Neural Network Training Journal

Date: 10/01/2024

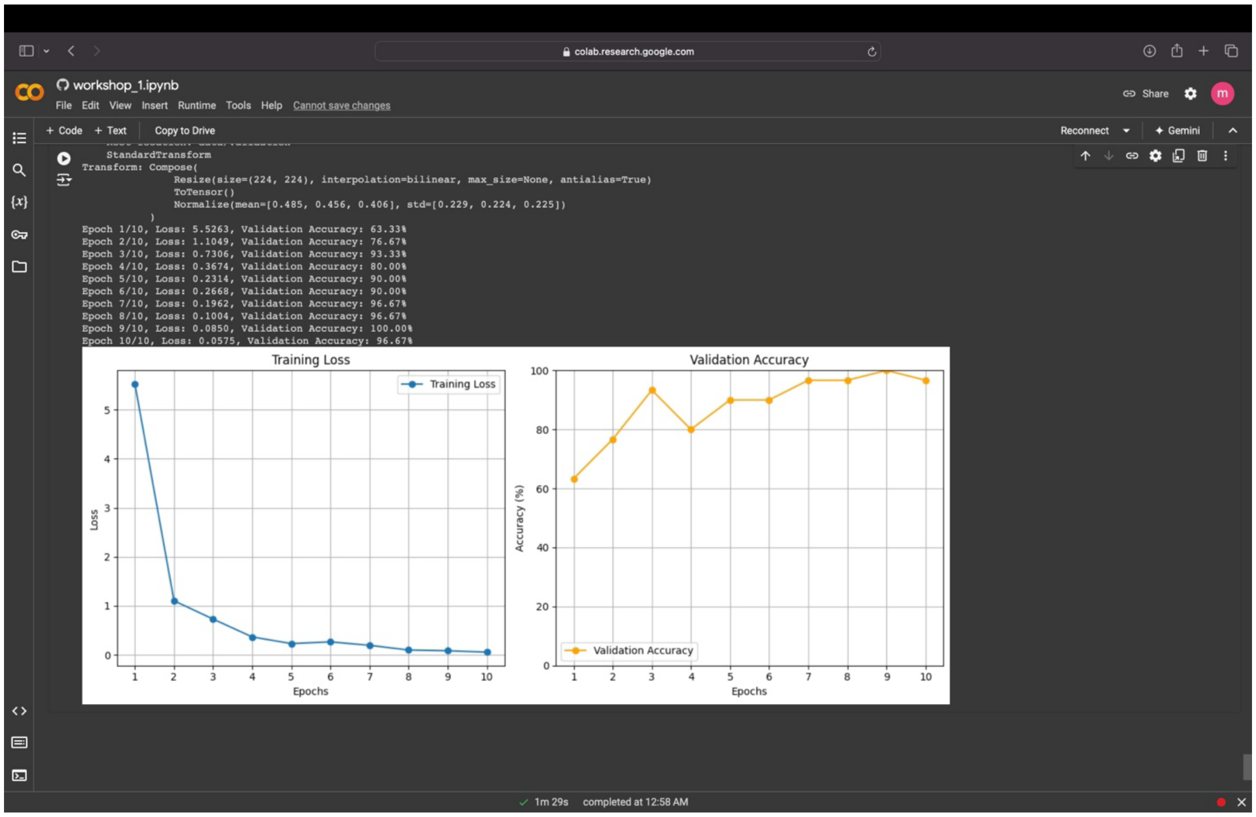
Project: Image Classification with a Simple CNN
Objective: Train a convolutional neural network (CNN) to classify images into distinct categories (Chihuahua vs. Muffin) using PyTorch.

During my recent project, I focused on training a convolutional neural network (CNN) using the PyTorch library to classify images into two categories: Chihuahuas and muffins. The main objective was to build a model that could achieve high validation accuracy while experimenting with different parameters to optimize its performance. I used the `ImageFolder` method from the `torchvision` library to load images from the training and validation directories, as provided by my professor, which were crucial for my dataset. To ensure that the model could generalize well, I applied data augmentation techniques such as resizing and normalization. These transformations were essential for preparing the images, as they standardize inputs, helping the model handle a variety of real-world cases (Paszke et al., 2019).

For the model architecture, I designed a CNN with two convolutional layers, each followed by a ReLU activation function and max-pooling layer. This structure was beneficial for extracting essential features from the images. Following the convolutional layers, fully connected layers were added to produce class probabilities. I used the Adam optimizer with a learning rate of 0.001, based on suggestions in the PyTorch documentation (Kingma & Ba, 2014). The cross-entropy loss function was chosen, as it's well-suited for multi-class classification problems. The batch size was set to 32, and the model was trained for 10 epochs.

While executing the training process, I encountered a `FileNotFoundError` when attempting to load the datasets. With help from the files and data provided by the professor, I resolved this issue by ensuring the correct dataset directory paths were used. Once the data-loading problem was fixed, I noticed a steady decrease in training loss, indicating that the model was learning effectively from the data. However, the validation accuracy fluctuated, eventually reaching a peak of 100%. Visualizing the training and validation loss using Matplotlib helped identify learning patterns and track when adjustments to the model were necessary.

An important lesson was understanding the role of data augmentation techniques in preventing overfitting. Applying transformations such as random horizontal flips improved the model's ability to generalize to unseen data (Shorten & Khoshgoftaar, 2019). While the model's performance showed significant improvement, there is room for further enhancements. In future iterations, I plan to explore different hyperparameters, such as varying the learning rate and optimizer types, to optimize the model further. Additionally, implementing more advanced techniques like transfer learning could boost performance. Overall, the project deepened my understanding of CNNs and how to utilize PyTorch for image classification, with crucial guidance from the course material.


colab.research.google.com
workshop_1.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
Reconnect
Gemini
Training Loss
Training Loss
Loss
Epochs
Validation Accuracy
Accuracy (%)
Validation Accuracy
Epochs
1m 29s completed at 12:58 AM

```
    StandardTransform
Transform: Compose(
               Resize(size=(224, 224), interpolation=bilinear, max_size=None, antialias=True)
               ToTensor()
               Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
           )
Epoch 1/10, Loss: 5.5263, Validation Accuracy: 63.33%
Epoch 2/10, Loss: 1.1049, Validation Accuracy: 76.67%
Epoch 3/10, Loss: 0.7306, Validation Accuracy: 93.33%
Epoch 4/10, Loss: 0.3674, Validation Accuracy: 80.00%
Epoch 5/10, Loss: 0.2314, Validation Accuracy: 90.00%
Epoch 6/10, Loss: 0.2668, Validation Accuracy: 90.00%
Epoch 7/10, Loss: 0.1962, Validation Accuracy: 96.67%
Epoch 8/10, Loss: 0.1004, Validation Accuracy: 96.67%
Epoch 9/10, Loss: 0.0850, Validation Accuracy: 100.00%
Epoch 10/10, Loss: 0.0575, Validation Accuracy: 96.67%
```

References

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv preprint arXiv:1412.6980. https://arxiv.org/abs/1412.6980

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems, 32, 8026-8037.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 60. https://doi.org/10.1186/s40537-019-0197-0