Reed Ballesteros
MSDS 458-DL: Artificial Intelligence & Deep Learning
Spring 2023
Prof. Syamala Srinivasan, Ph.D.
April 30, 2023

A.2: Second Research/Programming Assignment: Deep Neural Networks & Convolutional Neural Networks

Abstract

This study details the creation and comparison of several deep neural networks (DNNs) and convolutional neural networks (CNNs) trained and tested for their performance accuracy using the CIFAR-10 image dataset. While we find a combination of regularization techniques applied on a CNN can yield significant gains in accuracy, we also find that adding more depth to a model would eventually reach diminishing returns at the cost of longer training and increased complexity from the use of more parameters.

Introduction

Computer-aided image processing and recognition has been an ongoing topic of research in the areas of artificial intelligence and machine learning. With state-of-the-art models from the early 2010s such as AlexNet we've witnessed significant improvements in the image processing space through the use of deep-learning models. From self-driving cars to security systems these visualization models are becoming more integrated into everyday applications, and with that the strive to keep improving their accuracy continues today.

This report will document various deep-learning model configurations we created for performing image processing, particularly deep neural networks (DNNs) consisting of several fully connected layers of neural networks, and convolutional neural networks (CNNs) which utilize feature map layers applied as filters. We will choose the model configuration that yields the highest accuracy in image classification against the CIFAR-10 dataset. Model configurations include varying numbers of convolution and fully connected layers, batch normalization, dropout utilization, regularization implementation, and hyperparameter adjustments.

Literature Review

The CNN architecture for computer-aided image processing has been popularized by the development of models such as AlexNet and VGGNet. AlexNet, developed in 2012 by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, was the first CNN-based model to win the annual ImageNet competition. The architecture consisted of eight neural network layers which included five convolution layers organized

into three series grouped by max pooling, followed by two fully connected layers and a classification layer (Krizhevsky et al. 2017). AlexNet introduced the use of relu activation, dropout regularization, and popularized the leveraging of GPUs for training models, all of which are common deep-learning components using in many modern computer vision models today. VGGNet, created in 2014 by Karen Simonyan and Andrew Zisserman, builds upon the AlexNet architecture with a more complex model containing 16 total layers (with a 19-layer variant also available), in which the CNN framework utilizes five series of small 3x3 convolutional layers stacked upon each other (Simonyan et al. 2015).

The models we detail in this report contain elements from these architectures, such as the use of ReLU activation, dropout regularization, and small 3x3 convolutional layers, with model training performed using an NVIDIA RTX 3060Ti GPU-based desktop computer.

Methods

Research Design

We will conduct our research by first procuring the CIFAR-10 dataset available in the Karas framework in Python and perform exploratory data analysis (EDA) on it to understand the data and its corresponding labels we will be using to build, train, and test our models. Each model is represented as an experiment. We will first build and train 4 baseline DNN and CNN models that do not contain any regularization components and evaluate them based on their accuracy of classifying the test dataset. From those baseline models we will create variations of them by adding various regularization components such as batch normalization, dropout, and ridge regression (also called 'L2 Regularization') as well as evaluate their performance accuracy in classifying the test dataset.

We will also plot each model's training and validation loss and accuracy over their respective number of epochs of training. Training for each model will automatically stop after validation accuracy does not improve after a given number of epochs, in which the best training epoch will be saved before the model overfits to the training data. Confusion matrices for each model will also be charted to understand how well they perform for each image classification. Other visualizations include classification scatterplots via T-distributed Stochastic Neighbor Embedding (TSNE), and for CNNs, feature maps for each convolutional, max pooling, and dropout layer.

Implementation

Our models will be created using the Keras and TensorFlow frameworks available in Python. These are the most common tools used in developing such models based on their ease of use, customization options available, and their scalability for each convolutional or dense layer. Each convolutional and dense hidden layer will use the ReLU activation function as its simplicity in calculation can make the feedforward and backpropagation process fast, which will be helpful when we train deep models that contain a large number of parameters. The final output 10-way classification layer will use a softmax activation function.

The Keras model compiler will be configured to use the Adam optimizer, calculate cross entropy loss using the SparseCategorialCrossentropy class, and model performance will be based on accuracy metrics. For reproducibility and stabilize the random nature of the model training process, we will set a system seed of 43.

Dataset: CIFAR-10

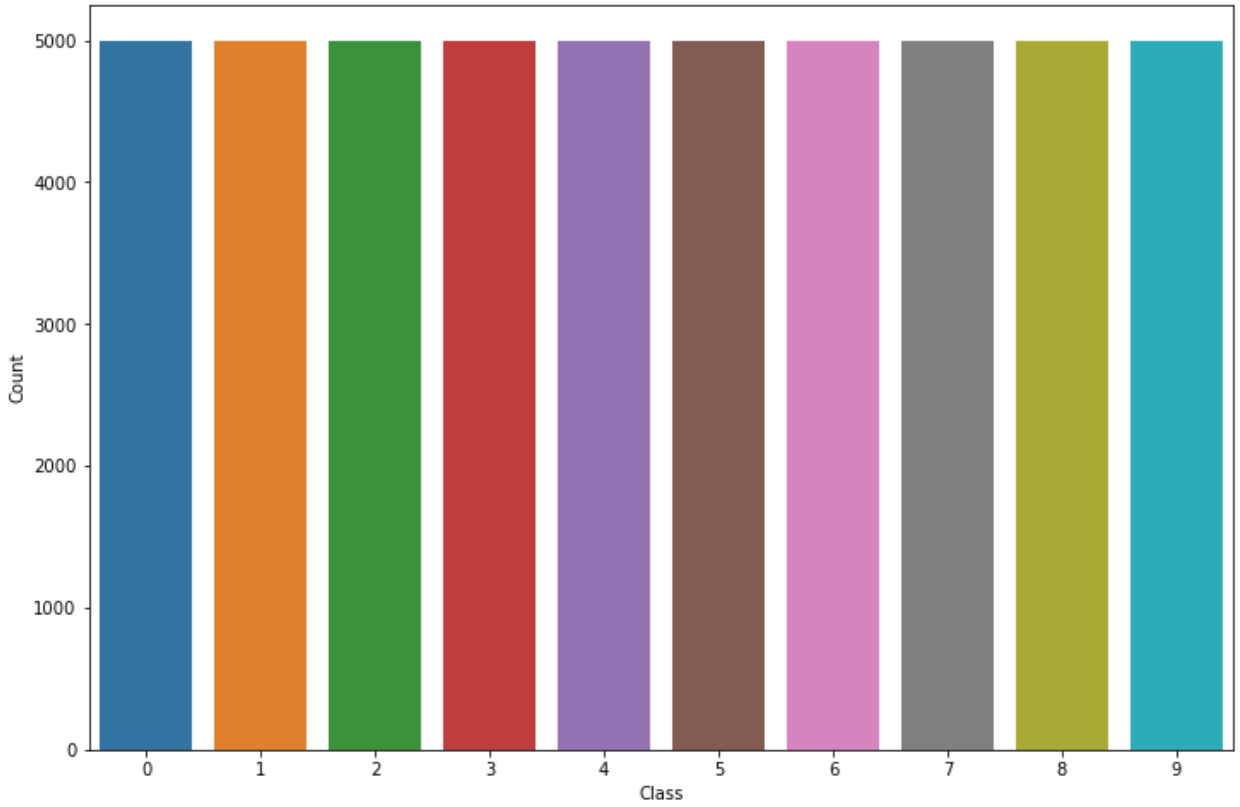
The CIFAR-10 dataset provided by the Canadian Institute for Advanced Research is a well-known collection of 60000 32x32 (1024 pixels) color images over 10 classes (airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck) often used to train computer vision algorithms. For our modeling research the dataset will be split into three sets for training (45000), validation (5000), and testing (10000). Each image is represented as a 32x32 matrix over 3 color channels (red, green, and blue), where each value in the matrix represents a pixel integer value between 0 to 255. We will transform the integer pixel data to a standardized value between 0 and 1 before feeding the image into the models.

Exploratory Data Analysis (EDA)

As described above, the CIFAR-10 dataset is a collection of 60000 32x32 color images, each classified as one of 10 categories: airplane, car, bird, cat, deer, dog, frog, horse, ship, and truck. Below is a sample of images from the dataset:

While the 32x32 pixel images are not high-resolution, the contents are visible for them to be identified by their respective classes.

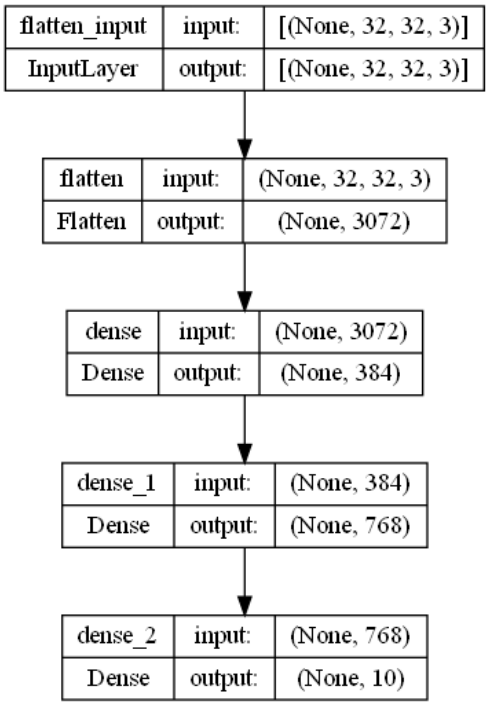
The combined training and validation datasets contain 5000 samples of each classification.



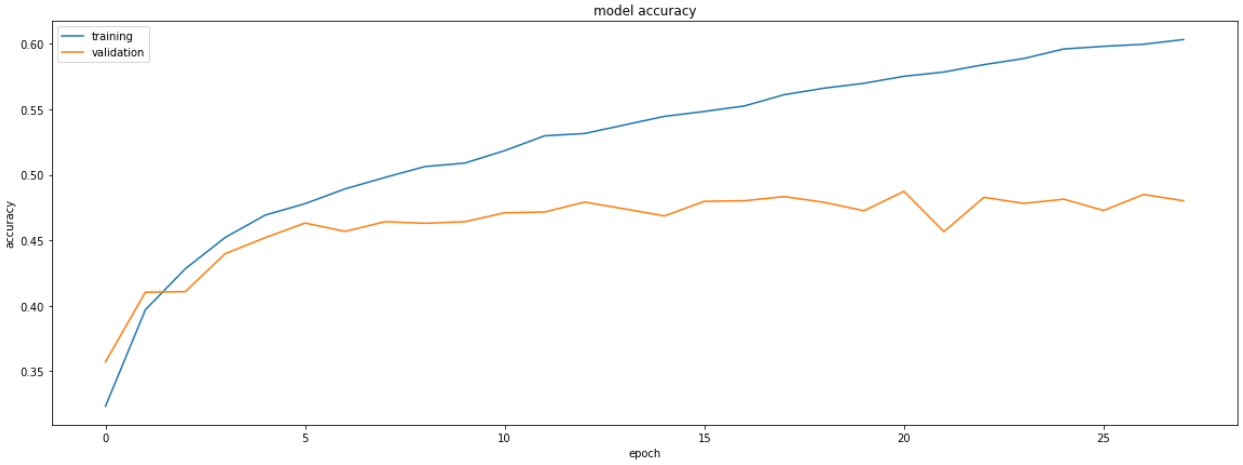
Results

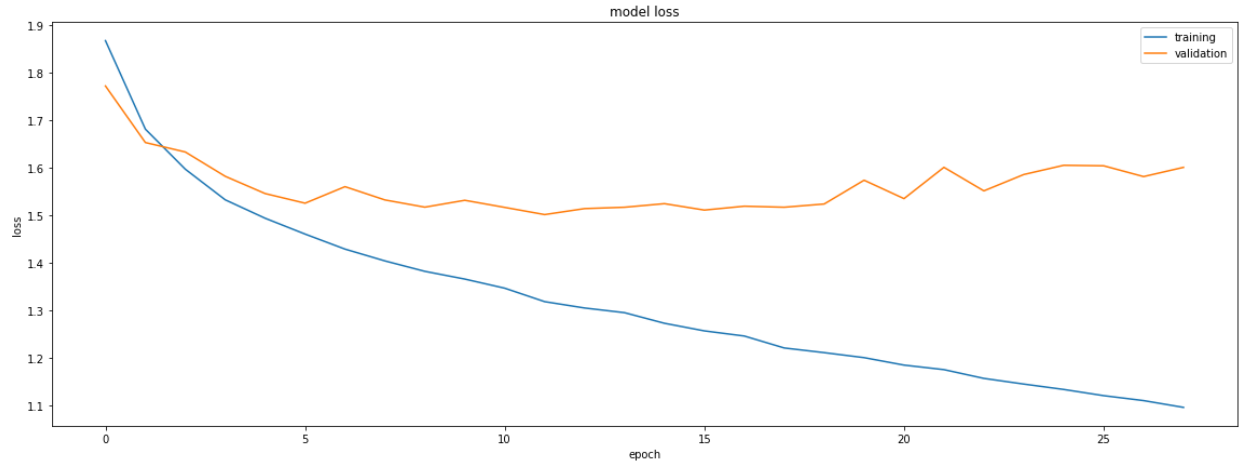
Experiment 1: DNN - 2 Fully Connected Layers

Our first experiment is a DNN consisting of 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, then fed into the resulting 10-way classification layer.



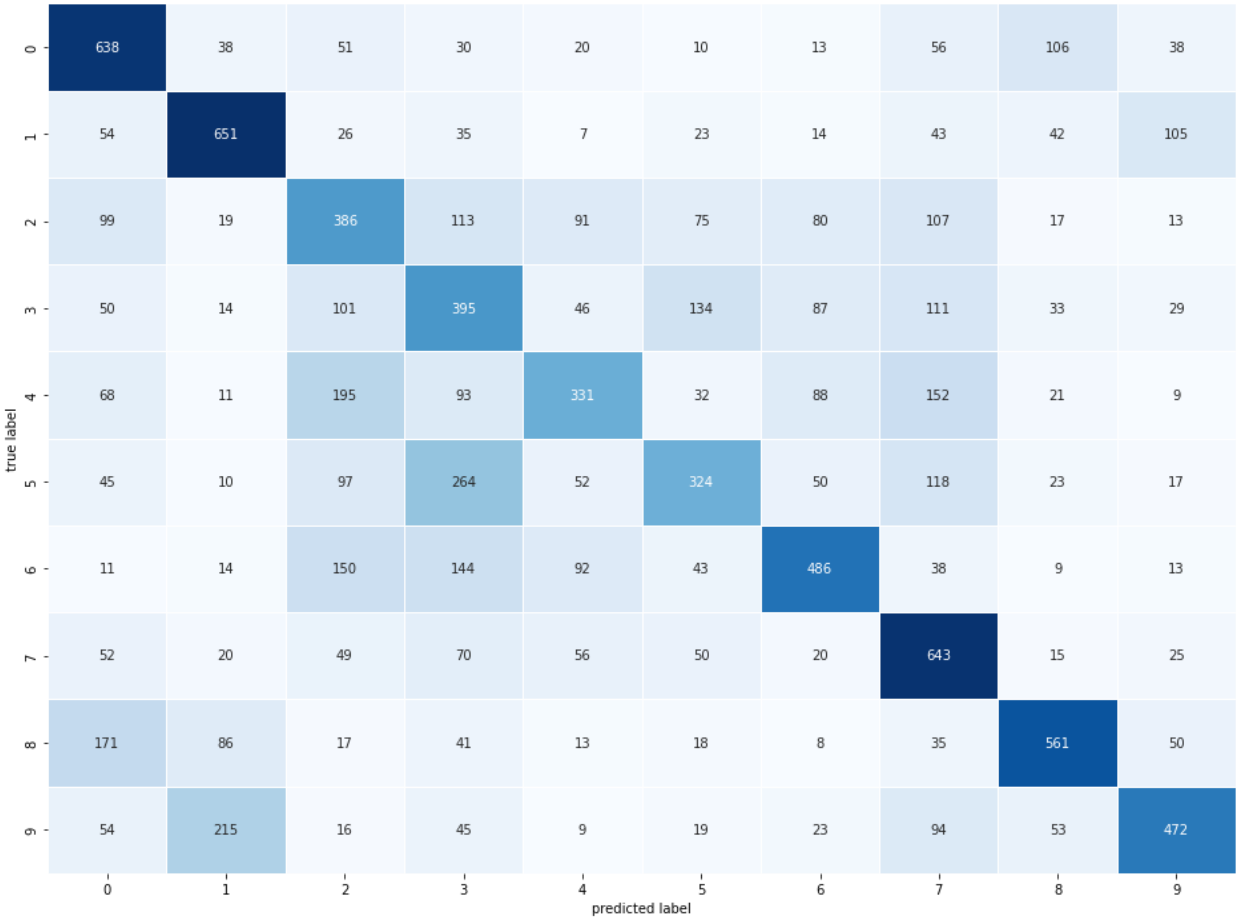
We chart the training and validation accuracy and cross entropy loss below.





From the charts above, we see validation results start to diverge from the training results after just 5 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was only 48.9%, which is not great, but better than a 10% random chance of classifying a CIFAR-10 image to one of the 10 different classifications.

The experiment yields the following confusion matrix:

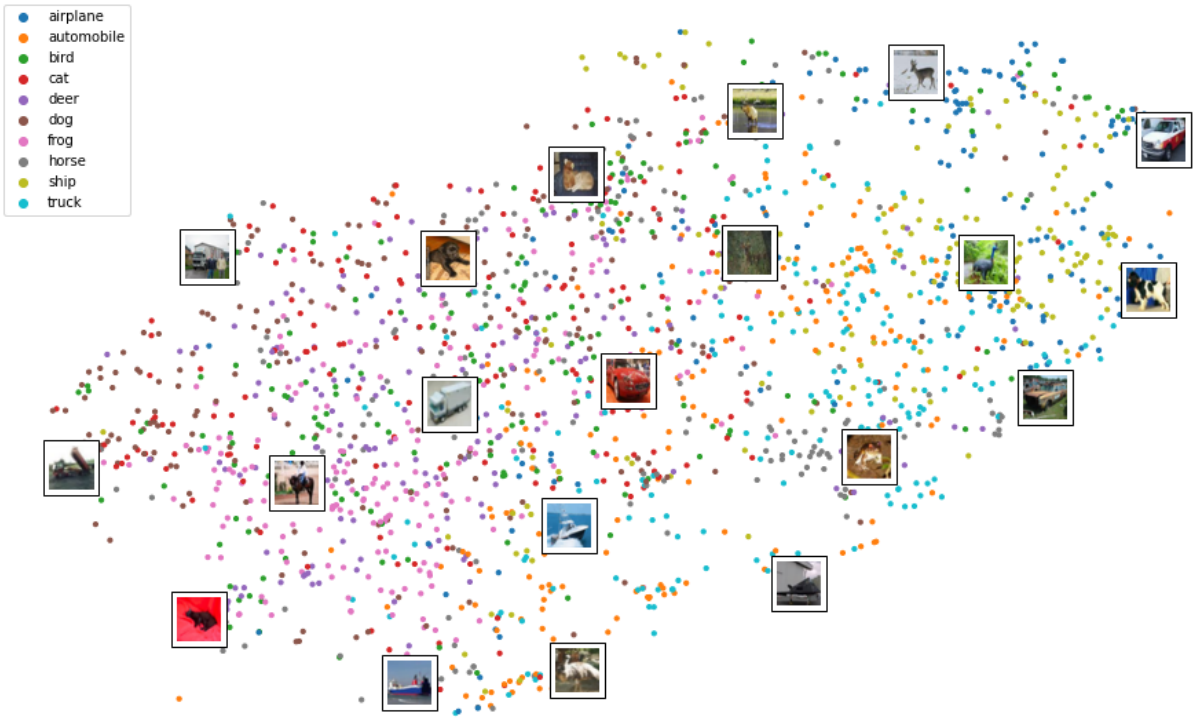


The numbered labels correspond to the following classification in the CIFAR-10 dataset:

Label	**Classification**
0	Airplane
1	Automobile
2	Bird
3	Cat
4	Deer
5	Dog
6	Frog
7	Horse
8	Ship
9	Truck

The confusion matrix above shows that while the model can classify most images of automobiles (label 1) in the test dataset, it also can mislabel images of trucks (label 9) as automobiles. The model has the most trouble correctly classifying images as dogs (label 5), where it can often mislabel images of cats (label 3) as dogs.

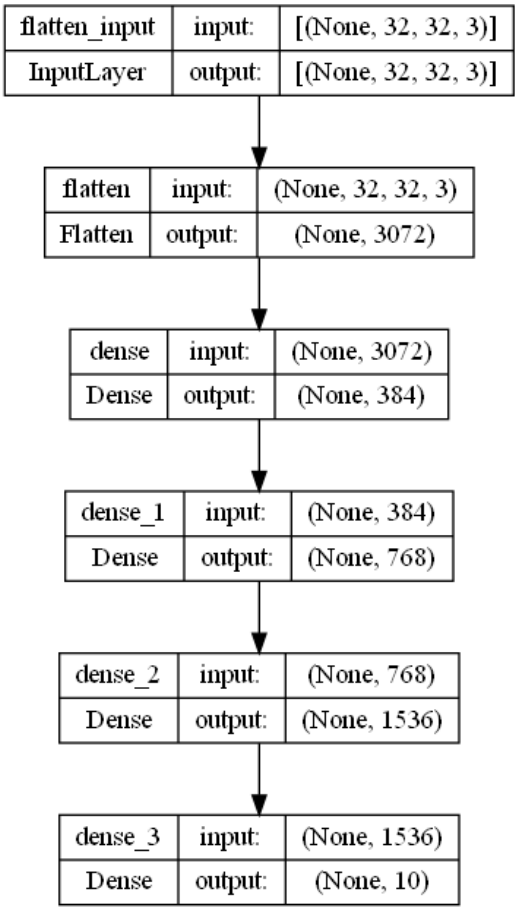
We can reduce the model to 2 features and create a scatterplot to observe the distribution of a sample of model classifications.



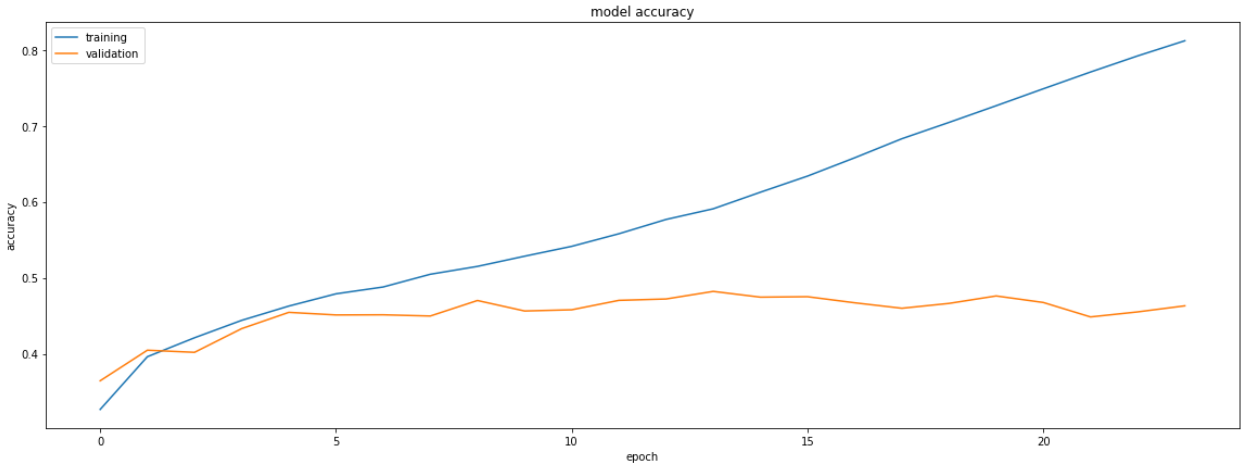
We see from the plot above classifications are not really grouped together, thus the low test accuracy of the model.

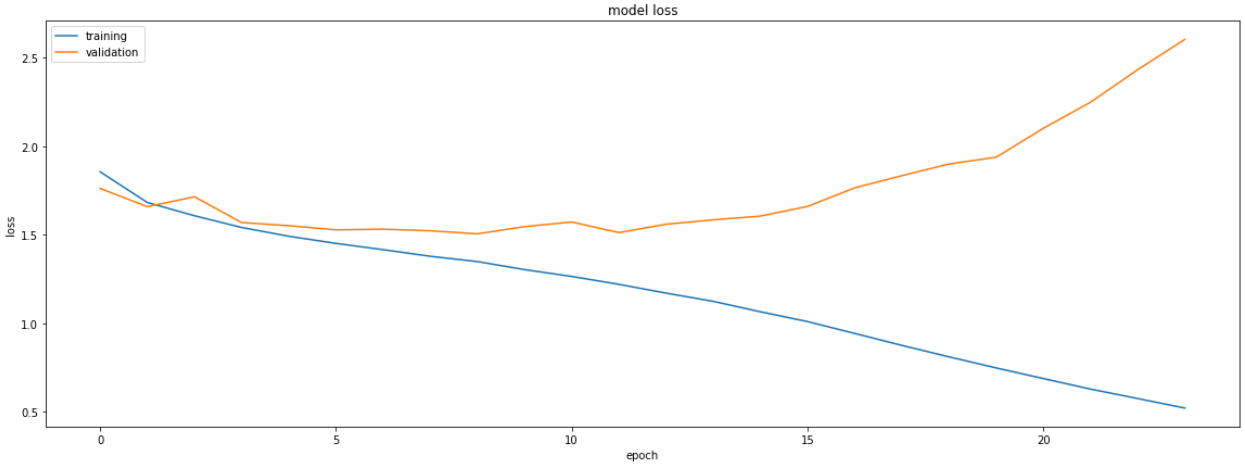
Experiment 2: DNN - 3 Fully Connected Layers

Experiment 2 is a DNN consisting of 3 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, and the third layer containing 1536 units. The processed image is then fed into the resulting 10-way classification layer.



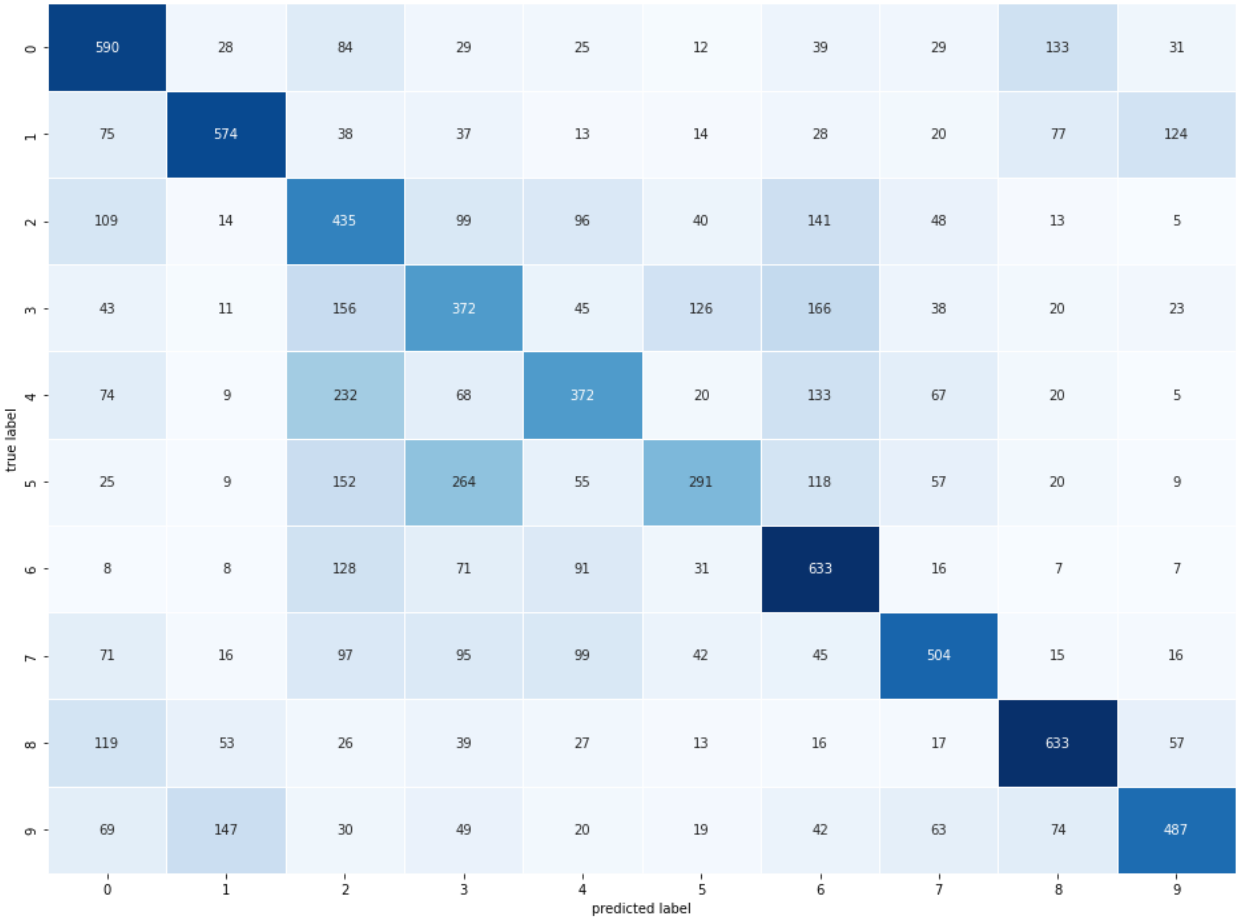
The training and validation accuracy and cross entropy loss is below.





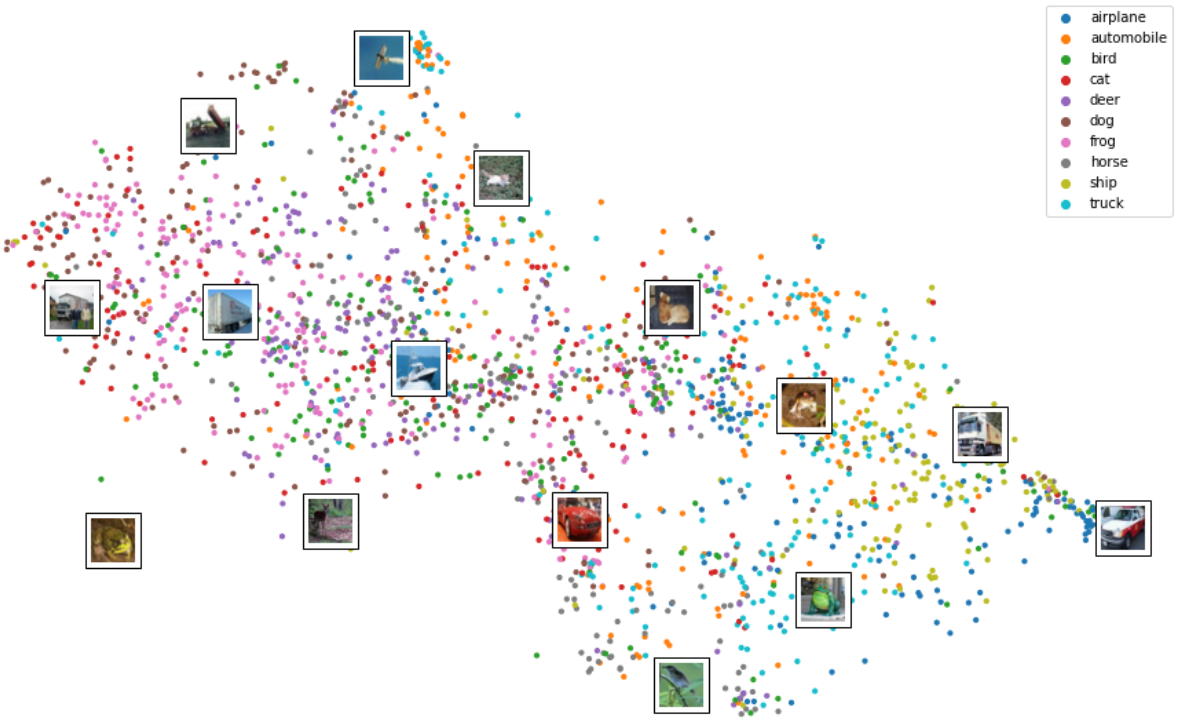
Similar to experiment 1, we see validation results start to diverge from the training results after just 5 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was also 48.9%.

The experiment yields the following confusion matrix:



The confusion matrix is very similar to that of experiment 1 as well, where it can often mislabel trucks (label 9) as automobiles (label 1), and cats (label 3) as dogs (label 5). While it can correctly classify many images as frogs (label 6) it can also mislabel other images of animals in the dataset as frogs as well.

The scatterplot distribution of a sample of model classifications is below:

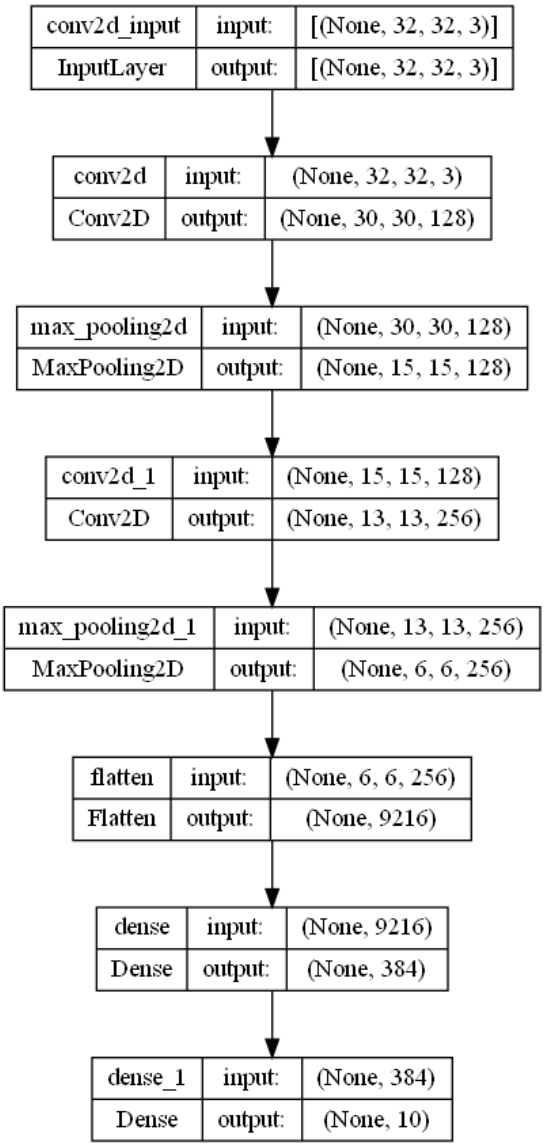


Again, we see that classifications are not grouped well in the scatterplot above.

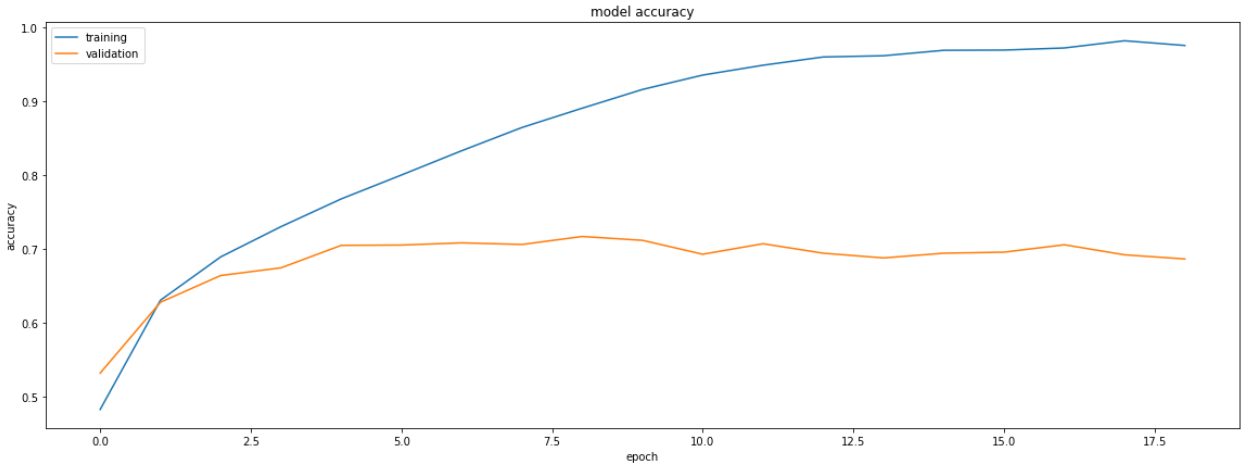
We find that adding another fully connected layer to a DNN model does not improve accuracy against the CIFAR-10 test dataset.

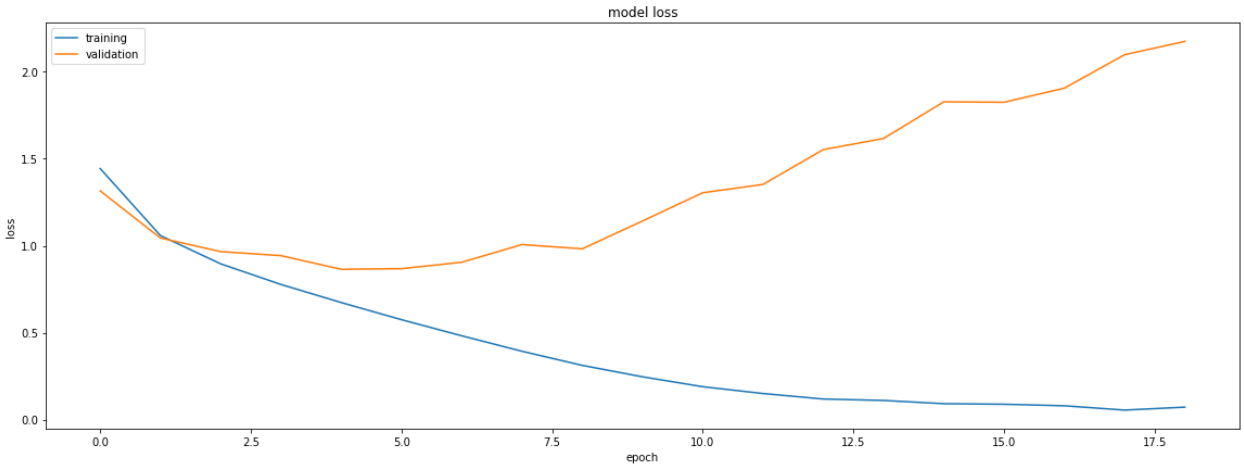
Experiment 3: CNN - 2 Convolutional Layers with 1 Fully Connected Layer

The model for experiment 3 uses a CNN consisting of 2 small 3x3 convolutional layers with the first layer containing 128 filters and the second layer containing 256 filters. The convolutional base will then feed the processed image into a DNN containing 1 fully connected layer with 384 units followed by the 10-way classification output layer.



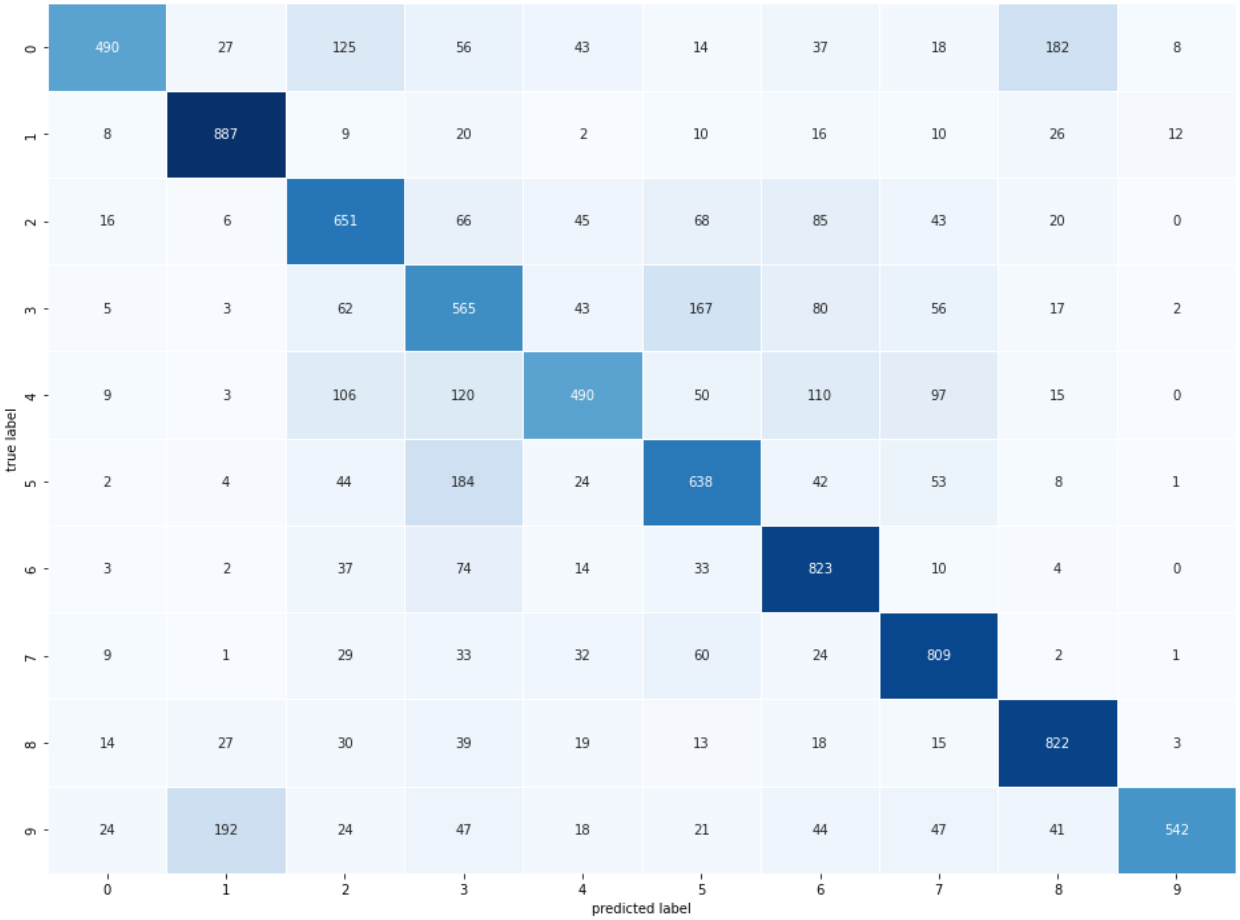
We chart the training and validation accuracy and cross entropy loss below.





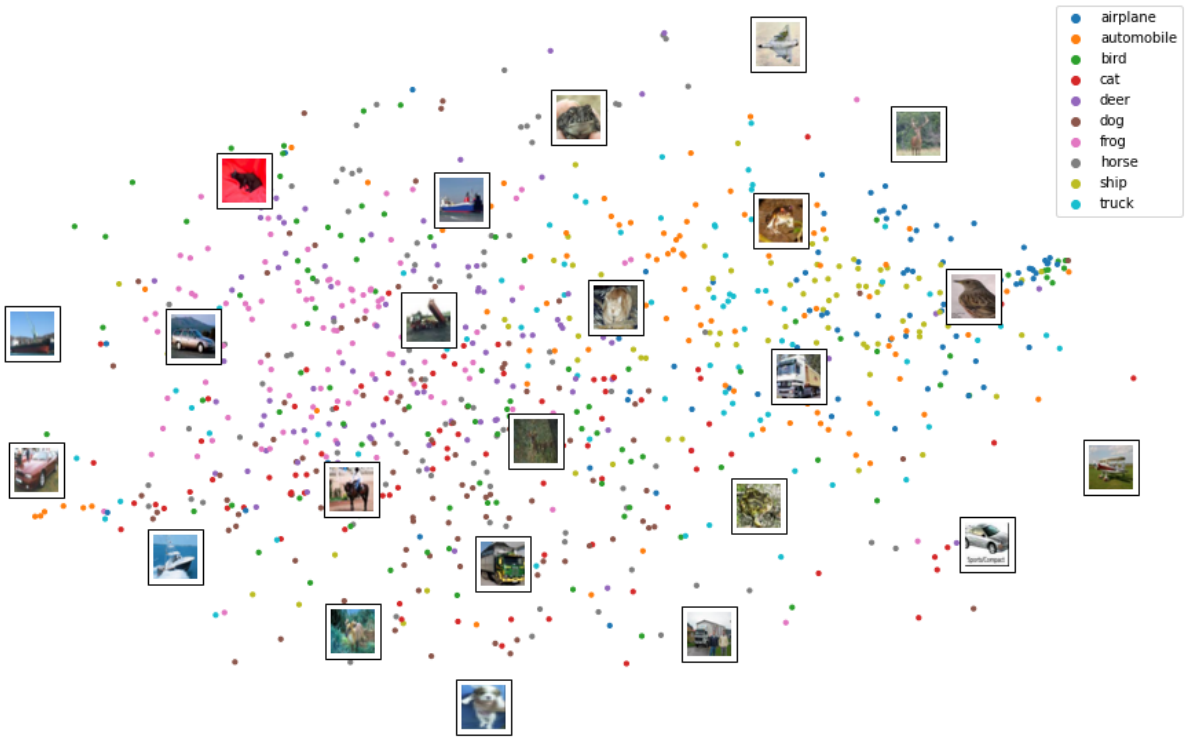
We see validation results start to diverge from the training results after just 5 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was 70.9%, a significant improvement over the DNN-based models.

The experiment yields the following confusion matrix:



From the matrix above we can see a general improvement in animal classification compared to the DNN-based models, but misclassifications still often occur.

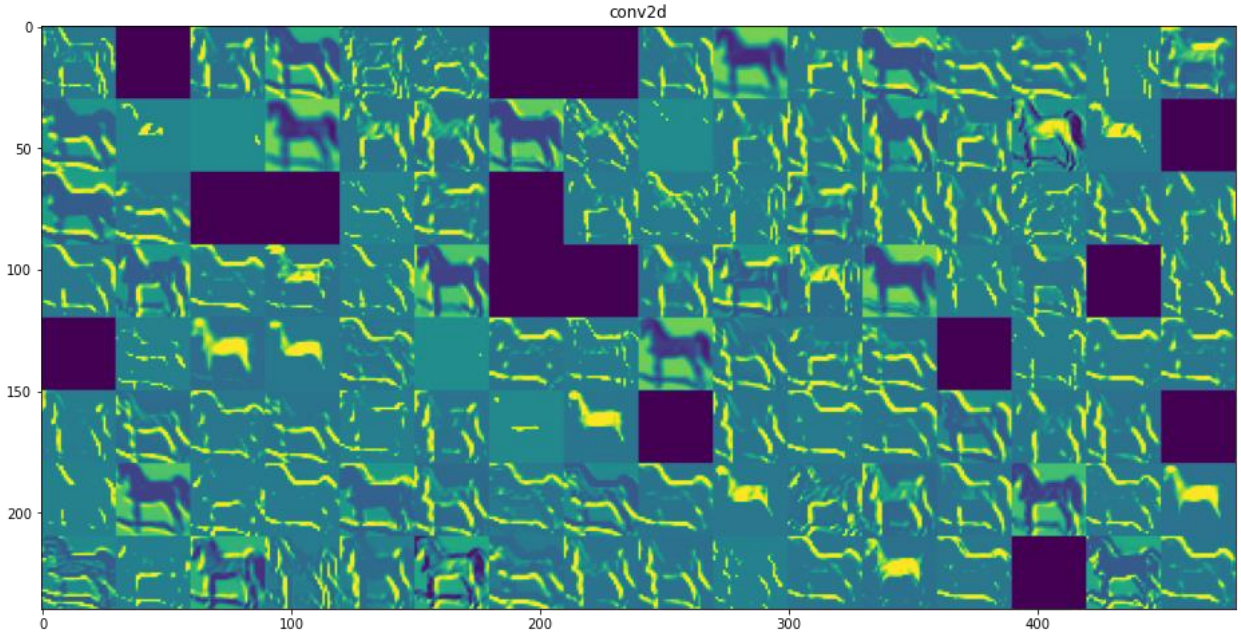
The scatterplot sample yields the following:



We can start to see some grouping of classifications, but at 70.9% test accuracy the plot is still pretty mixed.

We can also observe the feature maps of the convolutional model. For example, a horse sample of the CIFAR-10 dataset is the following:

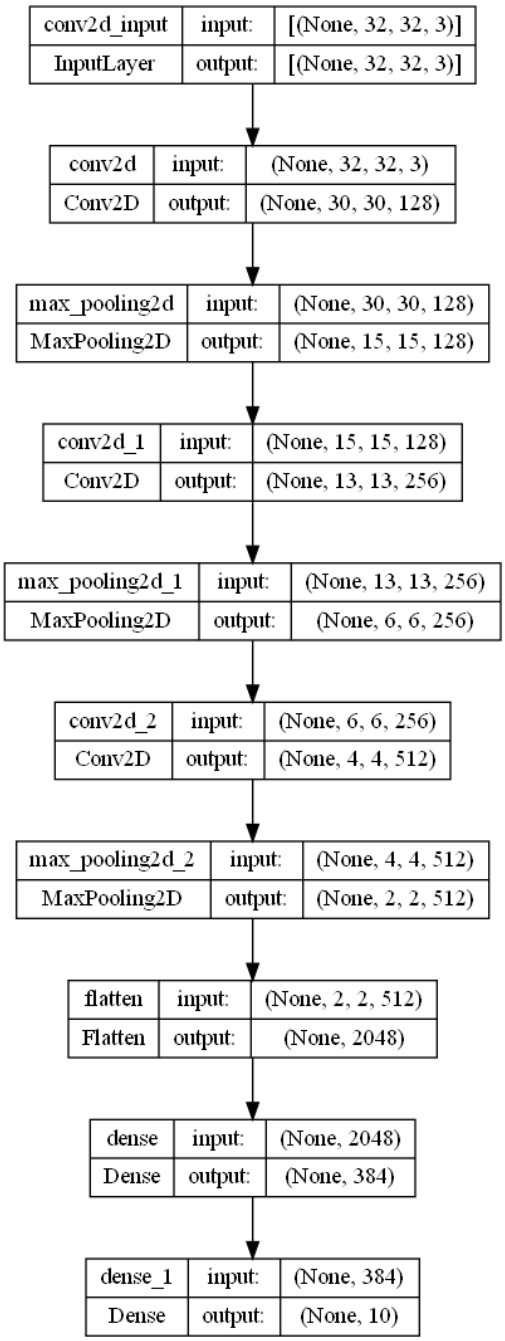
A sample of some of the feature maps created in the first convolutional layer of the CNN model is the following:



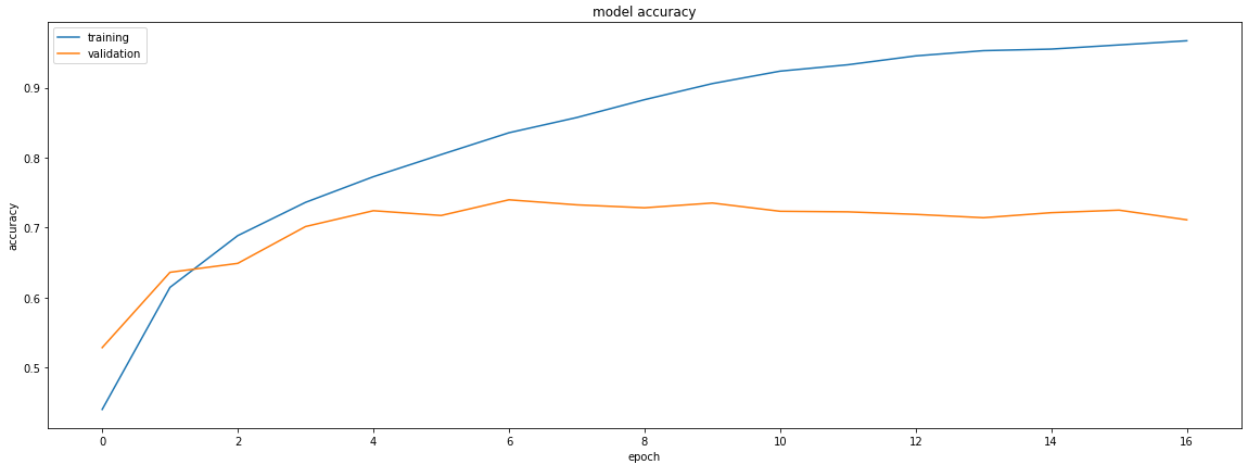
These individual maps reduce the sample image into a set of specific features via filters and are used for classification, resulting in a significant improvement in accuracy against the test dataset.

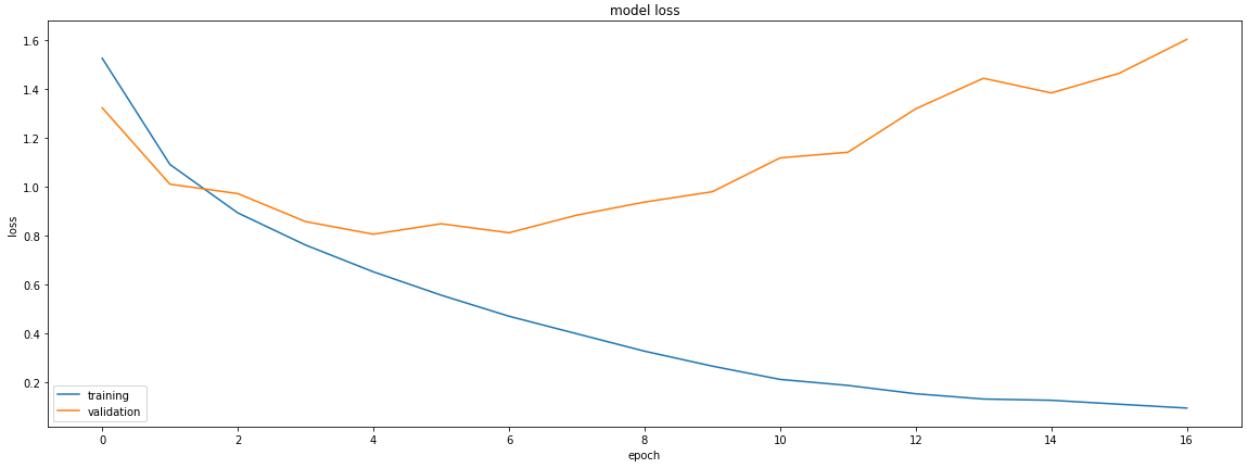
Experiment 4: CNN - 3 Convolutional Layers with 1 Fully Connected Layer

In experiment 4 we use a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. This convolutional base also feeds into a DNN containing 1 fully connected layer with 384 units followed by the 10-way classification output layer.



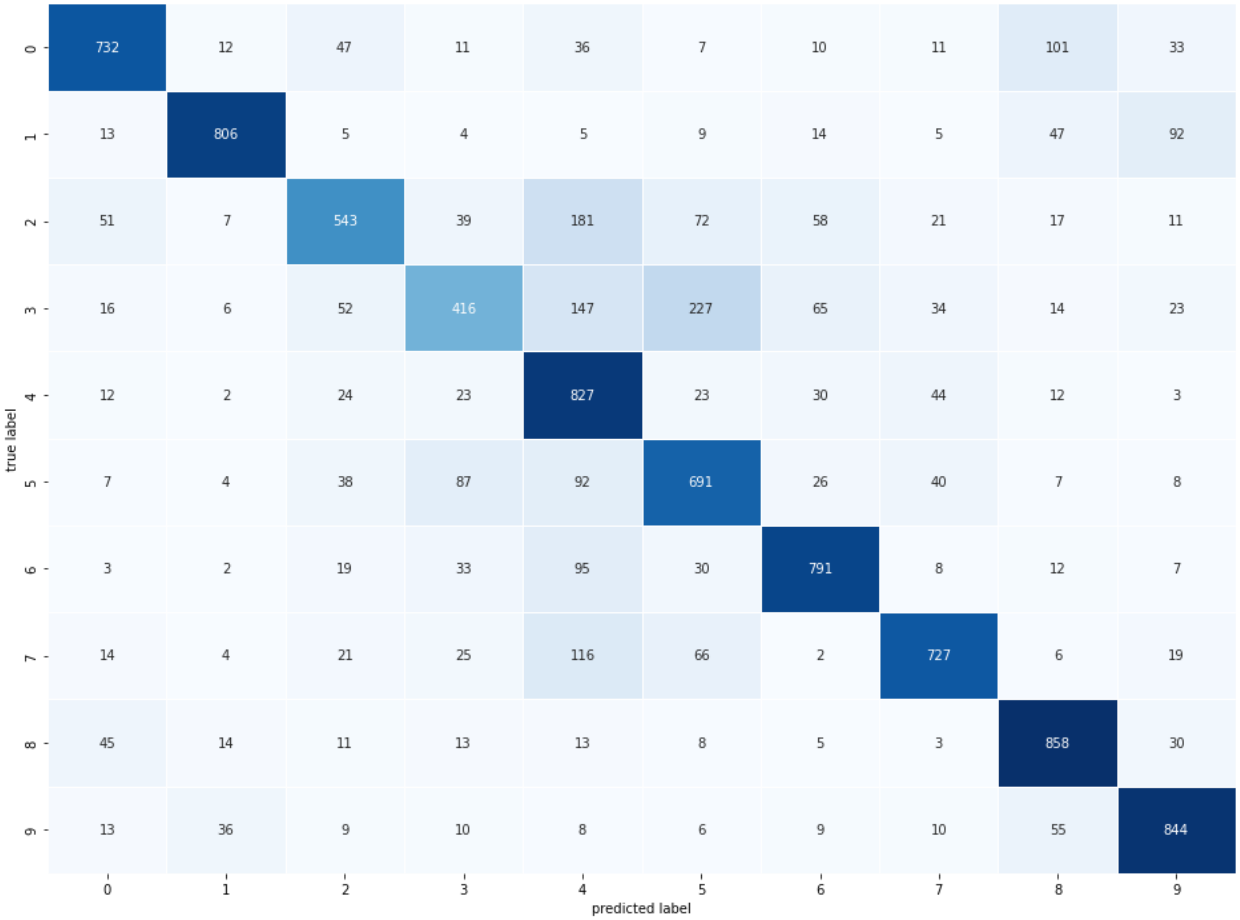
Training and validation accuracy and cross entropy loss is below.





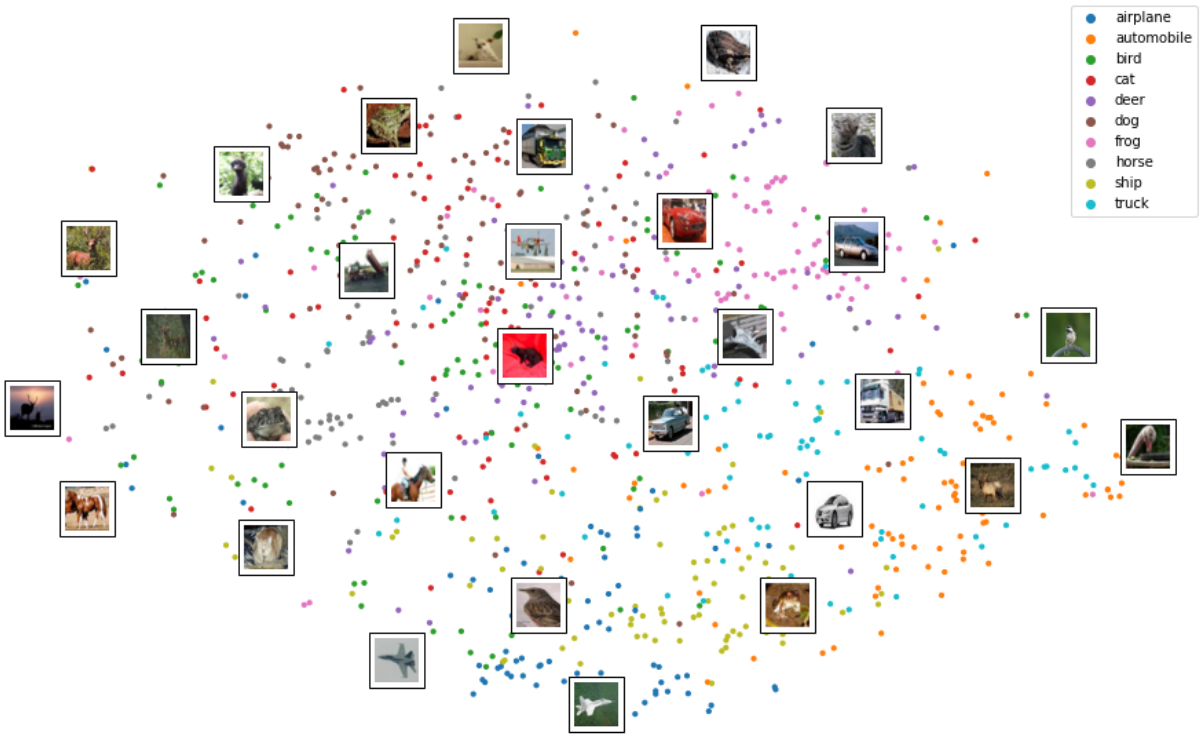
We see validation results start to diverge from the training results after just 5 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was 72.4%, an improvement of about 1.5% over experiment 3.

The experiment yields the following confusion matrix:



From the matrix above we can see a general improvement in animal classification compared to the DNN-based models, but misclassifications still often occur. Cats (label 3) are still often misclassified as dogs (label 5), while the model can misclassify various animals as deer (label 4).

The scatterplot sample yields the following:



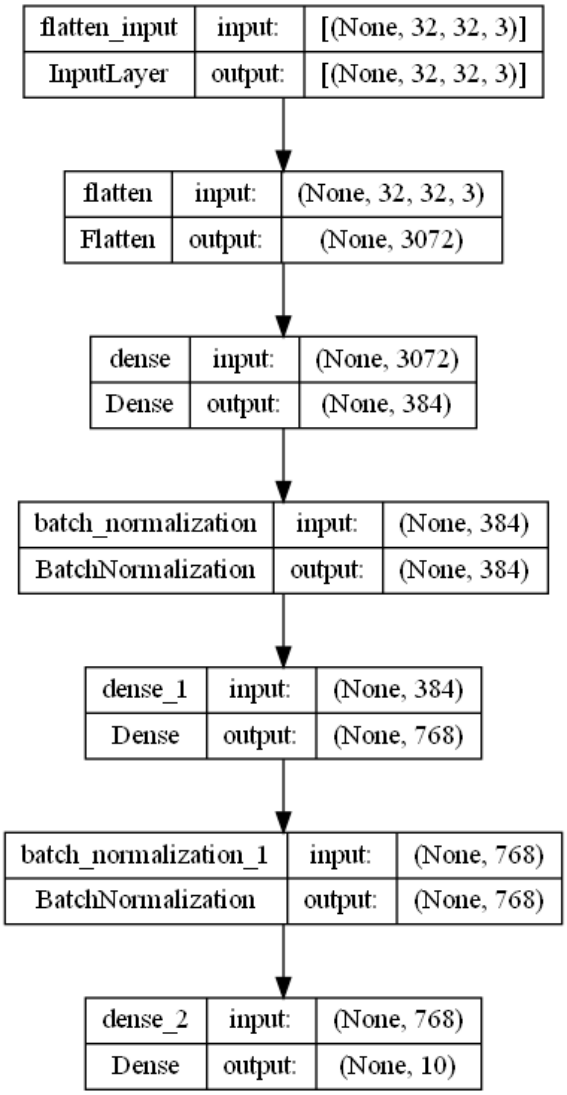
We can see groupings start to occur but the results are still mixed.

In experiments 1 to 4, we found that using convolutional layers in a neural network greatly improves accuracy compared to models using a network composed of only fully connected layers. Convolutional layers and their ability to create feature maps through the use filters helps the model process an image by specific parts, which makes for a much more efficient and accurate classification.

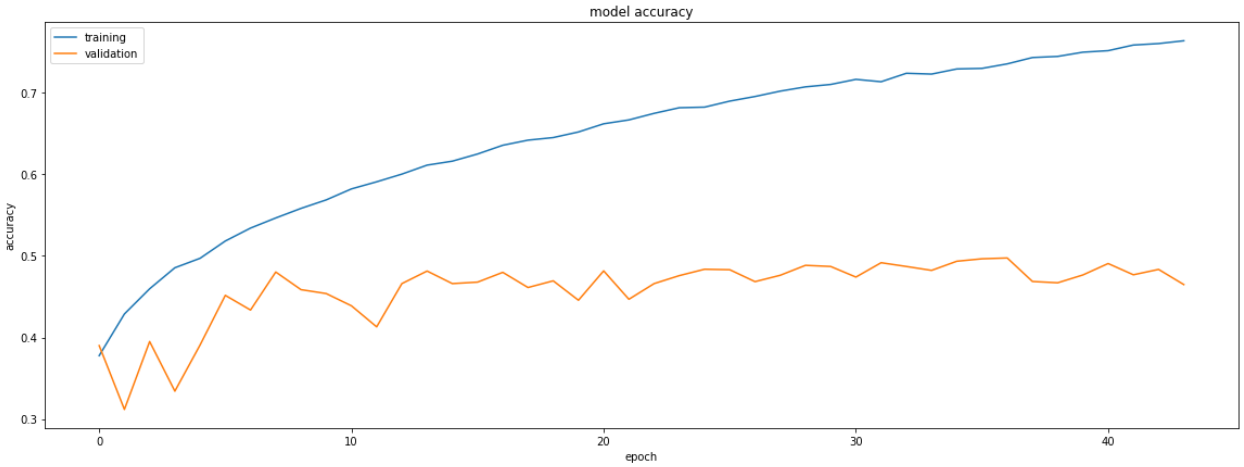
In experiments 5 to 8 we will create the same 4 previous models but will add a batch normalization regularization method and observe its effect on model accuracy.

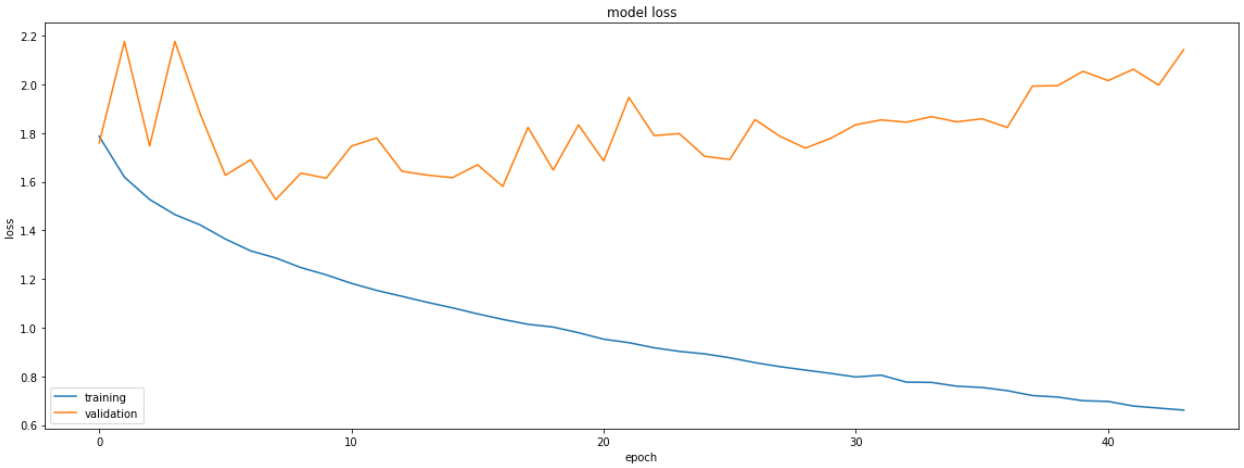
Experiment 5: DNN - 2 Fully Connected Layers and Batch Normalization

Our fifth experiment is similar to experiment 1, a DNN consisting of 2 fully connected layers, with the first layer containing 384 units, the second layer containing 768 units, and then fed into the resulting 10-way classification layer. In this experiment, batch normalization regularization processes follow the fully connected layers.



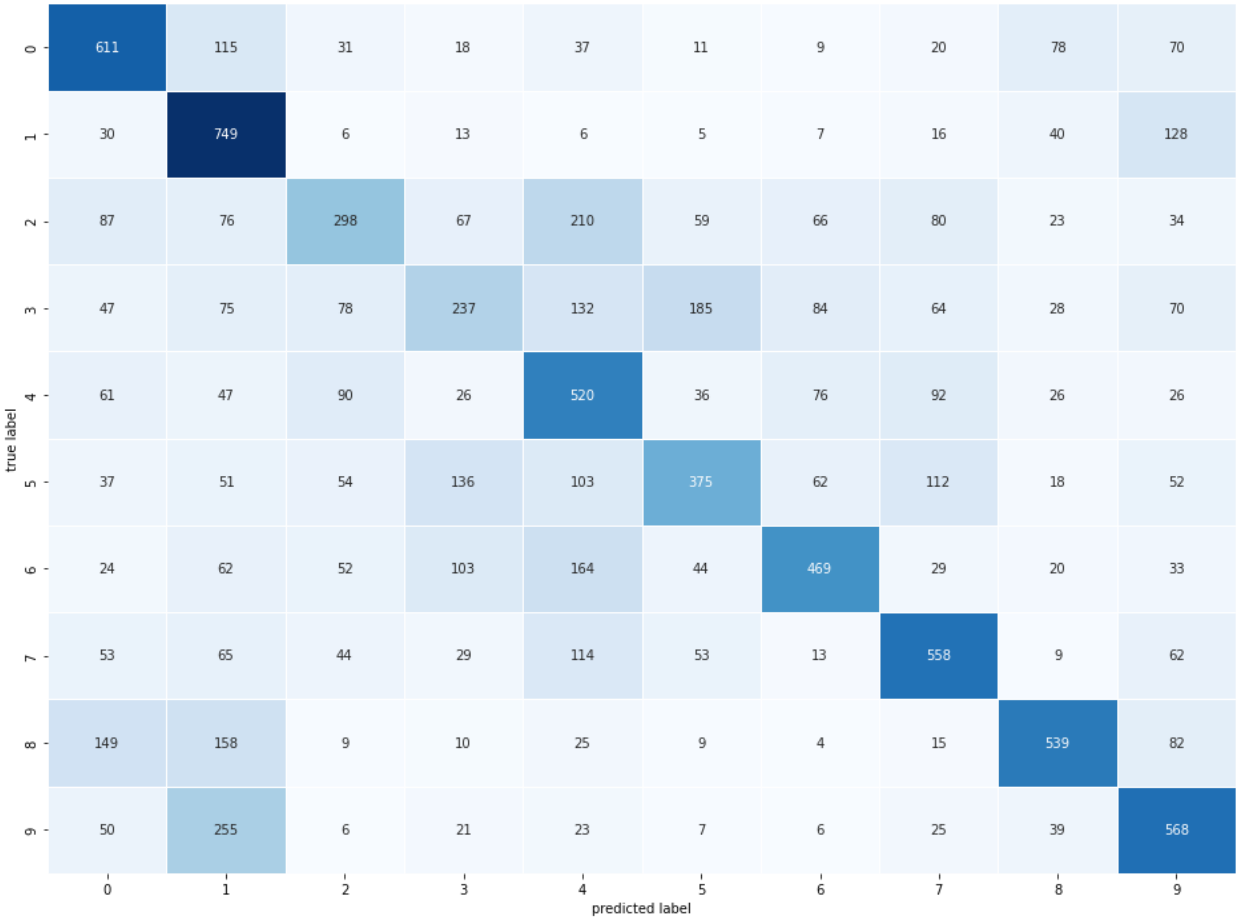
We chart the training and validation accuracy and cross entropy loss below.





With multiple batch normalization processes, validation results start to diverge from the training results after about 10 epochs, showing a slower process in the model overfitting the training data. The resulting test data accuracy is 49.2%, a marginal gain of only 0.3% over the 2-layer DNN model without batch normalization in experiment 1.

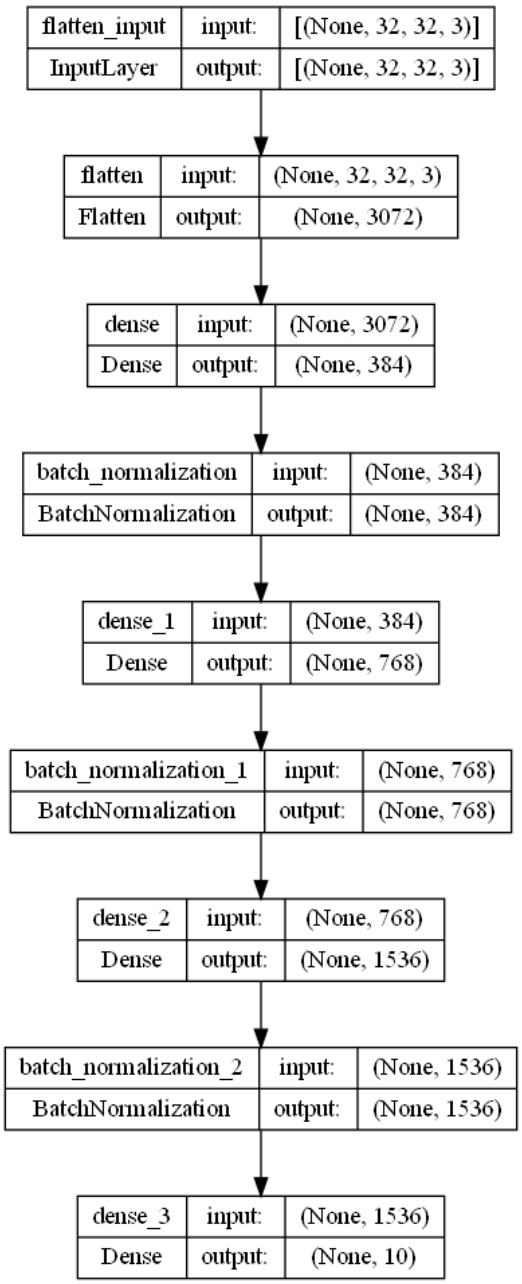
The experiment yields the following confusion matrix:



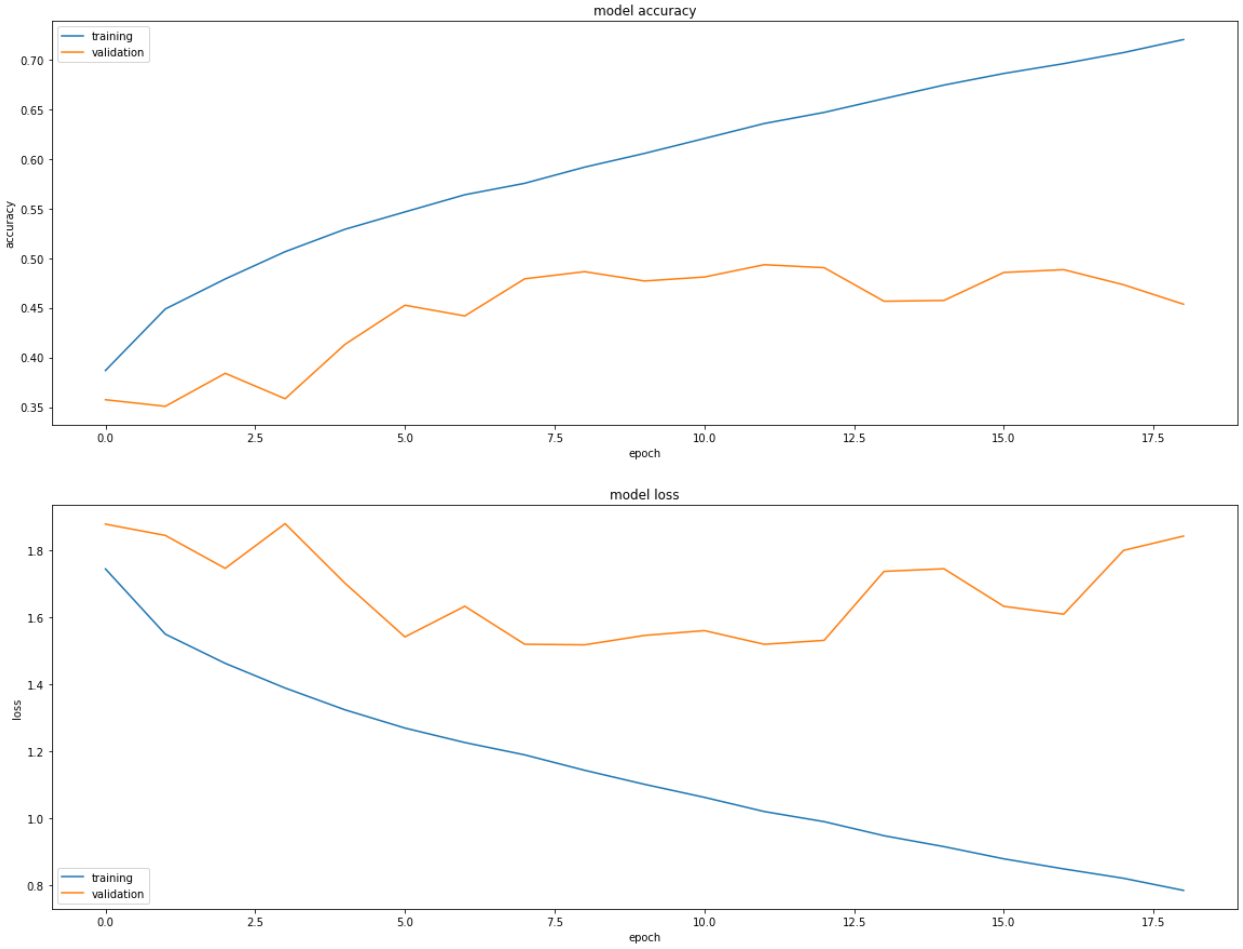
The confusion matrix above shows that while the model can classify most transportation objects (labels 0, 1, 8 and 9), it is still prone to mislabel animal images (labels 2-7) often. That said, adding batch normalization to a DNN-based model only provides a marginal improvement.

Experiment 6: DNN - 3 Fully Connected Layers and Batch Normalization

The sixth experiment is similar to experiment 2, a DNN consisting of 3 fully connected layers but with batch normalization regularization processes in-between them. The first layer contains 384 units, the second layer contains 768 units, the third layer contains 1536 units, and is then fed into the resulting 10-way classification layer.

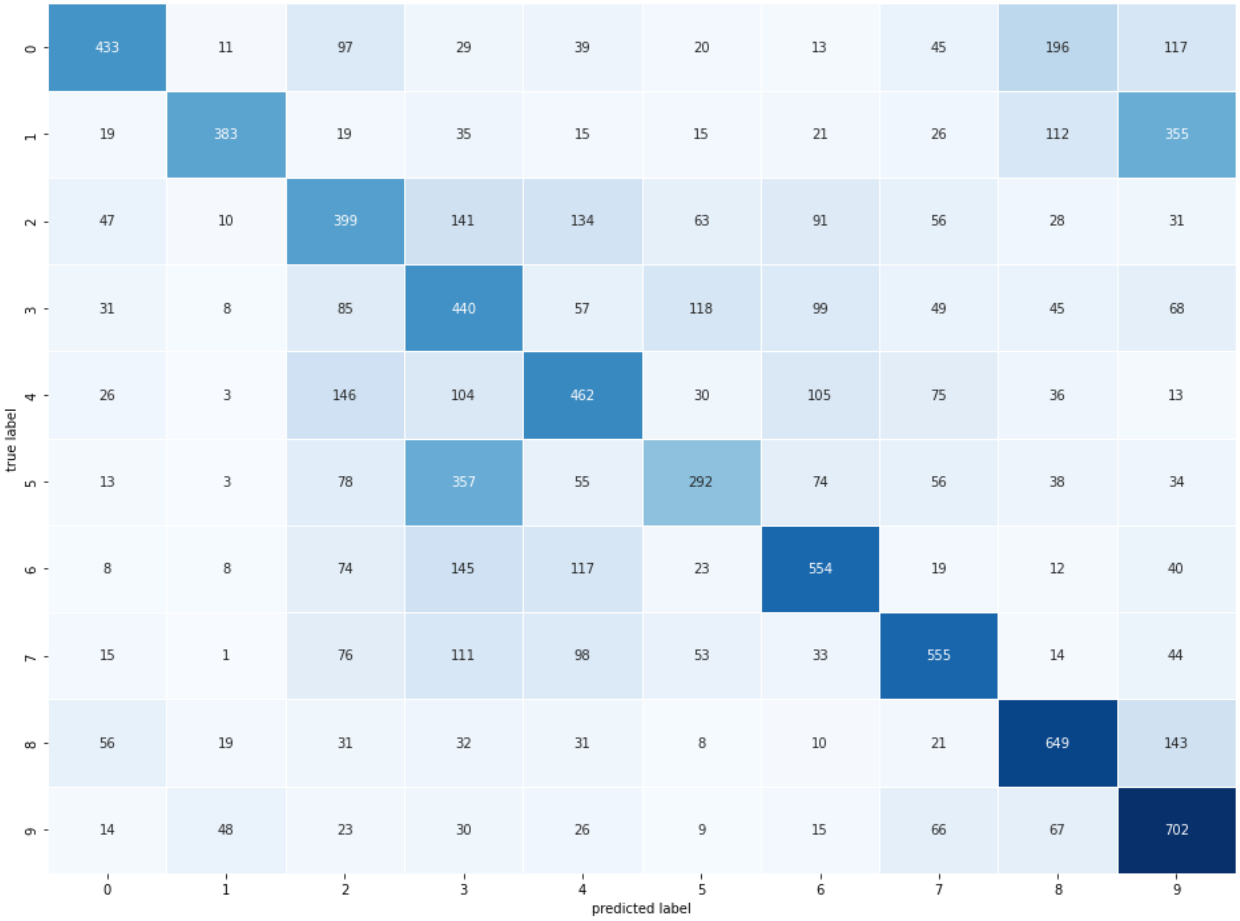


The training and validation accuracy and cross entropy loss charts are the following:



Similar to experiment 5, with multiple batch normalization processes, validation results start to diverge from the training results after about 10 epochs, showing a slower process in which the model overfits the training data. The resulting test data accuracy is 49.4%, a marginal gain of 0.5% over the 3-layer DNN model without batch normalization in experiment 3.

The experiment yields the following confusion matrix:

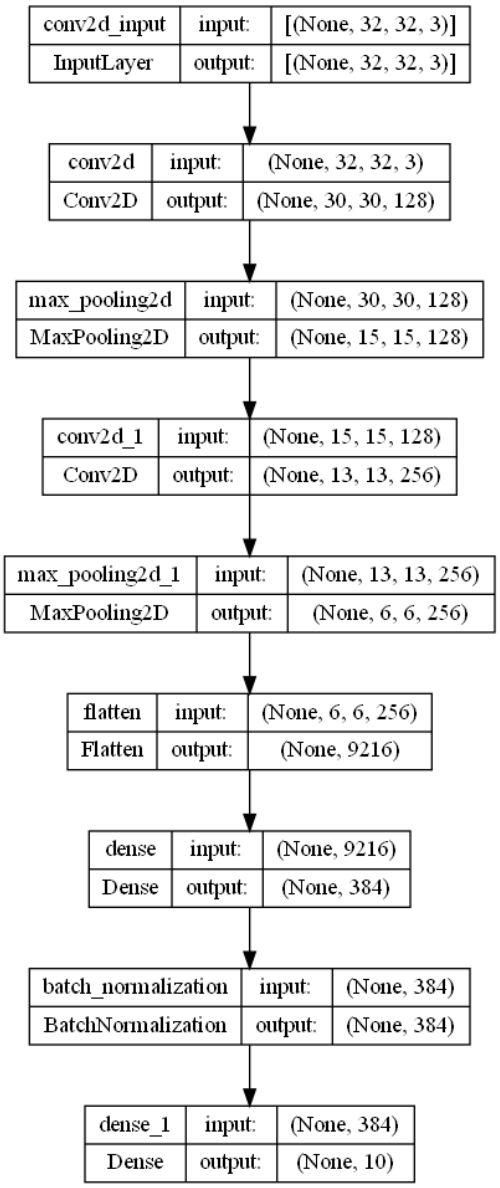


The confusion matrix above shows that while the model tends to be generous classifying most transportation objects as trucks (label 9), and a significant amount of animal images as cats (label 3).

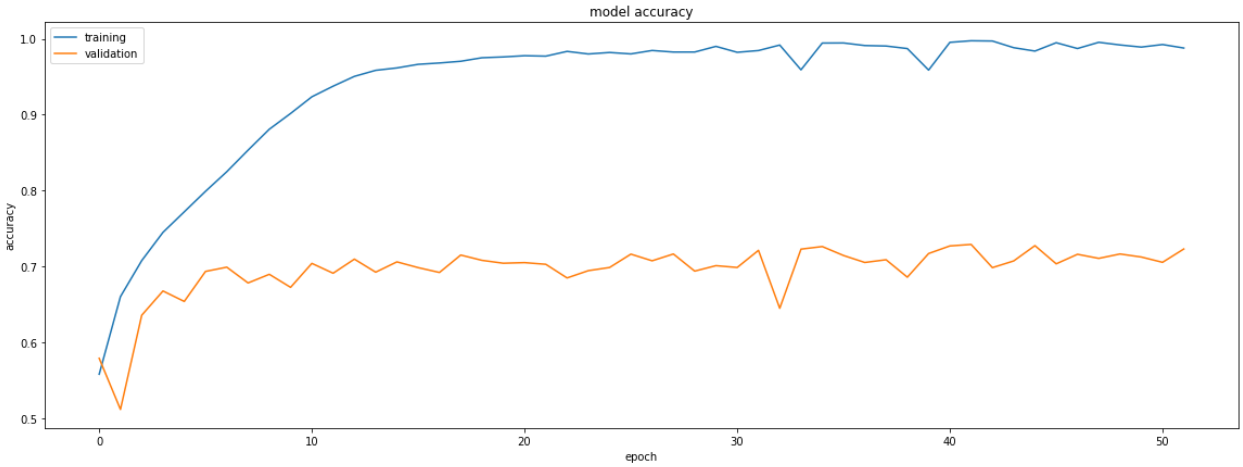
Overall, we see that batch normalization only provides a marginal improvement to DNN-based models.

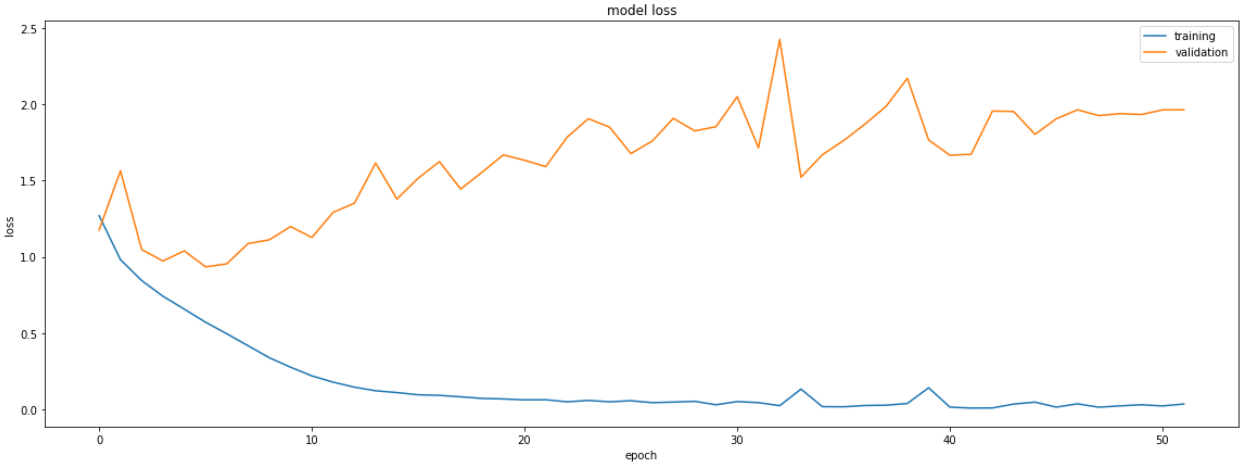
Experiment 7: CNN - 2 Convolutional Layers with 1 Fully Connected Layer and Batch Normalization

Experiment 7 is similar to experiment 3 in which we use a CNN consisting of 2 small 3x3 convolutional layers with the first layer containing 128 filters and the second layer containing 256 filters. The convolutional network base then feeds into a DNN containing 1 fully connected layer with 384 units then a 10-way classification output layer. The batch normalization regularization process follows the fully connected layer.



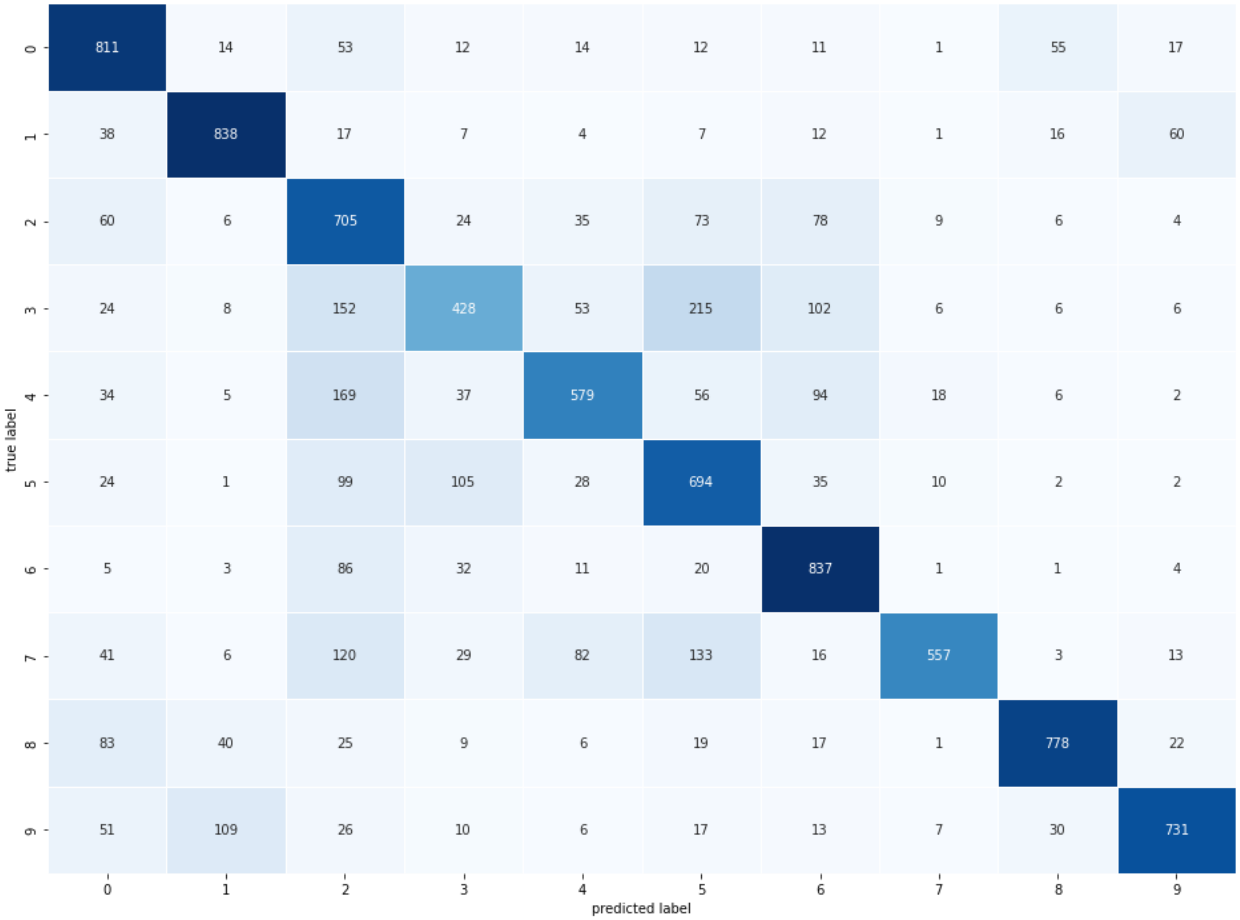
Training and validation accuracy and cross entropy loss are below.





Validation results diverge from the training results after about 5 epochs. The resulting test data accuracy is 69.6%, which is 1.3% worse than the similar model in experiment 3 but without batch normalization.

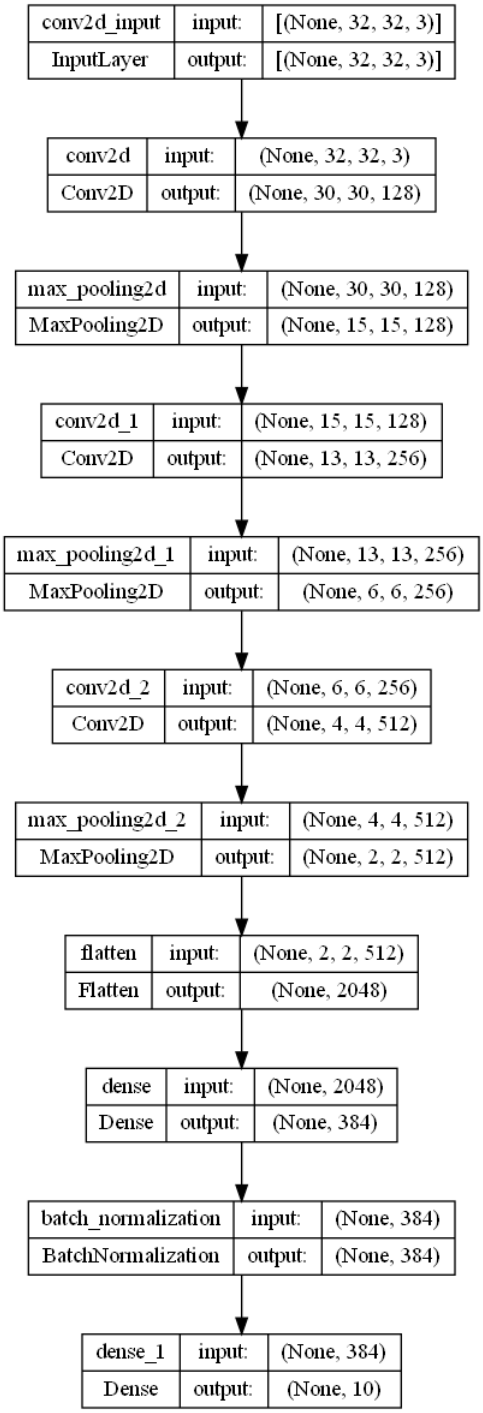
The experiment yields the following confusion matrix:



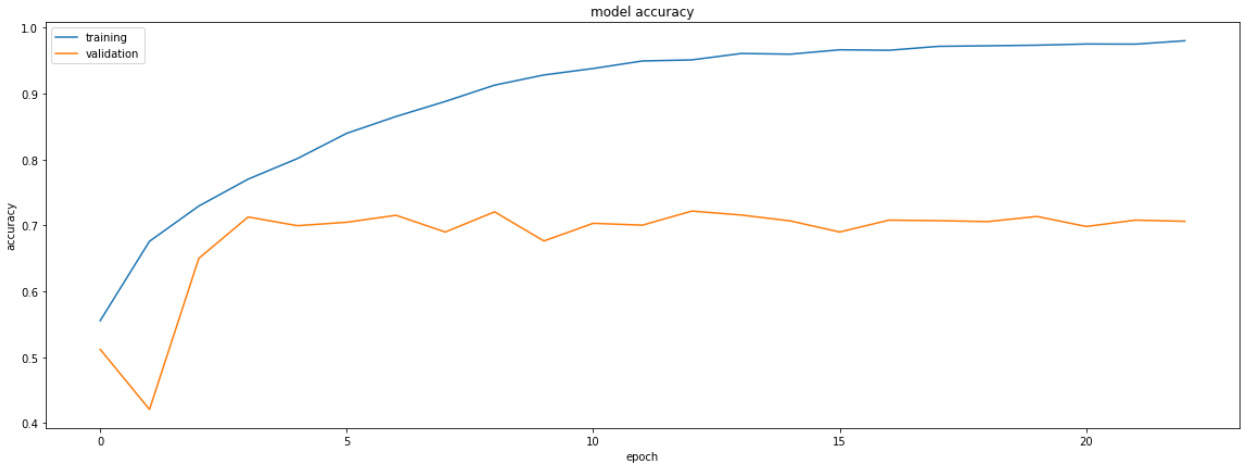
While the model can classify transportation images (labels 0, 1, 8, and 9) and frogs (label 6) fairly well, it can still mislabel other animal images (labels 2-5) often.

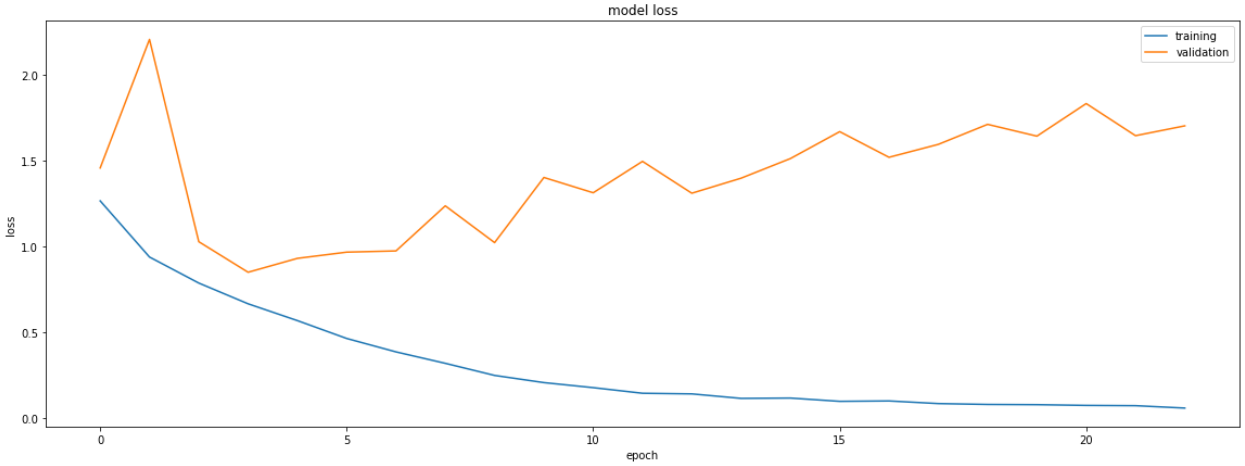
Experiment 8: CNN - 3 Convolutional Layers with 1 Fully Connected Layer and Batch Normalization

Experiment 8 is similar to experiment 4 where we use a CNN consisting of 2 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional network base will then feed into a DNN containing 1 fully connected layer with 384 units then a 10-way classification output layer. The batch normalization regularization process follows the fully connected layer.



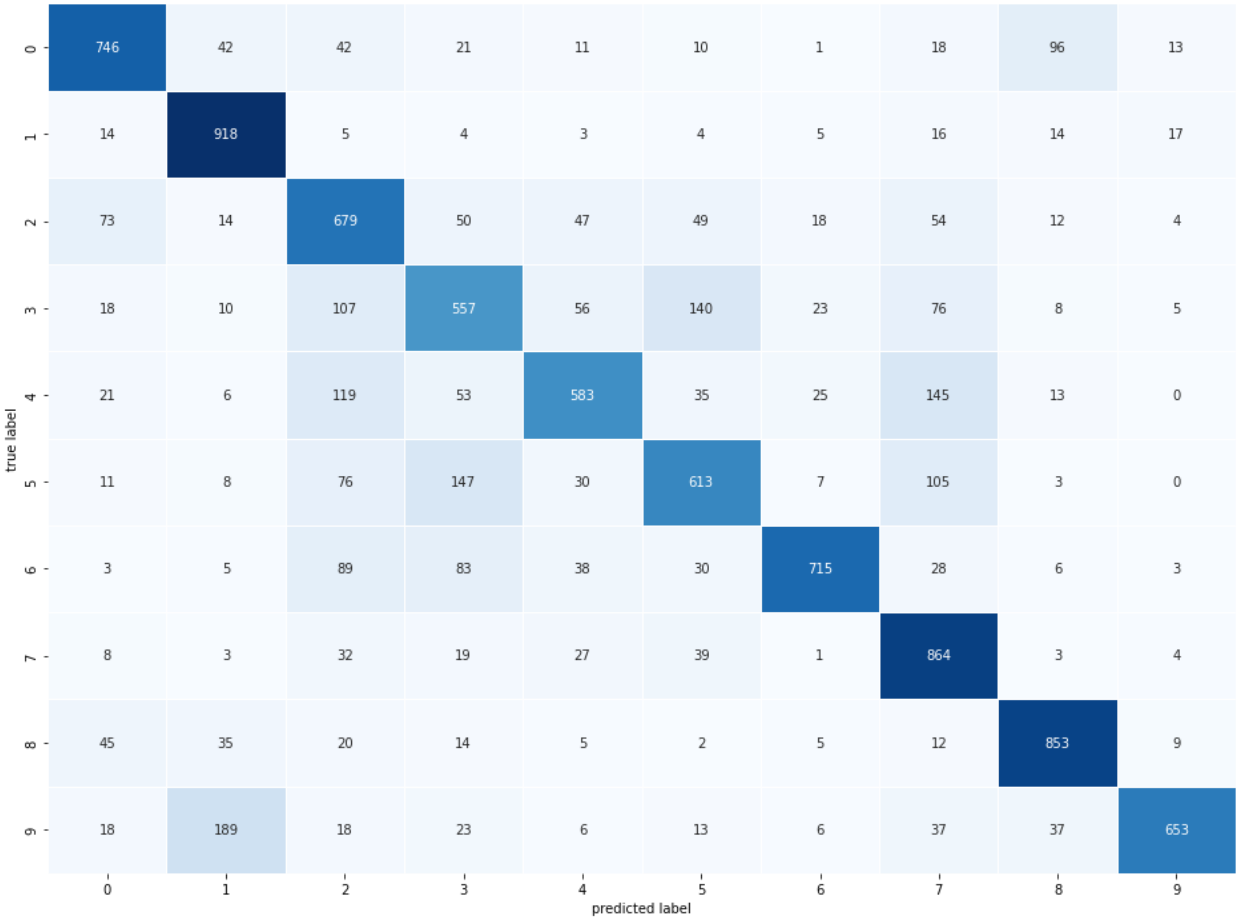
Training and validation accuracy and cross entropy loss are below.





We see validation results start to diverge from the training results after just 5 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was 71.8%, which is 0.6% worse than the model in experiment 4 without batch normalization.

The experiment yields the following confusion matrix:



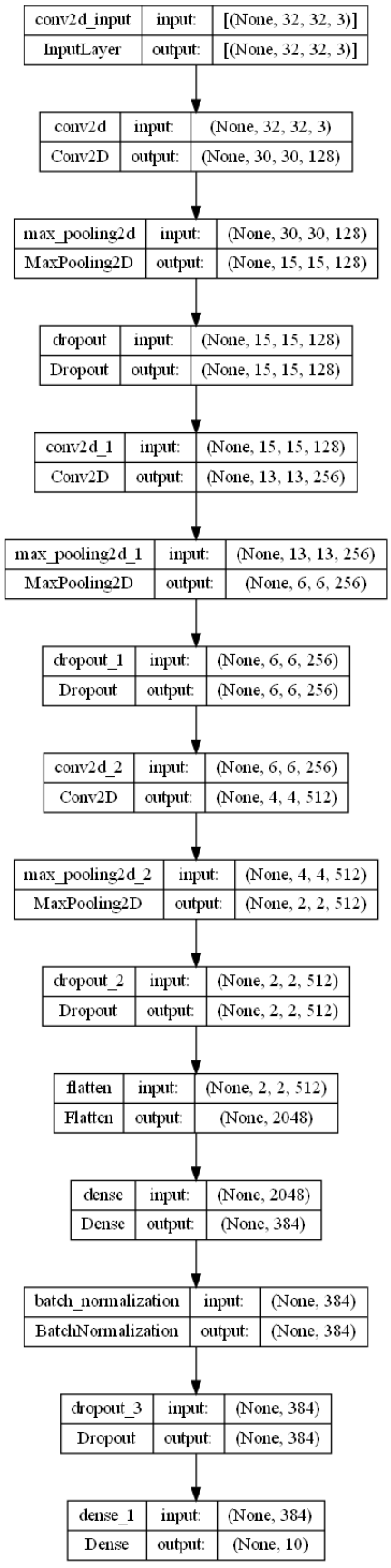
While the model can classify transportation-based images (labels 0, 1, 8, 9) well it still mislabels animal images (labels 2-7) often.

In experiments 5 to 8, we found that adding batch normalization as a regularization technique only yielded marginal gains in accuracy for DNN models but also yielded slightly worse accuracy for CNN models. Given that the CNN model in experiment 4 is the best model that yields the highest accuracy (72.4%) against the test dataset so far, in experiments 9 to 11 we will explore the possibility of using a combination of regularization methods to that model, along with adding more fully connected layers to it as well.

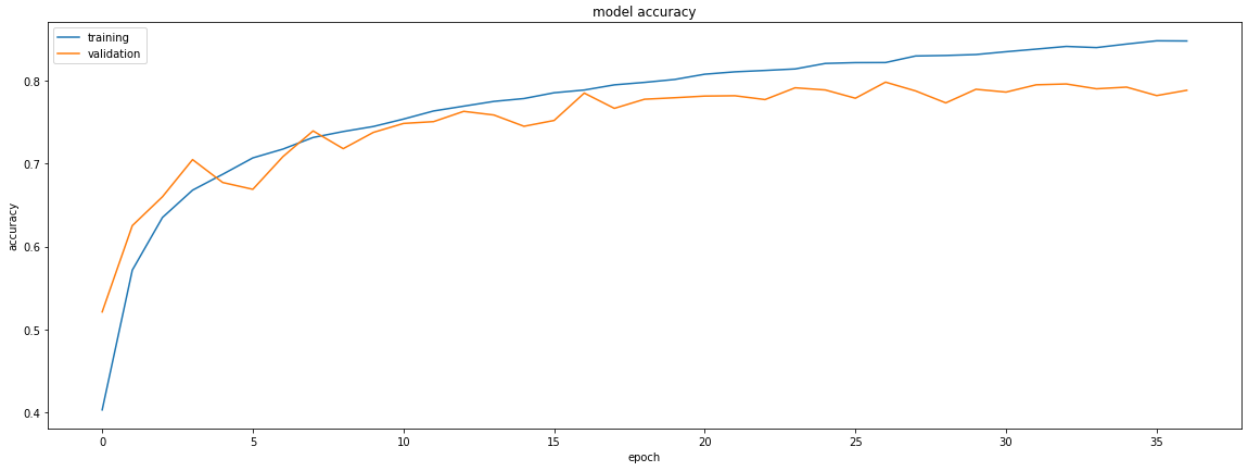
We will add a dropout regularization technique to all convolutional and fully connected layers, in which a percentage of random units in a layer will 'dropout' (i.e. are set to zero) during each training iteration. Doing so will help slow the overfitting process and encourage the model to learn more generalizable features of an image. We will also implement ridge regression (also called 'L2 regularization') on our fully connected layers which adds an additional term to the a model's loss function to penalize parameters with large weights in a layer, and helps discourage a model to be heavily dependent on those parameters overall.

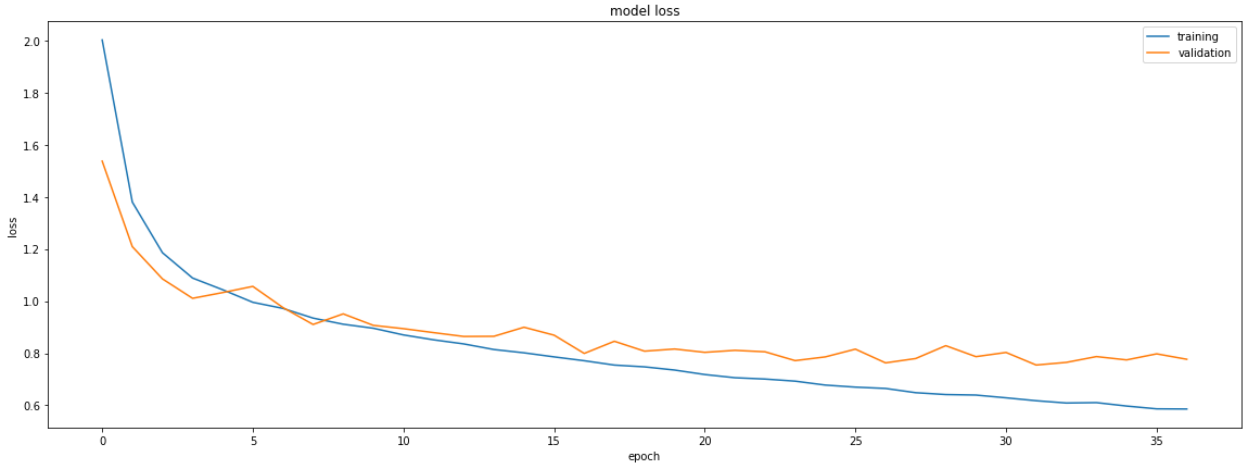
Experiment 9: CNN - 3 Convolutional Layers with 1 Fully Connected Layer and Regularization Combination

In experiment 9 we use a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. This convolutional base also feeds into a DNN containing 1 fully connected layer with 384 units and utilizes L2 regularization (0.001) followed by the 10-way classification output layer. The batch normalization regularization process follows the fully connected layer, while dropout (0.3) processes follow all convolution and fully connected layers.


conv2d_input | input: | [(None, 32, 32, 3)]
InputLayer | output: | [(None, 32, 32, 3)]
conv2d | input: | (None, 32, 32, 3)
Conv2D | output: | (None, 30, 30, 128)
max_pooling2d | input: | (None, 30, 30, 128)
MaxPooling2D | output: | (None, 15, 15, 128)
dropout | input: | (None, 15, 15, 128)
Dropout | output: | (None, 15, 15, 128)
conv2d_1 | input: | (None, 15, 15, 128)
Conv2D | output: | (None, 13, 13, 256)
max_pooling2d_1 | input: | (None, 13, 13, 256)
MaxPooling2D | output: | (None, 6, 6, 256)
dropout_1 | input: | (None, 6, 6, 256)
Dropout | output: | (None, 6, 6, 256)
conv2d_2 | input: | (None, 6, 6, 256)
Conv2D | output: | (None, 4, 4, 512)
max_pooling2d_2 | input: | (None, 4, 4, 512)
MaxPooling2D | output: | (None, 2, 2, 512)
dropout_2 | input: | (None, 2, 2, 512)
Dropout | output: | (None, 2, 2, 512)
flatten | input: | (None, 2, 2, 512)
Flatten | output: | (None, 2048)
dense | input: | (None, 2048)
Dense | output: | (None, 384)
batch_normalization | input: | (None, 384)
BatchNormalization | output: | (None, 384)
dropout_3 | input: | (None, 384)
Dropout | output: | (None, 384)
dense_1 | input: | (None, 384)
Dense | output: | (None, 10)

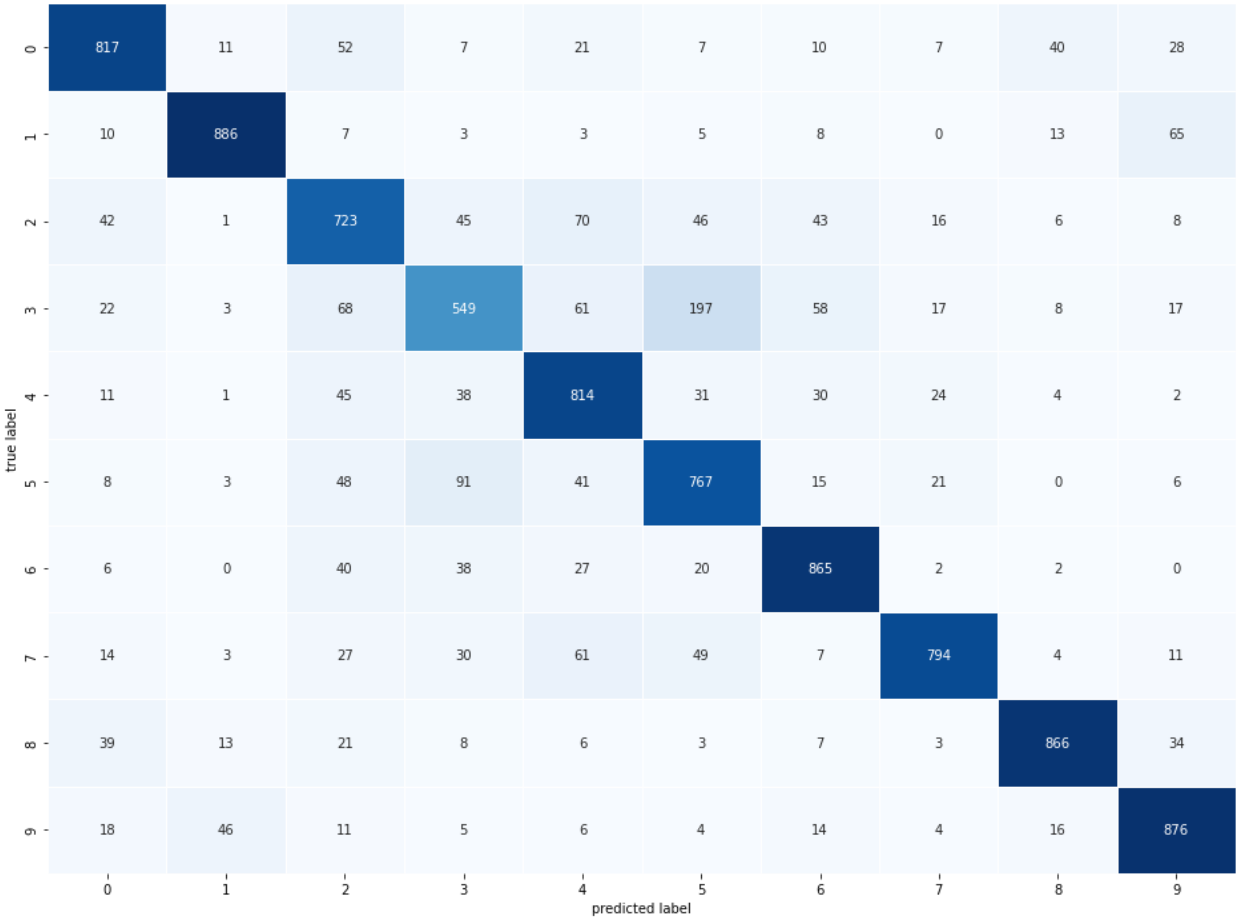
Training and validation accuracy and cross entropy loss are below.





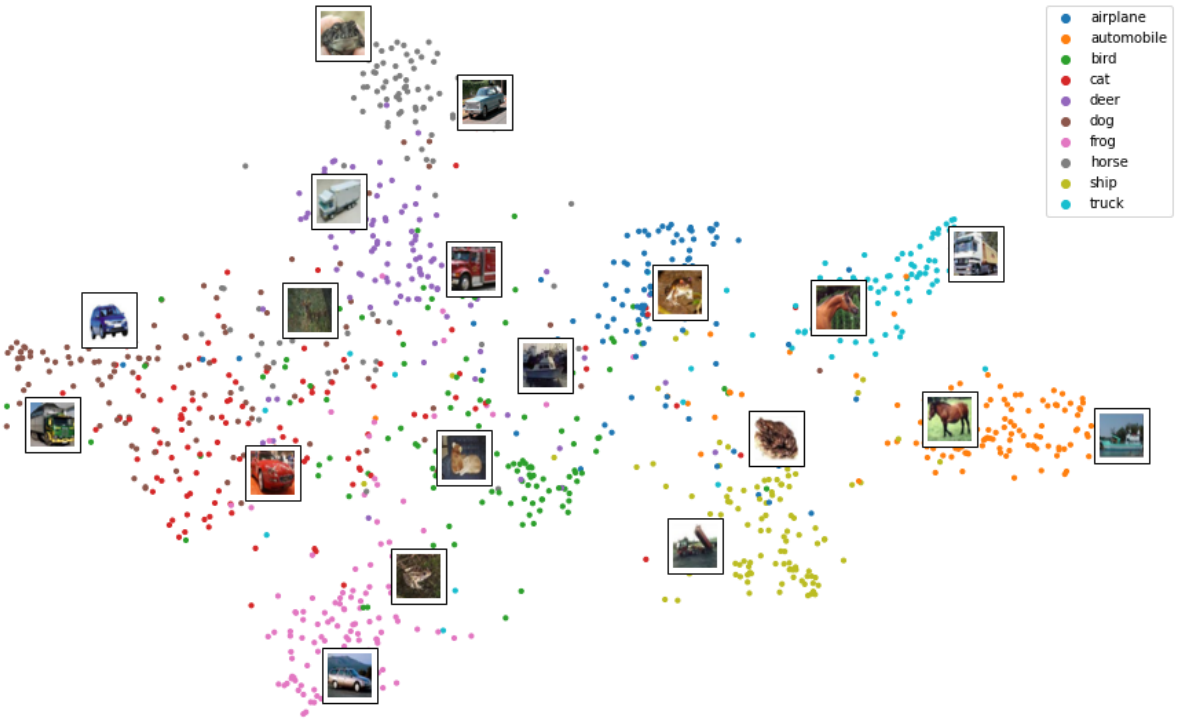
We see validation results start to diverge from the training results after 15 epochs, indicating signs of the model overfitting to the training data. The resulting test data accuracy was 79.6%, a significant improvement of about 7.2% over experiment 4.

The experiment yields the following confusion matrix:



From the matrix above we can see a general improvement in both transportation and animal image classification well, but there are still discrepancies between cats (label 3) and dogs (label 5) classifications.

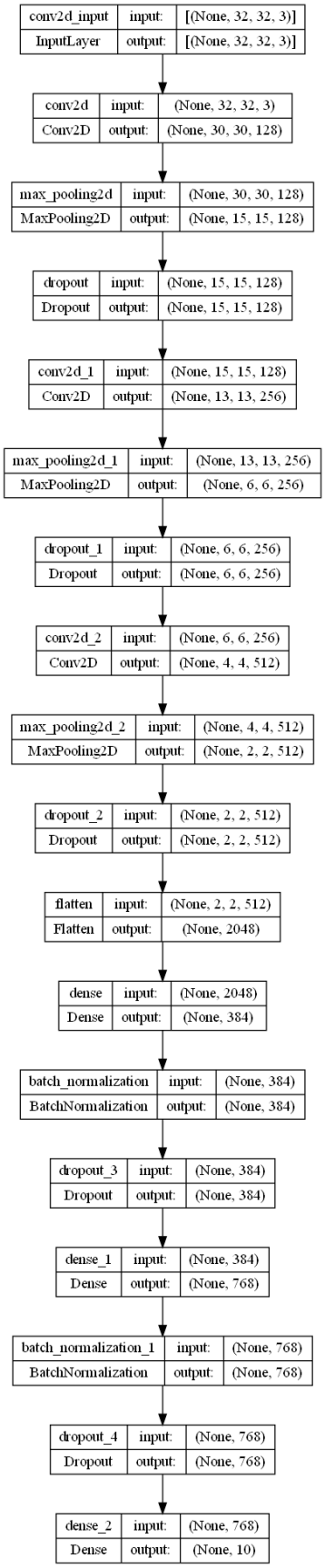
The scatterplot sample yields the following:



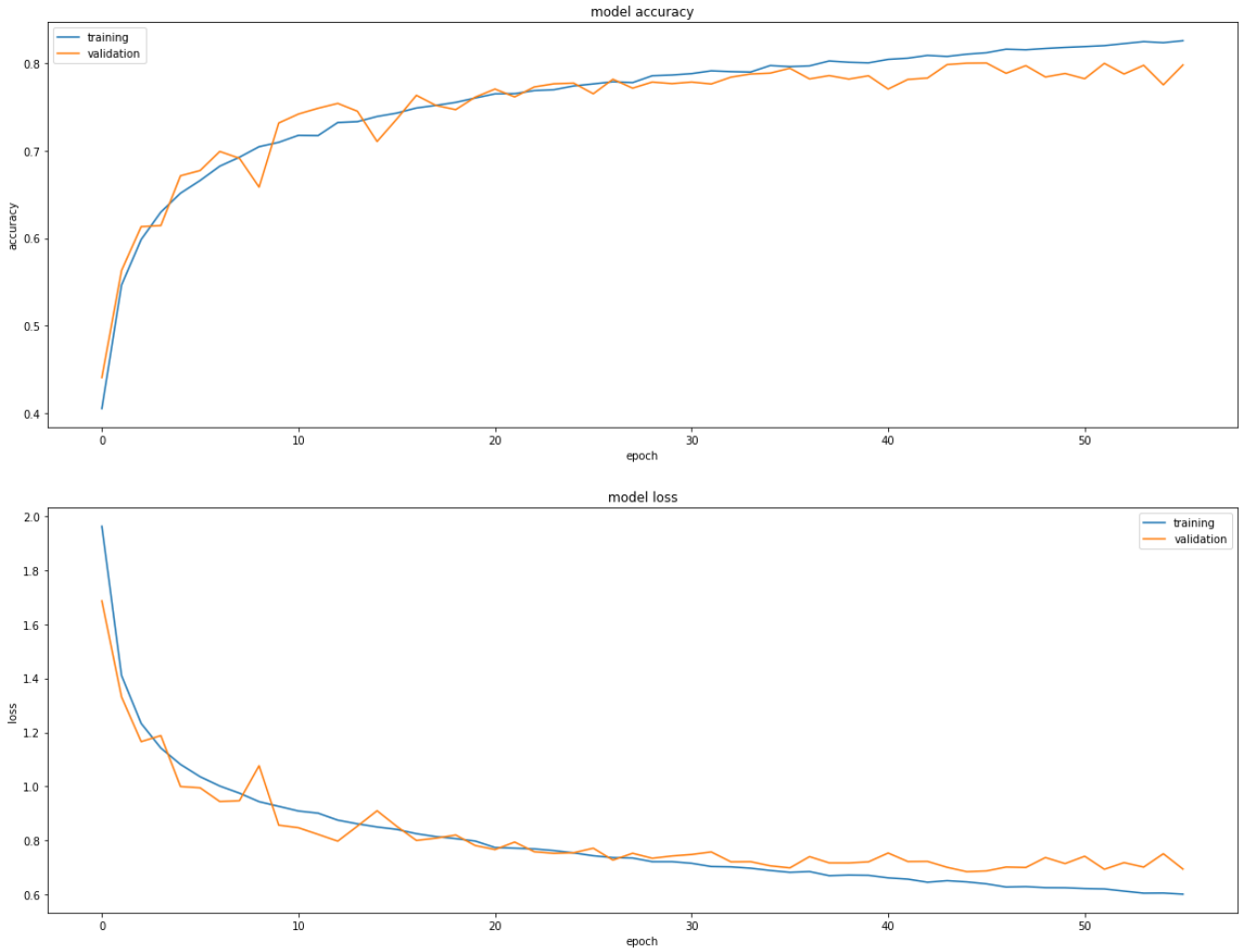
We can see distinct groups start to form in this scatterplot sample.

Experiment 10: CNN - 3 Convolutional Layers with 2 Fully Connected Layers and Regularization Combination

Similar to experiment 9 we use a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.3) processes follow all convolution and fully connected layers. All fully connected layers utilize L2 regularization (0.001).

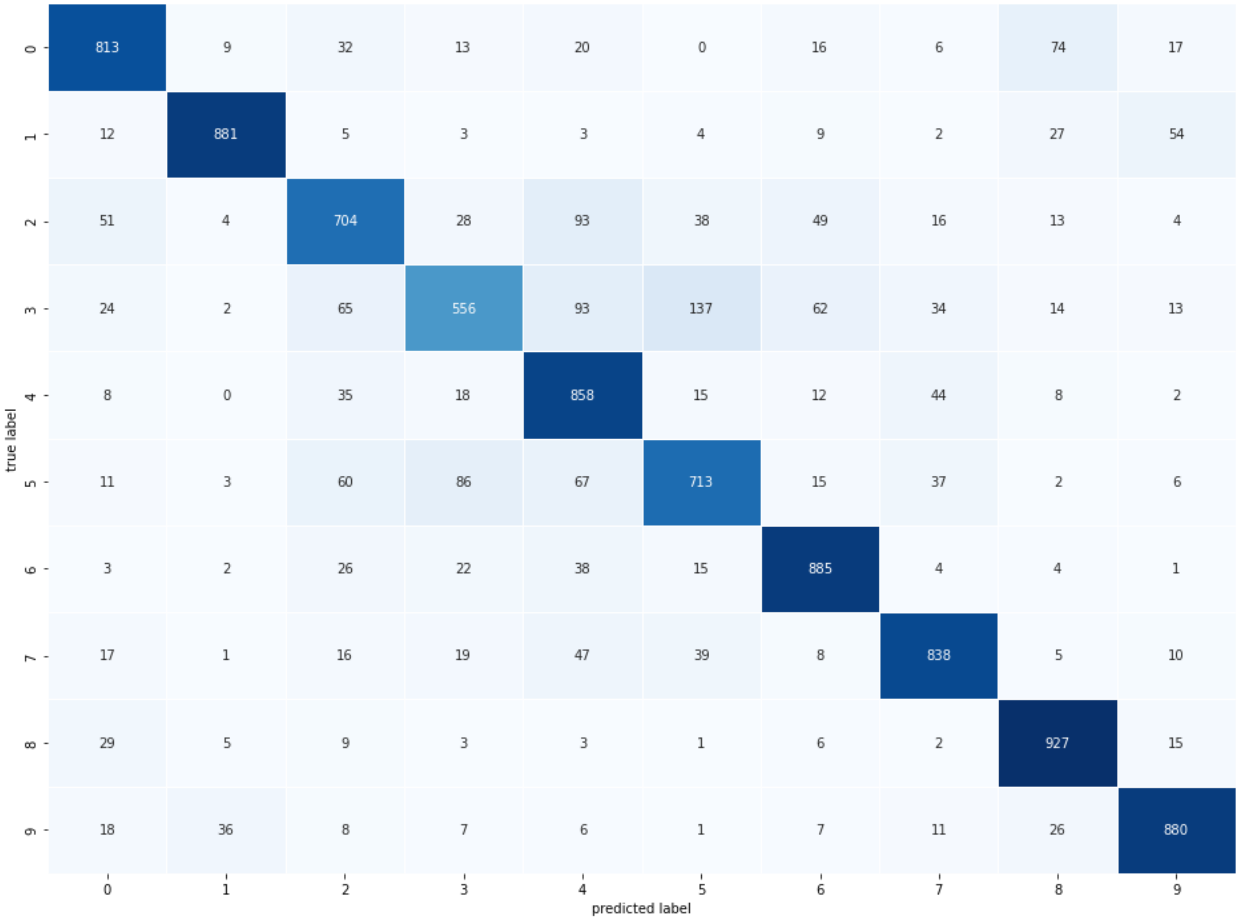

conv2d_input | input: | [(None, 32, 32, 3)]
InputLayer | output: | [(None, 32, 32, 3)]
conv2d | input: | (None, 32, 32, 3)
Conv2D | output: | (None, 30, 30, 128)
max_pooling2d | input: | (None, 30, 30, 128)
MaxPooling2D | output: | (None, 15, 15, 128)
dropout | input: | (None, 15, 15, 128)
Dropout | output: | (None, 15, 15, 128)
conv2d_1 | input: | (None, 15, 15, 128)
Conv2D | output: | (None, 13, 13, 256)
max_pooling2d_1 | input: | (None, 13, 13, 256)
MaxPooling2D | output: | (None, 6, 6, 256)
dropout_1 | input: | (None, 6, 6, 256)
Dropout | output: | (None, 6, 6, 256)
conv2d_2 | input: | (None, 6, 6, 256)
Conv2D | output: | (None, 4, 4, 512)
max_pooling2d_2 | input: | (None, 4, 4, 512)
MaxPooling2D | output: | (None, 2, 2, 512)
dropout_2 | input: | (None, 2, 2, 512)
Dropout | output: | (None, 2, 2, 512)
flatten | input: | (None, 2, 2, 512)
Flatten | output: | (None, 2048)
dense | input: | (None, 2048)
Dense | output: | (None, 384)
batch_normalization | input: | (None, 384)
BatchNormalization | output: | (None, 384)
dropout_3 | input: | (None, 384)
Dropout | output: | (None, 384)
dense_1 | input: | (None, 384)
Dense | output: | (None, 768)
batch_normalization_1 | input: | (None, 768)
BatchNormalization | output: | (None, 768)
dropout_4 | input: | (None, 768)
Dropout | output: | (None, 768)
dense_2 | input: | (None, 768)
Dense | output: | (None, 10)

Training and validation accuracy and cross entropy loss are below.



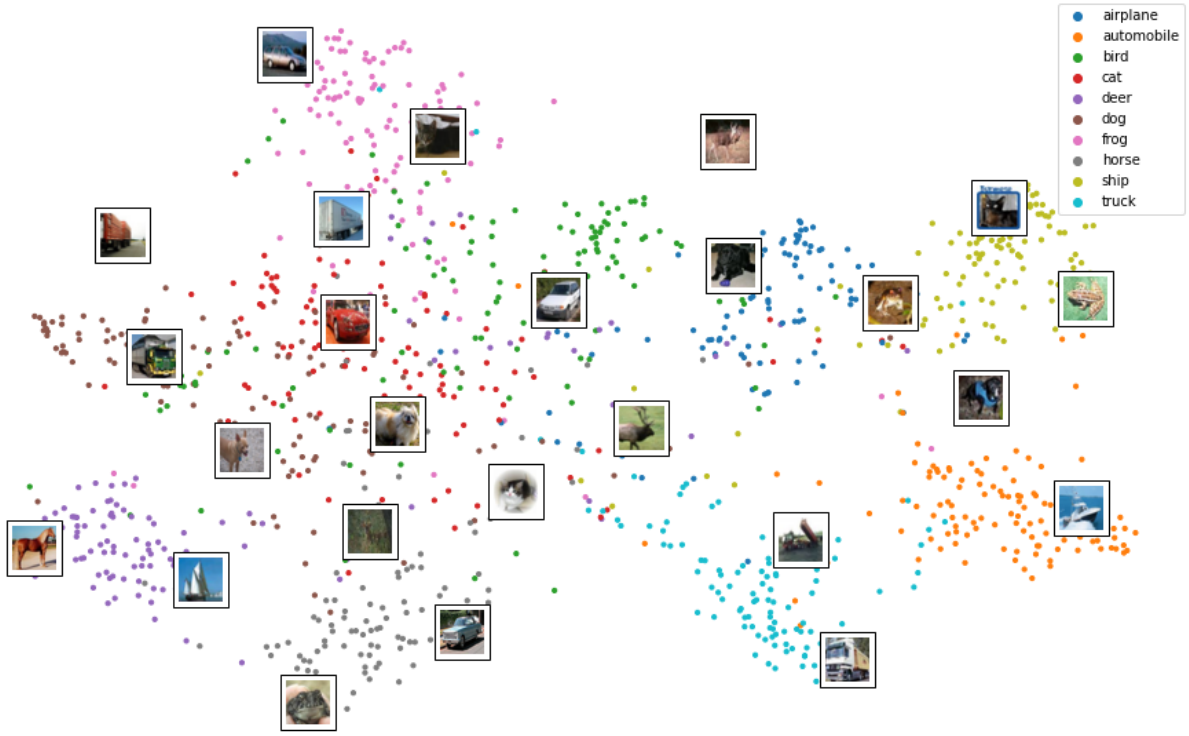
We see validation results start to diverge from the training results after 35 epochs. The chart shows the slower overfitting process due to the various regularization processes applied to the convolutional and fully connected layers. The resulting test data accuracy is 81%, a 1.4% improvement over experiment 9.

The experiment yields the following confusion matrix:



From the matrix above we can see an improvement in both transportation and animal image classification overall, but there are still discrepancies between cats (label 3) and dogs (label 5) classifications, while animal images can often be classified as deer (label 4).

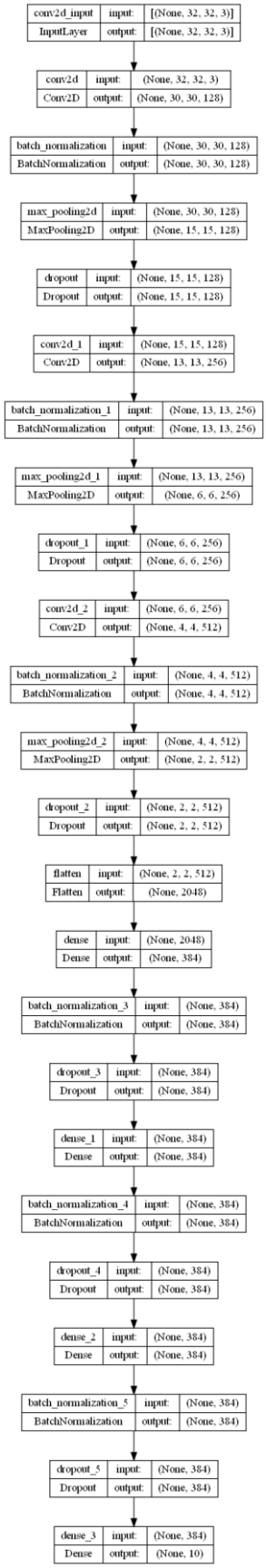
The scatterplot sample yields the following:



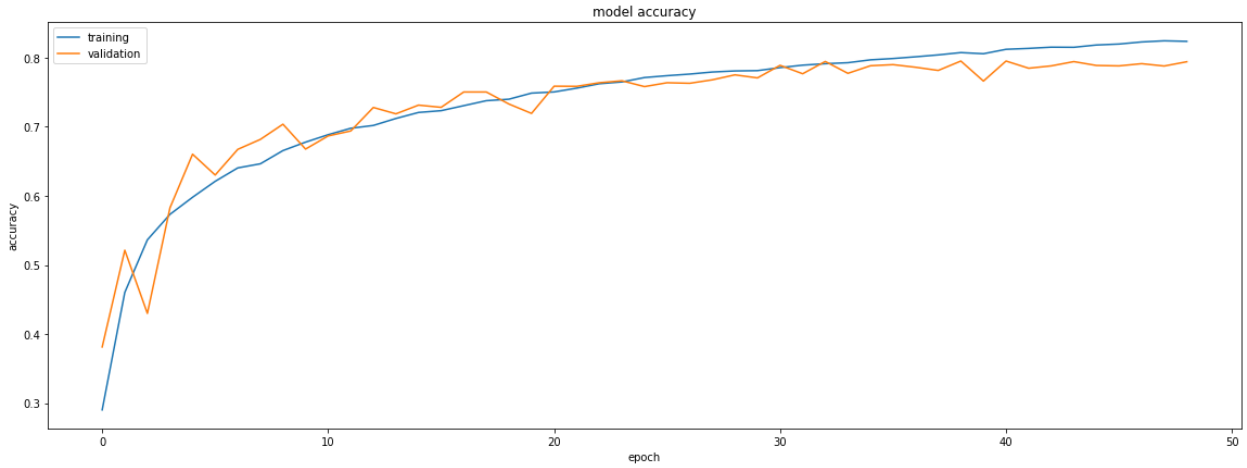
We see distinct groups in the scatterplot sample, particularly with transportation images.

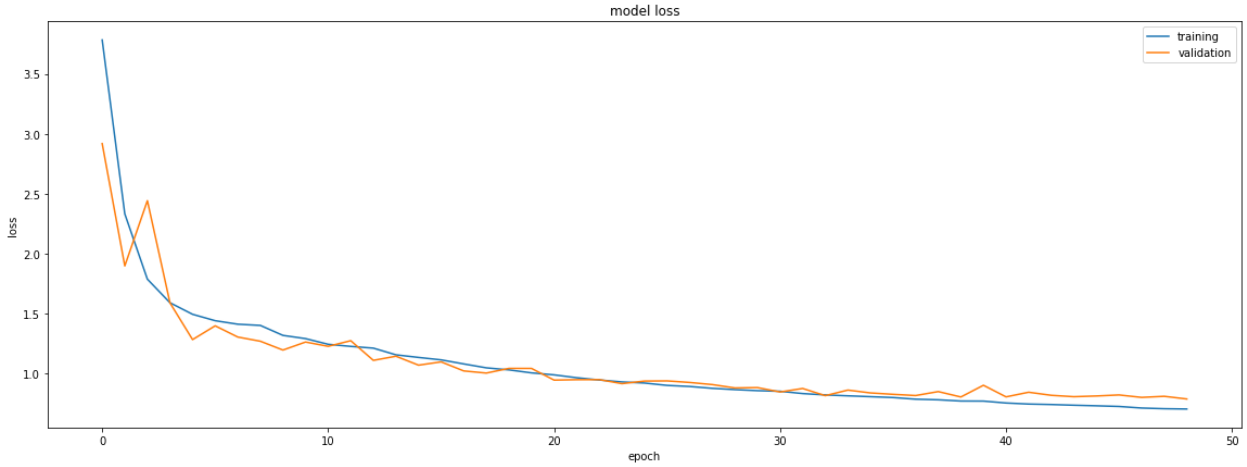
Experiment 11: CNN - 3 Convolutional Layers with 3 Fully Connected Layers and Regularization Combination

We use a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units, the second layer containing 768 units, and the third layer containing 1536 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.3) processes follow all convolution and fully connected layers. All fully connected layers utilize L2 regularization (0.001).


conv2d_input | input: | [(None, 32, 32, 3)]
InputLayer | output: | [(None, 32, 32, 3)]
conv2d | input: | (None, 32, 32, 3)
Conv2D | output: | (None, 30, 30, 128)
batch_normalization | input: | (None, 30, 30, 128)
BatchNormalization | output: | (None, 30, 30, 128)
max_pooling2d | input: | (None, 30, 30, 128)
MaxPooling2D | output: | (None, 15, 15, 128)
dropout | input: | (None, 15, 15, 128)
Dropout | output: | (None, 15, 15, 128)
conv2d_1 | input: | (None, 15, 15, 128)
Conv2D | output: | (None, 13, 13, 256)
batch_normalization_1 | input: | (None, 13, 13, 256)
BatchNormalization | output: | (None, 13, 13, 256)
max_pooling2d_1 | input: | (None, 13, 13, 256)
MaxPooling2D | output: | (None, 6, 6, 256)
dropout_1 | input: | (None, 6, 6, 256)
Dropout | output: | (None, 6, 6, 256)
conv2d_2 | input: | (None, 6, 6, 256)
Conv2D | output: | (None, 4, 4, 512)
batch_normalization_2 | input: | (None, 4, 4, 512)
BatchNormalization | output: | (None, 4, 4, 512)
max_pooling2d_2 | input: | (None, 4, 4, 512)
MaxPooling2D | output: | (None, 2, 2, 512)
dropout_2 | input: | (None, 2, 2, 512)
Dropout | output: | (None, 2, 2, 512)
flatten | input: | (None, 2, 2, 512)
Flatten | output: | (None, 2048)
dense | input: | (None, 2048)
Dense | output: | (None, 384)
batch_normalization_3 | input: | (None, 384)
BatchNormalization | output: | (None, 384)
dropout_3 | input: | (None, 384)
Dropout | output: | (None, 384)
dense_1 | input: | (None, 384)
Dense | output: | (None, 384)
batch_normalization_4 | input: | (None, 384)
BatchNormalization | output: | (None, 384)
dropout_4 | input: | (None, 384)
Dropout | output: | (None, 384)
dense_2 | input: | (None, 384)
Dense | output: | (None, 384)
batch_normalization_5 | input: | (None, 384)
BatchNormalization | output: | (None, 384)
dropout_5 | input: | (None, 384)
Dropout | output: | (None, 384)
dense_3 | input: | (None, 384)
Dense | output: | (None, 10)

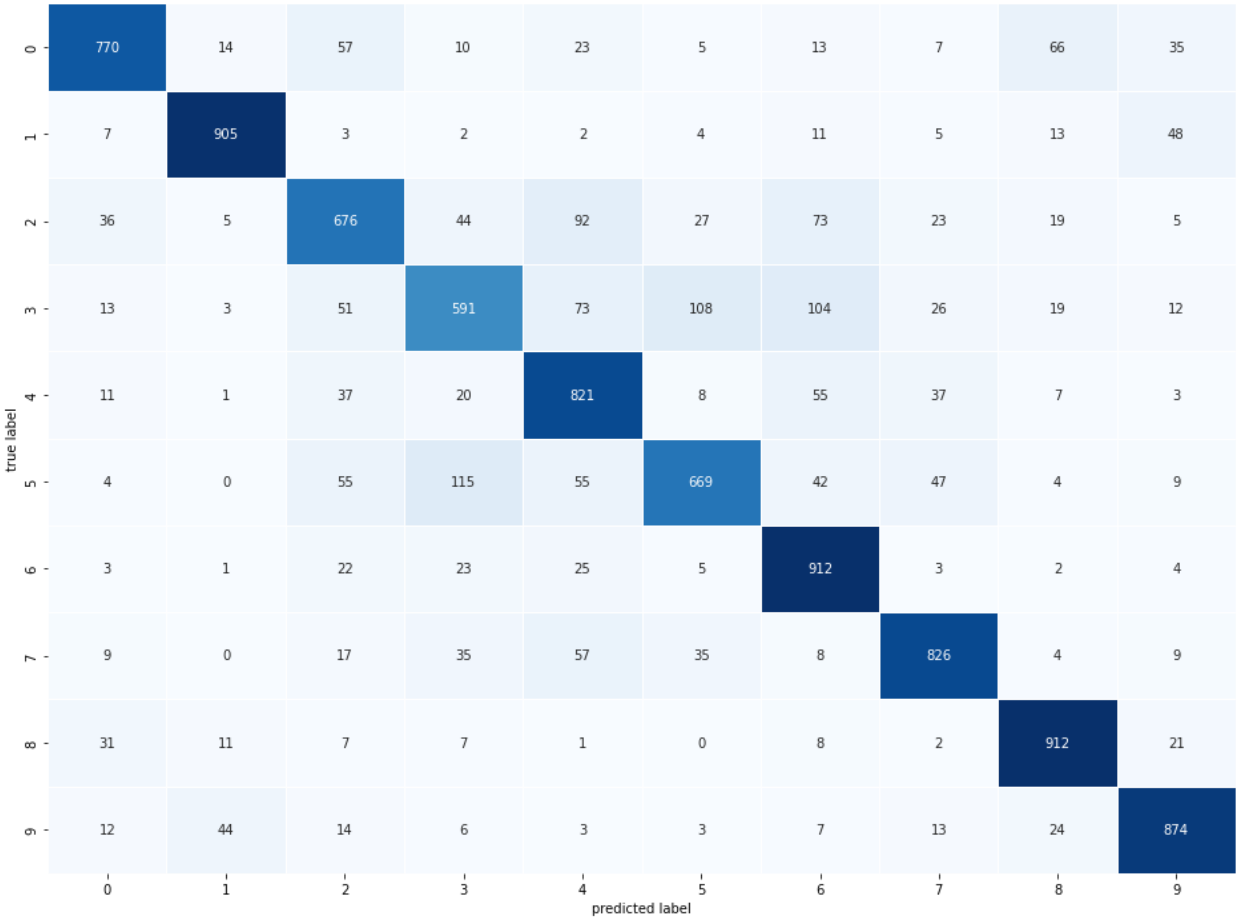
Training and validation accuracy and cross entropy loss are below.





We see validation results start to diverge from the training results after 35 epochs. The chart shows the slower overfitting process due to the various regularization processes applied to the convolutional and fully connected layers. The resulting test data accuracy is 79.6%, which is 1.4% worse compared to experiment 10, and about the same performance to experiment 9.

The experiment yields the following confusion matrix:



From the matrix above we see the model performs transportation image classification well, there are still discrepancies between cats (label 3) and dogs (label 5) classifications.

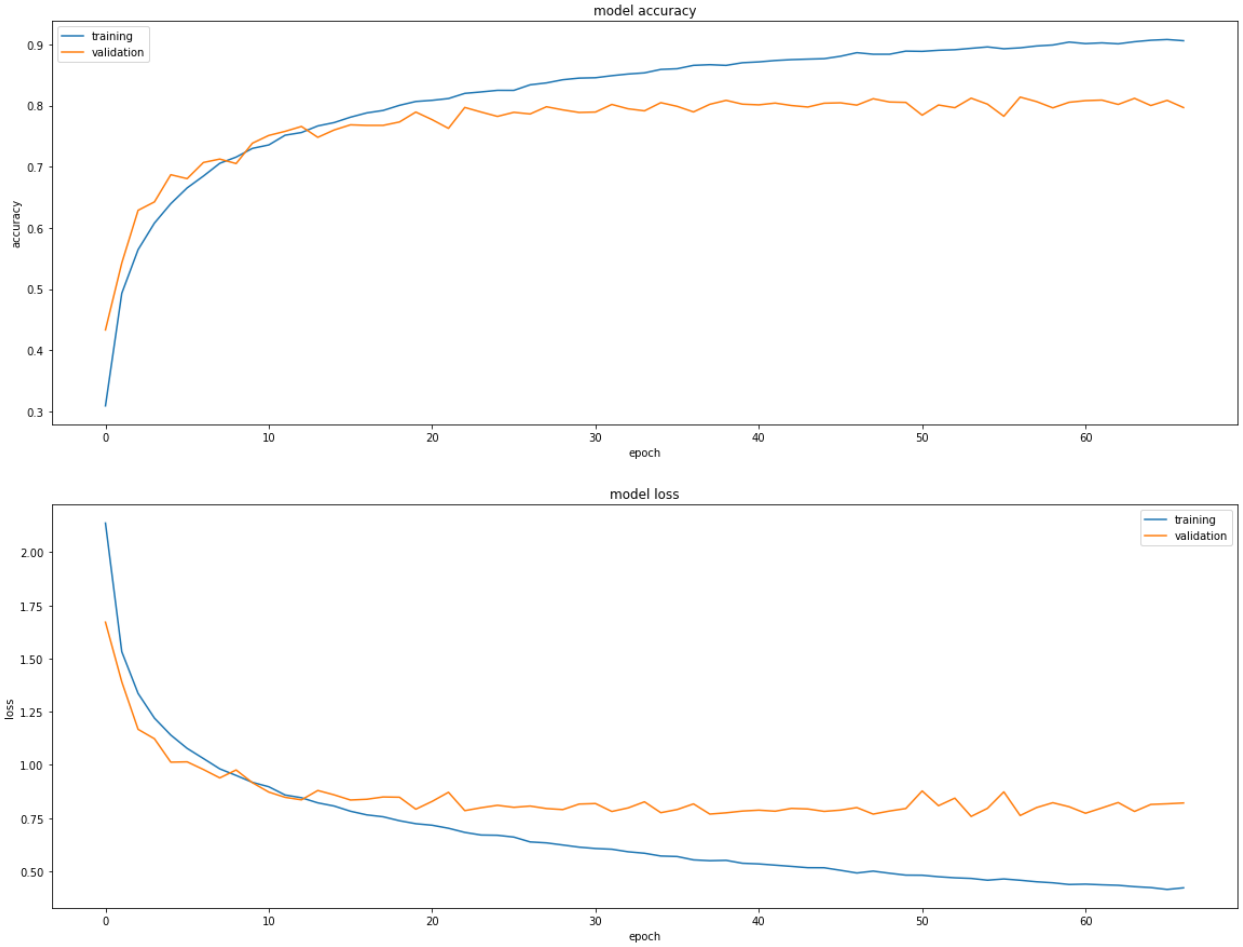
At this point, we're starting to find the limits to the amount of fully connected layers we can add to a CNN in which more fully connected layers can result in diminishing returns. The addition of over 1 million parameters compared to experiment 10 yields 1.4% less accuracy. Despite these findings, we have other options to explore to hopefully yield improved accuracy by configuring L2 and dropout regularization hyperparamenters, which we will perform in experiments 12 to 15. We will use the model from experiment 10 for hyperparameter configuration, as it yields the best accuracy against the test dataset so far at 81%.

Experiment 12: CNN - 3 Convolutional Layers with 2 Fully Connected Layers and Regularization Combination (L2 Regularization Hyperparameter Tweak Low)

We model a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.3) processes follow all

convolution and fully connected layers. All fully connected layers utilize L2 regularization configured with a learning rate of 0.0001 (model diagram is the same as in Experiment 10).

Training and validation accuracy and cross entropy loss are below.



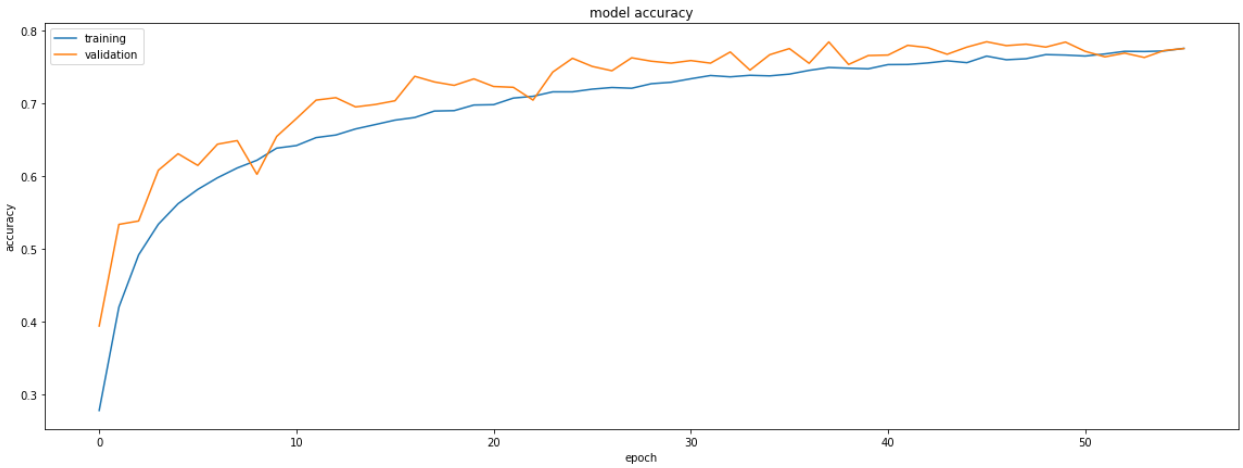
We see validation results start to diverge from the training results after 20 epochs. With a lower L2 regularization learning rate the overfitting occurs faster compared to the model in experiment 10. The resulting test data accuracy is 80.5%, which is 0.5% worse than experiment 10.

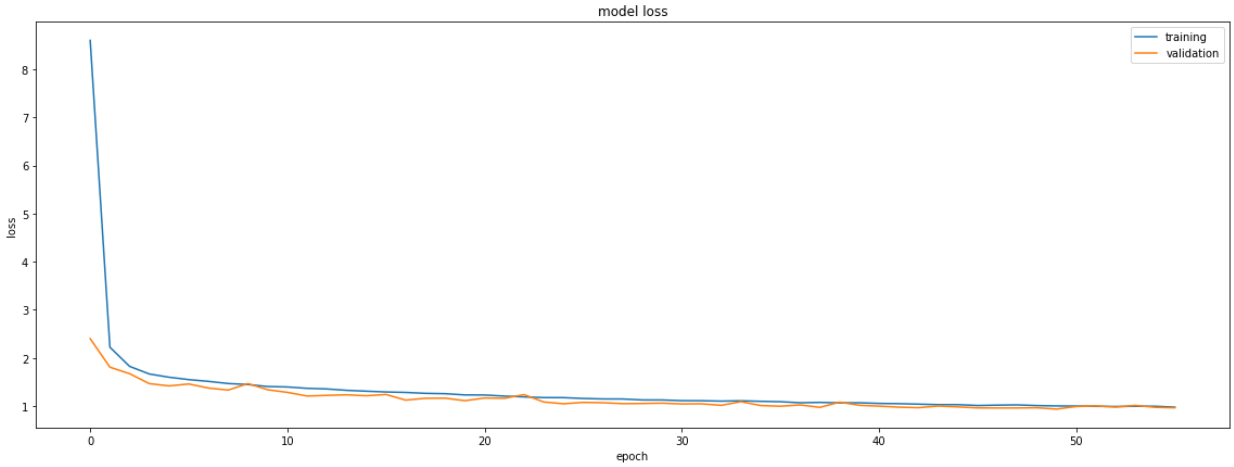
Experiment 13: CNN - 3 Convolutional Layers with 2 Fully Connected Layers and Regularization Combination (L2 Regularization Hyperparameter Tweak High)

We model a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.3) processes follow all

convolution and fully connected layers. All fully connected layers utilize L2 regularization configured with a learning rate of 0.01 (model diagram is the same as in Experiment 10).

Training and validation accuracy and cross entropy loss are below.





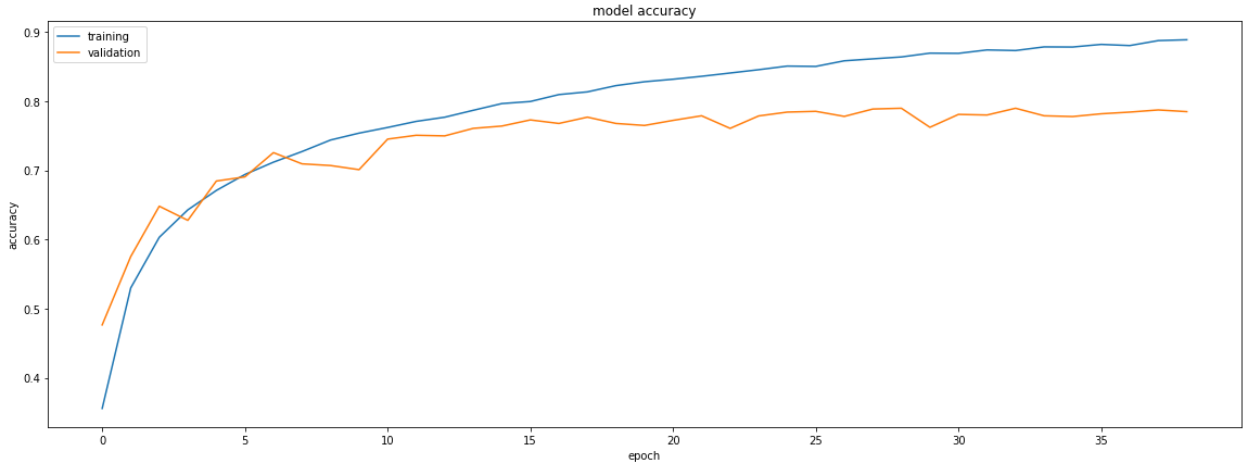
We see validation results stabilize with the training results after 50 epochs. The higher L2 regularization learning rate considerably slows the overfit process. That said, the resulting test data accuracy is 80.4%, performing 0.6% worse than experiment 10.

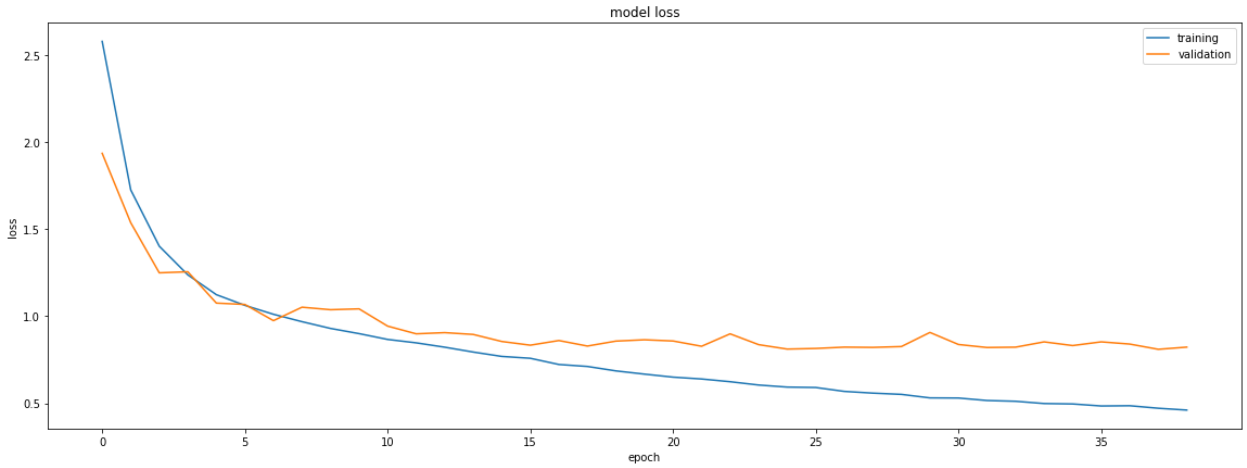
While we are able to vary the speed of the overfitting process with an L2 learning rate value over and under our initial value of 0.001 in experiment 10, doing so yields slightly worse accuracy each way. We will stay with the 0.001 L2 learning rate as we perform different configurations of dropout in experiments 14 and 15.

Experiment 14: CNN - 3 Convolutional Layers with 2 Fully Connected Layers and Regularization Combination (Dropout Hyperparameter Tweak Low)

We model a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.2) processes follow all convolution and fully connected layers. All fully connected layers utilize L2 regularization configured with a learning rate of 0.001 (model diagram is the same as in Experiment 10).

Training and validation accuracy and cross entropy loss are below.





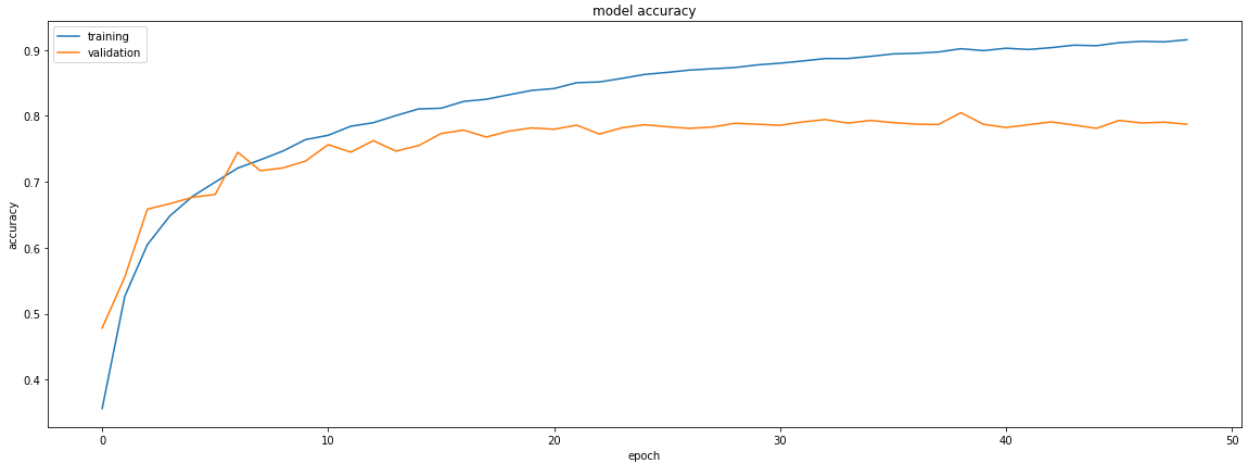
We see validation results start to diverge from the training results after 5 epochs. The lower dropout rate of 0.2 considerably speeds up the overfit process compared to experiment 10 with a dropout rate of 0.3. That said, the resulting test data accuracy is 78.8%, performing 2.2% worse than experiment 10.

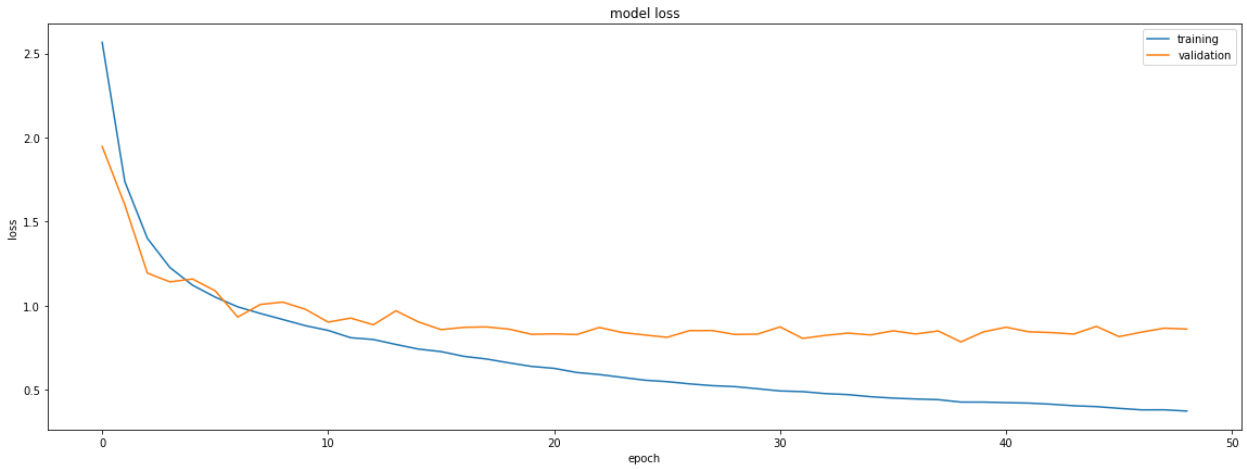
While we are able to vary the speed of the overfitting process with an L2 learning rate value over and under our initial value of 0.001 in experiment 10, doing so yields slightly worse accuracy each way. We will stay with the 0.001 L2 learning rate as we perform different configurations of dropout in experiments 14 and 15.

Experiment 15: CNN - 3 Convolutional Layers with 2 Fully Connected Layers and Regularization Combination (Dropout Hyperparameter Tweak High)

We model a CNN consisting of 3 small 3x3 convolutional layers with the first layer containing 128 filters, the second layer containing 256 filters, and the third layer containing 512 filters. The convolutional base then feeds into a DNN containing 2 fully connected layers, with the first layer containing 384 units and the second layer containing 768 units, followed by the 10-way classification output layer. Batch normalization processes follow the fully connected layers, while dropout (0.5) processes follow all convolution and fully connected layers. All fully connected layers utilize L2 regularization configured with a learning rate of 0.001 (model diagram is the same as in Experiment 10).

Training and validation accuracy and cross entropy loss are below.





We see validation results start to diverge from the training results after 5 epochs. The higher dropout rate of 0.5 also considerably speeds up the overfit process compared to experiment 10 with a dropout rate of 0.3. That said, the resulting test data accuracy is 79%, 2% worse than experiment 10.

While we've attempted to tweak L2 and dropout regularization hyperparameters in experiments 12 to 15, they didn't yield better test accuracy results compared to experiment 10. Using higher and lower

hyperparameters for L2 regularization (0.01 and 0.0001) in experiments 12 and 13 didn't improve accuracy compared to the setting used in experiment 10 (0.001), while higher and lower hyperparameters for dropout (0.2 and 0.5) in experiments 14 and 15 also didn't improve accuracy compared to the setting used in experiment 10 (0.001).

From all our experiments, experiment 10 containing 3 convolutional layers, 2 fully connected layers, and a combination of optimized regularization parameters yielded the best classification accuracy against the CIFAR-10 test dataset.

The results of our experiments are summarized in the table below, with the best model highlighted (Experiment 10).

Experiment	FC Hidden Layers	Convolutional Layers	Total Parameters	Regularization	Test Accuracy	Test Loss
1	2 (384, 768 units)	None	1,483,402	None	0.4887	1.4443
2	3 (384, 768, 1536 units)	None	2,672,266	None	0.4891	1.4526
3	1 (384 units)	2 (128, 256 filters)	3,841,930	None	0.7089	0.8573
4	1 (384 units)	3 (128, 256, 512 filters)	2,269,578	None	0.7235	0.8015
5	2 (384, 768 units)	None	1,488,010	Batch Normalization	0.4924	1.4786
6	3 (384, 768, 1536 units)	None	2,683,018	Batch Normalization	0.4937	1.5122
7	1 (384 units)	2 (128, 256 filters)	3,843,466	Batch Normalization	0.6958	0.9417
8	1 (384 units)	3 (128, 256, 512 filters)	2,271,114	Batch Normalization	0.7181	0.8391
9	1 (384 units)	3 (128, 256, 512 filters)	2,271,114	Dropout(0.3) L2Reg(0.001) Batch Norm	0.7957	0.7566
10	2 (384, 768 units)	3 (128, 256, 512 filters)	2,573,706	Dropout(0.3) L2Reg(0.001) Batch Norm	0.81	0.7363
11	3 (384, 768, 1536 units)	3 (128, 256, 512 filters)	3,768,714	Dropout(0.3) L2Reg(0.001) Batch Norm	0.7956	0.7916
12	2 (384, 768 units)	3 (128, 256, 512 filters)	2,573,706	Dropout(0.3) L2Reg(0.01) Batch Norm	0.8054	0.793
13	2 (384, 768 units)	3 (128, 256, 512 filters)	2,573,706	Dropout(0.3) L2Reg(0.0001) Batch Norm	0.8036	0.7911
14	2 (384, 768 units)	3 (128, 256, 512 filters)	2,573,706	Dropout(0.2) L2Reg(0.001) Batch Norm	0.7882	0.8425
15	2 (384, 768 units)	3 (128, 256, 512 filters)	2,573,706	Dropout(0.5) L2Reg(0.001) Batch Norm	0.7896	0.851

Conclusion

Based on our current knowledge base of deep neural networks (DNNs) and convolutional neural networks (CNNs), we were able to build a CNN model that can classify images from the CIFAR-10 test dataset with as much as 81%. We found that adding convolutional layers to a deep learning model to detect specific features of an image can dramatically improve accuracy from their DNN-based counterparts. We also found that there is a limit to the number of fully connected layers we can add to a model before we start seeing diminishing returns in accuracy. Our study also saw adding a combination of various regularization techniques to a CNN can also dramatically improve performance accuracy, and configuring their respective hyperparameters can further improve it as well. That being said, as our CNN domain and knowledge base expands over time, we can expect to improve model accuracy even more. We can also look into using pre-trained CNNs such as a VGGNet variant or other well-known CNN architectures that have been proven to yield very high accuracy for correctly classifying images in many popular image datasets, as a base framework and can be fine-tuned to classify a specific dataset. Taking this path can possibly yield shorter training times leading to improved workflow efficiency. Until then, if the tolerances for a system to integrate a computer-imaging classification system can accept 80% accuracy, the most optimal model we've built from experiment 10 is our best choice.

Resources

Krizhevsky, Alex, Ilya Sutskever, Ilya, and Geoffrey Hinton, 2017. "ImageNet Classification with Deep Convolutional Neural Networks." *Communications of the ACM* 60, no. 6 pp. 84–90. https://doi.org/10.1145/3065386.

Simonyan, Karen, and Andrew Zisserman, 2015. "Very Deep Convolutional Networks for Large-Scale Image Recognition." *International Conference on Learning Representations* 2015. https://doi.org/10.48550/arXiv.1409.1556.