Humanware Block 2

Alexandre Marcil
Montreal Institute of Machine Learning
University of Montreal
Montreal, Quebec

Hugo Lafortune Brunet
Montreal Institute of Machine Learning
University of Montreal
Montreal, Quebec

1 Introduction

Humanware is a company that helps people with vision loss. Our task is to help them develop a solution that can detect and read text in natural scene. This can help blind people know what is the text on bus stops, house numbers and storefronts, thereby improving their quality of life. More specifically we were task to find the sequence of numbers in house number pictures.

In this report we present the various solutions to the problem that we have explored. The image preprocessing and sequence length prediction was done in block 1. In block 2, we need to predict both the digits and the sequence length. We try several Convolutional Neural Network (CNN). CNN is a variant of the feedforward neural network designed to give good performance in image analysis. Our best model follow the architecture used in Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks (Goodfellow et al, 2013) [1]. We used Bayesian hyperparameter optimization to find the best hyerparameters. In Bayesian hyperparameter optimization a probability model of the objective function is build and use to select the most promising hyperparameters. We also used TensorboardX to visualize our results and learning curve. Finally, we implemented checkpoints in order to be able to restart training in case of interruption.

2 Methodology

2.1 Data description and Data Selection

The data that we used is the Street View House Numbers (SVHN) [3]. SVHN is a dataset of real-world street house number pictures taken from Google street view. There is about 200k pictures split between a training set, an extra set and a test set. For each pictures the bounding boxes of each digits is available and is used in the image preprocessing step. In our project we used the training set to find the best architecture, that was then trained on both the extra and training data set in order to get our final model.

We used 90% of the train data (images) for our training dataset and kept the remaining 10% (images) for validation. The extra has x images. After the prepossessing each image is 54x54 and therefore each data point is 3x54x54 since the images are in RGB. The given task is to predict the length of the sequence and the digits composing this number. For the length there is only one value to predict. The length varies between 1 and 5 however we used 7 classes to also include no digit and more than 5 in order to be consistent with [1]. For the digits we always predict the value of 5 digits even if there is fewer in the picture. The prediction of irrelevant digits is simply ignored. For each digits, there is 10 possible classes corresponding to all possible digits (0 to 9).

2.2 Description of the Algorithms and the Experiments

Preprocessing

For the preprocessing, we used the code developed during Block 1 (baseline from the teaching assistants). The data preprocessing follows the the steps described in Goodfellow and al. 2013 [1]. It

makes use of the individual digits bounding box coordinates to find a rectangle bounding box that can enclose all the digits. The size of this rectangle is increase by 30% and the image is cropped around the expanded rectangle. The resulting image is then resized to 64x64. For data augmentation a random crop of 54x54 pixels is performed. This random crop changes every epoch to help the model remain invariant to image translations.

Algorithms

We approached the problem with various models in mind. We tried several popular architectures: ResNet18, ResNet34, VGG19 and the CNN used in [1] with a custom head. The custom head consists of 6 separate fully connected layers (1 for the length and 1 for each digits) with a softmax activation function. A cross-entropy loss is calculated for the sequence length and the digits present in the image. The loss that we are trying to minimize is the sum of these individual losses.

For the body of the model:

ResNet (He et al., 2016) [2] is a network architecture which uses residual blocks leveraging the idea of identity shortcut connections which skips the blocks.

VGG (Simonyan & Zisserman, 2015) [4] is a CNN characterized by its simplicity, it makes use of only 33 convolutional layers stacked on top of each other in increasing depth.

CNN as in [1] has [48, 64, 128, 160] channels for the first four layers and 192 for the other 4 layers. The max pooling window size is 2 2. The stride alternates between 2 and 1 at each layer. All convolutions use zero padding on the input to preserve representation size. We used batchnorm instead of substractive normalisation. All convolution kernels were of size 55. Dropout is applied to all hidden layers but not the input. The version that we used as ReLu instead of maxout units and we did not implement the locally connected layer. Our best performing model also had only 1 fully connected layer instead of 2.

For test time prediction, we tried to follow [1] and and predict the length using:

$$argmax_{L,S_1,\ldots,S_L} logP(S|X) = argmax_{L,S_1,\ldots,S_L} P(L|X) \prod_{i=1}^{L} P(S_i|X)$$

we did not see any difference in the performance on the validation versus simply using $argmax_L P(L|X)$ so we chose to use this simpler method. Therefore all the predictions including the sequence length, were done independently at test time.

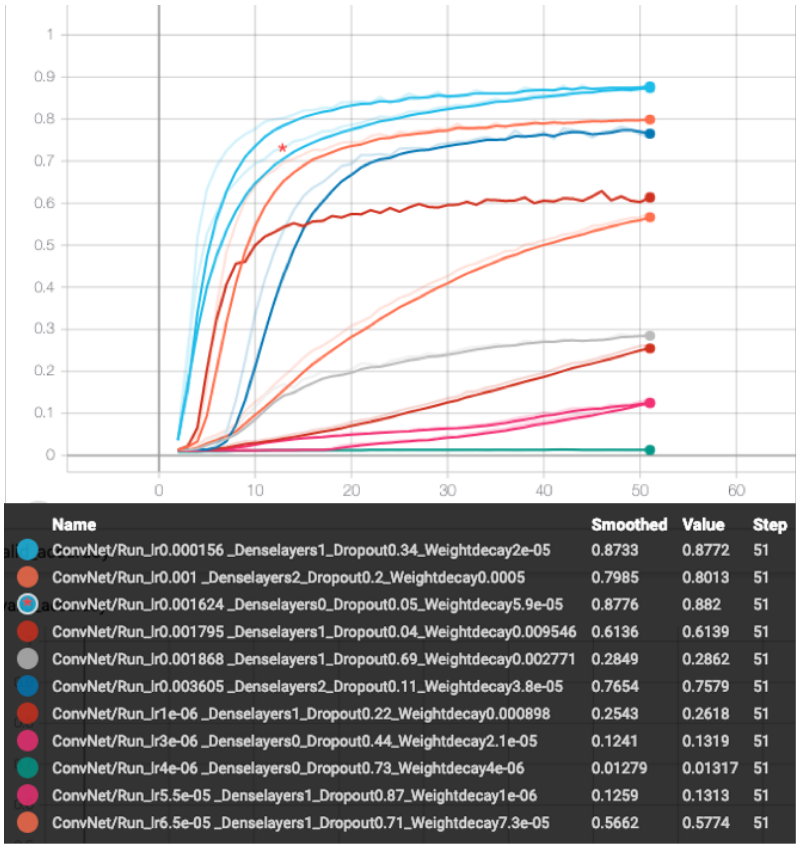
Experiments

The 4 different models were run for 50 epochs. We realized that we were getting great performance with the architecture used in [1]. We focus our effort on that particular architecture. We used Skopt library to perform Bayesian hyperparameter optimisation, 11 settings each for 50 epochs were tried. The hyperparameter that we optimized were the learning rate, the dropout rate, the number of fully connected layers at the end of the network and the weight decay (L2 regularization). Finally, we took our best performing architecture and trained it on both the extra and the train data for 200 epochs in order to come out with our final model. Checkpoints were done every 10 epochs. For these checkpoints, we saved the model, best performing model to date, best accuracy, number of epochs done, and optimizer state. The accuracy on individual digits, the length and the overall sequence was logged in TensorboardX, for both datasets (train and validation). Loss curves were also logged.

3 Results and Discussion

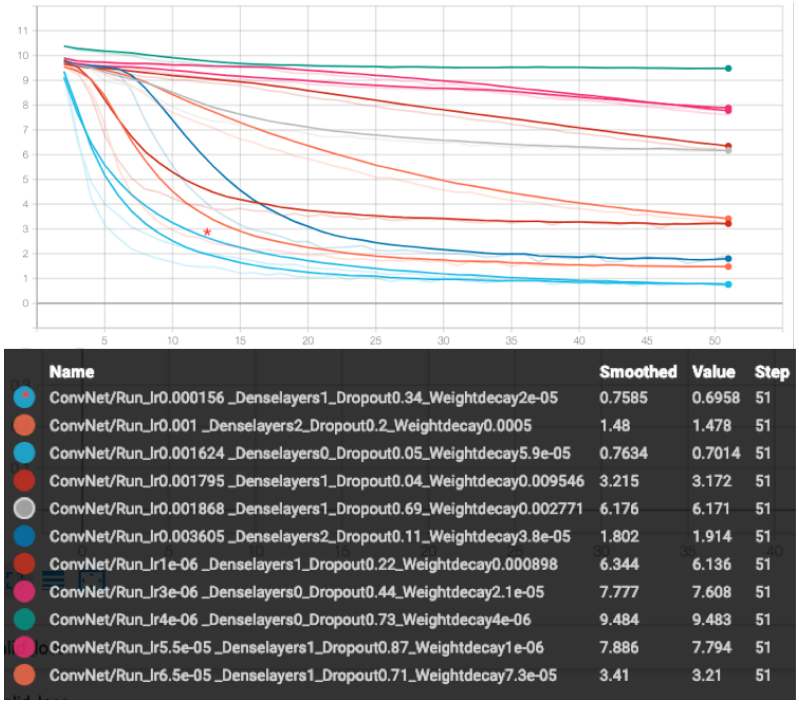
We first tested the CNN architecture used in [1], hereby referred as ConvNet. Various hyperparamaters were tested using Skopt, and these networks were trained for 50 epochs. Figure 1 shows the sequence accuracy obtained on the validation set, and figure 2 shows the related loss curves. Out of the 11 combinations tested, 2 were significantly better than the rest (both labeled in light blue by Tensorboard) with an accuracy of 88% each. The first model was selected as its trend suggested potential improvement in subsequent training epochs (marked with a red asterisk).

We also compared this selected ConvNet to other popular CNN architectures (VGG and ResNet). Figure 3 shows that after equivalent number of training epochs, ConvNet achieved the best accuracy



Name	Smoothed	Value	Step
ConvNet/Run_lr0.000156 _Denselayers1_Dropout0.34_Weightdecay2e-05	0.8733	0.8772	51
ConvNet/Run_lr0.001 _Denselayers2_Dropout0.2_Weightdecay0.0005	0.7985	0.8013	51
ConvNet/Run_lr0.001624 _Denselayers0_Dropout0.05_Weightdecay5.9e-05	0.8776	0.882	51
ConvNet/Run_lr0.001795 _Denselayers1_Dropout0.04_Weightdecay0.009546	0.6136	0.6139	51
ConvNet/Run_lr0.001868 _Denselayers1_Dropout0.69_Weightdecay0.002771	0.2849	0.2862	51
ConvNet/Run_lr0.003605 _Denselayers2_Dropout0.11_Weightdecay3.8e-05	0.7654	0.7579	51
ConvNet/Run_lr1e-06 _Denselayers1_Dropout0.22_Weightdecay0.000898	0.2543	0.2618	51
ConvNet/Run_lr3e-06 _Denselayers0_Dropout0.44_Weightdecay2.1e-05	0.1241	0.1319	51
ConvNet/Run_lr4e-06 _Denselayers0_Dropout0.73_Weightdecay4e-06	0.01279	0.01317	51
ConvNet/Run_lr5.5e-05 _Denselayers1_Dropout0.87_Weightdecay1e-06	0.1259	0.1313	51
ConvNet/Run_lr6.5e-05 _Denselayers1_Dropout0.71_Weightdecay7.3e-05	0.5662	0.5774	51

Figure 1: Sequence accuracy on validation set by ConvNet using various hyperparameters



Name	Smoothed	Value	Step
ConvNet/Run_lr0.000156 _Denselayers1_Dropout0.34_Weightdecay2e-05	0.7585	0.6958	51
ConvNet/Run_lr0.001 _Denselayers2_Dropout0.2_Weightdecay0.0005	1.48	1.478	51
ConvNet/Run_lr0.001624 _Denselayers0_Dropout0.05_Weightdecay5.9e-05	0.7634	0.7014	51
ConvNet/Run_lr0.001795 _Denselayers1_Dropout0.04_Weightdecay0.009546	3.215	3.172	51
ConvNet/Run_lr0.001868 _Denselayers1_Dropout0.69_Weightdecay0.002771	6.176	6.171	51
ConvNet/Run_lr0.003605 _Denselayers2_Dropout0.11_Weightdecay3.8e-05	1.802	1.914	51
ConvNet/Run_lr1e-06 _Denselayers1_Dropout0.22_Weightdecay0.000898	6.344	6.136	51
ConvNet/Run_lr3e-06 _Denselayers0_Dropout0.44_Weightdecay2.1e-05	7.777	7.608	51
ConvNet/Run_lr4e-06 _Denselayers0_Dropout0.73_Weightdecay4e-06	9.484	9.483	51
ConvNet/Run_lr5.5e-05 _Denselayers1_Dropout0.87_Weightdecay1e-06	7.886	7.794	51
ConvNet/Run_lr6.5e-05 _Denselayers1_Dropout0.71_Weightdecay7.3e-05	3.41	3.21	51

Figure 2: Validation loss by ConvNet using various hyperparameters

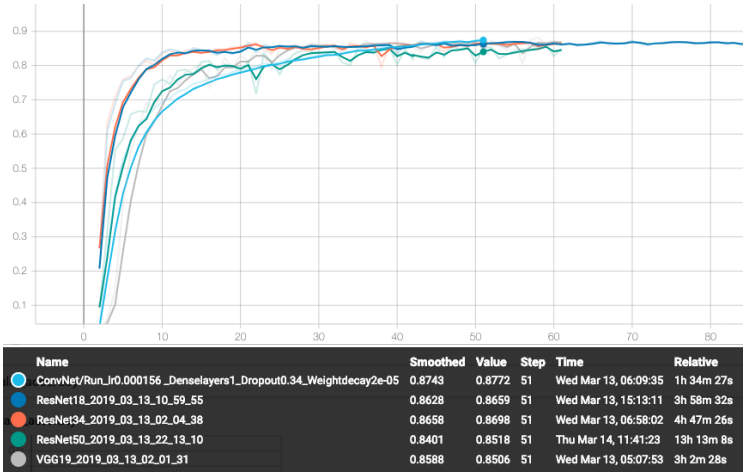


Figure 3: Sequence accuracy on validation set achieved by tested models

(87.7%). Training time for ConvNet was also much less than ResNet and VGG. We therefore selected this CNN as our model and trained it using the full dataset (train + extra) for an extended period of time (145 epochs in total, 38 hours). The best performing model was obtained at epoch 136.

Table 1 summarizes the results obtained with the various networks tested. Without using the full dataset, the best sequence accuracy was 87.7% by our selected ConvNet. When trained for much longer with the added extra dataset, this ConvNet achieves a sequence accuracy of 92.1% on our validation set.

We also compared the length accuracy obtained by these various networks with the best results obtained by the previous block. Team3 reported accuracies of 97.1% and 97.6% with ResNet18 and ResNet34 respectively. We obtained almost identical results with these same networks. Surprisingly, that seems to indicate that we are not seeing an improvement in the length accuracy from doing a multi-objective task. Note that our best model does reduce the length error by 25% but this can be explained by the use of the extra data rather than the multi-objective training.

We did not performed hyperparameter search on ResNets and VGGs because of their long training time. It is possible that with tuned hyperparameters, these networks might have outperformed the chosen ConvNet.

It is interesting to note that we are significantly lower than the 96.03% accuracy report in (Goodfellow, 2013) [1]. We did not implement the locally connected layer which might explained part of the difference. Also we lack time to fully train the model, accuracy was still improving when we stop the training. Finally, we don't know the composition of the test data. It is possible that it does not contain only difficult examples like our validation and training data but contain also some easier example like in the extra.

Table 1: Task Result on Validation Data

Name	Sequence Accuracy	Length Accuracy	Average Digits Accuracy
VGG19	0.851	0.970	0.937
ResNet18	0.866	0.971	0.936
ResNet34	0.870	0.973	0.934
ResNet50	0.852	0.966	0.929
ConvNet	0.877	0.973	0.936
ConvNet (extra)	0.921	0.982	0.959
ResNet18 (Block1)		0.971	
ResNet34 (Block1)		0.976	

Average digits accuracy is calculated as the number of correctly classified digits divided by the total number of digits.

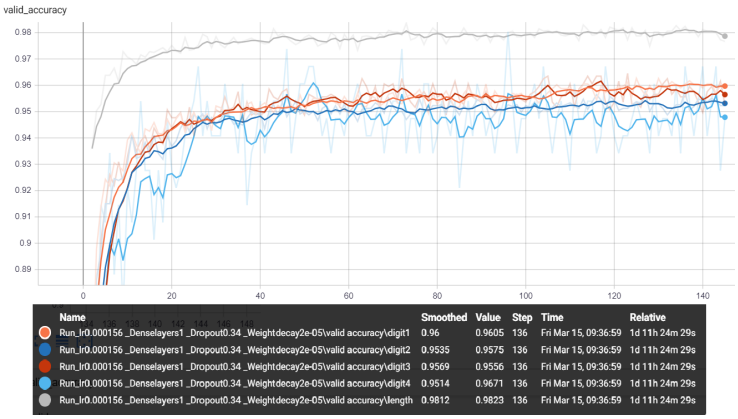


Figure 4: Individual digits accuracy by ConvNet (extra)

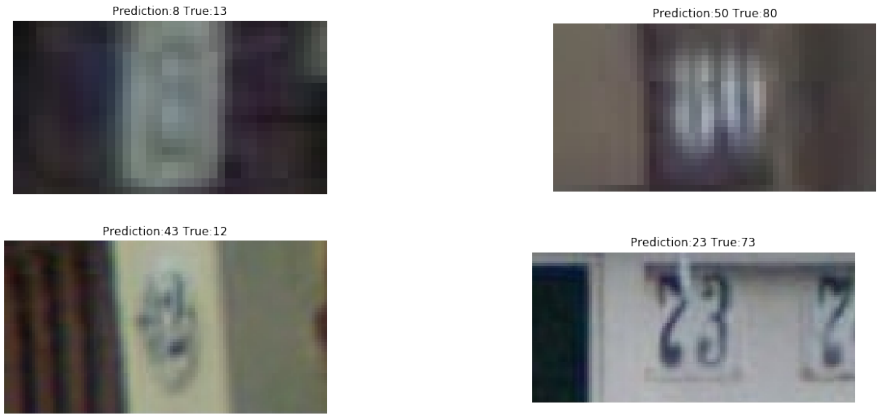


Figure 5: Incorrectly transcribed examples. Some of the picture are very blurry and difficult to tell even by a human.

4 Conclusion

In conclusion, during block 2 we have created a model that takes an image of house numbers for which we have the bounding boxes and predict the sequence of digits. We have tried several models and performed hyper parameter search and our best result were obtained by using an architecture similar to [1]. Our best model obtain 0.921. In order for this model to be useful for an application that aims to help people with vision impairments the bounding boxes will need to be found automatically by the model. While working on this project we realized that the training time can be quite long if the extra data is used. We therefore recommend experimenting with the training data only before using both the extra and training. We also recommend to keep one of the model used in block2 for the sequence prediction in order to save time.

References

[1] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay D. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *CoRR*, abs/1312.6082, 2013.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[3] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.