Street View House Numbers Digit Classification Using Convolutional Neural Networks

Hristo Stoev
College of Sciences and Health
Technological University Dublin
Dublin, Ireland
d18124660@mydit.ie

Mentor: Robert Ross
College of Sciences and Health
Technological University Dublin
Dublin, Ireland
robert.ross@dit.ie

***Abstract*—This paper presents a method for classifying digits from the SVHN dataset using a backpropagation-trained convolutional neural network architecture. This method achieves approximately a 73% accuracy rate on the test dataset.**

***Index Terms*—classification, convolutional, neural networks**

I. Model Design

A. Dataset

For building the classifier, the dataset that was used was the Street View House Numbers dataset [1]. It is composed of photos of Google Street View house numbers. There are two available versions of this dataset - the first containing the full images containing sequences of multiple house numbers, and the second being a set of cropped 32 by 32 pixel images centered on a single digit. The classifier model in this paper was trained using the second, cropped dataset.

B. Preprocessing

A number of operations were performed on the dataset before it was used for any model training.

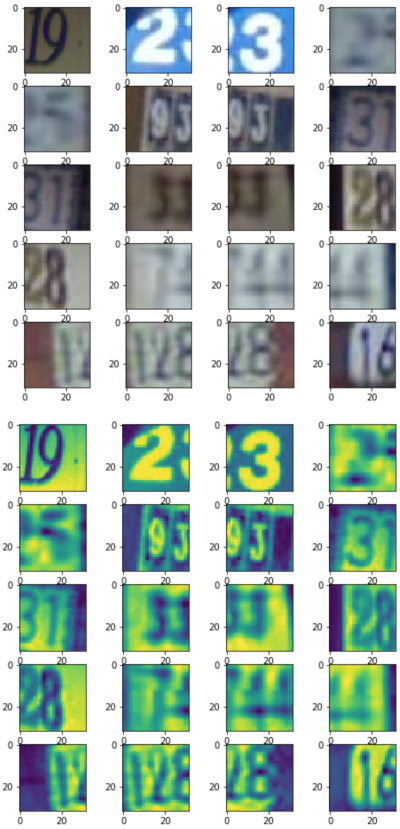
1) Centering: This involves subtracting the mean of the features from each feature value, essentially centering the data around zero rather than around the mean. Since this dataset is composed of three colour channels (RGB), centering was conducted on each channel separately.
2) Conversion to Greyscale: This decision was motivated by the circumstances of the classification problem. As can be observed in Figure 1, colours are not necessary for digit recognition, only the contrast present in the image. In fact, converting the images to greyscale tends to allow the model to focus only on learning features important to the problem.

Normalization was not conducted, as the feature space was already bounded in the range of [0,255]. As such, performing normalization was not necessary, since the variable range was the same.

C. Network Architecture

The image detection was performed entirely with the use of a multi-layer convolutional neural network, which was trained using backpropagation. The inputs of the network are 32 by 32 pixel images, and the output is one of a potential 10 classes,

Fig. 1. Subset of SVHN Dataset images in both RGB and Greyscale forms



each representing a digit classification. The activation function used at each neuron was the Rectified Linear Unit (ReLU), due to its superior convergence speed over alternative options [2]. The network architecture was made up of a combination of 4 main building blocks:

1) Convolutional Layers:
Since digits in the input images are not in constant positions, it is important that the network is able to detect features in a spatially invariant fashion. Additionally, using only fully-connected layers for feature extraction would lead to having too many parameters to train effectively. As such, convolutional layers are used to first

extract local features from the inputs.

2) Fully-Connected Layers:
   Convolutional layers are not the only hidden layer type used. The addition of a fully-connected layer at the end of the network architecture allows the network to combine and mix the local features detected by the convolutional layers to improve the model's learning capabilities.
3) Max Pooling:
   Pooling operations in CNNs are important as they reduce the dimensionality of the feature maps, making the model more generalizable. In this case, max pooling was chosen over average pooling due to the nature of the problem, and because experimental results show that max pooling leads to both improved accuracy and faster convergence in CNNs than average pooling [3].

In addition to the above components, different regularization techniques were applied so as to prevent the classifier model from overfitting the training data. These include:

1) Batch Normalization:
   In deep learning, the computation of the gradient of the error function is needed to update the parameters of a layer. However, since the parameters of each preceding layer are being updated simultaneously, the distribution of each layer's inputs changes during training. This introduces noise during training and makes it more difficult for learning algorithms to extract useful features. One technique for alleviating this is Batch Normalization [4].
2) Dropout:
   Another issue in deep learning is coadaptation of neurons on the training data. Essentially, the neurons in the same hidden layer will work together to model complex structures in the training data, leading to overfitting [5]. Dropout is a popular technique for reducing this coadaptation [6]. During training, neurons will have a random chance to be dropped (along with their connections) from the model, preventing them from coadapting with neighbouring neurons.

The layer architecture of the classifier network can be observed in Figure 2. It is broadly composed of two layer groups, each of which contains two convolutional layers and a maximum pooling layer.

The input layer consists of a single 32 by 32 channel, since this represents preprocessed images that have been converted from RGB to Greyscale. The first layer group consists of two convolutions, in which 32 filters of size 5 by 5 are used. Each convolution is followed by a Batch Normalization operation. Maximum pooling with a height and width of 2 is also applied, along with Dropout regularization. The Dropout technique is only used during training, and not during model inference.

The second group of layers follows the same structure as the first, but rather than the convolutional layers using 32 filters, they use 64. This then feeds into a fully-connected layer consisting of 32 hidden units. This amount of hidden units was chosen as any higher value would cause the training data to be overfit. and further into the output layer, which has 10 hidden units, one for each possible digit classification.

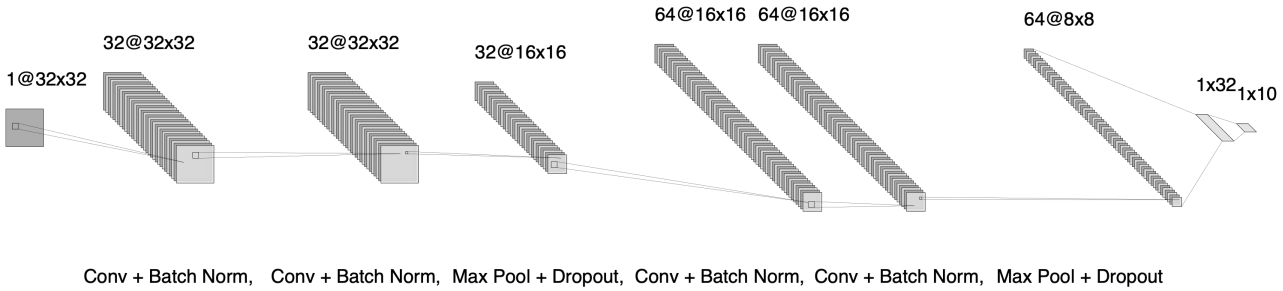
D. Model Training

The backpropagation algorithm was used to train the classification model [7]. The model cost function was calculated using softmax cross entropy, since the problem involved both classification and multiple potential classes.

The weights, or parameters, of the neural network were updated using the Adam Optimizer. This optimizer dynamically adapts the learning rate of the model, removing the need to manually implement learning rate decay over time [8]. The initial learning rate used was 0.0001.

The batch size was selected to be 256, as batch sizes over 200 lead to improved image recognition performance in CNNs [9]. Training was conducted for a total of 5 epochs. Often, the network would converge on a solution before all 5 epochs were completed. To reduce unnecessary model training time, and to reduce the chance of overfitting, early stopping was implemented. Model training was stopped when the error rate on the test set did not improve over subsequent epochs [10].

Fig. 2. SVHN Classifier Architecture



II. Results

A. Training Speed

Only a single epoch of training on the training dataset (n=73,257) was necessary before the network achieved convergence. This is likely due to the relatively large batch size of 256 that was selected for training the dataset.

B. Classification Performance

The classifier model achieved 72.8% accuracy on the test dataset (n=26,032). The F1 scores were also calculated for each individual classification label. The F1 score is an apt evaluation measure to use because it takes into account the balance between precision and recall. Table 1 shows the F1 scores for each classification category.

TABLE I
F1 Scores Across Categories

	Classification Categories (Digits)									
	1	2	3	4	5	6	7	8	9	0
F1	.76	.48	.60	.67	.68	.41	.78	.14	.32	.18

III. Conclusions

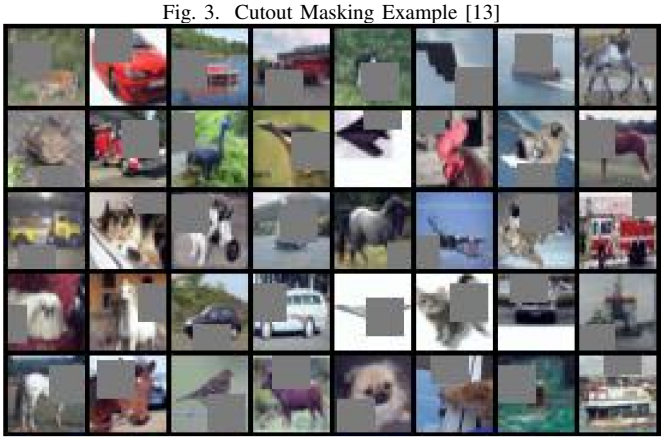
This work provides a successful implementation of a classifier for the SVHN dataset. However, the results are far off what is considered in the literature as 'state-of-the-art'. There are a number of potential reasons for this.

One significant contributor to model inaccuracy is the state of the input data. As can be seen from Figure 1, each image has been cropped to a constant size of 32 by 32 pixels. However, the digits represented in each differ in their sizes. In most cases, this means that there will be segments of other digits present in the image. This in turn introduces a significant amount of noise in the data, making it more difficult to train an effective classifier. The ideal approach to creating the dataset would have involved starting from the full images containing multiple digits and performing image localization and segmentation to determine the exact bounding box of each digit. Once this was done, the images would then be cropped to the dimensions of this box and padded with solid colour pixels to the size of 32 by 32. This approach was taken by [11], and resulted in a 4% error rate on the SVHN test set.

A number of additional techniques could have been made use of over the course of this analysis in order to improve classifier performance, along all steps of the deep learning pipeline. While the preprocessing stage did serve to improve model performance, further data augmentation could have been explored to expand on this. The current state-of-the-art model architecture for the SVHN dataset [12] achieves its supremacy primarily through improved data augmentation methods over other candidate models, specifically by using reinforcement learning to automatically determine the optimal data augmentation operations for each dataset separately. This leads to an error rate of 1%. In the case of the current analysis, data augmentation was limited to a simple RGB to Greyscale conversion, and the final model's performance would have benefited from additional work on this front.

The regularization techniques employed in this project to proved to be effective in reducing overfitting of the training data. However, as CNNs tend to have fewer parameters than Fully Connected Neural Networks, Dropout is not the most appropriate method of regularization to use when working with convolutions. Multiple researchers in the field have used more sophisticated techniques than simple Dropout, to high degrees of success. [13] utilize a technique known as Cutout, where rather than randomly dropping connections from the network during training, random sections of the input images are masked. This technique leads to improved classification effectiveness of the SVHN dataset and can be used in conjunction with any other CNN architecture. Making use of this technique would have been a prudent idea to improve model accuracy.

Fig. 3. Cutout Masking Example [13]



References

[1] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y Ng, "Reading digits in natural images with unsupervised feature learning," *NIPS*, 01 2011.

[2] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *CoRR*, vol. abs/1505.00853, 2015. [Online]. Available: http://arxiv.org/abs/1505.00853

[3] D. Scherer, A. Mller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," 01 2010, pp. 92–101.

[4] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[5] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[7] R. HECHT-NIELSEN, "Iii.3 - theory of the backpropagation neural network**based on nonindent by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593611, june 1989. 1989 ieee." in *Neural Networks for Perception*, H. Wechsler, Ed. Academic Press, 1992, pp. 65 – 93. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780127412528500108

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[9] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, 12 2017.

[10] R. Caruana, S. Lawrence, and C. L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," in *Advances in neural information processing systems*, 2001, pp. 402–408.

[11] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082*, 2013.

[12] E. D. Cubuk, B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation policies from data," *CoRR*, vol. abs/1805.09501, 2018. [Online]. Available: http://arxiv.org/abs/1805.09501

[13] T. Devries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *CoRR*, vol. abs/1708.04552, 2017.