Stacked Denoising and Stacked Convolutional Autoencoders

An Evaluation of Transformation Robustness for Spatial Data Representations

Ulrich Schmid, Johannes Günther, Klaus Diepold

Technical Report

Stacked Denoising and Stacked Convolutional Autoencoders

An Evaluation of Transformation Robustness for Spatial Data Representations

Ulrich Schmid, Johannes Günther, Klaus Diepold

September 9, 2017

Institute for Data Processing
Technische Universität München

Ulrich Schmid, Johannes Günther, Klaus Diepold. *Stacked Denoising and Stacked Convolutional Autoencoders. An Evaluation of Transformation Robustness for Spatial Data Representations.* Technical Report, Technische Universität München, Munich, Germany, 2017.

Supervised by Prof. Dr.-Ing. K. Diepold ; submitted on September 9, 2017 to the Department of Electrical Engineering and Information Technology of the Technische Universität München.

Institute for Data Processing, Technische Universität München, 80290 München, Germany, `http://www.ldv.ei.tum.de`.

Abstract

Making sense of visual data in raw pixel representation is a difficult task in machine learning because of the complex ways underlying factors interact to create images. Applying two methods for representation learning, the stacked convolutional autoencoder (SCA) and the stacked denoising autoencoder (SDA), however, image data can be encoded to new representations.

After training both methods on digit (MNIST) and data from laser welding videos, the image features produced by SCA improve on their SDA counterparts in two desirable properties. Firstly, SCA yield image features that are more useful for a specific purpose, namely the image classification with a subsequent SVM classifier. Secondly, SCA features exhibit a higher degree of invariance to input transformations than those representations generated by an SDA.

The superior performance of SCA over SDA image representations is likely a consequence of SCA preserving and exploiting the spatial structure of the input data, while also coming with pooling layers which by design lead to a degree of invariance.

Acknowledgment

I would like to use this opportunity to thank my supervisors, Johannes Günther and Sunil R. Tatavarty, both for their sheer endless patience and their much needed support in theoretical and implementation concerns.

Thanks are also due to my fellow students, specifically Bernd Prößner, Christoph Passenberg and Stefan Röhrl, for all the stimulating discussions over 242 cups of coffee and their brainstorming abilities, rescuing me out of countless dead ends.

Contents

1 Introduction

Understanding visual data is a difficult task in machine learning or, to be more exact, in computer vision. Real world environments come with a large number of factors such as light sources, object shapes and object surface properties that interact within a complex causal web to produce image data in its raw pixel representation [1]. Humans are capable of processing images of high resolution in rapid sequencial frequency, extracting very abstract information from their surroundings. Current machine learning methods, on the other hand, require vast learning capacity and prior knowledge [2] in order to achieve remotely comparable successes on raw image data.

The stacked convolutional autoencoder (SCA), introduced by Masci et al. [3], is a representation learning method that can unsupervisedly encode data into a new representation while exploiting its spatial relations. Comparing SCA with the widely known stacked denoising autoencoder (SDA) method, this work aims to evaluate the usefulness of their new image data representations.

After both autoencoders have compressed image data into small sets of features, the features are examined with respect to information loss, interpretability for image classification by a support vector machine as well as robustness to input data transformations.

1.1 Topic Relevance

Compared to other sensory information, image data carries a great amount of information about the world – not without reason have, throughout evolution, many different organisms developed photoreceptive systems to detect and interpret photons meeting their paths [4]. Visual data can be used to infer geometrical and location information on surrounding objects, the materials they consist of and, given video data, their interactions with one another, such as relative velocity. For manmade machines such kinds of capabilities could be of “extremely high value” [5]. Obvious use cases in the field of robotics spring to mind, but because of the enormous amount of information deducable from the input, sheer limitless applications are imaginable – current examples being optical character recognition, video surveillance, autonomous driving or product quality assurance [6]. The quite general field of computer vision includes a multitude of subareas, all of which come with an array of individual solutions to their respective problems. While very specific recognition tasks such as optical character recognition can be solved with relative ease, the "most challenging version of recognition is general category (or class) recognition" [6].

1.2 Related Work

The most prominent current advances in area of image classification have been documented in the results on the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC), which "has become the standard benchmark for large-scale object recognition" [7]. Recent years have shown continuous improvements, with the 2014 leading algorithm's classification error of 6.8% being the first one coming close to but not yet surpassing what one can expect of a human annotator – 5.1% [7].

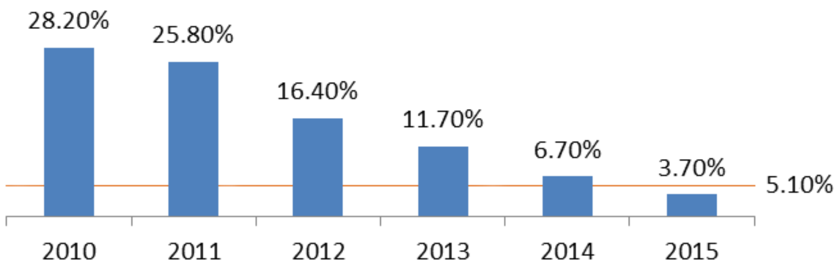


Figure 1.1: Steadily decreasing image classification error in the yearly ILSVRC challenge, surpassing human performance of 5.10% in 2015.

The successes of those recent ILSVRC advances have been made possible by the winning submission of 2012, Krizhevsky et al. [2], which introduced an innovation that "by 2014 all of the top contestants were relying heavily on" [2]: large scale convolutional neural networks (CNNs). Originally inspired by the human visual system, CNNs have today come into widespread use for almost all image processing and object recognition tasks. By applying identical filter patches throughout the whole input and subsampling of the so generated signals they achieve translation invariance which is particularly beneficial to imaging applications. Additionally, they are known as being able to be trained on many stacked layers, ultimately being able to make very high level interpretations of the raw low level data presented to them. The latest state of the art and 2015 ILSVRC image classification winner introduced a way to optimize enormous 152 layer networks to achieve its record breaking 3.57% test error [8].

Prior to 2012, however, first rate image processing solutions frequently employed a two step approach [9]. At first every raw image, which usually is quite high in its resolution or dimensionality, is reduced to a set of features – with significantly less dimensions. Having turned all images into sets of features, in a second step a classifier such as a Support Vector Machine (SVM) can be trained to link specific feature combinations to hypotheses on the original image, like an image class. Figure 2 exemplifies – in simplified manner – how these steps could work. The feature extraction method identifies features such as "stem", "leave" or "blossom" within an original image. A followup computer algorithm is then able to process this feature vector, for instance to determine that the image class must have been of a flower.

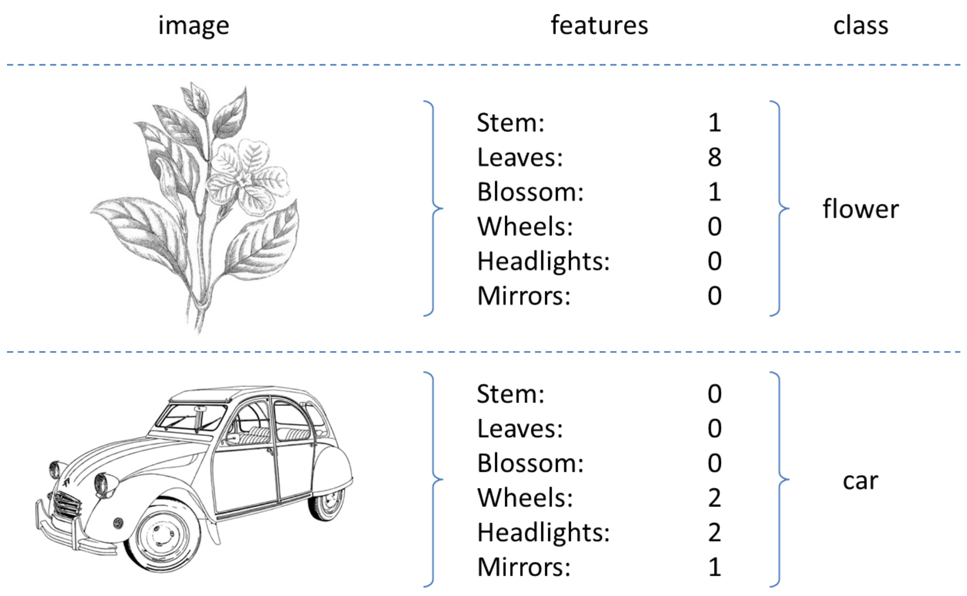


Figure 1.2: Raw images are reduced to sets of features, which can then be used to choose a prediction for their class label.

The transformation of the data into another representation is necessary since traditional classifiers cannot handle the large amount of information coming with raw, high-dimensional image data. They could, for instance, be prone to overfitting on irrelevant aspects of the data or simply require too high amounts of computational power to train models on the vast amounts of data. Feature extraction techniques have been developed for many kinds of highdimensional data, such as music, visual, and text [10, 11, 12]. They are usually designed in supervised manner by hand, and aim to obtain a specific piece of information from the raw data with every feature.

To advance the state of the art towards more intelligent and autonomous artificial systems, however, it might be beneficial to develop systems that can transform data into other representations unsupervisedly, reducing the amount of human input and thus allowing to scale systems to large data volumes.

An example for such an algorithm is the principal component analysis (PCA), which is capable of transforming images into vectors of linearly independent features through factorization [11]. The design of the principal components is only led by the data itself and does not require human input for image labeling - the algorithm merely stives to pack as much information as possible into a given number of principal components, condition to them being orthogonal to one another. While PCA have been frequently used for feature extraction purposes, their capabilities are limited as they are not able to perform nonlinear transformations. Desirable abstract representations of image data are, however, "generally highly non-linear functions of the raw input" [1].

Another unsupervised approach to feature extraction remedies this limitation - autoencoders (AE) can be seen as a "nonlinear generalization of PCA" [13]. Similarly to PCA, their new feature representation is not guided by human design, but by an error function that aims to maximize the recreatability of the original data from the given number of features. This is achieved by first applying an encoding and then a decoding neural network and comparing the original with the recreated data. Applying multiple of these units leads to stacked autoencoders (SAE).

Masci et al. [3] combine CNNs with SAEs to construct stacked convolutional autoencoders (SCA) for feature extraction. The approach promises to include CNN characteristics, which currently significantly outperform all other solutions in the realm of image processing. Additionally, however, SCA can be trained in unsupervised fashion and then be used for either extracting features or initializing the parameters of a traditional CNN.

1.3 Scope

While SCE have been introduced and applied to the standard datasets MNIST and CIFAR10 [3], they have primarily been examined as a means to initialize traditional CNNs, not as a tool for unsupervised feature extraction. By comparing features learnt by a SDA with those of a SCA, it should be investigated whether the latter, novel feature extraction method may be preferable for certain applications, particularly in the domain of image data. Considering the two different architectures, one could expect SCA to come with benefits such as more tolerance of variation in the input data (e.g. translational, rotational and scale variance). The methods at hand are to be examined with two quite different datasets. On the one hand, the MNIST digit recognition dataset provides well curated images that facilitate standardized comparison with other research approaches. On the other hand, a dataset of labeled high frequency image data of laser welding processes is to be analyzed and will allow for insights into the requirements of a real world application.

2 State of the Art

The following chapter provides an overview over the central concepts and technologies the subsequent sections build on. It is intended to introduce and solidify a basic understanding of the terminoloy and theoretical background.

2.1 Neural Networks

Determined to "understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking" [14], psychologists have examined the physiological setup of biological information processing in the brain. Investigating the inner workings of the neural cell networks that have been found, F. Rosenblatt introduced the Perceptron, "a probabilistic model for information storage and organization in the brain" [14].

A simple perceptron is a neural node that has a series of input and output connections. It stores a weight value w for every single of its i incoming connections as well as one bias value b. Making use of those parameters, it can compute a single output value

$$y = \varphi \left[\sum_{j=0}^{i} (w_j * x_j) + b \right] \tag{2.1}$$

for any given input vector x of size i. Specifically, the perceptron computes its output value y by summing up over all weighted inputs, adding the bias and then applying a so called activation function φ. The output value y is then passed on to all output channels, where it may become the input x_j for another perceptron. [15].

2.1.1 MLP

While a single basic Perceptron is only capable of learning linearly seperable patterns [15], combining multiple ones into bigger networks allows for the approximation of more complex relationships. The most prominent example of these larger networks is the multi layer perceptron (MLP), which comes with a set of input neurons, a set of output neurons, and a range of so-called hidden neurons inbetween. In this feedforward model every neuron is receiving inputs from all neurons of the preceding layer, performing its computation and sending the result on towards all neurons of the following layer.

Figure 2.1 shows an exemplary MLP schematic. Data can be fed to the red input neurons, is then being processed by the blue hidden neurons, and finally passed on to the green output nodes. While the number of neurons and hidden neuron layers can be

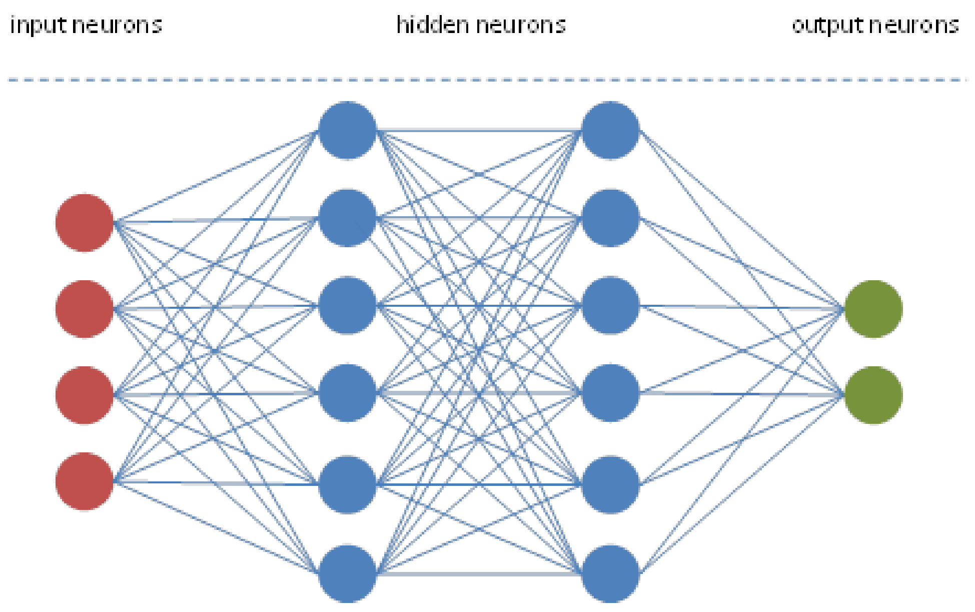


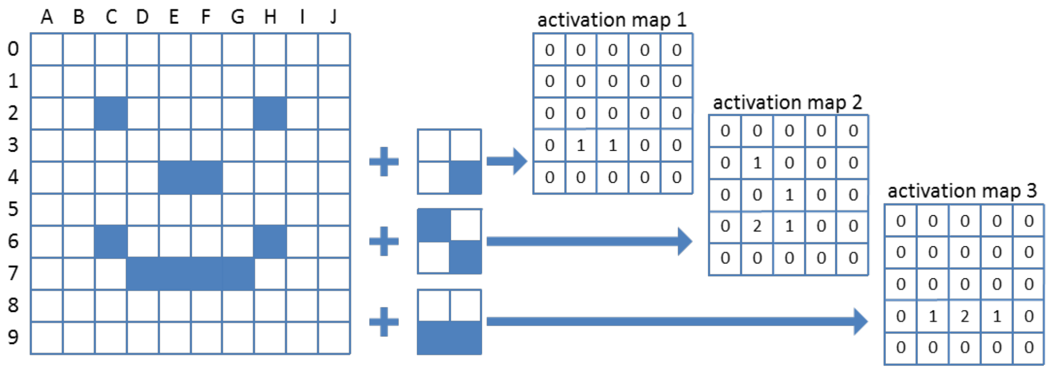
Figure 2.1: Schematic of a feedforward MLP neural network.

chosen freely when the network is set up, the direction of the dataflow will always be unidirectional, from input neurons to output neurons. In their theoretical analyses Hornik [16] and Cybenco [17] showed that if MLP networks are constructed with enough hidden units, they are capable of “approximating any Borel measurable function” [16], which has since fanned significant interest in artificial neural network research. Particularly concerning modern adaptation and extensions of the basic MLP architecture, however, “the theory of deep networks still poses many questions” [18].

2.1.2 Convolutional Networks

Convolutional Neural Networks (CNN) are specific type of neural network inspired by receptive fields found in the animal visual cortex [19, 11]. In the biological visual nervous system, neuron layers are not fully connected with all neurons in the preceding layer like in an MLP, but each focus on specific regions of the visual field [20, 11]. Reducing the number of connections inbetween neuron layers does also provide benefits to an artificial network’s training and has led machine learning to surpass human capabilities in some areas of image processing [7]. CNNs achieve this reduced connectivity with two core building blocks – convolutional and pooling layers. In practice, current architectures usually alternate between both layertypes and then conclude in a set of standard fully connected MLP layers.

Every convolutional layer comes with a set of filters, usually matrices with the same number of dimensions as the input but in some dimensions of smaller magnitude. An example could be an input image of 10 by 10 pixels that is convolved with three filters of 2 by 2. Figure 2.2 shows how a signal is processed by such a convolutional layer. Every filter moves through the whole original black and white image with a stride of 2 pixels and computes a convolution value at those positions. Putting these values together, every filter creates one activation or feature map. If the filter's stride length is greater than 1, every resulting activation map's size will usually be smaller than the input image's as depicted in 2.2. Some convolutional layer architectures may, however, decrease stride length to 1 or apply zero padding around the input image before performing the convolution to prevent the activation map dimensions from shrinking or, for very rare applications, even increase them.



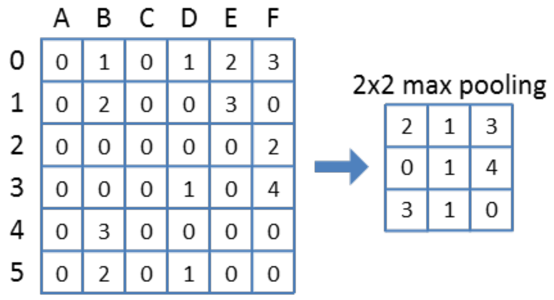
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	1	1	0	0
0	0	0	0	0

0	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	2	1	0	0
0	0	0	0	0

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	1	2	1	0
0	0	0	0	0

Figure 2.2: A 10x10 image is being processed by three 2x2 convolutional filters at a stride of 2.

Every convolution step essentially does a comparison between a subsection of the original image and a convolutional filter by multiplying pixels at the same location and summing up over those values. In effect, the activation maps are therefore pointing at the locations that resemble their feature map pattern within the original image.



	A	B	C	D	E	F
0	0	1	0	1	2	3
1	0	2	0	0	3	0
2	0	0	0	0	0	2
3	0	0	0	1	0	4
4	0	3	0	0	0	0
5	0	2	0	1	0	0

2	1	3
0	1	4
3	1	0

Figure 2.3: A 6x6 image is processed by 2x2 max pooling at a stride 2.

The second type of specialized layer, the pooling or subsampling layer, is depicted in figure 2.3. Pooling over a series of values can be done by aggregation methods such as summing up or averaging, but in practice “max pooling”, which means taking only the maximum of a a set of values, finds the most application. Including pooling layers has several benefits to the whole artificial neural network.

First and foremost, convolutional layers often raise the input’s size by substantially expanding the number of filter maps while not to an equal extent decreasing the image dimensions through stride. As this would make the network more computationally intensive and prone to overfitting, having pooling conveniently allows to greatly reduce it again and keep it manageable. Secondly, particularly max pooling introduces a degree of invariance towards small input shifts – it does not matter whether a singal spike moves within a pooling window, since it will always result in the same value after aggregation.

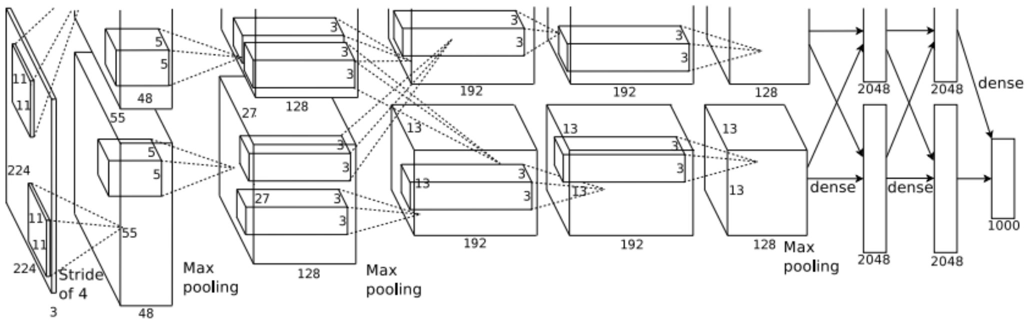


Figure 2.4: Schematic of the "AlexNet" architecture introduced by Krizhevsky et al. [2].

The breakthrough architecture for convolutional neural networks, bringing CNNs into the spotlight of the computer vision, is depicted in Figure 2.4. Krizhevsky et al. [2] achieved an impressive image classification performance by combining five convolutional, three max pooling and three fully connected dense layers. One can plainly see the data cubes increase in width (48 - 128 - 192 - 192 - 128) in most steps, which is due to the convolution layers’ increasing number of convolutional filters. On the other hand, the image width and height is greatly reduced (224*224 - 55*55 - 27*27 - 13*13), which is due to big strides in the max pooling and first convolutional layer. With a total of 650,000 neurons and 60 million parameters [2], however, the innovation was made possible not only by employing recent discoveries in CNN design and training methods, but also through advances in computational power and efficient algorithm implementations.

Following the "AlexNet" success, a multitude of advances has recently been made in all these areas, network architecture design, efficient training algorithms and the speedup of the training hardware itself. But while some of the recent state of the art solutions appear radically different [18, 8], they still rest upon the same basic building blocks - convolutional, pooling and fully connected layers. In essence, all CNNs are a adaptation of standard MLPs for image applications, but are advantageous since they come with "much fewer connections and parameters, are easier to train, while their theoretically-best performance is likely to be only slightly worse" [2].

2.1.3 Autoencoders

Autoencoding neural networks, or autoencoders, are in their base components quite similar to the MLP and CNN architectures discussed in the previous sections. Their main differing characteristic, however, lies with their set of output nodes which typically is of the exact same dimensionality as the input nodes. As a result, the output data can quite easily be compared to the input data, providing a training goal that does not rely on labels manually prepared by humans but only on the training data itself. This allows for unsupervised optimization of the full neural network, which in contrast to the usual supervised methods does not require additional input, enabling the exploration of unlabeled data [13].

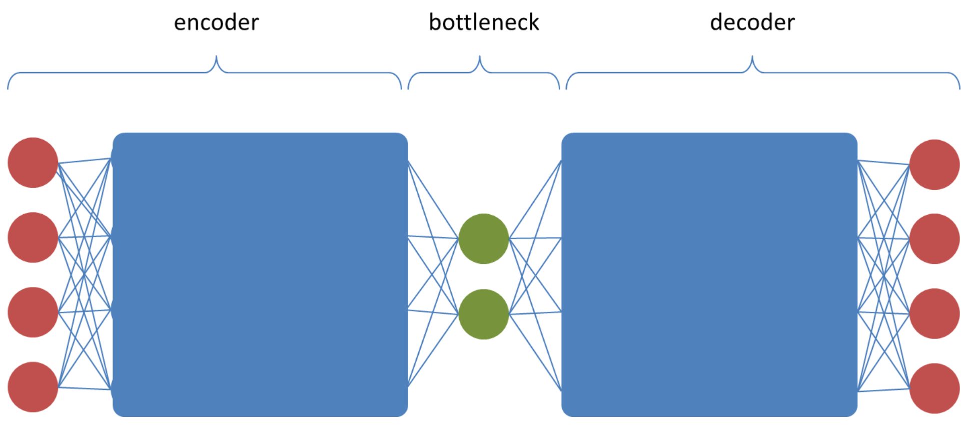


Figure 2.5: Schematic of an autoencoding architecture. While both input and output nodes (red) are of the same dimensionality, an encoder applies some type of neural network to reduce the input to a different internal bottleneck dimensionality (green). A decoder then reverts this process to arrive back at the original input shape.

While the output usually is of the same dimensionality as the input, an autoencoder's hidden layers are not. As seen in figure 2.5, the whole architecture can be divided into two subnetworks - an encoder that maps the input nodes to a new representation called the bottleneck, and a decoder which maps the bottleneck layer output back to the original dimensionality. While both parts are required to train the network, the whole decoding part is later dropped and only the encoder is used for transforming input data into a new representation. While MLP-style encoding and decoding networks are very common [21, 13, 22], they are not a necessity [1] and a combination of pooling and convolutional neural network elements could potentially be used in an autoencoder.

Summing up, autoencoders are a type of neural network that does not train for the prediction of a specific class label, but attempts to simply maintain all information that had originally been represented in the input data. They are constructed in such a fashion that their decoding part can be dropped after training, resulting in only an encoding neural network that transfoms data from its original into a new representation of the bottleneck's dimensionality.

2.1.4 Neural Network Training

As the neural network architectures discussed in the previous chapters almost always come with large numbers of bias and weight variables for neurons and their connections, tools are required to find suitable values for them.
Most state of the art approaches first apply a random initialization scheme to find appropriate initial values for all parameters. A popular strategy, introduced by Glorot and Bengio [23], samples them from uniform distributions whose bounds are dependent on current and preceeding neural network layer sizes.
Following the initialization, the values have to be fine-tuned so the neural network will produce not random but desired output. To achieve this, scholars typically use error backpropagation in combination with a form of gradient descent, which is an iterative optimization algorithm that utilizes the differentiation of the whole neural network's function with respect to those parameters that have to be tweaked [24, 25]. The output nodes of networks usually represent some definitive piece of information, such as one specific class in the case of most supervisedly trained neural networks - or the input data reconstruction for the case of unsupervisedly trained autoencoders. Backpropagation essentially nudges the weight and bias values into a specifc desired direction, driving the network's current outputs closer to their ideal - the true label or the true input data.

2.2 SVM

Support Vector Machines (SVMs) are a type of supervisedly trained classifier invented by Vladimir N. Vapnik in 1963 and introduced in its current form in 1992 and 1995 [26, 27]. SVMs classify labeled data vectors by first mapping the data into a highdimensional space, in which it is possible to then reasonably well seperate the data points into their classes with linear decision hyperplanes. While the optimization algorithms do strive to find a mapping into a space where all data is linearly seperable, so called soft margin hyperplanes have been introduced for cases "where the training data cannot be separated without error" [27], which also benefits classifier generalizability. Although SVMs in their basic form can only be used to classify data into one of two classes, multi-label problems can also be dealt with [28], for instance by training a series of SVMs - one for every class.

SVMs require the user to specify a series of hyperparameters. This includes the soft margin constant C, which sets the degree of penalty for data points violating the decision hyperplane. Large values of C will therefore force the SVM to find more closely fitting models. Another hyperparameter consists in the choice of kernel that maps the original data representation in its new space - popular alternatives include linear, polynomial and rbf kernels. Some of these kernels come with additional hyperparameters which have to be taken into consideration. The standard method to identify suitable SVM hyperparameters is grid search, in which several parameter options "are generally chosen on a logarithmic scale and classifier accuracy is estimated for each point on the grid" [29].

Presuming the adherence to the aforementioned advice, further improvements in SVM classification accuracy can primarily be obtained by improving the data or feature representation - one must find "features that make the classification task easy" [29].

3 Approach

Having established and defined the basic concepts, this section outlines the course of action for the experiments aimed at answering the theses.

3.1 Experimental Plan Overview

While the benchmark MNIST dataset consists of processed images in a standardized train- and test split, the welding data comes with several folders of raw image and label files. After thorough examination, initial focus therefore lies in preparing and structuring the welding data.

In a next step, several architectures are trained and tested for both SDA and SCA. Training hyperparameters as well as the architectures are adapted and optimized on the welding dataset, spending 14 working days on each autoencoder. Concluding this parameter search, the final autoencoding models are trained using the same computer setup and spending an equal time of about 50 hours on each model.

Finally, the quality of both autoencoders' features is assessed on both datasets. The trained SDA and SCA encoders are used to extract features from input images, which can then be evaluated in respect to their recreation error, predictive power and resiliance to input variation.

3.2 Dataset

3.2.1 Data Collection and Preprocessing

The benchmark MNIST dataset is comprised of 70,000 handwritten digits labeled into ten classes from zero to nine and split into a training set of 60,000 and a test set of 10,000 images. It has been created with great care, for instance by making sure the two sets of human writers for training and test images are disjoint [30]. Additionally, some preprocessing has been conducted as the digits "have been size-normalized and centered in a fixed-size image" [30].
As the welding data is not by any means an as well-curated and throroughly tested dataset as MNIST, its data and the data collection process should be examined as well as some basic preprocessing performed.
The welding data has been generated using several experimental setups (processes). Of these processes, four are of adequate quality for further experimentation, namely P76,

P77, P80 and P81. Every process itself comes with a range of videos (welds), each stemming from a single workpiece as it undergoes the laser welding procedure.

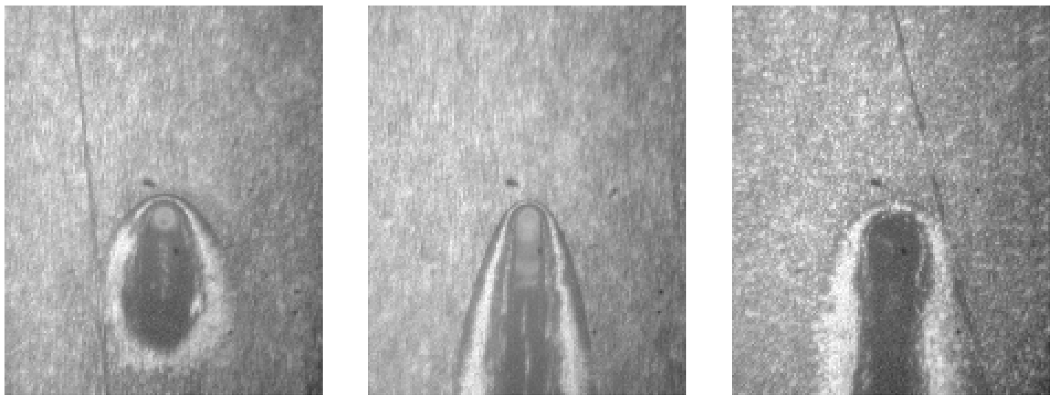

(a) Frame 0 - Beginning **(b)** Frame 879 - Continuing **(c)** Frame 1758 - Finishing

Figure 3.1: Raw welding images from several frames within the first weld video of process 76.

The greyscale weld videos are taken at a resolution of 144x176 and frequency of 1000 Hz or 1000 frames per second, and come with about 1,500 to 2,500 images each - three exemplary images can be seen in figure 3.1.
Over the course of every process, several scenarios have been simulated to encourage welding results of differing qualities. Such scenarios include, for instance, oil films or stains on the metal workpieces as well as variations in laser speed and power. After examining the processed workpiece, each video frame is assigned one of four quality labels, with category one representing a generally failed weld and category two to four referring to increasingly fine welds in accordance with the norm EN ISO 13919-1:1996 [31]. While there do exist 22 welds with multiple labels, the vast majority of the total 244 weld videos has been labeled with a single label assigned to every one of its frames. Table 3.1 provides an overview of the label distributions throughout all processes, revealing that some labels are severely underrepresented in certain processes.

	Class 1	Class 2	Class 3	Class 4
Process 76	52,636	59,135	44,500	11,099
Process 77	83,493	36,085	49,772	22,421
Process 80	1,951	22,589	14,620	14,821
Process 81	6,343	33,827	10,859	11,608

Table 3.1: Full label analysis of all images from the four welding processes. There are notable inbalances in label distribution that should be taken into consideration.

After diligent examination of the data at hand, several preprocessing steps have been

undergone to prepare the dataset.
First of all, several mislabeled welds have been dropped, excluding them from any further analysis. This includes three welds of process 76, namely W67, W69 and W70, which show labels that differ from the ones noted in the workpiece examiner's report. Furthermore, welds W26 to W30 of process 81 come with ambiguous labeling, for instance "3 to 4" or "1 to 4" instead of pointing to one definitive class. As the four used classes of the welding dataset are ordinally scaled and not necessarily have an interval interpretation, these labels are hard to interpret correctly and the mentioned welds have also been dropped from the dataset.
In a second step, the image data itself has been preprocessed. A box of 105x105 pixels was cropped from the center of the raw 144x176 pictures, assuming that most of the images' important information can be retrieved from the centrally located welding keyhole. Afterwards, the resulting square was downscaled to a more manageable 32x32 pixels using bicubic interpolation.
There are many other potential preprocessing measures that could be explored in future experiments, such as subtracting the dataset's mean image, adjusting for lighting conditions or increasing the number of training units by data augmentation measures like random image flipping and rotations.

3.2.2 Dataset Segmentation

Seperating a full dataset into training and testing parts is a crucial step that has to be conducted diligently. A correctly chosen test set can later be used to draw well-founded conclusions on the trained models' generalizability to classification problems not only within the train set, but the whole base population. If the test data is not sampled to be independent from the training data, however, the explanatory power of test set metrics is compromised and cannot be used to objectively draw conclusions on the base population.

Standard image datasets used for classification in literature generally avoid the aforementioned issues by only supplying the experimenter with absolutely independent units, sampled from a common and large base population. No matter how these units are later split into train and test sets, the generalizability of test metrics is therefore ensured.

The welding data used in this paper, however, has been generated differently and does come with inherent interdependencies between individual data units. Those data dependencies can be grouped on three levels: the image level, weld level and the process level.

- Image level: Stemming from a limited number of high-frequency video streams, individual images are obviously not independent from the rest of the dataset. Strong dependencies occur particularly with images in the same video, as the short 0.001 second time interval inbetween them makes adjacent frames visibly very similar and statistically highly interdependent.

- Video/weld level: Examining the dataset, one finds video labels not randomly distributed over the course of every process, but are correlated with time. The experiment operator obviously did not draw randomly but conducted similar experiments in close temporal proximity to one another. Other variables (e.g. lighting conditions, slight setup alterations or lense splatters such as the one above the keyhole in figure 3.1) are also correlated with time, resulting in potential correlation between videos/welds within the same process.
- Setup/process level: Within each setup or process, all units have been captured on the same day, at the same workstation, with the same physical equipment setup and type of lens. Comparing random samples of different processes, one finds obvious differences such as camera offset and varying levels of zoom that are strongly correlated with setups and could be identified by most analytical algorithms with ease.

Concluding from these examinations, it becomes apparent that the data at hand cannot be split into training and perfectly independent test set at any level. Strong dependencies inbetween images from the same weld video or from the same physical setup would render any conclusion drawn from a test set sampled at those levels less meaningful and generalizable. While there probabably also exist dependencies among images from different welds of the same setup, however, those interrelations do seem weaker and might not be completely inhibitory.

Faced with these insights, one has no choice but to separate training and testing units at the weld level, accepting a certain degree of dependency between the two. To circumvent inter-setup differences, the general population has to be restricted to a single setup or process. Since the welding process P77 showed the most balanced label distribution in table 3.1, all further experiments have been conducted on train/test splits sampled from different welds of a single process, P77.

This splitting approach leads however, as outlined above, not to perfectly independent train and test sets, which does in the end limit the explanatory power of test set metrics and should be taken into consideration whenever any generalizability conclusions are drawn. Furthermore, while all neural network architectures and training parameters are to be optimized using the welding dataset, identical models should also be trained on the MNIST dataset to verify any drawn conclusions. The MNIST dataset has been in use for an extensive period of time [30] and should not suffer from similar dependency issues as described above.

3.3 Autoencoding Networks

This section covers the decisions that ultimately resulted in the structuring and training of both the convolutional and traditional autoencoder.

While the general principles of both feature extraction approaches are preset by literature, the specific way a method is applied to the task at hand can differ immensely and depends on a large number of variables. To give both a fair chance, the optimization of each method has been constrained with the same time limit of 2 weeks as well as equivalently sized budgets of computational power for training and trying to improve models. All architectural decisions, parameters and training strategy insights have been obtained solely based on the welding dataset, although for the final evaluation have also been applied to the MNIST autoencoder training.

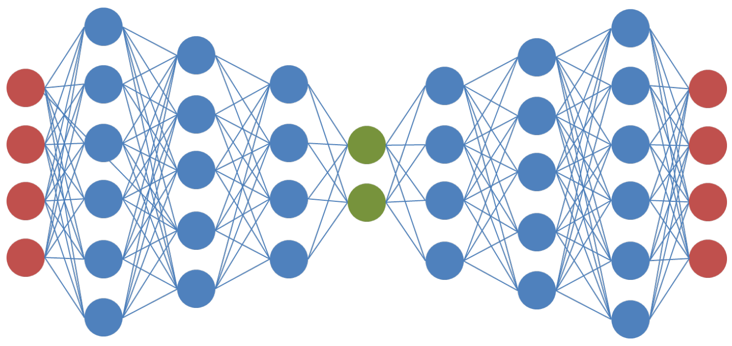
The parameter searching processes followed roughly the same pattern for all autoencoder variables. Holding other parameters constant at a standard default value, one or two would be varied over a narrow spectrum and resulting gains or losses in reconstruction performance noted. While this approach assumes independence between varied and held-constant parameters, an exhaustive search over all combinations seems unfeasable within a reasonable timeframe.

3.3.1 SDA Architecture

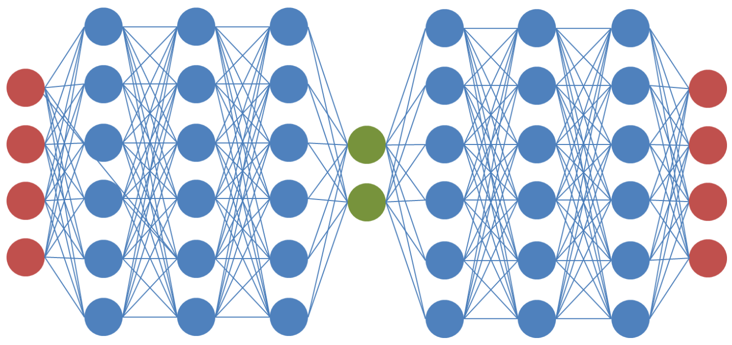
While one could imagine lots of potential autoencoding architectures, those covered in literature can be sorted into two major groups, such ones of triangular and others of rectangular shape - figure 3.2 provides an illustration of the major differences. With the number of input nodes and bottleneck nodes being dictated by the data and the intended purpose of the extracted features, both types of structures' first layer typically expands the input dimensions by a factor in the realm of 1.5 or 2.0 [13, 32]. Triangular SDA follow this expansion by continuously shrinking subsequent layers until sizing the last layer before the bottleneck at about 5.0 times its follower [13]. Rectangular schematics, however, do not decrease the size of subsequent layers, staying at high dimensionality right up to the bottleneck [33].

Preliminary experiments showed no advantage but in fact slightly inferior performance in sample triangular autoencoders against rectangulars of equivalent size. Furthermore, rectangular design makes varying the depth of the autoencoders, as in total number of layers, simpler as other layers' sizes do not have to be adjusted with every additional layer.

Within the realm of rectangular architectures, multiple layer sizes - $\{1024, 2048, 4096\}$ neurons - as well as different levels of depth - $\{2,3,4,5,6,7\}$ layers - have been examined. While reconstruction performance scores showed to improve with depth as well as layer size, both those factors increase the total number of nodes linearly, making training significantly more time intensive. Choosing the maximum value for both variables, most



(a) Triangular Shape



(b) Rectangular Shape

Figure 3.2: Autoencoders typically come in one of two different shapes. While both usually expand in dimensionality right after the input nodes (red), one type steadily decreases its layer volume towards the bottleneck nodes (green), while the other keeps it constant.

models seemed to reliably approach a good performance after only about 50 hours of total training time, which seems high but still manageable.

While in their original work Vincent et al. apply binomial masking noise to the training data of their denoising autoencoders, simply setting some values to zero at a preset probability [32], the welding and MNIST images of the following experiments are corrupted using additive gaussian noise, which is a "natural choice for real valued input" [22]. Before every individual SDA training step, to compute the corrupted input

$$X_{corrupted} = X + \mathcal{N}\left(0, \left[d * \sigma\left(X\right)\right]^2\right) \tag{3.1}$$

the input data X is modified by adding a value drawn from a gaussian distribution with mean 0 and a standard deviation of d times the standard deviation of X. The factor d therefore allows to control the degree of corruption in proportion to the standard deviation inherent to the signal. For all SDA training in the following experiments, d has been set to 30%.

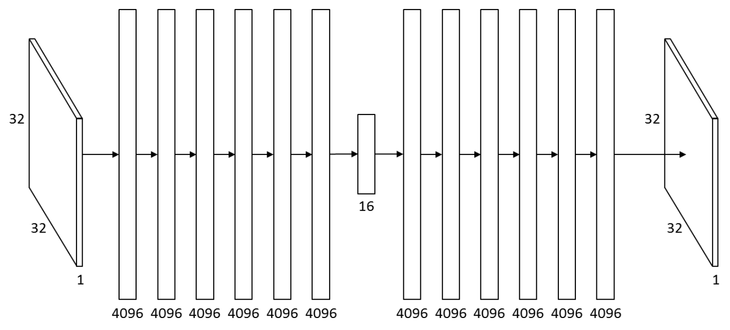


Figure 3.3: Schematic of the full final SDA architecture following the style of [2] - see figure 2.4 for reference - solid arrows represent the fully connected MLP layers. This network has a total number of 176,342,032 learnable parameters.

Summarizing this section's examinations, figure 3.3 depicts the final SDA architecture used for all following experiments. The encoding and decoding network parts both consist of six fully connected layers with 4096 neurons each, with a 16 neuron bottleneck inbetween. The input images' spatial structure is lost with the first encoding layer, and each of the consecutive layers then maps the data to a new representation. Finally the bottleneck enforces that the a compact feature set representing the original image is found. The decoding layers, on the other hand, ensure that the information lost in the encoding process remains minimal.

3.3.2 SCA Architecture

While research concerning standard denoising autoencoders and convolutional neural networks is quite abundantly available, a combination of both - namely convolutional autoencoders - can only scarcely be found in literature. Masci et al. introduce SCA, examine learnt filters and present classification scores, but only to very limited extent mention the architectures employed to achieve those results [3]. The following SCA architectural decisions therefore have - while certain key elements are the result of variation and experimentation - primarily been guided by state of the art CNN design, specifically the "AlexNet" architecture [2] and the very practical recommendations from Stanford University's CS231n lecture on visual recognition with CNNs [34].

Specifically, the examined architecture options follow a pattern described by Li and Karpathy as "the most common ConvNet architecture" [34] - alternating between several convolutional and max pooling layers, followed by 1 or 2 fully connected layers.

Table 3.2 outlines all convolutional and pooling layer combinations that have been considered. While additional convolution layers seemed helpful, significant amounts of max pooling quickly proved to be obstructive to good reconstruction performance. For any row in table 3.2, the rightmost architecture, using only a single subsampling layer, showed the smallest reconstruction error. Delving into larger structures with up to 7 convolutional layers, performance improved with network size but only up until 5 convo-

	3 Pooling Layers	2 Pooling Layers	1 Pooling Layer
1 Convolutional Layer	-	-	CP
2 Convolutional Layers	-	CPCP	CCP
3 Convolutional Layers	CPCPCP	CCPCP	CCCP
4 Convolutional Layers	-	CCPCCP	CCCCP
5 Convolutional Layers	-	CCCPCCP	CCCCCP
6 Convolutional Layers	CCPCCPCCP	CCCPCCCP	CCCCCCP
7 Convolutional Layers	-	-	CCCCCCCP

Table 3.2: The combinations and sequential order of convolutional (C) and max pooling (P) layers that have been considered in the SCA architecture optimization process.

lutional layers, and showed slightly decreasing results for 6 and 7 layers. While several architectures were tested with multiple max pooling layers at varying positions, the best performing model therefore only had a single max pooling layer, namely CCCCCP.

In line with recommendations [34] and the "AlexNet" reference architecture [2], the final, pooling layer is followed by two fully connected layers before leading up to the bottleneck. SCAs have showed best reconstruction errors if those are sized at 4096 neurons, although several alternatives {1024, 2048, 4096} have been considered.

Only limited experimentation was conducted with the size of convolutional layer filters, in most experiments and the final architecture only 3x3 filters of stride 1 and padding 1 are employed. Although both the ILSVRC 2012 and 2013 winners make use of large convolutional filters - {11x11, 5x5} and {7x7, 5x5} in their respective first two network layers [2, 35], these large filters typically are a only used as a compromise, helping to deal with large input images and GPU memory limitations [34]. In fact, a stack of small filter convolutional layers is preferable to one big filter layer as they allow "to express more powerful features of the input" [34]. The ILSVRC 2014 runner up confirms this as it successfully employs the same convolution strategy with 3x3 filters and padding of 1 [36]. One notable disadvantage, however, is the reduced interpretability of such small convolutional filters' visualizations. As the welding data raw images are available in substantially greater resolution than 32x32, future research may benefit from increasing the image resolution while at the same time also enlarging the first convolutional layer's filter size to counteract computational issues.

State of the art CNNs show similar strategies for setting the number of each convolutional layer's filters, namely increasing them to 150%-250% of the preceding layer's number, most commonly to 200%, while then keeping the number relatively constant in the second half of the network [2, 35, 36]. Following this observation and several trial runs to find the maximal number of filters while maintaining training time at SDA level, the final architecture's 5 convolutional layers come with 4, 8, 16, 16 and 16 convolution filters each.

Examining different pool sizes for the max pooling layers confirmed the aforementioned conclusions - any additional pooling, which includes increasing the pool size from 2x2 to 3x3 or 4x4, impedes reconstruction performance. The final architecture therefore only includes a single max pooling layer with a pool size of 2x2 and stride of 2.

While they have not been part of variation and optimization, the decoding layer strategy should be explained to allow the reproduction of all SCA experiments. SDA decoders are straightforward - as they can map from any desired input to any output dimensionality, constructing the corresponding decoding to a fully connected encoding layer merely means copying its number of nodes. SCA, on the other hand, also employ convolutional and subsampling layers that require more contemplation. With the convolution operations used in the encoder applying padding to maintain activation map dimensions (they all remain at 32x32), reversing the operation has been conducted by simply applying the same convolutional filter size, but reducing the number of filters to the desired output depth. Other padding strategies, such as full or valid convolutions, can also inverted by modifying the decoding convolution layer's padding. Finally, 2x2 max pooling has been undone by simply repeating the values at hand until a patch of desired size was filled. While autoencoders in literature sometimes use weight tying "forcing encoder and decoder matrices to be symmetric" [22], this strategy has for simplicity neither been applied to SDA nor to SCA decoding layers.

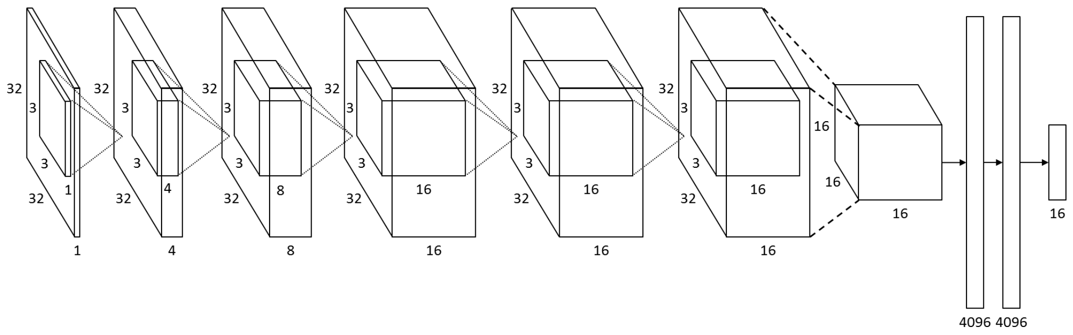


Figure 3.4: Schematic of the final SCA architecture up to the bottleneck. The identical decoding half has been omitted because of space limitations. The schematic follows the style of [2] - dotted lines represent convolution operations, dashed lines max pooling while solid arrows show fully connected layers. The SCA has a total number of 67,272,705 learnable parameters.

Concluding this section, figure 3.4 outlines the schematic of the final SCA encoder's architecture. Following five convolutional layers of an increasing number of filters, a 2x2 max pooling layer reduces the data throughput to 16x16x16. After two 4096 neuron fully connected layers, the bottleneck compresses the data to a representation of just 16 features. As the decoding half of the SCA is symmetrical to the encoder, it has been omitted because of space considerations.

3.3.3 Autoencoder Training Strategy

Bergstra and Bengio show theoretically and through empirical study that in deep neural network training "only a few of the hyper-parameters really matter" [37], concluding that hyperparameter optimization is ideally not done in a rigid grid search fashion, attempting to tweak every variable, but with random search. They suggest experimenters should prioritize the optimization of those variables they deem important for the dataset at hand, while allocating less time to the others.

In line with these suggestions, the majority of time for optimizing SDA and SCA has focused on finding suitable autoencoder network architectures, as this is a novel area that comes with comparatively few best practice references. Neural network training strategy, on the other hand, is extensively covered in literature which allows to make sound decisions without excessive experimentation. In contrast to many architectural choices of sections 3.3.1 and 3.3.2, which have been examined and tailored to a certain degree, the following variables and decisions have therefore been held constant for both SDA and SCA.

- Training batch size has been set to 128 images, which is within the realm of Bengio's recommendations [38].
- Rectifying nonlinearities have been used for all neuron activation functions. They are frequently employed in CNN architectures as they speed up model convergence [2] and experimental results evidence their benefits "especially for deep architectures" [39].
- Learning rate has been set to a schedule linearly decreasing over the course of training epochs, from initial 0.1 to final 0.0001.
- Momentum has been set to a schedule linearly increasing over the course of training epochs, from initial 0.9 to final 0.999. The combination of both the learing rate and momentum schedule have been motivated by the work of Sutskever et al. [40].
- Weights have been initialized according to the "GlorotUniform" strategy [23], drawing from a random uniform distribution that is bounded relative to incoming and outgoing network connectivity. No encoder uses any kind of weight tying with their respective decoder.

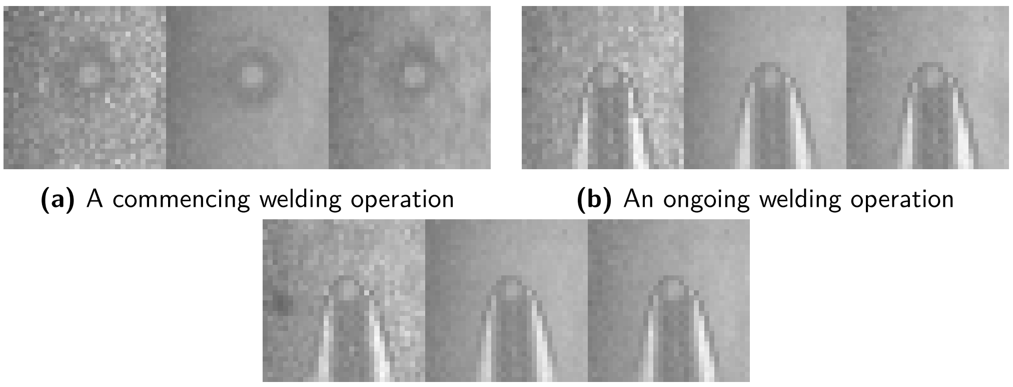
Finally, while Bengio et al. that suggest layer-wise pretraining initializes "weights in a region near a good local minimum" [41], despite significant experimentation no variant of either SDA or SCA has ever outperformed completely random initialization. Consequently, all final autoencoders have not been pretrained layer by layer before their full network optimization.

4 Results and Discussion

The following sections focus on key findings and resulting conclusions from the experiments. Beginning with the trained models themselves, the examination commences with an analysis on what happened during the autoencoding processes, namely how well they were able to reconstruct the original data and how this has been achieved. The subsequent, main part investigates on the utility and applications of the generated bottleneck features, specifically how useful they are for image classification as well as how robust they are against forcefully induced input variations.

4.1 Reconstruction Performance

Commencing the evaluation process, this preliminary comparison takes a look at the autoencoder training objective, the reconstruction of the original image.



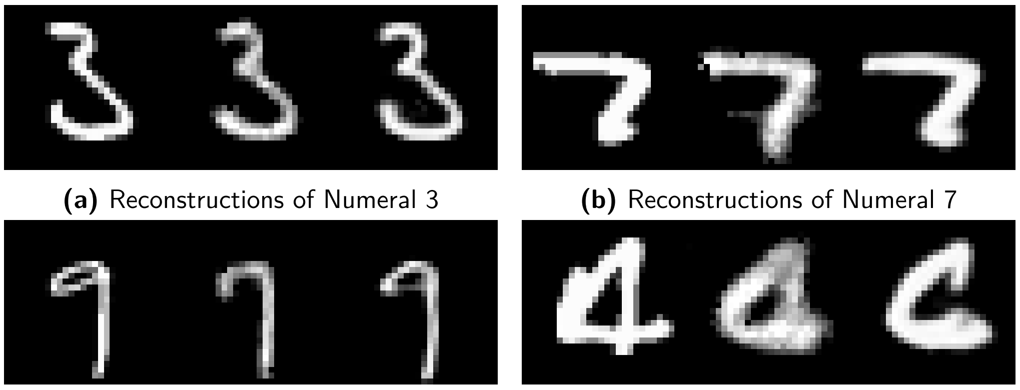
(a) A commencing welding operation **(b)** An ongoing welding operation

(c) Image with uncommon dark splatter

Figure 4.1: Original welding images and their reconstructions created by SDA and SCA. Although most characteristics of the originals seem present in the reconstructions, uncommon marks are not reproduced (c).This selection of images is not representative of the base population.

Figure 4.1 illustrates key differences between the welding image originals and their autoencoder reconstructions. Both autoencoding processes seem to smooth out background noise, particularly any information that is only ever represented within a very small number of input images. A dark splatter such as in the left half of Figure 4.1c, for instance, is not present in any of the two reconstructions since learning to encode its characteristics and location would only have helped to improve the reconstruction of a very limited number training images. It might also be interesting to note that the SCA reconstructions seem slightly less homogenous than the SDA's, carrying some additional information on their originals' texture and condition.

Examining reconstructed images from the MNIST dataset, one finds the overwhelming majority to be very close duplicates to their originals - as illustrated in Figure 4.2a. The autoencoders do experience problems, however - notably with such drawings that diverge from the "standard" form and shape of the depicted digit. In particular, an autoencoder may learn that some features typically appear together within many images - such as the three strokes within the digit seven. Faced with only two of those strokes, it then may attempt to add the third one on its own, especially if there have not been enough two stroke sevens within the training data set (see Figure 4.2b).
Another problem case consists of sloppy or unusual digit designs. Figure 4.2c, for instance, depicts a quite unique design of the numeral nine, with many features resembling the two strokes of the numeral one. Consequently, SDA and SCA both struggle to reconstruct the complete closed circle, since leaving out the lower semicircle is substantially more in line with what is to expected with the numeral one.
While these problems occur only rarely, they do lead to potential misinterpretation of the reconstructed data. Figure 4.2d shows such an instance, when a sloppily drawn four is compressed and reconstructed in something that could be interpreted as a six or zero, but probably almost never as a four.


(a) Reconstructions of Numeral 3 **(b)** Reconstructions of Numeral 7

(c) Reconstructions of Numeral 9 **(d)** Reconstructions of Numeral 4

Figure 4.2: Original MNIST input images and their reconstructions created by SDA and SCA. Most are well on point (a), however some types of input do cause problems (b,c,d). This selection of images is not representative of the base population.

While figure 4.6 will later go into detail on the improvement over the course of training, at this point the final mean squared reconstruction error of 0.000887 for SCA and 0.0012367 for SDA should be noted. Summing up, the reconstruction performances of SDA and SCA seem decent and generally indicate that only little information is lost when encoding images to a low bottleneck dimensionality. Both autoencoders do have issues, however, particularly when faced with information that is not represented very frequently within the training data.

4.2 Autoencoding Process Analysis

Attempting to reveal the inner workings of SCA and SDA, this section tries to examinine the SCA and SDA encoding and decoding processes themselves. Both autoencoders have experienced similar time budgets for architecture design and testing as well as the training and optimization of the final models, and yield features of comparable reproduction quality. They do, however, differ in their approaches to transforming data representations from layer to layer. SCA do maintain - albeit often reduce - the original images' two- or threedimensional structure, whereas SDA are fully connected from the very first layer, allowing all nodes to process input from any given location in the preceeding image data. As a result, visualizations of images as they pass through a stack of layers really only make sense for trying to understand the inner workings of a SCA.

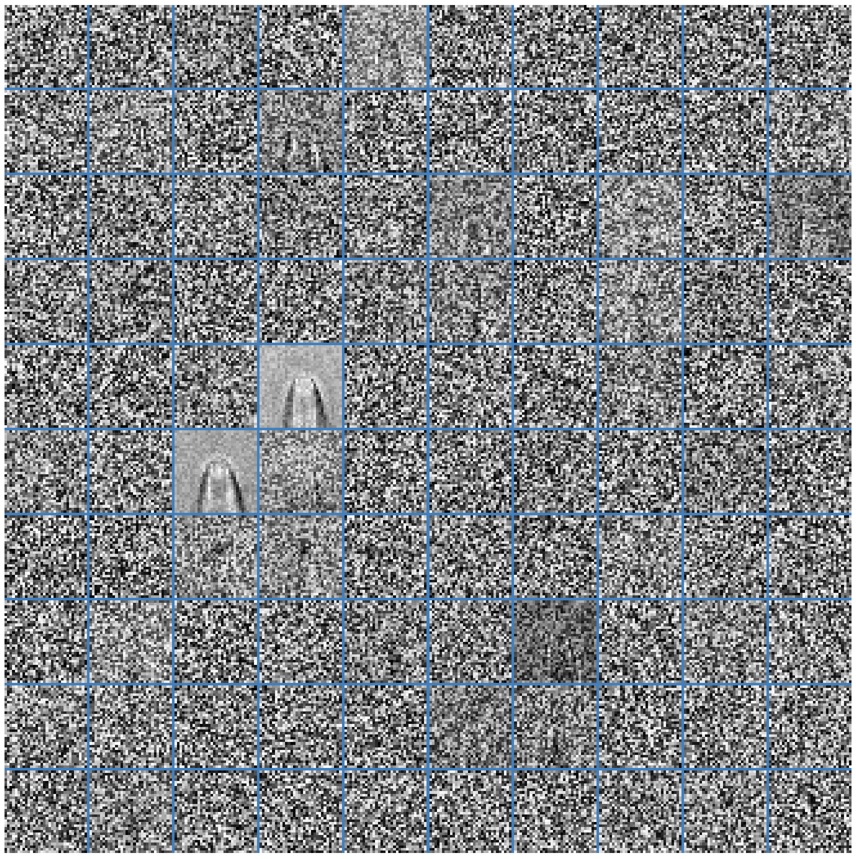


Figure 4.3: Weights learnt by a sample of 100 neurons of the SDA's first encoding layer.

The neurons of an SDA's first layer, however, have individual connections with and store a weight for every single input pixel. Plotting those weights arranged in the input's spatial structure therefore offers an opportunity to see what happens at least in this first SDA layer. Looking at a randomly sampled selection of 100 of these weight maps in Figure 4.3, one sees starkly varying levels of specialization from neuron to neuron. While some few appear to be heavily fitted to recognize specific input images, many appear to be able to recognize what could be described as faint depictions of geomentric shapes, mainly at the location of the welding keyhole and welding seam. Most neurons, however, seem not fitted at all. While it is entirely possible that the depiction of their weights just appears random to the human eye - they could, for instance, actually be detecting meaningful features in cooperation with other neurons - their output will most likely be either ignored or underprioritized in subsequent neuron layers.

Examining the autoencoding process within SCA, on the other hand, one has the opportunity to visually inspect how the data representation evolves from layer to layer. Figure 4.4 shows how a welding image is transformed by the weights of every convolutional layer, revealing multiple interesting insights.
First of all, one notices that the dissimilarities between layers' activation maps seem to slightly increase with each consecutive layer, possibly hinting at increasing levels of specialization. The activation maps right before the bottleneck layers, while probably not each encoding entirely distinct sets of features, could be seen as each having a focus, highlighting an admittedly vague but different aspect of the original input data.
Secondly, upon close examination, it becomes apparent that the max pooling layer within the bottleneck has an effect on the spatial arrangement of the activation maps. The two by two pixel max pooling essentially leads to 75% of all data being lost from every individual activation map. It seems like the autoencoder circumvents this issue by shifting several of its activations by one pixel, so that otherwise missing information can possibly be retrieved from these dedicated maps. This phenomenon can be observed by comparing the activation maps of Figure 4.4, upon which one notices that not all activation maps are vertically aligned with each other.
Lastly, while the majority of encoding activation maps seem to be generally activated and do resemble the original input image, a large portion are primarily under-activated and black. An examination of the corresponding weights and biases of those activations does not confirm that their sums are just generally of low absolute values, indicating that these activations do indeed represent convolutional filters that are strongly stimulated by a very small subsection of their input and much less in other areas. Albeit the argument certainly is a speculative one, one could identify the relatively homogenous nature of the welding data as the reason for why this phenomenon seems far less pronounced with the MNIST dataset in Figure 4.5, since not much information on the original image is required for reconstruction.
Looking at the reconstruction of the welding images, one notices that although some faint structures do seem to emerge layer by layer, shapes remotely resembling the input are only noticable after the fourth reconstructing convolutional layer. This stands in

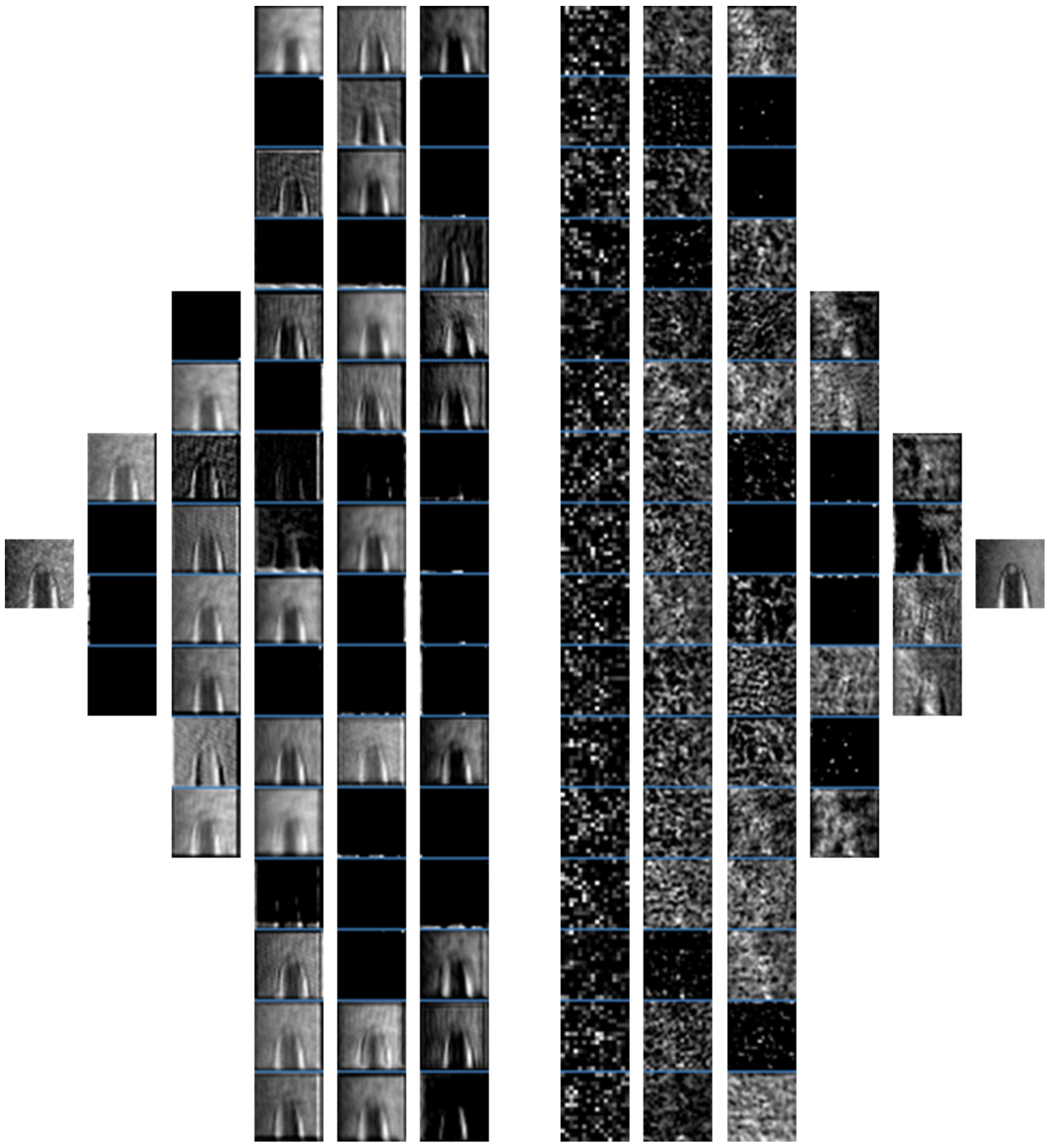


Figure 4.4: A welding image is processed by all layers of a convolutional autoencoder. Since they do not have a relevant human interpretable spacial structure, the fully connected and bottleneck parts inbetween are not displayed.

stark contrast to the MNIST data reconstruction of Figure 4.5, in which the original numeral is already decipherable at its first reconstruction layer's activation maps.

Taking aside the welding dataset and having a closer look at the activation maps stemming from the MNIST convolutional autoencoder in 4.5, a few other differences stick out. First and foremost, one notices that the encoder seems more proficient in

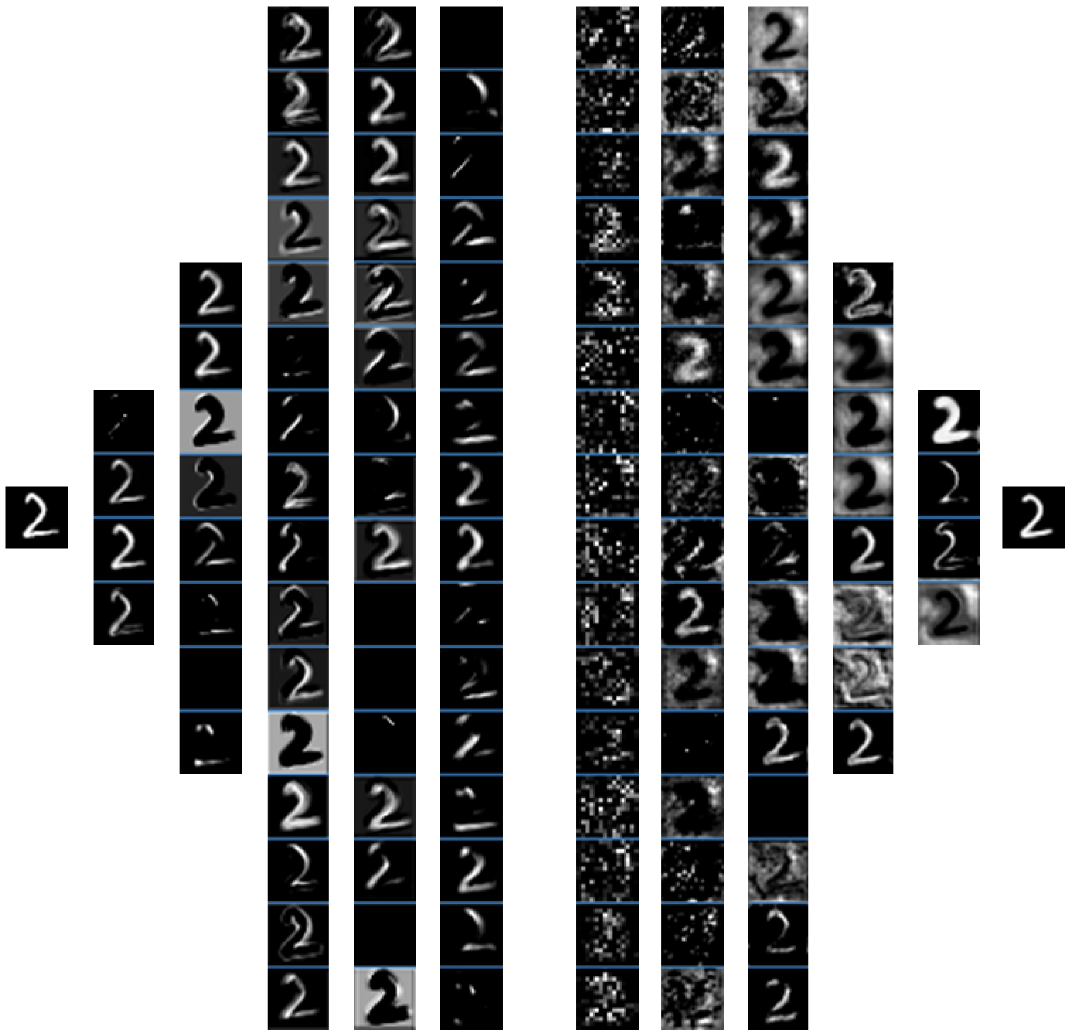


Figure 4.5: An image of the numeral two is processed by all layers of a convolutional autoencoder. Since they do not have a human interpretable spacial structure, the fully connected and bottleneck parts inbetween are not displayed.

deconstructing the input image into more destinct features. While most activation maps right before the bottleneck still are complete enough to guess the input numeral, they appear quite focused on individual features such as horizontal, vertical or inclinated stokes.

The reconstruction, on the other hand, takes a more holistic approach. Almost every activation map, starting from the very first layer of reconstruction, depicts not single features or strokes but shapes that resemble the complete original structure. While they are still quite noisy and vague in the beginning, every continuous layer seems to come with sharper and sharper activations, culminating in the second to last layer with four very distinct and well specialized convolution maps. One can easily comprehend how those four signals, two of which seem to carry the general shape and two the contours of the numeral two, can be combined to generate the sharp-edged reconstruction of the original input.

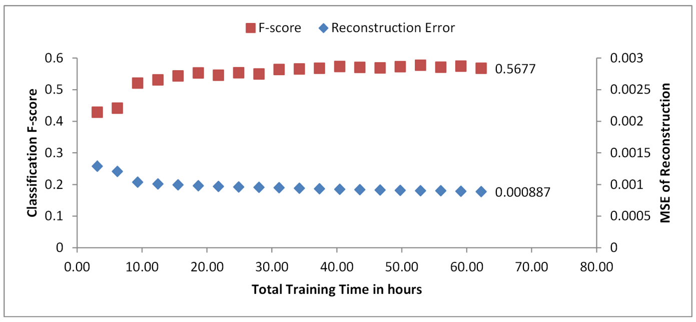
4.3 Feature Analysis

While the previous sections were covering the methodology and procedures of the autoencoding processes themselves, the following attempts to evaluate the eventual product of those actions, the lowdimensional bottleneck features.

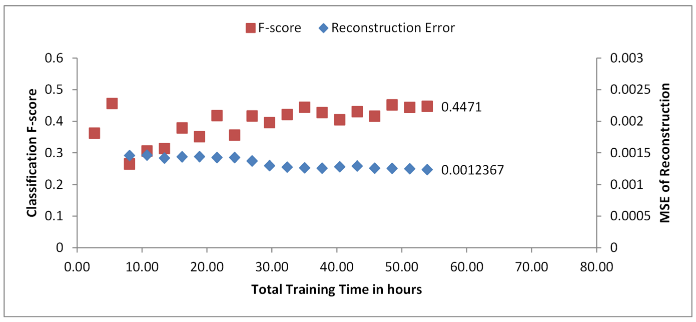
4.3.1 Classification Performance

Apart from measuring reconstruction error, which essentialy rates how much of the total original information can be recovered in the autoencoding process, there are other ways to objectively judge the features' potential utility. One can, for instance, measure how easily a classifier like an SVM can infer higher level information using those features, such as human given labels on the original images. For the welding dataset, this means identifying one of four distinct welding irregularity categories, three of whom are in accordance with the norm EN ISO 13919-1:1996 [31] and one representing general failure. The MNIST dataset, on the other hand, requires the identification of one in ten numerals. The metric for judging these SVM classification performances will in the following always be the F-score, which takes into account both classifiers' precision and recall concerning one class. Multi-class F-scores are then attained by computing the unit weighted mean of the individual class F-scores.

Figure 4.6 showcases the training progress for SCA and SDA on the welding dataset. For both autoencoders, prolonged training seems to result in decreasing reconstruction error as well as increasing classification F-score. The SCA metrics' development seems quite monotonous and quickly approaches a saturation level, particularly concerning its F-score. The SDA, on the other hand, shows a lot more volatile performance swings over the course of training. Its best F-score (0.456) is actually already obtainend after only 5 hours or 100 epochs of training, when its reconstruction error (0.11) is still exceptionally high. Looking at the complete figure, however, one can clearly identify this value as an outlier and make out a generally positive trend over time, signaling that in fact the SDA's F-score does benefit from continued training and lower reconstruction errors. Acknowledging the overall picture, it can be stated that in this application its convolutional elements seem to give the SCA an edge over the conventional SDA. The SCA generally outperforms over its counterpart with substantially lower reconstuction error (0.000887 against 0.0012367) and higher F-score (0.5677 against 0.4471) after both autoencoders have undergone 1000 epochs of training. Moreover, while an average epoch of training takes slightly longer on the SCA (224 sec. against 194 sec.), reliably



(a) SCA training on the full P77 dataset



(b) SDA training on the full P77 dataset

Figure 4.6: Performance metrics show decreasing reconstruction error and increasing F-score over the course of 1000 epochs of autoencoder training. Left vertical axis: F-score, right vertical axis: reconstruction error, horizontal axis: total training time in hours.

high performance can be expected to be reached a lot quicker with the SCA as its metrics seem a lot less prone to variance.

While those results appear coherent and their implications straightforward, they should be taken with a grain of salt. Training an SVM with the exact same parameters, but different input - not the 16 features stemming from SCA or SDA, but the 1024 original image pixel values - remarkably leads to an even superior F-score of 0.5895. While this finding is not completely counterintuitive - SVMs have been applied as a solution to raw image data of similar dimensionality [27, 42, 43, 44, 45] - it puts in question both the implementation of SCA and SDA as well as the welding dataset itself. Since both autoencoders seem to produce visually and statistically good reconstructions, however, one is inclined to suspect the data or its labeling as the primary reason for the overly high performing SVM baseline. On the one hand, the classification into four weld categories might therefore not be as hard a problem as one imagines - it might for instance be that most required information already comes with the few pixels of the welding keyhole and be relatively simple to identify. One could further imagine issues of data dependency between individual welds - as mentioned in section 3.2.2 - to make the problem at hand considerably simpler for the classifier as it no longer needs to focus on quite hard to identify high level weld characteristics and features to make a decision.

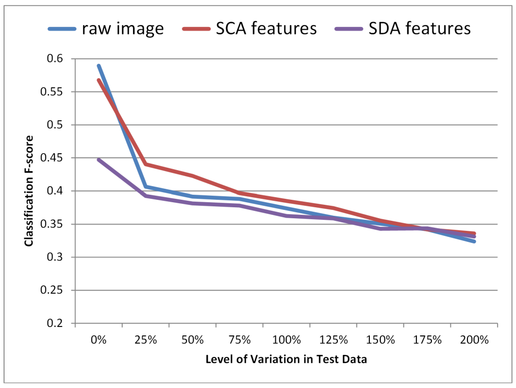
Format of the Classified Data	Best Reconstruction Error	SVM F-score
Welding Data raw	-	0.5895
Welding Data SCA features	0.000886926	0.5677
Welding Data SDA features	0.0012367	0.4471
MNIST raw	-	0.9816
MNIST SCA features	0.00703701	0.9811
MNIST SDA features	0.012068	0.9782

Table 4.1: Detailed performance report of the best performing architectures for both autoencoder features and the raw image data on the welding and MNIST datasets.

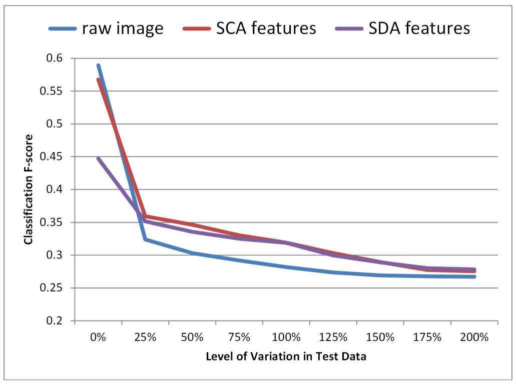
Examinig the architectures' performance on the reference MNIST dataset in table 4.1, one can identify similar patterns to the welding dataset results: SCAs appear to be better at image reconstruction than SDAs, and the overall best classification result comes from an SVM trained not on features but on the raw image data. The F-score based on SCA feature classification follows with only a small distance while the SVM working with SDA features shows the lowest overall classification performance.
The good result achieved with the SVM on raw MNIST image data is in line with similar experiments in literature [44, 45], indicating that the experiment setup itself is solid and not erroneous. One could still wonder why a black box SVM approach that does not even take into account the spatial relations of the images' data points can outperform a dedicated solution like SCA feature extraction. The differences in performance scores, however, are quite low and as the autoencoder architecture and training parameters have been optimized for the welding dataset a, there is still room for improvement by dedicated optimization of those parameters for the MNIST dataset. Furthermore, the following section will show that a feature extraction approach does provide benefits in robustness against input variation.

4.3.2 Feature Robustness

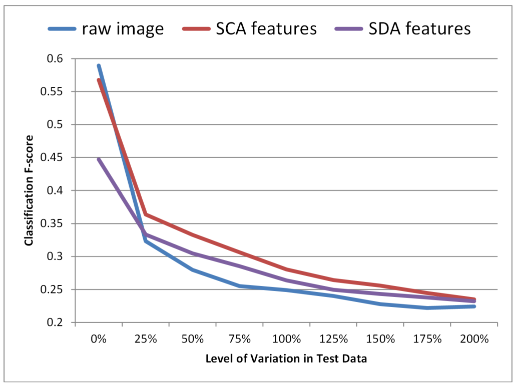
Finally, both autoencoders' susceptibility to varying input is put to the test. All base models have been trained and optimized on relatively uniform data with low levels of translational, rotational and scale variance. These trained autoencoders along with their fitted SVMs are now used to extract features and try to classify datasets with artificially induced variance of varying severity.



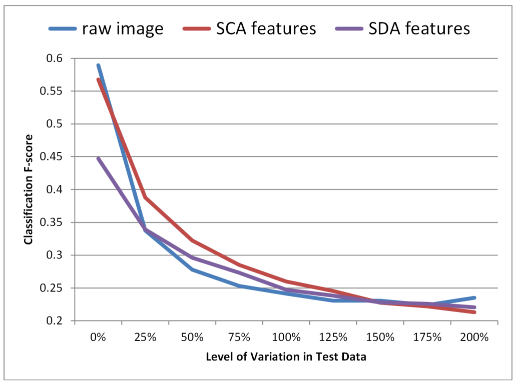
(a) Rotational Variation



(b) Scale Variation



(c) Translational Variation



(d) All three types of Variation

Figure 4.7: Weld classification performance decreases as features lose interpretability with increasing levels of variation in the welding test dataset.

Analyzing Figure 4.7, one can clearly see classification performances deteriorating gravely with only little levels of any type of unexpected variance. Even though the SDA features initially perform substantially below SCA levels (0.4471 against 0.5677), this gap quickly narrows, leaving less and less of the original advantage, until most differences seem to have completely disappeared at 200% variation. Although the gaps do narrow, one could argue that during most types and intensities of variation, a clear hierarchy can be made out - the SCA features seem least affected, followed by SDA features, and an SVM trained on raw input pixels is disrupted the most by input variation.
Another interesting conclusion lies in the different rates at which all models seem to suffer from different types of variance. Generally speaking, translations, i.e. shifts of the full image in a random direction, have the most severe effects on classification, followed by random rescaling and finally rotational variations. One could argue that particularly with rotations, the degree of impact changes heavily depending on a pixel's location

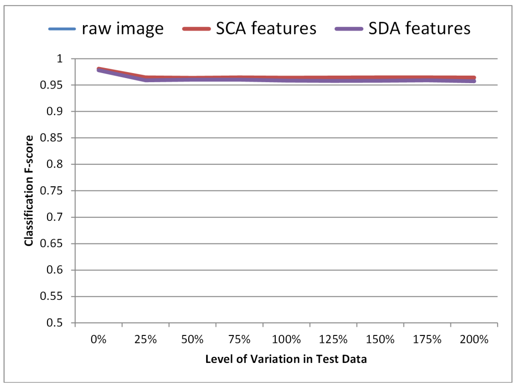
within the original image. Is information required for determining an image's class primarily encoded within the image center - a plausible assumption given that this is where typically the welding keyhole is located - then a small rotation of the whole image might not distort those essential features severely. Translational distortions, however, affect the whole image equally and could potentially only be dealt with by making the feature encoding process invariant to changes in location.
Lastly, one peculiar effect seems to be induced by high levels of translational variation: It pushes the classification F-score below 0.25, the theoretical minimum value for random guessing. Since the metric was obtained on a large sample size of 12800 units, an explanation can only stem from an issue in the experiment setup itself or an actual causal relationship linking high level of translational variation with misclassification. Table 4.2 allows for a more detailed examination of the overall worst result on record, an SCA's F-score of 0.2132 in Figure 4.7d, when faced with 200% of all three types of variance. Taking a close look at table 4.2, one finds the low overall low score not linked to low levels of precision - which seem to be reasonably random distributed around their expected level of 0.25 - but rather with low levels of recall, particularly for units of the classes 2 and 4.

class	precision	recall	f1-score	support
1	0.24	0.52	0.33	3200
2	0.23	0.14	0.17	3200
3	0.26	0.30	0.28	3200
4	0.29	0.04	0.07	3200
avg / total	0.26	0.25	0.21	12800

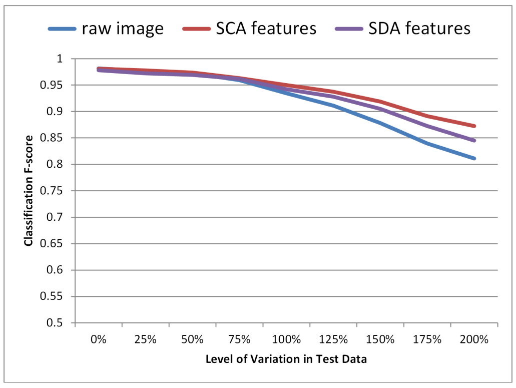
Table 4.2: Detailed by-class report of an SVM's performance on SCA features stemming from heavily variance induced welding data.

Consulting the SDA feature and raw data models that also inexplicably underperform at the same level of 200% induced variance confirms this finding. The SVM trained on raw input shows a recall of 0.10, the one trained on SDA features a recall of 0.09 for class 4. While definitely not an indisputable conclusion, a possible explanation for this low level of recall particularly with one specific label class and translational variance might be that those features required to make a determination for class 4 are all found in specific locations of the image, such as the central keyhole or the welding seam. In absence of these features at their expected location because of the translational shifts, a classifier might deem another class more likely, leading to an underrepresentation of class 4 in the predictions and explain the overall low recall and F-score.

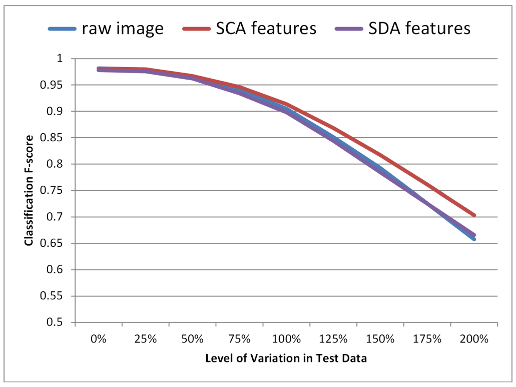
Moving on to the effects of induced variation on the MNIST dataset, figure 4.8 paints a quite different picture than its welding counterpart. While even slight levels of variance have a huge impact on the classification of the welding images, the MNIST features seem a lot more resilient.



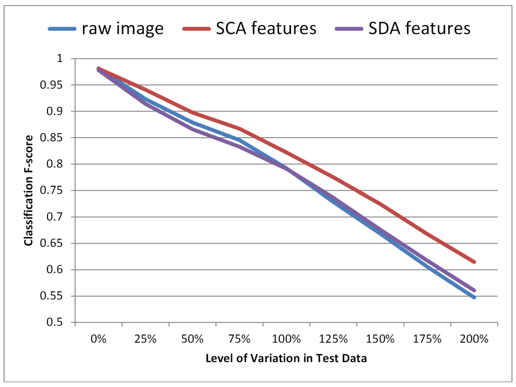
(a) Rotational Variation



(b) Scale Variation



(c) Translational Variation



(d) All three types of Variation

Figure 4.8: MNIST classification performance decreases as features lose interpretability with increasing levels of variation in the MNIST test dataset.

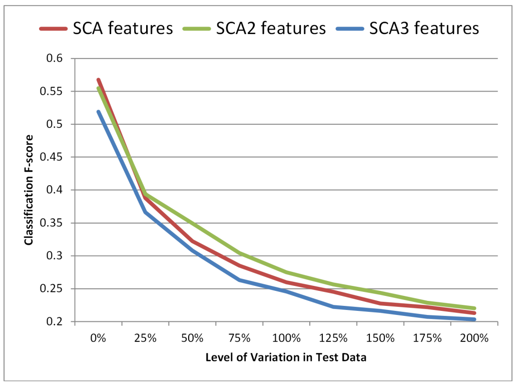
One could argue that since the individual MNIST images have been generated by many seperated, heterogenous processes - namely human writers, its training data already comes with a comparatively high variation, particularly of scale and rotational variance. The welding images, on the other hand, have been generated within a single experiment setup, resulting in relatively similar and homogenous training data. The MNIST data hence forces both autoencoders and classifiers to learn to cope with some degree of variance, a fact that manifests itself in figure 4.8, which in contrast to the welding plots of figure 4.7 shows curves of a rather slowly falling concave, not dwindling convex character. Another possible explanation for the excessive robustness of the MNIST features might lie in the nature and number of features that identify the image classes. It is conceivable that while the weld classification hinges on just a low number of features - as argued in the explanation for the diminutive F-scores above - the numerals of the

MNIST dataset each offer a full series of unequivocal features. Even if added variance were to mask several features, there would therefore still be a chance for some still being visible enough to make a correct classification.

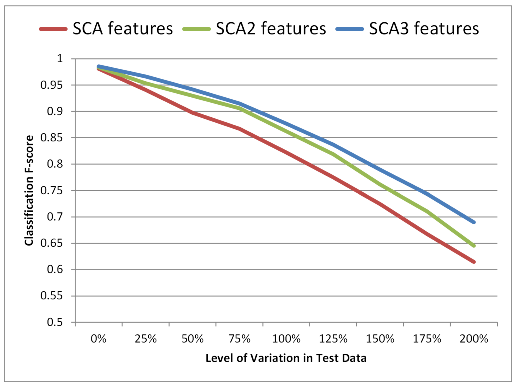
4.3.3 Impact of Additional Max Pooling

The previous sections' SCA architecture decisions have been guided solely by the optimization of reconstruction and classification scores. While that has lead to models performing well on the test dataset, particularly the welding data features are quite susceptible to even low amounts of input variation, as depicted in figure 4.7. Having a look at the SCA architecture employed, one might wonder whether the low usage of max pooling - only a single layer of 2x2 pooling filters is put to use - is one reason for these observations. In order to assess the influence of higher degrees of max pooling on SCA model robustness, two additional autoencoding architectures have therefore been trained on the welding and MNIST training datasets to test if indeed robustness can be influenced by altering max pooling.

The first of these new test models (SCA2) comes with a larger max pooling layer of 3x3 instead of the former 2x2, the second model (SCA3) employs no longer a single but three individual 2x2 max pooling layers that each follow one of the last three convolutional layers before the bottleneck. While they change composition and degree of max pooling, both new autoencoders keep all other architectual and training variables constant.



(a) Welding Dataset



(b) MNIST Dataset

Figure 4.9: Convolutional autoencoders with increasing levels of max pooling show different degrees of resilience against the mix of all three variance types. The red graphs show the previously evaluated SCA performances of figures 4.7d and 4.8d for reference.

Since the new architectures deviate from what has been deemed optimal for the welding dataset in section 3.3.2, one can reasonably expect a certain degree of deterioration in the test set scores. Unsurprisingly, figure 4.9a confirms these expected losses as the features stemming from the new architectures initially both show inferior classification

F-scores on the welding dataset, namely from 0.5549 (SCA2) and 0.519 (SCA3) compared to 0.5677 (SCA). After introducing variation to the test data, however, the SCA2 architecture outperforms its competitors at every single degree of induced variance, indicating the larger max pooling layer may entail superior feature robustness. The SCA3 features, on the other hand, underperform at any given level. While this particular finding cannot link increases in max pooling to feature resiliance, it should not be seen as evidence disproving a causal relation as the bad performance might be attributable to the for the welding dataset disadvantageous architecture.

Astonishingly, the MNIST classification seems to substantilly benefit from the new architecture designs as both show substantial gains in performance against the section 3.3.2 design. While the features extracted with the benchmark SCA obtain an F-score of 0.9811, SCA2 manages to achieve 0.9837 and SCA3 even 0.9857 on the unchanged testing data. These observations indicate that although the architecture tweaked for the welding dataset has performed quite adequately on the MNIST data, there may still remain room for improvement by dedicated optimization. After introducing variance to the test sets, both SCA2 and SCA3 retain their leads over the low max pooling architecture, particularly SCA3 which manages to further widen the gap, indicating that its features are indeed more robust.

While not all evidence is unequivocal, the tests illustrated in figure 4.9b indicate that increasing an SCA's degree of max pooling can positively affect feature robustness. Arbitrarily large or many max pooling layers will at some point certainly cause the loss of too much data and impede feature usability, but minimal max pooling approaches such as the section 3.3.2 design may render SCA encoded data representations more susceptibe to input variance than need be.

5 Conclusion

This work evaluates the stacked convolutional autoencoder (SCA), an in literature scarcely covered method for representation learning, in comparison with the more common stacked denoising autoencoder (SDA)

For finding suitable values for architectures and training hyperparameters, each of the two neural network types has been alotted an equal budget of 14 working days. Utilizing those values and giving both methods similar budgets of computing power, a final SCA encoder and an SDA encoder have been trained. The two encoders were then examined on two datasets of visual data. The first dataset was a real world application, namely videos from a series of 244 laser welding experiments, and the second one the standard MNIST dataset to allow verification of conclusions and comparability to other research.

Finally, both encoding methods' extracted features were evaluated. During the autoencoder training process, the SCA showed to loose slightly less information when transforming images into feature vectors than the SDA. Furthermore, a support vector machine was used attempting to infer images' labels from their new compressed feature representation. Experiments demonstrated the SCA encoded features to be more useful for this classification task than SDA generated representations. Finally, the invariance of SCA and SDA encoded representations was tested by adding increasing levels of transformational variation to the test datasets, expecting deterioration in classification scores. While the observations being more noticable on the MNIST dataset, the SCA features consistently showed slightly more robustness against unexpected variance than SDA features.

Apart from revealing practical recommendations to SCA and SDA architecture design and training (section 3.3), this work has also highlighted the importance of extensive analysis before splitting real world, non-curated datasets into training and test sets (section 3.2). The key takeaway, however, lies in the comparison of the different representations that SCA and SDA have learnt within the realm of visual data. Bengio et al. name both the disentanglement of the underlying factors causing data variation as well as the learning of abstract features that are invariant to local input changes as desirable properties for representations, particularly in conjunction with image data [1]. The aforementioned experiments have indicated that SCA encoded image representations are more useful for image classification applications. Specifically factors correlated with the datasets' labeling may therefore be present in the SCA features in a more disentangled fashion than in their SDA counterpart, simplifying the work of the following classification algorithm. Furthermore, SCA representations have shown higher levels of invariance towards small image transformations than SDA features. As these transformations are not informative to the task at hand, namely not masking or altering characteristics required for classification, a good data representation should be abstract and insensitive to such input changes [1].

Summing up, one can state that because SCA are able to exploit spatial relations and come with max pooling layers that naturally increase transformation invariance, they may be a better choice for representation learning in the realm of visual data than SDA. As particularly the time intensive autoencoder training experiments of this work have not been conducted in sufficiently large number and in enough architectural and hyperparameter configurations to yield statistically incontestable findings, these results should, however, be taken with a grain of salt and ideally confirmed by further research.

Bibliography

[1] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] Jonathan Masci, Ueli Meier, Dan Cireșan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 52–59. Springer, 2011.

[4] Dan-E Nilsson. Eye evolution: a question of genetic promiscuity. *Current opinion in neurobiology*, 14(4):407–414, 2004.

[5] Steven Tanimoto. *Structured computer vision: machine perception through hierarchical computation structures*. Elsevier, 2014.

[6] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[9] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Good practice in large-scale learning for image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):507–520, 2014.

[10] Juhan Nam, Jorge Herrera, Malcolm Slaney, and Julius O Smith. Learning sparse feature representations for music annotation and retrieval. In *ISMIR*, pages 565–570, 2012.

[11] Mark S Nixon and Alberto S Aguado. *Feature extraction & image processing for computer vision*. Academic Press, 2012.

[12] Erik Cambria, Bjorn Schuller, Yunqing Xia, and Catherine Havasi. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent Systems*, (2):15–21, 2013.

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[14] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386–408, 1958.

[15] Eric B Baum. On the capabilities of multilayer perceptrons. *Journal of complexity*, 4(3):193–215, 1988.

[16] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[17] G Cybenco. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals and Systems*, 2(4):303–3014, 1989.

[18] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pages 2924–2932, 2014.

[19] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[20] Thomas Serre, Gabriel Kreiman, Minjoon Kouh, Charles Cadieu, Ulf Knoblich, and Tomaso Poggio. A quantitative theory of immediate visual recognition. *Progress in brain research*, 165:33–56, 2007.

[21] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294, 1988.

[22] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[23] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[24] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[25] DE Rumelhart GE Hinton RJ Williams and GE Hinton. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[26] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[28] Jason Weston and Chris Watkins. Multi-class support vector machines. Technical report, Citeseer, 1998.

[29] Asa Ben-Hur and Jason Weston. A user's guide to support vector machines. *Data mining techniques for the life sciences*, pages 223–239, 2010.

[30] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998. Accessed: 2016-04-25.

[31] EN ISO. 13919-1, 1996. *Welding: Electrons and Laser Beam Welded Joints. Guidance on Quality Levels for Imperfections. Part 1: Steel*, pages 13919–1.

[32] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[33] Jonas Gehring, Yinping Miao, Florian Metze, and Alex Waibel. Extracting deep bottleneck features using stacked auto-encoders. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3377–3381. IEEE, 2013.

[34] Fei-Fei Li and Andrej Karpathy. Cs231n: Convolutional neural networks for visual recognition, 2015. Accessed: 2016-04-25.

[35] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer vision–ECCV 2014*, pages 818–833. Springer, 2014.

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

[37] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

[38] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer, 2012.

[39] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

[41] Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153, 2007.

[42] Jan Eichhorn and Olivier Chapelle. Object categorization with svm: kernels for local features. Technical report, 2004.

[43] Kwang In Kim, Keechul Jung, Se Hyun Park, and Hang Joon Kim. Support vector machines for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(11):1542–1550, 2002.

[44] Subhransu Maji and Jitendra Malik. Fast and accurate digit classification. Technical Report UCB/EECS-2009-159, EECS Department, University of California, Berkeley, Nov 2009.

[45] André Stuhlsatz, Jens Lippel, and Thomas Zielke. Discriminative feature extraction with deep neural networks. In *The 2010 International Joint Conference on on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

Declaration of Authorship