UNIVERSITÄT OSNABRÜCK

Institute of Cognitive Science

Bachelor's Thesis

Methods for Analzying the Internal Representations of Convolutional Neural Networks

Jonas Limpert

April 06, 2020

First supervisor: Hristofor Lukanov, M. Sc.
Second supervisor: Prof. Dr. Gordon Pipa

Abstract

Despite their role as one of the most important and interesting research objects of artificial intelligence in the 21st century, the exact mechanisms of their success are still unexplained in many ways. Importantly, the internal processes and representations that enable neural networks to outperform human performance in a variety of tasks suffer from a lack of interpretability. Therefore, neural networks are often described as black boxes. To overcome this black box handling is one of the most important goals to further optimize neural networks and to be up to the moral responsibility that the use of AI in the center of society entails. The understanding of the internal representation is the first step to make the computations in neural networks more transparent. The layers of a convloutional neural network learn a representation hierarchy, which ranges from simple structures in the initial layers to a higher and higher degree of complexity and abstraction, up to high level concepts in the top layers. In this thesis the representation that emerge in the feature hierarchies of two convolutional neural networks are analyzed. Both networks are trained with an identical architecture but different label granularities. Due to the different label positions in the same semantic hierarchy, the networks vary in their representations in terms of abstraction, complexity and translation invariance. The proposed methods show that available tools can help interpret the representations of neural networks and address complex mechanisms of neural information processing.

Contents

1 Introduction **1**

2 Background **3**
2.1 Machine Learning . . . 3
2.2 Artificial Neural Networks . . . 3
2.2.1 Perceptron / Artificial Neuron . . . 3
2.2.2 Multilayer Perceptron . . . 5
2.2.3 Backpropagation . . . 5
2.3 Convolutional Neural Networks . . . 6
2.3.1 Convolutional Layer . . . 7
2.3.2 Pooling Layer . . . 8
2.4 Representations in Convolutional Neural Networks . . . 8
2.4.1 Feature Hierarchy . . . 9
2.4.2 Label Granularity . . . 9
2.5 Analyzing internal Representations . . . 10
2.5.1 Visualizing internal Representations . . . 11
2.5.2 Deconvolution . . . 11

3 Methods **14**
3.1 Approach . . . 14
3.2 Dataset and Models . . . 15
3.3 Methods . . . 15
3.3.1 Visualizing Representations . . . 15
3.3.2 Conditional Layer Norm and Sample Intersections . . . 16
3.3.3 Measuring Translation Invariance . . . 16
3.3.4 Measuring Complexity . . . 16
3.4 Implementation . . . 17
3.4.1 Architecture . . . 17
3.4.2 Analysis . . . 18

4 Results and Discussion **20**
4.1 Results 20
4.2 Discussion 22

5 Conclusion **28**

6 Appendix **29**

List of Figures **37**

1 Introduction

In 2011, the IBM-developed Watson computer[1] beat its human opponent in the successful American show Jeopardy. In 2015, for the first time, an algorithm achieved a better result in a image classification task than a trained human[2] and in 2017, the computer program AlphaGo[3] managed to beat his human counterpart in the game Go, which was long considered extremely difficult to solve due to its intuitive gameplay. Artificial intelligence is in many cases equal to or even far exceeds the capabilities of humans. Artificial intelligences are not only to be found in research or on the big stages, but have found their way into the everyday life of millions of people and influence their everyday life consciously or unconsciously. It is almost impossible to imagine life any more without intelligent systems. Whether it is the use of translation programs [4], expert systems in industry for optimizing production processes[5] or in medicine for disease identification and diagnosis [6], the use of intelligent software delivers breathtaking results in many areas of our daily routine. This development is due in no small part to the success of deep learning models, which form the basis for many intelligent systems and are currently among the most promising areas of machine learning and artificial intelligence.

Despite the immense success of deep learning, its precise workflow is still considered a kind of black box, as the interactions of its individual components are difficult to understand. Neural networks consist of artificial neurons that are organized in layers and detect and extract, similar to most other self-learning machine learning approaches, structures and features in given data sets, which they map into internal representations. Ultimately, artificial neurons perform stereotypical nonlinear transformations of simple matrix multiplications but are able to learn abstract feature representations as part of a complex system.

The limited understanding of the interactions, the origin and the development of these internal representations is one of the main reasons for the lack of interpretability of the internal workflows and the mystification of neural networks as black boxes. A better understanding of these representations is key for the continued success of neural network models.

It is also important to reflect on moral responsibility. It should be the duty

of developers to understand the motives behind their algorithms when moral and ethical decisions are required in sensitive areas. There are a number of incidents where this has failed and, for example, certain ethnic groups have been systematically discriminated against by an AI because of their background[7]. Therefore, interpretability and comprehensibility should play a much greater role in the research and development of neural networks. The understanding of internal feature representations is of fundamental importance, because it allows to get an insight into the highly complex workflows of a neural network.

Thesis Outline In this thesis we investigate the representation of two convolutional neural networks. Due to their special structure convolutional neural networks learn a feature representation hierarchy. The representations in the initial layers are rather simple. These representations increase in complexity from layer to layer, resulting in high-level concepts in the upper layers. The two networks under investigation were trained with an identical architecture but different label granularities. The use of different labels from the same semantic hierarchy results in the emergence of different representations. The goal of this thesis is to identify these differences and to examine the representations of the two convolutional neural networks in terms of their abstractness, complexity and invariance using different analysis methods. First of all, an overview of the general principles of deep learing is given. The focus is on convolutional neural networks and their representations. In the second part the modelling and analysis techniques are explained. Expectations regarding the differences in the representations of the two models resulting from label granularity are defined. In the last part the results of the methods are compared with the expectations and put into a scientific context.

2 Background

In this chapter I will explain the basics of deep learning. The focus is on convolutional neural networks, their representations and the possibilities to analyze them. Special attention is given to a visualization method called 'deconvolution', which is the basis for relevant analyses used in this thesis.

2.1 Machine Learning

Machine Learning is a subfield of artificial intelligence which includes deep learning, but also many other techniques that are used to estimate complicated mathematical functions by learning from data. There are different approaches how a machine learning algorithm handles data, but this thesis focuses on supervised learning [8]. In supervised learning the input data is always a tuple, consisting of an input value x and a corresponding target value t. A supervised learning algorithm approximates a function f*, which describes the relationship between values x and their desired output t. In case of a classifier, f*(x) maps an input x to a class y. The algorithm defines a mapping $y = f(x; \theta)$ and learns during training the best approximation for parameters θ. After training this function can be used to predict the output y for a given, unknown input value x.

2.2 Artificial Neural Networks

2.2.1 Perceptron / Artificial Neuron

The artificial neuron is the most fundamental unit of a deep neural network. The modern approach to artificial neurons is inspired by the considerations of McCulloch and Pitts in 1943 [9], who tried to mimic the functionality of a neuron and formalised the neural process in a simple mathematical model. The so called M-P neuron is a linear threshold function, which generates a binary output for a given binary input based on an adjustable threshold. In

1958, Frank Rosenberg [10] extended the approach and created the Perceptron, a more general model which overcomes a few of the M-P's limitations, can process real-valued input data and introduces a transmission weight, which determines how strongly a neuron is affected by its input. The artificial neuron in modern neural networks works on exactly the same principle, but instead of using a threshold function, a so-called activation function is used [11]. The activation function represents a vector-to-scalar function that takes a weighted real-valued input and computes a non-linear transformation:

$$Y = \sigma(\sum_{i=1}^{n} w_i * x_i + b) \tag{2.1}$$

To obtain the output value Y an activation function σ is applied to the dot product of the input vector x and the weight vector w. Additionally a bias is added which allows to shift the function to the left or to the right. Different activation functions can be used. In order for the artificial neuron to learn (cf. backpropagation, Section 2.2.3), its activation function needs to be differentiable. Furthermore, to ensure learning beyond linear relationships, the activation function needs to be non-linear. One example of a non-linear function is the sigmoidal function.

$$h(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

An advantage of the sigmodial function that it is bound in a certain range, so that the activation won't 'blow up', resulting in increasingly high values or increasingly low values. Furthermore, for X being close to 0.5, Y values

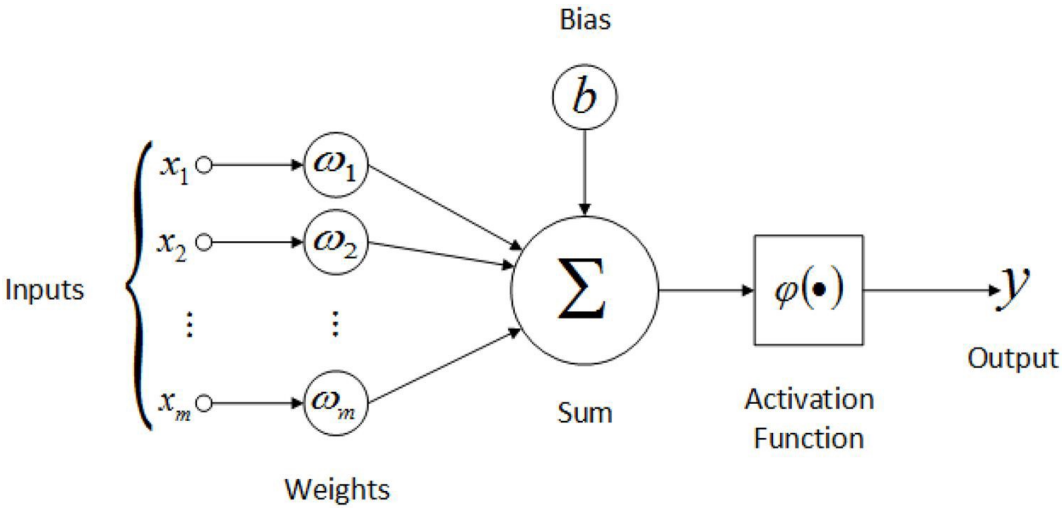


Figure 2.1: Artificial Neuron models and its parts. Taken from Haykin (1994)

are very steep. Small changes in X values results in large changes in Y values, which makes a clear distinction in prediction. However, the disadvantage here is that for very high or very low values of X there are hardly any changes in the activation.

Another example is the rectified linear unit (ReLU) function [12]:

$$h_\theta(x) = max(0, x) \tag{2.3}$$

The ReLU function gives as output the value of x if x is positive and 0 otherwise. Unlike the sigmodial function, ReLU is not bounded, which means that values between 0 and $infinity$ are possible. Moreover, according to the definition of ReLU half of the input domain is mapped to zero, which results in sparse activation in a complex neural network. Because of its linear behavior for the other half of the input domain it needs less computational effort caused by the simpler mathematical calculations. Another advantage of the ReLu function is that the derivative is always zero or one. This means that the gradient is not infinitely large or small, which plays an important role for backpropagation 2.2.3, since it allows to train much deeper nets.

2.2.2 Multilayer Perceptron

Unfortunately, a single artificial neuron can only handle linearly separable data. A multilayer perceptron overcomes this problem by organizing artificial neurons in layers. A multilayer perceptron is composed of an input layer, at least one hidden layer, and an output layer. Each layer except the input layer are called fully-connected, because each neuron in a layer receives a signal through weighted connections from all neurons in the previous layer. Each neuron in the input layer represents one dimension or feature from the input space. The signal is fed into the hidden layers where most of the computation happens. The number of hidden layers and their dimensions can vary according to the kind of problem the network is used for. The output layer receives its signal from the hidden layers and produces an output such as classification or regression. The structure of a multilayer perceptron with its different layers is shown in figure 2.2.

2.2.3 Backpropagation

To use a neural network for approximating a function $f*$ it has to be trained. The training of a neural network consists of two main steps which are repeated until a desired criterion is reached. The first step is called forward propagation, in which a given input signal is processed from input layer to the output layer in a forward manner, resulting in an output $\hat{y}$. The input x is used to compute

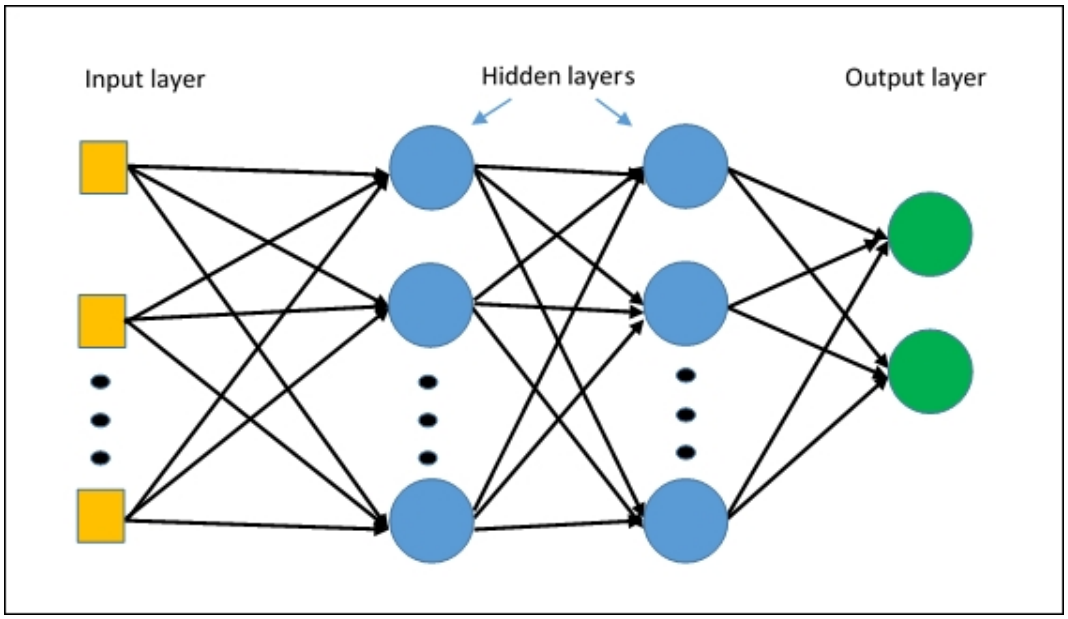


Figure 2.2: A typical multi-layer perceptron. An input is propagated from the input layers through the hidden layers to the output layers, which assign a value to the input. Source: https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow

a response of the first layer through an activation function. The output of the first layer is fed into the second layer and so on until the output layer is reached and the output $\hat{y}$ is calculated.

In the backward step, the so-called backward propagation [13], the weights of each neuron are adjusted according to an error term. The error term compares actual output vector $\hat{y}$ to the target vector t according to a loss function. The gradient of the loss function w.r.t. the weights is calculated and then the weights are updated in backward direction starting from the output layer and continue till input x is reached.

2.3 Convolutional Neural Networks

Convolutional neural networks (ConvNets) are a special kind of a feed-forward neural network for processing a grid-like topology, for example images with RGB channels. The idea of the functionality of CNNs is strongly inspired by the mechanisms of signal processing in the primary visual cortex [14]. In contrast to classical feed-forward networks, convolutional neural networks have at least one layer which performs a special form of linear operation, which is called convolution, instead of matrix multiplication [8].

The processing of signals is almost analogous to traditional neural networks.

Each neuron still receive an input and performs an operation, followed by non-linear function, and also minimizing the loss function will be done in the same way. A classical convolutional neural network consists of blocks of convolutional layers followed by pooling layers. In the convolutional layers the networks learns specific feature representations and the pooling layers are used to reduce the dimensionality of those representations. The flattened output of these convolution-pooling mechanism is fed into a block of fully-connected layers which predicts the label which describes the input the best.

The special combination of convolutional layer and pooling layers adds particular characteristics to the network. The convolutional network is invariant to certain transformation of the input [8]. This is especially true for translation transformations, i.e. the spatial shifting of the input. The network can thus detect whether an object is contained in different images, regardless of its exact position. However, this robustness against translation is not consistent [15]. The initial layers are extremely sensitive to shifts. This sensitivity decreases from layer to layer and results in a high degree of translation invariance in the top layers. Up to a certain degree further transformations like rotation and scaling are possible, but this quickly leads to inaccuracies.

2.3.1 Convolutional Layer

In figure 2.3 an operation of a convolutonal neural network is shown. Instead of using weight vectors like the fully connected layers, weights in convolutional layers are organized in learnable kernels. These kernels are small in width and height but cover the whole depth of the input volume. Importantly, the kernel is shared between neurons within one layer

During the forward pass a kernel moves gradually across the width and the height of the input space and generates a feature map, which represents the dot product between the kernel weights and the actual position in the input space.

The output of a convolutional layer is a three dimensional feature map stack, where the first two dimension represent the width and the height of a specific feature map and the third dimensions comprises the different features.

All units within a feature map share the same parameters such that they are sensitive to the same specific feature. They are connected to different subregions of the input. The area, which is covered by a specific unit, is called its receptive field. This kind of sparse connectivity results in a huge decrease of used weights, since every neuron is just connected to a small amount of neurons in the previous layer compared to the connectivity of a fully connected layer, where every neuron is connected to every neuron of the previous layer.

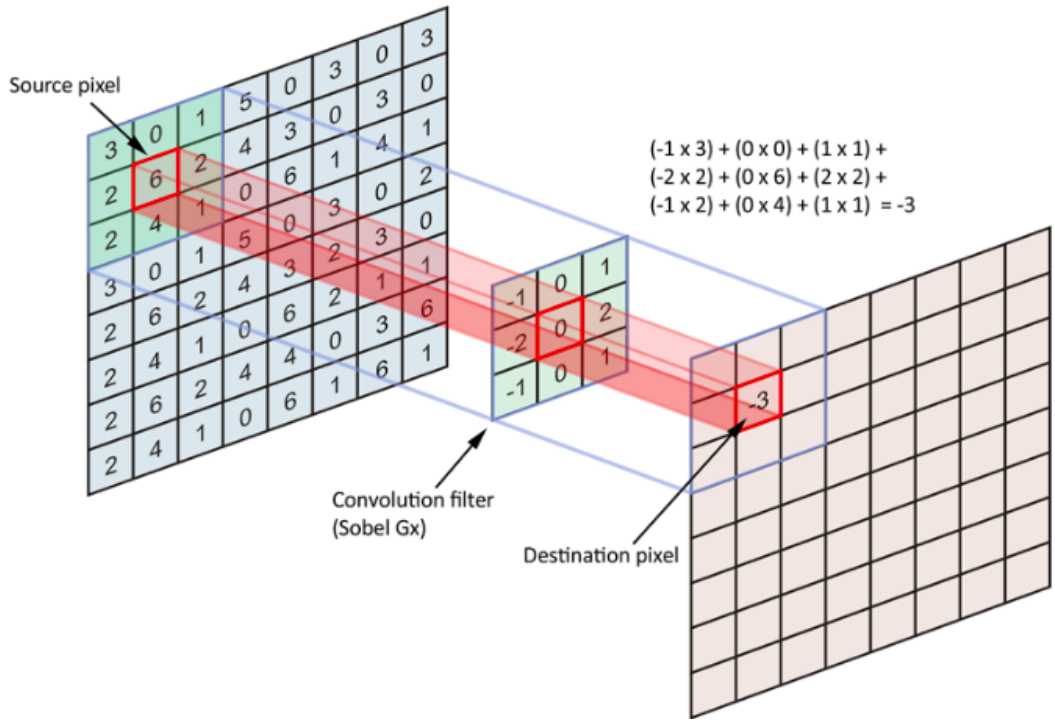


Figure 2.3: An illustration of a 2D convlution. Each filter is applied to each area in the input. Each unit in the resulting feature corresponds to the dot product between the kernel and the area in the input Source: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

2.3.2 Pooling Layer

Pooling is a down-sampling technique which applies a pooling function to the output of the previous layer to reduce the spatial dimensionality of the feature maps. Each feature map of the previous layer corresponds to a feature map in the pooling layer. In the pooling layer, several units of the feature map of the previous layer are combined into one value. Different pooling operations can be applied to scale the dimensionality, but the max-pooling function is most often used, which just takes the max of the rectangular shaped area as summarizing value. An max pooling operations is illustrated in figure 2.4.

2.4 Representations in Convolutional Neural Networks

A convolutional neural network learns to assign inputs to certain categories by recognizing representations of descriptive features. These representations are

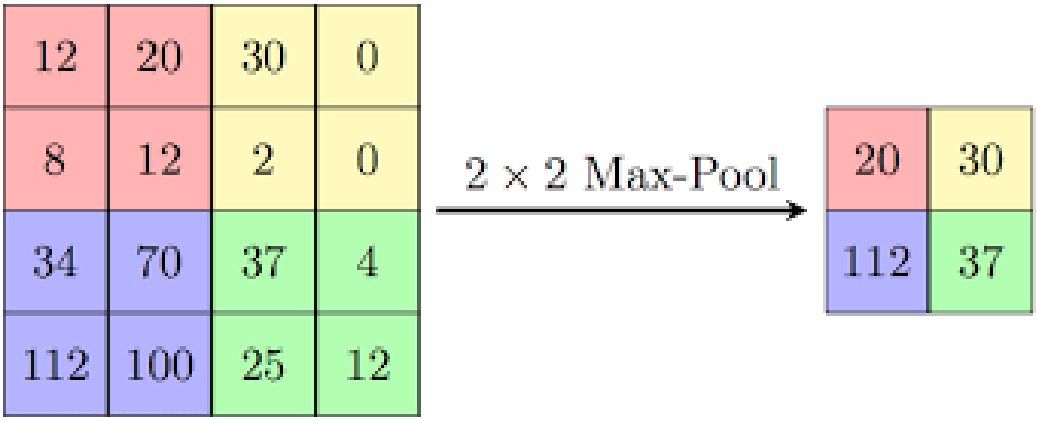


Figure 2.4: An illustration a max pooling operation. All values in an area of 2x2 are summarized by their max value. Source: http://cs231n.github.io/convolutional-networks/

then used by the network to assign a previously unknown input to the correct category.

If the learned filter of a neuron within a layer responds to a certain pattern in a given input, this results in a distinct representation of the pattern in the feature map corresponding to the intensity of the activation.

2.4.1 Feature Hierarchy

Convolutional neural networks learn different levels of representation due to their composition of convolutional and pooling layers. The CNN gradually combines low level features to more abstract concepts to form a feature hierarchy, in which less specific concepts are reused to form more specific ones. Since the filters in the initial layers tend to be more sensitive to simpler stimuli, the corresponding feature maps represent basic patterns like edges and corners. Feature maps in the intermediate layers transform the low level input into more complex representations like parts of objects. All these features are then combined to high-level concepts that are recognized in the deepest layers. Thus the CNN combines sequentially low level features to more abstract concepts to form a feature hierarchy [15]. A visualization of this feature hierarchy can be found in Figure 2.5.

2.4.2 Label Granularity

Applying supervised learning to a convolutional neural network results in a learned mapping between an input x and a target label y. However, there

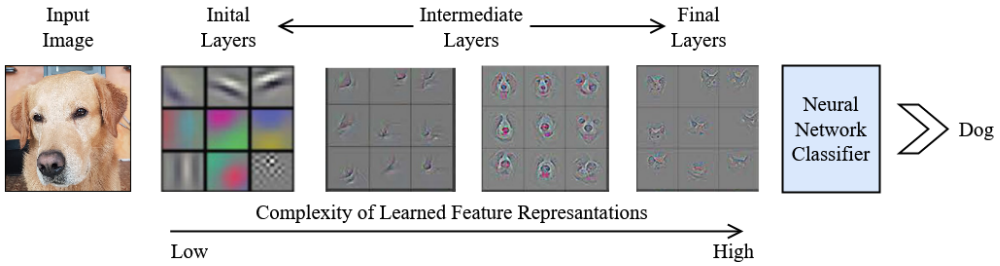


Figure 2.5: A demonstration of the feature Hierarchy in convolutional neural networks. The visualized filters are taken from the paper 'Visualising and Understanding Convolutional Neural Networks' by Zeiler et.al.

is no exact or correct label y for an entity x. Since the correct description always results from the context and, depending on the circumstances, a detailed description is sometimes better than a less detailed one or vice versa. In the context of deep learning the use of class labels from different positions in the semantic hierarchy is called 'Label Granularity' [16]. Labels that have a higher level of detail are called fine-grain labels and labels that are less detailed are called coarse-grain labels. The use of labels of varying degrees of detail also results in different developments of feature representations. By using fine-grain labels, the network is forced to distinguish between a higher number of semantically similar classes. This results in a higher number of learned high-level features in the upper convolutional layers, which are responsible for the recognition and classification of high-level concepts [16].

2.5 Analyzing internal Representations

Since the general focus in the analysis of CNNs in the field of image processing is mainly concerned with visualization techniques, the main focus of this thesis is the description of methods that analyze feature representations using visualization techniques. In addition to visualization methods, there are some other numerical methods for the analysis of internal feature representations and the interaction between different feature maps. One example is the approach of Hao Xu et al. [17], which computes the average precision of an individual filter and uses it as a quantitative measurement method to determine the discriminative capability of a trained CNN filter.

2.5.1 Visualizing internal Representations

Network visualization offers a simple and efficient way to improve the network interpretability. For this the internal representations are transformed into a visually recognizable format. While today modern approaches can make the entire workflow of a convolutional neural network visible, until a few years ago it was only possible to interpret the first layers [18]. In the following, the most common visualization methods will be briefly introduced, with a focus on the deconvolutional network (DeconvNet), which will be used in the further course of this work as a useful tool for visualization. The following descriptions are taken from the paper 'How convolutional Neural Networks see the world' by Zhuwei Qin et al.

Activation Maximization visualizes the preferred input pattern of a neuron by an artificially created input that maximizes the activation of the neuron. For this purpose, the original input is iteratively modified until the activation of the neuron is maximal. The resulting reconstruction then provided the basic characteristics of the learned representation of the neuron.

Network Inversion describes the representation from a layer level perspective. It does not represent the activation of a single neuron, but the coherent feature set of all neurons in a layer.

Network dissection describes the representation of a neuron by semantic concepts such as color, texture, parts, objects and scenes. It is investigated how neurons correlate with semantic concepts by systematically examining the neurons for their sensitivity to semantically specific elements in the input.

2.5.2 Deconvolution

The following description is mainly based on the paper 'Understanding and visualizing Convolutional Neural Networks' by Zeiler et al. published in 2012 [15].

The Deconvolutional Network, which we will call DeconvNet from now on, reveals descriptive patterns of the input image that activate the neuron to be examined. The revealed representation is projected back into image space in top-down manner by reversing the operations of the convolutional and pooling layers of the examined convolutional neural network [18]. The DeconvNet provides the same structure and the same components as the ConvNet, but for but while the Convent maps pixels to features, the DeconvNet does the opposite [15].

The visualization process can be divided into two parts. In the forward pass an image is presented to the convolutional network and the internal representations throughout all layers are calculated w.r.t the input. To examine the activity of a specific neuron, the activation of all other neurons in this layer is set to zero. In the backward pass the activity of the neuron is mapped back to input space. The feature map is presented to the DeconvNet, which reverses all the manipulations of the ConvNet layer by layer until the input space is reached.

While a ConvNet consists of a sequence of convolution, rectification and pooling operations, the DeconvNet is a series of transposed convolution and unpooling operations.

Transposed Convolution

The transposed convolution, also known as deconvolution, is the 'reversed filtering' in the DeconvNet. While the filtering in ConvNet uses learnable kernels to generate feature maps from a given input, reverse filtering means reconstructing the original input from these feature maps [19].

Unpooling

Due to the nature of a pooling layer its calculation cannot be inverted, but to reconstruct a representation nevertheless, the inversion is approximated. For this purpose, the location of the maximum in each pooling region is stored in a switch variable during the forward step. This switch variable is used by the DeconvNet to place the reconstruction of the overlying layer at the correct position.

Rectification

Just like in ConvNet, ReLU is used as activation function in DeconvNet. The ReLU function guarantees that the feature map resulting from the output of a neuron in a ConvNet is always positive. Therefore, the unpooled feature map is rectified during reconstruction to verify that these feature maps are also positive.

Workflow

Figure 2.6 shows the workflow and architecture of a DeconvNet. The output of the previous layer or the input data is passed to the convolution layer. A rectification function is applied to the resulting feature maps and these serve as input data for the pooling layer. During the pooling operation the location

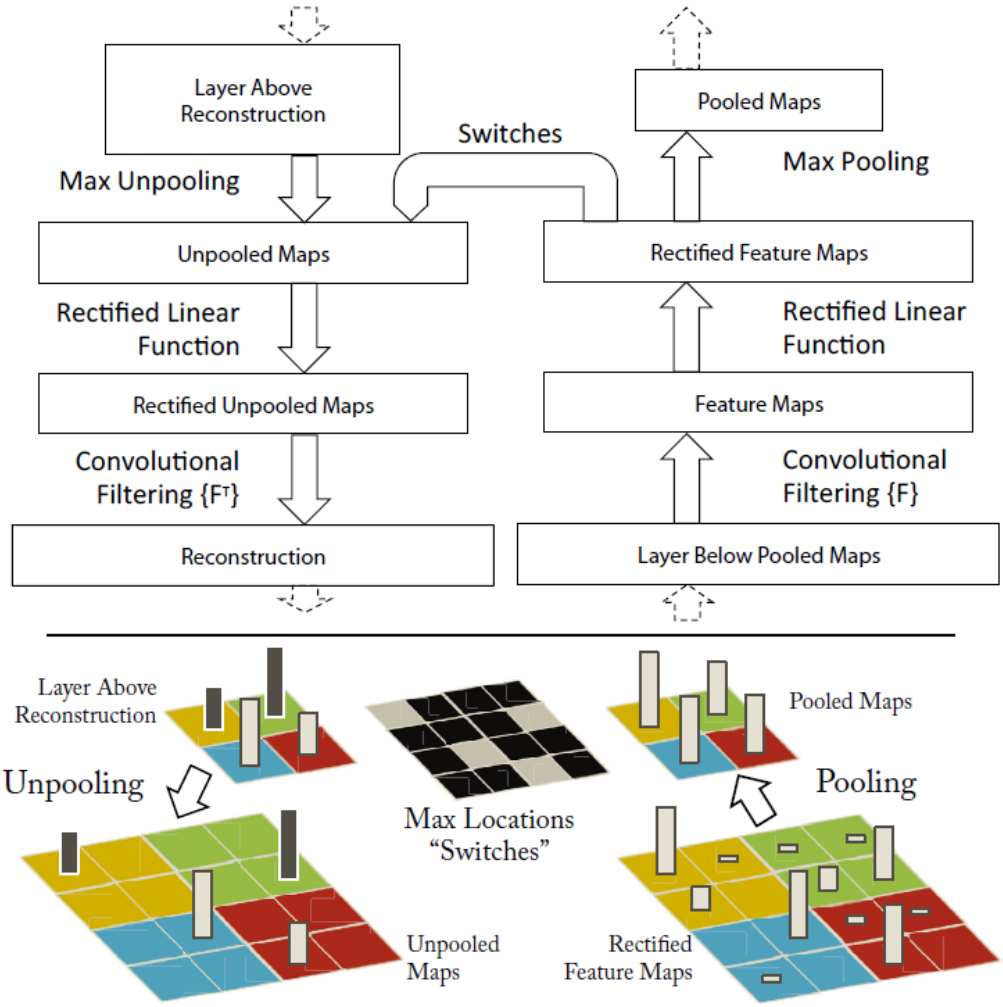


Figure 2.6: DeconvNet Architecture. The illustration is taken from the paper 'Visualising and Understanding Convolutional Neural Networks' by Zeiler et. al.

of the maximum value is stored, which is used in the unpooling method to reconstruct the feature maps. The unpooled feature maps are rectified and serve as input for the transposed filtering.

3 Methods

3.1 Approach

In this bachelor thesis, we apply different methods to analyze the internal representations of convolutional neural networks. This is done to detect differences in the representations of two networks, which are caused by using class labels from different positions of the same semantic hierarchy 2.4. Following the section Label Granuality 2.4.2, it is assumed that the use of finer labels, i.e. the use of more classes between which to distinguish, also favors the emergence of a higher number of finer representations. The use of coarser labels would therefore favour the creation of fewer and less complex representations. Since representations in convolutional neural networks are hierarchically organized, it can be assumed that the networks to be compared develop similar, simply structured representations in the lower layer, which, however, become more and more different in their complexity, abstractness and translation invariance the further one ascends in the hierarchy.

In the following we suppose that the representations of a finer-grained classifier develop more complex and abstract structures compared to the representations of a coarser-grained classifier. However, it is assumed that the representations of a coarse-grained classifier are more robust to spatial shiftings. A coarser labeling results in fewer and more distinct high-level representations. The translation-induced limitation of the perception of the input can be more easily assigned due to the clearer conceptual distinctions of the representations.

The goal of this work is to analyse the representations of a fine-grained and a coarse-grained classifier in order to show the differences in their feature hierarchies and compare them with respect to their complexity, abstractness and invariance. The meaning of these concepts in the context of my work is defined more precisely in the results.

3.2 Dataset and Models

For our experiment, we used the Cifar100 dataset[20], one of the most common datasets in the field of image processing and classification. The Cifar 100 data set contains 60000 color images with a dimension of 32x32x3 representing height, width and the RBG color channels. The images are divided into 50.000 training images and 10.000 test images. The Dataset is a subset of the 80 million tiny images dataset [21]. Unlike many other datasets, the samples have a semantic hierarchy that categorizes the images into sub and super classes. Accordingly, each sample has a 'coarse-grained' label coming with the sub class and a 'fine-grained' label coming with the super class. Each of the sub class contains 600 samples and each super class contains 3000 samples. An overview of the different classes and super classes can be found in the appendix. Two different models using the same architecture are trained with the same samples but different labels. The training details and the exact architecture will be described in detail in the implementation chapter. In the further course of this work the following classifiers will be examined:

- Fine-Grained Classifier (FGC): trained on sub classes
- Coarse-Grained Classifier (CGC): trained on super classes

3.3 Methods

To analyze the internal feature representations of the networks, the first step is to visualize the representations. To investigate the representations in the different networks, we first examine the individual layers for their sensitivity to the test set, as well as for the sensitivity to specific samples. Afterwards we will check if the networks have developed a recognizable and distinguishable robustness against translations and if the complexity of the representation can be distinguished by the different labeling.

3.3.1 Visualizing Representations

To visualize the representations of the network we use a deconvolutional Network. Since the DeconvNet only reconstructs the representation of an activated neuron in dependence of a given input, first suitable input images have to be selected. For this purpose, the five images that activate a feature map the most are selected from the Cifar100 test data set for each feature map individually. Then the representations of the top five activation samples are reconstructed by the DeconvNet.

3.3.2 Conditional Layer Norm and Sample Intersections

To investigate the representations in the different networks, we first examine the individual layers for their sensitivity to the Test Set. We define the the conditional layer norm (CLN) as the mean activation of all feature maps caused by their Top-One samples. Furthermore, it is examined to which sample the layer in a network react particularly sensitively. For that, the frequency of occurrence of the individual samples as a Top-one activation are counted for each layer. Ths will be used to examine in how far layers of both networks tend to be activated by the same samples frequently and whether the intersection decrease according to the label granularity as you move up in the feature hierarchy. An Intersection is a sample that is among the 10 most frequently occurring samples for the same layer of both networks. If both layers have many Intersections, this can be taken as an indicator for similar representations.

3.3.3 Measuring Translation Invariance

To determine the translation invariance of a network layer, the aproach by Marcel A. J. van Gerven et. al. [22] was used. First, a two-dimensional response surface is generated for each neuron in the layer under investigation. This response surface is a two-dimensional array that stores the activations of the neuron arising from the systematic spatial shift of the input. The value at the corresponding position in the activation array is determined by the resulting activation, with the activation of the initial input acting as the central point. Therefore, the value at position (0,5) corresponds to the activation of the neuron under consideration in response to the original input shifted by 5 pixels towards the top. As input the Top-One activation sample is used. In the second step, a two-dimensional Gaussian surface is fitted to the response surface. The layer invariance then corresponds to the mean value of the full-width at half-maximum (FWHM) of all two-dimensional Gaussian surfaces of the neurons belonging to that layer. Intuitively, a smaller FWHM value corresponds to a a narrower Gaussian, which in turn means that the activation of the neuron drops faster towards zero for shifted inputs.

3.3.4 Measuring Complexity

The DeconvNet generates the internal representation of a neuron of a given input that activates that neuron. To describe and quantify the information content of this representation, the so-called Kolmogrov complexity can be used[22]. The Kolmogrov complexity is defined as the shortest complete description of a pattern that can be reconstructed by an algorithm. However, since at no time all possible descriptions are known, it can never be assumed that the description

used is the shortest. Therefore, the Kolmogrov complexity of a pattern can only be approximated. Less complex patterns have a lower information content and show more a regular structure, which results in a shorter description. More abstract patterns, on the other hand, are more irregular, producing a more complex descriptor. A common method to estimate the Kolmorov complexity of image files is Zip compression, a lossless method for archiving data. Thus, to describe the complexity of the representation of a neuron, the reconstructed representation is compressed using Zip compression and the complexity is measured by the size of the compressed file [23].

3.4 Implementation

For the Implementation we mainly used Keras[24], an Extension of Tensorflow[25]. Other libraries that were frequently used are NumPy, SciPy and PIL.

3.4.1 Architecture

The network architecture is the same for both networks, except for the configuration of the output layer. During the training process, it was found that for the data set used, a flatter architecture achieves significantly better results than a deeper architecture. The first part of the network consists of three blocks, each composed of two successive convolutional layers followed by a max-pooling layer. The number of channels is systematically increased from layer to layer, starting with 32 channels in the first up to 512 channels in the last convolutional layer. To regularize the network during training, the output of the pooling layer is fed into a dropout layer [26] before further processing. The convolutional layers are followed by two fully-connected layers. The first has 1024 neurons, the neurons of the second fully-connected layer depend on the type of classifier and correspond to the number of class labels. Figure 3.1 graphically illustrates the architechtures.

All networks use categorical crossentropy [27] as loss function and Adam [28] as optimizer. However, different learning rates were used for the training. Since the networks tend to overfit, early stopping is used for regularization. The training data is divided into 40.000 training samples and 10.000 validation samples. The validation set is used to check after each epoch how well the model generalizes to unseen data. The training is stopped when the error of the training set starts to deviate from the error of the validation set. Furthermore, the data set is artificially extended using image augmentation. Image augmentation is a technique in which the images are subjected to certain transformations, such as rotations, shifts and scaling, before the training session, thus increasing

Hyperparameter	Fine-Grained Classifier	Coarse-Grained Classifier
Loss	Categorical Crossentropy	Categorical Crossentropy
Learning Rate	1.e-4	1.e-4
Batch Size	64	64
Optimizer	Adam	Adam
Weight Decay	1.e-4	1.e-4
Dropout	0.3	0.3
Trained Epochs	45	28
Test Accuracy	0.624	0.727
Top-5 Accuracy	0.864	0.944

Table 3.1: Hyperparameters and accuracies for fine- and coarse-grained classifier respectively.

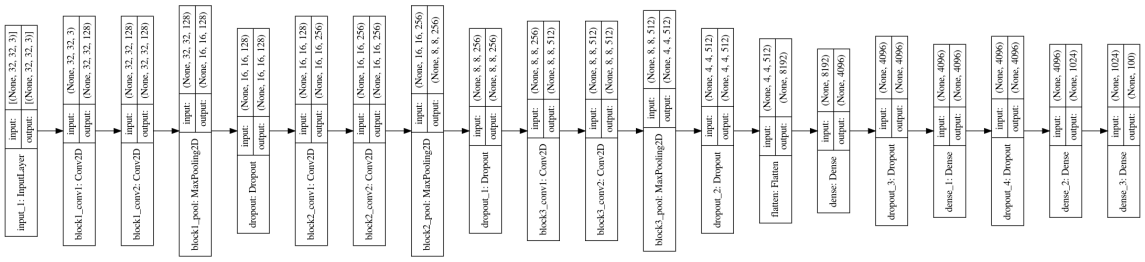
the amount of training data. A seed was placed to train the networks under the same conditions and to allow the experiments to be reconstructed All hyperparameters used for the training are summarized in the Table 1.2

3.4.2 Analysis

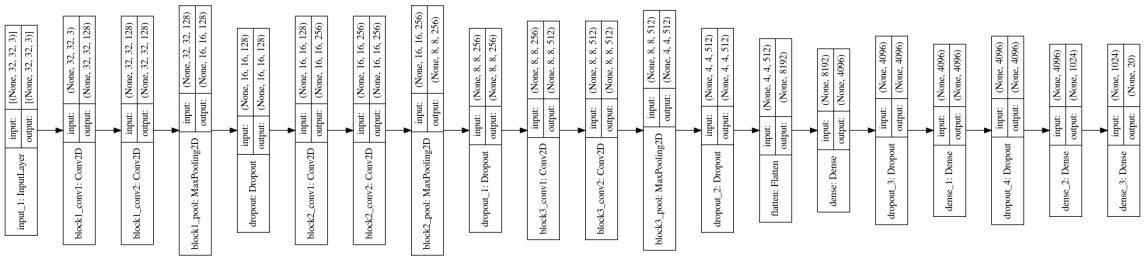
The implementation of the analysis corresponds for the most part to the methods in the previous section. Some processes concerning the analysis of the layer complexity as well as the translation invariance had to be adapted. Those changes are documented in the following section. The implementation of DeconvNet is based on a realization by Liang Jiang[29].

Translation Invariance To construct the response surface, the representation must be shifted step by step. One step corresponds to a spatial shift of one pixel in a predefined direction. The input is shifted along the main diagonal and the cardinal axes and the maximum number of steps is set to fifteen, due to the small size of the input. To manipulate the input images accordingly, the shift function from scipy.ndimage is used. Since shifting leaves a gap that can lead to miscalculation, the pixels in the gap are filled with zero.

Kolmogrov Complexity Following Section 3.3.4 the Kolmogrov Complexity of a feature map is approximated by its compressed file size. We describe the complexity of a layer by the mean of the size of its compressed representations.



(a) Architecture of the Fine-Grained Classifier



(b) Architecture of the Coarse-Grained Classifier

Figure 3.1: Architectures of the used classifers. Identical for both, except for the classification layers

Since we select the five maximum activated neurons for each feature map, but some feature maps do not respond to any input, some reconstructions may contain empty representations. These dead neurons, i.e. neurons that have no representation, are not considered in our calculation
.

4 Results and Discussion

In this chapter I describe the results of my experiments. In addition to the representation examples below, further examples can be found in the appendix.

4.1 Results

The figure 6.1 illustrates the representations of the fine-grained classifier. The reconstructions of the representations clearly show how the classifier develops ever more specific hierarchies of characteristics. The focus of the initial layers on rather coarse structures like corners and edges is also clearly visible. The receptive field of the reconstructed representations just covers a small area of the image, which size corresponds to the initialised kernel size. The deeper you go into the network, the more complex the structures become. In the second block more specific features become visible and the receptive field size covers a larger area of the input sample. Although the reconstruction shows the basic structures of the representations, it is not possible to show the object representation in its entirety. Especially the reconstructions in the upper layers appear blurred and rather unclear.

Figure 4.2 shows the intersections of samples that appear as a Top-One sample in a specific layer of both classifier. The distribution is quite similar for the first layers. Samples that often achieve a strong activation in lower layers of the FGC also tend to obtain similar results in the lower layers CGC. But with increasing depth it becomes much less likely that these samples match. While in the first layer nine of the ten Top-One samples from the FGC are still to be found among the ten most frequent samples of CGC, in the deepest layer only three of ten samples are found that match both networks. The diagrams showing the distribution of samples for each classifier can be found in the appendix.

In Figure 4.3 one can see the plotted conditional layer norm. The conditional layer norm corresponds to the mean of all activations of all neurons in a layer

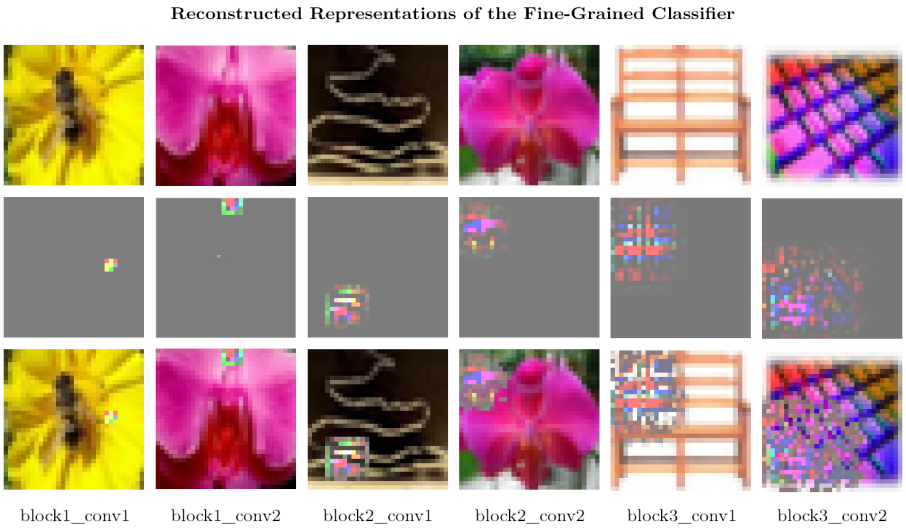


Figure 4.1: Shows the representations of the fine-grained classifier. The representations increase in complexity from layer to layer according to the feature hierarchy.

depending on the test set. For both networks the conditional layer norm of the second convolutional layers in the first two blocks show a considerable higher activation than the proceding layer. In the last block of both networks the first convolutional layer shows a stronger activation than the following layer. Furthermore, the conditional layer norm of the last layer of the FGC shows a considerably lower activation than the CGC.

Figure 4.4 shows the degree of the translation invariance. As explained in the chapter, the invariance is determined by the mean of the full width at half maximum of the 2D Gauss distribution plotted to the response map of all neurons in each layer. The layers of the first blocks of both networks are susceptible to translations, while the CGC is slightly more robust. However, the susceptibility of both classifiers decreases the further one moves on in the hierarchical structure, whereby the CGC also achieves better results in the first layer of each block. It is noticeable that the degree of robustness of the CGC decreases within each block.

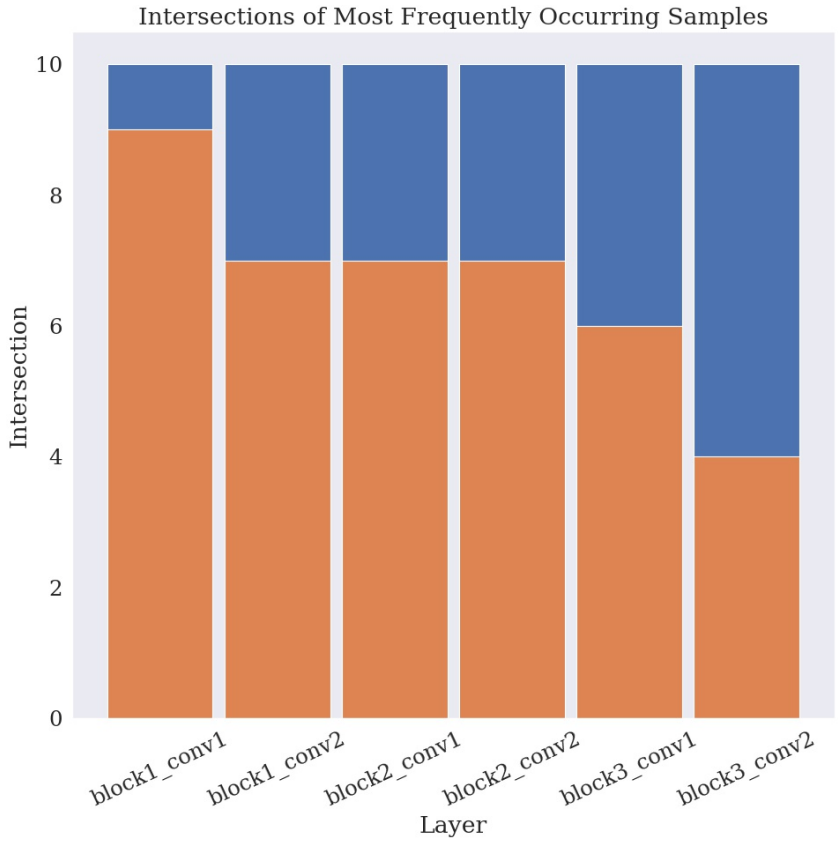
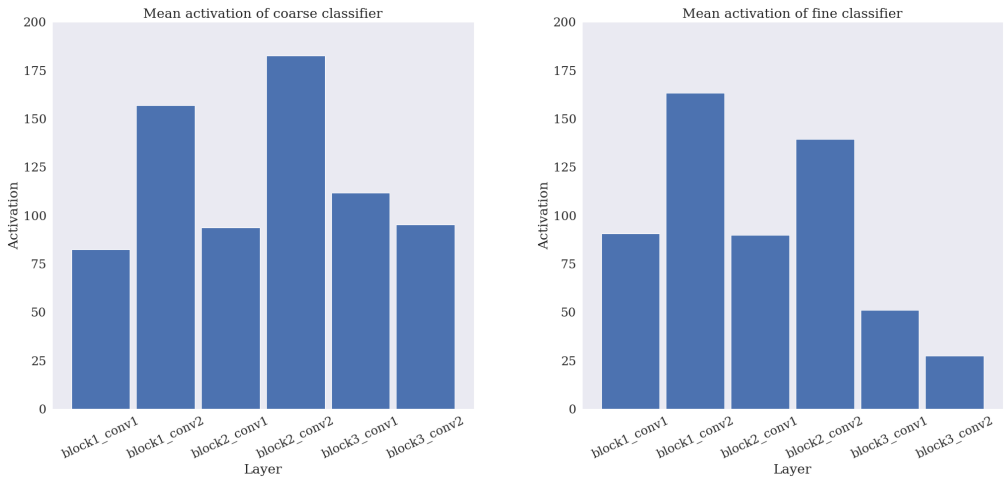


Figure 4.2: Shows the intersection of the most frequently occurring Top-One samples in both classifiers. The blue color indicates the number intersections, the red color corresponds to amount of most frequently Top-One samples that were different.

In Figure 4.5 the Kolmogrov complexities of the convolutional layers are plotted.While the lower layers with their low-level detectors tend to be less complex, it can be clearly seen that the complexity of the representations increases with increasing depth and reaches its highest value in the top layers. For the first two layers the complexity values are quite similar for both classifier, but from the third layer on the FGC develops more complex representations.

4.2 Discussion

The results of the experiments clearly show that label granuality has a significant influence on the emergence and development of internal representations and that these differences can be clearly demonstrated with the methods used.



(a) Coarse-Grained Classifier **(b)** Fine-Grained Classifier

Figure 4.3: Shows the Conditional Layer Norm. Each bar in the chart corresponds to one layer.

Sample Intersections The Intersections of the most common Top-One samples, shows clearly how the initial layers of both classifiers tend to respond to the same inputs to an equal extent, which can be attributed to the development of similar representations. In Agreement with the assumption that the representations differ more and more with increasing complexity, the number of intersections decreases as one progress in the feature hierarchy . The small amount of intersections in the top layers is due to the development of high level feature representation, which, as a result of its label granularity, has developed either finer concept representations due to finer-grained labelling or less complex high level features due to coarser-grained labelling.

Kolmogrov Complexity The fact that the use of different granular labels leads to the development of representations of varying complexity is clearly demonstrated by the approximation of the Kolmogrov complexity. And here again it can be seen that both classifiers tend to develop similar representations with a similar degree of complexity in the initial layers. As expected the FGC shows a considerable higher complexity in the intermediate and top layers.

Translation Invariance Comparing the invariance behaviour of the two classifiers, the considerable more robust handling of the CGF with spatial shifts becomes apparent. The reason for the low vulnerability of the CGC is the nature of its learned representations. This principle is best explained with an

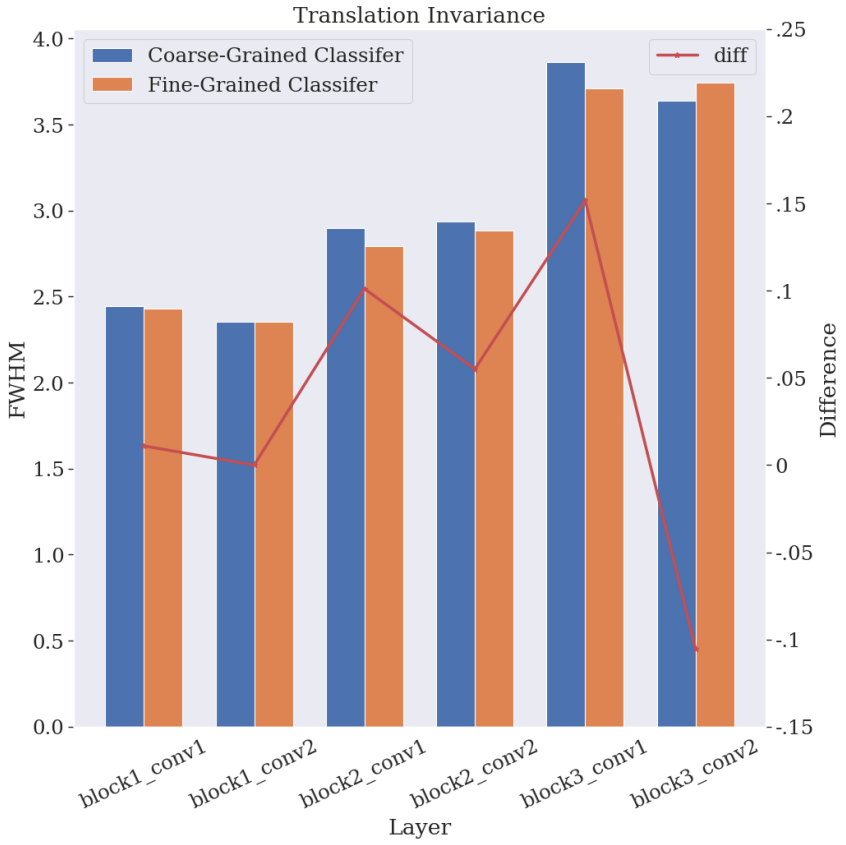


Figure 4.4: Degree of translation invariance determined by the mean of the FWHM of a 2D Gaussian plotted to the response surfaces of neurons within a layer. The figure has two axes. The left axis shows FWHM and the right shows the difference between the degree of invariance of the two classifier indicated by the red line.

example. We have a FGC that can distinguish between different dog and cat species and a CGC that only distinguishes between dogs and cats. The CGC assigns the label dog to each dog, whereas the FGC has to distinguish between Poodles and Dalmatians etc. The translation leads to a limited perception of the neurons responsible for the specific features. This limitation made it significantly more difficult to determine whether an input is a Dalmatian, a poodle or a cat species than to distinguish between cat and dog. While the use of a fine grain label forces the classifier to distinguish between many similarly designed features, the learned features of a classifier using a coarser grain label are more clearly distinguishable.

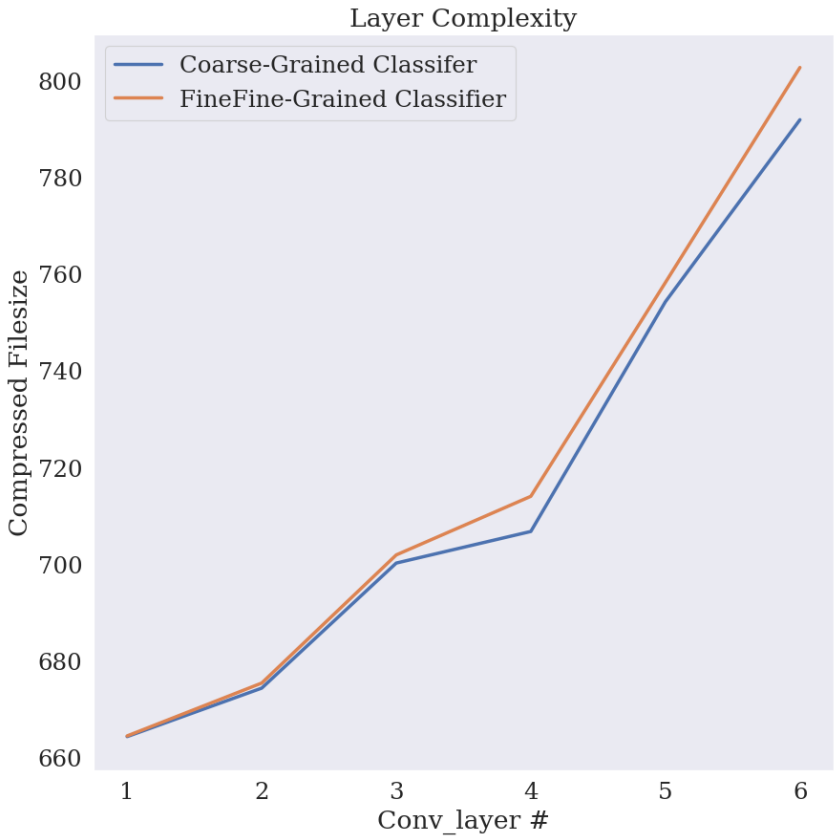


Figure 4.5: Layer complexity of the fine-grain classifier (orange) and the coarse-grain classifier(blue) determined mean of the compressed file size

Conditional Layer Norm The Conditional Layer Norm describes the mean of the activation of the individual layers depending on the test set. The first four layers provide similar results for both classifiers, but strong differences can be observed in the top layers responsible for high level concepts. The CGC is activated considerable stronger by the samples of the test set. In order to explain this difference, I would like to take up the previously mentioned example again. The fine-grained dog classifier learns to assign the label Dalmatian to the Dalmatians, the label Poodle to the Poodles, etc. Resulting in high activation in the more specific representations responsible for the specific concept of a certain dog species. The CGC, on the other hand, learns to assign the label dog to all these dogs species, resulting in a larger number of more general and more abstract feature representations. If a picture of a Dalmatian is classified with the FGC, only feature representations that correspond to the specific concept of Dalamtian are activated. Classifying the same image with a CGC results in less activation for individual concepts, but a wider range of general concepts

is addressed. This activation of general features in the CGC, in its entirety, exceeds the activation of the more specific concepts of the FGC and results in a demonstrably higher mean activation of neurons in top layers of the CGC depending on the used test set.

Implications The results of my experiments show, that the use of different label granularities exerts influence on the formation of representations in terms of their complexity, abstraction and translation invariance. While the representations do not vary noticeably in the initial layers, the differences in representation increase from layer to layer, which can be clearly be demonstrated by the frequently occurring sample intersections. The approximation of the Kolmogrov complexity indicates explicitly that the FGC develops representations with a higher degree of complexity due to more specific feature representations. The development of a larger number of more general feature concepts in the top layers of the CGC, which due to their abstractness respond in larger numbers to the same input, can be proven by the higher values of the conditional layer norm in the top layers of the CGC compared to the FGC. This is also shown by the CGC's more robust handling of spatial translations, whose layers show a higher degree of translation invariance due to the more abstract and more general feature representations.

Limitations The use of DeconvNet in this work has caused some problems, because the reconstructions are sometimes a bit blurred and details are difficult to recognize. This may be due to the size of the samples in the data set, which are rather coarsely structured with a dimension of 32x32 pixels and therefore do not develop any representations that can be reconstructed with Deconvnet in such a way that entire objects or scenes can be recognized. Another problem could be the architecture of the networks. Since the dataset with 50000 training samples is relatively small and the classifiers tended to overfit quickly, the architecture was deliberately chosen small to achieve the best possible test accuracy. This architectural flattening ensures that the top layer representations are less complex than they would have been with deeper model architectures. Nevertheless, the reconstructions are completely sufficient for the methods used.

Outlook We have shown that the use of labels from different label granulites, leads to the fact that the resulting representations show noticeable differences. Since the network architectures used are rather simple constructs, it is necessary to embed the question applied in this thesis in a more complex context. An interesting application would be to examine the representations of a network architecture that may contain several label granularities from the same semantic hierarchy. This could then be compared to the network architectures examined

in this thesis. Furthermore, it would be interesting to extend the scope of the invariance study to include not only translation invariance but also the influence of other transformations such as rotation, scaling etc. In addition, the analysis of internal representations is not limited to the analysis of convolutional neural networks. Other types of neural networks as well as the multitude of other machine learning models could and should be examined for their characteristics of their internal representations.

5 Conclusion

In this thesis the internal feature representations of two convolutional neural networks were visualized, analyzed and compared. The networks have an identical architecture, but were trained with different label granularities. The use of labels from different positions of the same semantic label hierarchy resulted in different internal feature representations. These representations were visualized using a deconvolutional network and these visualized reconstructions were examined for their translation invariance and complexity. Furthermore, two methods were applied to investigate the representation depending on the data set or certain samples from the data set (Conditional Layer Norm). The results show clearly how the representation differs in complexity, abstractness and invariance. A finer label granularity allows the representations to assume more complex and finer structures. The use of coarser label granularity leads to more general and abstract representations, and the property of creating more general concepts also results in a higher degree of translation invariance. To conclude, we have shown that the proposed methods are suitable to shed light on the mechanisms of neural networks and address the role of label granularity for the formation of neural representations. As such, this work provides a basis for further research in the field of interpretable AI and paves the way for better neural network models.

6 Appendix

Super class	Sub classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flower	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household	furniture bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table 6.1: Subclasses and superclasses of Cifar100. Five subclasses form one superclass.

Technique	Parameter
Rotation	20
Width Shift	0.2
Height Shift	0.2
Horizontal Flip	True

Table 6.2: Transformations used for image augmentation.

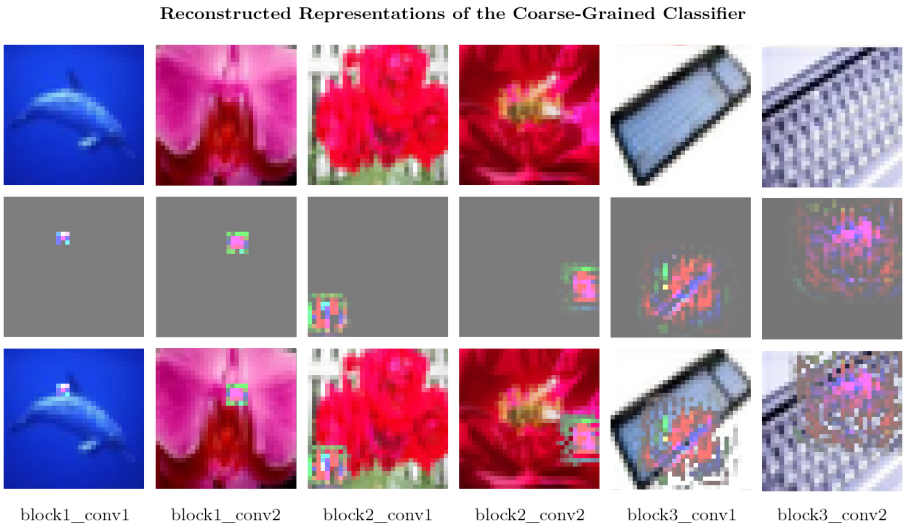


Figure 6.1: Shows the representations of the coarse-grained classifier. The representations increase in complexity from layer to layer according to the feature hierarchy.

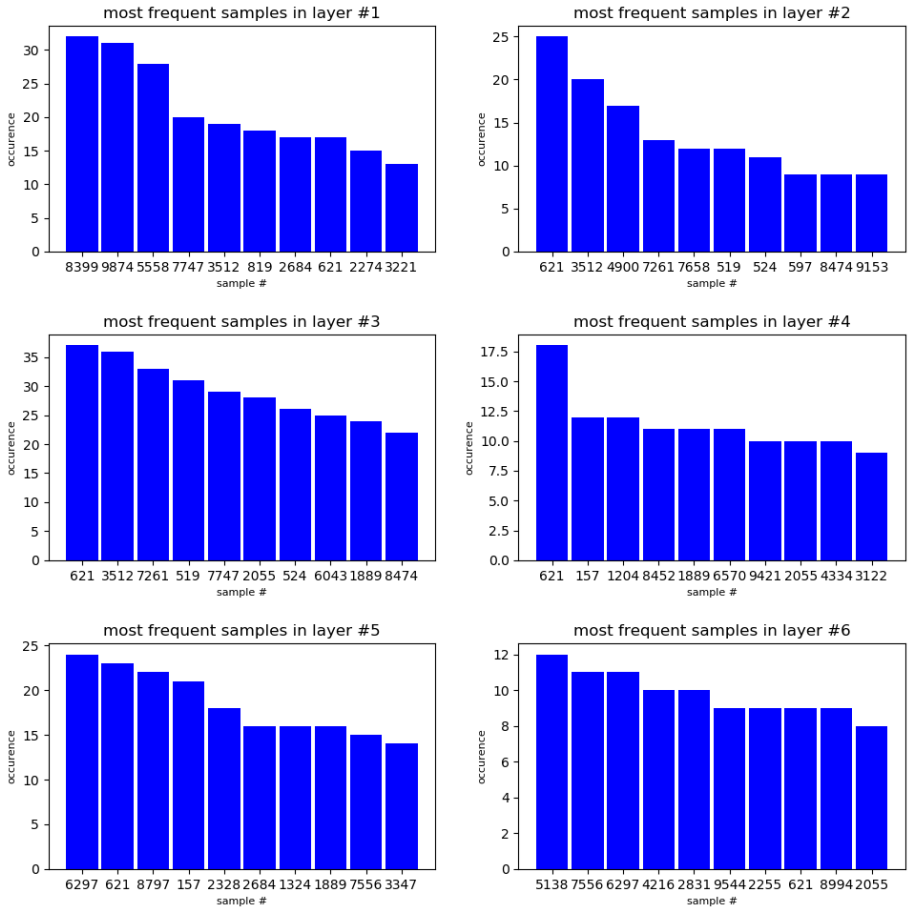


Figure 6.2: Shows the most frequent occurring samples for the fine-grained classifier. These values were used to calculate the Sample Intersections.

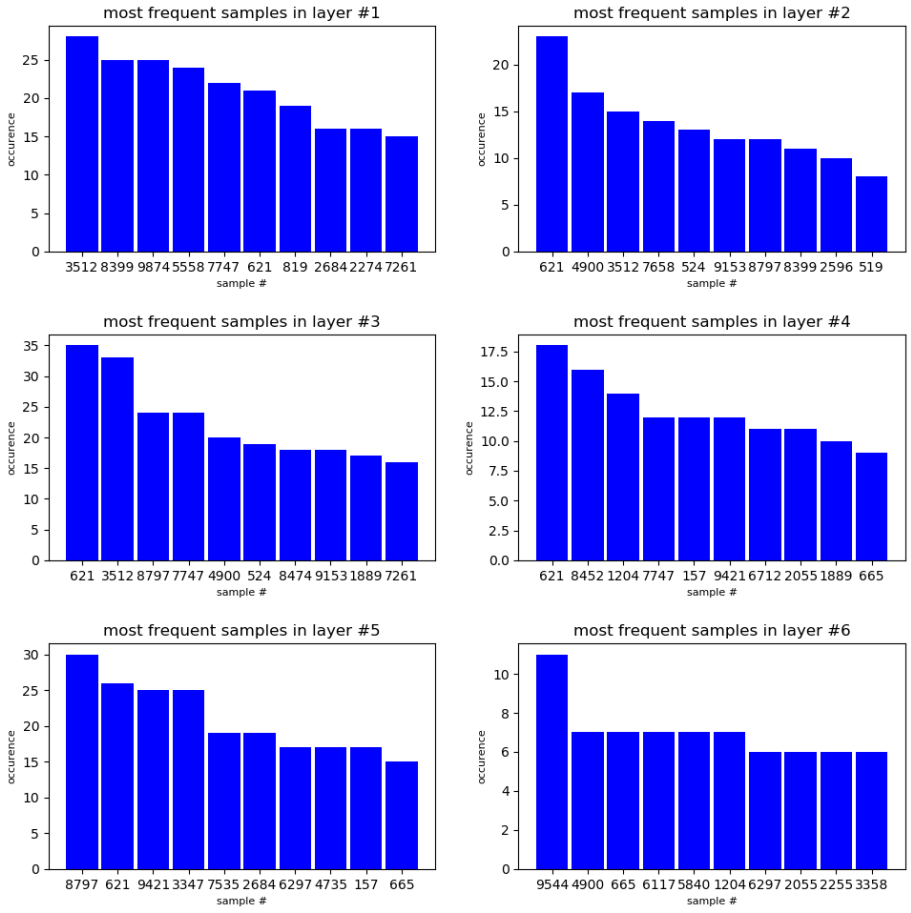


Figure 6.3: Shows the most frequent occurring samples for the coarse-grained classifier. These values were used to calculate the Sample Intersections.

Bibliography

[1] F. A, "Introduction to "This is Watson"," *IBM Journal of Research and Development*, May 2012. PUB749 Riverton, NJ, USA.

[2] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep Image: Scaling up Image Recognition," *arXiv:1501.02876 [cs]*, July 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[4] "Siri." https://www.apple.com/de/siri/. Library Catalog: www.apple.com.

[5] "Artificial intelligence in industry." https://new.siemens.com/global/en/company/stories/inc
in-industries.html. Library Catalog: new.siemens.com.

[6] S. Dilsizian and E. Siegel, "Artificial intelligence in medicine and cardiac imaging: Harnessing big data and advanced computing to provide personalized medical diagnosis and treatment," *Current cardiology reports*, vol. 16, p. 441, 01 2014.

[7] A. Todolí-Signes, "Algorithms, artificial intelligence and automated decisions concerning workers and the risks of discrimination: the necessary collective governance of data protection," *Transfer: European Review of Labour and Research*, vol. 25, no. 4, pp. 465–481, 2019.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] W. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 52, no. 1-2, pp. 99–115, 1990.

[10] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[11] K. Suzuki, ed., *Artificial Neural Networks - Methodological Advances and Biomedical Applications*. InTech, Apr. 2011.

[12] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," p. 8.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[14] G. W. Lindsay, "Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future," *Journal of Cognitive Neuroscience*, pp. 1–15, Feb. 2020.

[15] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013.

[16] Z. Chen, R. Ding, T.-W. Chin, and D. Marculescu, "Understanding the Impact of Label Granularity on CNN-based Image Classification," *arXiv:1901.07012 [cs]*, Jan. 2019.

[17] H. Xu, Y. Chen, R. Lin, and C.-C. J. Kuo, "Understanding CNN via deep features analysis," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1052–1060, Dec. 2017.

[18] Z. Qin, F. Yu, C. Liu, and X. Chen, "How convolutional neural network see the world - A survey of convolutional neural network visualization methods," *arXiv:1804.11191 [cs]*, May 2018.

[19] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv:1603.07285 [cs, stat]*, Jan. 2018.

[20] "CIFAR-10 and CIFAR-100 datasets." https://www.cs.toronto.edu/~kriz/cifar.html.

[21] A. Torralba, R. Fergus, and W. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1958–1970, Nov. 2008.

[22] Umut and M. A. J. van Gerven, "Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream," *Journal of Neuroscience*, vol. 35, no. 27, pp. 10005–10014, 2015.

[23] M. R. Quispe-Ayala, K. Asalde-Alvarez, and A. Roman-Gonzalez, "Image classification using data compression techniques," in *2010 IEEE 26-Th Convention of Electrical and Electronics Engineers in Israel*, (Eilat, Israel), pp. 000349–000353, IEEE, Nov. 2010.

[24] F. Chollet, "keras." `https://github.com/fchollet/keras`, 2015.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[27] K. Janocha and W. M. Czarnecki, "On Loss Functions for Deep Neural Networks in Classification," *arXiv:1702.05659 [cs]*, Feb. 2017.

[28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Jan. 2017.

[29] L. Jiang, "Jalused/Deconvnet-keras," Feb. 2020.

List of Figures

2.1 Illustration of an Artificial Neuron 4
2.2 Illustration of a Multi-layer Perceptron 6
2.3 Convolution Operation 8
2.4 MaxPooling Operation 9
2.5 Feature Hierarchy 10
2.6 DeconvNet Architecture 13

3.1 Architectures of both Classifiers 19

4.1 Representations of Fine-Grained Classifier 21
4.2 Sample Intersections 22
4.3 Conditional Layer Norm 23
4.4 Translation Invariance 24
4.5 Layer Complexity 25

6.1 Representations of Coarse-Grained Classifier 31
6.2 Most frequent Samples of the fine-grained classifier 32
6.3 Most frequent samples of coarse-grained Classifier 33

Declaration of Authorship

I hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Osnabrück, April 06, 2020