Computer Vision
Assignment 13

Alireza Moradi

December 18, 2020

1

1.1

Number of parameters in a CNN is independent of the size of input data(except for the number of channels). So the number of parameters is:

$$16 * 9 * 9 * 10 + 16 = 12976$$

Also we need padding of size 4 because a 9×9 filter goes at most 4 pixels beyond the borders of the image.

1.2

Input image dims = $32 \times 32 \times 3$ and we have:

- F = 5
- S = 1
- P = 0

According to the below formula:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

And the fact that we have 3 filters, the new dimensions will be $28 \times 28 \times 3$.
Similarly if we apply three 3×3 filters to the same image twice, the dimensions will be $28 \times 28 \times 3$.

1.3

- `Epoch`: One Epoch is when an entire dataset is passed forward and backward through the neural network only once. Passing the entire dataset through a neural network is not enough. And we need to pass the full dataset multiple times to the same neural network. As the number of epochs increases, more number of times the weight are changed in the neural network and the it goes from underfitting to optimal to overfitting.

- `Learning Rate`: The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.
- `Batch Size`: The number of training examples in one forward/backward pass. The higher the batch size, the more memory space we'll need. Neural networks are trained using the stochastic gradient descent optimization algorithm. This involves using the current state of the model to make a prediction, comparing the prediction to the expected values, and using the difference as an estimate of the error gradient. This error gradient is then used to update the model weights and the process is repeated. In other words, The number of training examples used in the estimate of the error gradient is called batch size. The batch size impacts how quickly a model learns and the stability of the learning process. Smaller batch sizes are used for two main reasons:
  - Smaller batch sizes are noisy, offering a regularizing effect and lower generalization error.
  - Smaller batch sizes make it easier to fit one batch worth of training data in memory.

Source

1.4

- Because convolutional layers are not fully-connected, The parameters of the network will be much less, resulting in faster training and less chance of overfitting.
- Another advantage is that the number of parameters on a CNN is independent from the input size.

1.5

- Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

Source

1.6

Suppose we have an input image of size 32×32. If we apply a 7×7 filter to this image, we have $26 \times 26 \times 49 = 33124$ (plus a few summations which won't change our results here) computations, while applying a 3×3 filter three times, will have $30 \times 30 \times 9 + 28 \times 28 \times 9 + 26 \times 26 \times 9 = 21240$ (plus a few summations which won't change our results here) computations.
Note that the input size should be big enough for the our assumption to hold.
Also because the output dimensions will be the same (26×26 in case of 32×32 input size), we can see that for the same output size we can save some resources by using multiple smaller filters instead of a large one.

2 Implementations

Fashion MNIST data set contains 60000 training and 10000 test samples. I splitted 20% of the training data (12000 samples) and used that as the validation data.

I used colab's TPU so there are some initialization blocks for that matter.

In the model, First there is a batch normalization layer, Adding this layer boosted the accuracy a little. Then there are 2 inception modules. I also added dropout with 25% probability after each inception module, because given the architecture and kind of data, The network could easily overfit the data and dropout helps preventing that. Finally, For the classification, There are 2 fully-connected layers, one with 128 neurons and the final layer (output) has 10 neurons with softmax activation function. categorical cross entropy was used as loss function.

This model achieved 90.89% accuracy and 90.84 F1-score on the test data with 20 epochs training. I was also able to achieve 93% accuracy on the test data using data augmentation and a different model (code for this model is in my file, it's commented out).

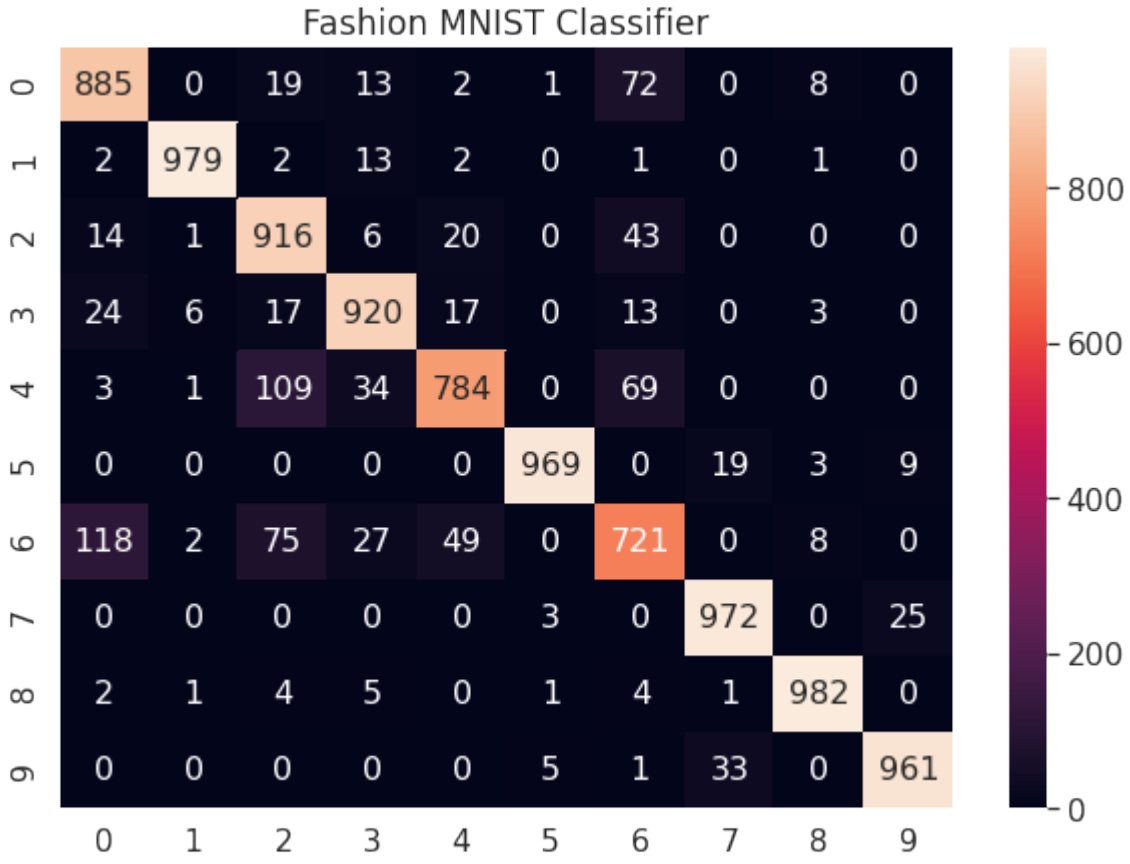

Figure 1: Confusion Matrix

Figure 2: Bad Loss Images

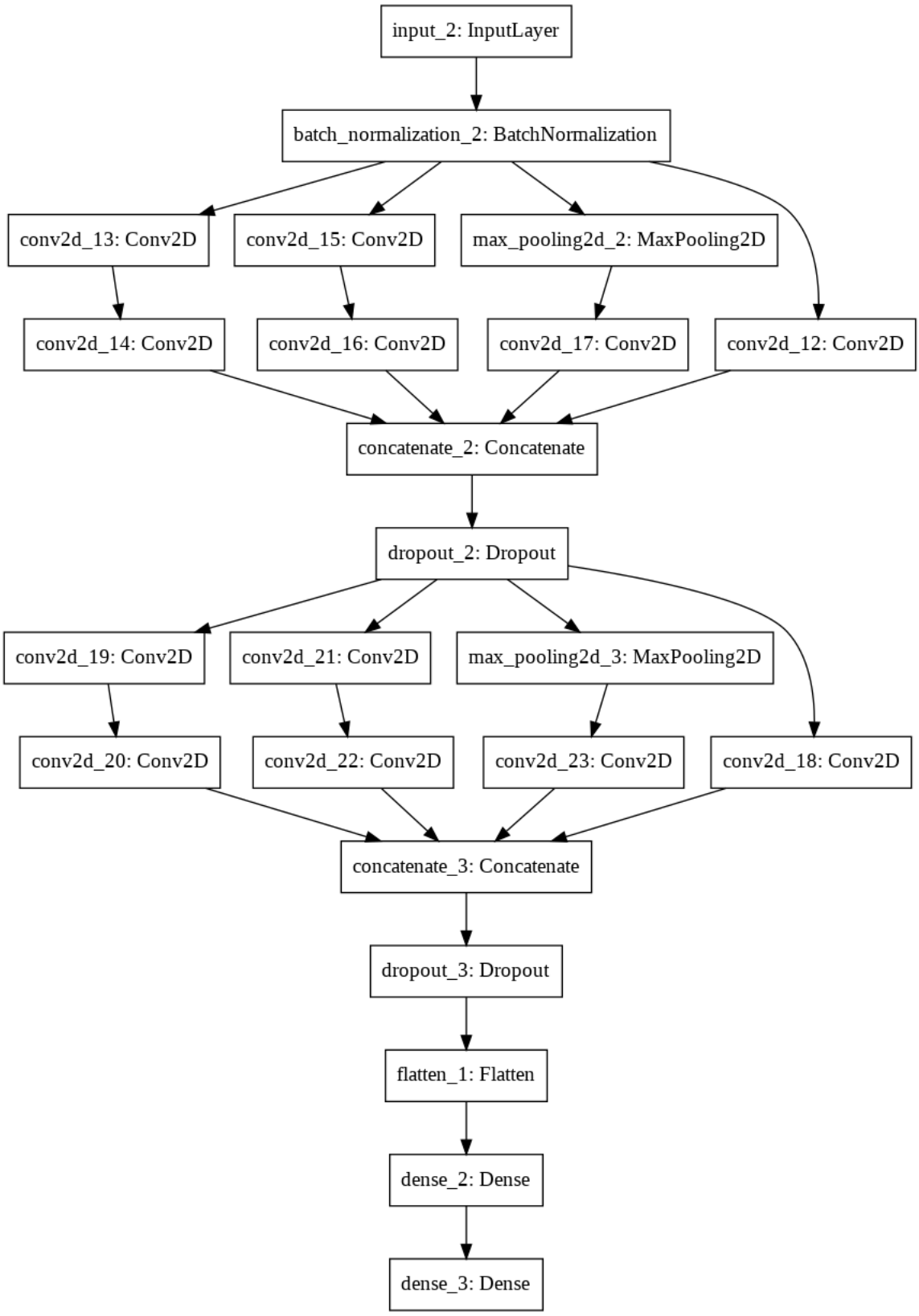

Figure 3: Model