Embedded-ViT: A Framework for Embedded Deployment of Vision-Transformer in Medical Applications

No Author Given

No Institute Given

Abstract. Transformer architectures have dramatically influenced the field of natural language processing and are becoming more popular in the computer vision field, too. However, the Transformer's core self-attention mechanism has quadratic computational complexity concerning the number of tokens. Thus, they usually require big GPUs for deployment, contrary to the Internet of Things trend, which enables the mobile deployment of AI applications, which involves the development of efficient, lightweight neural networks to meet the strict hardware limitations of the target platforms. The cost and ease of deployment are even more critical in the medical field, and not every clinic can afford to buy a lot of powerful GPUs to aid the physicians. Therefore, research proposed some methods to achieve more efficient transformer networks, but to our knowledge, very limited work targeted a level of complexity reduction that allows the embedded deployment of transformers in the medical field. In this paper, we propose our Embedded-ViT framework with which we can drastically reduce the complexity of standard vision transformer (ViT) networks. We accomplish that using several compression techniques: efficient model architecture changes, reduced input resolution, pruning, or quantization. Our optimizations can significantly compress the model, while maintaining a desired prediction quality level. We prove the capabilities of our framework by applying it to a state-of-the-art ViT and its variations. We tested the results of our Embedded-ViT on the publicly available Synapse dataset for multi-organ segmentation. Our framework cuts the computational load by half while maintaining a slightly higher level of prediction quality. Moreover, we will thoroughly analyze the hardware requirements and throughput achieved on different platforms, including the embedded Jetson Nano from Nvidia. The framework is open-source and accessible online at `https://BlindedLinkForReview`.

Keywords: Semantic Segmentation · Lightweight · Embedded Deployment · CAD · Computer Vision · Vision Transformer

1 Introduction

Diagnosing diseases is a very laborious and time-intensive task. The number of physicians available is minimal, and reducing their workload would greatly

benefit them and their patients. Thus, multiple use cases for computer-aided diagnostics (CAD) have emerged to automate some of the work of physicians. Examples of fields that research targeted are EEG analysis [1], super-resolution of MRI images [2], or classification to detect various diseases[3]. However, a simple' yes' or' no' classifier is not often sufficient when it comes to such delicate decisions as diagnosing if a tumor is benign or malignant. Therefore, a network that can highlight the specific area that led it to its decision would offer a much higher degree of explainability and usability for a physician. Towards this, bounding boxes were first explored [4], but the later emerging semantic segmentation networks [5] offered a much higher degree of explainability with their pixel-wise classification. State-of-the-art in semantic segmentation has also transformed a lot, starting with the simple AlexNet, to the deeper ResNet with its skip connections and the in the medical field very popular U-Net [6]. The U-Net architecture itself underwent several iterations to create even higher quality prediction while getting more efficient [7]. Regardless, the one thing familiar with each U-Net is the decoder and encoder structure. Nevertheless, in recent times, the Transformer architecture has emerged as a very effective tool for computer vision due to its ability to detect long-range dependencies. However, the benefits of the transformer architecture come with the drawback of quadratic computational complexity concerning the number of tokens in the self-attention module. Thus, research has focused on creating more efficient transformer models to address those problems. Some solutions propose hybrid models that combine Transformer and convolutional networks, while others propose redesigning the attention mechanism. However, we propose our Embedded-ViT framework, where we not only tackle a Transformer's architecture and apply multiple proven compression methods to increase efficiency [8]. Regarding architectural changes, we observe that state-of-the-art transformers also have adapted a decoder/encoder-like architecture like U-Net. Thus, we significantly reduced the parameters of a state-of-the-art ViT by removing one encoder and decoder level and adapting some hyperparameters. Furthermore, we also experimented with a reduction of the input resolution. Although this drastic measure comes with considerable losses in predictive quality, it effectively reduces the network's size and complexity, allowing for deployment on very limited embedded platforms. Moreover, we can also apply a pruning library that systematically removes model weights without significantly reducing prediction quality. However, this option limits us to using a specific deployment library, which excludes other methods like quantization due to its experimental nature.

Suppose we do not want to be limited to using a specific deployment library. In that case, we can compress our model via quantization to FP16 precision, giving us room for the trade-off between the degree of model compression and prediction quality. To summarize, most proposed methods for efficient ViTs do not focus on possibilities for model compression and thus do not offer a holistic approach toward real-world deployability. Meanwhile, our Embedded-ViT shows the broader potential of compression techniques to enable embedded deployment of ViTs. Fig. 1, compares the model complexity to the Dice score results

on the Synapse dataset between the state-of-the-art ViTs and Embedded-ViT. Embedded-ViT achieves comparable results in Dice score but at comparatively much lower model complexity.

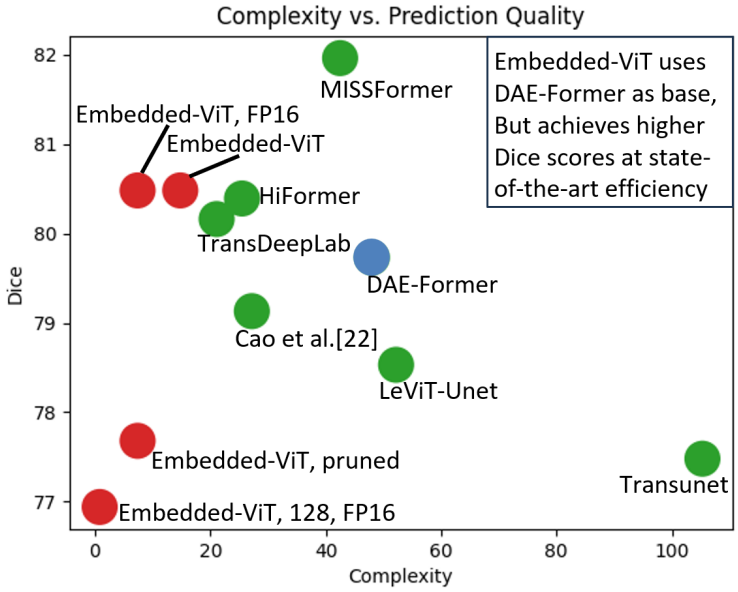

Fig. 1. Comparison of Dice score vs. model complexity between the state-of-the-art and our Embedded-ViT variations.

In this paper, we will show a comprehensive comparison of all our compression methods and their trade-offs. We use a publicly available multi-organ segmentation dataset and multiple deployment platforms for the performance analysis, specifically a GPU, a CPU, and Nvidia's Jetson Nano as an embedded platform to illustrate the benefits of our approach. **The key contributions of this work are:**

1. Our Embedded-ViT framework offers a range of compression techniques to greatly increase model efficiency while simultaneously minimizing the loss in prediction quality.
2. This framework shows a road map for efficient compression of any U-Net-shaped ViT architecture
3. We present a detailed analysis and ablations of the results of our framework applied to a state-of-the-art ViT on the Synapse dataset for multi-organ segmentation to prove its efficacy.
4. This work is accessible online at `https://BlindedLinkForReview`.

2 Related Works

This section reviews previous works that targeted efficient ViTs. Convolutional neural networks (CNN) dominate the vision field. Nevertheless, due to their filters, they cannot understand long-range dependencies, which is crucial for understanding medical images. Therefore, the first ViT [9] was introduced to offer an alternative to the convolution-based vision networks. Its self-attention mechanism is an excellent tool to understand long-range dependencies. However, the downside is that the Transformers are much more computationally demanding than CNNs and need bigger datasets for training to achieve high-quality predictions. Thus, much research has emerged on addressing those specific problems. Transunet by chen et al. [10] combines Transformers with U-Net. They achieved it by using the output of a CNN as input for the transformer to encode the feature maps. Then, the decoder upsamples the encoded feature maps and combines them with the original output of the CNN. HiFormer [11] improves the efficiency of a CNN/ Transformer hybrid architecture by using two encoders, a Transformer module and a CNN. Furthermore, they propose a Double-Level Fusion of a module in the skip connection between the encoder and decoder. Azad et al. [12] also utilized a combination of a CNN encoder to capture local information and a transformer to model the long-range dependency. Furthermore, they extract the hierarchical features and use them as input for their contextual attention module to recalibrate the representation space using local information adaptively. Ding et al. [13] chose a different approach; rather than combining CNN and Transformers; they restricted the self-attention mechanism to use spatial tokens and channel tokens, where for the spatial tokens, the spatial dimension defines the token scope, and the channel dimension defines the token feature dimension and vice versa for the channel tokens. . Ali et al. [15] proposed an efficient version of self-attention that operates across feature channels rather than tokens, thus achieving linear complexity. Cao et al. [16] introduced their Swin-Unet, a Unet-like pure Transformer. Swin-Unet is a hierarchical Swin

Table 1. Comparison of state-of-the-art efficient ViTs.

Methods	Architecture	Self attention-mechanism	Compression
Transunet [10]	✓	✗	✗
HiFormer [11]	✓	✗	✗
Azad et al. [12]	✓	✗	✗
Ding et al. [13]	✗	✓	✗
Shen et al. [14]	✗	✓	✗
Ali et al. [15]	✗	✓	✗
Cao et al. [16]	✓	✓	✗
Swiftron [17]	✗	✗	✓
DAE-Former. [18]	✓	✓	✗
Embedded-ViT (Ours)	✓	✓	✓

Transformer using shifted windows as the encoder. Shen et al. [14] developed a novel efficient attention mechanism equivalent to typical dot-product attention but with substantially less memory and computational costs. In SwiftTron [17] the authors proposed an efficient specialized hardware accelerator designed for Quantized Transformers. DAE-Former [18] combined the approaches of Ali et al. [15] and Shen et al. [14] to create an efficient self-attention model and proposed a novel skip connection path that includes a cross-attention module, which ensures the feature reusability and enhances the localization power.

Table. 1 highlights improvement aspects in the state-of-the-art for efficient ViTs. We observed that our method used most of the popular techniques to create efficient ViTs. However, state-of-the-art approaches focus on optimizing the self-attention mechanism and base architecture. Our Embedded-ViT is the only approach that focuses on minimizing the cost of deployment by using lower-resolution inputs, pruning, and quantization.

3 Our Framework

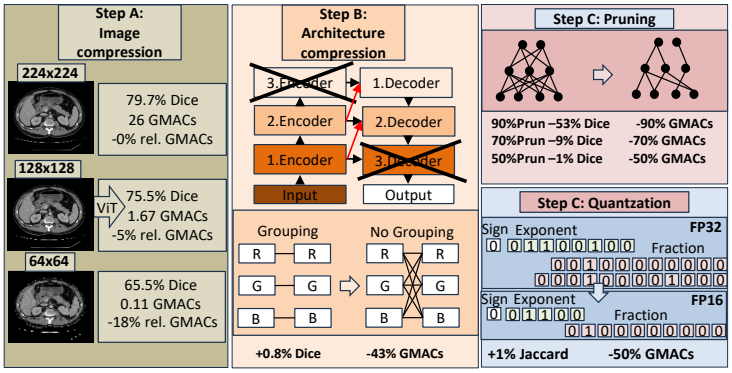

Fig. 2. Overview of Embedded-ViT framework. Our framework proposes four different compression methods: Step A, compression of input images; Step B, changes in model architecture towards efficiency; Step C, model pruning; and Step D, quantization of the model weights.

Fig. 2 presents an overview of our Embedded-ViT framework. We can use any pure U-Net-shaped Transformer as a basis for our Embedded-ViT framework, and all of our compression methods are applicable. Step A outlines the compression of the input image. Image compression is a simple but very effective step in reducing the computational load of the model. It is often necessary to be able to run the model on heavily memory-limited embedded platforms. Unfortunately, this measure comes with drops in prediction quality. Step B describes removing encoder and decoder levels and other minor architectural changes. The resulting

shallow network contains much fewer weight parameters and computations. We also observed a slightly positive effect regarding the prediction quality in our use case. Step C highlights the use of pruning for further network compression. Pruning describes the removal of network weights to reduce the overall number of parameters and computations. The user can decide the number of weights pruned; however, the higher the degree of compression, the lower the prediction quality. Finally, step D delineates quantization techniques to compress the model further; we limit ourselves to FP16 quantization as it is widely supported and results in minimal deterioration in prediction quality. Our baseline network for our experiments is the DAE-Former, a state-of-the-art efficient ViT.

Step A

In step A, we compress the input images from the original input resolution of 224×224 to two alternative input resolutions of 128×128 and 64×64. Compressing the input further, for example, to a 32×32 resolution, was impossible with our baseline network DAE-Former; this would require significant changes in model architecture as the model is too deep for such low input resolution. Since the input image runs through the network, after each encoder level, the network reduces the resolution of the feature map in exchange for creating more channels. Therefore, we must start with a sufficient high-resolution input image to go through all encoder layers. Note that in our baseline network, the user was never intended to specify the input resolution. Thus, for us to be able to use the smaller input image, we were required to adapt the hard-coded parameters inside the transformer network to make sure that all the shapes still fit together. The result of reducing the input resolution by half or even a quarter of the original is a reduction in computation complexity of roughly 80% to 95%, while often only losing around 5% to 15% points prediction quality.

Step B

Next, we move on to elaborate on step B. Our baseline network, the DAE-Former, is a U-Net-shaped pure Transformer consisting of three encoder and decoder levels. Note that our framework applies to any U-Net-shaped pure transformer architecture. We removed one encoder and decoder level, which means our updated network has two encoder and decoder levels instead of three encoder and decoder levels. We observed that the network's last encoder and decoder levels increased its complexity by 65%, as the later encoder and decoder levels are the layers with the most parameters. On the one side, removing one decoder layer for such a network can be done relatively quickly by simply re-shaping the output of the previous decoder layer. On the other side, removing an encoder layer proves to be more complicated, as we are now removing a significant block in the middle of the network and thus requires us to adapt many parameters and skip connections inside the network so that the shapes still fit. Usually, we would expect that reducing the depth of a neural network comes with a reduction in prediction quality; however, our use case saw even a slight improvement in prediction quality, showing that for the specific dataset, the original network was maybe too deep and maybe over-fitted. Moreover, we also experimented with removing the grouping in the convolutional layers of the network. Group-

ing is a method that splits input and output channels into groups and performs the convolutions independently for each group. Therefore, grouping leads to a more sparsely connected network and fewer computations. After we removed the grouping, the network became computationally more complex, requiring more memory but generating higher-quality prediction.

Step C

Step C details the Pruning process. Pruning describes setting a certain prede-

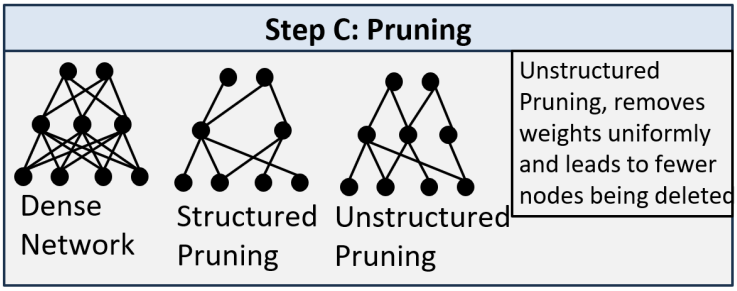

Fig. 3. A Comparison between dense networks, structured and unstructured pruning.

fined ratio of model weights to 0. Pruning leverages the fact that we can remove a surprising amount of weights and still achieve the same level of prediction quality in specific models. Removing up to 80% of model weights is possible depending on the use case while keeping the original prediction quality [19]. However, Pruning is not as popular as results suggest in real-world deployment. One problem is that the typical deep learning deployment engines do not leverage sparse matrix operation, as a standard neural network is usually very dense. Thus, if we successfully prune 80% of the model weights, it would not affect the throughput rate. Therefore, if we want to utilize the effects of Pruning, we are bound to use specific deployment engines that can exploit the benefits of pruned models. Our framework uses the Sparseml library for Pruning and the Deepsparse engine for deployment, both from Neural Magic [20]. In general, Pruning can be divided into two categories: structured Pruning and unstructured Pruning as Fig. 3 highlights. In structured Pruning, we remove model weights according to a strategy; for example, delete every fourth weight or delete weights in blocks of size 2×2. Conversely, unstructured Pruning does not limit the distribution of the removed weights and is, therefore, more effective. Hence, our framework will use unstructured Pruning, as the Sparseml library provides this option. Moreover, we can further categorize the pruning strategies into post-training and training-aware Pruning. The Sparseml library does have automatic pruning recipes, but only for very specific models, and for our use case, we had to create the pruning recipe independently. However, we are deploying the pruned model with the Deepsparse engine; this limits us to the CPU as Deepsparse's deployment engine is incapable of GPU deployment. Therefore, if we want to deploy our model on

a machine with a GPU, like Nvidia's Jetson Nano, we can not utilize Step C but can apply Step D.

Step D Since our chosen Pruning approach only applies to CPU deploy-

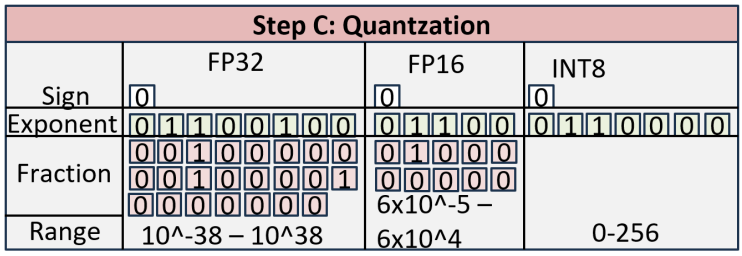

Fig. 4. A Visualization of the space occupied by FP32, FP16 and INT8 precision.

ment, we offer alternative quantization here. Quantization refers to mapping the model parameters, which are saved in FP32 precision, to a lower precision. Much research is dedicated to using INT8 quantization, which would result in just using a quarter of the model's original complexity and memory. Moreover, some research is investigating the use of INT4 quantization to compress models further. However, we encountered numerous problems when mapping the model to INT8. Using Sparseml and Deepsparse's built-in quantization methods led to unresolvable errors. Similarly, Pytorch's quantization approach required us to change the model architecture, which still produced a model that could not run. This is to be expected as we are dealing with state-of-the-art models that often use novel modules, while at the same time, quantization is still very limited to the operations it supports. Therefore, we resort to only using FP16 quantization in our framework. FP16 quantization is widely supported, and while it is still half the model's complexity, it leads to a very minimal decrease in prediction quality. Similar to Pruning, we will use training-aware quantization to maximize the prediction quality possible with FP16 operations. Note that Deepsparse's deployment engine is also incapable of handling FP16 operations and therefore using Step D excludes us from using Step C.

4 Results and Discussion

In this section, we review the results of our experiments and compare the different proposed compression methods and their efficacy.

For the performance evaluation, we also ran the networks on a GPU and a CPU and for the embedded platform, Nvidia's Jetson Nano. We used the Dice score as the evaluation metric for all experiments. We evaluate our method on the open-source Synapse multi-organ segmentation datasets [16,21]. The dataset contains 30 cases with 3779 axial abdominal CT slides and their corresponding ground truths. The task of the dataset set is to correctly segment eight organs,

Table 2. Comparison of the parameters and Dice score of Embedded-Vit with DAE-Former baseline and the state-of-the-art on the Synapse multi-organ segmentation dataset. Emb.-ViT gr. refers to the DAE-Former with Step B applied and keeping grouping convolutions.

Method	Par.	**All**	Aor.	Gall.	L.Kid.	R.Kid.	Liv.	Pan.	Spl.	Sto.
U-Net [6]	**14.8**	76.85	**89.1**	69.72	77.77	68.60	93.43	53.98	86.67	75.58
Transunet [10]	105.3	77.48	87.23	63.13	81.87	77.02	94.08	55.86	85.08	75.62
Cao et al. [16]	27.2	79.13	85.47	66.53	83.28	79.61	94.29	56.58	90.66	76.60
LeViT-Unet [22]	52.2	78.53	78.53	62.23	84.61	80.25	93.11	59.07	88.86	72.76
DAE-Former [18]	48.0	79.73	87.20	67.30	82.84	80.21	94.41	57.89	89.88	78.07
Tr.DeepLab [12]	21.1	80.16	86.04	69.16	84.08	79.88	93.53	61.19	89.00	78.40
HiFormer [11]	25.5	80.39	86.21	65.69	85.23	79.77	94.61	59.52	90.99	**81.08**
MISSFormer [21]	42.5	81.96	86.99	68.65	85.21	82.00	94.41	65.67	91.92	80.81
Emb.-ViT gr.	**14.8**	80.48	85.10	68.76	84.43	78.89	94.39	63.70	90.70	77.87
Embedded-ViT	127.5	**82.92**	88.39	**72.28**	**87.50**	**83.61**	**94.80**	**65.96**	**91.74**	79.03

namely: Aorta, Gallbladder, Left Kidney, Right Kidney, Liver, Pancreas, Spleen, and Stomach.

In Table 2, we compare the Dice scores and the number of parameters of the state-of-the-art on the Synapse dataset on a class-by-class basis. Embedded-ViT is a DAEFormer where we applied only Step B, so those results were generated without pruning, quantization, or compressed inputs. First, our Embedded-ViT model without grouping achieves the highest quality results in all classes except the stomach. However, this comes at the price of being the biggest model with 127.47 million parameters. Still, this model is a viable alternative if the only important factor is the Dice score. On the contrary, our Embedded-ViT with grouping has 14.8 million parameters, the fewest parameters together with the original U-Net. Moreover, although Embedded-ViT uses DAEFormer as a basis, which contains over 48 million parameters, it achieves 80.48 overall, the best Dice score next to MISSFormer and Embedded-ViT without grouping, and improves its basis by 0.7. Those results show that Step B alone successfully generates a state-of-the-art efficient ViT. Note that the state results for DAE-Former are the scores we could replicate, given their official GitHub repository.

In Table 3, we highlight the effect of our proposed compression methods on the baseline and our model in terms of the number of parameters and giga multiply and accumulate operations (GMac). First, the size of the respective models will be compared using the parameters. On the one side, we observe that our model with just Step B compression and without grouping is the biggest model with 333.98 million parameters, showing that the added dense connections through not using groups increased the model size eleven times compared to using groups. On the other side, when using grouping, we successfully reduced the number of parameters by 40% with Step B compared to the baseline DAE-Former. Moving over to the number of GMacs, we have a similar picture; Step B without grouping increases GMacs eleven times compared to the baseline, and Step B with grouping almost halves the Baseline GMacs Next, we compare the

Table 3. Comparison of the number of model parameters and GMacs of Embedded-Vit with DAE-Former baseline using all our compression methods.

Methods	Parameters (M)			GMacs		
Resolution	224^2	128^2	64^2	224^2	128^2	64^2
DAE-Former	48.07	9.83	2.53	26	1.67	0.11
Step B gr.	29.26	5.08	1.31	14.8	0.88	0.06
Step B	333.98	53.82	13.48	127.47	6.98	0.44

Table 4. Comparison of the Dice score of Embedded-Vit with DAE-Former baseline using all our compression methods.

Methods	Dice		
Resolution	224^2	128^2	64^2
DAE-F.	79.73	75.54	65.48
Step B gr.	80.48	76.61	67.47
Step B	82.92	78.78	67.23
DAE-Former 50%P.	78.74	73.23	55.63
Step B gr. 50%P.	77.68	74.07	63.92
Step B 50%P.	81.56	77.66	66.55
DAE-Former FP16	80.88	75.41	60.81
Step B gr. FP16	80.48	76.94	66.97
Step B FP16	82.36	78.71	67.22

effect of Step A compression. Reducing the input resolution by half results in at least 80% fewer parameters, which is a massive decrease. Quartering the resolution leads to a reduction of 95% in parameters. Regarding the GMacs, the reduction in input resolution has an even bigger effect; halving the resolution causes a 94% reduction in GMacs, and quartering the resolution leads to a network with 0.4% of the GMacs of the original. However, the next table shows that this also comes with a heavy penalty in prediction quality.

Table 4 highlights the effect of our compression techniques on the Dice score. We already discussed the difference between baseline and Step B with and without grouping with 224×224 input resolution. For 128×128 input resolution, we observe that all model variations lose around a 4% Dice score. Furthermore, decreasing the input resolution to 64×64, we lose around 14% in the Dice score. Moreover, Step B without grouping is, this time, even slightly worse than with grouping. Moving on with Step C, we are losing some Dice score percentage points by using a pruning factor of 50%. However, the degree of degradation depends on how much we compress the input resolution. For 224×244 and 128×128 input resolutions, all architectures lose around 1% Dice by applying pruning. We see a different reaction with the three architectures only at 64×64 input resolution. Step B without grouping has only around 1% loss, Step B with grouping loses 3.5% compared to not using pruning, and the baseline DAE-Former loses 10%, which is an unacceptable loss of real-world deployment. With Step D, the use of FP16 precision, we notice a very minimal change in Dice score overall, except DAE-Former with 64×64, where we lose 5% when using FP16. This

Table 5. Comparison of throughput on different devices for Embedded-Vit with DAE-Former baseline using all our compression methods.

Methods	GPU			CPU			Jetson Nano		
Resolution	224^2	128^2	64^2	224^2	128^2	64^2	224^2	128^2	64^2
DAE-F.	41.70	59.71	63.82	7.19	31.32	59.40	-	-	28.96
Step B gr.	68.78	101.77	111.56	11.26	51.01	97.59	-	5.26	45.49
Step B	29.76	90.35	96.52	3.65	38.32	89.65	-	-	20.98
DAE-F. 50%P.	-	-	-	7.06	23.36	208.42	-	-	28.96
Step B gr. 50%P.	-	-	-	4.72	139.52	587.88	-	6.57	55.99
Step B 50%P.	-	-	-	3.84	54.60	205.15	-	-	24.26
DAE-F. FP16	35.15	49.16	50.77	7.17	28.40	58.55	-	-	14.15
Step B gr. FP16	57.29	85.18	99.14	11.14	55.13	96.50	-	2.94	22.40
Step B FP16	67.14	108.77	117.63	3.80	38.88	94.89	-	-	10.06

stems from the network not being able to detect any instances of Gallbladder, which was a problem that only occurred with these specific parameters.

In Table 5, we compare the frames per second (FPS) of our different compression techniques on various platforms. Overall, we see the expected trend that when reducing the input resolution, we see a rise in FPS; on the GPU, the baseline gains 40% with half resolution and 50% with quartered resolution; on the CPU, the FPS even increases to more than 4-times with half resolution an more than 8-times with quartered resolution. The Jetson Nano can run the baseline only at 64×64 resolution but then at an impressive 29 FPS. Step B with grouping is on the GPU 65% faster than the baseline, without grouping 30% slower. However, we see that Step B with grouping gains 5% for each resolution compression and, more importantly, Step B without grouping is now 50% faster than the baseline and only 14% slower than with grouping, which shows that the denser model without grouping is a viable option for deployment after reducing the input resolution. On the CPU, Step B with grouping is still over 55% faster, but without grouping, it is almost 50% slower than the baseline at full input resolution. However, we observe again at 128×128 resolution that Step B without grouping is 7 FPS faster than the baseline. Moreover, at 64×64 resolution on the CPU, Step B with grouping almost catches up to Step B with grouping and gains over 20 FPS over the baseline. On Jetson Nano, we observe that Step B with grouping is the only model that can run with 128×128 input resolution. Furthermore, Step B with grouping increases the FPS compared to the baseline when both use 64×64 resolution by 50%. Step B, without grouping, can only run on the Jetson Nano with 64×64 resolution but at 20 FPS.

Moving on with Step C:

Step C does not affect the baseline's throughput rate at 224×224 input resolution on the CPU and compresses the network not enough to run the Jetson Nano. At 128×128 input resolution and on the CPU, we see that the baseline loses 8 FPS, while Step B with grouping almost triples the FPS, and Step B with grouping gains 16 FPS. On Jetson Nano, only Step B with grouping is compressed enough to run and reaches 6 FPS, one more FPS than without

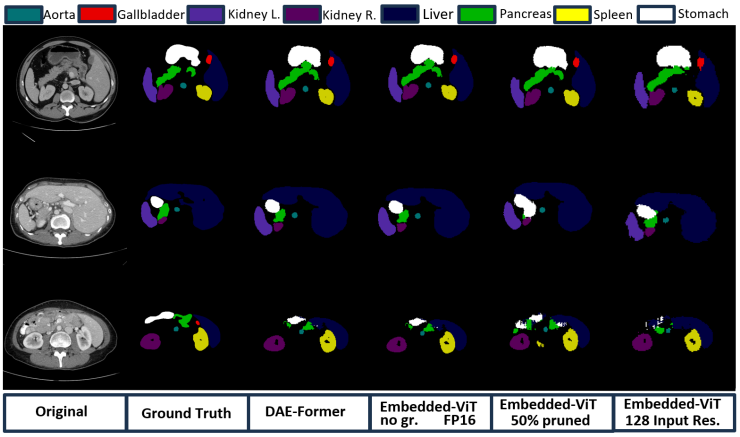

Fig. 5. A qualitative comparison of three example Synapse images. From left to right, showing the original synapse image, the ground truth, the baseline DAE-Former, our Embedded-ViT without grouping and FP16 precision, Embedded-ViT with grouping and 50% pruning, and the Embedded-ViT with 128×128 input resolution.

pruning. We now see huge gains for all networks at the smallest input resolution. The baseline almost quadruples its FPS, Step B with grouping reaches six times the FPS, and Step B without grouping still more than doubles its FPS. For the Jetson Nano, the baseline does not improve, but both Step B variations run faster than without pruning. Step B with grouping gains 10 FPS and without grouping 3 FPS. Unfortunately, the FP16 precision has no positive effect for most cases, except Step B, without grouping on the GPU. Here, on all resolutions, it is even the fastest network, which is surprising considering the extra number of GMacs and Parameters for Step B without grouping. All in all, Table 5 highlights that our proposed compression methods were necessary in order to allow embedded deployment of a state-of-the-art ViT model and allow us to deploy a model with 76.61 Dice score, compared to the 79.73 baseline which cannot deploy on such constraint platforms.

Fig. 5 shows our baseline results and variations of Embedded-ViT for a qualitative comparison. For comparison, we chose Embedded-ViT without and with grouping, combined with FP16 precision, pruning, and compressed input resolution, to showcase the effect of the different compression methods on the predictions. Furthermore, the last column shows the model configuration with the highest prediction quality and still able to run on Jetson Nano, which was the primary motivation of this work. Starting with comparing the different models results on the top image, we see that all models struggle with the correct prediction of the Pancreas but are otherwise all very close to the ground truth. Most notable is Embedded-ViT, which has reduced input resolution. This model generates the least accurate pancreas prediction, and we observe that the predicted shapes are not as smooth due to the input compression. When comparing the

middle image, we notice that this time, the Embedded-ViT with compressed input generates quite acceptable results compared to the baseline, and this time, Embedded-ViT with 50% pruned weights overpredicts the stomach to the detriment of the pancreas prediction. On the bottom image, we see that the baseline predictions and Embedded-ViT without grouping and FP16 precision have very similar predictions, and the pruned Embedded-ViT struggle again with stomach and pancreas predictions. Similarly, the Embedded-ViT with compressed input only predicts small artifacts for the stomach but handles the pancreas prediction better than the pruned version.

5 Conclusion

In this paper, we have proposed our Embedded-ViT framework, combining several state-of-the-art compression techniques to achieve a new efficient Vision Transformer. The compression techniques used in Embedded-ViT are all orthogonal to each other to achieve maximum efficiency at minimal prediction quality loss. The framework applies input image compression, efficient model architecture changes, pruning, and quantization. Our experiments highlighted that our compression methods are a very effective tool for reaching target throughputs at a minimal prediction quality loss. Our Embedded-ViT framework can be applied to any U-Net-shaped pure Transformer architecture and contribute to making it more cost-efficient. The Embedded-ViT framework is open-source to ensure reproducible research and accessibility. The source code is online at `https://BlindedLinkForReview`.

References

1. U. R. Acharya, S. L. Oh, Y. Hagiwara, J. H. Tan, and H. Adeli, "Deep convolutional neural network for the automated detection and diagnosis of seizure using eeg signals," *Computers in biology and medicine*, vol. 100, pp. 270–278, 2018.
2. Q. Lyu, C. You, H. Shan, and G. Wang, "Super-resolution mri through deep learning," *arXiv preprint arXiv:1810.06776*, 2018.
3. W. Shen, M. Zhou, F. Yang, C. Yang, and J. Tian, "Multi-scale convolutional neural networks for lung nodule classification," in *Information Processing in Medical Imaging: 24th International Conference, IPMI 2015, Sabhal Mor Ostaig, Isle of Skye, UK, June 28-July 3, 2015, Proceedings 24*, pp. 588–599, Springer, 2015.
4. B. D. De Vos, J. M. Wolterink, P. A. De Jong, M. A. Viergever, and I. Išgum, "2d image classification for 3d anatomy localization: employing deep convolutional neural networks," in *Medical imaging 2016: Image processing*, vol. 9784, pp. 517–523, SPIE, 2016.
5. L. Tan, W. Ma, J. Xia, and S. Sarker, "Multimodal magnetic resonance image brain tumor segmentation based on acu-net network," *IEEE Access*, vol. 9, pp. 14608–14618, 2021.
6. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.

7. Y. Deng, Y. Hou, J. Yan, and D. Zeng, "Elu-net: an efficient and lightweight u-net for medical image segmentation," *IEEE Access*, vol. 10, pp. 35932–35941, 2022.
8. M. Capra, B. Bussolino, A. Marchisio, G. Masera, M. Martina, and M. Shafique, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead," *IEEE Access*, vol. 8, pp. 225134–225180, 2020.
9. A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
10. 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," *arXiv preprint arXiv:2102.04306*, 2021.
11. M. Heidari, A. Kazerouni, M. Soltany, R. Azad, E. K. Aghdam, J. Cohen-Adad, and D. Merhof, "Hiformer: Hierarchical multi-scale representations using transformers for medical image segmentation," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 6202–6212, 2023.
12. R. Azad, M. Heidari, Y. Wu, and D. Merhof, "Contextual attention network: Transformer meets u-net," in *International Workshop on Machine Learning in Medical Imaging*, pp. 377–386, Springer, 2022.
13. M. Ding, B. Xiao, N. Codella, P. Luo, J. Wang, and L. Yuan, "Davit: Dual attention vision transformers," in *European conference on computer vision*, pp. 74–92, Springer, 2022.
14. Z. Shen, M. Zhang, H. Zhao, S. Yi, and H. Li, "Efficient attention: Attention with linear complexities," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3531–3539, 2021.
15. A. Ali, H. Touvron, M. Caron, P. Bojanowski, M. Douze, A. Joulin, I. Laptev, N. Neverova, G. Synnaeve, J. Verbeek, *et al.*, "Xcit: Cross-covariance image transformers," *Advances in neural information processing systems*, vol. 34, pp. 20014–20027, 2021.
16. H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang, "Swin-unet: Unet-like pure transformer for medical image segmentation," in *European conference on computer vision*, pp. 205–218, Springer, 2022.
17. A. Marchisio, D. Dura, M. Capra, M. Martina, G. Masera, and M. Shafique, "Swifttron: An efficient hardware accelerator for quantized transformers," in *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–9, IEEE, 2023.
18. R. Azad, R. Arimond, E. K. Aghdam, A. Kazerouni, and D. Merhof, "Dae-former: Dual attention-guided efficient transformer for medical image segmentation," in *International Workshop on PRedictive Intelligence In MEdicine*, pp. 83–95, Springer, 2023.
19. E. Iofinova, A. Peste, M. Kurtz, and D. Alistarh, "How well do sparse imagenet models transfer?," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12266–12276, 2022.
20. https://neuralmagic.com/ ; https://github.com/neuralmagic.
21. X. Huang, Z. Deng, D. Li, and X. Yuan, "Missformer: An effective medical image segmentation transformer," *arXiv preprint arXiv:2109.07162*, 2021.
22. G. Xu, X. Zhang, X. He, and X. Wu, "Levit-unet: Make faster encoders with transformer for medical image segmentation," in *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pp. 42–53, Springer, 2023.