University of Waterloo

ECE 657A:
Data and Knowledge Modeling and Analysis

Spring 2021

Assignment 3:

Clustering and Classification with Deep Learning
on the "FashionMNIST with a Twist!" Dataset

Due: April 2, 2022 at 11:59pm EST

Late Penalties: No late penalities until April 4 11:59pm. So if you are ready to hand in April 2, just do it, we can start grading. If not, you have until Monday night. But then you should switch to exam studying!

Overview

Broad objectives:

- Set up and train two types of Convolutional Neural Networks (one standard, one your own choice) using a modern python neural network library for classification on the FashionMNIST dataset.
- Shorter, concise report than previous assignments. The focus on design of your model and results, rather than data preprocessing or feature engineering.

Grading Rubric: Even though we are not using Kritik for grading Asg3, we will use the same rubric as the revised one shared for

Collaboration:

- You can do your work alone or in pairs.
- You can collaborate on the right tools to use and setting up your programming environment.
- If you are working in a pair you will need to join a "group" on Crowdmark. Even if you are working alone, you need to sign up for a group on LEARN to get access to a dropbox for your code.

Hand in:

- One report per team, via the CROWDMARK site in PDF format. You will need to divide the PDF up into multiple files and drag and drop each onto the relevant `[CM#]` questions. You should receive an invite to crowdmark by email.
- Your report should be as concise as possible, show the main results and analysis without showing repeated versions of the same output. You should include small pieces of code in your report to demonstrate the core aspects of what you did.
- You will also submit the code/scripts needed to reproduce your work as a python jupyter notebook to the LEARN dropbox.

Tools: You can use any libraries available in python, tensorflow, pytorch, keras, scikitlearn for this project. You need to mention explicitly which libraries you are using, any blogs or papers you used to figure out how to carry out your calculations.

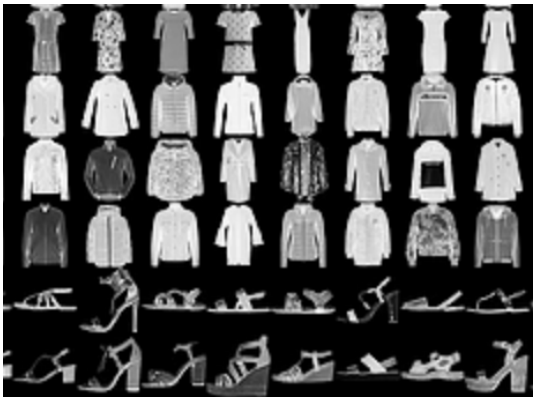

Figure 1: Small sample of greyscale elements of the FashionMNIST dataset.

Data Set - Fashion MNIST . . . with a Twist!

This is an image dataset based on publicly available dataset Fashion MNIST:

`https://github.com/zalandoresearch/fashion-mnist`

As shown in Fig. 1 The dataset is composed of small (28x28 pixels), grey-scale images of clothing items such as shoes, coats, pants, etc. It was created to serve as a direct drop-in replacement for the well known MNIST dataset (`http://yann.lecun.com/exdb/mnist/`) dataset for benchmarking machine learning algorithms. So the dataset shares the same image size and structure as the original MNIST.

The Twist...

The label was originally an assigned clothing type represented by an integer from 0-9. The training set contains 60,000 examples and the test set 10,000 examples.

For this assignment, we are providing you the same input features, but the **labels** you will be using will be a **new mystery label**. We have created a new label based on the data and calculated to be associated with each entry. The `y_train` and `y_test` files provide this modified label which is a category numbered 0-4. What does the number mean? That's for you to figure out through your analysis!

File Descriptions:

- `x_train.csv` - the training set for your model. The training file contains vectors of size 784 representing pixel values of a 28x28 image.
- `y_train.csv` - is the "mystery label", a numeric target obtained using our formulation.
- `x_test.csv` - are the features for the test set to use for final evaluation.
- `y_test.csv` - the "mystery label" for the test data.

- intput x
- conv 3x3 with 32 filters, padding 1x1
- max pool 2,2
- conv 3x3 with 32 filters, padding 1x1
- flatten output
- fully connected layer
- ReLU activation
- optimizer : any you choose, but Adam suggested
- softmax output layer to select the highest value for classification from the network outputs

Classification with Convolutional Neural Networks

Default Network

[CM1] Classify the data using a Convolutional Neural Network with the following setup.

Include a printout of the summary of the model and state the accuracy results of running this model.

- Two CNN layers with Kernel size of (3,3) and 32 filters, stride of 1 with padding size of (1,1) and max-pooling between these two CNN layer with pool size of (2,2) and at the end one fully that map the last CNN layer to 5 outputs.
- ReLU activation functions.
- Softmax output layer for the classification decision

Your Own Network

[CM2] Classify the data using any other architecture that you want to try. For example, you may want to put some Fully Connected layers after the last output of the CNN, you may want to try more CNN layers, or different parameters for the existing layers. Whatever you try, be sure to provide an explanation of your model (algorithms, network architecture, optimizers, regularization, design choices, numbers of parameters) that would be sufficient for someone to recreate it.

- You can use any CNN-based approach or other DNN variants to solve the classification problem. You can explore any architecture of the network you like. Be sure to cite any sources you used.
- Whatever you choose, provide a concise and clear description of your architecture sufficient to implement it, some justification for your choices and reference any materials you used along the way.
- Show some of the important code blocks to implement your model. We will also consult your full code on LEARN, so this is your chance to guide us to understand your code and how you achieved your result.

Results Analysis

Also report accuracy results based on the test set.

[CM3] Briefly report on the following:

- Runtime performance for training and testing.
- Comparison of the different parameters or designs you tried.
- You can use any plots to explain the performance of your approach. But at the very least **produce two plots**, one of **training loss vs. training epoch** and one of **classification accuracy vs. training epoch** on both your training and ~~test~~ set.

"both your training and validation set."

Note: Models need to be trained from scratch! You can't use a pretrained image processing model for this assignment. If the computational wait time is too heavy, reduce the size of the network and explain, I don't mind changes to the default network if you're computation is too slow. But you need to see it train from scratch.

Using Your Own Encoding

One of the exciting things about deep learning is that the learned model is not just a black box producing predictions, or classification mappings. Your network is a huge, multi-layered machine turning the input data into a series of complex encodings. The `softmax` layer at the end of your network reads off one interpretation of these encodings to give you a classifier. Another thing you could do is take the last fully connected layer (or any layer you choose actually) and use that as a compact representation of the data, just as we treated the output of PCA and LDA.

In this part, you should define a simple intermediate layer model using one of these final layers from your trained network. Then you will treat the elements of this encoding as features in a dataset for clustering and visualization to help you understand what the mystery label for our dataset might mean.

`[CM4]` Carry out the following activities:

All of these activities should be carried out by applying your trained network to the test data and using the encoding (as described) as your features.

- Visualize your encoding with the first two components from PCA, the colour mapping could be the label values.
- Perform DBSCAN and K-means clustering algorithms on the features that you have extracted from your own designed model with 5 clusters and visualize the results. Use the resulting clusters as alternate colour mappings for the PCA plot above.
- Apply t-SNE on the features that you have extracted for your own designed model and visualize the results in the same way.
- Based on the results of clustering and t-SNE can you guess what are the labels for the given dataset? It might also help to list out a random selection of data entries (the original images) for each cluster and their label value to help understand the patterns each cluster might represent.
- Feel free to try some other approach to using this encoding in a creative way.

This is not done inline, so the classifier is not another source for the training signal (although…that's interesting idea…no).

After training of your network is complete, run test data (or other data) through the network without training to extract the encoding for each datapoint, just like you would with PCA.