MACHINE LEARNING ENGINEER NANODEGREE

Capstone Proposal: Multi-Digit Classification with TensorFlow

Harley Jackson

May 8, 2017

Domain Background

Computer vision is a field of elevated importance, now that technology such as self-driving cars, delivery drones, and warehouse inventory robots are fast approaching scales of mass adoption and production. Within the field of computer vision, recognizing multiple digits from pictures or live visual input, has utility in real-world applications. For example, in the case of delivery drones, while the drones would probably locate delivery points with GPS, being able to read addresses on doors, mailboxes, and buildings could ensure that the drone is delivering its payload to the correct unit, more precisely than just GPS coordinates taken from an address database. As Goodfellow, et al, point out in their 2014 paper, in which a multi-digit classifier was trained on the SVHN dataset, recognizing address numbers from photos is also important to mapmaking, in order to automatically process the majority of street-level photographs [1].

My motivation for pursuing this problem is to gain more experience implementing convolutional neural networks using TensorFlow, especially in order to more deeply understand the effects of hyper-parameter tuning, and to gain more insight on how the deep learning network should be adjusted to suit different types of problems. Generally, deep learning has a wide range of applications, including natural language processing [2] and protein structure classification [3] in addition to computer vision, so having expertise in deep learning is valuable as a machine learning engineer. Beyond the machine learning nanodegree, I intend to continue developing skills in deep learning, including Kaggle competitions and other projects, so knowledge gained from the project is valuable to my professional development.

Problem Statement

The problem is to take a photograph depicting numbers up to 5 digits long and correctly recognize how many digits are present, up to 5 (length = 0, 1, 2, 3, 4, 5), and what each present digit is (n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, or BLANK). The classifier should only get credit for getting the entire number correct, i.e. predicting each of five digit placeholders correctly to be considered a correct classification, for accuracy measurement purposes. Once the model is trained, it should be able to accept photographs from an Android phone or other sources and classify multi-digit numbers depicted on them.

Datasets and Inputs

The first dataset that will be used is the MNIST database of handwritten digits [4]. The MNIST dataset is an appropriate starting point for this project, because it consists of 70,000 images of hand-written digits, from 0 to 9, split into 60,000 training images and 10,000 testing images. Each MNIST image is 28 x 28 pixels, with a label that can be one of 10 classifications (digits 0 through 9). It seems plausible that a model that can classify these images at a high degree of accuracy can be scaled up to classify multiple digits.

The next dataset to be used is the Street View House Numbers (SVHN) dataset [5]. The SVHN dataset is a database of over 600,000 digit images taken from Google Street View images, divided into 73,527 images for training, 26,032 for testing, and 531,131 extra images that can be used for additional training. Character level bounding boxes are included, along with labels that specify length in number of digits, and values for each digit. Since these images possess both the characteristics of being real photographs, and containing multi-digit numbers, a model trained to accurately classify on this dataset would presumably be able to correctly classify most pictures taken of numbers.

The majority of images in the SVHN dataset are from 1 to 3 digits, with 4 and 5 being less common, in declining order. It means that, for example, a high percentage of accuracy could just reflect classifying the numbers with less digits, and few or none of the 4 or 5-digit numbers. To address this imbalance in the data, a weighted penalty on the less common digits will be used in the loss function. As a secondary measure, oversampling or undersampling will be used. After building a model that can classify the SVHN images, the model will be further tested on additional images, taken from Google images and taken with an Android phone by the researcher.

Solution Statement

The solution will take the form of a convolutional neural network which can input 32x32 images of up to 5-digit numbers and classify them, including the correct value of each of 5 digit placeholders. Its evaluation will be carried out as outlined in the "Evaluation Metrics" section below.

Benchmark Model

The benchmark model for this project is the multi-digit classifier developed by Goodfellow, et al, in 2014 [1]. The model is a deep learning neural network that utilizes 11 sequentially arranged hidden layers, which are shared by 6 separate classifiers, one for the length and five for each of the digits. Of the 11 neural network layers, 8 were convolutional layers which used 5x5 patches, 2x2 max pooling with strides alternating between 2 and 1, same padding, and RELU rectifiers, except for the first hidden layer, which uses a 3x3 maxout activation. The other 3 layers include 1 locally connected layer and 2 fully connected layers. All layers use dropout, except for the input. Through using this deep network, Goodfellow, et al, achieved 96.03% accuracy – that is, the model accurately classified all length and digits correctly on 96.03% of images - which was just short of their 98% accuracy goal.

Evaluation Metrics

The evaluation metric of the classifier developed in this project will be similar to that used to measure the benchmark model. More specifically, the model will only get credit for accurately classifying an image if all 5 of its classifiers return the same label as what the picture actually depicts. So, each of the five digit classifiers should register either the digit present or the lack of a digit (meaning 11 possible classifications for each digit placeholder). The accuracy can be represented as:

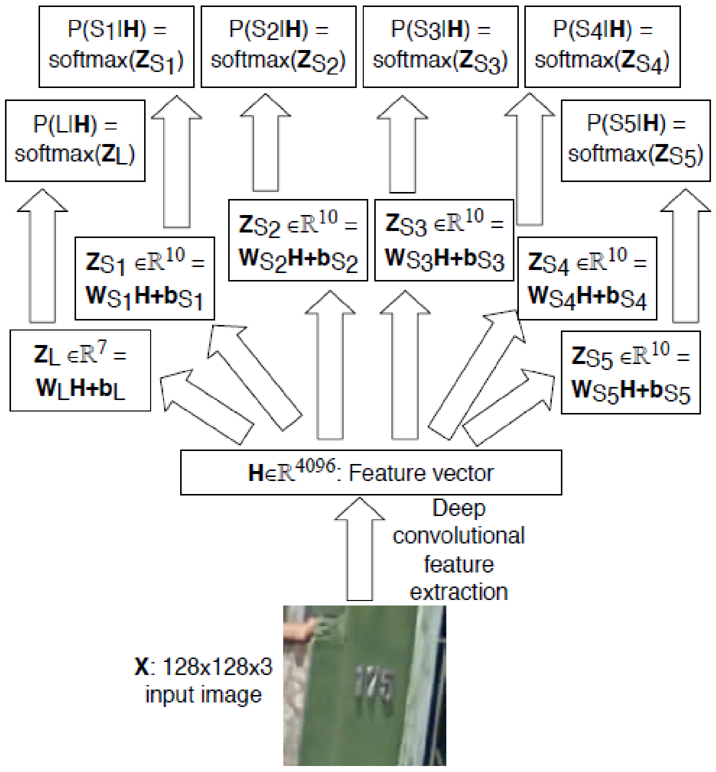
$$\frac{\sum I_{correct}}{\forall I}, \text{ where } I_{correct} \text{ is an image where}$$

Predictions(Digit1,Digit2,Digit3,Digit4,Digit5) == Labels(Digit1,Digit2Digit3,Digit4,Digit5)

and I represents images in the data input to the classifier

Project Design

The project will start by using the MNIST TensorFlow tutorial to train an established convolutional deep learning network on MNIST data [6]. After the model has been trained on the MNIST data, it will be incrementally improved to gain accuracy on classifying the SVHN dataset, first by just accurately classifying the length, then by adding digit classifiers as well. The Goodfellow benchmark model will be used as a guideline for ways to iteratively improve the model and increase performance. Specifically, the project will employ the benchmark method of using sequential layers, a combination of convolution layers, a flat layer, and fully connected layers, all shared by multiple classifiers (5 in this project, as opposed to 6 by Goodfellow). But each of the classifiers will have their own weights to be trained independently, connecting their logits to the shared hidden layers. An overall mapping of the Goodfellow model can be seen in the figure below [1]:



Computing will be carried out on a free-trial virtual machine instance on Google Cloud, using Debian/Linux, TensorFlow, and Jupyter Notebook with Python 3.5 (due to TensorFlow requirements). In light of computing limitations, images will be preprocessed as size 32x32, and 1 channel will be attempted, 3 if necessary. The model will also attempt to achieve 90% accuracy with less layers than the Goodfellow model. After testing accuracy with the SVHN test set, images from Google Images and pictures taken by the researcher will be used to test the model's accuracy on photos not included in the SVHN dataset.

References

[1] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In Proc. ICLR, 2014.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. ICLR Workshop, 2013

[3] Wang S. Peng J. Ma J. Xu J. Protein secondary structure prediction using deep convolutional neural fields Scientific Rep. 2016 6 18962

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, 86(11):2278-2324, November 1998

[5] 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*.

[6] https://www.tensorflow.org/get_started/mnist/pros