university of groningen / faculty of science and engineering / artificial intelligence

Transferability of Vision Transformers: The key to success on small art classification datasets?

Bachelor's Project Thesis

V. Tonkes, s3617157, v.tonkes@student.rug.nl,
Supervisor: M. Sabatelli, PhD

Abstract: Convolutional Neural Networks (CNNs) have become the de facto standard in Computer Vision, and played a major role in the advancement of this field. In recent years, however, Vision-Transformer-based models (VTs) have been outperforming CNNs across the board. While this is exciting, CNNs still have an advantage on small datasets, due to their so-called image-specific inductive bias. Recent work suggests that transfer learning methods allow VTs to compete with CNNs on some of these small datasets. This thesis investigates whether that also holds true for art classification problems within the digital humanities. To this end, it compares popular VTs and CNNs in terms of their off-the-shelf and fine-tuning transferability, when going from *ImageNet1K* to target tasks provided by the *Rijksmuseum Challenge* dataset. The results show that VTs possess superior off-the-shelf feature extraction capabilities here, and that in general, transfer learning allows VTs to become an interesting alternative to consider within the digital humanities.

1 Introduction

Since the introduction of AlexNet, roughly a decade ago, *Convolutional Neural Networks* (CNNs) have played an important role in *Computer Vision* (CV) (Krizhevsky et al., 2012). Recently, however, a new type of architecture, called *Vision Transformer* (VT), gained state-of-the-art performance on common learning benchmarks, including the ImageNet dataset (Deng et al., 2009)[1].

Exciting as this may be, a plain VT is not likely to become useful in circumstances with limited training data and/or only consumer-grade hardware available for learning. This is due to a limitation inherent to the VT architecture, which will be elaborated on in section 1.2. Consequently, the current paper investigates whether utilizing *Transfer Learning* (TL) capabilities of VTs can help overcome these limitations, and if this allows VTs to become the preferred architecture in the aforementioned circumstances as well.

Before going into more detail, however, some background information is needed. The remainder of this section will first briefly describe CNNs and VTs, and compare them to one another. What follows, is an explanation of TL, and why it might help VTs in circumstances with limited data/compute available. Finally, related work is discussed, and a research question is proposed.

1.1 Convolutional Neural Networks

The modern CNN architecture is often attributed to LeCun et al. (1998), which describes how the number of trainable weights per layer can be reduced in manners that are sensible for an image's topology (i.e. by restricting connections to a neuron's local receptive field, and by sharing weights).

CNNs are largely invariant to shifts and small distortions of the input image. More important is that they can be seen as automatic feature extractors. It is well established (e.g. LeCun et al., 1998; Yosinski et al., 2014) that the first layers of a CNN detect low-level features, such as edges and corners. These are combined in subsequent layers to detect higher-order features. The final layer should then give a representative summary of the input,

[1] At the time of writing, the VT `CoCa` is state of the art, according to `https://paperswithcode.com/sota/image-classification-on-imagenet`.

such that, for example, a Multilayer Perceptron-type (MLP-type) layer can extract all the information it needs from it to perform the task at hand.

1.2 Vision Transformers

Architecture VTs were first introduced in Dosovitskiy et al. (2020), and are based on the *encoder stack* of the *Transformer* architecture (Vaswani et al., 2017). This stack takes a sequence of vectors as input, and is composed of alternating *multi-headed self-attention* layers and position-wise MLP-type layers – position-wise meaning the MLP's input is the size of a single position (constituent vector), and is shared between all positions.

The self-attention mechanism plays a central role in VTs, and is worth elaborating on. Each head of a self-attention layer contains three learned projection matrices, which are used to produce *key*, *value*, and *query* vectors for every position in the input sequence. The output for a single position is a weighted sum of all *value* vectors. Weights are obtained by matching this position's *query* with all *key* vectors[2]. As such, an attention head decides itself how much of each position should be incorporated in subsequent representations.

The original Transformer architecture was designed to solve problems in *natural language processing* – a field it now dominates (Wolf et al., 2020). To make it work for CV applications, VTs divide an image into square patches, which get treated as tokens (words). Learned embeddings convert these into a sequence of vectors, to which 1D positional embeddings are added[3]. A learned [*class*] vector gets prepended to the sequence, and its representation at the end of the encoder stack is the final output, which can then be used for classification, segmentation, etc.

CNNs versus VTs While CNNs reduce the number of trainable weights by taking an image's topology into account, VTs do this by learning themselves what information they should incorporate in successive representations of the input. As such, they are not restricted to a local receptive field, but can potentially include information from all over the previous layer.

Moreover, weight sharing applies as much to VTs as it does to CNNs. In VTs, each position goes through the same computational pipeline of being multiplied by 3 projection matrices, followed by a weighted sum calculation, followed by being fed through an MLP-type layer[4]. In CNNs these pipelines take the form of convolution kernels.

A limitation of VTs While VTs are promising; the fact that they are not specifically designed for CV tasks makes that they lack a so-called image-specific *Inductive Bias*, which CNNs do have. Intuitively this means a VT first has to learn what an image is, before it can move on to the learning task at hand[5]. The result is that they require substantially more training data (and time) than CNNs do.

It is for this reason the paper that first introduced VTs (Dosovitskiy et al., 2020) mentions using pre-trained models rather than randomly initialized ones, as it allows general knowledge about images to be carried over to the new task. This technique falls within the category of TL, which will be covered next.

1.3 Transfer Learning

Given two *Machine Learning* (ML) problems, of which one is called the source, and the other the target task, Transfer Learning can be defined as *exploiting knowledge implied in the source task to improve performance of an ML model on the target task*. A formal definition is found on p. 56 of Sabatelli (2022), but for the purposes of this paper, the one above suffices.

In practice, many TL techniques exist. The example given in section 1.2 can be described as *fine-tuning* (FT). Here, a pre-trained model is used as a starting point, and then trained on the target task – possibly with different hyperparameters, and often with a new final layer that fits the task. Yosinski et al. (2014) studied transferability properties of CNNs pre-trained on ImageNet, and suggested that

[2] In principle by taking the softmax over all query-key inner products.

[3] Do mind that VTs have no built-in knowledge of an image's topology, as the architecture ignores the 2D layout.

[4] Neglecting skip connections, normalization and multi-headedness, as this is a comparison on an intuitive level.

[5] Recall, for example, that the architecture ignores the 2D layout of an image, and hence has to learn about this through training.

FT results in improved performance and generalizability compared to training from scratch. Moreover, while transferability decreased as the source and target task became more dissimilar, utilizing TL still seemed to be preferred.

Another TL technique is called *off-the-shelf* (OTS) learning. It is similar to FT, except that all but the final layer is kept frozen during training. Sharif Razavian et al. (2014) found that features extracted by a CNN pre-trained on ImageNet were general enough for OTS learning to work. More specifically, these OTS CNNs showed better performance than the (fully trained non-CNN) state-of-the-art at the time. Besides reducing computational cost of training, a benefit of OTS learning is that it can prevent overfitting in cases where FT would not (Yosinski et al., 2014).

1.4 Related work and the current study

As mentioned, the paper that first introduced VTs already demonstrated they benefit from TL (Dosovitskiy et al., 2020), and more has been discovered about VTs' TL properties since then. Matsoukas et al. (2021) investigated whether TL allowed VTs to replace CNNs in Medical Imaging. This domain is characterized by small datasets, so to overcome the inductive bias problem described earlier, TL is imperative. It was found that VTs benefit more from FT than CNNs do, and that VTs pre-trained on ImageNet perform on par with CNNs after this type of TL.

What Matsoukas et al. did not investigate, is the OTS performance of VTs. This was, however, done by Zhou et al. (2021), which found that VTs pre-trained on ImageNet had worse OTS performance than their CNN counterparts. An exception was the `Swin` architecture, which was the best performing model on all datasets, and uses *(shifted-)window-based* self-attention methods, to make Transformers better suited for CV tasks (Liu et al., 2021).

Another 'shortcoming' of Matsoukas et al. is that it merely examined whether VTs can use TL to outperform CNNs in a domain that happens to be characterized by limited-sized datasets, but that it did not actively focus on the extent to which VTs can be made useful in limited data/compute environments. More related to this is Touvron et al. (2021), which proposed transformer-specific distillation-based learning techniques to make VTs applicable for smaller datasets. This is not related to FT or OTS learning, of course.

The current thesis will examine to what extent TL techniques (both FT and OTS) allow VTs to become the preferred architecture when resources are limited, and attempts to find a methodology that is recommended in these circumstances. Several experiments will be performed, of which results will be analyzed both quantitatively as well as qualitatively.

Limited resources, in this case, primarily refers to the amount of available training data. Datasets should be so small that one can reasonably assume they were hand labeled by a single person. In addition, it also refers to compute resources, which should be restricted to consumer-grade hardware. In practice, of course, the small datasets already ensure that this restriction is met, because these naturally allow such hardware to suffice for learning.

Furthermore, this thesis restricts itself to art classification tasks specifically. These tasks fit in well with the described objectives, since the amount of effort needed to create and digitize an art collection often makes for small datasets. Moreover, they provide an interesting TL challenge, as knowledge might have to be carried over from natural to non-natural images.

To put all of this more formally, the following research question is proposed:

> *To what extent (and in what manners) does utilizing TL allow VTs to perform on par with CNNs on common art classification problems, when datasets contain 10,000 images or less?*

2 Methods

Much of the experiments and analysis discussed below, build upon the work done by Sabatelli et al. (2018). This paper investigated TL properties of CNNs in the art classification domain. The main focus was on a comparison between CNN architectures pre-trained on ImageNet1K when they were either used as OTS feature extractors, or fine-tuned to the new task. It was suggested that fine-tuning resulted in substantially better performance than

either training from scratch or using an OTS learning scheme.

2.1 Dataset

Similar to Sabatelli et al., used models are pretrained on ImageNet1K, and the *Rijksmuseum Challenge* dataset (Mensink & van Gemert, 2014) is used for the target tasks. This dataset consists of a large collection of digitized artworks, together with xml-formatted metadata. The classification tasks extracted from it are: (1) *Type classification*, where the model has to distinguish between classes as 'painting', 'sculpture', 'drawing', etc.; (2) *Material classification*, with classes as 'paper', 'porcelain', 'silver', etc.; and finally, (3) *Artist classification*, where the model has to predict who the creator is. Figure 2.1 shows labeled examples for all three tasks.

The full dataset contains 112,039 images, and allows for multiple or zero labels per classification task. Naturally, samples are excluded from a task if they contain no labels for it. Samples containing more than one label are excluded as well, since analysis showed that in all cases this is true for only a small portion of the dataset.

There are still many samples and classes left after doing the operations described above. Samples are also unevenly distributed among classes, with Gini-coefficients being as high as 0.98[6]. To counteract this, only the 30 most occurring classes are selected, and a cap of 1000 instances per class is enforced by taking a random sample when this limit is exceeded. Table 2.1 shows the result of these balancing and subsampling operations. The situation prior is shown between brackets.

It is worth noting that Sabatelli et al. does not perform these operations. They are performed here, because the current study is interested in small datasets specifically.

All datasets are split up into 80%, 10%, 10% training, validating, and testing sets. For robustness, five datasets are randomly generated per classification task, such that each experiment can be run five times. The average sample overlap between two of the five sets is shown in Table 2.1 as well.

[6]The Gini-coefficient is a measure of balance, with 0 meaning all classes have the same number of samples, and 1 meaning all samples belong to a single class. A uniform distribution has coefficient $\frac{1}{3}$.

Besides the tasks described above, Table 2.1 shows an experiment called *Scaling*. The goal of this experiment is to see how the findings of this paper hold up as dataset sizes are gradually shrunk. It does this by taking the top 15 classes of type classification, and scaling it 4 times by a factor $\sqrt[4]{\frac{1}{10}} \approx 0.56$. The 5 datasets produced in this manner are then respectively 100%, 56%, 32%, 18%, and 10% the size of the dataset shown in Table 2.1. Steps are taken to ensure that the distribution remains the same. Finally, also here 5 datasets are generated per scaling factor, leading to a total of 25 datasets. The same 80%, 10%, 10% split is used for all.

2.2 Models

Eight different neural network architectures are used, of which four are CNN-based and four VT-based. When multiple versions are available (i.e. 'base', 'small', 'tiny'), 'base'/'medium' ones are chosen.

To start off with CNNs, `ResNet50` (He et al., 2016) and `VGG19` (Simonyan & Zisserman, 2014) are chosen, as these were the best performing models in Sabatelli et al. for respectively FT and OTS learning. In addition, Matsoukas et al. (2021) used `ResNet50` too, while Zhou et al. (2021) used `ResNet101` and `ResNet152`. To also add some more recent CNNs, `ConvNext` (Liu et al., 2022) – a purely CNN-based model inspired by VTs' recent success – and `EfficientNetV2` (Tan & Le, 2021) are included.

For VTs, versions with a 16x16 patch size are chosen. The first model is the original VT (Dosovitskiy et al., 2020), which will be referred to as `ViT`. Next, `Swin` (Liu et al., 2021) is considered, as it showed promising performance in Zhou et al.. The final two are `DeiT` and `BeiT`. The small version of `DeiT` was used in Matsoukas et al. (2021), because it is similar in size to `ResNet50`, to which it was compared. In this thesis, however, also a base-sized version is chosen. This version does not include output for distillation learning, so no advantage is taken of that.

All VT-based models have a bit over 86 million parameters. `ConvNext` is of similar size, with 88.6 million parameters. `ResNet50` and `EfficientNetV2` are smaller, with 25.6 and 13.6

Task	# Samples	# Classes	Gini coefficient	Sample overlap
Type classification	9607 (77628)	30 (801)	0.466 (0.974)	0.686
Material classification	7788 (96583)	30 (136)	0.563 (0.980)	0.798
Artist classification	6530 (38296)	30 (8592)	0.236 (0.676)	1
Scaling experiment	7926 (77628)	15 (801)	0.300 (0.974)	-

Table 2.1: Overview of the used datasets. Values between brackets show the situation before balancing operations were performed. 'Sample overlap' gives the average overlap between 2 of the 5 randomly generated sets per task (i and j where $i \neq j$).

Figure 2.1: Example images from the *Rijksmuseum Challenge*. Each row depicts a different task.

million, respectively, while `VGG19` is the largest model, with 143.7 million parameters.

2.3 Hyperparameters and data augmentation

All images are resized to a 224 × 224 resolution. This is achieved by first scaling them to the desired size along the shortest axis (retaining aspect ratio), and then taking a center crop along the longer axis. In addition, all images are normalized to the RGB mean and standard deviation of ImageNet1K.

For all models, the linear classification layer is replaced to fit the new task. Cross-entropy loss uses the raw outputs of this layer (conform `PyTorch` documentation), together with one-hot encoded prediction targets. An early stop occurs after 10 epochs without improved loss on the validation set, and the model with the lowest loss is benchmarked on the testing set.

For OTS learning, the Adam optimizer (Kingma & Ba, 2014) is used with standard `PyTorch` parameters (`lr`=1e-3, $\beta_1 = 0.9$, $\beta_2 = 0.999$), and a batch size of 256. For FT, more is needed to achieve good convergence, especially for VT-based architectures. The learning scheme used here is partially inspired by Matsoukas et al. (2021) and Zhou et al. (2021). The Adam learning rate now starts off at 1e-4, and is reduced by a factor 10 after 3 epochs without improvement (i.e. patience=2); the batch size is reduced to 32; label smoothing of 0.1 is added to the cross-entropy loss, and a dropout layer with p=0.2 is inserted before the classification layer. Finally, input images are augmented with random horizontal flips and rotations in a ±10° range.

2.4 Hardware and software

All experiments are conducted on a single compute node containing one 32 GB Nvidia V100 GPU. These nodes are constituents of the *Peregrine high-performance computing cluster*, belonging to the *University of Groningen's Center for Information Technology*. The FT experiments take advantage of the V100's mixed precision acceleration. An exception is made for the type classification experiment, as this one is also used to compare OTS learning

and FT in terms of time/accuracy trade-offs. Preliminary findings suggested that using mixed precision does not affect performance for the aforementioned experiments, however.

The `PyTorch` machine learning framework is used (Paszke et al., 2019), and many pre-trained models are taken from its `Torchvision` library. Exceptions are `EfficientNetV2`, `Swin`, `DeiT`, and `BeiT`, which are taken from the `Timm` library[7].

Finally, all source code and data will be made available on `GitHub`[8].

3 Experiments

This section focuses on the results obtained from the conducted experiments. These will first briefly be presented in section 3.1. After that, section 3.2 will discuss them in more detail, and see how they compare to earlier studies.

3.1 Results

Results are presented in the form of tables and line plots. The former of these reports final testing performance, whereas the latter mostly visualizes validation accuracy during training.

To give a more precise description: tables show average testing performance over the 5 trials per experiment, with standard deviation (s) as a subscript. Besides accuracy, also balanced accuracy is reported. This measure is defined as the average recall over all classes, such that it penalizes errors on less occurring classes more than plain accuracy would. Lastly, green shaded cells mark best performance, while yellow and red mark second best and worst performance, respectively.

For line plots, shaded regions correspond to the *standard error of the mean* ($\pm s \div \sqrt{N}$, where $N = 5$). In cases where they show average validation accuracy, plots end when an early stop occurred for the first of 5 trials. To easily distinguish VTs from CNNs, VTs are plotted with continuous lines, and CNNs with dashed ones.

3.1.1 Off-the-shelf learning

Table 3.1 shows the testing performance of OTS-trained models on the 3 classification tasks, while Figure 3.1 shows corresponding validation accuracies. Performance on the testing set is comparable among all tasks in terms of rankings. Other observations are consistent as well, such as `ConvNext` taking the most epochs to converge.

VTs perform very much on par with CNNs, if not slightly better. The best-performing model, for instance, is the `Swin` VT, which shows the highest testing performance without exception. Moreover, all VTs except `BeiT` are positioned relatively high in the rankings. This is also reflected in their combined average accuracies being higher than those of CNNs. To give *Type classification* as an example: here VTs as a group achieve an 86.5% mean accuracy, while this is 85.5% for CNNs.

Finally, when comparing the 3 classification tasks to one another, it can be observed that the highest overall accuracies are achieved for *Artist classification*, and the lowest ones for *Material classification*. For balanced accuracy, the differences are more extreme, as this measure also falls behind plain accuracy the most for *Material classification*, and the least for *Artist classification*.

3.1.2 Fine-tuning

Once again, Table 3.2 shows testing performance, while Figure 3.2 shows validation accuracies.

Results are less favorable for VTs than they were after OTS learning in section 3.1.1. At the same time, it can still be said that VTs perform on par with CNNs. The `Swin` VT shows the best overall testing accuracy here as well, but the `ConvNext` CNN is not far behind, and outperforms `Swin` in terms of balanced accuracy. Also noteworthy is the `ResNet50` CNN, which is ranked higher here than it was for OTS learning, and now shows the second best testing performance for *Artist classification*.

VTs and CNNs grouped are much more evenly matched. To again take *Type classification* as an example: the mean testing accuracy of all VTs combined is 91.2% on this task, while it is 91.3% for CNNs. Recall that section 3.1.1 reported a differ-

[7] `https://timm.fast.ai/`.

[8] `https://github.com/IndoorAdventurer/ViTTransferLearningForArtClassification`.

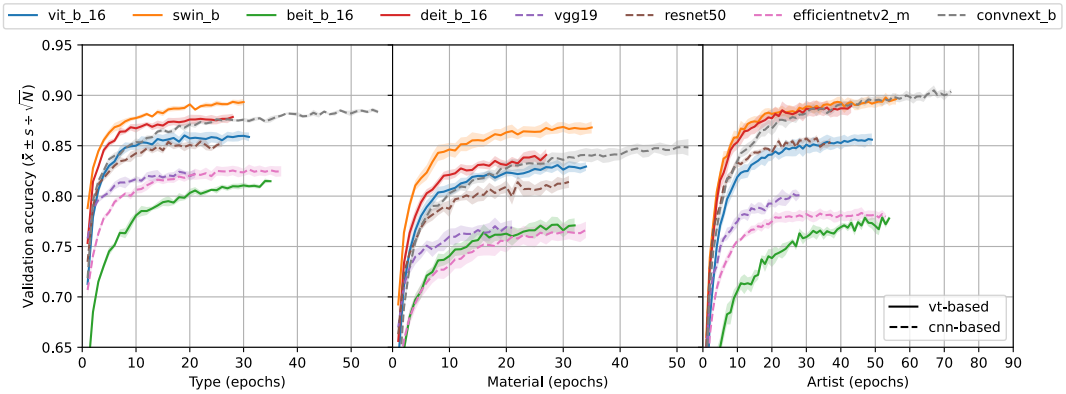


Figure 3.1: Average validation accuracy when training with an off-the-shelf learning scheme. It can be observed that the `Swin` VT achieves the highest performance overall, but that the `ConvNext` CNN is a close second. More general, VTs perform on par with CNNs, if not slightly better.

Model	Type		Material		Artist	
	Accuracy	Bal. accuracy	Accuracy	Bal. accuracy	Accuracy	Bal. accuracy
vit_b_16	86.06% ±1.06%	84.13% ±1.57%	81.78% ±0.48%	67.38% ±1.37%	84.89% ±0.46%	81.42% ±0.42%
swin_b	89.43% ±0.93%	87.47% ±1.02%	85.87% ±0.35%	71.19% ±1.60%	90.40% ±0.65%	88.64% ±0.78%
beit_b_16	82.26% ±0.72%	77.75% ±0.27%	76.87% ±0.96%	60.16% ±1.56%	79.70% ±0.69%	75.35% ±1.09%
deit_b_16	88.18% ±0.66%	85.36% ±0.54%	82.80% ±1.12%	66.46% ±1.03%	88.13% ±0.76%	85.62% ±0.87%
vgg19	83.93% ±0.72%	83.35% ±0.81%	76.87% ±0.44%	61.39% ±1.47%	82.01% ±0.66%	78.10% ±0.77%
resnet50	85.51% ±0.64%	82.33% ±1.85%	80.99% ±0.82%	65.51% ±0.93%	87.71% ±1.06%	85.12% ±1.34%
eff. netv2_m	83.41% ±0.76%	82.05% ±1.25%	75.96% ±1.24%	59.15% ±1.24%	78.62% ±1.07%	73.92% ±0.96%
convnext_b	89.19% ±0.64%	86.95% ±1.38%	84.14% ±0.92%	69.10% ±1.05%	90.13% ±0.94%	87.84% ±1.07%

Table 3.1: Testing performance after off-the-shelf learning. Results are similar to validation accuracy shown in Figure 3.1, with `Swin` and `ConvNext` showing the respective best and second best performance in all cases.

ence of 1% in favor of VTs here, with an accuracy of 86.5% for VTs, and 85.5% for CNNs.

More generally, it can be observed that FT leads to substantially better performance than OTS learning, and that models are now much closer together. The difference between the highest and lowest testing accuracy, for example, is at most 3.8% after FT, while it was between 7.2% and 11.8% after OTS learning. In addition, the worst testing accuracy after FT is most often still higher than the best accuracy after OTS learning. The only exception is *Material classification*, where the worst FT model (`BeiT`) has an accuracy of 85.74%, and the best OTS one (`Swin`) has 85.87%.

Lastly, note that the highest overall performances are again achieved on the *Artist classification* task, while the lowest ones are reported for *Material classification*. The difference between plain and balanced accuracy is also again the largest for *Material classification*, and lowest for *Artist classification*.

3.1.3 Scaling

Figure 3.3 shows how testing accuracy decreases when smaller portions of the full dataset are taken (see section 2.1). The x-axes show a logarithmic scale, where each successive value is roughly 56% the size of its predecessor.

For both OTS learning as FT, findings done in sections 3.1.1 and 3.1.2 seem to hold up as dataset sizes are shrunk. It is not true, for example, that at some point CNNs start to outperform VTs. In addition, `Swin` and `ConvNext` remain some of the best-performing models throughout.

Lastly, when going from largest to smallest dataset, the mean accuracy of all models taken together drops with 9.1% for OTS learning. For FT

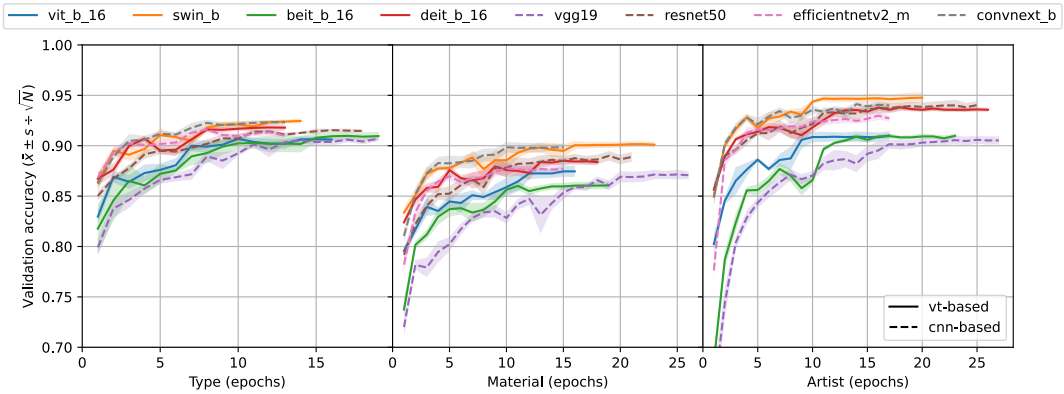


Figure 3.2: Average validation accuracy when fine-tuning models on the target task. Overall, performance is higher compared to off-the-shelf learning in Figure 3.1, while the differences between models are also smaller here. The `Swin` VT and `ConvNext` CNN remain the best performing models, while in general, results are less favorable for VTs than they were after off-the-shelf learning.

Model	Type		Material		Artist	
	Accuracy	Bal. accuracy	Accuracy	Bal. accuracy	Accuracy	Bal. accuracy
vit_b_16	90.11% ±0.35%	87.40% ±0.29%	87.42% ±0.45%	73.46% ±1.44%	92.05% ±0.44%	89.77% ±0.38%
swin_b	92.17% ±0.98%	89.71% ±1.03%	89.35% ±0.68%	77.16% ±2.98%	95.05% ±0.47%	93.94% ±0.81%
beit_b_16	90.81% ±0.41%	87.95% ±0.69%	85.74% ±0.37%	72.12% ±1.38%	91.27% ±1.13%	88.83% ±1.63%
deit_b_16	91.78% ±0.64%	89.22% ±0.90%	87.85% ±1.12%	74.42% ±1.99%	93.37% ±1.15%	91.67% ±1.55%
vgg19	90.54% ±0.37%	87.05% ±1.03%	85.74% ±1.40%	72.43% ±3.03%	92.20% ±0.49%	90.18% ±0.72%
resnet50	91.78% ±0.44%	88.24% ±0.59%	88.69% ±0.99%	77.97% ±2.25%	94.72% ±0.74%	93.41% ±1.05%
eff. netv2_m	90.87% ±0.67%	88.34% ±1.37%	87.55% ±1.15%	75.31% ±1.60%	92.65% ±0.54%	90.84% ±0.51%
convnext_b	92.15% ±0.40%	89.82% ±1.18%	88.79% ±1.07%	78.40% ±1.26%	94.60% ±0.54%	93.13% ±0.61%

Table 3.2: Testing performance after fine-tuning. Results are again similar to validation accuracy in Figure 3.2. Compared to Table 3.1, it is striking that the `ConvNext` CNN now often takes the lead with respect to balanced accuracy, and that the `ResNet50` CNN often takes second place.

this is 9.2%, which is very similar. This, then, does not suggest that at some point OTS learning becomes preferred due to overfitting problems for FT.

3.2 Discussion

While most results have now been presented, it is still important to discuss them in more detail, as this leads to a more complete interpretation. This subsection attempts to provide such a discussion, by answering questions one might have at this moment. It will first address some of the aforementioned experiments individually, but will conclude with more general remarks. Topics will include: comparisons with earlier studies, potential shortcomings, and hypotheses that might explain certain observations.

3.2.1 Off-the-shelf learning

How do these results compare to related studies? Of all related work mentioned in section 1.4, Zhou et al. (2021) was the only one that examined VTs' OTS learning properties. It reported the best OTS learning performance for the `Swin` VT, which is much in line with the results in Table 3.1.

Besides `Swin`, Zhou et al. also included `ViT` and `ResNet101` in its comparisons (among others), and consistently found that `ResNet101` outperformed `ViT`. The current study, on the other hand, uses `ResNet50`, which performs worse than `ViT` on all tasks except *Artist classification*. Running the same experiments with `ResNet101` does, however, lead to similar findings as Zhou et al.. On *Type classification*, for example, it achieves a mean testing accuracy of 86.21%(±0.70%), which is indeed slightly

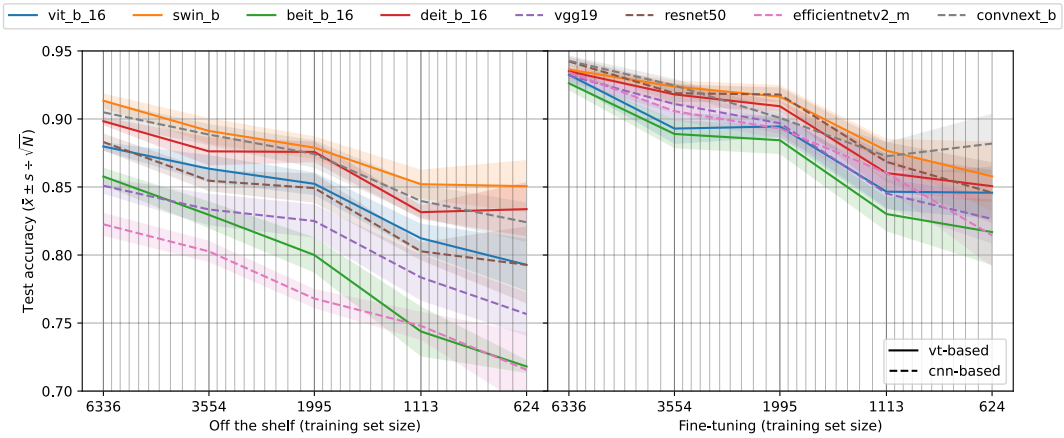


Figure 3.3: Testing accuracy as datasets gradually become smaller. The x-axes show a logarithmic scale, with the smallest value being roughly 10% the size of the largest one. Observations done in sections 3.1.1 and 3.1.2 appear to hold up well as the training set is shrunk.

higher than `ViT`'s 86.06% shown in Table 3.1. Mind that these differences are slight compared to the 89.43% accuracy achieved by `Swin` here, so replacing `ResNet50` with `ResNet101` would not have affected the overall observations that much.

Finally, because `VGG19` was specifically selected for its good OTS performance in Sabatelli et al. (2018), it is interesting that this model is outperformed by `ResNet50` (which that paper also included) in Table 3.1. `VGG19` is the largest model of all, so this discrepancy could be an indication that the smaller datasets used in the current study are too small for `VGG19`. Additionally, it might also be attributable to the smaller number of classes used here: `VGG19` has the largest final layer, containing 4096 inputs. While in Sabatelli et al. this might have provided the expressive power needed to differentiate between a few hundred classes after just OTS learning, here it likely causes overfitting before good convergence is reached.

Why do VTs perform so well here? While results are in line with previous studies, it was still surprising to see that VTs showed such good OTS learning performance in section 3.1.1. It suggests that these transformer-based architectures are able to give a very complete, high-level encoding of the input image. For CNNs this is commonly thought to be true, as they are often described as hierarchical feature extractors that build successively higher-level feature maps from lower-level ones (see section 1.1). If the same is true for VTs, is not well understood at this moment. Or at least to the best of the author's knowledge, that is.

A hint of an answer can, however, be found when plotting activation of successively deeper attention layers, as is done in Figure 3.4. While earlier layers retain much spatial information, later ones show more sporadic activation patterns. This is what one might expect from a hierarchical feature extractor: the random pixels lighting up might, for example, correspond to specific, high-level properties of the input. Nevertheless, it should be emphasized that no strong conclusions should be drawn from these images, and that more research with respect to explainability is desired.

3.2.2 Fine-tuning

How do these results compare to related studies? That the `Swin` VT performs well after FT, is again consistent with Zhou et al. (2021). In that paper, however, `ViT` outperforms the `ResNet101` and `ResNet152` CNNs. This differs from the results in Table 3.2, where `ViT` does not perform well overall, and even shows the worst testing accuracy on *Type classification*, with 90.11%. `ResNet50` achieves 91.78% accuracy here, and testing the same `ResNet101` and `ResNet152` versions that Zhou et al. used does not change much, as this

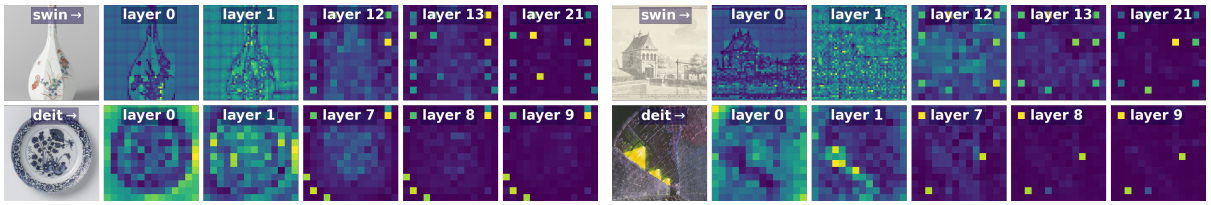


Figure 3.4: Attention layers of successively deeper transformer blocks. One can recognize the input image in earlier layers, while for later ones, activation seems sporadic. For `DeiT`, attention with respect to the class-token is plotted. `Swin`, on the other hand, does not have a class-token, so average overall attention is used instead. In addition, the window-shifting operations of this architecture are counteracted.

leads to 91.96% (± 0.59%) and 91.94% (± 0.25) accuracy, respectively.

Results are also less favorable for VTs compared to Matsoukas et al. (2021). In that paper, the 'tiny' version of `DeiT` outperforms `ResNet50`, while in Table 3.2 this same `ResNet50` performs slightly better than the 'base' version of `DeiT`. Going 'tiny' also does not help: on *Type classification*, for instance, 'tiny' `DeiT` gets 90.44% (± 0.93%), which is even worse than 'base' `DeiT`'s 91.78% in Table 3.1.

These slight performance differences could be attributable to the different datasets used. In that case, it seems, VTs have certain strengths that those other datasets take more advantage of. Alternatively, it could be that hyperparameters and regularisation still leave room for improvement. The reason for thinking this, is that preliminary experiments suggested that VTs benefit more from regularisation than CNNs do.

Finally, it was observed that `ResNet50` appears to flourish with FT, sometimes even beating `ConvNext` in the rankings. This is consistent with Sabatelli et al. (2018), where it was the best FT model overall.

Why are results less favorable for VTs after FT than after OTS learning? VTs and CNNs are much closer together after FT in Table 3.2 than after OTS learning in Table 3.1. It is as if OTS learning already unlocks much of VTs' potential, while CNNs need an FT learning scheme for this.

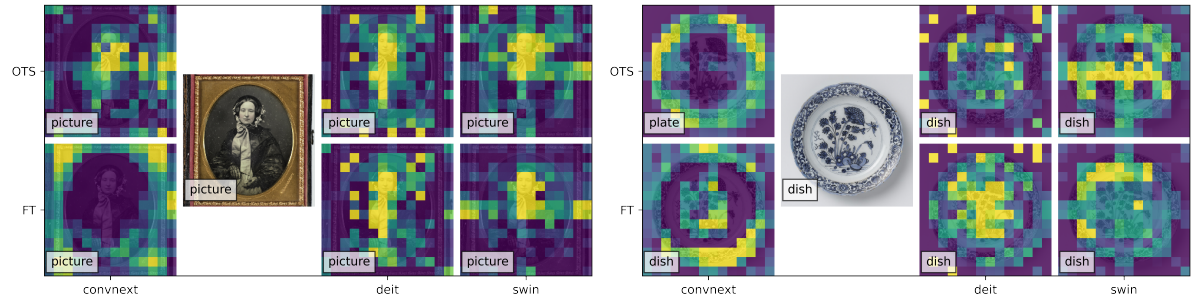
Supporting 'evidence' might be visible in Figure 3.5. It plots saliency maps for one CNN (`ConvNext`), and two VTs (`DeiT` and `Swin`). It is interesting that `ConvNext` shows much bigger differences between the OTS and FT plots. In Figure 3.5a, for example, it learns to just look at the picture frame after FT. In Figure 3.5b it learns to look more at the center, which allows it to correctly classify the input as a dish instead of a plate. These bigger differences for `ConvNext` align well with observations made earlier, as they suggest that there is still much improvement possible after just OTS learning for CNNs.

The images in Figure 3.5 are representative examples, and show observations that could have been made from other images too. How representative the 3 models are for all 8 is debatable, and was not tested. In general, a big disclaimer should be added to these results, because some hand waving is involved. Importantly, 3 different salience mapping techniques were used: GradCam (Selvaraju et al., 2017) was implemented for `ConvNext`, while Attention Rollout (Abnar & Zuidema, 2020) was implemented for `DeiT`. This already makes the comparison somewhat unfair, and it becomes even worse when `Swin` is added, for which a self-invented method was used. This method is similar to GradCam, but instead of multiplying *average* CNN channel gradients with said channel's activation, it directly multiplies activation with corresponding gradients in the output of one of the later transformer blocks[9]. Despite this, saliency maps were included, as they are interesting nonetheless.

3.2.3 Overall

Why does *Artist classification* show the best overall performance? This might seem counter-intuitive: one would think that it is much

[9]For the full implementation of all techniques, one is referred to the source code on: `https://github.com/IndoorAdventurer/ViTTransferLearningForArtClassification`.



(a) Picture **(b) Dish**

Figure 3.5: Saliency maps for `ConvNext`, `DeiT`, and `Swin` after both off-the-shelf learning (OTS), and fine-tuning (FT). `ConvNext` appears to benefit more when training the whole model instead of the last layer, because it shows the biggest differences between OTS learning and FT. In the case of Figure 3.5b, this even makes the difference between correct and incorrect classification.

easier to determine if something is a painting or a sculpture, than determining who the creator is. However, because only the top 30 classes are used, this is not necessarily true.

Figure 2.1 shows example images for all tasks. On the one hand, it shows much heterogeneity for the *Artist classification* task, with artists working in very different styles and mediums. On the other hand, it also shows difficulties for the other tasks: *Material classification*, which seems to be the most challenging task, shows examples of both big differences within classes (note how the two instances of porcelain are very different), and small differences between classes (note how the porcelain plate is similar to the faience plate).

Then there is the problem of balance. It was mentioned earlier that balanced accuracy falls behind plain accuracy the most for *Material classification*, and the least for *Artist classification*. At the same time, Table 2.1 shows that it is also *Material classification* which is the least balanced (Gini-coefficient of 0.563), while it is *Artist classification* that is the most balanced (coefficient 0.236). This suggests that balance has a positive effect on testing accuracy, from which *Artist classification*, in that case, benefits the most.

The same hyperparameters are used for all models. Is this fair? The final learning scheme was chosen after extensive hyperparameter optimization, and was partially inspired by related work. It is possible that doing this for each model individually would have resulted in *slightly* higher performances. This paper, however, does not aim to get the best possible performance, but instead wants to see how models compare to one another. It is unlikely that individually tuning each model would have changed overall findings.

A small asterisk to be made here, is that VTs might benefit more from better regularisation than CNNs (see section 3.2.2). Still, final accuracy should not be the only concern, because ease of use (i.e. effort needed to find the right hyperparameters) will also determine to what degree VTs are a valuable alternative to CNNs. Moreover, the main finding of this paper seems to be that VTs are able to perform on par with CNNs. Slightly better performance for VTs would not exactly hurt this finding.

To what extent are the chosen models representative of all VTs and CNNs? Most models were selected because of their mention in related work. Notable exceptions are `ConvNext` and `EfficientNetV2`, which were selected because they are equally recent as the used VTs. Many of these models are popular/common, but one can't say that they are a representative sample. The question, of course, also is not how VTs and CNNs as a group compare to one another. Instead, the question is if the VT landscape as a whole has something to offer that could be an interesting alterna-

tive to common CNNs.

It is also interesting to wonder if, for example, 'tiny'- or 'medium'-sized version should have been used instead of 'base' versions. All available 'tiny' models were tested as well, and results have been reported where appropriate (e.g. section 3.2.2). Overall, however, these results did not suggest that using 'tiny' versions would have made much of a difference for the rankings: 'tiny' `Swin` and `ConvNext`, for example, remained superior under that circumstance.

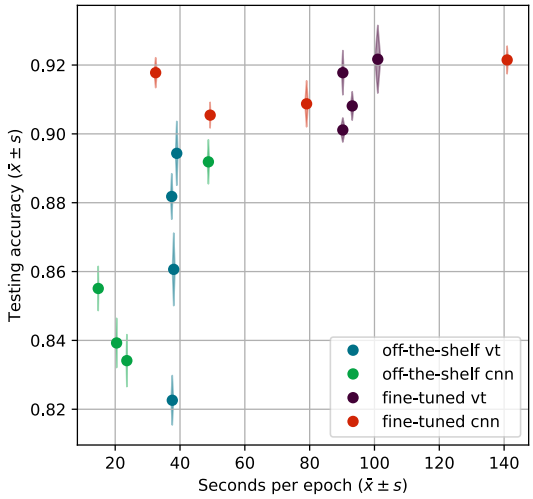


Figure 3.6: Time/accuracy trade-offs for VTs and CNNs either off-the-shelf or fine-tuned. Fine-tuned CNNs seem to make the best trade-off here, with one even appearing left above all off-the-shelf VTs. The best overall performance is achieved by a fine-tuned VT.

What can be concluded from the results presented? After OTS learning, results are slightly better for VTs, while after FT, results are almost the same for CNNs and VTs. With that said, it seems, VTs are very much able to perform on par with CNNs on the described art classification problems. Especially `Swin` and `DeiT` showed promising performance, and together with `ConvNext`, might be interesting models to consider.

Of course, final accuracy is not the full story, since time/accuracy trade-offs, for example, can also determine which model is preferred. Figure 3.6 plots the *Type classification* accuracies of Tables 3.1 and 3.2 against training time per epoch. Total training time was not chosen instead, as this measure depends much more on selected hyperparameters.

The best overall performing model is a fine-tuned VT (`Swin`). At the same time, it is perhaps a CNN that shows the best time/accuracy trade-offs, because it performs better than any OTS-trained model, while still taking less time per epoch than any OTS-trained VT. The particular CNN this concerns, is `ResNet50`. Recall that this model only had 25.6 million parameters, while most VTs had around 86 million.

All 'tiny' VTs are around the same size as `ResNet50`, and using these would have changed the situation completely. As such, taking Figure 3.6 with a grain of salt would be appropriate.

Still, it seemed fitting to conclude with this figure, as it neatly summarises all the main findings done in this section.

4 Conclusion

Seeing which is better, VTs or CNNs, has never been the goal of this paper. The underlying architecture is somewhat irrelevant, as one should always pick (or create) the model that fits the task best – regardless of whether it is a VT, or CNN. What this paper, then, was interested in, is whether VTs can become a valuable alternative to CNNs under circumstances in which:

- small datasets are used: specifically, small enough to have been hand-labeled by one person;
- only consumer-grade hardware is available.

In other words: this paper was interested in the type of machine learning one could try at home, and wanted to know if VTs could be made useful for it.

To that end, it tried to utilize transfer learning methods, as these might help overcome problems that arise from VTs' lack of an inductive bias. More concretely, models pre-trained on ImageNet1K were either used as off-the-shelf feature extractors, or fine-tuned to new tasks. These tasks

were all taken from the *Rijksmuseum Challenge* dataset (Mensink & van Gemert, 2014), and fall within the domain of art classification.

The results demonstrated that VTs are very well able to perform on par with CNNs here. Especially the `Swin` VT showed promising results, and might be worth considering for future learning problems.

`ConvNext` was a close second in the rankings. It is interesting that this best performing CNN was inspired by VTs, while the best performing VT, `Swin`, took inspiration from CNNs. It shows the kind of cross-fertilization that moves the field forward, but also hints at something bigger that would be nice to conclude with.

No matter what happens. No matter if VTs, CNNs, a hybrid architecture, or even something completely different rains supreme in the coming years; it is great that VTs came about. It shakes things up, and forces researchers to look at problems from a different perspective. Perhaps that, then, will be VTs' greatest contribution of all.

Acknowledgements

The author would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high-performance computing cluster. He would also like to thank dr. M. Sabatelli for his guidance during this project.

References

Abnar, S., & Zuidema, W. (2020). Quantifying attention flow in transformers. *arXiv preprint arXiv:2005.00928*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255).

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... others (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE* (Vol. 86, pp. 2278–2324).

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 10012–10022).

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. *arXiv preprint arXiv:2201.03545*.

Matsoukas, C., Haslum, J. F., Söderberg, M., & Smith, K. (2021). Is it time to replace cnns with transformers for medical images? *arXiv preprint arXiv:2108.09038*.

Mensink, T., & van Gemert, J. (2014). The Rijksmuseum Challenge: Museum-Centered Visual Recognition. In *ACM International Conference on Multimedia Retrieval (ICMR)*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc.

Sabatelli, M. (2022). *Contributions to deep transfer learning: from supervised to reinforcement learning* (Unpublished doctoral dissertation). Universitè de Liegè, Liegè, Belgique.

Sabatelli, M., Kestemont, M., Daelemans, W., & Geurts, P. (2018). Deep transfer learning for art classification problems. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops.*

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Sharif Razavian, A., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 806–813).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tan, M., & Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning* (pp. 10096–10106).

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., & Jégou, H. (2021). Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning* (Vol. 139, pp. 10347–10357).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... others (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations* (pp. 38–45).

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems* (Vol. 27). Curran Associates, Inc.

Zhou, H.-Y., Lu, C., Yang, S., & Yu, Y. (2021). Convnets vs. transformers: Whose visual representations are more transferable? In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 2230–2238).