Assignment 1

|  |  |
| --- | --- |
| Group 17 | 2021-04-13 |
| Fabian Holzmüller | e01126321 |
| Benjamin Schwendinger | e01225371 |

# What is image classification?

The basic question that "Image Classification" wants to answer is: What does this image show?

There are different approaches to answer this question. For example, keywords can be assigned to the images. A picture could be described with tree, water and mountain. Basically, objects in a picture have to be delimited and recognized. This also makes it possible to separate the objects in the image. Such applications will become essential for many areas, for example autonomous driving to identify road users.

Other possibilities that originate from the subject area of "Image Classification" influence the image itself. In addition to artistically creative possibilities to transform an image into a work of art of famous artists, up to so-called deep fakes, where a rough sketch is enough as a template to digitally recreate famous personalities, many topics are open. Even the motion recognition of people originates from the discipline of "Image Classification".

The basis of this technology is to assign a label to an image from a predefined selection. In the beginning it is important that an image can be assigned to exactly one class. Various difficulties arise in this process:

* **Pose and angle of view.** Each different pose and angle changes the image for the algorithm and makes it more difficult for it to recognize and assign the content.
* **Illumination** describes the difficulty of classifying due to poor lighting conditions.
* **Deformation** life forms tend not to always take the same structural shape and are therefore more difficult to recognize.
* **Occlusion,** describes the problem that objects on images are not always fully displayed, but the algorithm still has to recognize them.
* **Backgrounds** are especially problematic when the object is difficult to separate from it.
* **Intraclass variation**. Even images from and of the same class do not always look identical or comparable.

Despite these problems, images can be well described with new methods and approaches via "image classifiers".

# What is the purpose of the training, validation, and test sets and why do we need all of them?

In order to construct and subsequently test an "Image Classifier", three data sets are required, that do not build on each other or are connected in any other way.

* The **training set** is used to train the data. For some classifiers all data is stored, for others parameters are estimated.
* The **Validation Set** is used to tune the hyperparameters, since we do not want to fine tune on the test set to prevent information spillover. For example, in the k-NN approach, we search for a suitable value of parameter k. To determine these hyperparameters, there are multiple approaches, as e.g. grid search, random search, bayesian optimization based approaches or evoluationary approaches, etc. , but in this lecture we focused on grid search and random search.
* The **test set** is subsequently used to measure the performance of the algorithm. The result of the classification is compared with the real values and thus the accuracy is determined by contrasting the correctly classified values with the wrong ones.

The purpose of the evaluation is to measure the effectiveness of a certain model, hence, we want to estimate how well our model performs on new unseen data. If the tasks are executed on the same sets this can lead to problems. Decision trees or perceptrons on linear separable data can achieve 100% accuracy on the training data, while we have no real estimate of how well they would work on unseen data.

# How do nearest neighbour and linear classifiers work?

The basis for these simple classifiers is the representation of the image as a vector. The individual pixel values are transferred into a high-dimensional vector according to index and colour. Hence, we map images to points in a high dimensional space.

The nearest neighbour classifier uses this high dimensional representation to calculate the distance from a new given sample to all training samples. We can use various metrics, such as the Euclidean metric, the Manhattan Metric or other Minkowski Metrics for the distance calculation. To speedup this distance calculation process, various data structures such as, e.g. KD-Trees have been proposed. A new point is then classified by a weighted voting of the k closest (regarding the calculated distances) points of the training samples. Most of the times these weights are chosen uniformly or as the reciprocal of the distance to give more value to the votes of closer points. An advantage of k-NN is the fast training phase, since only the training data has to be stored. Major disadvantages arise when large datasets are used for training, since all training data has to be stored and the inference process might be slow.

The linear classifier is based on the formula:

**W**x + b = f

where **W** is a matrix of dimension (number of possible classes | vector dimension of the image vector), x is the image vector, b is a bias vector and f is the result vector. Hence, it is an affine function from the n-dimensional input space of the image vector samples to the m-dimensional output space of m classes.

To calculate W and b, a cost function is set up which is minimized by an optimization algorithm (e.g., iterative methods such as gradient descent, quasi-newton methods or interior point methods) in the training set. In this minimization, some hyperparameters can be specified, such as the step length in the direction of the largest negative partial derivative. Geometrically we interpret **W**x+b as a set of m (=number of possible classes) hyperplanes, where each hyperplane acts as an independent binary classifier. For the class determination then e.g. the highest value from the solution vector f can be taken. The advantage of linear classifiers clearly lies in the fast inference speed since only a single matrix multiplication has to be performed (if we apply the bias trick, where the bias vector is integrated into **W**).

# Results of simpe\_cats\_dogs.py

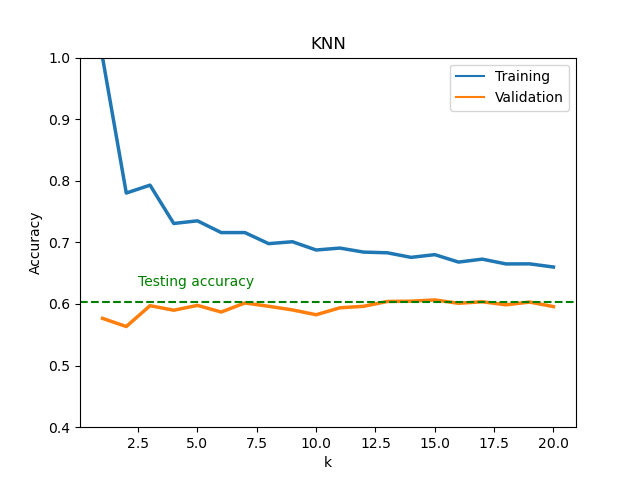
Dataset: The CIFAR-10 datasets consists of 60,000 labelled 32x32x3 images, which can be divided into 10 distinct classes. Since each class contains 6,000 images and we only consider the classes “cat” and “dog”, we end up with 12,000 images. As we can see from the distributions across training, validation and test sets (see also Figure 1), we have slightly more picture of cats in our training set and gradually more picture of dogs in our validation set. Thus, a naive dummy classifier would predict cats with a probability greater than 50% for the training set and vice versa for the validation set.

Figure - Distributions across datasets

The task at hand is to build an image classifier that can distinct pictures of cats and dogs. We implemented two different classifiers, namely a KNN classifier and a linear classifier.

For the KNN classifier we used the parameter k as hyperparameter and tuned it with a grid search in a range of [1,20].

For the linear classifier we used the penalty constant alpha as a hyperparameter and tuned it with a random search sampling 20 points each from a set of 1000 points of a logarithmically

Figure – Comparison training and validation accuracy for KNN

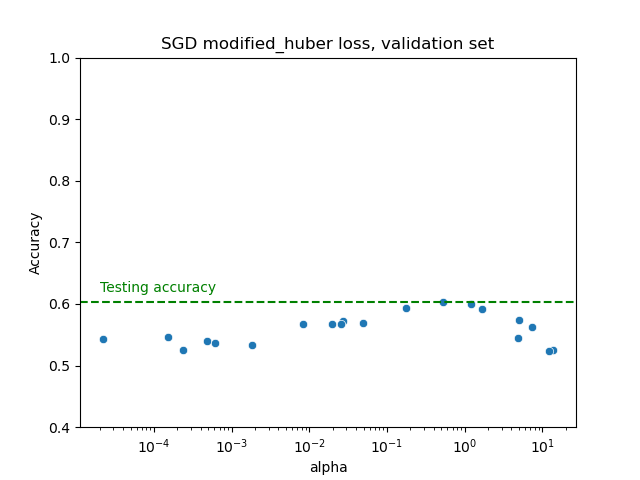
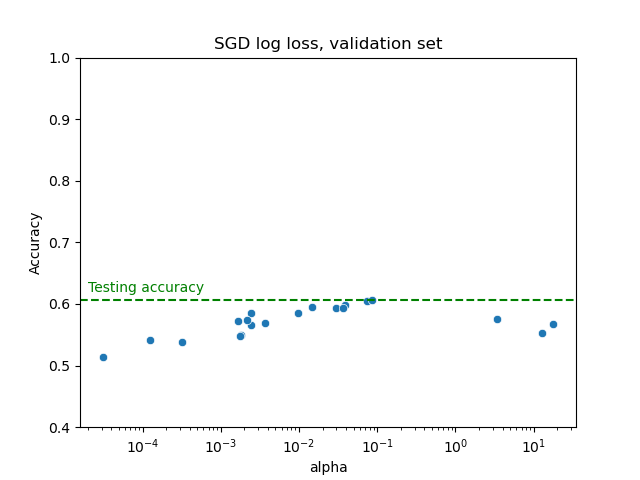


Figure - Linear classifier accuracies on validation set for logistic loss

Figure – Linear classifier accuracies on validation set for modified\_huber loss

spaced interval in the range of [1e-5, 1e1.5]. Moreover, we performed this random search for two different losses namely “log” which corresponds to logistic regression and the “modified\_huber” loss.

For the KNN classifier we found that k=15 achieved the best results on the validation set with an accuracy of 60.66% on the validation set and an accuracy of 60.40% on the test set. Here we can see a vast discrepancy between the accuracy on the training set and the accuracy on the validation set. Moreover, it can be observed (see also Figure 2) that as k increases, the accuracy of the training and validation set converge and stabilize.

The linear classifier worked best with the logistic loss function and an alpha hyperparameter of 0.0864, achieving an accuracy of 60.61% for the validation set and 60.20% for the test set.

Figure -Complete hyperparameter results for training and validation set

In total we can say that both classifier achieved similar results according to accuracy. The full results results of the hyperparameter tuning can be found in Figure 5.