Reimplementation of Swin Transformer

Jiangyan Feng Pengfei Gao Yilin Li Zekun Li
University of Michigan
Ann Arbor, MI

Abstract

In this project, we mainly investigate the trending computer vision network Swin Transformer[1] and its variants. We reimplemented Swin Transformer and its updated version V2[2]. We evaluated our models with an image classification task on ImageNet-1K[3] as well as a downstream classification task on a private LCD screen defect dataset. Furthermore, We also explored other application scenarios such as applying the swin transformer to the modified Alphazero[4] network training. Both two tasks achieved remarkable results.

1. Introduction

1.1. Swin Transformer

Transformer[5] is one of the most popular computer vision architectures in recent years. Originating from natural language processing problems, applying transformers in computer vision tasks comes across issues including different scale of visual objects and high-res pixels[1]. Swin Transformer successfully remedies these problems by introducing new techniques hierarchical architecture and shifted windows. Swin Transformer and its variances have achieved state-of-the-art results in many vision tasks.

1.2. Motivations and contributions

Since Swin Transformer has achieved such outstanding and subversive success in computer vision, we would like to enhance our understanding and mastery of this architecture by going through the whole process of building and evaluating this network. In our project, we firstly reimplemented the Swin-Tiny (Swin-T) architecture introduced in the original paper. We evaluated our model on ImageNet-1K dataset[3] and compared to the results in the paper, as well as on a downstream classification task. Moreover, we investigated the Swin Transformer V2 network, which is an updated version of Swin Transformer with better performance. Finally, we explore the applicability of Swin Transformer in other scenarios such as the AlphaZero network.

2. Related works

2.1. Convolutional Neural Networks(CNN)

CNN[6] and its variants have been the dominating deep learning architectures in computer vision tasks. Benefiting from more powerful computation hardwares and larger datasets, CNNs are able to catch different scales of features and representations in images for various downstream tasks. However, the success of Swin Transformer models in different vision tasks proves that the transformer also has the potential to be an important backbone of vision networks.

2.2. Transformer

Vision Transformer(ViT)[7] starts the trends of using transformers in computer vision tasks. It directly applies the transformer technique from NLP to image tasks by dividing the images into patches and calculating self-attention on them. However, ViT does not prove to be applicable as a general-purpose backbone in vision tasks. Swin Transformer explores the potential of the transformer as a general-purpose backbone by generating image representations and evaluating on different downstream vision tasks.

3. Method

3.1. General Architecture

Swin Transformer adopts a hierarchical design. The total architecture is made of four stages, each stage reduces the resolution of the input feature map and enlarges the receptive field. When the image comes into this architecture, we will do a patch partition first. We divide the image into patches and project their raw features to an arbitrary dimension. Then the feature will come into transformer blocks. The number of transformer blocks for four stages are 2, 2, 6, and 2, and there is a patch merging layer between every block in order to decrease the resolution. The overall architecture is shown at Figure 1.

3.2. Transformer Block

The structure of the Transformer Block is shown at Figure 1. For the odd number of blocks, it consists of a

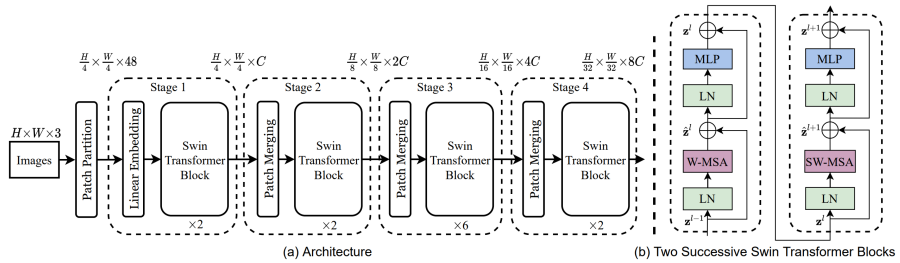

Figure 1. General Architecture of Swin Transformer

shifted window-based MSA module and a 2-layer MLP with GELU in between. In addition, there is one LayerNorm (LN) layer before each MSA module and each MLP. For the even number of blocks, it is the same as the odd number of blocks except the MSA module is replaced by an MSA module with a module based on sifted windows.

3.3. Shifted Window-based Self-Attention

The raw transformer conducts global attention, thus leading to extremely high calculation complexity. Thus, the raw transformer is not suitable for computer vision tasks with high-resolution images. In this architecture, we use self-attention within local windows to solve this problem.

For the self-attention method, the windows can split the image evenly into a bunch of non-overlapping manners. After computing them individually, we can decrease the time complexity from

$$4hwC^2 + 2(hw)^2C \tag{1}$$

to

$$4hwC^2 + 2M^2hwC \tag{2}$$

For MSA module, because it only concentrates on itself and does not consider connections between other windows, its power is limited. To solve this problem, we use a shifted window method to build the connection with different windows. For the shifted-MSA module, the window is shifted from the window of the preceding layer. We use torch.roll function to realize it.

$$z^l = WMSA(LN(z^{l-1}) + z^{l-1}) \tag{3}$$

$$z^l = MLP(LN(z^l)) + z^l \tag{4}$$

$$z^{l+1} = SWMSA(LN(z^l)) + z^l \tag{5}$$

$$z^{l+1} = MLP(LN(z^{L+1})) + z^{l+1} \tag{6}$$

This is the concise computation of the transformer block. The first module is MSA, which partition the feature map from 8 * 8 into 4 * 4. Then, the second module is shifted-MSA module, which can displace the window.

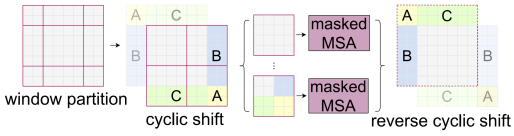

Figure 2. An efficient batch computation approach for self-attention in shifted window partitioning

3.4. Efficient Method for Shifted Configuration

Shifted window partitioning will lead to the increase of numbers of windows. To solve this problem, Swin Transformer uses a more efficient batch computation approach which make it cyclic-shift toward the top-left direction.Using this method, the number of batched windows remains the same. The figure of this method is shown at Figure 2.

3.5. Swin Transformer V2

In order to better scale model capacity and window resolution, several adaptions are made on the original Swin Transformer architecture, just as Figure 3 shows. First, we adapt a post-norm to replace the previous pre-norm configuration. Second, we adapt a scaled cosine attention to replace the original dot product attention. These two methods aims to make the model easier to be scaled up in capacity. And to make the model more effectively transferred across window resolutions, we adapt a log-spaced continuous relative position bias approach to replace the previous parameterized approach.

4. Experiments

In order to evaluate our implementation, we conducted two tasks on Swin-tiny (Swin-T) model and compare its performance with the ResNet-18 model. Both models are implemented based on PyTorch and both tasks are trained through Google Colab with a single Tesla V100-SXM2-16GB GPU.

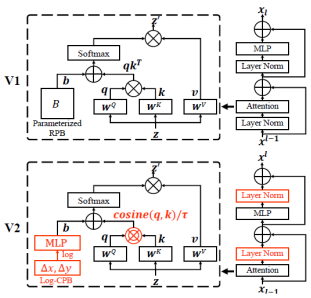

Figure 3. Swin Transformer V2

Parameter	Ours	In paper
Batch size	128	1024
Learning rate	0.0005	0.001
Gradient norm clip	2.0	5.0
Warmup epoch	60	20

Table 1. Hyperparameter setting

Model	#param	FLOPs
Swin-T	29M	4.5G
ResNet-18	11M	1.8G

Table 2. Model comparison

First, We train the swin-T and resnet-18 models from scratch on ImageNet-200 dataset. Second, we perform a downstream classification task on a private LCD screen defect dataset using public pretrained Swin-tiny (Swin-T) model and ResNet-18 model.

Besides, we conduct experiments on CIFAR10 to verify our implementation of Swin Transformer V2-T. Although we currently achieve an accuracy of over 70, there are no official implementation results to compare and our training time is limited, so we will omit it in this report and look forward to further work in the future.

4.1. Model Comparison

From table 2, we can tell that even though Swin-T is already the smallest model in the swin-transformer family, it's still more than two times larger than ResNet-18 in both number of parameters and number of FLOPs. One of the reason why we choose ResNet-18 as the baseline model to compare with is because it is relatively light and thus easier to train. The other reason is that ResNet-18 is a popular baseline and is usually the very first model we try when dealing with small industrial defect sample datasets, so we want to see if Swin-T could surpass the performance of ResNet-18 in both tasks.

4.2. Train from scratch

Datasets In the original paper, one of the experiments the authors conducted was using ImageNet-1K, which contains 1.28M training images and 50K validation images from 1K classes, to train the swin-transfomer from scratch. Here, due to the limitation of computational resource, we make a little change to the original dataset. We carefully choose 200 classes from the ImageNet-1K, each class has 500 training images and 50 validation images. So the total data we use for this task is 100K for training and 10K for validation.
Hyperparameters During the very first few of the training experiments, we try to copy the exact hyperparameters mentioned in the original paper as many as possible. But because we only have a single 16GB GPU, which gives a strict constraint on the batch size, we find that smaller learning rate, more warmup epochs and smaller gradient norm clip threshold are needed to work with a much smaller batch size, and thus prevent the training from gradient explosion and to ensure more stable convergence. So we change the hyperparameters in the training process, and the exact changes are shown in the table 1. And we use AdamW optimizer, linear cosine schduler for both models. And we also use the same transformers and data augmentation described in the original paper for both models.
Results and analysis We trained both Swin-T and ResNet-18 from scratch, and the curves of top-1 accuracy on the validation set of both models are shown in Figure 4. For ResNet-18, we only trained it for 50 epochs before the training process get cut off by Colab, and it already converges to an accuracy of 58%. Then we load in the saved model and continue training, ResNet-18 converges to 62% in the end. For Swin-T, it converges to an accuracy of around 67% with in 210 epochs. According to Figure 4, we can draw the conclusion that ResNet-18 converges much faster and earlier than Swin-T, especially in the first several epochs. But since ResNet-18 is much smaller than Swin-T and has less capacity, Swin-T achieves higher ultimate accuracy. Another thing we have to mention is that Swin-T takes a lot of small vibrations for to climb up, it's sensitive to parameter settings and is relatively harder to train.

(For comparison, we also tested the performance of the public pretrained Swin-T model on this ImageNet-200 validation set.)

4.3. Downstream classification task

We want to test the performance of Swin-T on downstream classification tasks. So we load in the public pretrained models of both Swin-T and ResNet-18, then finetune the models on a downstream classification task. Instead of choosing from natural image datasets, we use a private dataset in the field of industrial quality control that one of our group members once worked on. The industrial dataset is quite different from natural images under natural lights and backgrounds. The dataset is not curated and has

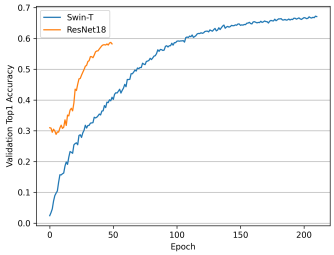

Figure 4. Top-1 Accuracy on Validation Set

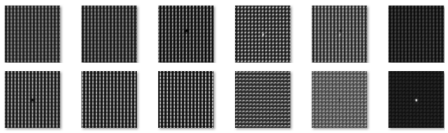

Figure 5. Defect Samples. Top one is highlight dark, bottom one is highlight2.

the problem of high bias and there might also have some mistakes in the label of several samples.

Dataset This is a LCD screen defect dataset which has four classes of defects. These four classes are dark spot, highlight, highlight dark and highlight2. The number of samples in each class is shown in Table. Each defect sample has 6 single-channel gray images, defects are classified by the different combination of dark spots and highlights in different channels. And these channels are captured by different settings of the industrial cameras. The defect is cut from a large LCD screen so it is in the central area of an image. Two examples from this dataset are shown in Figure 5.

Hyperparameters Both models are finetuned for 100 epochs with 10 warmup epochs, with learning rate 1e-3 and weight decay 0.05.

Results and analysis On this LCD defect classification validation dataset, Swin-T converges to an accuracy of 0.9411, while ResNet-18 converges to an accuracy of 0.9518. ResNet-18 achieves 1.17% higher over Swin-T. And from the loss curves of these models in Figure 6, we can tell that although they both get low loss, ResNet-18 converges relatively faster and smoother. There might be two reasons. One is that this is a relatively easy task and don't need a larger model like Swin-T. Another is that, since the defect is in the central part of the image, attention mechanism is actually not that necessary. But we have to mentioned that, unlike telling if an object is cat or dog, the classification in industrial defect detection field is different. The boundary between different defects might not be very clear, and sometimes a defect sample would be given different labels by different people. That's one of the biggest problem when developing defect classification and detection algorithms. From the confusion matrix in Figure 7 of these two models on validation set, we can tell that the biggest difference is that Swin-T messed up highlight defects with highlight2 defects, and has higher accuracy on dark spot class, which means it's more sensitive to small dark spot in image. But in industrial, a dark spot which is smaller than specific scale won't be considered as a defect.

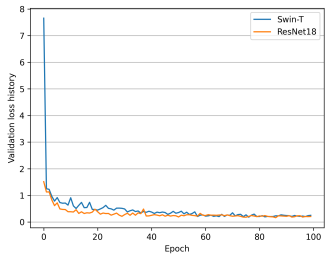

Figure 6. Validation Loss

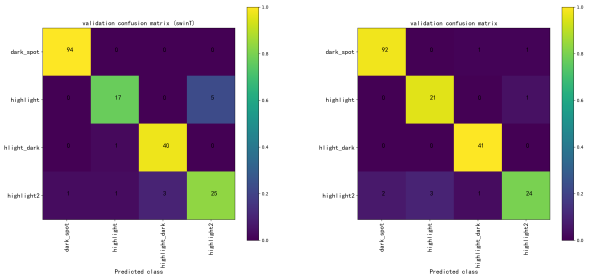

Figure 7. Confusion matrix on validation set

4.4. Further analysis

Swin has ability to learn useful representation of images, especially when have large datasets. Meanwhile, resnets are still a good start when faced with new tasks.

4.5. More application scenarios

Swin Transformer can be integrated into other network to improve their performance. So here we apply it to the Alphazero-gobang[8] network and we find that the performance of the agent is improved compared with other method.

5. Conclusion

In this paper, we reimplement the Swin Transformer and evaluate its performance by training it from scratch and downstream recognition tasks. In the case of limited computing resources, we made relevant adjustments to the training data set and related parameters, and finally achieved good results. We further explore its application as a general-purpose backbone for vision tasks.

References

[1] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[2] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. 2021.

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering chess and shogi by self-play with a general reinforcement learning algorithm, 2017.

[5] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064. PMLR, 10–15 Jul 2018.

[6] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[8] Ruitao Long Zhewei Ye, Yilin Li and Yechen Shi. Alphazero-gobang-3players. https://github.com/Flanker-E/AlphaZero$_{G}obang_{3}player$, 2022.