspective. Shapley value and other removal-based concepts of assigning relevance do so based on the effect of removing (or perturbing) features on the output. Alternatively, methods such as class activation maps assign relevance based on the activation values of the network. The higher the activation values (and the weights connecting the activation to the output), the higher the relevance. These two perspectives tell different stories regarding how the network is reaching its prediction.

Recently another perspective based on the feature's predictive information is introduced [97]. The predictive information of features can also be used as a proxy for identifying the relevance of those features for the neural network's prediction. The idea is first to insert an information bottleneck [38] somewhere in the network. Then find the bottleneck that restricts the passed information as much as possible while keeping the output prediction. Specifically, the bottleneck limits the mutual infor-

mation between the features right before and after the bottleneck while maximizing the mutual information between features after the bottleneck and the output of the neural network. The idea identifies which features are predictive at the location where the bottleneck is inserted. Therefore, we need to find a way to map these predictive features to input if we want to find the informative input features. There is a simple solution if we use a convolutional neural network and apply the bottleneck in convolutional layers. Since there is a spatial correspondence between convolutional features (neurons/activations) with the input, we can simply rescale the values to the input dimension. The rescaled values provide a (heat)map that shows which regions have high information for the output. One problem that we can spot easily is that the rescaling results in a low-resolution map. And thus introduces an approximation. More approximations are inherent in this approach which are discussed in our paper in this chapter. Most importantly, the information bottleneck approach is more accurate when applied in deeper layers. However, the deeper we get, the lower the resolution. Therefore we face a trade-off. This chapter introduces our idea to alleviate this issue and find the features with predictive information directly in the input space.

Our idea is to search for a bottleneck on the input that induces the optimal bottleneck in the deep layers. Remember that the information bottleneck optimization on the input causes approximation errors, and the error reduces as we move toward deeper layers. Then first, finding the optimal bottleneck on a deep layer sounds like a reasonable move. We can then search for an input bottleneck that induces the optimal bottleneck on the input. Finally, we can further refine the input bottleneck by an information bottleneck optimization on the input while using the result of the previous step as a prior. The final input bottleneck represents the input features having high predictive information. Since the optimization is done in input space, the bottleneck has the exact resolution as the input. E.g., it directly corresponds to image pixels or text tokens. The fine-grained attribution resulting from input information bottleneck is not only visually favorable but also performs well in various feature importance metrics [39], [52], [75].

The idea of information bottleneck attribution is tightly associated with feature masking. In fact, the information bottleneck optimization, as proposed in [97], essentially translates to a masking optimization scheme. The optimization tries to add gaussian noise to the features (at a selected layer) while keeping the output prediction consistent with the original prediction. The masking scheme for attribution is already investigated in several works [42], [44], [73]. The masking methods are directly accomplished in the input space. But if this method already achieves results in the input space, doesn't it make our methodology for input information bottleneck attribution irrelevant? No, the masking optimization on the input is not straightforward. The space of solutions that satisfy the masking optimization scheme is vast. Some of these solutions are of adversarial nature. I.e., we can find random and

noisy-looking masks that change the output significantly. To avoid these solutions, the masking methods add priors to the optimization. For instance, they add a smoothness prior to identifying smooth and consistent to limit the search within "natural" looking masks. These operations reduce the fine-graininess of the masks. Fine graininess is one property that gives our method a competitive edge.

But more importantly, our experiments in the previous chapter led us to form an intuition that constraining the search space using the deep features sidesteps irrelevant solutions. In the last chapter, we observed that the gradient of a pathway consisting of critical neurons is aligned with critical input features. Our auxiliary experiments reported in [78] show that performing an adversarial perturbation on the input while constraining the pathways to critical neurons leads to perturbing critical input features. One justification for this phenomenon is that deep features (neurons) encode the correlations and interactions between input features (e.g., certain pixels belonging to an object). Thus leveraging these deep features while searching for input features indirectly considers these input feature correlations. In essence, I consider the main contribution of this chapter to the feature attribution research, guiding the search for critical input features by critical deep features.

Fine-grained Neural Network Explanation by Identifying Input Features with Predictive Information

Yang Zhang[1,*], Ashkan Khakzar[1,*]
Yawei Li[2], Azade Farshad[1], Seong Tae Kim[3], Nassir Navab[2]

[1] Technical University of Munich
[2] LMU University Hospital, Munich
[2] Kyung Hee University, South Korea
[*] Equal contribution

Published in Advances in Neural Information Processing Systems 34 (NeurIPS 2021)

Published version at:
https://papers.nips.cc/paper/2021/hash/a6d259bfbfa2062843ef543e21d7ec8e-Abstract.html

Fine-Grained Neural Network Explanation by Identifying Input Features with Predictive Information

Yang Zhang *
Technical University of Munich
ya.zhang@tum.de

Ashkan Khakzar *
Technical University of Munich
ashkan.khakzar@tum.de

Yawei Li
LMU University Hospital ◇, Munich
yawei.li@med.uni-muenchen.de

Azade Farshad
Technical University of Munich
azade.farshad@tum.de

Seong Tae Kim †
Kyung Hee University, South Korea
st.kim@khu.ac.kr

Nassir Navab
Technical University of Munich
nassir.navab@tum.de

Abstract

One principal approach for illuminating a black-box neural network is feature attribution, i.e. identifying the importance of input features for the network's prediction. The predictive information of features is recently proposed as a proxy for the measure of their importance. So far, the predictive information is only identified for *latent* features by placing an information bottleneck within the network. We propose a method to identify features with predictive information in the *input* domain. The method results in fine-grained identification of input features' information and is agnostic to network architecture. The core idea of our method is leveraging a bottleneck on the input that only lets input features associated with predictive latent features pass through. We compare our method with several feature attribution methods using mainstream feature attribution evaluation experiments. The code [1] is publicly available.

1 Introduction

Feature attribution – identifying the contribution of input features to the output of the neural network function – is one principal approach for explaining the predictions of black-box neural networks. In recent years, a plethora of feature attribution methods is proposed. The solutions range from axiomatic methods [1, 2, 3] derived from game theory [4] to contribution backpropagation methods [5, 6, 7, 8, 9]. However, given a specific neural network and an input-output pair, existing feature attribution methods show dissimilar results. In order to evaluate which method correctly explains the prediction, the literature proposes several attribution evaluation experiments [10, 11, 12, 9]. The attribution evaluation experiments reveal that methods that look visually interpretable to humans or even methods with solid theoretical grounding are not identifying contributing input features [13, 12, 14]. The divergent results of different attribution methods and the insights from evaluation experiments show that the feature attribution problem remains unsolved. Though there is no silver bullet for feature attribution, each method is revealing a new aspect of model behavior and provides insights or new tools for getting closer to solving the problem of attribution.

*denotes equal contribution ◇Department of Dermatology and Allergology †corresponding author
[1] https://github.com/CAMP-eXplain-AI/InputIBA

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Recently a promising solution – Information Bottleneck Attribution (IBA) [15] – grounded on information theory is proposed. The method explains the prediction via measuring the predictive information of latent features. This is achieved by placing an information bottleneck on the latent features. The predictive information of *input* features is then approximated via interpolation and averaging of latent features' information. The interpolated information values are considered as the importance of input features for the prediction. One shortcoming with this approach is the variational approximation inherent in the method, leads to an overestimation of information of features when applied to earlier layers. Another problem is that the interpolation to input dimension and the averaging across channels only approximates the predictive information of input features and is only valid in convolutional neural networks (CNNs) where the feature maps keep the spatial correspondences.

In this work, we propose InputIBA to measure the predictive information of *input* features. To this end, we first search for a bottleneck on the input that only passes input features that correspond to predictive deep features. The correspondence is established via a generative model, i.e. by finding a bottleneck variable that induces the same distribution of latent features as a bottleneck on the latent features. Subsequently, we use this bottleneck as a prior for finding an input bottleneck that keeps the mutual information with the output. Our methodology measures the information of input features with the same resolution as the input dimension. Therefore, the attributions are fine-grained. Moreover, our method does not assume any architecture-specific restrictions. The core idea – input bottleneck/mask estimation *using deep layers* – is the main contribution of this work to feature attribution research (input masking itself is already an established idea [16, 17], the novelty is leveraging *information of deep features* for finding the input mask).

We comprehensively evaluate InputIBA against other methods from different schools of thought. We compare with DeepSHAP [1] and Integrated Gradients [2, 3] from the school of Shapley value methods, Guided Backpropagation [5] (from backpropagation methods), Extremal perturbations [17] (from perturbation methods), GradCAM [18] (an attention-based method), and the background method, IBA [19]. We evaluate the method on visual (ImageNet classification) and natural language processing (IMDB sentiment analysis) tasks and from different perspectives. Sensitivity to parameter randomization is evaluated by Sanity Checks [12]. Evaluating the importance of features is done using Sensitivity-N [9], Remove-and-Retrain (ROAR) [10], and Insertion-Deletion [11]. To quantify the degree of being fine-grained in visual tasks we propose a localization-based metric, Effective Heat Ratios (EHR).

2 Related Work

2.1 Explaining Predictions via Feature Attribution

We focus on explaining the models for single inputs and their local neighborhood, i.e. local explanation [20]. We categorize attribution methods within the local explanation paradigm. Some methods can belong to more than one category such as IBA which leverages perturbation and latent features.

Backpropagation-based: [21, 22] linearly approximate the network and propose the gradient as attribution. Deconvolution [23], Guided Backpropagation [5], LRP [7], Excitation Backprop [8], DeepLIFT[6] use modified backpropagation rules.

Shapley value: Considering the network's function as a score function, and input features as players, we can assign contributions to features by computing the Shapley value. Due to complexity, many approximations such as DeepSHAP [1], Integrated Gradients [2, 3] are proposed.

Perturbation-based: These methods perturb the input and observe its effect on the output value [16, 17, 24]. E.g. Extremal Perturbations[17] searches for the largest smooth mask on the input such that the remaining features keep the target prediction. LIME [25] linearly approximates the model around local input perturbations.

Latent Features: CAM/GradCAM [26, 18] use the activation values of final convolutional layers. IBA [19] measures the predictive information of latent features. Pathway Gradient [27] leverages critical pathways.

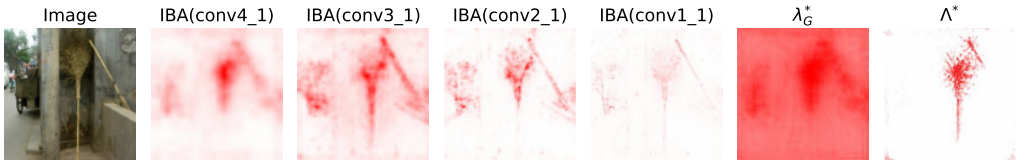


Figure 1: **Effect of $P(Z)$ Approximation (IBA [19] vs InputIBA (Λ^*))**: We see the result of applying IBA on different layers of a VGG16 network (from conv4_1 to conv1_1). The approximations (averaging across channels) in IBA result in the assignment of information to irrelevant areas of the image (the trashcan and areas around the image). As we move towards earlier layers (conv1_1), the information is distributed equally between features, and less information is assigned to the most relevant feature (the broom), due to the overestimation of mutual information $I[R, Z]$ in IBA. Using our approximation of $P(Z)$ the resulting mask Λ^* is representing only the relevant information (the broom) at input resolution. λ_G represents the prior knowledge we use for $P(Z)$. IBA uses the Gaussian distribution ($Q(Z) \sim \mathcal{N}(\mu_R, \sigma_R)$) to approximate $P(Z)$.

2.2 Do Explanations Really Explain?

The story starts with Nie et al. [13] demonstrating that Deconvolution [23] and Guided Backpropagation [5] are reconstructing image features rather than explaining the prediction. Later Adebayo et al. [12] propose sanity check experiments to provide further evidence. They show some methods generate the same attribution when the weights of the model are randomized.

Sixt et al. [14] and Khakzar et al. [28, 29] provide further proofs and experiments, and add other attribution methods to the circle. It is noteworthy that all these methods generate visually interpretable results. In parallel, several feature importance evaluations are proposed to evaluate whether the features recognized as important, are indeed contributing to the prediction. All these experiments are grounded on the intuition that, removing (perturbing) an important feature should affect the output function relatively more than other features. These methods are Remove-and-Retrain [10], Sensitivity-N [9], Insertion/Deletion [11], which are introduced in Section 4. Interestingly, such evaluations reveal that axiomatic and theoretically iron-clad methods such as Shapley value approximation methods (e.g. Integrated Gradients [2, 3] and DeepSHAP [1]) score relatively low in these feature evaluation methods.

3 Methodology

We first introduce the background method, IBA [15] in Section 3.1. We proceed in Section 3.2 with explaining IBA's shortcomings and proposing our solution. In Section 3.3 and Section 3.4 we explain details of our solution.

3.1 Background - Information Bottleneck Attribution (IBA) [15]

This method places a bottleneck Z on features R of an intermediate layer by injecting noise to R in order to restrict the flow of information. The bottleneck variable is $Z = \lambda R + (1 - \lambda)\epsilon$, where ϵ denotes the noise and λ controls the injection of the noise. λ has the same dimension as R and its elements are in $[0, 1]$. Given a specific input I and its corresponding feature map R ($R = f(I)$, function f represents the neural network up to the hidden layer of R), the method aims to remove as many features in R as possible by adding noise to them (via optimizing on λ), while keeping the target output. Therefore, only features with predictive information will pass through the bottleneck (parameterized by the mask λ). The bottleneck is thus optimized such that the mutual information between the features R and noise-injected features Z is reduced while the mutual information between the noise-injected features Z and the target output Y is maximized, i.e.

$$\max_{\lambda} I[Y, Z] - \beta I[R, Z] \tag{1}$$

where,

$$I[R, Z] = E_R[D_{KL}[P(Z|R)||P(Z)]] \tag{2}$$

3.2 Approximating the Distribution of Bottleneck P(Z)

The main challenge with computing $I[R, Z]$ is that the distribution $P(Z)$ is not tractable as we need to integrate over all possible values of R (since $P(Z) = \int P(Z|R)P(R)dR$). Therefore IBA methodology resorts to variational approximation $Q(Z) \sim \mathcal{N}(\mu_R, \sigma_R)$. The assumption is reasonable for deeper layers of the network [19]. However, as we move toward the input, the assumption leads to an over-estimation of mutual information $I[R, Z]$ [19]. The result of using such an approximation ($Q(Z) \sim \mathcal{N}(\mu_R, \sigma_R)$) is presented in Fig. 1 for various layers of a VGG-16 neural network. As the over-estimation of $I[R, Z]$ increases by moving towards the earlier layers, the optimization in Eq. (1) removes more features with noise. We can see that the predictive feature (the broom) in Fig. 1 disappears as we move IBA towards the early layers (to `conv1_1`). The approximation of $P(Z)$ is most accurate when IBA is applied on the deepest hidden layer.

In order to accomplish feature attribution for the *input* features, IBA method limits itself to convolutional architectures. First, it finds the informative features of a hidden layer by solving Eq. (1). Let λ^* denote the result of this optimization. IBA interpolates these values to the input dimension (similar to CAM [26]) and averages λ^* across channel dimension. Such interpolation is reasonable only for convolutional architectures as they keep spatial information. The interpolation and averaging introduce a further approximation into computing predictive information of input features. From this perspective, it is desirable to apply IBA to early layers to mitigate the effect of interpolation. At the same time, early layers impose an overestimation of $I[R, Z]$. Therefore, there is a trade-off between mitigating the adverse effect of interpolation/averaging and the $Q(Z) \sim \mathcal{N}(\mu_R, \sigma_R)$ approximation.

We aim to come up with a more reasonable choice for $Q(Z)$ such that it is applicable for the input I. This alleviates architecture dependency to CNNs, as the mutual information is directly computed on the input space and avoids the approximation error which results from interpolation to input dimension and the summation across channels. We take advantage of $Q(Z) \sim \mathcal{N}(\mu_R, \sigma_R)$ being reasonable for deep layers. Thus we first find a bottleneck variable Z^* parameterized by the optimal result λ^* ($Z^* = \lambda^* R + (1 - \lambda^*)\epsilon$) by solving Eq. (1). The bottleneck Z^* restricts the flow of information through the network and keeps deep features with predictive information. Then, we search for a bottleneck variable on input, Z_G, that corresponds to Z^* in the sense that applying Z_G on input inflicts Z^* on the deep layer. This translates to finding a mask on input features that admits input features which correspond to informative deep features. Thus, the goal is to find Z_G such that $P(f(Z_G)) = P(Z^*)$, where function f is the neural network function up to feature map R:

$$\min_{\lambda_G} D[P(f(Z_G))||P(Z^*)] \tag{3}$$

where $Z_G = \lambda_G I + (1 - \lambda_G)\epsilon_G$, and D represents the distance similarity metric. For details of Eq. (3) refer to Section 3.3.

The resulting Z_G^* (and its corresponding λ_G^*) from Eq. (3) is a bottleneck variable on input I that keeps input features associated with predictive deep features. The final goal is keeping only predictive features of input (Eq. (1)), therefore, Z_G can be used as prior knowledge for the distribution of input bottleneck $P(Z_I)$. To incorporate this prior, we condition $P(Z_I)$ on Z_G, i.e. $Z_I = \Lambda Z_G + (1 - \Lambda)\epsilon$ with a learnable parameter Λ as the input mask. We then proceed and solve Eq. (1) again, but this time for Z_I. The resulting mask from this optimization is denoted by Λ^*. We refer to this methodology resulting in Λ^* as **InputIBA**.

Remark 1 $P(Z_I) \sim \mathcal{N}(\lambda_G \Lambda I + (1 - \lambda_G \Lambda)\mu_I, (1 - \lambda_G \Lambda)^2 \sigma_I^2)$

Remark 2 $P(Z_I|I) \sim \mathcal{N}(\Lambda I + (1 - \Lambda)\mu_I, (1 - \Lambda)^2 \sigma_I^2)$

We can explicitly compute $D_{KL}[P(Z_I|I)||P(Z_I)]$ in order to compute mutual information $I[Z_I, I]$, the assumptions and detailed derivation are provided in the Appendix A.

Proposition 3 *For each element k of input I we have* $D_{KL}[P(Z_{I,k}|I_k)||P(Z_{I,k})] = \log \frac{1-\lambda_{G,k}\Lambda_k}{1-\Lambda_k} + \frac{(1-\Lambda_k)^2}{2(1-\lambda_{G,k}\Lambda_k)^2} + \frac{(I_k-\mu_{I,k})^2(\Lambda_k-\lambda_{G,k}\Lambda_k)^2}{2(1-\lambda_{G,k}\Lambda_k)^2\sigma_{I,k}^2} - \frac{1}{2}$

3.3 Estimating the Input Bottleneck Z_G

The objective is to find a bottleneck variable Z_G at the input level that inflicts the distribution $P(Z^*)$ on the latent layer where Z^* is computed. In other words, we search for λ_G that minimizes

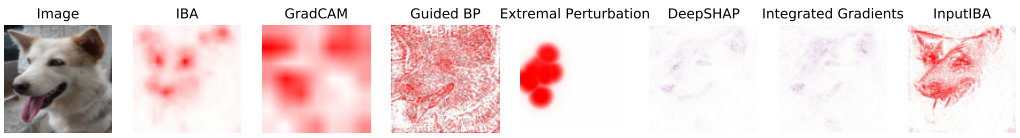


Figure 2: **Qualitative Comparison (ImageNet)**: We observe that the results of different attribution methods are substantially dissimilar for the same prediction. This is a caveat for the community that the attribution problem is far from solved. There is a consensus between GradCAM, IBA, and our method (InputIBA) that the ears and the snout are important for the prediction. Feature importance evaluations show that these methods reveal important features. InputIBA, DeepSHAP and Integrated Gradients are fine-grained. However, we observe in feature importance experiments that only the InputIBA is attributing to important features among fine-grained methods.

$D[P(f(Z_G))||P(Z^*)]$, where function f is the neural network function before bottleneck Z^*, and D is the similarity metric. We employ a generative adversarial optimization scheme to minimize the distance. Specifically, the generative adversarial model we use tries to fit the two distribution with respect to Wasserstein Distance (D) [30]. In this work we focus on local explanation, i.e. given a sample input, we aim to find an explanation that is valid for the neighbouring area of sample input ($B_\epsilon(R) := \{x \in R^n : d(x, R) < \epsilon\}$). We refer to this set, as the local explanation set. To generate $f(Z_G)$, we sample I from the local explanation set of the input. Variable $Z_G = \lambda_G I + (1 - \lambda_G)\epsilon_G$ is constructed from λ_G, μ_G and, σ_G. These are the learnable parameters of our generative model. We apply the reparametrization trick during sampling learnable noise ϵ_G (parameterized by mean μ_G and standard deviation σ_G), so that the gradient can be backpropagated to random variable ϵ_G. Once we have the sampled values (I, μ_G and σ_G) we get $Z_G = \lambda_G I + (1 - \lambda_G)\epsilon_G$, we pass this variable through the neural network to get a sample of $f(Z_G)$. The target distribution that the generator is approximating is the $P(Z^*)$. We construct the target dataset by sampling from $P(Z^*)$. The generative adversarial model also leverages a discriminator which is learned concurrently with the generator. The discriminator serves to discriminate samples from $P(f(Z_G))$ and $P(Z^*)$, thus its architecture can be determined based on the actual data type of the attribution sample.

3.4 Optimizing I[Y,Z]

Minimizing cross-entropy loss ($\mathcal{L}_{CE}$) is equivalent to maximizing the lower bound of the mutual information $I[Y, Z]$ [31]. Thus, in this work and in [19], instead of optimizing Eq. (1) we optimize:

$$\min_{\lambda} \mathcal{L}_{CE} + \beta I[R, Z] \tag{4}$$

In this paper, we provide more support for using cross-entropy instead of $I[Y, Z]$. Given the network with parameter set θ denoted as Φ_θ, we derive the exact representation of $I[Y, Z]$ instead of the upper bound of $I[Y, Z]$, Full derivation can be found in Appendix B.

Proposition 4 *Denoting the neural network function with Φ_θ, we have:*
$I[Y, Z] = \int p(Y, Z) \log \frac{\Phi_\theta(Y|Z)}{p(Y)} dY\, dZ + \mathbb{E}_{Z\sim p(Z)}[D_{KL}[p(Y|Z)||\Phi_\theta(Y|Z)]]$

We prove that the minimizer of the cross-entropy loss is the maximizer of the mutual information $I[Y, Z]$ in the local explanation setting (exact assumption and proof provided in the Appendix C).

Theorem 5 *For local explanation (local neighborhood around the input),*
$\arg\max \int p(Y, Z) \log \frac{\Phi_\theta(Y|Z)}{p(Y)} dY\, dZ = \arg\max I[Y, Z]$

4 Experiments and Results

Over the course of this section, first we provide the experimental setup in Section 4.1. Then, we present qualitative results in Section 4.2, and check the InputIBA's sensitivity to parameter randomization in Section 4.3. We proceed with evaluation of the attribution methods in terms of human-agnostic feature importance metrics (Sensitivity-N [9] in Section 4.4.1, Insertion/Deletion [11], and ROAR [10]). Finally, we evaluate the methods in terms of localization Section 4.5 using

Method	Text (Tokens Divided by Space)
InputIBA	this is easily and clearly the best . it features loads of cameos by big named comedic stars of the age , a solid script , and some great disneyesque songs , and blends them together in a culmination of the best display of henson ' s talent .
IBA	this is easily and clearly the best . it features loads of cameos by big named comedic stars of the age , a solid script , and some great disneyesque songs , and blends them together in a culmination of the best display of henson ' s talent .
LIME	this is easily and clearly the best . it features loads of cameos by big named comedic stars of the age , a solid script , and some great disneyesque songs , and blends them together in a culmination of the best display of henson ' s talent .
Integrated Gradients	this is easily and clearly the best . it features loads of cameos by big named comedic stars of the age , a solid script , and some great disneyesque songs , and blends them together in a culmination of the best display of henson ' s talent .

Table 1: **Qualitative Comparison (IMDB)**: We compare with IBA and methods that are widely used for NLP model interpretaion. To visualize the attribution, we highlight words that have attribution values from 0.33 to 0.66 with orange, and words that have attribution values from 0.66 to 1 with red (attribution value ranges from 0 to 1). We see that IBA does not translate to RNN architectures when attributing to input tokens. IBA can only identify latent important features. We observe that InputIBA and LIME, and IG attribute to relevant tokens.

our proposed EHR metric. To demonstrate the model-agnostic capability and show that our model is revealing the predictive information, we evaluate our method in both vision and NLP domains. We choose ImageNet [32] image classification for evaluation on vision domain. Apart from image classification tasks on ImageNet/VGG-16, we also apply InputIBA on a sentiment classification task (IMDB) and an RNN architecture.

4.1 Experimental Setup

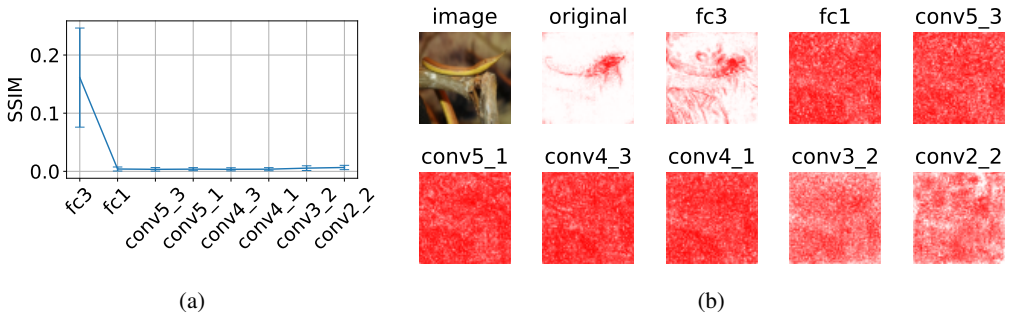
In our experiments on ImageNet, we insert the information bottleneck at layer `conv4_1` of VGG16 [33] pre-trained by `Torchvision` [34]. We use $\beta_{\text{feat}} = 10$ for IBA. We adopt Adam [35] as the optimizer with learning rate of 1.0, and optimize the feature bottleneck for 10 optimization steps. We optimize the generative adversary model for 20 epochs with RMSProp [36] optimizer, setting the learning rate to 5×10^{-5}. For optimizing Λ, we use $\beta_{\text{input}} = 20$ and run 60 iterations. The choice of hyper-parmeters of attribution methods affects their correctness [37]. For more details on the hyper-parameters please refer to Appendix D.

In the NLP experiment, we investigate the sentiment analysis task. A 4-layer LSTM model is trained on IMDB dataset [38] which consists of 50000 movie reviews labeled as either "positive" or "negative". After the model is trained, we generate attributions at embedding space. We compare InputIBA with 4 baselines: Integrated Gradients, LIME [20], IBA [15] and random attribution. In order to generate attribution using IBA, we insert IBA at a hidden layer and consider the hidden attribution as valid for the embedding space, since RNN does not change the size and dimension of input tensor.

We train our model on a single NVIDIA RTX3090 24GB GPU. For VGG16 [33], it takes 24 seconds to obtain the attribution map of one image with resolution 224×224.

4.2 Qualitative Comparison

ImageNet Fig. 2 presents the attribution resulting from various methods. IBA and GradCAM are interpolated into input space, thus they are blurry. Guided Backpropagation is reconstructing image features [13], thus looks interpretable. Extremal Perturbation has lost the shape information of the target object as it applies a smooth mask, moreover it has converged to a region (dog's snout) that is sufficient for the prediction. DeepSHAP and Integrated Gradients generate fine-grained attributions, however, the negative/positive (blue/red) assignment seems random. Moreover, feature



(a) (b)

Figure 3: **Parameter Randomization Sanity Check [12]:** The experiments evaluate whether the attribution changes after randomizing the parameters of neural networks. The randomization starts from FC3 layer and moves towards the image (a) Correlation between original attribution and the attribution after randomization (results averaged for ImageNet subset). We observe that the correlation rapidly moves to 0. The error bars denote ± 1 standard deviation; (b) Attribution map before randomization (label "original") and after randomization the parameters up the specified layer. We observe that attribution is also randomized after parameters are randomized (as opposed to methods such as Guided Backpropagation and LRP (Deep Taylor variant) [12, 14]).

importance experiments (Section 4.4) show the highlighted features are not important for the model. Our proposed method is fine-grained and visually interpretable while highlighting important features (Section 4.4). More qualitative examples (randomly selected) are provided in Appendix E.

IMDB Table 1 presents attribution on a text sample from IMDB. We see that IBA fails to generate a reasonable explanation by assigning high attribution to all words. The observation also shows that IBA is not a model-agnostic method, since IBA assumes the spatial dependency between input and hidden space, which doesn't hold for recurrent neural networks.

4.3 Parameter Randomization Sanity Check [12]

The purpose of this experiment is to check whether the attribution changes if model parameters are randomized. If the attribution remains unchanged, then the attribution method is not explaining model behavior. The experiment progressively randomizes the layers starting from the last layer, and generating an attribution at each randomization step. The generated attribution is compared in terms of similarity with the original attribution.

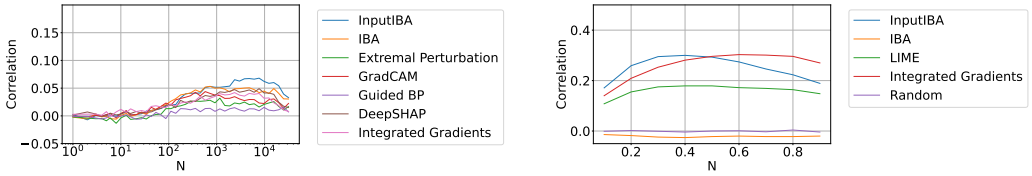
The similarity is measured in terms of structural similarity index metric (SSIM) [39]. In this experiment, we randomly select 1000 samples, and compute the average of the SSIM for all the samples. Fig. 3a, Fig. 3b demonstrate the SSIM error bars and an example for visual inspection respectively. We observe that the InputIBA is sensitive to this randomization (Fig. 3b).

4.4 Feature Importance Evaluation

The following experiments evaluate whether the features identified as important by each attribution method are indeed important for the model.

4.4.1 Sensitivity-N [9]

The experiment randomly masks n pixels and observes the change in the prediction. Subsequently, it measures the correlation between this output change and the sum of attribution values inside the mask. The correlation is computed using Pearson Correlation Coefficient (PCC). The experiment is repeated for various values of n. For each value n we average the result from several samples. For ImageNet, we run this experiment on 1000 images. We randomly sample 200 index sets for each n. In the IMDB dataset, texts have different lengths, thus we cannot perturb a fixed number of words across samples. We address this issue by slightly changing the method to perturb a fixed percentage of words in a text. On ImageNet (Fig. 4) the InputIBA outperforms all baseline methods when we perturb more than 10^2 pixels. When evaluating on IMDB dataset, the InputIBA exhibits a higher



(a) (b)

Figure 4: **Feature Importance - Sensitivity-N**: (a) ImageNet dataset: InputIBA, IBA and GradCAM show high scores in this feature importance metric. We observe that fine-grained (and also axiomatic) methods such as DeepSHAP and Integrated Gradients achieve relatively low scores. (b) IMDB dataset: Both LIME and InputIBA identify important features according to this metric.

correlation under 50% text perturbation rate, which implies the InputIBA is performing better in not assigning attribution to irrelevant tokens.

4.4.2 Insertion/Deletion [11]

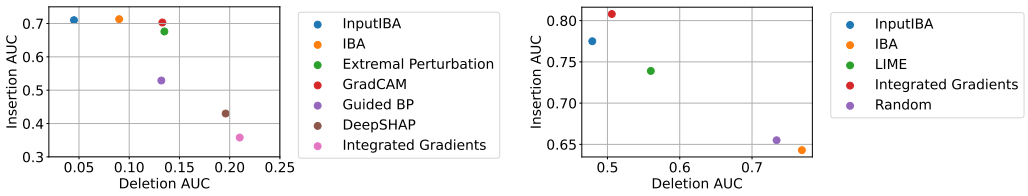
Deletion, successively deletes input elements (pixels/tokens) by replacing them with a baseline value (zero for pixels, <unk> for tokens) according to their attribution score. Insertion, inserts pixels/tokens gradually into a baseline input (image/text). In both experiments we start inserting or deleting the pixel/token with highest attribution. For each image, at each step of Insertion/Deletion we compute the output of the network, and then compute the area under the curve (AUC) value of the output for all steps on a single input. Then we average the AUCs for all inputs (On ImageNet, we average the AUC of 2000 images). For Insertion experiment, higher AUC means that important elements are inserted first. For Deletion experiment, lower AUC means important elements were deleted first. The results are presented in Fig. 5. For the vision task (ImageNet) InputIBA outperforms the rest. Note that the axiomatic methods DeepSHAP and Integrated Gradients achieve low scores in both Deletion/Insertion on ImageNet.

4.4.3 Remove-and-Retrain (ROAR) [10]

One underlying issue with Sensitivity-N and Insertion/Deletion experiments is that the output change may be the result of model not having seen the data during the training. Therefore, ROAR [10] retrains the model on the perturbed dataset. The more the accuracy drops the more important the perturbed features are. For each attribution method, the perturbation of the dataset is done from the most important elements (according to attribution values) to the least important element. As retraining the model is required for each attribution method at each perturbation step, the experiment is computationally expensive. Therefore, we run this experiment on CIFAR10. The results are presented in Fig. 6. We can observe two groups of methods. For the first group (DeepSHAP, Integrated Gradients and Guided Backpropagation) the accuracy does not drop until 70% perturbation, meaning if 70% of pixels are removed, the model can still have the original performance. For the rest of the methods, we see that the features are indeed important. GradCAM and Extremal perturbations are performing slightly better than IBA and InputIBA. We suspect that this is due to their smooth attribution maps. We test this hypothesis by applying smoothing on InputIBA (InputIBA*) and we observe that the performance becomes similar to the other two. The key observation is that these four methods are all successful in identifying important features.

4.5 Quantitative Visual Evaluation via Effective Heat Ratios (EHR)

We quantify the visual alignment between attribution maps and ground truth bounding boxes (on ImageNet [32]). This serves as a metric for *visual/human* interpretability, and is a *measure of fine-grainedness*. Previously [19], the proportion of top-n scored pixels located within the bounding box was computed. We argue that this method only considers the ranks of the pixels, but ignores the distribution of attribution outside the bounding box. We illustrate the limitation of the previous metric with synthetic examples in Appendix F. Instead, we suggest to vary the value of n to consider the distribution of the attribution. To this end, we propose EHR, where we consider multiple quantile thresholds from 0.0 to 1.0. At each quantile threshold, we compute the sum of attribution within



(a) (b)

Figure 5: **Feature Importance - Insertion/Deletion**: The closer the method is to the top-left the better (a) **ImageNet** dataset: InputIBA, IBA, Extremal Perturbations and GradCAM all reveal important features according to this metric. DeepSHAP and Integrated Gradients to do not idenfity important features. (b) **IMDB** dataset: InputIBA reveals important features according this metric. The results are consistent with Sensitivity-N.

Method	EHR
InputIBA	**0.476** $\pm$ 0.007
IBA	0.356 $\pm$ 0.005
GradCAM	0.283 $\pm$ 0.005
Guided BP	0.421 $\pm$ 0.005
Extremal Perturbation	0.421 $\pm$ 0.007
DeepSHAP	0.183 $\pm$ 0.002
Integrated Gradients	0.155 $\pm$ 0.002

Table 2: **Quantitative Visual Evaluation - EHR**: This metric evaluates how precisely the attributions localize the features by comparing them with ground truth bounding boxes. InputIBA, extremal perturbations, and Guided Backpropagation score highest. IBA and GradCAM also perform well, but due to their lower resolution maps, they receive lower scores. Standard error is also presented in the table.

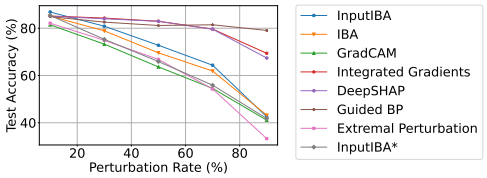


Figure 6: **Feature Importance - ROAR**: Integrated Gradients, DeepSHAP, and Guided Backpropagation are not identifying contributing features. On the other hand, InputIBA, IBA, Extremal Perturbations, and GradCAM point to important features. The latter two methods are performing slightly better, which is due to their attributions being smooth (covering more areas). We apply smoothing to InputIBA (InputIBA*) and achieve the same result.

the bounding box divided by the total number of pixels above this threshold, which is defined as the *effective heat ratio*. Finally, we compute the AUC of the ratios over the quantiles. We assess InputIBA along with other baselines on 1000 images with bounding box covering less than 33% of the whole image. Table 2 illustrates the results.

4.6 Discussion

Societal Impact Although our work is a step towards solving the attribution problem, the problem remains open. Therefore the interpretation tools (attribution methods) must be used with caution. The machine learning community is using these tools to interpret their results and is deriving conclusions. It is not clear if the findings would also show up with another interpretation tool. Wrong interpretations of results arising from the attribution tools can have a destructive effect in mission-critical applications such as medical imaging.

The community must be aware of the shortcomings of each interpretation tool. For example, [13, 14] prove several methods are not explaining the behavior. In our work, we also observe that two Shapley value-based methods with solid mathematical and axiomatic grounding, are not identifying important features on computer vision datasets according to all three feature importance evaluations (this also verified for Integrated Gradients in [10]). The Shapley value is hailed as the ultimate solution by many research works, our observation is telling another story. Therefore, our proposed method (and IBA [19]) should also be used with caution, even though they are grounded on theory and are performing well in current metrics.

Limitations Although our core idea – finding a bottleneck on input that corresponds to predictive deep features (described in Section 3.3) – is simple, the entire methodology for computing the

predictive information of input features is relatively complex (compared to a method such as CAM [26, 18]). This may be an impediment to the method's adoption by the community (however, we provide a simple interface in our code).

Another issue is that our idea adds an additional optimization term (Eq. (3)) to IBA, which increases the time complexity. Therefore, our method is introducing a trade-off between improved performance and speed. Nonetheless, for many interpretation applications, speed may not be an issue.

5 Conclusion

In this work, we propose a methodology to identify the *input* features with high predictive information for the neural network. The method is agnostic to network architecture and enables fine-grained attribution as the mutual information is directly computed in the input domain. The improved interpretation technique benefits the general machine learning community. Furthermore, it introduces a new idea – input bottleneck/mask estimation *using deep layers* – to the attribution research community to build upon and move one step closer to solving the attribution problem.

Acknowledgement Ashkan Khakzar and Azade Farshad are supported by Munich Center for Machine Learning (MCML) with funding from the Bundesministerium fur Bildung und Forschung (BMBF) under the project 01IS18036B. Yawei Li is supported by the German Federal Ministry of Health (2520DAT920). Seong Tae Kim is supported by the IITP Grant funded by the Korea Government (MSIT) (Artificial Intelligence Innovation Hub) under Grant 2021-0-02068.

References

[1] Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

[2] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017.

[3] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *37th International Conference on Machine Learning, ICML*, 2020.

[4] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[5] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.

[6] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning, ICML*, 2017.

[7] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 2017.

[8] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018.

[9] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *ICLR*, 2018.

[10] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9737–9748, 2019.

[11] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

[12] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018.

[13] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *International Conference on Machine Learning*, 2018.

[14] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. *International Conference on Machine Learning*, pages 9046–9057, 2020.

[15] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020.

[16] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, pages 3429–3437, 2017.

[17] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, 2019.

[18] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017.

[19] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020.

[20] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 1135–1144, 2016.

[21] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[22] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.

[23] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[24] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. *CVPR Workshop*, 2019.

[25] Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pages 537–543, 2017.

[26] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.

[27] Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. Neural response interpretation through the lens of critical pathways. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13528–13538, 2021.

[28] Ashkan Khakzar, Soroosh Baselizadeh, and Nassir Navab. Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods. *arXiv preprint arXiv:2012.00362*, 2020.

[29] Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Seong Tae Kim, and Nassir Navab. Explaining neural networks via perturbing important learned features. *arXiv preprint arXiv:1911.11081*, 2, 2019.

[30] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[31] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *ICLR*, 2017.

[32] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *ACM International Conference on Multimedia*, page 1485–1488, 2010.

[35] Adam: A method for stochastic optimization. *ICLR*, 2015.

[36] T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[37] Naman Bansal, Chirag Agarwal, and Anh Nguyen. Sam: The sensitivity of attribution methods to hyperparameters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[38] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. *ACL*, 2011.

[39] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[40] Alex Alemi, Ian Fischer, Josh Dillon, and Kevin Murphy. Deep variational information bottleneck. In *ICLR*, 2017.

7

Through the Lens of the Model

This chapter deals with the problem of evaluating feature attribution methods. How do we claim that a feature attribution method attributes to the features relevant to the prediction? We need to first clarify what the relevance of features means. It is a repeated theme in this dissertation that the effect of the removal of features on the prediction is an intuitive way (and not the only way) to define relevance. If removing a feature affects the prediction, it is relevant for the prediction. As a matter of fact, several evaluation methodologies are formulated based on this notion. The method introduced in [52] removes features sequentially based on the relevance score assigned by the attribution method under evaluation. Therefore we can observe the neural network's output change during the process of removing features and can relatively evaluate attribution methodologies against each other and against a random removal of features. The removal analysis can also take other forms, such as sensitivity-N [39], which is explained in previous sections.

A hidden issue lurking within such an analysis is that the output change resulting from removing a feature (or a set of features) might be due to the newly generated input being out of distribution. Thus [75] proposes a similar analysis but involving retraining. After removing features based on their relevance scores, we retrain the neural network from scratch on the new perturbed dataset. E.g., we can remove the top twenty percent of the important pixels (as identified by the feature attribution method) in all images in the training dataset. The retraining solution is proposed to alleviate the out-of-distribution problem. However, the new evaluation scheme will not show the feature's immediate effect on the network's output. In my view, this removal and retraining perturbation schemes are, in essence, evaluating different aspects, although being ground upon the same concept. Simply stated, the former is assessing how significant the effect of the absence of a feature is on the output, being relevant, out of distribution, or adversarial. The evaluation still gives us information in any of these cases, specifically if the effect differs from a random removal of features. The latter (retraining) gives a different perspective. It shows if a neural network can generalize or not, using a set of relevant (as identified by the attribution method) features kept or removed from the dataset. In short, [39], [52], [75] evaluate different aspects: effect on output and generalizability of selected features.

There are other perspectives through which we can evaluate feature attribution methods. Specific properties are (un)desirable for feature attributions. Assume we have an image input with two classes, and the neural network predicts the two classes correctly. This observation raises suspicion if the feature attribution method generates the same saliency maps for these two output classes. It is more probable that the network is using different features to predict two different classes. This behavior is observed in several feature attribution methods [31], [60], [79], [84]. It is interesting that [63] hypothesized this phenomenon would exist for one of the most popular attribution methods of the time, and they confirmed the observation through experiments. Subsequently, [95], [98] proposed more systematic experiments to quantitatively assess the class-sensitivity property of attribution. We also hypothesized that this behavior exists among several state-of-the-art attribution methods of the time and demonstrated that these attributions generate the same attributions for different classes [93]. Some methods, in fact, generate the same saliency for all output classes. Evaluation by looking for a desirable property got significant attention after the sanity checks proposed in [58]. The sanity checks evaluate if the feature attribution is sensitive to network parameter randomization. The experiment investigates whether the feature attribution changes when we replace the network weights with random values. If the attribution method generates the same saliency map after randomization, then it is probably not taking the network into account. Specifically, if the attribution is generating a saliency map visually similar to the image before and after randomization, then it is probably using the input itself to generate the saliency [63].

The desirable properties for feature attribution methods can also be formulated by axioms. For instance (based on a feature removal perspective of relevance), if a feature's removal does not affect the output of the neural network in any combination with other input features, then that feature does not contribute. An attribution method is therefore expected to not assign any relevance to this feature. This axiom is known as the null-player or dummy axiom [100]. We may be able to show mathematically that a feature attribution method is (not) conforming with an axiom. However, due to the proposed methods' complexity and the approximations involved within their implementations, it is not trivial to prove that an attribution conforms with an axiom. For instance, recent work [100] shows that many attributions proposed to approximate the Shapley value break the axioms. And the desirability of the Shapley value is due to the fact that it is the unique method that satisfies several axioms [46], [100].

This chapter proposes an experimental setup for evaluating feature attribution methods against axioms. Our experimental design, in principle, does not prove if an attribution method satisfies an axiom but can reveal examples of methods breaking axioms. The core idea behind our evaluation strategy is that we generate input features with the neural network in the loop, such that we know the effect on the network's output. For instance, we can create an image feature such that it is a null feature

for the neural network's output. We generate input features through optimization in the input space by adopting tools provided in feature visualization literature [41], [48], [65]. We add other optimization terms to impose the desired behaviors (e.g., a feature being null) on the generated features. To balance the optimization terms, we use dynamic task prioritization for multitask learning [61]. Note that we are using the neural network itself in the loop; otherwise, we cannot guarantee that a feature would have the behavior for the network. For instance, one may argue that we can add random value pixels to an image, and since they are random values, they will have no information for the neural network, and we can check if an attribution method is pointing at them. But the random value pixels added to the image might change the neural network's output (e.g., due to out-of-distribution effects). Therefore there is no guarantee that they are null features (their removal/addition can change the output). We are generating features using the model and can guarantee through optimization terms that they have the desired property. We use this controlled environment to test attribution methods against different axioms.

Though the idea is incarnated as an experimental setup for testing models against axioms, the core idea is: How do we know if an explanation is explaining the model? The model itself knows best. The main idea is to extract the knowledge within the model, then compare explanations to the extracted knowledge.

Do Explanations Explain? Model Knows Best

Ashkan Khakzar[1,*], Pedram Khorsandi[1,*], Rozhin Nobahari[2,*], Nassir Navab[1]

[1] Technical University of Munich
[2] Mila, Quebec Artificial Intelligence Institute, Canada
[*] Equal contribution

It is the accepted but not the published version of the paper due to copyright restrictions.

Published in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Published version: `https://ieeexplore.ieee.org/document/9878403`

Do Explanations Explain? Model Knows Best

Ashkan Khakzar[1*], Pedram Khorsandi[1*], Rozhin Nobahari[2*], Nassir Navab[1]
ashkan.khakzar@tum.de

[1] Technical University of Munich, Germany
[2] Mila, Quebec Artificial Intelligence Institute, Canada

Abstract

It is a mystery which input features contribute to a neural network's output. Various explanation (feature attribution) methods are proposed in the literature to shed light on the problem. One peculiar observation is that these explanations (attributions) point to different features as being important. The phenomenon raises the question, which explanation to trust? We propose a framework for evaluating the explanations using the neural network model itself. The framework leverages the network to generate input features that impose a particular behavior on the output. Using the generated features, we devise controlled experimental setups to evaluate whether an explanation method conforms to an axiom. Thus we propose an empirical framework for axiomatic evaluation of explanation methods. We evaluate well-known and promising explanation solutions using the proposed framework. The framework provides a toolset to reveal properties and drawbacks within existing and future explanation solutions.[1]

1. Introduction

Considering a neural network function, how do we know which features (patterns) within the input are important for its output? The problem is called feature attribution [16, 35], and the solutions are commonly known as explanation, attribution, or saliency methods. There is an extensive list of explanation methods in the literature [8,9,13,15,16,19,26,27,29,32,33,35,39,41]. One peculiar observation is that these solutions point to different features as being important. Though they are solutions to the same problem, feature attribution, the resulting explanations are curiously dissimilar. The phenomenon raises the question, which explanation is correct? Or are the explanations correct but revealing the problem in a different light?

One approach is to compare the explanations against ground truth (e.g., bounding box) annotations on the dataset [27,38,41]. But how do we know what is important for a human is also important for the model? There is no guarantee (or reason) that the model would use the same features as humans. To resolve this issue, we need to take a step back and ask what it means for a feature to be "important" for an output. The intuitive approach is to remove the feature and observe the output behavior [8, 11, 25]. Such removal of evidence is indeed the foundation of many explanation approaches [8, 9, 16, 26, 35]. However, such a conception could lead to ambiguities. Consider the scenario of having equivalent features (e.g., repeated features), where the existence of each feature alone suffices for a specific output value. Add to the scenario that the removal of any of these features does not affect the output value. In this case, the conception based on removal assigns zero importance to each feature. However, a desirable property, in this case, could be assigning equal importance to each feature.

The concept of importance can thus be further chiseled by specifying *desirable properties* that an importance assignment method ought to satisfy. Such *desirable properties are formalized via axioms* [16, 34, 35]. The axiomatic view provides a complementary framework for evaluating feature attribution solutions. Explanation methods can be evaluated whether they conform to an axiom. The axiomatic view has the advantage that the methods can be mathematically proven to comply with a particular axiom. For instance, solutions such as the Shapley value [16, 28] and integrated gradients [34, 35] are proven to conform with particular axioms. However, proofs can be broken in practical implementations. For instance, [34] show that inherent assumptions within methods that approximate the Shapley value result in methods not conforming with the axioms. Moreover, certain conditions might be overlooked in proofs. Thus *experiments are required to test whether final solutions comply with the axioms*. Even if methods are accompanied by elegant and solid mathematical derivations and proofs, they must comply with the axioms in observations

*denotes equal contribution
[1] https://github.com/CAMP-eXplain-AI/Do-Explanations-Explain

in designed experiments. If they do not comply with axioms in experiments, we may revisit our assumptions and methodologies. Such is the way of the scientific method.

This work lays out an experimental framework for evaluating attribution solutions axiomatically. We set up each experiment such that the solution can be tested whether it complies with a specific axiom. We generate input features that impose a particular behavior on the network's input/output relationship. Features are generated via optimization on the input space while the network parameters are kept constant. Using optimization, we can impose the desired relationship between the generated input and the output. We can thus engineer setups to evaluate axioms. For instance, one axiom that attribution methods are required to conform to is the Null-player axiom. The null-player axiom requires the following; If removal of a feature in all possible coalitions with other features does not affect the output, it should be assigned zero importance. With our proposed framework, we can generate a null player feature for the neural network function. Subsequently, we can test different feature attributions solutions and check whether they assign importance to the null player feature. Thus we can test whether a solution conforms to the Null-player axiom. We also devise experiments to evaluate the explanations in terms of other desirable properties; The class-sensitivity and the feature-saturation. With our framework, we evaluate well-known and recently introduced promising solutions. With our experiments, we intend to reveal properties and drawbacks within existing explanations.

2. Background and Related Work

2.1. Background

We first introduce the feature attribution literature as our framework is designed to evaluate these methods. Then we introduce feature visualization/generation methods since they can be used within our framework.

2.1.1 Explaining Predictions via Feature Attribution

The feature attribution problem is concerned with identifying the input features that contribute to the output value. The solutions can be roughly categorized as follows (some solutions belong to multiple categories).

Backpropagation [4, 30] linearly approximate the network and propose the gradient as attribution. Deconvolution [37], GuidedBackProp [32] backpropagate a modified gradient. Integrated Gradients [35] distributes the change in output with respect to a baseline input by integrating gradients between the two input states. LRP [19], DeepLIFT [29] bacpropagate contribution layer-wise. The contribution notion in LRP and DeepLIFT is also grounded on *removal*.

Perturbation/Removal Methods in this category are explicitly grounded on the removal of features. They mask/perturb input features and observe the output change [8, 9, 18, 23]. E.g., Extremal Perturbations [8] searches for the smallest region in the input such that the keep the region preserves the target prediction. [37] propose occluding pixels or a patch of pixels and measure the output change. IBA [26] inserts an information bottleneck by removing hidden features (via replacing them with noise) and keeps the smallest region that preserves the predictive information. InputIBA [39, 40] enables inserting the information bottleneck on the input.

Latent Features CAM/GradCAM [27, 41] leverage activation values (aka network's attention) of convolutional layers. GradCAM++ uses different summation rules on layers and is applicable to all layers. IBA [26] also utilizes latent features. FullGrad leverages the activation, gradient and, bias values from all layers. PathwayGrad [13] leverages critical pathways (pathway important neurons).

Game Theory The attribution problem can be considered as credit assignment in cooperative game theory. This is achieved by presuming the network's function is a score function, and input features are players. A solution to this problem that satisfies several axioms is the Shapley Value. This notion is also grounded upon the removal of players and the effect of removal on score function. Shapley Value considers the removal of a player in all possible coalitions. Due to computational complexity, several approximations are proposed for neural networks. DeepSHAP [16] backpropagates SHAP values via DeepLift [29] framework. It is recently shown [34] that Integrated Gradients [35] approximates Shapley value in continuous setting.

2.1.2 Generating Features that Activate a Neuron

These works identify what input patterns/features activate a neuron and are commonly referred to as feature visualization. In essence, the methods generate images that maximize certain neuron activations [7, 17, 20, 22, 30, 36]. This can be achieved by performing the optimization on the image while freezing networks parameters. We can use any of these solutions within our framework. We opt for deep image prior [36] (refer to Sec. 3.5). Another method that we utilize within our framework is the adversarial patch [5].

2.2. Related Work

This section introduces the works that evaluate explanations. Our framework belongs to both "ground truth" and "axiomatic" categories. We discuss the differences to existing works in each section separately (more in appendix).

Do Explanations Explain? An early work [21] demonstrates that Deconvolution [37] and Guided Backpropagation [32] are reconstructing image features rather than explaining the prediction. Thus an explanation can be visually interpretable but not really be an explanation. The works in this section investigate whether an explanation method is indeed explaining the prediction and whether it can be trusted. Each work evaluates an explanation from a different vantage point. We categorize the works as follows:

Perturbation/Removal The objective of these works is to evaluate whether features identified as salient by attribution methods are indeed contributing to the output. The intuition behind them is that if the identified features are important, perturbing (removing) them changes the output relatively more. Sensitivity-N [2] and [25] use various perturbation schemes on the input and observe the output change. Remove-and-Retrain [11] perturbs the input, then retrains the model and measures the accuracy drop.

Ground Truth These works compare the explanations with a ground truth of features that are important. Pointing game [38] and classic localization-based metrics [27] use annotations on natural images done by humans. However, here there is an underlying assumption that the model is using the same features as humans, which is a crude assumption. To solve this issue, CLEVR XAI [3] proposes generating a synthetic dataset using CLEVR [12]. The model is then trained on the generated dataset, and then explanations are compared with the ground truth. The approach adds partial control. However, there is no guarantee that the model picks up the intended features in the generated dataset. In our framework we can control what features contribute or do not contribute to the model's output.

Axiomatic Axiomatic approaches check whether the model complies with particularly desirable properties. Evaluation can be either theoretical, where the method is proven to satisfy an axiom (or a desirable property), [16, 31, 34, 35], or experimental. Sanity checks [1] experimentally check whether randomizing network parameters change the explanation. Another desirable property is class sensitivity, i.e., the explanation should not be the same for different outputs (classes) if their contributing features differ. [14] provides a reasoning on why several methods are insensitive to parameter randomization and different classes. [21, 24] propose experiments to evaluate class-sensitivity on natural image datasets. However, on natural images, there is no guarantee that the model uses different features for different classes. Our framework provides a controlled setup as features are generated.

3. Methodology

The objective is to have a controlled experimental environment in which we control what features contribute or do not contribute to the output of the neural network function. In this environment, we can devise scenarios to test the explanations against axioms. To this end, we leverage the model itself and run an optimization on the input, thus controlling how features contribute to the output.

Importance or contribution is understood only with respect to a reference/baseline state ("removal" is setting feature values to a reference value). In our setup, we compute the contribution of a feature with respect to a reference of normal random noise. X denotes the reference input. We refer to a group of pixels and their specific values as a "feature". In this work, we select a patch of pixels to form the feature. We denote the patch/feature by f and a baseline input that has a feature f added to it by $X_{\{f\}}$ and the neural network function for a target output t by $\Phi_t(.)$. To generate the feature f that corresponds to a target t we generate a patch on the baseline input X that activates the target t. Since the added feature changes the output value (by design), according to sensitivity axiom [35] it is guaranteed that it contributes to the output. The optimization is performed only on patch f (not on other areas of input X). The optimization loss for generating feature f corresponding to target t is denoted by $\mathcal{L}_f^t$. Depending on the scenario, $\mathcal{L}_f^t$ can be associated with either of the following. We can either generate a patch that maximizes the target value, $\min_f \ -\Phi_t(X_{\{f\}})$ or generate a patch that achieves a constant target value c, $\min_f \ \mathcal{L}_{CE}(\Phi_t(X_{\{f\}}), c)$ where $\mathcal{L}_{CE}$ denotes cross-entropy loss.

3.1. Null Feature

The objective in this section is to devise a setup for testing the null feature axiom. A null feature is one that does not contribute to the output score. If a feature is a null feature, it is a desirable property for the explanation not to assign any contribution to that feature. based on cooperative game theory and attribution literature, null feature can be formally defined as follows. Having a group of features (players), a feature is a null feature if its absence does not affect the output score function in all possible coalitions of features. I.e. if we have a set of n features $\{f_1, ..., f_n\}$, a feature f_i is null for output $\Phi_t(.)$ if $\Phi_t(X_{\{f_i \cup \mathcal{S}\}}) = \Phi_t(X_{\{\mathcal{S}\}})$, where $\mathcal{S}$ denotes all subsets of features excluding f_i, i.e. $\mathcal{S} \subset \{f_1, ..., f_n\} \setminus \{f_i\}$. Note that are 2^{N-1} possible coalitions.

In our experimental setup, we add two features to the baseline input X (one can add more features and devise more complex or creative experiments). We add feature f_a that corresponds to output $\Phi_a(.)$ and add another feature f_{null} that corresponds to an output $\Phi_b(.)$ but is a null feature

for output $\Phi_a(.)$. In order for the f_{null} to be a null feature, its absence in all possible coalitions with f_a should have no effect on output $\Phi_a(.)$. The are two possible coalitions, which are the subsets of f_a, namely f_a and . Therefore, the output $\Phi_a(.)$ must stay constant when f_{null} is removed when f_a exists in baseline input X. The output $\Phi_a(.)$ must also stay constant when f_{null} is added to the baseline X. Therefore, the optimization problem is defined as the following two concurrent optimizations,

$$\min_{f_a} \mathcal{L}^a_{f_a} \tag{1}$$

$$\begin{aligned}\min_{f_{null}} \mathcal{L}^b_{f_{null}} + (\Phi_a(X_{\{f_a,f_{null}\}}) - \Phi_a(X_{\{f_a\}}))^2 \\ +(\Phi_a(X_{\{f_{null}\}}) - \Phi_a(X_{\{\}}))^2\end{aligned} \tag{2}$$

where $\mathcal{L}^a_{f_a}$ generates feature f_a corresponding to output $\Phi_a(.)$. In Eq. (2) $\mathcal{L}^b_{f_{null}}$ generates a feature f_{null} corresponding to output $\Phi_b(.)$. The second and third term in Eq. (2) try to make f_{null} be a null feature for $\Phi_a(.)$ by removing it in possible coalitions with f_a. The result of the optimization is $X_{\{f_a,f_{null}\}}$, which is a baseline noise image X that contains patches/features f_a and f_{null}. In this setup, we aim to test whether an explanation method attributes the output $\Phi_a(X_{\{f_a,f_{null}\}})$ to the null feature. The proposed metric for evaluation is provided in Sec. 3.4

3.2. Class Sensitivity

Another property that is expected from an explanation method is class sensitivity, i.e. output sensitivity. Considering two outputs $\Phi_a(.)$ and $\Phi_b(.)$ of a neural network, if the contributing input features to the these outputs differ, the explanations for the outputs should also be different. To test such property we devise two scenarios:

3.2.1 Single Feature Scenario

In our first proposed setup, we only add one feature f_a to the reference input X. The feature is generated such that it corresponds to the output $\Phi_a(.)$ but is a null feature for another output $\Phi_a(.)$. Therefore,

$$\min_{f_a} \mathcal{L}^a_{f_a} + (\Phi_b(X_{\{f_a\}}) - \Phi_b(X_{\{\}}))^2 \tag{3}$$

where the first term $\mathcal{L}^a_{f_a}$ generates the patch f_a on reference input X, and the second term makes sure it is a null feature for output $\Phi_b(.)$. I.e. the removal of feature f_a should not affect the output $\Phi_b(.)$.

In this setup, the explanations for the two outputs $\Phi_a(.)$ and $\Phi_b(.)$ are compared. It is expected that the first explanation (for $\Phi_a(.)$) attributes the output (partly) to f_a. Whereas, the second expalanation (for $\Phi_b(.)$) should not attribute the prediction of $\Phi_b(.)$ to the the feature f_a. Our proposed metric for evaluating this effect is provided in Sec. 3.4.

3.2.2 Double Feature Scenario

In this setup we add two features f_a and f_b to the reference input X, each corresponding to the different outputs $\Phi_a(.)$ and $\Phi_b(.)$ respectively. In this setup the dominantly contributing feature to $\Phi_a(.)$ is feature f_a and the dominantly contributing feature to $\Phi_b(.)$ is f_b. Therefore we perform two concurrent optimizations. The first one,

$$\min_{f_a} \mathcal{L}^a_{f_a} + (\Phi_b(X_{\{f_a,f_b\}}) - \Phi_b(X_{\{f_b\}}))^2 \tag{4}$$

generates f_a which contributes to output $\Phi_a(.)$ but its removal in the presence of feature f_b does not affect output $\Phi_b(.)$. The second optimization,

$$\min_{f_b} \mathcal{L}^b_{f_b} + (\Phi_a(X_{\{f_a,f_b\}}) - \Phi_a(X_{\{f_a\}}))^2 \tag{5}$$

generates f_b which contributes to output $\Phi_b(.)$ but its removal in the presence of feature f_a does not affect output $\Phi_a(.)$. Thus the dominantly contributing feature for $\Phi_a(.)$ is f_a and for $\Phi_b(.)$ is f_b.

In this scenario we expect the explanations to switch from feature f_a to f_b when the output to be explained is changed from $\Phi_a(.)$ to $\Phi_b(.)$. Our proposed metric for capturing this metric is provided in section Sec. 3.4.

3.3. Feature Saturation

In this section, we devise a scenario where features saturate the output. Such that the features f_{a1} and f_{a2} together (i.e. $X_{\{f_{a1},f_{a2}\}}$) result in the same output value as when the features are added to reference input X individually. To achieve this, we solve two optimizations concurrently,

$$\min_{f_{a1}} \mathcal{L}^a_{f_{a1}} + (\Phi_a(X_{\{f_{a1},f_{a2}\}}) - \Phi_a(X_{\{f_{a2}\}}))^2 \tag{6}$$

where the first term generates f_{a1} such that the output is equal to a constant value c. The second term makes sure that feature f_{a1} removal from input does not affect the output when f_{a2} is present. The second optimization does this procedure on the second feature f_{a2},

$$\min_{f_{a2}} \mathcal{L}^a_{f_{a2}} + (\Phi_a(X_{\{f_{a1},f_{a2}\}}) - \Phi_a(X_{\{f_{a1}\}}))^2 \tag{7}$$

In this setup, the existence of one of the features is sufficient for the prediction. As they contribute equally to the output, an explanation solution is expected to attribute the output equally to both features. Our proposed metric for evaluating this property is provided in Sec. 3.4.

3.4. Metrics

In this section, we introduce our metrics for evaluating the properties in each of the generated setups. We denote an explanation generated for target output $\Phi_t(.)$ by S_t.

Null Feature Metric Defined as contributions assigned to null feature relative to total assigned contributions:

$$\frac{\sum_{f_a} S_a}{\sum_{S_a} S_a} \tag{8}$$

The sum operator $\sum_f S_t$ runs over all corresponding pixels in S_t that are in patch f.

Class Sensitivity Metric In the Double Feature Scenario, we measure the class sensitivity by:

$$\frac{\sum_{f_a \cup f_b} min(S_a, S_b)}{\sum_{f_a} S_a + \sum_{f_b} S_b} \tag{9}$$

where the $min(S_a, S_b)$ is the pixel-wise minimum of S_a and S_b. In an extreme case, for the explanation method being indifferent towards the target class, the $min(S_a, S_b)$ would be equal to S_a and S_b. Therefore, the metric evaluates to one. In the other extreme, where attributions shift from f_a to f_b, the $min(S_a, S_b)$ and the metric is zero.

For the Single Feature Scenario, the class sensitivity is:

$$corr(\frac{\overline{S_a - S_a \backslash f_a}}{\overline{S_a \backslash f_a}}, \frac{\overline{S_b - S_b \backslash f_a}}{\overline{S_b \backslash f_a}}) \tag{10}$$

The term $\frac{\overline{S_t - S_t \backslash f}}{\overline{S_t \backslash f}}$ determines the average amount of contribution score inside patch f, devided by the average outside the patch. The higher correlation implies the method is attributing to the same feature for both outputs Φ_a and Φ_b.

Feature Saturation Metric To evaluate how the attribution is distributed between the features, we evaluate the correlation between attributions assigned to feature f_{a1} and f_{a1}

$$corr(\sum_{f_{a1}} S_a, \sum_{f_{a2}} S_a) \tag{11}$$

A method that assigns the attribution to only one feature receives a lower score.

3.5. Implementation Details

Reference/Baseline Input: Importance is understood with respect to a reference state. The reference is chosen such that it represents the missingness of features. In the vision domain, it is customary to use zero value [29, 35], or noise [26]. In any case, our framework is not dependent on the reference. We do not make any assumptions about what features are in the reference. We ensure that a feature is null with respect to the reference, and our metric only considers the generated features and not the background (we use attributions in background only for scaling).

Deep Prior Network: If we perform the optimization on the patches without any regularization, it is easy to get trapped in local solutions. In this case, we may not achieve a satisfactory optimization solution for Eq. 1-7. In our work, we leverage "deep image prior" [36] to limit the space of solutions and avoid trivial local solutions. Using deep image prior methodology, we add a decoder network with random weights and a random seed input behind the generated patch. In other words, the patch is parameterized by the prior network. The optimizations are thus done on the parameters of the prior network instead of the patch. In [36] it is also demonstrated that the untrained network does capture some of the low-level statistics of natural images. Therefore the generated patches also look interpretable to us. Visual interpretability is not required within this framework, though it can make the experiments more intuitive.

The Model: The choice of the model does not affect the framework as long as the optimizations are solvable. The network is pre-trained on ImageNet [6]. However, the proposed framework does not depend on the network being trained. For a random network the generated features would not "look" interpretable.

Optimization During optimization steps, we place the patch in different locations to ensure that the results do not depend on the patch's location. Moreover, in order to balance the terms in Eq. 1-7 we use focal loss [10] (appendix).

4. Results and Discussion

The objective of our proposed framework is to reveal insights and shortcomings regarding explanation methods. We evaluate various explanation methods from different categories. DeepSHAP and IntegratedGradient are theoretically axiomatic methods. GradCAM and GradCAM++ are two popular methods that leverage network attention. We also evaluate the recently introduced FullGrad from this family. In addition, we evaluate two recent promising solutions, IBA and Extremal Perturbations.

4.1. Null Feature

The null feature experiment checks whether an explanation attributes the output to a null feature. I.e., it checks whether the explanation method identifies the null feature as important. The framework guarantees that the null feature is not contributing. Using the framework, we generate 1000 inputs. For each input sample, the feature is generated for a random output. We then proceed and compute the null feature metric on each generated inputs and report the average in Tab. 1. An example generated input is in Fig. 1.

FullGrad, DeepSHAP, Gradient, and GuidedBackProp perform the worst in this experiment. This performance may point to the fact that these methods identify all features within the input as important. It is previously shown [21] that GuidedBackProp reconstructs image features rather than explaining the prediction, and our results are aligned

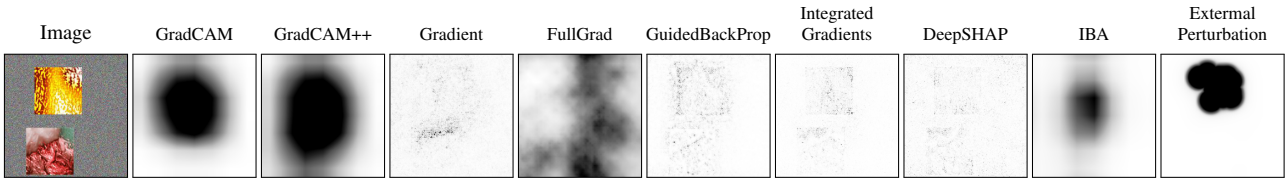


Figure 1. **Null Feature Experiment**: The image on the left represents the generated features on the reference (noise) input. The features are generated using the model itself. Within the image, the lower feature (patch) is generated such that it is a null feature for the output. The rest of the images represent different explanations. As the second feature is a null feature, an explanation method should not assign importance to it. We observe that GradCAM, IBA, and Extremal Perturbation perform best in avoiding the null feature.

Method	Null Feature	Class Sensitivity		Feature Saturation
		Double Feature Scenario	Single Feature Scenario	
GradCAM	0.135	0.176	0.050	0.243
GradCAM++	0.452	0.469	0.845	-0.571
Gradient	0.835	0.469	0.684	0.310
FullGrad	1.00	0.931	0.951	-0.130
GuidedBackProp	0.704	0.555	0.979	0.703
IntegretedGradient	0.534	0.344	0.759	0.212
DeepSHAP	1.03	0.507	0.934	0.221
IBA	0.211	0.191	0.295	-0.223
ExtermalPerturbation	0.047	0.039	0.759	-0.680

Table 1. Evaluation of Explanations with the Framework: **1) Null Feature**: Null feature experiment evaluates the extent to which each explanation attributes the output to a null feature. In this metric, the less the value, the better. Extremal Perturbation [], GradCAM, and IBA are the favorable methods from this null feature perspective. **2) Class Sensitivity**: For both experiments, the lower the value, the better 1) Double Feature Scenario: In the case where two features corresponding to two different classes are present, Extremal Perturbation, IBA, and GradCAM attribute to the correct feature when applied to the two outputs. 2) Single Feature Scenario: In the case where only feature is present, explanations for two different outputs are similar to all methods except GradCAM and, to some extent, IBA. **3) Feature Saturation**: the experiment evaluates how explanations distribute the importance between saturated features. In this metric, the higher the value, the better. The notable observation is Extremal Perturbation, as it identifies only one of the features as important.

with the finding. It can also be inferred that gradient is also sensitive to all features in the input. DeepSHAP is widely known as a solid method as it involves SHAP. However, it also has a backpropagation mechanism (as it is engineered on DeepLift). It seems the backpropagation is the culprit, as other gradient methods also fail this experiment. FullGrad does a weighted sum of gradients and biases of all layers. The gradients in the early layers can be the culprit in this case. We observe that GradCAM rarely assigns attribution to the null feature. And the assigned values may be due to CAM's low resolution. IBA and extremal perturbation are both grounded on the removal of features. We see that they also avoid attributing to the null feature.

We also evaluate IBA and GradCAM++ on different layers of a ResNet network. Among the advantages of these methods is that they can be applied to early layers to produce higher resolution maps. However, we observe in Fig. 2 and Tab. 2 that as we move towards early layers, the methods attribute to the null feature.

4.2. Class Sensitivity

Double Feature Scenario The objective is to observe how the explanations for two different outputs differ when both outputs have corresponding features present. The metric results are presented in Tab. 1, and visual examples are presented in Fig. 3. We observe that GradCAM, IBA, and Extremal Perturbation attribute the corresponding features when explaining the different outputs. FullGrad produces the same explanation when applied to the two outputs. We observe that Gradient, GuidedBackProp, DeepSHAP, and IntegratedGradient slightly switch explanations. We also perform layerwise experiments for IBA and GradCAM++. We observe that the explanations become less class sensitive in earlier layers.

Single Feature Scenario In this setting, we evaluate class sensitivity in the case where only one contributing feature is available. Suppose the explanation for the output of the

Metric	IBA				GradCAM++			
	layer 1	layer 2	layer 3	layer 4	layer 1	layer 2	layer 3	layer 4
Null Feature	0.315	0.311	0.201	0.211	0.827	0.906	0.815	0.453
Double Feature Scenario	0.327	0.337	0.207	0.191	0.977	0.948	0.899	0.469
Single Feature Scenario	0.219	0.237	0.158	0.295	0.979	0.823	0.761	0.845

Table 2. Evaluations of IBA and GradCAM++ explanations for various layers of ResNet: IBA and GradCAM++ are applicable to different layers of convolutional networks. However, we observe that as we move towards earlier layers (toward the input), more attribution is assigned to the null feature. We also observe the same trend with class sensitivity. The results significantly deteriorate for GradCAM++ (in both experiments, the lower the value, the better). It is thus advisable to apply these explanations to deep layers.

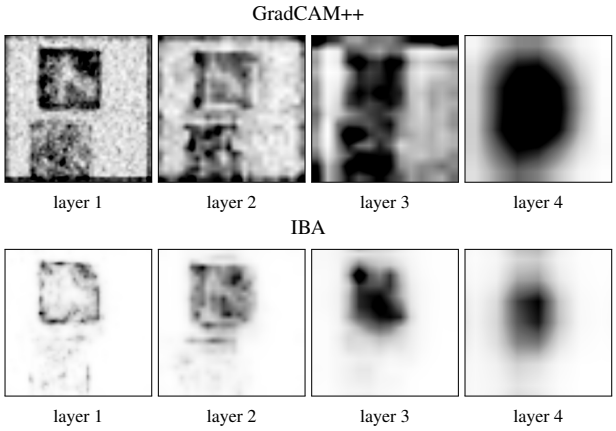


Figure 2. Null Feature Experiment for IBA and GradCAM++ on different layers of a network. The second (lower) feature is a null feature. We observe that as we move toward earlier layers, the explanations attribute to the null feature for both methods.

corresponding feature is similar to the explanation for an output to which the feature does not contribute. In that case, the explanation is not sensitive to the output. A visual example for this case is provided in Fig. 4. The results for the associated metric are provided in Tab. 1. In this scenario, more explanation methods are prone to attribute the output to the single feature within the image. Interestingly the only method that is sensitive to output, in this case, is GradCAM. Even IBA and Extremal perturbation that performed well in double feature scenario identify the same feature for the two outputs. This might be the property for all perturbation/removal-based methods, that they converge to the only predictive feature within the input, even if the feature is predictive for another class.

4.3. Feature Saturation

This experiment aims to check how an attribution method behaves in case there are saturated features present in the input. The desirable property, in this case, is to attribute to both features. The results of the metric are provided in Tab. 1. A visual example is presented in the appendix (Fig. 5). The two features (patches) in the input contribute equally to the output, and the presence of only one is enough for the exact output prediction. We expect to observe that a method such as Extremal Perturbation attributing to only one of the features. The method searches for the smallest region that keeping it would keep the output prediction. In the case of saturated features, this translates to keeping only one feature. The metric in Tab. 1 shows that statistically, the method converges to one of the features. The other methods are mostly attributing to both features, but considering the results from the Null player experiment and the Class Sensitivity experiment, the observation better be interpreted with caution. Several methods might be attributing to both features because they attribute to all features for other reasons. For instance, we observed with GuidedBackprop that the method is attributing to null feature and is attributing to both features when explaining different outputs. A method that is attributing to both features, but does that in null feature case as well, is not doing the attribution for fair distribution, but for other reasons.

4.4. Discussion on the Framework

Is the optimization feasible? To practically guarantee that the setups (Eq. 1-7) are realized, we set stopping criteria and continue the optimization until the desired setup is achieved. The stopping criteria checks if the properties (e.g., one feature being null) are satisfied within a certain threshold. The setups are variations of removing patches; we thus check the effect of removing patches on output in all setups after each epoch. We report the output change ratio (output-change / output) when removing patches in different setups (null, class-sensitivity, saturation) for an average of 1K samples. For the Null Feature, Feature Saturation, and Class-Sensitivity (double), the ratio is 0.0712. For Class-sensitivity (single) ratio is 0.0649.

Is the framework sensitive to how the samples are generated? By definition, given a function, for any input/output, the attribution method ought to identify the contribution of features to the output. The definition is for *any* input and is regardless of the generated input's properties (e.g. being out of distribution). The framework makes sure

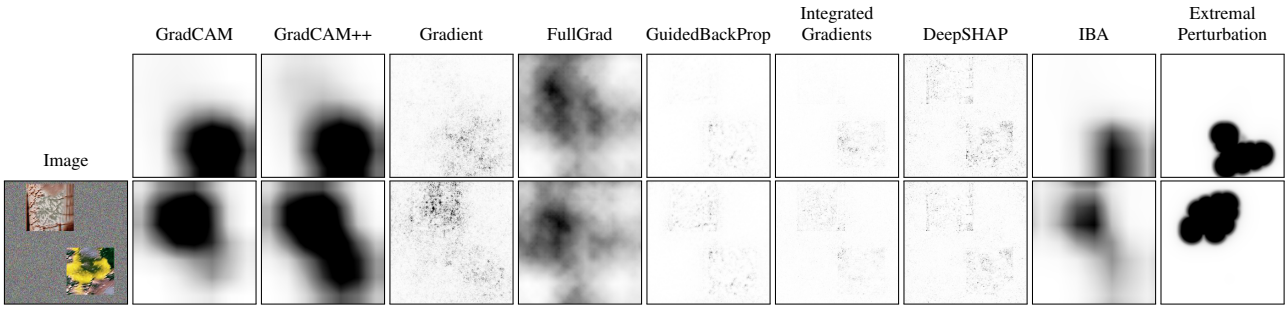


Figure 3. **Class Sensitivity - Double Feature Scenario**: The image on the left represents the generated features on the reference input. The features are generated such that each corresponds to a different output. The lower feature (patch) corresponds to the first output (first row), and the other patch corresponds to the second output (second row). It is expected that explanations for the two outputs differ and attribute to the corresponding feature of each output. GradCAM, IBA, and Extremal Perturbation manifest this property.

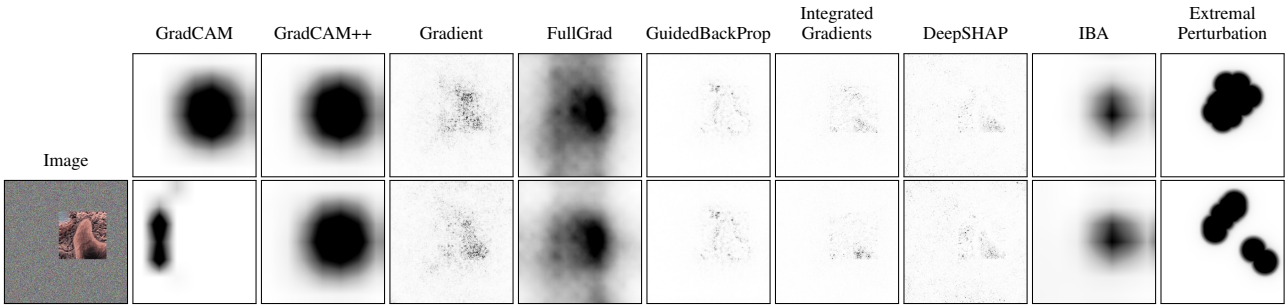


Figure 4. **Class Sensitivity - Single Feature Scenario**: The feature is generated such that it contributes to one output and is null for another output. The first output is presented in the first row. The output to which the feature is null is presented below. The explanations are presented for both outputs. It is expected that the explanations for the two outputs differ. Moreover, the explanations should not attribute to the feature for the null output (second row.). The only method that attributes correctly, in this case, is GradCAM.

that the feature has a specific behavior, e.g., having zero contribution to the output.

Novel insights from the framework in a nutshell: We reveal FullGrad, GradCAM++, Integrated Gradients and Gradient are attributing to null feature. We practically affirm DeepSHAP breaks axioms (theory in [34]). We show CAM, Extremal Perturbations (Exp), and IBA as trustworthy in terms of null and class-sensitivity axiom (though when only one feature is present, only CAM prevails). We reveal saturation properties within ExP and IBA. We reveal GradCAM++, FullGrad, Gradient, IG, DeepSHAP can be class-insensitive. We show (Tab. 2) IBA and GradCAM++ break axioms in early layers (though they were proposed to work on other layers than the deepest).

5. Conclusion

This work proposes an experimental framework for axiomatic evaluation of explanation methods using the model. Within the framework, the explanations are checked whether they comply with an axiom or satisfy a property. The experimental setup is realized through generating features using the model. Through feature generation, several scenarios for evaluating axioms are introduced. The framework reveals that many explanation methods identify a null feature as salient, even though the framework guarantees the feature to have no contribution. Moreover, the framework shows many explanations are not class sensitive and generate roughly equivalent explanations for different outputs. The only methods that do not attribute to null features and are class sensitive are GradCAM, IBA, and Extremal Perturbations. We further analyze IBA and GradCAM++ on various layers of a neural network and reveal that the axioms are complied with only if they are applied to the final layer. Our proposed framework can be used to evaluate upcoming explanation methods. Furthermore, researchers can add more creative experiments to the proposed framework to assess explanations from other perspectives.

Acknowledgement The work is supported by Munich Center for Machine Learning (MCML) with funding from the Bundesministerium fur Bildung und Forschung (BMBF) under the project 01IS18036B.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, 2018. 3

[2] Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *ICLR*, 2018. 3

[3] Leila Arras, Ahmed Osman, and Wojciech Samek. Ground truth evaluation of neural network explanations with clevr-xai. *arXiv preprint arXiv:2003.07258*, 2020. 3

[4] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010. 2

[5] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[7] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2

[8] Ruth Fong, Mandela Patrick, and Andrea Vedaldi. Understanding deep networks via extremal perturbations and smooth masks. In *ICCV*, 2019. 1, 2

[9] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *ICCV*, pages 3429–3437, 2017. 1, 2

[10] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 270–287, 2018. 5, 11

[11] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9737–9748, 2019. 1, 3, 11

[12] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017. 3

[13] Ashkan Khakzar, Soroosh Baselizadeh, Saurabh Khanduja, Christian Rupprecht, Seong Tae Kim, and Nassir Navab. Neural response interpretation through the lens of critical pathways. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13528–13538, June 2021. 1, 2

[14] Ashkan Khakzar, Soroosh Baselizadeh, and Nassir Navab. Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods. *arXiv preprint arXiv:2012.00362*, 2020. 3, 11

[15] Ashkan Khakzar, Yang Zhang, Wejdene Mansour, Yuezhi Cai, Yawei Li, Yucheng Zhang, Seong Tae Kim, and Nassir Navab. Explaining covid-19 and thoracic pathology model predictions by identifying informative input features. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 391–401. Springer, 2021. 1

[16] Scott M. Lundberg and Su In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017. 1, 2, 3, 11

[17] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015. 2

[18] Saumitra Mishra, Bob L Sturm, and Simon Dixon. Local interpretable model-agnostic explanations for music content analysis. In *ISMIR*, pages 537–543, 2017. 2

[19] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus Robert Müller. Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognition*, 2017. 1, 2

[20] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Advances in neural information processing systems*, pages 3387–3395, 2016. 2

[21] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. *International Conference on Machine Learning*, 2018. 3, 5

[22] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization. 2

[23] Zhongang Qi, Saeed Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. *CVPR Workshop*, 2019. 2

[24] Sylvestre-Alvise Rebuffi, Ruth Fong, Xu Ji, and Andrea Vedaldi. There and Back Again: Revisiting Backpropagation Saliency Methods. apr 2020. 3

[25] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, and Klaus Robert Müller. Evaluating the visualization of what a deep neural network has learned. *IEEE Transactions on Neural Networks and Learning Systems*, 2017. 1, 3, 11

[26] Karl Schulz, Leon Sixt, Federico Tombari, and Tim Landgraf. Restricting the flow: Information bottlenecks for attribution. In *International Conference on Learning Representations*, 2020. 1, 2, 5

[27] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 1, 2, 3

[28] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953. 1

[29] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *34th International Conference on Machine Learning, ICML*, 2017. 1, 2, 5

[30] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[31] Leon Sixt, Maximilian Granz, and Tim Landgraf. When explanations lie: Why many modified bp attributions fail. *International Conference on Machine Learning*, pages 9046–9057, 2020. 3, 11

[32] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. In *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015. 1, 2, 3

[33] Suraj Srinivas and Francois Fleuret. Full-gradient representation for neural network visualization. Technical report, 2019. 1

[34] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *37th International Conference on Machine Learning, ICML*, 2020. 1, 2, 3, 8, 11

[35] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning, ICML 2017*, 2017. 1, 2, 3, 5, 11

[36] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 2, 5

[37] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014. 2, 3

[38] Jianming Zhang, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *International Journal of Computer Vision*, 126(10):1084–1102, 2018. 1, 3, 11

[39] Yang Zhang, Ashkan Khakzar, Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. Fine-grained neural network explanation by identifying input features with predictive information. *Advances in Neural Information Processing Systems*, 34, 2021. 1, 2

[40] Yang Zhang, Ashkan Khakzar, Yawei Li, Azade Farshad, Seong Tae Kim, and Nassir Navab. Fine-grained neural network explanation by identifying input features with predictive information. *arXiv preprint arXiv:2110.01471*, 2021. 2

[41] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016. 1, 2

Part III

Rethinking Feature Attribution in Medical Image Analysis

8 Through Inverting the Game

In this chapter, we delve into explaining thoracic pathology models through feature attribution. The thoracic diagnosis models are of specific interest in the medical imaging community. First and foremost, at the time of the project, the world was hit with the Covid-19 pandemic. Second, the chest x-ray datasets are among the largest available to the public in the medical image analysis domain. The neural network models in this domain have long shown impressive performance in diagnosis [51]. However, the question remains whether these models use features relevant to the pathologies. Therefore we adopt one of the most potent tools in our arsenal, the information bottleneck for attribution.

Through information bottleneck attribution, we investigate which input features have predictive information for Covid-19 and thoracic pathology diagnosis. Moreover, we raise the point that the formulation of information bottleneck attribution (IBA) [97] leads to identifying features sufficient for the prediction. If there are other predictive features within the input, the methodology resorts to retrieving the features that are sufficient for prediction and ignores the rest. We discuss that this is due to the formulation of the method being a preservation game. In a preservation game, the objective is to preserve as few features as possible while keeping the output of the neural network unchanged. However, within a deletion game, the objective is to remove as few features as possible until there the output of the neural network flips. I.e., there are no more features left within the input to classify the input as the original label. We tweak the information bottleneck attribution such that it follows the deletion game principle. The new formulation, InverseIBA, identifies *any* predictive input feature. A property that benefits medical image analysis, as any predictive marker helps clinicians with diagnosis.

In addition to introducing InverseIBA, we further extend the methodology towards feature attribution for regression models. As discussed in chapter 6, the bottleneck attribution is most sound when applied to deep layers, resulting in attributions with low resolution (resolution equal to the deep feature maps). In this chapter, we also propose a trick for this issue for CNNs. The trick seems to work empirically. However, it is not as theoretically grounded as our solution in chapter 6. The main contribution of this chapter is the introduction of the inverse methodology for identifying all predictive features. Moreover, we analyze the alignment of the salient features with annotations

from experts to investigate the degree to which the models use pathologically relevant features.

Explaining COVID-19 and Thoracic Pathology Model Predictions by Identifying Informative Input Features

Ashkan Khakzar[1,*], Yang Zhang[1,*], Yuezhi Cai[1], Yawei Li[1], Yucheng Zhang[1], Seong Tae Kim[2] and Nassir Navab[1,3]

[1] Technical University of Munich
[2] Kyung Hee University, South Korea
[3] Johns Hopkins University
[*] Equal contribution

Published in: International Conference on Medical Image Computing and Computer-Assisted Intervention 2021 - MICCAI 2021 [105]

Published version: https://link.springer.com/chapter/10.1007/978-3-030-87199-4_37

Explaining COVID-19 and Thoracic Pathology Model Predictions by Identifying Informative Input Features

Ashkan Khakzar[1,⋆] , Yang Zhang[1,⋆], Wejdene Mansour[1], Yuezhi Cai[1], Yawei Li[1], Yucheng Zhang[1], Seong Tae Kim[2,*,†], Nassir Navab[1,3,*]

[1]Technical University of Munich
[2]Kyung Hee University
[3]Johns Hopkins University

Abstract. Neural networks have demonstrated remarkable performance in classification and regression tasks on chest X-rays. In order to establish trust in the clinical routine, the networks' prediction mechanism needs to be interpretable. One principal approach to interpretation is feature attribution. Feature attribution methods identify the importance of input features for the output prediction. Building on Information Bottleneck Attribution (IBA) method, for each prediction we identify the chest X-ray regions that have high mutual information with the network's output. Original IBA identifies input regions that have *sufficient* predictive information. We propose Inverse IBA to identify *all* informative regions. Thus all predictive cues for pathologies are highlighted on the X-rays, a desirable property for chest X-ray diagnosis. Moreover, we propose Regression IBA for explaining regression models. Using Regression IBA we observe that a model trained on cumulative severity score labels implicitly learns the severity of different X-ray regions. Finally, we propose Multi-layer IBA to generate higher resolution and more detailed attribution/saliency maps. We evaluate our methods using both human-centric (ground-truth-based) interpretability metrics, and human-agnostic feature importance metrics on NIH Chest X-ray8 and BrixIA datasets. The code[1] is publicly available.

Keywords: Explainable AI · Feature Attribution · Chest X-rays · Covid

1 Introduction

Deep Neural Network models are the de facto standard in solving classification and regression problems in medical imaging research. Their prominence is specifically more pronounced in chest X-ray diagnosis problems, due to the availability of large public chest X-ray datasets [30, 6, 19, 7]. Chest X-ray is an economical,

⋆ denotes joint first author, and * S. T. Kim and N. Navab shared senior authorship.
† denotes corresponding author (st.kim@khu.ac.kr)
[1] https://github.com/CAMP-eXplain-AI/CheXplain-IBA

fast, portable, and accessible diagnostic modality. A modality with the aforementioned properties is specifically advantageous in worldwide pandemic situations such as COVID-19 where access to other modalities such as Computed Tomography (CT) is limited [23, 16, 18]. Therefore, diagnostic chest X-ray neural network models can be of great value in large-scale screening of patients worldwide.

However, the black-box nature of these models is of concern. It is crucial for their adoption to know whether the model is relying on features relevant to the medical condition. In pursuit of interpretability of chest X-ray models, a class of works focuses on instilling interpretability into the models during optimization [29, 9, 26], another class pursues optimization semi-supervised with localization [13], and another class of works provides post-hoc explanations [30, 19, 8]. Post-hoc explanations have the advantage that they can be applied to any model without changing the objective function.

One principal method for post-hoc explanation is feature attribution (aka saliency methods), i.e. identifying the importance/relevance of input features for the output prediction [25, 28, 11, 4, 22, 21]. Feature attribution problem remains largely open to this date, however, many branches of solutions are proposed. The question is which attribution solution to use. Attributions are evaluated from several perspectives, and one crucial and *necessary* aspect is to evaluate whether the attributed features are indeed important for model prediction, which is done by feature importance metrics [20, 3, 17]. One *desirable* property is human interpretability of the results, i.e. if the attribution is interpretable for the user. For example, Class Activation Maps (CAM, GradCAM) [32, 22] being a solid method that is adopted by many chest X-ray model interpretation works, satisfies feature importance metrics. However, it generates attributions that are of low resolution, and while accurately highlighting the important features, they do not highlight these regions with *precision*. Such precision is of more importance in chest X-rays where features are subtle. On the other hand, some other methods (e.g. Guided BackPropagation [27], $\alpha 1 \beta 0$ [4], Excitation Backprop [31]) have pixel-level resolution and are human-interpretable, but do not satisfy feature important metrics and some do not explain model behavior [1, 5, 12, 10, 15].

Information Bottleneck Attribution (IBA) [21] is a recent method proposed in neural networks literature that satisfies feature importance metrics, is more human-interpretable than established methods such as CAMs [32], and is of solid theoretical grounding. The method also visualizes the amount of information each image region provides for the output in terms of bits/pixels, thus its attribution maps (saliency maps) of different inputs are comparable in terms of quantity of the information (bits/pixels). Such properties make IBA a promising candidate for chest X-ray model interpretation.

In this work, we build upon IBA and propose extended methodologies that benefit chest X-ray model interpretations.

1.1 Contribution Statement

Inverse IBA: The original IBA method finds input regions that have sufficient predictive information. The presence of these features is sufficient for the target

prediction. However, if sufficient features are removed, some other features can have predictive information. We propose Inverse IBA to find any region that can have predictive information.
Regression IBA: IBA (and many other methods such as CAMs) is only proposed for classification. We propose Regression IBA and by using it we observe that a model trained on cumulative severity score labels implicitly learns the severity of different X-ray regions.
Multi-layer IBA: We investigate approaches to use the information in layers of all resolutions, to generate high-resolution saliency maps that *precisely* highlight informative regions. Using Multi-layer IBA, for instance, we can precisely highlight subtle regions such as Mass, or we observe that the model is using corner regions to classify Cardiomegaly.
Effect of balanced training: We also observe that considering data imbalance during training results in learned features being aligned with the pathologies.

2 Methodology

Information Bottleneck for Attribution (IBA) [21] inserts a bottleneck into an existing network to restrict the flow of information during inference given an input. The bottleneck is constructed by adding noise into the feature maps (activations) of a layer. Let F denote the feature maps at layer l, the bottleneck is represented by $Z = \lambda F + (1 - \lambda)\epsilon$, where ϵ is the noise, the mask λ has the same dimension as F and controls the amount of noise added to the signal. Each element in the mask $\lambda_i \in [0, 1]$. Since the goal is post-hoc explanation for an input X, the model weights are fixed and *the mask λ is optimized* such that mutual information between the noise-injected activation maps Z and the input X is minimized, while the mutual information between Z and the target Y is maximized:

$$\max_{\lambda} I(Y, Z) - \beta I(X, Z) \tag{1}$$

The term $I(X, Z)$ is intractable, thus it is (variationally) approximated by

$$I(X, Z) \approx \mathcal{L}_I = E_F[D_{KL}(P(Z|F)||Q(Z))] \tag{2}$$

where $Q(Z) \sim \mathcal{N}(\mu_F, \sigma_F)$ (μ_F and σ_F are the estimated mean and variance of hidden feature F from a batch of data samples). In [21], the mutual information $I(Y, Z)$ is replaced by cross entropy loss L_{CE}. It is proven $-L_{CE}$ is a lower bound for I(Y,Z) [2]. Minimizing L_{CE} corresponds to maximizing $-L_{CE}$ and thus maximizing the lower bound of I(Y,Z). The objective becomes:

$$\mathcal{L} = \beta \mathcal{L}_I + \mathcal{L}_{CE} \tag{3}$$

2.1 Inverse IBA

In IBA formulation (Eq. 3), the first term tries to remove as many features as possible (by setting $\lambda = 0$) and while the second term tries to keep features (by

setting $\lambda = 1$) such that mutual information with target Y is kept. Therefore, if only a small region can keep the second term ($\mathcal{L}_{CE}$) minimized (keep the mutual information with target Y), the rest of the features are removed (their $\lambda = 0$). The identified regions ($\lambda = 1$) have *sufficient* predictive information, as their existence is sufficient for the prediction. However, there might exist other regions that have predictive information in the absence of these sufficient regions. From another perspective, IBA is playing a preservation game, which results in preserving features that keep the output close to the target.

To find all regions that have predictive information we change the formulation of IBA such that the optimization changes to a deletion game. I.e. deleting the smallest fraction of features such that there is no predictive information for the output anymore after deletion. In order to change IBA optimization to a deletion game we make two changes: 1) for the second term ($\mathcal{L}_{CE}$) in Eq. 3 we use an inverse mask: $Z_{inv} = \lambda\epsilon + (1 - \lambda)F$, and denote the new term with $\mathcal{L}_{CE}^{inv}$. 2) we maximize the $\mathcal{L}_{CE}^{inv}$ in order for the optimization to remove predictive features. Thus, the objective is:

$$\mathcal{L}_{inv} = \beta\mathcal{L}_I - \mathcal{L}_{CE}^{inv} \tag{4}$$

Minimizing $\mathcal{L}_I$ corresponds to the feature map becoming noise (similar to IBA) and pushes λ to 0. Minimizing $\mathcal{L}_{CE}^{inv}$ (maximizing $\mathcal{L}_{CE}^{inv}$) in Eq. 4 corresponds to removing *all* predictive information. In the $\mathcal{L}_{CE}^{inv}$ term, we use Z_{inv}, thus removing features corresponds to pushing the λ to 1 (if we instead use Z instead of Z_{inv}, λ moves to 0, and as $\mathcal{L}_I$ also pushes λ to 0, we get 0 everywhere). Therefore, λ is pushed to 1 for *any* predictive feature, and to 0 for the rest. As such, Inverse IBA identifies *any* predictive feature in the image and not just the *sufficiently* predictive features (examples in Fig. 1).

2.2 Regression IBA

Original IBA is proposed for classification setting. In this section, we discuss several variations of IBA for the regression case. We discuss three different regression objectives: 1) MSE Loss defined as $\mathcal{L}_{MSE} = (\mathbf{\Phi}(Z) - \mathbf{y})^2$. MSE loss has the property that if the target score is small, it identifies regions with small brixIA score as informative. Because in this case, the objective is trying to find regions that have information for output to be zero. 2) Regression Maximization (RM) Loss is simply defined as $\mathcal{L}_{RM} = \mathbf{\Phi}(Z)^2$. This loss has the property that it favors regions with high scores as informative. 3) Deviation loss defined as $\mathcal{L}_{DV} = (\mathbf{\Phi}(Z) - X)^2$. We subtract the score of the noisy feature map from the score of the original image. Similar to IBA for classification, this formulation identifies regions with sufficient information for the prediction. We also apply Inverse IBA to regression (see Fig. 1) to identify all preditive features.

2.3 Multi-layer IBA

For original IBA, the bottleneck is inserted in one of the later convolutional layers. As we move towards earlier layers, the variational approximation becomes

less accurate. Thus the optimization in Eq. 3 highlights extra regions that do not have predictive information in addition to highlighting the sufficient regions. However, as the resolution of feature maps in earlier layers are higher, the highlighted regions are crisper and more interpretable. In order to derive regions that are crips and have high predictive information we compute IBA for several layers and combine their results, thus introducing Multi-layer IBA:

$$\mathcal{T}(IBA_{L_1}) \cap \mathcal{T}(IBA_{L_2}) ... \cap \mathcal{T}(IBA_{L_L}) \tag{5}$$

where $\mathcal{T}$ denotes a thresholding operation to binarize the IBA maps.

2.4 Chest X-ray Models

Classification model: We denote a neural network function by $\mathbf{\Phi_\Theta(x)} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^C$ where C is the number of output classes. For a dataset $\mathbf{X} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$, and their labels $\mathbf{Y} = \{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(N)}\}$, where $\mathbf{y} = [\mathbf{y}_j]^C$, and $\mathbf{y_j} \in \{0, 1\}$. Chest X-rays can have multiple pathologies. We use Binary Cross Entropy (BCE) loss on each output for multilabel prediction.

$$\mathcal{L}_{BCE} = (\hat{\mathbf{y}}, \mathbf{y}) = -\sum_j \beta \mathbf{y_j} \log(\hat{\mathbf{y_j}}) + (1 - \mathbf{y_j}) \log(1 - \hat{\mathbf{y_j}}) \tag{6}$$

where β is a weighting factor to balance the positive labels.

Regression model: Consider a neural network $\mathbf{f_\Theta(x)} : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}$ and a dataset $\mathbf{X} = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}\}$ of N X-ray images, and their corresponding labels $\mathbf{Y} = \{\mathbf{y}^{(1)}, ..., \mathbf{y}^{(N)}\}$, where $\mathbf{y}_j \in 0, ..., 18$ is the cumulative severity score on each image. We model the regression problem with a MSE loss:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum (\mathbf{\Phi_\Theta(x)^{(n)}} - \mathbf{y^{(n)}})^2 \tag{7}$$

3 Experiments and Results

Implementation Details We use three models: 1) NIH ChestX-ray8 classification: Network with 8 outputs for the 8 pathologies. 2) BrixIA regression: Network with one output and predicts the total severity score (sum of severity scores of 6 regions) 3) BrixIA classifier: 3 outputs detecting whether a severity score of 3, 2, and 0/1 exists in the X-rays. We use Densenet 121, and insert the IBA bottleneck on the output of DenseBlock 3. For Multi-layer IBA we insert it on the outputs of DenseBlock 1,2 and 3.

3.1 Feature Importance (Human-Agnostic) Evaluations

Experiments in this section evaluate whether an attribution method is identifying important features for the model prediction.

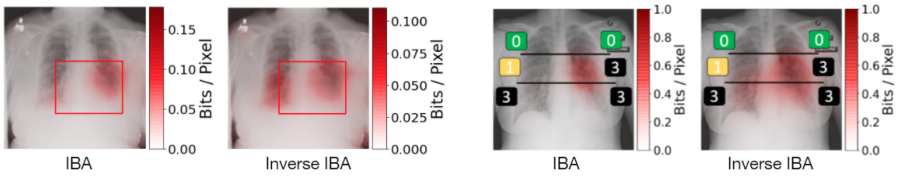


Fig. 1. Inverse IBA: Inverse IBA compared with IBA on a sample from the NIH Chest X-ray8 (left) and a sample from BrixIA (right). **NIH Chest X-ray8 (left)**: Inverse IBA is identifying both sides of Cardiomegaly as informative. The bounding box denotes the expert's annotation. **BrixIA (Right)**: IBA is identifying two regions with a severity score of 3 as sufficient for predicting the score of 3, however, Inverse IBA is identifying all regions with a severity score of 3, and the region with a score of 1. The horizontal lines denote the 6 regions within the lungs, and the numbers represent the sevirity score of each region.

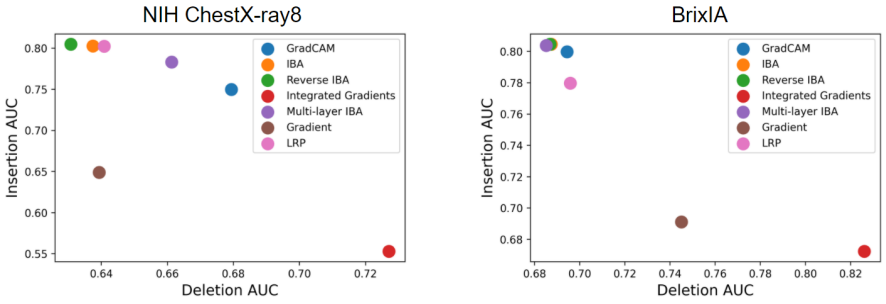
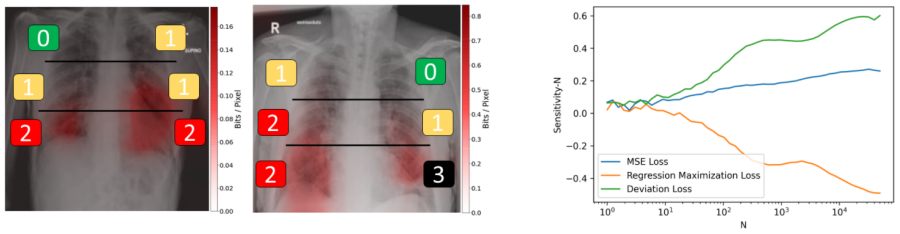


Fig. 2. Insertion/Deletion metric: Comparison of different attribution methods in terms of feature importance. Method with high Insertion AUC and low Deletion AUC is the best (top left corner is the best).

Insertion/Deletion [20, 17] Insertion: given a baseline image (we use the blurred image) features of the original image are added to the baseline image starting from the most important feature and the output is observed. If the attribution method is correct, after inserting a few features the prediction changes significantly, thus the AUC of output is high. Deletion: deleting important features first. The lower the AUC the better. Results are presented in Fig. 2.
Sensitivity-N [3] We adapt this metric to regression case (not trivial with Insertion/Deletion) and use it to evaluate Regression-IBA. In Sensitivity-n we mask the input randomly and observe the correlation between the output change and the values of attribution map overlapping with the mask. The higher the correlation the more accurate the attribution map. Results in Fig. 3b.



a) Regression IBA b) Sensitivity N

Fig. 3. Regression IBA: a) Regression IBA (with $\mathcal{L}_{DV}$) applied on a regression model that predicts the total severity score. Using Regression IBA we see that the model has identified the severity scores of different regions implicitly. b) Sensitivity N metric for evaluating feature importance of different Regression IBA losses

Table 1. Mean IOU on NIH ChestX-ray8 (all pathologies) for various methods

GradCAM[22]	InteGrad[28]	LRP[14]	Gradients[25]	IBA	Inverse IBA	Multi-layer IBA
0.077	0.076	0.025	0.114	0.114	0.088	**0.189**

3.2 Ground Truth based (Human-centric) Evaluations

Experiments in this section evaluate the attribution maps in terms of the human notion of interpretability, i.e. the alignment between what we understand and the map. Moreover, they measure how fine-grained the attribution maps are.

Localization For NIH ChestX-ray8 dataset the bounding boxes are available. To generate boxes for BrixIA score regions, similar to [24] we use a lung segmentation network with the same architecture in [24]. We divide each lung into 3 regions. We threshold the attribution maps and compute their IoU with these divided regions (over dataset).

Correlation Analysis (Regression Models) For the BrixIA dataset, we evaluate the performance of regression models by measuring the correlation between the attribution scores and the severity scores. For each image, we first assign each pixel with its severity score, obtaining a severity score map. We then flatten both the attribution and severity score maps and compute their Pearson correlation coefficient (PCC). The PCC results are as follows: for $\mathcal{L}_{MSE}$, 0.4766, for $\mathcal{L}_{RM}$, 0.4766, for $\mathcal{L}_{MSE}$, 0.4404, and for random heatmaps, 0.0004.

4 Discussion

Inverse IBA: We observe that (Fig. 1) Inverse IBA highlights all regions with predictive information. On BrixIA sample, IBA only identifies two regions with a score of 3 as being predictive, while Inverse IBA identifies all regions with a score of 3. On NIH sample, if we remove the highlighted areas of both methods (equally remove) from the image, the output change caused by the removal of

Table 2. Mean IOU on BrixIA for each detector and for various attribution methods

	GradCAM	InteGrad	LRP	Gradients	IBA	Inverse IBA	Multi-layer IBA
Detector 0/1	0.11	0.176	0.0	0.04	0.145	**0.194**	0.171
Detector 2	0.0	0.13	0.0	0.019	0.13	0.245	**0.257**
Detector 3	0.011	0.14	0.0	0.052	0.222	0.243	**0.257**

Table 3. Mean IOU on NIH ChestX-ray8 dataset for BCE and Weighted BCE models, reported for all the pathologies in NIH Chest X-ray8 using Inverse IBA

	Atelec.	Cardio.	Effusion	Infiltrate.	Mass	Nodule	Pneumo.	Pn. thorax	Mean
BCE	0.016	0.071	0.004	0.001	0.102	0.011	0.0	0.003	0.024
W. BCE	0.073	0.131	0.032	0.058	0.097	0.02	0.066	0.016	**0.065**

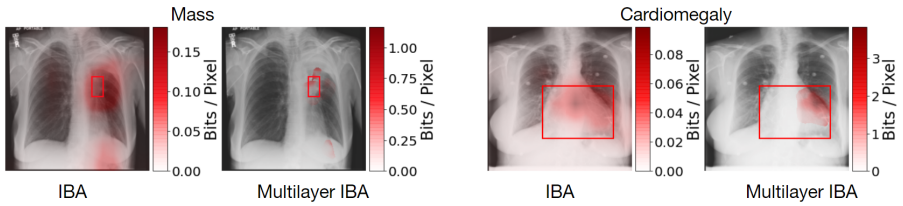


Fig. 4. Multi-layer IBA: Multi-layer IBA generates more fine-grained maps compared to IBA. (Left): Multi-layer IBA precisely highlights subtle features such as Mass (Right) Using Multi-layer we observe that for Cardiomegaly in this X-ray the corner regions of Cardiomegaly are used. IBA highlights the entire region.

Inverse IBA regions is higher. This is also quantitatively validated across dataset in the Deletion experiment (Fig. 2).

Regression IBA: Using Regression IBA we observe that (Fig. 3) a regression model which only predicts one cumulative severity score (0-18) for each X-ray implicitly identifies the severity scores of different regions.

Multi-layer IBA: We use Multi-layer IBA for obtaining fine-grained attributions. In Fig. 4 we see that such fine-grained attributions allow for identifying subtle features such as Mass. Moreover, Multi-layer IBA also uncovers some hidden insights regarding what features the model is using for the Cardiomegaly example. While IBA highlights the entire region, Multi-layer IBA shows precisely the regions to which IBA is pointing.

Imbalanced loss: We observe in Tab. 3 that using weighted BCE results in an increased IoU with the pathologies. This signifies that the contributing features of the weighted BCE model are more aligned with the pathology annotations. The observation is more significant when we consider the AUC of ROC and the Average Precision (AP) of these models. The AUCs (BCE=0.790, Weighted BCE=0.788) and APs (BCE=0.243, Weighted BCE=0.236) are approximately equivalent. The BCE even archives marginally higher scores in terms of AUC of ROC and AP but its learned features are less relevant to the pathologies.

5 Conclusion

In this work, we build on IBA feature attribution method and come up with different approaches for identifying input regions that have predictive information. Contrary to IBA, our Inverse IBA method identifies *all* regions that can have predictive information. Thus all predictive cues from the pathologies in the X-rays are highlighted. Moreover, we propose Regression IBA for attribution on regression models. In addition, we propose Multi-layer IBA, an approach for obtaining fine-grained attributions which can identify subtle features.

Acknowledgement This work was partially funded by the **Munich Center for Machine Learning (MCML)** and the **Bavarian Research Foundatio**n grant AZ-1429-20C. The computational resources for the study are provided by the **Amazon Web Services Diagnostic Development Initiative**. S.T. Kim is supported by the **Korean MSIT**, under the National Program for Excellence in SW (2017-0-00093), supervised by the IITP.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. arXiv preprint arXiv:1810.03292 (2018)
2. Alemi, A.A., Fischer, I., Dillon, J.V., Murphy, K.: Deep variational information bottleneck. arXiv preprint arXiv:1612.00410 (2016)
3. Ancona, M., Ceolini, E., Öztireli, C., Gross, M.: Towards better understanding of gradient-based attribution methods for deep neural networks. arXiv preprint arXiv:1711.06104 (2017)
4. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
5. Hooker, S., Erhan, D., Kindermans, P.J., Kim, B.: A benchmark for interpretability methods in deep neural networks. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/file/fe4b8556000d0f0cae99daa5c5c5a410-Paper.pdf
6. Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., et al.: Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 590–597 (2019)
7. Johnson, A.E., Pollard, T.J., Greenbaum, N.R., Lungren, M.P., Deng, C.y., Peng, Y., Lu, Z., Mark, R.G., Berkowitz, S.J., Horng, S.: Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. arXiv preprint arXiv:1901.07042 (2019)
8. Karim, M., Döhmen, T., Rebholz-Schuhmann, D., Decker, S., Cochez, M., Beyan, O., et al.: Deepcovidexplainer: Explainable covid-19 predictions based on chest x-ray images. arXiv preprint arXiv:2004.04582 (2020)
9. Khakzar, A., Albarqouni, S., Navab, N.: Learning interpretable features via adversarially robust optimization. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 793–800. Springer (2019)

10. Khakzar, A., Baselizadeh, S., Khanduja, S., Kim, S.T., Navab, N.: Explaining neural networks via perturbing important learned features. arXiv preprint arXiv:1911.11081 (2019)
11. Khakzar, A., Baselizadeh, S., Khanduja, S., Rupprecht, C., Kim, S.T., Navab, N.: Neural response interpretation through the lens of critical pathways. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021)
12. Khakzar, A., Baselizadeh, S., Navab, N.: Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods. arXiv preprint arXiv:2012.00362 (2020)
13. Li, Z., Wang, C., Han, M., Xue, Y., Wei, W., Li, L.J., Fei-Fei, L.: Thoracic disease identification and localization with limited supervision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8290–8299 (2018)
14. Montavon, G., Lapuschkin, S., Binder, A., Samek, W., Müller, K.R.: Explaining nonlinear classification decisions with deep taylor decomposition. Pattern Recognition **65**, 211–222 (2017)
15. Nie, W., Zhang, Y., Patel, A.: A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In: International Conference on Machine Learning. pp. 3809–3818. PMLR (2018)
16. Oh, Y., Park, S., Ye, J.C.: Deep learning covid-19 features on cxr using limited training data sets. IEEE Transactions on Medical Imaging **39**(8), 2688–2700 (2020)
17. Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. arXiv preprint arXiv:1806.07421 (2018)
18. Punn, N.S., Agarwal, S.: Automated diagnosis of covid-19 with limited posteroanterior chest x-ray images using fine-tuned deep neural networks. Applied Intelligence pp. 1–14 (2020)
19. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al.: Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)
20. Samek, W., Binder, A., Montavon, G., Lapuschkin, S., Müller, K.R.: Evaluating the visualization of what a deep neural network has learned. IEEE transactions on neural networks and learning systems **28**(11), 2660–2673 (2016)
21. Schulz, K., Sixt, L., Tombari, F., Landgraf, T.: Restricting the flow: Information bottlenecks for attribution. arXiv preprint arXiv:2001.00396 (2020)
22. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
23. Signoroni, A., Savardi, M., Benini, S., Adami, N., Leonardi, R., Gibellini, P., Vaccher, F., Ravanelli, M., Borghesi, A., Maroldi, R., et al.: End-to-end learning for semiquantitative rating of covid-19 severity on chest x-rays. arXiv preprint arXiv:2006.04603 (2020)
24. Signoroni, A., Savardi, M., Benini, S., Adami, N., Leonardi, R., Gibellini, P., Vaccher, F., Ravanelli, M., Borghesi, A., Maroldi, R., et al.: Bs-net: Learning covid-19 pneumonia severity on a large chest x-ray dataset. Medical Image Analysis **71**, 102046 (2021)
25. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
26. Singh, R.K., Pandey, R., Babu, R.N.: Covidscreen: Explainable deep learning framework for differential diagnosis of covid-19 using chest x-rays. Neural Computing and Applications pp. 1–22 (2021)

27. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806 (2014)
28. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning. pp. 3319–3328. PMLR (2017)
29. Taghanaki, S.A., Havaei, M., Berthier, T., Dutil, F., Di Jorio, L., Hamarneh, G., Bengio, Y.: Infomask: Masked variational latent representation to localize chest disease. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 739–747. Springer (2019)
30. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R.M.: Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2097–2106 (2017)
31. Zhang, J., Bargal, S.A., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down neural attention by excitation backprop. International Journal of Computer Vision **126**(10), 1084–1102 (2018)
32. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2921–2929 (2016)

9 Through the Lens of Concepts

Feature attribution, though still open for exploration, is merely one way to explain neural networks. In this chapter, we move beyond feature attribution and explore concepts encoded by neurons in the neural network. This can be achieved by analyzing the activation patterns of neurons. Network dissection [40] is a simple and elegant method that observes the activations of individual neurons throughout a dataset. We can analyze the neurons and understand if there is a correlation between the activation of a neuron and certain concepts. Thus, we can identify whether a neuron is associated with these concepts. We refer to a neuron that is associated with a concept as a concept detector. We explore this methodology for analyzing and interpreting thoracic pathology and Covid-19 diagnosis models. This chapter investigates the following questions:

If we train a regression neural network on a chest x-ray dataset labeled with Covid-19 severity scores, would the network implicitly learn concepts associated with the pathologies? We monitor the emergence of concept detectors during training and observe that the number of concept detectors increases. Moreover, we observe a variety of concept detectors appearing within the trained neural network.

If we train a classification neural network on a chest x-ray dataset labeled with binary labels, healthy/unhealthy, would the network implicitly learn pathology-related concepts? We observe that concept detectors indeed emerge within the network. Moreover, considering class imbalance within the dataset affects the number of concept detectors.

The results presented in this chapter are intriguing as they show that without any explicit cues regarding pathologies, concept detectors associated with pathologies emerge within the neural networks. The results help establish trust in using neural networks in medical image analysis. We further explore the methodology in our consequent work [112] for vertebrae fracture detection and ask radiologists to evaluate the detectors. We also identify the existence of clinically relevant concepts in networks trained on fracture detection [112].

Towards Semantic Interpretation of Thoracic Disease and COVID-19 Diagnosis Models

Ashkan Khakzar[1], Sabrina Musatian[1], Jonas Buchberger[1], Icxel Valeriano Quiroz[1], Nikolaus Pinger[1], Soroosh Baselizadeh[1], Seong Tae Kim *[2], Nassir Navab *[1,3]

[1] Technical University of Munich
[2] Kyung Hee University, South Korea
[3] Johns Hopkins University

Published in: International Conference on Medical Image Computing and Computer-Assisted Intervention 2021 - MICCAI 2021 [104]

Published version: https://link.springer.com/chapter/10.1007/978-3-030-87199-4_47

Towards Semantic Interpretation of Thoracic Disease and COVID-19 Diagnosis Models

Ashkan Khakzar[1], Sabrina Musatian[1], Jonas Buchberger[1], Icxel Valeriano Quiroz[1], Nikolaus Pinger[1], Soroosh Baselizadeh[1], Seong Tae Kim[2,*,⋆], Nassir Navab[1,3,*]

[1]Technical University of Munich
[2]Kyung Hee University
[3]Johns Hopkins University

Abstract. Convolutional neural networks are showing promise in the automatic diagnosis of thoracic pathologies on chest x-rays. Their black-box nature has sparked many recent works to explain the prediction via input feature attribution methods (aka saliency methods). However, input feature attribution methods merely identify the importance of input regions for the prediction and lack semantic interpretation of model behavior. In this work, we first identify the semantics associated with internal units (feature maps) of the network. We proceed to investigate the following questions; Does a regression model that is only trained with COVID-19 severity scores implicitly learn visual patterns associated with thoracic pathologies? Does a network that is trained on weakly labeled data (e.g. healthy, unhealthy) implicitly learn pathologies? Moreover, we investigate the effect of pretraining and data imbalance on the interpretability of learned features. In addition to the analysis, we propose semantic attribution to semantically explain each prediction. We present our findings using publicly available chest pathologies (CheXpert [5], NIH ChestX-ray8 [25]) and COVID-19 datasets (BrixIA [20], and COVID-19 chest X-ray segmentation dataset [4]). The Code[1] is publicly available.

Keywords: Interpretability · COVID-19 · Chest X-rays.

1 Introduction

Convolutional neural networks (CNN) have demonstrated outstanding performance in automatic diagnosis on Chest X-rays [25, 12, 6, 5]. There are reports of CNNs outperforming radiologists in chest x-ray pathology classification [17]. These diagnostic models can aid the clinicians and expedite the diagnosis resulting in more patients receiving the care they need. Such models can be especially beneficial in pandemic circumstances as the shortage of expert clinicians becomes an issue [20, 14, 16]. Despite their performance, neural networks lack of

⋆ denotes corresponding author (st.kim@khu.ac.kr), * S. T. Kim and N. Navab shared senior authorship.

[1] https://github.com/CAMP-eXplain-AI/CheXplain-Dissection

interpretability undermines their reliability. It is essential to understand the basis of the network predictions, and the networks' learned features to establish trust in the clinical domain. Therefore there have been efforts in explicitly making the models more interpretable during training [24, 8], or interpreting neural network models after they are trained [25, 17, 10, 7]. In this work, we investigate the latter case in order to see if these performant models trained without an infusion of interpretability, are learning human-interpretable concepts.

For post-hoc explanation of chest X-ray models, many works opt for feature attribution methodologies [21, 23, 1, 18, 9, 25]. These works use feature attribution methods, such as Class Activation Maps (CAM) [25, 18] to reveal which input regions are contributing to the output prediction. Albeit being insightful, the aforementioned methodology lacks semantic interpretation of the models and their predictions. Network Dissection [2] is a methodology for identifying the corresponding concept of internal units (feature maps) of the network.

Contributions - In this work, we first use Network Dissection [2] to quantify the interpretability of chest X-ray classification models. Then we proceed to investigate the following; Does a neural network regression model that is only trained on COVID-19 severity scores (on BrixIA) implicitly learn visual patterns associated with thoracic pathologies? We also study the effect of pretraining on CheXpert and ImageNet datasets, and the effect of considering data imbalance on the semantics of internal units. Does a network trained on a weakly labeled dataset (healthy/unhealthy labels) implicitly learn distinct pathologies? We combine NIH ChestX-ray8, CheXpert, and BrixIA datasets to generate a massive but weakly labeled ('healthy','unhealthy') dataset. In this case, we study the effect of considering data imbalance on the semantics of internal units. Moreover, for both cases, we observe the formation of semantic units during training. In all experiments, we use bounding boxes in the NIH ChestX-ray8 [25], and segmentation masks in COVID-19 chest X-ray segmentation dataset [4] for identifying the semantics of units. In addition to the analysis, we propose semantic attribution by combining feature attribution and network dissection to semantically explain the prediction for each chest x-ray.

Related Work - Interpreting internal units: There are two principal categories of methods for this purpose; Methods that generate images that maximally activate neurons/units [13, 15, 21], and methods that search over the dataset to find which images (also image regions) activates neurons/units [27, 2]. Methods in the first category are prone to subjective interpretation as the generated images are ambiguous. **Network Dissection** [2] is a prominent method of the second category, that does quantitative analysis of the semantics of units. Though we use the same method (albeit on the different domain of chest x-rays rather than natural images), our experiments and insights differ. Effect of pretraining, imbalanced datasets, studying regression models and trained models on weakly-labeled datasets are exclusive to our work. We also propose semantic attribution. **DeepMiner** [26] is a methodology inspired by network dissection and applied

to mammograms. The methodology differs from Network Dissection and our work. In DeepMiner, instead of automatic annotation, the most important units are annotated by experts. DeepMiner thus differs in methodology, discusses a different domain (mammograms), and does not address our research questions.

2 Methodology

2.1 Setup: Chest X-ray Models:

Classification model: Each chest X-ray can contain multiple pathologies. Therefore we model the problem as a multi-label classification problem. For C pathologies, the network function is defined as $\mathbf{f}_{\Theta}(\mathbf{x}) : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{C}$. The predicted probability for each category is $\hat{\mathbf{y}} = sigmoid(\mathbf{f}_{\Theta}(\mathbf{x}))$. We use Binary Cross Entropy (BCE) loss on each output. Thus the loss is defined as:

$$\mathcal{L}_{BCE} = (\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{c} \beta \mathbf{y_c} \log(\hat{\mathbf{y_c}}) + (1 - \mathbf{y_c}) \log(1 - \hat{\mathbf{y_c}}) \tag{1}$$

where β is a weighting factor to balance the positive labels, and defined as the ratio of the number of negative labels to the number of positive labels in a batch.

Regression model: We consider two regression modelings on BrixIA dataset. 1) For each input j the label is one global sevirity score $\mathbf{y}^{(j)} \in \{0, ..., 18\}$. For this case the neural network function is $\mathbf{f}_{\Theta}(\mathbf{x}) : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}$. We use a weighted Mean Square Error (MSE) loss for a batch of size N:

$$\mathcal{L}_{MSE} = \frac{1}{N} \sum \beta (\mathbf{f_{\Theta}}(\mathbf{x})^{(\mathbf{n})} - \mathbf{y}^{(\mathbf{n})})^2 \tag{2}$$

2) In BrixIA dataset for each X-ray, the lung is divided into 6 regions, and a severity score $\in \{0, 1, 2, 3\}$ is assigned to each region. Thus the network is defined by $\mathbf{f}_{\Theta}(\mathbf{x}) : \mathbb{R}^{H \times W} \rightarrow \mathbb{R}^{6}$. Similar to [20] we use a mixed regression/classification loss by using Sparse Categorical Cross Entropy (SCCE) and diffentiable Mean Absolue Error (MAE^d), and for each sample and its corresponding network output $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_c]^6$ it is defined as $\mathcal{L}_{SCCE} + \mathcal{L}_{MAE^d}$:

$$\mathcal{L}_{SCCE} = -\frac{1}{C} \sum_{c} \mathbf{y}_c log(\hat{\mathbf{y}}_c) \tag{3}$$

$$\mathcal{L}_{MAE^d} = \frac{1}{C} \left\| \mathbf{y} - \sum_{c} \frac{e^{\hat{\mathbf{y}_c}}}{\sum_c e^{\hat{\mathbf{y}_c}}} c \right\| \tag{4}$$

2.2 Background: Network Dissection [2]

Network Dissection annotates individual units (feature maps) of neural networks with semantic concepts. We refer to a feature map that is associated with an

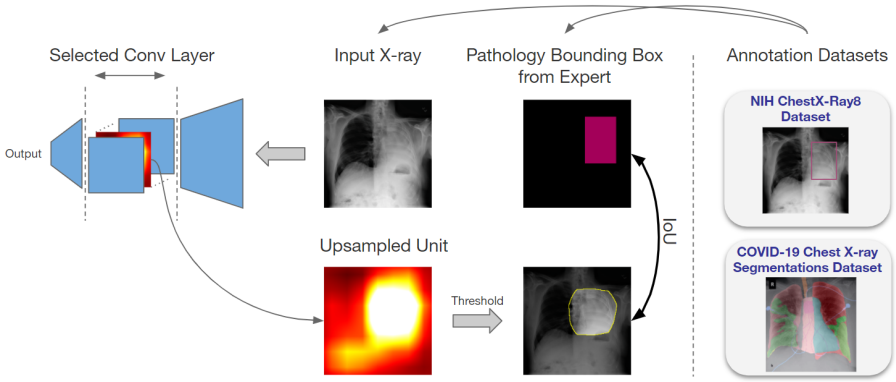


Fig. 1. Network Dissection on chest X-rays: To identify the corresponding concept of a unit in the network, the unit's activation for each input is compared with the ground truth mask/bounding-box of the concept in that input. For one unit the procedure is repeated for all inputs within the annotation datasets and average IoU with concepts in the dataset measures whether the unit is detecting a concept.

individual concept (e.g. consolidation) as an individual concept detector / a semantic unit. The method requires a dataset(s) where concepts are annotated with bounding box or segmentation masks. For each unit under investigation, the unit's feature map is computed for every image and is compared against the ground truth annotations (bounding boxes or segmentation masks) of the concepts in that image. The method compares the unit's feature map and the annotation in terms of their intersection over union. See Fig. 1 for a schematic overview. The activation map is first transformed in to a binary segmentation mask $S_k(\mathbf{x})$ via thresholding. For each unit k and its feature map $F_k(\mathbf{x})$, the threshold T_k is chosen such that $P(f_k > T_k) = 0.005$ [2] for every f_k given all images in the dataset. As the resolution of ground truth labels L_c for each concept c is different from S_k, bilinear interpolation is first applied to generate $S'_k(\mathbf{x})$. The binary segmentation mask is then derived by $M_k(\mathbf{x}) \doteq S'_k(\mathbf{x}) \geq T_k$. For each unit k and concept c pair, the intersection and the union between mask $M_k(\mathbf{x})$ derived from unit k and the label mask $L_c(\mathbf{x})$ is computed for all images containing concept c, and data-set-wide $IoU_{k,c}$ is defined as:

$$IoU_{k,c} = \frac{\sum |M_k(\mathbf{x}) \cap L_c(\mathbf{x})|}{\sum |M_k(\mathbf{x}) \cup L_c(\mathbf{x})|} \tag{5}$$

the sum is carried out over all the images with concept c, and $|.|$ denotes the cardinality of this set. The $IoU_{k,c}$ measures whether unit k detects concept c. We use a threshold of 0.04 similar to [2] for considering a unit as a detector. The threshold affects the number of concept detectors within a network, what is of interest is *comparing* the number of concept detectors between models (e.g. comparing trained and initial model). See Fig. 3 for units of different $IoU_{k,c}$.

Network Dissection for Chest X-ray Models: We use the NIH ChestX-ray8 [25] and COVID-19 chest X-ray segmentation [4] (Covid-CXR) dataset for annotating the chest X-ray models. NIH ChestX-ray8 contains bounding boxes for 8 pathologies, and we consider each pathology as a concept c in the Network Dissection framework. Covid-CXR contains segmentations for pathologies, lung components (e.g. right lung), and apparatus. we consider each one as a concept. The dataset contains 14 distinct concepts in total. For each model that we discuss in the paper, we use both NIH ChestX-ray8 and Covid-CXR datasets for annotating the models' internal units.

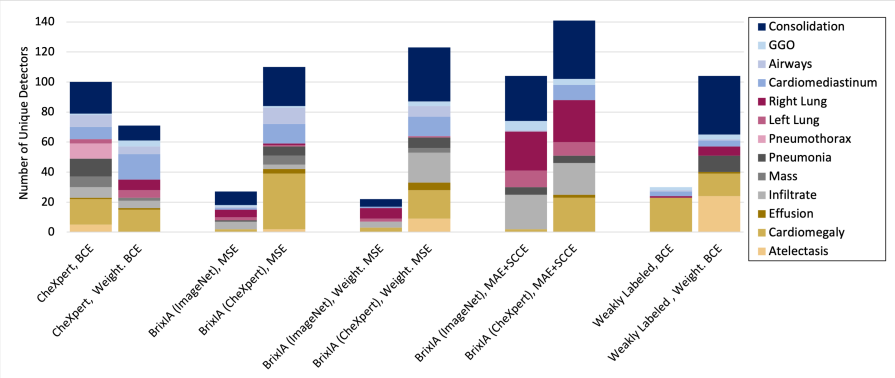


Fig. 2. The number/type of individual concept detectors: For all models under discussion, the number of concept detectors and the types of the detectors are presented. We can observe that pretraining on CheXpert is increasing the number and the variety of detectors. Moreover, considering data imbalance by using a weighted loss function also increases the number and variety of semantic detectors.

2.3 Semantic Attribution

For each prediction $\mathbf{f}_\Theta(\mathbf{x})$ on each input $\mathbf{x}$, we compute the importance of internal units A_k^L for the prediction. A_k^L denotes k_{th} feature map at layer L.

We follow a method, Integrated Gradients [23], that approximates the Aumann-Shapley value [22]. Shapley value [19] is the unique axiomatic definition of a feature's contribution to a final outcome. We first compute the individual neurons' importance a_k^i (activation i at channel k):

$$\mathbf{s}_k^i = \mathbf{a}_k^i \int_{\alpha=0}^{1} \frac{\partial \mathbf{f}_\theta(\alpha \mathbf{a}_k^i)}{\partial \mathbf{a}_k^i} \mathrm{d}\alpha \tag{6}$$

We use $\sum_i |\mathbf{s}_k^i|$ as the contribution of the unit k. The absolute value is for considering both positive and negative contributions by taking the magnitude.

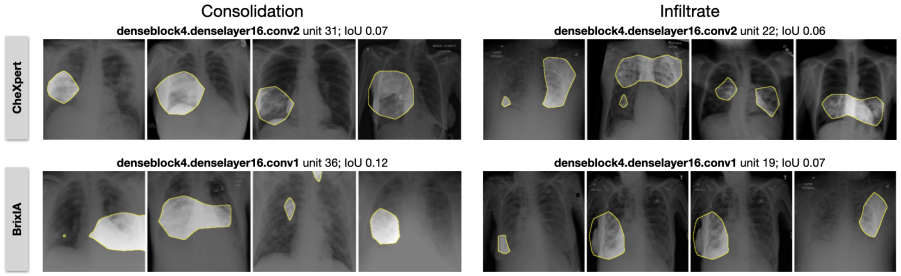


Fig. 3. Individual concept detectors: For CheXpert classification model (top) and BrixIA regression model (bottom) we visualize a Consolidation detector unit and an Infiltration detector unit for 4 different X-rays. The displayed IoU is not the IoU between the ground truth sample and the detector, it is the average from Eq. 5.

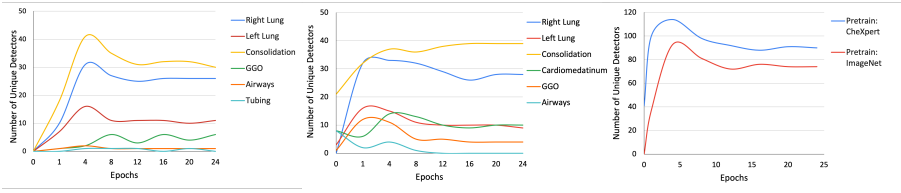


Fig. 4. Evolution of concepts: Number/Type of semantic detectors in two regression models at different epochs. (left): BrixIA (ImageNet), MAE+SCCE (middle): BrixIA (CheXpert), MAE+SCCE, (right) total number of semantic detectors for both models. We observe that the number of concept detectors is higher for trained models at different epochs compared to the initial model.

We then select the top contributing unit(s) and use their annotations (if it exists) for semantic explanation. We visualize $\mathbf{s}_k^i$ of all neurons of A_k^L to highlight each semantic unit's most contributing areas to the output.

3 Results and Discussion

In all our experiments and for all models in Fig. 2, the network is a DenseNet121 [3]. All models are trained using Adam [11] optimizer without weight decay. If the loss function is a weighted one, it is indicated by "Weight". For BrixIA models, the model is trained on a pre-trained model on either ImageNet or CheXpert. The name in parentheses shows this. When the last part of the name shows the loss function. "MAE+SCCE" indicates using the 6 region prediction as in the original BrixIA paper. The weakly labeled models are binary classifiers, trained on the combination of NIH, CheXpert, and BrixIA datasets, deciding if the input is healthy or not. While NIH dataset has a healthy label for inputs, for CheXpert, the input was considered healthy when no pathology existed. For BriXIA, an input was considered healthy when all 6 regions were zero.

3.1 Semantics of Thoracic Classification Models

In this section, we analyze the semantics of detectors in classification models trained on CheXpert. Although these models are trained directly on the pathology labels, this does not imply that the individual detectors within the network are detecting features relevant to these pathologies. The models are denoted by CheXpert BCE, and CheXpert Weight. BCE on Fig. 2.

Balanced Training Effect We see that the model trained with weighted BCE has a significantly higher number of concept detectors (100 vs 65). The number of detectors relevant to Pneumonia increases significantly in the weighted model. Pneumonia has a few positive instances in the dataset (0.027%), using a weighted loss causes the emergence of Pneumonia detectors. Consolidation is also imbalanced (0.066%), and we observe that the number of detectors is tripled. In addition, the weighted model shows more variety in terms of concepts than the unweighted version. This is particularly significant as we do not observe any improvement in terms of classification AUC, and also the F1 score. But the model interpretability increases with the weighted loss.

3.2 Semantics of COVID-19 Regression Models

In this section, we examine MSE and MAE SCCE segmentation losses for models trained on the BrixIA dataset. In addition, we investigate the effect of considering imbalance in the data in the MSE loss models. As the number of images in BrixIA is limited, we also investigate the effect of pretraining on the CheXpert dataset. The first observation in Fig. 2 is that training the model on 6 region regression loss (SCCE), results in a higher number of semantic detectors compared to global regression loss. The model has stronger supervision in the former case, and we observe that it also becomes more interpretable.

Pretraining Effect For all training losses, we observe in Fig. 2 that pretraining the model on the CheXpert dataset is significantly increasing the number of individual concept detectors. This increase in the number of detectors is more pronounced in models trained with the global regression score labels. This implies that with weaker supervision with one global regression score, the model struggles in learning individual concepts, and pretraining becomes necessary.

Balanced Training Effect We observe (Fig. 2) that using weighted MSE loss on regression models pre-trained on CheXpert datasets increases the number of unique concept detectors, specifically an increase in the number of detectors associated with Consolidation. For models that are not pretrained on CheXpert, the model struggles in both weighted and unweighted scenarios to learn concepts.

3.3 Semantics of Models Trained on Weakly Labeled Datasets

In this section we discuss models trained on a dataset of healthy / unhealthy labels. The dataset is comprised of CheXpert, NIH ChestX-ray8 and BrixIA.

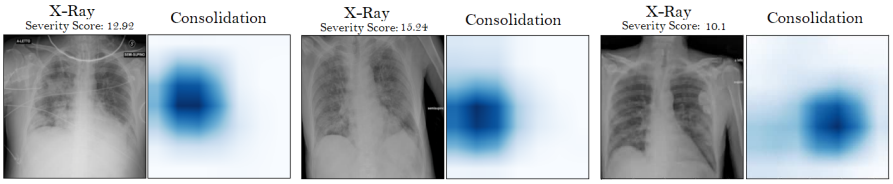


Fig. 5. Semantic attribution: X-rays with severe Covid condition from BrixIA. All have a unit with corresponding semantic annotation of Consolidation as their top contributing unit to the severity score. The contribution of corresponding unit is visualized.

Balanced Training Effect We observe in Fig. 2 that the model that does not use a weighted loss, learns few semantic concepts. Moreover, the majority of individual detectors are related to Cardiomegaly and Cardiomediastinum, and the rest of the concepts are not learned. This implies that Cardiomegaly is an easy signal for the model to pick up during training. However, when we train the model with a weighted loss, we observe that the diversity of the semantic detectors and their numbers increases significantly. Specifically, there were not Consolidation, Pneumonia, Atelectasis detectors in the former case, but they emerge in high numbers in the weighted case. We also observe that the concept detectors indeed emerge in models trained on healthy/unhealthy labels, although it is relatively less than full supervision.

3.4 Evolution of Semantics

In this section, we observe the number/type of individual unit detectors for different epochs. We analyze the BrixIA regression models trained with MAE+SCCE. We observe (Fig. 4) an increase in the number of important concepts for COVID-19 during training (e.g. Consolidation and Ground Glass Opacity) and a decrease in less important concepts (e.g. left lung). The relative increase in the number of concepts of trained model vs. initial model points to the fact that the models are indeed learning patterns associated with pathologies.

3.5 Semantic Attribution

We observe that Consolidation is the most related concept to severe Covid scores as in Fig. 5 on BrixIA dataset. The results in Fig. 5 are for the last conv layer of the model described in Fig. 2 as BrixIA (CheXpert), Weight, MSE. To visualize, the final integrated gradients map of unit k is divided by the maximum $\mathbf{s}_k^i$ of its neurons to normalize and then upsampled to the input image size. Note the methodology here is different from using crude activation maps since we aim to discover the most *contributing* feature maps to the prediction. Highly activated maps do not necessarily correspond to contribution. Hence, we employ an axiomatic approach as explained in Sec. 2.3. Although this is a first step in this direction, more annotations can boost the recognition of other important concepts, and more semantic units for each prediction can be visualized.

4 Conclusion

In this work, we analyze the semantics of internal units of classification and regression models on chest X-rays. We observe that pretraining and considering data imbalance affect the number of semantic detectors. We observe how training on severity score regression labels results in the emergence of pathology-related detectors within the networks. We observe the same phenomenon when the model is trained on weakly labeled datasets. We also propose a semantic attribution method to semantically explain individual predictions.

Acknowledgement This work is partially funded by the **Munich Center for Machine Learning (MCML)** and the **Bavarian Research Foundation** grant AZ-1429-20C. The computational resources for the study are provided by the **Amazon Web Services Diagnostic Development Initiative**. S.T. Kim is supported by the **Korean MSIT**, under the National Program for Excellence in SW (2017-0-00093), supervised by the IITP.

References

1. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PloS one **10**(7), e0130140 (2015)
2. Bau, D., Zhou, B., Khosla, A., Oliva, A., Torralba, A.: Network dissection: Quantifying interpretability of deep visual representations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6541–6549 (2017)
3. Huang, G., Liu, Z., Weinberger, K.Q.: Densely connected convolutional networks. CoRR **abs/1608.06993** (2016), http://arxiv.org/abs/1608.06993
4. Inc, G.B.: Covid-19 chest x-ray segmentations dataset, https://github.com/GeneralBlockchain/covid-19-chest-xray-segmentations-dataset
5. Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R., Shpanskaya, K., et al.: Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 590–597 (2019)
6. Johnson, A.E., Pollard, T.J., Greenbaum, N.R., Lungren, M.P., Deng, C.y., Peng, Y., Lu, Z., Mark, R.G., Berkowitz, S.J., Horng, S.: Mimic-cxr-jpg, a large publicly available database of labeled chest radiographs. arXiv preprint arXiv:1901.07042 (2019)
7. Karim, M., Döhmen, T., Rebholz-Schuhmann, D., Decker, S., Cochez, M., Beyan, O., et al.: Deepcovidexplainer: Explainable covid-19 predictions based on chest x-ray images. arXiv preprint arXiv:2004.04582 (2020)
8. Khakzar, A., Albarqouni, S., Navab, N.: Learning interpretable features via adversarially robust optimization. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 793–800. Springer (2019)
9. Khakzar, A., Baselizadeh, S., Khanduja, S., Rupprecht, C., Kim, S.T., Navab, N.: Neural response interpretation through the lens of critical pathways. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2021)

10. Khakzar, A., Zhang, Y., Mansour, W., Cai, Y., Li, Y., Zhang, Y., Kim, S.T., Navab, N.: Explaining covid-19 and thoracic pathology model predictions by identifying informative input features (2021)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Li, Z., Wang, C., Han, M., Xue, Y., Wei, W., Li, L.J., Fei-Fei, L.: Thoracic disease identification and localization with limited supervision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8290–8299 (2018)
13. Nguyen, A., Dosovitskiy, A., Yosinski, J., Brox, T., Clune, J.: Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In: Advances in neural information processing systems. pp. 3387–3395 (2016)
14. Oh, Y., Park, S., Ye, J.C.: Deep learning covid-19 features on cxr using limited training data sets. IEEE Transactions on Medical Imaging **39**(8), 2688–2700 (2020)
15. Olah, C., Mordvintsev, A., Schubert, L.: Feature visualization. Distill (2017). https://doi.org/10.23915/distill.00007, https://distill.pub/2017/feature-visualization
16. Punn, N.S., Agarwal, S.: Automated diagnosis of covid-19 with limited posteroanterior chest x-ray images using fine-tuned deep neural networks. Applied Intelligence pp. 1–14 (2020)
17. Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., et al.: Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225 (2017)
18. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. pp. 618–626 (2017)
19. Shapley, L.S.: A value for n-person games. Contributions to the Theory of Games **2**(28), 307–317 (1953)
20. Signoroni, A., Savardi, M., Benini, S., Adami, N., Leonardi, R., Gibellini, P., Vaccher, F., Ravanelli, M., Borghesi, A., Maroldi, R., et al.: End-to-end learning for semiquantitative rating of covid-19 severity on chest x-rays. arXiv preprint arXiv:2006.04603 (2020)
21. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
22. Sundararajan, M., Najmi, A.: The many shapley values for model explanation. 37th International Conference on Machine Learning, ICML 2020 (2020)
23. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic attribution for deep networks. In: International Conference on Machine Learning. pp. 3319–3328. PMLR (2017)
24. Taghanaki, S.A., Havaei, M., Berthier, T., Dutil, F., Di Jorio, L., Hamarneh, G., Bengio, Y.: Infomask: Masked variational latent representation to localize chest disease. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 739–747. Springer (2019)
25. Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., Summers, R.M.: Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2097–2106 (2017)
26. Wu, J., Zhou, B., Peck, D., Hsieh, S., Dialani, V., Mackey, L., Patterson, G.: Deepminer: Discovering interpretable representations for mammogram classification and explanation. arXiv preprint arXiv:1805.12323 (2018)

27. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. pp. 818–833. Springer (2014)

Part IV

Conclusion

10 Conclusion and Outlook

The dissertation puts forth the idea that feature attribution remains unsolved, and we need to rethink and revisit existing solutions. If we as a community still do not have a fully reliable tool to explain our neural network models, then our conclusions from the current explanations are questionable. Thus we need to continue our quest for solving the attribution problem.

The dissertation first discusses a multi-faceted approach to viewing feature attribution. It reminds us that thinking about attribution only from the perspective of removing evidence and the Shapley value is short-sighted. Approaches that leverage the network's attention or information flow also show the relevance of features for the output, albeit from a different perspective than that of methods that are based on removing features (such as Shapley). Since the axioms of attribution are also mostly erected based on the idea of removing features, they would favor attributions that are grounded on removing features. The multi-faceted view is also aimed at the evaluation of attributions. The dissertations emphasize the importance of attribution evaluations (that's why we see them before feature attribution literature in the introduction of this dissertation). Each evaluation views the attributions from a different perspective and has its own unique insights. We are required to evaluate attributions from all perspectives and beware of their pitfalls. We need to know what the evaluations tell us about the attributions. And we need to know the properties of each of the attributions to interpret them reliably.

We put the neural networks under the microscope and follow neural pathways and rethink how critical pathways are defined. We see that solving the pruning objective for identifying sparse pathways does not necessarily lead to pathways that encode critical input features. We see simple pruning algorithms can satisfy the pruning objective and deliberately choose irrelevant pathways. The critical pathways are thus hypothesized to be pathways that include critical neurons. The pathways of critical neurons show to have interesting properties. The gradient of the pathways and adversarial perturbations on them correspond to critical input features. Thus critical pathways are leveraged for input feature attribution. We observe that leveraging neurons for attribution plays a significant role in finding critical input features. The hypothesis is that neurons encode interactions and dependencies between input features; therefore, by leveraging neurons, we can implicitly consider them in feature

attribution. Another view is that by considering neurons, we can consider feature groups within attribution. Further work must be done to establish the significance of feature groups in feature attribution, and the findings within our work lay the groundwork.

By considering the predictive information of features as a proxy for their relevance, we look into feature attribution through the lens of information. Again, our intuition that deep features can have a key role in feature attribution comes to play. We first identify deep features with high predictive information and then find their corresponding input features. This is achieved by searching for an input information bottleneck that induces an optimal deep bottleneck. The optimal deep bottleneck refers to one that has the least mutual information with the features and has the highest mutual information with the predicted output. The attribution strategy achieves state-of-the-art results in all empirical evaluation strategies discussed in this dissertation. The strategy is not limited to specific architectures, thus, can be explored for upcoming neural network architectures. Furthermore, the core idea of using deep predictive features to guide the search for predictive input features can be transferred to other attribution strategies.

The dissertation also rethinks the evaluation of feature attribution methods by looking at them through the model's lens. Using the model, we can generate input features, and we can enforce their relevance through generation. Thus, we can have a controlled experimental setup for testing feature attributions. This dissertation proposes the design of setups that test attribution against several axioms and properties: 1) We devise a setup to test whether attributions satisfy the null-feature (dummy) axiom. That is, we check if they attribute to a feature that is guaranteed to be a null feature. 2) We devise a setup that checks whether attributions are class sensitive by generating two features associated with two different classes. 3) We devise feature saturation scenarios and observe how attributions work in these cases. The work can be extended to evaluate other axioms and properties. Some challenges remain for further extension of this work. The optimization of multiple objective functions is not trivial. This dissertation resorts to multi-task learning approaches to balance the optimization terms. More advanced axioms (e.g., completeness) require more complicated optimization objectives. Future work can explore addressing optimization issues. A worthwhile effort due to the simplicity and soundness of the core idea, generating features in tandem with the model.

The dissertation also rethinks attribution for medical image analysis models. The relevance of features can be viewed from different perspectives. We discuss two complementary perspectives: whether the attribution identifies features that are sufficient for a neural network prediction or identifies any feature with predictive information. We discuss how formulating attribution methodologies around deletion/preservation

strategies affects the final outcome. In medical attributions, identifying all predictive features can be beneficial, and the dissertation shows how this can be achieved via formulating the attribution based on the deletion game.

In the last chapter, the works rethink the use of attribution in medical image analysis by moving beyond feature attribution. Feature attribution is merely one way to explain neural networks. In the final chapter, we check neuron activations and identify their corresponding concepts (thoracic pathologies). Through this lens, we explore the following questions: Do neural networks trained as regression models on Covid-19 severity scores learn pathologies without any explicit cues? We observe that concept detectors indeed emerge inside networks during training. Do neural networks trained on binary datasets labeled as healthy/unhealthy learn pathologies? We again observe that concept detectors emerge during training and considering data imbalance during training significantly affects the number of concept detectors within the network. Analyzing the concepts encoded by the network can be extended to other medical applications (as we did for vertebrae fracture diagnosis [112]). The chapter offers a glimpse of moving beyond attribution toward analysis of concepts.

The dissertation rethinks feature attribution and looks at the problem through different lenses. However, the attribution problem remains open for exploration. Nevertheless, the ideas proposed in the dissertation move us one step closer to solving the feature attribution problem and neural network explanation.

Part V

Appendix

A

Through the Lens of Neural Pathways (Appendix)

A. Proofs

A.1. Proof of Lemma 1

Lemma 1 (Dead Neurons) *Considering $\mathbf{a}^i$ as the input at layer i to the following layers of the network defined by function $\Phi_\theta^{>i}(.) : \mathbb{R}^{N_i} \to \mathbb{R}$, the Shapley value of a neuron $\mathbf{a}_j^i$ defined by $\sum_{C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i} \frac{|C|!(N_i-|C|-1)!}{N_i} (\Phi_\theta^{>i}(C \cup \mathbf{a}_j^i) - \Phi_\theta^{>i}(C))$ is zero if the neuron is dead ($\mathbf{a}_j^i = 0$).*

For any layer i, the Shapley value (with baseline zero) of a neuron $\mathbf{a}_j^i$ is defined as:

$$\sum_{C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i} \frac{|C|!(N_i - |C| - 1)!}{N_i} (\Phi_\theta^{>i}(C \cup \mathbf{a}_j^i) - \Phi_\theta^{>i}(C)) \ , \tag{8}$$

where $\Phi_\theta^{>i}$ denotes the neural function after layer i. The input to $\Phi_\theta^{>i}$ is the activation vector $\mathbf{a}^i$. We need to show that for all $\mathbf{a}_j^i$ and all possible coalitions $C \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$:

$$\Phi_\theta(C \cup \mathbf{a}_j^i; \mathbf{x}) = \Phi_\theta(C; \mathbf{x}) \ . \tag{9}$$

We know for any $\mathbf{a}^i$ the outputs of neurons in the next layer are:

$$\mathbf{z}^{i+1} =^{i+1} \mathbf{a}^i + \mathbf{b}^{i+1} \ . \tag{10}$$

As the baseline is considered zero, ablating a neuron $\mathbf{a}_j^i$ is done by $\mathbf{a}_j^i = 0$. Thus $\mathbf{z}^{i+1}$ does not change by ablation of $\mathbf{a}_j^i$ for any coalition C. As $\mathbf{z}^{i+1}$ does not change, Φ_θ does not change, thus we get $\Phi_\theta(C \cup \mathbf{a}_j^i; \mathbf{x}) = \Phi_\theta(C; \mathbf{x})$.

A.2. Proof of Proposition 4

Proposition 4 *In a* ReLU *rectified neural network with $\Phi(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$, for a path defined by $[\mathbf{e}_j^i]^N$, if $\mathbf{a}_j^i > 0 \ \forall \ \mathbf{e}_j^i = 1$, then there exists a linear region $\hat{\epsilon}_{\mathbf{x},2} > 0$ for $\hat{\Phi}(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$.*

The linear region, $\hat{\epsilon}_{\mathbf{x},2}$ is the largest ℓ_2-ball around $\mathbf{x}$ where the $\mathcal{AP}$ is fixed, *i.e.*

$$\hat{\epsilon}_{\mathbf{x},2} \doteq \max_{\epsilon \geq 0 : \mathcal{B}_{\epsilon,2}(\mathbf{x}) \subseteq S(\mathbf{x})} \epsilon \tag{11}$$

$\hat{\epsilon}_{\mathbf{x},2}$ is the minimum ℓ_2 distance between $\mathbf{x}$ and the corresponding hyperplanes of all neurons $\mathbf{z}_j^i$ [28]. In Section 4.1 we discuss that the distance is goverened by neurons for which $\nabla_{\mathbf{x}} \mathbf{z}_j^i \neq 0$. We have [28] $\hat{\epsilon}_{\mathbf{x},2} = \min_{i,j} |\mathbf{z}_j^i| / ||\nabla_{\mathbf{x}} \mathbf{z}_j^i||_2$. Thus, the existence of a linear region $\hat{\epsilon}_{\mathbf{x},2}$ depends on $|\mathbf{z}_j^i|$ not being equal to zero. We are selecting a path $[\mathbf{e}_j^i]^N$ where for each neuron $\mathbf{a}_j^i > 0$ and thus we have $\mathbf{z}_j^i > 0$. If we replace every neuron not on the path with a constant value equivalent to the original value of the activation of that neuron, the activation pattern $\mathcal{AP}$ remains constant, and thus we get a new approximate neural network $\hat{\Phi}(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$, where all neurons $\mathbf{z}_j^i > 0$. Therefore $\hat{\epsilon}_{\mathbf{x},2} \neq 0$ and there exists a linear region.

A.3. Proof of Proposition 5

Proposition 5 *Using NeuronIntGrad and NeuronMCT, if $\mathbf{c}_\kappa > 0$, then $\hat{\Phi}(\mathbf{x}; \mathbf{e})$ at $\mathbf{x}$ is locally linear.*

For NeuronMCT and NeuronIntGrad the contributions are assigned by:

$$\mathbf{c}_j^i = |\Phi(\mathbf{x}) - \Phi(\mathbf{x}; \mathbf{a}_j^i 0)| = |\mathbf{a}_j^i \nabla_{\mathbf{a}_j^i} \Phi_\theta(\mathbf{x})| \tag{12}$$

and

$$\mathbf{c}_j^i = \mathbf{a}_j^i \int_{\alpha=0}^{1} \frac{\partial \Phi(\alpha \mathbf{a}_j^i; \mathbf{x})}{\partial \mathbf{a}_j^i} \mathrm{d}\alpha \tag{13}$$

respectively. It is evident that if $\mathbf{c}_j^i \neq 0$ then $\mathbf{a}_j^i \neq 0$. Therefore a path selected by $|\mathbf{c}_j^i| > 0$ we have all $\mathbf{a}_j^i > 0$. Hence according to Prop. 4 the selected paths and the approximate $\hat{\Phi}(\mathbf{x}; \mathbf{e})$ is locally linear.

A.4. Proofs for axioms of marginal contribution

Defining marginal contribution of neuron $\mathbf{a}_j^i$ at layer i as:

$$\mathbf{c}_j^i = \Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i}) - \Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i) \tag{14}$$

A.4.1 Null player

The null player axiom asserts that if a neuron is a null player, *i.e.*

$$\Phi^{>i}(S \cup \mathbf{a}_j^i) = \Phi^{>i}(S) \ , \tag{15}$$

for all $S \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$, then $\mathbf{c}_j^i$ must be zero.

Eq. 15 is assumed for all S, therefore by substituting $S = \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i$, in Eq. 15 we get:

$$\Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i}) = \Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i) \ , \tag{16}$$

which results in $\mathbf{c}_j^i = 0$.

A.4.2 Symmetry

The symmetry axiom asserts for all $S \subseteq \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \{\mathbf{a}_j^i, \mathbf{a}_k^i\}$ if

$$\Phi^{>i}(S \cup \mathbf{a}_j^i) = \Phi^{>i}(S \cup \mathbf{a}_k^i) \tag{17}$$

holds, then $\mathbf{c}_k^i = \mathbf{c}_j^i$.

Eq. 17 is assumed for all S, therefore by substituting $S = \{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \{\mathbf{a}_j^i, \mathbf{a}_k^i\}$, in Eq. 15 we have:

$$\Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_j^i) = \Phi^{>i}(\{\mathbf{a}_j^i\}_{j=1}^{N_i} \setminus \mathbf{a}_k^i) \ . \tag{18}$$

By substituting into Eq. 14, we obtain $\mathbf{c}_k^i = \mathbf{c}_j^i$.

B. Further Discussions

B.1. Computing contribution of neurons vs. pixels

If we compute the marginal contribution or Shapley value for a single feature of the input, *e.g.* a pixel, the distributional interdependencies, and correlations between the pixels are not considered. This is not to be confused with the interdependency that the Shapley value considers by taking different coalitions into account. For instance, ablating a single pixel from an object in an image does not affect the score of an Oracle classifier, in any coalition. One must consider that all the pixels are related and exist in one object when computing the marginal contribution and Shapley value for the object (all pixels considered as one feature). Shapley value of a set of features is known as generalized Shapley value [32]. We can observe a consequence of this phenomenon, in the different results obtained by [2] when removing single pixels (occlusion-1) or removing patches, where the latter results in more semantic attribution maps. Several works implicitly consider such correlations by masking a group of pixels. The question is what mask should we look for, as the prior information about the dependency of pixels is not available. There are 2^N possible masks that one can select. Moreover, a larger mask containing a feature might get the same or higher contribution score as the mask of the feature. Therefore in [11, 10] priors such as the size of the mask are used. These methods look for the smallest masks with the highest contribution. In the regime of neural networks, we encounter more problems with mask selection. If we do not enforce any prior, we can get adversarial masks [11, 15]. Therefore, several works[11, 10] enforce priors such as smoothness of the masks. On the other hand, if we use the prior encoded in the network (which is learned from the distribution of the data), we implicitly consider the group of pixels that are correlated with each other. Thus by computing the contribution of individual neurons, we are considering a complex group of pixels and their distributional relationships.

C. Implementation details

The sparsity level of ResNet-50 is 70% and VGG-16 is 90% in the experiments, unless stated otherwise.

C.1. Network parameter randomization sanity check [1]

All attribution methods are run on ResNet50 (PyTorch pretrained) and on 1k ImageNet images. The acquired attribution maps from all methods are normalized to [-1 1] as stated by [1]. The layers are randomized from a normal distribution with mean=0 and variance=0.01 in a cascading manner from the last to the first. After the randomization of each layer, the similarity metrics (SSIM and Spearman Rank Correlation) are calculated between the map from the new randomized model and the original pretrained network. Methods that are not sensitive to network parameters (like GBP) would hence lead to high levels of similarity between maps from normalized networks and the original map.

C.2. Input degradation - LeRF [49]

We report results on CIFAR using a custom ResNet8 (three residual blocks), Birdsnap using ResNet-50, and ImageNet (validation set) using ResNet-50. We show the absolute fractional change of the output as we remove the least important pixels. Lower curves mean higher specificity of the methods. Note that, for NeuronMCT and NeuronIntGrad, the pixel perturbation process is performed on the original model not on the critical paths selected by these methods. The critical paths are only used to obtain the attribution maps and not after.

C.3. Remove and Retrain (ROAR) [21]

We perform the experiments with top 30; 50; 70; 90 of pixels perturbed. The model is retrained for each attribution method (8 methods) on each percentile (5 percentiles) 3 times. Due to the large number of retraining sessions required, we cannot report this benchmark on other datasets. We evaluate this benchmark on CIFAR-10 (60k images, 32x32) with a ResNet-8 (three residual blocks).

D. Supplementary results

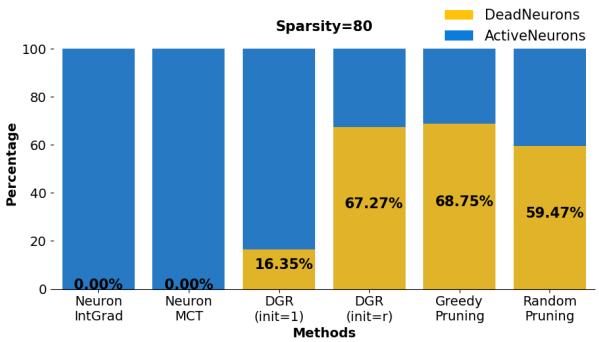


Figure 8. **Dead Neuron Selection of Pruning Objective (Sparsity %80).** The percentage of originally dead neurons in the selected paths of different methods reported for sparsity of 80%. All paths selected by pruning objective contain originally dead (now active) neurons

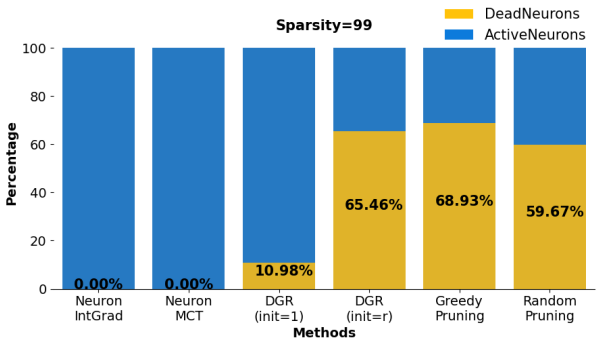


Figure 9. **Dead Neuron Selection of Pruning Objective (Sparsity %99).** The percentage of originally dead neurons in the selected paths of different methods reported for sparsity of 99%. All paths selected by pruning objective contain originally dead (now active) neurons

Table 1. **ROAR**: AUCs reported for each attribution method. The lower the AUC the better.

	GRADIENT	GBP	GRADCAM	INPUTMCT	INPUTINTGRAD	NEURONMCT	NEURONINTGRAD	NEURONINTGRAD*
CIFAR-10	0.728	0.702	0.584	0.723	0.741	0.580	0.574	**0.524**
BIRDSNAP	0.269	0.243	0.096	0.242	0.242	0.117	0.099	**0.090**

Table 2. **Input degredation (LeRF)**: AUCs reported for each attribution method. The lower the AUC the better.

	GRADIENT	GBP	GRADCAM	INPUTMCT	INPUTINTGRAD	NEURONMCT	NEURONINTGRAD	NEURONINTGRAD*
CIFAR-10	0.037	0.028	0.010	0.019	0.019	0.009	0.009	**0.007**
BIRDSNAP	0.090	0.085	0.012	0.090	0.090	0.010	0.010	**0.008**
IMAGENET	0.046	0.044	0.009	0.043	0.041	0.010	0.010	**0.009**

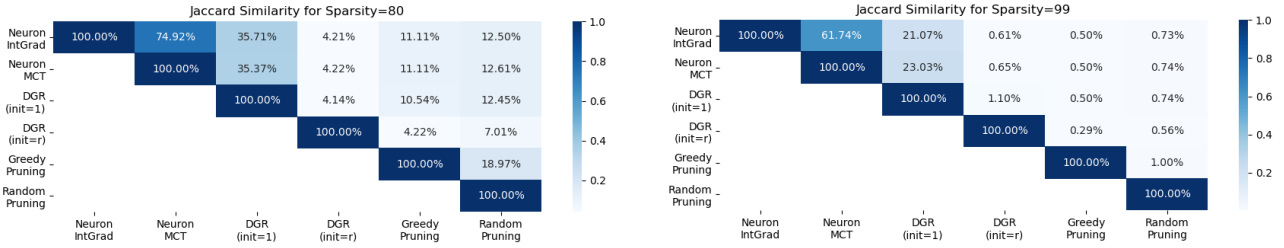


Figure 10. **Path Analysis - Entire Network (Sparsity 80 & 99).** Overlap between paths from different methods in entire network. Among the pruning-based methods, only the path selected by DGR(init=1) overlaps with contribution-based methods.

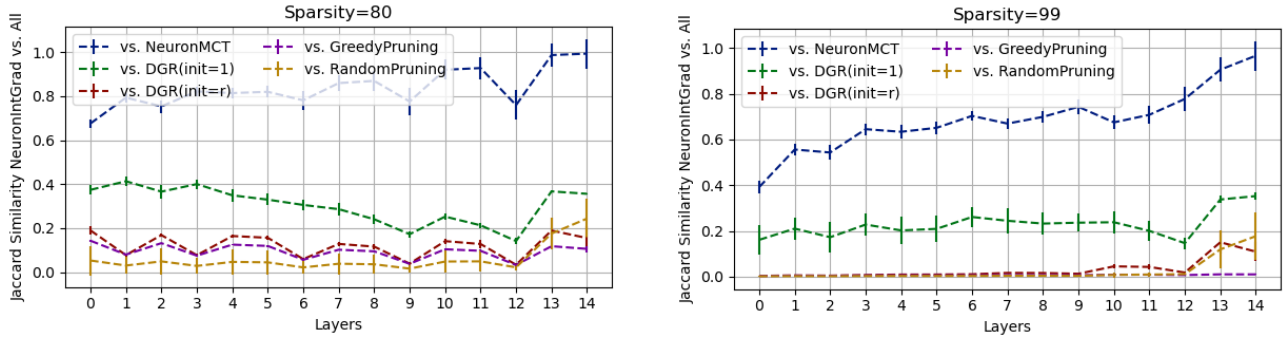


Figure 11. **Path Analysis - Layerwise (Sparsity 80 & 99).** Overlap between paths from different methods in different layers of VGG-16. Among the pruning-based methods, only the path selected by DGR(init=1) overlaps with NeuronIntGrad.

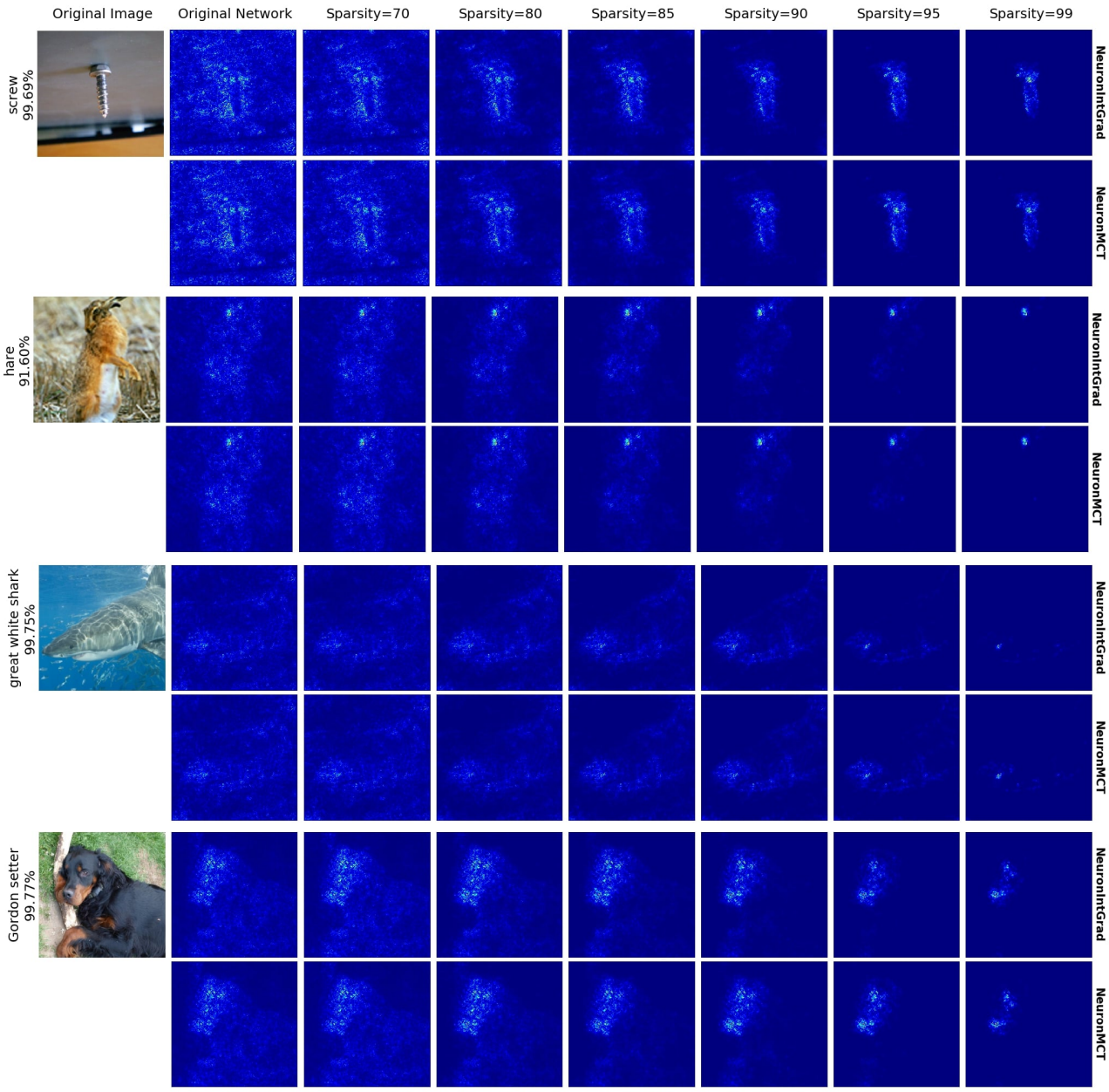


Figure 12. **Feature Attribution.** The gradients of the locally linear critical paths at different sparsity levels for NeuronIntGrad (top) and NeuronMCT (bottom).

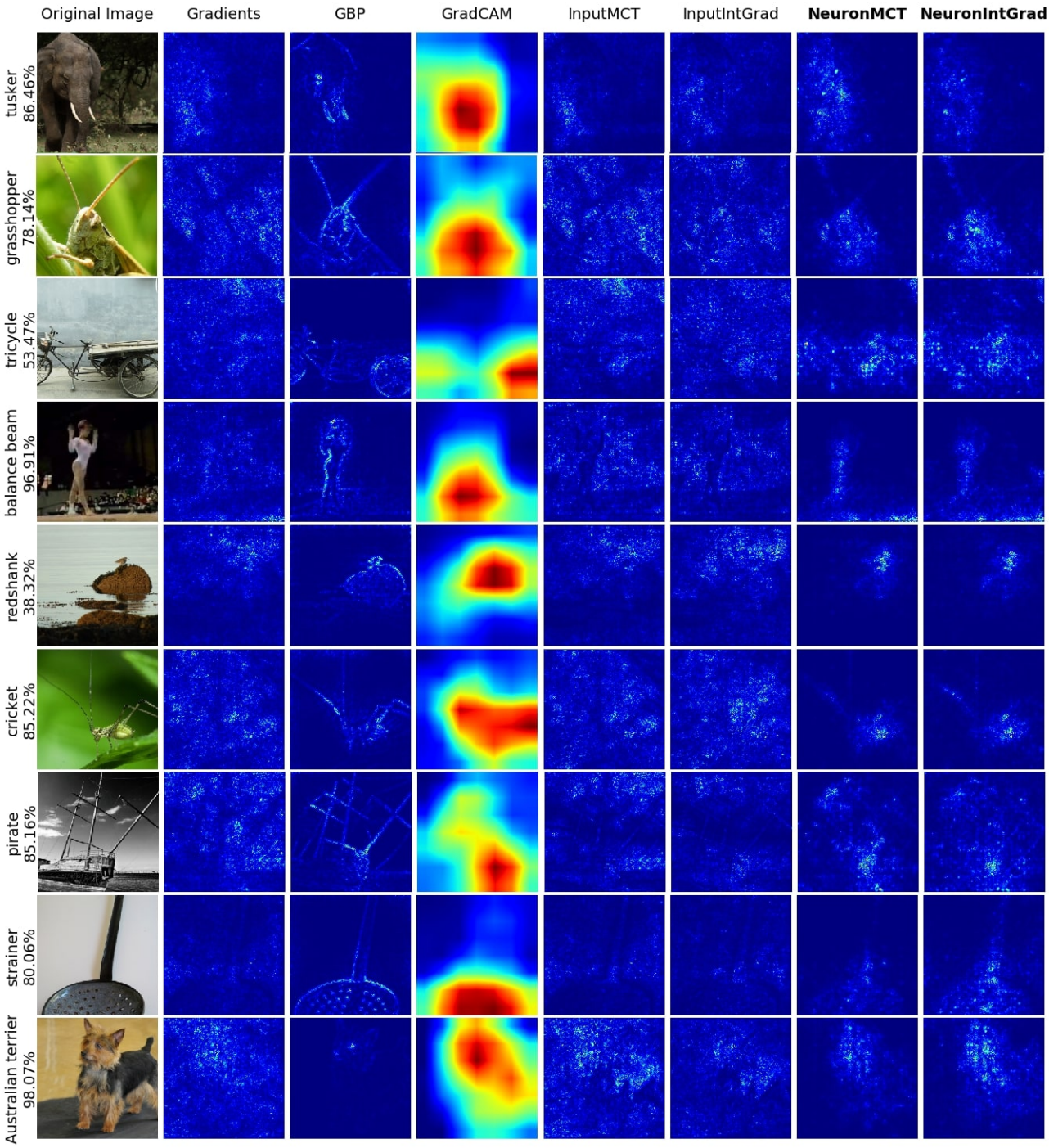


Figure 13. **Comparison with Feature Attribution Methods.** Comparison between attribution maps dervied our proposed methods (right) vs. gradient-based attribution methods on **ResNet-50**. Note the improvement of integrated gradients on the neurons (NeuronIntGrad) over integrated gradients on input (InputIntGrad).

Figure 14. **Comparison with Feature Attribution Methods.** Comparison between attribution maps dervied our proposed methods (right) vs. gradient-based attribution methods on **VGG-16**. Note the improvement of integrated gradients on the neurons (NeuronIntGrad) over integrated gradients on input (InputIntGrad).

B

Through the Lens of Information (Appendix)

A Computing Mutual Information $I[Z_I, I]$

For two random variables Z_I and I, where Z_I is dependent on I, we can calculate the mutual information via KL-divergence:

$$I(Z_I, I) = D_{KL}(P(Z_I|I)||P(Z_I)) \tag{5}$$

KL-divergence is defined as

$$D_{KL}(P, Q) = -\int_x P(x) \log \frac{Q(x)}{P(x)} dx \tag{6}$$

which means for calculating KL-divergence we need to integrate over the entire probability space. However, Given two random variables $P \sim \mathcal{N}(\mu_1, \sigma_1)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2)$ that are Gaussian distributed, the KL-divergence between this two random variables can be calculated in closed form:

$$D_{KL}(P, Q) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{7}$$

The bottleneck is constructed by masking the input $Z_I = \Lambda I + (1 - \Lambda)\epsilon_I$, where ϵ_I is the input noise and $\epsilon_I \sim \mathcal{N}(\mu_I, \sigma_I)$. The distribution of Z_I given input I, $P(Z_I|I)$, is thus (Remark 2):

$$P(Z_I|I) \sim \mathcal{N}(\Lambda I + (1 - \Lambda)\mu_I, (1 - \Lambda)^2 \sigma_I^2) \tag{8}$$

As explained in Section 3.2, Z_I is conditioned on Z_G by using $Z_I = \Lambda Z_G + (1 - \Lambda)\epsilon$, and $Z_G = \lambda_G I + (1 - \lambda_G)\epsilon_G$, by substitution we have:

$$P(Z_I) \sim \mathcal{N}(\lambda_G \Lambda I + (1 - \lambda_G \Lambda)\mu_I, (1 - \lambda_G \Lambda)^2 \sigma_I^2) \tag{9}$$

We are implicitly introducing an independence assumption for the random variable $P(Z_I)$ through our Gaussian masking ($Z_I = \Lambda Z_G + (1-\Lambda)\epsilon$) formulation of $P(Z_I)$. Where each elements of $P(Z_I)$ are constructed by an independent Gaussian masking. A better approximation would not impose such an independence assumption. Please note that in original IBA [19] the same independence assumption exists for $P(Z)$. Since KL-divergence is additive for independent distributions ($D_{KL}(P||Q) = \sum_k D_{KL}(P_k||Q_k)$), we can calculate the KL-divergence of $P(Z_I|I)$ and $P(Z)$ by summing over the KL-Divergence of there elements. Therefore (proposition 3):

$$\begin{aligned}
&D_{KL}[P(Z_{I,k}|I_k)||P(Z_{I,k})] \\
=&\log \frac{(1 - \lambda_{G,k}\Lambda_k)\sigma_{I,k}}{(1 - \Lambda_k)\sigma_{I,k}} + \frac{(1 - \Lambda_k)^2 \sigma_{I,k}^2}{2(1 - \lambda_{G,k}\Lambda_k)^2 \sigma_{I,k}^2} \\
&+ \frac{((\Lambda_k I_k + (1 - \Lambda_k)\mu_{I,k}) - (\lambda_{G,k}\Lambda_k I_k + (1 - \lambda_{G,k}\Lambda_k)\mu_{I,k}))^2}{2(1 - \lambda_{G,k}\Lambda_k)^2 \sigma_{I,k}^2} - \frac{1}{2} \\
=&\log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} \\
&+ \frac{(\Lambda_k I_k + \mu_{I,k} - \Lambda_k \mu_{I,k} - \lambda_{G,k}\Lambda_k I_k - \mu_{I,k} + \lambda_{G,k}\Lambda_k \mu_{I,k})^2}{2(1 - \lambda_{G,k}\Lambda_k)^2 \sigma_{I,k}^2} - \frac{1}{2} \\
=&\log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} + \frac{(\Lambda_k I_k - \Lambda_k \mu_{I,k} - \lambda_{G,k}\Lambda_k I_k + \lambda_{G,k}\Lambda_k \mu_{I,k})^2}{2(1 - \lambda_{G,k}\Lambda_k)^2 \sigma_{I,k}^2} - \frac{1}{2} \\
=&\log \frac{1 - \lambda_{G,k}\Lambda_k}{1 - \Lambda_k} + \frac{(1 - \Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2} + \frac{(I_k - \mu_{I,k})^2 (\Lambda_k - \lambda_{G,k}\Lambda_k)^2}{2(1 - \lambda_{G,k}\Lambda_k)^2 \sigma_{I,k}^2} - \frac{1}{2}
\end{aligned} \tag{10}$$

B Derivation of Proposition 4

The derivation is inspired by [40]. Contrary to [40], in this work we derive the exact representation of $I(Z, Y)$ instead of a lower bound. Given X as input, Y as output, Z as bottleneck (masked input), we

assume $Y \leftrightarrow X \leftrightarrow Z$, this means that Y and Z are independent given X. This assumption is fulfilled by masking scheme and data generation process: for Z we have $Z = m(X, \epsilon, \lambda)$ (where function m represents the masking scheme), thus Z depends on X given sampled noise and mask; for Y we have $Y = g(X)$, where function g represents the data generation process for the task. Therefore:

$$p(x, y, z) = p(y|x, z)p(z|x)p(x) = p(y|x)p(z|x)p(x) \tag{11}$$

Thus we can also calculate $p(y|z)$ by:

$$p(y|z) = \frac{p(z, y)}{p(z)} = \int \frac{p(y|x)p(z|x)p(x)}{p(z)} dx \tag{12}$$

And mutual information between Z and Y:

$$\begin{aligned} I(Z, Y) &= \int p(y, z) \log \frac{p(y, z)}{p(y)p(z)} dydz \\ &= \int p(y, z) \log \frac{p(y|z)}{p(y)} dydz \end{aligned} \tag{13}$$

However, calculating $p(y|z)$ requires integral over x, which is intractable. We then apply the variational idea to approximate $p(y|z)$ by $q_\theta(y|z)$ instead. $q_\theta(y|z)$ represents the neural network part after bottleneck, θ indicates parameter of the neural network part.

$$\begin{aligned} I(Z, Y) &= \int p(y, z) \log \frac{p(y|z)}{p(y)} \frac{q_\theta(y|z)}{q_\theta(y|z)} dydz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(y, z) \log \frac{p(y|z)}{q_\theta(y|z)} dydz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(z) (\int p(y|z) D_{KL}[p(y|z)||q_\theta(y|z)]dy)dz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \int p(z) D_{KL}[p(y|z)||q_\theta(y|z)]dz \\ &= \int p(y, z) \log \frac{q_\theta(y|z)}{p(y)} dydz + \mathbb{E}_{z\sim p(z)}[D_{KL}[p(y|z)||q_\theta(y|z)]] \end{aligned} \tag{14}$$

C Derivation of Theorem 5

Proof: Here we want to prove that for per-sample Information Bottleneck under classification task, optimizer of cross entropy loss is also the optimal value for mutual information $I[Y, Z]$. Previous work concludes that the optimizer of $\min \mathbb{E}_{\epsilon\sim p(\epsilon)}[-\log q_\theta(y_{sample}|z)]$ is a lower bound of $I(Y, Z)$ [40]. To ease the effort of readers searching across literatures, we summarize the proof in [40] in Appendix G. We now prove this optimizer is not only the lower bound of $I(Y, Z)$, but also the optimizer for $I(Y, Z)$ under per-sample setting (local explanation setting). Now we consider the second term in Eq. (14) which we neglected during mutual information calculation:

$$\mathbb{E}_{z\sim p(z)}[D_{KL}[p(y|z)||q_\theta(y|z)]] \tag{15}$$

This term equals to zero if $p(y|z) \equiv q_\theta(y|z)$. The local explainable set contains a batch of neighbours of X_{sample}. We make two assumptions on the data points in the local explainable set. One is all data points in the local explainable set have the same label distribution (for an Oracle classifier), as data points are only slightly perturbed from X_{sample}. Another one is that distribution of Z can be considered as equivalent inside local explainable set, i.e. the latent features are equivalent for samples that are in the local explanation set of X (we know this is not true for adversarial samples, however for many samples in the neighborhood which have the same label with X this is true). Now

we approximate $p(y|z)$ by:

$$\begin{aligned} p(y|z) &= \frac{1}{N}\sum_{n=1}^{N}\frac{p(y|x_n)p(z|x_n)}{p(z)} \qquad \text{(local set of sample)} \\ &= \frac{1}{N}\sum_{n=1}^{N}\frac{p(y|x_{sample}+\epsilon)p(z|x_{sample}+\epsilon)}{p(z)} \\ &= \frac{p(y|x_{sample})p(z|x_{sample})}{p(z)} \qquad \text{(use two assumptions above)} \\ &= p(y|x_{sample}) \end{aligned} \tag{16}$$

$p(z|x_{sample}) \equiv p(z)$ because we consider a local dataset, and use the second assumption that Z is equivalent given data in a local set:

$$p(z) = \int p(z|x)p(x)dx \approx \frac{1}{N}\sum_{n=1}^{N}p(z|x_n) = \frac{1}{N}\sum_{n=1}^{N}p(z|x_{sample}+\epsilon) = p(z|x_{sample}) \tag{17}$$

Now $p(y|z) \equiv q_\theta(y|z)$ can be easily proved, for $p(y|z)$:

$$p(y|z) = p(y|x_{sample}) = \begin{cases} 1, & \text{if } y = y_{sample} \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

For $q_\theta(y|z)$, the optimizer maximize $\mathbb{E}_\epsilon[log(q_\theta(y_{sample}|z))]$. Then we have:

$$q_\theta(y|z) = \begin{cases} 1, & \text{if } y = y_{sample} \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

Thus they are equivalent, and the KL divergence is zero. As a result, we can remove the inequality in variational step under assumption that we are generating attribution per sample, and assuming local explanation.

D Details and Hyper-parameters of Experiment Setup

D.1 Hyper-parameters and Implementation of Attribution Methods

Most hyper-parameters used for ImageNet experiments are presented in the body of the text. We would like to supplement the setting of generative model. The generative adversarial model for estimating Z_G uses a discriminator with 3 CNN layers followed by 2 fully connected layers, and contains 200 samples in target dataset. During training, we update the discriminator's parameter after every 5 generator updates. To attribute the 4-layer LSTM model trained on IMDB dataset, we insert a bottleneck after the last LSTM layer, the parameter β at this bottleneck is 15. The learning rate for this bottleneck at hidden layer is 5×10^{-5}. The generative adversarial model uses a single Layer RNN as discriminator. For the input bottleneck at embedding space, we use $\beta_{in} = 30$, $lr = 0.5$, and *optimization step* $= 30$.

For Extremal Perturbations, size of the perturbation mask is a hyper-parameter. We set the mask size to be 10% of the image size for EHR experiment, as all images for EHR have bounding box covering less than 33% of the image. For the rest of evaluations of Extremal Perturbation, we use default mask size implemented in public code framework.

Regarding the parameters of Integrated Gradients (IG), we use the default values (proposed in the original paper [2]). The number of integrated points is 50 and the baseline value is 0. We will add the details in the appendix. These values are commonly used and we observe that the method with default parameters performs well in the quantitative experiment (Fig. 5b) as explained.

D.2 Hyper-parameters of InputIBA

In Fig. 7 we observe the effect of β in InputIBA optimization. Similar to IBA [19] it controls the ammount of information passing through the bottleneck. Fig. 8 shows the effect of number of optimization steps in InputIBA. Fig. 9 shows how the attribution map varies while the number of optimizations steps of the generative adversarial model changes.

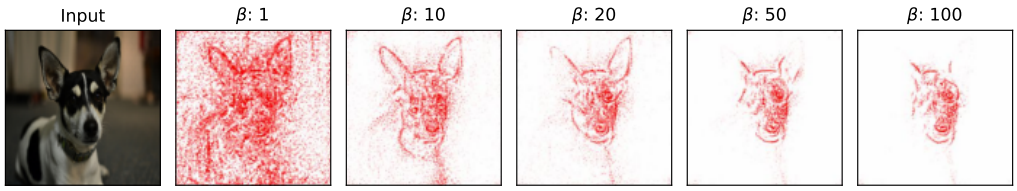


Figure 7: Influence of β of the InputIBA.

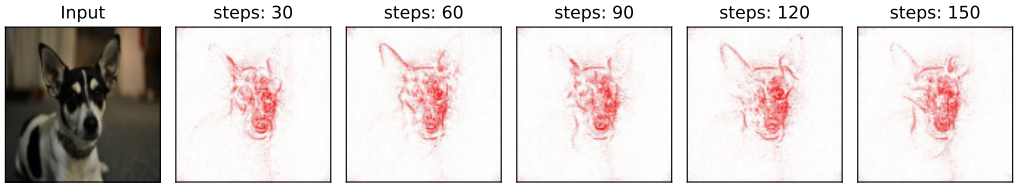


Figure 8: Influence of optimization steps of the InputIBA.

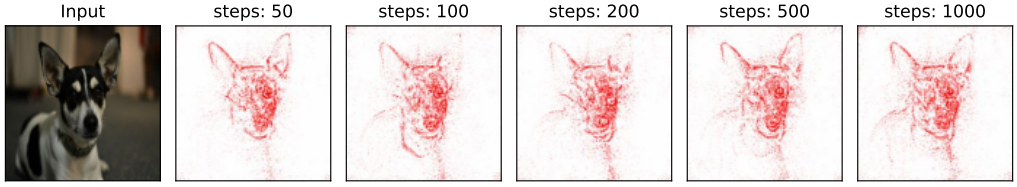


Figure 9: Influence of optimization steps of the generative adversarial model.

Method	Insertion AUC	Deletion AUC
InputIBA	0.710 ± 0.005	0.045 ± 0.001
IBA	0.713 ± 0.004	0.090 ± 0.002
GradCAM	0.703 ± 0.005	0.133 ± 0.003
Guided BP	0.529 ± 0.004	0.132 ± 0.002
Extremal Perturbation	0.676 ± 0.004	0.135 ± 0.003
DeepSHAP	0.430 ± 0.004	0.196 ± 0.003
Integrated Gradients	0.358 ± 0.004	0.210 ± 0.003

Table 3: Mean and the standard error of insertion/deletion AUC for ImageNet dataset. We show the statistical information with standard error in the table.

D.3 Insertion/Deletion

Instead of inserting or deleting one single pixel/token and report the model prediction on modified input at each step, we apply batch-wise pixel/token insertion and deletion. One advantage of batch-wise modification is that the evaluation is accelerated by leveraging the batch processing ability of existing deep learning framework. Also batch-wise modification on input stabilizes the output, as perturb only one pixel or token cannot change the overall information of the input effectively, thus introduces strong deviation on model prediction. On both ImageNet and IMDB dataset, we set the batch size to be 10. For images in ImageNet, we delete pixel by replacing it with black pixel. In insertion test, we insert pixel on a blurred image rather than a black canvas, since insert pixel on black canvas generates stronger adversarial effect than in deletion test. We generate the blurred image by applying Gaussian blur with kernel size equal to 29, standard deviation for Gaussian kernel is 15. For texts in IMDB dataset, both insertion and deletion are performed by replacing the token in text with <unk> token. Insertion/Deletion results on ImageNet dataset and their corresponding standard deviation are presented in Table 3.

D.4 Remove-and-Retrain (ROAR)

In Remove-and-Retrain (ROAR), we divide CIFAR10 dataset into train set, validation set and test set. In training phase, a classifier is trained on the train set with 30 training epochs. After training is complete, we apply the attribution method on trained model for all images in 3 subsets. In retrain phase, for each attribution method, we perturb all 3 subsets based on attribution maps. As a result, we generate 9 perturbed dataset using different perturbation rate (ranging from 10% to 90%, with 10% step). A model is trained from scratch on perturbed train set with 30 training epochs, and we report the final performance of this model on perturbed test set.

E Qualitative Results of Attribution Methods

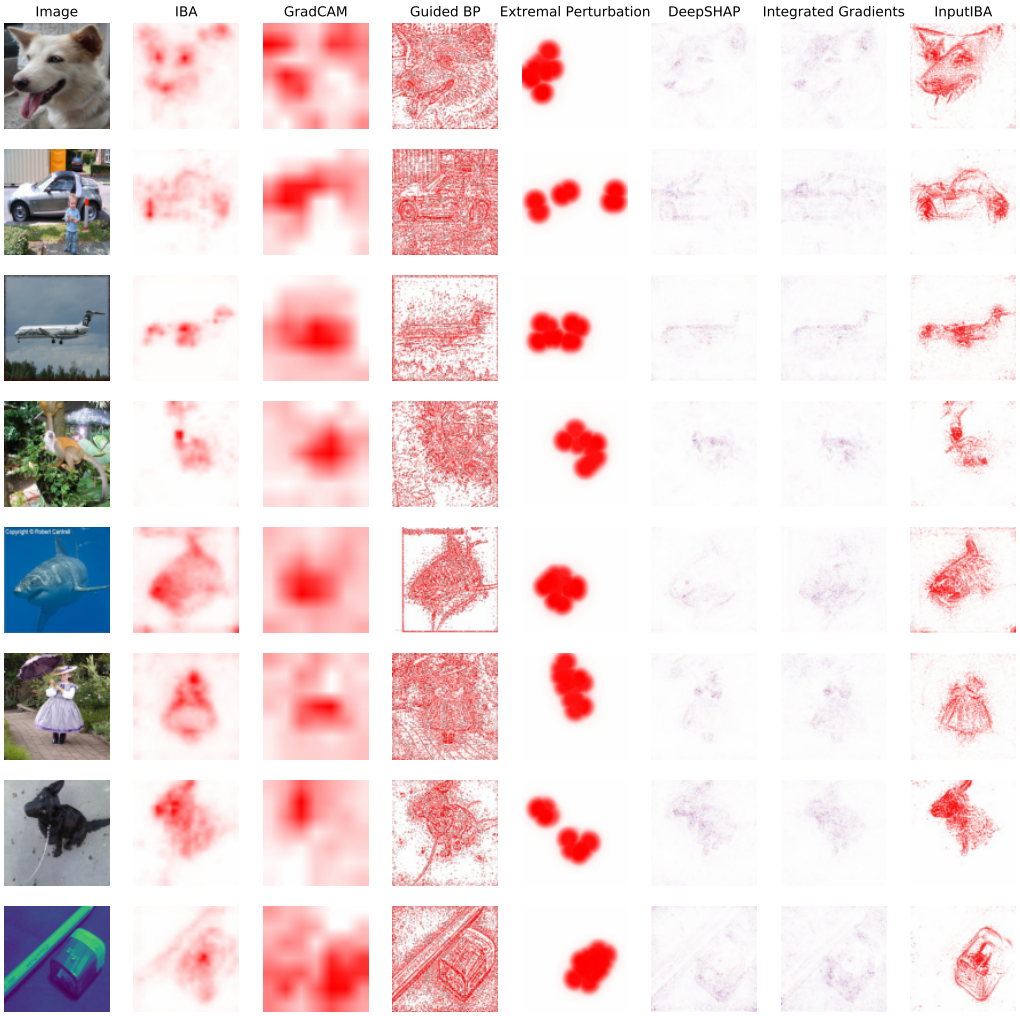


Figure 10: **Qualitative Comparison (ImageNet)**: We conduct qualitative comparision between attribution methods on more sample images form ImageNet validation set.

F The case for EHR (vs. BBox metric in IBA [15])

Here we explain the limitations of *bbox* evaluation metric used in the [15] with three synthetic examples. As illustrated in Fig. 11, we assume an object covers 25% area of an image. We also assume three attribution methods and their attribution maps. As illustrated in Fig. 11a, method (a)

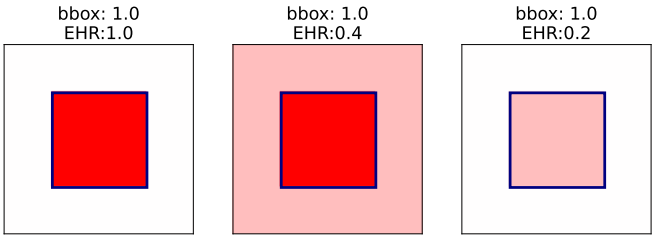


Figure 11: **Synthetic Examples**: We synthesize three attribution maps of an imaginary image. The object is surrounded by a bounding box, which is annotated with a blue box. Values of the *bbox* metric and EHR are also shown on the top of each figure. (a): Attribution scores within the bounding box are 1.0 (the maximal value), outside the bounding box are 0 (the minimal value). (b): Attribution scores within the bounding box are 1.0, outside the bounding box are 0.25. (c) Attribution scores within the bounding box are 0.25, outside the bounding box are 0.

only highlights the pixels within the bounding boxes. In this case, both the *bbox* metric and EHR are 1.0. In Fig. 11b, method (b) additionally highlights the region outside the bounding box, the scores outside the bounding box is lower than that within the bounding box. Thus, the *bbox* metric remains 1.0. However, method (b) is sub-optimal to method (a) as it erroneously highlights the non-object pixels. In Fig. 11c, method (c) only highlights the pixels within the bounding box, but with lower scores. Thus, method (c) is also considered to be sub-optimal to method (a). The *bbox* metric in [15] cannot distinguish these three cases, while EHR can: the EHR of method (b) and (c) are substantially lower than that of method (a).

G Optimization of $I[Y, Z]$[40]

For easier reference, we provide a summary of derivation of the lower bound of the mutual information $I(Z, Y)$ as done in [40]:

$$\begin{aligned} I(Z,Y) &\geq \int p(y,z) \log \frac{q_\theta(y|z)}{p(y)} dydz \\ &= \int p(y,z) \log q_\theta(y|z) dydz + H(Y) \end{aligned} \tag{20}$$

The second term $H(Y)$ is a constant and has no effect when optimize over mask. We can now further construct our optimization objective as

$$\max_{\lambda} OE(Z,Y) \tag{21}$$

To get the final result, we first expand $p(y, z)$ as $\int p(x, y, z)dx$, and reformulate $p(x, y, z)$ based on Eq. (11), then we construct an empirical data distribution

$$p(x,y) = \frac{1}{N} \sum_{n=1}^{N} \sigma_{x_n}(x) \sigma_{y_n}(y) \tag{22}$$

Lastly, we use reparameterization trick since $Z = f(x, \epsilon)$, and x is not a random variable.

$$p(z|x)dz = p(\epsilon)d\epsilon \tag{23}$$

Combine all tricks together:

$$\begin{aligned} OE(Z,Y) &= \int p(y,z)\log q_\theta(y|z)dydz \\ &= \int p(y|x)p(z|x)p(x)\log q_\theta(y|z)dxdydz \\ &= \frac{1}{N}\sum_{n=1}^{N}\int p(z|x_n)\log q_\theta(y_n|z)dz \\ &= \frac{1}{N}\sum_{n=1}^{N}\int p(\epsilon)\log q_\theta(y_n|z)d\epsilon \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbb{E}_{\epsilon\sim p(\epsilon)}[\log q_\theta(y_n|z)] \\ &= \mathbb{E}_{\epsilon\sim p(\epsilon)}[\log q_\theta(y_{sample}|z)] \end{aligned} \tag{24}$$

Summation is removed in the last derivation step, because we assume only one attribution sample. We can write the optimization problem as minimizing a loss function, then the loss function is [40]:

$$L = \mathbb{E}_{\epsilon\sim p(\epsilon)}[-\log q_\theta(y_{sample}|z)] \tag{25}$$

L is the cross entropy loss function for a single sample x_{sample}.

C

Through the Lens of the Model (Appendix)

6. Appendix

6.1. Null Feature Experiment

More visual examples for the null feature experiment are provided in Fig. 6.

6.2. Feature Saturation Experiment

The visual example for feature saturation is provided in Fig. 5.

6.3. Optimization (Focal Loss) Details

The optimization problems in the framework have multiple losses. Therefore it is a challenge to balance the optimization between the losses. We use the focal loss [10] method to balance the optimization. The following terms each indicate the keys used during optimization phase as focal loss entries.

$$\begin{aligned} \kappa_{\mathcal{L}} &= \sigma_t(\Phi_t(X)) \\ \kappa_{\{f_a\}} &= tanh(\frac{|\Phi_t(X_{\{f_a\}}) - \Phi_t(X_{\{\}})|}{\min(|\Phi_t(X_{\{f_a\}})|,\ |\Phi_t(X_{\{\}})|)}) \\ \kappa_{\{f_a,\,f_b\}} &= tanh(\frac{|\Phi_t(X_{\{f_a,\,f_b\}}) - \Phi_t(X_{\{f_a\}})|}{\min(|\Phi_t(X_{\{f_a,\,f_b\}})|,\ |\Phi_t(X_{\{f_a\}})|)}) \end{aligned} \tag{12}$$

where $\sigma_t(.)$ designates softmax function corresponding to the target t. The weighted average of the term $\kappa_{\mathcal{L}}$ is emploied in each of the loss terms as the multiplicand of the first term. The weighted average of other two are used in their corresponding scenarios as the multipicand of the latter terms. The weighted average through each iteration is calculated in the following manner:

$$\hat{\kappa}_{t+1} = \alpha\kappa_t + (1-\alpha)\hat{\kappa}_t \tag{13}$$

During all the experiments, the α is set to 0.1, and the initial value for κ is 0.5. To further faciliate the optimization phase, we initially optmize the features to maximize $\Phi_t(X)$ fo their designated target class.

6.3.1 Comparison to Existing Evaluations

Each evaluation in Sec. 2.2 evaluates explanations from a different perspective. [16, 34, 35] discuss the axioms *theoretically*. Proofs are broken in practice. E.g. our framework identifies issues with DeepSHAP (as also shown in [34]). Our framework is the practical incarnation for the axioms in [16, 34, 35]. [31] provides a class-sensitivity evaluation. Our results complement them. The metric in [31] considers the correlation between maps of different classes, thus identifies gradient as class sensitive. But the low correlation is due to a mere shift in noise, which our method avoids. The pointing game [38] assumes the model uses features that we humans use. We remedy this by having control over generated features (Sec 2.2). GBP, IG get high scores in [25], but we reveal they attribute to null-feature and are class insensitive. [11] aims to evaluate another aspect, feature importance (Sec 2.2 for limitations). A method such as FullGrad and GradCAM++ can highlight important features, but we show they attribute to Null and are class-insensitive. In cases where there is only one highly activating region in the input, these methods will reveal them ([14]).

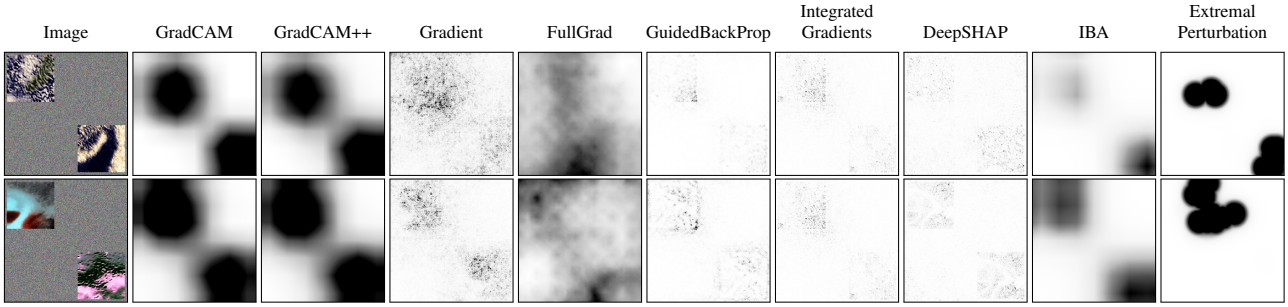


Figure 5. **Feature Saturation Experiment**: Each row is a sample from the feature saturation experiment. In this experiment, the features (patches) each saturate the output. In other terms, each individually generates the same output as their combination. A desired property for the attribution method is to distribute the contribution equally between the features. We observe that Extremal Perturbation and IBA can lean toward attributing the output to only one of the features. The formulation of these two method is based on keeping a region that keeps the output prediction. Thus, it is expected that they lean toward one feature.

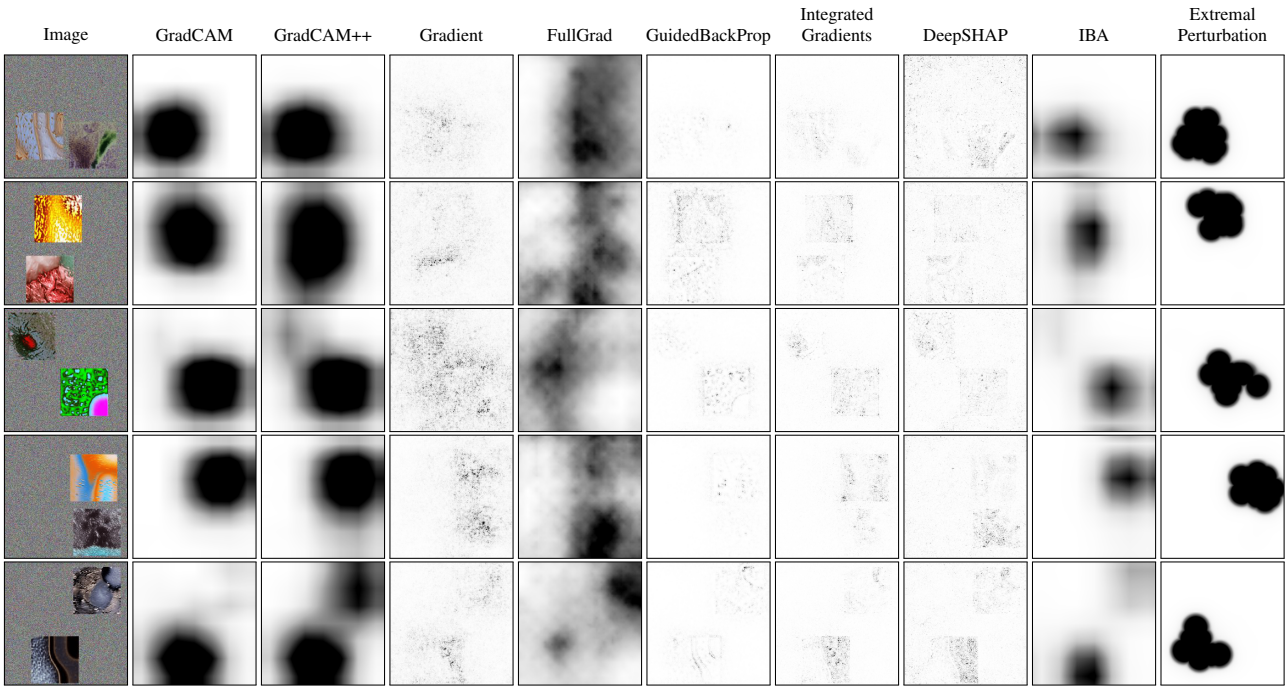


Figure 6. **Null Feature Experiment**: Each row represents a sample from the null feature experiment. In each row, the image on the left represents the generated features on the reference (noise) input. The features are generated using the model itself. Within the image, the lower feature (patch) is generated such that it is a null feature for the output. The rest of the images represent different explanations. As the second feature is a null feature, an explanation method should not assign importance to it. We observe that GradCAM, IBA, and Extremal Perturbation perform best in avoiding the null feature.

Bibliography

[1] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE", *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950, ISSN: 0026-4423. DOI: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. [Online]. Available: https://doi.org/10.1093/mind/LIX.236.433.

[2] L. S. Shapley, "A value for n-person games", *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.

[3] H. B. Barlow *et al.*, "Possible principles underlying the transformation of sensory messages", *Sensory communication*, vol. 1, no. 01, 1961.

[4] K. J. W. Craik, *The nature of explanation*. CUP Archive, 1967, vol. 445.

[5] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images", *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[6] R. L. Hahnloser, "On the piecewise analysis of networks of linear threshold neurons", *Neural Networks*, vol. 11, no. 4, pp. 691–697, 1998.

[7] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[8] B. A. Olshausen and D. J. Field, "Sparse coding of sensory inputs", *Current opinion in neurobiology*, vol. 14, no. 4, pp. 481–487, 2004.

[9] A. Turing, "465Intelligent Machinery, A Heretical Theory (c.1951)", in *The Essential Turing*, Oxford University Press, Sep. 2004, ISBN: 9780198250791. DOI: 10.1093/oso/9780198250791.003.0018. eprint: https://academic.oup.com/book/0/chapter/355746549/chapter-pdf/43817019/isbn-9780198250791-book-part-18.pdf. [Online]. Available: https://doi.org/10.1093/oso/9780198250791.003.0018.

[10] P. Dayan and L. F. Abbott, *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT press, 2005.

[11] J.-L. Marichal, I. Kojadinovic, and K. Fujimoto, “Axiomatic characterizations of generalized values”, *Discrete Applied Mathematics*, vol. 155, no. 1, pp. 26–43, 2007.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE conference on computer vision and pattern recognition*, Ieee, 2009, pp. 248–255.

[13] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, “Visualizing higher-layer features of a deep network”, *University of Montreal*, vol. 1341, no. 3, p. 1, 2009.

[14] D. Baehrens, T. Schroeter, S. Harmeling, M. Kawanabe, K. Hansen, and K.-R. MÃžller, “How to explain individual classification decisions”, *Journal of Machine Learning Research*, vol. 11, no. Jun, pp. 1803–1831, 2010.

[15] S. J. Russell, *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[16] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks”, in *Journal of Machine Learning Research*, 2011.

[17] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis”, *ACL*, 2011.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

[19] A. F. Rajesh P. N. Rao, *Computational neuroscience*, 2012.

[20] D. E. Broadbent, *Perception and communication*. Elsevier, 2013.

[21] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps”, *arXiv preprint arXiv:1312.6034*, 2013.

[22] T. Berg, J. Liu, S. W. Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, “Birdsnap: Large-scale fine-grained visual categorization of birds”, in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2014.

[23] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples”, *arXiv preprint arXiv:1412.6572*, 2014.

[24] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, “On the number of linear regions of deep neural networks”, in *Advances in Neural Information Processing Systems*, 2014.

[25] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, *arXiv preprint arXiv:1409.1556*, 2014.

[26] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks”, in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[27] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Object detectors emerge in deep scene cnns”, *arXiv preprint arXiv:1412.6856*, 2014.

[28] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”, *PloS one*, vol. 10, no. 7, e0130140, 2015.

[29] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network”, *arXiv preprint arXiv:1503.02531*, 2015.

[30] A. Mahendran and A. Vedaldi, “Understanding deep image representations by inverting them”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[31] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net”, in *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015. arXiv: 1412.6806.

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks”, in *Advances in neural information processing systems*, 2016, pp. 3387–3395.

[34] M. T. Ribeiro, S. Singh, and C. Guestrin, “"Why should i trust you?" Explaining the predictions of any classifier”, in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, New York, New York, USA: Association for Computing Machinery, Aug. 2016, pp. 1135–1144, ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. arXiv: 1602.04938. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2939672.2939778.

[35] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, “Adversarial manipulation of deep representations”, *International Conference on Learning Representations*, 2016.

[36] D. Silver, A. Huang, C. J. Maddison, *et al.*, “Mastering the game of go with deep neural networks and tree search”, *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[37] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization”, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[38] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck", *ICLR*, 2017.

[39] M. Ancona, E. Ceolini, C. Öztireli, and M. Gross, "Towards better understanding of gradient-based attribution methods for deep neural networks", *arXiv preprint arXiv:1711.06104*, 2017.

[40] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6541–6549.

[41] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch", *arXiv preprint arXiv:1712.09665*, 2017.

[42] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers", *Advances in neural information processing systems*, vol. 30, 2017.

[43] K. Daniel, *Thinking, fast and slow*, 2017.

[44] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation", in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3429–3437.

[45] J. Johnson, B. Hariharan, L. Van Der Maaten, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick, "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2901–2910.

[46] S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions", in *Advances in Neural Information Processing Systems*, 2017. arXiv: 1705.07874.

[47] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K. R. Müller, "Explaining nonlinear classification decisions with deep Taylor decomposition", *Pattern Recognition*, 2017, ISSN: 00313203. DOI: 10.1016/j.patcog.2016.11.008. arXiv: 1512.02479.

[48] C. Olah, A. Mordvintsev, and L. Schubert, "Feature visualization", *Distill*, 2017. DOI: 10.23915/distill.00007.

[49] V. Papyan, Y. Romano, and M. Elad, "Convolutional neural networks analyzed via convolutional sparse coding", *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2887–2938, 2017.

[50] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks", in *34th International Conference on Machine Learning, ICML 2017*, 2017, ISBN: 9781510855144.

[51] P. Rajpurkar, J. Irvin, K. Zhu, *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning", *arXiv preprint arXiv:1711.05225*, 2017.

[52] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. R. Müller, "Evaluating the visualization of what a deep neural network has learned", *IEEE Transactions on Neural Networks and Learning Systems*, 2017, ISSN: 21622388. DOI: 10.1109/TNNLS.2016.2599820. arXiv: 1509.06321.

[53] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization", in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[54] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences", in *34th International Conference on Machine Learning, ICML 2017*, 2017, ISBN: 9781510855144. arXiv: 1704.02685.

[55] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the game of go without human knowledge", *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[56] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: Removing noise by adding noise", *arXiv preprint arXiv:1706.03825*, 2017.

[57] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks", in *34th International Conference on Machine Learning, ICML 2017*, 2017. arXiv: 1703.01365.

[58] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps", in *Advances in Neural Information Processing Systems*, 2018. arXiv: 1810.03292.

[59] C.-H. Chang, E. Creager, A. Goldenberg, and D. Duvenaud, "Explaining image classifiers by counterfactual generation", *arXiv preprint arXiv:1807.08024*, 2018.

[60] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks", in *2018 IEEE winter conference on applications of computer vision (WACV)*, IEEE, 2018, pp. 839–847.

[61] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei, "Dynamic task prioritization for multitask learning", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 270–287.

[62] A. S. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization", in *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018. arXiv: 1803.06959.

[63] W. Nie, Y. Zhang, and A. Patel, "A theoretical explanation for perplexing behaviors of backpropagation-based visualizations", *arXiv preprint arXiv:1805.07039*, 2018.

[64] J. Sulam, V. Papyan, Y. Romano, and M. Elad, "Multilayer convolutional sparse modeling: Pursuit and dictionary learning", *IEEE Transactions on Signal Processing*, vol. 66, no. 15, pp. 4090–4104, 2018.

[65] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Deep image prior", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 9446–9454.

[66] Y. Wang, H. Su, B. Zhang, and X. Hu, "Interpret neural networks by identifying critical data routing paths", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8906–8914.

[67] F. Yu, Z. Qin, and X. Chen, "Distilling critical paths in convolutional neural networks", *arXiv preprint arXiv:1811.02643*, 2018.

[68] J. Zhang, S. A. Bargal, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-down neural attention by excitation backprop", *International Journal of Computer Vision*, vol. 126, no. 10, pp. 1084–1102, 2018.

[69] B. Zhou, Y. Sun, D. Bau, and A. Torralba, "Revisiting the Importance of Individual Units in CNNs via Ablation", Jun. 2018. arXiv: 1806.02891. [Online]. Available: http://arxiv.org/abs/1806.02891.

[70] M. Ancona, C. Öztireli, and M. Gross, "Explaining deep neural networks with a polynomial time algorithm for Shapley values approximation", in *36th International Conference on Machine Learning, ICML 2019*, 2019, ISBN: 9781510886988. arXiv: 1903.10992.

[71] S. Carter, Z. Armstrong, L. Schubert, I. Johnson, and C. Olah, "Activation atlas", *Distill*, 2019, https://distill.pub/2019/activation-atlas. DOI: 10.23915/distill.00015.

[72] A.-K. Dombrowski, M. Alber, C. Anders, M. Ackermann, K.-R. Müller, and P. Kessel, "Explanations can be manipulated and geometry is to blame", in *Advances in Neural Information Processing Systems*, 2019, pp. 13 589–13 600.

[73] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks", in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2950–2958.

[74] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile", in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3681–3688.

[75] S. Hooker, D. Erhan, P.-J. Kindermans, and B. Kim, "A benchmark for interpretability methods in deep neural networks", in *Advances in Neural Information Processing Systems*, 2019, pp. 9737–9748.

[76] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features", in *Advances in Neural Information Processing Systems*, 2019, pp. 125–136.

[77] A. Khakzar, S. Albarqouni, and N. Navab, "Learning interpretable features via adversarially robust optimization", in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, Cham: Springer International Publishing, 2019, pp. 793–800, ISBN: 978-3-030-32226-7.

[78] A. Khakzar, S. Baselizadeh, S. Khanduja, C. Rupprecht, S. T. Kim, and N. Navab, *Improving feature attribution through input-specific network pruning*, 2019. DOI: 10.48550/ARXIV.1911.11081. [Online]. Available: https://arxiv.org/abs/1911.11081.

[79] B. Kim, J. Seo, S. Jeon, J. Koo, J. Choe, and T. Jeon, "Why are saliency maps noisy? cause of and solution to noisy saliency maps", in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, IEEE, 2019, pp. 4149–4157.

[80] G. H. Lee, D. Alvarez-Melis, and T. S. Jaakkola, "Towards robust, locally linear deep networks", in *7th International Conference on Learning Representations, ICLR 2019*, 2019. arXiv: 1907.03207.

[81] C. Molnar, *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*. 2019, https://christophm.github.io/interpretable-ml-book/.

[82] Z. Qi, S. Khorram, and F. Li, "Visualizing deep networks by optimizing with integrated gradients", *CVPR Workshop*, 2019.

[83] Y. Qiu, J. Leng, C. Guo, *et al.*, "Adversarial defense through network profiling based path extraction", in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00491.

[84] S. Srinivas and F. Fleuret, "Full-gradient representation for neural network visualization", Tech. Rep., 2019.

[85] J. Wagner, J. M. Kohler, T. Gindele, L. Hetzel, J. T. Wiedemer, and S. Behnke, "Interpretable and fine-grained visual explanations for convolutional neural networks", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9097–9107.

[86] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis", in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2019. arXiv: 1702.04595.

[87] L. Arras, A. Osman, and W. Samek, "Ground truth evaluation of neural network explanations with clevr-xai", *arXiv preprint arXiv:2003.07258*, 2020.

[88] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[89] S. Denner, A. Khakzar, M. Sajid, *et al.*, "Spatio-temporal learning from longitudinal data for multiple sclerosis lesion segmentation", in *International MICCAI Brainlesion Workshop*, Springer, 2020, pp. 111–121.

[90] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.

[91] C. Izzo, A. Lipani, R. Okhrati, and F. Medda, "A baseline for shapely values in mlps: From missingness to neutrality", *arXiv preprint arXiv:2006.04896*, 2020.

[92] J. D. Janizek, P. Sturmfels, and S.-I. Lee, "Explaining explanations: Axiomatic feature interactions for deep networks", *arXiv preprint arXiv:2002.04138*, 2020.

[93] A. Khakzar, S. Baselizadeh, and N. Navab, "Rethinking positive aggregation and propagation of gradients in gradient-based saliency methods", *arXiv preprint arXiv:2012.00362*, 2020.

[94] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, "Zoom in: An introduction to circuits", *Distill*, 2020, https://distill.pub/2020/circuits/zoom-in. DOI: 10.23915/distill.00024.001.

[95] S.-A. Rebuffi, R. Fong, X. Ji, and A. Vedaldi, "There and Back Again: Revisiting Backpropagation Saliency Methods", Apr. 2020. arXiv: 2004.02866. [Online]. Available: http://arxiv.org/abs/2004.02866.

[96] L. K. Scheffer, C. S. Xu, M. Januszewski, *et al.*, "A connectome and analysis of the adult drosophila central brain", *Elife*, vol. 9, 2020.

[97] K. Schulz, L. Sixt, F. Tombari, and T. Landgraf, "Restricting the flow: Information bottlenecks for attribution", in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=S1xWh1rYwB.

[98] L. Sixt, M. Granz, and T. Landgraf, "When explanations lie: Why many modified bp attributions fail", in *International Conference on Machine Learning*, PMLR, 2020, pp. 9046–9057.

[99] P. Sturmfels, S. Lundberg, and S.-I. Lee, "Visualizing the impact of feature attribution baselines", *Distill*, vol. 5, no. 1, e22, 2020.

[100] M. Sundararajan and A. Najmi, "The many shapley values for model explanation", *37th International Conference on Machine Learning, ICML 2020*, 2020.

[101] A. Davies, P. Veličković, L. Buesing, *et al.*, "Advancing mathematics by guiding human intuition with ai", *Nature*, vol. 600, no. 7887, pp. 70–74, 2021.

[102] J. Jumper, R. Evans, A. Pritzel, *et al.*, "Highly accurate protein structure prediction with alphafold", *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[103] A. Khakzar, S. Baselizadeh, S. Khanduja, C. Rupprecht, S. T. Kim, and N. Navab, "Neural response interpretation through the lens of critical pathways", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 528–13 538.

[104] A. Khakzar, S. Musatian, J. Buchberger, *et al.*, "Towards semantic interpretation of thoracic disease and covid-19 diagnosis models", in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, Cham: Springer International Publishing, 2021, pp. 499–508, ISBN: 978-3-030-87199-4.

[105] A. Khakzar, Y. Zhang, W. Mansour, *et al.*, "Explaining covid-19 and thoracic pathology model predictions by identifying informative input features", in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Springer, 2021, pp. 391–401.

[106] S. T. Kim, L. Goli, M. Paschali, *et al.*, "Longitudinal quantitative assessment of covid-19 infection progression from chest cts", in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, Cham: Springer International Publishing, 2021, pp. 273–282, ISBN: 978-3-030-87234-2.

[107] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision", in *International Conference on Machine Learning*, PMLR, 2021, pp. 8748–8763.

[108] A. Ramesh, M. Pavlov, G. Goh, *et al.*, "Zero-shot text-to-image generation", in *International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[109] Y. Zhang, A. Khakzar, Y. Li, A. Farshad, S. T. Kim, and N. Navab, "Fine-grained neural network explanation by identifying input features with predictive information", *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 040–20 051, 2021.

[110] J.-B. Alayrac, J. Donahue, P. Luc, *et al.*, "Flamingo: A visual language model for few-shot learning", *arXiv preprint arXiv:2204.14198*, 2022.

[111] M. Atad, V. Dmytrenko, Y. Li, *et al.*, "Chexplaining in style: Counterfactual explanations for chest x-rays using stylegan", *arXiv preprint arXiv:2207.07553*, 2022.

[112] P. Engstler, M. Keicher, D. Schinz, *et al.*, "Interpretable vertebral fracture diagnosis", in *Interpretability of Machine Intelligence in Medical Image Computing*, Cham: Springer Nature Switzerland, 2022, pp. 71–81, ISBN: 978-3-031-17976-1.

[113] A. Fawzi, M. Balog, A. Huang, *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning", *Nature*, vol. 610, no. 7930, pp. 47–53, 2022.

[114] A. Khakzar, P. Khorsandi, R. Nobahari, and N. Navab, "Do explanations explain? model knows best", in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 244–10 253.

[115] A. Khakzar, Y. Li, Y. Zhang, *et al.*, "Analyzing the effects of handling data imbalance on learned features from medical images by looking into the models", *arXiv preprint arXiv:2204.01729*, 2022.

[116] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding", *arXiv preprint arXiv:2205.11487*, 2022.

[117] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)", [Online]. Available: `http://www.cs.toronto.edu/~kriz/cifar.html`.