Vehicle Classification

Neural Networks and Deep Learning

Abbiegael Chu

AJ23 SYD009

November 12, 2023

ABSTRACT

The recent progress of Artificial Intelligence has revolutionized numerous fields, including image processing and classification, which were greatly improved by the advancement of Convolutional Neural Networks. CNN has had many applications in a person's day-to-day life, but the most notable application is used for vehicle classification and traffic control. This report studies basic CNN models and other popular architectures, like AlexNet and LeNet-5, and its applications on a low resolution yet high quantity dataset of vehicles and non-vehicles. The report also delves into data augmentation, a regularization technique performed on images, to see its effects on overfitting. The Basic CNN model and AlexNet model showed high accuracy, with the Basic CNN model scoring the lowest test loss. LeNet-5 had the lowest accuracy. However, it was the only one to correctly classify uploaded test images.

TABLE OF CONTENTS

INTRODUCTION

With the rise of Artificial Intelligence, new developments have been utilized in government services, business development, and more. Image classification has many applications in a person's day-to-day life, from identifying what the object in the image is to identifying text on an image. Convolutional Neural Networks (CNN) is the main tool used by researchers for further developments in the field of computer vision, including vehicle classification.

A common CNN project for aspiring data scientists is vehicle classification, wherein the model is tasked to identify if the object in the image or video is a vehicle or not. The project can be applied to many real-life scenarios, such as traffic management, car surveillance, etc. In the Philippines, for example, the government had implemented a surveillance system that identifies cars who have failed to follow road regulations. The cars identified will be digitally sent a fine as well as sent the video of the incident.

LITERATURE REVIEW

CNN

Similar to how neural networks were inspired by biological neurons, the convolutional neural networks were inspired by the visual cortex (Géron, 2017). (Hubel D. H., 1959) and (Hubel & Wiesel, 1959) studied the visuals of cats and gain insights about the visual cortex. In those studies, the authors found that the cats reacted to only a limited region of their vision, which is the local receptive field. The local receptive fields had overlaps with each other, but overall, they encompassed the whole visual field. In their studies, they also realized that groups of neurons were reacting to lines with different orientations. Some neurons were only reacting to horizontal lines, while others reacted to diagonal lines.

Arguably, the most important type of layer in a CNN model is its convolutional layer (Géron, 2017). Convolutional layers pass through the

inputs and send the results into the next layer. These layers mimic neural responses to visual stimuli (O'Shea & Nash, 2015). In the same study, the authors also delved deeper into the pooling layer and fully-connected layers. The pooling layer reduces the parameters in the activation, while the fully-connected layer attempts to class scores in classification.

ALEXNET

AlexNet was a CNN architecture developed by (Krizhevsky, Sutskever, & Hinton, 2012). The network won the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), beating the competition by having a less than 25% error rate among the participants (Pinecone, n.d.). Retrieved from (Krizhevsky, Sutskever, & Hinton, 2012), Figure 1 shows AlexNet's structure starting with an input layer of 227x227x3, the parameters being height, width, and RGB colors respectively. The five convolutional layers had varying number of kernels, sizes, strides, and padding. The model also contains three max pooling layers to reduce the spatial sizes of some of the convolutional layers. The team incorporated two regularization techniques, dropout and data augmentation, to reduce overfitting, and a normalization technique called local response normalization, which encouraged competition among the strongly activated neurons in the same area (Krizhevsky, Sutskever, & Hinton, 2012). The activation function used for this network was Rectified Linear Unit (ReLU) to help with the vanishing gradient problem.

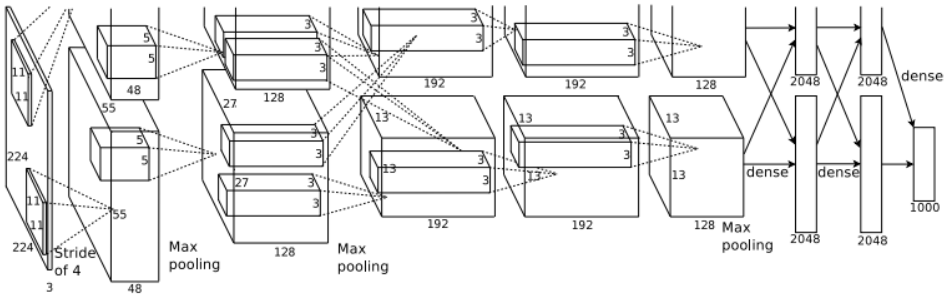

Figure 1 AlexNet Architecture

LENET-5

LeNet-5 was developed by Yann Lecun (Lecun, Bottou, Bengio, & Haffner, 1998). The model contains three convolution layers, two average pooling layers, and two fully-connected layers. However, the activation function of LeNet-5 is radial basis function (RBF). According to (Lecun, Bottou, Bengio, & Haffner, 1998), their training mainly used backpropagation and gradient descent. Figure 2 below shows a breakdown of the LeNet-5 model retrieved from (Géron, 2017).

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully connected	–	10	–	–	RBF
F6	Fully connected	–	84	–	–	tanh
C5	Convolution	120	1 × 1	5 × 5	1	tanh
S4	Avg pooling	16	5 × 5	2 × 2	2	tanh
C3	Convolution	16	10 × 10	5 × 5	1	tanh
S2	Avg pooling	6	14 × 14	2 × 2	2	tanh
C1	Convolution	6	28 × 28	5 × 5	1	tanh
In	Input	1	32 × 32	–	–	–

Figure 2 LeNet-5 Architecture

METHODOLOGY

The student opted for a pipeline that starts with loading the data. Although Google Collaboratory was the preferred choice, uploading the image dataset in each run, which contains 177k images, was not practical in terms of time, so the student chose to run the notebook locally. After connecting the notebook to the respective folders, the student split the data to train and test, having a test size of 20%.

Next in the pipeline was to use data augmentation to pre-process the training data. The ImageDataGenerator from TensorFlow increases training set size by artificially creating variants of the training set by rotating the image, shifting the focus to only one size, darkening the image, zooming in, cropping the image, and brightening the training set. The student's initial notebook,

which did not use data augmentation, had problems with overfitting and predicting false positives when evaluating the model, so with this in mind, the student added data augmentation in the new pipeline.

Since the images from the dataset are 64x64, the student set 64x64 as the size for training the models. After setting the parameters, the pipeline goes to the CNN model. To gain a better understanding of CNN, three models were a Basic CNN model, AlexNet model, and LeNet-5 model. The models were trained with the same parameters, with minor adjustments to AlexNet and LeNet-5 to suit the dataset's characteristics. The optimizer for the models was adam, loss was binary crossentropy, and metrics was accuracy. All were trained with 10 epochs with varying training times due to the architecture of the models.

The models were evaluated based on test accuracy and test loss. Moreover, the student opted to upload an image and test the models, images of a car and Simu Liu. After determining the best model among the three, the model was then applied to a Streamlit app that lets the user upload an image for a prediction.

RESULTS OF CNN MODELS

This section goes into detail about the layers of the respective models, highlighting the differences between each as well as the time it took to train. The section will also go over the accuracy and loss of both training and testing datasets. Overall, in this report, we will be comparing three CNN models and attempting to identify the reasons why these models have different results from each other while also formulating suggestions on what works best for this dataset.

BASIC CNN

The model used for this network was a combination of the basic CNN model showed in class as well as some models from (Géron, 2017). The model is sequential model with three layers of convolution and max pooling, a flattening layer, one fully-connected layer, and an output layer, totaling to 1,273,409 parameters as seen in Figure 3.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 62, 62, 32)        896

 max_pooling2d_3 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 29, 29, 64)        18496

 max_pooling2d_4 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 12, 12, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 flatten_1 (Flatten)         (None, 4608)              0

 dense_2 (Dense)             (None, 256)               1179904

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 1,273,409
Trainable params: 1,273,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 Basic CNN Model

The test accuracy for this model resulted in 0.99521, while the test loss resulted in 0.0196. Furthermore, the student also tried evaluating the basic CNN model by using a car image and Simu Liu image. Both images were predicted to be a vehicle, indicating overfitting.

ALEXNET

The AlexNet model was adopted in this project, with some adjustments to fit the dataset. For the input shape, instead of having 227x227, it was changed to 64x64. Moreover, the strides of the first convolutional layer were reduced to accommodate the smaller image sizes in the dataset. Padding parameters were set to 'same' to ensure regularity in the spatial dimensions of the input. Batch normalization was also added to help speed up training. The number of neurons on the fully-connected layers were also lowered to 1024 due

to the smaller image size of the dataset. Among the three models, AlexNet had the highest number of layers, totaling to 5,861,125.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 16, 16, 96)        34944

batch_normalization (BatchN  (None, 16, 16, 96)        384
ormalization)

activation (Activation)      (None, 16, 16, 96)        0

max_pooling2d (MaxPooling2D  (None, 8, 8, 96)          0
)

conv2d_1 (Conv2D)            (None, 8, 8, 256)         614656

batch_normalization_1 (Batc  (None, 8, 8, 256)         1024
hNormalization)

activation_1 (Activation)    (None, 8, 8, 256)         0

max_pooling2d_1 (MaxPooling  (None, 4, 4, 256)         0
2D)

conv2d_2 (Conv2D)            (None, 4, 4, 384)         885120

batch_normalization_2 (Batc  (None, 4, 4, 384)         1536
hNormalization)

activation_2 (Activation)    (None, 4, 4, 384)         0

conv2d_3 (Conv2D)            (None, 4, 4, 384)         1327488

batch_normalization_3 (Batc  (None, 4, 4, 384)         1536
hNormalization)

activation_3 (Activation)    (None, 4, 4, 384)         0

conv2d_4 (Conv2D)            (None, 4, 4, 256)         884992

batch_normalization_4 (Batc  (None, 4, 4, 256)         1024
hNormalization)

activation_4 (Activation)    (None, 4, 4, 256)         0

max_pooling2d_2 (MaxPooling  (None, 2, 2, 256)         0
2D)

flatten (Flatten)            (None, 1024)              0

dense (Dense)                (None, 1024)              1049600

batch_normalization_5 (Batc  (None, 1024)              4096
hNormalization)

activation_5 (Activation)    (None, 1024)              0

dropout (Dropout)            (None, 1024)              0

dense_1 (Dense)              (None, 1024)              1049600

batch_normalization_6 (Batc  (None, 1024)              4096
hNormalization)

activation_6 (Activation)    (None, 1024)              0

dropout_1 (Dropout)          (None, 1024)              0

dense_2 (Dense)              (None, 1)                 1025

batch_normalization_7 (Batc  (None, 1)                 4
hNormalization)

activation_7 (Activation)    (None, 1)                 0

=================================================================
Total params: 5,861,125
Trainable params: 5,854,275
Non-trainable params: 6,850
_________________________________________________________________
```

Figure 4 Modified AlexNet Model

The results of this model show a test accuracy of 0.9916, and a test loss of 0.0432. The test accuracy was as high as the basic CNN model performed previously, but this model seems to have a higher loss. It is also important to note that among the three models, the modified AlexNet model had the longest running time, averaging 497ms/ step, while the other networks had less than 200ms per step. Using the same car and Simu Liu images, the predicted classes for both were still vehicles.

LENET-5

The LeNet-5 model was also adopted for this project with some modifications to fit the dataset. First, the original LeNet-5 model was for grayscale images of 32x32. In the code, the student had adjusted the input shape as 63x63x3 to include RGB colors. The output layer was also updated to a sigmoid activation function to match the vehicle-non-vehicle nature of this

classification project. Among the three models, the LeNet-5 had the lowest number of parameters, totaling to 867,641.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)            (None, 60, 60, 6)         456

 average_pooling2d_2 (Averag  (None, 30, 30, 6)         0
 ePooling2D)

 conv2d_4 (Conv2D)            (None, 26, 26, 16)        2416

 average_pooling2d_3 (Averag  (None, 13, 13, 16)        0
 ePooling2D)

 conv2d_5 (Conv2D)            (None, 9, 9, 120)         48120

 flatten_1 (Flatten)          (None, 9720)              0

 dense_2 (Dense)              (None, 84)                816564

 dense_3 (Dense)              (None, 1)                 85

=================================================================
Total params: 867,641
Trainable params: 867,641
Non-trainable params: 0
_________________________________________________________________
```

Figure 5 Modified LeNet-5 Model

The model evaluation shows that the modified LeNet-5 model has the lowest accuracy among the three models, 0.85755, while having the highest loss at 0.34959. However, when the student tested the car and Simu Liu images, the model correctly identified that the car image is a vehicle, while the Simu Liu image was a non-vehicle.

RESULTS

	Basic CNN	AlexNet	LeNet-5
Test Accuracy	0.9952	0.9916	0.8575
Test Loss	0.0197	0.4321	0.3496
Avg ms/step	182.8	497	139.8
Classification: Car	/	/	/
Classification: Simu Liu	X	X	/

Figure 6 Model Results

The results show that the Basic CNN and AlexNet had high accuracies, while only Basic CNN had the lowest test loss. The Basic CNN model performed in the middle when compared to AlexNet, who was the highest among the group, and LeNet-5, the lowest among the group. All models have successfully identified the car image as a vehicle, while only LeNet-5 is the only one that classified Simu Liu as a non-vehicle.

Depending on individual needs, if accuracy is the most important parameter, then Basic CNN should be chosen among the three. However, if the individual is basing their decision on the correct classification done from manual testing, LeNet-5 should be chosen. Moreover, LeNet-5 had the lowest training time. In this case, the student has chosen LeNet-5 based on the model's ability to correctly classify the car and Simu Liu images.

CHALLENGES

Although the CNN architectures used in this report have already been well researched, the student faced many challenges when executing this project. Training the models proved difficult due to the long training time needed by their computer. The basic CNN model and the LeNet-5 took around the same amount of time. However, the AlexNet model took at least three times longer to run due to the model's architecture, having around 5.8 million parameters. The large dataset also contributed to the long processing time. Another challenge was that the images in the dataset were in low resolution, which may have contributed to the training and testing results of the model.

REFERENCES

Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*. O'Reilly Media, Inc.

Hubel, D. H. (1959). Single Unit Activity in Striate Cortex of Unretrained Cats. *J. Physiol*, 147, 226-238.

Hubel, D., & Wiesel, T. (1959). Receptove Fields of Single Neurones in the Cat's Striate Cortex. *J. Physiol*, 148, 574-591.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE* (pp. 86(11):2278 - 2324). IEEE.

O'Shea, K., & Nash, R. (2015, December 2). *An Introduction to Convolutional Neural Networks*. Retrieved from https://arxiv.org/pdf/1511.08458.pdf

Pinecone. (n.d.). *AlexNet and ImageNet: The Birth of Deep Learning*. Retrieved from Pinecone: https://www.pinecone.io/learn/series/image-search/imagenet/