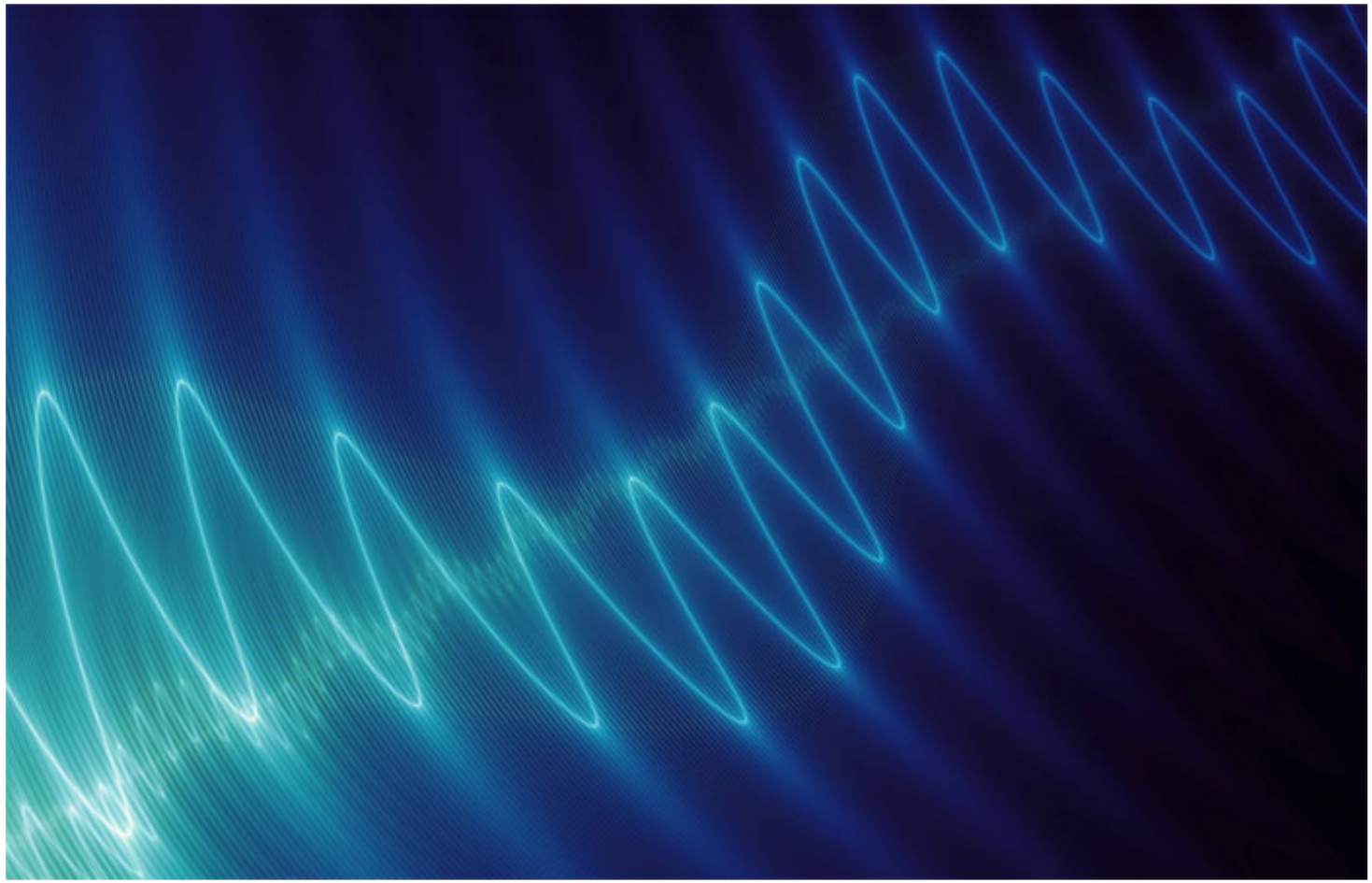
The Practical Machine Learning
PIRE-GEMADARC Lecture Series

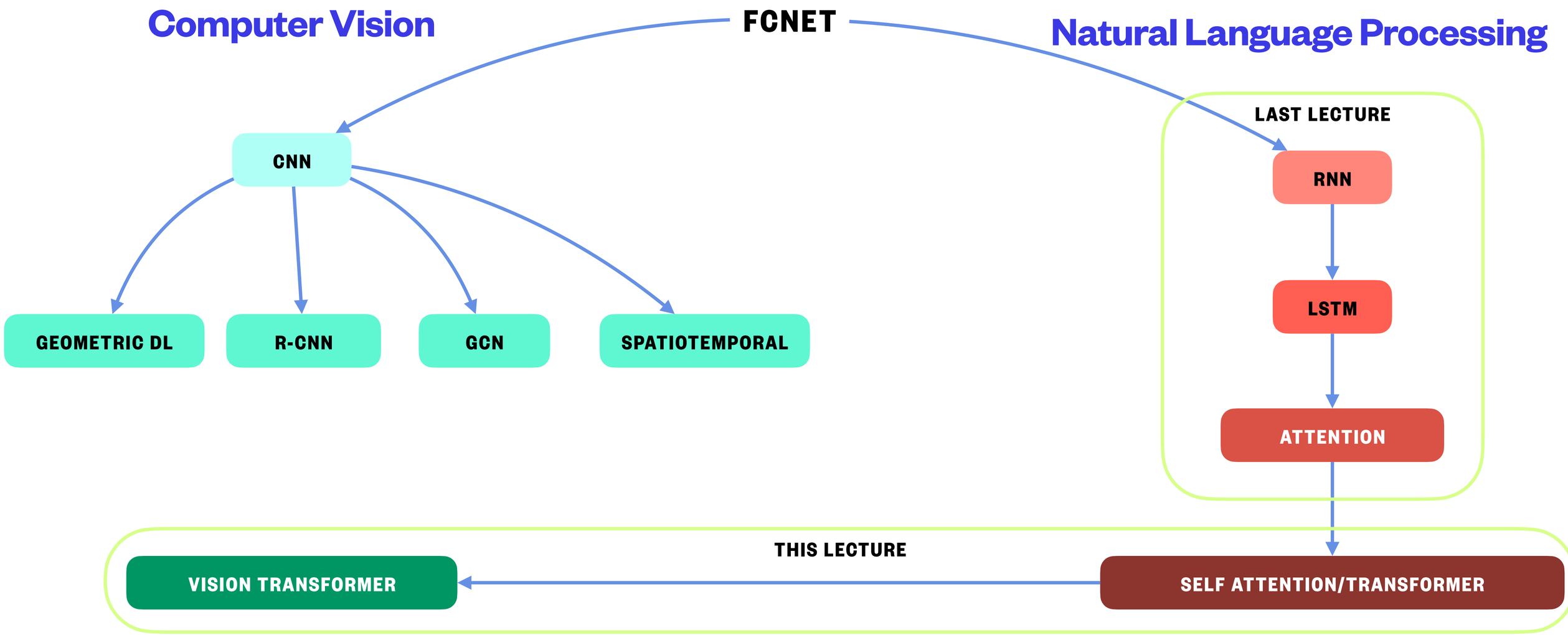
Lecture IV: Time Series II

AOBO LI, 8/6/21

Outline

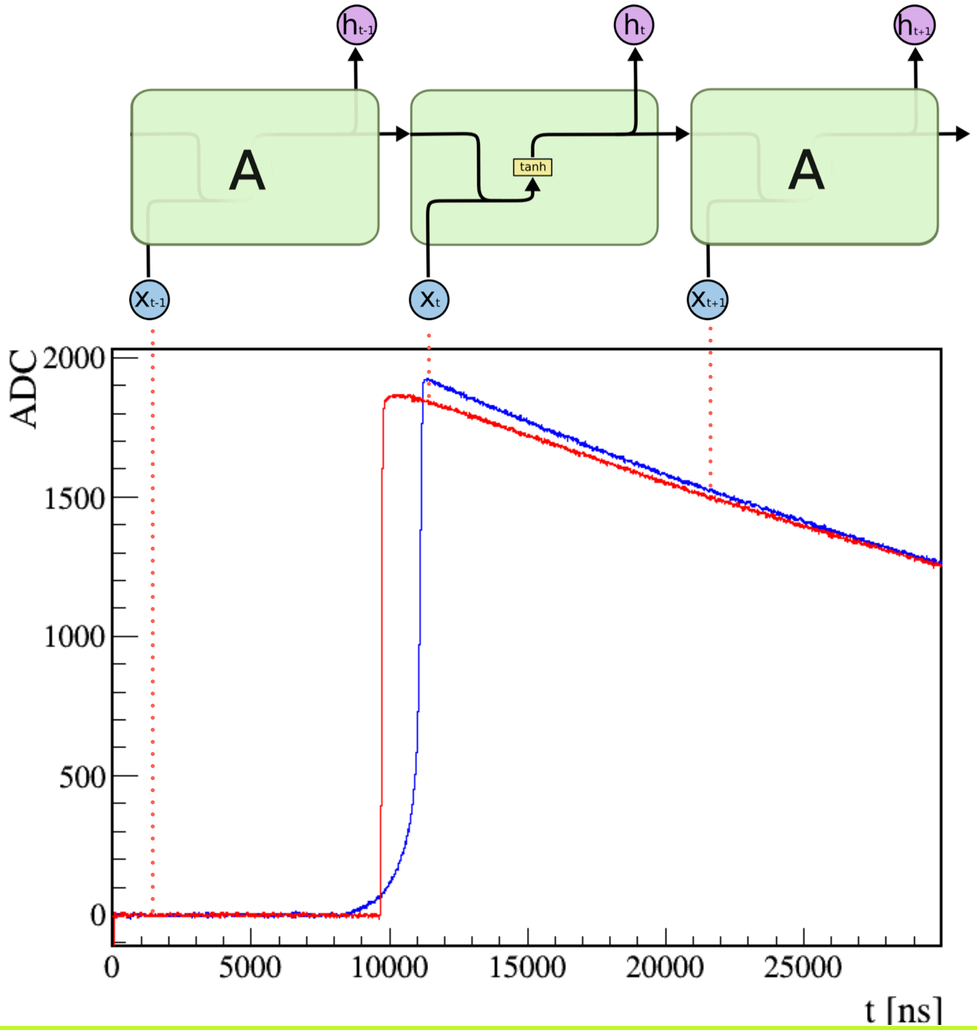
- Quick Review
- MHSA:
  - Attention is all you need
  - Positional Encoding
  - Residual Connection
- Transformer Model:
  - NLP Transformer
  - Vision Transformer




Computer Vision
FCNET
Natural Language Processing
CNN
GEOMETRIC DL
R-CNN
GCN
SPATIOTEMPORAL
LAST LECTURE
RNN
LSTM
ATTENTION
THIS LECTURE
VISION TRANSFORMER
SELF ATTENTION/TRANSFORMER

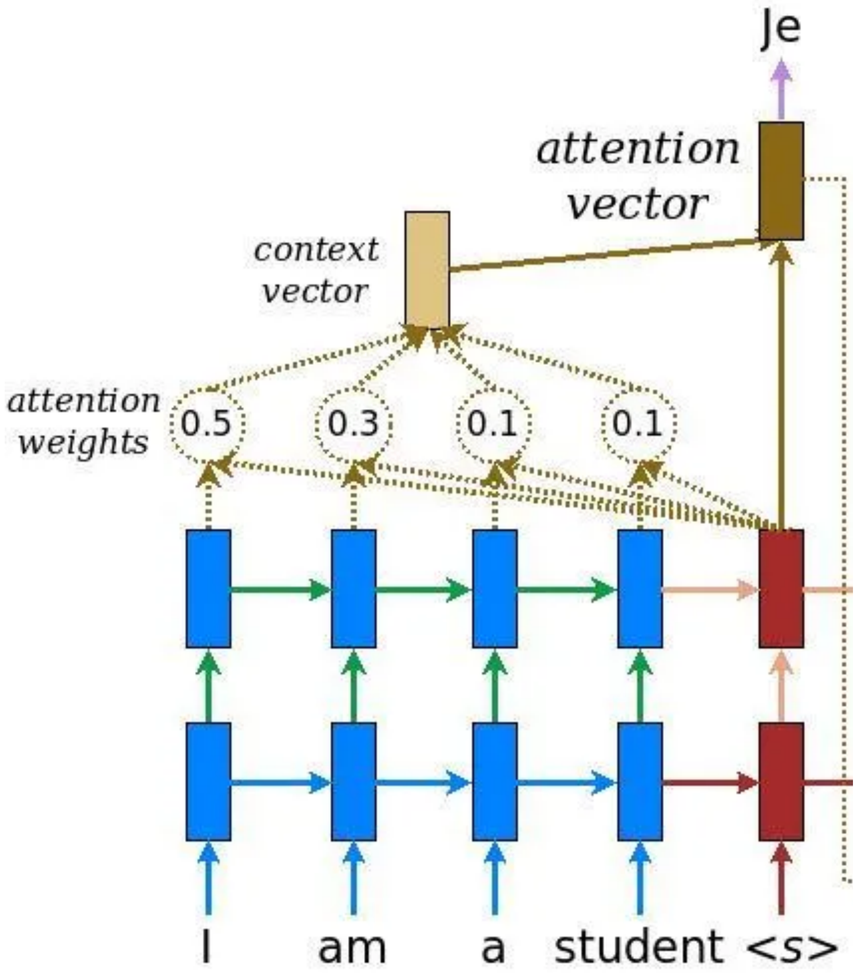
Quick Review: RNN

- When the recurrent unit goes to the current input :
  - Information from previous hidden state and current input are analyzed together
  - The output is produced by a hyperbolic tangent activation function
  - output serves as both the intermediate hidden state output and the hidden state to the next input
- Iteration goes until the end of series, and the last hidden state output is the output of network.

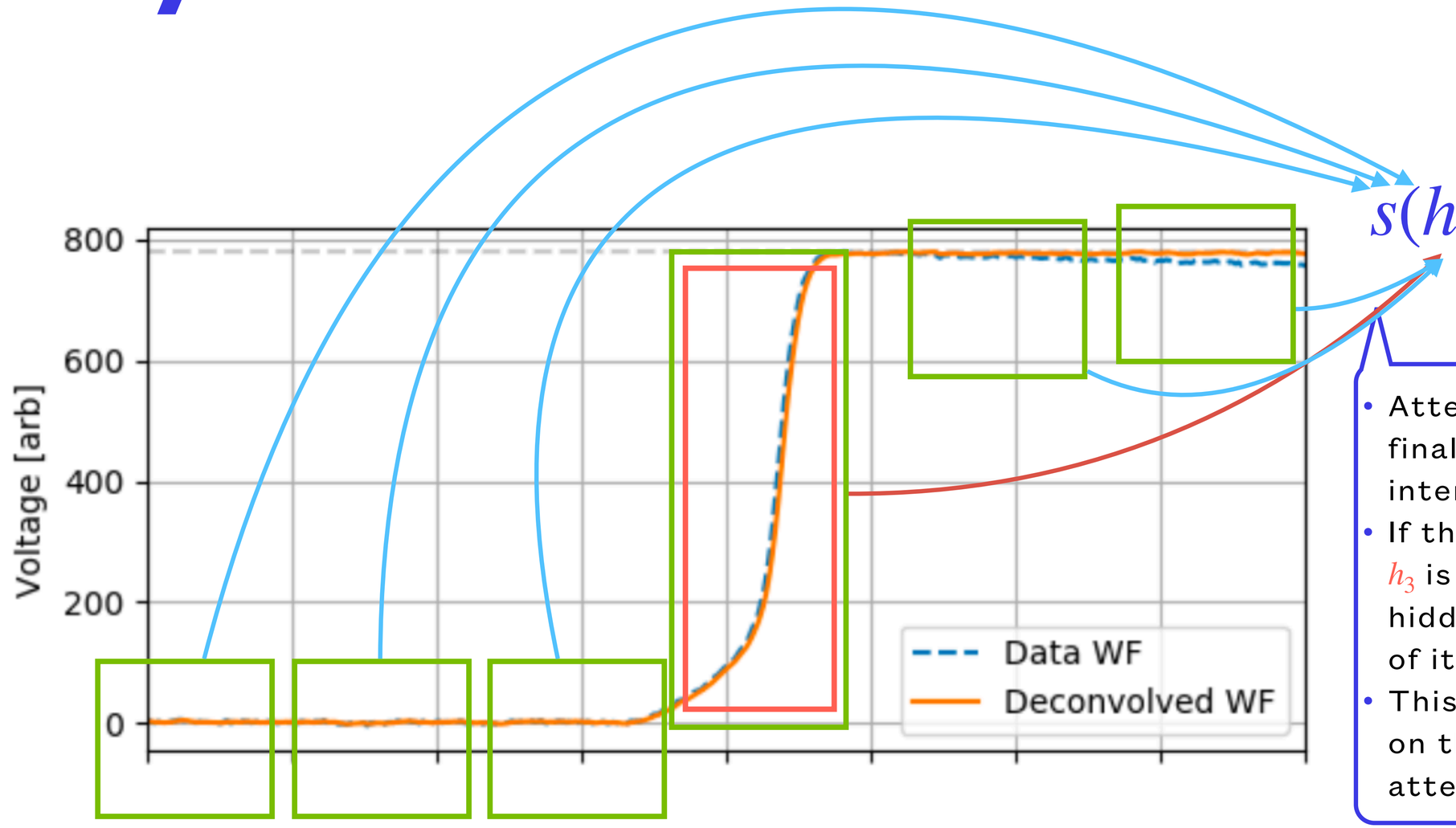


Quick Review: Attention

- Traditional RNN/LSTM utilizes only the last hidden state output is used
- Attention Mechanism weights all the intermediate hidden state outputs to form a context vector
  - Calculate a scalar quantity called attention score for each intermediate hidden state output
  - Perform weighted sum of attention score and all intermediate hidden state outputs to produce a context vector
  - Context vector and last hidden state output is concatenated together to form the layer output



Why Attention Mechanism Works?



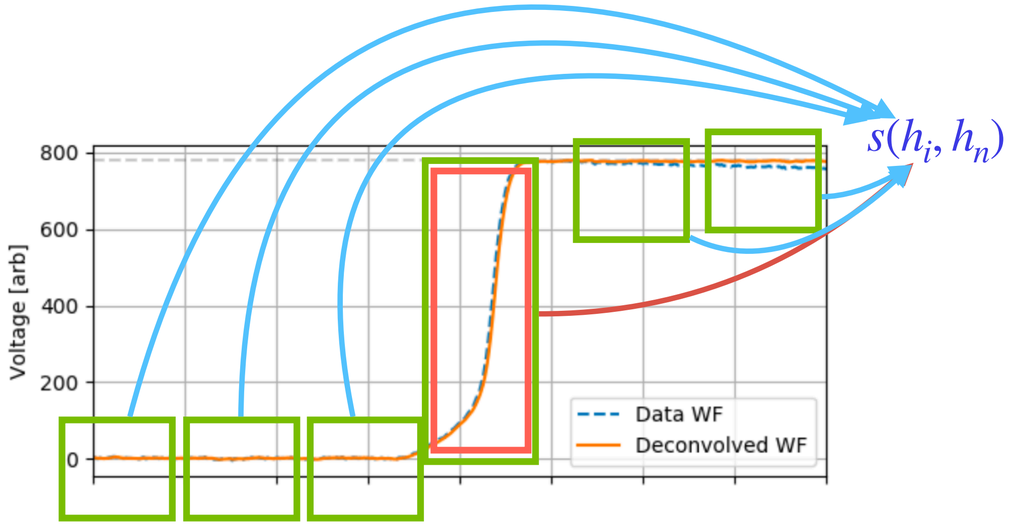
$s(h_i, h_n)$

- Attention Mechanism allows the final hidden state to “talk” to each intermediate hidden states
- If the feature in one hidden state h_3 is important, then the final hidden state should preserve most of its info
- This lead to high attention score on this state $s(h_3, h_n)$ and low attention score on other states

Revisit Attention

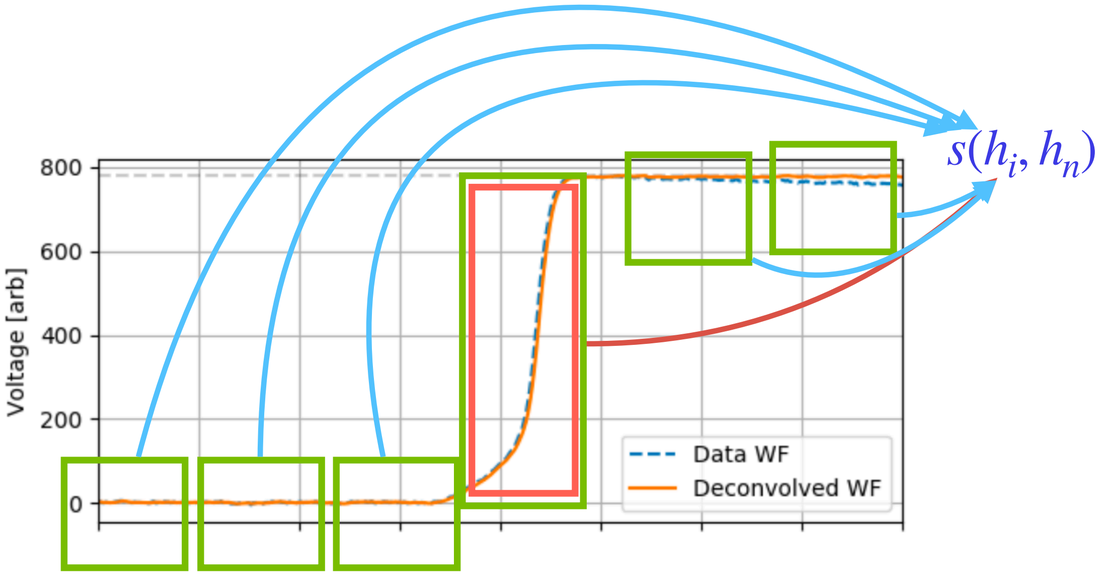
- Attention mechanism can be viewed in a QKV formulation
  - Q = Query = last hidden state output
  - K = Key = intermediate hidden state output
  - V = Values = intermediate hidden state output

- Attention score is obtained by query a specific key:
  - This is equivalent of the calculation $f(h_i, h_n)$
  - That is, the last hidden state output "talks" to each intermediate hidden state output simultaneously
  - If a given intermediate hidden state output is important, then the attention score is also larger
- The output is obtained by multiplying attention score to values:
  - $\text{Output} = \text{Softmax}(s(Q, K)) * V$



Revisit Attention

- RNN has many disadvantages:
  - Lack of long range correlation
  - Serial, thus not parallelizable
- The NLP community has move in the direction of abandoning RNN
- But without RNN, both last hidden state output and intermediate hidden state output will be unavailable
- How to deal with that?



Attention is All You Need

[PDF] **Attention** is **all you need**

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - papers.nips.cc

The dominant sequence transduction models are based on complex recurrent orconvolutional neural networks in an encoder and decoder configuration. The best performing such models also connect the encoder and decoder through an attentionm …

☆ 99 Cited by 23870 Related articles All 28 versions »

[PDF] nips.cc

- Probably one of the most important paper in the history of NLP
- Now is the basis of many important NLP models such as BERT, GPT-3 etc
- Also call the trend of naming machine learning paper with "…is all you need"
- A comprehensive review of this paper:
  - https://jalammar.github.io/illustrated-transformer/

Rethinking few-shot image classification: a good embedding is **all you need**?

Y Tian, Y Wang, D Krishnan, JB Tenenbaum… - Computer Vision–ECCV …, 2020 - Springer

The focus of recent meta-learning research has been on the development of learning algorithms that can quickly adapt to test time tasks with limited data and low computational cost. Few-shot learning is widely used as one of the standard benchmarks in meta-learning …

☆ 99 Cited by 119 Related articles All 7 versions »

Image augmentation is **all you need**: Regularizing deep reinforcement learning from pixels

I Kostrikov, D Yarats, R Fergus - arXiv preprint arXiv:2004.13649, 2020 - arxiv.org

We propose a simple data augmentation technique that can be applied to standard model-free reinforcement learning algorithms, enabling robust learning directly from pixels without the **need** for auxiliary losses or pre-training. The approach leverages input perturbations …

☆ 99 Cited by 88 Related articles All 6 versions »

All you need is dag

I Keidar, E Kokoris-Kogias, O Naor… - arXiv preprint arXiv …, 2021 - arxiv.org

We present DAG-Rider, the first asynchronous Byzantine Atomic Broadcast protocol that achieves optimal resilience, optimal amortized communication complexity, and optimal time complexity. DAG-Rider is post-quantum safe and ensures that **all** messages proposed by …

☆ 99 Cited by 3 Related articles All 2 versions »

Channel attention is **all you need** for video frame interpolation

M Choi, H Kim, B Han, N Xu, KM Lee - Proceedings of the AAAI …, 2020 - ojs.aaai.org

Prevailing video frame interpolation techniques rely heavily on optical flow estimation and require additional model complexity and computational cost; it is also susceptible to error propagation in challenging scenarios with large motion and heavy occlusion. To alleviate …

☆ 99 Cited by 22 Related articles All 3 versions »

All you need is boundary: Toward arbitrary-shaped text spotting

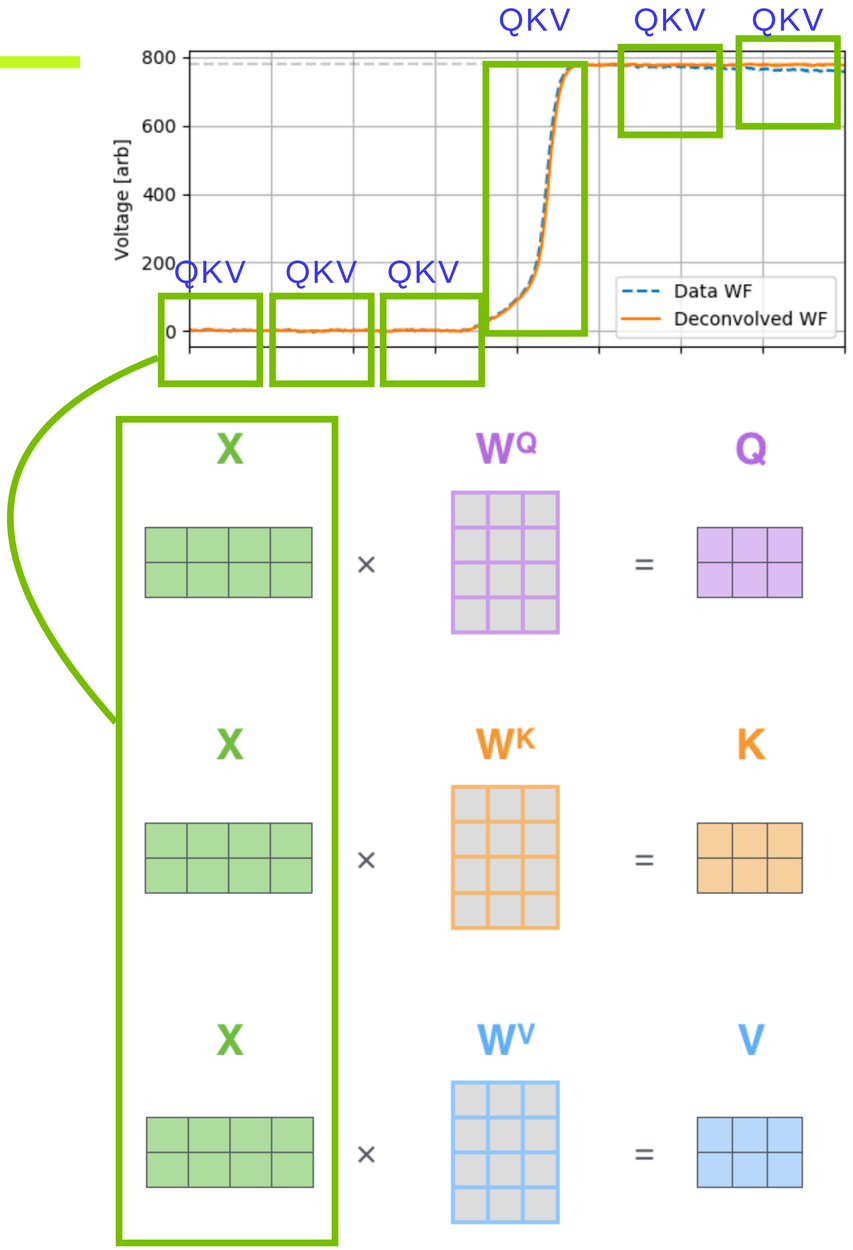
H Wang, P Lu, H Zhang, M Yang, X Bai, Y Xu… - Proceedings of the …, 2020 - ojs.aaai.org

Recently, end-to-end text spotting that aims to detect and recognize text from cluttered images simultaneously has received particularly growing interest in computer vision. Different from the existing approaches that formulate text detection as bounding box …

☆ 99 Cited by 28 Related articles All 4 versions »

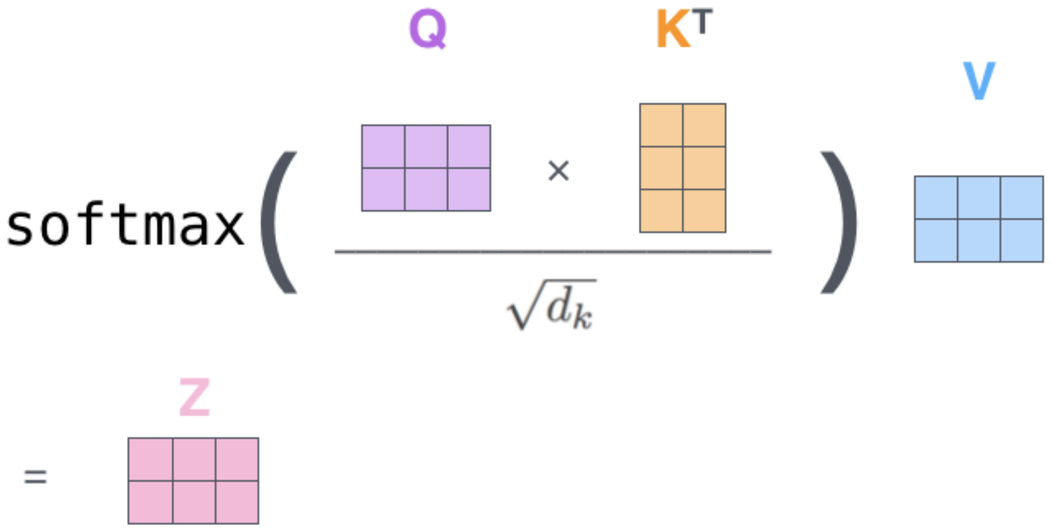
Self Attention

- Without RNN, we will have to produce the Queries, Keys and Values ourself
- This can be easily done by multiplying 3 independent kernel matrices to the input scalar/vectors
  - W^Q, W^K and W^V are the kernels of self attention layer, their values change during the training to fulfill the assigned task
  - Kernel is share by all time indices
- This operation is performed to every single time index of the time series data:
  - Now for each time index, we have 3 vector representations: Q, K and V



Self Attention

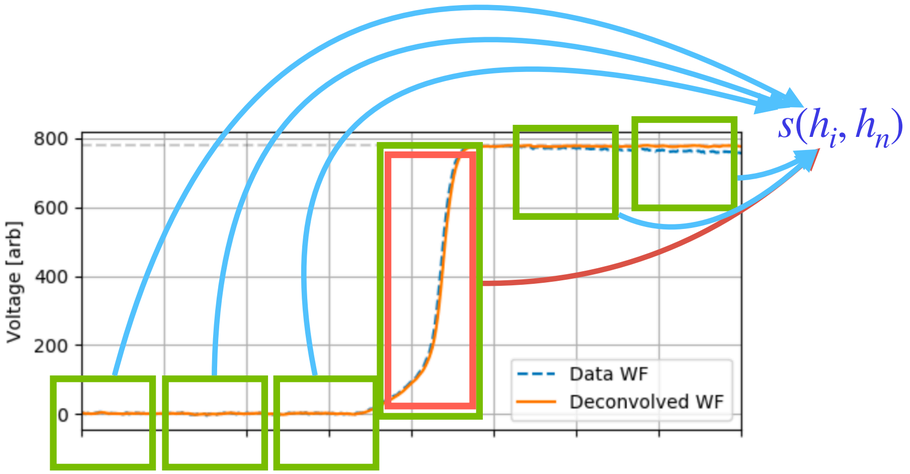
- Now for each time index i, we perform this calculation:
  - $z_j = \text{Softmax}(\frac{Q_i \times K_j^T}{\sqrt{d_k}})V_j$
  - Q_i is the query vector of current time index
  - z_j is evaluated for every time indices, including the current time index i
  - d_k is the dimension of data in this time slices, the division of $\sqrt{d_k}$ is just for numerical stability
- Note that the evaluation of z_j takes a very similar form as the traditional attention mechanism:
  - $\text{Output} = \text{Softmax}(s(Q, K)) * V$



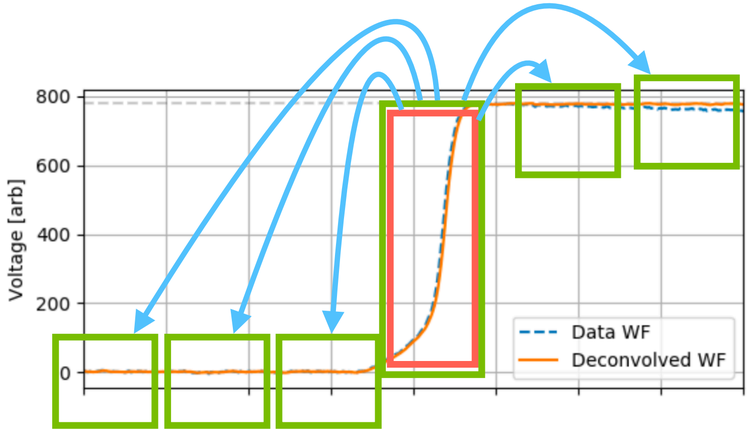
Self Attention

- In attention mechanism, the last hidden state output queries each intermediate hidden state output to produce output
  - The query is performed only once
- In self attention, each time index i queries all time indices to produce output
  - The query is performed n times, where n is the length of time series
- In attention mechanism, the attention score is evaluated using certain pre-defined score function $s(Q, K)$
- In self attention, the attention score is directly obtained by multiplying Q and K together
  - Why don't we insert the additional weight kernel as we did for RNN attention?

Attention Mechanism

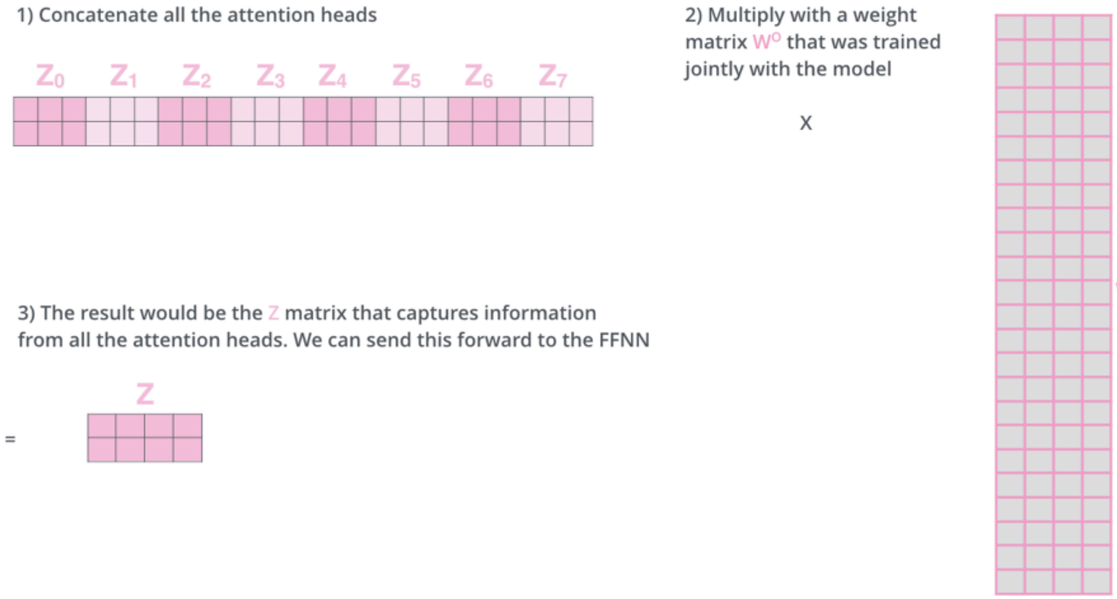


Self Attention



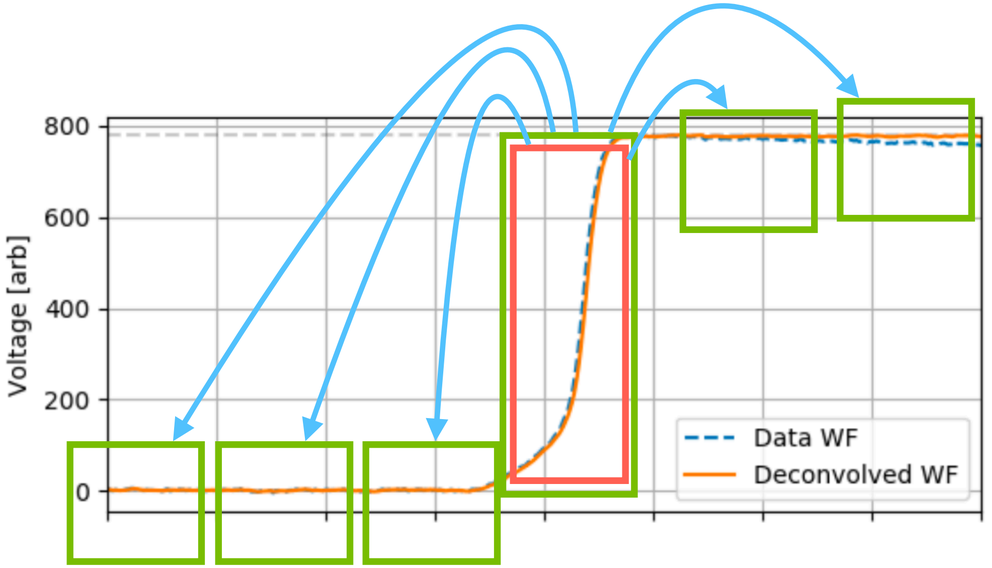
Multi-Head Self Attention

- Instead of having only one group of (W^Q, W^K, W^V), we can have m groups of them to look at the time series simultaneously
- Each group of (W^Q, W^K, W^V) is called a head
  - Head is like a probe to the time series data
- If we have 7 head, the output will contains 7 zs
- The 7 zs are concatenated together, then multiplied by a overall weight matrix to produce the final output



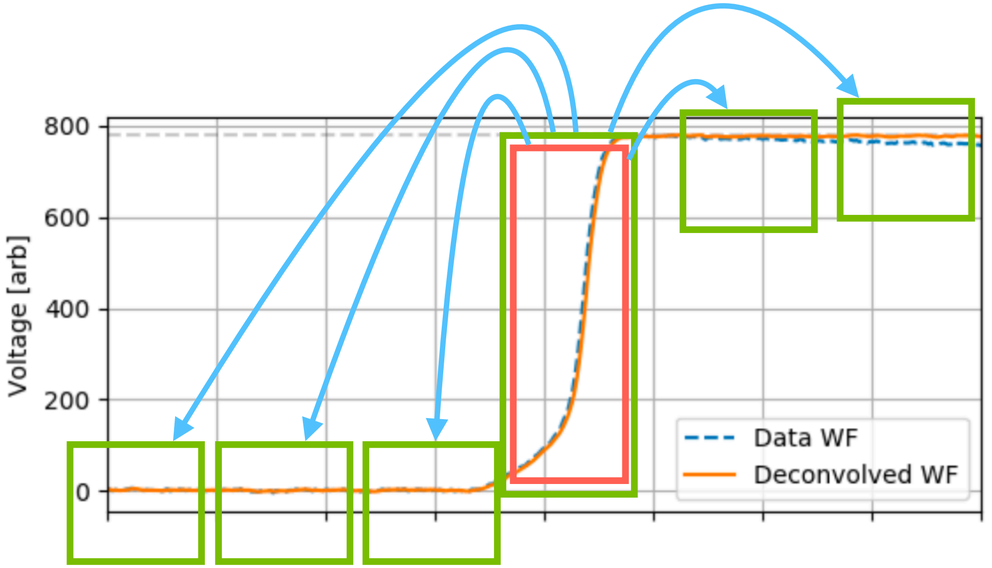
Why is MHSA better?

- MHSA exhibit global correlation:
  - Every time index can now “talk” to every other time indices to obtain their correlation
- MHSA is simple in terms of computation:
  - Everything is matrix multiplication and some simple linear algebra
  - No serial operation, meaning that we can easily parallelize the training process
- MHSA has been used on many state-of-the-art language models with extremely large training data size:
  - BERT, GPT-3 etc.



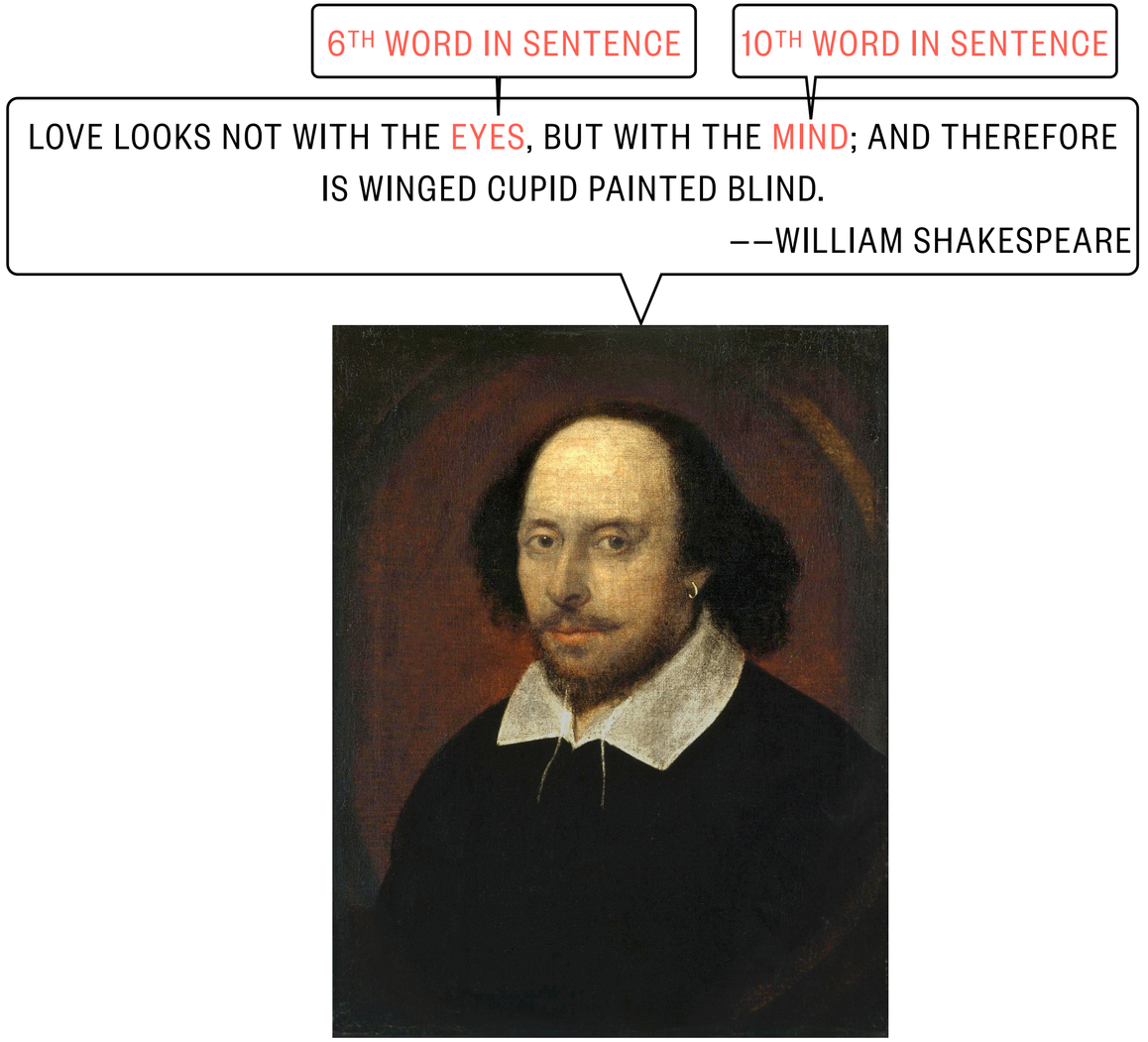
Problem with MHSA

- MHSA resolves all problems of RNN, but now it has another disadvantage: the order of time series is not properly encoded
- RNN enforce order information through the serial recurrent operation. Without that, MHSA won't be able to probe the positional information
- Positional Encoding is an important technique in the MHSA paper to resolve this issues
  - If you are planning to use CNN to analyze time series, then you should also add positional encoding to the data



Positional Encoding

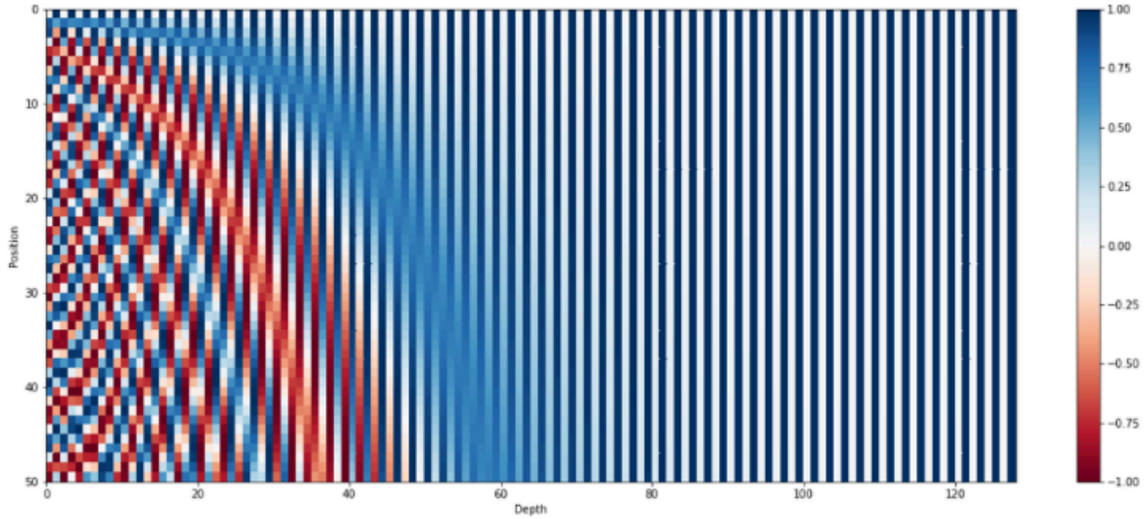
- Positional Encoding need to satisfy two features:
  - Absolute Position: *Eyes* is the 6th word in the sentence, *mind* is the 10th word in the sentence
  - Relative Position: "*mind*" is 4 words behind "*eyes*", just like "*the*" is 4 words behind "*love*"
- The Attention is All You Need paper uses a smart way to do positional encoding:
  - Position of each time index t_i is encoded using a d-dimensional vector
  - The d-dimensional vector contains alternating sine and cosine functions with different frequencies



Positional Encoding

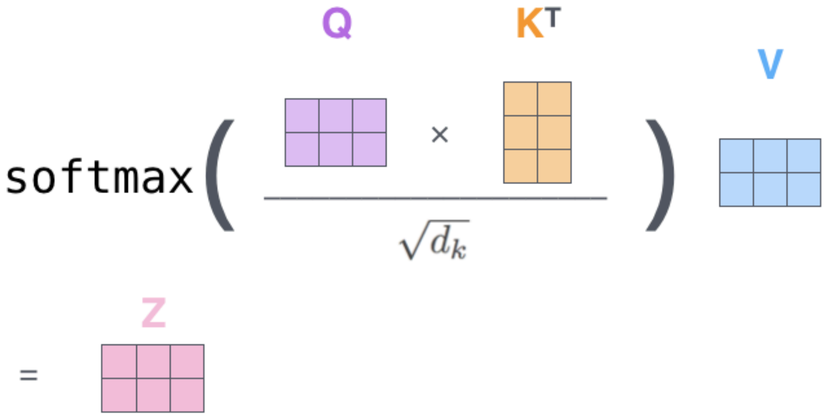
- The detail of the PE algorithm is quite mathematically heavy, so I decide not to show it here
- This article gives a very good simplified examples about why it works:
  - https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
- In a binary number representation:
  - Looking in vertical directions, the bits flip at different frequencies
  - Looking in the horizontal direction, each line is the binary representation of an integer number
- Converting this into float point space representation, we get the positional encoding in sin/cos
- This positional encoding vector is directly added upon the time series at the input of the network

0 :	0	0	0	0	8 :	1	0	0	0
1 :	0	0	0	1	9 :	1	0	0	1
2 :	0	0	1	0	10 :	1	0	1	0
3 :	0	0	1	1	11 :	1	0	1	1
4 :	0	1	0	0	12 :	1	1	0	0
5 :	0	1	0	1	13 :	1	1	0	1
6 :	0	1	1	0	14 :	1	1	1	0
7 :	0	1	1	1	15 :	1	1	1	1



Learned Positional Encoding

- The Zen of machine learning is that, you learn everything that cannot be figured out by hand
  - Learn = set up a trainable kernel, and let it figure out its value during training
- Since PE is tricky for MHSA model, we can also ask the network to learn the positional encoding by itself
  - This approach is adopted in BERT
- Modify the original MHSA, so that it does not only query the key, but also query the positional encoding:
  - $z_j = \text{Softmax}(\frac{Q_i \times K_j^T + \boxed{Q_i \times R}}{\sqrt{d_k}}) \times V_j$
  - R is the trainable positional encoding



Residual Connection

- Residual connection is a simple yet powerful technique to avoid gradient vanishing
  - If a neural network is too deep, the gradient might not be able to propagate into some parameters, and thus those parameters will never be updated
- Gradient vanishing is a natural consequence of derivative operations:
  - It's not too surprising to see $\frac{dF(x)}{dx} = 0$ for a arbitrary function $F(x)$
  - However, if you add the input x to the function $F(x)$, then $\frac{d(F(x)+x)}{dx} \neq 0$
- Residual connections also allow information to flow directly from input to output
  - PE is added to x, so it can easily get lost in deep MHSA, but residual connection allows the PE to flow to the end

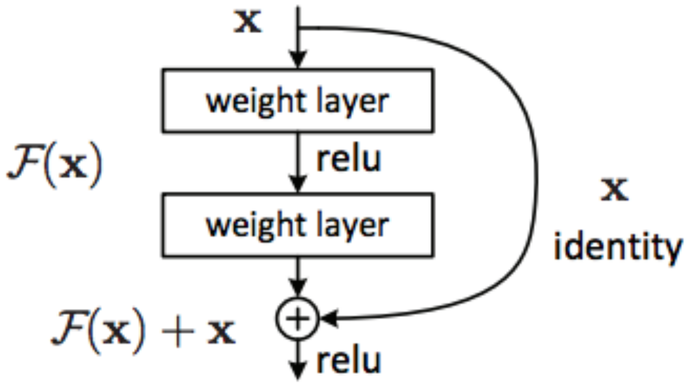
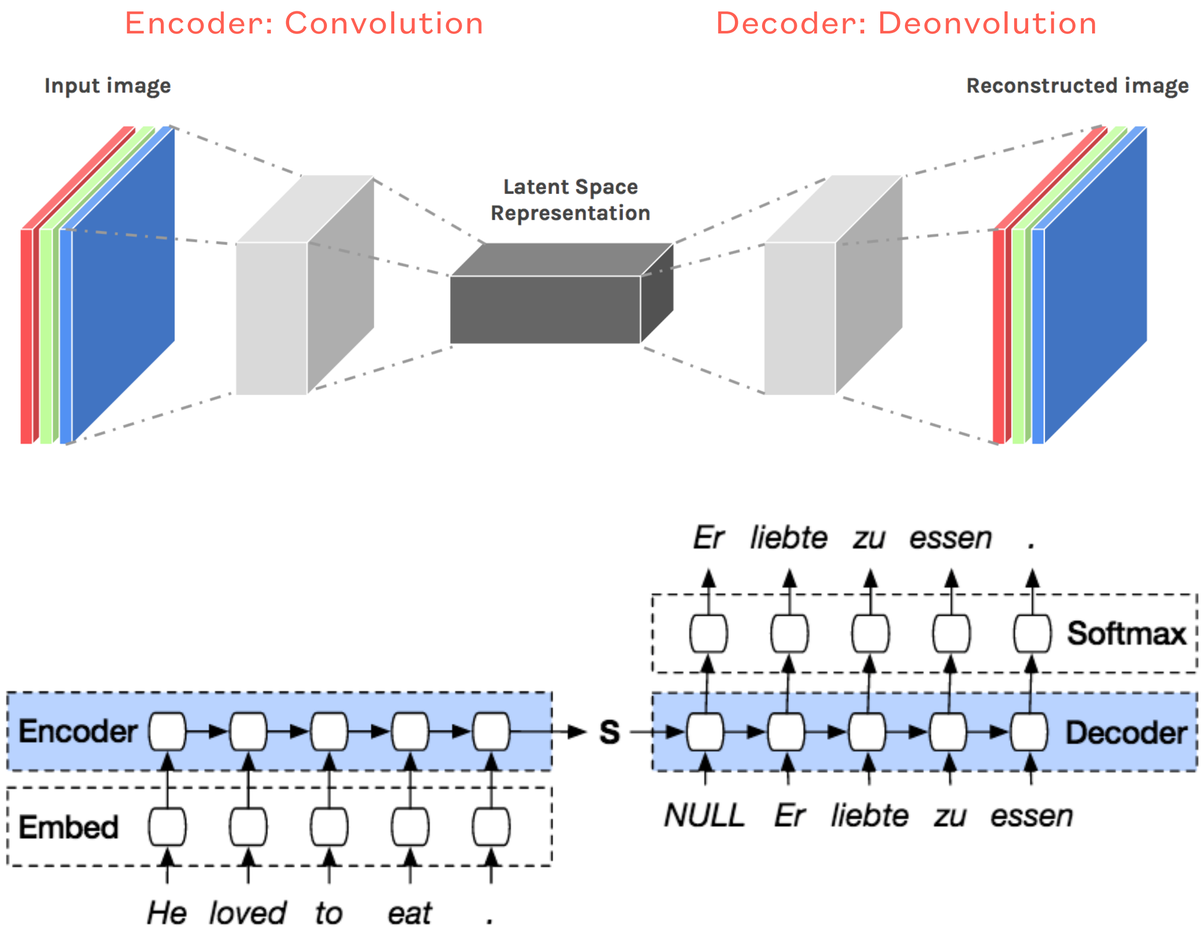


Figure 2. Residual learning: a building block.

Seq-to-Seq model

- This falls in the scope of NLP and machine translation, not necessarily useful for time series
- Encoder-decoder approach:
  - Easier to build, works well on time series
  - Training: Feed in English, train output toward Germany
  - Regressional loss (MSE for example)
- Seq-to-seq approach:
  - Works better on language model
  - Training: feed in both English and Germany
  - S is called the memory of Seq-to-Seq model
  - Store English/Germany words into their dictionaries, and treat each word as a class
  - For the given English word, ask the machine to classify which word to use
  - Classification loss: CrossEntropy



Transformer

- Transformer is a Seq-to-Seq version of MHSA network:
  - PE is added to both input and output
  - Input is analyzed by many MHSA block to produce memory:
    - MHSA layer + FC layers + Layer Norm + Residual Connection
  - Output is also analyzed by many MHSA blocks
  - Output and memory are jointly fed to another MHSA blocks followed by FC layers to produce classification decision
- I skipped a lot of technical details such as EOS tokens, teacher's enforcing, masking etc. If you are interested in it, I recommend you to read the blog on slide 9

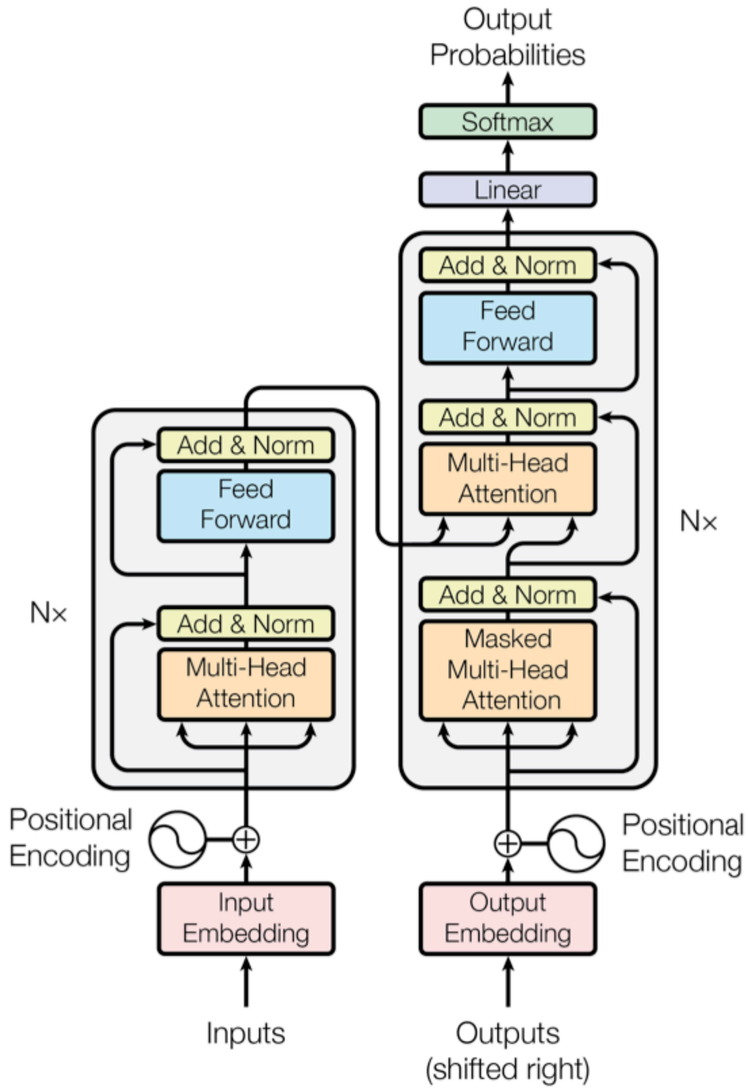
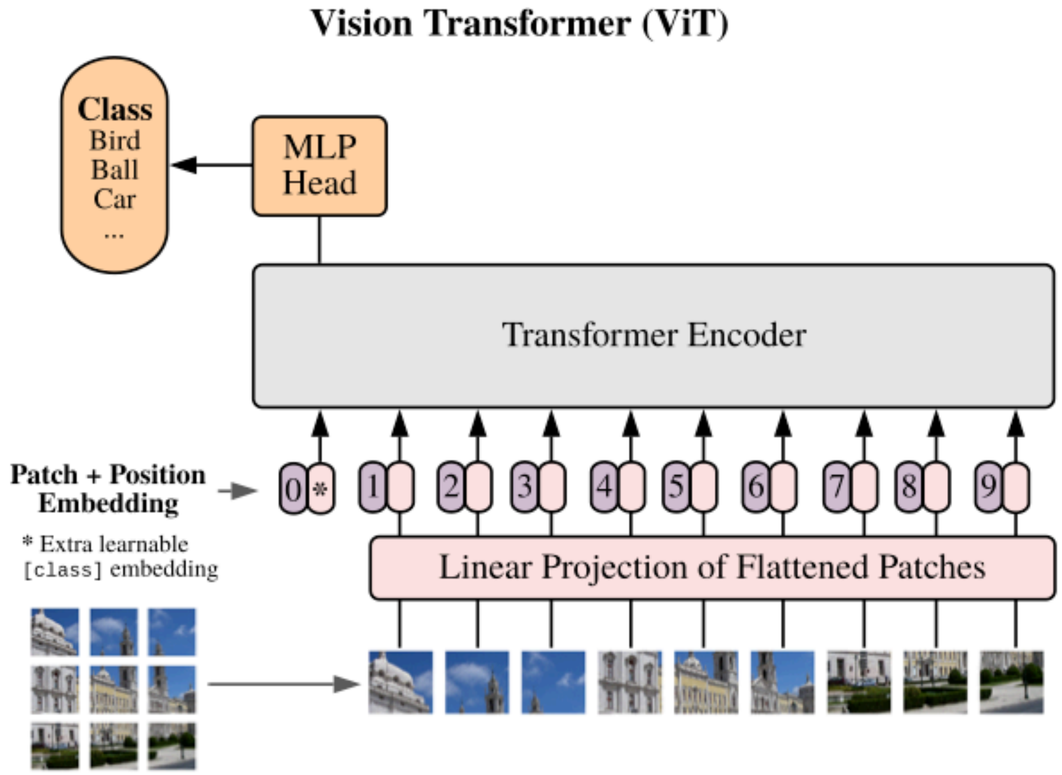


Figure 1: The Transformer - model architecture.

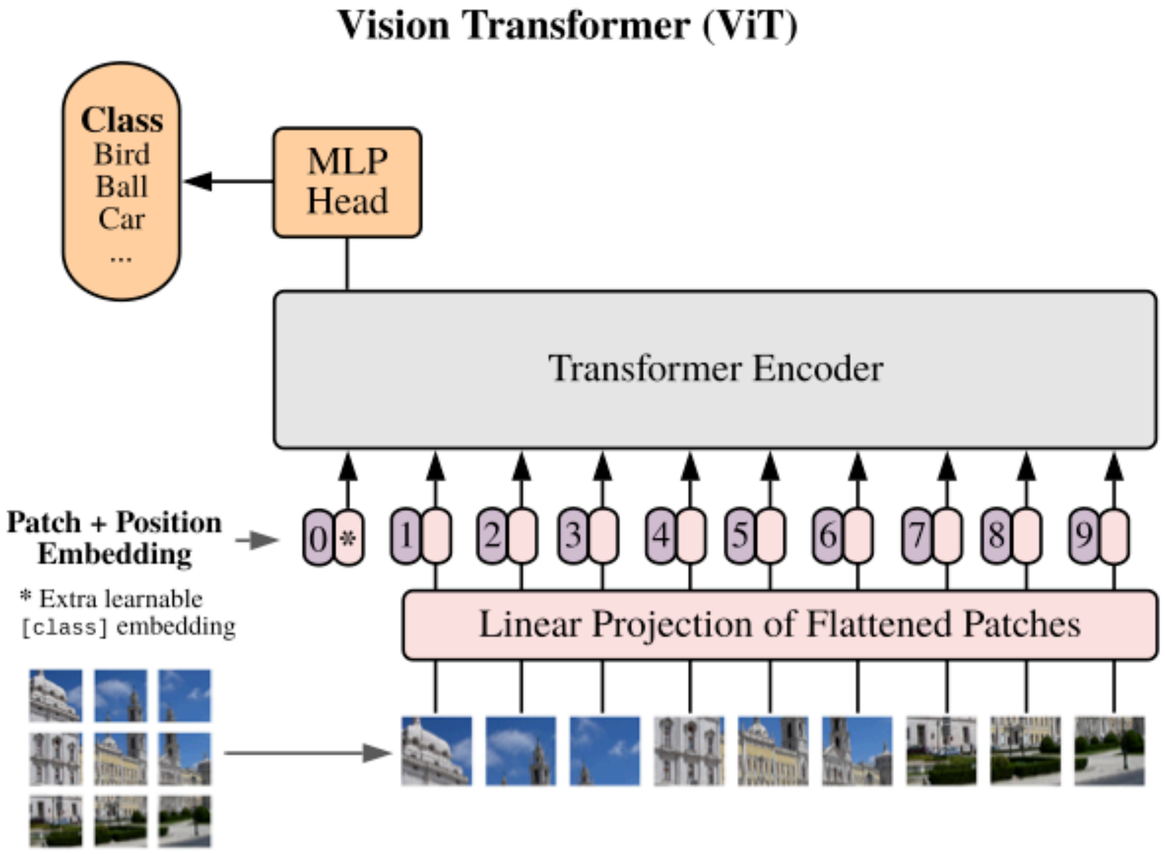
Vision Transformer

- First proposed in this paper:
  - https://arxiv.org/pdf/2010.11929.pdf
- Transformer has some very nice properties which makes it suitable for image data as well:
  - Global awareness
  - Strong at picking up correlations
  - Attention score, thus naturally interpretable
- In the second half of 2020, applying transformer to image data was a very hot research topic



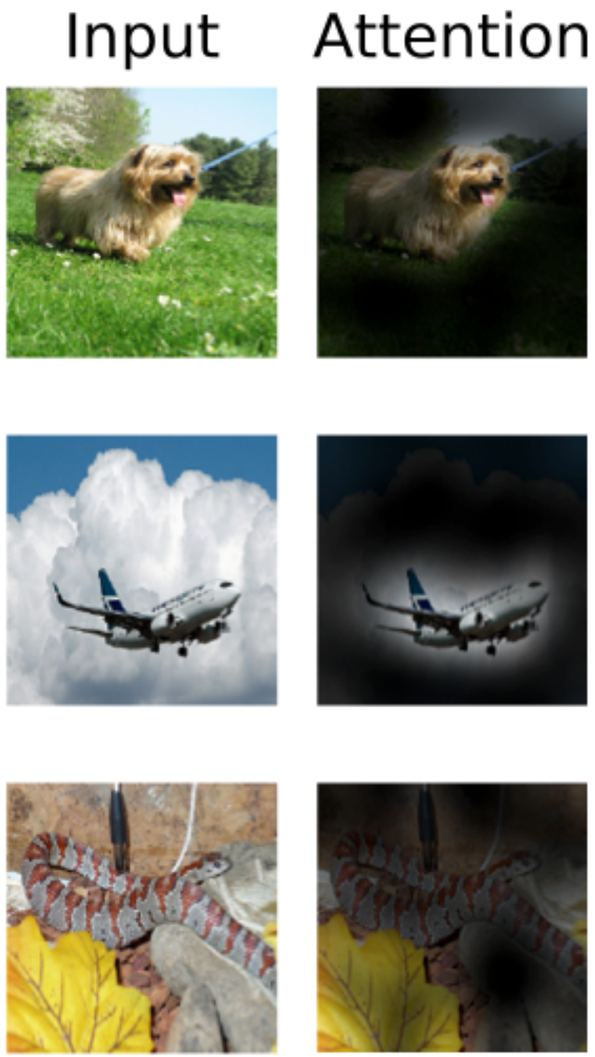
Vision Transformer

- Image is splitted into patches, each patch is flattened to a sequence
- The flattened patch is then fed to a FC layer to produce a linear projection
- An additional class embedding is added outside the patches
  - Class embedding will be able to query every other patches to extract useful information
- Positional encoding is added to the linear projections
  - The author tried both 1D PE and 2D PE, but did not find much performance differences
- Feed the class embedding to fully connected network for classification



Vision Transformer

- We can look at the attention score of vision transformer:
  - The network is correctly attending to the correct part of the image to make classification decisions
- These heatmap-style plot can be directly read out in Vision Transformer
  - If you want to make such a plot in CNN, you will need additional network interpretability tools such as GradCAM or guidedBP



Summary

- Review Attention Mechanism and RNN
- Revisit RNN as a Query Key Values formulation
- Introducing MHSA models:
  - Self-Attention as a QKV formulation
  - Multi-head Self Attention
  - Positional Encoding
  - Residual Connection
- Seq-to-Seq models and transformer
- Vision Transformer