Image from NASA:
https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

Geospatial Deep Learning

Transforms, ConvNeXt, and Mamba

WV VIEW

AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

This module briefly presents three modern architectures in deep learning: Transformer-based models for vision tasks, a modernization of CNNs (ConvNeXt), and Mamba, a state-space sequence model. The module is intended as an introductory survey rather than an exhaustive deep dive.

Module Topics

1. Transformers and multi-head self-attention
2. Tokenization and embeddings
3. Positional embeddings
4. Value, query, and key
5. Vision transformers (ViTs)
6. Patch embeddings
7. Swin transformer
8. Patch merging
9. Patch expansion
10. SegFormer
11. Swin-UNet
12. ConvNeXt
13. Mamba
14. VMamba
15. Mamba-UNet
16. RS^{3}Mamba

2

These are the topics covered in this module.

Transformers

The following slides introduce the Transformer architecture, originally proposed for natural language processing and later adapted for vision tasks. The key innovation is the self-attention mechanism, which allows the model to dynamically focus on different parts of an input sequence.

Concept of attention

- Learning what to focus on

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* † illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.

This slide introduces the concept of attention. The core idea is that rather than treating all inputs equally, the model learns to focus on the most relevant parts of the input for a given task. This is analogous to how humans direct visual or cognitive attention selectively.

Why transformers?

- Capture more spatial context
- CNNs generally focus on more local context, struggle with more global context

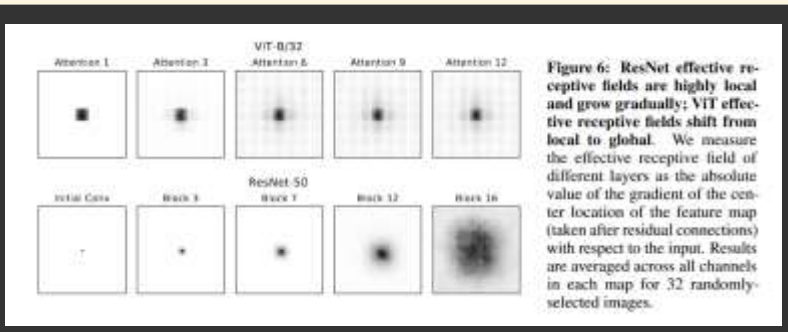
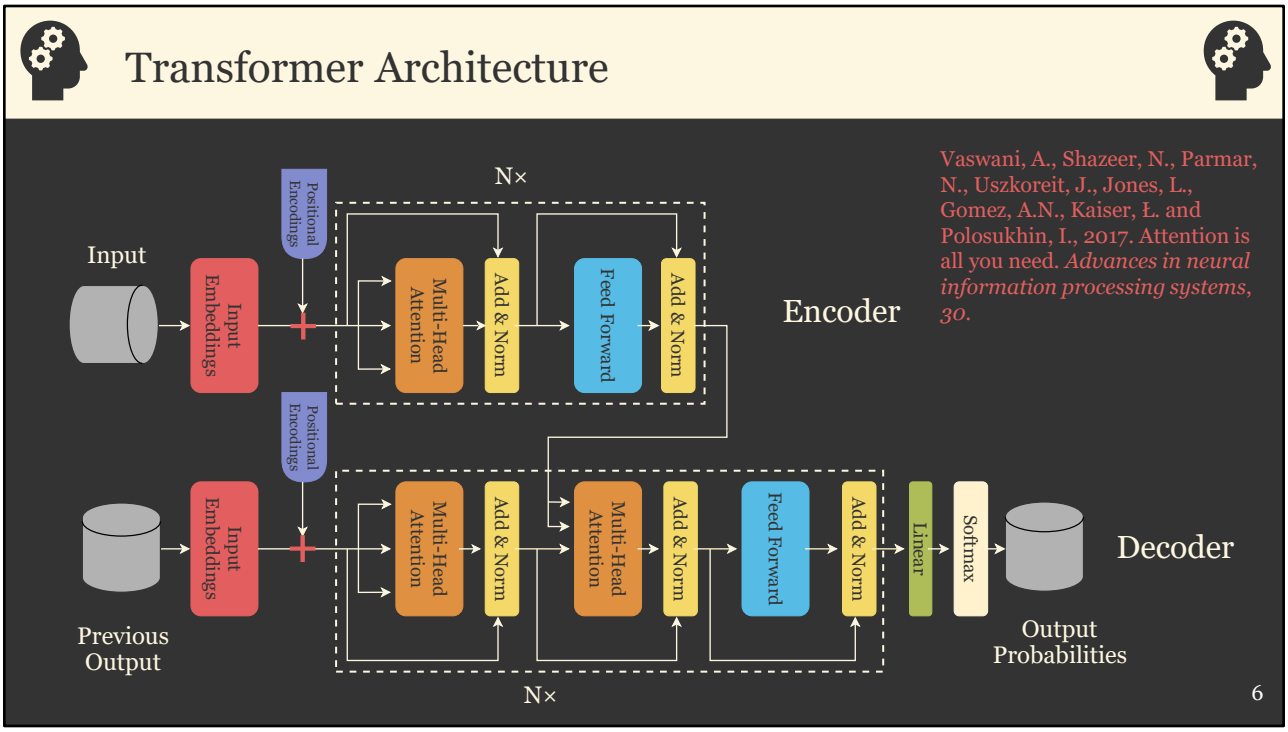


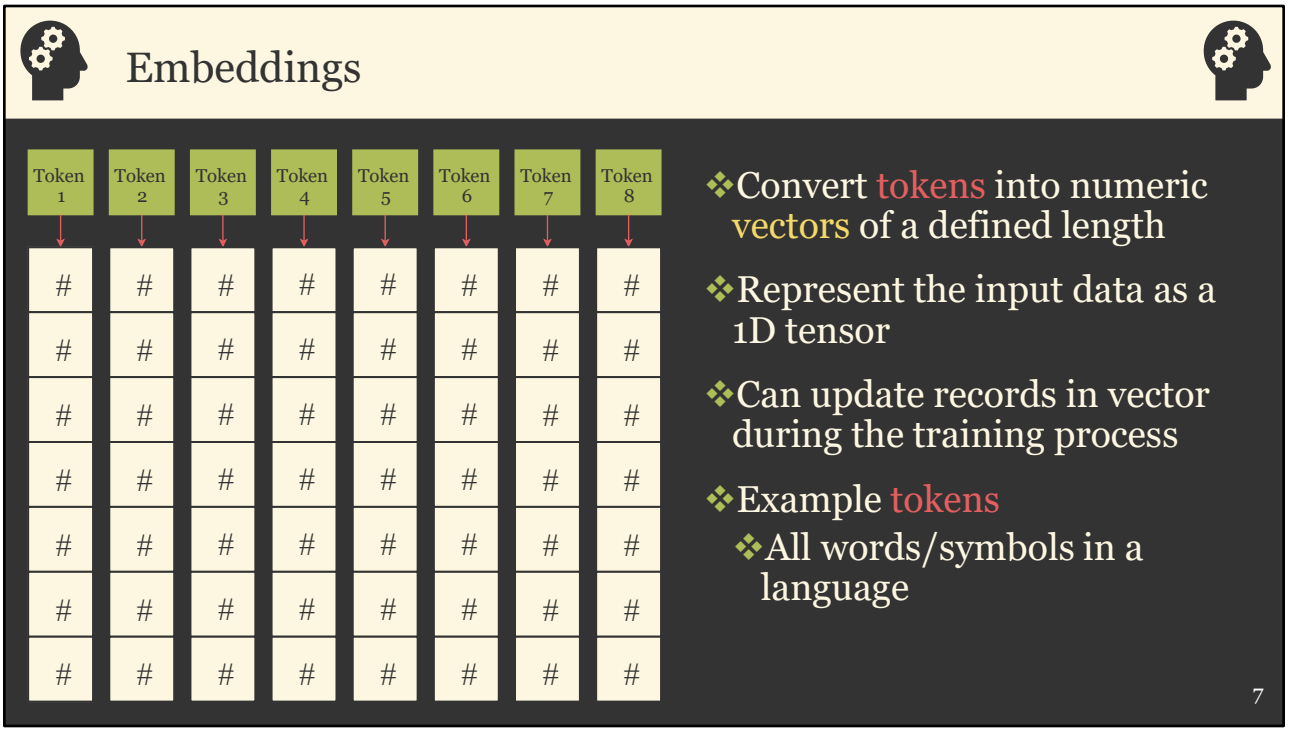
Figure 6: ResNet effective receptive fields are highly local and grow gradually; ViT effective receptive fields shift from local to global. We measure the effective receptive field of different layers as the absolute value of the gradient of the center location of the feature map (taken after residual connections) with respect to the input. Results are averaged across all channels in each map for 32 randomly-selected images.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C. and Dosovitskiy, A., 2021. Do vision transformers see like convolutional neural networks?. *Advances in neural information processing systems*, *34*, pp.12116-12128.

This slide motivates the use of Transformers for vision tasks. CNNs are inherently local in their processing due to fixed-size convolutional kernels, which makes capturing long-range dependencies difficult. Research has shown that Vision Transformers develop more global representations early in the network compared to CNNs, making them well suited for tasks requiring broad spatial context such as scene understanding and segmentation.



This diagram shows the full Transformer architecture from the original "Attention is All You Need" paper. It consists of an Encoder stack (top) and a Decoder stack (bottom). The encoder processes the input sequence using multi-head self-attention and feed-forward layers with residual connections and normalization. The decoder additionally attends to the encoder output using cross-attention. Positional encodings are added to input embeddings to provide sequence order information.



Embeddings convert discrete input tokens into continuous numeric vectors. Each token (e.g., a word or image patch) is mapped to a high-dimensional vector that can be learned during training. These vectors serve as the foundational representation fed into the Transformer. The embedding table allows the model to associate semantic meaning with each token through training. In our example, each token is represented by an embedding vector with a length of 7; however, a real model would use a much longer token length in order to capture more semantic information.

Positional Embeddings

- Embeddings do not natively store information about position in the sequence

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

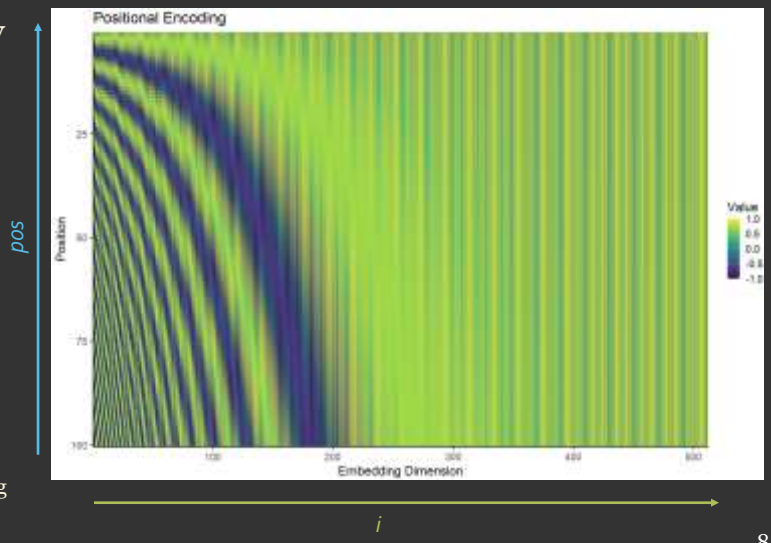
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d}}}\right)$$

pos = index of token

d = length of encoding vector

i = index of vector in position embedding

- sine method applied to odd vector embedding indices; cosine applied to even vector embeddings



Because Transformers process all tokens simultaneously (unlike recurrent neural networks (RNNs), which process them sequentially), they have no inherent notion of order. Positional encodings are added to the embedding vectors to inject order information. The sinusoidal encoding formula uses sine and cosine functions at different frequencies for each position-dimension pair, allowing the model to generalize to sequence lengths not seen during training.

Value, Query, and Key

- Query = request or search
- Key = what is returned from a query
- Value = data associated with key

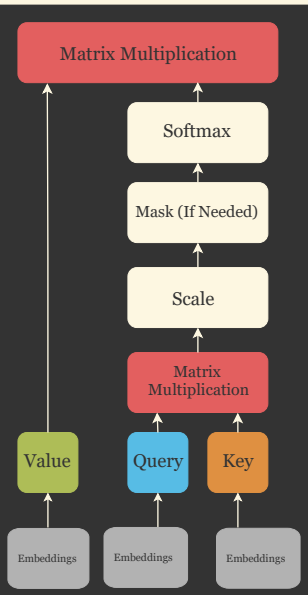
Example

Query = Which car should I buy?

Key = list of cars to consider

Value = characteristics of each car

Query, key, and value represented as vectors/ matrices using linear or fully connected layer



This slide explains the Query, Key, and Value abstraction used in attention. The Query represents what you are searching for; the Key represents what each element offers in response to a query; and the Value is the actual information returned. A dot product between the Query and each Key produces attention scores, which after scaling and softmax are used to weight the Values. This allows each token to attend to the most relevant other tokens.

Cosine Similarity

- If two vectors point in the same direction, cosine similarity = 0
- Most dissimilar, cosine similarity = -1 or 1

$$\text{Cosine Similarity} = \cos(\theta) = \frac{A \cdot B}{|A||B|}$$

$$\text{Similarity (Q, K)} = \frac{Q \cdot K^T}{\text{Scaling}}$$

$$Q \cdot K^T = \text{Attention Filter}$$

	Token 1	Token 2	Token 3	Token 4	Token 5	Token 6	Token 7	Token 8
Token 1	#	#	#	#	#	#	#	#
Token 2	#	#	#	#	#	#	#	#
Token 3	#	#	#	#	#	#	#	#
Token 4	#	#	#	#	#	#	#	#
Token 5	#	#	#	#	#	#	#	#
Token 6	#	#	#	#	#	#	#	#
Token 7	#	#	#	#	#	#	#	#
Token 8	#	#	#	#	#	#	#	#

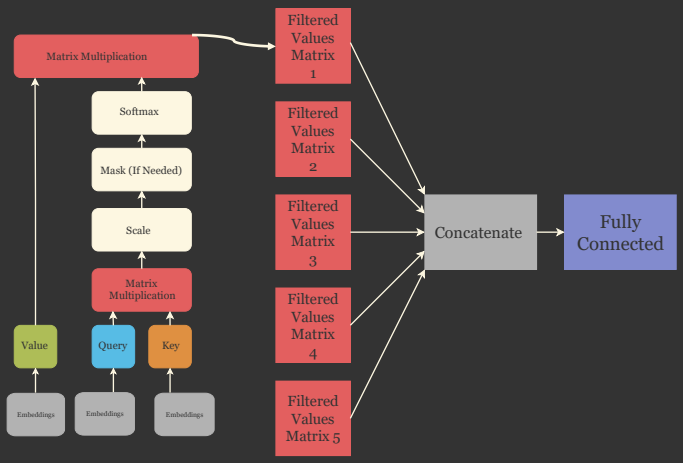
Cosine similarity measures how aligned two vectors are, regardless of their magnitude. In the attention mechanism, the dot product between the Query and Key vectors approximates their similarity. When two token embeddings point in similar directions, they have high cosine similarity and attend strongly to each other. This is how the model captures semantic relationships between tokens.

Self Attention

- Relationship between tokens in same sequence or input
- Use multiple heads that focus on different information

$$\text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

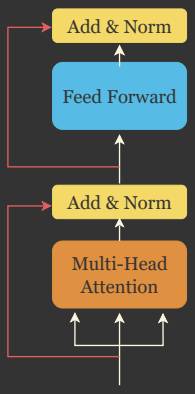
Q = query matrix
K = key matrix
V = values matrix
d_k = dimensionality of key matrix



Self-attention computes relationships between all tokens within the same input sequence. The multi-head variant runs multiple attention operations in parallel, each with different learned projections. This allows the model to attend to different types of relationships simultaneously. The outputs from all heads are concatenated and projected through a fully connected layer.

Residual Connections

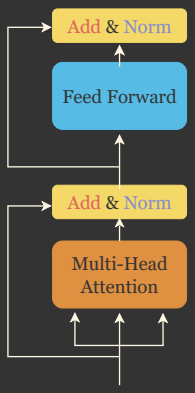
- Knowledge preservation
- Vanishing gradient problem



Residual connections (also called skip connections) pass the original input around a sub-layer and add it to the output. This serves two purposes: it helps preserve information that might otherwise be lost (knowledge preservation), and it helps gradients flow more easily during backpropagation, mitigating the vanishing gradient problem. They are used around both the multi-head attention and feed-forward layers.

Add & Norm

- Add = add data from residual connection and output from attention layer
- Normalization = normalize to *z*-score along embedding dimension (different from batch normalization)



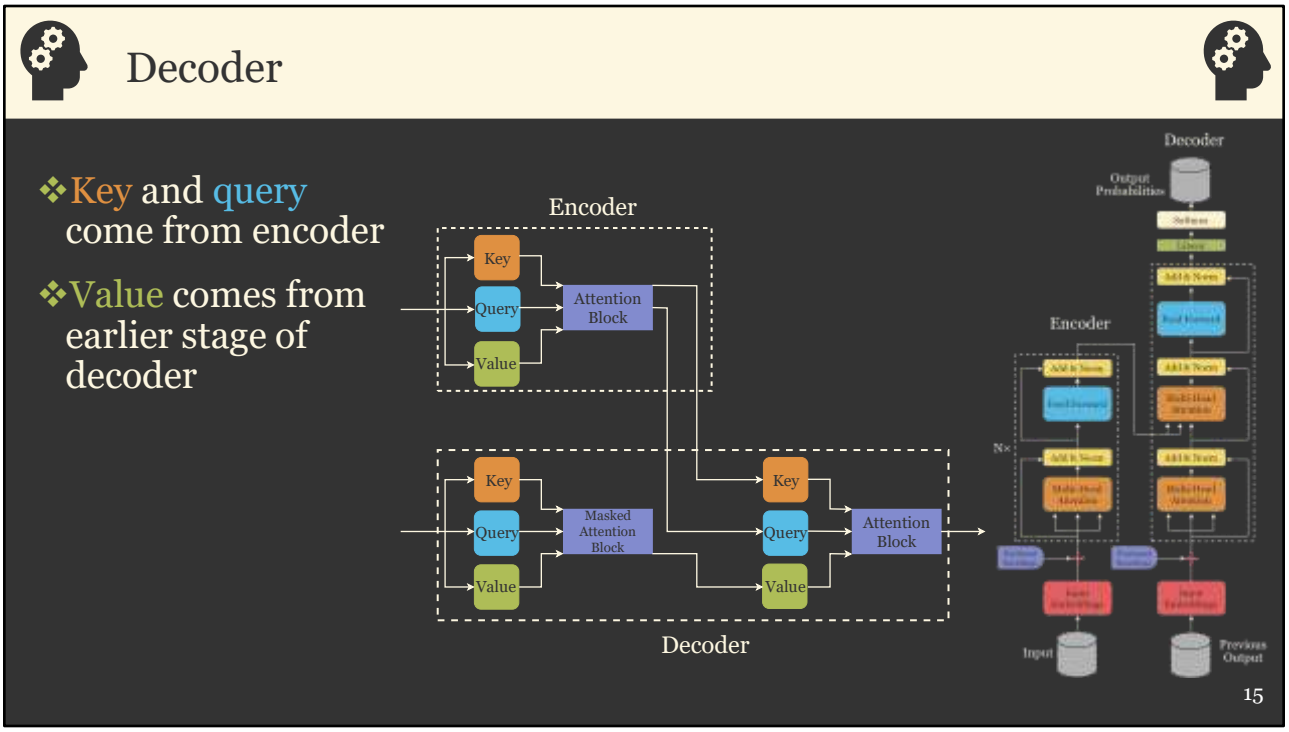
The Add & Norm operation combines a residual connection (Add) with Layer Normalization (Norm). The residual adds the sub-layer's input to its output, and the Layer Normalization standardizes the result along the embedding dimension. Importantly, this is Layer Normalization, as opposed to not Batch Normalization, because normalization is performed per sample rather than per batch.

Masking

	Token 1	Token 2	Token 3	Token 4	Token 5	Token 6	Token 7	Token 8
Token 1	#	#	#	#	#	#	#	#
Token 2	#	#	#	#	#	#	#	#
Token 3	#	#	#	#	#	#	#	#
Token 4	#	#	#	#	#	#	#	#
Token 5	#	#	#	#	#	#	#	#
Token 6	#	#	#	#	#	#	#	#
Token 7	#	#	#	#	#	#	#	#
Token 8	#	#	#	#	#	#	#	#

- Cannot allow the decoder information about tokens later in the sequence
- Mask out later tokens
- # → -∞
- softmax(-∞) = 0

Masking is used in the Decoder to prevent the model from attending to future tokens in the output sequence during training. Since the decoder generates tokens one-at-a-time during inference, it would be invalid for it to use future ground-truth tokens. Future positions are masked by setting their attention scores to negative infinity before the softmax, making their contribution effectively zero.



The Decoder uses two types of attention. First, it uses masked self-attention on the decoder's own partial output to prevent attending to future tokens. Second, it uses cross-attention: the Query comes from the decoder's current state, while the Key and Value come from the Encoder's output. This allows the decoder to selectively attend to relevant parts of the encoded input when generating each output token.

Adaption for Imagery Data

Published as a conference paper at ICLR 2021

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

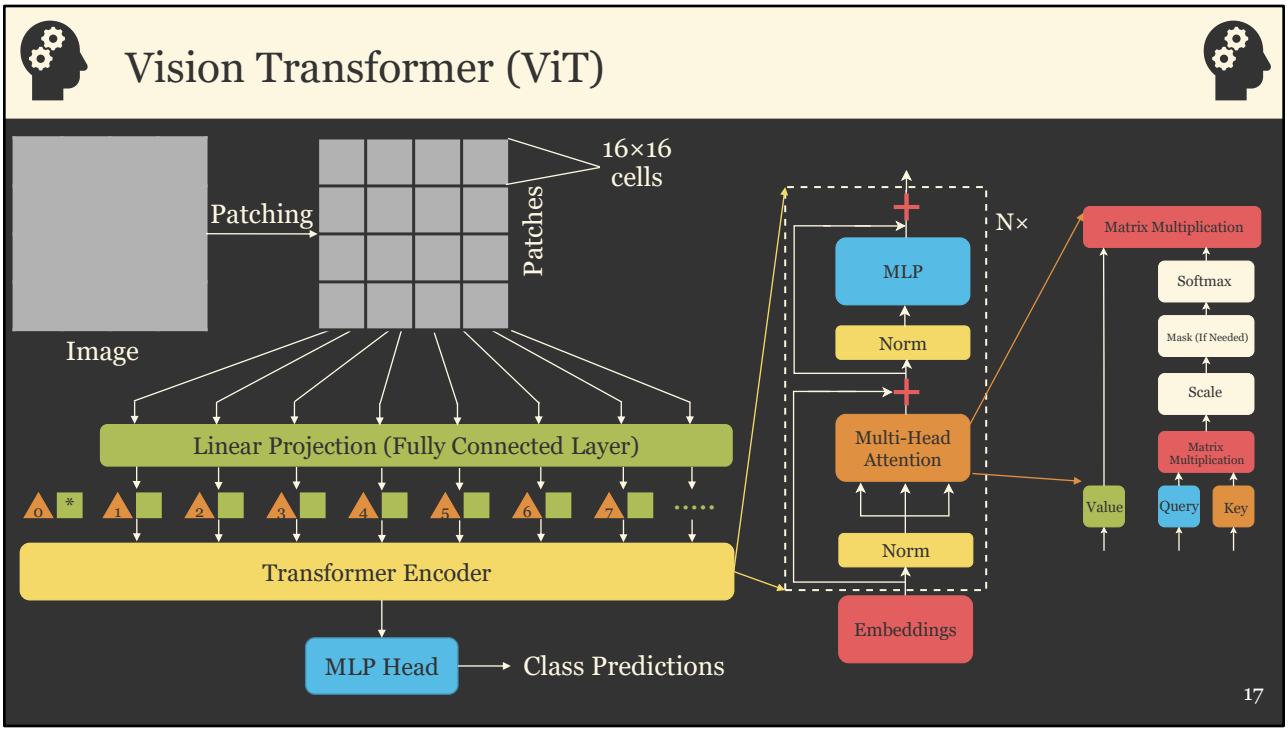
ABSTRACT

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.

- Treat image patches as tokens
- Create embeddings vector from image patches
- Add position information
- Use encoder component of transformer architecture

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S. and Uszkoreit, J., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

This slide introduces the Vision Transformer (ViT), which adapts the Transformer encoder for image classification. Images are divided into fixed-size, non-overlapping patches, which are then linearly projected into embedding vectors and treated as tokens. A learnable class token is prepended and positional embeddings are added. The resulting sequence is passed through the standard Transformer encoder.



The full ViT architecture is shown here. Image patches are projected via a linear layer into patch embeddings, positional encodings are added, and the sequence is passed through N stacked Transformer encoder blocks. Each block consists of Layer Norm, Multi-Head Self-Attention, another Layer Norm, and an MLP with residual connections. The class token's final representation is fed into an MLP classification head.

The classification token at position 0 is a learnable vector that gets prepended to the sequence of patch embeddings before they enter the Transformer encoder. It doesn't correspond to any actual patch of the image; it's just an extra "slot" in the sequence that the model can write information into. The Transformer encoder processes a sequence of tokens and produces an equally-sized sequence of output tokens. For image classification, you need to collapse that whole sequence down to a single vector that represents the entire image. You have a few options for how to do that:

1. Average all the patch output tokens (global average pooling)
2. Use a dedicated token whose sole job is to accumulate a global summary

ViT uses option 2. The idea is to give the model a dedicated "scratch pad" token with no positional or content bias, whose final output representation is handed to the classification head. Because self-attention is fully connected across all

tokens, the classification token can attend to every patch in every layer. Over the course of the N encoder blocks, it progressively aggregates information from across the image. By the final layer, its output embedding is treated as a summary of the whole image and fed into the MLP classification head to produce class probabilities.

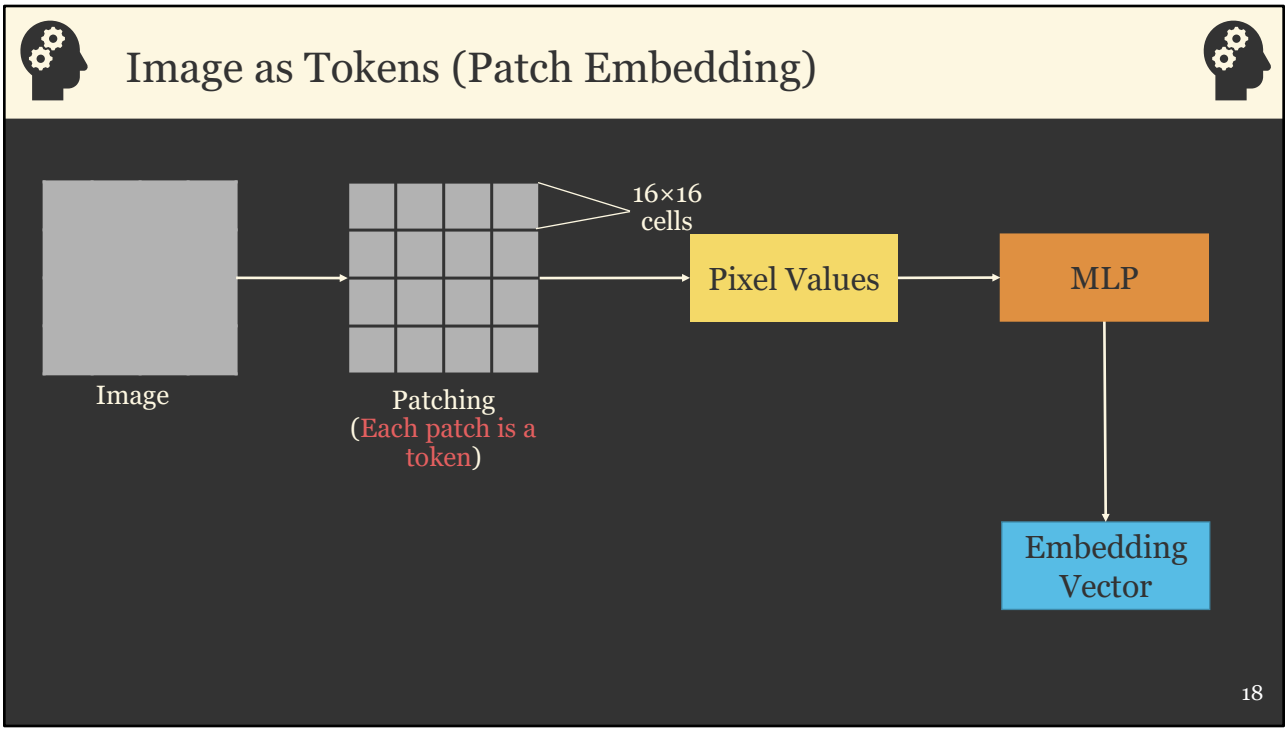
Here is a further elaboration on positional embeddings as implemented in ViTs. They solve a fundamental problem: self-attention is permutation-invariant, meaning if you shuffled all the patch tokens into a random order, the attention mechanism would produce the same output embeddings (just in a different order). The model would have no way of knowing that a patch came from the top-left vs. the bottom-right of the image. Positional embeddings inject that spatial information. A positional embedding is simply a learnable vector of the same dimensionality as the patch embeddings. One is assigned to each position in the sequence (i.e., each patch slot, plus the classification token). These vectors are added element-wise to the corresponding patch embeddings before the sequence enters the first Transformer block.

Unlike the original Transformer paper (which used fixed sinusoidal encodings), ViT uses fully learnable positional embeddings; they're just parameters in the model that get updated via backpropagation like any other weight. The model learns what spatial signal is most useful for the task at hand rather than having it hard-coded.

When you visualize the learned positional embeddings of a trained ViT (by computing cosine similarity between each position's embedding and every other position's embedding), a clear spatial structure emerges. Nearby patches end up with similar positional embeddings, and the embeddings reflect row/column structure — patches in the same row are more similar to each other than to patches in a different row. The model essentially rediscovers 2D spatial layout from a 1D sequence of learned vectors, which is a bit remarkable given that no 2D structure was explicitly encoded.

The standard ViT uses 1D positional embeddings: each patch position in the flattened sequence gets a single embedding, with no explicit encoding of row and column separately. An alternative is 2D positional embeddings, where you learn separate row and column embeddings and combine them (e.g., by concatenation or addition). Interestingly, the original ViT paper found that 1D learned embeddings performed about as well as 2D variants, suggesting the model can figure out the 2D structure on its own from the 1D embeddings given enough data.

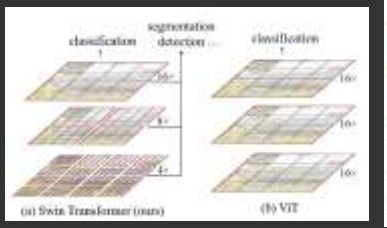
One weakness of learned positional embeddings is that they don't generalize well to image sizes not seen during training. If a ViT is trained on 224×224 images with 16×16 patches, it learns embeddings for a fixed 14×14 grid of positions. If you then want to run inference on a 384×384 image, you'd have a 24×24 grid — positions that never existed during training. The standard fix is to bicubically interpolate the learned positional embeddings to the new grid size, which works reasonably well in practice but is an inherent limitation compared to the sinusoidal encodings used in the original Transformer, which can extrapolate to arbitrary lengths.

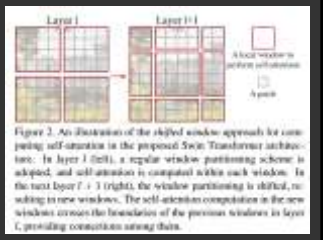


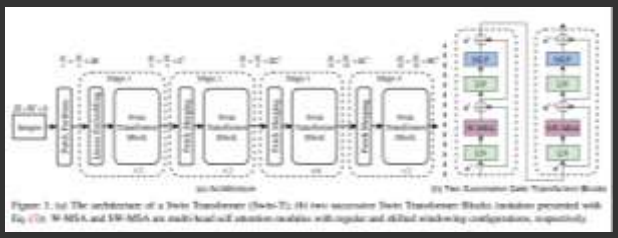
This slide details the patching and embedding process in ViT. Each 16x16 pixel patch is treated as a token. The raw pixel values within each patch are flattened and projected through an MLP (linear projection) to produce a fixed-length embedding vector. This creates a 1D sequence of token embeddings from the 2D image input, enabling the Transformer architecture to process it.

Swin Transformer

- Shifted Window
- Hierarchical feature extraction
- Patch image at multiple scales
- Linear as opposed to quadratic complexity (ViT)
- Shifted/overlapping windows
- Patch Merging layer
- Shifted window multi-head self attention





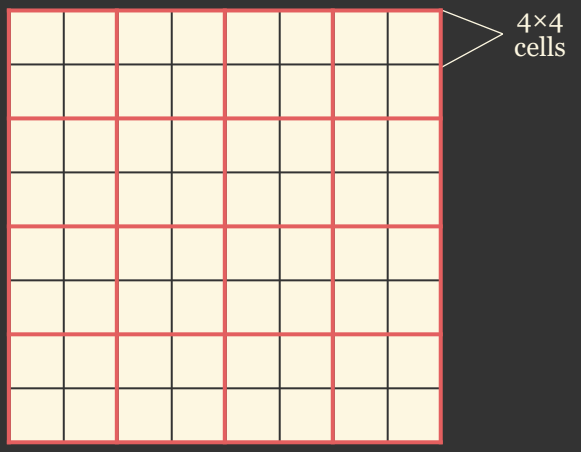


Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012-10022).

The Swin Transformer improves on ViT by using hierarchical feature extraction (like CNNs) and limiting self-attention to local windows rather than globally. Windows shift between layers to allow cross-window communication. Patch merging layers reduce spatial resolution while increasing channel depth, creating a multi-scale feature hierarchy. This makes Swin Transformers more efficient and better suited for dense prediction tasks.

W-MSA

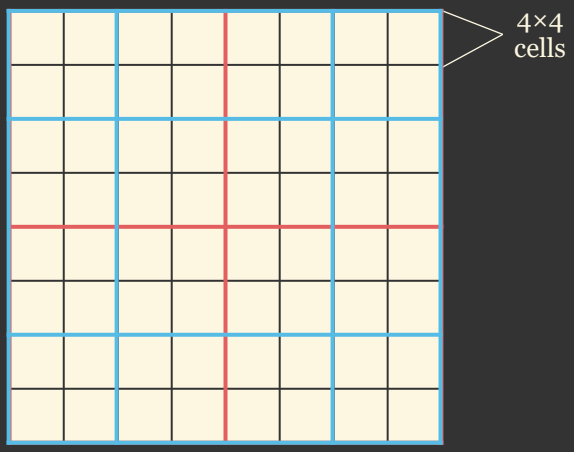
- Window-Based Multi-Head Self-Attention
  - Image split into non-overlapping windows
  - Self attention applied independently within each window
  - Computationally efficient
  - Scale linearly with image size
  - No communication between windows



Window-based Multi-Head Self-Attention (W-MSA) restricts self-attention to non-overlapping local windows. This reduces computational complexity from quadratic (in image size) to linear, since attention is computed independently within each window. The trade-off is that tokens in different windows cannot directly attend to each other, which is addressed by the shifted window mechanism.

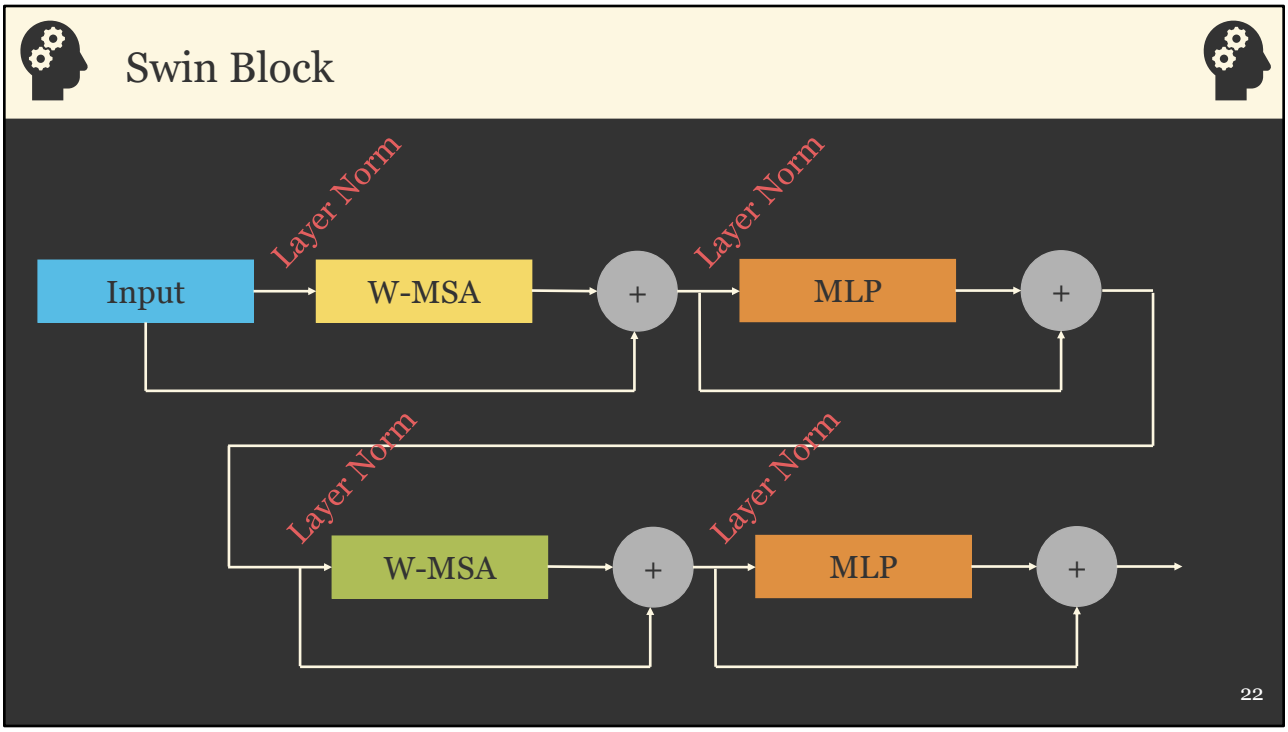
SW-MSA

- Shifted Window Multi-Head Self-Attention
  - Shifts the window partition (typically by half the window size)
  - Apply windowed attention again
  - Allows communication between windows



21

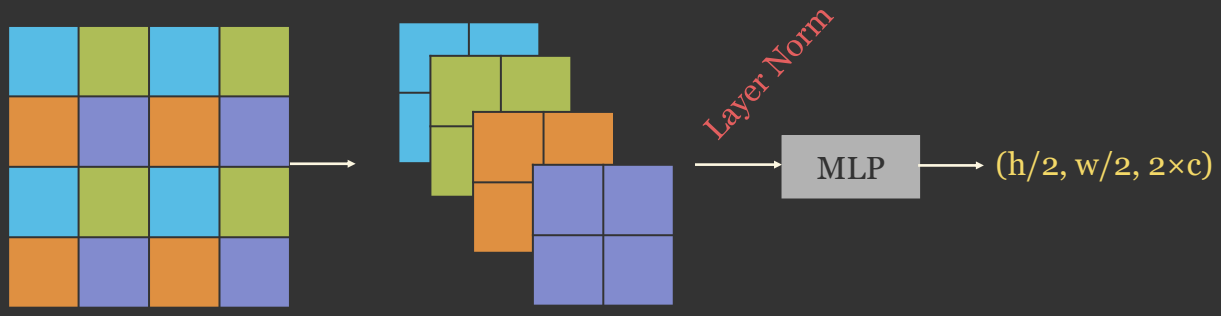
Shifted Window Multi-Head Self-Attention (SW-MSA) shifts the window partition by half the window size between consecutive Swin blocks. This causes previously separate windows to partially overlap in the next layer, enabling cross-window communication. Together, W-MSA and SW-MSA alternate in Swin blocks to efficiently capture both local and global context.



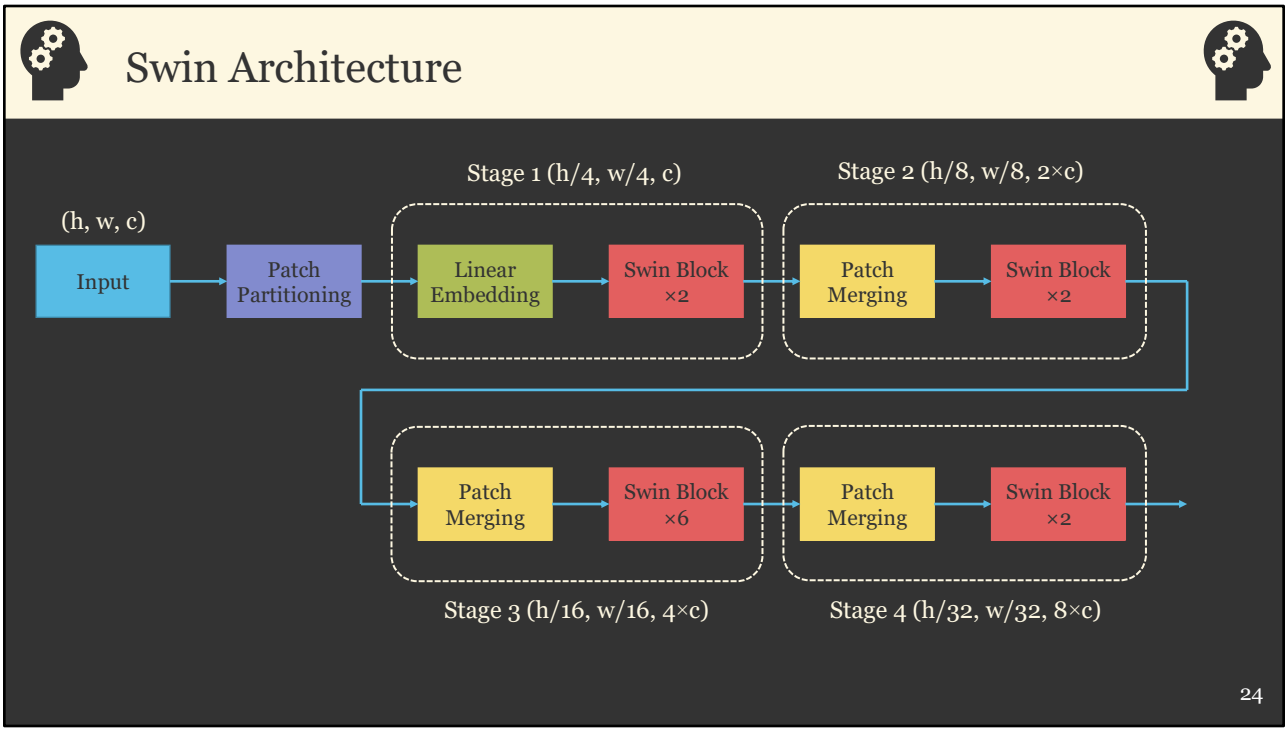
The Swin Block pairs two sub-blocks: one with W-MSA and one with SW-MSA. Each sub-block applies Layer Normalization before attention, followed by a residual connection. Similarly, each sub-block's attention is followed by a Layer Norm and MLP with another residual connection. This alternating structure ensures both local and shifted-window attention are applied at each stage.

Patch Merging

- Need a means to downsample (like max pooling or strided convolution)
- Merge the patch embeddings for a 2x2 window of tokens to a new, longer embedding
- (h, w, c) → (h/2, w/2, 2×C)



Patch Merging in Swin Transformer serves as a downsampling operation, analogous to max pooling in CNNs. It takes a 2x2 neighborhood of patches, concatenates their embeddings, and projects them to half the concatenated size. This reduces spatial dimensions by 2x in each direction while increasing the channel depth, producing a hierarchical feature map.



The full Swin Transformer architecture processes the image in four stages. It begins with Patch Partitioning and Linear Embedding, then alternates Swin Blocks with Patch Merging layers to progressively reduce spatial resolution and increase channel depth. The feature maps progress from (h/4, w/4) at Stage 1 to (h/32, w/32) at Stage 4, building a rich multi-scale representation.

ViTs for Semantic Segmentation

Architecture	Year	Citation
Segmentation Transformer (SETR)	2021	Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H. and Zhang, L., 2021. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 6881-6890).
Swin Transformer	2021	Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 10012-10022).
Segmenter	2021	Strudel, R., Garcia, R., Laptev, I. and Schmid, C., 2021. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 7262-7272).
SegFormer	2021	Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M. and Luo, P., 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34, pp.12077-12090.
Pyramid Vision Transformer (PVT) (v1/v2)	2021	Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P. and Shao, L., 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 568-578).
Twins	2021	Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H. and Shen, C., 2021. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, 34, pp.9355-9366.
Dense Prediction Transformer (DPT)	2021	Ranftl, R., Bochkovskiy, A. and Koltun, V., 2021. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 12179-12188).
High Resolution Transformer (HRFormer)	2021	Yuan, Y., Fu, R., Huang, L., Lin, W., Zhang, C., Chen, X. and Wang, J., 2021. Hrformer: High-resolution vision transformer for dense predict. *Advances in neural information processing systems*, 34, pp.7281-7293.
Mask2Former	2022	Cheng, B., Misra, I., Schwing, A.G., Kirillov, A. and Girdhar, R., 2022. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1290-1299).

Thisanke, H., Deshan, C., Chamith, K., Seneviratne, S., Vidanaarachchi, R. and Herath, D., 2023. Semantic segmentation using Vision Transformers: A survey. *Engineering Applications of Artificial Intelligence*, *126*, p.106669.

This slide provides a survey of Vision Transformer architectures adapted for semantic segmentation. Many architectures emerged in 2021 that leverage the global attention capabilities of ViTs or hierarchical designs like Swin for dense pixel-level prediction. Key examples include SETR, SegFormer, Swin Transformer, and Mask2Former, each with different strategies for encoding and decoding spatial information.

SegFormer

- Encoder = hierarchical transformer architecture
- Multilayer Perceptron = predict classification of pixel using multiscale features

Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M. and Luo, P., 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34, pp.12077-12090.

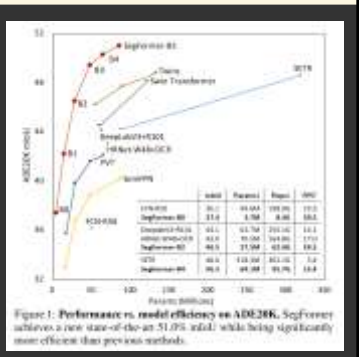


Figure 1: **Performance vs. model efficiency on ADE20K.** SegFormer achieves a new state-of-the-art 51.0% mIoU while being significantly more efficient than previous methods.

SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers

Enze Xie[1] Wenhai Wang[2] Zhiding Yu[3] Anima Anandkumar[3,4] Jose M. Alvarez[3] Ping Luo[1]
[1]The University of Hong Kong [2]Nanjing University [3]NVIDIA [4]Caltech

Abstract

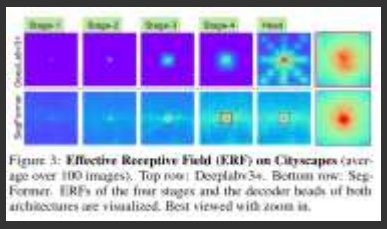


Figure 3: **Effective Receptive Field (ERF) on Cityscapes** (average over 100 images). Top row: Deeplabv3+. Bottom row: SegFormer. ERFs of the four stages and the decoder heads of both architectures are visualized. Best viewed with zoom in.

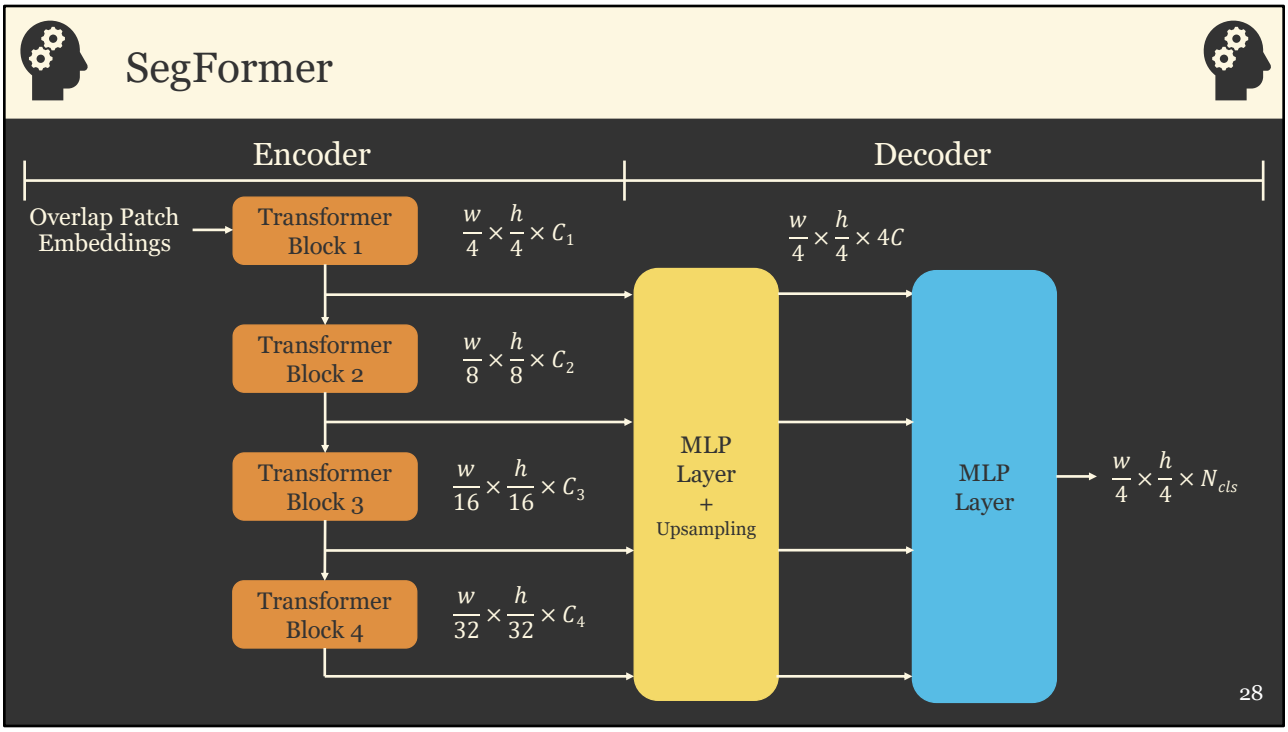
SegFormer is a semantic segmentation framework built on a hierarchical transformer encoder (Mix Transformer, MiT) and a lightweight MLP decoder. The encoder produces multi-scale features at four different resolutions, and the decoder aggregates and upsamples these features to produce pixel-level class predictions. The design prioritizes simplicity and efficiency without sacrificing performance.

SegFormer

- Different Mix Transformer Encoders = MiT-B0 to MiT-B5
- Hierarchical Feature Representation = capture coarse and fine patterns in input imagery
- Overlapped Patch Merging = coarsen data to generate hierarchical feature maps; merge patches with overlap
  - K = patch size
  - S = stride between adjacent patches
  - P = padding size
- Efficient Self-Attention = increase computational efficiency
- Mix-FFN = provide positional information
- MLP = perform pixel-level classification

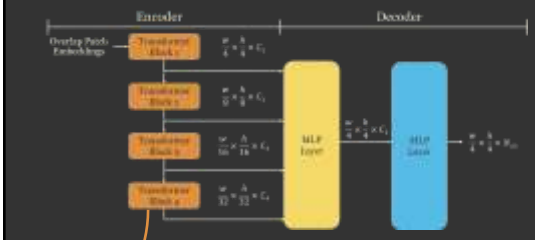
Xie, E., Wang, W., Yu, Z., Anandkumar, A., Alvarez, J.M. and Luo, P., 2021. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Advances in neural information processing systems*, 34, pp.12077-12090.

This slide details the SegFormer encoder components. The Mix Transformer (MiT) comes in six variants (B0 to B5) with increasing capacity. Hierarchical Feature Representation enables multi-scale understanding. Overlapped Patch Merging preserves local continuity between patches during downsampling. Efficient Self-Attention reduces attention cost by spatially reducing the key and value sequences. Mix-FFN injects positional information via a 3x3 depth-wise convolution within the feed-forward block.

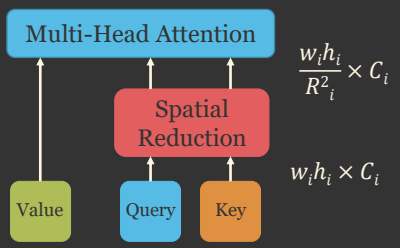


The SegFormer architecture diagram shows four Transformer encoder stages that extract features at progressively smaller spatial resolutions. Each stage outputs a feature map that is passed to the MLP decoder. The decoder upsamples and concatenates all four feature maps before a final MLP produces the segmentation output, enabling predictions that leverage both fine-grained and coarse semantic information.

SegFormer



Efficient Self Attention

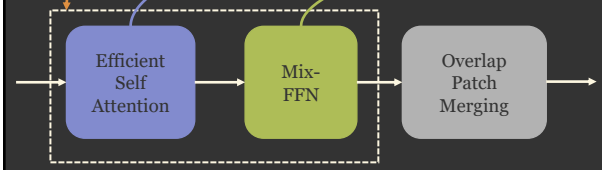


Mix-FFN

$$Out = MLP(GELU(Conv_{3\times3}(MLP(x_{in})))) + x_{in}$$

$$GELU \approx 0.5x\left[1 + tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right)\right] \text{ or } GELU \approx x\sigma(1.792x)$$

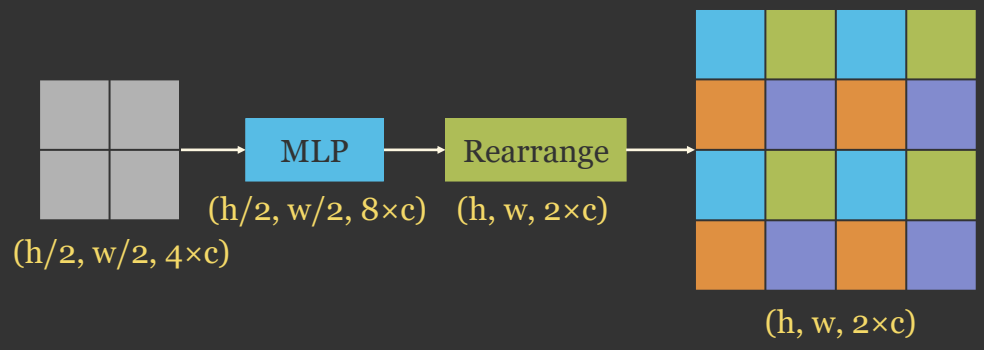
Hendrycks, D. and Gimpel, K., 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.



This slide details the key components of the SegFormer Transformer block. Efficient Self-Attention applies spatial reduction to the Key and Value sequences to reduce quadratic complexity. Mix-FFN replaces standard positional encoding with a 3x3 depth-wise convolution inside the feed-forward network, enabling position-aware feature mixing. The GELU activation is used throughout.

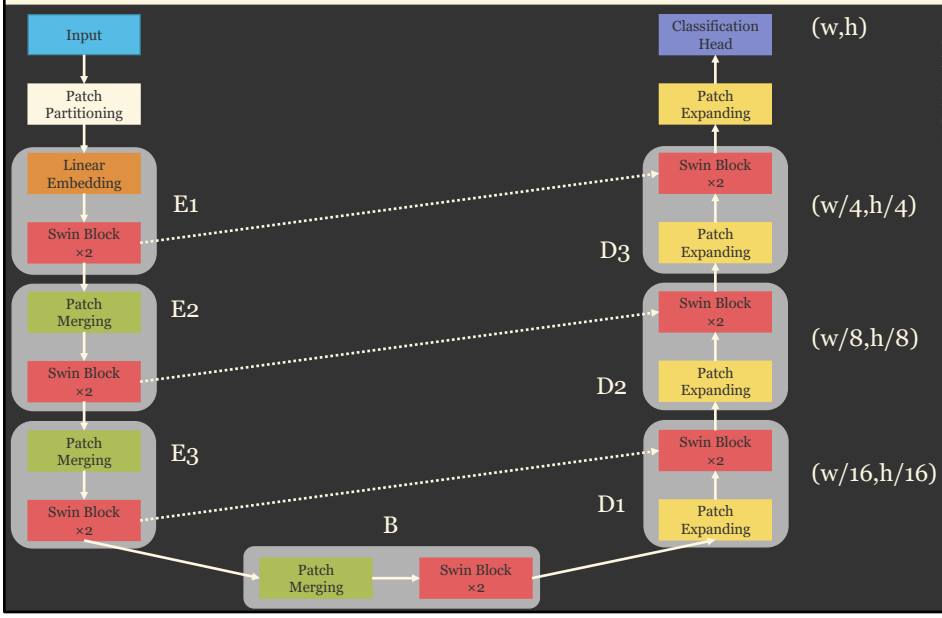
Patch Expansion

- Reshape to increase spatial resolution



Patch Expansion is the inverse of Patch Merging and is used in decoder paths (e.g., Swin-UNet). It upsamples feature maps by rearranging tokens from the channel dimension back into spatial dimensions. For example, a (h/2, w/2, 8C) tensor is rearranged and projected to produce a (h, w, 2C) tensor, effectively doubling spatial resolution while reducing channel depth.

Swin-UNet



Cao, H., Wang, Y., Chen, J., Jiang, D., Zhang, X., Tian, Q. and Wang, M., 2022, October. Swin-unet: Unet-like pure transformer for medical image segmentation. In *European conference on computer vision* (pp. 205-218). Cham: Springer Nature Switzerland.

Swin-UNet applies the UNet encoder-decoder architecture using pure Swin Transformer blocks, replacing all convolutions. The encoder downsamples via Patch Merging and Swin Blocks, producing hierarchical features (E1, E2, E3) and a bottleneck (B). The decoder symmetrically upsamples using Patch Expanding and Swin Blocks (D1, D2, D3), with skip connections from encoder stages. It was originally proposed for medical image segmentation.

ConvNeXt

This section introduces ConvNeXt, a modernized convolutional neural network architecture. ConvNeXt was developed by systematically incorporating design decisions from Vision Transformers (ViTs) back into a ResNet baseline, demonstrating that CNNs can match or exceed Transformer performance when updated with modern training recipes and architectural refinements.

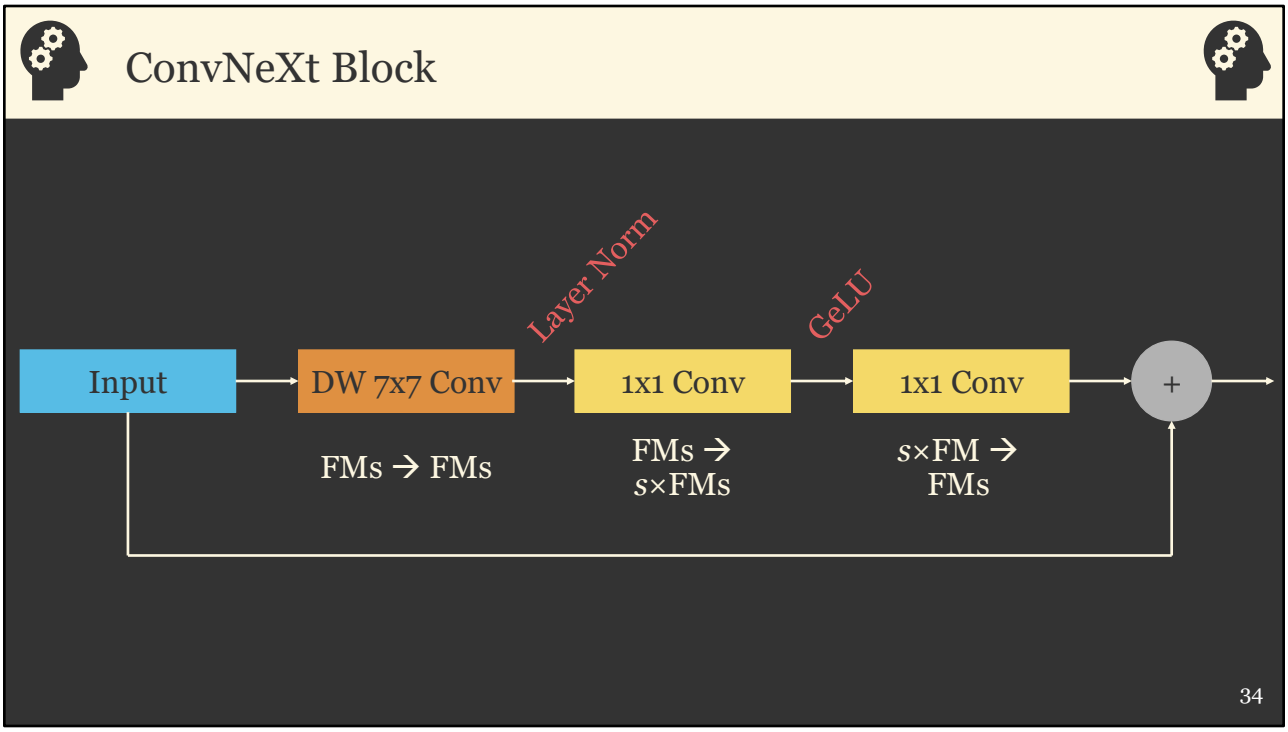
ConvNeXt Characteristics

- Patchify stem
- Large kernel (7x7)
- Inverted bottleneck block
- Depthwise separable convolution
- Layer normalization
- Less use of nonlinearities/activation functions
- GeLU

Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T. and Xie, S., 2022. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11976-11986).

This slide lists the key design characteristics of ConvNeXt. These features distinguish it from a standard ResNet and more traditional CNNs. Key characteristics include the following:

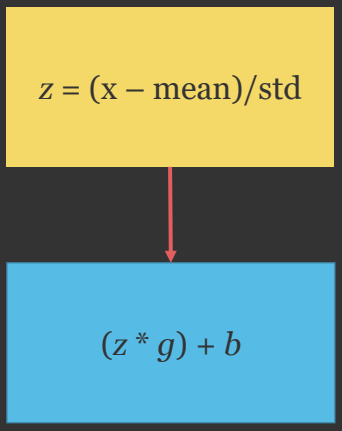
1. The patchify stem replaces the traditional 7x7 conv + max pool entry
2. Large 7x7 depthwise kernels expand receptive fields
3. The inverted bottleneck expands channels in the middle layer
4. Depthwise separable convolutions reduce computational cost
5. Layer normalization replaces batch normalization
6. Fewer nonlinearities reduce redundancy
7. Gaussian Error Linear Unit (GELU) replaces ReLU for smoother gradients



This diagram illustrates the ConvNeXt block structure. The input passes through a Depthwise 7x7 convolution for spatial mixing, followed by Layer Normalization, two 1x1 pointwise convolutions (with a GELU activation in between) for channel mixing, and a residual skip connection. The inverted bottleneck is visible here: the channel dimension expands (s×FMs) in the middle then contracts back. This mirrors the structure of a Transformer block.

Layer Normalization

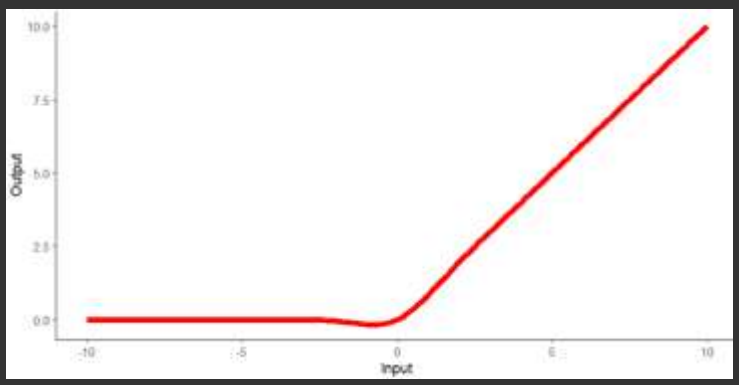
- Batch Normalization
  - Applies normalization per channel for samples within a mini-batch
  - Separate *g* and *b* terms for each channel
  - Make each feature channel have consistent statistics across the batch
  - Uses other samples to normalize each sample
- Layer Normalization
  - Applies normalization per sample
  - Separate *g* and b terms for each sample in mini-batch
  - Feature maps for each sample are collectively normalized
  - Each sample normalized separately
  - Normalize the entire feature vector of a single sample
  - No interaction between samples
  - More stable for small batches



This slide contrasts Batch Normalization and Layer Normalization. Batch Normalization normalizes across the batch dimension per channel, requiring statistics from multiple samples and can be unstable with small batches. Layer Normalization normalizes each sample independently across its entire feature vector, making it more suitable for variable batch sizes and self-supervised settings. ConvNeXt adopts Layer Normalization to better align with Transformer design practices.

GELU

- Gaussian Error Linear Unit
- Cumulative distribution function of the standard Gaussian distribution
- Approximated as:



$$= 0.5x(1 + (1 + \tanh(\sqrt{2/\pi}(x + 0.044715\, x^3)))$$

GELU (Gaussian Error Linear Unit) is a smooth activation function that gates the input based on its probability under a standard Gaussian distribution. Unlike ReLU which has a hard zero cutoff, GELU allows small negative values with a smooth transition, which is beneficial for gradient flow during training. It has become the standard activation in Transformer models and is adopted by ConvNeXt for consistency.

Mamba

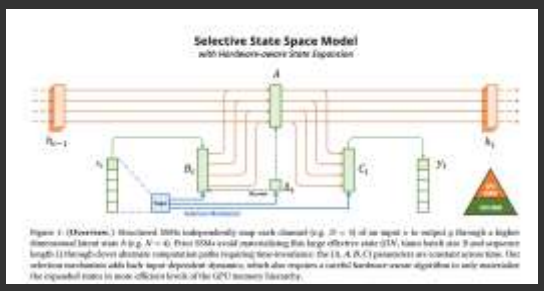
Mamba is a State Space Model (SSM) that offers an alternative to Transformers for processing sequential data. It is particularly notable for achieving linear rather than quadratic computational complexity with respect to sequence length, making it more efficient for long sequences.

Mamba

- Process time series or sequence data
- Selective state space model (SSM)
- More computationally efficient than transformers
- Hidden state maintains important information
- Problem must be discretized using zero-order hold method
- Selective since output depends on input

$$h' = Ah(t) + Bx(t)$$

$$y(t) = Ch(t) + Dx(t)$$



Gu, A. and Dao, T., 2023. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*.

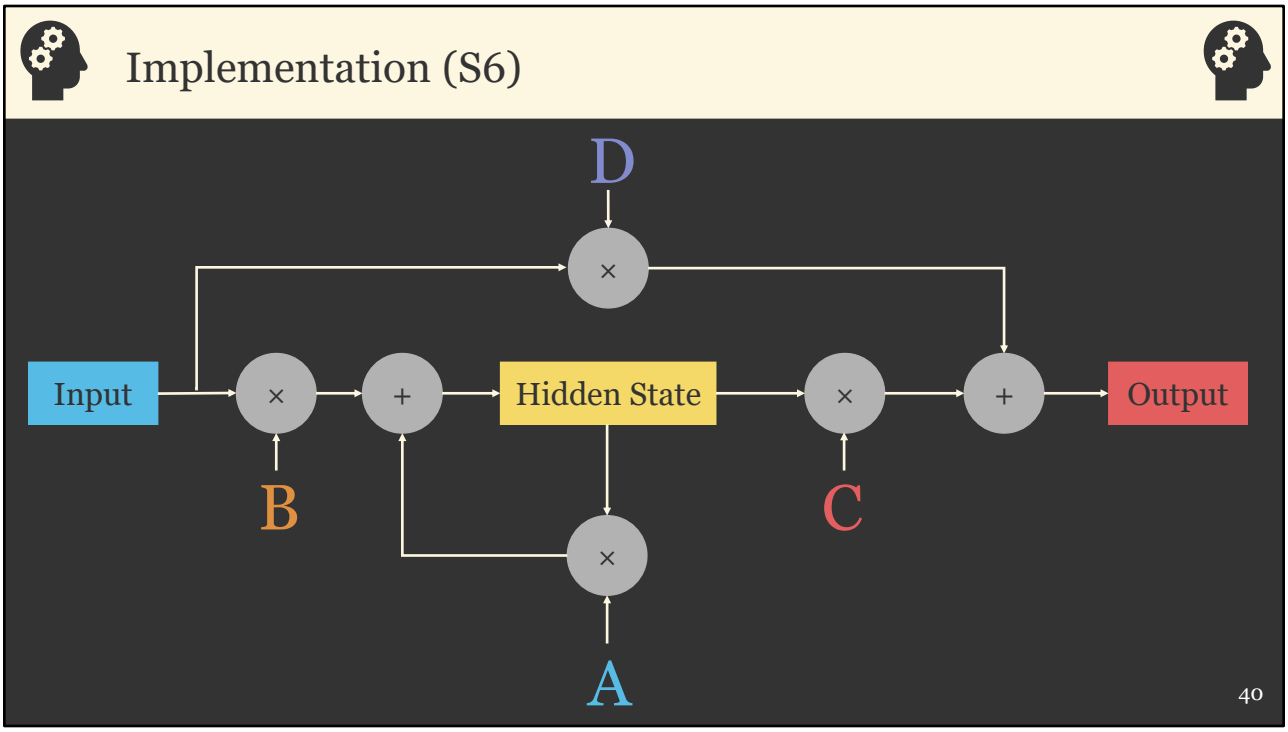
Mamba is a selective state space model designed for processing time series and sequential data. Unlike Transformers which use attention over all positions, Mamba maintains a hidden state that summarizes past information. The 'selective' nature means its transition matrices (A, B, C) depend on the input, allowing it to adaptively focus on or ignore parts of the input sequence. The continuous-time formulation must be discretized for use in practice.

Matrices

- A
  - how much past information is maintained
  - how fast information decays
  - Long-term vs. short-term memory
  - Input dependent
- B
  - How strongly current input impacts the memory
  - What from the current input should be remembered
  - Input dependent
- C
  - What part of the memory is used
  - Transform to useful output
  - Input dependent
- D
  - Bypass path
  - Immediate influence of the input
  - Current input not passed through memory
  - Helps with gradient flow

Matrix	Function	Analogy
A	Memory dynamics	How memory evolves
B	Write gate	What to store
C	Read gate	What to use
D	Skip connection	Immediate effect

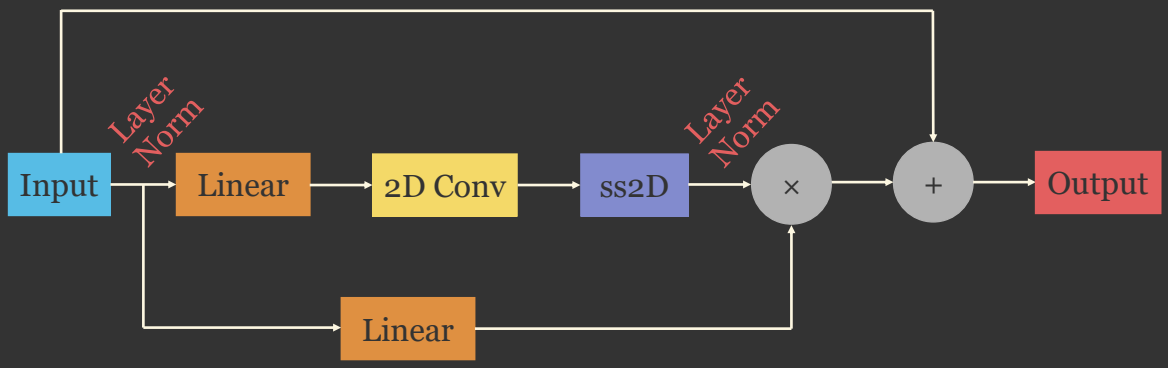
This slide explains the four key matrices in Mamba. Matrix A controls how fast information decays and how much the past state is retained (memory dynamics). Matrix B determines what from the current input gets written into memory (write gate). Matrix C determines what is read from the hidden state to produce the output (read gate). Matrix D is a bypass connection that allows the current input to directly influence the output without going through the memory state, aiding gradient flow.



This diagram shows the S6 (selective scan) implementation of Mamba. The input is projected via B to update the hidden state, which is also multiplied by A to implement memory decay. The hidden state is then read via C to produce the primary output, and a direct skip connection through D adds the raw input to the output. This allows Mamba to selectively remember, update, and output relevant information from a sequence.

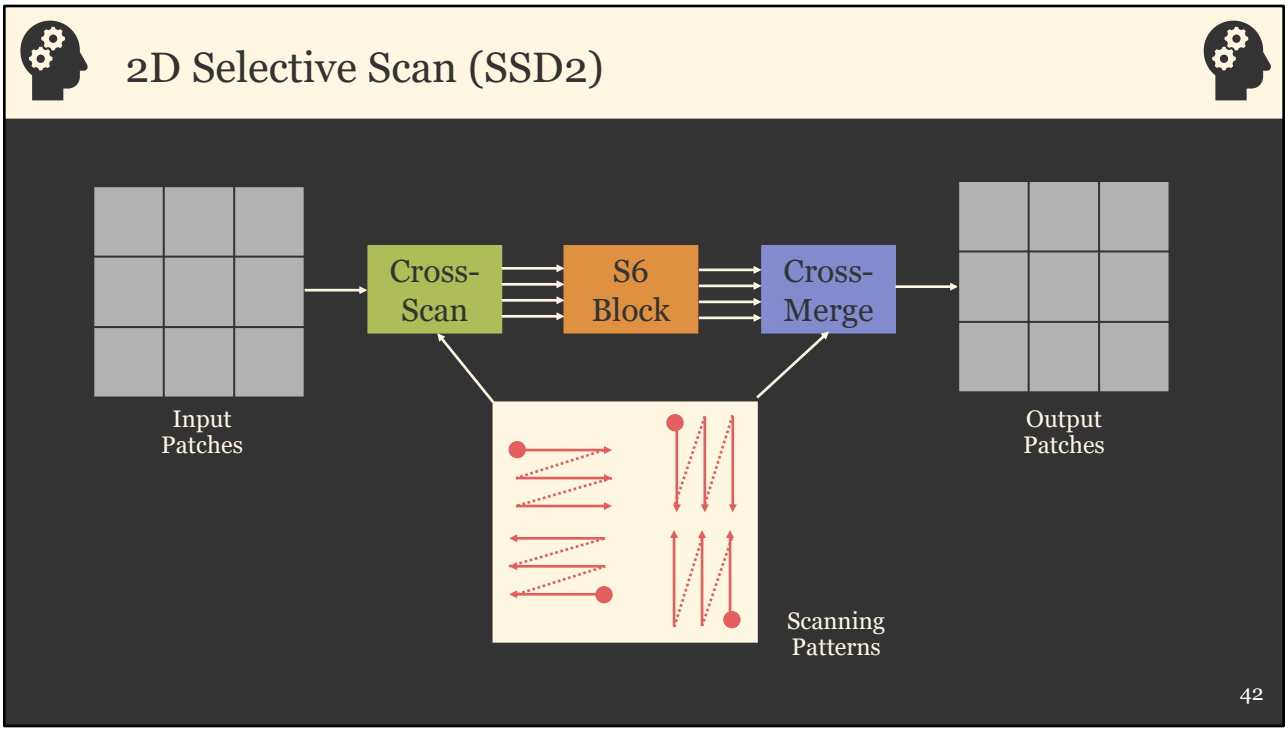
VMamba

- Augmentation of Mamba for vision tasks
- Composed of vision stated space (VSS) blocks

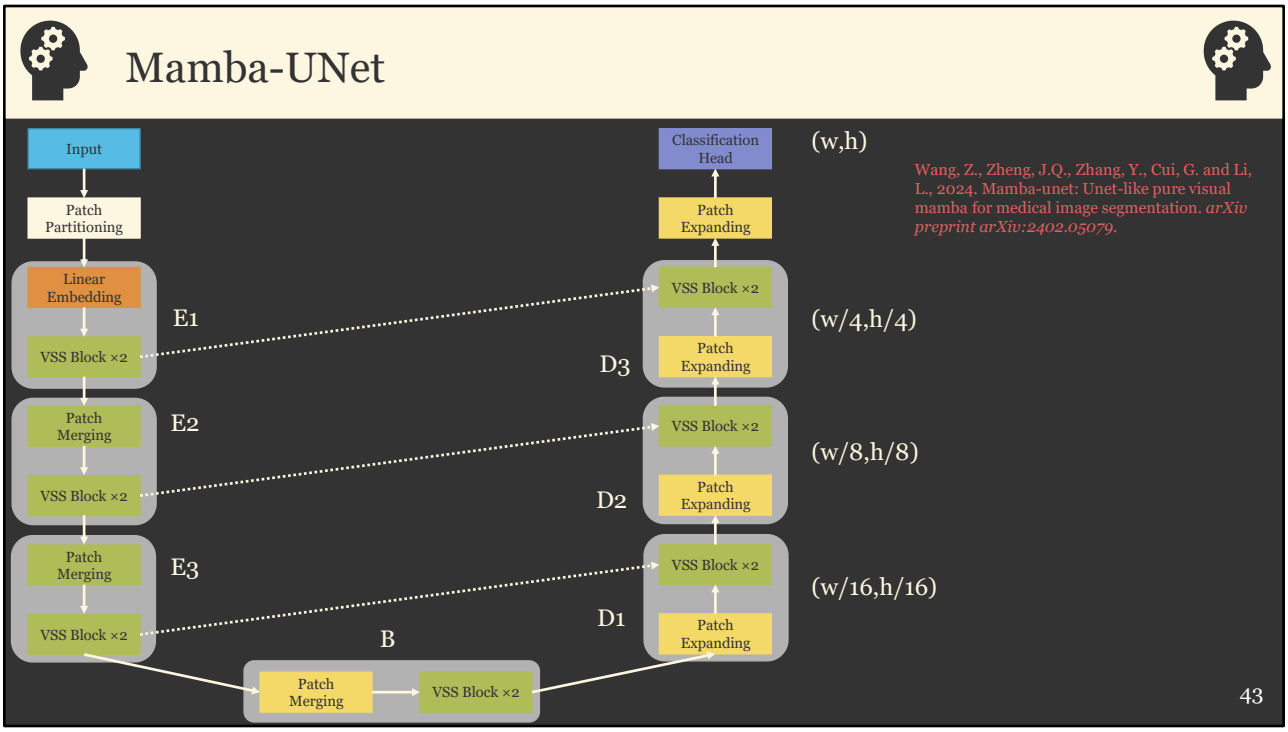


Liu, Y., Tian, Y., Zhao, Y., Yu, H., Xie, L., Wang, Y., Ye, Q., Jiao, J. and Liu, Y., 2024. Vmamba: Visual state space model. *Advances in neural information processing systems*, *37*, pp.103031-103063.

VMamba adapts the Mamba architecture for 2D vision tasks. It introduces the Vision State Space (VSS) block, which processes image feature maps using a 2D selective scan module (SS2D). The VSS block applies a cross-scan strategy to traverse the image in multiple directions before merging, enabling global spatial context without the quadratic cost of self-attention.

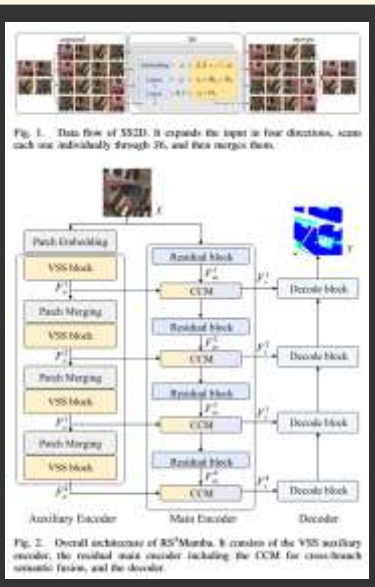


The 2D Selective Scan (SS2D) in VMamba addresses the challenge that image data is 2D while Mamba's SSM is inherently 1D. The cross-scan strategy unfolds the 2D image patches into four 1D sequences by scanning in four directions (e.g., top-left to bottom-right and its reversal). Each is processed by an S6 block, and the results are merged back into 2D, providing comprehensive spatial coverage.



Mamba-UNet replaces Swin Transformer blocks in Swin-UNet with Visual State Space (VSS) blocks. The overall architecture mirrors Swin-UNet, using Patch Partitioning, linear embedding, encoder stages with Patch Merging, a bottleneck, and decoder stages with Patch Expanding; however, all Swin blocks are substituted with VSS blocks.

RS³Mamba



- CNN-based encoder
- Auxiliary Mamba-based encoder
- Merge information using collaborative completion module (CCM)
- UNetformer-based decoder (transformer)

Ma, X., Zhang, X. and Pun, M.O., 2024. Rs 3 mamba: Visual state space model for remote sensing image semantic segmentation. *IEEE Geoscience and Remote Sensing Letters*, *21*, pp.1-5.

RS³Mamba is a hybrid architecture for remote sensing semantic segmentation. It uses a CNN-based encoder for local feature extraction and a Mamba-based auxiliary encoder for capturing long-range dependencies. A Collaborative Completion Module (CCM) fuses the features from both branches. The decoder follows a UNetformer design using Transformer blocks, combining the strengths of CNNs, Mamba SSMs, and Transformers in a single pipeline.

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

This concluding slide marks the end of the lecture module. Students are directed to return to the West Virginia View course page to access additional content, assignments, or the next module. This module covered ConvNeXt, the Transformer architecture and its vision adaptations (ViT, Swin, SegFormer, Swin-UNet), and Mamba-based models (VMamba, Mamba-UNet, RS3Mamba).