Introduction

Although a worthwhile opponent of machine learning decades ago, the MNIST dataset has served its time and is now ready to retire. Indeed, the advances in computer vision, namely convolutional neural networks (CNNs) has made possible great feats of accuracy on this dataset.

In order to test the power of CNNs and deep learning, a more advanced set of images must be created. Using the MNIST dataset, one can create images with 3 numbers and various background patterns, labeled with the highest number in the picture. Not only does this increase the complexity of the convolution step, but the dense layers face the additional challenge of learning how to rank integers.

Implementation

To tackle this problem, we first needed to import the data into a collaborative environment. Figuring out the mechanics of importing directly from Kaggle into Google Colab was tricky, but the problem was ultimately solved by Ryan. We used the Kaggle API to download the files into a folder and then unzipped them using Python.

In the meantime, Marwan and Louis-Philippe worked on developing the neural network architecture. Using Keras, they created a CNN with initially 1 convolutional layer and 1 dense layer. Quickly, we found that this was not enough to give good performance.

The first step towards a higher performance was to preprocess the images, which Ryan did. Each pixel in the image had a value between 0 and 255 representing the darkness. We noticed that the pixels representing the digits were the darkest (always had a value of 255). In order to remove the background behind each number, we first divided each pixel value by 255 and casted the result as an integer. Every pixel value under 255 became 0 which removed the background behind digits.

After the data was preprocessed correctly, we were able to experiment with our model and get much better results. We ended up with a model with 3 convolutional layers and 1 dense layer.

Results

The first iteration of the model was not optimal, even with the data preprocessing. We found that adding a second convolutional layer and a dense layer in the output increased the performance from 60% to about 80%. Looking online, having a kernel size of 3x3 seemed to be the best but experimenting with larger sizes, 5x5 and 10x10, we seemed to get much better results up to 85%.

We found that our model seemed to overfit rather quickly. To solve that problem, we added dropout after each convolutional layer. We found that a dropout of 0.2 enabled us to increase the accuracy to 93%

We initially had 15 filters on each layer, but increased to 30 on the second layer. We also tried with 3 and 4 convolutional layers and 5 and 6 dense layers, but we found that the performance decreased back to 60%.

As to prevent overfitting, we trained the model on 15 to 25 epochs. Above 25, we noticed that no improvements were made by the model and we started to fear overfitting. We also went from a batch size of 20 to 5, which improved the accuracy of 4%.

Challenges

One could say that the whole thing was hard from start to finish. First, we needed to find a way to import the dataset from kaggle to Colab, and this took a lot of time. We then needed to enhance the images and make the integers clearer to ease the learning process for our machine learning model. After that, we were faced with another fundamental problem: how exactly do we find the largest integer if we are not allowed to use external data sets? The problem prevented us from using external datasets, such as MNIST, so we needed to think of a machine learning model that would effectively study the images and learn how to pick the highest integer. We chose the convolutional neural network (CNN), rather than a regular neural network, because of one of its main advantages: feature learning. Indeed, CNNs have the ability to extract important features from images without human intervention (which was exactly what we needed to solve our problem).

We also encountered some problems with the creation of the convolutional neural network. For example, there was a mismatch between input dimensions and the expected dimensions of the loss function. We also needed to enhance various parameters for the model, which meant we had to fix all parameters and only change one at a time.

Finally, we were not able to use a module to randomize images for data augmentation. We tried introducing random rotations and displacements of the digits. However, the model resisted our deep learning black magic spells, and we had to make due without it.

Conclusion

The assignment was a lesson in the sense that it took more time to do everything than was planned. We expected to be able to work with keras easily, but the module presented differences from other modules previously used by the team members, like PyTorch.

Working on this problem helped us better understand how convolutional neural networks work. Specifically, our understanding of feature extraction and weight sharing was solidified, as we saw concrete examples of how they work. Trying to enhance the accuracy of our model prompted discussions about the various parameters in convolutional neural networks. For example, when we added a dropout layer to our CNN, we saw an increase in the model's accuracy. This helped us understand what dropout really does (it ignores random units from the neural network during training to prevent overfitting). We also recognized the importance of data preprocessing in the training step when we saw higher accuracies.

Overall, the assignment helped us better understand how machine learning models, like CNNs, can be used to solve real world problems.

Individual contributions

- Ryan: Ryan took care of creating the code to download the dataset from Kaggle to Colab, to preprocess the data and attempt to augment the data. He also adjusted parameters to significantly improve the performance of the model.

- Louis-Philippe: Louis-Philippe worked with Marwan to create the neural network and tried to implement a randomizer on the data.

- Marwan: Marwan helped create the architecture of the neural network and worked with Ryan to improve the performance of the model.