Machine Learning Engineer Nanodegree

Capstone Proposal

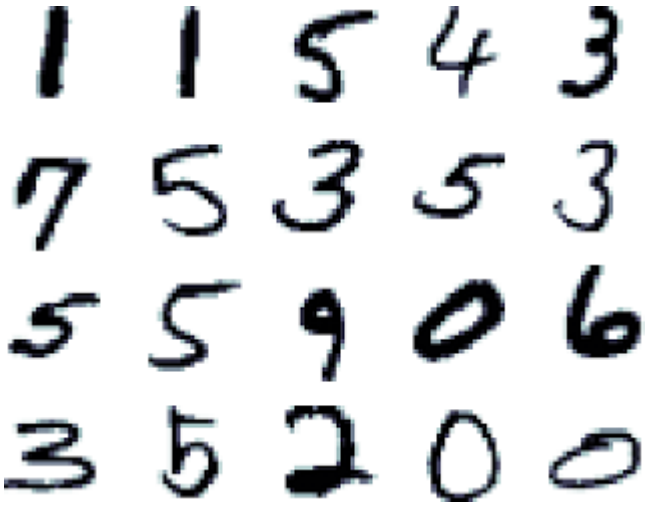
Daniel Resende
March 28th, 2017

Proposal

Domain Background

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 70000

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

Datasets and Inputs

So in order to attack and understand this real-world problem I'll use the Street View House Number(4) dataset, a benchmark dataset introduced by Yuval Netzer et al.(5) that comes in two flavors: (1) with original full numbers, variable resolution color house-number; and (2) in a character level MNIST-like format resized to a fixed 32-by-32 pixels resolution.

As said before, I will use the MNIST-like of the Street View House Number(2) dataset, because it can be consider is the next step from the MNIST(1) dataset. This is an open dataset that have been taken from the original house numbers character bounding boxes and extended in the appropriate dimension to become square windows, so that resizing them to 32-by-32 pixels does not introduce aspect ratio distortions. But still this preprocessing introduces some distracting digits to the sides of the digit of interest.

I'll use the `train_32x32.mat` file that contains 73,257 images with labels ranging from 1 to 10, with 10 corresponding to a 0 digit, that will permit me to use 25% of the dataset as test, 18,315 images and 54,942 for training (10% of the training dataset will actually be used for validation during training).

Solution Statement

Since Yann LeCunn's Le Net, deep learning has been established as a go to way of handling image recognition problems. Thus I am going to use a deep convolutional neural network architecture known as VGG16 to see if it can surpass the distractions and difficults of the real-world characters presented in the dataset.

Simonyan and Zisserman state that incresing depth using an architecure with very small convolution filters can significantly improve the results comparing to smaller versions, resulting in the winning architecture of the ImageNet Challenge 2014. Thus it is expectable that this model can be effective.

To help the neural network figure out which features, in this case pixels, give the best information and are more important for reconizing the center digit, I will preprocess the data to normalize and stand out the features that are most useful. This is a very important step because with raw data the model can diverge, due the large values of the pixels (original they may be from 0 to 255), not being able to learn the proper weights values during training(10).

Benchmark Model

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.

Evaluation metrics

Since it is a classification supervised problem, to evaluate the model and assess if it learn 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.

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.

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.

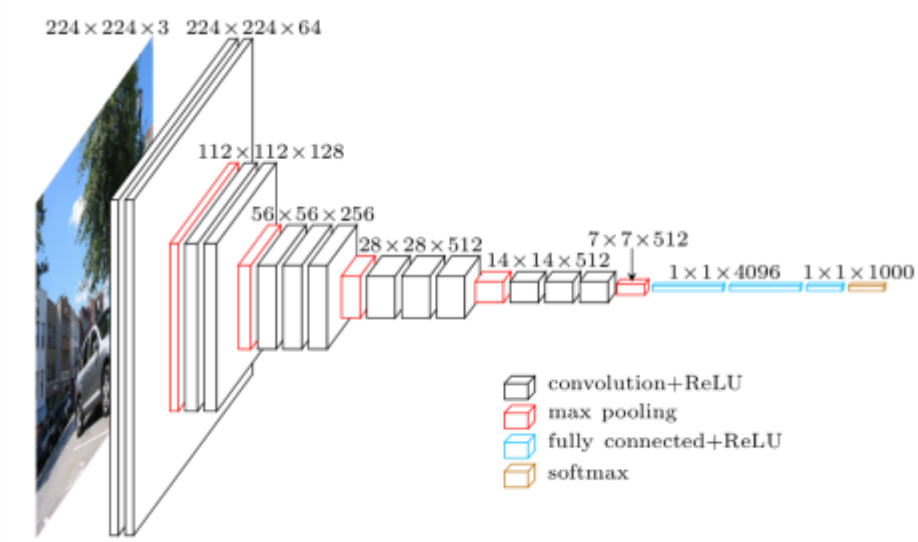
Project Design

To put in a better perspective this project will require downloading the data from the dataset website first. I'll use only the `train_32x32.mat`, because this file has already 73,257 images what I belive is more than enough to do this project. This file is a *.mat* file, to use it with Python will read it using the Scipy module.

Those images are, as said before, a 32-by-32 pixel color images with pixels ranging from 0 to 255 for each color, thus will need to normalize the data before using our neural network putting the pixel values between 0 to 1. Also the labels come in a numerical form ranging from 1 to 10, with 10 being 0 in the picture. So I'll one-hot enconde the labels so our model can predict the probability of an given image be of a number.

For better generalization, I'll split the data in three parts: train, validation and test. The first I'll user to train the data and find the optimal weights validating along the way with the validation set. Finnaly after training I'll use the test data to see if the model was able to learn and generalize to images that it never have seen before. This step is very important because the model can easily overfit and learn only how to recognize the train data. Keeping in mind that with 10% accuracy the model is just randomly guessing and not learning at all.

To recognize the image's number I'll use the VGG16 architecture, that consist in a 13 layer convolutional neural network with 3 fully connected neural network. For the desing of this architecture I will use Tensorflow6, building first the nodes that I'll be using to assemble the graph and, then, build the architecture.



To compute the loss and give the model tha ability to learn, I will use softmax with an Adam optimizer, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments(9).

Finally, will test it to see how well it does with new unseen images, hopping to get an high accuracy with this technic.

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