Data Augmentation - SVHN

Introduction

When training deep neutral networks, the availability of data is a frequent challenge. Acquisition of additional data is often difficult, due to one or both of logistical or financial reasons. As such, methods such as fine tuning and data augmentation are common practices to address the challenge of limited data.

Dataset Description

The dataset supplied is a subset of the original Street View House Numbers (SVHN) testing dataset. The training dataset `Q3/q3_train.mat` contains 1000 samples, and testing dataset `Q3/q3_test.mat` contains 10,000 samples in total.

Data Preparation

Both training and testing dataset are loaded from mat files provided. The `train_X` variable contains the images data and index. The `train_Y` variable contains the corresponding class, values 1 to 10.

Class Imbalance

There is a class imbalance in both training and testing datasets. However, the classification distributions in the two datasets are similar. This issue needs to be addressed when we fit the models.

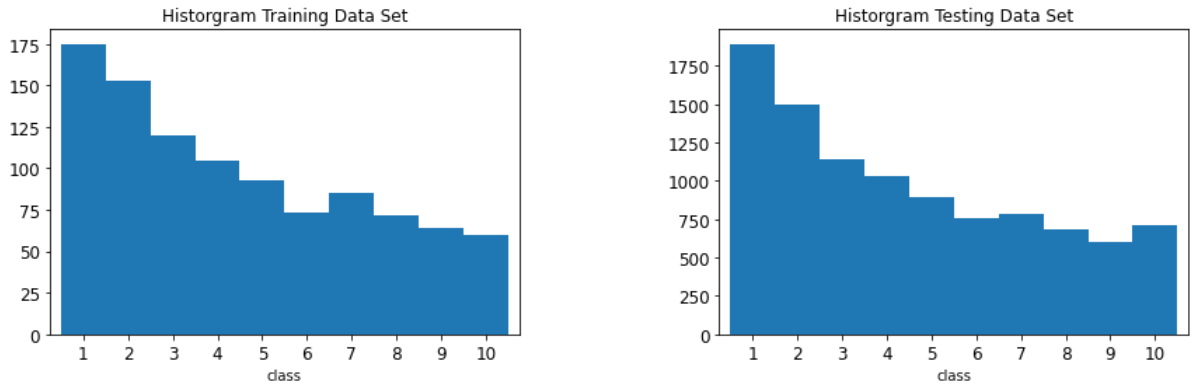

Figure 3.1: Distribution of classification in datasets

Classes

There are 10 classes, 1 for each digit. Digit '1' has label 1, '2' has label 2, and so on. However, as Python is zero-index, label 10 has been transformed to '0'.

Model 1 – No Data Augmentation

We build a simple convolutional neural network with 3 convolutional layers. In summary:

- Input is 32 x 32 x 3.

- First layer is 8 – 3 x 3 filters
- Second layer is 16 – 3 x 3 filters
- Third layer is 32 – 3 x 3 filters
- Dense layer is 64 units
- Output is 10 units.

There are max pooling layers with size of 2 x 2 in between each convolution layers. More details of the model is summarised in Figure 3.2.

```
Model: "model_baseline"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
img (InputLayer)             [(None, 32, 32, 3)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 32, 32, 8)         224
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 8)         584
_________________________________________________________________
batch_normalization (BatchNo (None, 32, 32, 8)         32
_________________________________________________________________
activation (Activation)      (None, 32, 32, 8)         0
_________________________________________________________________
spatial_dropout2d (SpatialDr (None, 32, 32, 8)         0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 16)        1168
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 16)        2320
_________________________________________________________________
batch_normalization_1 (Batch (None, 16, 16, 16)        64
_________________________________________________________________
activation_1 (Activation)    (None, 16, 16, 16)        0
_________________________________________________________________
spatial_dropout2d_1 (Spatial (None, 16, 16, 16)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 16)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 32)          4640
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 8, 8, 32)          9248
_________________________________________________________________
batch_normalization_2 (Batch (None, 8, 8, 32)          128
_________________________________________________________________
activation_2 (Activation)    (None, 8, 8, 32)          0
_________________________________________________________________
spatial_dropout2d_2 (Spatial (None, 8, 8, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 256)               524544
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448
_________________________________________________________________
dense_2 (Dense)              (None, 10)                650
=================================================================
Total params: 560,050
Trainable params: 559,938
Non-trainable params: 112
_________________________________________________________________
```

Figure 3.2: Model Summary

The model is trained on SVHN for 100 epochs. After 100 epochs the training accuracy is 74.5% and test accuracy is 65.79%. However, the model does not work well during evaluation, in fact it is very poor in predicting number 6 and 8. (Fig 3.2) But after rebuilding the model with class weighting assigned, the evaluation test turns out to be much better, with a test accuracy of 79.44%, and an improvement in predicting number 6 and 8. (Fig 3.3)

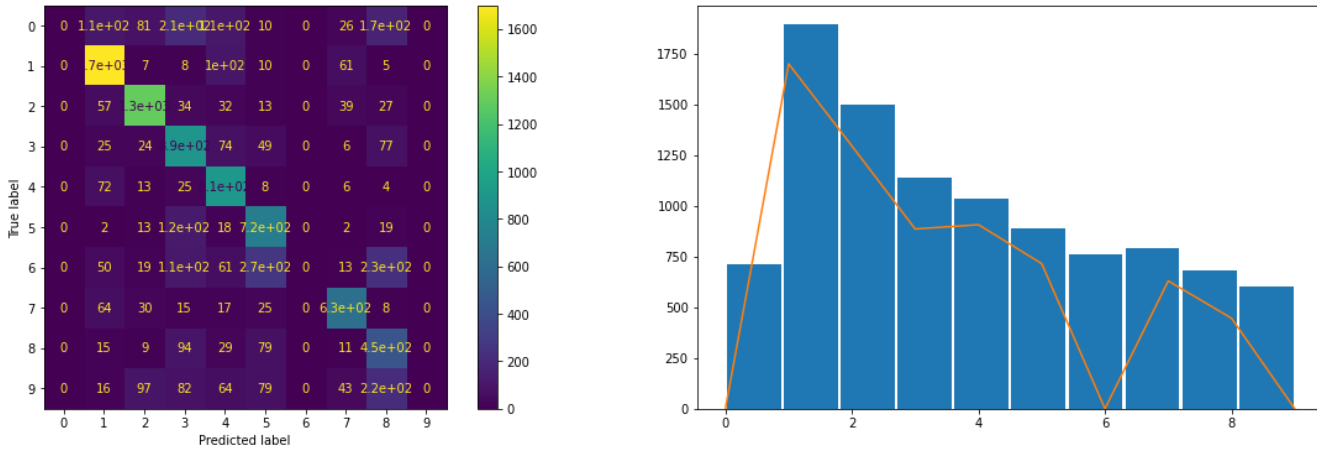

Figure 3.3: Evaluation: No Data Augmentation and No Class Weight

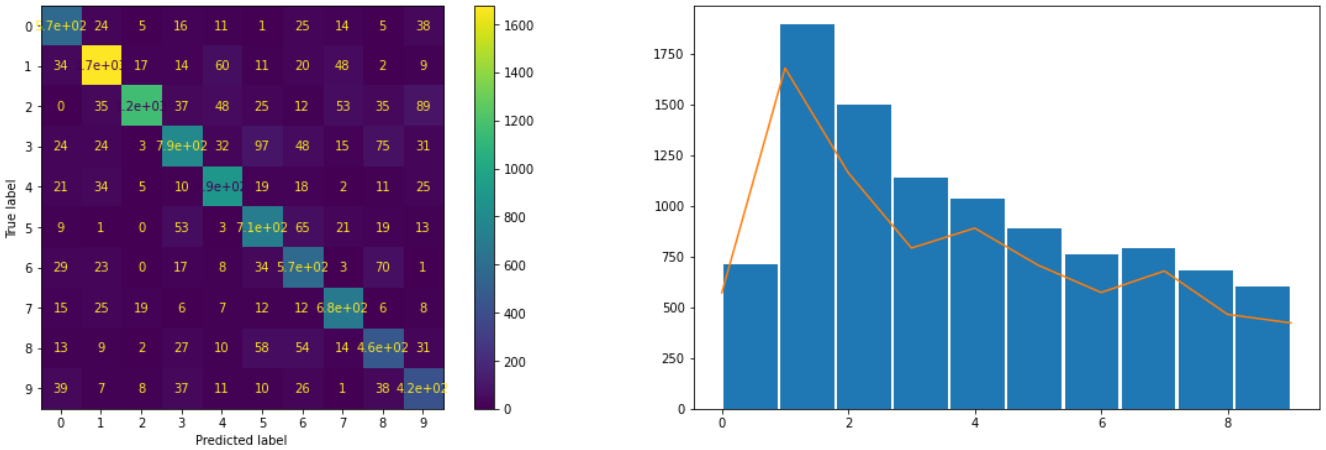

Figure 3.4: Evaluation: No Data Augmentation and Class Weight

Model 2 – With Data Augmentation

For Model 2, we apply the same convolutional neural network with more images. The intent is to expand the training dataset with new samples. 2,000 new images are generated from existing image sets by:

- Random small rotation: between -5 to 5 degrees.
- Horizontal/vertical shift: +/-5% of image width
- Zoom range: 0.1
- Fill mode: Nearest
- Brightness Range: 0.5 – 1.0

It is important to note that horizontal and vertical flips are deemed to be inappropriate, due to some numbers are not horizontal and/or vertical symmetrical which certainly is meaningless in our context and will not help in model training.

The test accuracy has reached >80% to 81.94%, with a further improvement on predicting number 8 correctly.

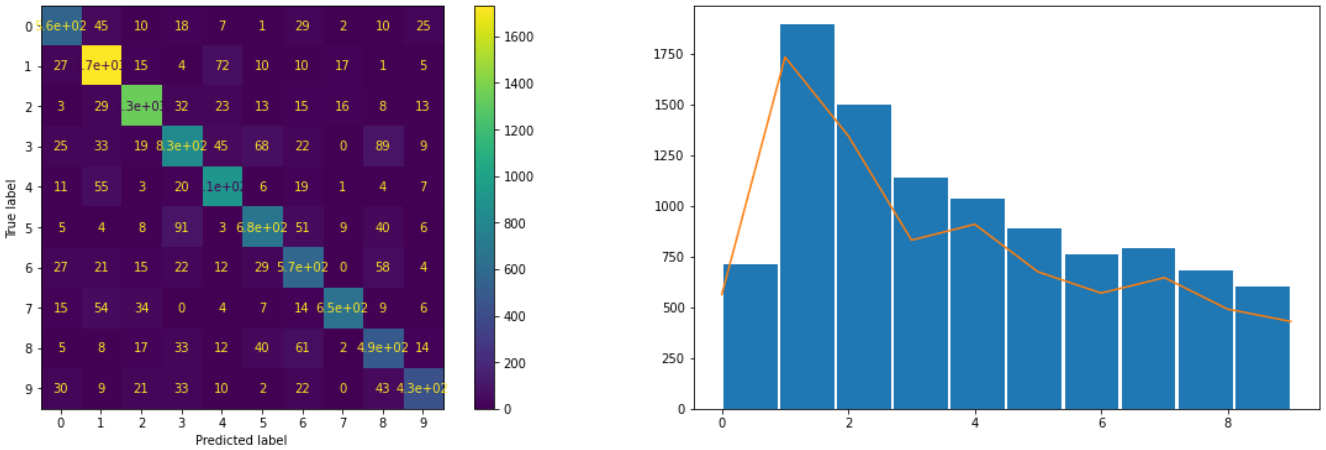

Figure 3.5: Evaluation: With Data Augmentation and Class Weight

Model 3 – Existing Trained Model

In CIFAR-10, images are in the size of 32x32x3 (32 wide, 32 height, 3 colour channels) which matches the image size of our dataset. Another reason of selecting CIFAR-10 VGG (`vgg_3stage_CIFAR_bigger.h5`) is that we hope to leverage the VGG trained model to extract richer features in a lesser time.

Since the network has already been trained, we will only make some minor fine-tuning adjustments when we train the network on SVHN dataset. We only train the last 5 layers of the network including the dense layers. In addition, we undertake small learning rate by setting the epoch to be 20 for training SVHN dataset and 10 for training data augmentation dataset. This is the same augmented dataset generated during Model 2 process. This would allow performance of both model 2 and 3 to be compared and evaluated.

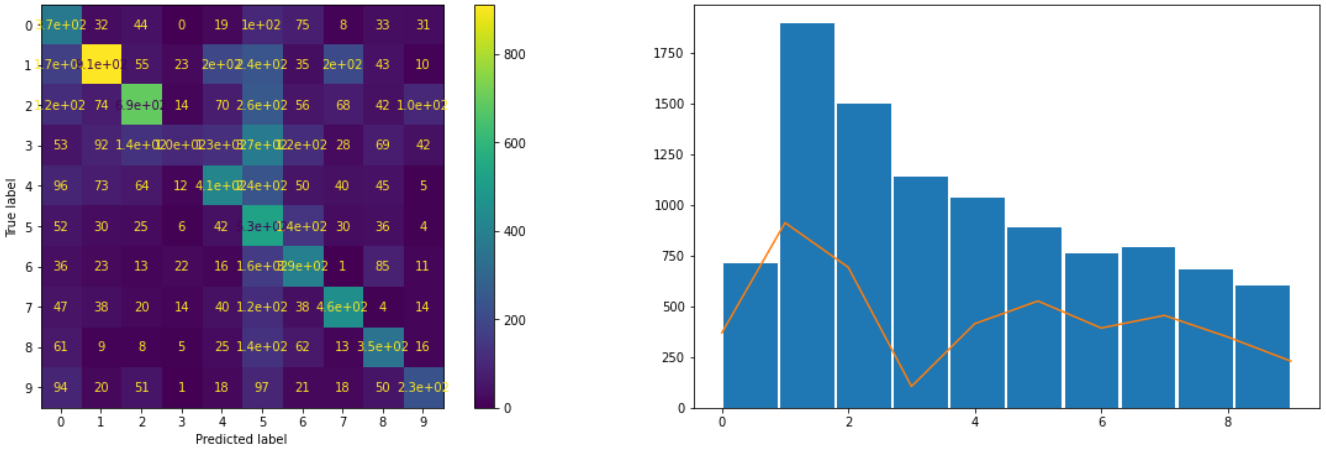

Figure 3.6: Evaluation: CIFAR-10 VGG Network

Evaluation

The performance of the 3 models on testing set is summarised as below:

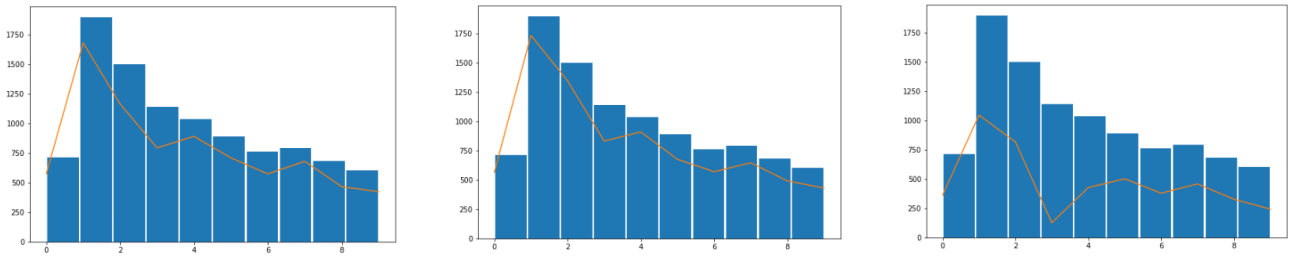

Figure 3.7: Prediction Performance (from left): Model 1, 2 and 3.

According to our testing, model 2 (with data augmentation) performed better than the other two models. Having more samples to train, model 2 has overcome some of the weakness in predicting some numbers, noticeably number “2”.

Surprisingly our CIFAR-10 VGG performed quite poorly unexpectedly. However, this may be because it’s trained for a different domain than SVHN, or more training is needed.