TECHNISCHE UNIVERSITÄT
CHEMNITZ

TU Chemnitz

Faculty of Natural Sciences
Institute of Physics

Bachelor Thesis

In the course of degree in computational science (B.Sc.)

To obtain the academic degree Bachelor of Science

Topic:	Investigation of strategies for image classification on small training data sets
Author:	Björn Hempel <bjoern@hempel.li> Matriculation number 025038
Version from:	February 26, 2020
First assessor:	Prof. Dr. Angela Thränhardt
Second assessor:	Dr. David Urbansky

Abstract

Neural networks are extraordinarily good at finding patterns in data. Therefore these networks (models) have to be trained with known data sets and adapted accordingly. Data sets are usually very expensive to obtain and should therefore be used with care and good quality. The training of the models takes place under the influence of many hyperparameters, which have to be determined beforehand. It is important to use the best possible model with its best possible parameters. In this thesis the creation of models for image classification by machine learning is discussed and the model accuracies are compared. The main goal of the thesis is to find optimal parameters and techniques for classification, which also allows to create an optimal model with little training data.

Keywords

Image Classification, Small Data Set, Hyperparameter, Data Augmentation, Convolutional Neural Network (CNN), Machine Learning, Deep Learning

Kurzfassung

Neuronale Netze sind außerordentlich gut darin Muster in Daten zu finden. Hierzu müssen diese Netze (Modelle) vorher mit bekannten Datensätzen trainiert und entsprechend angepasst werden. Datensätze sind meist sehr teuer in der Beschaffung und sollten deshalb mit Bedacht und guter Qualität eingesetzt werden. Das Training der Modelle findet unter dem Einfluss vieler Hyperparameter statt, welche es vorher zu bestimmen gilt. Dabei ist darauf zu achten das bestmögliche Modell mit seinen bestmöglichen Parametern zu verwenden. In dieser Arbeit wird auf das Erstellen von Modellen für die Bildklassifikation mittels maschinellem Lernen eingegangen und es werden die Modellgenauigkeiten miteinander verglichen. Das Hauptziel der Arbeit ist es, optimale Parameter und Techniken für die Klassifikation zu finden, die es auch ermöglichen, mit wenig Trainingsdaten ein optimales Modell zu erstellen.

Schlüsselwörter

Bildklassifizierung, Kleiner Datensatz, Hyperparameter, Data Augmentation, Convolutional Neural Network (CNN), Machine Learning, Deep Learning

Contents

1 Introduction **9**

1.1 Insufficient amount of data . . . 9

2 Background **10**

2.1 Image classification . . . 10

2.1.1 Deductive approach . . . 10

2.1.2 Inductive approach . . . 11

2.1.3 Balanced training data set . . . 11

2.1.4 Training, test and evaluation data set . . . 12

2.1.5 Methods of machine learning . . . 13

2.1.5.1 Supervised learning . . . 13

2.1.5.2 Unsupervised learning . . . 14

2.1.5.3 Reinforcement learning . . . 14

2.1.6 Classification metrics and confusion matrix . . . 14

2.1.6.1 Confusion matrix . . . 15

2.1.6.2 Accuracy . . . 15

2.1.6.3 Precision . . . 16

2.1.6.4 Recall . . . 16

2.1.6.5 F-Measure . . . 16

2.1.6.6 Loss function . . . 17

2.2 Machine learning . . . 17

2.2.1 Artificial neural network . . . 17

2.2.2 Convolutional neuronal network . . . 20

2.2.3 Transfer learning . . . 21

2.2.4 Overview of current and known convolutional neural networks . . . 22

2.3 Further definitions . . . 22

2.3.1 Learning epoch . . . 22

2.3.2 Learning rate . . . 23

2.3.3 Batch size . . . 23

2.3.4 Data augmentation . . . 23

3 Related work **24**

4 Results **25**

4.1 Working environment and model creation . . . 25

4.2 Performance . . . 25

4.3 Model validation . . . 26

4.3.1 Preamble . . . 26

4.3.2 Influence of number of trained images on accuracy . . . 27

4.3.3 Comparison of different CNN models . . . 28

4.3.4 Use of the transfer learning approach . . . 29

4.3.5 Influence of the number of trained layers on the accuracy 30
4.3.6 Influence of different error optimizers 31
4.3.6.1 Comparison optimizer 31
4.3.6.2 Influence of the momentum and the Nesterov momentum 32
4.3.6.3 Influence of a dynamic learning rate on accuracy (scheduling) . . . 33
4.3.7 Different batch sizes 34
4.3.8 Different activation functions 34
4.3.9 Different number of learned epochs 35
4.3.10 Influence of dropout 35
4.4 Optimization process 35
4.4.1 Preamble 36
4.4.2 Data augmentation 36
4.4.3 Enrichment of the data set from other data sources 37
4.4.4 Hierarchical classification 37
4.4.4.1 *k*-means clustering 39
4.4.4.2 Agglomerative hierarchical clustering 40
4.4.5 Binary classifiers 40

5 Summary and outlook **41**

List of acronyms **42**

List of literature **43**

List of links **45**

List of figures

1 Cat and dog comparison . . . 10
2 Deductive approach . . . 11
3 Inductive approach . . . 11
4 Example pictures of a burger class . . . 12
5 Example pictures of a donut class . . . 12
6 Example pictures of a pizza class . . . 12
7 The construction of an artificial neuron . . . 18
8 The construction of a simple neural network . . . 18
9 Simple class shattering . . . 19
10 Linear vs. nonlinear classification . . . 19
11 Simple neuronal network with one hidden layer . . . 20
12 Architecture of a traditional convolutional neural network . . . 21
13 Simple convolution and pooling . . . 21
14 Architecture of a traditional convolutional neural network with transfer learning . . . 21
15 Overview of current and known convolutional neural networks . . . 22
16 Overview of influence of number of trained images on accuracy . . . 28
17 Overview of known cnn models . . . 29
18 Overview of use of the transfer learning approach . . . 30
19 Overview of influence of the number of trained layers . . . 31
20 Overview of best optimizer . . . 32
21 Overview momentum vs Nesterov momentum . . . 32
22 Overview of a dynamic learning rate on accuracy . . . 33
23 Overview of the influence of a different batch size . . . 34
24 Overview of the influence of difference activation functions . . . 34
25 Overview of the influence of a longer training period with more epochs . . . 35
26 Overview of the dropout parameter . . . 35
27 Example of data augmentation . . . 36
28 Comparison of data augmentation . . . 37
29 Principal component analysis of the `food-50` model . . . 38
30 Grouped classes with k-means . . . 39
31 Grouped classes with agglomerative hierarchical clustering . . . 40

List of tables

1 Confusion matrix 15
2 Confusion matrix example 16
3 Performance comparison 26
4 Grouped classes with k-means 39
5 Grouped classes with agglomerative hierarchical clustering 40

1 Introduction

This thesis deals with the creation of models for image classification using machine learning. Many variable parameters (hyperparameters) will have a decisive influence on the accuracy of the model when creating the models and are compared here in detail. Not always only the accuracy is a decisive factor. Also, the required computing time, which is necessary to determine the model, should not be disregarded and should be included in the evaluation. I assume that a small learning rate combined with many learning epochs and correspondingly more computing time required will achieve better results than a few learning epochs combined with a high learning rate (slow adaptation vs. fast adaptation). I also assume that a high quality and a larger amount of data will have a decidedly positive influence on the result. New and more complex CNNs are more successful in model accuracy than older and smaller models.

1.1 Insufficient amount of data

If one gives a person a doughnut and explain to him that it is a doughnut, then after some repetition he is able to classify this doughnut in the future. With Machine Learning this problem is a bit more complex. As with most machine learning methods, a large amount of data is required. How much is not properly documented. Especially when one is dealing with many classes to be predicted, experience shows that the amount of data increases. Some opinions in forums and blog articles say (hypothesis) that there must be at least 1000 pictures per class.[1,2,3,4]

Depending on the number of classes to be trained, one will quickly arrive at the required data set, which consists of several gigabytes of data. With Transfer Learning it is possible to reduce this number a little bit, but the problem of the large amount of data remains. A paper from Microsoft in 2001 showed at that time that simple algorithms with enough data gave similar results as complex algorithms based on less data. The researchers referred to data which should classify language constructs:

"*We have shown that for a prototypical natural language classification task, the performance of learners can benefit significantly from much larger training sets.*"[5]

Another article only a few years later also addresses this issue. This referred to data that learn from texts and that usually only small or medium sized data sets are available. To improve the efficiency also in this case, it is a good idea to improve the algorithms and methods: Examples!

"....[6]"

[1] Abhinav Sagar. *Deep Learning for Image Classification with Less Data*. en. Library Catalog: towardsdatascience.com. Nov. 2019. URL: https://towardsdatascience.com/deep-learning-for-image-classification-with-less-data-90e5df0a7b8e (visited on 02/25/2020).

[2] Pete Warden. *How many images do you need to train a neural network?* en. Library Catalog: petewarden.com. Dec. 2017. URL: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/ (visited on 02/25/2020).

[3] *What is the minimum sample size required to train a Deep Learning model - CNN?*. en. Library Catalog: www.researchgate.net. URL: https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN (visited on 02/25/2020).

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, 2012, pp. 1097–1105.

[5] Michele Banko and Eric Brill. "Scaling to very very large corpora for natural language disambiguation". In: *Proceedings of the 39th annual meeting on association for computational linguistics*. Association for Computational Linguistics. https://www.aclweb.org/anthology/P01-1005.pdf, 2001, pp. 26–33.

[6] Alon Halevy, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12.

2 Background

2.1 Image classification

Classifications are a process of identifying to which class an unobserved object belongs. Several predefined classes can be specified and, based on their properties, an attempt can be made to classify unknown and previously unobserved objects. The procedure for image classification is similar. The previously mentioned objects are now simply images.

Figure 1: Is it a dog or a cat?[7]

For a long time, the automatic recognition of objects, people and scenes in images by computers was considered impossible. The complexity seemed too great to be programmatically taught to an algorithm. Until a few decades ago, attempts were made to achieve image classification by manually developed algorithms. Automated classification based on given and pre-classified images and the automated creation of models was a new step into a new approach. The neural networks developed in this process played a huge role and dramatically changed the way of approach! In the meantime, image recognition has become a widespread application area of machine learning. So-called "convolutional neural networks[8]" or "ConvNets" are often used for images.

The image classification algorithm takes an image as input and classifies it into one of the output categories. Deep Learning has revolutionized the field of image classification and has achieved great results. Various Deep Learning networks, such as ResNet, DenseNet, Inception, etc. have been developed as high-precision networks for image classification. At the same time, image data sets were created to capture tagged image data. These are now primarily used to train existing networks and to organize annual challenges that compete with the model accuracies already known and developed. ImageNet is such a large data set with more than 11 million images and over 11,000 categories. Once a network has been trained with ImageNet data, it can be generalized with other data sets by simple re-compilation or optimization. In this transfer learning approach, a network is initialized with weights that come from a previously trained network. This previously initialized network is now simply adapted for a new image classification task.

The underlying work here is mainly concerned with supervised learning, in which a mathematical model is trained based on existing known data sets. The goal of the trained model is to make the best possible predictions even for unknown images. These known data sets are usually created manually (ontologist), automatically determined based on known facts or determined in a semi-automatic process.

2.1.1 Deductive approach

Since the late 1960s, attempts have been made to classify images with self-written algorithms. This part of computer vision deals with techniques such as image creation, image processing and image segmentation. In the field of image processing, well-known techniques such as edge detection, feature detectors, edge linking, contrast enhancement, etc. are used.[9] Common to all techniques is

[7] Source: https://towardsdatascience.com/image-classifier-cats-vs-dogs-with-convolutional-neural-networks-cnns-and-google-colabs-4e9af21ae7a8

[8] *Convolutional neural network*. en. Page Version ID: 942501792. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=942501792 (visited on 02/25/2020).

[9] Richard Szeliski. "Computer Vision: Algorithms and Applications". en. In: (), p. 979.

the use of the deductive approach. With the deductive approach, one creates rules (feature detectors) which are supposed to predict the desired result. These rules are given and described and thus allow later classification of unknown objects. Since the model and its algorithm are sufficiently well known, this procedure is called a white-box procedure.

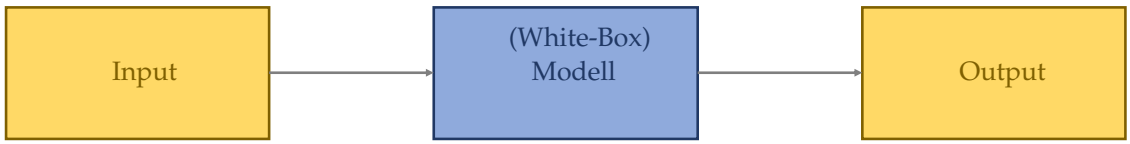

Figure 2: Deductive approach

2.1.2 Inductive approach

The inductive approach, on the other hand, takes a different approach to classifying images. The goal is not to specify a rule, but to learn a rule (model) automatically from already known individual objects. A model is usually a complex function and a mathematical representation of a space (VC dimension[10]), in which individual objects with their properties can be mapped and separated. The model is adapted piece by piece to the known objects in such a way that the input value corresponds to the output value or corresponds to a large extent (backpropagation). The goal is to create a function with this model, which can classify unknown objects in the best possible way. Because the space of this model is mostly far away from the imagination and the possibility of explanation, this procedure is also called a black-box procedure. The procedure described here is mostly used for any kind of supervised learning and is a part of machine learning.

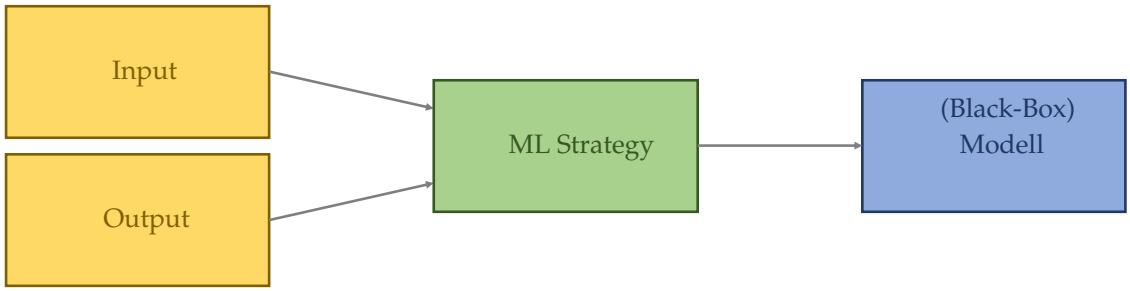

Figure 3: Inductive approach

2.1.3 Balanced training data set

Neural networks have made enormous progress in the field of pattern recognition in recent years. A decisive factor is that the data for learning must be of high quality and easy for the network to process. Wrongly classified or irrelevant data could cause the network to learn something wrong. This also applies to non-existent or unsuitable pre-processing.[11]

With the beginning of a classification project, the question arises what exactly one wants to classify and how extensive the classification should be. Assuming one wants to identify different classes of food, this could be classified as pizza, burgers, doughnuts and lasagna (etc.). For these classes, one now needs a large number of images. Ideally, this data should reflect reality as well as possible. A large variation is advantageous (balanced data set): different viewing angles, size, position, colour brightness, variations, number, etc. Images of e.g. only one colour brightness or

[10] *Vapnik–Chervonenkis dimension*. en. Page Version ID: 942482212. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Vapnik%E2%80%93Chervonenkis_dimension&oldid=942482212 (visited on 02/25/2020).

[11] Douwe Osinga. *Deep Learning Kochbuch: Praxisrezepte für einen schnellen Einstieg*. O'Reilly Verlag, 2019, pp. 19–26. ISBN: 9783960090977.

only one viewing angle should be avoided. If the data are not balanced, they must be corrected accordingly: e.g. by adding further data, image processing or by removing data that causes an imbalance. Furthermore, the selected classes should be clearly optically separable from each other. If two classes are visually very similar and not really distinguishable even by a human, consideration should be given to combining them (e.g. "burger" and "veggie burger"):

Figure 4: Example pictures of a burger class

Figure 5: Example pictures of a donut class

Figure 6: Example pictures of a pizza class

Accessing data is often not that easy. Every data source has its own special features. One way to access data would be an automatic crawling of image databases, search engines or reviews in which images appear. A certain amount of creativity is advantageous:

- Google
- Bing
- Flickr
- TripAdvisor
- etc.

Probably the most expensive way to obtain data is to search and classify them manually, e.g. by an ontologist. The ontologist evaluates and searches for different images and manually classifies them in the appropriate classes. A combined variant is also possible and probably preferable: automatic crawling and manual sorting out of incorrect, unfavorable or irrelevant images.

2.1.4 Training, test and evaluation data set

Before starting the training of balanced images, they must be divided into a training, a test and possibly a validation data set. This is necessary because neural networks will not generalize to some extent, but will learn by heart (overfitting[12]). The idea is to train with a training data set, while the validation data set is used to monitor the general validity of the network and its parameters.

[12] *Overfitting*. en. Page Version ID: 942053730. Feb. 2020. URL: `https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=942053730` (visited on 02/25/2020).

Based on the results, adjustments are made at runtime. Since the adjustment of the parameters is carried out using the test data, there is also an independent test data set, which carries out a renewed check of the model for previously uninvolved data. This ensures that hyperparameters are not inadvertently optimized for the validation data set only.[13] The use of the test data set is optional and simulates the model under real conditions. If the number of data is limited, this data record can also be added to the training data record, for example. In this thesis, the test data set is not used and all evaluations refer to the validation data set.

An optimal division of the training and validation data set depends on the existing classification problem and the amount of data available. In this thesis a ratio of 80 percent training data and 20 percent validation data is used, unless otherwise stated.

The question of why 80 percent training data and 20 percent validation data are used remains to be clarified. Is there a paper or a study on this? Or is another test required?

2.1.5 Methods of machine learning

Different machine learning systems can be classified according to the type and procedure of monitoring the training. A distinction is made as to which type of data is available or has to be determined by the user.

2.1.5.1 Supervised learning

Supervised learning refers to machine learning with known training data sets (see also chapter "inductive approach"). The learning process in turn refers to the ability of an artificial intelligence to reproduce regularities and patterns. The results are known by laws of nature or expert knowledge and are used to teach the system by creating a training set containing the desired solutions. This is also called labelled data. The learning algorithm now tries to find a hypothesis epoch by epoch, which allows the most accurate predictions on unknown data. A hypothesis in this case is an image that assigns the assumed output value (the predicted class) to each input value (the image itself). This work makes extensive use of supervised learning.

In supervised learning, an input vector is fed to a classification function (usually an artificial neural network). The input vector generates an output vector using the classification function, which produces this neural network in its current state[14]. This value is compared with the value that it should actually output. The comparison of the nominal and actual state provides information on how and in what form changes must be made to the network in order to further approximate the actual state and minimize the error. For artificial neural networks without a hidden layer (single-layer perceptron[15]), the delta rule[16] for correction can be applied. For networks with one or more hidden layers backpropagation[17] is used to minimize the error. Backpropagation is a generalization of the delta rule.

The neural network is only one algorithm from the category of supervised learning algorithms. For completeness here is a list of further algorithms:

- k-nearest neighbors[18]
- Linear regression[19]

[13] Osinga, *Deep Learning Kochbuch: Praxisrezepte für einen schnellen Einstieg*.

[14] The neural network consists of many (usually millions) parameters, which can be adjusted during the learning process to minimize the error.

[15] *Perceptron*. en. Page Version ID: 942271496. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Perceptron&oldid=942271496 (visited on 02/25/2020).

[16] *Least mean squares filter*. en. Page Version ID: 941899198. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Least_mean_squares_filter&oldid=941899198 (visited on 02/25/2020).

[17] *Backpropagation*. en. Page Version ID: 939314095. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=939314095 (visited on 02/25/2020).

[18] "k-nearest neighbors algorithm", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[19] "Linear regression", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/Linear_regression

- Logistic regression[20]
- Support-vector machine[21]
- Random forest[22]
- etc.

2.1.5.2 Unsupervised learning

In unsupervised learning, one tries to gain knowledge of patterns even without labelled data. Suppose one has several pictures of burgers, pizza and doughnuts, which are unsorted in a data set. Unsupervised learning now tries to find similarities in order to cluster these images. In the best case one gets three unnamed groups *A*, *B* and *C* at the end. Analysts will take a closer look at these groups afterwards and draw a conclusion if possible: Group *A* is then called Burger, Group *B* Pizzas, etc.

The following unsupervised learning algorithms can be used for clustering:

- k-means[23]
- Hierarchical clustering[24]
- Expectation–maximization [25]
- etc.

One technique, hierarchical clustering (see chapter "Hierarchical classification"), is used later to facilitate the introduction of hierarchies. For the general analysis, the finding of optimal parameters for learning models, this kind of learning is not used in this thesis.

2.1.5.3 Reinforcement learning

Reinforcement learning[26] is a type of machine learning in which an agent independently learns the best possible strategy for achieving a goal. To achieve the goal, actions are necessary which produce rewards at certain points in time. These rewards can also be negative (punishment). Based on these rewards, the aim is to achieve the best possible reward value. This type of learning is not relevant for the classification of images, which is why it will not be discussed further here.

2.1.6 Classification metrics and confusion matrix

Choosing the right metric is crucial in evaluating machine learning models. Metrics are used to monitor and measure the performance of a model during training and testing. Some important metrics are explained below.

[20] "Logistic regression", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/Logistic_regression

[21] "Support-vector machine", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/Support-vector_machine

[22] "Random forest", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/Random_forest

[23] "k-means clustering", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/K-means_clustering

[24] "Hierarchical clustering", Wikipedia contributors, February 2, 2020, https://en.wikipedia.org/wiki/Hierarchical_clustering

[25] "Expectation–maximization algorithm", Wikipedia contributors, February 2, 2020, Expectation\OT1\textendashmaximizationalgorithm

[26] "Reinforcement learning", Wikipedia contributors, January 31, 2020, https://en.wikipedia.org/wiki/Reinforcement_learning

2.1.6.1 Confusion matrix

The Confusion Matrix is a special quadratic matrix in the field of machine learning that allows the visualization of the performance of a predictive model. Each row of the matrix represents the actual class, while each column indicates the number or a numerical value as a percentage of the predicted class (or vice versa)[27]:

		predicted			
		$class_1$	$class_2$	...	$class_n$
actual	$class_1$	*TP*	*FN*		
	$class_2$	*FP*	*TN*		
	...				
	$class_n$				

Table 1: Confusion matrix

Finally, the Confusion Matrix has the following structure, where the number of elements in the class $C_{i,P}$ was predicted although ($\cong$) it should have been class $C_{j,A}$:

$$\boldsymbol{M}_{confusion} = \begin{bmatrix} \#C_{1,P} \cong C_{1,A} & \dots & \#(C_{n,P} \cong C_{1,A}) \\ \vdots & \ddots & \vdots \\ \#(C_{1,P} \cong C_{n,A}) & \dots & \#(C_{n,P} \cong C_{n,A}) \end{bmatrix} = (a_{nn}) \tag{1}$$

2.1.6.2 Accuracy

Top-1 accuracy is probably the most important accuracy. It tells one the percentage of the model's best prediction of the data in the validation set that matches the expected class.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\sum_{i,j=1}^{n} a_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}} = \frac{Correct_{all}}{CorrectPossible_{all}} \tag{2}$$

The **Top 5 Accuracy** is another accuracy specification. However, not only the best hit is included here, but also the next four. As soon as the correct class can be found within the first five predicted classes, this prediction is also true:

$$Accuracy_{top-5} = \frac{CorrectWithinTheBest5Classes_{all}}{CorrectPossible_{all}} \tag{3}$$

The accuracy of the entire model is a good indication of the performance of the model. However, a problem occurs in extreme cases where assumptions can no longer be made reliably. For example, if one is working with an unbalanced dataset.[28] Example: Suppose one has a model that always predicts the class $class_1$. The class $class_1$ consists of 9990 elements and from the other classes $class_2$ to $class_n$ one has exactly 10. Then the Confusion Matrix looks like this:

The model accuracy in this case is 99.9%, although it is a bad model:

$$Accuracy = 99,9\% \tag{4}$$

Therefore there are additional performance metrics like Precision, Recall and F-Measure.

[27] "Confusion matrix", Wikipedia contributors, February 5, 2020, `https://en.wikipedia.org/wiki/Confusion_matrix`

[28] Aurélien Géron. "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Konfusionsmatrix, pp. 86–88. ISBN: 9783960090618.

		predicted			
		$class_1$	$class_2$	...	$class_n$
actual	$class_1$	$TP = 9990$	$FN = 0$		
	$class_2$	$FP = 10$	$TN = 0$		
	...				
	$class_n$				

Table 2: Confusion matrix example

2.1.6.3 Precision

Precision[29] expresses how reliable the statement of a prediction of a class is:

$$Precision = \frac{Correct}{Actual} = \frac{TP}{TP + FP} \tag{5}$$

Or more precisely for class c:

$$Precision_{@c} = \frac{a_{cc}}{\sum_{i=1}^{n} a_{ic}} \tag{6}$$

2.1.6.4 Recall

Recall[30] is the accuracy of a class. This means how well the class could be predicted:

$$Recall = \frac{Correct}{CorrectPossible} = \frac{TP}{TP + FN} \tag{7}$$

Or more precisely for class c:

$$Recall_{@c} = \frac{a_{cc}}{\sum_{i=1}^{n} a_{ci}} \tag{8}$$

2.1.6.5 F-Measure

F-Measure[31] combines precision and recall, with the parameter β representing the weighting:

$$F_\beta = (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall} = \frac{(1 + \beta^2) \cdot TP}{(1 + \beta^2) \cdot TP + \beta^2 \cdot FN + FP} \tag{9}$$

The higher β the more importance is placed on precision instead of recall. This is important if one puts more importance on the quality of the prediction than on the accuracy of prediction. For example, when predicting diseases: assigning class "ill" to healthy people is just as fatal as assigning class "healthy" to sick people (although case two would be even more fatal than case one). A value of $\beta = 0.5$ one gets an equal distribution of both values and is called F1 score:

$$F_1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FN + FP} \tag{10}$$

[29] "Precision and recall", Wikipedia contributors, February 7, 2020, `https://en.wikipedia.org/wiki/Precision_and_recall`

[30] "Precision and recall", Wikipedia contributors, February 7, 2020, `https://en.wikipedia.org/wiki/Precision_and_recall`

[31] "F1 score", Wikipedia contributors, February 7, 2020, `https://en.wikipedia.org/wiki/F1_score`

2.1.6.6 Loss function

Since this is a classification problem and not a regression problem, the prediction is based on softmax regression. After each prediction, a vector λ of the size n is returned, where n corresponds to the number of classes to be distinguished.[32] Each individual $\hat{p}$ value corresponds to the probability that it is class $class_n$:

$$\lambda = \begin{pmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \vdots \\ \hat{p}_n \end{pmatrix} \quad \Big| \quad \sum_{i=1}^{n} \hat{p}_i = 1 \tag{11}$$

The expected value of the parameter function and thus of the current class is returned as a one hot vector. The value 1 corresponds to the expected class. All other classes return 0. This is also called one hot encoding:

$$g(\vartheta_2) = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} \tag{12}$$

The loss function[33] assigns a loss to each prediction, which results from the comparison with the true value or parameter. For this purpose, the distance between the predicted class and the true class is calculated (if they are equal, the distance is 0). If the adaptation of the model (learning process) improves the prediction of all predicted classes to their true classes, the value of the loss function is also reduced. A typical loss function is e.g. for an r-dimensional space:

$$L_r(\vartheta, \lambda) := \|\lambda - g(\vartheta)\|^r \tag{13}$$

λ represents the estimated value and $g(\vartheta)$ the parameter function which returns the actual value for ϑ. The average loss on the entire data set with n elements is thus:

$$\hat{L} = \frac{1}{k} \sum_{i=1}^{k} L_r(\vartheta_i, \lambda_i) \tag{14}$$

2.2 Machine learning

Machine learning is a generic term for the artificial generation of knowledge from experience. It follows the approach of inductive learning (see also chapter "inductive approach").

2.2.1 Artificial neural network

Artificial neural networks provide functions that are able to separate highly complex data in multidimensional space. For large and highly complex tasks, such as the classification of billions of images, speech and text recognition, neural networks usually perform better than other machine learning methods. The significant increase in computational capacity since the 1990s allows the training of large neural networks within a reasonable period of time. Artificial neural networks are the core component of deep learning, which denotes a method of machine learning.

Neural networks process an input vector $\bar{x}$ and convert it into a new output vector $\hat{\bar{x}}$. They are networks of many artificial neurons connected in series and parallel. An artificial neuron in turn

[32] Aurélien Géron. "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Entscheidungsgrenzen, pp. 138–140. ISBN: 9783960090618.

[33] "Verlustfunktion (Statistik)", Wikipedia contributors, February 7, 2020, `https://de.wikipedia.org/wiki/Verlustfunktion_(Statistik)`

converts a vector into a scalar by scaling and summing the inputs $\bar{x}$ with the changeable parameters $\bar{\omega}$ and correcting them with a bias b (the bias is also a changeable variable). The activation function ensures that the first degree polynomial (linear regression model) becomes a nonlinear function[34]:

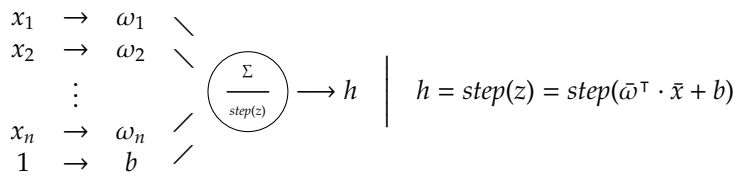

Figure 7: The construction of an artificial neuron

The artificial neural network is composed of many layers connected in series, which again contain neurons connected in parallel:

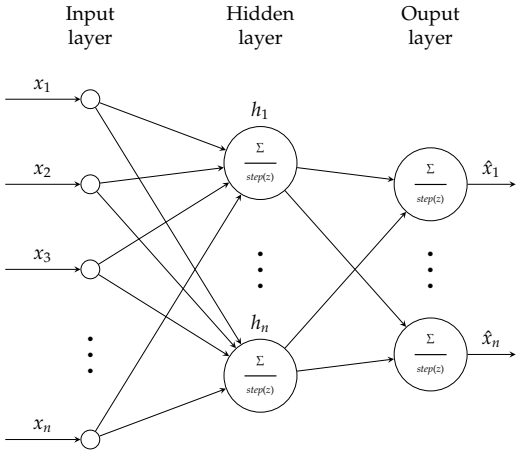

Figure 8: The construction of a simple neural network

A neural network is able to classify complex inputs. But how exactly does that work? The following classification function is able to separate simple class problems, where $\bar{x}$ represent the coordinates of the corresponding class points and $\bar{w}$ and b are learnable parameters:

$$f(\bar{x}, \bar{\omega}, b) = sgn(\bar{\omega}^\intercal \cdot \bar{x} + b) \tag{15}$$

With this linear function the following problem can be easily classified:

[34] "Activation functions, their types and uses", https://www.ai-united.de/, February 8, 2020, `https://www.ai-united.de/aktivierungsfunktionen-ihre-arten-und-verwendungsmoeglichkeiten/`

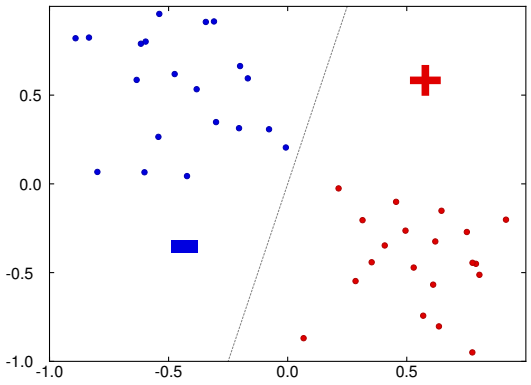

Figure 9: Simple class shattering

The function corresponds to a neural network without a hidden layer and contains only one input and one output layer with one artificial neuron without activation function. The dimension that this function can separate is two and is called VC dimension[35]. In other words, this function can separate exactly two classes linearly. But what about nonlinear problems?:

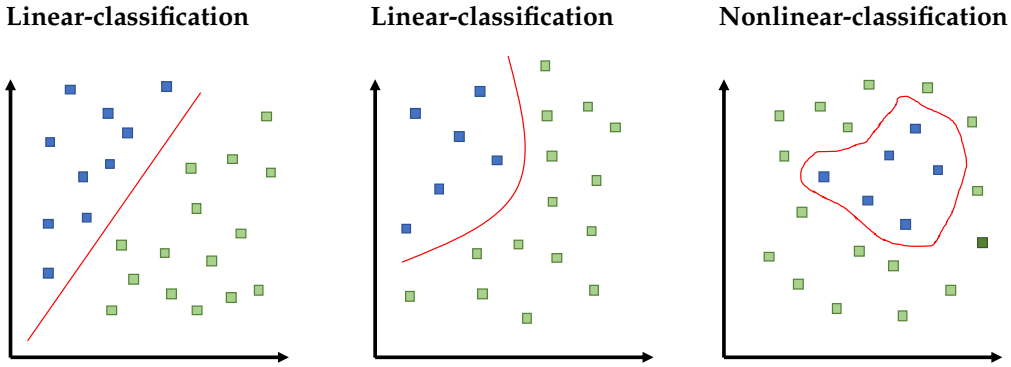

Figure 10: Linear vs. nonlinear classification

For the second problem one can still adjust the function. For the third nonlinear problem the classification space is no longer sufficient and requires a different algorithm. And this is where the neural networks come into play. A tool to visualize the separation of data and to test the functionality of the individual layers is `https://playground.tensorflow.org`[36]. In the simplest case, a nonlinear problem can be solved by adding a hidden layer with three additional neurons:

[35] Vapnik–Chervonenkis dimension: `https://en.wikipedia.org/wiki/Convolutional_neural_network`
[36] Neural Network Right Here in Your Browser: `https://playground.tensorflow.org`

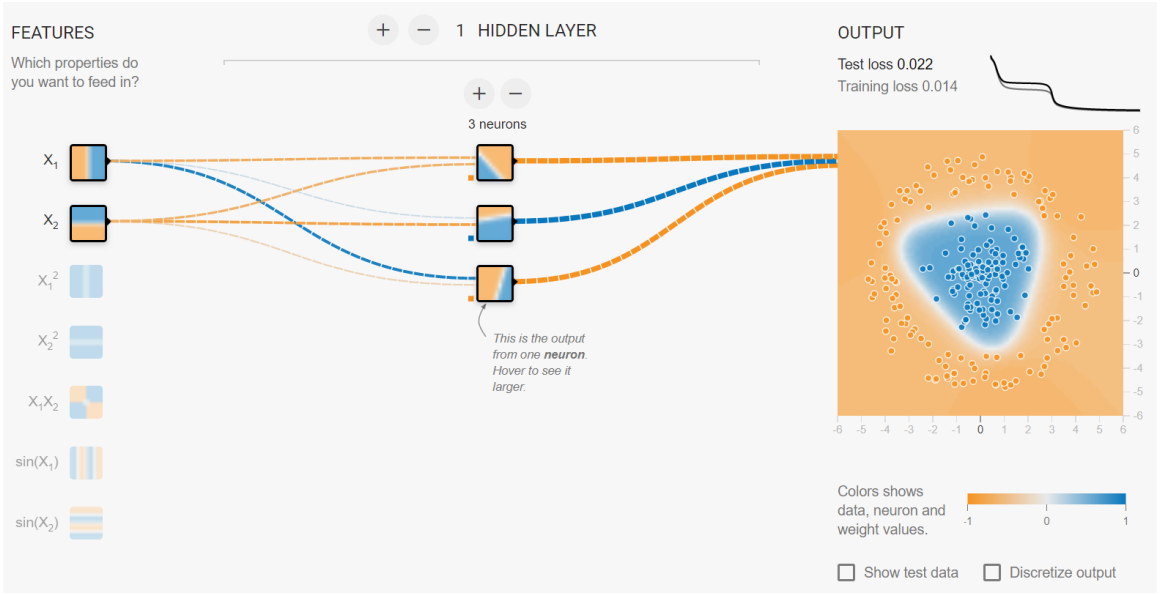

Figure 11: Simple neuronal network with one hidden layer[37]

2.2.2 Convolutional neuronal network

A neural network processes a vector and returns a new vector. The problem with input data such as images is that at first view they cannot be successfully described as vectors to be trained with a normal neural network. One needs an algorithm that can handle matrix-like inputs and that is able to recognize patterns. In the past the principle of the convolutional layer was developed. A convolutional layer receives a matrix input, transforms it and returns an output value (in this case another matrix). This output value is then passed on to the next layer. A convolutional layer contains a set n of square matrices (usually 3x3 or 5x5 matrices). These matrices are called filters. Each filter[38] is now calculated from top left to bottom right over the pixels of the image using a scalar product, which creates a new image. The so-called Feature Map. With a number of n filters, n feature maps are created at the end and highlight the features defined in the filters in the newly calculated image. This process is also called convolution.[39]

Neural networks that make use of convolutional layers are called convolutional neural networks (in short CNN) and have made a decisive contribution to the progress of image classification and also in other areas like speech recognition. In addition to the convolutional layers, a convolutional neural network has other special layers that differ from normal neural networks: For example the Pooling Layer. In a pooling layer, unnecessary information is discarded and feature maps are reduced in size. This process reduces memory requirements and increases calculation speed. The convolutional layer and the pooling layer usually alternate until a large vector is created at the end instead of a *nxn* matrix of the input image, which can be further processed by a normal neural network and finally ends in the already described one hot vector.

Idea? `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`

[37] Source: `http://playground.tensorflow.org/`

[38] Filters, which can recognize edges, corners, squares, etc. and in deeper layers things like eyes, ears, hair, etc.

[39] deeplizard. *Convolutional Neural Networks (CNNs) explained*. Youtube. 2017. URL: `https://www.youtube.com/watch?v=YRhxdVk{\_}sIs`.

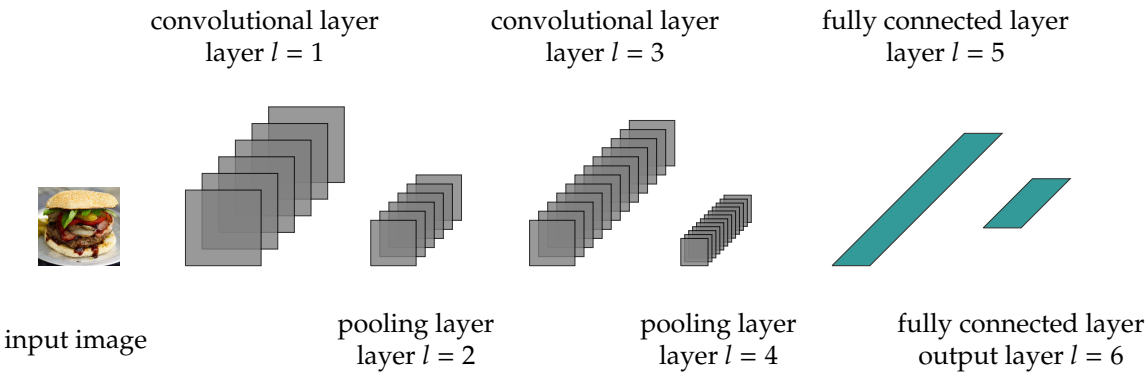

Figure 12: Architecture of a traditional convolutional neural network

A big advantage of convolutional neural networks should not remain unmentioned: They require relatively little preprocessing compared to other image classification algorithms. This means that the network independently learns the filters that are normally developed by hand in conventional algorithms, if trained with adequate training. This property of these networks is a great advantage because they can be automated and change independently when the input data changes and do not require human intervention.

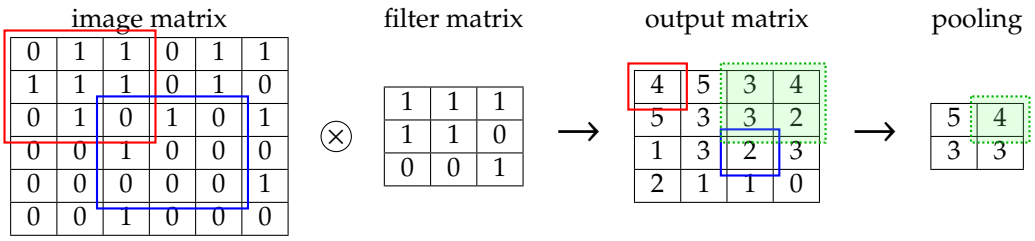

Figure 13: Simple convolution and pooling

2.2.3 Transfer learning

Convolutional neural networks are great and have made a significant contribution to the classification of images. With the foundation of the research database ImageNet in 2006, annual competitions are organized to compare developed neural networks. ImageNet is an image database with more than 14 million images. A CNN called AlexNet in 2012 got a top-5 error of 15.3% and is currently increasing steadily every year. But the architecture of a CNN has a problem. All convolutional layers are randomly initialized from the beginning and do not yet contain any patterns. For it to work reliably, it needs to be trained with many images. If one would develop and use a CNN from scratch, all convolutional layers have to be trained in advance.

The convolutional layers extract features such as edges, squares, circles, etc. These are present in almost every image and the question arises whether one can reuse them to reduce the training effort. The idea of Transfer Learning is to use an already pre-trained CNN and just adapt the neural network at the end of the convolutional neural network to the own problem:

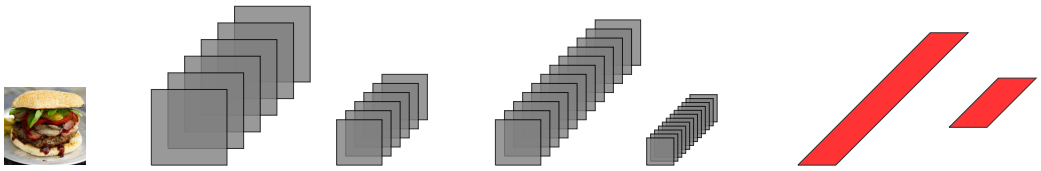

Figure 14: The green area (the neural network) was replaced by a network (red) adapted to the new problem (see Figure 12: Architecture of a traditional convolutional neural network).

The advantage of a pre-trained network can be seen in the chapter "Use of the transfer learning approach" of this thesis.

2.2.4 Overview of current and known convolutional neural networks

Last but not least, here are a few current and well-known convolutional neural networks. They differ mainly in the following metrics, whereby in combination each network has its advantages and disadvantages:

- the top-1 accuracy (based on the ImageNet image dataset)
- the computing operations which are required for a single forward pass (G-Ops)
- the model size (for comparison: the model size of InceptionV3 is about 180 MB)

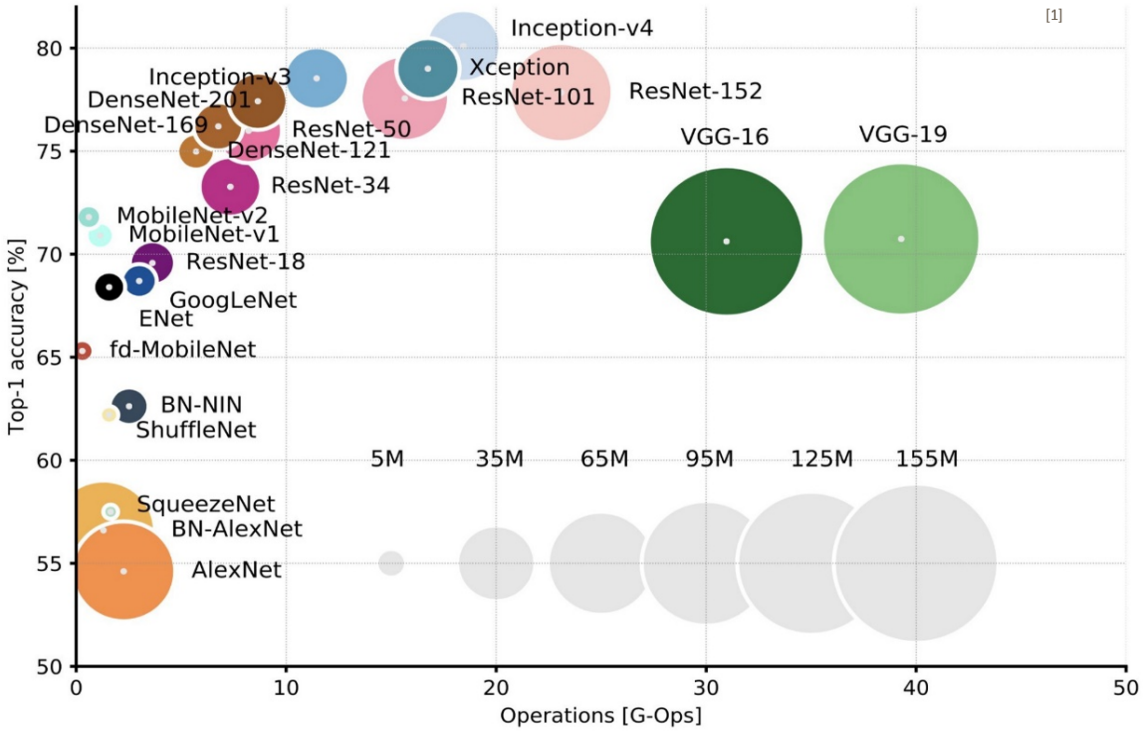

Figure 15: Overview of current and known convolutional neural networks[40]

2.3 Further definitions

2.3.1 Learning epoch

A learning epoch is understood to be the one time training run with all training data sets. Usually one training run is not sufficient, which is why further learning epochs follow. As soon as the performance of the network does not increase anymore with further epochs, the training is terminated.

[40]Source: `https://towardsdatascience.com/neural-network-architectures-156e5bad51ba`

2.3.2 Learning rate

The learning rate is a tuning parameter in an optimization algorithm. It expresses how large the step size is for each iteration with which the parameters move closer to the minimum of a loss function. It should not be too high (minimum is not found) and not too small (very slow learning). The learning rate is often indicated by the character η or α.

2.3.3 Batch size

The batch size defines the number of images that are transmitted simultaneously over the network before the parameters in the network are customized. Normally it is intended to train all training data per epoch with one process. Especially in the area of image classification this is usually not possible due to the limited memory, so an epoch is divided into many small batches. The advantage is that one only has to keep the data in memory which is necessary for the current batch. The disadvantage is the increased computing effort and an inaccurate estimation of the gradient, because the change only applies to the current batch and not to all images[41].

2.3.4 Data augmentation

Data augmentation is the artificial augmentation of a data set. This is primarily used when only little data is available. Existing images are rotated, mirrored, color adjusted, cropped, etc. Data Augmentation can improve model accuracy during training. Is there an example that proves this quickly?

[41] "What is batch size in neural network?", itdxer on stackexchange.com, February 9, 2020, https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network

3 Related work

There are a number of studies on image classification on very large datasets.[42,43,44] Most of the time this huge number of classification classes and datasets is not desired. Furthermore, there are many investigations on small data sets.[45,46] What these investigations usually lack is an overview and comparison of the common hyperparameters with respect to model accuracy in a single thesis. This thesis will deal with this. Data is expensive to obtain and usually not easy to get (see chapter "Balanced training data set"). And in contrast, image classifications are appearing in more and more areas of our life and are also being used directly in more and more companies that have not made use of them so far and are now considering introducing their own implementations. There are e.g. companies which try to classify products based on different data. Is it a good idea to implement your own implementation or is the step to the software tools of company giants like Microsoft, Google and Co. unavoidable? A person sees a product *X* and classifies it: From the text, the description or an image. Sometimes only an image remains, because e.g. texts have not been maintained properly or only return cryptic values. And even then, this classification is often not a challenge for humans: Because in this case, they recognize the product *X* based on the still existing image.

This thesis deals with the image classification part. Since these are companies with limited resources, the question is: With which tools, techniques and tricks is it possible to carry out a successful image classification. Do the conditions mentioned in subsection 1.1: Insufficient amount of data have to be fulfilled or are successful classifications already possible with less? Is it possible to get the most out of model creation by adjusting certain tuning parameters? For example, is a cluster analysis and the associated categorical breakdown of the classification a successful approach? These and other questions are to be clarified with this thesis. And I dare to say at this point that it is possible to reach a good goal even with fewer requirements.

[42] Jia Deng et al. "What does classifying more than 10,000 image categories tell us?" In: *European conference on computer vision*. Springer. http://vision.stanford.edu/pdf/DengBergLiFei-Fei_ECCV2010.pdf, 2010, pp. 71–84.

[43] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Sun_Deep_Learning_Face_2014_CVPR_paper.pdf, 2014, pp. 1891–1898.

[44] Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks".

[45] Francois Chollet. "Building powerful image classification models using very little data". In: *Keras Blog* (2016).

[46] Sagar, *Deep Learning for Image Classification with Less Data*.

4 Results

In this part of the thesis the investigations and the corresponding evaluations will be presented. For each chapter models with the given parameters and techniques have been created with which image classifications can be performed. The model has to be trained, whereby a decision has to be made on many variable parameters like the learning rate η, the optimizers, but also things such as the CNN model. The basis is a standard setup, which is explained in the following chapter "Preamble". Starting from this standard setup, the variable parameters are changed and discussed in the corresponding chapters. The goal is to find the parameters with which the recognition accuracy of the models is particularly high: high accuracy and small loss function.

4.1 Working environment and model creation

All source code for the environment and the framework to train models and which has been used for this thesis is available in the following Github repository: `https://github.com/bjoern-hempel/keras-machine-learning-framework`[47]. This framework was written in Python. Python has become one of the most important programming languages in the field of machine learning and data sciences in recent years.[48] The libraries of this programming language contain many ready-made and helpful functions and basic frameworks right from scratch. The community with its many forums and blog articles is also very broad. As Python Machine Learning Framework Keras was used, because Keras contains as many uniform interfaces to backends like TensorFlow and the produced model can also be reused in other programming languages like Java[49].

All models in the following chapters were created with the above mentioned framework, which allows to set all hyperparameters mentioned in this thesis: `http://bit.ly/2SNVnvE`[50]. A command line call with the default values could look like the following after successful installation:

```
user$ ml train \
      --use-train-val \
      --data-path=./data/raw/food-50 \
      --model-file=./data/processed/experiment1/type-of-experiment/model.h5
```

Listing 1: Example command line call to train a specified data path

4.2 Performance

If one intends to implement and optimize deep neuronal networks (DNN), the calculations must take place on the GPU. It is also possible to run calculations on the CPU. Also, the installation of Keras for CPU driven computations is much easier, because the installation of the GPU drivers is not necessary. The disadvantage of this, however, is that it takes much longer to train larger models. Good models for the classification of e.g. pictures are only achieved after several training units. Training units require a lot of computing power in the form of many matrix operations. A GPU is predestined for matrix operations.[51]

[47] Keras Machine Learning Framework, Github, Febrary 21, 2020, `https://github.com/bjoern-hempel/keras-machine-learning-framework`

[48] *The Most Popular Language For Machine Learning Is ... (IT Best Kept Secret Is Optimization).* en. CT904. Library Catalog: www.ibm.com. Aug. 2015. URL: `www.ibm.com/developerworks/community/blogs/jfp/entry/what_language_is_best_for_machine_learning_and_data_science` (visited on 02/25/2020).

[49] Keras Machine Learning Framework (Java Sources), Github, Febrary 21, 2020, `https://github.com/bjoern-hempel/keras-machine-learning-framework-java-sources`

[50] Keras Machine Learning Framework - Arguments, Github, Febrary 21, 2020, `https://github.com/bjoern-hempel/keras-machine-learning-framework/blob/master/markdown/image-classification/arguments.md#user-content-arguments-of-the-training-process`

[51] Kayvon Fatahalian, Jeremy Sugerman, and Pat Hanrahan. "Understanding the efficiency of GPU algorithms for matrix-matrix multiplication". In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. https://graphics.stanford.edu/papers/gpumatrixmult/gpumatrixmult.pdf, 2004, pp. 133–137.

The following is a tabular comparison of performance based on the Kaggle flower data set training[52]. This image set consists of 5 classes with about 4242 training images (10 epochs, InceptionV3)[53]:

Device	Preparation	Train	Train (Factor)	Save model
NVIDIA GeForce GTX 1060 6GB (Desktop) - Windows	17.5s	303.5s 00:05:03.5	1.00x	33.8s
NVIDIA GeForce GT 750M 2GB (Notebook) - Windows	18.4s	2415.0s 00:40:15.0	7.96x	29.8s
Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz (Single Core) - MacOS	19.1s	6393.7s 01:46:33.7	21.07x	41.4s
Intel(R) Core(TM) i7-4712HQ CPU @ 2.30GHz (Single Core) - Windows	16.9s	9016.8s 02:30:16.8	29.71x	28.7s
Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz (Single Core) - Windows	16.3s	25183.4s 06:59:43.4	82.97x	28.3s

Table 3: Performance comparison

While the type of computing device (CPU or GPU) makes no difference in terms of preparation and postprocessing, it does make a significant difference in training time. The slowest CPU takes more than 80 times as many time as the fastest graphics card unit. The choice of the computing device for all further tests clearly falls on the GPU.

4.3 Model validation

4.3.1 Preamble

Before the validation process can be started, the data set used must be prepared properly. The data set used in this thesis is a meal data set with a number of 50 classes: `food-50`. This data set contains 50 folders with the respective existing pre-classified images with a size of 765 MB. The folders must be divided into a validation record and a training record. The ratio is 20% validation and 80% training data:

Training data

- 11913 images
- classified within 50 classes
- different number of images per class (unbalanced)

Validation data

- 2953 images
- classified within 50 classes
- different number of images per class (unbalanced)

[52] Flowers Recognition, Kaggle, Febrary 21, 2020, `https://www.kaggle.com/alxmamaev/flowers-recognition`

[53] GPU vs CPU, Github, Febrary 21, 2020, `https://github.com/bjoern-hempel/keras-machine-learning-framework/blob/master/markdown/hardware/gpu-vs-cpu.md`

Default setup

With the exception of the cnn model tests, all tests were based on the following parameters (whereby one value of the parameters varied depending on the chapter):

- **model:** InceptionV3
- **learning rate:** 0,001 (decreases every 7 epochs to 50% of the previous value)
- **epochs:** 21 (learning rate from epoch 15 to 21: 0,00025)
- **image size:** 299x299 pixels
- **batch size:** 16
- **drop out:** 50%
- **cnn weights:** imagenet
- **activation function:** relu
- **optimizer:** sgd with Nesterov
- **momentum:** 0.9 (with decay 0.0)
- the entire training and validation set (14866 images - unless otherwise specified)

Different models were tried out in chapter "Comparison of different CNN models" with the same parameters as above:

- DenseNet121
- DenseNet201
- InceptionResNetV2
- InceptionV3
- NASNetLarge
- ResNet50
- VGG19
- Xception

4.3.2 Influence of number of trained images on accuracy

The first question that comes up: What influence does the amount of data to be trained have on the accuracy of the model? For this purpose, the standard setup from the chapter "Preamble" is used and the model is trained with a different number of training data. While the validation data set always stays the same, the total number of data to be trained is increased from 500 data sets to the total number of 11913. Since this is an unbalanced data set, the number remains the same in each class in percentage terms. Only the total number is changed to the corresponding values (500, 1000, 2000, ...):

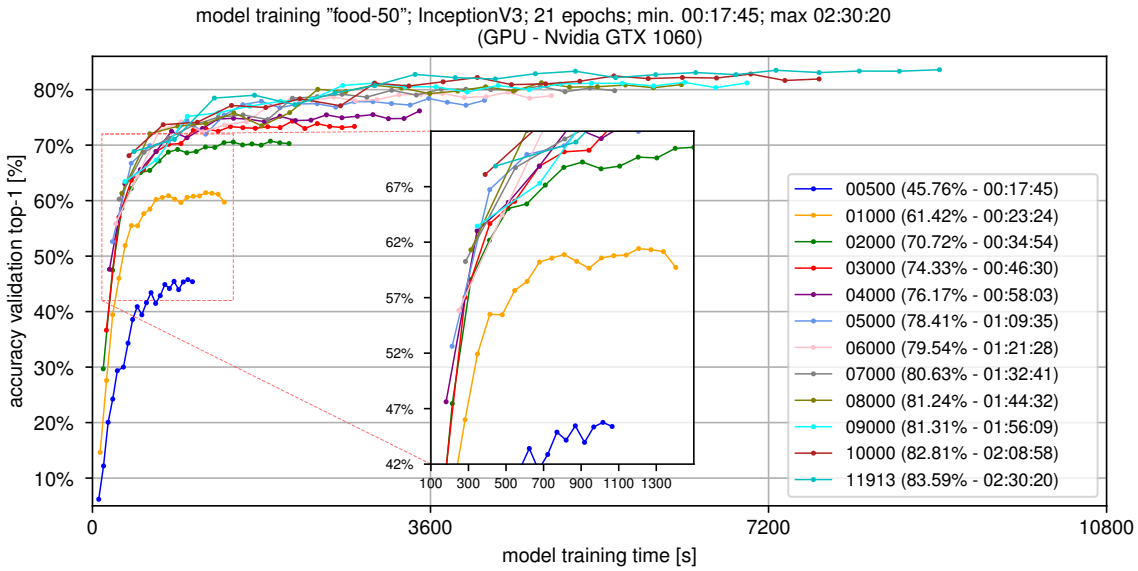

Figure 16: Overview of influence of number of trained images on accuracy

As expected, the number of images to be trained has a significant influence on the model accuracy. While with 500 images an accuracy of 45.76% can be achieved, with almost 12000 images it is already 83.59% after 21 epochs. With every increase in the number of images, an improvement in accuracy can be observed. The improvement in accuracy is not linear with an increase in training data sets. Although the accuracy in the upper range (>8000 images) still seems to increase, the jumps are not very big anymore and seem to be moving to a limit. For further research, it would be a good idea to find out from which number of training data sets no further significant increases in accuracy are possible. Since no further data was available in this data set, this project will not be pursued further here.

Another interesting point is the fact that increasing the training data theoretically saves computing time up to a certain point. If the data sets are significantly too small, one will never reach an accuracy during training, which models with much more data reach and exceed already with the first epoch. Clearly to see: With 1000 images (orange trace) one gets a maximum accuracy of 61.42% after 18 epochs and about 20 minutes of computing time. If one uses 10 times the amount of images (10000 images - brown trace), one reaches an accuracy of 68.1% already in the first epoch after six and a half minutes.

4.3.3 Comparison of different CNN models

There are currently many CNN models and each year with the annual ImageNet competition new models with better accuracy are released. These models are trained with the ImageNet dataset and the results are compared using this set of data. What is the result with smaller sets of data? The training result according to the standard setup and with different CNN models of eight models is as follows:

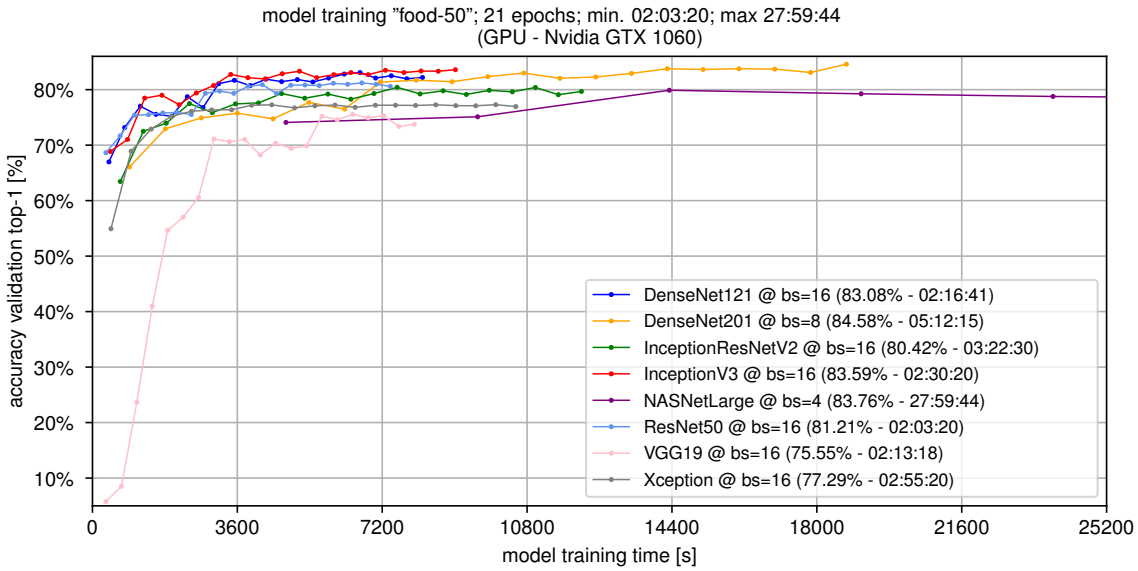

Figure 17: Overview of known cnn models

All models except DenseNet201 and NASNetLarge were trained with a batch size of 16 ($bs = 16$). Due to the limited 6GB memory of the Nvidia GTX 1060[54], DenseNet201 was trained with a batch size of 8 and NASNetLarge with a batch size of 4. A real comparison of these two exceptions isn't really possible, but shouldn't be omitted at this point. A larger batch value is associated with increased computing time and slower increase in accuracy (see chapter "Different batch sizes"). A comparison with Figure 15: Overview of current and known convolutional neural networks shows similar accuracies and required computing time: The best model was achieved with DenseNet201 with an accuracy of 88.58%. The model with the lowest accuracy is the somewhat older VGG19 model with 77.29%. If one compares the computing time and the accuracies to be achieved, one notices the InceptionV3 model (red trace). It is the second best model and reaches the result in a medium range of about two and a half hours. The best model DenseNet201 needs more than twice as much time for its result. The choice for all further experiments is therefore the CNN model InceptionV3, because it achieves a good result within a comparatively short period of time.

4.3.4 Use of the transfer learning approach

What influence does the use of transfer learning have on accuracy? As described in the chapter "Transfer learning", the transfer learning approach immediately improves accuracy because the model already has recognition features that an unlearned network has yet to learn. The result looks like this on the data set `food-50`:

[54]GeForce 10 series, Wikipedia contributors, February 22, 2020, `https://en.wikipedia.org/wiki/GeForce_10_series`

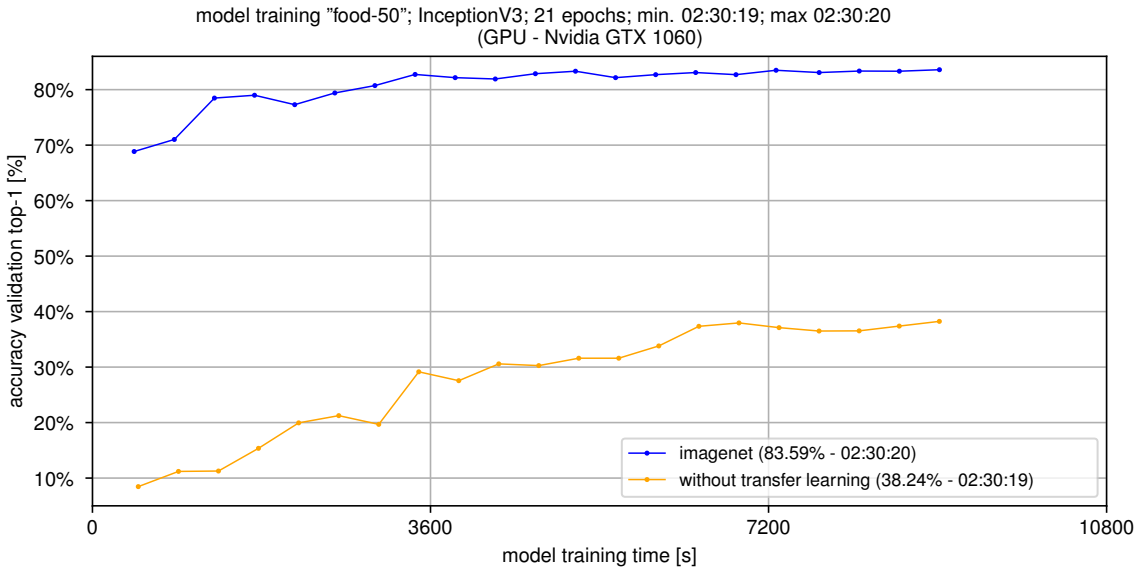

Figure 18: Overview of use of the transfer learning approach

As expected, the transfer learning model achieves a better result with the same computing time. In the first epoch the model without transfer learning approach reaches an accuracy value of 8.5% and increases to 58.8% in the 21 learning epochs. If one starts with a trained network, one achieves a better value of 68.9% already in the first epoch and can further increase this value to 83.6% in the 21st epoch. As a conclusion it should be mentioned that it is certainly possible to get the untrained network to this value, but one needs much more training epochs. Using transfer learning is a good idea in this case.

4.3.5 Influence of the number of trained layers on the accuracy

A pre-trained network has a decisive influence on the accuracy that can be achieved in a certain computing time. With increasing depth of the layers of a CNN model, the associated filters have learned more and more complex recognition features[55]: In the first layers, the filters learn basic shapes to recognize features such as edges and corners. The middle layers learn to recognize parts of objects. The last layers learn complete objects in different shapes and positions. The question that immediately comes up: Is it necessary to relearn the lower layers or is it possible to omit them to save computing time? This will be tested in the following experiment:

[55] Advanced Topics in Deep Convolutional Neural Networks, https://towardsdatascience.com, February 22, 2020, `https://towardsdatascience.com/advanced-topics-in-deep-convolutional-neural-networks-71ef1190522d`

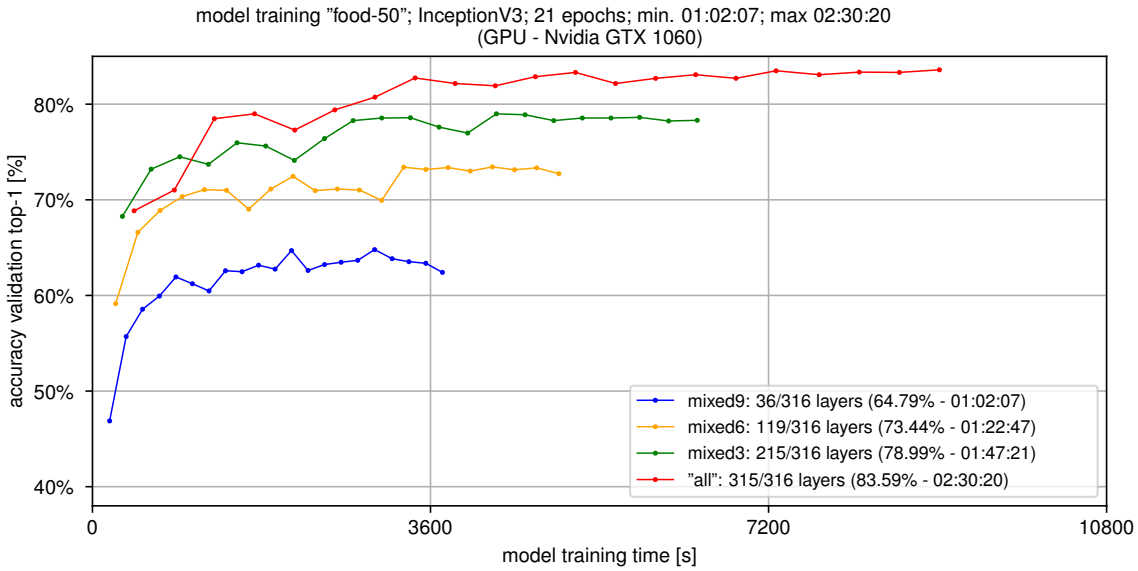

Figure 19: Overview of influence of the number of trained layers

The idea of not training the lowest layers is unfortunately not a good idea. One can save a lot of computing time if one only trains the last 36 levels (about one hour of computing time compared to two and a half hours if one trains all levels), but the accuracy of the model increases steadily with the training of additional layers. The model accuracy could be improved from 64.8% to 83.6% when training all levels. It is a good idea to invest in computing time to get good models.

4.3.6 Influence of different error optimizers

The optimizer in the training process is used to reduce the value of the loss function to reflect the value of the predictions as correctly as possible. The value of the loss function is the difference between the expected value and the estimated value of the function. Optimizers determine the value and adjust the error to minimize the loss function. The simplest approach is the gradient descent algorithm[56], which has been improved over time and further optimization algorithms have been developed, which sometimes work better and sometimes not so well according to their tasks and goals. The learning rate η determines the value of the weighting with which the error is corrected (see also chapter "Loss function"):

$$x_{n+1} = x_n - \eta \cdot \nabla F(x_n) = x_n - \eta \cdot L_r(\vartheta, \lambda),\ n \geq 0 \tag{16}$$

This area deals with the topic of optimizers and their parameters (especially the learning rate η) and what decisive influence they have on the accuracy of the model.

4.3.6.1 Comparison optimizer

In addition to the gradient descent algorithm, there are a lot of other methods which are intended to improve the descent to the optimum. For example, techniques are used which, together with the learning rate, prevent or limit jumping between the optimum and thus optimize convergence[57]. Below is a comparison of current optimization methods:

[56] Gradient descent, Wikipedia contributors, February 22, 2020, `https://en.wikipedia.org/wiki/Gradient_descent`

[57] A Look at Gradient Descent and RMSprop Optimizers, https://towardsdatascience.com, February 22, 2020, `https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b`

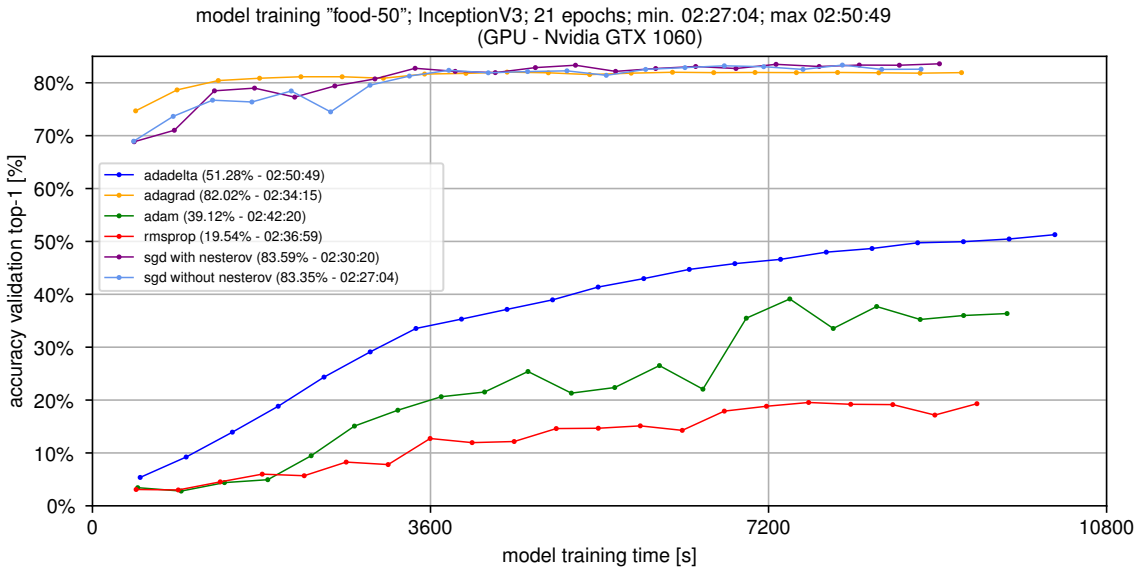

Figure 20: Overview of best optimizer

As one can see, the results are very different. While the gradient descent method with and without Nesterov and the Adagrad method provide good values (82.0% to 83.6% accuracy), the other methods do not look very useful at first sight. They converge very slowly: Adadelta, Adam and RMSprop. All procedures were performed with the recommended settings according to the Keras documentation[58]. A learning rate of $\eta = 0.001$ was chosen for all procedures.

4.3.6.2 Influence of the momentum and the Nesterov momentum

The two best methods from the previous chapter "Comparison optimizer" SGD with Nesterov and SGD without Nesterov will be examined here in more detail. The learning parameter η plays a decisive role. It is varied from 0.5 to the critical range of 0.98. The learning rate decreases by 50% after every seven epochs. The results will then be compared and discussed:

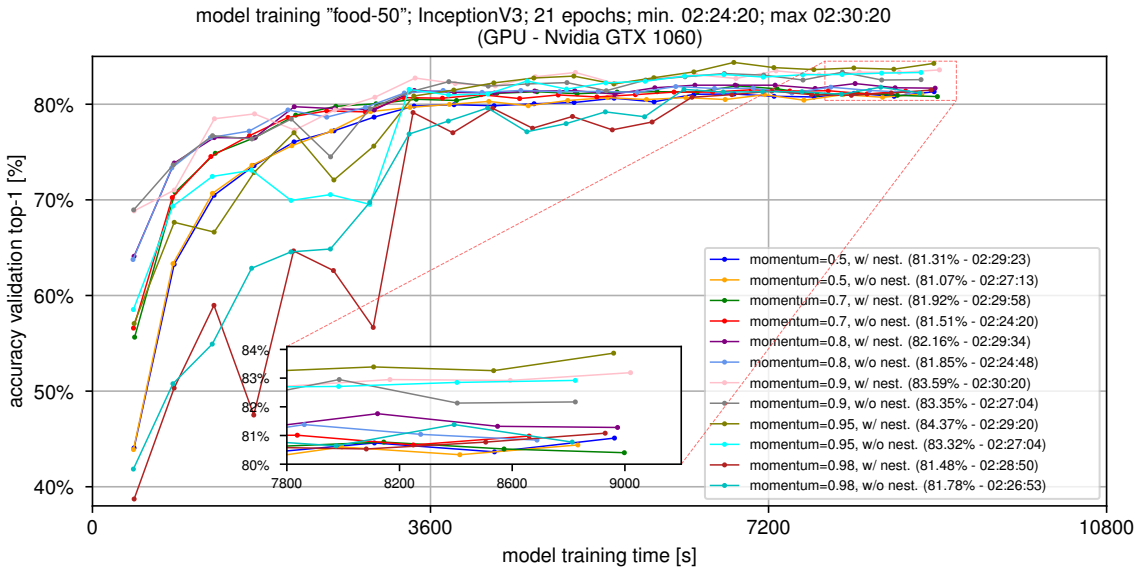

Figure 21: Overview momentum vs Nesterov momentum

[58] Usage of optimizers, https://keras.io, February 22, 2020, `https://keras.io/optimizers/`

The accuracy of both models with (w/ nest.) and without Nesterov (w/o nest.) increases steadily with increasing learning rate η until they reach their zenith at 83.35% and 84.37% with $\eta = 0,9$ (w/o nest.) and $\eta = 0,95$ (w/ nest.) respectively. It is noticeable that the smaller the learning rate η, the accuracy increases more evenly, but the "maximum" at 21 epochs is not reached. With increasing learning rate, e.g. at the maximum of $\eta = 0,98$ for the Nesterov model (brown trace), the model accuracies jump and can sometimes run completely in the negative direction with another epoch. With the best model (olive trace, w/ nest.) exactly the same happens, but in the end the best results are achieved.

4.3.6.3 Influence of a dynamic learning rate on accuracy (scheduling)

The learning rate η indicates how much of the error is returned to the model (step size). After a certain number of learning epochs, the model accuracy does not increase any more but jumps around a value (see green trace in the following figure), because the optimum cannot be achieved due to a too large step size in error correction. It is therefore a good idea to adjust and reduce the step size step by step over the epochs. For this purpose the learning rate η is adjusted after a certain number of epochs $\mathcal{E}$ with the momentum β. The value for η_1 corresponds to the initial value for the learning rate:

$$\eta_{m \to next} = \beta \cdot \eta_m, \; m \in [1 + 0 \cdot \mathcal{E}, 1 + 1 \cdot \mathcal{E}, 1 + 2 \cdot \mathcal{E}, \dots] \tag{17}$$

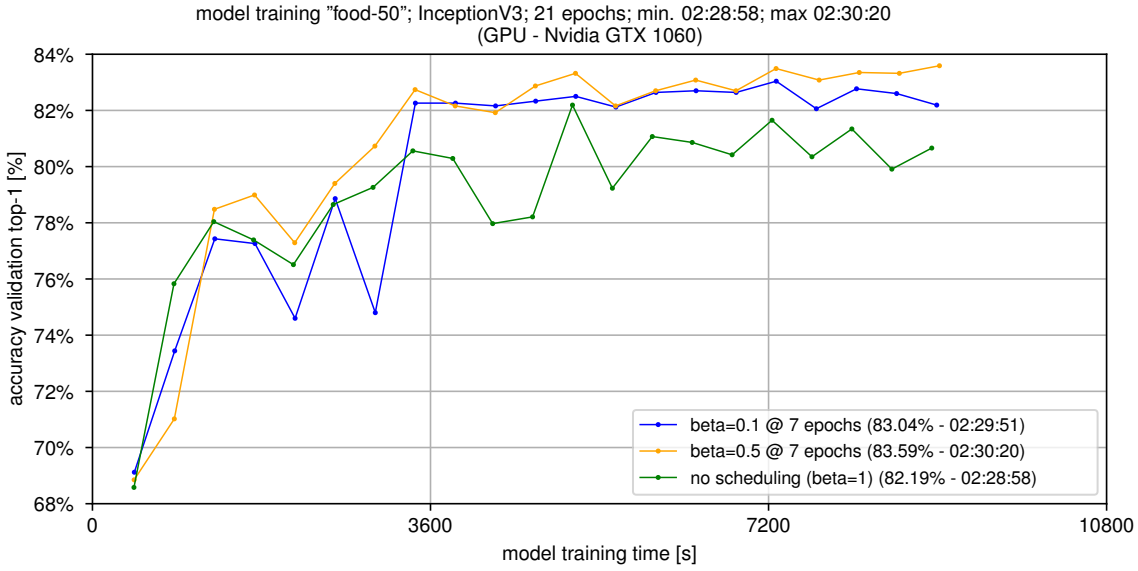

Figure 22: Overview of a dynamic learning rate on accuracy

As suspected, a reduction of the learning rate over time improves the possible model accuracy. However, it must not be reduced too much, as this would limit the further learning possibilities too much. While a momentum of $\beta = 0.5$ still provides a model accuracy of 83.6%, this decreases to 83.0% at $\beta = 0.1$. A static learning rate does not improve the learning ability from about the 8th epoch onwards. It then jumps around the value of 81% and even seems to worsen a little. Momentum belongs to the group of hyperparameters[59] and must be determined experimentally. An empirical value can be used as a starting value.

[59] Hyperparameter optimization, Wikipedia contributors, February 22, 2020, `https://en.wikipedia.org/wiki/Hyperparameter_optimization`

4.3.7 Different batch sizes

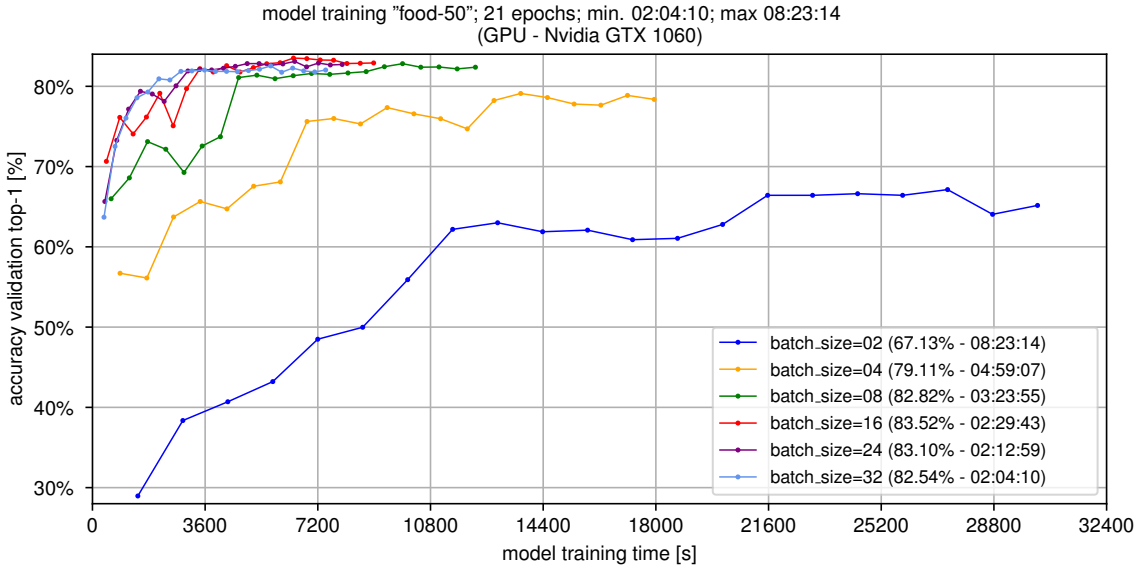

Figure 23: Overview of the influence of a different batch size

4.3.8 Different activation functions

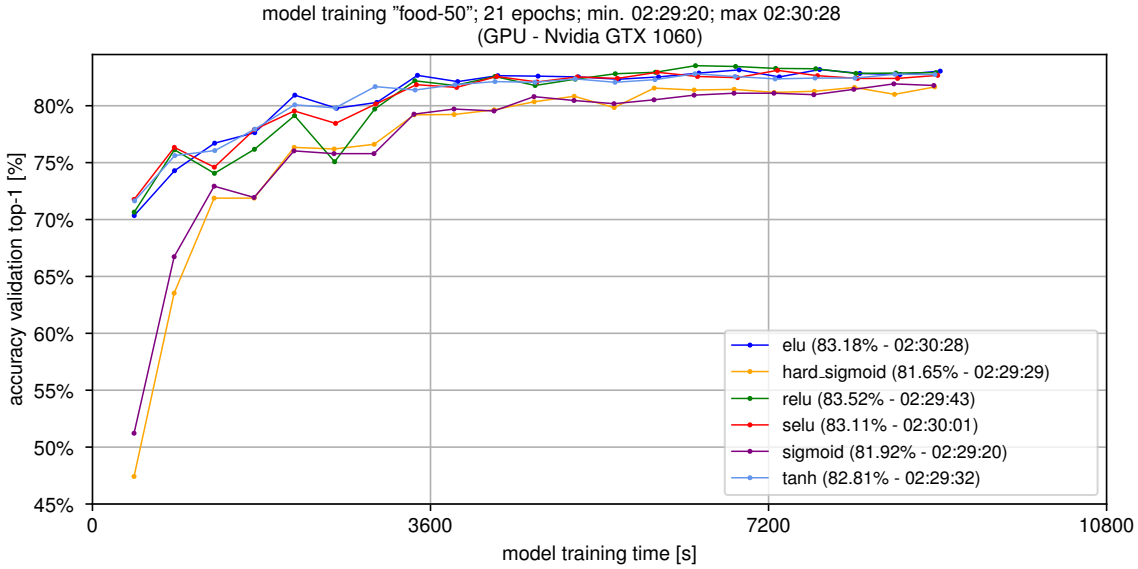

Figure 24: Overview of the influence of difference activation functions

4.3.9 Different number of learned epochs

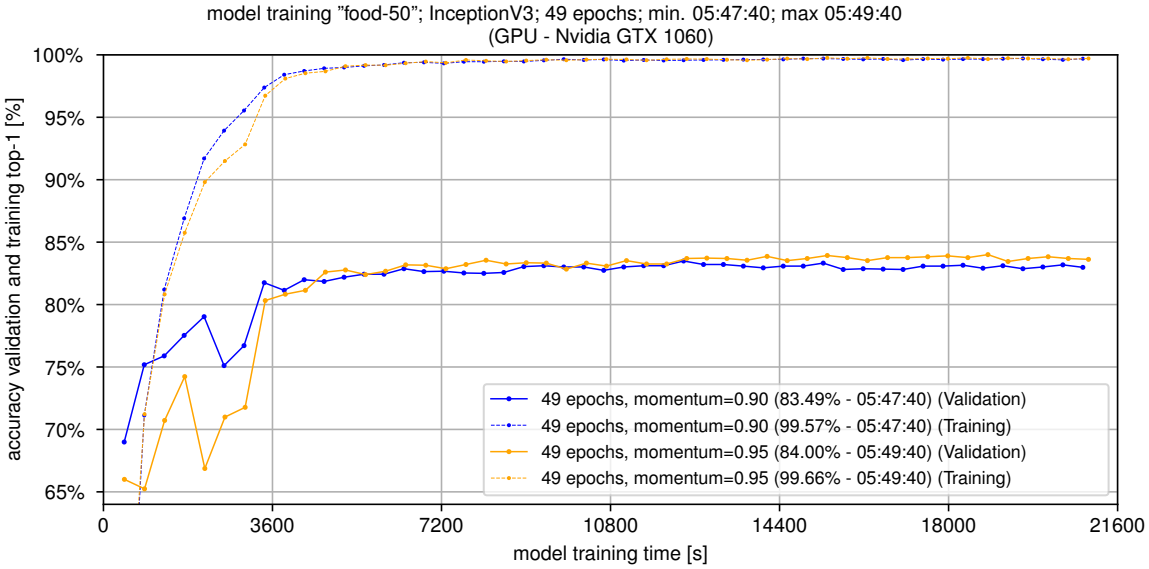

Figure 25: Overview of the influence of a longer training period with more epochs

Here are a few words about overfitting.

4.3.10 Influence of dropout

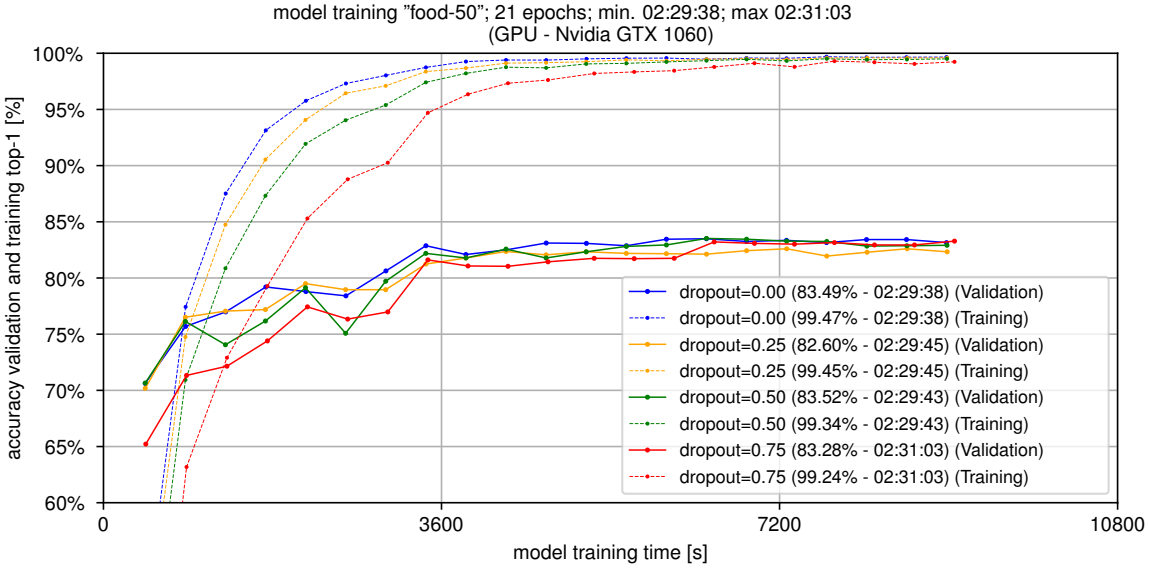

Figure 26: Overview of the dropout parameter

Here are a few words about overfitting.

4.4 Optimization process

In addition to hyperparameters (or tuning parameters), there are a number of other options that influence model accuracy. It should be mentioned, for example, the problem in chapter "Influence of the number of trained layers on the accuracy", where the model accuracy could not be further

investigated with more than the existing 12000 images due to missing data, although the accuracy seemed to increase even more. Or the problem with the unbalanced data set. Wouldn't it be an idea to have the same amount of data in all classes without having to discard data or get additional data? What about other classification ideas? Currently, one model is used for all classes. Do hierarchies make sense? All these things will be examined in the following chapters.

4.4.1 Preamble

As in chapter "Preamble" for Model Validation, the same settings are used as the default setup, unless otherwise specified. InceptionV3 is used as CNN. The data set and also the division into training and validation data set corresponds to the procedure of the previous chapters.

4.4.2 Data augmentation

As described in chapter "Data augmentation" the main task of data augmentation is to generate new training data from the existing data. The training data set is artificially augmented in this way. This technique is one of the regularization techniques, as it can counteract overfitting.[60] Below is an example data set in which the first original image is rotated, mirrored, distorted and the data set is enlarged from one image to a total number of 12 images:

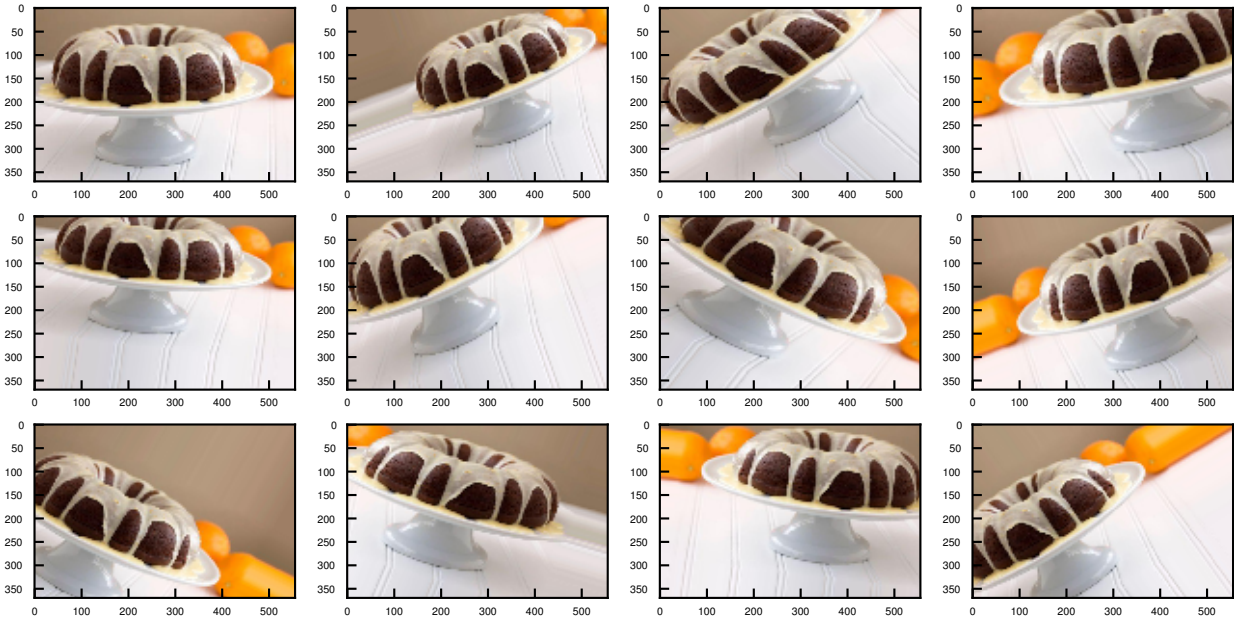

Figure 27: Example of data augmentation (The first image is the original image)

Using this technique, the existing `food-50` dataset is now adjusted until each class contains a total number of 1000 images and the dataset is balanced overall. Classes with few data will receive more artificially adjusted data, classes with many data will receive less. The result is that the original 11913 images have now become a total number of 50000 images and the data set has grown from 765 MB to 1.78 GB. The validation data set remains unchanged in order not to falsify the test result. The evaluation diagram is shown below:

[60] Aurélien Géron. "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Data Augmentation, pp. 311–312. ISBN: 9783960090618.

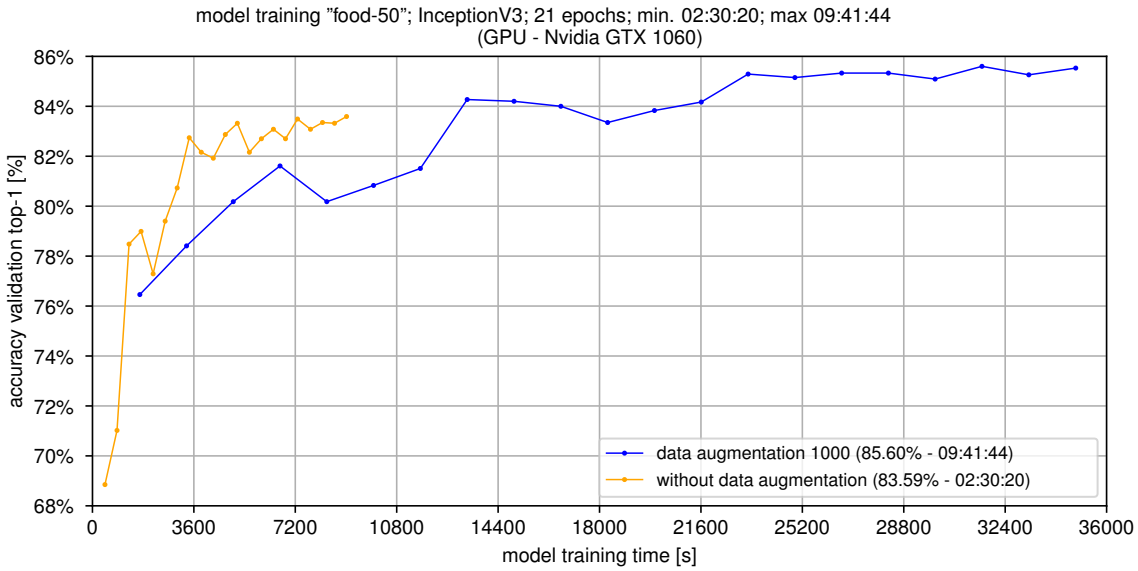

Figure 28: Comparison of data augmentation

The model accuracy increases from 83.6% to 85.6%. An overall increase of 2%. Note: The validation data set is still the same! However, this also means an increase in the required computing time. While the calculation of the data set without data augmentation takes only two and a half hours, the new data set takes four times as long. Conclusion: Data Augmentation is associated with improved accuracy, but also requires more computing time in the same breath. If computing time is not important, this is one way to improve the accuracy a little.

4.4.3 Enrichment of the data set from other data sources

...

4.4.4 Hierarchical classification

By using a single model for all classes, previous classifiers have been trained to minimize the loss of the class output vector. Each class used so far has the same rank in both training and classification. The prediction of "Pizza" costs the same as the prediction of "Martini".

The human ability to classify objects does not only work on one level. Categories will naturally overlap and have a hierarchical structure. For example, a human will classify a picture under "pizza", "tuna pizza" or even "fast food", which is correct from this point of view. Depending on the classification, there will only be a "loss of information". However, a person will not mistake a "pizza" as a "Martini", which is more likely to be classified as a "drink" or "cocktail".[61]

In order to find out whether a pre-classification improves the result, the classes are divided into groups. The idea is that no longer one model predicts all classes, but rather the images are pre-sorted into a group. The corresponding group then carries out the actual classification. For this purpose, a principal component analysis is performed to analyze the similarity of the classes from the `food-50` training data set. The result will then be used to create groups. In preparation, a model will again be trained for all classes: InceptionV3, 21 epochs, batch size 16, learning rate $\eta = 0.001$ decreasing by factor 0.5 every seven epochs. Using this model, a prediction is made of all training

[61] Eleanor Rosch et al. "Basic objects in natural categories". In: *Cognitive psychology: Key readings* 448 (2004).

and validation data and the probability vector is determined (One Hot Encoding):

$$\lambda_{class_m} = \begin{pmatrix} \hat{p}_1 \\ \hat{p}_2 \\ \vdots \\ \hat{p}_{50} \end{pmatrix} \quad \Big| \quad \sum_{i=1}^{50} \hat{p}_i = 1 \tag{18}$$

From these probability vectors, an average vector of all n elements is determined for each class, which really belong to the respective class. Thus, for this 50 class model, one obtains 50 class vectors with the dimension 50:

$$\forall\ 50\ classes : \overline{\lambda}_{class_m} = \frac{\sum_{i=1}^{n} \lambda_{class_m}}{n} \tag{19}$$

Principal component analysis is used to transform these multidimensional vectors into a two-dimensional space in order to display them visually:

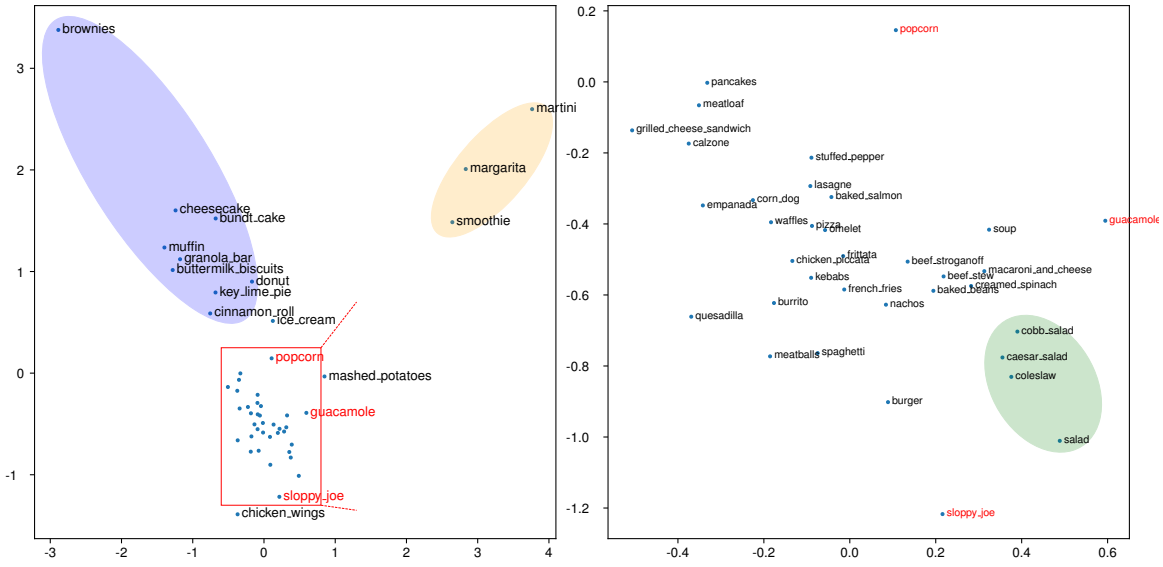

Figure 29: Principal component analysis of the `food-50` model. The x and y axis have no significance. They only stand for the room and the similarity between the classes.

As expected, one can now immediately recognize similar classes: drinks (orange), cakes (blue) and salads (green).

4.4.4.1 k-means clustering

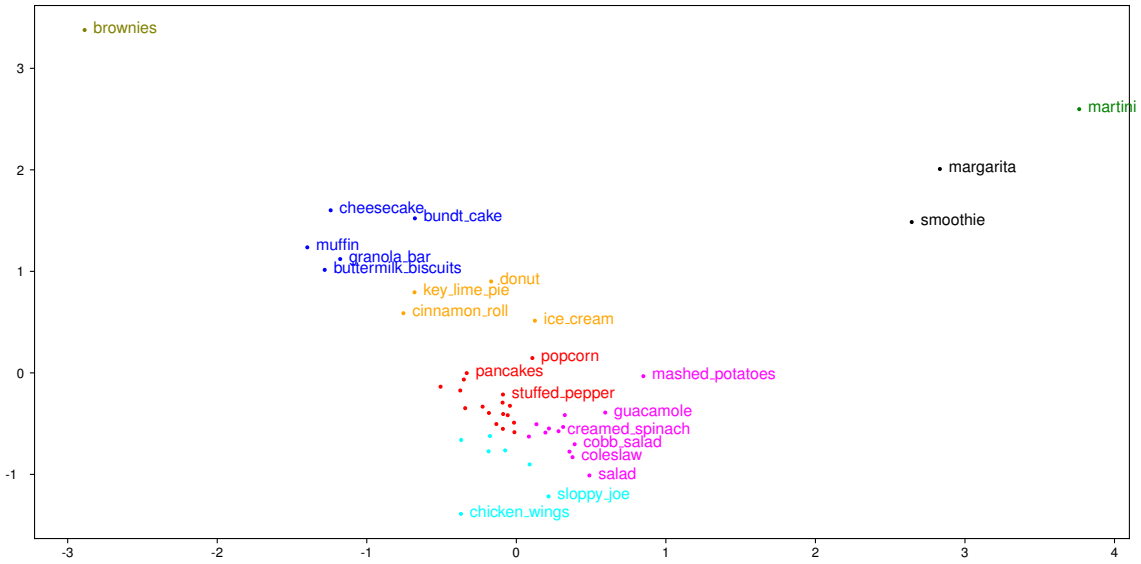

Figure 30: Grouped classes with k-means

Group	Classes
Group 0	baked_salmon, calzone, chicken_piccata, corn_dog, empanada, french_fries, frittata, grilled_cheese_sandwich, kebabs, lasagne, meatloaf, omelet, pancakes, pizza, popcorn, stuffed_pepper, waffles
Group 1	margarita, smoothie
Group 2	bundt_cake, buttermilk_biscuits, cheesecake, granola_bar, muffin
Group 3	brownies
Group 4	martini
Group 5	cinnamon_roll, donut, ice_cream, key_lime_pie
Group 6	baked_beans, beef_stew, beef_stroganoff, caesar_salad, cobb_salad, coleslaw, creamed_spinach, guacamole, macaroni_and_cheese, mashed_potatoes, nachos, salad, soup
Group 7	burger, burrito, chicken_wings, meatballs, quesadilla, sloppy_joe, spaghetti

Table 4: Grouped classes with k-means

4.4.4.2 Agglomerative hierarchical clustering

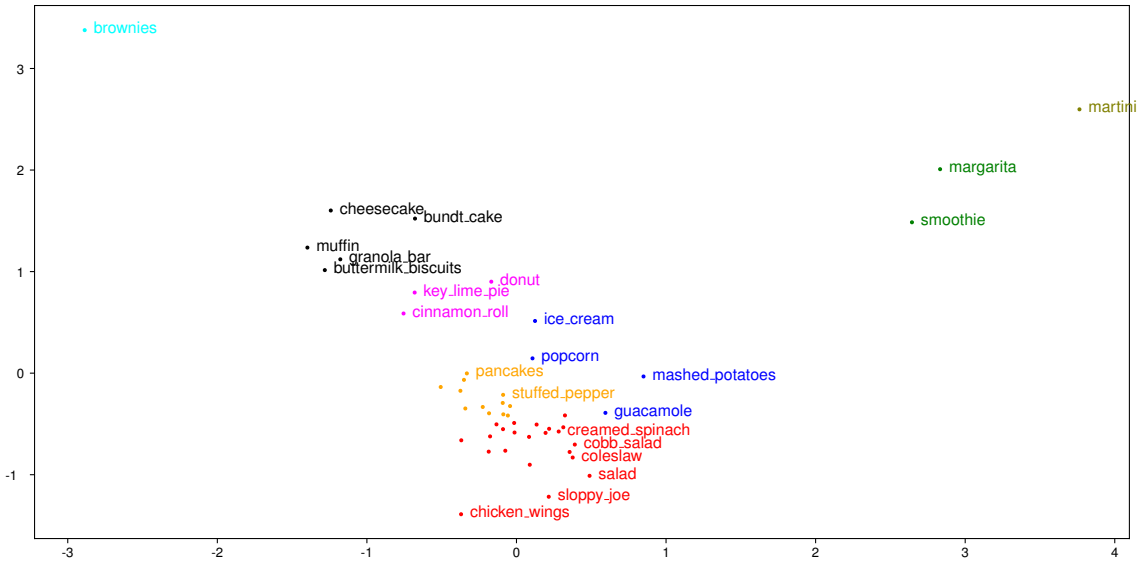

Figure 31: Grouped classes with agglomerative hierarchical clustering

Group	Classes
Group 0	baked_beans, beef_stew, beef_stroganoff, burger, burrito, caesar_salad, chicken_piccata, chicken_wings, cobb_salad, coleslaw, creamed_spinach, french_fries, frittata, kebabs, macaroni_and_cheese, meatballs, nachos, quesadilla, salad, sloppy_joe, soup, spaghetti
Group 1	bundt_cake, buttermilk_biscuits, cheesecake, granola_bar, muffin
Group 2	guacamole, ice_cream, mashed_potatoes, popcorn
Group 3	martini
Group 4	margarita, smoothie
Group 5	baked_salmon, calzone, corn_dog, empanada, grilled_cheese_sandwich, lasagne, meatloaf, omelet, pancakes, pizza, stuffed_pepper, waffles
Group 6	cinnamon_roll, donut, key_lime_pie
Group 7	brownies

Table 5: Grouped classes with agglomerative hierarchical clustering

4.4.5 Binary classifiers

...

5 Summary and outlook

What's the outcome? What else is possible? How can this work be continued? In here!

- What are the best parameters?
- What is the best approach?
- What else should be investigated?
  - More precise analysis of incorrectly classified data: Is there any room for improvement here?
- What else is missing in this thesis?
- Questions asked in this thesis:
  - Is it a good idea to implement your own implementation or is the step to the software tools of company giants like Microsoft, Google and Co. unavoidable?

List of acronyms

CNN convolutional neural network

List of literature

Backpropagation. en. Page Version ID: 939314095. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=939314095 (visited on 02/25/2020).

Banko, Michele and Eric Brill. "Scaling to very very large corpora for natural language disambiguation". In: *Proceedings of the 39th annual meeting on association for computational linguistics*. Association for Computational Linguistics. https://www.aclweb.org/anthology/P01-1005.pdf, 2001, pp. 26–33.

Chollet, Francois. "Building powerful image classification models using very little data". In: *Keras Blog* (2016).

Convolutional neural network. en. Page Version ID: 942501792. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=942501792 (visited on 02/25/2020).

deeplizard. *Convolutional Neural Networks (CNNs) explained*. Youtube. 2017. URL: https://www.youtube.com/watch?v=YRhxdVk{_}sIs.

Deng, Jia et al. "What does classifying more than 10,000 image categories tell us?" In: *European conference on computer vision*. Springer. http://vision.stanford.edu/pdf/DengBergLiFei-Fei_ECCV2010.pdf, 2010, pp. 71–84.

Fatahalian, Kayvon, Jeremy Sugerman, and Pat Hanrahan. "Understanding the efficiency of GPU algorithms for matrix-matrix multiplication". In: *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. https://graphics.stanford.edu/papers/gpumatrixmult/gpumatrixmult.pd 2004, pp. 133–137.

Géron, Aurélien. "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Konfusionsmatrix, pp. 86–88. ISBN: 9783960090618.

— "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Entscheidungsgrenzen, pp. 138–140. ISBN: 9783960090618.

— "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Data Augmentation, pp. 311–312. ISBN: 9783960090618.

— *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme*. O'Reilly Verlag, 2017, pp. 8–14. ISBN: 9783960090618.

Halevy, Alon, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf, 2012, pp. 1097–1105.

Least mean squares filter. en. Page Version ID: 941899198. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Least_mean_squares_filter&oldid=941899198 (visited on 02/25/2020).

Osinga, Douwe. *Deep Learning Kochbuch: Praxisrezepte für einen schnellen Einstieg*. O'Reilly Verlag, 2019, pp. 19–26. ISBN: 9783960090977.

Overfitting. en. Page Version ID: 942053730. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=942053730 (visited on 02/25/2020).

Perceptron. en. Page Version ID: 942271496. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Perceptron&oldid=942271496 (visited on 02/25/2020).

Rosch, Eleanor et al. "Basic objects in natural categories". In: *Cognitive psychology: Key readings* 448 (2004).

Sagar, Abhinav. *Deep Learning for Image Classification with Less Data*. en. Library Catalog: towardsdatascience.com. Nov. 2019. URL: https://towardsdatascience.com/deep-learning-for-image-classification-with-less-data-90e5df0a7b8e (visited on 02/25/2020).

Sun, Yi, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Sun_Deep_Learning_Face_2014_CVPR_paper 2014, pp. 1891–1898.

Szeliski, Richard. "Computer Vision: Algorithms and Applications". en. In: (), p. 979.

The Most Popular Language For Machine Learning Is ... (IT Best Kept Secret Is Optimization). en. CT904. Library Catalog: www.ibm.com. Aug. 2015. URL: www.ibm.com/developerworks/community/blogs/jfp/entry/what_language_is_best_for_machine_learning_and_data_science (visited on 02/25/2020).

Vapnik–Chervonenkis dimension. en. Page Version ID: 942482212. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Vapnik%E2%80%93Chervonenkis_dimension&oldid=942482212 (visited on 02/25/2020).

Warden, Pete. *How many images do you need to train a neural network?* en. Library Catalog: petewarden.com. Dec. 2017. URL: https://petewarden.com/2017/12/14/how-many-images-do-you-need-to-train-a-neural-network/ (visited on 02/25/2020).

What is the minimum sample size required to train a Deep Learning model - CNN? en. Library Catalog: www.researchgate.net. URL: https://www.researchgate.net/post/What_is_the_minimum_sample_size_required_to_train_a_Deep_Learning_model-CNN (visited on 02/25/2020).

List of links

- Deep learning unbalanced training data?
  - https://towardsdatascience.com/deep-learning-unbalanced-training-data-solve-it-like-this-6c528e9efea6
- Data Augmentation
  - https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/
- Stop Feeding Garbage To Your Model! — The 6 biggest mistakes with datasets and how to avoid them.
  - https://hackernoon.com/stop-feeding-garbage-to-your-model-the-6-biggest-mistakes-with-datasets-and-how-to-avoid-them-3cb7532ad3b7
  - https://towardsdatascience.com/advanced-topics-in-deep-convolutional-neural-networks-71ef1190522d

Acknowledgement

The hardest part for me is this acknowledgment. Because it doesn't make any difference to me who comes first or last. Each person has done his or her part and is equally important for this thesis and me. Nevertheless I try: At this point I would like to thank all the people who have helped me to make this work a success. First and foremost: Prof. Dr. Karl Heinz Hoffmann, who encouraged me to resume my studies after I had broken off my studies more than 15 years ago and to catch up on my degree. In second place the company "ressourcenmangel dresden GmbH", which gave me the freedom to catch up my studies besides my job. In third place David Urbansky from the semknox company, a great expert in the field of sentiment analysis and classifications of all kinds of things. He was the one who accompanied me skin to skin the whole time and supported me with helpful tips and ideas. In fourth place Prof. Dr. Angela Thränhardt Professor for "Theoretical Physics - Simulation of New Materials" at the Chemnitz University of Technology, which enabled me to tackle a somewhat unrelated area in this work. In fifth place was my family, who was permanently at my side and supported me with this topic. And last but not least... TODO

Declaration

I hereby declare that the work presented in this thesis is solely my work and that to the best of my knowledge this work is original, except where indicated by references to other authors. No part of this work has been submitted for any other degree or diploma.

Signature : *Place, Date* :