Neighborhood Attention Transformer

Sowmya Jayaram Iyer
jayarami@purdue.edu

PURDUE UNIVERSITY | Department of Computer Science

CS59200-TMP (Fall 2022)
October 20, 2022.

OUTLINE

1. Introduction
   1. Motivation
   2. Contributions
2. Related Works
3. Methods
4. Experiments
5. Conclusion

INTRODUCTION

- Transformer-based vision models gained significant attention from the research community since Vision Transformer (ViT) in late 2020
- ViT uses a single Transformer Encoder operating on an embedded space of image patches
- benefits large scale training in image classification
- Following works focused on
    - Data efficiency with minor changes in the architecture eg. Tokens-to-token ViT, Compact transformers, Xcit: Cross-covariance image transformers
    - Efficiency and transferability to downstream tasks eg. SWIN, VoLo

MOTIVATION

Two challenges identified in Vision Transformers:

1. **Not easily applicable to downstream vision tasks, such as object detection and semantic segmentation.**
   - Self-attention has a linear complexity w.r.t the embedding dimension, but a quadratic complexity w.r.t the number of tokens.
   - Number of tokens $\propto$ image resolution
   - Higher image resolution leads to quadratic increase in complexity and memory usage in models strictly using self-attention
   - Downstream tasks usually have higher resolution images compared to classification tasks

MOTIVATION

Two challenges identified in Vision Transformers:

1. **Not easily applicable to downstream vision tasks.**

2. **Inductive biases have to be learned with large sums of data or advanced training**

   - Dot-product self attention is a global 1-dimensional operation by definition
   - The MLP layer in self- attention are only local and translationally equivariant
   - Solution: local-attention transformers
   - SWIN : additional biases injected using shifted-window self attention
   - HaloNet: combination of Stand Alone Self-Attention (SASA) and convolutions provided the best trade-off between memory requirements and translational equivariance

CONTRIBUTIONS

1 Proposing **Neighborhood Attention (NA)**

- Localization of self-attention
- Limiting each query token's receptive field to a fixed-size neighborhood around its corresponding tokens in the key-value pair
- Complexity and Memory usage analysis in comparison with self attention, window - self attention, and convolutions

CONTRIBUTIONS

2 Introducing **Neighborhood Attention Transformer (NAT)**

- A new efficient, accurate and scalable hierarchical transformer made of levels of NA layers
- Each level is followed by a down sampling operation, reduces spatial size by half, as seen in local attention models such as SWIN
- However, NAT utilizes small-kernel overlapping convolutions for embedding and down sampling and not non-overlapping ones
- Introduces a more efficient set of architecture configurations than previous state-of-arts

CONTRIBUTIONS

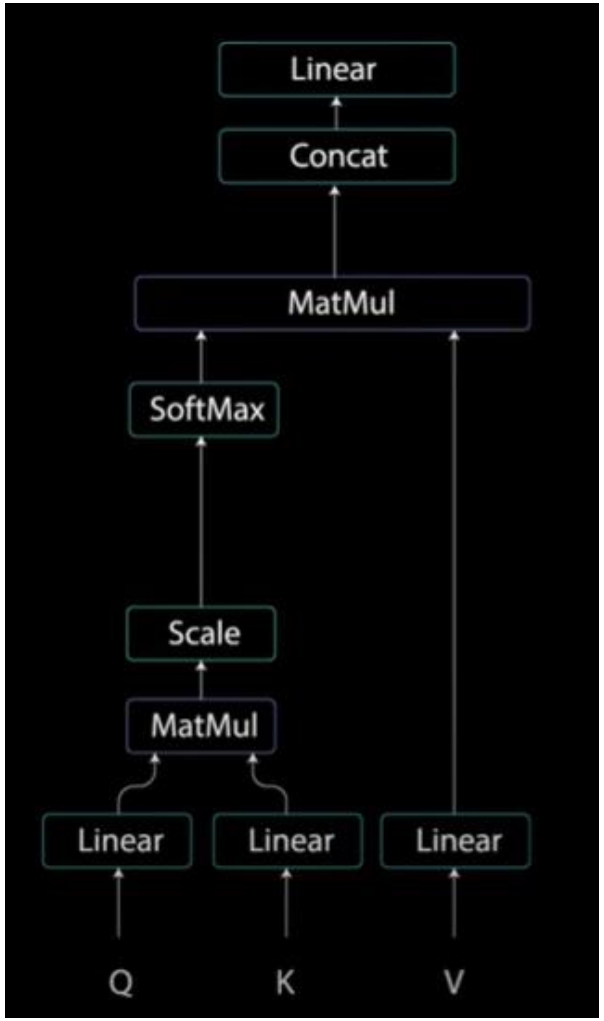
3 Demonstrating NAT’s effectiveness on both classification and downstream vision tasks.

NAT outperforms not only SWIN, but also new convolutional contenders

OUTLINE

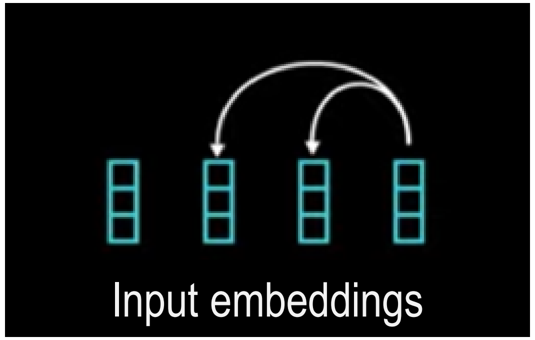
1. Introduction
2. Related Works
    1. Multi-Headed self attention
    2. Vision Transformer
    3. Local-Attention models
3. Methods
4. Experiments
5. Conclusion

MULTI-HEADED SELF ATTENTION



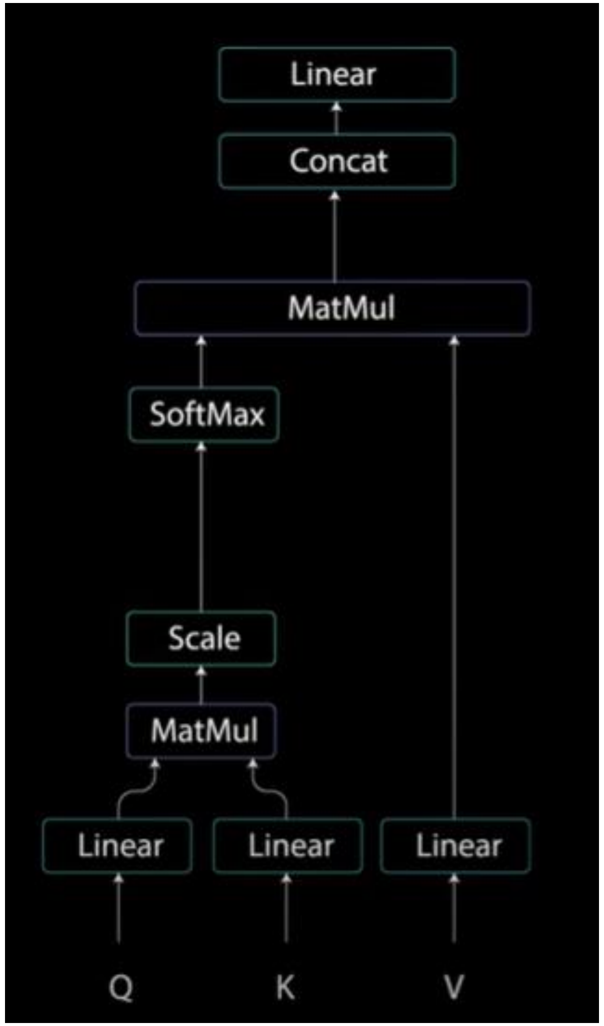
Self Attention:

- takes in an input sequence and uses it as both the query and key-value pairs
- In vision: transforms the pixel/object input feature by encoding its relationship with other pixels/objects



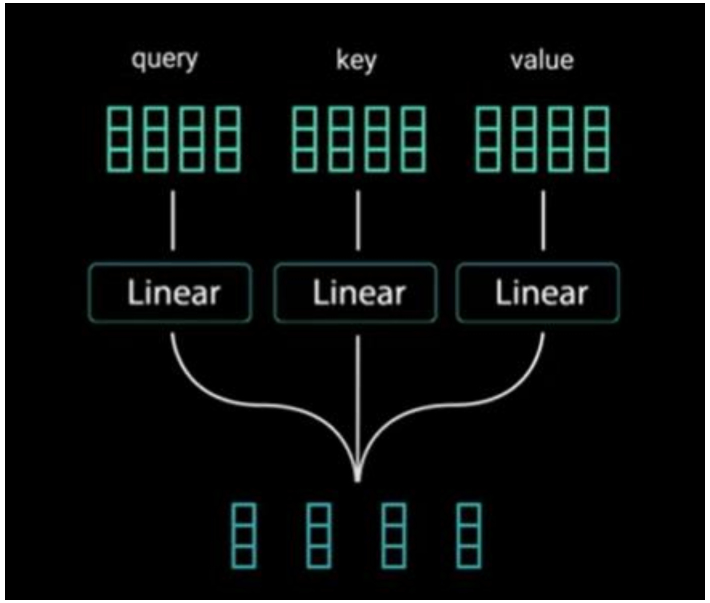
The A.I. Hacker - Michael Phi. (2020) “Illustrated Guide to Transformers Neural Network“. [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



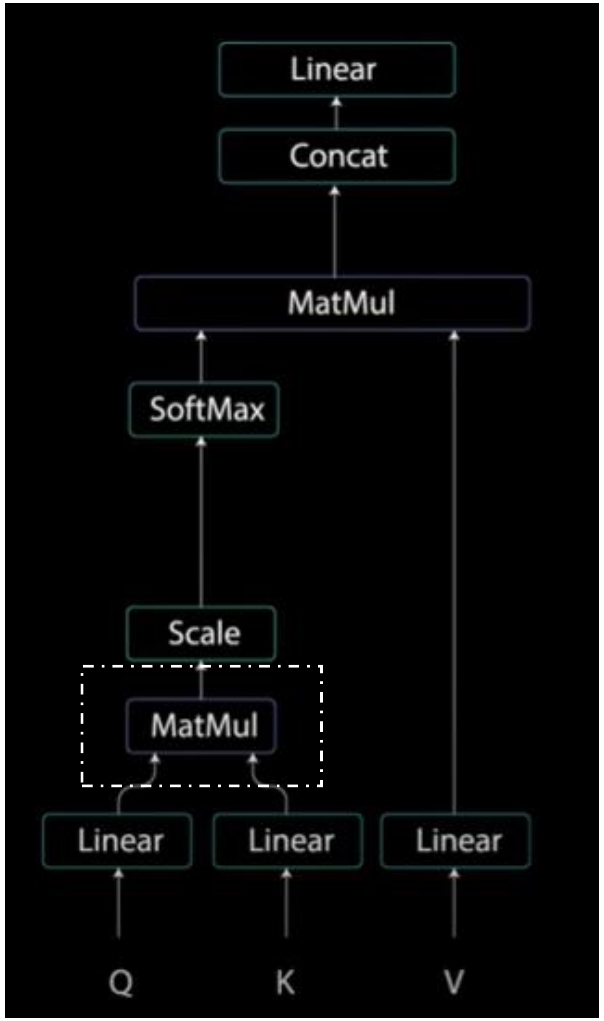
Self Attention:

- We feed the input to three distinct fully connected layers to create Query (Q), Key (K) and Value (V) vectors



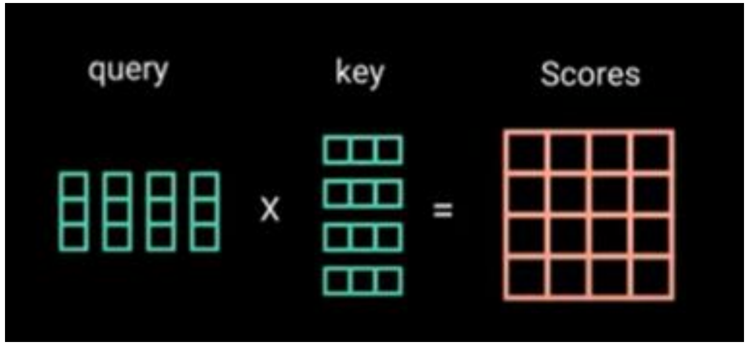
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



Self Attention:

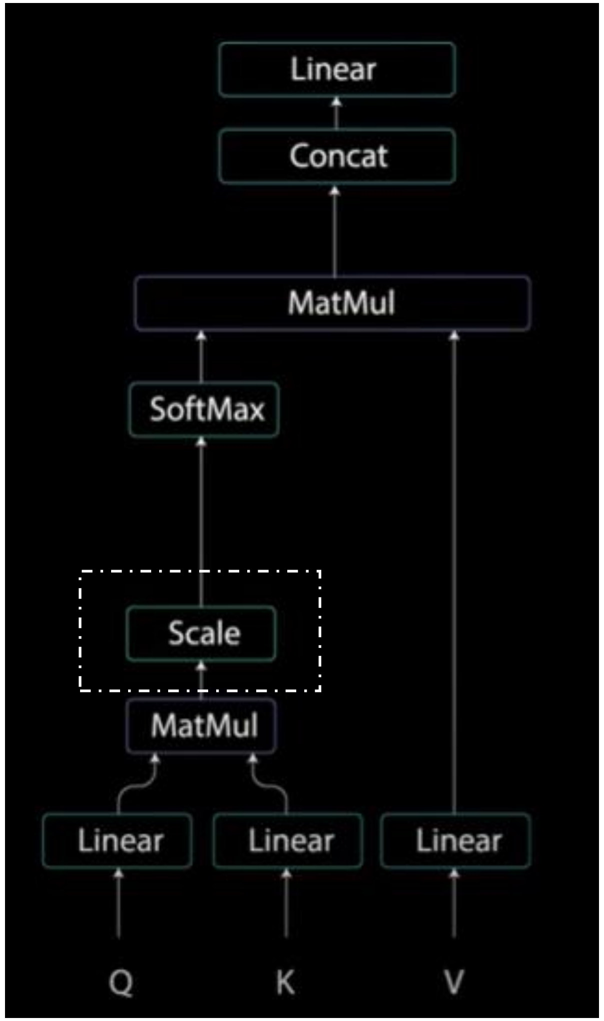
- **MatMul:** Queries (Q) and Keys (K) undergo a dot-product matrix multiplication to give a score matrix



$$QK^T$$

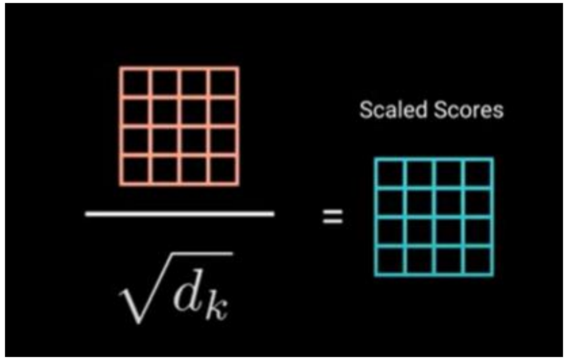
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



Self Attention:

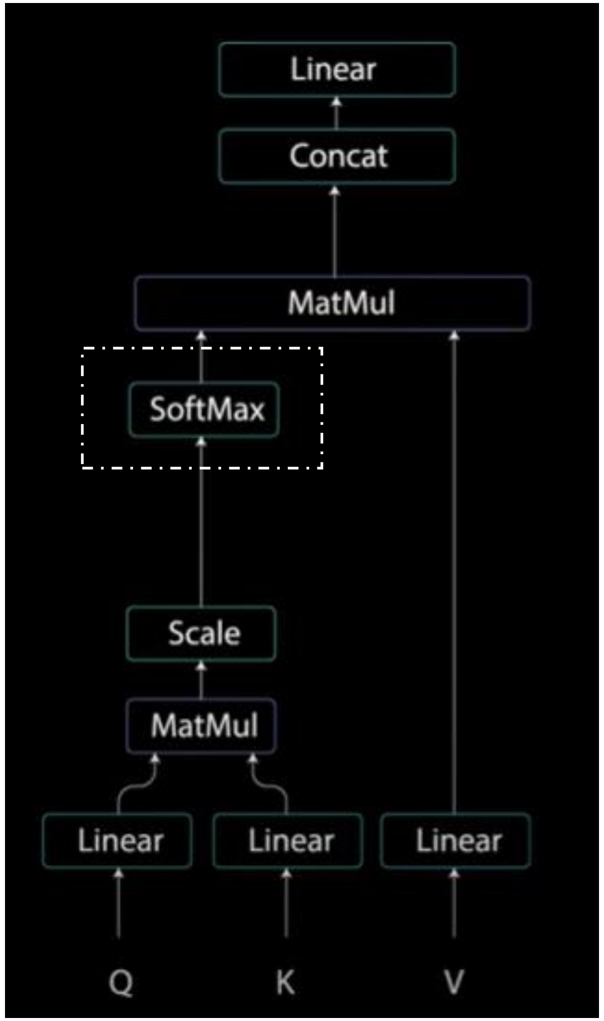
- **Scale:** Score matrix is divided by the square root of dimension of keys to allow stable gradients since multiplying values have exploding effects.



$$\frac{QK^T}{\sqrt{d_K}}$$

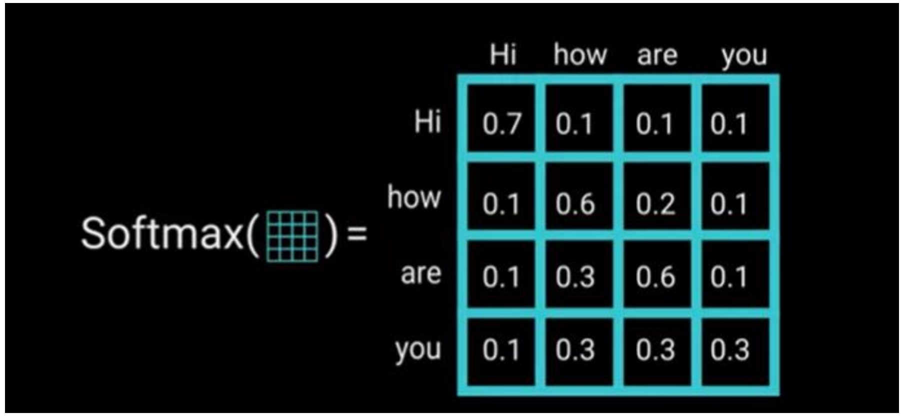
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



Self Attention:

- **Softmax:** Perform Softmax on scaled score matrix to get the *Attention weights*

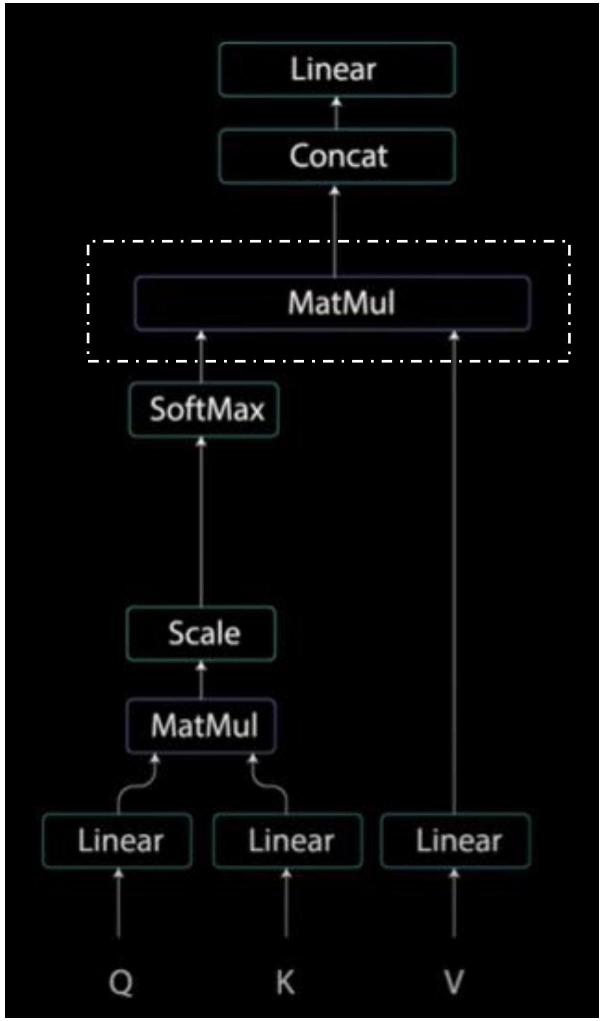


	Hi	how	are	you
Hi	0.7	0.1	0.1	0.1
how	0.1	0.6	0.2	0.1
are	0.1	0.3	0.6	0.1
you	0.1	0.3	0.3	0.3

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

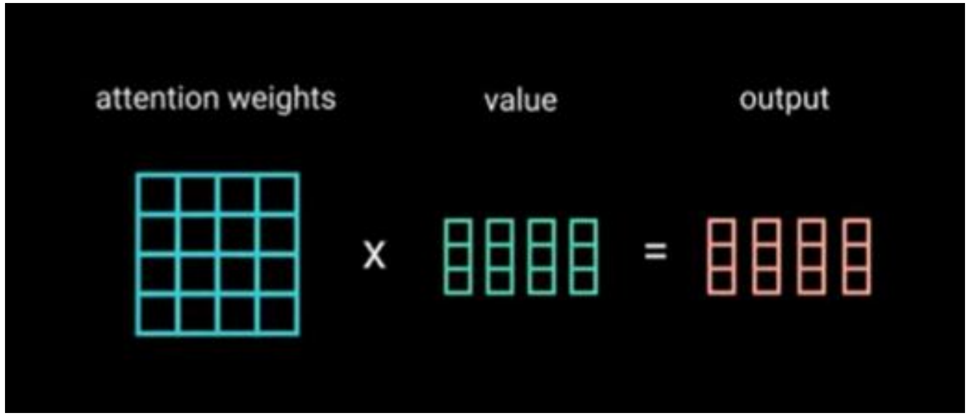
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



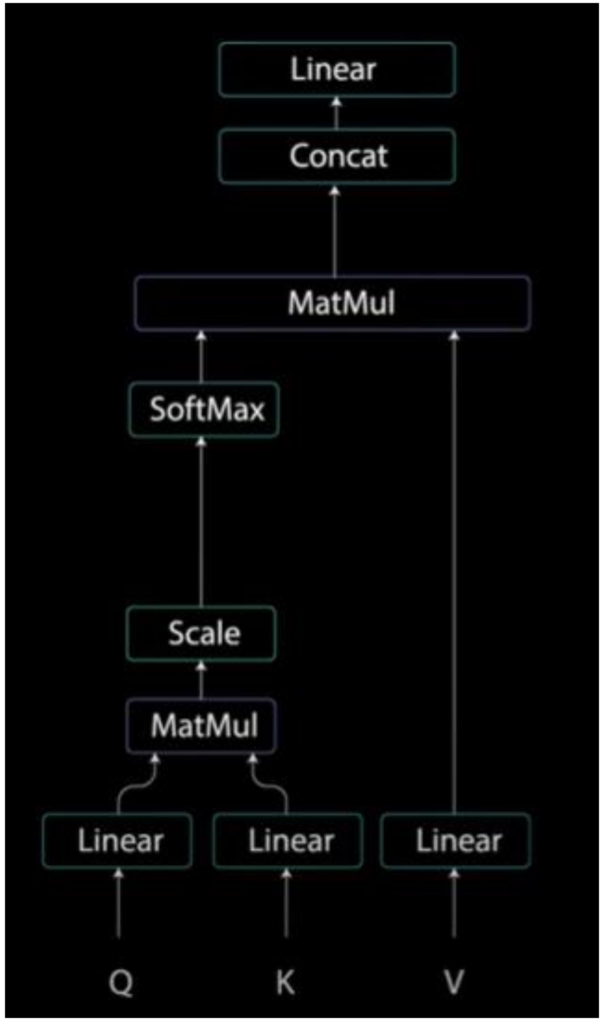
Self Attention:

- **MatMul:** Attention weights would drown out or highlight irrelevant and relevant pixels/objects in Value vectors to give the *output vector*



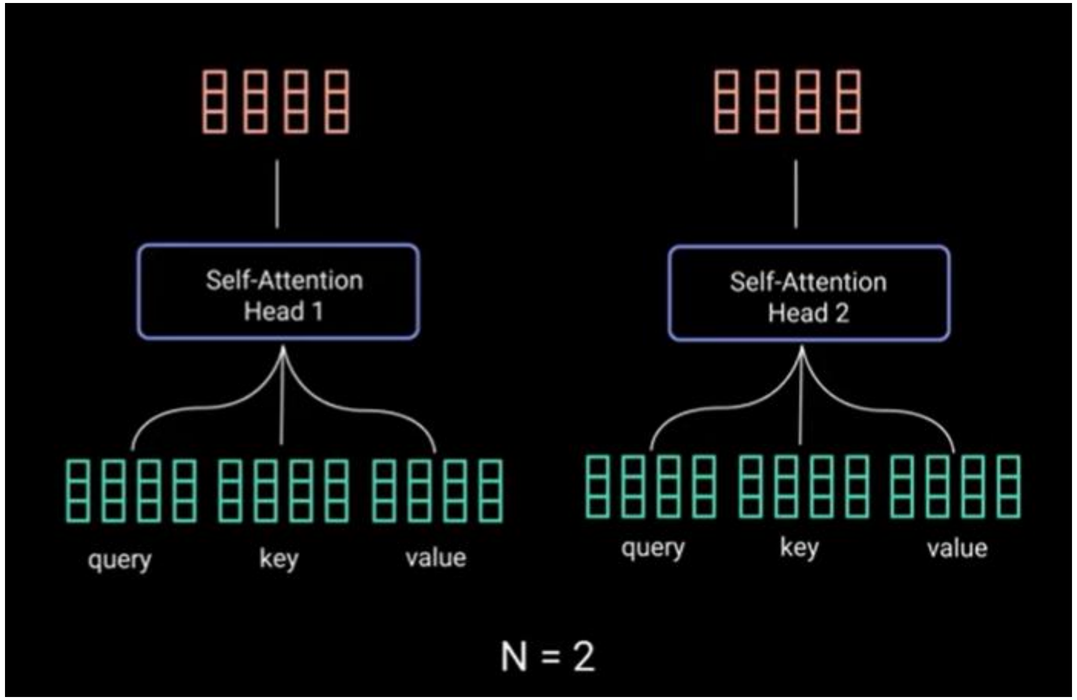
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



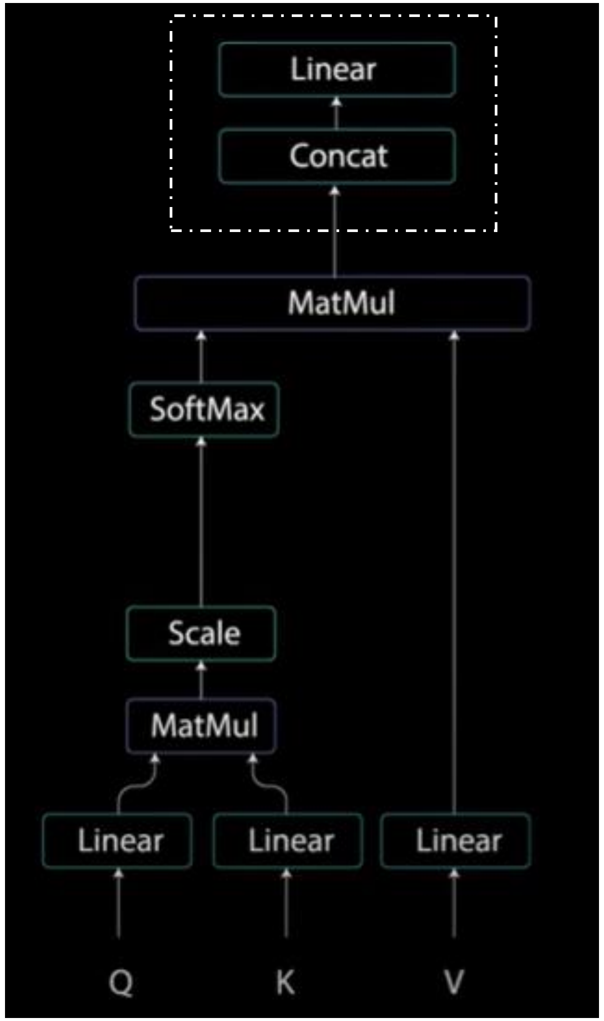
Multi-headed self-attention:

- To make this process multi-headed, input $Q, K \ and \ V$ are split into n-vectors. Each self-attention process is called a head



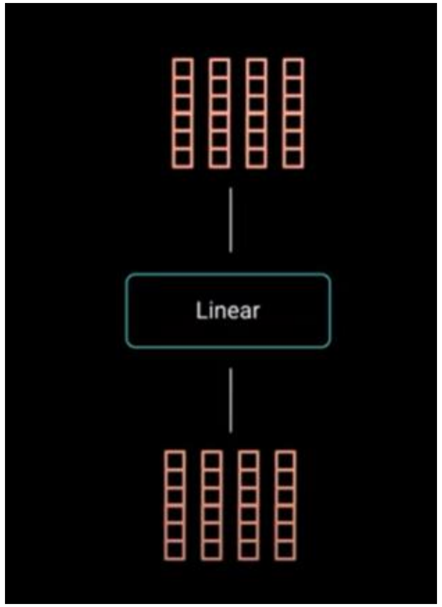
The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



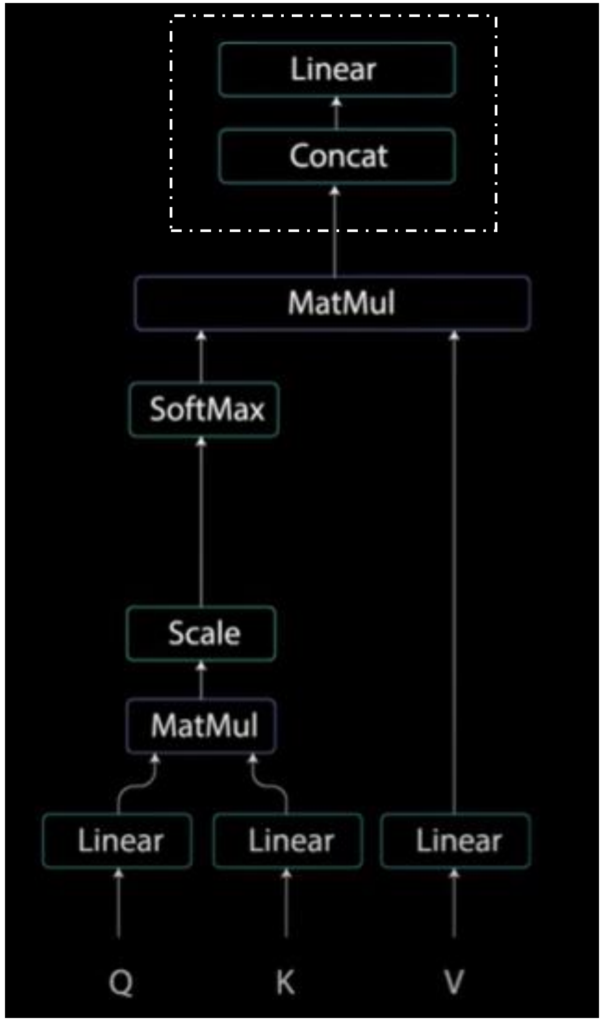
Multi-headed self-attention:

- Each output vector from self-attention process is concatenated before going through the final linear layer.



The A.I. Hacker - Michael Phi. (2020) "Illustrated Guide to Transformers Neural Network". [video clip]. *YouTube. Retrieved by https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

MULTI-HEADED SELF ATTENTION



Multi-headed self-attention:

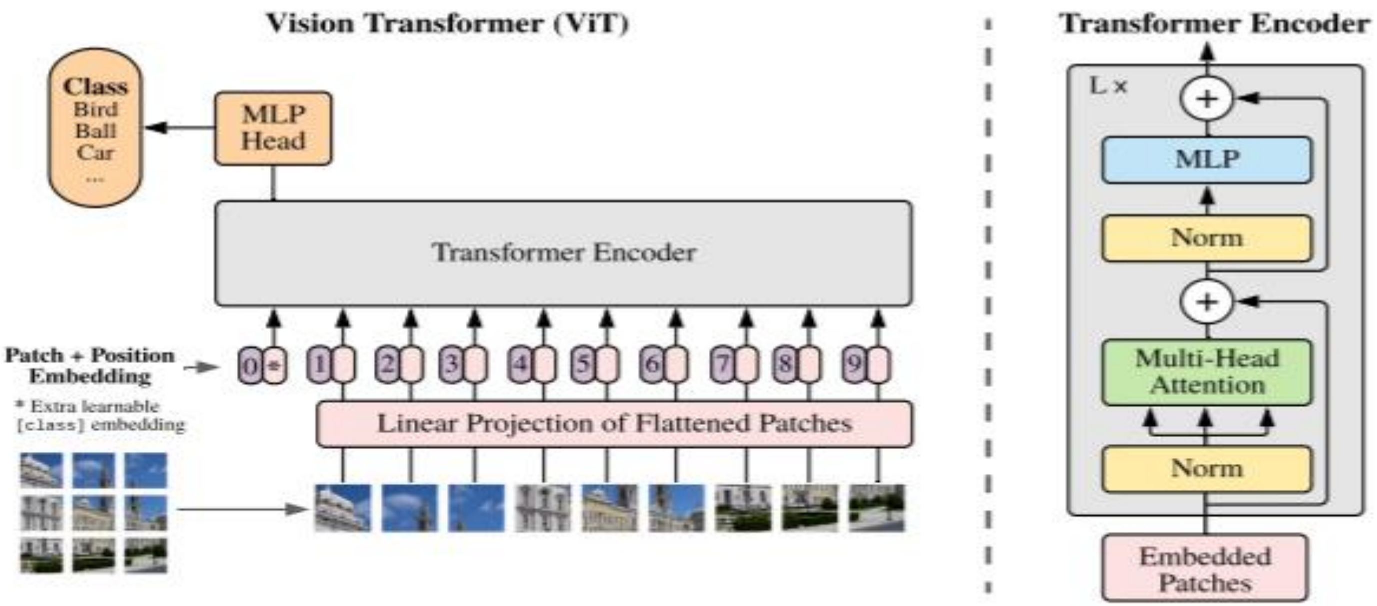
- Summarizing:

$$Attention\,(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_K}}\right)V$$

- Given an input $X \in \mathbb{R}^{M \times D}$, M – no. of tokens and D embedding dimension:
  - Complexity of the operation is $\mathcal{O}(M^2D)$
  - Space complexity $\mathcal{O}(M^2)$

The A.I. Hacker - Michael Phi. (2020) “Illustrated Guide to Transformers Neural Network“. [video clip]. *YouTube. Retrieved by* *https://youtu.be/4Bdc55j80l8?list=RDLV4Bdc55j80l8*

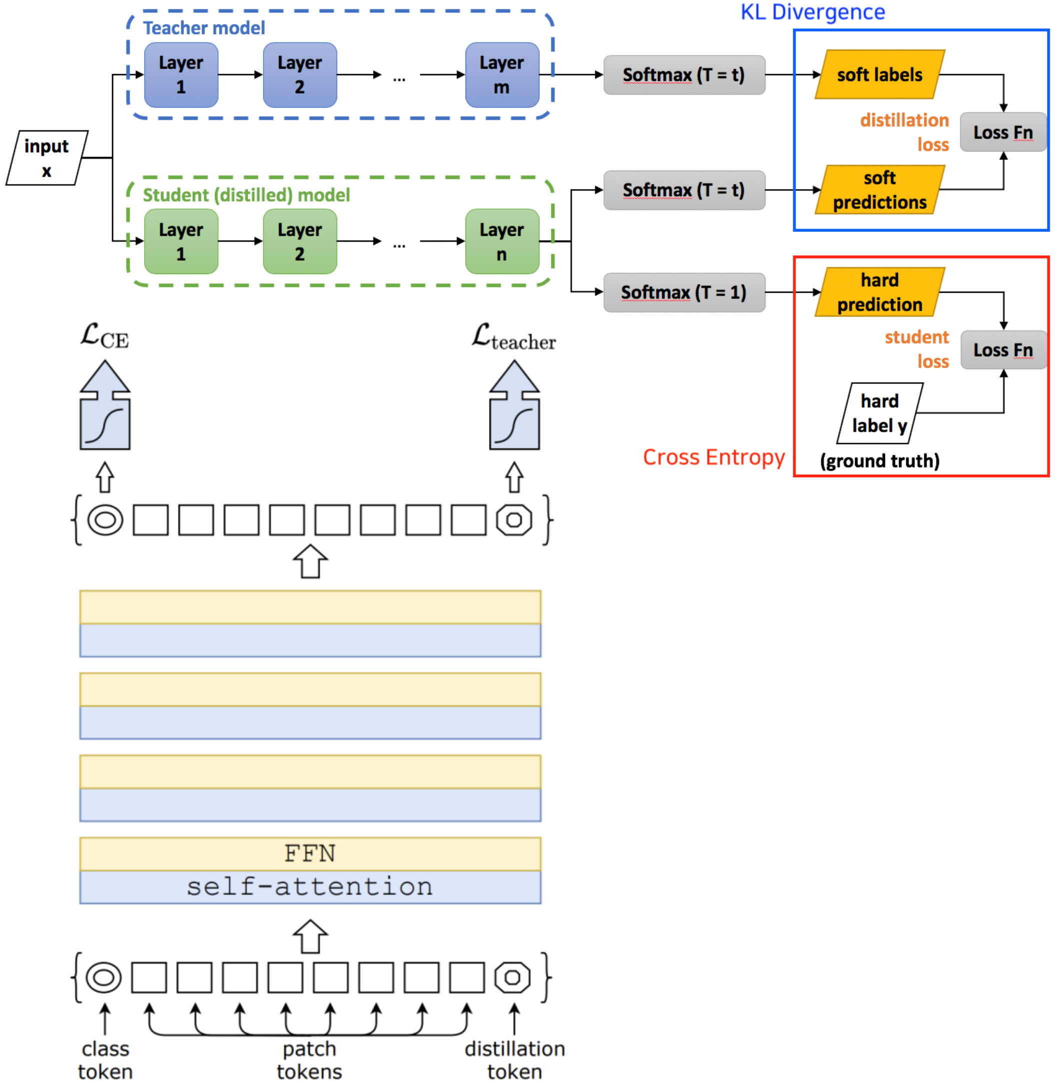
VISION TRANSFORMER



- ViT proposed a model that would only rely on a single non-overlapping convolutional layer (patching and embedding) and a mere transformer encoder.
- When pretrained on large dataset (JFT-300M) outperformed state-of-the-art CNNs on many benchmarks
- Did not achieve competitive results when pre-trained on medium scale datasets, such as ImageNet-1k and ImageNet-21k, due lack of inductive biases which are inherent to CNNs.

Dosovitskiy et al. An Image Is Worth 16x16 Words: Transformers For Image Recognition At Scale. *arXiv preprint arXiv:2010.11929, 2020.*

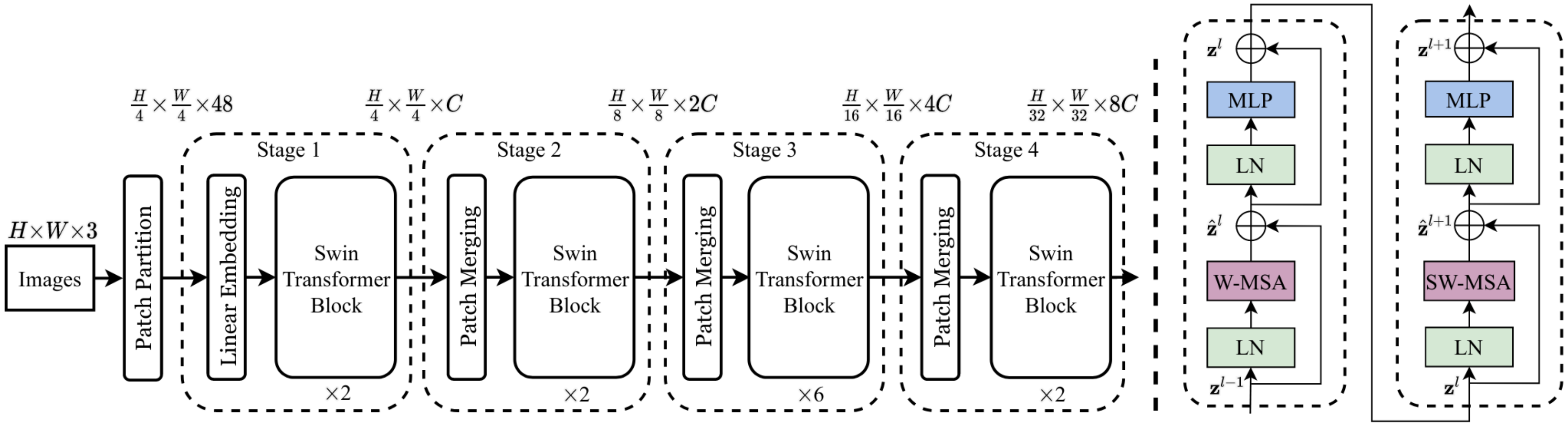
VISION TRANSFORMER



- Subsequent works like Data-efficient image Transformer **(DeiT)** model performed significantly better than ViT with very few architectural changes
- Used knowledge transfer through attention by introducing a distillation token, and a hard distillation loss.
- The distillation token is added to the initial embeddings (patches and class token) and used similarly as the class token in ViT
- Its target objective is given by the distillation component of the loss. The distillation embedding allows our model to learn from the output of the teacher, as in a regular distillation
- CNN as a choice of teacher model improved performance more significantly, as it is arguably transferring inductive biases DeiT lacks

Touvron et al. Training data-efficient image transformers & distillation through attention, 2021.

LOCAL ATTENTION MODELS



(a) Architecture

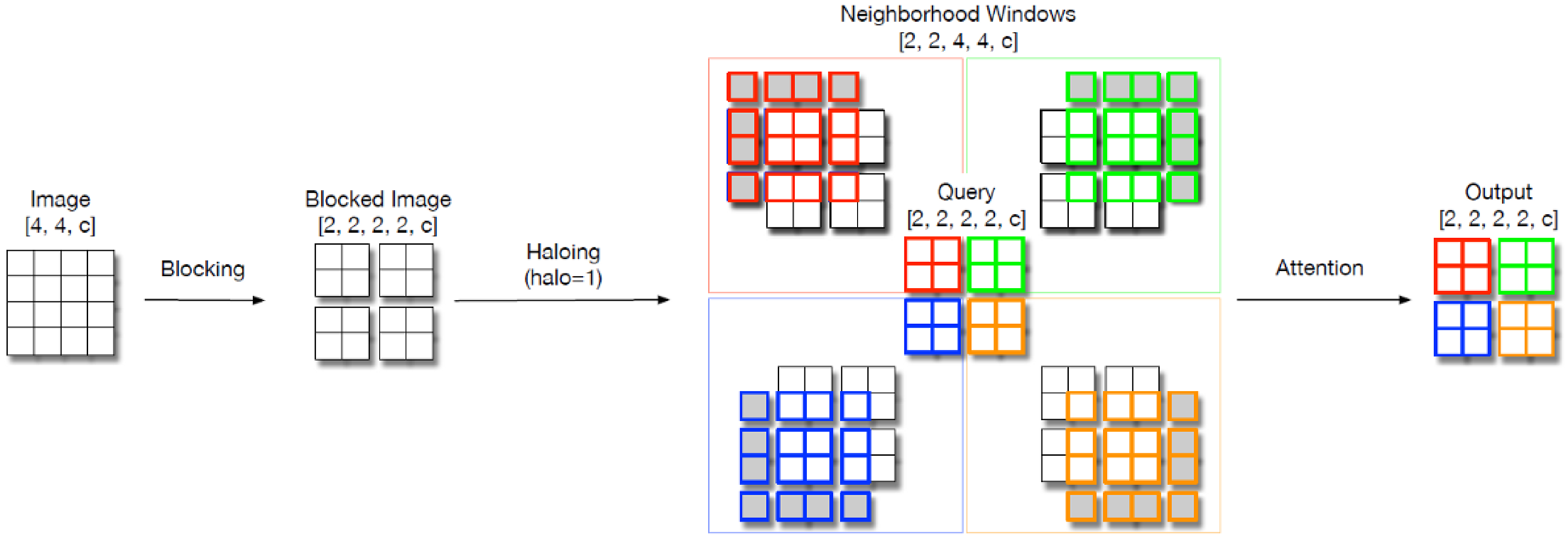
(b) Two Successive Swin Transformer Blocks

Shifted Window (Swin) Attention

- partitions input feature maps and applies self attention to each partition separately
- a shift in pixels prior to the window partitioning stage gives connections across the extracted windows
- produces pyramid-like feature maps, reducing spatial dimensionality while maintaining efficiency
- Outperformed DeiT with a convolutional teacher, at ImageNet-1k classification.
- SOTA method in object detection on the MSCOCO test set, and SOTA in semantic segmentation on ADE20K

Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *https://arxiv.org/abs/2103.14030 , 2021.*

LOCAL ATTENTION MODELS



HaloNet

- operated like a convolution with same zero padding
- extracted key-value pairs by striding the feature map
- Input feature maps are blocked into non-overlapping subsets, which will serve as queries.
- neighboring blocks of equal size are extracted, which will serve as keys and values then sent into SA module
- effective at both reducing cost and improving performance

Vaswani et al. Scaling Local Self-Attention for Parameter Efficient Visual Backbone. *2021*.

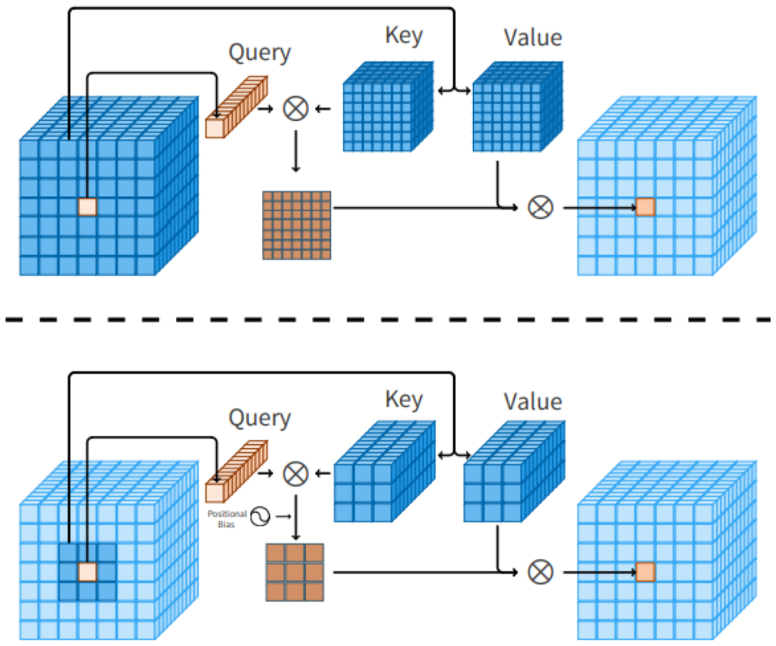
OUTLINE

1. Introduction
2. Related Works
3. Methods
    1. Neighborhood Attention
    2. Neighborhood Attention Transformer
    3. Complexity Analysis
4. Experiments
5. Conclusion

NEIGHBORHOOD ATTENTION

Self Attention

Neighborhood Attention



- allows each pixel in feature maps to only attend to its neighboring pixels
- neighborhood of a pixel at (i, j) is denoted as $\rho(i, j)$ which is a fixed-length $(\mathcal{L})$ set of indices of pixels nearest to (i, j)
- For a neighborhood size of $\mathcal{L} \times \mathcal{L}$: $\|\rho(i,j)\| = \mathcal{L}^2$
- NA on a single pixel:

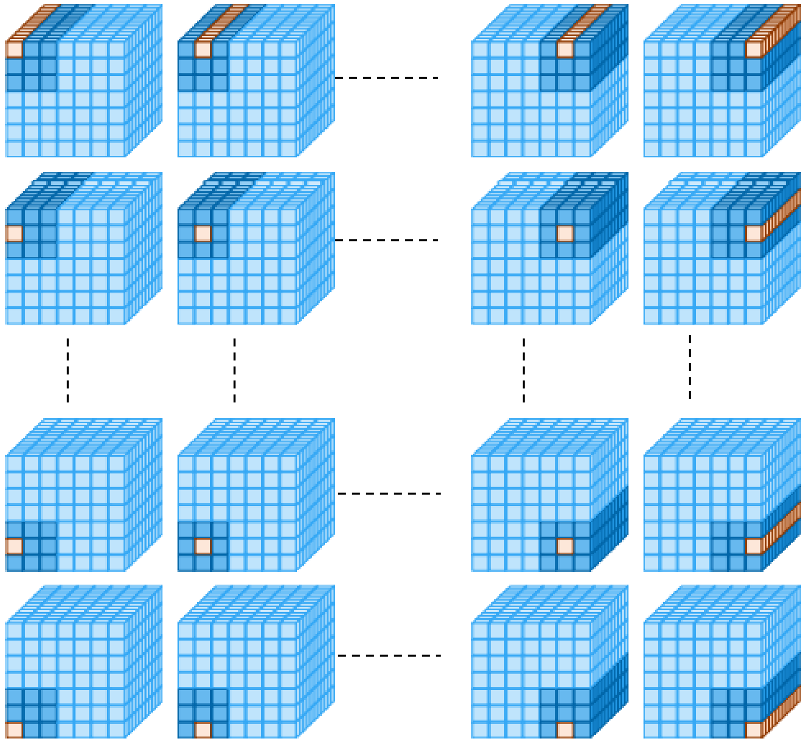
$$NA(X_{i,j}) = softmax\left(\frac{Q_{i,j}K^T_{\rho(i,j)} + B_{i,j}}{scale}\right)V_{\rho(i,j)}$$

where $B_{i,j}$ (relative positional bias) is added to each attention weight based on its relative position.

- If ρ maps each pixel to **all pixels,** $K_{\rho(i,j)} = K$ and $V_{\rho(i,j)} = V$, gives self-attention with additional positional bias

Hassani et al. Neighborhood Attention Transformer. *2022.*

NEIGHBORHOOD ATTENTION



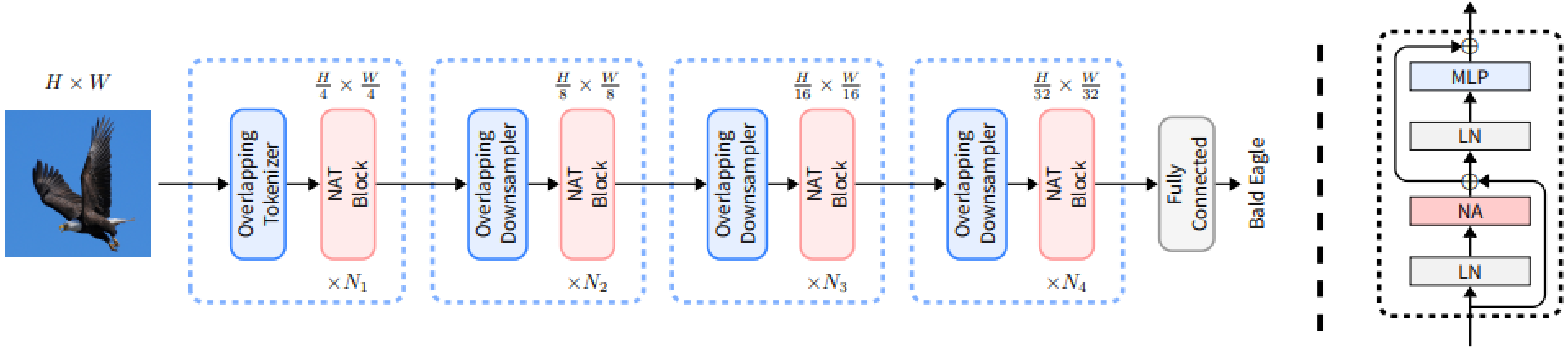
- NA operation is repeated for every pixel in the feature map

Key Design Choice:

- for corner pixels that cannot be centered, the neighborhood is expanded to maintain receptive field size.
- allows NA to generalize to self attention as neighborhood size grows towards the feature map resolution $\rho \to (H \times W) \Rightarrow \|\rho(i,j)\| = H \times W$
- For example, let $\mathcal{L} = 3$. For a corner pixel, the neighborhood is another 3×3 grid, but with the query not positioned in the center.
- **Contrast:** In SWIN, the windows are arranged to evenly partition the image in a non-overlapping manner

Hassani et al. Neighborhood Attention Transformer. *2022.*

NEIGHBORHOOD ATTENTION TRANSFORMER



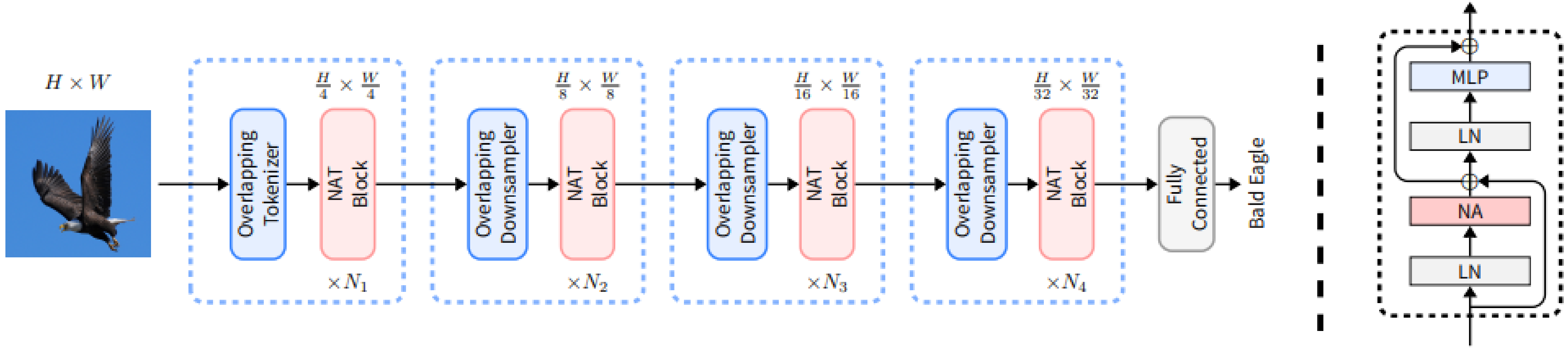
Neighborhood Attention Transformer Architecture

NAT Block

- NAT embeds inputs using 2 consecutive 3 × 3 convolutions with 2 × 2 strides, resulting in a spatial size 1/4th the size of the input.
- similar to using a patch and embedding layer with 4 × 4 patches but using overlapping convolutions
- using overlapping convolutions would increase cost and two convolutions incurs more parameters
- The model is reconfigured using a downsampler providing a better trade off

Hassani et al. Neighborhood Attention Transformer. *2022.*

NEIGHBORHOOD ATTENTION TRANSFORMER



Neighborhood Attention Transformer Architecture

NAT Block

- NAT consists of 4 levels, each followed by a downsampler (except the last)
- Downsampling operator computes a representation of the input at scale k_i of length of $\frac{n}{k_i}$.
- In this paper, downsampler decrease spatial size in half, while doubling the number of channels
- Use 3 × 3 convolutions with 2 × 2 strides, instead of 2 × 2 non-overlapping convolutions that Swin uses (patch merge)
- the tokenizer downsamples by a factor of 4
- It allows for easier transfer of pre-trained models to downstream tasks

Hassani et al. Neighborhood Attention Transformer. *2022.*

COMPLEXITY ANALYSIS

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Input feature maps: shape$(H \times W \times C)$
Neighborhood/ window sizes $\mathcal{L}$

Self Attention model: (ignoring SA heads for simplicity)

Computation Cost:

- Q, K, V linear projections - $\mathcal{O}(WHC^2)$
- Attention weight computation - $\mathcal{O}(H^2W^2C)$

Memory Usage

- Attention weights - $\mathcal{O}(WHC)$

Hassani et al. Neighborhood Attention Transformer. *2022.*

COMPLEXITY ANALYSIS

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Input feature maps: shape$(H \times W \times C)$
Neighborhood/ window sizes $\mathcal{L}$

Swin:

Computation Cost:

- Q, K, V linear projections - $\mathcal{O}(WHC^2)$
- Attention weight computation:
  - divides the queries, keys, and values into $\frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}}$ windows and then applies attention to each window costing
  - $\mathcal{O}\left(\frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}} C\, \mathcal{L}^4\right) = \mathcal{O}(HWC\mathcal{L}^2)$

Memory Usage:

- Shape of attention weights $= \frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}} \times \mathcal{L}^2 \times \mathcal{L}^2$, therefore: $\mathcal{O}(HW\mathcal{L}^2)$

Hassani et al. Neighborhood Attention Transformer. *2022.*

COMPLEXITY ANALYSIS

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Input feature maps: shape$(H \times W \times C)$
Neighborhood/ window sizes $\mathcal{L}$

NAT:

Computation Cost:

- Q, K, V linear projections - $\mathcal{O}(WHC^2)$
- Attention weight computation:
  - Each query $Q_{i,j}$ has both keys and values of size $\mathcal{L} \times \mathcal{L} \times C$ into $\frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}}$ windows and then applies attention to each window costing
  - $\mathcal{O}\left(\frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}} C\, \mathcal{L}^4\right) = \mathcal{O}(HWC\mathcal{L}^2)$

Memory Usage:

- Shape of attention weights $= \frac{H}{\mathcal{L}} \times \frac{W}{\mathcal{L}} \times \mathcal{L}^2 \times \mathcal{L}^2$, therefore: $\mathcal{O}(HW\mathcal{L}^2)$

Hassani et al. Neighborhood Attention Transformer. *2022.*

COMPLEXITY ANALYSIS

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Input feature maps: shape$(H \times W \times C)$
Neighborhood/ window sizes $\mathcal{L}$

2D Convolution:

Computation Cost:

- Quadratic with respect to Channels, kernel size and image dimension- $\mathcal{O}(WHC^2\mathcal{L}^2)$

Memory Usage:

- $\mathcal{O}(HWC)$

Hassani et al. Neighborhood Attention Transformer. *2022.*

COMPLEXITY ANALYSIS

Module	Computation	Memory
Self attention	$\mathcal{O}\left(3HWC^2 + 2H^2W^2C\right)$	$\mathcal{O}\left(3HWC + H^2W^2\right)$
2D Window attention (Swin)	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Neighborhood attention	$\mathcal{O}\left(3HWC^2 + 2HWCL^2\right)$	$\mathcal{O}\left(3HWC + HWL^2\right)$
2D Convolution	$\mathcal{O}\left(HWC^2L^2\right)$	$\mathcal{O}\left(HWC\right)$

Input feature maps: shape$(H \times W \times C)$
Neighborhood/ window sizes $\mathcal{L}$

NAT vs 2D Convolution:

- For L > 1, NAT grows less quickly than the 2D convolution as the C is increased
- Specifically for L = 3, NA is more efficient for all C > 3, which in practice is usually the case.
- For L ≥ 5, NA is more efficient for all C > 1.
- 2D NA is less computationally complex than a 2D convolution in practical scenarios
- Only additional memory usage due to the QKV projections.

Hassani et al. Neighborhood Attention Transformer. *2022.*

OUTLINE

1. Introduction
2. Related Works
3. Methods
4. Experiments
    1. Classification
    2. Object Detection
    3. Semantic Segmentation
5. Conclusion

EXPERIMENTS

NOTE: For all experiments, a 7 × 7 window size (similar to Swin) is used

Table 1: A summary of NAT Configurations. Channel double after every level until the final one.

Variant	Layers	Dim × Heads	MLP ratio	# Params	FLOPs
NAT-Mini	3, 4, 6, 5	32 × 2	3	20 M	2.7 G
NAT-Tiny	3, 4, 18, 5	32 × 2	3	28 M	4.3 G
NAT-Small	3, 4, 18, 5	32 × 3	2	51 M	7.8 G
NAT-Base	3, 4, 18, 5	32 × 4	2	90 M	13.7 G

Hassani et al. Neighborhood Attention Transformer. *2022.*

CLASSIFICATION

- All variants are trained on medium-scale ImageNet-1k dataset.
- Swin's training configurations were used:
  - AdamW optimizer for 300 epochs using a cosine decay learning rate scheduler and 20 epochs of linear warm-up. A batch size of 1024, an initial learning rate of 0.001, and a weight decay of 0.05

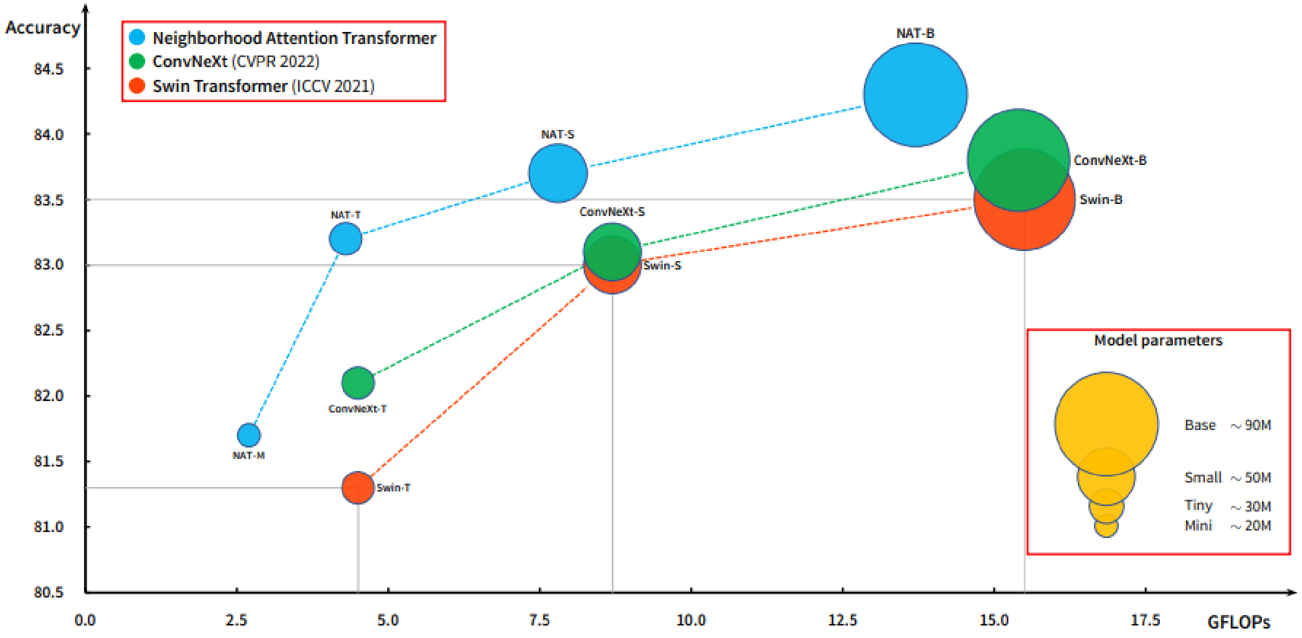


Figure 3: ImageNet-1k classification performance versus compute, with bubble size representing the number of parameters. NAT outperfoms both Swin and ConvNeXt in classification with fewer FLOPs, and a similar number of parameters.

Hassani et al. Neighborhood Attention Transformer. *2022.*

OBJECT DETECTION

- With NAT as backbones, used pretrained (on ImageNet) Mask and Cascade Mask R-CNN to train on MSCOCO with Swin's training configurations

Mask -RCNN

Backbone	# Params	FLOPs	Throughput (FPS)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
NAT-Mini	40 M	225 G	54.1	46.5	68.1	51.3	41.7	65.2	44.7
Swin-Tiny [18]	48 M	267 G	45.1	46.0	68.1	50.3	41.6	65.1	44.9
ConvNeXt-Tiny [19]	48 M	262 G	52.0	46.2	67.0	50.8	41.7	65.0	44.9
NAT-Tiny	48 M	258 G	44.5	47.7	69.0	52.6	42.6	66.1	45.9
Swin-Small [18]	69 M	359 G	31.7	48.5	70.2	53.5	43.3	67.3	46.6
NAT-Small	70 M	330 G	34.8	48.4	69.8	53.2	43.2	66.9	46.5

Cascade Mask -RCNN

Backbone	# Params	FLOPs	Throughput (FPS)	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
NAT-Mini	77 M	704 G	27.8	50.3	68.9	54.9	43.6	66.4	47.2
Swin-Tiny [18]	86 M	745 G	25.1	50.4	69.2	54.7	43.7	66.6	47.3
ConvNeXt-Tiny [19]	86 M	741 G	27.3	50.4	69.1	54.8	43.7	66.5	47.3
NAT-Tiny	85 M	737 G	24.9	51.4	70.0	55.9	44.5	67.6	47.9
Swin-Small [18]	107 M	838 G	20.3	51.9	70.7	56.3	45.0	68.2	48.8
ConvNeXt-Small [19]	108 M	827 G	23.0	51.9	70.8	56.5	45.0	68.4	49.1
NAT-Small	108 M	809 G	21.7	52.0	70.4	56.3	44.9	68.1	48.6
Swin-Base [18]	145 M	982 G	17.3	51.9	70.5	56.4	45.0	68.1	48.9
ConvNeXt-Base [19]	146 M	964 G	19.5	52.7	71.3	57.2	45.6	68.9	49.5
NAT-Base	147 M	931 G	18.6	52.3	70.9	56.9	45.1	68.3	49.1

- NAT-Mini outperforms Swin-Tiny with significantly fewer FLOPs.
- NAT-Tiny outperforms both its Swin and ConvNeXt counterparts, again with slightly fewer FLOPs, with both Mask and Cascade Mask R-CNN.
- NAT-Small and NAT-Base can reach similar-level performance with both detectors compared to their Swin and ConvNeXt counterparts with more efficiency.

Hassani et al. Neighborhood Attention Transformer. *2022.*

SEMANTIC SEGMENTATION

- trained UPerNet on ADE20K with ImageNet-pretrained backbones.
- Followed Swin's configuration for training

Backbone	# Params	FLOPs	Throughput (FPS)	mIoU	mIoU(ms)
ResNet101 [13]	47 M	-	-	38.8	-
DeiT-S [26, 18]	52 M	1094 G	-	44.0	-
NAT-Mini	50 M	900 G	24.5	45.1	46.4
Swin-Tiny [18]	60 M	946 G	21.3	44.5	45.8
ConvNeXt-T [19]	60 M	939 G	23.3	46.0	46.7
NAT-Tiny	58 M	934 G	21.4	47.1	48.4
Swin-Small [18]	81 M	1040 G	17.0	47.6	49.5
ConvNeXt-Small [19]	82 M	1027 G	19.1	48.7	49.6
NAT-Small	82 M	1010 G	17.9	48.0	49.5
Swin-Base [18]	121 M	1188 G	14.6	48.1	49.7
ConvNeXt-Base [19]	122 M	1170 G	16.4	49.1	49.9
NAT-Base	123 M	1137 G	15.6	48.5	49.7

- NAT-Mini outperforms Swin-Tiny, and also comes very close to ConvNeXt-Tiny
- NAT-Tiny outperforms ConvNeXt-Tiny significantly, and is slightly more efficient
- NAT-Small outperforms Swin-Small on single-scale performance, while matching the multi-scale performance.
- NAT-Base performs on-par with Swin-Base, while falling slightly short of ConvNeXt-Base.
- both NAT-Small and NAT-Base bear fewer FLOPs with them compared to their Swin and ConvNeXt counterparts, while their performance is within the same region

SEMANTIC SEGMENTATION

Backbone	# Params	FLOPs	Throughput (FPS)	mIoU	mIoU(ms)
ResNet101 [13]	47 M	-	-	38.8	-
DeiT-S [26, 18]	52 M	1094 G	-	44.0	-
NAT-Mini	50 M	900 G	24.5	45.1	46.4
Swin-Tiny [18]	60 M	946 G	21.3	44.5	45.8
ConvNeXt-T [19]	60 M	939 G	23.3	46.0	46.7
NAT-Tiny	58 M	934 G	21.4	47.1	48.4
Swin-Small [18]	81 M	1040 G	17.0	47.6	49.5
ConvNeXt-Small [19]	82 M	1027 G	19.1	48.7	49.6
NAT-Small	82 M	1010 G	17.9	48.0	49.5
Swin-Base [18]	121 M	1188 G	14.6	48.1	49.7
ConvNeXt-Base [19]	122 M	1170 G	16.4	49.1	49.9
NAT-Base	123 M	1137 G	15.6	48.5	49.7

Note: Swin especially suffers from more FLOPs even beyond the original difference

- Because image resolution input in this task specifically (512 × 512) will not result in feature maps that are divisible by 7×7 (Swin's window size)
- this forces the model to pad input feature maps with zeros to resolve that issue, prior to every attention operation.
- NAT does not require this, as feature maps of any size are compatible.

Hassani et al. Neighborhood Attention Transformer. *2022.*

SALIENCY ANALYSIS

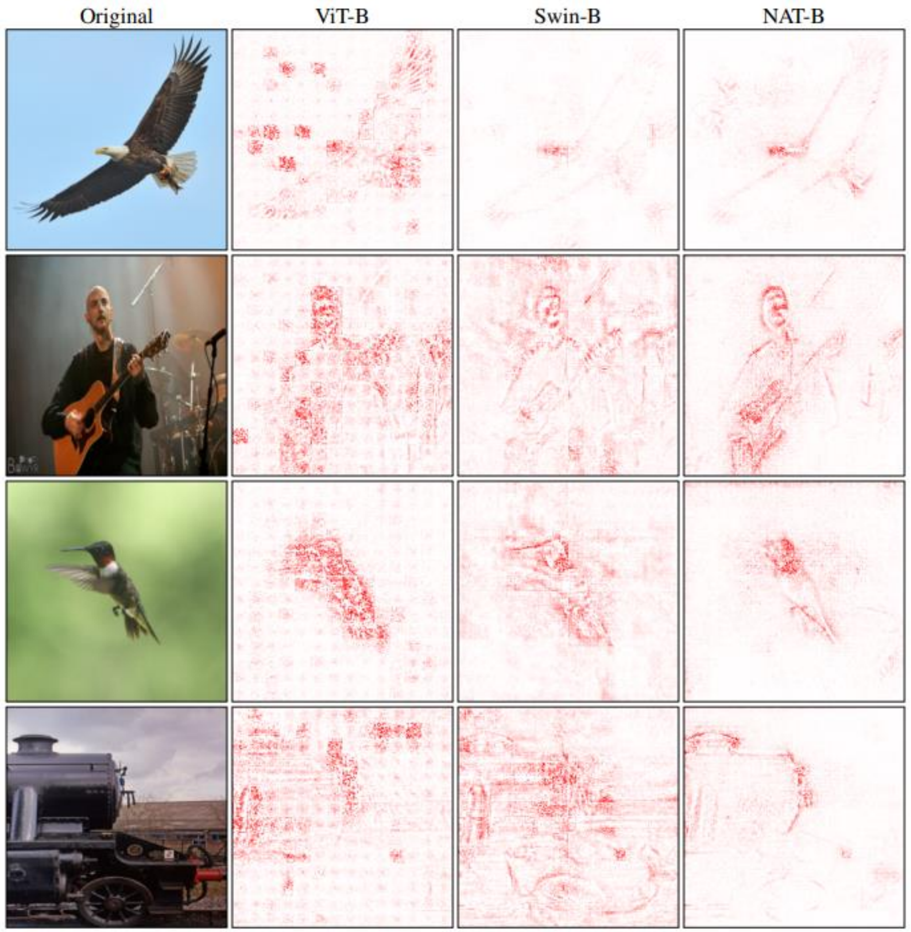


Figure 5: Salient Maps of selected ImageNet validation set images, comparing ViT-Base, Swin-Base, and NAT-Base. The ground truths for these images are: Bald Eagle, Acoustic Guitar, Hummingbird, and Steam Locomotive, respectively.

- all models have relatively good interpretability, though they focus on slightly different areas.
- NAT appears to be slightly better at edge detection
- due to the localized attention mechanism
- as well as the convolutional downsamplers.

Hassani et al. Neighborhood Attention Transformer. *2022.*

OUTLINE

1. Introduction
2. Related Works
3. Methods
4. Experiments
5. Conclusion

CONCLUSION

- Introduced an alternate way (Neighborhood attention) of localizing self attention with respect to the structure of data (computes key-value pairs dynamically for each token)
- NA also provides a more data efficient configuration of models.
- Presented a model NAT that utilizes both the power of attention, as well as the efficiency and inductive biases of convolutions.
- In image classification, NAT outperforms both Swin Transformer and ConvNeXt significantly
- Presented evidence that it can be seamlessly applied to segmentation and object detection tasks, where it also outperforms or competes with existing methods
- Since no operations exist in deep learning libraries that directly replicate NA, and generating key-values for every query is highly inefficient and memory-consuming, implemented a custom CUDA kernel for NA which is open-sourced

Thank You