Erik Mercado

CSE 244A Final Report

Running the Notebook:

1. Preparing the Environment

- Google Colab:
    - If using Google Colab, it's recommended to mount your Google Drive to the
    - After mounting the drive, upload the dataset to your Google Drive or download it directly into the drive for easier access.
- Local Environment:
    - If running the notebook locally, ensure the dataset is downloaded to a known directory on your machine.

2: Configuring the Notebook

After obtaining the dataset, update the configuration settings in the notebook.

- Training Directory:
    - Locate the config section in the notebook.
    - Change the dir path to the directory where your training data is stored.
- Test Directory:
    - Update the test_dir path in the configuration to point to your test data directory.
- Model Save Path:
    - Modify the saved model path to specify where the trained model will be saved.

3. Running the Notebook

You can start executing the notebook with the environment set up and the dataset in place.

- Sequential Execution:
    - Run each cell in the notebook from top to bottom. This sequential execution is essential as later cells often depend on the output or variables generated by earlier cells.

Model Selection and Weight Initialization:

In this project, I have chosen the Vision Transformer (ViT) with a specific set of pre-trained weights for our image classification task.

Model: Vision Transformer (ViT):

The Vision Transformer (ViT) represents a shift in image processing, moving from traditional convolutional neural networks (CNNs) to a transformer-based approach, which has been highly successful in natural language processing tasks. The key features of ViT that make it a suitable choice for image classification are:

- Attention Mechanism: Unlike CNNs, ViT uses a self-attention mechanism, allowing the model to weigh the importance of different image parts, leading to better feature extraction.
- Model Depth: The vit_b_16 model is considered to have a moderate depth compared to some larger transformer models. This can positively impact training speed, as fewer layers require less computation per forward and backward pass.
- Global Context: ViT captures global context in the image by considering the entire image at once, rather than local patches as in traditional CNNs

Pre-Trained Weights: ViT_B_16_Weights.IMAGENET1K_SWAG_E2E_V1:

For this project, I used the ViT_B_16_Weights.IMAGENET1K_SWAG_E2E_V1 pre-trained weights. The benefits of using these weights are:

- Robust Feature Learning: These weights are obtained from a model trained on the ImageNet-1K dataset, which is a large and diverse dataset. This pre-training helps in learning robust and generalizable features.
- Transfer Learning: By using pre-trained weights, we leverage transfer learning, which can significantly improve performance, especially when our dataset is limited or when training from scratch is computationally expensive.
- End-to-End Training (E2E): The 'E2E' in the weight's name signifies that the model was trained end-to-end, optimizing the entire model's weights, which often results in better performance.

Adaptation to Our Task:

To adapt the ViT model to our specific image classification task, we made the following modifications:

- Customizing the Output Layer:
    - We replaced the original output layer (head) of the ViT model with a new linear layer to match the number of classes in our dataset:

Data Setup:

This setup involves steps and configurations tailored to effectively prepare and process the image data.

Image Transformation and Augmentation

To enhance the model's ability to generalize and prevent overfitting, I employ a variety of image transformations and augmentation techniques.

- Training Transforms:
    - RandomResizedCrop, RandomHorizontalFlip, RandomRotation, and RandomAffine are used for data augmentation, introducing variability and robustness into the training process.
    - Resize and CenterCrop to a uniform size of 384x384 pixels, ensuring consistency in input size for the model.

        - ToTensor and Normalize transformations are applied to convert images into a format suitable for training and to normalize pixel values based on predefined mean and standard deviation.
- Validation Transforms:
    - I apply Resize, CenterCrop, ToTensor, and Normalize transformations for the validation dataset, focusing on the consistent and representative evaluation of the model's performance without the randomness of augmentation.

Advanced Data Augmentation Techniques: Mixup and Cutmix

- Mixup Data:
    - This technique creates new training examples by linearly combining pairs of images and their labels, enhancing the model's generalization ability.
    - Implemented in the mixup_data function, it uses the beta distribution to determine the mixing ratio.
- Cutmix Data:
    - Cutmix augmentation involves cutting and pasting patches among training images and adjusting the ground truth labels accordingly.
    - The cutmix_data function handles this process, creating a more challenging and diverse training data set.

Custom Dataset Class

- I define a CustomDataset class to load and transform images from our dataset.
- The class supports reading images from a directory structure, where each subdirectory represents a class.
- It provides mechanisms to handle both training and validation data splits.

Preparing Training and Validation Datasets

- The dataset directory is specified in the config["dir"].
- I use the train_test_split method to divide the dataset into training and validation sets, ensuring a balanced representation of classes.
- Separate instances of CustomDataset are created for training and validation datasets, with respective indices and transformations applied.

Training Setup:

Device Configuration

- The model training is configured to utilize a GPU if available, significantly speeding up the training process. The code automatically detects GPU availability using torch.cuda.is_available().

Class Weights Calculation

- Class weights are computed to handle class imbalance in the dataset. This ensures that the loss function gives more weight to under-represented classes, leading to a more balanced training process.

Optimizer, Loss Function, and Learning Rate Scheduler

- Optimizer: The AdamW optimizer is used, an extension of the standard Adam optimizer with improved handling of weight decay. It's configured with learning rate and weight decay parameters from the config.
- Loss Function: The Cross-Entropy Loss function is employed, with the class weights incorporated to manage class imbalance.
- Learning Rate Scheduler: A Cosine Annealing Learning Rate Scheduler adjusts the learning rate throughout training. This can help avoid local minima.
- Number of Epochs: 100

Gradient Scaling

- To enhance training stability and performance, especially when using mixed-precision training, gradient scaling is applied using GradScaler.

Model Preparation

- The model is loaded onto the designated device (GPU or CPU).
- If a saved model state is found, it's loaded to resume training. Otherwise, training starts from scratch.

Early Stopping and Model Saving

- Early stopping is implemented to prevent overfitting. Training is halted if the validation loss doesn't improve for a specified number (25) of epochs.
- When the model achieves a lower validation loss, its state is saved. This ensures that the best-performing version of the model is retained.

Results and Analysis of Model Training

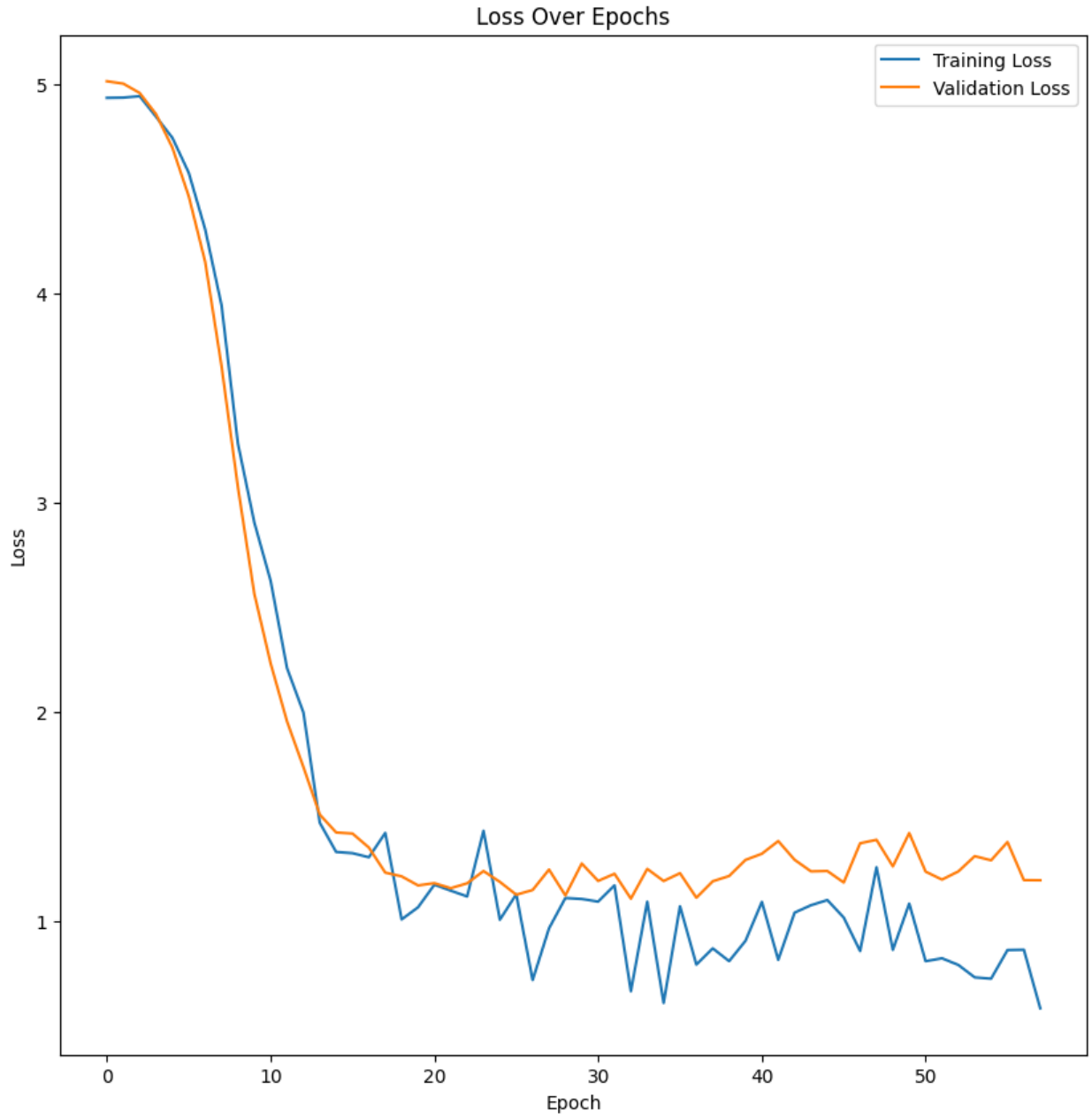
The training and validation loss and accuracy charts visually represent the model's learning progression over time. Here's a detailed analysis based on the provided data and charts:

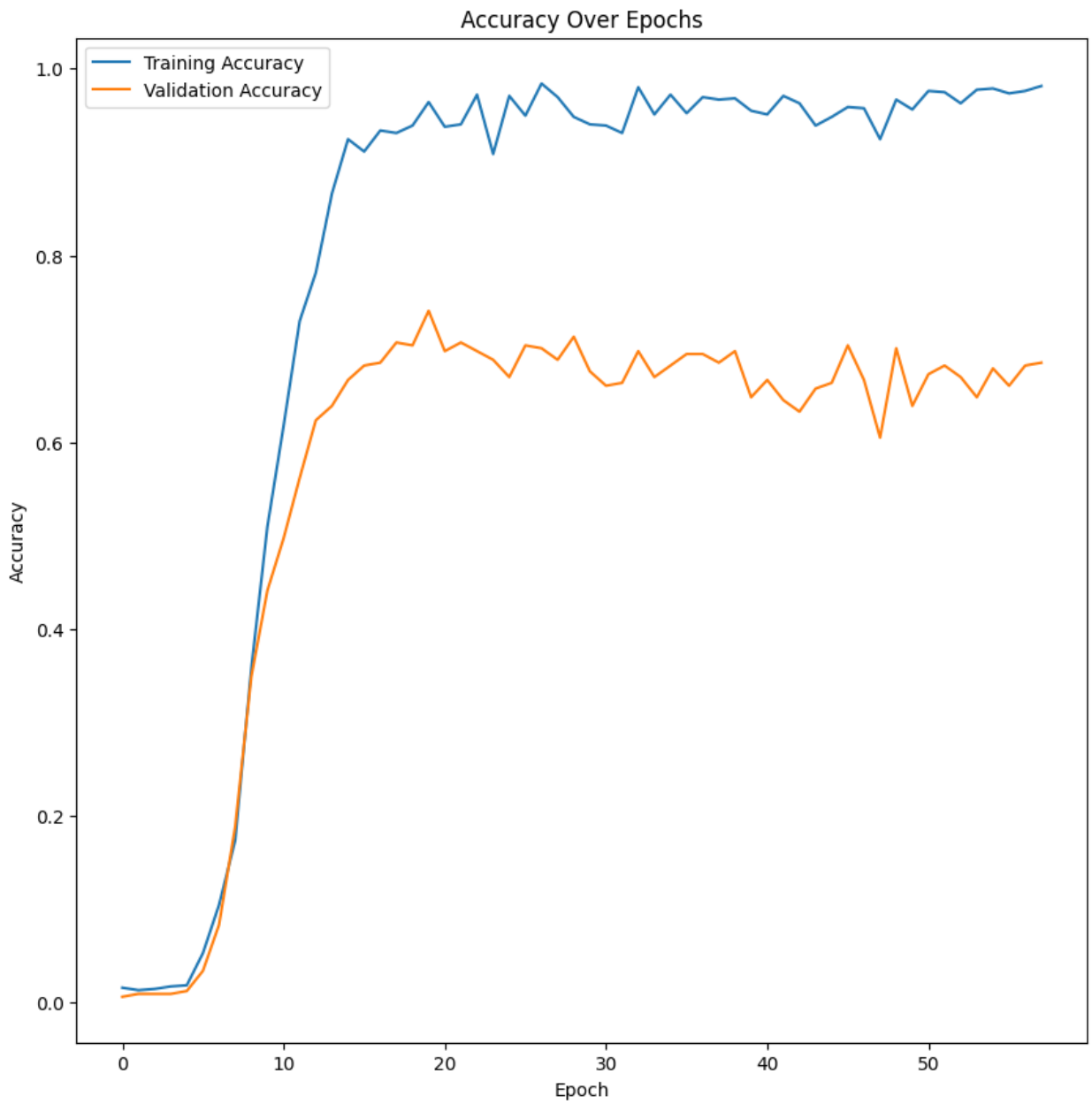
Loss and Accuracy Over Epochs

- Training Loss: The training loss shows a significant decline from the initial epoch, indicating that the model is learning and improving its predictions on the training data.
- Validation Loss: The validation loss also decreases over time, though it exhibits some fluctuations. This is expected as the model is exposed to unseen data, and the fluctuations indicate the model's generalization performance.
- Training Accuracy: There is a steady increase in training accuracy, suggesting that the model fits well with the training data.
- Validation Accuracy: The validation accuracy increases but is less smooth than the training accuracy, which is typical in the training process. While the model is learning generalizable features, there is room for improvement, possibly by fine-tuning or using more complex regularization techniques.

Graphical Analysis

- The graphs indicate that the model does not overfit, as the validation loss decreases alongside the training loss. This is an excellent sign of a well-generalizing model.
- The training accuracy approaches high levels, which indicates the model's capacity to learn from the dataset.




Accuracy Over Epochs
Training Accuracy
Validation Accuracy
Accuracy
Epoch
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
50