Fashion MNIST Object Identification

Aykut Karayel
aykut.karayel@stu.bahcesehir.edu.tr

Kutay Erkan
kutay.erkan@stu.bahcesehir.edu.tr

Tuğba Karakurt
zahidetugba.karakurt@stu.bahcesehir.edu.tr

Abstract — Object recognition is as we know a computer vision technique for identifying objects in images or videos. This is a key output of deep learning and machine learning algorithms. When humans look at a photograph or watch a video, we can spot objects and visual details. The base goal is to teach a computer to do this, to gain a level of understanding of what an image contains.

Object recognition and object detection are similar techniques for identifying objects. Object detection is the process of finding instances of objects in images. In the case of deep learning, object detection is a subset of object recognition, where the object is not only identified but also located in images. This enables for multiple objects to be identified and located within the same image.

Both deep learning and machine learning techniques are used learning how to identify objects in images, but they have some different modifications in their application. Deep learning techniques such as convolutional neural networks (CNN), are used to automatically learn an object's inherent features in order to identify that object. This technique offers high level of accuracy but requires a large amount of data to make accurate predictions.

In Machine learning models we can separate the features into their distinct categories, and then use this information when analyzing and classifying new objects. Using these techniques for object recognition ıt offers the flexibility to choose the best combination of features and classifiers for learning. It can achieve accurate results with minimal data.

In this case our goal is classifying our instances as correctly as possible to their respective classes. We want to apply both neural network (CNN , R-CNN, SVM) and machine learning techniques (Random Forest, K-Means) to overcome our analysis.

Keywords: Convolutional Neural Networks (CNN), Support Vector Machine (SVM), Random Forest , K-Means.

I. INTRODUCTION

In our project, as I mentioned above our aim will be classifying. We have 785 features and 60.000 examples, more basic Machine Learning models such as decision trees need, at the very least, feature selection and possibly more preprocessing to be computationally feasible and applicable. But how could we decide which of our features -pixels are less important?

An artificial neural network gives us the opportunity of using thousands of weights and we update those weights as our model learns more about the dataset, thus making 60.000 examples an advantage, not a burden. While training the model, we will use labeled examples to teach our model predetermined classes. This makes simple feedforward models a good starting point for our object detection problem. After that, we plan to try other ANN models such as CNN, R-CNN or SVM.

Alongside ANN Models, we will apply tree models. Similar to Neural Networks, the tree is built via a learning process using training data. The learning process creates the tree step by step according to the importance of the input features in the context of the specific application. Using all

training data objects, at first the most important feature is identified by comparing all of the features. According to the resulting splitting value the training data is subdivided. For every resulting subset the second most important feature is identified and a new split is created. The process will repeat on each resulting subset until the leaf nodes in all the branches of the tree are found. A Decision Tree is easy to create, is able to handle different types of input data (also categorical) and is interpretable. However, Decision Trees often show a lack of reliability in their application to new data. One reason for this is their tendency to perfectly fit all samples in the training data.

Convolutional Neural Networks are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (SVM/ Softmax) on the last (fully-connected) layer and all the tricks we developed for learning regular Neural Networks still apply.

So what changes? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and reduce the amount of parameters in the network.

II. EXPLORATORY DATA ANALYSIS

Our data is an image dataset of Zalando which is a European e-commerce company based in Berlin. The dataset consists of a training set of 60.000 and a test set of 10.000 examples. Each example is a 28x28 grayscale image, associated with a label with 10 classes.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels for an image. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel. Higher numbers means darker. This pixel-value is an integer between 0 and 255.

The training and test data sets have 785 columns. The first column consists of the clothing class labels, the rest of the columns contain the pixel-values of the associated image. Our training and test examples are assigned to one of the ten labels which are T-shirt/top, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot.

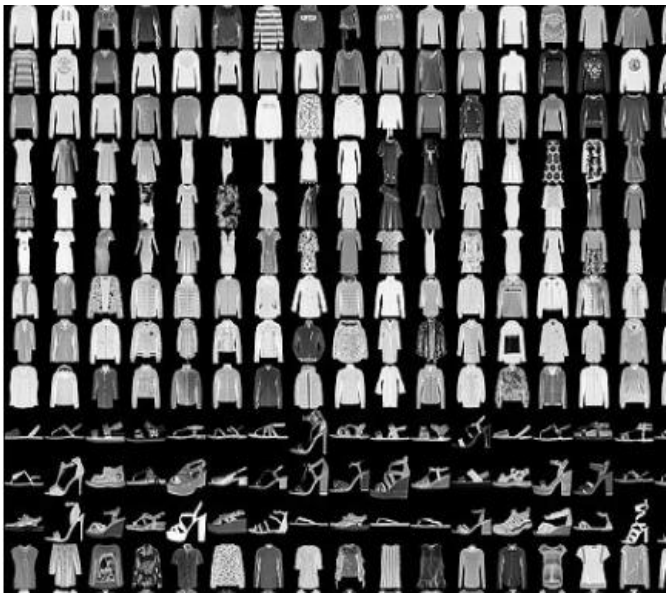


Figure:1 Some visual images of our data

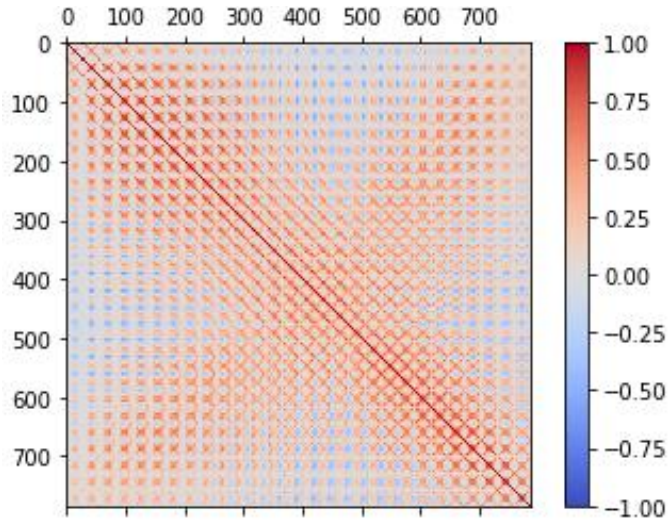


Figure:2 Correlations between location of pixels

Figure:1 shows visualization of our data in a little part. As we see, the pixels which belong to every image vary from pixel to pixel with their location.

Figure:2 shows how to correlate these whole pixels with their location in each image. As we see the edge and the middle pixels of the images most correlated. Because the edges are dark in a word black and the middles are mostly light - white.

III. DATA PREPROCESSING

Firstly, we will normalize our pixel-values which are between 0 and 255 , turning to 0 and 1.

We split the train set in train and validation set. The validation set will be 30% from the original train set, therefore the split will be train/validation of 70% / 30%.

We will apply PCA in some of our models for dimensionality reduction to reduce a large set of variables to a small set that still contains most of the information in the large set. This allows to identify the component of the dataset that maximizes variance, making standard models more powerful.

IV. METHODOLOGY

RANDOM FORESTS

As a baseline, we try Random Forests. It returns an accuracy of 0.875 on the test dataset with 64 estimators.

K-MEANS

As a second model, we try K-Means + SVM. Results do not improve with this combination.

```
...: model= svm.SVC()
...: history = model.fit(X_train, y_train)
...: y_pred = model.predict(X_test)
...: print('KMeans + SVM: {}'.format(accuracy_score(y_test, y_pred)))
KMeans + SVM: 0.59
```

K-means + SVM test acc : 0.59

LOGISTIC REGRESSION

Without PCA , our accuracy with Logistic Regression on the test dataset is 0.79.

```
In [61]: model = LogisticRegression(random_state=42)
    ...: history = model.fit(X_train, y_train)
    ...: y_pred = model.predict(X_test)
    ...: accuracy = accuracy_score(y_test, y_pred)
    ...: print('LogisticRegression: {}'.format(accuracy_score(y_test, y_pred)))
LogisticRegression: 0.7933333333333333
```

In logistic regression technique while applying PCA, we reach %81 accuracy with 24 Components.

```
PCA: PC 2 Logistic Regression Test Accuracy:  0.476
PCA: PC 3 Logistic Regression Test Accuracy:  0.633
PCA: PC 4 Logistic Regression Test Accuracy:  0.65
PCA: PC 5 Logistic Regression Test Accuracy:  0.7
PCA: PC 6 Logistic Regression Test Accuracy:  0.74
PCA: PC 7 Logistic Regression Test Accuracy:  0.736
PCA: PC 8 Logistic Regression Test Accuracy:  0.766
PCA: PC 9 Logistic Regression Test Accuracy:  0.783
PCA: PC 10 Logistic Regression Test Accuracy:  0.79
PCA: PC 11 Logistic Regression Test Accuracy:  0.77
PCA: PC 12 Logistic Regression Test Accuracy:  0.77
PCA: PC 13 Logistic Regression Test Accuracy:  0.79
PCA: PC 14 Logistic Regression Test Accuracy:  0.78
PCA: PC 15 Logistic Regression Test Accuracy:  0.8
PCA: PC 16 Logistic Regression Test Accuracy:  0.78
PCA: PC 17 Logistic Regression Test Accuracy:  0.80
PCA: PC 18 Logistic Regression Test Accuracy:  0.79
PCA: PC 19 Logistic Regression Test Accuracy:  0.79
PCA: PC 20 Logistic Regression Test Accuracy:  0.79
PCA: PC 21 Logistic Regression Test Accuracy:  0.8
PCA: PC 22 Logistic Regression Test Accuracy:  0.80
PCA: PC 23 Logistic Regression Test Accuracy:  0.8
PCA: PC 24 Logistic Regression Test Accuracy:  0.81
PCA: PC 25 Logistic Regression Test Accuracy:  0.79
```

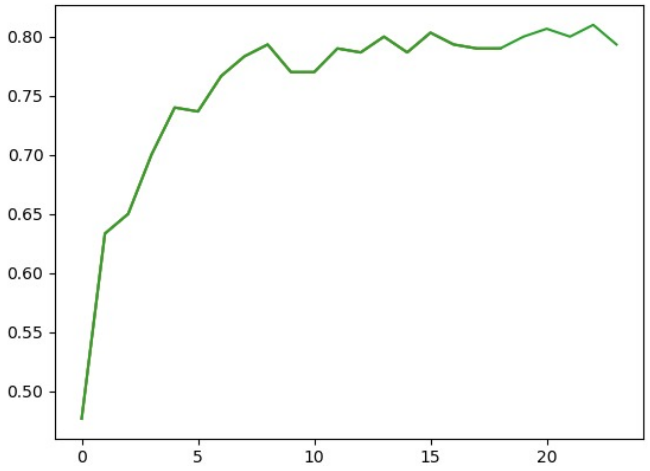


Figure:3 After PCA is applied.

CONVOLUTIONAL NEURAL NETWORK (CNN)

CNN is a sequence of layers, and every layer transforms one volume of activations to another through a differentiable function. We use three main types of layers to build the architecture. Convolutional Layer, Pooling Layer, and Fully-Connected Layer.

CNN transforms the original image layer by layer from the original pixel values to the final class scores. Some layers contain parameters and other don't. In particular, the fully-connected layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons).On the other hand, the Relu-Pool layers will implement a fixed function. The parameters in the Fully-connected layers will be trained with gradient descent so that the class scores that the CNN computes are consistent with the labels in the training set for each image.

We use Relu as activation function because it will not vanish the effect during your training process and this function more computationally efficient.

We use also softmax function in our output layer as a activation function. Softmax is just like a sigmoid function but it also divides each output such that the total sum of the outputs is equal to 1.The output of the softmax function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true.

With these settings, the initial models that have been tried with 5 epochs are as below.

Our first model is a basic MLP model, with two Dense layers.

```
# %% Model 1

model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

This model produces the following results, which are good but can be developed.

```
Final Scores
**************************
Accuracy: 0.887
Validation Accuracy: 0.868
Loss: 0.309
Validation Loss: 0.366
**************************
```

Our second model is more complex. We have a Batch Normalization layer at the start. In addition to Dense layers, Convolutional + Max Pooling layers are included. Also, Dropout layers are included to prevent overfitting.

```
model = keras.Sequential([

    keras.layers.InputLayer(input_shape=(28, 28, 1)),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters = 64, kernel_size = (
    keras.layers.MaxPool2D(pool_size = (2,2)),
    keras.layers.Dropout(0.2),
    keras.layers.Conv2D(filters = 64, kernel_size = (
    keras.layers.MaxPool2D(pool_size = (2,2)),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dropout(0.2),
    keras.layers.Dense(64, activation=tf.nn.relu),
    keras.layers.BatchNormalization(),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

Results are significantly better than our initial model.

```
Final Scores
**************************
Accuracy: 0.934
Validation Accuracy: 0.923
Loss: 0.178
Validation Loss: 0.225
**************************
```

Adding PCA to Model 2 doesn't improve the results.

```
Final Scores
**************************
PCA: PC256
Train Accuracy: 0.773
Validation Accuracy: 0.820
test Accuracy: 0.812
Train Loss: 0.621
Validation Loss: 0.502
Test Loss: 0.518
**************************
```

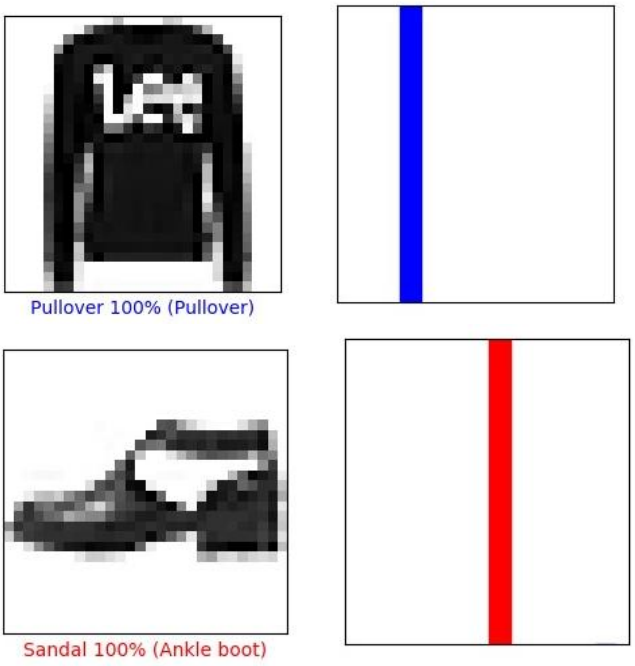
Comparing all different algorithms such as Random Forests, Logistic Regression, and Neural Networks; it is clear that our best model is Model 2. We try different optimizers for 5 epochs on this model. Results are as below:

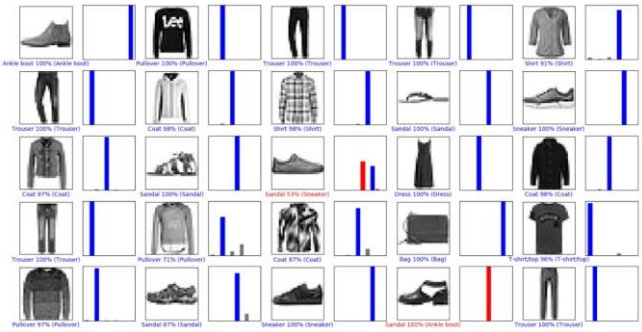
Optimizer	adam	SGD	RMSprop	Adadelta	Nadam
Loss	sparse_categorical_cross entropy	sparse_categorical_cross entropy	sparse_categorical_cross entropy	sparse_categorical_cross entropy	sparse_categorical_cross entropy
Accuracy	0,914	0,834	0,907	0,913	0,916
Validation Accuracy	0,909	0,855	0,911	0,891	0,913
Loss	0,232	0,454	0,258	0,241	0,221
Validation Loss	0,245	0,403	0,252	0,286	0,240

Best optimizers seem to be adam, RMSprop, and Nadam. For these optimizers we run the Model 2 for 10 epochs for a better comparison.

Optimizer	adam	RMSprop	Nadam
Loss	sparse_categorical_cross entropy	sparse_categorical_cross entropy	sparse_categorical_cross entropy
Accuracy	0,941	0,910	0,937
Validation Accuracy	0,923	0,901	0,920
Loss	0,156	0,252	0,166
Validation Loss	0,228	0,307	0,248

With our final model, some estimations can be seen below.





V.CONCLUSION

Contemporary neural networks are best suited for problems with many dimensions and huge sample sizes. Image recognition is such a problem, and it is no surprise neural networks perform better than other algorithms. CNNs (Convolutional Neural Networks) perform better than basic MLP, and PCA doesn't really improve the accuracy or significantly shortens training time. It is also seen that measures against Overfitting, such as Dropout in our case, are extremely important to maintain generalization for unseen data.

For personal reflections, we have found Fashion-MNIST dataset a good starting point to delve into practical applications of neural networs. We can say we have a much better grasp of both neural networks, and common tasks and issues in image recognition applications. We hope that we will be able to apply this knowledge in future applications in academia and business.

VI. REFERENCES

[1] Web Pages: https://medium.com/tensorist/classifying-fashion-articles-using-tensorflow-fashion-mnist-f22e8a04728a

[2] https://blog.valohai.com/clothes-detection-for-fashion-recommendation

[3] https://medium.com/nanonets/how-to-classify-fashion-images-easily-using-convnets-81e1e0019ffe

[4] http://cs231n.github.io/convolutional-networks/

[5] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/