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:	March 9, 2020
First assessor:	Prof. Dr. Angela Thränhardt
Second assessor:	Dr. David Urbansky

Abstract

Artificial neural networks have achieved an error rate of less than 5% in the ImageNet software competitions in recent years and are well suited to find patterns in data. Therefore, these networks are first trained with known data sets and adjusted accordingly. Data sets are usually very expensive to obtain and must therefore be selected and used with care. The choice of the right model for finding patterns in data depends on the current problem. The training of these models is influenced by many hyperparameters (HPs), which are determined in advance. The purpose is to train a model with the right choice of HPs, which allows good predictions on unknown data (data that the model has never seen before). In this thesis the creation of models for image classification by machine learning (ML) is discussed and the model accuracies are compared. The main purpose is to find optimal HPs and techniques for classification, which also allows to create an optimal model from a small training data set.

Keywords

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

Kurzfassung

Künstliche neuronale Netze haben es in den letzen Jahren bei den ImageNet Software-Wettbewerben auf unter 5% Fehlerrate geschaft und eignen sich gut dafür Muster in Daten zu finden. Hierzu werden diese Netze vorher mit bekannten Datensätzen trainiert und entsprechend angepasst. Datensätze sind meist sehr teuer in der Beschaffung und müssen deshalb mit Bedacht und entsprechender Sorgfalt ausgewählt und eingesetzt werden. Die Wahl des richtigen Modells für das Finden von Mustern in Daten ist abhängig vom aktuell vorliegendem Problem. Das Training dieser Modelle wird durch viele Hyperparameter beeinflusst, welche vorher festgelegt werden. Das Ziel ist es mit der richtigen Wahl der Hyperparameter ein Modell zu trainieren, welches gute Vorhersagen auf noch unbekannte Daten ermöglicht (Daten, die das Modell zuvor nie gesehen hat). 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 ist es, optimale Hyperparameter und Techniken für die Klassifikation zu finden, die es auch ermöglichen, mit einem kleinen Trainingsdatenset ein optimales Modell zu erstellen.

Schlüsselwörter

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

Contents

List of figures

List of tables

1 Introduction

This thesis deals with the creation of models for image classification using machine learning (ML). Many variable hyperparameters (HPs) will have a decisive influence on the accuracy of the model when creating the models and are compared here in detail. In addition to the accuracy of the models, the computing time required to create the models also plays a role 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 significantly positive influence on the result. New and more complex convolutional neural networks (CNNs) are more successful in model accuracy than older and smaller models. I also assume that HP settings can only change the model accuracy to a limited point and that more attention must be given to the optimization processes used in chapter "Optimization process". Chapter "Model validation" deals with this hypothesis and shows that the logical consideration in this case is only partially correct.

1.1 Insufficient amount of data

If one gives a person a doughnut and explain that it is a doughnut, then after some repetition one is able to recognize any kind of doughnut in the future. With ML this problem is a bit more complex. As with most ML methods, a large amount of data is required. The amount of required data depends on the current problem. Especially when one is dealing with many classes to be predicted, experience shows that the amount of data increases. A paper from 2012 with the title "ImageNet Classification with Deep Convolutional Neural Networks" describes how to train a model with 1.2 million high-resolution images.[1] The model could recognize 1,000 objects (1,000 classes), was trained with 1,000 images per class and won the top 5 test error rate with 15.3% in 2012 at the annual ImageNet competition. Other articles dealing with this topic confirm this number of images per class.[2,3]

Depending on the number of classes to be trained, one can quickly reach a required data set, which consists of many Gbytes of data. With transfer learning (TL) it is possible to reduce this number a little bit, but the problem of the large amount of data remains. A journal publication,[4] written by an author affiliated with Microsoft, 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.*"

Another article only a few years later also addresses this issue.[5] This article referred to data that learn from texts where usually only small or medium sized data sets are available. To improve the efficiency in this case, the approach referred to use data more efficiently and to optimize the algorithms:

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

[2] 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).

[3] 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).

[4] 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. 2001, pp. 26–33. URL: https://www.aclweb.org/anthology/P01-1005.pdf.

[5] Alon Halevy, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE Intelligent Systems* 24.2 (2009), pp. 8–12. URL: https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/35179.pdf.

"Choose a representation that can use unsupervised learning on unlabeled data, which is so much more plentiful than labeled data."

Whatever the approach is to train and use data. Depending on the problem, the main focus can be on the amount of required data, the type of data processing or perhaps a combination of both.

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 (as shown in Figure 1).

Figure 1: Cat and dog comparison

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 data sets and the automated creation of models was a new step into a new approach. The ever increasing amount of available computing time plays as much a decisive role in the success as parallel computations on several GPUs, the development of more efficient algorithms and methods such as data augmentation and the establishment of online databases with large amounts of labeled data (ImageNet).[6] Meanwhile, image recognition has become a widespread application area of ML. So-called "CNNs"[7] or "ConvNets" are often used for images.

The image classification algorithm accepts an image as input and classifies it into one of the possible output categories. The approach to learn knowledge from existing data is called machine learning (ML).[8] In combination with artificial neural networks (ANNs) it is more precisely called deep learning (DL).[9] Various convolutional neural networks (CNNs), 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.[10] 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 thesis here is mainly concerned with supervised learning, in which a mathematical model is trained based on existing known data sets. The purpose of the trained model is to make the best possible predictions even for unknown images. Known data sets are usually created

[6] George Seif. *Deep Learning for Image Recognition: why it's challenging, where we've been, and what's next*. en. Library Catalog: towardsdatascience.com. May 2019. URL: https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef (visited on 02/28/2020).

[7] *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).

[8] *Machine learning*. en. Page Version ID: 942989288. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=942989288 (visited on 02/28/2020).

[9] *Deep learning*. en. Page Version ID: 942561541. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=942561541 (visited on 02/28/2020).

[10] *ImageNet*. en. Page Version ID: 929993952. Dec. 2019. URL: https://en.wikipedia.org/w/index.php?title=ImageNet&oldid=929993952 (visited on 02/28/2020).

manually, 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.[11] Common to all techniques is 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).

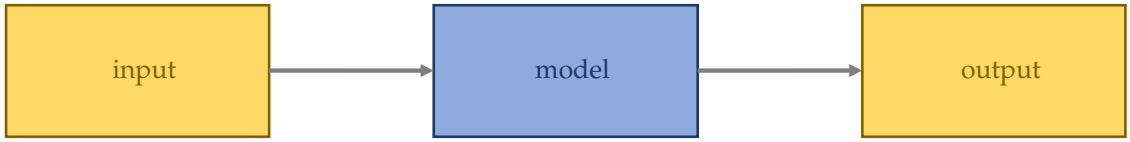

Figure 2: Deductive approach (the shown model is a white-box model)

2.1.2 Inductive approach

The inductive approach takes a different approach to classifying images. The purpose is not to specify a rule, but to learn a rule (model) automatically from already known individual objects.[12] A model is usually a complex function and a mathematical representation of a space, in which individual objects with their properties can be mapped and separated. This measure of the capacity that can be learned by this complex function is called Vapnik–Chervonenkis dimension (VCD).[13] 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 (Figure 3). The procedure described here is mostly used for any kind of supervised learning (see chapter "Supervised learning") and is a part of ML (see chapter "Machine learning").

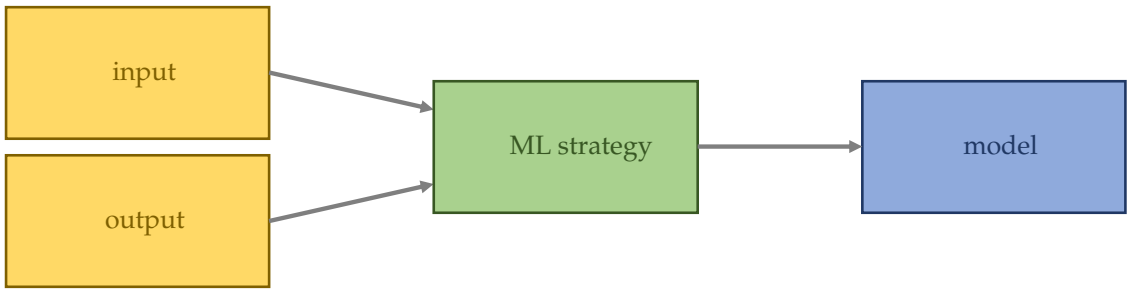

Figure 3: Inductive approach (the shown model is a black-box model)

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

[12] Ramon Lopez De Mantaras and Eva Armengol. "Machine learning from examples: Inductive and Lazy methods". In: *Data & Knowledge Engineering* 25.1-2 (1998), pp. 99–123. URL: https://www.sciencedirect.com/science/article/pii/S0169023X97000530/pdf.

[13] *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).

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.[14]

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, these could be classes like burgers (as shown in Figure 4a), doughnuts (Figure 4b) or pizza (Figure 4c), etc. For these classes, one now needs a large number of images. Ideally, this data should reflect reality as far 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 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").

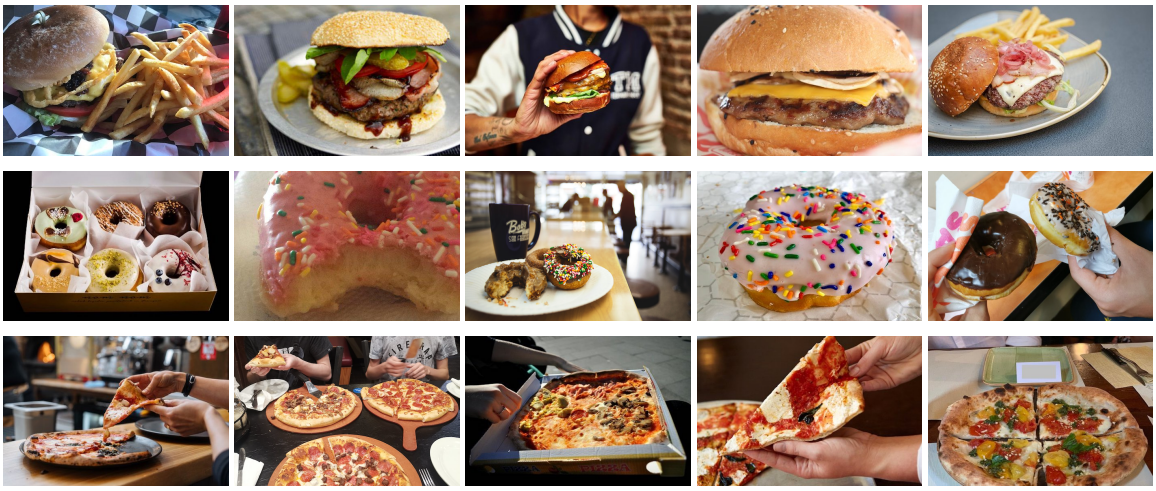

Figure 4: (a) Example of pictures of a burger class (top), (b) Example pictures of a doughnut class (middle), (c) Example of pictures of a burger class (bottom)

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 (Flickr[15]), search engines (Google[16], Bing[17]) or reviews (TripAdvisor), in which images appear and can be associated with the text ("This here is the best doughnut I've ever eaten."). A creative approach is an advantage here.

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, validation and test 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 (NNs) will not generalize to

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

[15] *Flickr images - Doughnut*. URL: https://www.flickr.com/search/?text=doughnut

[16] *Google images - Doughnut*. URL: https://www.google.de/search?q=doughnut&tbm=isch

[17] *Bing images - Doughnut*. URL: https://www.bing.com/images/search?q=doughnut

some extent, but will learn by heart (overfitting[18]). 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. 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.[19] 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% training data and 20% validation data is used according to the Pareto principle, unless otherwise stated.

2.2 Classification metrics and confusion matrix

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

2.2.1 Loss function

For the classification of an image ϑ a NN is usually used as a model. The function for this model is specified with $\boldsymbol{M}(\vartheta)$. The last layer of this NN-based classifier is mostly a softmax function.[20] The model assigns a vector $\boldsymbol{\lambda}$ of size n to an element ϑ and is the predicted value of this element. The size n corresponds to the number of classes to be distinguished.[21] Each individual value p of $\boldsymbol{\lambda}$ corresponds to the probability that it is class $class_n$ (Equation (1)).

$$\boldsymbol{M}(\vartheta) = \boldsymbol{\lambda} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{pmatrix} \quad \Big| \quad \sum_{i=1}^{n} p_i = 1 \tag{1}$$

The expected value of the parameter function $\boldsymbol{g}(\vartheta)$ and thus of the current element ϑ and its expected labelled class is returned as a so-called "one-hot vector". The vector element for the expected class has the value 1, all others have the value 0. This is also called one hot encoding (using the example of an element of $class_2$):

$$\boldsymbol{g}(\vartheta_{class_2}) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2}$$

The loss function[22] assigns a loss to each prediction $\boldsymbol{\lambda}$, which results from the comparison with the true value of the parameter function $\boldsymbol{g}(\vartheta_{class})$. Backpropagation (see chapter "Backpropagation") calculates for each element ϑ the gradient of the loss function which results with $\boldsymbol{M}(\vartheta)$. The gradient

[18] *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).

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

[20] *Softmax function*. en. Page Version ID: 928536872. Nov. 2019. URL: https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=928536872 (visited on 03/03/2020).

[21] 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.

[22] *Verlustfunktion (Statistik)*. de. Page Version ID: 177095272. May 2018. URL: https://de.wikipedia.org/w/index.php?title=Verlustfunktion_(Statistik)&oldid=177095272 (visited on 02/26/2020).

is used to adapt the parameters of function $\boldsymbol{M}(\vartheta)$ to reduce the prediction error. As an example, a typical loss function for an r-dimensional space for one element is shown in Equation (3).

$$L_r(\vartheta, \boldsymbol{\lambda}) := \|\boldsymbol{\lambda} - \boldsymbol{g}(\vartheta)\|^r \tag{3}$$

$\boldsymbol{\lambda}$ represents the estimated value and $g(\vartheta)$ the parameter function which returns the actual value for ϑ. The loss function is not only determined for one element. It is necessary to consider all elements together to create a model that generalises. The average loss on the entire data set with k elements is thus:

$$\hat{L} = \frac{1}{k} \sum_{i=1}^{k} L_r(\vartheta_i, \boldsymbol{\lambda}_i) \tag{4}$$

2.2.2 Confusion matrix

The confusion matrix is a special quadratic matrix in the field of ML that allows the visualization of the performance of a predictive model (Figure 5). 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). A special class $class_1$ is considered: True positive (TP) corresponds to the value of the correctly determined predictions of the class currently under consideration. False negative (FN) indicates how often other classes are detected as the class currently under consideration. False positive (FP) indicates how often classes other than the currently viewed class were determined. True negative (TN) in turn indicates that classes other than the current class were correctly predicted as the other classes.[23]

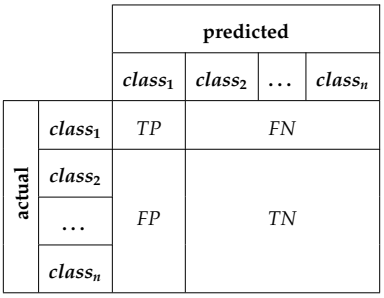

Figure 5: Confusion matrix

Finally, the confusion matrix has the following structure (Equation (5)). $P_{i,j}$ means $class_i$ was predicted although it should have been class $class_j$. All values on the main diagonal were correctly predicted ($P_{i,i}$). $\#P_{i,j}$ in turn is the total number that $class_i$ was recognized, although it should be $class_j$. If the model has a model accuracy of 100%, there are values greater than 0 only on the main diagonal and all other non-diagonal elements are 0 (Equation (6)).

$$\boldsymbol{M}_{confusion} = \begin{bmatrix} \#P_{1,1} & \dots & \#P_{n,1} \\ \vdots & \ddots & \vdots \\ \#P_{1,n} & \dots & \#P_{n,n} \end{bmatrix} = (m_{nn}) \tag{5}$$

$$\boldsymbol{M}_{confusion} = \begin{bmatrix} \#P_{1,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \#P_{n,n} \end{bmatrix} = (m_{nn}) \tag{6}$$

[23] *Confusion matrix*. en. Page Version ID: 940280604. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=940280604 (visited on 02/28/2020).

2.2.3 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.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\sum_{i,j=1}^{n} m_{ij}}{\sum_{i=1}^{n} \sum_{j=1}^{n} m_{ij}} = \frac{Correct_{all}}{CorrectPossible_{all}} \tag{7}$$

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:

$$Accuracy_{top-5} = \frac{CorrectWithinTheBest5Classes_{all}}{CorrectPossible_{all}} \tag{8}$$

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 data set.[24] Example: Suppose one has a model that always predicts the class $class_1$. The class $class_1$ consists of 9,990 elements and from the other classes $class_2$ to $class_n$ one has exactly 10. Then the confusion matrix looks like in the following figure (Figure 6).

<table>
<tr><td></td><td></td><td colspan="4">predicted</td></tr>
<tr><td></td><td></td><td>class₁</td><td>class₂</td><td>...</td><td>class_n</td></tr>
<tr><td rowspan="5">actual</td><td>class₁</td><td>TP = 9990</td><td colspan="3">FN = 0</td></tr>
<tr><td>class₂</td><td rowspan="3">FP = 10</td><td colspan="3" rowspan="3">TN = 0</td></tr>
<tr><td>...</td></tr>
<tr><td>class_n</td></tr>
</table>

Figure 6: Confusion matrix example

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

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

The example is an extreme example and is intended to illustrate that accuracy is not always the best choice of a classification metric, although it is very often used in this paper. The classes in the used data set should be balanced as far as possible. The data set used in this thesis is not very well balanced (see figure A.1), but far away from the extreme example above. The data set used in the chapter "Data augmentation" is very well balanced.

2.2.4 Other metrics

There are other metrics such as precision, recall and F-measure, but they will not be discussed here because they are not used in this thesis. Precision, for example, is used how reliable the statement of a prediction of a class is. It is calculated from the ratio of the correctly classified objects of a class to all predictions of this class.

2.3 Machine learning

Machine learning is a generic term for the artificial generation of knowledge from experience.[25] It follows the approach of inductive learning (see also chapter Inductive approach).

[24] 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.

[25] De Mantaras and Armengol, "Machine learning from examples: Inductive and Lazy methods".

2.3.1 Short definitions

2.3.1.1 Backpropagation

Backpropagation[26] is a procedure for tuning ANNs during training. At any time during the training the current output of the network is known and can be compared with the expected output. The backpropagation algorithm tries to adjust the parameters of the network to reduce the loss between current output and expected output (see also chapter "Loss function").

2.3.1.2 Overfitting und underfitting

The term overfitting and the term underfitting describe two opposite extremes and both result in poor performance of model accuracy on test data sets.[27] While with overfitting the model was trained too precisely on the details of the training data set, with underfitting the model was not trained sufficiently. An overfitted model usually provides very good training accuracies. The purpose is to find a middle way, where a generalized model is trained, which provides high accuracies on a test data set that the model has not seen before.

2.3.1.3 Batch size

The batch size defines the number of images that are simultaneously passed forward and backward through the NN before the parameters in the network are adjusted (gradient update). Normally it is intended to train all training data per epoch with one process. Especially in the field of image classification this is usually not possible due to the limited memory, which is why an epoch is divided into many small batches (mini-batches). The advantage is that one only has to keep the data in memory that is necessary for the current batch. Furthermore, overfitting can be more easily limited with small batches.[28] The disadvantage is an increased variance (noise in the accuracy of the model), since it is unlikely that a small batch is a good representation of the entire data set. The computing time increases as well, since the backpropagation increases according to the number of mini-batches. There is no magic rule for choosing the right batch size. This is a hyperparameter that must be determined before the training begins.

2.3.1.4 Class

In this thesis, a class is a collection of arbitrary objects. These objects are defined by a logical property that all objects in this class satisfy. For example, these could be pictures that contain a pizza. All these images are now assigned to the class pizza.

2.3.1.5 Data augmentation

Data augmentation is the artificial augmentation of a data set. This technique is primarily used when little data is available or the data set is not sufficiently balanced (unbalanced data set). Existing images are rotated, mirrored, color adjusted, cropped, etc. Data Augmentation can improve model accuracy during training.[29]

[26] *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).

[27] H Jabbar and Rafiqul Zaman Khan. "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)". In: *Computer Science, Communication and Instrumentation Devices* (2015). URL: https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf.

[28] *Epochs, Batch Size, & Iterations*. Library Catalog: docs.paperspace.com. URL: https://docs.paperspace.com/machine-learning/wiki/epoch (visited on 02/29/2020).

[29] Luis Perez and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning". In: *arXiv preprint arXiv:1712.04621* (2017). URL: https://arxiv.org/pdf/1712.04621.pdf.

2.3.1.6 Dropout

Dropout is a regularization technique to reduce overfitting in neural networks.[30] The term refers to ignoring connections between neurons in a neural network during the training period. This ensures that the currently trained object is only partially learned. The ignored connections are selected randomly.

2.3.1.7 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.

2.3.1.8 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.

2.3.2 Methods of machine learning

Different ML 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.

2.3.2.1 Supervised learning

Supervised learning refers to ML with known training data sets (see also chapter "Inductive approach"). The learning process in turn refers to the ability of an artificial intelligence (AI) 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 thesis 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[31]. 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 ANN without a hidden layer (single-layer perceptron[32]), the delta rule[33] for correction can be applied. For networks with one or more hidden layers backpropagation[34] is used to minimize the error. Backpropagation is a generalization of the delta rule.

[30] *Dropout (neural networks)*. en. Page Version ID: 901427811. June 2019. URL: https://en.wikipedia.org/w/index.php?title=Dropout_(neural_networks)&oldid=901427811 (visited on 03/06/2020).

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

[32] *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).

[33] *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).

[34] *Backpropagation*.

The neural network is only one algorithm from the category of supervised learning algorithms. Other possible algorithms are:

- k-nearest neighbors[35]
- Linear regression[36]
- Logistic regression[37]
- Support-vector machine[38]
- Random forest[39]
- etc.

2.3.2.2 Unsupervised learning

Unsupervised learning tries to gain knowledge of patterns 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 pizza, etc.

The following unsupervised learning algorithms can be used for clustering:

- k-means[40]
- Hierarchical clustering[41]
- Expectation–maximization[42]
- etc.

One technique called 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.

2.3.3 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 ML 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 ML.

Neural networks process an input vector $\bar{x}$ and convert it to a new output vector $\hat{\bar{x}}$. They are networks of many artificial neurons connected in series and parallel. An artificial neuron in turn converts a vector into a scalar by scaling and summing the inputs $\bar{x}$ with the changeable parameters

[35] *k-nearest neighbors algorithm*. en. Page Version ID: 942113305. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=942113305 (visited on 02/28/2020).

[36] *Linear regression*. en. Page Version ID: 935782381. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Linear_regression&oldid=935782381 (visited on 02/28/2020).

[37] *Logistic regression*. en. Page Version ID: 941157282. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Logistic_regression&oldid=941157282 (visited on 02/28/2020).

[38] *Support-vector machine*. en. Page Version ID: 942477636. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=942477636 (visited on 02/28/2020).

[39] *Random forest*. en. Page Version ID: 938369502. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=938369502 (visited on 02/28/2020).

[40] *k-means clustering*. en. Page Version ID: 942500957. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=942500957 (visited on 02/28/2020).

[41] *Hierarchical clustering*. en. Page Version ID: 934548831. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=934548831 (visited on 02/28/2020).

[42] *Expectation–maximization algorithm*. en. Page Version ID: 936223068. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Expectation%E2%80%93maximization_algorithm&oldid=936223068 (visited on 02/28/2020).

$\bar{\omega}$ and correcting them with a bias b (the bias is also a changeable variable). The activation function $step(z)$ ensures that the first degree polynomial (linear regression model) becomes a nonlinear function[43] (Figure 7).

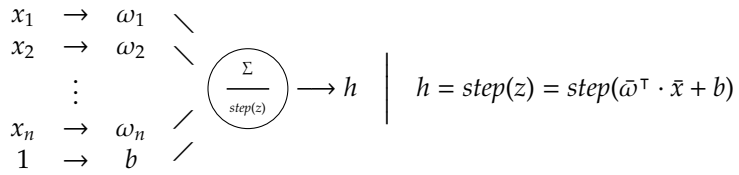

Figure 7: The construction of an artificial neuron

The artificial neural network is composed of many layers connected in series, which again contain neurons connected in parallel (Figure 8).

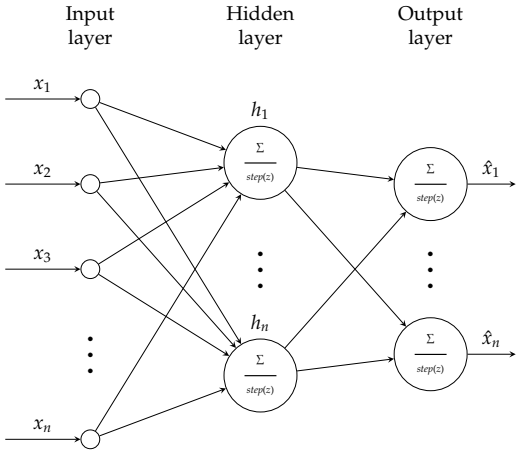

Figure 8: The construction of a simple neural network

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 $\bar{x}$ represents the coordinates of the corresponding class points and $\bar{w}$ and b are learnable parameters:

$$f(\bar{x}, \bar{\omega}, b) = sgn(\bar{\omega}^\intercal \cdot \bar{x} + b) \tag{10}$$

With this linear function, the classification example of the first representation of Figure 9 (i) can be easily classified. 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[44]. In other words, this function can separate exactly two classes linearly. But what about nonlinear problems?

[43] *Aktivierungsfunktionen, ihre Arten und Verwendungsmöglichkeiten*. de-DE. Library Catalog: www.ai-united.de Section: Mathematik. Jan. 2019. URL: https://www.ai-united.de/aktivierungsfunktionen-ihre-arten-und-verwendungsmoeglichkeiten/ (visited on 02/28/2020).

[44] Vapnik–Chervonenkis dimension: https://en.wikipedia.org/wiki/Convolutional_neural_network

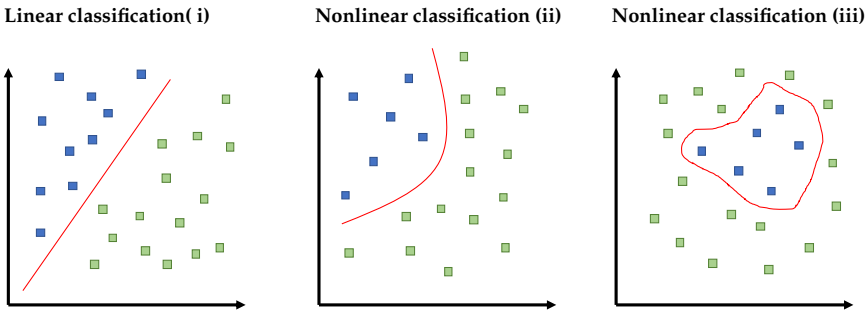

Figure 9: Linear vs. nonlinear classification

For the second classification example one could still adjust the function (Figure 9, ii). For the third nonlinear example (Figure 9, iii) 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`[45]. In the simplest case, the above mentioned second nonlinear example can can be solved by adding a hidden layer with three additional neurons (Figure 10).

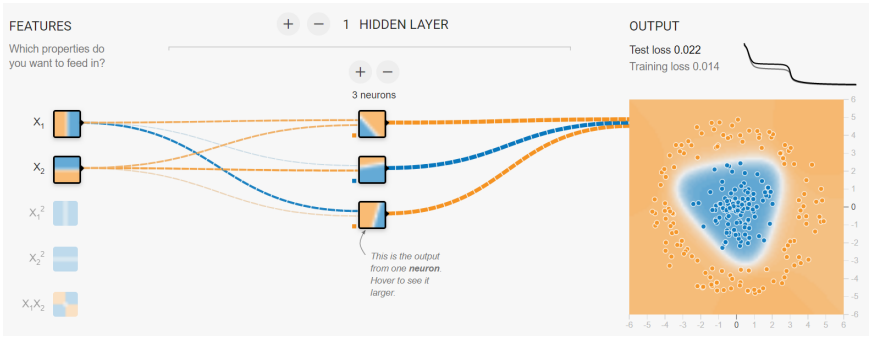

Figure 10: Simple neural network with one hidden layer[46]

2.3.4 Convolutional neural network

An artificial 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 a vector 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 t of square matrices (usually 3×3 or 5×5 matrices, $n \times n$). These matrices are called filters[47], convolution matrices or kernel[48,49] k_{nn} and are now applied to each area of the image I_{xy} (Equation (11)).[50] This process creates a new image (Figure 11c) and this image is commonly named

[45] Neural Network Right Here in Your Browser: `https://playground.tensorflow.org`

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

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

[48] Jianxin Wu. "Introduction to convolutional neural networks". In: vol. 5. 2017. Chap. 6.1 What is convolution?, pp. 11–13. URL: `https://pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf`.

[49] *Kernel (image processing)*. en. Page Version ID: 929690058. Dec. 2019. URL: `https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=929690058` (visited on 03/07/2020).

[50] Jianxin Wu. "Introduction to convolutional neural networks". In: vol. 5. 2017. Chap. 6.3 Convolution as matrix product, pp. 15–17. URL: `https://pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf`.

feature map.

$$I^*(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} I(x-i+a, y-j+a)\cdot k(i,j) \quad \Big| \quad a = \frac{n+1}{2} \tag{11}$$

With a number of t kernels, t feature maps are created in which features defined in the filters are highlighted. This process is also called convolution.[51]

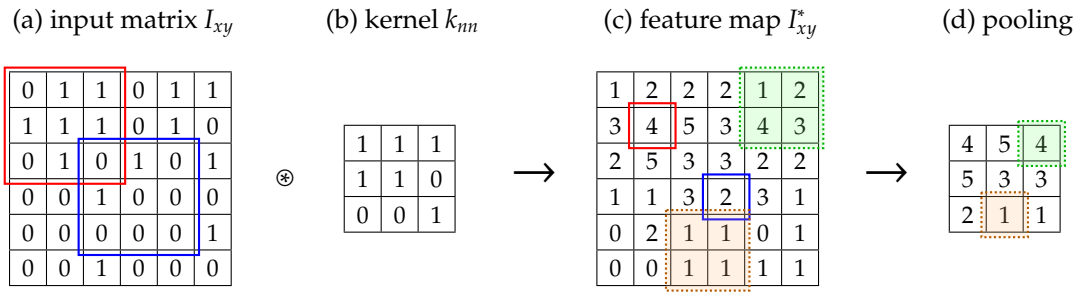

Figure 11: Simple convolution and simple pooling

Neural networks that make use of convolutional layers are called convolutional neural networks (CNNs) and have made a decisive contribution to the progress of image classification and also in other areas like speech recognition.[52] In addition to the convolutional layers, a CNN 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 (Figure 11d). This process reduces the number of parameters and the computational complexity of the model.[53] The convolutional layer and the pooling layer usually alternate until a large vector is created at the end (instead of matrix I_{xy}). This vector can be processed by a normal neural network and finally ends with the output of the probability vector $\boldsymbol{\lambda}$ as described in chapter "Loss function" (Figure 12).

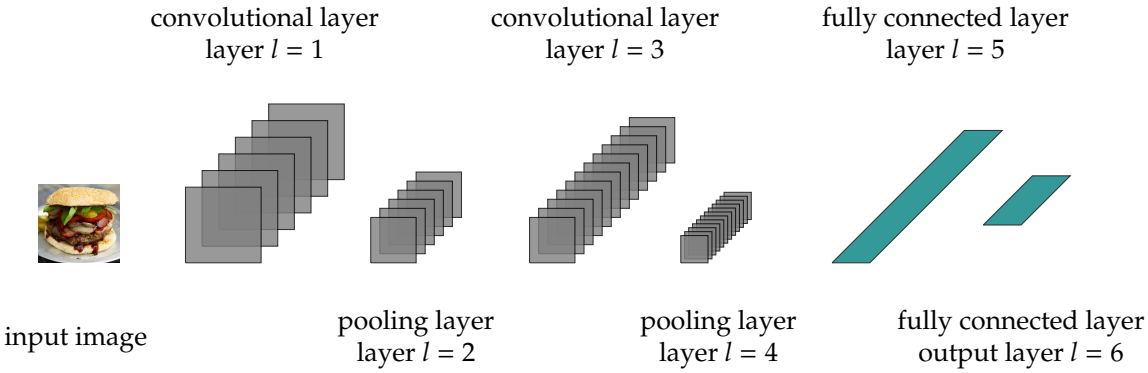

Figure 12: Architecture of a traditional convolutional neural network

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.

[51] Wu, "Introduction to convolutional neural networks".

[52] Aurélien Géron. "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme". In: O'Reilly Verlag, 2017. Chap. Convolutional Neural Networks, pp. 359–384. ISBN: 9783960090618.

[53] Keiron O'Shea and Ryan Nash. *An introduction to convolutional neural networks*. 2015. URL: https://arxiv.org/pdf/1511.08458.pdf.

2.3.5 Transfer learning

CNNs have made a significant contribution to the classification of images.[54,55] 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%. That doesn't sound a lot, but this model won the ImageNet competition ILSVRC 2012[56] far ahead of the second place with a top-5 error rate of more than 26%. The special thing about this is the innovation of the CNN technology. With this technology the model accuracy can be improved yearly. A CNN called `ResNet-50` from the year 2015 achieved a model accuracy of 5.3%.[57] 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 fully connected layer at the end of the convolutional neural network to the own problem (Figure 13).

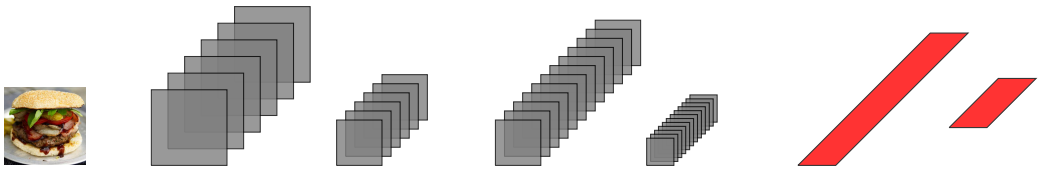

Figure 13: The green area (see Figure 12) was replaced by a new neural network (red) adapted to the new problem.

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

2.3.6 Overview of current and known convolutional neural networks

In the following a few current and well-known convolutional neural networks will be presented (Figure 14). 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 data set)
- 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 Mbyte)

[54] Géron, "Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme".

[55] Huan Ning. "Prototyping a Social Media Flooding Photo Screening System Based on Deep Learning and Crowdsourcing". In: 2019. Chap. 2.3 Image classification based on deep learning, pp. 13–15. URL: `https://scholarcommons.sc.edu/cgi/viewcontent.cgi?article=6207&context=etd`.

[56] *ILSVRC 2012*. URL: `http://image-net.org/challenges/LSVRC/2012/`

[57] Md Zahangir Alom et al. "The history began from alexnet: A comprehensive survey on deep learning approaches". In: 2018. Chap. TABLE II. The top-5% errors, p. 15. URL: `https://arxiv.org/pdf/1803.01164.pdf`.

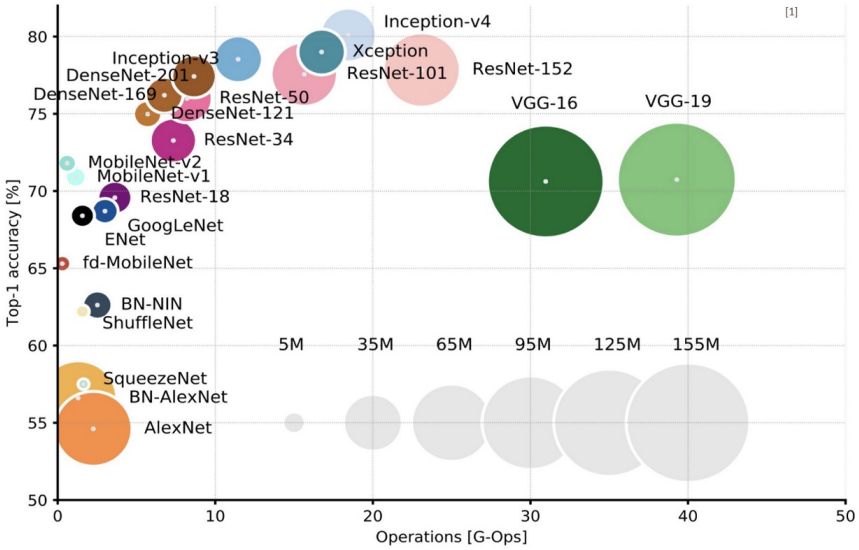

Figure 14: Overview of current and known convolutional neural networks[58]

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

3 Related work

There are a number of studies on image classification on very large data sets.[59,60,61] Most of the time this huge number of classification classes and data sets is not desired. Furthermore, there are many investigations on small data sets.[62,63] 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.

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

[60] 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*. 2014, pp. 1891–1898. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Sun_Deep_Learning_Face_2014_CVPR_paper.pdf.

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

[62] Francois Chollet. "Building powerful image classification models using very little data". In: *Keras Blog* (2016). URL: http://deeplearning.lipingyang.org/wp-content/uploads/2016/12/Building-powerful-image-classification-models-using-very-little-data.pdf.

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

4 Considerations and implemention

4.1 Research questions and hypothesis section

This thesis deals with image classification applications with limited resources. The following questions arise: With which tools, techniques and ideas is it possible to create a successful image classification model even with few resources? 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? The following chapters will show that it is possible to reach a good classification accuracy even with a small traning data set.

4.2 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 a GitHub repository.[64] This framework allows to set all the hyperparameters mentioned in this thesis.[65] Below is an example of a command line call with the default values from the chapter "Default setup" (Listing 1). The created model is located in the specified directory with the name `model.h5`:

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

Listing 1: Example command line call to train the given data path

4.3 Performance

If one intends to implement and optimize deep neural networks (DNNs), the calculations must take place on the GPU. It is also possible to run calculations on the CPU. Also, the installation of the Keras library (see chapter "") for CPU driven computations is much easier, because the installation of the GPU drivers is not necessary. The disadvantage of training without GPU support is that it takes longer to train larger models. Sufficiently trained models for the classification of images, for example, are only achieved after several training units. A Training unit requires a lot of computing power in the form of many matrix operations. A GPU is predestined for matrix operations.[66]

Table A.1 in the appendix in shows a tabular comparison of performance based on the Kaggle flower data set training.[67] This image set consists of 5 classes with about 4242 training images[68] and was trained with the default values from chapter "Default setup" (but with only 10 epochs, `InceptionV3`).

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

[64] Björn Hempel. *Keras Machine Learning Framework*. original-date: 2019-09-12T21:51:11Z. Feb. 2020. URL: `https://github.com/bjoern-hempel/keras-machine-learning-framework` (visited on 02/29/2020).

[65] Björn Hempel. *Keras Machine Learning Framework - Arguments of the training process*. en. Library Catalog: github.com. URL: `https://github.com/bjoern-hempel/keras-machine-learning-framework` (visited on 02/29/2020).

[66] 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*. 2004, pp. 133–137. URL: `https://graphics.stanford.edu/papers/gpumatrixmult/gpumatrixmult.pdf`.

[67] *Flowers Recognition*. en. Library Catalog: www.kaggle.com. URL: `https://kaggle.com/alxmamaev/flowers-recognition` (visited on 02/29/2020).

[68] Björn Hempel. *Keras Machine Learning Framework - GPU vs CPU*. en. Library Catalog: github.com. URL: `https://github.com/bjoern-hempel/keras-machine-learning-framework` (visited on 02/29/2020).

for all further tests clearly falls on the GPU. All experiments (unless otherwise specified) were trained on a Nvidia GTX 1060 graphic card with 6 Gbyte of memory.

4.4 Experimental Setup

4.4.1 Software specification

Python v3.6.9

Python has become one of the most important programming languages in the field of machine learning and data sciences in recent years.[69] The language was chosen as programming language because it has a large number of ML libraries to choose from. The fact that the TensorFlow application programming interface (API) is mainly designed for use with Python and thus allows stable development is also a contributing factor to the choice.

TensorFlow GPU v1.14.0 and Keras-GPU v2.2.4

TensorFlow GPU was chosen in conjunction with Keras GPU v2.2.4 to perform the calculations on the GPU and not on the CPU. This has significantly reduced the time of model creation (see chapter "Performance").

Keras v2.2.4

In the following experiments TensorFlow is used as backend. Keras translates the function calls into the corresponding functions of TensorFlow. Keras as the parent library simplifies the use of TensorFlow, makes the code easier to read and minimizes errors.

4.4.2 Used data set

The used data set is a food data set consisting of 50 classes, 14,866 image files and has a size of 765 Mbyte (`food-50`). This data set was created and labelled manually. The files are unevenly distributed between the classes (unbalanced). Before you can start the training process, the used data set must be split into a training and a validation data set. As a ratio 80% training and 20% validation data was chosen (11,913 images versus 2,953 images). The class with the smallest image set is `french_fries` and contains 36 training images and 8 validation images. The class with the largest image set is `grilled_cheese_sandwich` and contains over 390 images in the training data set and over 97 images in the validation data set. On average, all classes contain about 240 training data and 60 test data (see Figure A.1 for details). An additional test data set is not used in this thesis (see chapter "Training, validation and test data set").

4.4.3 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 (the learning rate η from epoch 15 to 21 is 0.00025)
- **image size:** 299x299 pixels[70]

[69] *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).

[70] The original image is reduced to 299 pixels. If it is not a square image, the larger side is scaled down to 299 pixels.

- **batch size:** 16
- **drop out:** 50%
- **CNN weights:** ImageNet (TL)
- **activation function:** rectified linear unit (ReLU)
- **optimizer:** stochastic gradient descent (SGD) with Nesterov
- **momentum:** 0.9 (with decay 0.0)
- training of all CNN layers (unless otherwise specified)
- the entire training and validation set (14,866 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` and `Xception`

5 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 was explained in chapter "Default setup". Starting from this standard setup, the variable parameters are changed and discussed in the corresponding chapters. The purpose is to find the parameters with which the recognition accuracy of the models is high.

5.1 Model validation

5.1.1 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 "Default setup" 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 elements to the total number of 11,913. 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, 1,000, 2,000, etc. (Figure 15).

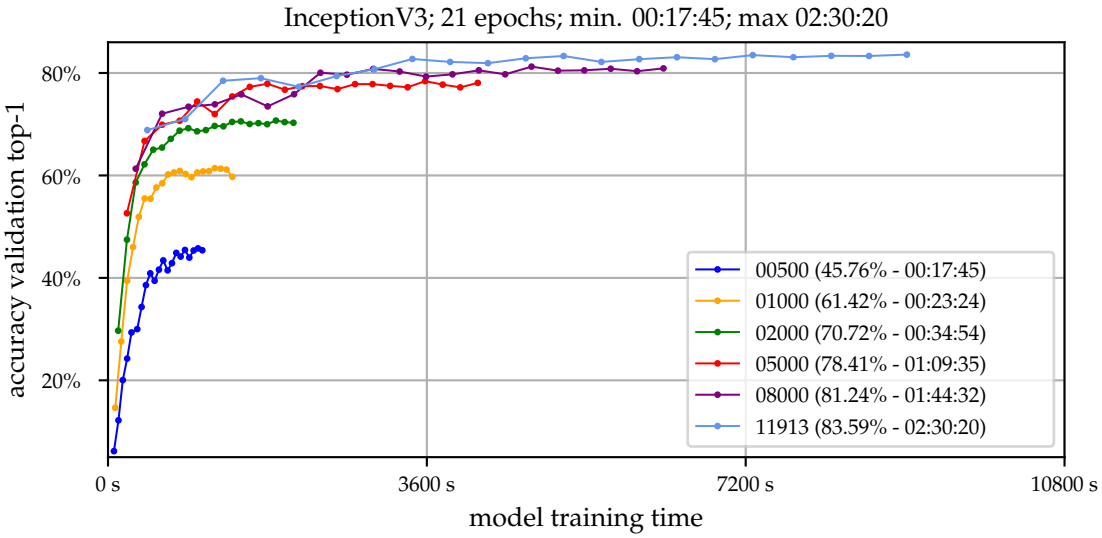

Figure 15: Overview of influence of number of trained images on accuracy

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 12,000 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 elements. Although the accuracy in the upper range (more than 5,000 images) still seems to increase, the jumps are not very big anymore and seem to be moving to a limit. For further research one could still find out from which number of training data elements no further significant increase in accuracy is 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 can theoretically save computing time. The example with a data set of 11,913 images (light blue trace) reaches an accuracy of 68.1% after six and a half minutes of training already in the first epoch. With 1,000 images (orange trace), the maximum accuracy is 61.42% after 18 epochs and about 20 minutes computing time. In this case, a significantly too small data set does not reach the model accuracy that a data set about ten times as large reaches with the first epoch and less time.

5.1.2 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 data set and the results are compared using this data set. What is the result with smaller data sets? The training result according to the standard setup and with different CNN models of eight models is as follows (Figure 16).

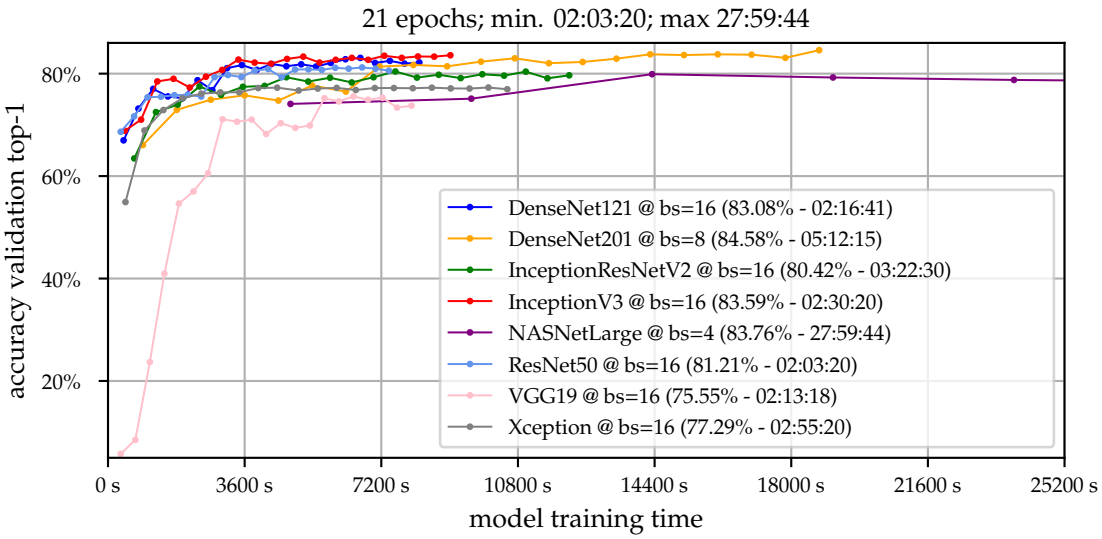

Figure 16: Overview of known CNN models

All models except `DenseNet201` and `NASNetLarge` were trained with a batch size of 16 ($bs = 16$). Due to the limited 6 Gbyte memory of the Nvidia GTX 1060[71], `DenseNet201` was trained with a batch size of 8 ($bs = 8$) and `NASNetLarge` with a batch size of 4 ($bs = 4$). A real comparison of these two exceptions is not really possible, but should not 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 14 ("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 model called `VGG19` 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 an average time 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.

5.1.3 Use of the transfer learning (TL) approach

What influence does the use of transfer learning have on accuracy? As described in the chapter "Transfer learning", the TL approach immediately improves accuracy because the model already has recognition features that an unlearned network has yet to learn. The approach is applied to the data set `food-50`. For the first time, not only the validation accuracy (thick trace) but also the training accuracy (thin trace) is drawn in.

[71] *GeForce 10 series*. en. Page Version ID: 943820023. Mar. 2020. URL: `https://en.wikipedia.org/w/index.php?title=GeForce_10_series&oldid=943820023` (visited on 03/07/2020)

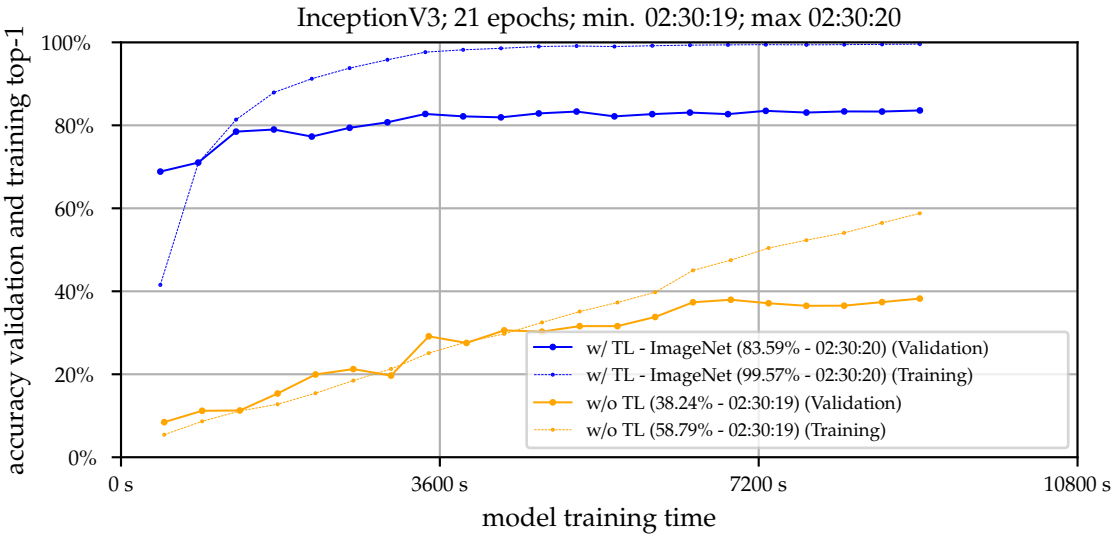

Figure 17: Overview of use of the TL approach

As expected, the model with the TL approach achieves a better result in the same computing time (blue trace versus orange trace). In the first epoch the model without transfer learning approach reaches an accuracy value of 8.5% and increases to 38.2% in 21 learning epochs. The training accuracy of 58.8% is still far below the values achieved with TL after a few epochs. This means that learning of features can be continued with the existing data set. If one starts with a pre-trained network, an accuracy value of 68.9% is achieved in the first epoch. This is a jump of 30% as the current best value without TL approach. The accuracy can be further increased to 83.6% in the 21st epoch. What happens now, if the data set without TL approach is trained with 49 epochs instead of the 21 epochs used so far? Two experiments follow: The first experiment uses a dropout of *dropout* = 0.5 as before from the default setup (green trace). In the second experiment (red trace), a dropout of *dropout* = 0.0 is used to force an overfitting and focus on learning the recognition features (Figure 18).

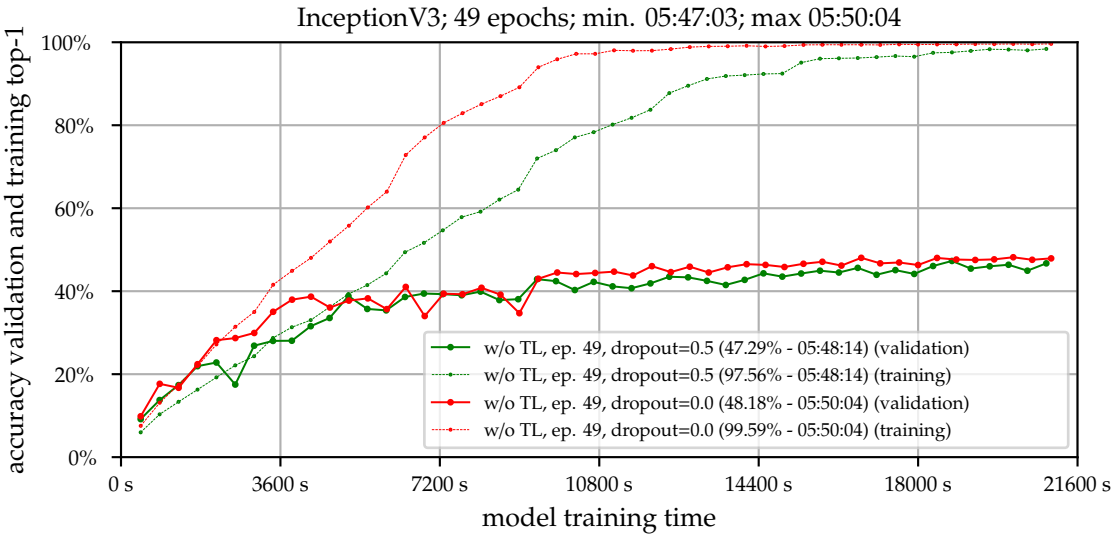

Figure 18: Overview of training without TL approach

With further 28 learning epochs, the model accuracy was again increased to 10% in both experiments. Especially with a dropout of *dropout* = 0.0 an accuracy of 48.2% could be achieved (still below the value with TL approach), whereas the training accuracy has already reached 99.59% and a further generalization is theoretically no longer possible. In order to make further progress and to achieve better accuracies on previously unseen images, more images are needed to learn features. Conclusion: For the training of approaches without TL the currently available image set

is no longer sufficient to achieve better or the same values as with TL approaches. Especially when using small data sets, the TL approach should be considered.

5.1.4 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[72]: 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 (Figure 19).

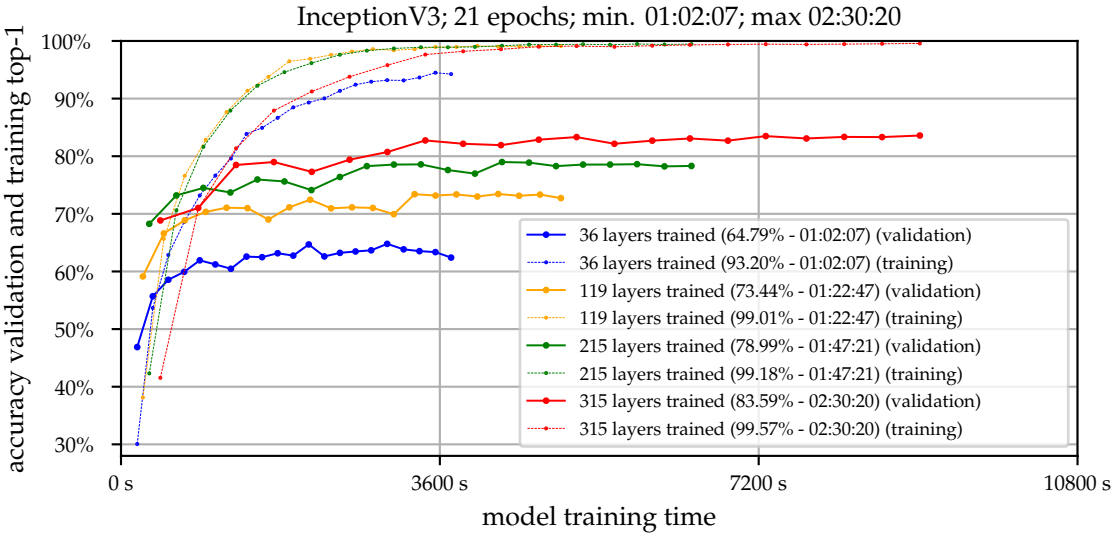

Figure 19: Overview of influence of the number of trained layers

Skipping the training of layers in the convolutional part of the network immediately reduces the model accuracy significantly. 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 315 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 315 layers from the InceptionV3 CNN. The extra computing time required to train all layers in the convolutional part should be scheduled.

5.1.5 Influence of different error optimizers

The optimizer in the training process is used to minimize the value of the loss function. This is an attempt 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 classification function. The simplest approach is the gradient descent algorithm[73], which has been improved over time and further optimization algorithms have been developed. The learning rate η determines the value of the weighting with which the error is corrected (see also chapter "Loss function"). The recalculated probability vector λ_{new} is passed back to the backpropagation algorithm:

$$\lambda_{new} = \lambda_{current} - \eta \cdot L_r(\vartheta, \lambda_{current}) \tag{12}$$

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

[73] *Gradient descent*. en. Page Version ID: 943737051. Mar. 2020. URL: https://en.wikipedia.org/w/index.php?title=Gradient_descent&oldid=943737051 (visited on 03/07/2020)

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.

5.1.5.1 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, should optimize the convergence[74]. Below is a comparison of current optimization methods (Figure 20).

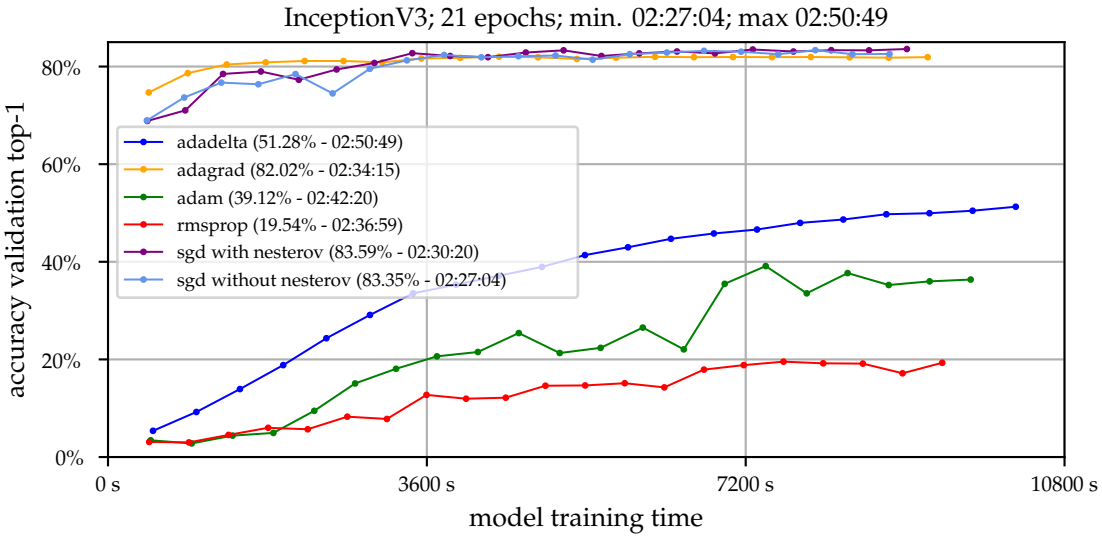

Figure 20: Overview of best optimizer (validation)

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 (see also training validation in Figure 21).

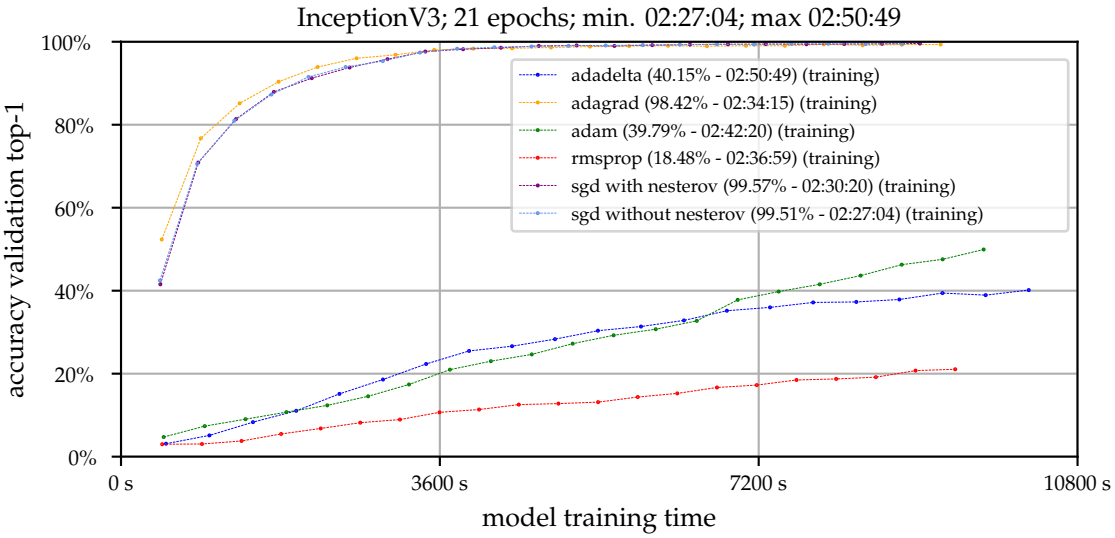

Figure 21: Overview of best optimizer (training)

[74] 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`

All procedures were performed with the recommended settings according to the Keras documentation[75]. A learning rate of $\eta = 0.001$ was chosen for all procedures.

5.1.5.2 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. This time, the focus is on the momentum, which is specified as 0.9 in the standard setup. In contrast to the simple SGD method, the momentum method remembers the value of the last update of λ and uses the momentum α to define how high the influence of the last change of λ additionally affects the current change of λ (Equation 13). The update is "accelerated" in the direction in which the error correction is obviously currently moving (Equation 14).

$$\nabla\lambda_{new} = \alpha \cdot \nabla\lambda_{old} - \eta \cdot L_r(\vartheta, \lambda_{current}) \tag{13}$$

$$\lambda_{new} = \lambda_{current} + \nabla\lambda_{new} \tag{14}$$

The Nesterov momentum is a simple modification of the normal momentum. The calculation of the error function is no longer based only on the current value of λ, but also includes the current change by the momentum (equation 15). This should help to correct directional changes faster and avoid overshoots.

$$\lambda_{new} = \lambda_{current} + \alpha \cdot \nabla\lambda_{old} - \eta \cdot L_r(\vartheta, \lambda_{current} + \alpha \cdot \nabla\lambda_{old}) \tag{15}$$

In the next two experiment, the momentum is varied from 0.5 to the range of 0.98 (Figure 23). The learning rate starts at 0.001 and decreases by 50% after every seven epochs. The first experiment starts without the Nesterov momentum (Figure 22), the second experiment uses the Nesterov momentum method (Figure 23). The results will then be compared and discussed.

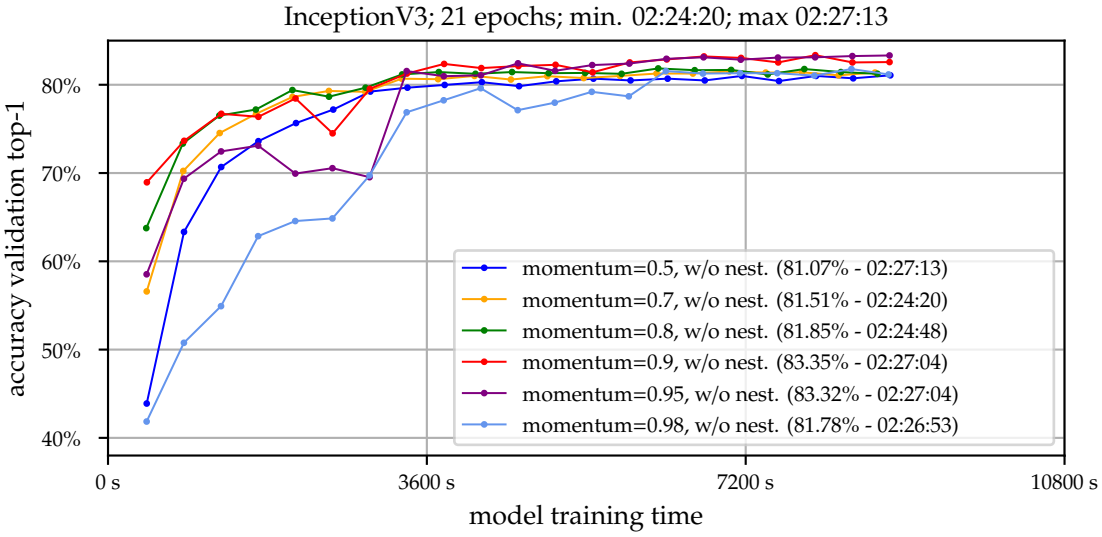

Figure 22: Overview of experiments of different momentum values without Nesterov (*momentum* $\hat{=}$ α)

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

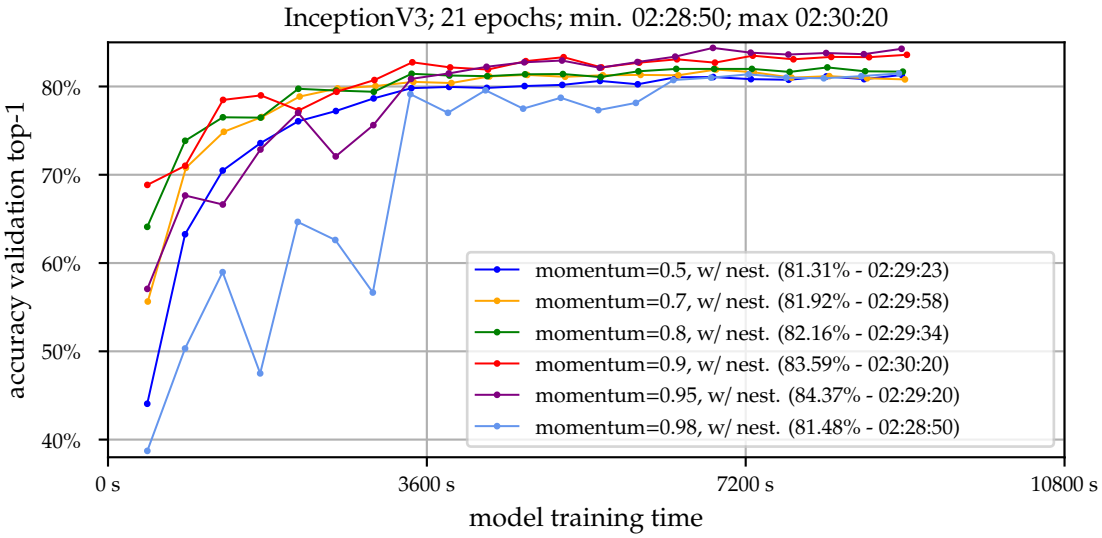

Figure 23: Overview of experiments of different momentum values without Nesterov (*momentum* $\hat{=} \alpha$)

The accuracy of both experiments with (w/ nest.) and without Nesterov (w/o nest.) increases steadily with momentum α until they reach their zenith at 83.35% and 84.37% with $\alpha = 0,9$ (w/o nest.) and $\alpha = 0,95$ (w/ nest.) respectively. It is noticeable that the smaller the momentum α, the accuracy increases more evenly, but the "maximum" at 21 epochs is not reached. With increasing momentum, e.g. at the maximum of $\alpha = 0,98$ for the Nesterov model (light blue trace), the model accuracies jump and can sometimes run completely in the negative direction with another epoch. With the best model (purple trace, w/ nest., $\alpha = 0.95$) exactly the same happens, but in the end the best results are achieved.

5.1.5.3 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:

$$\eta_{m \to next} = \beta \cdot \eta_m,\ m \in [1 + 0 \cdot \mathcal{E}, 1 + 1 \cdot \mathcal{E}, 1 + 2 \cdot \mathcal{E}, \dots] \tag{16}$$

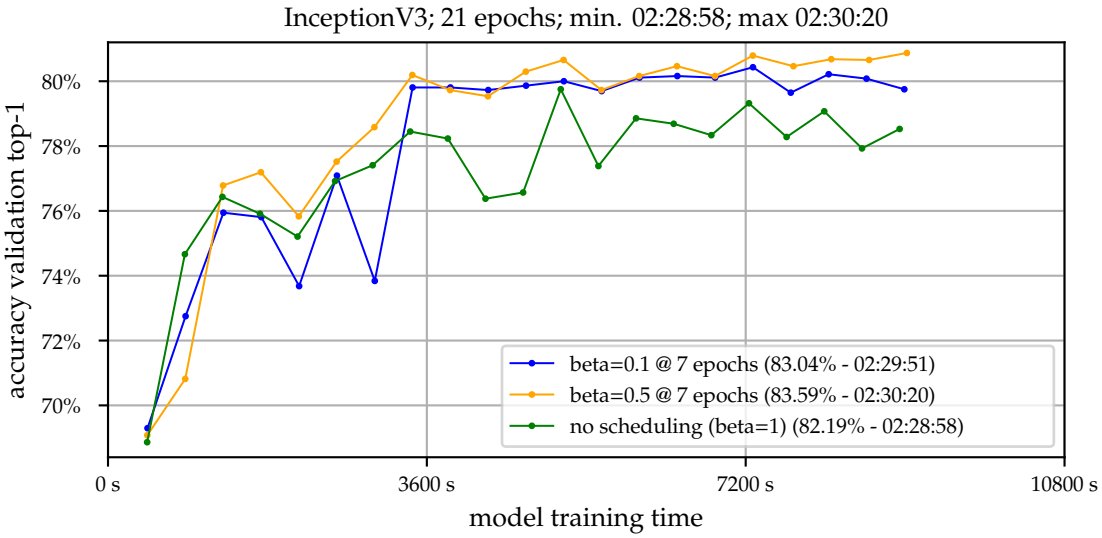

Figure 24: Overview of a dynamic learning rate on accuracy

As suspected, a reduction of the learning rate over time improves the possible model accuracy (Figure 24). 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[76] and must be determined experimentally. An empirical value can be used as a starting value.

5.1.6 Different batch sizes

The batch size is one of the regularization parameters, since it can counteract overfitting. As described in the chapter "Batch Size" the learning process should be divided into several smaller mini-batches. The question in this evaluation is how different sizes of batch size affect the accuracy of the model. In a paper by Pavlo M. Radiuk from 2017, the highest possible batch size of 1,024 provided the best accuracy and helped to avoid a high degree of variance. However, Pavlo M. Radiuk also found that the improvements in accuracy in high ranges were only very small.[77] It should also be mentioned that his data set consisted of 60,000 images distributed in 10 classes (compared to 50 classes with about 250 images per class in this thesis) and he used very small images (32x32 pixels) to allow a high value of batch size. The different batch size sizes are also applied to the `food-50` data set (Figure 25). A batch size higher than 32 could not be used due to technical reasons.

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

[77] Pavlo M Radiuk. "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets". In: *Information Technology and Management Science* 20.1 (2017), pp. 20–24. URL: `https://www.degruyter.com/downloadpdf/j/itms.2017.20.issue-1/itms-2017-0003/itms-2017-0003.pdf`.

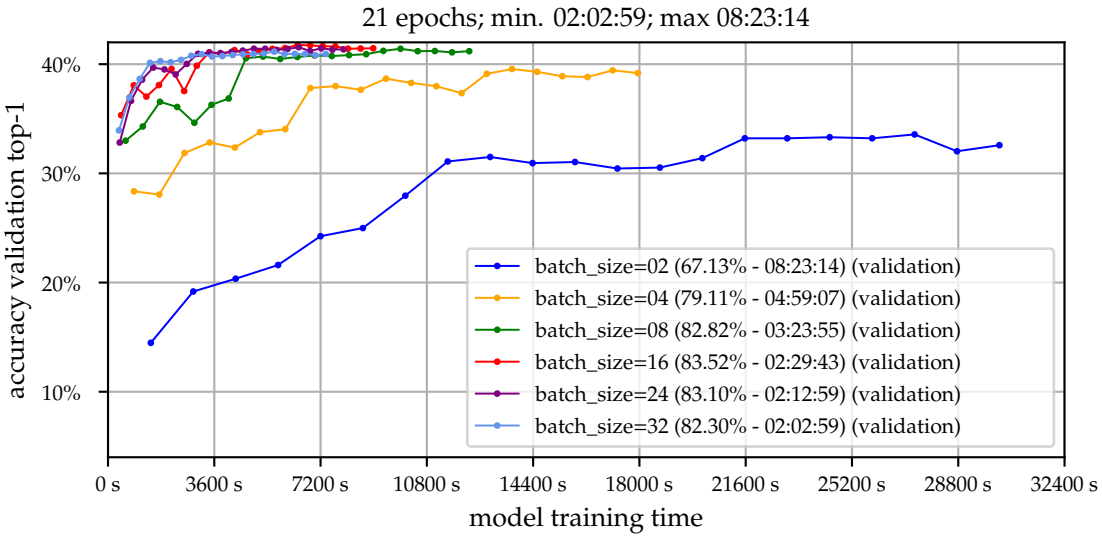

Figure 25: Overview of the influence of a different batch size (validation)

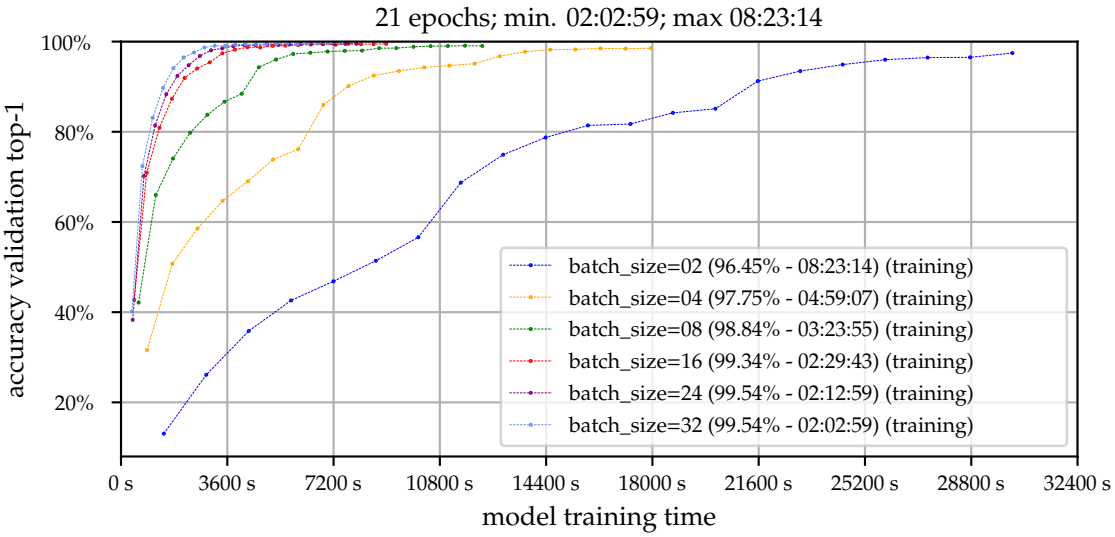

Figure 26: Overview of the influence of a different batch size (training)

A higher batch size value obviously leads to a faster and more predictable (more even) increase in validation accuracy in the first epochs (light blue trace with *batch_size* = 32), compared to a batch size of 16 (red trace), for example, where the accuracy "jumps" from epoch two to seven. However, a higher batch size does not give the best result at the end of the 21 epochs. While a value of 16 for the batch size achieves an accuracy of 83.52%, it only achieves 82.54% with a value of 32. Up to a batch size of 16, however, it can be said that as the batch size increases, a better result is always achieved. Increasing the batch size reduces the required computing time as expected. In this case, it has been improved from eight hours to two hours. Overfitting is achieved very slowly with a batch size of 2, but this model is also the worst one with a detection accuracy of 67.13%. The much faster increasing training accuracy up to 100% with increasing batch size can be seen very clearly in Figure 26. The batch size is in this case also a HP, which must be defined before the training and determined experimentally for optimal values. Investigations with a slightly more balanced data set and an even higher batch size should be aimed at to further investigate the influence of batch size on different data sets.

5.1.7 Different activation functions

The activation function ensures that the linear classification function becomes a non-linear function (see chapter "Artificial neural network") and with its help it is possible to create the classification space for image classifications..[78] There are a lot of different activation functions, which differ a little bit in their manner. The influence on the model accuracy to be achieved by these different activation functions is shown in Figure 27.

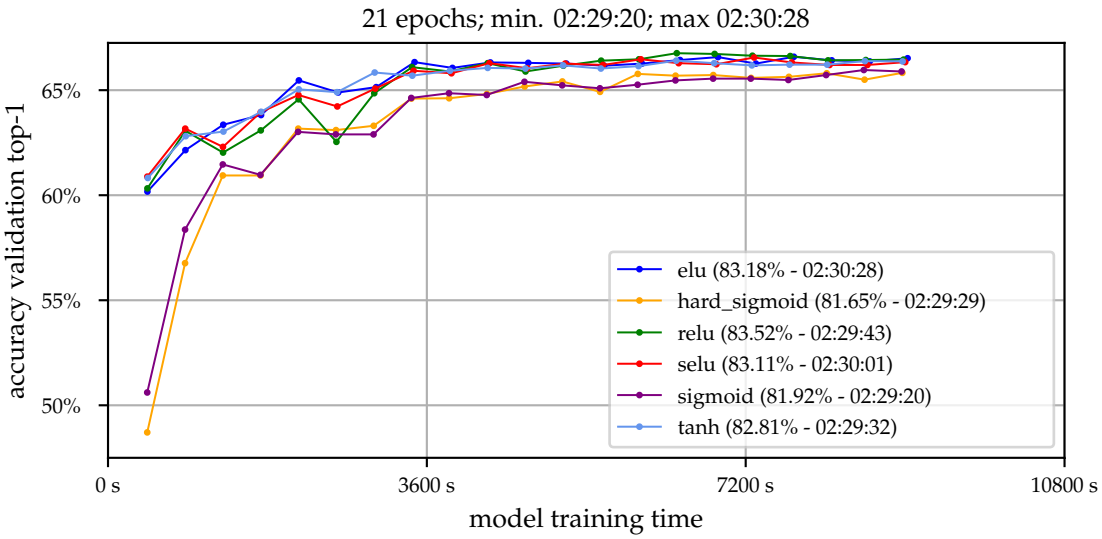

Figure 27: Overview of the influence of difference activation functions

The model accuracies to be achieved are very similar in the result. Nevertheless, there are small differences. The sigmoid function, as well as the activation function "Hard sigmoid" converge more slowly to the maximum possible model accuracy. The other four activation functions converge faster and are further ahead with about 20% more model accuracy in the first epoch. They do not differ very much in their results. The best result is achieved by the activation function "ReLU" with 83.52%.

5.1.8 Different number of learned epochs

All experiments performed here so far have been carried out with 21 learning epochs, whereby the learning rate dropped from 0.001 to 0.0025. The validation accuracies at the end of the training processes were mostly over 99% (see for example Figure 25, *batch_size* = 16) which indicates overfitting and theoretically no further improvements are possible. It will now be examined what happens if one trains over the 21 learning epochs with further decreasing learning rates η. SGD with Nesterov is used as optimizer. Once with 0.9 and once with the best value 0.95 from chapter "Influence of the momentum and the Nesterov momentum" (Figure 28).

[78] *Aktivierungsfunktionen, ihre Arten und Verwendungsmöglichkeiten.*

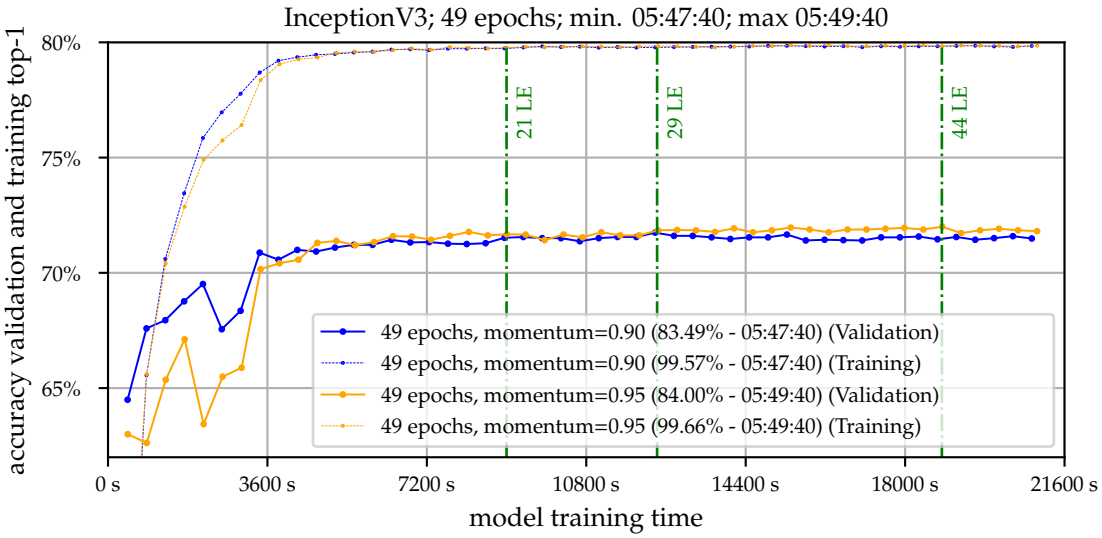

Figure 28: Overview of the influence of a longer training period with more epochs

As suspected, there is no significant improvement after 21 learning periods. The best result until the 21st learning epoch was achieved after the 21st epoch (*momentum* = 0.9, 83.04%) and after the 19th epoch (*momentum* = 0.95, 83.55%). This could then be improved in the 29th epoch to 83.49% (*momentum* = 0.9) and in the 44th epoch to 84.00% (*momentum* = 0.95). Conclusion: Increases are still possible, but an improvement of about 0.5% requires a doubling of the computing time.

5.1.9 Influence of dropout

The dropout parameter also belongs to the regularization parameter group. It also helps to avoid overfitting, in which individual neurons in the NN are randomly deactivated and do not contribute to training without affecting the rest of the model. This means that not all features are learned at once, but only some of them. In Figure 29 the results of different dropout values are compared. The dropout layer is used after the CNN within the connected neural network (for an example code see Listing A.1 in the appendix).

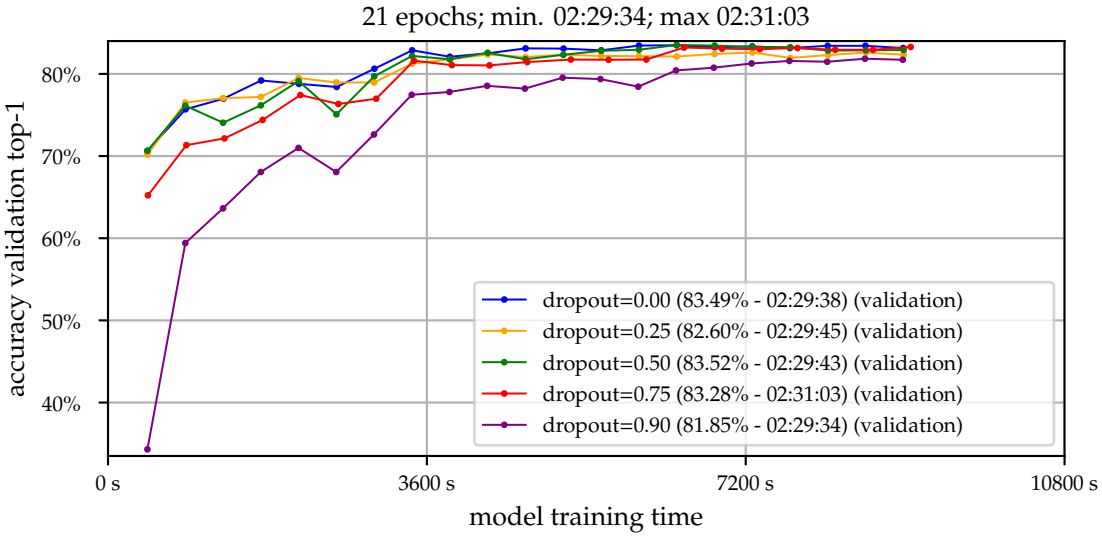

Figure 29: Overview of the dropout parameter (validation)

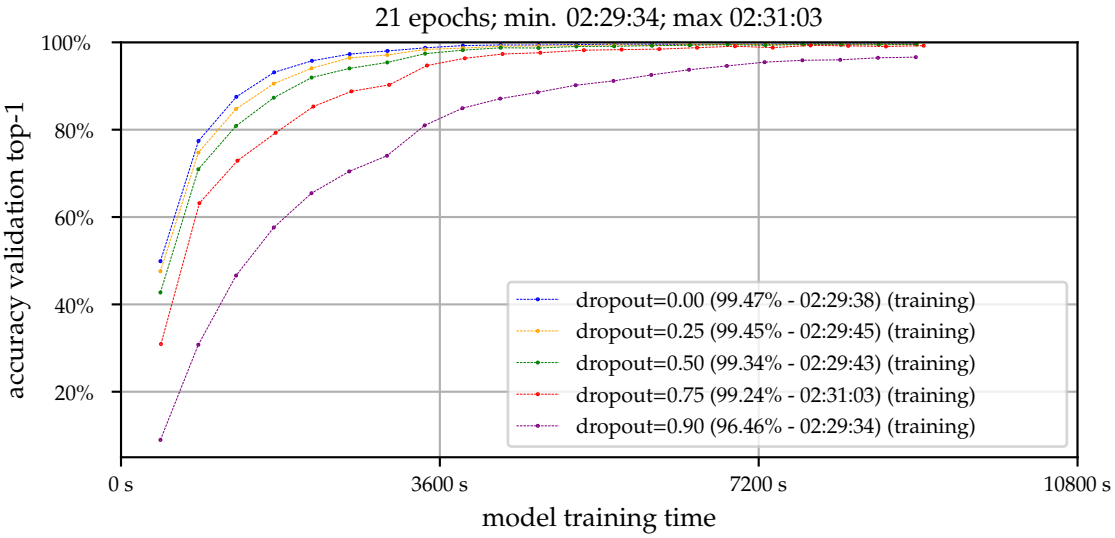

Figure 30: Overview of the dropout parameter (training)

As one can see the dropout technique helps to avoid overfitting. The higher the dropout value, the slower the training accuracy value increases. The best result with the same computing time of all experiments is obtained with a dropout value of 0.5 which reaches a model accuracy of 83.52%. Beyond that and below the model accuracy decreases again slightly with one exception: In the experiment without dropout (*dropout* = 0.0) a model accuracy of 83.49% is achieved in the 15th epoch. Compared to the best result, this result differs only minimally (by 0.03%). This result is interesting and should be investigated further. The assumption is that higher dropout values require longer training times and that the experiment quickly reached a maximum without dropout. Especially with a dropout value of 0.9 it is probable that with more training epochs an even better value can be determined. Conclusion: With current CNNs it is obviously not necessary to use the regularization technique dropout with equal computing time. Even without dropout you get the highest model accuracy. However, the experiment should be repeated with further training data sets.

5.2 Optimization process

In addition to hyperparameters (or tuning parameters), there are a number of other techniques that influence model accuracy. It should be mentioned, for example, the problem in chapter "Influence of number of trained images on accuracy", where the model accuracy could not be further investigated with more than the existing 12,000 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.

As in chapter "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.

5.2.1 Comparison of different neural network types

This chapter deals with the last layers at the end of the network. The CNN network at the end is usually completed with one or more layers of "normal" fully connected neurons (DL / fully connected layer, see chapter "Convolutional neural network"). These layers translate the properties from the convolutional part of the network into the class accuracies (see chapter "Loss function"). So far in all experiments this layer was a simple network, where a dropout layer followed by a softmax layer with size 50 was connected. What happens if you replace these layers with more complex or even simpler layers? (Figure 31)

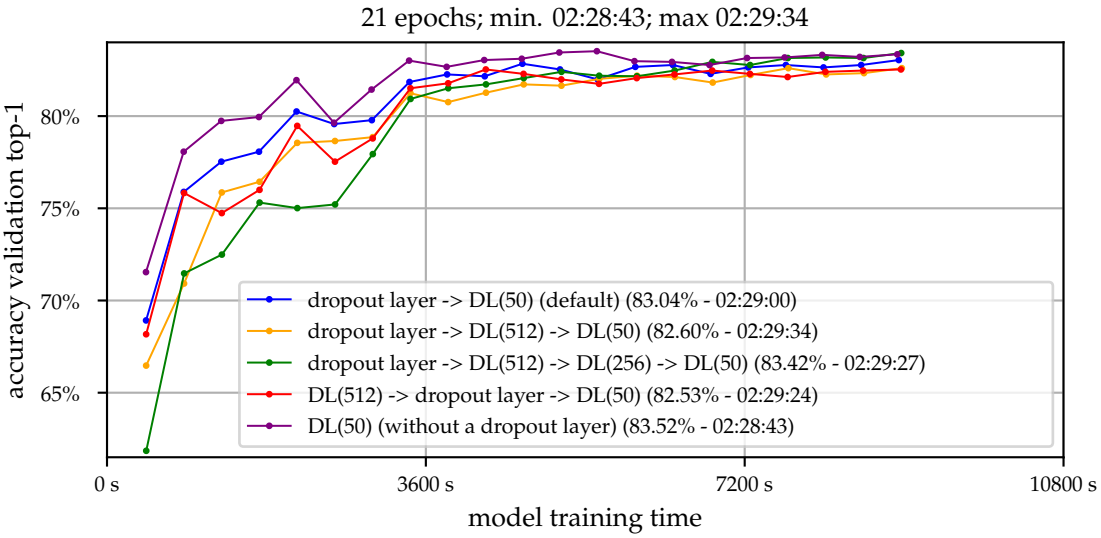

Figure 31: Comparison of different neural network types

In an article entitled "Impact of Fully Connected Layers on Performance of Convolutional Neural Networks for Image Classification[79]" one came to the conclusion that especially deep CNNs need less deep fully connected layers (FCLs) at the end to achieve good results. This could be confirmed with this experiment. The best result of 83.52% is achieved by the setup without dropout layer and only one DL. Otherwise the model accuracies do not differ too much in this experiment. Surprisingly, the most complex setup achieved the second best result of 83.43%. It consists of a dropout layer directly after the CNN output and is extended by three more DL afterwards. The last DL layer returns the probability vector λ. Adding more DLs and even a dropout layer is not necessary for current CNNs like `InceptionV3` for image classification on small data sets.

5.2.2 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.[80] 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 32).

[79] SH Shabbeer Basha et al. "Impact of fully connected layers on performance of convolutional neural networks for image classification". In: *Neurocomputing* 378 (2020), pp. 112–119. URL: `https://arxiv.org/pdf/1902.02771.pdf`.

[80] 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.

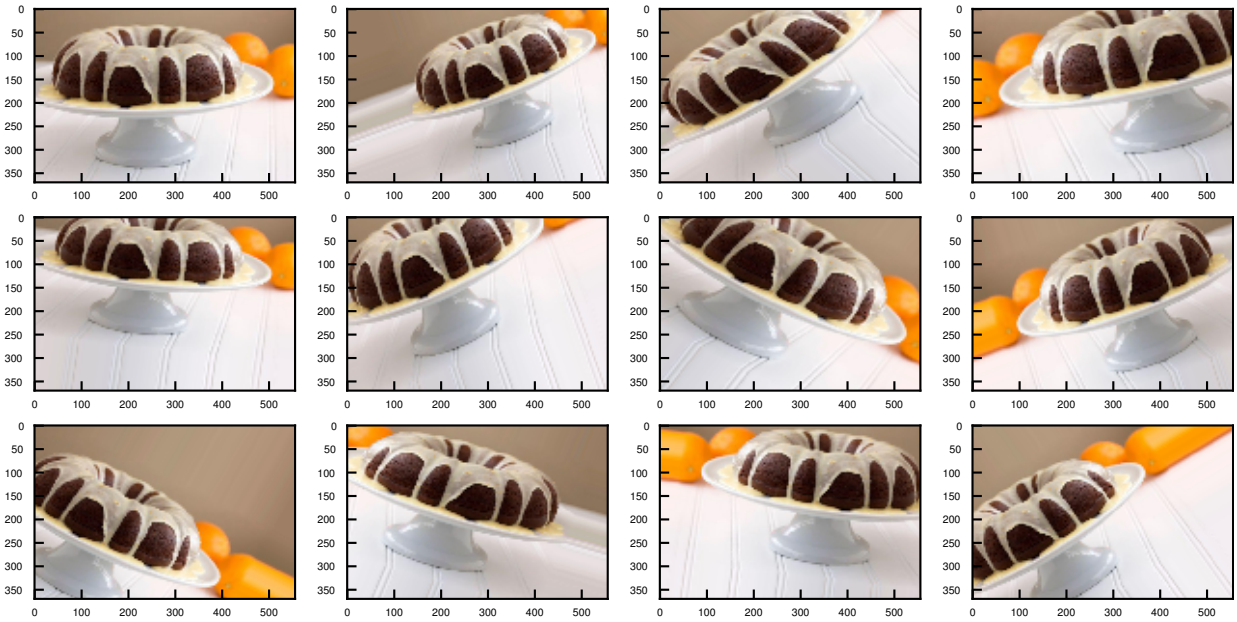

Figure 32: Example of data augmentation (The first image is the original image)

Using this technique, the existing `food-50` data set is now adjusted until each class contains a total number of 1,000 images and the data set is balanced overall. Classes with few data will receive more artificially adjusted data, classes with a lot of data will receive less. The result is that the original 11,913 images have now become a total number of 50,000 images and the data set has grown from 765 Mbyte to 1.78 Gbyte. The validation data set remains unchanged in order not to falsify the test result. The evaluation diagram is shown below (Figure 33).

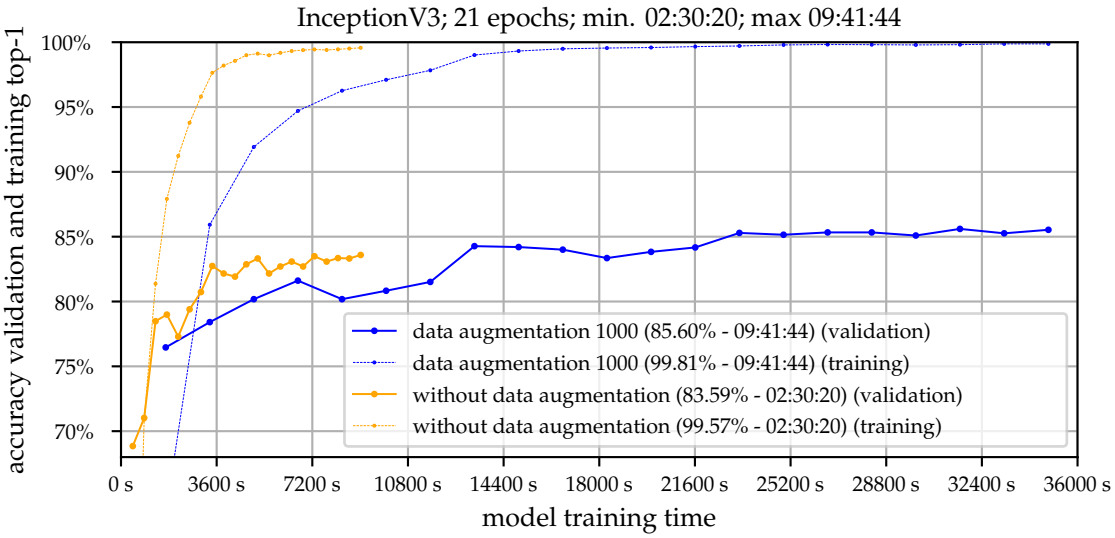

Figure 33: Comparison of data augmentation

The model accuracy increases from 83.6% to 85.6% and shows an overall increase of 2%. 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. If computing time is not important, this is one way to improve the accuracy a little. It should be mentioned that the data set was balanced with data augmentation. That means that there were the same number of elements for training in each class. At this point, it is not clear to say whether this improvement could only be achieved by increasing the available data or whether balancing the data set did also contribute the improvement. This should be investigated in further experiments.

5.2.3 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".[81]

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 and validation data and the probability vectors are determined:

$$\lambda_{class_m} = \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{50} \end{pmatrix} \quad \Big| \quad \sum_{i=1}^{50} p_i = 1 \tag{17}$$

From these probability vectors, an average vector of all n elements is calculated for each class, which really belongs to the respective class. For this 50 class model, 50 class vectors with the dimension of 50 are obtained.:

$$\forall\ 50\ classes, m = 1 \ldots 50 : \overline{\lambda}_{class_m} = \frac{1}{n} \cdot \sum_{i=1}^{n} \lambda_{class_m} \tag{18}$$

Principal component analysis is used to transform these multidimensional vectors into a two-dimensional space in order to display them visually (Figure 34).

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

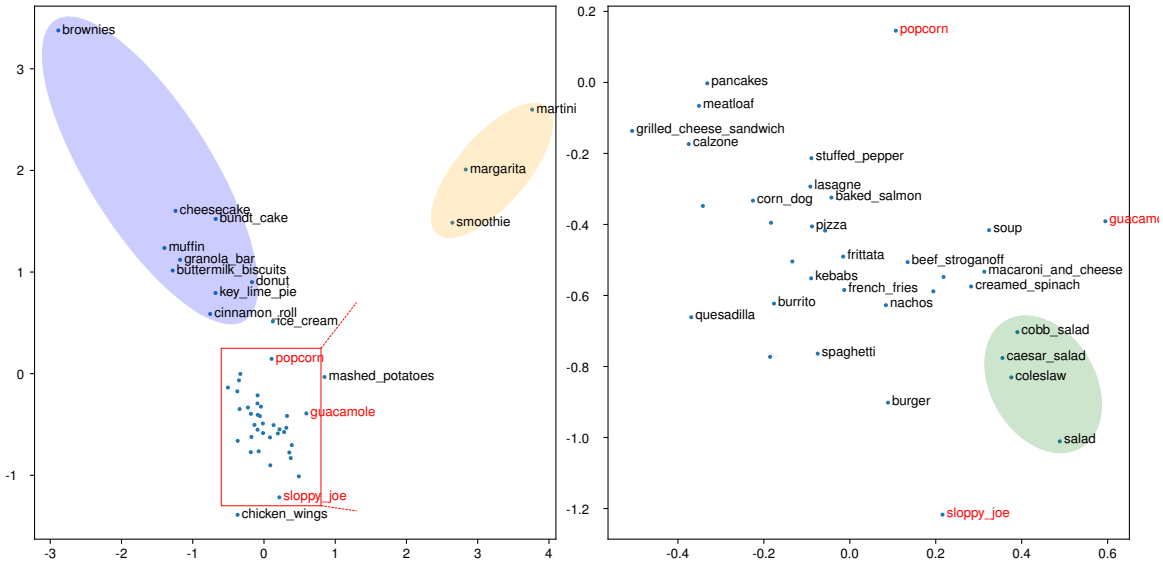

Figure 34: Principal component analysis of the `food-50` model. The x and y axis stand for the space and the similarity between the classes. A more detailed and complete version can be found in the appendix (Figure A.2).

As expected, one can now immediately recognize similar classes: drinks (orange), cakes (blue) and salads (green). In the following two chapters, the model accuracies for the pre-classification as well as the subclasses are determined as usual. In order to determine the overall accuracy of the models, this would have to be done manually one at a time, since classification using subgroups is not intended. Instead, the overall accuracy is determined statistically:

$$\widehat{acc}_{group} = acc_{preclassification} \cdot acc_{group} \tag{19}$$

$$acc_{top-1} = \sum_{group=0}^{7} ratio_{files_{val},group} \cdot \widehat{acc}_{group} \tag{20}$$

5.2.3.1 *k*-means clustering

All 50 vectors $\overline{\lambda}$ of the equation (18) are now examined by *k*-means clustering. The purpose of *k*-means is to divide the data set into *k* partitions in such a way that the sum of the squared deviations from the cluster centers is minimal.[82] Eight groups are now clustered ($k = 8$). The distribution is shown in Table 1. A more detailed version with all class names can be found in Table A.3 in the appendix. There is also a visual representation (Figure A.2).

The classes from the data set `food-50` were distributed in unbalanced groups. The group $\texttt{group}_0$ contains 18 classes in total. This corresponds to 31.1% of the entire data set. The group $\texttt{group}_4$ contains only one class with 63 files, which is 2.1% of the entire data set. With this distribution, nine training and validation data sets are now created. The first data set contains the training and validation data for the pre-classification model. This consists of eight new classes ($\texttt{group}_0$ to $\texttt{group}_7$). Each of these new classes contains all elements of the groups assigned to it (`baked_beans`, etc.). The other eight data sets are trained for models that will classify the subclasses. In each of these data sets, the classes from the data set `food-50` are distributed accordingly. For example, the new class $\texttt{group}_4$ contains only one class named `martini`. The nine data sets are trained according to the default setup, which allows to determine the top-1 validation accuracy acc^* in each case. The

[82] *k-means clustering*.

Group	$\#_{classes}$	$\#_{files_{val}}$	$ratio_{files_{val}}$	acc^*	$\widehat{acc}^*$
preclassification	8	2,953	100.0%	90.02%	
$\texttt{group}_0$	18	919	31.1%	83.94%	75.56%
$\texttt{group}_1$	2	158	5.4%	99.30%	89.39%
$\texttt{group}_2$	5	296	10.0%	93.93%	84.56%
$\texttt{group}_3$	1	83	2,8%	100.0%	90.02%
$\texttt{group}_4$	1	63	2,1%	100.0%	90.02%
$\texttt{group}_5$	4	221	7.5%	92.68%	83.43%
$\texttt{group}_6$	12	723	24.5%	81.90%	73.73%
$\texttt{group}_7$	7	490	16.6%	91.77%	82.61%
Overall	50	2,953	100,0%	acc_{top-1} = **79.23%**	

Table 1: Grouped classes with *k*-means. *=All accuracies *acc* are top-1 accuracies.

formula (20) calculates an overall accuracy using the determined accuracies and the distribution of the files. An overall accuracy of 79.23% was calculated for this model.

5.2.3.2 Agglomerative hierarchical clustering

The agglomerative hierarchical clustering algorithm describes a classification procedure in which each object first forms a cluster. Clusters that have already been formed will then be combined to form larger and larger clusters, until the number of desired groups is reached. A detailed explanation of the procedure and algorithm can be found at Wikipedia.[83] It will not be discussed here in detail. The function `AgglomerativeClustering`[84] from the sklearn library from Python was used. The parameters `n_clusters=8`, `affinity='euclidean'` and `linkage='ward'` were chosen.

The structure and the procedure for determining the model accuracy corresponds almost to the procedure mentioned in the chapter "*k*-means clustering". Only the clustering method is replaced by the agglomerative hierarchical clustering method, which results in a different distribution of groups. The distribution can be seen in table 2. A more detailed variant including class names can be found in the appendix (table A.3). There is also a visual representation of the classes (Figure A.4). The distribution is very similar this time, but a bit more unbalanced. The group $\texttt{group}_0$ contains 22 groups this time (42.91% of the total distribution). Groups with only one group are also present. The calculated overall accuracy is 78.60%, which is lesser than in the previous chapter.

Group	$\#_{classes}$	$\#_{files_{val}}$	$ratio_{files_{val}}$	acc^*	$\widehat{acc}^*$
preclassification	8	2,953	100.0%	88.93%	
$\texttt{group}_0$	22	1,267	42.91%	84.17%	74.85%
$\texttt{group}_1$	5	296	10.02%	93.21%	82.89%
$\texttt{group}_2$	4	249	8.43%	94.85%	84.35%
$\texttt{group}_3$	1	63	2.13%	100.0%	88.93%
$\texttt{group}_4$	2	158	5.35%	99.30%	88.31%
$\texttt{group}_5$	12	677	22.93%	85.63%	76.15%
$\texttt{group}_6$	3	160	5.42%	93.13%	82.82%
$\texttt{group}_7$	1	83	2.81%	100.0%	88.93%
Overall	50	2,953	100,0%	acc_{top-1} = **78.60%**	

Table 2: Grouped classes with agglomerative hierarchical clustering. *=All accuracies *acc* are top-1 accuracies.

[83] *Hierarchical clustering*.

[84] *sklearn.cluster.AgglomerativeClustering*. en. Page Mar. 2020. URL: `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html` (visited on 03/08/2020)

5.2.3.3 Conclusion of the hierarchical classification

As one can see, the complexity to create both of the above mentioned environments is very high. Also the computing time is doubled in this case (the groups $\texttt{group}_0$ to $\texttt{group}_7$ together contain the same data elements as the pre-classifier). Although the approach seemed to be promising, it is not. The specific accuracy is below average. Performing hierarchical classification for a few groups does not lead to an improvement. On the contrary, both accuracy has decreased. At this point it should be mentioned again that the groups were unbalanced. In the future, one could try a clustering algorithm with equal distribution to see if clustering with equally distributed groups leads to an improvement.

5.2.4 Binary classifiers

A binary classifier knows only two states. Either a property is true or it is not true. For example, if one uses a classifier of the class `salad` to classify an element, this classifier returns the probability returns the probability that this image is salad or not (Equation (21)).

$$\hat{\boldsymbol{\lambda}}_{class_m} = \begin{pmatrix} p_{True} \\ p_{False} \end{pmatrix} \quad \Big| \quad p_{True} + p_{Negative} = 1 \tag{21}$$

For a binary classification with more than two classes one needs a separate classifier for each class. For the used data set `food-50` in this thesis 50 classification models are created. Each of them is able to make a statement about the probability that an item to be classified belongs to its class or not. Although this statement basically corresponds to the statement of a single 50 class classifier used so far, the recognitions are distributed to individual models here. In theory, this should increase the Vapnik–Chervonenkis (VC) dimension. The question that now arises: Is it possible to increase the top-1 model total accuracy by distributing the classification over 50 classifiers?

To answer this question, the existing data set of 50 classes will be modified. For each class a separate data set is created, which consists of the class "True" and the class "False". The class "True" contains all elements of the currently considered class, the class "False" contains all other elements from all other classes. The result is a set of 50 training and validation data sets (`food-50-1` to `food-50-50`), which are trained individually according to the standard setup. At the end you get 50 classification models for each class. An overview of the result of the individual model accuracies can be taken from the table A.4 in the appendix. The individual model accuracies appear to look promising at first view. The worst model accuracy was achieved by the class `frittata` with an accuracy of 98.47%. The best model accuracy of 100.00% was achieved by the class `french_fries`.

The 50 created classifiers are now applied to the training data as well as to the validation data to obtain $\hat{\boldsymbol{\lambda}}_{class_m}$. From the obtained probability vectors $\hat{\boldsymbol{\lambda}}_{class_m}$ the "True" value is used to determine the actual vector (Equation (22) and (23)).

$$\boldsymbol{\lambda}^*_{class_m} = \begin{pmatrix} p_{True,class_1} \\ p_{True,class_2} \\ \vdots \\ p_{True,class_{50}} \end{pmatrix} \tag{22}$$

$$\boldsymbol{\lambda}_{class_m} = \frac{\boldsymbol{\lambda}^*_{class_m}}{||\boldsymbol{\lambda}^*_{class_m}||} \tag{23}$$

For the data set `food-50` one obtains for the training part 11,913 vectors $\boldsymbol{\lambda}^{train}$ and for the validation part 2,953 vectors $\boldsymbol{\lambda}^{val}$ with the dimension 50. Using the k-nearest neighbors classification algorithm[85] the model accuracy is now determined. A class assignment of $\boldsymbol{\lambda}^{val}$ is determined considering its next k neighbours of $\boldsymbol{\lambda}^{train}$. The resulting model accuracies at different k values are shown in the following table (Table 3).

[85] *k-nearest neighbors algorithm.*

k	top-1 acc.	k	top-1 acc.	k	top-1 acc.	k	top-1 acc.	k	top-1 acc.
1	73.42%	2	72.98%	3	74.84%	4	75.18%	5	75.14%
6	75.28%	7	75.35%	8	75.45%	9	75.75%	10	75.65%
11	75.55%	12	75.58%	13	75.58%	14	75.52%	15	75.52%
16	75.52%	17	75.28%	18	75.58%	19	75.45%	20	75.62%

Table 3: Comparison of the model accuracy at different k values.

The best result with a model accuracy of 75.75% was achieved with $k = 9$. Above or below this the accuracy decreases. The complexity of building models to create binary classifiers is very extensive and expensive compared to other methods. It requires 50 times more computing time to create the models compared to a single model, whereas the model accuracy that can be achieved is relatively low. For the data set `food-50` this experimental setup did not show any improvements. On the contrary, the result is actually below the expectations. Binary classifications are not well suited to improve the model accuracy of small data sets.

6 Summary and outlook

During this thesis different hyperparameters and techniques for training and validating of image classifiers were considered and compared. The training was performed with the default settings from the chapter "Default setup".

It could be demonstrated that a model accuracy up to 85.60% was achieved with an unbalanced, manually labelled data set consisting of 11,913 training elements and 2,953 validation elements. This required up to 10 hours of computing time on a NVIDIA GeForce GTX 1060 with 6 GByte of memory. With an average of two and a half hours of computing time for 21 training epochs, model accuracies of up to 83.59% were achieved.

The choice for most experiments is based on the CNN model `InceptionV3` (model accuracy 83.59%), although the CNN model `Densenet201` is 1% ahead with 84.58% model accuracy (see chapter "Comparison of different CNN models"). However, with `InceptionV3` it was possible to achieve good results with less memory consumption in average time, while `Densenet201` works significantly slower. For future investigations and with much more GPU performance the experiments mentioned here could be repeated with much larger CNN models like `Densenet201`. As an idea the GPU parallel computing technology should be mentioned here.[86,87]

It could also be shown that the accuracy of the model could be permanently improved with an increasing number of data elements (see chapter "Influence of number of trained images on accuracy"). It seems that the maximum model accuracy with 11,913 training images has not yet been reached at the end (see also chapter "Data augmentation"). It should be further investigated if further increasing data sets will improve the model accuracy and from which point the model accuracy does not or can not increase any more. Why were these images classified incorrectly? Is it perhaps not possible to differentiate these images? And should they therefore be removed from the data set?

In this thesis the models were trained with the default settings from the chapter "Default setup". It could be shown that many of the hyper parameters and training properties mentioned there can be used as a basic for the first model to achieve good model accuracy: The number of learning epochs, the activation function, the training of all CNN layers, the use of the TL approach and the use of the complete data set. There are exceptions where improvements can be achieved if they are tried out through comparative training. It should be mentioned that no big jumps are expected in these parameters compared to the standard setup. Examples are the momentum, the value for reducing the learning rate during epochs and the batch size. An interesting article on the hyperparameter batch size from 2018 brings an interesting approach that could also be included in future investigations: "Don't decay the learning rate, increase the batch size".[88]

Additionally, optimization techniques were investigated in this thesis. Some of them worked well within some limits. Other ideas seemed promising at first sight, but in the end they were not. One of the techniques with potential for improvement is data augmentation (see chapter "Data augmentation"). Investing more computing time is recommended here to improve the model accuracy a bit more. If the balanced data set has also influenced the model accuracy, should be investigated further. It could not be clarified completely. Also in which dimension unbalanced data sets influence the accuracy of the model is a point that can be included in future investigations. Current CNNs are good in performing classifications such as images. An implementation of hierarchical levels or binary classifications is not necessary here, even if it sounds logical at first sight and the data set, like the data set `food-50` used here, consists of only a few classes. It was also shown that it is sufficient to add a simple DL with the size of the number of classes to be classified at the end of the CNN. A dropout layer is not necessary.

Image classifications are a hot topic in business and science. The desire for increasingly higher model accuracies is huge, so many ideas and approaches for improvements are arising here as well.

[86] Sujatha R Upadhyaya. "Parallel approaches to machine learning—A comprehensive survey". In: *Journal of Parallel and Distributed Computing* 73.3 (2013), pp. 284–292. URL: `https://www.sciencedirect.com/science/article/abs/pii/S0743731512002705`.

[87] Stefano Cavuoti et al. "Genetic algorithm modeling with GPU parallel computing technology". In: *Neural Nets and Surroundings*. Springer, 2013, pp. 29–39. URL: `https://link.springer.com/chapter/10.1007/978-3-642-35467-0_4`.

[88] **smith2017don**.

For instance, how about enriching the data set mentioned here with additional data sets from other sources, like Flickr or Google? All this together with data augmentation?

List of acronyms

acc.	accuracy
AI	artificial intelligence
API	application programming interface
ANN	artificial neural network
CNN	convolutional neural network
CPU	central processing unit
DA	data augmentation
DL	deep learning
DNN	deep neural network
DS	data set
FN	false negative
FP	false positive
FCL	fully connected layer
GPU	graphics processing unit
HP	hyperparameter
ML	machine learning
LE	learning epochs
NN	neural network
ReLU	rectified linear unit
RL	reinforcement learning
SGD	stochastic gradient descent
TD	training data
TN	true negative
TP	true positive
TL	transfer learning
VC	Vapnik–Chervonenkis
VCD	Vapnik–Chervonenkis dimension
VD	validation data

List of literature

Aktivierungsfunktionen, ihre Arten und Verwendungsmöglichkeiten. de-DE. Library Catalog: www.ai-united.de Section: Mathematik. Jan. 2019. URL: https://www.ai-united.de/aktivierungsfunktionen-ihre-arten-und-verwendungsmoeglichkeiten/ (visited on 02/28/2020).

Alom, Md Zahangir et al. "The history began from alexnet: A comprehensive survey on deep learning approaches". In: 2018. Chap. TABLE II. The top-5% errors, p. 15. URL: https://arxiv.org/pdf/1803.01164.pdf.

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. 2001, pp. 26–33. URL: https://www.aclweb.org/anthology/P01-1005.pdf.

Basha, SH Shabbeer et al. "Impact of fully connected layers on performance of convolutional neural networks for image classification". In: *Neurocomputing* 378 (2020), pp. 112–119. URL: https://arxiv.org/pdf/1902.02771.pdf.

Cavuoti, Stefano et al. "Genetic algorithm modeling with GPU parallel computing technology". In: *Neural Nets and Surroundings*. Springer, 2013, pp. 29–39. URL: https://link.springer.com/chapter/10.1007/978-3-642-35467-0_4.

Chollet, Francois. "Building powerful image classification models using very little data". In: *Keras Blog* (2016). URL: http://deeplearning.lipingyang.org/wp-content/uploads/2016/12/Building-powerful-image-classification-models-using-very-little-data.pdf.

Confusion matrix. en. Page Version ID: 940280604. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=940280604 (visited on 02/28/2020).

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).

De Mantaras, Ramon Lopez and Eva Armengol. "Machine learning from examples: Inductive and Lazy methods". In: *Data & Knowledge Engineering* 25.1-2 (1998), pp. 99–123. URL: https://www.sciencedirect.com/science/article/pii/S0169023X97000530/pdf.

Deep learning. en. Page Version ID: 942561541. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Deep_learning&oldid=942561541 (visited on 02/28/2020).

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

Dropout (neural networks). en. Page Version ID: 901427811. June 2019. URL: https://en.wikipedia.org/w/index.php?title=Dropout_(neural_networks)&oldid=901427811 (visited on 03/06/2020).

Epochs, Batch Size, & Iterations. Library Catalog: docs.paperspace.com. URL: https://docs.paperspace.com/machine-learning/wiki/epoch (visited on 02/29/2020).

Expectation–maximization algorithm. en. Page Version ID: 936223068. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Expectation%E2%80%93maximization_algorithm&oldid=936223068 (visited on 02/28/2020).

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*. 2004, pp. 133–137. URL: https://graphics.stanford.edu/papers/gpumatrixmult/gpumatrixmult.pdf.

Flowers Recognition. en. Library Catalog: www.kaggle.com. URL: https://kaggle.com/alxmamaev/flowers-recognition (visited on 02/29/2020).

Géron, Aurélien. "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. 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. Convolutional Neural Networks, pp. 359–384. 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. URL: https://static.googleusercontent.com/media/research.google.com/de//pubs/archive/35179.pdf.

Hempel, Björn. *Keras Machine Learning Framework*. original-date: 2019-09-12T21:51:11Z. Feb. 2020. URL: https://github.com/bjoern-hempel/keras-machine-learning-framework (visited on 02/29/2020).

— *Keras Machine Learning Framework - Arguments of the training process*. en. Library Catalog: github.com. URL: https://github.com/bjoern-hempel/keras-machine-learning-framework (visited on 02/29/2020).

— *Keras Machine Learning Framework - GPU vs CPU*. en. Library Catalog: github.com. URL: https://github.com/bjoern-hempel/keras-machine-learning-framework (visited on 02/29/2020).

Hierarchical clustering. en. Page Version ID: 934548831. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Hierarchical_clustering&oldid=934548831 (visited on 02/28/2020).

ImageNet. en. Page Version ID: 929993952. Dec. 2019. URL: https://en.wikipedia.org/w/index.php?title=ImageNet&oldid=929993952 (visited on 02/28/2020).

Jabbar, H and Rafiqul Zaman Khan. "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)". In: *Computer Science, Communication and Instrumentation Devices* (2015). URL: https://www.researchgate.net/profile/Haider_Allamy/publication/295198699_METHODS_TO_AVOID_OVER-FITTING_AND_UNDER-FITTING_IN_SUPERVISED_MACHINE_LEARNING_COMPARATIVE_STUDY/links/56c8253f08aee3cee53a3707.pdf.

k-means clustering. en. Page Version ID: 942500957. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=942500957 (visited on 02/28/2020).

k-nearest neighbors algorithm. en. Page Version ID: 942113305. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=942113305 (visited on 02/28/2020).

Kernel (image processing). en. Page Version ID: 929690058. Dec. 2019. URL: https://en.wikipedia.org/w/index.php?title=Kernel_(image_processing)&oldid=929690058 (visited on 03/07/2020).

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

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).

Linear regression. en. Page Version ID: 935782381. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Linear_regression&oldid=935782381 (visited on 02/28/2020).

Logistic regression. en. Page Version ID: 941157282. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Logistic_regression&oldid=941157282 (visited on 02/28/2020).

Machine learning. en. Page Version ID: 942989288. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=942989288 (visited on 02/28/2020).

Ning, Huan. "Prototyping a Social Media Flooding Photo Screening System Based on Deep Learning and Crowdsourcing". In: 2019. Chap. 2.3 Image classification based on deep learning, pp. 13–15. URL: https://scholarcommons.sc.edu/cgi/viewcontent.cgi?article=6207&context=etd.

O'Shea, Keiron and Ryan Nash. *An introduction to convolutional neural networks*. 2015. URL: https://arxiv.org/pdf/1511.08458.pdf.

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).

Perez, Luis and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning". In: *arXiv preprint arXiv:1712.04621* (2017). URL: https://arxiv.org/pdf/1712.04621.pdf.

Radiuk, Pavlo M. "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets". In: *Information Technology and Management Science* 20.1 (2017), pp. 20–24. URL: https://www.degruyter.com/downloadpdf/j/itms.2017.20.issue-1/itms-2017-0003/itms-2017-0003.pdf.

Random forest. en. Page Version ID: 938369502. Jan. 2020. URL: https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=938369502 (visited on 02/28/2020).

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).

Seif, George. *Deep Learning for Image Recognition: why it's challenging, where we've been, and what's next*. en. Library Catalog: towardsdatascience.com. May 2019. URL: https://towardsdatascience.com/deep-learning-for-image-classification-why-its-challenging-where-we-ve-been-and-what-s-next-93b56948fcef (visited on 02/28/2020).

Softmax function. en. Page Version ID: 928536872. Nov. 2019. URL: https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=928536872 (visited on 03/03/2020).

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*. 2014, pp. 1891–1898. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Sun_Deep_Learning_Face_2014_CVPR_paper.pdf.

Support-vector machine. en. Page Version ID: 942477636. Feb. 2020. URL: https://en.wikipedia.org/w/index.php?title=Support-vector_machine&oldid=942477636 (visited on 02/28/2020).

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).

Upadhyaya, Sujatha R. "Parallel approaches to machine learning—A comprehensive survey". In: *Journal of Parallel and Distributed Computing* 73.3 (2013), pp. 284–292. URL: https://www.sciencedirect.com/science/article/abs/pii/S0743731512002705.

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).

Verlustfunktion (Statistik). de. Page Version ID: 177095272. May 2018. URL: https://de.wikipedia.org/w/index.php?title=Verlustfunktion_(Statistik)&oldid=177095272 (visited on 02/26/2020).

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).

Wu, Jianxin. "Introduction to convolutional neural networks". In: vol. 5. 2017. Chap. 6.1 What is convolution?, pp. 11–13. URL: https://pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf.

— "Introduction to convolutional neural networks". In: vol. 5. 2017. Chap. 6.3 Convolution as matrix product, pp. 15–17. URL: https://pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf.

— "Introduction to convolutional neural networks". In: *National Key Lab for Novel Software Technology. Nanjing University. China* 5 (2017), p. 23. URL: https://pdfs.semanticscholar.org/450c/a19932fcef1ca6d0442cbf52fec38fb9d1e5.pdf.

A Appendix

A.1 Performance comparison between GPU and CPU

Training device	Time to load the model	Training time	Training factor[89]	Time to save the model
NVIDIA GeForce GTX 1060 6GB (Desktop) - Windows 10	17.5s	303.5s 00:05:03.5	1.00x	33.8s
NVIDIA GeForce GT 750M 2GB (Notebook) - Windows 10	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 A.1: Performance comparison between GPU and CPU

A.2 Number of training and validation files

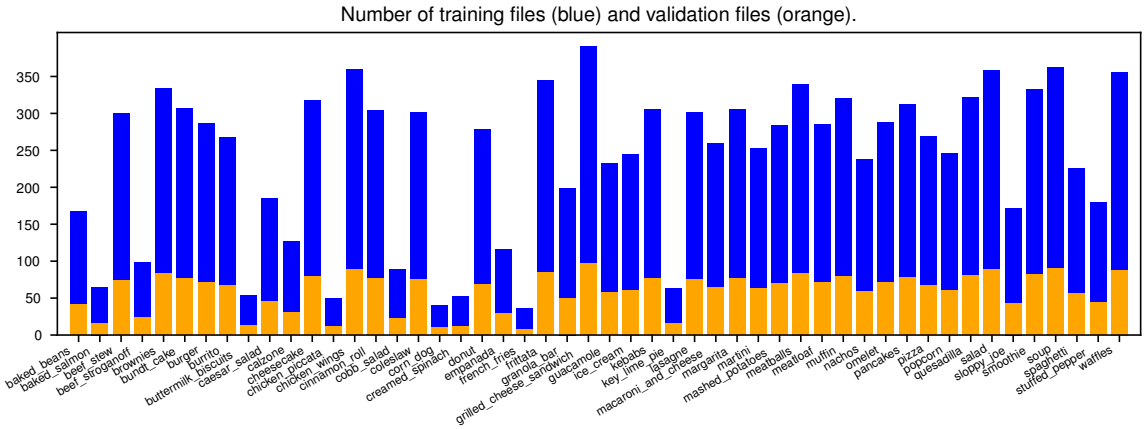

Figure A.1: Number of training and validation files.

[89] Depending on the best result, which is marked with factor 1.00x

A.3 Example of a dropout layer after the CNN model (Python code example)

```
# definition of some parameters
num_classes = 50
dim=299
dropout=0.5

# get the transfer learning model, add dropout and and build the final
# model
base_model = InceptionV3(input_shape=(dim, dim, 3), weights='imagenet',
                         include_top=False)
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(dropout)(x)
predictions = Dense(num_classes, activation= 'softmax')(x)
model = Model(inputs = base_model.input, outputs = predictions)
```

Listing A.1: Example of a dropout layer after the CNN model

A.4 Principal component analysis of the `food-50` model

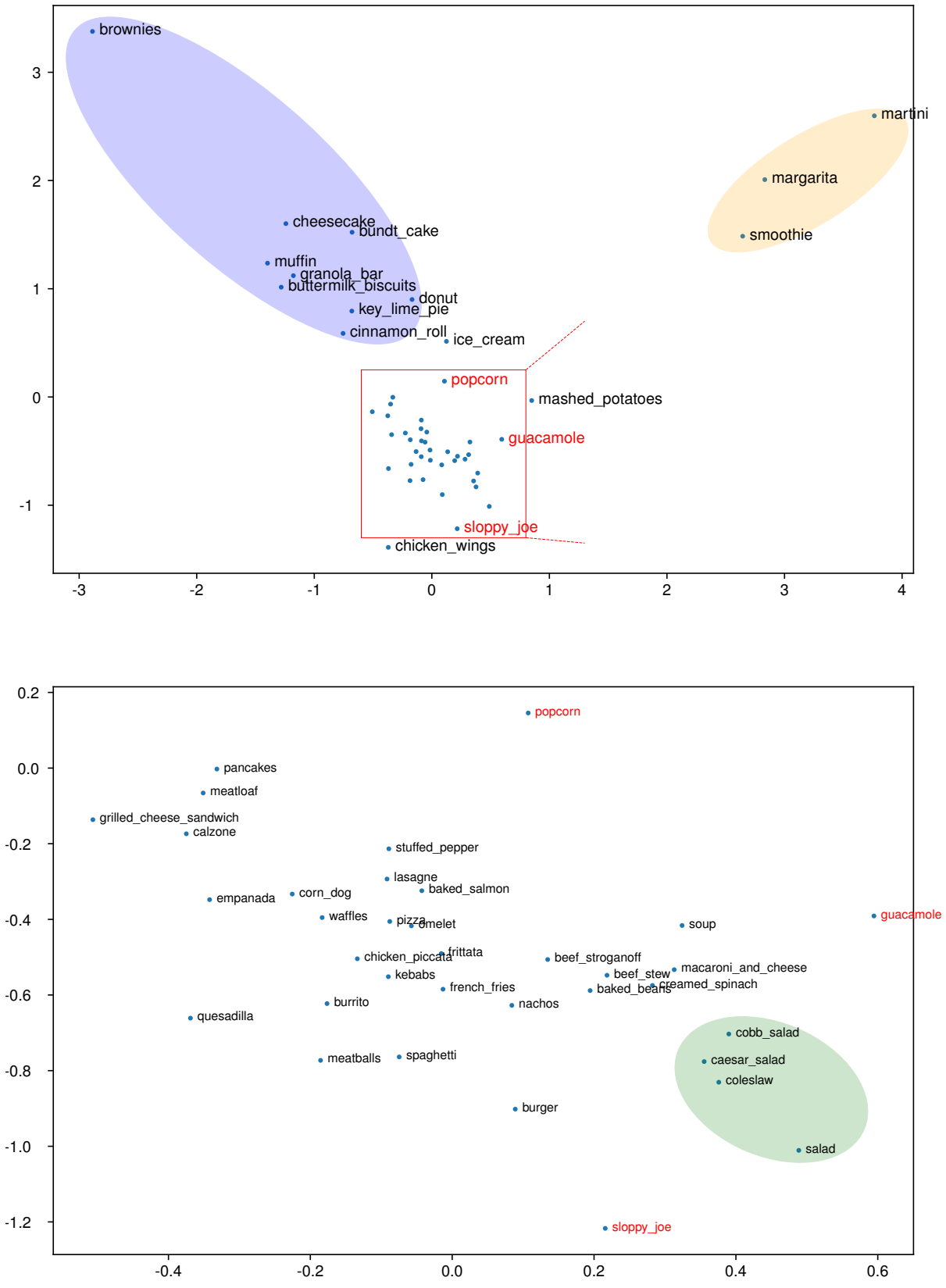

Figure A.2: Principal component analysis of the `food-50` model. The x and y axis stand for the space and the similarity between the classes.

.

A.5 Visualised representation of grouped classes with *k*-means

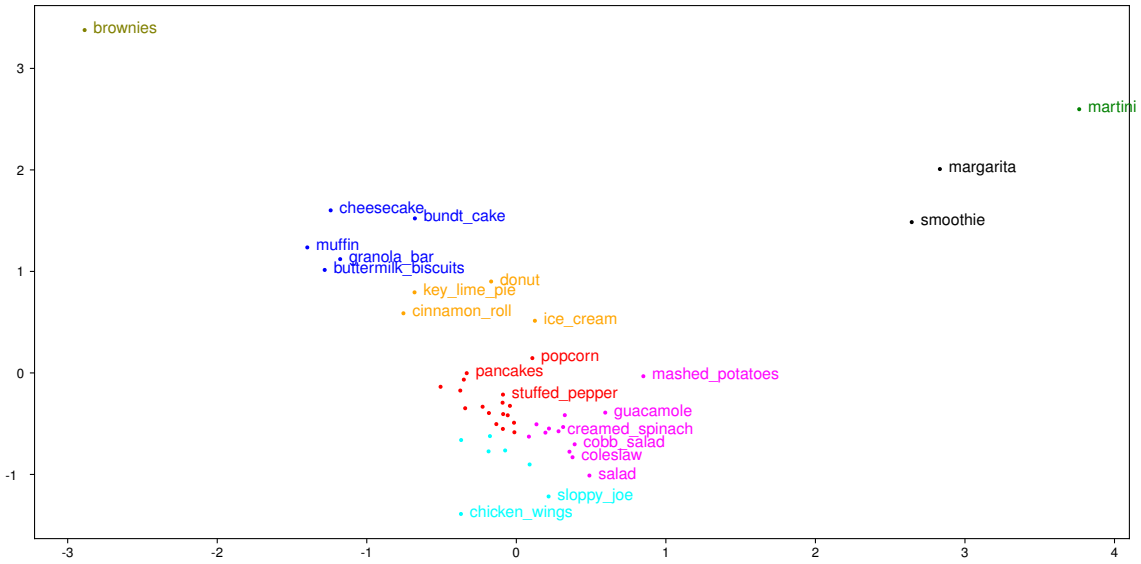

Figure A.3: Grouped classes with k-means

A.6 Model accuracy of grouped classes with *k*-means

Group	$\#_{classes}$	$\#_{files_{val}}$	$ratio_{files_{val}}$	acc^*	$\widehat{acc}^*$	**Classes**
preclassification	8	2,953	100.0%	90.02% @ ep. 15		$group_0$, $group_1$, $group_2$, $group_3$, $group_4$, $group_5$, $group_6$, $group_7$
$group_0$	18	919	31.1%	83.94% @ ep. 11	75.56%	baked_salmon, calzone, chicken_piccata, corn_dog, empanada, french_fries, frittata, grilled_cheese_sandwich, kebabs, lasagne, meatloaf, omelet, pancakes, pizza, popcorn, stuffed_pepper, waffles
$group_1$	2	158	5.4%	99.30% @ ep. 10	89.39%	margarita, smoothie
$group_2$	5	296	10.0%	93.93% @ ep. 16	84.56%	bundt_cake, buttermilk_biscuits, cheesecake, granola_bar, muffin
$group_3$	1	83	2,8%	100.0%	90.02%	brownies
$group_4$	1	63	2,1%	100.0%	90.02%	martini
$group_5$	4	221	7.5%	92.68% @ ep. 17	83.43%	cinnamon_roll, donut, ice_cream, key_lime_pie
$group_6$	12	723	24.5%	81.90% @ ep. 10	73.73%	baked_beans, beef_stew, beef_stroganoff, caesar_salad, cobb_salad, coleslaw, creamed_spinach, guacamole, macaroni_and_cheese, mashed_potatoes, nachos, salad, soup
$group_7$	7	490	16.6%	91.77% @ ep. 14	82.61%	burger, burrito, chicken_wings, meatballs, quesadilla, sloppy_joe, spaghetti
Overall	50	2,953	100,0%			acc_{top-1} = **79.23%**

Table A.2: Grouped classes with *k*-means. *=All accuracies *acc* are top-1 accuracies.

A.7 Visualised representation of grouped classes with agglomerative hierarchical clustering

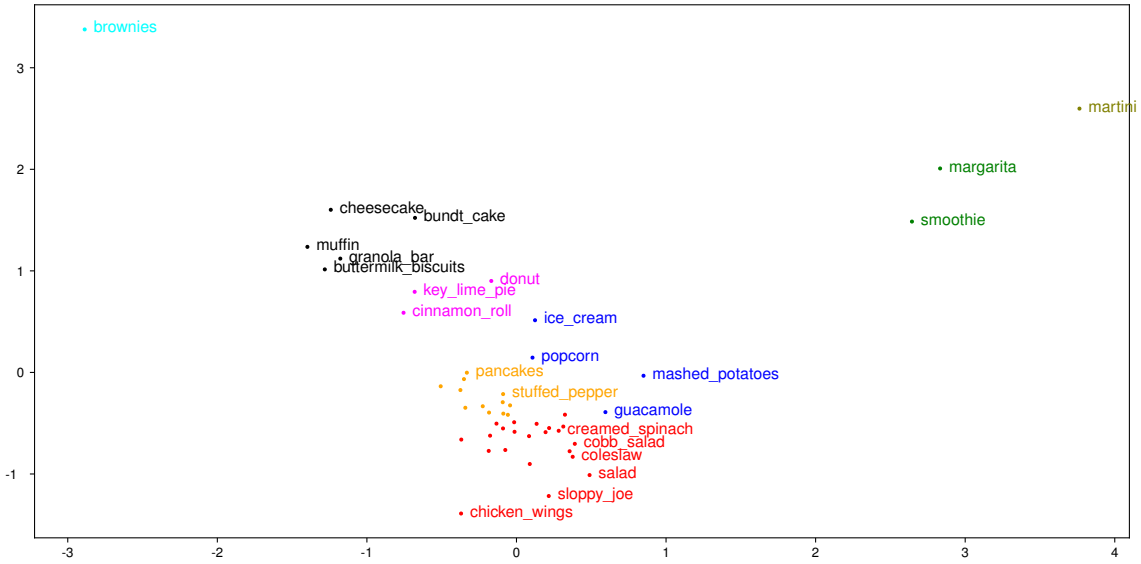

Figure A.4: Grouped classes with agglomerative hierarchical clustering

A.8 Model accuracy of grouped classes with agglomerative hierarchical clustering

Group	$\#_{classes}$	$\#_{files_{val}}$	$ratio_{files_{val}}$	acc^*	$\widehat{acc}^*$	**Classes**
preclassification	8	2,953	100.0%	88.93% @ ep. 14		$group_0$, $group_1$, $group_2$, $group_3$, $group_4$, $group_5$, $group_6$, $group_7$
$group_0$	22	1,267	42.91%	84.17% @ ep. 18	74.85%	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
$group_1$	5	296	10.02%	93.21% @ ep. 16	82.89%	bundt_cake, buttermilk_biscuits, cheesecake, granola_bar, muffin
$group_2$	4	249	8.43%	94.85% @ ep. 11	84.35%	guacamole, ice_cream, mashed_potatoes, popcorn
$group_3$	1	63	2.13%	100.0%	88.93%	martini
$group_4$	2	158	5.35%	99.30% @ ep. 10	88.31%	margarita, smoothie
$group_5$	12	677	22.93%	85.63% @ ep. 11	76.15%	baked_salmon, calzone, corn_dog, empanada, grilled_cheese_sandwich, lasagne, meatloaf, omelet, pancakes, pizza, stuffed_pepper, waffles
$group_6$	3	160	5.42%	93.13% @ ep. 5	82.82%	cinnamon_roll, donut, key_lime_pie
$group_7$	1	83	2.81%	100.0%	88.93%	brownies
Overall	50	2,953	100,0%			acc_{top-1} = **78.60%**

Table A.3: Grouped classes with agglomerative hierarchical clustering. *=All accuracies *acc* are top-1 accuracies.

A.9 Model accuracy of binary classification

		Training				Validation			
Class	**Best epoch**	**Acc. top-1**	**Loss**	**nr. files positive**	**nr. files negative**	**Acc. top-1**	**Loss**	**nr. files positive**	**nr. files negative**
baked_beans	3	99.55%	0.0117	167	11.746	99.66%	0.0138	41	2.912
baked_salmon	1	99.12%	0.0390	65	11.848	99.76%	0.0170	16	2.937
beef_stew	2	98.71%	0.0339	300	11.613	99.11%	0.0339	74	2.879
beef_stroganoff	2	99.38%	0.0151	98	11.815	99.80%	0.0069	24	2.929
brownies	4	99.82%	0.0053	334	11.579	99.22%	0.0261	83	2.870
bundt_cake	2	99.42%	0.0175	307	11.606	99.56%	0.0172	76	2.877
burger	6	99.97%	0.0013	286	11.627	99.69%	0.0178	71	2.882
burrito	4	99.89%	0.0039	268	11.645	99.46%	0.0223	67	2.886
buttermilk_biscuits	1	99.19%	0.0294	53	11.860	99.86%	0.0098	13	2.940
caesar_salad	2	99.28%	0.0216	185	11.728	99.11%	0.0322	46	2.907
calzone	2	99.08%	0.0226	126	11.787	99.42%	0.0154	31	2.922
cheesecake	2	98.74%	0.0334	318	11.595	98.84%	0.0363	79	2.874
chicken_piccata	1	99.24%	0.0311	49	11.864	99.90%	0.0088	12	2.941
chicken_wings	5	99.87%	0.0043	359	11.554	99.22%	0.0246	89	2.864
cinnamon_roll	3	99.77%	0.0075	304	11.609	99.32%	0.0284	76	2.877
cobb_salad	2	99.36%	0.0172	89	11.824	99.69%	0.0124	22	2.931
coleslaw	2	99.05%	0.0278	301	11.612	99.28%	0.0199	75	2.878
corn_dog	1	99.32%	0.0236	40	11.873	99.97%	0.0066	10	2.943
creamed_spinach	1	99.20%	0.0310	52	11.861	99.90%	0.0083	12	2.941
donut	3	99.77%	0.0070	278	11.635	99.49%	0.0213	69	2.884
empanada	2	99.13%	0.0222	116	11.797	99.63%	0.0127	29	2.924
french_fries	1	99.34%	0.0257	36	11.877	100.00%	0.0032	8	2.945
frittata	5	99.70%	0.0080	344	11.569	98.43%	0.0760	85	2.868
granola_bar	2	99.30%	0.0189	199	11.714	99.28%	0.0195	49	2.904
grilled_cheese_sandwich	5	99.76%	0.0070	390	11.523	99.05%	0.0374	97	2.856
guacamole	6	99.92%	0.0034	232	11.681	99.52%	0.0183	57	2.896
ice_cream	6	99.96%	0.0016	245	11.668	99.22%	0.0421	61	2.892
kebabs	3	99.79%	0.0074	305	11.608	99.11%	0.0385	76	2.877
key_lime_pie	1	99.13%	0.0355	63	11.850	99.80%	0.0109	15	2.938
lasagne	5	99.82%	0.0067	301	11.612	99.01%	0.0385	75	2.878
macaroni_and_cheese	2	99.00%	0.0281	259	11.654	99.25%	0.0256	64	2.889
margarita	5	99.92%	0.0030	305	11.608	99.32%	0.0179	76	2.877
martini	2	99.50%	0.0139	252	11.661	99.66%	0.0100	63	2.890
mashed_potatoes	2	99.29%	0.0210	284	11.629	99.15%	0.0290	70	2.883
meatballs	3	99.26%	0.0209	339	11.574	98.81%	0.0410	84	2.869
meatloaf	3	99.65%	0.0105	285	11.628	99.35%	0.0301	71	2.882
muffin	3	99.72%	0.0095	320	11.593	98.98%	0.0430	79	2.874
nachos	3	99.26%	0.0213	238	11.675	98.98%	0.0386	59	2.894
omelet	3	99.14%	0.0256	288	11.625	98.50%	0.0608	71	2.882
pancakes	3	99.75%	0.0080	312	11.601	99.42%	0.0214	78	2.875
pizza	4	99.77%	0.0076	269	11.644	98.74%	0.0575	67	2.886
popcorn	3	99.78%	0.0062	246	11.667	99.66%	0.0096	61	2.892
quesadilla	5	99.82%	0.0047	321	11.592	99.32%	0.0342	80	2.873
salad	6	99.83%	0.0064	358	11.555	98.47%	0.0721	89	2.864
sloppy_joe	6	99.95%	0.0012	172	11.741	99.76%	0.0160	43	2.910
smoothie	3	99.77%	0.0067	332	11.581	99.73%	0.0083	82	2.871
soup	5	99.70%	0.0088	362	11.551	98.50%	0.0625	90	2.863
spaghetti	2	99.30%	0.0183	226	11.687	99.25%	0.0212	56	2.897
stuffed_pepper	3	99.87%	0.0053	180	11.733	99.80%	0.0067	44	2.909
waffles	4	99.92%	0.0033	355	11.558	99.73%	0.0172	88	2.865

Table A.4: Overview of binary classification

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 thesis 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 *Dr. 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 third 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 fourth place was my *family*, who was permanently at my side and supported me with this topic. And last but not least: Thanks to all the people who helped me with corrections and hints for this thesis (even if they were already mentioned). I would like to mention the following persons:

Dr. David Urbansky from semknox. With his experience in this field in general. A lot of his ideas are based on the fact that papers have to deliver a demanding and interesting content. Many ideas and content suggestions would not have been possible without him.

Dr. Christoph Sohrmann from the Fraunhofer Institute. With his experience in writing papers. Many of his ideas are based on the fact that paper has to meet certain expectations in order to be respected in science.

Isabel Benkert from ressourcenmangel. With her experience in english texts and an eye for the form and appearance of this thesis. I thank her for the little things that I would never have seen myself. English texts are very different from german texts. ;)

Nadine Sander from ressourcemangel, with her experience for pragmatism and an eye for things that only make sense if they serve a purpose. I thank her for all english corrections and all hints about the topic "How thesis are seen by others."

Prof. Dr. Angela Thränhardt with the scientific claim and more than 100 written english publications. Revising her comments was the most difficult part for me. I thank her for the way of expression and the claim to describe things scientifically. Sentences are only sentences if they consist of subject and predicate ;)

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* :