Transformers

Sanchit Vohra

Presentation Outline

- Sparse Transformers
    - Background
    - Learned attention patterns
    - Architectural details
        - Factorized Self-attention
        - Recomputating attention
        - Scaling to deeper networks
        - Improved Embeddings
    - Results
- An Image is 16x16 words
    - Background
    - Architecture
    - Inductive Bias
    - Hybrid Architecture
    - Results

Sparse Transformers: Background

- Autoregressive sequence generation problem: model high-dimensional data as product of conditional probabilities
- Transformers are powerful but expensive: computation and memory grows quadratically with sequence length because each self-attention layer has a global receptive field
- Main Idea: Sparse factorization of the attention matrix to reduce computation and memory complexity to $O(n\sqrt{n})$

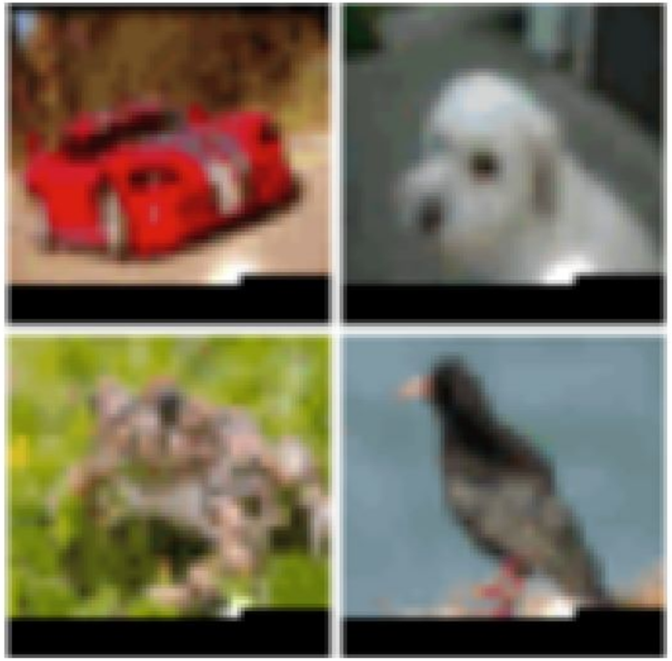
$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i | x_1, ..., x_{i-1}; \theta)$$

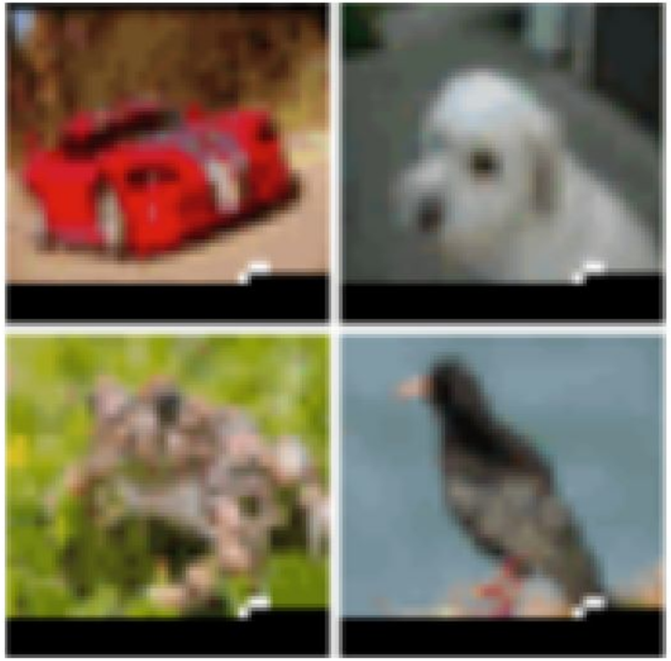
Sparse Transformers: Attention Visualization

- Analyze learned attention patterns for 128-layer dense transformer network on CIFAR-10
- Early layers: Attention pattern resembles convolution
- Layers 19-20: Row and column attention
- Data dependent global access patterns
- Layers 64-128: extremely sparse attention
- Most attention patterns are sparse!

Sparse Transformers: Attention Visualization

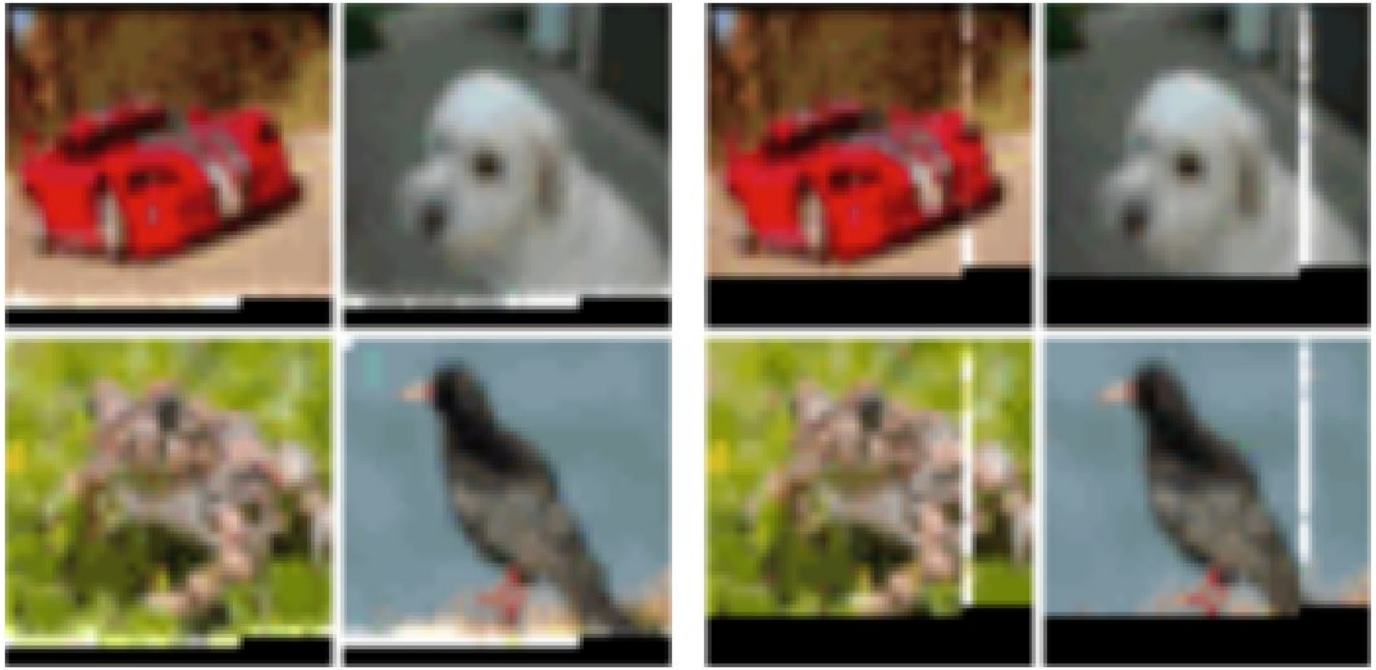
Early layers: Attention pattern resembles convolution





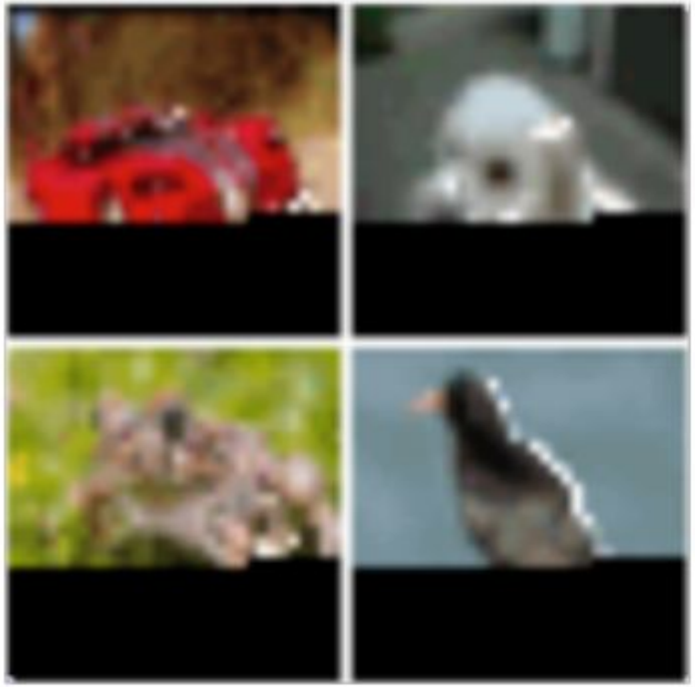
Sparse Transformers: Attention Visualization

Layers 19-20: Row and column attention



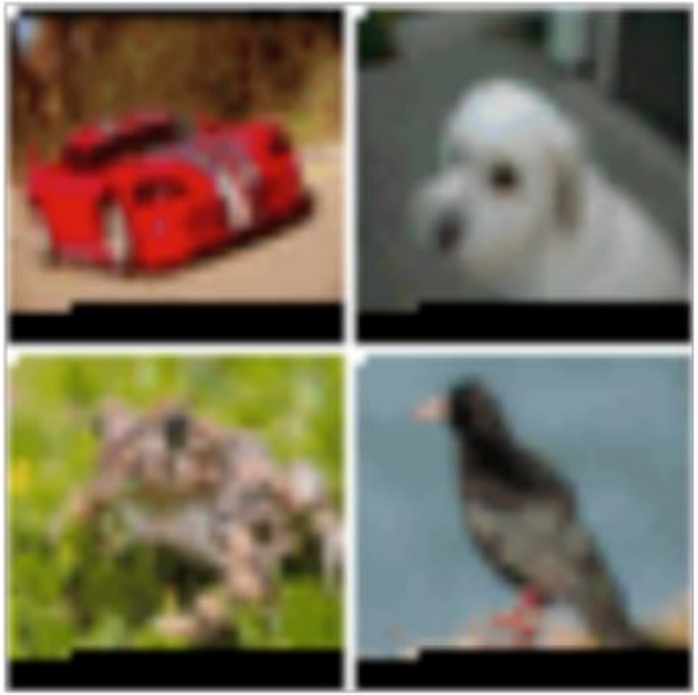
Sparse Transformers: Attention Visualization

Data dependent global access patterns



Sparse Transformers: Attention Visualization

Layers 64-128: extremely sparse attention



Sparse Transformers: Factorized Self-Attention

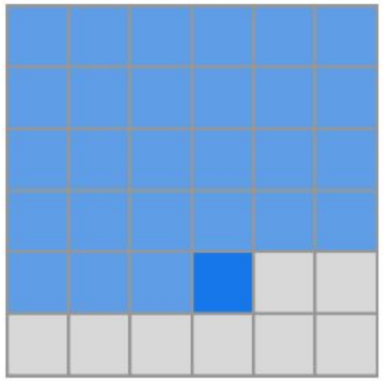
- S is connectivity pattern where S_i represents input indices which i'th output vector attends
- For vanilla transformer, each $S_i = \{j : j \leq i\}$

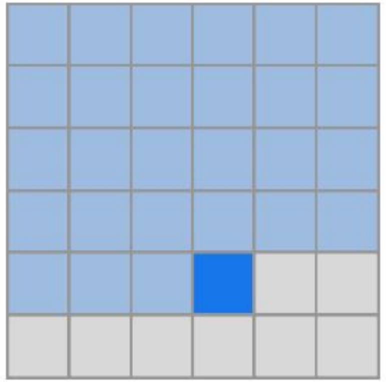
$$\text{Attend}(X, S) = \Big(a(\mathbf{x}_i, S_i) \Big)_{i \in \{1, \ldots, n\}}$$

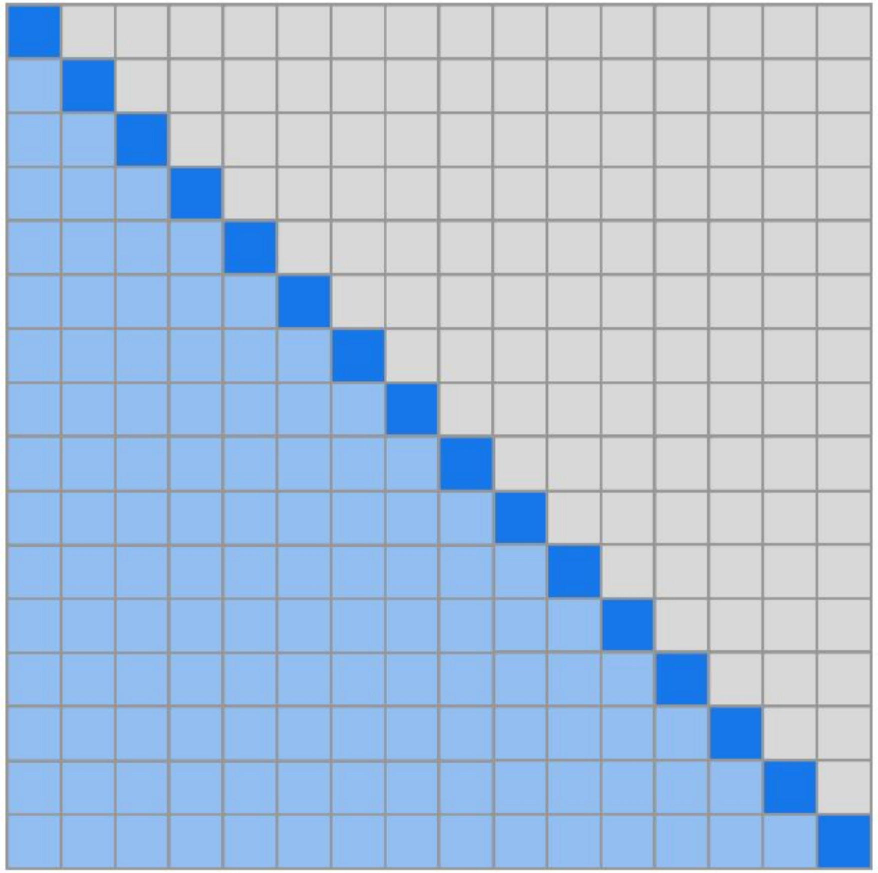
$$a(\mathbf{x}_i, S_i) = \text{softmax}\left(\frac{(W_q \mathbf{x}_i) K_{S_i}^T}{\sqrt{d}} \right) V_{S_i}$$

$$K_{S_i} = \Big(W_k \mathbf{x}_j \Big)_{j \in S_i} \quad V_{S_i} = \Big(W_v \mathbf{x}_j \Big)_{j \in S_i}$$

Sparse Transformers: Factorized Self-Attention







- Top images indicate which positions each attention head receive as input
- Right image indicates connectivity matrix for these attention heads

Sparse Transformers: Factorized Self-Attention

- Factorized self attention has p separate heads
- Each head defines a subset of indices $A_i^{(m)} \subset \{j : j \leq i\}$
- We chose efficient choices of of A so that $|A_i^{(m)}| \propto \sqrt[p]{n}.$

$$\text{Attend}(X, S) = \Big(a(\mathbf{x}_i, S_i) \Big)_{i \in \{1,\ldots,n\}}$$

$$a(\mathbf{x}_i, S_i) = \text{softmax}\left(\frac{(W_q \mathbf{x}_i) K_{S_i}^T}{\sqrt{d}} \right) V_{S_i}$$

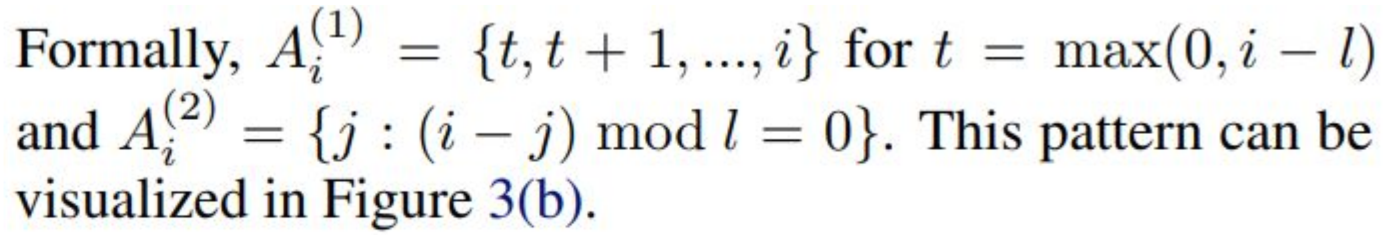
$$K_{S_i} = \Big(W_k \mathbf{x}_j \Big)_{j \in S_i} \quad V_{S_i} = \Big(W_v \mathbf{x}_j \Big)_{j \in S_i}$$

Sparse Transformers: Factorized Self-Attention

- Valid choices of A are where all input positions are connected to all future output positions across the p steps of attention
- This criteria along with ones on previous slide, enable sparse, factorized attention units that can propagate any input to any output while reducing effective computation to $O(n\sqrt[p]{n})$
- Next slides we will see some 2-dimensional (p=2) factorized attention examples which can easily be extended to higher dimensions

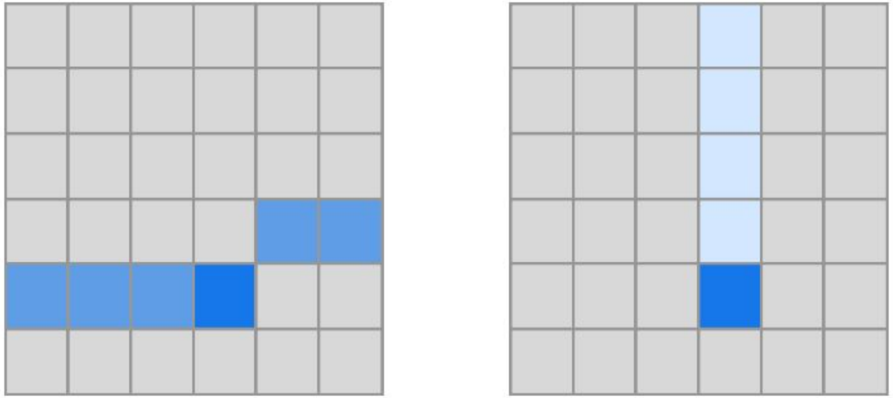
Sparse Transformers: Strided Attention Pattern

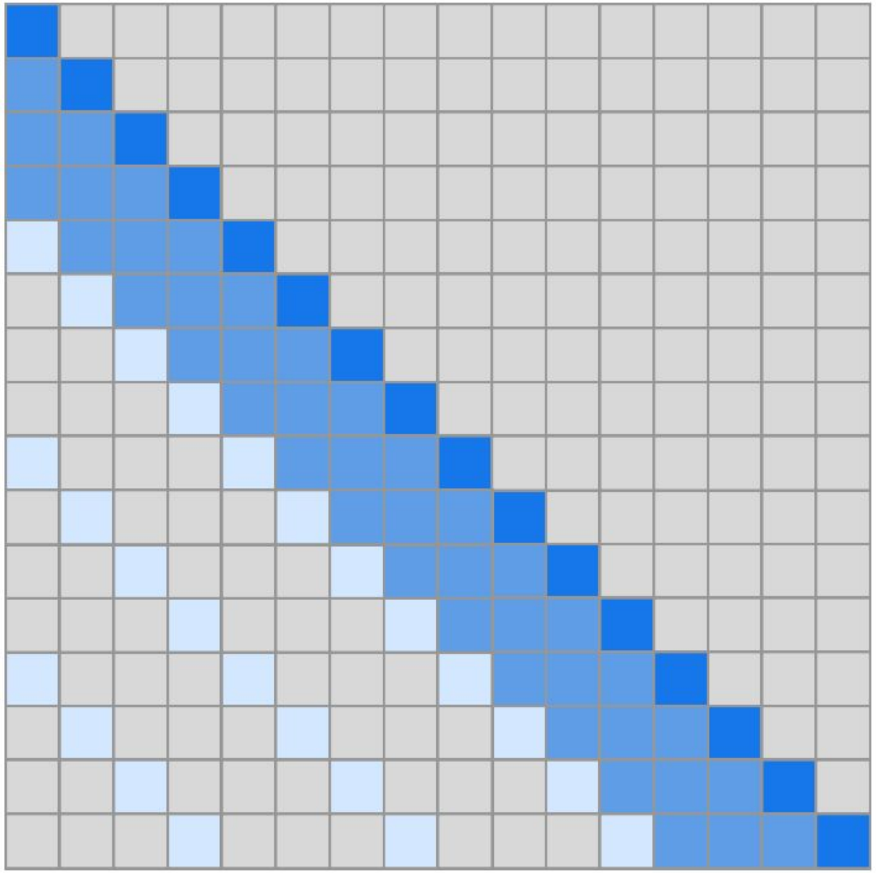
- l = stride and chosen to be close to $\sqrt{n}$
- p=1 head attends to previous l locations
- p=2 head attends to every lth location
- This formulation works well if data naturally has a structure that aligns with the stride, like images or some types of music
- For data without a periodic structure like text, this formulation fails to properly route information



Formally, $A_i^{(1)} = \{t, t+1, ..., i\}$ for $t = \max(0, i - l)$ and $A_i^{(2)} = \{j : (i - j) \bmod l = 0\}$. This pattern can be visualized in Figure 3(b).

Sparse Transformers: Strided Attention Pattern

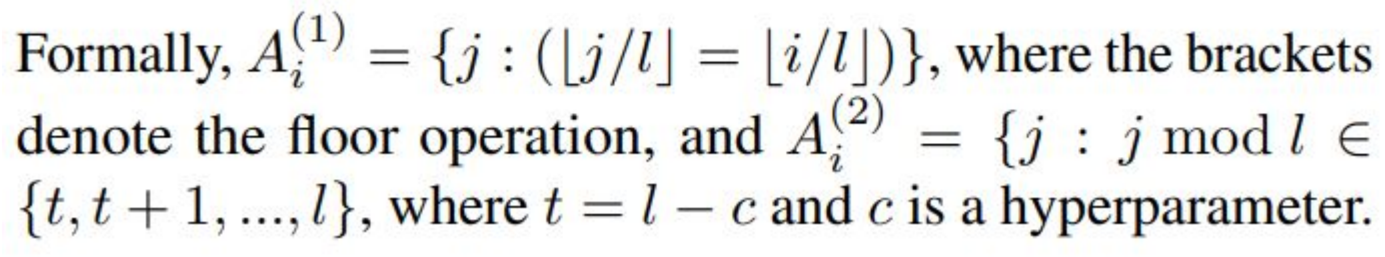




- Top images indicate which positions each attention head receive as input
- Right image indicates connectivity matrix for these attention heads

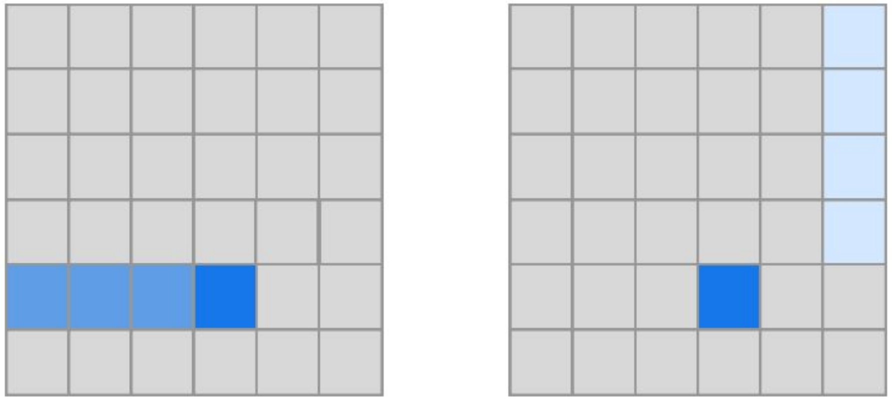
Sparse Transformers: Fixed Attention Pattern

- For data without a periodic structure like text, use fixed pattern instead

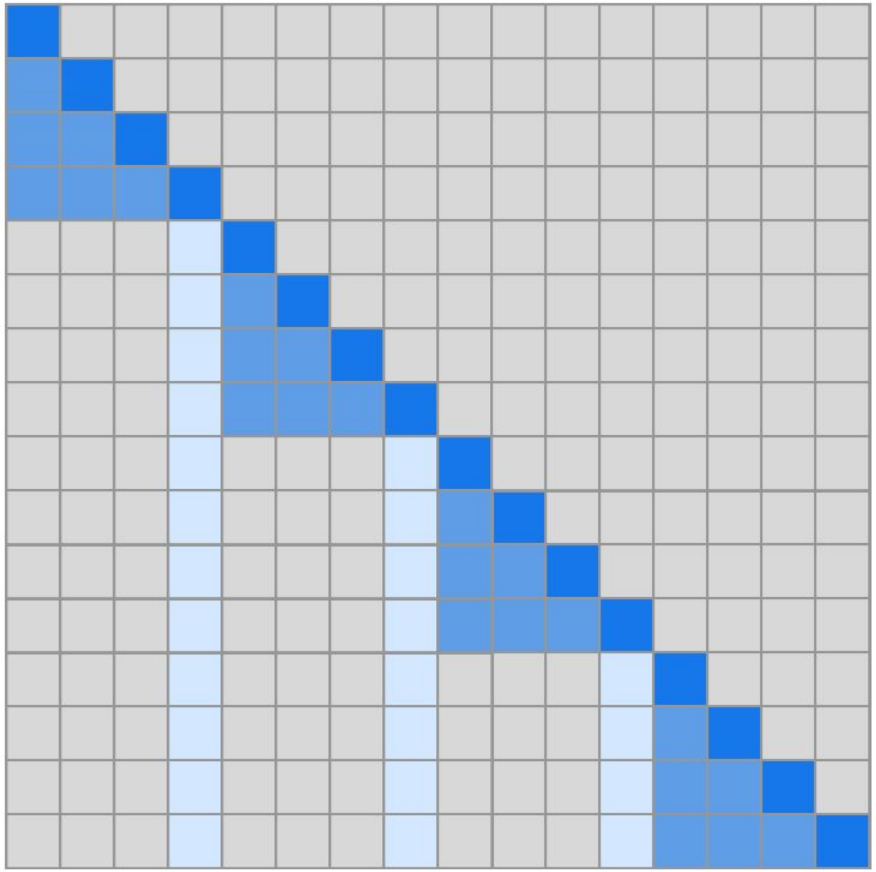


Formally, $A_i^{(1)} = \{j : (\lfloor j/l \rfloor = \lfloor i/l \rfloor)\}$, where the brackets denote the floor operation, and $A_i^{(2)} = \{j : j \bmod l \in \{t, t+1, ..., l\}$, where $t = l - c$ and c is a hyperparameter.

Sparse Transformers: Fixed Attention Pattern



- Top images indicate which positions each attention head receive as input
- Right image indicates connectivity matrix for these attention heads



Sparse Transformers: Factorized Self-Attention

- Technique 1: one attention type per residual block interleaved sequentially

$$\text{attention}(X) = W_p \cdot \text{attend}(X, A^{(r \bmod p)})$$

- Technique 2: single self attention head that has pixels of all factorized heads

$$\text{attention}(X) = W_p \cdot \text{attend}(X, \bigcup_{m=1}^{p} A^{(m)})$$

- Technique 3: use multi-head attention with the separate, merged or interleaved heads

$$\text{attention}(X) = W_p \Big(\text{attend}(X, A)^{(i)}\Big)_{i \in \{1, \ldots, n_h\}}$$

Sparse Transformers: Recompute Attention

- When backpropagating gradients, results from forward pass are stored in memory
- However, for sparse attention, memory usage >>> computation cost especially when sequences become long
- Recompute forward layers to enable networks with hundred of layers and sequence lengths of 16384 (128 x 128)

Sparse Transformers: Scaling to deep networks

- Transformers are difficult to train with many layers; change architecture to enable deeper transformers

$$H_0 = \text{embed}(X, W_e)$$

$$H_k = H_{k-1} + \text{resblock}(H_{k-1})$$

$$y = \text{softmax}(\text{norm}(H_N)W_{out})$$

$$a(H) = \text{dropout}(\text{attention}(\text{norm}(H)))$$

$$b(H) = \text{dropout}(\text{ff}(\text{norm}(H + a(H))))$$

$$\text{resblock}(H) = a(H) + b(H)$$

Sparse Transformers: Scaling to deep networks

- Add learned embeddings which either encoded structure (data embeddings) or factorized attention patterns (attention embeddings)
- For images, data embeddings work better and for text, audio two-dimensional attention embeddings work better

$$\text{embed}(X, W_e) = \left(\mathbf{x}_i W_e + \sum_{j=1}^{n_{emb}} \mathbf{o}_i^{(j)} W_j \right)_{\mathbf{x}_i \in X}$$

- o is encoded position in data/attention

Sparse Transformers: Recompute Attention

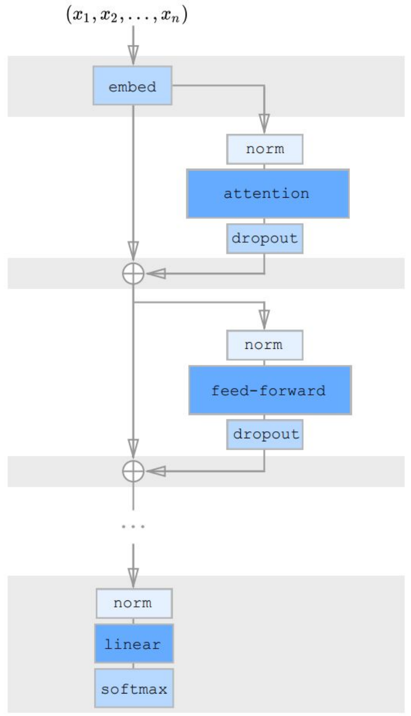
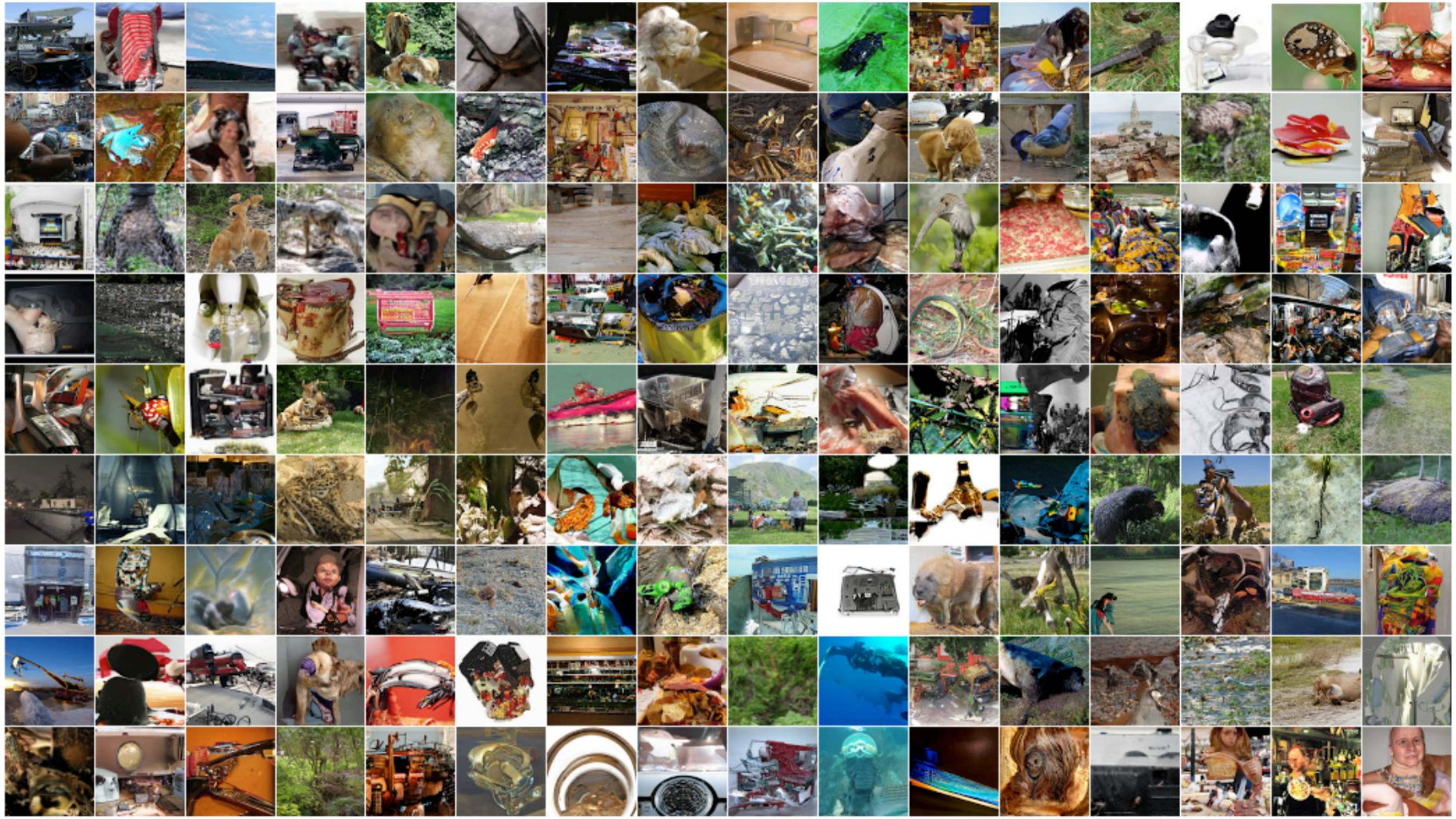


Figure 4. Diagram depicting one residual block of the Sparse Transformer. The shaded background indicates tensors which are *checkpointed* (Chen et al., 2016) and stored in GPU memory. The other tensors, including the attention weights and feedforward network activations, are recomputed during the calculation of gradients, reducing memory usage substantially.

Sparse Transformers: Results



Sparse Transformers: Results

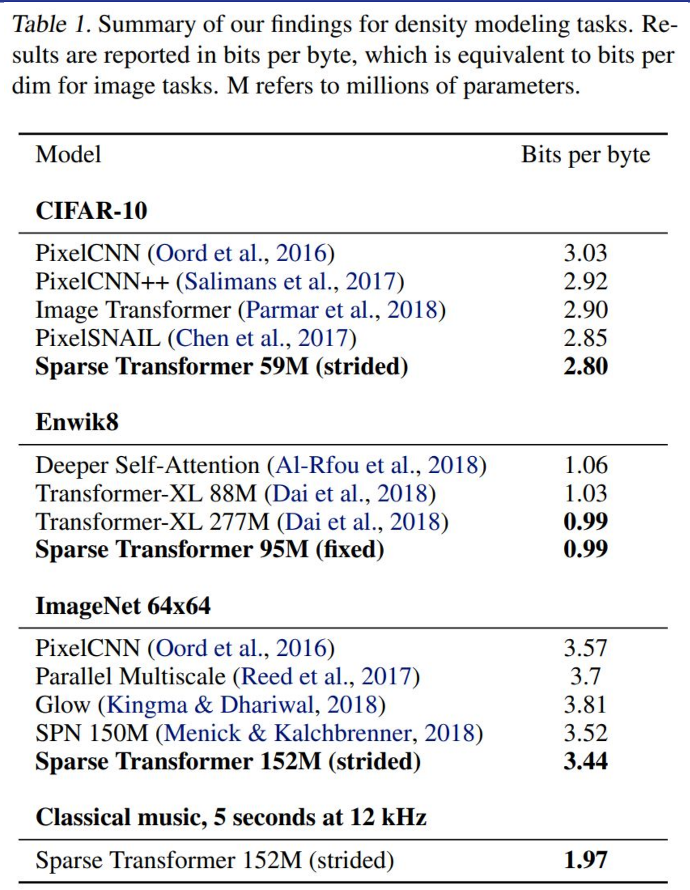


Table 1. Summary of our findings for density modeling tasks. Results are reported in bits per byte, which is equivalent to bits per dim for image tasks. M refers to millions of parameters.

Model	Bits per byte
CIFAR-10	
PixelCNN (Oord et al., 2016)	3.03
PixelCNN++ (Salimans et al., 2017)	2.92
Image Transformer (Parmar et al., 2018)	2.90
PixelSNAIL (Chen et al., 2017)	2.85
Sparse Transformer 59M (strided)	**2.80**
Enwik8	
Deeper Self-Attention (Al-Rfou et al., 2018)	1.06
Transformer-XL 88M (Dai et al., 2018)	1.03
Transformer-XL 277M (Dai et al., 2018)	**0.99**
Sparse Transformer 95M (fixed)	**0.99**
ImageNet 64x64	
PixelCNN (Oord et al., 2016)	3.57
Parallel Multiscale (Reed et al., 2017)	3.7
Glow (Kingma & Dhariwal, 2018)	3.81
SPN 150M (Menick & Kalchbrenner, 2018)	3.52
Sparse Transformer 152M (strided)	**3.44**
Classical music, 5 seconds at 12 kHz	
Sparse Transformer 152M (strided)	**1.97**

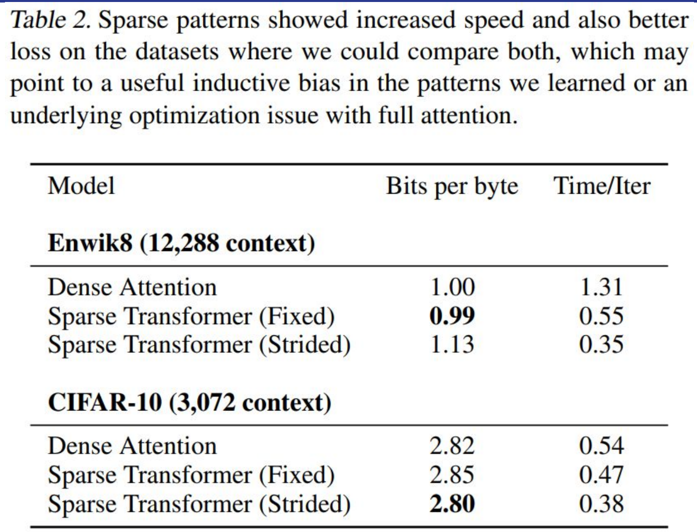


Table 2. Sparse patterns showed increased speed and also better loss on the datasets where we could compare both, which may point to a useful inductive bias in the patterns we learned or an underlying optimization issue with full attention.

Model	Bits per byte	Time/Iter
Enwik8 (12,288 context)		
Dense Attention	1.00	1.31
Sparse Transformer (Fixed)	**0.99**	0.55
Sparse Transformer (Strided)	1.13	0.35
CIFAR-10 (3,072 context)		
Dense Attention	2.82	0.54
Sparse Transformer (Fixed)	2.85	0.47
Sparse Transformer (Strided)	**2.80**	0.38

An image is 16x16 words: Background

- Despite success of transformer in NLP (BERT, GPT), in computer vision, convolutional patterns remain dominant
- Naively applying pixel-to-pixel attention scales quadratically and becomes unrealistic for reasonable image sizes
- Models that replace convolutions with self-attention (Sparse Transformers) are hard to scale on hardware acceleration due to use of specialized attention patterns
- Apply standard transformer encoder on images directly with the fewest possible changes

An image is 16x16 words: Background

- When trained on mid-size datasets, the transformer model’s performance is a few points lower than ResNets of comparable size
- This is expected since transformers lack inductive bias of convolutions: translation equivariance and locality
- However, on large datasets (14M-300M images), the transformer overcomes these inductive biases and achieves SOTA performance on classification task

An image is 16x16 words: Architecture

- Divide image into flattened patches of size (P, P) -> P^2
- Linearly project flattened patched into D dimensions
- Add learned 1-D positional encoding to inputs
- Append class embedding to the input, whose output is the classifier input
- Attach MLP classifier and classify image

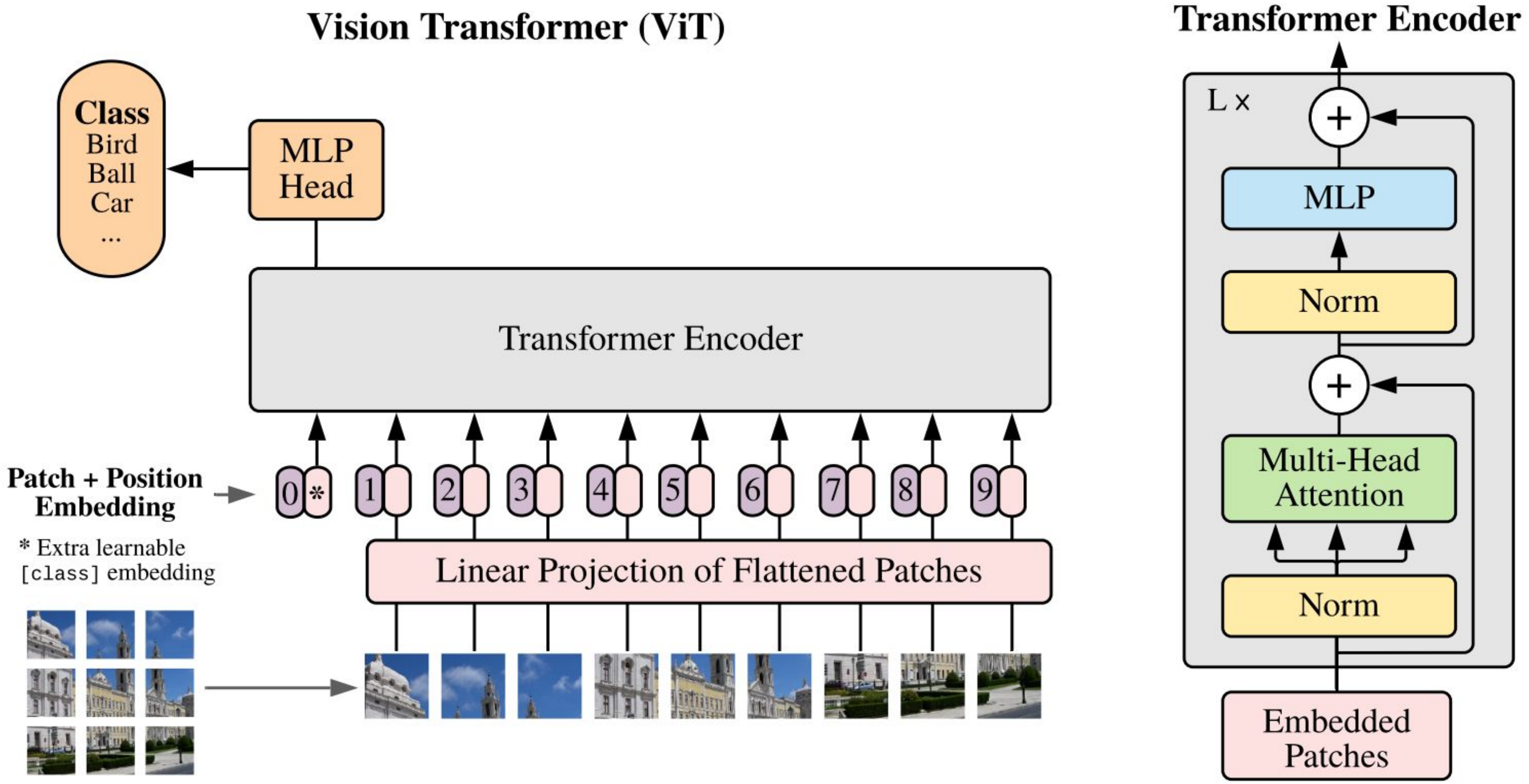
$$\mathbf{z}_0 = [\mathbf{x}_\text{class}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \cdots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2\cdot C)\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(N+1)\times D}$$

$$\mathbf{z'}_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z'}_\ell)) + \mathbf{z'}_\ell, \qquad \ell = 1 \ldots L$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$

An image is 16x16 words: Architecture



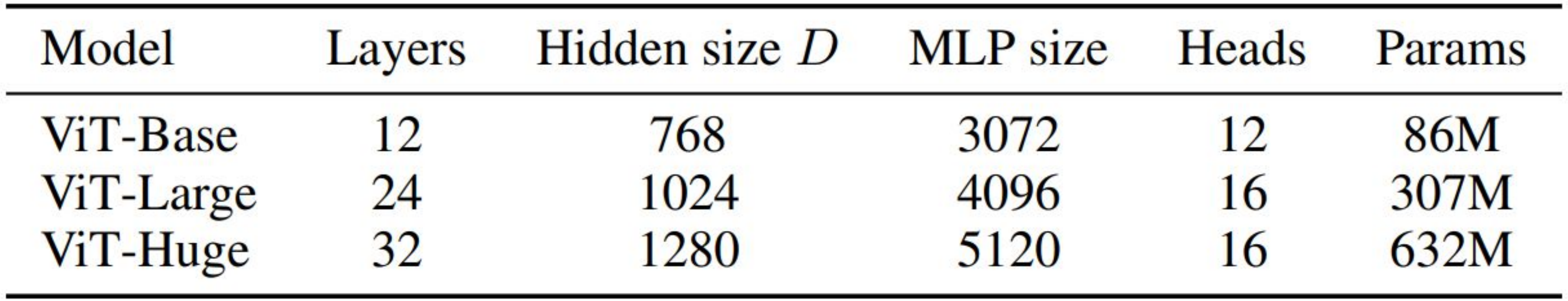
An image is 16x16 words: Inductive Bias

- As mentioned Earlier, ViT has much less image-specific inductive bias when compared to CNN
- Convolutions exploit locality, two-dimensional neighborhood structure, and translation equivariance
- In ViT, multi-head self attention layers are global!
- Two-dimensional structure is only used during the beginning when image is split into patches (P, P)
- Even positional embeddings are learned in 1-D

An image is 16x16 words: Hybrid Architecture

- Instead of using patches from image, use patches from output of CNN feature map
- Linearly project CNN patches using learned embedding + learned 1-D positional embedding
- Can use spatial 1x1 patches by flattening features
- Everything else remains the same

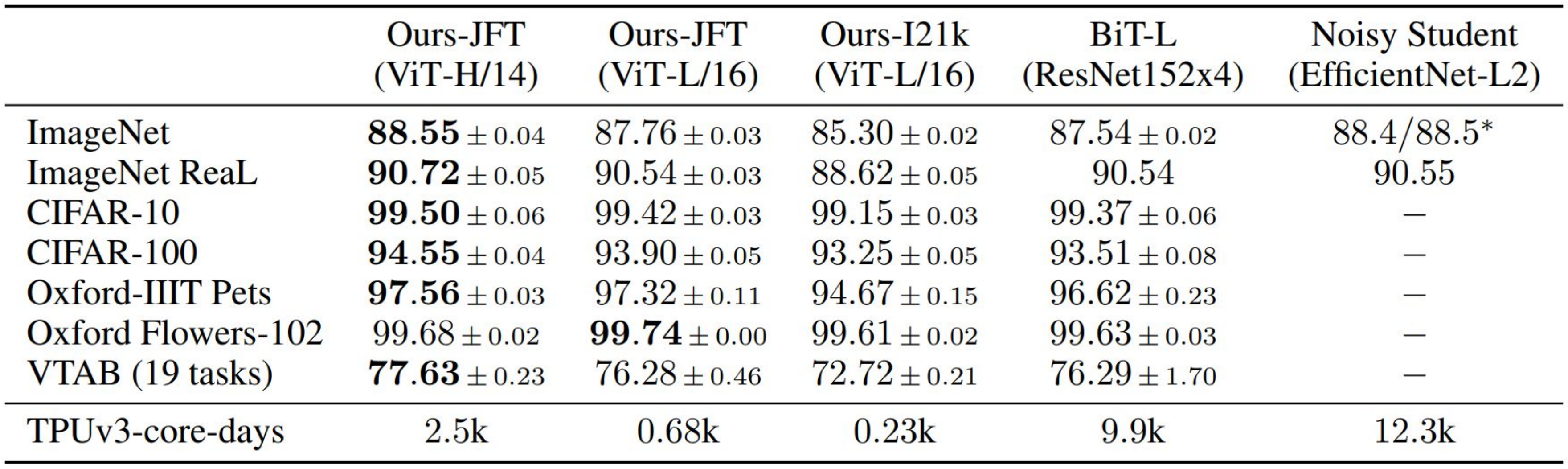
An image is 16x16 words: Results



Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

Table 1: Details of Vision Transformer model variants.

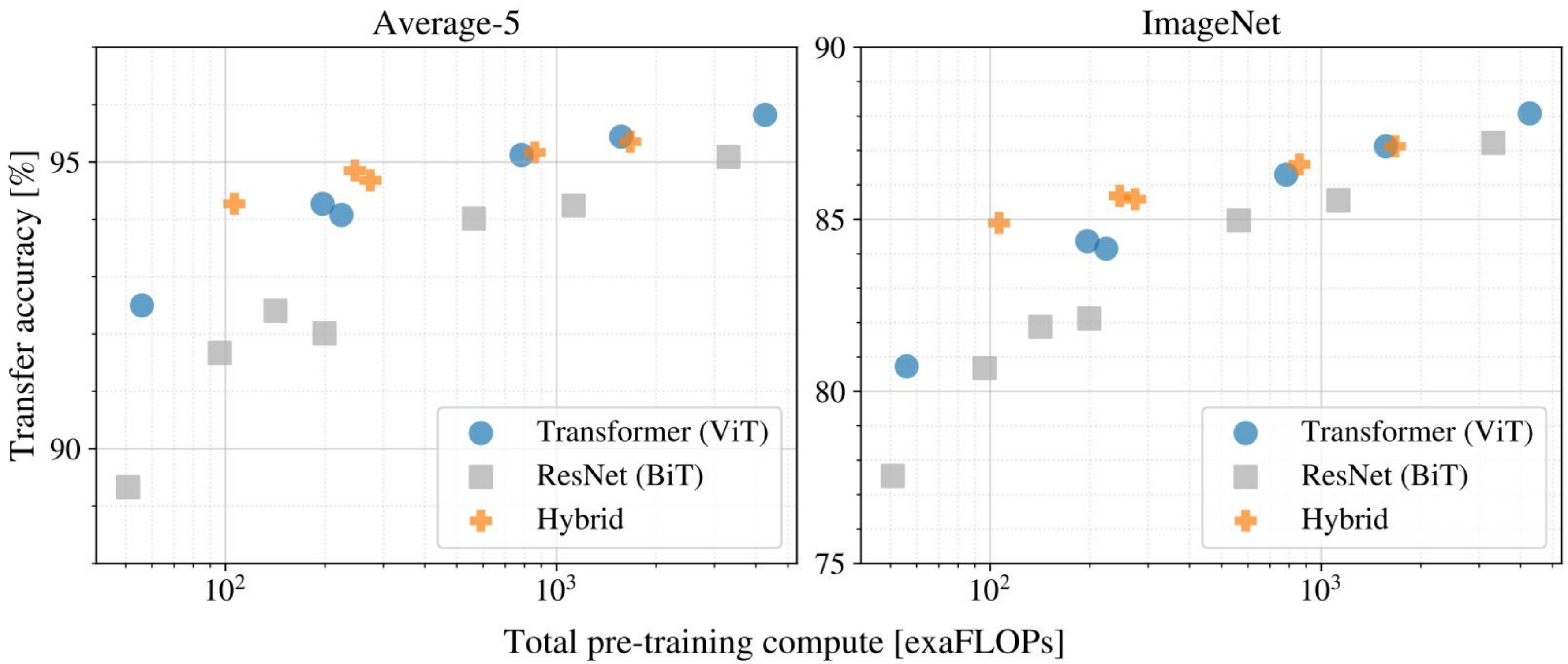
An image is 16x16 words: Results



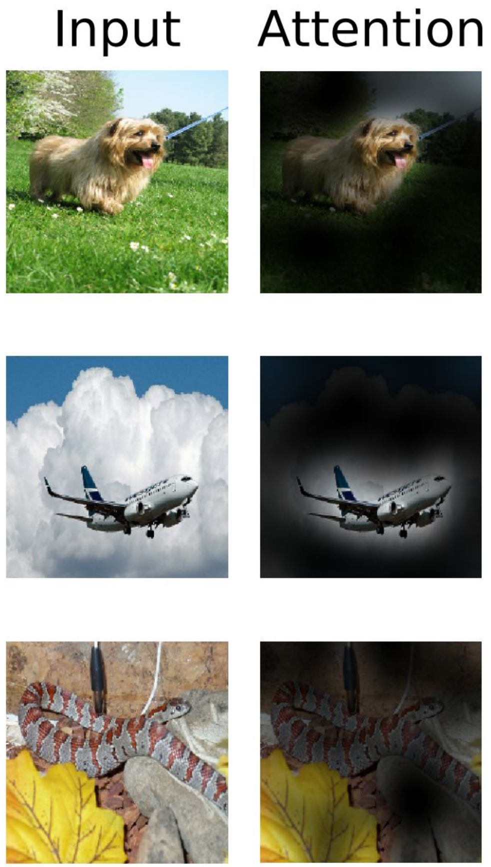
	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

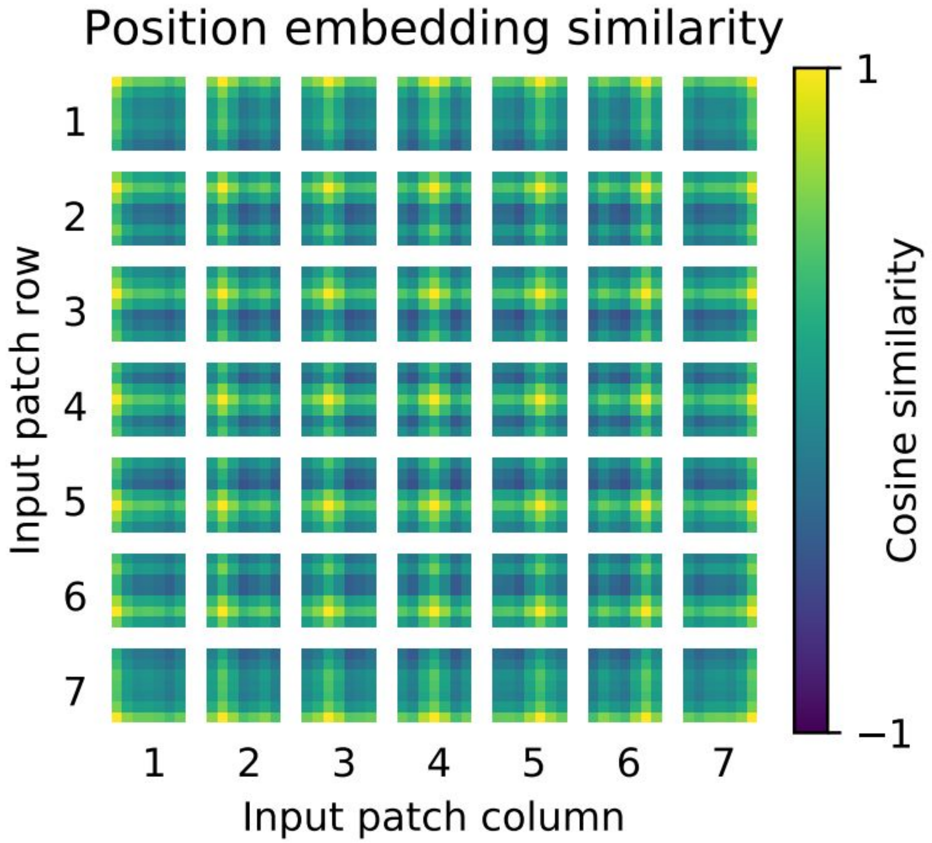
Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

An image is 16x16 words: Results



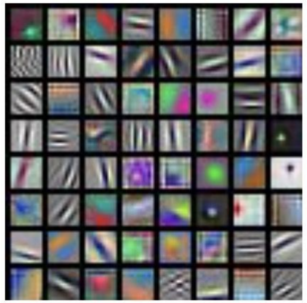
An image is 16x16 words: Results





An image is 16x16 words: Results

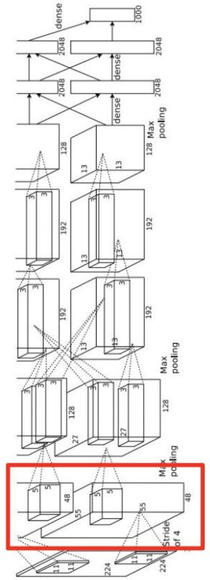
First Layer: Visualize Filters

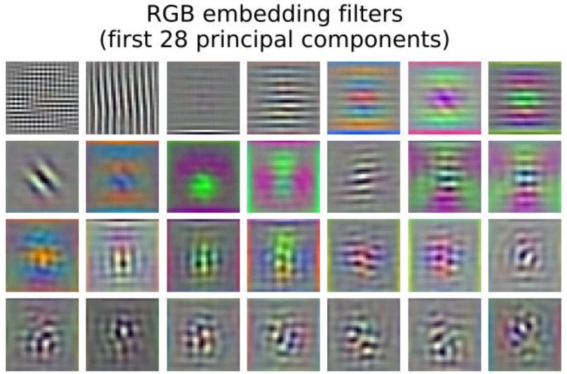

AlexNet:
64 x 3 x 11 x 11

ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7

DenseNet-121:
64 x 3 x 7 x 7





Krizhevsky, "One weird trick for parallelizing convolutional neural networks", arXiv 2014
He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected Convolutional Networks", CVPR 2017

References

- Dosovitskiy, Alexey, et al. “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale.” ArXiv:2010.11929 [Cs], June 2021. arXiv.org, http://arxiv.org/abs/2010.11929
- Child, Rewon, et al. “Generating Long Sequences with Sparse Transformers.” ArXiv:1904.10509 [Cs, Stat], Apr. 2019. arXiv.org, http://arxiv.org/abs/1904.10509