Image Classification – ELEC 375 Lab 3 Report

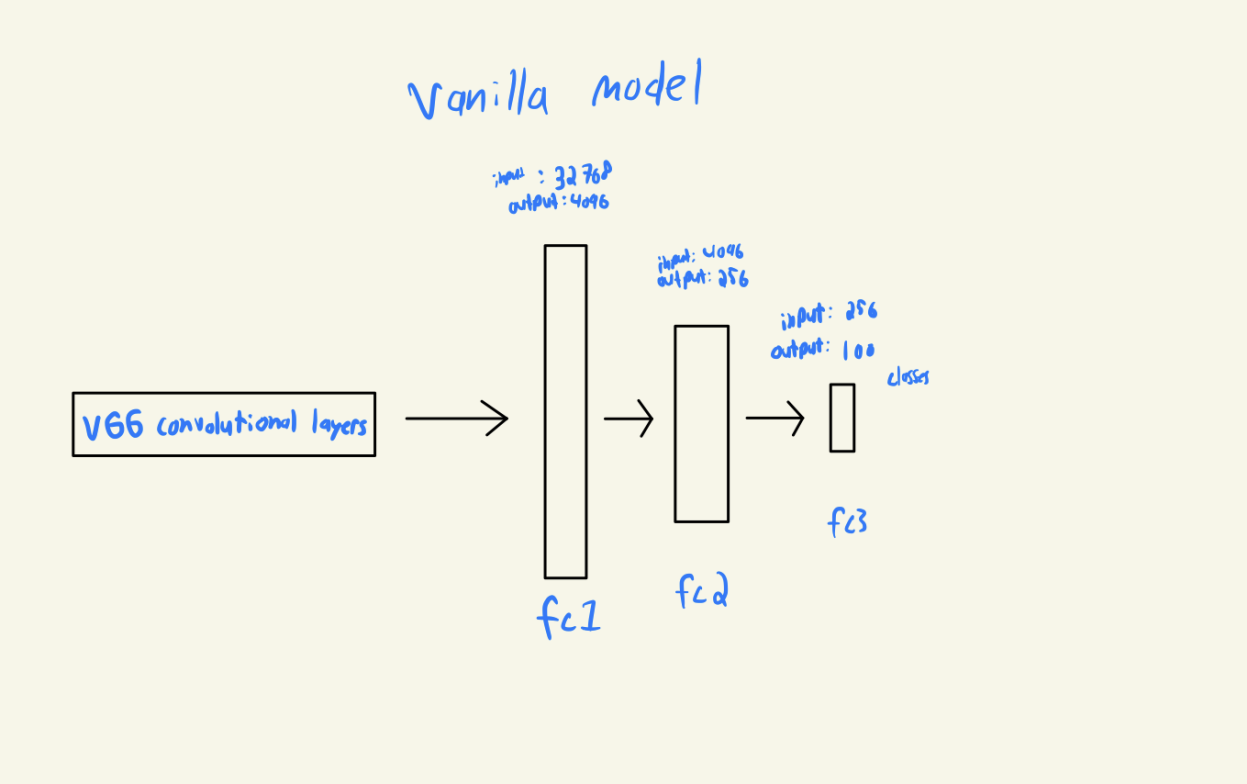
Noah Serhan

Section 1 – Vanilla Architecture

My vanilla model begins by implementing the VGG backend provided in lab 2 to encode the 32 x32 x3 images from the CIFAR100 dataset into a bottleneck tensor. This was done in a separate neural network class called VGG.

The next part in my vanilla model file was the encoder_decoder class. This class serves as a container for the architectural components used in the Vanilla Model. It consists of two main parts: the VGG-based backend and the frontend. It was made to be the vanilla version of the decoder frontend to the provided VGG backend. This is responsible for feature extraction from the input images; it applies several convolutional layers with ReLU activations and max pooling to transform the input image into a set of high-level features. The final convolutional layer with 512 channels represents the bottleneck of the VGG architecture, and the extracted features are then passed to the frontend for further processing. The frontend, also defined within the encoder_decoder class, is a series of fully connected layers. It includes multiple linear layers, dropout, batch normalization, and ReLU activations.

The purpose of the frontend is to take the high-level features generated by the VGG-based backend and further transform them to make class predictions. The frontend includes dropout layers for regularization, batch normalization for training, and ReLU activations for non-linearity. The final linear layer in the frontend produces class predictions, aligning with the number of class labels in the CIFAR-100 dataset.



Section 2 – Mod Architecture

The Mod Model is designed to improve the performance of image classification on the CIFAR-100 dataset, which was proven to be necessary after seeing the poor performance of the vanilla model with an unseen test set of CIFAR100 data.

The Mod Model's architecture incorporates the ResNet building blocks. These are designed to address the vanishing gradient problem; ResNet introduces skip connections that allow gradients to flow more effectively during training. This section includes convolutional layers, batch normalization, and skip connections. The motivation behind incorporating ResNet blocks is to improve my model's ability to learn complex representations, making it well-suited for a task like image classification.

The Model class is responsible for defining the overall architecture of the Mod Model. It initializes with the choice of ResNet, the configuration of layers for each stage, and the number of output classes. The layers are stacked to form a deep neural network, including multiple stages with varying numbers of ResNet blocks. The motivation behind this architectural modification is to create a deeper model that can capture complex features in the data. Ultimately, stacking ResNet blocks enables it to make more accurate predictions on the CIFAR-100 dataset.

Section 3 – Experiments

Training

The training was done using the CIFAR-100 dataset with images resized to 64x64 with a padding of 4. The vanilla network utilizes a VGG backend for feature extraction and has three fully connected layers for classification. The modified network, on the other hand, still uses the VGG backend, but it introduces residual blocks within the fully connected layers, taking a ResNet-like approach.

Hyper-parameters:

- The optimization method used for this training is “Adam”, which is an adaptive learning rate optimization algorithm with efficient convergence properties that combines the advantages of both the AdaGrad and RMSProp gradient decent methods.
- The optimizer is initialized with a learning rate of 0.001 and has a weight decay of 0.0002.
- The batch size can be specified by the user, however, a batch size of 100 was used for both training and testing of the vanilla and modified networks.
- Like the batch size, the number of epochs can also be changed based on the user’s inputs, however, there was no need to go past 20 as the loss plots would seem to plateau around or before that point.
- The scheduler used was the StepLR, which adjusts the learning rate by decaying it by a factor of gamma every set number of epochs. In this case, it was reduced every 5 epochs by a factor of 0.1.

Hardware

The hardware used for training was a GTX 1080 graphics card, which made use of NVIDIA's CUDA capabilities to allow for faster training times. If CUDA is not available however, a CPU can be used to train and test the models, with the consequence of it taking more time to do so.

Time taken

The training of both the vanilla and the modified model took just over 20 minutes, equating to around 1:02 minutes for each individual epoch.

Training plots

The vanilla network's loss starts very high, above 40, which suggests that the initial model predictions were poor. There is a sharp decrease at the second epoch that indicates rapid learning. There is then a slight spike that follows which can be indicative of a tougher batch or just some instability in the training. The plot then stabilizes and flattens almost completely by the time it reaches 20 epochs.

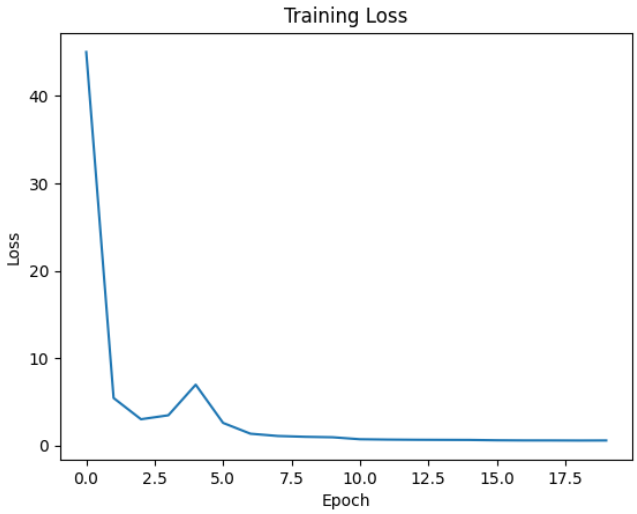


Figure 1: Vanilla network training loss

The modified network's initial loss is just above 3, which is much lower than the vanilla one. It consistently decreases throughout the process and also stabilizes as it approaches the 20 epoch mark, indicating proper model training assuming no overfitting.

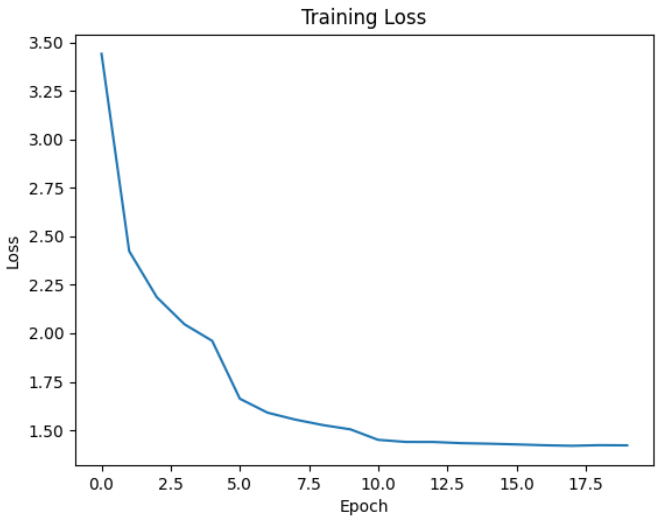


Figure 2: Modified network training loss

Section 4 – Discussion

Expectations:

The performance of the Mod Model was as expected. Incorporating ResNet blocks proved to be a successful architectural modification for enhancing the model's performance in image classification. It allowed the model to train effectively, even when deep architectures are involved.

Improvements:

The Mod Model demonstrated clear improvements over the Vanilla Model in terms of accuracy. This can be explained by the architectural modifications enabling the model to learn hierarchical features, making it more adept at handling the complexity of the CIFAR-100 dataset.

Trade-offs: Efficiency vs Accuracy

One evident trade-off with the Mod Model is the significantly increased training time due to the deeper architecture. However, the gain in accuracy justifies this trade-off.

The Mod Model's primary aim was to improve accuracy, and it succeeded in this regard. The architectural changes in the ResNet blocks were effective in enhancing the model's ability to capture fine-grained details in the images. In conclusion, the architectural modifications introduced in the Mod Model resulted in a significant improvement in accuracy on the CIFAR-100 dataset. These improvements demonstrate the importance of carefully designed architectures in achieving high-performance results in image classification tasks.