A Review of Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

ECBM E4040
Final Project
Fall 2023

Abhik Biswas
Data Science
Columbia University
New York City, USA
ab5640@columbia.edu

Abhilash Praveen Kumar
Electrical Engineering
Columbia University
New York City, USA
ap4478@columbia.edu

Aarya Raghavan
Biomedical Engineering
Columbia University
New York City, USA
ar4634@columbia.edu

***Abstract*— The Swin Transformer, a breakthrough in computer vision, presents a hierarchical architecture designed to address challenges in processing visual information at various scales. Unlike traditional transformer designs, the Swin Transformer adopts a shift-based window mechanism, allowing the model to efficiently capture long-range dependencies. Its hierarchical structure incorporates a multi-scale processing approach, enabling effective feature extraction from both local and global contexts. This innovative approach not only enhances the model's ability to comprehend complex visual patterns but also provides scalability for handling diverse and intricate datasets. This study involves the implementation of the original paper, with different Swin Transformer architectures and evaluates its performance on an image dataset from Kaggle.**

Keywords—transformer, shift based window, scalability

I. INTRODUCTION

The realm of computer vision has predominantly been shaped by convolutional neural networks (CNNs), starting from the groundbreaking performance of AlexNet in the ImageNet image classification challenge. Over time, CNN architectures have progressed by increasing scale, incorporating extensive connections, and employing advanced convolution techniques. CNNs have established themselves as foundational networks for various vision tasks, resulting in widespread performance enhancements across the field[1].

In contrast, natural language processing (NLP) has seen a different trajectory in terms of network architectures, with the transformer becoming the prevailing model today. Originally designed for sequence modeling and transduction tasks, the Transformer stands out for its utilization of attention mechanisms to capture long-range dependencies in data. Its remarkable success in language-related tasks has prompted researchers to explore its adaptation to computer vision, where it has recently shown promise in specific areas such as image classification and joint vision-language modeling.

The research paper aims to broaden the Transformer's utility as a versatile backbone for computer vision, akin to its role in NLP and the role of CNNs in vision tasks. The challenges in achieving seamless performance transfer from the language to the visual domain are attributed, in part, to notable differences between these modalities. One such difference, as identified in our observations, revolves around the concept of scale.

In contrast to language Transformers, which process basic units in the form of word tokens, visual elements exhibit significant scale variations, a challenge particularly addressed in tasks like object detection. Existing Transformer-based models have fixed-scale tokens, a characteristic unsuitable for accommodating the diverse scales present in vision applications. Another distinction arises from the considerably higher pixel resolution in images compared to the words in textual passages[2]. Vision tasks, such as semantic segmentation, necessitate pixel-level dense prediction, a demand that proves impractical for Transformers on high-resolution images due to the quadratic computational complexity of self-attention with respect to image size.

To address these challenges, the research paper introduces a Transformer backbone named Swin Transformer. This model constructs hierarchical feature maps and maintains linear computational complexity concerning image size, providing a solution to the issues encountered in adapting Transformers to diverse visual tasks.

As depicted in Figure 1(a), Swin Transformer establishes a hierarchical representation starting from smaller patches (outlined in gray), progressively merging neighboring patches

in deeper layers of the Transformer. This hierarchical feature map structure enables the Swin Transformer model to effectively utilize advanced techniques for dense prediction, such as feature pyramid networks (FPN) or U-Net. The model achieves linear computational complexity by locally computing self-attention within non-overlapping windows that divide the image (outlined in red). The fixed number of patches in each window ensures linear complexity relative to the image size. These advantages position Swin Transformer as a versatile backbone for various vision tasks, in contrast to earlier Transformer-based architectures that yield feature maps with a single resolution and exhibit quadratic complexity.

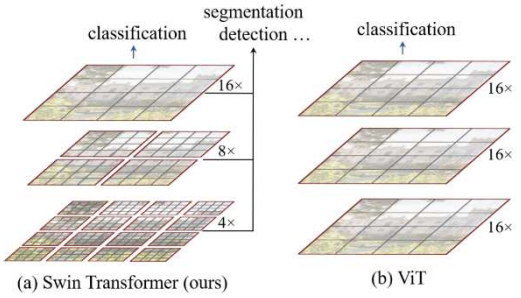


Figure 1. (a) The Swin Transformer proposed in this work constructs hierarchical feature maps by merging image patches (illustrated in gray) in deeper layers. Its distinctive feature lies in its linear computational complexity with respect to the input image size. This efficiency is achieved by limiting the computation of self-attention to each local window (depicted in red). Consequently, the Swin Transformer is well-suited to serve as a versatile backbone for various tasks, including both image classification and dense recognition tasks. (b) In comparison, prior vision Transformers generate feature maps with a singular low resolution, and their computation complexity with respect to the input image size is quadratic. This arises from the global computation of self-attention across the entire image. The contrast with the proposed Swin Transformer, which maintains linear complexity by employing local window-based self-attention, highlights the efficiency and versatility achieved by the proposed architecture[1].

A pivotal design feature of the Swin Transformer involves the shifting of window partitions between consecutive self-attention layers, as depicted in Figure 2. These shifted windows act as connectors, spanning across the windows of the previous layer and establishing connections among them, significantly boosting modeling capabilities (refer to Table 4 for details). This approach also proves efficient in terms of real-world latency: all query patches within a window share the same key set, streamlining memory access in hardware. In contrast, earlier self-attention approaches based on sliding windows encounter latency issues on general hardware due to varying key sets for different query pixels[3].

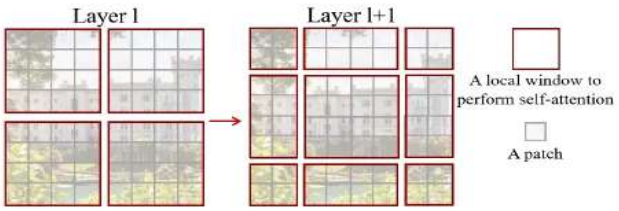


Figure 2. An illustration of the shifted window approach

The experiments demonstrate that the suggested shifted window approach not only exhibits significantly lower latency compared to the sliding window method but also maintains a comparable level of modeling power. This shifted window strategy is also advantageous when applied to all-MLP architectures. The proposed Swin Transformer attains robust performance on recognition tasks of image classification.

In our implementation, the Tiny ImageNet dataset, CIFAR100 and Flower Classification is used to depict how the Swin Transformer is a better model for classification when compared to CNNs and other networks.

II. RELATED WORK

The research paper describes different models and their performance with image classification in comparison to Swin Transformer.

CNN and variants CNNs and their variations have long been the go-to network models in the field of computer vision. Despite the CNN's existence for several decades, it wasn't until the introduction of AlexNet that it gained widespread acceptance and became a mainstream approach. Subsequently, there has been a continuous evolution, resulting in the proposal of deeper and more efficient convolutional neural architectures that have played a crucial role in advancing the wave of deep learning in computer vision. Examples include VGG, GoogleNet, ResNet, DenseNet, HRNet, and EfficientNet. Alongside these architectural advancements, there has been considerable research focused on enhancing individual convolution layers, incorporating techniques such as depthwise convolution and deformable convolution.

Despite the continued dominance of CNNs and their variants as the primary backbone architectures for computer vision applications, it's important[4] to recognize the emerging potential of Transformer-like architectures for fostering unified modelling across both vision and language domains. The research and the implementation that we have performed, demonstrates robust performance on various fundamental visual recognition tasks, suggesting the possibility of a paradigm shift in modelling approaches.

Self-attention/Transformers to complement CNNs In response to the notable success of self-attention layers and Transformer architectures in natural language processing (NLP), some research endeavors have adopted self-attention layers to replace spatial convolution layers in widely-used architectures like ResNet. In these studies, self-attention computations occur within a local window around each pixel to expedite optimization. Notably, these approaches achieve slightly improved accuracy/FLOPs trade-offs compared to the conventional ResNet architecture. However, despite these advantages, their practical implementation faces challenges due to elevated memory access costs, resulting in significantly larger actual latencies compared to convolutional networks.

In contrast to the use of sliding windows in these approaches, the paper introduces a novel strategy of shifting windows between consecutive layers. This innovative approach enables a more efficient implementation on general hardware, addressing concerns related to computational costs and latency associated with self-attention in the context of computer vision tasks.

Transformer based vision backbones Most closely related to this models are Transformer-based vision backbones, particularly the Vision Transformer (ViT) and its subsequent iterations. The pioneering ViT work directly employs a Transformer architecture on non-overlapping medium-sized image patches for image classification, showcasing an impressive speed-accuracy trade-off compared to traditional convolutional networks. Notably, ViT's performance benefits from large-scale training datasets, such as JFT-300M. However, DeiT introduces various training strategies that enable ViT to also excel with smaller datasets like ImageNet-1K.

While ViT demonstrates encouraging results in image classification, its architecture proves less suited for serving as a general-purpose backbone network in dense vision tasks or scenarios involving input images with a high level of detail.

III. METHOD

A. Overall Architecture

Figure 3 provides an overview of the Swin Transformer architecture, featuring the compact version (SwinT). The process begins by dividing an input RGB image into non-overlapping patches using a patch splitting module, akin to the approach in ViT. Each patch is treated as a "token," and its feature is formed by concatenating the raw pixel RGB values. In our implementation, we employ a patch size of 4 × 4, resulting in a feature dimension of 4 × 4 × 3 = 48. A linear embedding layer is then applied to this raw-valued feature, projecting it to an arbitrary dimension denoted as C[6].

Several Transformer blocks, incorporating modified self-attention computations (Swin Transformer blocks), operate on these patch tokens. These Transformer blocks, along with the linear embedding, collectively form "Stage 1" and maintain the number of tokens (H/4 × W/4)[7]. As the network progresses deeper to create a hierarchical representation, the number of tokens is reduced through patch merging layers. The initial patch merging layer combines the features of each group of 2 × 2 neighboring patches, applying a linear layer to the 4C-dimensional concatenated features. This results in a reduction of tokens by a factor of 2×2 = 4 (equivalent to a 2× downsampling of resolution), with the output dimension set to 2C. Subsequent Swin Transformer blocks are then applied for feature transformation, while the resolution is maintained at H/8 × W/8.

The initial block involving patch merging and feature transformation is labeled as "Stage 2." This process is iterated twice, forming "Stage 3" and "Stage 4," resulting in output resolutions of H/16 × W/16 and H/32 × W/32, respectively. Collectively, these stages generate a hierarchical representation with feature map resolutions equivalent to those found in conventional convolutional networks like VGG and ResNet. Consequently, the proposed architecture is well-suited for seamlessly substituting the backbone networks in existing methods across a spectrum of vision tasks.

Swin Transformer Block The construction of the Swin Transformer involves substituting the standard multi-head self-attention (MSA) module within a Transformer block with a module based on shifted windows, as detailed in Section 3.2, while keeping the other layers unchanged. As depicted in Figure 3(b), a Swin Transformer block comprises a shifted window-based MSA module[5], succeeded by a 2-layer MLP with GELU nonlinearity in between. A LayerNorm (LN) layer is employed before each MSA module and each MLP, and a residual connection is applied following each module.

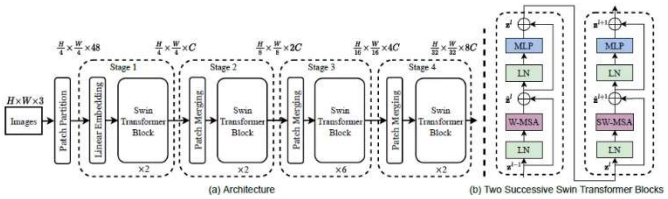


Figure 3. (a) The architecture of a Swin Transformer (Swin-T); (b) two successive Swin Transformer Blocks. W-MSA and SW-MSA are multi-head self-attention modules with regular and shifted windowing configurations, respectively[1].

B. Shifted Window Based Self-Attention

The conventional Transformer architecture and its modification for image classification both employ global self-attention, calculating relationships between a token and all other tokens. However, this global computation results in quadratic complexity concerning the number of tokens, rendering it impractical for numerous vision challenges that demand an extensive set of tokens for dense prediction or for representing a high-resolution image[8].

Self-Attention in Overlapped Windows To enhance efficiency in modeling, the research paper's proposed model involves computing self-attention within local windows. These windows are organized to evenly partition the image in a non-overlapping manner. Assuming each window comprises M × M patches, the computational complexity of a global multi-head self-attention (MSA) module and a window based MSA module on an image with h × w patches is given by:

$$\Omega(\mathrm{MSA}) = 4hwC^2 + 2(hw)^2C, \quad (1)$$

$$\Omega(\mathrm{W-MSA}) = 4hwC^2 + 2M^2hwC, \quad (2)$$

where the former is quadratic to patch number hw, and the latter is linear when M is fixed (set to 7 by default). Global self-attention computation is generally unaffordable for a large hw, while the window based self-attention is scalable[9].

Shifted Window Partitioning in Successive Blocks In successive blocks, the window-based self-attention module faces a limitation as it lacks connections across windows, thereby restricting its modeling power. To address this and introduce cross-window connections while preserving the efficiency of non-overlapping windows, the research paper describes a shifted window partitioning approach. This approach alternates between two partitioning configurations in consecutive Swin Transformer blocks.

As depicted in Figure 2, the initial module employs a standard window partitioning strategy that commences from the top-left pixel. The 8 × 8 feature map is evenly divided into 2 × 2 windows of size 4 × 4 (M = 4). Subsequently, the subsequent module adopts a windowing configuration that is shifted from the preceding layer, displacing the windows by ([M/2], [M/2]) pixels from the regularly partitioned windows. This shifted window partitioning approach ensures that consecutive Swin Transformer blocks are computed as:

$$\begin{aligned}\hat{\mathbf{z}}^{l} &= \text{W-MSA}\left(\text{LN}\left(\mathbf{z}^{l-1}\right)\right) + \mathbf{z}^{l-1},\\ \mathbf{z}^{l} &= \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l}\right)\right) + \hat{\mathbf{z}}^{l},\\ \hat{\mathbf{z}}^{l+1} &= \text{SW}-\text{MSA}\left(\text{LN}\left(\mathbf{z}^{l}\right)\right) + \mathbf{z}^{l},\\ \mathbf{z}^{l+1} &= \text{MLP}\left(\text{LN}\left(\hat{\mathbf{z}}^{l+1}\right)\right) + \hat{\mathbf{z}}^{l+1},\end{aligned} \tag{3}$$

W-MSA and SW-MSA denote window based multi-head self-attention using regular and shifted window partitioning configurations, respectively. The shifted window partitioning approach introduces connections between neighboring non-overlapping windows in the previous layer and is found to be effective in image classification.

Efficient batch computation for shifted configuration A challenge arising from shifted window partitioning is an increase in the number of windows, transitioning from [h/M] × [w/M] to ([h/M] + 1) × ([w/M] + 1) in the shifted configuration. Some of these windows may end up smaller than M × M. A straightforward solution involves padding the smaller windows to a size of M × M and masking out the padded values during attention computation. However, this naive solution significantly increases computation, especially when the number of windows in regular partitioning is small (e.g., 2 × 2 → 3 × 3, resulting in a 2.25 times increase)[10].

To address this, the research paper proposes a more efficient batch computation approach using cyclic shifting towards the top-left direction. Following this shift, a batched window may consist of several sub-windows that are not adjacent in the feature map[11]. A masking mechanism is employed to restrict self-attention computation within each sub-window. This cyclic-shift approach maintains the same number of batched windows as regular window partitioning, ensuring efficiency.

C. Architecture Variants

We have implemented all 4 variations of the Swin Transformer described in the paper with their hyperparameter variants as shown below:

- Swin-T: C = 96, layer depths = {2, 2, 6, 2} - 'swin_tiny_224'
- Swin-S: C = 96, layer depths ={2, 2, 18, 2} - 'swin_small_224'
- Swin-B: C = 128, layer depths ={2, 2, 18, 2} - 'swin_base_224'
- Swin-L: C = 192, layer depths ={2, 2, 18, 2} - 'swin_large_224'

where C is the channel number of the hidden layers in the first stage.

The model summaries are shown below:

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 224, 224, 3)       0

 swin_tiny_224 (SwinTransfo  (None, 768)               27769058
 rmerModel)

 dense (Dense)               (None, 104)               79976

=================================================================
Total params: 27849034 (107.19 MB)
Trainable params: 27599330 (105.28 MB)
Non-trainable params: 249704 (1.91 MB)
_________________________________________________________________
```

(a)

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 224, 224, 3)       0

 swin_small_224 (SwinTransf  (None, 768)               49173398
 ormerModel)

 dense (Dense)               (None, 104)               79976

=================================================================
Total params: 49253374 (189.17 MB)
Trainable params: 48917234 (186.60 MB)
Non-trainable params: 336140 (2.56 MB)
_________________________________________________________________
```

(b)

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 224, 224, 3)       0

 swin_base_224 (SwinTransfo  (None, 1024)              87079364
 rmerModel)

 dense (Dense)               (None, 104)               106600

=================================================================
Total params: 87185964 (333.87 MB)
Trainable params: 86849824 (331.31 MB)
Non-trainable params: 336140 (2.56 MB)
_________________________________________________________________
```

(c)

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lambda (Lambda)             (None, 224, 224, 3)       0

 swin_large_224 (SwinTransf  (None, 1536)              195331616
 ormerModel)

 dense (Dense)               (None, 104)               159848

=================================================================
Total params: 195491464 (747.02 MB)
Trainable params: 195155324 (744.46 MB)
Non-trainable params: 336140 (2.56 MB)
_________________________________________________________________
```

(d)

Figure 4. (a) Model summary of Swin-T (b) Model summary of Swin-S (c) Model summary of Swin-B (d) Model summary of Swin-L

IV. EXPERIMENTS

For our implementation, we have conducted experiments on Tiny ImageNet, CIFAR100, and a Kaggle dataset - Flower Classification. The Flower dataset has been completely trained using our Swin Transformer model while the Tiny-Imagenet and CIFAR100 dataset has only been ran for a few epochs due to the lack of computational resources and exceedingly high training time. We have also loaded pre-trained weights of the Swin Transformer model to train on these datasets. Only the classification tasks from the original paper have been implemented.

A. Image Classification on Flower Dataset - Kaggle

This dataset is taken from a Kaggle competition - 'Flower Classification with TPU's'. The dataset contains 104 types of flowers based on their images drawn from five different public datasets. Some classes have a small number of samples while others contain many sub-types. The dataset contains certain imperfections like flowers in different orientations and some are also in the background rather than the foreground. The dataset contains 12753 training images and 3712 validation images. The size of the images is 224x224x3. The Swin-Transformer is trained using pre-trained weights and all the Swin configurations.

B. Image Classification on Tiny – ImageNet and CIFAR100

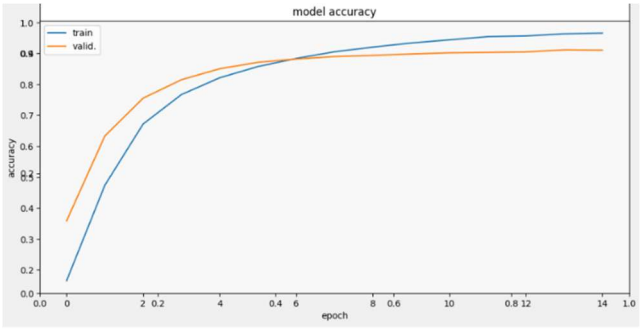
The Tiny-imagenet dataset is a subset of the Imagenet dataset with 200 classes and each class has 500 images. The image sizes are 64x64 and they have been upscaled to 224x224 and fed into our model.

The CIFAR100 is a benchmark dataset in the field of computer vision and consists of 100 classes with 600 images in each class. The image sizes are 32x32. They have been upscaled to 224x224 and fed into our model.

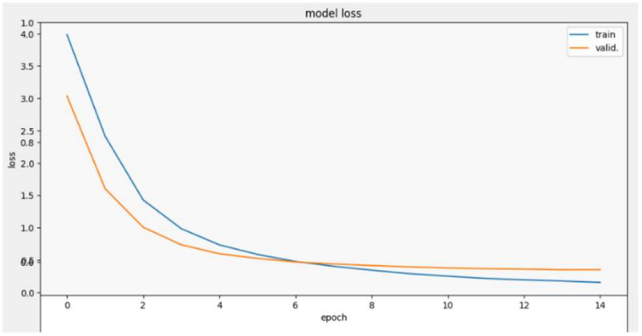
V. RESULTS

A. Results of Flower Dataset – Kaggle

The results we obtained for the Flower Dataset showed the most promising results. For this dataset, we employed an Adam Optimizer for 15 epochs to obtain the best outcome. The initial learning rate was chosen as 1e-5, the default weight decay was set to 0, $\beta 1 = 0.9$ and $\beta 2 = 0.999$. The figures below represent the model accuracies and model loss for each of the variations of Swin Transformer that we have implemented. The confusion matrices are also plotted. Table 1 shows accuracies across different architectures and Table 2 shows training time and training loss across different architectures.

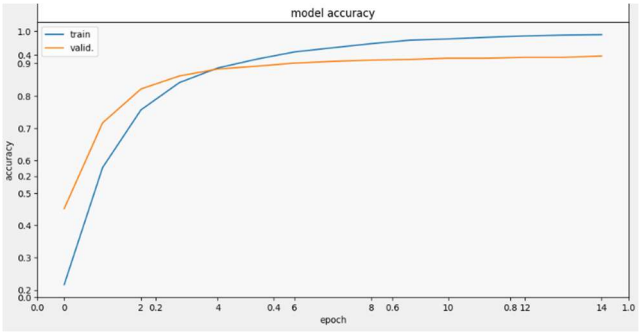


(a)

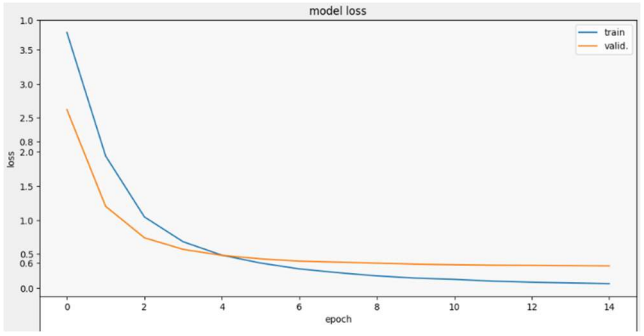


(b)

Figure 5. (a) Model Accuracy of Swin -T (b) Model Loss of Swin - T

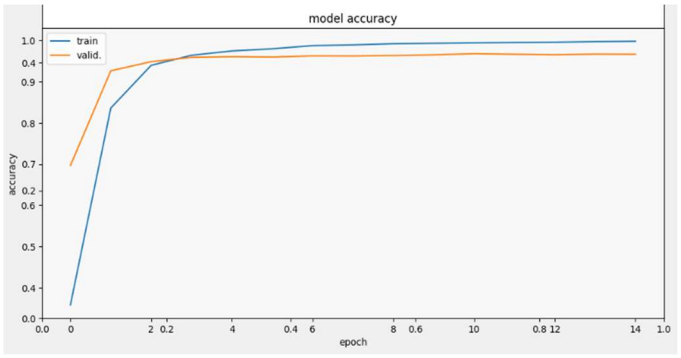


(a)

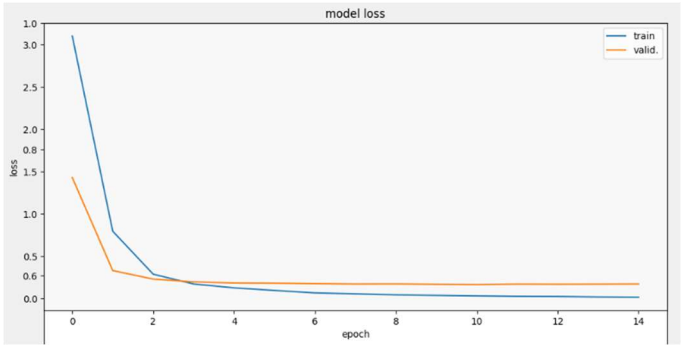


(b)

Figure 6. (a) Model Accuracy of Swin -S (b) Model Loss of Swin - S

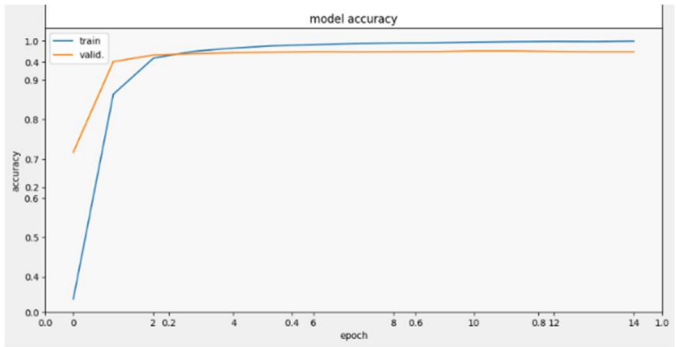


(a)

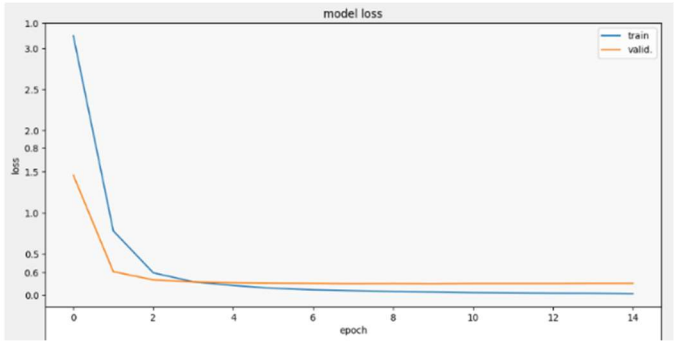


(b)

Figure 7. (a) Model Accuracy of Swin-B (b) Model Loss of Swin - B



(a)



(b)

Figure 8. (a) Model Accuracy of Swin -L (b) Model Loss of Swin - L

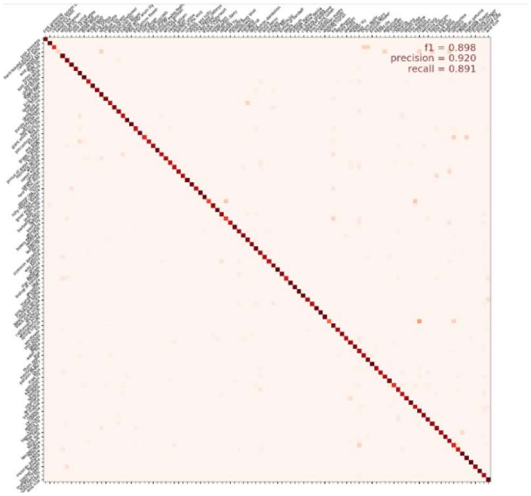


Figure 9. Confusion Matrix of Swin T

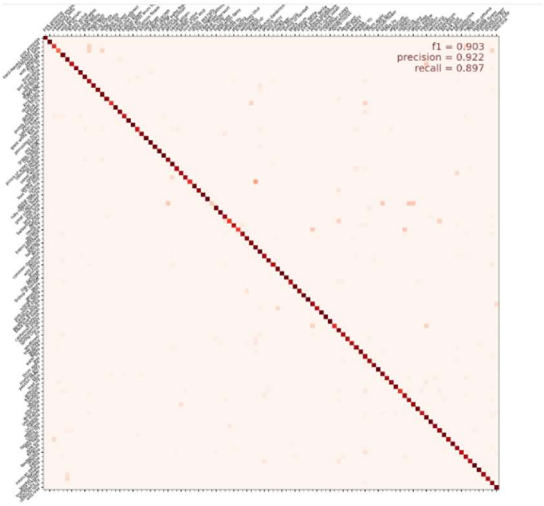


Figure 10. Confusion Matrix of Swin S

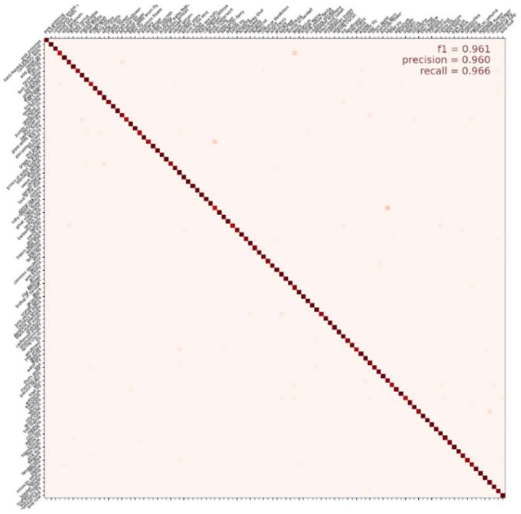


Figure 11. Confusion Matrix of Swin B

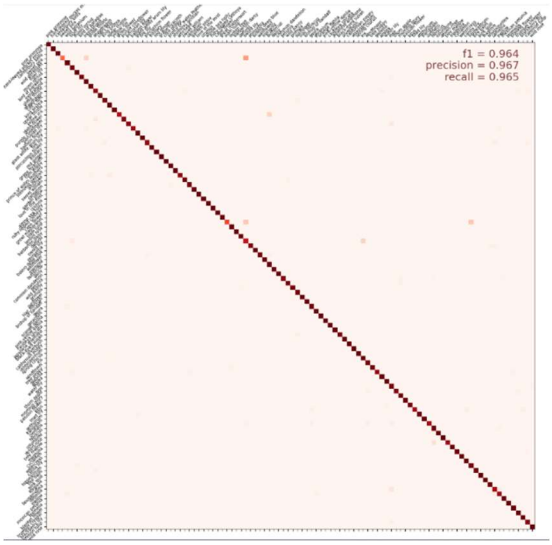


Figure 12. Confusion Matrix of Swin L

TABLE I. ACCURACY ON FLOWER DATASET ACROSS DIFFERENT ARCHITECTURES

Model Architecture	**Training Accuracy**	**Validation accuracy**
Swin-T	0.9653	0.9103
Swin-S	0.9893	0.9232
Swin-B	0.9980	0.9666
Swin-L	0.9987	0.9717

TABLE II. TRAINING LOSS AND TRAINING TIME ACROSS DIFFERENT ARCHITECTURES

Model Architecture	**Training Loss**	**Training Time**
Swin-T	0.1538	Trained for: 15 epochs Time: ~400s on Kaggle TPU VM v3-8
Swin-S	0.0660	Trained for: 15 epochs Time: ~550s on Kaggle TPU VM v3-8
Swin-B	0.0160	Trained for: 15 epochs Time: ~600s on Kaggle TPU VM v3-8
Swin-L	0.0139	Trained for: 15 epochs Time: ~800s on Kaggle TPU VM v3-8

B. Results of Tiny ImageNet and CIFAR100

For both of these datasets, we used pre-trained weights to train our model. The accuracies reported are poor and incomplete due to time and resource constraints. The poor performance could be attributed to the upscaling of images to 224x224. The poor training time is because of a lack of available resources for computation. On average for the Tiny-ImageNet dataset, an epoch took approximately 4500 seconds. The Tiny-ImageNet dataset has been run for only 3 epochs as we constantly experienced the kernel crashing.

TABLE III. ACCURACIES OF TINY IAMAGNET AND CIFAR100 DATASETS

Dataset	**Training Accuracy**	**Time Taken**
Tiny-imagenet - upscaled to 224 x 224	0.02	4500 seconds per epoch
CIFAR100 - upscaled to 224 x 224	0.5203	~18 hours

VI. CONCLUSION

We have implemented the entire Swin Transformer model with all the different architectures. It has been trained on the 'Flower Classification with TPU's' dataset from Kaggle with good results.

However, we have not been able to reproduce the results on the datasets used in the paper due to a restriction on the resources available. We were unsuccessful in trying to obtain V100 or A100 machines.

In our project implementation, we closely followed a repository[12] available online, adopting a similar structure to facilitate the seamless integration of pre-trained weights without encountering errors. This decision was driven by the necessity to ensure compatibility and consistency, especially when utilizing pre-existing weight files. Importantly, our objective was not to replicate the code verbatim but rather to understand its architecture and design principles. By aligning

our implementation with the established structure, we aimed to create a framework that could readily accommodate the pre-trained weights. This approach, rooted in understanding and adapting rather than copying, was undertaken to enhance efficiency and build upon existing knowledge, emphasizing a legitimate and constructive use of available resources rather than engaging in plagiarism.

Looking ahead, we would like to test our model implementation on benchmark datasets like ImageNet and also other tasks such as object detection.

VII. Individual Contribution

TABLE IV. Contribution of Teammates

Name	Contribution	Fraction
Aarya Raghavan (ar4634)	Building the model architecture, testing datasets, compiling results, and formulating the report.	1/3
Abhik Biswas (ab5640)	Building the model architecture, testing datasets, compiling results, and formulating the report.	1/3
Abhilash Praveen Kumar (ap4478)	Building the model architecture, testing datasets, compiling results, and formulating the report.	1/3

References

[1] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012-10022).

[2] Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Jianfeng Gao, Songhao Piao, Ming Zhou, et al. Unilmv2: Pseudo-masked language models for unified language model pre-training. In International Conference on Machine Learning, pages 642–652. PMLR, 2020.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and HongYuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020..

[4] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 702–703, 2020.

[5] Cheng Chi, Fangyun Wei, and Han Hu. Relationnet++: Bridging visual representations for object detection via transformer decoder. In NeurIPS, 2020.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization.

[7] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. Biological cybernetics, 20(3):121– 136, 1975.

[8] Yann LeCun, Patrick Haffner, Leon Bottou, and Yoshua Ben- ´ gio. Object recognition with gradient-based learning. In Shape, contour and grouping in computer vision, pages 319– 345. Springer, 1999.

[9] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In Proceedings of the European conference on computer vision (ECCV), 2020.

[10] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9759–9768, 2020.

[11] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In BMVC, 2016.

[12] Rishigami. (n.d.). GitHub - rishigami/Swin-Transformer-TF: Tensorflow implementation of Swin Transformer model. GitHub. https://github.com/rishigami/Swin-Transformer-TF