Capstone Project

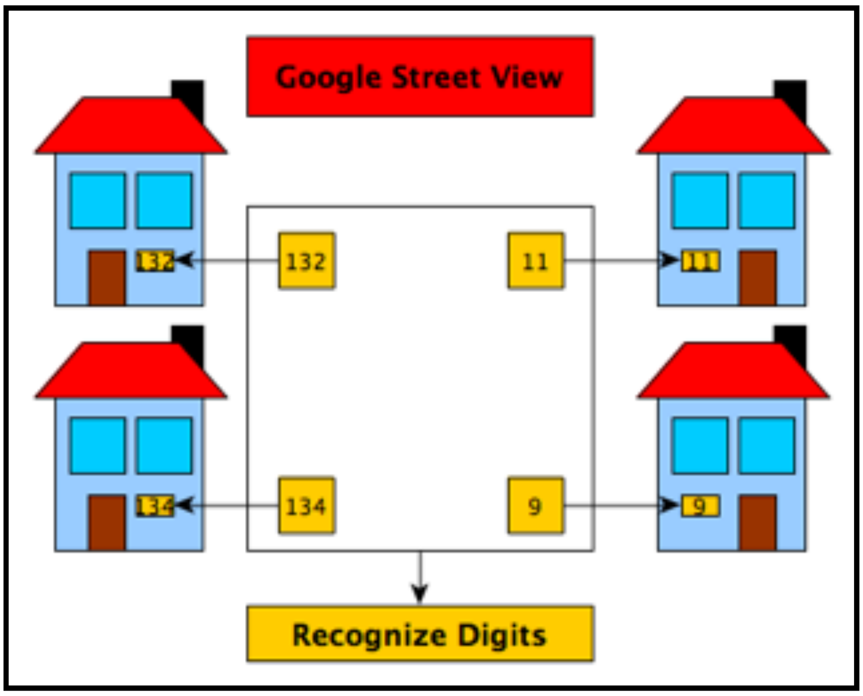
Machine Learning Engineer Nanodegree

Nico Axtmann
October 19th, 2016

1. Definition

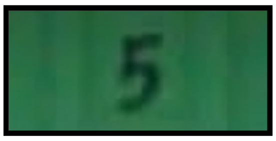
Project Overview

This project is a deep learning task. Furthermore it is about image classification. A model will be developed and trained to decode sequences of digits from image and recognise them.



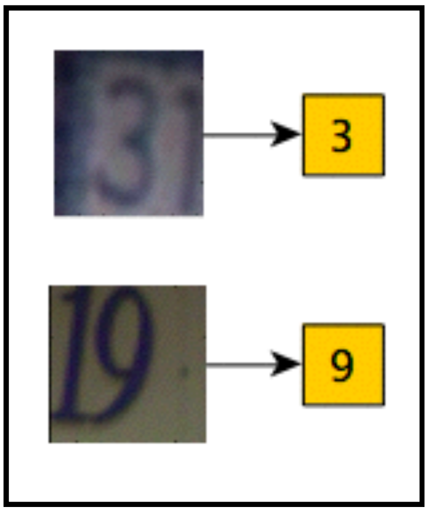
OVERVIEW OF THE DATASET

For this project I use the dataset The Street View House Numbers (SVHN). The data of this dataset consists of images of Google Street View. The images contain a sequence of house numbers.



EXAMPLE OF SVHN

The are different formats of the dataset. One format is full numbers SVHN provides 10 classes of digits (0-9) and the other format is cropped digits. The full numbers are the original images. The cropped digits format have a fixed 32x32 pixels resolution and the image show only the number. There is no overhead data like “blank” space. For this project we will use the full number digits format.



CLASSIFIER TASK

In order to solve this task we need to build several classifiers which understands the images and recognise the digits. I am using a convolutional neural network (CNN) to scan through the image and extract the features.

Problem Statement

To wrap it up, in this project I develop a model to decode a sequence of digits from images. In order to achieve this result I use a 5 step plan:

Step 1: Setup

In the first step the SVHN dataset is downloaded and saved locally. Also it is checked if the data is straight.

Step 2: Preparation

In the second step the downloaded data will be manipulated. Since the features of the data are stored in a MAT-file I have to extract its values with some libraries. To finish this step, the extracted data will be saved locally in a pickle file.

Step 3: Exploration

I use some exploration techniques of the data to get more insights and information about the available data.

Step 4: Preprocessing (Preparation of the data for training)

In this step the images are scaled down to a certain size. I designed this process in a way that most of the important information will be present in the picture.

Step 5: Training

Now it comes to the fun part. In this step I use a simple end-to-end LeNet-5-like architecture for my deep learning model. The model will be trained and saved. Also the model will be optimised by different parameters.

Step 6: Deployment

In the last step the previous trained model is restored and applied unseen images. In this step I explore the functionality of model in reality.

Metrics

The performance of the model depends on the correct classified numbers of an image. The more numbers are classified correctly, the better the model performs. Of course the easiest measure model would be to say that it only works if all numbers are classified correctly. However, I think that it is most important to look at the correct classified numbers at one pictures.
I chose Accuracy as a metrics. The this metrics computes the percentage of correctly classified predictions.

	Predicted: True	Predicted: False
Actual: True	True positive (TP)	False negative (FN)
Actual: False	False positive (FP)	True Negative (TN)

The accuracy can be calculated by the following formula:
Accuracy= (TN+TP) / (TP+FN+FP+TN)

The abbreviations always represent the count in a certain field.
I selected this metric because it is easy to use. Also this metrics gives a fast overview if a model performs better an other model.

2. Analysis

Data Exploration

There are three relevant dataset in the SVHN corpus.

1. train: In the train dataset there are 33402 samples present.
2. test: In the test dataset there are 13068 samples present.
3. extra: The extra dataset has a total of around 2 GB (500000+) of samples present.

Every dataset has one file called **digitStruct.mat** which has the extracted features of a file present. This file is a struct array which contains information about the digits of one image. Furthermore the array contains information about the position, size and label of each digit. The position and size of each digit are represented in bounding boxes.

The following pictures shows an example of one house number.

SAMPLE OF ONE IMAGE

Now with the information of the digitStruct.mat file this image would look like the following.

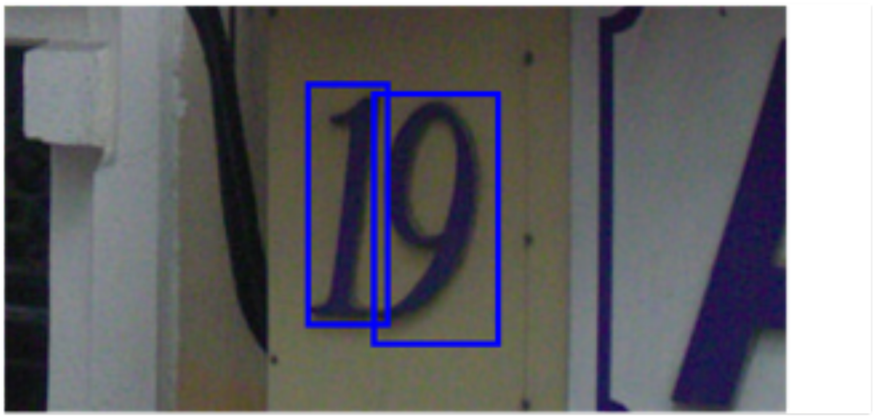


IMAGE SAMPLE WITH BOUNDING BOX

In a nutshell the file digitStruct.mat contains information about:

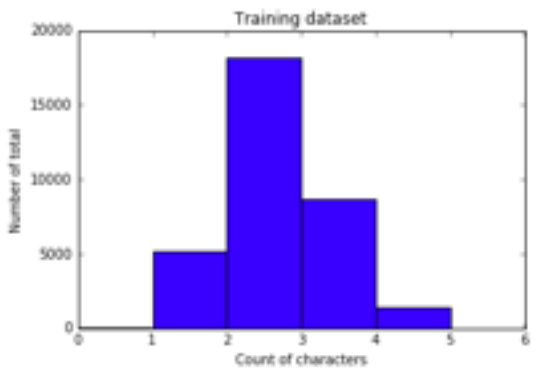
1. Digits in an image
2. Bounding box of each digit (position, size)
3. Filename of the image

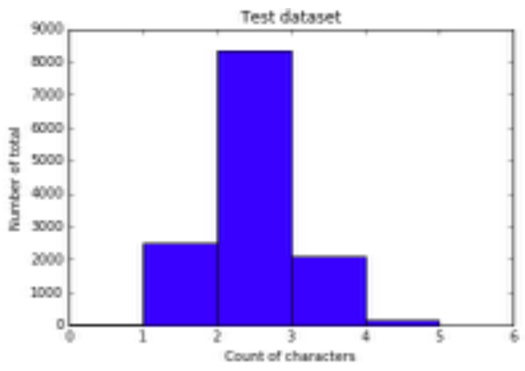
“SVHN is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting.”[1]

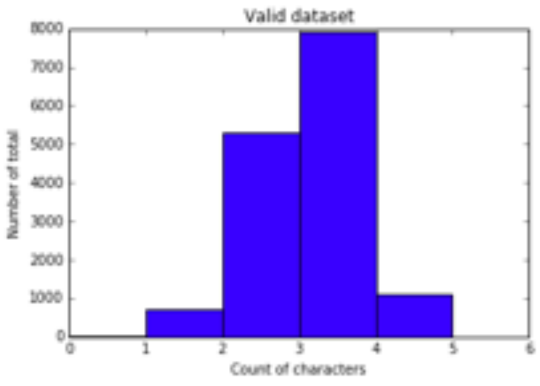
In order to work with this dataset there are several libraries required. The dataset or more specific the digitStruct.mat can be opened with a library called h5py.[2] This library is an interface for the python language in order to work with the HDF5 binary format.

Exploratory Visualization

The first thing I explored in the dataset is the distribution of size of the house numbers. This point is very important for further design steps.







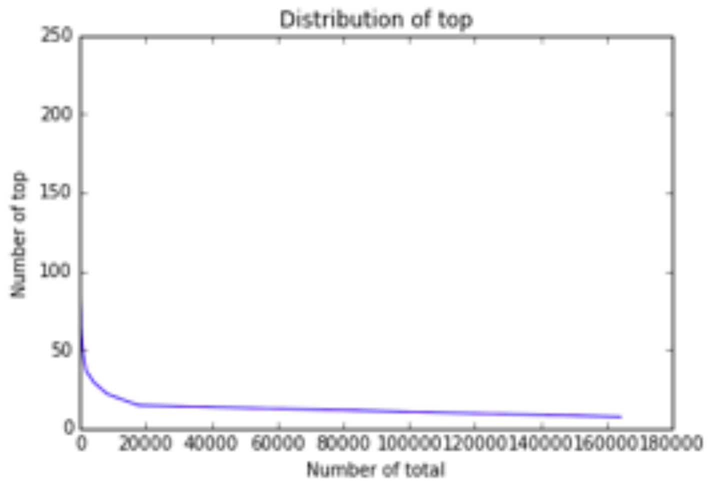
[1] http://ufldl.stanford.edu/housenumbers/

[2] http://www.h5py.org/

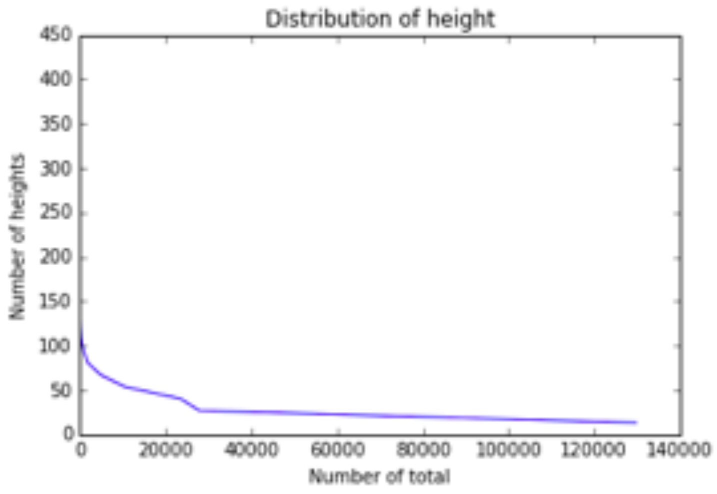
The diagrams shows the there are almost no pictures which contains 5 or more digits in the dataset. Mostly the pictures contains 2 and 3 digits. The following tables shows more information about the occurrences of the number of digits in each dataset.

Digits in a picture	Training Dataset	Testing Dataset	Validation Dataset
1	5137	2483	696
2	18130	8356	5287
3	8691	2081	7907
4	1434	146	1103
5	9	2	7
6	1	0	0

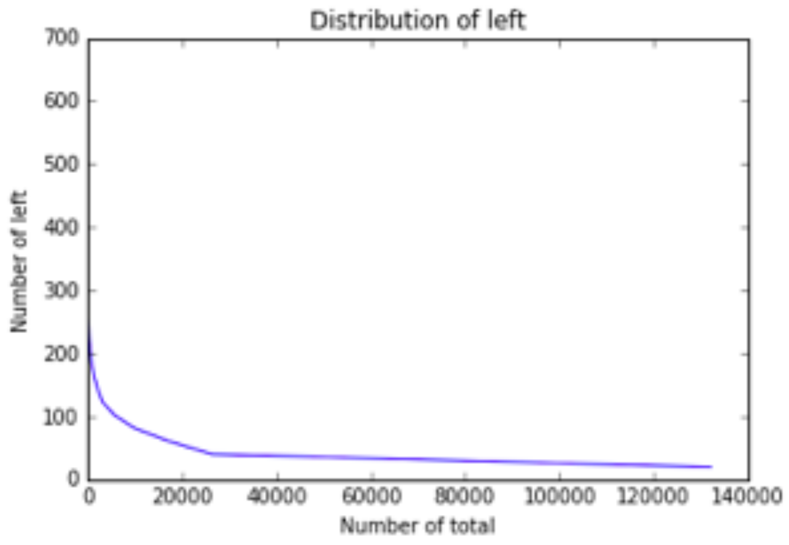
Another point which should be explored is the statistical information about the bounding boxes. To recap, the bounding boxes is described by the features top, left, width and height.



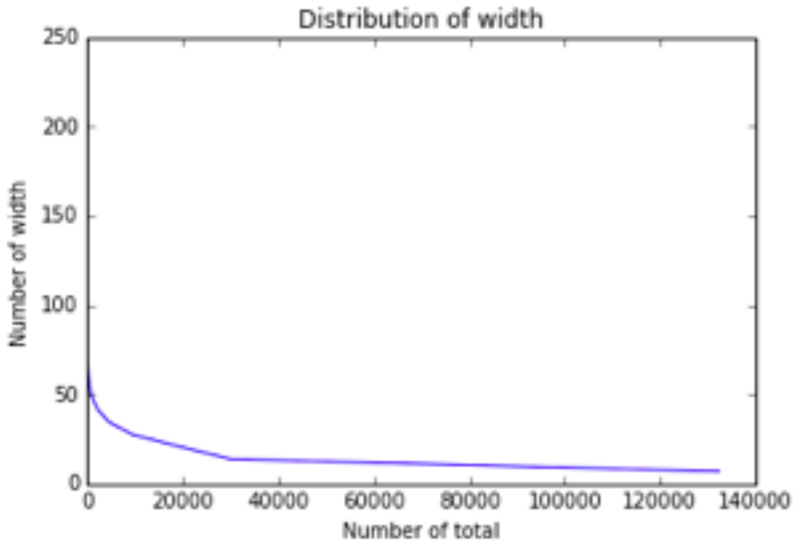
DISTRIBUTION OF TOP



DISTRIBUTION OF HEIGHT



DISTRIBUTION OF LEFT



DISTRIBUTION OF WIDTH

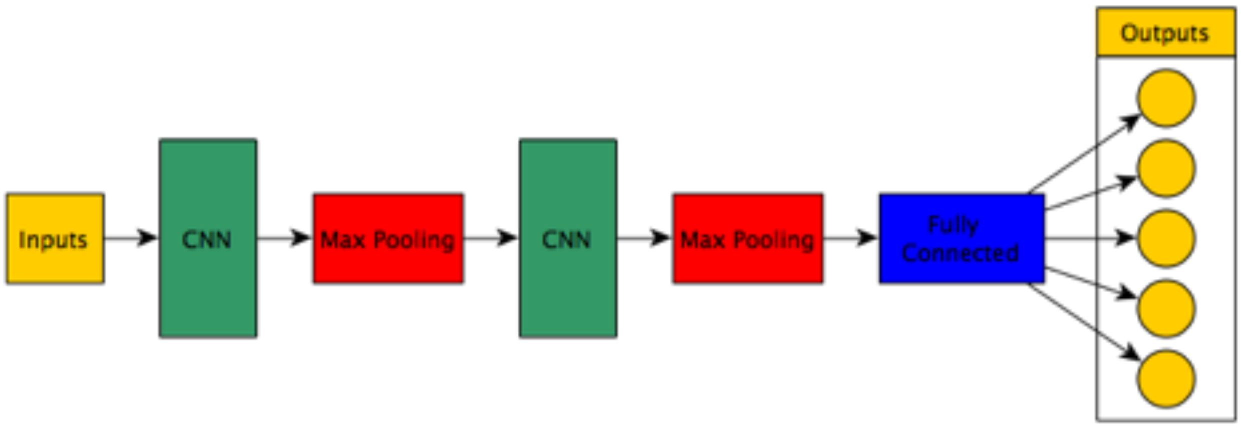
This table show the statistical information of the features.

	top	height	left	width
min	7,3	13,4	19,6	6,9
max	219	403	618	207
mean	113	208,2	318,81	106,97

Algorithms and Techniques

I am using a convolutional neural network to predict the digits in a picture. Convolutional neural networks are also known as convnets. Convnets are considered to work very well for image classification because they analyse the pictures piece by piece and are statistical invariants.

I choose a simple end-to-end LeNet-5-like architecture for my deep learning model.



LENET-5 ARCHITECTURE

Inputs

The inputs of this deep learning network are the images of the dataset in a normalised form. The pixels of the images are 32x32.

Convolutional Neural Network (CNN)

Convolutional Neural Networks or short CNN are very effective for tasks like image recognition and classification. The primary purpose of CNN is to extract features of input data. The input data is e.g. an image. Every image can be represented in a matrix of pixel values. Now the CNN tries to create a spatial relationship between the pixels by learning the image features. This happens through a kernel. A kernel is like a filter. The CNN scans with the kernel a small square of the image to learn about the relationship of the pixels. This can happen in several ways. Another term which I like to introduce at this point is stride. The stride can be seen as the pixel movement of the kernel. A low stride results in extracting more features, because it will not tempt to skip information. A high stride result into extracting less features and is regarded as an aggressive strategy. The outcoming result by sliding with the kernel over the image is called feature map.

The size of the feature map is dependent on some parameters like the stride, depth and padding. I already explained the stride.
The depth is the size of filters we are using to extract the features.
The padding decides about the size of the feature map. There are different parameters for the padding. The same padding results into creating the same size of the feature map like the input. The valid padding result into creating a little smaller feature map of the input.

Max pooling

Since a high stride has some disadvantage like losing a lot of information, another technique which I like to introduce is pooling. Pooling is a technique to combine the neighbour hood. It reduce the dimensionality of each feature map. However it retains the most important information. There various of types like max, average and sum pooling.
I chose the max pooling type. Instead of calculating the sum or the average, the max pooling chose the highest element.

Rectified Linear Unit

The rectified linear unit (relu) is an additional operation to the CNN. The relu is applied on all pixels and modifies the feature map. In detail it will replace all negative pixel values with zeros.

Dropout

One additional technique is called Dropout. This technique prevents the neural network from overfitting. To prevent overfitting the dropout only active certain connections of a layer to build a robust learner.

Benchmark

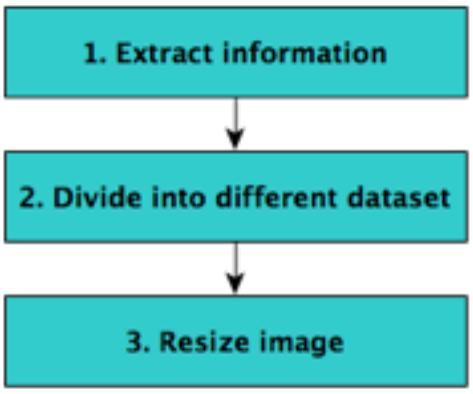
The benchmark of this model is measured by the accuracy. This means that we compare the predictions to its true values. In several experiments before I chose the LeNet-5, my models did not predict very well. Mostly the predicted like 80% of the time correctly. So guess that if i come closer to 100%, this model architecture will workout better as the other architectures. The paper "Reading Digits in Natural Images with Unsupervised Feature Learning"[3] claims that Humans classify the numbers correctly is 98% for every case. I would guess that if the model predict 93-95% correctly for every number that it would work every well.

[3] http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

3. Methodology

Data Preprocessing

There is some data preprocessing required, because the images do not have the same sizes. The first task is to extract the information of the digitStruct.mat file. The second task is to divide the data into train, test, valid and prediction dataset. The third task is to shrink the size of the images. At this task I took the information of the bounding box to generate high quality data.



VISUALISATION OF THE TASKS

Implementation

I created several notebook to break down the workflow. Every notebook has one different responsibility.

1_Setup.ipynb

This notebook sets up the project. It will download the dataset from the official SVHN side and save them as a pickle file.

2_Preparation.ipynb

This notebook extracts the features of the digitStruct.mat file and divides the data into train, test, valid and predict dataset.

3_Exploration.ipynb

This notebook is responsible for the analytics of the dataset. It extracts statistical information of the data and visualises the features.

4_Preprocessing.ipynb

This notebook is responsible for the preprocessing of the data. The images will be scaled down to 32x32 pixels without losing the important information. The results are saved again in a pickle file.

5_Training.ipynb

This notebook contains the code which generates the LeNet-5 architecture. Also the training of model take place inside this notebook. After the training the model is saved for future purposes. The following explains the implementation of the LeNet-5 architecture.

Inputs
The inputs of this deep learning network are the images of the dataset in a normalised form. The pixels of the images are 32x32.

Convolutional Neural Network (CNN) 1
The first layer is a convolutional neural network. The kernel or the patch size is 5x5 pixels. The kernel is equal to a magnifying glass which analyses the pixels piece by piece. The movement of kernel is defined by the stride. In the first CNN I chose the stride 1. This means that the kernel is moving by 1 pixel after it read. I chose the same padding. This means that the output of this CNN is equal to size of the input images. The result of the CNN are so called feature map which contain information about the generated output.
In a nutshell a CNN creates a larger depth of the image which results into more input values.

Max Pooling 1
In the CNN we took a low stride. I also could choose a more aggressive strategy by taking an higher stride. However, this would result into losing a lot of information. This is the reason why I took a slow stride. The next layer is a pooling layer. The pooling layer combines the neighbourhood. In this context this means the nearby pixels. I choose the max pooling layer because it is parameter free and often more accurate than other advanced convnet techniques. The max pooling is a technique to reduce the dimension. It reduces the dimensionality by taking the max value of a region.
The max pooling layer also posses stride and padding. I chose a stride of 2x2 pixels and same padding.

Convolutional Neural Network (CNN) 2
The third layer is again a CNN. It has the same kernel size as the previous. The differences between this layer and the previous one is the input size.This

CNN has a bigger input size which results in an deeper depth of the input values.

Max Pooling 2
The fourth layer is again a max pooling layer.

Fully Connected
The fully connected converts the previous network a one dimensional network. This network produces the values of the outputs.

Output
The output layer is represented by different logits. The logits represent different values which can be transformed by softmax to a probability. This probability shows the percentage of recognising a character as a certain value.

6_Deployment.ipynb

This notebook contains a dataset which the model tries to predict. It contains the code which generates the graph for the LeNet-5 architecture. Furthermore the trained model is restored and applied on the dataset.

Complications at implementation

I had several issue to extract the attributes of the digitStruct.mat file. After some debugging I found a solution on how to extract the attributes. Further complications are reflected in the conclusion.

Refinement

I implemented different sorts of deep learning architecture. After some experiments I decided to stick to the LeNet-5 architecture, because it showed the best performance according to the accuracy.
In a first attempt I created 5 classifiers with a parallel LeNet-5 architectures. This attempt reaches a 82% accuracy after 30000 steps. While I thought I am on a right track the model did mess up a lot when it came up to the prediction of unseen data. It did not classify a single number of a lot picture correctly. At this point I decide to change my implementation.

In order to get better results I decided to add the five classifiers directly at the end of fully connected model. I solved the problem of the prediction and the model started to classify the images correctly. Also I used different parameters in order to generate the best model. I added a dropout layer to make the model more robust with a keep prob of 0.7. Also I tried different learning rates and chose 0.001. The combination of these approached improved the accuracy by 10%. Also the prediction was now better.

The finished model implementation reaches a 93.2% accuracy after 100000 steps.

4. Results

Model Evaluation and Validation

After training my model several times with different parameters on a GPU, the model achieved the following values:

Minibatch loss at step 90000: 1.037044
Minibatch accuracy: 93.8%
Validation accuracy: 84.6%
Test accuracy: 93.2%

The final model has different the following parameters:

- patch size = 5x5 pixels
- 2 CNNs with different depths
  - CNNs have depths of 32 and 64 pixels
- 2 Max pooling layers
- I added 5 classifier which predict the digits connected to fully connected layer
- learning rate 0.001 and decay rate at 0.9
- Trained with dropout layer with a keep probability of 0.7
- trained with 100000 steps
- the model used the gradient descent optimiser

Justification

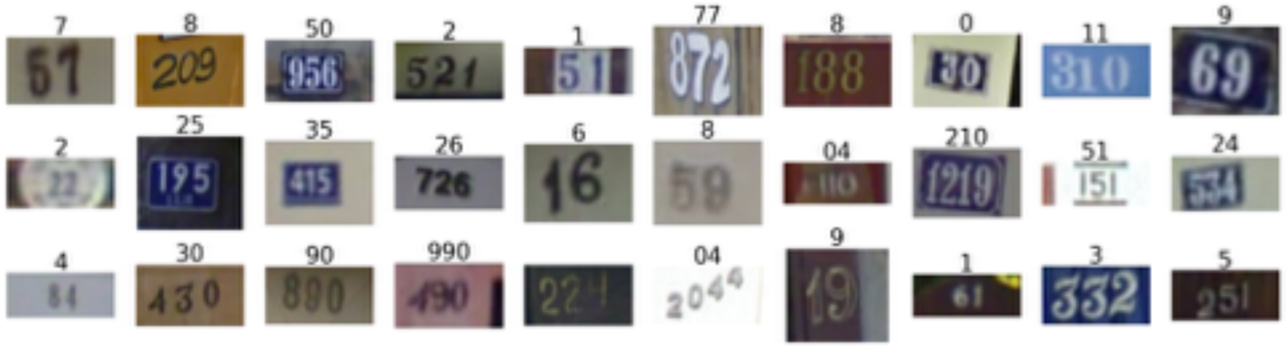
I think that my model does a good job. It was just trained by 35000 images. In the valid dataset there are more images which could be used to train the model more in depth. The model achieved a accuracy of 93.2% which is very close to the recognition level of the human.
The robustness of the model could be improved a little bit more on the unseen dataset. While the mini batch accuracy is 93.8% and the test accuracy is 93.2% it seems for the model to be very robust. The validation accuracy is 84.6%. This could be an indicator that the model is not robust enough for some unseen data. However by comparing the distribution of the size of the numbers between the training and valid dataset, it shows some

explanation of the decrease of robustness from the valid dataset. This is because there is the same size of house numbers with the size of 4 in the validation dataset than in the training dataset. This could probably be an explanation. However, I still think that the model is robust enough and it does quite a good job.

5. Conclusion

Free-Form Visualisation

For my deployment process I decided to run the trained model against unseen data. The unseen data was picked out of the extra dataset. I picked 30 images out of the extra dataset in order to see how well the model works in reality. It identifies a lot of numbers correctly out of the images. It does work a worser than expected. However, I am satisfied with the results, because the training size was not high. The following pictures shows an overview the predictions.



OVERVIEW OF THE PREDICTIONS

Reflection

This project taught me a lot about data extraction and deep learning. I think that deep learning is something very special. You do not have to do the same amount of work in feature extraction like in machine learning or predictive analytics. One throw in the data into an architecture and grabs a coffee. After drinking the coffee a model is computed which performs really good in contrast to the amount of time spent to develop the model. These models work better than some special developed algorithms.

So in this project I developed a model which can recognise digits from house numbers. In order to achieve the model several task have been done. The project could be split into the following tasks.

The first task was to download the data and extract the features. This was not so easy for me, because the lack of documentation on how to access the features.

The second task was to resize the images. I made a lot of mistakes in this task. After several different attempts and brute force methods it finally worked and I felt proud to achieve this.

The third task was to implement a model in tensorflow. At this step I had to do a lot of research which model architecture could work and which could not work so well. Also another point which I did in this task, was to learn tensorflow. Tensorflow is an awesome library but it lacks documentation and examples at the moment. This step was tough. I implemented several model and other dataset until I got the concept of programming with tensorflow. At this point the udacity deep learning was helpful in order to learn the theory of deep learning. But it did not help me to improve the practical skills. I did several research and finally understood what I had to do in order to implement the LeNet-5 architecture.

The fourth task was to deploy the model. At this step I realised I made mistake, because my previous model was not generating any output. Finally, I did it.

To sum the project up, I did learn a lot about data preprocessing and deep learning. I implemented several algorithms to extract the features and implemented a deep learning architecture. The architecture consist of 2 CNNs and 2 max pooling layer as well as one fully connected layer. The output was implemented by 5 classifier which predict the output of the house number.

GitHub and the udacity helped me a lot to understand deep learning and tensorflow. The course helped me a lot to get a good foundation in machine learning. Also I am thankful to member of the udacity forum. I got many ideas of the udacity and approaches to solve this project.

Improvement

There are several way on how this project could be improved. The easiest way would be to train the model with more data. But this is also most expensive approach, because I would need a better GPU to compute the model faster.

One approach would be to play around with the parameters to receive a better accuracy. Also it would be possible to use an other optimiser to get better results.
An other approach would be to use other deep learning architectures to get better results.
Also I could think about the idea of using recurrent neural networks (RNN) in order to read the digits. Through this way the model would not be limited to 5 digits and could predict more digits. The RNN could serve the inputs for the CNN. This probably would generated better results.