0.1 Team Member Contributions

Name	Contribution
Moniek Smink (smink2@wisc.edu)	• 3.0 – 3.6: Training Deep Neural Networks
Venkata Saikiranpatnaik (vbalivada@wisc.edu)	
Sourav Suresh (sourav.suresh@wisc.edu)	

1.0 Overview

2.0 Understand Convolutions

3.0 Design and Training Deep Neural Networks

3.1.1 Training

Unless otherwise noted, training was done using stochastic gradient descent with a batch size of 256, momentum of 0.9, and starting learning rate of 0.1 decayed according to a linear warm-up, cosine annealing schedule. A weight decay warm-up of 5 epochs and weight decay parameter of 1e-4 was used for regularization. Cross entropy loss was used to compare the predictions made by the model to the ground truth labels. Data was augmented during training to improve the generalization potential of the model. Images were randomly cropped, flipped, color-manipulated, scaled, and rotated during training. During validation, images were randomly scaled and center cropped. The Top-1 and Top-5 accuracy was computed at each epoch. A Top-k accuracy marks a prediction as correct if the ground truth label is in the model's top k guesses. Both the Top-1 and Top-5 accuracies are displayed because the Top-1 accuracy can be misleading in cases where class distributions are unequal. Training was done on a Google Cloud Virtual Machine equipped with a Nvidia Tesla T4 GPU.

3.1.2 Data

Training and validation was done on the MiniPlaces dataset. A dataset created by MIT with 120k images and 100 mutually exclusive classes of scenes such as 'auditorium', 'highway', and 'playground'. The training set consisted of 100k images with 1k images per class. The validation set consisted of 10k images mixed classes. The training set was used to minimize the loss function with stochastic gradient descent during training. The validation set was used during training to monitor the generalization progress and calculate the Top-1 and Top-5 accuracies for each model at each training step.

3.2.1 SimpleNet

A simple convolutional model, called SimpleNet, was trained for 60 epochs with default parameters. SimpleNet is made up of 7 convolutional layers with ReLU activation and occasional max pooling layers, with a final average pool and dense layer. The Top-1, Top-5, learning rate schedule, and training loss graphs for training are displayed below. This model's Top-1 and Top-5 accuracy on the validation set was 44.880 and 73.630, respectively. During training, 3113 MiB of the GPU memory was used.

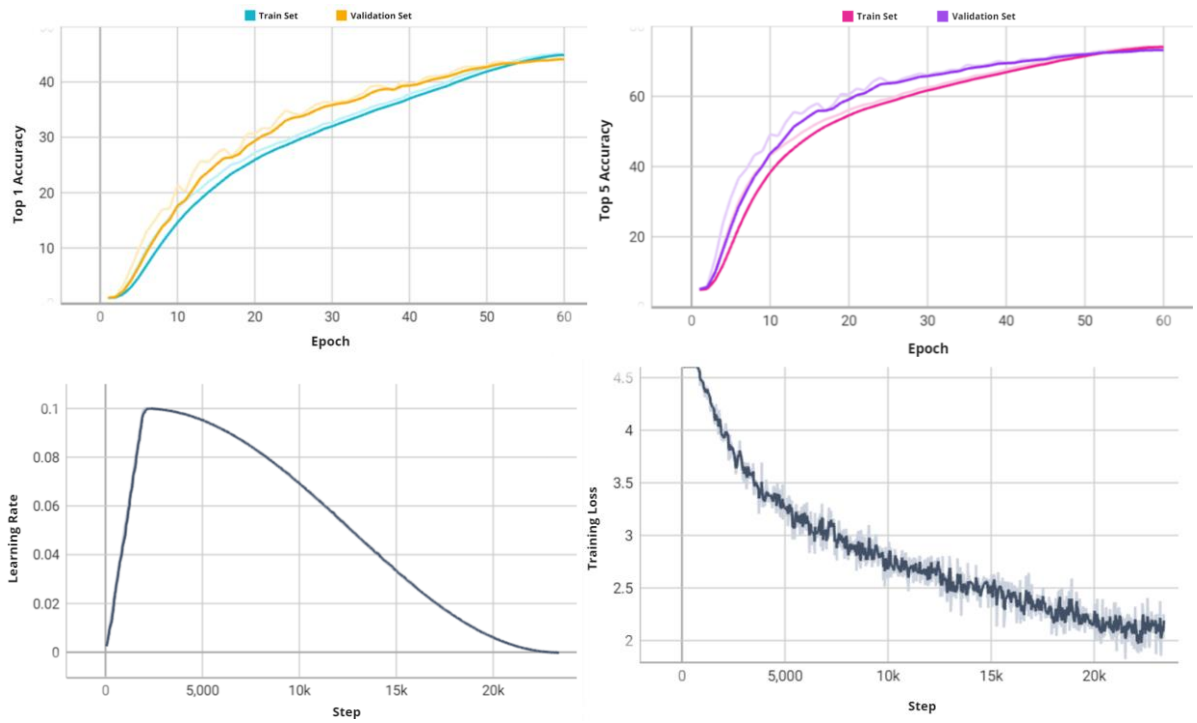

Figure 1: The Top-1 (top left), Top-5 (top right), learning rate schedule (bottom left), and training loss (bottom right) graphs for training SimpleNet.

3.2.2 SimpleNetCustConv2D

A SimpleNet with custom-made 2D convolution operation, called SimpleNetCustConv2D, was trained for 60 epochs with default parameters. The Top-1, Top-5, learning rate schedule, and training loss graphs for training are displayed below. This model's Top-1 and Top-5 accuracy on the validation set was 44.900 and 73.330, respectively. During training, 4881 MiB of the GPU memory was used.

We see that the accuracies for this model are very comparable to the original SimpleNet model with Pytorch's Conv2D operation, likewise, the rate of convergence seems to be very similar, as demonstrated in the training loss vs step graph shown below. The only discernible difference we can see is the GPU memory used. Using nvidia-smi, we see that the average amount of GPU memory used during the traditional SimpleNet training was 3113 MiB. Meanwhile, the average GPU memory used during the SimpleNet training with the custom Conv2D operation was 4881 MiB. This is likely explained by some memory efficiency optimization Pytorch does in its Conv2D operation such as reducing intermediate variables or efficiently handling biases.

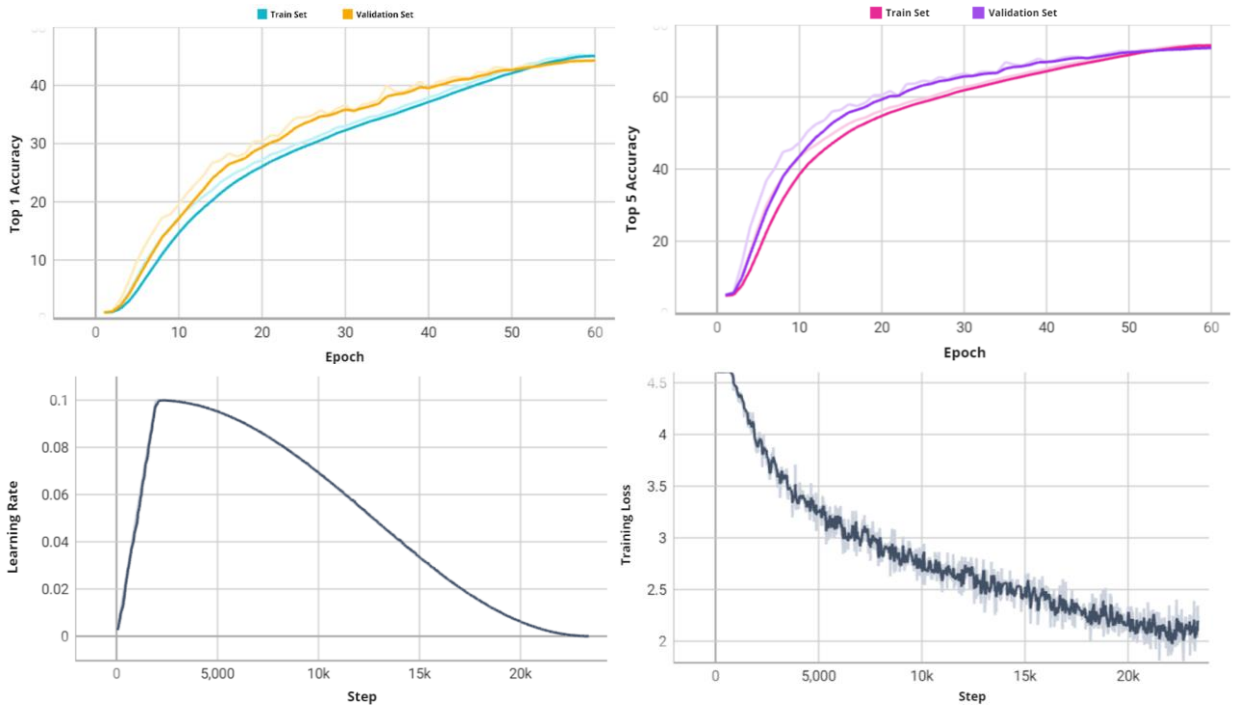

Figure 2: The Top-1 (top left), Top-5 (top right), learning rate schedule (bottom left), and training loss (bottom right) graphs for training SimpleNet with a custom Conv2D operation.

3.3 SimpleViT

A vision transformer model, called SimpleViT, was implemented. Transformers are commonly used on sequences of inputs like in NLP. A Vision Transformer classifies an image by separating an image into a series of patches, creating a patch embedding, adding a positional embedding, running this series into a transformer encoder, aggregating the patch-level outputs, and finally creating the class logits with an MLP head.

The patch embedding is done with a Conv2D layer where the kernel size is equivalent to the stride with a patch embedding dimension parameter. The positional embedding is initialized to the image size and is later learned through training. The transformer encoder is made up of a series of transformer blocks utilizing interleaving local and global self-attention, layer norms, and MLP heads as described in [1]. Self-attention layers transform an input sequence using representations of the sequence called queries, keys, and values. Global self-attention allows for long-distance interactions within the image. Local self-attention allows for smaller-distance interactions within an image but reduces a lot of the computational complexity that comes with global self-attention. Interleaving global and local self-attention allows the model to learn long- and short-distance relationships while still maintaining reasonable computational complexity. By default, the transformer encoder contains four transformer blocks with the first and third utilizing local self-attention. After the transformer encoder, in order to aggregate the patch-level outputs, global average pooling is done. Final normalization and dense layers create the final output logits.

SimpleViT was trained for 90 epochs using an AdamW optimizer [2] with a decreased starting learning rate of 0.01 and increased weight decay parameter of 0.05 to allow for slower and more regularized training. All other parameters remained the same as the defaults. The Top-1, Top-5, learning rate schedule,

and training loss graphs for training are displayed below. This model's Top-1 and Top-5 accuracy on the validation set was 43.920 and 73.600, respectively. During training, 2117 MiB of the GPU memory was used.

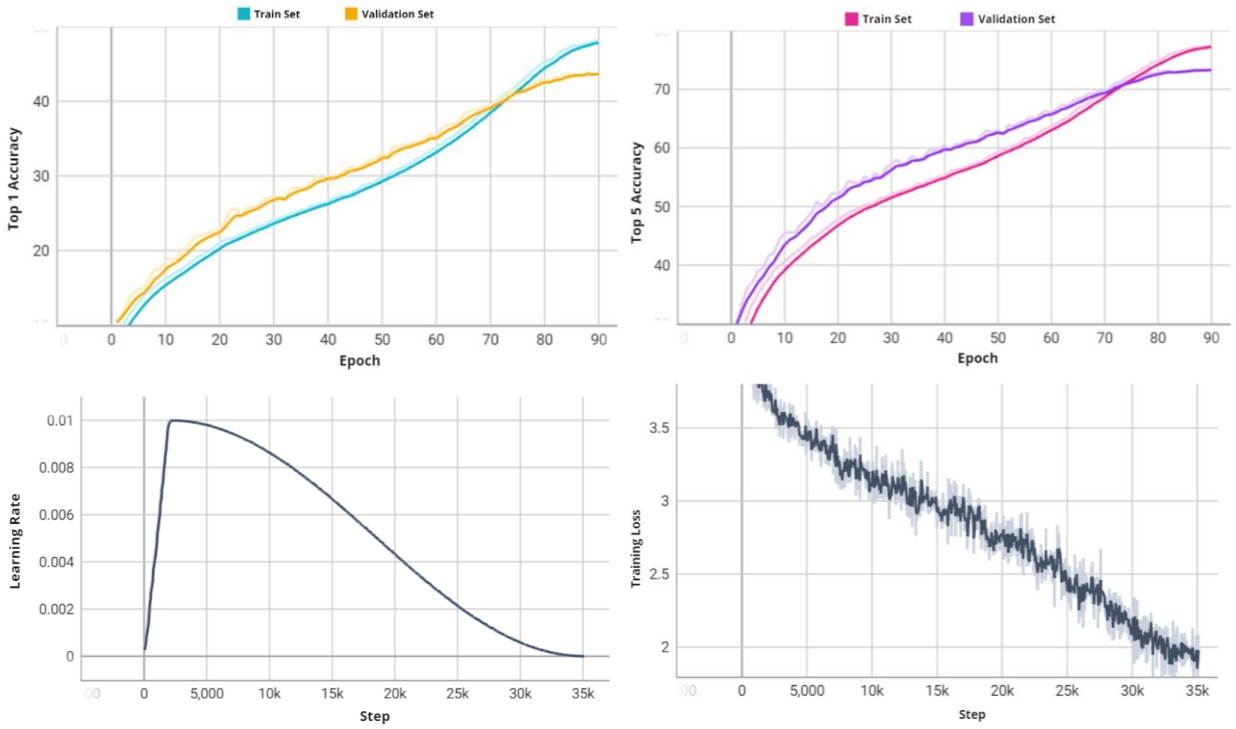

Figure 3: The Top-1 (top left), Top-5 (top right), learning rate schedule (bottom left), and training loss (bottom right) graphs for training SimpleViT.

3.4 CustomNet

To improve its performance, SimpleNet was augmented using batch normalization and skip connections, resulting in CustomNet. Batch normalization allows for simpler and more stable training of deep learning models as it makes the optimization landscape significantly smoother [3]. Skip connections help to preserve information and gradients that might otherwise be lost or diluted by passing through multiple layers, allowing for better training and deeper models without overfitting. The architecture of CustomNet is detailed further below.

The first two convolutional blocks of SimpleNet were kept, each of these blocks includes three convolutional layers, ReLU activations after each convolution layer, and a 2D max pooling layer. In addition, at the end of each of these convolution blocks, 2D batch normalization layers were added.

After these first two convolutional blocks, a series of four convolutional blocks, called 'skip blocks' were added that have skip connections over them. Each skip block has three convolutional layers, ReLU activations, and a 2D batch normalization layer at the end.

After the skip blocks, the last convolutional block and adaptive max pooling layer remained unchanged from SimpleNet. Finally, instead of the one fully connected layer in SimpleNet, two fully connected

layers bridged by a ReLU activation and 1D batch normalization were added to finish the model. A diagram of CustomNet's architecture is shown below.

CustomNet was trained for 60 epochs with default parameters. The Top-1, Top-5, learning rate schedule, and training loss graphs for training are displayed below. This model's Top-1 and Top-5 accuracy on the validation set was 49.690 and 78.700, respectively. During training, 4277 MiB of the GPU memory was used.

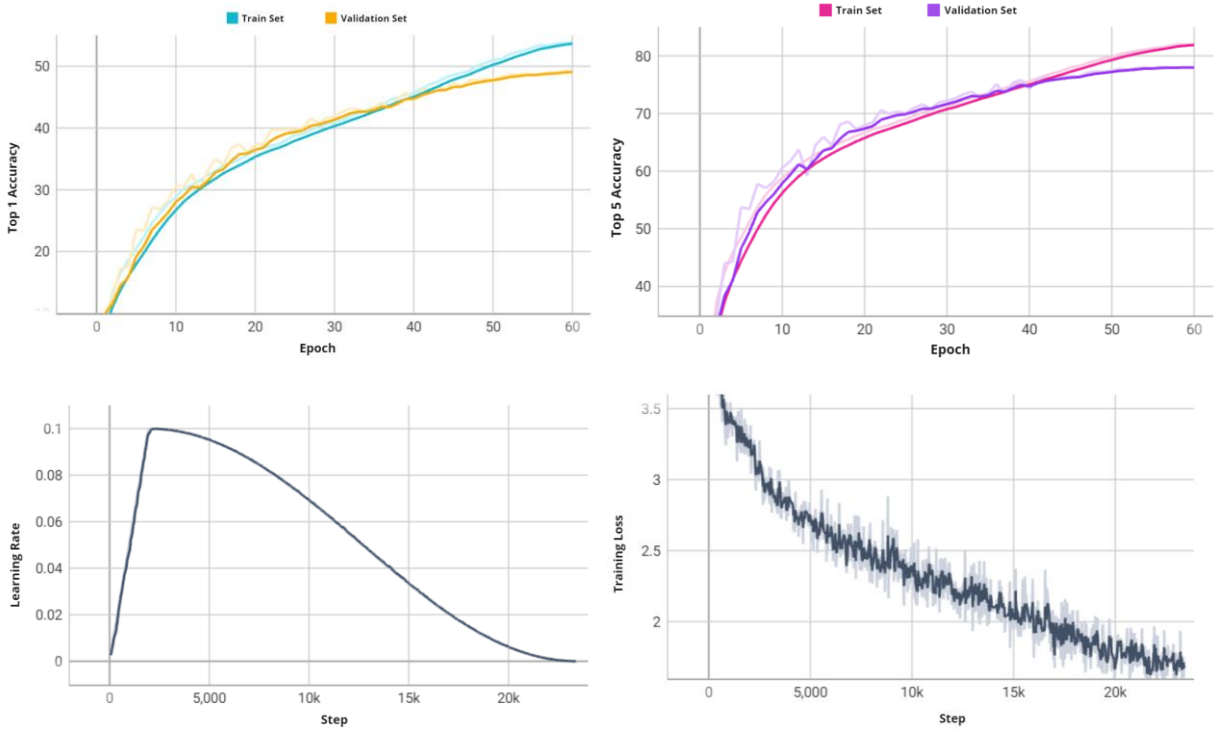

Figure 4: The Top-1 (top left), Top-5 (top right), learning rate schedule (bottom left), and training loss (bottom right) graphs for training CustomNet.

3.5 ResNet18

A pre-trained ResNet18 model was fine-tuned for 60 epochs with the default parameters. The Top-1, Top-5, learning rate schedule, and training loss graphs for training are displayed below. This model's Top-1 and Top-5 accuracy on the validation set was 53.250 and 80.230, respectively. During training, 3192 MiB of the GPU memory was used.

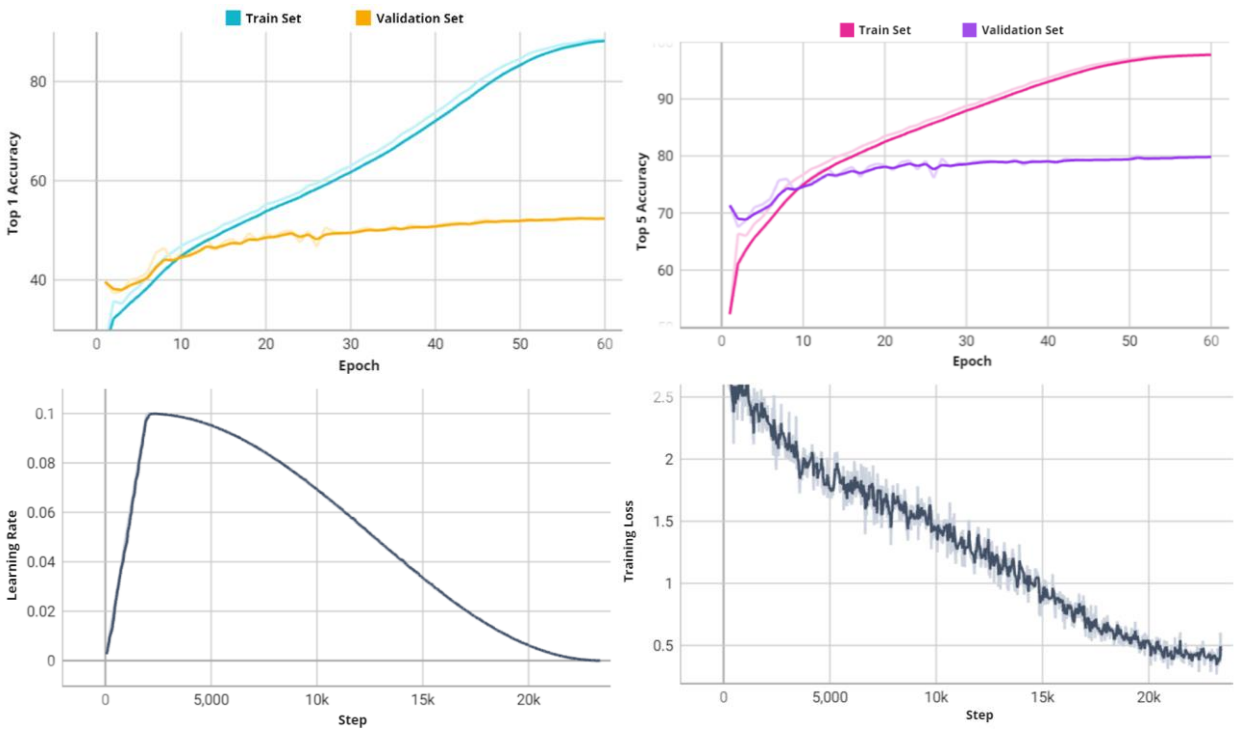

Figure 5: The Top-1 (top left), Top-5 (top right), learning rate schedule (bottom left), and training loss (bottom right) graphs for fine-tuning ResNet18.

3.6 **Discussion: comparing the models**

In Table 1, we see that the Top-1 and Top-5 accuracies of SimpleNet with and without custom convolutions and the SimpleViT model were extremely similar. Meanwhile, the accuracies of ResNet18 and CustomNet were higher, likely due to their more complex architecture. Furthermore, we see the effect of batch normalization and skip connections in the accuracy jump between SimpleNet and CustomNet.

It is interesting that SimpleViT was actually the smallest user of GPU memory out of the models, likely due to the smaller number of parameterized layers compared to the other models. Meanwhile, CustomNet had the largest GPU memory use even though it technically has less layers in total than ResNet18. This is likely due to some memory optimization Pytorch's version of the skip blocks have and because our skip blocks contain three convolutional layers per block instead of only two in traditional ResNet18.

Table 1: The final Top-1 and Top-5 accuracies as well as some training metrics for models presented in this paper.

Model Architecture	Accuracy		Epochs	GPU Mem Used (MiB)
	Top-1	Top-5		
SimpleNet	44.880	73.630	60	3113
SimpleNetCustConv2D	44.900	73.330	60	4881
SimpleViT	43.920	73.600	90	2117
CustomNet	49.690	78.700	60	4277
ResNet18	53.250	80.230	60	3192

For the differences in Top-1 and Top-5 accuracies seen throughout training for each model please see Figure 6. Through this figure, we can see that the accuracies for each model at first exponentially increase as parameters get optimized and plateau as training goes on. We can see at which point each model starts to overfit as the accuracy for the training set significantly overtakes the accuracy for the validation set. Curiously, we see that the final train and validation accuracies are fairly close together for most models except for the ResNet18 model where both the Top-1 and Top-5 accuracy for the training set is far higher than the validation set. This likely indicates the ResNet18 model was significantly overfitted to the training set. This is likely because we are fine-tuning the ResNet18 model instead of training it from scratch.

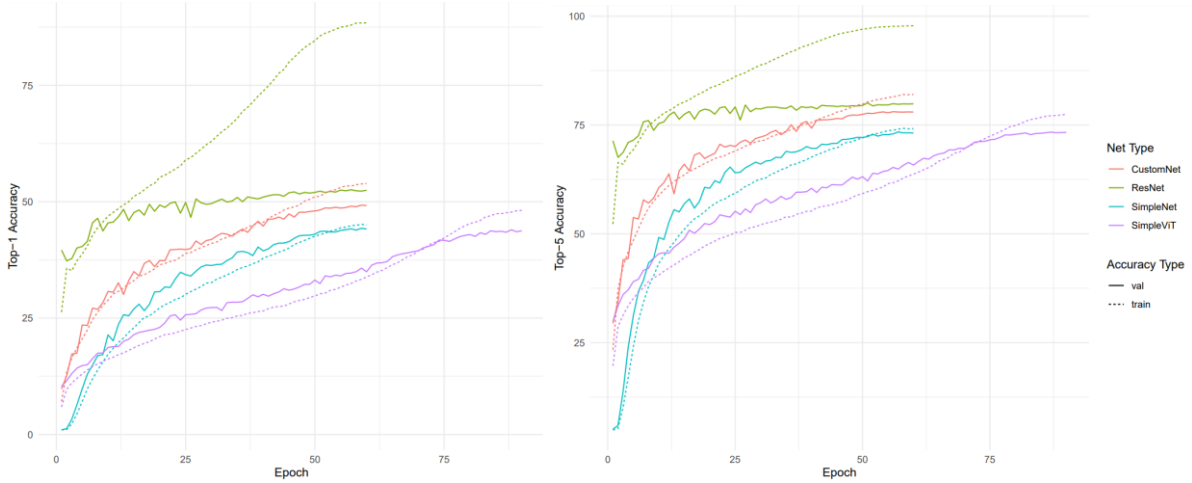

Figure 6: The Top-1 (left) and Top-5 (right) training and validation accuracies computed while training the models presented in this paper.

4.0 Attention and Adversarial Samples

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. *An image is worth 16x16 words: Transformers for image recognition at scale*. In International Conference on Learning Representations, 2021.

[2] I. Loshchilov and F. Hutter. *Decoupled weight decay regularization*. In International Conference on Learning Representations, 2018.

[3] Santurkar, Shibani, et al. *How does batch normalization help optimization?.* Advances in neural information processing systems, 2018.