A.2 Second Research/Programming Assignment

By: Robin Singh

Abstract:

In this experiment, I will be progressing on to working with Convolutional Neural Networks and analyzing how they work and differ from the Dense Neural Networks that I worked with before. Unlike before, where we worked with a single hidden layer and a different number of nodes in a DNN, we will discuss the impact of using a CNN for more complex image classification and having multiple hidden layers to help separate the features, both Conv2D layers and Max Pooling layers. The Conv2D layers and Max Pooling layers allow for the neural network to parse through the image via a section of the image and analyze what prominent features exist in that section and what makes that image be classified as so. We work with the CIFAR-10 dataset, a dataset consisting of images of 10 types of things: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, or truck. We compare the performance of a DNN and a CNN when classifying the images. IN the end, we will be able to answer the management questions of what type of neural network should be used for facial recognition on a mobile device such as a smartphone, how would the model be trained, whether deep learning should be necessary, whether there is an advantage of adding Convolutional layers to the neural network, and how we would measure the model's accuracy with real world examples.

Introduction:

This research is conducted to analyze various parameters and features of a dense neural network and a convolutional neural network and determine what they mean and how they help make up a successful neural network for predicting objects in the CIFAR-10 dataset. We need to determine how to preprocess the data, split the data into train, validation, and test sets, and build a neural network capable to accepting the data in the processed format and successfully classify them into one of 10 categories in the end. The accuracy will be judged by how well it is predicting the set of pixel values into objects after being trained by the training set, validated by the validation set, and finally tested on the test set that had not yet been seen by the model, which would be used to determine the essential features of a facial recognition system that would require a neural network to compare one's face to the face of the device's user to unlock the device. This way, in a real-world scenario, had this occurred, we would now what it takes for a neural network to successfully analyze the precise details and features of one's face by determining what kind of neural network be used, what sort of layers the neural network should have, how many hidden layers the neural network should have, how the results should be categorized, and what we need to be weary of when feeding the data in (such as similar numbers, human error, etc.).

Literature Review:

Many people have worked on classifying the CIFAR-10 dataset, as it is one of the first examples of more complex image classifications students of the data science field interact with, especially when getting to know about CNNs. One such example is by Aarya Barhmane, who posted his walkthrough on creating a neural network on Towards Data Science. His neural network was created with a Conv2D input layer consisting of 32 filters and kernel size of 3 as well, followed by a max pooling layer of size 2 and strides 2 as well. He is creating one of the more simple CNNs to help those get started with creating CNNs by learning more about them. A link to his experiment will be below, under sources.

Methods:

For the first neural network, I began with a DNN consisting of one layer with 384 nodes and analyzed the performance. For the DNN, I had to add a Flatten Layer because it only accepts single dimensional arrays. As advised, this was performed without regularization. I wanted to see how much of the variance could be analyzed by a simple dense neural network and what level accuracy such a simple model could bring.

I created another dense neural network with two layers consisting of 384 nodes and 192 nodes each. This was also created with no regularization. Both previous neural networks were set to run through 200 epochs with a batch size of 512 but had early callback methods in case the accuracy did not improve over the course of three epochs.

Afterwards, I created a 3-layer DNN with 384, 192, and 96 nodes respectively (no regularizers).

Afterwards, I moved on to creating neural networks with convolutional and max pooling layers. The first CNN consisted of two Conv2D and Max pooling layers each, without regularizers, with the first Conv2D layer consisting of 128 nodes and the second one consisting of 256 nodes. The kernel_size was 3 by 3 pixels with a stride of 1 and each Conv2D layer was followed by a Max pooling layer with a 2 by 2 shape and stride of 2.

I followed this up with another CNN with an additional 512 node layer and the same setup as before.

I then went back to creating another set of the 2- and 3-layer DNNs, but this time with the addition of regularizers with parameter 0.001, but the same setup otherwise.

This was then followed by another 3-layer DNN but an alternate 0.05 parameter regularizer.

I also redid the 2- and 3-layer CNN setups with 0.001 regularizers and another 3-layer CNN with a 0.05 regularizer.

Some personal experimental neural networks I had also created to improve accuracy was another CNN with the 0.001 regularizer but with more nodes in each layer. This means starting with 256 nodes in the first Conv2D layer and moving to a 512-node layer, and finally a 640-node layer. I also increased the classification dense layer to have 480 nodes as opposed to the previous 384. Next, I kept the same number of nodes within each layer as the original 3-layer CNN but reduced the kernel size of the Conv2D layers to a 2 by 2 size and the Max Pooling layer to a 1 by 1 size and strides of 1.

Finally, I created another CNN with 256, 512, and 1024 nodes layers with the same 384 node classification dense layer node setup.

For the final three Neural Networks, I also increased the callback patience from 3 to 10 in case of some learning that occurs down the road.

Results:

Here is a chart regarding the accuracy and performance evaluations of the various neural network setups:

NN	**Training Set Accuracy**	**Validation Set Accuracy**	**Test Set Accuracy**
Dense layer without Regularization (1 Layer)	0.4929	0.4770	0.47360000014305115
Dense layer without regularization (2 layers)	0.5664	0.5230	0.5102999806404114
Dense layer without regularization (3 layers)	0.5672	0.5147	0.5152999758720398

CNN without regularization (2 layers)	0.8018	0.7537	0.7386999726295471
CNN without regularization (3 layers)	0.8405	0.808	0.79339998960495
DNN with regularization (2 layers)	0.5008	0. 4857	0.4812000095844269
DNN with regularization (3 layers)	0.4878	0.4767	0.4857000112533569
DNN with alternate regularization (3 layers)	0.2579	0.2617	0.24619999527931213
CNN with regularization (2 layers)	0.8154	0.7573	0.7515000104904175
CNN with Regularization (3 layers)	0.8350	0.8073	0.7903000116348267
CNN with alternative Regularizer (3 layers)	0.7542	0.7570	0.7405999898910522
CNN with More nodes in each layer (256, 512, 640 in Conv2Ds and 480 in Dense)	0.8544	0.8037	0.8036999702453613
CNN with smaller kernel sizes	0.8824	0. 6850	0.678600013256073
CNN with More nodes in each layer (256, 512, 1024 in Conv2Ds and 384 in Dense)	0.9422	0.8180	0.8008000254631042

Conclusion:

From the results above we can see that the dense neural network can only do so much in terms of account for the variance and analyze the features in an image. A dense neural network was well suited for the MNIST dataset we worked with in the previous assignment, however, with a similar setup as before, the first DNN only achieves a 47.4% accuracy, while another with two layers achieves a 51% accuracy. AN addition of the third layer offered no improvement to the performance of the neural network, meaning the DNN has a limited ability in this scenario and is not the optimal method when it comes to more complex image classification. The regularizer in this case seems to have almost no effect, as it is usually used to avoid overfitting. However, in this case, more fitting and learning of the dataset is necessary, so it was not of much help. Moving on to the CNNs, we can see a significant improvement in the accuracy starting at 73% with 2 Conv2D and Max Pooling layers. The Conv2D layer allows the neural network to parse through the image and learn its features more effectively, such as lines, shapes, and shades of color in the images. The max pooling layer then helps to identify the most prominent feature from the Conv2D layer and highlights that feature for classification. The Dense layer in the CNN then helps classify the images with their analyzed features into the designated categories. The more layers added, the more features are extracted and more in-depth the neural network goes into extracting those features.

The regularization for the CNNs improve its performance, as the regularizer increased the accuracy by approximately 2% in the 2-layer CNN but was not as effective in the 3-layer setup, as it stayed the same. A bigger regularizer value in the CNN of 0.05 as opposed to 0.001 seemed to have hindered its performance, as this caused the model to be underfit, as seen in the alternate regularizer accuracy score of 74% compared to the previous 79% in the 3-layer CNN setups.

Next, I had experimented with a smaller Kernel size of 2 by 2 as a stride of 2 and a max-pooling size of 1 by 1 with a stride of 1. This drastically reduced the accuracy of the currently best performing 3-layer CNN model to a 67%. This may be a result of trying to learn too much about an image and overfitting, later finding fewer similarities between the classes and separating them.

Adding more nodes to each of the Conv2D layers, I did see a very slight, but consistent improvement, such as the CNN with 256, 512, 640 nodes in the Conv2D layers and 480 nodes in the Dense layer, as well as the CNN with 256, 512, 1024 nodes in the Conv2D layers and 356 nodes in the Dense layer. Both held an 80% accuracy, which may have resulted from better separation or some features from each other and more effective classification in the end because of it.

In the end, my best model was the one with three Conv2D layers with 256, 512, and 640 nodes in those layers with max pooling sizes 2 by 2 and strides of 2. In this model, the train and validation accuracy also stayed similar to each other for longer, which indicated well-fitting and almost no overfitting. I had also reduced this batch size to 64, which may have had little contribution to the performance, however when run with the original batch size of 512, results were not much different.

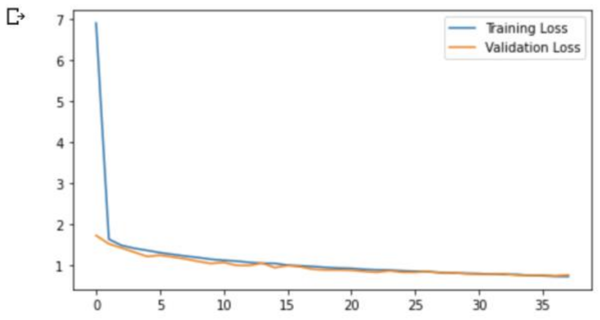
Sources:

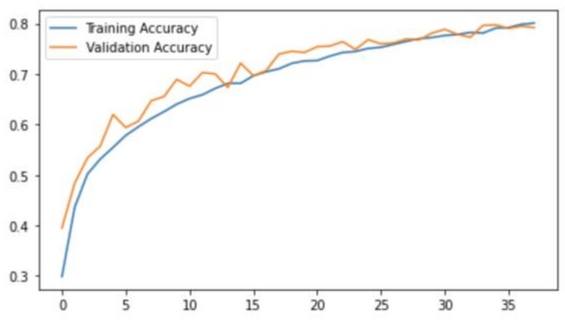
Brahmane, Aarya. “Deep Learning with CIFAR-10 Image Classification.” Medium. Towards Data

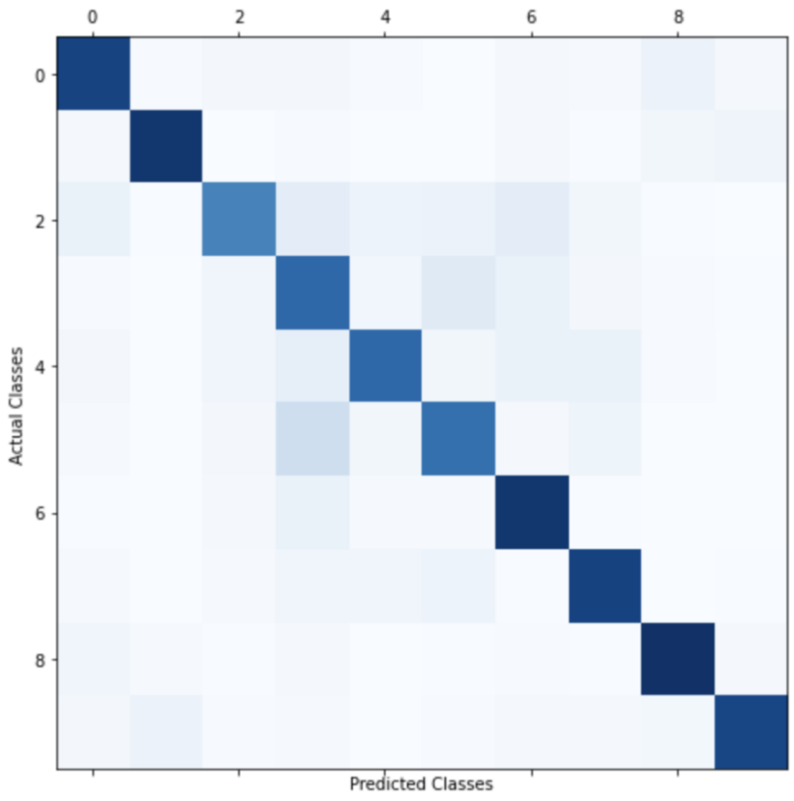
Science, October 31, 2020. https://towardsdatascience.com/deep-learning-with-cifar-10-image-classification-64ab92110d79.

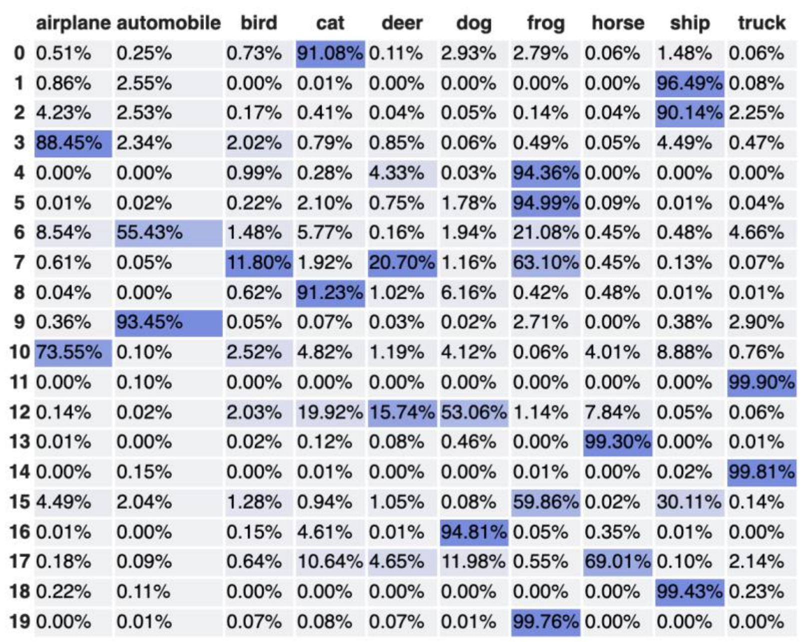
Appendix:

From CNN with more nodes in each layer (256, 512, 640 in Conv2Ds and 480 in Dense):





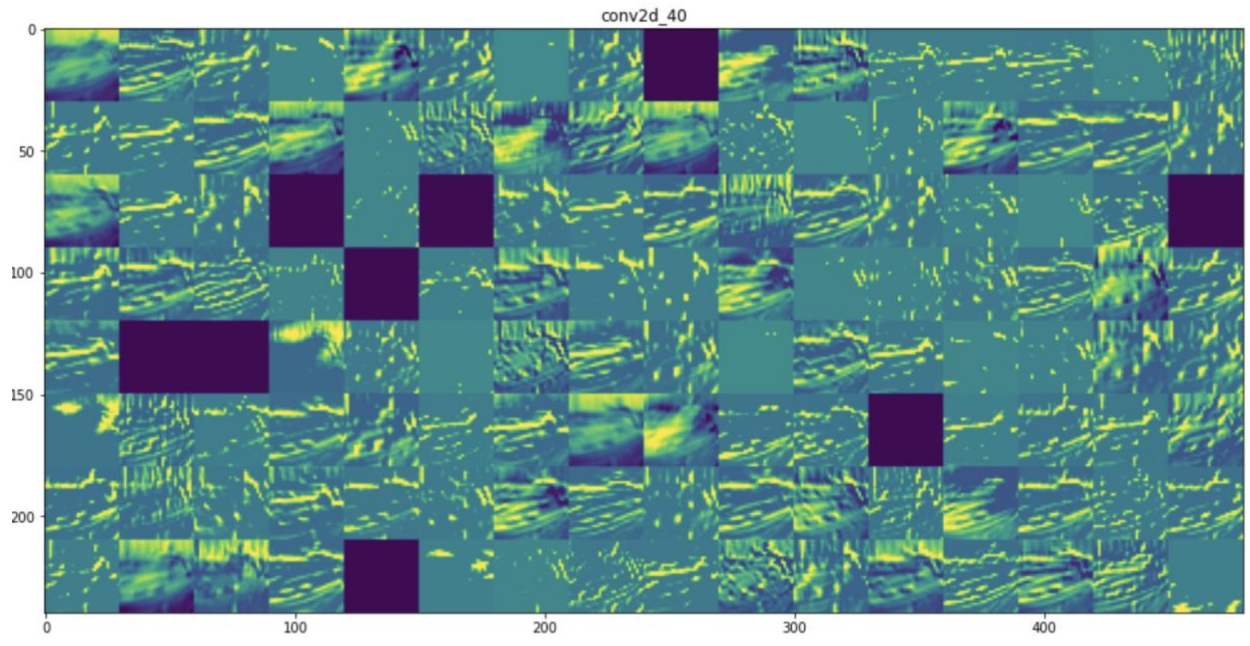


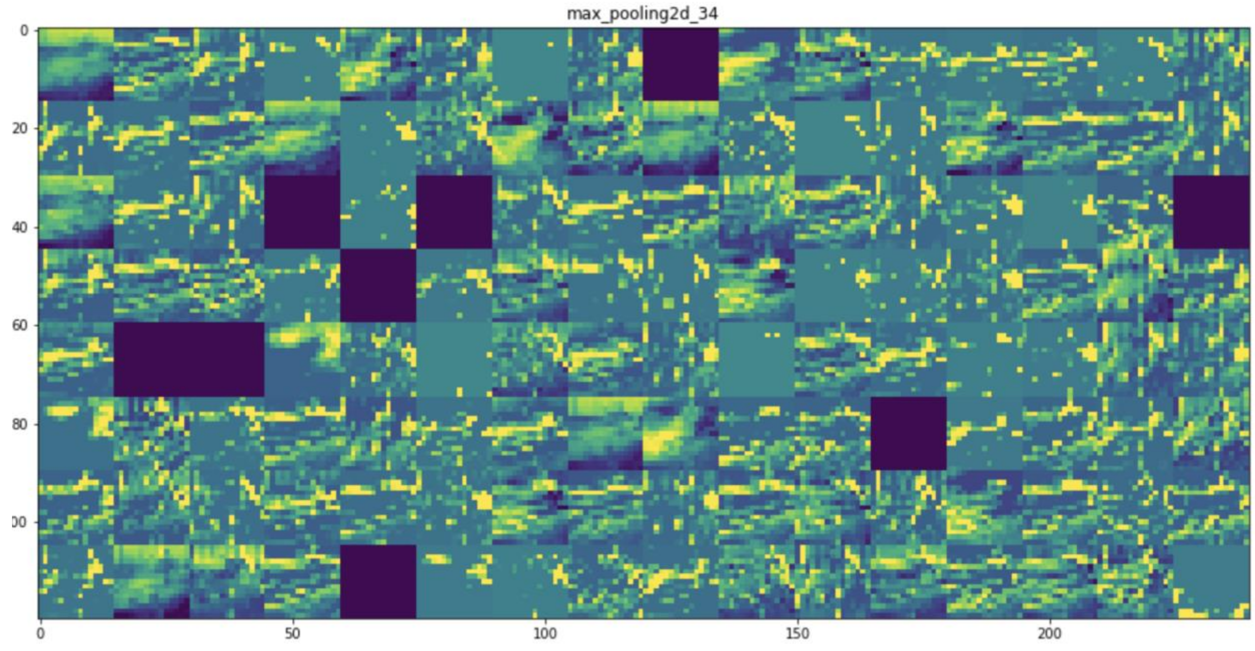


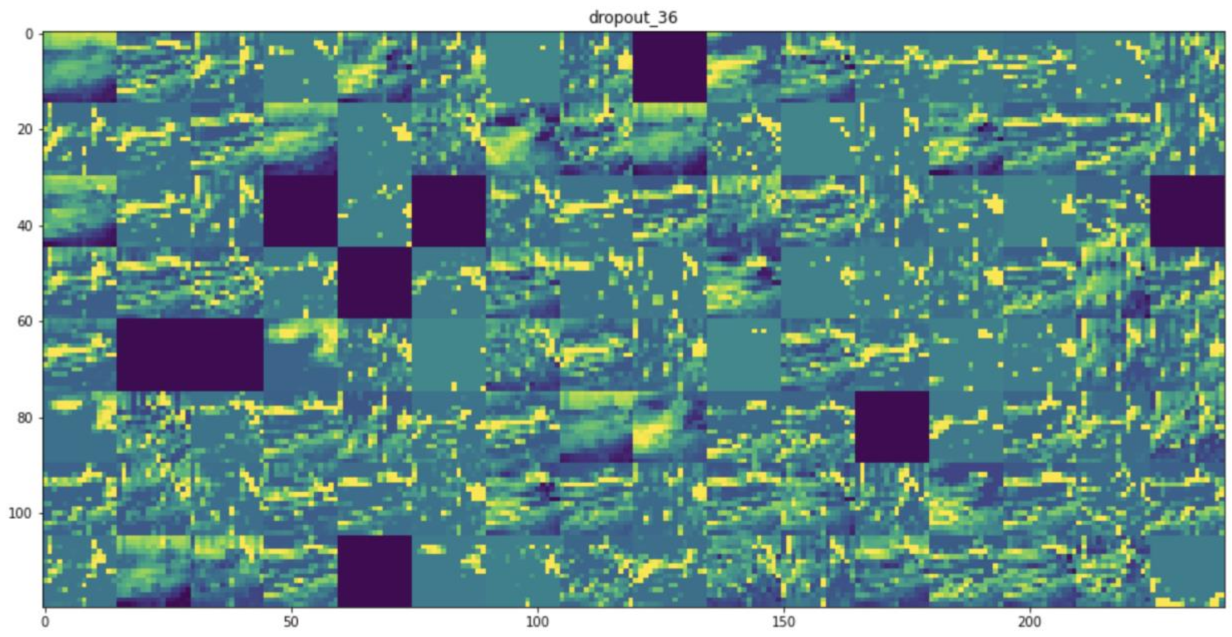
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
0	0.51%	0.25%	0.73%	91.08%	0.11%	2.93%	2.79%	0.06%	1.48%	0.06%
1	0.86%	2.55%	0.00%	0.01%	0.00%	0.00%	0.00%	0.00%	96.49%	0.08%
2	4.23%	2.53%	0.17%	0.41%	0.04%	0.05%	0.14%	0.04%	90.14%	2.25%
3	88.45%	2.34%	2.02%	0.79%	0.85%	0.06%	0.49%	0.05%	4.49%	0.47%
4	0.00%	0.00%	0.99%	0.28%	4.33%	0.03%	94.36%	0.00%	0.00%	0.00%
5	0.01%	0.02%	0.22%	2.10%	0.75%	1.78%	94.99%	0.09%	0.01%	0.04%
6	8.54%	55.43%	1.48%	5.77%	0.16%	1.94%	21.08%	0.45%	0.48%	4.66%
7	0.61%	0.05%	11.80%	1.92%	20.70%	1.16%	63.10%	0.45%	0.13%	0.07%
8	0.04%	0.00%	0.62%	91.23%	1.02%	6.16%	0.42%	0.48%	0.01%	0.01%
9	0.36%	93.45%	0.05%	0.07%	0.03%	0.02%	2.71%	0.00%	0.38%	2.90%
10	73.55%	0.10%	2.52%	4.82%	1.19%	4.12%	0.06%	4.01%	8.88%	0.76%
11	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.90%
12	0.14%	0.02%	2.03%	19.92%	15.74%	53.06%	1.14%	7.84%	0.05%	0.06%
13	0.01%	0.00%	0.02%	0.12%	0.08%	0.46%	0.00%	99.30%	0.00%	0.01%
14	0.00%	0.15%	0.00%	0.01%	0.00%	0.00%	0.01%	0.00%	0.02%	99.81%
15	4.49%	2.04%	1.28%	0.94%	1.05%	0.08%	59.86%	0.02%	30.11%	0.14%
16	0.01%	0.00%	0.15%	4.61%	0.01%	94.81%	0.05%	0.35%	0.01%	0.00%
17	0.18%	0.09%	0.64%	10.64%	4.65%	11.98%	0.55%	69.01%	0.10%	2.14%
18	0.22%	0.11%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.43%	0.23%
19	0.00%	0.01%	0.07%	0.08%	0.07%	0.01%	99.76%	0.00%	0.00%	0.00%

Original Picture:

Layers Output:


conv2d_40
0
50
100
150
200
0
100
200
300
400


max_pooling2d_34
0
20
40
60
80
00
0
50
100
150
200


dropout_36
0
20
40
60
80
100
0
50
100
150
200