Assignment -4

Experiment -1 (A basic CNN)

Complete implementation along with metrics can be found at – assignment4.ipynb

A architecture with Convolution layers and pooling layers followed by filly connected layers is considered for this experiment .

Code-

A screen shot of a computer program

Description automatically generated

Instead of implementing all the methods from scratch, BaselineModel is inherited as parent class and necessary methods are overridden.

No of trainable parameters – 2.1 M

Maxepochs is not set but early stopping callback is used with patience 5. Following metrics are obtained on validation and test set-

Best epoch – Epoch 4

Best validation loss 1.31

Test loss: 1.35

Test Accuracy: 57%

Experiment -2 (AllConvNet)

Paper on all conv net discusses three different architectures out of which we tried implementing model C.

Code –



Forward pass-

A screen shot of a computer program

Description automatically generated

No of trainable parameters – 1.4 M

Following metrics are obtained on validation and test set-

Best epoch – Epoch 34

Best validation loss 1.96

Test loss: 1.98

Test Accuracy: 47%

Comparison of basic CNN vs AllConvNet – While no of parameters for AllConvNet is significantly less than basic CNN , the accuracy for basic CNN is higher than AllConvNet in this case.

Experiment -3 (Basic CNN + Regularization)

We chose basic CNN from the above both as it gave better performance compared to AllConvNet. Now we can either add dropouts to our network or modify the transform pipeline to include data augmentation as part of training data. Here we choose to add dropouts for simplicity. Following is the updated architecture after adding dropout –

A computer screen with text and images

Description automatically generated

The remaining code is same as basic CNN (experiment -1)

Comparison –

|  |  |  |
| --- | --- | --- |
|  | **Basic CNN** | **CNN with Dropout** |
| **Epochs trained** | **34** | **9** |
| **No of params** | **2.1 M** | **2.1 M** |
| **Validation Loss** | **1.31** | **1.32** |
| **Test loss** | **1.35** | **1.39** |
| **Test Accuracy** | **57%** | **54%** |

We can observe that the model with dropout included converged must faster compared to model with no dropout. Although the accuracy is slightly on the lower side.

Experiment -4 (Cifar-10 +Transfer learning on CNN + Dropout model)

The same transformation steps as Imagenette are performed for CIFAR-10 downloaded dataset.

A computer screen shot of text

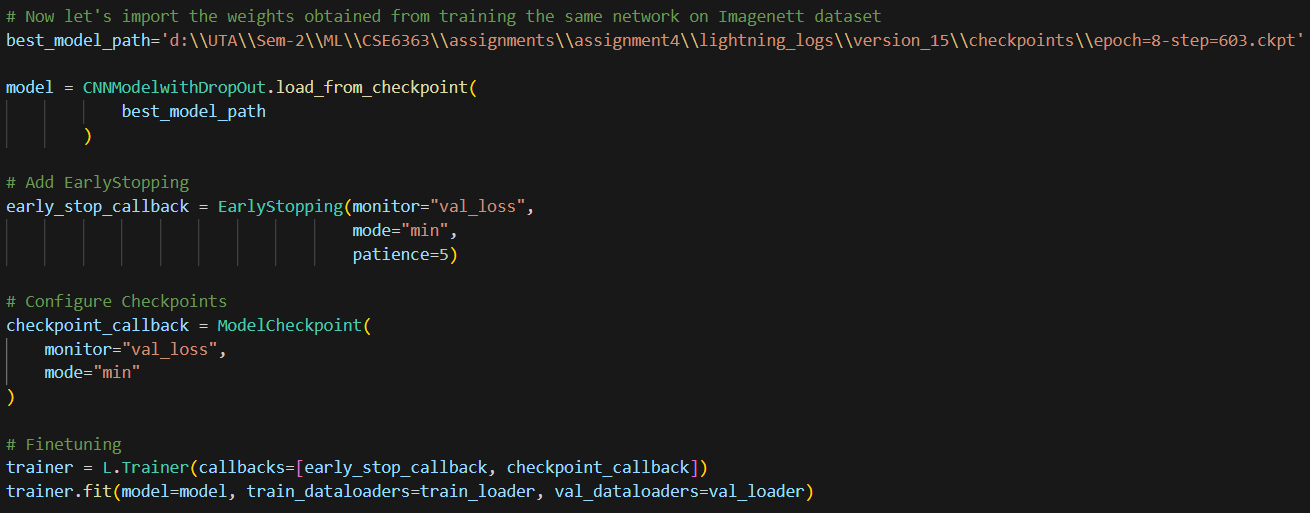
Description automatically generated

Now, a model from scratch is trained using CNN with dropout architecture.

A screen shot of a computer program

Description automatically generated

For second part of experiment, to implement transferlearning, a previously trained CNN with dropout architecture model is imported and trained on cifar -10 dataset as follows –



Results obtained –

|  |  |  |
| --- | --- | --- |
|  | **Cifar-10 using CNN with Dropout** | **Cifar-10 using Imagenett pretrained CNN with Dropout** |
| **Epochs trained** | **64** | **19** |
| **No of params** | **2.1 M** | **2.1 M** |
| **Validation Loss** | **1.17** | **1.31** |
| **Test loss** | **1.17** | **1.30** |
| **Test Accuracy** | **58%** | **53%** |

Conclusion – pretrained model although has slightly less accuracy compared to model that was trained from scratch, it converged a lot quicker and has descent accuracy.