Attention and Vision Transformers

Computer Vision (CSCI 5520G)

Faisal Z. Qureshi

http://vclab.science.ontariotechu.ca

Week 7 at a glance

- **The problem with locality**
  - What convolution assumes, and what it costs
- **Attention**
  - Queries, keys, values; scaled dot-product attention
  - Self-attention, positional encoding, multiple heads
- **The transformer block**
- **Vision transformers**
  - ViT, and the data-hunger result
  - DeiT, Swin, DETR
- **What attention does and does not tell us**

Reading: the ViT paper (Dosovitskiy et al., ICLR 2021) and "Attention Is All You Need" (Vaswani et al., NeurIPS 2017).

Where we are

Five weeks of convolutional networks. Convolution buys us two things:

Where we are

Five weeks of convolutional networks. Convolution buys us two things:

- **Locality.** A unit looks at a small neighbourhood.
- **Weight sharing.** The same filter is applied everywhere.

Where we are

Five weeks of convolutional networks. Convolution buys us two things:

- **Locality.** A unit looks at a small neighbourhood.
- **Weight sharing.** The same filter is applied everywhere.

Together these encode a strong prior: *what matters is local, and it matters the same way everywhere in the image*.

Where we are

Five weeks of convolutional networks. Convolution buys us two things:

- **Locality.** A unit looks at a small neighbourhood.
- **Weight sharing.** The same filter is applied everywhere.

Together these encode a strong prior: *what matters is local, and it matters the same way everywhere in the image*.

That prior is why CNNs work so well on so little data. This lecture is about what it costs.

The cost of locality

A 3×3 convolution mixes information from a 3×3 neighbourhood.
To relate two pixels R apart, you need depth.

The cost of locality

A 3×3 convolution mixes information from a 3×3 neighbourhood. To relate two pixels R apart, you need depth.

Stacking L layers of $k \times k$ convolution gives a receptive field of

$$1 + L(k - 1)$$

so relating pixels 224 apart with 3×3 filters needs $L \approx 112$ layers.

The cost of locality

A 3×3 convolution mixes information from a 3×3 neighbourhood. To relate two pixels R apart, you need depth.

Stacking L layers of $k \times k$ convolution gives a receptive field of

$$1 + L(k - 1)$$

so relating pixels 224 apart with 3×3 filters needs $L \approx 112$ layers.

Striding and pooling shortcut this — at the price of spatial resolution. Dilated convolution shortcuts it — at the price of sampling density.

The cost of locality

A 3×3 convolution mixes information from a 3×3 neighbourhood. To relate two pixels R apart, you need depth.

Stacking L layers of $k \times k$ convolution gives a receptive field of

$$1 + L(k - 1)$$

so relating pixels 224 apart with 3×3 filters needs $L \approx 112$ layers.

Striding and pooling shortcut this — at the price of spatial resolution. Dilated convolution shortcuts it — at the price of sampling density.

Every solution trades something. The question is whether an architecture can relate two distant positions in *one* step, at full resolution.

A different question

Convolution asks: *what is in this neighbourhood?*

A different question

Convolution asks: *what is in this neighbourhood?*

Attention asks: *given what I am looking for, which positions in the image are relevant — wherever they are?*

A different question

Convolution asks: *what is in this neighbourhood?*

Attention asks: *given what I am looking for, which positions in the image are relevant — wherever they are?*

The weights in a convolution are **fixed after training** and depend only on *offset*. The weights in attention are **computed at inference** and depend on *content*.

A different question

Convolution asks: *what is in this neighbourhood?*

Attention asks: *given what I am looking for, which positions in the image are relevant — wherever they are?*

The weights in a convolution are **fixed after training** and depend only on *offset*. The weights in attention are **computed at inference** and depend on *content*.

This is the single idea of the lecture. Everything else is mechanics.

Attention as soft dictionary lookup

A hard dictionary: given a key, return the matching value.

Attention as soft dictionary lookup

A hard dictionary: given a key, return the matching value.

Attention softens this. Every entry is returned, weighted by how well its key matches the query.

- **Query** $\mathbf{q}$ — what I am looking for
- **Keys** $\mathbf{k}_1 \dots \mathbf{k}_N$ — what each position advertises
- **Values** $\mathbf{v}_1 \dots \mathbf{v}_N$ — what each position contributes

Attention as soft dictionary lookup

A hard dictionary: given a key, return the matching value.

Attention softens this. Every entry is returned, weighted by how well its key matches the query.

- **Query** $\mathbf{q}$ — what I am looking for
- **Keys** $\mathbf{k}_1 \dots \mathbf{k}_N$ — what each position advertises
- **Values** $\mathbf{v}_1 \dots \mathbf{v}_N$ — what each position contributes

$$\text{attn}(\mathbf{q}, K, V) = \sum_{i=1}^{N} \alpha_i \mathbf{v}_i, \qquad \alpha_i = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i / \sqrt{d})}{\sum_j \exp(\mathbf{q}^\top \mathbf{k}_j / \sqrt{d})}$$

Attention as soft dictionary lookup

A hard dictionary: given a key, return the matching value.

Attention softens this. Every entry is returned, weighted by how well its key matches the query.

- **Query** $\mathbf{q}$ — what I am looking for
- **Keys** $\mathbf{k}_1 \dots \mathbf{k}_N$ — what each position advertises
- **Values** $\mathbf{v}_1 \dots \mathbf{v}_N$ — what each position contributes

$$\text{attn}(\mathbf{q}, K, V) = \sum_{i=1}^{N} \alpha_i \mathbf{v}_i, \qquad \alpha_i = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i / \sqrt{d})}{\sum_j \exp(\mathbf{q}^\top \mathbf{k}_j / \sqrt{d})}$$

The α_i are non-negative and sum to one: a distribution over positions.

Why divide by $\sqrt{d}$

Suppose the entries of $\mathbf{q}$ and $\mathbf{k}$ are independent, zero-mean, unit-variance. Then

$$\mathbf{q}^\top \mathbf{k} = \sum_{j=1}^{d} q_j k_j$$

has mean 0 and **variance** d.

Why divide by $\sqrt{d}$

Suppose the entries of $\mathbf{q}$ and $\mathbf{k}$ are independent, zero-mean, unit-variance. Then

$$\mathbf{q}^\top \mathbf{k} = \sum_{j=1}^{d} q_j k_j$$

has mean 0 and **variance** d.

For $d = 64$ the logits have standard deviation 8. A softmax over logits that large is nearly one-hot — and its gradient is nearly zero.

Why divide by $\sqrt{d}$

Suppose the entries of $\mathbf{q}$ and $\mathbf{k}$ are independent, zero-mean, unit-variance. Then

$$\mathbf{q}^\top \mathbf{k} = \sum_{j=1}^{d} q_j k_j$$

has mean 0 and **variance** d.

For $d = 64$ the logits have standard deviation 8. A softmax over logits that large is nearly one-hot — and its gradient is nearly zero.

Dividing by $\sqrt{d}$ restores unit variance and keeps the softmax in a regime where it can still learn.

Why divide by $\sqrt{d}$

Suppose the entries of $\mathbf{q}$ and $\mathbf{k}$ are independent, zero-mean, unit-variance. Then

$$\mathbf{q}^\top \mathbf{k} = \sum_{j=1}^{d} q_j k_j$$

has mean 0 and **variance** d.

For $d = 64$ the logits have standard deviation 8. A softmax over logits that large is nearly one-hot — and its gradient is nearly zero.

Dividing by $\sqrt{d}$ restores unit variance and keeps the softmax in a regime where it can still learn.

A one-line fix for a problem that otherwise stops training. Worth remembering when you read the paper.

Self-attention

So far the queries came from somewhere else. In **self-attention**, all three come from the same set of N tokens $X \in \mathbb{R}^{N \times d}$:

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

Self-attention

So far the queries came from somewhere else. In **self-attention**, all three come from the same set of N tokens $X \in \mathbb{R}^{N \times d}$:

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

$$\mathsf{SA}(X) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

Self-attention

So far the queries came from somewhere else. In **self-attention**, all three come from the same set of N tokens $X \in \mathbb{R}^{N \times d}$:

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

$$\mathrm{SA}(X) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

Every token attends to every token, in one layer. The receptive field is the whole input, immediately.

Self-attention

So far the queries came from somewhere else. In **self-attention**, all three come from the same set of N tokens $X \in \mathbb{R}^{N \times d}$:

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

$$\mathrm{SA}(X) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

Every token attends to every token, in one layer. The receptive field is the whole input, immediately.

The learned parameters are only W_Q, W_K, W_V — the *interaction pattern* is computed from the data.

Self-attention is permutation equivariant

Permute the input tokens and the output permutes identically.

Self-attention is permutation equivariant

Permute the input tokens and the output permutes identically.

Nothing in the equations knows that token 5 sits beside token 6.

Self-attention is permutation equivariant

Permute the input tokens and the output permutes identically.

Nothing in the equations knows that token 5 sits beside token 6.

For a bag of words this is a feature. **For an image it is a disaster** — we have thrown away the spatial structure that convolution gave us for free.

Self-attention is permutation equivariant

Permute the input tokens and the output permutes identically.

Nothing in the equations knows that token 5 sits beside token 6.

For a bag of words this is a feature. **For an image it is a disaster** — we have thrown away the spatial structure that convolution gave us for free.

The fix is to put the position back in, by adding a **positional encoding** to each token embedding. Fixed sinusoids, or learned vectors; ViT uses learned.

Self-attention is permutation equivariant

Permute the input tokens and the output permutes identically.

Nothing in the equations knows that token 5 sits beside token 6.

For a bag of words this is a feature. **For an image it is a disaster** — we have thrown away the spatial structure that convolution gave us for free.

The fix is to put the position back in, by adding a **positional encoding** to each token embedding. Fixed sinusoids, or learned vectors; ViT uses learned.

Note what has happened: convolution had geometry built in. Attention has it *bolted on*.

Multiple heads

One attention map forces one notion of relevance. A token may need several — “same object”, “similar texture”, “symmetric counterpart”.

Multiple heads

One attention map forces one notion of relevance. A token may need several — "same object", "similar texture", "symmetric counterpart".

Run h attention operations in parallel with separate projections, each of dimension d/h, then concatenate:

$$\mathrm{MHA}(X) = [\mathrm{head}_1; \ldots; \mathrm{head}_h]\, W_O$$

Multiple heads

One attention map forces one notion of relevance. A token may need several — "same object", "similar texture", "symmetric counterpart".

Run h attention operations in parallel with separate projections, each of dimension d/h, then concatenate:

$$\mathrm{MHA}(X) = [\mathrm{head}_1; \ldots; \mathrm{head}_h]\, W_O$$

Cost is unchanged — the per-head dimension shrinks to compensate.

Multi-head attention

[Figure placeholder]

Multi-head attention: the same tokens, several attention maps computed in parallel, concatenated and projected

The cost

The attention matrix is $N \times N$.

$$\text{time and memory } \sim \mathcal{O}(N^2 d)$$

The cost

The attention matrix is $N \times N$.

$$\text{time and memory} \sim \mathcal{O}(N^2 d)$$

For a 224×224 image at one token per pixel, $N = 50{,}176$ and $N^2 \approx 2.5 \times 10^9$ **per head, per layer**.

The cost

The attention matrix is $N \times N$.

$$\text{time and memory} \sim \mathcal{O}(N^2 d)$$

For a 224×224 image at one token per pixel, $N = 50{,}176$ and $N^2 \approx 2.5 \times 10^9$ **per head, per layer**.

This is why the first vision transformers did not operate on pixels. The whole design of ViT follows from needing to make N small.

The transformer block

Attention alone is a weighted average — it mixes but does not transform. A block interleaves mixing with per-token computation:

$$X' = X + \mathrm{MHA}(\mathrm{LN}(X))$$
$$X'' = X' + \mathrm{MLP}(\mathrm{LN}(X'))$$

The transformer block

Attention alone is a weighted average — it mixes but does not transform. A block interleaves mixing with per-token computation:

$$X' = X + \mathrm{MHA}(\mathrm{LN}(X))$$
$$X'' = X' + \mathrm{MLP}(\mathrm{LN}(X'))$$

- **Attention** mixes *across* tokens.
- **MLP** transforms *within* each token, independently.
- **Residual connections** keep gradients flowing, as in ResNet.
- **Layer normalisation**, not batch norm — no dependence on batch statistics.

The transformer block

Attention alone is a weighted average — it mixes but does not transform. A block interleaves mixing with per-token computation:

$$X' = X + \mathrm{MHA}(\mathrm{LN}(X))$$
$$X'' = X' + \mathrm{MLP}(\mathrm{LN}(X'))$$

- **Attention** mixes *across* tokens.
- **MLP** transforms *within* each token, independently.
- **Residual connections** keep gradients flowing, as in ResNet.
- **Layer normalisation**, not batch norm — no dependence on batch statistics.

Placing LN *inside* the residual branch (pre-norm) rather than after it makes deep stacks trainable without careful warm-up. Most modern implementations do this.

An image is worth 16x16 words

ViT's answer to the N^2 problem: **do not use pixels as tokens.**

An image is worth 16x16 words

ViT's answer to the N^2 problem: **do not use pixels as tokens.**

- Split the image into non-overlapping 16×16 patches
- Flatten each patch and project it linearly to d dimensions
- Add a learned positional embedding
- Prepend a learned `[CLS]` token
- Apply a standard transformer encoder
- Classify from the final `[CLS]` representation

An image is worth 16x16 words

ViT's answer to the N^2 problem: **do not use pixels as tokens.**

- Split the image into non-overlapping 16×16 patches
- Flatten each patch and project it linearly to d dimensions
- Add a learned positional embedding
- Prepend a learned `[CLS]` token
- Apply a standard transformer encoder
- Classify from the final `[CLS]` representation

For 224×224 images, $N = (224/16)^2 = 196$ tokens, plus `[CLS]`.

An image is worth 16x16 words

ViT's answer to the N^2 problem: **do not use pixels as tokens.**

- Split the image into non-overlapping 16×16 patches
- Flatten each patch and project it linearly to d dimensions
- Add a learned positional embedding
- Prepend a learned `[CLS]` token
- Apply a standard transformer encoder
- Classify from the final `[CLS]` representation

For 224×224 images, $N = (224/16)^2 = 196$ tokens, plus `[CLS]`.

$196^2 \approx 3.8 \times 10^4$ — five orders of magnitude below the per-pixel cost.

What the patch embedding really is

Flattening a $16 \times 16 \times 3$ patch and multiplying by a matrix is exactly a convolution with kernel size 16 and stride 16.

What the patch embedding really is

Flattening a $16 \times 16 \times 3$ patch and multiplying by a matrix is exactly a convolution with kernel size 16 and stride 16.

So ViT is not "no convolution". It is **one convolutional layer, then pure attention**.

What the patch embedding really is

Flattening a $16 \times 16 \times 3$ patch and multiplying by a matrix is exactly a convolution with kernel size 16 and stride 16.

So ViT is not "no convolution". It is **one convolutional layer, then pure attention**.

That first layer is doing real work: it is the only place in the architecture where local spatial structure is treated as local.

[Figure placeholder]

Image to patches to token sequence, with CLS token and positional embeddings

The result that mattered

On ImageNet-1k alone, ViT is **worse** than a comparable ResNet.

The result that mattered

On ImageNet-1k alone, ViT is **worse** than a comparable ResNet.

Pretrained on ImageNet-21k, it is comparable.

The result that mattered

On ImageNet-1k alone, ViT is **worse** than a comparable ResNet.

Pretrained on ImageNet-21k, it is comparable.

Pretrained on JFT-300M, it is better — and keeps improving as data grows, while the ResNet saturates.

The result that mattered

On ImageNet-1k alone, ViT is **worse** than a comparable ResNet.

Pretrained on ImageNet-21k, it is comparable.

Pretrained on JFT-300M, it is better — and keeps improving as data grows, while the ResNet saturates.

The interpretation. Convolution's inductive bias is a substitute for data. When data is scarce the bias helps; when data is abundant it constrains.

The result that mattered

On ImageNet-1k alone, ViT is **worse** than a comparable ResNet.

Pretrained on ImageNet-21k, it is comparable.

Pretrained on JFT-300M, it is better — and keeps improving as data grows, while the ResNet saturates.

The interpretation. Convolution's inductive bias is a substitute for data. When data is scarce the bias helps; when data is abundant it constrains.

This is the most important sentence in the ViT paper, and it is worth being sceptical about: JFT-300M is proprietary, and "abundant" here means 300 million labelled images.

Discussion: is the bias really the story?

Consider the counter-evidence before you accept the narrative.

Discussion: is the bias really the story?

Consider the counter-evidence before you accept the narrative.

- **DeiT** (Touvron et al., 2021) trains ViT on ImageNet-1k alone to competitive accuracy, using strong augmentation, regularisation, and distillation from a CNN teacher.

Discussion: is the bias really the story?

Consider the counter-evidence before you accept the narrative.

- **DeiT** (Touvron et al., 2021) trains ViT on ImageNet-1k alone to competitive accuracy, using strong augmentation, regularisation, and distillation from a CNN teacher.
- So was the original gap about *inductive bias*, or about *optimisation and training recipe*?

Discussion: is the bias really the story?

Consider the counter-evidence before you accept the narrative.

- **DeiT** (Touvron et al., 2021) trains ViT on ImageNet-1k alone to competitive accuracy, using strong augmentation, regularisation, and distillation from a CNN teacher.
- So was the original gap about *inductive bias*, or about *optimisation and training recipe*?
- Modern CNNs trained with transformer-era recipes (ConvNeXt) close much of the gap in the other direction.

Discussion: is the bias really the story?

Consider the counter-evidence before you accept the narrative.

- **DeiT** (Touvron et al., 2021) trains ViT on ImageNet-1k alone to competitive accuracy, using strong augmentation, regularisation, and distillation from a CNN teacher.
- So was the original gap about *inductive bias*, or about *optimisation and training recipe*?
- Modern CNNs trained with transformer-era recipes (ConvNeXt) close much of the gap in the other direction.

A recurring pattern in this literature: architectural claims are entangled with training-recipe claims, and papers rarely separate them cleanly. Ask this question of every architecture paper you present.

Putting some structure back: Swin

ViT throws away hierarchy — every layer runs at one scale, and attention is global. Dense prediction wants both.

Putting some structure back: Swin

ViT throws away hierarchy — every layer runs at one scale, and attention is global. Dense prediction wants both.

Swin transformer:

- Compute attention only within local windows — cost becomes **linear** in N
- **Shift** the windows between consecutive blocks so information crosses boundaries
- Merge patches between stages to build a pyramid

Putting some structure back: Swin

ViT throws away hierarchy — every layer runs at one scale, and attention is global. Dense prediction wants both.

Swin transformer:

- Compute attention only within local windows — cost becomes **linear** in N
- **Shift** the windows between consecutive blocks so information crosses boundaries
- Merge patches between stages to build a pyramid

The result is a transformer that behaves like a CNN backbone: multi-scale feature maps you can hang a detection or segmentation head on.

Putting some structure back: Swin

ViT throws away hierarchy — every layer runs at one scale, and attention is global. Dense prediction wants both.

Swin transformer:

- Compute attention only within local windows — cost becomes **linear** in N
- **Shift** the windows between consecutive blocks so information crosses boundaries
- Merge patches between stages to build a pyramid

The result is a transformer that behaves like a CNN backbone: multi-scale feature maps you can hang a detection or segmentation head on.

Notice the direction of travel — having removed locality and hierarchy, the field promptly reintroduced them.

Attention beyond backbones: DETR

Detection, traditionally: anchors, non-maximum suppression, hand-tuned matching.

Attention beyond backbones: DETR

Detection, traditionally: anchors, non-maximum suppression, hand-tuned matching.

DETR treats detection as **direct set prediction**:

- A CNN or transformer backbone produces features
- A transformer decoder attends to them with N learned "object queries"
- Each query emits one box and class, or "no object"
- A **Hungarian matching** loss pairs predictions to ground truth

Attention beyond backbones: DETR

Detection, traditionally: anchors, non-maximum suppression, hand-tuned matching.

DETR treats detection as **direct set prediction**:

- A CNN or transformer backbone produces features
- A transformer decoder attends to them with N learned "object queries"
- Each query emits one box and class, or "no object"
- A **Hungarian matching** loss pairs predictions to ground truth

No anchors, no NMS. The architecture is simpler and end-to-end.

Attention beyond backbones: DETR

Detection, traditionally: anchors, non-maximum suppression, hand-tuned matching.

DETR treats detection as **direct set prediction**:

- A CNN or transformer backbone produces features
- A transformer decoder attends to them with N learned "object queries"
- Each query emits one box and class, or "no object"
- A **Hungarian matching** loss pairs predictions to ground truth

No anchors, no NMS. The architecture is simpler and end-to-end.

The costs are real: slow convergence (hundreds of epochs), and weak small-object performance — which subsequent work (Deformable DETR) addressed.

What attention maps do not tell you

It is tempting to visualise α and call it an explanation.

What attention maps do not tell you

It is tempting to visualise α and call it an explanation.

Be careful:

- Attention weights are **not** a faithful account of what determined the output — residual connections and the MLP carry information around them.
- Different attention distributions can give identical predictions.
- With multiple heads and layers, "the" attention map is a choice of aggregation, and different choices give different pictures.

What attention maps do not tell you

It is tempting to visualise α and call it an explanation.

Be careful:

- Attention weights are **not** a faithful account of what determined the output — residual connections and the MLP carry information around them.
- Different attention distributions can give identical predictions.
- With multiple heads and layers, "the" attention map is a choice of aggregation, and different choices give different pictures.

There is a substantial literature on this (*Attention is not Explanation*; *Attention is not not Explanation*).

What attention maps do not tell you

It is tempting to visualise α and call it an explanation.

Be careful:

- Attention weights are **not** a faithful account of what determined the output — residual connections and the MLP carry information around them.
- Different attention distributions can give identical predictions.
- With multiple heads and layers, "the" attention map is a choice of aggregation, and different choices give different pictures.

There is a substantial literature on this (*Attention is not Explanation*; *Attention is not not Explanation*).

Attention maps are a useful diagnostic and weak evidence. They are not proof.

Summary

- Convolution fixes weights by offset; attention computes them from content
- Scaled dot-product attention, made stable by the $\sqrt{d}$ factor
- Self-attention is permutation equivariant, so position must be added back
- Cost is $\mathcal{O}(N^2)$ — which dictates patch tokens
- A transformer block mixes across tokens, then transforms within them
- ViT is one strided convolution followed by attention
- Its advantage appears at very large pretraining scale — though the recipe may matter as much as the architecture
- Swin and DETR reintroduce hierarchy and apply attention to dense tasks

For discussion next week

Pick one and be ready to argue it:

For discussion next week

Pick one and be ready to argue it:

1. ViT's patch embedding is a convolution. In what meaningful sense, then, is ViT "convolution-free"?

For discussion next week

Pick one and be ready to argue it:

1. ViT's patch embedding is a convolution. In what meaningful sense, then, is ViT "convolution-free"?

2. If DeiT can train ViT on ImageNet-1k, what exactly did the JFT-300M experiment demonstrate?

For discussion next week

Pick one and be ready to argue it:

1. ViT's patch embedding is a convolution. In what meaningful sense, then, is ViT "convolution-free"?

2. If DeiT can train ViT on ImageNet-1k, what exactly did the JFT-300M experiment demonstrate?

3. Swin restores locality and hierarchy. Is it a transformer with CNN properties, or a CNN with content-dependent weights? Does the distinction matter?

For discussion next week

Pick one and be ready to argue it:

1. ViT's patch embedding is a convolution. In what meaningful sense, then, is ViT "convolution-free"?

2. If DeiT can train ViT on ImageNet-1k, what exactly did the JFT-300M experiment demonstrate?

3. Swin restores locality and hierarchy. Is it a transformer with CNN properties, or a CNN with content-dependent weights? Does the distinction matter?

4. You are given 5,000 labelled medical images and no pretraining budget. What do you use, and why?

Copyright and License