CSCE 636: Deep Learning
Fall 2021
Final Project Submission

Rohan Singh Wilkho
PhD Student - Civil Engineering
MS Student - Computer Science

November 22, 2021

Abstract

This report is in part requirement of the Final Project Submission for the CSCE 636 (Deep Learning) course in Fall 2021. This report contains details on the base neural network model architecture used for the assignment, innovations made to the basic architecture, training and testing results on the public dataset.

1 Basic Network Model

Convolutional Neural Networks (CNNs) are widely used for the objective of image classification. Over the years, there have been many CNN models introduced with state-of-the-art results on image classification tasks. They include ResNet [1], Squeeze Net [2], DenseNet [3], among other. The basic neural network architecture considered for this assignment was DenseNet.

1.1 DenseNet

A DenseNet is a type of convolutional neural network that utilises dense connections between layers, through Dense Blocks, which connects all layers (with matching feature-map sizes) directly with each other.

This is done to preserve the feed-forward nature, each layer obtaining additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers. This also aids in managing the issue of vanishing gradients, enabling the network to have deeper layers, strengthen feature propagation, encourage feature reuse and reduce number of parameters.

The basic architectures are shown in Figures 1 and 2.

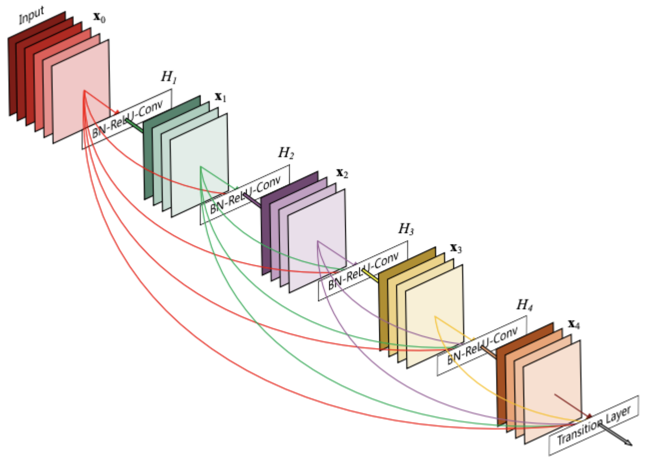

Figure 1: A 5-layer DenseNet Block

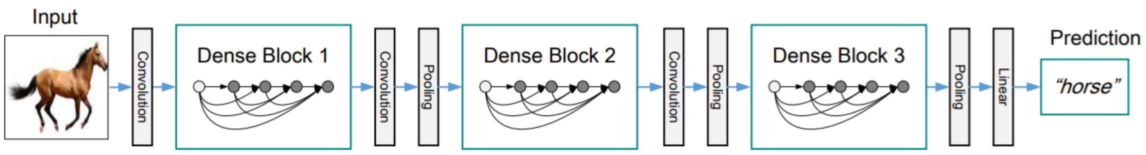

Figure 2: A deep DenseNet with 3-Dense Block

2 Innovations in this project

For the purpose of this project, the DenseNet model wasn't directly applied. This project involved the following innovations:

- Implementing Data Augmentation: For the training images, data augmentation was performed, where each image was flipped (using np.fliplr) and mirrored (using np.flipud).
- Weight initialization: Instead of relying on pytorch, the model weigths were initialized for the convolution (using He initialization [4]) and batch normalization layers (using constant weights of 1 and 0 for bias).
- Bottleneck layer: Instead of having a normal BN-ReLU-Conv3D layer, the implemented model had a BN-ReLU-Conv1D-BN-ReLU-Conv3D layer, which reduced the number of parameters.
- Learning Rate Adjustment: Instead of following the learning rate adjustment as mentioned in the original paper, this model adjusted learning rate (divided by 10) for every 100 epochs.

3 Data Preparation

In this section, I will describe the data preparation done for the image classification task.

The following steps were followed to prepare the data for training, validation and testing.

- The image first normalized (by dividing all the pixel values by 255)
- Then, the array of shape 3072 was transformed in a multi-dimension array of shape 3*32*32
- Then, for each channel of the image (red, green and blue), the pixel values were standardized by subtracting the mean pixel value and dividing the difference with the standard deviation of the pixel values for that channel.

In addition to the above steps, data augmentation was also performed, only for the training data. The following data augmentation was performed.

- The original image.
- The flipped original image.
- The mirrored original image.

  This resulted in tripling the size of training data, which helped in making the model more generalizable.

Table 1: Comparison of Validation Accuracy for different model variants

Model	Validation Accuracy
DenseNet-LR1	0.8404
DenseNet-LR2	0.8479
DenseNet-Bottleneck-LR1	**0.8789**

4 Training and Validation

This section contains the training and validation steps followed to come up with the most suitable model. I had compared the basic DenseNet and DenseNet Bottleneck models. Among these two models, I had compared the varibale learning rate schedulers as well. Table 1 compares the validation accuracy of the different variants considered for this project.

In Table 1, DenseNet means the original DenseNet model being applied. DenseNet-Bottleneck means the normal Convolution layer is replaced by a Bottleneck Convolution Layer. LR1 and LR2 stand for different learning rate schedulers, where LR1 implements the scheduler as described in the original DenseNet paper, and LR2 divides the learning rate by 100 for every 100 epochs.

Among the models tested, DenseNet-Bottleneck-LR1 gave the best validation accuracy. So, it was chosen for final training.

Please note that the submission code has files for both creating Networks of both DenseNet and DenseNet-Bottleneck. The network for DenseNet and DenseNet-Bottleneck are stored in 'Network-Dense.py' and 'Network.py' codes, respectively.

5 Final training and testing

After the final training, the testing accuracy obtained was **0.8714**. The final model was saved in the folder **'saved-models'**.

6 Testing on Private Dataset

The private testing dataset was saved in the same directory path as the other python files. The predictions are also saved in the same folder under the name 'Predictions.npy'

References

[1] He, K., Zhang, X., Ren, S., Sun, J. (2015). Deep Residual Learning for Image Recognition.

[2] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size.

[3] Huang, G., Liu, Z., van der Maaten, L., Weinberger, K. Q. (2018). Densely Connected Convolutional Networks.

[4] He, K., Zhang, X., Ren, S., Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. Proceedings of the IEEE International Conference on Computer Vision, 2015 International Conference on Computer Vision, ICCV 2015, 1026–1034. https://doi.org/10.1109/ICCV.2015.123