Compact Transformers for Small-Scale Learning: A Reproducibility Study

Athanasios Charisoudis[*]
KTH Royal Institute of Technology
Stockholm, Sweden
`thacha@kth.se`

Simon Ekman von Huth[*]
KTH Royal Institute of Technology
Stockholm, Sweden
`simonevh@kth.se`

Emil Jansson[*]
KTH Royal Institute of Technology
Stockholm, Sweden
`emijan@kth.se`

Abstract

The Transformer model, proposed by Vaswani et al. [2017] has had prominent success on natural language processing tasks, inspiring efforts to adapt the architecture to other domains such as computer vision. Dosovitskiy et al. [2021] introduced the Vision Transformer (ViT), in which images are interpreted as sequences of 16x16 patches. While demonstrating state-of-the-art performance on image classification tasks, the ViT has an Achilles heel: the large amounts of data needed for pre-training. Recently, Hassani et al. [2021] presented three variants of the ViT aimed at democratizing its use: ViT-Lite, Compact Vision Transformer and Compact Convolution Transformer. This paper investigates these models in the light of data and compute efficiency. We run training benchmarks on CIFAR-10, including ablation studies on data augmentation and positional encoding. The results suggest that hybrid Transformer architectures involving convolutions are robust and learn more efficiently than ViTs when trained from scratch on small amounts of data.

1 Introduction

The introduction of the Transformer network has had a major impact within the field of natural language processing. Through the self-attention mechanism suggested by Vaswani et al. [2017], each encoding Transformer block has the ability to attend to all parts of the input sequence; greatly improving the modelling capacity compared to recurrent models. Following the success story in natural language processing, several attempts were made at applying the Transformer architecture to image data. The Vision Transformer (ViT) architecture, developed by Dosovitskiy et al. [2021], marked the first successful attempt at surpassing the performance of convolution neural networks (CNNs) in image classification tasks.

The ViT model was pre-trained on very large datasets before being transferred to various tasks through fine-tuning. The dependency on immense datasets and computational resources for pre-training sparked a plethora of concurrent efforts to make ViT more effective, often proposing hybrid architectures that combine convolutions with Transformer blocks; an effort towards *democratizing* ViT training and applicability. The idea behind the hybrid architecture is to utilize

[*]Authors are listed alphabetically. All authors contributed equally to this work.

Based on the NeurIPS 2022 publication template.

the locality and translational invariance inherent to convolutions and pooling while maintaining the Transformers' ability to attend to important parts of the image, regardless of the distance between them. In this way the set of assumptions implied by CNN models (i.e. their *inductive bias*) can be exploited by a Transformer.

This paper investigates Vision Transformers from the perspective of data and compute efficiency. The main concerns under question is: Can ViTs be scaled down to efficiently learn on little data? What is the effect of introducing convolutional layers to the original ViT? How do the inductive biases inherent to CNN's affect the data requirements and training stability of a Transformer-based model? In order to address these questions three variants of the ViT, as proposed by Hassani et al. [2021], are implemented and compared to the original ViT when training from scratch on a small dataset. Additionally, ablation studies are performed on data augmentation and positional encoding, in order to measure its effect on the generalizability and stability of the trained models.

The results suggest that ViT models can be trained from scratch on small datasets if the patch size is decreased or if the token embeddings are generated from a CNN. Neither the positional embedding nor the class token components are vital and could be removed or replaced by sequence pooling respectively. Data Augmentation is shown to increase the generalization performance significantly. Our best model managed to achieve over 91% on CIFAR-10 after less than 2 hours of training on a GTX980Ti.

1.1 Related Work

The most successful models for image classification have historically been deep CNNs such as ResNet introduced by He et al. [2016]. ResNet marked an important shift within deep learning, moving from learning unreferenced functions, to learning residual ones with respect to the input. This led to more stable gradients flow and thus allowed for deeper architectures. In recent years, a big success story around CNNs has been that of the *EfficientNet*. By rethinking model scaling rather than just making the architectures deeper, Tan and Le [2019] managed to achieve state-of-the-art performance with significantly fewer parameters than the previous best performing CNN architecture.

The Transformer architecture was first proposed by Vaswani et al. [2017] in the domain of sequence modelling. The Transformer architecture builds on the attention mechanism which had previously been used in recurrent encoder-decoder models (Bahdanau et al. [2014]). Self-attention was the novelty introduced with the Transformer model. By linearly transforming the input sequence to queries, keys and values, the self-attention mechanism attends to all parts of the input sequence in parallel. The first major step towards applying Transformers to the spatial domain came with the ViT, where an input image was split and rearranged into a sequence of patches. This sequence was then fed to a Transformer, to the end of which a classification head was attached. ViT models were heavily inspired by BERT Devlin et al. [2019], and similarly to them, they were really data-hungry architectures. This sparked numerous concurrent efforts aimed at democratizing the ViT model and making it more efficient.

In a recent work by Hassani et al. [2021], three versions of the ViT model were proposed. ViT-Lite is a scaled down version of the original ViT model, with fewer parameters and, possibly, smaller patch sizes. The Compact Vision Transformer (CVT) introduces a learnable *sequence pooling* operation after the final Transformer block, removing the need for a separate classification token (as introduced by Devlin et al. [2019]). The Compact Convolutional Transformer (CCT) replaces the initial patching operation with one or more convolutional blocks. All these models were shown to handily outperform the ViT-Base model when trained from scratch on small to medium sized datasets. Other similar efforts include those of Trockman and Kolter [2022] which presented ConvMixer, a simple yet well performing convolution-based model. In another line of work, Jeevan et al. [2022] achieved state-of-the-art performance by using a linear attention mechanism in their *XFormer* model. Hybrid models such as these have also been shown to allow more stable training with a wider range of optimizers, compared to the ViT (Xiao et al. [2021]).

1.2 Outline

Section 2 will cover the model architectures, training setup as well as the details of the data augmentation ablation study. In section 3 the results are presented, followed by a discussion in section 4. Finally, the work is summarized and concluded in section 5.

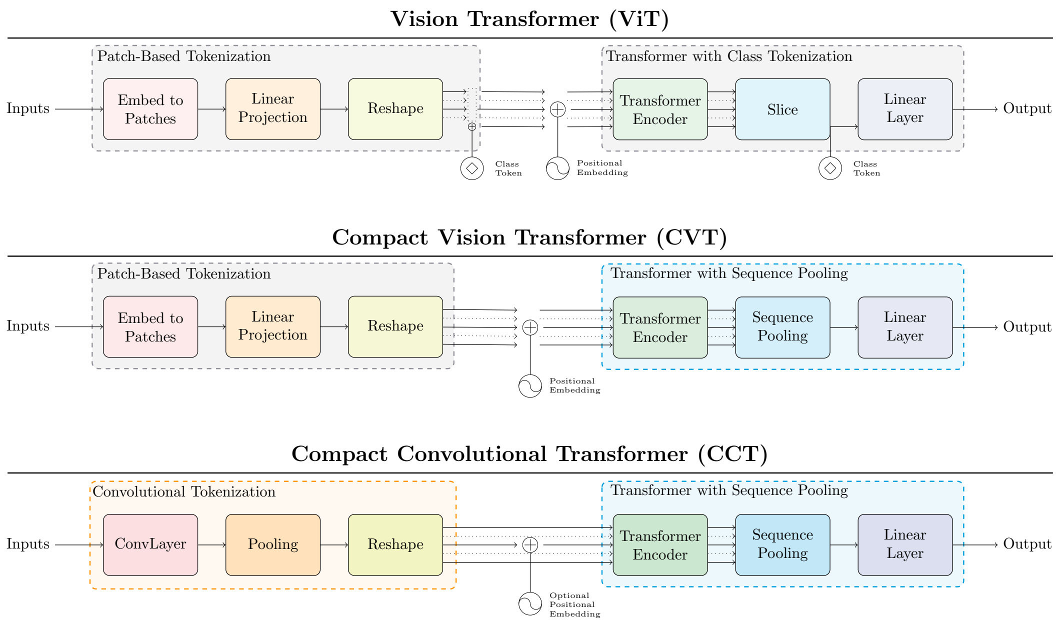

Figure 1: Comparison between ViT, CVT and CCT architectures from Hassani et al. [2021].

2 Methodology

We have implemented a selection of models from scratch using the PyTorch library and compared them with respect to classification performance.[*] The models include the two transformer architectures ViT and CVT, as well as the hybrid architecture CCT. Our implementations follow closely that of the original papers by Dosovitskiy et al. [2021] and Hassani et al. [2021]. The training scheme and hyper-parameters largely correspond to those of Hassani et al. [2021]. All models were trained and evaluated on the CIFAR-10 dataset; a small dataset (both in terms of number of images, 60K, and image size, 32x32) that at the same time contains a wide variety of natural images. Successfully applying Transformer networks to such a small dataset would extend the possible cases where Vision Transformers could be used, especially in scenarios where data and/or computational resources are limited.

2.1 Model Architectures

The architectures are outlined in Figure 1. In the following, the notation from Dosovitskiy et al. [2021] is broadly followed.

Transformers operate on sequences of data. Therefore images have to be converted to sequences before they can be fed into a transformer. As shown in Figure 1, the ViT architecture does so using *patch-based tokenization*, which means that the ViT reshapes the input image into a sequence of patches of size $P \times P$ and flattens each patch producing $\mathbf{x}^{i=1\ldots N} \in \mathbb{R}^{P^2C}$, where C is the number of channels in the image. Then it linearly projects each flattened patch into D dimensions using a learnable mapping $\mathbf{E} \in \mathbb{R}^{P^2C \times D}$. It concatenates all the projected patches, prepends a learnable class token $\mathbf{x}_{\text{class}}$ and adds 1D position embeddings $\mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$, producing a sequence $\mathbf{z}_0$:

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}^1\mathbf{E}; \ldots; \mathbf{x}^N\mathbf{E}] + \mathbf{E}_{pos} \tag{1}$$

The *transformer encoder* receives this sequence. The transformer encoder's key building block is *multi-head self-attention* (MSA), which is explained in Appendix B. The transformer encoder has L layers, where each layer consists of an MSA block and a regular multilayer perceptron (MLP) block

[*]`https://gits-15.sys.kth.se/thacha/cifar-vit`

Architecture	**# Layers** (L)	**# Heads** (k)	**Factor** (m_{MLP})	**Dim** (D)
ViT	12	12	4	768
ViT-Lite	7	4	2	256
CVT	7	4	2	256
CCT	7	4	2	256

Table 1: Key architectural hyperparameters of the Transformer network backbones.

separated by Layer normalization (LN) and residual connections:

$$\mathbf{z}'_l = \text{MSA}(\text{LN}(\mathbf{z}_{l-1})) + \mathbf{z}_{l-1} \qquad l = 1 \dots L \tag{2}$$

$$\mathbf{z}_l = \text{MLP}(\text{LN}(\mathbf{z}'_l)) + \mathbf{z}'_l \qquad l = 1 \dots L \tag{3}$$

The MLP block consists of two layers with a GELU (Hendrycks and Gimpel [2016]) activation after the first layer. The width of the first layer is $m_{\text{MLP}} \times D$ where m_{MLP} is a given factor, while the width of the second layer is D.

Finally, the output from the Transformer encoder is calculated by taking $\mathbf{z}^0_L$, the first element of the output sequence from the last transformer encoder layer, and feeding that into a single linear layer which outputs the predicted class label.

The CVT model removes the class token in favour of a sequence pooling operation which produces the input to the final linear classification layer. The sequence pooling operation takes $\mathbf{z}_L$, the output sequence from the last Transformer encoder layer, projects it using a learnable mapping $\mathbf{G} \in \mathbb{R}^{D\times 1}$, applies the softmax operator and then performs the dot product with $\mathbf{z}_L$ (in way like *attending to the sequential data*):

$$\text{SeqPool}(\mathbf{z}_\text{L}) = \text{softmax}\left(\mathbf{G}\left(\mathbf{z}_L\right)^\top\right) \times \mathbf{z}_L \quad \in \mathbb{R}^{1\times D} \tag{4}$$

The CCT model replaces the ViT's and CVT's *patch-based tokenization* with a *convolutional tokenization*, consisting of one or two convolutional blocks followed by flattening the resulting convolutional feature map into a sequence. Each convolutional block consists of a convolution operation followed by a ReLU activation and then a max pooling operation. If the model has one convolutional block, then the convolution operation has D filters. If the model has two convolutional blocks, then the convolution operation in the first block has 64 filters while the one in the second block has D filters. For the convolution operations, the kernel size is a hyperparameter that varies with the model, while the stride and the padding is always set to 1. For the max pool operation, the kernel size is 3, the stride is 2 and the padding is 1.

In addition, we stick to the sinusoidal position embedding presented by Vaswani et al. [2017], ablating on its importance on our trained CCT models. Its encoding is depicted in Figure 4 that follows. In Figure 5 the original *Patchify* operation used in ViT-like models is depicted.

The architectural hyperparameters of the models are detailed in Table 1. ViT-Lite uses the same architecture as ViT, just with different hyperparameters.

2.2 Training Setup

All models were trained on CIFAR-10 for 200 epochs with a batch size of 128. The learning rate was scheduled with cosine annealing, following 10 epochs of linear warm up. Weight decay was applied along with dropout in the transformer MLP ($p = 0.1$). The base learning rate and the weight decay used for the various architectures are specified in Table 2a. The training objective was to decrease the cross entropy loss between the predicted labels and the smoothed one-hot encoded label vector. As for data augmentation, random cropping and horizontal flipping ($p = 0.5$) were coupled with AutoAugment, as proposed by Cubuk et al. [2019]. AutoAugment applies various kinds of colorization, light manipulations and geometrical transformations according to policies which have been learnt specifically for CIFAR-10. Some experiments were run on Kaggle and GCP, and others on a local GTX980Ti machine.

2.3 Data Augmentation

Three different augmentation profiles were applied and compared in terms of test set classification accuracy (Table 2b). According to Hassani et al. [2021], CCT-7/3x2 (i.e. CCT model with 7 encoder layers and 2 convolutional blocks with 3×3 kernels) was the most efficient model when training from scratch on CIFAR-10. CCT-7/3x2 was thus chosen as the target model for the augmentation ablation study.

Hyper Param	**ViT**	**ViT-Lite**	**CVT**	**CCT**
LR	1e-4	5e-4	5e-4	5e-4
Weight Decay	0	3e-2	3e-2	3e-2

(a) Training hyperparameters.

Augmentation	**F**	**D**	**B**	**C**
Resize & Crop		✓		✓
Horizontal Flip		✓		✓
AutoAugment			✓	✓

(b) The data augmentations included in each augmentation profile.

Table 2: Training setup variability factors

The authors are using a more sophisticated data augmentation scheme also based on (modified version of) AutoAugment presented in Cubuk et al. [2019]. This is grounded in the fact that ViTs are large networks with enough capacity to easily overfit to such a small network. We show comparable results even though we use more conventional data augmentation schemes.

3 Results

Our models were trained from scratch on CIFAR-10 using data augmentation profile *C*, i.e. using *AutoAugment* combined with random horizontal flipping and random resized crop. The final test set accuracy scores are given in Table 3. The names of the models are formed as "*Architecture-Number of transformer layers* (L)*/Patch size or kernel size* (P)". For the CCT models, an extra "$\times$*Number of convolutional blocks*" is appended to the end. Thus, **CCT-7/3×2** denotes a CCT model with 7 transformer layers, kernel size 3 and 2 convolutional blocks. The table also includes information about network size and the results reported by Hassani et al. [2021] for the same models. Next, we ablated the data augmentation used for training as well as the importance of position embeddings in CCT models. Table 4 contains both the former and the latter study's results.

Model	**# Params**	**MACs**	**Our results**	**Results by Hassani et al. [2021]**
Vision Transformers				
ViT-12/16	85.63 M	0.43 G	72.59%	76.42%
ViT-Lite-7/16	3.89 M	0.02 G	73.42%	76.52%
ViT-Lite-7/8	3.74 M	0.06 G	79.66%	87.49%
ViT-Lite-7/4	3.72 M	0.24 G	87.99%	91.38%
Compact Vision Transformers				
CVT-7/8	3.74 M	0.06 G	80.33%	89.66%
CVT-7/4	3.72 M	0.24 G	87.70%	92.43%
Compact Convolutional Transformers				
CCT-7/3×2	3.85 M	0.28 G	**91.18**%	93.65%
CCT-7/3×1	3.76 M	0.95 G	88.73%	**94.47**%

Table 3: Top-1 test accuracy comparisons between the trained models on CIFAR-10 for 200 epochs.

4 Discussion

As can be seen in table 3, some of the models exhibit really satisfactory test set accuracy, surpassing the original ViT model by a large margin. This is consistent with the results reported by Hassani et al.

Model	PE	CIFAR-10			
		F	D	B	C
Compact Convolutional Transformers					
CCT-7/3×1	Sinusoidal				88.73% *(+0.91%)*
	None				87.82% (*baseline)*
CCT-7/3×2	Sinusoidal	82.95%	86.63%	89.20%	91.18% *(+0.22%)*
	None	81.49%	86.63%	88.25%	90.96% (*baseline)*

Table 4: Top-1 test accuracy comparison when changing the positional embedding method and/or the data augmentation scheme. The numbers reported are for training with the same training hyperparameters as in Table 3.

[2021]. Furthermore, the relative results across models are in general comparable to the reference results. The deviations in absolute accuracy can be mainly attributed to more sophisticated data augmentation schemes and the use of learnable position embeddings. A similar trend is observed as in Hassani et al. [2021], where subsequent models in the table achieve greater test accuracies. Simply scaling down the ViT-12/16 model to the ViT-Lite-7/16 yielded marginal performance gains, whereas decreasing the patch size had a significant impact. This is likely due to the smaller size of the input images, and confirms that indeed - *an image is not always worth 16x16 words*.

Removing the classification token in favour of sequence pooling did not substantially affect the classification accuracy in the CVT models. Furthermore, removing the positional encoding from the CCT seems to affect the performance but to a much lesser extent, as can be seen in Table 4. What seems to matter the most is the inductive biases that are introduced from spatially processing the input image using convolutional tokenization. Disclosure of learning curves for further comparison of training stability are deferred to Appendix A.

Data augmentation plays, unsurprisingly, a significant role in the CCT's ability to generalize to unseen samples, according to the results presented in Table 4. The introduction of basic augmentations (profile D) brought a moderate increase in performance of roughly 4%. Applying AutoAugment (profile C) resulted in another gain of 4%, emphasizing importance of augmentation for efficient use of the training data. In addition, the results from using only AutoAugment (profile B), shows that AutoAugment is the salient augmentation method with respect to performance gains.

5 Conclusion

In this work we investigated three variants of Vision Transformer networks, namely the ViT-Lite, CVT and CCT. ViT-Lite is a lighter version of the original ViT where, as was shown, reducing the patch size majorly contributed in boosting performance, even with a fraction of the number of network parameters. The results for the CVT model indicate that the class token can be replaced with a sequence pooling operation without affecting the performance. Finally, the introduction of convolutional layers at the beginning of the network yielded the CCT model; a so-called *hybrid* ViT architecture. Our results suggest that including convolutional layers in a transformer model leads to performance gains for image classification tasks. Another important conclusion is that comprehensive data augmentation is key to achieving the good performance demonstrated with the CCT. Finally, our findings indicate that ViT-Lite, CVT and CCT are efficient learners for small image data; comfortably outperforming former models with 20× the number of parameters.

References

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 113–123, 2019.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

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

A. Hassani, S. Walton, N. Shah, A. Abuduweili, J. Li, and H. Shi. Escaping the big data paradigm with compact transformers. *arXiv preprint arXiv:2104.05704*, 2021.

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

D. Hendrycks and K. Gimpel. Gaussian Error Linear Units (GELUs). *arXiv e-prints*, art. arXiv:1606.08415, June 2016.

P. Jeevan et al. Convolutional xformers for vision. *arXiv preprint arXiv:2201.10271*, 2022.

M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

A. Trockman and J. Z. Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

T. Xiao, M. Singh, E. Mintun, T. Darrell, P. Dollar, and R. Girshick. Early convolutions help transformers see better. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 30392–30400. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/ff1418e8cc993fe8abcfe3ce2003e5c5-Paper.pdf`.

A Training Stability

Figure 2 compares the learning curve for the CVT model which does not involve any convolutions, and the CCT model which uses convolutions during the tokenization instead of patching. The training becomes more stable and converges quicker. Figure 3 illustrates the learning curve for the down-scaled ViT model when using patch size of 16 and 4. As evident by the early difference in loss, the learning becomes more stable with smaller patch size.

B Multihead Self-Attention

Multihead self-attention (MSA) starts with a *self-attention* block. The self-attention block takes as input a sequence $\mathbf{z} \in \mathbb{R}^{N \times D}$, where N is the sequence length, which is variable, and D is the dimension of the sequence elements, which is fixed. First it projects the sequence elements to another dimension D_h using three learnable mappings, $\mathbf{U}_q \in \mathbb{R}^{D \times D_h}$, $\mathbf{U}_k \in \mathbb{R}^{D \times D_h}$ and $\mathbf{U}_v \in \mathbb{R}^{D \times D_h}$:

$$\mathbf{q} = \mathbf{z}\mathbf{U}_q, \quad \mathbf{k} = \mathbf{z}\mathbf{U}_k, \quad \mathbf{v} = \mathbf{z}\mathbf{U}_v \tag{5}$$

The resulting sequences $\mathbf{q}$, $\mathbf{k}$ and $\mathbf{v}$ are called queries, keys and values. Then *attention weights* $A \in \mathbb{R}^{N \times N}$ are calculated as a measure of pairwise similarity between the queries and the keys (i.e. between the projections of two sequence elements; called *dot-product attention*):

$$A = \mathrm{softmax}\left(\mathbf{q}\mathbf{k}^\top / \sqrt{D_h}\right) \tag{6}$$

Finally, the result of the self-attention is the sum of the values, weighted by the attention weights:

$$\mathrm{SA}(\mathbf{z}) = A\mathbf{v} \tag{7}$$

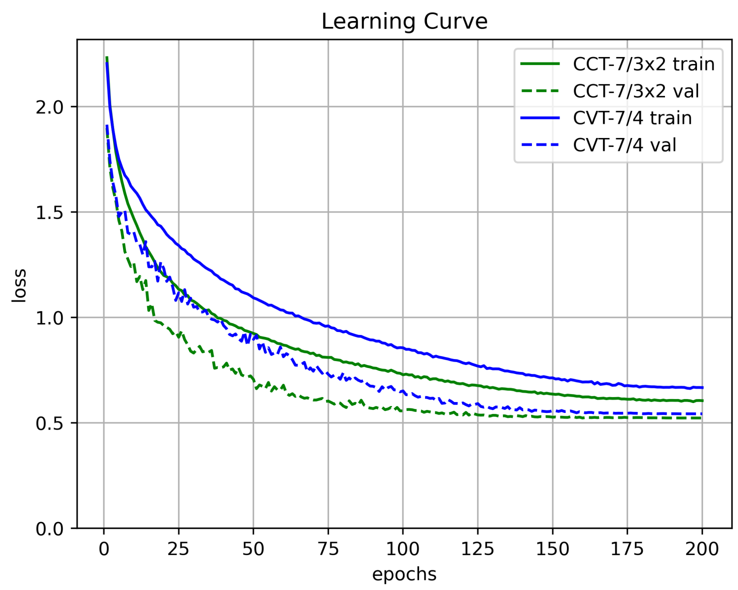

Figure 2: The learning curve for the CVT-7/4 and the best performing model (CCT-7/3x2).

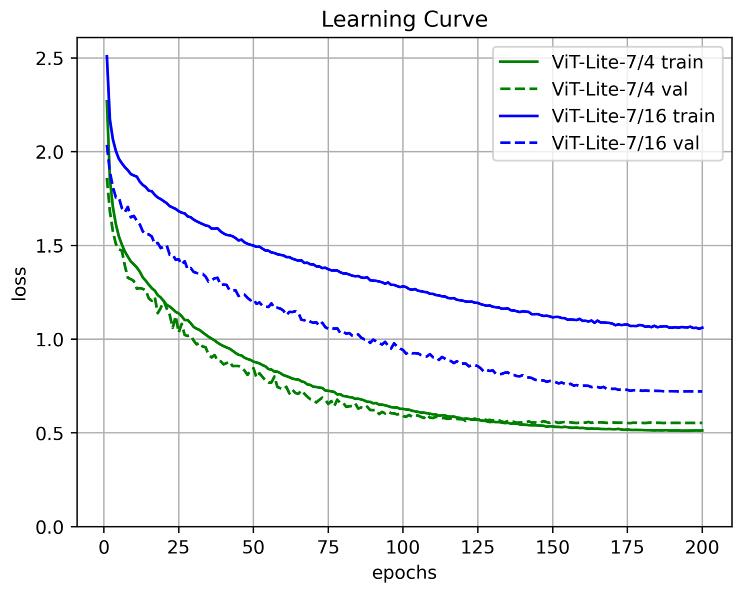

Figure 3: The learning curve for the ViT-Lite-7/16 and ViT-Lite-7/4.

Next, MSA consists of k self-attention blocks $\mathrm{SA}_{i=1..k}$ fed with the same sequence. Each individual self-attention block is called a *head*. The outputs of the individual blocks are concatenated and then projected using another learnable mapping, $\mathbf{U}_{msa} \in \mathbb{R}^{kD_h \times D}$:

$$\mathrm{MSA}(\mathbf{z}) = [\mathrm{SA}_1(\mathbf{z}); \ldots; \mathrm{SA}_k(\mathbf{z})]\, \mathbf{U}_{msa} \tag{8}$$

where in most cases (as in ours) $D_h = D/k$ to keep the number of parameters constant when changing the number of heads.

C Position Embedding and Patchify

In the figure that follows the sinusoidal embedding of the position is depicted, followed by a demonstration of the *Patchify* operation on a random image from CIFAR-10.

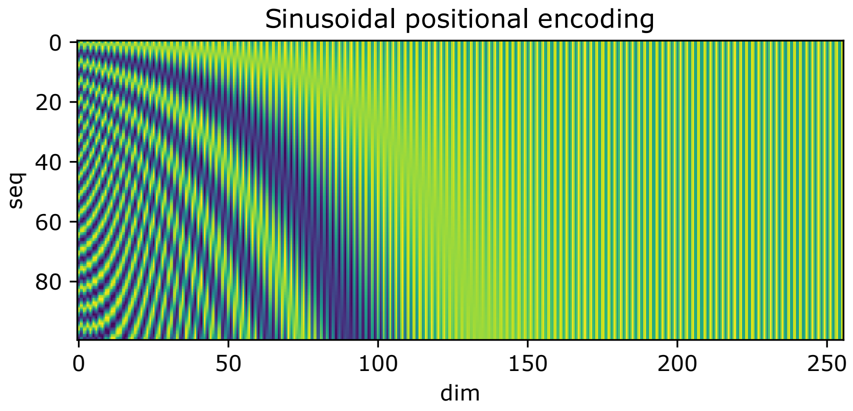

Figure 4: Sinusoidal position encoding

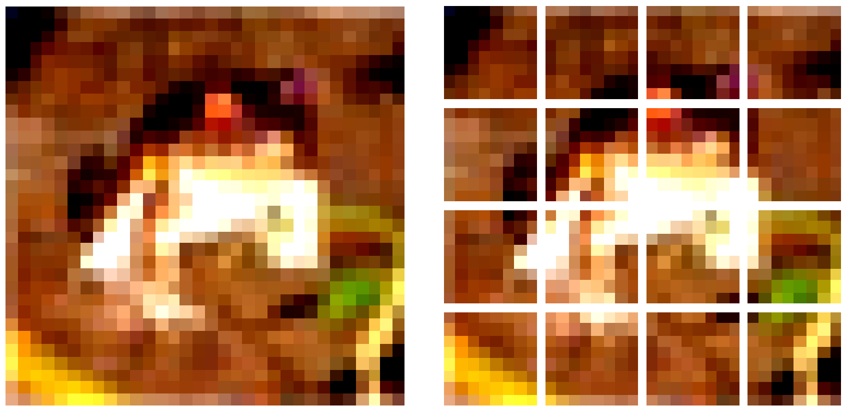

Figure 5: Demonstration of *Patchify* operation on a 32×32 image using 8×8 patches