Jesus Chavez

ITAI 2376

Reflective Journal

From this lab experiment, I have greatly improved my understanding of how convolutional neural networks (CNNs) work and my appreciation of their applicability to image recognition tasks. I also learned how CNNs are structurally very different from normal dense neural networks in that they can keep the relative positions of the image data. The concept of convolutional layers was even clearer, seeing how filters move over images to detect the lowest-level features like curves and edges, and then combine them in the deeper layers to create higher-level patterns. This made me realize that CNNs can find most visual features on their own without the need for feature engineering.

The next most significant idea that I have understood is pooling, in particular, max pooling, a method that reduces the spatial dimensions of the feature maps while still retaining the most important information. This makes the model computationally more efficient and also more resistant to small distortions or transformations of the input images. Besides that, I was introduced to the dropout mechanism that helps in the removal of overfitting by randomly dropping the neurons at the time of training, and thus allowing the model to learn more general features instead of memorizing the data. My experiment was heavily dependent on my prior knowledge of neural networks.

Before, I was working with fully connected networks where all the neurons were accepting input from all the features. Those models were not performing well on image data as they were treating each pixel as an independent object. In this experiment, I learned how CNNs make use of the spatial information present in images, and therefore, the training process is both efficient and accurate. There was nothing difficult in the MNIST dataset, and that is why it was perfect for me to focus on learning how to get proficient in architecture design without preprocessing. What impressed me the most was how CNNs could be so efficient and powerful even with relatively fewer layers.

My model was able to achieve an accuracy of more than 98% on the MNIST test set, which is great considering the epochs under which it was trained. Another thing that surprised me was that such minor changes in architectural or hyperparameter values, whether it was the filters, kernel size, or batch size, would result in performance changes of a huge magnitude. Manipulating these values made the entire learning process more logical and interesting. One of the issues I had was picturing how the data format changed as it went through each of the subsequent layers of the CNN.

When I initially attempted to understand how feature map sizes were changing due to the convolution and pooling operations, I was completely confused. To solve this issue, I used the model.summary() function so that I could know the output shape of each layer, and it was obvious after that how each of these operations was changing the data. They also had an overfitting issue during the training phase.

I noticed that my training accuracy kept increasing, while the validation accuracy was the same. After researching extensively about the subject, I was still not able to find a solution to the problem, so I decided to add a dropout layer and shorten the epochs. In fact, this allowed my model to generalize even better. I also came to the point that fine-tuning with early stopping would prevent wasting time when the model can no longer improve on the validation sets. I was mainly dependent on visuals to get my head around the difficult concepts.

While TensorFlow and Keras documentation were very good from the point of view of explanation and example, I also took a few brief video tutorials. The videos showed visually how convolutional filters scan an image and how dimensionality reduction is done through pooling. The videos gave me a more natural understanding of the convolution operation.

I was literally doing this in Colab—changing kernel size or adding layers—so that I could further justify these theoretical concepts by seeing the results. Because of all these difficulties, I have become proficient at debugging neural network models and understanding their performance metrics. I have become proficient at looking at training and validation curves to find underfitting or overfitting problems. Most importantly, I have come to view hyperparameter tuning as having a very real, tangible effect on the results.

This experiment has brought my deep learning from theory to practice. I see now how mathematical concepts are applied in solving real-world problems. Deep learning, to me, is a process where you construct, test, and refine models by designing and implementing incremental changes to them. It is not just coding but also analysis and critical thinking. I found that tiny changes—like batch size or learning rate—would influence the way the model was learning.

I would like to try out deeper CNN architectures like VGGNet, ResNet, and Inception, and see how deep networks perform on more complex datasets. Also, I want to know about transfer learning, where a trained model is changed to be used for a different task, and data augmentation techniques, which create more training sets by applying transformations like rotations, flipping, or other modifications to the original images.

I had an intuition about CNNs already, but through this lab, I got to know it practically and in real life. I realized that it is not about getting high accuracy, but how each layer is helping towards feature extraction. I learned about methods such as plotting filters and feature maps so that I have an idea of what the network is learning internally.

In short, this laboratory was both a theoretical and practical learning journey for me, which enabled me to combine my theoretical knowledge and deepen my technical skills. It acted as a link between its theoretical source and its practical application and thereby made me more enthusiastic to learn more about the deep learning topic, which is really fascinating.