**ECE 09495/09595**

**Assignment 2 – Jacob Matteo**

**Instructions**

1. Max Credit: 100 Points
2. All questions are from the Textbook – Dive into Deep Learning [(https://d2l.ai/)](https://d2l.ai/).
3. Submit a single PDF.
4. Please do not include code. Upload the code to your GitHub, share it publicly and add link in the assignment.

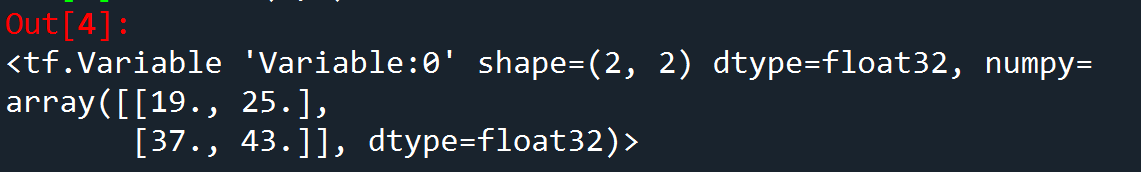
Any code I used not straight from the textbook are located on this github, organized by assignment: https://github.com/jmatteo/Machine-Learning-Fall-20.git

**Questions**

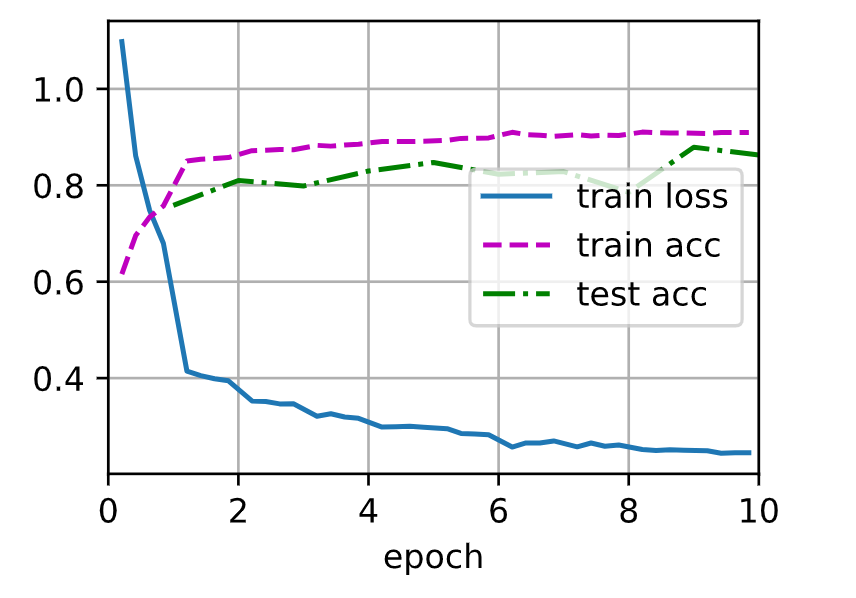
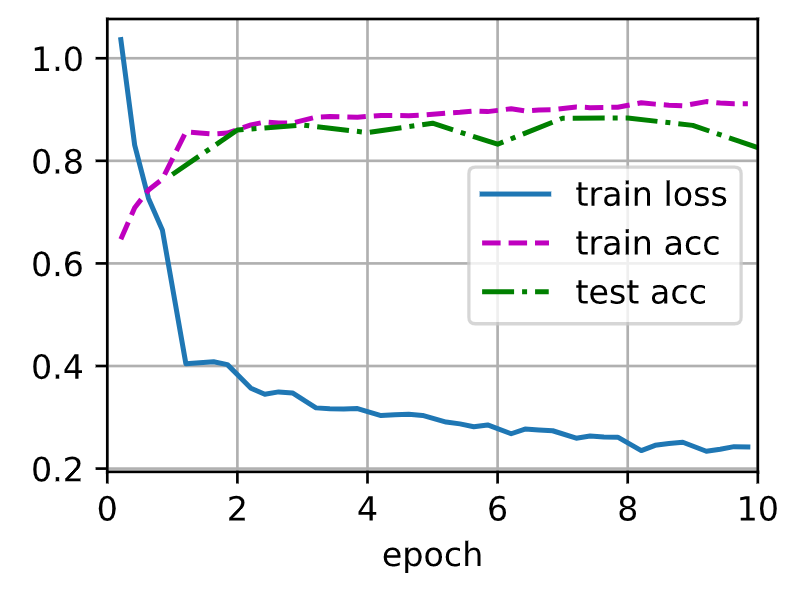
1. Section 6.6.4 (all questions) 40 points

Expected Answers:

* 1. Q1 - Provide epoch vs loss/accuracy (training and validation) curves (as given at the end of section 6.2.2).



* 1. Q2 - Provide a single set of epoch vs loss/accuracy (training and validation) curves for the best model you could train. This will require substantial hyperparameter tuning. Please provide only one graph of the MLP that performs the best.
  2. Q3 - Provide a set of epoch vs loss/accuracy (training and validation) curves for the best model.
  3. Q4 – A plot for the activations.

1. Section 7.1.6 – Question 4 5 points
   1. The dominant part of AlexNet’s memory usage footprint is the large amount of convolutional layers required to store and compute.
   2. The dominant part of AlexNet’s computational usage is, like in the previous question, the large amount of convolutional layers the algorithm needs to compute through.
   3. Because AlexNet works on large amounts of small convolutional neural networks that get combined to create more complex shapes, high memory bandwith and capacity is a must during compute.
2. Section 7.2.5 – Question 3 5 points
   1. Decreasing the height and width of the images decreases the accuracy of the neural network but slightly decreases the amount of time needed to calculate the network.
3. Section 7.4.5 – Question 3 5 points
   1. With GoogLeNet and NiN, rather than making their neural networks deeper, as in AlexNet and VGG, they make it wider by having multiple convolutions running in parallel and concatenating the results, whereas the latter made theirs deeper involving multiple layers getting processed and reprocessed over and over until the final result is made.
4. Section 7.4.5 – Question 4 5 points
   1. We need a long range convolution initially because not only do input files tend to be larger and larger, but these data sets get individually smaller as they get processed down the chain of convolutions until finally they get to one output.
5. Section 7.5.8 – Question 2 5 points
   1. Without:  With: 
6. Section 7.5.8 – Question 3 5 points
7. Section 7.5.8 – Question 4 5 points
8. Section 7.6.6 – Question 5 5 points
9. Section 7.7.7 – Question 1 5 points
10. Section 7.7.7 – Question 2 5 points
11. Write a 2-page summary on the EfficientNet paper 10 points

Blog [- https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html](https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html)  Paper - <https://arxiv.org/abs/1905.11946>

EfficientNet is a new neural network which is more efficient and powerful than previous convolutional neural networks. It was built off of a new scaling method that uniformly scales all dimensions of depth, width, and resolution using a simple yet highly effective compound coefficient. This results in a neural network that ends up being more accurate in its outputs, yet takes less parameters to do so. To make it even more powerful, the team used another neural architecture to design a new baseline NN and scale it up to contain a family of models. These models are called EfficientNets and achieve higher accuracy and efficiency than regular convolutional neural networks (CNNs). One of the EfficientNets, EfficientNet-B7 placed #1 on ImageNet, a deep learning tool which supplied masses of data to train and test neural networks, with a score of 84.3% which being 8.4 times smaller (in terms of parameters) and 6.1 times faster than the best CNNs. It also received top placements on other tests, such as CIFAR-100, Flowers, and more with many less parameters than its competitors.

When EfficientNet is being optimized, it is looking for better floating-point performance rather than better latency like other neural nets. This is because EfficientNet is designed to be run on many different types of hardware. To do this optimization, EfficientNet does a type of search called a grid search which allows it to find the relationship between different scaling dimensions of the baseline neural network. These parameters are width, depth, and resolution. Width adds more nodes to layers, depth adds more duplicate blocks, and resolution adds more layers to a block. These three scaling parameters can be combined to create a compound scaling method. Also, these parameters can be both increased and decreased, meaning if better performance may be achieved by going back to a smaller neural network, the system can do so. This is important as the baseline network is relied on heavily, so optimizing it can increase performance many times over.

Neural networks can be adjusted for better accuracy by feeding in larger images. Because larger images can better bring out details of what is stored in the image, it is easier for the neural network to discover the patterns inside it. However, with image sizes too big, some diminishing returns take place. This typically happens around image resolutions of 560x560. Scaling up the dimensions of the network’s with, depth, and resolution creates the same outcome. The best way to obtain better accuracy is to balance all dimensions of the network width, depth, and resolution during convolutional neural network scaling.

After a certain point, no matter which way you scale your model, you will only gain marginal performance increases. It is at this point where editing the baseline of your neural network needs to take place. By increasing the performance of your baseline network, you increase the performance of each subsequent layer in your network. This allows EfficientNet to target any resource constraints that may be thrown at it while maintaining the efficiency of its model and allowing for a more effective scaling-up of itself.

EfficientNet is trained on ImageNet with a RMSProp optimizer with a decay of 0.9 and momentum of 0.9, a batch norm momentum of 0.99, a weight decay of 1e-5, and an initial learning rate of 0.256 that decays by 0.97 every 2.4 epochs. This is similar settings to many other neural networks. EfficientNet is also set to have a drop out ratio from a minimum of 0.2 for EfficientNet-B0, up to 0.5 for EfficientNet-B7.

In different neural network tests, EfficientNet tends to be faster and much smaller (in parameters) than its competition. For example, in one of the tests, EfficientNet beat the previous first place network, GPipe, while being 8.4 times smaller. This is the result of better architecture, scaling, and training settings. In a different test, EfficientNet beat out a different network, ResNeXt-101, while using 18 times fewer floating point operations (FLOPS).