Automatic Mail Delivery Detection

An Artifical Neural Networks Design Report

Sarah Brown
College of Engineering
University of Oklahoma
Norman, OK

***Abstract* — A neural network project that features the use of computer vision to create a dataset for a U-Net convolutional neural network model. By creating a dataset from archived security footage, a model is created to predict the delivery of mail. This model can be used to create an alert for when the mail is about to arrive and for when it is delivered.**

Keywords—U-Net, Image Recognition, Neural Networks

I. Introduction

A. Description of Problem

Mail can often be delivered at unpredictable times, which can disrupt habits and lead to mail being forgotten. Forgotten mail can result in issues with bills or missing other important information. In addition, with social distancing over the past year, avoiding contact with postal workers can provide peace of mind and offers essential workers additional safety.

B. Problem Solution

By creating a model that detects when the post truck is stopped at the mailbox, an alert can be sent to notify inhabitants of mail delivery. Archived footage from home security cameras with a view of the mailbox was used to create a dataset to train this model. The original approach to this project included the creation of the alert system in addition to the model. However, this is something that will be appended later. Future additions to this project will take the finished model and incorporate it into live camera footage to provide notifications of mail delivery.

II. Dataset

The raw dataset was created by extracting archived footage and sorting photos into negative and positive directories. This data was then split into three subcategories for training, validation, and testing. This data was then processed with two functions. One function was created to process individual images and the other was created to process the different folders. The images were cropped to a 400x400 pixel square region surrounding the mailbox and then was resized to be a 64x64 pixel square. After the images were processed, they were saved to a new directory to be read into the model. Some adjustments had to be made to the dataset throughout the process. The main adjustment was to remove some data due to the camera having changed angles partway through the saved images. This angle difference resulted in a large change in the mailbox position in the camera frame and also changed the point of view of the mail truck. As the old angle was not relevant to the current problem, the images were removed from the dataset. Fortunately, this did not make the size of the dataset too much smaller. In addition, as time goes on more data can be collected from the new camera angle. Another created function is used to display the different sections of the dataset for ease of use. This allows for quick glances to identify abnormalities in the dividing of the dataset. [2][4]

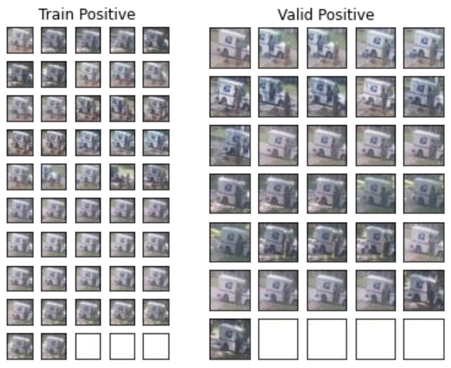

Figure 1. Positive images containing a mail truck that were used for training and validation purposes.

III. Data Manipulation

Once the processed image directories have been created, Keras' ImageDataGenerator class is used to load the dataset and generate batches of data. The datasets are made using the flow_from_directory function to quickly import data. The training dataset uses the shuffle feature, while the other two do not. Shuffling the training set helps to decrease overfitting. The ImageDataGenerator class also has a rotation option that automatically rotates images within a certain degree range. This feature was included for testing while the dataset contained images from the other camera angle. However, this introduced some errors of the mail truck being identified as at the mailbox when it was actually going back down the street in the opposite direction. Therefore, when the other images were removed from the dataset the rotation aspects were removed from the image augmentation process. [1][5]

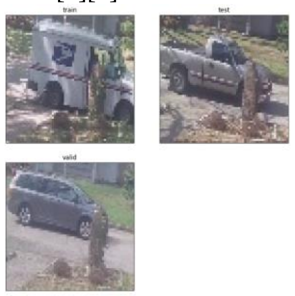

Figure 2. Individual images shown from the train, test, and validation datasets.

IV. Model

The model for this project is implemented using a U-Net. This model expands on the U-Net like predictor that was used for the stock market data in homework 4. The model takes in the cropped and resized image as input and performs a series of operations to predict whether the mail truck is stopped at the mailbox. The contracting path starts at 64x64x3 and ends at 16x16x64. The data is then upsampled on the expansive path. A flatten and a dense layer are then added to condense the result to an output of shape 1. After the model is constructed with use of the Tensorflow package, it is trained using the constructed datasets. Validation data is used to avoid overfitting. The model was recreated with binary accuracy with very similar results.

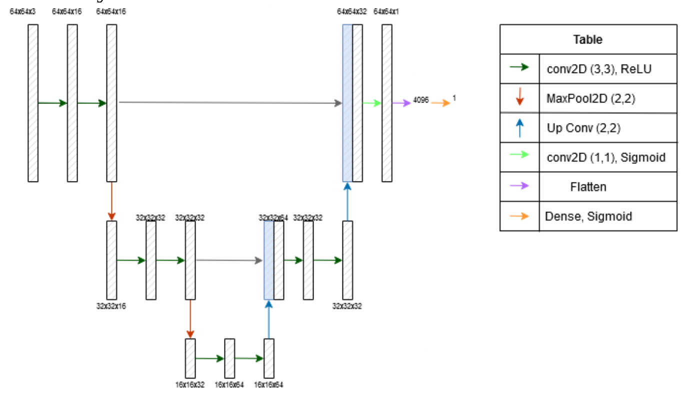

Table	
→	conv2D (3,3), ReLU
↓	MaxPool2D (2,2)
↑	Up Conv (2,2)
→	conv2D (1,1), Sigmoid
→	Flatten
→	Dense, Sigmoid

Figure 3. Graphic showing U-Net model.

V. Model Results

The model is then tested with use of the test dataset and the labels are compared against the file names to determine accuracy [3]. The training results ended with a 99% accuracy with a loss of 0.0173. The accuracy on the test data was measured to be 98.8% or 2 photos being misclassified out of 164. However, these two incorrectly labeled photos may be in part due to the mail truck appeared earlier in the frame than other positive images. Since the mail truck pauses at the mailbox to deliver mail, this will provide multiple frames to sample from and should be very consistent with this level of accuracy.

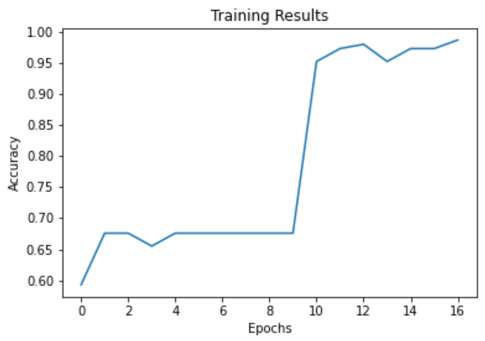

Figure 4: Training results from the accuracy-based U-Net model.

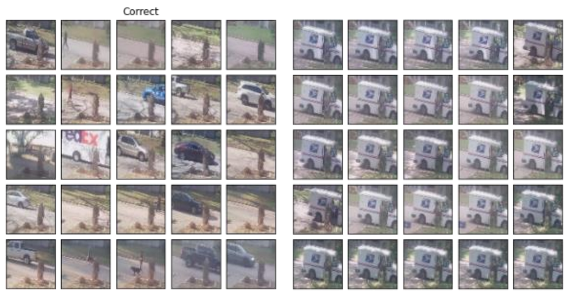

Figure 5: Some of the correctly identified photos.

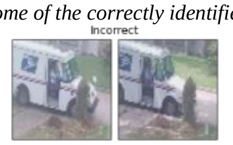

Figure 6: The two incorrectly identified photos.

VI. Testing Model on Video

This model was testing on various recorded videos containing negative and positive results. By reading in the video with the use of OpenCV, the saved model was then applied to each frame. The probability that the given frame is positive is then shown in the upper left of the outputted image. A threshold is applied to this probability to classify the incoming data. If the probability is greater than 0.5, it is a positive result. Otherwise, it is a negative and the mail truck is not stopped in front of the mailbox.

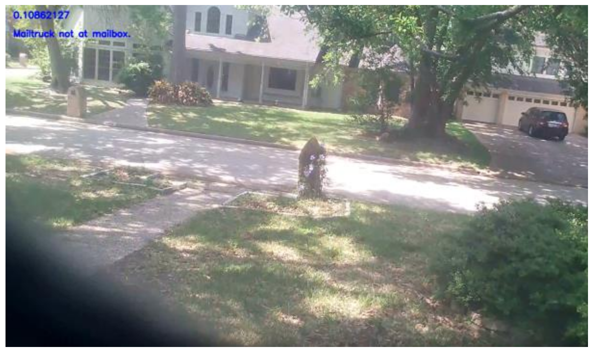

Figure 7: Test result showing that the mail truck is not at the mailbox.

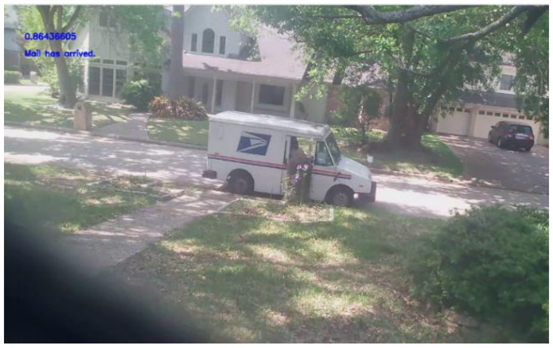

Figure 8: Test result showing that the mail has arrived.

VII. Additional Camera Results

In addition to being able to tell when the mail has arrived, it is also important to be able to tell when the mail is coming. Having an alert that notifies users of this event is beneficial as it creates a chance to place mail in the mailbox before the truck arrives. A different model was created by repeating the same process, but with a different camera view. However, this camera view did not have as many positive results archived and therefore is not optimized to different lighting conditions yet. Out of 83 test images, 1 was incorrectly labeled as negative. This model also has a very high training accuracy and shows signs of having a high testing accuracy. Validation was skipped while creating this model due to the smaller amount of data available. As more data is collected in the future, validation will be added to help decrease risks of overfitting as well. More data will also help with different lighting conditions as the current model only works well with minimal shade.

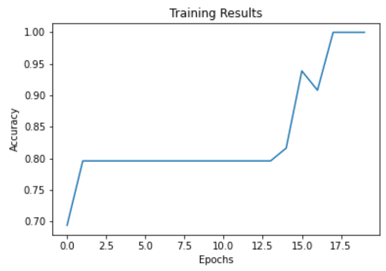

Figure 9: Training results from the other camera view model.

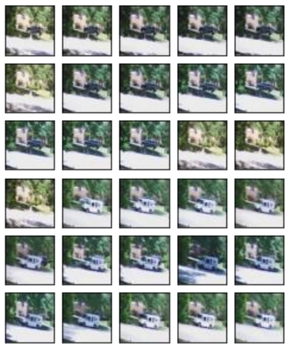

Figure 10: Correctly identified test pictures.

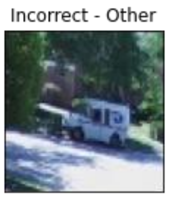

Figure 11: Incorrectly identified test picture (lots of shade in photo).

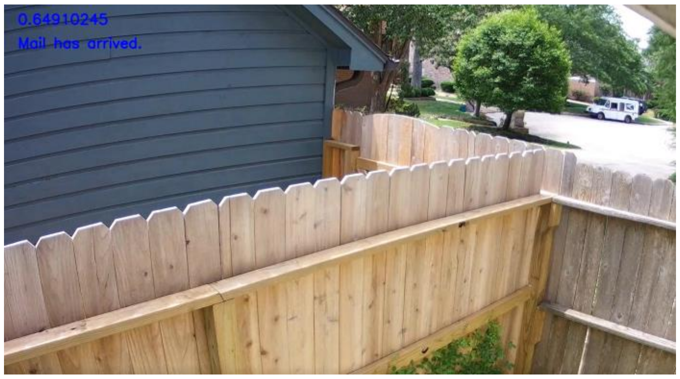

Figure 12: Model test on video showing that the mail has arrived.

VIII. Conclusions

A. Challenges

There were several especially challenging aspects while tackling this project. Some of these challenges came from manipulating the dataset to be cropped to the correct portion of the frame. In addition, when using the flow_from_directory function from Keras' ImageDataGenerator, it was unclear that this defaulted to 256x256 pixels as it output which caused issues with the input dimensions into the model. There were also other issues in getting dimensions to match all of the way through the created model. In addition, there were some issues related to obtaining a negative probability in the results until a Sigmoid activation was added to the last dense layer of the model. This Sigmoid activation limited the probability value to be between 0 and 1, solving the issue.

B. Further Work

This project can be extended by adding this model to the camera system to monitor for the delivery of mail. As the camera system is on a separate network within the house, it will have to be implemented in a way to accommodate being hosted on a different network. By monitoring the video frames during the afternoon hours, the model will be able to detect if the mail truck stops at the house. In addition, by measuring how long the truck is stopped in front of the mailbox, false positives can be avoided on days that the truck just drives by without delivering mail. This system would then generate an alert depending on whether or not the mail was delivered and notify the inhabitants.

Code

Code for this project can be viewed at: https://github.com/SarahBrown/ECE5973-ANN/tree/main/Final%20Project

References

[1] B. Kanani, “Keras ImageDataGenerator with flow_from_directory(),” *Machine Learning Tutorials*, 11-Oct-2019. [Online]. Available: https://studymachinelearning.com/keras-imagedatagenerator-with-flow_from_directory/. [Accessed: 13-May-2021].

[2] “Long explanation of using plt subplots to create small multiples,” *Jonathan Soma makes things*. [Online]. Available: http://jonathansoma.com/lede/data-studio/classes/small-multiples/long-explanation-of-using-plt-subplots-to-create-small-multiples/. [Accessed: 13-May-2021].

[3] “Project Idea: Cat vs Dog Image Classifier using CNN implemented using Keras,” *GeeksforGeeks*, 09-Aug-2018. [Online]. Available: https://www.geeksforgeeks.org/project-idea-cat-vs-dog-image-classifier-using-cnn-implemented-using-keras/. [Accessed: 13-May-2021].

[4] “Simple ImageGrid,” *Simple ImageGrid - Matplotlib 3.4.2 documentation*. [Online]. Available: https://matplotlib.org/stable/gallery/axes_grid1/simple_axesgrid.html. [Accessed: 13-May-2021].

[5] V. J, “Tutorial on using Keras flow_from_directory and generators,” *Medium*, 02-Dec-2019. [Online]. Available: https://vijayabhaskar96.medium.com/tutorial-image-classification-with-keras-flow-from-directory-and-generators-95f75ebe5720. [Accessed: 13-May-2021].