

ML FOR SCIENCE · LECTURE 9

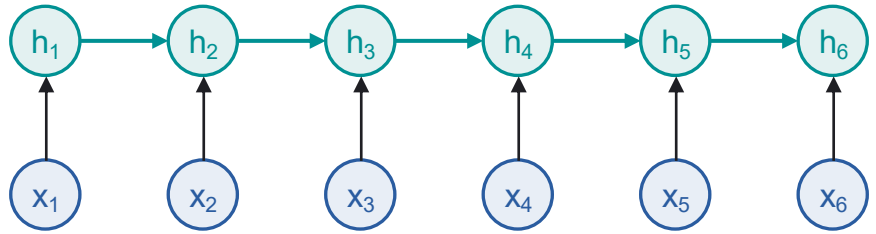
Attention and Transformers

Aparna Chandramowlishwaran · EECS 195/221 · Fall 2026

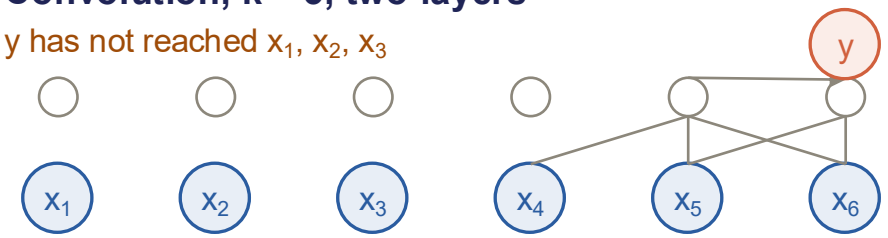
Limits of recurrence and convolution

- **Recall**: RNN carries the past in a hidden state h_n of fixed size
  - Information from k steps back passes through k updates of h
  - Steps run sequentially during training and prediction
- **Recall**: Convolution with kernel size k sees a small neighborhood
  - The receptive field grows by k − 1 cells per layer
  - Cells L apart interact after about L/(k − 1) layers
- **Incompressible flow**: pressure solves a Poisson equation. A change anywhere changes p everywhere at once
- **Attention**: each output reads every input directly, with weights computed from the data

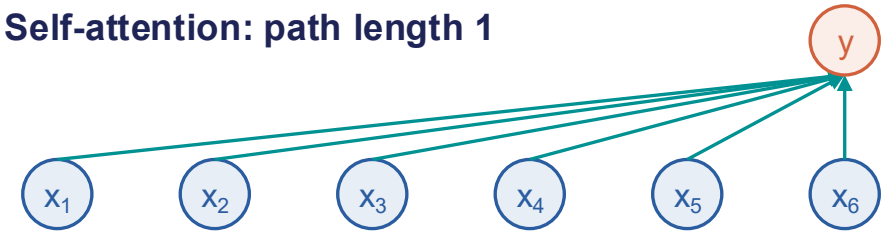
Recurrent: path length up to N



Convolution, k = 3, two layers



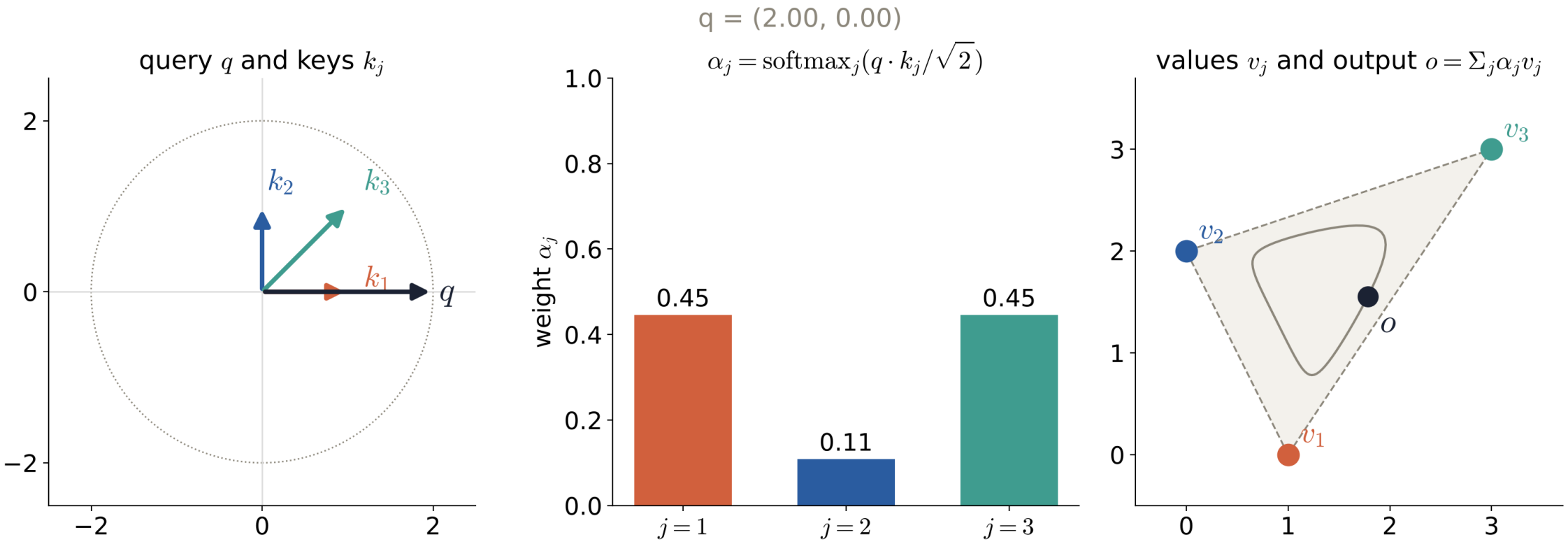
Self-attention: path length 1



Attention connects every pair of inputs in one step

N: sequence length or number of cells. x: inputs, h: hidden states, y: one output. Vaswani et al., NeurIPS 2017

Attention as a weighted lookup



- A **query** q is compared with the **key** k_j of every input. The softmax turns the scores into weights $\alpha_j \geq 0$ that sum to 1
- The output o is the weighted average of the **values** v_j, so it stays inside the triangle of values

Bahdanau, Cho & Bengio, ICLR 2015

Scaled dot-product attention

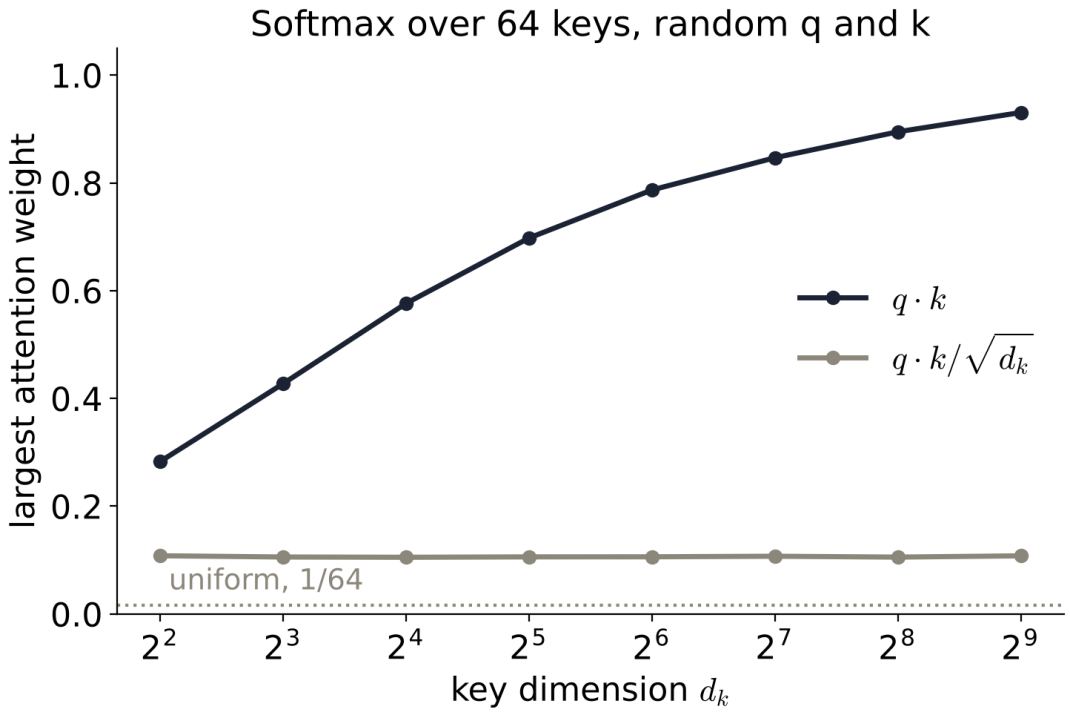
- Stack the N queries, keys and values as the rows of **Q**, **K** and **V**:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

- $\boldsymbol{Q}\boldsymbol{K}^\top$ is N x N: one score per pair (i, j). Softmax normalizes each row
- Why $\sqrt{d_k}$: for entries with mean 0 and variance 1,

$$\boldsymbol{q}\cdot\boldsymbol{k} = \sum_{m=1}^{d_k} \boldsymbol{q}_m \boldsymbol{k}_m, \qquad \text{Var}(\boldsymbol{q}\cdot\boldsymbol{k}) = d_k$$

- Large scores make the softmax pick one key, and its gradient vanishes
- Without scaling, the largest weight approaches 1 as d_k increases. With scaling, it stays the same



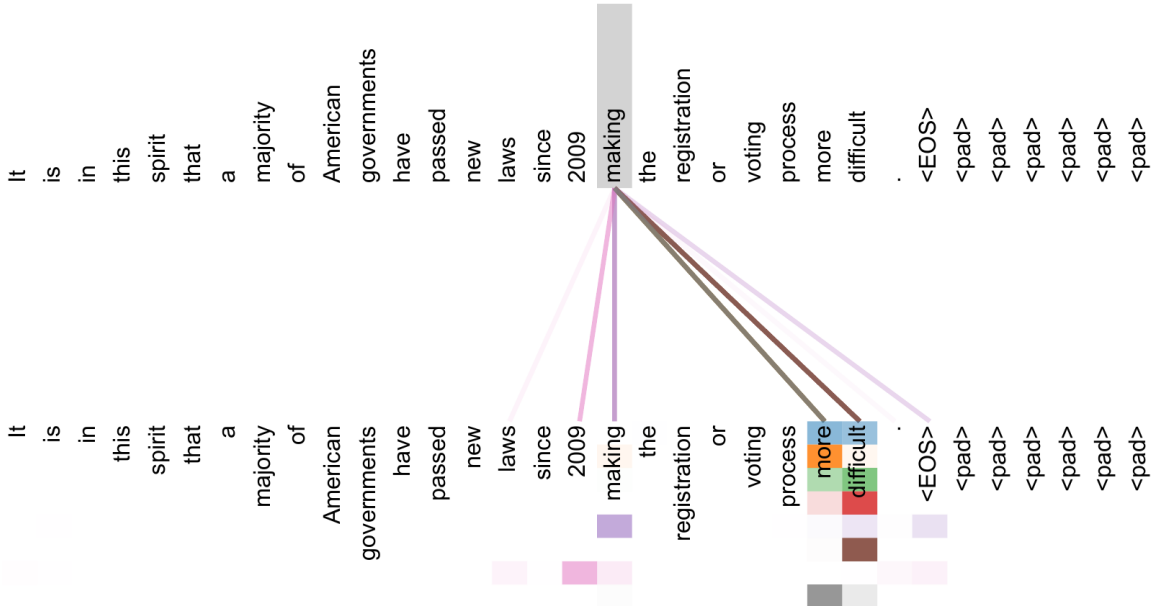
Q: N x d_k, **K**: N x d_k, **V**: N x d_v. Vaswani et al., NeurIPS 2017

Self-attention

- Queries, keys and values all come from the same tokens X (N x d):

$$Q = XW_Q, \qquad K = XW_K, \qquad V = XW_V$$

- W_Q, W_K, W_V: learned d x d_k matrices, shared by all tokens
- Convolution applies the same kernel weights to every input. Attention weights α_{ij} are computed from the input and **different for every input**
- Every token attends to every other token
- The number of parameters does not depend on N
- Figure: translation model, layer 5. The word "making" attends to "more difficult", six words later



X: token matrix, one row per token. N: number of tokens, d: token dimension. Figure: Vaswani et al., NeurIPS 2017, (colors: attention heads)

Multi-head attention

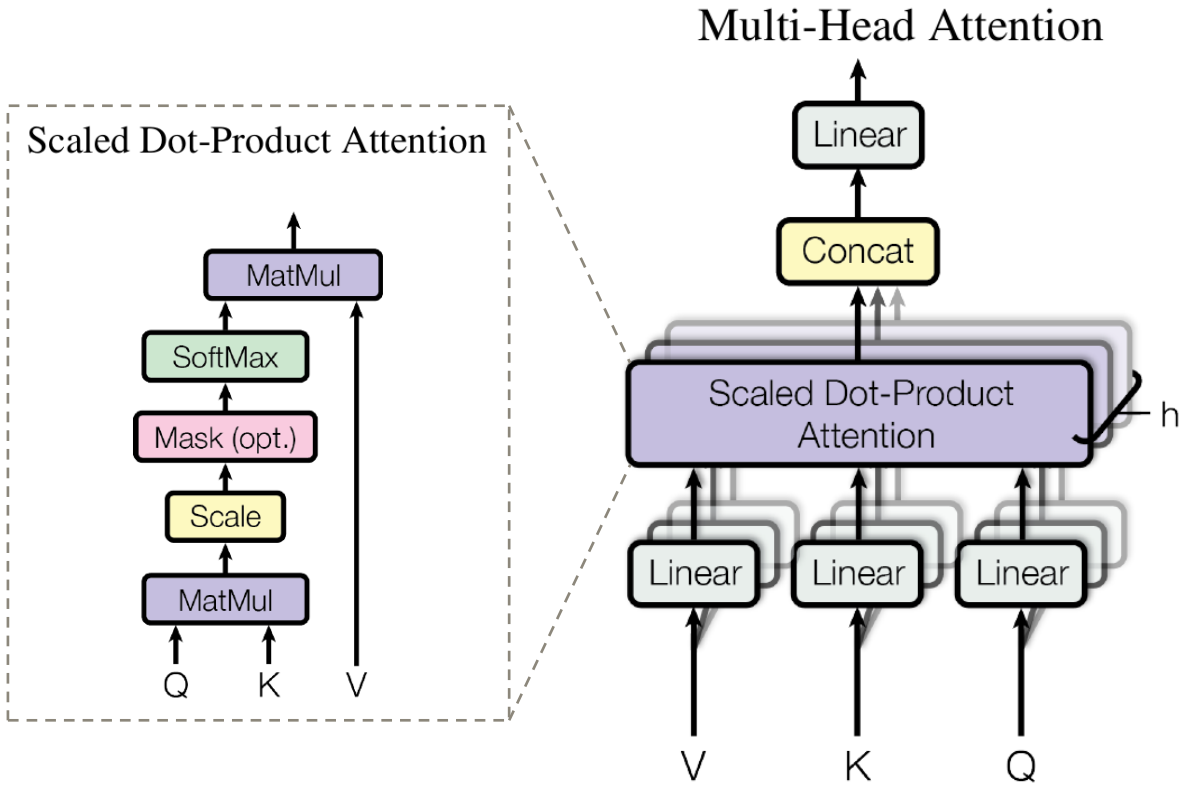
- H heads. Head h has its own projections $\mathbf{W_Q^{(h)}}$, $\mathbf{W_K^{(h)}}$, $\mathbf{W_V^{(h)}}$, each d x d/H:

$$\text{head}_h = \text{Attention}\left(X\boldsymbol{W}_{\boldsymbol{Q}}^{(h)}, X\boldsymbol{W}_{\boldsymbol{K}}^{(h)}, X\boldsymbol{W}_{\boldsymbol{V}}^{(h)}\right)$$

- W_O (d x d) mixes the concatenated heads:

$$\text{MHA}(X) = \left[\text{head}_1, \ldots, \text{head}_H\right] W_O$$

- Each head computes its own weights, so heads can attend to different inputs
- Same number of parameters and nearly the same cost as one head of size d
- Vaswani et al.: d = 512, H = 8 heads, d_k = 64



H: number of heads, W_O: output projection. Figures: Vaswani et al., NeurIPS 2017

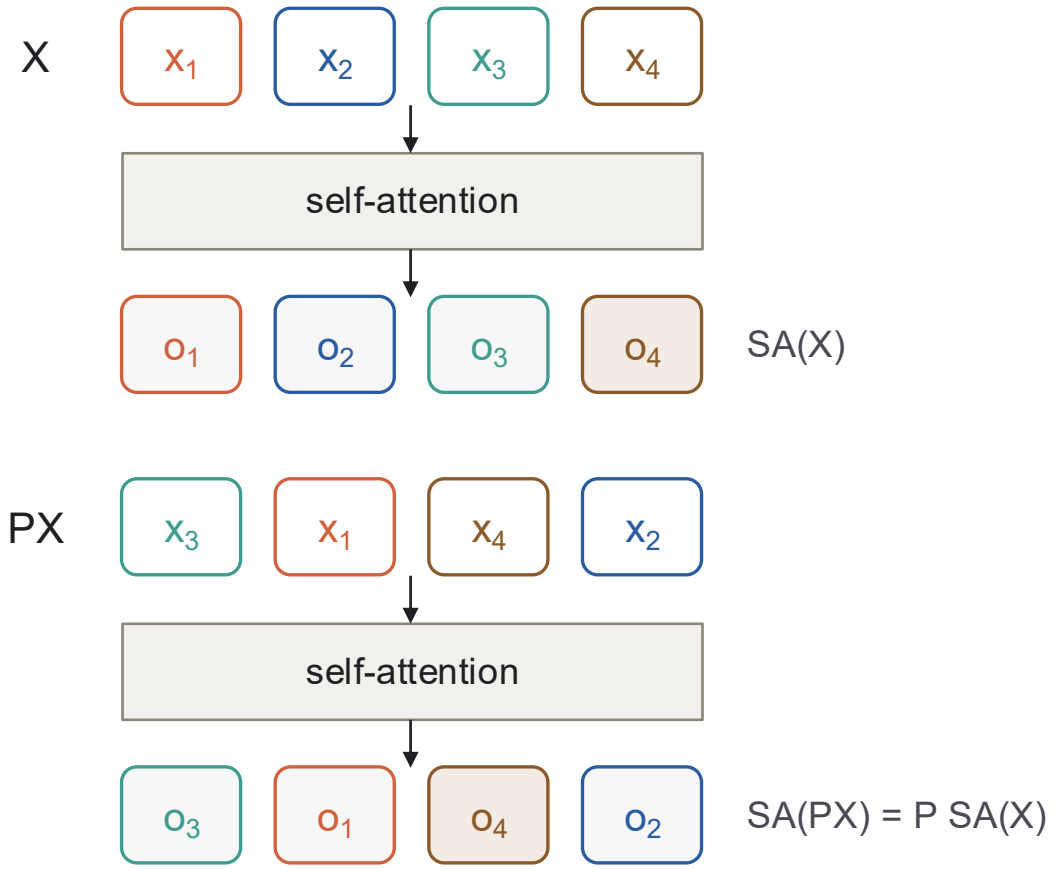
Self-attention is permutation equivariant

- **Recall**: **f** is equivariant to **g** if f(g x) = g f(x)
- Here **g** reorders the tokens. With **P** a permutation matrix:

$$\mathrm{SA}(PX) = P\,\mathrm{SA}(X)$$

- Every token uses the same $\mathbf{W_Q}$, $\mathbf{W_K}$, $\mathbf{W_V}$, and the softmax sums over all **j** in any order

- **Self-attention alone cannot tell which token is first, or which patch touches the heater**
- Add a position code **E** to each token



Without position information, a transformer treats its input as an unordered set

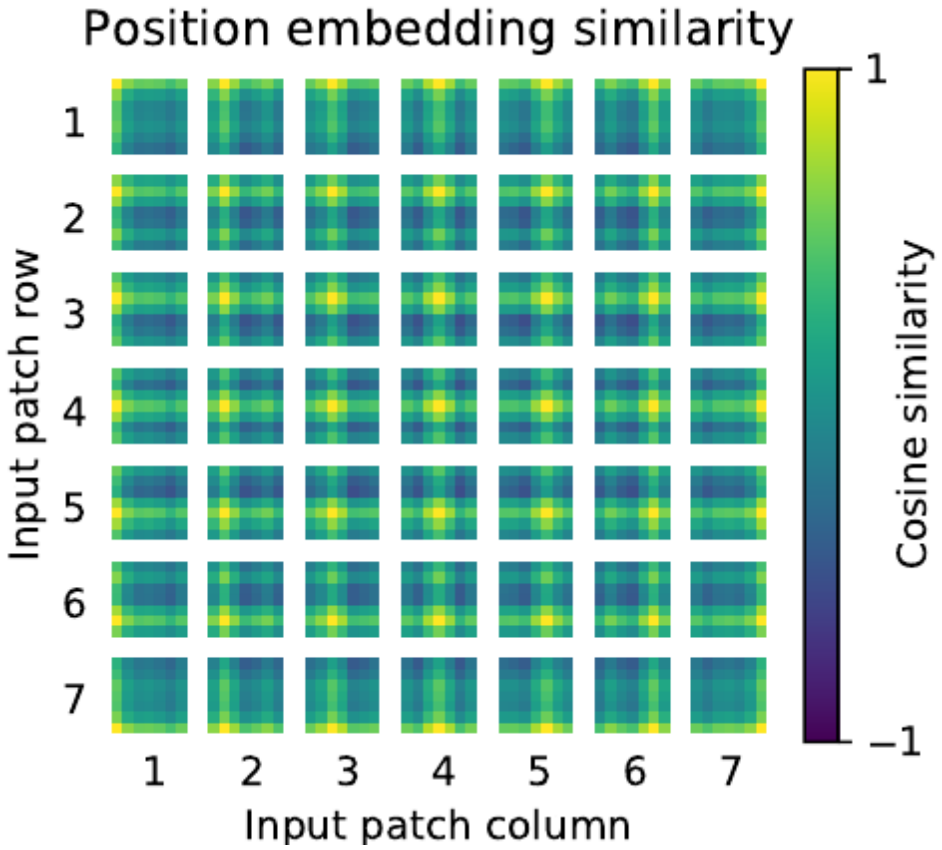
P: permutation matrix. SA: self-attention.

Position encodings

- Add a vector $\mathbf{E_n}$ to token n: $x_n + E_n$
- **Sinusoidal** (Vaswani et al. 2017), wavelengths from 2π to 10000 · 2π:

$$E_{n,2i} = \sin\left(\frac{n}{10000^{2i/d}}\right), \qquad E_{n,2i+1} = \cos\left(\frac{n}{10000^{2i/d}}\right)$$

- **Learned**: one trained vector per position. ViT (right): model learns the 2D grid structure from data
- **Relative**: the score depends on the offset i − j. T5 learns a bias per offset; RoPE rotates **q** and **k** by an angle proportional to position (Su et al. 2024)
- Fields: give the coordinates (x, y) of each cell as input channels (Galerkin transformer)
- **A learned code for each position fixes the grid size**. Relative codes and coordinates carry over to another grid



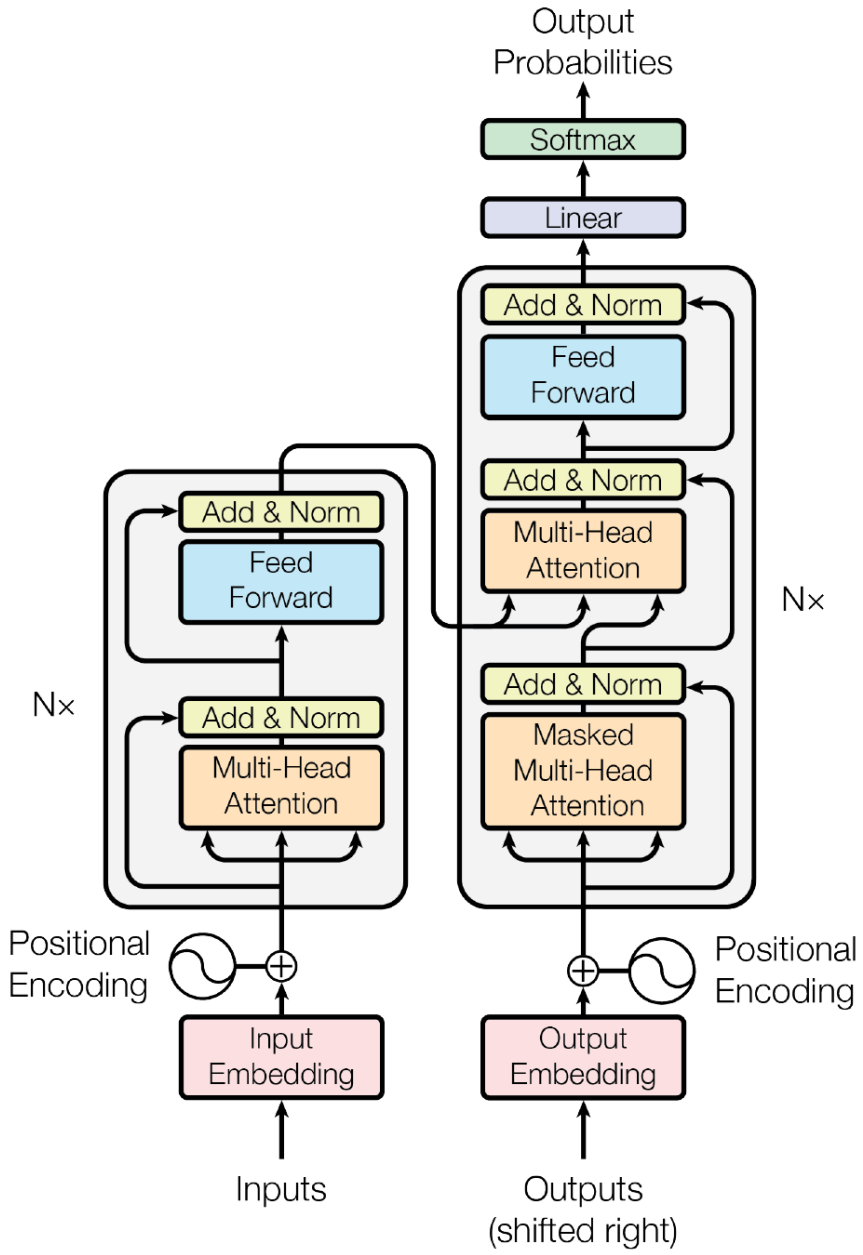
n: position, i: index of the entry, d: token dimension. Figure: Dosovitskiy et al., ICLR 2021
Raffel et al. (T5), JMLR 2020. Su et al., Neurocomputing 2024

The transformer block

- One block, in the pre-norm form used by ViT:

$$Z = X + \mathrm{MHA}\big(\mathrm{LN}(X)\big), \qquad Y = Z + \mathrm{MLP}\big(\mathrm{LN}(Z)\big)$$

- LN: layer normalization of each token. MLP: two layers, applied to each token separately
- **Attention** mixes information across tokens. **MLP** transforms each token
- Residual connections: a residual block is a forward Euler step in the layer index
- **Figure**: the **original transformer**. Encoder (left stack) and decoder (right stack, with masked attention). It applies LN after the addition (post-norm); pre-norm trains more stably in deep stacks. 6 encoder and 6 decoder blocks, d = 512, 8 heads
- ViT-Base: encoder only, 12 blocks, d = 768, 12 heads



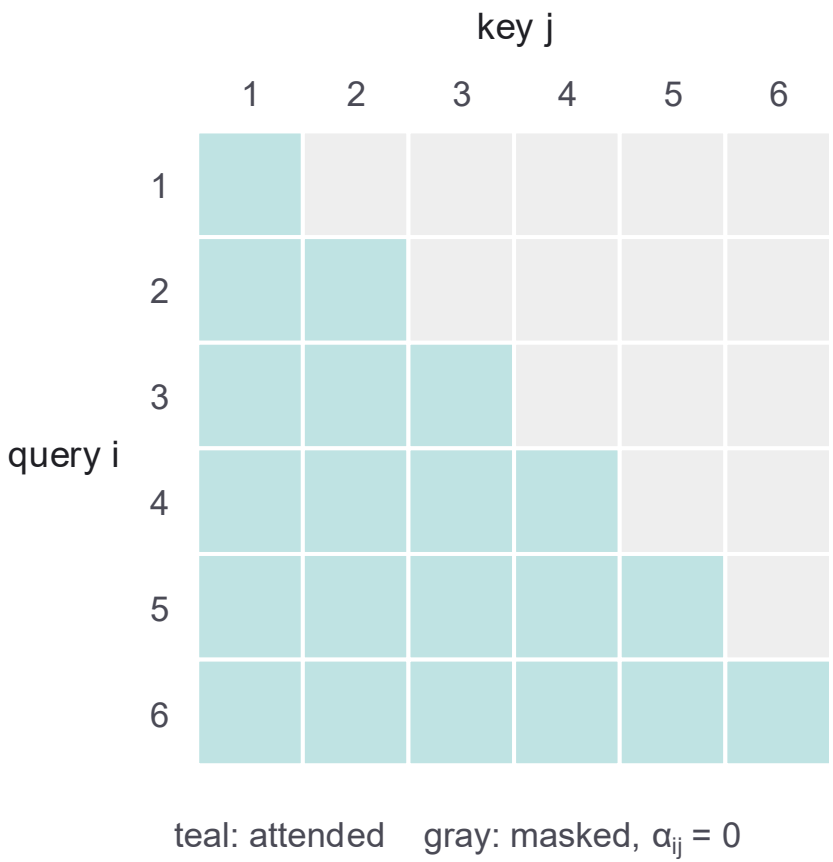
X: input tokens, Z: after attention, Y: block output. Figure: Vaswani et al., NeurIPS 2017. Xiong et al., ICML 2020

Causal attention for sequences in time

- Prediction at step i must not use later steps. Set the score $\mathbf{s_{ij}}$ to $-\infty$ for j > i:

$$\alpha_{ij} = \frac{\exp(s_{ij})}{\sum_{j' \le i} \exp(s_{ij'})} \quad \text{for } j \le i, \qquad \alpha_{ij} = 0 \quad \text{for } j > i$$

- **Training**: all N predictions in one forward pass, with true frames as input (teacher forcing). An RNN must run the N steps sequentially
- Rollout is still one step at a time, feeding back predictions
- **Encoder**: no mask. **Decoder**: causal mask. GPT-style language models are causal decoders



$s_{ij} = \mathbf{q}_i \cdot \mathbf{k}_j / \sqrt{d_k}$: score

Cost of self-attention

Layer	Work per layer	Sequential operations	Maximum path length
Self-attention	$O(N^2\ d)$	$O(1)$	$O(1)$
Recurrent	$O(N\ d^2)$	$O(N)$	$O(N)$
Convolution, kernel k	$O(k\ N\ d^2)$	$O(1)$	$O(N/k)$
Self-attention within r neighbors	$O(r\ N\ d)$	$O(1)$	$O(N/r)$

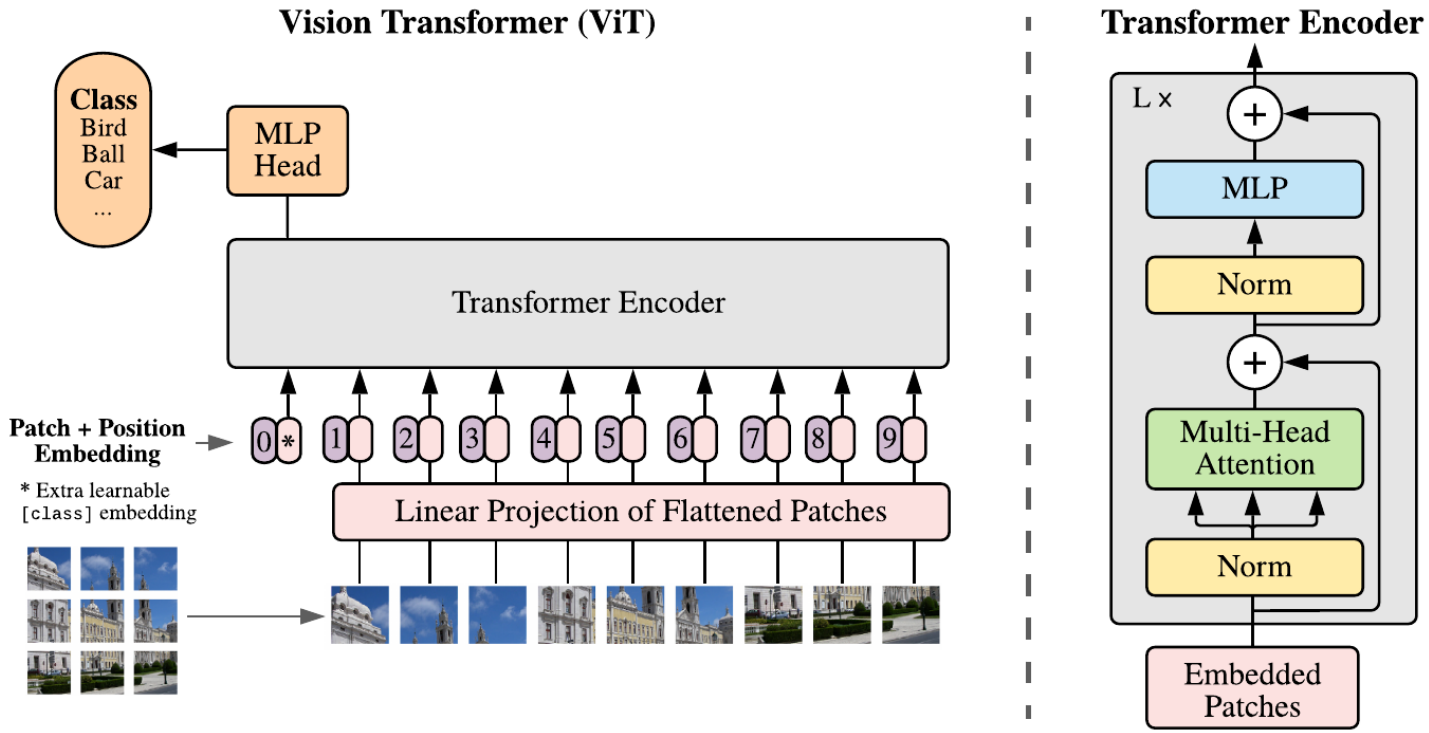
- N x N score matrix: work and memory grow as N^2
- Attention is cheaper than recurrence when $N < d$. Sentences: $N \approx 50$, $d = 512$. Fields: $N \gg d$
- 64 x 64 field, one token per cell: $N = 4096$ and 1.7×10^7 scores per head per layer
- 512 x 512 BubbleML frame: $N = 262{,}144$ and 6.9×10^{10} scores: **275 GB** in 32-bit floats

- Rest of the lecture: reduce N, or the number of pairs that attend

N: number of tokens, d: token dimension, k: kernel size, r: neighborhood size. Table: Vaswani et al., NeurIPS 2017

Vision transformer

- Decompose the field into p x p patches. Flatten each patch to p^2C numbers (C channels)
- **Patch embedding**: one learned matrix maps each patch to a token of dimension d. Same as a convolution with kernel p and stride p
- Add a position encoding. Run a standard transformer encoder
- Classification: an extra learned [class] token. Field output: a linear layer maps each token back to a p x p patch
- ViT: 224 x 224 images, 16 x 16 patches, N = 196 tokens



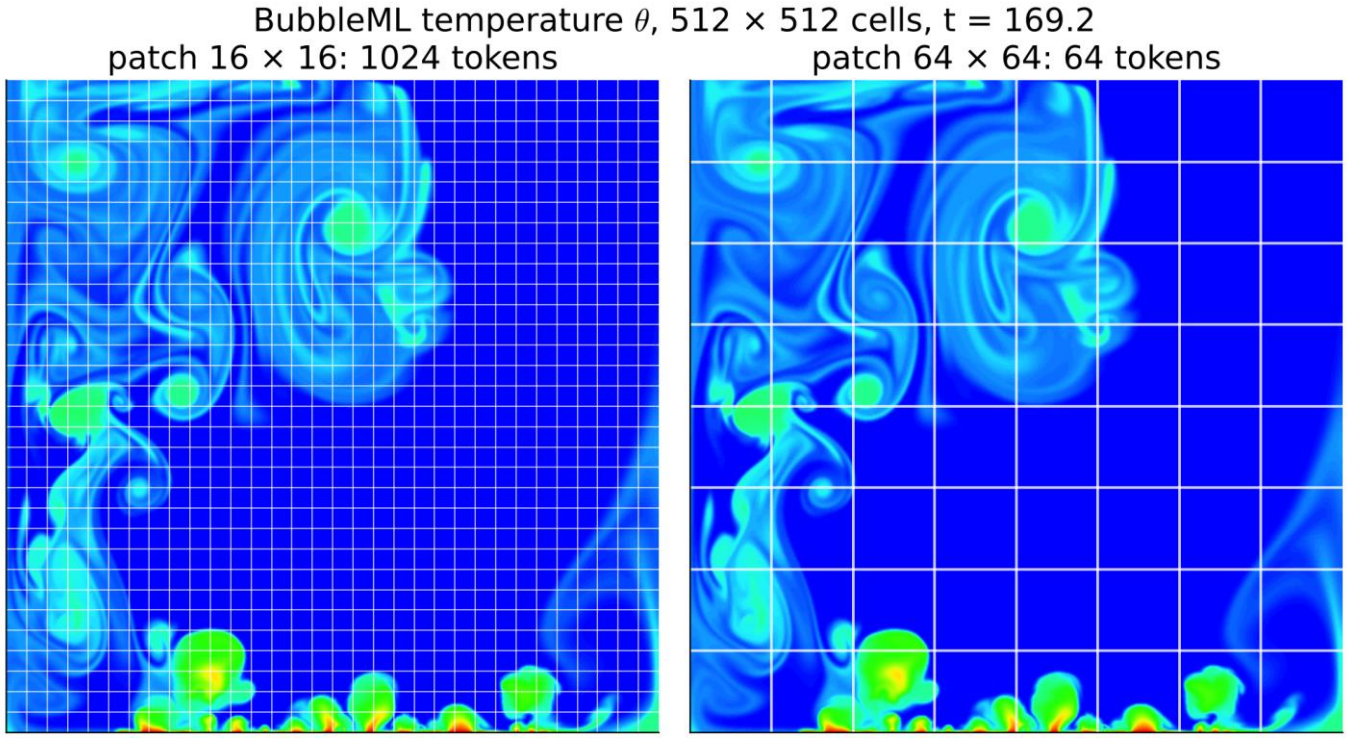
C: channels, p: patch size. Figure: Dosovitskiy et al., An Image is Worth 16x16 Words, ICLR 2021

Patches of a BubbleML frame

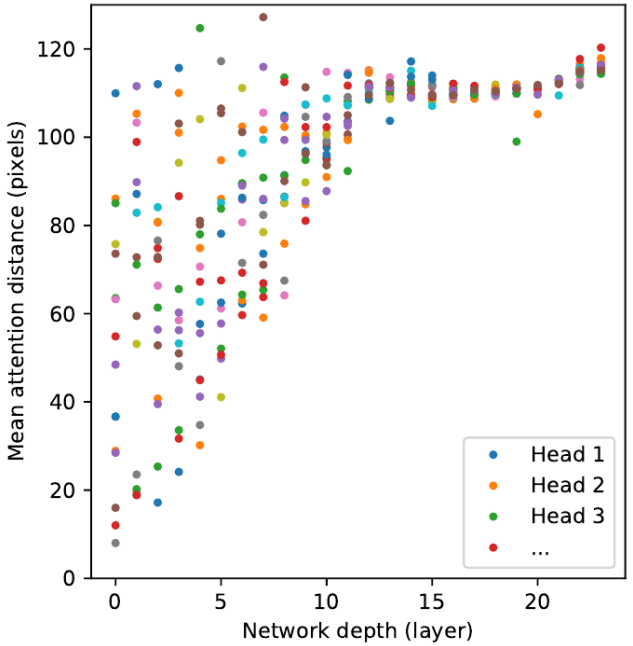
- Number of tokens and cost of attention:

$$N = \frac{H}{p} \cdot \frac{W}{p}, \qquad \text{cost of attention } \propto N^2 = \frac{H^2 W^2}{p^4}$$

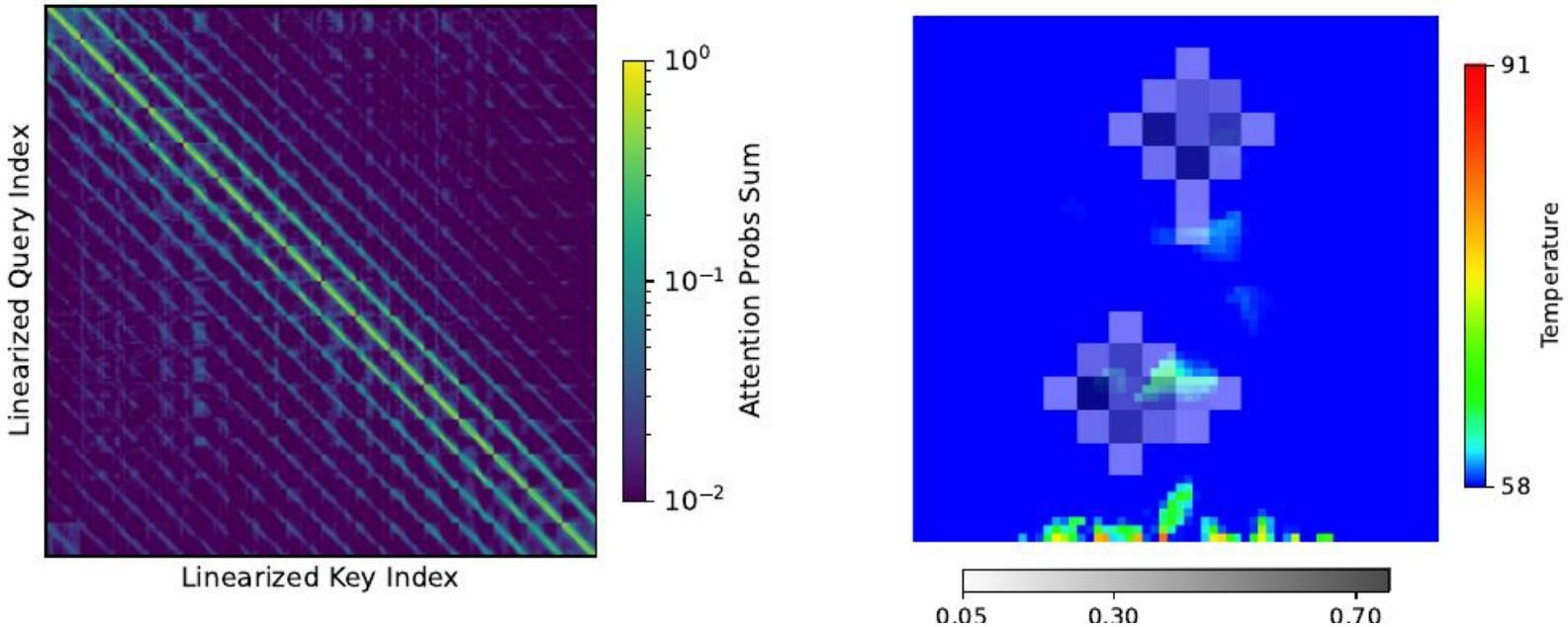
- 512 x 512 frame: patch 16 gives 1024 tokens; patch 64 gives 64 tokens
- Halving p multiplies N by 4 and the attention cost by 16
- Each token is decoded into a p x p patch. Detail smaller than a patch, such as plumes and interfaces, must be encoded in the d numbers of one token
- Large patches: blocky output at patch edges, lost small scales
- NUCLEUS: patch 4 on 64 x 64 inputs. Pangu-Weather: 4 x 4 on a 1440 x 721 grid



Attention distance in trained models



ViT on images: each dot is one head



Boiling ViT with global attention: (a) all heads, first block; (b) two query patches

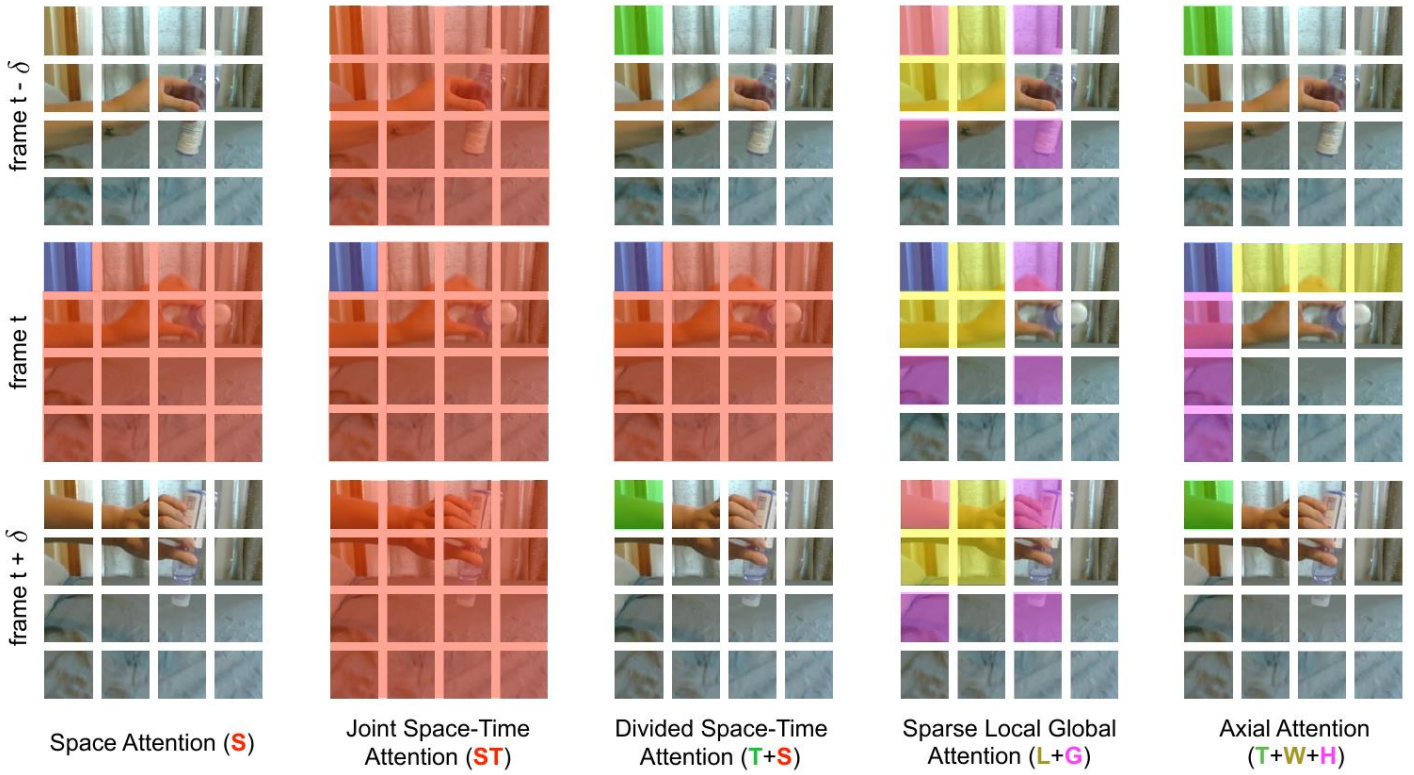
- Mean attention distance: query-to-key distance weighted by α_{ij}. ViT: early layers mix local and global heads
- Boiling: weights concentrate on nearby patches (stripes), and each query attends to a region around itself

Trained global attention on boiling data is mostly local

Left: Dosovitskiy et al., ICLR 2021. Right: Feeney et al., NUCLEUS-MoE, KDD 2026

Attention in space and time

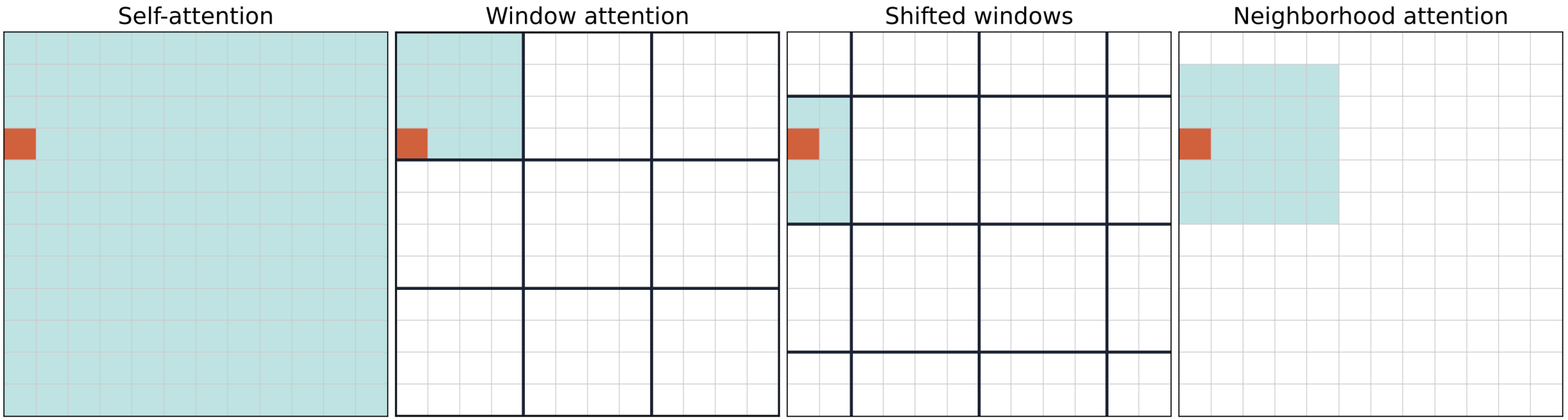
- T frames x N patches = TN tokens
- **Joint** space-time attention: every token attends to all TN tokens. Cost $\propto T^2N^2$
- **Divided**: attend over time at the same patch, then over space within the frame. Cost $\propto TN(T + N)$
  - TimeSformer: the most accurate of the five patterns in its tests
- **Axial**: one axis at a time: time, width, height. Cost per token $\propto T + W + H$



Blue: query patch. **Colored**: keys it attends to; several colors are separate attention steps

T: frames, N = HW: patches per frame. Figure: Bertasius, Wang & Torresani, Is Space-Time Attention All You Need for Video Understanding?, ICML 2021

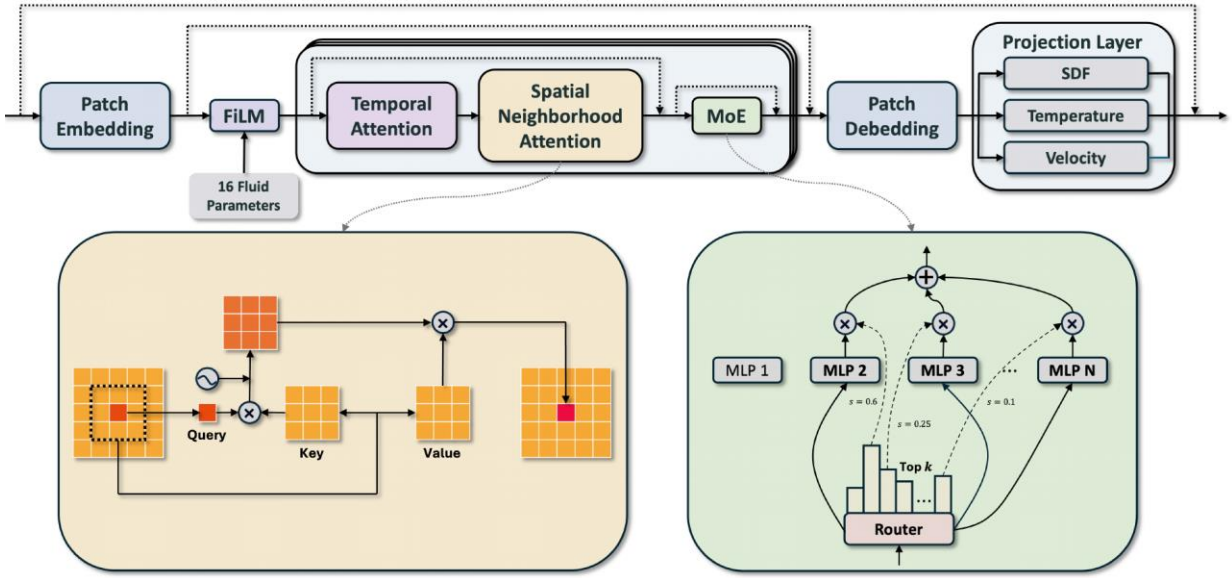
Windowed and neighborhood attention



- **Window** attention (Swin): attend within fixed windows; shift the windows by half a window in the next layer (Liu et al., ICCV 2021)
- **Neighborhood** attention: each token attends to the k x k tokens around it; translation equivariant, like a convolution (Hassani et al., CVPR 2023)
- Work $O(N k^2 d)$, linear in N. The receptive field grows with depth, as in a CNN
- Pangu-Weather: 3D shifted windows. Poseidon (scOT): SwinV2. NUCLEUS: neighborhood attention

Orange: query. Teal: keys it attends to. 12 x 12 tokens, 4 x 4 windows shifted by 2, 5 x 5 neighborhood. After Hassani et al, CVPR 2023. Bi et al., Nature 2023. Herde et al., NeurIPS 2024

Neighborhood attention for boiling



- NUCLEUS: temporal attention, then spatial neighborhood attention, then a mixture-of-experts layer
- At 2048 x 2048, full attention takes about twice the time of neighborhood attention per block. The difference grows with resolution
- Accuracy: neighborhood and global attention reach similar error

Grid	Block	Inference (ms)	Training (ms)	Memory (GB)
512²	Full attention + MLP	**16.1**	**48.4**	1.3
512²	Neighborhood + MLP	**15.3**	**46.3**	1.3
512²	NUCLEUS	6.5	17.1	0.5
2048²	Full attention + MLP	**498.6**	**1659.6**	19.7
2048²	Neighborhood + MLP	**242.7**	**745.5**	19.6
2048²	NUCLEUS	96.3	250.5	8.0

One transformer block, patch size 8, d = 256, one GPU. Figure and table: Feeney, Zou, Hassan, Rachabathuni & Chandramowlishwaran, NUCLEUS-MoE, KDD 2026

Linear attention

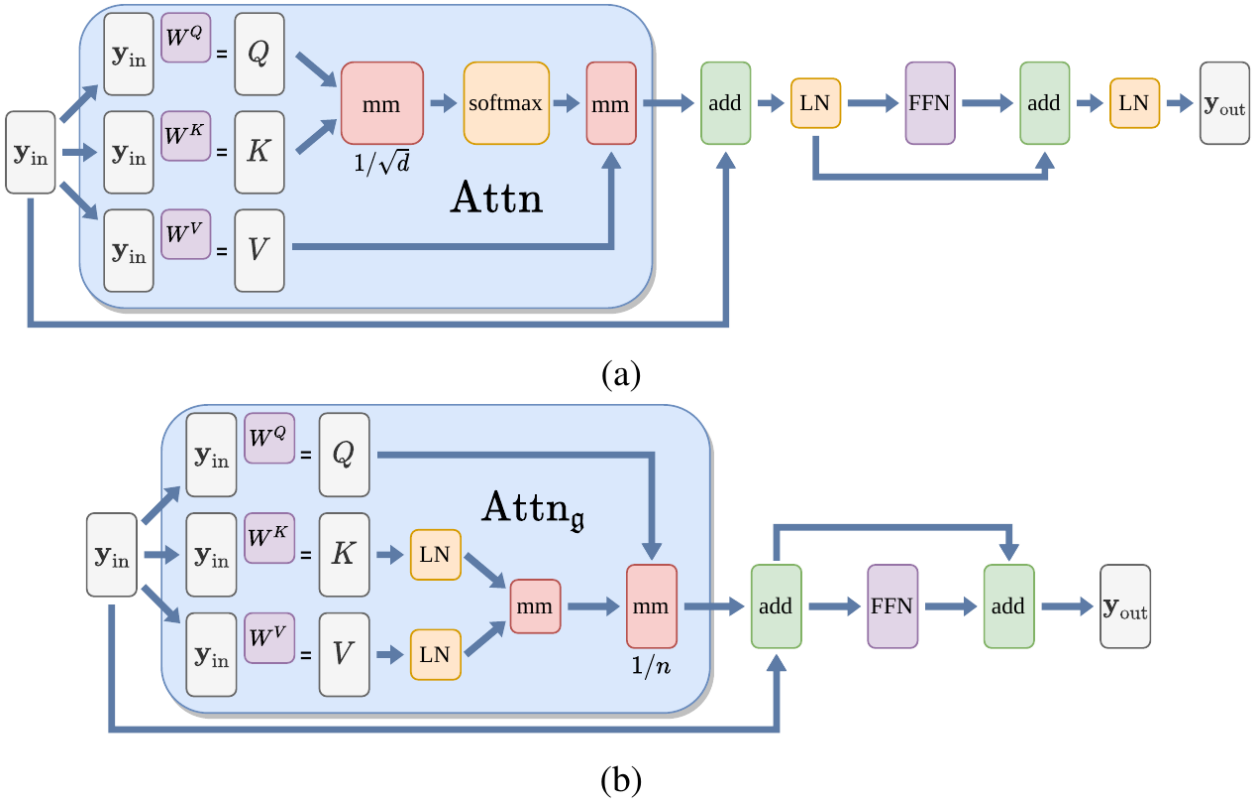
- Softmax attention forms the N x N matrix first:

$$\underbrace{\mathrm{softmax}\left(\boldsymbol{Q}\boldsymbol{K}^\top\right)}_{N\times N}\boldsymbol{V} \qquad O(N^2 d)$$

- Without the softmax, the product regroups: $\boldsymbol{K}^\top\boldsymbol{V}$ is d x d, so the work is linear in N (Katharopoulos et al. 2020)
- **Galerkin attention** (Cao 2021): layer-normalize **K** and **V** ($\tilde{\boldsymbol{K}}$, $\tilde{\boldsymbol{V}}$), no softmax, divide by N:

$$\frac{1}{N}\boldsymbol{Q}\underbrace{\left(\tilde{\boldsymbol{K}}^\top\tilde{\boldsymbol{V}}\right)}_{d\times d} \qquad O(Nd^2)$$

- 8192 grid points: 2.4 GB of GPU memory vs 18.4 GB for softmax attention
- **Weights can be negative and do not sum to 1**
- Transolver: attention among M learned groups of mesh points; work linear in N (Wu et al. 2024)



$\tilde{\boldsymbol{K}}$, $\tilde{\boldsymbol{V}}$: layer-normalized **K**, **V**. Figure: Cao, Choose a Transformer: Fourier or Galerkin, NeurIPS 2021
Katharopoulos et al., ICML 2020. Wu et al., ICML 2024

Attention as an integral operator

- Token j is at point y_j of the domain Ω and has the value $\boldsymbol{v(y_j)}$
- The output at x is a weighted sum over all points, which approximates an integral:

$$o(x_i) = \sum_{j=1}^{N} \kappa(x_i, y_j)\, \boldsymbol{v(y_j)} \;\approx\; \int_{\Omega} \kappa(x_i, y)\, \boldsymbol{v(y)}\, dy$$

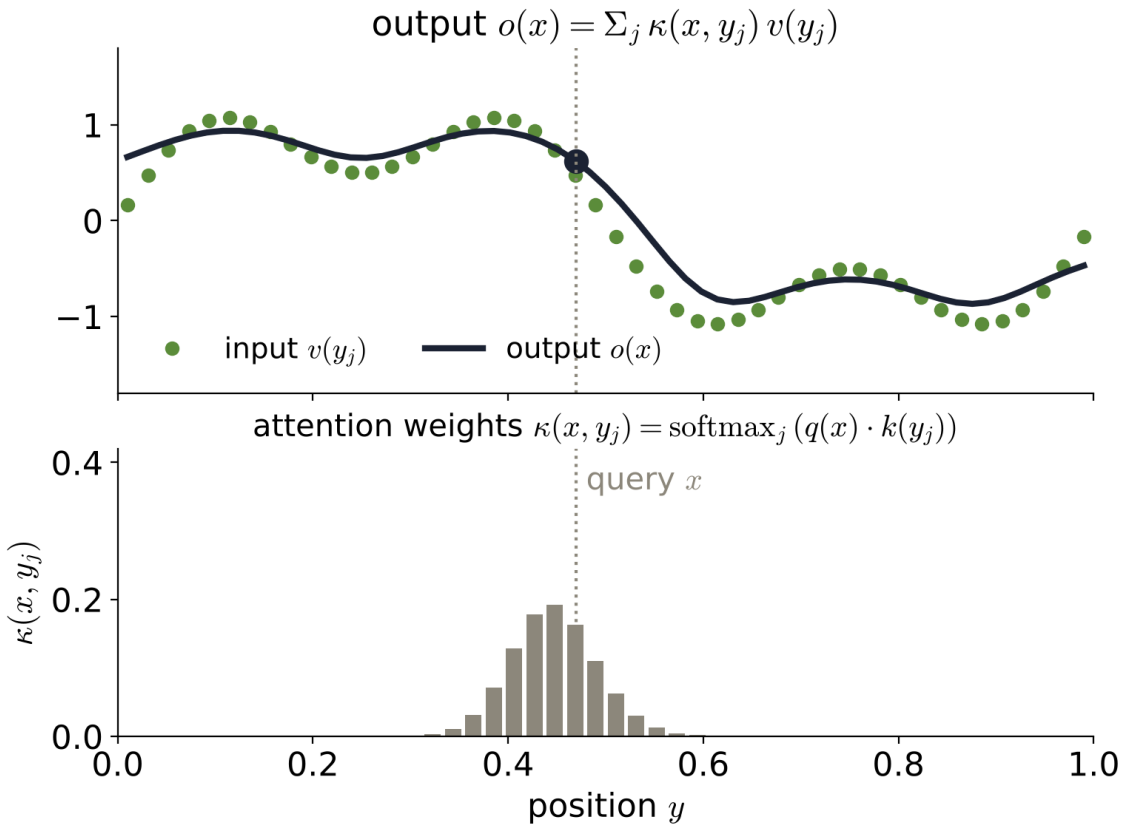
- The kernel is the attention weight, computed from the input:

$$\kappa(x_i, y_j) = \alpha_{ij} = \mathrm{softmax}_j\left(\boldsymbol{q_i} \cdot \boldsymbol{k_j} / \sqrt{d_k}\right)$$

- Figure: 1D attention with queries and keys chosen to get a local kernel:

$$\boldsymbol{q(x)} = \left(x/\sigma^2,\ 1\right), \qquad \boldsymbol{k(y)} = \left(y,\ -y^2/(2\sigma^2) + \gamma\, \boldsymbol{v(y)}\right)$$

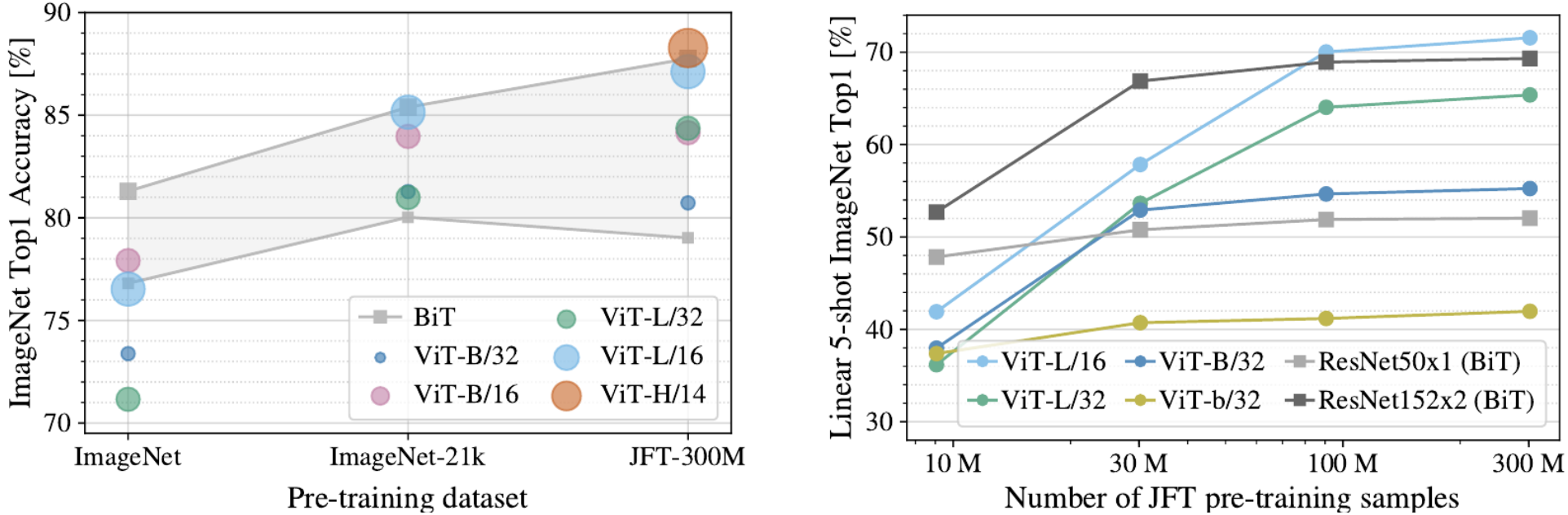
- The weights form a bump around x, pulled toward larger **v**. The output is a smoothed copy of the inputs



Attention is a learned, input-dependent integral operator

Kovachki et al., Neural Operator, JMLR 2023

Transformers and CNNs at different data sizes



- A CNN assumes locality and translation equivariance. Global attention must learn them from data
- Left: pretrained on 1.3M images, ResNets (gray band) beat every ViT. On 300M images, large ViTs are better.
- Right: same trend on subsets of 10M to 300M images
- Scientific datasets are small: BubbleML 2.0 has about 160 simulations. Pretraining on many PDEs as in Poseidon is an active research direction

Built-in structure matters most when data are scarce

Figure: Dosovitskiy et al., ICLR 2021 (BiT: ResNets of Kolesnikov et al. 2020).
BubbleML 2.0: Hassan et al., NeurIPS 2025. Herde et al., Poseidon, NeurIPS 2024

Transformers for weather and PDEs

Model	Data	Tokens	Attention
Pangu-Weather (Bi et al. 2023)	global weather (ERA5)	3D patches, 2 x 4 x 4	3D shifted windows
Poseidon (Herde et al. 2024)	many 2D PDEs, pretrained	2D patches	shifted windows (SwinV2)
Galerkin transformer (Cao 2021)	1D and 2D PDE operators	grid points	linear (Galerkin)
Transolver (Wu et al. 2024)	PDEs on meshes	M learned groups of points	among the M groups
NUCLEUS (Feeney et al. 2026)	boiling, several fluids	spatiotemporal patches	temporal + neighborhood

- Each model reduces the number of tokens (patches, groups) or restricts which tokens attend to which

Choose the token and the attention pattern from the physics and the grid

Bi et al., Nature 2023. Herde et al., NeurIPS 2024. Cao, NeurIPS 2021. Wu et al., ICML 2024. Feeney et al., KDD 2026

Comparison of temporal models

	Frame window	RNN, LSTM, GRU	Neural ODE	Transformer
Memory	last k frames	hidden state h	none beyond the state	all frames in the window
Time step	fixed Δt	fixed Δt	any time; irregular data	fixed Δt; times as position codes
Training	one-step pairs	BPTT over sequences	through the solver or adjoint	parallel over the sequence (causal mask)
Cost per prediction	one forward pass	one cell step	NFE forward passes	$O(T^2)$ per patch for T frames
Limitation	fixed reach	sequential; long lags hard	solver cost; needs a complete state	cost grows with window and patches; needs data
Reference	Lecture 7	Hochreiter & Schmidhuber 1997; Cho et al. 2014	Chen et al. 2018	Vaswani et al. 2017

Attention reads any past frame in one step, at a cost quadratic in the window

k: window length, T: frames attended. BPTT: backpropagation through time. NFE: number of function evaluations

Summary

- Attention: a weighted average of values, with weights from the similarity of a query to each key
- Self-attention connects every pair of tokens in one step and computes its weights from the input
- Self-attention is permutation equivariant; position encoding give it the order or the grid
- Transformer block: attention across tokens, MLP on each token, residual connections and LayerNorm
- Work and memory grow as N^2. Patches reduce N; factorized, windowed and neighborhood attention restrict the pairs; linear attention avoids the N x N matrix
- Attention is a learned, input-dependent integral operator
- With little data, built-in structure (locality, equivariance) helps

After class

- Vaswani et al., Attention Is All You Need, NeurIPS 2017
- Dosovitskiy et al., An Image is Worth 16x16 Words, ICLR 2021
- Cao, Choose a Transformer: Fourier or Galerkin, NeurIPS 2021
- Prince, Understanding Deep Learning, Chapter 12 (transformers)
- 3Blue1Brown, Neural Networks, Chapters 5 and 6 (transformers and attention)

- Project abstract due today

Next, Lecture 10 (Tue, Oct 27): Evaluating SciML

NEXT: HANDS-ON

From concepts to code

→ Open the Lecture 9 notebook

1. Write scaled dot-product attention; check that it is permutation equivariant; add position codes and check again

2. Train a small vision transformer on BubbleML (64 x 64, patch 4); compare one-step and rollout error with the Lecture 7 SmallCNN

3. Plot attention maps for a query patch near a bubble and one in the liquid