

Deep Learning and Neural Network

Mnist Fashion Classification

Name	NetID
Nourhan Abdelkerim	21nmma1
Sondos Mahmoud	21smam1
Farah Saber	21fsae

Project description:

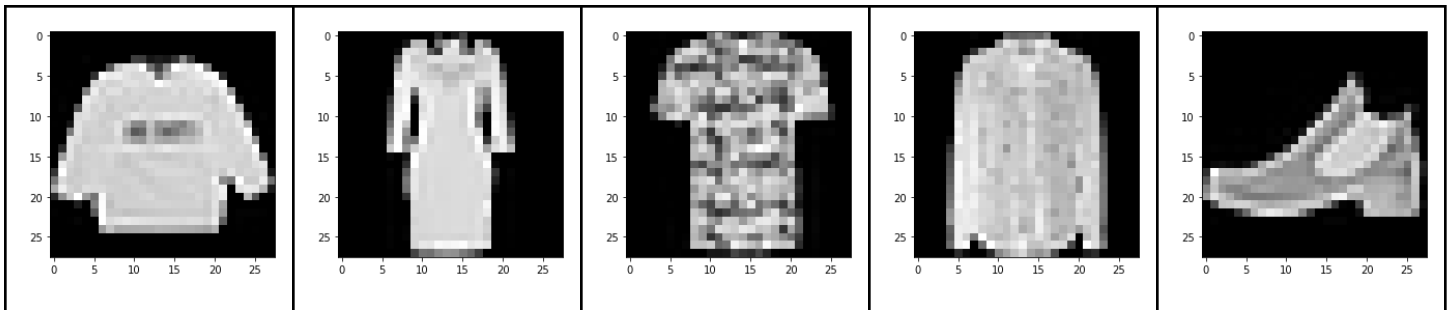
The objective of this project is to use extracted pixels from images, each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255.

In this project, we'll try 3 models one of them is LeNet-5 and we'll use transfer learning by using other two ones which are ResNet Architecture and Vgg-16. We will use these three models to classify mnist dataset to 10 classes.

The project includes final trial of best fitting hyper parameters of each one of them—, optimizers, learning rate and number of neurons in each layer to be able to reach the optimal solution that will best classify our data.

Dataset:

The dataset consists of fashion-mnist. Fashion Mnist is a dataset of Zalando's 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. Zalando seeks to replace the original Mnist dataset.



Data fields:

1. id - an anonymous id unique to an image.
2. Inputs: Pixels from 1 to 785.

3.

label	classification
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Methods:

Firstly, we check if there are missing values, duplicate values and drop it , then we make normalization to change range of pixels from 0 to 1 (0-255) .In addition to , encode the labels into category. Reshape data from (28,28) and 1 channel to (32,32) with 3 channels. To be sure that the LeNet-5 model can perform well on unseen data, we use cross validation = 5.After that, we made function contains the whole model with all hyperparameters and the evaluation of the model. This function contains the following hyperparameters:

1. Batch size
2. Kernel Size
3. Hidden nodes
4. Learning rate
5. Epochs
6. L2-regularisation
7. Dropout

We set The EarlyStopping callback to stop training when a monitored metric has stopped improving. . Also, we used a learning rate scheduler to change the learning rate for each epoch. Then we plot a graph for each model to represent the difference between training loss and validation loss, training accuracy, and validation accuracy.

Discussion:

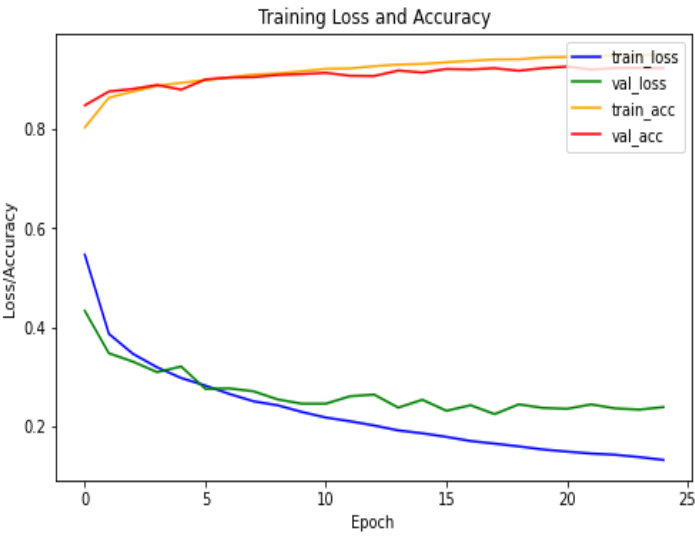
1- Let me talk about **LeNet-5** for fashion mnist dataset were digitized in a 28*28-pixel input image. The architecture is very small and simple compared to other architectures. The image set consisted of images of numbers from 0–9 in black and white.

-The LeNet-5 architecture utilises two significant types of layer construct: convolutional layers and subsampling layers.

- Convolutional layers
- Sub-sampling layers

-The network has 5 layers with learnable parameters and hence named Lenet-5. It has three sets of convolution layers with a combination of max pooling. After the convolution and average pooling layers, we have two fully connected layers. At last, a Softmax classifier which classifies the images into respective class.

The LeNet-5 architecture has the ability to process high resolution images, and is better only with higher resolution images, and this is a limitation in itself.

Graph	Accuracy
	<p>training datasets accuracy is 95.28%</p> <p>validation datasets accuracy is 92.42%</p> <p>with cross validation=5</p> <p>test datasets accuracy is 88%</p>

-Transfer learning

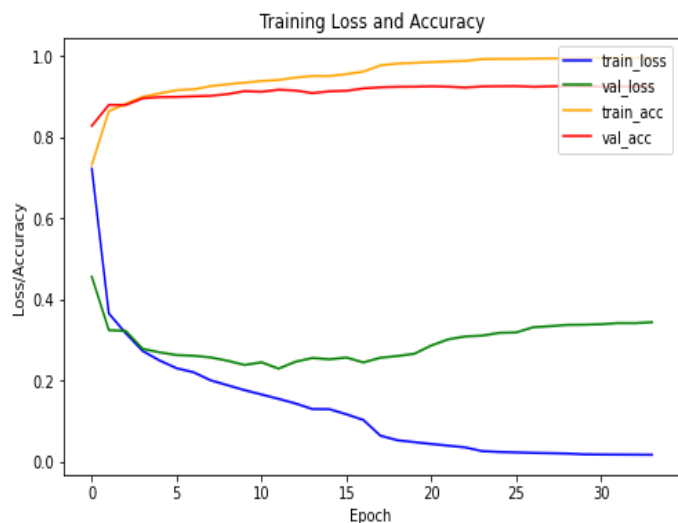
Transfer learning is the method that uses a neural network trained on a large and generalized enough dataset and is being used for another problem. These neural networks are called Pre-trained networks. Using a pre-trained model for feature extraction: When working with a small dataset, it is a common practice to take advantage of features learned by a model trained on a larger dataset in the same domain. This is done by instantiating the pre-trained model and adding a fully-connected classifier on top. The pre-trained model is "frozen" and only the weights of the classifier get updated during training. In this case, the convolutional base extracted all the features associated with each image and you just trained a classifier that determines the image class given that set of extracted features.

1. VGG-16

VGG16 is a simple and widely used Convolutional Neural Network (CNN) Architecture used for ImageNet. During training, the input 32*32 RGB image. It consists of 16 convolutional layers. It follows this arrangement of convolution and max pool layers consistently throughout the whole architecture. In the end it has 2 FC (fully connected layers) followed by a softmax for output. The VGG also has only 3x3 convolutions.

VGG makes use of lots of filters

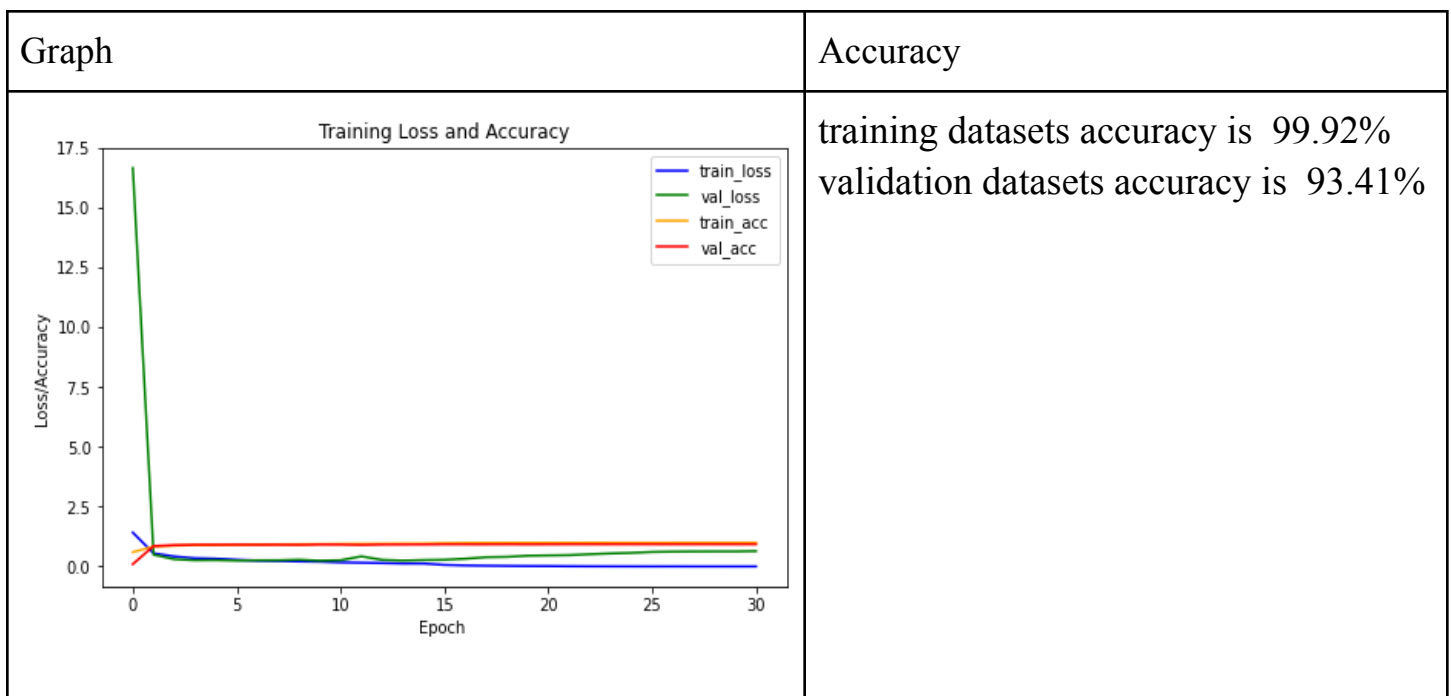
We add a flatten layer and two dense layers. This will also require the use of a softmax activation function. Between the feature extractor and the output layer, we can add a dense layer to interpret the features, in this case with 100 nodes.

Graph	Accuracy																																								
 <p>The graph, titled 'Training Loss and Accuracy', plots four metrics over 30 epochs. The y-axis is labeled 'Loss/Accuracy' and ranges from 0.0 to 1.0. The x-axis is labeled 'Epoch' and ranges from 0 to 30. The legend indicates: train_loss (blue line), val_loss (green line), train_acc (yellow line), and val_acc (red line). train_loss starts at ~0.75 and drops to ~0.02. val_loss starts at ~0.45 and drops to ~0.35. train_acc starts at ~0.75 and rises to ~0.99. val_acc starts at ~0.85 and rises to ~0.93.</p> <table><tr><th>Epoch</th><th>train_loss</th><th>val_loss</th><th>train_acc</th><th>val_acc</th></tr><tr><td>0</td><td>0.75</td><td>0.45</td><td>0.75</td><td>0.85</td></tr><tr><td>5</td><td>0.25</td><td>0.28</td><td>0.90</td><td>0.90</td></tr><tr><td>10</td><td>0.18</td><td>0.25</td><td>0.92</td><td>0.91</td></tr><tr><td>15</td><td>0.12</td><td>0.26</td><td>0.94</td><td>0.91</td></tr><tr><td>20</td><td>0.08</td><td>0.30</td><td>0.96</td><td>0.92</td></tr><tr><td>25</td><td>0.05</td><td>0.33</td><td>0.98</td><td>0.92</td></tr><tr><td>30</td><td>0.02</td><td>0.35</td><td>0.99</td><td>0.93</td></tr></table>	Epoch	train_loss	val_loss	train_acc	val_acc	0	0.75	0.45	0.75	0.85	5	0.25	0.28	0.90	0.90	10	0.18	0.25	0.92	0.91	15	0.12	0.26	0.94	0.91	20	0.08	0.30	0.96	0.92	25	0.05	0.33	0.98	0.92	30	0.02	0.35	0.99	0.93	<p>training datasets accuracy is 99.59%</p> <p>validation datasets accuracy is 93%</p>
Epoch	train_loss	val_loss	train_acc	val_acc																																					
0	0.75	0.45	0.75	0.85																																					
5	0.25	0.28	0.90	0.90																																					
10	0.18	0.25	0.92	0.91																																					
15	0.12	0.26	0.94	0.91																																					
20	0.08	0.30	0.96	0.92																																					
25	0.05	0.33	0.98	0.92																																					
30	0.02	0.35	0.99	0.93																																					

-The plots for the models show that there is little over-fitting

2. ResNet50

ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pre trained version of the network trained on more than a million images from the ImageNet database. ResNet improves the efficiency of deep neural networks with more neural layers while minimizing the percentage of errors. In other words, the skip connections add the outputs from previous layers to the outputs of stacked layers, making it possible to train much deeper networks than previously possible. Keras allows you to easily generate a detailed summary of the network architecture you built. This can be saved or printed for future use.



-As we show, LeNet-5 was not going to give me any better results without exponentially increasing my training time and working with high resolution images. You will have to do plenty of calculations and experiments to build a proper CNN architecture. So this technique is constrained by the availability of computing resources. It does not do as well with colour images. Most image recognition problems would require RGB images for better recognition. Since the model isn't very deep, it tries to scan for all features thus producing poor performing models. If the Neural Network isn't fed with enough features from the training images then it would be difficult for the model to generalize and create an accurate model. So I used ImageNet thus, the pre-trained models have been trained to work on a lot of different things. So we used pre trained models to see how they perform on this dataset and whether I would be able to increase my training accuracy.

Conclusion

As we show ,ResNet with the high number of layers did show better performance, as compared to LeNet-5, and VGG-16.

In the case of LeNet, as epoch is increased,the validation accuracy increases. The same trend is noticed in the case of VGG and ResNet, but in ResNet the validation accuracy rates are high. so the ResNet can be chosen and then it can be trained with real world images.For almost all the models, a better accuracy is obtained in the testing set if the training epoch is set higher.

Loss calculates the difference between the output and the target variable. It measures the accuracy of the model during training and we want to minimize this function. In this project, we choose the `sparse_categorical_crossentropy` loss function. Cross-entropy is the default loss function to use for a multi-class classification problem and it's sparse because our targets are not one-hot encodings but are integers.

Optimizer how the model is updated and is based on the data and the loss function. Adam is an extension to the classic stochastic gradient descent and is popular because it's shown to be effective and efficient.