Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Object Detection with Limited Labels

Rahul Parthasarathy Srikanth

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Object Detection with Limited Labels

Objekterkennung mit limitierten Labeln

Author:	Rahul Parthasarathy Srikanth
Examiner:	Dr. rer. nat. Felix Dietrich
Advisor 1:	M. Sc. Mathias Sundholm (PreciBake GmbH)
Advisor 2:	M. Sc. Alexander Dolokov (PreciBake GmbH)
Submission Date:	December 1st, 2022

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

December 1st, 2022 Rahul Parthasarathy Srikanth

Acknowledgments

This thesis is a cumulative effort of many individuals. I want to thank myself for taking the risk to drop out of a premier institute and change career paths, believing that I am more suited to building AI products than automobile engines.

This thesis was only possible with the support of M. Sc. Mathias Sundholm and M. Sc. Alexander Dolokov from Precibake GmbH. I appreciate them for providing the freedom to choose the scientific problem that interested me the most. Furthermore, I am grateful for the constant motivation and support provided during the thesis period. Finally, I thank them for instilling the courage to explore new directions, irrespective of how the results may turn out. I also thank the Machine Learning team at Precibake for their discussions and support during the thesis.

I am grateful to Dr. Felix Dietrich for graciously agreeing to examine this thesis despite his numerous commitments. His insights were crucial to building this thesis into more rigorous scientific standards. I also thank him for his methodical feedback and discussions throughout the thesis.

Finally, I thank my parents, brother, and friends for supporting me when times looked tough and for their constant motivation and encouragement.

Abstract

Most modern machine learning techniques based on neural networks need large amounts of well-annotated training data to demonstrate outstanding performance. However, obtaining instance-level annotations like object bounding boxes in real-world settings is tedious, and it is still a significant challenge in many industrial applications of neural networks. Additionally, the distribution of classes in real-world datasets is inherently imbalanced. This thesis deals with the challenge of few-shot object detection. Given a set of base classes with abundant labeled data and a set of novel classes with few support images, this thesis aims to devise algorithms that can predict and classify the object instance belonging to both the base and novel classes in a query image.

In the first part of the thesis, a representative multi-modal model CLIP [38] as well as a self-supervised model, DINO [6], are experimented with to test the efficacy of different pretraining strategies. Then, algorithms based on class codes are introduced to classify the query image into target novel classes. Finally, experiments were performed on CUBS 200 dataset [50]. With the proposed classification algorithm, the CLIP encoder demonstrated strong few-shot capability with just ten support images a class and competed with a fully supervised ResNet-50 [19] model.

In the second part of the thesis, a novel few-shot object detector CLIPtheCenter is designed that extends CenterNet for the task of few-shot object detection. Generalization experiments are performed on a dataset consisting of fifteen base classes and five novel classes from the PASCAL VOC dataset [12]. As a result, CLIPtheCenter doubled the class agnostic detection performance compared to a vanilla CenterNet in this base class splits. Furthermore, when classification performance is also taken into account, CLIPtheCenter tripled the performance of the CenterNet on the base classes split when CLIP Encoder-based class codes are used for class assignments. Finally, ablation studies verified the underlying intuitions for these performance gains.

Contents

1 Introduction

Over the past decade, machine learning based methods have significantly impacted various fields, such as computer vision, natural language processing, and medical diagnosis. In particular, with the reemergence of deep learning models since the period the early 2010s, the applications of machine learning are steadily expanding. Two fundamental reasons exist for this explosion of deep learning in driving performance benchmarks. One is the amount of organized large-scale datasets available for training models. One of the earliest computer vision datasets, CIFAR-10 and CIFAR 100 [24], was released by Krizhevsky in 2009, with each consisting of 60k lower resolution color images of 10 and 100 classes, respectively. Subsequently, the ImageNet [9] dataset and ImageNet Large Scale Visual Recognition Challenge (ILSVRC) were also proposed in 2009, serving as a benchmark dataset for image classification. These large-scale, diverse datasets led to increased research in applying neural models for computer vision, resulting in higher performances. The second driving factor for this explosion is the tremendous improvements in GPU hardware. Compute Unified Device Architecture (CUDA) [36] programming model, released in 2008, enabled the utilization of massively parallel codes in the NVIDIA Graphics Processing Unit (GPU). The model training speeds have improved multi-folds with improvements in transistor technology, specialized processors such as tensor cores, and algorithmic advancements.

With all these performance improvements evaluated on benchmark datasets, there is an important caveat. These benchmark datasets, such as ImageNet [9], PASCAL Visual Object Classes (PASCAL VOC) [13], or Common Objects in Context [30] have object classes that are quite varied in visual appearance. While this is helpful for the learning visual feature with widely distributed dataset classes, it is far apart from real-world applications of neural models. In most applications, the dataset classes are visually very similar in appearance. Therefore, the separation of objects into different classes is more challenging. Thus performance on the benchmark dataset may only partially reflect on real-world applications. The classification experiments of this thesis take this into account by choosing an evaluation dataset with similar-looking dataset classes.

Another crucial missing piece in translating performance in benchmark datasets to real-world scenarios is the class imbalances that are more prominent in reality. Most benchmark datasets use a balanced number of images across classes to evaluate the model's performance without introducing any uncertainties due to the class imbalance. However, in real-world scenarios, all object classes are not observed with the same frequencies. Es-

pecially in industrial applications, it is extremely unusual for different object classes to have uniformly the same number of images. These differences may come from product demands, production cycles, camera positions that capture the data, and other possible reasons. This thesis aims to tackle this problem of the ability of the model to detect classes that are not frequently seen, hereafter called the novel classes, by using both the abundantly seen data, called the base classes, and the limited images of the novel classes. This problem is referred to in the literature as few-shot learning.

More specifically, this thesis is motivated by the task of few-shot object detection from color images. Provided that abundant data is available for a set of base classes with bounding box and class annotations along with similar limited annotated images for a set of novel classes, the thesis proposes algorithms that can localize and detect object instances in both the base and novel classes.

The thesis is organized into five major chapters. The chapter 2 takes a deep dive into the current state-of-the-art feature extraction, object detection, and few-shot learning methodologies. Chapter 3 proposes algorithms for few-shot image classification, leveraging the advances in self-supervised and multi-modal learning. Chapter 4 builds a few-shot object detection architecture from scratch using the findings from the experiments of few-shot classification. The final chapter 5 summarizes the insights derived from the thesis and looks ahead into the horizon for further possible developments.

2 State of the Art

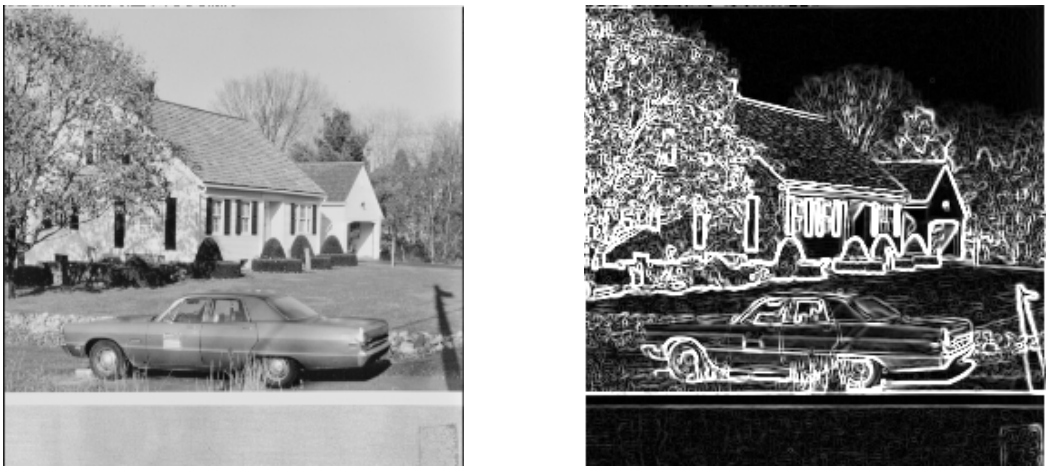
2.1 Visual Feature Extraction

2.1.1 What are Visual Features?

The perception mechanism of humans has evolved over millennia to understand visual inputs in the color space to more valuable patterns in the image, such as object boundaries, relative depth, and distances. Visual features are the fundamental building blocks to imitate this visual recognition system for practical semantic tasks such as classifying the presence of an object or localizing where the object is. In addition, these features act as a bridge between the commonly used Red Green Blue (RGB) color space to more numerical descriptive matrices. Through the development of computer vision as a scientific field, what defines visual features and the methods of estimation have continuously undergone advancements. Before the era of deep learning and learning visual features, the extraction of features was through handcrafted algorithms and rules. These algorithms tried to identify the primitive geometrical properties of the image, such as the location of lines, curves, or splines and retrieved more semantic high-level features from these. For example, by carefully designing convolutional kernels, the edges of different objects in an image can be detected. One such kernel is demonstrated in figure 2.1.

In order to describe these visual features into numerical matrices, more methods such as SIFT[33], SURF[2], and ORB[43] were proposed. These methods come from the intuition that the image's most useful visual features are those invariant in geometric transformations, such as different rotations and scales, and transformations in appearance, such as brightness and contrast. These methods pick up those features from the image that are robust to these transformations and provide a numerical encoding for each feature.

With the advent of deep learning, especially since the publication of AlexNet[25] in 2012, deep learning has taken over extracting features from an image. Instead of designing the handcrafted features for the downstream semantic task, the model can learn to define a visual feature and how to encode them. The model learns to extract features most suitable for the semantic task, such as classification, detection, or segmentation, by designing suitable loss functions. These learned features have demonstrated exceptional generalization ability across datasets.



(a) Source Image (b) Output Image after Sobel Operator

Figure 2.1: **Handcrafted Convolutional Kernels for Edge Detection.** The figures show the edge detections generated from the application of the Sobel operator [22]. High-level features such as object boundaries can be detected by carefully designing convolution kernels. Image Source: [8].

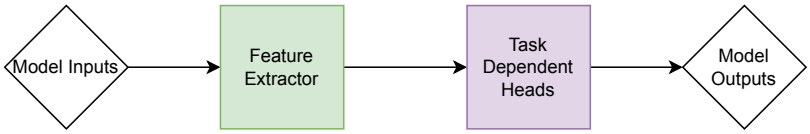


Figure 2.2: **Abstraction of Deep Learning Models**. Most deep learning models can be decomposed into a feature extractor backbone and a set of task-specific heads to regress or classify the model outputs.

2.1.2 Learned Feature Extraction Backbones

The feature extractor backbone is the most vital part that drives the model's performance, independent of the task of the deep learning model. Most deep learning modules can be simplistically abstracted to two major components - a feature extractor and task-dependent heads, as shown in the figure 2.2. For example, this task-dependent head is usually a fully connected linear layer for classification or a series of convolutions for a segmentation task. The feature extraction networks are derived from the backbones of classification models trained on large-scale datasets such as ImageNet [9]. Since the classification model is trained to identify diverse dataset classes, the backbone provides great generalization ability, and the final backbone layer's output is considered a feature descriptor. The further sections discuss some of the most landmark and state-of-the-art feature extractor backbones proposed in the literature.

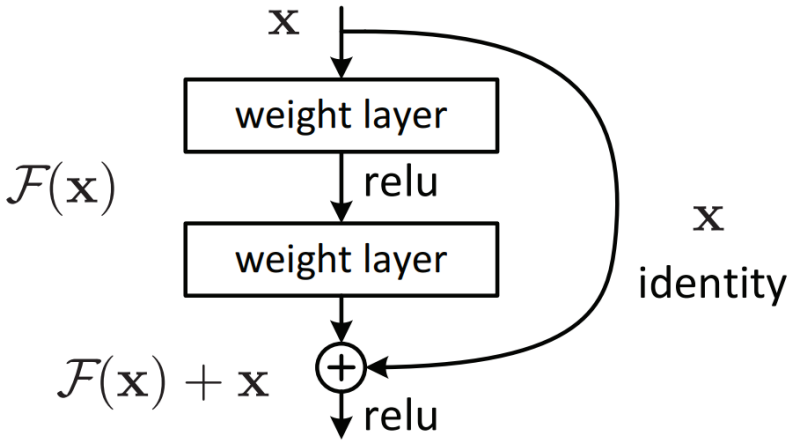


Figure 2.3: **Residual Block of the ResNet Architecture**. The concatenation of the input of the block to its output provides an additional pathway for the flow of gradients to prevent vanishing gradient problems. Image Source: [19].

Convolution Based Networks

Convolutional Neural Networks (CNN) comes from the fundamental idea of handcrafted convolution kernels. Instead of manually fixing the kernel values to target the image primitives, CNN learns these kernel values. This way, the CNN model decides the kernel values most suitable for the target task. One of the most commonly used feature extraction backbones is the idea of Residual Network (ResNet) [19] proposed by He et al. The residual blocks allow the construction of deeper neural networks with numerical stabilities during training iterations. An individual residual block is shown in figure 2.3. By concatenating the input of the residual block to its output, the vanishing gradient problems are minimized to a large extent. This auxiliary concatenation branch provides an additional path for the gradients. In cases where the value of gradient flow becomes small through weight layers, this auxiliary branch prevents the gradients from vanishing.

Several other backbone architectures exist for the classification task that can condense the global context of the image into a unifying feature map. Howard et al. proposed the MobileNet [20] family of architectures to improve the training and inference speeds from the ResNet models. This set of models uses depthwise separable convolutional to reduce the number of operations carried in both the forward and backward passes through the network. Tan and Le proposed the EfficientNet[45] family of architecture that minimizes the number of model parameters without sacrificing performance on classification. It achieves this through independent scaling of the layer dimensions and the number of convolutional filters through empirical observations.

However, for instance-level tasks such as semantic segmentation or object detection, the

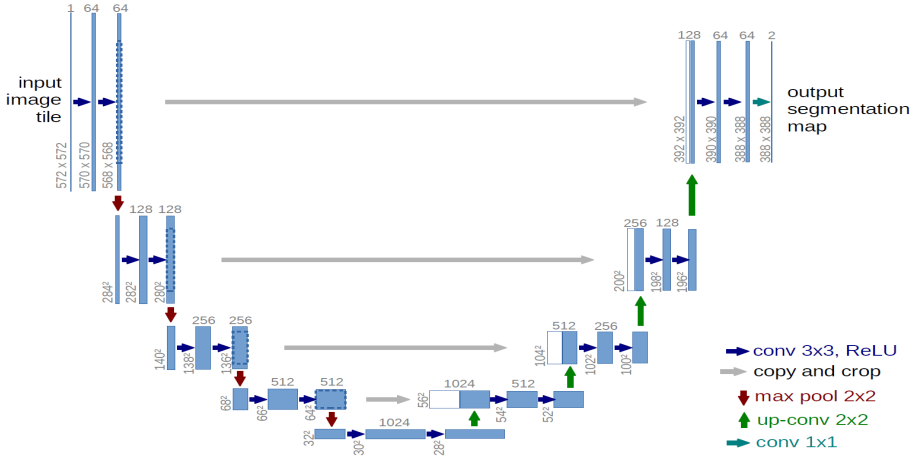
features of the different scales of images must be considered during the calculation of the feature description. Ronneberger et al. proposed the U-Net [42] architecture for medical image segmentation. This proposed model was a pioneer model that used features across multiple scales to accurately capture local primitives in the image. On a similar note, Newell et al. proposed the Hourglass[35] module for the task of human pose estimation. The main novelty of this hourglass architecture is the introduction of additional convolutional layers before concatenating the feature maps from different stages of the encoder and decoder. Deep Layer Aggregation [56] by Yu et al. extended this intuition by building a hierarchy of feature maps across different layers and fusing these features iteratively for better fine-grained image recognition. These architectures are illustrated in the figure 2.4.

Vision Transformers Based Frameworks

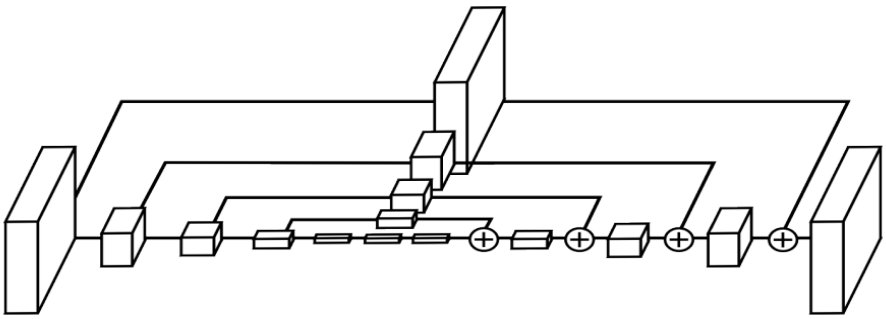
Transformer-based architectures have been at the forefront of driving benchmarks in natural language processing. Especially models such as Generative Pre-trained Transformer 3 (GPT-3) [4] have demonstrated human-level fluency in generating prose from text prompts. Dosovitskiy et al. proposed the Vision Transformer architecture that used the power of a multi-head self-attention mechanism for vision-based inputs. The multi-head self-attention mechanism is a cornerstone of transformers' strong performance. Subsequently, a lot more optimizations in performance, as well as training schedules, led to a more diverse variety of transformer mechanisms such as Swin Transformer [32], Focal Transformer [53] and Pyramid Vision Transformer [47].

A schematic of the vision transformer architecture is shown in figure 2.5. The input image is divided into a set of patches of uniform dimension and fed into a learned linear projection layer that encodes each patch into feature embedding. A positional encoding is also added to the patch encodings to preserve the local contexts. This positional encoding for each patch is also learned through backpropagation by passing the index of the patch. Finally, the transformer encoder takes the projected patches as input and provides a resultant feature map. Since the model is designed for classification purposes, a Multi-Layer Perceptron (MLP) head is used for assigning the probabilities to the individual classes. This process is shown in figure 2.5a.

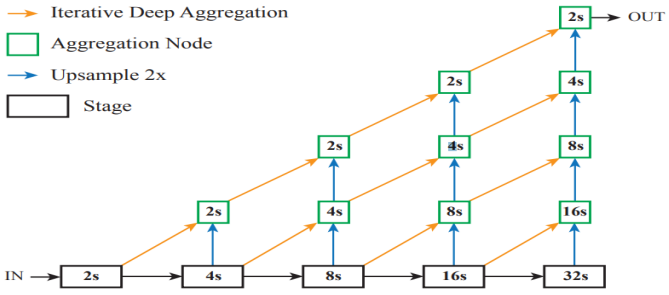
As shown in figure 2.5b, the encoder consists of a multi-head attention module at its core sandwiched between normalization layers for numerical stability. Each attention head takes the encoded patches as the input and provides an output feature map considering all the image patches. The model also learns the weighting coefficients assigned to different patches. This mechanism is in contrast to convolution-based networks, where the context that layers receive is limited to the neighborhood whose boundaries depend on the dimensions of the kernel. Three different variants of the vision transformer were proposed: Base, Large and Huge, depending on the number of layers and number of attention heads used.



(a) U-Net Architecture

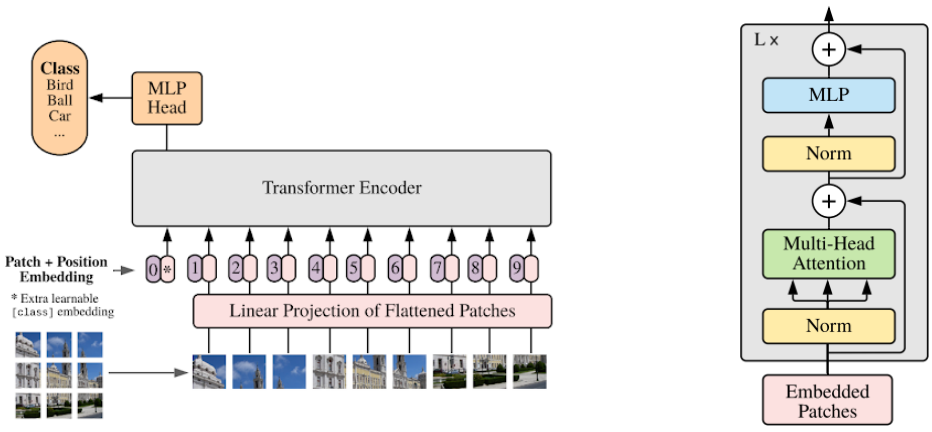


(b) A block of the Hourglass Architecture



(c) Deep Layer Aggregation Algorithm

Figure 2.4: **Different Feature Aggregation Methods for Instance Level Tasks.** Instance-level tasks rely on the presence of local context information in the feature maps. The figure illustrates some of the different aggregation schemes used for combining features at different scales. Images (a), (b), and (c) were taken from [42], [35], and [56], respectively.



(a) Schematic of Vision Transformer

(b) Encoder block of the transformer

Figure 2.5: **Vision Transformer Architecture.** The ViT model takes the input images as patches along with local context through a positional embedding. These are projected and fed into an Encoder block consisting of multi-head attention mechanisms. Multi-head attention mechanism enables the encoder to perceive the global context of the image with varying patch weights for a more descriptive embedding. Images Source: [10].

2.1.3 Self-Supervised Methods

In recent years, self-supervised learning methods have found a lot of attention, especially for image classification. The most significant advantage of these models is the freedom of not requiring annotated datasets for training. Since the targets for the model training are derived from the inputs, these models can use volumes of non-annotated data, saving annotation effort. Momentum Contrast (MoCo) [18] proposed by He et al. investigated the use of contrastive training for learning visual representation. By using different views of the same image and among different images for generating positive and negative samples, MoCo demonstrated strong performance in various benchmarks task. While MoCo is suited to image-level tasks due to the global pooling of information, other training methods also aim to capture the local context in the image. Xie et al. proposed Detco [51] for pretraining object detection backbones. DetCo builds upon MoCo by using additional losses for local-local, global-global patches, and cross-local-global patch correlations.

On a similar note, DINO [6] proposed by Caron et al. investigated the representational ability of self-supervised vision transformer models. DINO uses a similar momentum encoding framework as proposed in MoCo [18]. However, it uses a wise choice of image views for encoder inputs. As shown in figure 2.6, DINO uses a student and a teacher model from the standpoint of knowledge distillation. The teacher encoder receives only the global crops of the image. In contrast, the student encoder receives global and local

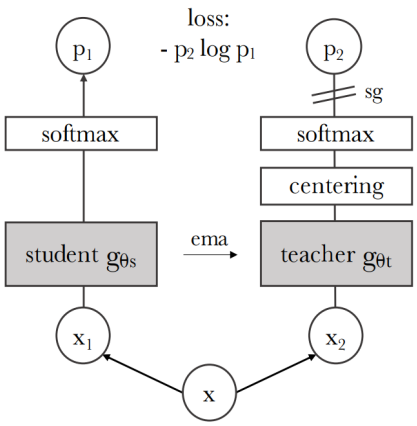


Figure 2.6: **Training Methodology of DINO.** DINO combines the effectiveness of momentum-based distillation with multiple image views for self-supervised learning. The student model needs to match the outputs for the teacher model even though input views are different between them. This produces more powerful visual representations that capture global and local image contexts. Image Source: [6].

crops of images. Enforcing the similarity of outputs between the two networks enables the models to learn effective representations for both local and global image contexts. This learning paradigm is also adapted to other tasks, such as segmentation and object discovery. For example, TokenCut [49] proposed by Wang et al. adapted DINO as the feature extraction backbone for unsupervised object discovery. Similarly, Kyriazi et al. demonstrated using Deep Spectral Methods [34] along with the DINO-based feature descriptions for unsupervised object segmentation.

2.1.4 Multi-Modal Learning Methods

Combining vision and text inputs is well-demonstrated for image-captioning and visual question-answering applications. However, the research in unifying representation for pretraining feature extraction backbones is a relatively new application of multi-modal models. This is inherently a more challenging task since vision and text are modalities of widely different characteristics. Combining them into a shared latent space would enable images to be classified only by providing the object class name present in the image as the prompt. In 2021, Radford et al. proposed the Contrastive Language Pre-Training (CLIP), widely considered a pioneer work in unifying visual and text information into a common latent space, where cosine distances could directly compute feature similarities.

Figure 2.7 shows the training schematic of the CLIP architecture. The model was trained on a large-scale dataset of 400 million text-image pairs scraped from the internet. CLIP

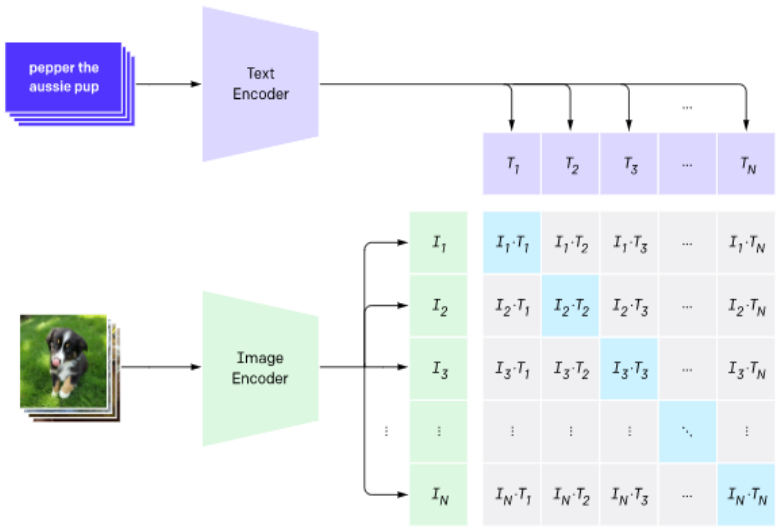


Figure 2.7: **Training Methodology of CLIP.** CLIP uses paired image-text inputs for training each of the text and vision encoders. The diagonal elements become the positive samples, and the non-diagonal elements become the negative samples for calculating the contrastive loss across all input combinations. Image Source: [39].

uses an independent vision and text encoder to extract the feature from image and text inputs. During training, these paired images and prose are fed into the respective encoders to generate embedding across all combinations of the inputs. The diagonal elements denoting the right pairing are considered positive samples, and all non-diagonal elements become negative samples. The contrastive loss is used to train the model with these positive and negative samples.CLIP pre-trained backbones have demonstrated solid zero-shot performance across various classification benchmarks.

Recently, a lot more similar methods for vision-text distillation have been proposed. Li et al. proposed Align before Fuse (ALBEF) [28] that uses an additional momentum encoder to improve downstream performance. Yang et al. proposed the use of Triple Contrastive Learning [54] that builds upon ALBEF by using separate momentum-based learning for vision and text.

2.2 Object Detection with Deep Learning

While the neural networks designed for classification had demonstrated substantial performance in the benchmark datasets as early as 2012 with the introduction of AlexNet [25], architectures for object detection took a slightly lower progress rate. This is because feature

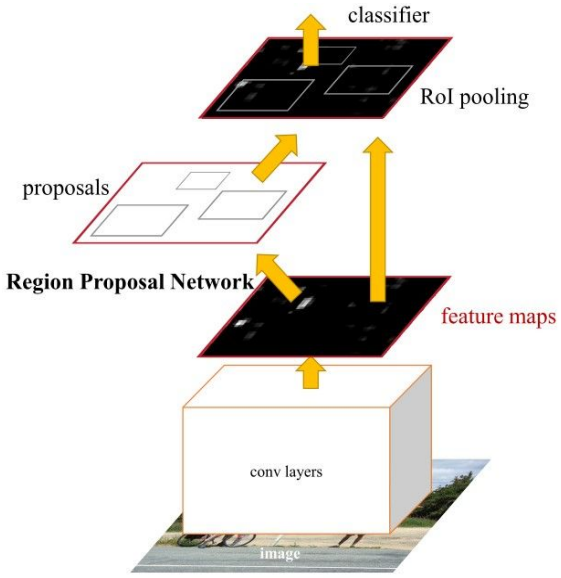


Figure 2.8: **Architecture of Faster-RCNN.**. Faster-RCNN, being a two-stage detector, has a Region Proposal Network (RPN) to detect the presence of objects through learned objectness scores. RPN is also trained simultaneously along the detection heads. Image Source: [41].

extractor backbones for classification do not need to capture all the local details in an input image. However, object detection tasks require preserving the positional information to localize the object correctly.

The Regional Convolutional Neural Network (RCNN) [15] architecture proposed by Girshick et al. first investigated the use of learned feature backbones for regressing the positions of the object as well as classifying the object proposals. Since then, multiple models have been proposed that have bought improvements in performance and inference rates. Most object detection architectures can be schematically represented as multi-stage or single-stage architecture. The further sections discuss them in greater detail.

2.2.1 Multi-Stage Detectors

The biggest roadblock in object detection during its inception was the generation of bounding box proposals. Since feature extractors for classification are quite mature, early detection models used non-learning-based object region proposals such as Selective Search [46]. These methods often used texture definitions to connect potential object regions. Ren et al., with the proposal of Faster-RCNN [41], successfully demonstrated a learned region proposal network that trains along with the detection heads. Figure 2.8 illustrates the schematic structure of Faster-RCNN.

Faster-RCNN uses a backbone feature extractor, usually pre-trained on ImageNet, to

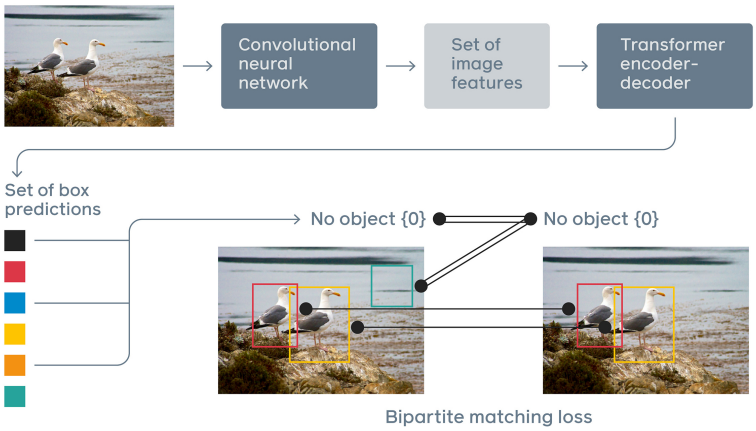


Figure 2.9: **Architecture of Detection Transformer.** The detection transformer uses a vision transformer-based encoder-decoder to predict the bounding box from the feature maps of a pre-trained encoder. The predicted and ground-truth bounding boxes are matched to encourage bipartite connections. Image Source: [5].

help the model achieve faster convergence. The feature maps generated by this backbone serve as a common precursor to both a Region Proposal Network (RPN) and the detection heads. A set of predefined anchors with varying aspect ratios are attached to each output feature map pixel to detect potential objects of different sizes. The Region proposal network calculates the probability of an object's presence in each of these anchors. The RoI pooling operation then re-sizes the anchors of different shapes with top objectness scores into a unifying feature map dimension for feeding as input to the detection heads. The detection head consists of an object classifier for scoring the probabilities of different classes and a regression head to refine the anchor dimensions. The success of Faster-RCNN led to a lot of derivative work such as R-FCN [7], R-FCN++ [29], which further improved the performance of Faster-RCNN by optimized training speed as well as performance by reducing the costs of RoI pooling operations.

More recently, with the success of vision transformer backbones, Carion et al. l proposed the detection transformer (DETR) as an end-to-end object detection framework. As shown in the figure 2.9, DETR used a vision transformer to convert the feature maps from a pre-trained encoder into a set of bounding box predictions along with a class label. These predicted bounding boxes are matched with the ground-truth boxes using a bi-partite matching mechanism. The loss functions for the bi-partite matching are defined to encourage one prediction to be matched to precisely one ground truth box by maximizing the number of edges. Several architectures have also been proposed, such as Deformable DETR [61] and Efficient DETR [55], that build upon mechanisms proposed in DETR.

2.2.2 Single-Stage Detectors

While multi-stage detectors use separate modules for generating object proposals apart from a feature extractor and the detection heads, single-stage detectors are designed to combine the region proposal and feature extractor into a common module. This unification eliminates the need for multiple stages. As a result, the inference performance is greatly improved for potential real-time applications by eliminating the need for this additional proposal stage. In 2016, Redmon et al. pioneered the single-stage object detection framework with the You Look Only Once (YOLO) model. Instead of requiring predefined or learned object proposals, YOLO divides the image into a set of patches and assumes the presence of a bounding box within each grid. Even though this restricts the maximum number of proposals that can be generated depending on the patch dimensions, the gains in inference speed make up for the lack of detection performance. Liu et al. improved this further with the Single Shot Detector (SSD) [31] by using feature maps at different receptive fields for regression and classification heads. SSD aimed at improving the detection performance by considering multiple scales of the anchor box to reflect different possible object scale variations in an image.

CornerNet [27] proposed by Law and Deng redefined object localization into a paired keypoint problem. Once the top left corner and bottom right are predicted, the bounding box can be completely defined. CornerNet models these keypoints as gaussian peaks of the heatmaps, which are then used as detection targets for training the model. Zhou et al. improved this further with CenterNet [60]. CenterNet used the object center as the heatmap peaks instead of the corner points of the bounding box. This greatly improved the detection performance, surpassing some of the more computationally heavy two-stage detectors and at the same time having high-inference speeds as shown in the figure 2.10.

For each input image, the CenterNet architecture predicts three heatmaps. They are the object centers, bounding box dimensions, and offsets. As seen from the figure 2.11, the output heatmap is downsized to a scale of 25 percent of the original image dimensions. The input image is fed into an encoder-decoder-style network to extract useful visual features. This extractor feature becomes the precursor for all the heatmap heads. The center heatmap head calculates the probability of a pixel in the feature map being an object center. Similarly, the other two heads regress the height and width of the potential bounding box. To prevent the loss of center localization accuracy due to the downsizing of the heatmap, offset heads regresses the fractional pixel value to add to the center heatmap values as an adjustment for pixel shifts. The heatmap head also carries out the classification of the proposals. While the bounding box and offset heatmap heads are independent of the object classes, the center heatmap has as many output dimensions as the number of classes in the dataset. During inference, the top-valued center heatmap pixels and the corresponding bounding box and offset head values are used to predict objects.

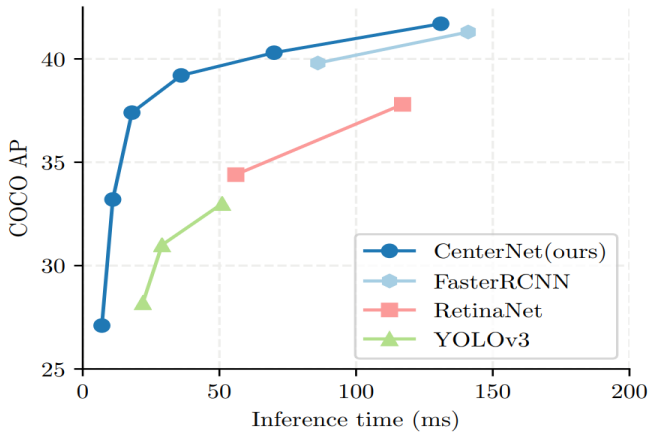


Figure 2.10: **Performance - Inference Time trade-off of different object detection architectures on the COCO validation dataset.**. Different points within the same series indicate different encoder backbones used. The larger the encoders are, the more the inference time and detection performance. CenterNet achieves the right balance between performance and speed. Image taken from [60].

2.2.3 Evaluation of Detection Models

Object detection, being an instance-level task, requires a more careful design of evaluation criteria. For the tasks such as classification, metrics like precision and recall are easier to calculate since there is a one-to-one relationship between the predicted class and the target class. In contrast, in object detection, the number of predictions generated could be multiples of the number of ground-truth objects. In order to reduce the number of redundant and overlapping bounding box proposals, iterative refinements like Non-Maximum Suppression are used for each object class. NMS removes redundant bounding boxes successively by considering the value of intersection over the union of overlapping boxes and the confidence score of the box proposals. The process of NMS is formalized in the algorithm 1. Figure 2.12 shows a sample input image processed through NMS to retain only more relevant non-overlapping bounding boxes.

Mean Average Precision (MAP) is the most commonly used metric for evaluating object detection frameworks. The presence of ”Mean” and ”average” denotes two iterative calculations upon which the metric is calculated. One, the value of precision is averaged across all the classes. Two, for each class, the precision is again averaged over multiple recall values through a precision-recall curve. For example, in most common benchmarks datasets such as Common Objects in Context (COCO) [30], evaluation is carried out across varying Intersection over Union (IoU) thresholds (usually in the range of 50 percent to 95 percent at 5 percent), and the precision is averaged at the level of these thresholds.

For determining the positive and negative samples for the calculation of the confusion

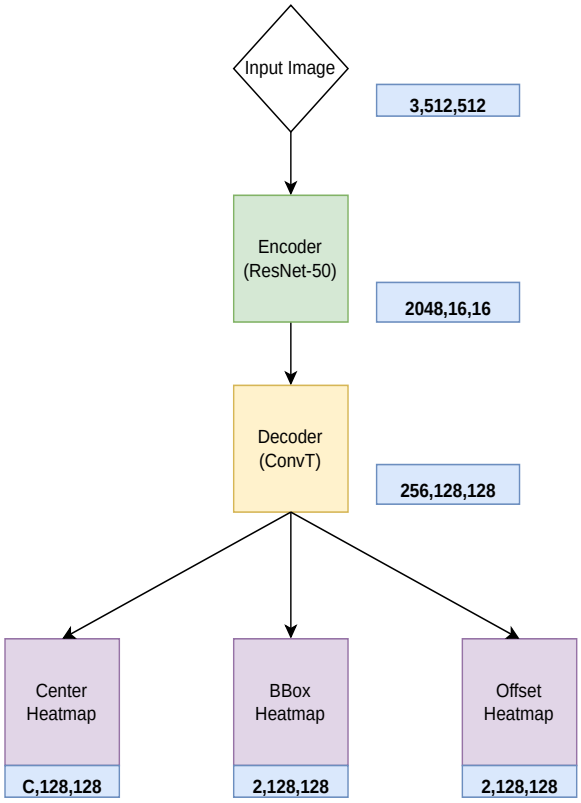


Figure 2.11: **Architecture of CenterNet.** CenterNet consists of an encoder-decoder backbone to extract visual features followed by three detection heads. Center heatmap heads predict the positions of the object center as peaks in the output dimension. Similarly, the BBox heatmap and offset head regress the bounding box dimensions. Since the heatmap dimensions are downsized compared to the input image, the offset head predicts the fractional pixel positions of the object center.

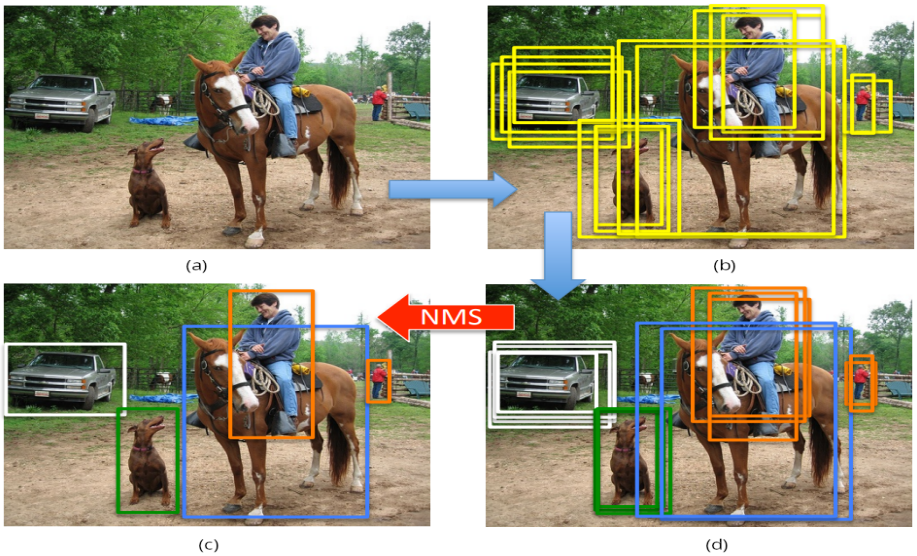


Figure 2.12: **The process of Non Maximum Suppression**. Subfigure (a) is the input image fed into the model. Subfigure (b) visualizes the different bounding box proposals without considering classes. Subfigure (c) distinguishes bounding boxes based on the proposed class. After processing through NMS, subfigure (d) shows that only the most appropriate bounding box was retained, and all the redundant, overlapping boxes were eliminated. Image Source: [3].

Algorithm 1 Non Maximum Suppression

Input : A List of Bounding Box Predictions B with corresponding confidence score $B.S$, IOU threshold T
Output : A List of Filtered Bounding Box Predictions P

```
procedure NMS(B, S)
    P = EmptyList()
    while B not NULL do                          ▷ Iterate till Bounding Boxes remain.
        B_max = B[ArgMax(B.S)]                   ▷ Pick Bounding Box of Highest Score.
        B.Remove(B_max)                          ▷ Pick Bounding Box of Highest Score.
        P.Append(B_max)                          ▷ Add it to list of filtered boxes.
        for B_i ∈ B do                           ▷ Iterate over the remaining set of Bounding Boxes.
            IOU_{P,B_i} = CalculateIOU(P,B_i)    ▷ Calculate IoU for each pair.
            if IOU_{P,B_i} > T then
                B.Remove(B_i)                    ▷ Remove Boxes with IoU greater than set threshold.
            end if
        end for
    end while
    return P                                     ▷ Return the filtered boxes.
end procedure
```

matrix, the IoU threshold needs to be set. Those predictions with an IoU value more than the threshold are considered positive samples, and those intersections with an IoU value less than the threshold are considered negative samples. This process is also illustrated in figure 2.13. For experiments across the thesis, IoU threshold values are fixed at 0.5, and evaluations are performed only for this IoU value.

2.3 A Review of Few-Shot Object Detection Methods

Few-Shot Learning is a particular case of limited data learning. Limited data learning focuses on methodologies to train neural models with scarce labeled data. Even though the advancement in self-supervised methods has demonstrated strong performance in image classification benchmarks, translating this into a detection task is a much more complex challenge. Moreover, even with the availability of volumes of data, annotating this dataset for object bounding boxes is labor intensive.

Few-Shot Object Detection (FSOD) divides the classes in the dataset into two categories depending on the number of samples that are available to each of them. This splitting of classes simulates real-world scenarios where the universal object class distribution can be considered long-tailed. The first category, called the base classes, has abundant bounding box annotated data. The second category of classes, called Novel classes, is a set of classes with only small instances for the model to train. For example, in most literature studies,

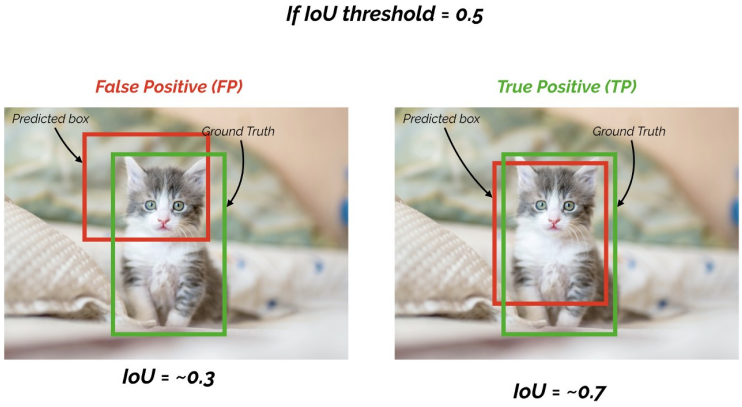


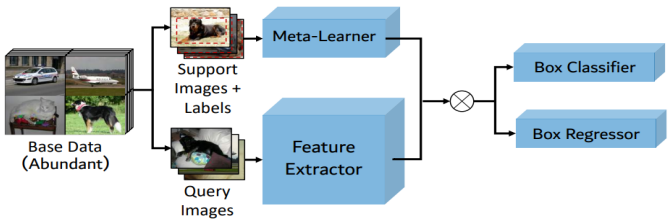
Figure 2.13: **Illustration of positive and negative samples for calculating MAP.** Those bounding box predictions with an IOU value more than the threshold with the ground truth box are considered the positive sample and vice-versa. Image Source: [44].

novel classes dataset are constructed such that each class has 10-30 images as learning support. The fundamental motive of FSOD is to build algorithms such that the performance of a model that is trained on the base class can be adapted to the set of novel classes. Most FSOD methods can be represented as one of the two learning schemes. One is the idea of meta-learning, which uses the concept of class prototypes at its core. The other uses intelligent fine-tuning of components to enable adaption into novel classes.

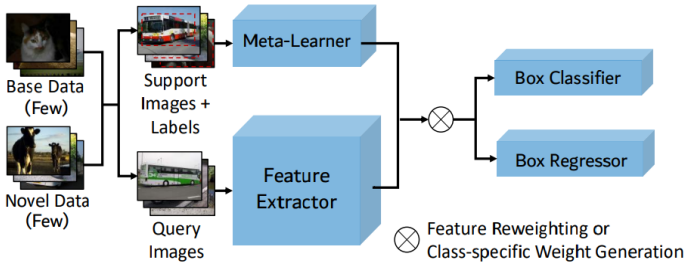
2.3.1 Meta-Learning Based Methods

Meta-learning-based methods are fundamentally motivated by the idea that for each object class, a prototype or a set of prototypes can be used as representative of all the image variations within the class. However, this ideal prototype may not necessarily be described in the RGB color space but also as a discriminative class separable encoding. The learning of this class prototype is aided by a functional block called the Meta-Learner. The distinguishing characteristic of meta-learning-based frameworks is the use of episodic learning. Episodic learning simulates limited image support even for the base classes, making it easier for the model to adapt to similar data-scarce settings in novel classes.

The training happens in two stages, as described in 2.14. The first training step is commonly referred to as base training. The function of this training step is two-fold. One, the meta-learning needs to learn to derive a functional class prototype with a set of support images. Two, from the query images, the feature extractor needs to learn to represent useful visual features that can help the bounding box classifier and detector heads. The base dataset is set up into pairs of query images and support images belonging to the



(a) Base Training Step



(b) Meta Training Step

Figure 2.14: **Meta-Learning Based Methods.** In meta-learning-based architectures, episodic learning is used to train the models. Episodic training rearranges the dataset into pairs of the query image and support images for each object instance in the query image. In the first stage, only the base classes are used for training the extractor, meta-learner, and detector heads to adapt to object detection. In the second stage, a combination of base and novel classes is used to help the model adapt to the novel classes. Images Source: [48].

same class. By feeding these pairs to the meta-learner and the feature extractor, the meta-learner learns to encode these support images. A functional operator such as reweighing or concatenation combines the support encoding with the feature extractor output. This functional operator is designed to highlight only the instances that belong to the same classes on the feature map with the help of support encoding. By training on a set of pairs on the large base training set, the meta-learner becomes an effective support encoder for the base classes. Similarly, the feature extractor effectively provides useful visual features that are compatible with both the detection heads and the meta-learner.

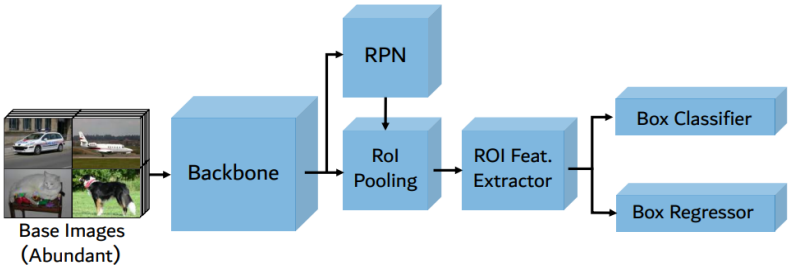
In the second stage of training, referred to as meta-training, both the base and novel classes are used to recreate a set of query and support image pairs. The function of this training stage is to help the meta-learner and feature extractors to adapt to novel classes. The mixture of the base classes is also added to avoid catastrophic forgetting. Catastrophic forgetting is a phenomenon in which the model can no longer perform effectively on the base classes due to completely overfitting the novel class.

Most of the state-of-the-art meta-learning-based detectors vary on the two aspects. The first is how the meta-learner block is modeled, and the second is the functional operator that combines the meta-learning encoding with the feature extractor. For example, Kang et al. extended YOLO [40] architecture to few-shot detection using a Feature Reweighting Module [21] module as the meta-learner block. This meta-learner block learns an embedding that signifies the weight that needs to be provided to each class based on the presence of the object instance. Yan et al. proposed the Meta-RCNN [52] framework with Faster-RCN as the detection module. Here, a Predictor Head Remodelling Network was proposed to perform the same function as the Feature Reweighting Module for a two-stage network. Fan et al. extended this further with a Multi-Relation Detector [14] that used contrastive losses between the support and query encoding to optimize class separations in the latent space. More recently, Zhang et al. proposed Meta-DETR [58] that demonstrated substantial performance gains in benchmarks using vision transformer-based encoders. Additionally, a Correlational Aggregation Module based on an attention mechanism was proposed as the meta-learner.

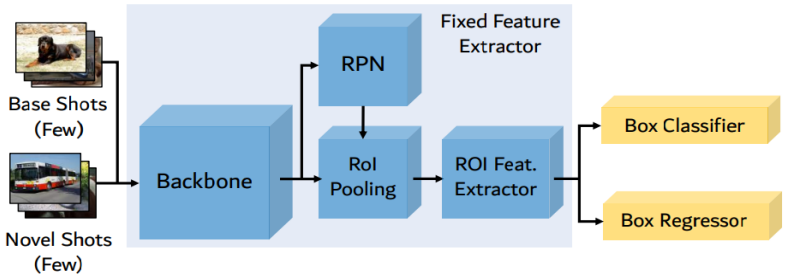
2.3.2 Fine Tuning Based Methods

While meta-learning-based methods have demonstrated strong performance in the few-shot object detection benchmarks, they come with the added complexity of meta-learning modules and the feature aggregation function. Fine-Tuning based methods circumvent this by taking a more simplistic approach to few-shot detection. The fundamental assumption that these approaches take is the existence of a set of learnable weights such that the performance on the novel and the base classes can be equally optimized. Therefore, by intelligent training optimizations, the model can learn this exact solution without the requirement of additional modules.

As shown in the figure 2.15 for a Faster-RCNN-based detection framework, the training again happens in two stages. In the first stage, the detector trains on the set of base classes.



(a) Base Training Stage



(b) Fine Tuning Stage

Figure 2.15: **Fine Tuning Based Methods.**. Fine-tuning-based methods rely on intelligent hyperparameter tuning to converge to an optimal solution that performs well on the base and novel class. In the first stage of training, the model is trained only on the base class, as in the traditional object detection paradigm. The second stage includes a mixture of base and novel classes for training. This helps the model also adapt to the novel classes. Image Source: [48].

This training stage is to help the backbone, Region Proposal Network, and the detection heads to learn useful visual features for detecting objects. This step is no different from training a Faster RCNN network in a traditional non-few-shot training paradigm. A balanced dataset of both the base and novel classes is sampled in the second stage. Unlike the previous stage, specific components of the module, such as the backbone and region proposal network, are frozen. This is based on the intuition that visual features and region proposals generated are independent of the classes. Therefore, these two modules do not require retraining. Only the detection heads are trained during the fine-tuning stage to adapt to the novel classes.

The methods that rely purely on fine-tuning are far and few. Finding an optimal solution that works on the novel and base classes is fundamentally complex. This requires precise and intelligent fine-tuning that is hard to control in iterative gradient propagation schemes. Fine-tuning-based methods were introduced by Wang et al. with the Frustratingly Simple Few-Shot Object Detection [48] framework and demonstrated significant performance on the benchmark datasets. However, to the best of author's knowledge, no other methods that only use fine-tuning have demonstrated remarkable results on the common benchmark datasets.

2.3.3 Other Notable Works

Apart from the meta-learning and fine-tuning approach, some similar research directions are also quite complimentary to the fundamental motivations of few-shot object detection. Incremental Few-Shot Object Detection is an extension of few-shot object detection where the novel classes are not fully limited to a known set of classes. Even after fine-tuning on a set of novel classes, the model must adapt to newer novel classes when and if they may arise. This is a much harder problem to tackle since the models must also remember the classes that were previously learned. Pérez-Rúa et al. proposed the Open Ended CenterNet (ONCE) [37] to approach this problem with a meta-learning-based method and is considered a pioneer for introducing this heavily constrained learning paradigm. One of the important highlights of this work was the use of CenterNet [60] as the base detection architecture, which was quite unusual in studies of few-shot object detection.

Other works focusing on open vocabulary object detection problems are also relevant to meta-learning-based methods. These models aim to localize and detect the position of an object by using prompts to specify which class to focus on. With the availability of large-scale zero-shot capable multi-modal like CLIP, several architectures have been proposed to take advantage of these pre-training schemes. Gu et al. proposed Vision and Language knowledge Distillation (ViLD) [16] that combined the discriminative CLIP embedding into a Faster-RCNN detection backbone towards open-world adaption. More recently, Zhong et al. adapted the pre-training methodologies for CLIP by matching the text-image pairs at the level of cropped object instances instead of complete images. The proposed RegionCLIP [59] improved the downstream detection further than the vanilla CLIP backbones.

3 Few Shot Image Classification

3.1 Problem Definition and Goals

Any generic object detector has two fundamental challenges. First is the precise localization of the object proposal regions. The second is classifying the proposed object region into the accurate semantic class. This thesis section focuses on the classification task in object detection pipelines. The algorithms and experiments described in the section aim to perform few-shot classification from the detected object proposals, assuming methods to identify the object proposals from images are immaculate and precise.

Given a query image Q and k support samples $S = \{S_j\}, j \in 1, 2, \ldots, k$ for each of the N classes, determine the class query image Q belongs to among the N classes. This problem is generally referred to in the literature as N way k shot detection. Here, the support samples can be both images and textual descriptions of the classes. To ascertain the few shot classification performance of a different training methods for image classification, the performance is evaluated on the following neural models:

1. Self-Supervised Learning - DINO [6].
2. Multi-Modal Learning - CLIP [38].
3. Supervised Learning - ResNet50 [19], ViT B-16 [10].

The selection of these models is based on the training paradigm that each used and the demonstrated benchmark performances shown by them. Therefore, the performance of the individual models is considered representative of the training method it uses. As congruent with the aim, a dataset that is designed for image-level classification benchmark is chosen. This dataset chosen is class balanced and converted into a few-shot classification dataset. A more detailed discussion follows in section 3.2.3. Since the dataset is class-balanced, classification accuracy serves as a suitable metric to assert the performance of the models. In the cases of class-imbalanced datasets, accuracy would not have been a justifiable metric. However, this is not applicable in this experimental setup.

3.2 Methodology

This section deals with data flow from images and class to a comparable metric across the different learning paradigms. First, the choice of dataset and the process of converting it to a few-shot compatible dataset is discussed. This is followed by the introduction of the

concept of class codes and methods to extract them from visual or text inputs. Finally, the proposed method for class assignments based on the class codes is illustrated.

3.2.1 Choice of Classification Dataset

The dataset used for classification is a crucial choice for the few-shot classification experiments and for building the detection modules. The most accurate classification model forms the basis for the object detection architecture, as described in the section 4. The classification performance is heavily dependent on the experimental dataset used. Fundamentally, since not all datasets are suitable for the few-shot training paradigm, the insights may not be fully transferable across all image classification benchmarks. There were three main considerations when selecting the datasets, as follows

1. The dataset must primarily contain only one object instance in an image. This requirement comes from this chapter's fundamental motive: that the images in the dataset are simulated to be the output from an ideal object proposal generator. Having multiple instances of an object in an image negates this assumption. An ideal object proposal generator would only output one object instance in a proposed region.

2. The classes in the dataset must have low inter-class visual variance. To illustrate this with an example, a cat and a horse have considerably large visual distances, whereas a black horse and a white horse are close to each other on the visual scale. This requirement comes from the application use cases at Precibake. Most of the actual product classes look similar to each other. To improve the transferability of the proposed models into use cases, datasets with similar-looking object classes are more appropriate.

3. The scale of the dataset must be suitable for fast experimentation based on the practical GPU hardware constraints. Since the experiments act as a test of the algorithms than pure performance tuning with optimal hyper-parameters, the scale of datasets consisting of around 10k images is optimal.

After a thorough investigation of various datasets that are frequently used as benchmarks, the Caltech-UCSD Birds 200 Dataset [50] was deemed most suitable. More details on the dataset follow in the next section.

3.2.2 Caltech-UCSD Birds 200 Dataset

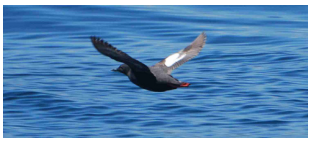
The dataset consists of images of primarily North American birds belonging to 200 species. Each of the species of the bird becomes the classes into which they are divided. Sample images from the dataset are illustrated in figure 3.1. The figure shows that the visual distance between images of birds belonging to different classes is relatively small. This satisfies the first requirement from the previous section. Additionally, most of the images

(a) Black Footed Albatross

(b) Frigatebird

(c) Herring Gull

(d) Pelagic Cormorant



(e) Pigeon Guillemot

(f) Slaty Backed Gull

Figure 3.1: **Samples from Caltech-UCSD Birds 200 datasets.** Some images from the dataset and the classes are visualized. These samples illustrate how the classes are visually close to each other. Even for top human performance benchmarks, this task requires specialized domain knowledge. Image Source: [50].

in the dataset consist of only one instance of a bird in an image, gratifying the second requirement. From the total of 11788 images belonging to different classes, the training, validation, and test split are made in the ratio of 60 percent, 20 percent, and 20 percent of images, respectively. To preserve the class distribution of the dataset, the ratios are applied for every class. The dataset was well-balanced before the splitting. This balance is also thus conserved after splitting.

3.2.3 Preparation of a Few Shot Classification Dataset

The few-shot classification paradigm operates on disjoint sets of base and novel classes. Base classes have ample training data for the models to learn from, whereas novel classes have only a few images or text describing them. For the experiments, a specific case of the few-shot classification is explored. All the 200 classes of the CUBS200 dataset are considered novel classes. The classes the experimental model was pre-trained on are the base classes. These base classes vary for each of the models used. For example, most large-scale pretraining datasets such as ImageNet [9] consists of everyday context objects like airplanes, cats, or dogs. Few-shot classification framework additionally requires the definition of support and query images. The query image is the test image to which the class needs to be assigned. Support images assist the model in identifying the class the query

image objects belong to. In the following experiments, the test split becomes the query images, and the support images are picked from the training set.

3.2.4 Concept of Class Codes

The objective of class code is to condense the visual or textual information of all the possible variations of objects belonging to a class into a unifying embedding. Given that this unifying embedding is in a common latent space across classes, it is straightforward to compute the similarity between the classes directly by cosine distance. As elegant as the idea of class code sounds, extracting such descriptive ideal embedding for every class is challenging. To map every visual variation of an object class into a singular representation requires a powerful many-to-one mapping function. One feasible candidate to derive such a mapping is using a powerful image or text encoder. This way, the mapping function becomes agnostic to different classes and unifies representations into a common latent space.

For experiments with self-supervised and multi-modal learning paradigms, the mapping functions are the models themselves. The output feature map from the encoder is considered the class code. This intuition comes from the demonstrated ability of these models to adapt to new dataset classes without retraining the encoder. Even though these encoders may not provide the ideal mapping across all possible classes, they are still viable candidates to experiment with and improve upon. This transferrable mapping performance results from the large-scale datasets and the long training schedules that these encoders have been trained with. More details on these encoders are discussed in sections 2.1.3 and 2.1.4. For experiments with fully supervised learning, these class codes are essentially the integers assigned to each of them as labels. Even though these integers are of the position in the output one-hot encoding of the target label, they are the most simplistic formulation of the idea of class codes, with each of the classes separated by a unit distance.

3.2.5 Few Shot Classification through Class Codes

With this concept of class code, it is now possible to reconceptualize the few-shot paradigm. Instead of query input as an image, it is encoded into query class code. Similarly, the support images or texts are also encoded as class codes into the same latent space. More details on this extraction of support class codes are discussed in the sections 3.2.6 and 3.2.7. Once both the query and support class codes are in a common latent space, the assignment of query class code becomes direct. By comparing the cosine distance between the query class code and each support class code, the query class code is assigned the class with the least cosine distance. This process is formalized in the algorithm 2.

The class codes for self-supervised and multi-modal models are the output feature map of respective encoders. The CLIP model supports text and visual inputs, whereas DINO only works with image inputs. In fully supervised models, there is no explicit requirement

for class code. Instead, the classes are directly assigned through the peak of a softmax function.

Algorithm 2 Classify the Query Image

Input : Query image Q, Support embedding S, Embedding function f
Output : Class k of the query image Q

procedure CLASSIFYQUERY(Q, S, f)
    q = PreProcess(Q)
    $\phi = f(q)$ ▷ Get the Embedding of the query image
    for $S_c \ in \ S$ **do** ▷ For all Classes
        $\Delta_c = CosineSimilarity(S_c, \phi)$ ▷ Calculate similarity score
    end for
    $k = argmax(\Delta)$ ▷ Assign class to maximum cosine score
    return k ▷ Return the class index
end procedure

3.2.6 Extraction of Image based Support Class Codes

For CLIP and DINO encoders, the support class codes are generated from the training split of the dataset. Suppose more than one support image is used for code generation. In that case, the extracted feature map is averaged across the support samples to maintain the dimension of support embedding, irrespective of the number of images used for support generation. This process is formalized in algorithm 3.

Algorithm 3 Extract Image Support Information

Input : Support images S, Embedding function f, Number of supports k
Output : Support embedding of all classes Ω

procedure EXTRACTIMAGESUPPORT(S, f, k)
    for $c \ in \ N_c$ **do** ▷ For all Classes
        for $i \ in \ range(k)$ **do** ▷ For all support images
            S_c = PreProcess(S_c) ▷ Preprocess the Image
            $\Omega_{ci} = f(S_{ci})$ ▷ Calculate the embedding of the image
        end for
        $\Omega_c = mean(\Omega_{ci})$ ▷ Average over class supports
    end for
    return Ω ▷ Return the support embedding
end procedure

Part	Attributes	Part	Attributes	Part	Attributes
Beak	*HasBillShape, HasBillColor , HasBillLength*	**Back**	*HasBackColor, HasBackPattern*	**Breast**	*HasBreastPattern, HasBreastColor*
Belly	*HasBellyPattern, HasBellyColor*	**Fore-head**	*HasForehead Color*	**Bird** (all parts)	*HasSize, HasShape*
Throat	*HasThroatColor*	**Nape**	*HasNapeColor*	**Head**	*HasHeadPattern*
Crown	*HasCrownColor*	**Eye**	*HasEyeColor*	**Leg**	*HasLegColor*
Tail	*HasUpperTailColor, HasUnderTailColor, HasTailPattern, HasTailShape*	**Wing**	*HasWingPattern, Has WingColor, HasWingShape*	**Body**	*HasUnderpartsColor, HasUpperPartsColor, HasPrimaryColor*

Figure 3.2: This table shows the list of attributes that the dataset provides for each of the bird classes. Image Source: [50]

3.2.7 Extraction of Text based Support Class Codes

This section deals with generating text-based support class codes using the multi-modal model CLIP. Unlike extracting support codes using images, extracting text features require the input text prompts to be defined. The performance of the text encoder heavily depends on these text prompts. Furthermore, the more descriptive these texts are, the performance of few-shot classification is enhanced.The CUBS 200 Dataset [50] provides attributes describing each of the bird classes. The list of attribute keys provided in the dataset is in figure 3.2. An important thing to note is that the dataset provides probabilities for all attribute key-value pairs instead of boolean values for every possible class. These probability scores help filter for the most correlated attributes. Two approaches were demonstrated to generate text prompts for encoding the novel classes. One was automatic sentence generation. The other was handcrafted feature selection before generating the sentences.

Automatic Text Prompt Generation

In the case of automatic text generation, the attributes are selected based on the top probability scores. The number of attributes (k) used can be considered equivalent to the number of support images used. For experimentation, k with values of 0, 1, 3, and 5 are tried out. The first sentence is always of the template "This is a Photo of a *class name*." For each of the selected attributes, an additional sentence follows with the template "It has a *attribute value attribute key*." For example, "This is a Photo of a Black-footed Albatross. It has black eye color" .

Hand-Crafted Text Prompt Generation

An additional filter was supplemented in the attribute selection process to optimize the text generation process further. As seen in figure 3.2, some attributes are not visually

descriptive. For example, the attribute Bird-HasSize had the size provided in inches. Even in the case of visually descriptive attributes such as the Tail-HasUnderTailColor, these may not be the most effective descriptors, irrespective of the probability score it holds.

To overcome these limitations of automatic selection, the five most visual attributes were chosen before sorting by the class-level probability scores. These selected attributes are Wing-HasWingColor, Eye-HasEyeColor, Body-HasPrimaryColor, Leg-HasLegColor, and BeakHasBillColor. With handcrafted feature selection, the intuition was to improve the descriptiveness of the support class codes and, ultimately, few-shot image classification. Once the text prompts were generated using the above methods, the support class codes were generated by tokenizing the text and subsequent extraction from the text encoder. This process is formalized in the algorithm 4.

Algorithm 4 Extract Text Support Information

Input : Support text S, Embedding function f
Output : Support embedding of all classes Ω

1: **procedure** EXTRACTTEXTSUPPORT(S, f)
2: **for** $c \ in \ N_c$ **do** ▷ For all Classes
3: S_c = PreProcess (S_c) ▷ Tokenize the Text
4: $\Omega_{ci} = f(S_{ci})$ ▷ Calculate the text embedding
5: **end for**
6: **return** Ω ▷ Return the support embeddings
7: **end procedure**

3.3 Experiments and Results

With the re-imagined few-shot paradigm, given a query image, the classes are assigned using either textual support class codes as well as visual support class codes. The further sections discuss the setup of the experiments and the results inferred from them.

3.3.1 Experiment 1 - Few-shot Classification through Text Supports

In the first set of experiments, the performance of the Multi-modal learning method CLIP is evaluated for its few-shot classification performance. The algorithms for extraction of textual support class codes and class assignment are discussed in detail in the sections 3.2.7 and 3.2.5. Figure 3.3 shows the performance of both the automatically generated and handcrafted textual prompts on the test split of the dataset. There are three main observations from this experiment.

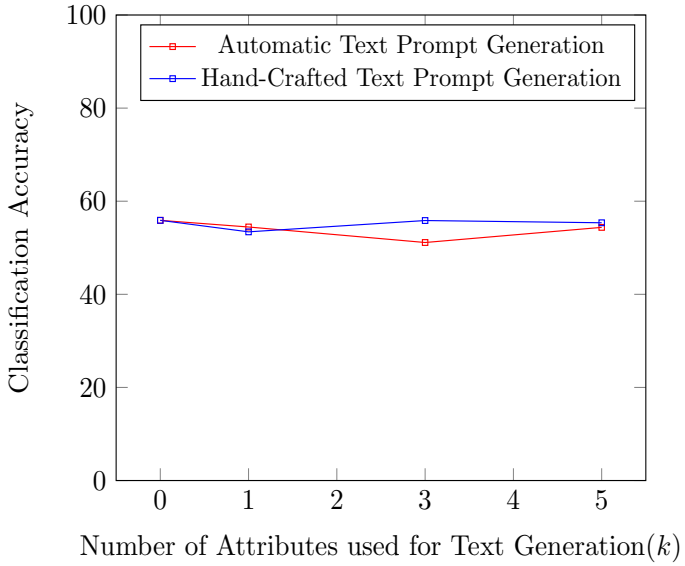


Figure 3.3: **Performance of Textual Supports on Few-shot Classification**. Class names are the most critical textual prompt for classification accuracy. Adding more textual descriptions only has marginal gains.

1. CLIP provides up to 55 percent accurate zero-shot classification performance.

This result illustrates the efficiency of multi-modal models, in particular, CLIP. Without any visual support and purely driven by text descriptions, the algorithm proposed was able to classify roughly half of the images into the correct classes. Had the results been tested with datasets with common object classes such as Aeroplane, Cat, or Dog, this same accuracy result would have been less impressive. However, this is a noteworthy zero-shot performance in a domain-specific dataset.

2. Class names are the most useful class descriptors. Adding more descriptions does not always translate to large performance gains.

In both automatic and handcrafted experiments, the best performance was extracted when only the class names were used to generate textual prompts ($k = 0$). With an increasing number of attributes, the classification accuracy takes a minor drop in performances on the scale of 1-3 percent. This fundamentally means that the class names, which are the species the bird belongs to, help CLIP almost single-handedly to assign the query image to the correct class. Any additional descriptions, such as the wing color or primary color of the bird, are only of little significance.

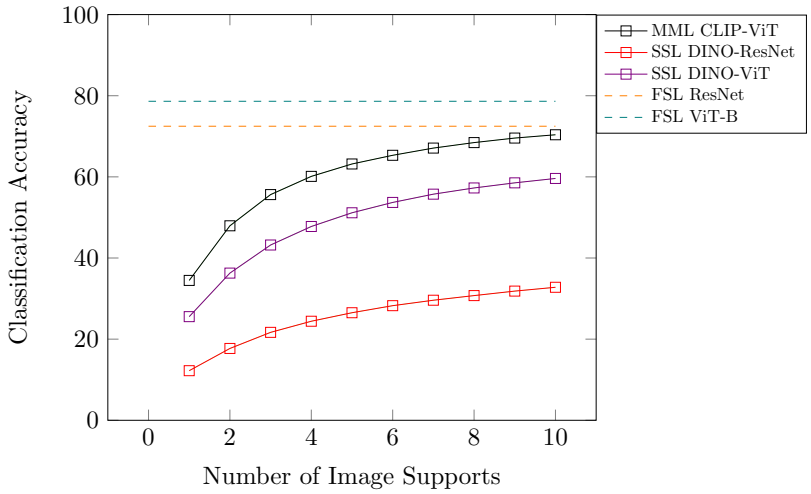


Figure 3.4: **Performance of Visual Supports on Few-shot Classification**. CLIP model can compete with fully supervised models with just 10 support images a class without any fine-tuning on the domain specific dataset.

3. Handcrafted text prompts perform marginally better than automatic text prompts.

The use of handcrafted text prompts had a meager effect on improving accuracy. This is in direct correlation with the previous observation. Other than the class name, other attributes did not serve great importance. Additionally, with all the handcrafted features being color based, there is a plausibility that too much color information can confound the model. The text encoder would require a great semantic language understanding to match the colors to the appropriate parts and output a distinguishable query embedding.

In the next section, a similar few-shot classification was carried out with image supports instead of textual supports. The classification performance is expected to improve since it is easier to correlate images with images than images with text.

3.3.2 Experiment 2 - Few-shot Classification through Visual Supports

In this set of experiments, the support class codes used for assigning the classes are generated from support images belonging to different classes. The algorithms for extraction for visual image supports and the class assignment process are explained in the sections 3.2.6 and 3.2.5. To compare the performance among different paradigms, five models were chosen that were representative of the corresponding training technique.

Fully Supervised Learning (FSL)

In fully supervised trained models, the entire training data consisting of 7129 images are used to train the model. Two variations of the standard classification backbones are used for experimentation. They were based on a ResNet50[19], and Vision Transformer Base-16 [10] model. These models are explained in more detail in the sections 2.1.2 and 2.1.2, respectively. The fully supervised models are expected to be the upper bound on the classification performance since they are trained on the complete CUBS200 training set. This is in stark contrast to the self-supervised and multi-modal models, where no fine-tuning was done on the CUBS 200 dataset.

Self Supervised Learning (SSL)

As a representative model for self-supervised training, DINO [6] is used to evaluate the few-shot performance. The official repository provides both a ResNet50 as well as a Vision Transformer Base-16 based parameter weights. They are used directly for experiments without fine-tuning on the CUBS 200 dataset.

Multi Modal Learning (MML)

Similar to the experiments in the previous section 3.3.1, CLIP is used to test the efficacy of multi-modal training. The fundamental difference here is that the image encoder is used to generate the support codes as opposed to the text encoder that was used in the previous experiment.

Figure 3.4 shows the performance of the different learning paradigms on the test set of the CUBS-200 dataset with varying support images. One important caveat here is that the performance of classification heavily depends on the support image used in the class assignment. More closer the support image is to the query image in terms of color, pose, or photographic style of the image, the easier it is to assign the query image to the correct class. To eliminate any inconsistencies that may occur due to the selection of support images, especially when the number of support images (k) is low, the experiment was conducted over 100 runs, and the mean of the results is reported along with error scores. However, the error bars are almost imperceptible in the figure due to minimal error values. There were four important results from this experiment.

1. The Multi-Modal CLIP model can compete with a fully supervised trained model without finetuning.

Using only ten images supports for generating each class code, the CLIP model can compete with a fully supervised ResNet50 model. This is a compelling performance from the multi-modal model due to two crucial aspects of the experiment. One, the CLIP model was not finetuned for the CUBS-200 dataset, whereas the fully supervised model was trained

on the training split of the dataset. This shows the noteworthy few-shot performance of the CLIP model to translate to datasets with which it was not trained or finetuned. Second is the domain specificity of the dataset. Even with models that can translate well to different datasets, performing successfully in a domain-specific dataset is quite challenging.

2. Increasing the number of support images boosts classification accuracy with diminishing returns.

As seen from the figure, the classification accuracy increases with the number of support images used to generate class codes in both multi-modal and self-supervised models. However, returns in performance diminish with an increase in support images. Intuitively, the performance of the classification directly correlates with how descriptive each of the class codes is and how well separated they are in the latent space. This distinctiveness of classes comes from the ability of the image encoder model to output well-separated latent embedding. This again shows the utility of the CLIP's image encoder. Additionally, since the embedding is averaged over support images, this inherently means that as the number of support image increase, the value of the average embedding approaches towards the ideal class code with reducing errors. This is only possible if the output embedding provided by the model encoder for each of the support images independently is close to the average class code, irrespective of the variations in the pose, lighting, and view of the object variations within the class.

3. Multi-Modal models produce more discriminative class codes than self-supervised models.

Another important observation is that the performance of the self-supervised models is lower than that of multi-modal models. This performance gain in multi-modal models comes from the intelligence that is inherently distilled into human language through natural evolution. This is also discussed in more detail in the section 2.1.4.

4. Vision Transformer backbones are more effective feature extractors than ResNet-based backbones.

In correlation with findings in the recent literature, the vision transformer backbones perform better than the ResNet backbone. This is observed in cases of both fully supervised and self-supervised DINO-trained models. As discussed in the section 2.1.2, this improvement in the classification comes from the attention mechanism that is the foundation of the vision transformer architecture.

3.4 Summary of Few-Shot Classification

This section dealt in detail with the experiments on few-shot classifications across multiple learning paradigms. By reframing the few-shot classification experiments through the lens of class code, experiments were conducted across multiple modalities of the support class codes, particularly vision and text. The details on the generation of query embedding, support class codes, as well as the class assignment process were discussed in detail. The experiments provided us with some exciting results. One, the CLIP model competed with fully supervised models by using just a quarter of the training data without any finetuning on the dataset. This illustrated the power of learning across multiple modalities as well as the discriminative power of CLIP embedding. Two, visual support class codes perform better than text-based support class codes. However, there is also a lot more potential in improving classification accuracy purely using text support class codes, primarily by improving the text generation process and providing image-specific descriptions instead of descriptions for the entire class.

This concludes the experiments on the second part of the pipeline, where the focus is on building a few-shot algorithm that classifies the query images into the correct classes from the proposed object regions. In the next chapter of this thesis, the impetus is to use the CLIP visual encoder to build an object detector that can provide accurate object proposals.

4 Translating Few Shot Classification to Few Shot Detection

4.1 Problem Definition and Goals

This chapter deals with the challenge of localizing the object proposal regions in images. One of the important insights from the last chapter was the effectiveness of the multi-modal model, CLIP, for few-shot object classification. Thus the fundamental motive is to build a detection module that can utilize the class codes generated from CLIP for few-shot classification. If the detection modules are class agnostic and since the classification modules are already verified for the few-shot performance, the entire framework combined completes the few-shot object detection pipeline. This proposed object detection framework is referred to as CLIPtheCenter. Given a set of training images belonging to the base classes B with abundant labeled data and only k labeled images of novel class N, the aim is to build a detector that can localize and classify the instances of both the base and novel classes in a query image, subject to the following constraints:

$$1 \leq k \leq M \tag{4.1}$$

$$B \cap N = \emptyset \tag{4.2}$$

The value of M is the least maximum number of possible support images available for all the novel classes. In practical applications, this comes from the sampling strategies that are used to generate the dataset. Additionally, some classes are not frequently observed due to naturally occurring category distribution. Thus the value of M will depend on the combination of these two factors. For the experiment, the value of M is chosen as 30, in alignment with the state-of-the-art benchmark studies, as most illustrate the model efficiency with 10 or 30 support samples a class.

4.2 Building a Few Shot Object Detector Ground Up - CLIPtheCenter

This section describes in detail the intuition behind the architecture of CLIPtheCenter as well algorithmic tests for verification of the design. Once the model's performance is demonstrated, additional experiments are designed to check its generalization ability to larger, more challenging datasets.

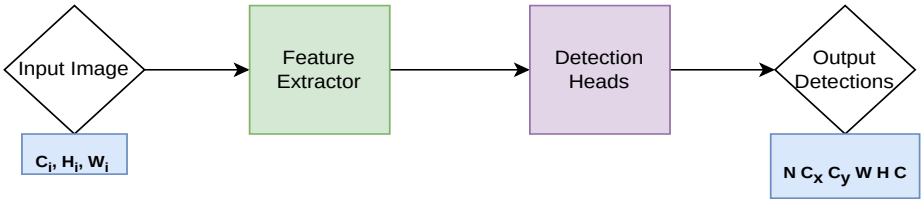


Figure 4.1: **Abstraction of Object Detection Modules.** Most popular single-stage object detection architectures can be decomposed into a feature extractor and a detection module consisting of different heads. These same fundamental modules are used to build CLIPtheCenter. Here, C_i, H_i, and W_i represent the input number of channels, height, and width of the image, respectively. Each of the N proposals in the output detection is represented by the object center coordinates C_x, C_y in the width and height coordinates along with the object proposal width W and height H. Additionally, the class C of the object proposal is also classified into a class, denoted by C.

4.2.1 Architectural Design

The fundamental design of the CLIPtheCenter is derived from the CenterNet[60]. A detailed discussion of CenterNet is in the section 2.2.2. There were two main reasons for this choice. One, for higher inference performance measured as the number of image frames processed per second, single-stage detectors are most suitable. Two, as seen in figure 2.10, even though CenterNet is a single-stage architecture, CenterNet demonstrated more precise object detection performance comparable to large and more computationally expensive multi-stage architectures. However, CLIPtheCenter was built from scratch, only using the spirit of its design. As shown in figure 4.1, the most commonly used single-stage architectures, including CenterNet, can be decomposed into two modules. These two modules are the feature extractor and the detection head.

Feature Extraction Module

The feature extraction module's utility is learning descriptive visual features from an image. Here, the visual feature can be a very primitive visual feature such as a line, spline, or circle, as well as more significant context features such as object boundaries. If the feature extractor also learns through the backpropagation of gradients when training a model, then the model determines the features to inherently lower the loss function. In the original CenterNet model, the feature extractor is an encoder-decoder neural network. The purpose of such a design is to condense all the visual information into a smaller dimensional latent space and upsample these latent features into a large scale for downstream postprocessing. The encoder takes in a three-channel RGB image input and distills it into the higher channel but much lower resolution feature maps. The decoder takes these as

input, which usually processes it to provide a much smaller channel and higher resolution feature map. For practical reasons, the input dimension of the image is a multiple of the output dimension of the decoder. This enables the estimation of correspondences between the input and output pixels, if necessary.

The CenterNet architecture provides four different feature extractors, which are Hourglass-104 [35], DLA-34 [56], ResNet-101, and ResNet-18 [19] as the feature extractors. Each of these backbones is discussed in detail in the section 2.1. Out of these four possible choices, only Hourglass and DLA-based architectures take into account the hierarchical representation of features of different scales. This is particularly useful in instance-level tasks such as object detection and semantic segmentation. In image-level tasks such as classification, the features are required only in the global context of the image. This is quite contrastive with object detection, where the features need to be localized due to the possibilities of multiple object instances in an image. To capture objects of different scales with respect to the image size, features with different receptive field scales must be used to calculate the detection parameters. With this possibility of multi-scale object instances in consideration for building a feature extractor, a U-Net [42] style encoder-decoder is used. U-Net is a heavily cited architecture proposed for medical image segmentation and then adapted to similar tasks in the RGB image domains. It uses a staged downsampling and a subsequent upsampling strategy with the cross-concatenation of the feature maps between each encoder to the corresponding decoder stage. This process is also explained in detail in the section 2.1.

Detection Heads

The detection heads perform the crucial task of transforming the feature maps of the decoder into different object proposals. The object proposals consist of the bounding box localization coordinates and the classification of the proposal. In CenterNet, the bounding box coordinates are parameterized as the tuple of the bounding box center coordinates, width, and height of the bounding box. The CenterNet architecture consists of three heads: Center Heatmap head, Bounding Box Heatmap Head, and Offset Heatmap Head. The center heatmap provides probability scores for each pixel of the feature map being an object instance center. Similarly, the bounding box and offset heatmap provide the dimensions of the bounding box and offset values for each feature map pixel.

In the design of CLIPtheCenter, the bounding box heatmap head and center heatmap are used with the same functionality as that of the CenterNet. One notable difference in the center heatmap is that the center heatmap is made class agnostic by using only one center heatmap filter in CLIPtheCenter as opposed to a center heatmap filter for each of the object classes in CenterNet. Since the bounding box heatmap and center heatmap are class agnostic, the challenge of assigning the classes for each proposal lies with the Region of Interest (RoI) heatmap head. The RoI heatmap head acts as a precursor for the Pooling Encoder to learn the CLIP embedding for each proposed object region. This is discussed in more detail in the upcoming sections.

The idea of an offset head is no longer required since the dimensions of each heatmap head are the same as that of the input image dimension. This is a deliberate choice made not to downscale the heatmap dimension due to two significant reasons. One, the pixel correspondences are preserved by retaining the same dimension as input for each output heatmap. Thus, there is no need for an additional pixel adjustment through an offset head. This reduces the amount of information the model needs to learn, thus improving the learning potential. Two, downscaling the heatmap dimensions makes it harder for the model to learn due to a decrease in the number of learnable parameters compared to a non-downscaled counterpart. The model does not have to distill information in an image region into a smaller output pixel region and thus learns more pixel-level information, which is helpful for instance-level tasks.

Introduction of the Pooling and CLIP Encoder Modules

The Pooling and CLIP encoders provide the few-shot capability to the CLIPtheCenter model. The CLIP encoder is the frozen CLIP Vision transformer-based encoder on which the previous experiments of few-shot classification described in the section 3.3.2 were performed. From a knowledge distillation viewpoint, the CLIP Encoder acts as a teacher model while the Pooling Encoder acts as a Student model. The fundamental motive of aligning the outputs of these two encoders through a cosine embedding loss is to train the pooling encoder to learn to output the CLIP embedding for each object proposal. Once the model is fully trained, the output of the pooling encoders can be directly used for classifying the object region proposals. This process is described in more detail in the next section.

4.2.2 Training and Inferring Object Proposals from CLIPtheCenter

From the individual modules described in the previous sections, this section discusses in detail the training and inference process of the proposed CLIPtheCenter model. The flow of the image can be visualized as the composition of two different functions that finally classify and localize the object proposals. This is illustrated with a sample input image in figure 4.3. This illustrates the ideal case, where the detections are perfectly localized from the model.

Detection Data Flow

The detection data flow transfers the input RGB image to a set of object proposal bounding boxes along with a score for each proposal. This score indicates the probability that the proposal contains the object, also referred to as the objectness score. This function of CLIPtheCenter resembles the detection process of the CenterNet. From the input RGB image, the features are extracted from a learned encoder-decoder-style neural model. With these features, the center heatmap and bounding box heatmap learns to predict the probability of

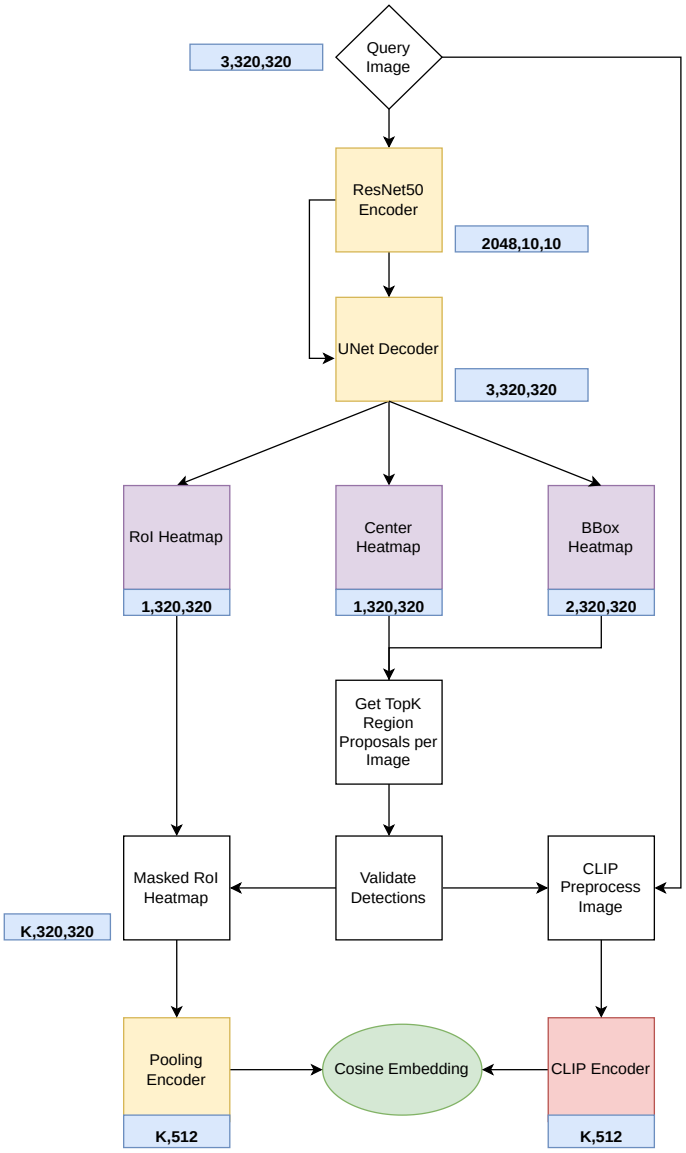


Figure 4.2: **Architecture of the proposed CLIPtheCenter.** CLIPtheCenter adapts CenterNet for a few-shot paradigm primarily by addition of a RoI Heatmap head and an auxiliary branch to generate CLIP embedding for targets for the pooling encoder.

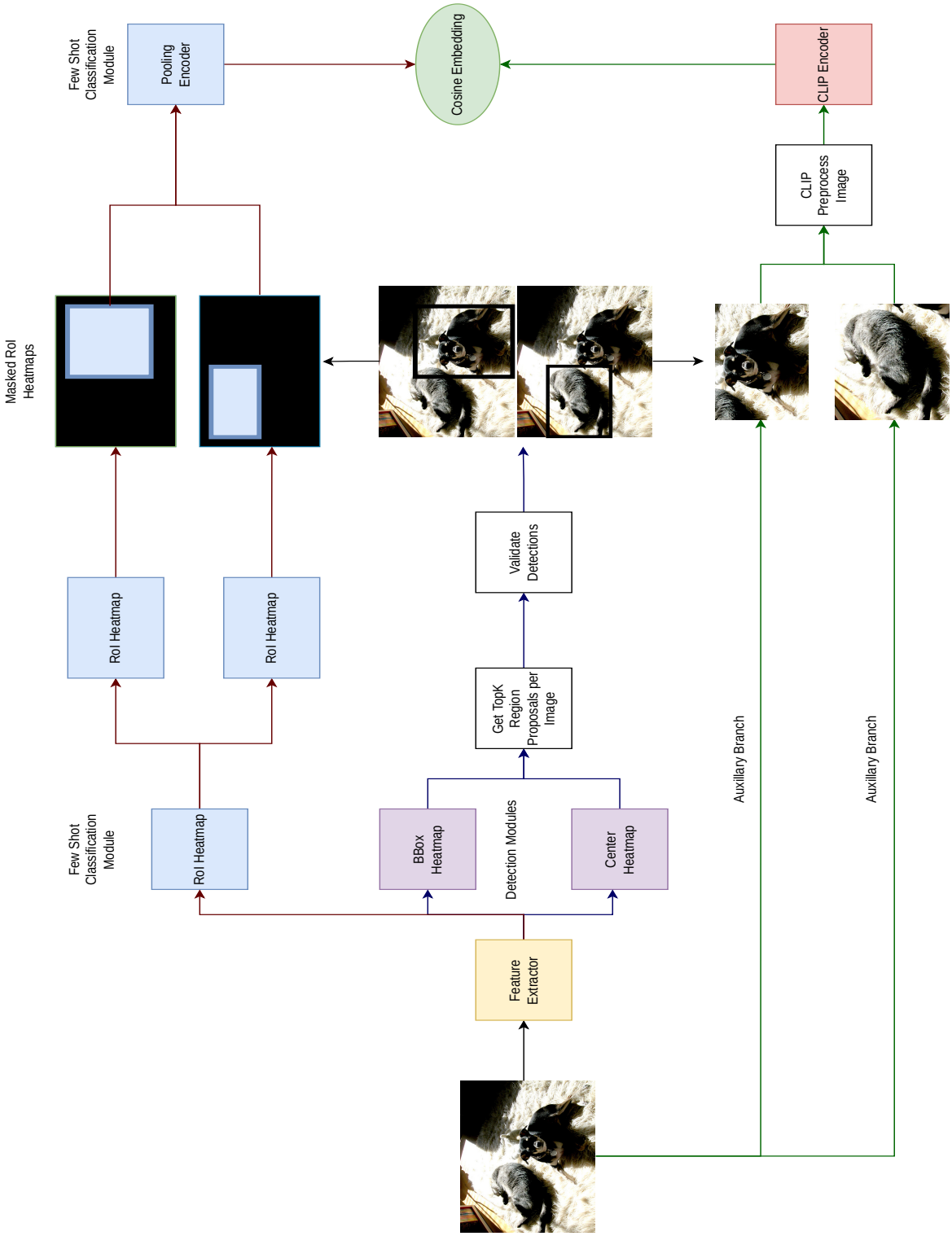


Figure 4.3: **Sample illustration of tensor flow in CLIPtheCenter.** In the example shown, the top two proposals are retrieved from the bounding box and center heatmap. The RoI heatmap uses these two object proposal coordinates to generate a binary mask for each proposal before feeding it to the pooling encoder. Similarly, the output of object proposals is also cropped out of the original image to calculate the CLIP encodings that act as targets for cosine embedding loss. The input image is taken from [13].

the object center and bounding box values for each pixel in the input image. To find the object proposal coordinates, the positions with high values of the center heatmap are extracted along with the corresponding values of bounding box coordinates in the same pixel positions. Thus, the probability of the pixel being the center of an object is taken as the objectness score of the proposal in the model. The number of proposals extracted from the center heatmap is a hyperparameter of this model. For training, fewer proposals are used to reduce the training time and GFLOPS. However, more proposals are preferred during inference, and with additional post-processing methods like Non-Maximum Suppression, the overlapping and redundant bounding box can be filtered out.

Additionally, the derived bounding box coordinates are validated for inherent boundary conditions that come with it. The Validate detections module has two primary operations. One, it ensures that the minimum dimensions of the bounding box are at-least 10 pixels each. This is useful, especially during the initial training step, where the regression of the width and height values can be error-prone and provide too low values. Two, the module confirms that the object proposals are within the image size. In the cases where the bounding box proposal extends outside the image space, they are truncated up to the size of the image. At the end of the detection flow, the bounding box coordinates of the top object proposals are extracted.

Few Shot Classification Data Flow

The few shot classification modules in CLIPtheCenter are derived from the algorithms proposed in the classification experiments on the CUBS-200 dataset, described in the section 3.3.2. From the input image passed into the feature extractor, the object proposals can be constructed as described in Detection Data Flow. The fundamental motive of the few shot classification modules is to learn the CLIP embeddings of these object proposals. Once these modules learn the CLIP embeddings of the object proposals, the classification of the proposal into individual classes can be performed using the algorithm 3.2.5 as a downstream task. To generate the CLIP embeddings, the frozen CLIP encoder is used in the auxiliary branch of the CLIPtheCenter. First, cropped region proposals are generated using the output of detection modules and the input image. These cropped images are then preprocessed in accordance with the requirement of the CLIP encoder by re-sizing the image and adjusting the mean and standard deviation values for each channel.Finally, the target CLIP embeddings for training the Pooling Encoders are calculated by passing the processed image into the encoder.

The RoI heatmap and the pooling encoder aim to learn the CLIP embedding of the object proposals. Once the model is fully trained, the output of the pooling encoder must resemble that of CLIP. These outputs can be directly used for few-shot classification instead of an additional downstream step to classify the object proposals. This would also eliminate the need for the auxiliary branch containing the CLIP encoder, improving the inference speed. The RoI heatmap is repeated as many times as the number of proposals generated from the detection module. In each RoI heatmap filter in the repeated stack generated, the

values are activated only within the regions proposed and made zero outside the bounding box proposal. This is done by multiplying a binary mask of the regional proposal with the corresponding RoI heatmap filter layer. This stack of masked RoI heatmap is fed into the pooling encoder, which outputs an embedding of the same shape as CLIP. With the outputs of the pooling encoder and the CLIP encoder, the losses can be calculated using a cosine similarity function.

The process is formalized in the algorithm 5. In the following algorithm, the input image is represented as I along with the ground-truth bounding boxes B. The output bounding box coordinates are represented as B_o, along with the corresponding embedding output from the pooling encoder as Ω. The feature extractor is modeled as function F. The center heatmap head, bounding box heatmap head, and the RoI heatmap head are denoted as H_C, H_B, and H_R, respectively. The CLIP encoder is denoted as F_{CLIP} while the pooling encoder is denoted as F_{POOL}. The stage variable S signifies if the model is training or in inference.

4.2.3 Algorithmic Test on Single Instance Object Detection

The idea behind devising this test is to check for the algorithmic sanity of the proposed model. The model overfits to the dataset's training split, and the results are evaluated on the same training split. This acts as a fundamental test of the proposed algorithm before moving on to larger-scale training.

Dataset, Model and Metrics

In accordance with the goal of the test, a miniature dataset of three images consisting of three classes- an airplane, a sheep, and a dog, is designed. These images are shown in figure 4.4. These images are picked from the PASCAL VOC 2012 dataset [13]. The details of the architecture used for the experiment are presented in table 4.1. Mean Average Precision(MAP) is used to evaluate the algorithmic test's performance. Mean Average Precision is the standard benchmark criterion for object detection tasks. This evaluation metric is also discussed in detail in the section 2.2.3.

Results and Discussion

The CLIPtheCenter model was trained for 1000 epochs to allow the model to overfit the training set completely. Figure 4.5 shows the detection results of CLIPtheCenter on the dataset. As seen from the figure, the model was able to overfit the three images of the training set and provide accurate detections in the image. For classification purposes, algorithm 2 is used. Groundtruth bounding box crops are used to generate the support class codes and to calculate the cosine similarity metrics for classification. There were two main results from this experiment.

Algorithm 5 CLIPtheCenter - Data Flow

Input : Input Image I, GroundTruth Bounding Boxes B, number of object proposals k, stage S

Output : Bounding Box proposals parametrized as coordinate B and the corresponding embedding Ω

procedure CLIPTHECENTER(I, B, k, S)
  $f = F(I)$ ▷ Extract the Image Features
  h_C, h_B, h_R = $H_C(f)$, $H_B(f)$, $H_R(f)$ ▷ Get the output heatmap from the heads
  B_o = ArgMax(h_C, h_B, top$_k$ =k) ▷ Extract the top object proposals
  if S equals TRAIN **then**
    loss + = L2Loss(B,B_o) ▷ Calculate the L2 loss on Center and Bounding Box Heatmap
  end if
  B_o= ValidateDetections(B_o)
  for $P \ in \ B_o$ **do** ▷ Iterate through object proposals
    P = GenerateBinaryMasks(P) ▷ Create a binary mask for each object region
    $P = P * h_R$ ▷ Extract RoI heatmap values filtered through the binary mask
    $\Omega_i = F_{POOL}(P)$ ▷ Perform Forward Pass on the Masked RoI heatmap
    I_{CROP} = CropInputImage(I, P) ▷ Crop Input Image to the Proposed Region
    I_{CROP} = PreProcessImage(I_{CROP}) ▷ Preprocess the cropped Image for CLIP Encoder
    $\Omega_{CLIP} = F_{CLIP}(I_{CROP})$ ▷ Calculate CLIP Embedding for the proposed region
    if S equals TRAIN **then**
      loss + = CosineEmbeddingLoss(Ω_i, Ω_{CLIP})
      loss.backward()
    end if
    return B_o, Ω
  end for
end procedure

(a) Aeroplane

(b) Sheep

(c) Dog

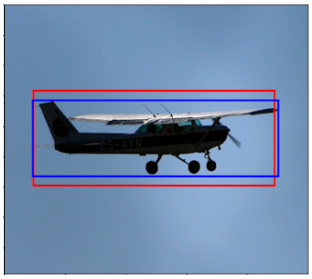
Figure 4.4: **Dataset used for Single Instance Object Detection.** The dataset has three images belonging to three different classes.

Table 4.1: Architecture Details of the CLIPtheCenter.

Module	Details
Feature Extractor Encoder	ResNet-50
Feature Extractor Decoder	U-Net
Image Input Dimension	320
Dimensions of Heatmap	320
Bounding Box Heatmap Head	3 blocks of [Conv2D, BatchNorm, ReLU] layers
Center Heatmap Head	1 block of [Conv2D, ReLU] layers
RoI Heatmap Head	3 blocks of [Conv2D, BatchNorm, ReLU] layers
Pooling Encoder	2 blocks of [Conv2D, BatchNorm, ReLU] layers followed by Linear Layer
Number of Proposals per Image	1 (for both Training and Inference)

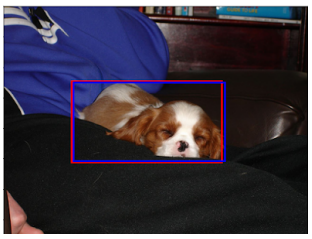
1. CLIPtheCenter was able to both detect the positions of the bounding boxes as well as classify them into the correct semantic categories with the Mean Average Precision of 100. This confirms the algorithmic sanity and practical feasibility of the proposed architecture.

2. In order to check for the effectiveness of the generated embeddings of the pooling encoder for few-shot classification, an additional experiment was devised. Instead of using the ground-truth bounding box to generate the support class codes, alternate instances of the object were used. These alternate images are shown in figure 4.6. Even with these alternate support codes, there was no drop in the Mean Average Precision Metric. This shows that the pooling encoder can learn the discriminative class codes from CLIP. This also again confirms the effectiveness of CLIP embedding in few-shot classification.



(a) Aeroplane

(b) Sheep



(c) Dog

Figure 4.5: **Results on the Single Instance Object Detection.** Blue Rectangles denote the ground-truth bounding box, and the Red rectangles are the predicted bounding box. The model could correctly detect the position of the object classes in all three images.

(a) Aeroplane

(b) Sheep

(c) Dog

Figure 4.6: **Alternate support images used for class assignments.** Class supports in polar contrast in the training object instances are used to check for the effectiveness of the pooling encoder of CLIPtheCenter.

(a) Input Image (b) Groundtruth Center Heatmap

Figure 4.7: **Data Sample for Multi-Instance Object Detection.** The heatmap has multiple peaks to account for multiple object instances of the aeroplane within an image.

4.2.4 Extending to Multi-Instance Object Detection

In the previous section, the algorithmic test was performed on images that contained only a single object instance. This test is devised as an extension of the previous test to check the algorithmic efficiency in cases where the image contains more than one object instance. The following changes were made to make the object compatible with multi-object detection.

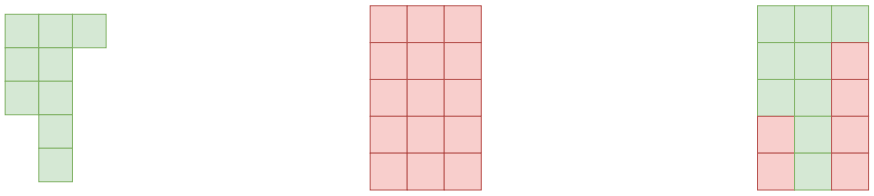
1. The ground-truth heatmap has multiple peaks to account for multiple instances. A sample ground-truth heatmap peak is shown in figure 4.7.

2. The number of object proposals extracted from the image was made to a hundred during inference. However, only the top five object proposals were used during training to speed up training and reduce the computation load.

3. For loss computation and backpropagation through the network, the losses are only considered to be the actual number of instances in the object, irrespective of the number of proposals generated by the model. This is illustrated in figure 4.8.

Dataset, Model and Metrics

A dataset of 30 images with ten samples from each dog, aeroplane, and sheep class is used to test multi-object detection performance. These images are again chosen from the PASCAL VOC 2012 dataset [13]. Figure 4.9 shows some sample images from the dataset. As in the last experiment, Mean Average Precision is used as the evaluation metric for estimating the detection performance. Again, the same model architecture from the single object detection experiments was used.



(a) Number of Groundtruth instances

(b) Number of proposals generated from the Model

(c) Instances used for loss calculation

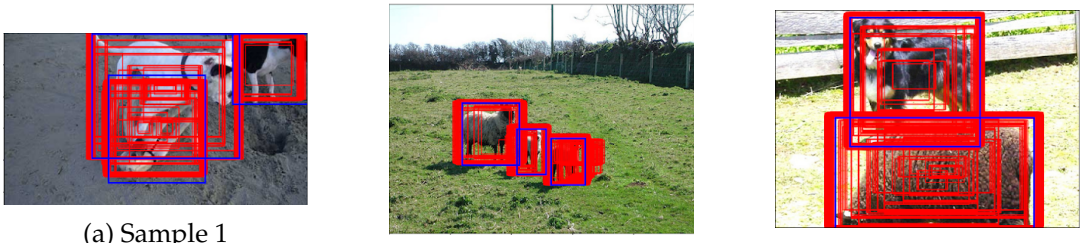
Figure 4.8: **Schematic of Loss Calculation for Multi-Object Detection.** Columns represent the images in a batch, and rows represent object instances in each image. Irrespective of the number of proposals generated per image in the architecture, the losses are calculated only for the actual number of object instances in each image in the batch.

(a) Sample 1

(b) Sample 2

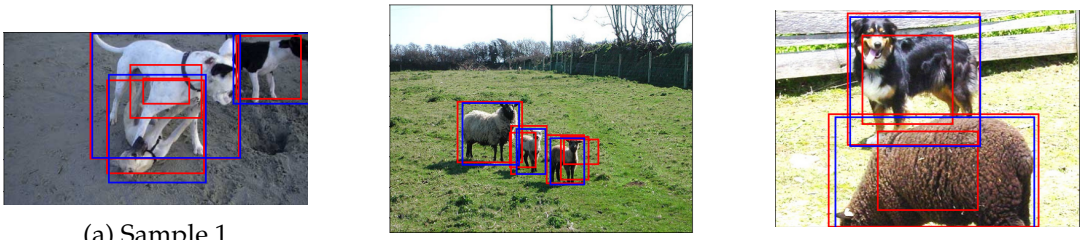
(c) Sample 3

Figure 4.9: **Dataset used for Multi-Instance Object Detection.** The dataset has 30 images belonging to three different classes.



(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 4.10: **Object Proposals generated from the model.** Hundred proposals were generated from the model for each of the images. Due to the clustering of peaks in the heatmap, multiple redundant bounding boxes are generated for each of the object instances.



(a) Sample 1 (b) Sample 2 (c) Sample 3

Figure 4.11: **Object Proposals after Non-Maximum Suppression.** Non-maximum suppression filters the redundant boxes by considering the objectness score and Intersection over Union between different proposals.

Results and Discussion

The model was trained again for 1000 epochs to overfit the training set, and the results were also evaluated on the same training split. An additional Non-Maximum Suppression (NMS) post-processing step is used to remove redundant bounding boxes and improve the detection's precision. This algorithm is explained in detail in the section 2.2.3. There were two significant results from the experiment.

1. With NMS postprocessing included, the model achieved 84.8 percent mean average precision on the training split based on pure detection performance. Some sample results before and after NMS postprocessing are shown in the figures 4.10 and 4.11 respectively. This confirms the proposed detection model's effectiveness in localizing multiple instances in an image.

2. Additionally the detections are also classified using the same support class codes generated from the ground-truth objects as in the previous single-instance object detection experiment. When also considering classification, the Mean Average Preci-

sion of the model was reduced to 38.6 percent. However, this drop in performance is expected. Since the number of images was increased to thirty, the Pooling encoder has to learn a more diverse visual representation for each of the three classes, which is quite hard to learn from just ten images a class. Ideally, the classification performance must improve by using a larger training split and a more powerful model to improve the learning capacity of the pooling encoder.

4.3 Generalization to Large Scale Datasets

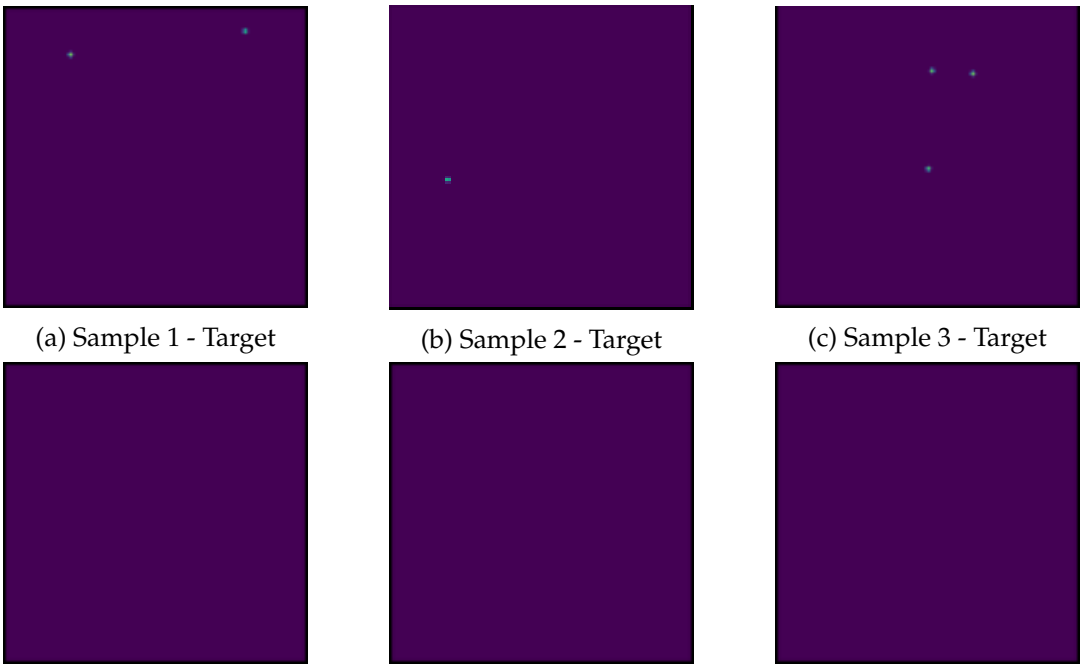
In the previous section, a novel few-shot object detection framework was built from scratch. The framework was also tested for algorithmic feasibility to make sure that the model has the potential to provide accurate object proposals. In addition, the ability of the pooling encoder embedding was verified to discriminate different classes in accordance with the few-shot paradigm. In the sections, experiments are designed to test the model's generalization ability. Therefore, the model is trained on a larger dataset and tested for the detection performance on an unseen test split. This is in contrast to the previous experiments where the model was overfitting on the training set, and the evaluations were also performed on the same seen training set.

4.3.1 Dataset and Metrics

A few-shot dataset was designed to test generalization ability based on the PASCAL VOC 2007 dataset [12]. In cognizance of other state-of-the-art works in few-shot object detection, a base to the novel class split of 15-5 is made. For each class in the dataset, the dataset contained at least a hundred instances of the classes in each of the splits. The dataset details are cataloged in the table 4.2.

Table 4.2: Details of the generalization dataset.

Parameter	Base Classes	Novel Classes
Number of Classes	15	5
Class Names	Bicycle, bird, boat, bus, car, cat, chair, diningtable, dog, motorbike, person, pottedplant, sheep, train, tvmonitor	Aeroplane, bottle, cow, horse, sofa
Training Set	1230 Images	462 Images
Test Set	1354 Images	494 Images



(a) Sample 1 - Target (b) Sample 2 - Target (c) Sample 3 - Target

(d) Sample 1 - Prediction (e) Sample 2 - Prediction (f) Sample 3 - Prediction

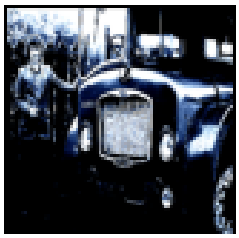
Figure 4.12: **Trivial solution convergence of the center heatmap.** The figure shows the ground-truth and predicted center heatmap for three sample images of the training set. The model reaches a local minimum where all the center heatmap being zeros still leads to low loss function values, thus leading to convergence.

4.3.2 Experiment 1 - Performance of CLIPtheCenter

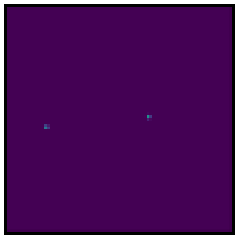
The same model architecture defined in the table 4.1 with the multi-object detection extensions was trained in the training set of the prepared dataset. However, even though the losses reduced during training, the model did not provide any observable performance on both the training and test set with nearly near zero Mean Average Precision. On closer evaluation, it was observed that the model had converged to a trivial solution of zero center heatmap. However, since the loss of bounding box values is only calculated at the peaks of the heatmap, it leads to incorrect learning of the bounding box values at the wrong pixel locations. This phenomenon is illustrated in the figures 4.12 and 4.13.

4.3.3 Troubleshooting and Evolving Towards CLIPtheCenter v2

After the initial experiment on training CLIPtheCenter on the generalization dataset, several deficiencies in the design of the proposed model and the foundational CenterNet model were illuminated in great detail. To address these deficiencies, several tweaks and

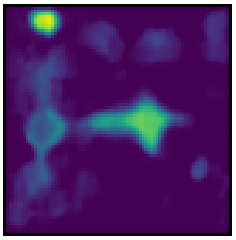


(a) Processed Input Image

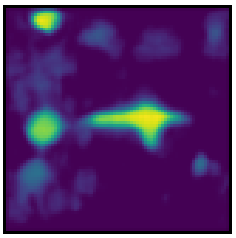


(b) Target Peak positions of the Heatmaps

(c) Predicted Center Heatmap



(d) Predicted Bounding Box Width Heatmap



(e) Predicted Bounding Box Height Heatmap

Figure 4.13: **Incorrect Learning of Bounding Box Values.** Due to incorrectly predicted heatmap during advanced training epochs, the bounding box values are learned at the wrong pixel positions. This is reflected in highly diffuse bounding box peaks leading to false detections.

fixes were tried out. A complete list of performed troubleshooting experiments is tabulated in the table 4.3. After a thorough troubleshooting and exploration phase, any other possibility of improving the model was ruled out within the project timelines. Additionally, it was observed that the detection modules, even without the embedding losses, did not provide quantifiable results to confirm the detection performance on a larger dataset.

A new version of CLIPtheCenter is proposed to circumvent this challenge, hereafter referred to as CLIPtheCenter v2. Instead of building the detection modules of CLIPtheCenter from scratch, the newer version uses the foundational CenterNet without any changes to the individual components. With the detection modules from the original CenterNet, the RoI head, the Pooling Encoder, and the auxiliary branches are re-implemented on this base architecture to add the few-shot capability to the CenterNet model. The details of this proposed architecture are tabulated in table 4.4 and illustrated in figure 4.14.

4.3.4 Experiment 2 - Class Agnostic Detection Performance

In this experiment, the pure detection performance of the models is evaluated on the training and test split of the generalization dataset. In addition, this experiment aims to assess the impact of the added few-shot classification as auxiliary branches on localization performance. Two experimental models were considered to evaluate the performance.

1. **CenterNet** - A vanilla Centernet is trained on training split with all the classes provided a zero class index to make the evaluation class agnostic. For fairness of the experiment, the same feature extractor and head layers are used as in table 4.4.

2. **CLIPtheCenter** - The proposed CLIPtheCenter v2 model is trained on the training split with the architecture described in table 4.4.

The models were trained only on the base classes, and no finetuning was performed on the novel classes. The performance on the dataset splits are tabulated in the table 4.5. Three main results are to be inferred.

1. Across both the base and novel classes, CLIPtheCenter performed better than CenterNet. This is a surprising finding since the proposed extension modules are motivated toward few-shot classification. However, as this is an experiment of class agnostic detection performance, these extensions are not directly evaluated for their performance. This gain in performance of CenterNet could be intuitively explained from the cross-interaction mechanisms between the RoI head and the detection modules, namely, the center, bounding box, and the offset heatmap heads. The RoI head and the pooling encoder proposed as extensions could be functionally viewed as a classification module. However, there is an essential difference in how the targets for this classification function are formulated. In most object detection networks, the class targets are integer codes in the range of a number of classes in the training set. In contrast, the target for the pooling encoders is the CLIP embedding, on which the

Table 4.3: Failure Analysis of CLIPtheCenter on generalization dataset.

Problem	Causes	Solutions Explored
The center heatmap is very sparse.	Only the pixels close to the object centers have non-zero values. It is rare for an image to contain more than five object instances, leading to many zero-valued pixel positions.	1. The focal loss was used to combat the sparsity of heatmap values. 2. The gaussian radius of the heatmap peaks was increased to reduce sparsity. 3. The L2 loss function is changed to an L1 loss function for a sparse solution space that is closer to the target
Learning of Bounding Box values depends on accurate center heatmap peaks.	To reduce the model learning load of three separate heatmaps, the CenterNet architecture proposes learning the width and height values only at the peaks of the center heatmaps.	1. Bounding box heatmap losses are trained once the center heatmap loss stabilizes to stagger the learning objective of the model. 2. The bounding box loss is calculated across the entirety of the heatmap instead of just the peaks to decouple with the performance of the center heatmap.
The solution does not generalize and is prone to underfitting the dataset.	With a larger generalization dataset, the feature extractors and pooling encoder may not have the learning capacity to learn from more diverse object instances.	1. Increased the number of layers of the pooling encoder to improve its learning capacity. 2. ResNet-18[19] was used in each of the heads pre-trained on the ImageNet dataset to extract better visual features.
The model converges to a local minimum instead of a global minimum that can generalize to the dataset.	By the design of loss function and sparsity of the targets, the trivial solution of zero heatmaps still leads to decreased loss	1. The learning rate was optimized using Optuna[1] to find more optimized gradient step sizes. 2. Different Optimisers such as Adam[23], Adagrad[11], and Adadelta[57] were tried out to improve model learning capacity.

Table 4.4: Architecture Details of the CLIPtheCenter v2.

Module	Details
Feature Extractor Encoder	ResNet-18
Feature Extractor Decoder	3 blocks of [ConvTranspose2d, ReLU] layers
Image Input Dimension	512
Dimensions of Heatmap	128
All Heatmap Head	1 blocks of [Conv2D, ReLU, Conv2D] layers
Pooling Encoder	ResNet 18 followed by Linear Layer
Number of Proposals per Image	25 for Training. 100 for Inference.

Table 4.5: **Class Agnostic Detection Performance on the generalization dataset.** CLIPtheCenter doubled the performance of the vanilla CenterNet on base classes. Significant performance increases are also observed in the novel classes, even though the models were only trained on the base classes.

Serial Number	Model	Dataset Split		
		Base Classes		Novel Classes
		Training Set	Test Set	Test Set
1	CenterNet	28.4	17.2	7.6
2	CLIPtheCenter	**62.3**	**34.3**	**10.1**

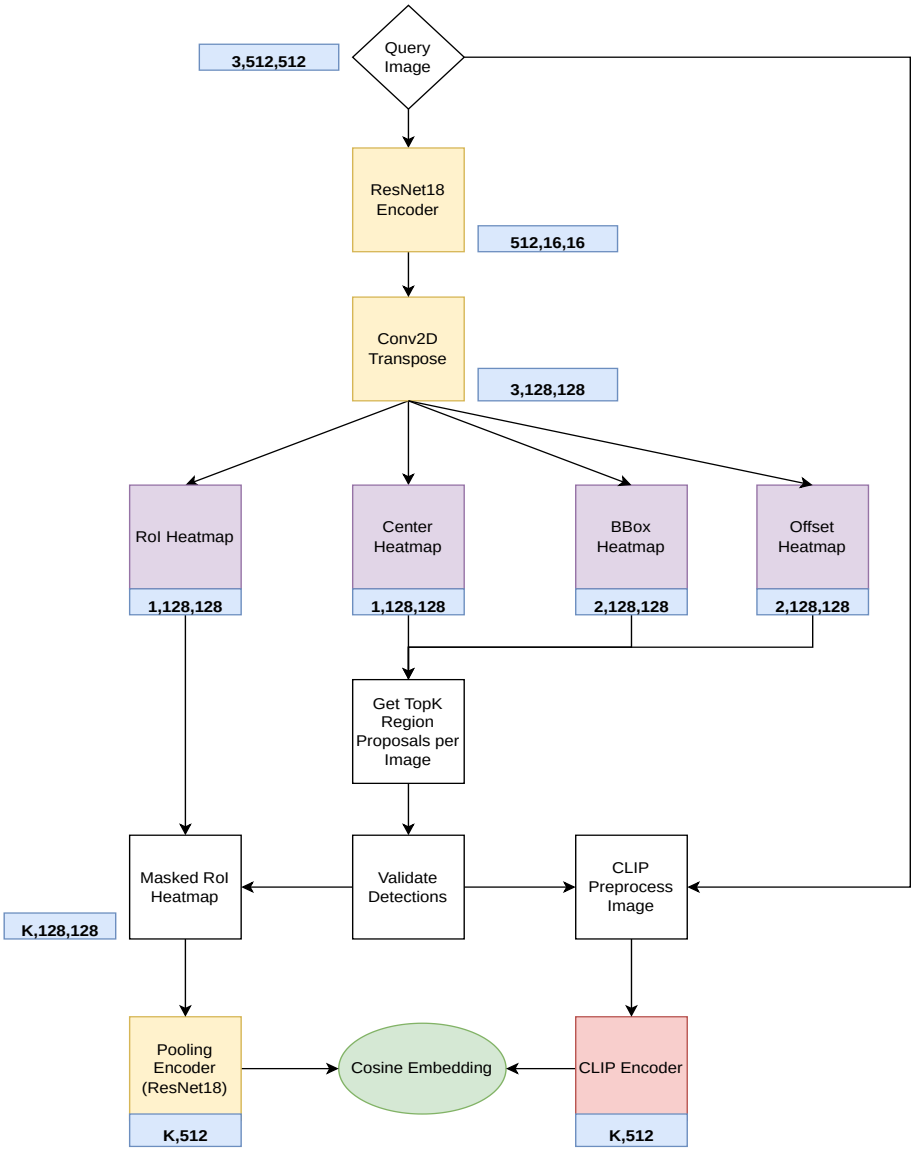


Figure 4.14: **Architecture of the proposed CLIPtheCenter v2.** The revised version uses the vanilla CenterNet as the detection module with extended few-shot capabilities through a RoI head, Pooling Encoder, and an auxiliary branch.

losses are backpropagated. From the previous experiments with CLIP, the descriptive, as well as the discriminative ability of the CLIP encoder is well demonstrated. Furthermore, by classifying each of the object proposals in the latent space of the CLIP encoding instead of a primitive integer space, the model has more useful information available to detect the presence of an object within a proposed region. The objectness learning potential of the center heatmap head has improved object information from the gradient propagation of the RoI head. Thus, the peaks of the Center heatmap are more accurate due to cross-interactions through the gradient sharing through the unifying decoder module.

2. Both the models have translating ability on the novel classes even though the model was not trained on any of the novel classes. Out of these, CLIPtheCenter has significant performance gains compared to the CenterNet model. This translation ability to the CenterNet comes from the priors that the feature extractor encoder and pooling encoder modules provide. Since both of these models were pre-trained on ImageNet and the novel classes are also a subset of the thousand classes of ImageNet, these modules may extract useful visual features for detecting these novel classes. The performance gain in CLIPtheCenter comes from the improved discriminative embeddings of the CLIP encoder that aids the detection modules to localize better.

4.3.5 Experiment 3 - Class Inclusive Detection Performance

In this experiment, the performance of the models was evaluated for the complete object detection performance, considering both the localization and the classification tasks. For experimentation, three frameworks were considered.

1. **CenterNet** - The vanilla CenterNet model was trained on the 15 base classes with class indexes. Once again, the same feature extractors as described in the table 4.4 were used for fairness of comparison.

2. **CLIPtheCenter - Pooling Encoder** - The CLIPtheCenter model was trained on the 15 base classes without providing class labels to the modules. The assignment of the classes was performed downstream by extracting the support class codes using algorithm 3. The classes of object proposal are assigned the algorithm 2.

3. **CLIPtheCenter - CLIP Encoder** - Similar to the previous detection framework, the same CLIPtheCenter model trained was used. However, for the assignment of classes, the embeddings of the CLIP encoder were used instead of the pooling encoder, similar to the few-shot classification experiments described in the section 3.3.2.

In all of these frameworks, the model was trained only on the training set of the base classes and evaluated for performance on both the base and novel classes. For CLIPtheCenter-based frameworks, the support class code was generated from 30 random images from the

Table 4.6: **Class Inclusive Detection Performance on the generalization dataset.** Using the detection proposals from CLIPtheCenter and the CLIP Encoder for class code-based classification, the performance is tripled compared to the vanilla CenterNet. However, the embedding from Pooling Encoder reports poor performance due to incomplete knowledge distillation.

Serial Number	Model	Dataset Split		
		Base Classes		Novel Classes
		Training Set	Test Set	Test Set
1	CenterNet	17.8	8.2	-
2	CLIPtheCenter - CLIP Encoder	**50.2**	**32.3**	**15.9**
3	CLIPtheCenter - Pooling Encoder	0.5	0.4	1.4

training set for each class. The results of this experiment are illustrated in figure 4.6. There are three main inferences to be derived from this experiment.

1. CLIPtheCenter, when used with CLIP encoder for class assignments, triples the performance on the base classes. This performance gain comes from two main factors. One is the improvement in localization performance due to the cross-interaction phenomenon between the RoI head and the detection heads. Two, the class assignment are also improved due to the CLIP embeddings' effectiveness in generating discriminative class codes and a more rigorous classification through class codes instead of pure integer class targets.

2. Additionally, CLIPtheCenter adds a few-shot capability to the CenterNet model with up to 15.9 percent Mean Average Precision. Due to architectural constraints of the CenterNet model, the set of classes that are used in training and test set must be identical. Therefore, performance metrics cannot be evaluated unless CenterNet is retrained on the novel classes. However, doing so will make different frameworks no longer comparable.

3. However, when using the pooling encoder for the assignment of the classes, the Mean Average precision across both the base and novel classes was reduced to near negligible values. This is due to non-discriminative embeddings generated by the pooling encoder. This is also reflected in the encoder training loss curves shown in figure 4.15 The gradient of the loss curve is relatively small, and it would require much longer training schedules for the error to reduce significantly. Unless the pooling encoder can reduce embedding loss by learning to output embeddings closer to CLIP, the classification results are expected to be not optimal.

   From a knowledge distillation viewpoint, the pooling encoder with a relatively meager learning capacity of a ResNet-18 model needs to reproduce the outputs of a much more powerful vision encoder-based CLIP encoder by training on just 1230

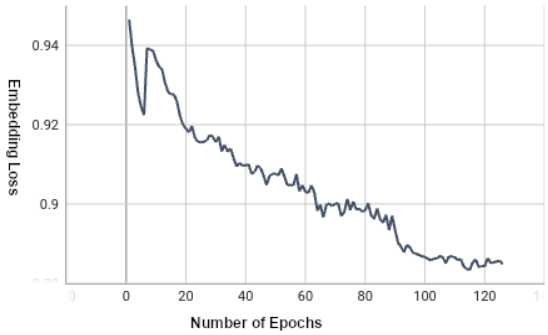


Figure 4.15: **Embedding Loss Curves in CLIPtheCenter Training.** The revised architecture uses vanilla CenterNet as the detection module and extends it for few shot object detection.

images of the training set. This pushes the distillation challenge to a much harder bound of transferring the knowledge of a powerful teacher to a relatively less powerful student, which is an allied research direction in itself.

.

4.3.6 Experiment 4 - Ablation Studies

The previous experiments demonstrated the few-shot performance of the proposed model across both the base and novel classes. This experiment is designed to dive deeper to understand the reasons for the performance gains. There are two possibilities of where the improved detection performance could come from. One, it could be due to the effectiveness of the CLIP embeddings used as the target of the Cosine Embedding loss to help the pooling encoder learn useful embeddings. Two, there is a possibility that the use of descriptive class code for scoring the object proposals makes it more useful for the model to find object instances instead of usual integer targets. To check which of the above two ideas boosts model performance, additional frameworks were devised in addition to the ones proposed in sections 4.3.4 and 4.3.5. This involved changing the CLIP encoder of the auxiliary branch to an ImageNet pre-trained ResNet-34 [19] encoder. This change is shown in the schematic figure 4.16.

All the frameworks are trained on the training set of the base classes and evaluated for their performance on both the base and novel classes without any finetuning. Figure 4.7 shows the performance of the detection frameworks trained without consideration of class indexes on the base classes split. Similarly, figure 4.8 considers the detection and classification performance of the frameworks on the base classes. Similarly to the CLIPtheCenter experiment 4.3.5 with the CLIP Encoder, the classification is performed with the embeddings of the pooling encoder and ResNet34 encoder for the ablation model described in the figure 4.16. There are two main results from the experiment.

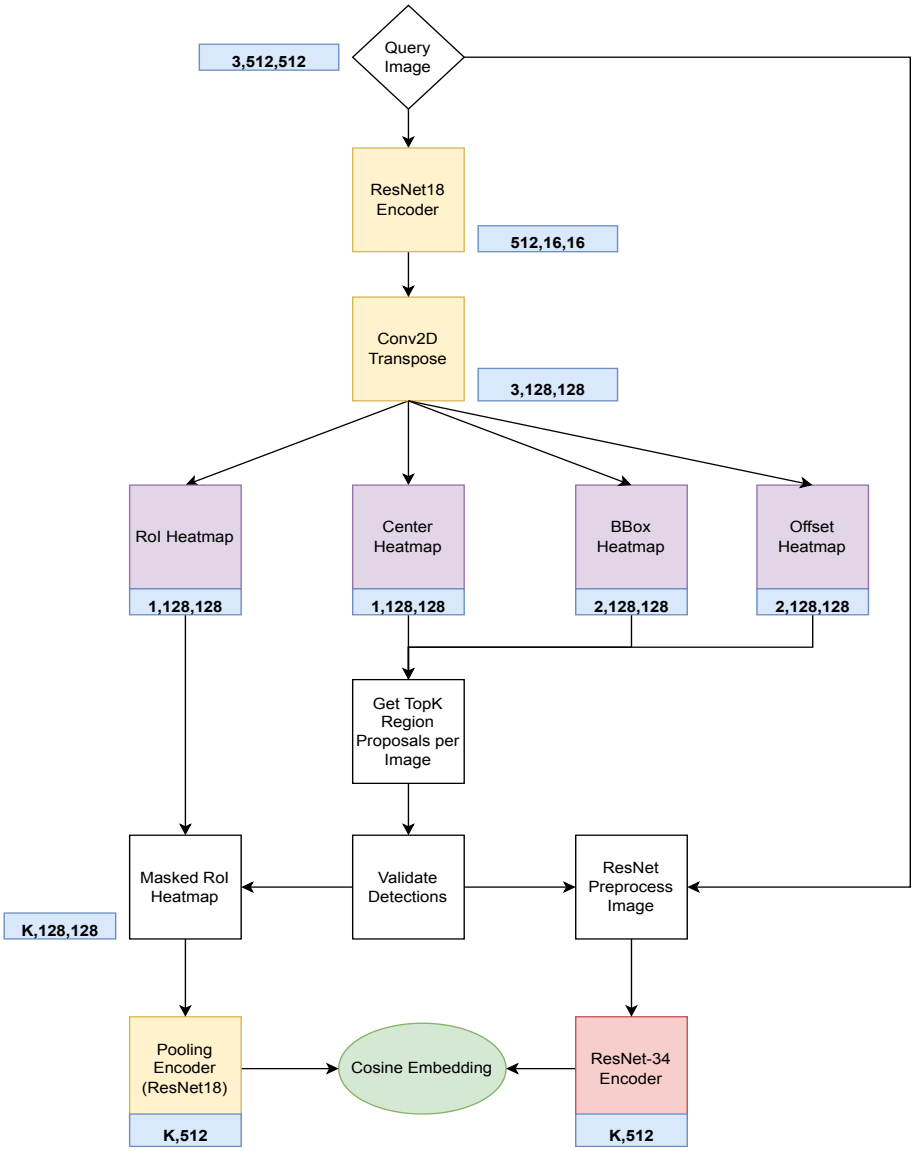


Figure 4.16: **Architecture of the ablation studies on CLIPtheCenter v2.** To study the underlying reasons for performance gains, the CLIP model is replaced by a ResNet-34 in the auxiliary branch to generate targets for embedding loss.

Table 4.7: **Ablation Study Results on Class Agnostic Detection Performance.** When the CLIP encoder is exchanged for a pre-trained ResNet 34 encoder, significant performance gains can still be observed on the base classes. Even though descriptive class encodings were formulated for classification purposes, pure detection performance improves due to potential cross-head interactions.

Serial Number	Model	Dataset Split	
		Base Classes	
		Training Set	Test Set
1	CenterNet	28.4	17.2
2	CLIPtheCenter - CLIP Distillation	**62.3**	**34.3**
3	CLIPtheCenter - ResNet-34 Distillation	57.3	32.8

Table 4.8: **Ablation Study Results on Class Inclusive Detection Performance.** Following the previous studies, when the encodings of ResNet-34 are used to classify the object proposals from CLIPtheCenter, the object detection performances take a huge hit. This shows the importance of class discriminative encodings, which the CLIP model has demonstrated throughout several experiments. While detection performance can be improved with descriptive feature encodings, classification performance relies on discriminative ability.

S.No	Model		Dataset Split	
			Base Classes	
			Training Set	Test Set
1	CenterNet		17.8	8.2
2	CLIPtheCenter - CLIP Distillation	CLIP Encoder	**50.2**	**32.3**
3		Pooling Encoder	0.5	0.4
4	CLIPtheCenter - ResNet34 Distillation	ResNet34 Encoder	0.5	0.5
5		Pooling Encoder	0.6	0.4

1. As seen from table 4.7, by using ResNet-34 as the teacher model instead of the CLIP model, there is still a significant improvement in the performance when compared to the vanilla CenterNet model. However, the model's performance based on the ResNet-34 encoder still lags behind the model that uses CLIP as the teacher. This provides a meaningful inference. For pure detection performance that aims at localization of the bounding box, it is only essential that the target class codes are descriptive in the visual feature space. Since both the ResNet-34 and CLIP are pre-trained on classes that are very similar to the base and novel classes of the generalization, there is not a huge drop in the performance when a different teacher model is used.
2. In the class inclusive detection performance as shown in the table 4.8, the performance of the model that uses ResNet-34 as the teacher shows an abysmal detection performance irrespective of Pooling encoder or the ResNet-34 encoder is used for classifying the object instances. This shows the vital requirement of the encoder's discriminative ability to separate the classes in the latent space. Even though descriptive embeddings were a sufficient condition for improving class agnostic detection performance, they are not sufficient to perform the downstream task of object proposal classification. This is because few-shot classification requires the discriminative ability of the embedding, which CLIP encoders have thoroughly demonstrated throughout the experiments.

4.4 Summary of Few-Shot Object Detection

This section of the thesis was motivated by the challenge of translating few-shot classification results to the task of object detection. To achieve this effect, a novel few-shot object detection, CLIPtheCenter, was built from the bottom up that uses CLIP encoding to classify the object proposals into different object classes. The model was thoroughly tested for algorithmic feasibility, and extensions were also proposed for multi-instance object detection. A custom dataset consisting of fifteen base classes and five novel classes was constructed from the PASCAL VOC [12] dataset. The ability of the CLIPtheCenter model to generalize results to a larger scale dataset was experimented on with this custom dataset.

From the experiments, several vital results were inferred. When only pure detection performance was evaluated, CLIPtheCenter performed almost twice as well on the vanilla CenterNet on the base classes. The model also demonstrated significant performance gains in the novel classes. The intuition behind these performance gains comes from cross-head interactions between the RoI head and detection heads. Classifying the object proposals into a more descriptive latent space instead of pure integers enhances the detection heads to discover the object regions more precisely, as the model is trained end to end. Additional ablation studies were performed to verify this intuition by exchanging the CLIP encoder with an ImageNet pre-trained ResNet encoder. When experimented with for both classification and detection performance, CLIPtheCenter almost tripled the performance on the vanilla CenterNet, when the CLIP encoder is used to assign the class through class codes.

At the same time, it provides up to 15 percent Mean Average Precision on the novel class without any fine-tuning on the novel classes. The experiments successfully demonstrated the effectiveness of CLIPtheCenter for few-shot object detection.

5 Conclusions

The thesis focussed on solving the task of few-shot object detection. Methods were proposed to translate the performance of a model trained in classes with abundant data into a set of classes with limited support instances. In order to tackle this challenge, the object detection task was decomposed into two different stages of classification and detection.

The first stage of the thesis aimed to solve a special case of few-shot classification. Given a query image and support images belonging to a set of target classes, the goal was to classify which class the query image belonged to among the target classes. A representative self-supervised method DINO [6] and a multi-modal model CLIP [38] were experimented with. These models with different learning paradigms were evaluated for their ability to provide discriminative class encoding. A classification algorithm was designed using the concepts of class code, and the performance of the encoders was evaluated on a domain-specific CUBS-200 [50] dataset. This dataset is specifically chosen to simulate real-world scenarios where the object classes look visually very similar. The most important results from the experiment were as follows:

1. Classification through CLIP encoded class codes can compete with a fully supervised ResNet-50 model without any finetuning. This demonstrates the discriminative ability of multi-modal embeddings for classification tasks.

2. With the increase in the number of support images for the generation of class codes better was the classification performance across both the DINO and CLIP encoders.

3. Transformer-based backbones are better feature extractors than ResNet encoders due to the self-attention mechanism that enables the model to learn the global context of the image.

The second stage of the thesis focused on translating the results from the few-shot classification into object detection. A few shot object detector was built from the ground up that used the CLIP embedding for classification instead of traditional integer-based class labels. To test the performance of this proposed model, CLIPtheCenter, experiments were carried out in various stages of development for single instance detection, extensions to multi-object detection, and a larger-scale generalization test. Finally, an ablation study was conducted by exchanging the CLIP as the target encoder for a less potent ImageNet pre-trained ResNet-34 model. The important revelations from the experiment were:

1. When evaluated for pure detection performance, CLIPtheCenter doubled the base classes' performance compared to vanilla CenterNet. A significant performance increase is also seen in novel classes.

2. Considering performance on classification and detections, CLIPtheCenter almost triples the performance on the vanilla CenterNet, when the pre-trained CLIP encoder is used for class code-based classification. Additionally, CLIPtheCenter extends the few-shot capability to CenterNet.

3. CLIPtheCenter provides up to fifteen percent Mean Average Precision on the novel classes, without any finetuning on the novel classes and having trained only on the base classes.

4. Use of descriptive class code such as CLIP encodings, instead of primitive integer codes, greatly enhanced detection performance. This result was verified and confirmed with the ablation study.

The performance of CLIPtheCenter shows a lot of promise with the test on the generalization dataset. However, these results raise an important and exciting question. Can similar gains in performance be observed on the larger benchmark datasets splits such as COCO [30] or Large Vocabulary Instance Segmentation (LVIS) [17] datasets that are designed with a long-tailed class distribution. Even if half of the observed performance gains are realized, the results would still be significant. However, there are two main constraints to carrying out large-scale training schedules.

1. **Lack of CPU and GPU optimized implementation.**

   During the training of CLIPtheCenter, a few CPU loop-intensive operations do not take advantage of multicore processors. For instance, when the masked RoI heatmaps are generated from the object proposal, the current implementation has $\mathcal{O}(BK)$ operations, B being the batch size and K, the number of object proposals per image. This makes scaling up the training more challenging due to the GPU VRAM constraints. Similarly, to generate the targets for training, the pooling encoder requires $\mathcal{O}(BK)$ forward passes on the CLIP encoder since the inputs are not batched to take advantage of GPU hardware parallelism. This can be fixed by using Numba [26] for vectorizing CPU-based operations and PyTorch batch operations solving GPU-based bottlenecks. With these two fixes, speed-ups in $\mathcal{O}(mK)$ are expected with $m \in [0.5, 1]$.

2. **Challenging of distilling CLIP into a smaller ResNet model.**

   The function of the pooling encoder is to learn to generate CLIP-like class discriminative embedding so that the outputs of it can be used for class code-based classification instead of the CLIP encoder. However, from the standpoint of knowledge distillation, this is a more challenging task with minimal data. The student model,

a ResNet-18 encoder, in the current implementation, must learn from a much more powerful vision transformer-based CLIP encoder by using only 1230 images. This pushes the theoretical possibility of the effectiveness of such imbalanced distillation to its limit. There are two ways to solve this issue. One is by using a more powerful encoder for a student model so that model capacities between the student and teacher are comparable. The second remedy is to train on a much larger dataset to enhance knowledge transfer. Unfortunately, both these are currently blocked by the previous constraint of ineffective code optimization to the training hardware architectures.

The above optimizations provide a lot of potential to improve the CLIPtheCenter architecture further. By carrying out these optimizations and performing a more extended training schedule on a larger benchmark dataset, the intuitions and the algorithms proposed can be verified with even more assurances. Furthermore, if the performance gains still translate to these benchmark datasets, it opens a new direction of rethinking object detection through the lens of class codes.

List of Figures

List of Tables

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *ECCV*, 2006.

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms — improving object detection with one line of code. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5562–5570, 2017.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *ArXiv*, abs/2005.14165, 2020.

[5] Nicolas Carion, Sergey Zagoruyko, and Francisco Massa. End-to-end object detection with transformers. https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/. [Online; accessed 18-November-2022].

[6] Mathilde Caron, Hugo Touvron, Ishan Misra, Herv'e J'egou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640, 2021.

[7] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. *ArXiv*, abs/1605.06409, 2016.

[8] Daniel de Souza Carvalho. Sobel edge detection filter - wolfram demonstrations project. https://demonstrations.wolfram.com/SobelEdgeDetectionFilter. [Online; accessed 16-November-2022].

[9] 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, 2009.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2021.

[11] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *J. Mach. Learn. Res.*, 2011.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[14] Qi Fan, Wei Zhuo, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4012–4021, 2020.

[15] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[16] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2022.

[17] Agrim Gupta, Piotr Dollár, and Ross B. Girshick. Lvis: A dataset for large vocabulary instance segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5351–5359, 2019.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2020.

[19] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[20] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[21] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8419–8428, 2019.

[22] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

[26] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: a llvm-based python jit compiler. In *LLVM '15*, 2015.

[27] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *International Journal of Computer Vision*, 128:642–656, 2019.

[28] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq R. Joty, Caiming Xiong, and Steven C. H. Hoi. Align before fuse: Vision and language representation learning with momentum distillation. In *Neural Information Processing Systems*, 2021.

[29] Zeming Li, Yilun Chen, Gang Yu, and Yangdong Deng. R-fcn++: Towards accurate region-based fully convolutional networks for object detection. In *AAAI Conference on Artificial Intelligence*, 2018.

[30] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[31] W. Liu, Dragomir Anguelov, D. Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021.

[33] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[34] Luke Melas-Kyriazi, C. Rupprecht, Iro Laina, and Andrea Vedaldi. Deep spectral methods: A surprisingly strong baseline for unsupervised semantic segmentation and localization. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8354–8365, 2022.

[35] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016.

[36] John R. Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda. *2008 IEEE Hot Chips 20 Symposium (HCS)*, pages 1–2, 2008.

[37] Juan-Manuel Pérez-Rúa, Xiatian Zhu, Timothy M. Hospedales, and Tao Xiang. Incremental few-shot object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13843–13852, 2020.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[39] Alec Radford, Ilya Sutskever, Jong Wook Kim, Gretchen Krueger, and Sandhini Agarwal. Clip: Connecting text and images. https://openai.com/blog/clip/. [Online; accessed 16-November-2022].

[40] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[41] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015.

[43] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to sift or surf. *2011 International Conference on Computer Vision*, pages 2564–2571, 2011.

[44] Deval Shah. Mean average precision (map) explained: Everything you need to know. https://www.v7labs.com/blog/mean-average-precision. [Online; accessed 19-November-2022].

[45] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *ArXiv*, abs/1905.11946, 2019.

[46] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[47] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 548–558, 2021.

[48] Xin Wang, Thomas E. Huang, Trevor Darrell, Joseph Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *ArXiv*, abs/2003.06957, 2020.

[49] Yangtao Wang, Xiaoke Shen, Yuan Yuan, Yuming Du, Maomao Li, Shell Xu Hu, James L. Crowley, and Dominique Vaufreydaz. Tokencut: Segmenting objects in images and videos with self-supervised transformer and normalized cut. *ArXiv*, abs/2209.00383, 2022.

[50] Peter Welinder, Steve Branson, Catherine Wah, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[51] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8372–8381, 2021.

[52] Xiaopeng Yan, Ziliang Chen, Anni Xu, Xiaoxi Wang, Xiaodan Liang, and Liang Lin. Meta r-cnn: Towards general solver for instance-level low-shot learning. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9576–9585, 2019.

[53] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *ArXiv*, abs/2107.00641, 2021.

[54] Jinyu Yang, Jiali Duan, S. Tran, Yi Xu, Sampath Chanda, Liqun Chen, Belinda Zeng, Trishul M. Chilimbi, and Junzhou Huang. Vision-language pre-training with triple contrastive learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15650–15659, 2022.

[55] Zhuyu Yao, Jiangbo Ai, Boxun Li, and Chi Zhang. Efficient detr: Improving end-to-end object detector with dense prior. *ArXiv*, abs/2104.01318, 2021.

[56] Fisher Yu, Dequan Wang, and Trevor Darrell. Deep layer aggregation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 2018.

[57] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[58] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, Shijian Lu, and Eric Xing. Meta-detr: Image-level few-shot detection with inter-class correlation exploitation. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2022.

[59] Yiwu Zhong, Jianwei Yang, Pengchuan Zhang, Chengkun Li, Noel C. F. Codella, Liunian Harold Li, Luowei Zhou, Xiyang Dai, Lu Yuan, Yin Li, and Jianfeng Gao. Regionclip: Region-based language-image pretraining. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16772–16782, 2022.

[60] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *ArXiv*, abs/1904.07850, 2019.

[61] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *ArXiv*, abs/2010.04159, 2021.