The Street View House Numbers Identification

Team Members:

Sushaanth Srirangapathi (ssrira7@uic.edu)

Kirti Sanchana (ksanch2@uic.edu)

Kanika Mohpal (kmohpa2@uic.edu)

Problem Description

The Street View House Numbers (SVHN) is a real-world image dataset for developing machine learning and object recognition algorithms with minimal requirement on data preprocessing and formatting. It can be viewed as similar in flavor to MNIST (e.g., the images are of small cropped digits), but incorporates an order of magnitude more labeled data (over 600,000 digit images) and comes from a significantly harder, unsolved, real world problem (recognizing digits and numbers in natural scene images). Our motivation is to identify the numbers from various Street View images which would immensely help a person looking for a address/house on a particular street. SVHN dataset is obtained from house numbers in Google Street View images. From the given dataset, we try to pre-train the model and then try to predict the class labels of the digits from 1 to 10.

Data Description

Given data consists of 73257 images of digits for training, 26032 images of digits for testing and an additional 531131 images of digits containing somewhat lesser complex samples of extra training data. The dataset comes in 2 formats:

1. Original images with character level bounding boxes – These are the original, variable-resolution, different color house-number images with character level

bounding boxes, as shown in the examples images below. (The blue bounding boxes here are just for illustration purposes. The bounding box information are stored in digitStruct.mat instead of drawn directly on the images in the dataset and can be loaded using Matlab.)

2. MNIST-like 32-by-32 images centered around a single character - Character level ground truth in an MNIST-like format. All digits have been resized to a fixed resolution of 32-by-32 pixels. The original character bounding boxes are extended in the appropriate dimension to become square windows, so that resizing them to 32-by-32 pixels does not introduce aspect ratio distortions. However, this preprocessing introduces some distracting digits to the sides of the digit of interest. Loading the .mat files creates 2 variables: X which is a 4-D matrix containing the images, and y which is a vector of class labels. For convenience, we would be using this dataset for our project.

Proposed Model, Algorithm & Techniques

We have divided the project into three main phases:

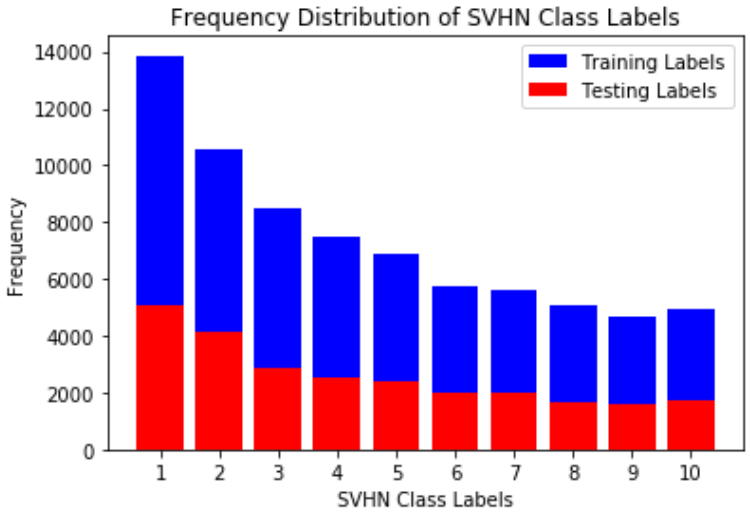
1. In the first phase, we would be implementing feed-forward neural network on the given dataset using TensorFlow library. This phase has been completed and the notebook file containing the code is being submitted. We have prepared the dataset such that it is compatible with the TensorFlow framework by transforming the dependent variable using one hot encoding. The resultant vector of class labels has 10 columns (each column corresponding to a class). As we can see from the output, the accuracy is only 15% as we have not optimized the number of epochs (only 150) and not normalized or preprocessed the data. This forms the base model and reference for the accuracies obtained during the future phases.
2. During this phase, we plan to improve the code developed in phase 1 by applying multi-layered convolutional neural networks including max-pooling, fully-connected and dropout layers. We also would normalize and pre-process the data before applying the various models.
3. As part of this phase, we want to further enhance the accuracy by tweaking the various hyper-parameters and try out different non-linearities including Relu, LeakyRelu and Maxout. We also would be analyzing the model by comparing and checking how the training and testing accuracies would vary w.r.t. number of epochs, batch sizes, learning rates, and number of iterations. Based on all these criteria, we would be choosing the best model.

Dataset

From the above matrix of SVHN images, we can notice the following features:

- Images and/or digits/numbers are multi-colored
- Images are viewed or taken from different angles and in some cases the digits are at multiple angles w.r.t. other digits
- Some images contain random objects at the edges which can be treated as noise
- Fonts of the digits vary across the images
- Few of the images (e.g. below image) are very blurred to not be recognized even by humans.

As we can observe from the below frequency distribution plot of the training and testing data w.r.t. the SVHN class labels the digit 9 is the least common to appear in the images among the rest of the digits and the dataset is biased towards digit 1.



References and papers for Detailed Study

1. 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
2. http://ufldl.stanford.edu/housenumbers
3. Convolutional Neural Networks applied to House Numbers Digit Classification by Pierre Sermanet, Soumith Chintala and Yann LeCun
4. CIFAR-10 tutorial to demonstrate Convolutional Neural Network
5. CS231n -Convolutional Neural Networks for Visual Recognition

These are the list of references that we have studied so far. We will update this list as we progress through our project implementation.

Possible Experiments and Metrics of Success

As mentioned in the proposed phases of building and choosing the model, we plan on reducing the loss and increasing the accuracy of the identifying house numbers classification by using deep networks with different non-linearities and search optimal hyper parameters. We would also be using the extra images to train and validate the model and thereby boosting the accuracy on the testing dataset.

We will be using mean squared error as our performance measure for reducing loss. We plan on trying different optimizers provided by TensorFlow for backpropogation. The best model would help us classify the house numbers with highest possible accuracy. From our online research, we notice that the best accuracy achieved by CNN architecture is 94.85% vs Human performance of 98% [3]. We are aiming to achieve testing accuracy in the range of $85 - 95\%$.

Results Obtained so far

We have implemented feed forward neural network with 0.01 learning rate and 150 epochs without normalizing the data. This resulted in an average accuracy of 15%. The code for this implementation can be found on GIT at the following link:

https://github.com/sushaanths/DeepLearning˙Project

Possible Conclusions from Phase 1

We have observed low accuracy (15%) in the existing model of Feed Forward Neural networks from phase 1, which we plan on improving in the future phases by implementing the following features:

- Normalization of the data
- Optimizing hyperparameters
- Deep Network architecture
- Different non-linearities