Heng Chang

3/30/2025

DS549 Deep Learning

Professor Gardos

DS549 Midterm Report

AI Disclosure

I have used Copilot and ChatGPT for this assignment in the following aspects:

1. Using the auto-fill suggestions to complete codes(a few lines). I have used this feature to complete the train() and validate() function in the starter code and also when making miscellaneous adjustments.
2. I used AI for hyper parameter suggestions(often). Example prompts: “Best normalization mean and std for CIFAR100”,“Best RandAugment hyperparameters to use for CIFAR100”
3. I used AI for suggestions: for choosing different optimizers and schedulers(often), as well as when I try to figure out why a model is underperforming.
4. I used AI for code generation: I have used ChatGPT to generate code for two functions: an early stopping mechanism and a warm-up training scheduler. The AI-generated portion is marked with comments in the submission code.
5. I used AI for debugging. Example prompt(to VScode Copilot): “Explain the ValueError”, “Try to identify potential errors in this file”

Model Description:

- Part 1: I used a VGG-like structure borrowed from a Kaggle notebook(Fares Sayah). I made this choice because I have used this architecture for CIFAR10 previously for DS549 and it worked well. I did RandomCrop and RandomHorizontalFlip as simple data augmentation. I used Adam and StepLR which worked well for my previous task on CIFAR10 along with the parameters I have used previously. The simple model gets about 16% on the odd testset.

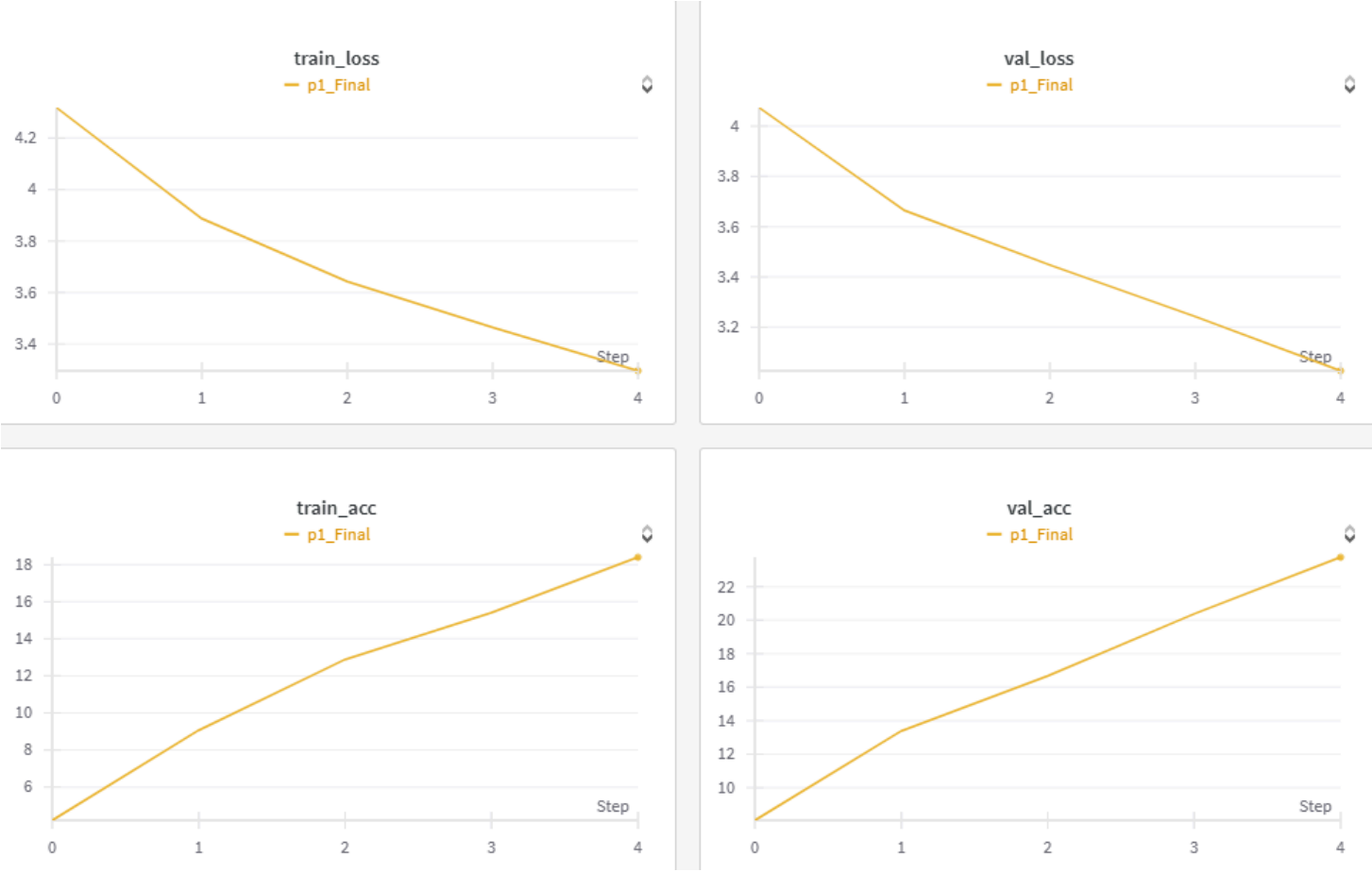

Figure 1. Part 1 Model

- Part 2: For Part 2, I have experimented with a few different predefined models including ResNet50, ResNet18, Efficientnet_b0 and different combinations of optimizer and scheduler. The result seems that the choice of optimizer and scheduler does have a significant impact on the model performance, as in this part we are training for only 5 epochs. For similar reasons, I kept the simple data augmentation used in Part 1 as it seems that the model would tend to underfit with too much augmentation. The final model I used for Part 2 is ResNet1, which is complex to an extent so that it could understand deeper features of the image but also light enough so that it could somewhat effectively learn for only 5 epochs. Its accuracy on the odd testset is around 23%.

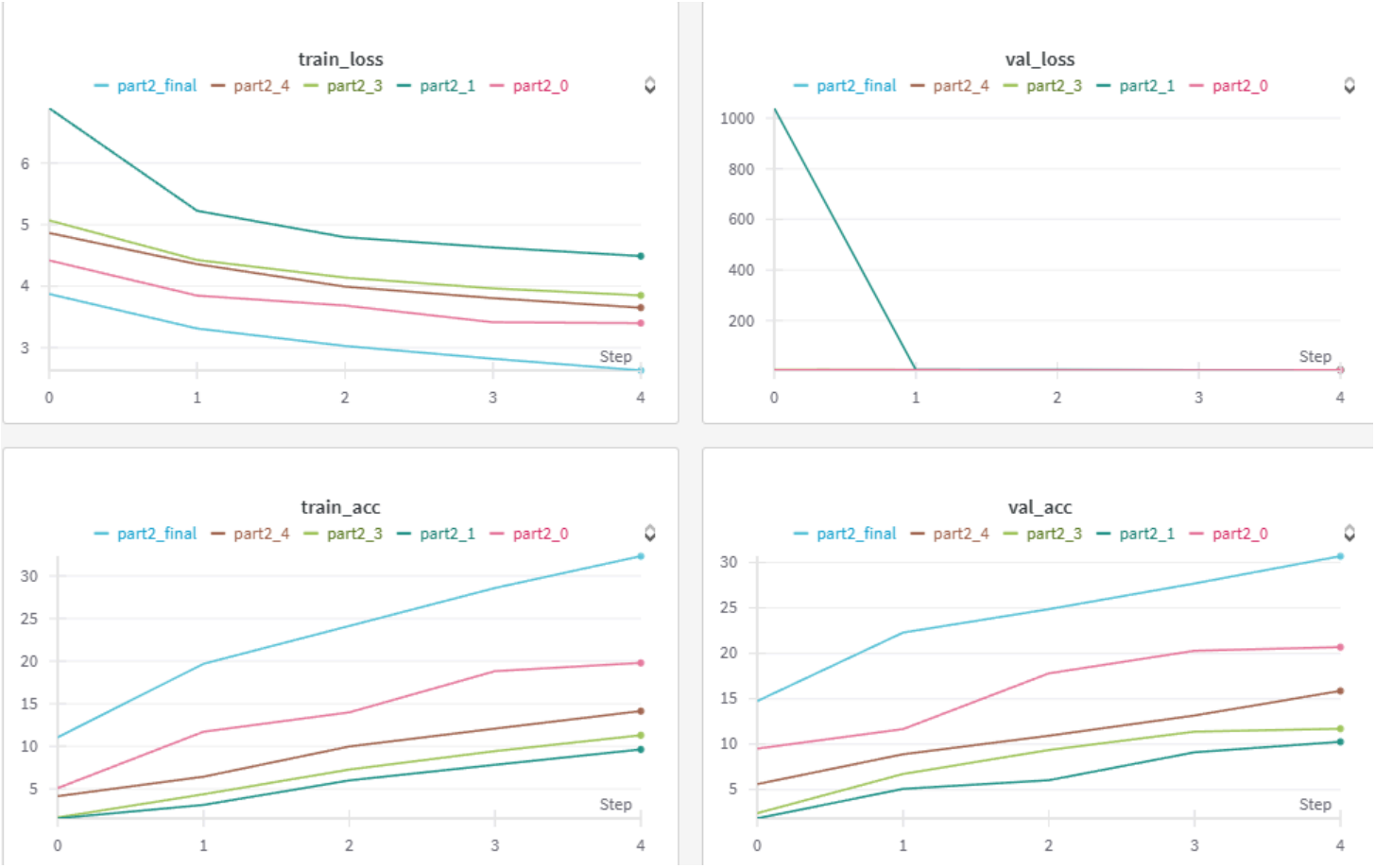

Figure 2. Part 2 Models

- Part 3: At the early stage of Part 3, my main strategy was experimenting with different pre-trained networks for small epochs to screen out the model architectures that have potential. I have experimented with architectures including Resnet, Efficientnet, Shufflenet, pretrained on different datasets. To my surprise, Despite performing strongly on the training set, all the models pretrained on CIFAR100 that I have experimented with performed poorly on both the clean and odd testset. I have experimented with different regularization approaches such as more intense data augmentation and layer freezing, but did not result in a significant boost. This is likely due to overfitting and the failure to actually leverage transfer learning. This is shown in the figure below. The part3_6 and part3_7 family are models pre-trained on CIFAR100.

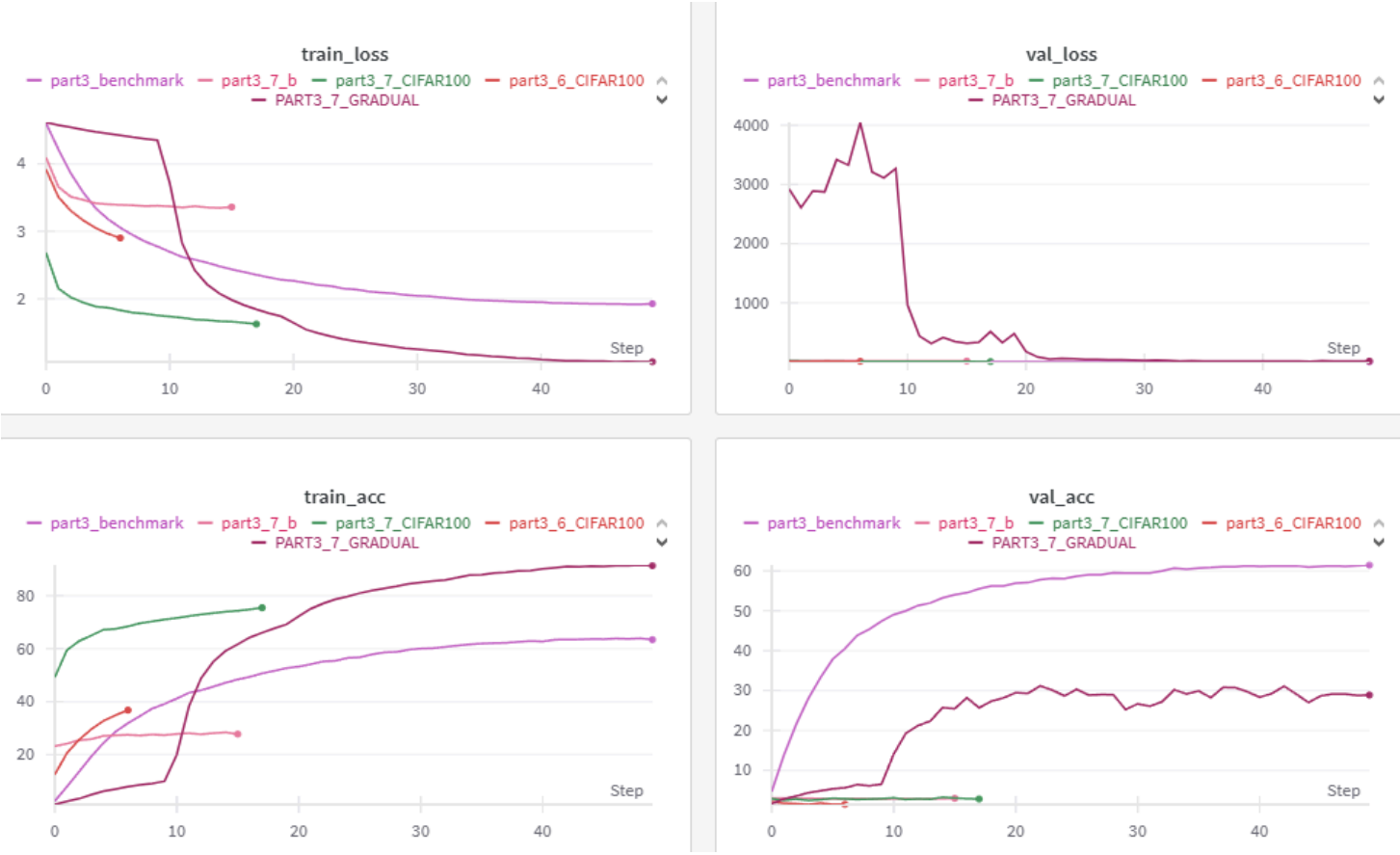

Figure 3. CIFAR100 models vs BenchMark model pre-trained on ImageNet

I then shifted focus to models pre-trained on ImageNet. After some browsing on kaggle(Omar Zakaria), asking AI, and experimenting with different models, I settled on EfficientNetv2 with weights pre-trained on ImageNet. These powerful models converge quickly and can easily score a high validation accuracy. However, they also overfit extremely fast and perform terribly on the odd testset.

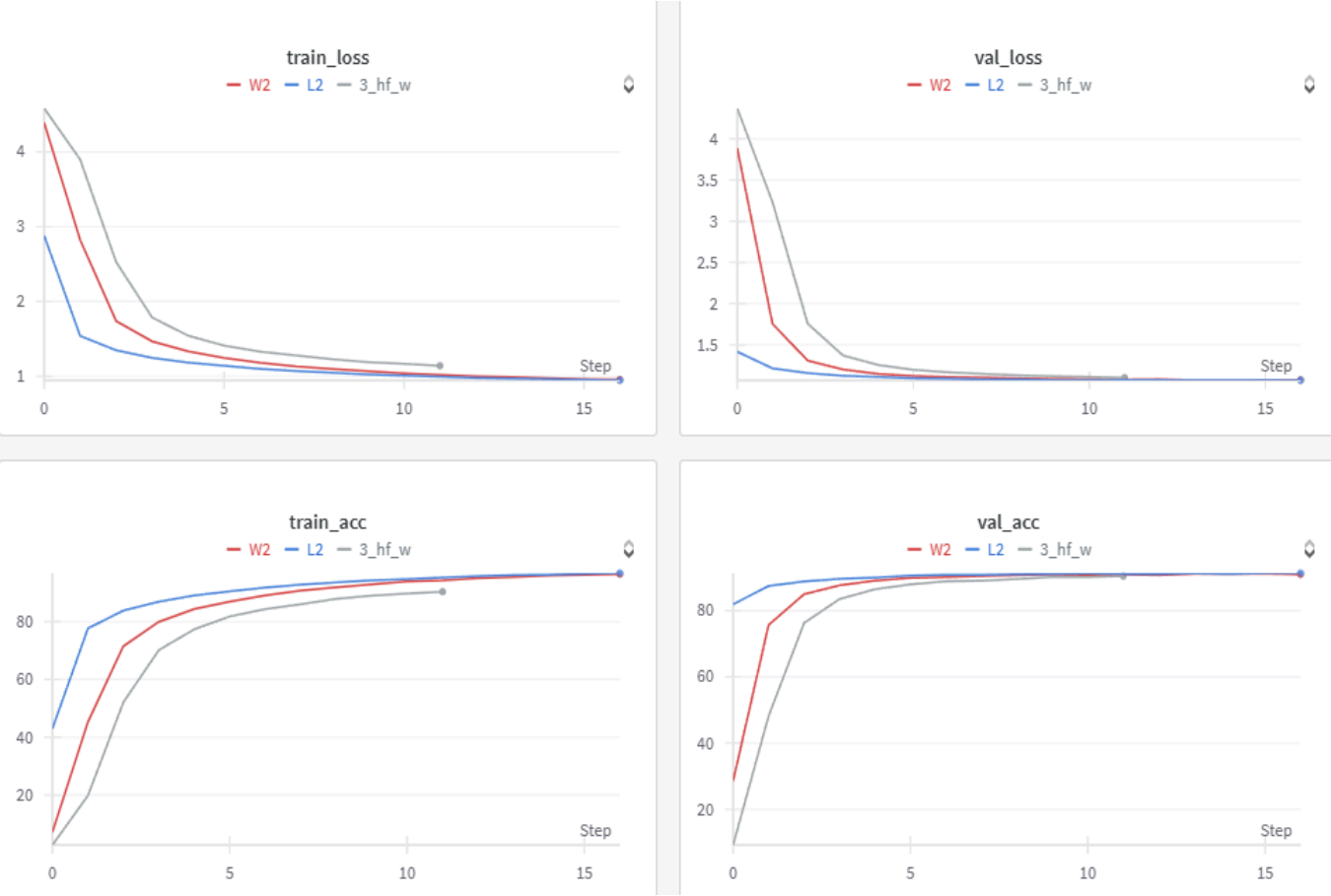

Figure 4. Overfitting EfficientNetv2 models

To address the overfitting and failure to generalize, I have experimented with several regularization tools including freezing layers, more intense augmentation, and focal loss. The most effective approach I found was reducing the size of the training set images. The typical

input dimensions for EfficientNetv2 is 224 * 224, and reducing the size could effectively reduce the overfitting and grant better generalization as it blocks the model from memorizing the details specific to the images in the training set.

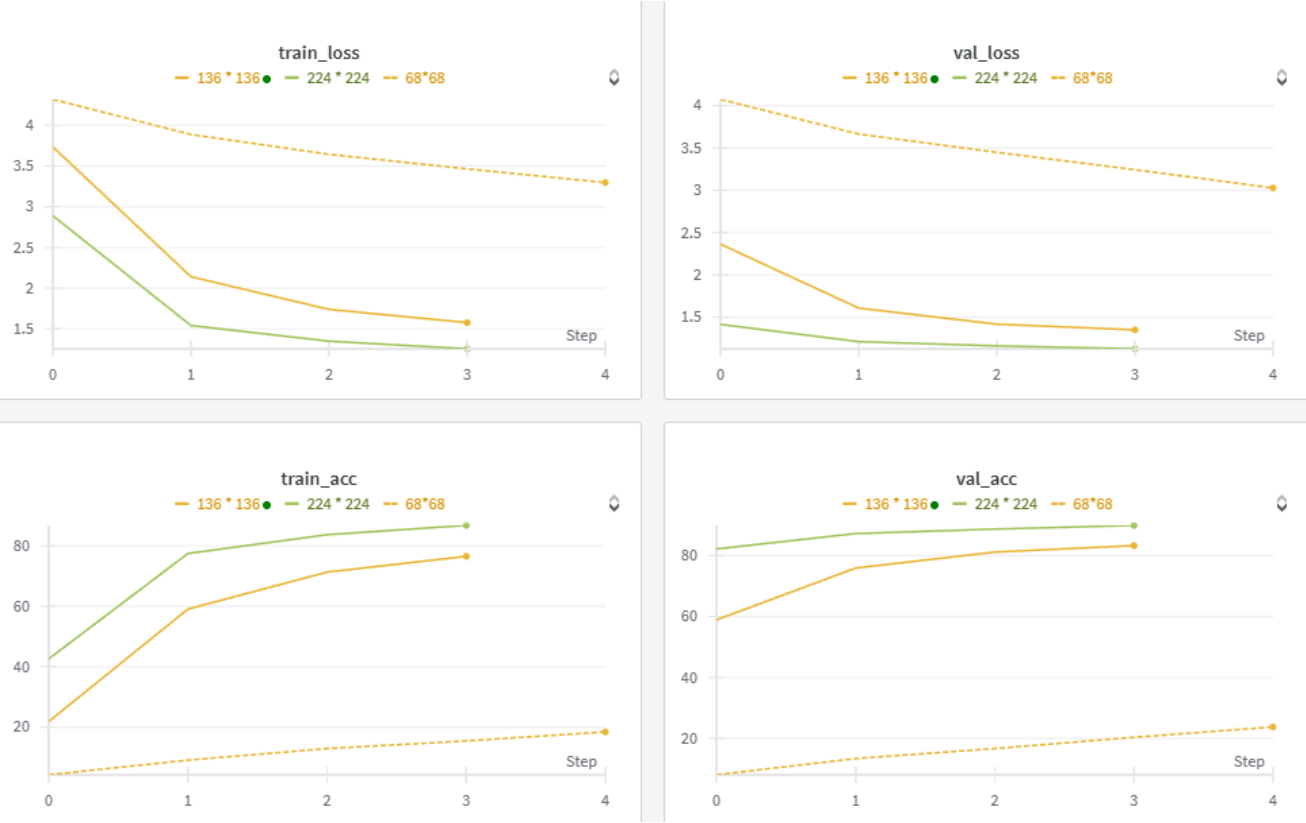

Figure 5. Same networks trained on different sized training images

After testing for different image size for training sets, I settled down on 68 * 68. Then, I tested for a few different augmentation methods, including Autoaugment, popular choices, and combination of transformations I came up with myself(did not work well), I got my final model that is fine-tunes on a EfficientNetv2 pre-trained on ImageNet dataset that yields about 49% accuracy on the odd testset after 50 epochs of training.

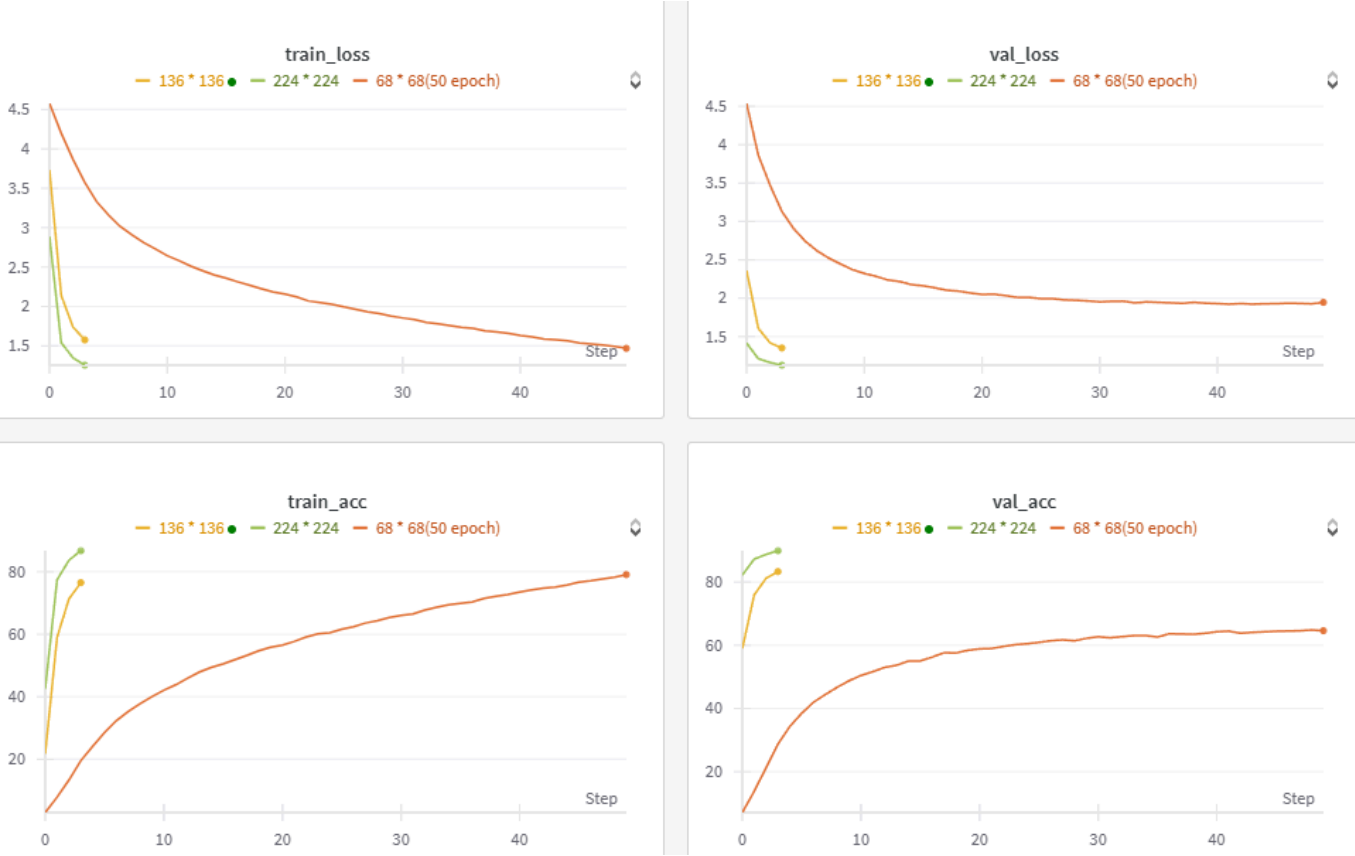

Figure 6. Same networks trained on different sized training images

Citation 1:

Author/Username: Omar Zakaria

Title: EfficientNet-Cifar-100

Platform: Kaggle

URL: https://www.kaggle.com/code/omarazakaria/efficientnet-cifar-100/notebook

Citation 2:

Author/Username: Fares Sayah

Title: Cifar-10 Images Classification using CNNs

Platform: Kaggle

URL: https://www.kaggle.com/code/faressayah/cifar-10-images-classification-using-cnns

Citation 3:

Author/Username: Professor Thomas Gardos

Title: Python Virtual Environments and Package Management

Platform: DS 549 Course notebook

URL:https://trgardos.github.io/ml-549-fa24/02_python_environments.html#exercise-instructions