L04 Reflective Journal

Throughout this lab, I delved into Convolutional Neural Networks (CNNs) and deepened my understanding of their architecture and practical applications. Initially, my knowledge of neural networks was mostly confined to fully connected layers and basic optimization methods. However, this project, which involved building a CNN to classify MNIST digits, not only expanded my theoretical knowledge but also provided substantial hands-on experience with key deep learning concepts.

**Learning Insights**

One of the primary lessons I learned was how the various components of a CNN work synergistically. The Conv2D layers act as the feature extractors—they scan images with a set of filters to capture essential features such as edges, textures, and shapes. I experimented with different kernel sizes (e.g., 3×3 vs. 5×5) and filter counts (32, 64, and 128) to see how these parameters affect the network’s learning process. I observed that increasing the number of filters allowed the model to learn more detailed and complex features, which improved convergence speed and overall performance. However, this also increased computational demands—a balance must be struck based on the complexity of the dataset.

MaxPooling2D layers were also pivotal, as they down sampled the feature maps, reducing both the computational load and the likelihood of overfitting. I noticed that these layers helped the network become more robust to minor shifts and distortions in the input images. In addition, the one-hot encoding of labels was crucial because it allowed the softmax activation in the output layer to generate a probability distribution, ensuring that each class was treated independently.

Beyond the core CNN components, this lab encouraged me to explore enhancements that significantly improved both performance and training efficiency. I integrated data augmentation techniques using Keras’s ImageDataGenerator to expand the training dataset by applying random transformations (like rotation, zoom, and shifts). This not only improved the model’s generalization but also prepared it for handling more diverse inputs in future applications.

Another important enhancement was the use of TensorBoard for visualization. This tool provided valuable insights into training metrics and the model architecture, which made it easier to understand the impact of various hyperparameters on learning dynamics. Additionally, incorporating early stopping and model checkpoints helped prevent overfitting by halting training once the validation loss ceased to improve and saving the best-performing model automatically.

A significant practical improvement was the switch from the default CPU runtime in Colab to the T4 GPU. This change led to dramatic speed improvements; epoch durations were considerably shorter, which enabled me to run more experiments in a shorter amount of time and iterate on the model design much faster.

**Challenges and Growth**

Tuning hyperparameters posed one of the greatest challenges. Determining the optimal number of filters, kernel sizes, batch sizes, and epochs required multiple rounds of experimentation. Initially, small tweaks often led to unexpected results, which necessitated a more systematic approach. To manage this, I meticulously documented each experiment, noting the training and validation accuracies, losses, and even the epoch timings using a custom callback. This documentation allowed me to see clear trends and informed subsequent adjustments.

To overcome these challenges, I documented each experiment meticulously. I added a custom callback in Cell 5 to record the duration of each training epoch, which provided valuable insights into how TensorFlow optimized the computation graph over time. Additionally, I switched from using the CPU that Colab initially provided to utilizing the T4 GPU, and I observed dramatic improvements in training speed. This transition significantly reduced the time per epoch and allowed for more rapid experimentation. I also consulted various online resources and discussion forums to aid me on my learning journey.

**Personal Development**

This lab has significantly transformed my perspective on deep learning and CNNs. I now understand that designing a CNN involves much more than simply stacking layers—it requires a thoughtful arrangement of components to extract hierarchical features effectively. The hands-on experience of modifying the network architecture and observing the immediate impact on training performance was particularly enlightening.

Looking forward, I am eager to explore more advanced architectures such as ResNets and DenseNets, which address challenges like vanishing gradients and promote efficient parameter usage. I am also interested in investigating further data augmentation techniques and other regularization methods to improve model generalization on more complex datasets.

**Summary**

In summary, this lab was a transformative experience that not only enhanced my technical skills but also deepened my appreciation for the intricacies of CNNs. It taught me the importance of balancing model complexity with computational efficiency and provided a solid foundation for future exploration in the field of deep learning.