Machine Learning Engineer Nanodegree

Capstone Project

Edward Storey
September 12nd 2019

Proposal

The following sections will give an overview of my full Capstone Project for the Machine learning Engineer Nanodegree.

I. Definition

Project Overview

This project investigates improved machine learning methods for recognising digits in real world google map photos of address numbers. There are many previous examples of digit detection, most notabley in handwriting detection [1] [2] [3]. This particular project concentrates on the case of address numbers of real world buildings. This dataset The Street View House Numbers (SVHN) Dataset [4] takes images of building numbers from around the world and separates them into both separate digit images and the full multi-digit number images.

The recognition of real world address numbers is particularly useful in the application of Google Maps and other map software. As is the case with other machine learning applications there is simply too much data available, vast numbers of address numbers are shown in Google maps street view images. Commonly on long streets address numbers can vary in distance by a large margin from where they actually exist. This can be frustrating for users who can find themselves far from their intended destination. One possible solution to issue like this is using the images as a marker for each building address, this approach will be explored over the course of this project.

Problem Statement

The problem addressed in this project is identifying street addresses based on google map street view images taken from a Stanford made dataset [4]. The approach will be to identify numbers in real life images taken from google map images from around the world. The data set contains thousands of images of address numbers on doors and signs in variety of lightings, angles and colours. The aim of the project will be to identify single digits in these images and then convert those to whole address numbers. Digit image recognition is already a well researched area of machine learning, with digit recognition in “real life images” a sub set of this research. Commonly real life recognition is largely in the form of handwriting recognition, in this case I will be researching the less covered form of real life images from the open world. I will be taking a dataset of 33402 .png images taken from real world address on Google street view images and contains 73257 separate digits within the larger images.

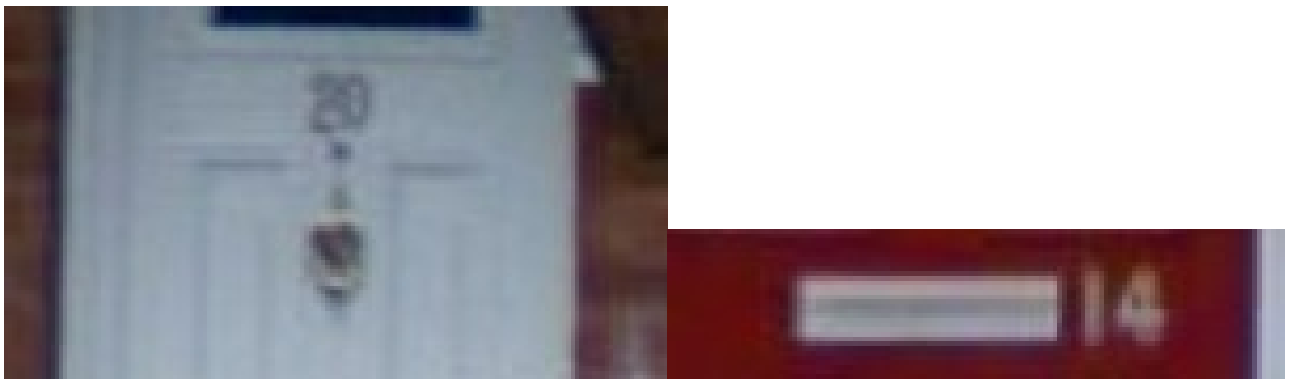


Figure 1: image 70.png (left), image 90.png (right) showing the diversity of size of the images, size of the numbers and the colour variations

The main hurdles in this problem come from the quality and variety of the images themselves, as shown in Figure 1. Not only do the images vary greatly in size but some images are clearer than others. Certain images are taken from different angles and distances and they also vary greatly in size and resolution. Another hurdle for the project to overcome is that the accuracy of the algorithm has to be at or almost at 100% whilst not succumbing to over fitting. If even one digit in an image is classified incorrectly the address is essentially useless.

I have taken inspiration from human behaviour when reading multi-digit numbers day to day life for this project. I theorise that when a person recognises a number they first identify each of the individual digits and then compile them into the full number. For example the number 153 would be first seen as 1, 5 and 3. Once the viewer had identified each digit individually they would then compile the number and recognise it as one hundred and fifty three. I aim to emulate this approach to full number recognition with my algorithm by having 2 separate learning models recognising different aspects of the full number in sequence. The full algorithm will have a first stage machine learning model that will learn to recognise single digits based on cropped single digit layer, similarly to the Stanford paper [5]. I then plan to apply this trained algorithm to the full images by scanning portions of each image and recording the likeliness of digits existing in sections of the image. This data will then become the input to a second learning model which will learn to recognise multi digit numbers. The second algorithm will aim to emulate the deep neural networks approach from a Google published paper [6].

Metrics

Due to the nature of the dataset calculating accuracy is a relatively simple task. Since the dataset contains the value of each individual data and the file name of the .png file it is associated with it all that is required is to is record how many full numbers have been recognised correctly and take a mean average where success is marked as a 1 and a fail is marked a 0.

Two separate studies have been used as criterion for measuring accuracy in this project. The first is the Stanford study directly associated with the project [5] and the second is a Google published paper that uses Deep Neural Networks (DNNs) to recognise the full address numbers [6]. Both studies emphasise the need for close to 100% accuracy to rival human accuracy. The Stanford paper [5] estimates human performance as having a score of 98% accuracy on this dataset, I will use this as the benchmark accuracy for this project. 90.6% accuracy was achieved with the k-means model in this paper and was the highest scoring single digit classifier [5]. When DNNs were applied to the dataset for full number recognition the highest score obtained was 97.84% [6]. For whole digit recognition the aim

of this project will be to exceed the performance of the DNNs any increase on 97.84% will be a success. Since the benchmark human accuracy is at 98% the aim will be to exceed the 98% accuracy of human viewers and the previous two projects. A secondary evaluation metric for the success of this project will be in the time it takes to analyse the whole data set. When DNNs were applied to the data it is stated that training the model took 6 days. If a smaller time frame can be achieved this will add to the success of the project. I will also state that if a lower score than 98% accuracy is given by the end of the project, but the training time for the algorithm is sufficiently short then the project can be considered a success as long as the accuracy is still higher than the 90.6% .

II. Analysis

Data Exploration

The dataset that I am using for this project is The Street View House Numbers (SVHN) Dataset and is a Standford University open source dataset [4]. The dataset takes its images from house numbers from Google Street View images sourced from around the world. It consists of two multidimensional arrays of data about the images and the .png images themselves. The first array contains data on the positions within the image of each digit shown and what number each digit is. The second array contains cropped images of every digit in each image and normalises them into 32x32 pixel squares. Both sets of formatted data are further split into three parts: train, test and extra. The dataset are currently in MATLAB file formats specifically .m and .mat. Part of this project will be to convert them to Python friendly formats.

This data set differentiates itself from other digit recognition data by containing solely images of real-world places. Previously the majority of work has been done on computer generated images or hand written digits. This data set aims to open up the possibility of even more unclear digits to be successfully. This application would be most useful for digital maps such as Google Maps and GPS navigation systems, but also has potential for real-time use through a camera whilst travelling.

Unlike previous digit recognition data sources this dataset poses more of challenge. Single digital images of numeric digits are easy to identify and whilst hand written numbers are harder to recognise they will still usually be written in dark colours with a white background. Real life images such as these and others taken from google maps are not unified in many aspects of the data. The pictures have been taken from a variety of different perspectives, this includes variety of distance, angle and lighting among other variables. None of the images could be described as anomalous or as outliers as such since they all contain numeric digits, but some will contain less obvious digit recognition data than others.

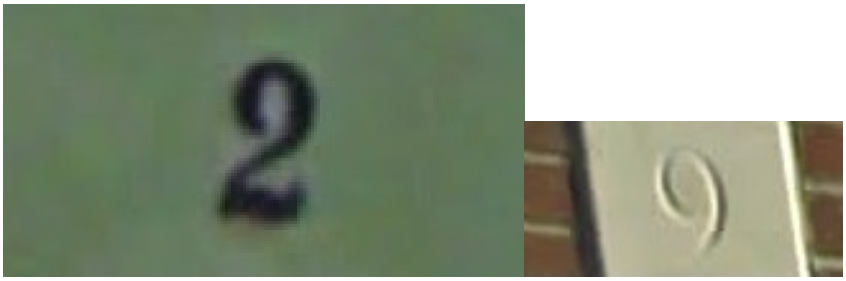


Figure 2: Image 99.png and image 362.png (right)

Figure 2 above shows two typically different images file 99.png (left) shows a clear number 2 on a lighter background with a dark coloured number and the number of pixels is relatively large within the set. Image 362.png (right) shows a number 9, the digit however is the same colour as the background material and the colours and contours of the wall either side seem to stand out more than the number itself. This forces to algorithm to identify many more parameters as it learns to recognise the digits. One advantage to this dataset is the volume of images available, with 3402 .png images and 73257 separate digits there is a lot of information to draw from.

Exploratory Visualization

Figure 3: Six images from the dataset with the individual digits outlined (taken from [4])

Figure 3 above shows several of the images in this dataset. Each image has had the separate digits separated and highlighted. Figure 3 shows the breadth of difference between the images in the dataset. Some images will have clearly defined digits with a large depth of range between colour values, such as the black digits on the white background in the top left. The image in the bottom left for example has two digits, but the digits are a similar colour to the background, there are also edges displayed on the left of the images that could confuse any algorithm learning from it.

Figure 3 also shows the difference in placing of the digits within the larger image. The middle left image showing the number 394 has 3 clearly placed digits moving sequentially at close y coordinates. This type of image should be easier for the proposed algorithm to analyse, although in this case the lines in the background may pose some difficulties. The bottom left image containing the number 104 shows the opposite scenario, with the three digits shown at an angle and tilted to approximately 45 degrees. Unforeseen issues could occur across the dataset when there are large numbers of very different images.

The data also contains separated images of the digits normalized to a 32x32 pixel height and width. This was intended to allow the algorithm to learn for a smaller number of outputs, 10. This technique also minimises the chance of any patterns in the background confusing a learning algorithm. For the application of my project I will be using both single digit recognition and multi-digit number recognition and the separation of single digits will be invaluable to the algorithm.

Benchmark Model

There has been much prior research done on image recognition in the past and within that there are many papers studying digit recognition. Within this Machine Learning Engineer Nanodegree Program the subject of handwriting recognition is explored in the module Intro to TensorFlow. Then more broadly image recognition was explored in the assessment Dog Breed Classifier in this course. Both of these course modules will be used to inspire the development of this Capstone Project. There is also many research papers to draw inspiration from with the subject of digit recognition having been research in many studies across the last few decades [1] [2] [3].

ALGORITHM	SVHN-TEST (ACCURACY)
HOG	85.0%
BINARY FEATURES (WDCH)	63.3%
K-MEANS	90.6%
STACKED SPARSE AUTO-ENCODERS	89.7%
HUMAN PERFORMANCE	98.0%

Figure 4: Algorithm results obtained from the Stanford study on single digit recognition [5]

I will be using the two studies mentioned previously as my benchmark models for this [5] [6]. The benchmark for the single digit recognition comes from the Stanford paper on real world digit recognition [5]. As shown in Figure 5 above this paper the researchers found that a k-means algorithm gave the best results for single digit real world recognition. K-means will be tested during the course of this project against a neural network approach to single digit recognition. Neural networks will be tested due to the commonality of their use in digit recognition. There are many papers and resources for designing efficient neural network architectures to digit recognition [7] [8]. The other reason for using neural networks is that the second half of this algorithm will use them to recognise the full numbers non processed images.

The second model, will recognise multi-digit numbers through the use a Deep Neural Network (DNN) as seen in [6] and discussed in the previous section. In this paper the researchers used a DNN of multiple layer with hidden layers of 3,071 units each. I will attempt to create a smaller neural net architecture with less units per layer to increase the efficiency of the model. I believe this will be possible as my neural net will only have to compute the frequency of digits occurring and not the recognise the shapes of the digits themselves. The input data will be a one dimensional array of likeliness rather than 4 dimensional matrix of images.

The final benchmark taken from these studies is the accuracy to aim for with this algorithm. The Stanford paper estimates the accuracy of a human observer to be at approximately 98% accuracy. Their single digit model achieves 90.6% with the k-means approach. The Google DNN approach to full numbers achieves 97.84%, which is significantly closer to the aim. This project will aim for 98% accuracy, but equalling the accuracy of either of the two prior studies would be acceptable.

Algorithms and Techniques

The approach problem will involve an algorithm with two defined models. The first will be a single digit classifier, cropped portions of the full images will then be tested for the likeliness of a single digit occurring within them. An array storing all the single digit information will the become the input data for a second model which will then compute the likeness of a certain multi-digit number occurring based on that data.

Figure 5: The workflow of the proposed algorithm

Figure 5 shows the overall workflow of the algorithm. The first two steps load the data into an appropriate format, likely with resizing of the full number images and some conversion to python friendly data types. The third stage trains the first model, the first model will aim to recognise single digits with a similar approach as the Stanford study. The fourth stage will involve taking 32x32 portions of the full images and finding the likeliness of a digit occurring there. The full images will be split into multiple smaller images likely moving through both the x and y directions across the image. An output array containing the likeliness of each of the 10 digits occurring for each cropped image will be stored for use in training the second model. The second model will take this 1 dimensional array as an input and learn to recognise the overall likeliness of a full number existing. The idea being that if a high likeliness is found at two different y positions but at the same x value it is more likely that this is the two parts of one digit and the neural net will learn to recognise these patterns. Finally the output of this second model will be tested against the full numbers as recorded in the Stanford SVHN dataset. This will be a simple case of taking the mean of correct and incorrect predictions for an overall accuracy metric.

III. Methodology

Data Preprocessing

With any dataset there some preprocessing will be required, this dataset is no different. In this section W will outline the data preprocessing that were undertaken for this project. Most notably this dataset was created with intention for use in the mathematical analysis software MATLAB. As such the first preprocessing steps were done using MATLAB scripts. Some preprocessing was then needed as a final step in Python.

The first step was to create a viable dataset for use, this was easiest done in MATLAB. Both full sets of images were included in the dataset in as .png files, which could be converted into usable formates in either Python or MATLAB so did not need any processing in MATLAB. The number data where single digit values had been recorded did however. For the 32x32 pixel images a .mat file existed containing all the image values at matching points in an array. This was contained in a .mat file that needed conversion into a .csv file or similar for use in Python. I undertook some experimentation and found that the format that MATLAB exported files to .csv was not easy for Python to read when loaded in Jupyter Notebooks. MATLAB also includes conversion to .txt files, which although can contain lacking information are universally accepted by most programming languages. In the end I found that converting to .txt files was the most reliable method of transferring the data. For the full number data a .mat file containing strings of the image names i.e. ‘100.png’ and each of the single digits associated with that file was included in the dataset. The digits were stored as separate points in the structure but were ordered appropriately from digits occurring from left to right. I wrote a short MATLAB script “Data_PreProcess.m” that aggregated each digit to one multi digit number and stored them in an array. This was then also exported to a .txt file for use in Python.

Once the data and metadata had been exported from MATLAB and imported into Python some more preprocessing was required before use in the algorithm. All of the images from each set needed to be stored in multi-dimensional matrices of size x.32.32. I made the decision to import them as greyscale images later on in the development stage as this did not seem to decrease accuracy and greatly sped up processing time. This had the effect of reducing the input algorithms from 4d matrices of x.32.32.3 to 3D matrices of x.32.32 size. The full scale images also required some resizing to make them appropriate size to be scanned in 32x32 portions. With some testing it was found that images with a height of 64 pixels and a width of 96 pixels gave the optimum shape. Finally the .txt files exported from MATLAB needed to be converted to usable arrays in Python and this was achieved through the use of the pd.read_csv() function. All data preprocessing was compressed into the class image_Reader() which can be fund in the Imange_Reader.py file in /Capstone_Project/Classes folder in the project directory.

Implementation

The overall algorithm can be split to 3 major sections:

- Data preprocessing
- Single digit recognition
- Multi-Digit recognition

The first stage is somewhat self explanatory and covered in depth in the previous section, as such this section will concentrate on the latter 2 stage of single and multi digit recognition.

The second stage single digit recognition breaks down into 2 further sections, training and recognition. The training portion involves training a learning algorithm to recognise single digits in 32x32 images. With inspiration from the Stanford paper [5] K-Means and neural network models were tested, overall it as found that for this application neural networks gave the better results so was used as the basis for this model.

The second part of the single digit recognition portion of this algorithm involved dividing the larger images into smaller 32x32 pixel segments and finding the likeliness of a digit occurring within them.

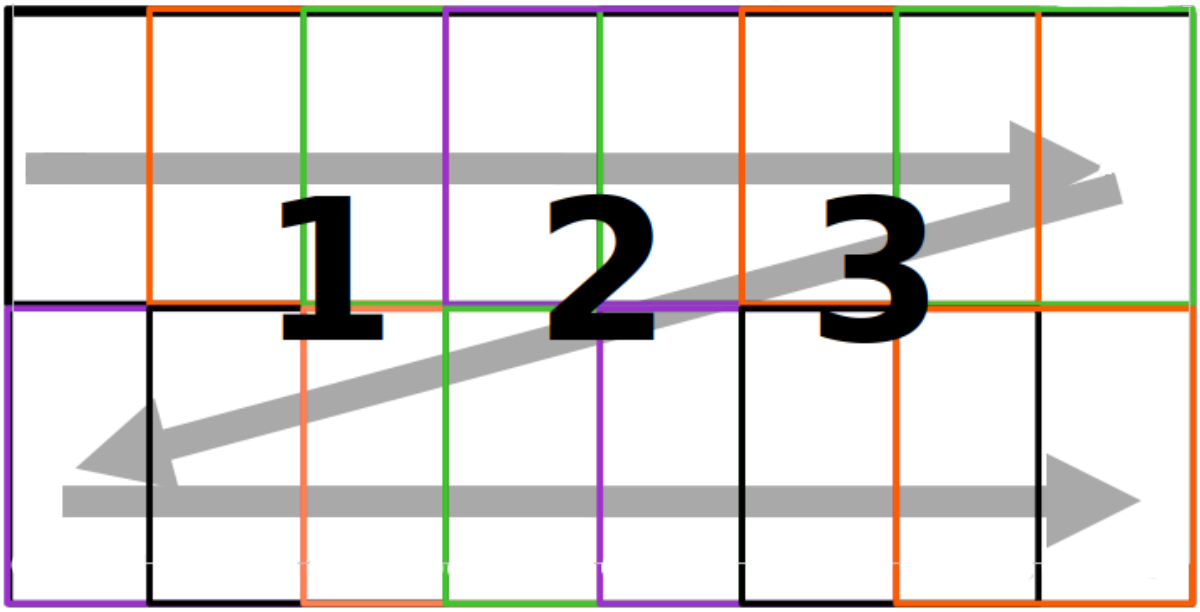


Figure 6: The scanning directions and number of cropped images

Figure 6 above shows how the scanning algorithm was implemented. In this case the images split into 2 main sections on the y axis and then 7 windows on the x axis. In the actual implementation 5 windows were used on the x axis and 3 windows across the y axis. However any number of windows could be added or taken away in both the x and the y directions. For the implementation of this portion of the algorithm the overall size of the full number images needed to be carefully examined.All python code for this portion can be found in the image_Scan() class found in: Capstone_Project/Classes/Full_Image_Scan.py.

The final stage of the algorithm involved taking an array of the likeliness of a digit occurring in portions of the input images and feeding that into a neural network. The input array was of length 150, this length was calculated based on the full number of windows (15) multiplied by the number of digits detectable (10). Since this was a comparability small input compared to the 32x32 images of the previous model a larger neural net architecture could be used.

Refinement

Each of the three major steps were refined during the design process. Firstly during the data preprocessing the stage and the training of the single digit classifier in the second stage the structure of the input data was decided upon. When the images were first imported they were imported and full RGB colour images. The digit classifier was run many times with multiple structures, with the large input data each test took long periods of time to train. With improved efficiency in mind as an aim for the project the images were imported as greyscale images, reducing the image dimensions by 3. The same tests were run with the same variety of input structures and it was found that there was little impact on the accuracy of the algorithm, but greatly decreased training times. With this in mind greyscale images were chosen as the input to the algorithm.

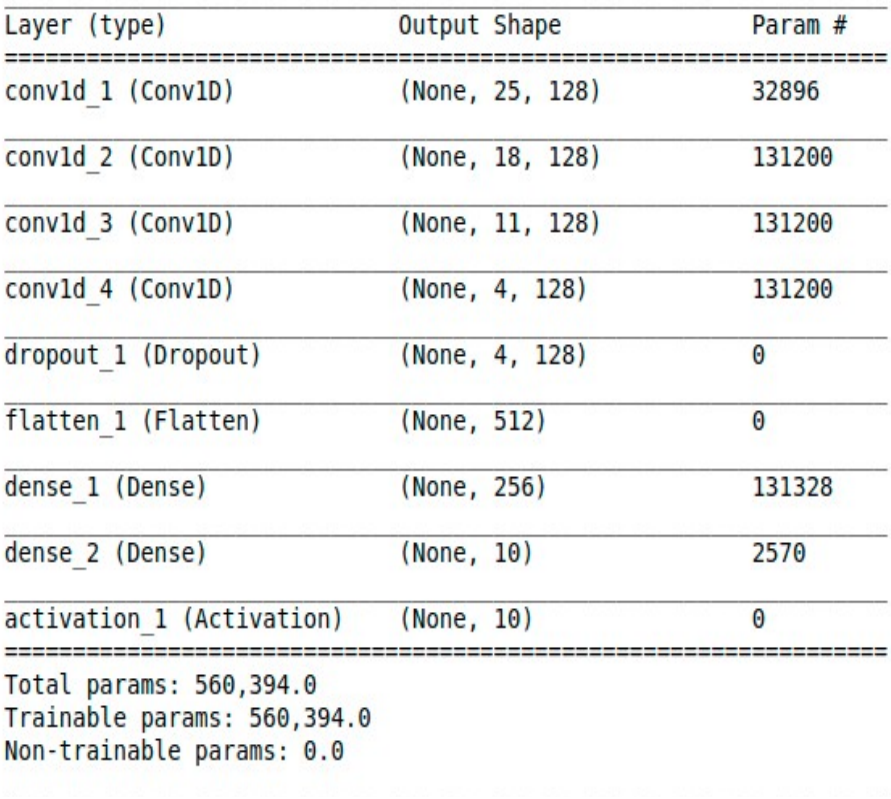
For the second part of the second stage of the algorithm the full number images were divided into smaller 32x32 portions and fed into the previously trained neural net to find the likeliness of a digit occurring in that space. The images needed resizing and normalising so that a consistent input size for the second model could be created. Several different combinations of height and width were tested but eventually it was found that a height of 64 pixels and a width of 96 pixels was most appropriate. When the image was lengthened beyond 96 the image became stretched and the digits lacked resemblance to the original single digit images and a similar phenomenon occurred when increasing the height without increasing the width proportionally. Increased but proportional dimensions could have been used but due to the high variance of dimensions in the input images the 96x64 ratio gave the most consistent either expansion or reduction of the input dimensions across all input images whilst loosing the least numeric information.

Finally for the second model the most refinement time was spent on the architecture of the neural net. Since the benchmark model [6] used vast DNN structures it seemed appropriate to increase the size of this architecture beyond the scope of the first model. This was further defined by the small size of the input array, only being a 1 dimensional 150 point array. Larger numbers of units and increased layers were implemented but also increased numbers of epoch and very large batch sizes were added. As the architecture was designed it was found that training time was much smaller than the single digit model and whilst a full deep neural network was not strictly implemented the architecture was substantially larger in size.

IV. Results

Model Evaluation and Validation

Model 1 Architecture:



Layer (type)	Output Shape	Param #
conv1d_1 (Conv1D)	(None, 25, 128)	32896
conv1d_2 (Conv1D)	(None, 18, 128)	131200
conv1d_3 (Conv1D)	(None, 11, 128)	131200
conv1d_4 (Conv1D)	(None, 4, 128)	131200
dropout_1 (Dropout)	(None, 4, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 10)	2570
activation_1 (Activation)	(None, 10)	0

Total params: 560,394.0
Trainable params: 560,394.0
Non-trainable params: 0.0

Figure 7: The model architecture of the single digit classifier

The architecture above was the final architecture used in the model. It was chosen for it's overall accuracy, but does suffer in time efficiency taking 74 seconds per epoch. It was run for 128 epochs, has a batch size of 256 and a learning rate of 0.0003.

Window numbers used in scanning algorithm:

- 5 across the x axis
- 3 across the y axis

This number of windows in both directions were chosen as they gave the best compromise between the size of the data outputted (as large as possible) and speed in running the algorithm, since predictions per window can take long periods of times.

Model 2 Architecture:

Layer (type)	Output Shape	Param #
dense_3 (Dense)	(None, 128)	19328
dropout_2 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 128)	16512
dropout_3 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 256)	33024
dropout_4 (Dropout)	(None, 256)	0
dense_6 (Dense)	(None, 512)	131584
dropout_5 (Dropout)	(None, 512)	0
dense_7 (Dense)	(None, 1024)	525312
dropout_6 (Dropout)	(None, 1024)	0
dense_8 (Dense)	(None, 20)	20500
activation_2 (Activation)	(None, 20)	0

Total params: 746,260.0
Trainable params: 746,260.0
Non-trainable params: 0.0

Figure 8: The model architecture of the full number classifier

A larger architecture when compared to the single digit model could be used for his full number model due to the smaller input data. Even with the larger network architecture it on took 5 seconds to run an epoch. It was run for 1000 epochs, has a batch size of 1024 and a learning rate of 0.0003.

To overall design of the algorithm is robust for many situations. As a base model it is not overly complicated, but also has a large array of parameters to adjust. For example the number of windows scanned could be increased and the input images could be more meticulously examined, or either learning model could both be increased or decreased in size dependant on much emphasis on speed there was in the application. Extra to this the dataset itself is very robust with many extra images for further testing included in the dataset. The model will also work effectively with other number images, such as text or hand written images.

Justification

The final results for this algorithm are as follows:

- Model 1: 18.92% accuracy
- Model 2 and overall accuracy: >1%

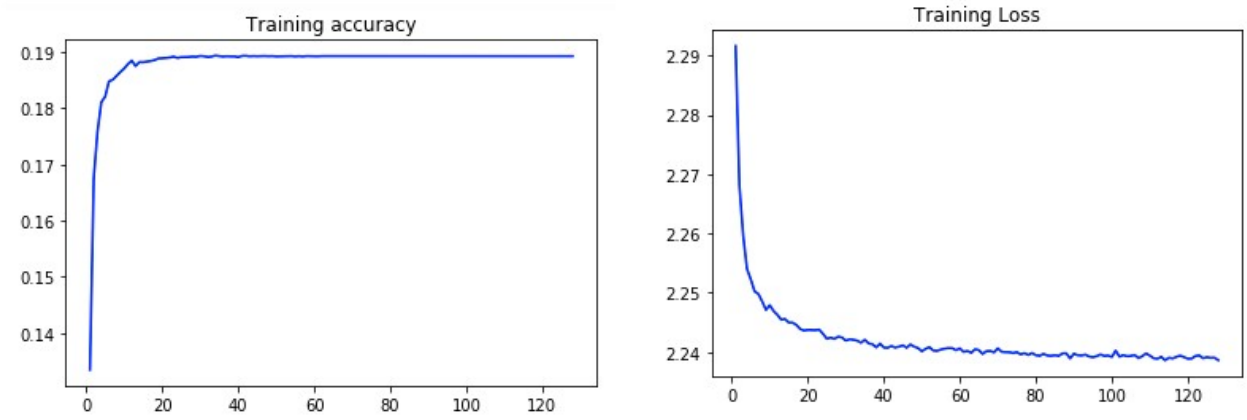


Figure 9: The Accuracy and Loss over time of the single digit classifier

The above results show that the model did not obtain enough accuracy to surpass the benchmark studies. The accuracy aimed at for this project based on the benchmark studies was a minimum of 90.6% with an aim to surpass the 98% human accuracy. With the overall accuracy of this project being at >1% the project did not achieve this aim. The first model was tested with a variety of different structures and inputs both large and small. The model was trained over large numbers of epochs such as 1024 and smaller such as 64. The learning rate was also experimented with, testing both larger (i.e. 0.1) and smaller (i.e. 0.000001) values, but every model started to converge towards small accuracy ratings, usually converging to between 18% and 19% accuracy as seen in Figure 9.

When examining the 10 class outputs of the network when applied to the testing data it was found that almost every image no matter the input always predicted the same output. For example in both images containing 2, 5 and 6 it might predict the every one is a 2. Since classifying all images as only 1 number would give approximately a 10% accuracy it is reasonable to assume that the more successful models were able to classify a small number of images correctly outside of chance.

Being unable to obtain more than an 18.92% accuracy in the first model meant that analysing the performance of the second half of the algorithm was not reasonably possible. When running the full algorithm >1% accuracy was achieved, however without an accurate digit detector it is impossible to evaluate the performance of the full digit detection model.

V. Conclusion

Visualization

Figure 10: A Google Map image of a street identified as door number 16 by Google Maps, with the image showing a door number 9 (Street address redacted) [9]

Figure 10 shows a typical problem faced by Google map users, where street information has been incorrectly classified. The image shows a house door showing the address number to be number 9 however as shown in the top right hand corner it has been classified as door number 16 by Google Maps. This situation is a common frustration of many users of the application and projects such as these have huge potential for interest from the public and commercial applications. Having accurate maps that require little to no human interaction in creating are useful across any application and could increase accuracy and efficiency in travel time for people at all levels.

Many machine learning applications start from a point of inspiration in the real world and my project was no different. Using the influence of human interaction with numbers, by splitting the recognition of digits within a whole number and then combining into a full number them is a process that is seldom seen in this area of machine learning. Although this attempt at image recognition of whole numbers has not achieved satisfactory results both the subject matter and approach to the algorithm design have great potential for day to day and commercial applications among others.

Reflection

The entire end to end process was as follows:

1. Data preprocessing in MATLAB
2. Final data preprocessing in and loading into Python
3. Training a single digit classifier on 32x32 images
4. Dividing full number multi-digit images into 32x32 sections
5. Predicting and storing the likeliness of a digit occurring in each 32x3 section
6. Training a multi digit number classifier on the output of the initial model
7. Measuring the overall accuracy of the algorithm

The most difficult aspect of the project was training the single digit classifier accurately. The highest accuracy achieved was 18.92%, multiple different architectures were tested and none could achieve a higher performance. Since efficiency in time was a key aim for the project deep neural networks taking several days were not an option. Although accuracy were not met at the end of this project the exploration of neural networks, learning how seemingly minor changes could affect large changes in the outcomes was both the most challenging and interesting portion of the project. Specific improvements that could be made on the algorithm will be explored in the next section.

Using multiple models in one project is an approach that has been seen in other applications. Part of the inspiration for this project was the Actor-Critic method as seen in Part 2, Lesson 19 of this course. The Actor-Critic method is a popular approach to learning and shows multi-model approaches can be used in general settings. With the complexity of the problem this project aims to overcome, this could be an effective approach and with a higher initial accuracy there is good reason to believe it could be successful. With precedence for this type of approach it is also reasonable to say that it could be used for other wider image classification problems.

Improvement

Neural Net Architecture

Many different architectures were tested over the course of this project. While developing the algorithm several different approaches were taken. It is common in digit recognition that neural nets are designed with relatively small architectures. More complex problems such as handwritten digits require larger structures but not often what might be classified as Deep Neural Networks (DNNs). Both approaches were tested during this project, with layers as small as 32 units and as large as 512 units. Larger than this created very inefficient algorithms taking long periods of time to run. Ultimately neither large or small neural nets gave appropriate accuracy values so the final architecture was a compromise that gave the best results that could be obtained.

Improvements could be made in several areas, more testing on learning rates could be done. The project tested large and small learning rates, but did not incrementally test them across the same networks. More data on how learning might affect this algorithm at various layer depths would be valuable. Randomisation of weights in the network was used in the final architecture, however again more testing could be done on which layers would most benefit from randomisation and the benefit or loss from using either normalised randomisation vs uniform randomisation of weights. Finally greatly increased number of epochs, the highest currently was 1024, but more could be tested sacrificing time for accuracy.

Alternative Learning Models

The original Stanford paper attached to this data set [5] tested multiple learning methods when approaching the problem of single digit classification. They found that the k-means approach was the most effective having the highest accuracy. Neural networks were chosen for this project as it gave a uniform approach to the problem across the two models, but investigation into other possible approaches to the first model could improve the overall algorithm. K-means was tested during the development of this project but this implementation gave worse results than the final architecture made. Further exploration into k-means would still however be valuable alongside other learning algorithms in the paper [1].

Increased Dataset

One final improvement that could be made is to the dataset itself. A common issue across this dataset is that the background non digit portions of the image often contain patters that a machine learning algorithm may not be able to effectively process. For example there are often straight or angled lines in patterns in the background. An algorithm that at its first stage is trying to identify angled and non angled lines to recognise digits such as 1 or 7 would often confuse background patterns for numbers.

One approach that could improve performance on this problem is to include text images of digits and even handwritten digit images into the dataset. When looking at how a human learns to recognise numbers in childhood they are first introduced to clear printed or written numbers before they are given more complex images. Introducing clearer images into the dataset and then inputting them early on in the learning stage could help the algorithm learn the difference between irrelevant patterns and digits.

Conclusion

Overall the project did not achieve the aims that it set out to, however there is much potential in this approach to digit recognition in real life image. The problem as a whole is an important one with much potential in both in day to day life and for commercial applications. The approach of splitting the learning into separate models and the problem into separate smaller sections is a valid way of addressing machine learning obstacles. The overall results did not achieve their full aims, but did show potential for learning and improvements with more experimentation. Overall the project did not achieve its aims at the outset but does set the stage for interesting future work.

References

[1] Le Cun, Y., Boser, B., Denker, S., Henderson, D., Howard, R., Hubbard, W. and Jackel, D. (1990). Handwritten Digit Recognition with a Back-Propagation Network. *Advances in Neural Information Processing Systems*, 2, 396–404.

[2] Cireşan, D., Meier, U., Gambardella, L. and Schmidhuber, J. (2010). Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*, volume 22, p.3207-3220.

[3] Cireşan, D., Meier, and Schmidhuber, J. (2012). Multi-column Deep Neural Networks for Image Classification. *Technical Report No. IDSIA-04-12*, IDSIA / USI-SUPSI.

[4] http://ufldl.stanford.edu. (2011). The Street View House Numbers (SVHN) Dataset. Available at: http://ufldl.stanford.edu/housenumbers/

[5] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, Andrew Y. Ng. (2011). Reading Digits in Natural Images with Unsupervised Feature Learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*. Available at: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[6] Goodfellow, I., Bulatov, Y., Ibrarz, J., Arnoud, S. and Shet, V. (2014). Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. *International Conference on Learning Representations*.

[7] Le Cun, Y., Jackel, L.D., Boser, B., Denker, J.S., Graf, H.P. Guyon, I., Henderson, D., Howard, R.E., Hubbard, W. (1989). Handwritten digit recognition: applications of neural network chips and automatic learning. *IEEE Communications Magazine.* Volume: 27, Issue: 11, Nov. 1989.

[8] Lee, Y. (1991). Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks. *MIT Press Cambridge.* Volume 3 Issue 3, Pages 440-449.

[9] Google Maps, April 2019. Google Maps [online] Available through: Google Maps <https://www.google.com/maps> [Accessed 14 September 2019].