Analysis of Fine-Grained Classification Performance using Image Transformers

Rohith Ravindranath
Sunnyvale, USA
rohithravin@gmail.com

***Abstract*—In recent years, Vision Transformers (ViTs) have revolutionized the field of computer vision, offering a powerful alternative to traditional Convolutional Neural Networks (CNNs) for image classification tasks. This project investigates the performance of various pre-trained Vision Transformer models, including Google's ViT, Facebook's DeiT, and Microsoft's Swin, specifically in the context of fine-grained image classification. Using the Stanford Dog Dataset, we applied transfer learning techniques to fine-tune these models and evaluated their performance based on accuracy, precision, recall, and F1 score.**

Our results indicate that while the ViT model exhibited superior validation metrics and faster convergence rates, the DeiT model outperformed it on the test set, suggesting better generalization capabilities and robustness to unseen data. Notably, the Swin Base model performed unexpectedly worse than the Swin Small model, raising questions about the scalability and optimization of larger Swin architectures. This study not only highlights the potential of transformer-based models in fine-grained classification tasks but also provides insights into their comparative performance and practical implications for model selection and deployment.

***Index Terms*—Image Transformers, Deep Learning, Transfer Learning, Fine-Tuning**

I. Introduction

In recent years, the field of computer vision has undergone a transformative shift with the emergence of Vision Transformers (ViTs), which have demonstrated remarkable capabilities in image classification tasks traditionally dominated by Convolutional Neural Networks (CNNs). This project focuses on evaluating the performance of various pre-trained ViT models specifically for fine-grained image classification tasks.

Fine-grained image classification presents a unique challenge in distinguishing between closely related categories within a broader class, such as different species of birds, types of flowers, or models of cars. The precision required for such tasks is critical, as misclassification can have significant implications across diverse fields including healthcare, agriculture, biodiversity monitoring, and manufacturing.

The primary objective of this project is to leverage transfer learning techniques to assess performance on various Image Transformer models for fine-grained image classification. This report outlines the methodology employed, the datasets used for evaluation, and the results obtained from comparative analysis.

II. Transfer Learning

Transfer learning is a pivotal technique in machine learning and deep learning, particularly beneficial when dealing with tasks where labeled data is limited or expensive to acquire. It involves leveraging knowledge gained from solving one problem and applying it to a different, but related, problem domain. This approach can significantly enhance model performance and efficiency by transferring learned features or representations from a pre-trained model to a new task.

A. When to Use Transfer Learning

Transfer learning is particularly advantageous in the following scenarios:

- Limited Data Availability: When the target task lacks sufficient labeled data, transfer learning allows models to generalize better by utilizing knowledge learned from a related task with abundant data.
- Computational Efficiency: Training deep neural networks from scratch can be computationally intensive. Transfer learning enables the reuse of pre-trained models, reducing training time and resource requirements.
- Domain Adaptation: When the distribution of data in the target task differs slightly from the source task, transfer learning helps in adapting the model's representations to better suit the new domain.

B. Fine-Tuning vs. Transfer Learning

Fine-tuning is a specific application of transfer learning where a pre-trained model, typically trained on a large-scale dataset like ImageNet, is adapted to a new task by adjusting its parameters with the new task's dataset.

Advantages of Fine-Tuning:

- Task-Specific Adaptation: Fine-tuning allows models to adapt more closely to the nuances of the target task by adjusting weights in the later layers of the network while retaining the basic feature extraction capabilities learned from the pre-trained model.
- Efficiency: Since fine-tuning modifies only a portion of the model, it often requires less training data and computational resources compared to training from scratch.

Disadvantages of Fine-Tuning:

- Overfitting Risk: Fine-tuning can lead to overfitting when the target dataset is small, as the model may excessively adjust to noise or specific characteristics of the training data.
- Task Dependency: The success of fine-tuning depends on how closely related the source task (pre-training dataset)

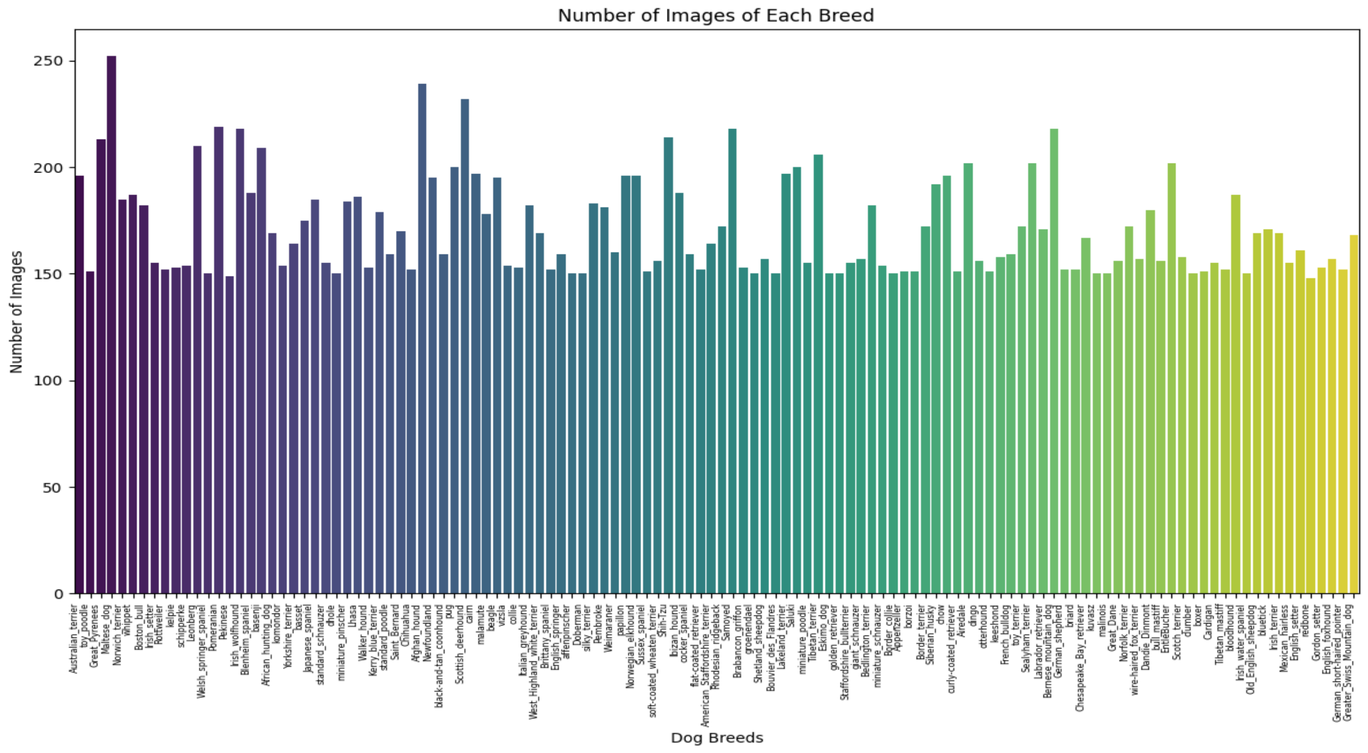

Fig. 1. Distribution of the number of images for each label

is to the target task. In cases of vastly different tasks, fine-tuning might not yield optimal results.

C. How Transfer Learning Works

During transfer learning, the layers of a pre-trained model are typically divided into two parts: the feature extractor and the classifier. The feature extractor, which comprises the initial layers of the model, learns to extract generic features from the input data (such as edges, textures, and shapes). These features are considered transferable because they are generally applicable across different tasks and domains.

When applying transfer learning, the feature extractor of the pre-trained model is often kept frozen or only partially modified, retaining the learned representations. This allows the model to focus on adapting the higher-level features or the classifier layers to the specifics of the target task. By fine-tuning these higher layers using the target task's dataset, the model can learn task-specific patterns and improve its performance.

In the context of my project evaluating Image Transformer models for fine-grained image classification, transfer learning will be pivotal. We will initialize Image Transformer models with weights pre-trained on a large dataset like ImageNet. The initial layers of the Image Transformers will retain their learned visual features, while the later layers, responsible for more abstract representations and classification, will be fine-tuned on our specific fine-grained classification dataset. This approach leverages the generalization capabilities of the pre-trained model while adapting it to excel in our targeted classification tasks.

III. Dataset

For this project, we will be using the Stanford Dog Dataset. It is a large-scale dataset containing annotated images of dogs belonging to 120 different breeds. Created by researchers at Stanford University, this dataset is widely used for training and evaluating computer vision models in tasks related to fine-grained image classification and object recognition. Each breed in the dataset is represented by a varying number of images, capturing diverse poses, backgrounds, and lighting conditions. The Stanford Dog Dataset serves as a benchmark for testing the ability of models to distinguish between visually similar dog breeds, making it invaluable for advancing research in computer vision and machine learning.

The dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet. Contents of this dataset:

- Number of categories: 120
- Number of images: 20,580

Figure 1 shows the distribution of the number of images for each label in the dataset. There is on average about 150 images per label. Figure 2 show examples of the variety of images in the dataset (i.e. diverse poses, backgrounds, and lighting conditions).

Fig. 2. Example Images from the Stanford Dog Dataset

IV. Models

In this project, I used a variety of image transformers and different sizes to evaluate their performance. Specifically, the three main types of image transformers I used were Google's ViT (vit_base_patch16_224), Facebook's DeiT (deit_tiny_patch16_224, deit_small_patch16_224, deit_base_patch16_224), and Microsoft's Swin (swin_tiny_patch4_window7_224, swin_small_patch4_window7_224, swin_base_patch4_window7_224).

A. *Google's ViT*

Google's Vision Transformer (ViT) represents a groundbreaking advancement in the field of computer vision, introduced by researchers at Google Research. Unlike traditional Convolutional Neural Networks (CNNs), which have long dominated image processing tasks, ViT adopts a transformer architecture originally designed for natural language processing tasks.

The core innovation of ViT lies in its ability to process images as sequences of tokens, similar to how transformers handle sequences of words in text. This approach allows ViT to capture long-range dependencies and relationships within images effectively, without relying on predefined hierarchical features extracted by CNNs.

Key features of Google's ViT include its adaptability to different image sizes through tokenization, scalability to handle large datasets, and the capability to learn complex visual patterns directly from raw pixel data. By leveraging self-attention mechanisms, ViT excels in capturing global context and fine-grained details crucial for tasks such as image classification, object detection, and segmentation.

Google's ViT has demonstrated state-of-the-art performance on various benchmark datasets, showcasing its potential to surpass traditional CNNs in accuracy and efficiency for a wide range of computer vision applications. Its development marks a significant milestone in advancing the capabilities of deep learning models beyond established paradigms, paving the way for new possibilities in visual understanding and analysis.

B. *Facebook's DeiT*

Facebook's DeiT, or Data-efficient image Transformers, represents a pioneering approach to leveraging transformer architectures for image classification tasks. Developed by researchers at Facebook AI, DeiT aims to achieve high performance in image recognition while requiring less labeled data compared to traditional approaches.

Unlike ViT, which initially relied on large-scale pre-training on datasets like ImageNet, DeiT focuses on improving data efficiency by training on smaller datasets. It achieves this through a novel method called distillation, where knowledge from a teacher model (such as a large-scale ViT) is transferred to a smaller, student DeiT model.

Key features of DeiT include its ability to distill knowledge from a teacher model's learned representations, enabling effective transfer of visual understanding even with limited annotated data. By adopting a distillation strategy, DeiT can generalize well across diverse image classification tasks, making it more accessible for applications where collecting extensive labeled datasets is challenging.

Facebook's DeiT has demonstrated competitive performance on benchmark datasets, showcasing its potential to achieve state-of-the-art results with fewer computational resources and labeled examples. This approach not only advances the field of image classification but also underscores the importance of efficient and scalable deep learning models in real-world applications.

C. *Microsoft Swin*

Microsoft's Swin Transformer (Swin) is a cutting-edge architecture designed to handle computer vision tasks with a focus on scalability and efficiency. Introduced by Microsoft Research, Swin Transformer builds upon the transformer model's success in natural language processing and adapts it to the domain of image processing.

Swin Transformer introduces a hierarchical structure where the image is divided into non-overlapping patches, which are then processed by multiple transformer layers. Unlike traditional transformers or CNNs, Swin Transformer dynamically partitions the image into smaller regions, allowing it to capture both local and global dependencies effectively.

Key features of Microsoft's Swin Transformer include its ability to scale efficiently with image size and dataset complexity, making it suitable for a wide range of tasks from image classification to object detection and segmentation. By leveraging hierarchical processing and self-attention mechanisms, Swin Transformer excels in capturing fine-grained details and long-range dependencies within images.

Swin Transformer has demonstrated state-of-the-art performance on various benchmark datasets, showcasing its capability to surpass traditional CNNs and rival other transformer-based architectures like ViT. Its development represents a significant advancement in advancing the frontiers of computer vision, offering robust solutions for tasks requiring comprehensive visual understanding and analysis.

TABLE I
TRAINING TIME FOR EACH MODEL

Model Name	Time (hrs)
ViT Base	8.367
DeiT Base	12.467
DeiT Small	4.776
DeiT Tiny	2.65
Swin Base	114.35
Swin Small	12.267
Swin Tiny	7.133

TABLE II
TEST METRICS FOR EACH MODEL

Model	Accuracy	Precision	Recall	F1 Score
ViT Base	0.910	0.910	0.910	0.904
DeiT Base	**0.934**	**0.938**	**0.934**	**0.933**
DeiT Small	0.890	0.896	0.890	0.890
DeiT Tiny	0.756	0.766	0.756	0.754
Swin Base	0.834	0.851	0.834	0.830
Swin Small	0.885	0.893	0.885	0.881
Swin Tiny	0.722	0.747	0.723	0.711

V. METHODS

A. Image Preprocessing

With respect to image processing, all images were resized to 224 x 224 pixels, converted to Pytorch tensors and normalize using Imagneet mean and standard deviation. I used Imagneet mean and standard deviation as all the Image Transformers used in this project were pre-trained on the ImageNet dataset.

Figure 3 shows examples of how the images looks after all the transformations are applied.

B. Training

In terms of training, all models were trianed used the same parameteres in order to maintain consistency and comparable to each other. Models were trained over 100 epochs with EarlStopping (patience = 5). I used an Adam Optimizer with an initial learning rate of 5e-5 and had a ReduceLROnPlateau scheduler with a patience of 3 and a factor of 0.5. The loss function was CrossEntropyLoss. The pre-train model parameters were kept frozen. The output head of the models were replace with a Linear layer that was trainable. The dataset was split into 80:10:10 train/val/test split.

VI. RESULTS

A. Rate of Convergence

Figure 4 illustrates the convergence rates of various models by depicting the loss function over each epoch. From this plot, it is evident that the ViT Base model achieved the best rate of convergence. Notably, the ViT Base model also stopped training earlier, suggesting that it reached convergence much faster than the other models.

Following the ViT Base, the DeiT models demonstrated impressive convergence rates. DeiT models are specifically designed for efficient training of vision transformers (ViTs), which is reflected in their rapid convergence compared to some traditional methods.

Interestingly, the Swin Base and Swin Small models exhibited similar convergence rates. This is surprising because one might expect the base model, with more parameters, to perform better and converge faster than the smaller model.

B. Training Time

Table I shows the training duration for each model on the downstream task. A few notable observations emerge from this data. Firstly, among all the base models, ViT Base was the fastest to train, whereas Swin Base took the longest by a significant margin.

The extended training time for Swin models (Base, Tiny, and Small) can be attributed to several factors. One key reason is their inherently complex architecture, which employs a hierarchical structure with shifted windows, adding complexity and increasing computational overhead. Additionally, the Swin models' use of multi-scale feature representation, where the model builds feature maps at different stages, enhances performance by capturing various levels of abstraction but also escalates the computational cost and training time.

As expected, the training times decrease with model size: Small models train faster than Base models, and Tiny models have the quickest training times.

C. Validation Metrics

Figure 5 shows how the metrics improve over time during training. Overall, the results are what we expect, we all the models showing improvement over time. ViT seems to be showing the best performance across all metrics. Interestingly Swin Base performs worse than Swin Small in all the metrics. Swin Base has simillar performance to that of DeiT Tiny, which is surprising. This could be due to the fact the training settings where not optimized for each model individually.

D. Test Metrics

Table II and Figure 6 illustrate the performance of each model on the test set, evaluated using Accuracy, Precision, Recall, and F1 Score metrics. Notably, the DeiT Base model achieved the highest overall performance, surpassing the ViT Base model. This is particularly interesting given that ViT Base had better validation scores and faster convergence during training. This suggests that DeiT Base may generalize better to unseen data and be more robust to variations outside the training distribution.

Interestingly, Swin Base performed significantly worse than Swin Small, which could indicate potential issues with scaling or model optimization at larger sizes within the Swin architecture. As anticipated, the smaller models, specifically Swin Tiny, delivered the poorest performance, with Swin Tiny being the worst overall.

VII. CONCLUSION

Overall, this project focused on analyzing the performance of image transformers on a fine-grained classification task. The findings highlighted that while ViT achieved better validation

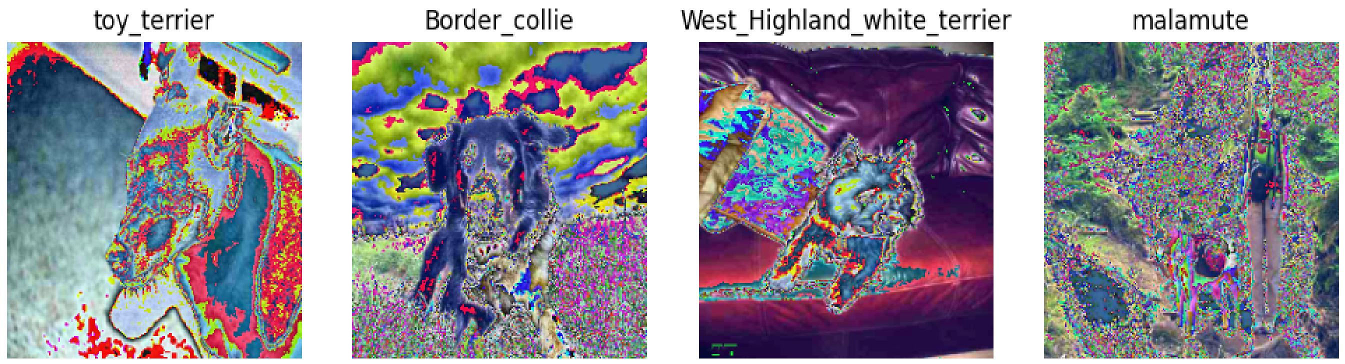

Fig. 3. Example Images Post Pre-Processing

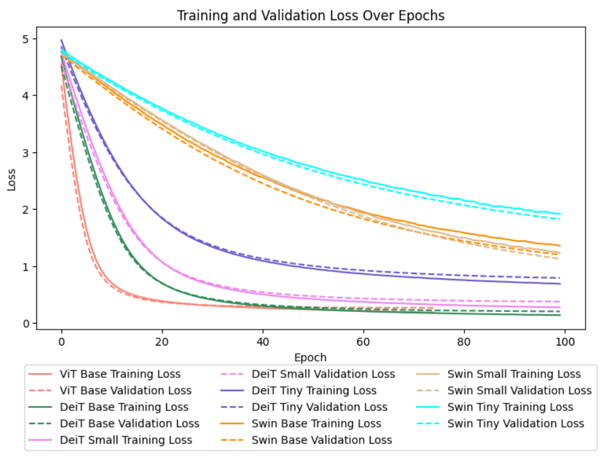

Fig. 4. Example Images from the Stanford Dog Dataset

metrics and a faster rate of convergence, DeiT demonstrated greater robustness and superior performance on the test set.

Throughout the project, I gained valuable hands-on experience in several key areas:

- Fine-Tuning Transformer Models: I learned how to fine-tune transformer models for specific tasks using HuggingFace's Transformers library and PyTorch. This included setting up the training pipeline, optimizing hyperparameters, and managing computational resources efficiently.
- Model Evaluation and Analysis: I developed skills in evaluating model performance using various metrics such as Accuracy, Precision, Recall, and F1 Score. I also learned to interpret these metrics to understand the strengths and weaknesses of different models, particularly in terms of generalization and robustness.
- Handling Fine-Grained Classification Tasks: I gained insights into the challenges associated with fine-grained classification tasks, such as dealing with subtle differences between classes and ensuring the model can generalize well to new, unseen data.
- Practical Use of Computational Tools: I improved my proficiency with tools and libraries essential for machine learning workflows, including PyTorch for model training and HuggingFace for leveraging pre-trained transformer models.
- Code Management and Reproducibility: I learned the importance of maintaining clean, well-documented code and using version control (GitHub) to ensure reproducibility and facilitate collaboration. This included organizing code in a way that others can easily understand and reproduce the results.
- Critical Analysis of Model Architectures: By comparing different transformer architectures (ViT, DeiT, Swin), I learned how architectural choices impact model performance. This included understanding the trade-offs between model complexity, convergence speed, and generalization ability.

All of my code, along with the model checkpoints for each model, is available on GitHub. This project not only enhanced my technical skills but also deepened my understanding of transformer-based models and their applications in fine-grained classification tasks.

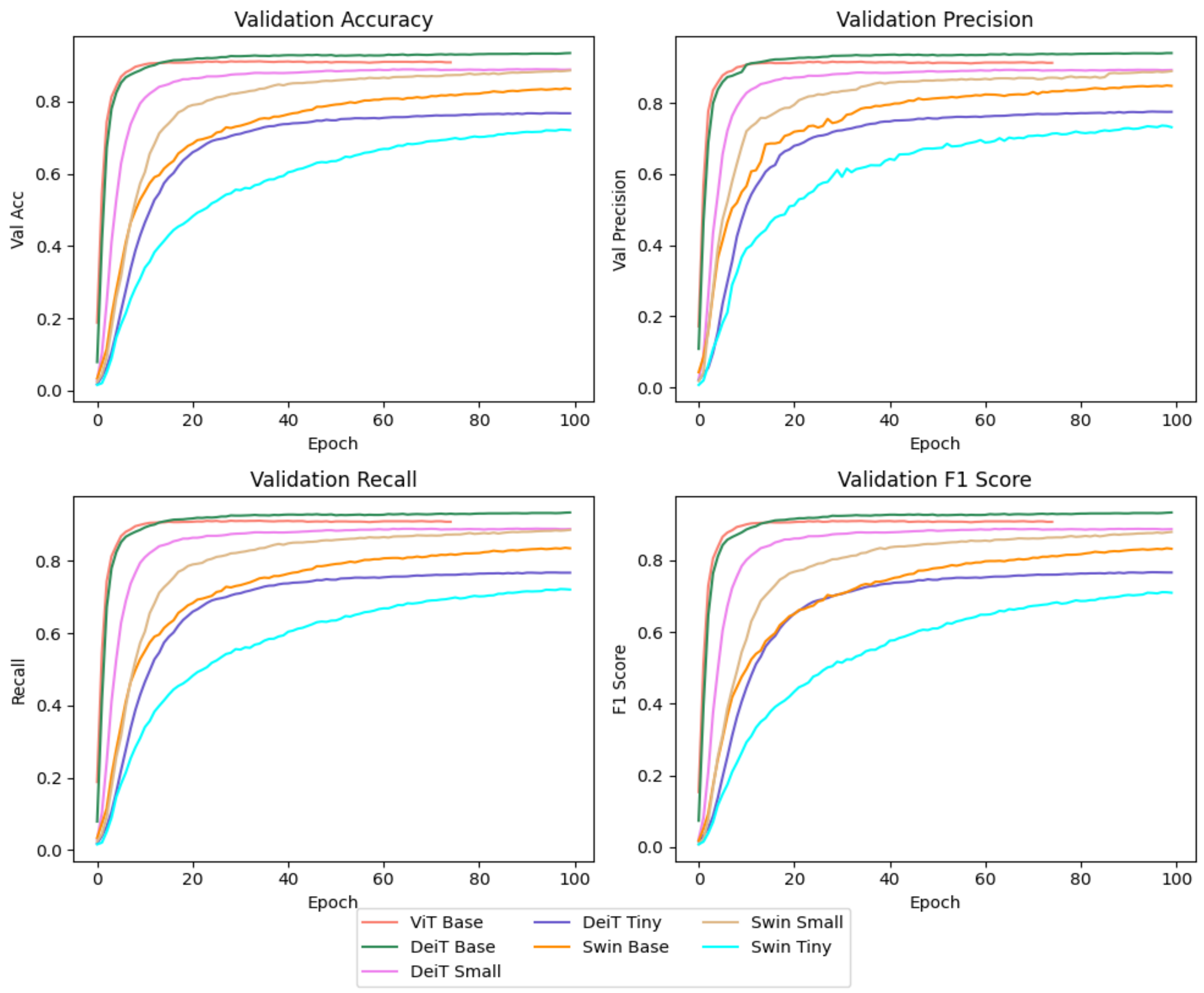

Fig. 5. Validation Metrics Over Time

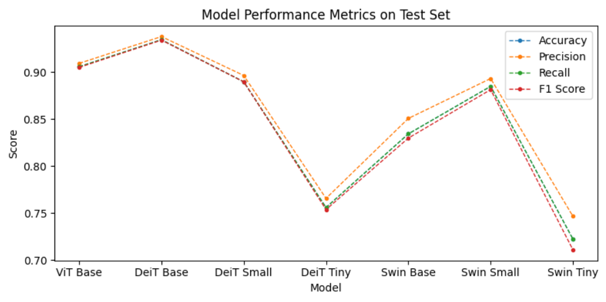

Fig. 6. Plotted Test Metrics for Each Model