June 27, 2020

Machine Learning

Machine Learning Challenge



Contents

[General Information 2](#_Toc43985580)

[Introduction 3](#_Toc43985581)

[Task 1 3](#_Toc43985582)

[Features & Preprocessing 3](#_Toc43985583)

[Learning Model, Algorithms and Methods used 3](#_Toc43985584)

[Parameter Tuning 4](#_Toc43985585)

[Discussion of the results 5](#_Toc43985586)

[Task 2 6](#_Toc43985587)

[Features & Preprocessing 6](#_Toc43985588)

[Learning Model, Algorithms and Methods used 6](#_Toc43985589)

[Parameter Tuning 6](#_Toc43985590)

[Discussion of the results 6](#_Toc43985591)

[References 7](#_Toc43985592)

[Appendix 8](#_Toc43985593)

# General Information

**Group 40**

**Name of all group members:**

Dovile Kliusovaite Anr. Snr

Meret Schädler Anr.: 2050924 Snr: 748225

**Tasks allocation:**

Dovile Kliusovaite Task 2

Meret Schädler Task 1

# Introduction

The purpose of this challenge is to classify and predict the handwritten letters of a subset of the EMNIST dataset which is an extension of the widely known MNIST dataset.

The challenge is split into two tasks:

**Task 1:** “Train a model to recognize handwritten English letter”

**Task 2:** “Predict Top-5-accuracy of a series of 5 letters”

The dataset contains images of handwritten letter (upper and lowercases) of the whole English alphabet and its corresponding label. For example, if the image shows a “W”, the label would be 23 and if we see a “K” it would be 11 (see appendix figure 1). Since the dataset covers the whole alphabet, we have 26 classes (A to Z). Thus, this is a multiclass prediction problem which can be solved with different linear and non-linear algorithms such as logistic regression, random forests, SGD classifier, multi-layer perceptron classifier or with a convolutional neural network. To calculate and compare the accuracies of the different models, we used two packages “sklearn” or “keras”.

The dataset used for this task is not the original EMNIST dataset. It only contains 124’000 images and labels equally distributed over the 26 letters (see appendix figure 2). Moreover, the shape of the input data has already been adjusted. Originally, each picture had to be converted to a 28x28 size image in a gray-scale of 0 – 255 (Alsaafin, A. and Elnagar, A., 2017).

In the past, a lot of models have been built to predict digits from the MNIST dataset which show an accuracy rate of close to 100%. However, the EMNIST dataset has not been explored to this extend yet. A good reference is the paper from Cohen, Afshar, Tapson and van Schaik (2017) which states an accuracy score of 55.78% for the letters of the EMNIST dataset when using a linear classifier and 85.15% +/- 0.12% for the OPIUM classifier (neural network). Our goal was to find a model which shows at least the same accuracy rates as the once stated by Cohen, Afshar, Tapson and van Schaik (2017).

# Task 1

## Features & Preprocessing

As mentioned in the introduction, the size of each image has already been normalized to 28 x 28 pixels what is equal to 784 pixels. These 784 pixels represent the features of the digits. For the multi-layer perceptron, we used this shape while for the convolutional neural network classifier we had to reshape the size to 28 x 28 pixels.

Moreover, in figure 1 (see appendix) it can clearly be seen, that the values (i.e. grey-scale) ranges from 0 (black) to 255 (white). For a neural network the input values have to be close to unity. Thus, we normalized all data points to values between 0 and 1 with the “MinMaxScaler” from “sklearn”.

Before we could start training the model, we changed the labels into binary classes by applying the “LabelBinarizer” from “sklearn” which is a so called “one-hot encoding-procedure”. What this does is changing the 26 categories to a binary matrix. This is needed if we want to train a multi- layer perceptron or a convolutional neural network in “keras”.

## Learning Model, Algorithms and Methods used

For all models, we firstly split the 124’000 images into a training and a test set based on the widely used splitting rule of 80% (train) versus 20% (test). The test set is used to evaluate if the model generalizes well on previously unseen data. In addition, for the MLP and the CNN models (see table 1) we performed a cross validation by splitting the training data into a train (80%) and validation dataset (20%).

We trained the data on the following linear and non-linear models by either using the “sklearn” or “keras”.

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Model** | **Library** | **Parameters1** |
| Linear | Logistic Regression | sklearn | multi\_class = “Multinominal”  C = 10 |
| SGD Classifier | sklearn | loss='log' |
| Non-linear | Random Forest Classifier | sklearn |  |
| Multi-Layer Perceptron (MLP) | sklearn | hidden\_layer\_sizes=(128,)  validation\_fraction=0.1,  early\_stopping=True,  learning\_rate\_init=.001 |
| MLP Keras | keras | see text below |
| Convolutional Neural Network (CNN) | keras |

Table 1 - Models

1 Only mentioned if it deviates from the default

**MLP Keras**

We built a sequential model with two hidden layers. The input dimension for the first layer is 784 (features), for the activation we used “relu” and the output dimension is 256, for the second layer, we only changed the output dimension to 128 and the last layer has an output dimension of 26 (number of letters) and uses the “softmax” activation function which is used for multiclass classification problems. To compile the model, we used the “adam” optimizer with a learning rate of 0.001 and the loss “categorical\_crossentropy”. We ran the model with a batch size of 128, over 5 epochs including a cross-validation based in the training and validation data.

**Convolutional Neural Network**

For the CNN classifier, we built a sequential model with 3 convolutional layers with a filter size of (3,3) and starting with a number of filter (32, 64, 64) followed by max pooling layers (2,2). After the second max pooling layer, the filter maps get flatten to provide features to the classifier. We used the same activation functions, loss function and fitting parameters as for the MLP Keras model but only a batch size of 32.

## Parameter Tuning

In table 3 (see appendix) it can be seen, that the train accuracy score of the CNN model is with around 0.9535 significantly higher compared to the other models. Thus, we decided to only tune some hyperparameters of the CNN model. Namely, the activation function can either be “tanh” or “relu” and the epoche varies between 1, 5 and 10 to find the best fitting model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Activation** | **Epoche** | **Train accuracy** | **Validation accuracy** | **Test accuracy** |
| “relu” | 10 | 0.9696 | 0.9361 | 0.9336 |

Table 2 – Best CNN model - parameter tuning

The test and validation accuracy are close to each other. However, the train accuracy is higher and thus this model might overfits. Thus, the number of epochs should be lower. When the validation accuracy stops increasing the have to stop training the model.

In addition, we also tuned the number of layers and added a dropout layer to see if that would have an impact on the performance. The accuracy didn’t improve, thus we did not further elaborate it.

However, other hyperparameters such as number of layers, number of neurons per layer or number of features by the convolutional layers could have been further adjusted. But this would be too much the purpose of this task.

## Discussion of the Results

To compare the models, we looked at the accuracy scores. As mentioned in the introduction, the classes are balanced, thus the accuracy score is the right metric to use.

Firstly, we compared our results with those for the paper written by Cohen, Afshar, Tapson and van Schaik in 2017 to get an idea if our accuracy scores are in the same range. The accuracy scores presented in the table 3 (see appendix) show higher scores for both the linear classifiers as well as the non-linear classifiers compared to the results of Cohen, Afshar, Tapson and van Schaik.

In table 3 (see appendix) it can be seen that the CNN model has the highest training accuracy of 0.9535 and the smallest gap between the train and test accuracy of 0.9349. However, the lower accuracy on the test versus training set indicates that the model overfits the training data. To find a better fit, we did some parameter tuning as describe above.

Also worth to mention is that the Random Forest Classifier shows a 0.997 accuracy on the training data while the accuracy on the test data is with 0.829 significantly lower. This is clearly an overfitting problem.

The accuracy per class of the CNN model is very high above 0.9 for nearly all labels. Only a few labels seem to be hard to predict – G, I, L, Q. The same can also be seen in the confusion matrix (see figure 4 appendix).

# Task 2

## Features & Preprocessing

xxx

## Learning Model, Algorithms and Methods used

xxx

## Parameter Tuning

xx

## Discussion of the Results

# References

<https://www.machinecurve.com/index.php/2020/02/18/how-to-use-k-fold-cross-validation-with-keras/>

<https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/>

<https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-mnist-handwritten-digit-classification/>

<https://scikit-learn.org/stable/modules/cross_validation.html>

<https://www.kaggle.com/darkside92/simple-best-digit-recognizer-with-cnn-top-5/comments#497794>

<https://medium.com/@mjbhobe/mnist-digits-classification-with-keras-ed6c2374bd0e>

<https://www.codementor.io/@mgalarny/making-your-first-machine-learning-classifier-in-scikit-learn-python-db7d7iqdh>

<https://thedatafrog.com/en/articles/handwritten-digit-recognition-scikit-learn/>

<https://hub.packtpub.com/how-to-classify-digits-using-keras-and-tensorflow/>

<https://www.kaggle.com/raahat98/digit-classification-with-keras-99-accuracy>

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre van Schaik, EMNIST: an extension of MNIST to handwritten letter in: International Joint Conference on Neural Networks (IJCNN), 2017

Alsaafin, A. and Elnagar, A. (2017) A Minimal Subset of Features Using Feature Selection for Handwritten Digit Recognition. Journal of Intelligent Learning Systems and Applications, 9, 55-68.

<https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/>

# Appendix

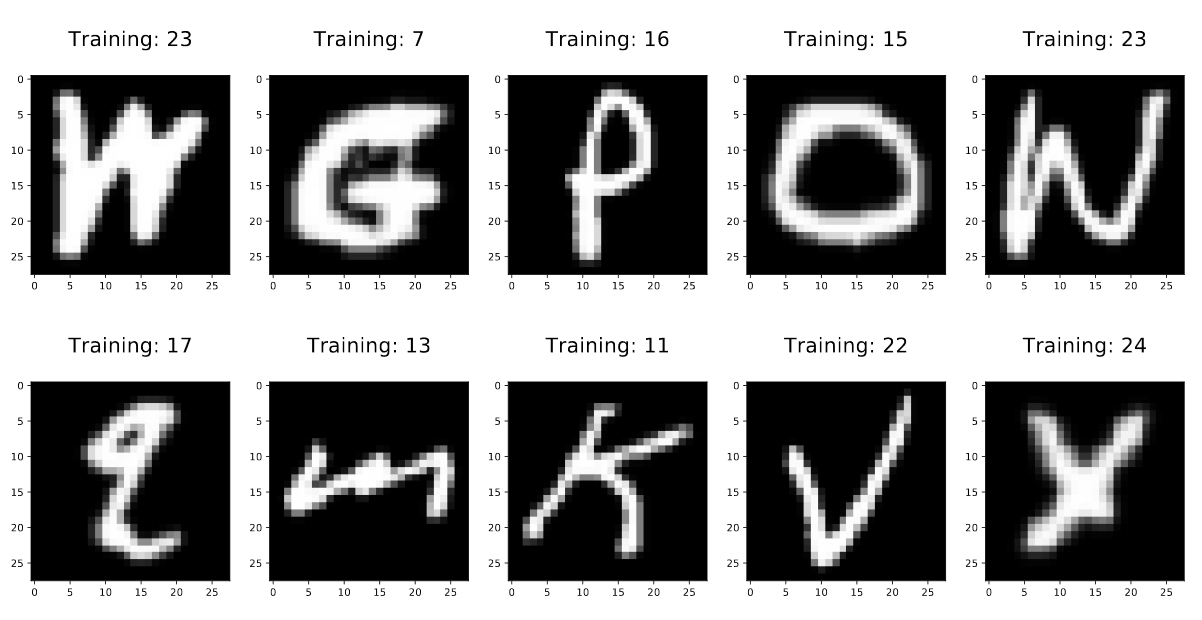


Figure 1 Example labels

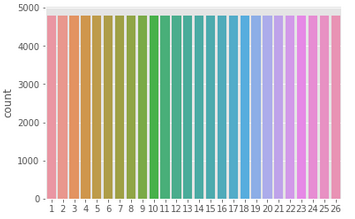


Figure 2 Balanced classes

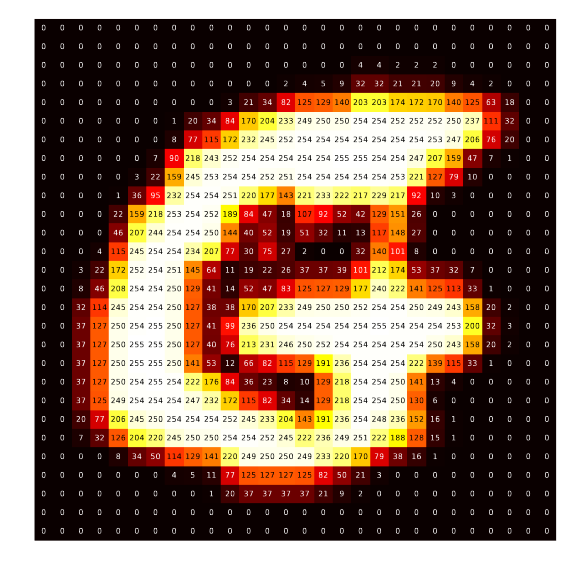


Figure 3 Feature values before normalizing the data points

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Models** | **Train accuracy** | **Val accuracy** | **Test accuracy** | **Comments** |
| Logistic Regression | 0.736 | N/A | 0.71 |  |
| SGD Classifier | 0.65 | N/A | 0.637 |  |
| Random Forest Classifier | 0.997 | N/A | 0.829 | Overfitting, CV needed |
| Multi-Layer Perceptron | 0.926 | N/A | 0.887 | Overfit, parameter tuning needed |
| MLP Keras | 0.9255 | 0.8993 | 0.8940 |
| Convolutional Neural Network | 0.9535 | 0.9327 | 0.9349 |

Table 3 – Accuracy scores - comparison

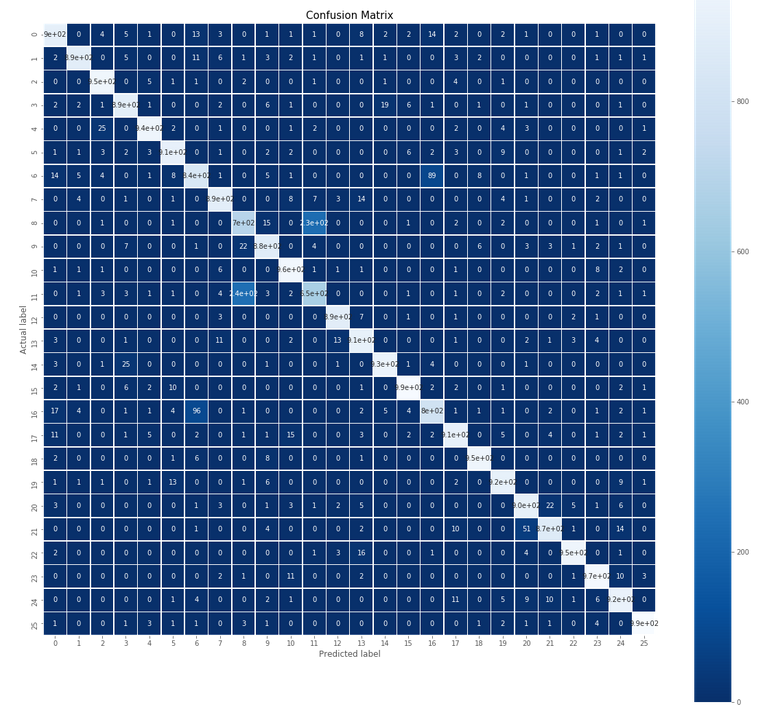


Figure 4 Confusion Matrix - CNN