Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS

Упрощение использования визуальных трансформеров обычными людьми / Bringing Vision Transformers Closer to Individuals

Обучающийся / Student Детков Никита Сергеевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет цифровых трансформаций
Группа/Group J42332c
Направление подготовки/ Subject area 01.04.02 Прикладная математика и информатика
Образовательная программа / Educational program Большие данные и машинное обучение 2020
Язык реализации ОП / Language of the educational program Английский
Статус ОП / Status of educational program МОП
Квалификация/ Degree level Магистр
Руководитель ВКР/ Thesis supervisor Ковальчук Сергей Валерьевич, кандидат технических наук, Университет ИТМО, исследовательский центр в сфере искусственного интеллекта "Сильный искусственный интеллект в промышленности", старший научный сотрудник

Обучающийся/Student

Документ подписан
Детков Никита Сергеевич
20.05.2022

(эл. подпись/ signature)

Детков Никита Сергеевич
(Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Документ подписан
Ковальчук Сергей Валерьевич
20.05.2022

(эл. подпись/ signature)

Ковальчук Сергей Валерьевич
(Фамилия И.О./ name and surname)

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ / OBJECTIVES FOR A GRADUATION THESIS

Обучающийся / Student Детков Никита Сергеевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет цифровых трансформаций
Группа/Group J42332c
Направление подготовки/ Subject area 01.04.02 Прикладная математика и информатика
Образовательная программа / Educational program Большие данные и машинное обучение 2020
Язык реализации ОП / Language of the educational program Английский
Статус ОП / Status of educational program МОП
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Упрощение использования визуальных трансформеров обычными людьми / Bringing Vision Transformers Closer to Individuals
Руководитель ВКР/ Thesis supervisor Ковальчук Сергей Валерьевич, кандидат технических наук, Университет ИТМО, исследовательский центр в сфере искусственного интеллекта "Сильный искусственный интеллект в промышленности", старший научный сотрудник

Основные вопросы, подлежащие разработке / Key issues to be analyzed
Обучение современных Deep Learning моделей зачастую является очень тяжёлым: необходимы огромные GPU мощности, большие датасеты и долгое время для обучения. Это справедливо и к недавно показавшим себя Vision Transformer'ам. Цель работы заключается в том, чтобы найти способ облегчить применение такого рода моделей для людей/компаний, которые не имеют достаточных для этого ресурсов, то есть оптимизировать обучение: уменьшить количество необходимых вычислительных мощностей и/или данных для получения сопоставимых результатов. / Modern Deep Learning models are often very hard to train: huge GPU powers, large datasets, and long training time are required. This is also true for the recently introduced Vision Transformers. The goal of the work is to find a way to ease the application of these kinds of models for people/companies that do not have sufficient resources for it, that is, to optimize training: to reduce the amount of computing power and/or data required to obtain comparable results.

Дата выдачи задания / Assignment issued on: 11.01.2022

Срок представления готовой ВКР / Deadline for final edition of the thesis 18.05.2022

Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет /

not
Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:

Руководитель ВКР/ Thesis supervisor

Документ подписан
Ковальчук Сергей Валерьевич
01.04.2022

(эл. подпись)

Ковальчук Сергей Валерьевич

Задание принял к исполнению/ Objectives assumed BY

Документ подписан
Детков Никита Сергеевич
05.04.2022

(эл. подпись)

Детков Никита Сергеевич

Руководитель ОП/ Head of educational program

Документ подписан
Насонов Денис Александрович
29.04.2022

(эл. подпись)

Насонов Денис Александрович

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

АННОТАЦИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ SUMMARY OF A GRADUATION THESIS

Обучающийся / Student Детков Никита Сергеевич
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет цифровых трансформаций
Группа/Group J42332c
Направление подготовки/ Subject area 01.04.02 Прикладная математика и информатика
Образовательная программа / Educational program Большие данные и машинное обучение 2020
Язык реализации ОП / Language of the educational program Английский
Статус ОП / Status of educational program МОП
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Упрощение использования визуальных трансформеров обычными людьми / Bringing Vision Transformers Closer to Individuals
Руководитель ВКР/ Thesis supervisor Ковальчук Сергей Валерьевич, кандидат технических наук, Университет ИТМО, исследовательский центр в сфере искусственного интеллекта "Сильный искусственный интеллект в промышленности", старший научный сотрудник

ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ DESCRIPTION OF THE GRADUATION THESIS

Цель исследования / Research goal
Оптимизация процесса обучения Vision Transformer для уменьшения необходимого объёма вычислений и данных / Optimize the Vision Transformer training process to reduce required computations and data volume
Задачи, решаемые в ВКР / Research tasks
Обзор архитектуры нейронной сети Vision Transformer, её производных, решений в области оптимизации её обучения и области обучения без разметки. Разработка метода обучения Vision Transformer, позволяющая улучшить качество обучения с меньшим размером входного батча и меньшим объёмом и разнообразием данных, а также ускорить время схождения параметров сети к оптимуму. Проведение экспериментов по обучению Vision Transformer с помощью эталонного, референсного подхода. Проведение экспериментов по обучению Vision Transformer с помощью разработанного метода. Оценка и анализ полученных результатов и проведение тестирования значимости компонентов решения. / An overview of the architecture of the Vision Transformer neural network, its derivatives, solutions for optimizing its training process and the area of unsupervised learning. Development of a training method for the Vision Transformer, which improves the quality of training with a smaller input batch size and a smaller volume and diversity of the dataset, as well as accelerating

the time of convergence of the network parameters to the optimum. Conducting experiments to train Vision Transformer using a baseline state-of-the-art approach. Conducting experiments to train Vision Transformer using the developed method. Evaluate and analyze the results and test the significance of the solution components.

Краткая характеристика полученных результатов / Short summary of results/findings

Разработан метод обучения нейронной сети Vision Transformer, значительно улучшающий значения метрик, полученных эталонным путём обучения. Достигнуто двукратное ускорение времени обучения, повышение качества и скорости сходимости обучения на датасетах с небольшим объёмом данных. / We developed a method for training the Vision Transformer neural network, which significantly improves the metrics values obtained by the baseline training method. A double acceleration of training time, an increase in the quality and speed of convergence of training on datasets with a small amount of data were achieved.

Обучающийся/Student

Документ подписан
Детков Никита Сергеевич
20.05.2022

(эл. подпись/ signature)

Детков Никита Сергеевич

(Фамилия И.О./ name and surname)

Руководитель ВКР/ Thesis supervisor

Документ подписан
Ковальчук Сергей Валерьевич
20.05.2022

(эл. подпись/ signature)

Ковальчук Сергей Валерьевич

(Фамилия И.О./ name and surname)

ABSTRACT

Recently, models free of prior knowledge of the data have been gaining popularity, and they owe this to the opportunity for people to utilize huge clusters for machine learning tasks and use huge size datasets. Without this it is almost impossible to train such a model that can not make assumptions about the data it is receiving. One of such a models is Vision Transformer. We stated our goal as optimizing the Vision Transformer training process to reduce required computations and data volume. Analysing this problem we understood that the signal from supervised objective may be not enough and we added several self-supervised tasks for the model so it can obtain more information every backward pass during training. Also we manipulate input batch so it provide more details keeping memory footprint almost the same. We compared our method of training ViT to existing state-of-the-art recipe of training ViT and report valuable advances, including twice faster speed of the training, significant classification metrics improvements and accelerated convergence.

Contents

GLOSSARY

Classification — The task of classifying an object into one of arbitrary class.

Artificial perceptron — A model that describes the brain's perception, response, and processing of information in terms of mathematics and computer computations.

Neural Network — A function that takes a tensor of an arbitrary shape as an input and performs different mathematical functions over it. It can also be defined as a multi-layer perceptron.

Artificial neuron — One of the nodes of a neural network, which is represented as a mathematical function similar to a neuron – an electrically excitable cell of the nervous system.

Loss function — A function that describes the disagreement between the actual and predicted values.

Backpropagation algorithm — An iterative method for calculating the gradient that is used when updating the weights of a neural network in order to minimize its loss function. It is applicable only in the case of differentiability of activation functions of neurons and other components of the neural network.

Neural network training — The process of training a neural network, which is possible due to the backpropagation algorithm.

Neural network finetuning — A process of additional training of a neural network, using new data, task or components on the model.

Neural network inference — A process of getting the result of the work of the neural network.

Augmentation — A method of data transformations which allows to increase the generalization ability of the model that will be trained on it.

Optimizer — A function optimizing the given parameters. For example, SGD or Adam.

Learning rate — A parameter of the optimizer that determines the step size in the direction of the gradient.

Learning rate scheduler — A "scheduler" that determines the change in the learning rate during training.

Train set — A set of the [labeled] data used for training.

Validation set — A set of the [labeled] data used for validation for the trained model. Has no intersections with train set.

Vanilla — The original, initially proposed version.

Transfer Learning — Application of the preliminarily trained model for the new data it was not trained on.

SYMBOLS AND ABBREVIATIONS

NN — Neural Network.

CNN — Convolutional Neural Network.

lr — Learning rate.

IoT — Internet of Things.

DL — Deep Learning.

CV — Computer Vision.

ViT — Vision Transformer.

GT — Ground Truth.

IoU — Intersection over Union.

FFN — Feed-Forward Network

EMA — Exponentially Moving Average

MLP — Multilayer Perceptron

INTRODUCTION

Computer Vision nowadays is a major part of the intellectual systems used in the variety of gadgets like phones, IoT devices and so on. Since the boom of Deep Learning approaches in Computer Vision a.k.a. ImageNet moment [1], there have been plenty of CNN architectures and methods allowing its application in everyday life scenarios like phone face recognition, photos enhancement and Snapchat filters. Even if those cases are massively studied in the academia and production, there are a lot of open issues for the research community.

Recent achievements in NLP with different application of Transformer architecture [2] pushed the scientists to find a way to apply this architecture to Computer Vision problems. It turns out that the proven standard of CV — CNN models — have the feature that simultaneously bring it to the top metrics and constraining from getting even bigger. This feature is inductive bias.

The domain is now moving towards to Transformer-like architectures because of the lack of the image inductive bias and ability to have a better generalization ability than CNNs. The pioneer of this area is Vision Transformer [3], and together with it presented an outstanding results on image classification task and became a real competitor to CNNs, it introduced a lot of questions for scientific community to research. Among them: how to scale in depth, how to apply for dense prediction tasks and a lot of other issues. But two of the most important — how to train it with small dataset, how to train if you do not have a cluster? The ViT architecture is very hard to train since it does not have any knowledge of the data you pass to it and even what the image is. So you have to give this model to construct own inductive bias itself — this is the difficulty. For example, in the original ViT paper [3] authors reported that pre-training ViT-H/14 takes 2500 TPUv3-core-days — this is extremely unaffordable not even for regular people, but for small companies an institutions also.

In this work we present the training framework which enforces the quicker and more efficient understanding of the data by the model and allowing to train it using affordable batch size and datasets of small size.

Achievements of the work:

1) ViT can be trained using just 1 GPU.

2) ViT can be trained using small dataset.
3) Two times faster training procedure.
4) Up to +25 accuracy on training.
5) Up to +3 accuracy on fine-tuning.
6) Up to 5x times faster convergence on fine-tuning.

We compare all of the stated achievements with the state-of-the-art approach of how to train ViT from the corresponding paper [4]. We call our framework **Tort**. The code can be found here: `https://github.com/detkov/tort`. The logs of experiments can be found here: `https://wandb.ai/detkov/tort`.

1 BACKGROUND

AlexNet a.k.a. ImageNet moment [1] gave academia and companies the proof that Neural Networks, and Convolution Neural Networks in particular, is a very promising direction of study and research. Years after, we have every relevant domain covered by CNN architectures, but we are still trying to find the most efficient way to make image classification, analysis, segmentation, retrieval, detection and apply it to the dozens of other topics.

Why are we still in search, what are the reasons that don't allow us to stay satisfied with the brilliant results that CNNs have already given us? The problem is inductive bias. Inductive bias, in a sense, is the prior knowledge and/or assumption that we give to the model. We can give it with the precise architecture construction that forces the model to process better, for example, sequences with recurrent layers or images with convolution layers, or with the data we train it on — it also have some domain, degree of diversity and quality. Moreover, the way we train neural networks — the backpropagation itself forces it. Thus, inductive bias helps the model to prioritize one generalization method over another, it gives additional information on how to do it.

Convolution Neural Networks have convolution layers, those layers were introduced with the several ideas:

a) To reduce computations w.r.t image resolution, because image processing with linear layers was almost impossible both in 2013 and years after.

b) To give the model an opportunity to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns.

c) To give information about the location of the extracted features.

With this base, CNN models were advancing through the time, but even if its inductive bias helped them all the way up, now it has become the reason for limiting its performance. Local receptive field, the major feature of CNNs, is the constraining factor in achieving the next height.

At the same time Natural Language Processing models were issuing the problem with long sequences that was solved by the introduction of Attention mechanism [5]. The idea behind the attention was to permit the decoder to utilize the most relevant parts of the input sequence in a flexible manner, by a

weighted combination of all of the encoded input vectors, with the most relevant vectors being attributed the highest weights. Thus, it uses the global context, all the sequence, no matter how long it is. Moreover, the next year the Transformer [2] paper appeared and made a revolution in NLP. Transformer is the architecture that uses only attention, without the legacy of GRU [6], LSTM [7] and other methods. After that, Transformer based models became the standard for NLP purposes.

1.1 Vision Transformer

The major step in findings of alternatives of CNN in CV is the ViT [3]. ViT (Vision Transformer) is classification neural network free of convolutions, it mainly uses attention mechanism to extract global information, transforming input image into N patches and processing them independently through the Transformer encoder as a sequence, it is presented in Fig. 1.1.

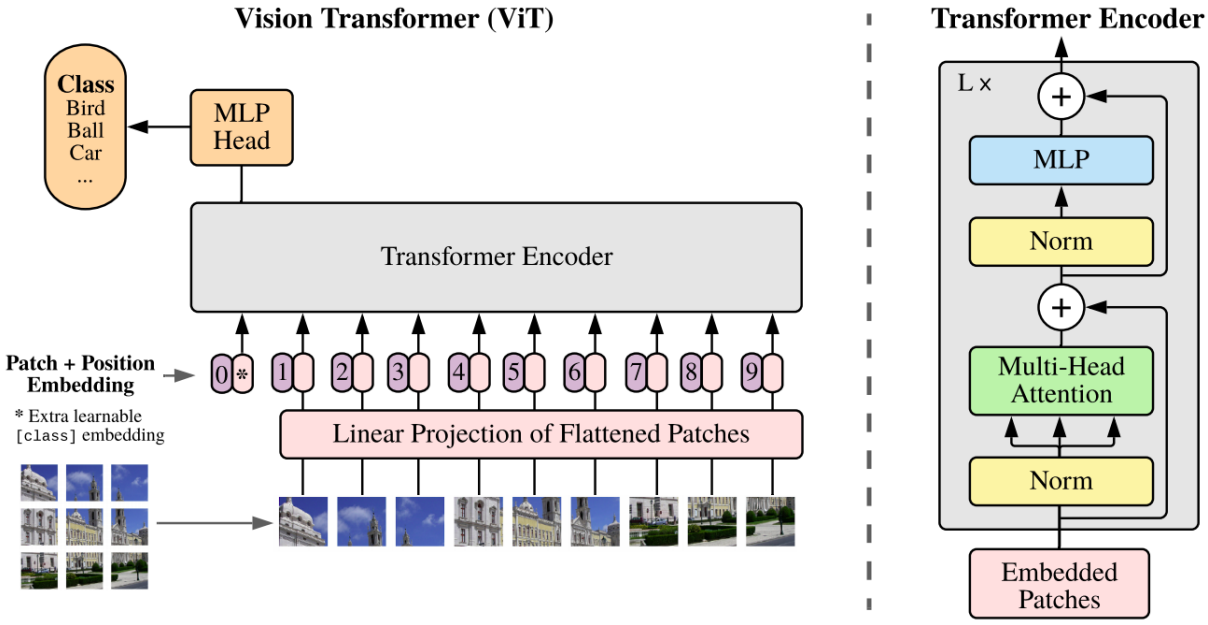

Figure 1.1 — ViT model overview. Taken from original paper [3]

The main value of this work is proof-of-concept that the model, fully free of convolutions can perform effectively with metrics close to CNNs. Authors explored the model and reported that with the lack of convolutional inductive bias it is harder for model to find the way to generalize the data in comparison to vanilla ResNet [8]. They trained ViT and ResNet on different sizes datasets:

ImageNet, ImageNet-21k [9] and JFT-300M [10] and understood that ViT outperforms ResNet only using the largest dataset JFM-300M, because only on this amount of data we see the contribution of the freedom from convolutions bias, when the model can truly look at the global context of the image. This is clearly depicted in Fig. 1.2.

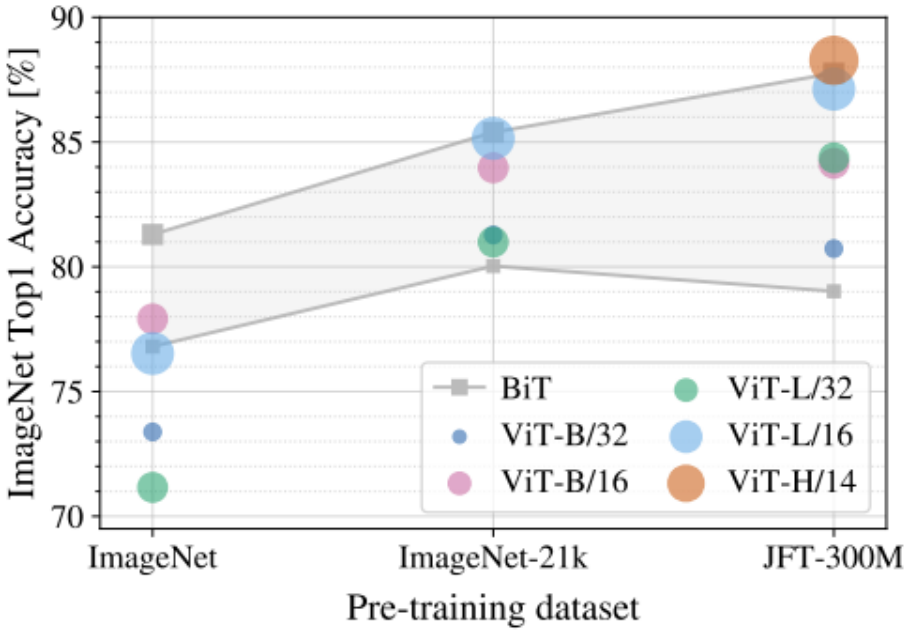

Figure 1.2 — Transfer to ImageNet. While large ViT models perform worse than BiT ResNets (shaded area) when pre-trained on small datasets, they shine when pre-trained on larger datasets. Similarly, larger ViT variants overtake smaller ones as the dataset grows. Taken from original paper [3]

After that promising result, a lot of papers with improvements were published. Among them we can highlight DeiT [11], TNT [12], T2T-ViT [13], ConViT [14], DeepViT [15], Swin Transformer [16], CaiT [17], DVT [18], Shuffle Transformer [19], CoAtNet [20] and UViT [21]. Same way, there were attempts on application of ViT to the dense prediction task, like segmentation and detection. Among them we can highlight DETR [22], SETR [23], SegFormer [24], Segmenter [25], FTN [26] and Lawin Transformer [27].

1.1.1 Details of implementation

The general idea of treat image as sequence is presented in the following form: Transformer [2] taking 1D sequence of embeddings as input, thus we have to transform image to sequence of 1D vectors. Well, we can do it straightforward:

take image, cut it into patches, flatten them and pass sequentially. Speaking more strict, we take image $x \in \mathbb{R}^{H \times W \times C}$, reshape it into $output \in \mathbb{R}^{N \times (P^2 \times C)}$ which is the sequence of flattened pathes, where H, W and C are the image's height, width and number of channels respectively, P is the resolution of the patch and N is the number of patches.

Then we prepare class token as the zeroth embedding received by Transformer encoder so it can be learnable parameter for classification. After that, there are patch embeddings.

As for the position embeddings, in this case they are presented with learnable 1D vectors.

This way we are able to process images using Transformer architecture and utilize it for the CV tasks, like classification.

1.1.2 Vision Transformer variations

Since the introduction of this architecture a lot of efforts were aimed at improving the stability of the training, speed of the convergence, optimization of inner operations and the benchmark results with the help of changing the architecture of the network. In this part we present and describe the main ideas under the hood of the above mentioned papers.

1.1.2.1 DeiT

DeiT [11] stands for Data-efficient image Transformers, authors of this work show that such large networks, as ViTs, can produce a high performance using only open source academic datasets with less computations using the knowledge distillation. They are using RegNetY-16GF [28] (84M parameters, 82.9% top-1 accuracy on ImageNet) as teacher and DeiT as student, optimizing Kullback–Leibler divergence loss and the cross-entropy loss, adding distillation token as class token to the input sequences. The way class token passes to the network can be seen in Fig. 1.3.

1.1.2.2 TNT

Authors of the paper use the ViT model as the base to show that the granularity of patches is not enough to process rich image details and color information in a small scale. Thus, they propose a novel framework called Transformer-iN-Transformer (TNT) [12] that addresses this problem. The idea behind the method is the following: instead of only processing patches with Transformer layers, we can also use subpatches of each path to extract additional information. This way, they propose slicing each patch into subpatches, and process patches and all subpatches with 2 different Transformer blocks. Also, to keep locality information they provide learnable positional encodings for both patches and subpatches. After processing patch and its subpatches, the latter is then projected to the needed shape so that they can be added to each other element-wise. You can see the details in the Fig. 1.4. Those 2 transformers do not have the same parameters, so if the outer Transformer follows the ViT, the inner one is much less complex in terms of parameters and its computational costs are not significant, compared to the outer.

1.1.2.3 T2T-ViT

Authors of T2T-ViT [13] paper analysed the features extracted by ViT and found out that ViT is struggling to catch information contained between patches on edges — shown on Fig. 1.5, and moreover, it contains invalid feature maps.

As the solution, they propose soft split for image, progressive tokenization 1.6 with T2T process 1.7 and deep-narrow based backbone. The main idea of those methods is the following: similar to ViT, we take patches of image and flatten them, feed it to Transformer layer, but every time when we get the output from the Transformer layer, we do not pass it further, but reshape in a specific way to add more positional information.

1.1.2.4 ConViT

ConViT [14] tries to combine two biases: attention and convolution. Since on early stages of training it is beneficial to have more prior knowledge of

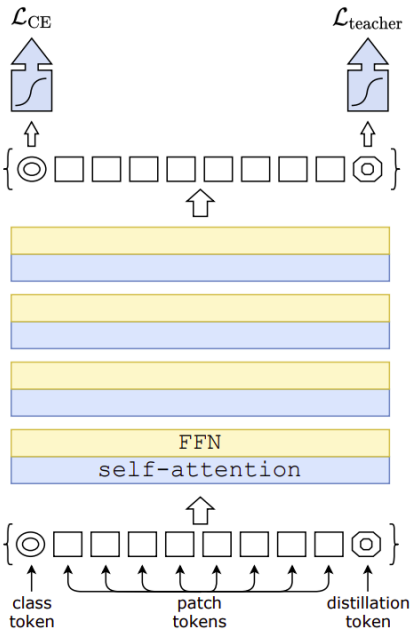

Figure 1.3 — DeiT architecture. Taken from original paper [11]

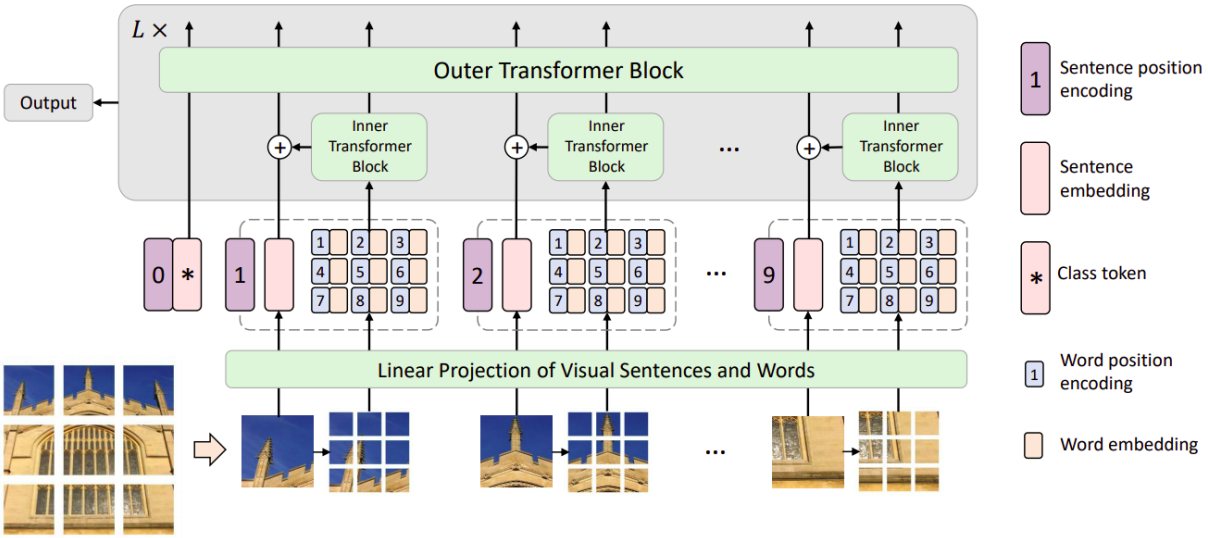

Figure 1.4 — TNT architecture. Taken from original paper [12]

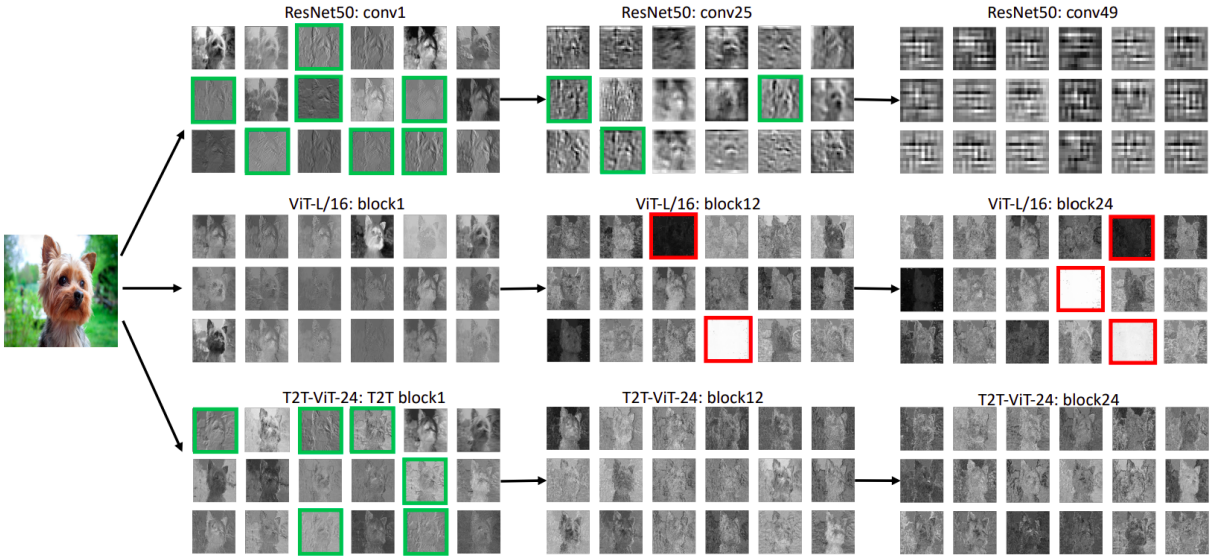

Figure 1.5 — Feature visualization of ResNet50, ViT-L/16 and proposed T2T-ViT-24 trained on ImageNet. Green boxes highlight learned low-level structure features such as edges and lines; red boxes highlight invalid feature maps with zero or too large values. Taken from original paper [13]

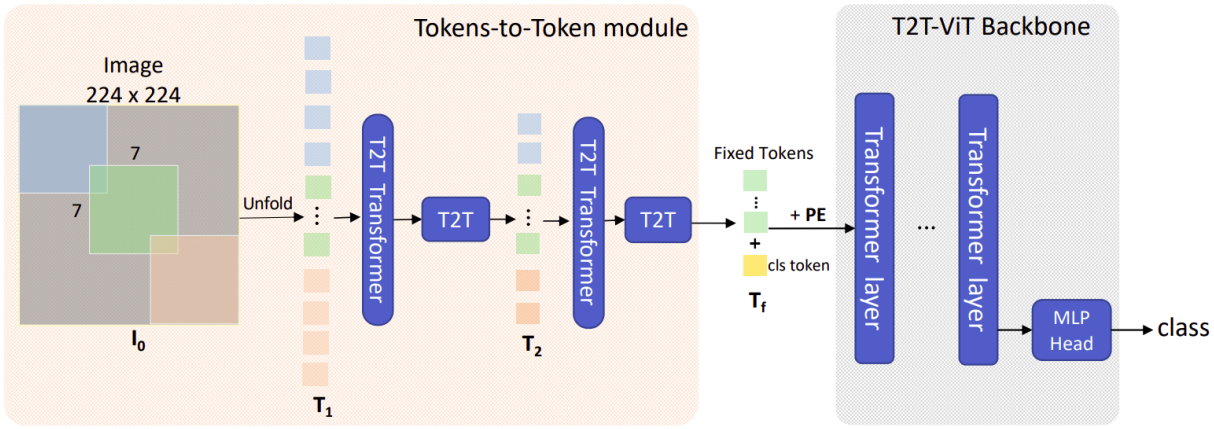

Figure 1.6 — T2T-ViT architecture. In the T2T module, the input image is first soft split as patches, and then unfolded as a sequence of tokens T_0. The length of tokens is reduced progressively in the T2T module (we use two iterations here and output T_f). Then the T2T-ViT backbone takes the fixed tokens as input and outputs the predictions. The two T2T blocks are the same as Fig. 1.7 and PE is Position Embedding. Taken from original paper [13]

the data, and later this knowledge adding constrains, authors propose self-attention with gating parameter weighting content and positional features shown in Fig. 1.8. Thus, network decides itself, what information to use.

1.1.2.5 DeepViT

Authors of DeepViT [15] paper show that attention maps over different layers have close similarity, e. g. no matter how deep the ViT is, it is not learning new features after a certain number of Transformer layers. To alleviate this problem, they introduce Re-attention block, which is a normalized version of attention over heads, it can be seen in Fig. 1.9. This move allows to extend number of effective layers from 16 to 24. Details can be found on Fig. 1.10.

1.1.2.6 Swin Transformer

Swin Transformer [16] tries to better extract fine-grained details and exchange information between image's patches, changing ViT architecture to extract features more hierarchically. As a novel feature, they introduce merging shifting windows with patches to reduce self-attention computations and to process information on the edges of patches as it is shown in Fig. 1.11.

1.1.2.7 CaiT

CaiT [17] model's authors try to improve ViT's ability to expand in depth by introducing two ideas: channel-wise weighting in SA and FFN and dividing Transformer layers in two parts: the first one is taking only image patches, and the last one using the class token also. This is called LayerScale and Class-Attention layers, they are shown in Fig. 1.12, 1.13 respectively.

1.1.2.8 DVT

In DVT [18] paper authors argue that using the fixed patch size in ViT based models can not be optimal for every processed image, so they propose Dynamic Vision Transformer — a framework which utilizes several ViT based models with different input patch size, starting from the easiest in terms of computations, to the most demanding, until the network will not output enough

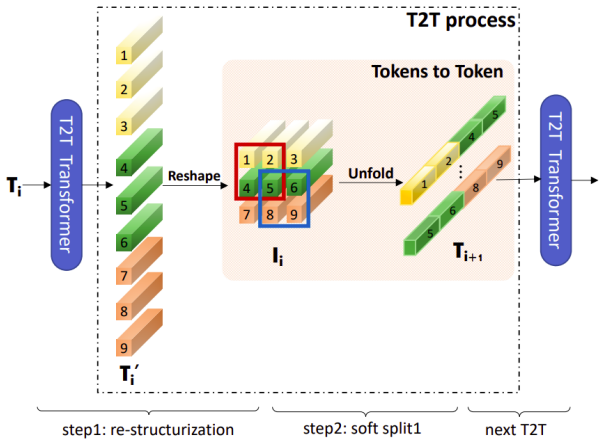

Figure 1.7 — Illustration of T2T process. The tokens T_i are restructurized as an image I_i after transformation and reshaping; then I_i is split with overlapping to tokens T_{i+1} again. Specifically, as shown in the pink panel, the four tokens (1,2,4,5) of the input I_i are concatenated to form one token in T_{i+1}. Taken from original paper [13]

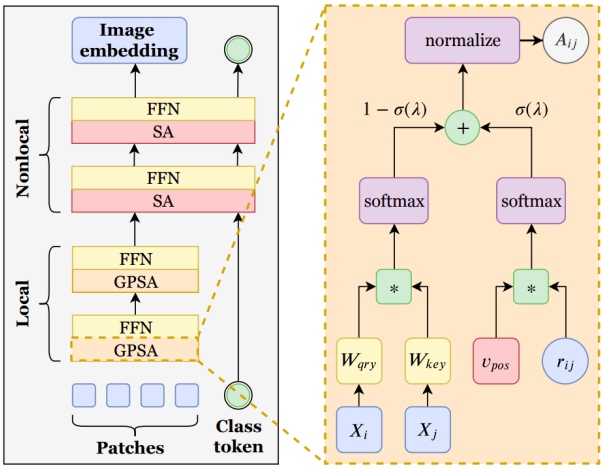

Figure 1.8 — ConViT architecture. Taken from original paper [14]

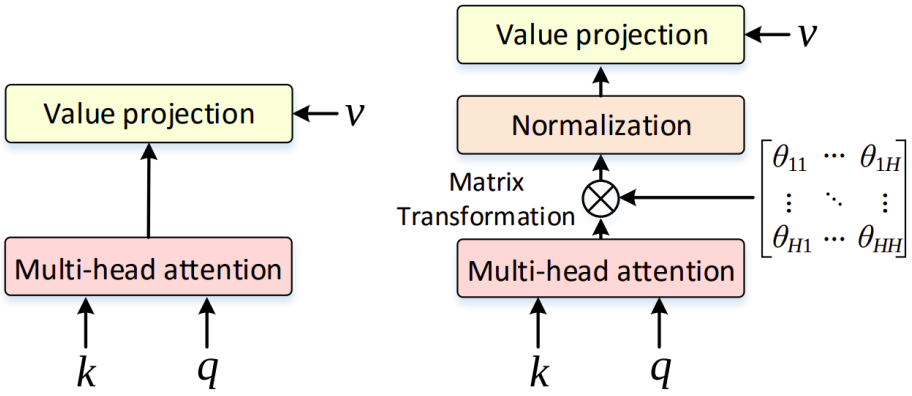

Figure 1.9 — (Left): The original self-attention mechanism; (Right): Proposed re-attention mechanism. Taken from original paper [15]

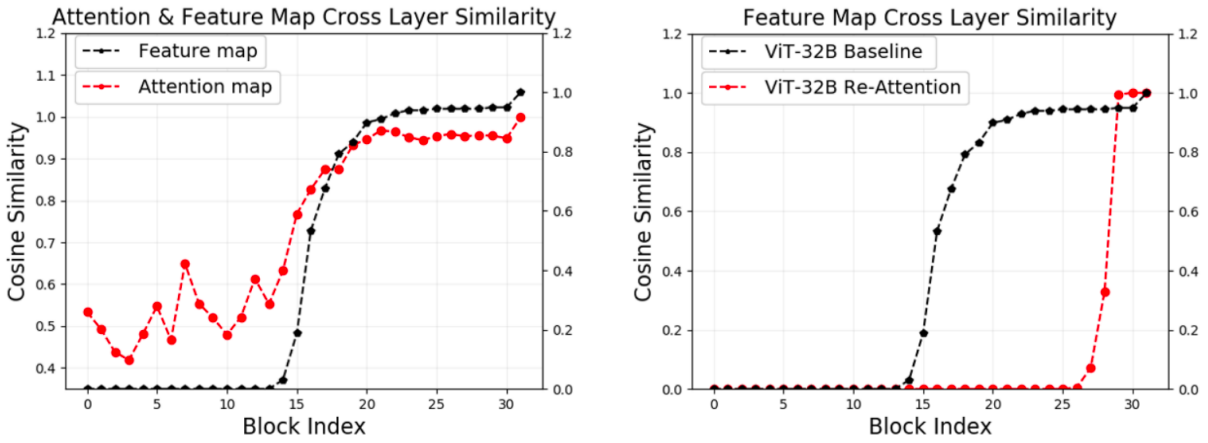

Figure 1.10 — (Left): Cross layer similarity of attention map and features for ViTs. The black dotted line shows the cosine similarity between feature maps of the last block and each of the previous blocks. The red dotted line shows the ratio of similar attention maps of adjacent blocks. (Right): Feature map cross layer cosine similarity for both the original ViT model and ours. Taken from original paper [15]

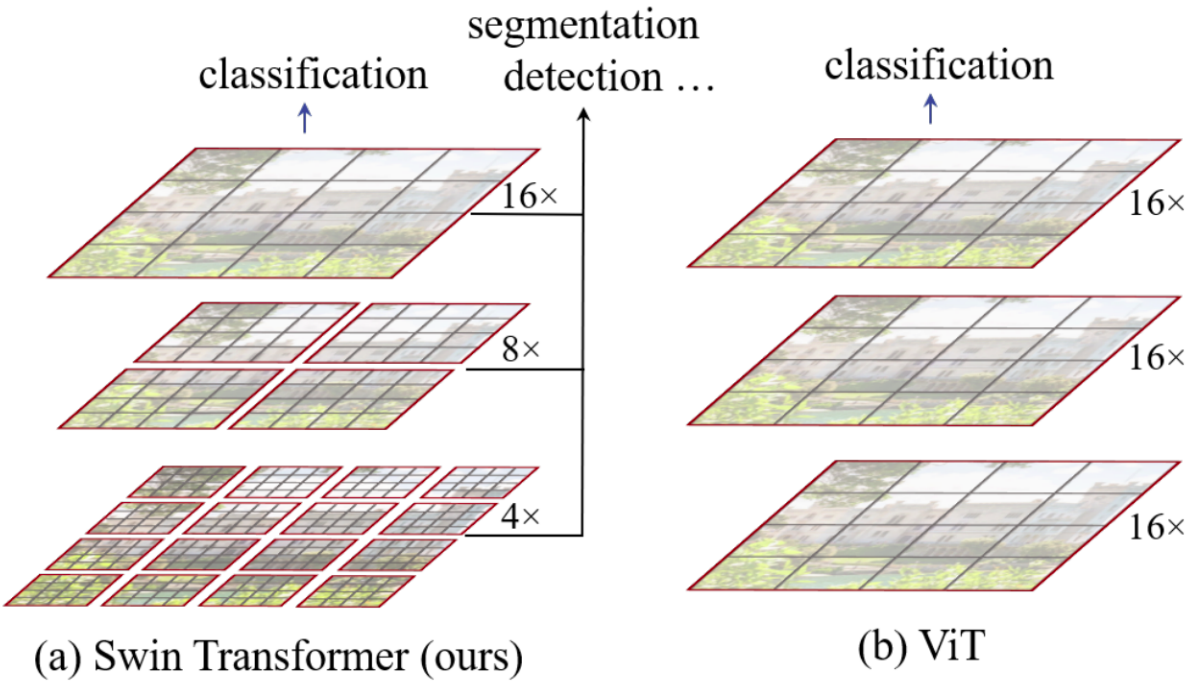

Figure 1.11 — (a) The proposed Swin Transformer builds hierarchical feature maps by merging image patches (shown in gray) in deeper layers and compute self-attention only within each local window (shown in red). (b) ViT architecture. Taken from original paper [16]

$$x'_l = x_l + \mathrm{diag}(\lambda_{l,1}, \ldots, \lambda_{l,d}) \times \mathrm{SA}(\eta(x_l))$$
$$x_{l+1} = x'_l + \mathrm{diag}(\lambda'_{l,1}, \ldots, \lambda'_{l,d}) \times \mathrm{FFN}(\eta(x'_l))$$

Figure 1.12 — LayerScale formulation. Taken from original paper [17]

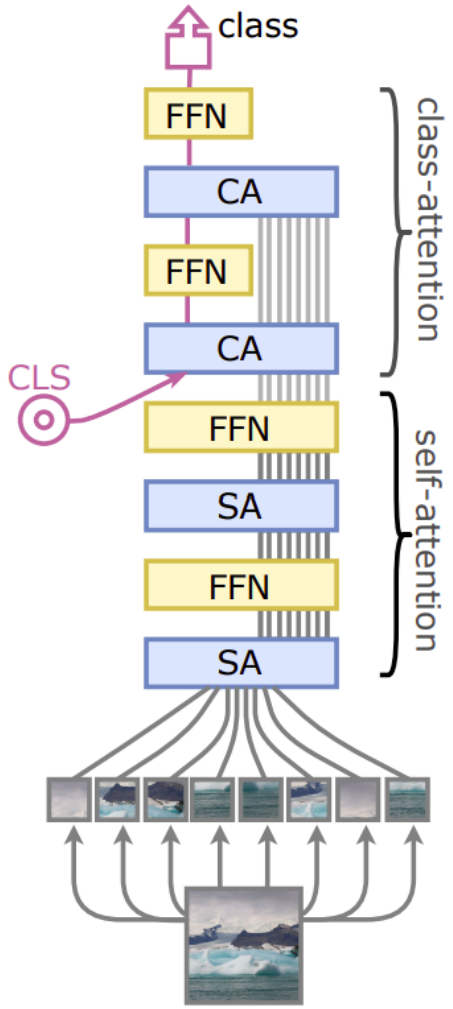

Figure 1.13 — CaiT architecture with class-attention. Taken from original paper [17]

confident prediction. Thus, if the object on the image is easily recognizable with high confidence, DVT uses only the first, the most lightweight model and terminates the inference process. The other option is to continue inference with a more demanding and accurate model and so on. Scheme of work can be seen in Fig. 1.14. DVT can be adapted to popular ViT based architectures, like DeiT and T2T-ViT.

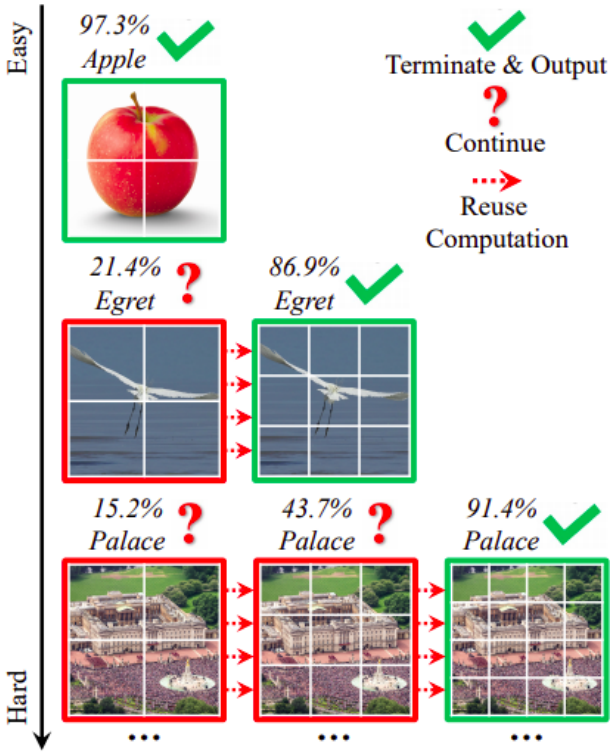

Figure 1.14 — DVT framework scheme. Taken from original paper [18]

If the situation where the image requires all inferences occurs, we spend significant time on processing just 1 image with several models in a row, which is unacceptable in terms of efficiency. To address this problem, authors propose two methods aiming on sharing knowledge and reusing computations from model to model: feature reuse and relationship reuse. The first one proposes to reuse output from the last Transformer layer and fuse it with all Transformer layers in the following model. The second way is to share self-attention maps from the previous model. Those methods allow us to reuse computations and ease the predictions.

1.1.2.9 Shuffle Transformer

Shuffle Transformer [19] is the architecture inspired by the ShuffleNet [29], but the idea authors follow is not the computation efficiency, as in [29], but the ability for model of sharing information between different patches. It is already a well-known problem, that ViT's way of independent patch processing does not allow sharing information between patches. To alleviate this problem, authors propose a method in which output tokens from a Transformer block are then rearranged so that their parts are passed as parts of other patches, more clearly it is depicted in Fig. 1.15. As it can be seen, Fig. 1.15 (a) shows vanilla patch processing pipeline, Fig. 1.15 (b) shows an intuitive way on how to do this shuffle, while Fig. 1.15 (c) presents an introduced method which is done using the "spatial shuffle - SA - spatial alignment" chain, where the spatial alignment step is done to reverse the shuffling. This way, authors show the way on how cross-window connections can be constructed.

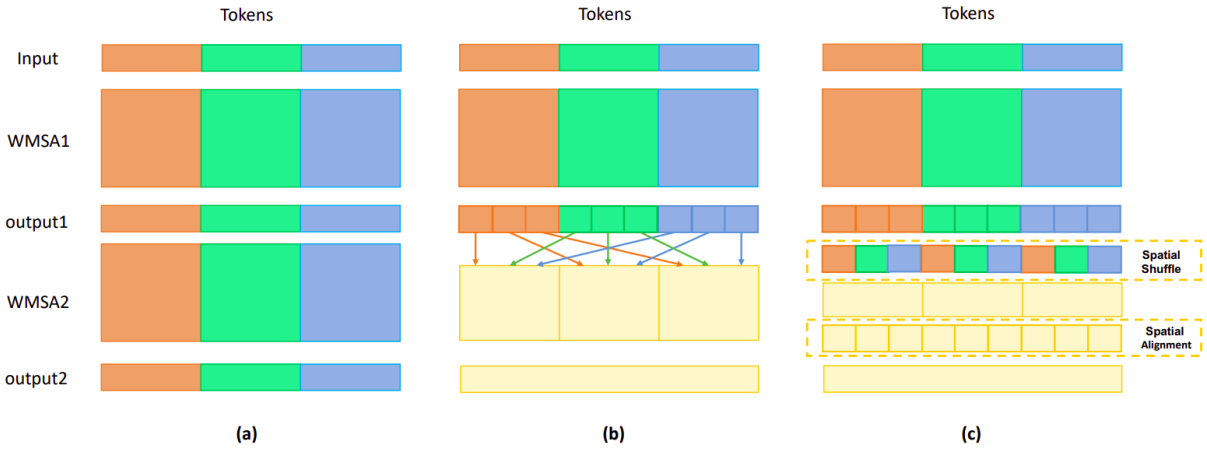

Figure 1.15 — Spatial shuffle method visualization. Taken from original paper [19]

1.1.2.10 CoAtNet

CoAtNet [20] authors systematically analyze options of combining convolution and self-attention blocks and came up with the formulation of new block and the way of stacking layers that is optimal in terms of generalization and capacity.

Authors came up with idea of combining 3 properties: translation equivariance from convolutions and input-adaptive weighting and global receptive field from self-attention. Thus, they proposed MBConv block, formulation is presented in Fig. 1.16. Overall architecture is shown in Fig. 1.17.

$$y_i^{\text{post}} = \sum_{j \in \mathcal{G}} \left(\frac{\exp\left(x_i^\top x_j\right)}{\sum_{k \in \mathcal{G}} \exp\left(x_i^\top x_k\right)} + w_{i-j} \right) x_j$$

Figure 1.16 — MBConv block formulation. Taken from original paper [20]

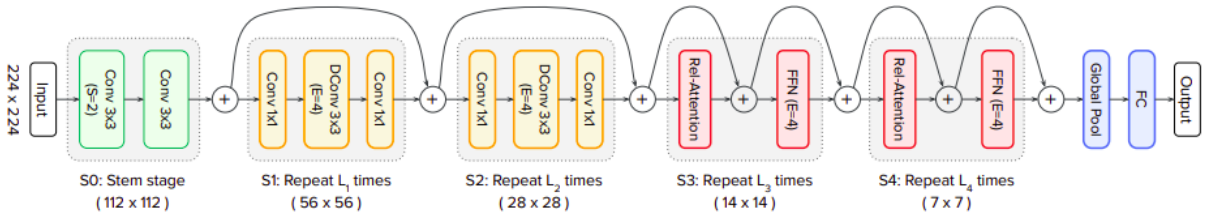

Figure 1.17 — CoAtNet architecture. Taken from original paper [20]

1.1.2.11 UViT

Authors of UViT [21] paper analyze different ViT based models and investigate the reasonability of the usage of different methods that were inspired by CNNs. Among them there are spatial downsampling, multi-scale pyramid features and double channels. After careful study on the benefits of such designs, they came up with the vanilla ViT architecture, but with different patch size, attention window size and Transformer block parameters. Thus, they show that ideas adopted from CNNs do not necessarily give impact when applying to ViTs for dense prediction. The main insights authors discovered are: using global attention in all Transformer blocks is not the recipe to get the best result, because having huge computational costs, it provides almost the same quality of predictions as the window of half scale; earlier attention blocks can use even 1/4 window scale, while deeper ones require the whole picture; depth of 18 block is the most optimal with the image size of 896x896 and different width for different model sizes.

The final architectures they present are UViT and UViT+, the main difference between them is that UViT has constant size attention window of 1/2 among all 18 blocks, while UViT+ has 1/4 in the first 14 blocks, 1/2 in the next 2 following ones, and global attention in the last 2, details can be seen in Fig. 1.18.

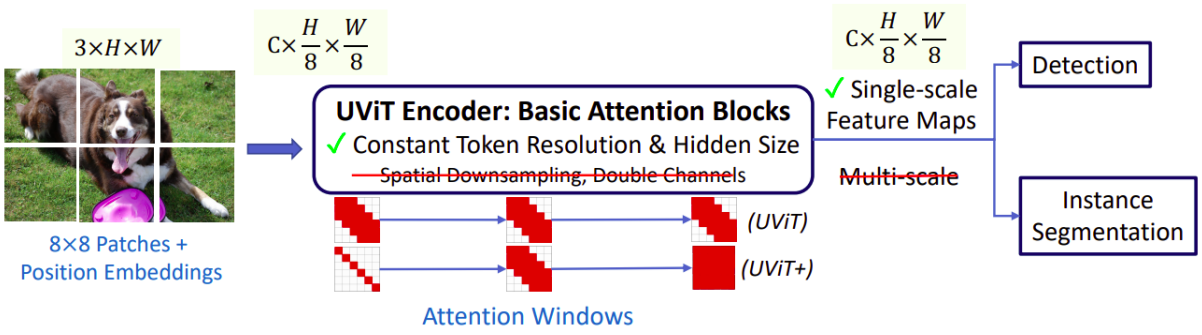

Figure 1.18 — UViT and UViT+ architectures. Taken from original paper [21]

1.1.2.12 Other methods

While we reviewed number of papers focused on application of Transformers for computer vision recognition task, there are a lot of works that try to combine benefits from Convolution networks and Transformer networks, but we have not presented any paper which uses convolution blocks explicitly. Nevertheless, we can not fail to mention such works as PVT [30], CvT [31], LeViT [32], ConTNet [33], Twins [34], XCiT [35] and MaskFormer [36].

1.1.3 Vision Transformer for dense prediction

In Deep Learning it is preferable to unite tasks where we get dense output for the given inputs under the name of the dense prediction task. Moreover, frequently researches in their works classify detection task as dense, so will we.

Just to recall what is detection and segmentation, we have to say that the detection task is the task of predicting the location and borders of the object, while during segmentation the desired result is the pixel-wise classification. You can see the visualization in Fig 1.19

When we think about the segmentation problem, UNet architecture [37] immediately appears in mind. The model, which encodes an input matrix into

vector embedding, and then decodes it into the same dimensions back, became the standard and the take-off architecture for the future state-of-the-art solutions. That architecture aggregates features from low-level to high-level with the usage of convolution blocks, but what about some ViT-based models? If we look into vanilla UNet, it produces 1D vector, which is not supposed to be reshaped, but to be deconvolved into initial image's shape. On the other hand is ViT, and the output of each Transformer layer is 2D matrix: $output \in \mathbb{R}^{N \times (P^2 \times C)}$. As the simplest idea, we can apply layers aggregation operation and linear layer to map output of Transformer layer to 1D embedding and use any CNN decoder to decode it into class maps. Another one simple idea, but without off-the-shelf backbones, is the following: the first component can be transformed into compenent of class channels, and the second component can be reshaped into small image and upsampled to initial image's size. Those ways, we can use ViT model for segmentation.

As for above-mentioned works in Section 1.1.2, Swin Transformer, Shuffle Transformer and UViT authors provided results on semantic segmentation problem with ADE20K dataset [38, 39].

Nevertheless, there also are papers that focuses on dense prediction among which we have selected: DETR [22], SETR [23], SegFormer [24], Segmenter [25], FTN [26] and Lawin Transformer [27].

1.1.3.1 DETR

The authors of this article aim to change the approach to object detection pipeline and introduce a new architecture called DETR (DEtection TRansformer)

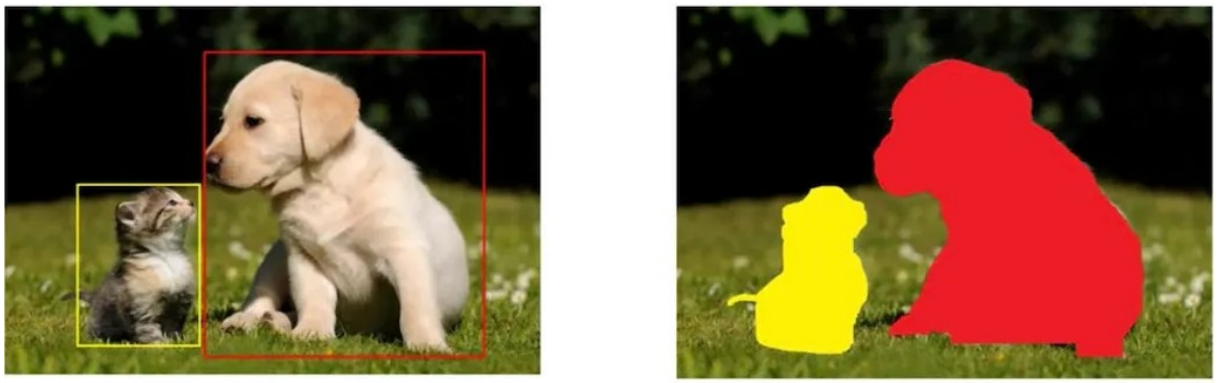

Figure 1.19 — Detection is on the left side, segmentation is on the right

[22]. The standard pipeline consists of applying the CNN model, obtaining the bouncing boxes, further filtering and deduplication, and then outputting the prediction. DETR, on the other hand, is an end-to-end model that doesn't need intermediate steps and third-party libraries. It's design can be seen in Fig. 1.20

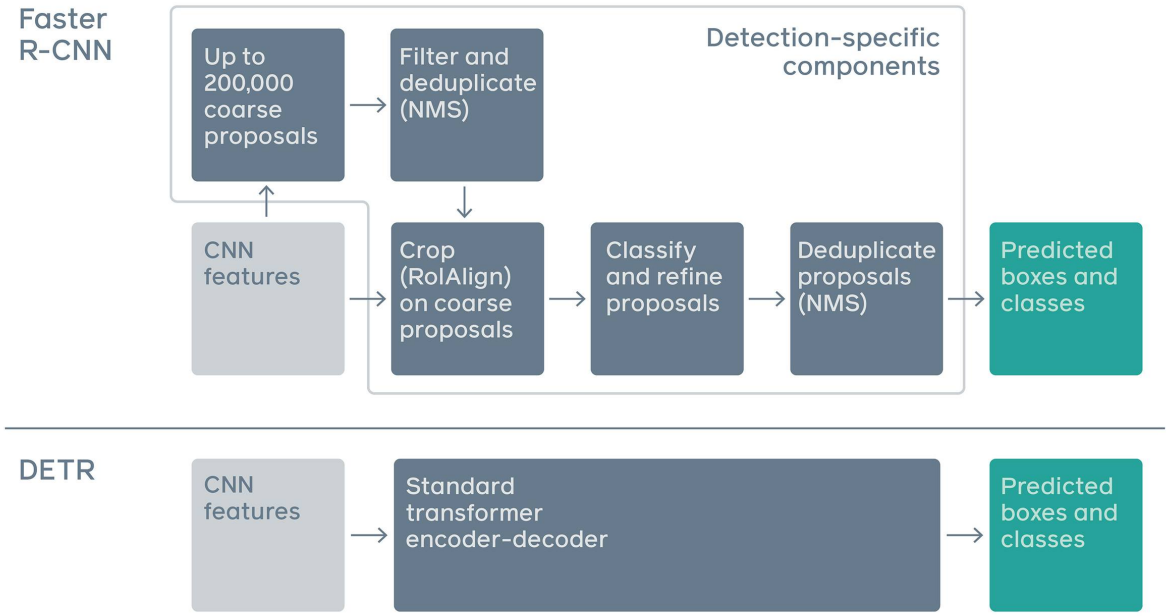

Figure 1.20 — DETR design. Taken from the blog post [40] from authors

Authors propose two main components that do the job:

a) Bipartite matching loss, which uses correspondence between the prediction and the GT.

b) End-to-end architecture predicting sets of bouncing boxes and their classes.

DETR generally outputs prediction for N objects, where N is larger than the possible number of the objects in the image. Next step is in finding matches between the predictions and the GT to start calculating losses. To find a bipartite correspondence between the predictions and the GT, algorithm look for a permutation of N predictions such that they minimize the match loss function between GT and prediction. This permutation is found optimally using the Hungarian algorithm. Next, the combined L1 and IoU loss is used, so as not to encounter the problem of incorrect processing of large and small boxes.

The architecture itself consists of four parts: CNN backbone, Transformer encoder, decoder and prediction heads, as it can be seen in Fig. 1.21.

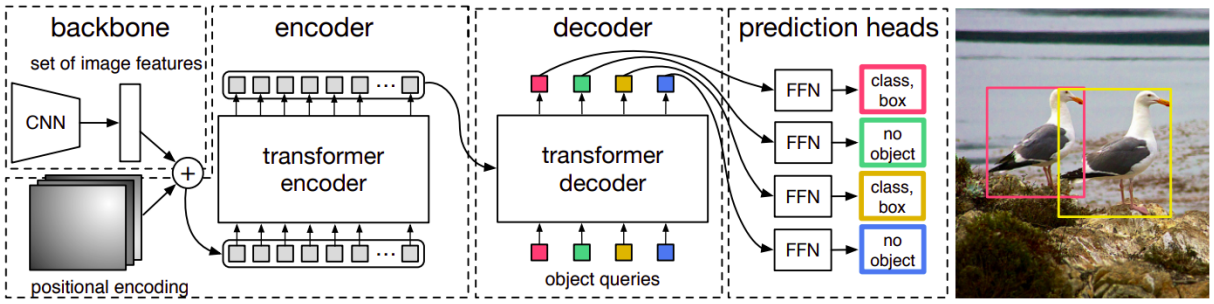

Figure 1.21 — DETR architecture. Taken from original paper [22]

Image is fed into CNN, the output of the CNN is feature maps which then are passed through 1x1 convolutions to reduce its depth. The resulting feature maps are then flattened, projected and passed to the vanilla Transformer encoder with the fixed positional encoding as sequences. After the decoding, resulting embeddings are then fed into FFN to get final predictions. As a detail, all FFNs share weights and output normalized coordinates of the center of the object and its width and height.

Interesting addition is that authors propose to use this design for panoptic segmentation [1], the scheme can be seen in Fig. 1.22.

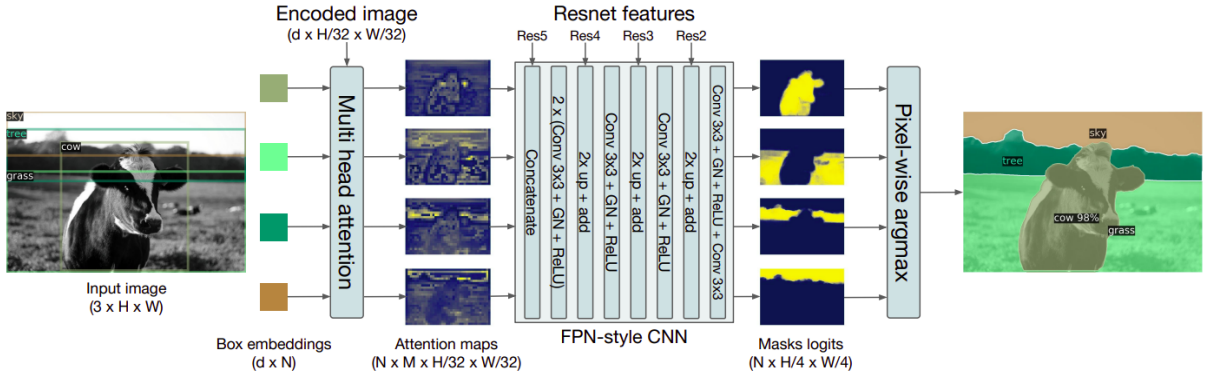

Figure 1.22 — DETR architecture for panoptic segmentation. Taken from original paper [22]

To do this, they propose to change the fourth part of the network from FFNs to the visualization part. This visualization part takes output of the

[1]Panoptic segmentation is the task combining both semantic and instance segmentation. During this process we do semantic segmentation and instance segmentation within each semantic class.

decoder, pass through MHSA block, get the low resolution heatmap and increase the resolution with FPN-like network to get segmentation masks.

1.1.3.2 SETR

SETR (SEgmentation TRansformer) [23] proposes ViT-based architecture for semantic segmentation. Authors propose to use vanilla ViT as encoder and further usage of 2D embeddings from Transformer layers and 3 different variants of backbone. First variant is to naively reshape and bilinearly upsample output from the last Transformer layer Z^L to desired resolution. Second option called PUP (b) is to do it in a more complex way, switching from bilinear upsampling to progressive upsampling using convolutions. Third way called MLA is the most promising, because it is using not the last output but outputs from different layers in a spirit of a feature pyramid network [41], e.g. $\{Z^{\frac{L}{4}}, Z^{\frac{2L}{4}}, Z^{\frac{3L}{4}}, Z^L\}$. Then they combine information from different layers with top-down aggregation, including element-wise addition followed by convolution. After aggregation, 4× bilinear upsampling is applied to get the final resolution. Visualization of variant can be found in Fig. 1.23.

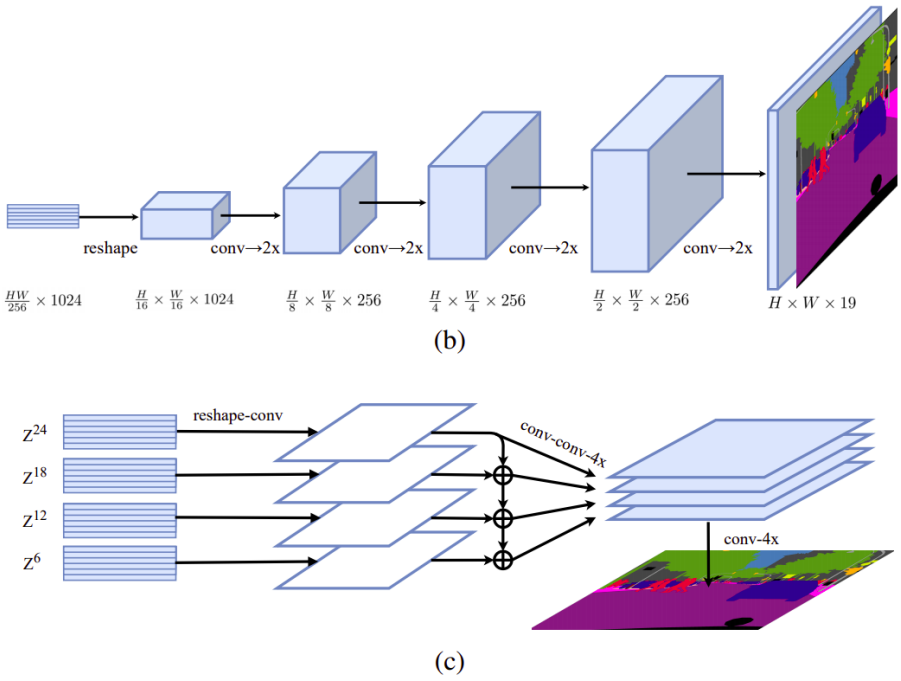

Figure 1.23 — SETR backbones: (b) Progressive UPsampling and (c) Multi-Level feature Aggregation. Taken from original paper [23]

1.1.3.3 SegFormer

SegFormer [24] model proposes completely different approach with carefully designed encoder and decoder shown in Fig. 1.24. Authors propose a hierarchical transformer encoder that outputs multi-scale features, called MiT; important thing is that it has not positional encoding, which can bring difficulties when applying model to resolution that differs from the one used during training phase. As a decoder it uses a MLP that aggregates the multi-scale features whick then are 4× bilinear upsampled to get the final class maps. Major difference from ViT is the patch size — here, image is being divided into overlapping patches of size 4×4 to provide more low-resolution fine information. Also authors use efficient version of self-attention, using sequence reduction process from [30]. Lightweight All-MLP decoder, as they call it, processes multi-scale features in 4 steps: unify channel dimensions, upsample and concatenate, fuse multi-scale features, predict class feature maps.

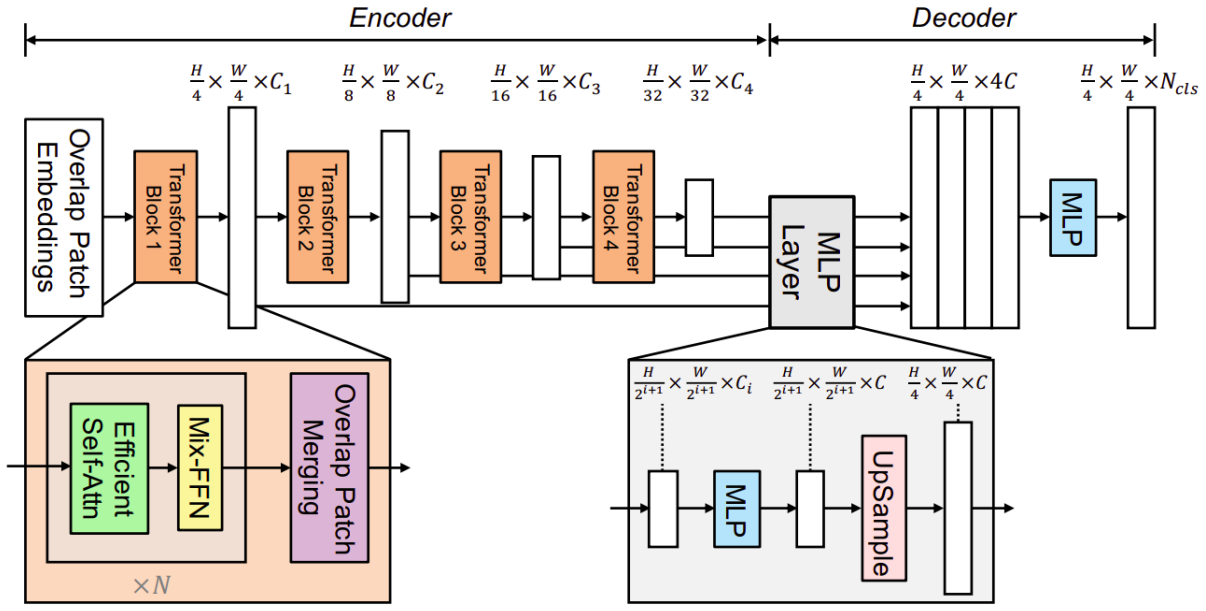

Figure 1.24 — SegFormer architecture. Taken from original paper [24]

1.1.3.4 Segmenter

Authors of Segmenter [25] paper faced the issue of constructing class maps from 2D embeddings, described in Section 1.1.3 and tried to use rich information of ViT-based encoder, which extract knowledge using global con-

text of the image and combine it with decoder free of convolutions, using pure Transformer-like architecture. As the result, they came up with Segmenter. Segmenter is a Transformed-based encoder-decoder model for semantic segmentation, which fully utilizes Vision Transformer idea in both extraction of features and generating pixel-level class maps.

Segmenter's encoder completely follows ViT architecture, but is not using token class and classification head, so as an output of the last Transfromer layer, we have $output \in \mathbb{R}^{N \times D}$, where $N = \frac{H \times W}{P^2}$ and $D = P^2 \times C$ or chosen manually for different size model versions. Also, to extract positional information, the learnable positional embedding are being added to patch embeddings before feeding to Transformer encoder which consists of L Transformer layers.

After obtaining 2D embeddings from encoder, we have to generate pixel-level class maps. As the solution, they proposed simple linear projection, also described in Section 1.1.3, and more advanced Mask Transformer. Mask Transformer is a decoder which takes as input 2D embeddings from the output of the encoder and learnable class embeddings, where every embedding assigned randomly and assigned to a single semantic class. Inside it is again Transformer encoder consisted of M layers, which processes patch tokens jointly with class tokens. After that authors decided to apply scalar product of patch and class tokens, reshape into 2D masks and and bilinearly upsample them to the original size. Details of the architecture can be found in Fig. 1.25.

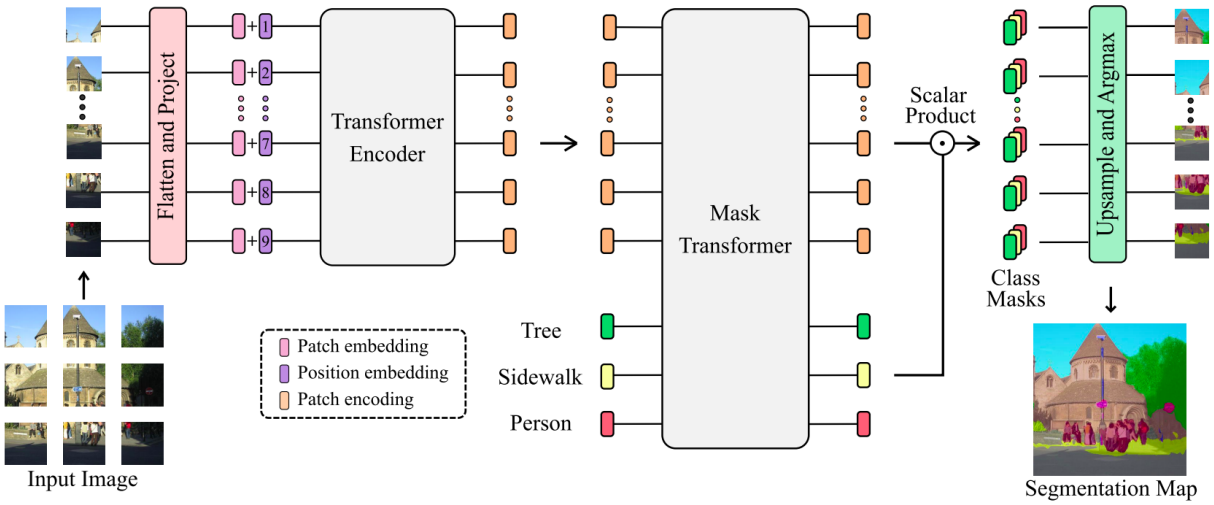

Figure 1.25 — Segmenter architecture. Taken from original paper [25]

1.1.3.5 FTN

Authors of Fully Transformer Network (FTN) [26] present new architecture with Transformer-based both encoder and decoder following feature pyramid network [41] idea. FTN consists of Pyramid Group Transformer (PGT) as encoder and Feature Pyramid Transformer (FPT) as decoder. PGT takes 4x4 patches as input, processing it with N_i PGT blocks, then on the next stage transforming output in a way to reduce resolution and increase the depth, and conducting processing with mentioned PGT blocks, as shown in Fig. 1.26. This way, the encoder produces features of different scales. It is important to mention that on each stage we have a different number of PGT blocks, this number is controlled by the size of the network. As for the decoder, it utilizes outputs from the 3 last stages and fuses different-scale information to high resolution with hierarchical upsampling and addition, processing and combination with transforming to match all the classes. The processing blocks are spatial reduction Transformer blocks from [30], which is used to reduce memory and computation cost by spatially reducing the number of key and value tokens. Then, output from the decoder is $4\times$ bilinearly upsampled to the initial resolution. Also, architecture uses conditional position encoding from [42].

1.1.3.6 Lawin Transformer

Authors of the Lawin Transformer [27] paper, inspired by dilated convolutions [43], ASPP [44, 45] and PPN [46], came up with solution on how to adapt those ideas from CNN world to the attention reality to use multi-scale representations. Lawin Transformer consists of the MiT encoder from SegFormer [24] and decoder LawinASPP. MiT encoder with 4 Transformer blocks produces features of different spatial sizes, the last three of which are then combined to be fed into LawinASPP. It consists of 5 parallel streams, including skip-connection, pooling and large window attention (Lawin) with different number of heads and ratio of the size of context patch to query patch. After obtaining all 5 stream results, they are passed through MLP, upsampled and fused with features from the linearly projected first stage output of MiT. On the last step, the result is

mapped with MLP to restore the number of channels and bilinearly upsampled to initial resolution. Clear visualization of this can be found in Fig. 1.27.

MLP-Mixer [47] is a model which consists of fully-connected layers, activation functions, poolings and normalization layers; it lacks both convolutions and attention mechanisms, and thereby is free of any inductive bias. Large window attention, the main processing block of the network, is inspired by MLP-Mixer and can be seen in Fig. 1.28.

To control the context patch size at which attention can "look", additional parameter R is passed to block — it is the ratio of the context patch size to the query patch size — and in a sense, it controls how bigger the context surrounding patch is. As a result, it leads to increasing complexity in attention computation, which is solved with pooling to initial query size. As the problem of such pooling, authors see the loss of information about query and context relationship. To solve this, they propose to have exactly R^2 heads of this attention mechanism, having the fact that the number of heads has no impact on the computational complexity. After pooling, context is reshaped, passed through a set of head-specific position-mixing MLP layers and mapped in adapted attention formulation in Fig. 1.29.

1.1.4 Vision Transformer insights

Aside of the attempts to change the design of the ViT, there are attempts to understand on how to work with it more efficiently.

1.1.4.1 Second order optimization

Authors of the paper "When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations" [48] asks if there are differences in the inner processes of the ViT and ResNet training to understand whether it can be exploited or not.

Researches analyse the loss landscape and find out that ViT converge in the much more sharp local minima than ResNet have. Thus, it is proposed to change the optimization algorithm to behave not only using the knowledge about

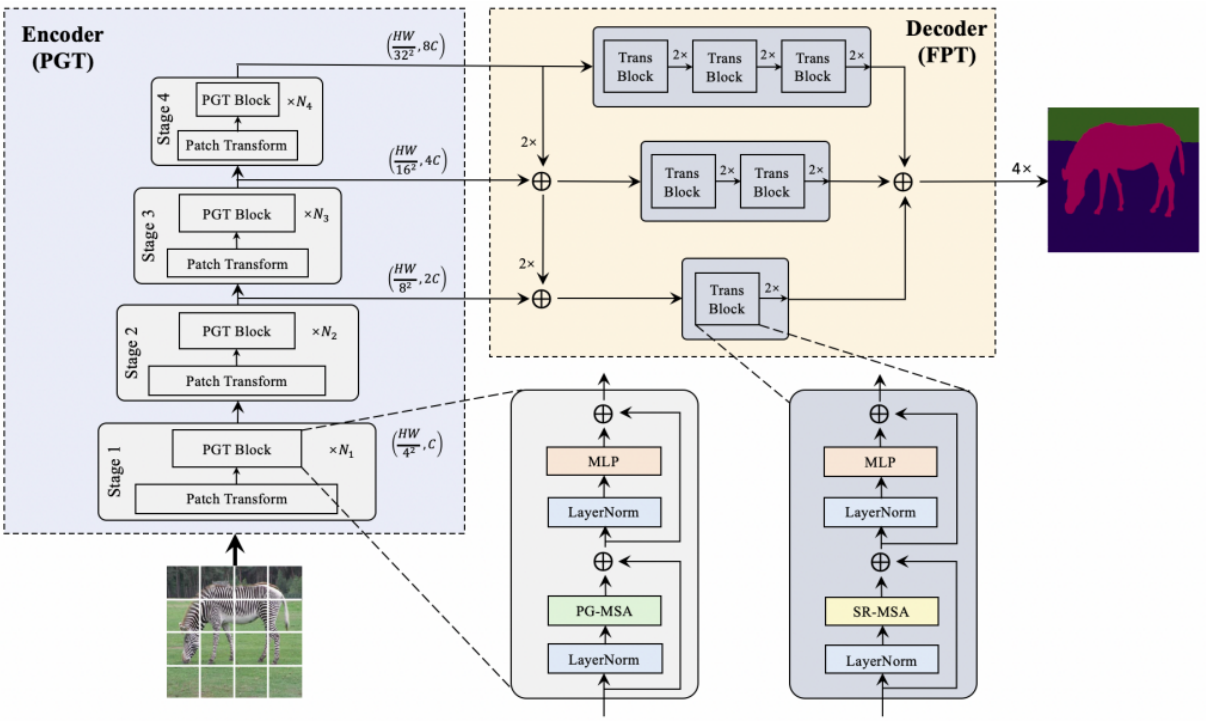

Figure 1.26 — Fully Transformer Network architecture. Taken from original paper [26]

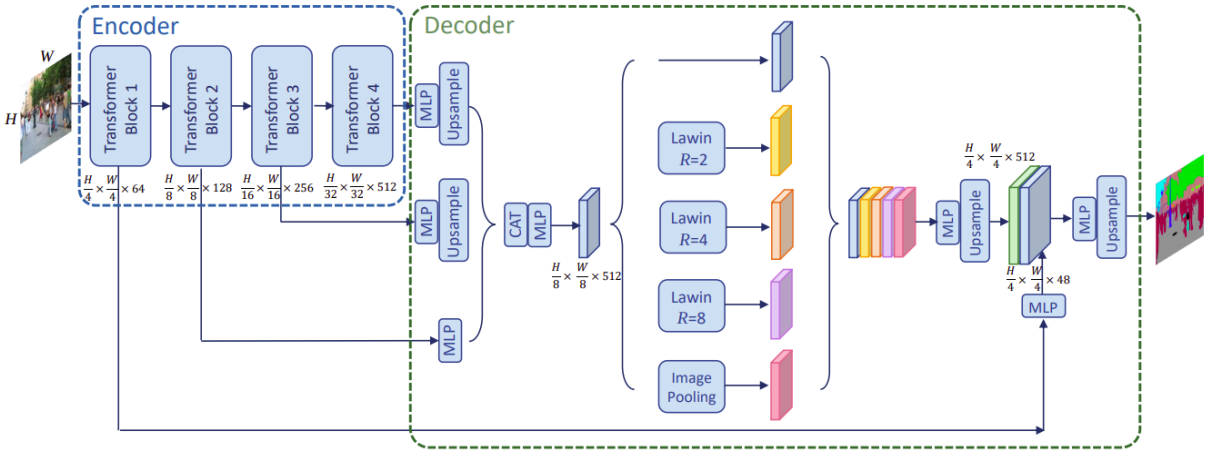

Figure 1.27 — Fully Transformer Network architecture. Taken from original paper [27]

the local loss gradient, but also about the smoothness of the loss surface around, i.e. use second-order optimizers.

As the result of this study, they took SAM (Sharpness-Aware Minimization) [49] optimization algorithm and without any changes of the architecture or training pipeline, achieve a 5.4 points higher absolute top-1 accuracy, moving from 74.6 to 79.9. However, main drawback of this is the twice lower training speed because of two backpropagations on every iteration.

1.1.4.2 Train and fine-tune recipes

In the "How to train your ViT?" [4] authors study the ways on getting better results without any modifications of the architecture of ViT itself but with the help of the proven training techniques. Those training techniques include data augmentation, regularization and precise tuning of the hyper parameters.

Key findings for regularization:

a) Use dropout and stochastic depth regularization [50] which in a sense is a dropout for layers — it deactivates processing blocks depending on the depth of the layer.

b) Use combination of RandAugment [51] and Mixup [52] techniques as augmentation.

c) Use and vary weight decay w.r.t. volume of augmentations.

Key findings for (pre-)training:

a) Use Adam [53] optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

b) Use cosine lr scheduler with linear warmup and initial lr equal 1e-3.

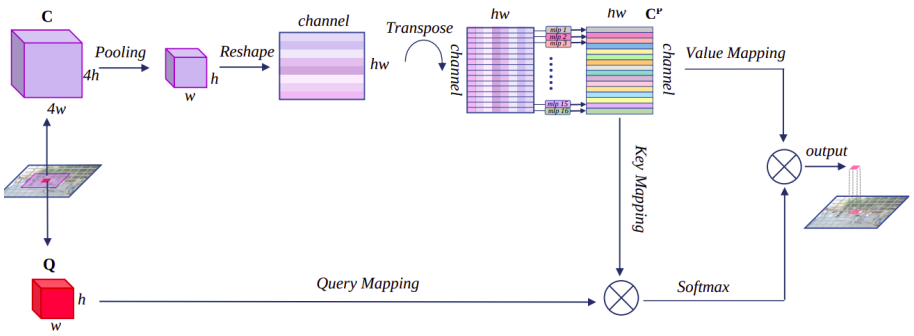

Figure 1.28 — Large window attention scheme. Taken from original paper [27]

c) Use gradient clipping of global norm 1.

Key findings for fine-tuning:

a) Use SGD optimizer with momentum equal 0.9.
b) Use cosine lr scheduler with linear warmup and initial lr equal 1e-3.
c) Use gradient clipping of global norm 1.

General recipes are the following: it is better to use pre-trained version of ViT and fine-tune it on downstream task than train it from scratch; between more complex augmentation and regularization choose the first; hard augmentation can achieve similar results as big datasets.

1.2 Self-Supervised Learning

When we speak about Machine Learning, we speak also about the data, since the key idea of regular ML tasks is to mimic the way of how human think, and to do so, we have to train some model to do so. One of the major options here is to use what is called Supervised Learning (SL), when we have training dataset of some objects and the classes corresponding to them, so we can make mapping $f(x) = y$, where f is function analysing the object x and y is the class label. This function can be either human or ML model and in general, it has to be strict mapping without errors.

But in real world scenarios we do not always have the labeled data, for example we can not label the required amount of the data or we do not have it at all and can not get. Furthermore, in some cases it is not trivial to define what is label itself.

Thus, when we speak about such a circumstances, we should exploit the technique called Self-Supervised Learning (SSL). It is a method of giving the model information about the data, giving the understanding the domain and

$$A = \text{softmax}\left(\frac{(\mathbf{W}_{\text{q}}\mathbf{Q}_{\text{h}})(\mathbf{W}_{\text{k}}\mathbf{C}_{\text{h}}^{\text{P}})^{\text{T}}}{\sqrt{D_h}}\right)(\mathbf{W}_{\text{v}}\mathbf{C}_{\text{h}}^{\text{P}})$$

Figure 1.29 — Large window attention formulation. Taken from original paper [27]

helping to build the relationship between classes and objects, to get to know the nature of the data itself. Without any labels.

After the Self-Supervised Learning process is done, we can use the trained model, which now understands the nature of things to use it for your specific task. For example, your goal is to train model which would do both classification and segmentation, you can take any backbone you want, for example ResNet without classification head and apply Self-Supervised Learning on the dataset of your domain without labels. Then you can use this pre-trained backbone in the freezed state to fine-tune only classification or segmentation head. Additionally, you can use it to further train Image Retrieval or Inpainting or anything you want on this base. Moreover, since your pre-trained base model understands the domain, you can use much less volume of labeled dataset to fine-tune it on the downstream task.

The task of train model to understand the data is often calls Representation Learning. One simple example of this is to train the model to understand the degree of the rotation of the original image, as it was done in [54]. In [55] authors force model to predict context using image patches, the main idea in the work is to predict relative positioning of two patches, and if it is done, then we conclude that model can understand relationship between objects. Common and more advanced example is to train model to solve jigsaw from patches [56] and even more, solve jigsaw in black and white that lacks patch, recolor it and reconstruct the lacking patch [57]. Also there are similar ideas of reconstructing the color channels [58] or inpainting the missing regions of the image [59].

Another important topic which we face in this area is Knowledge Distillation. Knowledge Distillation is a learning paradigm helping to accelerate convergence to optimum, but this optimum is not the one that best fits the GT labels, but one that best fits prediction of another, usually, complex and big model or ensemble of such a models. This mapping to the big model's output allows the model to learn not from the exceptionally correct labels, but from the learned, *a priori* knowledge of the data domain.

Now we will show the recent and proven approaches in this area.

1.2.1 SimCLR

SimCLR (Simple Contrastive Learning Representation) [60] paper delivers the very natural and logically understandable idea on how to make model understand the data. This idea uses representation learning approach and we train the model to output embedding or latent vector or representation so that the obtained latent space would be semantically correct, i.e. vectors of similar objects would be close to each other, while the different would oppose.

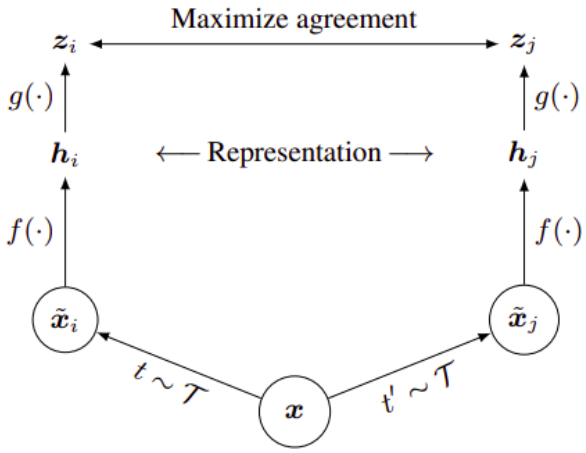

Figure 1.30 — SimCLR design. Taken from original paper [60]

In this work we operate with positive pairs of images and negative. First one is the pair of differently augmented version of selected image from the batch, the latter is the augmented version of the selected image from the batch and augmented version of another image from the batch. Thus, if we have batch of N images and two augmented versions of each one, on each iteration we extract the latent vector of $2N$ images. Then for each image in batch we measure the distance between positive pair and the distance between all possible negative pairs $(2N - 1)$ and use it to calculate and minimize the contrastive loss.

Thus, we optimize the latent space through the maximization of the agreement between the augmented versions of the selected image and minimization of the agreement of the different images. Design can be found in Fig. 1.30.

1.2.2 BYOL

BYOL (Boost Your Own Latent) [61] authors propose to use two identical networks in the student-teacher scheme without the usage of negative pairs opposed to the SimCLR [60] and MoCo [62] methods.

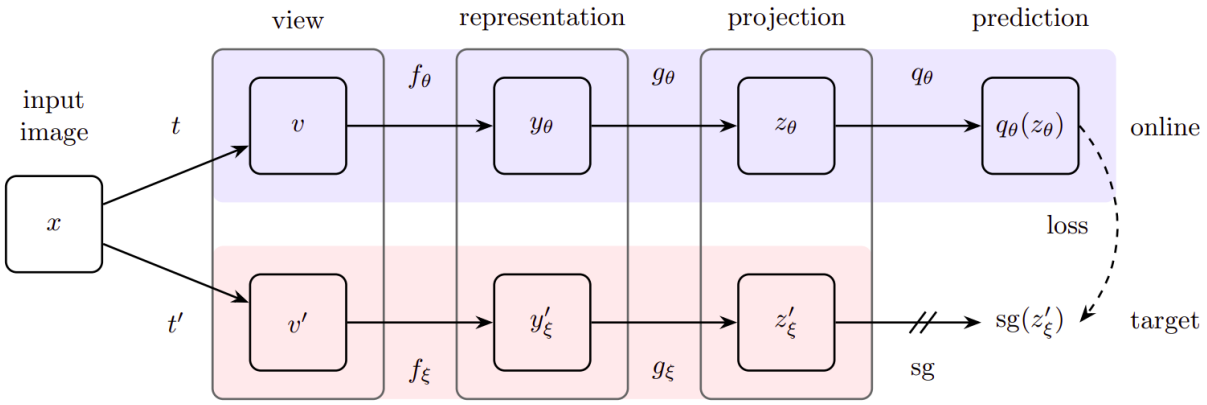

Figure 1.31 — BYOL design. Taken from original paper [61]

In this pipeline authors have two network, one of them is called online network or student and target network or teacher, both share the same architecture and in the training phase, we apply backpropagation [1] only to student network, while teacher is the exponentially moving average (EMA) of the student. After that, everything but student weights are discarded.

During the forward pass we apply augmentation to the input image and have 2 different views, one of which is passed to the student, and another one to the teacher. After obtaining representation vectors, they are projected in the higher dimension latent space and then student's projected representation is received by MLP predictor, which makes the method asymmetric in terms of student-teacher relations. After that, student's normalized prediction and teacher's representation projections are used to calculate the MSE loss as it is shown in Fig. 1.31.

Important analysis was done on the possibility of the training collapse — situation in which networks will output constant output on different inputs. Since this problem is possible in cases when only positive pairs are used, researchers have shown that the convergence to such a loss is permitted by the design of networks and pipeline.

[1] Backpropagation algorithm, mentioned in glossary.

1.2.3 SwAV

SwAV (Swapping Assignments between multiple Views) [63] framework delivers solution without direct comparison of representation vectors but instead proposes prototypes codes and solving assignment problem, it's scheme can be seen in Fig. 1.32.

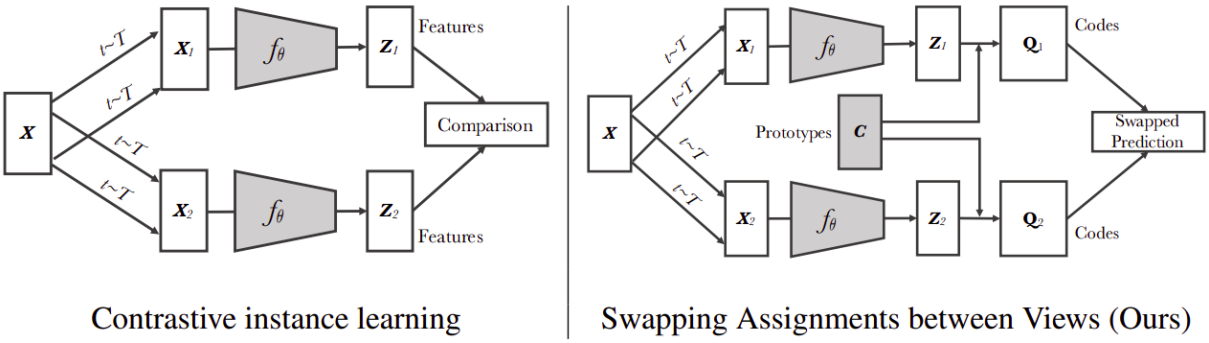

Figure 1.32 — SwAV design. Taken from original paper [63]

As for the prototype codes, for optimization authors propose to predict the prototype vector of another view.

Introduced novelty here is called a multi-crop views. Original approach of constructing positive pairs of images is to take selected image and apply different augmentations to it, while in the proposed method besides those two versions of initial images we create N more images of the lower resolution and use them in the pipeline also. Typically, resolution of two global (initial) views is 224x224, while for additional small views it is 96x96. Moreover, we apply different crop scales to global and additional views to meet the domain's regular sizes of the objects.

1.2.4 SiT

Authors of the SiT (Self-supervised vision Transformers) [64] paper aim directly at Self-Supervised Learning for the ViT and small datasets. As the SSL objective they use three components: rotation loss, discussed earlier in this Section, reconstruction loss and contrastive loss, weighting them using introduced learnable parameters [65].

Vanilla ViT architecture is modified to have rotation head, contrastive head and linear projection so the encoded patch embeddings to be turned into

reconstructed image patches back, also there are 2 additional token embeddings instead of class token: rotation and contrastive. When the image is passed to the ViT, it is hardly corrupted with different augmentations and randomly rotated. You can see it in details in Fig. 1.33.

1.2.5 DINO

DINO (self-DIstillation with NO labels) [66] in a sense is a derivative and a combination of BYOL [61] and SwAV [63] for ViT [3], it's design can be seen in Fig. 1.34.

Here we have student and teacher networks, both are ViTs, and teacher is the EMA of the student. In the training phase we have an image from the batch, create two general views and eight additional views and pass all the views to the student and only general views to the teacher. Teacher network on the forward pass also adding centering unit since it helps to prevent collapsing. After getting representations from both teacher and student, the softmax with carefully chosen temperature is applied as another one term to avoid collapse. Loss function here is a cross-entropy loss.

1.2.6 drloc

Authors of the paper "Efficient Training of Visual Transformers with Small Datasets" [67] introduce dense relative localization loss (drloc) as the additional loss to the standard cross-entropy loss when training ViT based models for classification.

Researchers show that without changes in the architecture of the network, it is possible to significantly improve training convergence speed while improving accuracy score too, using a small dataset for training. It is possible with the proposed self-supervised auxiliary task to regularize the model, which is defined as an ability to measure the relative distance between two random patch embeddings from the output of the network and their location on the grid. It is done with an additional Localization MLP layer which takes as input concatenated embeddings and calculates the distance, the details can be seen in Fig. 1.35.

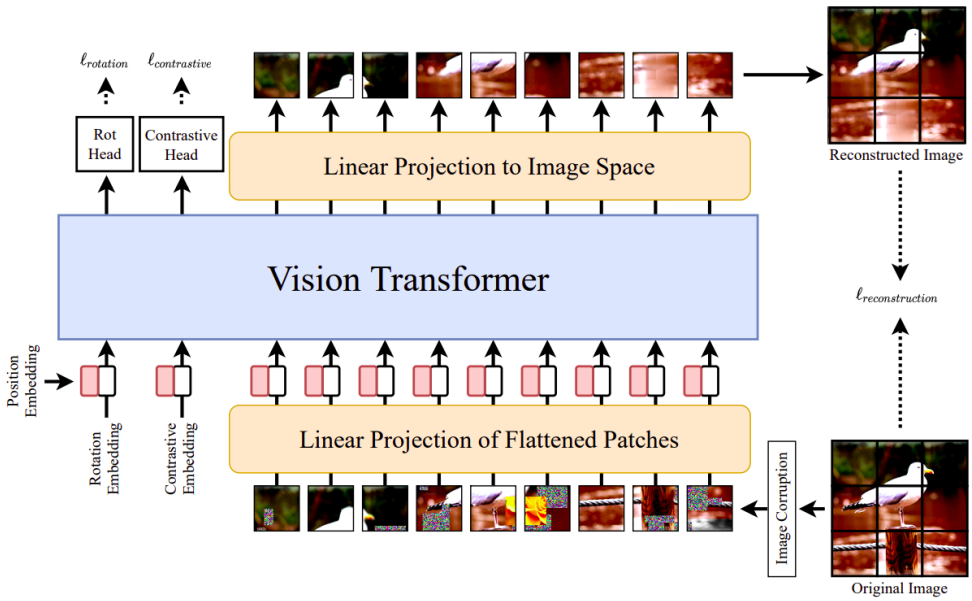

Figure 1.33 — SiT architecture. Taken from original paper [64]

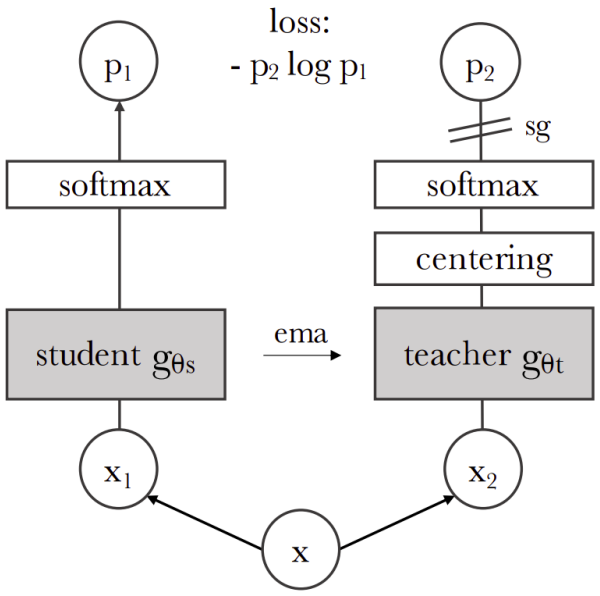

Figure 1.34 — DINO design. Taken from original paper [66]

Authors provide results with training on large datasets with standard settings and nevertheless drloc loss application shows minor improvement. Also, authors claim that positional encoding usage is not sufficient to solve the auxiliary task.

1.2.7 data2vec

Authors of data2vec [68] propose a Self-Supervised Learning framework which is acceptable for different modalities. It uses masking technique, not for additional reconstruction task as regularly, but to use it within the teacher-student learning paradigm. Scheme can be seen in Fig. 1.36.

Authors of this work exploit the versatility of Transformer and expand the images application to the text and audio, it is done via masking and the task for SSL is to predict the state of the teacher's layers. Thus, data2vec forces the model to predict the several representations of different level features.

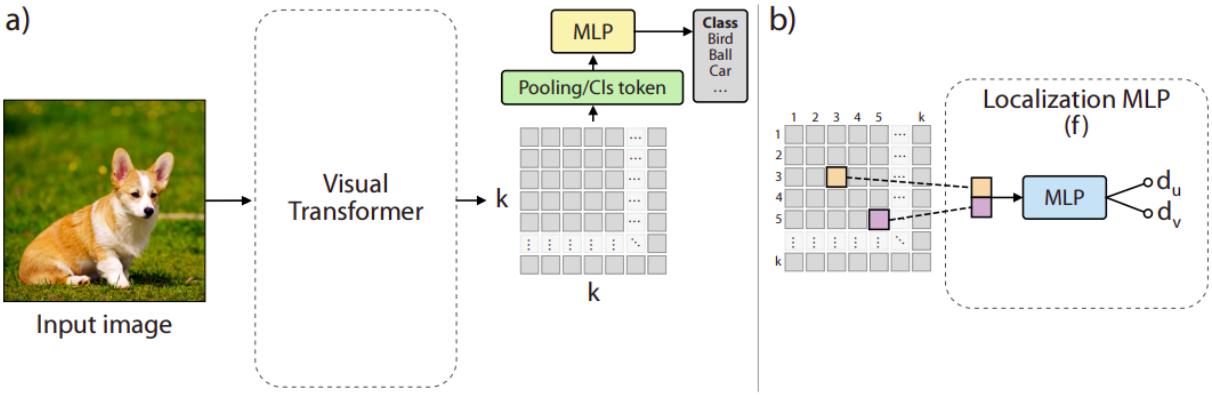

Figure 1.35 — a) Vanilla ViT output; b) Localization MLP scheme. Taken from original paper [67]

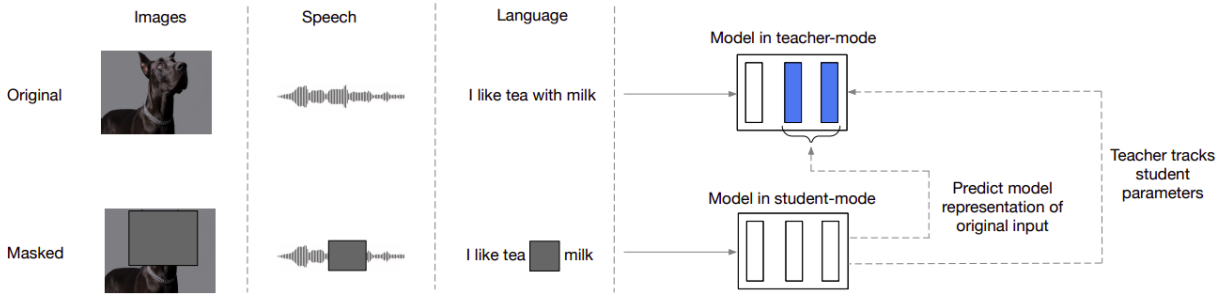

Figure 1.36 — data2vec design. Taken from original paper [68]

2 RELATED WORKS

Complexity of computations is a limiting factor of training ViT. Training a ViT is not the best option if you want to get best trade-off between metrics and computations, and it can be better to use another state-of-the-art model from CNN world. But almost 2 months after presenting ViT, DeiT [11] was shown and presented the way to reduce training time of ViT by an order of magnitude. Authors of this paper proposed to use not only class token, but also a distillation token and train the network using student-teacher paradigm. Results show that using this technique, it is possible to train on open academic datasets like ImageNet at least in days. Nevertheless, authors still used unaffordable computational powers.

Definitely there are other papers that addressed the efficiency problems. CaiT [17] and DeepViT [15] propose changes in Transformer block to improve in-depth scaling, UViT [21] changing the self-attention window size to reduce computations, XCiT [35] is "transposing" self-attention with the same goal. But crucial, we think, is the way to improve performance of ViT using it's architecture as is, but changing the way we train it. And this aim is followed by Steiner et al. [4] and Chen et al. [48]: those works improve training process and results using proven training techniques and changing the optimization strategy.

Significant approach is self-supervised learning. Self-supervised learning gives the model ability to understand the data without any labels solving auxiliary task. After that self-supervised model can solve tasks much more easily. For that one just need to do fine-tuning on the downstream task, e.g. classification. There are works that use this technique. SiT [64] appends ViT so it solves rotation prediction task [54], image reconstruction task and supports contrastive representation learning [69]. BYOL [61] and DINO [66] uses knowledge distillation with momentum teacher [62], SwAV [63] exploiting multi-crop contrastive learning. data2vec [68] proposes redesigned version of mask filling task and student-teacher paradigm: it uses masking technique, but not for additional reconstruction task, but to predict the state of the teacher's layers.

Another interesting approach is one that uses the best from both of the worlds: supervised and self-supervised. Liu et al. [67] proposed to use dense

relative localization (drloc), which is defined as an model's ability to measure the relative distance between two random patch embeddings from the output of the network and their location on the grid. This drloc is used together with supervised loss, allowing the model to understand the data better. Thus, it is multi-task learning method [70]. All of those ideas and others major ones are reviewed and analysed in Chapter 1.

Our approach goes along with a mentioned ideas and combining them to create multi-task learning framework with knowledge distillation.

3 FRAMEWORK

Tort framework is a multi-task learning framework, using self-supervised knowledge distillation, applicable to majority of downstream tasks, such as classification, segmentation and detection. Despite the fact that Tort was made for ViT architecture training, it is model-agnostic, so one can use any NN model. Multi-task learning means that framework has several objectives, each one of which exploits different features of the model, data or training environment. In this case, those tasks are different losses that framework is optimizing to train the model.

Key idea of the Tort is in optimizing the training process: improving the quality of training with a smaller input batch size and a smaller volume and diversity of the dataset. Using multi-task learning and multi-crop strategy we expand the amount of received knowledge by the model so it understands the data and domain better and converges faster.

Framework consists of four main parts, including data preparation, feature extraction, features processing and objectives computation. The outline of the framework is depicted in Fig. 3.1.

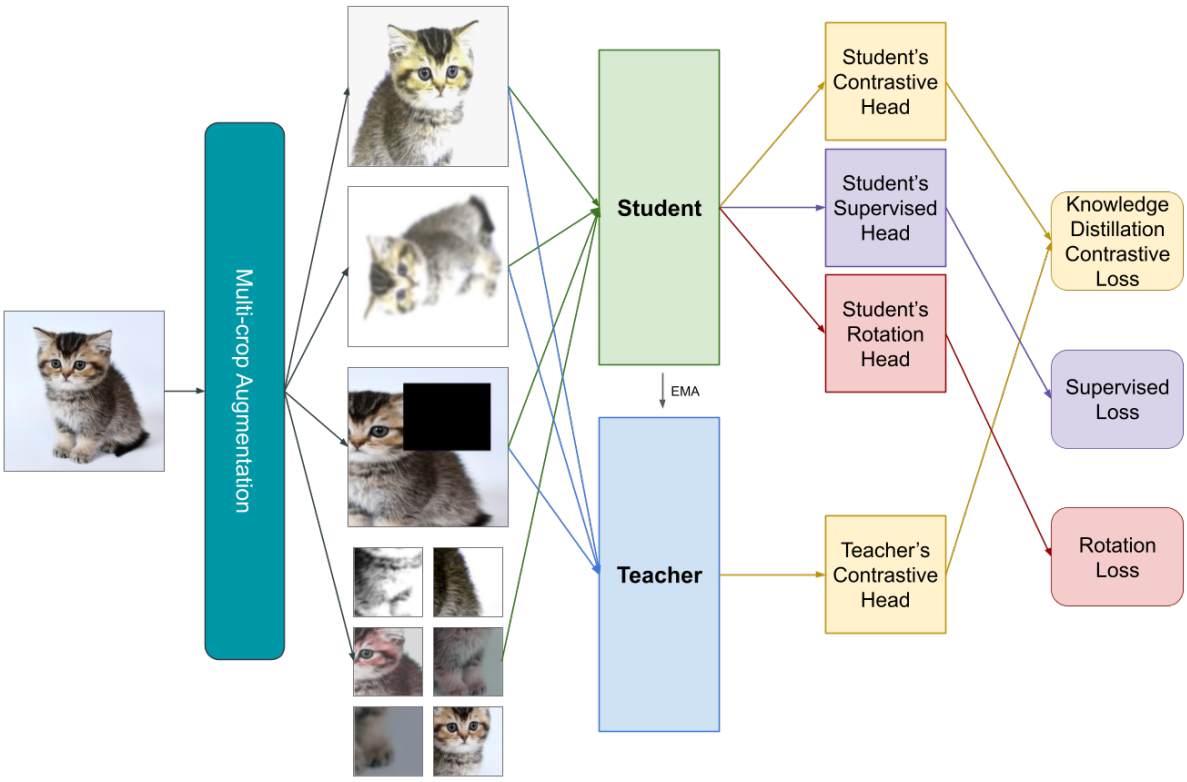

Figure 3.1 — Tort design

3.1 Pipeline

On the first step we construct multiple views of the same image, creating "positive pair" with two global views, masked view and N_a additional views. After that we pass everything to Student model and everything but additional views to Teacher model to process and obtain embeddings extracted from received images. Next we utilize hand-crafted heads for different tasks, passing corresponding embeddings to heads. Following step is calculation of losses and weighting. After backpropagating we update weights of the Teacher model and repeat everything stated above in the new iteration.

Now we describe the mentioned stages deeply. You can follow the detailed version of design in Fig. 3.2.

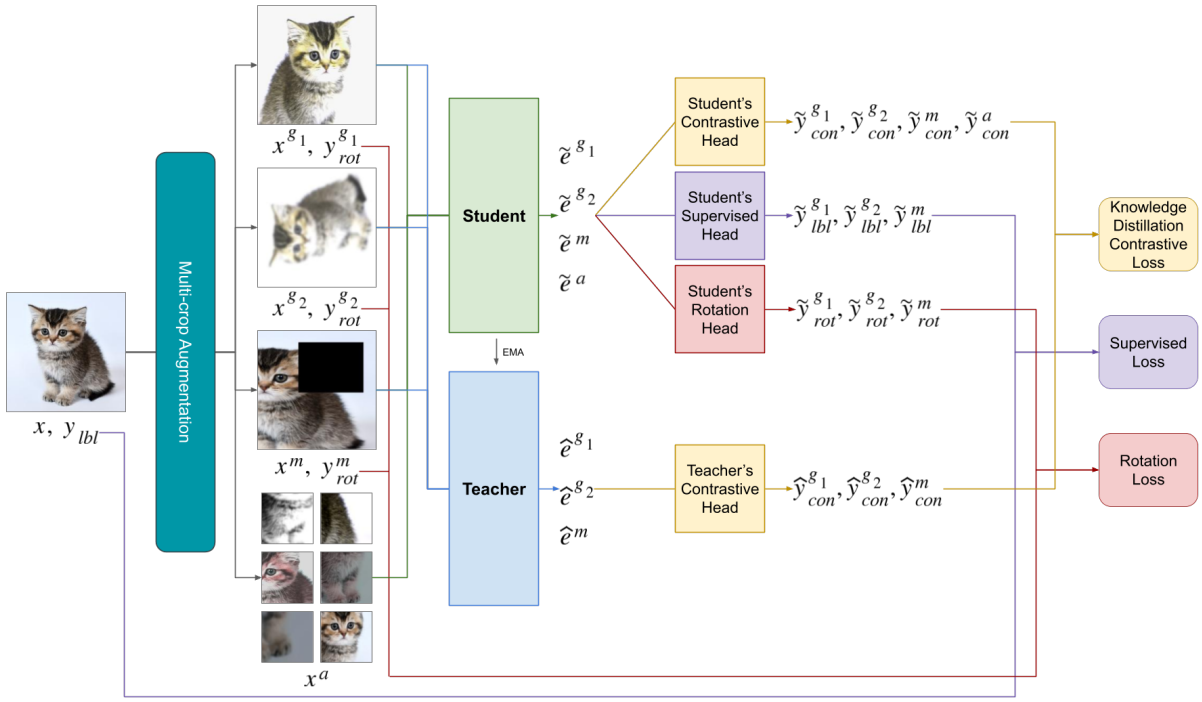

Figure 3.2 — Tort design in details

3.2 Multi-Crop Augmentation

To obtain as much information as possible and give the ViT understanding of the nature of the domain, we should use rich combination of possible augmentations. Thus, we provide ViT knowledge about possible state of objects and it learns better. The fact that more complex augmentations is better than

bigger dataset was proposed and proven in [4] and presented in Section 1.1.4.2. So we follow this guideline since we aim at datasets available for regular people.

Going along with SwAV [63], we employ multi-crop strategy and provide 2 global views x^{g_1}, x^{g_2} of the image x and 6 additional crops x^a. Also, we append 2 global crops with masked global crop x^m, so number of global crops equals to 3. The size of the global and masked views is 224x224, while additional views are 96x96.

Additionally, we randomly rotate global crops by 90, 180 or 270 degrees and save labels for rotation $y^{g_1}_{rot}, y^{g_2}_{rot}, y^m_{rot}$. Also we can vary probability of applying rotation.

3.3 Student and Teacher

During training we utilize student-teacher learning paradigm, and use two equally initialized ViT-B/16 networks. We define Student's weights as g_{θ_s} and Teacher's as g_{θ_t}. While Student's weights g_{θ_s} are being updated via backpropagation, Teacher is the Exponential Moving Average of the Student's weights g_{θ_s} with a teacher decay rate $\tau \in [0,1]$:

$$g_{\theta_t} \leftarrow \tau g_{\theta_t} + (1-\tau) g_{\theta_s} \tag{3.1}$$

Closely to models, there are heads for different tasks which also constructed similarly for Student and Teacher. Thus, Teacher's Contrastive Head is EMA of the Student's one.

We use Student's embeddings $\tilde{e}^{g_1}, \tilde{e}^{g_2}, \tilde{e}^m, \tilde{e}^a$ to compute predictions in all three tasks heads we are using — Contrastive, Supervised and Rotation, but Teacher's embeddings $\hat{e}^{g_1}, \hat{e}^{g_2}, \hat{e}^m$ are only processed in the Contrastive one.

3.4 Tasks Heads

In the training stage with Tort framework a model has three tasks: to compute representation vector from embedding, predict the class and rotation degree. For all of that objectives we use different heads.

Contrastive Head constructed as MLP with three layers, GELU activation [71] between and a weight normalized fully connected layer on the top.

Supervised Head is taking embedding e and projecting it to the label prediction vector y_{lbl} of the shape equal to the number of classes in the dataset.

Rotation Head is presented by the linear layer, projecting embedding e into vector y_{rot} of the shape 4.

3.5 Losses

We have 2 self-supervised losses and 1 supervised. First self-supervised loss is a contrastive loss, but expanded with sharing the Teacher's knowledge. Second one is rotation loss which forces the model to predict the rotation of the image. Supervised loss is about predicting the class of the object, but it can be applied not only to classification tasks, but also to other CV tasks like segmentation or detection.

After obtaining the loss values, we weight them using weight of 1.0 for Knowledge Distillation Contrastive Loss, 2.5 for Supervised Loss and 1.0 for Rotation Loss.

Knowledge Distillation Contrastive Loss Following the DINO [66] approach, we use outputs of the Contrastive Head to further softmax it and obtain comparable probabilities. We also control the sharpness of those predictions with a temperature parameter. Using this scheme, we learn to match the probability distribution by using Cross Entropy loss between Student's probabilities P_s and Teacher's probabilities P_t. Since we have several views of one image, we force to learn the relationship between global and additional crops, minimizing the following sum:

$$\sum_{x \in \{x^{g1}, x^{g2}, x^m\}} \sum_{\substack{x' \in x^a \\ x' \neq x}} -P_t(x) \log P_s(x'). \tag{3.2}$$

In this scheme we can adjust number of global and additional views.

Supervised Loss is calculated using Cross Entropy Loss between Student's label prediction $\tilde{y}^{g1}_{lbl}, \tilde{y}^{g2}_{lbl}, \tilde{y}^{m}_{lbl}$ and GT y_{lbl}. Also optionally label smoothing can be applied.

Rotation Loss is calculated using Cross Entropy Loss between Student's rotation prediction $\tilde{y}^{g1}_{rot}, \tilde{y}^{g2}_{rot}, \tilde{y}^{m}_{rot}$ and GT rotation states $y^{g1}_{rot}, y^{g2}_{rot}, y^{m}_{rot}$.

4 EXPERIMENTS

This chapter demonstrates experiments on training and fine-tuning ViT using baseline settings and Tort framework. Also we provide rich ablation study in Section 4.6.

4.1 Methodology

To show the significance of our method over the baseline, we need to make straight comparison of both computation costs and evaluation metrics.

The most convenient and successful analysis on how to train ViT was done by Steiner et al. [4]. Because of this we use recipes on training and fine-tuning ViT from this paper and mention it as a SOTA method further, we strictly follow the hyper parameters setup training pipeline and all other details, using original training scripts from timm library [72].

4.2 Data

Important point is the generalization ability of the framework. To show the applicability to the different domains we take 4 small size datasets and conduct experiments using them. Those 4 datasets are image classification datasets which are obtained using torchvision [73] library to achieve reproducibility:

a) Flower102 [74] — dataset of flowers (8,190 images of 102 classes).

b) FGVC-Aircraft [75] — dataset of aircrafts (10,000 images of 100 classes).

c) StanfordCARS [76] — dataset of cars (16,185 images of 196 classes).

d) OxfordIIITPet [77] — dataset of cats and dogs (7,349 images of 2 classes).

4.2.1 Preprocessing

After passing image into Multi-Crop Augmentation block, we get 2 global views, 1 masked view and N_a additional views. Global views are obtained using random resize crop of size 224 with scale from 0.4 to 1.0, then we apply horizontal flip, color jittering, blurring and solarization. Masked view is the same crop as global views with flip and color jittering, but it has masked

rectangle with the scale from 0.4 to 0.6. Additional crops are similar to global views, but of the size 96. Also, every view is normalized before passing further.

4.3 Implementation

In this work we base on the timm training pipeline and DINO [66] code with its implementation of multi-crop and self-supervised loss we use. Everything is implemented using Python and PyTorch DL framework.

Tort framework use batch size of 32 with 2 global views, 1 masked views and N_a=6 additional views — this is the maximum that can be fitted if NVIDIA V100 16Gb GPU. We use cosine decay learning rate scheduler for 100 epochs with initial lr=5e-04, minimal lr=1e-06 and warmup lr=1e-08 for the first 10 epochs, then there are 3 cooldown epochs with the minimal lr. As optimizer we use AdamW optimizer [78] with a Lookahead wrapper [79], weight decay equals 0.04. Also we do clip gradients by the norm of 3.0.

We update Teacher by EMA from the momentum of 0.9998 to the 1 using cosine scedule. Also for sharpening probabilities of the Student and Teacher we use temperature of 0.1 and 0.04 respectively. Next thing is centering output of the Teacher over the batch, we do it with the momentum of 0.9.

4.3.1 Architecture

In all of the experiments we used ViT-B/16 architecture from timm library and adapted it for the needs of research. We do not use predefined classification head. Key difference in the architecture is the change of positional embeddings behavior, since we use images of different sizes we do embeddings interpolation when passing additional views.

4.3.2 Setup

During experiments we used VMs only with NVIDIA V100 16Gb GPU. We also seed every experiment to get reproducibility. Every experiment was tracked using wandb service [80] and open-sourced with `https://wandb.ai/detkov/tort`.

4.4 Evaluation

For supervised evaluation we use test or validation splits of the corresponding datasets. For unsupervised evaluation we use k-NN evaluation on model's output.

4.4.1 k-NN evaluation

To better understand the potential of the pre-trained model we use k-NN classifier, built upon the representation vectors from Tort Transformer head. In our experiments we ended up with neighbors number equal 20 but this number may vary a lot from dataset to dataset.

4.5 Results

Here we show significance of Tort over SOTA in training and fine-tuning using 4 above mentioned datasets [1]. For training we use the randomly initialized model and for fine-tuning we use pre-trained weights. As the metric we use Accuracy.

Now we show train results. In the following plots, X-axis is epochs, Y-axis is accuracy and blue line is Tort, red is SOTA.

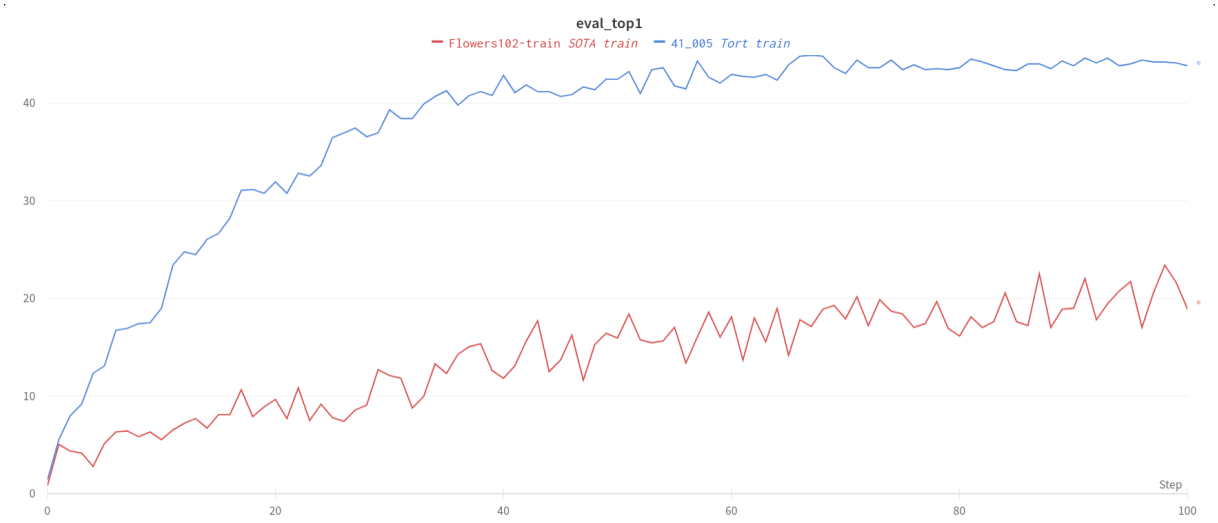

Figure 4.1 — Train results on Flowers102 dataset

[1] You can watch them in interactive mode here: https://wandb.ai/detkov/tort/reports/Tort-results--VmlldzoyMDI1Mzg0

In the Fig. 4.1 with Flowers102 you can see that the difference in absolute accuracy points is almost 25: 44 versus 19.

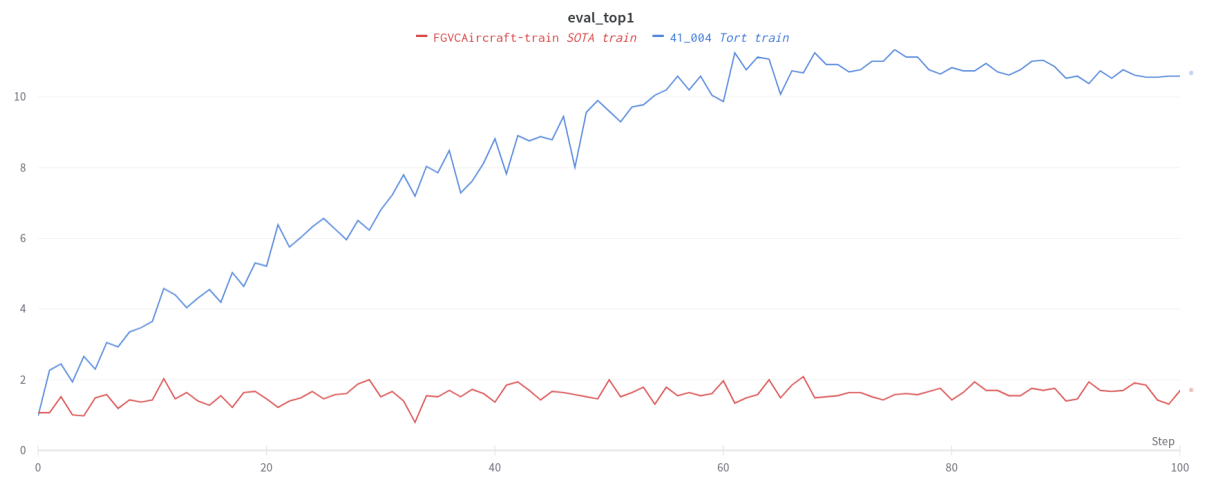

Figure 4.2 — Train results on FGVCAircraft dataset

In the Fig. 4.2 with FGVCAircraft you can see that the difference in absolute accuracy points is almost 9: 10.5 versus 1.5.

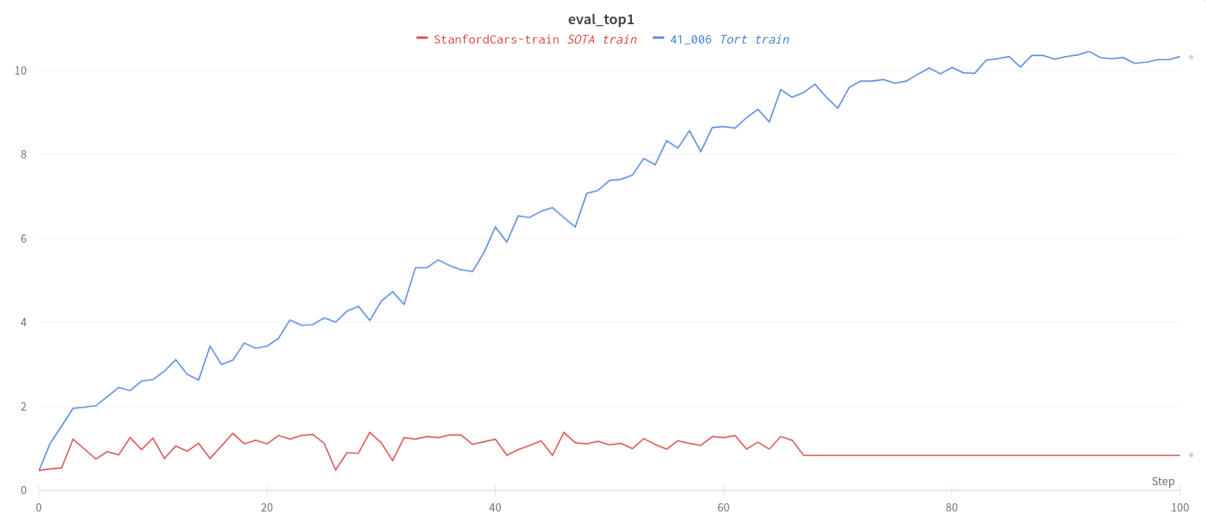

Figure 4.3 — Train results on StanfordCars dataset

In the Fig. 4.3 with StanfordCars you can see that the difference in absolute accuracy points is almost 9: 10 versus 1.

In the Fig. 4.4 with OxfordIIIPet you can see that the difference in absolute accuracy points is almost 22: 24 versus 2.

Now we show fine-tune results. In the following plots, X-axis is epochs, Y-axis is accuracy and turquoise (blue) line is Tort, yellow is SOTA.

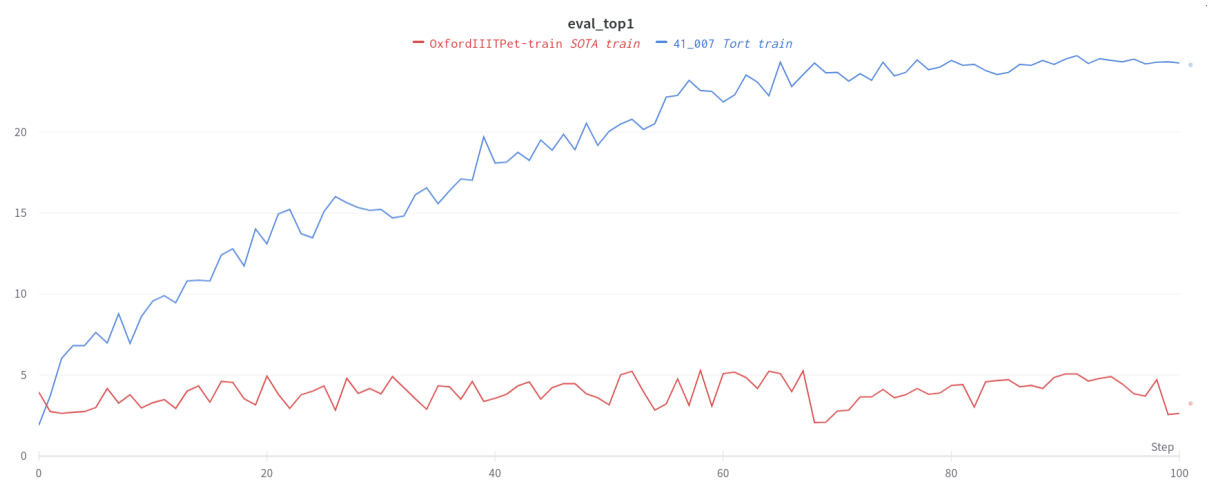

Figure 4.4 — Train results on OxfordIIIPet dataset

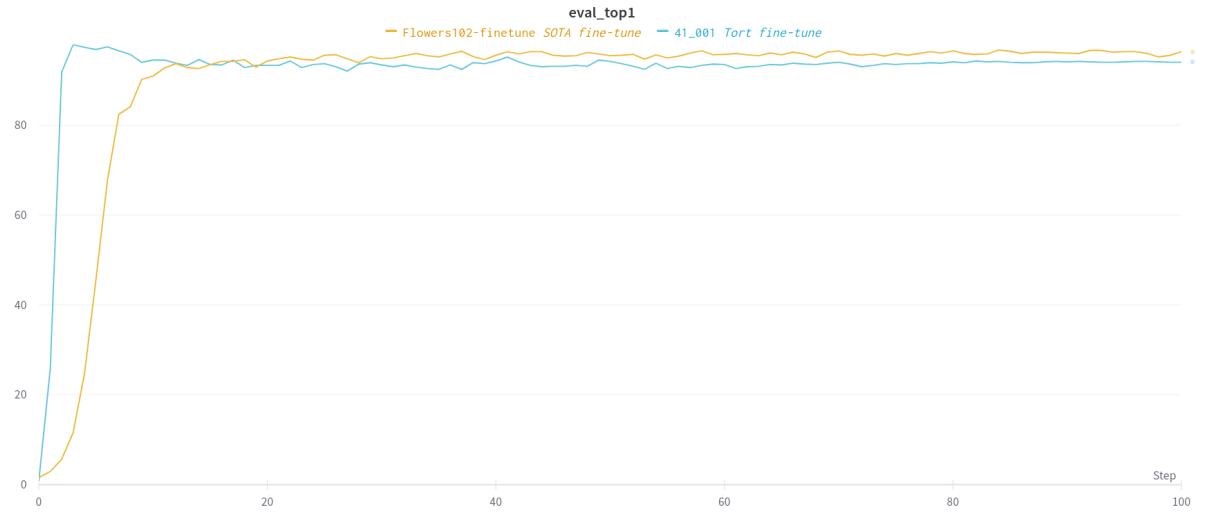

Figure 4.5 — Fine-tune results on Flowers102 dataset

In Fig 4.5 of fine-tuning on Flowers102 we see that Tort reaches best accuracy of 98 on the 3rd epoch, while SOTA is on accuracy of 12, then SOTA reaches 90 on the 9th epoch and its supremum of 96.5 on 37th.

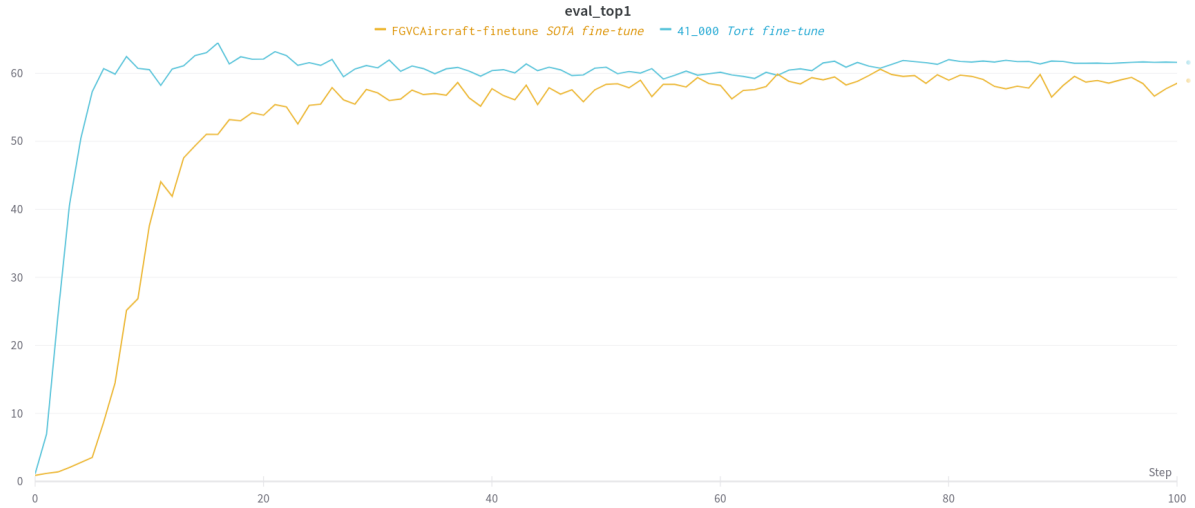

Figure 4.6 — Fine-tune results on FGVCAircraft dataset

In Fig 4.6 of fine-tuning on FGVCAircraft we see that Tort reaches close to optimum accuracy of 61 on the 6th epoch, while SOTA is on accuracy of 9, then SOTA reaches its supremum of 61 on the 74th epoch, while Tort's supremum is 64.5.

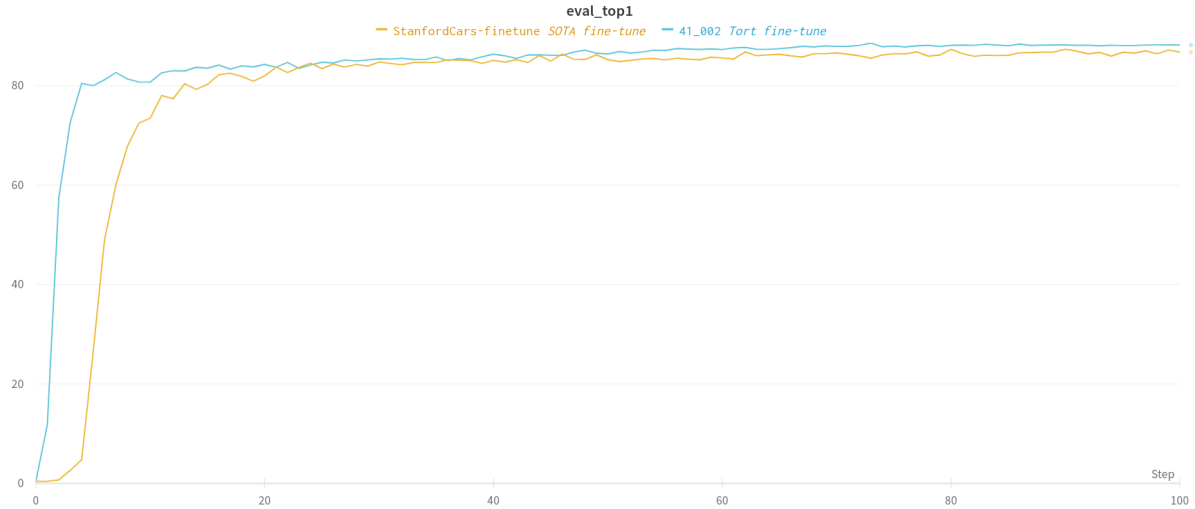

Figure 4.7 — Fine-tune results on StanfordCars dataset

In Fig 4.7 of fine-tuning on StanfordCars we see that Tort reaches optimal accuracy of 80.5 on the 4th epoch, while SOTA is on accuracy of 5, then SOTA reaches maximum of 87 and Tort reaches 88.

In Fig 4.8 of fine-tuning on OxfordIIIPet we see that Tort reaches its best accuracy of 90.6 on the 3rd epoch, while SOTA is on accuracy of 85, but on the 6th epoch SOTA get ahead of Tort and leading with a little margin all to the end of the training.

We conclude that Tort can get up to +25 absolute points of accuracy on training, because in SOTA settings ViT simply can not start converging. Also on fine-tuning we observe up to +23 absolute points of accuracy and up to 5x times faster convergence to optimum.

If we compare training time of Tort and SOTA, Tort has 2x times faster training pipeline than SOTA in our environment.

4.6 Ablation Study

A large study was produced on testing different components of the solution to get understanding on what is improving the result, and what is not.

Objectives We tested the method using independently only one of the three losses. This study showed that Rotation Loss as the only loss can not bring stable training procedure because with default hyper parameters and needed to be carefully adjusted. As about Supervised Loss and Knowledge Distillation Contrastive Loss — they can work alone and bring the optimum alone. However, if 2 we use Rotation Loss together with Knowledge Distillation Contrastive Loss it slightly improves the metrics. And if we use all three together, it provides the best possible result.

Masked view In the framework we use scale of mask from 0.4 to 0.6, but we tried the set [0.1, 0.3], [0.2, 0.4], [0.3, 0.5], [0.4, 0.6], [0.5, 0.7] and found out a strong correlation of the scale with loss, because the more we mask out, the harder it is to create correct representation of the image.

Student temperature In the framework we use Student temperature of 0.1, but we tried the values 0.1, 0.2, 0.3, 0.4, 0.5. It turns out that greater temperature forces the Knowledge Distillation Contrastive Loss to converge faster. Moreover, there is no correlation between the value and the acceleration: the only condition is for the value to be higher than 0.1.

Teacher temperature In the framework we use Teacher temperature of 0.04, but we tried the values 0.02, 0.04, 0.06, 0.08, 0.10. This situation is

similar to the Student temperature, because if we make the value lower than 0.04, the Knowledge Distillation Contrastive Loss converges faster, while higher values do not show statistically significant results.

Number of epochs By the default we use 100 epochs training scheduler, but also tried 200 and 300 epochs. We report that bigger number of epochs delivers better accuracy. But the difference between the 100 epochs training and 200 epochs is in 1.2 points of accuracy, additional 100 epochs get another 1.3 points. We conclude that spending 2x time or 3x time for 0.3% or 0.6% of relative gain in accuracy is ineffective move.

We also conduct ablation study on label smoothing ([0.0, 0.1], rotation probability ([0.5, 1.0]), weight decay ([0.0, 0.1, 0.2, 0.3, 0.4]), minimum lr ([0.000001, 0.00000001, 0]) but they have not shown statistical significance.

4.7 Analysis

Now we discuss the plots for training. If we look at the SOTA training curves we can see either low increase in metric value or stagnation with noise on the same level. We assume that this is happening due to the fact that the model can not get the gradients useful for training, i.e. the backpropagation signal from supervised Cross Entropy Loss is not sufficient and aligned with the true direction to the optimum for a randomly initialized complicated model. One can disagree and argue that the problem is not the signal itself but the step size on optimization. We parry with the fact that during training we use cosine learning rate scheduler with the difference between minimum and maximum in up to 5 orders so if there is some learning rate that is possibly make the model learn, during the training it would happen, but this is not happening. 3 dataset's plots have stagnation, while Flowers102 have upward movement. We think that this is happening because this dataset is more simple than the others for the model.

Thus, this is the fact that something does not allow to train ViT from scratch with small batch size using 1 supervised signal and our assumption is that this is happening because of the not well aligned gradients. Our proposition is to add more information that can be used by the model on the backward pass. With the ablation study on objectives and the plots showing advances of the

method we can prove that this framework indeed allows the model to get more information than just using one supervised signal and learn notably better.

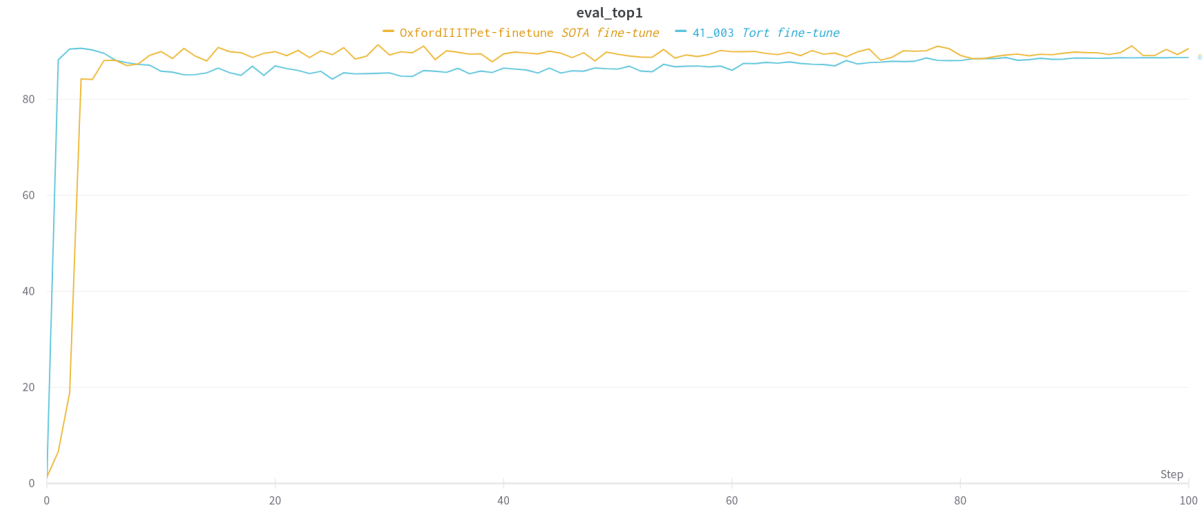

Figure 4.8 — Fine-tune results on OxfordIIIPet dataset

CONCLUSION

During this work we dived deeply into the topics related to Vision Transformers and Self-Supervised Learning, we analysed and designed a new framework which combines recent advances in those domains. Carefully comparing baseline method of training ViT and our framework Tort we shown the significance of the latter over another on the task of classification using small datasets. We demonstrated the importance of components of the solution and make the code and experiments logs open and available to the community with respect to reproducibility of our results.

Indeed, we provided the solution for the task of giving an opportunity to use ViT with small datasets without cluster for making computations. Nevertheless, there are open questions not touched in this work, among them is the usage for dense prediction task and usage with other NNs. We design the method so it is model-agnostic and can be used for any downstream task but never tested it. Also we know that there is the room for improvement, for example with the changing of the ViT architecture to less resource-demanding variant or the usage of different optimization algorithms.

However, the Tort framework reached the stated goal and now using our method one can apply it for own datasets and tasks and be sure that the ViT training would not collapse and output the desired results.

REFERENCES

1. *Krizhevsky, Alex*. ImageNet Classification with Deep Convolutional Neural Networks / Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton // Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. — Nips'12. — Red Hook, NY, USA: Curran Associates Inc., 2012. — P. 1097–1105.

2. Attention Is All You Need / Ashish Vaswani, Noam Shazeer, Niki Parmar et al. // *CoRR*. — 2017. — Vol. abs/1706.03762. http://arxiv.org/abs/1706.03762.

3. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale / Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov et al. // *CoRR*. — 2020. — Vol. abs/2010.11929. https://arxiv.org/abs/2010.11929.

4. How to train your ViT? Data, Augmentation, and Regularization in Vision Transformers / Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai et al. // *CoRR*. — 2021. — Vol. abs/2106.10270. https://arxiv.org/abs/2106.10270.

5. *Bahdanau, Dzmitry*. Neural Machine Translation by Jointly Learning to Align and Translate. — 2016.

6. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation / Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre et al. // *CoRR*. — 2014. — Vol. abs/1406.1078. http://arxiv.org/abs/1406.1078.

7. *Hochreiter, Sepp*. Long short-term memory / Sepp Hochreiter, Jürgen Schmidhuber // *Neural computation*. — 1997. — Vol. 9, no. 8. — Pp. 1735–1780.

8. Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // *CoRR*. — 2015. — Vol. abs/1512.03385. http://arxiv.org/abs/1512.03385.

9. ImageNet: A large-scale hierarchical image database / Jia Deng, Wei Dong, Richard Socher et al. // 2009 IEEE Conference on Computer Vision and Pattern Recognition. — 2009. — Pp. 248–255.

10. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era / Chen Sun, Abhinav Shrivastava, Saurabh Singh, Abhinav Gupta // 2017 IEEE International Conference on Computer Vision (ICCV). — 2017. — Pp. 843–852.

11. Training data-efficient image transformers & distillation through attention / Hugo Touvron, Matthieu Cord, Matthijs Douze et al. // *CoRR*. — 2020. — Vol. abs/2012.12877. https://arxiv.org/abs/2012.12877.

12. Transformer in Transformer / Kai Han, An Xiao, Enhua Wu et al. // *CoRR*. — 2021. — Vol. abs/2103.00112. https://arxiv.org/abs/2103.00112.

13. Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet / Li Yuan, Yunpeng Chen, Tao Wang et al. // *CoRR*. — 2021. — Vol. abs/2101.11986. https://arxiv.org/abs/2101.11986.

14. ConViT: Improving Vision Transformers with Soft Convolutional Inductive Biases / Stéphane d'Ascoli, Hugo Touvron, Matthew L. Leavitt et al. // *CoRR*. — 2021. — Vol. abs/2103.10697. https://arxiv.org/abs/2103.10697.

15. DeepViT: Towards Deeper Vision Transformer / Daquan Zhou, Bingyi Kang, Xiaojie Jin et al. // *CoRR*. — 2021. — Vol. abs/2103.11886. https://arxiv.org/abs/2103.11886.

16. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows / Ze Liu, Yutong Lin, Yue Cao et al. // *CoRR*. — 2021. — Vol. abs/2103.14030. https://arxiv.org/abs/2103.14030.

17. Going deeper with Image Transformers / Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles et al. // *CoRR*. — 2021. — Vol. abs/2103.17239. https://arxiv.org/abs/2103.17239.

18. Not All Images are Worth 16x16 Words: Dynamic Vision Transformers with Adaptive Sequence Length / Yulin Wang, Rui Huang, Shiji Song et al. // *CoRR*. — 2021. — Vol. abs/2105.15075. https://arxiv.org/abs/2105.15075.

19. Shuffle Transformer: Rethinking Spatial Shuffle for Vision Transformer / Zilong Huang, Youcheng Ben, Guozhong Luo et al. // *CoRR*. — 2021. — Vol. abs/2106.03650. https://arxiv.org/abs/2106.03650.

20. CoAtNet: Marrying Convolution and Attention for All Data Sizes / Zihang Dai, Hanxiao Liu, Quoc V. Le, Mingxing Tan // *CoRR*. — 2021. — Vol. abs/2106.04803. https://arxiv.org/abs/2106.04803.

21. A Simple Single-Scale Vision Transformer for Object Localization and Instance Segmentation / Wuyang Chen, Xianzhi Du, Fan Yang et al. // *CoRR*. — 2021. — Vol. abs/2112.09747. https://arxiv.org/abs/2112.09747.

22. End-to-End Object Detection with Transformers / Nicolas Carion, Francisco Massa, Gabriel Synnaeve et al. // *CoRR*. — 2020. — Vol. abs/2005.12872. https://arxiv.org/abs/2005.12872.

23. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers / Sixiao Zheng, Jiachen Lu, Hengshuang Zhao et al. // *CoRR*. — 2020. — Vol. abs/2012.15840. https://arxiv.org/abs/2012.15840.

24. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers / Enze Xie, Wenhai Wang, Zhiding Yu et al. // *CoRR*. — 2021. — Vol. abs/2105.15203. https://arxiv.org/abs/2105.15203.

25. Segmenter: Transformer for Semantic Segmentation / Robin Strudel, Ricardo Garcia Pinel, Ivan Laptev, Cordelia Schmid // *CoRR*. — 2021. — Vol. abs/2105.05633. https://arxiv.org/abs/2105.05633.

26. Fully Transformer Networks for Semantic Image Segmentation / Sitong Wu, Tianyi Wu, Fangjian Lin et al. // *CoRR*. — 2021. — Vol. abs/2106.04108. https://arxiv.org/abs/2106.04108.

27. *Yan, Haotian*. Lawin Transformer: Improving Semantic Segmentation Transformer with Multi-Scale Representations via Large Window Attention. — 2022.

28. Designing Network Design Spaces / Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick et al. // *CoRR*. — 2020. — Vol. abs/2003.13678. https://arxiv.org/abs/2003.13678.

29. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices / Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun // *CoRR*. — 2017. — Vol. abs/1707.01083. http://arxiv.org/abs/1707.01083.

30. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions / Wenhai Wang, Enze Xie, Xiang Li et al. // *CoRR*. — 2021. — Vol. abs/2102.12122. https://arxiv.org/abs/2102.12122.

31. CvT: Introducing Convolutions to Vision Transformers / Haiping Wu, Bin Xiao, Noel Codella et al. // *CoRR*. — 2021. — Vol. abs/2103.15808. https://arxiv.org/abs/2103.15808.

32. LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference / Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron et al. // *CoRR*. — 2021. — Vol. abs/2104.01136. https://arxiv.org/abs/2104.01136.

33. ConTNet: Why not use convolution and transformer at the same time? / Haotian Yan, Zhe Li, Weijian Li et al. // *CoRR*. — 2021. — Vol. abs/2104.13497. https://arxiv.org/abs/2104.13497.

34. Twins: Revisiting Spatial Attention Design in Vision Transformers / Xiangxiang Chu, Zhi Tian, Yuqing Wang et al. // *CoRR*. — 2021. — Vol. abs/2104.13840. https://arxiv.org/abs/2104.13840.

35. XCiT: Cross-Covariance Image Transformers / Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron et al. // *CoRR*. — 2021. — Vol. abs/2106.09681. https://arxiv.org/abs/2106.09681.

36. *Cheng, Bowen*. Per-Pixel Classification is Not All You Need for Semantic Segmentation / Bowen Cheng, Alexander G. Schwing, Alexander Kirillov // *CoRR*. — 2021. — Vol. abs/2107.06278. https://arxiv.org/abs/2107.06278.

37. *Ronneberger, Olaf*. U-Net: Convolutional Networks for Biomedical Image Segmentation / Olaf Ronneberger, Philipp Fischer, Thomas Brox // *CoRR*. — 2015. — Vol. abs/1505.04597. http://arxiv.org/abs/1505.04597.

38. Scene Parsing through ADE20K Dataset / Bolei Zhou, Hang Zhao, Xavier Puig et al. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. — 2017.

39. Semantic understanding of scenes through the ade20k dataset / Bolei Zhou, Hang Zhao, Xavier Puig et al. // *International Journal of*

Computer Vision. — 2019. — Vol. 127, no. 3. — Pp. 302–321.

40. *Carion, Nicolas*. — 2020. — May. https://ai.facebook.com/blog/end-to-end-object-detection-with-transformers/.

41. Feature Pyramid Networks for Object Detection / Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick et al. // *CoRR*. — 2016. — Vol. abs/1612.03144. http://arxiv.org/abs/1612.03144.

42. Do We Really Need Explicit Position Encodings for Vision Transformers? / Xiangxiang Chu, Bo Zhang, Zhi Tian et al. // *CoRR*. — 2021. — Vol. abs/2102.10882. https://arxiv.org/abs/2102.10882.

43. *Yu, Fisher*. Multi-Scale Context Aggregation by Dilated Convolutions. — 2016.

44. Rethinking Atrous Convolution for Semantic Image Segmentation / Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam // *CoRR*. — 2017. — Vol. abs/1706.05587. http://arxiv.org/abs/1706.05587.

45. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs / Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos et al. // *CoRR*. — 2016. — Vol. abs/1606.00915. http://arxiv.org/abs/1606.00915.

46. Pyramid Scene Parsing Network / Hengshuang Zhao, Jianping Shi, Xiaojuan Qi et al. // *CoRR*. — 2016. — Vol. abs/1612.01105. http://arxiv.org/abs/1612.01105.

47. MLP-Mixer: An all-MLP Architecture for Vision / Ilya O. Tolstikhin, Neil Houlsby, Alexander Kolesnikov et al. // *CoRR*. — 2021. — Vol. abs/2105.01601. https://arxiv.org/abs/2105.01601.

48. *Chen, Xiangning*. When Vision Transformers Outperform ResNets without Pretraining or Strong Data Augmentations / Xiangning Chen, Cho-Jui Hsieh, Boqing Gong // *CoRR*. — 2021. — Vol. abs/2106.01548. https://arxiv.org/abs/2106.01548.

49. Sharpness-Aware Minimization for Efficiently Improving Generalization / Pierre Foret, Ariel Kleiner, Hossein Mobahi, Behnam Neyshabur // *CoRR*. — 2020. — Vol. abs/2010.01412. https://arxiv.org/abs/2010.01412.

50. Deep Networks with Stochastic Depth / Gao Huang, Yu Sun, Zhuang Liu et al. // *CoRR*. — 2016. — Vol. abs/1603.09382. http://arxiv.org/abs/1603.09382.

51. RandAugment: Practical data augmentation with no separate search / Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, Quoc V. Le // *CoRR*. — 2019. — Vol. abs/1909.13719. http://arxiv.org/abs/1909.13719.

52. mixup: Beyond Empirical Risk Minimization / Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, David Lopez-Paz // *CoRR*. — 2017. — Vol. abs/1710.09412. http://arxiv.org/abs/1710.09412.

53. *Kingma, Diederik P.* Adam: A Method for Stochastic Optimization. — 2014. https://arxiv.org/abs/1412.6980.

54. *Gidaris, Spyros*. Unsupervised Representation Learning by Predicting Image Rotations / Spyros Gidaris, Praveer Singh, Nikos Komodakis // *CoRR*. — 2018. — Vol. abs/1803.07728. http://arxiv.org/abs/1803.07728.

55. *Doersch, Carl*. Unsupervised Visual Representation Learning by Context Prediction / Carl Doersch, Abhinav Gupta, Alexei A. Efros // *CoRR*. — 2015. — Vol. abs/1505.05192. http://arxiv.org/abs/1505.05192.

56. *Noroozi, Mehdi*. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles / Mehdi Noroozi, Paolo Favaro // *CoRR*. — 2016. — Vol. abs/1603.09246. http://arxiv.org/abs/1603.09246.

57. Learning Image Representations by Completing Damaged Jigsaw Puzzles / Dahun Kim, Donghyeon Cho, Donggeun Yoo, In So Kweon // *CoRR*. — 2018. — Vol. abs/1802.01880. http://arxiv.org/abs/1802.01880.

58. *Zhang, Richard*. Colorful Image Colorization / Richard Zhang, Phillip Isola, Alexei A. Efros // *CoRR*. — 2016. — Vol. abs/1603.08511. http://arxiv.org/abs/1603.08511.

59. Context Encoders: Feature Learning by Inpainting / Deepak Pathak, Philipp Krähenbühl, Jeff Donahue et al. // *CoRR*. — 2016. — Vol. abs/1604.07379. http://arxiv.org/abs/1604.07379.

60. A Simple Framework for Contrastive Learning of Visual Representations / Ting Chen, Simon Kornblith, Mohammad Norouzi, Geof-

frey E. Hinton // *CoRR*. — 2020. — Vol. abs/2002.05709. https://arxiv.org/abs/2002.05709.

61. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning / Jean-Bastien Grill, Florian Strub, Florent Altché et al. // *CoRR*. — 2020. — Vol. abs/2006.07733. https://arxiv.org/abs/2006.07733.

62. Momentum Contrast for Unsupervised Visual Representation Learning / Kaiming He, Haoqi Fan, Yuxin Wu et al. // *CoRR*. — 2019. — Vol. abs/1911.05722. http://arxiv.org/abs/1911.05722.

63. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments / Mathilde Caron, Ishan Misra, Julien Mairal et al. // *CoRR*. — 2020. — Vol. abs/2006.09882. https://arxiv.org/abs/2006.09882.

64. *Ahmed, Sara Atito Ali*. SiT: Self-supervised vIsion Transformer / Sara Atito Ali Ahmed, Muhammad Awais, Josef Kittler // *CoRR*. — 2021. — Vol. abs/2104.03602. https://arxiv.org/abs/2104.03602.

65. *Kendall, Alex*. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics / Alex Kendall, Yarin Gal, Roberto Cipolla // *CoRR*. — 2017. — Vol. abs/1705.07115. http://arxiv.org/abs/1705.07115.

66. Emerging Properties in Self-Supervised Vision Transformers / Mathilde Caron, Hugo Touvron, Ishan Misra et al. // *CoRR*. — 2021. — Vol. abs/2104.14294. https://arxiv.org/abs/2104.14294.

67. Efficient Training of Visual Transformers with Small-Size Datasets / Yahui Liu, Enver Sangineto, Wei Bi et al. // *CoRR*. — 2021. — Vol. abs/2106.03746. https://arxiv.org/abs/2106.03746.

68. data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language / Alexei Baevski, Wei-Ning Hsu, Qiantong Xu et al. // *CoRR*. — 2022. — Vol. abs/2202.03555. https://arxiv.org/abs/2202.03555.

69. *Chopra, S.* Learning a similarity metric discriminatively, with application to face verification / S. Chopra, R. Hadsell, Y. LeCun // 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). — Vol. 1. — 2005. — Pp. 539–546 vol. 1.

70. *Crawshaw, Michael*. Multi-Task Learning with Deep Neural Networks: A Survey / Michael Crawshaw // *CoRR*. — 2020. — Vol. abs/2009.09796. https://arxiv.org/abs/2009.09796.

71. *Hendrycks, Dan*. Bridging Nonlinearities and Stochastic Regularizers with Gaussian Error Linear Units / Dan Hendrycks, Kevin Gimpel // *CoRR*. — 2016. — Vol. abs/1606.08415. http://arxiv.org/abs/1606.08415.

72. *Wightman, Ross*. PyTorch Image Models. — https://github.com/rwightman/pytorch-image-models. — 2019.

73. *pytorch*. Torchvision. — https://github.com/pytorch/vision.

74. *Nilsback, Maria-Elena*. Automated Flower Classification over a Large Number of Classes / Maria-Elena Nilsback, Andrew Zisserman // Indian Conference on Computer Vision, Graphics and Image Processing. — 2008. — Dec.

75. Tech. Rep.: / S. Maji, J. Kannala, E. Rahtu et al.: 2013.

76. 3D Object Representations for Fine-Grained Categorization / Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei // 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13). — Sydney, Australia: 2013.

77. Cats and Dogs / Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, C. V. Jawahar // IEEE Conference on Computer Vision and Pattern Recognition. — 2012.

78. *Loshchilov, Ilya*. Fixing Weight Decay Regularization in Adam / Ilya Loshchilov, Frank Hutter // *CoRR*. — 2017. — Vol. abs/1711.05101. http://arxiv.org/abs/1711.05101.

79. Lookahead Optimizer: k steps forward, 1 step back / Michael R. Zhang, James Lucas, Geoffrey E. Hinton, Jimmy Ba // *CoRR*. — 2019. — Vol. abs/1907.08610. http://arxiv.org/abs/1907.08610.

80. *Biewald, Lukas*. Experiment Tracking with Weights and Biases. — 2020. — Software available from wandb.com. https://www.wandb.com/.