Machine Learning Engineer Nanodegree

Capstone Project

Jakob Schmollgruber
2018-04-07

1 Definition

Project Overview

In this project we will take a look at the fields of hand written symbol recognition. Handwriting recognition (HWR) belongs to the domain of image processing. Altogether we are looking for algorithms having answers to questions like that:

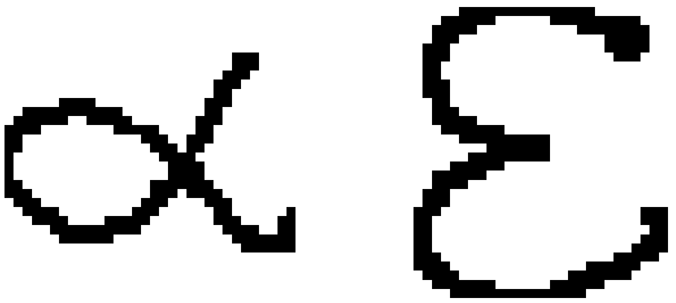


Abbildung 1: Can you name the mathematical symbols?

We used the HASYv2 dataset to train Machine Learning algorithms to recognize 369 mathematical symbols. MNIST, USPS and the EMNIST are probably the most famous topic related datasets (digit & letter recognition). Yann LeCun who released the MNIST Dataset is also known for being the founding father of Convolutional Neural Networks. In 1962 Sheloa Gubermann was one of the first scientist who was active in this field of research. Google scientists including Ian J. Goodfellow published in 2014 a groundbreaking paper about Multi-digit Number Recognition from Street View Imagery. There is a huge economic growth in the sector of Handwriting recognition. This field of research has a high impact on everyday life - In the US more than 95 % of handwritten mail is sorted automatically.

Problem Statement

As a classification problem image recognition is most commonly processed by a supervised learning algorithm. With labeled pictures of mathematical symbols we want to train a Multilayer Neural Network and a Convolutional Neural Network to be able to determine unknown pictures (with symbols). The major steps of work-flow are

- Import symbol data
- Pre- process the data
- Split data
- Train Machine Learning algorithms
- Evaluate
- Test
- Generate Visualization

Metrics

We will use the accuracy as evaluation metric. The accuracy is defined as follows

$$\text{accuracy}(y, \tilde{y}) = \frac{1}{n} \sum_{n=0}^{n-1} 1_{\tilde{y}=y}$$

for y being the true label and $\tilde{y}$ the predicted value.

In layman's terms: The accuracy determines the ratio of correct classified pictures. It's like a multiple-choice test in school: The teacher counts the amount of correct answers and divides this number through the amount of given questions. As symbol recognition is a classification problem it's reasonable to state the ratio of correct classified pictures. It says how reliable a prediction is. Whom do you trust more: A doctor being wrong half the time or a doctor being wrong just in one out of 20 times? Exactly this is measured by the accuracy.

Due to the fact that data is unbalanced some classes of symbols are under-represented - there's a 51 picture class - we will consider the recall score and false negative rate of all class too (Confusion Matrix) to see if some special classes are misclassified all the time. Which are defined as follows

$$\begin{aligned}\text{recall\_score} &= \frac{\text{tp}}{\text{tp} + \text{fn}} \\ \text{false\_negative\_rate} &= \frac{\text{fp}}{\text{tp} + \text{fn}}\end{aligned}$$

where tp means true positive, fp false positive and fn false negative.

2 Analysis

Data Exploration

All 168233 data pictures are saved as png in a folder. The pictures have a 32 $\times$ 32 $=$ 1024 resolution in $[0, 255]$. The are csv files looking like:

```
   path                         symbol_id    latex    user_id
0  hasy-data/v2-00000.png       31             A      50
1  hasy-data/v2-00001.png       31             A      10
2  hasy-data/v2-00002.png       31             A      43
3  hasy-data/v2-00003.png       31             A      43
4  hasy-data/v2-00004.png       31             A      4435
```

Each row represents a picture with a mathematical symbol. The path feature leads us to the storage location. The latex feature names the corresponding latex code. The csv file is pretty much a hashtable. The dataset is by default designed for two possible validation approaches. There is one csv file containing all the data which can be processed according to one's wishes. In addition there are 10 csv fields building up stratified 10-fold cross-validation training sets.

We have to consider a very important fact. Achieving a high accuracy similar to the MNIST database seems not to be realistic. One the hand we have a huge amount of 369 symbols with a high variance in available data per class. One the other hand lets take a closer look at the pictures.
Different labeled symbols sometimes look very similar. The data might need to be tidied up. Being asked - how would you classify the right picture?

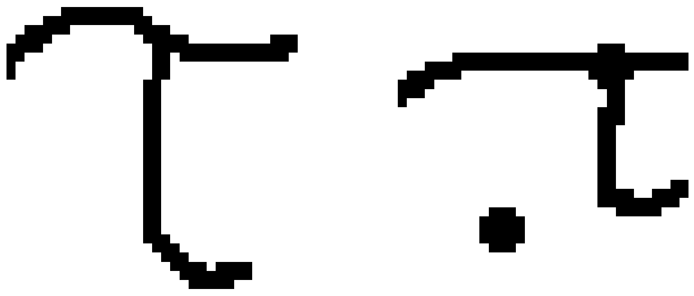


Abbildung 2: The left symbol is a τ, the right symbol is labeled as π

Exploratory Visualization

A picture is worth a thousand words. Let's take a closer look.

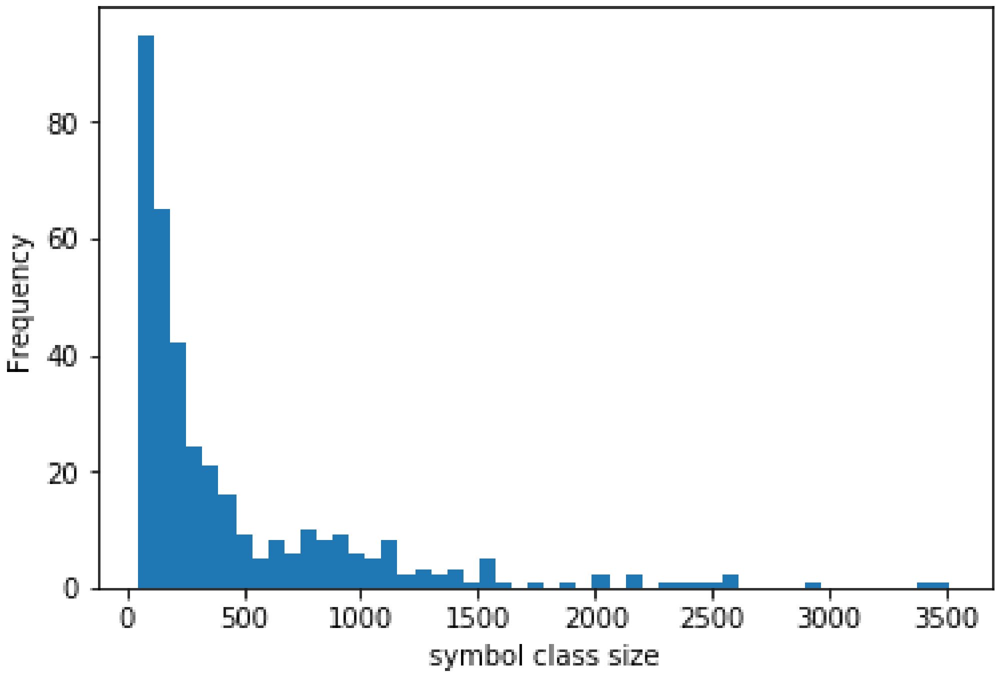


Abbildung 3: Distribution of class size

In the picture above we can see the distribution of the class size. The major part of the hand written symbols is underrepresented. This point is related to the evaluation and validation task and we will discuss it later. Due to the fact that we are processing picture data we have to notice that we cannot visualize further feature properties.

Algorithms and Techniques

Let's take a closer look at our Supervised Learning approach. We used two different Deep Neural Network Architectures

- Convolutional Neural Network
- Fully connected feed-forward Neural Network

Let's first consider the Convolution Neural Network (CNN) since it's the classic image processing architecture. CNN consists of

- Convolutional Layers (filter Kernel)
- Pooling Layers

A Convolutional Layer is a set of filters applied on the input. To understand what filters are supposed to do I suggest to read this article - and to understand how they do it watch this little gif.

Pooling layer are reducing the picture's size by compressing the pictures information. This property is particularly used in Autoencoders.

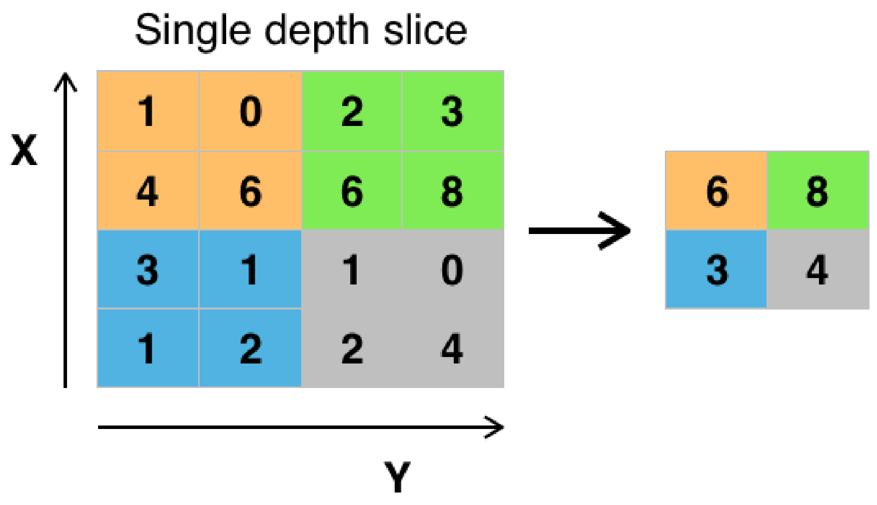


Abbildung 4: Here we can see a max pooling layer reducing the input size of data taking the maximum value of all quarters

The backpropagation algorithm is used to calculate the gradient that is needed to update the Network's weights - thats the way how Multilayer Neural Networks learn. The big advantage of CNN's is the low amount of weights that have to be tuned during the training process.

We choose the CNN because it's the commonly used architecture in this field of research- we choose the fully connected feed-forward Neural Network for historical reasons, since their use generated many breakthroughs.

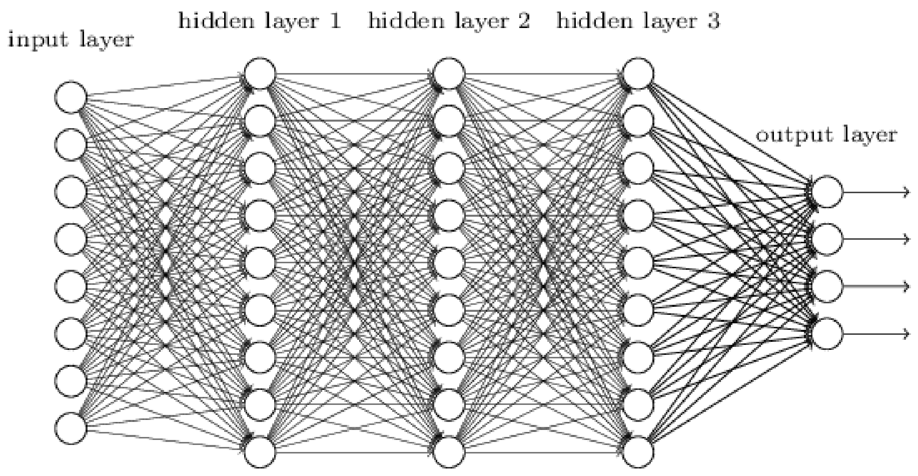


Abbildung 5: Fully connected feed-forward Neural Network

Benchmark

The CIFAR-100 dataset seems to be a suitable benchmark dataset. This dataset has 100 different classes of objects and animals - so there are no symbols. I think this dataset is nevertheless suitable for our purposes. Let's start this section by a little summary of the HASYv2 & CIFAR-100 dataset:

Tabelle 1: Summary Datasets

	CIFAR-100	HASYv2
number pictures / classes	500	456
resolution	32×32	32×32
type	classification	classification

As we can see the ratio of the number of pictures and amount of classes is similar- we think this is a very important point. In addition the pictures have the same resolution and both are used for classification purposes. On the CIFAR-100 dataset an accuracy of 75.72 % has been achieved. A Convolutional Neural Network with Exponential Linear Units (ELUs) obtained the best result on this dataset. To be honest one can find some difference between this datasets - one is balanced the other unbalanced. But the HASYv2 dataset is unique in most properties - so this is also a matter of availability: there is no better choice. Even though it's not a benchmark it's necessary to mention here that the best topic related result (99.7 % accuracy) has been achieved on the MNIST dataset. We think this value can be seen as an upper bound of do-ability in this field. In the justification section we will compare the accuracy of a CNN with the fully connected multilayer Neural Network one can see below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 369)               378225
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              378880
_________________________________________________________________
dense_4 (Dense)              (None, 369)               378225
=================================================================
Total params: 1,135,330
Trainable params: 1,135,330
Non-trainable params: 0
```

3 Methodology

Data Preprocessing

As already mentioned the dataset consist of a folder with images and a hash table containing all the picture's paths and the corresponding labels. In this section we will provide an overview how we transformed this dataset in a format suitable for Keras. Keras needs to be trained with a numpy array of shape

$$\text{(number\_of\_training\_data, rows , columns , depth).}$$

In our case we have to reshape the data such that a vector

$$\text{(i, 32,32,1)}$$

represents a single picture $\forall\ i \in [0, \text{number\_of\_training\_data} - 1]$. At first we import the csv file into a pandas DataFrame. Then we use the String value of the path feature to locate the images. The keras.preprocessing function *image* expects a rgb picture as standard and saves it in numpy arrays. Due to the fact that we are processing black & white symbols we just need the first depth. This 32×32 array has values in $[0, 255]$. We divide by 255 to obtain values in $[0, 1]$- this helps to prevent overfitting (similar to regularization). We suggest to use increased training batch size - This could relax the bad influence of hard to be classified pictures (Data Exploration).

Implementation

In this section we will discuss the implementation process. I think we proceeded kind of straight forward. We used

- Python 3.5
- libraries: pandas, pandas_ml, keras, numpy, sklearn, seaborn, matplotlib, PIL
- Dataset: https://zenodo.org/record/259444

Most commonly classification problems are solved by using supervised learning algorithms. In this particular case we decided to use a CNN. In the refinement section we will give a clear outline about the training process and the model architecture. To mention work-flow in particular

- We first imported and transformed picture data - then split it up (using random states).
- We used the preprocessed data with a batch size of 20 and 15 training epochs to train the Neural Networks.
- Afterwards we evaluated & visualized results.

The most important library was keras. Keras is a neural network API written in python. Keras is user friendly and designed for humans. Since we are inexperienced in the use of python we had some little problems importing the picture data but this is actually not worth mentioning it. Actually I had some real troubles with the factor of time. The k- fold evaluation approach is probably the most reliable one but also costly in terms of training time. Adjusting parameters for various CNN architectures to acquire improved solutions took ages. As a matter of fact the factor of time caused me implement both, the

- k-fold evaluation approach for quality purposes
- train- test split approach to be able to make some quick modifications, visualizations

Refinement

Finding the right Neural Network architecture is very difficult but important. There are so many parameter which have to be set. One could suggest to run a Grid Search to find the best values but unfortunately there are too many. Due to this fact it's very important to do some topic related research. It's reasonable to assume that similar problems can be solved by related methods.

Here we can see our first Convolution Neural Network approach. We used just one convolutional, one maximum pooling and one global-average pooling layer.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 16, 16, 16)        0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 369)               6273
=================================================================
Total params: 6,433
Trainable params: 6,433
Non-trainable params: 0
_________________________________________________________________
```

As expected the acquired accuracy was not satisfying (16 %) .

It's common to add max_pooling layers after convolutional layers. As output layer it's reasonable to use the simple dense layer. As second last layer- therefore the last hidden layer- it's convenient to use a global average_layer.

We started to add sequences of this alternating layers. The first try had just one series of convolutional and pooling layer. In our second approach we used two series of convolutional and pooling layer. This lead to an increase of accuracy. We continued adding layers to the model end ended up with a series of 4 convolutional and pooling layers. Since max_pooling layers halve the size of pictures four is the maximum number of alternating series.
We tried to increase the kernel_size but this approach did not succeed. We used to play around with filter number of the convolutional layers but we did not recognize any particular pattern which lead to improvement. We achie-

ved the most significant improvement by choosing relu as activation function.

Raising the number convolutional and pooling layers is necessary to be able to generalize our classification problem. The first model was not able to understand the complexity of the problem (underfitting). For validation purposes we used the callback function.

Here we can see our final model. We achieved a very satisfying accuracy increase (now 74 %).

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_2 (Conv2D)            (None, 32, 32, 32)        160       
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 32)        0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 32)        4128      
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 7, 32)          0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 64)          8256      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 3, 3, 64)          0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 2, 2, 128)         32896     
_________________________________________________________________
global_average_pooling2d_2 ( (None, 128)               0         
_________________________________________________________________
dense_5 (Dense)              (None, 369)               47601     
=================================================================
Total params: 93,041
Trainable params: 93,041
Non-trainable params: 0
_________________________________________________________________
```

We used a batch size of 20 and 15 training epochs. For all hidden layers the relu activation function was used and for the output layer the softmax. We decided to use the rmsprop optimizer and the categorical_crossentropy as loss function. The choice of the number of training epochs was a matter of time.

4 Results

Model Evaluation and Validation

The commonly used evaluation and validation method is splitting the dataset into training, validation and test data. To obtain a stable model we decided to choose the 10- fold validation training approach, since this method is handled as the gold standard of validation. We split the data in 10 different sets. Each of the 10 sets is used once as test set for the 9 remaining sets - split into training (80 %) and validation data (20 %). We receive 10 different score values. Commonly the mean is used. The advantage of this more complex approach is that validation data is re-used. In layman terms one can say we validate our learner without loosing any training data. This model is very trustworthy - but remember, this means we can believe in the correctness of accuracy received by the Machine Learning algorithm on unseen data. This does not necessarily mean the accuracy is good. Using this method we can expect to earn a very robust and reasonable model. Small perturbations in training data will not carry a penalty in reliability. **We did non break the law: Never train or validate on test data !**
Using the 10-fold validation approach we just achieved a accuracy of 70 % but I think this is just a matter of the little number of training epochs used (factor of training time) . To be honest we learned in this section that the use of the k-fold validation approach for private purposes might not be the wisest decision - it's necessary to make use of some high-end hardware we don't own. With the commonly used split into train, test and validate data and more training epochs we obtained a accuracy of 74 % - in both case the same Network architecture - described in the section above- is used. As already mentioned for all hidden layers we used the relu- activation function and the softmax function for the dense output layer. We used the rmsprop optimizer and the categorical_crossentropy as loss function. I think the most significant parameters to be tuned are the depth of the Convolutional Neural Network and the choice of the relu function.

Justification

In this section we will discuss the different results we received. It's not a big surprise that the CNN provided a better accuracy than the Fully connected Multilayer NN approach. Nevertheless the performance was far better (6 %) than guessing. Our first CNN achieved an accuracy of 16 %. Our final predictor achieved an accuracy of 74 %. This solution is amazing in consideration of the circumstances. Let's consider topic related results. As expected we missed the result achieved on the MNIST database by far. I think the reasons are obvious. In the MNIST database there exist more that 6000 pictures per class. In our case there are many symbols with less than 200 pictures. In addition our Supervised Learning algorithm has to handle 369 classes instead of 10. As I mentioned already the HASYv2 dataset needs to be tidied up. But when we consider the CIFAR-100 dataset our result is very satisfying.

Tabelle 2: As we can see the accuracy received on the HASYv2 dataset is similar to CIFAR-100.

Dataset	Number Data	min. Class size	Number Classes	Accuracy
MNIST	70000	7000	10	99.7 %
CIFAR-100	50000	500	100	75.72 %
HASYv2	168233	51	369	74 %

5 Conclusion

Free-Form Visualization

The next visualization is called confusionplot. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class. In the perfect model - accuracy 1 - we should see a continuous deep blue diagonal line. We can see some gaps on the diagonal and some points off the diagonal. Like commonly color is used to be the measurement of something. How to correctly interpret the obtained results ? Each point lying on the diagonal represents the true positive rate (sensitivity, recall) of those class- any other point in the plot represents the false negative rate.

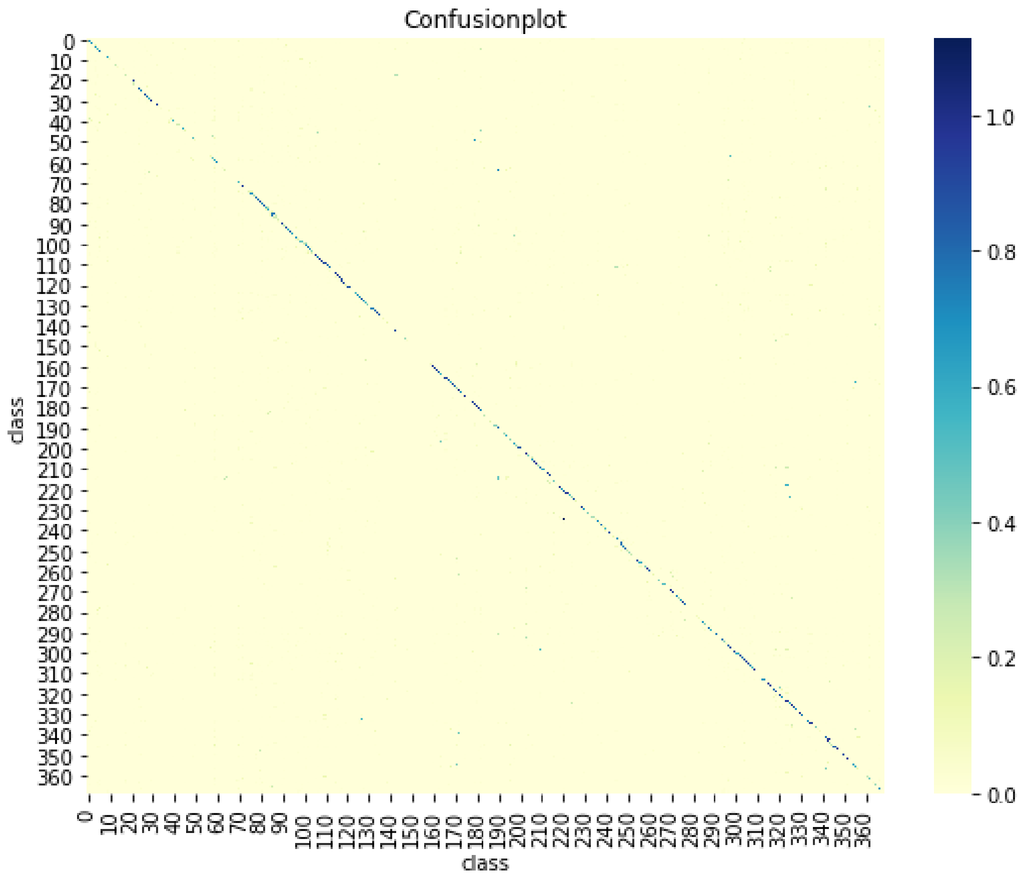
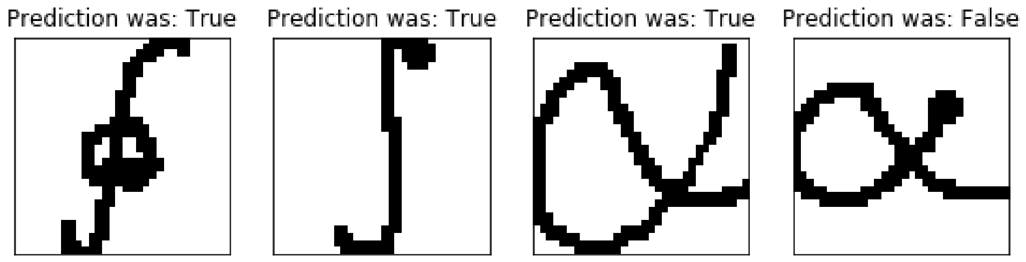


Abbildung 6: We can use the confusion plot to read of recall score and false negative rate

Here we can see some prediction outcome. After some random draws from the testing data we obtained some very interesting illustration.



As we can see the first three symbols have been classified correct. The last is incorrect. The first symbol is the closed path integral, the second the simple integral and the third & fourth alpha. Isn't it interesting that on the one hand the first two some kind of similar looking pictures -representing different symbols- are correctly classified. But on the other hand the two last pictures representing the same symbol are misclassified. This reminds me why Neural Networks are called black boxes.

Reflection

The first time I heard about Machine Learning was when I read an article about digit recognition. It was a reason for me to enroll this course. So I was looking forward to this project. It was even more fascinating than I had thought.
There are many different reasons for amplifying the scope of science. In some cases scientist conduct research on a 'major' problem - research as a matter of interest - and need to solve little problems appearing along the way - research as a matter of needs. The big difference is that in the first case you are the one who is looking for a problem to solve. So I had to find a specific problem first. As mentioned above I am interested in the field of symbol recognition a classification problem. I wanted to train a Machine Learning algorithm to be able to determine the symbol on a picture. Core points of my work-flow have been

- looking for a suitable dataset (HASYv2) with labeled pictures - supervised Learning approach
- importing picture data & getting an overview of the situation
- I decided to use Convolutional Neural Networks as supervised learners

- pre-processing pictures appropriate for keras & splitting test and training data
- training Neural Networks & validating
- generating pictures

After all our trained CNN achieved a accuracy of 74 % on test data. I think the most difficult part (technical) was to transform the pictures to a trainable format- for the simple reason that I'm a python newbie. I think the most interesting part was to find the right architecture of the Convolutional Neural Network. I have to admit that I had the most struggles writing this Capstone Report since it was challenging for me to write in English. I think the Neural Network architecture I used is some kind of commonly used one - so in fact it seems to be a general setting to solve problems like this [2].

Improvement

To raise the dropout probability could improve accuracy. But the high amount of additional training time makes this impossible for me (this is a matter of days since we are using 10- fold cross validation). I think there is always a better solution for almost every problem- it has just to be found. In that special case someone already came up with a better solution [1]. To reach some major improvements we probably need more training data on the one hand and a data clean up on the other. To be honest I read something about the use of Recurrent Neural Networks in Handwriting recognition - but sadly I didn't know how to use the architecture [3].

Literatur

[1] Thoma Martin, The HASYv2 dataset, arXiv:1701.08380v1, 2017

[2] Murugan Pushparaja, Implementation of Deep Convolutional Neural Network in Multi-class Categorical Image Classification, arXiv:1801.01397v1, 2018

[3] Wang Jiang, CNN-RNN: A Unified Framework for Multi-label Image Classification, arXiv:1604.04573v1, 2016