Image Classification with Fashion-MNIST

Giorgio Dilda
giorgio.dilda@studenti.unipd.it

Gaurav Anand
gaurav.anand@studenti.unipd.it

1. Introduction

Image classification is a supervised learning problem, models are trained on labelled sample images to classify new unseen images. There is a growing application of image classification techniques in a huge variety of sectors such as health (disease detection through X-ray or ECG), automotive (driver less cars), security (person recognition) and industry (recognition of defective pieces in production lines).

Even if there are many traditional ML models able to reach satisfying performances in this field they usually heavily rely on field related knowledge and data manipulation. On the other hand the highest performances are usually reached by neural networks, in particular the convolutional ones. We chose to implement algorithms from both these areas of machine learning to display a wider choice of model that may be useful in different kinds of applications.

As we will see through the report CNN reaches as expected higher performances with respect to other models and for this reason we decided to tune this model in order to reach even higher levels of accuracy.

2. Dataset

We used the Fashion - MNIST dataset. The Fashion - MNIST is a dataset of Zalando's article images consisting of a training set of 50,000 examples, a validation set of 10,000 examples and a test set of 10,000 examples. Each example is a 28x28 gray scale image, associated with a label from 10 classes. Each pixel has a value from 0 to 255 representing the color of the pixel. Since this dataset is widely used as model benchmarking (it was ment to replace the MNIST dataset which had become too easy) the only preprocessing that was needed was standardizing the values.

Each training and test example are assigned to one of the following labels below.

Analyzing the dataset, we concluded that most of the classic ML models (Logistic Regression, Support Vector Machine, Random Forests, and k-Nearest Neighbors) as well as simple Neural Network achieve an accuracy of 85%-90% and in all of them the main struggle was differentiating between two out of the ten classes, namely the shirt and t-shirt classes. In order to better evaluate the model we tried Convolutional Neural Network using multiple filters which are trained to detect different features.

T-shirt/top	0
Trousers	1
Pullover	2
Dress	3
Coat	4
Sandal	5
Shirt	6
Sneaker	7
Bag	8
Ankle boot	9

Table 1. Classification code

3. Methods

It is known from the literature that the best results in the field of image recognition are reached by neural networks, however other types of traditional machine learning models can provide satisfying results with some additional advantages. For this reason we chose to experiment some of these models to see how they perform and to highlight their pros and cons. Except for the CNN, which has been developed with Keras, the other models have been build using the Scikit-learn library

3.1. Support Vector Machine (SVM)

In machine learning, support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. SVM is a variant of the linear model that exploits a different cost function in order For this reason it offers good performance and interpretability.

3.2. Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. Logistic regression measures the relationship between

the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function, which is the cumulative distribution function of logistic distribution.

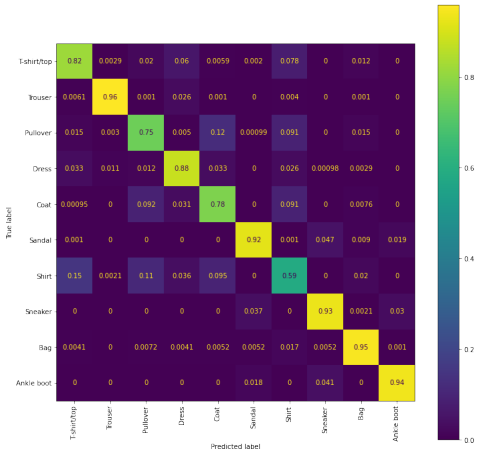

Figure 1. Logistic regression's confusion matrix

3.3. k-Nearest Neighbors (k-NN)

k-NN is a very simple algorithm that reaches surprisingly high performances in image recognition. k-NN is a type of instance-based learning, where the function is only approximated locally and all computation is deferred until function evaluation. In k-NN a useful technique can be to assign weights to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. It has the lowest training runtime (notice that it doesn't actually train anything) but by far the highest prediction time.

3.4. Random Forest Classifier

Random forest is an ensemble model that exploits decision trees. The random forest fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The features are always randomly permuted at each split. Even this method takes a while to be trained but offers good performances and interpretability.

3.5. Simple Neural Network

This is a simple NN model composed by a fully connected hidden layer (500 units) followed by a fully connected output layer. It reaches an accuracy similar to other traditional models but it takes quite a bit of time to be trained and is a black box (i.e. we can't use the trained model to understand something about the data distribution).

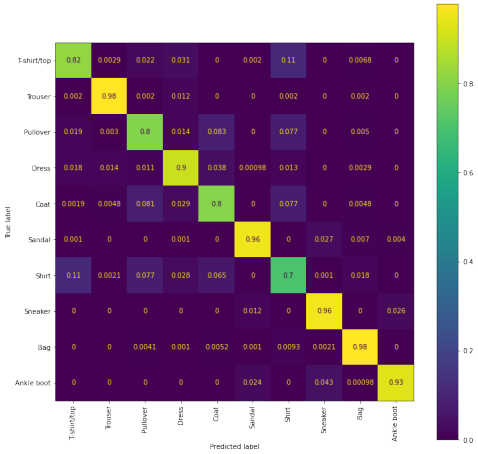

Figure 2. Simple NN's confusion matrix

3.6. Convolutional Neural Network (CNN)

Convolutional Neural Networks are Neural networks that exploit at least one convolutional layer. In convolutional layers only a few input neurons are connected to a next layer neuron, moreover the parameters of the connection are shared all across the layer significantly reducing the amount of learnable parameters. In image recognition this has a straightforward interpretation: the network learns to recognise simple edges in the image that, as the number of number of convolutional layers increases, become characteristic traits of the image. This is the architecture that nowadays reaches the best performances between the ones we have experimented. (In the plots only the tuned version of this model is shown)

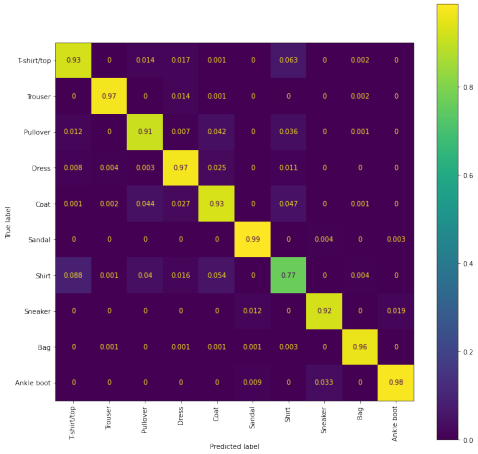

Figure 3. CNN's confusion matrix

4. Experiments

After seeing the results of all the models we chose to spend some time tuning the CNN in order to reach higher

Architecture	Accuracy	Runtime
Logistic Regression	0.85	11.12
SVM	0.86	489.60
k-NN	0.85	9.39
Random Forest	0.88	93.43
Simple NN	0.88	403.04
Convolutional NN	0.93	7943.79

Table 2. Result Analysis

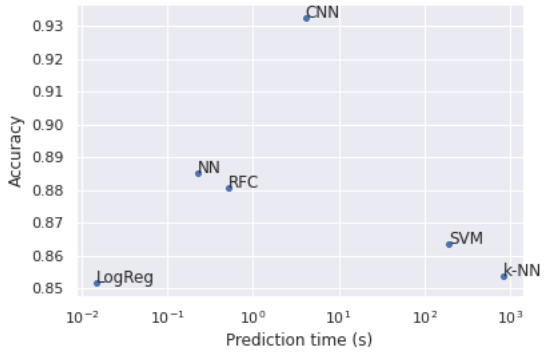

Figure 4. The different models compared by accuracy and prediction time

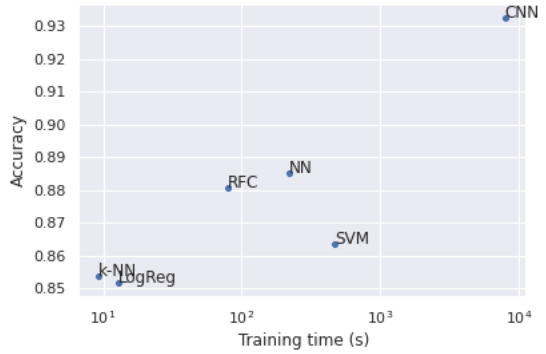

Figure 5. The different models compared by accuracy and training time

levels of accuracy.

First we decided to increase the number of convolutional layers to 3 and to add a MaxPool layer as regularizer. At the same time to increase the variety of the database we applied data augmentation to the training set (image rotation +-5° and horizontal flipping). We chose not to rotate too much the images since in the training set they are all straight.

We tried to apply l1 and l2 regularizers for the dense layers after the convolutional part but we didn't get satisfying results so we decided to switch to dropout for 2 of the dense layers (not the one immediately before the output layer).

A lot of time was spent in tuning the learning rate in order to not waste the potential capacity of our network, we ultimately chose to use an exponential decay with SGD optimizer.

To avoid over fitting we applied early stopping too.

Since the over fit problem was under control we started increasing the capacity of our model increasing the depth of the dense part of the network passing from 0.9 to 0.93 accuracy just adding a couple of layers.

Notice that using a GPU accelerator for the CNN training the time lowered to 1382 s.

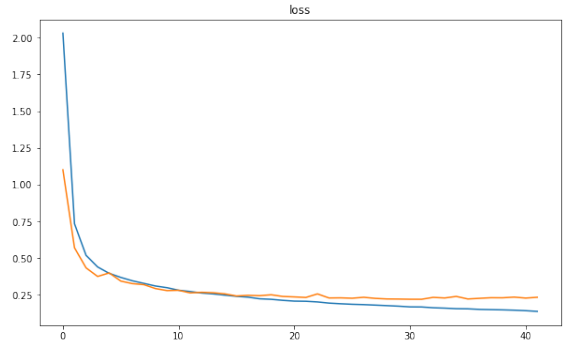

Figure 6. Loss of CNN through epochs

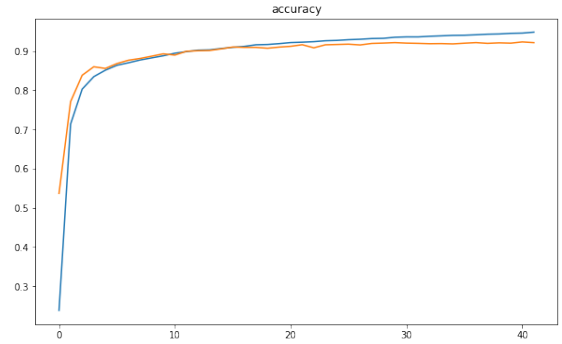

Figure 7. Accuracy of CNN through epochs

References

[1] Zalando. Why we made fashion-mnist.
[2] Wikipedia. Universal approximation theorem.
[3] Scikit-learn. (`scikit-learn.org`)
[4] Keras. (`keras.io`)