Vision Transformers and Multimodal Models

From image patches to CLIP and vision-language models

Prof. Nipun Batra

ES 667 · Deep Learning · IIT Gandhinagar

From CNNs to Vision Transformers

Where we are

We have two big architectures: **CNNs** for images (L8) and **Transformers** for sequences (L15–16).

Where we are

We have two big architectures: **CNNs** for images (L8) and **Transformers** for sequences (L15–16).

$$\text{CNN}: \quad I \in \mathbb{R}^{H \times W \times C} \longrightarrow \text{conv, pool} \qquad \text{Transformer}: \quad X \in \mathbb{R}^{T \times d} \longrightarrow \text{SelfAttention}(X)$$

Where we are

We have two big architectures: **CNNs** for images (L8) and **Transformers** for sequences (L15–16).

$$\text{CNN}: \quad I \in \mathbb{R}^{H \times W \times C} \longrightarrow \text{conv, pool} \qquad \text{Transformer}: \quad X \in \mathbb{R}^{T \times d} \longrightarrow \text{SelfAttention}(X)$$

The Transformer consumes a **sequence of tokens**. It knows nothing about grids, pixels, or convolutions. This lecture asks a simple question:

Where we are

We have two big architectures: **CNNs** for images (L8) and **Transformers** for sequences (L15–16).

$$\text{CNN}: \quad I \in \mathbb{R}^{H \times W \times C} \longrightarrow \text{conv, pool} \qquad \text{Transformer}: \quad X \in \mathbb{R}^{T \times d} \longrightarrow \text{SelfAttention}(X)$$

The Transformer consumes a **sequence of tokens**. It knows nothing about grids, pixels, or convolutions. This lecture asks a simple question:

can we make an image look like a sequence — and run a plain Transformer on it?

An image is a grid; a Transformer wants a sequence

D DERIVATION

The mismatch is structural:

An image is a grid; a Transformer wants a sequence

D DERIVATION

The mismatch is structural:

CNN view — an image is a **2-D grid**; a conv kernel mixes each pixel with its **local** neighbours.

Transformer view — the input must be a **sequence of vectors** $x_1, \ldots, x_T$, each a **token**.

An image is a grid; a Transformer wants a sequence

D DERIVATION

The mismatch is structural:

CNN view — an image is a **2-D grid**; a conv kernel mixes each pixel with its **local** neighbours.

Transformer view — the input must be a **sequence of vectors** $x_1, \ldots, x_T$, each a **token**.

so we need to turn the $H \times W \times C$ grid into a **short list of visual tokens**

The ViT main idea

D DERIVATION

Vision Transformer (Dosovitskiy et al. 2021): cut the image into **patches** and treat each patch as a token.

The ViT main idea

D DERIVATION

Vision Transformer (Dosovitskiy et al. 2021): cut the image into **patches** and treat each patch as a token.

1. split the image into a grid of $P \times P$ patches;

The ViT main idea

D DERIVATION

Vision Transformer (Dosovitskiy et al. 2021): cut the image into **patches** and treat each patch as a token.

1. split the image into a grid of $P \times P$ patches;
2. **flatten** each patch and **linearly project** it to an embedding;

The ViT main idea

D DERIVATION

Vision Transformer (Dosovitskiy et al. 2021): cut the image into **patches** and treat each patch as a token.

1. split the image into a grid of $P \times P$ patches;
2. **flatten** each patch and **linearly project** it to an embedding;
3. feed the resulting sequence of patch tokens to a **pure Transformer encoder**.

The ViT main idea

D DERIVATION

Vision Transformer (Dosovitskiy et al. 2021): cut the image into **patches** and treat each patch as a token.

1. split the image into a grid of $P \times P$ patches;
2. **flatten** each patch and **linearly project** it to an embedding;
3. feed the resulting sequence of patch tokens to a **pure Transformer encoder**.

no convolutions — a patch is to an image what a word is to a sentence

The image-to-token pipeline

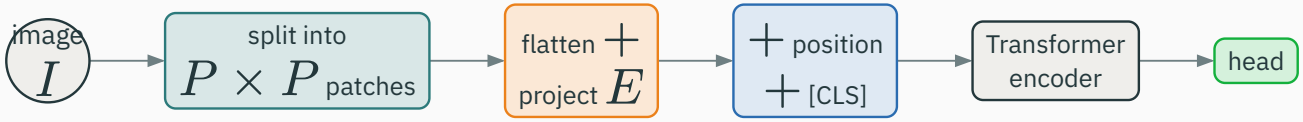
V VISUAL

The whole ViT front-end, end to end:

The image-to-token pipeline

V VISUAL

The whole ViT front-end, end to end:



The image-to-token pipeline

V VISUAL

The whole ViT front-end, end to end:

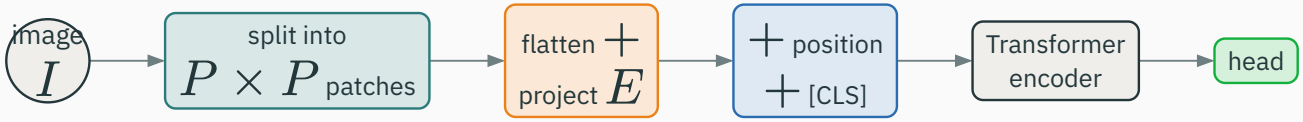


image ⟶ patches ⟶ flatten ⟶ project ⟶ add position & [CLS] ⟶ Transformer ⟶ head

Patchification

How many patches?

D DERIVATION

A patch size P tiles the $H \times W$ image into a grid of non-overlapping squares:

How many patches?

D DERIVATION

A patch size P tiles the $H \times W$ image into a grid of non-overlapping squares:

$$N = \frac{H}{P} \cdot \frac{W}{P} \qquad \text{(patches per column} \times \text{patches per row)}$$

How many patches?

D DERIVATION

A patch size P tiles the $H \times W$ image into a grid of non-overlapping squares:

$$N = \frac{H}{P} \cdot \frac{W}{P} \qquad \text{(patches per column} \times \text{patches per row)}$$

N is the **sequence length** the Transformer sees — usually a few hundred, not thousands

Worked: how many patches?

D DERIVATION

The standard ViT setup — a 224×224 image with patch size $P = 16$:

Worked: how many patches?

D DERIVATION

The standard ViT setup — a 224×224 image with patch size $P = 16$:

$$\frac{224}{16} = 14 \qquad \text{(patches along each side)}$$

Worked: how many patches?

D DERIVATION

The standard ViT setup — a 224×224 image with patch size $P = 16$:

$$\frac{224}{16} = 14 \qquad \text{(patches along each side)}$$

$$N = 14 \times 14 = \mathbf{196} \qquad \text{patch tokens}$$

Worked: how many patches?

D DERIVATION

The standard ViT setup — a 224×224 image with patch size $P = 16$:

$$\frac{224}{16} = 14 \qquad \text{(patches along each side)}$$

$$N = 14 \times 14 = \mathbf{196} \qquad \text{patch tokens}$$

a 224×224 image becomes a sequence of just **196 tokens** — very manageable

What is inside one patch?

D DERIVATION

Each $P \times P$ patch has $P \times P$ pixels, each with C channels — flatten it into one vector:

What is inside one patch?

D DERIVATION

Each $P \times P$ patch has $P \times P$ pixels, each with C channels — flatten it into one vector:

$$x_i \in \mathbb{R}^{P^2 C} \qquad (\text{for } P = 16, C = 3\text{: } 16^2 \cdot 3 = 768)$$

What is inside one patch?

D DERIVATION

Each $P \times P$ patch has $P \times P$ pixels, each with C channels — flatten it into one vector:

$$x_i \in \mathbb{R}^{P^2 C} \qquad (\text{for } P = 16, C = 3\text{: } 16^2 \cdot 3 = 768)$$

Project it to the model width d with a **shared** embedding matrix E:

$$z_i = x_i E + b, \qquad E \in \mathbb{R}^{P^2 C \times d}, \quad z_i \in \mathbb{R}^d$$

What is inside one patch?

D DERIVATION

Each $P \times P$ patch has $P \times P$ pixels, each with C channels — flatten it into one vector:

$$x_i \in \mathbb{R}^{P^2 C} \qquad (\text{for } P = 16, C = 3\text{: } 16^2 \cdot 3 = 768)$$

Project it to the model width d with a **shared** embedding matrix E:

$$z_i = x_i E + b, \qquad E \in \mathbb{R}^{P^2 C \times d}, \quad z_i \in \mathbb{R}^d$$

the **same** E projects **every** patch — one learned linear map, reused N times

Worked: patch-embedding parameters

D DERIVATION

With $P = 16, C = 3$ so $P^2C = 768$, and model width $d = 768$:

Worked: patch-embedding parameters

D DERIVATION

With $P = 16, C = 3$ so $P^2C = 768$, and model width $d = 768$:

$$E \in \mathbb{R}^{768 \times 768} \quad \text{(weights)}, \quad b \in \mathbb{R}^{768} \quad \text{(bias)}$$

Worked: patch-embedding parameters

D DERIVATION

With $P = 16, C = 3$ so $P^2C = 768$, and model width $d = 768$:

$$E \in \mathbb{R}^{768 \times 768} \quad \text{(weights)}, \quad b \in \mathbb{R}^{768} \quad \text{(bias)}$$

$$768^2 + 768 = 589{,}824 + 768 = \mathbf{590{,}592} \quad \text{parameters}$$

Worked: patch-embedding parameters

D DERIVATION

With $P = 16, C = 3$ so $P^2C = 768$, and model width $d = 768$:

$$E \in \mathbb{R}^{768\times768} \quad \text{(weights)}, \quad b \in \mathbb{R}^{768} \quad \text{(bias)}$$

$$768^2 + 768 = 589{,}824 + 768 = \mathbf{590{,}592} \quad \text{parameters}$$

the entire patch embedding is **one** small linear layer — cheap compared to the encoder

Patch embedding is a strided convolution

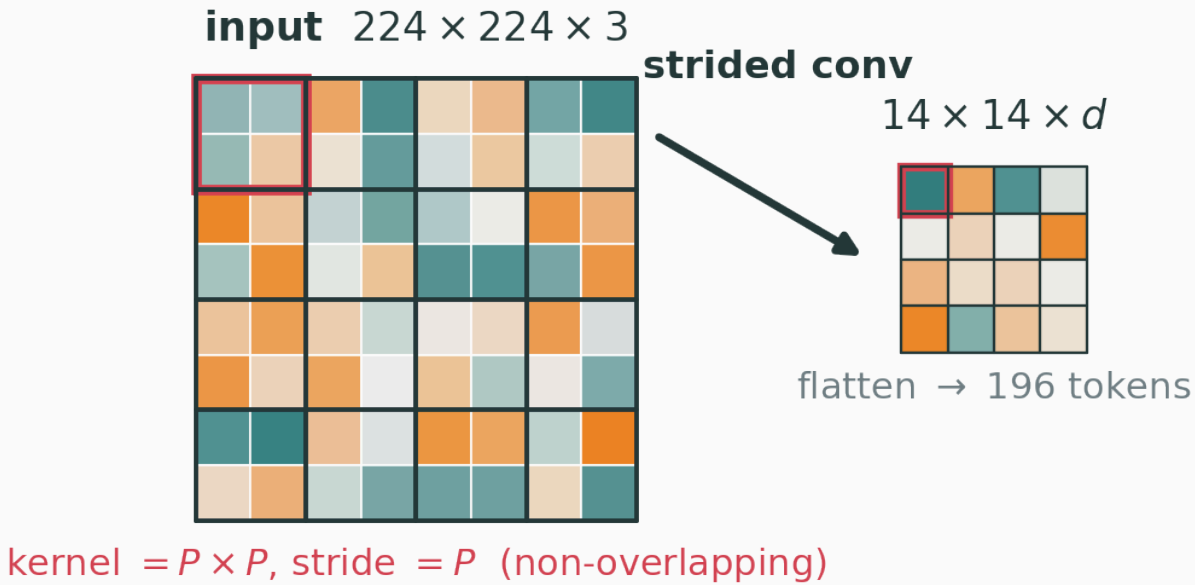
V VISUAL

Flattening every $P \times P$ patch and applying a shared E is **exactly** a convolution:

Patch embedding is a strided convolution

V VISUAL

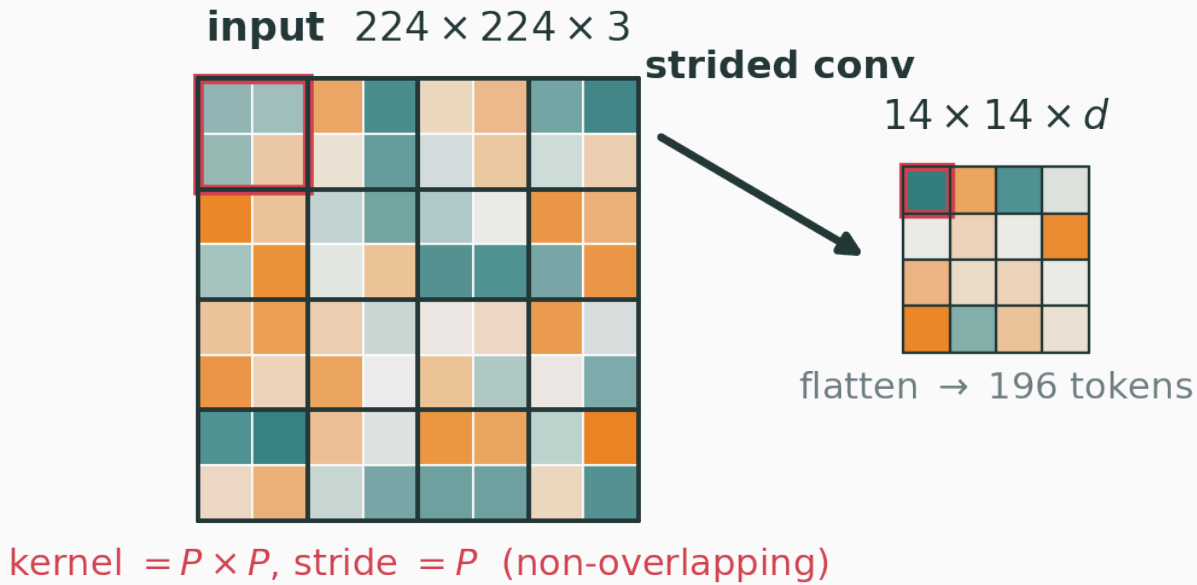
Flattening every $P \times P$ patch and applying a shared E is **exactly** a convolution:



Patch embedding is a strided convolution

V VISUAL

Flattening every $P \times P$ patch and applying a shared E is **exactly** a convolution:



kernel $= P \times P$, stride $= P$, $C_{\text{out}} = d$ – one conv layer turns the image into patch tokens

Patchification, drawn

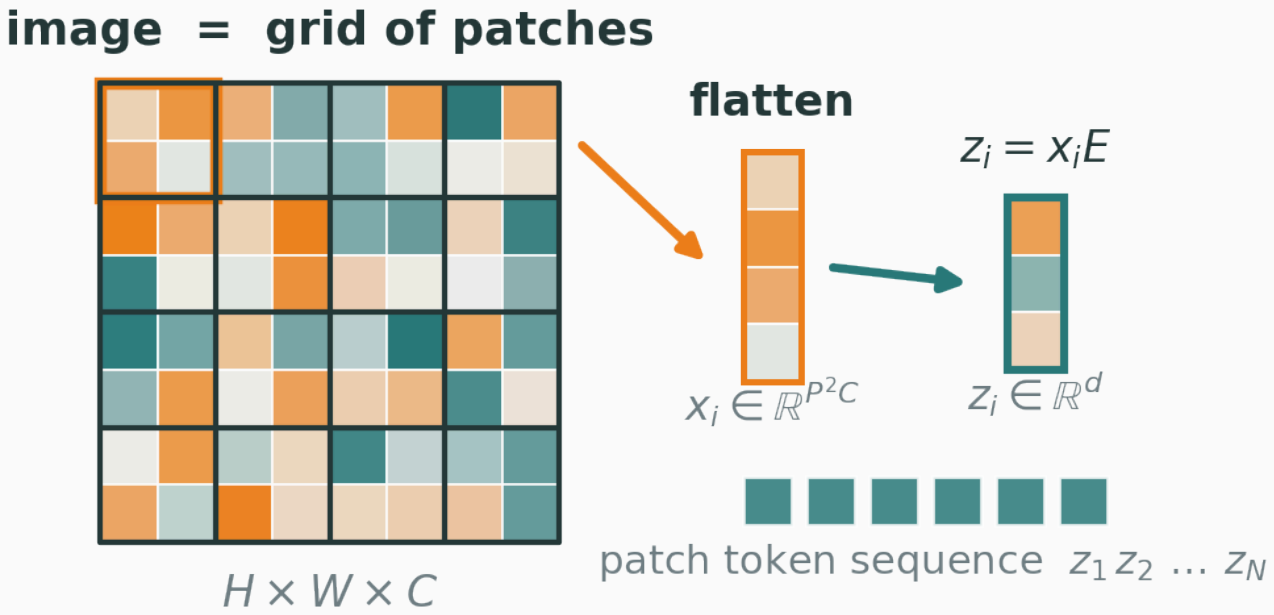
V VISUAL

One patch: cut it out, flatten its pixels, project to a d-dim token — repeat for all N:

Patchification, drawn

V VISUAL

One patch: cut it out, flatten its pixels, project to a d-dim token — repeat for all N:



Interactive: patchification

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/vision-transformer

Drag the patch size P and watch the image re-tile: see $N = (H/P)(W/P)$ change, one patch flatten into $x_i \in \mathbb{R}^{P^2C}$, and project to a token z_i.

The class token and positions

A token to summarize the image

D DERIVATION

Prepend one **learned** vector — the class token $z_{\text{cls}} \in \mathbb{R}^d$ — to the patch sequence:

A token to summarize the image

D DERIVATION

Prepend one **learned** vector — the class token $z_{\text{cls}} \in \mathbb{R}^d$ — to the patch sequence:

$$Z_0 = [\, z_{\text{cls}}; z_1; z_2; \ldots; z_N] \qquad (\text{length } N+1)$$

A token to summarize the image

D DERIVATION

Prepend one **learned** vector — the class token $z_{\text{cls}} \in \mathbb{R}^d$ — to the patch sequence:

$$Z_0 = [\, z_{\text{cls}}; z_1; z_2; \ldots; z_N] \qquad (\text{length } N+1)$$

z_{cls} carries no patch content of its own. Its job is to **gather** information from all patches through the attention layers, and end up as the image's representation.

Reading out the class token

D DERIVATION

Use the final class-token state h_{cls} as the whole-image vector:

Reading out the class token

D DERIVATION

Use the final class-token state h_{cls} as the whole-image vector:

$$p(y \mid I) = \text{softmax}(W_c\, h_{\text{cls}} + b_c)$$

Reading out the class token

D DERIVATION

Use the final class-token state h_{cls} as the whole-image vector:

$$p(y \mid I) = \text{softmax}(W_c\, h_{\text{cls}} + b_c)$$

Or **global-average-pool** the patch outputs:

$$h_{\text{image}} = \frac{1}{N} \sum_{i=1}^{N} h_i$$

Reading out the class token

D DERIVATION

Use the final class-token state h_{cls} as the whole-image vector:

$$p(y \mid I) = \text{softmax}(W_c\, h_{\text{cls}} + b_c)$$

Or **global-average-pool** the patch outputs:

$$h_{\text{image}} = \frac{1}{N} \sum_{i=1}^{N} h_i$$

[CLS] token **or** mean-pool — both turn N tokens into one image vector

Self-attention has no sense of order

D DERIVATION

Attention is a weighted sum over tokens — permute the patches and the outputs just permute too:

Self-attention has no sense of order

D DERIVATION

Attention is a weighted sum over tokens — permute the patches and the outputs just permute too:

$$\text{SelfAttention}([z_1; z_2; z_3]) \quad \text{is a permutation of} \quad \text{SelfAttention}([z_3; z_1; z_2])$$

Self-attention has no sense of order

D DERIVATION

Attention is a weighted sum over tokens — permute the patches and the outputs just permute too:

$$\mathrm{SelfAttention}([z_1; z_2; z_3]) \quad \text{is a permutation of} \quad \mathrm{SelfAttention}([z_3; z_1; z_2])$$

Self-attention is **permutation-equivariant**: on its own it cannot tell the **top-left** patch from the **bottom-right** one. Spatial layout is invisible.

Adding position embeddings

D DERIVATION

Add a learned position vector to every token — patches **and** the class token:

Adding position embeddings

D DERIVATION

Add a learned position vector to every token — patches **and** the class token:

$$Z_0 = [\, z_{\text{cls}}; z_1; \ldots; z_N] + E_{\text{pos}}, \qquad E_{\text{pos}} \in \mathbb{R}^{(N+1)\times d}$$

Adding position embeddings

D DERIVATION

Add a learned position vector to every token — patches **and** the class token:

$$Z_0 = [\, z_{\text{cls}}; z_1; \ldots; z_N] + E_{\text{pos}}, \qquad E_{\text{pos}} \in \mathbb{R}^{(N+1)\times d}$$

ViT uses **learned 1-D** position embeddings, one vector per slot in the flattened patch order. They give attention a location signal; at a new resolution, the learned grid must be adapted (usually by interpolation).

Position choices and changing resolution

Several ways to encode **where** a patch sits:

Position choices and changing resolution

D DERIVATION

Several ways to encode **where** a patch sits:

- **learned** (ViT default) · **separate row / column** · **relative** (Swin) · **rotary**;

Position choices and changing resolution

D DERIVATION

Several ways to encode **where** a patch sits:

- **learned** (ViT default) · **separate row / column** · **relative** (Swin) · **rotary**;

all inject a position signal so attention can use spatial layout

Position choices and changing resolution

D DERIVATION

Several ways to encode **where** a patch sits:

- **learned** (ViT default) · **separate row / column** · **relative** (Swin) · **rotary**;

all inject a position signal so attention can use spatial layout

Changing resolution. Test at a higher resolution → more patches than trained.
Fix: **interpolate** the position embeddings from the 14×14 grid up to, say, 24×24.

The ViT encoder

One ViT block (pre-norm)

D DERIVATION

A ViT block is a standard **pre-norm** Transformer block over the patch sequence Z:

One ViT block (pre-norm)

D DERIVATION

A ViT block is a standard **pre-norm** Transformer block over the patch sequence Z:

$$Z'_\ell = Z_{\ell-1} + \mathrm{MHA}(\mathrm{LN}(Z_{\ell-1}))$$

One ViT block (pre-norm)

D DERIVATION

A ViT block is a standard **pre-norm** Transformer block over the patch sequence Z:

$$Z'_\ell = Z_{\ell-1} + \mathrm{MHA}(\mathrm{LN}(Z_{\ell-1}))$$

$$Z_\ell = Z'_\ell + \mathrm{MLP}(\mathrm{LN}(Z'_\ell))$$

One ViT block (pre-norm)

D DERIVATION

A ViT block is a standard **pre-norm** Transformer block over the patch sequence Z:

$$Z'_\ell = Z_{\ell-1} + \mathrm{MHA}(\mathrm{LN}(Z_{\ell-1}))$$

$$Z_\ell = Z'_\ell + \mathrm{MLP}(\mathrm{LN}(Z'_\ell))$$

identical to the language Transformer — only the **tokens** changed, from words to patches

The per-token MLP

D DERIVATION

Inside each block, an MLP is applied **independently** to every token:

The per-token MLP

D DERIVATION

Inside each block, an MLP is applied **independently** to every token:

$$\mathrm{MLP}(x) = W_2\,\varphi(W_1 x + b_1) + b_2, \qquad d_{\text{hidden}} > d$$

The per-token MLP

D DERIVATION

Inside each block, an MLP is applied **independently** to every token:

$$\mathrm{MLP}(x) = W_2\,\varphi(W_1 x + b_1) + b_2, \qquad d_{\text{hidden}} > d$$

- typical sizing: $d = 768 \to d_{\text{hidden}} = 3072 \to d = 768$ (a $4\times$ expansion);
- φ is usually GELU.

The per-token MLP

D DERIVATION

Inside each block, an MLP is applied **independently** to every token:

$$\mathrm{MLP}(x) = W_2\,\varphi(W_1 x + b_1) + b_2, \qquad d_{\text{hidden}} > d$$

- typical sizing: $d = 768 \to d_{\text{hidden}} = 3072 \to d = 768$ (a $4\times$ expansion);
- φ is usually GELU.

attention **mixes** patches; the MLP then **processes** each patch on its own

Attention between patches

D DERIVATION

Every patch becomes a query, a key and a value — and attends to **all** patches:

Attention between patches

D DERIVATION

Every patch becomes a query, a key and a value — and attends to **all** patches:

$$q_i = W_Q h_i, \quad k_j = W_K h_j, \quad v_j = W_V h_j$$

Attention between patches

D DERIVATION

Every patch becomes a query, a key and a value — and attends to **all** patches:

$$q_i = W_Q h_i, \quad k_j = W_K h_j, \quad v_j = W_V h_j$$

$$\alpha_{ij} = \mathrm{softmax}_j\left(q_i^\top \frac{k_j}{\sqrt{d_k}}\right), \qquad o_i = \sum_j \alpha_{ij}\, v_j$$

Attention between patches

D DERIVATION

Every patch becomes a query, a key and a value — and attends to **all** patches:

$$q_i = W_Q h_i, \quad k_j = W_K h_j, \quad v_j = W_V h_j$$

$$\alpha_{ij} = \mathrm{softmax}_j\left(q_i^\top \frac{k_j}{\sqrt{d_k}}\right), \qquad o_i = \sum_j \alpha_{ij}\, v_j$$

each patch gathers content from **every other patch** — a global operation

Global context from the very first layer

D DERIVATION

This is the key difference from convolution:

Global context from the very first layer

D DERIVATION

This is the key difference from convolution:

CNN — the receptive field grows **slowly** with depth; distant pixels interact only in **late** layers.

ViT — any patch can attend to any other in a **single** block; some heads integrate large distances **early**.

Global context from the very first layer

D DERIVATION

This is the key difference from convolution:

CNN — the receptive field grows **slowly** with depth; distant pixels interact only in **late** layers.

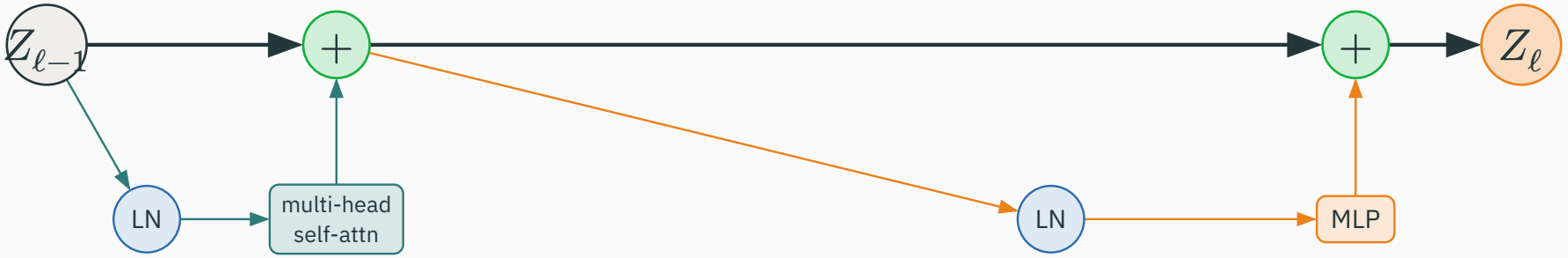
ViT — any patch can attend to any other in a **single** block; some heads integrate large distances **early**.

global from layer one — but **not every** long-range interaction is useful; the model must **learn** which

The ViT encoder block

V VISUAL

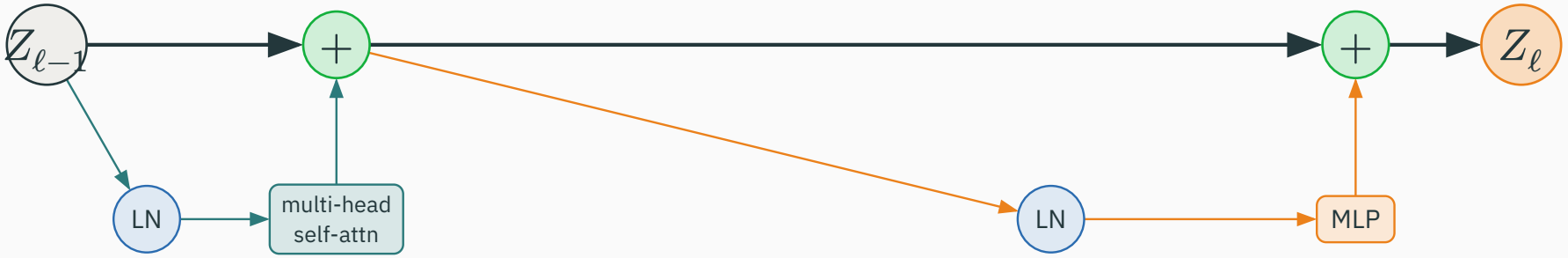
pre-norm: LN sits on the **branch**, the residual spine runs clean



The ViT encoder block

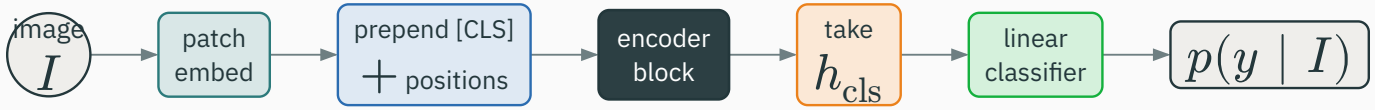
V VISUAL

pre-norm: LN sits on the **branch**, the residual spine runs clean



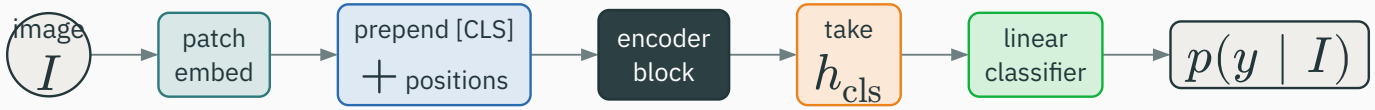
the residual spine runs straight through; each sublayer normalizes, computes, and adds back

The full ViT architecture



$\times L$

The full ViT architecture



$\times L$

patch-embed $\longrightarrow$ prepend [CLS] $+$ positions $\longrightarrow$ L encoder blocks $\longrightarrow$ classify from h_{cls}

Tensor shapes end to end

V VISUAL

Batch B, patches N, width d:

Tensor shapes end to end

V VISUAL

Batch B, patches N, width d:

Stage	Shape	Note
image	$B \times C \times H \times W$	raw pixels
patch embeddings	$B \times N \times d$	N patch tokens
$+$ class token	$B \times (N+1) \times d$	prepend [CLS]
attention matrix	$B \times (N+1) \times (N+1)$	token-to-token
block output	$B \times (N+1) \times d$	same shape in and out

A worked tiny ViT

The setup

D DERIVATION

A 4×4 grayscale “image”, patch size 2 — so $N = 4$ patches, each of dimension 4:

The setup

D DERIVATION

A 4×4 grayscale "image", patch size 2 — so $N = 4$ patches, each of dimension 4:

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

The setup

D DERIVATION

A 4×4 grayscale "image", patch size 2 — so $N = 4$ patches, each of dimension 4:

$$I = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 2 \end{pmatrix}$$

Flatten each 2×2 patch (row-major):

$$p_1 = (1\ \ 1\ \ 1\ \ 1), p_2 = (0\ \ 0\ \ 0\ \ 0), p_3 = (0\ \ 0\ \ 0\ \ 0), p_4 = (2\ \ 2\ \ 2\ \ 2)$$

Projecting the patches

D DERIVATION

Project each 4-dim patch to a 2-dim token with a shared E:

Projecting the patches

D DERIVATION

Project each 4-dim patch to a 2-dim token with a shared E:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad z_i = p_i E$$

Projecting the patches

D DERIVATION

Project each 4-dim patch to a 2-dim token with a shared E:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad z_i = p_i E$$

$$z_1 = (2\ \ 2), \quad z_2 = (0\ \ 0), \quad z_3 = (0\ \ 0), \quad z_4 = (4\ \ 4)$$

Projecting the patches

D DERIVATION

Project each 4-dim patch to a 2-dim token with a shared E:

$$E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad z_i = p_i E$$

$$z_1 = (2\ \ 2), \quad z_2 = (0\ \ 0), \quad z_3 = (0\ \ 0), \quad z_4 = (4\ \ 4)$$

the bright bottom-right patch p_4 becomes the token with the **largest** norm

Class token and one-head attention

D DERIVATION

Prepend $z_{\text{cls}} = (1 \;\; 1)$ (drop positions for clarity). Take $W_Q = W_K = W_V = I$, so $q_{\text{cls}} = (1 \;\; 1)$:

Class token and one-head attention

D DERIVATION

Prepend $z_{\text{cls}} = (1 \;\; 1)$ (drop positions for clarity). Take $W_Q = W_K = W_V = I$, so $q_{\text{cls}} = (1 \;\; 1)$:

$$s_j = q_{\text{cls}}^\top \frac{k_j}{\sqrt{2}} \qquad \text{(scores from the class token to each patch)}$$

Class token and one-head attention

D DERIVATION

Prepend $z_{\text{cls}} = (1 \;\; 1)$ (drop positions for clarity). Take $W_Q = W_K = W_V = I$, so $q_{\text{cls}} = (1 \;\; 1)$:

$$s_j = q_{\text{cls}}^\top \frac{k_j}{\sqrt{2}} \qquad \text{(scores from the class token to each patch)}$$

$$s = \left(\underbrace{(1 \;\; 1) \cdot (2 \;\; 2)}_{4}, 0, 0, \underbrace{(1 \;\; 1) \cdot (4 \;\; 4)}_{8} \right) / \sqrt{2}$$

Class token and one-head attention

D DERIVATION

Prepend $z_{\text{cls}} = (1 \;\; 1)$ (drop positions for clarity). Take $W_Q = W_K = W_V = I$, so $q_{\text{cls}} = (1 \;\; 1)$:

$$s_j = q_{\text{cls}}^\top \frac{k_j}{\sqrt{2}} \qquad \text{(scores from the class token to each patch)}$$

$$s = \left(\underbrace{(1 \;\; 1) \cdot (2 \;\; 2)}_{4}, 0, 0, \underbrace{(1 \;\; 1) \cdot (4 \;\; 4)}_{8} \right) / \sqrt{2}$$

patch 4 scores highest, patch 1 next — the class token will **lean toward the bright region**

Class-token output and the loss

D DERIVATION

Softmax the scores into weights, then blend the values:

Class-token output and the loss

D DERIVATION

Softmax the scores into weights, then blend the values:

$$o_{\text{cls}} = \sum_j \alpha_{\text{cls},j}\, v_j \qquad \text{(a content-dependent summary of the patches)}$$

Class-token output and the loss

D DERIVATION

Softmax the scores into weights, then blend the values:

$$o_{\text{cls}} = \sum_j \alpha_{\text{cls},j}\, v_j \qquad \text{(a content-dependent summary of the patches)}$$

Classify from it, then backprop the cross-entropy:

$$z = W_c\, o_{\text{cls}} + b_c, \quad p = \text{softmax}(z), \quad \mathcal{L} = -\log p_y$$

Class-token output and the loss

D DERIVATION

Softmax the scores into weights, then blend the values:

$$o_{\text{cls}} = \sum_j \alpha_{\text{cls},j}\, v_j \qquad \text{(a content-dependent summary of the patches)}$$

Classify from it, then backprop the cross-entropy:

$$z = W_c\, o_{\text{cls}} + b_c, \quad p = \text{softmax}(z), \quad \mathcal{L} = -\log p_y$$

The gradient flows back into **everything**: the classifier, attention W_Q, W_K, W_V, the MLP, the class token, positions, and the patch projection E.

Interactive: tiny ViT

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/vision-transformer

Paint a tiny image, watch it patchify and project, and see the class token's attention scores s_j shift toward the patches you brighten.

ViT vs CNN

Inductive bias

D DERIVATION

The two architectures build in **different amounts** of prior knowledge about images:

Inductive bias

D DERIVATION

The two architectures build in **different amounts** of prior knowledge about images:

CNN — bakes in **locality**, **translation equivariance** and **hierarchy**. Strong image priors, works from little data.

ViT — bakes in **far less**. A patch can attend anywhere; the model must **learn** spatial structure from data.

Inductive bias

D DERIVATION

The two architectures build in **different amounts** of prior knowledge about images:

CNN — bakes in **locality**, **translation equivariance** and **hierarchy**. Strong image priors, works from little data.

ViT — bakes in **far less**. A patch can attend anywhere; the model must **learn** spatial structure from data.

fewer built-in priors → more flexible, but **hungrier** for data

Data efficiency

D DERIVATION

The ViT trade-off in one line:

Data efficiency

D DERIVATION

The ViT trade-off in one line:

- ViT **underperforms** CNNs on small datasets — it has no locality prior to fall back on;

Data efficiency

D DERIVATION

The ViT trade-off in one line:

- ViT **underperforms** CNNs on small datasets — it has no locality prior to fall back on;
- but **given enough data** (large-scale pretraining + transfer), it **matches or beats** CNNs.

Data efficiency

D DERIVATION

The ViT trade-off in one line:

- ViT **underperforms** CNNs on small datasets — it has no locality prior to fall back on;
- but **given enough data** (large-scale pretraining + transfer), it **matches or beats** CNNs.

DeiT (Touvron et al.) showed strong augmentation, regularization and **distillation** can train a competitive convolution-free Transformer on ImageNet **alone** — no giant private dataset needed.

CNN vs ViT, side by side

V VISUAL

	CNN	ViT
core unit	convolution	self-attention
spatial prior	strong (built in)	weak (learned)
receptive field	grows with depth	global per block
cost vs resolution	linear in pixels	quadratic in patches
data efficiency	strong	recipe-sensitive

CNN vs ViT, side by side

V VISUAL

	CNN	ViT
core unit	convolution	self-attention
spatial prior	strong (built in)	weak (learned)
receptive field	grows with depth	global per block
cost vs resolution	linear in pixels	quadratic in patches
data efficiency	strong	recipe-sensitive

same task, opposite philosophies: hand-built structure vs structure learned from scale

The patch-size trade-off

D DERIVATION

Patch size P sets both the detail and the cost:

The patch-size trade-off

D DERIVATION

Patch size P sets both the detail and the cost:

$$N = \frac{HW}{P^2}, \qquad \text{attention cost} \sim O(N^2 d)$$

The patch-size trade-off

D DERIVATION

Patch size P sets both the detail and the cost:

$$N = \frac{HW}{P^2}, \qquad \text{attention cost} \sim O(N^2 d)$$

- **smaller** $P \to$ **more** patches $\to$ finer spatial detail, but a **longer** sequence;
- longer sequence $\to$ the $N \times N$ attention matrix grows **quadratically**.

The patch-size trade-off

D DERIVATION

Patch size P sets both the detail and the cost:

$$N = \frac{HW}{P^2}, \qquad \text{attention cost} \sim O(N^2 d)$$

- **smaller** $P \to$ **more** patches $\to$ finer spatial detail, but a **longer** sequence;
- longer sequence $\to$ the $N \times N$ attention matrix grows **quadratically**.

resolution of detail vs compute — the central knob of any ViT

Worked: the cost of halving P

D DERIVATION

V VISUAL

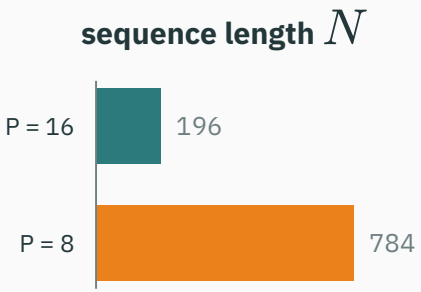
224×224 image, comparing $P = 16$ and $P = 8$:

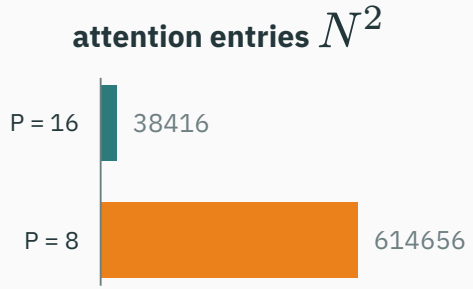
Worked: the cost of halving P

D DERIVATION

V VISUAL

224×224 image, comparing $P = 16$ and $P = 8$:





Worked: the cost of halving P

D DERIVATION V VISUAL

224×224 image, comparing $P = 16$ and $P = 8$:

$P : 16 \to 8$ quadruples N $(196 \to 784)$ and makes the attention matrix $\approx 16 \times$ bigger

Hierarchical ViT (Swin)

Vanilla ViT is awkward for dense vision

D DERIVATION

Classification needs **one** global vector. But **detection** and **segmentation** need more:

Vanilla ViT is awkward for dense vision

D DERIVATION

Classification needs **one** global vector. But **detection** and **segmentation** need more:

- **multiple resolutions** (small and large objects);
- **high-resolution** feature maps (per-pixel prediction);
- **manageable** compute at high resolution.

Vanilla ViT is awkward for dense vision

D DERIVATION

Classification needs **one** global vector. But **detection** and **segmentation** need more:

- **multiple resolutions** (small and large objects);
- **high-resolution** feature maps (per-pixel prediction);
- **manageable** compute at high resolution.

Vanilla ViT keeps a **single, fixed** resolution and pays $O(N^2)$ globally — poorly suited to dense, high-resolution tasks.

A hierarchy of feature maps

D DERIVATION

Swin (Liu et al.) rebuilds the CNN-style pyramid inside a Transformer:

A hierarchy of feature maps

D DERIVATION

Swin (Liu et al.) rebuilds the CNN-style pyramid inside a Transformer:

$$\frac{H}{4} \rightarrow \frac{H}{8} \rightarrow \frac{H}{16} \rightarrow \frac{H}{32}$$

A hierarchy of feature maps

D DERIVATION

Swin (Liu et al.) rebuilds the CNN-style pyramid inside a Transformer:

$$\frac{H}{4} \rightarrow \frac{H}{8} \rightarrow \frac{H}{16} \rightarrow \frac{H}{32}$$

- spatial resolution **shrinks**, channel count **grows** — exactly like a CNN backbone;
- yields multi-scale maps that plug into detection / segmentation heads (FPN).

A hierarchy of feature maps

D DERIVATION

Swin (Liu et al.) rebuilds the CNN-style pyramid inside a Transformer:

$$\frac{H}{4} \rightarrow \frac{H}{8} \rightarrow \frac{H}{16} \rightarrow \frac{H}{32}$$

- spatial resolution **shrinks**, channel count **grows** — exactly like a CNN backbone;
- yields multi-scale maps that plug into detection / segmentation heads (FPN).

a Transformer that **looks like** a conv backbone from the outside

Window attention

D DERIVATION

Instead of global attention, attend only **within** $M \times M$ windows:

Window attention

D DERIVATION

Instead of global attention, attend only **within** $M \times M$ windows:

$$\text{global} : O(N^2) \qquad \longrightarrow \qquad \text{windowed} : O(N \cdot M^2)$$

Window attention

D DERIVATION

Instead of global attention, attend only **within** $M \times M$ windows:

$$\text{global} : O(N^2) \qquad \longrightarrow \qquad \text{windowed} : O(N \cdot M^2)$$

for a **fixed** window size M, cost is **linear** in the number of patches — scalable to high resolution

Shifted windows

V VISUAL

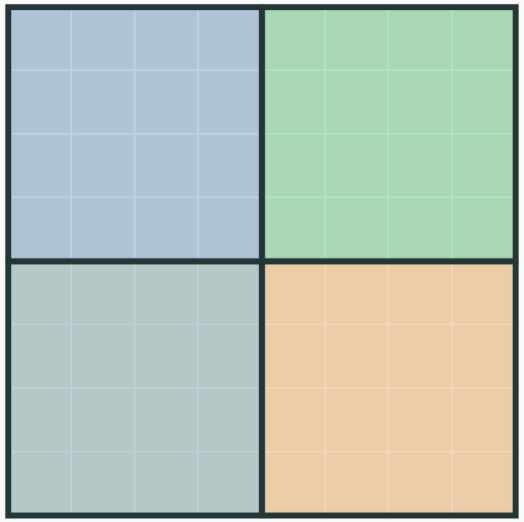
Fixed windows never talk to each other. Swin **shifts** the partition every other block:

Shifted windows

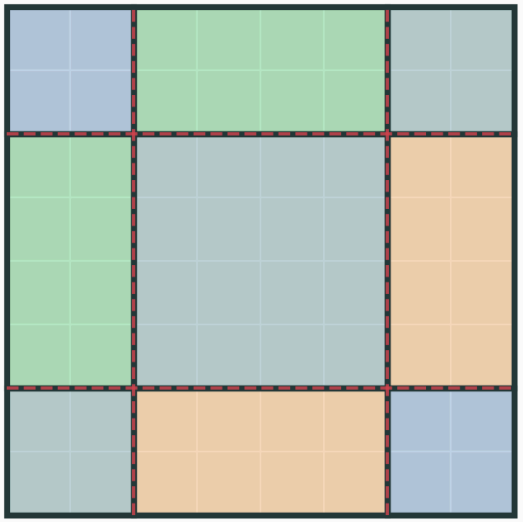
V VISUAL

Fixed windows never talk to each other. Swin **shifts** the partition every other block:

standard windows
attention inside each window



shifted windows (next block)
windows straddle old borders → mix

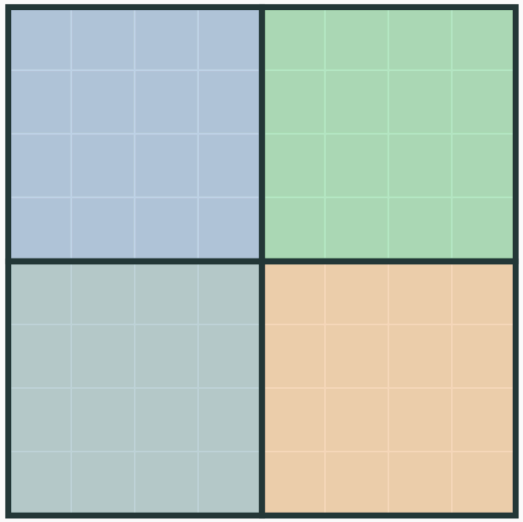


Shifted windows

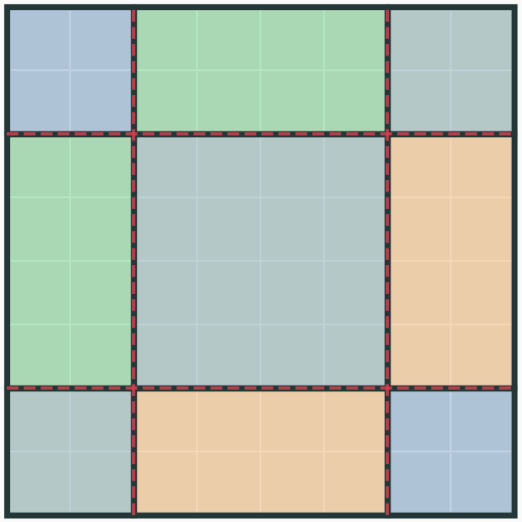
V VISUAL

Fixed windows never talk to each other. Swin **shifts** the partition every other block:

standard windows
attention inside each window



shifted windows (next block)
windows straddle old borders → mix



block 1 uses standard windows; block 2 shifts them so neighbouring windows **exchange** information

Patch merging

D DERIVATION

To go **down** a level of the hierarchy, merge 2×2 neighbouring patches:

Patch merging

D DERIVATION

To go **down** a level of the hierarchy, merge 2×2 neighbouring patches:

$$H \times W \times C \xrightarrow{\text{concat } 2\times 2} \frac{H}{2} \times \frac{W}{2} \times 4C \xrightarrow{\text{linear}} \frac{H}{2} \times \frac{W}{2} \times 2C$$

Patch merging

D DERIVATION

To go **down** a level of the hierarchy, merge 2×2 neighbouring patches:

$$H \times W \times C \xrightarrow{\text{concat } 2\times 2} \frac{H}{2} \times \frac{W}{2} \times 4C \xrightarrow{\text{linear}} \frac{H}{2} \times \frac{W}{2} \times 2C$$

halve the resolution, grow the channels — the Transformer analog of a strided conv / pool

ViT vs Swin

	ViT	Swin
resolution	single, fixed	hierarchical pyramid
attention scope	global (all patches)	local windows + shift
cost in patches	quadratic $O(N^2)$	linear $O(NM^2)$
best for	classification	detection, segmentation

ViT vs Swin

V VISUAL

	ViT	Swin
resolution	single, fixed	hierarchical pyramid
attention scope	global (all patches)	local windows + shift
cost in patches	quadratic $O(N^2)$	linear $O(NM^2)$
best for	classification	detection, segmentation

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/shifted-window

A shifted-window widget: watch windows tile, then shift, so neighbours mix.

Vision pretraining (briefly)

Supervised vs self-supervised

D DERIVATION

How do we **train** a vision backbone before transfer?

Supervised vs self-supervised

D DERIVATION

How do we **train** a vision backbone before transfer?

Supervised — map image $I \to$ label y, minimize $\mathcal{L}_{\text{CE}}$. Needs labels; the label vocabulary is **limited**.

Self-supervised — **no labels**. Build the target from the image itself: mask, augment, or match.

Supervised vs self-supervised

D DERIVATION

How do we **train** a vision backbone before transfer?

Supervised — map image $I \to$ label y, minimize $\mathcal{L}_{\text{CE}}$. Needs labels; the label vocabulary is **limited**.

Self-supervised — **no labels**. Build the target from the image itself: mask, augment, or match.

self-supervision unlocks **web-scale** image data without human annotation

Masked image modeling

D DERIVATION

The BERT idea, ported to patches: **hide** some, **predict** them:

Masked image modeling

D DERIVATION

The BERT idea, ported to patches: **hide** some, **predict** them:

- randomly **mask** a subset of patch tokens;
- the encoder predicts the **content** (or a learned representation) of the masked patches.

Masked image modeling

D DERIVATION

The BERT idea, ported to patches: **hide** some, **predict** them:

- randomly **mask** a subset of patch tokens;
- the encoder predicts the **content** (or a learned representation) of the masked patches.

> reconstruct-the-missing forces the model to learn how patches relate — no labels required

Self-distilled features (DINOv2)

D DERIVATION

Match **two views** of the same image through a **teacher** and a **student**:

Self-distilled features (DINOv2)

D DERIVATION

Match **two views** of the same image through a **teacher** and a **student**:

- student and teacher see **different augmentations**; the student is trained to **agree** with the teacher;
- the teacher is a slow-moving average of the student — no labels anywhere.

Self-distilled features (DINOv2)

D DERIVATION

Match **two views** of the same image through a **teacher** and a **student**:

- student and teacher see **different augmentations**; the student is trained to **agree** with the teacher;
- the teacher is a slow-moving average of the student — no labels anywhere.

DINOv2 (Oquab et al.) produces strong **general-purpose** features this way — useful across many tasks **without** task labels.

Probing vs finetuning

D DERIVATION

Two ways to **use** a pretrained backbone with parameters θ:

Probing vs finetuning

D DERIVATION

Two ways to **use** a pretrained backbone with parameters θ:

Method	**Update**	**Measures**
linear probe	freeze θ; train $y = Wh + b$	direct usability of features
finetune	update **all** of θ	best task accuracy

Probing vs finetuning

D DERIVATION

Two ways to **use** a pretrained backbone with parameters θ:

Method	Update	Measures
linear probe	freeze θ; train $y = Wh + b$	direct usability of features
finetune	update **all** of θ	best task accuracy

a strong **linear probe** means the features are already **linearly** useful — the mark of good pretraining

Image-text learning: CLIP

Why connect images and text?

D DERIVATION

A single label is a thin signal. A **caption** is far richer:

Why connect images and text?

D DERIVATION

A single label is a thin signal. A **caption** is far richer:

“a brown dog running through snow”

Why connect images and text?

D DERIVATION

A single label is a thin signal. A **caption** is far richer:

“a brown dog running through snow”

object **dog** · attribute **brown** · action **running** · scene **snow** · relations **through**

Why connect images and text?

D DERIVATION

A single label is a thin signal. A **caption** is far richer:

“a brown dog running through snow”

object **dog** · attribute **brown** · action **running** · scene **snow** · relations **through**

far more supervision than the one word “dog” — without manually annotating every image

Why connect images and text?

D DERIVATION

A single label is a thin signal. A **caption** is far richer:

“a brown dog running through snow”

object **dog** · attribute **brown** · action **running** · scene **snow** · relations **through**

far more supervision than the one word “dog” — without manually annotating every image

web pairs are still noisy and require collection, filtering, and careful attention to provenance, bias, and consent

The dual encoder

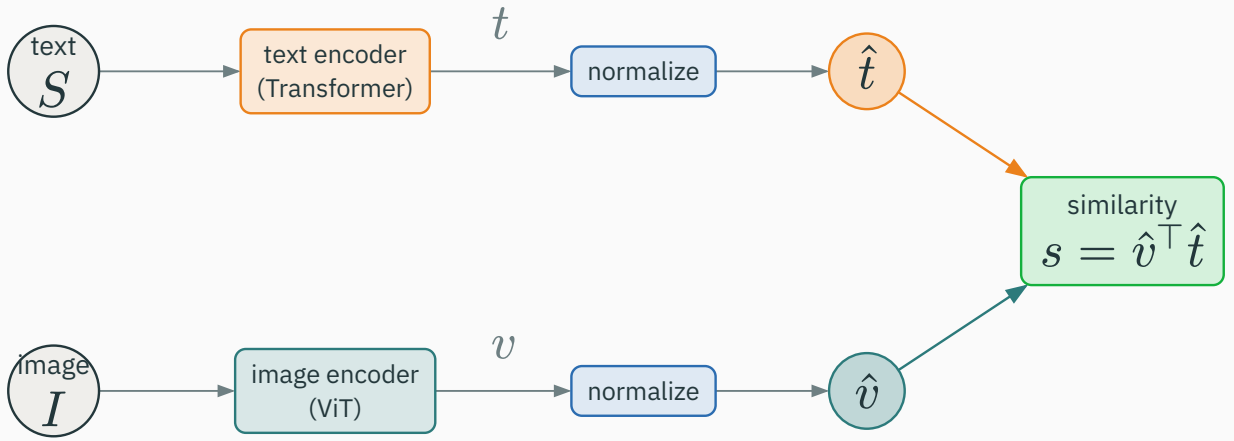
V VISUAL

CLIP (Radford et al. 2021): one encoder per modality, projected into a **shared** space:

The dual encoder

V VISUAL

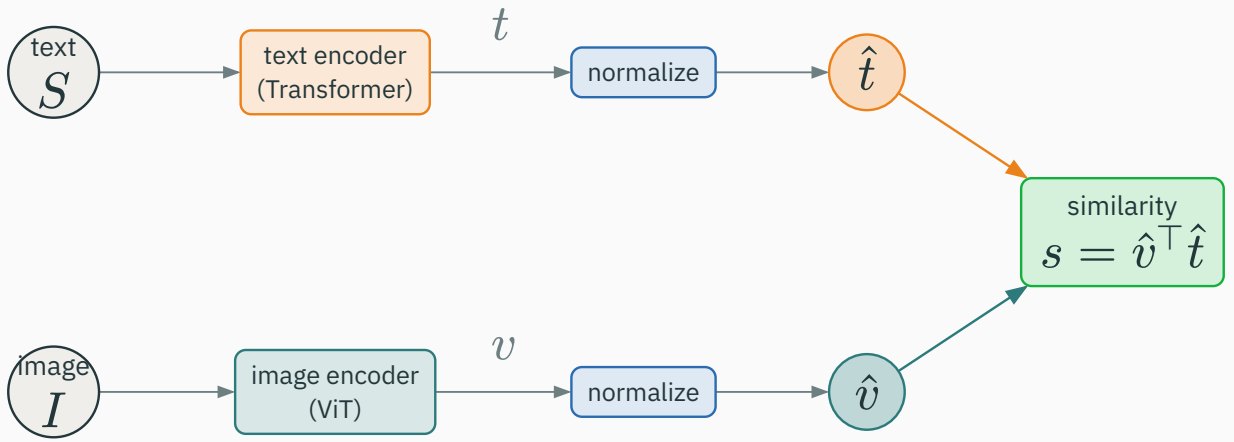
CLIP (Radford et al. 2021): one encoder per modality, projected into a **shared** space:



The dual encoder

V VISUAL

CLIP (Radford et al. 2021): one encoder per modality, projected into a **shared** space:



$v = f_{\text{img}(I)}, t = f_{\text{text}(S)}$; normalize $\hat{v} = v/\|v\|, \hat{t} = t/\|t\|$; compare by cosine $\hat{v}^\top \hat{t}$

How CLIP trains

D DERIVATION

Take a **batch** of B matched image-text pairs. Score **every** image against **every** caption:

How CLIP trains

D DERIVATION

Take a **batch** of B matched image-text pairs. Score **every** image against **every** caption:

$$S_{ij} = \hat{v}_i^\top \hat{t}_j / \tau \qquad (\tau = \text{learned temperature})$$

How CLIP trains

D DERIVATION

Take a **batch** of B matched image-text pairs. Score **every** image against **every** caption:

$$S_{ij} = \hat{v}_i^\top \hat{t}_j / \tau \qquad (\tau = \text{learned temperature})$$

- the B **correct** pairs sit on the **diagonal** S_{ii};
- all $B^2 - B$ off-diagonal pairs are **mismatches**.

How CLIP trains

D DERIVATION

Take a **batch** of B matched image-text pairs. Score **every** image against **every** caption:

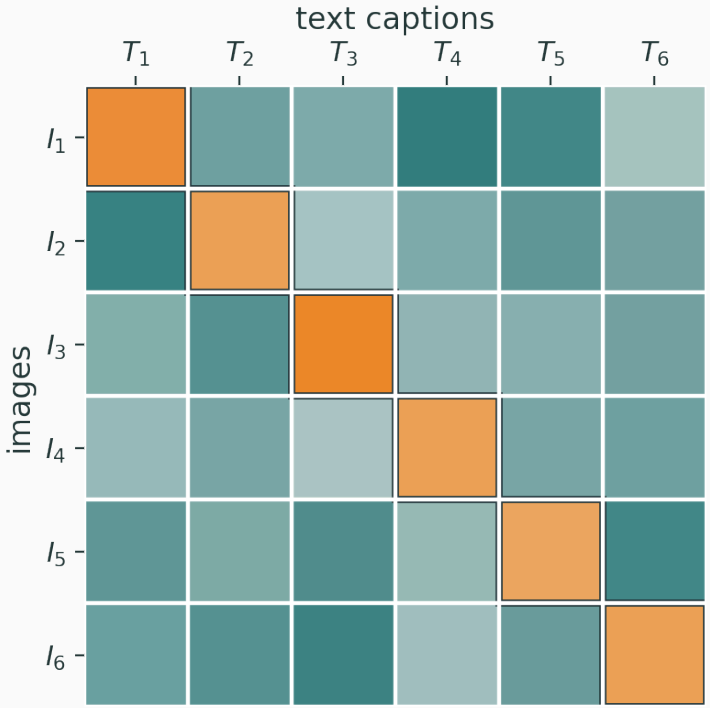
$$S_{ij} = \hat{v}_i^\top \hat{t}_j / \tau \qquad (\tau = \text{learned temperature})$$

- the B **correct** pairs sit on the **diagonal** S_{ii};
- all $B^2 - B$ off-diagonal pairs are **mismatches**.

train the model to pick the matching caption for each image — and vice versa

The similarity matrix

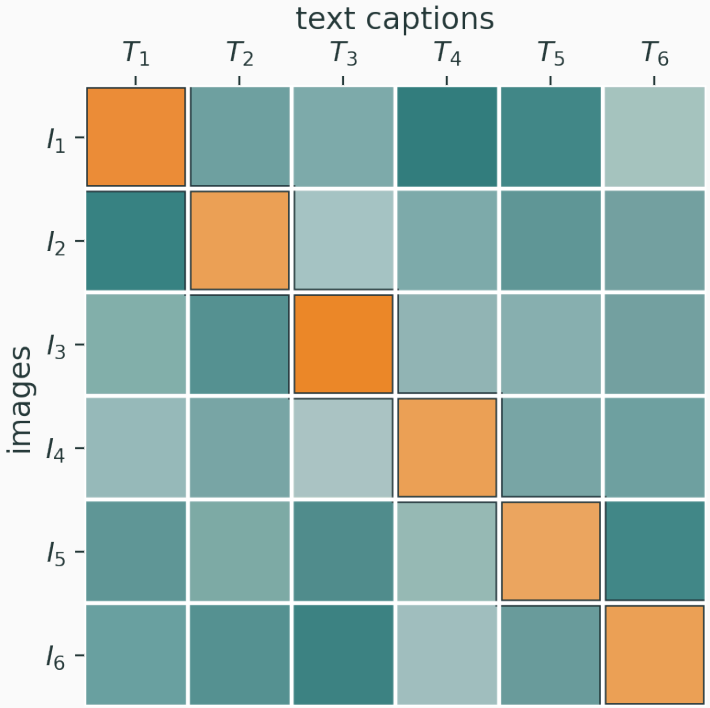
$S_{ij} = \hat{v}_i^\top \hat{t}_j / \tau$ — maximize the diagonal



The similarity matrix

V VISUAL

$S_{ij} = \hat{v}_i^\top \hat{t}_j / \tau$ — maximize the diagonal



a $B \times B$ grid of similarities; push the **diagonal** up and everything **off-diagonal** down

The contrastive loss

D DERIVATION

For image i, softmax across the **row** (all captions) and reward the correct one:

The contrastive loss

D DERIVATION

For image i, softmax across the **row** (all captions) and reward the correct one:

$$\mathcal{L}_{I \to T}^{(i)} = -\log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}$$

The contrastive loss

D DERIVATION

For image i, softmax across the **row** (all captions) and reward the correct one:

$$\mathcal{L}_{I\to T}^{(i)} = -\log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}$$

Symmetrically, softmax down the **column** for the text-to-image direction, then average:

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_{I\to T} + \mathcal{L}_{T\to I})$$

The contrastive loss

D DERIVATION

For image i, softmax across the **row** (all captions) and reward the correct one:

$$\mathcal{L}_{I\to T}^{(i)} = -\log \frac{\exp(S_{ii})}{\sum_{j=1}^{B} \exp(S_{ij})}$$

Symmetrically, softmax down the **column** for the text-to-image direction, then average:

$$\mathcal{L} = \frac{1}{2}(\mathcal{L}_{I\to T} + \mathcal{L}_{T\to I})$$

a cross-entropy where the “classes” are the **other items in the batch**

Worked: contrastive loss

D DERIVATION

A batch of $B = 3$ with similarity matrix S (rows = images, columns = captions):

Worked: contrastive loss

D DERIVATION

A batch of $B = 3$ with similarity matrix S (rows = images, columns = captions):

$$S = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 4 \end{pmatrix}$$

Worked: contrastive loss

D DERIVATION

A batch of $B = 3$ with similarity matrix S (rows = images, columns = captions):

$$S = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 4 \end{pmatrix}$$

Image 1: softmax across row $1 = (3, 1, 0)$:

$$p(T_1 \mid I_1) = \frac{e^{3}}{e^{3} + e^{1} + e^{0}} = \frac{20.09}{23.81} \approx \mathbf{0.844}$$

Worked: contrastive loss

D DERIVATION

A batch of $B = 3$ with similarity matrix S (rows = images, columns = captions):

$$S = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 2 & 1 \\ 1 & 0 & 4 \end{pmatrix}$$

Image 1: softmax across row $1 = (3, 1, 0)$:

$$p(T_1 \mid I_1) = \frac{e^3}{e^3 + e^1 + e^0} = \frac{20.09}{23.81} \approx \mathbf{0.844}$$

$$\mathcal{L}^{(1)}_{I \to T} = -\log(0.844) \approx \mathbf{0.170}$$

Negatives come from the batch

D DERIVATION

CLIP has no separate negative set — the batch **is** the negatives:

Negatives come from the batch

D DERIVATION

CLIP has no separate negative set — the batch **is** the negatives:

- for image i, the positive is caption T_i; the negatives are all $T_{j \neq i}$ **in the same batch**;
- **bigger batch** $=$ **more negatives** $=$ a harder, more informative task.

Negatives come from the batch

D DERIVATION

CLIP has no separate negative set — the batch **is** the negatives:

- for image i, the positive is caption T_i; the negatives are all $T_{j \neq i}$ **in the same batch**;
- **bigger batch** $=$ **more negatives** $=$ a harder, more informative task.

False negatives. Two different images in a batch may genuinely match the same caption — the loss still pushes them apart. Large, diverse batches make this rare.

Interactive: contrastive loss

I INTERACTIVE

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/clip-zero-shot

Edit a batch of image-text similarities and watch the row/column softmax, the diagonal light up, and the symmetric contrastive loss respond.

Zero-shot classification

Classify with no trained head

D DERIVATION

CLIP can classify into **new** classes it never had a labeled head for. Turn each class into text:

Classify with no trained head

D DERIVATION

CLIP can classify into **new** classes it never had a labeled head for. Turn each class into text:

$$\{\text{cat}, \text{dog}, \text{car}\} \xrightarrow{\text{prompt}} \text{a photo of a } \{\text{class}\}$$

Classify with no trained head

D DERIVATION

CLIP can classify into **new** classes it never had a labeled head for. Turn each class into text:

$$\{\text{cat}, \text{dog}, \text{car}\} \xrightarrow{\text{prompt}} \text{a photo of a } \{\text{class}\}$$

$t_{\text{cat}}, \quad t_{\text{dog}}, \quad t_{\text{car}}$ (encode each prompt with the text tower)

Classify with no trained head

D DERIVATION

CLIP can classify into **new** classes it never had a labeled head for. Turn each class into text:

$$\{\text{cat}, \text{dog}, \text{car}\} \xrightarrow{\text{prompt}} \text{a photo of a } \{\text{class}\}$$

$t_{\text{cat}}, \quad t_{\text{dog}}, \quad t_{\text{car}}$ (encode each prompt with the text tower)

the class names become **text embeddings** — no supervised classifier needed

Compare the image to each class

D DERIVATION

Encode the image once, then score it against every class text:

Compare the image to each class

D DERIVATION

Encode the image once, then score it against every class text:

$$v = f_{\text{img}(I)}, \qquad s_c = \hat{v}^\top \hat{t}_c / \tau \quad \left(\hat{v}, \hat{t}_c \text{ unit-normalized, as in training}\right)$$

Compare the image to each class

D DERIVATION

Encode the image once, then score it against every class text:

$$v = f_{\text{img}(I)}, \qquad s_c = \hat{v}^\top \hat{t}_c / \tau \quad \left(\hat{v}, \hat{t}_c \text{ unit-normalized, as in training}\right)$$

$$p(c \mid I) = \text{softmax}_c(s_c)$$

Compare the image to each class

D DERIVATION

Encode the image once, then score it against every class text:

$$v = f_{\text{img}(I)}, \qquad s_c = \hat{v}^\top \hat{t}_c / \tau \quad \left(\hat{v}, \hat{t}_c \text{ unit-normalized, as in training}\right)$$

$$p(c \mid I) = \text{softmax}_c(s_c)$$

add a new class? just add its **prompt** — no retraining, no labeled examples

Worked: zero-shot prediction

D DERIVATION

Similarities of the image to three class prompts, $s = (4, 2, -1)$:

Worked: zero-shot prediction

D DERIVATION

Similarities of the image to three class prompts, $s = (4, 2, -1)$:

$$\text{softmax}(4, 2, -1) = \frac{e^4, e^2, e^{-1}}{e^4 + e^2 + e^{-1}} \approx (\mathbf{0.876}, 0.118, 0.006)$$

Worked: zero-shot prediction

D DERIVATION

Similarities of the image to three class prompts, $s = (4, 2, -1)$:

$$\mathrm{softmax}(4, 2, -1) = \frac{e^4, e^2, e^{-1}}{e^4 + e^2 + e^{-1}} \approx (\mathbf{0.876}, 0.118, 0.006)$$

predict cat — the highest-similarity class, with no trained classifier

Prompts and retrieval

D DERIVATION

Two practical extensions of the same idea:

Prompts and retrieval

D DERIVATION

Two practical extensions of the same idea:

- **prompt ensembling** — average the text embeddings of several templates (“a photo of a {c}”, “a blurry photo of a {c}”, …) for a more robust class vector;

Prompts and retrieval

D DERIVATION

Two practical extensions of the same idea:

- **prompt ensembling** — average the text embeddings of several templates (“a photo of a {c}”, “a blurry photo of a {c}”, …) for a more robust class vector;
- **image-text retrieval** — rank a gallery by $\cos(v_i, t_q)$ to find images for a text query (or captions for an image).

Prompts and retrieval

D DERIVATION

Two practical extensions of the same idea:

- **prompt ensembling** — average the text embeddings of several templates (“a photo of a {c}”, “a blurry photo of a {c}”, …) for a more robust class vector;
- **image-text retrieval** — rank a gallery by $\cos(v_i, t_q)$ to find images for a text query (or captions for an image).

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/clip-zero-shot

Type your own class names, watch the prompts encode, and see the image’s softmax over classes update live.

Fusion and vision-language models

Dual encoders: strengths and a limit

D DERIVATION

The dual encoder is fast and flexible — but shallow in how it combines modalities:

Dual encoders: strengths and a limit

D DERIVATION

The dual encoder is fast and flexible — but shallow in how it combines modalities:

Strengths — precompute embeddings, **fast retrieval**, zero-shot transfer, a simple contrastive objective.

Limit — image and text meet **only** at the final $\hat{v}^\top \hat{t}$. Too shallow for "is the **red cup** left of the **blue plate**?"

Dual encoders: strengths and a limit

D DERIVATION

The dual encoder is fast and flexible — but shallow in how it combines modalities:

Strengths — precompute embeddings, **fast retrieval**, zero-shot transfer, a simple contrastive objective.

Limit — image and text meet **only** at the final $\hat{v}^\top \hat{t}$. Too shallow for "is the **red cup** left of the **blue plate**?"

compositional, relational questions need **deeper** image-text interaction

Cross-attention fusion

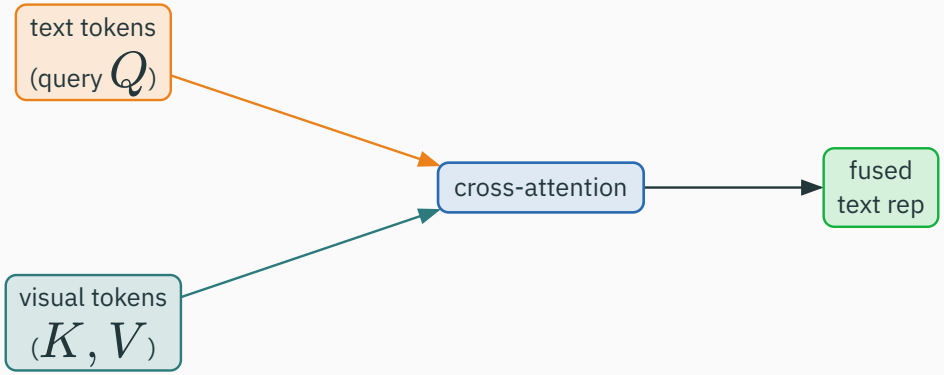
V VISUAL

Let text **query** the visual tokens: Q from text, K, V from the image.

Cross-attention fusion

V VISUAL

Let text **query** the visual tokens: Q from text, K, V from the image.



$$\text{CrossAttention}(Q_{\text{text}}, K_{\text{vision}}, V_{\text{vision}})$$

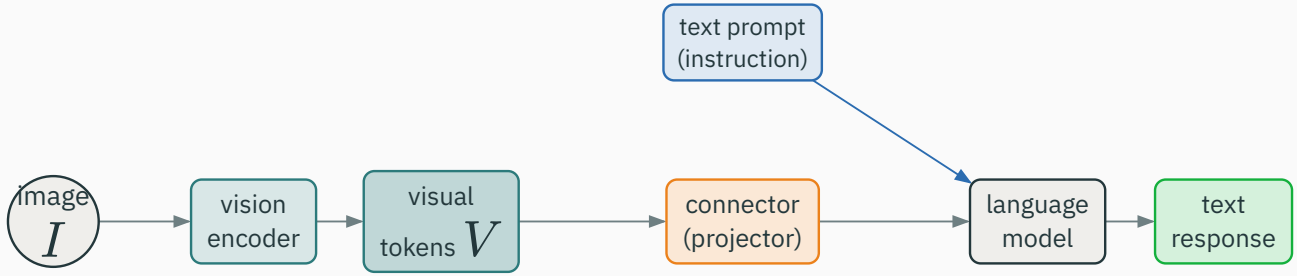
The modern VLM pattern

V VISUAL

Connect a **vision encoder** to a **language model** through a small **connector**:

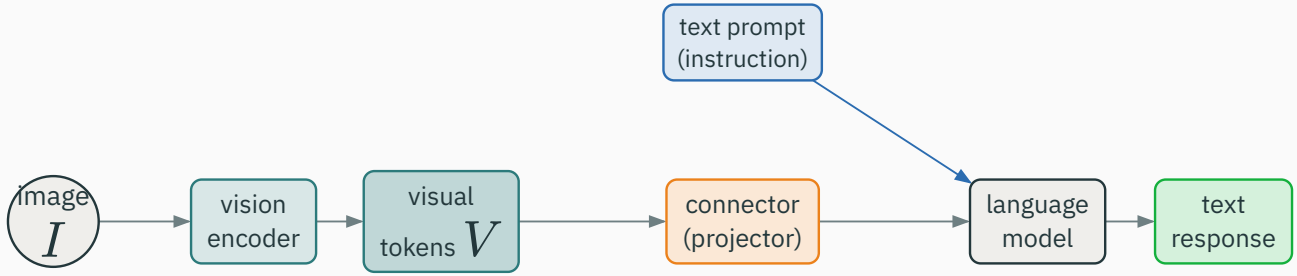
The modern VLM pattern

Connect a **vision encoder** to a **language model** through a small **connector**:



The modern VLM pattern

Connect a **vision encoder** to a **language model** through a small **connector**:



$VW_p \rightarrow$ prefix

image $\rightarrow$ vision encoder $\rightarrow$ visual tokens $\rightarrow$ connector $\rightarrow$ LM token space $\rightarrow$ text

The connector and the objective

D DERIVATION

The connector maps visual features into the LM’s token space as a **visual prefix**:

The connector and the objective

D DERIVATION

The connector maps visual features into the LM's token space as a **visual prefix**:

$$\tilde{V} = VW_p + b, \qquad W_p \in \mathbb{R}^{d_v \times d_{\text{LM}}}$$

The connector and the objective

D DERIVATION

The connector maps visual features into the LM's token space as a **visual prefix**:

$$\tilde{V} = VW_p + b, \qquad W_p \in \mathbb{R}^{d_v \times d_{\text{LM}}}$$

Then generate text with the **same next-token objective** — now conditioned on the image:

$$\mathcal{L} = -\sum_t \log p(y_t \mid y_{<t}, I)$$

The connector and the objective

D DERIVATION

The connector maps visual features into the LM's token space as a **visual prefix**:

$$\tilde{V} = VW_p + b, \qquad W_p \in \mathbb{R}^{d_v \times d_{\text{LM}}}$$

Then generate text with the **same next-token objective** — now conditioned on the image:

$$\mathcal{L} = -\sum_t \log p(y_t \mid y_{<t}, I)$$

nothing new in the loss — the image just becomes extra **context tokens** for the LM

BLIP-2: bridge two frozen models

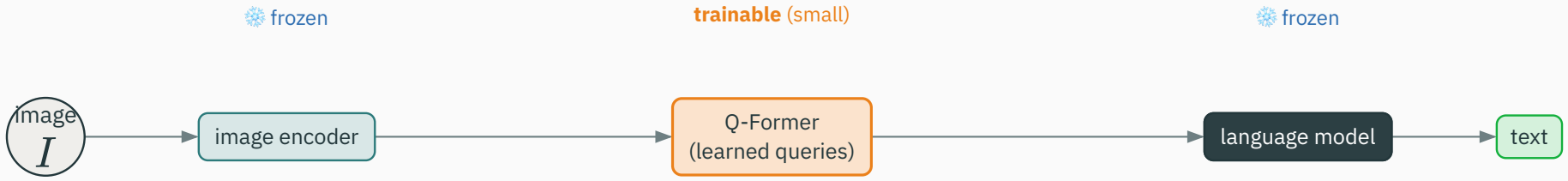
V VISUAL

BLIP-2 (Li et al.): keep the image encoder and the LM **frozen**, learn only a light bridge:

BLIP-2: bridge two frozen models

V VISUAL

BLIP-2 (Li et al.): keep the image encoder and the LM **frozen**, learn only a light bridge:

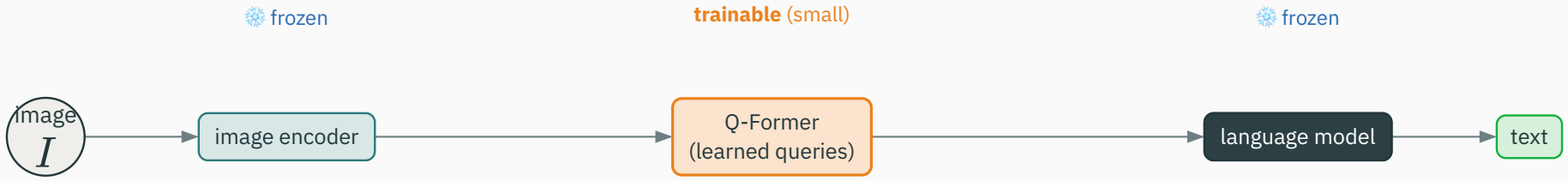


reuse strong **frozen** models, learn a small **connector**

BLIP-2: bridge two frozen models

V VISUAL

BLIP-2 (Li et al.): keep the image encoder and the LM **frozen**, learn only a light bridge:



reuse strong **frozen** models, learn a small **connector**

reuse strong unimodal models; train **only** the small Q-Former between them

Q-Former intuition

D DERIVATION

A handful of **learned queries** $q_1, \ldots, q_M$ pull the image down to a few language-ready tokens:

Q-Former intuition

D DERIVATION

A handful of **learned queries** $q_1, \ldots, q_M$ pull the image down to a few language-ready tokens:

$$z_m = \mathrm{CrossAttention}(q_m, V, V) \qquad \text{(queries attend to image features } V\text{)}$$

Q-Former intuition

D DERIVATION

A handful of **learned queries** $q_1, \ldots, q_M$ pull the image down to a few language-ready tokens:

$$z_m = \mathrm{CrossAttention}(q_m, V, V) \qquad \text{(queries attend to image features } V\text{)}$$

compress hundreds of patches into M (e.g. 32) **language-relevant** visual tokens

LLaVA: a simple, strong recipe

D DERIVATION

LLaVA (Liu et al.) shows a projector can be almost trivial:

LLaVA: a simple, strong recipe

D DERIVATION

LLaVA (Liu et al.) shows a projector can be almost trivial:

$$\text{CLIP vision encoder} \quad \rightarrow \quad \text{linear / MLP projector} \quad \rightarrow \quad \text{LLM}$$

LLaVA: a simple, strong recipe

D DERIVATION

LLaVA (Liu et al.) shows a projector can be almost trivial:

$$\text{CLIP vision encoder} \quad \rightarrow \quad \text{linear / MLP projector} \quad \rightarrow \quad \text{LLM}$$

- project CLIP patch features straight into the LM's embedding space;
- **visual instruction tuning** — train on (image, instruction, response) triples.

LLaVA: a simple, strong recipe

D DERIVATION

LLaVA (Liu et al.) shows a projector can be almost trivial:

$$\text{CLIP vision encoder} \quad \rightarrow \quad \text{linear / MLP projector} \quad \rightarrow \quad \text{LLM}$$

- project CLIP patch features straight into the LM's embedding space;
- **visual instruction tuning** — train on (image, instruction, response) triples.

a tiny projector plus instruction data turns an LLM into a capable vision-language model

Visual instruction tuning

D DERIVATION

Train the model to **follow instructions about images**, supervising only the **response**:

Visual instruction tuning

D DERIVATION

Train the model to **follow instructions about images**, supervising only the **response**:

$$\mathcal{L} = -\sum_{t} \log p(y_t \mid y_{<t}, I, \text{instruction})$$

Visual instruction tuning

D DERIVATION

Train the model to **follow instructions about images**, supervising only the **response**:

$$\mathcal{L} = -\sum_{t} \log p(y_t \mid y_{<t}, I, \text{instruction})$$

The instruction tokens are **masked out** of the loss — we only teach the model to produce the **answer**, not to re-predict the question.

What to freeze, what to train

D DERIVATION

The main design axis of a VLM:

What to freeze, what to train

D DERIVATION

The main design axis of a VLM:

Strategy	Trade-off
freeze vision + LM, train connector	cheapest; least forgetting; limited alignment
freeze vision, finetune LM	better alignment; risk of forgetting
LoRA on the LM	cheap adaptation, keeps base intact
joint finetune	strongest; most compute, most forgetting

What to freeze, what to train

D DERIVATION

The main design axis of a VLM:

Strategy	Trade-off
freeze vision + LM, train connector	cheapest; least forgetting; limited alignment
freeze vision, finetune LM	better alignment; risk of forgetting
LoRA on the LM	cheap adaptation, keeps base intact
joint finetune	strongest; most compute, most forgetting

INTERACTIVE

↗ nipunbatra.github.io/interactive-articles/vlm

Send an image through a vision encoder and connector into an LM and watch visual tokens become a prefix the model attends to.

Limitations

Spatial and fine-grained reasoning

D DERIVATION

Fluent captions can still get the **details** wrong:

Spatial and fine-grained reasoning

D DERIVATION

Fluent captions can still get the **details** wrong:

- **counting** objects; locating **small** objects; **left / right** and precise coordinates;
- reading **fine text**; exact spatial relations.

Spatial and fine-grained reasoning

D DERIVATION

Fluent captions can still get the **details** wrong:

- **counting** objects; locating **small** objects; **left / right** and precise coordinates;
- reading **fine text**; exact spatial relations.

fluent language about an image $\neq$ **grounded** understanding of it

Hallucination

D DERIVATION

The classic failure of vision-language models:

Hallucination

D DERIVATION

The classic failure of vision-language models:

Hallucination — the model produces **plausible, fluent** text that the image does **not** support (an object that isn't there, a wrong colour or count).

Hallucination

D DERIVATION

The classic failure of vision-language models:

Hallucination — the model produces **plausible, fluent** text that the image does **not** support (an object that isn't there, a wrong colour or count).

Cause — a **strong language prior** (what usually goes together) combined with **imperfect grounding** in the actual pixels.

Hallucination

D DERIVATION

The classic failure of vision-language models:

Hallucination — the model produces **plausible, fluent** text that the image does **not** support (an object that isn't there, a wrong colour or count).

Cause — a **strong language prior** (what usually goes together) combined with **imperfect grounding** in the actual pixels.

language fluency $\neq$ visual correctness

Data, bias, and resolution

D DERIVATION

Data, bias, and resolution

D DERIVATION

- **data & bias** — web captions are **noisy**, and carry **stereotypes** and **imbalance** straight into the model;

Data, bias, and resolution

D DERIVATION

- **data & bias** — web captions are **noisy**, and carry **stereotypes** and **imbalance** straight into the model;
- **resolution bottleneck** — high-res images make **many** tokens ($N = HW/P^2$), expensive to feed an LLM;

Data, bias, and resolution

D DERIVATION

- **data & bias** — web captions are **noisy**, and carry **stereotypes** and **imbalance** straight into the model;
- **resolution bottleneck** — high-res images make **many** tokens ($N = HW/P^2$), expensive to feed an LLM;

mitigations: downsample · pool · learned query tokens · image **tiling**

Evaluation

D DERIVATION

Different tasks, different metrics:

Evaluation

D DERIVATION

Different tasks, different metrics:

Task	Metric
classification	accuracy
retrieval	Recall@k
captioning	CIDEr / BLEU / human
visual question answering	VQA accuracy

Evaluation

D DERIVATION

Different tasks, different metrics:

Task	Metric
classification	accuracy
retrieval	Recall@k
captioning	CIDEr / BLEU / human
visual question answering	VQA accuracy

automatic metrics are imperfect — human evaluation still matters for open-ended generation

Summary

One pipeline, several models

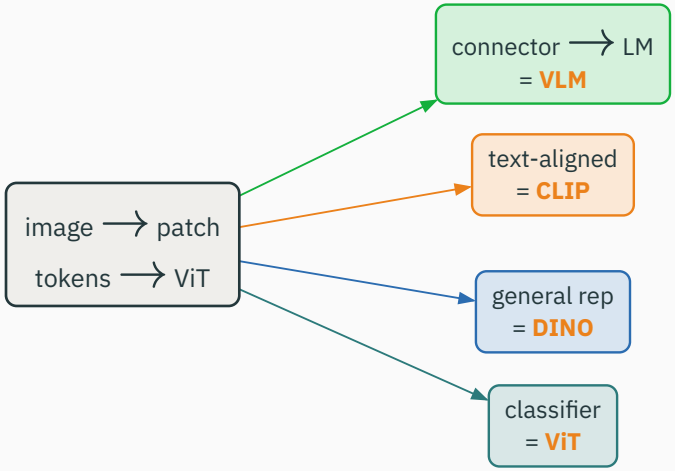
V VISUAL

Every model in this lecture starts the **same way** — patchify, then a Transformer:

One pipeline, several models

V VISUAL

Every model in this lecture starts the **same way** — patchify, then a Transformer:



Architecture comparison

Model	Main output	Training signal
ViT	class label	supervised labels
self-sup ViT (DINOv2)	general features	self-supervision, no labels
CLIP	image-text similarity	contrastive on pairs
BLIP-2	text from image	frozen towers + Q-Former
LLaVA	instruction response	visual instruction tuning

The key equations

Idea	Equation
patch sequence	$Z_0 = [z_{\text{cls}}; z_1; \ldots; z_N] + E_{\text{pos}}$
attention	$A = \text{softmax}\left(QK^\top/\sqrt{d}\right)$
CLIP similarity	$S_{ij} = \hat{v}_i^\top \hat{t}_j/\tau$
multimodal LM	$p(y_{1:T} \mid I) = \prod_t p(y_t \mid y_{<t}, I)$

Final mental model — one idea

ViT: patches become tokens; self-attention lets regions exchange information.

CLIP: images and language are aligned in one shared space.

VLMs: visual tokens are fed to a language model.

one front-end — patchify — powers classification, features, alignment and generation

A Transformer sees an image as a **sequence of patches**.

Next: **Representation Learning & Autoencoders** → VAEs → GANs → diffusion.