Problem statement:

We have image dataset from FashionMNIST. It was 70 thousand samples and 10 classes ranging from shirts to pents to trousers to shoes. We want to a model that can classify them

We are going to use:

torchvision.**datasets**// it has FashionMNIST dataset

torchvision.**transforms**// it has ToTransfrom method, we use it to transfrom our images to tensors

We use squeeze() method to remove any extra dimention from tensor to only give the image size to the plot function (plt)

torchvision. **DataLoader**

It turns our data into small chunks of iterables for our model as it is more computationally efficent

Gradient decent is performed once per mini-batch rather then on the whole dataset once per epoch which would be computationally expensive and inefficent as well

We are flattening our tensor from [1,28,28] to [1,784]. NN.flatten compresses two dimentions into one

We are going to make one model

FashionMNISTModelV2 our first CNN model that micmics TinyVGG architecture

We have helper method **accuracy_fn()**

We have **print_train_time().** We are using tdqm.auto for progress bar

We are creating a function named **eval_model** to evaluate our model performance, We will create a model that helps us give only model performance in terms of **training time, loss and accuracy** we will give it data in form of **data_loader**

We are also creating **train_step** and **test_step** for mudularizing our training loop and testing loop

We are building a CNN now convolutional neural network

Objects in class

- Get the data from FNIST

- Prepare the data, put in into batches of 32
  It has test_dataloader and train_dataloader

- Accuarcy_function()
  A function we downloaded, we can use it to check the accuracy of our class

- Print_train_time()
  A function we created to measure the training time of our model

- Eval_model()
  A function we create to evaluate our models perforamce, it takes model as a parameter,data_loader object as a parameter, loss_fn and accuracy_fn as a parameter)
  Returns model name, accuracy and loss

- We set up device agnostic code to put our device on GPU (if available)

- Train_step() and Test_step()
  We create training the testing loops so we don't have to write trainging and testing our model loop. It take model, data_loader, loss_fn, optimizer, device
  Returns training loss and accuracy

- Building CNN (Convolutional Neural Network)
  CNNs are known for their ability to find patterns in visual data. We will use TinyVGG CNN model

- FashionMNISTModelV2
  Our model is CNN TinyVGG model. It used Conv2d, Relu layer and Maxpool2d layer

- Make_predictions()
  Takes model and data, make predictions on trained model

- We can do something better, we can make confusion matrix to better see what is our model doing

```
100%|██████████| 5148/5148 [00:00<00:00, 17554696.74it/s]Extracting /content/FMNIST_Dataset/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to /content/FMNIST_Dataset/FashionMNIST/raw

100% 3/3 [02:48<00:00, 56.55s/it]
Epoch: 0
---------
Train loss: 0.59518 | Train accuracy: 78.38%
Test loss: 0.39500 | Test accuracy: 85.99%

Epoch: 1
---------
Train loss: 0.36536 | Train accuracy: 86.90%
Test loss: 0.35244 | Test accuracy: 86.91%

Epoch: 2
---------
Train loss: 0.32588 | Train accuracy: 88.13%
Test loss: 0.32719 | Test accuracy: 88.07%

Train time on cpu: 168.387 seconds
Test sample image shape: torch.Size([1, 28, 28])
Test sample label: 5 (Sandal)
Making predictions: 100% 313/313 [00:04<00:00, 76.45it/s]
```

-

```
728.8/728.8 kB 20.8 MB/s eta 0:00:00
1.4/1.4 MB 49.7 MB/s eta 0:00:00
mlxtend version: 0.22.0
```