**1. Definition**

**Large-scale Identification of Multiple Digits**

**From Real-World Images with Deep Convolutional Networks**

Ritchie Ng

National University of Singapore

ritchieng@u.nus.edu

**1.1 Overview**

This project explores how deep convolutional neural networks (ConvNets) can be used to effectively identify a series of digits from real-world images that are obtained from “The Street View House Numbers (SVHN) Dataset” (Netzer, Wang and Coates). ConvNets have evolved dramatically every year since the inception of the ImageNet Challenge in 2010.

A proverbial ConvNet structure is the “LeNet-5” that has relatively few layers of convolutions, poolings, and full connections (LeCun, Bottou and Bengio). Subsequently, with the advent of the ImageNet Challenge, we are experiencing a gradual trend towards deeper ConvNets with more layers and higher accuracy such as AlexNet (Krizhevsky, Sutskever and Hinton), ZFNet (Zeiler and Fergus), VGGNet (Simonyan and Zisserman), GoogLeNet (Szegedy, Liu and Jia), and the ResNet (He, Zhang and Ren) being the latest state-of-the-art implementation of ConvNets.

To this point, I used VGGNet’s (Szegedy, Liu and Jia) framework as a base where I made modifications to the fully connected layers to suit our problem of identifying multiple digits and I optimized the hyperparameters to determine my optimal model for identifying multiple digits from real-world images.

**1.2 Problem Statement**

I am attempting to predict a series of numbers given an image of house numbers from the SVHN dataset. An important thing to take note is that instead of the standard identification of numbers, as with the MNIST dataset, I now need to correctly detect the numbers and the sequence of numbers.

**1.3 Metric**

***Accuracy***

**2. Analysis**

2.1 Data Exploration

2.2 Exploratory Visualization

2.3 Algorithms and Techniques

2.4 Benchmark

3. Methodology

3.1 Data Pre-processing

**3.2 Implementation**

***Programming Language and Libraries***

Our programming language of choice is Python 2.7 and we will be using Keras with the backend as TensorFlow to build our deep ConvNets.

***ConvNet Topology: Trial 1***

In this first trial, I started off with a simple model comprising the following layers.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Layer**  32 x 32 x 1  *Images with 32 x 32 dimensions with 1 channel (grey scale)* | | | |
| **Convolution: 1**  Filters: 32  Receptive fields: 3 x 3  Stride: 2  Padding: Valid | | | |
| **Rectified Linear Unit (ReLU) Activation** | | | |
| **Max Pooling (Sub-sampling)**  Filter: 2 x 2  Stride: 2  Padding: Valid | | | |
| **Convolution: 2**  Filters: 64  Receptive fields: 3 x 3  Stride: 2  Padding: Valid | | | |
| **ReLU Activation** | | | |
| **Max Pooling (Sub-sampling)**  Filter: 2 x 2  Stride: 2  Padding: Valid | | | |
| **Convolution: 3**  Filters: 128  Receptive fields: 3 x 3  Stride: 2  Padding: Valid | | | |
| **Dropout**  Keep Probability = 0.5 | | | |
| **Fully Connected (FC) Layer**  Nodes: 1024 | | | |
| **Softmax**  **Activation** | **Softmax Activation** | **Softmax Activation** | **Softmax Activation** |

***Machine Specifications: Trial 1***

The model’s computations are accelerated with a CUDA-enabled GPU on Amazon Web Service (AWS) with a g2.2xlarge EC2 instance and a 60 GB EBS volume attached to it.

**3.3 Refinement**

***ConvNet Topology: Trial 2***

I adopted VGGNet’s (Szegedy, Liu and Jia) topology. In particular, I chose the ConvNet configuration with 16 weight layers also called VGG16 and added 15 more weight layers (5 digits x 3 full-connected layers) to predict up to 5 digits. Without this modification, we are unable to predict 5 digits.