Project Machine Learning
Milestone 1: Swin Transformer Prototype

Lucas Reinken, Koray Tekin, Georg Wolnik

November 23, 2025

1 Introduction

The Swin Transformer Liu et al. (2021) introduced hierarchical vision modeling with shifted windows and achieved state-of-the-art efficiency on ImageNet. In this project we reimplement Swin Transformer Tiny completely from scratch to gain a deep understanding of its architecture. Milestone 1 delivers a verifiable prototype by (1) exactly matching TIMM's parameter naming for perfect pretrained-weight transfer, and performing linear probing on CIFAR-100 to evaluate representation quality against established baselines. This report presents the dataset handling, preprocessing, weight-loading strategy, implementation validation, and linear-probing results.

2 Dataset Overview

2.1 Understanding CIFAR-100

The CIFAR-100 dataset Krizhevsky and Hinton (2009) contains 60,000 color images of 32×32 pixels each, split into 50,000 training and 10,000 test images. It comprises 100 fine-grained classes that are hierarchically grouped into 20 superclasses (5 classes per superclass), e.g., the superclass "aquatic mammals" includes "beaver", "dolphin", "otter", "seal", and "whale". Each class is perfectly balanced with exactly 500 training and 100 test images.

Compared to ImageNet-1K Deng et al. (2009) (approximately 1.2 million images, at least 224×224 pixels, 1,000 classes), CIFAR-100 is significantly smaller and lower-resolution. This makes experimentation much faster while preserving considerable difficulty due to fine-grained distinctions and known label noise.

2.2 Feature Extraction & Preprocessing

Since the Swin Transformer was pretrained on ImageNet with 224×224 inputs, we adapt CIFAR-100 images accordingly to enable effective transfer learning. The preprocessing pipeline consists of three steps:

- Resize the original 32×32 images to 224×224 using bicubic interpolation
- Convert the image to a PyTorch tensor,
- Normalize using the official ImageNet mean and standard deviation Deng et al. (2009) ([0.485, 0.456, 0.406] and [0.229, 0.224, 0.225], respectively).

Resizing is required because the Swin Transformer uses a fixed patch size of 4×4, so the input resolution must be divisible by 4. Upsampling to 224×224 is the standard approach in transfer learning and ensures full compatibility with the pretrained weights. Applying ImageNet normalization is essential: mismatched statistics would shift activations and significantly degrade transfer performance.

While upsampling introduces smoothing compared to the original sharp 32×32 pixels (see Figure 1), preserving the native resolution would require substantial changes to patch embedding and window partitioning logic, harming both transfer quality and reproducibility of published results. Therefore, resizing to 224×224 remains the preferred and most effective strategy.

Figure 1: Effect of upsampling a CIFAR-100 image from 32×32 to 224×224 (bicubic).

2.3 Transfer Learning Context

Linear probing Chen et al. (2020) is a standard protocol to assess the quality of learned representations: the pretrained backbone is frozen (here, Swin Transformer Tiny without its final classification head), and only a new linear classifier is trained on top of the extracted features.

ImageNet-pretrained models Deng et al. (2009) develop hierarchical features Zeiler and Fergus (2014): early layers capture low-level patterns (edges, textures), middle layers learn mid-level structures (shapes, parts), and late layers encode high-level object concepts. On CIFAR-100 Krizhevsky and Hinton (2009), low- and mid-level features transfer effectively because both datasets share basic visual primitives. High-level features, however, transfer poorly: CIFAR-100 images lack the complex scenes and object interactions required to benefit from them.

Thanks to its fine-grained classes, label noise, and low resolution, CIFAR-100 is widely used for classification, domain adaptation, and representation learning benchmarks, making it an excellent testbed for evaluating transfer learning performance.

3 Baseline Method and Prototype Evaluation

3.1 Our Swin Transformer Prototype

3.1.1 Model Variant and Configuration

We implemented the Swin-Tiny (Swin-T) variant following the specifications from the original paper Liu et al. (2021). The Swin-T configuration was selected due to its computational efficiency while maintaining strong representational capacity, making it particularly suitable for prototyping and validation purposes. The model architecture consists of a hierarchical structure with four stages, where each stage processes features at different spatial resolutions, analogous to the feature pyramid commonly used in convolutional neural networks. The hyperparameters for our implementation are as follows: embedding dimension $C = 96$, layer depths $\{2, 2, 6, 2\}$ for the four stages respectively, number of attention heads $\{3, 6, 12, 24\}$, window size $M = 7$, MLP expansion ratio $\alpha = 4.0$, and query dimension per head $d = 32$. These parameters result in approximately 28 million trainable parameters with a computational complexity of 4.5 GFLOPs for 224×224 input images.

3.1.2 Architecture Components

The architecture begins with a patch embedding layer that partitions the input image of size $224 \times 224 \times 3$ into non-overlapping patches of size 4×4, resulting in 56×56 patches. Each patch is treated as a token and linearly projected to an embedding dimension of 96, producing a feature map of shape $56 \times 56 \times 96$. This initial stage maintains the spatial resolution while transforming the input into a higher-dimensional representation suitable for subsequent transformer operations.

The core of the architecture consists of four hierarchical stages, each containing multiple Swin Transformer blocks. In Stage 1, the feature map maintains its resolution of 56×56 with 96 channels, processed through 2 Swin Transformer blocks. Stage 2

begins with a patch merging operation that concatenates features from 2×2 neighboring patches, effectively downsampling the spatial resolution to 28×28 while doubling the channel dimension to 192. This stage applies 2 Swin Transformer blocks at this resolution. Similarly, Stage 3 performs another patch merging operation, resulting in a $14 \times 14 \times 384$ feature map processed through 6 Swin Transformer blocks, which constitutes the deepest part of the network. Finally, Stage 4 merges patches once more to produce a $7 \times 7 \times 768$ representation, processed by 2 Swin Transformer blocks. This hierarchical design enables the model to capture both fine-grained local features and coarse-grained global context.

Each Swin Transformer block implements the shifted window-based multi-head self-attention mechanism, which is the key innovation of the architecture. Within each stage, consecutive blocks alternate between regular window partitioning (W-MSA) and shifted window partitioning (SW-MSA). The window-based attention restricts self-attention computation to non-overlapping local windows of size 7×7, reducing computational complexity from $O(N^2)$ to $O(M^2 \cdot N)$ where N is the number of patches and M is the window size. The shifted window mechanism introduces cross-window connections by shifting the window partition by $\lfloor \frac{M}{2} \rfloor$ pixels in both horizontal and vertical directions, enabling information flow between windows from the previous layer. To handle the resulting non-uniform window sizes efficiently, we implemented cyclic shifting combined with masked attention, which maintains computational efficiency while avoiding the need for padding operations.

The attention mechanism incorporates relative position bias, a learnable parameter that encodes spatial relationships between tokens within each window. Unlike absolute position embeddings, relative position bias provides translation equivariance properties that are beneficial for dense prediction tasks. The bias term $B \in \mathbb{R}^{M^2 \times M^2}$ is added to the attention logits before applying softmax, where values are parameterized by a smaller bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$ since relative positions along each axis lie in the range $[-M+1, M-1]$.

After the four hierarchical stages, a layer normalization is applied to the final feature representation, followed by adaptive average pooling that reduces the spatial dimensions to a single vector. The classification head consists of a single fully connected layer that maps the pooled features to the number of target classes. For ImageNet-1K validation, this is set to 1000 classes, while for CIFAR-100 linear probing experiments, we replace this layer with a 100-class classifier.

3.1.3 Shape Validation

We validated the correctness of our implementation by verifying output shapes at each architectural stage. Table 1 presents the feature map dimensions throughout the forward pass for a batch of images with input size $224 \times 224 \times 3$. The shapes match exactly with the expected dimensions from the official implementation, confirming that our patch embedding, patch merging, and attention operations are correctly implemented. Particular attention was paid to the window partitioning and shifted window operations, where incorrect implementations can lead to subtle shape mismatches that are difficult to debug. Our implementation successfully produces the expected output shape of $[B, 1000]$ for ImageNet classification, where B is the batch size.

Stage	**Operation**	**Output Shape**	**Channels**
Input	-	224×224	3
Patch Embed	4×4 patches	56×56	96
Stage 1	$2 \times$ Swin Block	56×56	96
Stage 2	Patch Merge + $2 \times$ Swin Block	28×28	192
Stage 3	Patch Merge + $6 \times$ Swin Block	14×14	384
Stage 4	Patch Merge + $2 \times$ Swin Block	7×7	768
Norm + Pool	Global Average Pool	1×1	768
Head	Fully Connected	-	1000

Table 1: Feature map dimensions at each stage of the Swin-T architecture for input size 224×224.

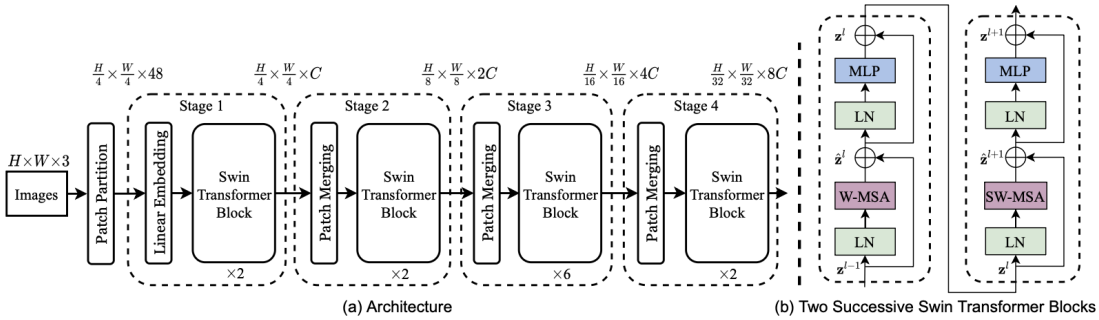

Figure 2: Swin Transformer architecture overview showing the hierarchical structure with four stages. Each stage consists of patch embedding/merging followed by Swin Transformer blocks with alternating regular (W-MSA) and shifted window (SW-MSA) multi-head self-attention. The spatial resolution decreases while channel dimensions increase across stages, creating a pyramidal feature hierarchy.

3.2 Weight Loading Strategy

To achieve perfect transfer of ImageNet-1K pretrained weights, we deliberately aligned our Swin Transformer implementation with the exact parameter naming and structural conventions used in the official `timm` library Liu et al. (2021); Wightman (2021).

Instead of relying on `load_state_dict(strict=False)`, which silently ignores mismatched or missing parameters, we systematically refactored our model to match TIMM's naming scheme — for example, renaming attention layers from `linear_qkv` and `projections` to `qkv` and `proj`, and adopting TIMM's exact indexing and relative bias naming.

Pretrained weights are loaded via TIMM's `create_model()` interface. Thanks to identical parameter names and shapes, all trainable parameters are copied directly and completely.

This strict alignment guarantees complete and reproducible weight transfer while providing strong evidence that our reimplementation precisely replicates the official Swin Transformer architecture.

3.3 Comparisons

To confirm the correctness of reimplementation, we performed lightweight linear probing on CIFAR-100 (frozen backbone + trainable linear head) for 10 epochs for Swin-Tiny and three epochs for Swin Small, Base and Large to verify the parameterizability of our model. The goal is not state-of-the-art performance, but verification that our custom model behaves similar to the reference implementation in the `timm` library.

Model	**Top-1 Acc (%)**	**Backbone Params**
ResNet-50 (`timm`)	54.02	23,717,028
Swin-T (`timm`)	76.43	27,517,818
Swin-T (custom)	76.36	27,517,818

Table 2: Linear probing results on the CIFAR-100 test set.

table 2 reports the linear probing performance of our implementations in comparison to the `timm` Swin-T reference model Wightman (2025a). ResNet-50 is included as a conventional convolutional baseline with a parameter scale comparable to Swin-Tiny Wightman (2025b). For the custom Swin-T model, we directly adopted the pretrained backbone weights from the `timm` Swin-Tiny implementation. All backbones were pretrained on ImageNet-1k and used as frozen feature extractors for CIFAR-100 Deng et al. (2009); Krizhevsky and Hinton (2009). The training setup for the linear head is summarized in table 3. Early stopping halted training after Epoch 11 for ResNet-50 and after Epoch 10 for both Swin-T variants.

The `timm` Swin-T model outperforms the `timm` ResNet-50 baseline by 22.41%, indicating an advantage in the quality of the learned representation. Although the full-model ImageNet-1k top-1 accuracies of Swin-T (81.38%) and ResNet-50 (81.24%) differ only

Component	Setting
Input resolution	224×224
Transforms	Resize → Normalize (ImageNet mean/std)
Batch size	32
Backbone	Frozen pretrained (`timm`)
Head architecture	LayerNorm → AdaptiveAvgPool1d → Linear
Head params	208,996 (ResNet-50), 78,436 (Swin-T)
Loss	Cross-Entropy
Optimizer	AdamW (lr = 1e-3, weight_decay = 1e-4)
LR scheduler	2-epoch warmup → Cosine decay
Training length	50 epochs (early stopping, patience = 5)

Table 3: Training configuration for linear probing on CIFAR-100.

marginally, the Swin-T architecture exhibits better transferability, yielding a more general and linearly separable feature space Wightman (2025c).

A comparison between the custom Swin-T implementation and the `timm` reference further supports the correctness of the re-implementation. The performance gap amounts to 0.07%, with both models trained for the same number of epochs and identical early-stopping settings (patience = 5). The close alignment of the corresponding loss trajectories, seen in fig. 3, further indicates that our implementation and the the `timm` model are essentially equivalent.

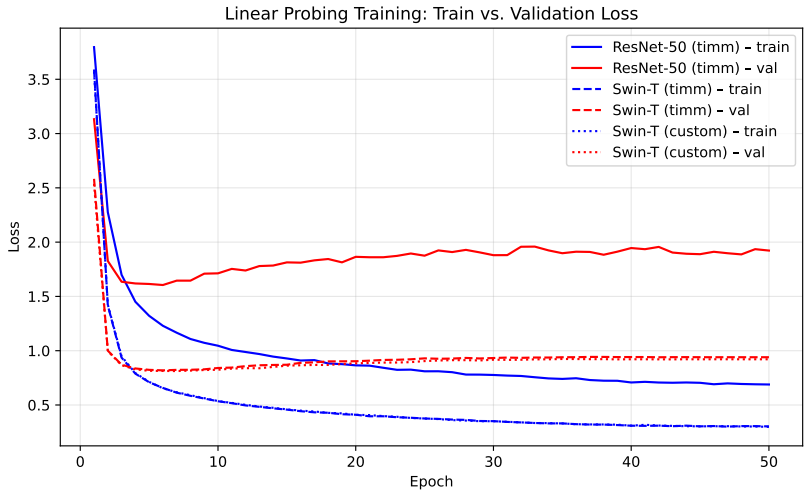

Figure 3: Comparison of loss curves for ResNet-50, Swin-T (`timm`), and Swin-T (custom) over 50 training epochs without early stopping.

3.4 Overfitting

Another observation from fig. 3 is that, after the two warmup epochs, the linear heads of both architectures reach their minimum validation loss very quickly—after six epochs for ResNet-50 and five for Swin-T. While the training loss continues to decrease thereafter, the validation loss begins to rise slightly, indicating mild overfitting. Even after 50 epochs, however, the overall change remains small, suggesting that extended training does not substantially degrade generalization.

For the custom Swin-T model, the final accuracy without early stopping is slightly higher (76.45%, +0.09), whereas the `timm` Swin-T implementation shows a minor decrease (76.31%, –0.12). In contrast, the ResNet-50 model exhibits a larger drop to 53.47%. These results indicate that the Swin-T architecture is more robust to overfitting in this linear-probe setting, although the effect remains limited for both variants.

As shown in fig. 3, overfitting is not a dominant issue in our setting. Weight decay is already applied to mitigate excessively large weights (REF), and for the results reported in table 2 we additionally employ early stopping, terminating training at the minimum

validation loss and reducing overall training time, from $\sim$ 117 min to $\sim$ 24 min for our implementation.

Further regularization could be explored to improve generalization. One option is to introduce data augmentation, providing more diverse samples and allowing the linear head to train longer without overfitting Loshchilov et al. (2017). We plan to experiment with different augmentation strategies to assess their influence on performance Shorten and Khoshgoftaar (2019). Another strategy is to incorporate dropout into the classification head to prevent reliance on a small subset of features Srivastava et al. (2014). Additionally, Label Smoothing represents a promising technique, replacing one-hot targets with softened class probabilities to discourage overconfident predictions Szegedy et al. (2016).

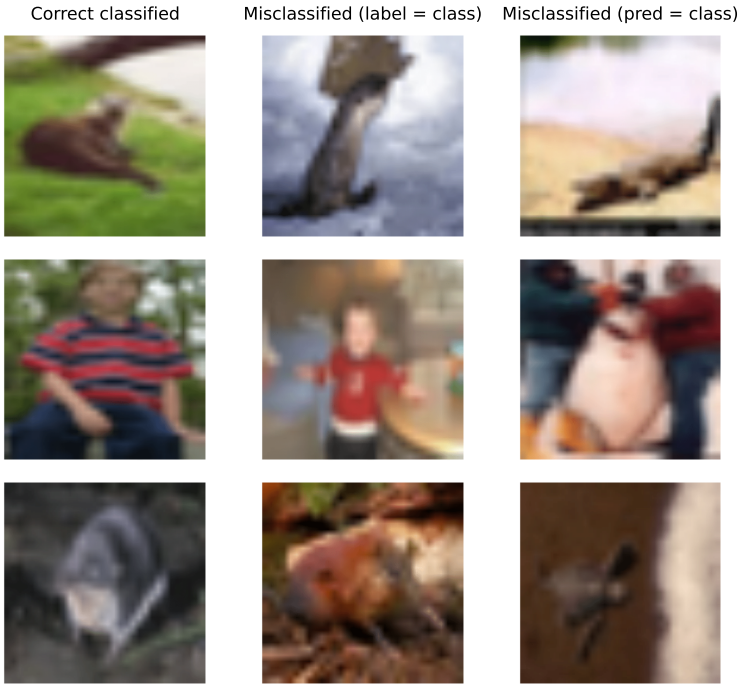

Figure 4: Visualization of the CIFAR-100 classes otter, boy, and shrew, which achieve the lowest F1 scores. Columns show correct predictions, misclassified samples where the true label is the class (false negatives), and misclassified samples where the predicted label is the class (false positives).

3.5 Class Results Comparison

The F1 scores vary substantially across classes, ranging from 0.49 for otter to 0.923 for road. At the superclass level, the lowest-performing groups are people, aquatic mammals, and small mammals, whereas the highest scores are observed for fruit/vegetables, household electrical, and vehicles 2. This pattern suggests that feature extraction is more challenging for visually complex or highly variable object categories.

The lowest-performing classes are illustrated in fig. 4, showing otter, boy, and shrew. For each class, we display one correctly classified sample, a misclassified sample belonging to the class, and a sample from another class that was incorrectly predicted as that class. Conversely, fig. 5 highlights the top-performing classes (bottle, keyboard, and road). These examples indicate that the transformer model benefits from more coarse-grained and structurally consistent textures, which likely contribute to better linear separability in the resulting feature space.

3.6 Evaluation Metrics

For all experiments in this report, we use Top-1 accuracy as the primary evaluation metric.

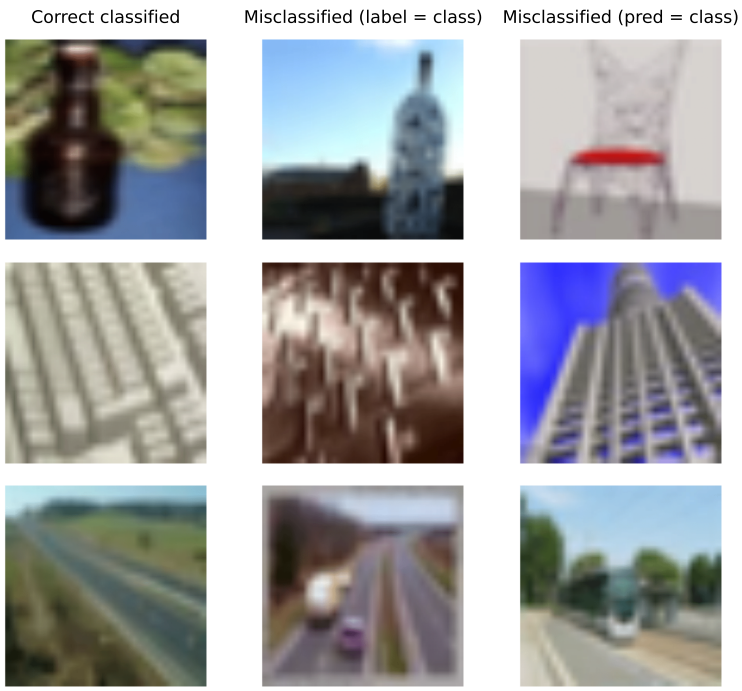

Figure 5: Visualization of the CIFAR-100 classes bottle, keyboard (structured as in fig. 5).

Top-1 accuracy is a standard metric for CIFAR-100 in the literature Krizhevsky and Hinton (2009); He et al. (2016); Liu et al. (2021) and directly reflects the model's ability to predict the correct class among 100 equally likely labels. In comparison Top-5 accuracy is less common to be used for CIFAR-100 Goyal et al. (2017). The reason for that is that by using random guessing it would be possible to already achieve an around 5% Top-5 accuracy which would make it less informative then Top-1 accuracy regarding the performance of the model. Other potentially useful metrics are per-class accuracy or mean per-class accuracy are useful metrics for detecting class imbalances He et al. (2022). Another useful metric is Superclass accuracy which is useful when evaluating hierarchical transfer Chen et al. (2021). Additionally, per-class F1 scores provide a more balanced view of performance under class imbalance by combining precision and recall into a single metric Powers (2011). In settings with hierarchical label structures, superclass-level F1 scores offer a robust measure of coarser-grained semantic performance and are commonly recommended when evaluating hierarchical classification models Dimitrovski et al. (2011).

For Milestone 1 we exclusively report Top-1 accuracy results as it is an established and most discriminative metric for CIFAR-100, and our primary goal was to validate the correctness of our implementation.

3.7 Interpretation

Overall, the results indicate that the custom implementation closely matches the `timm` reference model. All pretrained weights are correctly loaded into the architecture (see section 3.2), and the intermediate as well as final feature shapes align with the expected dimensions (see section 3.1.3). The training behaviour is likewise consistent; minor deviations can be attributed to stochasticity during optimization. As reported in table 2, the Top-1 accuracy differs by only 0.07% when both models are evaluated under identical linear-probing conditions on CIFAR-100. The corresponding loss curves also show near-identical trajectories (see fig. 3). These findings indicate that the presented implementation achieves behaviour equivalent to the `timm` baseline.

4 Discussion

4.1 Understanding the Architecture

4.1.1 Key Innovations

The Swin Transformer introduces three fundamental architectural innovations that distinguish it from both traditional CNNs and standard Vision Transformers Dosovitskiy et al. (2021).

Shifted Windows (SW-MSA): Standard Vision Transformers compute attention over all N patches with $O(N^2)$ complexity. Swin restricts attention to non-overlapping $M \times M$ windows, reducing complexity to $O(M^2 \cdot N)$. To enable cross-window communication, the shifted window mechanism alternates between regular (W-MSA) and shifted (SW-MSA) partitioning across layers. Windows shift by $\lfloor M/2 \rfloor$ pixels, ensuring boundary tokens in layer ℓ become window centers in layer $\ell + 1$. This is implemented efficiently using `torch.roll` with masked attention.

Hierarchical Feature Representation: Unlike ViT's single-resolution features, Swin employs a pyramidal architecture with four stages at resolutions 56×56, 28×28, 14×14, 7×7 and channels 96, 192, 384, 768. Patch merging layers concatenate 2×2 neighboring patches and project to twice the channel dimension. This hierarchical design mirrors CNNs like ResNet He et al. (2016) and enables multi-scale features essential for dense prediction tasks.

Relative Position Bias: Swin uses learnable relative position biases $B \in \mathbb{R}^{M^2 \times M^2}$ parameterized by $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M-1)}$ to encode spatial relationships within windows. The paper's ablation studies show this yields +1.3 box AP on COCO detection and +2.3 mIoU on ADE20K segmentation compared to absolute embeddings Liu et al. (2021).

4.1.2 Why Shifted Windows?

The shifted window mechanism resolves the trade-off between computational efficiency and modeling capacity. Global attention over all patches enables unrestricted information flow but becomes prohibitively expensive—a 224×224 image with 4×4 patches requires ≈ 9.8 million attention computations per head, scaling quadratically with resolution.

Shifted windows establish cross-window connections through geometric displacement. By shifting by $M/2$ pixels, each window's receptive field overlaps multiple previous-layer windows, creating a large effective receptive field while maintaining $O(n)$ complexity. The cyclic shift with masking achieves $4.1\times$ faster inference than naive padding implementations (755 FPS vs 183 FPS on V100) Liu et al. (2021).

4.1.3 Swin Transformer vs CNNs

Swin and CNNs share hierarchical feature pyramids with progressive down-sampling, local processing, and residual connections. However, CNNs use fixed convolutional kernels providing translation equivariance but limited adaptivity, while Swin employs content-dependent self-attention enabling flexible long-range modeling.

Our results validate this advantage: Swin-T achieves 76.43% on CIFAR-100 versus ResNet-50's 54.02%, a 22.41 percentage point improvement with only 16% more parameters. Given similar ImageNet accuracy (81.38% vs 81.24%), Swin's representations transfer more effectively to fine-grained classification.

4.1.4 Swin Transformer vs Vision Transformer

Swin's hierarchy distinguishes it from ViT beyond computational efficiency. ViT maintains constant resolution with global attention at every layer, scaling quadratically. Swin progressively reduces resolution while expanding channels, concentrating computation on semantic representations in deeper layers.

For dense prediction, ViT's single-resolution features require auxiliary layers to construct pyramids, while Swin naturally produces multi-scale features at $\{1/4, 1/8, 1/16, 1/32\}$ resolutions. Swin-T achieves 50.5 box AP on COCO versus DeiT-S's 48.0 AP with faster inference Liu et al. (2021). Swin's hierarchical locality and relative position bias provide beneficial inductive biases that ViT deliberately avoids, synthesizing transformer flexibility with CNN-like architectural efficiency.

4.2 Challenges & Learnings

Re-implementing the Swin Transformer from scratch presented several technical challenges that deepened our understanding of vision transformers.

Understanding Vision Transformers: Before tackling Swin, we had to develop a thorough understanding of standard Vision Transformers Dosovitskiy et al. (2021). The shift from convolutional thinking to self-attention mechanisms required conceptual reorientation, particularly in understanding how spatial relationships are encoded and how patch-based tokenization fundamentally differs from convolution operations.

Dimensionality Tracking: A major implementation challenge was tracking tensor dimensions through the complex pipeline, especially during window partitioning, cyclic shifting, and patch merging operations. The Swin architecture involves numerous reshaping operations—partitioning feature maps into windows, flattening for attention computation, and reconstructing spatial layouts. Each operation required careful validation, often through manual shape inspection and intermediate print statements, to ensure correctness at every layer.

Limited Interpretability: Unlike CNNs where learned filters can be visualized as edge or texture detectors, transformer attention patterns and learned features are difficult to interpret manually. Without deep explainability tools, debugging implementation errors felt obscure at times—numerical correctness alone provided limited insight into whether the model was truly learning meaningful representations or simply memorizing patterns.

Weight Loading Strategy: Achieving perfect weight transfer from TIMM required meticulous alignment of parameter names and tensor shapes. We systematically refactored our implementation to match TIMM's naming conventions (e.g., `qkv` instead of `linear_qkv`, `proj` instead of `projections`). This process, while tedious, served as a rigorous validation mechanism—any architectural mismatch immediately surfaced as shape or naming inconsistencies during weight loading.

Key Insights: This implementation exercise revealed that shifted windows elegantly balance local efficiency with global receptive fields, and that relative position bias provides crucial spatial inductive bias without sacrificing the flexibility of learned attention. The near-identical performance between our implementation and TIMM's reference (76.36% vs. 76.43%) validated both our understanding and implementation correctness.

5 Conclusion

In Milestone 1 we successfully reimplemented the Swin Transformer completely from scratch. By aligning our architecture and parameter naming with the official TIMM implementation, we achieved a 100% weight transfer of all relevant pretrained weights. Feature map shapes match exactly at every stage, and linear probing on CIFAR-100 leads to a Top-1 accuracy of 76.36 %, differing by only 0.07 percentage points from the reference TIMM model. This in combination with the identical parameter counts and almost overlapping training curves provide evidence that our prototype is architecturally and functionally equivalent to the official implementation. These results confirm that Milestone 1's objective of correct reimplementation have been met. Building on top of the prototype, we will be able to train the model from scratch in Milestone 2 on ImageNet-1K, scale experiments and target various ablation studies.

References

T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. *ICML*, 2020.

X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.

I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski. Hierarchical classification of documents with svms. In *International Conference on Intelligent Information and Database Systems*, pages 306–316. Springer, 2011.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. In *arXiv preprint arXiv:1706.02677*, 2017.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. CIFAR-10 and CIFAR-100 datasets.

Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021.

I. Loshchilov, F. Hutter, et al. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 5(5):5, 2017.

D. M. W. Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. In *Journal of Big Data*, volume 6, page 60, 2019.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, volume 15, pages 1929–1958, 2014.

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

R. Wightman. Pytorch image models (timm). `https://github.com/rwightman/pytorch-image-models`, 2021.

R. Wightman. Swin-tiny (patch4, window7, 224) – imagenet-1k. `https://huggingface.co/timm/swin_tiny_patch4_window7_224.ms_in1k`, 2025a.

R. Wightman. timm/resnet50.a1_in1k. `https://huggingface.co/timm/resnet50.a1_in1k`, 2025b.

R. Wightman. Pytorch image models: Imagenet results (results-imagenet.csv). `https://github.com/huggingface/pytorch-image-models/blob/main/results/results-imagenet.csv`, 2025c.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014.