COMP 551: Assignment 3

David C— (2607—)
Mounir Hamdar (2607—)
Jacob B— (2606—)

Abstract

We implement various deep learning methods for a computer vision task: from a noisy background picture containing three handwritten digits, we identify the highest such digit.

Attempted approaches include SVMs, dense neural networks, CNNs like the VGG structures, as well as a multi-input variant involving plausible identified regions for the digits to be contained in. The highest performing model was the latter, achieving an accuracy of 93.066% on the *Kaggle* test set while being trained on a dataset of 50,000 labelled examples.

Introduction

We aim to find a model with highest accuracy score for classifying the modified MNIST pictures. This model is then used to compete in a Kaggle competition. To do this, we trained a variety of models. Some of the models were trained on the raw training data provided by the competition, and others trained on a pre-processed version of the training data. The original model chosen had a heavy pre-processing, then a model taking only the raw data was trained and the best accuracy scores were achieved by combining a pre-processed and a raw data input model. Other pre-built models were tested, such as VGG19.

Finally, believing that 50000 images were too few to train an efficient neural network, we considered data augmentation.

0.1 Neural Networks

Neural networks are a structure organized by *layers* made up of *neurons*, where each neuron contains a value derived as a combination of values from neurons in a previous layer. Specifically, this value is computed as a linear combination of values from previous neurons which are then passed to a nonlinear *activation function*. The values of these neurons as well as the weights attached to the connections between neurons are learned as data is fed to any particular neural network through a variety of optimization algorithms, notably *Stochastic Gradient Descent* and the *Adam* algorithm. Neural networks have developed as some of the most powerful classification methods in recent years, notably [example]. In particular, neural networks are flexible and well suited to processing "raw" forms of data such as pixel-matrix representations of images, and are able to discern complex relationships between variables that are generally undetected by more classic approaches. Neural networks are considered the state-of-the-art when it comes to image classification and hence will be used in this project.

0.2 Convolutional Neural Networks

Convolutional neural networks are a variant of a neural network where some of the layers are replaced with convolutional layers. Convolutional layers perform a discrete convolution with a learned finite convolution mask onto an image. These layers are mostly effective in scenarios where the spatial or temporal dimension of the input data is relevant, such as for images or videos. [2]

In order to reduce dimensionality, maximum pooling layers are added to the networks. These agglomerate adjacent pixels together in the convolved image, helping reduce the size of the convolved image.

In our case, we also added a few dense layers at the end of the convolution layers: the dense layers are insensitive to the spatial organization of their input, and as such can help perform inferences across larger parts

of the input image.

Related Work

Image classification is a well known open ended problem in machine learning for which a variety of deep learning techniques have been designed. With the improvement of high-performance computing systems such as GPUs, and the construction of big data collections such as the MNIST and ImageNet databases have pushed progress in the image classification problem. In [1], Ilya Sutskever, Alex Krizhevsky and Geoffrey E. Hinton, designed a Convolutional Network used in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2010 and ILSVRC-2012 where they achieved the best results ever reported on the ImageNet dataset. This made us focus mainly on convolutional networks. In [3] Simonyan, Karen and Zisserman, Andrew, build on the ConvNet designed in [1] to make the VGG net, scoring 2nd overall in the ILSVRC-2014. In particular VGG scores highest on single-net performance, outperforming the winner GoogLeNet by 0.9%. Being easily accessible and training relatively quickly we train this model on our data set. The authors also argue towards the optimality of 3x3 filters, which is mainly what was used in our own models.

Dataset, Setup and Preprocessing

0.3 Image Segmentation

The training dataset of this project consists of 50k images. These 50k images are monochrome 128x128 images, which contain three digits from the MNIST dataset, placed and rotated randomly within a noisy background.

As training was difficult and often did not converge properly, use use a method

Instead, we resorted to isolating the digits among the background noise.

We first impose a binary brightness threshold, with pixel values above 220 getting mapped to 255 and all other values getting mapped to 0. This step eliminates most background noise.

All pixel clusters containing 5 pixels or fewer are then removed, as they most likely do not stem from a digit. We then dilate the remaining positive pixel regions by one pixel and select the three largest clusters out of the remaining ones.

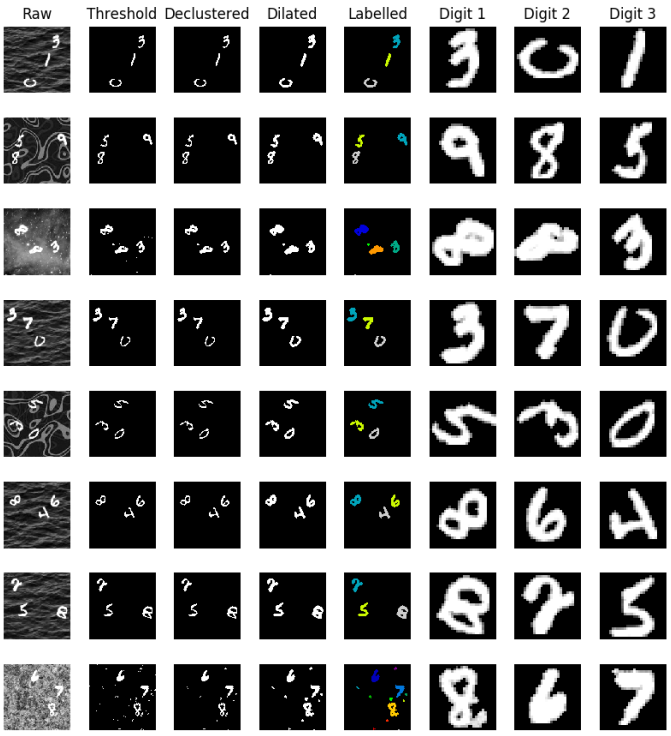


Figure 1: The digit isolation process.

At this point, we have a binary threshold mask indicating the position of the three digits. For every digit, we then seek the pixels in the original image at the position indicated by the digit mask. A cutout is then made and padded in order to fit the 28x28 pixel square from the original MNIST dataset.

However, since some of the written MNIST digits are made up of several disconnected pixel clusters, the above approach may fail to capture the entirety of a digit in favor of the largest section. The multi-input model therefore contains a second channel, consisting a 28x28 cutout of the raw picture around the digit in question. This second channel also includes the background noise of the raw picture.

We then use as input to our model these three dual-channel pictures, as well as the original raw image. The pixel values are also normalized from a range of $[0, 255]$ to $[0, 1]$ in order to facilitate the optimization.

Building the model

Proposed Approaches

1 Neural Networks, single-digit CNN, SVM

In all of these approaches, we segment the digits into three plausible clusters as in Figure 1. We then train each of these models on the original MNIST dataset, augmented with rotations and distortions.

For each of the three segmented digits, we then issue a prediction, and select the most plausible maximal digit out of the predicted probabilities for the three segmented digits.

2 Multi-input CNN

For this model, whose architecture is inspired by (but less deep than) the VGG models, we select as input the raw image, as well as the three dual-channel segmented digits as shown in Figure 2. They are then respectively pushed through 12 and 6 convolutional layers of kernel size 3 and stride 1, before a single dense layer. The final layers widths settled on were as described in Figure 3.

Maximum pooling layers are used after every two convolutional layers in both cases.

We then have a dense layer for each model, before both branches of the input are then merge and undergo three more dense layers, prior to the output layer.

Various types of regularization are also employed. We use batch regularization at every two or three layers, and we employ dropouts (or spatial dropouts at the first few layers) in order to improve performance.

2.1 VGG-16 and VGG-19

VGG-16 is a specific architecture of convolutional neural network known to perform well on image classification tasks, achieving an accuracy of 92.7% on the popular *ImageNet* dataset. It is a 16-layer convolutional layer network, with 5 pooling layers distributed throughout. Similarly, VGG-19 is a convolutional network designed in [3].

2.2 Support Vector Machines

A Support Vector Machine is a classification algorithm by finding decision boundaries in the form of a hyperplane over the parameters. An SVM does this by solving the optimization problem of maximizing the distance between the hyperplane and the closest data point of any class. These separations do not have to be linear, and in fact non-linear SVMs can be used to solve more complex problems, although they become much more computationally intensive, with the training runtime complexity scaling at least quadratically.

Results

2.3 Neural Networks

Several MLP architectures were trained on this dataset, primarily varying the number of layers as well as neurons. All networks used the `Adam` optimizer, `relu` activation for intermediary layers, `categorical cross-entropy` loss and `softmax` activation for the final layer. The three best performing architectures were the following:

- Neural Network 1: 1 hidden layer of width 512 with two dropout layers.
- Neural Network 2: 1 hidden layer of width 512 with no dropout.
- Neural Network 3: 1 hidden layer of width 2048.

	Accuracies	
	Original MNIST	Modified MNIST
NN1	93.4%	35%
NN2	92.7%	21%
NN3	93.5%	15%

We notice that simple neural networks performed very poorly on the new data, even with data augmentation. Note that the `Modified MNIST` accuracy column above and in subsequent columns represent the accuracy achieved in predicting the **maximum of three digits**.

2.4 Support Vector Machines

A Support Vector Machine with an RBF kernel was trained on the original MNIST data set in order to attempt predicting modified MNIST digits. Grid search was used to optimize the SVM, yielding an optimal regularization parameter C of 5, and a γ parameter of 0.05. While SVM performance on the original MNIST data set rivaled the deep learning approaches, reaching an accuracy of 98.2%, it failed to generalize to the extracted Modified MNIST digits and was discarded. This was a clear example of the ability of neural networks to better generalize to new data.

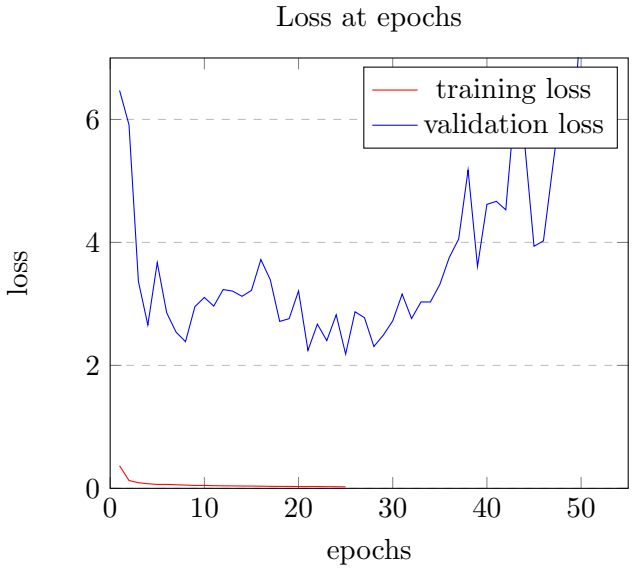
2.5 Single-digit Convolutional Neural Networks

Several MLP architectures were trained on this dataset, varying the number of convolutional and pooling layers, the number of neurons, and dropout. The best performing models had the following architectures:

- Convolutional Neural Network 1: 4 hidden layers of width 32, 64, 64, and 128 respectively.
- Convolutional Neural Network 2: 4 hidden layers of width 32, 64, 64, and 128 respectively, 1 dropout layer.
- Convolutional Neural Network 3: 4 hidden layers of width 64, 128, 128, 256 respectively.

	Accuracies	
	Original MNIST	Modified MNIST
CNN 1	98.7%	88.1%
CNN 2	99.4%	88.744%
CNN 3	99.3%	84.336%

We moreover optimized over number of epochs in order to prevent overfitting. In practice, a good heuristic to use is to stop training when validation loss starts increasing:



Based on the above figure, we stop training around epoch 30.

2.6 Multi-input CNN

Various variants on the previously defined multi-input model were tried. However, virtually all sensible variants of layer depths and layer widths started declining in validation accuracy between 93.5% and 94.5%, provided that between 12 to 36 million parameters were being used.

However, as compute resources were limited and each one of these model variants take between 4 to 5 hours to train on GPU, the architecture described in the previous section was used for the final submission to *Kaggle*, at an accuracy on the validation set of 94.9% and a Kaggle accuracy of 93.066%.

On this specific model, stochastic gradient descent was used, without momentum, and with an initial learning rate of 0.03. The learning rate was diminished by a factor of 0.25 after every 6 stagnant epochs and training was stopped after 15 stagnant epochs, for a total training time of 6 hours on Google Colaboratory's K80 GPU.

2.7 VGG-19

Although our original approach to input images without prior pre-processing was a failure, tuning the parameters of a VGG-19 net from [3] proved fruitful. Indeed it permitted us to reach a validation accuracy of

93.4% after 21 epochs before stabilizing. Our original problem was that the model only learned to predict the digit 9 for every image, reaching a local minimum of 27% accuracy. However, using a momentum of 0.9 got us out of the local minimum after a surprisingly long wait of 4 epochs. After tuning the hyper-parameters we found our optimal VGG-19 model to be set with a categorical cross entropy loss function, a learning rate of 0.0001, a decay of 1e-6 and a momentum of 0.9. Unfortunately this was realized very late and therefore no further experiments were tested.

Discussion and Conclusion

Broadly, this project confirmed that neural networks, and specifically convolutional neural networks, are generally the best approach to image classification problems. While it was already largely known that the MNIST dataset lends itself well to classification, the additional difficulty of this problem was that noise as well as multiple digits were introduced into each image. While some data preprocessing helps in this regard, the surprisingly strong performance of VGG-19 on displays one the primary benefits of deep learning techniques, which is that they require much less preprocessing of the data before being able to discover patterns in it. Ultimately, our best performing model was []. In future work, it might be worth focusing on the optimization of a VGG19-type network rather than on data preprocessing.

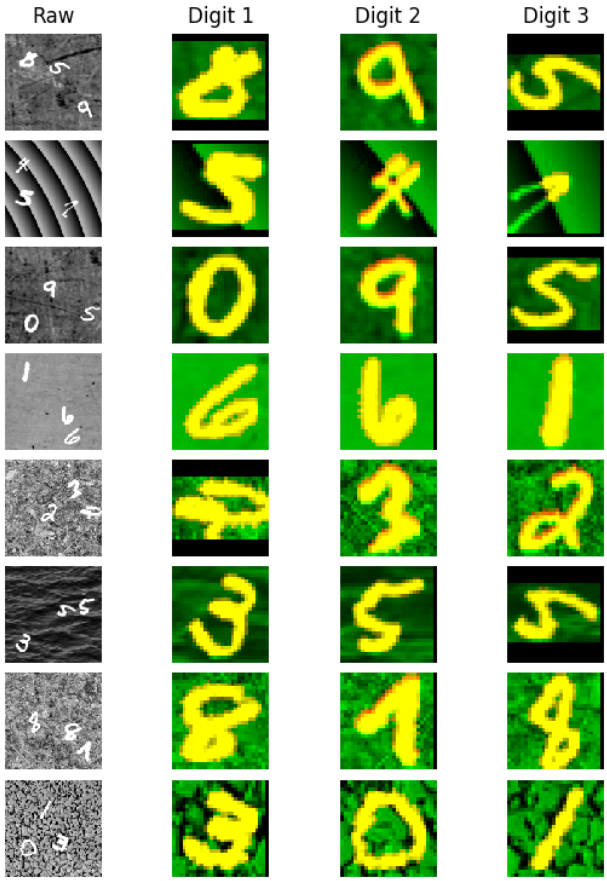


Figure 2: An example of the inputs given to the model. The digit input is on the red channel for the raw box around the digit, and in green for the plausible cutout. Notice that for the digit 7 in the second training example, the cutout does not capture the whole digit but the raw box captures a bigger portion of the digit.

References

[1] Ilya Sutskever Alex Krizhevsky and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, pages 1106–1114, 2012.

[2] Qiuhong Ke, Jun Liu, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid. Computer vision for human–machine interaction. Computer Vision and Pattern Recognition, page 131. 2018.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

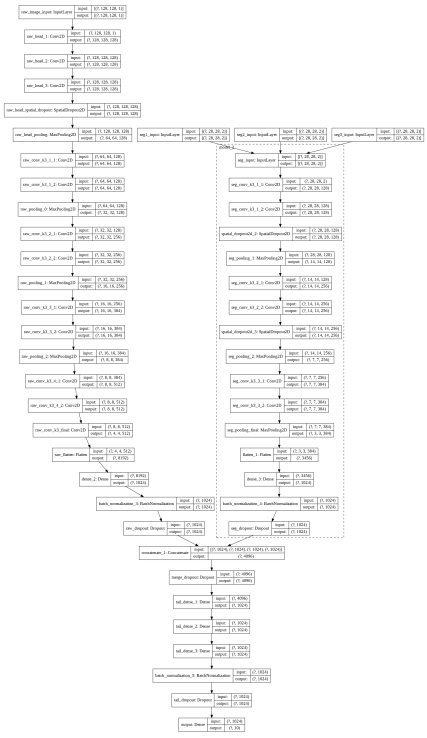


Figure 3: The architecture of the model used for the *Kaggle* submission