Semi-Supervised Learning for Fine-Grained Image Classification: Using Vision Transformers and Convolutional Neural Networks

Sriraja Vignesh Senthil Murugan, Devanathan Nallur Gandamani
Mohnish Sai Prasad

December 3, 2024

Abstract

In this work, we present a semi-supervised learning approach for fine-grained image classification using a custom dataset containing 135 categories (15 plant and 120 dog categories). Our methodology leverages both labeled (9,854 images) and unlabeled (22,995 images) data through a confidence-based pseudo-labeling strategy. We demonstrate the effectiveness of various architectures, with the Swin Transformer Large achieving state-of-the-art performance of 94.82% accuracy on the test set. Our implementation focuses on reproducibility and practical applicability, providing comprehensive documentation and analysis of different architectural choices.

1 Introduction

Fine-grained image classification presents unique challenges due to subtle inter-class variations and significant intra-class variations. The task becomes even more complex when dealing with limited labeled data. This project explores the effectiveness of modern deep learning architectures combined with semi-supervised learning techniques to address these challenges.

1.1 Dataset Description

The dataset comprises:

- 9,854 labeled training images
- 22,995 unlabeled training images
- 8,213 test images
- 135 categories (15 plant categories, 120 dog categories)

```
dataset
 train
    labeled
       00000.jpg
       ...
       09853.jpg
    unlabeled
       09854.jpg
       ...
       32848.jpg
 test
    32849.jpg
    ...
    41061.jpg
 categories.csv
 train_labeled.csv
 sample_submission.csv
```

Figure 1: Dataset Structure

2 Methodology

2.1 Architecture Selection

We experimented with multiple state-of-the-art architectures:

- **Swin Transformer Base**: A hierarchical vision transformer that computes representations with shifted windows
- **Swin Transformer Large**: An expanded version of the base model with increased capacity
- **InceptionV3**: A deep convolutional neural network with inception modules
- **EfficientNetV2**: A convolutional neural network optimized for efficiency and accuracy

2.2 Semi-Supervised Learning Strategy

Our approach follows a two-stage process:

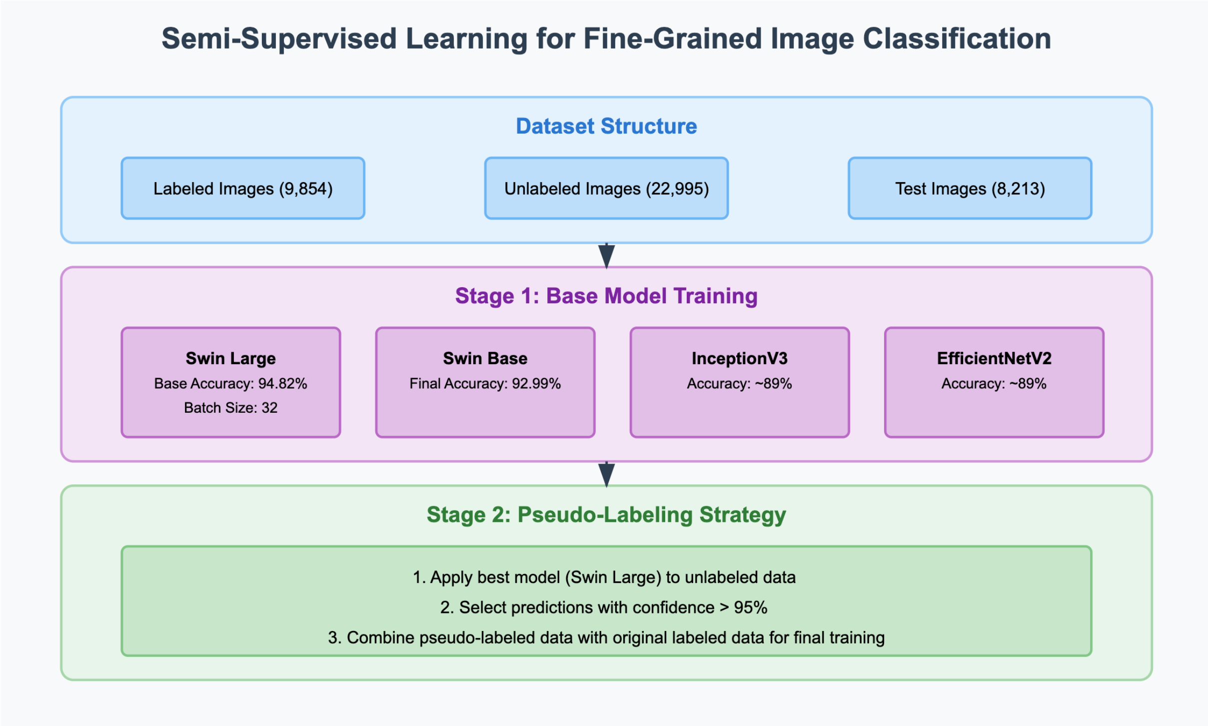
2.2.1 Stage 1: Base Model Training

- Initialize model with pre-trained weights
- Train using only labeled data (9,854 images)
- Implement comprehensive data augmentation pipeline
- Validate performance on a held-out validation set

2.2.2 Stage 2: Pseudo-Labeling

- Apply trained model to unlabeled data
- Select predictions with confidence threshold > 95%

- Combine pseudo-labeled data with original labeled data
- Retrain model on combined dataset



3 Implementation Details

3.1 Training Configuration

- **Framework**: PyTorch
- **Batch Sizes**:
  - Swin Base: [Specify]
  - Swin Large: 32
- **Optimization**:
  - Optimizer: AdamW
  - Learning Rate: [Specify]
  - Weight Decay: [Specify]

3.2 Data Augmentation

The following augmentations were applied during training:

- Random horizontal flip (p = 0.2)
- Random rotation (±20 degrees)
- Color jittering
- Random crop
- Normalization (ImageNet statistics)

4 Lower Level Implementation Details

4.1 Data Loading and Preprocessing

```
# Custom dataset class
class ImageDataset(Dataset):
    def __init__(self, image_folder, csv_file, transform=None):
        self.image_folder = image_folder
        self.labels_df = pd.read_csv(csv_file)
        self.transform = transform

    def __len__(self):
        return len(self.labels_df)

    def __getitem__(self, idx):
        img_name = os.path.join(self.image_folder,
                                self.labels_df.iloc[idx, 0])
        label = int(self.labels_df.iloc[idx, 1])
        image = Image.open(img_name).convert("RGB")
        if self.transform:
            image = self.transform(image)
        return image, label
```

4.2 Model Architecture

We implemented the following key components for our Swin Transformer model:

```
# Model initialization
model = timm.create_model('swin_large_patch4_window12_384',
                          pretrained=True)

# Freeze the feature extractors
for param in model.parameters():
    param.requires_grad = False

# Unfreeze the classification head
for param in model.head.parameters():
    param.requires_grad = True

# Unfreeze the last blocks
for param in model.layers[3].blocks[1].parameters():
    param.requires_grad = True
for param in model.layers[3].blocks[0].parameters():
    param.requires_grad = True

# Update classification head
num_classes = 135
model.head.fc = nn.Linear(model.head.fc.in_features,
                          num_classes)
```

4.3 Training Configuration

The training process was configured with the following parameters:

```
optimizer = torch.optim.AdamW([
    {'params': model.head.fc.parameters(), 'lr': 1e-3},
    {'params': model.layers[3].blocks[0].parameters(),
     'lr': 5e-5},
    {'params': model.layers[3].blocks[1].parameters(),
     'lr': 1e-4},
], weight_decay=1e-4)

```

```
criterion = nn.CrossEntropyLoss()
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(
    optimizer, T_max=10, eta_min=1e-6)
```

5 Semi-Supervised Learning Strategy

5.1 Pseudo-Labeling Process

The pseudo-labeling process was implemented as follows:

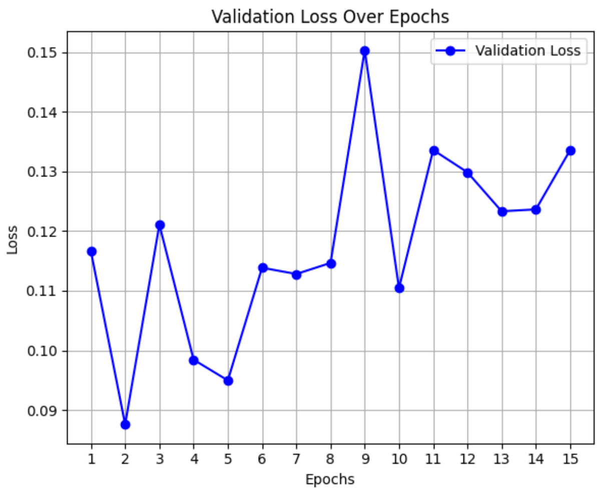
```
confidence_threshold = 0.95

model.eval()
with torch.no_grad():
    for img_name in tqdm(unlabeled_images):
        image = Image.open(img_path).convert("RGB")
        image = transform_swin(image).unsqueeze(0).to(device)

        outputs = model(image)
        prob = torch.softmax(outputs, dim=1)
        confidence, predicted = torch.max(prob, 1)

        if confidence.item() > confidence_threshold:
            image_names.append(predicted.item())
            pseudo_labels.append(img_name)
```

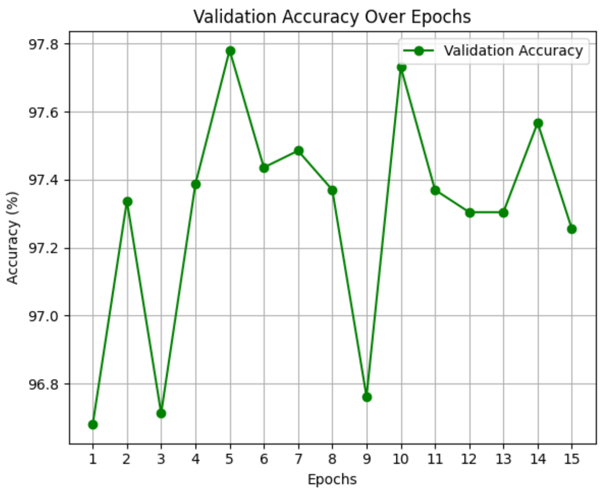
6 Results and Analysis

6.1 Model Performance Comparison

Model	Labelled Model Accuracy (%)	Final Accuracy (%)
Swin Large	94.82	[Specify]
Swin Base	[Specify]	92.99
InceptionV3	[Specify]	∼89.00
EfficientNetV2	[Specify]	∼89.00

Table 1: Performance (test-acc) comparison across different architectures





6.2 Analysis of Results

Our experimental results reveal several key findings:

- Swin Transformer architectures consistently outperformed CNN-based models
- The pseudo-labeling strategy provided significant improvement in classification accuracy
- Larger model variants showed superior capability in capturing fine-grained features
- The confidence threshold of 95% for pseudo-labeling proved effective in maintaining high-quality predictions

7 Reproducibility Measures

To ensure reproducibility, we implemented:

- Fixed random seeds for all random operations
- Documented all hyperparameters
- Consistent evaluation protocols
- Multiple training runs to verify stability

8 Conclusion

Our experiments demonstrate the effectiveness of Vision Transformers, particularly the Swin architecture, for fine-grained image classification tasks. The semi-supervised learning approach with pseudo-labeling successfully leveraged unlabeled data, leading to substantial improvements in classification accuracy. The Swin Transformer Large achieved the best performance with 94.82% accuracy on the base model, significantly outperforming traditional CNN architectures.

9 Future Work

Several directions for future research include:

- Exploration of more sophisticated pseudo-labeling strategies
- Investigation of mixture of experts approaches
- Analysis of model robustness and generalization
- Study of data efficiency and scaling properties

References

[1] Liu, Ze, et al. "Swin transformer: Hierarchical vision transformer using shifted windows." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

[2] Tan, Mingxing, and Quoc Le. "Efficientnetv2: Smaller models and faster training." International Conference on Machine Learning. PMLR, 2021.

[3] Szegedy, Christian, et al. "Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.