# Introduction to Machine Learning (SS 2021)
# Programming Project

**Author 1**
Last name: Gründlinger
First name: Diana
Matrikel Nr.: 01480296

**Author 2**
Last name: Huber
First name: Marcel Alexander
Matrikel Nr.: 11909190

## I. Introduction

Out of the three given options, we chose the "Sign Language" assignment for our programming project. The objective of this task is to construct a machine learning model that is able to accurately classify given images of hand gestures which express letters or numbers in sign language.

The dataset consists of 9680 samples, each containing exactly one grayscale image of one hand sign. Each pixel in a 128x128 picture, having a certain brightness value, represents a feature. The brightness level of the images is slightly inconsistent and the pictures suffer from small irregularities (e.g. missing pixels). There are a total of 36 different classes, namely letters (a-z) and digits (0-9). The data is somewhat imbalanced for certain classes, as there are between 100 and 600 samples contained in the dataset for each class.

## II. Implementation / ML Process

Our machine learning method of choice is the artificial neural network (ANN). According to virtually every source we could find on the internet [4], neural networks are the go-to approach to perform serious image recognition. Artificial neural networks can be seen as an interconnected group of nodes, inspired by a simplification of neurons in a brain. ANNs are very powerful because they provide a way of constructing non-linear decision boundaries easily and efficiently, even for data with lots of features. In comparison to other methods (e.g. SVM), ANNs are more compact and faster to evaluate [1]. It is worth mentioning that there are more advanced neural network algorithms, like convolutional neural networks (CNN) or recurrent neural networks (RNN). Especially CNNs could be better suited for our image recognition problem [2] because they implicitly perform feature selection (filtering unnecessary features, e.g. black pixels), but since we did not cover this algorithm in depth in the lecture, we decided to stick with the simpler ANNs.

To implement the artificial neural network we use PyTorch, which is an open source machine learning framework that provides the most important tools to quickly create and train an ANN. Our particular network has one hidden layer, in which ReLUs are used as activation functions. The size of the input layer is equal to the amount of pixels in one image, and each neuron in the output layer corresponds to one of the 36 classes.

As for preprocessing, we need to normalize the brightness values of the images to a range from 0 to 1 in order for PyTorch to accept our data. To achieve this we simply divide each brightness value by the maximum brightness value of 255.

For choosing values for the hyperparameters, we first made an educated guess based on what we learned in the lecture and the proseminar notebook on ANNs. Then we performed "grid search", i.e. tried out different values for all parameters in a predefined range. The choice of range was based on experiments that we performed manually beforehand. To measure the performance of the network with different parameters and to ensure that our artificial neural network performs well, we split the dataset randomly into a training set and test set and evaluated the accuracy, i.e. the number of samples that got labelled correctly, on the test set after each iteration in grid search. Due to limited computational resources we had to keep the search space rather small. Still, one run of grid search already took us 6 hours. Having said that, we are confident that the search space we defined is sufficient for this application.

The hyperparameters for the final model have the following values:

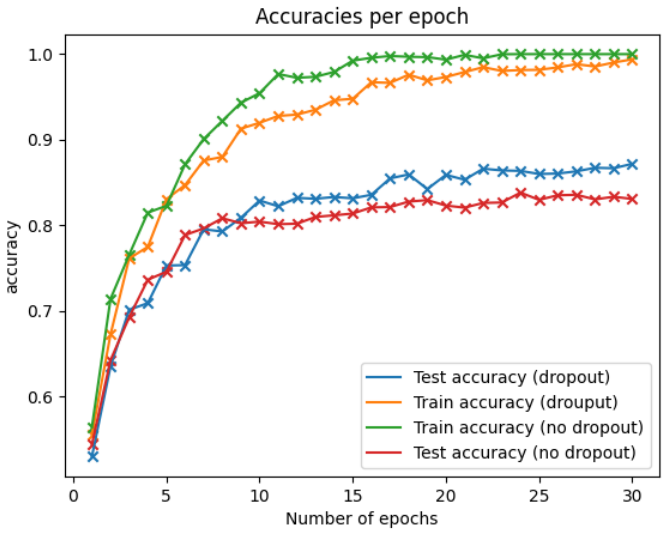
- Learning Rate: 0.0065
- Moment 0.94
- Hidden layer size: 400
- Number of epochs: 25

To decrease the chance of overfitting we decided to introduce "dropout" into our neural network. Dropout basically zeros some values, randomly by a certain chance, before inputting them into the next layer. Since we did not learn this in the lecture, we used an online resource [3] as guidance. Our experiments lead us towards a dropout rate of 25%.

## III. Results

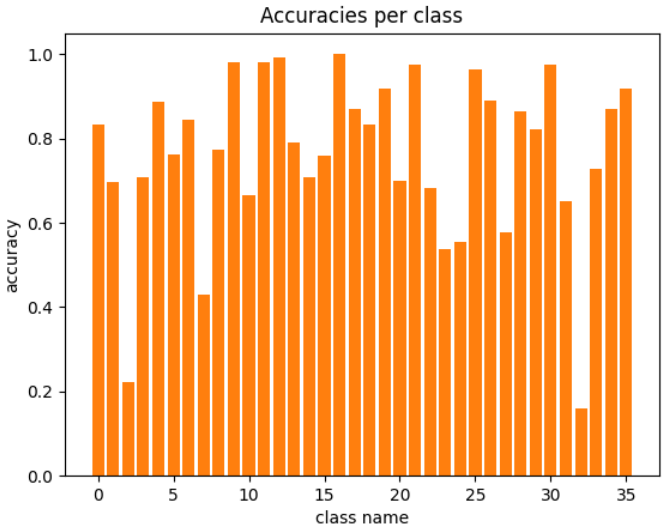
The following plot shows both test accuracy and train accuracy after a certain number of epochs, both with and without dropout activated. For this run-through, we used the optimal parameters we found in grid search and partitioned

the dataset into 80% training data and 20% test data.




As one can observe in the plot, we reach a test accuracy of around 85%. One can also see that the version without dropout reaches a higher train accuracy quicker, while resulting in a test accuracy that is worse than in the version with dropout. Also, there is not much benefit of using more than 25 epochs since neither the test accuracy nor the train accuracy increase after that point.

One more thing worth mentioning is the accuracy per class, shown in the following plot.




It seems that the original data imbalances do not have a significant impact on the accuracy of the model. The variation in the accuracy value per class is more likely due to imbalances produced by the random splitting of the dataset into test dataset and training dataset.

## IV. Discussion

It was pretty clear for the start that we would choose a neural network, however, we attempted different approaches to determine whether our assumption that neural networks are the most suitable type of model for image recognition is correct. With the help of Scikit we were able to swiftly build different types of models and evaluate the image recognition performance of said models. Among our choice of reference models were decision trees and K-nearest neighbour classification. For decision trees the training performance was astonishingly good (100%). However, this classifier had a very poor performance on the test dataset. We tried to resolve this issue by pruning the tree, but this did not yield satisfying results. Moving on to the K-nearest neighbour method, we discovered that the learning process is very fast, and the resulting performance is almost acceptable. Yet, the artificial neural network implementation of Scikit outperformed the other models on the task, which reaffirmed us in our initial decision to use ANNs.

We tried different approaches to potentially improve the performance of our ANN, some of which worked well and some of which did not contribute in a beneficial way.

For example, we tried out using different activation functions in our hidden layer, such as Sigmoid, LogSigmoid, Tanh or LeakyReLU. None of these functions had positive impact on the networks overall performance, so we stayed with ReLU. Furthermore, experimenting with (one or two) additional hidden layers didn't yield any relevant results.

In the preprocessing phase we attempted to balance the number of samples per class in order to improve the training performance. There was also an attempt to improve our normalization by individually normalizing the brightness values for each image with respect to the maximum brightness value of the respective image. Unfortunately, the lack of improvement in performance of our ANN resulted in the rejection of both aforementioned concepts.

To our delight, some minor improvements could be accomplished by introducing the previously described dropout functionality between layers. This concept helped us to reduce overfitting, which in return allowed us to increase the number of epochs, ultimately resulting in a better performance of our model.

In order to optimize the network even further, we anticipate a deep learning approach or a CNN to yield even better results. Since we did not cover these advanced methods in detail in the lecture, we kept from using them, but it would be an interesting experience to experiment with those methods.

## V. Conclusion

Our main takeaway from this machine learning project is that a large part of the process of finding a machine learning solution for a problem is trial and error. Fortunately, with the help of libraries like Scikit we can experiment with different models in a convenient way, without the overhead of implementing said models ourselves.

Aside from that, we learned that finding the optimal hyperparameters is a difficult and time consuming task which can get quite computationally expensive.

## References


[1] Artificial neural networks. https://lms.uibk.ac.at/auth/RepositoryEntry/4655448784/CourseNode/103304823763900. Accessed: 2021-06-08.

[2] Cnn vs. rnn vs. ann – analyzing 3 types of neural networks in deep learning. https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/. Accessed: 2021-06-08.

[3] Dropout in pytorch – an example. https://wandb.ai/authors/ayusht/reports/Dropout-in-PyTorch-An-Example--VmlldzoxNTgwOTE. Accessed: 2021-06-15.

[4] What is image recognition? https://deepomatic.com/en/what-is-image-recognition. Accessed: 2021-06-08.