Computer Vision:

Experimentation on Convolutional Neural Networks with CIFAR-10 Dataset

Wicky Woo[1]

Northwestern University School of Professional Studies

633 Clark St, Evanston, IL, 60208

April 30, 2022

[1] Contact email: wicky818@gmail.com

Abstract

The computer vision problem with the CIFAR-10 dataset is a classic dataset for new machine learning enthusiasts to explore how neural networks work. The dataset contains a total of 60,000 color images, each image presented as a 32 by 32 pixels matrix. Each pixel is represented by an integer ranging from 0 to 255 in a particular color plane within the RGB (Red, Green, and Blue) color model. Zero represents no color and 255 represents very intense coloration.

This research experiments with different structures of neural networks and shows the different performance metrics for these structures. The results show the drastic model performance improvements with CNNs compared to DNNs.

These experiments also show the performance improvement of utilizing ResNet, which is a short form for Residual Network. Over the years, image recognition and classification have made a series of breakthroughs with Residual Networks. This research will explore the performance of ResNet50, which is a 50-layer deep Residual Network.

Introduction

The focus of this study is to understand neural networks with multiple hidden layers and experiment with different types of multi-hidden-layer neural networks' behaviors and performances.

This exercise utilized a Jupyter Lab instance and launched the Keras library for TensorFlow 2.0 with Python. Keras is a popular deep learning API for the development of machine learning models such as computer vision and natural language processing. Keras also allows the utilization of GPUs when training neural network models. During the experiments, Python codes were executed to develop the models.

CNN models require high computing resources for training and without high-performing hardware like GPUs and TPUs, would take a significant amount of time to train. Thanks to cloud computing technologies, the experiments were conducted with a GPU which helped with training time during the development of these models.

Literature Review

A number of computer vision algorithm techniques would be applicable to the analysis of CIFAR-10 dataset, and a selection of relevant resources was explored. The important findings are summarized below.

In the paper *Transfer Learning with ResNet-50 for Malaria Cell-Image Classification*, authors A. Sai Bharadwaj Reddy and D. Sujitha Juliet were interested in solving Malaria misdiagnosis by human error. They experimented with different machine learning methods including VGG, Google Inception Model, and the Microsoft ResNet Model to teach the model to distinguish between an infected cell and an uninfected cell as shown in Image 1. The research concluded the best performing model was the ResNet-50 which achieved a validation accuracy of 95.4% in the classification of Malaria infected cells (Reddy, 2019). In this research, ResNet-50 will be evaluated with simple CNNs and DNNs.

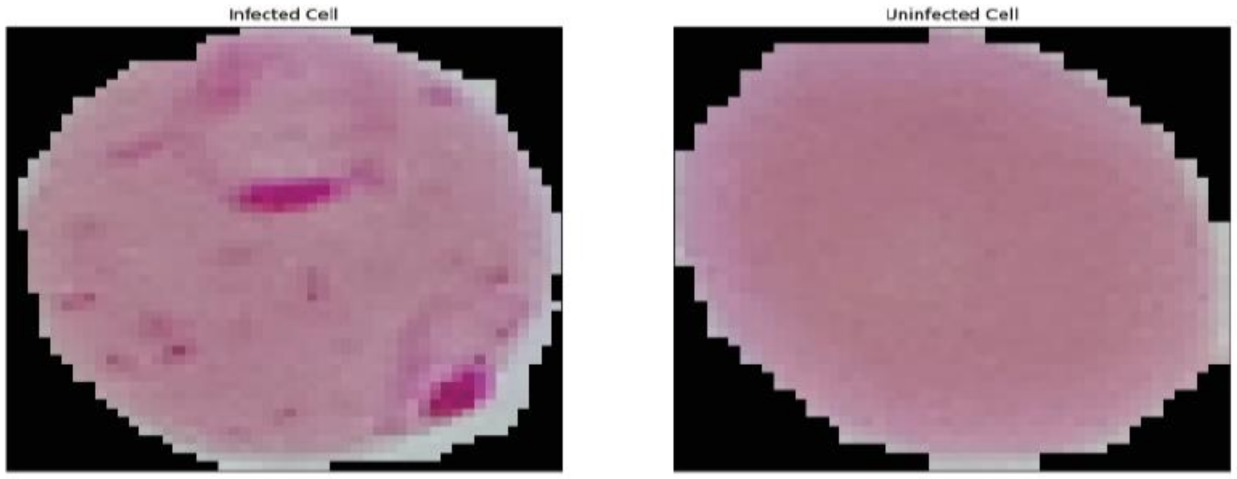


Image 1. Image of Infected and Uninfected cells (Reddy, 2019)

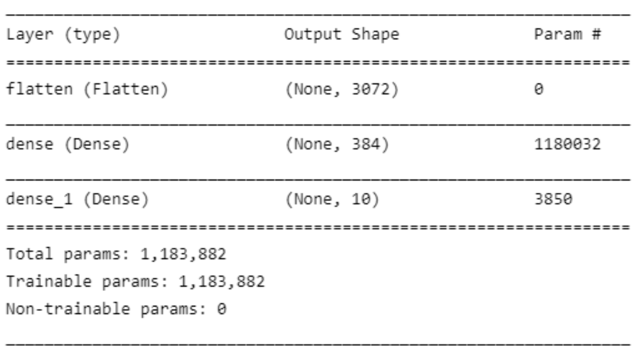
Methods

The Keras library allows easy data imports for the CIFAR-10 dataset which contains 70,000 total images. The dataset is split into a training dataset with 60,000 images and a testing dataset with 10,000 images. The dataset also contains the corresponding training dataset labels and testing dataset labels that specify the true class of the images from the training and testing datasets. Image 2 shows some examples of the images and the corresponding classes of the images.

Image 2. Example images from the training dataset and the corresponding image labels

Before diving right into the experimentation, data normalization was performed on both the test dataset and the training dataset to reduce the amount of computation during training. Furthermore, a validation set that includes 5,000 images was split from the training dataset to prevent overfitting for all of the neural network training experiments. Using a validation data set for hyperparameter tuning is a best practice for model training.

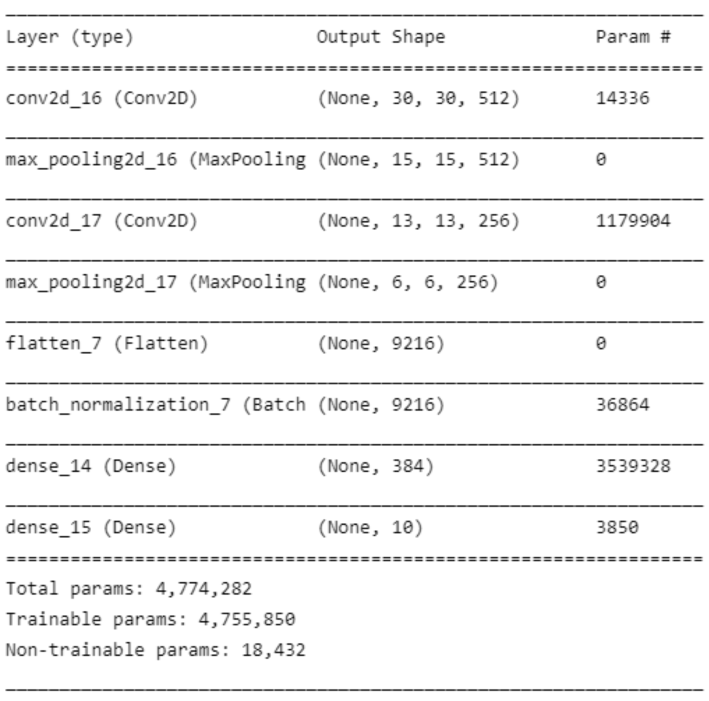
For Experiment 0.5, a Deep Neural Network (DNN) is constructed with a flattened one-dimensional input layer with 3072 nodes (32 x 32 x 3), a single hidden layer with 384 nodes with the ReLU activation function, and a SoftMax output layer of 10 nodes:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 3072)              0
_________________________________________________________________
dense (Dense)                (None, 384)               1180032
_________________________________________________________________
dense_1 (Dense)              (None, 10)                3850
=================================================================
Total params: 1,183,882
Trainable params: 1,183,882
Non-trainable params: 0
_________________________________________________________________
```

Image 3. Experiment 0.5 DNN Architecture Summary

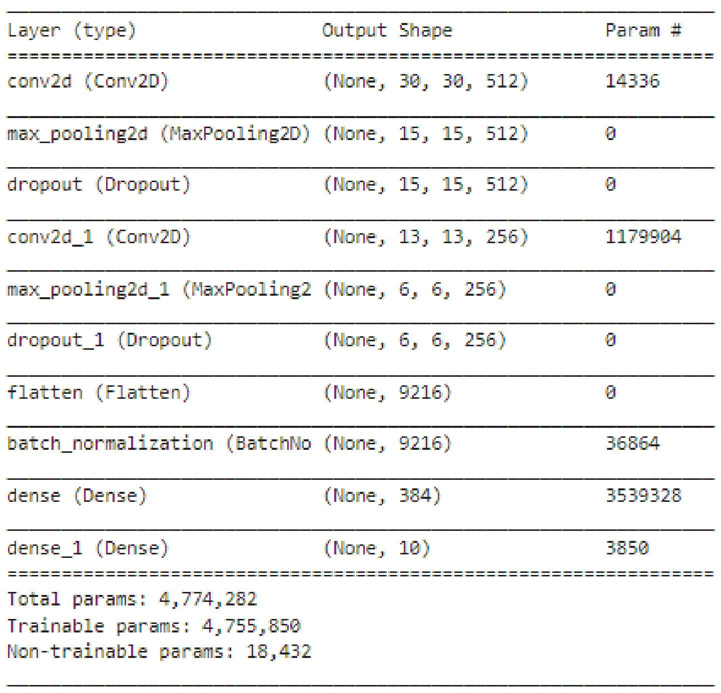
For Experiment 1, the DNN is expanded to two hidden layers with 384 nodes per hidden layer. Similarly, Experiment 2 expanded the DNN to three hidden layers with 384 nodes per hidden layer. Experiments 3 and 4 utilized CNNs with convolution/ max-pooling layers. The difference between these two experiments is the number of convolution/ max-pooling layers. Experiment 3 has two convolution/ max-pooling layers versus three convolution/ max-pooling layers in Experiment 4. Image 3 shows the CNN architecture for Experiment 3.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 30, 30, 512)       14336
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 15, 15, 512)       0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 13, 13, 256)       1179904
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 6, 6, 256)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 9216)              0
_________________________________________________________________
batch_normalization_7 (Batch (None, 9216)              36864
_________________________________________________________________
dense_14 (Dense)             (None, 384)               3539328
_________________________________________________________________
dense_15 (Dense)             (None, 10)                3850
=================================================================
Total params: 4,774,282
Trainable params: 4,755,850
Non-trainable params: 18,432
_________________________________________________________________
```

Image 4. Experiment 3 CNN with two convolution and max-pooling layers

For Experiments 5.1 to 5.4, Experiments 1-4 were repeated with regularization. Here's an example of an architecture of a CNN with regularization in Experiment 5.3:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 512)       14336
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 512)       0
_________________________________________________________________
dropout (Dropout)            (None, 15, 15, 512)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 256)       1179904
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 9216)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 9216)              36864
_________________________________________________________________
dense (Dense)                (None, 384)               3539328
_________________________________________________________________
dense_1 (Dense)              (None, 10)                3850
=================================================================
Total params: 4,774,282
Trainable params: 4,755,850
Non-trainable params: 18,432
_________________________________________________________________
```

Image 5. Experiment 5.3 CNN with two convolution and max-pooling layers with dropout layers for regularization

Experiment 5.3 can be compared to Experiment 3. They are similar outside of the fact that Experiment 5.3 is utilized dropout layers for regularization. Dropout layers temporarily disconnect some of the neurons on the previous layer. They delay overfitting by preventing any neurons from over-specializing and dominating (Glassner 2021).

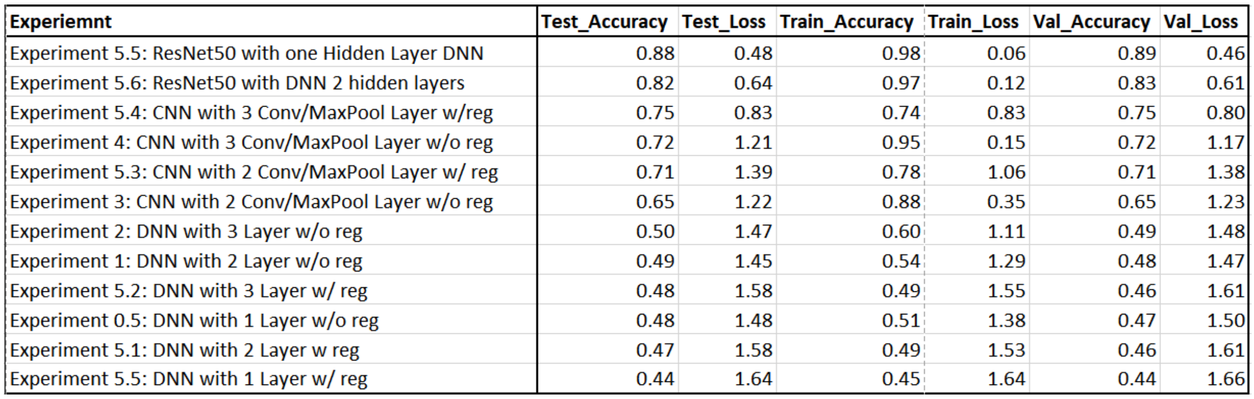
For Experiments 5.5 and 5.6, ResNet50 transfer learning is explored to predict the image classes. ResNet50 is a CNN that is 50 layers deep and is one of the best models for image recognition.

Results

Before looking at the results of the experiments, let's look at the Experiment's model training time per epoch with a GPU. The model training time can be interpreted as the amount of computation needed for model training. Although high-performance GPUs and TPUs are getting more accessible and more popular, computation resources still need to be considered when determining the ideal model. For Experiments 0.5, 1, 2, 5.1, and 5.2 which utilize DNN models, it takes about 1.3 seconds per epoch to complete. For Experiments 3, 4 , 5.3, and 5.4 which utilize simple CNNs, the model training takes about 5.1 seconds per epoch to complete. Therefore, CNNs are clearly more computationally demanding than DNNs. For Experiments 5.5 and 5.6, the model training takes about 156.3 seconds per epoch to train because these two experiments utilize ResNet50. From reviewing the training time, it was concluded that ResNet50 was the most computationally demanding neural network model.

Result 1

Result 1 is a table with the accuracy and loss for the training, test, and validation datasets:



Experiemnt	Test_Accuracy	Test_Loss	Train_Accuracy	Train_Loss	Val_Accuracy	Val_Loss
Experiment 5.5: ResNet50 with one Hidden Layer DNN	0.88	0.48	0.98	0.06	0.89	0.46
Experiment 5.6: ResNet50 with DNN 2 hidden layers	0.82	0.64	0.97	0.12	0.83	0.61
Experiment 5.4: CNN with 3 Conv/MaxPool Layer w/reg	0.75	0.83	0.74	0.83	0.75	0.80
Experiment 4: CNN with 3 Conv/MaxPool Layer w/o reg	0.72	1.21	0.95	0.15	0.72	1.17
Experiment 5.3: CNN with 2 Conv/MaxPool Layer w/ reg	0.71	1.39	0.78	1.06	0.71	1.38
Experiment 3: CNN with 2 Conv/MaxPool Layer w/o reg	0.65	1.22	0.88	0.35	0.65	1.23
Experiment 2: DNN with 3 Layer w/o reg	0.50	1.47	0.60	1.11	0.49	1.48
Experiment 1: DNN with 2 Layer w/o reg	0.49	1.45	0.54	1.29	0.48	1.47
Experiment 5.2: DNN with 3 Layer w/ reg	0.48	1.58	0.49	1.55	0.46	1.61
Experiment 0.5: DNN with 1 Layer w/o reg	0.48	1.48	0.51	1.38	0.47	1.50
Experiment 5.1: DNN with 2 Layer w reg	0.47	1.58	0.49	1.53	0.46	1.61
Experiment 5.5: DNN with 1 Layer w/ reg	0.44	1.64	0.45	1.64	0.44	1.66

Image 6. Results from all Experiments conducted in this research

Image 6 shows the model performances from all the experiments conducted, sorted by test accuracy. The best performing model is "ResNet50 with one Hidden Layer DNN" with a test accuracy of 88%. Image 7 shows the model architecture of the best performing model:

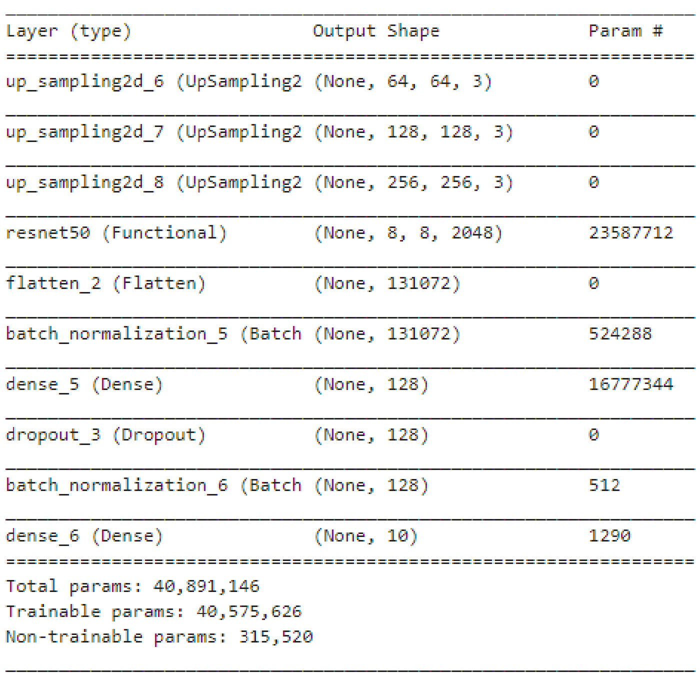

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
up_sampling2d_6 (UpSampling2 (None, 64, 64, 3)         0
_________________________________________________________________
up_sampling2d_7 (UpSampling2 (None, 128, 128, 3)       0
_________________________________________________________________
up_sampling2d_8 (UpSampling2 (None, 256, 256, 3)       0
_________________________________________________________________
resnet50 (Functional)        (None, 8, 8, 2048)        23587712
_________________________________________________________________
flatten_2 (Flatten)          (None, 131072)            0
_________________________________________________________________
batch_normalization_5 (Batch (None, 131072)            524288
_________________________________________________________________
dense_5 (Dense)              (None, 128)               16777344
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
batch_normalization_6 (Batch (None, 128)               512
_________________________________________________________________
dense_6 (Dense)              (None, 10)                1290
=================================================================
Total params: 40,891,146
Trainable params: 40,575,626
Non-trainable params: 315,520
_________________________________________________________________
```

Image 7. ResNet50 with one hidden layer DNN

In this model, three up-sampling layers were added to increase the size of the images because ResNet50 only accepts images that are greater than 200 by 200 pixels. Although the training time is longer, ResNet50's performance significantly increased from the simple CNN with three convolutional and max-pooling layers.

Result 2

For Result 2, convolution layers and max-pooling layers' outputs from Experiment 3 were graphed to visualize the creation of feature maps. Please refer to the appendix for the image outputs for the first four layers of CNN. From the image outputs, it is shown that the first convolution layer was picking up some of the important features of the image. After going through the max-pooling layer, the features are summarized as a feature map. The deeper the layers, the feature map becomes more pixelated because it's summarizing the features from the shallower layers.

Conclusion

In summary, convolutional neural networks significantly increase the model performances. Although CNNs require a longer model training time and are more computational demanding compared to DNNs, the increase in model performance easily outweighs the disadvantages. Especially when the increase in training time is typically within only 10 to 20 seconds total between DNN models and CNN models in a GPU. However, when comparing a simple CNN model to a residual network model like the ResNet50, the computational demands increased significantly from under 20 seconds to more than 20 minutes. Although the performance of ResNet50 is significant, one should consider the trade-offs and evaluate model selection carefully.

Acknowledgments

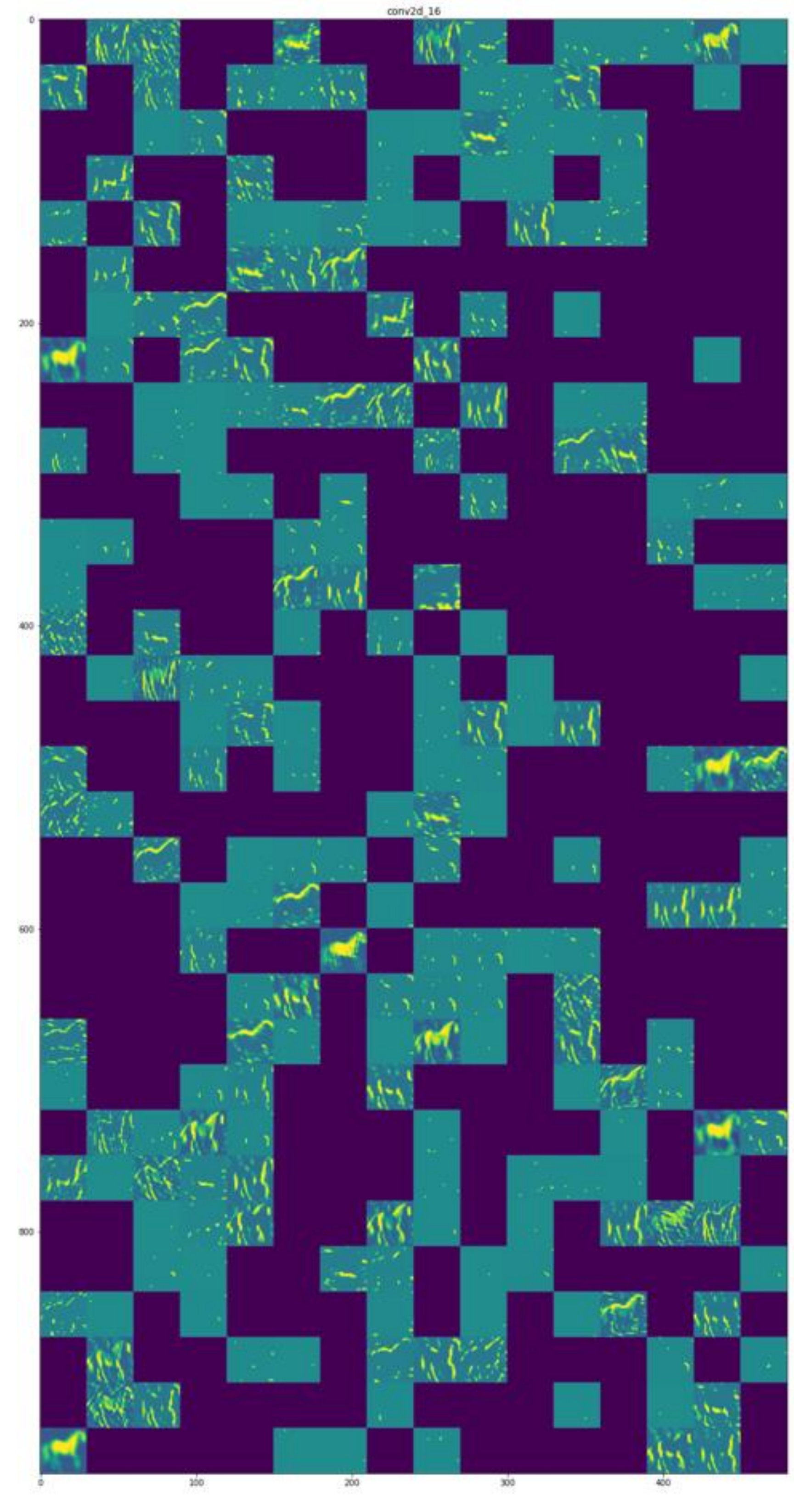
The author would like to thank Daniel Jensen for providing a code starter.

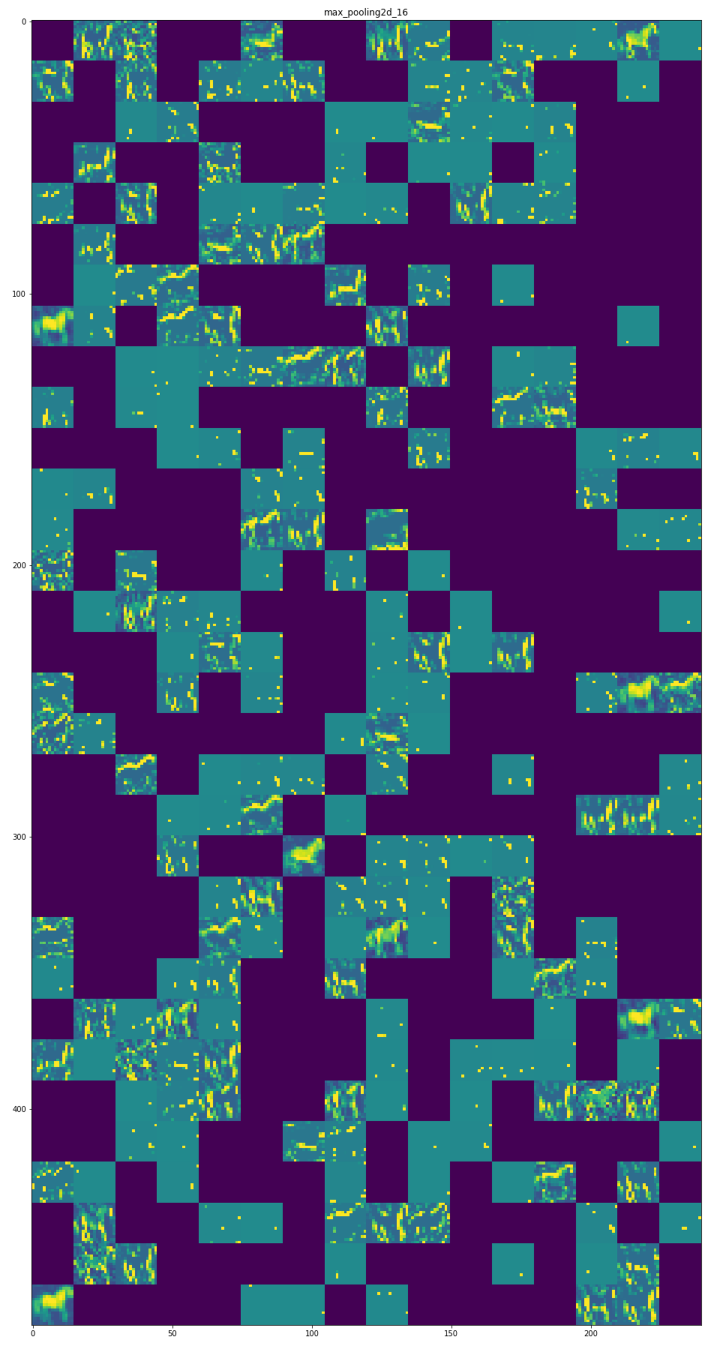
References

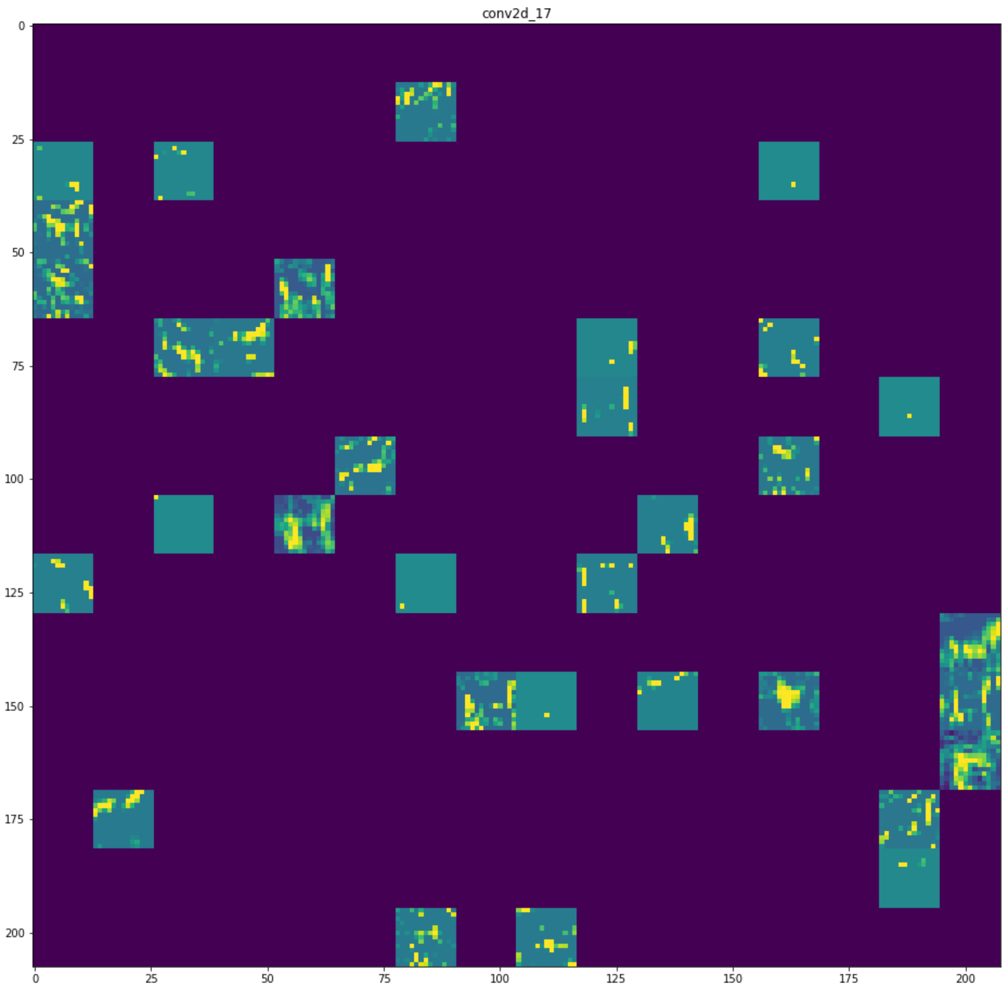
Glassner, Andrew. “Optimizers.” Essay. In *Deep Learning: A Visual Approach*, 422–25. San Francisco, CA: No Starch Press, 2021.

Reddy, A. Sai, and D. Sujitha Juliet. “Transfer Learning with Resnet-50 for Malaria Cell-Image Classification.” *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019. https://doi.org/10.1109/iccsp.2019.8697909.

Appendix




max_pooling2d_16
0
100
200
300
400
0
50
100
150
200


conv2d_17
0
25
50
75
100
125
150
175
200

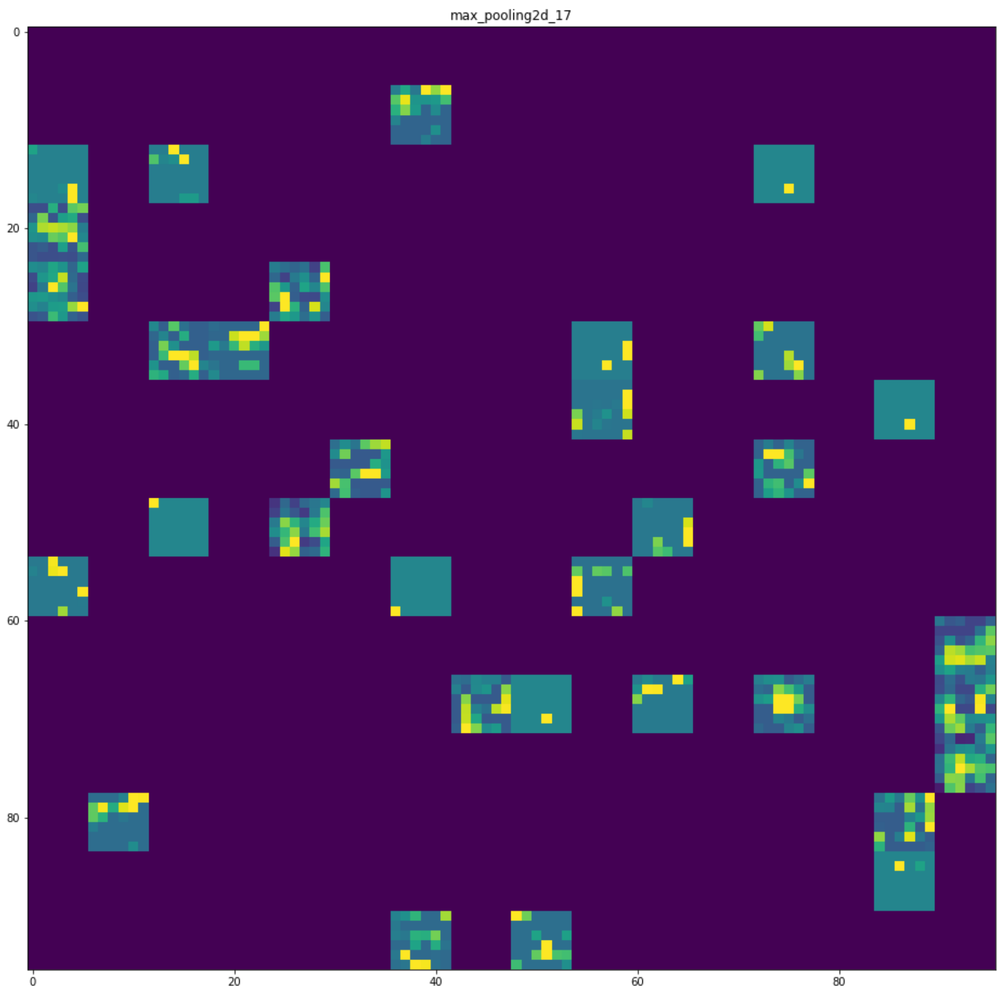


Figure 5. Looking inside CNN's convolution layers to understand and visualize the filters