Title Page

Assignment 4: Deep Learning

Melissa Hunfalvay

Email: Melissa.Hunfalvay@gmail.com

Data 640 9040

Spring 2022

Professor Steve Knode

Date: March 8th, 2022

Introduction

The dataset chosen was CIFAR-10 (Krizhevsky, 2009). The purpose of the analysis was to accurately identify objects in the images, such as a truck or car. To accomplish this purpose, a method of deep learning referred to as Convolutional Neural Networks (CNN), was employed.

The "real world" value of developing a model for this problem type may include:

a) Identification of images within a photo library, for example, finding images of a specific person or object from a past family holiday.
b) Identification of threatening objects at a security checkpoint, for example identification of a knife in a passenger bag via a TSA scanner.
c) Automatic identification of real-world images can also be used for real time analysis and training. Should we know the location of objects within a scene, we can redirect gaze to appropriate regions. One example may include redirecting an air traffic controllers' vision, to a "busy" region of the screen improving safety for travelers.

Inspired by nature and the human brain, deep learning models are designed to extract key features from an image to identify its contents (Panyam, 2017; Knode, 2022). There are several types of deep learning models, however, this analysis will use Convolutional Neural Networks (CNN) which are a form of neural network most often used for image processing (LeCun, Yann, et al., 1998).

Dataset

The CIFAR-10 dataset is a subset of 80 million images collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton (Figure 1, Krizhevsky, 2009). The images are labeled and in color. They consist of a total of 60,000 images, with ten categories, having 6,000 images per category. Some examples of the categories include images of birds, cats, trucks, ships. Each of the categories are mutually exclusive. The size of each image is 32 x 32-pixels.

The dataset is randomly divided into five training batches, each with 10,000 images (total training *n* = 50,000) and one test batch, with 1,000 images from each class (total test *n* = 10,000). Training and test images are equally distributed between classes, thereby representing the same number of horses, for example, in each training and test set.

Deep Learning Model Development

Convolutional Neural Networks (CNN's) are a form of deep learning that follow a process flow or architecture (see Figure 2 and Table 1).

Table 1: Convolutional Neural Network Architecture and Parameters Defined

	Input Layer: the image	**Feature Learning Layer: whereby the machine learns features of the image**		**Classification Layer: whereby the CNN predicts the image content**	
	Image	***Convolution-RELU***	***Pooling***	***Flatten***	***Fully Connect***
Defining role/purpose	An image is a series of pixels which can be understood in terms of their pixel values from 0 to 255 and spatial relationships within the image.	Purpose to extract features from the input image by using the pixel matrix to preserve spatial relationships. Resulting in a feature map, then transforming that map using a ReLU function.	Purpose to reduce the dimensionality of the features and extract only the most important features. This step reduces "noise" in the image.	Purpose to convert data into a 1-dimensional array for inputing into the next layer. Creates a single long feature vector in which all the pixel data is in one line to make connections with the final output layer (Jong, 2019)	Purpose to form a multi layer perceptron. Every neuron in the pervious layer is connected to every neuron in the next layer.
Parameters	Image of dog, cat, horse, truck, car…	Various filtering parameters or "kernels" extract features of size, depth, stride and padding.	Spatial pooling can have three different types: maximum, average, sum.	Flattens the input, not the batch size (Keras, n.d.)	Purpose of *Softmax* is as an activation function that takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one and form a total of one, i.e. a probability distribution.
Key parameter based metrics	*Channel:* three channels for a standard color image, 2 for a grayscale image.	*Kernel size:* refers to the size of the scan or "window" across the image	*Maximum (max) pooling:* takes the largest element from the rectified feature map within that window.	*Dense Units:* are positive integers of the dimensionality of the output space (Keras, n.d.)	*Accuracy:* calculates how often the prediction equals the lables (Kera, n.d.)
		Depth: refers to the number of filters or layers used for the convolution problem (Ujjwalkarn, 2016)	*Average Pooling:* takes the average element from the rectified feature map within that window.		*Loss:* computes the quatitiy that the model should seek to minimize during training (Keras, n.d.)
		Stride: refers to the number of pixels by which the filter slides across the image	*Sum pooling:* takes the sum of all the elements in that window.		
		Padding: refers to the number of zeros around the border of the filter.	*Epochs:* number of times the models passes through the whole training data.		
		The ReLU (*Rectified Linear Unit Layer*) is used to introduce non-linearity by replacing all the negative pixel values in the feature map to zero.	*Batch normalization:* normalizes the values across examples and within the image in order to ensure no outliers are within the data.		
			Dropout: refers to the percentage of neurons it pauses updates on to prevent overfitting and helps the model generalize.		
			Batch size: is the number of examples the model will look at before it goes back to update the weights.		

For this specific project, these steps were implemented in Google Colab (Google Colab) using a Python script (see Table 2).

Table 2: Python Script Steps for creation of Convolutional Neural Network

CNN Layer	Step	Description	Sample Code
Input	1	Import libraries for math, data science e.g. tensorflow.keras, pandas	import numpy as np
Input	2	Import the data needed for th analysis i.e. CIFAR-10	(train_x, train_y), (test_x, test_y) = cifar10.load_data()
Input	3	Import Kera parts needed to build the network e.g. pooling, dropout, sequential model type	from tensorflow.keras.models import Sequential
Feature Learning	4	Define model type as sequential	model = Sequential()
Feature Learning	5	Build the network	
Feature Learning	5.a	Define the input shape (one time)	input_shape = (32, 32, 3)
Feature Learning	5.b	Define filter size	filters = 128
Feature Learning	5.c	Define kernal size	kernel_size = (2, 2)
Feature Learning	5.d	Define dropout rate	model.add(Dropout(0.2))
Feature Learning	5.e	Define batch normalization	model.add(BatchNormalization())
Feature Learning	5.f	Repeat 5.b to 5.f to create the number of layers in the model	
Classification	6	Flatten the input	model.add(Flatten())
Classification	7	Define the dense units	model.add(Dense(units = 10,
Classification	8	Active the softmax function	activation = 'softmax'))
Classification	9	Compile the network to output the results to show accuracy and loss scores of the training data set	model.fit(train_x, train_y,
Classification	10	Fit the model with the number of iterations (epochs)	epochs = 5
Classification	11	Fit the model with the batch size	batch_size = 128
Classification	12	Run the model on the test data	model.fit(test_x, test_y,
Classification	13	Compile the network to output the results to show accuracy and loss scores of the test data set	score = model.evaluate(test_x, test_y)

Using Google Colab various architectural changes and parameters were experimented with to create twelve new CNN models (Table 3). Note, the highlighted blue cells show changes from one model to the next.

Table 3: Convolutional Neural Network Models Described

Model Characteristics										Rational for Changes
Model Name	Layers: #	Number of Filters	Fliters: Size(s)	Kernel: Size	Pooling: Type	Pooling: Size	Dropout	Epochs: #	Batch: Size	Explain the "why" behind model changes.
Sequential 1 (Baseline)	1	1	128	2,2	Max	2,2	0.2	10	128	This is the baseline model given in the script. My goal overall, was to examine how changes in the parameters affected the initial model. Then, to see if I could improve the accuracy.
Sequential 2	2	2	128, 64	2,2	Max	2,2. 2,2	0.2	5	128	Reduced the epochs,i.e. the number of times the model passes through the training data. My expectation would be this would reduce accuracy as there are fewer opportunities for the model to learn (Karani, 2020)
Sequential 3	3	3	128,64,32	2,2	Max	2,2. 2,2. 2,2	0.2	10	128	Layers were added to this model in order to increase the number of weights in the network. Epochs were returned to baseline so as to isolate layer changes and see their effect.
Sequential 4	3	3	128,64,32	4,4. 3,3. 2,2.	Max	2,2	0.2	10	128	Progressively reduced the kernel and filter sizes. Rational was to see if the smaller kernel size improved accuracy as the model learned.
Sequential 5	3	3	128,64,32	4,4. 4,4. 4,4.	Max	2,2	0.2	10	128	Kept kernel size large at 4, but reduced the filter size across the layers. Rationale was to reduce the size of the receptive field to see if this focus would improve accuracy per Han et. al (2020) experimental with facial recognition.
Sequential 6	1	1	128	4,4	Max	4,4	0.1	50	32	Increased the size of the pooling. Increasing the pooling size increases the complexity of the image. It would be expected to reduce accuracy and increase relative training time. Reduced the dropput rate which may lead to overfitting.
Sequential 7	1	1	64	4,4	Average	2, 2	0.2	50	16	Changed the pooling to average. Changed the batch size. This decreases the number of examples (and inversely increases the "check rate") the model will look at before it goes back to update the weights.
Sequential 8	2	2	128, 128	2,2. 4,4	Max, avg	2,2. 2,2	0.2	100	16	Doubles the number of epochs. Increasing the number of times the model passes through the training data should increase the accuracy.
Sequential 9	2	2	128, 128	2,2. 4,4	Max, avg	2,2. 2,2	0.2	10	10	Futher reduced the batch size in order to force the model to check the results more often, ideally looking to increase accuracy.
Sequential 10	2	2	128, 128	2,2. 4,4	Max, avg	2,2. 2,2	0.2	10	5	Futher reduced the batch size in order to force the model to check the results more often, ideally looking to increase accuracy.
Sequential 11	2	2	128, 128	2,2. 4,4	Max, avg	2,2. 2,2	0.2	20	5	Kept the batch size very low and increased the number of times the model read the whole training data (epochs = 20). Expect increased accuracy and increased time for the model to run.
Sequential 12	2	2	128, 128	2,2. 4,4	Max, avg	2,2. 2,2	0.2	100	5	Kept batch size very low. Kept epochs very high. Expect best accuracy as the model has many chances to learn. However, also expect longest training time.

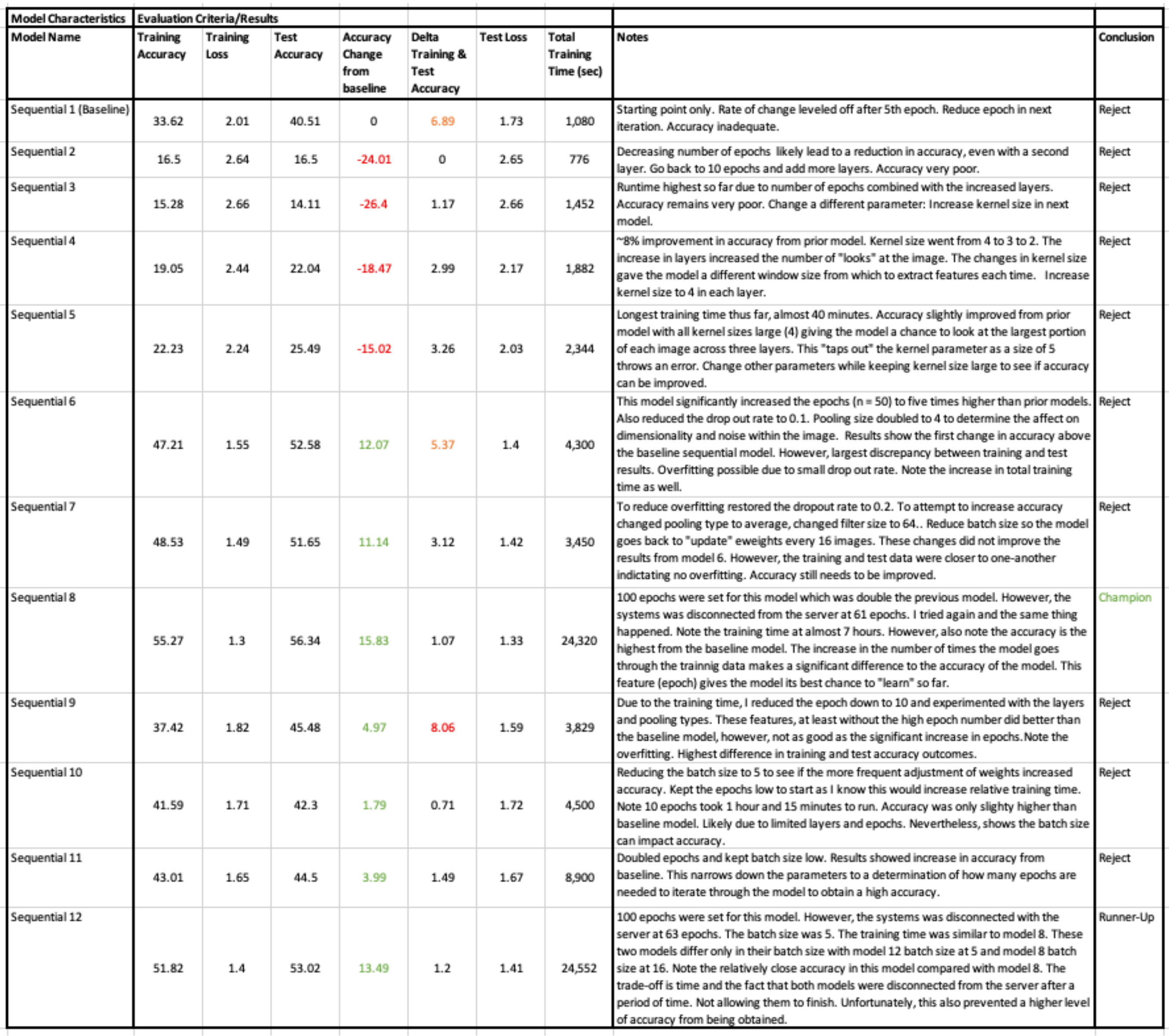
Accuracy Measures and Results

The purpose of this assessment was to correctly identify the object within an image. Two key metrics were used to examine each model created. The first was accuracy, measured as a percentage of correct identification of the object within the image. A high accuracy was a better result. Second was loss, which computes the quantity that the model should seek to minimize during training (Keras, n.d.). A lower loss metric is a better result. A final metric for consideration is the time to complete the analysis. Given that these are very large files with complex deep learning models, the time to complete the analysis can be significant and may become a consideration in the adoption of the results. Table 4 summarizes these evaluation metrics and interprets the results in the notes section.

Accuracy ratings increase with number of epochs for each model (Figure 4). However, it is evident that a higher rate of change in accuracy occurs early in the epoch cycles, especially in the first ten cycles (Figure 4). Furthermore, accuracy after the first cycle is also a good indicator of the relative accuracy across models.

The final accuracy measure of each model is shown in Figure 5. Model 8 slightly edged out model 12 showing accuracy ratings of 56.34 and 53.02 respectively (Figure 6). Model 8 improved accuracy by more than 15% compared with the baseline model (model 1).

Table 4: Results and Interpretation of Convolutional Neural Networks



Model Characteristics	Evaluation Criteria/Results								
Model Name	Training Accuracy	Training Loss	Test Accuracy	Accuracy Change from baseline	Delta Training & Test Accuracy	Test Loss	Total Training Time (sec)	Notes	Conclusion
Sequential 1 (Baseline)	33.62	2.01	40.51	0	6.89	1.73	1,080	Starting point only. Rate of change leveled off after 5th epoch. Reduce epoch in next iteration. Accuracy inadequate.	Reject
Sequential 2	16.5	2.64	16.5	-24.01	0	2.65	776	Decreasing number of epochs likely lead to a reduction in accuracy, even with a second layer. Go back to 10 epochs and add more layers. Accuracy very poor.	Reject
Sequential 3	15.28	2.66	14.11	-26.4	1.17	2.66	1,452	Runtime highest so far due to number of epochs combined with the increased layers. Accuracy remains very poor. Change a different parameter: Increase kernel size in next model.	Reject
Sequential 4	19.05	2.44	22.04	-18.47	2.99	2.17	1,882	~8% improvement in accuracy from prior model. Kernel size went from 4 to 3 to 2. The increase in layers increased the number of "looks" at the image. The changes in kernel size gave the model a different window size from which to extract features each time. Increase kernel size to 4 in each layer.	Reject
Sequential 5	22.23	2.24	25.49	-15.02	3.26	2.03	2,344	Longest training time thus far, almost 40 minutes. Accuracy slightly improved from prior model with all kernel sizes large (4) giving the model a chance to look at the largest portion of each image across three layers. This "taps out" the kernel parameter as a size of 5 throws an error. Change other parameters while keeping kernel size large to see if accuracy can be improved.	Reject
Sequential 6	47.21	1.55	52.58	12.07	5.37	1.4	4,300	This model significantly increased the epochs (n = 50) to five times higher than prior models. Also reduced the drop out rate to 0.1. Pooling size doubled to 4 to determine the affect on dimensionality and noise within the image. Results show the first change in accuracy above the baseline sequential model. However, largest discrepancy between training and test results. Overfitting possible due to small drop out rate. Note the increase in total training time as well.	Reject
Sequential 7	48.53	1.49	51.65	11.14	3.12	1.42	3,450	To reduce overfitting restored the dropout rate to 0.2. To attempt to increase accuracy changed pooling type to average, changed filter size to 64.. Reduce batch size so the model goes back to "update" eweights every 16 images. These changes did not improve the results from model 6. However, the training and test data were closer to one-another indictating no overfitting. Accuracy still needs to be improved.	Reject
Sequential 8	55.27	1.3	56.34	15.83	1.07	1.33	24,320	100 epochs were set for this model which was double the previous model. However, the systems was disconnected from the server at 61 epochs. I tried again and the same thing happened. Note the training time at almost 7 hours. However, also note the accuracy is the highest from the baseline model. The increase in the number of times the model goes through the trainnig data makes a significant difference to the accuracy of the model. This feature (epoch) gives the model its best chance to "learn" so far.	Champion
Sequential 9	37.42	1.82	45.48	4.97	8.06	1.59	3,829	Due to the training time, I reduced the epoch down to 10 and experimented with the layers and pooling types. These features, at least without the high epoch number did better than the baseline model, however, not as good as the significant increase in epochs.Note the overfitting. Highest difference in training and test accuracy outcomes.	Reject
Sequential 10	41.59	1.71	42.3	1.79	0.71	1.72	4,500	Reducing the batch size to 5 to see if the more frequent adjustment of weights increased accuracy. Kept the epochs low to start as I know this would increase relative training time. Note 10 epochs took 1 hour and 15 minutes to run. Accuracy was only slighty higher than baseline model. Likely due to limited layers and epochs. Nevertheless, shows the batch size can impact accuracy.	Reject
Sequential 11	43.01	1.65	44.5	3.99	1.49	1.67	8,900	Doubled epochs and kept batch size low. Results showed increase in accuracy from baseline. This narrows down the parameters to a determination of how many epochs are needed to iterate through the model to obtain a high accuracy.	Reject
Sequential 12	51.82	1.4	53.02	13.49	1.2	1.41	24,552	100 epochs were set for this model. However, the systems was disconnected with the server at 63 epochs. The batch size was 5. The training time was similar to model 8. These two models differ only in their batch size with model 12 batch size at 5 and model 8 batch size at 16. Note the relatively close accuracy in this model compared with model 8. The trade-off is time and the fact that both models were disconnected from the server after a period of time. Not allowing them to finish. Unfortunately, this also prevented a higher level of accuracy from being obtained.	Runner-Up

Loss rates for these models were also lowest compared with the other models (Table 4: Model 8: 1.33; Model 12: 1.41, Figure 8). Neither of these models were overfit as indicated by the difference in training and test data (Model 8: 1.07%; Model 12: 1.2%). Therefore, the winning model is model 8 as it reached the highest level of accuracy and lowest loss rate (Figure 7). The runner up is model 12 (Figure 8).

Conclusions and Takeaways

The purpose of the analysis was to accurately identify objects in images, such as a truck or car using a method of deep learning referred to as Convolutional Neural Networks (CNN). Model 8 was significantly improved from the baseline model (Model 1) by 15.83% resulting in a total accuracy of 56.34% in the test dataset. Loss rate was 1.33 (Table 4, Figure 8). Although an improvement this was still not a result that could be employed in a production environment.

The challenges to improving the models were interrelated. The two major challenges included the time to process the models and due to the extended time, being disconnected with the server. In a catch-22, it was observed that what improved accuracy also significantly increased processing time. The number of epoch and batch size were the most important elements to improve accuracy. Epochs, enable the model to "learn" by passing the parameters through the entire training data set. Batch size is the number of examples the model will look at before it updates the weights. Both these parameters were key to improving accuracy as these parameters effectively give the model more "looks" at the data to make corrections. However, they were also key in increasing the time the model took to run. Model 8 took 24,320 seconds and model 12 took 24,552 seconds. This equates to almost 7 hours for each model!

Each model was run twice to see if it could progress beyond the 61 and 63 epochs that were run. Curiously the models stopped at the same point and around the same processing time. After further investigation, it was realized that using the free version of the software, the runtimes disconnected as resources are not guaranteed in this version (FAQ's). This limits the runtimes available. Furthermore, the use of one GPU slowed the processing. Finally, once the connection was lost, there was no way to restart the processing from the point it was disconnected, requiring the process to start over at the first epoch. One positive note however was the processing was saved prior to the disconnection.

To remedy these challenges and to provide recommended improvements for future development include:

1. Google Colab has two paid versions (Colab Pro and Colab Pro +) which have faster GPUs, more memory, and longer runtimes (Signup). I would certainly take advantage of this for future projects.
2. For future iterations of this project, I was curious as to where the accuracy would be high enough to warrant an acceptable production model. Therefore, "projections" were made whereby the model would reach 100% accuracy (Figures 9 and 10) and 0% loss (Figures 11 and 12). Using a linear and exponential function the results were plotted in excel and projected the pathways. This could serve two purposes in the future:
    a. To determine the number of epochs needed prior to running the model
    b. To determine which version of Google Colab would be most appropriate to use

Although not a perfect solution, these projections revealed some interesting findings. For the winning model (Model 8) to reach 100% accuracy using the linear projection would require 153 epochs with the current model parameters including a low batch size of 16 (Figure 9). Using the exponential function this would require 106 epochs. To reach 0% loss would require 196 epochs via the linear function and 500 epochs via the exponential function (Figure 12). These findings provide guidance for future development toward a production accepted model of this dataset.

In summation, the baseline model was significantly improved and future steps for additional improvements were outlined to identify the objects more accurately within the images.

References

Google Colaboration (n.d.). *Purchasing plans.* Retrieved on March 7th, 2022, from: https://colab.research.google.com/signup

Google Colaboration (n.d.). *Frequently Asked Questions.* Retrieved on March 7th, 2022, from: https://research.google.com/colaboratory/faq.html#resource-limits

Jong, J. (2019). *The Most Intuitive and Easiest Guide for Convolutional Neural Network.* Retrieved on February 24th, 2022 from: https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-convolutional-neural-network-3607be47480

Karani, D. (2020). *Experiments on Hyperparameter Tuning in Deep Learning – Rules to Follow.* Retrieved on February 28th, 2022, from: https://towardsdatascience.com/experiments-on-hyperparameter-tuning-in-deep-learning-rules-to-follow-efe6a5bb60af

Knode, S. (2022). *Introduction and Overview of Deep Learning.* Retrieved February 24th, 2022 from: https://learn.umgc.edu/d2l/le/content/627222/viewContent/25080888/View

Krizhevsky, A. (2009). *Learning Multiple Layers of Features from Tiny Images.* Retrieved February 24th, 2022 from: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

LeCun, Yann, et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*.11: 2278-2324.

Panyam, V. (2017). *Deep Learning Made Simple [Part 1].* Retrieved February 24th, 2022 from: https://blog.vivekpanyam.com/

Ujjwalkarn. (2016). *An Intuitive Explanation of Convolutional Neural Networks.* Retrieved on February 24th, 2022, from: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Appendix

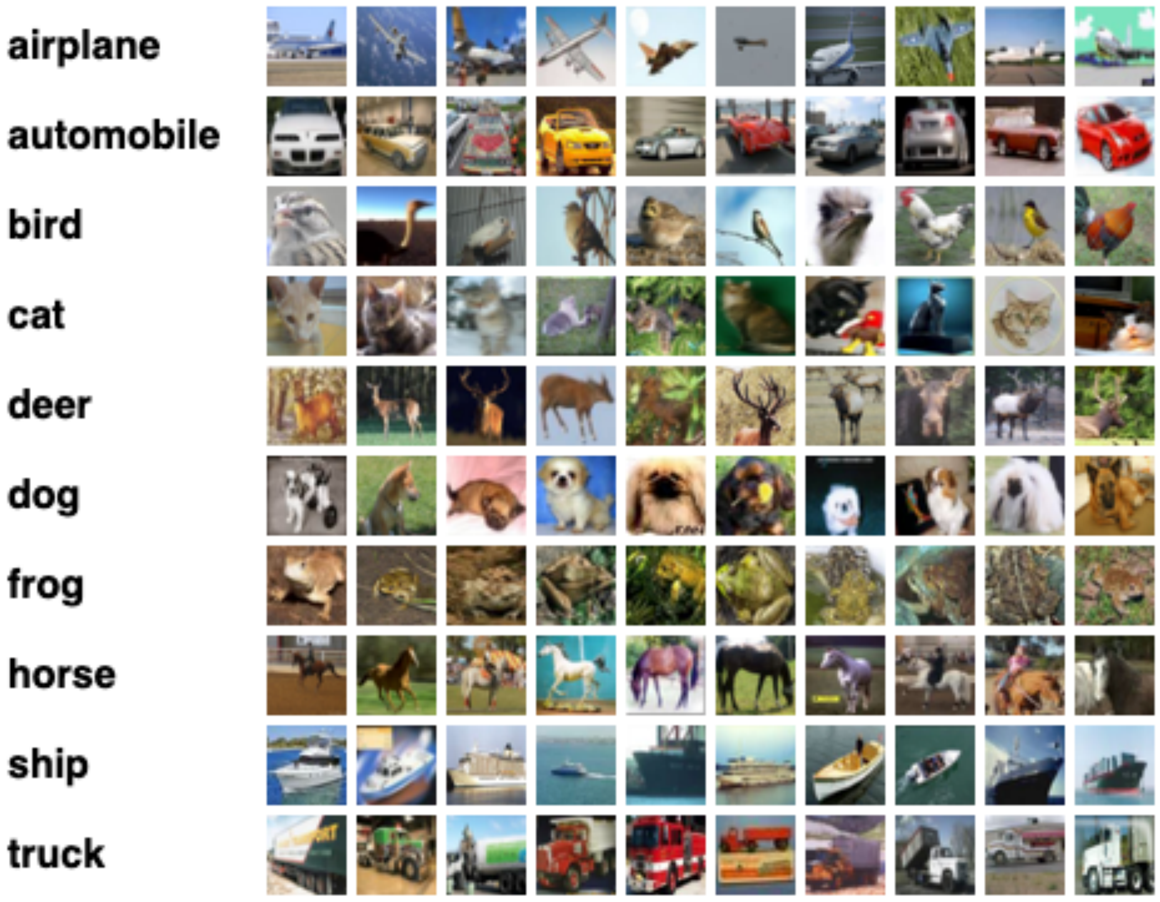


Figure 1: Screenshot of the data with all relevant class variables

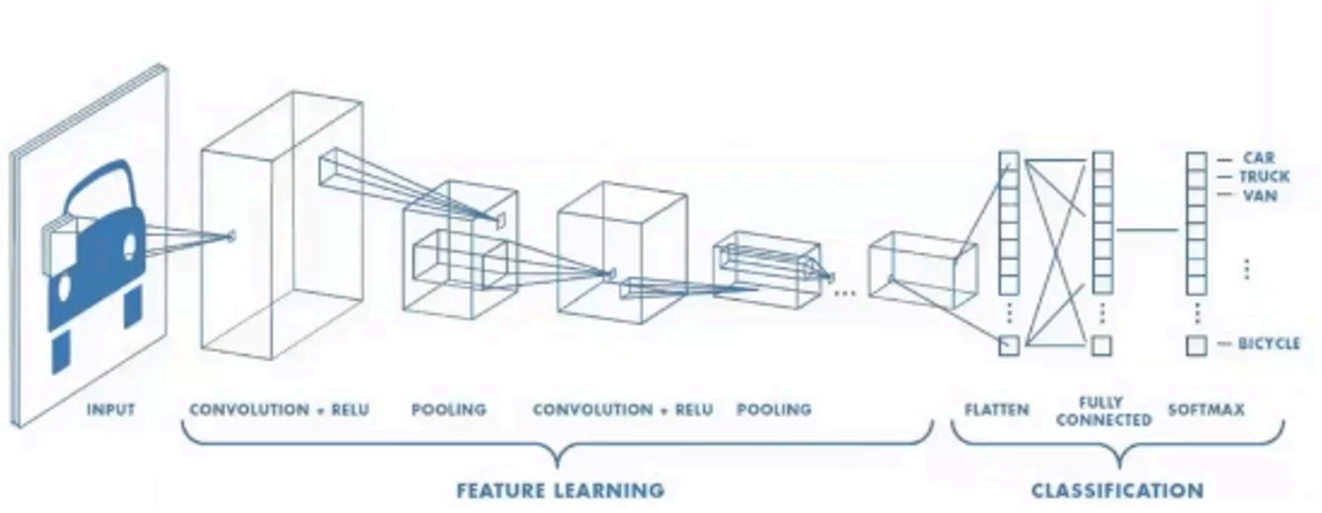
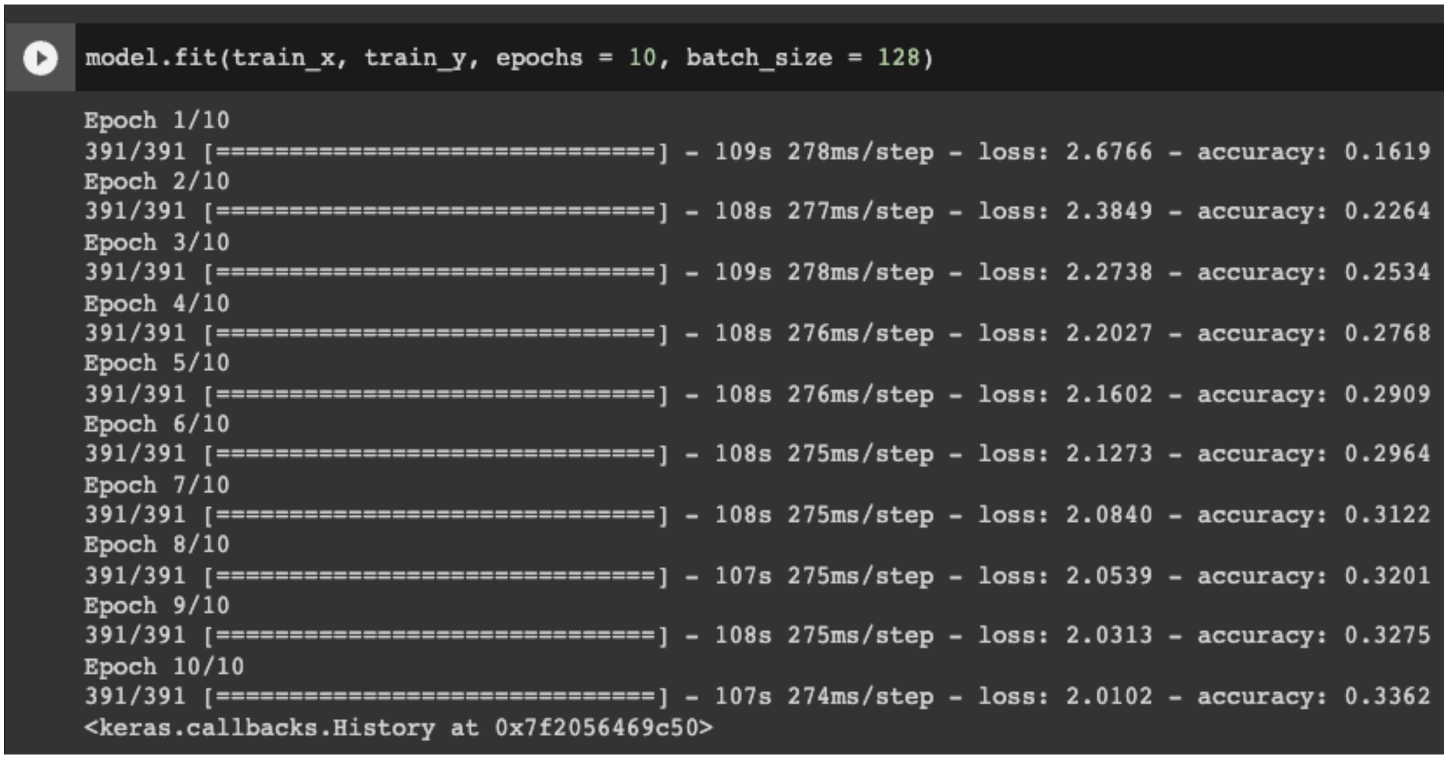


Figure 2: Sample architecture of a Convolutional Neural Network (CNN)



```
model.fit(train_x, train_y, epochs = 10, batch_size = 128)

Epoch 1/10
391/391 [==============================] - 109s 278ms/step - loss: 2.6766 - accuracy: 0.1619
Epoch 2/10
391/391 [==============================] - 108s 277ms/step - loss: 2.3849 - accuracy: 0.2264
Epoch 3/10
391/391 [==============================] - 109s 278ms/step - loss: 2.2738 - accuracy: 0.2534
Epoch 4/10
391/391 [==============================] - 108s 276ms/step - loss: 2.2027 - accuracy: 0.2768
Epoch 5/10
391/391 [==============================] - 108s 276ms/step - loss: 2.1602 - accuracy: 0.2909
Epoch 6/10
391/391 [==============================] - 108s 275ms/step - loss: 2.1273 - accuracy: 0.2964
Epoch 7/10
391/391 [==============================] - 108s 275ms/step - loss: 2.0840 - accuracy: 0.3122
Epoch 8/10
391/391 [==============================] - 107s 275ms/step - loss: 2.0539 - accuracy: 0.3201
Epoch 9/10
391/391 [==============================] - 108s 275ms/step - loss: 2.0313 - accuracy: 0.3275
Epoch 10/10
391/391 [==============================] - 107s 274ms/step - loss: 2.0102 - accuracy: 0.3362
<keras.callbacks.History at 0x7f2056469c50>
```

Figure 3: Sequential model 1 baseline output from training dataset

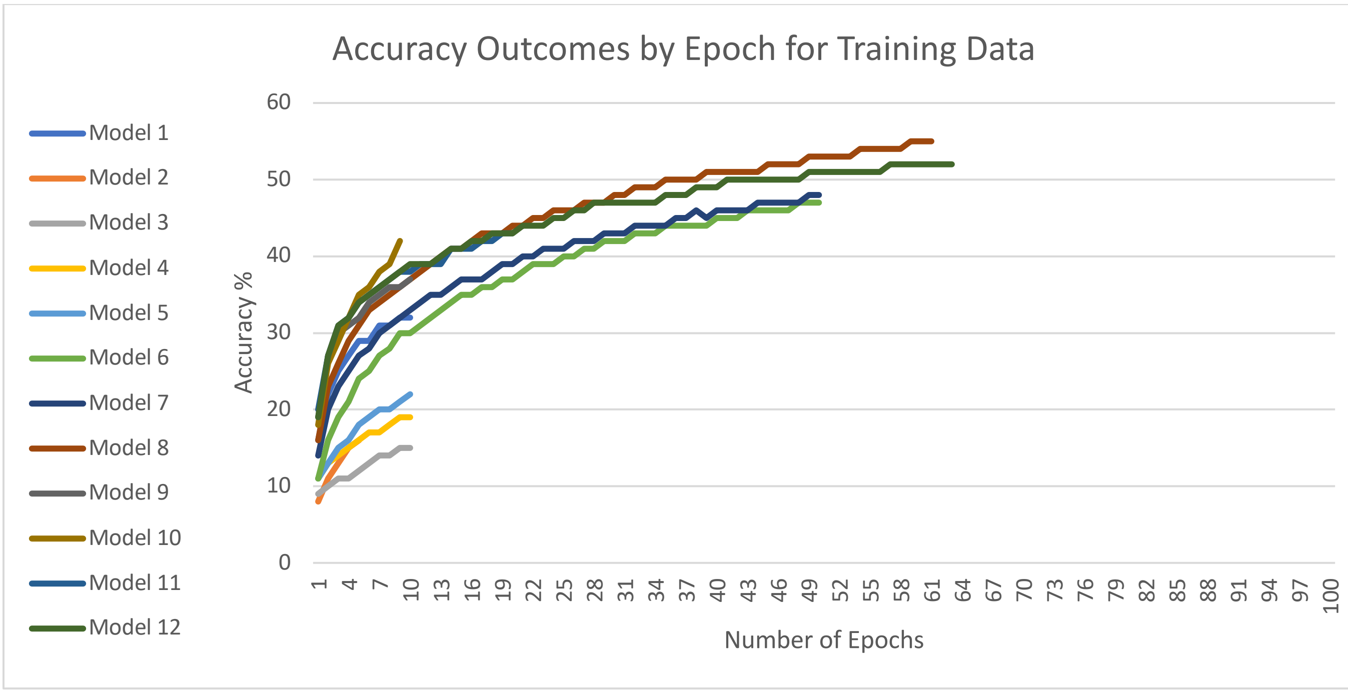


Figure 4: Accuracy percentage of each CNN across epochs

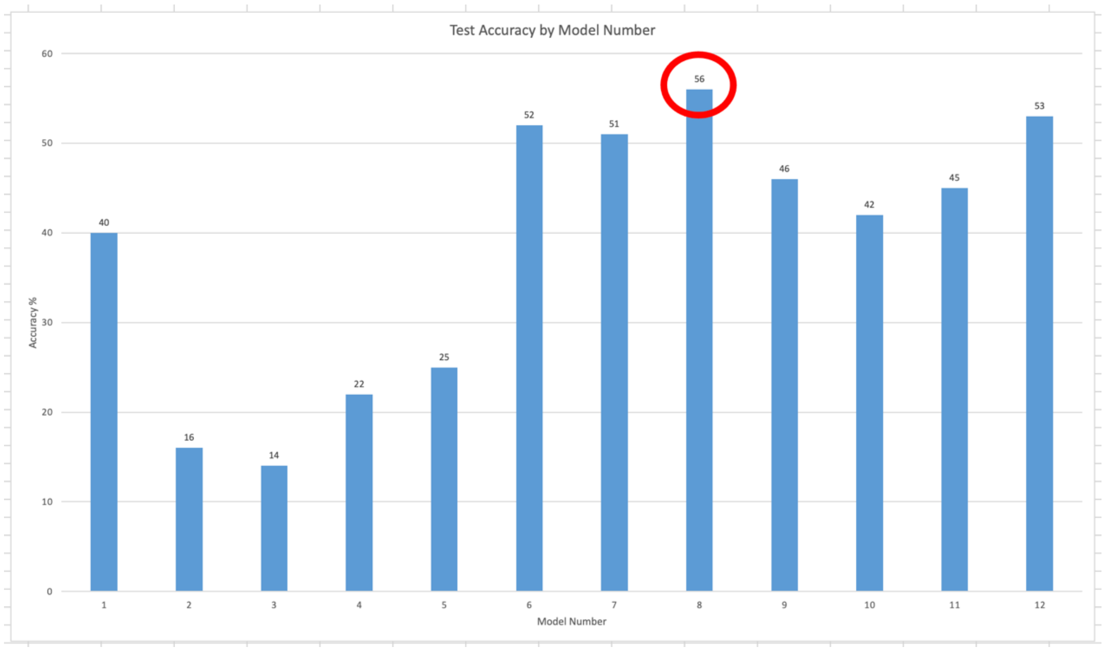


Figure 5: Accuracy end point percentage for each CNN model

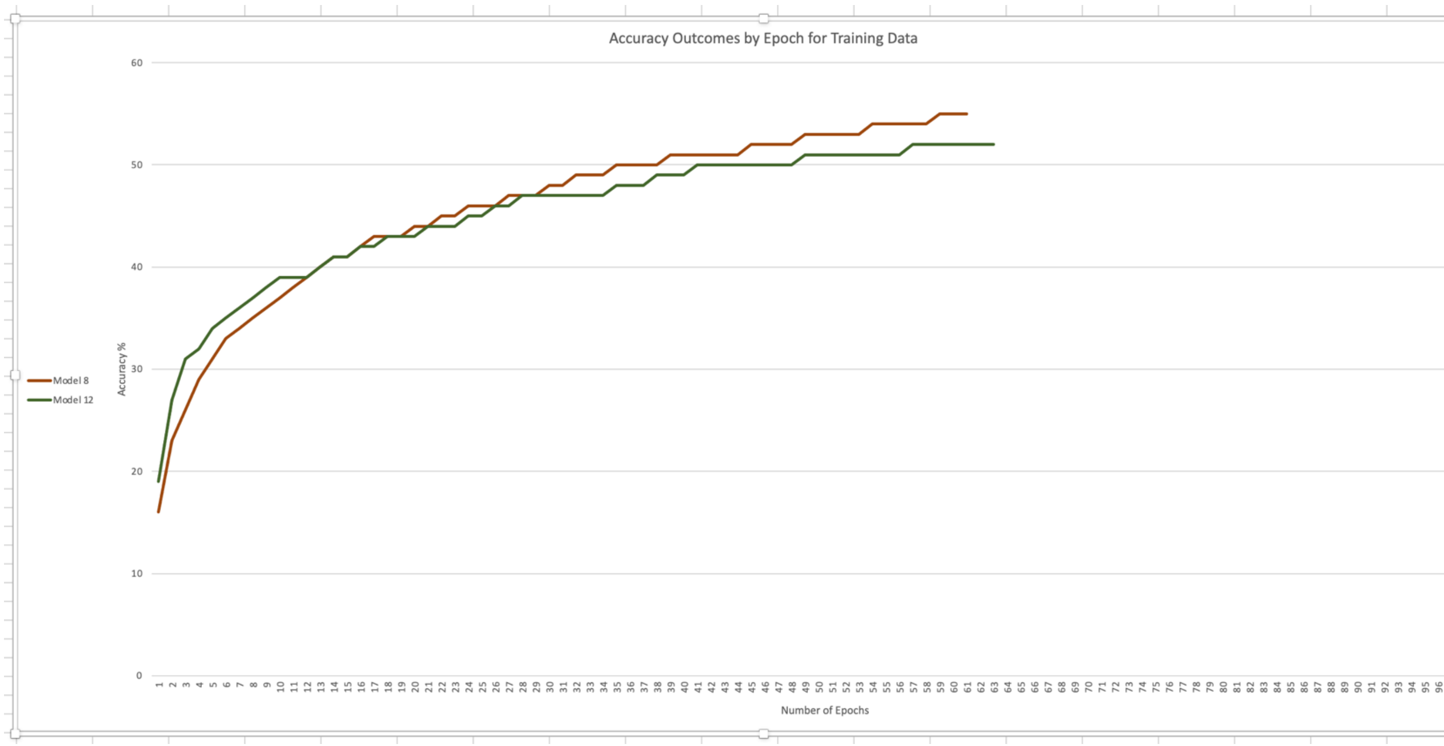
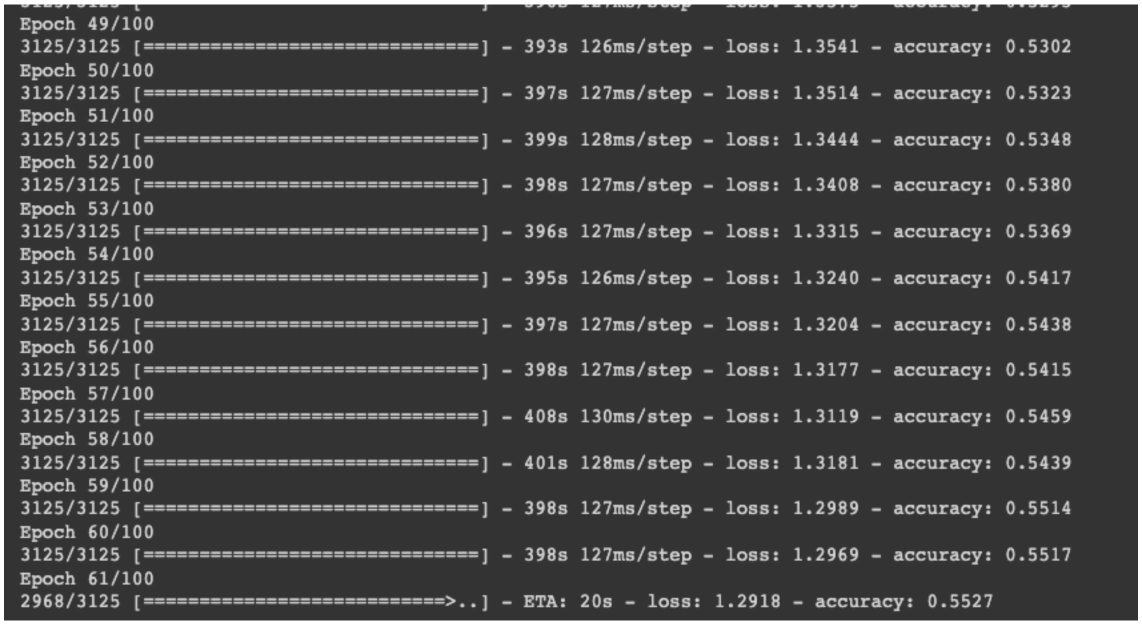


Figure 6: Accuracy outcome for the two finalist CNN models



```
Epoch 49/100
3125/3125 [==============================] - 393s 126ms/step - loss: 1.3541 - accuracy: 0.5302
Epoch 50/100
3125/3125 [==============================] - 397s 127ms/step - loss: 1.3514 - accuracy: 0.5323
Epoch 51/100
3125/3125 [==============================] - 399s 128ms/step - loss: 1.3444 - accuracy: 0.5348
Epoch 52/100
3125/3125 [==============================] - 398s 127ms/step - loss: 1.3408 - accuracy: 0.5380
Epoch 53/100
3125/3125 [==============================] - 396s 127ms/step - loss: 1.3315 - accuracy: 0.5369
Epoch 54/100
3125/3125 [==============================] - 395s 126ms/step - loss: 1.3240 - accuracy: 0.5417
Epoch 55/100
3125/3125 [==============================] - 397s 127ms/step - loss: 1.3204 - accuracy: 0.5438
Epoch 56/100
3125/3125 [==============================] - 398s 127ms/step - loss: 1.3177 - accuracy: 0.5415
Epoch 57/100
3125/3125 [==============================] - 408s 130ms/step - loss: 1.3119 - accuracy: 0.5459
Epoch 58/100
3125/3125 [==============================] - 401s 128ms/step - loss: 1.3181 - accuracy: 0.5439
Epoch 59/100
3125/3125 [==============================] - 398s 127ms/step - loss: 1.2989 - accuracy: 0.5514
Epoch 60/100
3125/3125 [==============================] - 398s 127ms/step - loss: 1.2969 - accuracy: 0.5517
Epoch 61/100
2968/3125 [===========================>..] - ETA: 20s - loss: 1.2918 - accuracy: 0.5527
```

Figure 7: Final accuracy results for Model 8

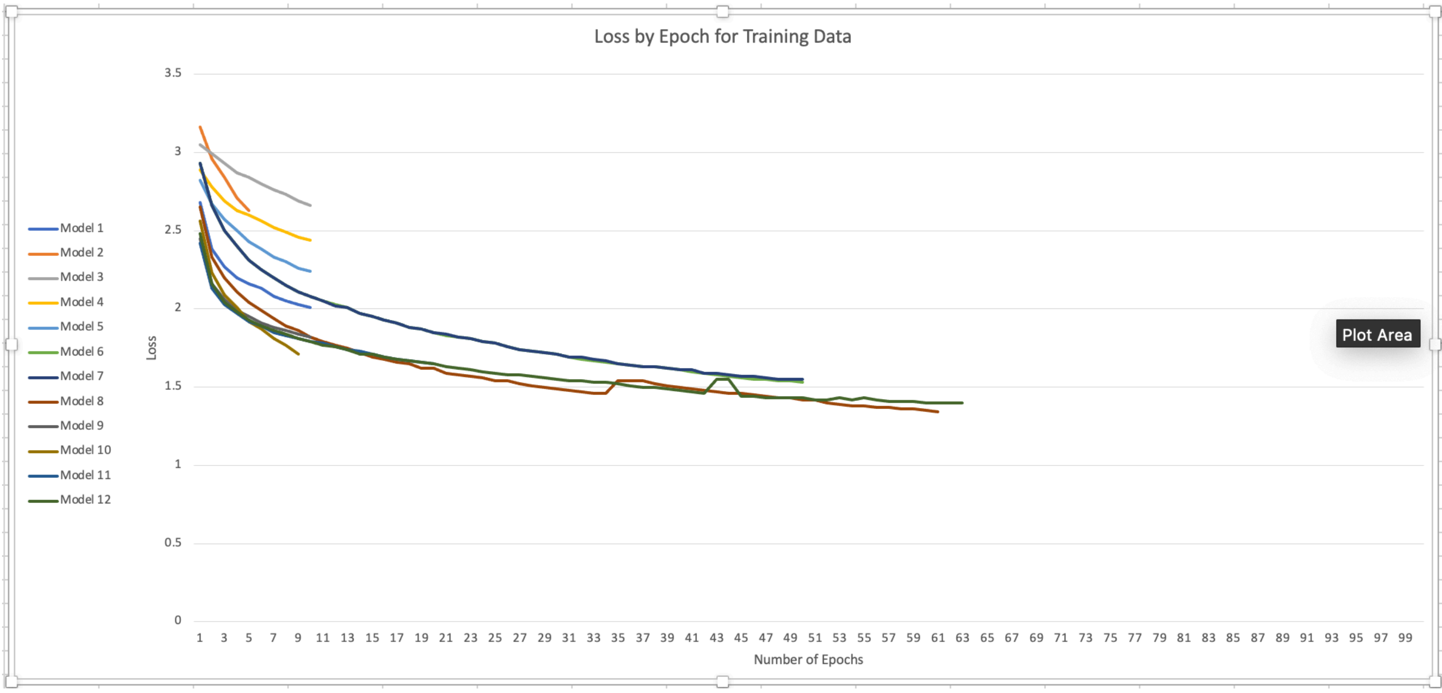


Figure 8: Loss rate for all CNN models across epochs

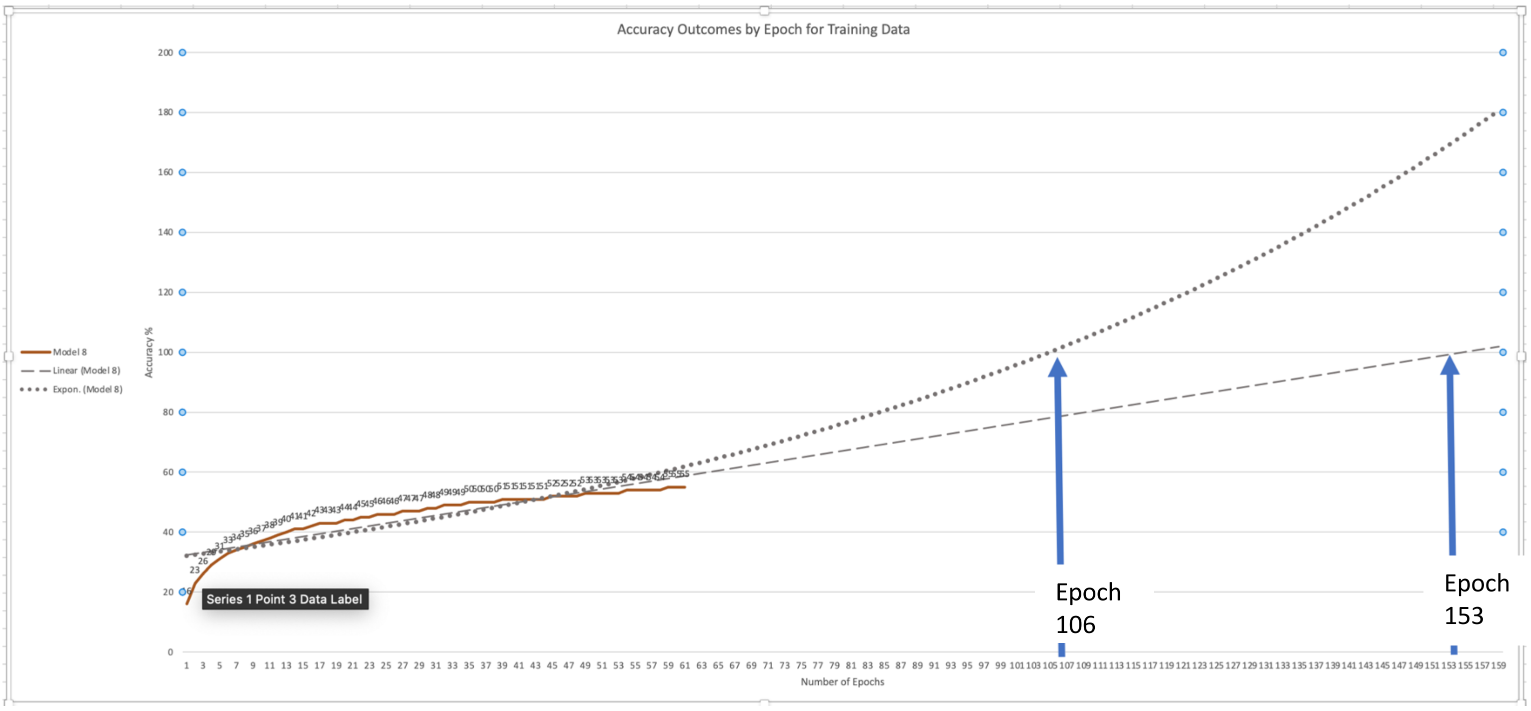


Figure 9: Linear and Exponential projections for Model 8 to 100% accuracy

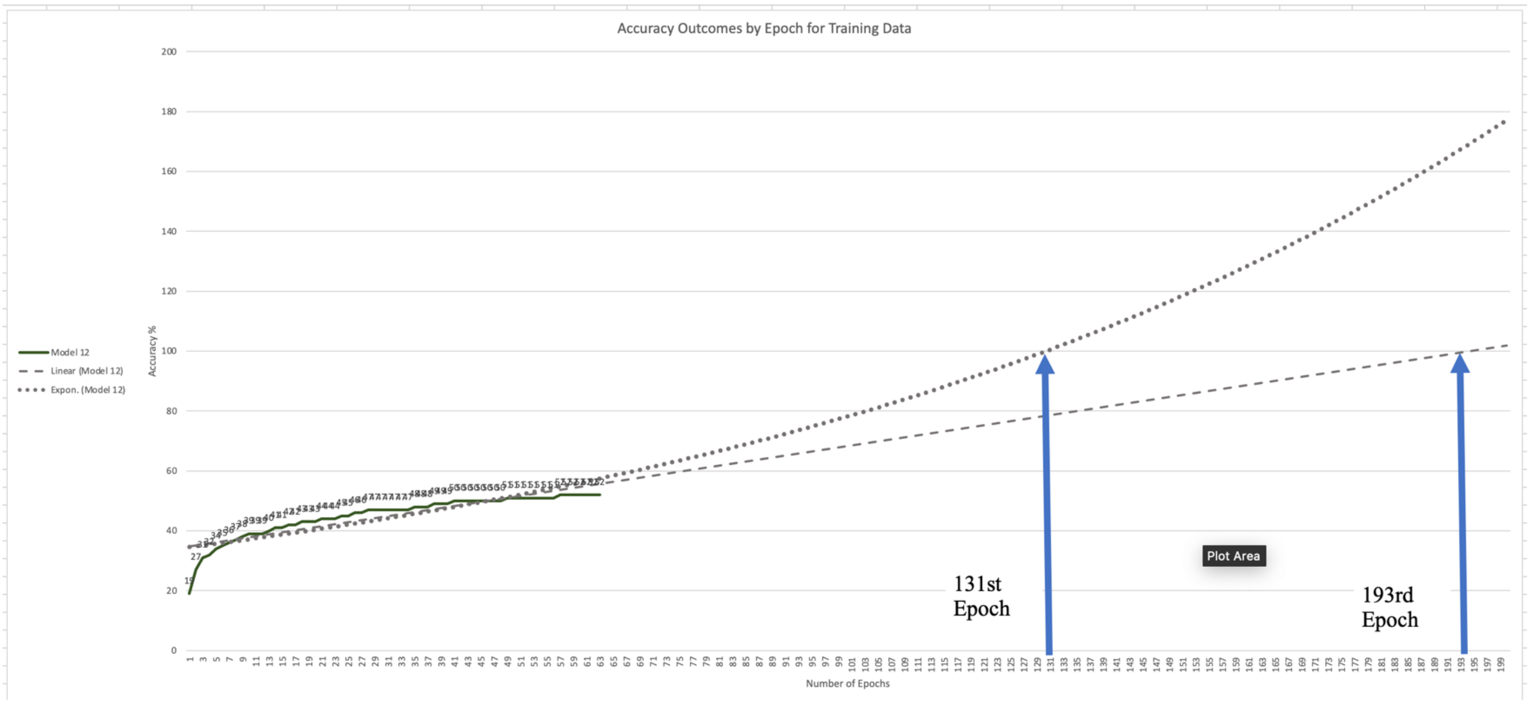


Figure 10: Linear and Exponential projections for Model 12 to 100% accuracy

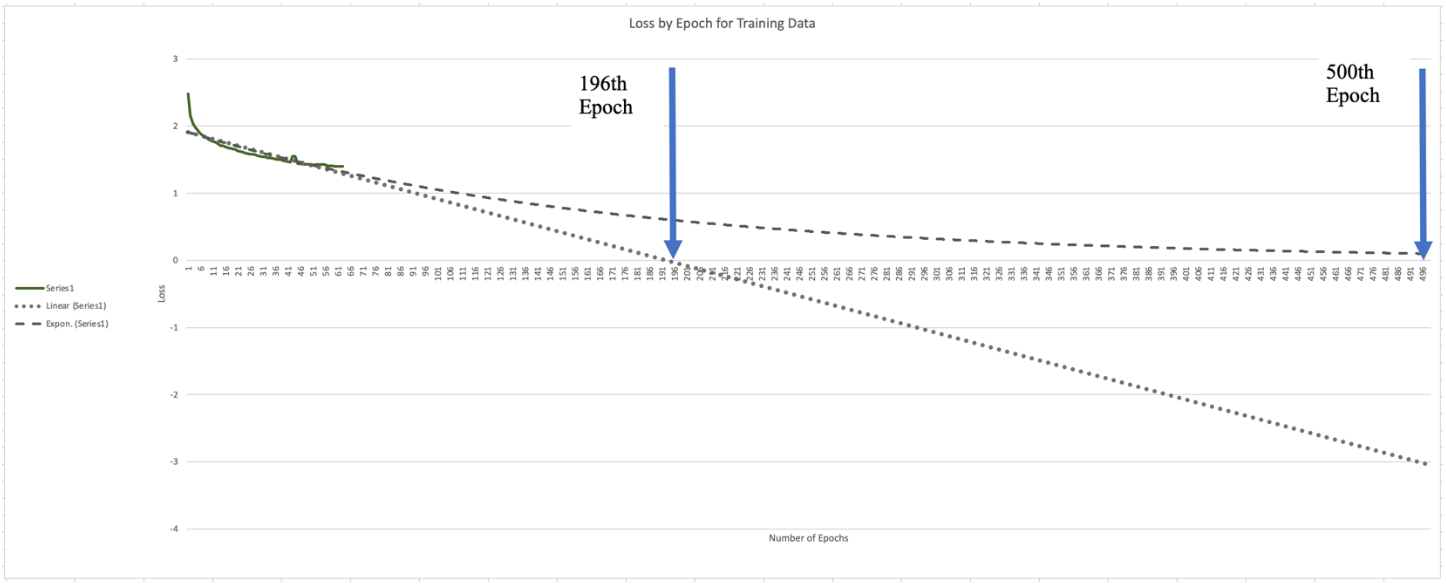


Figure 11: Linear and Exponential projections for Model 8 to 0% Error

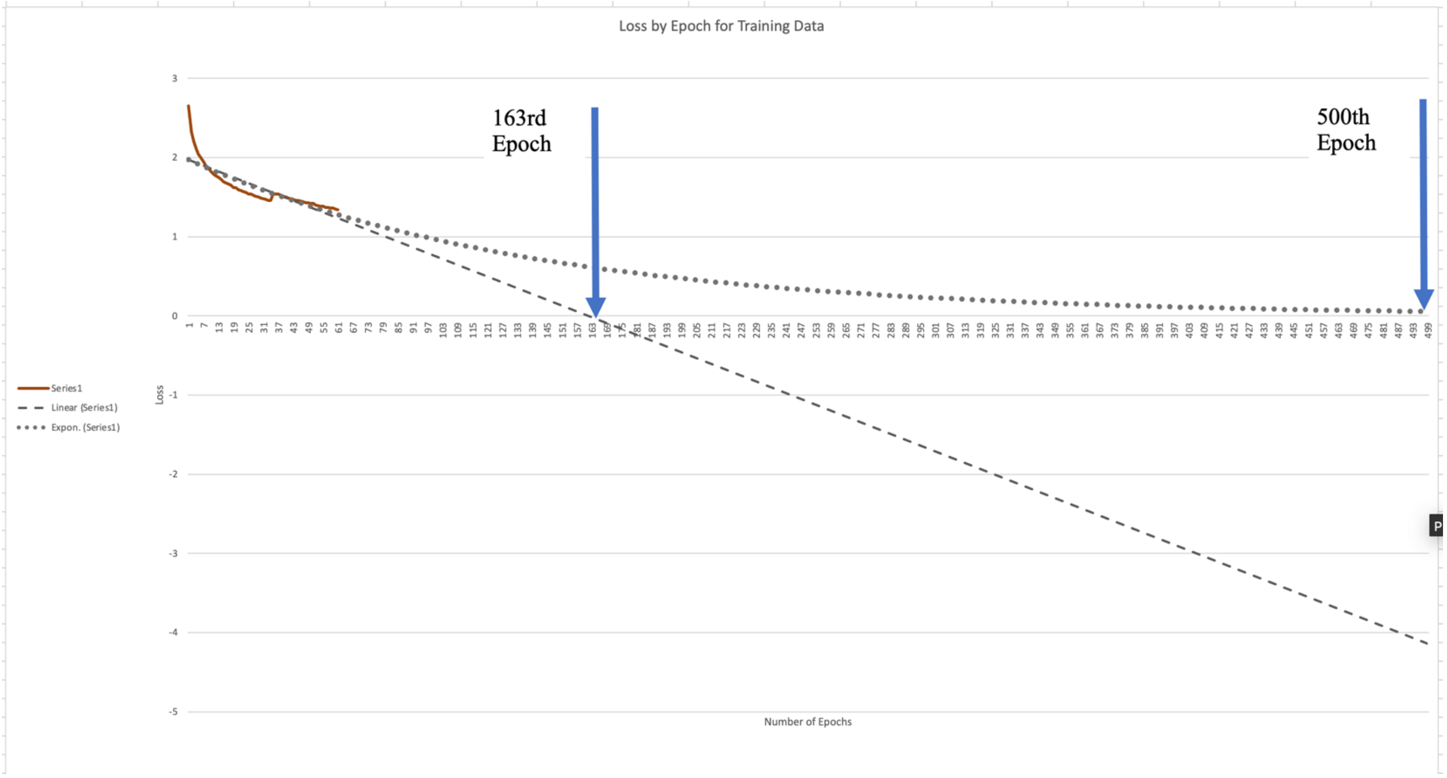


Figure 12: Linear and Exponential projections for Model 12 to 0% Error