

ViT, U-Net, Dense Prediction

Donald Shenaj

donald.shenaj@di.unipi.it

Oct 22, 2026

Lecture Outline

1. Dense Prediction Tasks
2. Encoder-Decoder Architectures
3. Vision Transformers

Beyond Classification: Dense Prediction Tasks

Classification: one label for the whole image. Many vision tasks need **one prediction per pixel**:

- **Semantic segmentation**: a class label per pixel
- **Instance segmentation**: class and instance identity for each object pixel
- **Panoptic segmentation**: semantic labels for "stuff" and instance IDs for "things"
- **Depth estimation**: a continuous depth value per pixel
- **Optical flow**: a 2D motion vector per pixel
- **Keypoint / pose heatmaps**: a per-pixel likelihood map



Figure: [*Cordts et al. 2016 – Cityscapes; Ranftl et al. 2022 – MiDaS; Butler et al. 2012 – MPI-Sintel; Cao et al. 2017 – OpenPose*]

Task: Semantic / Instance / Panoptic Segmentation

- **Input**: one RGB image, $H \times W \times 3$ – **GT**: $H \times W$ class id per pixel (semantic); add a per-pixel **instance id** for countable "things" (instance); unify both for "stuff" + "things" (panoptic)
- A trained model (right) should match the GT (middle)



Figure: [*Zhao et al. 2017 – PSPNet, Fig. 8, Cityscapes*] – Input → Ground truth → Prediction

Task: Monocular Depth Estimation

- **Input**: a *single* RGB image, $H \times W \times 3$ – no stereo pair, no LiDAR at inference time
- **GT**: $H \times W$ array, one **continuous depth** value per pixel – here from an RGB-D sensor; real sensors only cover part of the scene and disagree on scale



Figure: [*Eigen et al. 2014, Fig. 3, NYU Depth*] – Input → Ground truth → Prediction

Task: Optical Flow

- **Input**: *two* consecutive RGB frames ($H \times W \times 3$)
- **GT**: $H \times W \times 2$ array, a motion vector (u, v) per pixel – color-coded (hue = direction, saturation = magnitude)
- The flow field (third panel) is color-coded with the wheel on the right; real dense flow GT is nearly impossible to collect – mostly *synthetic* training data [*Baker et al. 2011 – Middlebury, IJCV*]



Figure: [*visionbook*]

Task: Keypoint / Pose Estimation

- **Input**: one RGB image, $H \times W \times 3$ (one or more people)
- **GT**: K keypoint coordinates (x_k, y_k) per person, annotated the same way the output (right) is drawn here, usually *trained* as K per-pixel likelihood heatmaps, one per joint, peaked at the true location



Figure: [*Cao et al. 2017 – OpenPose, Fig. 2*] – Input → predicted skeleton (same format as GT annotations)

Dense Prediction: Same Shape, Different Content

Common constraint: every task outputs one array at (roughly) **input resolution** – but what each pixel *holds* is completely different:

Task	**Per-pixel output**	**Type**
Semantic segmentation	C class scores	discrete (softmax)
Depth estimation	1 value	continuous, > 0
Optical flow	2D vector (u, v)	continuous, signed
Pose / keypoints	K likelihood maps	continuous, peaked

Consequence: for most tasks, the same *encoder-decoder* structure can be used: only the **task head** (1×1 conv to the right number of output channels) and the **loss** need to change

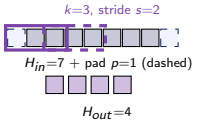
Why Plain CNNs Are Not Enough

- A classification CNN (e.g. ResNet, VGG) is a **funnel**: conv + pooling/stride progressively *downsample* the spatial map while increasing channel depth
  - $224 \times 224 \times 3 \to \cdots \to 7 \times 7 \times 2048 \to 1$ vector
- This trade-off (resolution $\downarrow$ for semantics/receptive field $\uparrow$) answers "what is in the image" well, and "what is *where*" poorly
- By the last layer, fine spatial detail (edges, boundaries) is gone
- **Dense prediction needs a way back up** to the input resolution

Shape Arithmetic: Conv Output Size & Receptive Field

Output size (kernel k, stride s, padding p, dilation d):

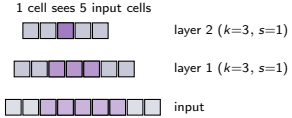
$$H_{out} = \left\lfloor \frac{H_{in} + 2p - d(k-1) - 1}{s} \right\rfloor + 1$$



- **Dilation:** $d > 1$ enlarges the effective kernel $k_{\text{eff}} = d(k-1)+1$ without extra parameters or inherent downsampling (DeepLab's atrous convolution).

Receptive field after L layers (kernel k_l, stride s_l):

$$r_L = r_0 + \sum_{l=1}^{L} (k_l - 1) \prod_{i<l} s_i, \qquad r_0 = 1$$



- **Downsampling** gains context at the cost of spatial resolution.

The key design trade-off in dense prediction is how to give each pixel enough global context without losing the spatial detail needed to localize it accurately.

Encoder-Decoder Architectures for Dense Prediction

- **Encoder**: a (pretrained) classification-style backbone progressively downsamples, building rich *semantic* features
- **Decoder**: progressively *upsamples* those features back to input resolution
- **Skip connections**: fuse high-resolution, low-level encoder features into the decoder to recover fine spatial detail
- **Task head**: a 1×1 conv maps decoder features to the desired output (class logits, depth, ...)

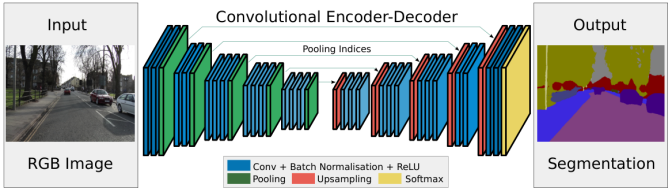
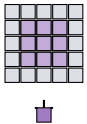


Figure: [*Badrinarayanan et al. 2015 – SegNet*]

What is a 1×1 Convolution?

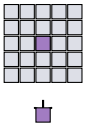
Usual conv rule, just k=1: the output at (h, w) depends *only* on the input at (h, w).

3×3 conv



merges a 3×3 *spatial* neighborhood

1×1 conv



merges *only the channels* at this one pixel

$$y_{h,w} = W x_{h,w} + b, \quad W \in \mathbb{R}^{C_{out} \times C_{in}}\text{: } \textbf{merges information across channels} \text{ at that pixel}$$

Used for: changing channel count (bottleneck/projection) or the final **task head** (features $\rightarrow$ class logits/depth/flow) without touching H, W

“Fully Connected Layers”



Yann LeCun
April 6, 2015 ·

In Convolutional Nets, there is no such thing as "fully-connected layers". There are only convolution layers with 1x1 convolution kernels and a full connection table.

It's a too-rarely-understood fact that ConvNets don't need to have a fixed-size input. You can train them on inputs that happen to produce a single output vector (with no spatial extent), and then apply them to larger images. Instead of a single output vector, you then get a spatial map of output vectors. Each vector sees input windows at different locations on the input.

In that scenario, the "fully connected layers" really act as 1x1 convolutions.

Figure: `https://www.facebook.com/yann.lecun/posts/10152820758292143`

Fully Convolutional Networks (FCN)

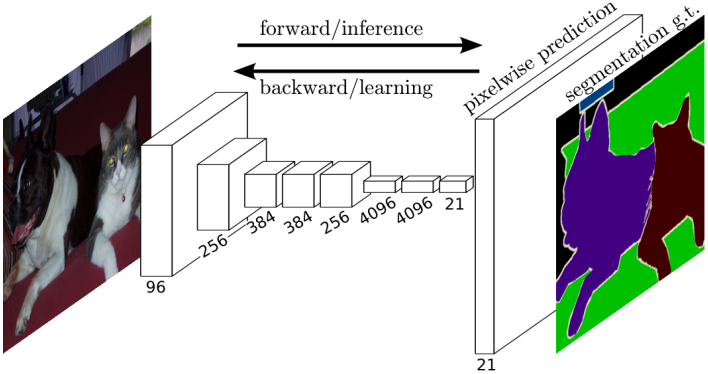


Figure: [*Long, Shelhamer & Darrell, 2015 – FCN, Fig. 1*]

Fully Convolutional Networks (FCN)

- **Key idea:** replace a classifier's final FC layers with equivalent convolutions $\Rightarrow$ accepts varying input sizes and outputs a coarse spatial score map
- A **transposed convolution**, initialized to bilinear upsampling, restores spatial resolution – but upsampling a $32\times$ downsampled prediction cannot recover lost detail
- **FCN-8s fix:** progressively fuse score maps from pool5, pool4, and pool3 before final upsampling, combining deep *semantic* with shallow *spatial* detail – an early, influential use of **skip connections** for dense prediction

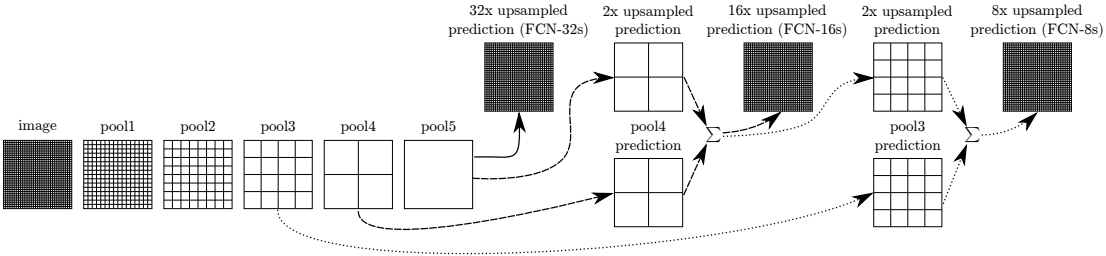


Figure: [*Long et al., 2015 – FCN-32s/16s/8s skip architecture*]

U-Net: Backbone for Semantic Segmentation

- Classic **encoder-decoder** with **skip connections**
- Originally proposed for **biomedical** (cell/membrane) segmentation
- Contracting path builds semantic features while expansive path recovers spatial resolution
- Became a general-purpose dense-prediction backbone far beyond its original domain

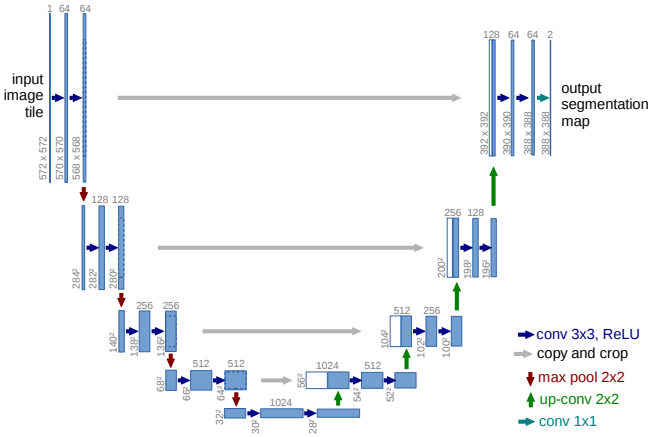


Figure: [*Ronneberger et al. 2015 – U-Net, Fig. 1*]

U-Net: Building Blocks

- **Contracting path** (encoder): repeated [conv 3×3 + ReLU] $\times 2$, then 2×2 max-pool – doubles channels, halves resolution at each stage
- **Expansive path** (decoder): 2×2 up-conv, then **concatenate** with the matching encoder feature map (skip connection), then [conv 3×3 + ReLU] $\times 2$
- Final 1×1 conv $\rightarrow$ per-pixel class scores
- Symmetric "U" shape, trained end-to-end with pixel-wise loss

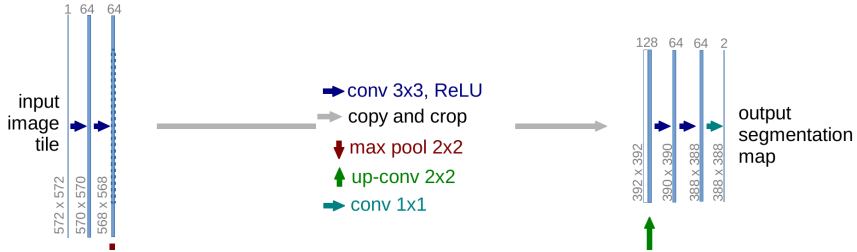


Figure: [*Ronneberger et al. 2015 – U-Net, Fig. 1 (detail)*] – first encoder block (left) and last decoder block (right)

Upsampling Operators: Deconv vs. Resize + Conv

- **Nearest/bilinear interpolation + conv:** fixed, non-learned resize followed by a learnable conv to refine – cheap, smooth, no checkerboard artifacts
- **Transposed convolution ("deconv"):** learnable upsampling, but **uneven kernel overlap** can produce **checkerboard artifacts**
- In practice: U-Net/FCN variants use transposed conv or upsample+conv for stability

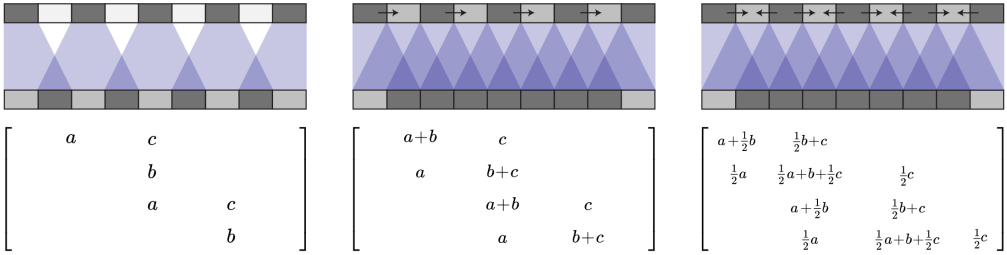


Figure: [*Odena, Dumoulin, Olah – Distill, "Deconvolution and Checkerboard Artifacts"*]

Upsampling Operators: Sub-Pixel Convolution

- **Sub-pixel / pixel-shuffle conv:** convolve at *low* resolution into r^2 extra channels, then rearrange channels into spatial resolution – no learned upsampling filter needed
- Efficient (all compute stays at low resolution) and avoids the uneven-overlap artifacts of transposed convolution
- Popular for super-resolution; also usable as a general upsampling block in dense-prediction decoders

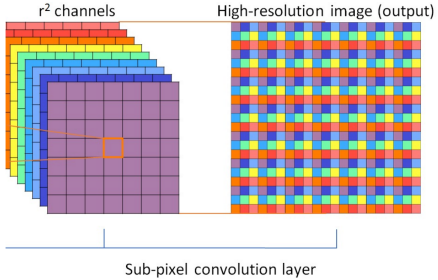


Figure: [*Shi et al. 2016 – ESPCN, Fig. 1*]

Monocular Depth Estimation: MiDaS

- **Task**: predict a dense depth value per pixel from a *single* RGB image: encoder-decoder with a 1-channel regression head
- **Supervision problem**: no sensor measures "true" depth everywhere – ground truth comes from *incompatible* sources (LiDAR/RGB-D = metric, stereo = unknown scale & shift, SfM = no scale at all)
- **Fix** [*Ranftl et al. 2022 – MiDaS*]: train on *many* such datasets at once with a **scale-and-shift-invariant loss** – align prediction to each sample's own scale/shift before computing the loss
- **Result**: one model that generalizes *zero-shot* to datasets never seen during training



Figure: [*Ranftl et al. 2022 – MiDaS, Fig. 1*] – RGB input (top) → predicted inverse depth (bottom)

Optical Flow: RAFT

- **Task**: predict a 2D motion vector (u, v) per pixel between *two* consecutive frames
- **Why plain encoder-decoder isn't enough**: it never compares the two images – needs an explicit **matching** step: a **4D correlation volume**, the dot product of every feature in frame 1 against every feature in frame 2
- **RAFT** [*Teed & Deng 2020*]: looks up this volume and **iteratively refines** the flow with a recurrent update (10+ iterations) instead of predicting it in one shot
- **Supervision**: real per-pixel flow ground truth is nearly impossible to collect – trained almost entirely on *synthetic* data (FlyingChairs, MPI-Sintel)

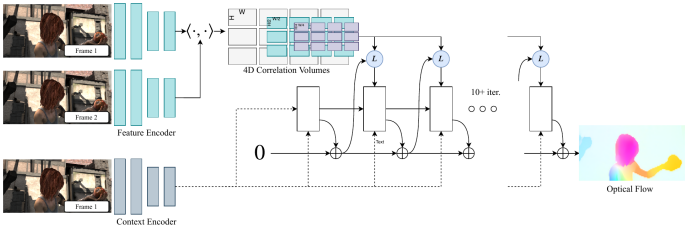


Figure: [*Teed & Deng 2020 – RAFT, Fig. 1*]

Losses for Dense Prediction

- **Pixel-wise cross-entropy**: sum/average CE over every pixel: default for semantic segmentation
- **Focal loss**: for severe class imbalance (e.g. small lesion vs. large background); scales CE by $(1 - p_t)^\gamma$, down-weighting easy, already-confident pixels so that hard (often rare-class) ones dominate the gradient
- **Dice loss / Soft-IoU loss**: optimizes a differentiable (soft) approximation of the overlap between predicted and ground-truth masks; robust to imbalance since it ignores true negatives (often combined with CE)
- **Regression tasks** (depth, flow): L1/L2 or robust variants, scale-invariant loss for monocular depth, often with edge-aware smoothness regularizers

Cross-entropy (per pixel i, classes c):

$$\mathcal{L}_{\text{CE}} = -\sum_i \sum_c y_{i,c} \log \hat{p}_{i,c}$$

Focal loss (γ down-weights easy pixels):

$$\mathcal{L}_{\text{focal}} = -\sum_i (1 - \hat{p}_i)^\gamma \log \hat{p}_i$$

Dice loss (from the IoU overlap formula):

$$\mathcal{L}_{\text{Dice}} = 1 - \frac{2\sum_i \hat{p}_i y_i + \epsilon}{\sum_i \hat{p}_i^2 + \sum_i y_i^2 + \epsilon}$$

Evaluating Dense Predictions

- **Loss $\neq$ metric**: a loss must be differentiable to drive training; a metric just needs to be meaningful for comparing models
- **Segmentation – mIoU**: per class IoU $= \dfrac{|\hat{Y} \cap Y|}{|\hat{Y} \cup Y|}$ averaged over classes, more robust to class imbalance, unlike pixel accuracy
- **Depth** – AbsRel $=$ mean $\frac{|\hat{d}-d|}{d}$ and RMSE $= \sqrt{\text{mean}(\hat{d}-d)^2}$, often computed in log-depth
- **Optical flow** – End-Point-Error (EPE): mean Euclidean distance between predicted and ground-truth flow vectors, per pixel
- **Pose** – PCK: % of predicted keypoints within a scale-normalized threshold distance of ground truth

An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale

Alexey Dosovitskiy[*,†], **Lucas Beyer**[*], **Alexander Kolesnikov**[*], **Dirk Weissenborn**[*], **Xiaohua Zhai**[*], **Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby**[*,†]

[*]equal technical contribution, [†]equal advising

Google Research, Brain Team

{adosovitskiy, neilhoulsby}@google.com

Abstract

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.[1]

Vision Transformer (ViT)

Core idea

- Split image into fixed-size **patches** (e.g. 16×16); flatten and linearly project each into a token
- Prepend a learnable `[CLS]` token – borrowed from **BERT** [*Devlin et al. 2019*], and add positional embeddings
- Feed tokens through standard **Transformer encoder** blocks (multi-head self-attention + MLP)

Beyond classification

- ViT is an **encoder**: it can slot into the same encoder-decoder recipe we used for FCN/U-Net
- SETR, SegFormer, DPT pair a transformer encoder with a dense-prediction decoder

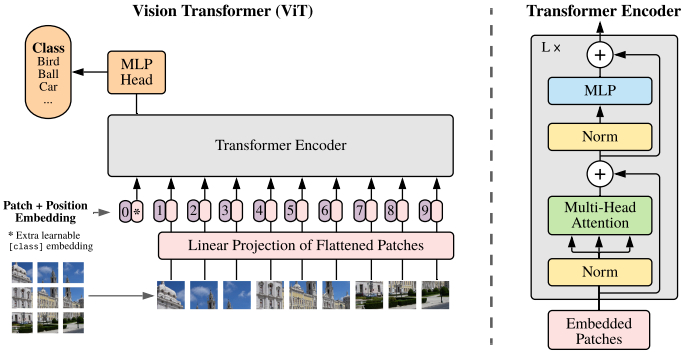


Figure: [*Dosovitskiy et al. 2021 – ViT, Fig. 1*]

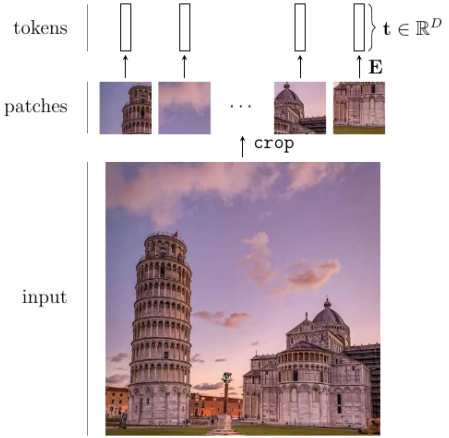
ViT: Patch Embedding

Step by step: split the image into a grid of $N{=}(H/P){\times}(W/P)$ fixed-size $P{\times}P$ patches $\rightarrow$ flatten each into a P^2C-dim vector $\rightarrow$ **linearly project** with the same matrix E for every patch into a D-dim token.

- Prepend a learnable `[CLS]` token $x_{\text{cls}} \in \mathbb{R}^D$; its final state is used for classification
- Add a learnable positional embedding E_{pos} to *every* token (incl. `[CLS]`), otherwise the transformer has no information about where each patch came from
- Result: a sequence of $N{+}1$ tokens, each D-dim, ready for the Transformer encoder
  e.g. 224^2 image, $P{=}16$: $N{=}196$ patches $\rightarrow$ 197 tokens

$$z_0 = [\ \underbrace{x_{\text{cls}}}_{\texttt{[CLS]}};\ \underbrace{x_p^1 E}_{\text{patch 1}};\ \ldots;\ \underbrace{x_p^N E}_{\text{patch } N}\ ] + \underbrace{E_{\text{pos}}}_{\text{learned}}$$

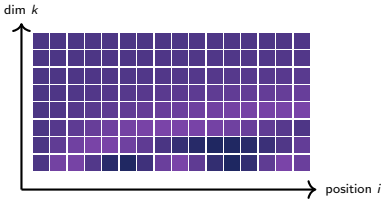
$$E \in \mathbb{R}^{(P^2C)\times D}, \quad E_{\text{pos}} \in \mathbb{R}^{(N+1)\times D}$$



Positional Embeddings

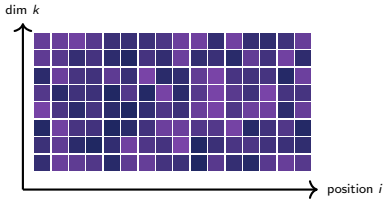
Why needed: self-attention is *permutation-equivariant*: if the patch tokens are shuffled, the outputs are simply shuffled in the same way, so the model has no notion of where each patch sits in the image. Adding E_{pos} to each token breaks this symmetry and injects the spatial information that attention alone cannot recover.

Sinusoidal (fixed) [*Vaswani et al. 2017*]



- Fixed function of i: low dims oscillate fast, high dims slowly
- No parameters; defined for any position

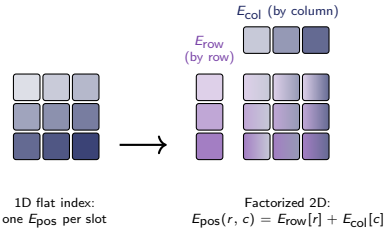
Learned absolute (ViT default)



- One learnable vector per position, no imposed structure
- Fixed-size table: new resolution $\Rightarrow$ interpolate

Positional Embeddings for 2D

Vanilla ViT flattens the patch grid into a 1D sequence and typically uses learned absolute embeddings indexed by the flattened position



- **Factorized 2D (row + col):** separate learned $E_{\text{row}}, E_{\text{col}}$, summed per patch – $O(H+W)$ params instead of $O(HW)$
- **Relative position bias** [*Liu et al. 2021 – Swin*]: a learned bias $B(\Delta i, \Delta j)$ added to the attention logits, indexed by relative offset – translation-invariant
- **Rotary (RoPE)** [*Su et al. 2021*]: rotates Q and K by an angle $\propto$ position, so $q^\top k$ depends only on the relative offset – standard in LLMs, extended to 2D in some ViT variants

Why it matters: segmentation operates at varying resolutions. Learned 1D absolute embeddings typically require reshaping into a 2D grid and interpolating when resolution changes (as in DPT), whereas relative position schemes, such as Swin's bias, can adapt more naturally to different input sizes.

ViT: Self-Attention

Each token (an embedded image patch) attends to every token, including itself, and combines their information. Learned linear projections transform each token into:

- **Query**: what this token is looking for
- **Key**: features used to determine relevance to a query
- **Value**: information this token contributes to other tokens
- **Attention weight** = scaled query–key similarities, normalized by softmax
- **Output** = weighted sum of all value vectors

$\Rightarrow$ Global receptive field from the first layer: global self-attention enables direct interactions between all patch tokens, whereas standard CNNs aggregate local information across successive layers.

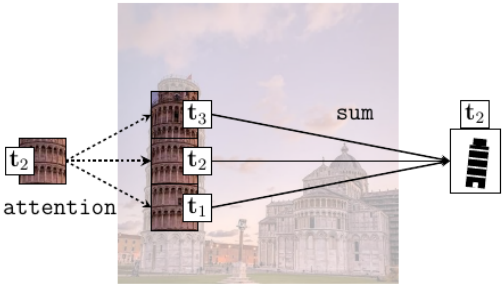


Figure: t_2 acts as query: its attention weights over t_1, t_2, t_3 (dotted arrows) set how much of each token's value is summed into the new t_2.

ViT: Self-Attention

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

$X \in \mathbb{R}^{(N+1)\times D}$: input token embeddings to the layer (N patches + `[CLS]`, with position embeddings)

$QK^\top \in \mathbb{R}^{(N+1)\times(N+1)}$ (softmax row-wise) cost $O(N^2 D)$

- **Why $\sqrt{d_k}$?** Assuming independent, zero-mean, unit-variance query and key components, $\text{Var}(q^\top k) = d_k$. Scaling by $\sqrt{d_k}$ keeps the logit variance approximately 1, helping prevent softmax saturation as d_k grows.
- **Multi-head attention:** h heads with $d_k = D/h$ have approximately the same leading-order attention cost as one head of dimension D: $O(N^2 D)$. Multiple heads allow different representation subspaces to be attended to in parallel.
- **Implementation**: Q, K, V all come from X, so one fused matmul $W^{QKV} \in \mathbb{R}^{D\times 3D}$ computes all three

ViT: Residual Blocks

- **Residuals** let each block learn a *correction* rather than a full transform – gradients flow directly through the $+$, avoiding vanishing gradients in 12+ stacked blocks (same reason as ResNet's skips)
- Two residual sub-blocks per layer: **MHSA** mixes information *across* tokens, **MLP** transforms *each* token independently – both wrapped the same way

$$x \leftarrow x + \mathrm{MHSA}(\mathrm{LN}(x)), \qquad x \leftarrow x + \mathrm{MLP}(\mathrm{LN}(x))$$

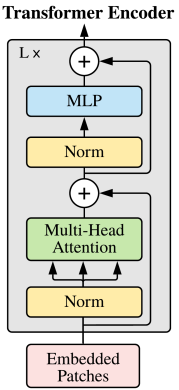


Figure: [*Dosovitskiy et al. 2021 – ViT, Fig. 1*]

ViT: Residual Blocks – Compute & Memory

- **Compute/memory:** $QK^\top$ is $N \times N \Rightarrow O(N^2 D)$ compute, $O(N^2)$ attention-matrix memory when materialized, $N = (H/P)(W/P)$ – halving patch size P **quadruples** N, and since cost $\propto N^2$, attention-score compute grows **16**$\times$ (why hierarchical encoders like SegFormer's MiT exist)
- **Params/block** (D=embed dim, MLP ratio 4): QKV $3D^2$ + output D^2 + MLP $8D^2 \approx 12D^2$, **dominated by the MLP**



e.g. ViT-B/16 (D=768): ≈7.1M params/block $\times$ 12 ≈85M parameters across 12 blocks

Normalization: BatchNorm vs. LayerNorm vs. GroupNorm

- All three normalize activations using the mean and variance over selected dimensions, then typically apply a learned affine scale and shift.
- They differ in **which dimensions are averaged over**:
- **BatchNorm** – batch and spatial dimensions, independently per channel.
- **LayerNorm** – feature dimensions, independently per sample (e.g., channels and spatial dimensions in this CNN-style diagram, or embedding dimensions per token in a ViT).
- **InstanceNorm** – spatial dimensions, independently per sample and channel.
- **GroupNorm** – groups of channels and spatial dimensions, independently per sample.

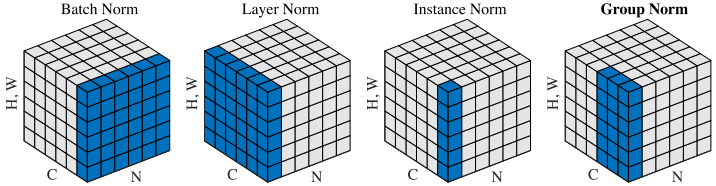


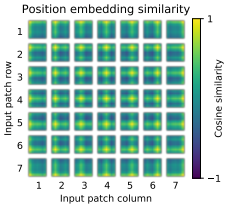
Figure: [*Wu & He 2018 – Group Normalization, Fig. 2*] – blue = dimensions averaged together (N=batch, C=channel, H, W=spatial)

Why Transformers Use LayerNorm

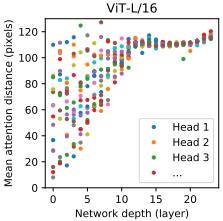
- **BatchNorm** uses batch statistics during training and running statistics during evaluation. Its behavior can depend on batch size, and training/evaluation statistics may differ.
- **LayerNorm** normalizes each token independently across its feature dimensions: batch-size independent, same computation at training and evaluation, and no running statistics.
- Self-attention operates on tokens within each input sequence, while sequence lengths can vary across inputs. LayerNorm avoids dependence on other samples or on batch-level statistics.
- **GroupNorm** normalizes groups of channels independently per sample, making it useful for CNNs trained with small batch sizes.

What ViT Learns

- Each tile shows the cosine similarity between the positional embedding at one patch location and those at all other locations.
- Patches in the same row or column often have more similar positional embeddings.
- This suggests that ViT learns to encode the image's 2D spatial structure through its positional embeddings.



- Each dot represents one attention head at one layer. Its height shows the mean attention-weighted spatial distance, in pixels: larger values indicate attention to more distant patches.
- Even in early layers, some heads attend locally while others already capture information from distant patches.



[*Dosovitskiy et al. 2021 – ViT, Figures 7 & 10*]

ViT Attention Rollout

- **Attention rollout**: recursively combines attention maps across layers to estimate how the final [CLS] representation connects to input patches.
- Produces a saliency-like visualization without pixel-level supervision, using attention patterns learned during classification training.
- Offers an intuitive view of spatial information flow in ViTs, although it is an approximation rather than a definitive explanation of the model's prediction

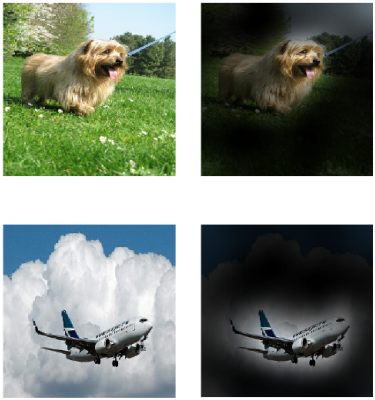


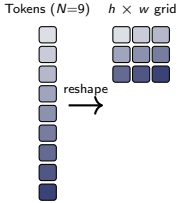
Figure: [*Dosovitskiy et al. 2021 – ViT, Figure 6*]

ViT: CNNs in Disguise

- **Tokenization**: splitting an image into non-overlapping $(P \times P)$ patches and linearly projecting each patch is equivalent to a convolution with kernel size (P), stride (P), and no padding.
- **QKV projections**: W^Q, W^K, W^V apply shared linear transformations independently to each token, equivalent to (1×1) convolutions on the spatial feature map.
- **MLP**: the two linear layers in a Transformer block also correspond to (1×1) convolutions, expanding the feature dimension (typically to $(4D)$) and then projecting back to (D).
- ViTs and CNNs both use linear transformations, normalization, and residual connections, but organize spatial information differently.
- **The one real difference**: conventional CNNs mix spatial information through fixed, local convolution kernels; standard ViTs use self-attention to compute input-dependent interactions between tokens, potentially across the entire image.

ViT Patch Tokens as a Feature Grid

- Drop the [CLS] token: the remaining ($N = h \times w$) patch tokens correspond to the spatial patch grid.
- Each token is a (D)-dimensional feature vector. Reshape the sequence into an ($h \times w \times D$) feature map, preserving the original patch ordering.
- This provides a coarse spatial feature representation that can serve as the encoder output for dense prediction.



Consequence: ViT features can feed encoder–decoder architectures for tasks such as semantic segmentation (e.g., SETR and DPT). Hierarchical models such as SegFormer provide multiscale features more like conventional CNN backbones.

ViT vs. CNN

	CNN	ViT
Receptive field	Local → global	Global from layer 1
Inductive bias	Strong locality bias	Weaker spatial bias
Data efficiency	Often strong with less data	Benefits from large-scale pretraining
Scaling w/ data & compute	Scales well	Scales very well in studied settings
Dense prediction	encoder-decoder	encoder-decoder
Multimodal fusion	Possible; architecture-dependent	Convenient token-based fusion

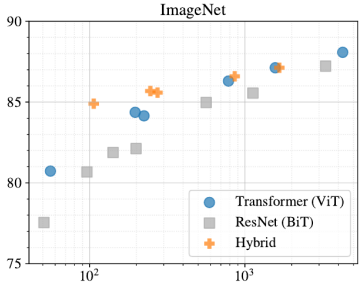


Figure: [*Dosovitskiy et al. 2021, Fig. 5*]

Takeaway: CNNs encode stronger spatial priors; ViTs enable flexible global interactions and often benefit from large-scale pretraining.

One Architecture, Many Tasks?

- **Mostly yes:** encoder-decoder + skip connections is a flexible baseline. The task head and loss can be changed while keeping the backbone
- **Not a free lunch:** some tasks need more than a new head.
- **Optical flow:** needs two frames and an explicit matching step.
- **Depth:** monocular depth has scale ambiguity; multi-scale context and suitable losses help, but absolute scale generally requires additional cues or assumptions.
- **Human pose:** accurate keypoint localization benefits from high-resolution features; some methods refine predictions iteratively (e.g., stacked hourglass), but this is not universal.

Takeaway: FCN/U-Net (CNN) and ViT-based encoders are strong starting points for dense prediction; state-of-the-art methods often add task-specific components.

Summary

- Dense prediction produces spatially organized outputs: segmentation, depth, optical flow, human pose
- Classification CNNs progressively downsample spatial features. Dense prediction often restores resolution using an **encoder-decoder** and **skip connections**.
- **U-Net:** a canonical encoder-decoder with skip connections, widely used for segmentation, especially in medical imaging.
- **ViT:** global self-attention over image patches, with weaker built-in spatial inductive bias than CNNs; patch tokens provide a spatial feature grid, often followed by a decoder or upsampling for dense outputs.
- Losses (CE, Dice, focal) guide training; task-specific metrics (mIoU, AbsRel, EPE, PCK) evaluate performance.
- Next lecture: advanced segmentation (DeepLab, SegFormer, ...) and object detection.

References

- Long, Shelhamer & Darrell (2015). Fully Convolutional Networks for Semantic Segmentation. CVPR 2015.
- Ronneberger, Fischer & Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI 2015.
- Dosovitskiy et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale (ViT). ICLR 2021.
- Devlin et al. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (origin of the `[CLS]` token). NAACL 2019.
- Zheng et al. (2021). Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers (SETR). CVPR 2021.
- Xie et al. (2021). SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. NeurIPS 2021.
- Ranftl et al. (2021). Vision Transformers for Dense Prediction (DPT). ICCV 2021.
- Milletari, Navab & Ahmadi (2016). V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation (Dice loss).
- Odena, Dumoulin & Olah (2016). Deconvolution and Checkerboard Artifacts. Distill.
- Shi et al. (2016). Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network.
- Cordts et al. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding. CVPR 2016.
- Ranftl, Lasinger, Hafner, Schindler & Koltun (2022). Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer (MiDaS). TPAMI 2022.
- Butler, Wulff, Stanley & Black (2012). A Naturalistic Open Source Movie for Optical Flow Evaluation (MPI-Sintel). ECCV 2012.
- Baker, Scharstein, Lewis, Roth, Black & Szeliski (2011). A Database and Evaluation Methodology for Optical Flow. IJCV 2011.
- Cao, Simon, Wei & Sheikh (2017). Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields (OpenPose). CVPR 2017.
- Badrinarayanan, Kendall & Cipolla (2015). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.
- Teed & Deng (2020). RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. ECCV 2020.
- Zhao, Shi, Qi, Wang & Jia (2017). Pyramid Scene Parsing Network (PSPNet). CVPR 2017.
- Eigen, Puhrsch & Fergus (2014). Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. NeurIPS 2014.
- Liu et al. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. ICCV 2021.
- Su et al. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. arXiv 2021.
- Vaswani et al. (2017). Attention Is All You Need. NeurIPS 2017.
- Wu & He (2018). Group Normalization. ECCV 2018.



1



2



3



4



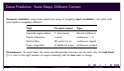
5



6



7



8



9



10



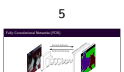
11



12



13



14



15



16



17



18



19



20



21



22



23



24



25



26



27



28



29



30



31



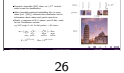
32



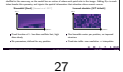
33



34



35



36



37



38



39



40

41

42