CS4347 Introduction to Machine Learning – Homework 2

Alexander Muyshondt
Texas State University
601 University Dr.
adm1@txstate.edu

Abstract

Fashion-MNIST is a dataset of article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits. The goal of this paper is to train a Convolutional Neural Network (CNN) using this database.

1. Introduction

A CNN is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNN have the ability to learn these filters/characteristics.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

They have applications in image and video recognition, recommender systems, image classification, medical image analysis, and natural language processing.

2. Data

This project uses the Fashion-MNIST dataset. It consists of 60,000 training images and 10,000 test images. It is a MNIST-like fashion product database. Each example is a 28x28 grayscale image, associated with a label from 10 classes. We intend Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. Each image is in greyscale and associated with a label from 10 classes. Each training and test example is assigned to one of the following labels: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

3. Methods

This project was run using Python 3.7 along with Tensorflow and its associated packages. The train set and test set are given in two separate datasets. We began by examining the class distribution of our dataset as shown below in Figure 1.

```
Ankle Boot          :   6000 or 10.0%
Bag                 :   6000 or 10.0%
Sneaker             :   6000 or 10.0%
Shirt               :   6000 or 10.0%
Sandal              :   6000 or 10.0%
Coat                :   6000 or 10.0%
Dress               :   6000 or 10.0%
Pullover            :   6000 or 10.0%
Trouser             :   6000 or 10.0%
T-shirt/top         :   6000 or 10.0%
```

Figure 1: Class distribution of Fashion-MNIST training set.

A data preprocessing step was taken to prepare for the model.

The columns were reshaped from (784) to (28,28,1). Both the train_data and the test_data were processed and further split the train set in train and validation set. The validation

set will be 20% from the original train set, therefore the split will be train/validation of 0.8/0.2. A sequential model, a linear stack of layers, was used. The layers added are as follows:

- Conv2D is a 2D Convolutional layer (i.e. spatial convolution over images). The parameters used are: filters - the number of filters (Kernels) used with this layer; here filters = 32; kernel_size - the dimmension of the Kernel: (3 x 3); activation - is the activation function used, in this case relu; kernel_initializer - the function used for initializing the kernel; input_shape - is the shape of the image presented to the CNN: in our case is 28 x 28 The input and output of the Conv2D is a 4D tensor.
- MaxPooling2D is a Max pooling operation for spatial data. Parameters used here are: *pool_size*, in this case (2,2), representing the factors by which to downscale in both directions; Conv2D with the following parameters: filters: 64; kernel_size : (3 x 3); activation : relu;
- MaxPooling2D with parameter: *pool_size* : (2,2);
- Conv2D with the following parameters: filters: 128; kernel_size : (3 x 3); activation : relu;
- Flatten. This layer Flattens the input. Does not affect the batch size. It is used without parameters;
- Dense. This layer is a regular fully-connected NN layer. It is used without parameters; units - this is a positive integer, with the meaning: dimensionality of the output space; in this case is: 128; activation - activation function : relu;
- Dense. This is the final layer (fully connected). It is used with the parameters: units: the number of classes (in our case 10); activation : softmax; for this final layer it is used softmax activation (standard for multiclass classification)

The model was ran with the training set and also with the validation set (a subset from the orginal training set) for validation.

4. Results

Test accuracy is around 0.91.

The models accuracy was evaluated based on the predicted values for the test set. Figure 2 shows the plot of the train and validation accuracy and loss, from the train history.

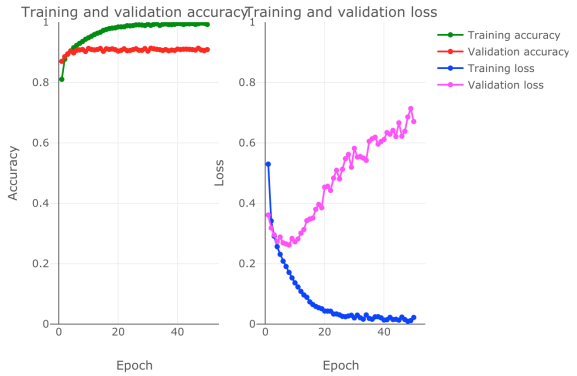

Figure 2

The validation accuracy does not improve after few epochs and the validation loss is increasing after few epochs. This confirms the assumption that the model is overfitted. By adding Dropout layers we can help avoiding overfitting.

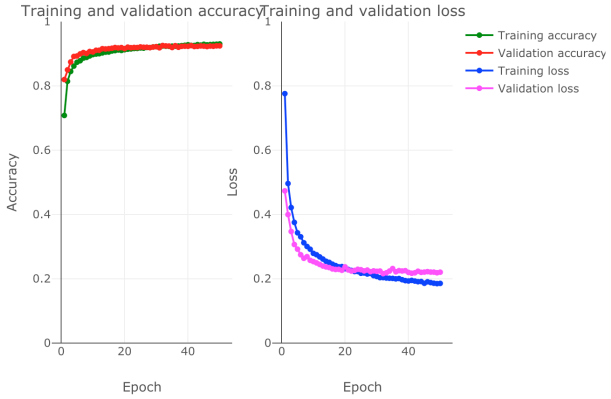

Figure 3

After adding the Dropout layers, the validation accuracy and validation loss are much better as shown in Figure 3.

With a complex sequential model with multiple convolution layers and 50 epochs for the training, we obtained an accuracy ~0.91 for test prediction. After investigating the validation accuracy and loss, we understood that the model is overfitting.
We retrained the model with Dropout layers to the model to reduce overfitting. We confirmed the model improvement and with the same number of epochs for the training we obtained with the new model an accuracy of ~0.93 for test prediction. Only few classes are not

correctly classified all the time, especially Class 6 (Shirt) and Class 2 (Pullover).

5. Discussion

While the Fashion MNIST dataset is slightly more challenging than the MNIST digit recognition dataset, unfortunately, it cannot be used directly in real-world fashion classification tasks, unless you preprocess your images in the exact same manner as Fashion MNIST (segmentation, thresholding, grayscale conversion, resizing, etc.).

In most real-world fashion applications mimicking the Fashion MNIST pre-processing steps will be near impossible.

References

[1] Fashion MNIST, An MNIST-like dataset of 70,000 28x28 labeled fashion images, https://www.kaggle.com/zalando-research/fashionmnist

[2] DanB, CollinMoris, Deep Learning From Scratch, https://www.kaggle.com/dansbecker/deep-learning-from-scratch

[3] DanB, Dropout and Strides for Larger Models, https://www.kaggle.com/dansbecker/dropout-and-strides-for-larger-models

[4] BGO, CNN with Keras, https://www.kaggle.com/bugraokcu/cnn-with-keras

[5] NAIN, EagerFMINST, https://www.kaggle.com/aakashnain/eagerfmnist