Big Data Project Review -I

Mentor: Dr Srinivas Naik

Madhav Sharma 121AD0013

Shenbagsujan VS 121AD0011

Escaping the Big Data Paradigm with Compact Transformers

Authors: Ali Hassani, Steven Walton

Date: 7 June 2022

Journal :

Link : Click Here

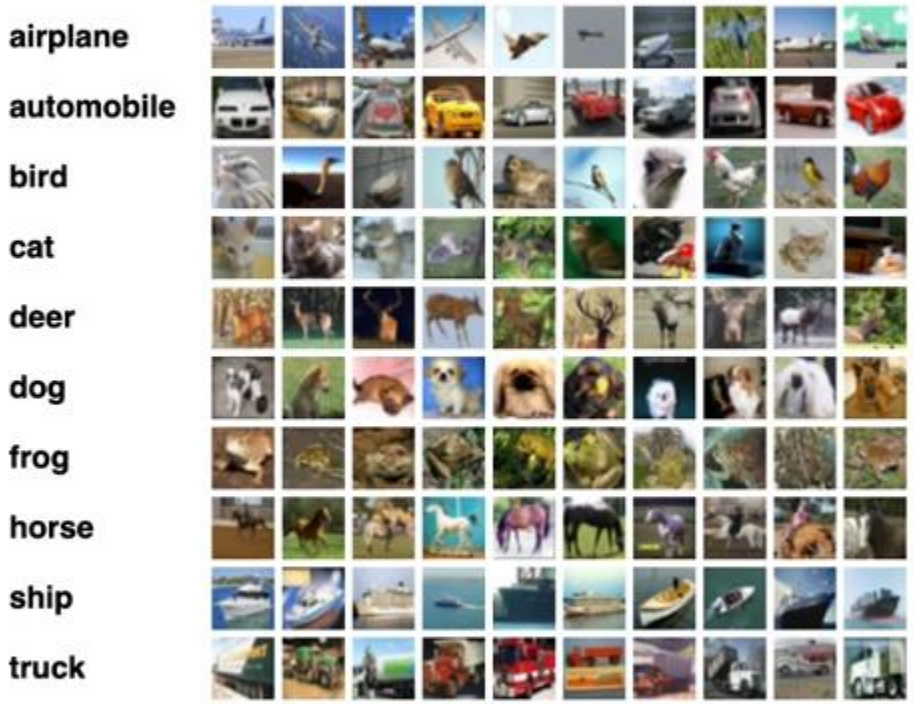
Methodology Overview

Problem Statement

- Transformers are very useful machine learning models but the problem is that they are very data thirsty and they need a lot of computational resources
- The research methodology describes architecture of a compact transformer which aims to do classification task by using limited computational resources

To Democratize AI Research for Transformers

- To help beginners with limited resources to play with transformers.
- And the less fortunate researchers without
  - > 1000 powerful TPUs
  - or huge datasets like JFT 300M
- And those who are dealing with important small dataset problems
  - Medicine
  - Science



Returning to the good old CIFAR-10 Dataset

Introduction

- Transformers are one of recent machine learning architectures which are extremely useful in Image Processing and Natural Language Processing[1]
- The transformers need a lot of computational resources hence the paper discusses the approach to design a compact transformer
- The architecture combines the approach of Swin Transformer (patching)[2] and CNN (convolution) to improve our accuracy main idea is given by Attention is all you need[3]
- The transformer architecture discussed is only used in classification tasks
- The training is done one CIFAR-10 and Flowers-102 datasets

Important Terms

- **Image Tokenization**
  - What: Converts images into a sequence of tokens.
  - How: Splits image into non-overlapping patches, flattens them into 1D vectors, and transforms into latent vectors.
  - Limitation: Potential loss of information at boundary regions.
- **Positional Embedding**
  - Purpose: Adds spatial information to the sequence.
  - Method: Uses learned embeddings or sine wave frequencies to encode positional relationships.
- **Transformer Encoder**
  - Structure: Series of stacked encoding layers.
  - Components: Each layer has Multi-Headed Self-Attention (MHSA) and Multi-Layer Perceptron (MLP) head, with layer normalization and residual connections.
- **Classification**
  - Process: Adds a [class] token to the sequence for representing the image class.
  - Function: The [class] token accumulates information through self-attention for classification.

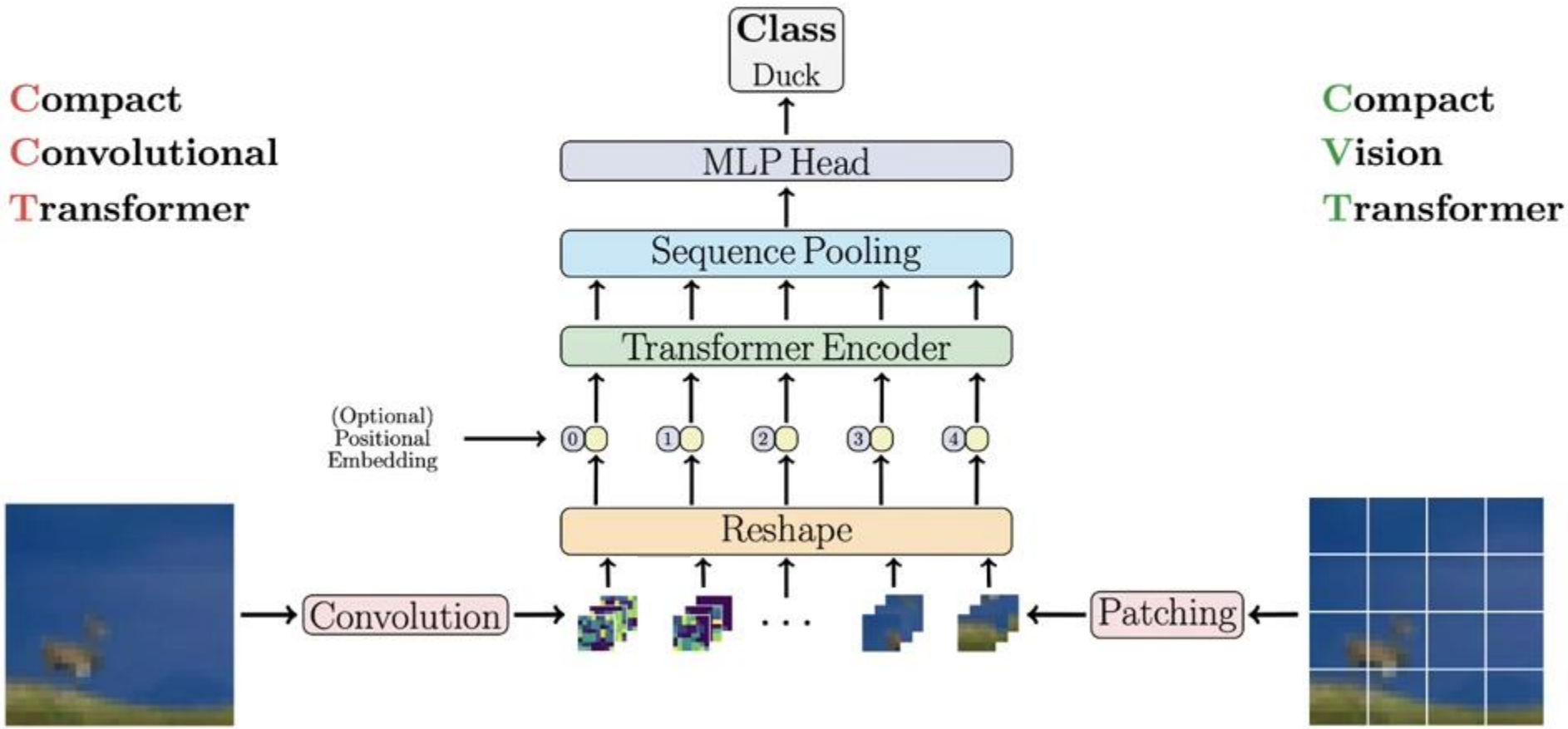
SeqPool

- **Traditional Approach:** ViT and other classifiers use a learnable class token passed through the network for classification.
- **Limitations:** Global average pooling is common but may not capture all relevant information.
- **SeqPool Introduction:** An attention-based pooling method that aggregates the output sequence of tokens to preserve valuable image information.
- **Mathematical Process:**
  - Transformation: $\mathbb{R}^{b\times n\times d} \mapsto \mathbb{R}^{b\times d}$.
  - Given $\mathbf{x}_L = \mathrm{f}(\mathbf{x}_0) \in \mathbb{R}^{b\times n\times d}$
  - Importance weighting: $\mathbf{x}'_L = \mathrm{softmax}\left(\mathrm{g}(\mathbf{x}_L)^T\right) \in \mathbb{R}^{b\times 1\times n}$
  - Output: $\mathbf{z} = \mathbf{x}'_L\mathbf{x}_L = \mathrm{softmax}\left(\mathrm{g}(\mathbf{x}_L)^T\right) \times \mathbf{x}_L \in \mathbb{R}^{b\times 1\times d}$
- **Advantages:**
  - No additional parameters needed compared to the class token.
  - Reduces computation by eliminating one token.
  - Learnable pooling outperforms static methods like global average pooling.
  - Impact: SeqPool allows for better utilization of information across spatially sparse data, leading to the creation of Compact Vision Transformer (CVT).

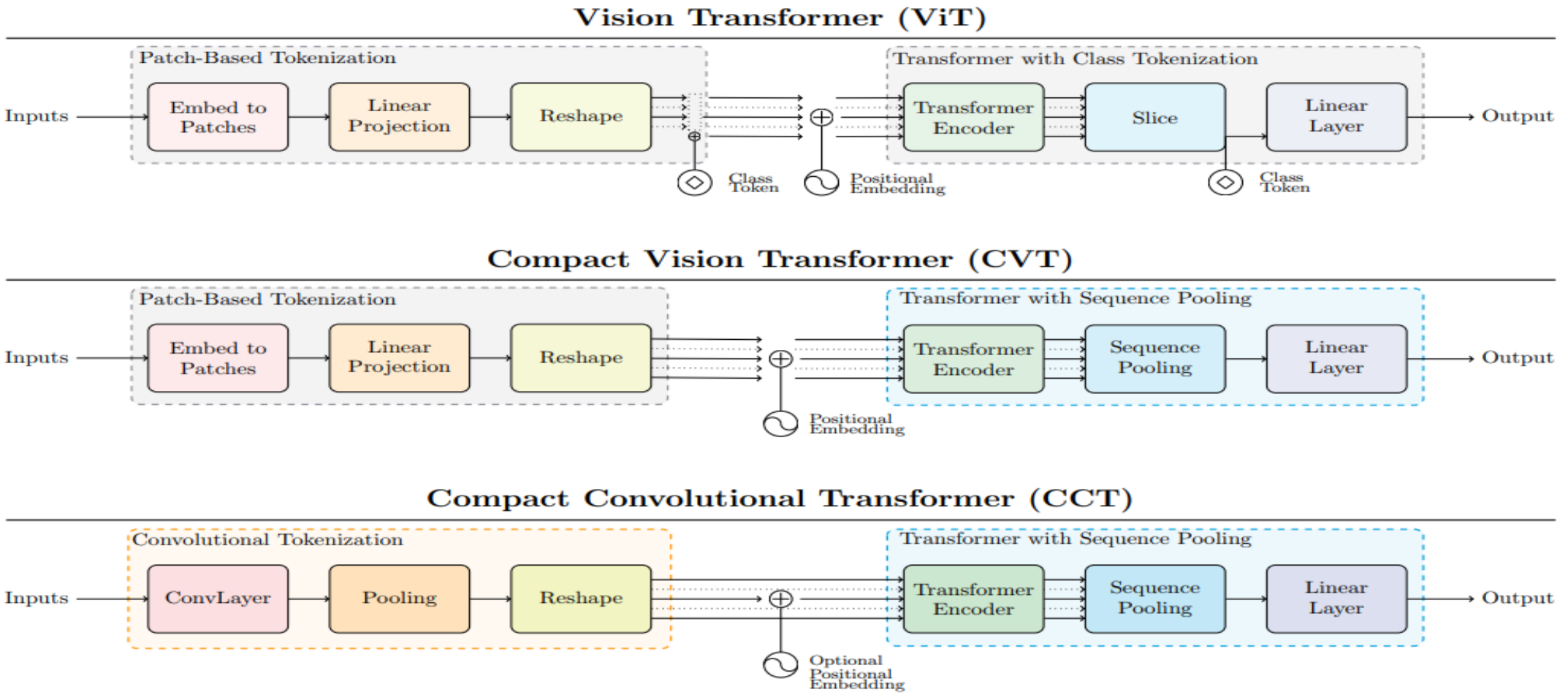
Real Life Example

- **Transformation:** Imagine you have a bunch of photos (tokens) from a trip, each representing a different place you visited. The transformer encoder is like a photo album that arranges these photos in order.
- **Importance Weighting:** Now, suppose you want to show a friend the essence of your trip in just one picture. You decide to create a collage that emphasizes the most memorable places more. In SeqPool, a linear layer acts like your judgment, deciding how much each photo (token) should contribute to the final collage (output). The softmax function then ensures that the importance scores add up to 1, just like making sure your collage fits in one frame.
- **Output:** You multiply each photo by its importance score and combine them all to create your collage. This final image represents your entire trip in one picture, ready to be shared.

Compact Transformers!

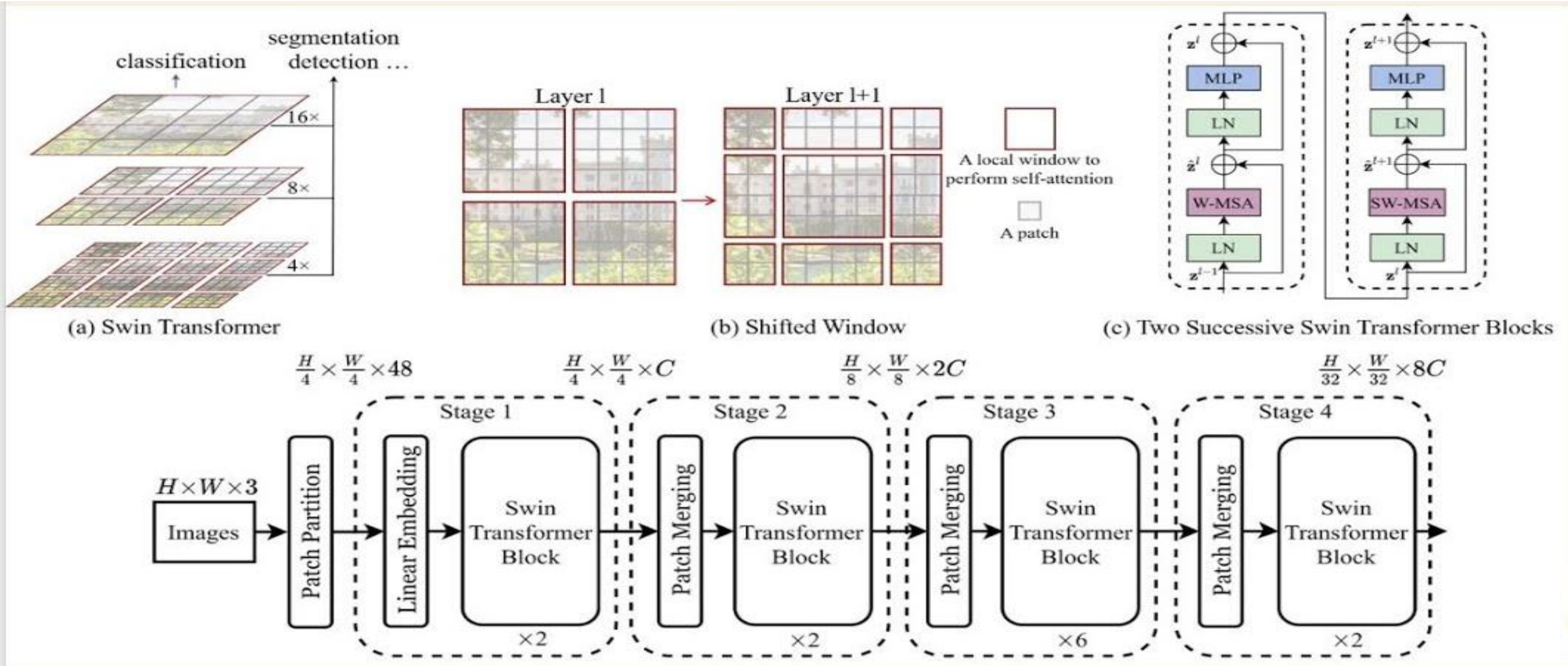


Three Versions



Literature Review

“Image is worth 16X16 words”



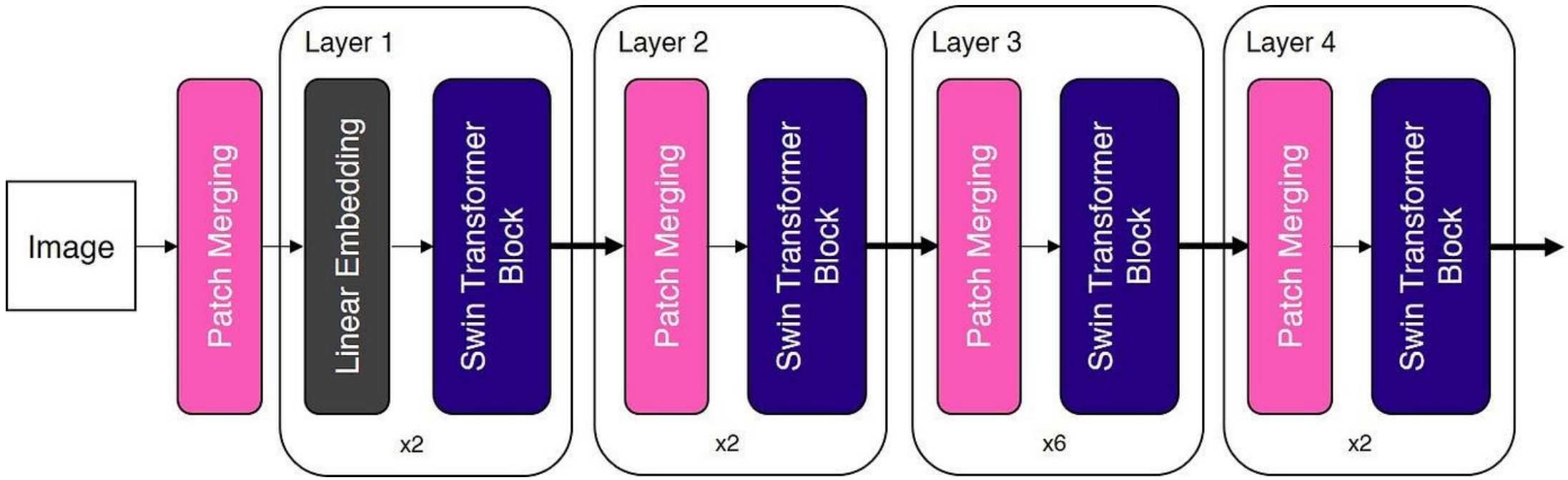
Introduction to Vision Transformers (ViT)

- **Background:** Traditional image recognition relies heavily on Convolutional Neural Networks (CNNs). However, the transformer architecture, successful in NLP, has potential in computer vision.
- **ViT Concept:** ViT applies the transformer directly to images by treating image patches as tokens, akin to words in NLP.
- **Image Tokenization:** Images are divided into fixed-size patches (e.g., 16x16), flattened, and linearly embedded. Positional embeddings are added to retain positional information.
- **Key Insight:** The paper demonstrates that pure transformers can achieve excellent results on image classification tasks without CNNs, especially when pre-trained on large datasets.

ViT Architecture and Performance

- **Architecture:** ViT uses a standard transformer encoder. It introduces a learnable [class] token for classification, whose state at the output of the transformer encoder is used for the final prediction.
- **Training and Results:** ViT, when pre-trained on large datasets like JFT-300M, outperforms state-of-the-art CNNs while being more resource-efficient.
- **Implications:** This approach challenges the necessity of CNNs for image tasks and opens up new avenues for using transformers in computer vision.

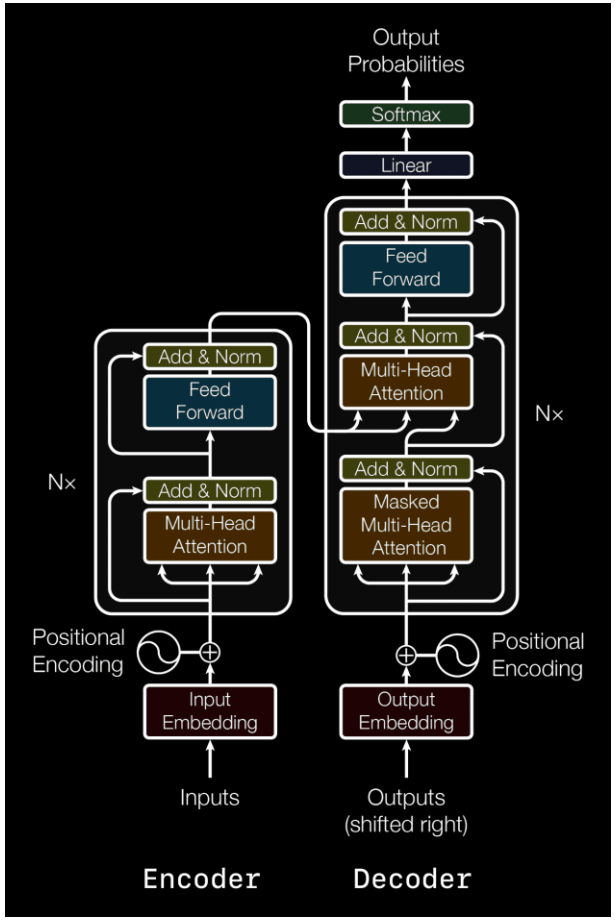
“Image is worth 16X16 words”



Sequence-to-Sequence Models

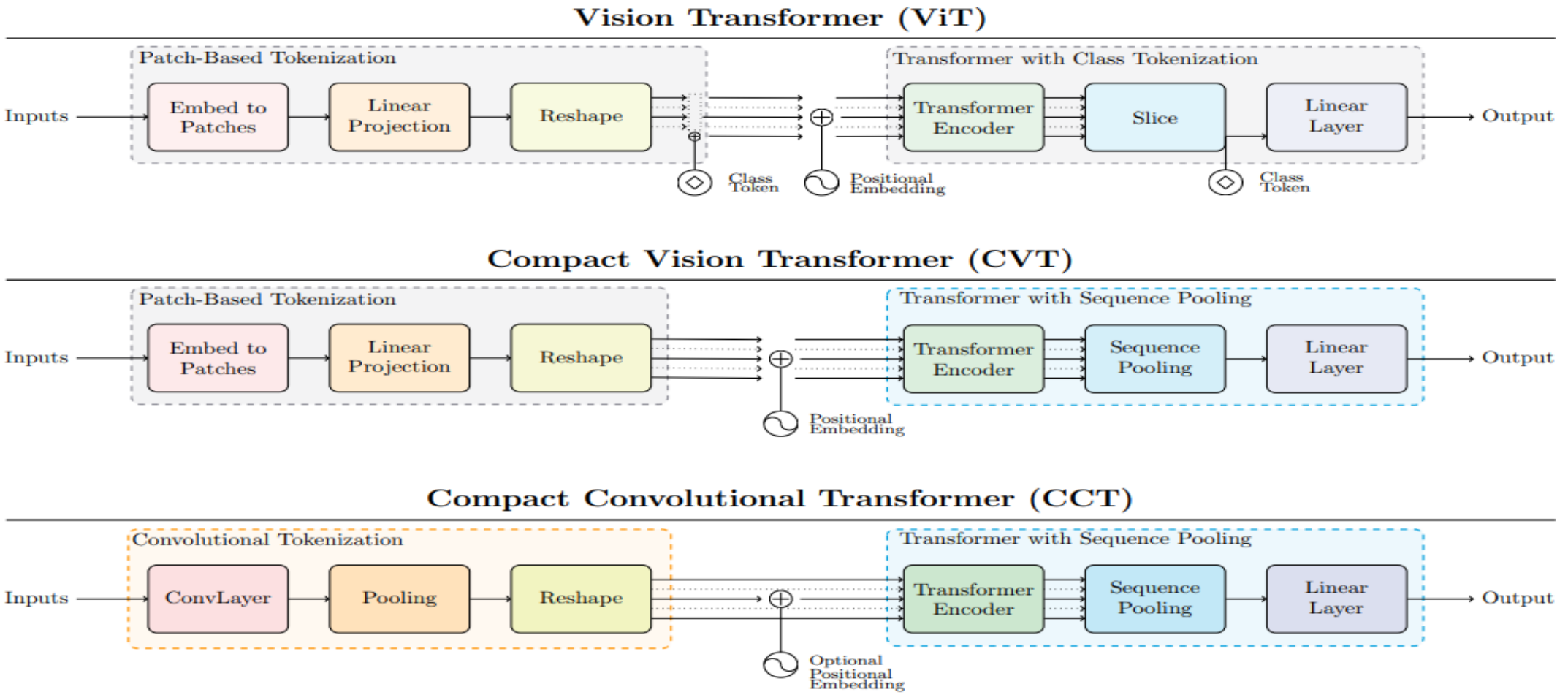
- **Innovation:** Introduced the Transformer model, revolutionizing NLP by replacing RNNs and CNNs with self-attention mechanisms.
- **Architecture:** Consists of an encoder and decoder, each with multiple layers of multi-head self-attention and position-wise feed-forward networks.
- **Efficiency:** More parallelizable and requires less training time compared to previous models.
- **Performance:** Achieved state-of-the-art results in machine translation tasks, proving the effectiveness of attention mechanisms.
- **Impact:** Paved the way for advancements in NLP, inspiring models like BERT and GPT.

“Attention is all you need”



Experiments and Results

Three Versions



Performance on CIFAR-10 : Training from scratch

Original ViT-Base	**CVT**	**CCT**
12 layers	6 layers	6 layers
Non-overlapping Convolutions 16x16	Non-overlapping Convolutions 4x4	Overlapping Convolutions 3x3
Class Token	SeqPool	Seqpool
69.82% Top-1	92.58% Top-1	94.81 % Top-1

Compact Transformers: ViT-Lite

Make it Smaller

- Let's try making it smaller? Not all images are 16x16 words.
- Less layers, less MLP heads, even smaller images!

You can even run these on a CPU!

Layers	# Heads	MLP Ratio	Params (M)	CIFAR-10
2	2	1	0.215	72.01
4	2	1	0.413	82.43
6	4	2	3.191	90.94
7	4	2	3.717	91.38

Compact Transformers: CVT

Make it Compact

- Making it smaller isn't enough!
- SeqPool instead of Class token

You can even run these on a CPU!

Layers	# Heads	MLP Ratio	Params (M)	CIFAR-10
2	2	1	0.215	79.02
4	2	1	0.413	86.15
6	4	2	3.191	92.58
7	4	2	3.717	92.43

Compact Transformers: CCT

Make it Convolutional

- Making it compact isn't enough!
- Add convolutions to the mix!

You can even run these on a CPU!

Layers	# Heads	MLP Ratio	Params (M)	CIFAR-10
2	2	1	0.284	89.17
4	2	1	0.482	91.45
6	4	2	3.327	94.81
7	4	2	3.760	94.78

Compact Transformers: Position Encoding

Thanks to the convolutional layers, CCT is less sensitive to the choice of PE.

Model	PE	CIFAR-10	CIFAR-100
Conventional Vision Transformers are more dependent on Positional Embedding			
ViT-12/16	Learnable	69.82% (+3.11%)	40.57% (+1.01%)
	Sinusoidal	69.03% (+2.32%)	39.48% (−0.08%)
	None	66.71% (*baseline*)	39.56% (*baseline*)
ViT-Lite-7/8	Learnable	83.38% (+7.25%)	55.69% (+7.15%)
	Sinusoidal	80.86% (+4.73%)	53.50% (+4.96%)
	None	76.13% (*baseline*)	48.54% (*baseline*)
CVT-7/8	Learnable	84.24% (+6.52%)	55.49% (+7.23%)
	Sinusoidal	80.84% (+3.12%)	50.82% (+2.56%)
	None	77.72% (*baseline*)	48.26% (*baseline*)
Compact Convolutional Transformers are less dependent on Positional Embedding			
CCT-7/7	Learnable	82.03% (+0.21%)	63.01% (+3.24%)
	Sinusoidal	81.15% (−0.67%)	60.40% (+0.63%)
	None	81.82% (*baseline*)	59.77% (*baseline*)
CCT-7/3×2	Learnable	90.69% (+1.67%)	65.88% (+2.82%)
	Sinusoidal	89.93% (+0.91%)	64.12% (+1.06%)
	None	89.02% (*baseline*)	63.06% (*baseline*)
CCT-7/3×2†	Learnable	93.65% (+0.86%)	74.77% (+1.34%)
	Sinusoidal	93.67% (+0.88%)	74.49% (+1.06%)
	None	92.79% (*baseline*)	73.43% (*baseline*)
CCT-7/7×1-noSeqPool	Learnable	82.41% (+0.12%)	62.61% (+3.31%)
	Sinusoidal	81.94% (−0.35%)	61.04% (+1.74%)
	None	82.29% (*baseline*)	59.30% (*baseline*)
CCT-7/3×2-noSeqPool	Learnable	90.41% (+1.49%)	66.57% (+1.4%)
	Sinusoidal	89.84% (+0.92%)	64.71% (−0.46%)
	None	88.92% (*baseline*)	65.17% (*baseline*)

Drawbacks and Proposed Methodology

Drawbacks

- The approach is only good when we have limited computational resources , we can implement this paper of high performance machine. The architecture can be implemented using Big Data approach using Spark
- The transformer only deals with classification we can implement semantic segmentation on medical imagery datasets means I2M(Image to model) [5]
- The paper only trains some limited datasets we can train brain tumor dataset by using U-Net[6]

Approach for Semantic Segmentation

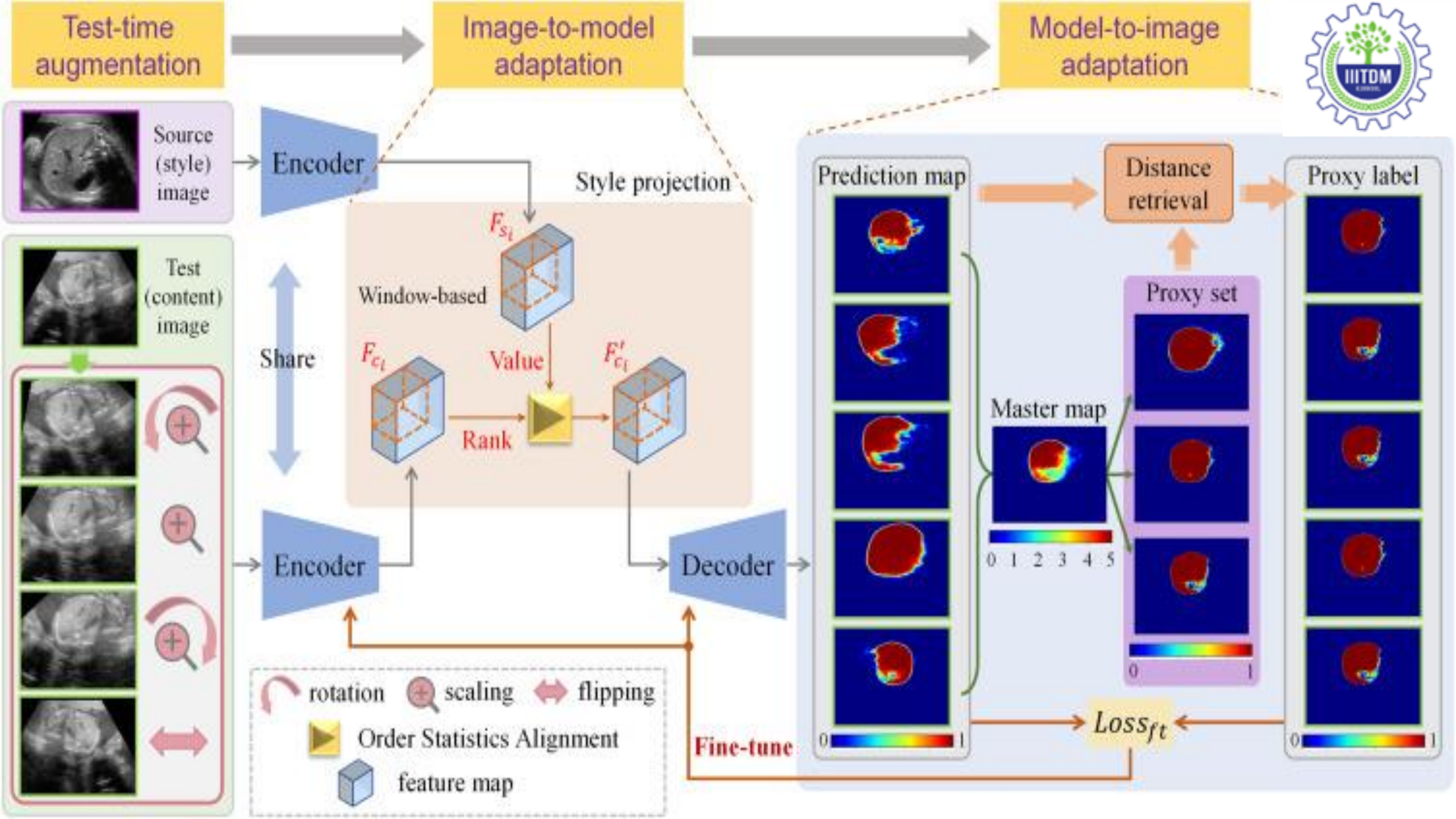
- The transformers can be trained of semantic segmentation as well
- The brain tumor dataset can be used for the above purpose
- The task is to develop a mask so as to point out exact region where mask is present
- Now our compact transformer is good for unsupervised semantic segmentation
- This process is called domain adaptation[4]

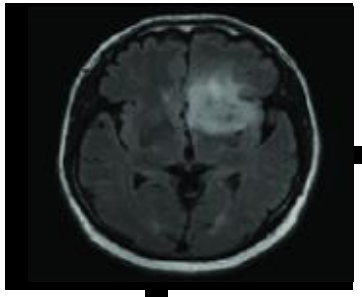
Detailed Approach

- **Image Tokenization:** The input image is tokenized into patches and embedded into a sequence of vectors, similar to how words are tokenized in NLP.
- **Positional Embedding:** Positional information is added to the sequence of image tokens to retain spatial relationships.
- **Transformer Encoder:** The sequence of embedded patches is passed through a transformer encoder, which consists of multiple layers of self-attention and feed-forward networks.
- **Semantic Segmentation (I2M):** The output of the transformer encoder is used to predict a segmentation mask, where each pixel is classified into a semantic category.
- **Reconstruction (M2I):** Optionally, a reconstruction process can be applied where the segmentation mask is used to generate a reconstructed version of the original image.
- **Comparison:** The original image and the reconstructed image are compared using metrics such as Intersection over Union (IoU) or pixel accuracy to evaluate the quality of the segmentation and reconstruction.

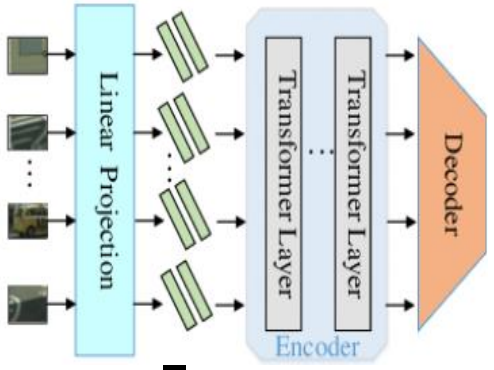
Approach for Semantic Segmentation

- Generating masks for brain tumor images is a tiresome process
- We will convert image to mask using transformer integrated with unet
- Then masked image to again reconstructed image
- The original and reconstructed image can be compared using Euclidian distances

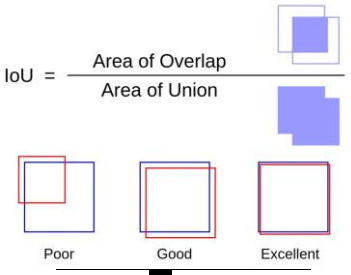

Test-time augmentation
Image-to-model adaptation
Model-to-image adaptation
Source (style) image
Encoder
Style projection
Window-based
Value
Rank
Share
Test (content) image
Encoder
Decoder
Prediction map
Distance retrieval
Proxy label
Proxy set
Master map
0 1 2 3 4 5
rotation
scaling
flipping
Order Statistics Alignment
feature map
Fine-tune
$Loss_{ft}$



Original Image

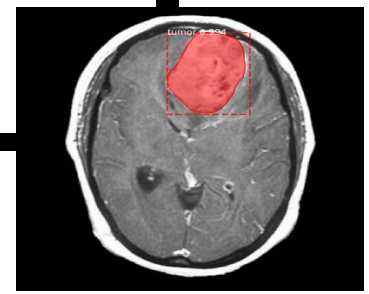


I2M

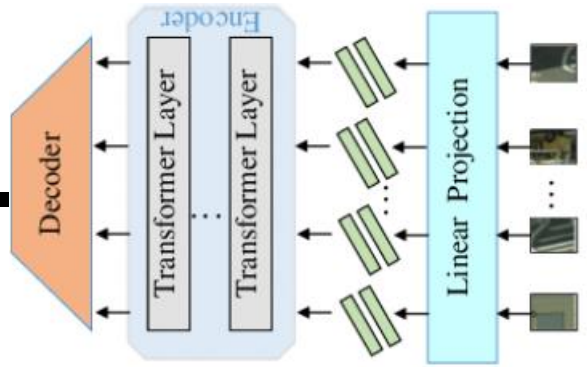


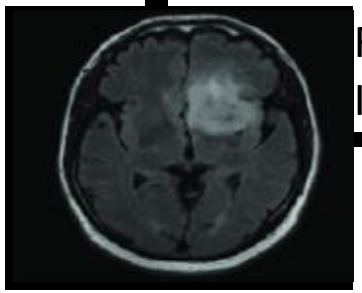
IoU

Masked Image



M2I





Reconstructed Image

Timeline

- Literature Survey : 30 February 2024 to 2 March 2024
- Algorithm Analysis
- Review – I : 5 March 2024
- Code and Algorithm Analysis : 4 March 2024 to 12 March 2024
- I2M and M2I Code Generation : 12 March 2024 to 20 March 2024
- Code integration and debugging : 20 March 2024 to 25 March 2024
- Presentation : 25 March 2024 to 30 March 2024
- Review – II : 1 April 2024

References

[1] Deep Learning by Aaron Courville, Ian Goodfellow, and Yoshua Bengio

[2] An Image is Worth 16x16 Words by Alexey Dudovskiy, Lucas Beyer

[3] Attention Is All You Need by Ashish Vaswani, Noam Shazeer

[4]Test-time bi-directional adaptation between image and model for robust segmentation Xiaoqing Huang, Xin Yang

[5] Segmenter: Transformer for Semantic Segmentation by Robin Strudel, Ricardo Garcia

[6] Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation Hu, Hu Cao, Yueyue Wang,

Thank You

Escaping the Big Data Paradigm with Compact Transformers

Ali Hassani[1]*, Steven Walton[1]*, Nikhil Shah[1],
Abulikemu Abuduweili[1], Jiachen Li[2,1], Humphrey Shi[1,2,3]
[1]SHI Lab @ University of Oregon, [2]University of Illinois at Urbana-Champaign, [3]Picsart AI Research (PAIR)

Abstract

With the rise of Transformers as the standard for language processing, and their advancements in computer vision, there has been a corresponding growth in parameter size and amounts of training data. Many have come to believe that because of this, transformers are not suitable for small sets of data. This trend leads to concerns such as: limited availability of data in certain scientific domains and the exclusion of those with limited resource from research in the field. In this paper, we aim to present an approach for small-scale learning by introducing Compact Transformers. We show for the first time that with the right size, convolutional tokenization, transformers can avoid overfitting and outperform state-of-the-art CNNs on small datasets. Our models are flexible in terms of model size, and can have as little as 0.28M parameters while achieving competitive results. Our best model can reach 98% accuracy when training from scratch on CIFAR-10 with only 3.7M parameters, which is a significant improvement in data-efficiency over previous Transformer based models being over 10x smaller than other transformers and is 15% the size of ResNet50 while achieving similar performance. CCT also outperforms many modern CNN based approaches, and even some recent NAS-based approaches. Additionally, we obtain a new SOTA result on Flowers-102 with 99.76% top-1 accuracy, and improve upon the existing baseline on ImageNet (82.71% accuracy with 29% as many parameters as ViT), as well as NLP tasks. Our simple and compact design for transformers makes them more feasible to study for those with limited computing resources and/or dealing with small datasets, while extending existing research efforts in data efficient transformers.

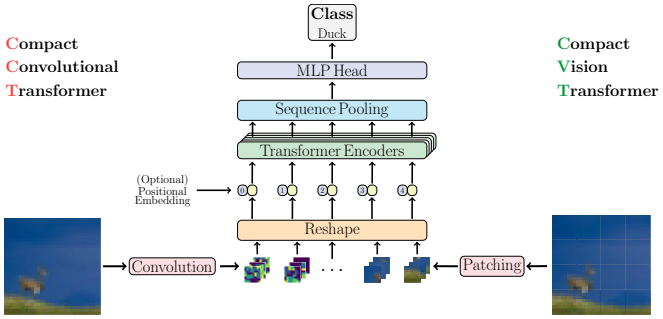


Figure 1: Overview of **CVT** (right), the basic compact transformer, and **CCT** (left), the convolutional variant of our compact transformer models. CCT can be quickly trained from scratch on small datasets, while achieving high accuracy (in under 30 minutes one can get 90% on an NVIDIA 2080Ti GPU or 80% on an AMD 5900X CPU on CIFAR-10 dataset).

1. Introduction

Convolutional neural networks (CNNs) [23] have been the standard for computer vision, since the success of AlexNet [22]. Krizhevsky *et al*. showed that convolutions are adept at vision based problems due to their invariance to spatial translations as well as having low relational inductive bias. He *et al*. [16] extended this work by introducing residual connections, allowing for significantly deeper models to perform efficiently. Convolutions leverage three important concepts that lead to their efficiency: *sparse interaction*, *weight sharing*, and *equivariant representations* [14]. Translational equivariance and invariance are properties of the convolutions and pooling layers, respectively [14, 36]. They allow CNNs to leverage natural image statistics and subsequently allow models to have higher sampling efficiency [34, 34].

*Equal contribution. Our code and pre-trained models are publicly available at https://github.com/SHI-Labs/Compact-Transformers

On the other end of the spectrum, Transformers have become increasingly popular and a major focus of modern machine learning research. Since the advent of Attention is All You Need [41], the research community saw a spike in transformer-based and attention-based research. While this work originated in natural language processing, these models have been applied to other fields, such as computer vision. Vision Transformer (ViT) [12] was the first major demonstration of a pure transformer backbone being applied to computer vision tasks. ViT highlights not only the power of such models, but also that large-scale training can trump inductive biases. The authors argued that "*Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data.*" Over the past few years, an explosion in model sizes and datasets has also become noticeable which has led to a "data hungry" paradigm, making training transformers from scratch seem intractable for many types of pressing problems, where there are typically several orders of magnitude less data. It also limits major contributions in the research to those with vast computational resources.

As a result, CNNs are still the go-to models for smaller datasets because they are more efficient, both computationally and in terms of memory, when compared to transformers. Additionally, local inductive bias shows to be more important in smaller images. They require less time and data to train while also requiring a lower number of parameters to accurately fit data. However, they do not enjoy the long range interdependence that attention mechanisms in transformers provide. Reducing machine learning's dependence on large sums of data is important, as many domains, such as science and medicine, would hardly have datasets the size of ImageNet [10]. This is because events are far more rare and it would be more difficult to properly assign labels, let alone create a set of data which has low bias and is appropriate for conventional neural networks. In medical research, for instance, it may be difficult to compile positive samples of images for a rare disease without other correlating factors, such as medical equipment being attached to patients who are actively being treated. Additionally, for a sufficiently rare disease there may only be a few thousand images for positive samples, which is typically not enough to train a network with good statistical prediction unless it can sufficiently be pre-trained on data with similar attributes. This inability to handle smaller datasets has impacted the scientific community where they are much more limited in the models and tools that they are able to explore. Frequently, problems in scientific domains have little in common with domains of pre-trained models and when domains are sufficiently distinct pre-training can have little to no effect on the performance within a new domain [54]. In addition, it has been shown that strong performance on ImageNet does not necessarily result in equally strong performance in other domains, such as medicine [20]. Furthermore, the requisite of large data results in a requisite of large computational resources and this prevents many researchers from being able to provide insight. This not only limits the ability to apply models in different domains, but also limits its reproducibility. Verification of state of the art machine learning algorithms should not be limited to those with large infrastructures and computational resources.

The above concerns motivated our efforts to build more efficient models that can be effective in less data intensive domains and allow for training on datasets that are orders of magnitude smaller than those conventionally seen in computer vision and natural language processing (NLP) problems. Both Transformers and CNNs have highly desirable qualities for statistical inference and prediction, but each comes with their own costs. In this work, we try to bridge the gap between these two architectures and develop an architecture that can both attend to important features within images, while also being spatially invariant, where we have sparse interactions and weight sharing. This allows for a Transformer based model to be trained from scratch on small datasets like CIFAR-10 and CIFAR-100, providing competitive results with fewer parameters and low computational requirements.

In this paper we introduce ViT-Lite, a smaller and more compact version of ViT, which can obtain over 90% accuracy on CIFAR-10. We expand on ViT-Lite by introducing a sequence pooling and forming the Compact Vision Transformer (CVT). We further iterate by adding convolutional blocks to the tokenization step and thus creating the Compact Convolutional Transformer (CCT). Both of these simple additions add to significant increases in performance, leading to a top-1%accuracy of 98% on CIFAR-10. This makes our work the only transformer based model in the top 25 best performing models on CIFAR-10, without pre-training, and significantly smaller than the vast majority. Our model also outperforms most comparable CNN-based models within this domain, with the exception of certain Neural Architectural Search techniques [5]. Additionally, we show that our model can be lightweight, only needing 0.28 million parameters and still reach close to 90% top-1% accuracy on CIFAR-10. On ImageNet, CCT achieves 80.67% accuracy while still maintaining a small number of parameters and reduced computation. CCT outperforms ViT, while containing less than a third of the number of parameters with about a third of the computational complexity (MACs). Additionally, CCT outperform similarly sized and more recent models, such as DeiT [19]. This demonstrates the scalability of our model while maintaining compactness and computational efficiency.

The main contributions of this paper are:

- Extending transformer-based research to small data regimes, by introducing ViT-Lite, which can be trained from scratch and achieve high accuracy on datasets such as CIFAR-10.
- Introducing Compact Vision Transformer (CVT) with a new sequence pooling strategy, which pools over output tokens and improves performance.
- Introducing Compact Convolutional Transformer (CCT) to increase performance and provide flexibility for input image sizes while also demonstrating that these variants do not depend as much on Positional Embedding compared to the rest.

In addition, we demonstrate that our CCT model is fast, obtaining 90% accuracy on CIFAR-10 using a single NVIDIA 2080Ti GPU and 80% when trained on a CPU (AMD 5900X), both in under 30 minutes. Additionally, since our model has a relatively small number of parameters, it can be trained on the majority of GPUs, even if researchers do not have access to top of the line hardware. Through these efforts, we aim to help enable and extend research around Transformers to cases with limited data and/or researchers with limited resources.

2. Related Works

In NLP research, attention mechanisms [15, 2, 28] gained popularity for their ability to weigh different features within sequential data. Transformers [41] were introduced as a fully attention-based model, primarily for machine translation and NLP in general. Following this, attention-based models, specifically transformers have been applied to a wide variety of tasks beyond machine translation [11, 25, 46], including: visual question answering [27, 38], action recognition [4, 13], and the like. Many researchers also leveraged a combination of attention and convolutions in neural networks for visual tasks [42, 18, 3, 51]. Ramachandran *et al*. [33] introduced one of the first vision models that rely primarily on attention. Dosovitskiy *et al*. [12] introduced the first stand-alone transformer based model for image classification (ViT). In the following subsections, we briefly revisit ViT and several other related works.

2.1. Vision Transformer

Dosovitskiy *et al*. [12] introduced ViT primarily to show that reliance on CNNs or their structure is unnecessary, as prior to it, most attention-based models for vision were used either with convolutions [42, 3, 51, 6], or kept some of their properties [33]. The motivation, beyond self-attention's many desirable properties for a network, specifically its ability to make long range connections, was scalability. It was shown that ViT can successfully keep scaling, while CNNs start saturating in performance as the number of training samples grew. Through this, they concluded that large-scale training triumphs over the advantage of inductive bias that CNNs have, allowing their model to be competitive with CNN based architectures given sufficiently large amount of training data. ViT is composed of several parts: Image Tokenization, Positional Embedding, Classification Token, the Transformer Encoder, and a Classification Head. These subjects are discussed in more detail below.

Image Tokenization: A standard transformer takes as input a sequence of vectors, called tokens. For traditional NLP based transformers, word ordering provides a natural order to sequence the data, but this is not so obvious for images. To tokenize an image, ViT subdivides an image into non-overlapping square patches in raster-scan order. The sequence of patches, $\mathbf{x_p} \in \mathbb{R}^{H\times(P^2C)}$ with patch size P, are flattened into 1D vectors and transformed into latent vectors of dimension d. This is equivalent to a convolutional layer with d filters, and $P \times P$ kernel size and stride. This simple patching and embedding method has a few limitations, in particular: loss of information along the boundary regions.

Positional Embedding: Positional embedding adds spatial information into the sequence. Since the model does not actually know anything about the spatial relationship between tokens, adding extra information to reflect that can be useful. Typically, this is either a learned embedding or tokens are given weights from two sine waves with high frequencies, which is sufficient for the model to learn that there exists a positional relationship between these tokens.

Transformer Encoder: A transformer encoder consists of a series of stacked encoding layers. Each encoder layer is comprised of two sub-layers: Multi-Headed Self-Attention (MHSA) and a Multi-Layer Perceptron (MLP) head. Each sub-layer is preceded by a layer normalization (LN), and followed by a residual connection to the next sub-layer.

Classification: Vision transformers typically add an extra learnable `[class]` token to the sequence of the embedded patches, representing the class parameter of an entire image and its state after transformer encoder can be used for classification. `[class]` token contains latent information, and through self-attention accumulates more information about the sequence, which is later used for classification. ViT [12] also explored averaging output tokens instead, but found no significant difference in performance. They did however find that the learning rates have to be adjusted between the two variants: `[class]` token *vs*. average pooling.

2.2. Data-Efficient Transformers

In an effort to reduce dependence on data, Touvron *et al*. [40] proposed Data-Efficient Image Transform-

ers (DeiT). Using more advanced training techniques, and a novel knowledge transfer method, DeiT improves the classification performance of ViT on ImageNet-1k without large-scale pre-training on datasets such as JFT-300M [39] or ImageNet-21k [10]. By relying only on more augmentations [8] and training techniques [50, 49], it is shown that much smaller ViT variants that were unexplored by Dosovitskiy *et al*. can outperform the larger ones on ImageNet-1k without pre-training. Furthermore, DeiT variants were pushed even further through their novel knowledge transfer technique, specifically when using a convolutional model as the teacher. This work pushes forward accessibility of transformers in medium-sized datasets, and we aim to follow by extending the study to even smaller sets of data and smaller models. However, we base our work on the notion that *if a small dataset happens to be sufficiently novel, pre-trained models will not help train on that domain* and the model will not be appropriate for that dataset. While knowledge transfer is a strong technique, it requires a pre-trained model for any given dataset, adding to training time and complexity, with an additional forward pass, and as pointed out by Touvron *et al*. is usually only significant when there's a convolutional teacher available to transfer the inductive biases. As a result, it can be argued that if a network utilized just the bare minimum of convolutions, while keeping the pure transformer structure, it may need to rely less on large-scale training and transfer of inductive biases through knowledge transfer.

Yuan *et al*. [48] proposed Tokens-to-token ViT (T2T-ViT), which adopts a window- and attention-based tokenization strategy. Their tokenizer extracts patches of the input feature map, similar to a convolution, applies three sets of kernel weights, and produces three sets of feature maps, which are fed to self-attention as query and key-value pairs. This process is equivalent to convolutions producing the QKV projections in a self-attention module. Finally, this strategy is repeated twice, followed by a final patching and embedding. The entire process replaces patch and embedding in ViT. This strategy, along with their small-strided patch extraction, allows their network to model local structures, including along the boundaries between patches. This attention-based patch interaction leads to finer-grained tokens which allow T2T-ViT to outperform previous Transformer-based models on ImageNet. T2T-ViT differs from our work, in that it focuses on medium-sized datasets like ImageNet, which are not only far too large for many research problems in science and medicine but also resource demanding. T2T tokenizer also has more parameters and complexity compared to a convolutional one.

2.3. Convolution-inspired Transformers

Many works have been motivated to improve vision transformers and eliminate the need for large-scale pre-training. ConViT [9] introduces a *gated positional self-attention* (GPSA) that allows for a "soft" convolutional inductive bias within their model. GPSA allows their network to have more flexibility with respect to positional information. Since GPSA is able to be initialized as a convolutional layer, this allows their network to sometimes have the properties of convolutions or alternatively having the properties of attention. Its *gating parameter* can be adjusted by the network, allowing it to become more expressive and adapt to the needs of the dataset. Convolution-enhanced image Transformers (Ceit) [47] utilize convolutions throughout their model. They propose a convolution-based Image-to-Token module for tokenization. They also re-design the encoder with layers of multi-headed self-attention and their novel Locally Enhanced Feedforward Layer, which processes the spatial information form the extracted token. This allows creates a network that is competitive with other works such as DeiT on ImageNet. Convolutional vision Transformer (CvT) [45] introduces convolutional transformer encoder layers, which use convolutions instead of linear projections for the QKV in self-attention. They also introduce convolutions into their tokenization step, and report competitive results compared to other vision transformers on ImageNet-1k. All of these works report results when trained from scratch on ImageNet (or larger) datasets.

2.4. Comparison

Our work differs from the aforementioned in several ways, in that it focuses on answering the following question: **Can vision transformers be trained from scratch on small datasets?** Focusing on a small datasets, we seek to create a model that can be trained, from scratch, on datasets that are orders of magnitude smaller than ImageNet. Having a model that is compact, small in size, and efficient allows greater accessibility, as training on ImageNet is still a difficult and data intensive task for many researchers. Thus our focus is on an accessible model, with few parameters, that can quickly and efficiently be trained on smaller platforms while still maintaining SOTA results.

3. Method

In order to provide empirical evidence that vision transformers are trainable from scratch when dealing with small sets of data, we propose three different models: ViT-Lite, **C**ompact **V**ision **T**ransformers (CVT), and **C**ompact **C**onvolutional **T**ransformers (CCT). ViT-Lite is nearly identical to the original ViT in terms of architecture, but with a more suitable size and patch size for small-scale learning. CVT builds on this by using our **Seq**uence **Pool**ing method (SeqPool), that pools the entire sequence of tokens produced by the transformer encoder. SeqPool replaces the conventional `[class]` token. CCT builds

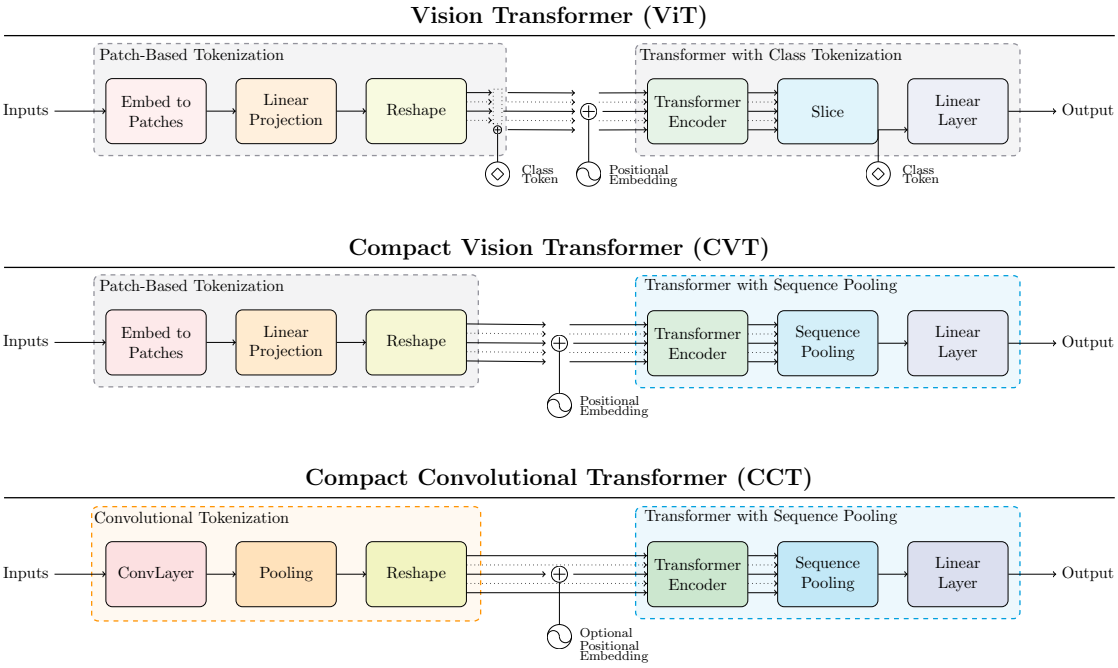


Figure 2: Comparing ViT (top) to CVT (middle) and CCT (bottom). CVT can be thought of as an ablated version of CCT, only utilizing sequence pooling and not a convolutional tokenizer. CVT may be preferable with more limited compute, as the patch-based tokenization is faster.

on CVT and utilizes a convolutional tokenizer, generating richer tokens and preserving local information. The convolutional tokenizer is better at encoding relationships between patches compared to the original ViT [12]. A detailed modular-level comparison of these models can be viewed in Figure 2.

The components of our compact transformers are further discussed in the following subsections: Transformer-based Backbone, Small and Compact Models, SeqPool, and Convolutional Tokenizer.

3.1. Transformer-based Backbone

In terms of model design, we follow the original Vision Transformer [12], and original Transformer [41]. As mentioned in Section 2.1, the encoder consists of transformer blocks, each including an MHSA layer and an MLP block. The encoder also applies Layer Normalization, $GELU$ activation, and dropout. Positional embeddings can be learnable or sinusoidal, both of which are effective.

3.2. Small and Compact Models

We propose smaller and more compact vision transformers. The smallest ViT variant, ViT-Base, includes a 12 layer transformer encoder with 12 attention heads, 64 dimensions per head, and 2048-dimensional hidden layers in the MLP blocks. This, along with the classifier and 16x16 patch and embedder results in over 85M parameters. We propose variants with as few as 2 layers, 2 heads, and 128-dimensional hidden layers. In Appendix A, we summarized the details of the variants we propose, the smallest of which can have as little as 0.22M parameters, while the largest (for small-scale learning) only have 3.8M parameters. We also adjust the tokenizer (patch size) according to the dataset we're training on, based on its image resolution. These variants, which are mostly similar in architecture to ViT, but different in size, are referred to as ViT-Lite. In our notation, we use the number of layers to specify size, as well as tokenization details: for instance, ViT-Lite-*12*/**16** has *12* transformer encoder layers, and a **16×16** patch size.

3.3. SeqPool

In order to map the sequential outputs to a singular class index, ViT [12] and most other common transformer-based classifiers follow BERT [11], in forwarding a learnable class or query token through the network and later feeding it to the classifier. Other common practices include global average pooling (averaging over tokens), which have been shown to be preferable in some scenarios. We introduce SeqPool, an attention-based method which pools over the output sequence of tokens. Our motivation is that the output sequence contains relevant information across different parts of the input image, therefore preserving this information can improve performance, and at no additional parameters compared to the learnable token. Additionally, this change

slightly decreases computation, due one less token being forwarded. This operation consists of mapping the output sequence using the transformation $T : \mathbb{R}^{b\times n\times d} \mapsto \mathbb{R}^{b\times d}$. Given:

$$\mathbf{x}_L = \mathrm{f}(\mathbf{x}_0) \in \mathbb{R}^{b\times n\times d}$$

where $\mathbf{x}_L$ is the output of an L layer transformer encoder f, b is batch size, n is sequence length, and d is the total embedding dimension. $\mathbf{x}_L$ is fed to a linear layer $\mathrm{g}(\mathbf{x}_L) \in \mathbb{R}^{d\times 1}$, and softmax activation is applied to the output:

$$\mathbf{x}'_L = \mathrm{softmax}\left(\mathrm{g}(\mathbf{x}_L)^T\right) \in \mathbb{R}^{b\times 1\times n}$$

This generates an importance weighting for each input token, which is applied as follows:

$$\mathbf{z} = \mathbf{x}'_L\mathbf{x}_L = \mathrm{softmax}\left(\mathrm{g}(\mathbf{x}_L)^T\right) \times \mathbf{x}_L \in \mathbb{R}^{b\times 1\times d} \quad (1)$$

By flattening, the output $z \in \mathbb{R}^{b\times d}$ is produced. This output can then be sent through a classifier.

SeqPool allows our network to weigh the sequential embeddings of the latent space produced by the transformer encoder and correlate data across the input data. This can be thought of this as attending to the sequential data, where we are assigning importance weights across the sequence of data, only after they have been processed by the encoder. We tested several variations of this pooling method, including learnable and static methods, and found that the learnable pooling performs the best. Static methods, such as global average pooling have already been explored by ViT as well, as pointed out in section 2.1. We believe that the learnable weighting is more efficient because each embedded patch does not contain the same amount of entropy. This allows the model to apply weights to tokens with respect to the relevance of their information. Additionally, sequence pooling allows our model to better utilize information across spatially sparse data. We will further study the effects of this pooling in the ablation study (Sec 4.4). By replacing the conventional `class` token in ViT-Lite with SeqPool, Compact Vision Transformer is created. We use the same notations for this model: for instance, CVT-*7*/**4** has *7* transformer encoder layers, and a **4×4** patch size.

3.4. Convolutional Tokenizer

In order to introduce an inductive bias into the model, we replace patch and embedding in ViT-Lite and CVT, with a simple convolutional block. This block follows conventional design, which consists of a single convolution, $ReLU$ activation, and a max pool. Given an image or feature map $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$:

$$\mathbf{x}_0 = \mathrm{MaxPool}(\mathrm{ReLU}(\mathrm{Conv2d}(\mathbf{x}))) \quad (2)$$

where the $\mathrm{Conv2d}$ operation has d filters, same number as the embedding dimension of the transformer backbone. Additionally, the convolution and max pool operations can be overlapping, which could increase performance by injecting inductive biases. This allows our model to maintain locally spatial information. Additionally, by using this convolutional block, the models enjoy an added flexibility over models like ViT, by no longer being tied to the input resolution strictly divisible by the pre-set patch size. We seek to use convolutions to embed the image into a latent representation, because we believe that it will be more efficient and produce richer tokens for the transformer. These blocks can be adjusted in terms of downsampling ratio (kernel size, stride and padding), and are repeatable for even further downsampling. Since self-attention has a quadratic time and space complexity with respect to the number of tokens, and number of tokens is equal to the resolution of the input feature map, more downsampling results in fewer tokens which noticeably decreases computation (at the expense of performance). We found that on top of the added performance gains, this choice in tokenization also gives more flexibility toward removing the positional embedding in the model, as it manages to maintain a very good performance. This is further discussed in Appendix C.1.

This convolutional tokenizer, along with SeqPool and the transformer encoder create Compact Convolutional Transformers. We use a similar notation for CCT variants, with the exception of also denoting the number of convolutional layers: for instance, CCT-*7*/**3**x2 has *7* transformer encoder layers, and a 2-layer convolutional tokenizer with **3×3** kernel size.

4. Experiments

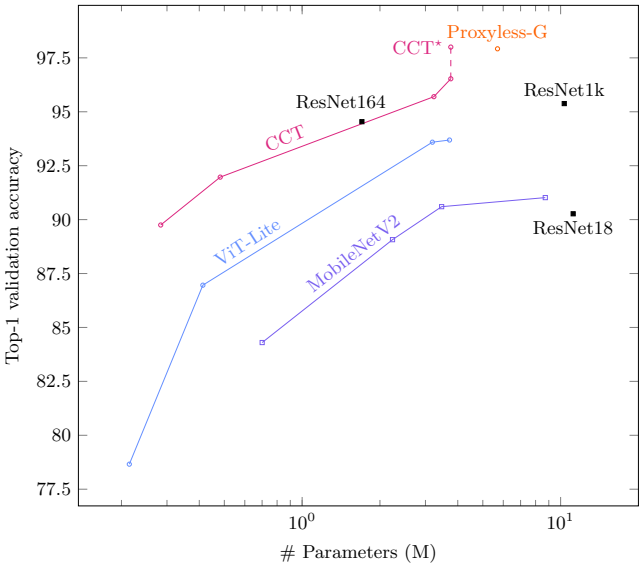


Figure 3: CIFAR-10 accuracy *vs*. model size (sizes < 12M). CCT* was trained longer.

Table 1: Top-1 validation accuracy comparisons. ⋆ variants were trained longer (see Table 2)

Model	C-10	C-100	Fashion	MNIST	# Params	MACs
Convolutional Networks (Designed for ImageNet)						
ResNet18	90.27%	66.46%	94.78%	99.80%	11.18 M	0.04 G
ResNet34	90.51%	66.84%	94.78%	99.77%	21.29 M	0.08 G
MobileNetV2/0.5	84.78%	56.32%	93.93%	99.70%	0.70 M	< **0.01** G
MobileNetV2/2.0	91.02%	67.44%	95.26%	99.75%	8.72 M	0.02 G
Convolutional Networks (Designed for CIFAR)						
ResNet56[16]	94.63%	74.81%	95.25%	99.27%	0.85 M	0.13 G
ResNet110[16]	95.08%	76.63%	95.32%	99.28%	1.73 M	0.26 G
ResNet1k-v2⋆[17]	95.38%	–	–	–	10.33 M	1.55 G
Proxyless-G[5]	97.92%	–	–	–	5.7 M	–
Vision Transformers						
ViT-12/16	83.04%	57.97%	93.61%	99.63%	85.63 M	0.43 G
ViT-Lite-7/16	78.45%	52.87%	93.24%	99.68%	3.89 M	0.02 G
ViT-Lite-7/8	89.10%	67.27%	94.49%	99.69%	3.74 M	0.06 G
ViT-Lite-7/4	93.57%	73.94%	95.16%	99.77%	3.72 M	0.26 G
Compact Vision Transformers						
CVT-7/8	89.79%	70.11%	94.50%	99.70%	3.74 M	0.06 G
CVT-7/4	94.01%	76.49%	95.32%	99.76%	3.72 M	0.25 G
Compact Convolutional Transformers						
CCT-2/3×2	89.75%	66.93%	94.08%	99.70%	**0.28** M	0.04 G
CCT-7/3×2	95.04%	77.72%	95.16%	99.76%	3.85 M	0.29 G
CCT-7/3×1	96.53%	80.92%	**95.56%**	**99.82%**	3.76 M	1.19 G
CCT-7/3×1⋆	**98.00%**	**82.72%**	–	–	3.76 M	1.19 G

Table 2: **CCT-7/3×1** top-1 accuracy on CIFAR-10/100 when trained longer

# Epochs	Pos. Emb.	CIFAR-10	CIFAR-100
300	Learnable	96.53%	80.92%
1500	Sinusoidal	97.48%	82.72%
5000	Sinusoidal	**98.00%**	**82.87%**

4.1. Datasets

We conducted image classification experiments using our method on the following datasets: CIFAR-10, CIFAR-100 (MIT License) [21], MNIST, Fashion-MNIST, Oxford Flowers-102 [30], and ImageNet-1k [10]. The first four datasets not only have a small number of training samples, but they are also small in resolution. Additionally, MNIST and Fashion-MNIST only contain a single channel, greatly reducing the information density. Flowers-102 has a relatively small number of samples, while having relatively higher resolution images and 102 classes. We divided these datasets into three categories: small-scale small resolution datasets (CIFAR-10/100, MNIST, and Fashion-MNIST), small-scale larger resolution (Flowers-102), and medium-scale (ImageNet-1k) datasets. We also include a study on NLP classification, presented in appendix G.

4.2. Hyperparameters

We used the timm package [43] to train the models (see Appendix E for details), except for cited works which are reported directly. For all experiments, we conducted a hyperparameter sweep for every different method and report the best results we were able to achieve. We will release all checkpoints corresponding to the reported numbers, and detailed training settings in the form of YAML files, with our

Table 3: ImageNet Top-1 validation accuracy comparison (no extra data or pretraining). This shows that larger variants of CCT could also be applicable to medium-sized datasets

Model	Top-1	# Params	MACs	Training Epochs
ResNet50 [16]	77.15%	25.55 M	4.15 G	120
ResNet50 (2021) [44]	79.80%	25.55 M	4.15 G	300
ViT-S [19]	79.85%	**22.05** M	**4.61** G	300
CCT-14/7×2	**80.67%**	22.36 M	5.53 G	300
DeiT-S [19]	81.16%	22.44M	**4.63** G	300
CCT-14/7×2 Distilled	**81.34%**	**22.36** M	5.53 G	300

Table 4: Flowers-102 Top-1 validation accuracy comparison. CCT outperforms other competitive models, having significantly fewer parameters and GMACs. This demonstrates the compactness on small datasets even with large images

Model	Resolution	Pretraining	Top-1	# Params	MACs
CCT-14/7×2	224	-	97.19%	22.17 M	18.63 G
DeiT-B	384	ImageNet-1k	98.80%	86.25 M	55.68 G
ViT-L/16	384	JFT-300M	99.74%	304.71 M	191.30 G
ViT-H/14	384	JFT-300M	99.68%	661.00 M	504.00 G
CCT-14/7×2	384	ImageNet-1k	**99.76%**	**22.17** M	**18.63** G

code. We also provide a report on hyperparamter settings in Appendix E. Unless stated otherwise, all tests were run for 300 epochs, and the learning rate is reduced per epoch based on cosine annealing [26]. All transformer based models (ViT-Lite, CVT, and CCT) were trained using the AdamW optimizer.

4.3. Performance Comparison

Small-scale small resolution training: In order to demonstrate that vision transformers can be as effective as convolutional neural networks, even in settings with small sets of data, we compare our compact transformers to ResNets [16], which are still very useful CNNs for small to medium amounts of data, as well as to MobileNetV2 [35], which are very compact and small-sized CNNs. We also compare with results from [17] where He *et al*. designed very deep (up to 1001 layers) CNNs specifically for CIFAR. The results are presented in Table 1, all of which are of models trained from scratch. We highlight the top performers. CCT-7/3x2 achieves on par results with the CNN models, while having significantly fewer parameters in some cases. We also compare our method to the original ViT [12] in order to express the effectiveness of smaller sized backbones, convolutional layers, as well our pooling technique. As these datasets were not trained from scratch in the original paper, we attempted to train the smallest variant: ViT-B/16 (ViT-12/16). We trained our best performing model, CCT-7/3x1, for longer than the 300 epochs to see how far it can go. Surprisingly, this model can get as high as 98% accuracy on CIFAR-10, and 82.87% accuracy on CIFAR-100 when trained for 5000 epochs, which is still fewer iterations an ImageNet pre-training would have. We present results from training on CIFAR-10/100 for 300, 1500 and 5000 epochs in Table 2. We observed that sinusoidal positional embedding had a small but noticeable edge over learnable when training longer. This represents the only transformer based model in the top 25 results on PapersWithCode for CIFAR-10 where models have no extra data or pre-training[1]. In addition to this, it is also one of the smallest models, being 15% the size of ResNet50 while maintaining similar performance. We present a plot of different models in Table 1 in Figure 3.

Medium-scale training: ImageNet training results are presented in Table 3, and compared to ResNet50 [16], ViT, and DeiT. We report ResNet50 from the original paper [16], as well as from Wightman *et al*. [44] which uses a similar training schedule to ours, and is therefore a fairer comparison. We also report a smaller ViT variant as proposed by Touvron *et al*. [40]. We also report CCT's performance with knowledge distillation, in order to compare it to DeiT [40]. Similar to DeiT, we trained our CCT-14/7x2 with a convolutional teacher and hard distillation loss. We used a RegNetY-16GF [32] (84M parameters), the same model

[1]https://paperswithcode.com/sota/image-classification-on-cifar-10

DeiT selected as the teacher. It is noticeable that distillation does not have as significant of an effect on CCT it does on DeiT. This can be attributed to the already existing inductive biases from the convolutional tokenizer. DeiT authors argued that a convolutional teacher would be able to transfer inductive biases to the student model.

Small-scale higher-resolution training: We also present our results on Flowers-102, in which we successfully reach reasonable performance without any pre-training, and with the same model size as our ImageNet model. We also claim state of the art with **99.76%** top-accuracy with ImageNet pretraining, which exceeds even far larger models pre-trained on JFT-300M. In addition to this we note that our model is at least a quarter the size of the next best model and almost $30\times$ smaller than ViT-H/14. It can also be seen that CCT is $3-27\times$ more computationally efficient.

4.4. Ablation Study

We extend our previous comparisons by doing an ablation study on our methods. In this study, we progressively transform the original ViT into ViT-Lite, CVT, and CCT, and compare their top-1 accuracy scores. In this particular study, we report the results on CIFAR-10 and CIFAR-100 in Table 8 in Appendix F.

5. Conclusion

Transformers have commonly been perceived to be only applicable to larger-scale or medium-scale training. While their scalability is undeniable, we have shown within this paper that with proper configuration, a transformer can be successfully used in small data regimes as well, and outperform convolutional models of equivalent, and even larger, sizes. Our method is simple, flexible in size, and the smallest of our variants can be easily loaded on even a minimal GPU, or even a CPU. While part of research has been focused on large-scale models and datasets, we focus on smaller scales in which there is still much research to be done in data efficiency. We show that CCT can outperform other transformer based models on small datasets while also having a significant reduction in computational costs and memory constraints. This work demonstrates that transformers do not require vast computational resources and can allow for their applications in even the most modest of settings. This type of research is important to many scientific domains where data is far more limited that the conventional machine learning datasets which are used in general research. Continuing research in this direction will help open research up to more people and domains, extending machine learning research.

References

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007. 13

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. 3

[3] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3286–3295, 2019. 3

[4] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? *arXiv preprint arXiv:2102.05095*, 2021. 3

[5] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations*, 2018. 2, 7, 17

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc 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. 13

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020. 4, 13

[9] Stéphane d'Ascoli, Hugo Touvron, Matthew Leavitt, Ari Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *arXiv preprint arXiv:2103.10697*, 2021. 4

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 4, 7

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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, 2019. 3, 5

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 3, 5, 8

[13] Rohit Girdhar, Joao Carreira, Carl Doersch, and Andrew Zisserman. Video action transformer network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2019. 3

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*. MIT press Cambridge, 2016. 1

[15] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014. 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 7, 8, 17

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 7, 8, 17

[18] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3

[19] Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In *International Conference on Machine Learning*, pages 4475–4483. PMLR, 2020. 2, 8

[20] Alexander Ke, William Ellsworth, Oishi Banerjee, Andrew Y. Ng, and Pranav Rajpurkar. Chextransfer: performance and parameter efficiency of imagenet models for chest x-ray interpretation. *Proceedings of the Conference on Health, Inference, and Learning*, Apr. 2021. 2

[21] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009. 7

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. 1

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[24] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002. 13

[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 3

[26] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017. 8

[27] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019. 3

[28] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015. 3

[29] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011. 13

[30] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 7

[31] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 14

[32] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 8

[33] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. 3

[34] Daniel L Ruderman and William Bialek. Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814, 1994. 1

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 8

[36] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. 1

[37] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013. 13

[38] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. *arXiv preprint arXiv:1908.08530*, 2019. 3

[39] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017. 4

[40] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020. 3, 8

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008, 2017. 2, 3, 5

[42] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017. 3

[43] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 7, 13

[44] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 8

[45] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808*, 2021. 4

[46] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019. 3

[47] Kun Yuan, Shaopeng Guo, Ziwei Liu, Aojun Zhou, Fengwei Yu, and Wei Wu. Incorporating convolution designs into visual transformers. *arXiv preprint arXiv:2103.11816*, 2021. 4

[48] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021. 4

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019. 4, 13

[50] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 4, 13

[51] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019. 3

[52] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *arXiv preprint arXiv:1509.01626*, 2015. 13

[53] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020. 13

[54] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 2

A. Variants

Within this appendix, we present architectural details of our variants in Tables 5 and 6.

Table 5: Transformer backbones in each variant.

Model	# Layers	# Heads	Ratio	Dim
ViT-Lite-6	6	4	2	256
ViT-Lite-7	7	4	2	256
CVT-6	6	4	2	256
CVT-7	7	4	2	256
CCT-2	2	2	1	128
CCT-4	4	2	1	128
CCT-6	6	4	2	256
CCT-7	7	4	2	256
CCT-14	14	6	3	384

Table 6: Tokenizers in each variant.

Model	# Layers	# Convs	Kernel	Stride
ViT-Lite-7/8	7	1	8×8	8×8
ViT-Lite-7/4	7	1	4×4	4×4
CVT-7/8	7	1	8×8	8×8
CVT-7/4	7	1	4×4	4×4
CCT-2/3x2	2	2	3×3	1×1
CCT-7/3x1	7	1	3×3	1×1
CCT-7/7x2	7	2	7×7	2×2

B. Computational Resources

For most experiments, we used a machine with an Intel(R) Core(TM) i9-9960X CPU @ 3.10GHz and 4 NVIDIA(R) RTX(TM) 2080Tis (11GB). The exception was the CPU test which was performed with an AMD Ryzen 9 5900X. Each ImageNet experiment was performed on a single machine either with 2 AMD EPYC(TM) 7662s and 8 NVIDIA(R) RTX(TM) A6000s (48GB), or 2 AMD EPYC(TM) 7713s and 8 NVIDIA(R) A100s (80GB).

C. Additional analyses

Within this appendix we present some additional performance analyses which were conducted.

C.1. Positional Embedding

To determine the effects of our small & compact design, sequence pooling, and convolutional tokenizer, we perform an ablation study focused on positional embedding, seen in Table 7. In this study, we experiment with ViT (original sizing), ViT-Lite, CVT, and CCT, and investigate the effects of: a learnable positional embedding, a standard sinusoidal embedding, as well as no positional embedding. We finish the table with our best model, which also has augmented training and an optimal tuning (refer to Appendix E). In these experiments, we find that positional encoding matters in all variants, but to varying degrees. In particular, CCT relies less on positional encoding, and it can be safely removed much impact in accuracy. We also tested our CCT model without SeqPool, using the standard `[class]` token instead, and found that there was little to no effect from having a positional encoder or not, depending on model size. This suggests that convolutions are what helps provide spatially sparse information to the transformer, while also helping the model overcome some of the previous limitations, allowing for more efficient use of data. We do find that SeqPool helps slightly in this respect, but overall has a larger effect on increasing total accuracy. Lastly, we find that with proper data augmentation and tuning, the overall performance can be increased, and a low dependence on positional information can be maintained.

C.2. Performance vs Dataset Size

In this experiment, we evaluated model performance on smaller subsets of CIFAR-10 to determine the relationship between performance and the number of samples within a dataset. Samples were removed uniformly from each class in CIFAR-10. For this experiment, we compared ViT-Lite and CCT. In Figure 4, we see the comparison of each model's accuracy vs the number of samples per class. We show how each model performs when given only 500, 1000, 2000, 3000, 4000, or 5000 (original) samples per class, meaning the total training set ranges from one tenth the size to full. It can be ovserved that CCT is more robust since it is able to obtain higher accuracy with a lower number of samples per class, especially in the low sample regime.

C.3. Performance vs Dimensionality

In order to determine whether transformers are dependant on high dimensional data, as opposed to the number of samples (explored in Appendix C.2), we experimented with downsampled and upsampled versions of CIFAR-10. In Figure 5, we present the image dimensionality vs the performance of CCT *vs*. ViT-Lite. Both models were trained with images of sizes ranging from 16×16 to 64×64. It can be observed that CCT performs better on all image sizes, with a widening difference as the number of pixels increases. From this, it can be inferred that CCT is able to better utilize the information density of an image, while ViT does not see continued performance increases after the standard 32x32 size.

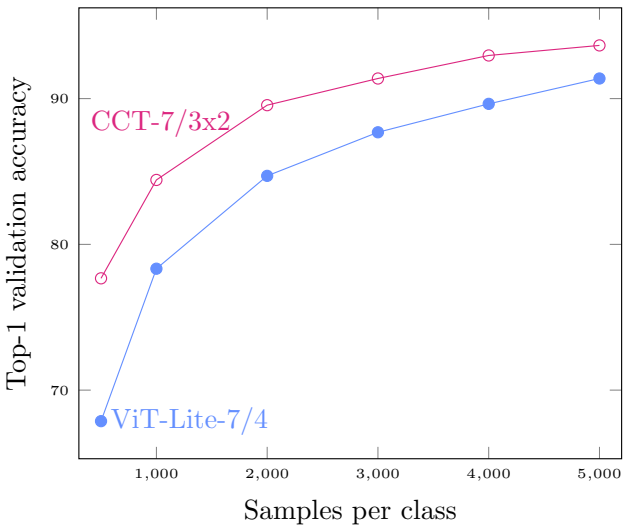


Figure 4: Reduced # samples / class (CIFAR-10)

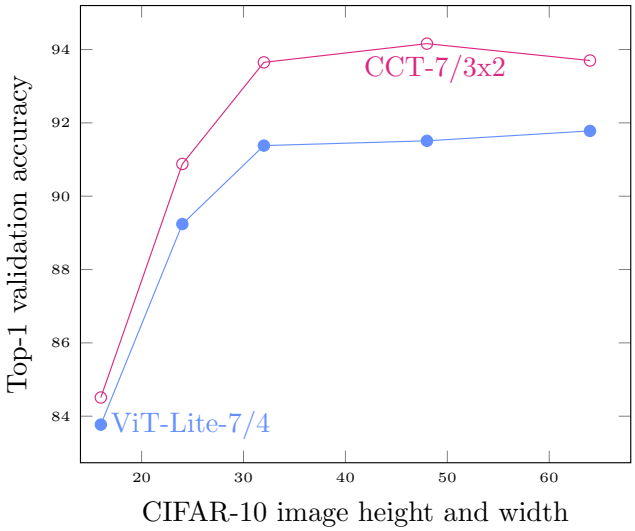


Figure 5: Image Size vs Accuracy (CIFAR-10)

D. Dimensionality Experiments

Within this appendix, we extend the analysis from Appendix C.3, showing the difference in performance when using different types of positional embedding. Figure 6 shows the difference of the accuracy when models are being trained from scratch. On the other hand, Figure 7 shows the performance difference when models are only used in inference and pre-trained on the 32×32 sized images. We note that in Figure 7(a) that we do not provide inference for image sizes greater than the pre-trained image because the learnable positional embeddings do not allow us to extend in this direction. We draw the reader's attention to Figure 6(c) and Figure 7(c) to denote the large difference between the models when positional embedding is not used. We can see that in training CCT has very little difference when positional embeddings are used. Additionally, it should be noted that when performing inference our non-positional embedding CCT model has much higher generalizability than its ViT-Lite counterpart.

E. Hyperparameter tuning

We used the timm package [43] for our experiments (excluding NLP experiments). We also sued CutMix [49], Mixup [50], Randaugment [8], and Random Erasing [53]. For our small-scale small-resolution experiments, we conducted a hyperparameter sweep for each model on each dataset separately. However, all experiments that trained models from scratch, were trained for 300 epochs, unless mentioned otherwise. ViT, CVT and CCT all used the weighted Adam optimizer ($\beta_1 = 0.9$ and $\beta_2 = 0.999$). For CNNs, we observed that some models and datasets achieved their best results using AdamW, while most others performed best with SGD with momentum (0.9). We will release model checkpoints (PyTorch pickle files), as well as a full list of hyperparameters and training settings (in the form of YAML files readable by `timm`) along with our code for reproduction.

F. Ablation Study

Here in Table 8 we present the results from section 4.4. We provide a full list of ablated terms showing which factors give the largest boost in performances. "Model" column refers to variant (see Table 5 for details), "Conv" specifies the number of convolutional blocks (if an), and "Conv Size" specifies the kernel size. "Aug" denotes the use of AutoAugment [7]. "Tuning" specifies a minor change in dropout, attention dropout, and/or stochastic depth (see Table 9). The first row in Table 8 is essentially ViT. The next three rows are modified variants of ViT, which are not proposed in the original paper. These variants are more compact and use smaller patch sizes. It should be noted that the numbers reported in this table are best out of 4.

G. NLP experiments

To demonstrate the general purpose nature of our model we extended it to the domain of Natural Language Processing, focusing on classification tasks. This shows that our model is a general purpose classifier and is not restricted to the domain of image classification. Within this section, we present our text classification results on 5 datasets: AGNews [52], TREC [24], SST [37], IMDb [29], DBpedia [1]. The results are summarized in Table 10. As can be seen, our model outperforms the vanilla transformer, demonstrating that the techniques we use here also help with NLP tasks.

Table 7: Top-1 validation accuracy comparison when changing the positional embedding method. Augmentations and training techniques such as Mixup and CutMix were turned off for these experiments to highlight differences better. The numbers reported are best out of 4 runs with random initializations. † denotes model trained with extra augmentation and hyperparameter tuning.

Model	PE	CIFAR-10	CIFAR-100
Conventional Vision Transformers are more dependent on Positional Embedding			
ViT-12/16	Learnable	69.82% *(+3.11%)*	40.57% *(+1.01%)*
	Sinusoidal	69.03% *(+2.32%)*	39.48% *(−0.08%)*
	None	66.71% *(baseline)*	39.56% *(baseline)*
ViT-Lite-7/8	Learnable	83.38% *(+7.25%)*	55.69% *(+7.15%)*
	Sinusoidal	80.86% *(+4.73%)*	53.50% *(+4.96%)*
	None	76.13% *(baseline)*	48.54% *(baseline)*
CVT-7/8	Learnable	84.24% *(+6.52%)*	55.49% *(+7.23%)*
	Sinusoidal	80.84% *(+3.12%)*	50.82% *(+2.56%)*
	None	77.72% *(baseline)*	48.26% *(baseline)*
Compact Convolutional Transformers are less dependent on Positional Embedding			
CCT-7/7	Learnable	82.03% *(+0.21%)*	63.01% *(+3.24%)*
	Sinusoidal	81.15% *(−0.67%)*	60.40% *(+0.63%)*
	None	81.82% *(baseline)*	59.77% *(baseline)*
CCT-7/3×2	Learnable	90.69% *(+1.67%)*	65.88% *(+2.82%)*
	Sinusoidal	89.93% *(+0.91%)*	64.12% *(+1.06%)*
	None	89.02% *(baseline)*	63.06% *(baseline)*
CCT-7/3×2†	Learnable	95.04% *(+0.64%)*	77.72% *(+0.20%)*
	Sinusoidal	94.80% *(+0.40%)*	77.82% *(+0.30%)*
	None	94.40% *(baseline)*	77.52% *(baseline)*
CCT-7/3×1†	Learnable	96.53% *(+0.29%)*	80.92% *(+0.65%)*
	Sinusoidal	96.27% *(+0.03%)*	80.12% *(−0.15%)*
	None	96.24% *(baseline)*	80.27% *(baseline)*
CCT-7/7×1-noSeqPool	Learnable	82.41% *(+0.12%)*	62.61% *(+3.31%)*
	Sinusoidal	81.94% *(−0.35%)*	61.04% *(+1.74%)*
	None	82.29% *(baseline)*	59.30% *(baseline)*
CCT-7/3×2-noSeqPool	Learnable	90.41% *(+1.49%)*	66.57% *(+1.40%)*
	Sinusoidal	89.84% *(+0.92%)*	64.71% *(−0.46%)*
	None	88.92% *(baseline)*	65.17% *(baseline)*

The network is slightly modified from the vision CCT. We use GloVe (Apache License 2.0) [31] to provide the word embedding for the model, and do not train these parameters. Note that model sizes do not reflect the number of parameters for GloVe, which is around 20M. We treat text as single channel data and the embedding dimension as size 300. Additionally, the convolution kernels have size 1. Finally, we include masking in the typical manner. By doing so, CCT can get upwards of a 3% improvement on some datasets while using less parameters than vanilla transformers. Similar to our vision results, we find that CCT performs well on small NLP datasets. We note that the CCT models that perform best all have less than 1M parameters, which are significantly smaller than there vanilla counterparts, while out performing them.

Table 8: CIFAR Top-1 validation accuracy when transforming ViT into CCT step by step. We disabled advanced training techniques and augmentations for these runs.

Model	CLS	# Conv	Conv Size	Aug	Tuning	C-10	C-100	# Params	MACs
ViT-12/16	CT	✗	✗	✗	✗	69.82%	40.57%	85.63 M	0.43 G
ViT-Lite-7/16	CT	✗	✗	✗	✗	71.78%	41.59%	3.89 M	0.02 G
ViT-Lite-7/8	CT	✗	✗	✗	✗	83.38%	55.69%	3.74 M	0.06 G
ViT-Lite-7/4	CT	✗	✗	✗	✗	83.59%	58.43%	3.72 M	0.26 G
CVT-7/16	SP	✗	✗	✗	✗	72.26%	42.37%	3.89 M	0.02 G
CVT-7/8	SP	✗	✗	✗	✗	84.24%	55.49%	3.74 M	0.06 G
CVT-7/8	SP	✗	✗	✓	✗	87.15%	63.14%	3.74 M	0.06 G
CVT-7/4	SP	✗	✗	✗	✗	88.06%	62.06%	3.72 M	0.25 G
CVT-7/4	SP	✗	✗	✓	✗	91.72%	69.59%	3.72 M	0.25 G
CVT-7/4	SP	✗	✗	✓	✓	92.43%	73.01%	3.72 M	0.25 G
CVT-7/2	SP	✗	✗	✗	✗	84.80%	57.98%	3.76 M	1.18 G
CCT-7/7×1	SP	1	7×7	✗	✗	87.81%	62.83%	3.74 M	0.26 G
CCT-7/7×1	SP	1	7×7	✓	✗	91.85%	69.43%	3.74 M	0.26 G
CCT-7/7×1	SP	1	7×7	✓	✓	92.29%	72.46%	3.74 M	0.26 G
CCT-7/3×2	SP	2	3×3	✓	✓	93.65%	74.77%	3.85 M	0.29 G
CCT-7/3×1	SP	1	3×3	✓	✓	**94.47%**	**75.59%**	3.76 M	1.19 G

Table 9: Difference between **tuned** and not tuned runs in Table 8.

Hyper Param	**Not Tuned**	**Tuned**
MLP Dropout	0.1	0
MSA Dropout	0	0.1
Stochastic Depth	0	0.1

H. Additional experiments

H.1. Extended small-scale experiments

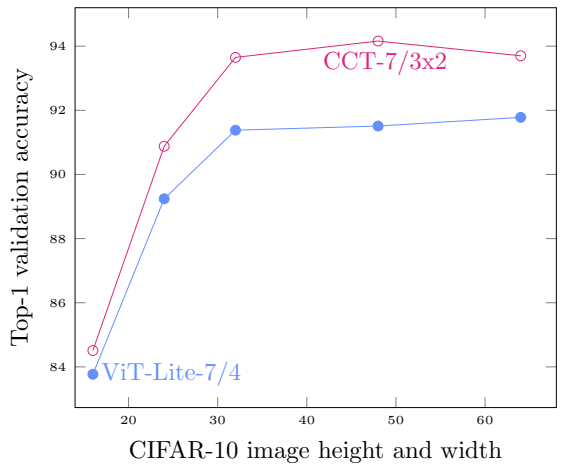
We present the extended version of Table 1 here with additional models in Table 11.

Table 10: Top-1 validation accuracy on text classification datasets. The number of parameters does not include the word embedding layer, because we use pretrained word-embeddings and freeze those layers while training.

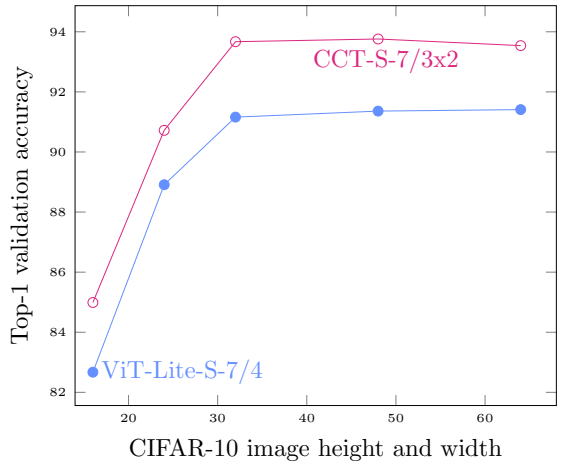
Model	AGNews	TREC	SST	IMDb	DBpedia	# Params
Vanilla Transformer Encoders						
Transformer-2	93.28%	90.40%	67.15%	86.01%	98.63%	1.086 M
Transformer-4	93.25%	92.54%	65.20%	85.98%	96.91%	2.171 M
Transformer-6	93.55%	92.78%	65.03%	85.87%	98.24%	4.337 M
Vision Transformers						
ViT-Lite-2/1	93.02%	90.32%	67.66%	87.69%	98.99%	0.238 M
ViT-Lite-2/2	92.20%	90.12%	64.44%	87.39%	98.88%	0.276 M
ViT-Lite-2/4	90.53%	90.00%	62.37%	86.17%	98.72%	0.353 M
ViT-Lite-4/1	93.48%	91.50%	66.81%	87.38%	99.04%	0.436 M
ViT-Lite-4/2	92.06%	90.42%	63.75%	87.00%	98.92%	0.474 M
ViT-Lite-4/4	90.93%	89.30%	60.83%	86.71%	98.81%	0.551 M
ViT-Lite-6/1	93.07%	91.92%	64.95%	87.58%	99.02%	3.237 M
ViT-Lite-6/2	92.56%	89.38%	62.78%	86.96%	98.89%	3.313 M
ViT-Lite-6/4	91.12%	90.36%	60.97%	86.42%	98.72%	3.467 M
Compact Vision Transformers						
CVT-2/1	93.24%	90.44%	67.88%	87.68%	98.98%	0.238 M
CVT-2/2	92.29%	89.96%	64.26%	86.99%	98.93%	0.276 M
CVT-2/4	91.10%	89.84%	62.22%	86.39%	98.75%	0.353 M
CVT-4/1	93.53%	92.58%	66.64%	87.27%	99.04%	0.436 M
CVT-4/2	92.35%	90.36%	63.90%	86.96%	98.93%	0.474 M
CVT-4/4	90.71%	90.14%	61.98%	86.77%	98.80%	0.551 M
CVT-6/1	93.38%	92.06%	65.94%	86.78%	99.02%	3.237 M
CVT-6/2	92.57%	91.14%	64.57%	86.61%	98.86%	3.313 M
CVT-6/4	91.35%	91.66%	61.63%	86.13%	98.76%	3.467 M
Compact Convolutional Transformers						
CCT-2/1x1	93.40%	90.86%	**68.76%**	88.95%	99.01%	0.238 M
CCT-2/2x1	93.38%	91.86%	67.19%	**89.13%**	99.04%	0.276 M
CCT-2/4x1	**93.80%**	91.42%	64.47%	88.92%	99.04%	0.353 M
CCT-4/1x1	93.49%	91.84%	68.21%	88.71%	99.03%	0.436 M
CCT-4/2x1	93.30%	**93.54%**	66.42%	88.94%	**99.05%**	0.474 M
CCT-4/4x1	93.09%	93.20%	66.57%	88.86%	99.02%	0.551 M
CCT-6/1x1	93.73%	91.22%	66.59%	88.81%	98.99%	3.237 M
CCT-6/2x1	93.29%	92.10%	65.02%	88.74%	99.02%	3.313 M
CCT-6/4x1	92.86%	92.96%	65.84%	88.68%	99.02%	3.467 M

Table 11: Top-1 comparisons. $\star$ were trained longer (see Tab 2).

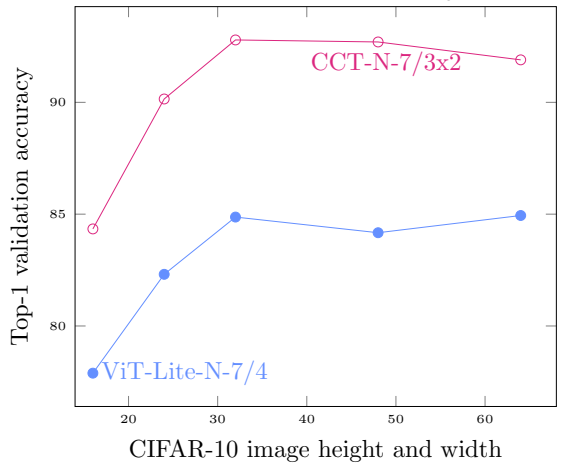
Model	C-10	C-100	Fashion	MNIST	# Params	MACs
Convolutional Networks (Designed for ImageNet)						
ResNet18	90.27%	66.46%	94.78%	99.80%	11.18 M	0.04 G
ResNet34	90.51%	66.84%	94.78%	99.77%	21.29 M	0.08 G
ResNet50	91.63%	68.27%	94.99%	99.79%	23.53 M	0.08 G
MobileNetV2/0.5	84.78%	56.32%	93.93%	99.70%	0.70 M	$<$ **0.01** G
MobileNetV2/1.0	89.07%	63.69%	94.85%	99.75%	2.24 M	0.01 G
MobileNetV2/1.25	90.60%	65.24%	95.05%	99.77%	3.47 M	0.01 G
MobileNetV2/2.0	91.02%	67.44%	95.26%	99.75%	8.72 M	0.02 G
Convolutional Networks (Designed for CIFAR)						
ResNet56[16]	94.63%	74.81%	95.25%	99.27%	0.85 M	0.13 G
ResNet110[16]	95.08%	76.63%	95.32%	99.28%	1.73 M	0.26 G
ResNet164-v1[17]	94.07%	74.84%	–	–	1.70 M	0.26 G
ResNet164-v2[17]	94.54%	75.67%	–	–	1.70 M	0.26 G
ResNet1k-v1[17]	92.39%	72.18%	–	–	10.33 M	1.55 G
ResNet1k-v2[17]	95.08%	77.29%	–	–	10.33 M	1.55 G
ResNet1k-v2*[17]	95.38%	–	–	–	10.33 M	1.55 G
Proxyless-G[5]	97.92%	–	–	–	5.7 M	–
Vision Transformers						
ViT-12/16	83.04%	57.97%	93.61%	99.63%	85.63 M	0.43 G
ViT-Lite-7/16	78.45%	52.87%	93.24%	99.68%	3.89 M	0.02 G
ViT-Lite-6/16	78.12%	52.68%	93.09%	99.66%	3.36 M	0.02 G
ViT-Lite-7/8	89.10%	67.27%	94.49%	99.69%	3.74 M	0.06 G
ViT-Lite-6/8	88.29%	66.40%	94.36%	99.73%	3.22 M	0.06 G
ViT-Lite-7/4	93.57%	73.94%	95.16%	99.77%	3.72 M	0.26 G
ViT-Lite-6/4	93.08%	73.33%	95.14%	99.74%	3.19 M	0.22 G
Compact Vision Transformers						
CVT-7/8	89.79%	70.11%	94.50%	99.70%	3.74 M	0.06 G
CVT-6/8	89.50%	68.80%	94.53%	99.74%	3.21 M	0.05 G
CVT-7/4	94.01%	76.49%	95.32%	99.76%	3.72 M	0.25 G
CVT-6/4	93.60%	74.23%	95.00%	99.75%	3.19 M	0.22 G
Compact Convolutional Transformers						
CCT-2/3×2	89.75%	66.93%	94.08%	99.70%	**0.28** M	0.04 G
CCT-4/3×2	91.97%	71.51%	94.74%	99.73%	0.48 M	0.05 G
CCT-6/3×2	94.43%	77.14%	95.34%	99.75%	3.33 M	0.25 G
CCT-7/3×2	95.04%	77.72%	95.16%	99.76%	3.85 M	0.29 G
CCT-6/3×1	95.70%	79.40%	95.41%	99.79%	3.23 M	1.02 G
CCT-7/3×1	96.53%	80.92%	**95.56%**	**99.82%**	3.76 M	1.19 G
CCT-7/3×1*	**98.00%**	**82.72%**	–	–	3.76 M	1.19 G



(a) Learnable PEs

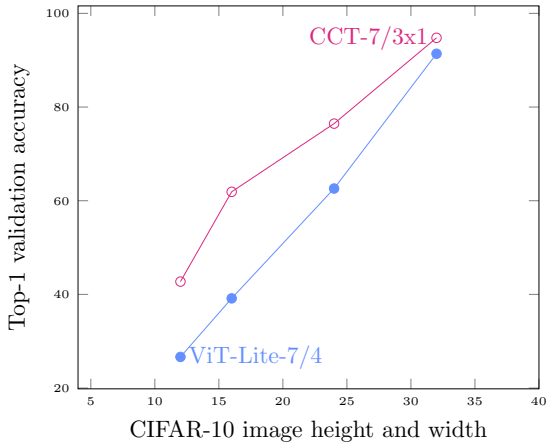


(b) Sinusoidal Positional Embedding

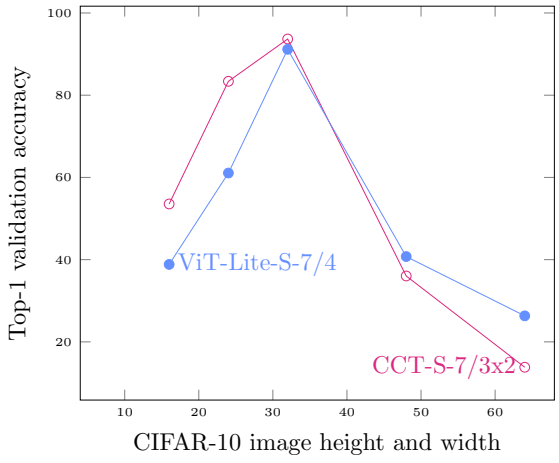


(c) No Positional Embedding

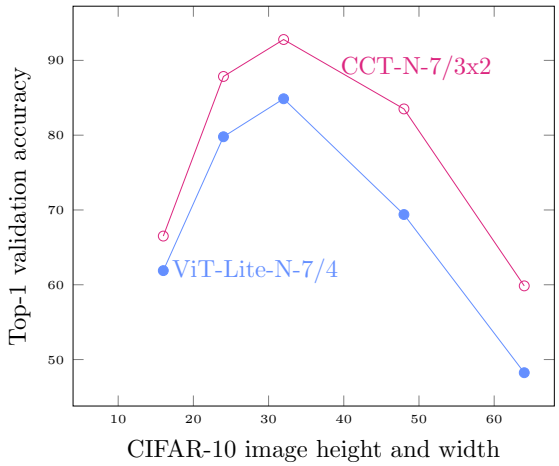
Figure 6: CIFAR-10 resolution vs top-1% validation accuracy (training from scratch). Images are square.



(a) Learnable PEs (only possible up to 32x32 without changing weights)



(b) Sinusoidal PEs



(c) None

Figure 7: CIFAR-10 resolution vs top-1% validation accuracy (inference only). Images are square.

MASKED MODELING DUO: LEARNING REPRESENTATIONS BY ENCOURAGING BOTH NETWORKS TO MODEL THE INPUT

Daisuke Niizumi, Daiki Takeuchi, Yasunori Ohishi, Noboru Harada, and Kunio Kashino

NTT Corporation, Japan

ABSTRACT

Masked Autoencoders is a simple yet powerful self-supervised learning method. However, it learns representations indirectly by reconstructing masked input patches. Several methods learn representations directly by predicting representations of masked patches; however, we think using all patches to encode training signal representations is suboptimal. We propose a new method, Masked Modeling Duo (M2D), that learns representations directly while obtaining training signals using only masked patches. In the M2D, the online network encodes visible patches and predicts masked patch representations, and the target network, a momentum encoder, encodes masked patches. To better predict target representations, the online network should model the input well, while the target network should also model it well to agree with online predictions. Then the learned representations should better model the input. We validated the M2D by learning general-purpose audio representations, and M2D set new state-of-the-art performance on tasks such as UrbanSound8K, VoxCeleb1, AudioSet20K, GTZAN, and SpeechCommandsV2.

***Index Terms*—** Self-supervised learning, Masked Autoencoders, Masked Image Modeling, Masked Spectrogram Modeling

1. INTRODUCTION

Recently, self-supervised learning (SSL) methods using masked image modeling (MIM) have progressed and yielded promising results in the image domain. Among them, Masked Autoencoders [1] (MAE) have inspired numerous subsequent studies and influenced not only the image domain [2–5] but also the audio domain [6–9].

An MAE effectively learns a representation by reconstructing a large number (i.e., 75%) of masked input patches using a small number of visible patches, encouraging the learned representation to model the input. However, it learns representations indirectly by minimizing the loss between the original input and the reconstructed result, which may not be optimal for learning a representation.

In contrast, several previous methods [2, 3, 10] achieve direct learning of representations, typically by using a momentum encoder in a Siamese architecture [11] to obtain the masked patch representations as a training signal. In this case, all input patches are used to encode these representations, not encouraging to model the input.

We hypothesize that the learned representation would become more useful if the training signal were encoded using only masked patches instead of all patches in order to encourage modeling the input in the training signal. While an MAE effectively encourages modeling the input signal by limiting the number of visible patches fed to the encoder, using all the input patches to obtain a training signal does not benefit from the inductive bias of the MAE.

In this paper, we propose a new method, Masked modeling duo (M2D), that learns representations directly by predicting the representations of masked patches from visible patches only. As illustrated in Fig.1, the target representations are encoded from only the masked patches, not from all the input patches as they are in the previous methods [2, 3, 10]. Although our method adds a target momentum encoder to the MAE, the entire framework remains simple.

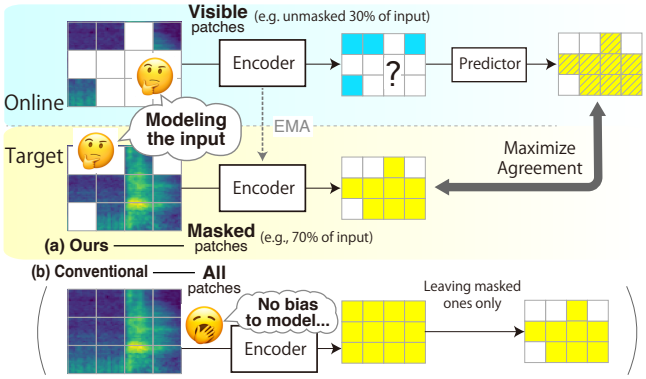


Fig. 1. M2D pre-training scenario. The online network encodes visible patches and predicts masked patch representations, while the target network encodes masked patches. The M2D maximizes the agreement between these two outputs to learn representations. We provide only the masked patches to the target illustrated as (a), unlike conventional methods (e.g., data2vec [10]) depicted as (b), encouraging representations to model the input from both the online and target networks.

The M2D promotes complementary input modeling by feeding mutually exclusive patches to its networks. For example, to reduce the prediction error for a heartbeat audio input consisting of two sounds (S1 and S2), the online network should encode the given visible patches around S2 into representations modeled as part of the whole heartbeat in order to predict the representations of masked patches around S1. Conversely, the masked patch representations around S1 encoded by the target network are more likely to agree with the prediction if it is encoded as part of the whole heartbeat. Therefore, our method encourages input modeling from both sides.

In our experiments, we validated our method by learning a general-purpose audio representation using an audio spectrogram as input and confirmed the effectiveness of learning the representation directly and providing only masked patches to the target. In addition, M2D set new state-of-the-art (SOTA) performance on several audio tasks. Our code is available online[1].

2. RELATED WORK

This study was inspired by MAE [1] for an MIM and Bootstrap Your Own Latent [12] (BYOL) as a framework for directly learning latent representations using a target network. An MAE learns to re-

[1] https://github.com/nttcslab/m2d

construct the input data, whereas our M2D learns to predict masked latent representations. BYOL differs from ours in that it is a framework for learning representations invariant to data augmentation.

SIM [2], MSN [3], and data2vec [10] learn to predict masked patch representations using a target network, but, unlike ours, all input patches are fed to the target. CAE [4] and SplitMask [5] encode target representation using only masked patches, which is similar to ours but without the use of a target network. While SIM, MSN, CAE, and SplitMask learn image representations, data2vec also learns audio representations.

In this work, we experimented through learning general-purpose audio representations. To learn speech and audio, various methods learn representations using masked input, such as Mockingjay [13], wav2vec2 [14], HuBERT [15], and BigSSL [16] for speech, and SSAST [17] for audio. Methods more closely related to ours are MAE-AST [7], MaskSpec [8], MSM-MAE [9], and Audio-MAE [6], which adapt MAE to learn audio representations. However, they differ from our method in that they do not use a target network.

Other SSL methods for learning audio representations include Wang et al. [18] and DeLoRes [19], and especially BYOL-A [20], BYOL-S [21], and ATST [22], which use BYOL as the learning framework; they do not mask the input. For supervised learning, AST [23], EAT [24], PaSST [25], and HTS-AT [26] have shown SOTA performance.

3. MASKED MODELING DUO

Our method learns representations by using only visible patches to predict the masked patch representations. As shown in Fig. 2, it consists of two networks, referred to as the online and target networks.

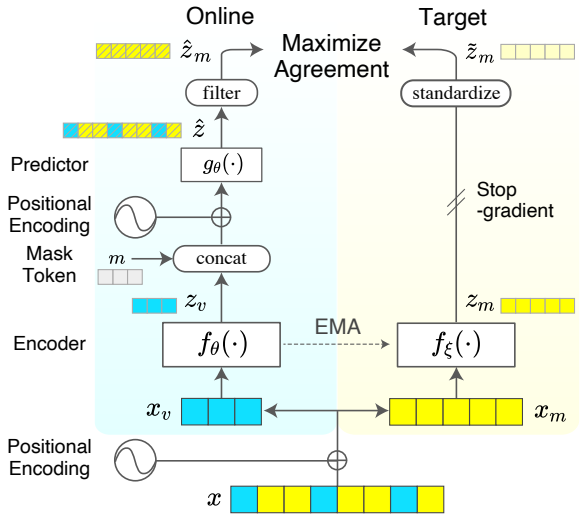


Fig. 2. Overview of the M2D framework.

Processing input The framework partitions the input data x (audio spectrogram, image, etc.) into a grid of patches, adds positional encoding, and randomly selects a number of patches according to a masking ratio as masked patches x_m (e.g., 60% of the input) and the rest as visible patches x_v (e.g., the remaining 40%). While we use the same positional encoding as MAE [1], we tested various masking ratios as discussed in Section 4.4.

Online and target networks The online network, defined by a set of weights θ, encodes the visible patches x_v using the online encoder f_θ into the representation $z_v = f_\theta(x_v)$. It concatenates shared, learnable masked tokens m to z_v, adds the position encoding p, and predicts $\hat{z}$, representations of entire input patches, using the predictor g_θ.

$$\hat{z} = g_\theta(\text{concat}(z_v, m) + p) \tag{1}$$

It then filters the prediction result $\hat{z}$ to output $\hat{z}_m = \{\, \hat{z}[i] \mid i \in I_M \,\}$, containing only masked patch representations, where I_M is the set of indices of the masked patches.

The target network is defined by parameter ξ and consists only of momentum encoder f_ξ, which is identical to the online encoder except for the parameter. The network encodes masked patches x_m using f_ξ to output the representation $z_m = f_\xi(x_m)$. We then standardize z_m to $\tilde{z}_m = (z_m - \text{mean}(z_m))/\sqrt{\text{var}(z_m)}$, for stabilizing the training, which we empirically confirmed in preliminary experiments, rather than for performance gain as in MAE.

Calculating loss The loss is calculated using the standardized target output $\tilde{z}_m$ as a training signal against the online prediction output $\hat{z}_m$. Inspired by BYOL [12], we calculate the loss L by the mean square error (MSE) of l_2-normalized $\hat{z}_m$ and $\tilde{z}_m$.

$$L \triangleq ||l_2(\hat{z}_m) - l_2(\tilde{z}_m)||_2^2 = 2 - 2 \cdot \frac{\langle \hat{z}_m, \tilde{z}_m \rangle}{||\hat{z}_m||_2 \cdot ||\tilde{z}_m||_2}, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

Updating network parameters Our framework updates parameters θ and ξ after each training step. It updates θ only by minimizing the loss L as depicted by the stop-gradient in Fig. 2, whereas updating ξ is based on a slowly moving exponential average of θ with a decay rate τ:

$$\xi \leftarrow \tau\xi + (1 - \tau)\theta \tag{3}$$

It has been empirically shown that stop-gradient operation can avoid collapsing to an uninformative solution, and the moving-average behavior may lead to learning effective representations [11]. After the training, we transfer only the f_θ as a pre-trained model.

4. EXPERIMENTS

We validated the M2D step-by-step experiments that examined the effectiveness of learning representations directly by comparing our M2D with MAE (Section 4.2), the effectiveness of feeding only masked patches to the target (Section 4.3), the impact of various masking ratios (Section 4.4), and comparing ours with SOTA (Section 4.5).

In all experiments, we applied our M2D to masked spectrogram modeling (MSM) [9], with an audio spectrogram as input to learn general-purpose audio representations. We evaluated the performance of pre-trained models in both a linear evaluation and fine-tuning on a variety of audio downstream tasks spanning environmental sounds, speech, and music.

4.1. Experimental Setup

We mainly focused on comparing M2D with an MAE, then adapted MAE implementations and settings with as few changes as possible. We implemented an additional target network on top of the MAE code and adopted the MAE decoder as our predictor g_θ without changes. We used vanilla ViT-Base [27] with a 768-d output feature as our encoders (f_θ and f_ξ) and fixed the patch size to 16×16 for all experiments. We tested with masking ratios of 0.6 and 0.7, which showed good performance in preliminary experiments.

We used the MSM-MAE [9] as an MAE for comparison, an MAE variant optimized for MSM by making the decoder smaller

Table 1. Fine-tuning and linear evaluation results. AS20K result is an mAP, and all others are accuracies (%) with 95% CI.

	(a) Fine-tuning				(b) Linear evaluation									
					Env. sound tasks		Speech tasks				Music tasks			
Model	AS20K	ESC-50	SPCV2	VC1	ESC-50	US8K	SPCV2	VC1	VF	CRM-D	GTZAN	NSynth	Surge	Avg.
MSM-MAE [9]	36.7 ± 0.5	94.0 ± 0.2	98.4 ± 0.1	**95.3 ± 0.1**	88.6 ± 1.5	86.3 ± 0.3	94.5 ± 0.2	72.2 ± 0.2	97.5 ± 0.1	70.2 ± 1.1	78.4 ± 2.8	75.9 ± 0.2	**42.5 ± 0.7**	78.5 ± 0.8
M2D ratio=0.6	36.8 ± 0.1	94.7 ± 0.3	**98.5 ± 0.0**	94.8 ± 0.1	89.7 ± 0.2	**87.6 ± 0.2**	**95.4 ± 0.1**	**73.1 ± 0.1**	**97.9 ± 0.1**	**71.7 ± 0.3**	83.3 ± 1.0	75.3 ± 0.1	41.0 ± 0.2	**79.5 ± 0.3**
M2D ratio=0.7	**37.4 ± 0.1**	**95.0 ± 0.2**	98.5 ± 0.1	94.4 ± 1.3	**89.8 ± 0.3**	87.1 ± 0.3	94.5 ± 0.1	71.3 ± 0.4	97.7 ± 0.1	71.6 ± 0.3	**83.9 ± 1.4**	**76.9 ± 1.3**	41.8 ± 0.4	79.4 ± 0.5

with four layers, six heads, and a width (embedding dimension) of 384-d. Other parameters, including a default masking ratio of 0.75, were the same as in the original MAE. Preliminary experiments verified that the MSM-MAE outperforms a vanilla MAE with an eight-layer decoder.

We employed the MSM-MAE feature calculation in evaluations to optimize M2D representations for MSM as in MSM-MAE. The MSM-MAE outputs $z' \in R^{B \times N_T \times N_F D}$ by concatenating the features of frequency bins for each time frame, instead of simply averaging the $z \in R^{B \times N_F N_T \times D}$ output from ViT, where B is batch size, N_F is the number of patches along frequency, N_T is the number of patches along time, and D is a patch feature dimension. Then, we summarized an audio sample level feature $z'' = 1/N_T \sum_{t=1}^{N_T} z'[t]$, averaging z' over time. The z'' becomes a 3,840-d feature, where $D = 768$ and $N_F = 5$.

We preprocessed audio samples to a log-scaled mel spectrogram with a sampling frequency of 16,000 Hz, window size of 25 ms, hop size of 10 ms, and mel-spaced frequency bins $F = 80$ in the range of 50 to 8,000 Hz and normalized them with a dataset statistics. All downstream task audios were cropped to the dataset's average duration or added with zero padding at the end. Long audio samples were split into the input length of the model without overlapping, encoded to z' each, concatenated along time, and averaged over time to an audio sample level feature z''.

Pre-training details The input audio duration of the ViT used in the experiments was set to 6s, the same as ATST, for comparison with the SOTA methods. The input spectrogram has a size of 80×608 (Freq. bins $\times$ Time frames), making $N_F = 80/16 = 5$ and $N_T = 608/16 = 38$ with a patch size of 16×16.

We set the number of epochs as 300, warm-up epochs as 20, batch size as 2048, and the base learning rate as 3e-4. All other settings were the same as in the MAE, including the learning rate scheduling and optimizer. The EMA decay rate τ for the target network update was linearly interpolated from 0.99995 at the start of training to 0.99999 at the end. We used AudioSet [28] as a pre-training dataset with 2,005,132 samples (5,569 h) of 10s audio from the balanced and unbalanced train segments. We randomly cropped 6s audio from a 10s sample. All these settings were common in the M2D and MSM-MAE pre-training, except for the EMA decay rate.

Linear evaluation details In the linear evaluation, we trained a *linear classifier* on top of *frozen* pre-trained models, and tested the performance on a variety of downstream tasks.

All evaluation details and downstream tasks are the same as in our previous study [20]. Tasks include environmental sound classification ESC-50 [29] and UrbanSound8K [30] (US8K), speech-command classification Speech Commands V2 [31] (SPCV2), speaker identification VoxCeleb1 [32] (VC1), language identification VoxForge [33] (VF), speech emotion recognition CREMA-D [34] (CRM-D), music genre recognition GTZAN [35], musical instrument classification NSynth [36], and a pitch audio classification Pitch Audio Dataset (Surge synthesizer) [37]. All the tasks are classification problems, and all the results are accuracies.

Fine-tuning details We used the tasks commonly used in previous studies: ESC-50, SPCV2, and VC1, the same as in the linear evaluation, plus AudioSet20K (AS20K), which learns only the balanced train segments of AudioSet [28] and results in a mean average pooling (mAP) of multi-label classification with 527 classes.

The fine-tuning pipeline follows ATST. A linear classifier was added on top of the pre-trained model to train the entire network. All evaluations were trained for 200 epochs, and the learning rate was optimized for each task and scheduled with cosine annealing [38] after five epochs of warm-up. We used SGD and AdamW for the optimizer, Mixup [20,39], Random Resize Crop (RRC) [20] for data augmentation, and Structured Patchout (SPO) proposed in PaSST [25] that masks patches during training. When using the SPO, we used 768-d features calculated by averaging ViT outputs over time because the MSM-MAE feature calculation is not applicable to the masked patches. Table 2 summarizes the settings.

Table 2. Fine-tuning settings

Parameter	AS20K	ESC-50	SPCV2	VC1
LR	1.0	0.5	0.5	0.001
Optimizer	SGD	SGD	SGD	AdamW
Mixup	0.3	0.0	0.3	0.0
RRC	✓	✓	✓	-
SPO ratio	0.5	0.5	0.5	0.0

4.2. Validation of Learning Representations Directly

We validated the effectiveness of learning representations directly instead of through the reconstruction task by comparing the M2D with MAE. Note that the pre-trained models shared exactly the same ViT, enabling us to compare the difference of pre-training schemes.

Table 1 shows the comparison results with the MAE. Both (a) fine-tuning and (b) linear evaluation results show that the M2D improves the performance of the conventional MAE in most tasks. However, the performance degrades on Surge (pitch classification) in the linear evaluation and on VoxCeleb1 (speaker classification) in fine-tuning. For these tasks, pitch information is considered important, while for the other tasks, pitch is considered less important. This suggests that the M2D representations are less sensitive to pitch than the MAE ones, and thus performance is degraded in Surge and VoxCeleb1, while it is improved in the other tasks that typically discriminate events regardless of pitch.

These results also suggest that learning by reconstructing data, as in the MAE, may facilitate detailed and local representations, while learning latent representations directly may facilitate more abstract representations.

4.3. Validation of Input to the Target

We validate the effectiveness of feeding masked patches only to the target by comparing our proposal with providing all patches to the target, which is employed in methods such as data2vec [10]. We compare the average results of the linear evaluation. The results for both masking ratios in Table 3 shows that giving the target only the masked patch improves the result more than giving it all the patches, confirming our hypothesis.

Table 3. Linear evaluation average results (%) for target inputs.

	Input ratio		Masking ratio (r)	
Target input	Online	Target	0.6	0.7
Masked patches only (ours)	$1.0 - r$	r	**79.5 ± 0.3**	**79.4 ± 0.5**
All patches (conventional)	$1.0 - r$	1.0	79.1 ± 0.3	79.3 ± 0.3

Table 4. Linear evaluation comparison with SOTA models (%). Supervised learning methods and non-standard linear evaluation results are grayed out as a reference.

	Env. sound tasks		Speech tasks				Music tasks		
Model	ESC-50	US8K	SPCV2	VC1	VF	CRM-D	GTZAN	NSynth	Surge
Wav2Vec2 [14] †	57.6 ± 0.8	66.9 ± 0.4	**96.6** ± 0.0	40.9 ± 0.6	**99.2** ± 0.1	65.5 ± 1.7	57.8 ± 1.3	56.6 ± 0.6	15.2 ± 0.9
DeLoRes-M [19]	-	82.7	89.7	45.3	88.0	-	-	75.0	-
SF NFNet-F0 [18]	91.1	-	93.0	64.9	90.4	-	-	**78.2**	-
BYOL-A [20]	83.2 ± 0.6	79.7 ± 0.5	93.1 ± 0.4	57.6 ± 0.2	93.3 ± 0.3	63.8 ± 1.0	70.1 ± 3.6	73.1 ± 0.8	37.6 ± 0.3
ATST Base [22] †	**92.9** ± **0.3**	84.1	95.1	72.0	97.4 ± 0.2	68.6 ± 0.2	76.4 ± 1.8	75.6	37.7 ± 0.2
MSM-MAE [9]	88.6 ± 1.5	86.3 ± 0.3	94.5 ± 0.2	72.2 ± 0.2	97.5 ± 0.1	70.2 ± 1.1	78.4 ± 2.8	75.9 ± 0.2	**42.5** ± **0.7**
M2D ratio=0.6	89.7 ± 0.2	**87.6** ± **0.2**	95.4 ± 0.1	**73.1** ± **0.1**	97.9 ± 0.1	**71.7** ± **0.3**	83.3 ± 1.0	75.3 ± 0.1	41.0 ± 0.2
M2D ratio=0.7	89.8 ± 0.3	87.1 ± 0.3	94.5 ± 0.1	71.3 ± 0.4	97.7 ± 0.1	71.6 ± 0.3	**83.9** ± **1.4**	76.9 ± 1.3	41.8 ± 0.4
ConformerXL-P Non-RA [16]	-	-	**97.5**	50.3	**99.7**	**88.2**	-	-	-
AST-Fusion#5#12 [40]	**94.2**	85.5	80.4	24.9	87.6	60.7	82.9	77.6	34.6
BYOL-S [21]	-	-	-	-	-	76.9	-	-	-

† Underlined results were obtained in this study and [20] using publicly available pre-trained models.

Table 5. Fine-tuning comparison with SOTA models. Supervised learning method results are grayed out as a reference.

Model	AS20K mAP	ESC-50 acc(%)	SPCV2 acc(%)	VC1 acc(%)
DeLoRes-M [19]	-	-	96.0	62.0
MAE-AST Patch/Frame [7]	30.6	90.0	98.0	63.3
MaskSpec/MaskSpec-small [8]	32.3	90.7	97.7	-
SSAST 250/400 [17]	31.0	88.8	98.2	66.6
data2vec [10]	34.5	-	-	-
Audio-MAE (local) [6]	37.1	94.1	98.3	94.8
ATST Base [22]	**37.4**	-	98.0	94.3
MSM-MAE [9]	36.7 ± 0.5	94.0 ± 0.2	98.4 ± 0.1	**95.3** ± **0.1**
M2D ratio=0.6	36.8 ± 0.1	94.7 ± 0.3	**98.5** ± **0.0**	94.8 ± 0.1
M2D ratio=0.7	**37.4** ± **0.1**	**95.0** ± **0.2**	98.5 ± 0.1	94.4 ± 1.3
AST (Single), AST-P/S [23]	34.7	95.6	98.11	-
EAT-S/M [24]	-	96.3	98.15	-
PaSST [25]	-	96.8	-	-
HTS-AT [26]	-	**97.0**	98.0	-

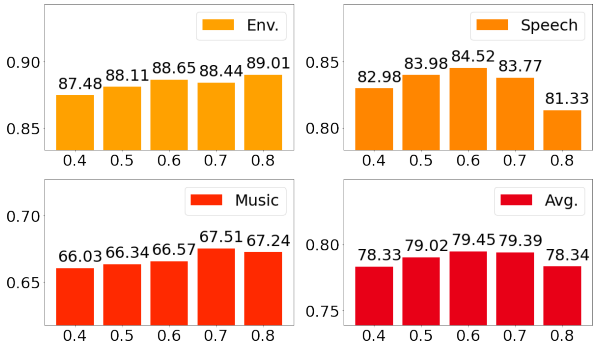


Fig. 3. Masking ratio ablations: linear evaluation results (%).

4.4. Masking Ratio Ablations

We evaluate the impact of various masking ratios using linear evaluation. Fig. 3 shows the results for the three task groups and the overall average, showing the best average result (Avg.) at 0.6. However, the three groups show different trends, indicating that the optimal masking ratio depends on the task. The environmental sound tasks show the best result at 0.8, while the speech tasks have a peak of 0.6. The music task shows the best result at 0.7. Thus, it is difficult to set a single common optimal value.

We suspect that these optimal masking ratios are due to differences in the density of the target information, as discussed in the MAE paper. The sounds in the speech tasks consist of successive phonemes, making many masks difficult to predict, while the environmental and music tasks include long continuous sounds (e.g., the sound of an air conditioner or instrumental sounds with long notes), making them easier to predict even at higher masking ratios. The results may indicate that the usefulness of the learned representations for various sounds can be varied and even controlled by masking ratios.

4.5. Comparison with SOTA

We compare the M2D with SOTA methods in this section. Linear evaluation results shown in Table 4 confirm that our method outperforms existing SSL methods in four tasks: UrbanSound8K, VoxCeleb1, CREMA-D, and GTZAN. Among them, M2D outperforms all previous methods with 87.6% on UrbanSound8K, 73.1% on VoxCeleb1, and 83.9% on GTZAN.

The fine-tuning results in Table 5 show that our method outperforms existing SSL methods on ESC-50 and Speech commands V2 tasks and gives the same SOTA result as ATST [22] on AudioSet20K. In particular, the result of 98.5% on Speech commands V2 outperforms all previous methods, including supervised learning. MSM-MAE [9] shows a new SOTA VoxCeleb1 result, and M2D with a masking ratio of 0.6 shows a comparable result.

While the M2D showed SOTA performance on many tasks, it underperforms supervised learning methods (EAT [24], PaSST [25], and HTS-AT [26]) on ESC-50 and speech models (Wav2Vec2 [14], BigSSL [16], and BYOL-S [21]) on the speech tasks, suggesting a future research direction. In summary, the experiments demonstrate the effectiveness of M2D among SOTA methods.

5. CONCLUSION

In this study, we proposed a new method, Masked Modeling Duo (M2D), to learn representations directly by predicting masked patch representations using two networks. Unlike previous methods, we encode training signal representations using only masked patches rather than all input patches. We made the M2D so that it encourages both online and target networks to model the entire input in visible and masked patch representations, respectively, achieving an effective representation. To evaluate our method, we applied it to masked spectrogram modeling to learn general-purpose audio representations with an audio spectrogram as input. We evaluated the performance of our method on a variety of downstream tasks. Experiments validated the effectiveness of our method, and M2D showed state-of-the-art results on UrbanSound8K, VoxCeleb1, CREMA-D, and GTZAN in linear evaluation, and on AudioSet20K, ESC-50, and Speech commands V2 in fine-tuning. Our code is available online.

6. REFERENCES

[1] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022.

[2] C. Tao, X. Zhu, G. Huang, Y. Qiao, X. Wang, and J. Dai, "Siamese image modeling for self-supervised vision representation learning," *arXiv preprint arXiv:2206.01204*, 2022.

[3] M. Assran, M. Caron, I. Misra, P. Bojanowski, F. Bordes, P. Vincent, A. Joulin, M. Rabbat, and N. Ballas, "Masked siamese networks for label-efficient learning," in *ECCV*, 2022.

[4] X. Chen, M. Ding, X. Wang, Y. Xin, S. Mo, Y. Wang, S. Han, P. Luo, G. Zeng, and J. Wang, "Context autoencoder for self-supervised representation learning," *arXiv preprint arXiv:2202.03026*, 2022.

[5] A. El-Nouby, G. Izacard, H. Touvron, I. Laptev, H. Jegou, and E. Grave, "Are large-scale datasets necessary for self-supervised pre-training?," *arXiv preprint arXiv:2112.10740*, 2021.

[6] P.-Y. Huang, H. Xu, J. Li, A. Baevski, M. Auli, W. Galuba, F. Metze, and C. Feichtenhofer, "Masked autoencoders that listen," in *NeurIPS*, 2022.

[7] A. Baade, P. Peng, and D. Harwath, "MAE-AST: Masked Autoencoding Audio Spectrogram Transformer," in *Interspeech*, 2022, pp. 2438–2442.

[8] D. Chong, H. Wang, P. Zhou, and Q. Zeng, "Masked spectrogram prediction for self-supervised audio pre-training," *arXiv preprint arXiv:2204.12768*, 2022.

[9] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Masked Spectrogram Modeling using Masked Autoencoders for Learning General-purpose Audio Representation," in *HEAR: Holistic Evaluation of Audio Representations (NeurIPS 2021 Competition)*, 2022, vol. 166, pp. 1–24.

[10] A. Baevski, W.-N. Hsu, Q. Xu, A. Babu, J. Gu, and M. Auli, "data2vec: A general framework for self-supervised learning in speech, vision and language," in *ICML*, 2022, pp. 1298–1312.

[11] X. Chen and K. He, "Exploring simple siamese representation learning," in *CVPR*, Jun 2021.

[12] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - a new approach to self-supervised learning," in *NeurIPS*, 2020.

[13] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP*, 2020, pp. 6419–6423.

[14] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *NeurIPS*, 2020.

[15] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "HuBERT: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Trans. Audio, Speech, Language Process.*, p. 3451–3460, 2021.

[16] Y. Zhang, D. S. Park, W. Han, J. Qin, A. Gulati, J. Shor, A. Jansen, Y. Xu, Y. Huang, S. Wang, et al., "BigSSL: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition," *IEEE J. Sel. Top. Signal Process.*, vol. 16, no. 6, pp. 1519–1532, 2022.

[17] Y. Gong, C.-I. Lai, Y.-A. Chung, and J. Glass, "SSAST: Self-supervised audio spectrogram transformer," in *AAAI*, 2022, vol. 36, pp. 10699–10709.

[18] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira, and A. van den Oord, "Towards learning universal audio representations," in *ICASSP*, 2022, pp. 4593–4597.

[19] S. Ghosh, A. Seth, and S. Umesh, "Decorrelating feature spaces for learning general-purpose audio representations," *IEEE J. Sel. Topics Signal Process.*, vol. 16, no. 6, pp. 1402–1414, 2022.

[20] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for Audio: Exploring pre-trained general-purpose audio representations," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 31, pp. 137–151, 2023.

[21] N. Scheidwasser-Clow, M. Kegler, P. Beckmann, and M. Cernak, "SERAB: A multi-lingual benchmark for speech emotion recognition," in *ICASSP*, 2022, pp. 7697–7701.

[22] X. LI and X. Li, "ATST: Audio Representation Learning with Teacher-Student Transformer," in *Interspeech*, 2022, pp. 4172–4176.

[23] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," in *Interspeech*, 2021, pp. 571–575.

[24] A. Gazneli, G. Zimerman, T. Ridnik, G. Sharir, and A. Noy, "End-to-end audio strikes back: Boosting augmentations towards an efficient audio classification network," *arXiv preprint arXiv:2204.11479*, 2022.

[25] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," in *Interspeech*, 2022, pp. 2753–2757.

[26] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "HTS-AT: A hierarchical token-semantic audio transformer for sound classification and detection," in *ICASSP*, 2022, pp. 646–650.

[27] 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," in *ICLR*, 2021.

[28] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *ICASSP*, 2017, pp. 776–780.

[29] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *ACM-MM*, 2015, pp. 1015–1018.

[30] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM-MM*, 2014, pp. 1041–1044.

[31] P. Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition," *arXiv preprint arXiv::1804.03209*, Apr. 2018.

[32] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Interspeech*, 2017, pp. 2616–2620.

[33] K. MacLean, *"Voxforge"*, 2018, Available at http://www.voxforge.org/home.

[34] H. Cao, D. G. Cooper, M. K. Keutmann, R. C. Gur, A. Nenkova, and R. Verma, "CREMA-D: Crowd-sourced emotional multimodal actors dataset," *IEEE Trans. Affective Comput.*, vol. 5, no. 4, 2014.

[35] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Speech Audio Process.*, vol. 10, no. 5, 2002.

[36] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *ICML*, 2017.

[37] J. Turian, J. Shier, G. Tzanetakis, K. McNally, and M. Henry, "One billion audio sounds from GPU-enabled modular synthesis," in *DAFx2020*, 2021.

[38] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *ICLR*, 2017.

[39] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *ICLR*, 2018.

[40] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Composing general audio representation by fusing multilayer features of a pre-trained model," in *EUSIPCO*, 2022, pp. 200–204.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.

NOTE: The following appendix will not be on the ICASSP paper. Please cite the arXiv version if you reference the appendix of this paper.

A. EXPERIMENTS WITH IMAGES

We validate the effectiveness of our M2D for images by conducting evaluations on ImageNet [41]. We pre-trained on ImageNet-1K [41], followed by fine-tuning. We report top-1 validation accuracy for a single crop image of 224×224 same as in the MAE [1].

We used the MAE code as a base, as in Section 4 using audio, and made minimal changes to it, allowing comparison of differences only in the experimental subjects of interest. We also used ViT-Base [27] for the backbone. For pre-training, we set the number of epochs as 300 and the batch size as 2048. All other settings were the same as in the MAE, including the masking ratio of 0.75. The EMA decay rate τ for the target network update was also the same as in Section 4, linearly interpolated from 0.99995 at the start of training to 0.99999 at the end. For fine-tuning, we used the same source code and parameters as in the MAE.

We compare three models, MAE, M2D, and an M2D variant, that feed all patches to the target encoder. Table 6 shows the results. The results show the performance of the M2D variant with all patches input to the target, which is conventional, is comparable to MAE, and our M2D outperforms these methods, validating that our method is also effective on images in addition to audios.

Table 6. Fine-tuning results on ImageNet-1K.

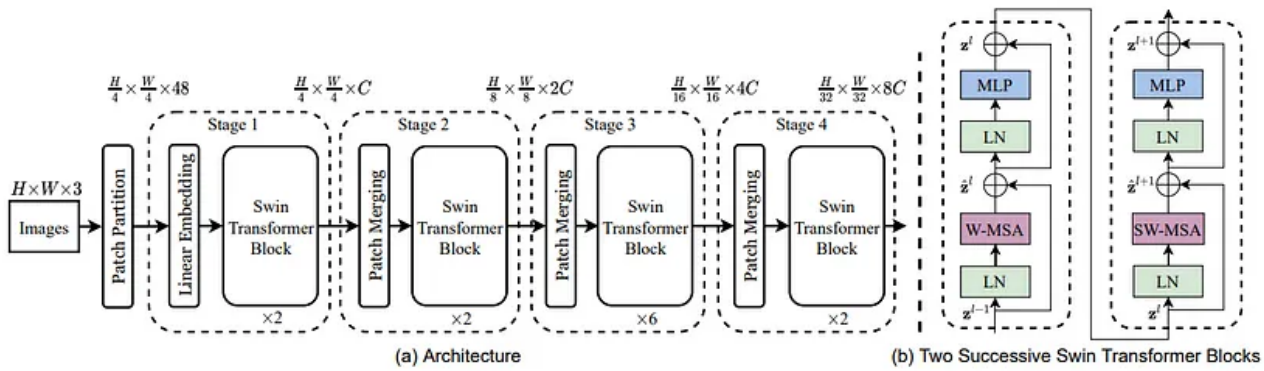
Model	Target encoder	Target input	Top-1 acc(%)
MAE		-	83.22 $\pm$ 0.024
M2D variant (conventional)	✓	All patches	83.22 $\pm$ 0.085
M2D (ours)	✓	Masked patches only	83.35 $\pm$ 0.211

Swin-Transformer from Scratch in PyTorch

Nickd · Follow

Published in Python in Plain English

12 min read · Sep 12, 2023



Overall Architecture

Introduction

Due to the promising performance of the transformer in the field of Natural Language Processing (NLP), researchers have looked to incorporate the transformer architecture into the field of Computer Vision. Although the Vision Transformer (ViT) from the paper “An Image is Worth 16x16 Words” has showed promising results for some high-level vision tasks such as image classification, the global attention mechanism in ViT scales with quadratic computational complexity with respect to the resolution of an image which is not sustainable for tasks that work with images in the high resolution space. To combat this issue, Microsoft proposed the Swin-Transformer which features a local attention mechanism based on shifting windows whose computational complexity scales linearly and could serve as an all-purpose backbone for general vision tasks.

The purpose of this article is to build the Swin-Transformer architecture from scratch using PyTorch. Readers of this article will be able to

gain both a comprehensive understanding of the Swin-Transformer architecture and a better understanding on how to implement academic papers. To get the most out of this article, feel free to read the original academic paper https://arxiv.org/abs/2103.14030 before proceeding to the code/ explanation.

Also a note: This tutorial assumes basic familiarization with PyTorch and Transformers as this tutorial deals with a complex attention mechanism.

Overview

This section will provide a high-level summary of the model. We will then go deeper and look at each piece in detail with code. The model starts by splitting an image into p x p non-overlapping patches with a linear embedding exactly like ViT. Our image transforms from (h,w,c) to (h/p,w/p,c*p**2) from patch partitioning, and then to (h/p * w/p, C) after the linear projection. We treat the h*w patches as the tokens of the

transformer sequence and C as our embedding dimension.

We then send the image into a series of transformer blocks and patch merging blocks. The transformer blocks feature attention computed locally within windows with an alternating shifted window configuration to allow the model to gain global input information. The attention computation also features relative embeddings to directly encode positional information into the model.

In between these transformer blocks we use patch merging layers to decrease the height and width of our images while increasing the channel size/embedding dimension of the model. This is similar to how CNN's transform the input as you go deeper into the model. The goal is to have the Swin-Transformer work as a computer vision backbone for various vision tasks just like how the CNN has formally been used in computer vision.

We will build each component of the architecture and connect them together at the end. We will follow this sequence of steps:

1. Imports
2. Patch Partition + Linear Embedding
3. Patching Merging Layer
4. Shifted Window Attention Mechanism
5. Relative Embeddings
6. Transformer Encoder + Final Architecture

Imports

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import math
from einops import rearrange
```

The first 3 imports we need are the basic PyTorch imports: torch, torch.nn, and torch.nn.functional for basic tensor operations and neural network modules. The math import is for the square root normalization in attention. Einops rearrange is for reshaping and permuting tensors in a read friendly and efficient way. For people unfamiliar with einops rearrange, you can refer to their documents here https://einops.rocks/api/rearrange/.

Patch Partition + Linear Embedding

> *"It first splits an input RGB image into non-overlapping patches by a patch splitting module, like ViT. Each patch is treated as a "token" and its feature is set as a concatenation of the raw pixel RGB values. In our implementation, we use a patch size of 4 × 4 and thus the feature dimension of each patch is 4 × 4 × 3 = 48. A linear embedding layer is applied on this raw-valued feature to project it to an arbitrary dimension (denoted as C)."*

```
class SwinEmbedding(nn.Module):

    '''
    input shape -> (b,c,h,w)
    output shape -> (b, (h/4 * w/4), C)
    '''

    def __init__(self, patch_size=4, C=96):
        super().__init__()
        self.linear_embedding = nn.Conv2d(3, C
        self.layer_norm = nn.LayerNorm(C)
        self.relu = nn.ReLU()

    def forward(self,x):
        x = self.linear_embedding(x)
        x = rearrange(x, 'b c h w -> b (h w) c
        x = self.relu(self.layer_norm(x))
        return x
```


The most common way to do the ViT style patch partition + linear embedding step is to use a convolution where kernel size = stride = patch size and output channels = C. The resulting shape of this image is the shape we are looking for where the height and width are both divided by the patch

size and each of the h*w “tokens” are linear transformations of the original 4x4x3 image pixels.

We first define our SwinEmbedding class and inherit from nn.Module. We initialize our p x p stride p convolution and set output channels to C. We also initialize layer_norm to C for the embedding dimension size and initialize ReLU. In our forward we pass our input through our convolutional linear embedding then rearrange and permute our tensor by combing h,w into h*w tokens and move our embedding dimension to the right side. We conclude by adding our normalization and nonlinearity.

Patch Merging Layer

> *To produce a hierarchical representation, the number of tokens is reduced by patch merging layers as the network gets deeper. The first patch merging layer concatenates the features of each group of 2 × 2 neighboring patches, and applies a linear layer on the 4C-dimensional concatenated features. This reduces*

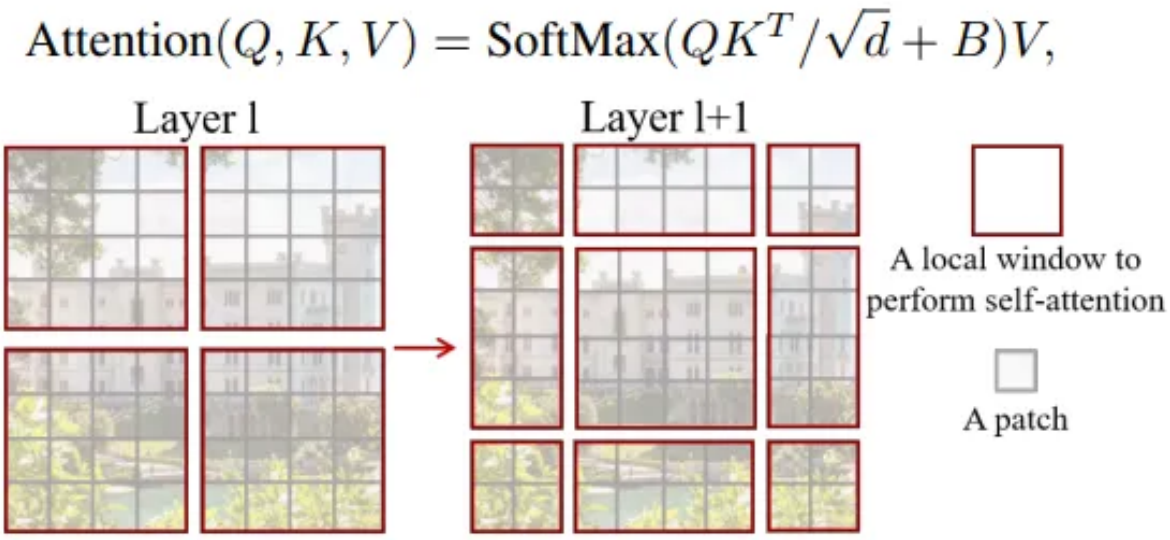
> *the number of tokens by a multiple of 2×2 = 4 (2× downsampling of resolution), and the output dimension is set to 2C.*


```
class PatchMerging(nn.Module):

    '''
    input shape -> (b, (h*w), C)
    output shape -> (b, (h/2 * w/2), C*2)
    '''

    def __init__(self, C):
        super().__init__()
        self.linear = nn.Linear(4*C, 2*C)
        self.layer_norm = nn.LayerNorm(2*C)

    def forward(self, x):
        height = width = int(math.sqrt(x.shape
        x = rearrange(x, 'b (h s1 w s2) c -> b
        return self.layer_norm(self.linear(x))
```

The patch merging layer is straightforward. We initialize a linear layer with 4C input channels to 2C output channels and initialize a layer norm

with the output embedding size. In our forward function we use einops rearrange to reshape our tokens from 2x2xC to 1x1x4C. We finish by passing our inputs through the linear projection and layer norm.

Window Attention Mechanism



Window Attention Example

In the Swin Transformer, attention is computed with the familiar attention formula shown in the image above but in parallel across non-overlapping windows. We will start by first coding the standard window based self attention mechanism and we will deal with the alternating shifted windows later.

```
class ShiftedWindowMSA(nn.Module):

'''
 input shape -> (b,(h*w), C)
 output shape -> (b, (h*w), C)
'''

    def __init__(self, embed_dim, num_heads, w
        super().__init__()
        self.embed_dim = embed_dim
        self.num_heads = num_heads
        self.window_size = window_size
        self.proj1 = nn.Linear(embed_dim, 3*em
        self.proj2 = nn.Linear(embed_dim, embe

    def forward(self, x):
        h_dim = self.embed_dim / self.num_head
        height = width = int(math.sqrt(x.shape
        x = self.proj1(x)

        x = rearrange(x, 'b (h w) (c K) -> b h
        x = rearrange(x, 'b (h m1) (w m2) (H h

      '''
        H = # of Attention Heads
        h,w = # of windows vertically and hori
        (m1 m2) = total size of each window
        E = head dimension
        K = 3 = a constant to break our matrix
      '''
```

```
        Q, K, V = x.chunk(3, dim=6)
        Q, K, V = Q.squeeze(-1), K.squeeze(-1)
        att_scores = (Q @ K.transpose(4,5)) /
        att = F.softmax(att_scores, dim=-1) @

        x = rearrange(att, 'b H h w (m1 m2)
        x = rearrange(x, 'b h w c -> b (h w

        return self.proj2(x)
```


We start by initializing our parameters embed_dim, num_heads, and and window_size and defining two linear projections. The first is our projection from inputs to Queries, Keys, and Values which we do in one parallel projection so the output size is set to 3*C. The second projection is a linear projection applied after the attention computation. This projection is for communication between the concatenated parallel multi-headed attention units.

We begin our forward function by getting the size of our head dim, height, and width of our input because we need these parameters for rearrange.

We then do our Q,K,V projection on our input of shape ((h*w), c) to ((h*w), 3C). Our next step is in two parts where we will rearrange our input ((h*w), C*3) into windows and parallel attention heads for our attention computation.

The rest of the function is simple. We break up our matrix into 3 Q,K,V matrices and compute attention by the standard attention formula found in the above image of this section. Because of how we shaped our matrices, the attention computation in the windows are done efficiently in parallel across the windows and attention heads. In the end we rearrange the tensors back to ((h*w),C) and return our final projected input.

Shifted Window Attention Mechanism

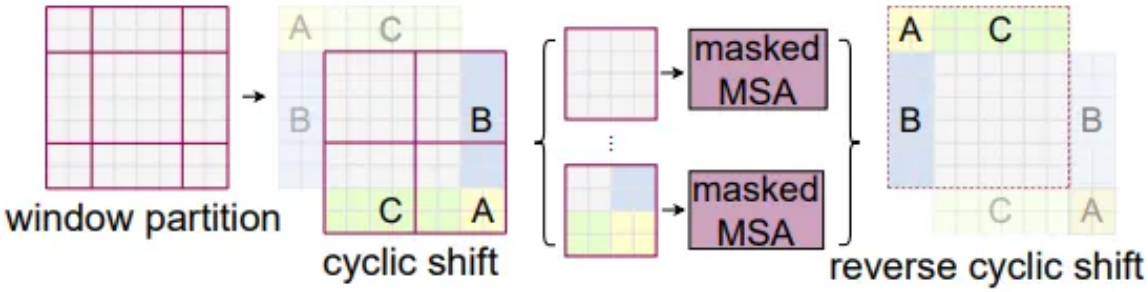


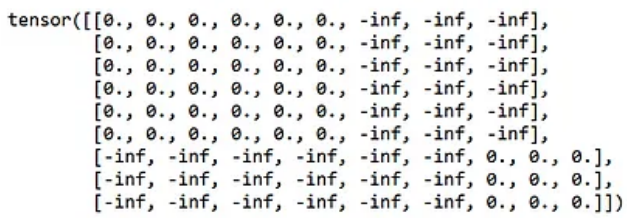
Figure 4. Illustration of an efficient batch computation approach for self-attention in shifted window partitioning.

The idea of shifted window attention is that on alternating attention computation layers we shift our windows so that our shifted windows overlap over the previous layers windows to allow cross window communication for the model. We can achieve this efficiently by a cyclic shift as depi in the image above. PyTorch has a function torch.roll we can use that will perform our cyclic shift of size window_size/2 on our input.

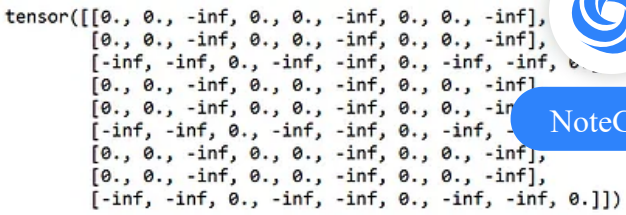
This is great! However one issue will come from using the cyclic shift to perform shifted window attention. Because we are now computing attention on the new shifted image, the model will be confused on where sections A,B,C in the above example actually belong in the image. We can solve this issue by masking the communication between tokens that should not be next to each other in the original image.

Because we will shift our image size by window_size/2, only half of the last row and column of windows will be affected. For this

reason, we can apply the left mask shown below to all of the last rows and the mask on the right to all of the last columns of the image.



```
tensor([[0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [0., 0., 0., 0., 0., 0., -inf, -inf, -inf],
        [-inf, -inf, -inf, -inf, -inf, -inf, 0., 0., 0.],
        [-inf, -inf, -inf, -inf, -inf, -inf, 0., 0., 0.],
        [-inf, -inf, -inf, -inf, -inf, -inf, 0., 0., 0.]])
```



```
tensor([[0., 0., -inf, 0., 0., -inf, 0., 0., -inf],
        [0., 0., -inf, 0., 0., -inf, 0., 0., -inf],
        [-inf, -inf, 0., -inf, -inf, 0., -inf, -inf,
        [0., 0., -inf, 0., 0., -inf, 0., 0., -inf],
        [0., 0., -inf, 0., 0., -inf, 0., 0., -in
        [-inf, -inf, 0., -inf, -inf, 0., -inf, -
        [0., 0., -inf, 0., 0., -inf, 0., 0., -inf],
        [0., 0., -inf, 0., 0., -inf, 0., 0., -inf],
        [-inf, -inf, 0., -inf, -inf, 0., -inf, -inf, 0.]])
```


For more insight on the specific details of the masks, this video is very useful. https://www.youtube.com/watch?v=s0yiRi_pr10&list=PL9iXGo3xD8jokWaLB8ZHUkjjv5Y_vPQnZ&index=5&t=114s

```
class ShiftedWindowMSA(nn.Module):
    def __init__(self, embed_dim, num_heads, w
        super().__init__()
        self.embed_dim = embed_dim
        self.num_heads = num_heads
        self.window_size = window_size
        self.mask = mask
        self.proj1 = nn.Linear(embed_dim, 3*em
        self.proj2 = nn.Linear(embed_dim, embe
        # self.embeddings = RelativeEmbeddings

    def forward(self, x):
```

```
        h_dim = self.embed_dim / self.num_heac
        height = width = int(math.sqrt(x.shape
        x = self.proj1(x)
        x = rearrange(x, 'b (h w) (c K) -> b h

        if self.mask:
            x = torch.roll(x, (-self.windov

        x = rearrange(x, 'b (h m1) (w m2) (H I
        Q, K, V = x.chunk(3, dim=6)
        Q, K, V = Q.squeeze(-1), K.squeeze(-1)
        att_scores = (Q @ K.transpose(4,5)) /
        # att_scores = self.embeddings(att_sc

        '''
          shape of att_scores = (b, H, h, w,
          we simply have to generate our row/c
          to the last row and columns of windc
        '''

        if self.mask:
            row_mask = torch.zeros((self.windc
            row_mask[-self.window_size * (self
            row_mask[0:-self.window_size * (se
            column_mask = rearrange(row_mask,
            att_scores[:, :, -1, :] += row_mas
            att_scores[:, :, :, -1] += column_

        att = F.softmax(att_scores, dim=-1) @
        x = rearrange(att, 'b H h w (m1 m2) E

        if self.mask:
            x = torch.roll(x, (self.window_siz
```

```
        x = rearrange(x, 'b h w c -> b (h w) c
        return self.proj2(x)
```

We have now edited the code to account for th shifted window part of the attention computation. We added a boolean mask to our init method to see whether or not we must mask our input. In our forward we added the cyclic shift and reverse cyclic shift to our height and width representing the # of windows and added our attention masks.

The commented parts of the code is for the relative position embeddings we will cover in the following section. Be sure if you are copying the code to uncomment these lines after understanding the relative position embedding section.

Relative Position Embeddings

> *In computing self-attention, we follow [49, 1, 32, 33] by including a relative position bias B ∈ R(M2×M2)*

> *to each head in computing similarity: Attention(Q, K, V) = SoftMax(QKT / √ d + B)V, where Q, K, V ∈ R(M2×d) are the query, key and value matrices; d is the query/key dimension, and M2 is the number of patches in a window. Since the relative position al each axis lies in the range [−M + 1, M −1], we parameterize a smaller-sized bias matrix Bˆ ∈ R (2M−1)×(2M−1), and values in B are taken from Bˆ.*

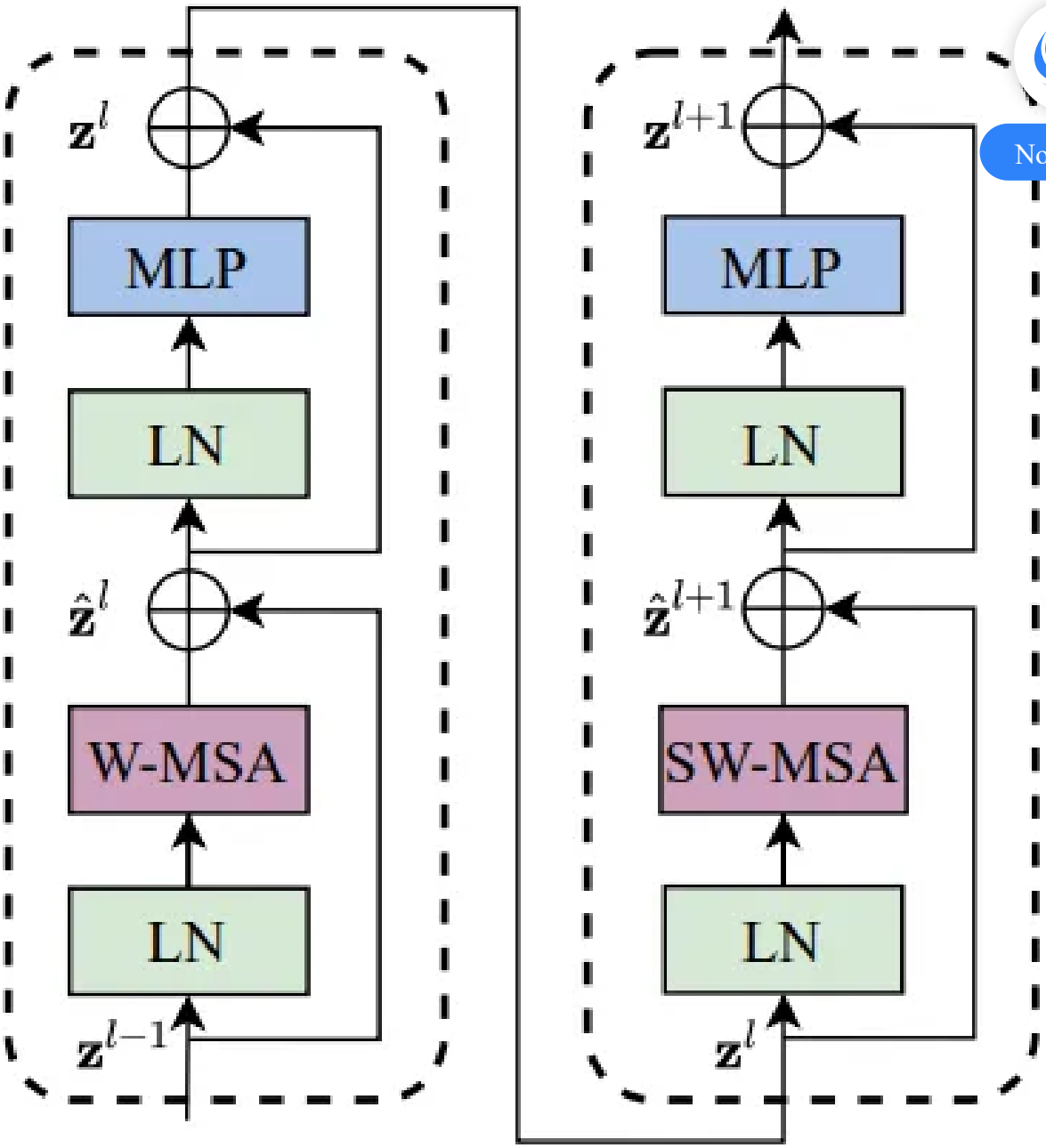

```
class RelativeEmbeddings(nn.Module):
    def __init__(self, window_size=7):
        super().__init__()
        B = nn.Parameter(torch.randn(2*window_
        x = torch.arange(1,window_size+1,1/wi
        x = (x[None, :]-x[:, None]).int()
        y = torch.concat([torch.arange(1,windo
        y = (y[None, :]-y[:, None])
        self.embeddings = nn.Parameter((B[x[:

    def forward(self, x):
        return x + self.embeddings
```

Our goal is to insert our position embeddings directly into our attention computation. These embeddings will be learned through training rather than using sinusoidal frequencies. Since we are adding the embeddings element-wise to our Query-Key product, they must be of shape (..., M*M, M*M). This is because our Query matrix is shape (..., M*M, E) and our Key matrix is shape (..., E, M*M). One important fact to notice however is that in each attention window of size M*M, a token's relative position can only be different by at most [-M+1, M-1] positions on each axis. This allows us to, as explained by the quote from the paper, only have to keep track of parameters of the smaller sized matrix and populate our larger matrix with these learned values.

Transformer Encoder Block

Now that we have created our shifted window attention, we can create the rest of the transformer block. The transformer block is straightforward as it is the same typical encoder block found in most Transformer architectures.

The only specific variation is that we use the gelu activation in our MLP.




```
class SwinEncoderBlock(nn.Module):
    def __init__(self, embed_dim, num_heads, w
        super().__init__()
        self.layer_norm = nn.LayerNorm(embed_d
```

```
            self.dropout = nn.Dropout(0.1)
            self.WMSA = ShiftedWindowMSA(embed_dim
            self.MLP1 = nn.Sequential(
                nn.Linear(embed_dim, embed_dim*4)
                nn.GELU(),
                nn.Linear(embed_dim*4, embed_dim
            )

        def forward(self, x):
            height, width = x.shape[1:3]
            res1 = self.dropout(self.WMSA(self.lay
            x = self.layer_norm(res1)
            x = self.MLP1(x)
            return self.dropout(x + res1)

    class AlternatingEncoderBlock(nn.Module):
        def __init__(self, embed_dim, num_heads, w
            super().__init__()
            self.WSA = SwinEncoderBlock(embed_dim=
            self.SWSA = SwinEncoderBlock(embed_dim
```


We also include a class for alternating encoder blocks. This is because all encoder blocks in the model are in pairs where the first block has

normal windows and the second block has windows shifted by window_size/2.

Final Swin-Transformer Class

Now that we have all the components of the Swin Transformer coded, we can make our final class. We will follow the same encoder block structure, size of embedding dimension, and number of attention heads as the original paper.


```
class SwinTransformer(nn.Module):
    def __init__(self):
        super().__init__()
        self.Embedding = SwinEmbedding()
        self.Embedding = SwinEmbedding()
        self.PatchMerge1 = PatchMerging(96)
        self.PatchMerge2 = PatchMerging(192)
        self.PatchMerge3 = PatchMerging(384)
        self.Stage1 = AlternatingEncoderBlock
        self.Stage2 = AlternatingEncoderBlock
        self.Stage3_1 = AlternatingEncoderBloc
        self.Stage3_2 = AlternatingEncoderBloc
        self.Stage3_3 = AlternatingEncoderBloc
        self.Stage4 = AlternatingEncoderBlock

    def forward(self, x):
        x = self.Embedding(x)
        x = self.PatchMerge1(self.Stage1(x))
```

```
        x = self.PatchMerge2(self.Stage2(x))
        x = self.Stage3_1(x)
        x = self.Stage3_2(x)
        x = self.Stage3_3(x)
        x = self.PatchMerge3(x)
        x = self.Stage4(x)
        return x
```


Finally, we will create a dummy tensor to pass through the model to make sure there are no errors and we get the shape we expect at the end. We will use a single batched image of size (1,3,224,224). Because we will pass the image through the initial embedding layer and 3 patch merging layers, we expect our final shape (1,49,768) where 1 is the batch dimension, 49 is the 7*7 height and width reshaped into one dimension, and 768 is the final number of channels/ embedding dimension size.

```
def main():
    x = torch.randn((1,3,224,224)).cuda()
    model = SwinTransformer().cuda()
```

```
        print(model(x).shape)

    if __name__ == '__main__':
        main()
```

And its a success! We have now fully implemented the Swin Transformer from scratch using PyTorch! I hope now you have a great understanding of the architecture and you enjoyed this article.

In Plain English

Thank you for being a part of our community! Before you go:

- *Be sure to **clap** and **follow** the writer!* 👏
- *You can find even more content at **PlainEnglish.io*** 🚀
- *Sign up for our **free weekly newsletter**.* 🗞️
- *Follow us on **Twitter(X)**, **LinkedIn**, **YouTube**, and **Discord**.*

Data Science Transformers

Machine Learning Swin Transformer

Deep Learning

Written by Nickd

8 Followers · Writer for Python in Plain English

Student studying Computer Vision and Mathematics

Swin-Unet: Unet-like Pure Transformer for Medical Image Segmentation

Hu Cao[1†], Yueyue Wang[2†], Joy Chen[1], Dongsheng Jiang[3*], Xiaopeng Zhang[3*], Qi Tian[3*], and Manning Wang[2]

[1] Technische Universität München, München, Germany
[2] Fudan University, Shanghai, China
[3] Huawei Technologies, Shanghai, China

Abstract. In the past few years, convolutional neural networks (CNNs) have achieved milestones in medical image analysis. Especially, the deep neural networks based on U-shaped architecture and skip-connections have been widely applied in a variety of medical image tasks. However, although CNN has achieved excellent performance, it cannot learn global and long-range semantic information interaction well due to the locality of convolution operation. In this paper, we propose Swin-Unet, which is a Unet-like pure Transformer for medical image segmentation. The tokenized image patches are fed into the Transformer-based U-shaped Encoder-Decoder architecture with skip-connections for local-global semantic feature learning. Specifically, we use hierarchical Swin Transformer with shifted windows as the encoder to extract context features. And a symmetric Swin Transformer-based decoder with patch expanding layer is designed to perform the up-sampling operation to restore the spatial resolution of the feature maps. Under the direct down-sampling and up-sampling of the inputs and outputs by 4×, experiments on multi-organ and cardiac segmentation tasks demonstrate that the pure Transformer-based U-shaped Encoder-Decoder network outperforms those methods with full-convolution or the combination of transformer and convolution. The codes and trained models will be publicly available at https://github.com/HuCaoFighting/Swin-Unet.

1 Introduction

Benefiting from the development of deep learning, computer vision technology has been widely used in medical image analysis. Image segmentation is an important part of medical image analysis. In particular, accurate and robust medical image segmentation can play a cornerstone role in computer-aided diagnosis and image-guided clinical surgery [1,2].

*Corresponding author
† Work done as an intern in Huawei Technologies

Existing medical image segmentation methods mainly rely on fully convolutional neural network (FCNN) with U-shaped structure [3,4,5]. The typical U-shaped network, U-Net [3], consists of a symmetric Encoder-Decoder with skip connections. In the encoder, a series of convolutional layers and continuous down-sampling layers are used to extract deep features with large receptive fields. Then, the decoder up-samples the extracted deep features to the input resolution for pixel-level semantic prediction, and the high-resolution features of different scale from the encoder are fused with skip connections to alleviate the loss of spatial information caused by down-sampling. With such an elegant structural design, U-Net has achieved great success in a variety of medical imaging applications. Following this technical route, many algorithms such as 3D U-Net [6], Res-UNet [7], U-Net++ [8] and UNet3+ [9] have been developed for image and volumetric segmentation of various medical imaging modalities. The excellent performance of these FCNN-based methods in cardiac segmentation, organ segmentation and lesion segmentation proves that CNN has a strong ability of learning discriminating features.

Currently, although the CNN-based methods have achieved excellent performance in the field of medical image segmentation, they still cannot fully meet the strict requirements of medical applications for segmentation accuracy. Image segmentation is still a challenge task in medical image analysis. Since the intrinsic locality of convolution operation, it is difficult for CNN-based approaches to learn explicit global and long-range semantic information interaction [2]. Some studies have tried to address this problem by using atrous convolutional layers [10,11], self-attention mechanisms [12,13], and image pyramids [14]. However, these methods still have limitations in modeling long - range dependencies. Recently, inspired by Transformer's great success in the nature language processing (NLP) domain [15], researchers have tried to bring Transformer into the vision domain [16]. In [17], vision transformer (ViT) is proposed to perform the image recognition task. Taking 2D image patches with positional embeddings as inputs and pre-training on large dataset, ViT achieved comparable performance with the CNN-based methods. Besides, data-efficient image transformer (DeiT) is presented in [18], which indicates that Transformer can be trained on mid-size datasets and that a more robust Transformer can be obtained by combining it with the distillation method. In [19], a hierarchical Swin Transformer is developed. Take Swin Transformer as vision backbone, the authors of [19] achieved state-of-the-art performance on Image classification, object detection and semantic segmentation. The success of ViT, DeiT and Swin Transformer in image recognition task demonstrates the potential for Transformer to be applied in the vision domain.

Motivated by the Swin Transformer's [19] success, we propose Swin-Unet to leverage the power of Transformer for 2D medical image segmentation in this work. To our best knowledge, Swin-Unet is a first pure Transformer-based U-shaped architecture that consists of encoder, bottleneck, decoder, and skip connections. Encoder, bottleneck and decoder are all built based on Swin Transformer block [19]. The input medical images are split into non-overlapping image

patches. Each patch is treated as a token and fed into the Transformer-based encoder to learn deep feature representations. The extracted context features are then up-sampled by the decoder with patch expanding layer, and fused with the multi-scale features from the encoder via skip connections, so as to restore the spatial resolution of the feature maps and further perform segmentation prediction. Extensive experiments on multi-organ and cardiac segmentation datasets indicate that the proposed method has excellent segmentation accuracy and robust generalization ability. Concretely, our contributions can be summarized as: (1) Based on Swin Transformer block, we build a symmetric Encoder-Decoder architecture with skip connections. In the encoder, self-attention from local to global is realized; in the decoder, the global features are up-sampled to the input resolution for corresponding pixel-level segmentation prediction. (2) A patch expanding layer is developed to achieve up-sampling and feature dimension increase without using convolution or interpolation operation. (3) It is found in the experiment that skip connection is also effective for Transformer, so a pure Transformer-based U-shaped Encoder-Decoder architecture with skip connection is finally constructed, named Swin-Unet.

2 Related work

CNN-based methods : Early medical image segmentation methods are mainly contour-based and traditional machine learning-based algorithms [20,21]. With the development of deep CNN, U-Net is proposed in [3] for medical image segmentation. Due to the simplicity and superior performance of the U-shaped structure, various Unet-like methods are constantly emerging, such as Res-UNet [7], Dense-UNet [22], U-Net++ [8] and UNet3+ [9]. And it is also introduced into the field of 3D medical image segmentation, such as 3D-Unet [6] and V-Net [23]. At present, CNN-based methods have achieved tremendous success in the field of medical image segmentation due to its powerful representation ability.

Vision transformers : Transformer was first proposed for the machine translation task in [15]. In the NLP domain, the Transformer-based methods have achieved the state-of-the-art performance in various tasks [24]. Driven by Transformer's success, the researchers introduced a pioneering vision transformer (ViT) in [17], which achieved the impressive speed-accuracy trade-off on image recognition task. Compared with CNN-based methods, the drawback of ViT is that it requires pre-training on its own large dataset. To alleviate the difficulty in training ViT, Deit [18] describes several training strategies that allow ViT to train well on ImageNet. Recently, several excellent works have been done baed on ViT [25,26,19]. It is worth mentioning that an efficient and effective hierarchical vision Transformer, called Swin Transformer, is proposed as a vision backbone in [19]. Based on the shifted windows mechanism, Swin Transformer achieved the state-of-the-art performance on various vision tasks including image classification, object detection and semantic segmentation. In this work, we attempt to use Swin Transformer block as basic unit to build a U-shaped Encoder-Decoder

architecture with skip connections for medical image segmentation, thus providing a benchmark comparison for the development of Transformer in the medical image field.

Self-attention/Transformer to complement CNNs : In recent years, researchers have tried to introduce self-attention mechanism into CNN to improve the performance of the network [13]. In [12], the skip connections with additive attention gate are integrated in U-shaped architecture to perform medical image segmentation. However, this is still the CNN-based method. Currently, some efforts are being made to combine CNN and Transformer to break the dominance of CNNs in medical image segmentation [2,27,1]. In [2], the authors combined Transformer with CNN to constitute a strong encoder for 2D medical image segmentation. Similar to [2], [27] and [28] use the complementarity of Transformer and CNN to improve the segmentation capability of the model. Currently, various combinations of Transformer with CNN are applied in multi-modal brain tumor segmentation [29] and 3D medical image segmentation [1,30]. Different from the above methods, we try to explore the application potential of pure Transformer in medical image segmentation.

3 Method

3.1 Architecture overview

The overall architecture of the proposed Swin-Unet is presented in Figure. 1. Swin-Unet consists of encoder, bottleneck, decoder and skip connections. The basic unit of Swin-Unet is Swin Transformer block [19]. For the encoder, to transform the inputs into sequence embeddings, the medical images are split into non-overlapping patches with patch size of 4×4. By such partition approach, the feature dimension of each patch becomes to $4 \times 4 \times 3 = 48$. Furthermore, a linear embedding layer is applied to projected feature dimension into arbitrary dimension (represented as C). The transformed patch tokens pass through several Swin Transformer blocks and patch merging layers to generate the hierarchical feature representations. Specifically, patch merging layer is responsible for down-sampling and increasing dimension, and Swin Transformer block is responsible for feature representation learning. Inspired by U-Net [3], we design a symmetric transformer-based decoder. The decoder is composed of Swin Transformer block and patch expanding layer. The extracted context features are fused with multiscale features from encoder via skip connections to complement the loss of spatial information caused by down-sampling. In contrast to patch merging layer, a patch expanding layer is specially designed to perform up-sampling. The patch expanding layer reshapes feature maps of adjacent dimensions into a large feature maps with $2\times$ up-sampling of resolution. In the end, the last patch expanding layer is used to perform $4\times$ up-sampling to restore the resolution of the feature maps to the input resolution ($W \times H$), and then a linear projection layer is applied on these up-sampled features to output the pixel-level segmentation predictions. We would elaborate each block in the following

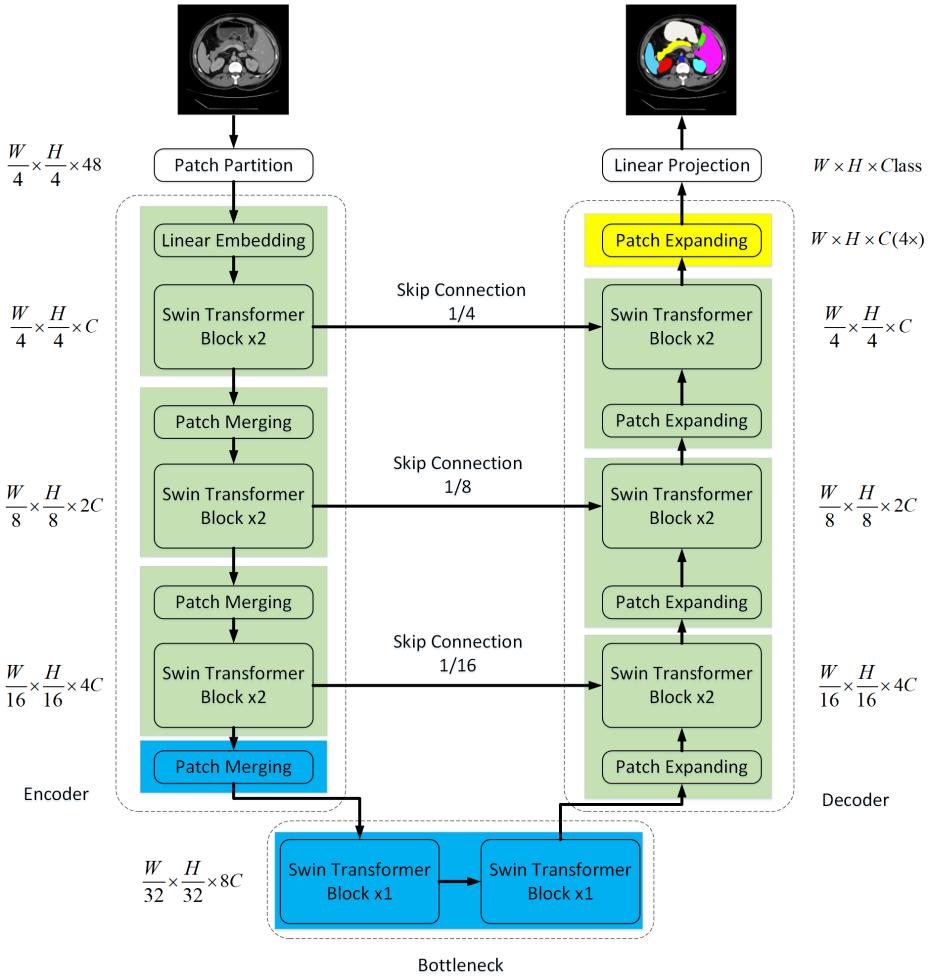


Fig. 1. The architecture of Swin-Unet, which is composed of encoder, bottleneck, decoder and skip connections. Encoder, bottleneck and decoder are all constructed based on swin transformer block.

3.2 Swin Transformer block

Different from the conventional multi-head self attention (MSA) module, swin transformer block [19] is constructed based on shifted windows. In Figure. 2, two consecutive swin transformer blocks are presented. Each swin transformer block is composed of LayerNorm (LN) layer, multi-head self attention module, residual connection and 2-layer MLP with GELU non-linearity. The window-based multi-head self attention (W-MSA) module and the shifted window-based multi-head self attention (SW-MSA) module are applied in the two successive

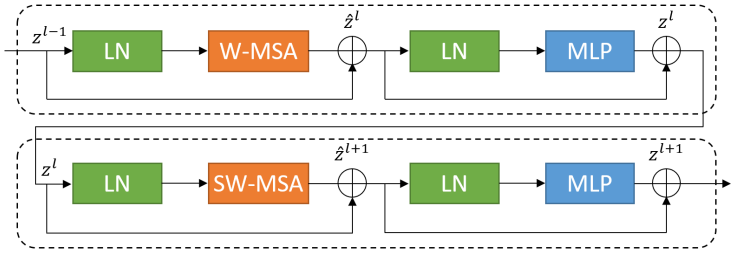


Fig. 2. Swin transformer block.

transformer blocks, respectively. Based on such window partitioning mechanism, continuous swin transformer blocks can be formulated as:

$$\hat{z}^l = W\text{-}MSA(LN(z^{l-1})) + z^{l-1}, \tag{1}$$

$$z^l = MLP(LN(\hat{z}^l)) + \hat{z}^l, \tag{2}$$

$$\hat{z}^{l+1} = SW\text{-}MSA(LN(z^l)) + z^l, \tag{3}$$

$$z^{l+1} = MLP(LN(\hat{z}^{l+1})) + \hat{z}^{l+1}, \tag{4}$$

where $\hat{z}^l$ and z^l represent the outputs of the (S)W-MSA module and the MLP module of the l^{th} block, respectively. Similar to the previous works [31,32], self-attention is computed as follows:

$$Attention(Q, K, V) = SoftMax(\frac{QK^T}{\sqrt{d}} + B)V, \tag{5}$$

where $Q, K, V \in \mathbb{R}^{M^2 \times d}$ denote the query, key and value matrices. M^2 and d represent the number of patches in a window and the dimension of the *query* or *key*, respectively. And, the values in B are taken from the bias matrix $\hat{B} \in \mathbb{R}^{(2M-1)\times(2M+1)}$.

3.3 Encoder

In the encoder, the C-dimensional tokenized inputs with the resolution of $\frac{H}{4} \times \frac{W}{4}$ are fed into the two consecutive Swin Transformer blocks to perform representation learning, in which the feature dimension and resolution remain unchanged. Meanwhile, the patch merging layer will reduce the number of tokens (2× downsampling) and increase the feature dimension to 2× the original dimension. This procedure will be repeated three times in the encoder.

Patch merging layer : The input patches are divided into 4 parts and concatenated together by the patch merging layer. With such processing, the feature resolution will be down-sampled by 2×. And, since the concatenate operation results the feature dimension increasing by 4×, a linear layer is applied on the concatenated features to unify the feature dimension to the 2× the original dimension.

3.4 Bottleneck

Since Transformer is too deep to be converged [33], only two successive Swin Transformer blocks are used to constructed the bottleneck to learn the deep feature representation. In the bottleneck, the feature dimension and resolution are kept unchanged.

3.5 Decoder

Corresponding to the encoder, the symmetric decoder is built based on Swin Transformer block. To this end, in contrast to the patch merging layer used in the encoder, we use the patch expanding layer in the decoder to up-sample the extracted deep features. The patch expanding layer reshapes the feature maps of adjacent dimensions into a higher resolution feature map (2× up-sampling) and reduces the feature dimension to half of the original dimension accordingly.

Patch expanding layer : Take the first patch expanding layer as an example, before up-sampling, a linear layer is applied on the input features ($\frac{W}{32} \times \frac{H}{32} \times 8C$) to increase the feature dimension to 2× the original dimension ($\frac{W}{32} \times \frac{H}{32} \times 16C$). Then, we use rearrange operation to expand the resolution of the input features to 2× the input resolution and reduce the feature dimension to quarter of the input dimension ($\frac{W}{32} \times \frac{H}{32} \times 16C \rightarrow \frac{W}{16} \times \frac{H}{16} \times 4C$). We will discuss the impact of using patch expanding layer to perform up-sampling in section 4.5.

3.6 Skip connection

Similar to the U-Net [3], the skip connections are used to fuse the multi-scale features from the encoder with the up-sampled features. We concatenate the shallow features and the deep features together to reduce the loss of spatial information caused by down-sampling. Followed by a linear layer, the dimension of the concatenated features is remained the same as the dimension of the up-sampled features. In section 4.5, we will detailed discuss the impact of the number of skip connections on the performance of our model.

4 Experiments

4.1 Datasets

Synapse multi-organ segmentation dataset (Synapse): the dataset includes 30 cases with 3779 axial abdominal clinical CT images. Following [2,34],

Table 1. Segmentation accuracy of different methods on the Synapse multi-organ CT dataset.

Methods	DSC↑	HD↓	Aorta	Gallbladder	Kidney(L)	Kidney(R)	Liver	Pancreas	Spleen	Stomach
V-Net [35]	68.81	-	75.34	51.87	77.10	**80.75**	87.84	40.05	80.56	56.98
DARR [36]	69.77	-	74.74	53.77	72.31	73.24	94.08	54.18	89.90	45.96
R50 U-Net [2]	74.68	36.87	87.74	63.66	80.60	78.19	93.74	56.90	85.87	74.16
U-Net [3]	76.85	39.70	89.07	**69.72**	77.77	68.60	93.43	53.98	86.67	75.58
R50 Att-UNet [2]	75.57	36.97	55.92	63.91	79.20	72.71	93.56	49.37	87.19	74.95
Att-UNet [37]	77.77	36.02	**89.55**	68.88	77.98	71.11	93.57	**58.04**	87.30	75.75
R50 ViT [2]	71.29	32.87	73.73	55.13	75.80	72.20	91.51	45.99	81.99	73.95
TransUnet [2]	77.48	31.69	87.23	63.13	81.87	77.02	94.08	55.86	85.08	75.62
SwinUnet	**79.13**	**21.55**	85.47	66.53	**83.28**	79.61	**94.29**	56.58	**90.66**	**76.60**

18 samples are divided into the training set and 12 samples into testing set. And the average Dice-Similarity coefficient (DSC) and average Hausdorff Distance (HD) are used as evaluation metric to evaluate our method on 8 abdominal organs (aorta, gallbladder, spleen, left kidney, right kidney, liver, pancreas, spleen, stomach).

Automated cardiac diagnosis challenge dataset (ACDC): the ACDC dataset is collected from different patients using MRI scanners. For each patient MR image, left ventricle (LV), right ventricle (RV) and myocardium (MYO) are labeled. The dataset is split into 70 training samples, 10 validation samples and 20 testing samples. Similar to [2], only average DSC is used to evaluate our method on this dataset.

4.2 Implementation details

The Swin-Unet is achieved based on Python 3.6 and Pytorch 1.7.0. For all training cases, data augmentations such as flips and rotations are used to increase data diversity. The input image size and patch size are set as 224×224 and 4, respectively. We train our model on a Nvidia V100 GPU with 32GB memory. The weights pre-trained on ImageNet are used to initialize the model parameters. During the training period, the batch size is 24 and the popular SGD optimizer with momentum 0.9 and weight decay 1e-4 is used to optimize our model for back propagation.

4.3 Experiment results on Synapse dataset

The comparison of the proposed Swin-Unet with previous state-of-the-art methods on the Synapse multi-organ CT dataset is presented in Table. 1. Different from TransUnet [2], we add the test results of our own implementations of U-Net [3]and Att-UNet [37] on the Synapse dataset. Experimental results demonstrate that our Unet-like pure transformer method achieves the best performance with segmentation accuracy of 79.13%(DSC↑) and 21.55%(HD↓). Compared with Att-Unet [37] and the recently method TransUnet [2], although our algorithm did not improve much on the DSC evaluation metric, we achieved accuracy improvement of about 4% and 10% on the HD evaluation metric, which

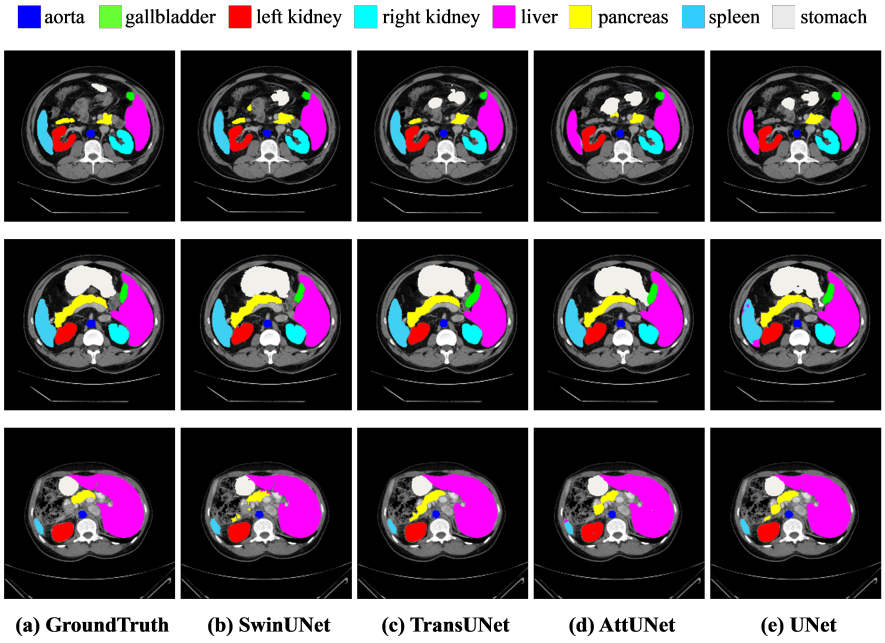


Fig. 3. The segmentation results of different methods on the Synapse multi-organ CT dataset.

Table 2. Segmentation accuracy of different methods on the ACDC dataset.

Methods	DSC	RV	Myo	LV
R50 U-Net	87.55	87.10	80.63	94.92
R50 Att-UNet	86.75	87.58	79.20	93.47
R50 ViT	87.57	86.07	81.88	94.75
TransUnet	89.71	88.86	84.53	95.73
SwinUnet	**90.00**	88.55	**85.62**	**95.83**

indicates that our approach can achieve better edge predictions. The segmentation results of different methods on the Synapse multi-organ CT dataset are shown in Figure. 3. It can be seen from the figure that CNN-based methods tend to have over-segmentation problems, which may be caused by the locality of convolution operation. In this work, we demonstrate that by integrating Transformer with a U-shaped architecture with skip connections, the pure Transformer approach without convolution can better learn both global and long-range semantic information interactions, resulting in better segmentation results.

Table 3. Ablation study on the impact of the up-sampling

Up-sampling	DSC	Aorta	Gallbladder	Kidney(L)	Kidney(R)	Liver	Pancreas	Spleen	Stomach
Bilinear interpolation	76.15	81.84	66.33	80.12	73.91	93.64	55.04	86.10	72.20
Transposed convolution	77.63	84.81	65.96	82.66	74.61	**94.39**	54.81	89.42	74.41
Patch expand	**79.13**	**85.47**	**66.53**	**83.28**	**79.61**	94.29	**56.58**	**90.66**	**76.60**

Table 4. Ablation study on the impact of the number of skip connection

Skip connection	DSC	Aorta	Gallbladder	Kidney(L)	Kidney(R)	Liver	Pancreas	Spleen	Stomach
0	72.46	78.71	53.24	77.46	75.90	92.60	46.07	84.57	71.13
1	76.43	82.53	60.44	81.36	79.27	93.64	53.36	85.95	74.90
2	78.93	**85.82**	66.27	**84.70**	**80.32**	93.94	55.32	88.35	**76.71**
3	**79.13**	85.47	**66.53**	83.28	79.61	**94.29**	**56.58**	**90.66**	76.60

4.4 Experiment results on ACDC dataset

Similar to the Synapse dataset, the proposed Swin-Unet is trained on ACDC dataset to perform medical image segmentation. The experimental results are summarized in Table. 2. By using the image data of MR mode as input, Swin-Unet is still able to achieve excellent performance with an accuracy of 90.00%, which shows that our method has good generalization ability and robustness.

4.5 Ablation study

In order to explore the influence of different factors on the model performance, we conducted ablation studies on Synapse dataset. Specifically, up-sampling, the number of skip connections, input sizes, and model scales are discussed below.

Effect of up-sampling: Corresponding to the patch merging layer in the encoder, we specially designed a patch expanding layer in the decoder to perform up-sampling and feature dimension increase. To explore the effective of the proposed patch expanding layer, we conducted the experiments of Swin-Unet with bilinear interpolation, transposed convolution and patch expanding layer on Synapse dataset. The experimental results in the Table 3 indicate that the proposed Swin-Unet combined with the patch expanding layer can obtain the better segmentation accuracy.

Effect of the number of skip connections: The skip connections of our Swin-UNet are added in places of the 1/4, 1/8, and 1/16 resolution scales. By changing the number of skip connections to 0, 1, 2 and 3 respectively, we explored the influence of different skip connections on the segmentation performance of the proposed model. In Table 4, we can see that the segmentation performance of the model increases with the increase of the number of skip connections. Therefore, in order to make the model more robust, the number of skip connections is set as 3 in this work.

Table 5. Ablation study on the impact of the input size

Input size	DSC	Aorta	Gallbladder	Kidney(L)	Kidney(R)	Liver	Pancreas	Spleen	Stomach
224	79.13	85.47	66.53	83.28	79.61	94.29	56.58	**90.66**	**76.60**
384	**81.12**	**87.07**	**70.53**	**84.64**	**82.87**	**94.72**	**63.73**	90.14	75.29

Table 6. Ablation study on the impact of the model scale

Model scale	DSC	Aorta	Gallbladder	Kidney(L)	Kidney(R)	Liver	Pancreas	Spleen	Stomach
tiny	79.13	85.47	66.53	83.28	79.61	**94.29**	56.58	**90.66**	**76.60**
base	**79.25**	**87.16**	**69.19**	**84.61**	**81.99**	93.86	**58.10**	88.44	70.65

Effect of input size: The testing results of the proposed Swin-Unet with 224×224, 384×384 input resolutions as input are presented in Table. 5. As the input size increases from 224×224 to 384×384 and the patch size remains the same as 4, the input token sequence of Transformer will become larger, thus leading to improve the segmentation performance of the model. However, although the segmentation accuracy of the model has been slightly improved, the computational load of the whole network has also increased significantly. In order to ensure the running efficiency of the algorithm, the experiments in this paper are based on 224×224 resolution scale as the input.

Effect of model scale: Similar to [19], we discuss the effect of network deepening on model performance. It can be seen from Table. 6 that the increase of model scale hardly improves the performance of the model, but increases the computational cost of the whole network. Considering the accuracy-speed trade off, we adopt the Tiny-based model to perform medical image segmentation.

4.6 Discussion

As we all known, the performance of Transformer-based model is severely affected by model pre-training. In this work, we directly use the training weight of Swin transformer [19] on ImageNet to initialize the network encoder and decoder, which may be a suboptimal scheme. This initialization approach is a simple one, and in the future we will explore the ways to pre-train Transformer end-to-end for medical image segmentation. Moreover, since the input images in this paper are 2D, while most of the medical image data are 3D, we will explore the application of Swin-Unet in 3D medical image segmentation in the following research.

5 Conclusion

In this paper, we introduced a novel pure transformer-based U-shaped encoder-decoder for medical image segmentation. In order to leverage the power of Transformer, we take Swin Transformer block as the basic unit for feature representation and long-range semantic information interactive learning. Extensive ex-

periments on multi-organ and cardiac segmentation tasks demonstrate that the proposed Swin-Unet has excellent performance and generalization ability.

References

1. A. Hatamizadeh, D. Yang, H. Roth, and D. Xu, "Unetr: Transformers for 3d medical image segmentation," 2021.
2. J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "Transunet: Transformers make strong encoders for medical image segmentation," *CoRR*, vol. abs/2102.04306, 2021.
3. O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9351. Springer, 2015, pp. 234–241.
4. K. S. P. J. M.-H. K. Isensee F, Jaeger PF, "nnu-net: a self-configuring method for deep learning-based biomedical image segmentation," *Nat Methods*, vol. 18(2):203-211, 2021.
5. Q. Jin, Z. Meng, C. Sun, H. Cui, and R. Su, "Ra-unet: A hybrid deep attention-aware network to extract liver and tumor in ct scans," *Frontiers in Bioengineering and Biotechnology*, vol. 8, p. 1471, 2020.
6. Ö. Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: Learning dense volumetric segmentation from sparse annotation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, vol. 9901. Springer, Oct 2016, pp. 424–432.
7. X. Xiao, S. Lian, Z. Luo, and S. Li, "Weighted res-unet for high-quality retina vessel segmentation," *2018 9th International Conference on Information Technology in Medicine and Education (ITME)*, pp. 327–331, 2018.
8. Z. Zhou, M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation." Springer Verlag, 2018, pp. 3–11.
9. H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "Unet 3+: A full-scale connected unet for medical image segmentation," 2020.
10. L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2018.
11. Z. Gu, J. Cheng, H. Fu, K. Zhou, H. Hao, Y. Zhao, T. Zhang, S. Gao, and J. Liu, "Ce-net: Context encoder network for 2d medical image segmentation," *IEEE Transactions on Medical Imaging*, vol. 38, no. 10, pp. 2281–2292, 2019.
12. J. Schlemper, O. Oktay, M. Schaap, M. Heinrich, B. Kainz, B. Glocker, and D. Rueckert, "Attention gated networks: Learning to leverage salient regions in medical images," *Medical Image Analysis*, vol. 53, pp. 197–207, 2019.
13. X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.
14. H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6230–6239.

15. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.
16. N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *CoRR*, vol. abs/2005.12872, 2020.
17. 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," in *International Conference on Learning Representations*, 2021.
18. H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," *CoRR*, vol. abs/2012.12877, 2020.
19. Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," *CoRR*, vol. abs/2103.14030, 2021.
20. A. Tsai, A. Yezzi, W. Wells, C. Tempany, D. Tucker, A. Fan, W. Grimson, and A. Willsky, "A shape-based approach to the segmentation of medical imagery using level sets," *IEEE Transactions on Medical Imaging*, vol. 22, no. 2, pp. 137–154, 2003.
21. K. Held, E. Kops, B. Krause, W. Wells, R. Kikinis, and H.-W. Muller-Gartner, "Markov random field segmentation of brain mr images," *IEEE Transactions on Medical Imaging*, vol. 16, no. 6, pp. 878–886, 1997.
22. X. Li, H. Chen, X. Qi, Q. Dou, C.-W. Fu, and P.-A. Heng, "H-denseunet: Hybrid densely connected unet for liver and tumor segmentation from ct volumes," *IEEE Transactions on Medical Imaging*, vol. 37, no. 12, pp. 2663–2674, 2018.
23. F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016.
24. 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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423
25. W. Wang, E. Xie, X. Li, D. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," *CoRR*, vol. abs/2102.12122, 2021. [Online]. Available: https://arxiv.org/abs/2102.12122
26. K. Han, A. Xiao, E. Wu, J. Guo, C. Xu, and Y. Wang, "Transformer in transformer," *CoRR*, vol. abs/2103.00112, 2021. [Online]. Available: https://arxiv.org/abs/2103.00112
27. J. M. J. Valanarasu, P. Oza, I. Hacihaliloglu, and V. M. Patel, "Medical transformer: Gated axial-attention for medical image segmentation," *CoRR*, vol. abs/2102.10662, 2021.
28. Y. Zhang, H. Liu, and Q. Hu, "Transfuse: Fusing transformers and cnns for medical image segmentation," *CoRR*, vol. abs/2102.08005, 2021. [Online]. Available: https://arxiv.org/abs/2102.08005
29. W. Wang, C. Chen, M. Ding, J. Li, H. Yu, and S. Zha, "Transbts: Multimodal brain tumor segmentation using transformer," *CoRR*, vol. abs/2103.04430, 2021. [Online]. Available: https://arxiv.org/abs/2103.04430

30. Y. Xie, J. Zhang, C. Shen, and Y. Xia, "Cotr: Efficiently bridging CNN and transformer for 3d medical image segmentation," *CoRR*, vol. abs/2103.03024, 2021. [Online]. Available: https://arxiv.org/abs/2103.03024
31. H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3588–3597.
32. H. Hu, Z. Zhang, Z. Xie, and S. Lin, "Local relation networks for image recognition," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 3463–3472.
33. H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. Jégou, "Going deeper with image transformers," *CoRR*, vol. abs/2103.17239, 2021. [Online]. Available: https://arxiv.org/abs/2103.17239
34. S. Fu, Y. Lu, Y. Wang, Y. Zhou, W. Shen, E. Fishman, and A. Yuille, "Domain adaptive relational reasoning for 3d multi-organ segmentation," in *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, 2020, pp. 656–666.
35. F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.
36. S. Fu, Y. Lu, Y. Wang, Y. Zhou, W. Shen, E. Fishman, and A. Yuille, "Domain adaptive relational reasoning for 3d multi-organ segmentation," Germany, 2020, pp. 656–666.
37. O. Oktay, J. Schlemper, L. L. Folgoc, M. Lee, M. Heinrich, K. Misawa, K. Mori, S. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *IMIDL Conference*, 2018.

U-Net: Convolutional Networks for Biomedical Image Segmentation

Olaf Ronneberger, Philipp Fischer, and Thomas Brox

Computer Science Department and BIOSS Centre for Biological Signalling Studies,
University of Freiburg, Germany
ronneber@informatik.uni-freiburg.de,
WWW home page: http://lmb.informatik.uni-freiburg.de/

Abstract. There is large consent that successful training of deep networks requires many thousand annotated training samples. In this paper, we present a network and training strategy that relies on the strong use of data augmentation to use the available annotated samples more efficiently. The architecture consists of a contracting path to capture context and a symmetric expanding path that enables precise localization. We show that such a network can be trained end-to-end from very few images and outperforms the prior best method (a sliding-window convolutional network) on the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks. Using the same network trained on transmitted light microscopy images (phase contrast and DIC) we won the ISBI cell tracking challenge 2015 in these categories by a large margin. Moreover, the network is fast. Segmentation of a 512x512 image takes less than a second on a recent GPU. The full implementation (based on Caffe) and the trained networks are available at http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net.

1 Introduction

In the last two years, deep convolutional networks have outperformed the state of the art in many visual recognition tasks, e.g. [7,3]. While convolutional networks have already existed for a long time [8], their success was limited due to the size of the available training sets and the size of the considered networks. The breakthrough by Krizhevsky et al. [7] was due to supervised training of a large network with 8 layers and millions of parameters on the ImageNet dataset with 1 million training images. Since then, even larger and deeper networks have been trained [12].

The typical use of convolutional networks is on classification tasks, where the output to an image is a single class label. However, in many visual tasks, especially in biomedical image processing, the desired output should include localization, i.e., a class label is supposed to be assigned to each pixel. Moreover, thousands of training images are usually beyond reach in biomedical tasks. Hence, Ciresan et al. [1] trained a network in a sliding-window setup to predict the class label of each pixel by providing a local region (patch) around that pixel

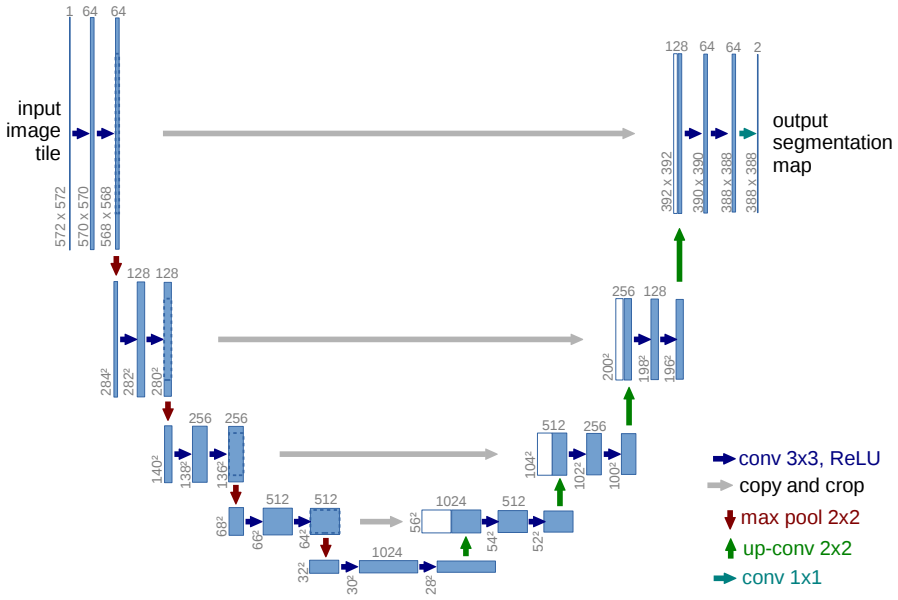


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

as input. First, this network can localize. Secondly, the training data in terms of patches is much larger than the number of training images. The resulting network won the EM segmentation challenge at ISBI 2012 by a large margin.

Obviously, the strategy in Ciresan et al. [1] has two drawbacks. First, it is quite slow because the network must be run separately for each patch, and there is a lot of redundancy due to overlapping patches. Secondly, there is a trade-off between localization accuracy and the use of context. Larger patches require more max-pooling layers that reduce the localization accuracy, while small patches allow the network to see only little context. More recent approaches [11,4] proposed a classifier output that takes into account the features from multiple layers. Good localization and the use of context are possible at the same time.

In this paper, we build upon a more elegant architecture, the so-called "fully convolutional network" [9]. We modify and extend this architecture such that it works with very few training images and yields more precise segmentations; see Figure 1. The main idea in [9] is to supplement a usual contracting network by successive layers, where pooling operators are replaced by upsampling operators. Hence, these layers increase the resolution of the output. In order to localize, high resolution features from the contracting path are combined with the upsampled

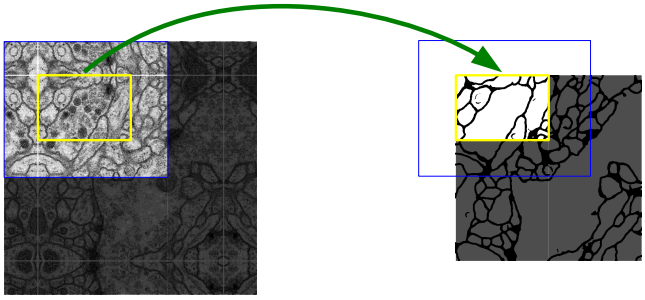


Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

output. A successive convolution layer can then learn to assemble a more precise output based on this information.

One important modification in our architecture is that in the upsampling part we have also a large number of feature channels, which allow the network to propagate context information to higher resolution layers. As a consequence, the expansive path is more or less symmetric to the contracting path, and yields a u-shaped architecture. The network does not have any fully connected layers and only uses the valid part of each convolution, i.e., the segmentation map only contains the pixels, for which the full context is available in the input image. This strategy allows the seamless segmentation of arbitrarily large images by an overlap-tile strategy (see Figure 2). To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image. This tiling strategy is important to apply the network to large images, since otherwise the resolution would be limited by the GPU memory.

As for our tasks there is very little training data available, we use excessive data augmentation by applying elastic deformations to the available training images. This allows the network to learn invariance to such deformations, without the need to see these transformations in the annotated image corpus. This is particularly important in biomedical segmentation, since deformation used to be the most common variation in tissue and realistic deformations can be simulated efficiently. The value of data augmentation for learning invariance has been shown in Dosovitskiy et al. [2] in the scope of unsupervised feature learning.

Another challenge in many cell segmentation tasks is the separation of touching objects of the same class; see Figure 3. To this end, we propose the use of a weighted loss, where the separating background labels between touching cells obtain a large weight in the loss function.

The resulting network is applicable to various biomedical segmentation problems. In this paper, we show results on the segmentation of neuronal structures in EM stacks (an ongoing competition started at ISBI 2012), where we out-

performed the network of Ciresan et al. [1]. Furthermore, we show results for cell segmentation in light microscopy images from the ISBI cell tracking challenge 2015. Here we won with a large margin on the two most challenging 2D transmitted light datasets.

2 Network Architecture

The network architecture is illustrated in Figure 1. It consists of a contracting path (left side) and an expansive path (right side). The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.

To allow a seamless tiling of the output segmentation map (see Figure 2), it is important to select the input tile size such that all 2x2 max-pooling operations are applied to a layer with an even x- and y-size.

3 Training

The input images and their corresponding segmentation maps are used to train the network with the stochastic gradient descent implementation of Caffe [6]. Due to the unpadded convolutions, the output image is smaller than the input by a constant border width. To minimize the overhead and make maximum use of the GPU memory, we favor large input tiles over a large batch size and hence reduce the batch to a single image. Accordingly we use a high momentum (0.99) such that a large number of the previously seen training samples determine the update in the current optimization step.

The energy function is computed by a pixel-wise soft-max over the final feature map combined with the cross entropy loss function. The soft-max is defined as $p_k(\mathbf{x}) = \exp(a_k(\mathbf{x}))/\left(\sum_{k'=1}^{K}\exp(a_{k'}(\mathbf{x}))\right)$ where $a_k(\mathbf{x})$ denotes the activation in feature channel k at the pixel position $\mathbf{x} \in \Omega$ with $\Omega \subset \mathbb{Z}^2$. K is the number of classes and $p_k(\mathbf{x})$ is the approximated maximum-function. I.e. $p_k(\mathbf{x}) \approx 1$ for the k that has the maximum activation $a_k(\mathbf{x})$ and $p_k(\mathbf{x}) \approx 0$ for all other k. The cross entropy then penalizes at each position the deviation of $p_{\ell(\mathbf{x})}(\mathbf{x})$ from 1 using

$$E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x})\log(p_{\ell(\mathbf{x})}(\mathbf{x})) \tag{1}$$

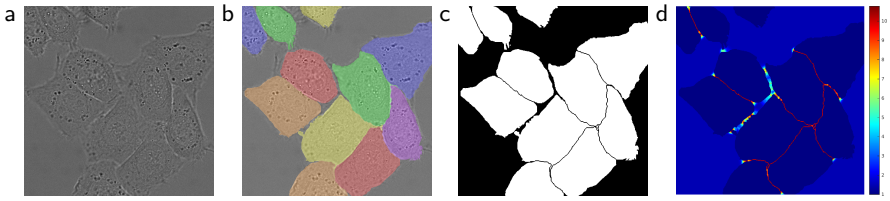


Fig. 3. HeLa cells on glass recorded with DIC (differential interference contrast) microscopy. (**a**) raw image. (**b**) overlay with ground truth segmentation. Different colors indicate different instances of the HeLa cells. (**c**) generated segmentation mask (white: foreground, black: background). (**d**) map with a pixel-wise loss weight to force the network to learn the border pixels.

where $\ell : \Omega \to \{1, \dots, K\}$ is the true label of each pixel and $w : \Omega \to \mathbb{R}$ is a weight map that we introduced to give some pixels more importance in the training.

We pre-compute the weight map for each ground truth segmentation to compensate the different frequency of pixels from a certain class in the training data set, and to force the network to learn the small separation borders that we introduce between touching cells (See Figure 3c and d).

The separation border is computed using morphological operations. The weight map is then computed as

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \tag{2}$$

where $w_c : \Omega \to \mathbb{R}$ is the weight map to balance the class frequencies, $d_1 : \Omega \to \mathbb{R}$ denotes the distance to the border of the nearest cell and $d_2 : \Omega \to \mathbb{R}$ the distance to the border of the second nearest cell. In our experiments we set $w_0 = 10$ and $\sigma \approx 5$ pixels.

In deep networks with many convolutional layers and different paths through the network, a good initialization of the weights is extremely important. Otherwise, parts of the network might give excessive activations, while other parts never contribute. Ideally the initial weights should be adapted such that each feature map in the network has approximately unit variance. For a network with our architecture (alternating convolution and ReLU layers) this can be achieved by drawing the initial weights from a Gaussian distribution with a standard deviation of $\sqrt{2/N}$, where N denotes the number of incoming nodes of one neuron [5]. E.g. for a 3x3 convolution and 64 feature channels in the previous layer $N = 9 \cdot 64 = 576$.

3.1 Data Augmentation

Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available. In case of

microscopical images we primarily need shift and rotation invariance as well as robustness to deformations and gray value variations. Especially random elastic deformations of the training samples seem to be the key concept to train a segmentation network with very few annotated images. We generate smooth deformations using random displacement vectors on a coarse 3 by 3 grid. The displacements are sampled from a Gaussian distribution with 10 pixels standard deviation. Per-pixel displacements are then computed using bicubic interpolation. Drop-out layers at the end of the contracting path perform further implicit data augmentation.

4 Experiments

We demonstrate the application of the u-net to three different segmentation tasks. The first task is the segmentation of neuronal structures in electron microscopic recordings. An example of the data set and our obtained segmentation is displayed in Figure 2. We provide the full result as Supplementary Material. The data set is provided by the EM segmentation challenge [14] that was started at ISBI 2012 and is still open for new contributions. The training data is a set of 30 images (512x512 pixels) from serial section transmission electron microscopy of the Drosophila first instar larva ventral nerve cord (VNC). Each image comes with a corresponding fully annotated ground truth segmentation map for cells (white) and membranes (black). The test set is publicly available, but its segmentation maps are kept secret. An evaluation can be obtained by sending the predicted membrane probability map to the organizers. The evaluation is done by thresholding the map at 10 different levels and computation of the "warping error", the "Rand error" and the "pixel error" [14].

The u-net (averaged over 7 rotated versions of the input data) achieves without any further pre- or postprocessing a warping error of 0.0003529 (the new best score, see Table 1) and a rand-error of 0.0382.

This is significantly better than the sliding-window convolutional network result by Ciresan et al. [1], whose best submission had a warping error of 0.000420 and a rand error of 0.0504. In terms of rand error the only better performing

Table 1. Ranking on the EM segmentation challenge [14] (march 6th, 2015), sorted by warping error.

Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [1]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

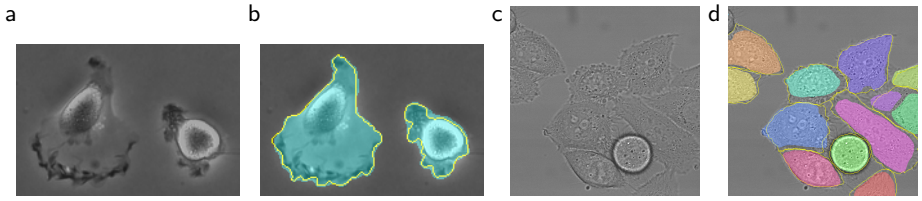


Fig. 4. Result on the ISBI cell tracking challenge. (**a**) part of an input image of the "PhC-U373" data set. (**b**) Segmentation result (cyan mask) with manual ground truth (yellow border) (**c**) input image of the "DIC-HeLa" data set. (**d**) Segmentation result (random colored masks) with manual ground truth (yellow border).

Table 2. Segmentation results (IOU) on the ISBI cell tracking challenge 2015.

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5323	-
second-best 2015	0.83	0.46
u-net (2015)	**0.9203**	**0.7756**

algorithms on this data set use highly data set specific post-processing methods[1] applied to the probability map of Ciresan et al. [1].

We also applied the u-net to a cell segmentation task in light microscopic images. This segmenation task is part of the ISBI cell tracking challenge 2014 and 2015 [10,13]. The first data set "PhC-U373"[2] contains Glioblastoma-astrocytoma U373 cells on a polyacrylimide substrate recorded by phase contrast microscopy (see Figure 4a,b and Supp. Material). It contains 35 partially annotated training images. Here we achieve an average IOU ("intersection over union") of 92%, which is significantly better than the second best algorithm with 83% (see Table 2). The second data set "DIC-HeLa"[3] are HeLa cells on a flat glass recorded by differential interference contrast (DIC) microscopy (see Figure 3, Figure 4c,d and Supp. Material). It contains 20 partially annotated training images. Here we achieve an average IOU of 77.5% which is significantly better than the second best algorithm with 46%.

5 Conclusion

The u-net architecture achieves very good performance on very different biomedical segmentation applications. Thanks to data augmentation with elastic defor-

[1] The authors of this algorithm have submitted 78 different solutions to achieve this result.

[2] Data set provided by Dr. Sanjay Kumar. Department of Bioengineering University of California at Berkeley. Berkeley CA (USA)

[3] Data set provided by Dr. Gert van Cappellen Erasmus Medical Center. Rotterdam. The Netherlands

mations, it only needs very few annotated images and has a very reasonable training time of only 10 hours on a NVidia Titan GPU (6 GB). We provide the full Caffe[6]-based implementation and the trained networks[4]. We are sure that the u-net architecture can be applied easily to many more tasks.

Acknowlegements

This study was supported by the Excellence Initiative of the German Federal and State governments (EXC 294) and by the BMBF (Fkz 0316185B).

References

1. Ciresan, D.C., Gambardella, L.M., Giusti, A., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: NIPS. pp. 2852–2860 (2012)
2. Dosovitskiy, A., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with convolutional neural networks. In: NIPS (2014)
3. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
4. Hariharan, B., Arbelez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization (2014), arXiv:1411.5752 [cs.CV]
5. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015), arXiv:1502.01852 [cs.CV]
6. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding (2014), arXiv:1408.5093 [cs.CV]
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1106–1114 (2012)
8. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural Computation 1(4), 541–551 (1989)
9. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation (2014), arXiv:1411.4038 [cs.CV]
10. Maska, M., (...), de Solorzano, C.O.: A benchmark for comparison of cell tracking algorithms. Bioinformatics 30, 1609–1617 (2014)
11. Seyedhosseini, M., Sajjadi, M., Tasdizen, T.: Image segmentation with cascaded hierarchical models and logistic disjunctive normal networks. In: Computer Vision (ICCV), 2013 IEEE International Conference on. pp. 2168–2175 (2013)
12. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556 [cs.CV]
13. WWW: Web page of the cell tracking challenge, http://www.codesolorzano.com/celltrackingchallenge/Cell_Tracking_Challenge/Welcome.html
14. WWW: Web page of the em segmentation challenge, http://brainiac2.mit.edu/isbi_challenge/

[4] U-net implementation, trained networks and supplementary material available at http://lmb.informatik.uni-freiburg.de/people/ronneber/u-net

2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on

An Efficient Hybrid Approach for Brain Tumor Detection in MR Images using Hadoop-MapReduce

Prabhjot Kaur Chahal*, Shreelekha Pandey†,
*† Computer Science & Engineering Department, Thapar institute of Engineering and Technology, Patiala (Punjab), India
(e-mail: prabhjot_kaur@thapar.edu, shreelekha.pandey@thapar.edu.)

***Abstract*—The sheer expanding of brain tumor MR image in terms of volume demands swift and accurate processing mechanism. To segregate tumor from brain MR effectively an efficient computer aided detection system is required. The research to maximize the classification accuracy rate is improving at fast pace in field of medical image processing. However, somehow setbacks the execution time of the overall system. The objective of the research is now slanting to be proficient enough in both the performance parameters. In this manuscript, the focus is to develop a framework using fusion based approach to balances the accuracy and time simultaneously. The hybrid fuzzy c-means and k-means (FKM) methodology is implemented using Hadoop MapReduce platform to increase the scalability of clusters. Experimentation is verified using DICOM images of variable size datasets processed through multi-node Hadoop platform. The size of the images is taken is slots of 200, 400 and 600 images. The accuracy rate core fusion approach (FKM) to be 96% considering the maximum number of images. The execution time decreases as the number of nodes increases due to the parallel and distributed image data. The system without hadoop framework takes 30% more time as compare to the system with single nodes using 1000 images.**

***Index Terms*—Computer aided detection, Segmentation, Brain tumor, Magnetic resonance images, Hadoop**

I. Introduction

The composite inner anatomy of brain makes the errand of brain tumor detection pivotal in the field of biomedical imaging. The imaging diagnosis procedure such as magnetic resonance imaging (MRI) is adapted to divulge various tumor types in a harmless way [1] [2]. The varying size, position and volume of the brain tumor hurdles the detection process, however effecting the segregation of tumorous part from soft brain tissues. Moreover, research in the domain of computer aided detection (CAD) is growing exponentially to assist the human experts in decision making [3] [4]. For the effective extraction, variability of MR image modalities, tumor types and data sources collectively leave an impact of the segmented output image. Traditional as well as trending techniques facilitate the research for accurate segmentation with minimal time span. According to World Health Organization (WHO) brain tumor can be classified into four grades based on how rapidly the abnormal tissue cells spreads. The more the aggressive nature of tumor tissue, the immediate is the solution required to treat.

The MR image contrast quality and the modality are the essential prerequisites for the proper tuning of any system. According to researchers the contrast density function is the backbone for the tumor detection in MR image [5]. The popularity existing MR image modalities namely, T1-weighted (T1-W), T2 weighted (T2-W), fluid attenuated inversion recovery (FLAIR), proton density weighted (PD-W), contrast enhanced T1 weighted (CE-T1W), apparent diffusion coefficient (ADC), and diffusion weighted image (DWI). The precise and accurate output of the detection system is also linked with the type of input considered, the contrast parameters concerned with the MR images, and the most important techniques applied for segmentation. The performance of the system is measured on achieving higher accuracy rates in sufficient time. Also, a range of parameters are used to evaluate the efficiency, such as accuracy (ACC), sensitivity (SEN), specificity (SPEC), similarity index (SI), Jaccard similarity coefficient (JSC), and Dice similarity coefficient (DSC). In this manuscript the performance of the system in terms of accuracy in improved by implementing hybrid fuzzy k-means approach. Furthermore, the response time of the detection is evaluated by incorporating the latest big data analytic with healthcare. The platform fused with FKM techniques is Hadoop MapReduce with helps to reduce the processing time of MR brain images. The manuscript mainly contributes in three folds:

1) For the effective brain tumor detection the knowledge related to data acquisition, modalities and tumor type all act as the preliminary requirements in preparing a CAD system. The focus is to cover all the hindrances occurring during these phases and set forth a real-time mapping diagnosis to accelerate the detection and assists experts in decision making
2) A hybrid approach is proposed to improve the performance of system. The integration of fuzzy c-means and k-means boosts the accuracy, however, covering all the existing limitations. For the reduction in response time of the proposed system, Hadoop MapReduce platform is implied. The multi-node, parallel, distributed features of the Hadoop increase the scalability of the fuzzy approach.
3) Recent trends in the MR brain tumor have expanded the utility of the data in multiple ways such as hospitals, laboratories, diagnostic centers and online repositories. Therefore, storing and managing the massive data needs to be swift enough to pass to and fro the data during processing. Hadoop distributed file system (HDFS) aid

978-1-7281-7647-5/20/$31.00 ©2020 IEEE

DOI 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics50389.2020.00144

all the data files intact.

The rest of the manuscript is organized in five more sections. Section 2 discusses the related literature work in field of MR image segmentation and approaches using Hadoop. Section 3 briefly explains the Hadoop MapReduce system model and its implication on medical imaging. Section 4 highlights the proposed hybrid fuzzy k-means system for brain tumor segmentation. Along with this, blending of hybrid approach with Hadoop platform is done. Section 5 summarizes as well as focuses on the experimental results based on segmentation accuracy and response time. Finally, Section 6 winds up with a conclusion of the manuscript and future scope.

II. Related Work

Research to precisely investigate tumor from brain MR images is of great interest now-a-days. The contribution by various researchers in the core part of segmentation is explored at high pace to achieve accuracy. Segmentation techniques based on Fuzzy c-means (FCM), k-means, self-organizing map (SOM) and other common clustering are frequently applied [6], [7]. A novel automatic segmentation technique is proposed to handle the non-enhancing tumor tissues effectively [8]. The clustering FCM is clubbed with knowledge based imaging domains proving its excellence over supervised techniques. The FCM approach not only for brain tumor proves efficient, in fact when applied on various medical imaging data gives significant segmentation results [9] [10]. The performance after experimentation using Harvard Brain Repository dataset turns to be 96.18%. The collaboration of one or more techniques continued to improve the functionality of the segmentation phase. However, FCM and k-means hybrid form KIFCM is developed to detect the tumorous part in minimal time [11]. Results of the system are tested on three popular datasets (Digital Imaging and Communications in Medicine (DICOM), BrainWeb, and BRATS). The proposed approach reports acceptable accuracy rate ranging from 90.5% to 100%. The minimum time span of 3.46s is recorded using BRATS dataset considering 81 images for experimentation.

The works solely connected with image segmentation tries hard to automate the systems to detect tumor in fraction of seconds. Reducing the processing time in this field is a difficult task as more focus is always on accuracy. However, a study is proposed to generate a balance between both the performance parameters by inculcating the optimization and clustering approaches [12] . The bacteria foraging optimization (BFO) is optimized using the modified FCM (MFKM) to detect tumor such as astrocytoma, metastatic bronchogenic carcinoma, meningioma, and neuro ectodermal. The similarity index, sensitivity and specificity obtained is 95.77%, 97.14%, and 93.94% respectively. Along with reduced computation time of 1.98 minutes to render the system performance. Further, to decrease the processing time of the system soft fuzzy based algorithm is developed known as SFRCM. The rough fuzzy sets combined so to extract the soft tissue from brain tumor. The small ranged MR images in sets of 20 and 10 images is tested and achieved an accuracy of 94.04 with gaussian noise and 87.94% with salt and pepper noise. The SFRCM dropped the execution time to 17.06s with before use to be in minutes [13]. Recently, a comparative study of clustering techniques (FCM and k-means) is performed based on the computational time and segmentation results [14]. However, the analysis proves superiority of FCM over k-means in terms of time. The achieve time rate of FCM and k-means is 8.639s and 22.831 respectively.

It is now evident that messing with accuracy and time simultaneously with any external platform is near to impossible. To make sure the massive MR image data is handled in a organised and parallel way multiple cloud and network based platforms are applied. To manage large scale MR brain image data on visual parameters analysis, content based image retrieval (CBIR) approach is applied [15]. The study is to map and anticipate between healthy and effected MR image brain on Hadoop platform. To classify the tissue abnormality support vector machine (SVM) is applied for final decision making. To speed up the performance Lucene indexing technique is used, which in turn reduces the searching time. Research on medical image retrieval is growing fast to explore the platforms of Hadoop, HDFS and MapReduce to extract texture features. Content based Image retrieval (CBIR) approach is commonly used as efficient retrieval system for medical images incorporating Hadoop (Hadoop-CBIR) [16]. The experimental evaluation of the system with high intake of images performs at minimum retrieval time with considerable accuracy. On examination, 80 image of 5GB size bestow with response time of 72ms measured with replication factor (RF) 5. Various methodologies are carried forward to segment and extract the features of the MR images in both 2D and 3D brain modalities [17] [18]. In case of 2D MR brain image data the retrieval is carried using traditional CBIR supported Hadoop MapReduce. The response time performance is tested using 10 images with RF 2 and 3 achieved 29s and 27s respectively. The 3D category of the brain MR image data is processed through Scale Invariant Feature Transform (SIFT) as the feature descriptor. The Hadoop platform for experimentation perform on varying data size such as 20, 60 and 100 images and passed to multiple nodes giving an accuracy of 80%. A study is proposed to verify and compare the cloud and network based platforms for parallel genetic approach (PGA) [19].

To test the validation of proposed framework dataset from Connectome project of varying sizes (2GB, 4GB, 6GB, 8GB and 10GB) is considered. The parallel and distributed environment through multiple clustering nodes leads to fast processing of the data. The cloud based Hadoop platform has more response time ranging from 12s to 24s than network based Spark platform which falls to 4s to 7s. The researchers developed detection systems to evaluate the performance in terms of accuracy and response time with and without distributed platforms. In this manuscript a framework with hybrid approach combining fuzzy c-means with k-means is developed to club with parallel and distributed Hadoop-MapReduce platform to increase the overall efficiency and scalability.

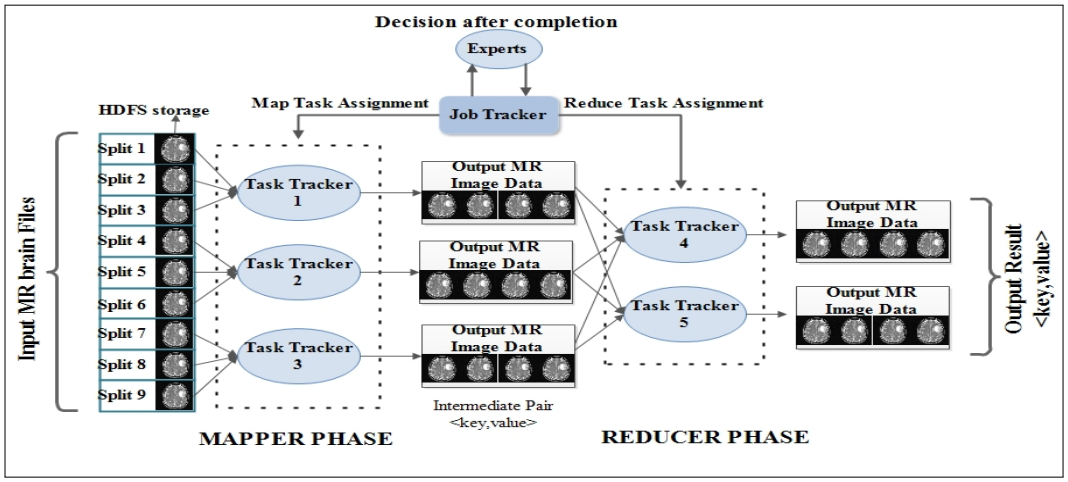


Fig. 1. The Hadoop-MapReduce system model for MR brain tumor iamges

III. Hadoop MapReduce System Model

Massive increase of MR image data is continuous due to real world utilization; however it becomes essential for image data to be processed from native storage. Managing and processing of these MR image data requires a well-defined platform so as to communicate the data to and fro. Therefore, to ensure swift processing, an open source platform comes into picture to handle massive traffic in distributed environment. Fig. 2 depicts various components of a generic Hadoop system model for MR brain images. Two basic parts of Hadoop model includes MapReduce for data processing and Hadoop Distributed File System (HDFS) for storing huge data [20]. There are certain objectives while integrating brain image processing with Hadoop, they are bulky image data needs a platform for proper storage to maintain consistency and scalability; platform reliability is another important factor influencing the image data through distributed and parallel infrastructure; lastly, the hybrid approach to extract the required output must be efficient enough to further ease the response time of the system.

The working of the Hadoop system follows a sequential phase starting from the inputting the MR image data through the links /URLs of the datasets or downloaded paths. The gathered MR image data is storage in Hadoop distributed file system (HDFS) in form of contiguous blocks of 64 MB (default) size. These splitted chunks of image data are stacked in each replicated racks with NameNode and DataNodes. The daemon associated with NameNode and DataNode is JobTracker and TaskTracker respectively. The main task of the JobTracker is the assignment of the jobs to TaskTracker of Mapper phase for further processing. The Map function is the first programming phase which considers the $< key, value >$ pair form of input and generates an intermediate $< key, value >$ pair after processing. Further, this set is passed to the Reducer phase, where Reducer function agglomerate the summation of the intermediate outcomes to generate the final output.

IV. Proposed Model

The clustering approaches applied on MR images for processing lacks scalable solution in case of massive data. However, the approach of FCM always excels any clustering approach based on the accuracy parameter whereas effecting the computation time. This time parameter is managed and controlled by the iterative k-means approach. Practically, individually approaches fails to fulfill the desired outcomes of the system. Moreover, even the integration of FCM and k-means (FKM) limits the meaningful cluster formation due to increased number of iterations, in return hampering the execution time. Though as the size of data accelerates, the difficulty to manage in the main memory persists. Nevertheless, to withstand such problem a platform (Hadoop-MapReduce) to store, process and distribute bulk data becomes mandatory. The proposed model of hybrid approach integrated with Hadoop-MapReduce Platform is depicted in Fig. 2.

In the first phase of the proposed system the MR brain tumor dataset is collected and preprocessed for further processing of the heavy sized data images. The prepossessing module goes through all the noise removal and filtering mechanism to improve the image quality. These preprocessed images are storage in HDFS, which spilt the image data into the chunks of 128 MB for next level processing. The task is divided here and forwarded to various mapper function to spilt the work and process in parallel fashion. The core proposed segmentation approach of FKM is implemented in MapReduce phase to segment the tumor from MR brain images sequentially. At the same time through multiple nodes this process of segmentation is carried to save the processing time of the system.

A. Segmentation using hybrid fuzzy k-means clustering

The Fuzzy K-mean (FKM) scheme works by fuzzifying all the extracted pixel values of the image based on the membership function. The spatial context of FCM utilizes the information such as illumination and intensity to formulate the clusters. The membership function of the proposed

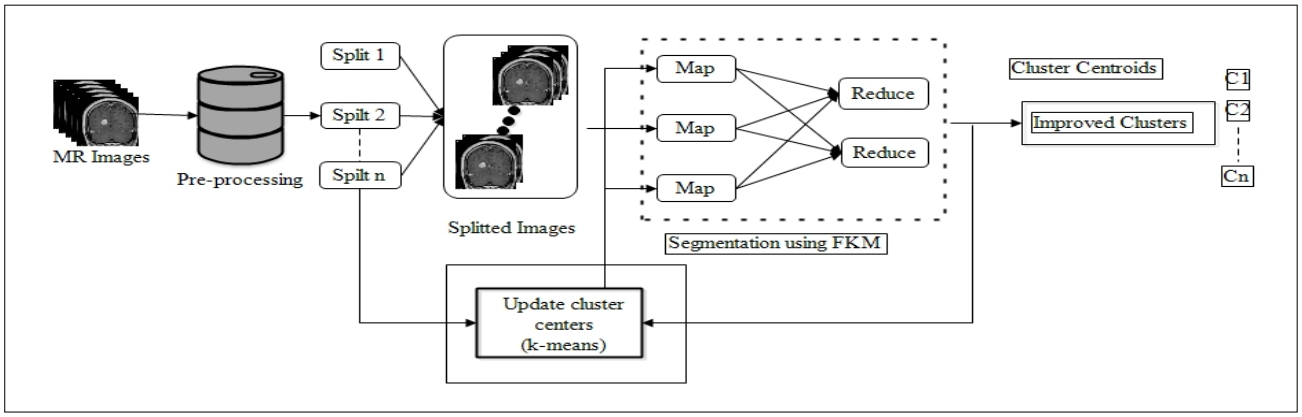


Fig. 2. Proposed fusion based approach for brain tumor detection using Hadoop-MapReduce mechanism

approach is the modification of the standard FCM method which is defined below.

$$J(U,V) = \sum_{i=1}^{n}\sum_{j=1}^{k}(u_{ij})^m \parallel x_i - \nu_j \parallel^2 \quad (1)$$

Thus, the issues occurring due to FCM in handling scattered or multiple membership clusters leads to unclear segments which are conquered by the modified Fuzzy c-means.

The membership function is denoted by $U = u_{i,j} \in [0,1], i = 1,2,\ldots,n,\ j = 1,\ldots.c.$ The value associated with each pixel $u_{i,j}$ tells the degree to which pixel x_i belongs to cluster j. Membership function $(u_{i,j})$ is calculated using Eq. 2.

$$u_{i,j} = \frac{1}{\sum_{k=1}^{c}(d_{ij} - d_{ik})^{\frac{2}{m-1}}} \quad (2)$$

$$\sum_{j=1}^{c} u_j(x_i) = 1 \quad (3)$$

$$v_j = \frac{\sum_{i=1}^{n}(u_{ij})^m x_i}{\sum_{i=1}^{n}(u_{ij})^m} \quad (4)$$

According to Eq. 3 the summation of each membership for each data point should be equal to 1. The Euclidean distance between the i^{th}data and j^{th}cluster centre as defined in Eq. 4. The proposed fuzzy c-means algorithm works towards minimization of the objective function $J(U,V)$ defined in Eq. 1.

The FKM approach for MR Image segmentation is discussed in Alg. 1. To overthrow the issue of shattered clusters and blurred segments created by FCM, The membership function is updated.

B. The algorithm for MapReduce Approach

MapReduce approach split the complete datasets into set of individual jobs further computation and processing is done on this split dataset in a parallel fashion. This chunk of processing is executed by the maps, then, forwarded to the reducer for processed result.

Algorithm 1 Standalone FKM based algorithm

Input: MR Images I
Output: Segmented image $I^{'}$

1: **for** $i \neq null$ **do**
2: **for** $j \leq n$ **do**
3: Randomly select the cluster centers
4: Calculate fuzzy membership $u_{ij} = \frac{1}{\sum_{k=1}^{c}(d_{ij}-d_{ik})^{\frac{2}{m-1}}}$
5: Calculate fuzzy center $v_j = \frac{\sum_{i=1}^{n}(u_ij)^m x_i}{\sum_{j=1}^{n}(u_{ij})^m}$
6: Update the clusters using Euclidean distance $(d_{ij} - d_{ik})$
7: Update the centers and repeat the steps 3 to 6 till u_{ij} is achieved.
8: Image gradient $\rightarrow \mathbb{I}^{'}$
9: **end for**
10: **end for**

V. Results and Discussions

The prime motive of the manuscript is to propose an efficient segmentation system for MR images to retrieve the tumorous part of the brain. The hybrid FKM is tested and validated using variable size data set taken from The Cancer Imaging Achieve (TCIA) of Digital Imaging and Communications in Medicine (DICOM) format. To prove the effectiveness of the approach high grade and low grade gliomas MR image segmentation is compared with some traditional approaches. The visual representation of the tumor segregation is depicted in the Fig. 3 and Fig. 4, hence showing the evince of improvement. On comparison with existing approaches, the current approach excels in terms of accuracy as well as response time. The accuracy of FKM turns to be 96% using 1000 DICOM images. The exact performance evaluation comparative study has been summarized in Table I with same dataset. The proposed hybrid approach is able to handle large

TABLE I
PERFORMANCE EVALUATION BASED ON ACCURACY USING SAME DATASET

Reference	Approach	Size of Dataset (DICOM)	Performance (%)
[11]	KIFCM	22 images	ACC: 90.5
[21]	FKM + (Wndchrm classifier & VGG 19 DNN)	168 images & 228 images	ACC: 92.86 (Wndchrm classifier) ACC: 98.25 (VGG-19 DNN classifier)
[22]	FKM & FSVM	200 images	ACC: 96
[23]	FSVM	80 images	ACC: 94
Proposed Approach	Hybrid FKM + Hadoop plaform	1000 images	ACC: 96

ACC: Accuracy, DNN: deep convolutional neural network, FKM: Fuzzy c-means with k-means, FSVM: Fuzzy support vector machine, KIFCM: K-means clustering with fuzzy c-means algorithm, Wndchrm classifier: Weighted Neighbor Distance using Compound Hierarchy of Algorithms Representing Morphology, VGG-19.

Algorithm 2 Hadoop implementation for FKM segmentation algorithm

Input: Cluster of I images
Output: Report file of selected I images

1: **FUNCTION: FKM_MAP(key, value)** ▷ key: Directory index, value: Image index
2: Calculate the Euclidean distance $(d_{ij} - d_{ik})$ using Eq. 2
3: Compute the fuzzy membership u_{ij} using Eq. 1
4: Create a buffer memory to maintain the log that map the vector point with a cluster.
5: Segmented Image $I^{'} \leftarrow$ Algo. I
6: **if** $\mathbb{I}^{'} \neq null$ **then**
7: **file.Write**(key,$\mathbb{I}^{'}$)
8: **end if**
9: **END FUNCTION**
10: **FUNCTION *FKM_REDUCE(key, value)*** ▷ key: HDFS path, value: Image index
11: **for** ($R_{img} \in \mathbb{I}^{'}$) **do**
12: M_{file}=HDFS.**generateNewFile**(R_{img}.**pullName**() +."pixels")
13: **for** (l=0 to l $\leq$R$_{img}$.**pullWidth**()) **do**
14: **for** (q=0 to q $\leq$R$_{img}$.**pullHeight**()) **do**
15: $P_{val} \leftarrow \mathbb{I}^{'}$.**pullPixel**($l.q$)
16: **if** M_{val} is active **then**
17: M_{file}.**Write**(l,q)
18: **end if**
19: **end for**
20: **end for**
21: **end for**
22: **ENDFUNCTION**

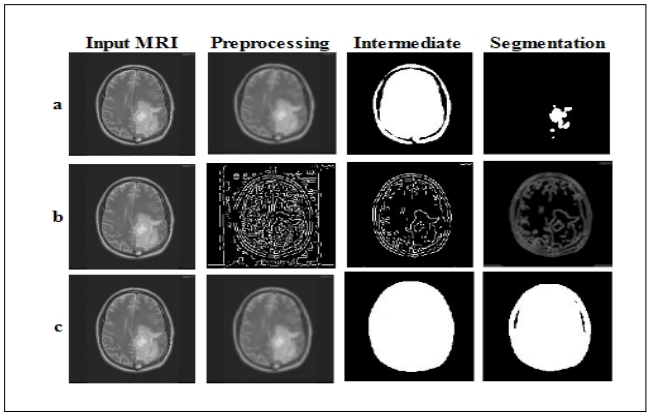


Fig. 3. Visual Segmentation results of High grade gliomas (HGG) using MR Images. a. Proposed FKM, b. Canny edge detection c. k-means

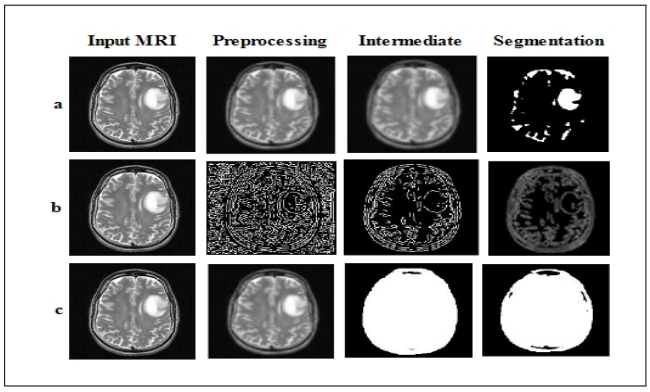


Fig. 4. Visual Segmentation results of Low grade gliomas (LGG) using MR Images. a. Proposed FKM, b. Canny edge detection c. k-means

dataset with good accuracy results.

The response time parameter is improved with Hadoop

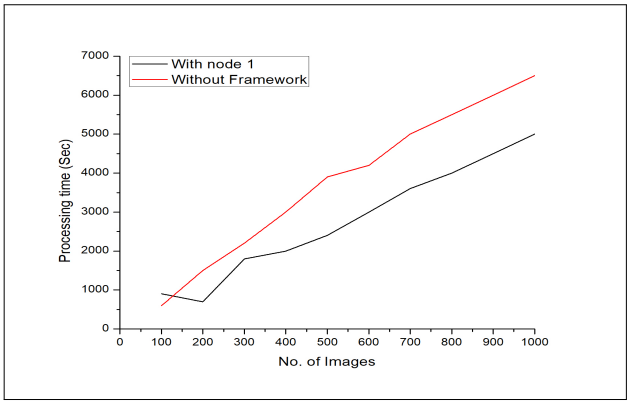


Fig. 5. FKM detection computation executed using hadoop framework with one node and without using framework

platform and for comparison, different bunch of MR images are used like, 100-1000 DICOM images. Two platforms are used for testing the proposed approach. In first approach, the testing is done on the standalone system using proposed scheme. In other approach, Hadoop based platform is used for testing the bunch of MR images as shown in Fig. 5. In this system, the initial computation time is higher as compared to the time consumed to test the same set of MR images on the standalone platform. Here, HDFS ease the computation by creating bunches of MR images which help the DataNodes.

VI. Conclusion and Future Directions

A deliberation can be concluded that the medical image processing is gaining rapid interest in the research domain. The first and the foremost objective of this manuscript is to implement an approach which excels in terms of accuracy and ease the segregation with Hadoop-MapReduce platform. The hybrid approach overcomes the limitations of existing FCM and k-means algorithms, however covering the multiple membership and iterations issues. Furthermore, the second motive is to target the response time of the complete framework which is improved using Hadoop. The result of segmentation of gliomas with accuracy rate turns to be sufficient well than existing ones. The time parameter compared with single- node on Hadoop is less than without the platform due to parallel and distributed nature of Hadoop. Moreover, the research can be extended in the field of brain tumor detection using variable datasets on multi node applications.

References

[1] N. Gordillo, E. Montseny, and P. Sobrevilla, "State of the art survey on mri brain tumor segmentation," *Magnetic resonance imaging*, vol. 31, no. 8, pp. 1426–1438, 2013.

[2] A. F. Gaillard. Brain tumors. Www.radiopaedia.org/articles/brain-tumours.

[3] M. Liu, F. Li, H. Yan, K. Wang, Y. Ma, L. Shen, M. Xu, A. D. N. Initiative *et al.*, "A multi-model deep convolutional neural network for automatic hippocampus segmentation and classification in alzheimer?s disease," *NeuroImage*, vol. 208, p. 116459, 2020.

[4] R. Saouli, M. Akil, R. Kachouri *et al.*, "Fully automatic brain tumor segmentation using end-to-end incremental deep neural networks in mri images," *Computer methods and programs in biomedicine*, vol. 166, pp. 39–49, 2018.

[5] E.-S. A. El-Dahshan, H. M. Mohsen, K. Revett, and A.-B. M. Salem, "Computer-aided diagnosis of human brain tumor through mri: A survey and a new algorithm," *Expert systems with Applications*, vol. 41, no. 11, pp. 5526–5545, 2014.

[6] M. Y. Siyal and L. Yu, "An intelligent modified fuzzy c-means based algorithm for bias estimation and segmentation of brain mri," *Pattern recognition letters*, vol. 26, no. 13, pp. 2052–2062, 2005.

[7] F. G. Zöllner, K. E. Emblem, and L. R. Schad, "Svm-based glioma grading: optimization by feature reduction analysis," *Zeitschrift für medizinische Physik*, vol. 22, no. 3, pp. 205–214, 2012.

[8] S. Madhukumar and N. Santhiyakumari, "Evaluation of k-means and fuzzy c-means segmentation on mr images of brain," *The Egyptian Journal of Radiology and Nuclear Medicine*, vol. 46, no. 2, pp. 475–479, 2015.

[9] C. Chakraborty, "Computational approach for chronic wound tissue characterization," *Informatics in Medicine Unlocked*, vol. 17, p. 100162, 2019.

[10] C. Chakraborty, B. Gupta, S. K. Ghosh, D. K. Das, and C. Chakraborty, "Telemedicine supported chronic wound tissue prediction using classification approaches," *Journal of medical systems*, vol. 40, no. 3, p. 68, 2016.

[11] E. Abdel-Maksoud, M. Elmogy, and R. Al-Awadi, "Brain tumor segmentation based on a hybrid clustering technique," *Egyptian Informatics Journal*, vol. 16, no. 1, pp. 71–81, 2015.

[12] A. Vishnuvarthanan, M. P. Rajasekaran, V. Govindaraj, Y. Zhang, and A. Thiyagarajan, "An automated hybrid approach using clustering and nature inspired optimization technique for improved tumor and tissue segmentation in magnetic resonance brain images," *Applied Soft Computing*, vol. 57, pp. 399–426, 2017.

[13] A. Namburu, S. kumar Samay, and S. R. Edara, "Soft fuzzy rough set-based mr brain image segmentation," *Applied Soft Computing*, vol. 54, pp. 456–466, 2017.

[14] B. Srinivas and G. S. Rao, "Unsupervised learning algorithms for mri brain tumor segmentation," in *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*. IEEE, 2018, pp. 181–184.

[15] A. Palle and R. Kulkarni, "Classification of medical mri brain images based on hadoop," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies*, 2016, pp. 1–4.

[16] R. K. Grace, R. Manimegalai, and S. S. Kumar, "Medical image retrieval system in grid using hadoop framework," in *2014 International Conference on Computational Science and Computational Intelligence*, vol. 1. IEEE, 2014, pp. 144–148.

[17] J. Patil and D. G. Predeepini, "Two dimensional medical images diagnosis using mapreduce," *Indian Journal of science and Technology*, vol. 9, no. 17, 2016.

[18] J. S. Patil and G. Pradeepini, "Three-dimensional mri brain image analysis on hadoop platform," in *International Conference on Intelligent Computing and Applications*. Springer, 2018, pp. 131–142.

[19] D. P. Augustine and P. Raj, "Performance evaluation of parallel genetic algorithm for brain mri segmentation in hadoop and spark," *Indian J. Sci. Technol*, 2016.

[20] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*. IEEE, 2010, pp. 1–10.

[21] A. M. KV, V. Rajendran *et al.*, "Glioma tumor grade identification using artificial intelligent techniques," *Journal of medical systems*, vol. 43, no. 5, p. 113, 2019.

[22] P. K. Chahal, S. Pandey, and S. Goel, "Hybrid approaches for brain tumor detection in mr images," in *International Conference on Advances in Computing and Data Sciences*. Springer, 2019, pp. 264–274.

[23] A. Jayachandran and R. Dhanasekaran, "Brain tumor detection using fuzzy support vector machine classification based on a texton co-occurrence matrix," *Journal of imaging Science and Technology*, vol. 57, no. 1, pp. 10 507–1, 2013.