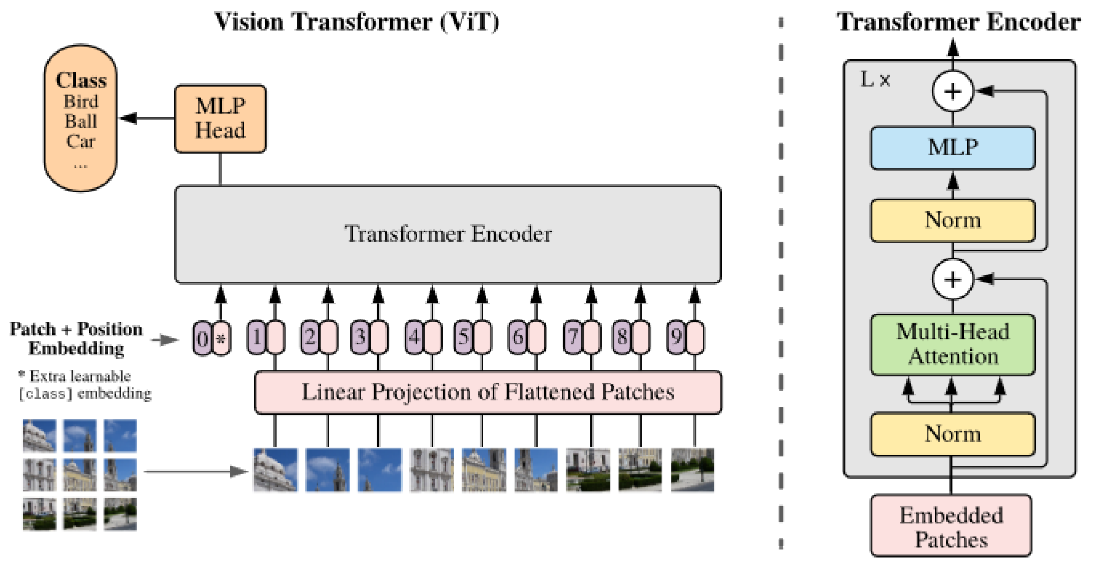
- Basics of Vision Transformer
- Explainability
  - Direct Visualization
  - Attention Rollout
  - Attention Gradient Rollout (TIBA)
- DEIT

- TRANSUNET
- SEGMENTER

VISION TRANSFORMER



Important Features

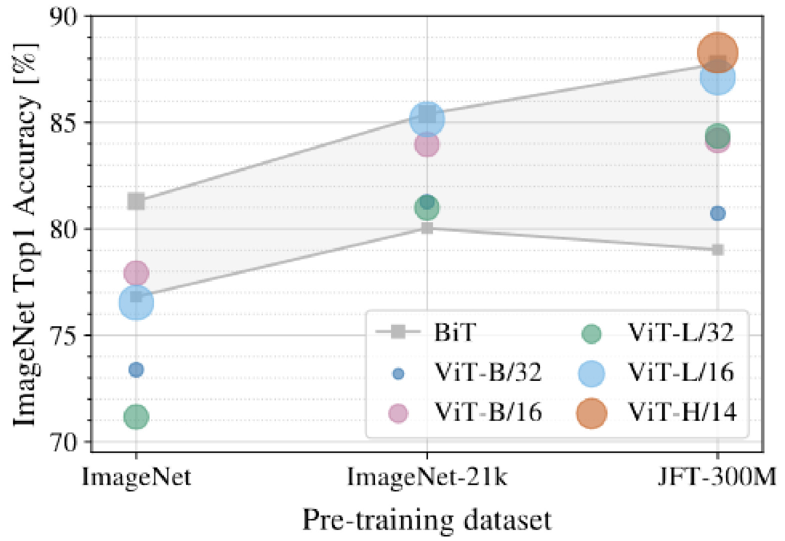
- The image is divided into patches. Each patch is flattened and fed through a linear projection to get a token. The set of tokens is then fed into the Transformer Encoder.
- [CLS] Token is prepended to the set of patches. It helps in capturing global features.
- Positional Embedding is added to each patch to bring some sense of position.
- [CLS] Token and position embeddings are learnable
- Output from only the [CLS] token is taken and fed to a MLP Classification Head

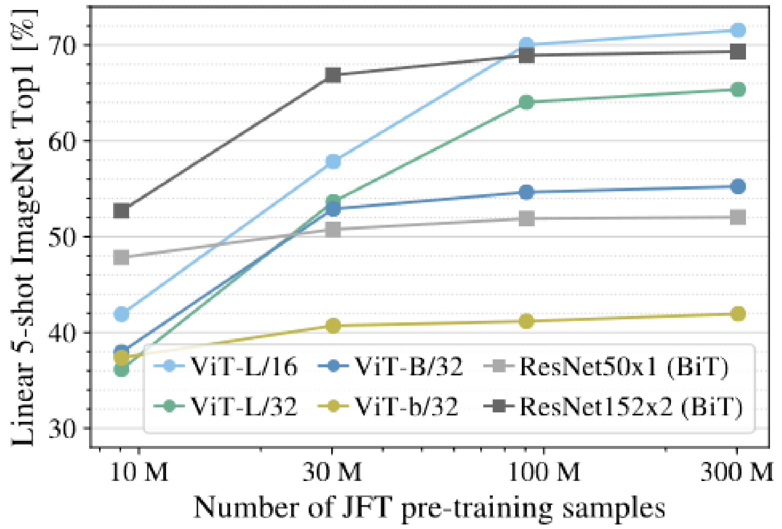
No inductive bias is present like in CNNs
Requires large amount of training data. Surpassed CNNs on imageNet classification like BiTs (ResNets pre-trained on ImageNet21k) when pre-trained on large training sets like ImageNet21k or JFT-300M

Alexey et.al **An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale ICLR 2021**

- **My implementation of Vision Transformer** https://colab.research.google.com/drive/1yuCld4ykAnpAI98GRzkWi30lENcEDE6g?usp=sharing
- I trained it on MNIST Dataset. Training time is very slow per epochs. After about 10 epochs, 70% accuracy was obtained on test set. Compared to CNNs, this performance is much worse. Its maybe because of smaller dataset.

Performance on ImageNet





Vit is better than Bit when pretrained on JFT-300M

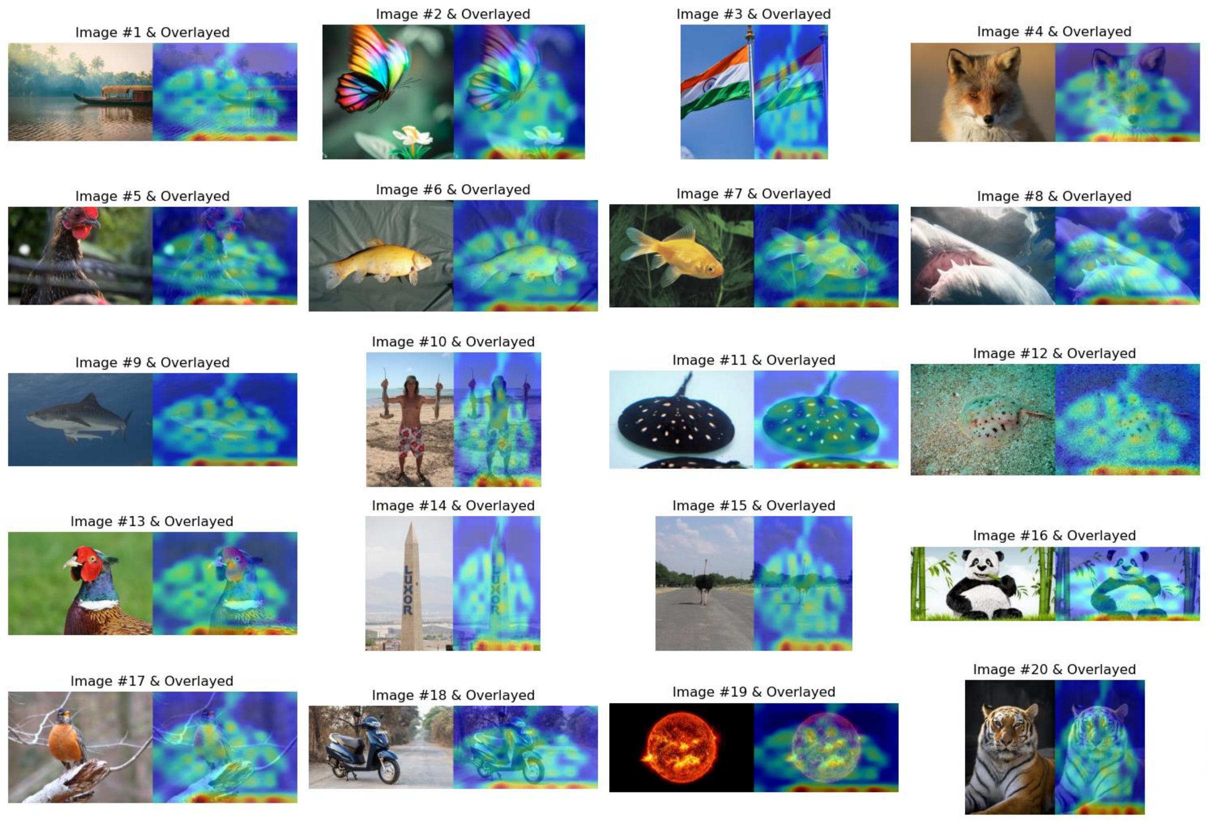
Visualising Attentions

- I did some experiments to visualise the attention matrix
- Considering attention values of [CLS] Tokens. Then, mapping patches back to the image and overlaying attention values
- Averaging is done across all the heads

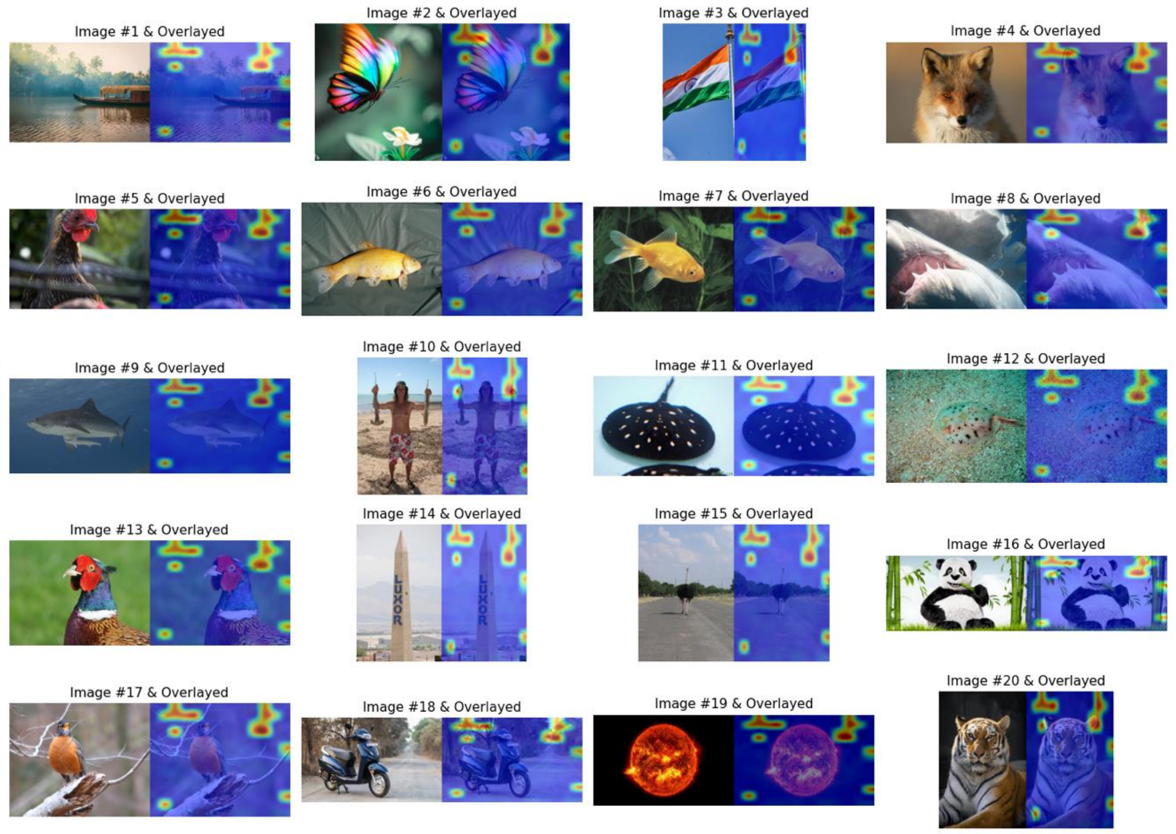
[CLS] Row (captures global features)

	[CLS]	T1	T2	T3	T4
[CLS]		🟩	🟩	🟩	🟩
T1					
T2					
T3					
T4					

Heat Maps at Layer 1



Heat Maps at Layer 12



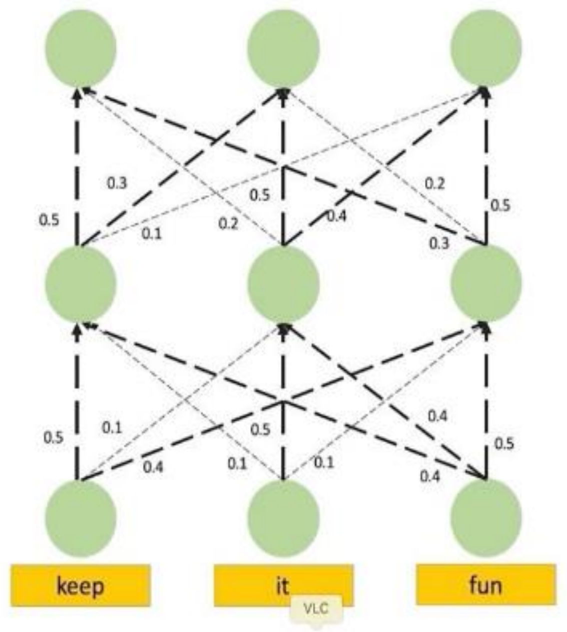
Interestingly, the heat maps at first encoder layer make much more sense when compared to the final layer
Overall representation is quite bad

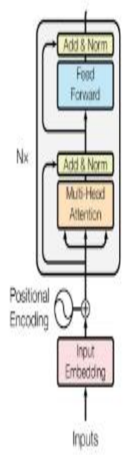
ATTENTION ROLLOUT

$$\hat{\mathbf{A}}^{(b)} = I + \mathbb{E}_h \mathbf{A}^{(b)}$$

$$\text{rollout} = \hat{\mathbf{A}}^{(1)} \cdot \hat{\mathbf{A}}^{(2)} \cdot \ldots \cdot \hat{\mathbf{A}}^{(B)}$$

- Aggregation across layers : matrix multiplication across layer attention maps of all layers are multiplied
- Identity matrix is added to account for residual connections
- Aggregation across heads : mean (in paper). I have tried visualising max of the heads (from this blog)

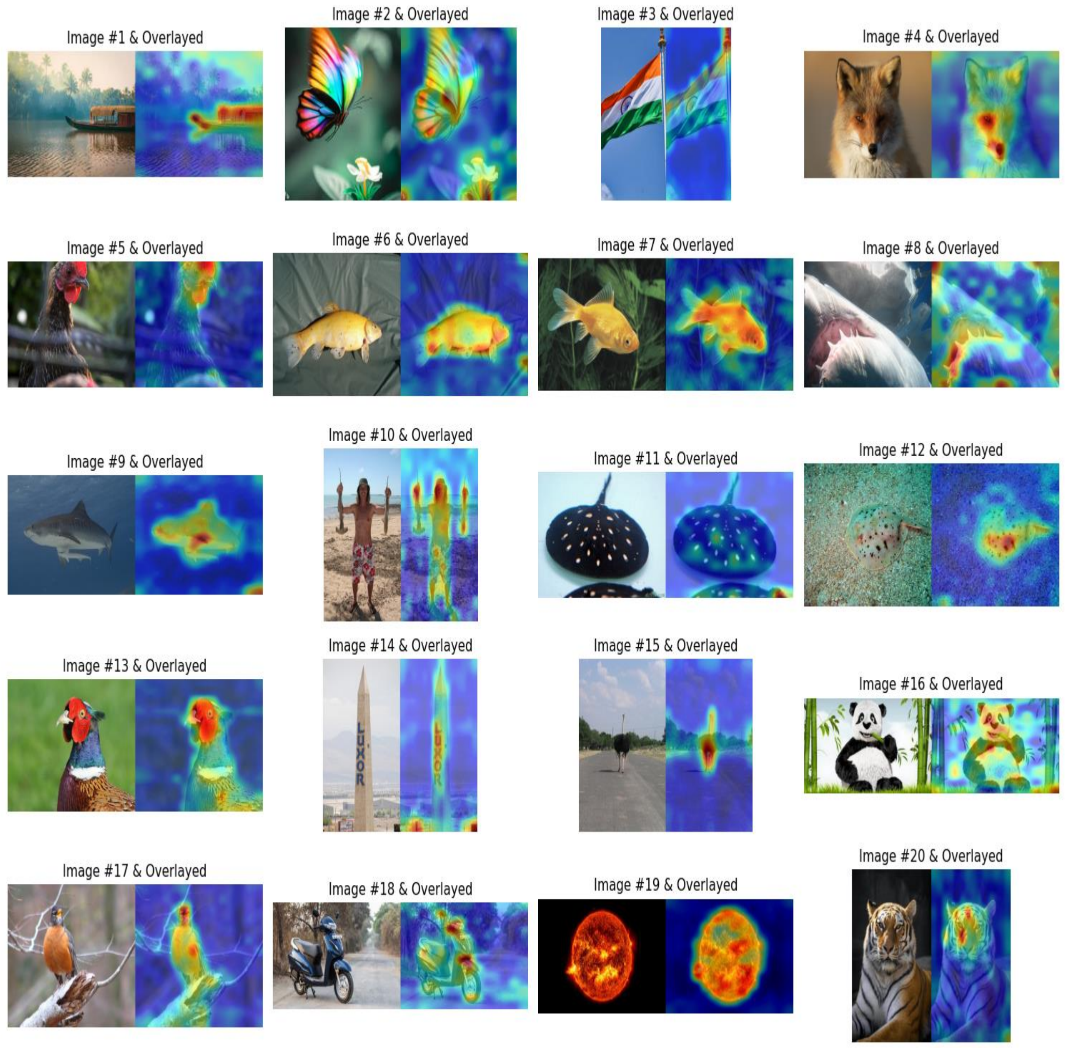




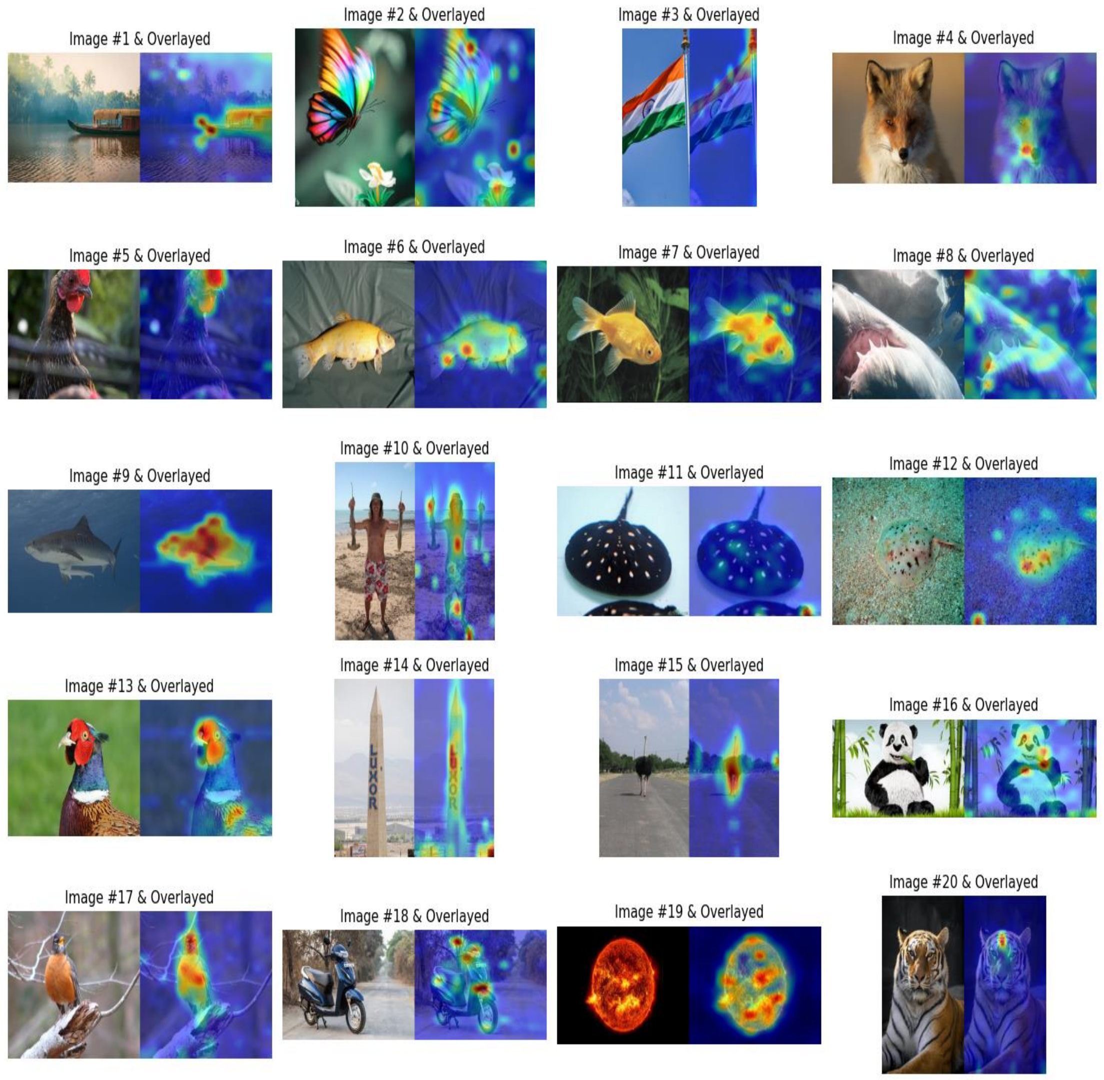
$$A = 0.5W_{att} + 0.5I,$$

Quantifying Attention Flow in Transformers- Abnar et al
https://jacobgil.github.io/deeplearning/vision-transformer-explainability

Mean of Attention Heads



Maximum among Attention Heads



Attention Rollout – Limitations

- Averaging across the attention heads is too simplistic
- Each head has its own uses, some heads may not be as relevant for the task at hand.
- Ignorance of other parts of the Network – MLP, Activations

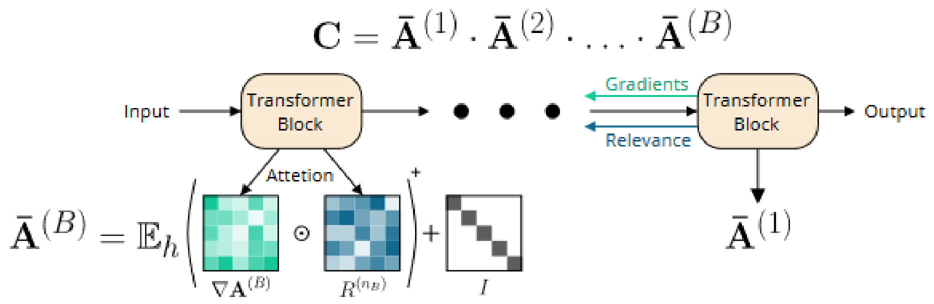
Gradient Rollout and LRP (TIBA)

- Use gradients (w.r.t output logit) as weights across different heads instead of averaging
- Use Relevance Values obtained through Layerwise Relevance Propagation(LRP) instead of attentions
- LRP conserves the sum of relevance in each layer, similar to the attention mechanism that upholds the convention that the sum of each row is 1 (softmax).
- LRP was used for explainabilty in CNNs

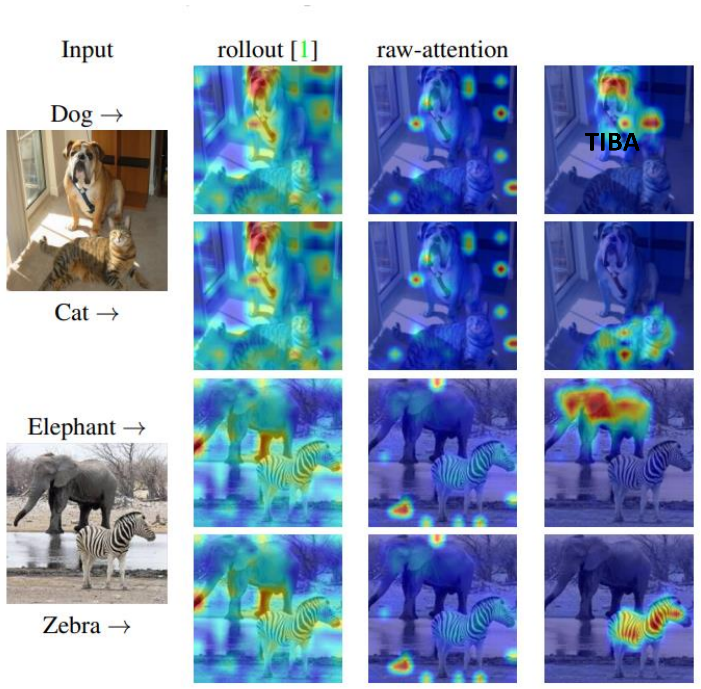
Hila Chefer et.al Transformer Interpretability Beyond Attention Visualization at CVPR 2021

TIBA

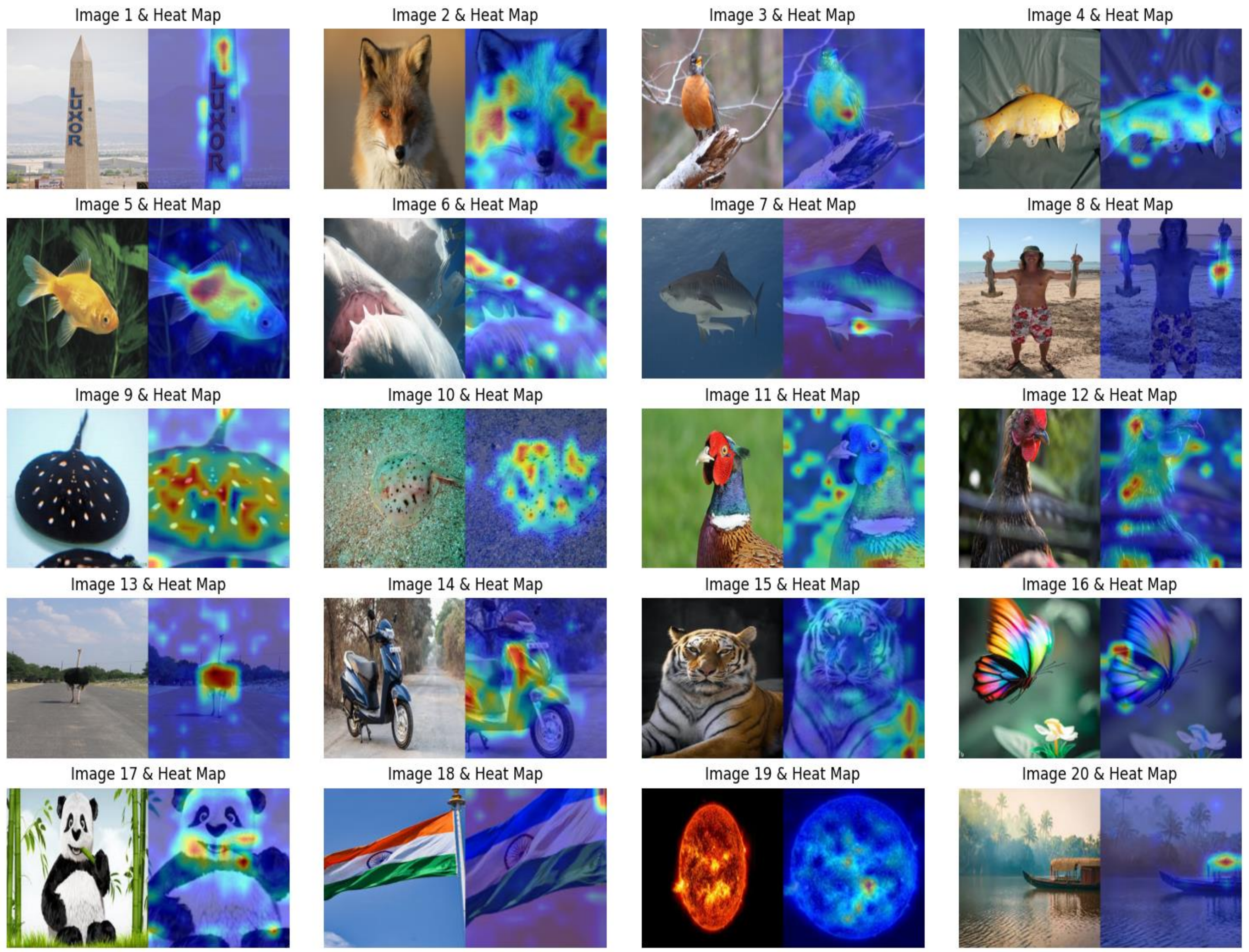
$$\bar{\mathbf{A}}^{(b)} = I + \mathbb{E}_h(\nabla \mathbf{A}^{(b)} \odot R^{(n_b)})^+$$
$$\mathbf{C} = \bar{\mathbf{A}}^{(1)} \cdot \bar{\mathbf{A}}^{(2)} \cdot \ldots \cdot \bar{\mathbf{A}}^{(B)}$$


$\mathbf{C} = \bar{\mathbf{A}}^{(1)} \cdot \bar{\mathbf{A}}^{(2)} \cdot \ldots \cdot \bar{\mathbf{A}}^{(B)}$
Input
Transformer Block
Gradients
Relevance
Transformer Block
Output
Attetion
$\bar{\mathbf{A}}^{(B)} = \mathbb{E}_h$
$\nabla \mathbf{A}^{(B)}$
$R^{(n_B)}$
I
$\bar{\mathbf{A}}^{(1)}$

Transformer Interpretability Beyond Attention Visualization - Hila Chefer et.al
(Detailed Explanation available in paper)



Class specific Heat Map (taken from paper)



This paper claims to be a superior visualisation tool compared to Attention Rollout but that doesn’t seem to be true for the above samples. Few of the images are from not ImageNet. The performance on TIBA on those images is quite poor

DEIT (Data-efficient image transformers)

- Focus on improving performance of Vit on smaller datasets
- Distillation is a way to train a student model (generally smaller) with the help of a teacher (larger) using soft-labels
- It can be regarded as a form of compression of the teacher model into a smaller one the student
- **Distillation through Attention** : In DEIT , hard distillation is used

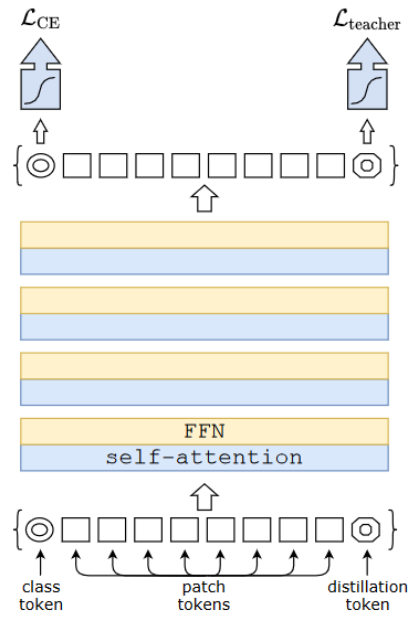
Let Z_t be the logits of the teacher model, Z_s the logits of the student model.

$$y_{\mathrm{t}} = \mathrm{argmax}_c Z_{\mathrm{t}}(c)$$

$$\mathcal{L}_{\mathrm{global}}^{\mathrm{hardDistill}} = \frac{1}{2}\mathcal{L}_{\mathrm{CE}}(\psi(Z_s), y) + \frac{1}{2}\mathcal{L}_{\mathrm{CE}}(\psi(Z_s), y_{\mathrm{t}}).$$

Training data-efficient image transformers & distillation through attention - Touvron et.al

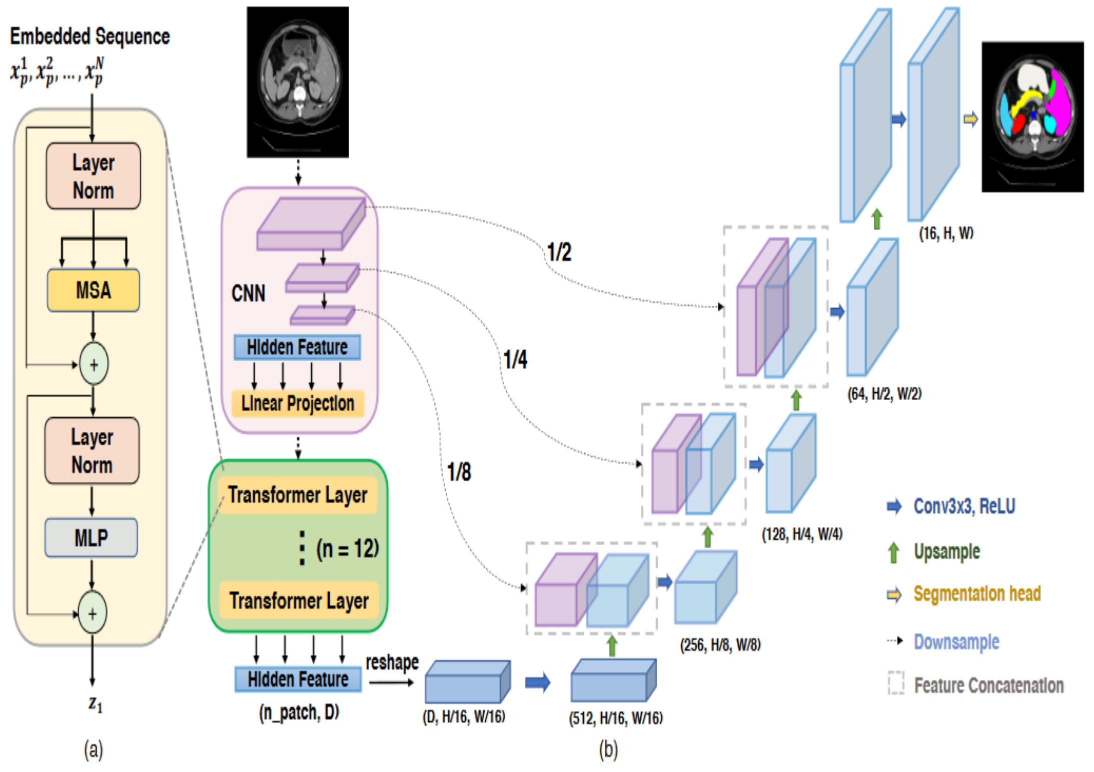
- **It has the same architecture as ViT** except the input token part that having an additional **distillation token (as shown in the figure)**
- The distillation embedding allows our model to learn from the output of the teacher. The distillation token interacts with the rest of tokens just like the class token
- The class and distillation tokens converge to different vectors in the first layer whereas it tends to be more similar in the higher layers
- The teacher is a strong image classifier model



Important Observations:

- ConvNet teacher gives a better performance than a transformer teacher
- **Hard distillation significantly outperforms soft distillation**
- **The two tokens provide complementary information useful for classification.** The classifier on the two tokens is significantly better than the independent class and distillation classifiers, which by themselves already **outperform the distillation baseline**
- **DeiT is slightly below EfficientNet**, which shows that DeiT have almost closed the gap between Vision Transformers and convnets when training with ImageNet only.

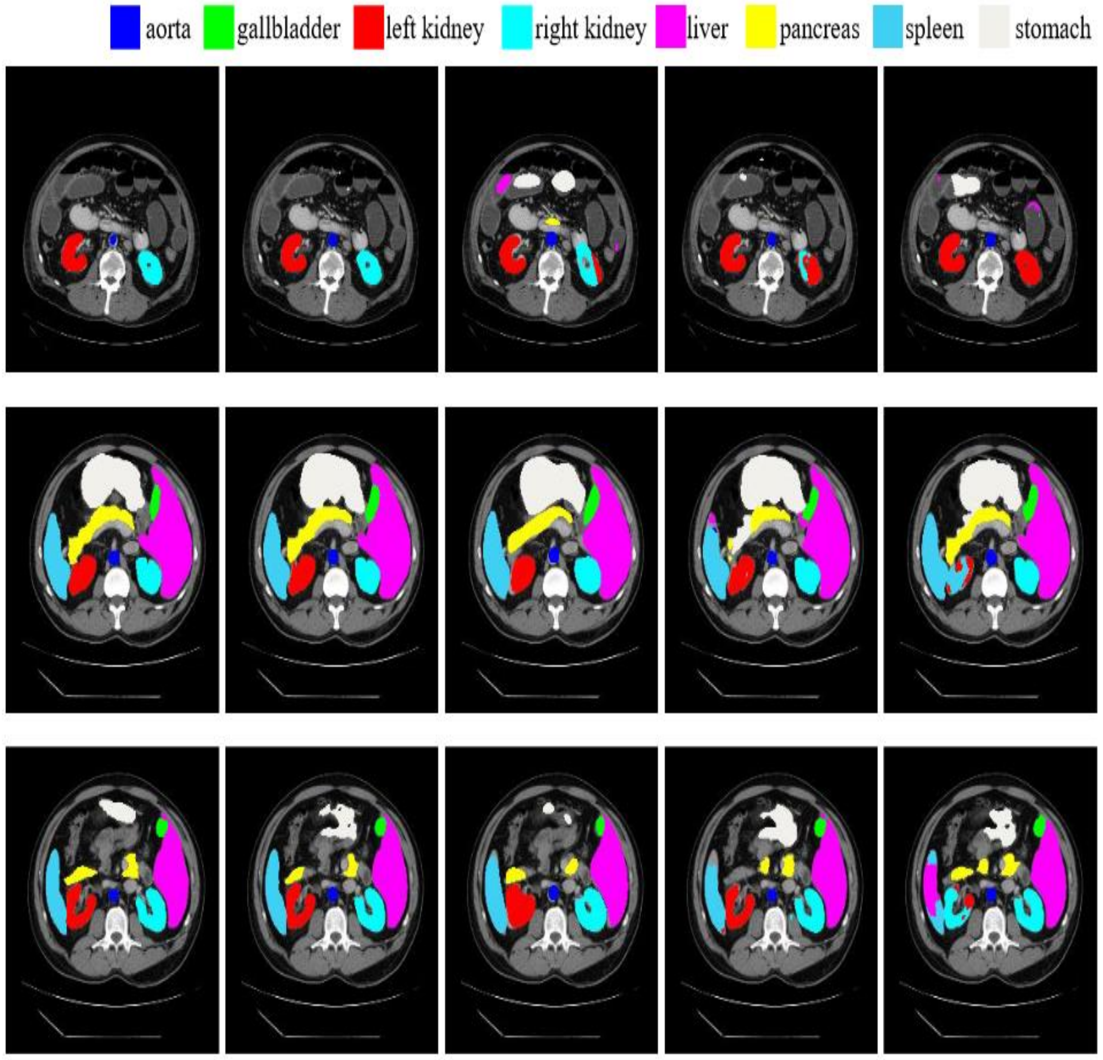
TRANSUNET



- ResNet-50 Encoder
- Base ViT
  - Input resolution – (224,224)
  - P = 16
  - D = 768
  - MLP size = 3072
  - no. of layers = 12
  - no. of heads = 12
- Cascading Upsampler (CUP) Decoder: Bilinear Upsampling and concatenate features from encoder
- Training :
  - SGD with learning rate 0.01, momentum 0.9 and weight decay 1e-4

TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation - Chen et.al

RESULTS



(a) GroundTruth (b) TransUNet (c) R50-ViT-CUP (d) AttnUNet (e) UNet

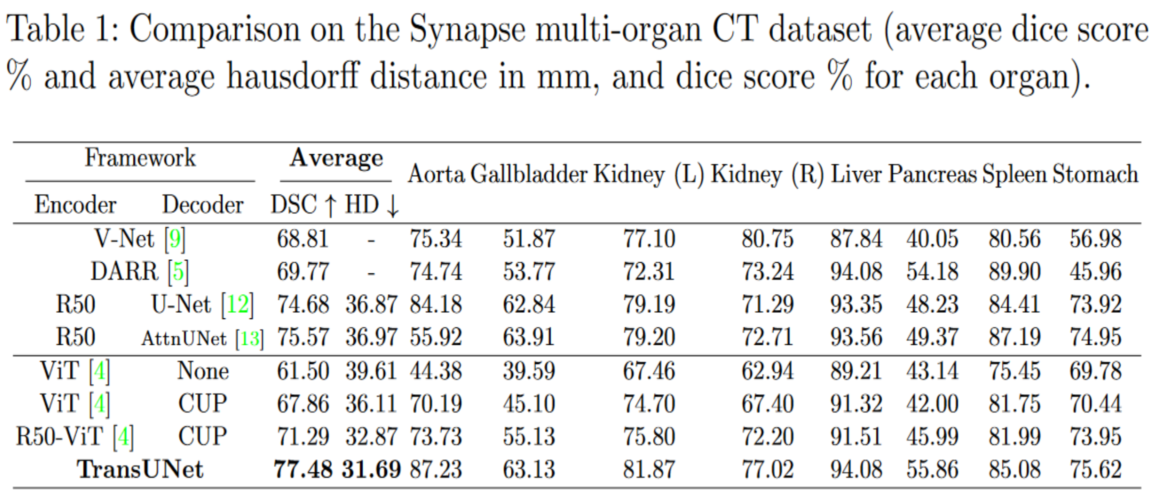


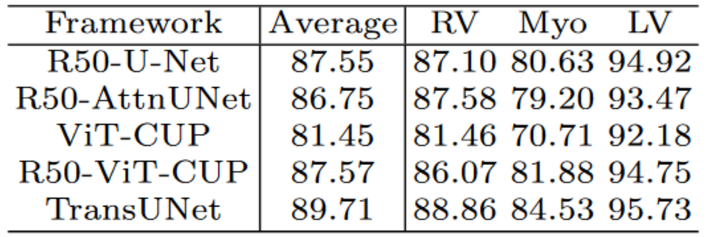
Table 1: Comparison on the Synapse multi-organ CT dataset (average dice score % and average hausdorff distance in mm, and dice score % for each organ).

Framework		**Average**		Aorta	Gallbladder	Kidney (L)	Kidney (R)	Liver	Pancreas	Spleen	Stomach
Encoder	Decoder	DSC ↑	HD ↓								
V-Net [9]		68.81	-	75.34	51.87	77.10	80.75	87.84	40.05	80.56	56.98
DARR [5]		69.77	-	74.74	53.77	72.31	73.24	94.08	54.18	89.90	45.96
R50	U-Net [12]	74.68	36.87	84.18	62.84	79.19	71.29	93.35	48.23	84.41	73.92
R50	AttnUNet [13]	75.57	36.97	55.92	63.91	79.20	72.71	93.56	49.37	87.19	74.95
ViT [4]	None	61.50	39.61	44.38	39.59	67.46	62.94	89.21	43.14	75.45	69.78
ViT [4]	CUP	67.86	36.11	70.19	45.10	74.70	67.40	91.32	42.00	81.75	70.44
R50-ViT [4]	CUP	71.29	32.87	73.73	55.13	75.80	72.20	91.51	45.99	81.99	73.95
TransUNet		**77.48**	**31.69**	87.23	63.13	81.87	77.02	94.08	55.86	85.08	75.62

Synapse multi-organ segmentation dataset.

- 30 abdominal CT scans with 3779 axial clinical CT images of 512 × 512 pixels
- 18 training cases (2212 axial slices) and 12 cases for validation

Comparison on the ACDC dataset in DSC (%).



Framework	Average	RV	Myo	LV
R50-U-Net	87.55	87.10	80.63	94.92
R50-AttnUNet	86.75	87.58	79.20	93.47
ViT-CUP	81.45	81.46	70.71	92.18
R50-ViT-CUP	87.57	86.07	81.88	94.75
TransUNet	89.71	88.86	84.53	95.73

Automated cardiac diagnosis challenge

- Segment into left ventricle (LV), right ventricle (RV) and myocardium (MYO).
- 70 training cases (1930 axial slices), 10 cases for validation and 20 for testing

Results from Ablation Studies

- **6.88%** improvement when trained on **(512,512)** patch at the expense of much larger compute cost
- Consistent improvement in performance while increasing the skip connections [0,1,3]
- Marginal Improvement (**0.35%)** while using **(8,8)** patches
- **1.04 %** improvement while using **Large ViT** (h = 16, L = 24, hidden = 1024,mlp = 4096) compared to base ViT

HYPERPARAMETERS & RESULTS ON DeCAM Data

Vision Transformer :

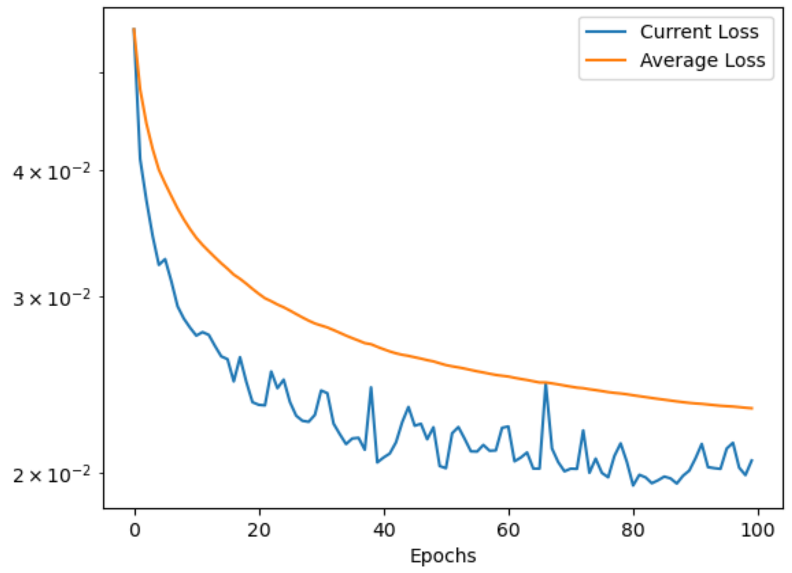
- Patch Size : 16X16
- Hidden Size : 384
- MLP DIM : 768
- No. of heads : 6
- No of encoder layers 6

- **CNN :** No of convolutional layers : [2,2,2]
- **Loss Function :** 0.5*BCE_Loss + 0.5*(1-Dice_Score)
- Learning Rate Scheduler

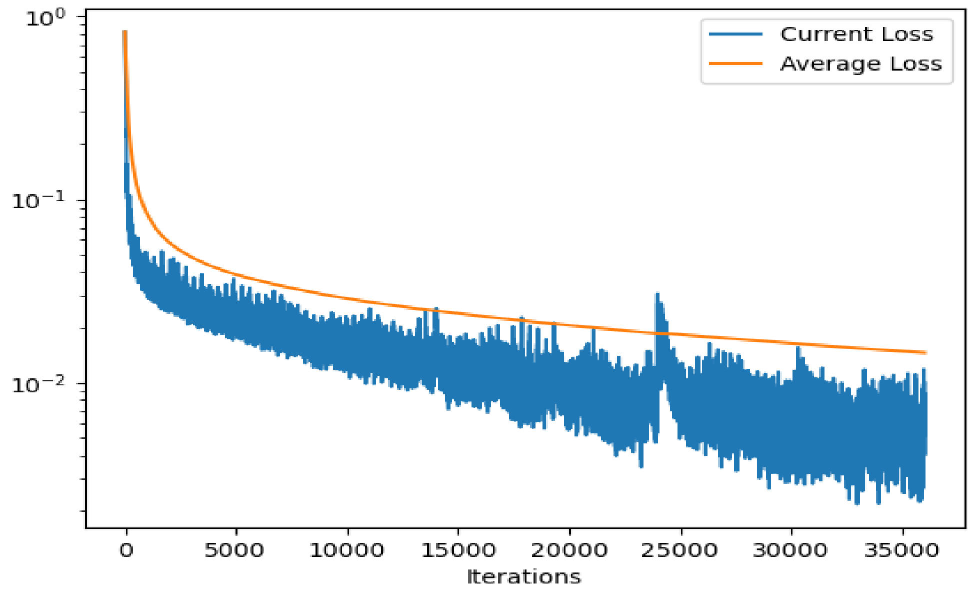
Fixed LR = 0.001

Model	Parameters	TPR @ 0.01% FPR
Above Configuration	16,657,809	0.972736
Loss = 0.1*BCE +0.9*Dice	16,657,809	0.970416
Conv layers : [3,3,3]	18,125,457	**0.972775**

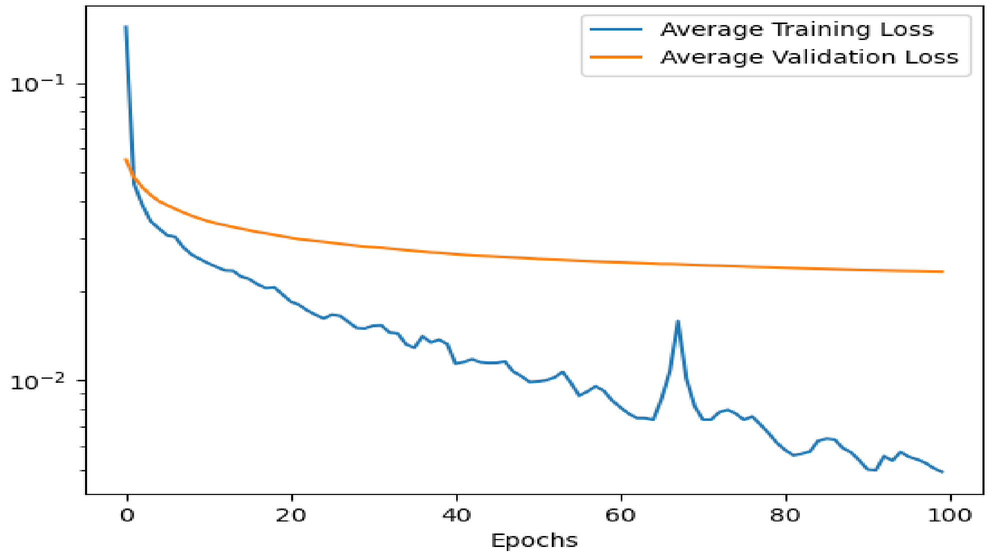
Using dropout = 0.1 in transformer result in a considerable dip in the performance



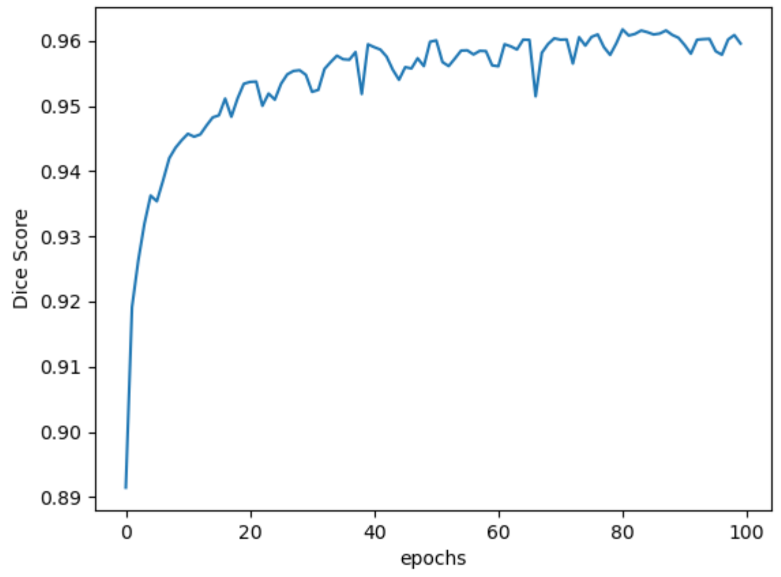
Validation Loss computed at each epoch



Training Loss for each iteration through a batch

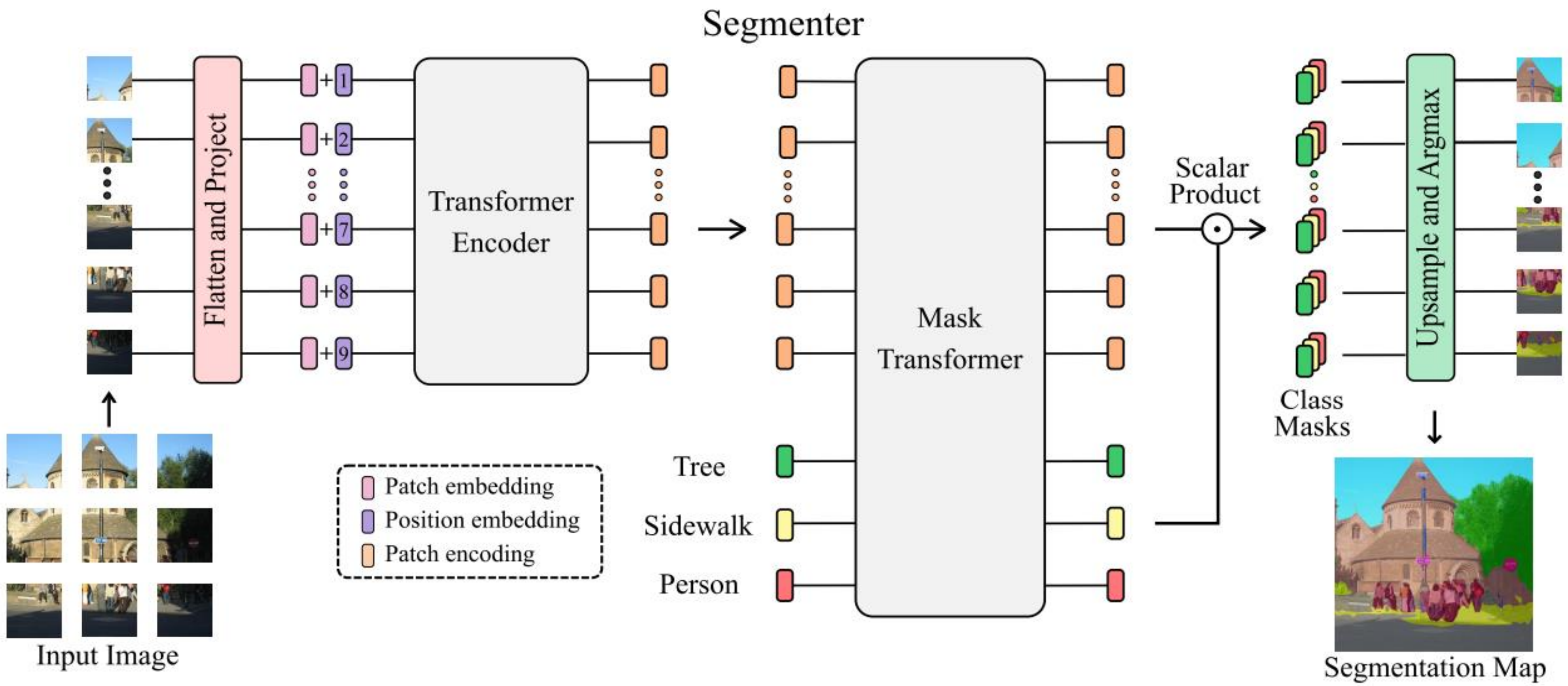


Training Loss(average per epoch) vs Validation for Loss



Dice Score on Validation Set

Segmenter - Transformer for Semantic Segmentation



Segmenter: Transformer for Semantic Segmentation Strudel et.al

References

- LRP - https://iphome.hhi.de/samek/pdf/MonXAI19.pdf
- https://jacobgil.github.io/deeplearning/vision-transformer-explainability
- Attention Gradient Rollout Code - https://colab.research.google.com/drive/1nEvnIC7_fmBi_6UHh9LfYq2p8G8OpxSj?usp=sharing
- https://github.com/hila-chefer/Transformer-Explainability.git
- TransUNet code : https://github.com/Beckschen/TransUNet