Computer Vision: Predictive Models Using Deep and Convolutional Neural Networks

W. Andy Holst

School of Professional Studies, Northwestern University

MSDS 458: Artificial Intelligence and Deep Learning

Dr. Syamala Srinivasan

July 25, 2021

Abstract

This paper addresses how the Convolutional Neural Network (Convnet) concept provides exceptional performance in computer vision multi-class classification models over Deep Neural Networks. In our experiments we were able to achieve **80.6% Test Accuracy** with convolutional neural network techniques. Building and deploying predictive models using convolutional neural networks, which benefits today's image-based industries -- everything from facial recognition, to scene interpretation, to self-driving cars -- has experienced dramatic improvements thanks to concepts introduced in the 2010's by University of Toronto researchers who developed the CIFAR-10 dataset. Opportunities for advancement to deploy predictive models using convolutional neural network multi-layer perceptrons is now a reality and benefitting millions of people every single day.

Computer Vision: Predictive Models Using Deep and Convolutional Neural Networks

Computer vision has many intriguing business use cases, one of which includes providing enhanced security for mobile devices, such as the iPhone, via facial recognition. With today's technology, and especially with recent advances in computer vision, the prospect of facial recognition for enhanced security is now a reality. Specifically, convolutional neural networks (Convnets) are a subset of neural networks, and their technology utilizes automated feature engineering through the application of different self-learned "filters" across images. The rapid development of Convnets in the early 2010's, led by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (University of Toronto), revolutionized the field of computer vision.

Computer vision at large experienced a significant amount of progress during the 2010's. Every year from 2010 until 2017, a group from Stanford University and Princeton University called "ImageNet" hosted an annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluated algorithms for object detection and image classification at large scale. By 2012 the team of Krizhevsky, Sutskever, and Hinton had won the ILSVRC competition multiple times using Convnets, in 2012 they posted a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry (Krizhevsky et al., 2017).

With this development in mind and all of the prospective business opportunities that arise from computer vision -- everything from facial recognition, to scene interpretation, to self-driving cars -- we explore convolutional neural networks in full detail in this research paper. The different hyperparameters explored include number of Convnets layers, number of filters, size of filters, how quickly filters move across images (known as "stride"), consolidating images via a process called max-pooling, and a final, fully connected Dense neural network layer that provides the interpretation of the convoluted layers and filters that are applied in the modeling process.

Data

The dataset used for this research -- the CIFAR-10 database of color images -- was originally compiled by Antonio Torralba, Rob Fergus and William T. Freeman and then painstakingly curated and labeled by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It consists of a training set of 60000 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.. The digits have been size-normalized to 32 pixels by 32 pixels square. The data preparation, exploratory data analysis and visualization steps undertaken to prepare the dataset for subsequent research, analysis, and modeling were as follows:

- A Google Colaboratory Notebook (Python 3.7.11) was created, with GPU enabled.
- Python Packages imported as necessary for pre-processing and modeling, including: Numpy (1.19.5), Pandas (1.1.5), Tensorflow (2.5.0), Sci-kit Learn (0.22.2) (*for Confusion Matrix*), Matplolib (3.2.2). Seaborn (0.11.1), and Time.
- Importing the CIFA-10 database of color images (60,000 training images and 10,000 test images), which is available using the Tensorflow API:

  `tensorflow.keras.datasets.cifar10.load_data()`
- Splitting the training dataset into 47,000 training images and a 3,000 validation images. So the shapes of all six datasets ready for research and modeling are as follows:
  - `x.train` shape `(47000,32,32,3)`: Input variables for training the models
  - `y.train` shape `(47000,10)`: Output labels for training the models
  - `x.val` shape `(3000,32,32,3)`: Input validation variables during training
  - `y.val` shape `(3000,10)`: Output validation labels during training
  - `x.test` shape `(10000,32,32,3)` : Test input variables for prediction
  - `y.test` shape `(10000,10)` : Test output labels to measure accuracy

Methods: Research Design and Modeling

Over the course of the fourth week of the Summer Term 2021, I conducted ten (10) experiments on the CIFAR-10 dataset, all of which consisted of a multiclass classification problem. The images are labeled as one of 10 different types of images: `['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']`, and are all evenly distributed in count. The neural network models explored in these experiments consisted initially of 2-hidden-layer and 3-hidden-layer deep neural networks (Experiments 1 and 2). Later models utilized the Convolutional Neural Network (Convnet) concept, and different hyperparameters explored include number of Convnets layers, number of filters, size of filters, how quickly filters move across images (known as “stride”), consolidating images via a process called max-pooling, and a final, fully connected Dense neural network layer that provides the interpretation of the convoluted layers and filters that are applied in the modeling process. All models were constructed in a `Sequential` layer-fashion using Keras (Python library was built by and described eloquently by Francois Chollet in “Deep Learning with Python”) (Chollet, 2018). The input layer consists of input in the shape of `(32, 32, 3)`. The single hidden layers are all fully connected (`Dense`) layers with `relu` activation function, and various nodes (ranging from `1` to `256`) are explored throughout the Experiments. The last layer uses a `softmax` activation with `10` nodes, outputting a probability distribution over the 10 different output classes. The loss function used in these experiments was `sparse_categorical_crossentropy`, and the optimization algorithm used was `Adam`. A validation set was created by setting apart `3,000` samples from the original training data, to minimize overfitting.

Because this problem involves image recognition, and the evolution of this type of data is expected to be frequently updated as more and more images proliferate the Internet, it is

expected that this model will require periodic updating. However, as Chollet outlines in his text *Deep Learning with Python,* there are several means of “freezing” portions of computer vision models and utilizing them for “transfer learning.” Thus, updating of these types of image recognition models is expected to be infrequent and will only enhance performance of future Convnets.

Experiment 1

Our very first deep neural network consisted of a first flatten layer which flattens each `(32x32x3)` image array into a `(3072)` linear array. Next are two hidden layers of `256` input nodes each, and then a final output layer of `10` output nodes (corresponding to the 10 image categories). Regularization is explored for the hidden layers of the model, by first compiling and fitting the model without regularization and then compiling and fitting the model with L2 Regularization (with `L2=0.003`) This model consists of a total of `855,050` parameters (including weights and bias). After training the model using `50` Epochs and Batch Size of `1024`, the test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 2

Our very first deep neural network consisted of a first flatten layer which flattens each `(32x32x3)` image array into a `(3072)` linear array. Next are two hidden layers of `256` input nodes each. Then a third hidden layer of `64` input nodes is added. A final output layer of `10` output nodes (corresponding to the 10 image categories). Regularization is explored for the hidden layers of the model, by first compiling and fitting the model without regularization and then compiling and fitting the model with L2 Regularization (with `L2=0.003`) This model consists of a total of `869,578` parameters (including weights and bias). After training the model

using `50` Epochs and Batch Size of `1024`, the test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 3

Experiment 3 consists of a convolutional neural network consisting of two “layer pairs” of Convnet Layer + Max Pool layer. The 2D Convnet Layer contains `256` filters, uses a filter size of `(3x3)` pixels, with a stride of `(1x1)`, and no padding (thus reducing each array from `(32x32)` down to `(30x30)`. Following the 2D Convnet Layer is a 2D Max-Pool Layer which uses a max-pool of `(2x2)` and a stride of 1 in order to reduce the array size from `(30x30)` down to `(15x15)`. A second Convnet Layer + May Pool Layer is added with the same specifications. Next comes a flatten layer which flattens the final `(6x6x256)` filter array into a `(9216)` linear array. A fully-connected Dense layer of `256` nodes processes the filter outputs. A final output layer of `10` output nodes (corresponding to the 10 image categories). Regularization is explored for the hidden layers of the model, by first compiling and fitting the model without regularization and then compiling and fitting the model with L2 Regularization (with `L2=0.003`) This model consists of a total of `2,959,370` parameters (including weights and bias). After training the model using `50` Epochs, Batch Size of `1024`, and Early Stopping with `Patience=3`. The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 4

Experiment 4 consists of a convolutional neural network consisting of **three** “layer pairs” of Convnet Layer + Max Pool layer. The 2D Convnet Layer + Max Pool layer “pair” matches the specifications of Experiment 3. A second and a third Convnet Layer + Max Pool Layer are

added with the same specifications. Next comes a flatten layer which flattens the final `(2x2x256)` filter array into a `(1024)` linear array. A fully-connected Dense layer of `256` nodes processes the filter outputs. A final output layer of `10` output nodes (corresponding to the 10 image categories). Regularization is explored for the hidden layers of the model, by first compiling and fitting the model without regularization and then compiling and fitting the model with L2 Regularization (with `L2=0.003`). **In addition a 20% Drop Out Layer is added between each Convnet + Max Pool pairing.** This model consists of a total of `1,452,298` parameters (including weights and bias). After training the model using `50` Epochs, Batch Size of `1024`, and Early Stopping with `Patience=3`. The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 5

Experiment 5 consists of a convolutional neural network consisting of **three** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 4**, ***except*** the number of filters is ***increased*** from `64` filters in the First Convnet, to `128` filters in the Second Convnet, and `256` filters in the Third Convnet. A fully-connected Dense layer of `512` (increased in this experiment) nodes processes the filter outputs. This model consists of a total of `900,746` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 6

Experiment 6 consists of a convolutional neural network consisting of **three** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 4**, ***except*** the option of `padding = 'same'` is added to each Convnet Layer. This adds a border around

each image to ensures that the image size is not reduced in the Convnet Layer. This model consists of a total of `2,238,730` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 7

Experiment 7 consists of a convolutional neural network consisting of **four** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 6** (including `padding = 'same'` but one more Convnet Layer + Max Pool pair is added. This model consists of a total of `2,042,378` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 8

Experiment 8 consists of a convolutional neural network consisting of **four** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 7** (including `padding = 'same'` but this time **the stride of the first layer is increased from `(1,1)` to `(2,2)`**. This model consists of a total of `1,845,770` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 9

Experiment 9 consists of a convolutional neural network consisting of **three** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 6** (including

`padding = 'same'` but this time **the stride of the last layer is increased from `(1,1)` to `(2,2)`**. This model consists of a total of `1,845,770` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Experiment 10

Experiment 10 consists of a convolutional neural network consisting of **three** "layer pairs" of Convnet Layer + Max Pool layer, with the same specifications as **Experiment 6** (including `padding = 'same'`) but this time **the size of filters is increased from `(3,3)` to `(5,5)`**. The idea was that these images are very small, only 32x32 pixels. So, there is likely important features in each image which span more pixels than higher resolution images. This model consists of a total of `5,200,394` parameters (including weights and bias). The test loss and accuracy, and model build time, are recorded and the `confusion_matrix`; is reviewed. Visual lineplots of accuracy and loss over time (ie. each epoch) are created for the training and validation datasets.

Results

Experiment 1

In Experiment 1, it was determined that the **Test Accuracy = 52.7% but then reduced to 51.5% when correcting for overfitting**. This is better than a "random" guessing (which has an expected accuracy of 10%). But this is certainly not good enough to place into production.

Experiment 2

In Experiment 2, it was determined that the **Test Accuracy = 52.4% but then reduced to 50.2% when correcting for overfitting**. This is not much better than the Experiment 1 results.

Experiment 3

It was expected and found that the Test Accuracy and model performance increases rather dramatically with the introduction of Convolutional Neural Networks. In Experiment 3 it was determined that the **Test Accuracy = 72.8% and then decreased slightly to 71.2% after correcting for overfitting**. One interesting aspect of L2 Regularization that I noticed was the Validation Accuracy became much more erratic, even though the Validation Loss and Training Loss were much more stable. The reasons for this erratic Validation Accuracy isn't entirely known, but as Experiment 4 shows, it appears to temper once Drop Out layers are added.

Experiment 4

In Experiment 4 it was determined that the **Test Accuracy = 72.6% and then increased significantly to 77.4% after correcting for overfitting**. Two interesting aspects of L2 Regularization with Drop Out added: 1) previously noted "erratic" behavior of the Validation Loss became tempered, as shown in Figure 1 below, and 2) even though inserting Drop Out layers in the Convnet+Max Pool pairs has been determined to not have a mathematical reason behind it, taking this approach actually led to a significant increase in model accuracy. This should be explored further in future research (Srinivasan, 2021).

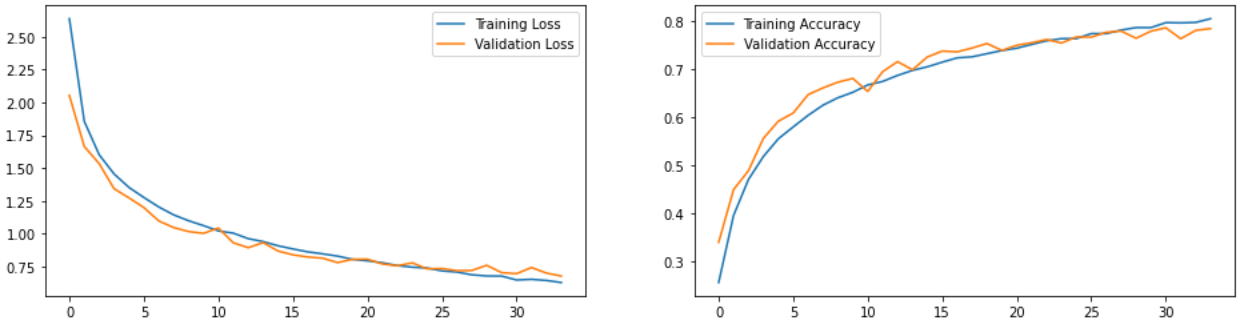

Figure 1 - Controlling overfitting by introducing L2 Regularization and Drop Out Layer

Experiment 5

In Experiment 5 it was determined that the **Test Accuracy = 75.7%**. Interestingly, the changing of the number of filters actually decreased the accuracy of the model. Thus, future experiments returned to a consistent `256` filters across all Convnet Layers. As the next Experiment points out this decision was rewarded with the addition of padding.

Experiment 6

In Experiment 6 it was determined that the **Test Accuracy = 76.7%**. Padding around the edges did not improve the overall accuracy of the model, however it did allow for the final pairing of Convnet + Max Pooling to have a dimension of `4` rather than `2`, which allows for an additional pair of layers to be added to the network (see Experiment 7)

Experiment 7

In Experiment 7 it was determined that the **Test Accuracy = 80.6%**. Improvement over previous models was accomplished by introducing a setting of `padding = 'same'` within each Convnet layer and then adding a fourth pair of Convnet+Max Pool layers. Padding corrects for loss of information around the edges of each image by adding a border prior to applying the filters, so that the filters can capture characteristics all the way to the edge of each image. It also

maintains the size of the matrix at each layer (prior to max pooling), which allows for an additional pair of layers to be added to the network.

This model performed the best of all models in this research. performed the best of all models in this research paper. This improvement is most likely due to the fact that these images are very small, only 32x32 pixels. So, there is likely important features towards the edge of each image. Since this is the best performing model, I present in Figure 2 and Training / Validation Loss and Accuracy curves and in Figure 3 I present the Confusion Matrix which displays very few misrepresented values.

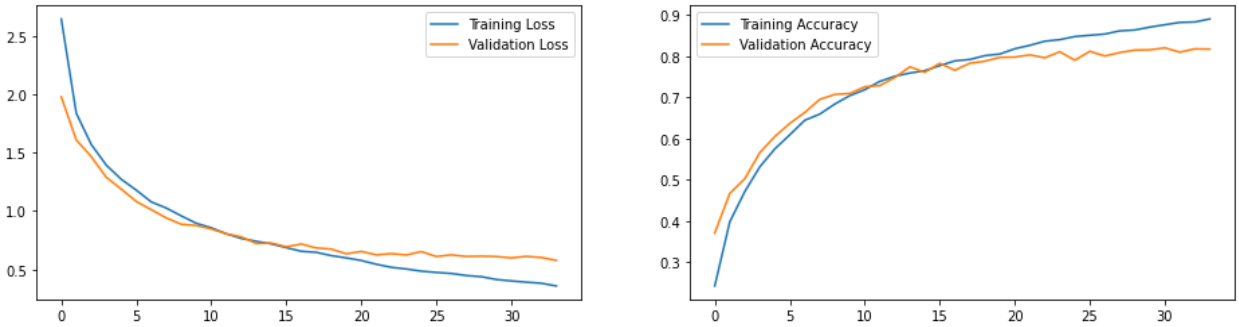

Figure 2 - Training / Validation Loss and Accuracy Curves for Experiment 7 (Larger Filter Size)

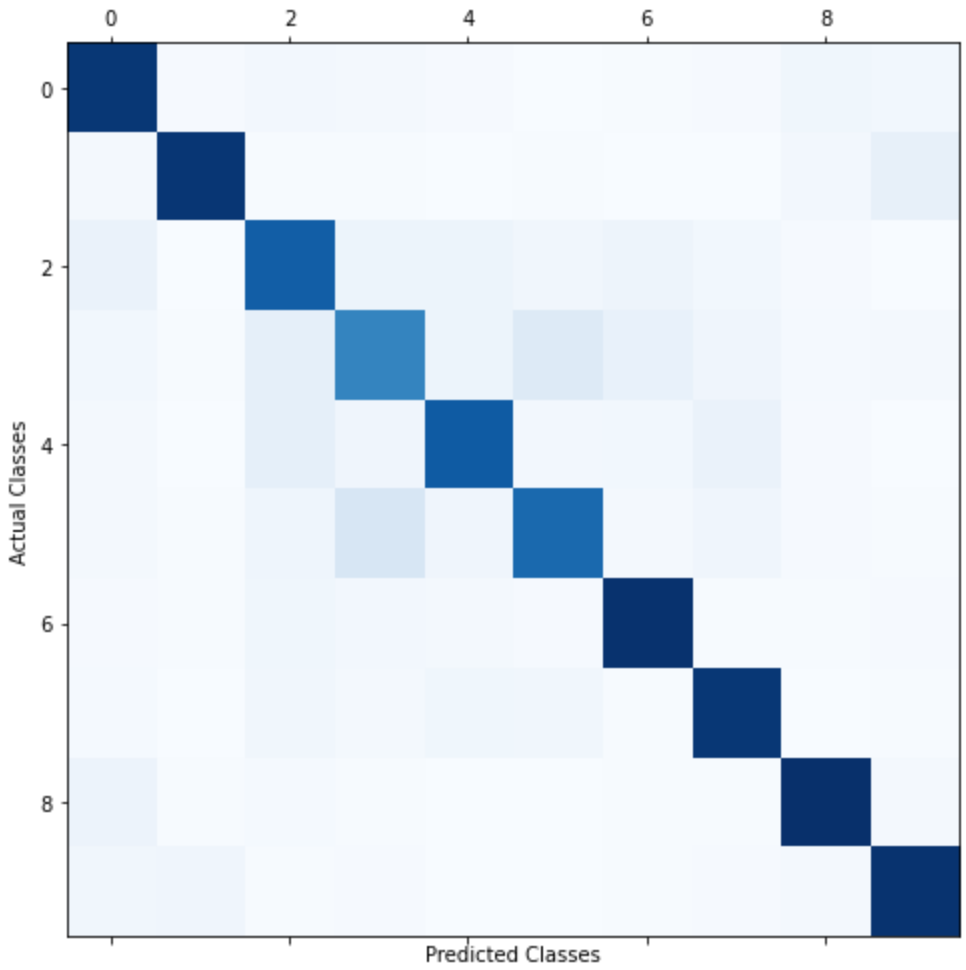

Figure 2 - Final Confusion Matrix for Experiment 7

Experiment 8

In Experiment 8 it was determined that the **Test Accuracy = 77.1%**. Therefore, changing the magnitude of the stride on the first Convnet Layer does not help or improve the performance on this set of images.

Experiment 9

In Experiment 9 it was determined that the **Test Accuracy = 71.6%**. Therefore, changing the magnitude of the stride on the final Convnet Layer does not help or improve the performance on this set of images.

Experiment 10

In Experiment 10 it was determined that the **Test Accuracy = 81.4%**. Therefore, changing the size of each filter (from (3,3) to (5,5) does not necessarily help or improve the performance on this set of images, and it dramatically increased model build time.

Analysis and Interpretation

The following Table produces final results for all ten (10) Experiments:

Experiment	Build Time (sec)	Train Accuracy (%)	Train Loss	Valid. Accuracy (%)	Valid. Loss	Test Accuracy (%)	Test Loss
1 “Deep”	31.3	62.9	1.06	54.7	1.33	52.7	1.37
1R “Deep”	32.3	53.3	1.46	52.5	1.50	51.5	1.52
2 “Deep”	42.1	64.4	1.01	54.0	1.37	52.4	1.38
2R “Deep”	41.9	52.0	1.54	50.6	1.57	50.5	1.58
3 Convnet	328.9	84.2	0.47	74.0	0.79	72.8	0.83
3R Convnet	316.0	78.2	0.80	72.6	0.95	71.2	1.00
4 Convnet	386.5	85.9	0.42	73.5	0.83	72.6	0.87
4R Convnet	547.0	80.5	0.63	78.4	0.68	77.4	0.73
5 Convnet	205.4	78.2	0.72	77.2	0.76	75.7	0.80
6 Convnet	517.5	82.5	0.63	79.4	0.75	76.7	0.80
7 Convnet	**691.0**	**89.1**	**0.36**	**81.8**	**0.58**	**80.6**	**0.63**
8 Convnet	217.0	85.8	0.43	78.1	0.68	77.1	0.72
9 Convnet	443.5	81.0	0.58	77.8	0.67	76.2	0.73
10 Convnet	1350	79.3	0.63	81.5	0.60	79.5	0.64

The following steps were instrumental towards increasing model performance: First of all, the use of Convolution Neural Networks with Max-Pooling results in a significant increase from around 50% accuracy to low 70% accuracy. Next, use of Drop Out Layers “tames” the validation accuracy and produces better results that are less susceptible to overfitting. Next, moving from two (2) Convnet + Max Pool “layer pairs” to three (3) Convnet + Max Pool “layer pairs” increase the accuracy by another ~4-5%. Then, with padding turned on and a fourth Convnet + Pool pair, the accuracy increases yet again, topping 80% accuracy. The best model performance was achieved in Experiment 7 with a **Test Accuracy of 80.6%**. This final improvement is most likely due to the fact that these images are very small, only 32x32 pixels. So, there are likely important features at the edges of each image.

Conclusions

This paper addressed how the Convolutional Neural Network (Convnet) concept provides exceptional performance in computer vision multi-class classification models over Deep Neural Networks. In our experiments we were able to achieve **80.6% Test Accuracy** with relatively convolutional neural network techniques. By building and deploying a predictive model using convolutional neural networks, industries -- everything from facial recognition, to scene interpretation, to self-driving cars -- experienced dramatic improvements in the 2010’s with regards to image recognition. Opportunities for advancement to deploy predictive models using convolutional neural network multi-layer perceptrons is now a reality and benefitting millions of people every single day.

References

Chollet, F. (2018). *Deep Learning with Python*. Manning Publications Co.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386.

Srinivasan, S. (2021, July 21). *MSDS 458 Sync Session #3* [Lecture notes, PowerPoint slides]. Northwestern University. https://canvas.northwestern.edu/courses/142844/modules/items/1977860.

Appendix

As indicated in the assignment instructions, this is “Result 2”. This provides the outputs from 2 filters from the 2 max pooling layers and visualizes them in a grid as images. As you can see, there appear to be a number of nodes not activated. This should be explored further to understand why. The original image is a BOAT: