Swin Transformer: Hierarchical Vision Transformer using Shifted Windows

E4040.2023Fall.ZKY3.report
Zirui Zhang, Xinyi Ke, Yuning Han
Columbia University

Abstract

In our group's attempt to replicate, this paper introduces a new type of vision transformer, called Swin Transformer, which can function as a universal support structure in computer vision. This paper addresses the challenge of adapting Transformers from language to vision domains. It features a hierarchical structure with shifted windows for efficient processing, enabling it to handle various scales and complexities in image analysis. The Swin Transformer performs highly in tasks like image classification, object detection, and semantic segmentation, with notable accuracy on standard datasets like ImageNet-1K and COCO. The model was replicated using TensorFlow, closely matching the original paper's results.

1. Introduction

In the field of computer vision, the construction of models has long been dominated by Convolutional Neural Networks (CNNs). This trend was cemented with the revolutionary performance of AlexNet in the ImageNet image classification challenge. Since then, CNN architectures have evolved to become increasingly powerful, thanks to larger scales, more extensive connections, and more sophisticated convolution methods. As CNNs became the backbone for various vision tasks, these architectural advancements led to significant performance improvements, propelling the entire field forward.

On the other hand, the evolution of network architectures in the natural language processing (NLP) field has diverged, with the Transformer becoming the prevalent architecture. Designed for sequence modeling and transduction tasks, the Transformer is known for its use of attention mechanisms to model long-range dependencies in data. Its tremendous success in NLP has led researchers to explore its application in computer vision, showing promising results in specific tasks, especially in image classification and joint vision-language modeling. In this paper, we aim to extend the applicability of Transformers to serve as a universal backbone in computer vision, similar to their role in NLP and the way CNNs function in vision. We note that transferring its high performance from language to vision faces significant challenges due to differences between the two modalities. One such difference involves scale. Unlike word tokens in language transformers, visual elements can vary greatly in scale, which is a challenge in tasks like object detection. Existing Transformer-based models treat tokens at a fixed scale, which is not suitable for these visual applications. Another difference is the much higher resolution of pixels in images compared to words in text. Many vision tasks, such as semantic segmentation, require dense prediction at the pixel level, which is impractical for Transformers on high-resolution images due to the quadratic computational complexity of their self-attention with image size.

To deepen our understanding and exploration in the field of image processing, our team decided to replicate the Swin Transformer model as designed in the referenced literature. This model constructs hierarchical feature maps, and its computational complexity is linear with respect to image size. The primary challenge and focus during replication was implementing the model's two-tier structure: the first level is the hierarchical window-based self-attention mechanism. In standard Transformer architectures and their adaptations for image classification, global self-attention is executed, computing relationships between one token and all other tokens. However, global computation leads to quadratic complexity relative to the number of tokens, making it unsuitable for vision problems requiring dense predictions with a large number of tokens or representing high-resolution images. For efficient modeling, the original authors proposed computing self-attention within local windows, arranged to partition the image in a non-overlapping manner. The second level is the shifted window partitioning in consecutive blocks. In previous methods, the window-based self-attention module lacked connections across windows, limiting its modeling capabilities. To introduce cross-window connections while maintaining efficient computation in non-overlapping windows, the authors proposed a shifted window partitioning method, alternating between two partitioning configurations in consecutive Swin Transformer blocks. Detailed explanations of the principles are provided later in the text. Ultimately, we successfully replicated the model according to the implementation and optimization ideas in the paper and achieved results close to the original on the image classification task based on the ImageNet-1K dataset, which is quite satisfactory.

In our group's attempt to replicate, this paper introduces a new type of vision transformer, called Swin Transformer, which can function as a universal support structure in computer vision. This paper addresses the challenge of adapting Transformers from language to vision domains. It features a hierarchical structure with shifted windows for efficient processing, enabling it to handle various scales and complexities in image analysis. The Swin Transformer performs highly in tasks like image classification, object detection, and semantic segmentation, with notable

accuracy on standard datasets like ImageNet-1K and COCO. The model was replicated using TensorFlow, closely matching the original paper's results.

2. Summary of the Original Paper

2.1 Background and Objective

This paper addresses the challenge of adapting the Transformer architecture, originally designed for natural language processing, to computer vision tasks. The objective is to introduce a scalable and efficient Transformer model, named Swin Transformer, capable of handling high-resolution images for various vision-based applications.

2.2 Innovative Approach

The Swin Transformer introduces a novel hierarchical architecture with shifted windows, optimizing the self-attention mechanism for vision tasks. This approach contrasts with traditional Transformers that apply self-attention globally and face scalability issues with large images.

Important methodologies

Hierarchical Design: The model processes images in a multi-scale manner, allowing for efficient feature extraction at different levels.

Shifted Window Mechanism: By shifting the window partition between consecutive self-attention layers, the model integrates local and global context, enhancing feature representation.

Benchmarks and Evaluation

Extensive experiments are conducted on various datasets and tasks, including ImageNet for classification, COCO for object detection, and ADE20K for semantic segmentation, demonstrating the model's versatility and robustness.

Key Findings

Computational Efficiency: The Swin Transformer is shown to be more computationally efficient than existing Vision Transformer models, especially in handling larger images.

Performance Excellence: It achieves state-of-the-art results on multiple benchmarks, setting new records in different vision tasks.

Adaptability: The model's design allows it to adapt effectively across various scales and tasks, a crucial feature for practical applications.

2.3 Critical Evaluation

Contributions and Impact: The paper significantly advances the field of computer vision by adapting Transformer models to image analysis more effectively than previous methods. Its innovative approach to scaling and computational efficiency has potential implications for future Transformer-based models in vision tasks.

Limitations and Future Directions: While the model shows impressive performance, it still demands substantial computational resources, which might limit its deployment in resource-constrained environments. Future research could focus on further optimizing the model's efficiency and exploring its applications in other vision tasks beyond those tested.

Broader Implications: The Swin Transformer's success suggests a promising direction for integrating Transformer architectures in various domains beyond natural language processing, potentially leading to breakthroughs in other areas of artificial intelligence.

3. Methodology (of the Students' Project)

3.1. Objectives and Technical Challenges

To reimplement this paper, here are several challenges we will meet:

- Swin Transformer has specific requirements for the size and format of input data. Improper data preprocessing can lead to a decline in model performance.
- The structural complexity of Swin Transformer, with multiple layers and intricate components, makes accurately replicating the model a challenge, especially for those without an extensive background in deep learning. This is also the most important part in this project.
- Swin Transformer requires substantial computational resources for effective training, which may exceed the capabilities of typical research or development environments. We met a great deal of difficulties when training the model because the GCP usually shut down, which was very unstable.
- Properly configuring the development environment, including library versions, dependencies, and system settings, is crucial for replicating complex models, but this process is often challenging.

3.2.Problem Formulation and Design Description

In our project, we can break this problem into 6 parts: Patch embedding, Self attention and multi-head attention; Window shift, Path merging, relative position, drop path and warm-up learning schedule. Let's talk about them one by one.

3.2.1 Overall structure of the Swin-Transformer model

The Swin Transformer's design allows it to process images more efficiently than standard Transformers because it reduces the resolution as it moves through the stages, decreasing the computational load. Additionally, by shifting the windows, it ensures that the model captures both local and global context, which is essential for understanding the entire image. This design makes the Swin Transformer particularly well-suited for tasks that require understanding the image at multiple scales, such as object detection and semantic segmentation. An overall

structure can be viewed as below, and for structure (a), we have:

Image Input and Patch Partitioning: The first image seems to show the overall pipeline starting from the input image. The model begins by dividing the input image (denoted as HxWx3, for height, width, and 3 color channels) into patches. These patches are then linearly embedded into tokens with a certain dimension (denoted by C).

Hierarchical Representation: The architecture of the Swin Transformer is hierarchical, consisting of several stages. Each stage corresponds to a different resolution and feature dimension: Stage 1 processes the embedded tokens with a lower resolution; Stage 2 doubles the dimension of the features while reducing the resolution further; Stages 3 and 4 continue this pattern, progressively increasing the feature dimension and decreasing the resolution, allowing the network to capture more abstract representations at higher levels.

Swin Transformer Blocks: The second image illustrates the internal structure of a Swin Transformer block, which is the basic building block of the model. Each block consists of two layers: The first layer includes a Layer Normalization (LN) followed by a Multihead Self-Attention mechanism (either W-MSA, which stands for Window-based Multihead Self-Attention, or SW-MSA, which stands for Shifted Window Multihead Self-Attention). The attention mechanism allows the model to focus on different parts of the image. The shifting window mechanism is crucial for connecting the representations of adjacent non-overlapping windows.

Residual Connections: Each layer (both self-attention and MLP) has a residual connection, meaning the output of the layer is added to its input. This is denoted by the plus signs in the diagram. Residual connections help with the flow of gradients during training, which can improve the learning of deep networks.

Layer Normalization (LN): Before each block (W-MSA/SW-MSA and MLP), there is a layer normalization step. This helps stabilize the learning process and accelerates the training of deep networks.

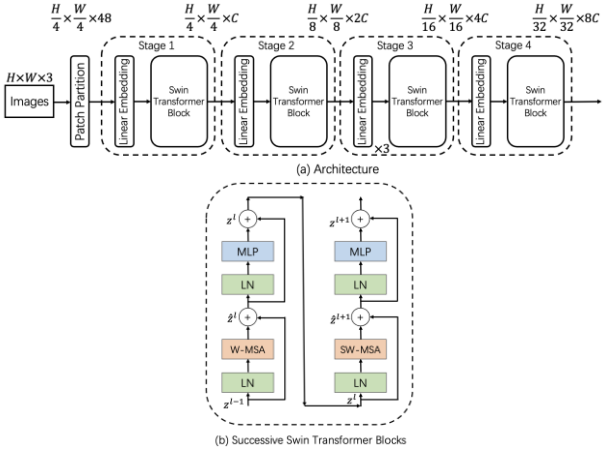

Image1 Overall Structure of Swin-transformer

3.2.2 Patch embedding

Prior to the input being fed into the Block, it is imperative to segment the image into distinct patches, subsequently embedding them into vectors.

This process entails cropping the original image into numerous windows, each measuring patch_size by patch_size, followed by their respective embeddings.

To achieve this, one can employ a two-dimensional convolutional layer, setting both the stride and kernel size to the dimensions of patch_size. The size of the embedding vector is determined by specifying the output channels. Finally, the dimensions H and W are unfolded and then repositioned to the foremost dimension.

3.2.3 Patch Merging

The primary function of this module is to perform downsampling at the commencement of each Stage, aiming to reduce resolution and adjust the number of channels. This facilitates a hierarchical design, simultaneously contributing to a reduction in computational load.

In the context of CNNs, downsampling at the start of each Stage is typically achieved using convolutional or pooling layers with a stride of 2, to decrease resolution. This downsampling is conducted at a rate of two-fold, hence elements are selected every two intervals in both row and column directions.

Subsequently, these elements are concatenated to form a single tensor, which is then unfolded. At this juncture, the channel dimension increases to four times its original size (since both H and W dimensions are reduced by half). Finally, this expanded channel dimension is readjusted to twice its initial size through the application of a fully connected layer. Details can be viewed in the image 2.

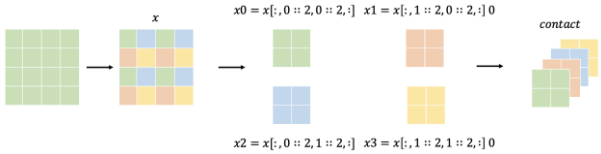

Image2 Patch merging description

3.2.4 Self attention and multi-head attention

Window attention: This component is pivotal in the discussed paper. Traditional Transformers compute attention on a global scale, resulting in high computational complexity. In contrast, Swin Transformer confines the computation of attention within individual windows, thereby reducing the computational load.

Let's briefly examine the formula:

$$Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d}} + B\right)V$$

The primary distinction lies in the incorporation of relative position encoding into the original formula for

computing Attention with Q (queries) and K (keys). Subsequent experiments have demonstrated that the inclusion of relative position encoding enhances model performance.

Now let's see the codes about window-attention:

```
def call(self, x, training=True):
  """
  Args:
  x: input features with shape of [B, H, W, C]
  training: training mode or inference mode
      """
  # [B, H, W, C] -> [num_windows * batch_size, Mh *
Mw, Channels]
  B, H, W, C = x.shape
```

For the dimensions entered in the code, we can see that the first dimension was originally supposed to be equal to the batch size, which here becomes the batch size multiplied by the number of windows, because here there is no information interaction between the different windows, therefore, to speed up the training, we merge these two dimensions

Shift Window Attention: In addition to computing attention within each individual window, the Swin Transformer introduces the concept of "shifted window" operation to facilitate better information exchange between different windows.

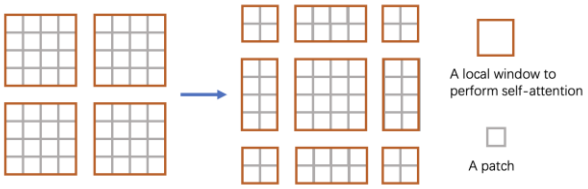

Image3 Shift window attention figure

In image3, the figure on the left illustrates Window Attention without overlap, while the figure on the right shows Shifted Window Attention, where the windows are shifted to include elements from adjacent original windows. This shifting results in an increase in the number of windows, transforming the original four windows into nine.

However, this shift introduces a new challenge: the doubling of the number of windows. In practical implementation, this issue is addressed by shifting the feature map and indirectly applying a mask to the Attention mechanism. This approach maintains the original number of windows while ensuring that the final computational results are equivalent.

In the actual code, this is achieved by manipulating the feature map and setting masks for the Attention operation, thus maintaining the initial count of windows and achieving the desired outcome effectively.

Attention mask: The essence of the Swin Transformer lies in its ability to achieve equivalent computational results with Shifted Window Attention, using the same number of windows as Window Attention, all through the strategic setting of masks. To illustrate this, consider the scenario where each window, after undergoing a shift (for example, with window_size=2, shift_size=-1), is assigned an index, as the image5 below.

The goal during the computation of Attention is to ensure that queries (Q) and keys (K) with the same index are computed together, while disregarding the computation results of QK pairs with different indices.

The desired outcome is depicted in the following illustration:

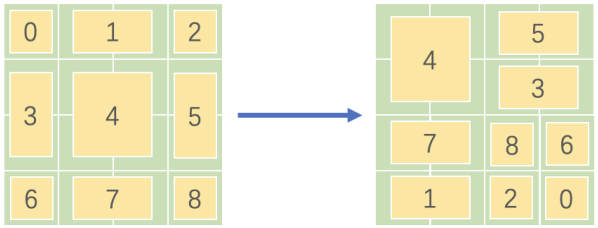

Image4

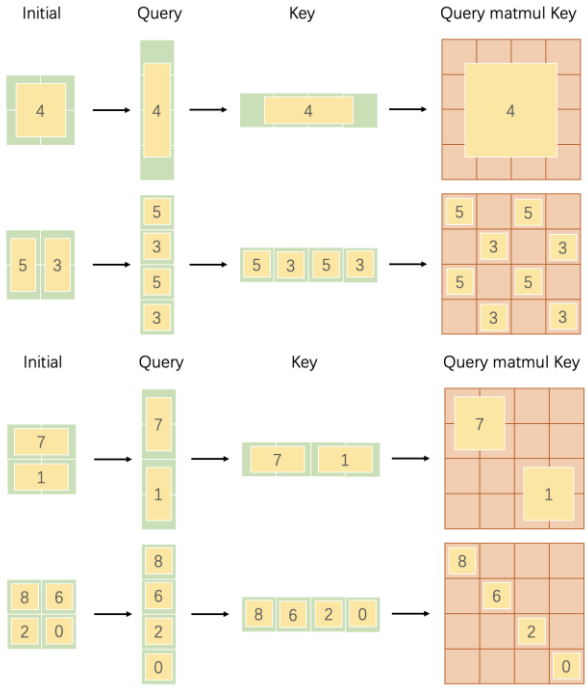

Image5

To achieve accurate results under the original four-window configuration, it becomes necessary to introduce a mask to the Attention results (as shown on the far right of the illustration).

This mask ensures that only the appropriate QK pairs contribute to the Attention computation, effectively

simulating the effect of having more windows for information exchange while maintaining the original window count. This innovative approach allows the Swin Transformer to efficiently handle the increased complexity brought about by the shifted windows, without a proportional increase in computational demand.

3.2.4 Relative position coding

The calculation process of relative position encoding can be seen in the figure below:

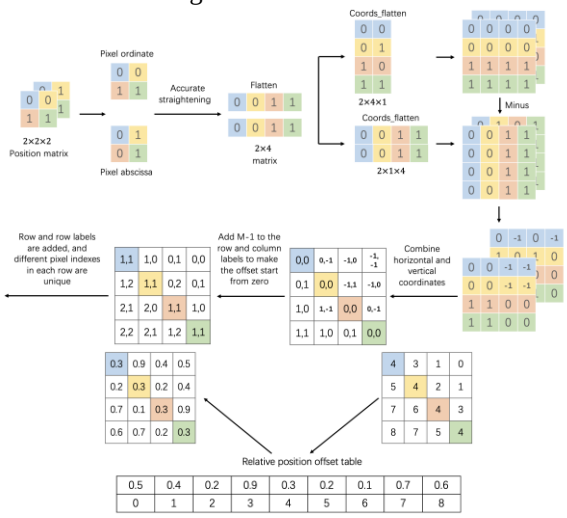

Image6 Relative position coding figure

Unlike standard Transformers, which process data in sequences, Swin Transformers operate on images and use a hierarchical structure that allows for efficient processing of high-resolution images.

The concept of position encoding is critical in Transformer models because they are based on attention mechanisms that do not inherently consider the order or position of data points. In natural language processing (NLP), position encoding is used to give the model information about the order of words in a sentence. Similarly, in computer vision, position encoding gives the model information about the location of pixels or patches in an image.

We can deduce the following steps in computing the relative position encoding:

Position Matrix: The initial matrix represents the pixel positions in a 2x2 grid, indicating the relative positions of pixels in a small patch.

Flattening and Straightening: The pixel coordinates are flattened into a vector to represent their positions linearly. This process involves converting the 2D structure of image data into a 1D structure that can be processed by the Transformer.

Relative Position Offsets: A relative position offset table is created, which represents the relative distances between different pixel positions. This is crucial for the model to understand how pixels are positioned concerning each other.

4. Coordination Flatten: The coordinates are further processed to create flattened versions that can be manipulated to obtain relative positions.

5. Combining Horizontal and Vertical Coordinates: The final step involves combining the horizontal and vertical coordinates to form a matrix that encodes the relative position information. This step is critical for allowing the model to understand the 2D structure and relationships between pixels in the image.

The Swin Transformer uses these relative position encodings to allow self-attention mechanisms to consider the relative positions of patches within the image. This way, the model can make more informed predictions based on not only the features within a patch but also how those patches relate to one another spatially within the image.

The numerical values in the matrices and the color-coding likely represent different aspects of the position encoding process, such as the indices of pixels or the relative strength of position signals. This kind of encoding enables the model to learn and retain spatial hierarchies and relationships within the image, which are vital for tasks such as object detection, image segmentation, and image classification.

The calculation process of relative position encoding can be seen in the Figure6.

3.2.5 Cosine learning rate warmup schedule

The Cosine Learning Rate Warmup schedule is a learning rate scheduling strategy combining learning rate warmup with cosine decay. At the initial stage of training, a relatively small learning rate is used to avoid large weight updates at the beginning of training, which can lead to training instability. As training progresses, the learning rate is gradually increased, allowing the network to accept larger weight updates. When the learning rate reaches a baseline, it start cosine decay using the shape of the cosine function:

$$\mathrm{lr(t)} = lr_{min} + \frac{1}{2}(lr_{in} - lr_{min})(1 + \cos(\frac{t - t_{base}}{t_{total}}\pi))$$

Here, lr_{min} is the learning rate at the epoch t, lr_{in} and lr_{min} is the initial and final learning rate, t_{base} is the end epoch of the warmup phase, t_{total} is the total number of the training epochs.

3.3.Algorithm and software design

Algorithm 1 Swin Transformer Pseudocode

```
Input: Image I
Output: Feature map for downstream tasks
procedure SwinTransformer(I)
    patches ← DivideIntoPatches(I)
    tokens ← LinearEmbedding(patches)
    for each stage in hierarchical stages do
        tokens ← Downsample(tokens)    ▷ Downsampling between stages
        for each block in stage do
            windows ← DivideIntoWindows(tokens)
            if block is even then
                windows ← ShiftWindows(windows)
            end if
            for each window in windows do
                tokens ← WindowBasedSelfAttention(window)
            end for
        end for
    end for
    return GenerateOutput(tokens)
end procedure
```

Algorithm1

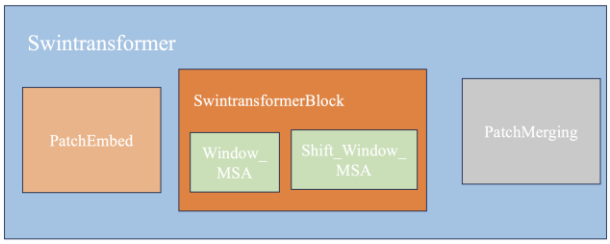

Image7 Code structures

The largest surrounding box is labeled "Swin Transformer", indicating the overall architecture. Within this box, there are three main components, each within their own box and connected sequentially from left to right:

- The leftmost box is labeled "Patch Embed", suggesting the initial stage of the architecture where input images are divided into patches and embedded into a higher-dimensional space.
- The middle box contains two components and is labeled "Swin Transformer Block", which represents a core unit within the architecture:
- The top sub-component of this block is labeled "Window MSA" (Multihead Self-Attention), which likely refers to the use of self-attention mechanisms within local windows of the embedded patches.
- The bottom sub-component is labeled "Shift Window MSA", which implies a variant of the self-attention mechanism that shifts the window positions to capture relations between different parts of the image.
- The rightmost box is labeled "Patch Merging", which likely describes the process of merging the patches or reducing the resolution to form a hierarchical representation.

This diagram is a high-level representation of the Swin Transformer architecture, showcasing the flow of data from input patch embedding, through transformer blocks with self-attention mechanisms, to patch merging for further processing or output.

(codes URL: **https://github.com/ecbme4040/e4040-2023fall-project-zkh3-zz3093-xk2161-yh3613.git)**

4. Experiments

4.1 Data

We choose Tiny Imagenet((Stanford CS231N) as our training dataset. The dataset consists of 200 classes, each with 500 training images.

Training samples	Validation Samples	Shape	Number of classes
90000	10000	(64,64,3)	200

Table 4.1: Tiny Imagenet Dataset Statistics

The preprocessing pipeline involved reading and decoding JPEG images, resizing them to a standardized size of 224x224 pixels, normalization of pixel values to the range [0, 1], one-hot encoding of class labels, and shuffling of the dataset. We incorporated a fixed random seed, specifically seed 42, throughout the experiments, allowing for consistent results across multiple runs.

4.2 Experiment Results

Model	Val loss	Val accuracy
Swin-transformer	0. 0605	0. 9964
w/o relative position	1. 6809	0. 5779
w/o shifted window	0. 7796	0. 9073
w/o drop path	0. 4260	0. 9188

Table 4.2: Baseline Swin Transformer Model with Shifted Window, Relative Position, and Drop Path Rate=0.1. "w/o" Represents "Without" and implementations on a T4 GPU.

The removal of the relative position, shifted window, and Drop Path mechanism led to a substantial decline in performance, with the absence of the shifted window having the most pronounced impact. These observations underscore the crucial role of specific model

components in optimizing the Swin Transformer for the task.

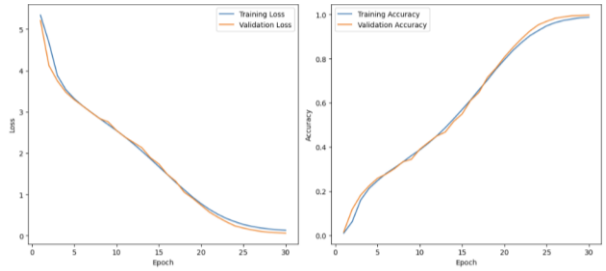

Image a: Baseline Swin Transformer

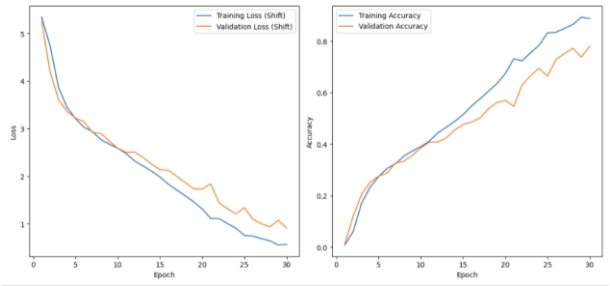

Image b: Swin Transformer without shift window

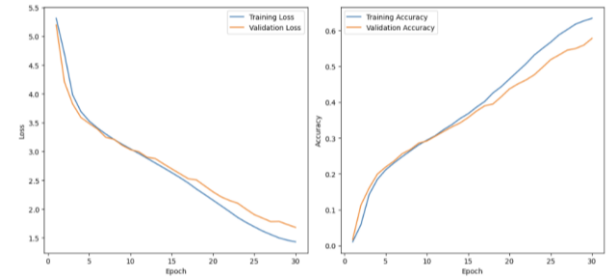

Image c: Swin Transformer without relative position

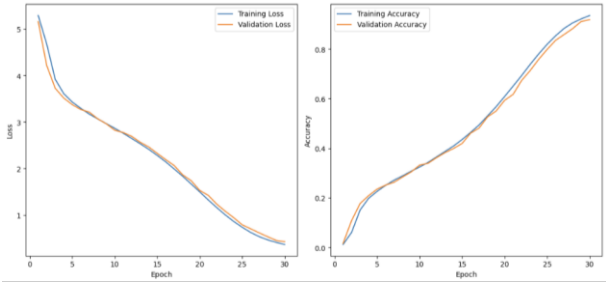

Image d: Swin Transformer without drop path

Image8(a, b, c, d): Experiments results

The presented training and validation loss and accuracy figures provide insights into convergence speed and generalization capabilities. Notably, the model with drop path exhibits limited sensitivity to this regularization technique. However, in the absence of shift window and drop path, the training accuracy increases more rapidly, and the training loss decreases more swiftly than the validation set. These observations are indicative of overfitting, wherein the model tends to capture noise and intricacies specific to the training data. Employing relative position and shift window mechanisms appears to be advantageous in mitigating overfitting and enhancing generalization performance.

4.3 Training and Inference time

The total training times for the three experiments are in close proximity, with an average of approximately **76020.32 seconds**. Similarly, the validation time per image exhibits consistency across the experiments, averaging around **0.011 seconds** per image. These findings suggest a comparable computational efficiency in terms of both overall training duration and the time required for each validation image across the conducted experiments.

4.4 Comparison of the Results Between the Original Paper and Students' Project

	Paper's	Replication
Baseline Swin	0.813	0.987
w/o shift window	0.802	0.634
w/o relative pos	0.801	0.887

Table 4.4 Comparison between the Paper's results and the replication with accuracy.

Due to spatiotemporal constraints, our experiments were conducted on the Tiny Imagenet dataset rather than the ImageNet-1K dataset used in the original paper, making a direct comparison somewhat impractical. Nevertheless, our replication demonstrates commendable results, showcasing a favorable alignment with the trends observed in the paper. Specifically, both our replication and the paper indicate superior performance for the baseline Swin Transformer compared to models without window shifting and relative positional encoding.

Moreover, our findings underscore the heightened sensitivity of the replication model to window shifting, evidenced by a notable drop in accuracy to 0.634. In contrast, the paper's model exhibits substantial stability, with minimal fluctuations in accuracy across its variants. While acknowledging the dataset disparity, our replication not only captures the essence of the paper's outcomes but also highlights how the replicated model is sensitive to window shifting.

5 Discussion / Insights Gained

To furhter explore the disadvantages, and capabilities of the Swin Transformer, we reviewed recent papers from the past few years.
Potential shortcomings of the original Swin Transformer paper include:

- **Computational Efficiency:** Although designed for efficiency, the Swin Transformer may still require substantial computational resources, especially for large datasets and complex tasks.
- **Scale Invariance:** The Swin Transformer might struggle with features of varying scales, which is critical in computer vision.
- **Generalization Ability:** While performing well on specific benchmarks, the Swin Transformer may have limited generalization capability on new or unseen data.

Several follow-up studies have addressed the limitations of the original Swin Transformer and proposed enhancements:

- **Swin Transformer V2:** This version addresses scaling issues in the original Swin Transformer. It introduces techniques like residual post-normalization and scaled cosine attention to stabilize training when scaling up model capacity. A log-spaced continuous position bias approach is used to improve model performance when transferring across different window resolutions.
- **Hybrid CNN-Transformers:** A study proposed a local perception Swin transformer (LPSW) backbone, integrating it with CNNs to improve standard transformers' performance, especially for detecting small-sized objects.
- **Swin-APT for Intelligent Transportation:** Swin-APT is a deep learning-based approach designed for semantic segmentation and object detection in intelligent transportation systems. It includes a lightweight network and a multiscale adapter network, enhancing the original Swin Transformer's capabilities in specific application areas.
- **ESwin Transformer:** This version redesigns the patch embedding and merging modules of the original Swin Transformer, focusing on hierarchical and data-efficient transformations

6. Conclusion

In This project, we rebuild the swintransformer model and do many ablation study on it. It features a hierarchical structure with shifted windows for efficient processing, enabling it to handle various scales and complexities in image analysis. As a key element of Swin Transformer, the shifted win- dow based self-attention is shown to be effective and effi- cient on vision problems. Besides, other tricks like relative position and drop path can also imporve model performance.

7. References

[1] Liu Z, Lin Y, Cao Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows[C]//Proceedings of the IEEE/CVF international conference on computer vision. 2021: 10012-10022.

Besides, We refer to the pytorch code provided by the authors of the original paper to understand the details of the model as a way to build our model

8. Appendix

The Appendix should contain an abundance of detail which may not be suitable for the main text)

8.1 Individual Student Contributions in Fractions

	UNI1	UNI2	UNI3
Name	Zirui Zhang	Xinyi Ke	Yuning Han
Fraction of (useful) total contribution	34%	34%	32%
What I did 1	Build models with Tensorflow. Assist in modifying training scripts.	Write training scripts. Refine the overall code framework.	
What I did 2		Processing datasets. Conduct most of the experiments	Participation in a small number of experimental sections
What I did 3	Modify the lab report	Modify the lab report	Complete most of the lab reports

7.2 Support Material

The pytorch implementation by the authors of the original paper：
https://github.com/microsoft/Swin-Transformer