DMKM - UPO

Image Data Mining

Optical Character Recognition using MNIST dataset and SVM

Carlos López Roa
me@mr3m.me
github.com/mr3m/OpenCVMNISTSVM

January 28, 2017

1 Introduction

The task of Optical Character Recognition (OCR) consists in identifying text in images or video automatically, it has gained importance in the field of Computer Vision for applications such as Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt, Automatic number plate recognition, Extracting business card information into a contact list, Make electronic images of printed documents searchable, e.g. Google Books, Converting handwriting in real time to control a computer (pen computing), etc. It is a common method of digitising printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, text-to-speech, key data and text mining. [1]

The MNIST Dataset [2] was introduced in 2013 as a optimised version of the NIST dataset for training machine learning models in pattern recognition [3]. It consists of 60,000 training images and 10,000 testing images. Each image is 28x28 image containing a hand-written digit form 0 to 9. Both training and testing set are labeled so one can train an test the ML model. An overview of the dataset can be seen in figure 1

Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks. An SVM model represents each example as a point in n-dimensional space, and then computes the decision boundary using, either linear or non linear functions. SVMs can be seen as efficient classifiers, since they can compute sparse representation of the dataset based only in the so-called *support vectors*, which are a small subset of the dataset.

In this work, the aim was to use Machine Learning algorithms to correctly classify video input of digits into the corresponding classes from 0 to 9 using supervised learning with the OpenCV Library [4]. Code it's available in the repository github.com/mr3m/OpenCVMNISTSVM

Figure 1: Sample of the MNIST dataset. Each hand-written digit it's equiped with a label for supervised learning.

2 Acronyms and Abbreviations

OCR: Optical Character Recognition
SVM: Support Vector Machines
CV: Computer Vision
ML: Machine Learning

3 Theoretical assumptions

- One assumption about the model is that we don't need (want) to hand-craft features for the dataset, that is, we want to design a model that can correctly classify the dataset by only reading the raw pixels.
- During training, we're mapping our dataset from an array of $20x20$ to a vector $400x1$ with each value ranging in $[0, 255]$. And then, since we're using SVMs, we want to find w, b such that for each class C_i

$$\mathbf{I}_{x \in C_i} = \text{sign}(w \cdot x + b), \tag{1}$$

  that is, x belongs to class C_i iff $\sum_i^n w_i x_i + b > 0$. In this case we have 10 classes, so, we need to find 10 sets of weights w_{C_i} and biases b_{C_i}. [5]
- By the previous point, we are assuming that each class is linearly separable from the others on the domain of it's raw pixels.
- In this case we're assuming that the dataset elements are vectors in $[0, 255]^{400}$, that is vectors in 400-dimensional space.
- Since the dataset resulted too massive to compute in short time, using a small computer, we are only using 5,000 images for training and 2,500 for testing. Hence, we're assuming we can find the corresponding classifier with a small subset of the training set.

4 Algorithm description

A general overview of the algorithm can be seen in figure 2

One should set correctly three boolean variables inside the code according to one's needs.

- Hence, If Testing is set to True, the process will begin by loading the dataset from disk, this is done by taking slices of an image containing an ordered array of 5,000 images. The labels are assumed from the position of the digit in the array. An overview of the dataset can be see in figure 3
- Then, If Training is set to True, the SVM will be instantiated, the structure and parameters given and the training process will begin, this is a very costly process taking around $2 * 10^3$ seconds. Thereby the desire to cut this time by saving a pre-trained model into disk, this is done after training is finished.
- If the Training is not set to True, then the pre-loaded model will be loaded into memory.
- The model will be tested against the testing set, giving an accuracy score.
- If Testing is not True, the process will load the pre-trained model to memory.
- If Live is set to true, it will begin the acquisition of images.
- After the images is acquired it's converted into grayscale, equalised using the histogram, filtered using Gaussian filter. Then threshold is applied to binarize the image and color inversion. The image it's resized and reshaped to fit the format of the training set. A panorama of the transformations can be seen in figure 4

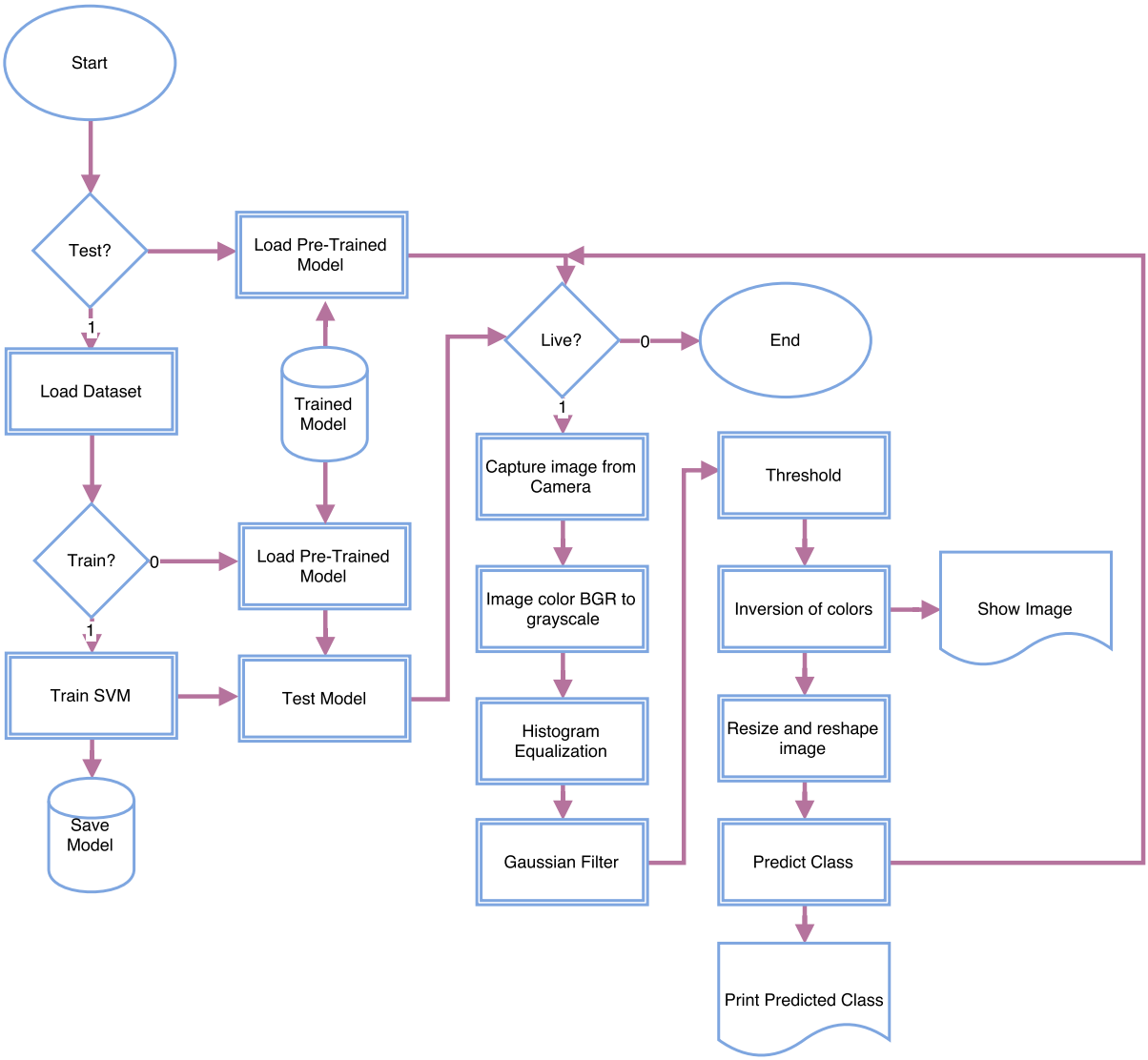

Figure 2: General flow diagram of the implemented algorithm. It contain three boolean variables that control the flow inside the algorithm; Test, Train and Live. The first controls wether the dataset should be loaded and tested, the second wether the model should be trained with the loaded dataset, and the third wether images should be captured from the camera and classified with the loaded model. If no training is selected a pretrained model will be loaded from disk.

- Then the preprocessed image it's used to predict a class using the model and the prediction it's printed in console.
- The program loops acquiring images until it's interrupted.

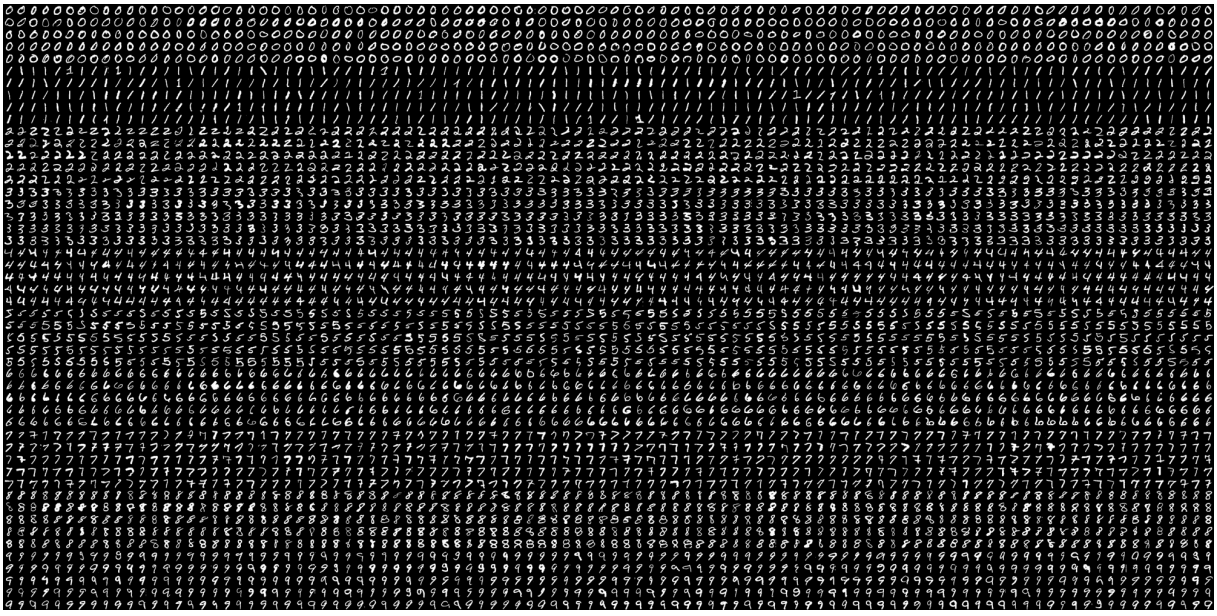

Figure 3: Dataset used for training and testing.It contains 5,000 digits evenly distributed amongst the classes.

5 Tests description

It's worth mentioning that the original scope of the project was to use Deep Learning models to further refine the precision of the predictions, however, difficulties with the libraries and languages restricted the development to C++ only. Efforts to cross compile, Tensorflow[1] where not successful, furthermore the implementation of Artificial Neural Networks in OpenCV, it's not mature enough to even consider non-sigmoid activation functions, dooming the development. Also, it was not possible to read the entire dataset and train on it in a reasonable time.

So, only one conclusive test was carried out, regarding the architecture of the SVM. A parameter exploration over the parameters of the SVM could be done, however, the tests took around $8 * 10^3$ seconds making very difficult to obtain conclusive results.

Three architectures were tested for the SVM. One was linear Kernel, and two Polynomial Kernels with 3 and 5 degrees of freedom.

Live testing of the digit recognition provided subjective satisfactory results, that is, the predicted class is almost always the true class of the presented number, but no objective measure could be drawn.

[1]The Google's Deep Learning Library, implemented in C++ but interfaced in Python.

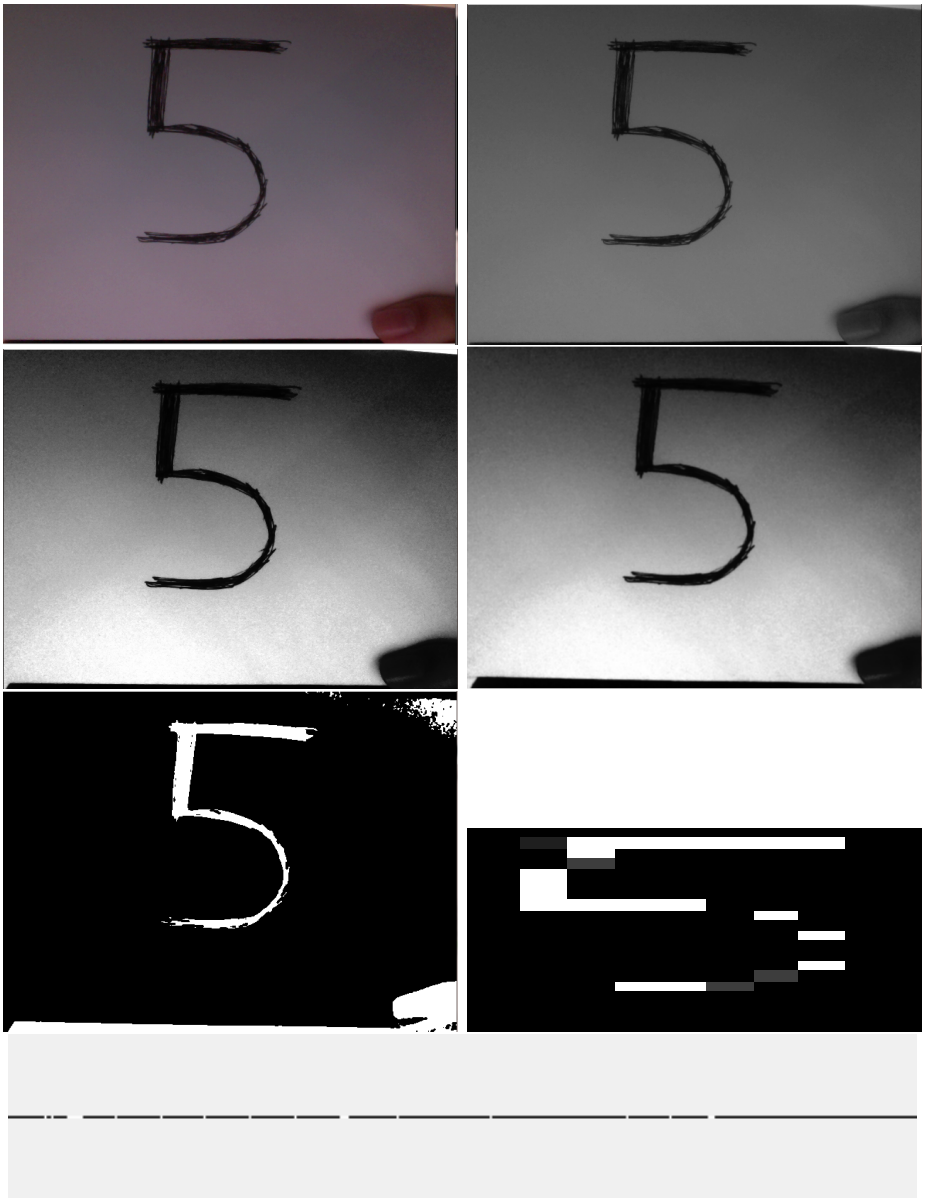

Figure 4: Preprocessing steps of the captured image for prediction. From top to bottom and left to right. a) Colored image, b) Grayscale, c) Histogram equalization, d) Gaussian Filter, e) Threshold and Inversion of color, f) Resize, g) Reshape.

6 Obtained Results

We can see the results of the first test on the kernel of the SVM in figure 5, we can see that the linear kernel has good precision (88.8%) confirming our assumption that each class is linearly separable from the others.

Also we can see an example of text recognition using live video in figure 6, each 40 seconds the system performs a prediction and print it into the console. In this case all predictions are correct.

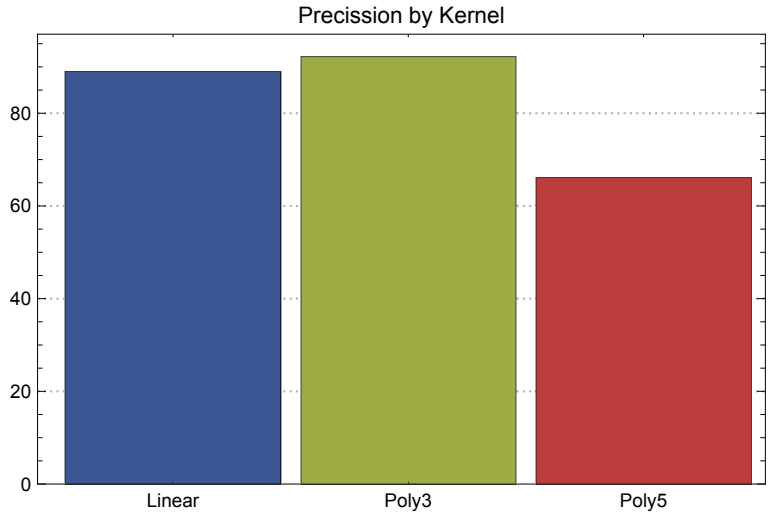

Figure 5: Precission by Kernel. The linear kernel performed second best with 88.8% of precission in the test set, while the Polynomial Kernel of third degree performed best with 92.2% of precission, however the Polynomial of order 5 decreased to 65.92% of precission

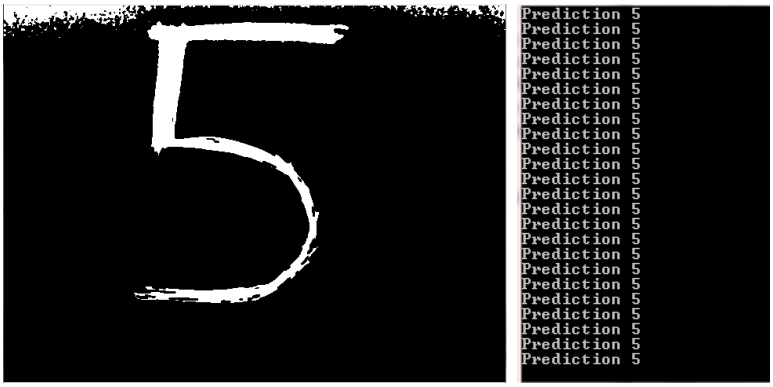

Figure 6: The prediction it's consistently correct when the presented picture is still, big, centered and without noise.

7 Conclusions and future work

We can say the following conclusions

- We were able to implement a classifier without crafting extra features other than the raw pixels.
- We were able to model the classifier as a Linear SVM, that is, making the dataset linearly separable class by class.
- The full dataset resulted too heavy for computing in a laptop, hence we were able to train the model with a only 8% of the full dataset.
- Approaches for using Deep Learning Techniques failed due to the modelling language and the lack of programming skills in C++.

- The implementation of Artificial Neural Networks of OpenCV was not mature enough to model this dataset.

And regarding the future work:

- In the future, I would rather like to train in the full dataset
- Using convolutional neural networks, which have proven 99.77% accuracy in the literature
- This can be achieved using the Python implementation of OpenCV together with the Python interface of Tensorflow.
- The segmentation of the image it's not perfect and some noise get's into the classifier input leading to miss-prediction.
- Even tough the polynomial kernel of order three has more precision this can be recognised as overfitting since the testing set is part of the training set.
- A comparison of the benchmark can be seen in figure 7

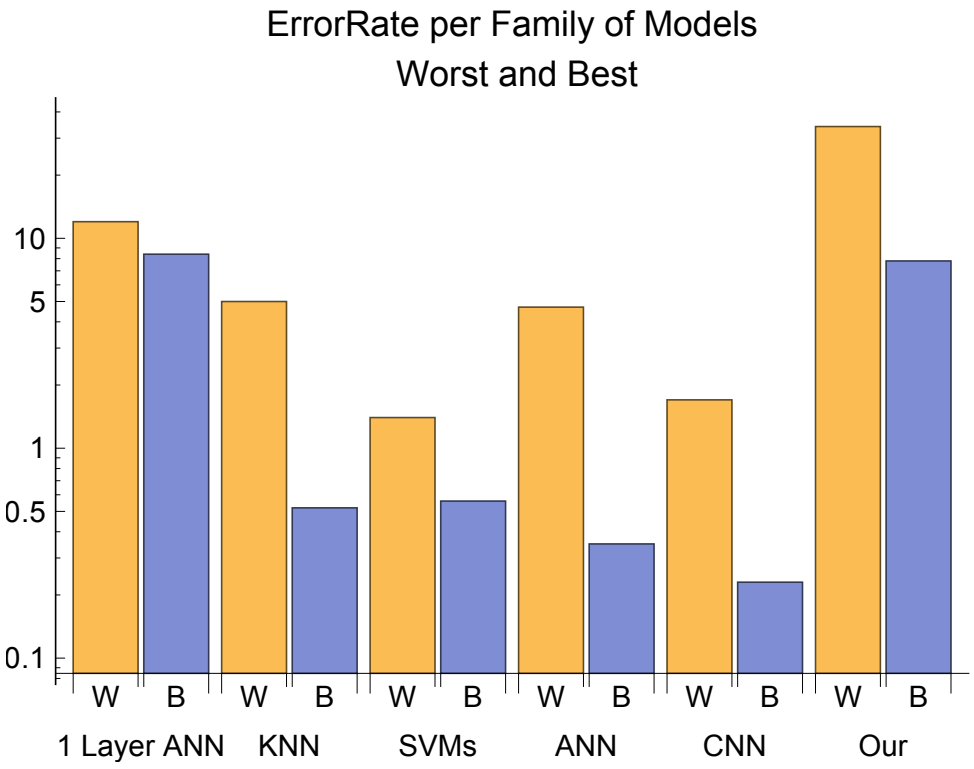

Figure 7: ErrorRate Benchmark for different families of models, Worst and Best. Logarithmic scale. We can see that Artificial Neural Networks and Convolutional Neural Networks can achive errors of about 10^{-1} whereas our model performs at best with an error rate of 10^2, big improvements can be done regarding the accuracy of the model. Our model it's at the level of linear models, which is what we expect.

References

[1] *Optical character recognition* , Wikipedia, 28-01-17

[2] *The MNIST database of handwritten digits*, Yann Lecun, Corinna Cortes

[3] *Timeline of optical character recognition*, Wikipedia, 28-01-17

[4] The OpenCV Reference Manual, Itseez, 2014, `http://opencv.org/`

[5] Burges. A tutorial on support vector machines for pattern recognition, Knowledge Discovery and Data Mining 2(2), 1998