

Machine Learning Nanodegree Program

Capstone Project

On

Image classification Of Fashion-MNIST

dataset

By

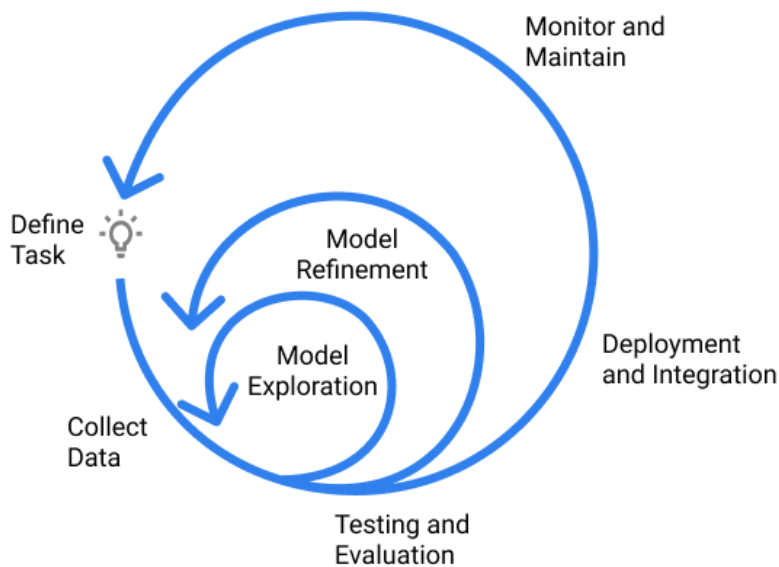
Ikwuonwu Jonathan Ukaegbu

Project Overview

Online fashion market is a constantly growing sector, and an algorithm capable of identifying different clothes can help companies (small, medium and large) in the clothing sales sector to understand the profile of potential customers and focus on sales targeting specific trends, as well as the taste of their customers and to also improve user experience. In this capstone project, I used the Fashion-MNIST (fashion modified national institute of technology) dataset. The fashion-MNIST dataset is made to help researchers find models to classify products such as clothes and in this project I used the Fashion-MNIST for image classification to create an efficient model with convolutional neural network. The aim of this capstone project, is basically to master the use of image classification models using AWS SageMaker resources with PyTorch to perform a computer vision task.

Problem statement

To solve the classification problem with the help of a convolutional neural network (CNN) I will build a model to sort clothes into 10 different categories. I will load the data set into S3 and perform training on the preprocessed data sets, I will perform debugging and create referencing and prediction. The diagram below gives a general idea of the lifecycle of this project



Source: <https://www.jeremyjordan.me/ml-projects-guide/#data>

Solution statement

The proposed solution is to create a deep learning model that is able to accurately identify different clothing categories with a high percentage accuracy by using Amazon SageMaker. To finish this project, I will have to perform the following tasks:

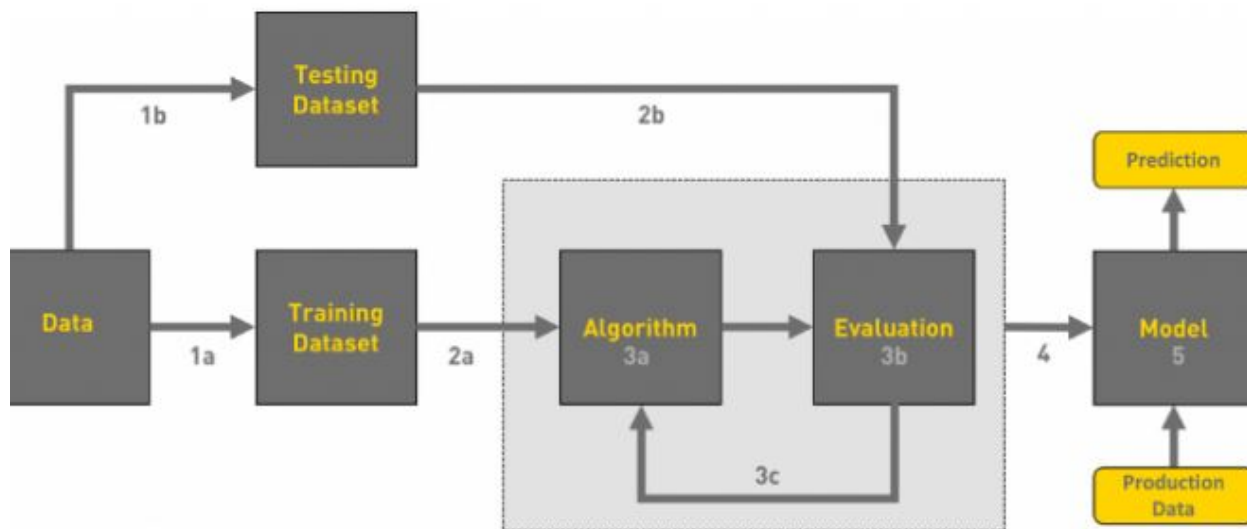
Upload Training Data: First I will have to upload the training data to an S3 bucket.

Model Training Script: Once I am done, I will have to write a script to train a model on that dataset.

Train in SageMaker: Finally, I will use SageMaker to run that training script and train the model

The dataset is loaded into python using pytorch library. The loaded data will be first explored and visualized using numpy and matplotlib library to understand the nature of the data. Exploring the data will help us in deciding how to approach and whether any preprocessing of the data is needed. Preprocessing of the data is done as required. Then the compiled model will be trained on the training data and evaluated using accuracy score against the testing data. Then the results can be analyzed and compared with respect to the benchmark model to know the overall performance of the model.

The different to be taken to achieve this solution is as follows:



Overview of the Workflow of ML

Metrics

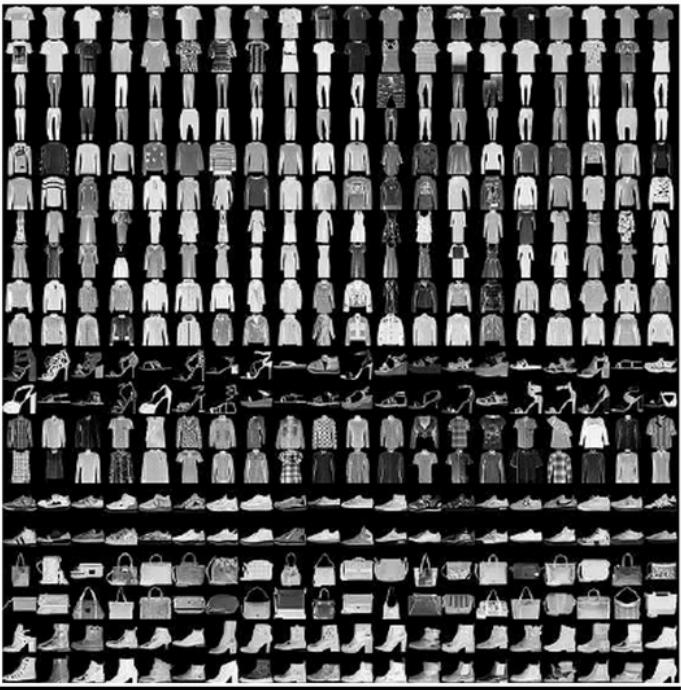
Accuracy is the proportion of samples predicted correctly among the total number of samples examined. For example, in our problem, it is the ratio of the clothing image predicted correctly to the total number of clothing images evaluated. My aim to achieve a high accuracy after completing the project was met. Here is a screenshot of the percentage accuracy of the different classes represented in the dataset.

```

In [46]: 1 # We know the size of each class in testset is 1000
          2 cls_correct = [0 for _ in range(10)]
          3 for k in range(correct_preds.shape[0]):
          4     n = correct_preds['label'][k]
          5     if correct_preds['prediction'][k] == n:
          6         cls_correct[n] += 1
          7 cls_accu = []
          8 for n in range(10):
          9     cls_accu.append(cls_correct[n]/1000)
          10     print(f'The accuracy of label {n} is: {cls_accu[n]: .04f} or {cls_accu[n]*100: .0f}%')
          11
          12
The accuracy of label 0 is: 0.8580 or 86%
The accuracy of label 1 is: 0.9780 or 98%
The accuracy of label 2 is: 0.8610 or 86%
The accuracy of label 3 is: 0.9060 or 91%
The accuracy of label 4 is: 0.8450 or 84%
The accuracy of label 5 is: 0.9590 or 96%
The accuracy of label 6 is: 0.7040 or 70%
The accuracy of label 7 is: 0.9550 or 96%
The accuracy of label 8 is: 0.9780 or 98%
The accuracy of label 9 is: 0.9540 or 95%
  
```

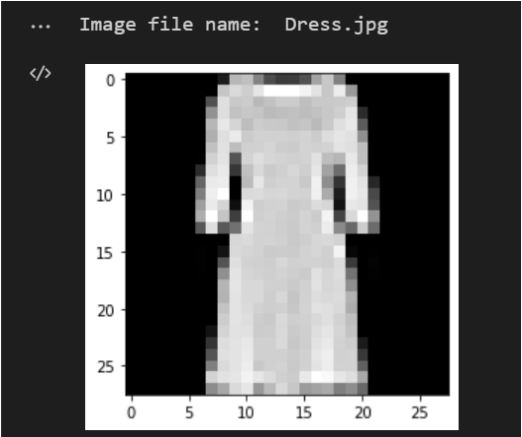
Datasets

The dataset that would be used for this project is the fashion- MNIST dataset, made publicly available. It consists of about 60000 images grouped in 10 classes of fashion wear. This dataset contains grey scale images of sizes 28 by 28. For easy usage and processing, I will split the data into train, valid and test folders, containing subfolders subsequently. In training the model, I will use inputs such as pre-trained models (ResNet18) to perform training and evaluation as well. After preparing the data, I will upload it to AWS S3 where training was carried out with pyTorch

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

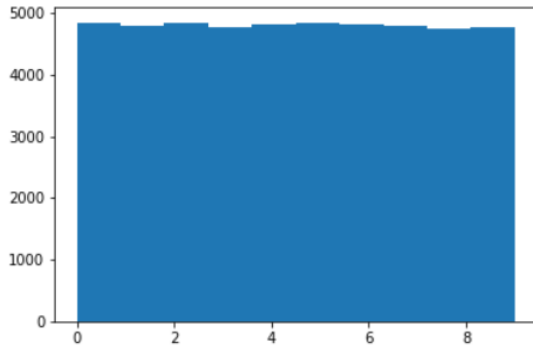
Exploratory Visualization

Let’s visualize an image of a digit in the training set of the Fashion-MNIST dataset. The image contains 28x28 pixels as represented in the image below.The image is a grayscale image.



The fashion-MNIST database has a training set of 60,000 samples, and a test set of 10,000 samples. The clothes have been size-normalized and centered in a fixed-size image. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. the images were centered in a 28x28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28x28 field.

```
1 x = trainset.label
2 plt.hist(x, bins = 10)
3 plt.show()
```



As seen in the image above, the graph shows the distribution of the 10 classes contained in the fashion MNIST. The training dataset is very balanced having a similar number of images (a little bit less than 5000) in each class. This, among other characteristics, makes the fashion MNIST dataset a standard dataset

Benchmark of the model

The benchmark created for this project is gotten from this

link: <https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-fashion-mnist-clothing-classification/>

The benchmark model solves the same problem using keras, operated on 10 training epochs with a default size of 32, data transformation was also done by using normalization etc, the stochastic Gradient Descent (SGD) was used as the optimizer as well. The evaluation metric used is also accuracy and the model achieved an accuracy of 90.990%. My model performed fairly well with an accuracy of 89.98%.

iii Methodology

Data processing

Since pytorch library was used to download and then loaded the fashion-MNIST dataset directly into numpy arrays, not too intensive preprocessing steps are needed, The fashion-MNIST dataset contains 60000 images, the images are grayscale(as a result of the anti-aliasing technique used by the normalization algorithm), the images that are small (28x28) and of the same size and the size distribution of the dataset is highly balanced. In order to train on a neural network algorithm, i :

- I converted the images to a tensor which basically generalizes vectors or matrices.
- I applied transforms.grayscale, which returns the images in a single channel
- I applied random horizontal flips of 50% probability to avoid bias (as much as possible) when training the model.

- I also normalized since it helps to improve the model. Normalization helps get data within a range and reduces the skewness which helps learn faster and better. It may also tackle the diminishing and exploding gradients problems that may occur during training.

Algorithms and techniques

Given that the problem is a supervised learning problem and more specifically a classification problem, the algorithm chosen for this is the convolutional neural network, a machine learning tool perfect for learning complex patterns on a large dataset. The hyperparameter chosen for tuning is the learning rate and batch size, these parameters greatly affect the performance of a model. After finding a very good parameter for tuning with use of `fm_hpo.py` script as `entry_point` in the estimator, adequate data preprocessing was used to conduct model training (see data preprocessing section) on the fashion-MNIST dataset. A convolutional neural network for fashion MNIST classification from scratch. For optimizers, which is used to specify how the neural network will learn, the stochastic Gradient Descent (SGD) was used as the optimizer to run with back propagation during model training.

Convolutional neural network (CNN): is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex.

Stochastic Gradient Descent (SGD) Optimizer: In Gradient Descent (GD) optimization, we compute the cost gradient based on the complete training set; hence, we sometimes also call it batch GD.

```
1 # TODO: Get the best estimators and the best HPs
2 best_estimator = tuner.best_estimator()
3
4 #Get the hyperparameters of the best trained model
5 best_hypers = best_estimator.hyperparameters()
6
7 # To know exact data from best_estimator.hyperparameters()
8 print(best_hypers)
```

```
2022-02-21 15:27:18 Starting - Preparing the instances for training
2022-02-21 15:27:18 Downloading - Downloading input data
2022-02-21 15:27:18 Training - Training image download completed. Training in progress.
2022-02-21 15:27:18 Uploading - Uploading generated training model
2022-02-21 15:27:18 Completed - Training job completed
{'_tuning_objective_metric': 'Test Loss', 'batch_size': '32', 'lr': '0.03138192028805053', 'sagemaker_container_log_level': '20', 'sagemaker_estimator_class_name': 'PyTorch', 'sagemaker_estimator_module': 'sagemaker.pytorch.estimator', 'sagemaker_job_name': 'pytorch_fashionmnist-2022-02-21-15-03-22-137', 'sagemaker_program': 'fm_hpo.py', 'sagemaker_region': 'us-east-1', 'sagemaker_submit_directory': 's3://sagemaker-us-east-1-324194532919/pytorch_fashionmnist-2022-02-21-15-03-22-137/source/sourcedir.tar.gz'}
```

This is the result of the best estimator best parameter used

Project Design

The workflow followed are as follows:

LOADING THE DATASET INTO THE NOTEBOOK

The dataset was downloaded with matplotlib's `plt.imsave` code as follows:

```

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-idx3-ubyte.gz to fmnist/FashionMNIST/raw/train-images-idx3-ubyte.gz
  0%|          | 0/26421880 [00:00<?, ?it/s]

```

LABELING THE CLASSES

NUMBER OF IMAGES FOR TEST, TRAIN AND VALID

```
1 print(len(test_df), len(train_df), len(valid_df))
```

10000 48000 12000

UPLOADING DATA TO S3 BUCKET

```

1 sess = sagemaker.Session()
2 bucket = sess.default_bucket()
3 print("Default Bucket: {}".format(bucket))
4
5 my_session = boto3.session.Session()
6 region = my_session.region_name## TODO: fill in
7 print("AWS Region: {}".format(region))
8
9
10 role = get_execution_role()## TODO: fill in
11 print("RoleArn: {}".format(role))

```

```

Default Bucket: sagemaker-us-east-1-324194532919
AWS Region: us-east-1
RoleArn: arn:aws:iam::324194532919:role/service-role/AmazonSageMaker-ExecutionRole-20220220T161955

```

```

1 # Upload data to S3 bucket.
2 inputs = sagemaker_session.upload_data(path=datapath, bucket=data_bucket, key_prefix=datapath)
3 print(inputs)

```

FITTING THE TUNER AND ESTIMATOR FOR THE TRAINING JOB

The learning rate and batch size as well as the estimator and tuner were set up resulting in fitting:

Fit the tuner

```

7]: 1 s3_data = "s3://{}/{}/".format(bucket, "images")
2 s3_output_dir = "s3://{}/{}/".format(bucket, "output")
3 s3_model_dir = "s3://{}/{}/".format(bucket, "model")
4
5
6
7 os.environ['SM_CHANNEL_TRAIN']=s3_data
8 os.environ['SM_MODEL_DIR']=s3_model_dir
9 os.environ['SM_OUTPUT_DATA_DIR']=s3_output_dir
10
11
12 tuner.fit({'train' : s3_data},wait=True)

```

```

.....
.....
.....!

```

The entry_point for fitting was the fm_hpo.py script which contains code that transforms the data by flipping, normalizing and transforming it to tensor during preprocessing. Other functions in the script were the train, test, create data loaders using arguments such as a criterion, model, data loader, optimiser, etc.

DESCRIBING THE TUNING RESULT

Prepare to perform Training on Best Estimator ¶

```
[170]: 1 best_estimator.hyperparameters()

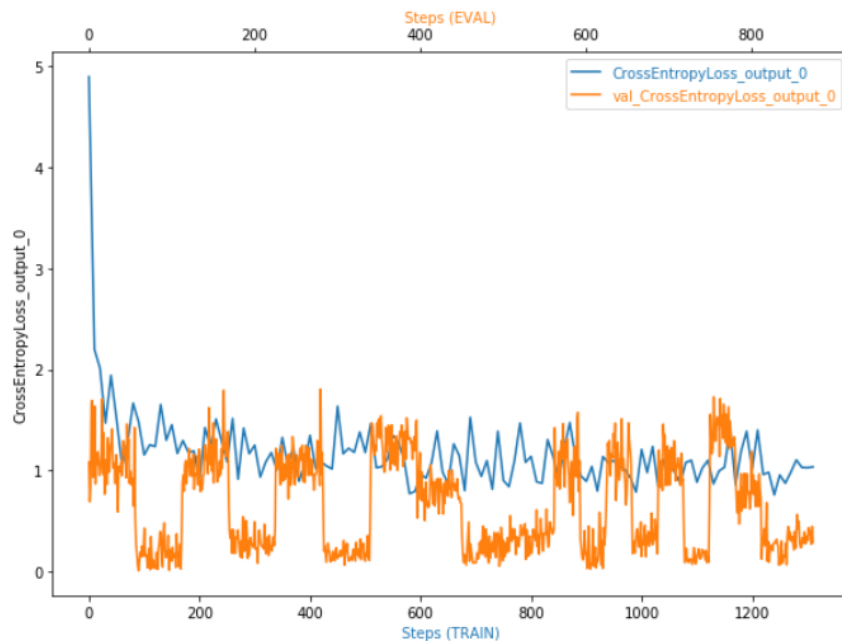
!t[170]: {'_tuning_objective_metric': 'Test Loss',
'batch_size': '"32"',
'lr': '0.03138192028805053',
'sagemaker_container_log_level': '20',
'sagemaker_estimator_class_name': 'PyTorch',
'sagemaker_estimator_module': '"sagemaker.pytorch.estimator"',
'sagemaker_job_name': '"pytorch_fashionmnist-2022-02-21-15-03-22-137"',
'sagemaker_program': '"fm_hpo.py"',
'sagemaker_region': '"us-east-1"',
'sagemaker_submit_directory': '"s3://sagemaker-us-east-1-324194532919/pytorch_fashionmnist-2022-02-21-15-03-22-137/source/source.tar.gz"'}
```

The best parameters were used to perform training. using of fm_model.py script as entry_point, fashion cnn model created was used on the best parameters

PROFILING AND DEBUGGING

```
1 plot_tensor(trial, "CrossEntropyLoss_output_0")
```

loaded TRAIN data
loaded EVAL data
completed TRAIN plot
completed EVAL plot



Cross-Entropy Loss Function. Also called logarithmic loss, log loss or logistic loss. Each predicted class probability is compared to the actual class desired output 0 or 1 and a score/loss is calculated that penalizes the probability based on how far it is from the actual expected value. The loss function which computes a value that estimates how far away the output is from the target was calculated during this project as cross entropy loss. The graph above simply shows the reduction in cross-entropy output which is the loss function of the model.

PROFILER REPORT

```
: 1 #import os
2
3 # get the autogenerated folder name of profiler report
4 profiler_report_name = [
5     rule["RuleConfigurationName"]
6     for rule in estimator.latest_training_job.rule_job_summary()
7     if "Profiler" in rule["RuleConfigurationName"]
8 ][0]
```

```
: 1 ! tar czf ProfilerReport.tgz ProfilerReport/
```

MODEL DEPLOYMENT

After carrying out the the training job with the best hyperparameter i went on to do model deployment

```

1 jpeg_serializer = sagemaker.serializers.IdentitySerializer("image/jpeg")
2 json_deserializer = sagemaker.deserializers.JSONDeserializer()
3
4
5 class ImagePredictor(Predictor):
6     def __init__(self, endpoint_name, sagemaker_session):
7         super(ImagePredictor, self).__init__(
8             endpoint_name,
9             sagemaker_session=sagemaker_session,
10            serializer=jpeg_serializer,
11            deserializer=json_deserializer,
12        )

```

```

1 pytorch_model = PyTorchModel(model_data=model_location,
2                               role=role,
3                               entry_point='fm_reference.py',
4                               py_version='py36',
5                               framework_version='1.8',
6                               predictor_cls=ImagePredictor)

```

```

1 from time import time

```

```

1 begin = time()
2 predictor = pytorch_model.deploy(initial_instance_count=1, instance_type='ml.m5.large')
3 c_time = time()-begin
4 print('Creating Endpoint Time: {:.01f}s'.format(c_time))

```

-----!Creating Endpoint Time: 187.4s

PREDICTIONS

```

1 output = predictor.predict(input)
2 output

```

```

[[0.5479651689529419,
 1.8651485443115234,
-2.5988097190856934,
 5.487130641937256,
 1.931625485420227,
-6.758813381195068,
 3.092844247817993,
-2.8216121196746826,
-1.2999895811080933,
-0.9873822927474976]]

```

```

1 import numpy as np
2 prediction = np.argmax(output)
3 print('prediction:', prediction, ', ', filename, 'label: ', label)
4 if prediction == label:
5     print('Prediction is correct')
6 else:
7     print('Prediction is not correct')

```

prediction: 3 , images/test/6/Shirt_586.jpg label: 2
Prediction is not correct

	fullname	label	prediction
0	images/test/9/Ankle-Boot_1.jpg	9	9
1	images/test/2/Pullover_1.jpg	2	2
2	images/test/1/Trouser_1.jpg	1	1
3	images/test/1/Trouser_2.jpg	1	1
4	images/test/6/Shirt_1.jpg	6	6

```
1 accu = sum(equal_indexes)/testset.shape[0]
2 print(f'The accuracy of all test images is: {accu:0.4f} or {accu*100: .0f}%')
```

The accuracy of all test images is: 0.8998 or 90%

```
1 cls_names = list(pd.read_csv('classes.csv')['class_name'])
2 cls_names
```

```
['T-Shirt',
 'Trouser',
 'Pullover',
 'Dress',
 'Coatadd_label',
 'Sandal',
 'Shirt',
 'Sneaker',
 'Bag',
 'Ankle-Boot']
```

IMPROVEMENT

The model used in this project uses a simple convolutional neural network, there are ones which can be used like the VGG-16(13). There is always room for improvement on an image classification problem like this. In other words, these are possible areas that can improve on the performance of the model such as using a different optimizer which prove to improve on the model's performance, as learning is very critical when it comes to model training

CONCLUSION

In conclusion when comparing my result to the benchmark with 90% accuracy, my model performed pretty well when compared to the benchmark. I will say I succeeded in creating a good model, although the benchmark created the convolutional neural network from scratch with Keras and I used the ResNet18 pretrained model on a PyTorch framework. Thus, the model solved the problem with a good accuracy score and is reliable. I found some areas of the project tasking as I am still new to deep learning but was able to scale through by watching some tutorials. By concluding this project, I learnt how to use PyTorch for image classification.

REFERENCE

<https://medium.com/@aaysbt/fashion-mnist-data-training-using-pytorch-7f6ad71e96f4>

<https://towardsdatascience.com/how-to-train-an-image-classifier-in-pytorch-and-use-it-to-perform-basic-inference-on-single-images-99465a1e9bf5>

<https://github.com/zalandoresearch/fashion-mnist>