Trevon Woods
Computer Vision - ITAI - 1378
Patricia McManus
09/25/2025

Chihuahua or Muffin – Convolution

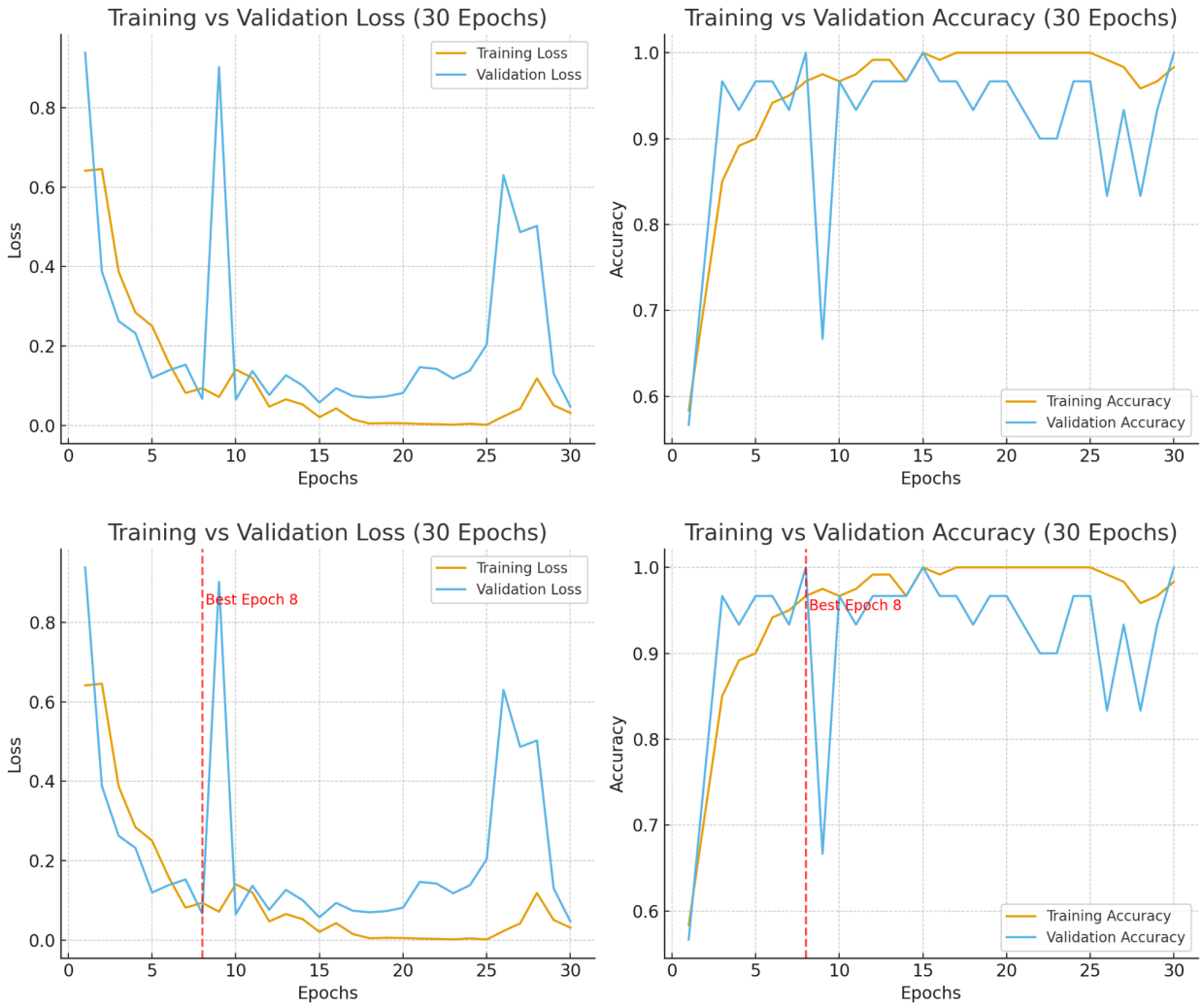
The main difference between the CNN architecture and the traditional neural network from lab 4 is how features are passed and processed from one layer to the next. In the traditional approach all of the neurons in each layer are connected to every other neuron in preceding layers. So, data is passed directly to the next layer without any additional computation happening in between — this is what is called a Fully Connected Neural Network (FCNN). A CNN on the other hand is specifically used for processing data with a grid-like structure like images. It uses convolutional layers that apply filters to input images to extract complex features like edges, corners, and textures. Then what precedes these layers is an activation function that introduces non linearity and a pooling layer to reduce the spatial dimensions of the images. After the data runs through this process the features are then passed through Fully Connected layers for classification.

Now I'll dive into my experience training the model and convey my challenges and achievements along the way. I initially started with an image size of 28 x 28 just to get a base line accuracy from which I could improve the model. It achieved a peak and consistent accuracy of around 83-86%. This is a way better initial outcome than the traditional neural network from the previous lab. So just like I did previously, I increased the size to 128 x 128. This allowed me to obtain a decent accuracy around 86-93% accuracy but it was unstable throughout the training run.

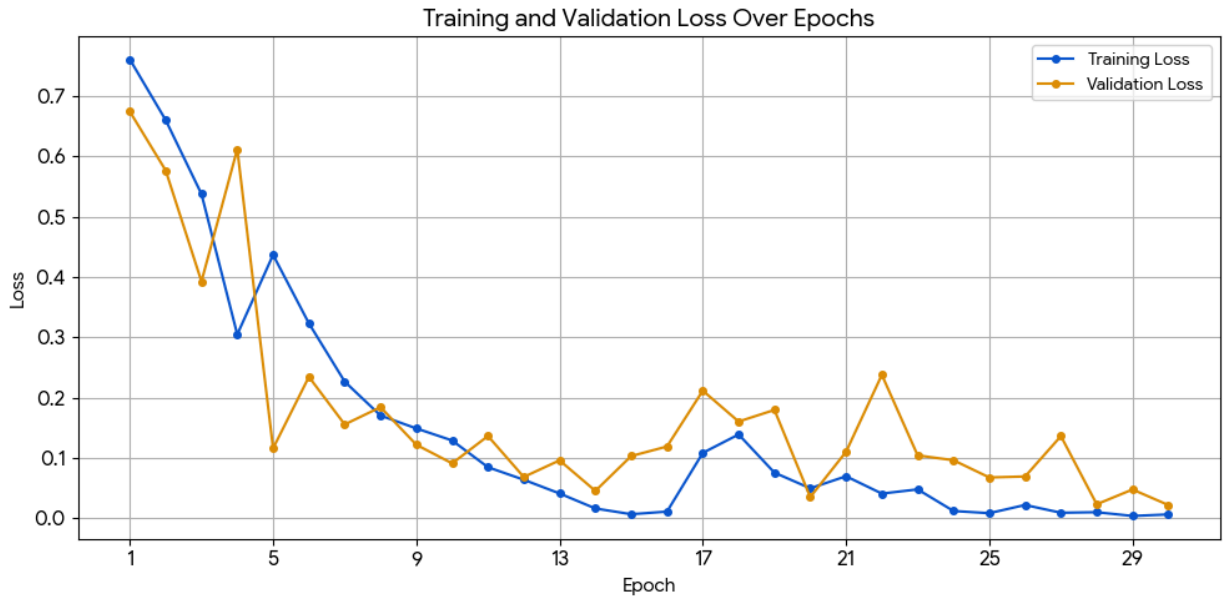
I then experimented with the optimizer, SGD seemed to start off with a lower training and validation accuracy and gradually increase as opposed to Adam which started a little higher than SGD and converged at higher accuracies earlier in the training cycle. SGD with a learning rate of 0.01 worked well and achieved a peak of 100% accuracy around 20 epochs but it was not consistent and the model ended around 93.3%. Next I did some tests by tuning the learning rate. With SGD I tried learning rates of 0.001, 0.01, and 0.1. The only one that seemed to work well enough was 0.01. Anything higher or lower the model was either stuck around 40-60% accuracy range or bounced from high accuracies of 70% or 80% and back down. For Adam, the base value was the best one to use. Any changes to it either higher or lower had no positive effects on the model's accuracy.

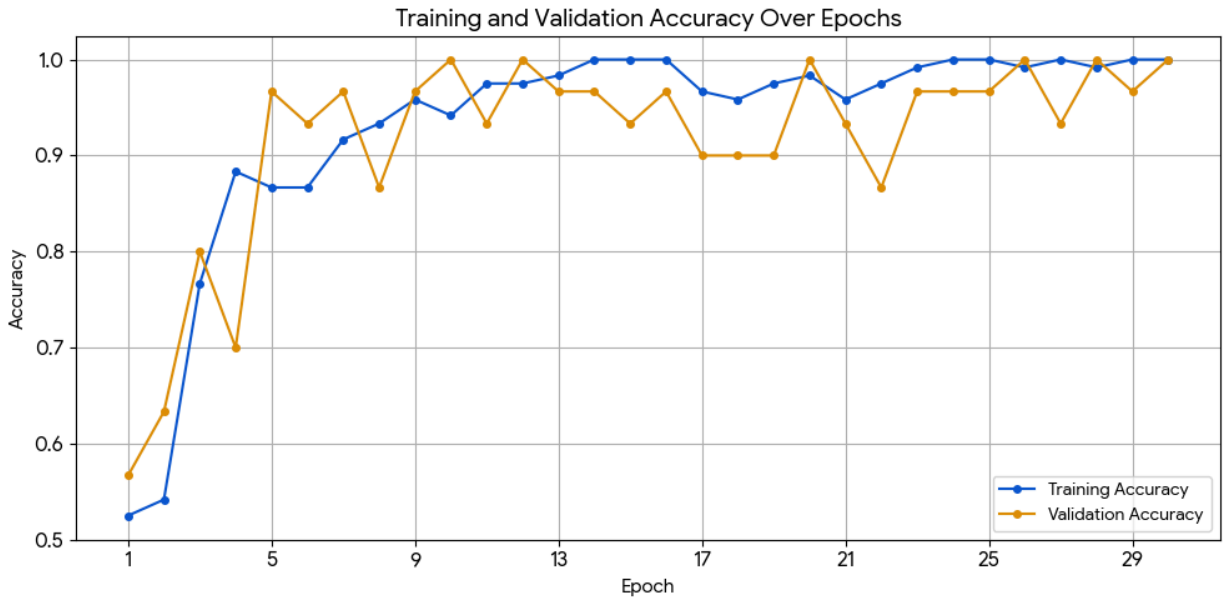
The batch size was next, I did experiments by changing the value lower and higher than my base which was 32 like it was in the previous lab. When I used the SGD optimizer, a value higher than 32 kept the model's accuracy at a consistent 60-70%. Even if the model happened to achieve a higher accuracy in the later epochs it was soon met with a big drop and never recovered. The sweet spot seemed to be a batch size of either 10 or 20. This gave my model the

ability to reach around 96.6% validation accuracy.



Finally, in the hopes of obtaining a few percentage points higher in accuracy I decided to add another convolution. This became a bit of a challenge because I kept getting a shape error. I took a step back and tried to understand how the features from the last convolution were being flattened. Once I understood this I was then able to calculate the right input value to the linear layer so I could begin training again. To reiterate, the initial network configuration with the base network layers achieved a consistent accuracy of 93.3%. The addition of another larger convolutional layer allowed me to obtain a final accuracy of 100% on both training and validation.





Comparing the two neural network architectures from lab 4 and this lab is like comparing David to Goliath. The traditional neural network works but from my experience needs to be trained for more epochs and requires higher quality images (bigger image sizes) to learn any meaningful features. Also in terms of training time I recall that each training run took maybe 15-20 mins. For the convolutional network it was very easy to obtain a decent initial accuracy baseline to improve on. Also it didn't need an image size bigger than 128 x 128 and took 20 epochs less than the traditional network to achieve 100% validation accuracy. The training times were around 5-10 mins which was exorbitantly faster. Another major plus for the CNN is that the architecture only contained around 8.7 million parameters while the FCNN contained around 100.7 million parameters. I should also mention that all of my training was done using the CPU

because I wanted to see how slow or fast training would be on an image classification task without a GPU.

Now to get into the real world applications and ethical considerations. The CNN seems to be the more practical for real world applications due entirely to the speed and size of the network architecture. Also the fact that the convolutions use filters to essentially see what is going on in the images, and extract useful features for inference puts the CNN in a league of its own. This could be used for image tasks where the fine-grained or ambiguous details of the image need to be considered for accurate classification. It would more than likely excel for facial recognition, surveillance, and medical imaging to find abnormalities in x-rays or photos of injuries, wounds, or irritations. As for the ethical considerations they remain the same as just about any machine learning system. For image classification in particular, when collecting data for a dataset, one must make sure that the data collected encompasses all demographics. The disregard of this consideration would create a bias that discludes or inherently hurts a demographic that has been underrepresented.