Medical Image Segmentation Using Squeeze-and-Expansion Transformers

Shaohua Li[1*] , **Xiuchao Sui**[1] , **Xiangde Luo**[2] , **Xinxing Xu**[1] , **Yong Liu**[1] , **Rick Goh**[1]
[1]Institute of High Performance Computing, A*STAR, Singapore
[2]University of Electronic Science and Technology of China, Chengdu, China
{shaohua, xiuchao.sui}@gmail.com, xiangde.luo@std.uestc.edu.cn, {xuxinx, liuyong, gohsm}@ihpc.a-star.edu.sg

Abstract

Medical image segmentation is important for computer-aided diagnosis. Good segmentation demands the model to see the big picture and fine details simultaneously, i.e., to learn image features that incorporate large context while keep high spatial resolutions. To approach this goal, the most widely used methods – U-Net and variants, extract and fuse multi-scale features. However, the fused features still have small *effective receptive fields* with a focus on local image cues, limiting their performance. In this work, we propose Segtran, an alternative segmentation framework based on transformers, which have unlimited *effective receptive fields* even at high feature resolutions. The core of Segtran is a novel Squeeze-and-Expansion transformer: a squeezed attention block regularizes the self attention of transformers, and an expansion block learns diversified representations. Additionally, we propose a new positional encoding scheme for transformers, imposing a continuity inductive bias for images. Experiments were performed on 2D and 3D medical image segmentation tasks: optic disc/cup segmentation in fundus images (REFUGE'20 challenge), polyp segmentation in colonoscopy images, and brain tumor segmentation in MRI scans (BraTS'19 challenge). Compared with representative existing methods, Segtran consistently achieved the highest segmentation accuracy, and exhibited good cross-domain generalization capabilities. The source code of Segtran is released at https://github.com/askerlee/segtran.

1 Introduction

Automated Medical image segmentation, i.e., automated delineation of anatomical structures and other regions of interest (ROIs), is an important step in computer-aided diagnosis; for example it is used to quantify tissue volumes, extract key quantitative measurements, and localize pathology [Schlemper *et al.*, 2019; Orlando *et al.*, 2020]. Good segmentation demands the model to see the big picture and fine details at the same time, i.e., learn image features that incorporate large context while keep high spatial resolutions to output fine-grained segmentation masks. However, these two demands pose a dilemma for CNNs, as CNNs often incorporate larger context at the cost of reduced feature resolution. A good measure of how large a model "sees" is the *effective receptive field* (effective RF) [Luo *et al.*, 2016], i.e., the input areas which have non-negligible impacts to the model output.

Since the advent of U-Net [Ronneberger *et al.*, 2015], it has shown excellent performance across medical image segmentation tasks. A U-Net consists of an encoder and a decoder, in which the encoder progressively downsamples the features and generates coarse contextual features that focus on contextual patterns, and the decoder progressively upsamples the contextual features and fuses them with fine-grained local visual features. The integration of multiple scale features enlarges the RF of U-Net, accounting for its good performance. However, as the convolutional layers deepen, the impact from far-away pixels decay quickly. As a results, the effective RF of a U-Net is much smaller than its theoretical RF. As shown in Fig.2, the effective RFs of a standard U-Net and DeepLabV3+ are merely around 90 pixels. This implies that they make decisions mainly based on individual small patches, and have difficulty to model larger context. However, in many tasks, the heights/widths of the ROIs are greater than 200 pixels, far beyond their effective RFs. Without "seeing the bigger picture", U-Net and other models may be misled by local visual cues and make segmentation errors.

Many improvements of U-Net have been proposed. A few typical variants include: U-Net++ [Zhou *et al.*, 2018] and U-Net 3+ [Huang *et al.*, 2020], in which more complicated skip connections are added to better utilize multi-scale contextual features; attention U-Net [Schlemper *et al.*, 2019], which employs attention gates to focus on foreground regions; 3D U-Net [Çiçek *et al.*, 2016] and V-Net [Milletari *et al.*, 2016], which extend U-Net to 3D images, such as MRI volumes; Eff-UNet [Baheti *et al.*, 2020], which replaces the encoder of U-Net with a pretrained EfficientNet [Tan and Le, 2019].

Transformers [Vaswani *et al.*, 2017] are increasingly popular in computer vision tasks. A transformer calculates the pairwise interactions ("self-attention") between all input units, combines their features and generates contextualized features. The contextualization brought by a transformer is analogous to the upsampling path in a U-Net, except that

[*] Corresponding Author.

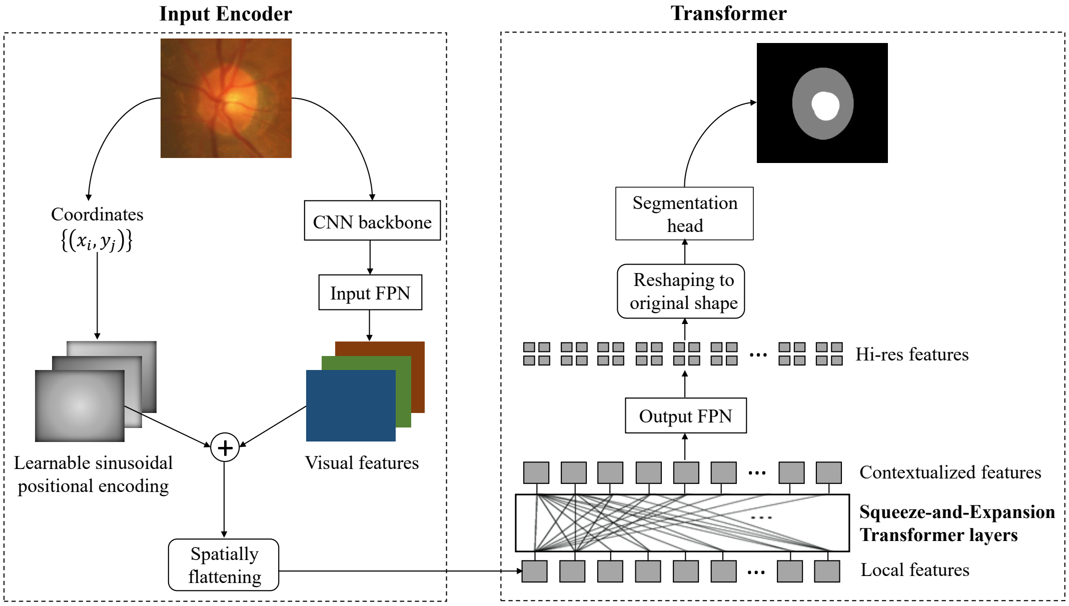
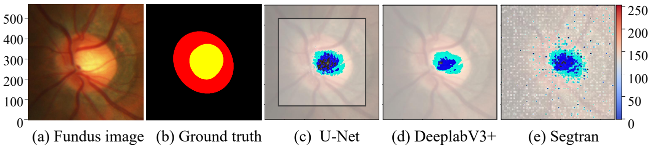


Figure 1: Segtran architecture. It extracts visual features with a CNN backbone, combines them with positional encodings of the pixel coordinates, and flattens them into a sequence of local feature vectors. The local features are contextualized by a few Squeeze-and-Expansion transformer layers. To increase spatial resolution, an input FPN and an output FPN upsamples the features before and after the transformers.



(a) Fundus image (b) Ground truth (c) U-Net (d) DeeplabV3+ (e) Segtran

Figure 2: Effective receptive fields of 3 models, indicated by non-negligible gradients in blue blobs and light-colored dots. Gradients are back-propagated from the center of the image. Segtran has non-negligible gradients dispersed across the whole image (light-colored dots). U-Net and DeepLabV3+ have concentrated gradients. Input image: 576 × 576.

it has unlimited effective receptive field, good at capturing long-range correlations. Thus, it is natural to adopt transformers for image segmentation. In this work, we present Segtran, an alternative segmentation architecture based on transformers. A straightforward incorporation of transformers into segmentation only yields moderate performance. As transformers were originally designed for Natural Language Processing (NLP) tasks, there are several aspects that could be improved to better suit image applications. To this end, we propose a novel transformer design *Squeeze-and-Expansion Transformer*, in which a squeezed attention block helps regularize the huge attention matrix, and an expansion block learns diversified representations. In addition, we propose a learnable sinusoidal positional encoding that imposes a continuity inductive bias for the transformer. Experiments demonstrate that they lead to improved segmentation performance.

We evaluated Segtran on two 2D medical image segmentation tasks: optic disc/cup segmentation in fundus images of the REFUGE'20 challenge, and polyp segmentation in colonoscopy images. Additionally, we also evaluated it on a 3D image segmentation task: brain tumor segmentation in MRI scans of the BraTS'19 challenge. Segtran has consistently shown better performance than U-Net and its variants (UNet++, UNet3+, PraNet, and nnU-Net), as well as DeepLabV3+ [Chen *et al.*, 2018].

2 Related Work

Our work is largely inspired by DETR [Carion *et al.*, 2020]. DETR uses transformer layers to generate contextualized features that represent objects, and learns a set of object queries to extract the positions and classes of objects in an image. Although DETR is also explored to do panoptic segmentation [Kirillov *et al.*, 2019], it adopts a two-stage approach which is not applicable to medical image segmentation. A followup work of DETR, Cell-DETR [Prangemeier *et al.*, 2020] also employs transformer for biomedical image segmentation, but its architecture is just a simplified DETR, lacking novel components like our Squeeze-and-Expansion transformer. Most recently, SETR [Zheng *et al.*, 2021] and TransU-Net [Chen *et al.*, 2021] were released concurrently or after our paper submission. Both of them employ a Vision Transformer (ViT) [Dosovitskiy *et al.*, 2021] as the encoder to extract image features, which already contain global contextual information. A few convolutional layers are used as the decoder to generate the segmentation mask. In contrast, in Segtran, the transformer layers build global context on top of the local image features extracted from a CNN backbone, and a Feature Pyramid Network (FPN) generates the segmentation mask.

[Murase *et al.*, 2020] extends CNNs with positional encoding channels, and evaluates them on segmentation tasks. Mixed results were observed. In contrast, we verified through an ablation study that positional encodings indeed help Segtran to do segmentation to a certain degree.

Receptive fields of U-Nets may be enlarged by adding more downsampling layers. However, this increases the number of parameters and adds the risk of overfitting. Another way of increasing receptive fields is using larger stride sizes of the convolutions in the downsampling path. Doing so, however, sacrifices spatial precision of feature maps, which is often disadvantageous for segmentation [Liu and Guo, 2020].

3 Squeeze-and-Expansion Transformer

The core concept in a transformer is *Self Attention*, which can be understood as computing an affinity matrix between different units, and using it to aggregate features:

$$\text{Att\_weight}(\boldsymbol{X},\boldsymbol{X}) = f(\boldsymbol{K}(\boldsymbol{X}),\boldsymbol{Q}(\boldsymbol{X})) \in \mathbb{R}^{N\times N}, \quad (1)$$

$$\text{Attention}(\boldsymbol{X}) = \text{Att\_weight}(\boldsymbol{X},\boldsymbol{X}) \cdot \boldsymbol{V}(\boldsymbol{X}), \quad (2)$$

$$\boldsymbol{X}_{out} = \text{FFN}(\text{Attention}(\boldsymbol{X})), \quad (3)$$

where $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{V}$ are key, query, and value projections, respectively. f is softmax after dot product. $\text{Att\_weight}(\boldsymbol{X},\boldsymbol{X})$ is the pairwise attention matrix between input units, whose i, j-th element defines how much the features of unit j contributes to the fused (contextualized) features of unit i. FFN is a feed-forward network to further transform the fused features.

The basic transformer above is extended to a *multi-head attention* (MHA) [Vaswani *et al.*, 2017; Voita *et al.*, 2019], aiming to capture different types of associations between input units. Each of the N_h heads computes individual attention wights and output features (C/N_h-dimensional), and their output features are *concatenated* along the channel dimension into C-dimensional features. Different heads operate in exclusively different feature subspaces.

We argue that transformers can be improved in four aspects make them better suited for images:

1. In Eq.(2), the intermediate features $\text{Attention}(\boldsymbol{X})$ are obtained by linearly combining the projected input features, where the attention matrix specifies the combination coefficients. As the attention matrix is huge: $N \times N$, with typically $N > 1000$, it is inherently vulnerable to noises and overfitting. Reducing the attention matrix to lower rank matrices may help.

2. In traditional transformers, the output features are *monomorphic*: it has only one set of feature transformations (the multi-head transformer also has *one set* of transformations after concatenation), which may not have enough capacity to fully model data variations. Just like a mixture of Gaussians almost always better depicts a data distribution than a single Gaussian, data variations can be better captured using a mixture of k transformers.

3. In traditional transformers, the key and query projections are independently learned, enabling them to capture *asymmetric* relationships between tokens in natural language. However, the relationships between image units are often *symmetric*, such as whether two pixels belong to the same segmentation class.

4. Pixels in an image have strong locality and semantic continuity. The two mainstream positional encoding schemes [Carion *et al.*, 2020; Dosovitskiy *et al.*, 2021] do not fully impose such an *inductive bias*. This bias could be imposed by an improved positional encoding.

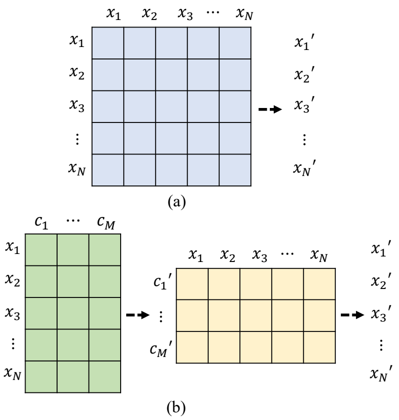


Figure 3: (a) Full self-attention ($N \times N$) vs. (b) Squeezed Attention Block (SAB). In SAB, first input units $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$ attend with a codebook $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_M$, yielding projected codebook features $\boldsymbol{c}'_1, \cdots, \boldsymbol{c}'_M$, which then attend back with the input $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N$. The two attention matrices are $N \times M$ and $M \times N$, respectively.

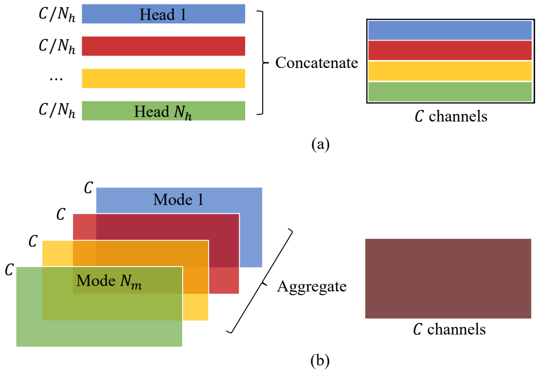


Figure 4: (a) Multi-head attention (MHA) vs. (b) Expanded attention block (EAB). In MHA, each head outputs an exclusive feature subset. In contrast, EAB outputs N_m sets of complete features from N_m modes, and aggregates them with dynamic mode attention.

The Squeeze-and-Expansion Transformer aims to improve in all the four aspects. The *Squeezed Attention Block* computes attention between the input and M inducing points [Lee *et al.*, 2019], and compresses the attention matrices to $N \times M$. The *Expanded Attention Block* is a mixture-of-experts model with N_m modes (“experts”). In both blocks, the query projections and key projections are *tied* to make the attention *symmetric*, for better modeling of the symmetric relationships between image units. In addition, a *Learnable Sinusoidal Positional Encoding* helps the model capture spatial relationships.

3.1 Squeezed Attention Block

[Lee *et al.*, 2019] proposes *Induced Set Attention Block* (ISAB) by bringing *inducing points* into the transformer. It was originally designed to learn good features of a set of unordered objects. Here we employ this design to “squeeze” the bloated attention matrix, so as to reduce noises and overfitting in image tasks. We rename ISAB as *Squeezed Attention Block* (SAB) to highlight its new role in this context[1].

In SAB, inducing points are a set of M learned embeddings $\boldsymbol{c}_1, \cdots, \boldsymbol{c}_M$ in an external *discrete codebook*. Usually $M \ll N$, the number of input units. The inducing points are first transformed into new embeddings $\boldsymbol{C}' = \boldsymbol{c}'_1, \cdots, \boldsymbol{c}'_M$ after attending with the input. The combination of these embeddings form the output features $\boldsymbol{X}_{out} = \boldsymbol{x}'_1, \cdots, \boldsymbol{x}'_N$ (Fig.3):

$$\boldsymbol{C}' = \text{Single-Head}(\boldsymbol{X}, \boldsymbol{C}), \tag{4}$$

$$\boldsymbol{X}_{out} = \text{EAB}(\boldsymbol{C}', \boldsymbol{X}), \tag{5}$$

where Single-Head$(\cdot,\cdot)$ is a single-head transformer, and EAB$(\cdot,\cdot)$ is an Expanded Attention Block presented in the next subsection. In each of the two steps, the attention matrix is of $N \times M$, much more compact than vanilla transformers.

SAB is conceptually similar to the codebook used for discrete representation learning in [Esser *et al.*, 2020], but the discretized features are further processed by a transformer. SAB can trace its lineage back to low-rank matrix factorization, i.e., approximating a data matrix $\boldsymbol{X}_{n\times n} \approx \boldsymbol{P}_{n\times d} \cdot \boldsymbol{Q}_{d\times n}$, which is a traditional regularization technique against data noises and overfitting. Confirmed by an ablation study, SAB helps fight against noises and overfitting as well.

3.2 Expanded Attention Block

The Expanded Attention Block (EAB) consists of N_m modes, each an individual single-head transformer. They output N_m sets of contextualized features, which are then aggregated into one set using dynamic mode attention:

$$\boldsymbol{X}^{(k)}_{out} = \text{Mode}^{(k)}(\boldsymbol{X}), \tag{6}$$

$$\boldsymbol{B}^{(k)} = \text{Linear}^{(k)}(\boldsymbol{X}^{(k)}_{out}), \tag{7}$$

$$\text{with } k \in \{1, \cdots, N_m\},$$

$$\boldsymbol{G} = \text{softmax}\left(\boldsymbol{B}^{(1)}, \cdots, \boldsymbol{B}^{(N_m)}\right), \tag{8}$$

$$\boldsymbol{X}_{out} = \left(\boldsymbol{X}^{(1)}_{out}, \cdots, \boldsymbol{X}^{(N_m)}_{out}\right) \cdot \boldsymbol{G}^\top, \tag{9}$$

where the mode attention $G \in \mathbb{R}^{N_u \times N_m}$ is obtained by doing a linear transformation of each mode features, and taking softmax over all the modes. Eq.(9) takes a weighted sum over the modes to get the final output features $\boldsymbol{X}_{out}$. This dynamic attention is inspired by the Split Attention of the ResNest model [Zhang *et al.*, 2020b].

EAB is a type of Mixture-of-Experts [Shazeer *et al.*, 2017], an effective way to increase model capacity. Although there is resemblance between multi-head attention (MHA) and EAB, they are essentially different, as shown in Fig.4. In MHA, each head resides in an exclusive feature subspace and provides unique features. In contrast, different modes in EAB share the same feature space, and the representation power largely remains after removing any single mode. The modes join forces to offer more capacity to model diverse data, as shown in an ablation study. In addition, EAB is also different from the Mixture of Softmaxes (MoS) transformer [Zhang *et al.*, 2020a]. Although MoS transformer also uses k sets of queries and keys, it shares one set of value transformation.

3.3 Learnable Sinusoidal Positional Encoding

A crucial inductive bias for images is the pixel locality and semantic continuity, which is naturally encoded by convolutional kernels. As the input to transformers is flattened into 1-D sequences, positional encoding (PE) is the only source to inject information about spatial relationships. On the one hand, this makes transformers flexible to model arbitrary shapes of input. On the other hand, the continuity bias of images is non-trivial to fully incorporate. This is a limitation of the two mainstream PE schemes: Fixed Sinusoidal Encoding and Discretely Learned Encoding [Carion *et al.*, 2020; Dosovitskiy *et al.*, 2021]. The former is spatially continuous but lacks adaptability, as the code is predefined. The latter learns a discrete code for each coordinate without enforcing spatial continuity.

We propose *Learnable Sinusoidal Positional Encoding*, aiming to bring in the continuity bias with adaptability. Given a pixel coordinate (x, y), our positional encoding vector $\boldsymbol{pos}(x, y)$ is a combination of sine and cosine functions of linear transformations of (x, y):

$$pos_i(x,y) = \begin{cases} \sin(a_i x + b_i y + c_i) & \text{if } i < C/2 \\ \cos(a_i x + b_i y + c_i) & \text{if } i \geq C/2, \end{cases} \tag{10}$$

where i indexes the elements in $\boldsymbol{pos}$, $\{a_i, b_i, c_i\}$ are learnable weights of a linear layer, and C is the dimensionality of image features. To make the PE behave consistently across different image sizes, we normalize (x, y) into $[0, 1]^2$. When the input image is 3D, Eq.(10) is trivially extended by using 3D coordinates (x, y, z).

The encoding in Eq.(10) changes smoothly with pixel coordinates, and thus nearby units receive similar positional encodings, pushing the attention weights between them towards larger values, which is the spirit of the continuity bias. The learnable weights and sinusoidal activation functions make the code both adaptable and nonlinear to model complex spatial relationships [Tancik *et al.*, 2020].

4 Segtran Architecture

As a context-dependent pixel-wise classification task, segmentation faces a conflict between larger context (lower resolution) and localization accuracy (higher resolution). Segtran partly circumvents this conflict by doing pairwise feature contextualization, without sacrificing spatial resolutions. There are five main components in Segtran (Fig.1): 1) a CNN backbone to extract image features, 2) input/output feature pyramids to do upsampling, 3) learnable sinusoidal positional encoding, 4) Squeeze-and-Expansion transformer layers to contextualize features, and 5) a segmentation head.

4.1 CNN Backbone

We employ a pretrained CNN backbone to extract features maps with rich semantics. Suppose the input image is $X_0 \in \mathbb{R}^{H_0 \times W_0 \times D_0}$, where for a 2D image, $D_0 = 1$ or 3 is the number of color channels. For a 3D image, $D_0 \gg 3$ is the

[1] We clarify that our contribution is a novel transformer architecture that combines SAB with an Expanded Attention Block.

number of slices in the depth dimension. For 2D and 3D images, the extracted features are $\text{CNN}(X_0) \in \mathbb{R}^{C\times H\times W}$, and $\text{CNN}(X_0) \in \mathbb{R}^{C\times H\times W\times D}$, respectively.

On 2D images, typically ResNet-101 or EfficientNet-D4 is used as the backbone. For increased spatial resolution, we change the stride of the first convolution from 2 to 1. Then $H, W = H_0/16, W_0/16$. On 3D images, 3D backbones like I3D [Carreira and Zisserman, 2017] could be adopted.

4.2 Transformer Layers

Before being fed into the transformer, the visual features and positional encodings of each unit are added up before being fed to the transformer: $\boldsymbol{X}_{\text{spatial}} = \boldsymbol{X}_{\text{visual}} + \boldsymbol{pos}(\text{coordinates}(\boldsymbol{X}))$. $\boldsymbol{X}_{\text{spatial}}$ is flattened across spatial dimensions to a 1-D sequence $\boldsymbol{X}_0 \in \mathbb{R}^{N_u\times C}$, where N_u is the total number of image units, i.e., points in the feature maps.

The transformer consists of a few stacked transformer layers. Each layer takes input features $\boldsymbol{X}$, computes the pairwise interactions between input units, and outputs contextualized features $\boldsymbol{X}_{out}$ of the same number of units. The transformer layers used are our novel design *Squeeze-and-Expansion Transformer* (Section 3).

4.3 Feature Pyramids and Segmentation Head

Although the spatial resolution of features is not reduced after passing through the transformer layers, for richer semantics, the input features to transformers are usually high-level features from the backbone. They are of a low spatial resolution, however. Hence, we increase their spatial resolution with an input Feature Pyramid Network (FPN) [Liu *et al.*, 2018] and an output FPN, which upsample the feature maps at the transformer input end and output end, respectively.

Without loss of generality, let us assume the EfficientNet is the backbone. The stages 3, 4, 6, and 9 of the network are commonly used to extract multi-scale feature maps. Let us denote the corresponding feature maps as $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3, \boldsymbol{f}_4$, respectively. Their shapes are $\boldsymbol{f}_i \in \mathbb{R}^{C_i\times H_i\times W_i}$, with $H_i = \frac{H_0}{2^i}, W_i = \frac{W_0}{2^i}$.

As described above, $\boldsymbol{f}(X_0) = \boldsymbol{f}_4$ is $1/16$ of the original image, which is too coarse for accurate segmentation. Hence, we upsample it with an **input FPN**, and obtain upsampled feature maps $\boldsymbol{f}_{34}$:

$$\boldsymbol{f}_{34} = \text{upsample}_{\times 2}(\boldsymbol{f}_4) + \text{conv}_{34}(\boldsymbol{f}_3), \tag{11}$$

where conv_{34} is a 1×1 convolution that aligns the channels of $\boldsymbol{f}_3$ to $\boldsymbol{f}_4$, and $\text{upsample}_{\times 2}(\cdot)$ is bilinear interpolation.

$\boldsymbol{f}_{34}$ is $1/8$ of the original image, and is used as the input features to the transformer layers. As the transformer layers keep the spatial resolutions unchanged from input to output feature maps, the output feature maps $\boldsymbol{g}_{34}$ is also $1/8$ of the input image. Still, this spatial resolution is too low for segmentation. Therefore, we adopt an **output FPN** to upsample the feature maps by a factor of 4 (i.e., $1/2$ of the original images). The output FPN consists of two upsampling steps:

$$\begin{aligned} \boldsymbol{f}_{12} &= \text{upsample}_{\times 2}(\boldsymbol{f}_2) + \text{conv}_{12}(\boldsymbol{f}_1), \\ \boldsymbol{g}_{1234} &= \text{upsample}_{\times 4}(\boldsymbol{g}_{34}) + \text{conv}_{24}(\boldsymbol{f}_{12}), \end{aligned} \tag{12}$$

where conv_{12} and conv_{24} are 1×1 convolutional layers that align the channels of $\boldsymbol{f}_1$ to $\boldsymbol{f}_2$, and $\boldsymbol{f}_2$ to $\boldsymbol{f}_4$, respectively.

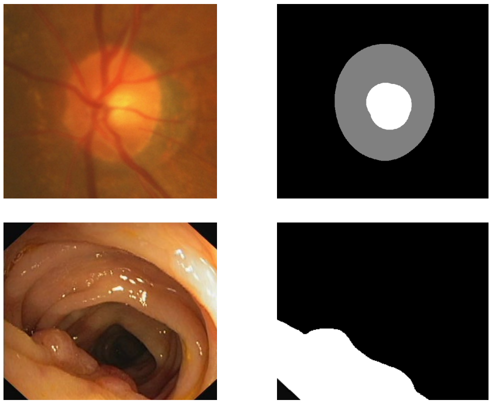


Figure 5: Top: Optic disc/cup segmentation in fundus images into 3 classes: disc (grey), cup (white), and background (black). Bottom: Polyp segmentation in colonoscopy images into 2 classes: polyp (white) and background (black).

This FPN scheme is the bottom-up FPN proposed in [Liu *et al.*, 2018]. Empirically, it performs better than the original top-down FPN [Lin *et al.*, 2017], as richer semantics in top layers are better preserved.

The segmentation head is simply a 1×1 convolutional layer, outputting confidence scores of each class in the mask.

5 Experiments

Three tasks were evaluated in our experiments:

REFUGE20: Optic Disc/Cup Segmentation in Fundus Images. This task does segmentation of the optic disc and cup in fundus images, which are 2D images of the rear of eyes (Fig. 5). It is a subtask of the REFUGE Challenge[2] [Orlando *et al.*, 2020], MICCAI 2020. 1200 images were provided for training, and 400 for validation. We also used two extra datasets, Drishti-GS dataset [Sivaswamy *et al.*, 2015] and RIM-ONE v3 [Fumero *et al.*, 2011] when training all models. The **Disc/Cup** dice scores of validation images were obtained from the official evaluation server.

Polyp: Polyp Segmentation in Colonoscopy Images. Polyps are fleshy growths in the colon lining that may become cancerous. This task does polyp segmentation in colonoscopy images (Fig. 5). Two image datasets [Fan *et al.*, 2020] were used: CVC612 (**CVC** in short; 612 images) and **Kvasir** (1000 images). Each was randomly split into 80% training and 20% validation, and the training images were merged into one set.

BraTS19: Tumor Segmentation in MRI Images. This task focuses on the segmentation of gliomas, a common brain tumor in MRI scans. It was a subtask of the BraTS'19 challenge[3] [Menze *et al.*, 2015; Bakas *et al.*, 2017], MICCAI 2019. It involves four classes: the whole tumor (**WT**), the tumor core (**TC**), the enhancing tumor (**ET**) and background. Among them, the tumor core consists of the necrotic regions and non-enhancing tumors (red), as well as the enhancing tumor (yellow). 335 scans were provided for training, and 125

[2] https://refuge.grand-challenge.org/Home2020/

[3] https://www.med.upenn.edu/cbica/brats-2019/

for validation. The dice scores of ET, WT and TC on the validation scans were obtained from the official evaluation server.

5.1 Ablation Studies

Two ablation studies were performed on REFUGE20 to compare: 1) the Squeeze-and-Expansion Transformer versus Multi-Head Transformer; and 2) the Learnable Sinusoidal Positional Encoding versus two schemes as well as not using PE.

All the settings were variants of the standard one, which used three layers of Squeeze-and-Expansion transformer with four modes ($N_m = 4$), along with learnable sinusoidal positional encoding. Both ResNet-101 and EfficientNet-B4 were evaluated to reduce random effects from choices of the backbone. We only reported the cup dice scores, as the disc segmentation task was relatively easy, with dice scores only varying ± 0.005 across most settings.

Type of Transformer Layers. Table 1 shows that Squeeze-and-Expansion transformer outperformed the traditional multi-head transformers. Moreover, Both the squeeze attention block and the expansion attention block contributed to improved performance.

Transformer Type	ResNet-101	Eff-B4
Cell-DETR ($N_h = 4$)	0.846	0.857
Multi-Head ($N_h = 4$)	0.858	0.862
No squeeze + Expansion ($N_m = 4$)	0.840	**0.872**
Squeeze + Single-Mode	0.859	0.868
Squeeze + Expansion ($N_m = 4$)	**0.862**	**0.872**

Table 1: REFUGE'20 Fundus Optic Cup dice scores change with the type of transformer layers. Single-Mode implies No Expansion. Cell-DETR uses a multi-head transformer and discretely learned PE. N_h: number of attention heads in a MHA. N_m: number of modes in a Squeeze-and-Expansion transformer.

Positional Encoding. Table 2 compares learnable sinusoidal positional encoding with the two mainstream PE schemes and no PE. Surprisingly, without PE, performance of Segtran only dropped 1~2%. A possible explanation is that the transformer may manage to extract positional information from the CNN backbone features [Islam *et al.*, 2020].

Positional Encoding	ResNet-101	Eff-B4
None	0.857	0.853
Discretely learned	0.852	0.860
Fixed Sinusoidal	0.857	0.849
Learnable Sinusoidal	**0.862**	**0.872**

Table 2: REFUGE'20 Fundus Optic Cup dice scores change with the type of positional encoding (PE) schemes.

Number of Transformer Layers. Table 3 shows that as the number of transformer layers increased from 1 to 3, the performance improved gradually. However, one more layer caused performance drop, indicating possible overfitting.

5.2 Comparison with Baselines

Ten methods were evaluated on the 2D segmentation tasks:

Number of layers	ResNet101	Eff-B4
1	0.856	0.854
2	**0.862**	0.857
3	**0.862**	**0.872**
4	0.855	0.869

Table 3: REFUGE20 Optic Cup dice scores change with the number of transformer layers. Best performers with each backbone are highlighted.

- **U-Net** [Ronneberger *et al.*, 2015]: The implementation in a popular library *Segmentation Models.PyTorch* (SMP) was used[4]. The pretrained ResNet-101 was chosen as the encoder. In addition, U-Net implemented in U-Net++ (below) was evaluated as training from scratch.
- **U-Net++** [Zhou *et al.*, 2018]: A popular PyTorch implementation[5]. It does not provide options to use pretrained encoders, and thus was only trained from scratch.
- **U-Net3+** [Huang *et al.*, 2020]: The official PyTorch implementation[6]. It does not provide options to use pretrained encoders.
- **PraNet** [Fan *et al.*, 2020]: The official PyTorch implementation[7]. The pretrained Res2Net-50 [Gao *et al.*, 2020] was recommended to be used as the encoder.
- **DeepLabV3+** [Chen *et al.*, 2018]: A popular PyTorch implementation[8], with a pretrained ResNet-101 as the encoder.
- **Attention based U-Nets** [Oktay *et al.*, 2018]: Attention U-Net (**AttU-Net**) and **AttR2U-Net** (a combination of AttU-Net and Recurrent Residual U-Net) were evaluated[9]. They learn to focus on important areas by computing element-wise attention weights (as opposed to the pairwise attention of transformers).
- **nnU-Net** [Isensee *et al.*, 2021]: nnU-Net generates a custom U-Net configuration for each dataset based on its statistics. It is primarily designed for 3D tasks, but can also handle 2D images after converting them to pseudo-3D. The original pipeline is time-consuming, and we extracted the generated U-Net configuration and instantiated it in our pipeline to do training and test.
- **Deformable U-Net** [Jin *et al.*, 2019]: Deformable U-Net (**DUNet**) uses deformable convolution in place of ordinary convolution. The official implementation [10] of DUNet was evaluated.
- **SETR** [Zheng *et al.*, 2021]: SETR uses ViT as the encoder, and a few convolutional layers as the decoder. The SETR-PUP model in the official implementation[11] was evaluated, by fine-tuning the pretrained ViT

[4] https://github.com/qubvel/segmentation_models.pytorch/

[5] https://github.com/4uiiurz1/pytorch-nested-unet

[6] https://github.com/ZJUGiveLab/UNet-Version

[7] https://github.com/DengPingFan/PraNet

[8] hhttps://github.com/VainF/DeepLabV3Plus-Pytorch

[9] https://github.com/LeeJunHyun/Image_Segmentation

[10] https://github.com/RanSuLab/DUNet-retinal-vessel-detection

[11] https://github.com/fudan-zvg/SETR/

	REFUGE20		Polyp		Avg.
	Cup	Disc	Kvasir	CVC	
U-Net	0.730	0.946	0.787	0.771	0.809
U-Net (R101)	0.837	0.950	0.868	0.844	0.875
U-Net++	0.781	0.940	0.753	0.740	0.804
U-Net3+	0.819	0.943	0.708	0.680	0.788
PraNet (res2net50)	0.781	0.946	0.898	0.899	0.881
DeepLabV3+ (R101)	0.839	0.950	0.805	0.795	0.847
AttU-Net	0.846	0.952	0.744	0.749	0.823
AttR2U-Net	0.818	0.944	0.686	0.632	0.770
DUNet	0.826	0.945	0.748	0.754	0.818
nnU-Net	0.829	0.953	0.857	0.864	0.876
SETR (ViT)	0.859	0.952	0.894	0.916	0.905
TransU-Net (R50+ViT)	0.835	0.958	0.895	0.916	0.901
Segtran (R101)	0.862	0.956	0.888	0.929	0.909
Segtran (eff-B4)	**0.872**	**0.961**	**0.903**	**0.931**	**0.917**

Table 4: Dice scores on REFUGE20 and Polyp validation sets. R101: ResNet-101; R50: ResNet-50; eff-B4: EfficientNet-B4.

weights.

- **TransU-Net** [Chen *et al.*, 2021]: TransU-Net uses a hybrid of ResNet and ViT as the encoder, and a U-Net style decoder. The official implementation[12] was evaluated, by fine-tuning their pretrained weights.
- **Segtran**: Trained with either a pretrained ResNet-101 or EfficientNet-B4 as the backbone.

Three methods were evaluated on the 3D segmentation task:

- **Extension of nnU-Net** [Wang *et al.*, 2019]: An extension of the nnU-Net[13] with two sampling strategies.
- **Bag of tricks (2nd place solution of the BraTS'19 challenge)** [Zhao *et al.*, 2019]: The winning entry used an ensemble of five models. For fairness, we quoted the best single-model results ("BL+warmup").
- **Segtran-3D**: I3D [Carreira and Zisserman, 2017] was used as the backbone.

5.3 Training Protocols

All models were trained on a 24GB Titan RTX GPU with the AdamW optimizer. The learning rate for the three transformer-based models were 0.0002, and 0.001 for the other models. On REFUGE20, all models were trained with a batch size of 4 for 10,000 iterations (27 epochs); on Polyp, the total iterations were 14,000 (31 epochs). On BraTS19, Segtran was trained with a batch size of 4 for 8000 iterations.

The training loss was the average of the pixel-wise cross-entropy loss and the dice loss. Segtran used 3 transformer layers on 2D images, and 1 layer on 3D images to save RAM. The number of modes in each transformer layer was 4.

5.4 Results

	BraTS19			
	ET	WT	TC	Avg.
Extension of nnU-Net	0.737	0.894	0.807	0.813
Bag of tricks	0.729	**0.904**	0.802	0.812
Segtran (i3d)	**0.740**	0.895	**0.817**	**0.817**

Table 5: Dice scores on BraTS19 validation set. Only single-model performance is reported.

Tables 4 and 5 present the evaluation results on the 2D and 3D tasks, respectively. Overall, the three transformer based methods, i.e., SETR, TransU-Net and Segtran achieved best performance across all tasks. With ResNet-101 as the backbone, Segtran performed slightly better than SETR and TransU-Net. With EfficientNet-B4, Segtran exhibited greater advantages.

It is worth noting that, Segtran (eff-B4) was among the top 5 teams in the semifinal and final leaderboards of the REFUGE20 challenge. Among either REFUGE20 or BraTS19 challenge participants, although there were several methods that performed slightly better than Segtran, they usually employed ad-hoc tricks and designs [Orlando *et al.*, 2020; Wang *et al.*, 2019; Zhao *et al.*, 2019]. In contrast, Segtran achieved competitive performance with the same architecture and minimal hyperparameter tuning, free of domain-specific strategies.

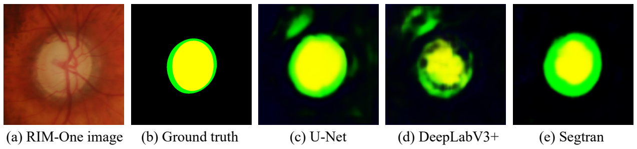


Figure 6: Soft segmentation masks produced by different methods on a RIM-One image. The mask by Segtran has the fewest artifacts.

5.5 Cross-Domain Generalization

To explore how well different methods generalize to new domains, we trained three representative methods, U-Net, DeepLabV3+ and Segtran on the 1200 training images of REFUGE20. All the methods used a pretrained ResNet-101 as the encoder/backbone. The trained models were evaluated on both the REFUGE20 training images and the RIM-One dataset [Fumero *et al.*, 2011]. As RIM-One images have drastically different characteristics from REFUGE20, all models suffered severe performance drop, as shown in Table 6. Nevertheless, Segtran had the least performance degradation, showing the best cross-domain generalization. Fig.6 shows a RIM-One image and the corresponding soft segmentation masks (before thresholding) produced by different methods. The mask produced by Segtran contains the fewest artifacts.

5.6 Computational Efficiency

Table 7 presents the number of parameters and FLOPs of a few representative methods. In general, transformer-based methods consume more computation and GPU RAM than conventional methods.

Our profiling showed that the number of parameters/FLOPs of Segtran are dominated by the output FPN, which vary drastically across different backbones. As the

[12] https://github.com/Beckschen/TransUNet

[13] https://github.com/woodywff/brats_2019

	REFUGE	RIM-One	Drop
U-Net	0.862	0.680	-0.182
DeepLabV3+	0.846	0.653	-0.193
Segtran	**0.938**	**0.796**	**-0.142**

Table 6: Generalization of three methods, measured by drop of Optic Cup dice scores from the REFUGE20 training images to a new test domain RIM-One. The smaller the drop is, the better. All used ResNet-101 as the encoder/backbone.

bottom-up FPNs we adopt are somewhat similar to EfficientDet [Tan *et al.*, 2020], the model size/FLOPs are optimal when using EfficientNets. With ResNets as the backbone, Segtran has a significantly higher model size/FLOPs, and hence this choice of backbone is not recommended for efficiency-sensitive scenarios.

	Params (M)	FLOPs (G)
nnU-Net	41.2	16.3
AttU-Net	34.9	51.0
SETR (ViT)	307.1	91.1
TransU-Net (R50+ViT)	93.2	32.2
Segtran (R101)	166.7	152.8
Segtran (eff-B4)	93.1	71.3

Table 7: Number of parameters / FLOPs on a 256x256 input image.

5.7 Impact of Pretraining

Models for medical image tasks usually benefit from initialization with weights pretrained on natural images (e.g. ImageNet [Deng *et al.*, 2009]), as medical image datasets are typically small. To quantitatively study the impact of pretraining, Table 8 compares the performance of using pretrained weights vs. training from scratch of a few methods. Pretraining brought ~2.5% increase of average dice scores to the two transformer-based models, and 1% to U-Net (ResNet-101).

	REFUGE20		Polyp		Avg.
	Cup	Disc	Kvasir	CVC	
U-Net (R101 scratch)	0.827	0.953	0.847	0.835	0.865
U-Net (R101 pretrain)	0.837	0.950	0.868	0.844	0.875
TransU-Net (R50+ViT scratch)	0.817	0.943	0.869	0.872	0.875
TransU-Net (R50+ViT pretrained)	0.835	**0.958**	**0.895**	0.916	0.901
Segtran (R101 scratch)	0.852	0.939	0.858	0.851	0.875
Segtran (R101 pretrain)	**0.862**	0.956	0.888	**0.929**	**0.909**

Table 8: Impact of using pretrained encoder weights.

6 Conclusions

In this work, we present Segtran, a transformer-based medical image segmentation framework. It leverages unlimited receptive fields of transformers to contextualize features. Moreover, the transformer is an improved Squeeze-and-Expansion transformer that better fits image tasks. Segtran sees both the global picture and fine details, lending itself good segmentation performance. On two 2D and one 3D medical image segmentation tasks, Segtran consistently outperformed existing methods, and generalizes well to new domains.

Acknowledgements

We are grateful for the help and support of Wei Jing. This research is supported by A*STAR under its Career Development Award (Grant No. C210112016), and its Human-Robot Collaborative Al for Advanced Manufacturing and Engineering (AME) programme (Grant No. A18A2b0046).

References

[Baheti *et al.*, 2020] B. Baheti, S. Innani, S. Gajre, and S. Talbar. Eff-unet: A novel architecture for semantic segmentation in unstructured environment. In *CVPR Workshops*, 2020.

[Bakas *et al.*, 2017] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos. Advancing The Cancer Genome Atlas glioma MRI collections with expert segmentation labels and radiomic features. *Nature Scientific Data*, 4, 2017.

[Carion *et al.*, 2020] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020.

[Carreira and Zisserman, 2017] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[Chen *et al.*, 2018] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[Chen *et al.*, 2021] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[Çiçek *et al.*, 2016] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *MICCAI*, 2016.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[Esser *et al.*, 2020] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. *arxiv:2012.09841*, 2020.

[Fan *et al.*, 2020] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. Pranet: Parallel reverse attention network for polyp segmentation. In *MICCAI*, 2020.

[Fumero *et al.*, 2011] F. Fumero, S. Alayon, J. L. Sanchez, J. Sigut, and M. Gonzalez-Hernandez. Rim-one: An open retinal image database for optic nerve evaluation. In *24th International Symposium on CBMS*, 2011.

[Gao *et al.*, 2020] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE TPAMI*, 43, 2020.

[Huang *et al.*, 2020] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y. Chen, and J. Wu. Unet 3+: A full-scale connected unet for medical image segmentation. In *ICASSP*, 2020.

[Isensee *et al.*, 2021] Fabian Isensee, Paul F. Jaeger, Simon A. A. Kohl, Jens Petersen, and Klaus H. Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature Methods*, 18, 2021.

[Islam *et al.*, 2020] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? In *ICLR*, 2020.

[Jin *et al.*, 2019] Qiangguo Jin, Zhaopeng Meng, Tuan D. Pham, Qi Chen, Leyi Wei, and Ran Su. Dunet: A deformable network for retinal vessel segmentation. *Knowledge-Based Systems*, 2019.

[Kirillov *et al.*, 2019] Alexander Kirillov, Kaiming He, Ross B. Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019.

[Lee *et al.*, 2019] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, 2019.

[Lin *et al.*, 2017] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, July 2017.

[Liu and Guo, 2020] Sun'ao Liu and Xiaonan Guo. Improving brain tumor segmentation with multi-direction fusion and fine class prediction. In *BrainLes workshop, MICCAI*, 2020.

[Liu *et al.*, 2018] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.

[Luo *et al.*, 2016] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, 2016.

[Menze *et al.*, 2015] Bjoern H. Menze, Andras Jakab, Stefan Bauer, Jayashree Kalpathy-Cramer, Keyvan Farahani, Justin Kirby, et al. The multimodal brain tumor image segmentation benchmark (brats). *IEEE TMI*, 34, 2015.

[Milletari *et al.*, 2016] F. Milletari, N. Navab, and S. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[Murase *et al.*, 2020] Rito Murase, Masanori Suganuma, and Takayuki Okatani. How Can CNNs Use Image Position for Segmentation? *arXiv:2005.03463*, 2020.

[Oktay *et al.*, 2018] Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, et al. Attention u-net: Learning where to look for the pancreas. In *MIDL*, 2018.

[Orlando *et al.*, 2020] José Ignacio Orlando, Huazhu Fu, João Barbosa Breda, Karel van Keer, Deepti R. Bathula, Andrés Diaz-Pinto, et al. Refuge challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical Image Analysis*, 59, 2020.

[Prangemeier *et al.*, 2020] Tim Prangemeier, Christoph Reich, and Heinz Koeppl. Attention-based transformers for instance segmentation of cells in microstructures. In *IEEE International Conference on Bioinformatics and Biomedicine*, 2020.

[Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[Schlemper *et al.*, 2019] Jo Schlemper, Ozan Oktay, Michiel Schaap, Mattias Heinrich, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention gated networks: Learning to leverage salient regions in medical images. *Medical Image Analysis*, 53, 2019.

[Shazeer *et al.*, 2017] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[Sivaswamy *et al.*, 2015] Jayanthi Sivaswamy, Subbaiah Krishnadas, Arunava Chakravarty, Gopal Joshi, and Ujjwal. A comprehensive retinal image dataset for the assessment of glaucoma from the optic nerve head analysis. *JSM Biomedical Imaging Data Papers*, 2, 2015.

[Tan and Le, 2019] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019.

[Tan *et al.*, 2020] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *CVPR*, June 2020.

[Tancik *et al.*, 2020] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*, 2020.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[Voita *et al.*, 2019] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*, 2019.

[Wang *et al.*, 2019] Feifan Wang, Runzhou Jiang, Liqin Zheng, Chun Meng, and Bharat Biswal. 3d u-net based brain tumor segmentation and survival days prediction. In *BrainLes Workshop, MICCAI*, 2019.

[Zhang *et al.*, 2020a] Dong Zhang, Hanwang Zhang, Jinhui Tang, Meng Wang, Xiansheng Hua, and Qianru Sun. Feature pyramid transformer. In *ECCV*, 2020.

[Zhang *et al.*, 2020b] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *arXiv:2004.08955*, 2020.

[Zhao *et al.*, 2019] Yuan-Xing Zhao, Yan-Ming Zhang, and Cheng-Lin Liu. Bag of tricks for 3d mri brain tumor segmentation. In *Brainles Workshop, MICCAI*, 2019.

[Zheng *et al.*, 2021] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.

[Zhou *et al.*, 2018] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *DLMIA workshop (MICCAI)*, 2018.