Reflective Journal on Chihuahua or Muffin Workshop

The main objective of the workshop was to understand and implement fundamental techniques in image classification using machine learning models. We specifically worked with a dataset distinguishing between images of chihuahuas and muffins—a classic example highlighting the challenges of visual similarity in machine learning. The workshop primarily utilized convolutional neural networks (CNNs), as well as an original neural network architecture named "MySkynet," to understand how different network architectures and learning techniques can be used to classify images effectively.

One of the key activities in the workshop involved cloning a GitHub repository to gain access to the notebook and data files, followed by preprocessing the data and training a simple neural network. We also learned how to make use of PyTorch and torchvision libraries for building and optimizing models. Furthermore, we applied various image transformations to preprocess the images and improve model generalization, such as resizing, normalization, and data augmentation.

The training was done on a powerful GPU-enabled system, which allowed us to leverage the computational benefits of CUDA for efficient model training. The project culminated in using different hyperparameters and batch sizes to find an optimal configuration that yields good classification accuracy.

One of the major concepts learned during the workshop was image classification using CNNs. Convolutional Neural Networks are highly effective in extracting spatial features from images, which are then used to classify images into different categories (Simonyan & Zisserman, 2015). We also explored transfer learning, a technique where we utilize a pre-trained model, such as VGG or ResNet, and adapt it to our specific dataset (He et al., 2016). Transfer learning allows us to achieve good performance without training the entire network from scratch, which is crucial for practical applications with limited data or computational resources.

Another important concept we covered was hyperparameter tuning. We learned about optimizing batch size, learning rate, and network architecture to improve model performance. I experimented with a batch size of 32, which struck a good balance between training speed and stability (Goodfellow et al., 2016). The use of 224x224 image size, a standard in popular pre-trained models like VGG and ResNet, was instrumental in ensuring compatibility and maintaining a baseline level of image quality.

Additionally, I encountered and overcame several technical issues, such as module errors related to PyTorch and torchvision. These issues were solved by installing missing packages directly within the notebook using %pip install commands.

One of the primary challenges I faced during the workshop was setting up the environment correctly. Initially, I encountered ModuleNotFoundError for both torch and torchvision. To overcome this, I used the commands %pip install torch and !pip install torchvision to install the missing dependencies directly in the notebook, which resolved the issue.

Another challenge was understanding how to effectively represent image data for the neural network. Unlike CNNs, which inherently recognize spatial relationships in images, our simple network, MySkynet, required the images to be flattened into vectors. Flattening the image data removed spatial information, which can be critical for distinguishing similar images, such as chihuahuas and muffins. Despite this, by incorporating additional layers and experimenting with different activation functions, I was able to optimize the network and achieve a reasonable accuracy rate of 0.547.

Hyperparameter tuning was also a major challenge, particularly in achieving the right balance between overfitting and underfitting. I used cross-validation and grid search to determine the best values for parameters like the learning rate and number of neurons in each layer. This iterative process, although time-consuming, provided valuable insights into the role of each hyperparameter in determining model performance.

The workshop provided deep insights into the fundamentals of machine learning, particularly in the context of image classification. One significant insight was the importance of data preprocessing and augmentation. Techniques like resizing, normalizing, and applying random transformations such as rotations can significantly improve model generalization by providing more diverse training data (Howard & Gugger, 2020).

Another valuable insight was the role of computational resources. Training deep learning models on GPUs is significantly faster than using CPUs, especially for large datasets. Leveraging CUDA capabilities allowed us to train models in reasonable timeframes, which is critical in practical applications.

Moreover, I gained a greater appreciation for the limitations of traditional neural networks in comparison to CNNs. While fully connected networks can work for image data, they struggle to maintain spatial relationships that CNNs inherently preserve. This experience underscored the importance of selecting the appropriate architecture based on the problem at hand.

The techniques learned during the workshop have a wide range of potential real-world applications. Image classification can be applied to facial recognition, medical imaging, and object detection in autonomous vehicles. For example, using a model like MySkynet or a pre-trained network through transfer learning could help classify medical images to detect diseases or identify defective products in a manufacturing line.

The ability to use transfer learning is particularly impactful in scenarios where labeled data is limited. Instead of training a model from scratch, leveraging pre-trained models can significantly reduce the computational cost and improve performance. This approach is useful for applications such as satellite image analysis, where collecting labeled data is challenging.

Overall, this workshop was a valuable learning experience that helped me bridge the gap between theory and practical implementation. Working with image data and building a custom neural network was both challenging and rewarding. I learned to navigate common challenges in machine learning, such as setting up environments, handling large datasets, and tuning hyperparameters.

I found it particularly gratifying to train a model that could classify images with reasonable accuracy, especially after overcoming the technical hurdles of environment setup and

understanding data representation. This experience has not only deepened my understanding of machine learning concepts but has also enhanced my problem-solving skills.

Looking ahead, I am excited to further explore Convolutional Neural Networks and transfer learning in more depth. I believe these techniques will be essential as I continue my journey in machine learning, particularly in projects involving computer vision and natural image classification.

Reference:

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. Retrieved from https://www.deeplearningbook.org/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Retrieved from https://arxiv.org/abs/1512.03385

Howard, J., & Gugger, S. (2020). *Deep Learning for Coders with Fastai and PyTorch: AI Applications Without a PhD*. O'Reilly Media.

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*. Retrieved from https://arxiv.org/abs/1409.1556