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

DEEP LEARNING FINAL PROJECT

BAHAAR KHALILIAN, HESAM MOHEBI

Table of Contents

1. Paper Summary *2*

2. Methodology *2*

2.1 The Pipeline 2

2.2 Swin Transformer Block 3

2.3 Architecture Variants 4

3. Experiments *4*

3.1 Evaluation Results for Image Classification on ImageNet-1K 4

3.2 Evaluation Results for Object Detection on COCO 5

3.3 Semantic Segmentation on ADE20K 5

3.4 Ablation Studies 5

4. Code *6*

4.1 Image Classification 6

4.2 Object Detection 7

4.3 Semantic Segmentation 7

5. Limitations *7*

5.1 Instability with Large-Scale Models and Datasets 7

5.2 Relative Positional Bias Limitations 8

5.3 Limited Generalization to Downstream Tasks 8

6. References *8*

1. Paper Summary

The Swin Transformer [1] was first presented at ICCV 2021 (International Conference on Computer Vision). It is introduced as a general-purpose vision backbone, capable of a variety of computer vision tasks, three of which are tackled in this paper: image classification, object detection, and semantic segmentation.

The goal of this paper is to modify the original Transformer architecture to address domain-specific challenges in vision. It also sets out to solve the limitations of previous models like ViT (Vision Transformer) [2] and DeiT (Data-efficient Image Transformers) [3]. Limitations such as:

- The quadratic computational complexity that comes with applying global self-attention in high-resolution images.
- Inflexibility for dense prediction tasks such as object detection.

Swin Transformer overcomes these limitations by introducing a hierarchical architecture with shifted window-based self-attention. This design significantly improves efficiency and scalability, reducing the complexity to linear with respect to image size while still enabling long-range context modeling.

The proposed model not only outperforms previous models but also introduces and encourages the possibility of joint modeling of visual and textual signals.

2. Methodology

In this section, we first start by going through the complete pipeline of Swin-T (Swin Tiny), explaining each stage of the architecture. Next, we will dive into specific components and explain them thoroughly.

2.1 The Pipeline

As seen in Figure 1, the pipeline starts with an RGB image as input, with H as height and W as width. Then, the image is split into fixed-size non-overlapping patches. In this paper, the initial patch size is considered as 4 pixels. Each patch will have a feature space of $4 \times 4 \times 3 = 48$.

In stage one, we have a total feature space of size $\frac{H}{4} \times \frac{W}{4} \times 48$. These features first go through a linear embedding layer, where the feature space is mapped into an arbitrary dimension C. The next step is the Swin Transformer Block, which we will explain later on. The output of stage one is a feature space of size $\frac{H}{4} \times \frac{W}{4} \times C$.

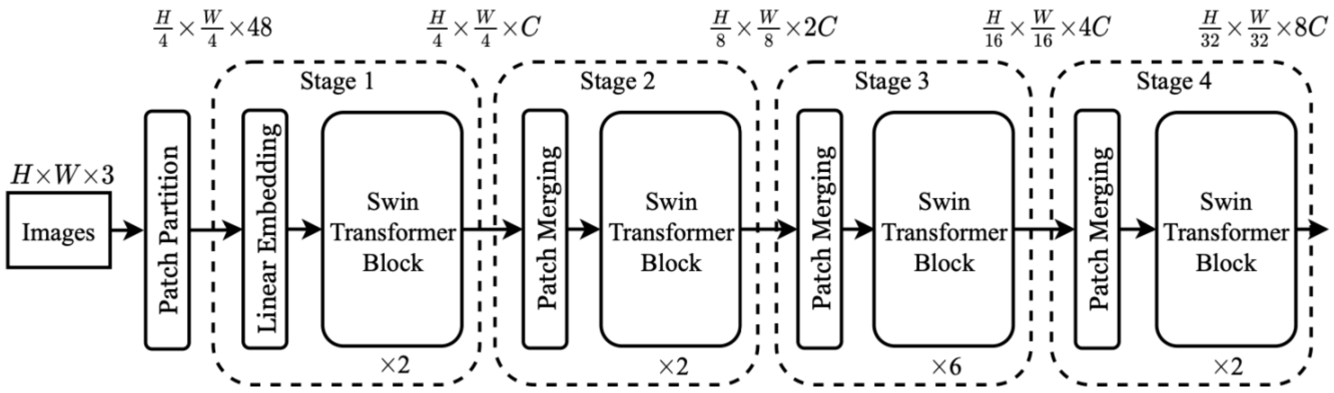

Figure 1. Swin-T architecture

Stage two is where we start with the hierarchical architecture. In this step, neighbourhoods of 2×2patches are merged to form one larger patch. After the merge, each patch has a size of 8×8 pixels. Since the feature spaces of all four neighbours are concatenated, the new feature space will have $\frac{H}{8} \times \frac{W}{8} \times 4C$ dimensions. After a linear mapping, the final output is a feature space with $\frac{H}{8} \times \frac{W}{8} \times 2C$dimensions.

The pipeline continues with the next stages. Spatial resolution is reduced while the channel depth is increased. The overall summary can be seen in Table 1.

Stage	Spatial Size	Channel Size
Input	H × W × 3	3
Patch Embedding	H/4 × W/4	48 → C
Stage 1	H/4 × W/4	C
Stage 2	H/8 × W/8	2C
Stage 3	H/16 × W/16	4C
Stage 4	H/32 × W/32	8C

Table 1. Pipeline summary

2.2 Swin Transformer Block

As seen in Figure 2, each Swin Transformer Block replaces the standard multi-head self-attention (MSA) of the original transformer with Window-based MSA (W-MSA) and Shifted Window MSA (SW-MSA) modules.

- W-MSA calculates self-attention within the local windows and not globally. This modification reduces the quadratic complexity to linear complexity.

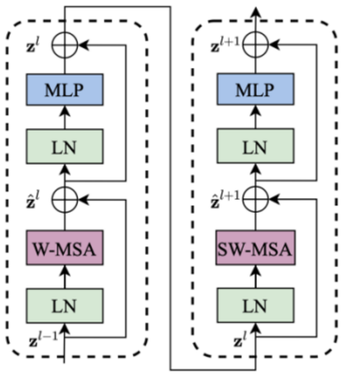

Figure 2. Two Successive Swin Transformer Blocks

- SW-MSA enables connection between the windows. So the partitions that at first had no connection are now shifted to bridge non-overlapping windows. This is done via a cyclic shift (see Figure 3).

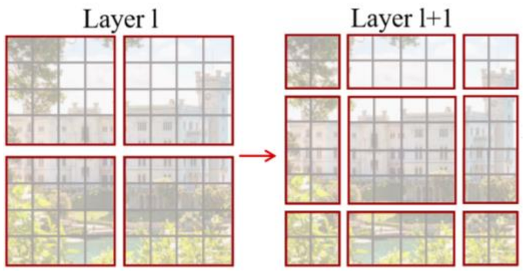

Figure 3. Shifting Window Mechanism

Other components of the architecture are Layer Normalization (LN), Multi-Layer Perceptrons (MLPs), and residual connections in a transformer-style structure.

2.3 Architecture Variants

Swin Transformer is replicated in multiple model sizes to meet different applications and computational requirements. The difference among the architectures is shown in Table 2. Each variant uses a window size of 7×7, a head dimension of 32, and an MLP expansion ratio of 4.

Variant	Channels (C)	Layers per Stage	Relative Size
Swin-T	96	{2, 2, 6, 2}	~0.25× ViT-B
Swin-S	96	{2, 2, 18, 2}	~0.5× ViT-B
Swin-B	128	{2, 2, 18, 2}	ViT-B/DeiT-B
Swin-L	192	{2, 2, 18, 2}	~2× ViT-B

Table 2. Architecture Variants settings

3. Experiments

In this section, we will go over the results obtained by multiple experiments conducted by the authors. It is worth noting that we only cover the information provided in the paper. In the next section, we will discuss the reproducibility of the code and the results we obtained.

3.1 Evaluation Results for Image Classification on ImageNet-1K

The ImageNet-1k dataset (1.28M training, 50K validation, 1,000 classes) was used for this task. The training was done in two settings.

- Regular: AdamW optimizer, 300 epochs, cosine decay LR scheduler, linear warm-up (20 epochs), batch size 1024, LR=0.001, weight decay=0.05. Data augmentations mostly follow DeiT except for repeated augmentation and EMA.
- Pre-trained: Pre-train on ImageNet-22K (14.2M images, 22K classes), then fine-tune on ImageNet-1 K. Batch size 4096, LR=0.001, weight decay=0.01, then fine-tuned with batch size 1024 and constant LR=1e-5.

Compared to CNNs (e.g., RegNet, EfficientNet) and earlier ViTs (e.g., ViT, DeiT), the Swin Transformer outperforms at different model sizes (Swin-L (384^2) achieves state-of-the-art top-1 accuracy (87.3%) among all models tested). They show the best balance between accuracy, speed (throughput), and model size. And their design scales well with ImageNet-22K pretraining, achieving state-of-the-art results in a computationally efficient manner.

3.2 Evaluation Results for Object Detection on COCO

The dataset used here is COCO 2017 (118K train, 5K val, 20K test-dev). Several backbones were evaluated with frameworks. Such as Cascade Mask R-CNN, ATSS, RepPoints v2, and Sparse RCNN. The training started with Multi-scale training (480–800 short side, max 1333), AdamW (LR=0.0001), weight decay=0.05, batch size 16, 3x schedule (36 epochs).

The comparison is done in three areas:

- Framework-Level Comparison: Swin-T is compared with CNN-based backbones across multiple object detection frameworks. As a result, replacing ResNet-50 with Swin-T improves Average Precision (AP) across all detection methods without major computational cost.
- Backbone-Level Comparison within Cascade Mask R-CNN: This time, we evaluate how Swin Transformers compare to other backbones in a fixed detection framework (Cascade Mask R-CNN). The insight here is that Swin Transformers are stronger and more scalable backbones than CNN-based models at every size tier (Tiny, Small, Base).
- System-Level Comparison: Finally, we compare full object detection systems. Here we again come to the conclusion that Swin-L + HTC++ is the best overall model on COCO as of this benchmark, especially with multi-scale testing.

3.3 Semantic Segmentation on ADE20K

With the ADE20K dataset (150 classes, 20K train, 2K val, 3K test). And Upernet as a framework, we evaluate Swin using the val mIoU and test scores, alongside compute and speed metrics.

As a result, Swin Transformers are the new state-of-the-art for semantic segmentation. We see that their performance scales smoothly with model size. With a 62.8 test score, they outperform traditional CNN backbones and even earlier Transformer-based models (like SETR).

3.4 Ablation Studies

The results for this section are shown below. Shifted Windows provides consistent gains across all tasks.

- +1.1% top-1 accuracy on ImageNet-1K.
- +2.0 box AP on COCO detection.
- +0.8 mIoU on ADE20K.

4. Code

In this section, we try to run the code provided for this paper and evaluate various metrics.

4.1 Image Classification

For this task, the authors had provided the GitHub[1] link to the trained models and also a walkthrough for implementing Swin Transformer for image classification.

We started by cloning the official Swin Transformer repository. The next step was to make sure the dependencies were met in the virtual environment. Then, we needed the validation[2] dataset of ImageNet-1K as well as its Dev kit. Then we moved validation images into class subfolders (initially with numerical names). Finally, we attempted to run the evaluation script provided.

In Table 3, we provide a list of the challenges we met along the way and their solution.

Challenge	Discription	Solution
DistributedDataParallel error	The script relied on PyTorch DDP but failed due to missing init_process_group() and use of libuv, which is unsupported on Windows.	Bypassed DDP by changing the script and adjusting it for single-GPU evaluation.
Poor accuracy due to class folder mismatch	Numeric labels (class0, class1,...) were used instead of required WordNet synsets (n01440764,...).	Mapped folders to correct WordNet synsets (nxxxxx/ format).
Incorrect meta.mat usage	meta.mat contained 1860 classes instead of 1000; needed to filter the correct ones.	Reorganized dataset using ILSVRC2012_validation_ground_truth.txt and first 1000 entries from meta.mat.

Table 3. Challenges of running the code along with their solutions

As you can see in Table 4, after these adjustments, we were able to reproduce the results obtained by the authors.

Model	Pre-trained	Top-1	Top-5	Loss	Avg Time (s)	Memory (MB)
Swin-Tiny	No	81.172%	95.520%	1.1826	0.294	515
Swin-Small	No	83.212%	96.238%	1.0400	0.320	597
Swin-Base	No	83.420%	96.448%	1.0938	0.348	876
Swin-B (384^2)	Yes	80.910%	96.006%	1.0576	0.299	515
Swin-L (224^2)	Yes	86.440%	98.052%	0.9390	0.465	1539
Swin-L (384^2)	Yes	86.256%	97.882%	1.0186	0.650	2719
Swin-L (384^2, fine-tuned)	Yes	87.260%	98.240%	0.9766	0.952	4285

Table 4. Our results

[1] https://github.com/microsoft/Swin-Transformer
[2] https://image-net.org/challenges/LSVRC/2012/2012-downloads.php

4.2 Object Detection

The goal was to reproduce the Swin Transformer results on the COCO dataset using the official Swin-Object-Detection implementation[3], which is based on MMDetection. We tried this in a Windows environment using WSL (Windows Subsystem for Linux) and Anaconda. However, we faced some compatibility issues, primarily with mmcv-full. Additionally, the original Swin-OD implementation is built on MMDetection 2.x and MMCV 1.x, while the current ecosystem has shifted to MMDetection 3.x and MMCV 2.x. Massive API and architecture shifts make it difficult to update old scripts without extensive refactoring.

4.3 Semantic Segmentation

Similarly, our attempt to reproduce Swin Transformer's performance on the ADE20K dataset for semantic segmentation was unsuccessful. The implementation used MMSegmentation, which also depends on the MMCV ecosystem. Like in the object detection case, the segmentation scripts were written for older versions of MMCV and do not work with the updated MMEngine-based design. Any attempt to upgrade the dependencies caused numerous import and configuration errors, while downgrading MMCV to older versions broke support for newer CUDA or PyTorch versions, creating a dependency deadlock.

5. Limitations

In this part, we will discuss the limitations of Swin Transformers. It is worth noting that although these limitations are not explicitly mentioned in the original paper, the authors published a follow-up paper in 2022, titled "Swin Transformer V2: Scaling Up Capacity and Resolution"[4], which addresses some of the limitations of the original Swin Transformers. This section is based on the second paper and our understanding.

5.1 Instability with Large-Scale Models and Datasets

Even though this issue was not visible with smaller scales, as the model and the dataset grow (e.g., training on ImageNet-22K or scaling the model to 1 billion parameters), Swin becomes unstable and ends in exploding gradients. So, it needs careful hyperparameter tuning.

Swin V2 uses pre-normalization (instead of Swin V1's post-normalization) to help with gradient flow. Also, RMSNorm (Root Mean Square LayerNorm) replaces standard LayerNorm. These two changes make the Swin transformer more reliable, even with 1B+ parameters.

[3] https://github.com/SwinTransformer/Swin-Transformer-Object-Detection?tab=readme-ov-file#swin-transformer-for-object-detection

5.2 Relative Positional Bias Limitations

Swin V1 introduces learnable relative position bias, but it's tied to the size of the window. So, when the model is transferred to a new resolution or window size, it won't be able to generalize and must be retrained (it is not scale-invariant).

Swin V2 solves this issue by introducing continuous log-spaced relative position bias. This allows generalization to arbitrary window sizes or image resolutions without retraining (scale-invariance is achieved).

5.3 Limited Generalization to Downstream Tasks

While V1 performs well on classification, its segmentation and detection performance still lag behind some CNN hybrids. Swin V2 achieves state-of-the-art on both these tasks. Results can be seen in Table 5.

Task	Swin V1(Base)	Swin V2(Base)	Swin V2(Giant)
ImageNet-1K Top-1	~83.5%	~85.5%	87.1%
COCO box AP	~51.6	~54.4	87.1%
ADE20K mIoU	~48.1	~51.6	55.5

Table 5. Swin V1 vs Swin V2

6. References

[1] Z. Liu *et al.*, "Swin Transformer," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[2] A. Dosovitskiy *et al.*, "AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE," in *ICLR 2021 - 9th International Conference on Learning Representations*, 2021.

[3] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of Machine Learning Research*, 2021.

[4] Z. Liu *et al.*, "Swin Transformer V2: Scaling Up Capacity and Resolution," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2022. doi: 10.1109/CVPR52688.2022.01170.