FAA Binary Digit Recognition with CNNs

Mitchell Kraft, Eric Kryzanekas, Alexander Wiese, Cailyn Rhoads, Helen Pan

kraftm4@students.rowan.edu, kryzaneke0@students.rowan.edu, wiesea4@students.rowan.edu, rhoadsc2@students.rowan.edu, panh1@students.rowan.edu

May 10, 2019

Abstract **—The purpose of this project was to design a Deep Convolutional Neural Network (CNN) in order to recognize the digits on dials in an aircraft cockpit from a video stream. This was done by implementing object detection algorithms within the network. Thousands of images of three-digit numbers were parsed into three separate images, and then manually labelled by members of the team. These labels were then used as input in the network to train on. The network's test accuracy was then obtained and recorded. A relationship between certain changes in the network's parameters and the test accuracy is also considered and analyzed. Future improvements are also noted.**

I. Introduction

Convolutional Neural Network (CNN) was used to find possible solutions to this project. Algorithms such as regional-CNN, single shot detector (SSD) and YOLO (you only look once) are all common approaches for object detection. R-CNN takes an image and splits it into many regions. Each region is classified to have a certain object in it and then the regions are reconstructed. In SSD, the object is found and classified in a single pass of the network; SSD aims to detect objects during this single pass. Digit Recognition for this project was aimed towards identifying the digits from digital gauges in aircraft cockpits. Additionally, this project has more applications than simply identifying digits in aviation equipment and aircraft. For instance, cars and other vehicles also contain digits that monitor the speed of the car or the mileages on the car, this could prove useful if a truck driver needed to monitor speed and distance of a trip. Instead of logging everything, a camera could be set up to monitor the dashboard, then stills could be taken and the digits could be recognized. This would allow for a more accurate logging of trips and this same approach can be utilized by many other vehicles such as planes, rockets, or trains.

The Federal Aviation Administration (FAA) has sponsored this project in order to use advances in Deep Learning technologies to aid in a multitude of aviation operations. The main obstacle for this project is the difficulty in creating a network that can accurately identify a number that includes more than one digit. First, the data has been provided to the team in the form of thousands of different images of digits. The network will then be trained on this data. Below is a list of objectives for this project. Objectives for this project consisted of firstly downloading a sufficient number of images to train and test the model and establishing a method for data segmentation. 4,000 images were then sectioned and labeled to produce 12,000 total digit labels. This data was then used in the production of a network to train digit recognition. Finally, the effects of changes to the network's hyperparameters were tested and analyzed.

II. Methods and results

A. Data Segmentation

The most important part of any convolutional neural network is the input, or labeled data. For this project we were provided with a data set that contained roughly one-hundred thousand images. These images are taken from a live video feed from inside the aircraft and display binary digits from different flight gauges. The provided images either represent the longitude or latitude values and are 25x25 pixel images. *Figure 1* demonstrates one of the images that will be used to train the model.

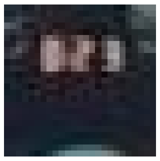


Figure 1: 25x25 pixel image of the Longitude Gauge

Figure 1 demonstrates not only the binary digits that will be recognized by the model, but also just how small these images are. Just increasing the size of the image by a small amount leads to a very blurry picture that makes the digits hard to read.

The next challenge for segmenting data is the fact that the model that we will train will not be able to identify the three binary numbers that are located in one single image. In order to tackle this problem we decided to collaborate with other groups in order to develop a Python script that would allow us to manually label the data.

This Python script applies a mask to the image to cut a 15x7 pixel segment the contains the three individual binary digits. Next the 15x7 pixel image is cut again into three separate 5x7 pixel images, where each one represent a single binary digit. *Figure 2* demonstrates the initial plot from the Python script that shows the user what specific image they will be segmenting is.

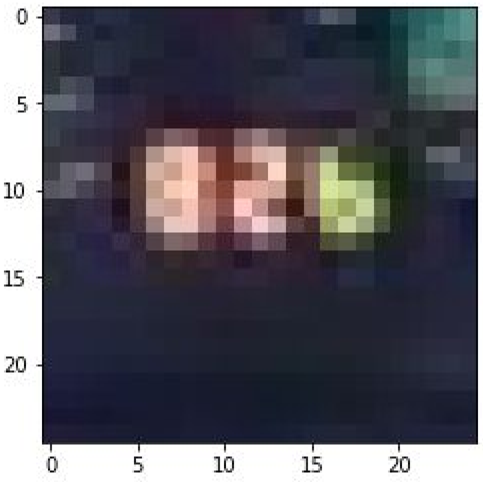


Figure 2: First plt.show from the Python script

Figure 2 then gets turned into the following figures, which allow for the team members to label the individual binary digits.

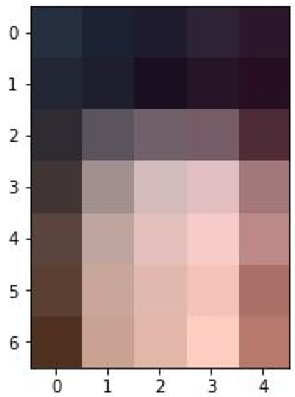


Figure 3: plt.show of the first Binary Digit

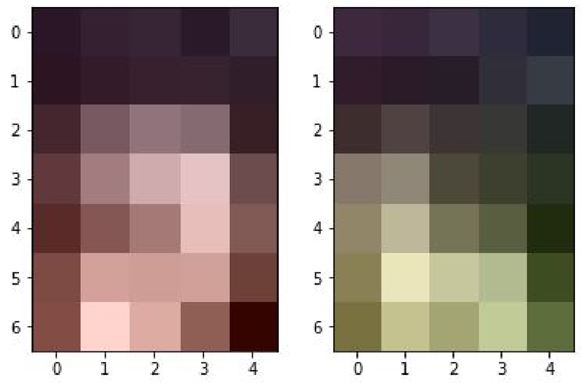


Figure 4: plt.show of the second and third binary digits

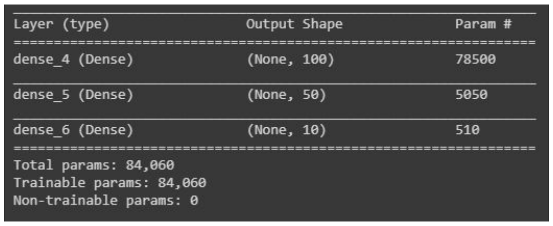
These image plots demonstrate the challenge that is associated with labeling the data for this project. You can clearly identify from *Figure 2* that the number appears to be 826; however, *Figure 3* and *Figure 4* demonstrate how much harder it is to discern between the individual bits. Inconsistencies rise from illumination behind the digits as well as slight shifts in the numbers. This system does not account for large shifts meaning that some images did not cut into 5x7 pixel images correctly and had to be omitted from the training.

Each team member was tasked with completing this process for 1,000 images. This yielded a data set that contained the segmented images from 4,000 unique gauges. The final dataset contained 12,000 actual 5x7 pixel images because each unique gauge produced 3 separate images. The output images have the tag "33960crop_3." This first number before crop relates to the actual gauge image from the FAA data set, and the crop_3 relates to the position of the image. For instance, "33960crop_3" would refer to the final image in *Figure 4*.

This data segmentation will allow for the training of the Convolutional Neural Network Model. Section *B* of this report will go into detail about how the model was actually trained.

B. Training of the Model

The best trainable model for image identification is with a Convolutional Neural Network. The model that was used to train this data set was modeled after one that trained the MNIST data set. The MNIST database is simply hand drawn digits, which makes sense to do transfer learning because the model is already trained to handle digits. Our model was trained by using Google Collab and their cloud service graphics cards, this allowed for easy collaboration on the same piece of code amongst team members. Each team member iterated on the regularization in order to determine the highest model accuracy. The following model summary discusses the layer types and output shapes of the model.



```
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 100)               78500
_________________________________________________________________
dense_5 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_6 (Dense)              (None, 10)                510
=================================================================
Total params: 84,060
Trainable params: 84,060
Non-trainable params: 0
```

Figure 5: Model Summary

This model contains three separate dense layers. These dense layers have decaying output sizes, going from 100 to 50 and finally to 10. This decreases the number of trainable parameters per stage, and finally yields 84,060 parameters.

This model was ran for 100 epochs, or every piece of data was seen 100 different times in order to reach the final model accuracy. A batch size of 1024 was also chosen because it yielded the highest final accuracy while also not restricting the usable RAM of our system. The first two layers used relu activation and the final layer used softmax activation. The final model was compiled using categorical cross entropy and an Adam optimizer. The final model accuracy and model loss can be seen in the following figures.

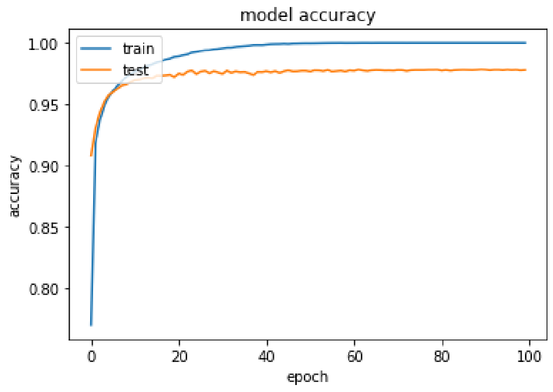


Figure 6: Model Accuracy over epochs

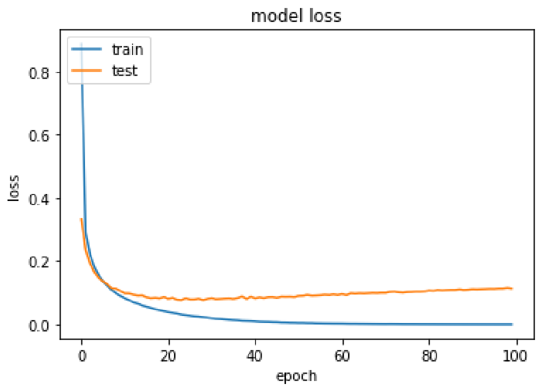


Figure 7: Model Loss over epochs

Figure 6 and *Figure 7* demonstrate the model learning over time . The final accuracy of the model was roughly 97.8% and the final loss of the model was around 11.34%. Although final accuracy is very high and the loss is very low, the model was not performing well in the testing area.

In order to test the model cv2, a built in Python library, was required to be imported. This library allows us to read in the individual binary digit images and reshape them so that they match the input image size of our network. This allows us to individually test images that the network has not seen yet. Unfortunately because of the size of the reshaping of images I do not believe that the network will be able to successfully determine what digit is being represented.

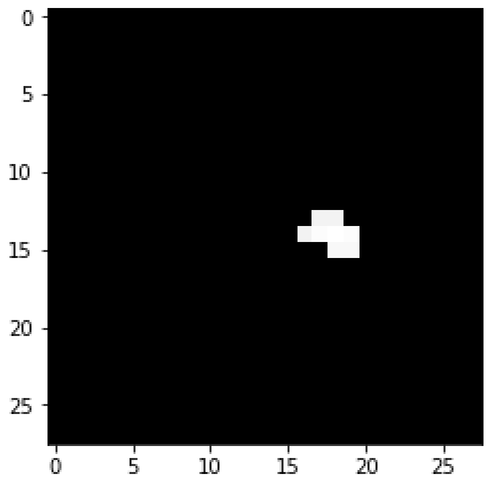


Figure 8: Pixel output of Binary Digit

Figure 8 is the output of the image that the model saw to test. You can see that the only space that contains any real information is very small and roughly 5x5 pixels in dimension. The model predicted that this image was a four, when in fact is was actually a seven.

Figure 9: Binary Digit that was tested by the model

Figure 9 is the input to the cv2 function that was used to test the model. You can see that the pixels that represent the number are actually correctly mapped to the output in *Figure 8*; however, the image is so small that it is not able to be successfully determined by the mode.

III. Conclusion

The project demonstrates the power of convolutional neural networks, and how good segmented data leads to good test accuracies. It is very clear from *Figure 6* and *Figure 7* that training a CNN can lead to incredibly high accuracies when the input data is labeled correctly and large enough. It was determined that trained networks, such as MSNIST, might not be generalized enough to work with all forms of number recognition. This was determined because the network that was used was unable to successfully determine the individual digits, and this was determined to be a result of the overall small size of the individual digits.

There are plenty of next steps that could allow this model to be more accurate as well as fix mistakes. The most noticeable change that can be made would be the addition of an actual Python script that would perform a similar action as the data segmentation step. This would allow the model to actually take the full gauge input, segment the digits, identify each digits, then output the values of all three digits back to the user. This would allow for more functionality of the network than simply identifying the individual digits.

The Git repository that contains all of the Python code, data, and .csv files can be found at: https://github.com/MitchellEK/FAA-Machine-Learning-Project This repository also contains a copy of this report.

IV. Acknowledgements

The project is sponsored by the federal aviation administration (FAA), who we thank for providing us with the source data. This allowed us to develop a network to solve real world problems. We would also like to acknowledge Rowan University for allowing us to take part in this project.