Assignment 4

MAIS 202

Ritchie Yu, Roland Killian, Yichen Wu

Nov 23, 2020

Model Implementations

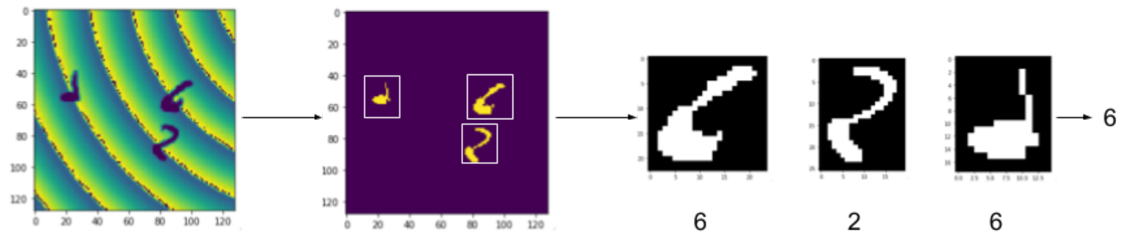
Bounding Box Approach

Our first idea was to process the image data by cropping it into smaller .npy images of each number in the input. Then, we would run these images through a number classifier, compare the three outputs, and manually return the largest value.

To approach this idea, the first step was to remove as much noise as possible from the images. We found that a very consistent way of doing this was to reduce all pixel values below or equal 0.99 down to 0.00, and all numbers above to 1.00. This returned very clean, binarized images.

From here, we extract the numbers in each image by drawing a box around each of the cropped numbers, and building 3 matrices with the internal data of each box.

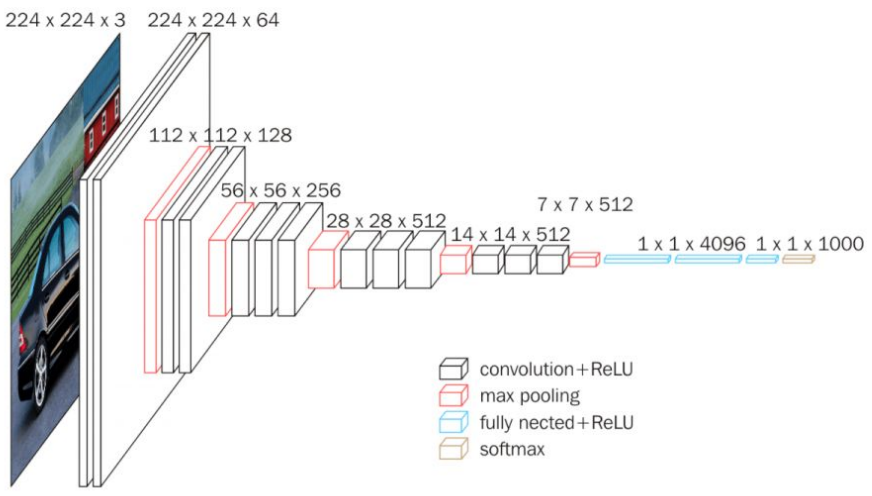
We found data as well as an algorithm online that can be easily used to build a simple number classifier. Using this model with our cropped images, we can predict the integer values contained in the original image and return the largest.



VGGNet Approach

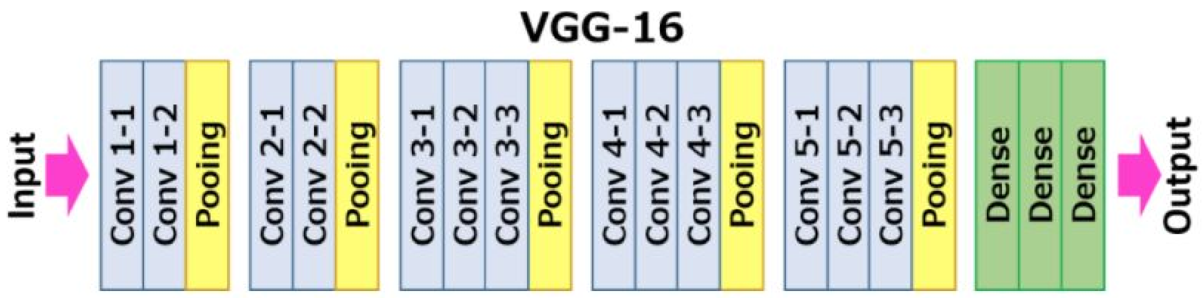
While we intended to use the bounding box approach, we found from online implementations that this approach was actually quite weak compared to passing the images directly into a CNN network. The reason was because the bounding box approach required using pre trained MNIST weights which are not trained on rotated images, and so it fails to accurately capture the features in each image. Therefore, rather than cropping the images after performing thresholding to remove the background, we decided to feed the images directly into a CNN network.

From further research, we found VGGNet to be a promising model. The following is what a typical VGGNet architecture looks like:



We proposed two different VGGNet models for this assignment. The first was based on a modified VGGNet implementation found online, which performed very well on the same modified MNIST dataset given. Their repository is located at: https://github.com/nikhilpodila/Modified-MNIST. Their implementation used 4 convolutional blocks, with each block containing 3 convolutional layers, followed by batch normalization, max pooling, and dropout. The final convolutional block fed into a fully connected network with 3 layers. Adam optimizer was used for gradient descent coupled with the categorical cross-entropy loss function.

The second VGGNet used was the traditional VGG16 model as illustrated:



For our implementation, after each convolutional block we included batch normalization to stabilize the learning process, as well as dropout to reduce overfitting. Rather than using Adam optimizer, we opted to use stochastic gradient descent (SGD) with momentum. Like before, categorical cross-entropy was used for the loss function.

Results

Bounding Box Approach

Since we found out that the bounding box approach does not work well, we did not use it to obtain any of our predictions.

VGG Approach

Using the modified VGG approach, we obtained a final validation accuracy of 97.2%. On the actual Kaggle test set, our best implementation obtained an accuracy of 96.7%.

This was not our best score, however. Using the VGG16 approach, we were able to improve, increasing our final validation accuracy to 97.7%, with an accuracy of 97.5% on the Kaggle test set.

The VGG16 approach varied slightly in architecture, with 5 convolutional blocks and 16 deep layers rather than the 4 blocks in the modified VGGNet approach, which had 15 deep layers. We suspect this helped VGG16 to achieve a better accuracy, as it is likely it was able to learn more and better abstract representations of the image data. Furthermore, the optimizers used were also different. SGD + momentum was used with VGG16, while Adam was used with the modified VGGNet. This may also be a reason why VGG16 had greater performance, because recent work has suggested that for very deep networks, SGD + momentum generalizes better than Adam.

Challenges

Bounding Box Approach

While the bounding box approach ultimately was not used to produce our predictions, we spent considerable effort trying to implement it. For example, before discovering that a simple basic threshold would remove the image backgrounds nicely, we tested many different complicated methods including canny edge detection and Otsu thresholding, none of which worked very well.

The most troubling part of this approach was outsourcing a number classifier. It is pretty easy to find an implementation of a number classifier, however like mentioned earlier, the boxed numbers had a rotational dimension that we could not account for. Had there been pre-trained weights for rotated digits available for us to use, we predict that the results would have been promising.

VGG Approach

After scrapping the bounding box approach, we removed all image processing, including threshold background removal. Our first time training yielded many problems, as we had consistently low validation accuracy, and were unsure how to stop overfitting. We tried choosing different hyperparameters, different optimizers, and variations of the VGG architecture, but none of these approaches fixed the stuck validation accuracy. However, after playing with the code, we finally suspected that the data was under prepared. By re-implementing background removal, it became clear that this was the source of our problem, and our training became much more successful.

Conclusion

This project taught us about the ML pipeline, and how to approach a problem given the type of data and the expected output. We needed to research a specific solution given the problem at hand, and went through multiple different implementations before settling on an optimal solution.

None of us expected machine learning to require so much trial and error. It seems like for any problem there are endless approaches, and the trick is to pick the right one, which is not an easy task. Even after, however, messing with data and hyperparameters until something works is less intuitive than we would have guessed.

The problem with our code that took us the longest to find was simply the fact that we had underprocessed the data. We made the incorrect assumption that the CNN would just do everything for us after feeding it the data. It turns out that machine learning algorithms require more nurturing than we previously expected.

We also found that when it comes to machine learning, our intuitions can often fail us. For this problem specifically, bounding boxes seemed like an obvious answer, but appeared to be a dead end after more research. This does seem rather problem specific however, as if there had been pre-trained weights specific to rotated images, this approach likely would have seen success.

Individual Contributions

Ritchie Yu
Most of my work focussed on researching the model and the different approaches that have been successfully used in the past. I helped with model implementation/training, and I contributed to image processing when we worked on the bounding box approach plus the final writeup.

Roland Killian
For this assignment, I spent most of my time dedicated to data processing, and preparing the data. This was a lot of guess and check, such as randomly deciding to divide the entire matrix by 255 such that it would run on our initial version of the CNN. I helped with the training of the VGG, and with trouble shooting whenever we got stuck. In addition, I contributed to the final writeup.

Yichen Wu
I was involved in the discussion of the project and tried to look into the gray scale of the image during the initialization phase of the assignment.

Citations

Hassan, M. U. (2018, November 21). VGG16 - Convolutional Network for Classification and Detection. Retrieved from https://neurohive.io/en/popular-networks/vgg16/

Norouzi, S., & Ebrahimi, M. R. (n.d.). A Survey on Proposed Methods to Address Adam Optimizer Deficiencies. Retrieved from http://www.cs.toronto.edu/~sajadn/sajad_norouzi/ECE1505.pdf

Podila, N, Bagchi, S, & A, M. Modified MNIST, (2020), GitHub Repository. Retrieved from https://github.com/nikhilpodila/Modified-MNIST/blob/master/ReadMe.md

Yang, X., & Pu, J. (n.d.). MDig: Multi-digit Recognition using Convolutional Nerual Network on Mobile. Retrieved from https://web.stanford.edu/class/cs231m/projects/final-report-yang-pu.pdf