Use of deep learning to classify the fashion mnist dataset

Intelligent Systems report

by

Kévin Naudin

Neumann Promotion

ESIREM

Year 2018-2019

Index

Introduction to the project

Problem

We have at our disposal the fashion mnist dataset. It's a dataset containing 60,000 images of clothes, all in grey levels and with a size of 28x28. Each of these clothing item can be classified in one of these 10 categories :

1. T-shirt/Top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot

We want to be able to use a neural network that would recognize correctly each of these items, meaning it will be able to take any of these images as input and as an output give us the category in which it belongs.

A neural network can be evaluated thanks to many criteria. In our case, we will be measuring efficiency by looking at the val_acc (the real prediction accuracy of our model) and the number of parameters it uses (the number of connexion in the hidden layers of the network).

Keras

Keras is a high-level neural networks Application Programming Interface, written in Python and capable of running on top of Tensorflow, CNTK or Theano. It was developed with a focus on enabling fast experimentation. It allows us as developers to quickly create models of neural networks and use a list of already existing datasets to train on, fashion mnist for instance.

Building our model

Choosing the type of model

For the simple mnist problem we saw during class that we could use 2 different efficient type of model for the image classification problem: multi layered perceptron (MLP) and convolutional neural networks (CNN). What type should we use? According to the article “When to use MLP, CNN and RNN Neural Networks”[1], MLP are best used when the input data comes from tabular datasets whereas CNN are best for image data. We can confirm that thanks to articles like “A comparison study between MLP and Convolutional Neural Network models for character recognition“[2] which analyses the different results from CNN and MLP in a context where the input are little images. What information we gather though is that there are no good answer to our question. It also seems that during the learning phase, MLP are more robust than CNN and that makes them in theory better classifier in general. What we are going to do then is to assemble a hybrid model composed with a MLP on top of a CNN.

The CNN will be able to work with image data efficiently because one of its purpose is the extraction of feature in 2D-arrays that have cells that share a spatial relationship, which is exactly the case in an image since a group of pixel always is going to represent something. We’ll then send all the data coming from the image feature to an MLP that will be able to learn efficiently the difference between each set of feature extracted.

Note: By using the simple 2 networks that we used during class (a simple CNN and a simple MLP) we only achieved a val_accuracy of 90%.

Setting up the layers in our model

To assemble my model I decided to use as base the CNN we previously did in class. Our model is going to be a sequential one, meaning each layer will follow the previous one and so on. We begin by assembling the first CNN:

- 2D convolutional layer of 32 and a kernel size of (3,3) in which we state that our images, our inputs, are going to be of size 28,28,1. Convolutional layer create filter of convolution of the chosen kernel size and apply them to the input. That way they’re able to create filters that will extract features in the image.
- 2D convolutional layer of 64 and a kernel size of (3,3).
- A 2D maxpooling layer with a pool size of 3,3. The maxpooling layer is used to reduce the data so that our model don’t extrapolate too much from every filter used.
- A 2D dropout of 0,2. The dropout is a layer that is used to randomly set neurons to 0 so that our model do not learn our dataset to quickly. If it did, it would perfectly recognize our dataset but not other images we would give it later.

Note about dropout: The value they are set at are not random. According to the article “Dropout: A Simple Way to Prevent Neural Networks from Overfitting“[3] and multiple thread on the MachineLearning and LearnMachineLEarning subreddits, it is apparently best to use values between 0,4 and 0,5 for dropouts right after a MLP; 0,2 to 0,3 after a CNN.

For more feature extraction, we assemble a second CNN on top of the previous one:

- 2D convolutional layer of 64 and a kernel size of (3,3).
- 2D convolutional layer of 128 and a kernel size of (5,5).
- A 2D maxpooling layer with a pool size of 3,3. The maxpooling layer is used to reduce the data so that our model don’t extrapolate too much from every filter used.
- A 2D dropout of 0,2. The dropout is a layer that is used to randomly set neurons to 0 so that our model do not learn our dataset to quickly. If it did, it would perfectly recognize our dataset but not other images we would give it later.

At the beginning stage of the project we tried using a third CNN but it didn’t improve our results and just added time and parameters to our model. The idea of combining multiple CNN went from how the YOLOv3[4] detector was made. YOLOv3 is used in image detection and recognition and it works by adding CNNs over CNNs and ending with a MLP. Hence we assemble our MLP here:

- Flattening our data so it can fit into the MLP
- A dense layer of 32; Dense layers connect everything they can to them.
- A dropout of 0,4 (see above for explanation of value choice)
- A dense layer of 10, corresponding to the 10 possible image category. As to now, each of the layer we added had a relu activation. For this layer, we use softmax so the model can choose between multiple values which one is the highest (obviously corresponding to the predicted class).

Parameters of the model

The optimizer

In models, the optimizer is the module that will regulate how the learning process works and how the learning rate is supposed to be set. To choose what optimizer to use, we tried them one by one on our model and looked at the one that gave the best result. This one was RMSprop at first. Then we discovered the amsgrad option of Adam. According to the article “ON THE CONVERGENCE OF ADAM AND BEYOND“[5], this Adam extension if made for optimizing problems that deal with spatial features, like problem with images. The use of Adam with this option did give us a better result for our model.

The data augmentation

One of the problem we had when learning was that our model learnt our dataset too quickly, even with the dropout. The idea behind data augmentation is to take our dataset and modify a batch of images coming from it. That way our network will never encounter twice the same image when dealing with data generation.

The batch size and number of epochs

Again, the values here come from experimenting with Keras. We found that using a batch size of 242 gave us the best values. For the number of epochs, we used 500 as a max value so we could check whether or not the model kept learning and learning. It is not the case.

The learning rate reducer

There would be moment where the model would stop learning and the loss wouldn't change. In that case, Keras allows us to use a plateau to check if the loss value becomes stuck. In that case, it will automatically change the learning rate of our model to try and un-stuck us.

Results

As a result of our model, we obtain a val_accuracy of 94,61% for using less than 300K parameters. It gets close to 95% for a third of the parameters used by a tiny detector.

Bibliography

1: Jason Brownlee, When to Use MLP, CNN, and RNN Neural Networks, 2018
2: Syrine Ben Driss, M Soua, Rostom Kachouri, Mohamed Akil, A comparison study between MLP and ConvolutionalNeural Network models for character recognition, 2017
3: Nitish Srivastava nitish@cs.toronto.eduGeoffrey Hinton hinton@cs.toronto.eduAlex Krizhevsky kriz@cs.toronto.eduIlya Sutskever ilya@cs.toronto.eduRuslan Salakhutdinov rsalakhu@cs.toronto.edu, Dropout: A Simple Way to Prevent Neural Networks fromOverfitting, 2014
4: Joseph Redmon, Ali Farhadi, YOLOv3: An Incremental Improvement, 2018
5: Sashank J. Reddi, Satyen Kale & Sanjiv Kumar, ON THE CONVERGENCE OF ADAM AND BEYOND, 2018