Machine Learning Engineer Nanodegree

Capstone Project

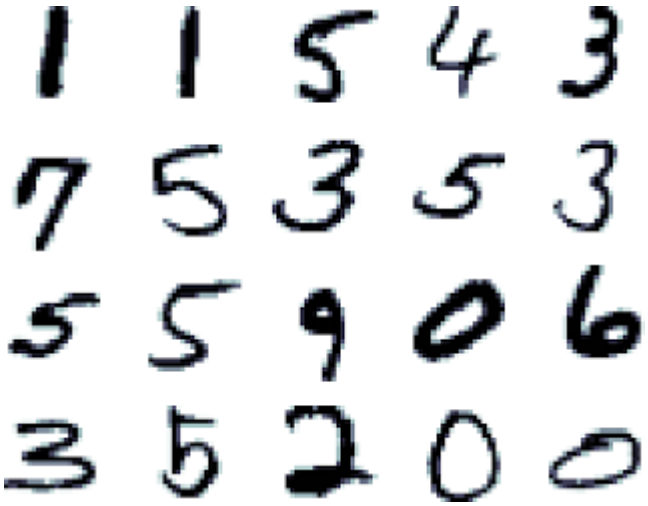
Daniel Resende
April 16th, 2017

I. Definition

Project Overview

Image recognition is a deeply studied field due to its vast applicability, e.g. accessibility with image caption, readding document, etc. With that in mind, recognizing characters became a well attacked challenge, thus datasets like the MNIST(1) are thoroughly addressed with algorithms approching perfect performance by now.

MNIST dataset example



But recognizing characters in real-world image imposes a lot more difficult due to the variation and distractions as it can be seen in papers addressing this kinds of datasets like Multi-digit Number Recognittion from Street View Imagery using Deep Convolutional Neural Networks(2) and Very Deep Convolutional Networks for Large-Scale Image Recognition(3) and others.

Problem Statement

With the MNIST(1) dataset we can see how a character recognition problem can be addressed in a document clean like dataset. On the other hand in Multi-digit Number Recognittion from Street View Imagery using Deep Convolutional Neural Networks(2), the authors used the Street View House Number dataset as a representation of real-world image to understand the difficults imposed by this kind of dataset. Thus the problem I'll try to address with this project is going to be a mixture of both.

I'll use the MNIST-like format of the Street View House Number(4) dataset, with over 73000

32-by-32 pixel color (RGB) images taken from house numbers found in Google Street View. Those images contain a digit in the center, from 0 through 9, but may or may not contain other digits or parts of other digits. Thus the objective will be to recognize the center digit, i.e. a multiclass classification problem, using the pixels of the images as features, giving a real-world challenge to a now well know problem, character recognition, and see if it can achieve at least human-level accuracy with this dataset.

SVHN dataset example

Metrics

Since it is a classification supervised problem, to evaluate the model and assess if it learns how to efficiently recognized the center digit in the images I'll measure how accurate the model is, i.e. if the result of the model is the same as the label that it has.

This metric is a good choice for the task at hand, because it is very simple to calculate and, since the objective is to tell in which category the input is, without the possibility of being on more than one label at the same time, it clearly tell how well the model is doing its job.

To do this in practice, I'll take the argmax of the softmax output, the probability of the image being a given number, and compare to the label of that image.

In sum, the accuracy it the number of correct outputs divided by the total of outputs.

Just as a sanity check, it can be said that the model is completely useless if the accuracy is equal or less than 10%, because, since the data is label from 1 to 10 (with the label 10 meaning the digit 0), a random guess would give a 10% accuracy.

II. Analysis

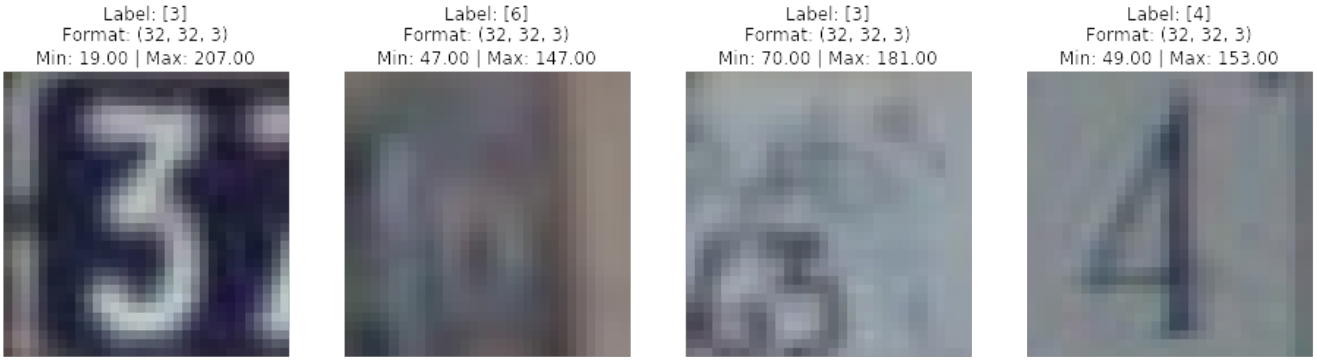
Data Exploration

As said before this dataset used in this project has 73257 32 by 32 RGB images of house numbers croped to display a center digit and with each pixel value being between 0 and 255 for each color. Those images are labeled according to the center digit with the digit 0 being represented by the number 10, thus the labels go from 1 to 10. The images are not evenly distributed through the labels, it has more examples of digits 1, 2, 3 and 4 (13861, 10585, 8497, 7458 images respectivly), this will need to be adjusted so the model doesn't get bias.

Exploratory Visualization

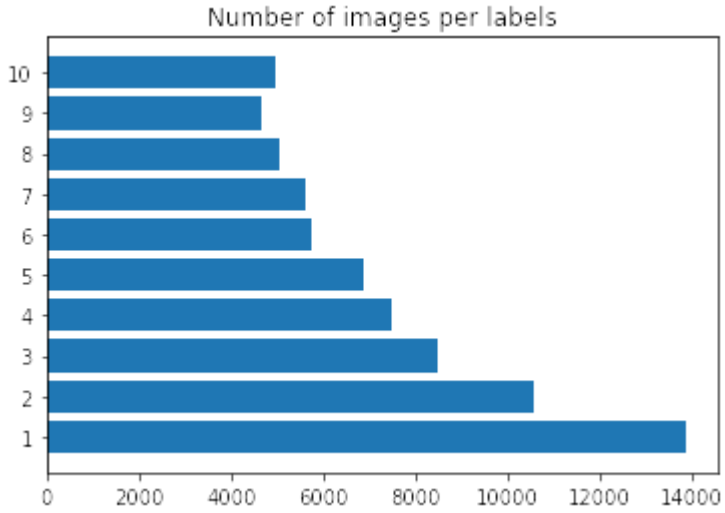
Here is possible to see how the digits are represented in the images.

SVHN dataset example taken from the downloaded data



The graph below show the distribution per label where is possible to see how uneven are the data with the digit 9 being represented in just 4659 images, i.e. just 33% of the images showing digit 1.

Frequency of digits represented in the dataset



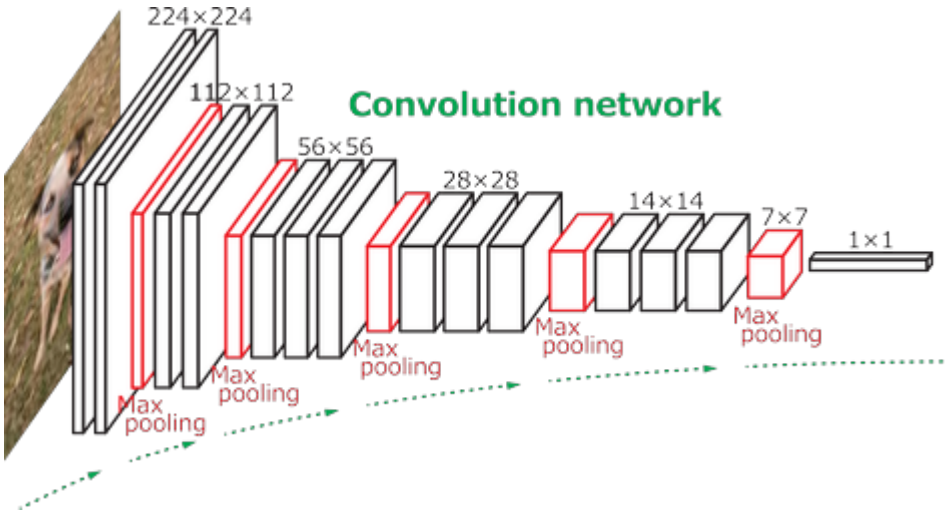
This means that if our model recognize every image as an image of a digit 1 the accuracy would be 18.92%, in other word if the dataset is used in total it would bring a problem because the model

could give a bigger weight to features that appear in images with digit 1.

Algorithms and Techniques

To tackle this problem I will use the VGG16 neural network architecture. This is a combination of 13 convolutional neural network layers and 3 fully connected layers created by Karen Simonyan and Andrew Zisserman (3) and used to win the 2014 ImageNet Large-Scale Visual Recognition Challenge.

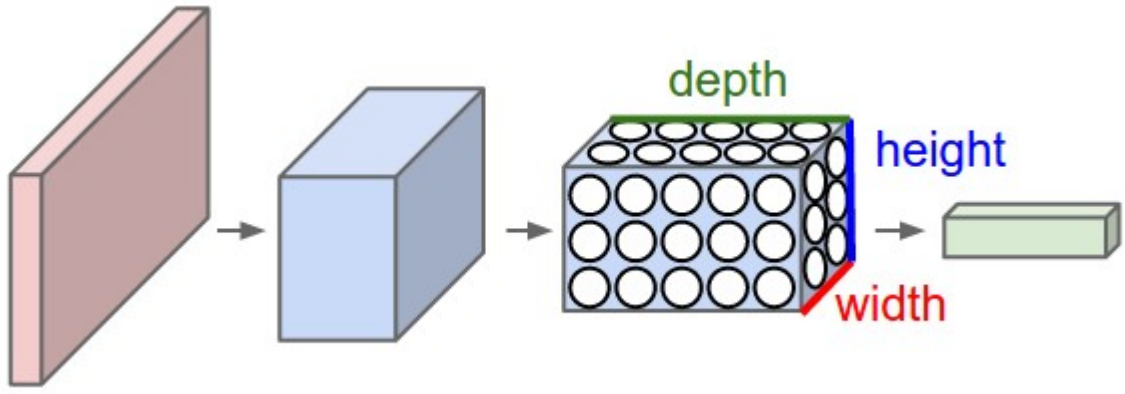
VGG16 layout



In a broader sense the neural network I will use here is a parametric supervised learning procedure that tries to find patterns and correlations using the pixels of the image as inputs to output, in this case, the center digit of the image. It is a parametric procedure because it has a fixed number of parameters, 138 million parameters in this model, and it is a supervised procedure because it will 'learn' how to configure it self by comparing the result it got to the labels pre defined.

A convolutional neural network it is a set of weights and bias that receives part of the input which allows to encode certain proprities into the architecture making the forward function more efficient to implement and reduces the amount of paramenters in the network.

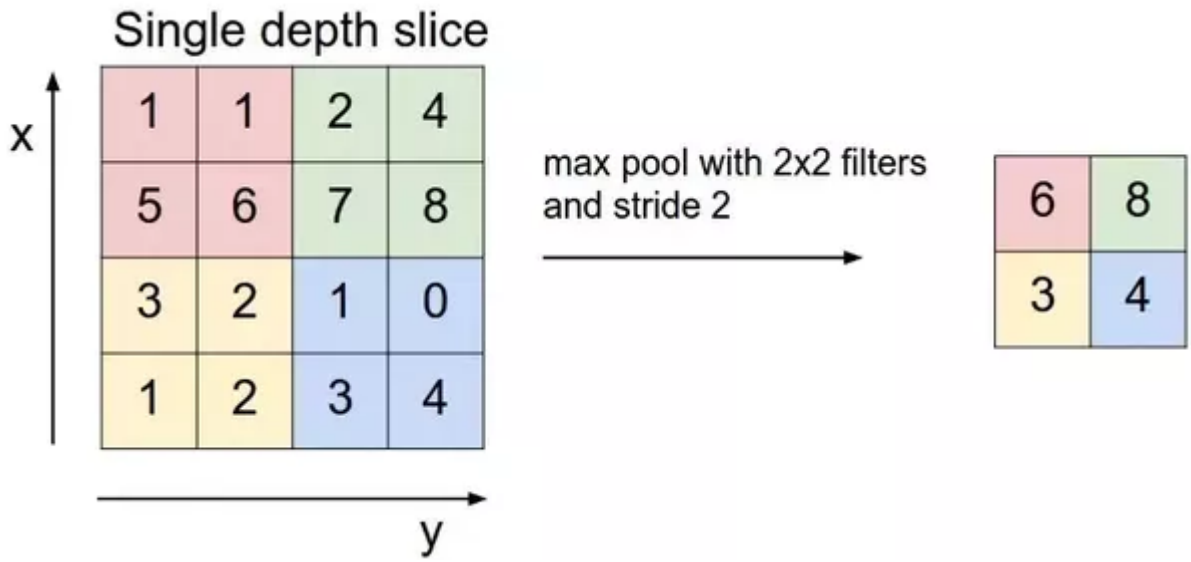
Convolutional neural network 4-layer configuration



A Max-pool will be performed over 2 by 2 pixel window with the stride of 2 pixels to reduce the size of representation and by that reduce the amount of features and the computational complexity of

the network.

Max pooling



To prevent overfitting I also use a regularizer called dropout, that turn to zero some random output from one layer before feeding it to the next in this way the model can never trust fully one feature making necessary to create more ways to output the correct final result.

Unfortunatly training this model take a long time with the use of powerfull GPUs to process all the computation, thus to accomplish this project I'll use a pre-trained model available by Chris on his GitHub page. But in order to use this model I'll have to resize the image to a 224 by 224 pixel, no other preprocessing technique will be used.

Benchmark

Multi-digit Number Recognittion from Street View Imagery using Deep Convolutional Neural Networks(2), Goodfellow et al. achieve over 96% accuracy in recognizing complete street number digit in the same Street View dataset. They also realised that performance increase with deeper networks and to achieve the mentioned accuracy they used a eleven layer network.

Thus I expect that the VGG16 will performe as good as the one used in the Multi-digit paper being able to come around the noise created by the resizing of the images as mentioned before.

III. Methodology

Data Preprocessing

First, because the dataset is uneven, i.e. it has much more samples images of numbers 1, 2, 3 and 4 I'm going to take an equal set of random sample images of the each label so the model doesn't give more weight to features that only apear on images of those digits and overfits. This limits the amount of data that can be used for training, digit 9 has the lowest count, 4659 as said before, but I can't use this number as a parameter because then the model would be trained on all examples of

9 and I wouldn't have any image for testing. Thus I'll use only 10000 images, 1000 per digit, randomly selected.

Another important step is to resize the images so it is possible to use the pre-trained model. The problem here is that it is not possible to resize all the images in advance, for that I create a function that resizes the batch of images before pass them to the model.

From the pre-trained model I took the result of the first fully connected layer to use as input to train my own 3-layer fully connected neural network to recognize the digits, as it was the continuation of the VGG16 architecture. Also the labels, since it is a categorical variable, were one hot encoded.

After all this the input used to train my part of the model consists in 8000 training vectors, 1000 testing vectors and 1000 validation vectors that were used to train and verify the rest of the model.

Implementation

To implement all this first I took the 1000 random sample images for each digit, a total of 10000 images, from the dataset and feed them in batches of 10 images to the pre-trained model, but resizing them before run them through the model, taking the result of the first fully connected layer.

Then I splitted the data into training, validation and test set and start to build the model that it would be trained using those as inputs. As cost function I used the cross entropy function and as optimizer the Adam algorithm, a first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments.

Next I trained the fully connected model and validated it every other 50 batch iterations.

With the trained model I tested it using the 1000 test vectors and finally I took a random image from the original dataset and feed it to the model so it was possible to visualize if the results made sense.

Refinement

Since one of the objectives of this project was to see how well the VGG16 architecture would be with this particular dataset I tried as much as possible to not diverge from the original way that the authors used it with the 2014 ImageNet Challenge.

The architecture used in the ImageNet Challenge can be separete in four groups of convolutinal structure all of them using a 3 by 3 window and 2 of fully connected. The width of convolutional layers (the number of channels) starts at 64 for the first group and grows by a factor of 2 after each max-pooling until it reaches 512. The fuly connected layers starts with two layers with 4096 nodes and the last one is the one changed to fit this problem specificaly. In the original dataset the last layer has 1000 nodes because the ImageNet Challenge had a 1000 different labels, but here, since I'm trying to recognize the digits, it needs to be change to 10 nodes.

Below, it is possible to see the original architecture (model D), where the only difference between

the one used here is the last fully connected layer as commented before.

The original VGG16 architecture

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

IV. Results

Model Evaluation and Validation

With just 10000 image samples the VGG16 turned out to be a very powerful model.

It's accuracy on the test set achieve 88.2% and I only use a little over 13% of the data available. And playing with other sample images it is possible to see that it is a very robust model and it has the capability to recognize the center digit even when the image shows more then one number.

Justification

The results show that this model is very capable and comparing it to the benchmark it is just a short

way behind. But the model builded here still benefits of more data available for training and testing that could bring it more close to the benchmark and it has still to be account that the benchmark didn't used the rescaled image that has more noise and distractions, like more than one digit in the image.

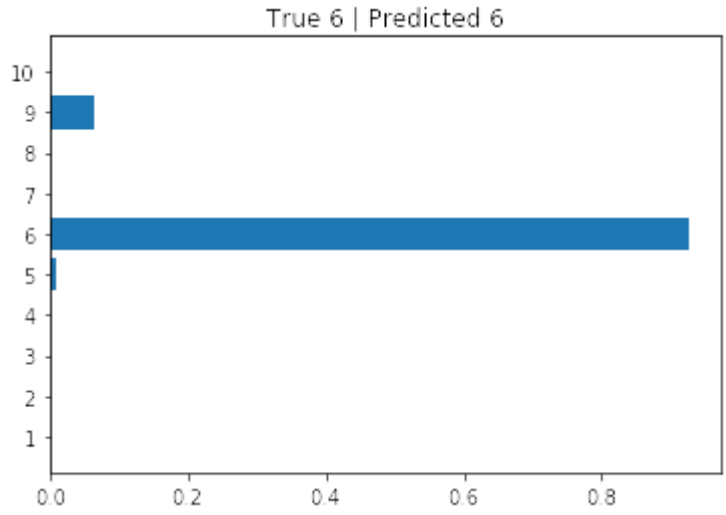
With all that in mind we can say that the model presented as a successfully solution to the problem layed out here.

V. Conclusion

Free-Form Visualization

Here we can see an example of how well the model is capable of recognizing the center digit of an house number. The image below was taken from the dataset and it is possible to see, besides the low resolution, that it consists of 3 numbers, 2, 6, 1, forming the number 261.

Then, passing it through the model, the output are the probabilities of a given number being the center digit of the image according with the model, remembering that the number 0 here represents the digit 0. With this, the graph below shows what was the output of the model for the image above, on the horizontal axis is the probability and the vertical axis the digits.



Reflection

This project was really able to show the capabilities of Deep Learning by showing that even with a very variable and complex dataset it has the power to tackle those kinds of problems even though it needs a powerful hardware to train.

The most difficult part in particularly for me was how to use the resized images because of the storage limitations, thus the solution was to gradually resize the images and process them with the pre-trained model and maybe the overall training time was increased by this process.

Overall the expectations with the model were attended and VGG16 proved to be a very useful architecture that could be used in more broader image recognition problems. Even because the model shown to be able to deal with very different dataset with almost no preprocessing, which is very time consumming.

Improvement

There is a lot more space to improve this kind of solutions. Only the use of more data could improve the results shown here. But also adding more convolutional layers to the model could improve the results, for example using the VGG19. Or it is possible to add the Inception architecure layer to extract more features.

Besides all that it is possible to improve the model by preprocessing more the input before feed it to the model, one way is to normalize the pixels.

But all this come at a cost, being that time to train or increase of the computational resources needed, thus it is a trade off that must be consider.

VI. References

1. The MNIST Database of handwritten digits - http://yann.lecun.com/exdb/mnist/
2. GOODFELLOW, I. J. et al. - Multi-digit Number Recognittion from Street View Imagery using Deep Convolutional Neural Networks - https://arxiv.org/abs/1312.6082
3. SIMONYAN, K. and ZISSERMAN, A. - Very Deep Convolutional Networks for Large-Scale Image Recognition - https://arxiv.org/abs/1409.1556
4. The Street View House Numbers (SVHN) Dataset - http://ufldl.stanford.edu/housenumbers/
5. NETZER, Y. et al. - Reading Digits in Natural Images with Unsupervised Feature Learning - http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf
6. Tensorflow.org - https://www.tensorflow.org
7. VGG in TensorFlow - https://www.cs.toronto.edu/~frossard/post/vgg16/
8. A Brief Report Of The Heuritech Deep Learning Meetup #5 - https://blog.heuritech.com/2016/02/29/a-brief-report-of-the-heuritech-deep-learning-meetup-5/
9. Adam: A Method for Stochastic Optimization - https://arxiv.org/abs/1412.6980
10. Grokking Deep Learning - https://www.manning.com/books/grokking-deep-learning