Assignment 2: Computer Vision

Vivian Xia

Northwestern University

MSDS458: Artificial Intelligence & Deep Learning

Syamala Srinivasan

February 6, 2022

Abstract

Computer vision processes images or videos in order to extract features from its pixels to be used to make decisions. The objects in these images can be hard to identify if there are many different variations that the object can look like as well as if the object does not have many unique features. A multilayer perceptron model with only one hidden layer may be able to extract enough complex features to identify the class very well. Deep neural networks, on the other hand, have two or more number of hidden layers within the model. The multiple layers increase the model complexity by abstracting the data to a higher level as it passes through each layer. The increase in model complexity often results in high variance and overfitting. The use of regularization techniques can be used to prevent overfitting. This assignment will experiment with the different types of regularization techniques as well as observe their ability to prevent overfitting when training the models.

The experiments will also include the building and analyzing of two different types of deep learning models, deep neural networks and convolutional neural networks. While deep neural networks have multiple hidden layers, of which are usually Dense, convolutional neural networks have convolutional and max pooling layers that process and extract features from the images to use to discriminate classes. CNNs are designed for computer vision, whereas DNNs are more general purpose. The experiments will compare the DNN and CNN model architectures and their results in discriminating the classes.

Introduction

The construction of deep neural networks and convolutional neural networks are used to experiment on how to develop a neural network that can accurately classify images. The goal of building models to classify the CIFAR-10 dataset is to understand how dense and convolutional

network networks extract features and spatial correlations to discriminate between classes. Deep neural networks are more of a general-purpose model while convolutional neural networks are specifically for computer vision problems. By understanding the DNN and CNN architectures, the differences between the two will be understood including why CNN is more suited for image processing. These experiments will compare the two architectures and their effect on the accuracy of the classifications, which can also be visualized. With these complex models, it is important to also understand the concept and effect of regularization techniques in preventing overfitting. The regularization techniques will be useful when using these model architectures in future image processing models. These architectures and techniques will be used to comprehend how to build dense and convolutional neural networks for image processing in similar future applications. For each model, visualizations will be created to assess the performance of the model. The accuracy scores and visualizations will be used to define what a good image processing model looks like as well as what a bad one looks like.

The goals of this project will be accomplished by building both DNN and CNN models to classify images from the CIFAR-10 dataset into ten classes. The CIFAR-10 dataset contains 60,000 32x32 colored images in ten classes. The ten classes include airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 6,000 images in each class. The classes do not overlap, so the automobile class includes car types while the truck class only includes big trucks. There were no pickup trucks included in the images, so the model will not be trained to discriminate pickup trucks (Krizhevsky, 2009).

Literature Review

Computer vision extracts features from images or videos in order to discriminate between the objects within them (Nishani & Cico, 2017). For computer vision, DNN, including CNN,

which is a type of DNN, approaches are often considered. Deep learning allows models to learn data representations and extract useful information from raw data through processing. Multilayer perception models with their one hidden layer have a disadvantage due to the limited capability of processing the data in its raw form. Deep neural networks have a multiple layer architecture that allows its non-linear activation functions to pass through the original data to get higher abstraction levels of the data each time (Chauhan & Singh, 2018).

In a systematic mapping study where publications on image processing, specifically human pose estimation were classified, CNN was chosen as the architecture in 65% of the papers (Nishani & Cico, 2017). Each convolutional layer has a filter that extracts features from the input of that layer, resulting a build up to a hierarchal extraction of features. CNN consists of alternating convolutional and pooling layers. The pooling layers reduce the spatial dimension and overall computational complexity. The high dimensional feature map is then converted to one dimension to connect to the nodes of the fully connected layers which are used to predict the classes and output. Throughout other studies in computer vision, there has been much success in processing of images and videos using deep learning (Chauhan & Singh, 2018). In applications such as detecting facial expressions, one of the studies done compared the design of DNN and CNN models with the facial recognition results, and this study found that CNN had better performance than DNN (Jung et al., 2015). However, a study on lung nodule detection found that their DNN model performed better in terms of less false positives (Tan et al., 2017). Deep neural networks and their architecture can extract features that can be used to classify the images. DNN and CNN are both commonly used models for computer vision and understanding their structures can help with understanding how each model extracts features as well as in deciding which one to use for different types of problems.

Methods

In order to classify the images in this and similar future datasets, deep neural networks and convolutional neural networks, a type of deep neural network for image processing, were built to discriminate the images into ten classes. The libraries used to build the models are os, time, numpy, pandas, seaborn, packaging, sklearn, matplotlib, and tensorflow.

The library os was used to connect to Google Drive and define the path within it to save the models and refer to them later. The library time was imported to measure the processing times of each model. The numpy and pandas libraries were imported for data formatting. From the library packaging, the package version was imported to verify the tensorflow and keras version needed to build the models. Tensorflow and its package keras was imported to build, compile, and train the model. The package keras also includes the CIFAR-10 dataset that will be used in training the model. Matplotlib, seaborn, and sklearn are used to create visualizations of model performance.

The CIFAR-10 dataset is loaded in and split into a training and test set. The training set contains 50,000 32x32 colored images and its corresponding labels. The test set contains 10,000 32x32 colored images and its corresponding labels. Because the images are colored, the input shape for each image is (32,32,3) where 3 represents the RGB channels. The labels range from 0 to 9 which can be cross-referenced to its corresponding class name of airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, respectively. The data is further explored by plotting examples of the images with its corresponding class name.

After exploratory data analysis, the validation set was created from ten percent of the training set, resulting in 5,000 images and labels in the validation set and 45,000 in the training set. The validation set will be used for hyperparameter tuning. The images in the training,

validation, and test set are then normalized. Each pixel has a value from 0, white, to 255, black, so it can be normalized by dividing 255 to rescale the values to be between 0 and 1.

The networks are then built for each experiment ranging from deep neural networks to convolutional neural networks. They are all built using a Sequential class. The DNN are used as a function fitter and feature extractor. In these experiments, DNNs will be built with two or three hidden layers. The parameters of 30 epochs and batch size of 500 are used. These two parameters will be held constant throughout all the experiments to focus and compare other parameters and regularization techniques within the different models. The DNN will then extract features from the input images to discriminate the ten classes.

Deep neural networks are more of general-purpose model that are not specifically designed for any specific problem. Because convolution neural networks are specifically designed for image recognition problems, they will also be created and experimented with using different numbers of layers and regularization techniques. Each hidden layer will include a convolutional layer that captures spatial correlations. The filters used are 3x3 matrices that are learned through the optimization process. The filters scan over the image to capture different patterns to produce a feature map. The feature map is the dot product between the image and filter, which is also thought of as the correlation or covariance between the image and filter. Each filter picks up different patterns. The ReLu activation function within the layer applies nonlinearity to the feature map to prune the features that have low correlation and only keep the features with high correlations. The max pooling layer for this convolution layer summarizes the high correlations by extracting the largest correlations or most important features in the ReLu output. Depending on the experiment, this convolution and max pooling layer is used two or three times to capture all the relevant and spatial features of the image. The features that were

extracted are flattened to make them independent and then connected into a Dense layer with a ReLu activation function. The activated values are then connected to another Dense layer but an output layer with a Softmax activation function to output probabilities for each of the classes **(**Srinivasan, 2022).

Experiments 1 through 4 will not use any regularization techniques. Experiments 5 through 10 will use regularization techniques to help with overfitting and underfitting. When the model underfits and has high training and validation error, there is high bias. This problem is solved by increasing model complexity and training for longer. On the other hand, when training error is and validation error is high, indicating overfitting, there is high variance. High variance is solved by adding more training data or regularization techniques. L1 and L2 regularization is one method to reduce overfitting. This method minimizes the loss function by adding an extra regularization term for the model to zero out instead of the actual weights that need to be adjusted. This prevents the vanishing or exploding gradient problem and, in turn, reduces the complexity of the model and the variance. Dropout is another method to help with overfitting. In this hidden layer, the specified percent of the nodes, selected randomly, are dropped each iteration so that the model is not learning from those nodes anymore **(**Srinivasan, 2022). This leaves the leftover neurons to have to represent the missing neurons, preventing the neurons from settling into the context of those specific weights of the neurons. This results in more generalization and less overfitting in the model (Sanagapati, 2019). However, dropout is more straightforward in deep neural networks than convolutional neural networks since the neurons dropped can be seen. The dropped neurons in CNNs may not help in regularizing the model because it leaves the model with partial neurons that are used by filters to extract features from and pooling to summarize the important features with. Another method is data augmentation

where more training images are generated through rotating, blurring, adding pixels, etc. to the existing training data. Early stopping is used to stop training once the validation data is no longer improving after a specified number of iterations, helping the model reduce its error. Batch normalization acts like a regularization technique, despite not being one. This method normalizes the inputs of the layer, before the activation function, to output the activation values bounded between 0 and 1. Normalizing these inputs optimizes the model to train faster and prevent vanishing and exploding gradients **(**Srinivasan, 2022).

Each model and its performance will be evaluated and visualized. Using matplotlib, a performance metric plot of the training and validation loss and accuracy are visualized. A confusion matrix will also be created to see how well the model accurately predicted each class. The matrix compares the predicted labels of the test set to its actual labels. To comprehend the values more meaningfully, the values in the confusion matrix are normalized to get the error rates. The ideal model would show the confusion matrix and its cells as all black except for its diagonal, representing that the model classified all the images to its corresponding label correctly. T-distributed stochastic neighboring embedding is used to reduce the dimensionality of the features to 2 and plot these features on to a T-distribution curve to analyze how well the features from each model discriminate the ten classes. For the convolutional neural networks, the filters can be visualized to observe the features that are picked up by each filter.

Results

Experiment 1

A deep neural network was built with two hidden layers and no regularization. It was built using a Sequential class. The input was first flattened from a three-dimensional (32,32,3) shape to 3,072 one-dimensional pixels using a Flatten layer. The model also included three

Dense layers. These Dense layers connect the nodes to its preceding nodes. The first two Dense layers used ReLu as its activation function. The first Dense layer had 108 nodes, and the second layer had 200. The activated values are used as features to discriminate the classes. The output layer used the Softmax activation function with ten nodes to return values between 0 and 1 to represent the probabilities of that image belonging to each of the ten classes. The model is compiled with the Adam optimizer and Sparse Categorical Cross Entropy loss function to minimize the loss and measure the difference between the predicted output and actual output, respectively. The model is trained on the 45,000 training images and validated on the 5,000 validation images. The model is then evaluated on 10,000 images from the test set.

This model had a training accuracy of 0.53, validation accuracy of 0.48, and test accuracy of 0.49, as seen in Figure 1. There is a bit of a difference between the training and validation accuracy 0.05, indicating overfitting. The performance plot in Figure 2 shows the loss graphs for training and validation are going down in the right direction but are nowhere near 0, indicating underfitting. Similarly, the accuracy plot is also going in the right direction but is nowhere near 1. However, the score is still better than the accuracy of randomly classifying at 0.10. The model is underfitting but also soon overfitting on the unimportant features.

A confusion matrix, Figure 3, was visualized to evaluate the classification performance of the model on the test set. The matrix has a notably lighter diagonal than its surrounding cells of the matrix. However, the classification accuracy is low for the values in the diagonal, ranging from 0.26 to 0.70. The model classified the class 7, 8, and 9, which are horse, ship, and truck the best, respectively. Other than its actual class truck, the model often classified them as an automobile. It did a poor job classifying class 3 or bird, with the lowest classification rate of 0.26. Overall, the model did not do a great job classifying any of the classes.

Another visualization, Figure 4, is used to see the probabilities associated with the first twenty test images and its predicted label. For example, the model predicted that, for the first test image, there was a 7.11% probability that the image is of an airplane, 24.86% probability that it is a cat, 28.18% probability that it is a deer, etc. Because the probability was highest in the deer class, the image was predicted to be a deer. The visualization shows that the blue hues are spread all throughout this subset of probabilities, indicating that the model struggled to classify the images into one specific class.

The 200 activated values are reduced into 2 dimensions using t-SNE to be plotted into a t-distribution curve. This TSNE plot, Figure 5, shows how well the features discriminated the ten classes. There is a lot of overlap and not a lot of clear clusters in this plot. As seen in the confusion matrix, the classes horse, ship, and truck have better classification rates than the other classes. This can also be seen in this plot since the coloring of those areas as more concentrated with less overlap of other colors compared to the other classes. However, this model and its extracted features did not do a good job discriminating the classes from each other.

Experiment 2

Because of the underfitting of Experiment 1's model, Experiment 2 uses an additional hidden layer, resulting in a DNN model with 3 hidden layers. The data is first flattened then connected to the nodes of the Dense layers. Similar to Experiment 1's model, the first Dense layer has 108 nodes, and second Dense layer has 200 nodes. The new addition of the third Dense layer will have 282 nodes. The output layer is the same as Experiment 1's with 10 nodes and a Softmax activation function.

The training accuracy is 0.62, validation accuracy is 0.51, and testing accuracy is 0.52. Despite adding another layer, the test accuracy only increased by 0.01 compared to Experiment

1's model. The difference between the training and validation accuracy is big, indicating overfitting. The training on this model may have also been run on too many epochs because the validation accuracy stops improving and lies around 0.50 accuracy after around 22 epochs. The performance metric plots, Figure 6, confirm that the model is overfitting. The validation loss and accuracy graphs are both flattening as the training loss and accuracy graphs continue to go down and up, respectively. Despite overfitting, the loss plot shows that the model is nowhere closer to 0, and the accuracy plot shows that the model is nowhere close to 1. Similar to Experiment 1, this model still does better than randomly guessing. The model is overfitting on noise and features that are not useful in discriminating the ten classes.

The confusion matrix, Figure 7, looks a bit better than Experiment 1's. This model classifies the images in the test set into its actual label more accurately than Experiment 1's since there is less gray and white hues on the surrounding cells other than the diagonal and the smallest classification rate along the diagonal is 0.31 compared to 0.27 from Experiment 1. However, the visualization of the predictions and its probabilities for each class still show a similar distribution of blue hues throughout as Experiment 1's, so this model also struggled to classify the images into one specific class. The TSNE plot show a lot of overlapping colors and no clear clustering of any classes. The features extracted do not do a good job in discriminating the classes.

Experiment 3

For this experiment, a convolutional neural net with two convolution and max pooling layers will be used. The first convolutional layer has 64 filters, each 3x3. The max pooling layer uses a 2x2 matrix with a stride of 2. The second convolutional layer has 108 3x3 filters and a max pooling layer of 2x2. After flattening, the features are connected to 210 nodes in the Dense

layer and then another 10 nodes in the output Dense layer. The model summarizes 3072 input pixels into 210 features.

The training and validation accuracy of 0.94 and 0.72, respectively, have a large difference, indicating overfitting. This model may have run on too many epochs since the validation accuracy stopped improving around 17 epochs. The test set accuracy is 0.71, so the model does a better job than the DNNs in classifying but still does not do great. The performance metric plots, Figure 8, support the overfitting conclusion. The validation loss goes in the opposite direction of the training loss graph. Similarly, the validation accuracy graph flattens out while the training accuracy continues to go up. There is low training error but high validation error, so there is high variance, which can be handled through regularization.

The confusion matrix, Figure 9, shows hues of white and gray in its diagonal. The surrounding cells are black, and the largest error rate is 0.18. Otherwise, the model does a better job than Experiment 1 and 2's models in accurately predicting the labels for each class. The model performed the worst on classifying birds and automobiles correctly. The visualization on the probabilities of the predicted test images, Figure 10, has fewer blue hues distributed throughout than Experiment 1 and 2's, indicating that the model was able to discriminate the classes better than Experiment 1 or 2's model. The model was able to identify the images to its corresponding class more accurately.

The output of each filter and max pooling layer is visualized, Figure 11, to see the patterns extracted. For the first convolutional layer and its filter, the regions that are lit up seem to capture the features of the shape of the image. The first max pooling layer extracted the high correlation features pertaining to the pattern to summarize the important components from the convolutional layer output. The visual also shows the dimensional reduction from the feature

map to the max pooling output. The second convolutional layer and its filter captured a more abstract feature from the first max pooling output. This filter picked up what seems to be legs from the images to form a feature map of the correlations between the images and that pattern. The max pooling layer then takes the feature map and summarizes it by extracting only the high correlations and important features. These features are then used to predict and discriminate between the classes. A TSNE plot, Figure 12, was also constructed to visualize how the model predicts for the test set images. There are clusters, but also overlapping among the colors, indicating that this model did better than Experiment 1 and 2 but still does not do a great job with discriminating classes from one another. The model also overfits as seen from the performance metric plots.

Experiment 4

This CNN model will use three convolution and max pooling layers. Similar to Experiment 3, the number of filters for first and second convolution layer is 64 and 108, respectively. The third convolution layer will use 180 filters. The number of nodes in the hidden Dense layer is 210 nodes, and the output layer will have 10 nodes.

Similar to Experiment 3, this model is overfitting the training data. The training accuracy is 0.91 and the validation accuracy is 0.72, so there is a big difference between them. The test accuracy is 0.71, so the model does not do well in classifying the images. This model's test accuracy is the same as Experiment 3's, showing that adding complexity to Experiment 3's model through an additional convolution and max pooling layer did not solve the problem of high variance within the model. The extra layer reduced the training accuracy but did not solve the overfitting problem. The performance metric plot, Figure 13, looks the same as Experiment

3's. It shows that the validation loss and accuracy are flattening and going in the opposite direction of the training, indicating overfitting.

The confusion matrix shows that the model is not classifying any of the classes very well. In particular, the model does a bad job classifying class 2 or bird correctly. The model does an okay job at classifying classes 0, 8, and 9 or airplane, ship, and truck, respectively, but still not great. The visualization on the probability outputs show that the model is better at classifying an image to one specific class compared to Experiment 1 and 2, but still shows some distribution of blues among the rows similar to Experiment 3.

Visualizing the feature map and max pooling outputs, the patterns are captured and summarized by each pair of layers. Similar to Experiment 3, the first filter seems to have picked up the outline of the object in the image. The max pooling then summarized that feature map to only the most important features. The second filter may have picked up if the image has legs or not. This feature map is again summarized by max pooling. The third filter extracts very abstract features that cannot be easily identified by the human eye. Nevertheless, those aspects with a correlation to the filter are extracted into the feature map and summarized. The TSNE plot shows clusters, so the features extracted are able to discriminate between the classes. There is still overlap, so similar to Experiment 3's model, this model still does not do a great job classifying.

Experiment 5

This experiment uses the same model architecture as Experiment 1 but also uses the regularization techniques of batch normalization, dropout, and early stopping. The dropout was set to 0.3, so a random 0.3 of the nodes will be dropped each iteration. The early stopping will stop the training if the validation accuracy does not improve after 3 epochs.

The training and validation accuracy are close in value to one another, indicating that there is no overfitting. The validation and testing accuracy scores are also larger than they were in Experiment 1. The early stopping stopped the training after 20 epochs, which helped to reduce the processing time despite the more parameters this model had than Experiment 1's. The performance metric plots, Figure 14, show the loss and accuracy graphs are going in the right direction, although they are not close 0 and 1, respectively. The training and validation graphs similar to each other in value so there is no overfitting.

Despite the confusion matrix having a diagonal that is lighter than the rest of the matrix, the correct classification rates for each class are still low. However, the range of the rates are better and narrower than Experiment 1's. Experiment 1's correctly classified rates ranged from 0.26 to 0.70 while this confusion matrix's ranges from 0.33 to 0.70, which is reflected in the better testing accuracy of this model. The prediction plot of the probabilities looks similar to Experiment 1's. There is still a lot of blue hues distributed throughout each row and the plot, so the model does not do a very good job in discriminating the classes from one another. The TSNE plot also looks like Experiment 1's, so there is still overlapping of colors and no clear clusters. This model architecture with regularization did better than the model architecture without regularization. The test accuracy was better, and the model did not overfit when used with regularization. Despite the increase in accuracy, the model still does not do a great job in discriminating the ten classes.

Experiment 6

This experiment will use the same model architecture as Experiment 2 but will use regularization techniques, batch normalization, dropout, L2 regularization, and early stopping, to help with overfitting. The dropout parameter is set to 0.3, so that 30% of the nodes are dropped

each iteration. Early stop will stop training once the validation accuracy stops improving for 3 epochs.

The validation and test accuracy were the same as Experiment 2's. However, the training accuracy was lower than that of Experiment 2's. The accuracy scores of all three sets were very similar to one another in this model, indicating that there is no overfitting for this model. Regularization helped with the overfitting problem seen in Experiment 2's model. There was a very big difference between the training and validation accuracy scores in Experiment 2. In contrast, the performance metric plot, Figure 15, for this model shows little to no gap between the two values. The loss and accuracy plots are going in the right direction, but they are nowhere near 0 and 1, respectively. The model is underfitting so adding some model complexity may help.

The confusion matrix, Figure 16, shows some gray hues other than in the diagonal. This model seems to have lower classification rate accuracy than Experiment 2's. Experiment 2's matrix had fewer gray hues outside the diagonal. This experiment and Experiment 2 had around the same correct classification rates in their diagonals. One noticeable error rate from this model's matrix is that the model misclassifies trucks as automobiles 29% of the time. The error rate for the misclassification of cats as dogs and vice versa are around the same as Experiment 2's. For the prediction probabilities visual, there is a lot of blue hues distributed throughout the plot, showing that these features do not do a good job discriminating between classes. The TSNE plot also supports that the model does not a good job because there is still overlapping colors with no clear clusters.

Experiment 7

This model is built using the same architecture as Experiment 3's model with regularization techniques. L2 regularization, dropout, and early stopping are used to help the model not overfit. The hidden Dense layer will have the L2 regularization method. The early stopping is set with a patience of 3, so if the validation accuracy does not improve after three epochs, the training will stop.

The training and validation accuracy have a small difference of 0.04, a much smaller difference than that of Experiment 3's 0.22. The testing accuracy did better than the corresponding score in Experiment 3. The performance metric plots, Figure 17, show that both loss and accuracy plots are going in the right direction, but are still not very close to 0 and 1, respectively. There is no overfitting evident on both plots. The loss plot shows the validation loss graph going in the same direction as the training loss graph. Similarly, the validation accuracy also is similar to the training accuracy. The model is not overfitting. Overall, the model does not do a good job in discriminating between the classes with a test accuracy of 0.73 but is better than all the experimented models so far.

The confusion matrix, Figure 18, looks similar to Experiment 3's in that there are no lighter hues other than the diagonal and the classification rates in the diagonal are also about the same. This model does do a better job classifying classes 1, 2, 4, 5, and 7 or automobile, bird, deer, dog, and horse correctly than the model from Experiment 3. However, the classification rate for ship and truck or class 8 and 9 is worse than Experiment 3. The prediction probabilities visual shows a similar amount of blue hues distributed in each row and plot as Experiment 3.

The visualization of the feature maps and output of the max pooling layers shows the features that are extracted from the images. Each filter used extracts a feature from the image and the feature map shows the correlations between the filter and images. Each of the convolution

layers is followed by a max pooling layer to summarize the highest correlations. As mentioned above, the use of dropout in CNNs are a bit more ambiguous in how the filter and max pooling layers use the partial neurons to train on, but this model shows that the dropout did help in regularizing the model from overfitting. The TSNE plot has less overlap than the other experiments. There is some clear clustering for some of the classes such as horse and frog, which supports the observations made in the confusion matrix. This model is better at discriminating the ten classes than the other models, but it still does not do a very good job.

Experiment 8

This experiment uses the same architecture as Experiment 4, but with L2 regularization, dropout, and early stopping regularization techniques. Like Experiment 7, the early stopping is set with a patience of 3 and only the hidden Dense layer will have the L2 regularization.

The test accuracy is higher than any of the other models that were experimented so far with a score of 0.76. The score is not very high but is still good. The training and validation accuracy are very similar. Like the conclusions about the regularization methods in Experiment 7, these methods including dropout do seem to have helped with overfitting since Experiment 4's model had a 0.19 training and validation score difference and this model only has a 0.02 training and validation score difference. The performance metric plots, Figure 19, show that the training and validation graphs are very close in value to each other in both loss and accuracy. The accuracy and loss plots are both moving in the right direction but are not very close to 1 and 0, respectively. The validation graphs actually have better scores than the training graphs, so this model could have possibly been improved with more complexity and training epochs. There might have not been a need for the early stopping regularization in this model.

The confusion matrix, Figure 20, shows that there are no large error rates from misclassifying the images since there are no gray hues in the matrix except in the diagonal. The diagonal's values show that each class is classified correctly from 61 to 89% of the time. The model does a better job overall in classifying the images correctly compared to Experiment 4 as well as Experiment 7 which had one less convolution, max pooling, and dropout layer. This model misclassifies class 5, dog, as 3, cat, and vice versa. The prediction probabilities visualization does still have some blue hues among the rows, showing that this model is still not great at discriminating between classes. Similar to Experiment 4, the features that have been extracted from each filter can be seen where with each next filter, the features extracted are more and more abstract. The TSNE plot, Figure 21, shows clusters. There is some overlap especially between dog and cat, brown and red. Overall, the clusters are clearer than the other models' TSNE plots, so the features extracted from this model were able to discriminate among the classes.

Experiment 9

This experiment used the same number of layers as Experiment 8 including the same regularization techniques. However, the number of filters for the convolutional layers and nodes for the hidden Dense layer are doubled from Experiment 8 to grow the model complexity. The first convolutional layer has 128 filters, the second layer has 216, and the third has 360. The number of nodes in the fully connected layer is 420. In addition, the fixed number of epochs used in the other experiments have also been increased from 30 to 50. The early stopping callback is still implemented in this experiment.

The training process for this model took a significantly longer time than Experiment 8's, which was expected as the number of parameters that need to be trained are significantly greater

than Experiment 8's. It also took 37 epochs before the callback stopped the model from training. The training accuracy score was 0.84 and the validation accuracy score was 0.80. The difference in accuracy is small, so the regularizations did a good job in preventing overfitting in this much more complex model. The test accuracy is 0.80, which is better than the other models so far. The performance metric plots, Figure 22, show that the loss and accuracy are going in the right direction and pretty good in terms of their respective values in loss and accuracy. There is no overfitting since the training and validation graphs are similar to each other.

The confusion matrix, Figure 23, shows that the diagonal is the only part of the plot that has gray and white hues. The surrounding cells are black. The classification rate for classes 2, 3, and 5 or bird, cat, and dog have the lowest classification rates at around 0.65, respectively. The class cat is often misclassified as class dog at the misclassification rate of 0.11. The class dog is misclassified as class cat at a rate of 0.16. Other than those classes, the classification rates along the diagonal are high ranging from 0.80 to 0.91, indicating that the model does a good job discriminating between those classes. This conclusion is also supported by the prediction probabilities visualization, Figure 24. There are fewer blue hues and gradient across each row with most rows having one solid blue color with a high probability that the image belongs into that specific class. The TSNE plot, Figure 25, shows clustering. There is still some overlapping, but the clusters are distinguishable. There does not seem to be as much overlapping in this plot as Experiment 8's plot. The model does a good job discriminating between the classes. Despite this model having a better test accuracy than the other models, it did take significantly longer for the model to train as well as more computational power in terms of RAM and GPU.

Experiment 10

This model was built using four convolutional layers and two max pooling and dropout layers. Instead of one convolutional layer followed by a max pooling and dropout layer, this model has two convolutional layers followed by a max pooling and dropout layer. This structure is repeated once more before the feature map is flattened and undergoes batch normalization. It then is connected to a fully connected Dense layer with L2 regularization. The number of filters and nodes resembles the same number as Experiment 8's except the fourth convolutional layer will have 210 filters instead of the fully connected layer. The fully connected layer will have 260 nodes. A visualization of this architecture is shown in Figure 26. The number of parameters in this model is greater than Experiment 9's. The training of this model goes through only 13 epochs due to early stopping, so the processing time ends up being shorter than that of Experiment 9's.

The training and validation accuracy have a 0.10 difference, which is big and indicates that there is overfitting. There may need to be more regularization in this model. The test accuracy is 0.79, which is very similar to the test accuracy of 0.80 in Experiment 9. This test accuracy is also better than that of the other models excluding Experiment 9. The performance metric plots, Figure 27, shows that there is some overfitting within the model. The validation loss and accuracy graphs flatten out after five or six epochs. However, the training loss and accuracy continue to go towards 0 and 1, respectively. The test accuracy score of 0.79 is also good.

The confusion matrix, Figure 28, shows a very good classification rates along the diagonal. This model does misclassify class 4, deer, the most. Other than that class, the classification rates range from 0.70 to 0.92, which show that the features do a good job in discriminating between those classes. One larger error rate of 0.16 that can be seen from the plot is the misclassification of class 5, dog, as class 3, cat. The prediction probabilities visualization

also supports that this model does a good job with discriminating between classes. There is not many blue hues distributed along each row rather just one darker blue block. This shows that the features can discriminate the classes more definitively. The TSNE plot, Figure 29, also looks good. There are distinct clusters especially with automobile, truck, frog, horse, and ship classes. There is less overlap that in the other models. Overall, this model and its features are able to discriminate the ten classes well.

Summary

Based on the test accuracy score, the best model would be Experiment 9's with a score of 0.80. This model had 3 convolutional and max pooling layers with regularization and 1.5 million parameters. This model took the most processing time and ran for 37 epochs. The training took a lot of computational power.

The best model for implementation would be Experiment 8's model. The model does not overfit and has a good test accuracy of 0.76. This test accuracy is better than all the other models except Experiment 9 and 10. Experiment 10's model, however, overfit the training data and also had close 2 million parameters to train. Experiment 8's model only ran 400,000 parameters each epoch. This model from Experiment 8 can also be improved by letting it run for more epochs because the early stopping regularization did not stop it from training to the set parameter of 30 epochs, so the model could have probably been trained for more epochs and improved its accuracy scores without overfitting.

Conclusion

From these experiments, the DNN and CNN models do better at discriminating the classes than randomly classifying. However, the CNN architecture is much better at extracting important features from the input images than DNN. The DNN architectures from Experiment 1

and 2 had a test accuracy of around 0.50 while the CNN architectures from Experiment 3 and 4 had a score of 0.71. The convolutional layers in the CNN architecture use filters to extract the high correlation features from the images for each filter. The filters are able to capture different features from the images to then be processed in the max pooling layer to be summarized. The effectiveness of these two layers in discriminating between classes are evident by the test accuracies in the CNN models. One tradeoff of using CNN rather than DNN is processing time. DNN models had a much faster processing time than CNN models as well as using less computation power.

The regularization techniques helped in multiple aspects of training the data such as stopping training early when validation accuracy stops improving, minimizing the loss function, and normalizing inputs. The use of regularization helped with preventing high variance and overfitting in the models especially as they became more complex. The models from Experiment 1 through 4 had no regularization techniques implemented and suffered from overfitting as seen by difference in the training and validation accuracies and performance metric plots. With the use of regularization, those experiments' counterparts, Experiment 5 through 8, did not overfit as well as increased the resulting test accuracy compared to that of Experiment 1 through 4. These techniques allowed Experiments 9 and 10 to be attempted as well because the regularization helped with balancing out the model complexity.

References

Chauhan, N. K., & Singh, K. (2018). A Review on Conventional Machine Learning vs Deep Learning. *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*, 347–352. https://doi.org/10.1109/gucon.2018.8675097

Jung, H., Lee, S., Park, S., Kim, B., Kim, J., Lee, I., & Ahn, C. (2015). Development of deep learning-based facial expression recognition system. *2015 21st Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV)*, 1–4. https://doi.org/10.1109/fcv.2015.7103729

Krizhevsky, A. (2009). *The CIFAR-10 dataset*. Retrieved 2022, from https://www.cs.toronto.edu/~kriz/cifar.html

Nishani, E., & Cico, B. (2017). Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation. *2017 6th Mediterranean Conference on Embedded Computing (MECO)*, 1–4. https://doi.org/10.1109/meco.2017.7977207

Sanagapati, P. (2019, July 21). *What is dropout regularization? find out :)*. Kaggle. Retrieved 2022, from https://www.kaggle.com/pavansanagapati/what-is-dropout-regularization-find-out

Srinivasan, S. (2022). *MSDS 458 AI/Deep Learning: Sync Session #3* [Zoom Cloud Recordings].

Tan, J., Huo, Y., Liang, Z., & Li, L. (2017). A Comparison Study on the Effect of False Positive Reduction in Deep Learning Based Detection for Juxtapleural Lung Nodules: CNN vs DNN. *Modeling and Simulation in Medicine Symposium (MSM 2017)*. https://doi.org/10.22360/springsim.2017.msm.012

Appendix

Figure 1. Table of experiments and their accuracy scores and processing time.

Experiment	Description	Training	Validation	Testing	Process Time (seconds)
1	DNN with 2 layers (no regularization)	0.56	0.50	0.51	28.21
2	DNN with 3 layers (no regularization)	0.62	0.51	0.52	19.16
3	CNN with 2 convolution/max pooling layers (no regularization)	0.94	0.72	0.71	77.74
4	CNN with 3 convolution/max pooling layers (no regularization)	0.91	0.72	0.71	76.63
5	DNN with 2 layers with regularization (batch normalization, dropout, early stopping)	0.54	0.52	0.53	15.92
6	DNN with 3 layers with regularization (batch normalization, L2 regulariazation, dropout, early stopping)	0.51	0.51	0.52	14.77
7	CNN with 2 convolution/max pooling layers with regularization (L2 regularization, dropout, early stopping)	0.78	0.74	0.73	36.71
8	CNN with 3 convolution/max pooling layers with regularization (L2 regularization, dropout, early stopping)	0.75	0.77	0.76	80.61
9	CNN with 3 convolution/max pooling layers with regularization (L2 regularization, dropout, early stopping) and increased complexity using a larger number of filters, nodes, and epochs than Experiment 8.	0.84	0.80	0.80	121.36
10	CNN with 4 convolution and 2 max pooling layers with regularization (batch normalization, L2 regulariazation, dropout, early stopping)	0.90	0.80	0.79	55.42

Figure 2. Experiment 1 performance metric plot.

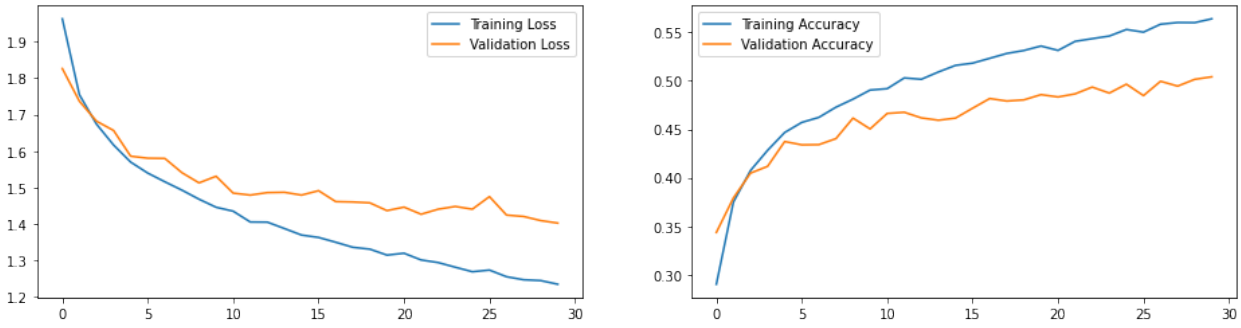

Figure 3. Experiment 1 confusion matrix.

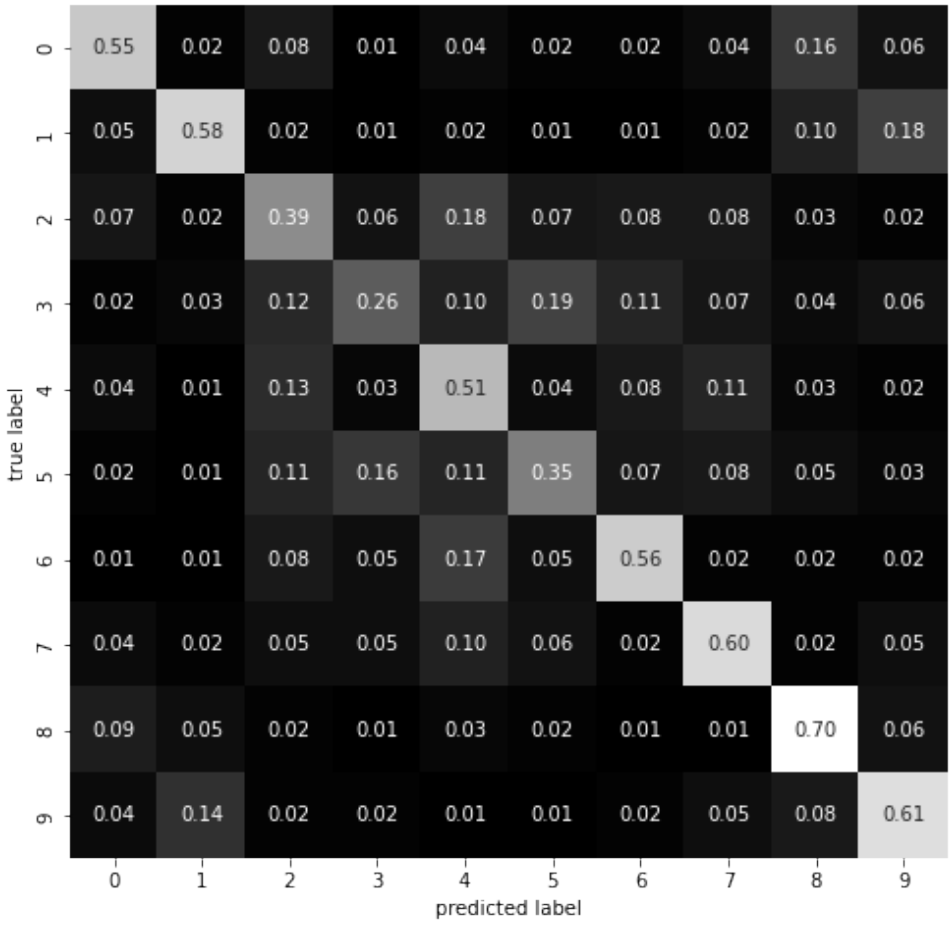

Figure 4. Experiment 1 class probabilities for test set images.

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	7.11%	15.35%	11.38%	24.86%	28.18%	5.53%	1.32%	0.11%	4.60%	1.56%
1	5.53%	21.65%	0.15%	0.01%	0.03%	0.01%	0.00%	0.03%	34.50%	38.08%
2	46.61%	6.41%	0.64%	0.04%	0.08%	0.03%	0.00%	0.17%	36.88%	9.14%
3	14.23%	5.78%	5.62%	2.19%	7.41%	1.48%	0.03%	12.40%	43.87%	6.99%
4	0.07%	0.10%	3.84%	1.05%	73.90%	2.35%	18.00%	0.27%	0.38%	0.04%
5	1.15%	0.43%	2.93%	20.54%	5.30%	22.99%	34.79%	10.44%	0.29%	1.14%
6	3.99%	14.68%	12.07%	40.38%	0.03%	24.16%	0.89%	1.17%	0.15%	2.47%
7	0.48%	0.27%	13.93%	9.95%	10.59%	7.68%	54.72%	1.18%	0.59%	0.62%
8	2.85%	0.30%	20.31%	14.27%	36.65%	15.57%	1.02%	8.07%	0.90%	0.06%
9	1.45%	69.60%	1.69%	0.74%	0.14%	0.11%	0.28%	0.14%	4.87%	20.96%
10	42.23%	0.34%	5.59%	3.12%	2.82%	4.41%	0.95%	0.27%	40.21%	0.05%
11	0.05%	22.26%	0.01%	0.09%	0.01%	0.05%	0.05%	0.03%	2.90%	74.56%
12	0.63%	1.98%	18.02%	18.72%	12.00%	25.33%	13.32%	6.84%	2.64%	0.51%
13	4.02%	0.08%	0.86%	0.12%	0.28%	0.99%	0.03%	93.53%	0.04%	0.05%
14	3.51%	50.06%	11.29%	1.80%	0.13%	1.96%	0.78%	1.66%	0.23%	28.57%
15	4.08%	0.50%	2.83%	2.51%	13.27%	11.21%	1.49%	0.30%	63.56%	0.24%
16	0.71%	0.72%	0.90%	9.03%	0.15%	9.86%	0.39%	72.34%	0.19%	5.71%
17	17.96%	2.29%	9.84%	7.20%	16.67%	2.83%	3.83%	12.21%	4.83%	22.35%
18	0.37%	0.11%	0.03%	0.01%	0.07%	0.00%	0.00%	0.01%	99.32%	0.09%
19	0.04%	0.08%	3.16%	1.93%	1.75%	1.78%	77.22%	12.91%	0.04%	1.08%

Figure 5. Experiment 1 TSNE plot.

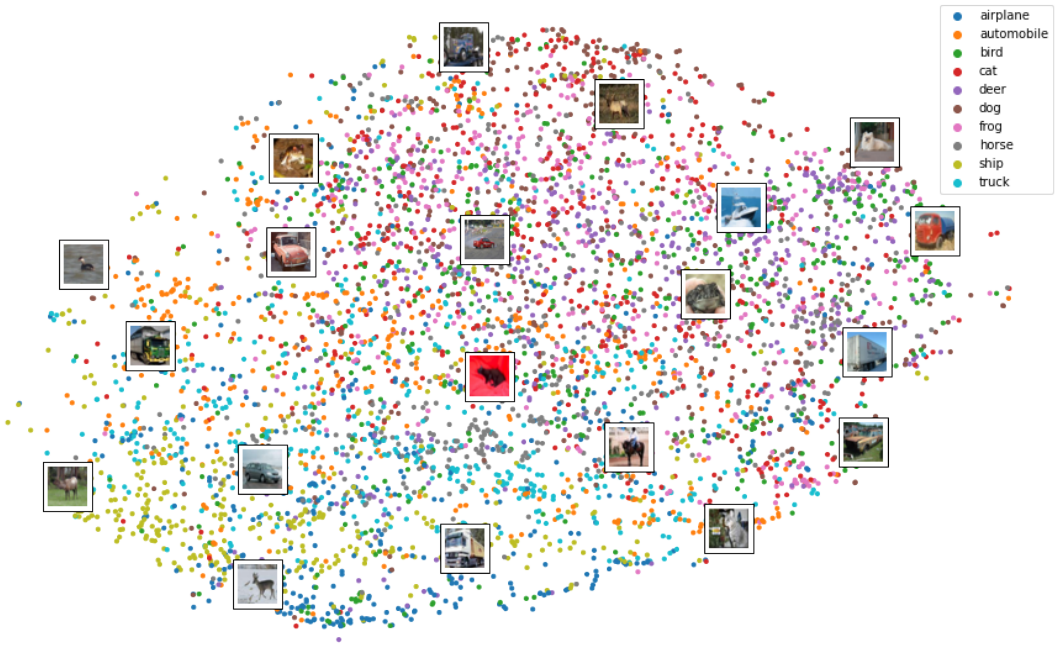

Figure 6. Experiment 2 performance metric plot.

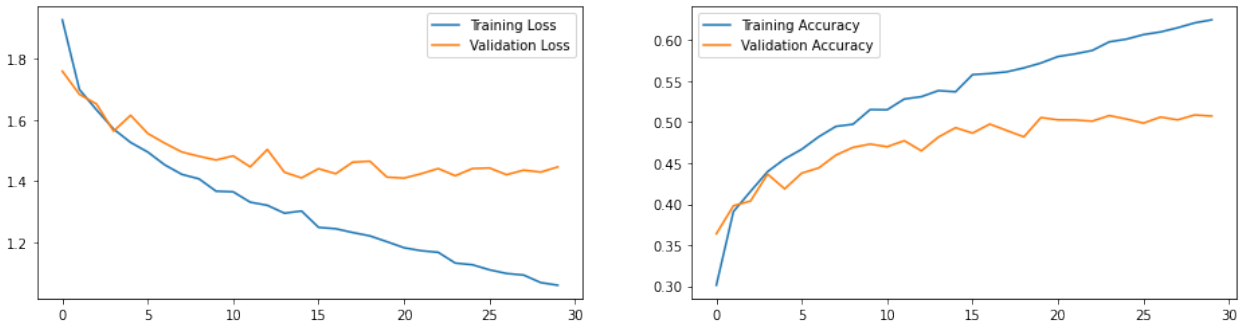

Figure 7. Experiment 2 confusion matrix.

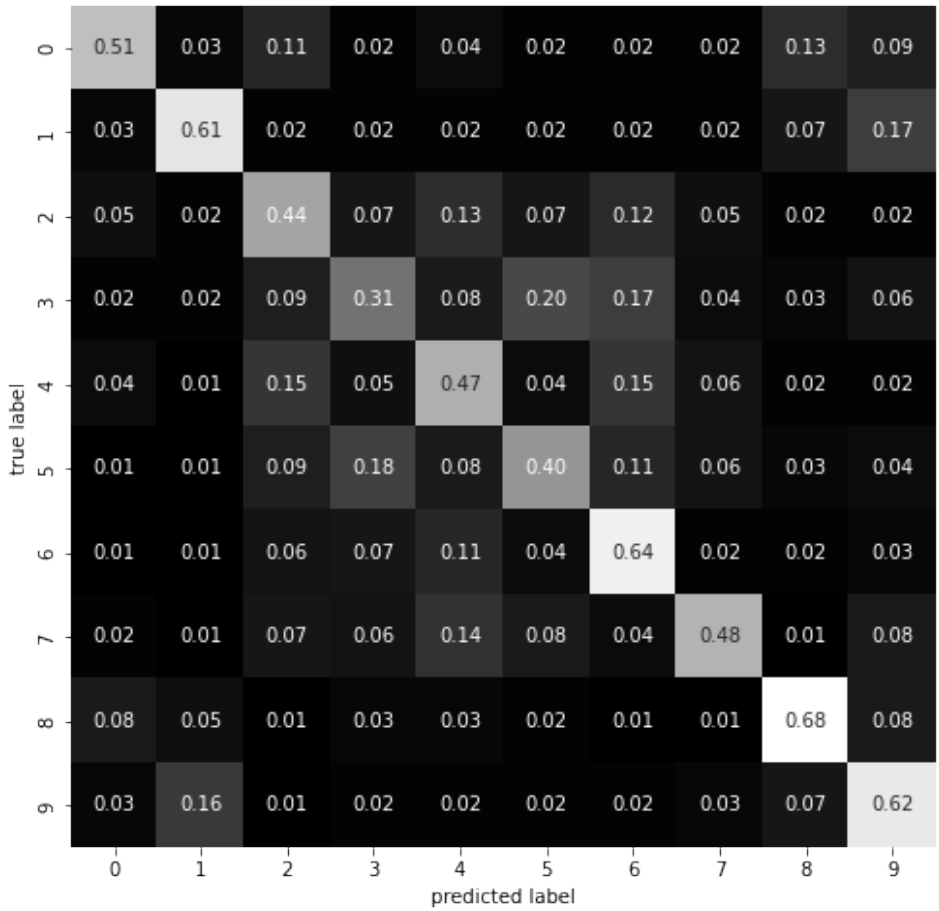

Figure 8. Experiment 3 performance metric plot.

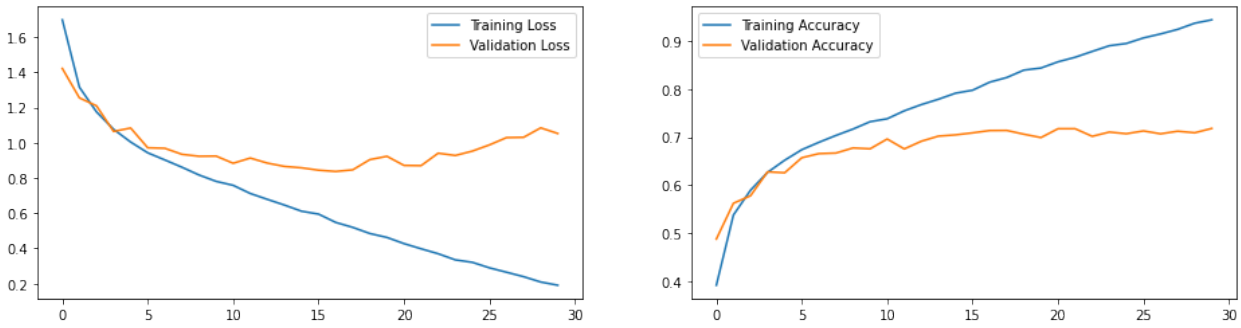

Figure 9. Experiment 3 confusion matrix.

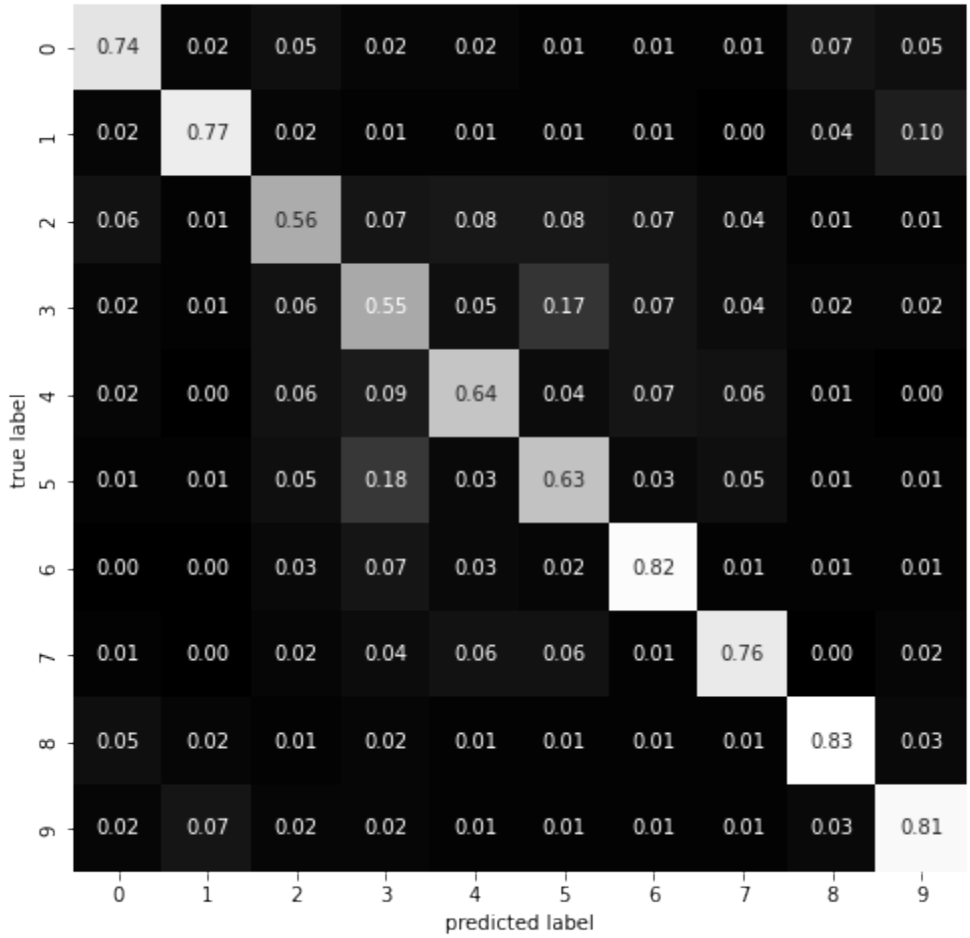

Figure 10. Experiment 3 class probabilities for test set images.

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	0.01%	0.02%	0.00%	88.46%	0.00%	10.75%	0.72%	0.00%	0.02%	0.01%
1	0.04%	5.98%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	93.99%	0.00%
2	4.20%	18.31%	0.00%	1.27%	0.04%	0.00%	0.00%	0.03%	75.39%	0.75%
3	80.11%	0.01%	0.02%	0.13%	0.26%	0.00%	0.00%	0.00%	19.45%	0.01%
4	0.00%	0.00%	1.93%	5.79%	40.64%	0.31%	51.33%	0.00%	0.00%	0.00%
5	0.00%	0.00%	0.09%	0.36%	0.00%	8.25%	91.26%	0.04%	0.00%	0.00%
6	0.00%	0.35%	0.00%	0.26%	0.00%	0.01%	0.00%	0.00%	0.00%	99.38%
7	0.27%	0.00%	1.31%	1.55%	3.36%	1.07%	92.26%	0.06%	0.12%	0.01%
8	0.00%	0.00%	0.04%	84.87%	0.11%	14.11%	0.08%	0.79%	0.00%	0.00%
9	0.00%	99.96%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.04%
10	15.93%	0.04%	0.84%	3.91%	42.98%	35.66%	0.31%	0.18%	0.16%	0.00%
11	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
12	0.00%	0.00%	0.57%	0.20%	0.00%	99.23%	0.00%	0.00%	0.00%	0.00%
13	0.00%	0.00%	0.00%	0.00%	0.00%	0.02%	0.00%	99.98%	0.00%	0.00%
14	0.00%	0.02%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.98%
15	0.00%	0.00%	0.00%	0.01%	0.00%	0.00%	43.47%	0.00%	56.52%	0.00%
16	0.01%	0.02%	0.00%	0.21%	0.00%	96.82%	0.00%	2.94%	0.00%	0.01%
17	0.14%	0.00%	1.04%	14.25%	8.29%	0.43%	0.28%	75.00%	0.18%	0.38%
18	0.01%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.93%	0.03%
19	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.99%	0.00%	0.00%	0.00%

Figure 11. Experiment 3 plot feature map of filters and max pooling layers.

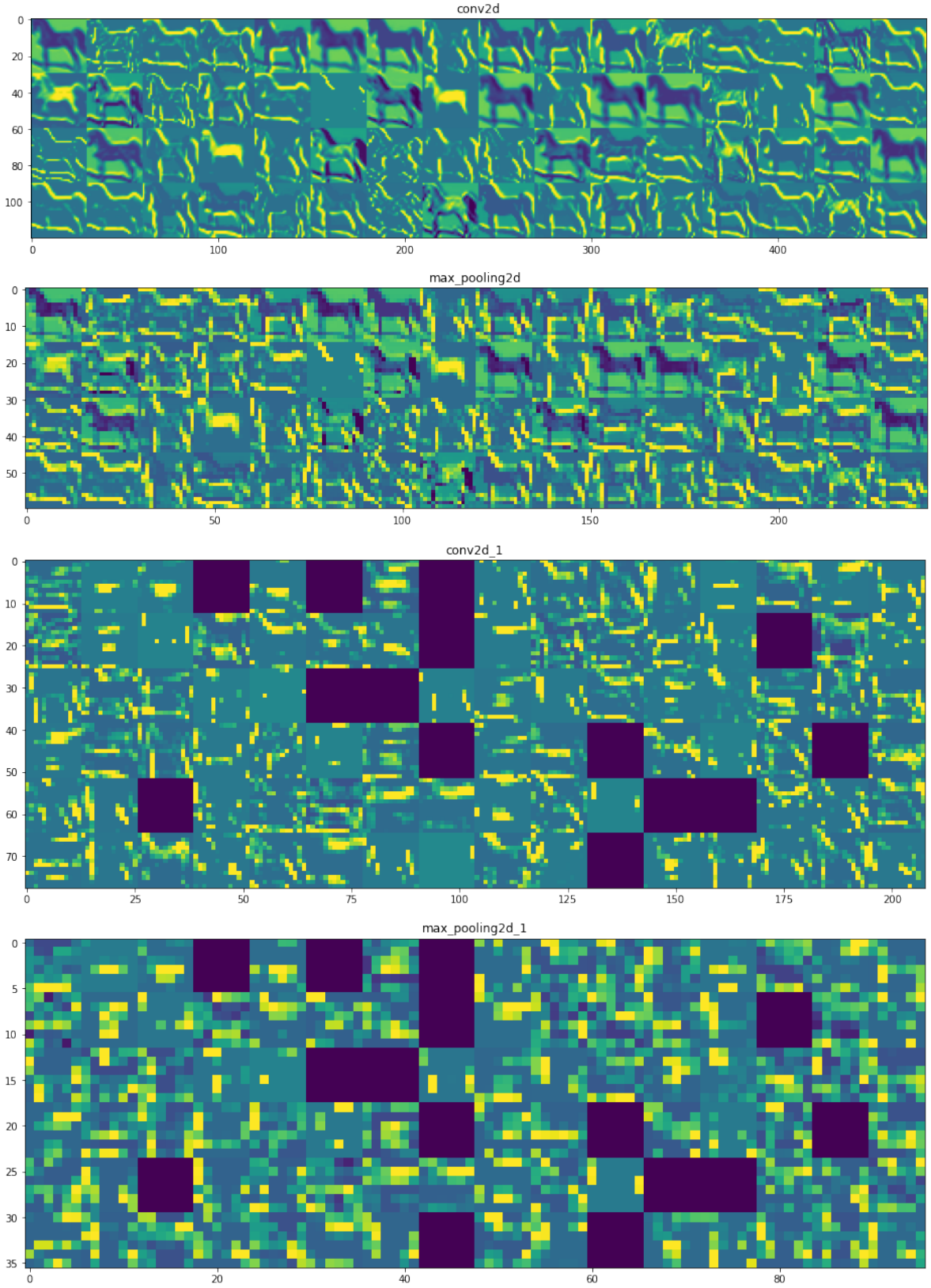

Figure 12. Experiment 3 TSNE plot.

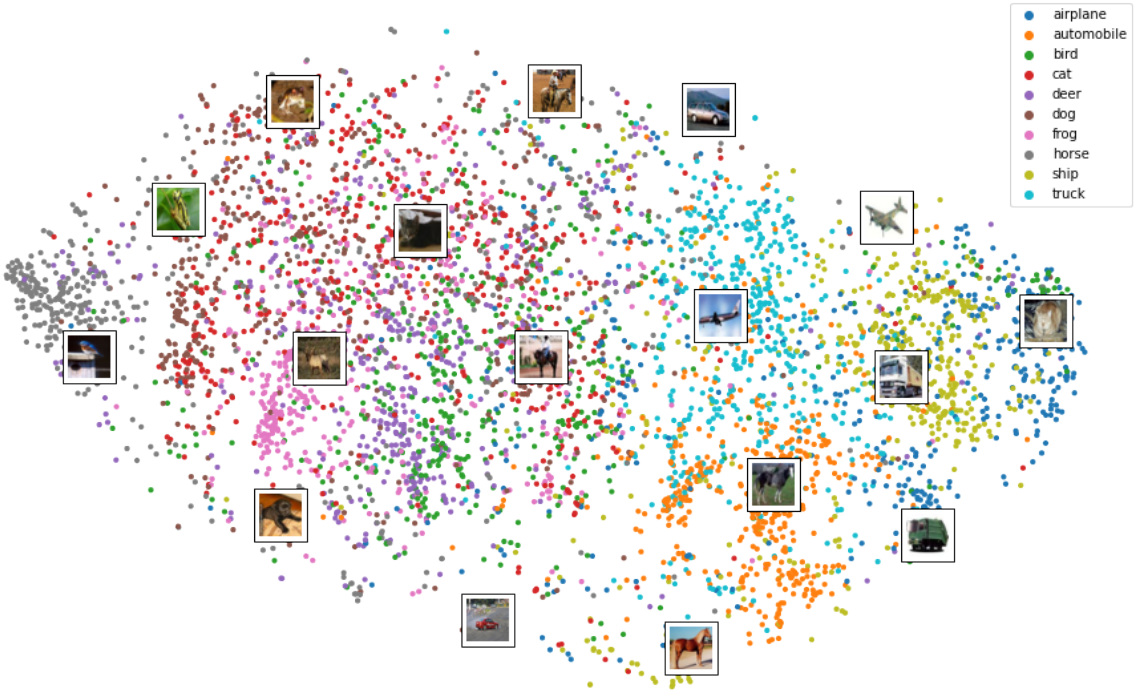

Figure 13. Experiment 4 performance metric plot.

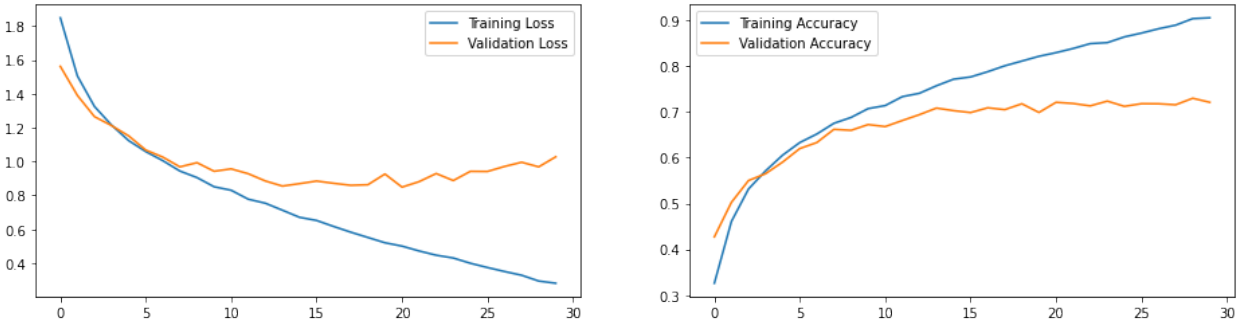

Figure 14. Experiment 5 performance metric plot.

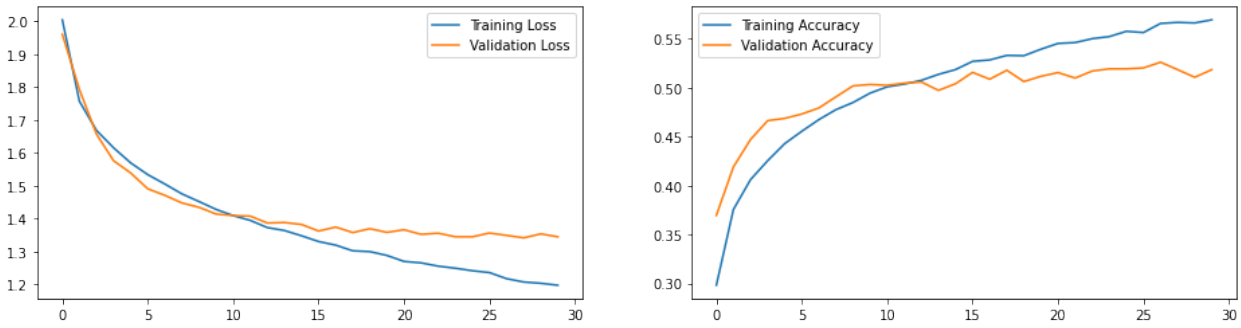

Figure 15. Experiment 6 performance metric plot.

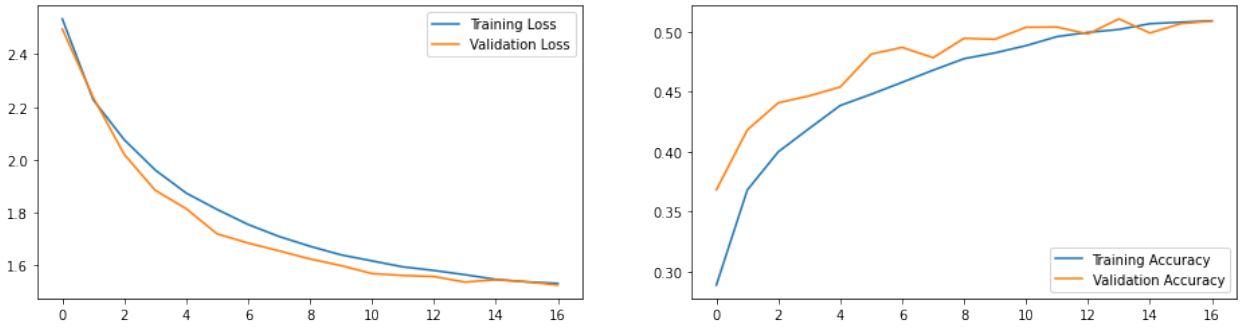

Figure 16. Experiment 6 confusion matrix.

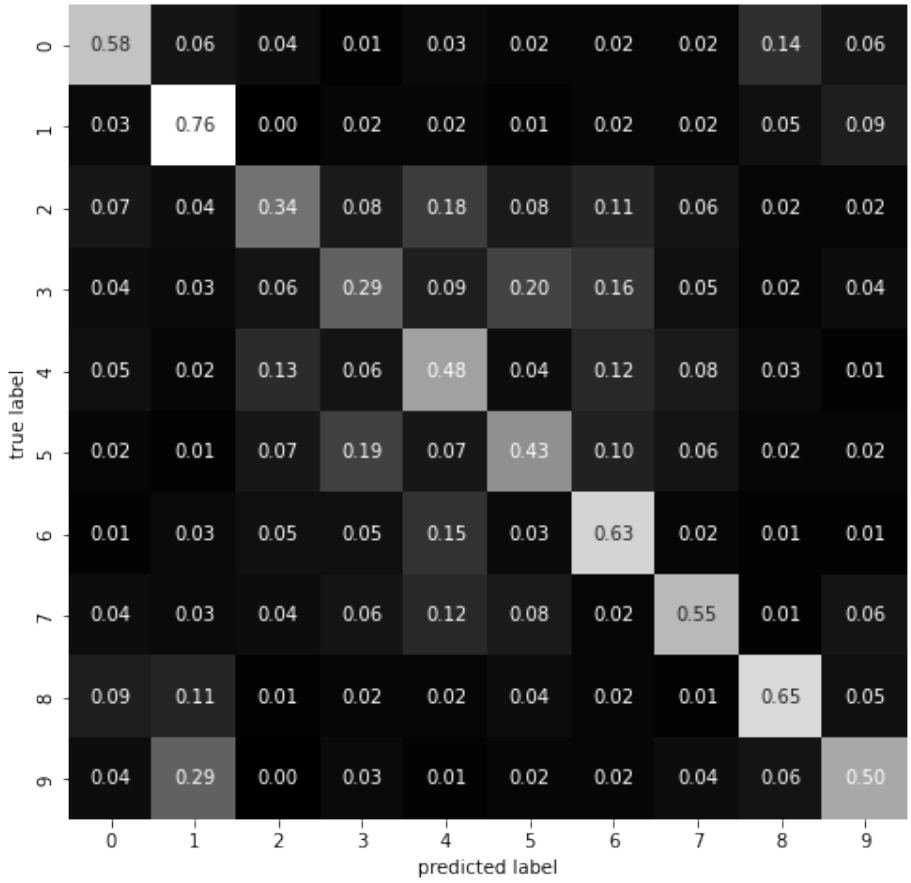

Figure 17. Experiment 7 performance metric plot.

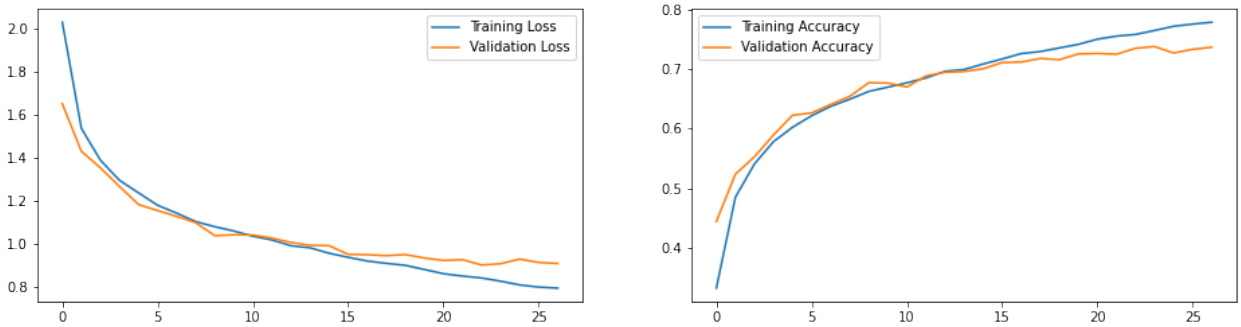

Figure 18. Experiment 7 confusion matrix.

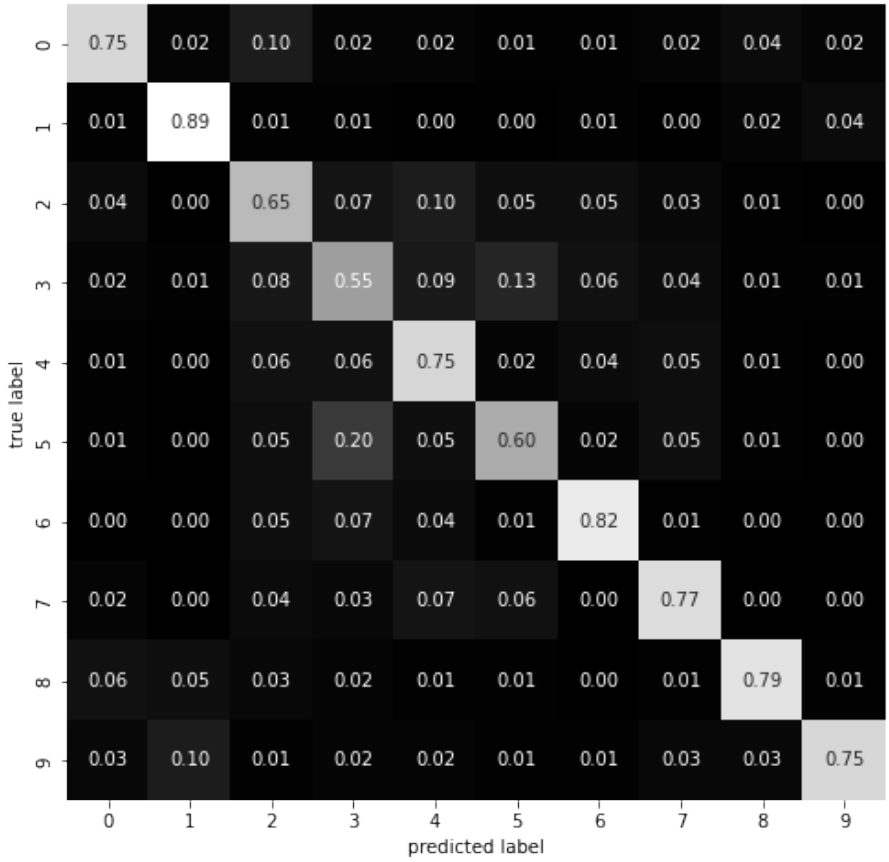

Figure 19. Experiment 8 performance metric plot.

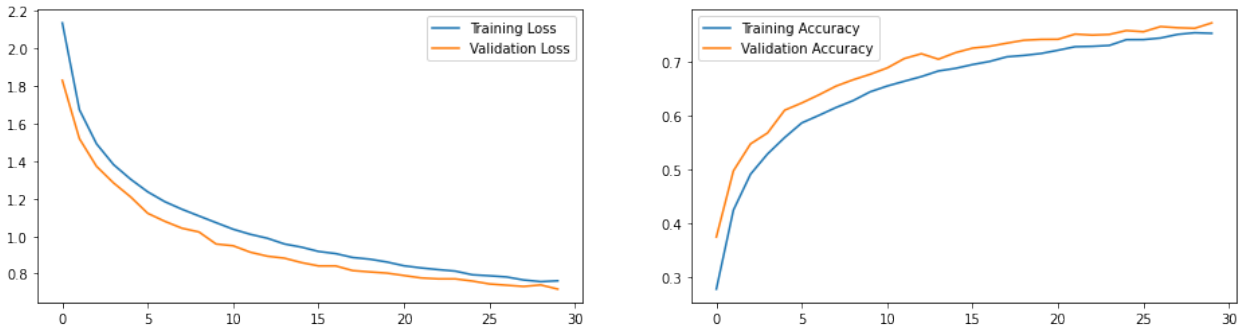

Figure 20. Experiment 8 confusion matrix.

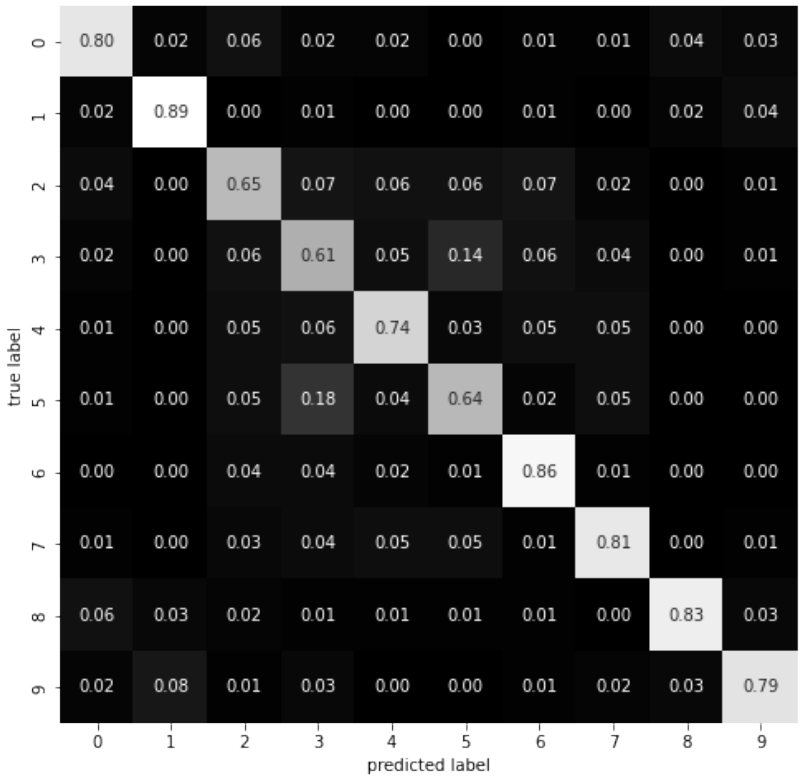

Figure 21. Experiment 8 TSNE plot.

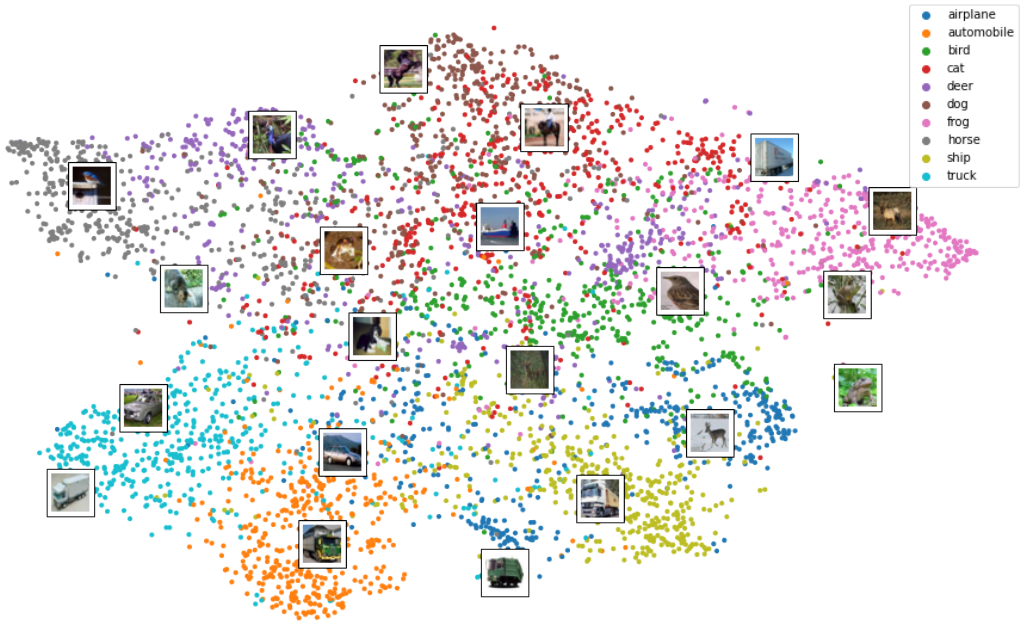

Figure 22. Experiment 9 performance metric plot.

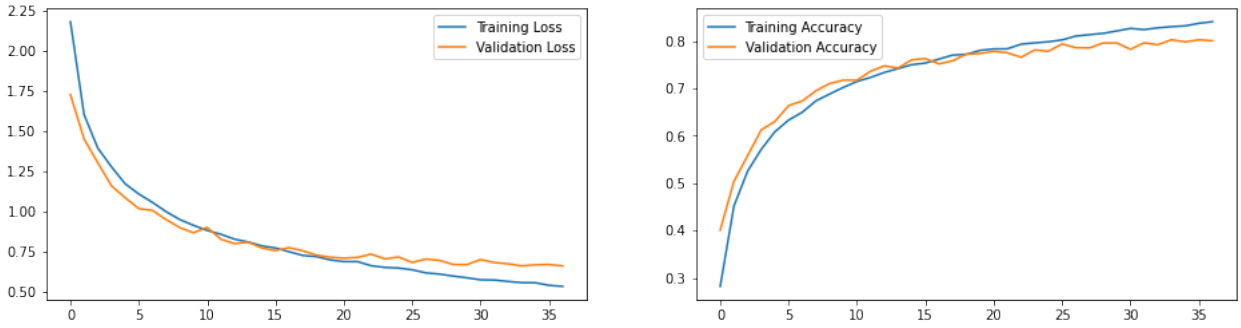

Figure 23. Experiment 9 confusion matrix.

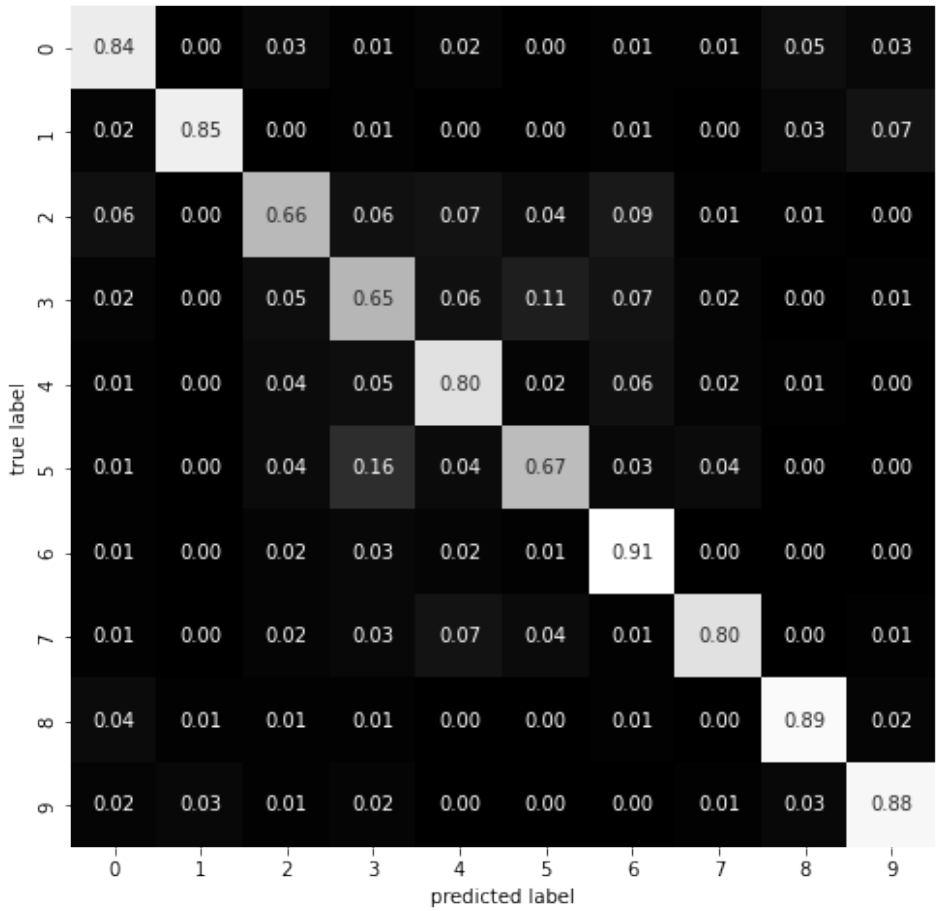

Figure 24. Experiment 9 class probabilities for test set images.

	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	1.72%	0.02%	0.23%	93.11%	0.06%	4.17%	0.09%	0.06%	0.52%	0.02%
1	0.30%	0.34%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.35%	0.01%
2	2.25%	4.50%	0.03%	0.58%	0.04%	0.04%	0.10%	0.03%	90.44%	1.99%
3	95.06%	0.41%	0.35%	1.00%	0.39%	0.00%	0.01%	0.01%	2.72%	0.06%
4	0.00%	0.00%	0.46%	0.14%	23.18%	0.00%	76.21%	0.00%	0.00%	0.00%
5	0.00%	0.00%	0.09%	0.53%	0.07%	0.19%	99.13%	0.00%	0.00%	0.00%
6	0.13%	81.42%	0.07%	1.94%	0.00%	1.55%	0.25%	0.52%	0.07%	14.05%
7	0.09%	0.01%	5.44%	1.62%	1.02%	0.24%	91.45%	0.01%	0.03%	0.07%
8	0.00%	0.00%	0.36%	94.15%	1.06%	2.56%	1.53%	0.33%	0.00%	0.00%
9	0.27%	48.61%	0.02%	0.04%	0.01%	0.00%	0.44%	0.00%	0.35%	50.26%
10	16.81%	0.02%	4.63%	11.12%	3.28%	7.67%	0.14%	5.91%	49.02%	1.40%
11	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	100.00%
12	0.08%	0.00%	6.54%	31.66%	13.92%	42.35%	1.14%	4.26%	0.04%	0.00%
13	0.00%	0.00%	0.00%	0.00%	0.05%	0.03%	0.00%	99.91%	0.00%	0.00%
14	0.00%	0.03%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.01%	99.95%
15	6.06%	0.25%	13.92%	7.41%	37.03%	0.07%	20.29%	0.03%	14.86%	0.09%
16	0.00%	0.01%	0.14%	16.43%	0.02%	82.17%	0.16%	1.06%	0.01%	0.01%
17	0.34%	0.05%	0.90%	3.82%	0.80%	7.38%	0.51%	80.67%	0.10%	5.44%
18	0.59%	0.04%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	98.70%	0.66%
19	0.00%	0.00%	0.02%	0.01%	0.00%	0.00%	99.96%	0.00%	0.00%	0.00%

Figure 25. Experiment 9 TSNE plot

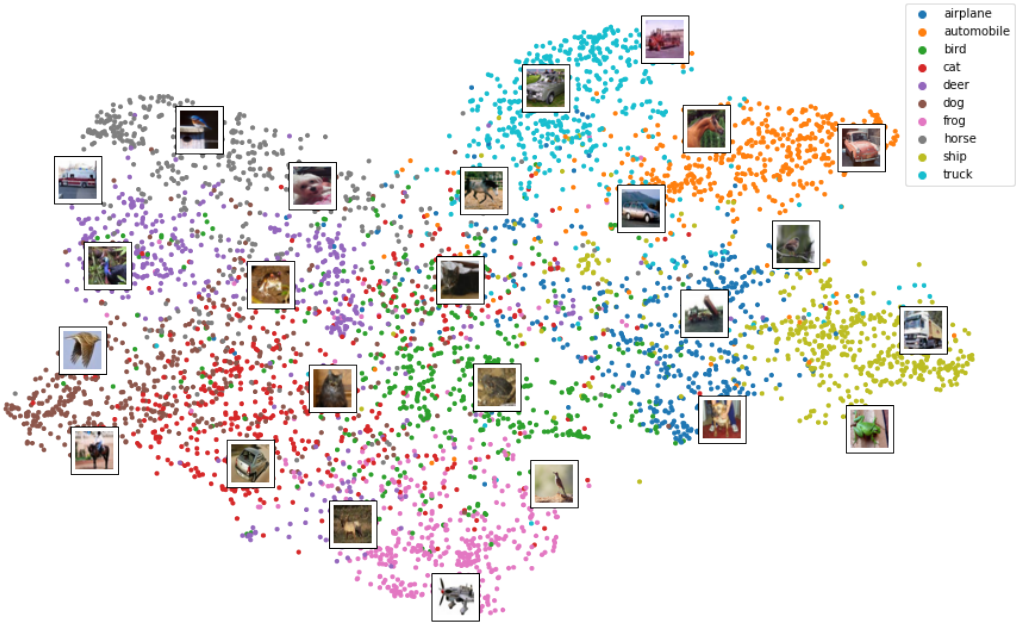

Figure 26. Experiment 10 CNN model architecture.

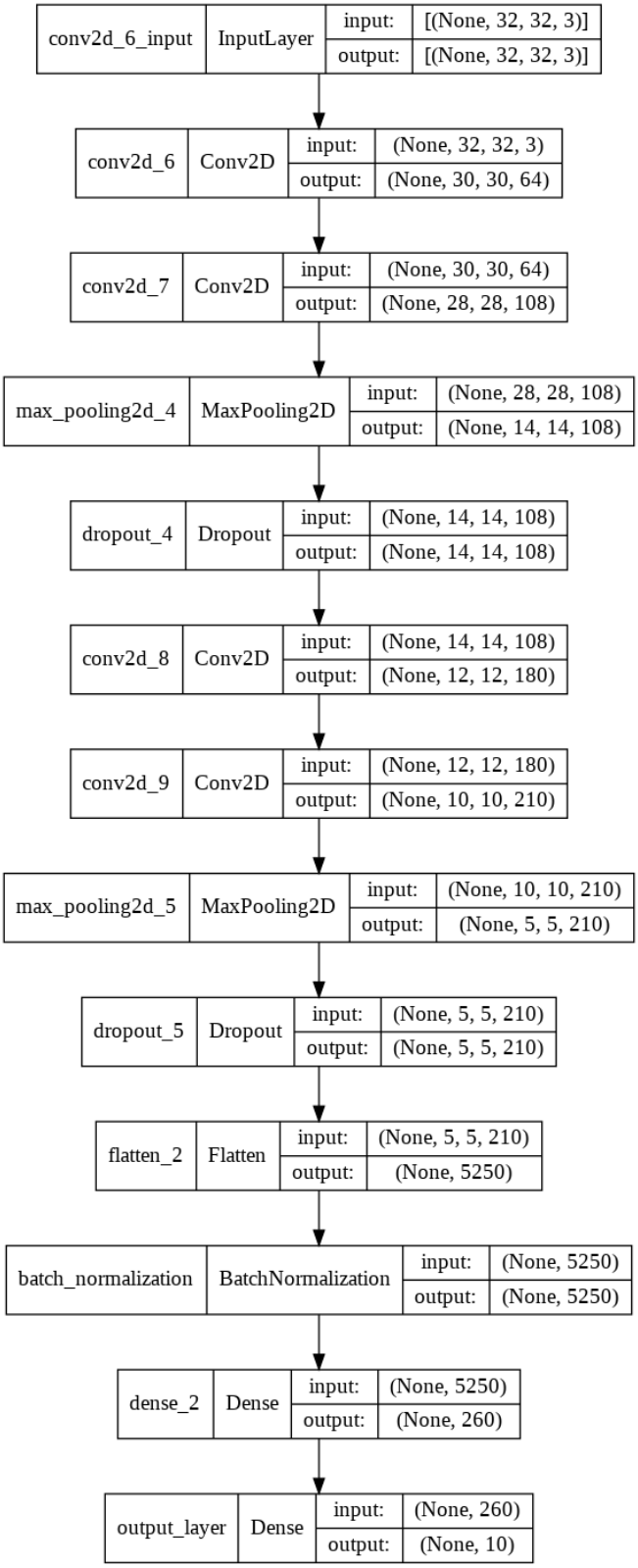

Figure 27. Experiment 10 performance metric plot.

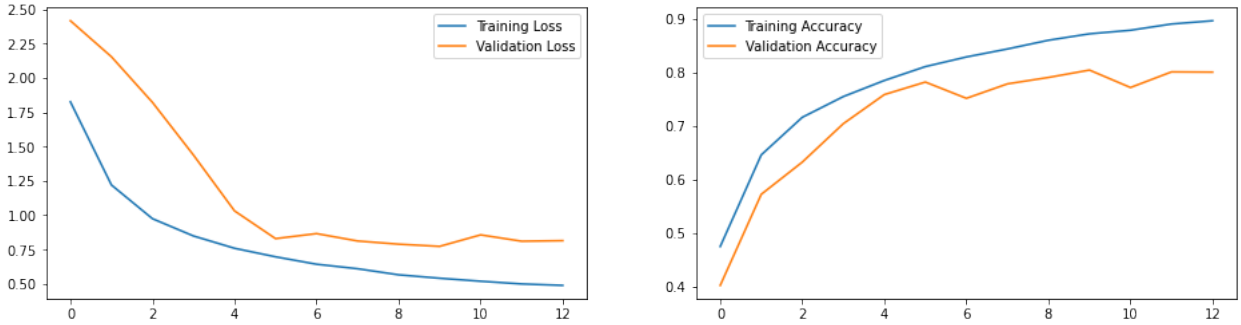

Figure 28. Experiment 10 confusion matrix.

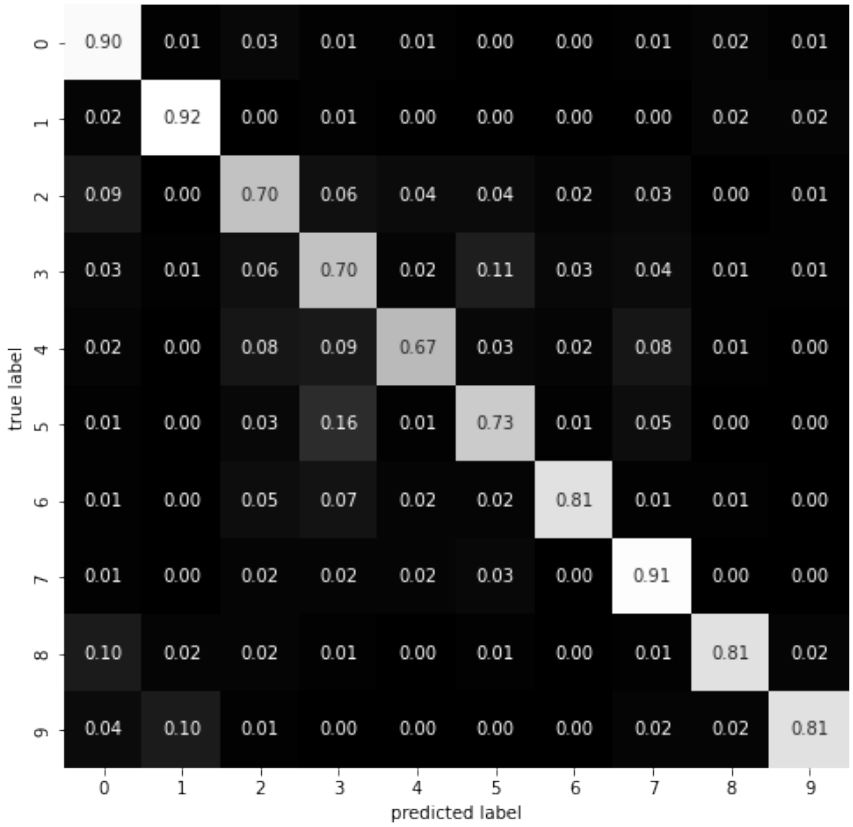

Figure 29. Experiment 10 TSNE plot.