Swin-transformer: Applying Transformer in Computer Vision Tasks (Deep Learning 2021 Course)

Alina Smolina [1] **Liliya Lemikhova** [1] **Elizaveta Kovtun** [1]
Dmitrii Korzh [1] **Viacheslav Pronin** [1] **Gleb Komissarov** [1]

Abstract

Swin Transformer is a new vision of Transformer that can serve as a general-purpose backbone for computer vision. This project provides experiments on applying Swin Transformer to classification, segmentation, and detection tasks on specialized datasets. In addition, robustness of Swin Transformers was studied through evaluating the performance of adversarial attacks. Metrics, encountered problems and analysis of the results of using this novel and not fully studied architecture are described in details.

1. Introduction

Our team has initially decided to make a project on applying transformer in computer vision tasks for a number of reasons. The underlying motivation is that this understudied architecture has high potential, since it expands the applicability of Transformer such that it can serve as a general-purpose backbone for computer vision, as it does for NLP and as CNNs do in vision. And certainly the scientific attractiveness of this segment also lies in the fact that there are many industrial applications for these tasks, ranging from document fraud detection to marine plastic detection, which are able to optimize the activities of people in various fields significantly. We applied Swin Transformer architecture to visual computing tasks of fundamentally different types, starting with classification task and continuing with semantic segmentation, object detection, and adversarial attack using various datasets, technically described below in the experimental part of the report.

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Alina Smolina <Alina.Smolina@skoltech.ru>.

Final Projects of the Deep Learning 2021 Course, Skoltech, Moscow, Russian Federation, 2021.

2. Literature review

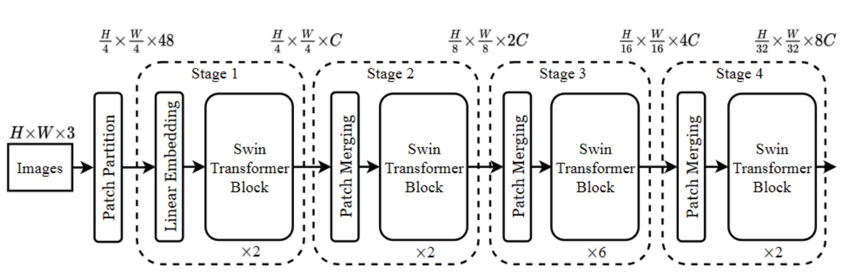
The Swin Transformer was introduced in the article (Liu et al., 2021) to expand the applicability of Transformer (Vaswani et al., 2017). The main idea of the method is based on the construction of hierarchical feature maps where at each level of hierarchy self-attention is applied within local non-overlapping windows, which leads to linear computational complexity to image size. That is one of the major benefits of this architecture in comparison to Vision Transformer (ViT) (Dosovitskiy et al., 2020) which is another application of Transformer in CV. The result of ViT on image classification are encouraging, but its architecture has low-resolution feature maps and the quadratic increase in complexity with image size, so that it is unsuitable for use as a general-backbone network when input resolution is high.

In Swin-Transformer architecture the size of the windows is progressively increased with the network depth (there are some similarities with CNNs). This enables building architectures similar to feature pyramid networks (FPN) or U-Net for dense pixel-level tasks. Also window-based self-attention reduces the computational overhead. As a result, we get flexible and fast method with similar accuracy to the EfficientNet on ImageNet-1K classification.

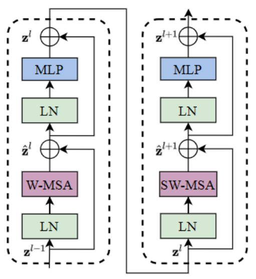
3. Method description

First of all RGB image should be split into non-overlapping patches, for example, with size of 4×4. Each patch is treated as token and its feature is set of concatenated raw RGB pixels (feature dimension then is $4 \times 4 \times 3 = 48$). A linear embedding layer is applied on this raw-valued feature to project it to an arbitrary dimension (denoted as C) as shown on Figure1). Several Transformer blocks with modified self-attention computation (Swin Transformer blocks) are applied on these patch tokens. Two Transformer blocks with the linear embedding are referred to as "Stage 1".

Swin Transformer block (Figure 6b) consist of a shifted window based multi-head attention (MSA) module, followed by MLP with GELU non-linearity in between. A LayerNorm (LN) is applied before each MSA module and each MLP, and a residual connection is applied after each module.



(a) The architecture of a Swin Transformer (Swin-T);



(b) Two Successive Swin Transformer Blocks

Figure 1. (a) Architecture; (b) Swin Transformer Blocks; MSA and SW-MSA are multi-head self attention modules with regular and shifted windowing configurations, respectively.

W-MSA and SW-MSA denote window based multi-head self-attention using regular and shifted window partitioning configurations, respectively.

The shifted window partitioning performs as follows. As illustrated in Figure 2, the first module (on the left) uses a regular window partitioning strategy which starts from the top-left pixel, and the 8×8 feature map is evenly partitioned into 2×2 windows of size 4×4. Then,the next module adopts a windowing configuration that is shifted from that of the preceding layer, by displacing the windows by $(\lfloor \frac{M}{2} \rfloor, \lfloor \frac{M}{2} \rfloor)$ pixels from the regularly partitioned windows.

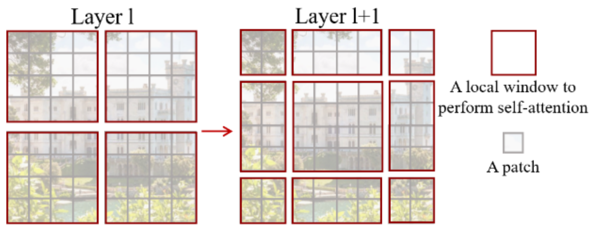


Figure 2. Shifted window approach for computing self-attention

4. Experiments and Results

4.1. Classification

Firstly, we started with running the model and evaluating on ImageNet, as written in start guide. We achieved claimed top-1 accuracy 0.835 on SWIN-B (224x224). Then we decided to apply SWIN transformer for our dataset.

For this task we used original model SWIN-T, SWIN-S and SWIN-B. We used provided pretrained on ImageNet-1K weights for further fine-tuning. For our experiments we used Arctic classification dataset provided by Skoltech's Digital Agro Lab. This dataset contains 1000×1000 images of 20 different plants, growing in arctic station. We applied data augmentation: CenterCrop to delete numerical plant characteristics in the images, RandomResizedCrop to 224×224, HorizontalFlip and Normalization for train dataset; Resize, CenterCrop and normalization for validation and holdout datasets. After this, train dataset had 1903 elements, validation - 636, holdout - 645.

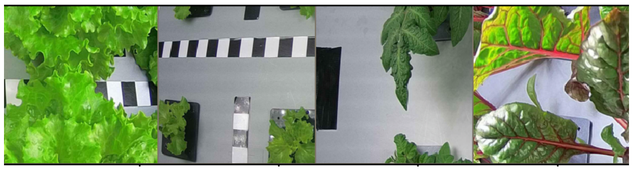


Figure 3. Examples of images in Arctic train dataset.

We trained our model for 25 epochs, using such setup: batch size = 32, cross entropy loss, SGD optimizer (lr = $1e^{-3}$, momentum = 0.9) and CosineAnnealingWarmRestarts scheduler. As a baseline we tried ResNet50 with same training settings. The results you can see in table 1. Results of all configurations of Swin-Transformer outperform results of Digital Agro Lab (DAL), which were obtained with autoencoder and classification of embeddings with SVM.

Table 1. Accuracy of classification on arctic dataset.

Model	params	Accuracy
SWIN-T	27M	99.53
SWIN-S	49M	99.07
SWIN-B	86M	99.07
ResNet(50)	23M	91.6
DAL method	-	92.6

4.2. Semantic segmentation of apples

For this task we used model proposed in original paper, pretrained on ADE20k. This model utilizes UperNet in mmsegmentation and has Swin-Transformer as a backbone. ADE20K is a widely-used semantic segmentation dataset, covering a broad range of 150 semantic categories. It has 25K images in total, with 20K for training, 2K for validation, and another 3K for testing.

We fine-tuned this model with tiny Swin-Transformer as a backbone to segment damaged parts of apples on dataset, provided by Skoltech Digital Agro Lab. There are 3 semantic categories: background, healthy apple and spoiled parts. We had 82 images for train, 26 for validation and 37 for test. Images were collected by different cameras with different light and from different points of view. Top and right viewpoints were in train, left in validation and back in test. All images were resized to 600×400. In training, we employed the AdamW optimizer with an initial learning rate of 6×10^{-5}, a weight decay of 0.01, a scheduler that uses linear learning rate decay, and a linear warmup of 1,500 iterations. Models were trained in Colab for 200 epochs. For augmentations, we adopted the default setting in mmsegmentation of random horizontal flipping, random re-scaling within ratio range [0.5, 2.0], random photometric distortion and random crop of 512×512. The results are presented in tables 2 and 3. An example of segmentation and target mask are provided in figure 4.

Table 2. Per class results of Swin-T for segmentation.

Class	IoU	Acc
background	97.92	98.87
spoiled area	69.06	78.12
healthy apple	74.64	89.6

Table 3. Global results of Swin-T for segmentation.

Scope	mIoU	mAcc	aAcc
global	80.54	88.86	94.45

4.3. Object Detection

All experiments in this part were conducted in MMDetection framework. This is an object detection toolbox that contains a rich set of object detection and instance segmentation methods as well as related components and modules (Chen et al., 2019).

In the first iteration of our object detection experiments we used TACO dataset (Proença & Simões, 2020). This dataset contains 1500 images (100 of them in validation part and 100 - in test) of litter taken under diverse environments. Initially there are 4784 annotations of 60 classes in the dataset. The images have different sizes with the mean height and width of 3320×2350 px. There are 3 annotations per image in average and 68% of them (3253) have area size less than 96×96 px. The samples from the TACO dataset are shown at the Fig.5. It is worth noting that this dataset is relatively recent, so there is not much data on other benchmarks.

We experimented with different sizes of Swin-Transformers pretrained on ImageNet-1K (Swin-T, Swin-B) and with frameworks for object detection (Mask R-CNN, Cascade Mask R-CNN, Faster R-CNN, RepPoints V2). We also varied training parameters such as learning rates, schedule and number of epochs, however the results for bounding boxes were still depressing: average recall $< 12\%$, average precision $< 5\%$, mAP $< 5\%$. Our hypothesis was that the model does not work well with data such as in TACO dataset, namely large input pictures and tiny target objects. In the training pipeline we utilize multi-scale training (resizing the input such that the shorter size is between 480 and 800 while the longer side is at most 1333), as it was mentioned in the original paper (Liu et al., 2021). We also tried to add random cropping before resizing the images, still it did not work.

In order to explore the possible reasons of bad performance on TACO dataset, we decided to experiment with simpler data for object detection with 5000+ images and only one class. We took Elephant Detection Dataset, which is subsampled from COCO dataset and thus has images' sizes much smaller than in TACO. We tested SWIN-T, SWIN-S and SWIN-B with Cascade Mask R-CNN framework to solve the problem of detection elephant species in natural and drone images. As a baseline, we calculated metrics for Cascade Mask R-CNN framework with ResNet-50 backbone. All results are performed in Table 4. It can be seen that model with Swin-T backbone performs better than others. Such results might be due to the fact that the model with less number of parameters is easier to train within the same number of epochs. Mask R-CNN framework showed worse results than Cascade Mask R-CNN with Swin-T backbone: $mAP = 0.400$ for Mask R-CNN against $mAP = 0.745$ for Cascade Mask R-CNN. After training our model, we made an inference on the test set to analyze the obtained results visually. In Figure 6a we can see that model successfully identify two elephants with high confidence. However, there are quite a few examples when several bounding boxes refer to one elephant (as in Figure 6b). This might happen due to big sizes of elephants compared with other things in the picture. But the majority of excessive boxes have low confidence and could be filter out.

Table 4. Results on Elephant dataset.

Model	#epochs	mAP	recall	precision
Swin-T	12	0.745	0.564	0.465
Swin-S	12	0.731	0.561	0.450
Swin-B	12	0.713	0.549	0.441
ResNet-50	12	0.111	0.168	0.029

After all, we have found the source of the problem of low metrics on TACO dataset. We noticed that many classes were never identified, so the recall and precision for these classes was zero, and after macro-averaging this resulted

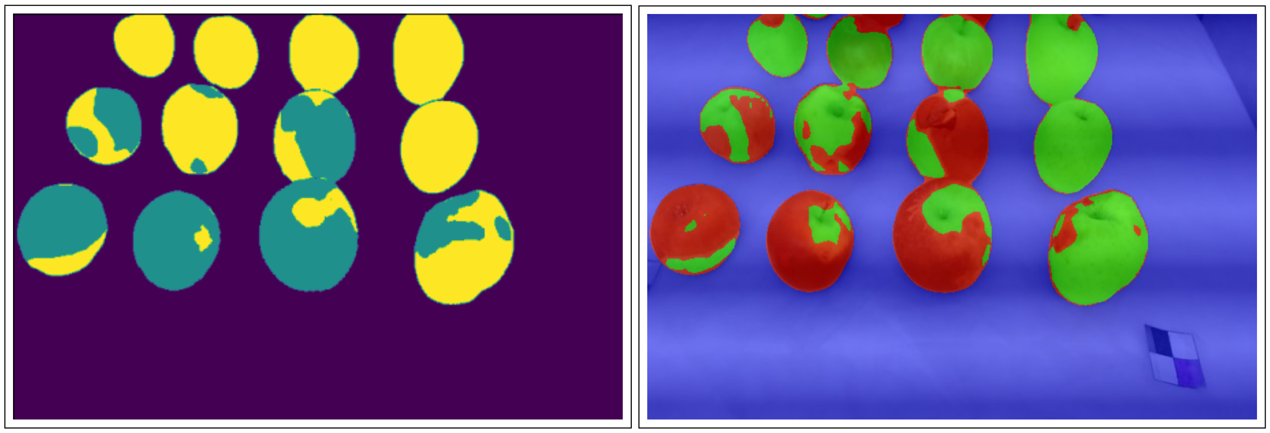

(a) Target (b) Predicted segmentation

Figure 4. Example of segmentation of damaged apples with Swin-T.

in a low metric value. Then we decided to switch from multiclass object detection problem on TACO dataset to a binary one, so that all objects from it were assigned to one class of 'waste'. This indeed produced metrics much higher than before. The results of these experiments are shown in Table 5. In these experiments Cascade Mask R-CNN framework for detection was used.

Table 5. Results for test part of TACO-binary dataset.

Model	**#epochs**	**mAP**	**recall**	**precision**
Swin-T	12	0.221	0.352	0.149
Swin-B	36	0.368	0.518	0.338

Therefore, using Swin Transformers as backbone in object detection frameworks out-of-box gives remarkable results for datasets with small number of classes. However, to get sufficient quality on more complex datasets, models should be investigated and adjusted accordingly.

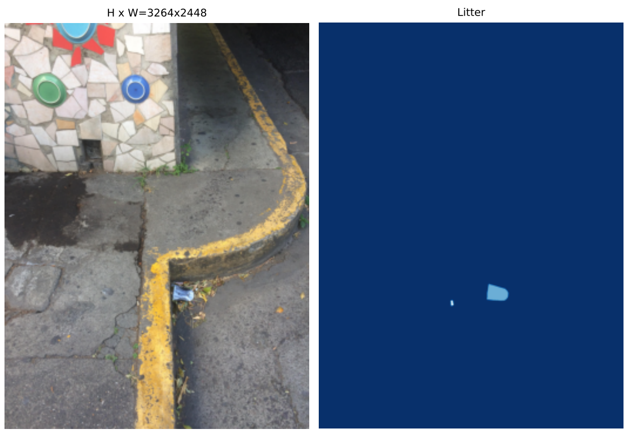


Figure 5. Example of an image and its mask from TACO dataset.

4.4. Adversarial experiments

We decided to investigate a straightforward question: whether SWIN transformer is more robust to adversarial attacks (then, e.g. simple CNN) or not. Although by the time of starting these experiments there was already published an article (Mao et al., 2021), that introduces more the robustest transformer and makes a wide comparison with different models and benchmarks (and with SWIN, outperforming it), we decided to make some experiments.

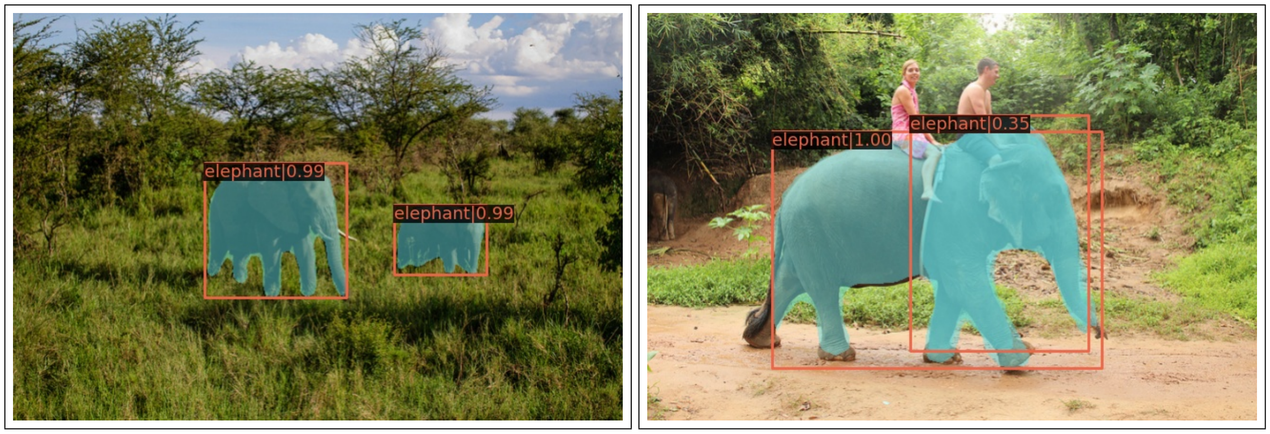
For benchmark we chose MNIST dataset (as simplest), FGSM (Goodfellow et al., 2014) (and generalized FGSM for l_2 norm) and Square attacks (Andriushchenko et al., 2020) (as simplest gradient-based and novel gradient-free).

We faced the problem, that imported model doesn't want to work with number of input dimensions equal to 1 (although it has such initializing parameter). To overcome this problem, we add 2 formal dimensions (filled with zeros) to MNIST images. Also we resized images (bilinear mode), as model works with 224x224 input size.

We changed output layer and used a pretrained on ImageNet-1K weights to finetune model on custom MNIST with cross-entropy loss and SGD optimizer for 1 epoch and achieved ≈ 0.90 accuracy.

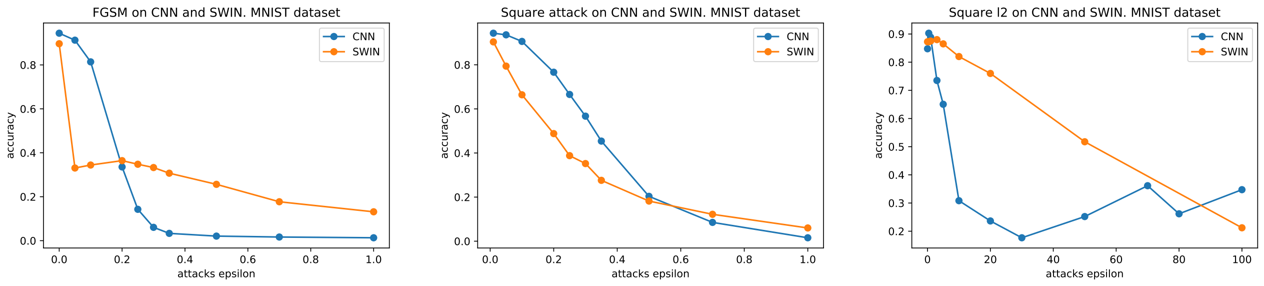
In contrast, we trained simple CNN on original dataset. CNN model has two convolution layers (20 and 50 channels, kernel size=5), ReLU activation functions, max-pooling, and linear layers ($800 \to 500$, $500 \to 10$).

Attack were made via adversarial robustness toolbox (for SWIM model we were resizing input images with numpy function kron). For FGSM SWIN model performed better



(a) Correct detection (b) Incorrect detection

Figure 6. Example of object detection of elephants with Cascade Mask R-CNN framework and Swin-T backbone.



(a) FGSM attack (b) Square attack (c) Square l_2 attack

Figure 7. Accuracy after adversarial attacks.

then CNN for epsilons 0.2 - 1 7a. For square attack CNN mostly outperforms SWIN 7b. The most strange result is shown on 7c. We can't explain it for this moment.

5. Conclusion

In this project, we have learned and experimented with the new Swin Transformer architecture in several computer vision tasks: classification, semantic segmentation and object detection. In addition, we have successfully performed adversarial experiments.

For all the sub-tasks, we had to learn modern open-source toolkits: MMSegmenation and MMDetection, which will definitely be of great help for our future projects.

References

Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C. C., and Lin, D. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

Mao, X., Qi, G., Chen, Y., Li, X., Ye, S., He, Y., and Xue,

H. Rethinking the design principles of robust vision transformer. *arXiv preprint arXiv:2105.07926*, 2021.

Proença, P. F. and Simões, P. Taco: Trash annotations in context for litter detection. *arXiv preprint arXiv:2003.06975*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

A. Team member's contributions

Explicitly stated contributions of each team member to the final project.

Alina Smolina

- Coding for object detection;
- Conducted experiments on TACO dataset;
- Report (object detection);
- Presentation slides;
- Video presentation;

Liliya Lemikhova

- Coding for classification;
- Coding, all experiments and main results for segmentation;
- Report (classification and segmentation);

Elizaveta Kovtun

- Coding for object detection;
- Conducted experiments on Elephant dataset;
- Report (object detection);
- Presentation slides;
- Video presentation;

Dmitrii Korzh

- Coding for adversarial attacks;
- Coding for classification;
- Report (adversarial attacks);
- Presentation slides;
- Video presentation;

Viacheslav Pronin

- Preliminary coding and research on segmentation;
- Report;
- Presentation slides;

Gleb Komissarov

- Preliminary research on segmentation;
- Report;
- Presentation slides;

B. List of third-party code with links

https://github.com/SwinTransformer

https://github.com/open-mmlab/mmsegmentation/tree/v0.11.0

https://github.com/Trusted-AI/adversarial-robustness-toolbox/tree/main/examples

https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

https://mmdetection.readthedocs.io/en/latest/

https://github.com/pedropro/TACO