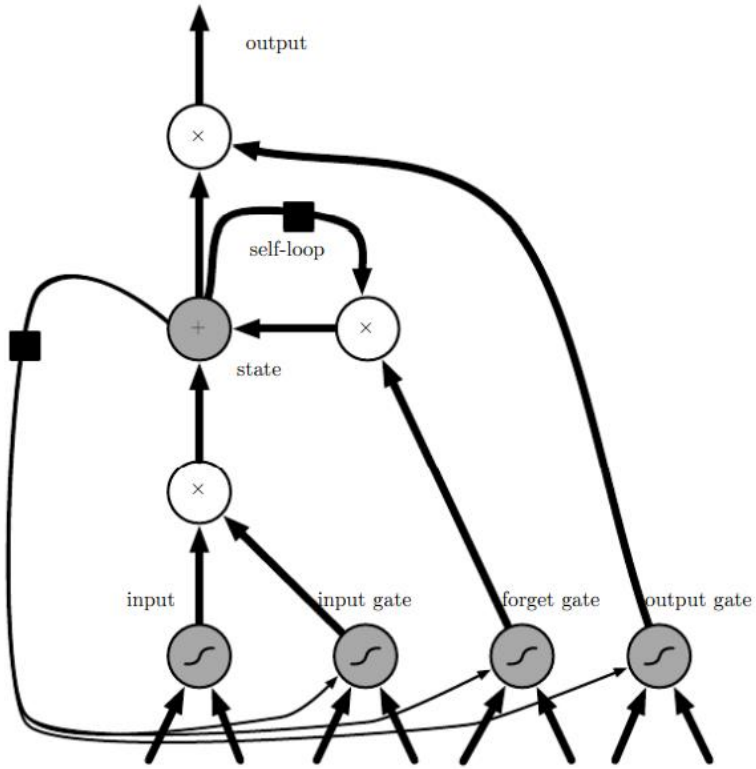
ECE 4252/8803: Fundamentals of Machine Learning (FunML) Fall 2024

Lecture 21: Sequence Modeling



OLIVES @GeorgiaTech

Georgia Tech

Overview

In this Lecture..

Temporal Convolutional Networks

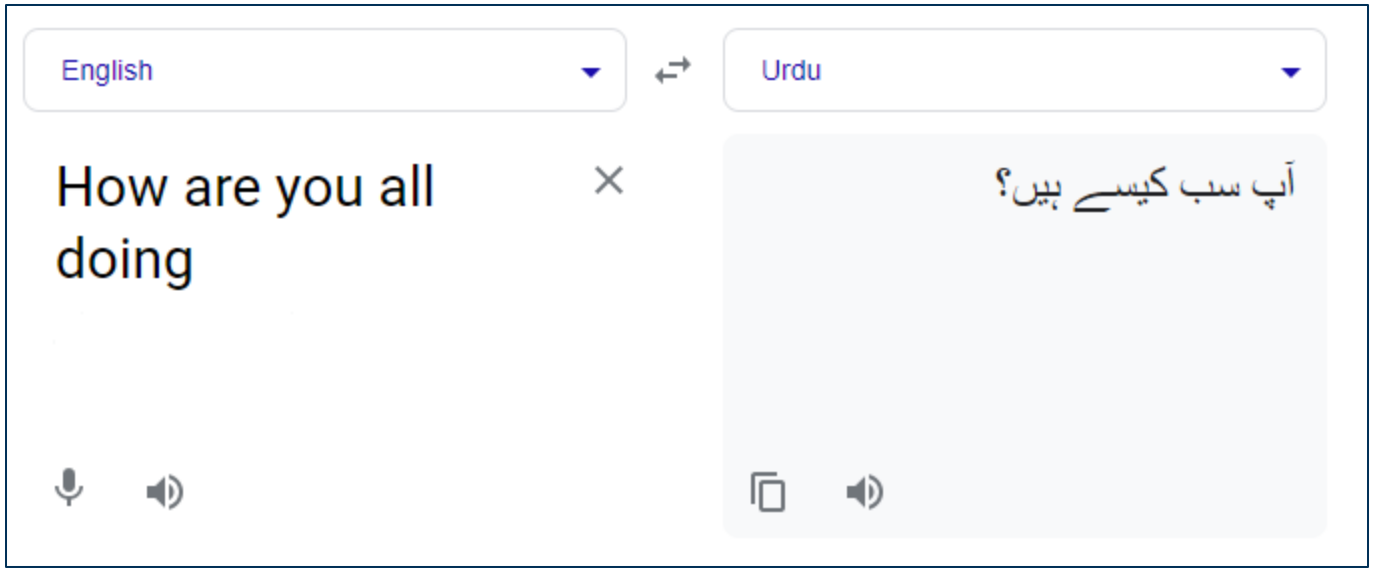
- Causal Convolutions
- Dilated Convolutions
- Weight Normalization
- Residual Units

Encoder-Decoder Architectures

Vision Transformer

Sequence Modeling

Sequence to Sequence Generation

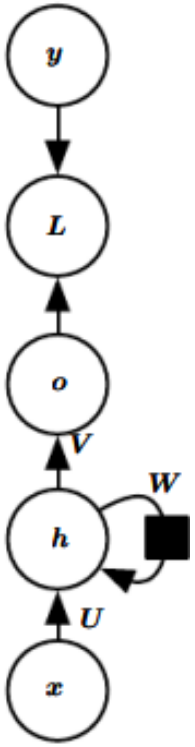


Machine translation

[1] https://blog.floydhub.com/gentle-introduction-to-text-summarization-in-machine-learning/

Recurrent Neural Networks

RNN Update Equations



$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$ (Computing the neuron activations)

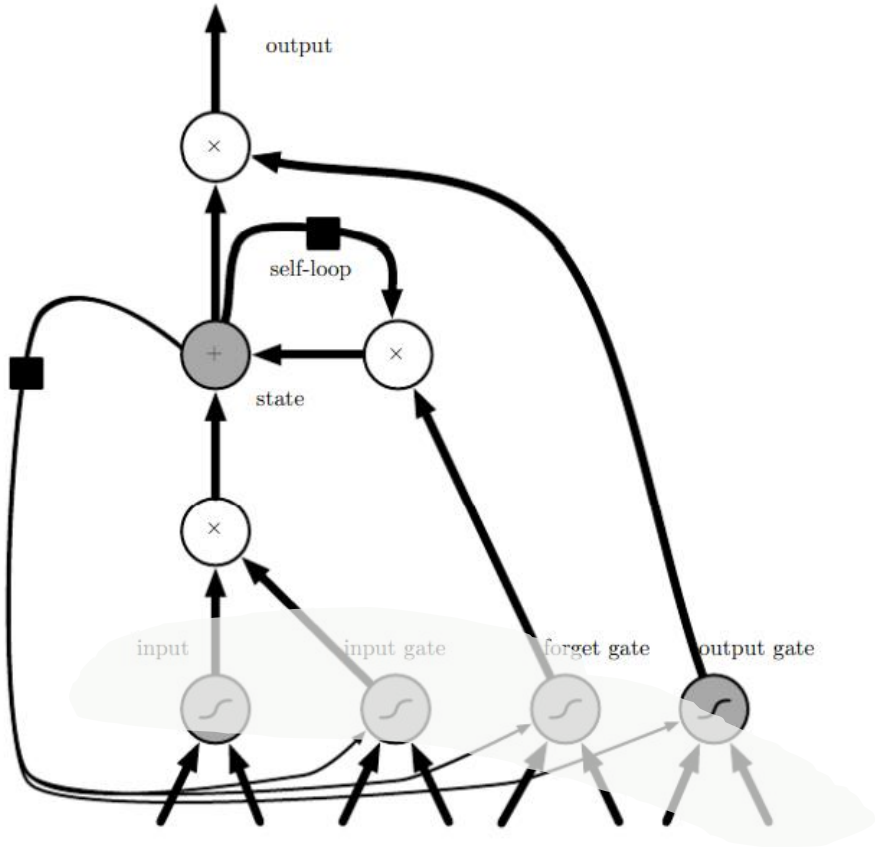
$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)})$ (Applying non-linear activation)

$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$ (Computing output vector as a function of present hidden state vector)

$\boldsymbol{y}^{(t)} = \mathrm{softmax}(\boldsymbol{o}^{(t)})$ (Applying softmax to obtain pseudo-probabilities)

Long-Short Term Memory (LSTMs) Networks

LSTM Cell



An LSTM cell for one time step [1]

- The gate update equations are:

$$\boldsymbol{f}^{(t)} = \boldsymbol{\sigma}\left(\boldsymbol{b}^{f} + \boldsymbol{U}^{f}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{f}\boldsymbol{h}^{(t-1)}\right)$$
$$\boldsymbol{g}^{(t)} = \boldsymbol{\sigma}\left(\boldsymbol{b}^{g} + \boldsymbol{U}^{g}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{g}\boldsymbol{h}^{(t-1)}\right)$$

Hence the gates themselves are learned parameters conditioned on the context!

- The output equations are:

$$\boldsymbol{h}^{(t)} = \tanh\left(\boldsymbol{s}^{(t)}\right) \circ \boldsymbol{q}^{(t)}$$
$$\boldsymbol{q}^{(t)} = \boldsymbol{\sigma}\left(\boldsymbol{b}^{o} + \boldsymbol{U}^{o}\boldsymbol{x}^{(t)} + \boldsymbol{W}^{o}\boldsymbol{h}^{(t-1)}\right)$$

- $\boldsymbol{q}^{(t)}$ is output gate also determined via learned parameters through context.

[1] Chapter 10, *Deep Learning*, Yoshua Bengio and Ian Goodfellow

Temporal Convolution Networks

Computation

1D CNNs work by sliding a fixed set of kernels through the sequence. This operation can be actualized via multiplication of a Toeplitz matrix with a vector, and hence parallelizable, leading to faster computations.

Convolution of the above sequence (blue) with the kernel (red) is equivalent to below:

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Temporal Convolution Networks

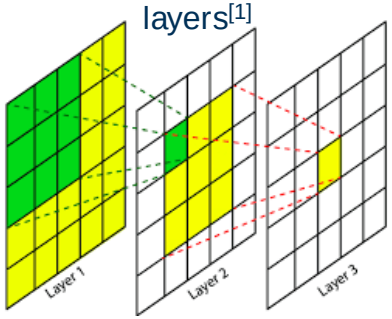
Dilated Convolution

- However, CNNs require very deep architectures to cover the input receptive field
- Solution: Dilated convolution

- Dilated convolution expands kernels by adding ‘holes’
- Allows larger receptive field in non-deep layers

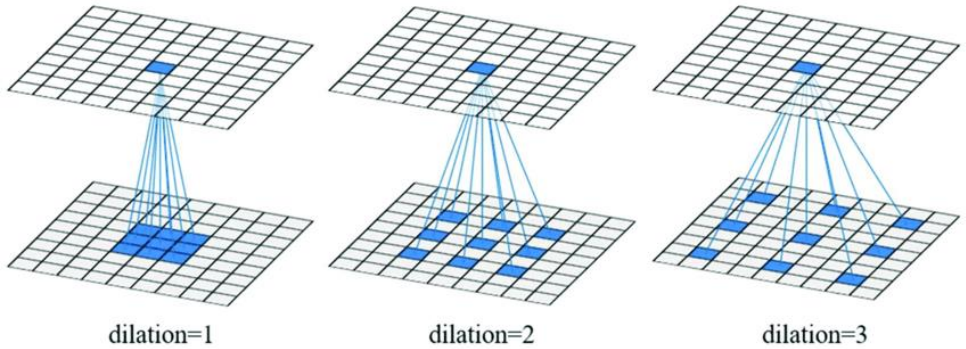
- Used in image processing and audio processing applications

Stacking multiple layers[1]



Multiresolution kernels

Dilated Convolutions[2]

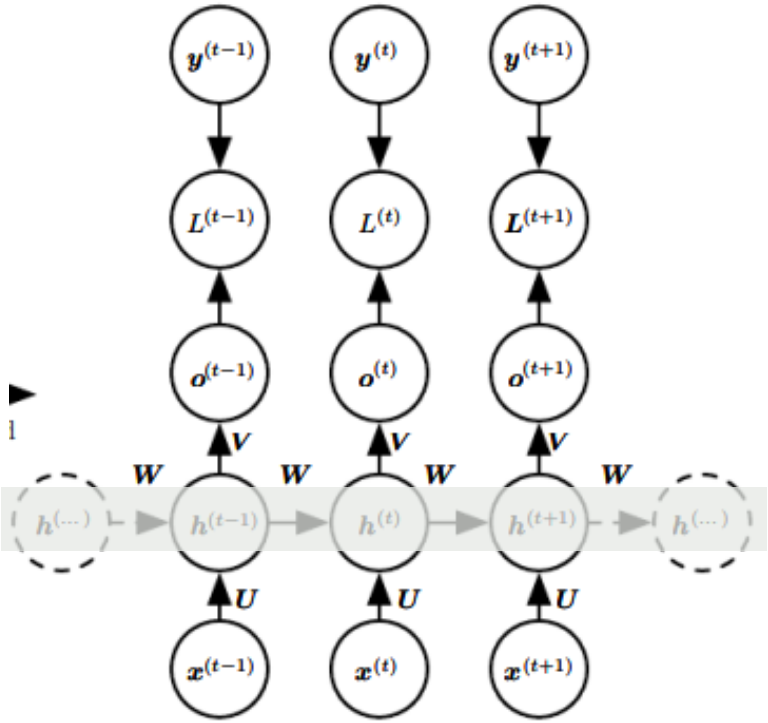


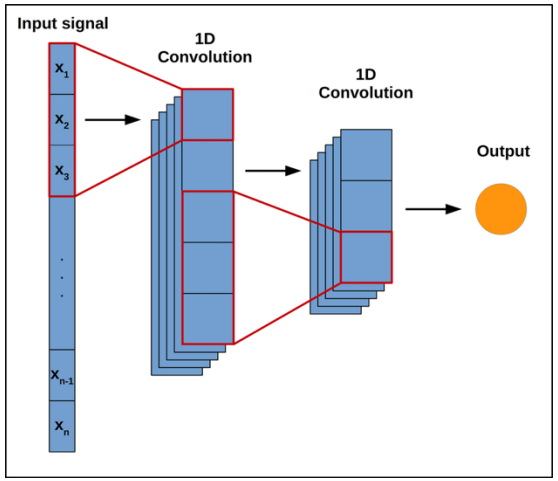
[1] https://theaisummer.com/receptive-field/
[2] Cui, Ximin, et al. "Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification." *Remote Sensing* 11.19 (2019): 2220.

Temporal Convolution Networks

Computational Complexity

- CNNs only use a fixed set of kernels per layer. RNNs store many intermediate state variables increasing the computational bottleneck





Temporal Convolution Networks

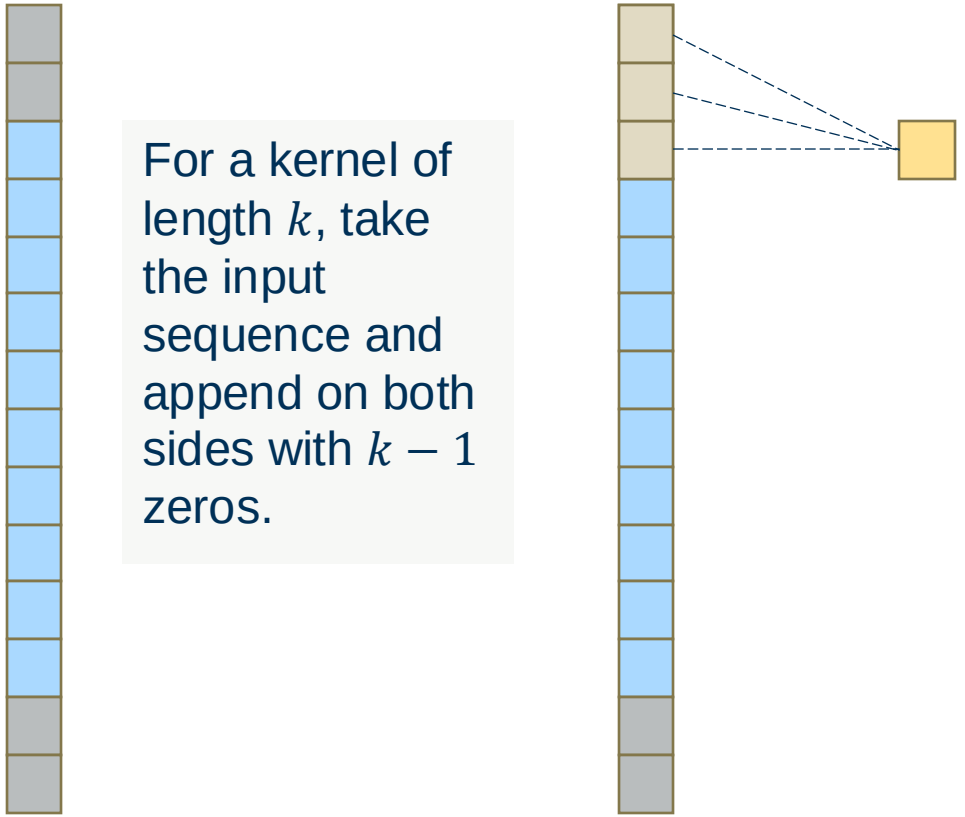
Advantages

- **Input sequence invariability:** The TCN operates on a sequence regardless of its length to produce another sequence of the same length (similar to an RNN)
- **Causal sequence modeling:** Each term in the output sequence is a function of only past samples in the input sequence. This is achieved by specifying certain combinations of dilation and padding hyperparameters (more on this later)
- **Residual units for ease of optimization:** The TCN is composed of multiple, stacked units mimicking residual units in 2D CNNs for vision tasks. This allows for a deeper network that can be efficiently optimized without running into gradient-related issues
- **Optimal receptive field design:** The network uses dilated causal convolutions over multiple layers to sufficiently cover the entire context of past input samples needed to predict a given sample for the output sequence

Temporal Convolution Networks

Causal Convolutions

$$y^{(t)} = f(x^{(t)}, x^{(t-1)}, \dots, x^{(1)})$$



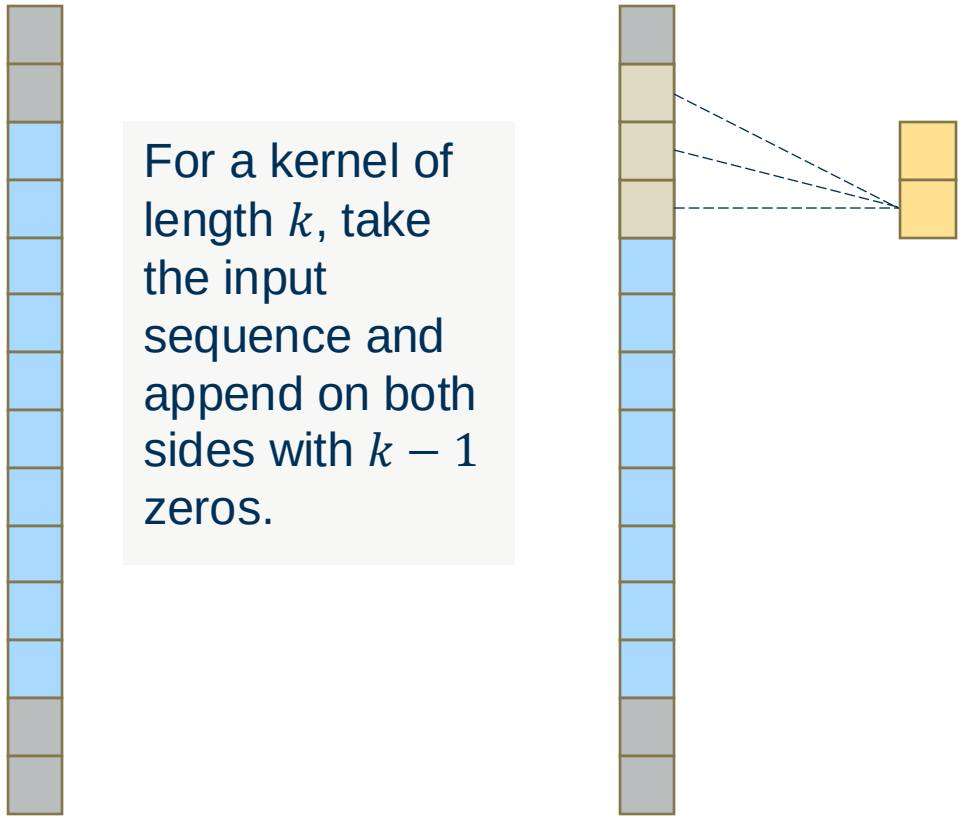
For a kernel of length k, take the input sequence and append on both sides with $k-1$ zeros.

As the kernel slides along the padded input sequence, each output sample is produced as function of only past input samples

Temporal Convolution Networks

Causal Convolutions

$$y^{(t)} = f(x^{(t)}, x^{(t-1)}, \dots, x^{(1)})$$



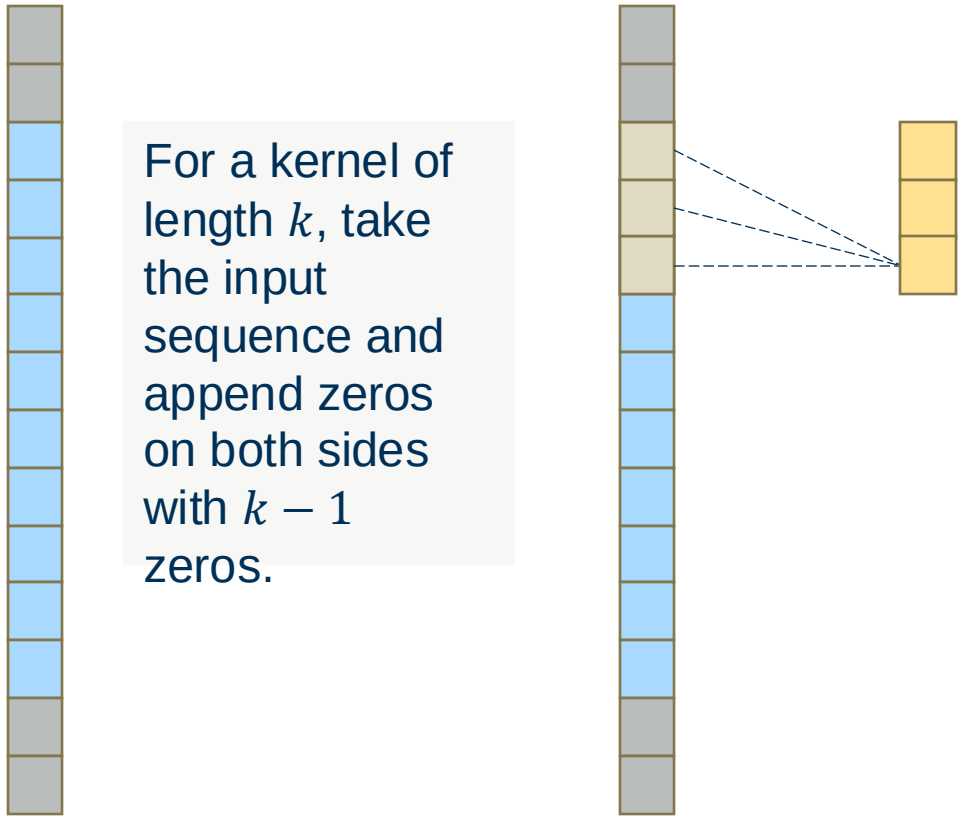
For a kernel of length k, take the input sequence and append on both sides with $k-1$ zeros.

As the kernel slides along the padded input sequence, each output sample is produced as function of only past input samples

Temporal Convolution Networks

Causal Convolutions

$$y^{(t)} = f(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)})$$



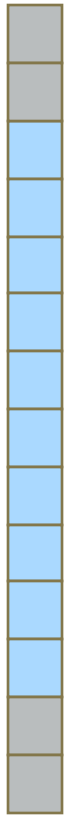
For a kernel of length k, take the input sequence and append zeros on both sides with $k-1$ zeros.

As the kernel slides along the padded input sequence, each output sample is produced as function of only past input samples

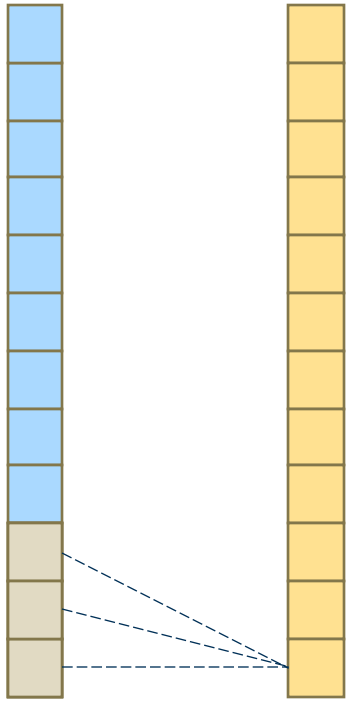
Temporal Convolution Networks

Causal Convolutions

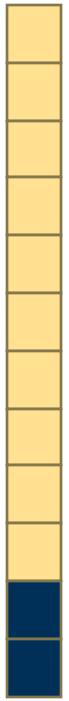
$$y^{(t)} = f(x^{(t)}, x^{(t-1)}, \ldots, x^{(1)})$$



For a kernel of length k, take the input sequence and append on both sides with $k-1$ zeros.



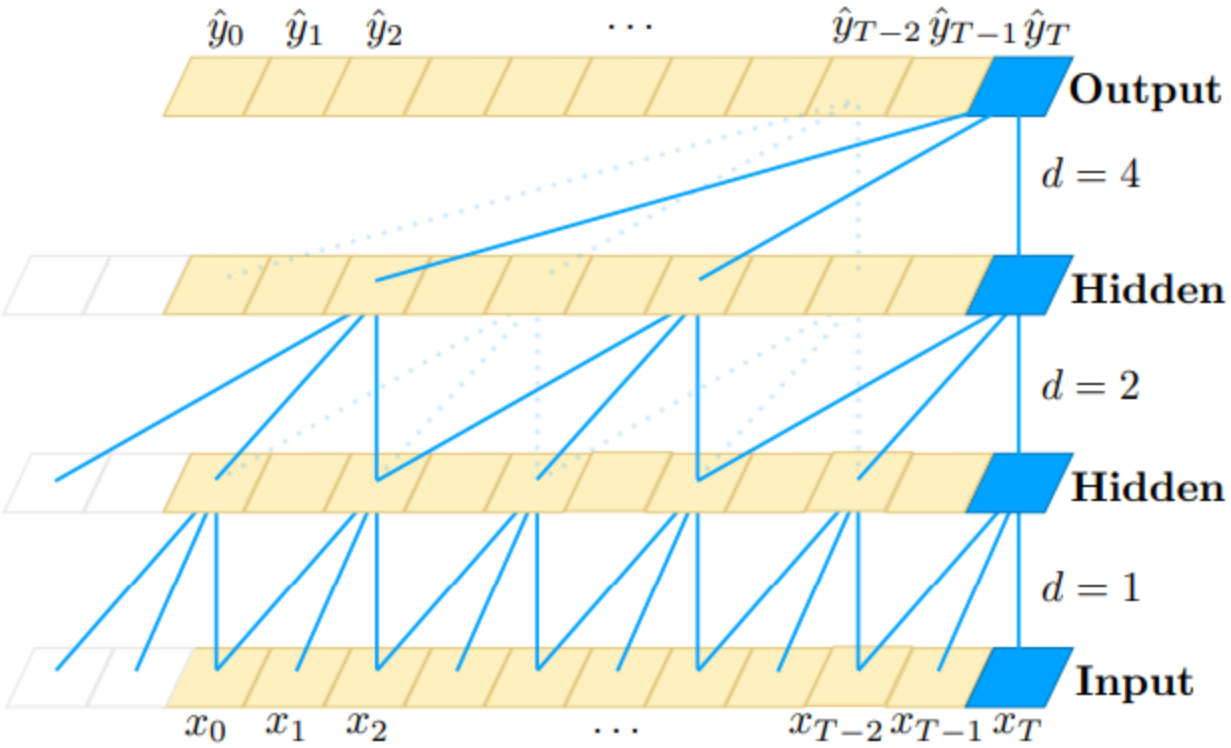
As the kernel slides along the padded input sequence, each output sample is produced as function of only past input samples



Remove the last $k-1$ samples from the output to obtain a sequence equal in length to the input and a causal function of the input samples

Temporal Convolution Networks

Dilated Convolutions



Multiple, stacked dilated convolutions [1]

Network should have sufficient depth to cover the entire context of input length to predict the last output sample

- Increasing the dilation factor helps expand the effective receptive field of the kernel while maintaining the number of learnable parameters constant.
- TCN used exponentially increasing dilation factors to increase the receptive field at an exponential rate through the network layers.
- Number of zeros padded to each side of the input sequence then becomes $d(k-1)$, where d and k refer to the dilation factor and kernel size respectively. The convolution still occurs causally as before.

[1] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." *arXiv preprint arXiv:1803.01271* (2018).

Temporal Convolution Networks

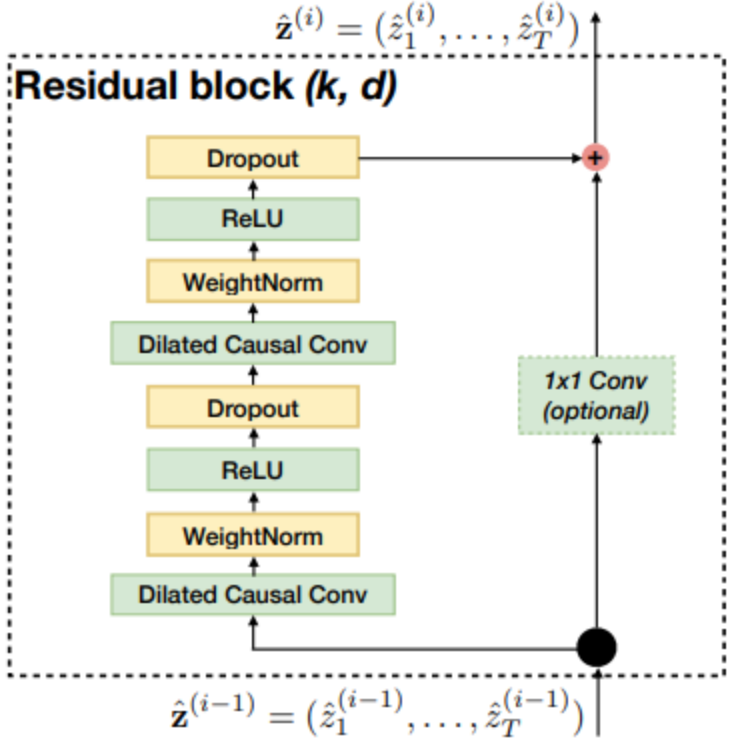
Weight Normalization

- Batch Norm: Requires a large batch size for stable calculations
  - Not good for RNNs/LSTMs
- Alternative is Weight Normalization $w = \frac{g\, v}{||v||}$
- Directly optimize for g and v.
  - g = weight vector
  - $v/||v||$ = direction of weight vector
- Stochastic gradient descent convergence is sped up
- Weight norm is more robust to learning rates

[1] Salimans, Tim, and Durk P. Kingma. "Weight normalization: A simple reparameterization to accelerate training of deep neural networks." *Advances in neural information processing systems* 29 (2016)..

Temporal Convolution Networks

Residual Units



Structure of a single residual block in a TCN [1]

- The network architecture consists of **multiple residual blocks** laterally stacked together. Each block has the same general structure.
- **Dilated causal convolution** layers extract features from the input sequence as discussed earlier
- **Weight normalization** layers decouple gradient magnitudes from their direction to help speed convergence
- **ReLU layers** introduce non-linearity into the learning process
- **Dropout layers** reduce the risk of overfitting by randomly turning a certain prespecified portion of activations off for every minibatch of examples
- **Skip connection** through a 1×1 conv layer maps the input sequence to have the same number of channels as the output of the last dropout layer in the block. This allows the input and output sequences to be added together via a skip connection

[1] Bai, Shaojie, J. Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling." *arXiv preprint arXiv:1803.01271* (2018)., https://github.com/locuslab/TCN?tab=readme-ov-file

Overview

In this Lecture..

Temporal Convolutional Networks

Encoder-Decoder Architectures

- Word Embeddings
- Word2Vec
- Encoder-Decoder
- Skip-grams
- Continuous Bag-of-Words
- Seq2seq
- Dot product attention

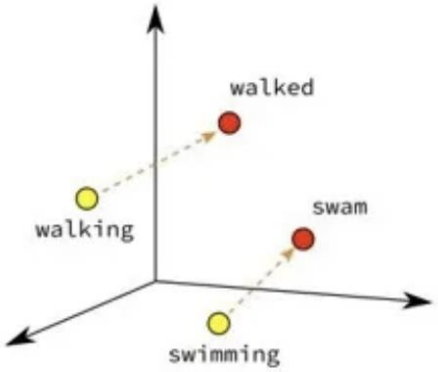
Vision Transformer

Encoder-Decoder Architectures

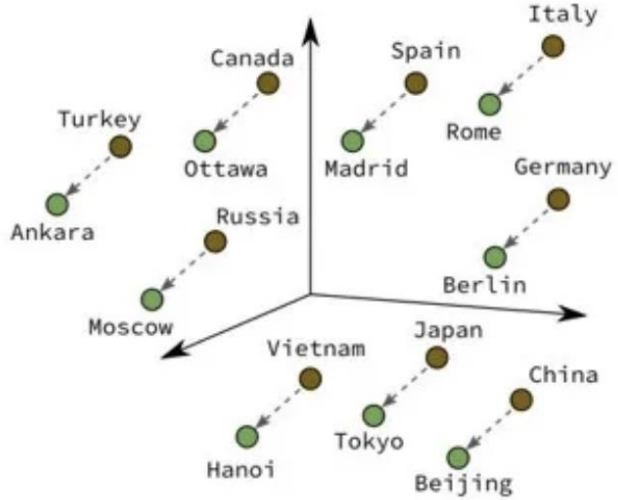
Word Embeddings

- Embeddings: Projections of the smallest meaningful elements in data into meaningful vectors
- In NLP, smallest meaningful elements are words

- In Lectures 18-19, Autoencoders were used to obtain embeddings



Verb Tense



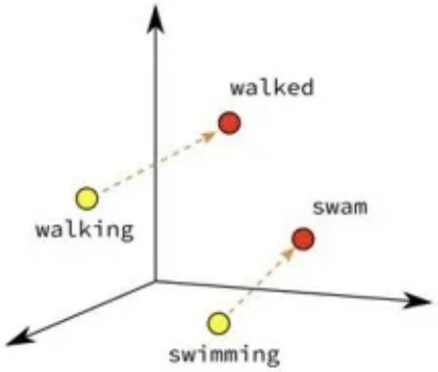
Country-Capital

https://medium.com/@RobinVetsch/nlp-from-word-embedding-to-transformers-76ae124e6281

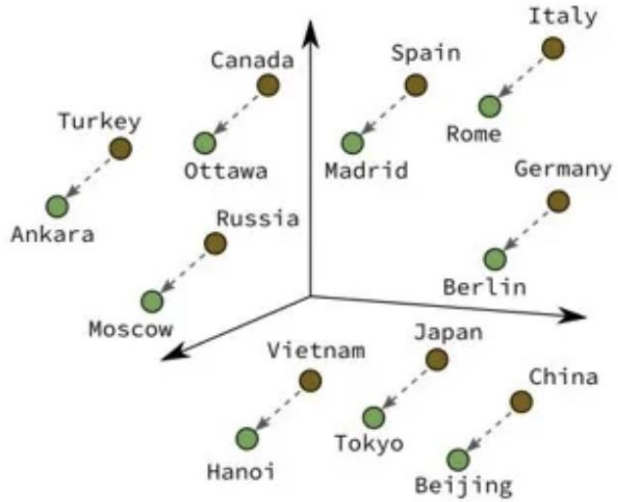
Encoder-Decoder Architectures

Properties of Embeddings Space

- Embedding space consists of contextually similar words in close proximity
- Encoder-Decoder architectures can be used to obtain contextual embeddings
- Autoencoders ≠ Encoder-Decoder



Verb Tense



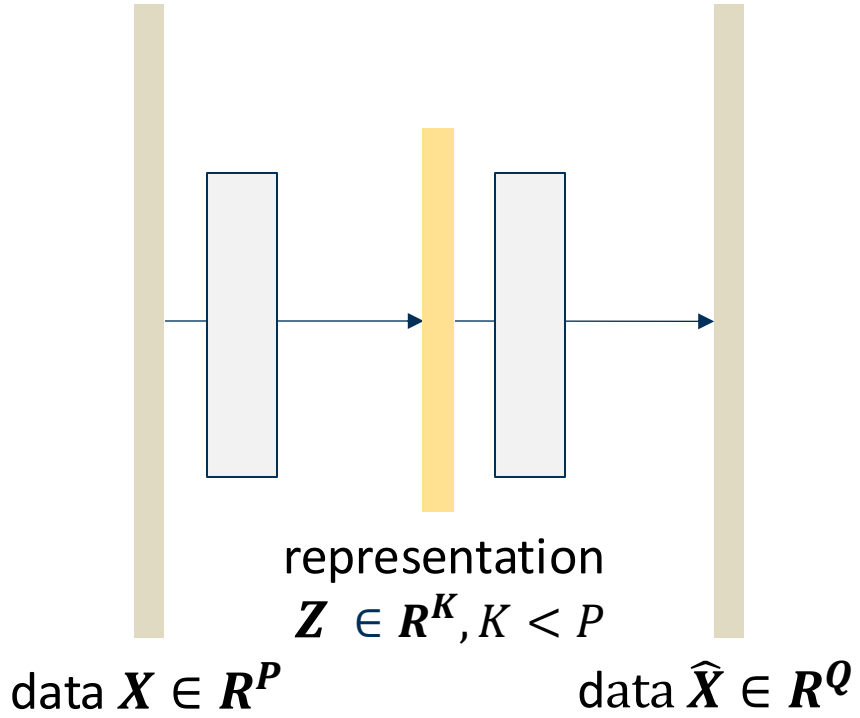
Country-Capital

https://medium.com/@RobinVetsch/nlp-from-word-embedding-to-transformers-76ae124e6281

Encoder-Decoder Architectures

Word2Vec: One-word context

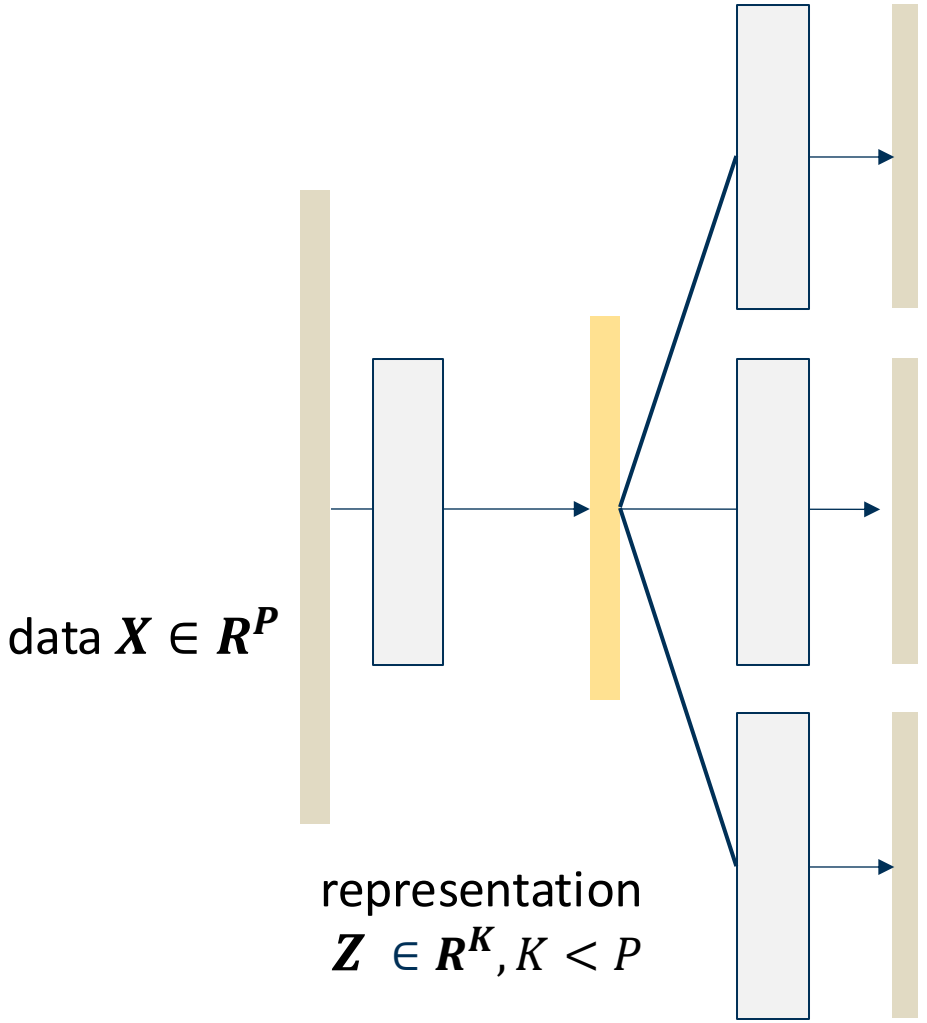
- Given an input word, learn to predict output word
- Rarely used since context, by definition, depends on surrounding words in the sequence



Encoder-Decoder Architectures

Word2Vec: Skip-gram model

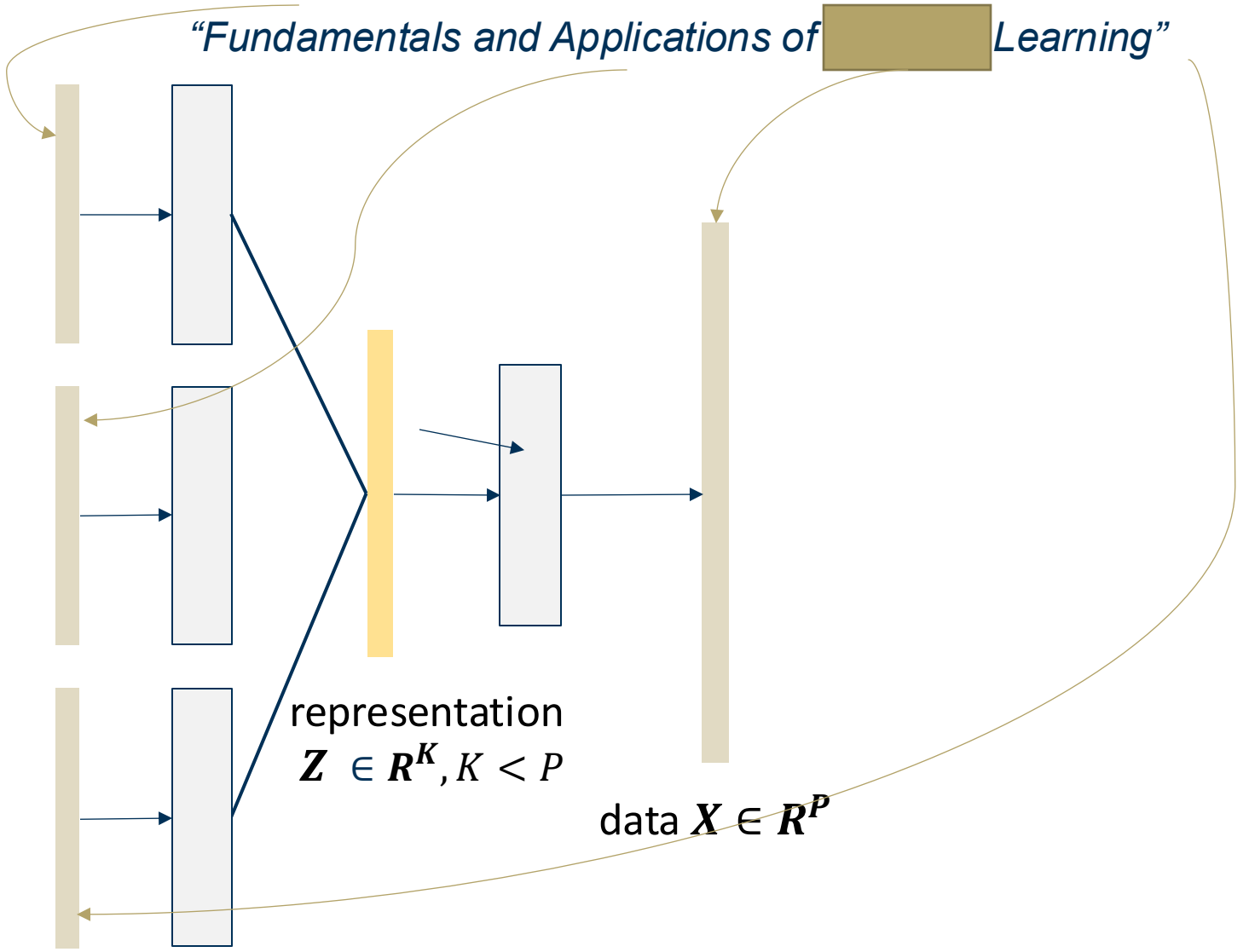
- Given an input word, learn to predict neighboring words within the sentence
- Neighboring words provide the context



Encoder-Decoder Architectures

Word2Vec: Continuous Bag-of-Words

- Learn to predict a target given neighboring words within the sentence
- Neighboring words provide the context
- Intuition: We mask the target word and allow the network to predict it

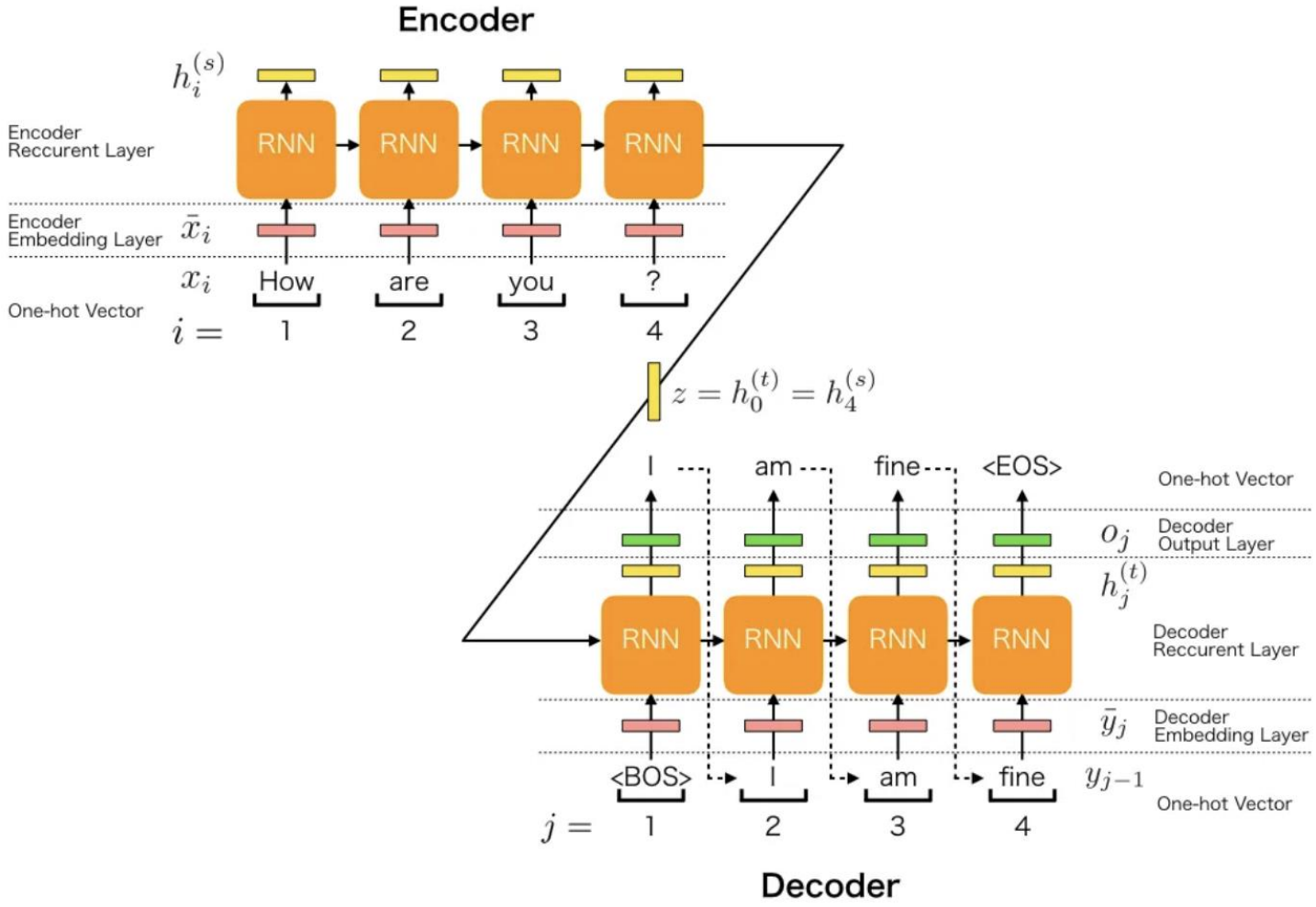


Encoder-Decoder Architectures

Seq2seq

- Word2vec: Learn contextual embedding of words
- Seq2seq: Learn contextual embeddings of sequences

- Advantages: Map sequences of any length to an embedding of fixed dimensions
- Works well on short sequences



Encoder-Decoder Architectures

Seq2seq: Limitations

- Limitations: Decoder must construct different sequences based on same embedding
- Solution: Add attention mechanism at the decoder stage

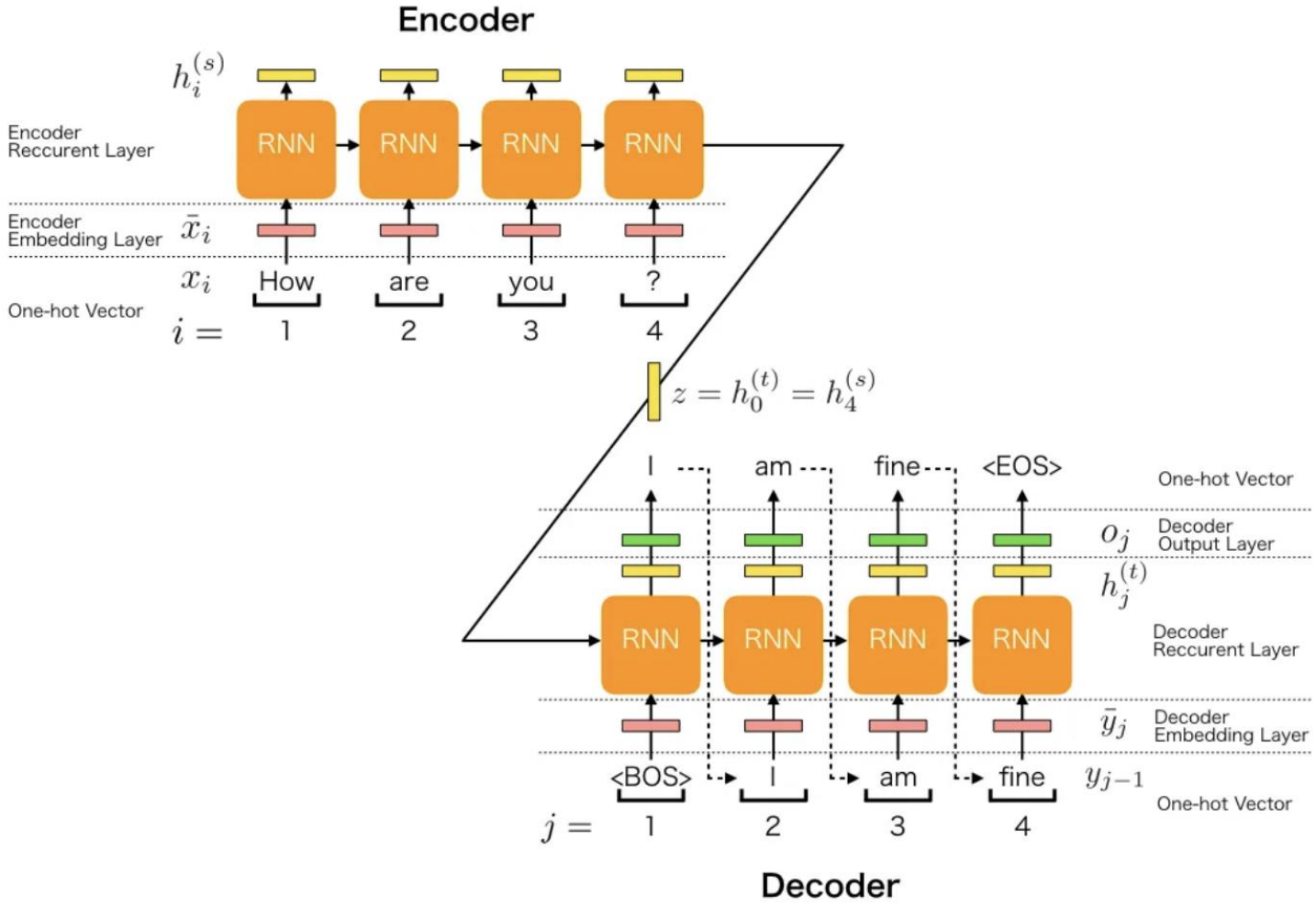
Question: How many people are wearing a plain brown shirt?

Answer: One person is wearing a plain brown shirt

Encoder-Decoder Architectures

Seq2seq: Dot product attention

- At each decoder step, add an attention mechanism based on input
- Simple attention: Dot product
  - Learn attention vector s_k for every decoder stage k
  - Attention = $h_t^T s_k$ where h_t is the contextual embedding



Overview

In this Lecture..

Temporal Convolutional Networks

Encoder-Decoder Architectures

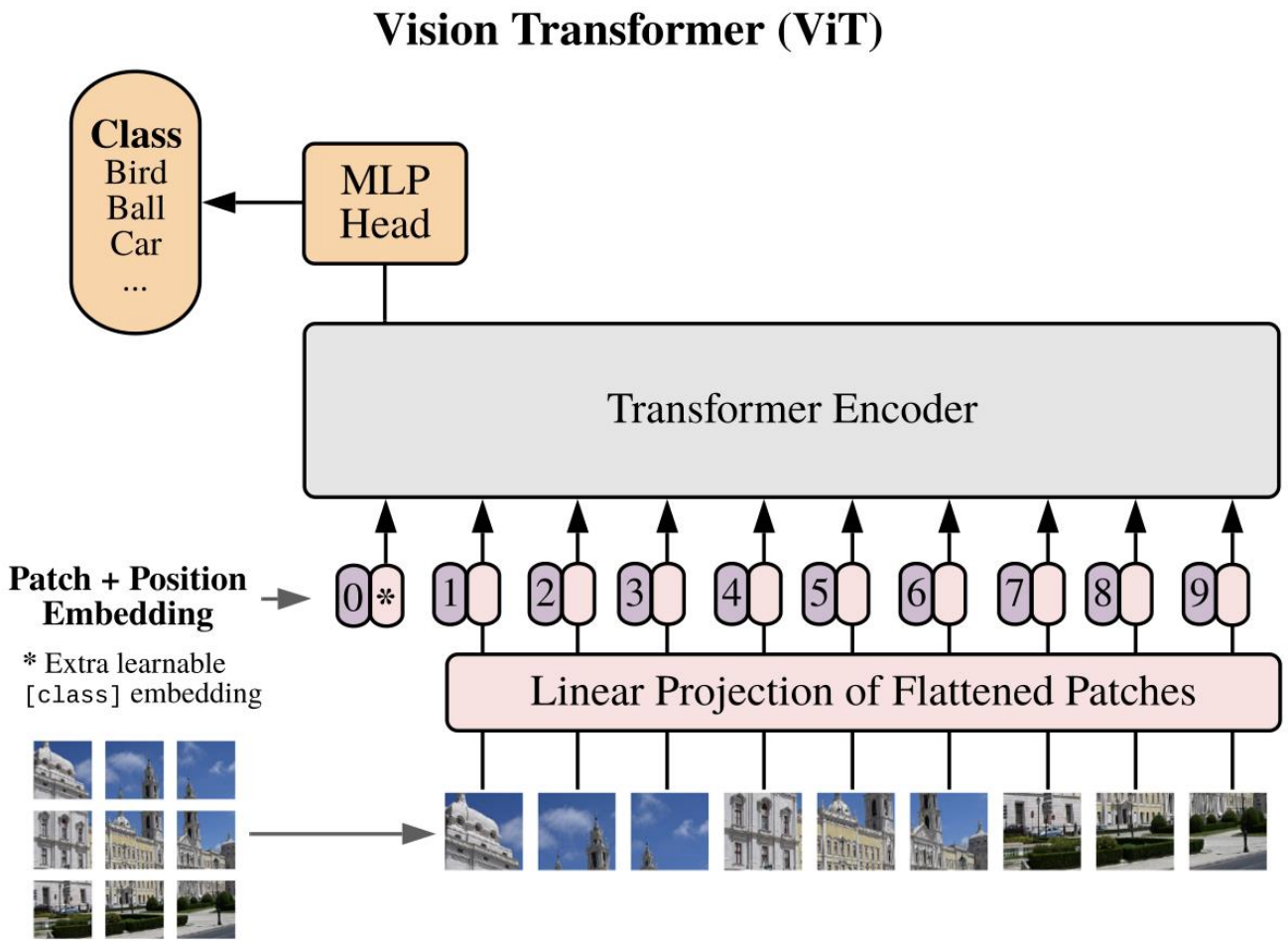
Vision Transformer

- Encoder-Decoder architecture in image processing
- Vision transformer architecture
- Transformer architecture

Vision Transformer

Image processing as a sequence: Vision Transformer

- Each image patch is fed sequentially into an encoder model
- Depending on application, the embedding is either fed into a classifier like MLP (classification) or a decoder (segmentation or dense prediction)

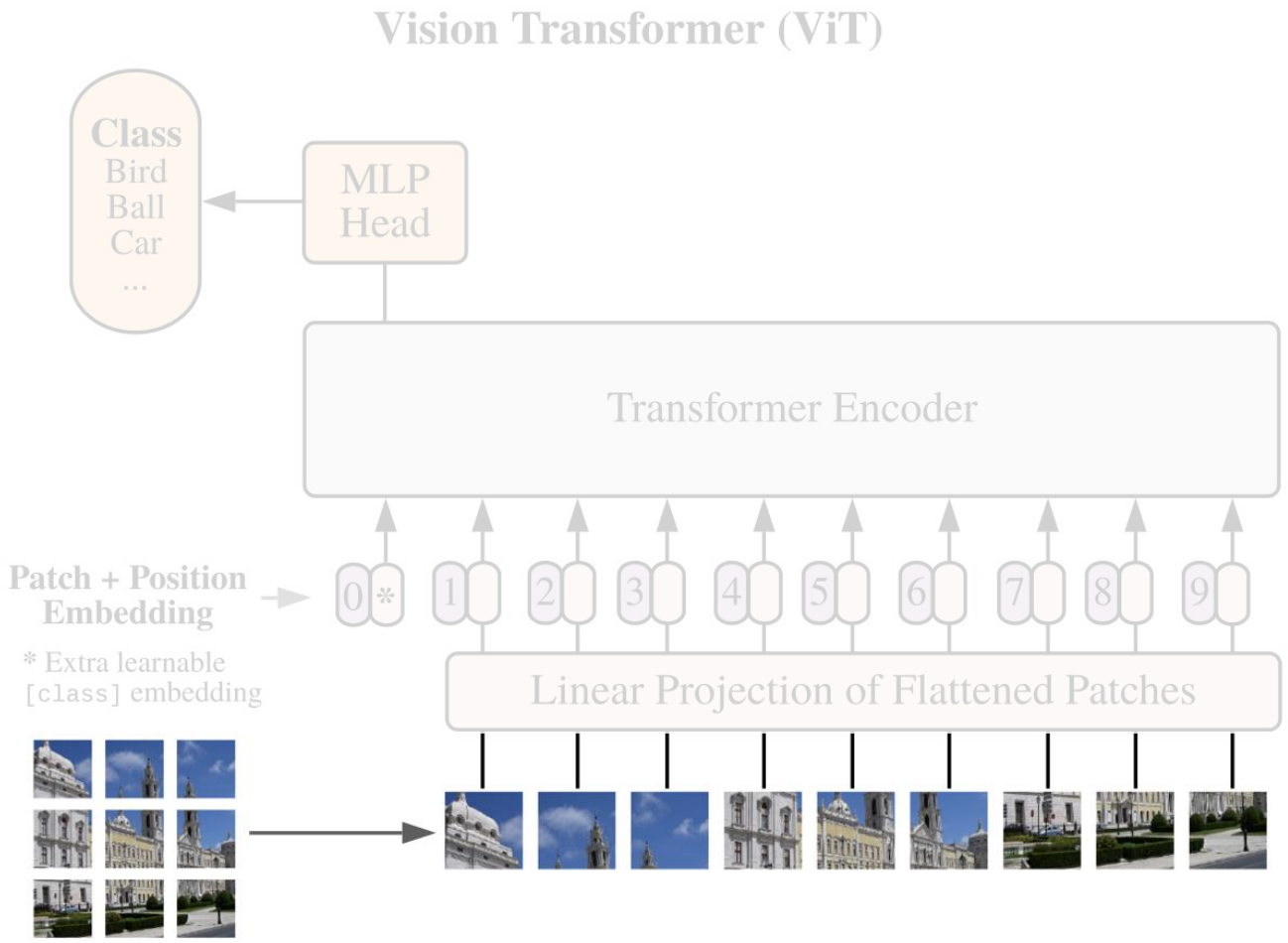


Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Vision Transformer

Step 1: Patchify

- Divide the given image into 16x16 patches
- Each patch can be considered as a ‘word’
- Based on application, the size of the patches can vary

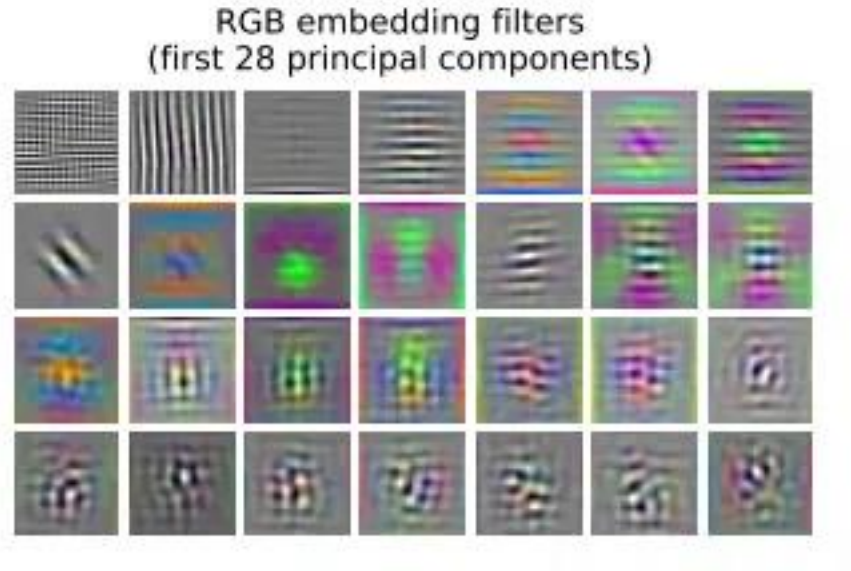


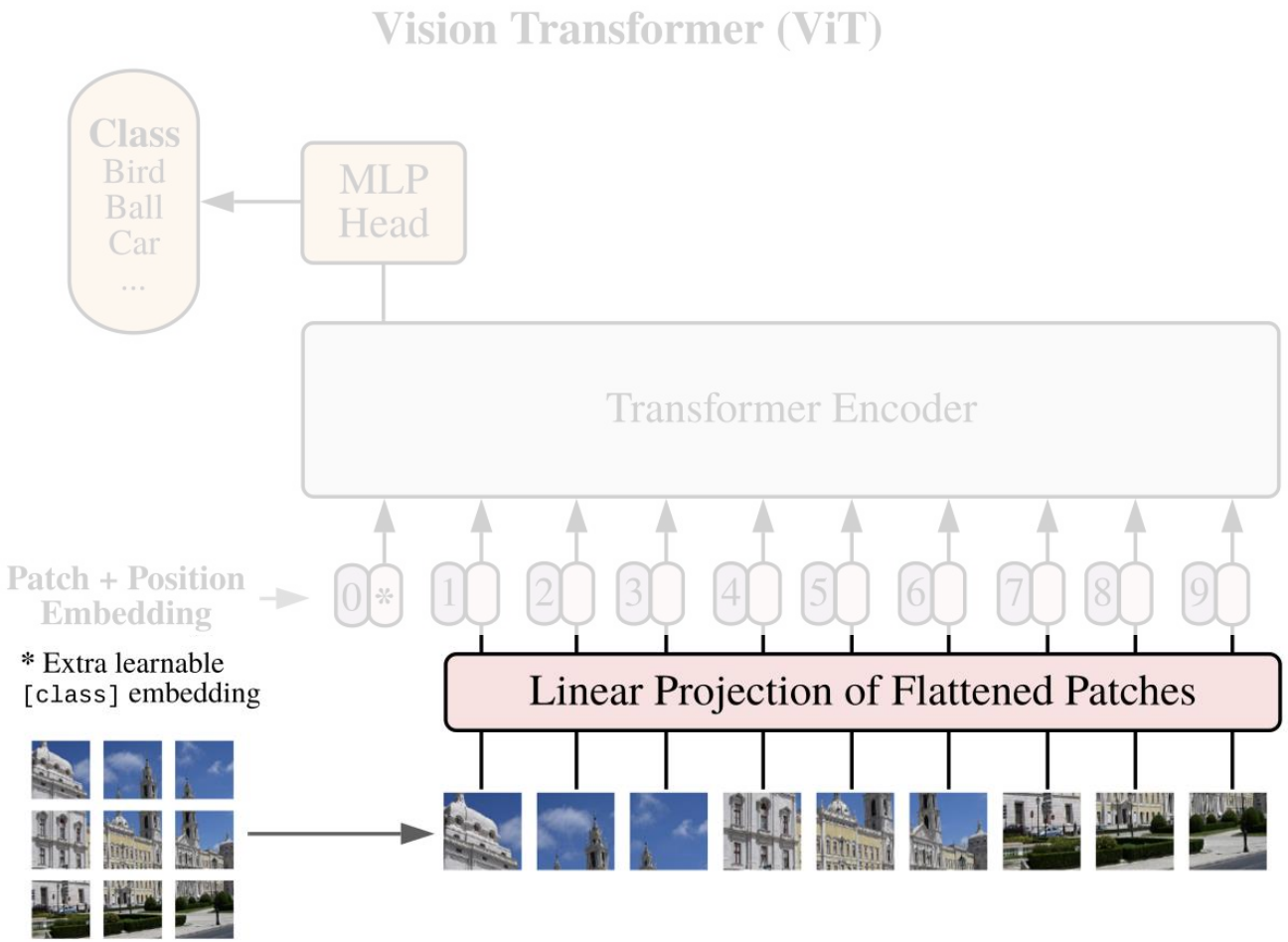
Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Vision Transformer

Step 2: Linear projection

- Each patch is flattened and passed through a set of linear filters



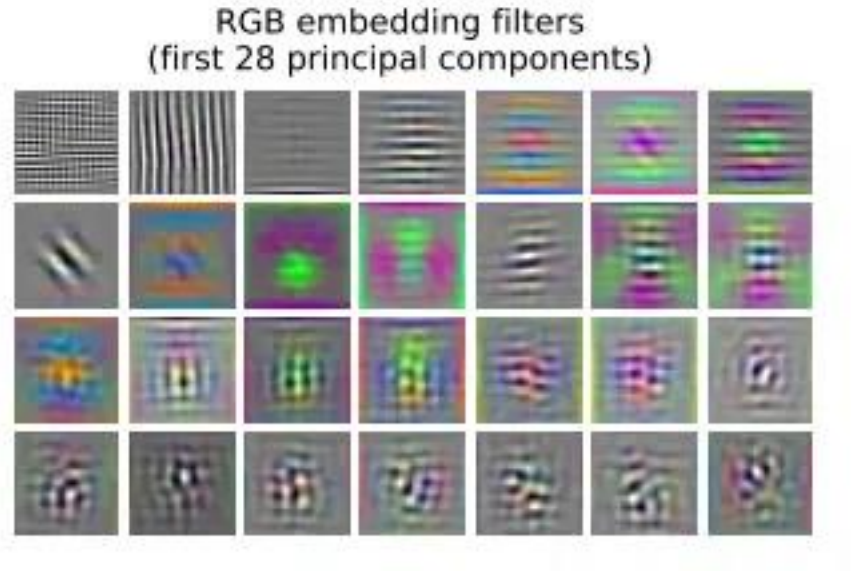


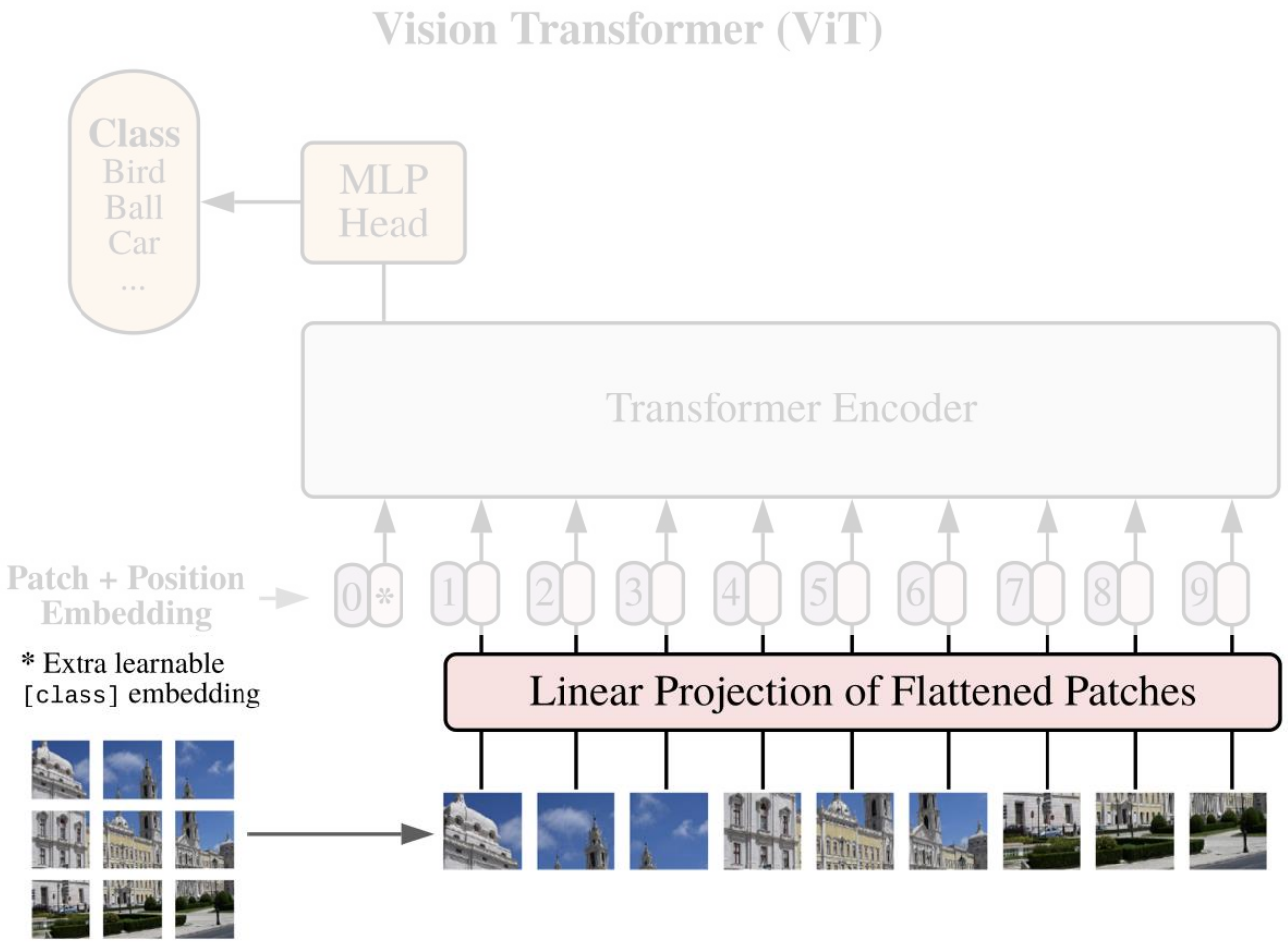
Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Vision Transformer

Step 2: Linear projection

- Each patch is flattened and passed through a set of linear filters



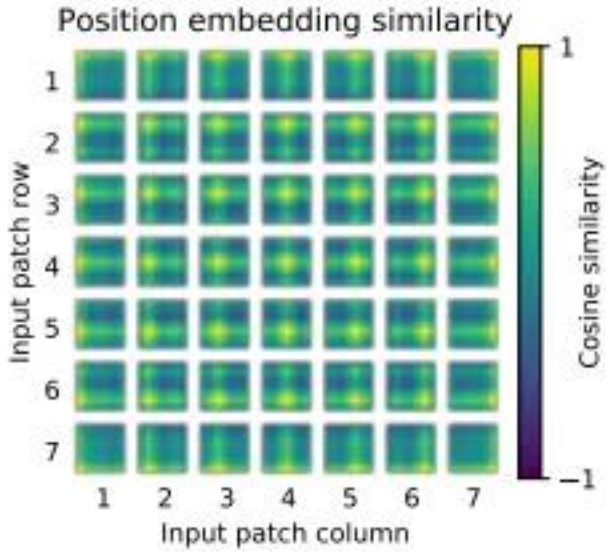


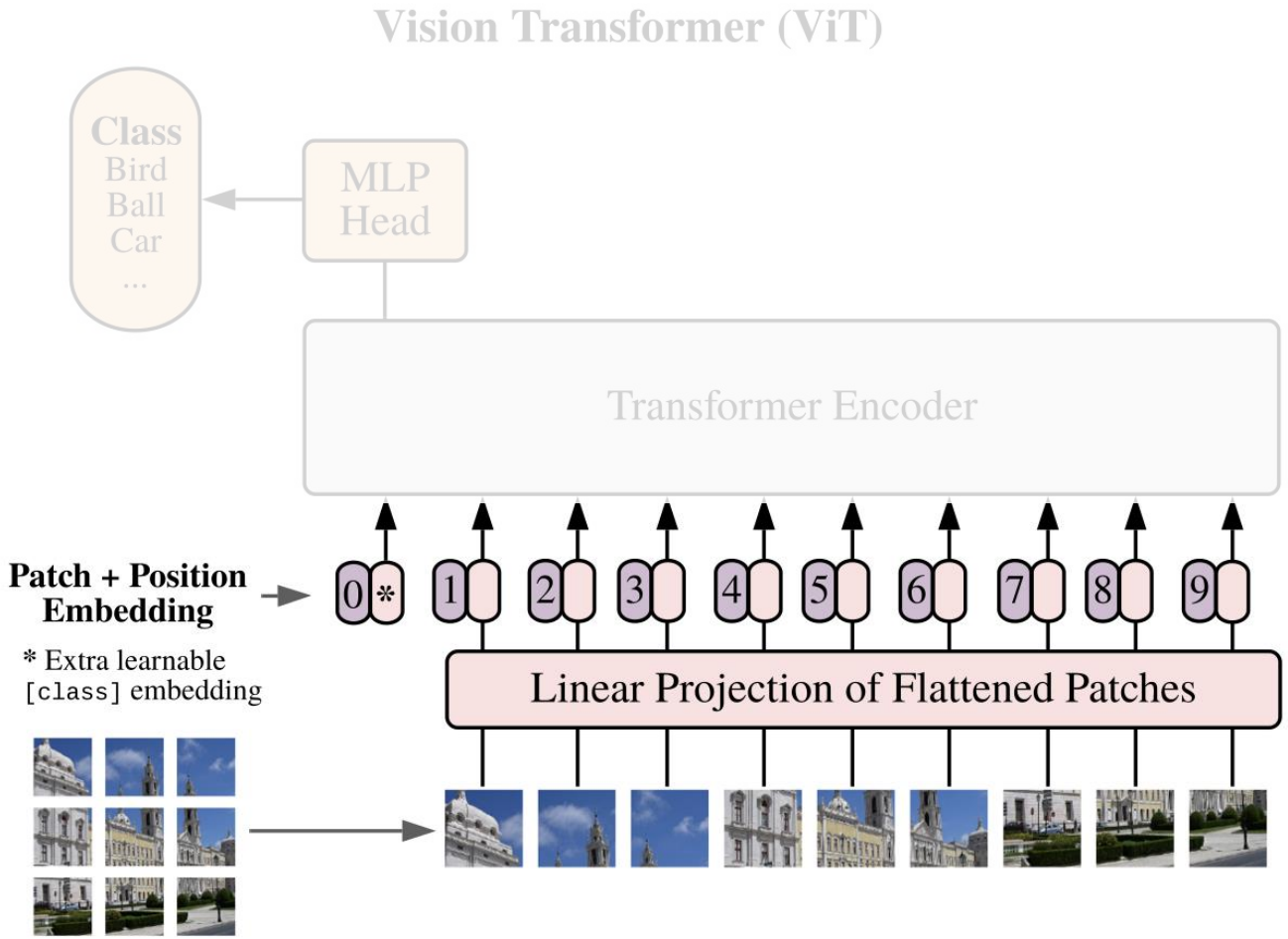
Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Vision Transformer

Step 3: Position embeddings

- Position embeddings are learned vectors with same size as patches
- They are added to patch embeddings



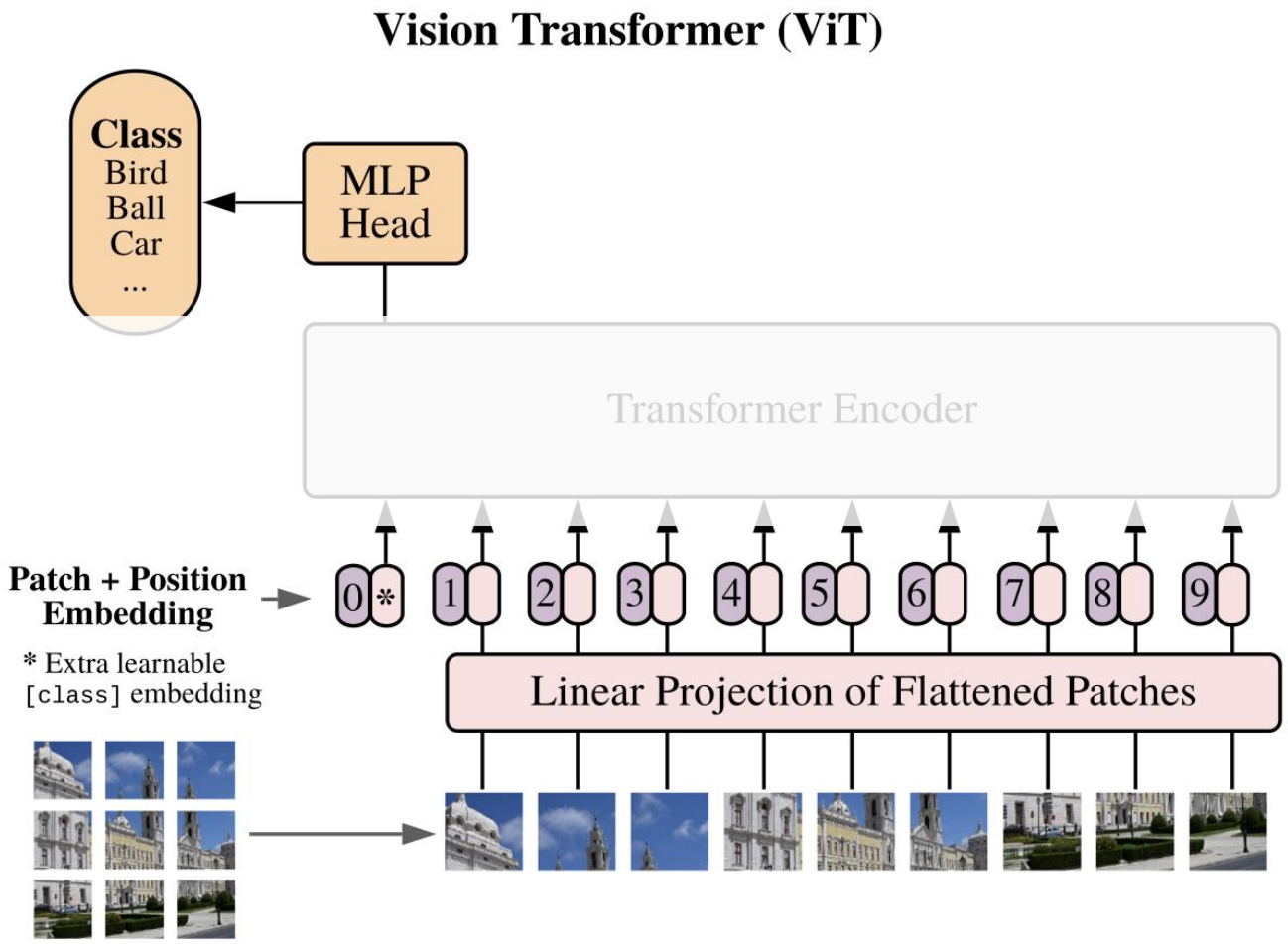


Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Vision Transformer

Step 4: Classification

- The outputs from the encoder model are passed through an MLP (multi layer perceptron)
- MLP is a single fully connected layer followed by ReLU non-linearity

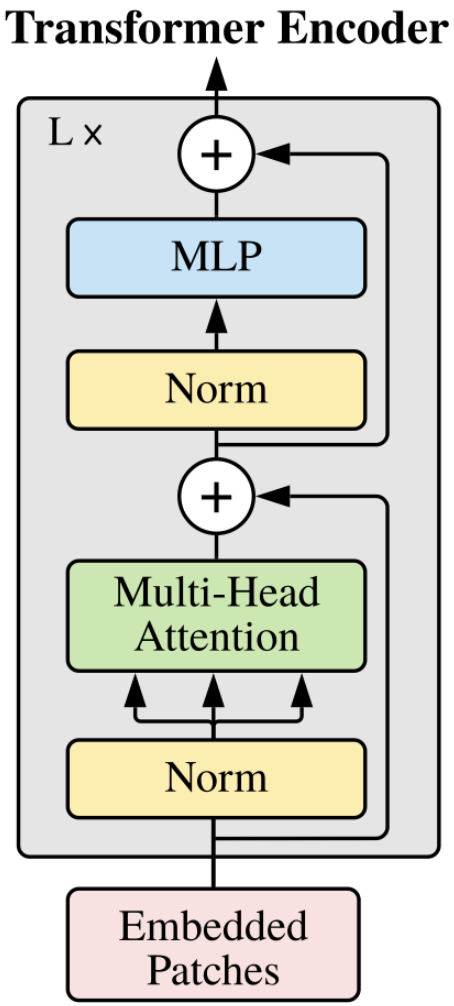


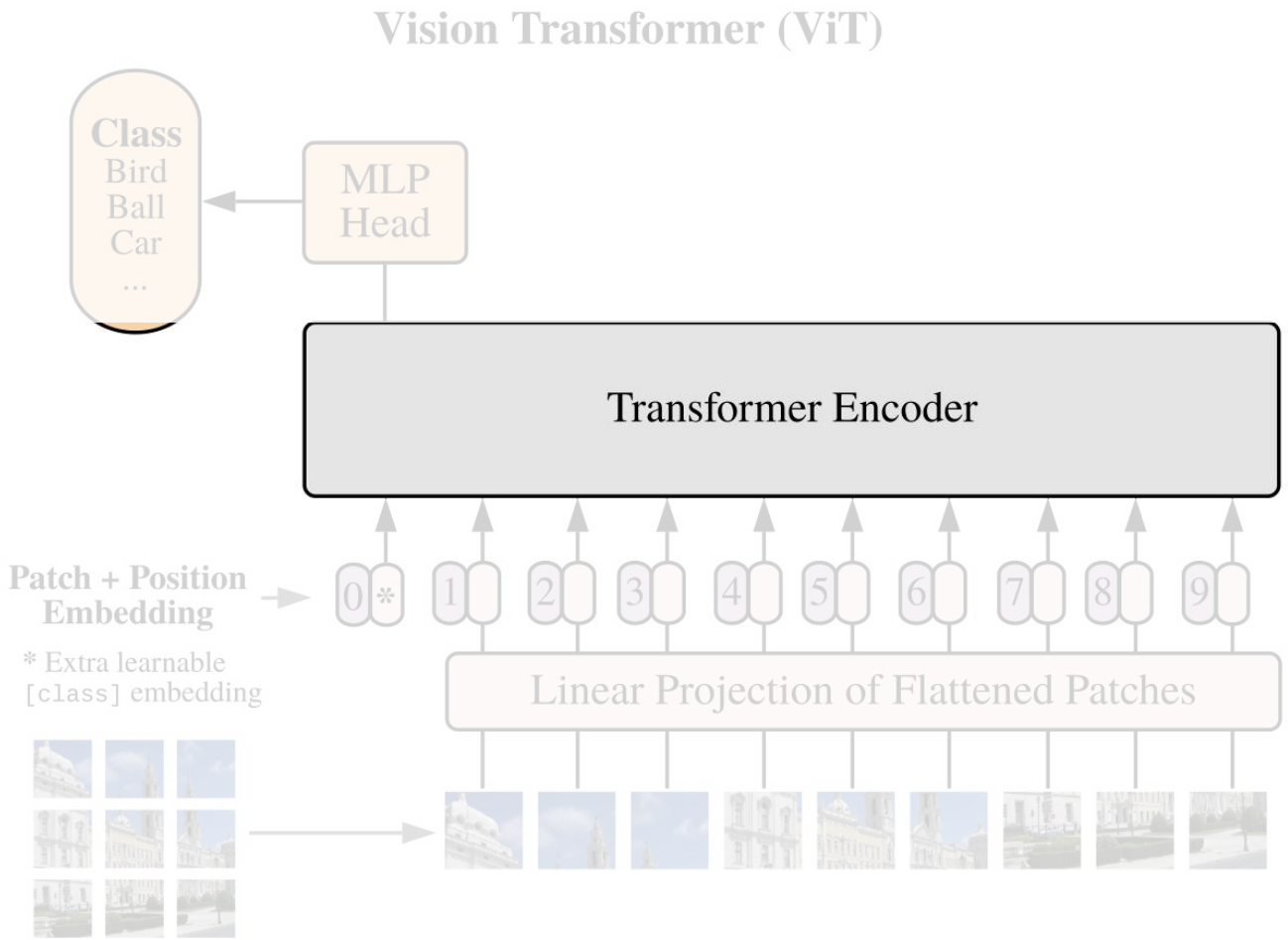
Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Transformer Encoder

Architecture

- Layernorm
- Self-attention
- L Residual blocks





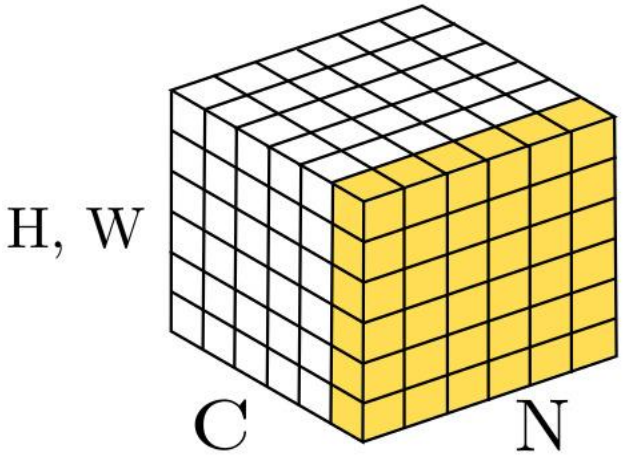
Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

Transformer Encoder

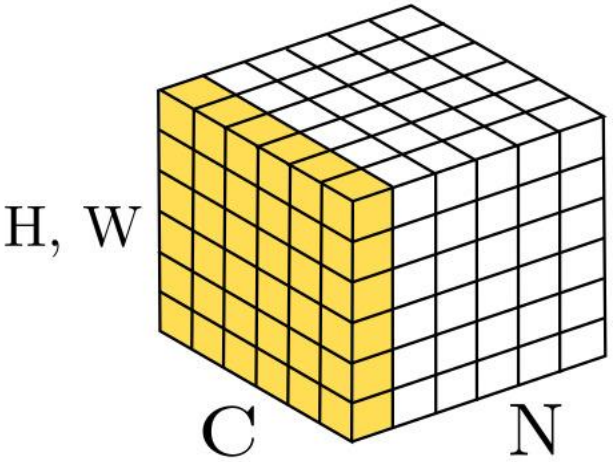
Layer Norm

- Batch normalization:
  - Batch norm is not applicable in sequence modeling since it is unknown during training how many ‘words’ will be processed per batch
- Layer norm
  - Layer norm normalizes channel-wise for a batchsize of N = 1 at inference
  - Similar to batch norm, beta and gamma are learnable parameters
- Advantages over Batch norm:
  - Layer norm can operate with any batchsize
  - Batch norm requires storage of mean and variance during training, to be used at inference

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$



Batch Normalization



Layer Normalization

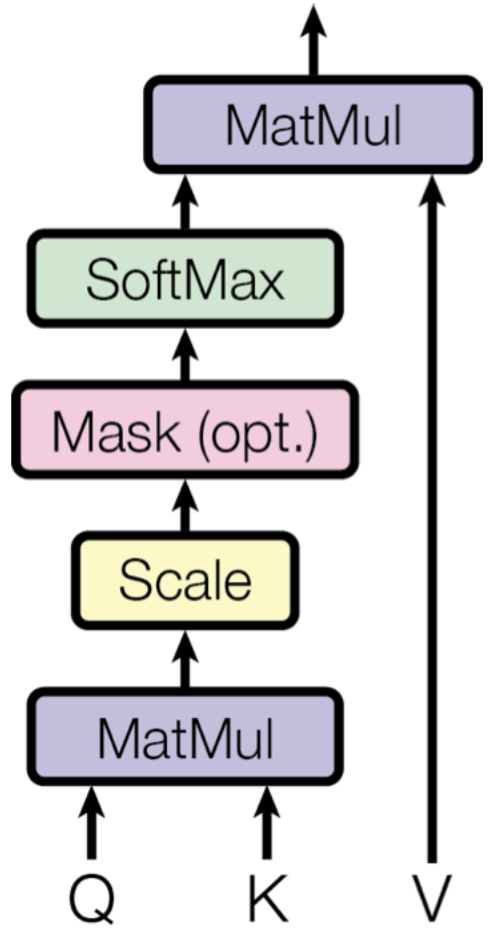
https://wandb.ai/wandb_fc/LayerNorm/reports/Layer-Normalization-in-Pytorch-With-Examples---VmlldzoxMjk5MTk1

Transformer Encoder

Attention

- RNNs
  - Process inputs using U
  - Perform state updates using W
  - Perform output updates using V
  - All three MLPs are learned
- Transformers
  - Q = Query matrix
  - K = Key matrix
  - V = Value matrix

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



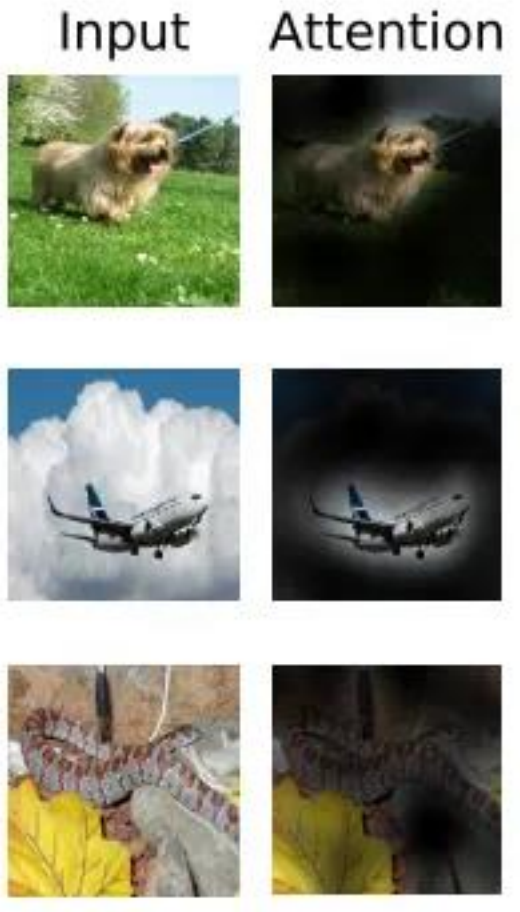
Vaswani, A. "Attention is all you need." *Advances in Neural Information Processing Systems* (2017).

Transformer Encoder

Attention

- Self-attention mechanism
  - Captures relationships between input patch embeddings
  - Computes a weighted sum of all patch embeddings, where the weights are determined by the relevance of each patch to the current one
  - Multi-head attention employs multiple sets of learnable parameters (attention heads) to capture different types of relationships.

- Transformers are trained via self supervison



More on attention and training transformers in Lecture 25

Appendix

Notations

- x_i: a single feature
- $\boldsymbol{x}_i$: feature vector (a data sample)
- $\boldsymbol{x}_{:,i}$: feature vector of all data samples
- $\boldsymbol{X}$: matrix of feature vectors (dataset)
- N: number of data samples
- $\boldsymbol{W}$: weight matrix
- $\boldsymbol{b}$: bias vector
- $\boldsymbol{v}(t)$: first moment at time t
- $\boldsymbol{G}(t)$: second moment at time t
- $\boldsymbol{H}(\boldsymbol{\theta})$: Hessian matrix
- P: number of features in a feature vector
- α: learning rate
- Bold letter/symbol: vector
- Bold capital letters/symbol: matrix