Convolutional Neural Network for Detecting Vehicles in a Parking Lot

Girgis Shihataa
Department of Computer Science
Middle Tennessee State University
Murfreesboro, United States of America
grs2w@mtmail.mtsu.edu

William Smith
Department of Computer Science
Middle Tennessee State University
Murfreesboro United States of America
wbs3d@mtmail.mtsu.edu

Michael Ketzner
Department of Computer Science
Middle Tennessee State University
Murfreesboro, United States of America
mdk4g@mtmail.mtsu.edu

Justin Hill
Department of Computer Science
Middle Tennessee State University
Murfreesboro, United States of America
jph5s@mtmail.mtsu.edu

Mubarek Mohammed
Department of Computer Science
Middle Tennessee State University
Murfreesboro, United States of America
mmm9b@mtmail.mtsu.edu

Carolous Ghobrial
Department of Computer Science
Middle Tennessee State University
Murfreesboro, United States of America
cag6p@mtmail.mtsu.edu

***Abstract*—Monitoring the number of open parking spaces with a camera is useful because it allows people to save time when commuting. We trained a CNN on the CNRPark-Patches data-set which contains 12,584 images of busy/free parking spaces. The accuracy of our model was .9753. Compared to other network architectures, a CNN was the best choice for this project.**

***Index Terms*—MTSU, Keras, RGB, convolutional, YOLO-V3, Faster R-CNN, CNN, neural network, ReLU, LeNet**

I. Introduction and Background

A. Introduction

If you are a student at MTSU, Middle Tennessee State University, and you commute, you know the struggles of finding a parking space near your class during the day. MTSU currently has a problem with parking lots for all the students who commute to campus. The number of students commuting to MTSU continues to increase every year and the problem has only increased. This always results in a large number of students funneling into and circling around parking lots at very desired locations on a daily basis. The parking lots can be completely full, but students will continue to drive around hoping they come across an empty spot. Our goal for this project was to build and train a neural network that can help provide a solution for this problem.

Throughout the length of this project, we went through multiple ideas, or iterations, of the neural network. At first, we quickly recognized that for this project, we need to build and train a CNN, convolutional neural network. The main advantage for building a CNN is that it automatically detects important features without any supervision, thus allowing for more complex networks [1]. There are many versions of convolutional networks that we researched, but eventually, we ended up with three main prospects: a basic CNN, a Faster R-CNN,or a YOLO-V3.

Through the basic CNN and the Faster R-CNN, we can achieve our neural network goal by having the neural network look at one vehicle at a time. This means that the network is binary and has the ability to observe if a vehicle is in the image or not [1]. This opens up an avenue of methods that we can achieve our goal by. Mainly, the neural network can observe vehicles going and going out of the parking lot, or it can observe the parking lot spaces separately. The former being more efficient in hardware required, and the latter being more accurate.

For the final prospect, the YOLO-V3 neural network has the ability to detect multiple vehicles in a singular picture [1]. This network allows us to have the most efficient setup for hardware, a singular camera on top of a post in any of the parking lots. The camera will take an aerial view picture of an entire lot, and the neural network will process the image by observing how many vehicles are in the parking lot. For obvious reasons, this has the least amount of accuracy, and thus will result in far more errors for our ideal goal.

B. Background

Parking at MTSU for students has been a rising issue in the recent years and it seems as if there will not be any solutions offered by the school in the near future. This neural network was built and trained in light of this problem as to provide a possible solution. Our plan for this project was to build a neural network that can detect vehicles in a parking lot. For the near future, we hoped that we could get in contact with MTSU's mobile development team in hopes of implementing this network into MTSU's mobile application which will receive input from the neural network as to which parking lots are open and which are full. As of now and the current quarantine situation, this is not possible.

This project can help better the lives of students commuting at MTSU, and potentially even other universities by lessening the time it takes to find a parking spot. From personal experience amongst our group, we had an average of 15 minutes to find a parking spot, and almost 90 percent of the time, it was no where near our destination building. By reducing the time it takes to find a parking spot and potentially even, at a closer lot to the destination building, this will free any and all future students from the stresses of searching for a parking space and improve the time it takes for them to commute. By reducing commute time, less students will be late because of the traffic in parking lots.

After many deliberations, we finally decided on going through with a basic CNN for three important reasons.

- Simpler implementation
- Less training time/epochs compared to other networks
- Accuracy is within acceptable range

As listed above, the basic CNN does everything the other networks do, as good as, if not better. It is also much easier to implement than the other aforementioned networks.

II. Methods

A. Data

We decided to use the CNRPark-Patches data-set. It contains a total of 12,584 images. The images were collected at varying weather conditions and light conditions [2]. The images are taken at different perspectives and angles which helps the network generalize. Cameras will not always be mounted in the same way. The addition of noise to the images helps improve/prepare the network for noisy pictures [3].

10% of the images were reserved for testing. The other 90% were split 80/20 for testing and validation.

B. Input Processing

We downsized the images to an input shape of 150X150 before feeding the images to the network. The image set has been divided into two batches, training and testing image data set. The training data set consists 11326 images. The testing data set consists 1258 images. We normalized the image from a scale of 0.0 to 255.0 to 0.0 to 1.0 float32 value. The normalization makes of the image data reduces the memory utilized [4]. We utilized Keras image preprocessing tools to perform slight transformations on the image that helps in the creation of a stronger and more robust model. We performed preprocessing techniques including rotation of up to 90%, re-scaling factor to reduce the RGB value to range between 0.0 to 1.0 (Fig 1).

Fig. 1. Pre-processed image of a car.

C. Convolution Neural Networks

For our neural network architecture, we built a Convolution Neural Network which have been known for providing a higher accuracy of object/image recognition compared to other neural network architectures [4]. We built our network based on further research on well known CNN architectures and a starter tutorial provided by Francois Chollet [3]. In comparison, our architectures is built to fit a smaller scale visual recognition where the solution is binary choice between parking space being empty or full.

Our architecture accepts an input image shape of 150X150 with a number of channels equal to 3. Our architecture consists of three convolution layers that are essential to extract the visual features of the image. The filters selected for the first and second convolution layers is equal to 32. We increased the filters in the third convolution layer to 64 in order to grab the image features and patterns like corners and edges [5]. We decided upon a kernel size of 3X3 in our layers which aligns with the objective of capturing the images features.The convolution layers are followed by rectified linear unit activation function (ReLU) and max pooling layers with a pool size of 2X2. We flattened the 3D feature maps to 1D feature vectors. We pass the flatten layer to a dense layer consisting of 64 neurons followed by rectified linear unit activation function (ReLU). We chose a larger number of neuron size to improve the CNN architectures robustness and the ability to extract patterns. The dense layer is followed by another rectified linear unit activation function (ReLU). With the target of reducing over-fitting of the training data-set, we introduced the dropout layer that turns off 50 percent of the neurons randomly during training process. The final dense layer consists of a single neuron which is the representation of the solution of our network. A single neuron represents the fact of the outcome of the network as a 0 or 1 where 0 signals an image of a busy parking space while 1 represents an image of a free parking space. The model is compiled with binary crossentropy loss function which lines with the binary classification problem of our architecture. Figure 2 displays the detailed structure of our CNN.

III. Results

We trained the model using twenty epochs, one-tenth of the training images for steps per epoch during training, and one-tenth of the validation images for steps per epoch during testing. Using this setup, the accuracy and loss were brought

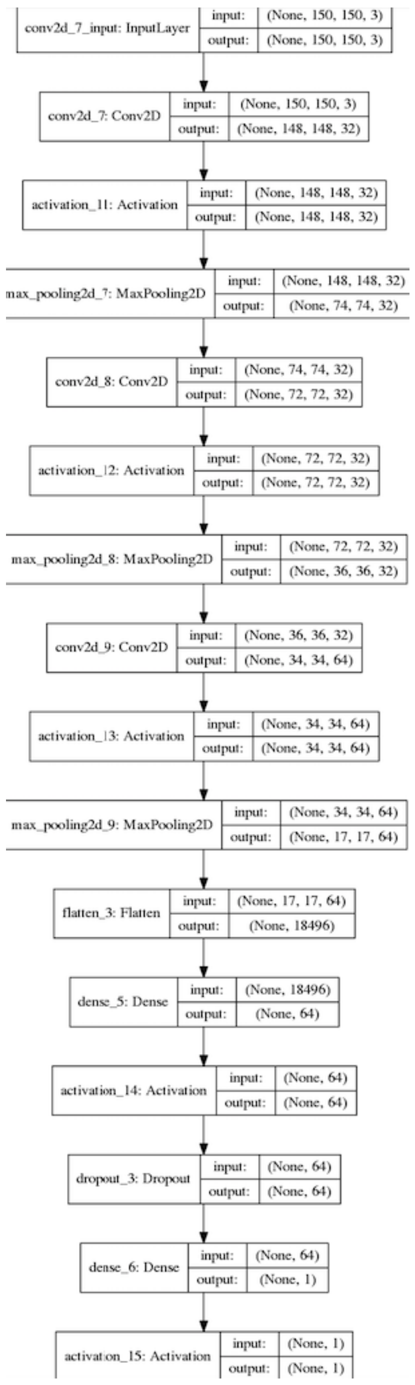
conv2d_7_input: InputLayer
input: (None, 150, 150, 3)
output: (None, 150, 150, 3)
conv2d_7: Conv2D
input: (None, 150, 150, 3)
output: (None, 148, 148, 32)
activation_11: Activation
input: (None, 148, 148, 32)
output: (None, 148, 148, 32)
max_pooling2d_7: MaxPooling2D
input: (None, 148, 148, 32)
output: (None, 74, 74, 32)
conv2d_8: Conv2D
input: (None, 74, 74, 32)
output: (None, 72, 72, 32)
activation_12: Activation
input: (None, 72, 72, 32)
output: (None, 72, 72, 32)
max_pooling2d_8: MaxPooling2D
input: (None, 72, 72, 32)
output: (None, 36, 36, 32)
conv2d_9: Conv2D
input: (None, 36, 36, 32)
output: (None, 34, 34, 64)
activation_13: Activation
input: (None, 34, 34, 64)
output: (None, 34, 34, 64)
max_pooling2d_9: MaxPooling2D
input: (None, 34, 34, 64)
output: (None, 17, 17, 64)
flatten_3: Flatten
input: (None, 17, 17, 64)
output: (None, 18496)
dense_5: Dense
input: (None, 18496)
output: (None, 64)
activation_14: Activation
input: (None, 64)
output: (None, 64)
dropout_3: Dropout
input: (None, 64)
output: (None, 64)
dense_6: Dense
input: (None, 64)
output: (None, 1)
activation_15: Activation
input: (None, 1)
output: (None, 1)

Fig. 2. Detailed structure of the CNN.

into acceptable ranges. The network brought a final validation accuracy of .9895 and a validation loss of .0556. The test accuracy was 0.9753 and the test loss was 2.667. This means after 20 epochs, our network was capable of identifying 98% of the images correctly as busy or free (car or no car present in the parking spot.)

Interestingly, the number of epochs could have been decreased greatly and gotten a similar result on accuracy. As can be seen in Figure 3, the accuracy of the network maxes out at 98% around epoch 6 or 7. After that point, the network pretty much flat lines (there are some minor deviations). Loss is a bit of a different story. After about ten epochs, the loss drops below .1. However, while the loss has reached an acceptable value, it is still decreasing as time goes on. Also, the loss does increase from about 12 to 15, and then decreases again from then on.

Based on the distribution of busy and free parking lots, the network model tended to perform better for busier parking lots (Fig. 4). For the busy parking lots, the neural network performed at 98% accuracy, while only 72% for the free parking lots.

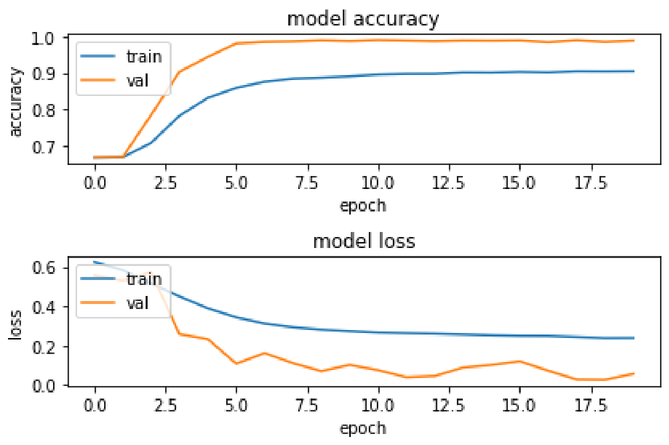

Fig. 3. Model accuracy and loss

IV. Discussion

Even though the network preforms better on busy spaces, it shouldn't affect our end goal too much. The neural network will be in use for busy parking lots during the day and it's better to error on the side of busy than free.

There were many versions of this network and many implementations, but overall, the version of a simple CNN worked the best. It gave us the best accuracy with the best potential efficiency. After the completion of the network, we were surprised to see how accurate the network was. This is one step forward to helping all students who commute, not just students at MTSU.

After researching many implementations, our final solution for the neural net was a simple CNN. While the YOLO-V3 and the Faster R-CNN do provide good results, they are more complex and require more time to train. In comparison, the

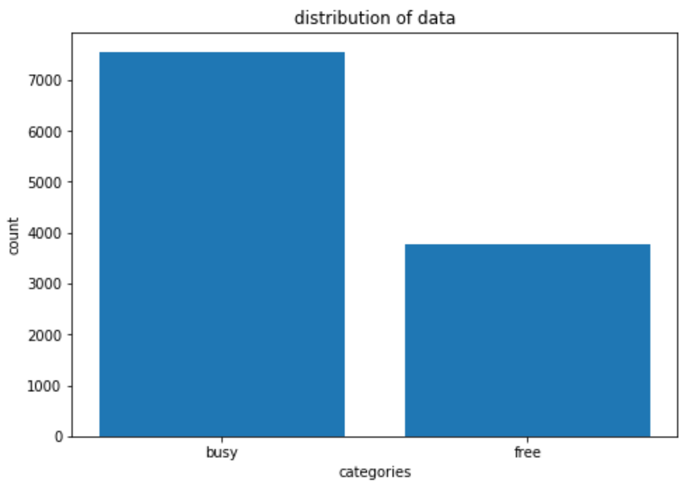

Fig. 4. Busy vs Free lots

simple CNN accomplishes the goals of our project, and at very little cost compared to other network architectures.

Future work could involve utilizing real-time cameras with a resolution degree of higher magnitude which improves the image quality that is going be feed into the CNN. In order to implement the network to solve the growing parking lot congestion issue, a more robust method needs to be devised that incorporates the accuracy of the R-CNN and faster recognition property of YOLO-V3.

Further effort could be put in designing a User Interface that displays the number of spaces available based on a real-time data processed and updated. The User Interface could be included in the already functioning MTSU's mobile app.

References

[1] M. A. Adel Ammar, Anis Koubaa and A. Saad, "Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-cnn and yolov3," *arXiv, Cornell University*, vol. 1, 2019.

[2] F. F. C. G. Giuseppe Amato, Fabio Carrara and C. Vairo, "Car parking occupancy detection using smart camera networks and deep learning," *IEEE Xplore Digital Library*, 2016.

[3] F. Chollet, "Building powerful image classification models using very little data," *The Keras Blog*, 2016.

[4] I. S. Alex Krizhevsky and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *NIPS Proceedings*, 2012.

[5] Y. B. Yann LeCun, Leon Bottou and P. Haffner, "Gradientbased learning applied to document recognition," *Stanford Vision*, 1998.