MAIS202 - Assignment 4 - Write-up

Solim LeGris, Yuliya Shpunarska, Maya Scott-Lourenço

November 23th, 2020

1 Model

The Xception architecture [1] was implemented in order to identify the largest digit in an image of handwritten digits. The model was supplied with cleaned images and optimized with Adam and stochastic gradient descent with a decaying learning rate.

1.1 Dataset

The dataset used to train a model to accomplish this goal contains 40 000 training images and 10 000 testing images, structured as `numpy` arrays. [2] Each image consisted of 128x128 pixels where each pixel has a single pixel-value associated with it to indicate the brightness of that pixel. For our dataset these pixel-values are integers between 0 and 255 where a higher number means a lighter colour. Each individual image depicts three handwritten digits atop a noisy background.

1.2 Data pre-processing

The first step of creating a model to achieve this goal was to clean up the dataset and decide on a proper representation. In order to remove the noisy backgrounds and isolate the digits, OpenCV's binary image thresholding [SOURCE 1] was used to set pixels below 235 to be 255.

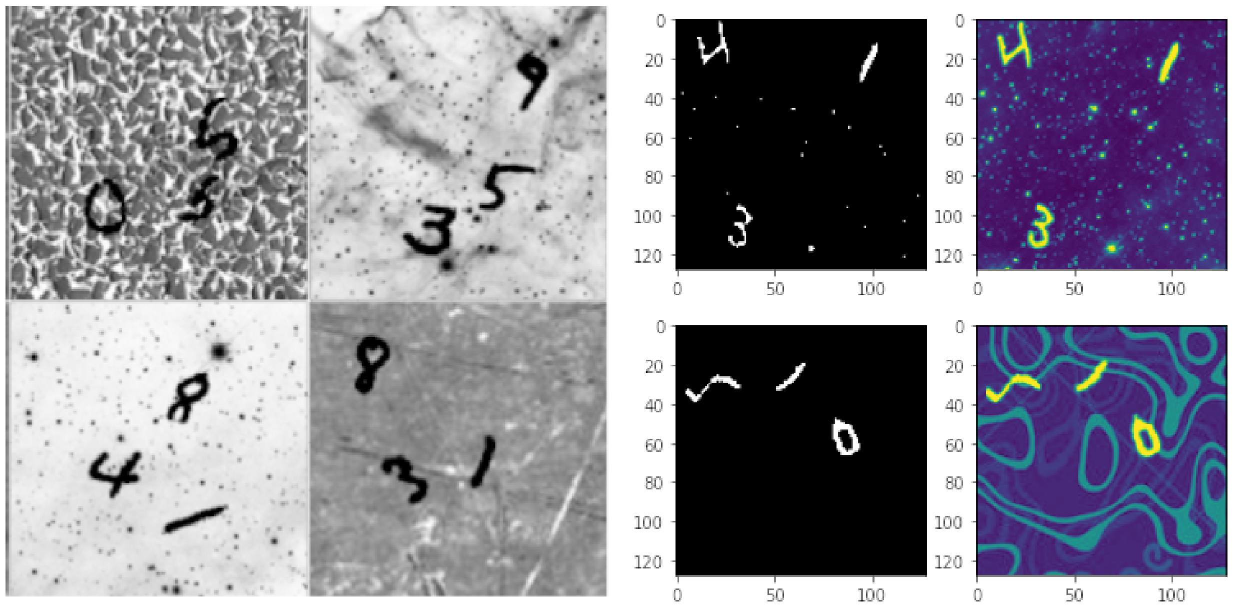

Figure 1: Sample of dataset images (left) and pre-processed images (right)

The pixels were then vectorized in order to use as input for the model. In the last part of data preprocessing, the cleaned data was resized to fit the selected model and was augmented by creating random but realistic distortions of an image (such as rotation) each time it is passed as input to the neural network in order to diversify the training data. [3]

1.3 Architecture

The machine learning model chosen to perform the modified mnist challenge was a convolutional neural network (CNN) that uses early stopping as a regularization method to avoid overfitting. A learning rate scheduler function was designed to improve the training time and improve the performance by adjusting the learning rate after each epoch. The pre-trained keras Xception network, a modified depthwise separable convolution neural network [1] was then used to classify the training set images with no input weights, the following specifications:

```
include_top=True,weights="imagenet",input_tensor=None,input_shape=None,pooling=None,
classes=1000,classifier_activation="softmax"
```

and the following hyperparameters:

```
target_size=(128, 128), color_mode="rgb", batch_size=32, class_mode="categorical",
shuffle=True, epochs-40
```

the unspecified parameters and specifications were kept their default values.

1.4 Metrics

The model's performance was evaluated using cross entropy loss on held-out data to evaluate the performance of the CNN quantitatively [1]. As the training progressed, the model's accuracy increased and the loss decreased [Figure 2] and it improved its ability to make accurate predictions of the largest number in the input images. There is a stabilization in the improvement of the accuracy and loss after 25 epochs where the model's performance does not change after which the training stops. A confusion matrix to compare the predicted and true labels but it suggested the model is only predicting 7 and decided to stick with loss and accuracy as metrics of model performance.

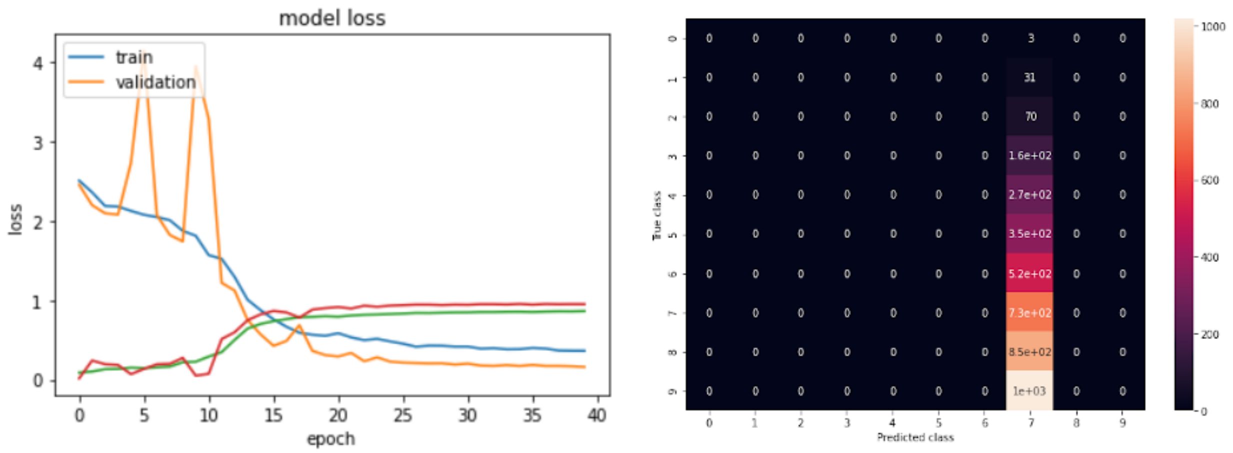

Figure 2: Training loss and accuracy per epoch (left) and confusion matrix (right)

2 Results

Initial implementations of keras' naive CNNs to achieve the outlined goal did not have any meaningful accuracy results. Hyperparameters were not optimized for these initial implementations as the results using the Xception architecture were far more promising. The Xception architecture with early stopping achieves an accuracy of 95% on the validation data and an accuracy 96% on the test data.

3 Challenges

3.1 Assignment challenges

Based on what has been seen in the lectures and the widespread knowledge of MNIST as a machine learning dataset the choice to use a CNN was not difficult but deciding how to clean the data and which model architecture to use was what proved to be a bit challenging. Starting from methodology described in a final project report for a Mobile Computer Vision class at Stanford. [4] Another challenge was working collaborative on a machine learning project which required a lot of back a forth to avoid repetition and keeping tracking of many small changes.

3.2 Implementation challenges

From the confusion matrix, we can see that our implementation struggles with classifying uncommon images (in the lower digits), despite data augmentation. Our attempts at using class weights were unsuccessful, probably due to the dataset being extremely unbalanced (in a previous implementation, class 0 had a weight 242 and class 9 had a weight 0.37, which were probably too aggressive). Due to time constraints, we chose to train the model without class weights, on the basis that the nature of this data makes the classes unbalanced.

4 Conclusion

The main takeaways from this assignment include how important it is to clean your dataset, the importance of understanding the goal of an implementation and code clarity. It's very easy to get "lost in the sauce" when it comes to machine learning programming and if the code isn't clear it makes it incredibly difficult for teammates to come in and understand what is being done and more importantly why. Overall, it was a good learning experience for solving a problem using machine learning without the guidance normally provided in the assignments and doing so in a team.

5 Individual Contributions

5.1 Solim

I suggested using thresholding to clean the images which was used in the final implementation. I implemented the bulk of the initial model implementations first trying a naive CNN then using the Xception architecture. I also contributed heavily to the discussion of the model design, hyperparameters and implementation.

5.2 Yuliya

I contributed mostly to optimizing the model by implementing data augmentation and learning rate decay. I also attempted to use class weights to influence the model's training but these were not used in the end. Lastly, I tried to use a confusion matrix to look at the class predictions and compare predictions to true values.

5.3 Maya

As a first step for data pre-processing I implemented Canny Edge detection [4]. I thought this would allow for the detection of the number outlines however the backgrounds are much too noisy and this did not help us isolate the handwritten digits. I also participated in design discussion and did the write-up+README.

References

[1] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *CoRR* abs/1610.02357 (2016). arXiv: `1610.02357`. URL: `http://arxiv.org/abs/1610.02357`.

[2] *MAIS 202 Fall 2020 - Kaggle Competition Dataset*. URL: `https://www.kaggle.com/c/mais-202-fall-2020-kaggle-competition/data` (visited on 11/20/2020).

[3] Connor Shorten and T. Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6 (2019), pp. 1–48.

[4] Xuan Yang and Jing Pu. “MDig: Multi-digit Recognition using Convolutional Neural Network on Mobile”. In: (June 2015). URL: `https://web.stanford.edu/class/cs231m/projects/final-report-yang-pu.pdf`.