Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Introduction:

This lab is an exploratory exercise in deep neural networks. Specifically, this lab uses Digits, the NVIDIA Deep Learning GPU Training System. This tool is used to rapidly train highly accurate deep neural networks for image classification, segmentation and object detection tasks. This lab uses two different datasets, The Minist dataset, which is a data set of hand drawn digits, and the Caltech256 data set, which is a dataset of images for use in image classification. Both of these datasets are loaded into Digits and then students are tasked with varying the hyper parameters of the models in order to see the effects that result.

Part 1 Data:

The Mnist Database Description:

The Mnist database is a database of handwritten digits, whose training set is 60,00 examples, with a testing set of 10,00 examples, each example comes as an image paired with a label. The training set images are 99,912,422 bytes, and the labels are 28,881 bytes. The testing set images are 1,648877 bytes, and the labels are 4,542 bytes. The images are black and white and normalized to fit in a 20x20 pixel box while persevering their aspect ratio. The data set is comprised of 30,000 examples from NIST's Special Database 3 and 30,000 examples from Special Database 1, which contain binary images of handwritten digits. SD-3 contains digits written by Census Bureau employees, while SD-1 was collected from high school students.

Part 2 Train an Initial Model:

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 2.1

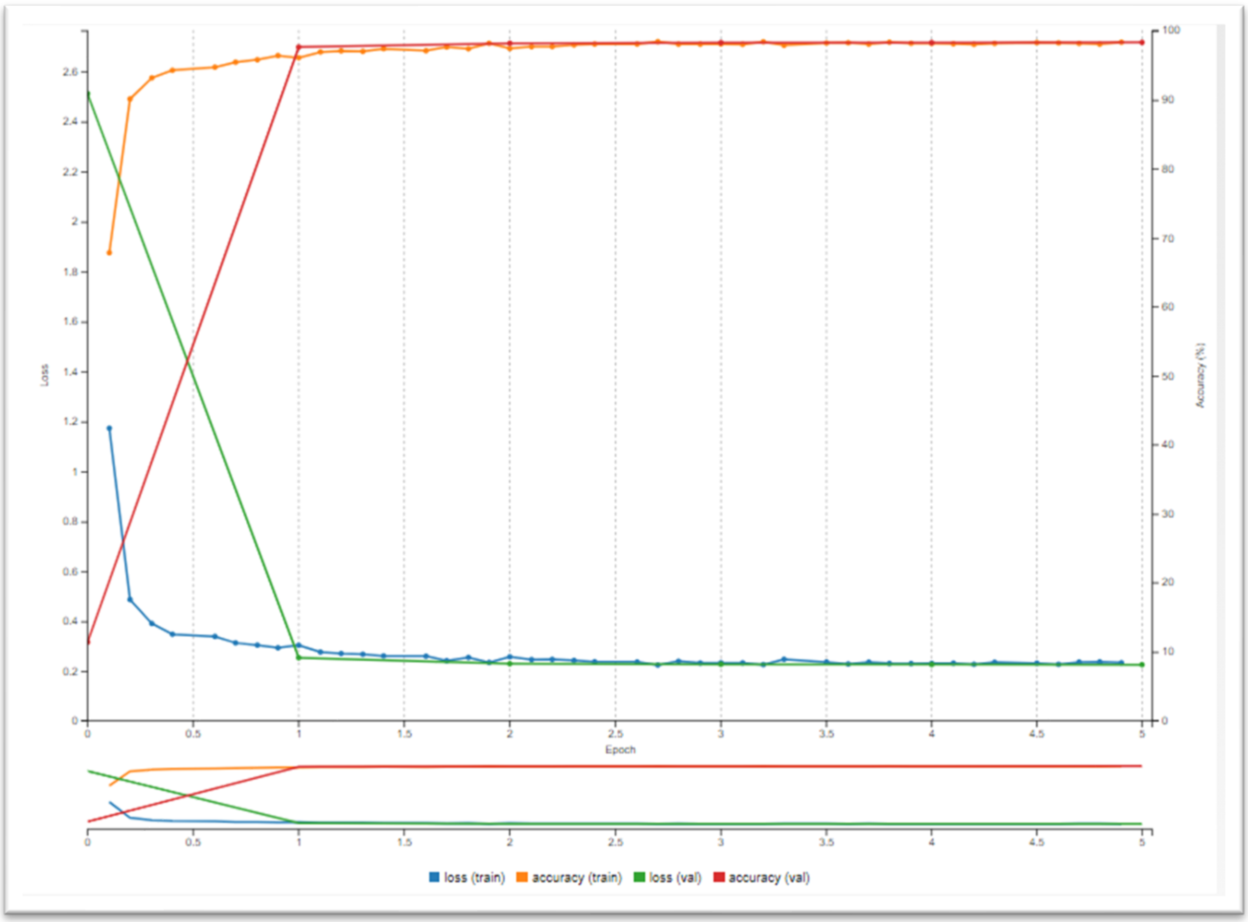


Image 2.1 denotes the training loss plot of a Digits model that is trained on the Minist database, using a LeNet network, a default batch size of 8, a learning rate of 0.01, no data transformations, and is trained for 5 epochs.

<u>Part 3 Inference:</u>

Image 3.1

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

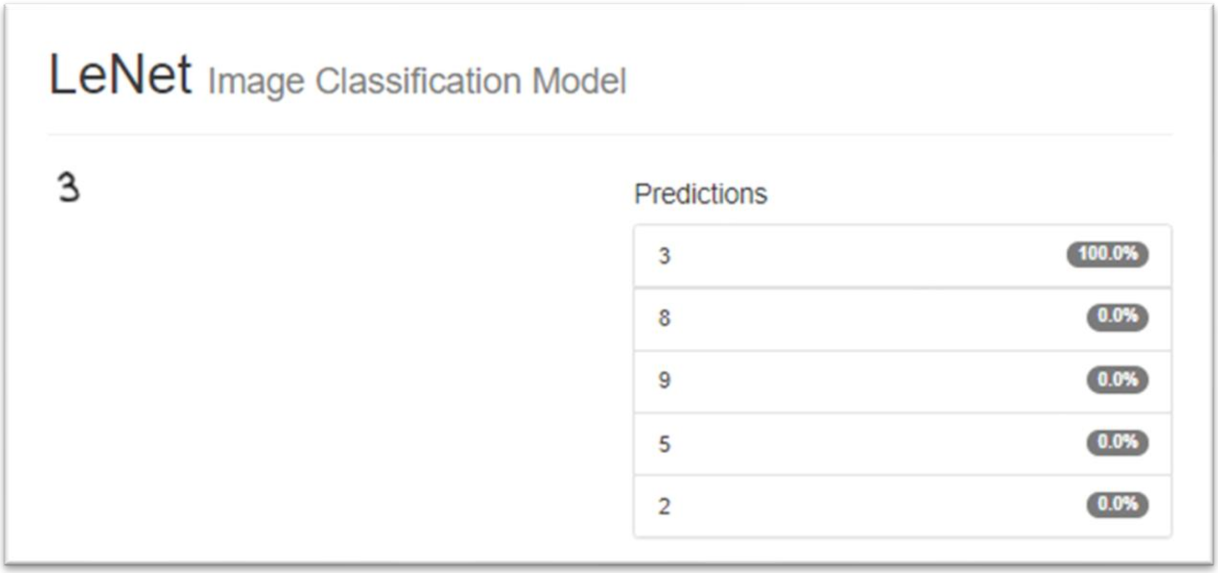
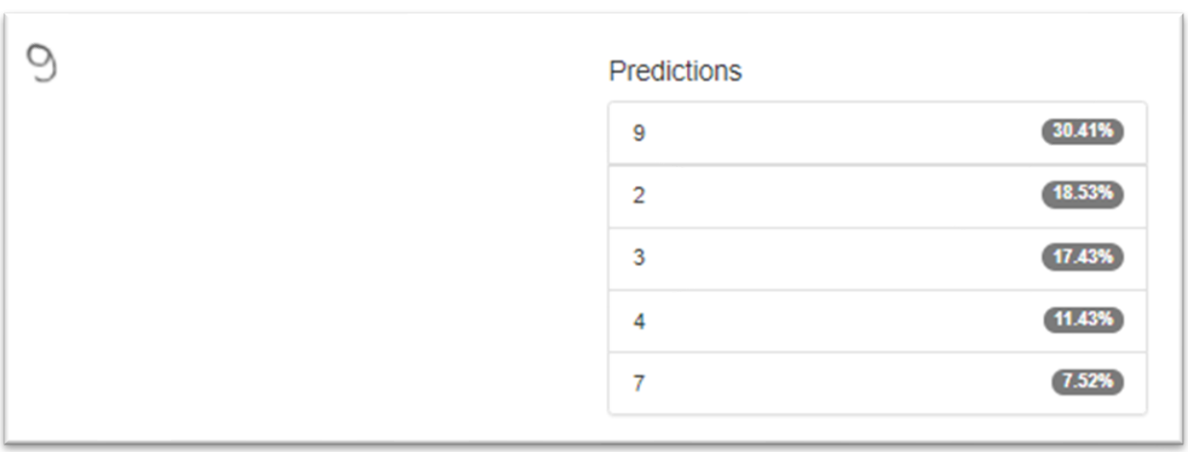


Image 3.1 displays a prediction for what the numerical representation is of the hand drawn digit in the upper left of the image, which comes from the same data base that the training data came from. As seen above, the prediction is correct. It is not surprising that the model got this correct, due to the fact that I would expect 3 to be one of the easier numbers to differentiate, however, it is surprising to me that 100.0% of its prediction was that it was a 3. I am surprised at the certainty of the model and how it did not predict that it was another number even less than 1% of the time. It is also interesting to note the numbers that were listed as potential competitors even though no weight was given to them, most all of them somewhat emulate the 3's vertically aligned semi circles.

Image 3.2



Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 3.2 depicts the same model guessing what the hand drawn digit is that is in the upper left hand corner, which was found via a google image search. The model did significantly worse for a "googled" image of a handwritten 9. While it still predicted the correct number, it only predicted the digit to be a 9 with 30.41% certainty, which is a vast difference from the previous test's 100.0%. A reason for this may be that 9 is a more difficult number to identify, as many numbers are curvy with a contained circle and a tail. In addition, for the image that I classified the image was 122x135 pixels, which is a much different format than the small, square 20x20 format that the model was trained on. Lastly, this 9 has a slight lean to left, which may not be accounted for in the training data where digits are size normalized and centered.

Part 4: Epochs, Batch Size

Image 4.1

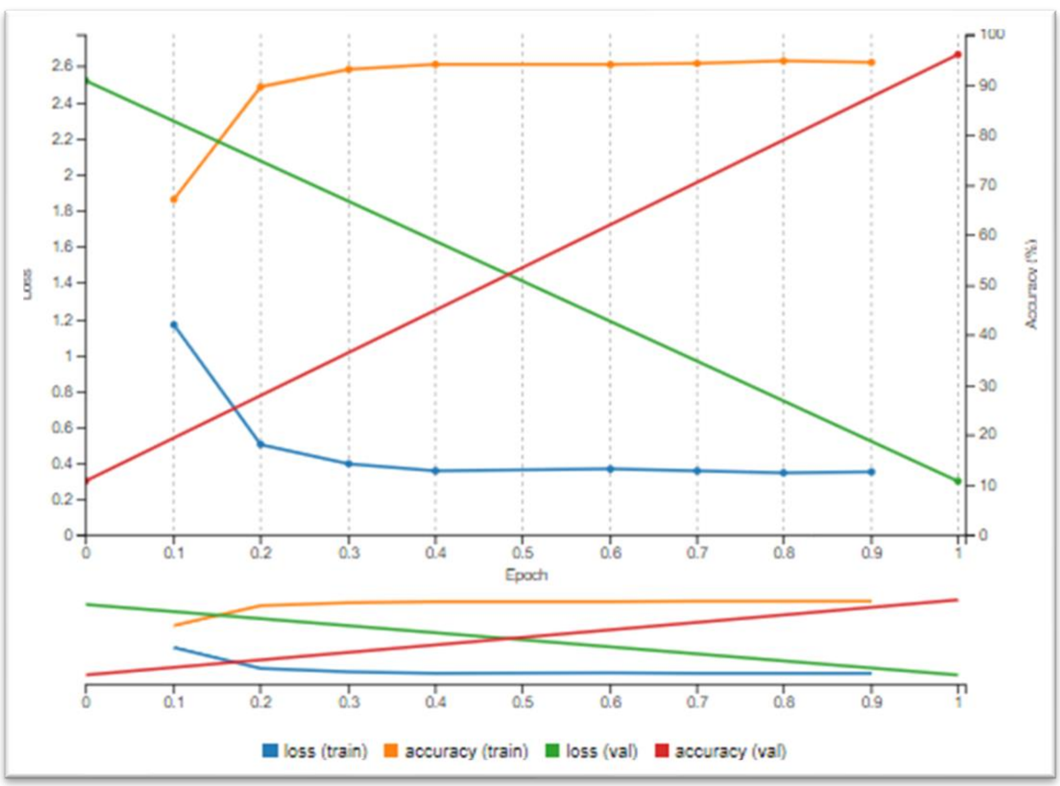


Image 4.1 represents the training loss plot of a model trained in the same manor as the one depicted in image 2.1, but instead is only trained for a single epoch.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 4.2

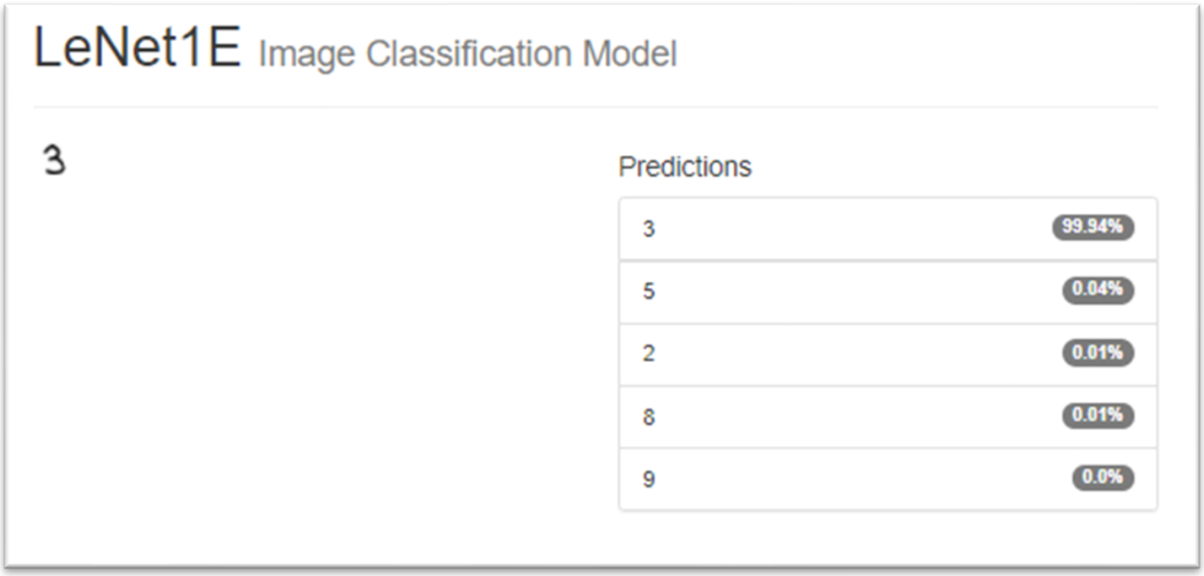


Image 4.2 uses the model from image 4.1, but is tasked with classifying the same hand drawn digit as image 3.1 displays.

Image 4.3

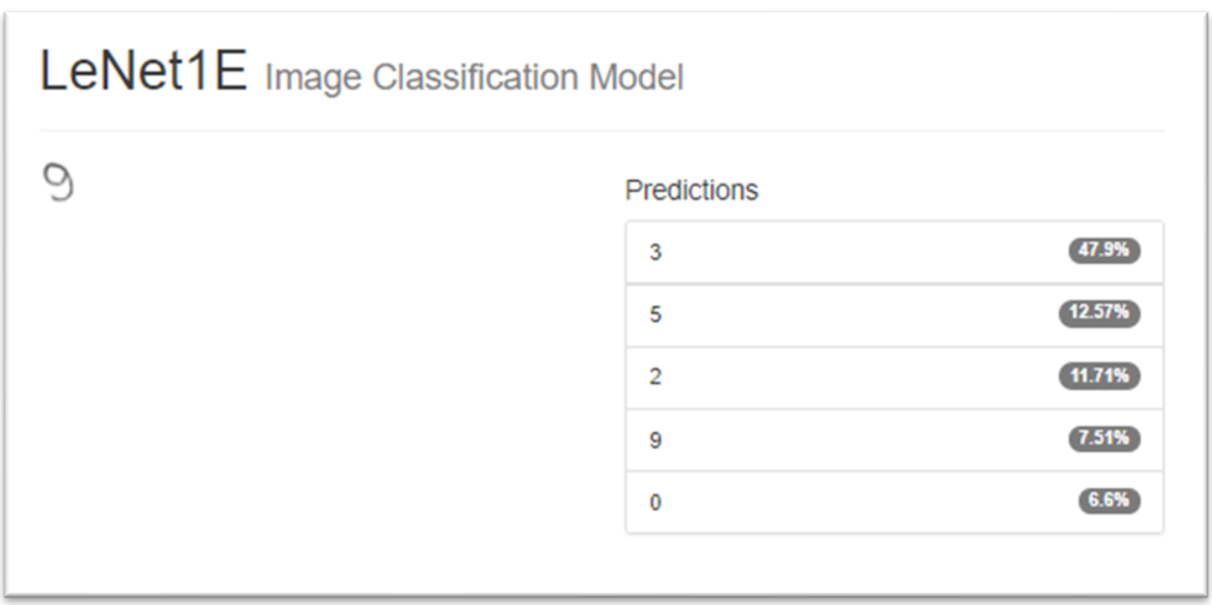


Image 4.3 uses the model from image 4.1, but is tasked with classifying the same hand drawn digit as image 3.2 displays.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Reflection on images 4.1-4.3:

Surprisingly, for the digit 3 from the testing set, the model was not significantly worse at correctly identifying the number as a 3. Which is likely due to the fact that much of the accuracy increase is seen in the first epoch of training, as well as the loss decrease. However, when the digit 9 from a google image search was tested the modeled was almost 50% certain that the digit was a 3, and even more surprisingly, the correct digit was precited as the 4th most likely digit contained in the image. This is likely due to the fact that while the largest accuracy gains and loss decreases are seen in the first epoch, it is likely that further epochs, within reason, increase the robustness of the predictive capabilities of the model, whereas a model trained on only a single epoch is somewhat brittle to inputs that vary significantly from its training data set, such as the left lean seen in this 9 as well as its higher native pixel count.

Image 4.4

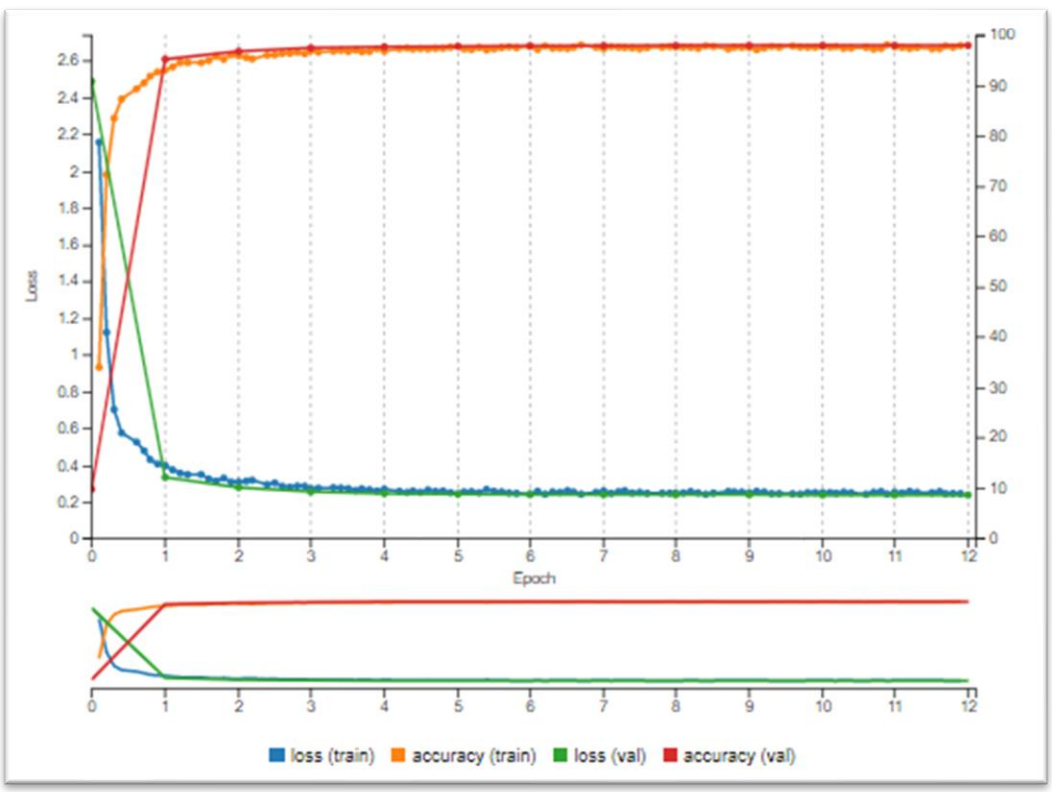


Image 4.4 displays the training loss plot of a new model trained in the same manner as image 4.1, but with a batch size of 64, and trained for 12 epochs.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 4.5

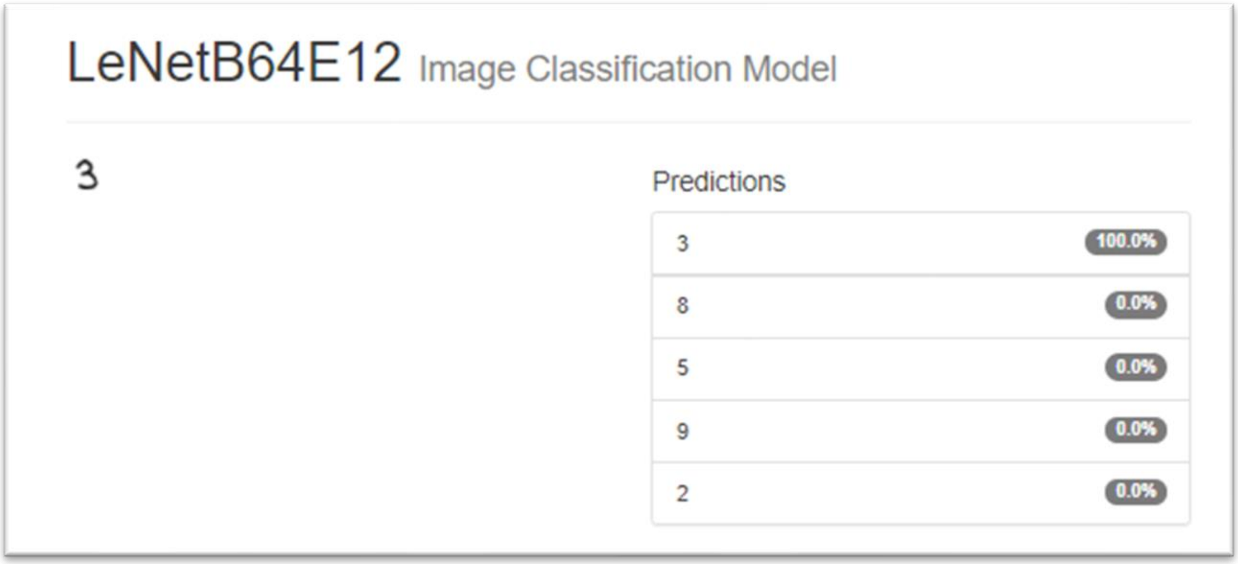


Image 4.5 shows the model from image 4.4, being tasked with identifying the same hand drawn digit as image 3.1.

Image 4.6

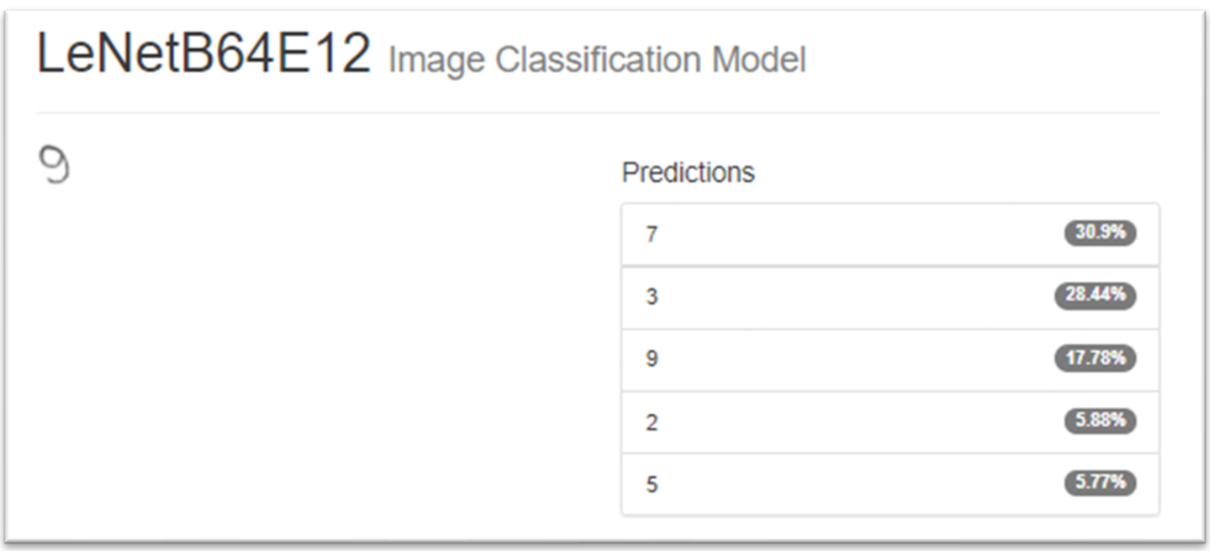


Image 4.6 shows the model from image 4.4, being tasked with identifying the same hand drawn digit as image 3.2.

Lab 7: Digits

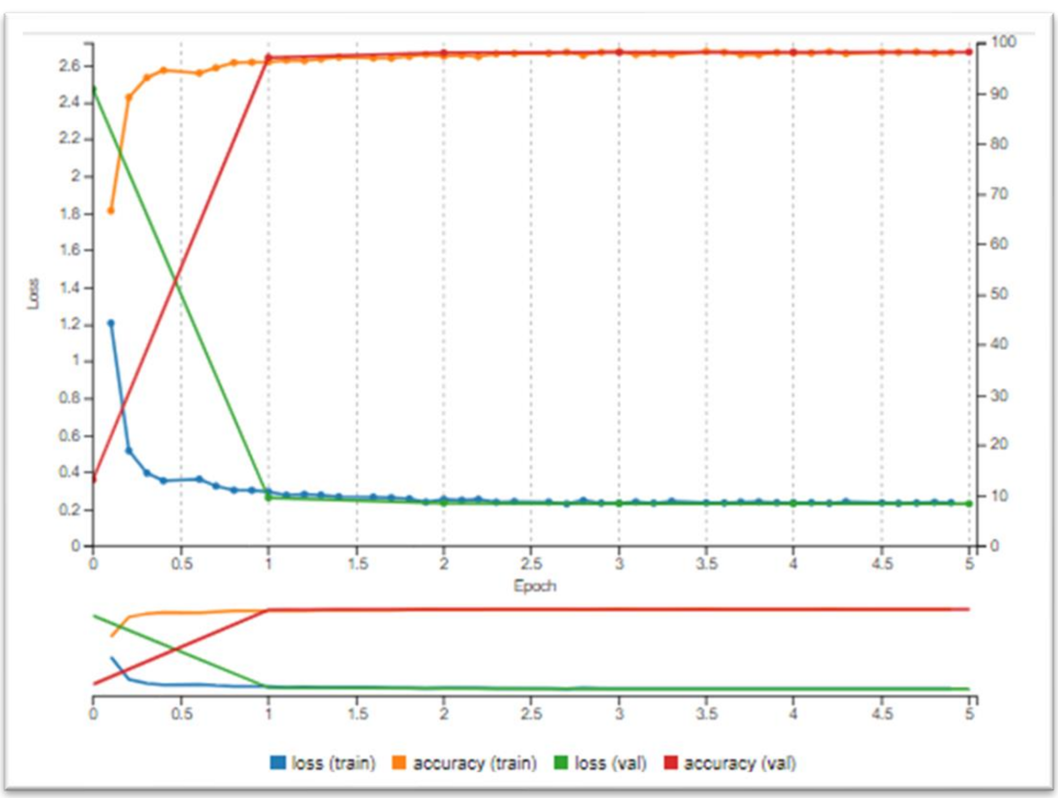
Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Reflection on images 4.4 – 4.6:

As seen in the training loss plot seen in image 4.4, when the batch size was increased to 64, the model needs several more epochs in order to reach the same level of accuracy as the model that uses batch sizes of 8 seen image 2.1. Overall, the effect on accuracy is negligible for this data particular data set, both models eventually reach the same training and validation accuracies near the 98% mark. A difference noted between the batch size of 64 and batch size of 8 was the GPU usage. Throughout the training of the model with a batch size of 64, the GPU peaked at 11% usage and remained near 8% for the duration of the training. However, the model with the batch size of 8 had GPU usage peak at 38% and remain around 25% usage for most of the training. The reason for this may be that the task of back propagating can be parallelized when batch sizes are greater than one, and as such when batch sizes are greater, the GPU can work more efficiently in a parallel manner. This hypothesis would also mean that if batch sizes were smaller, then the GPU would be forced to work in a more serialized manner, which it is less suited for, resulting in the high GPU usage seen for the lower batch size.

Part 5: Data Augmentation

Image 5.1



Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 5.1 represents the training loss plot of the same model as image 2.1, except for the fact that data set has been augmented by increasing the noise to a level of 0.05.

Image 5.2

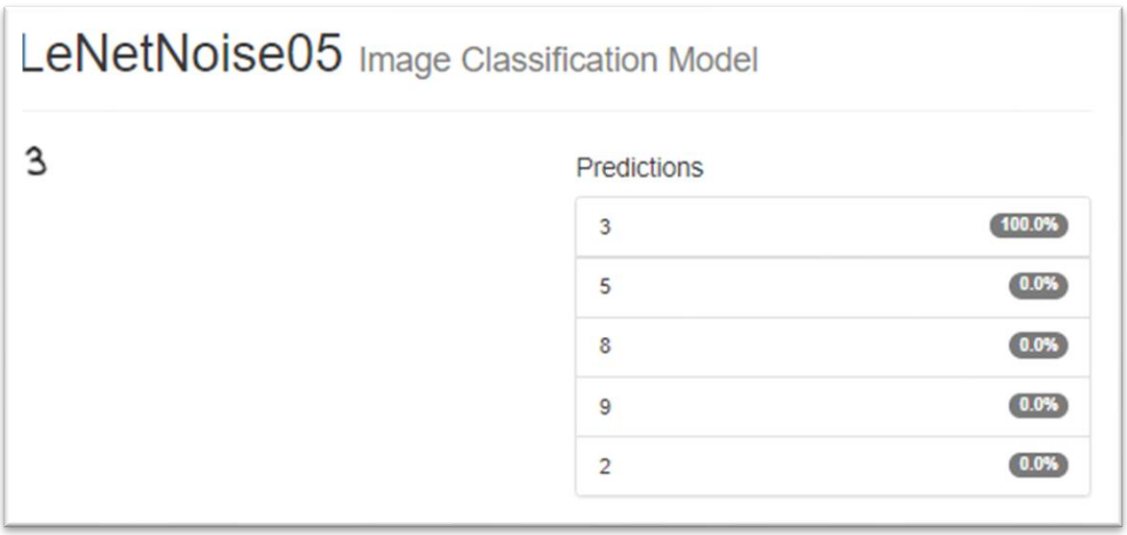


Image 5.2 shows the model from image 5.1, being tasked with identifying the same hand drawn digit as image 3.1.

Image 5.3

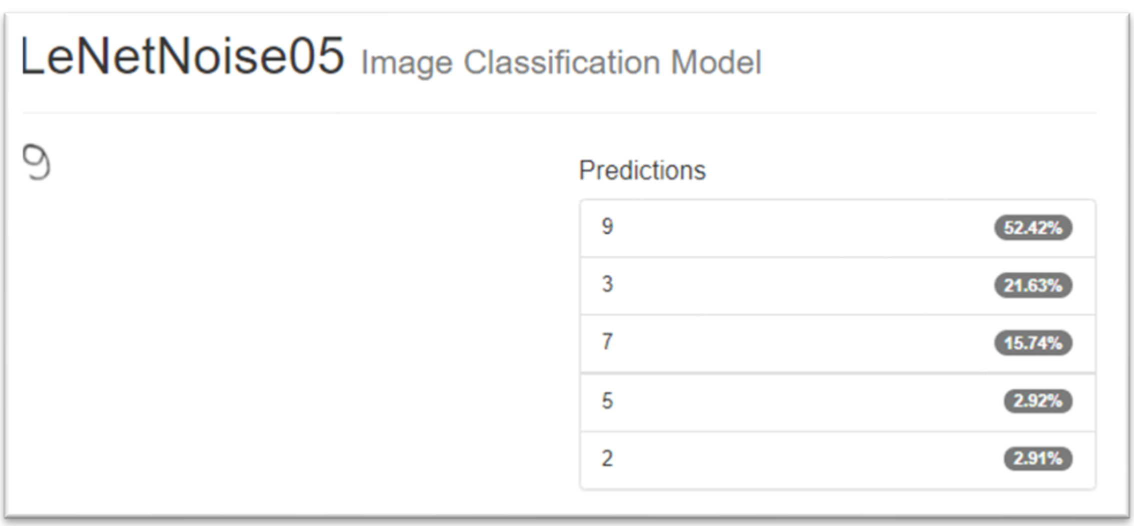


Image 5.3 shows the model from image 5.1, being tasked with identifying the same hand drawn digit as image 3.2.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 5.4

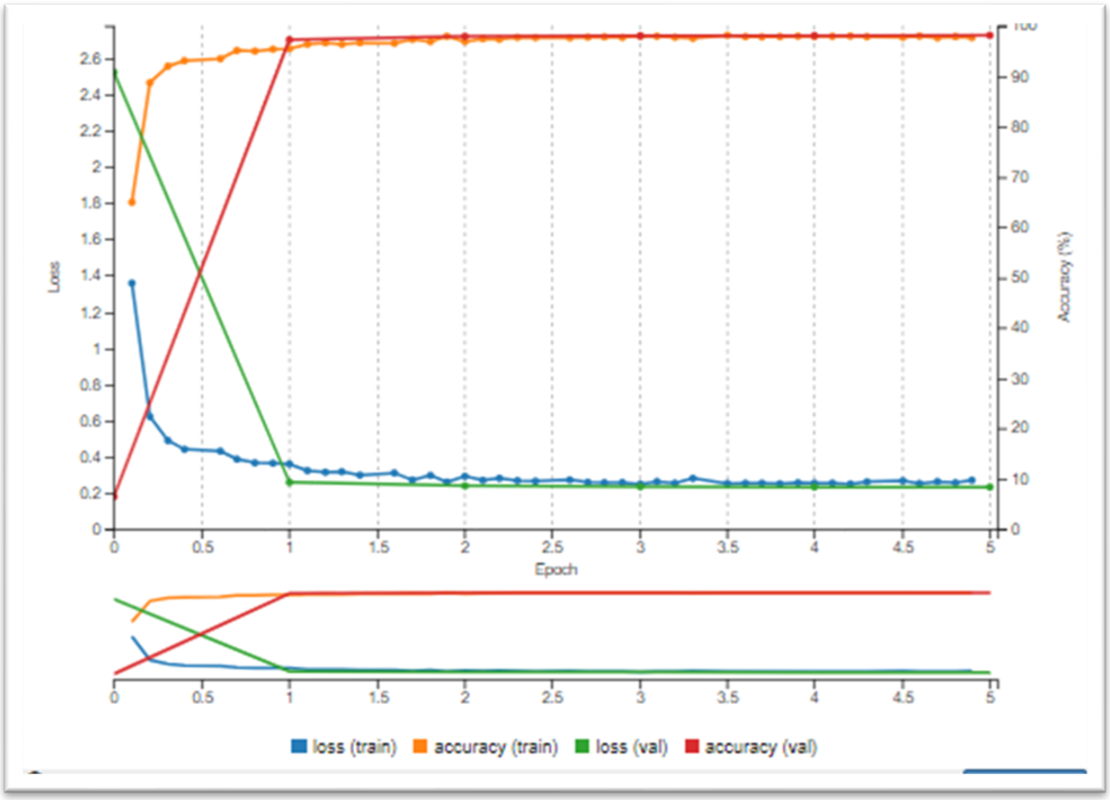


Image 5.4 represents the training loss plot of the same model as image 2.1, except for the fact that data set has been augmented by increasing the contrast to a level of 0.8.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 5.5

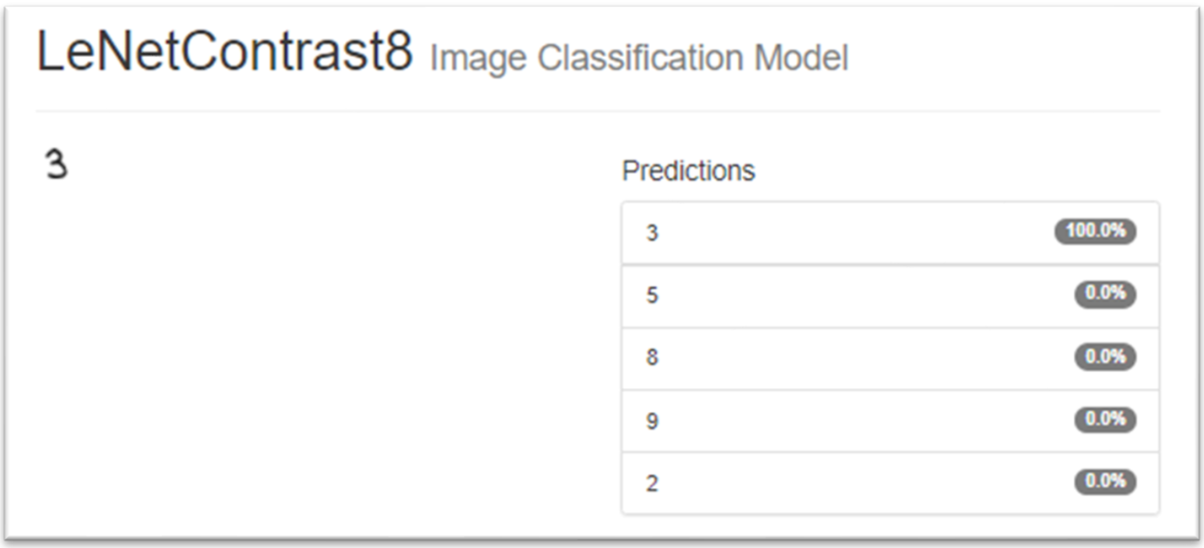


Image 5.5 shows the model from image 5.4, being tasked with identifying the same hand drawn digit as image 3.1.

Image 5.6

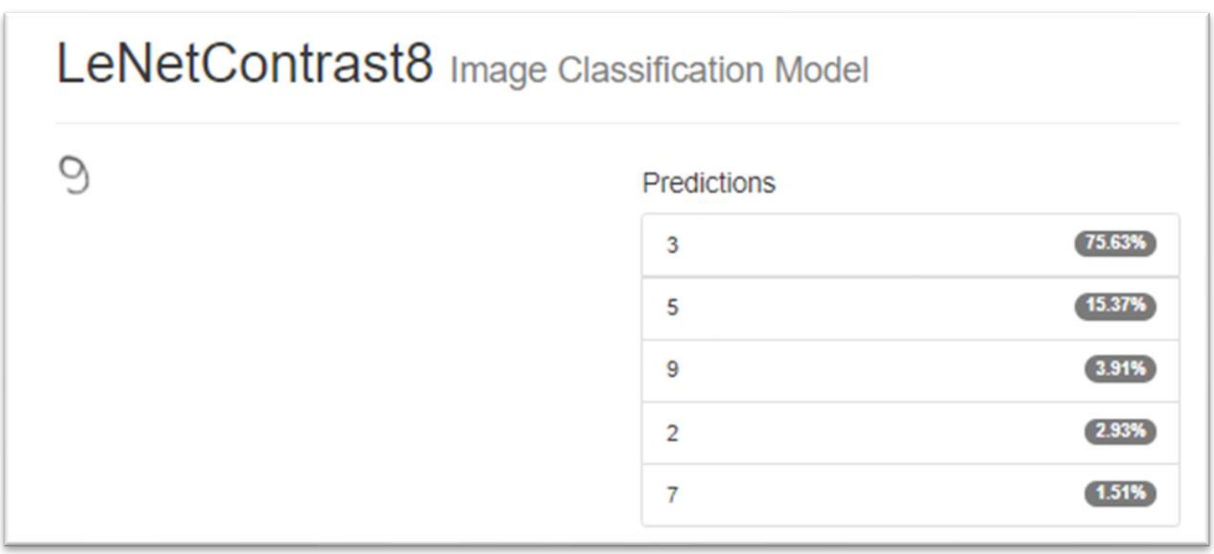


Image 5.5 shows the model from image 5.4, being tasked with identifying the same hand drawn digit as image 3.2.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Reflection on The Effect of Data Augmentation:

Overall, using data augmentation had effects consistent with what I was expecting. The reason for this is that noise essentially decreases the clarity of the image. Being that the data used is black and white, this reduces the amount of clean information that the model had to detect noticeable patterns in digits, reducing its reliance on the exact clarity of data that it was trained on. This trend is not noticeably seen in the models ability to detect a 3 from the native MNIST database, which is likely due to the fact that it was trained on similar data. However, when tested on the google image of a hand drawn 9, that decreased reliance on the exact composition of its training data set comes into use as the model sees an almost 20% increase in its ability to correctly identify the number 9. However, when setting the contrast to a level of 0.8, the model displayed a decreased ability to differentiate realistic data. The model with contrast set to a level of 0.8 only predicts the correct digit less than 4% of the time. This is likely due to the fact that this augmentation makes the model overly reliant on its native training data set, and as such, when the google image is attempted to be inferred, the model struggles with the data that it never received comparable training for.

<u>Part 6: Another Dataset</u>

Caltech 256 dataset description:

The Caltech 256 dataset is 2.59 GB in size, and contains 30,607 images which spans 257 object categories. Each category of images has a minimum of 80 unique images, a maximum of 827 unique images, and a mean of 119 unique images. Each image has a mean of 351 pixels, and the images range from a picture of an iPod, to a picture of Buddha.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

+Image 6.1

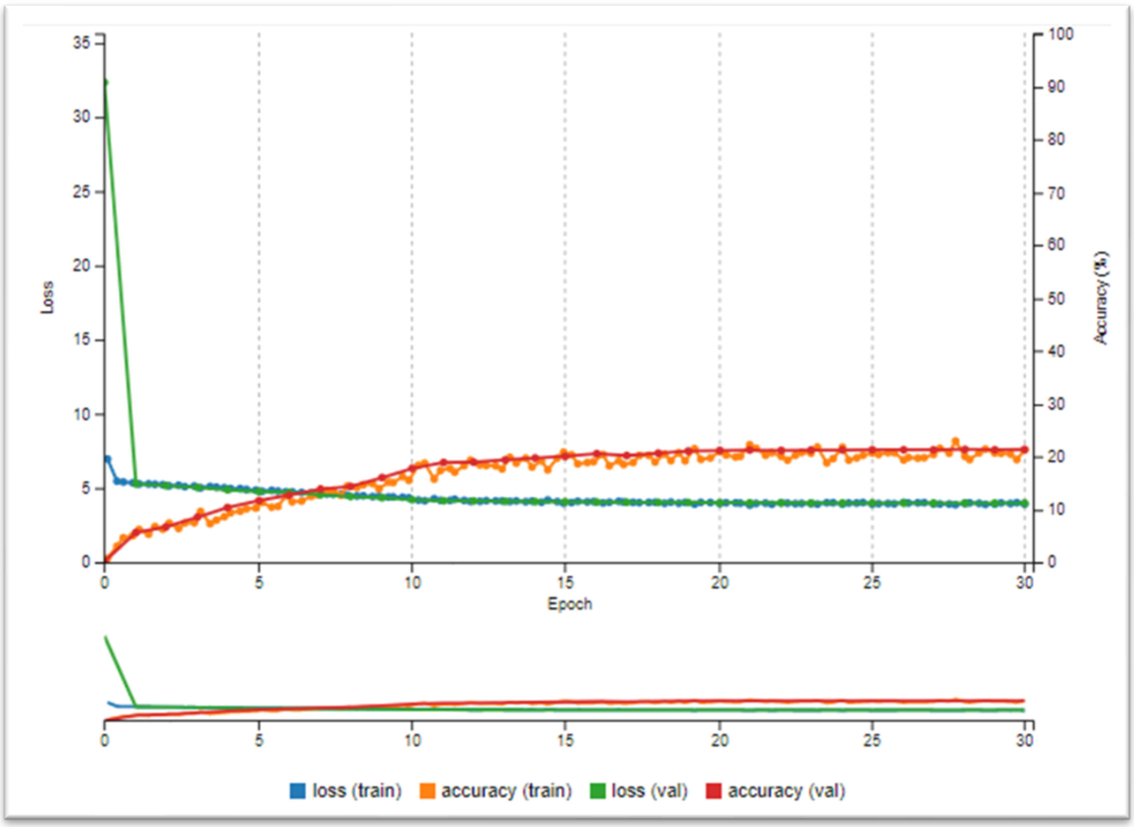


Image 6.1 shows the training loss plot of a model trained on the Caltech 256 dataset using an AlexNet network, with a batch size of 32, with a learning rate of .001, no data augmentations, and trained for 30 epochs. This model in total took 22:24 to train.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 6.2

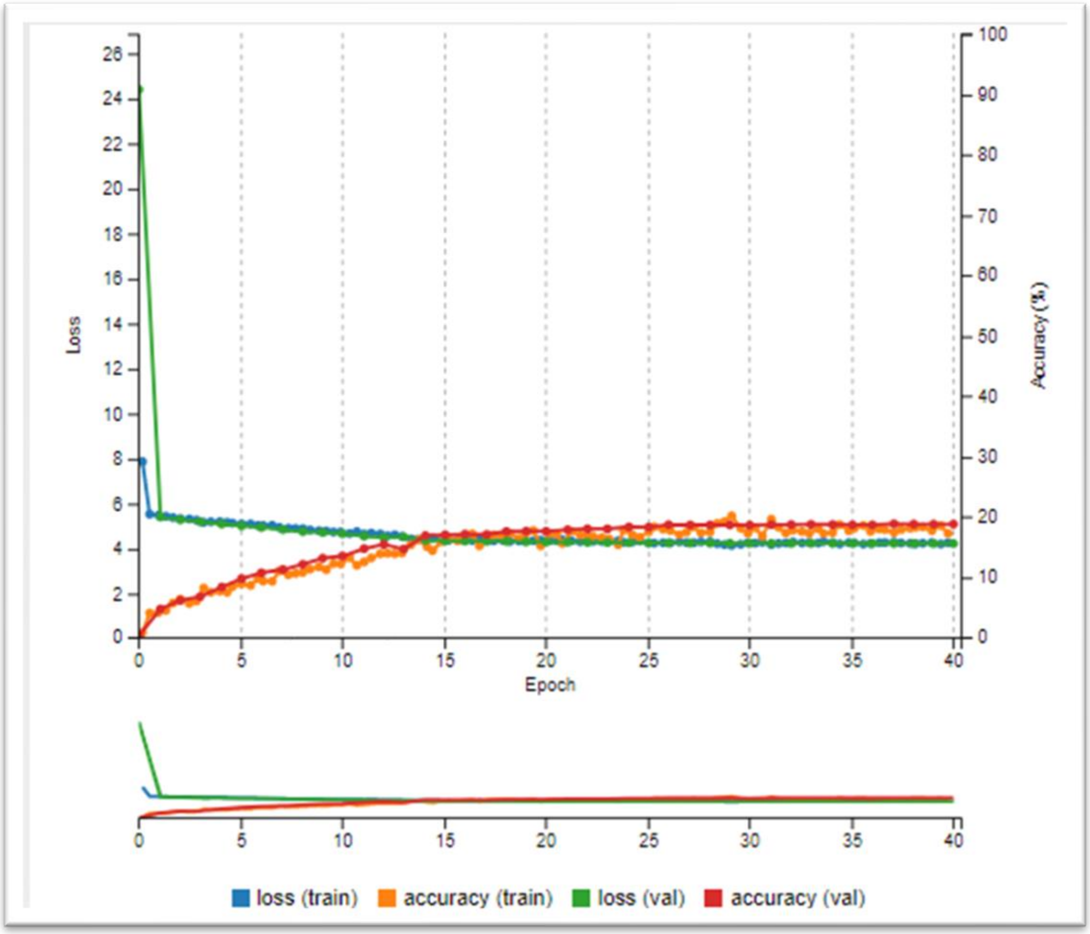


Image 6.2 shows the training loss plot of a model trained on the Caltech 256 dataset using an AlexNet network, with a batch size of 64, with a learning rate of .001, no data augmentations, and trained for 40 epochs. This model in total took 31:38 to train.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Image 5.3

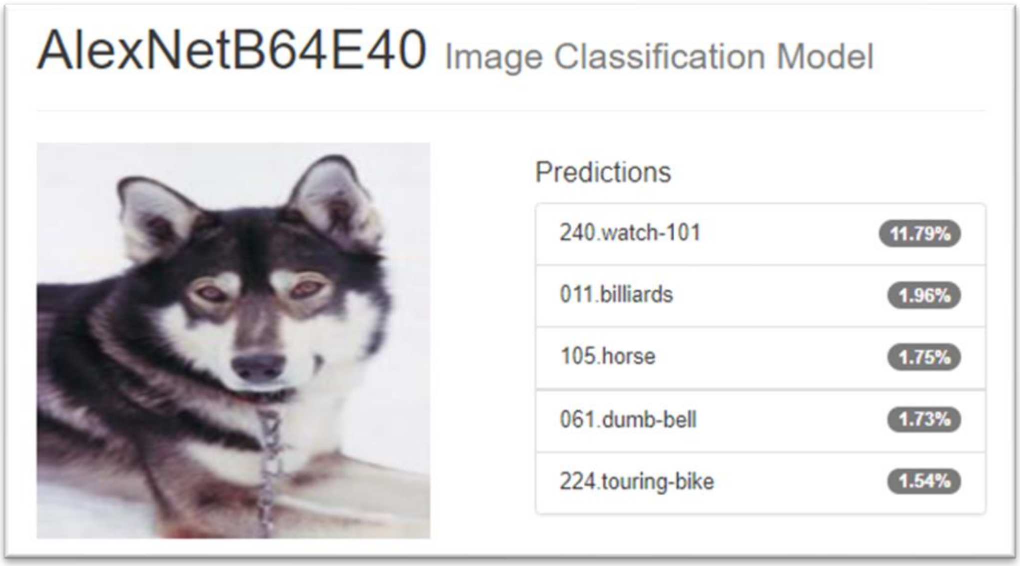


Image 5.3 shows the predictions of the model shown in image 5.2, when it is tasked with identifying the picture in the left side of the image.

Image 5.4

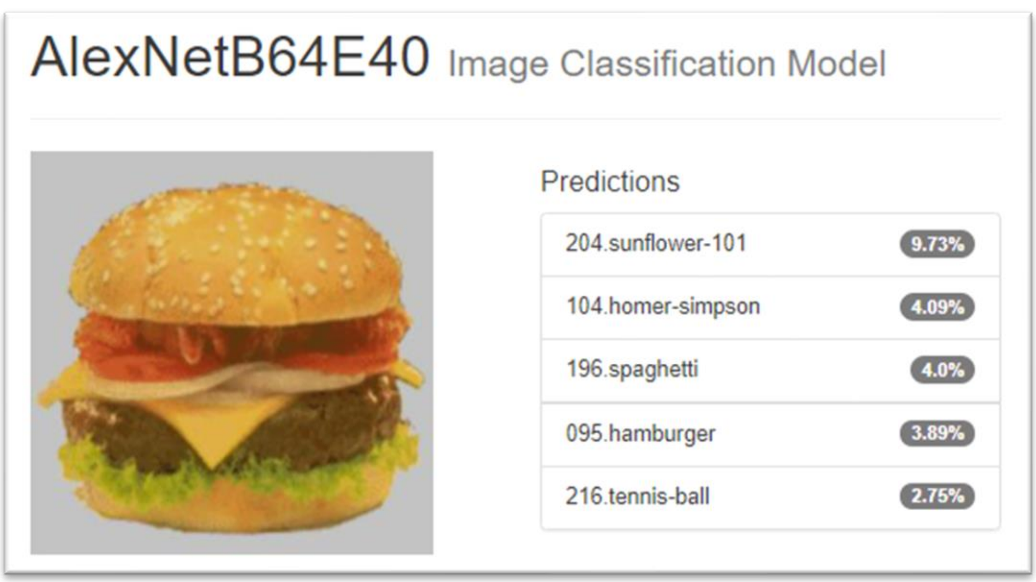


Image 5.4 shows the predictions of the model shown in image 5.2, when it is tasked with identifying the picture in the left side of the image.

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

Part 6 reflection:

While this section only shows the training loss plot for two different models, over a dozen models were attempted in order to find which hyper parameters could achieve the best training loss plot. However, many adjustments to the hyper parameters resulted in horrible accuracy, the model diverging, or overfitting occurred. The training loss plots shown above do not display any overfitting, as there is no clear trend where the training accuracy increases, and the validation accuracy decreases. However, overfitting was observed when whitening was used for data augmentation. In this model, the training accuracy reached 25%, but validation accuracy plummeted to almost 0%. Another trend that was observed was that by attempting to increase the learning rate to anything above 0.001, the model diverged almost immediately, but, when the learning rate was decreased, the model did not learn as quickly and resulted in a less accurate model. Furthermore, when most data augmentations were applied, the model responded with a deep decline in accuracy, even for the subtlest augmentations. This is likely due to the fact that the Caltech 256 dataset is already such a diverse collection of images, that training on the native images is challenge enough, and adding anymore augmentation makes it extremely hard to identify any patterns. In addition, by decreasing the batch size, the data tended to converge quicker, however, this approach led to longer training times and also had the effect of not reducing loss to anything much less than the accuracy percentage. Lastly, it was noticed that when increasing batch sizes, the model converged more slowly and also decreased the overall precision of the model. With that all being said, it was somewhat frustrating to realize that the recommended parameters of a batch size of 32, and a learning rate of 0.001, with no augmentations resulted in the best training loss plot. The lesson learned from this is that when creating models, there is often a “goldilocks” level for each hyper parameter, where increasing it too much makes the model suffer, and decreasing it too much results in the model suffering as well. In terms of the predictions made by the model from image 5.2, it was surprising how inaccurate the model was. For image 5.3, the image of the dog, the models best guess was that the image was a watch, with a certainty of about 11%. In fact, the model did not even have the correct classification in its top 5 predictions. As for the image of the hamburger, image 5.4, the resulting predictions were surprising as well. The reason for this is that the image of the hamburger is as stereotypical as a hamburger picture can be, yet, the model’s first prediction was that the image was a sunflower. However, the model was able to predict with 3.89% certainty that the image was in fact a hamburger.

<u>Conclusion:</u>

Lab 7: Digits

Course: CS2300
Author: Nigel Nelson
Date: 2/1/21

This lab was an exploratory exercise in deep neural networks. Specifically, this lab used Digits, the NVIDIA Deep Learning GPU Training System. The first dataset that a model was built for using Digits was the Mnist database. Through experimentation of the batch sizes used by the models trained on this data set, it was discovered that larger batch sizes required a greater number of epochs to attain a similar accuracy as smaller batch sizes. The tradeoff observed was that they are faster to train per epoch, but are less precision per each trained epoch. In addition, it was discovered that by using data augmentations, you can effect the accuracy of your model, specifically in the classification of realistic data, which in this case was data that came from outside the dataset used to train the model. However, it was observed that while some augmentations can improve the accuracy of the model for realistic scenarios, other augmentations can have a negative effect. Lastly, the Caltech 256 dataset was used to train a model that classified images, which had significantly lower accuracy levels when compared to the model trained on the Mnist database. By adjusting the hyper parameters, it was discovered that for an already diverse data set, data augmentation led to significant decreased accuracy. Lastly, a common theme was observed in this dataset that seemed to be prevalent throughout this entire lab, that hyperparameters have no definite right or wrong setting, and that there is often a level that is just right for a given application that can be found through experimentation.