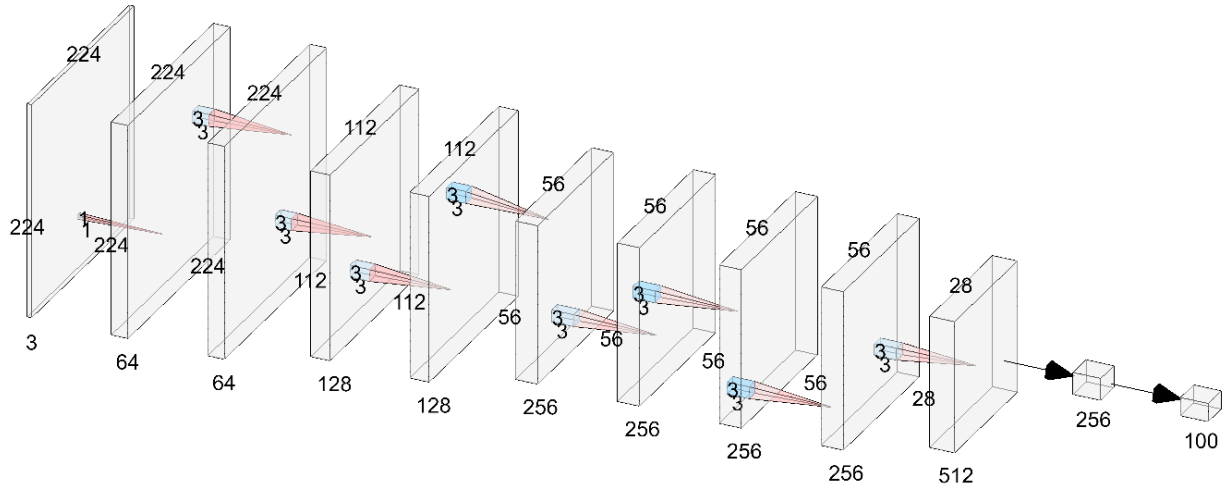
ELEC 475 Lab 3: Image Classification

Jordan Hunter – 20178182,

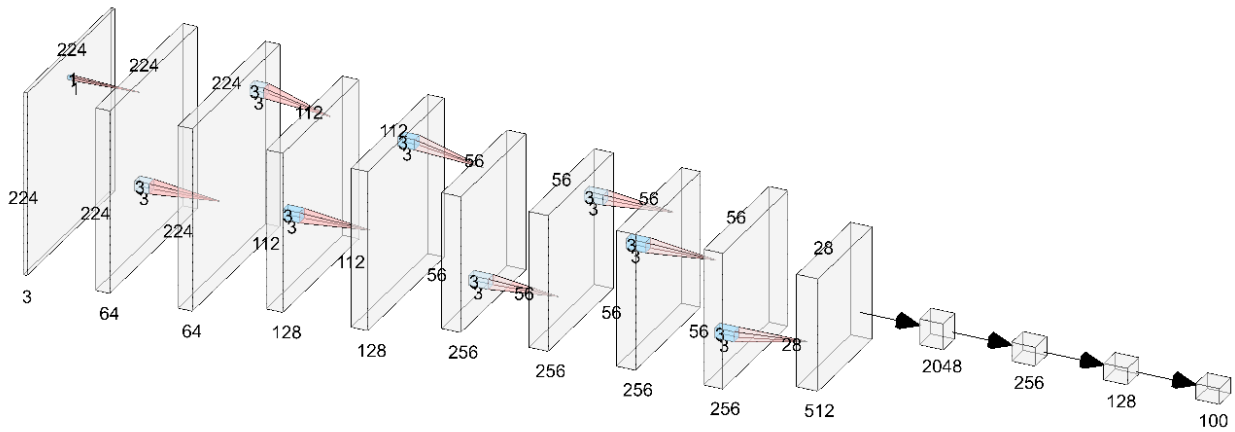
Nicholas Krsikapa – 20102039

Section 1 – Vanilla Architecture:



The vanilla model is built using the predefined Vgg encoder provided by Dr. Greenspan. This encoder uses 10 layers of convolutions and has been already trained using the ImageNet dataset. The frontend decoder portion consists of 2 fully connected layers one going to 256 output and the last splitting the data into 100 categories for classification. Between the layers, a Relu operation is done and after the last layer, a Softmax operation was applied. The reason the team used this model was because with other models that contained more layers with larger sizes, took the model longer to compute with little advancement in loss or accuracy.

Section 2 – Mod Architecture (Mod model):



The mod model is like that of the vanilla model. While the backend encoder remains the same, the frontend decoder utilizes a dropout of 30% probability before running 4 fully connected layers used in image classification. The dropout method was used as a method to optimize performance, by de-activating 30% of the neurons at random during training. Consequently, increasing the robustness of the model due to the requirement of the active neurons to correctly classify the input images based on the

learned features without aid from the deactivated neurons. Furthermore, through the addition of more fully connected layers the model had the opportunity to gain more non-linearity, and theoretically an opportunity to develop more in-depth features to better optimize the classification process.

Section 3 – Experiments:

To train the system on CIFAR100 data, the team used the following hyper parameters for both architectures:

- Epochs - 1000
- Gamma – 1.0
- Initial learning rate – 0.0003
- Optimizer – Adam
    - Weight_Decay – 0.0001
- Scheduler – ReduceLROnPlateau
    - Factor – 0.1
    - Min_lr – 0.001
    - Patience – 2
- Loss function – nn.functional.cross_entropy
- Batch size – 1024

During training, between every epoch, the models were evaluated for their accuracy against the training and testing dataset. Both systems were trained on a personal laptop that had a dedicated GPU which allowed the system to only take about 21 seconds each epoch. Each system took slightly over 6 hours to train where intermediate version of the model were saved to continue if anything failed.

The top 5 and top 1 error ratings for each model can be seen below; please note error were tested against training and test data.

Model	Error (Training Data) [%]		Error (Testing Data) [%]	
	Top-5	Top-1	Top-5	Top-1
Vanilla Model	49.34	49.76	55.38	67.82
Mod Model	43.29	43.59	52.64	67.19

The following two images show a graph of the losses recorded for each model after every epoch. The final losses for the vanilla and mod model are 4.119 and 4.063 respectively.

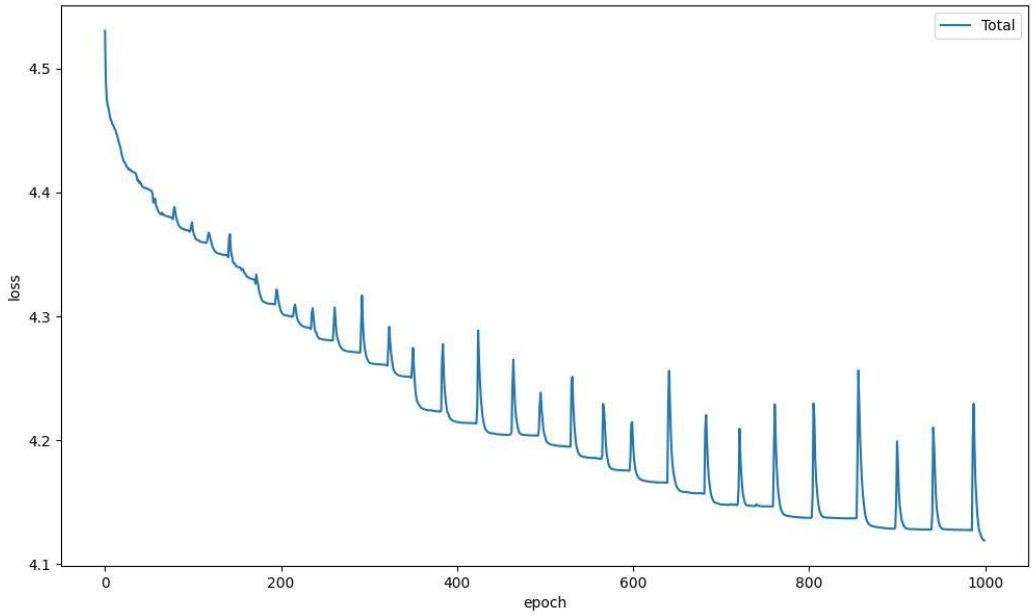


Figure 1: loss plot of vanilla model

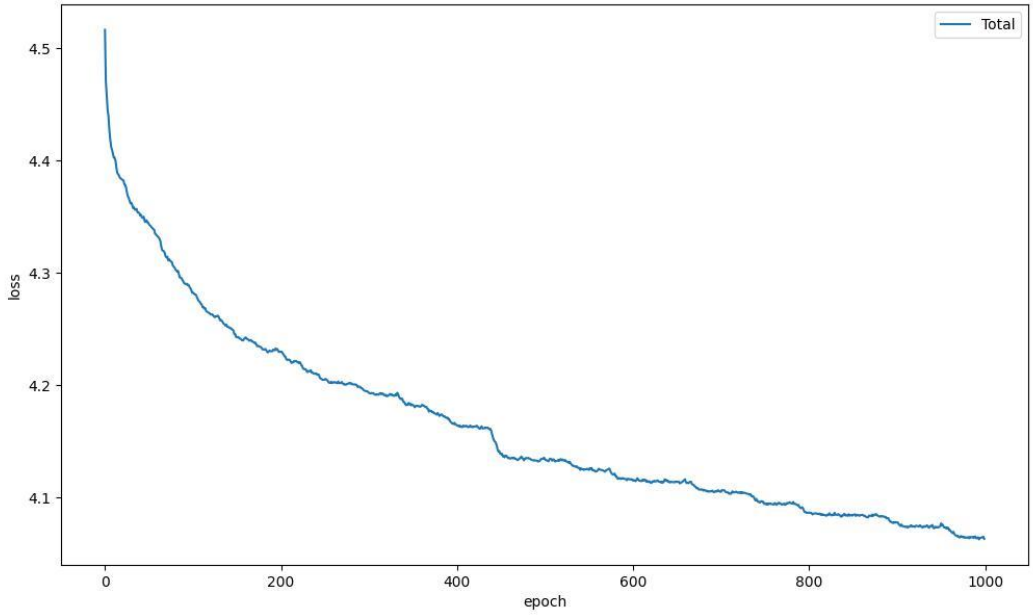


Figure 2: Loss plot of modded model

Section 4 – Discussion:

The mod improved the overall loss convergence of our model, with the loss of the vanilla and modded models being 4.11919 and 4.06336, respectively. Showing that the mods slightly increased optimization of the model. As expected, this can be seen in the difference of top-1 and top-5 error rates between the vanilla and modded models, with the modded model having lower error rates on both the training and test datasets. This is because the modded model has more robustness and non-linearity due to the implementation of the dropout method, in which 30% of the neurons shutdown at random, allowing for more variation in the model output. However, due to a decrease in the number of active neurons during any given batch run, as well as the 2 extra fully connected layers in the frontend, the training and test times were slightly increased for the modded model.

It can also be observed that the test dataset returned higher error rates than the training dataset. This is expected because the model was trained throughout multiple iterations, on significantly more images, than the testing set. As a result, the model had more opportunities to better learn training dataset as opposed to the testing dataset.

Furthermore, top-5 error rates are all lower than their respective top-1 error rates. This is expected because the top-5 error rate is a broader classification technique, where the correct classification has to be one of the top 5 most likely classifications. As opposed to the top-1 error, where the model output must result in the expected classification class to be considered correct.