# Semi-Supervised Learning with Dual Attention Vision Transformers for Image Classification

**Akashleena Sarkar**
aksarkar@ucsc.edu
**Nick Wang**
nijwang@ucsc.edu

## Abstract

We propose a semi-supervised learning approach for image classification leveraging the Dual Attention Vision Transformer (DaViT). Our work integrates advanced data augmentation, a fine-tuned training setup, and hyperparameter optimization to classify images across 134 categories. Experimental results show that DaViT outperforms ResNet50 and DeiT, achieving a validation accuracy of 91.58%. Future labeling techniques were employed to enhance performance, while ensemble methods are identified as future work. Instructions for reproducing our results and access to the model weights are provided.

## 1 Introduction

Image classification with limited labeled data is a challenging task, especially when the dataset is imbalanced and includes noisy unlabeled data. Our project addresses these challenges using semi-supervised learning techniques. We evaluate the effectiveness of the Dual Attention Vision Transformer (DaViT) by integrating advanced data augmentation, fine-tuning strategies, and hyperparameter optimization.

## 2 Experimental Setup

### 2.1 Dataset

The dataset consists of 134 classes (15 plant categories and 120 dog categories):

- **Labeled Images**: 9854 for training and 1971 for validation.
- **Unlabeled Images**: 22,995.
- **Test Images**: 8213.

### 2.2 Data Augmentation

To address overfitting and improve generalization, we employed:

- **RandomResize**: Uniform image resizing.
- **RandomHorizontalFlip**: Adds robustness to mirrored images.
- **RandomAugment**: Reduces search space while enhancing data variability.
- **Normalize**: Standardization with ImageNet statistics.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

### 2.3 Model Choice

We selected DaViT (base configuration) from the timm library due to its:

- Combination of CNN and transformer features.
- Dual attention mechanism capturing global and local contexts.
- Robustness for image classification tasks.

### 2.4 ResNet50 Baseline Implementation

ResNet50 was used as the initial baseline model due to its proven track record in image classification tasks. Key implementation details include:

- **Pretrained Weights**: Initialized with ImageNet weights
- **Layer Adjustments**: Final fully connected layer replaced to match 134 classes. Layers 3 and 4 were unfrozen for fine-tuning to adapt high-level features to the task.
- **Hyperparameters**:
  - Optimizer: AdamW with learning rate of 1e-4 for the head and 1e-5 for unfrozen layers.
  - Loss Function: CrossEntropyLoss.
  - Scheduler: StepLR reducing the learning rate by 0.1 every 5 epochs.
- **Training Setup**: 15 epochs with a batch size of 32 on an NVIDIA A100 GPU.
- **Data Augmentation**: Standard augmentations such as RandomResize, RandomHorizontalFlip, Color Jitter, and Normalize.

### 2.5 DeiT Baseline Implementation

DeiT (Data-efficient Image Transformer) was selected as a transformer-based baseline to compare against DaViT. Key implementation details include:

- **Pretrained Weights**: Loaded DeiT-base model from the timm library, pretrained on ImageNet.
- **Advanced Data Augmentation**:
  - Gaussian Blur to reduce noise.
  - CutMix for regularization by combining image pairs with interpolated labels.
  - Random Erasing to mask parts of images for robustness.
- **Fine-Tuning Strategy**:
  - Unfroze patch embedding and earlier transformer blocks for task-specific adaptation.
  - Increased dropout rate to 0.3 to combat overfitting.
- **Hyperparameters**:
  - Optimizer: AdamW with learning rate of 3e-4 and weight decay of 1e-4.
  - Scheduler: Cosine Annealing Warm Restarts.
  - Loss Function: Focal Loss to handle class imbalances.
- **Mixed Precision Training**: Enabled GradScaler for efficient GPU memory use.
- **Early Stopping**: Integrated to avoid overfitting, with the best model saved at epoch 10.

### 2.6 Hyperparameters

- **Optimizer**: AdamW for adaptive learning with weight decay of 1e-4.
- **Learning Rates**:
  - Head: 1e-4 (to handle newly initialized parameters).
  - Backbone: 1e-5 (to preserve pretrained weights).
- **Scheduler**: Cosine annealing for gradual learning rate reduction.
- **Criterion**: CrossEntropyLoss for multi-class classification.

### 2.7 Training Setup

Training was performed on an NVIDIA A100 GPU using Python 3.10 and PyTorch 2.0. The learning process included early stopping to prevent overfitting and mixed precision training for efficiency.

## 3 Results

### 3.1 ResNet50

- Validation Accuracy: 85.24%
- Validation Loss: 0.5325

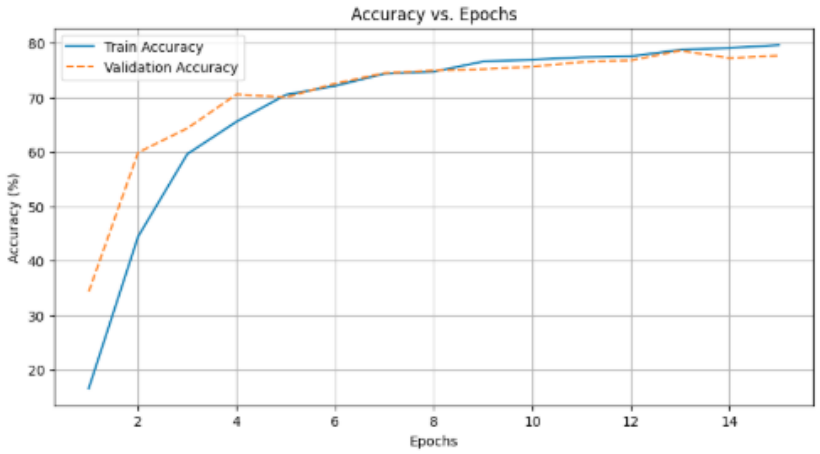


Figure 1: Accuracy for ResNet50 Baseline.

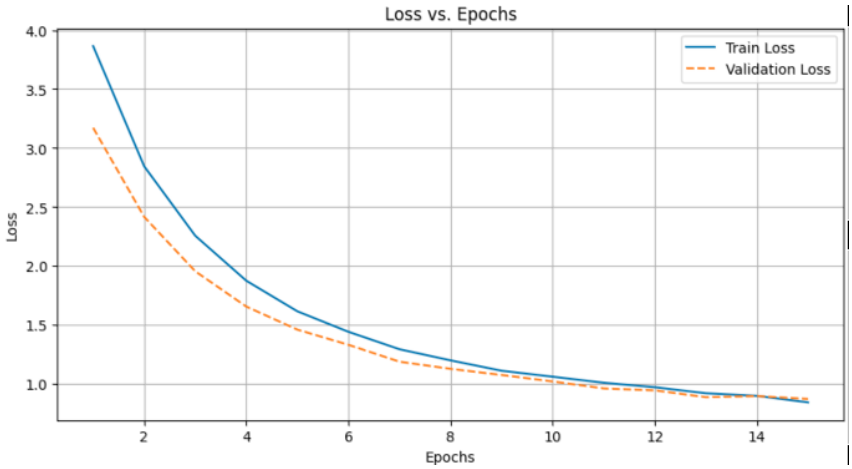


Figure 2: Validation Loss for ResNet50 Baseline.

### 3.2 DeiT

- Validation Accuracy: 88.99%
- Validation F1-Score: 0.8289
- Best Model Saved at Epoch 10.

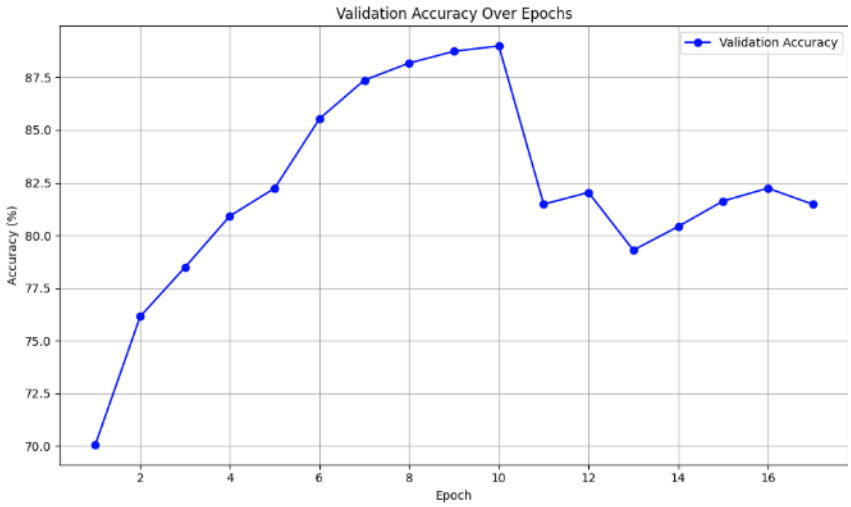


Figure 3: Training and Validation Accuracy for DeiT Baseline.

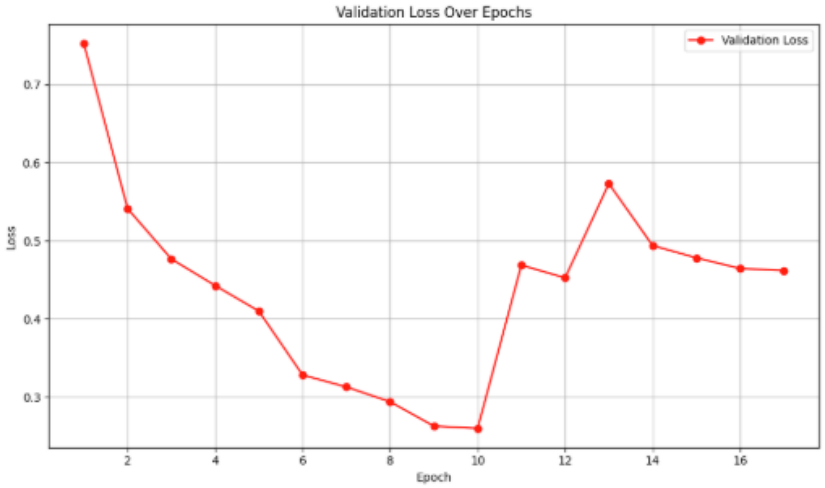


Figure 4: Training and Validation Loss for DeiT Baseline.

### 3.3 DaViT

- Validation Accuracy: **91.58%**
- Validation Loss: 0.3320
- Training Accuracy: 95.08%

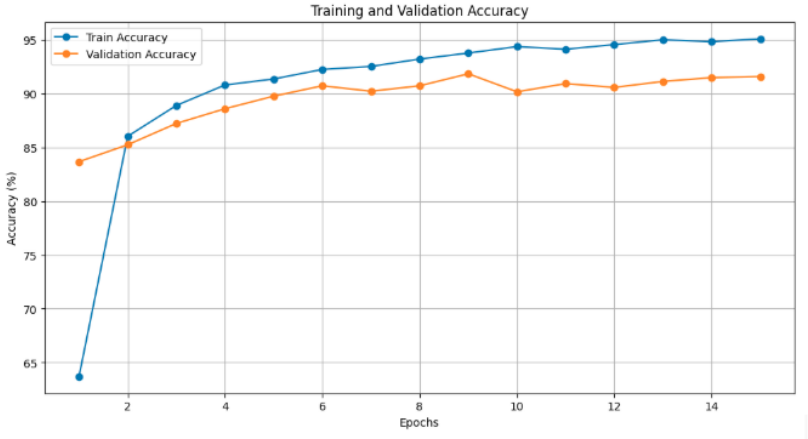


Figure 5: Training and Validation Accuracy for DaViT.


Figure 6: Training and Validation Losses for DaViT.

## 4 Conclusion

From our experimentation, we observe that ResNet50, while effective, achieves a lower accuracy and slower convergence compared to transformer-based models, indicating its limitations in capturing complex global relationships. DeiT outperforms ResNet50 by leveraging advanced augmentations and the attention mechanism, yet it is more sensitive to overfitting. DaViT demonstrates superior performance, achieving the highest accuracy and fastest convergence due to its dual attention mechanism, effectively combining the strengths of CNNs and transformers.

The plots reveal that while ResNet50 converges steadily, its accuracy plateaus early. DeiT shows rapid initial improvement but fluctuates due to sensitivity to hyperparameters. DaViT achieves consistent and smooth improvements across epochs, with lower validation loss and minimal overfitting, showcasing its robustness and adaptability.

## 5 Future Work

- Ensemble methods to further improve accuracy.
- Incorporating pseudo-labeling with the unlabeled dataset.
- Fine-tuning additional layers and hyperparameters.
- Exploring additional data augmentation strategies.

## 6 Reproducibility Instructions

1. Access the Kaggle competition and ensure the presence of a `kaggle.json` file.
2. Clone the project repository and mount Google Drive.
3. Run the provided Colab notebook, which includes:
   - Importing required packages.
   - Data preprocessing and augmentation.
   - Model definition and hyperparameter initialization.
   - Training and validation with loss/accuracy plots.
   - Saving test results to a CSV file.
4. Access the final model weights at: `https://drive.google.com/file/d/1-JCgbkOEHwbHt2tb75Nqt5n5p_o8CxNR/view?usp=share_link`.

## Acknowledgments

We thank Professor Yujin Zhou for guidance and access to Google Cloud resources, and TA Siwei Yang for technical support.