André Filipe Pinto Caetano Leitão - 519008 – andrefp
Francisco dos Reis Centeio dos Santos Branco - 518941 - fdcentei

Machine Learning - Assignment 5

Introduction

For this assignment, we used the programming language of python. Multiple libraries such as Pytorch and sci-kit learning. The Feature Engineering part also relies on skimage and PIL to manipulate images. Full list of libraries:

- os
- Numpy
- PIL
- Skimage
- Sklearn
- More_itertools
- Matplotlib
- Torch
- Torchvision
- Tqdm
- Collections

The Feature Detector part runs the developed code altogether, while the Feature Classification also runs the Feature Engineering part, and finally this last runs only itself. The char dataset needs to be in the following directory: /dataset/char74k-lite, and the detection images in the /dataset/detection-images. The slashes from the paths in the code should be changed in Linux or Mac O.S.'s.

Feature Engineering

To load the data from the directories of each letter, we selected two different ways of pre-processing the data. In particular, we had selected Histogram of Oriented Gradients (HOG) and Scaling as features engineering.

We are interested in modelling features that allow us to discriminate between classes according to the dissimilarities. Therefore, one possible approach is the Feature Descriptor, an algorithm that takes an image and outputs locations, *i.e.*, pixel coordinates of significant areas in the image, corresponding to feature vectors. As an example, we applied HOG, from the scikit-image library, which counts occurrences of gradient orientation in localized portions of an image by applying gradient image and histograms, normalizing across blocks and flattening into a feature vector. This technique tends to keep the good features, while the undesired ones are omitted.

At that point, we are able to apply a different technique, Feature Scaling. It normalizes the range of features of data, that is, it eliminates the sparsity by bringing all the values onto one scale, so a wide range may not give an unfair advantage to some feature. Hence, the

features reside in the 0 and 1 range, have zero mean and unit variance. As a result, in our Support Vector Machine method, the time to find support vectors is reduced.

As suggested, other approaches could be considered, but for our purpose, these are likely to be suitable for our specific dataset, maintaining a fit threshold and a trade-off between efficiency and accuracy.

Character Classification

d) After a first glimpse of the dataset, we can see that it consists on various images of characters. Intuitively, it is normal to assume that it is not a simple task to train a computational model to identify these letters. Therefore, a fine-tuned deep learning model (with convolutional neural networks as preferred) would be the best to perform this task, since the great number of hidden layers gives the necessary complexity to the model to correctly extract features from each image.

e) The two models our group used were simple Artificial Neural Networks with 400 inputs (20*20 image), 32 Neurons on the Hidden Layer and 26 outputs (as final classes) and an SVM polynomial model of a degree of 5. Each were fine-tuned as best as possible, in terms of learning rates, epochs, etc. These models were selected due to the complexity of mounting a good deep learning algorithm. This way, the SVM and ANN were easier to manipulate.

f) We decided to evaluate the models in terms of cross-correlation entropy loss and accuracy (number of correct/number of total). An 80 % accuracy, give or take, was reached on the SVM model, being this model the one used on the detection part. For the ANN, the maximum accuracy reached was between 60 % and 70%, which is not as satisfactory. Therefore, the loss in the ANN model was higher than the one seen in the SVM.

	ANN	SVM
Cross Entropy Loss	1.1796887930896547	0.7065420407221508
Accuracy	0.7040960451977402	0.8029661016949152

g) As it was said previously, the preferred model for this problem was the deep learning type of convolutional neural network (CNN), as it is the model of choice for feature detection in images. The kernels are powerful tools when it comes to extract features from images and classify them in the end. However, because of short time to test and explore these kinds of models, our group could not manage to get a hold of these models. A faster approach with simple ANN of one hidden layer and SVM was the our choice.

Character Detection

In order to load the data from both images of the detection directory, we start by pre-processing the data with Histogram of Oriented Gradients, the same way we did in Feature Engineering, and, therefore, feature vectors are selected by effectiveness.

The next step is scanning the image, done by selecting a square of size 20x20 and moving it with a step of 1 on each axis. From each sample, the detector determines the ratio of contrast between the pixels, and if it is greater than a lower bound, the detector considers it as a hit. The result is a list of boxes.

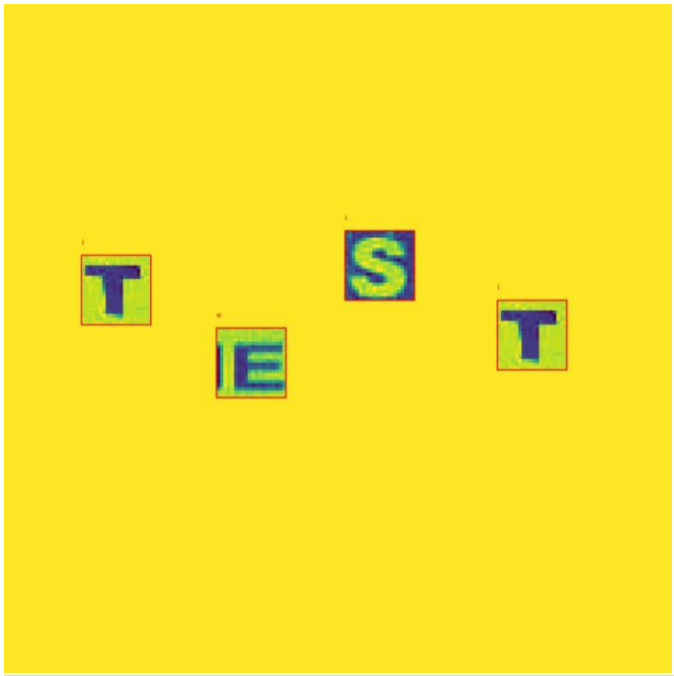
Rather than detecting a letter just once, it might detect it multiple times. Hence, we apply Non-Maximum Suppression (NMS), so it can eliminate some candidates that are in fact different detections of the same letter, without removing the candidates for different letters. In our implementation, the input is a list of proposal boxes B, corresponding confidence scores S and overlap threshold N, and the output is a list of filtered proposals D, empty at the begin. The algorithm computes as follows:

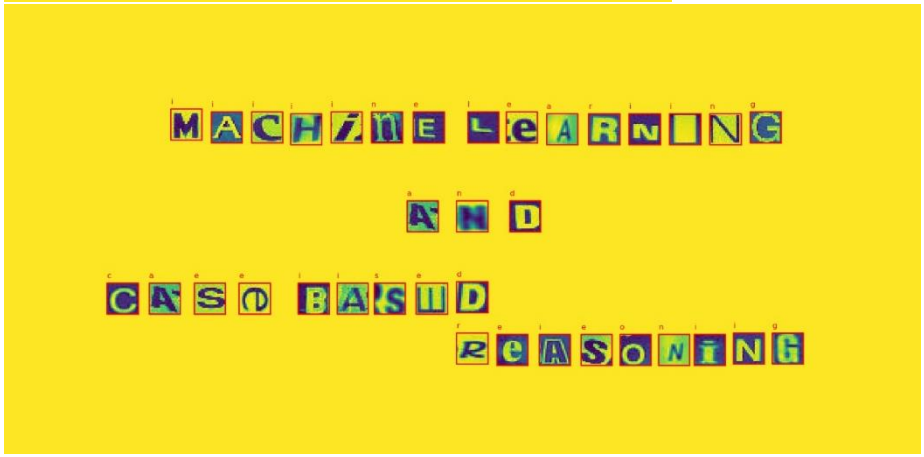
1. Select the proposal with highest confidence score, remove it from B and add it to the final proposal list D;
2. Compare this proposal with all the proposals: calculate the Intersection Over Union (IOU), implemented in another method, of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B;
3. This process of taking the proposal with the highest confidence from the remaining proposals, calculating its IOU and eliminating the boxes with highest IOU than threshold is repeated until there are no more proposals left in B.

Once done, we can use a previous classifier method, this is, SVM, and thus get a classification, the corresponding letter, of each filtered proposal.

The main problem of this method is the evaluation given to the predictions. Some values are defined by us, such as the detect score threshold, strides and IOU upper bound. Thus, we need to adjust them by experiments and give to them a reasonable value.

We may get better results, or just be more efficient, by selecting feature engineering algorithms that help us detecting and classifying letters. One of them is manually cropping since there are only two samples each one with a wide blank space. Other techniques include feature scaling as applied in Feature Engineering, digital image processing (noise removal) and even another type of feature descriptor, like Scale-Invariant Feature Transform.





Conclusion

The technique Feature Scaling is likely to not return better accuracy from the classifiers. As mentioned, its effectiveness is measured, as an example, by the reduction of the time to find support vectors. Furthermore, since we had applied a Histogram of Oriented Gradients as the first pre-processing technique, the accuracy from the next feature engineering technique has got a slight improvement.

The classifier was solid, for 80% accuracy on SVM, thus being a strong part of the overall OCR system.

The detection system also is very good at detecting the letter, maybe being the best part of our work.

We would appreciate understanding more deeply the techniques and different methods that we could apply to each OCR component. Therefore, the missing time and its inherent problems might have resulted in a worse evaluation from the system as we should expect.

Interconnect each component of the system was not an easy task since we could not lose our line of thought.

Apart from that, we liked to apply our knowledge from all the Machine Learning topics and methods and gain more experience by general-purpose ML libraries, choosing what fits better for a real implementation. Optical Character Recognition seems to be a complex procedure if we pretend to get efficient calculations and, even more, suitable detections. Although, it's interesting how we can apply unsupervised learning and turning it into a useful system, that, otherwise, we would not be able to do it in a feasible time.

References

https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html

https://en.wikipedia.org/wiki/Visual_descriptor

https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients

https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c