5-1-2024

Street View House Numbers Classifier

Connor Paul Lough
University of Alabama in Huntsville

Street View House Numbers Classifier

by

Conner Paul Lough

An Honors Capstone

submitted in partial fulfillment of the requirements

for the Honors Diploma

to

The Honors College

of

The University of Alabama in Huntsville

5/1/2024

Honors Capstone Project Director: Dr. Chaity Mukherjee

[illegible] 5/2/2024

Student (signature) Date

Dr. Chaity Banerjee Mukherjee

Digitally signed by Dr. Chaity Banerjee Mukherjee
DN: OU=Computer Science, O=UAH, CN =Dr. Chaity Banerjee Mukherjee, E= chaity.banerjee.mukherjee@uah.edu
Reason: I have reviewed this document
Location: Huntsville AL
Date: 2024.05.02 01:08:49-05'00'
Foxit PDF Editor Version: 12.1.1

Project Director (signature) Date

Letha Etzkorn

Digitally signed by Letha Etzkorn
Date: 2024.05.02 09:52:52 -05'00'

Department Chair (signature) Date

Sean Lane

Digitally signed by Sean M Lane
Date: 2024.05.02 16:43:37 -05'00'

Honors College Dean (signature) Date

HONORS COLLEGE
THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Honors College

Frank Franz Hall

+1 (256) 824-6450 (voice)

+1 (256) 824-7339 (fax)

honors@uah.edu

Honors Thesis Copyright Permission

This form must be signed by the student and submitted with the final manuscript.

Conner Lough
Student Name (printed)

[illegible]
Student Signature

5/2/2024
Date

Table of Contents

Abstract

The mission for this project is to create an effective classifier to determine the house address numbers from images taken by Google's Street View camera. By utilizing a combination of machine learning through neural networks and creative data manipulation and processing, this classifier is able to read visual image data and produce number information that can be easily managed. Recognition and classification of visual data has always been an important feature of AI and machine learning, and it is our hope that this classifier is able to stretch that field by introducing a lightweight yet accurate model. The goal of this project was to create a classifier that was faster and more lightweight than other preexisting optical recognition models (such as VGG-16) while still maintaining an extremely high level of accuracy. To accomplish this we went through a few prototypes and stages of neural network design whilst continuing to analyze both the data we were putting into and getting out of the model. By doing this we were successful in creating the classifier we first sought; one both lightweight and accurate.

Datasets

For this project we utilized the two separate datasets found from the Street View House Number website cited in the appendix. These datasets contained images of house (address) numbers of various sizes and lengths taken from a Google Street View camera. These samples were labeled with classes corresponding to the correct number(s) (i.e. the number 1 was class '1', 2 was '2', but 0 was '10'). Each dataset contains 73257 image samples in the training set and 26032 image samples for the test set; additionally, there are an extra 531131"less difficult" samples for training use, but we did not use these.

One dataset we used for this project was the "Full Numbers" set. This dataset contained the uncropped versions of the samples found in the "Cropped Digits" dataset. Unlike the previous dataset, the image sizes for each sample was neither fixed nor square. In some cases, the full uncropped image size was less than 32x32, which posed some additional complications. Another feature of this dataset was that it provided boxes around individual numbers so that you could easily find the location of each individual digit in the image. This helped a lot in our classification since the pixel sizes of the numbers varied between samples, making it hard to capture the entire number in a window consistently.

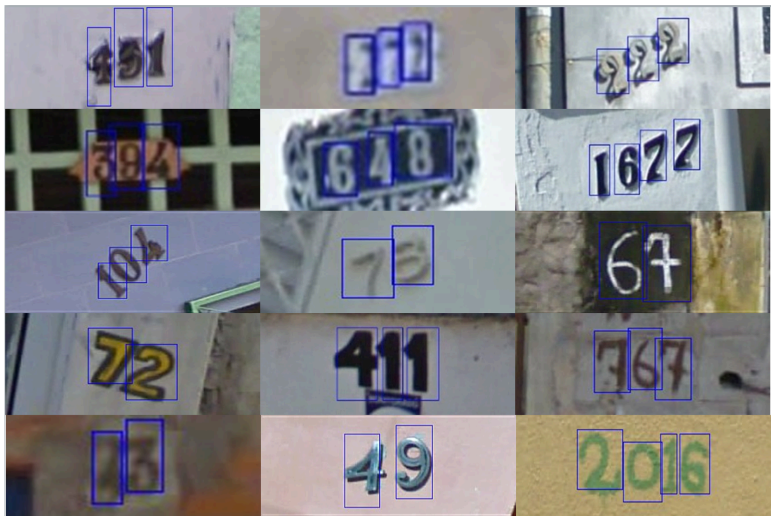


FIGURE 1: Shows examples of samples from the "Full Numbers" dataset with bounding boxes included

The primary dataset we trained and tested on was the "Cropped Digits" set. This set contains the cropped versions of the "Full Numbers" dataset, where each sample is a 32x32 pixel image centered on one of the numbers shown in the full image. These numbers were picked out from the blue bounding boxes and resized and cropped to fit a 32x32 space. An example of the cropped and full image can be seen below in Figure 2.

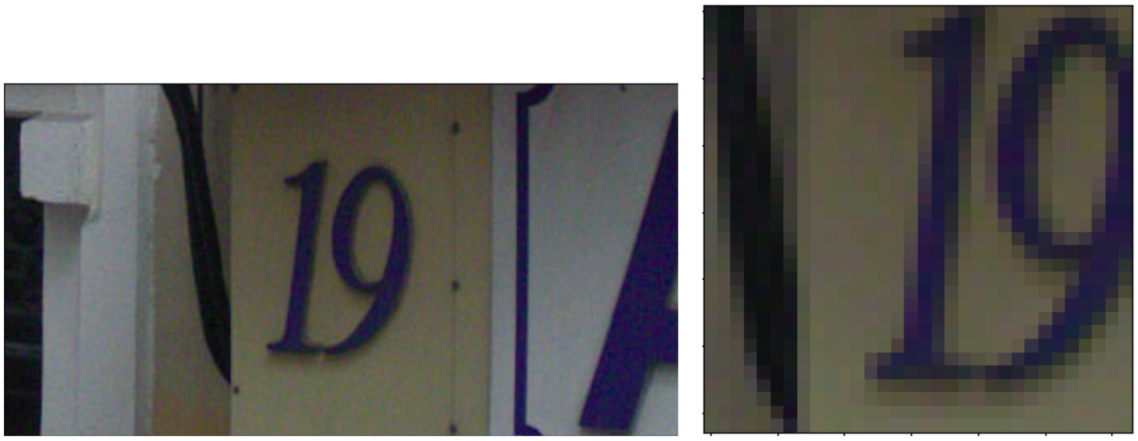


FIGURE 2: Shows example from “Full Numbers” dataset and one of its associated samples from the “Cropped Digits” dataset

Data_Preprocessing

Initially, the only data preprocessing planned was to convert the sample images from RGB into grayscale in hopes of reducing the training time. However, after experimenting with the actual training for the neural network, we decided to keep the RGB values since the training time was not decreased too much by grayscale the images and there was a decent increase in the ceiling for RGB classification accuracy. The data preprocessing we did implement was applied solely to the “Full Numbers” dataset. As mentioned previously, not all these images were bigger than the 32x32 cropped images that the model was initially trained on. To fix this, we expanded the bounding box so that its height and width were even (removing potential distortions when resizing), and then resized the image to a flat 32x32. The scaling algorithm we employed for this task was Lánczos interpolation, since it did a good job of increasing image size without too much distortion or blurriness.

Architecture

In correspondence to the two separate yet linked datasets we used, the classifier model's architecture can be split into two stages.

Base Neural Network

The first stage of the model consists of a neural network–which we refer to as the Base Neural Network–focused on taking in the cropped 32x32 images as input and outputting a guess for the digit shown in the image. To accomplish this, we constructed the neural network around, what we will call, primary computation units. Each primary computation unit contains a sequence of 2 convolutional layers, a max pool layer, and a dropout layer. In total, the Base Neural Network has three of these primary computation units; subsequently, each concurrent computation unit contains slightly different hyper-parameter values from the others, with both the number of output channels and the dropout percentage steadily increasing and the stride length for the max pools going from 2 in the first two computation units to 1 in the last.

In addition to the layers of the neural network accounted for with the primary computation units, we also included a padding layer to add a two-wide border of zeros around the input image, and we added two fully connected linear layers at the very end of the network. Another important note about the Base Neural Network is that we used ReLU exclusively as the activation function between applicable layers. Lastly, the final result was plugged into a logarithmic-softmax function to get probabilities for each class type.

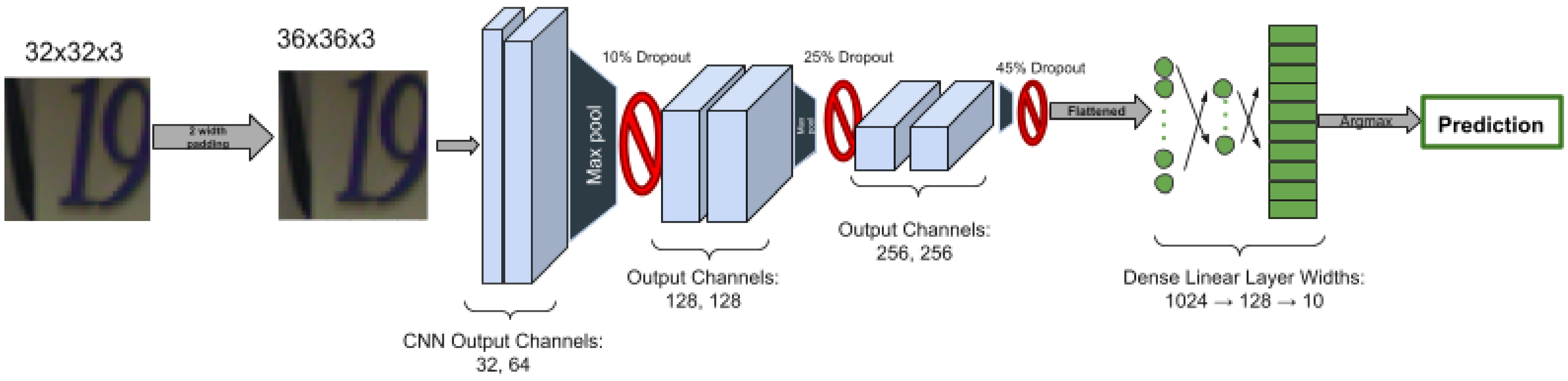

32x32x3
2 width padding
36x36x3
Max pool
10% Dropout
25% Dropout
45% Dropout
Flattened
Argmax
Prediction
CNN Output Channels: 32, 64
Output Channels: 128, 128
Output Channels: 256, 256
Dense Linear Layer Widths: 1024 → 128 → 10

FIGURE 3: Base Neural Network Architecture

Full SVHN Classifier

The second stage of the model is responsible for classifying the full, uncropped, images from the SVHN "Full Numbers" dataset. For this stage, instead of creating a new neural network, we opted to let the Base Neural Network from stage 1 do the majority of the heavy lifting. In order to come up with the classifications for the "Full Numbers" samples, we grabbed each digit specified by that sample's bounding box list, as well as all the windows immediately adjacent to the primary bounding box. Then, after using the Base Neural Network to predict a value for each of the 9 sub-samples, the most common prediction was chosen as the class prediction for that digit. This process was repeated for each digit found in the “Full Numbers” sample.

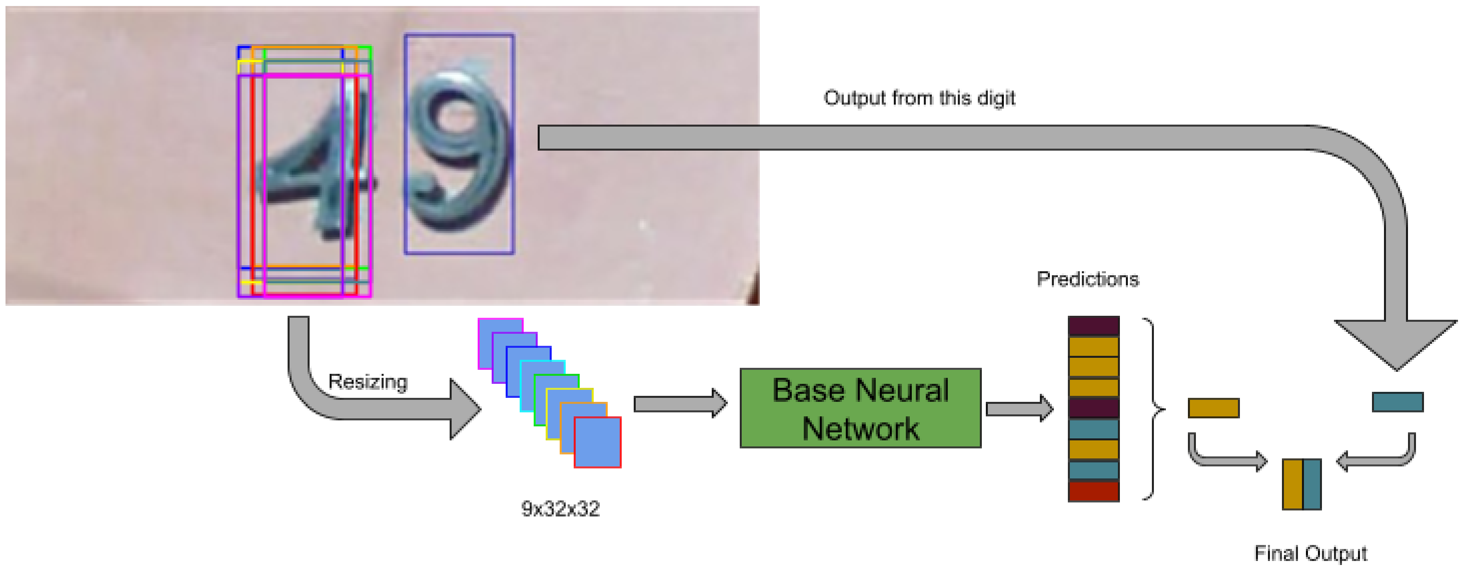
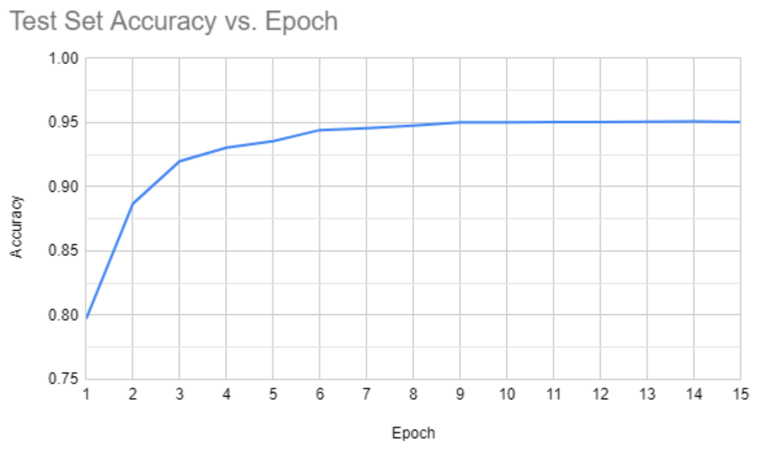


FIGURE 4: Full SVHN Classifier Architecture

Results

Base Neural Network

Since the Base Neural Network is set to be the backbone of the entire classifier, it was important to create a model that produced results that could meet or exceed expectations. Therefore, after deciding on a final architecture for the neural network and continuously tweaking its hyperparameters, we were able to produce a model that can determine a digit from an image with just over 95% accuracy. Figures 4 and 5 give a visual representation of the model's test set accuracy and test set loss with respect to the number of sweeps through the training dataset (epochs). By looking at those figures you can see that the accuracy and loss evens out at about epoch 10; though in actual training I would stop the learning process at that point, I decided to go to epoch 15 to accentuate the plateau effect for these graphs.



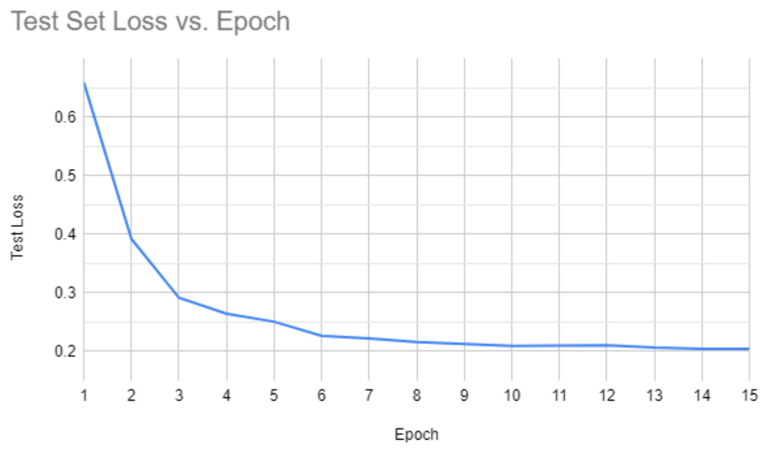


FIGURE 5 (top) **& FIGURE 6** (bottom)**:** Shows the test set accuracy and loss, respectively, for the final Base Neural Network model across 15 passes through the training data set

Full SVHN Classifier

Since the Full SVHN Classifier was exclusively using the Base Neural Network to make predictions, the test set accuracy on the “Full Numbers” dataset was guaranteed to be lower than the 95% accuracy achieved from the “Cropped Images” dataset. The “Full Numbers” dataset contained images that averaged about 2 digits per sample, meaning that our 95% accuracy would only translate to a theoretical maximum of less than ~90% accuracy (0.95*0.95). With this being said, our implementation of the Full SVHN Classifier achieved 81.9% accuracy on the “Full Numbers” dataset, which is 91% of the maximum I mentioned before. This difference can probably be attributed to an issue with the image/digit cropping and resizing algorithm causing the digits to be harder to recognize than in the cropped dataset provided. Additionally, we found the average classification time per sample to be ~0.04 seconds per sample classification.

	Accuracy	Time per Sample
Bicubic	0.81554	0.0428
Bilinear	0.8173	0.0428
Lanczos	0.8189	0.0432
Hamming	0.814	0.0454

FIGURE 7: Shows the accuracy and average sample classification time for different image scaling methods

Conclusion

Overall, this project was pretty successful, especially with our stage 1 prediction model. Our Base Neural Network model was able to make 95% accurate predictions with an architecture less than half the size of other major models like VGG-16. Though we had hoped for slightly better accuracy on stage two of the classification, the potential ceiling for the full classification seems within reach with just implementing an improved image resizing algorithm. Our goal for this project was to create a lightweight and accurate prediction model for Street-View house numbers, and I think as a whole this was accomplished.

Appendix

Citation for Dataset

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng Reading Digits in Natural Images with Unsupervised Feature Learning *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Citation for Dataset Loading Library:

Skjerns, mat7.3, (2024). GitHub repository. https://github.com/skjerns/mat7.3/tree/master

Acknowledgment

Special thanks to my partner Mishal Mansur for helping me to complete this project.