Gentle? Introduction to Attention and Transformers

Andrew H. Fagg
Symbiotic Computing Laboratory
University of Oklahoma

Spatial / Temporal Data

Feature vector as a function of space and/or time

- Channels in a 1D/2D/3D image
- Features across time
- Combination of both

Spatial and/or Temporal Data

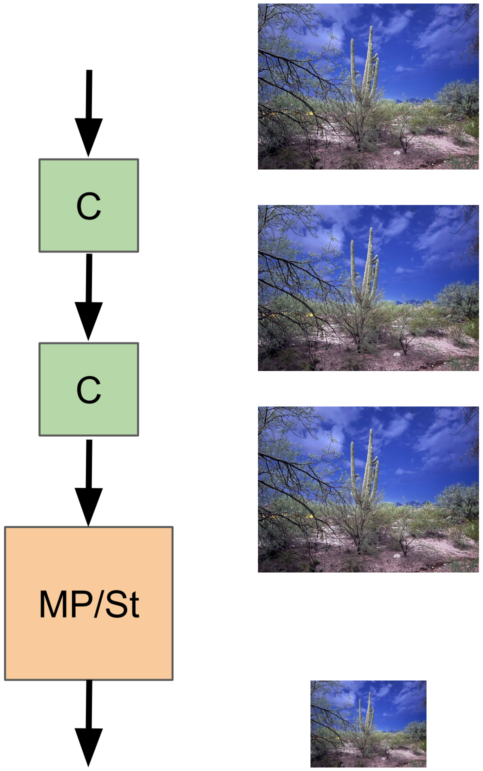
- Often apply the same computational operators across each of the feature vectors
- "Neighboring" feature vectors are also informative as to how we should interpret the current feature vector

Parts of a Convolutional Neural Network

- **Convolutional operators**: search for specific patterns within the spatial or temporal neighborhood
- **Max pooling operators**: does there exist a feature *somewhere* within the pool?
- **Striding** (often coupled with pooling): decrease the spatial or temporal resolution

Image CNNs

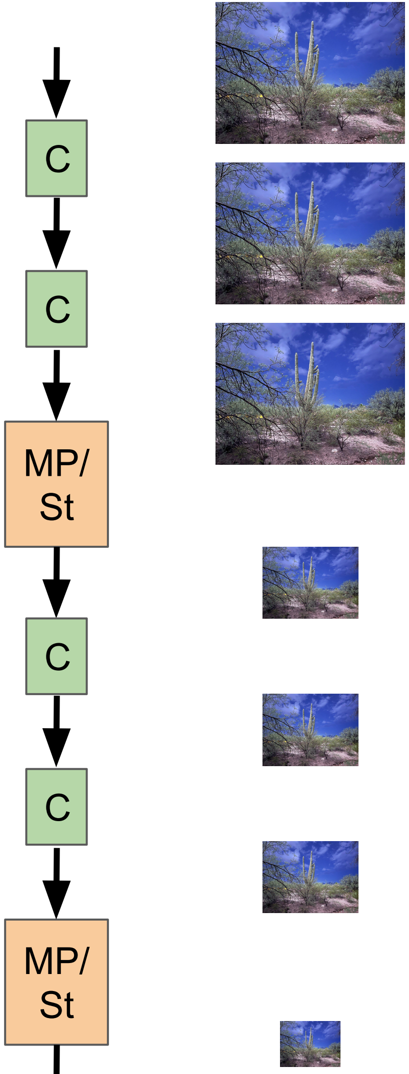
Typical CNN module



Sequences of Modules

With striding: a constant sized convolution mask (e.g., 5x5) can “see” a larger region of the image than in the previous module

- Can recognize larger spatial patterns
- Higher level abstractions

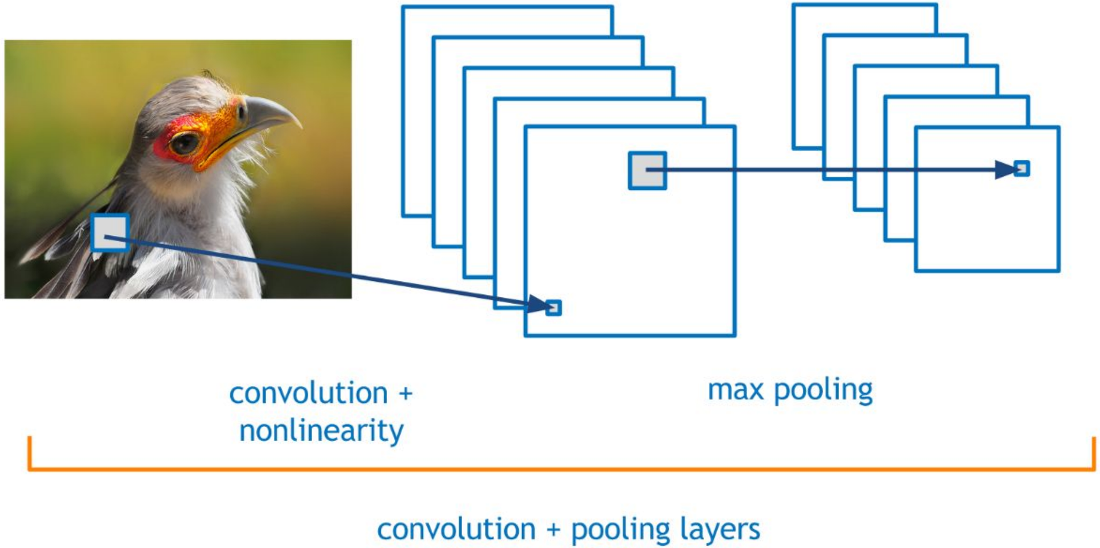


Typical Image Classification CNN

Modules learn different types of abstractions

- Module 1: very short line segments or edges
- Module 2: longer segments/edges
- Module 3: corners, curves
- Modules 4+: basic shapes
- …

Local Operators

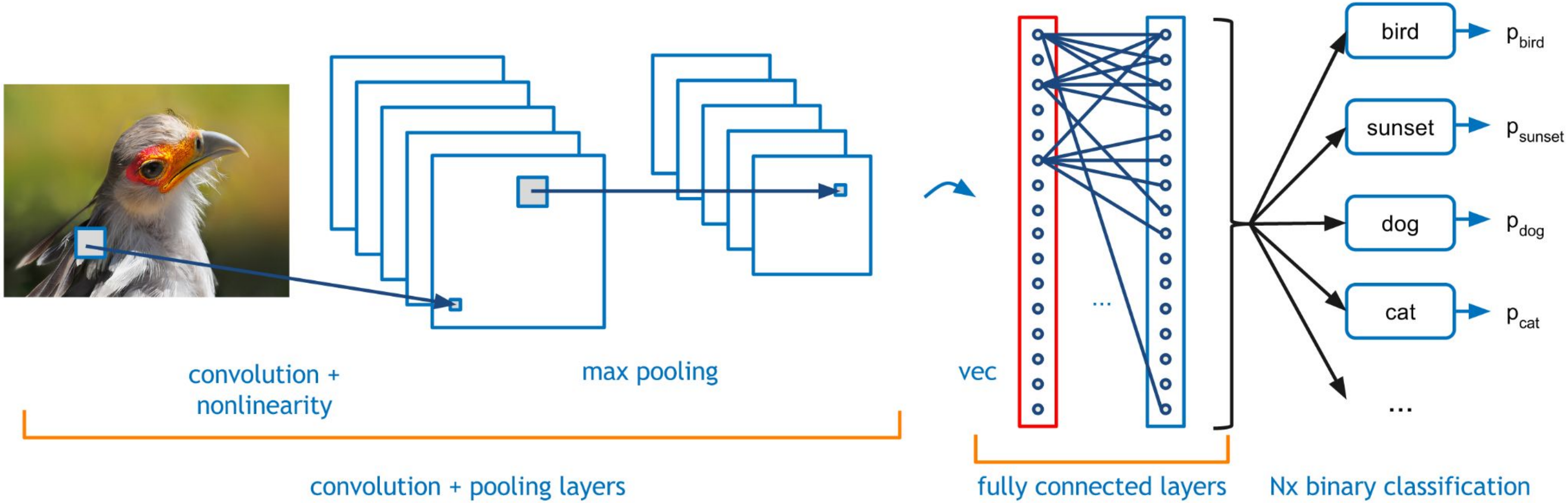


wp.flickr.net

Typical Image Classification CNN

- Higher levels of abstraction -> larger number of possible patterns for a given abstraction
- For the bird:
    - Easy to build feature detectors for eyes, beak, feathers, feet…
    - But a very large number of options for how these relate spatially together to form a “bird” abstraction
- Typical approach: Flatten or GlobalMaxPool and then a sequence of Fully Connected layers

Combining Local Operators to Recognize Global Patterns



wp.flickr.net

Typical Image Classification CNN

- Flatten: unwind rows x cols x channels into a 1-D vector
  - Dense layers can then learn arbitrary spatial relationships
- GlobalMaxPooling: for each channel, compute max over rows x cols
  - Completely throw out the spatial relationships between the high-level features
  - There is a beak, an eye, a foot and a feather -> it is a bird

Challenges

Spatial relationships between the high-level features are often important!

- GlobalMaxPool architecture throws away the spatial relationships
- The 'Flattened' architecture allows us to preserve the spatial relationships, but need a lot more training data to capture all the possibilities

Really want some compromise …

The Plan

- 1D data & Recurrent Neural Networks
  - Compact approach for integrating information across long sequences
- Example: Sentence-to-sentence translation
- Attention: tool for focusing on specific pieces of information across the sequences
- Transformers (attention^2) (session 2)
- Transformers with 2D data (session 3)

Recurrent Neural Networks

Processing feature vectors in time and/or: producing some output in time

- Sequential steps for a robot control signal
- Processing textual input
- Producing textual output

Each time step: use the same network to get to the next time step

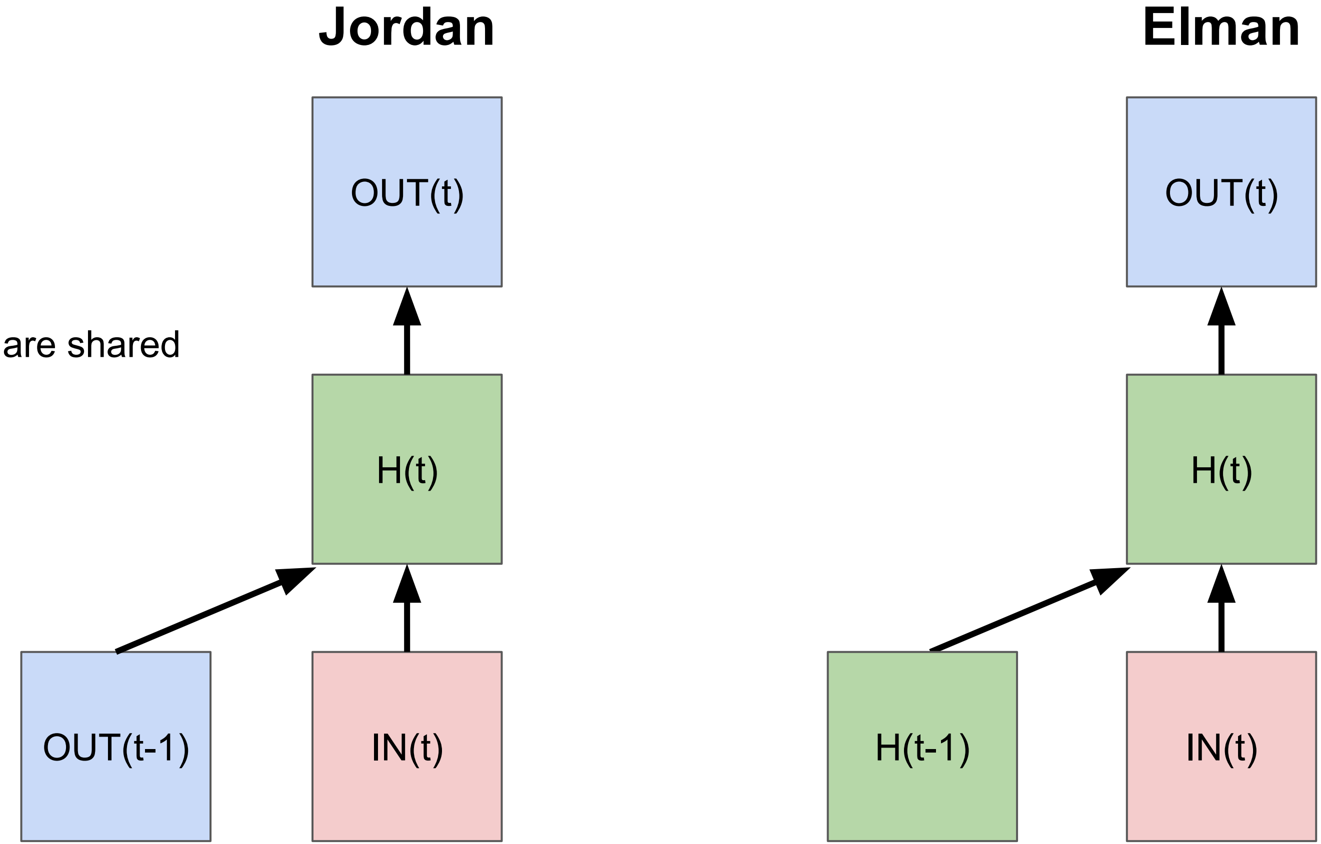
Recurrent Neural Networks

- Jordan (1997): output at time t is an input to the network at time t+1
- Elman (1990): hidden layer state at time t is an input to the network at time t+1

Either way: the extra input acts as a context for producing the next output

Recurrent Neural Networks

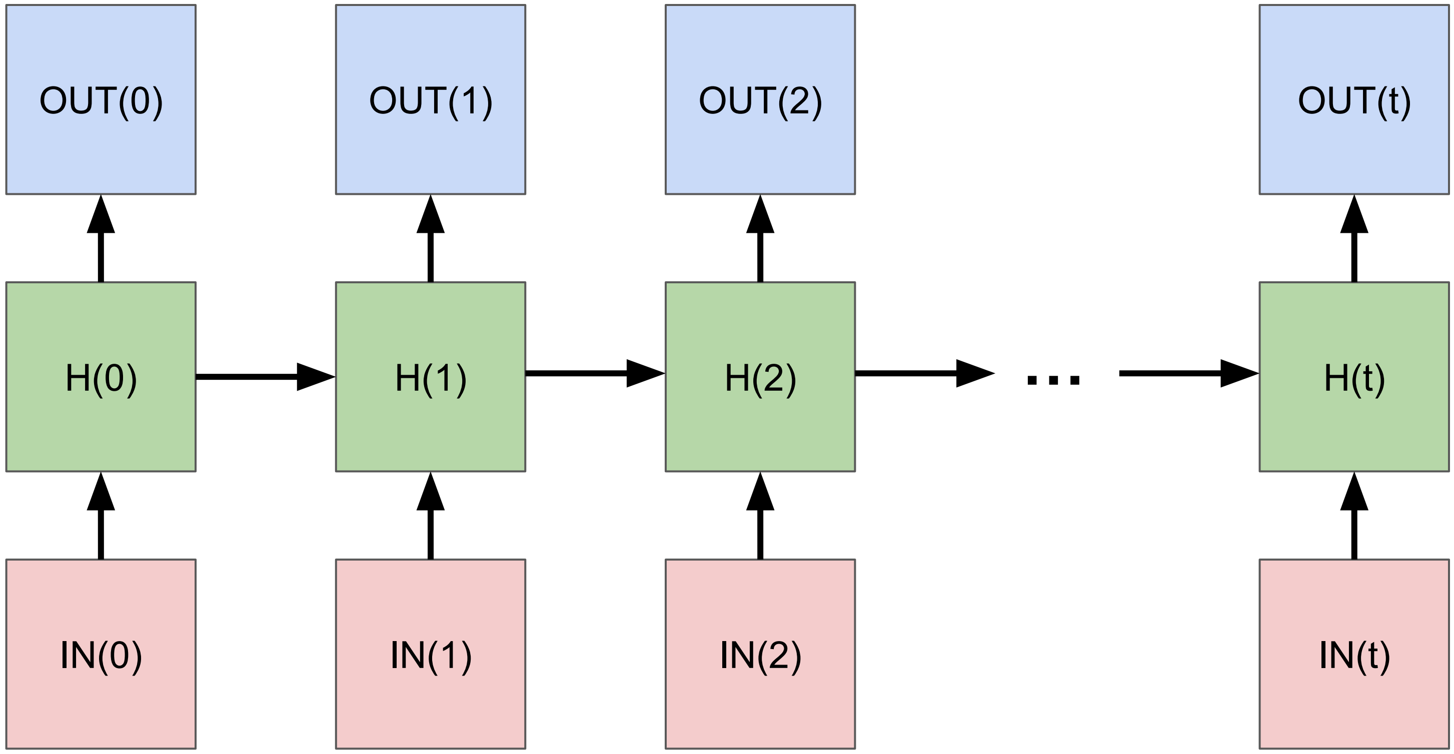
Parameters are shared across time



Backpropagation Through Time

- Jordan and Elman: error gradient only flowed through the network for one time step
  - Still had to supply input/output pairs for each time step
  - Could only hope that the extra input provided sufficient information
- Werbos (1988): Backpropagation through time: error gradients flow across time

Unrolling the Recurrent Network



Unrolling the Recurrent Network

- Parameters are shared at each time step
- Error gradients can pass across time
- Hidden state can carry key information across many time steps

Note: there are key similarities with 1D Convolution

RNN Architectures

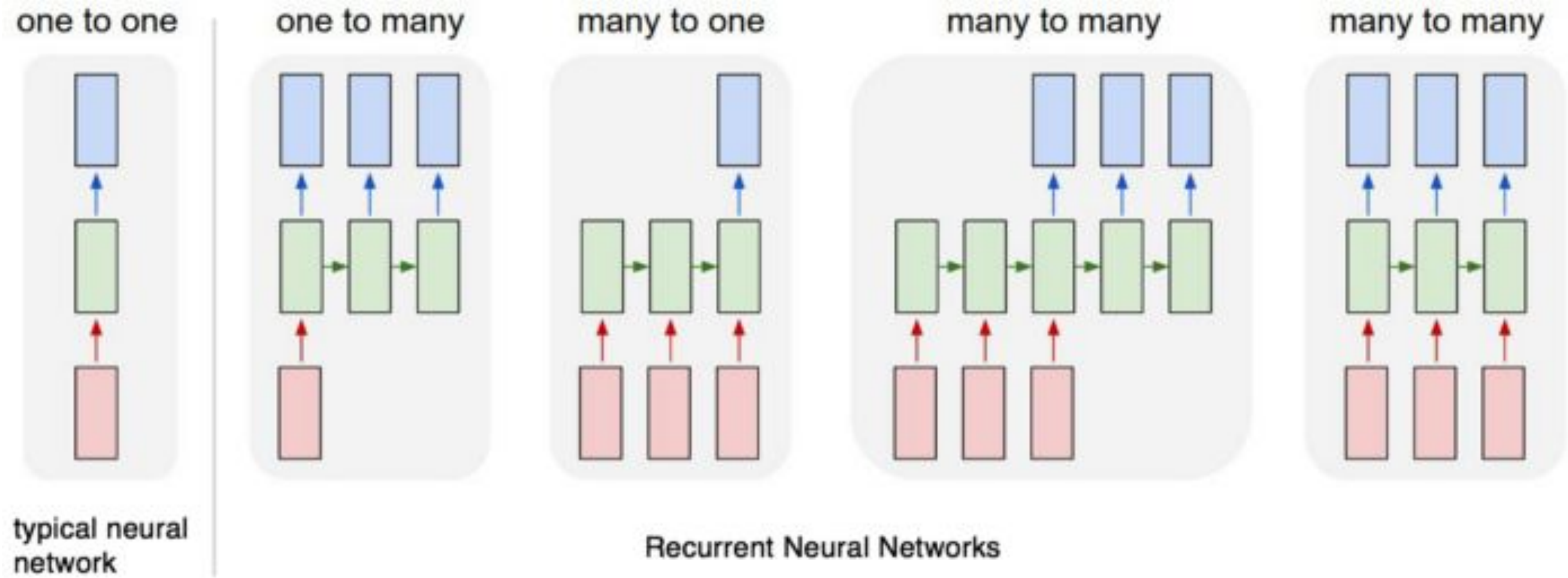


Image from: Andrej Karpathy

Use in Machine Translation

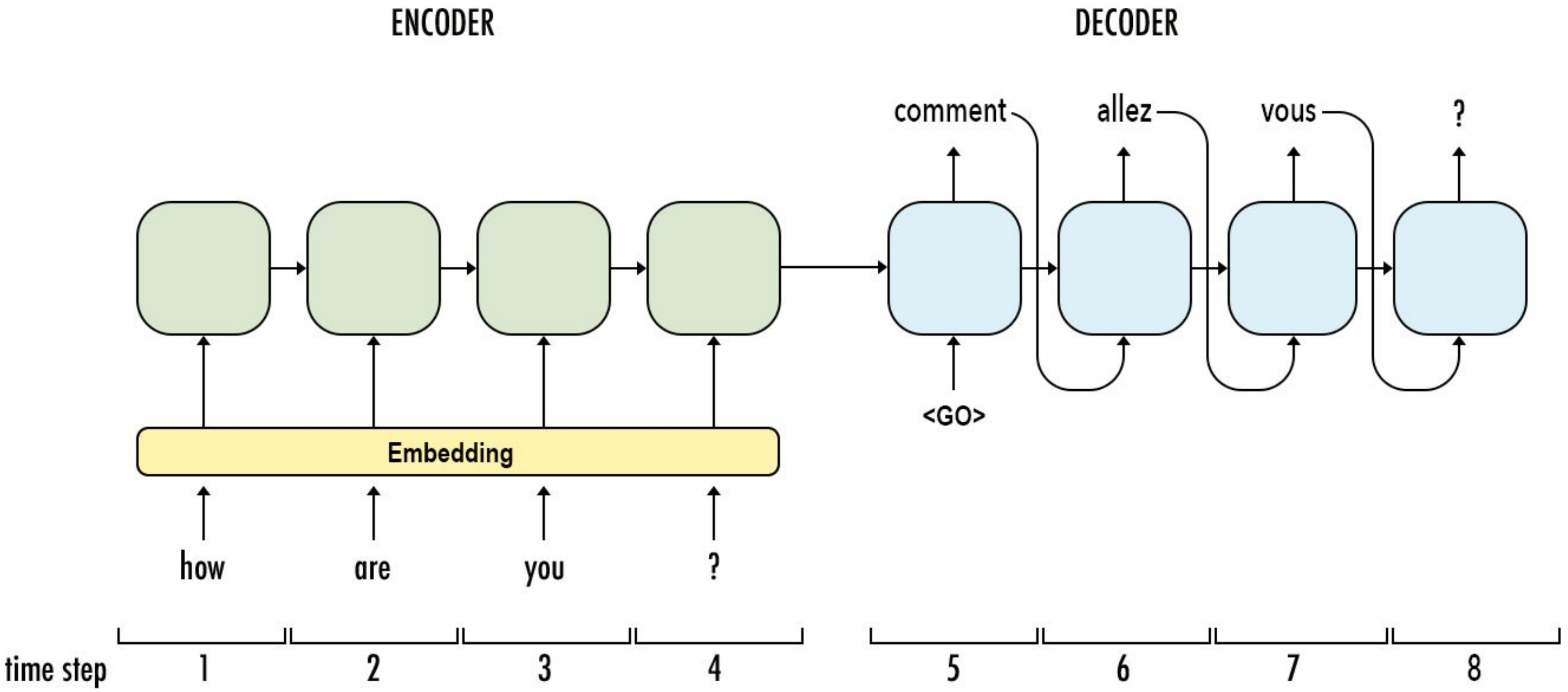


Image from: Udacity

Challenges

- Hidden state may need to carry critical information from the first token in the input to the final token in the output sequence
- Learning these representations requires propagating error information through all of these hidden state layers
- Can be many steps, especially when we are translating one paragraph at a time
- Vanishing error gradient can prevent a network from learning a mapping from input to output in feasible time

Vanishing Gradient

Many solutions to the vanishing gradient problem:

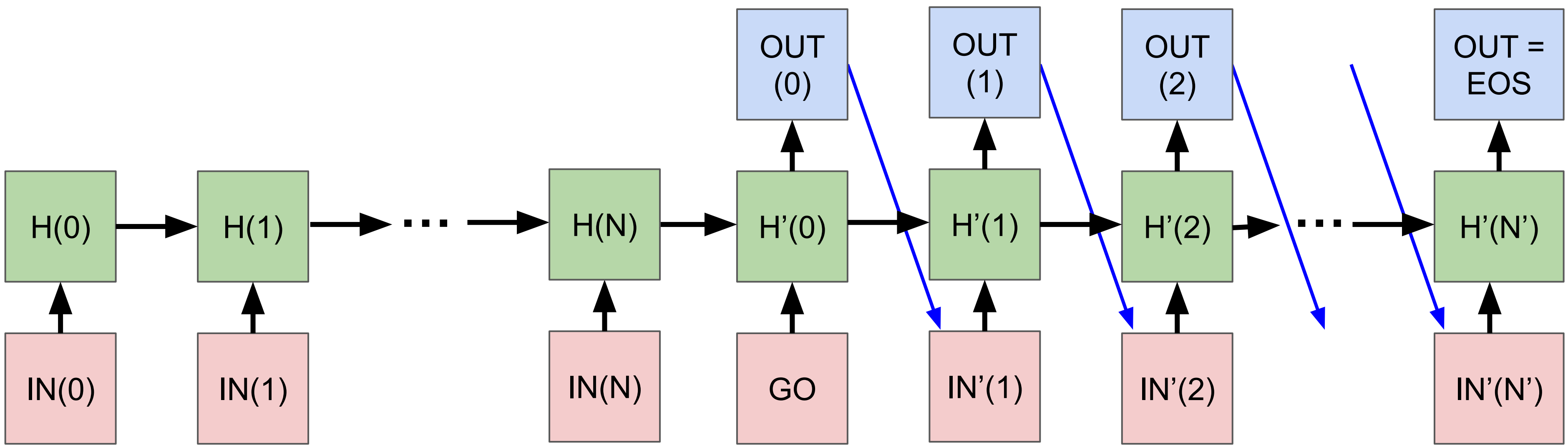
- Ioffe & Szegedy (2015): Batch normalization
- Hochreiter & Schmidhuber (1997): Long/short-term memories (LSTM)
- Cho et al. (2014): Gated Recurrent Unit (GRU)

An Alternative Approach

Key insight:

- To decide which token to generate at time t, we don't need the context of the entire input sentence (or paragraph)
- Really only need to know a handful of the input words & their spatial relationship to the current word
- **Attention**: blend the representations of only the tokens of interest & use the result to decide on the current output token

RNN for Machine Translation



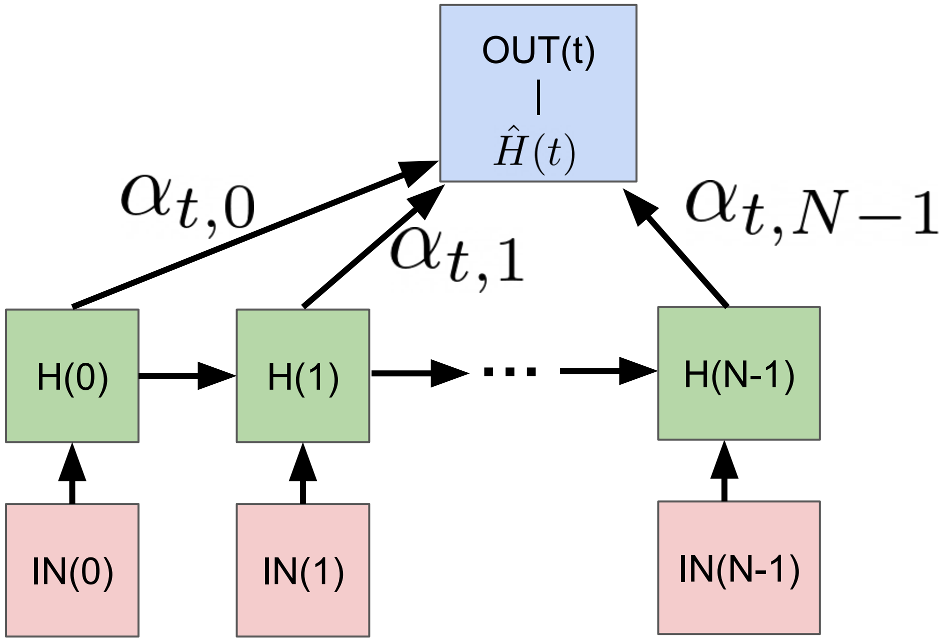
Attention for Machine Translation

N time steps. Each time step t has:

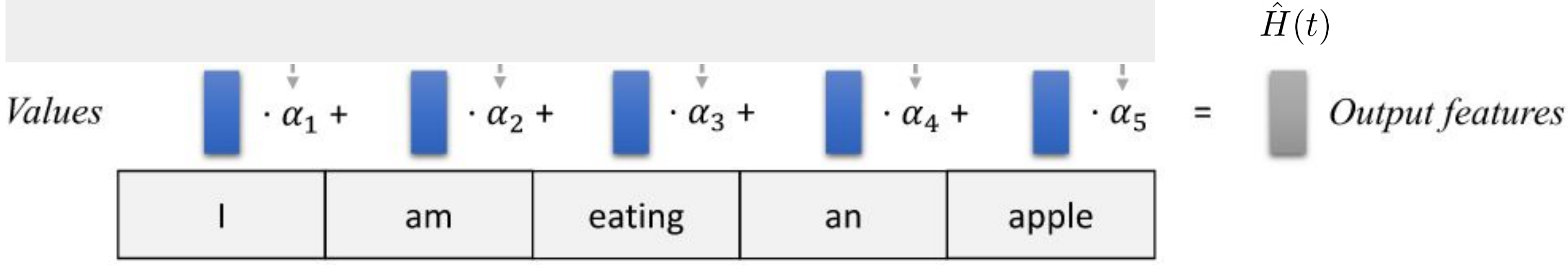
- Set of alphas that sum to 1
- A blended version of all N latent states

$$\sum_{i=0}^{N-1} \alpha_{t,i} = 1$$

$$\hat{H}(t) = \sum_{i=0}^{N-1} \alpha_{t,i}\, H(i)$$



Attention for Machine Translation

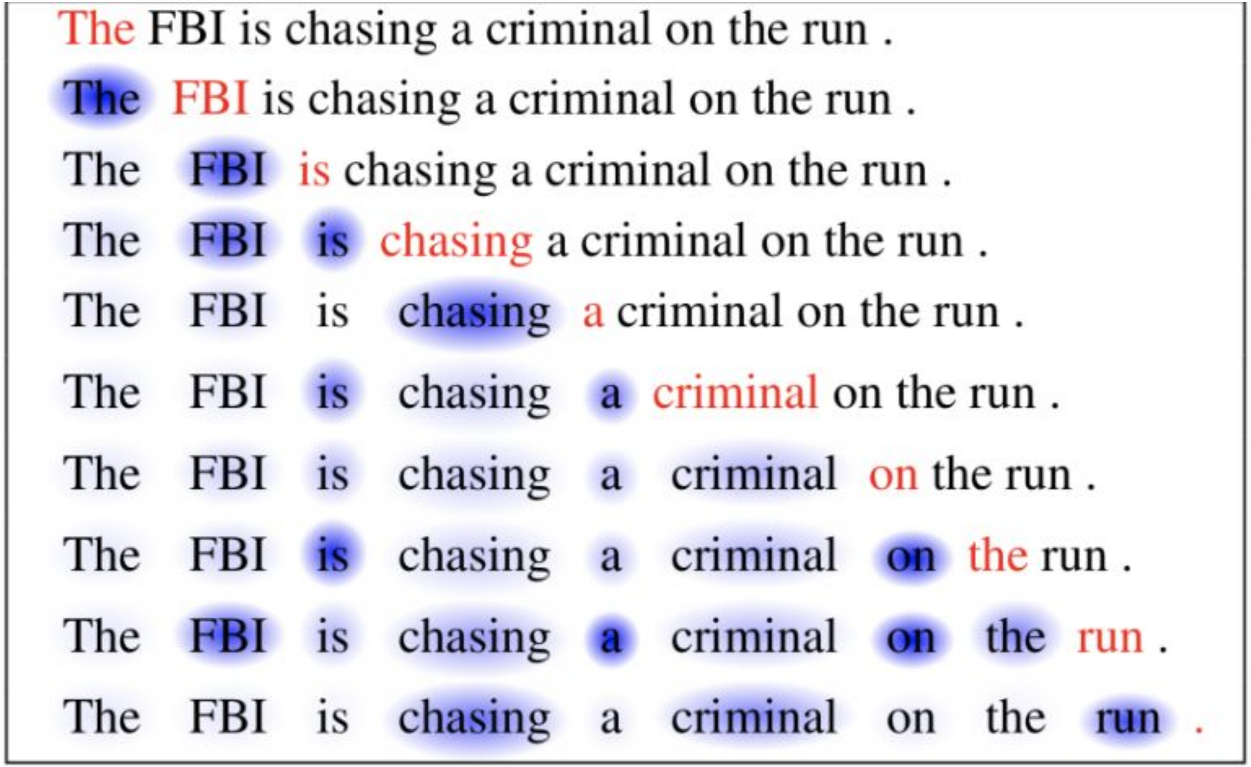


https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial6/Transformers_and_MHAttention.html

Attention for Machine Translation

$\alpha_{t,i}$ is the degree that H(i) plays in $\hat{H}(t)$

- By selecting a small number of non-zero $\alpha_{t,i}$'s, the output generator can choose to focus on a small number of H(i)'s
- This means that many of the H(i)'s are ignored while generating the output for time t
  - … and these ignored H(i)'s do not propagate error gradients



https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Content-Addressable Memories

- Memory is composed of a set of key/value pairs
    - **k** and **v** are row vectors
- A query is compared to the set of keys
    - Hard version: the value for the one matching key is "returned"
    - Soft version: a blend of the best matching keys is "returned"

Content-Addressable Memories

- Degree of match between a query (**q**) and a single key (**k**):

$$s = q\,k^T = \|\,q\,\|\;\|\,k\,\|\;\cos\left(\theta_{q\rightarrow k}\right)$$

- Degree match between a query (**q**) and a set of keys (**K**)

$$S = q\,K^T$$

This gives us a row vector of scores

Content-Addressable Memories

- Row vector of scores: $S = q\, K^T$
- Use softmax to translate scores into alphas:

$$\alpha = \mathbf{softmax}(q\, K^T) \qquad\qquad \alpha_i = e^{s_i} \Big/ \sum_{j=0}^{N-1} e^{s_j}$$

- Blend in each value according to its corresponding alpha:

$$\hat{v} = \mathbf{softmax}(q\, K^T)\, V = \sum_{i=0}^{N-1} \alpha_i\, V_i$$

Content-Addressable Memories

- Blend in each value according to its corresponding alpha:

$$\hat{v} = \mathbf{softmax}(q\ K^T)\ V = \sum_{i=0}^{N-1} \alpha_i\ V_i$$

- Can also parallelize for a set of queries:

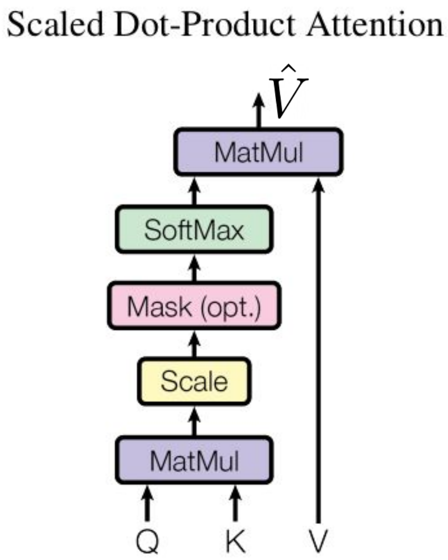
$$\hat{V} = \mathbf{softmax}\left(Q\ K^T\right)\ V$$

Note: comparing all N keys against all N queries

Implementing Attention

Scaled Dot-Product Attention: DL Implementation

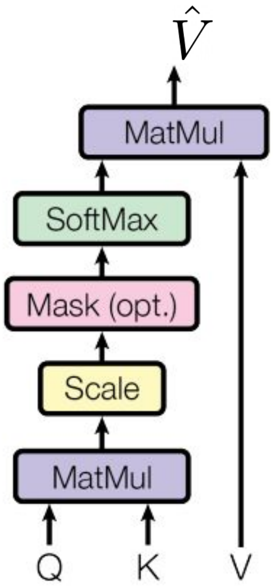
- All inputs are matrices of the same size (N x #features)
  - Q: Queries
  - K: Keys
  - V: Values
- Output is also N x #features



Vaswani et al. (2017)

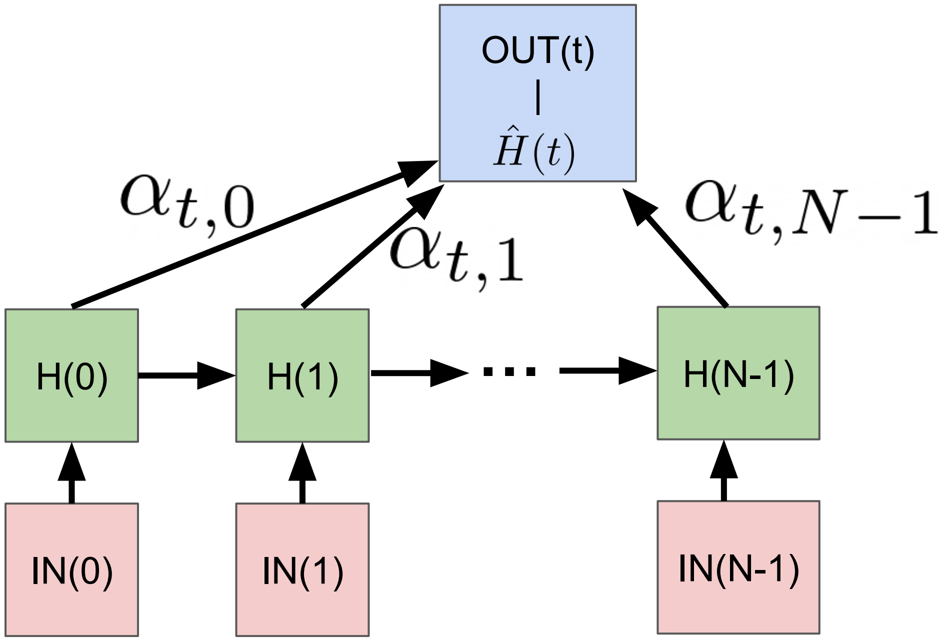
$$\hat{V} = \mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Mapping Attention to our RNN



Many options, one possibility:

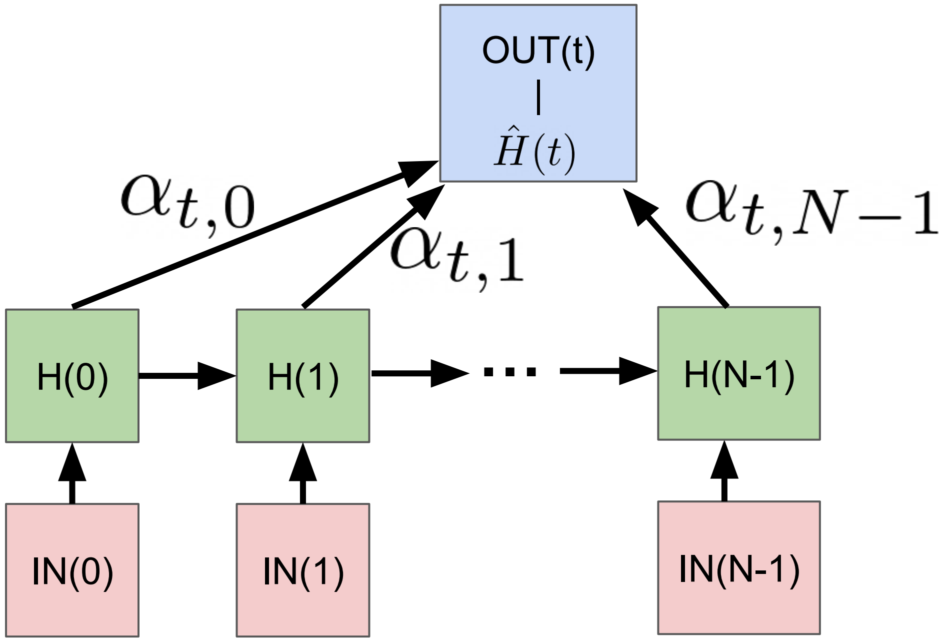
- Outputs: $\hat{H} = \hat{V}$
- Inputs:
  - K(t), V(t) = H(t)
  - Q(t) = $\hat{H}(t-1)$



RNN Training

In this simple form:

- Attention is fixed
- Use backpropagation to simultaneously learn:
  - Encoder that produces H(t) from H(t-1) and IN(t)
  - Decoder that produces OUT from $\hat{H}(t)$



RNN Training

- Through attention, every output token has the opportunity to examine every input token, so we are doing N^2 comparisons
- We are still doing backpropagation through N latent layers (our H’s)

Multi-Headed Attention

- Explicitly acknowledge that categories of information need to be extracted and represented separately
- For example, may want to separate the input words that describe:
  - The action (verb)
  - Modifications to the action (adverbs)
  - The subject
  - …

Multi-Headed Attention

Approach: multiple single-headed Attention modules are used in parallel. For each head:

- Input: its own "perspective" on the MH Attention inputs (implemented as three linear projections)
- Output: its own $\hat{H}_i(t)$

Grand output is a linear combination of the the individual heads: $\hat{H}(t) = \sum_{i=0}^{K-1} w_i^O \; \hat{H}_i(t)$

Implementation Details

Single-Head Attention:

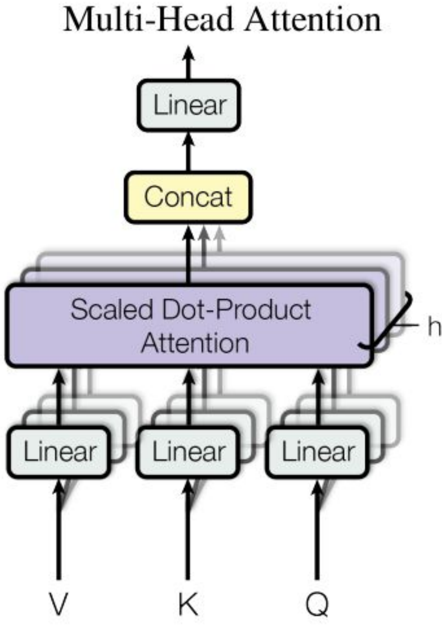
$$\text{Attention}(Q, K, V) = \text{softmax}\left(Q\, K^T\right)\ V$$

Multi-Head Attention:

$$Q_i = Q\, W_i^Q \quad K_i = K\, W_i^K \quad V_i = V\, W_i^V$$

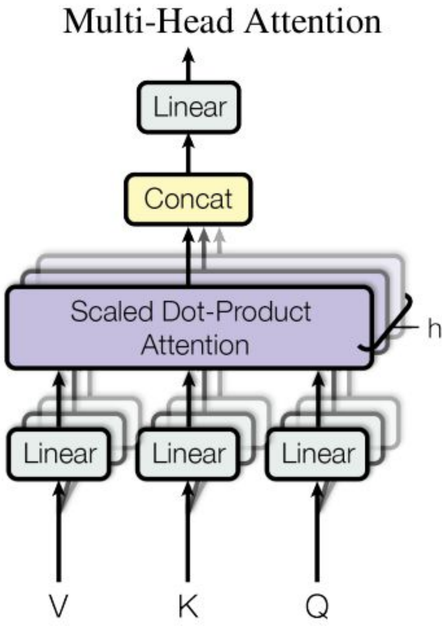
$$\hat{H}_i = \text{Attention}\left(Q_i, K_i, V_i\right)$$

$$\hat{H} = \sum_{i=0}^{K-1} w_i^O\ \hat{H}_i$$



Implementation Details

- Each head has its own linear parameters
  - Linear parameters are shared across the input tokens
- These parameters are selected as part of the larger learning process
- This allows each head to specialize in what types of information it extracts

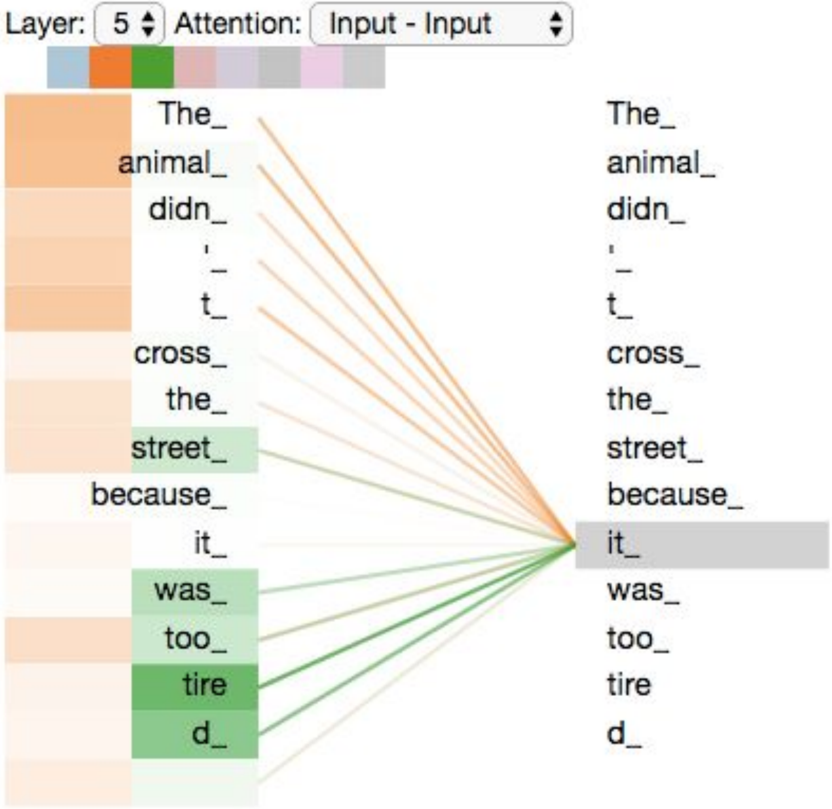


Multi-Headed Attention

The animal didn't cross the street because it was too tired

Two Attention heads:

- What does ***it*** refer to?
- What is the description of ***it***?



http://jalammar.github.io/illustrated-transformer/

RNN Training with Attention

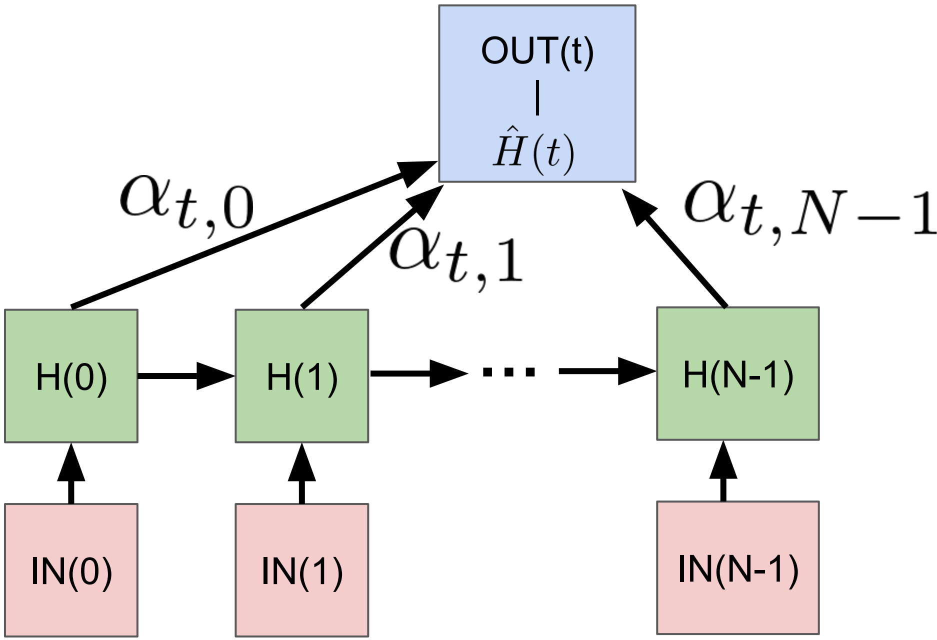
- Through attention, every output token has the opportunity to examine every input token, so we are doing N^2 comparisons
- We are still doing backpropagation through N latent RNN layers (our H's)

The deep backpropagation is still a big computational problem

Re-Examining the RNN

- H(t) is a function of all of the input tokens IN(0) … IN(t)
- H(t) must contain information that is useful for H(t+1) …

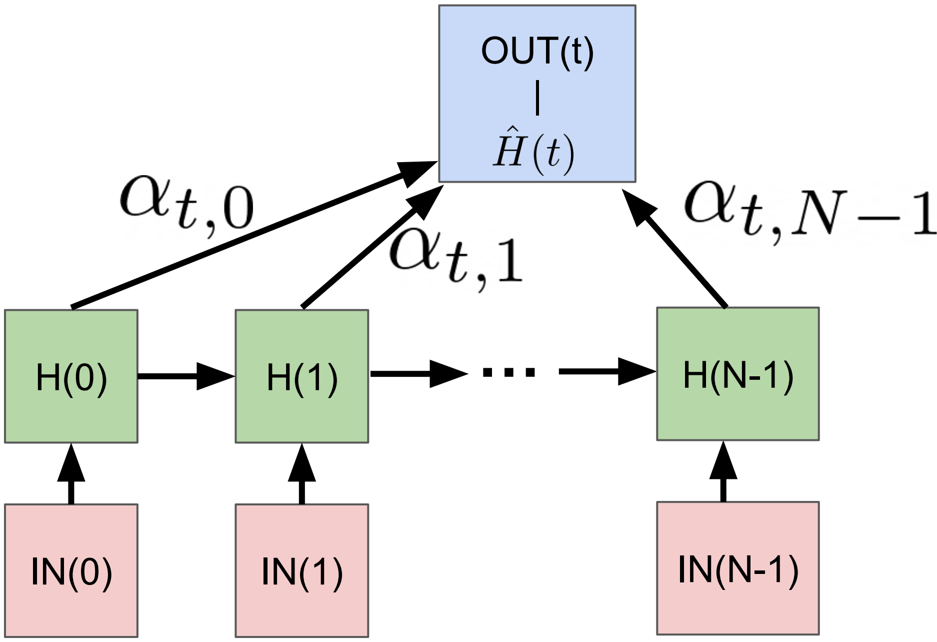
What if H(t) could just focus on the input tokens that are relevant specifically to the decisions that it needs to make?



Re-Examining the RNN

What if H(t) could just focus on the input tokens that are relevant specifically to the decisions that it needs to make?

-> This sounds just like Attention!



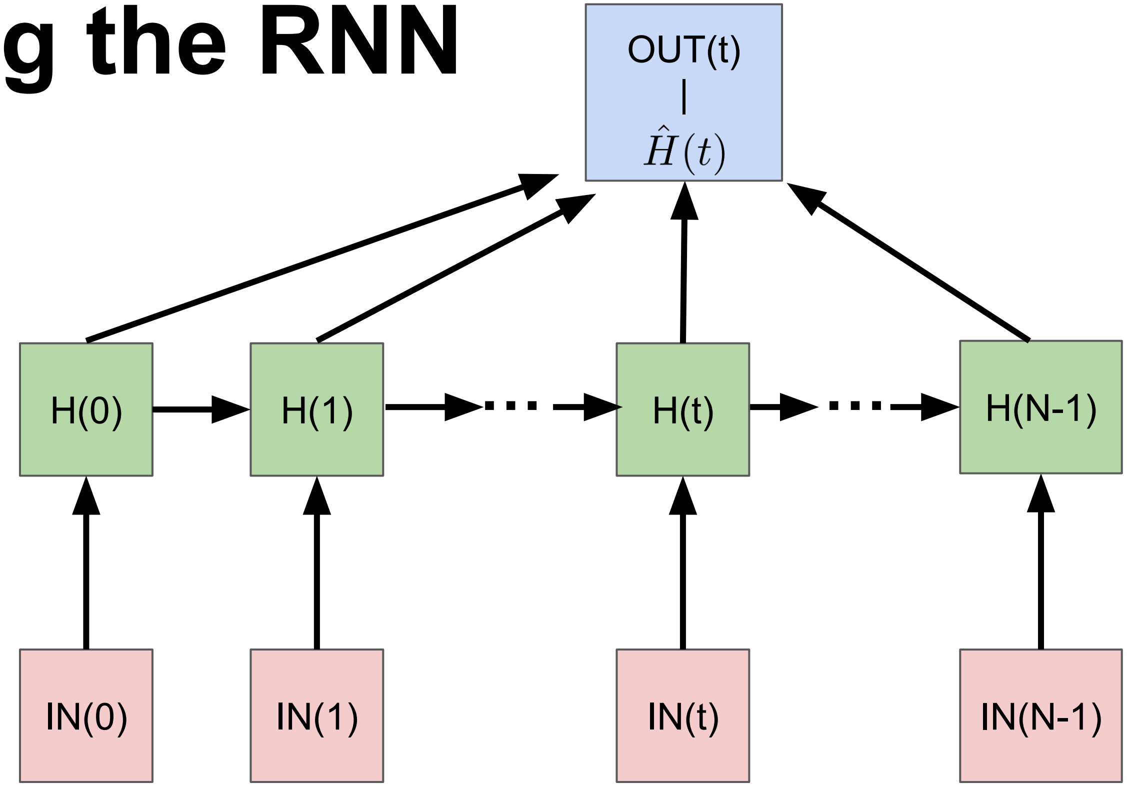
Attention is All You Need

Vaswani et al. (2017):

- Attention to process the input tokens
- Attention to generate the output tokens

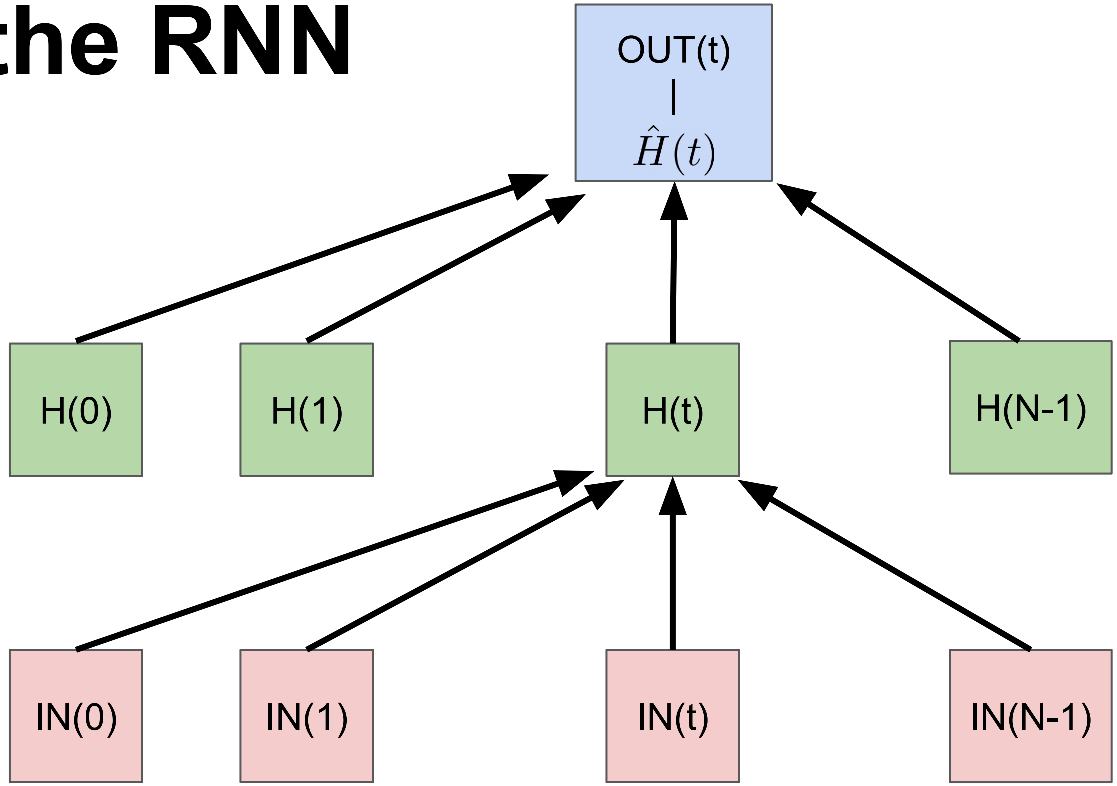
Re-Examining the RNN

Primary challenge comes from the connection from H(t) to H(t+1)



Re-Examining the RNN

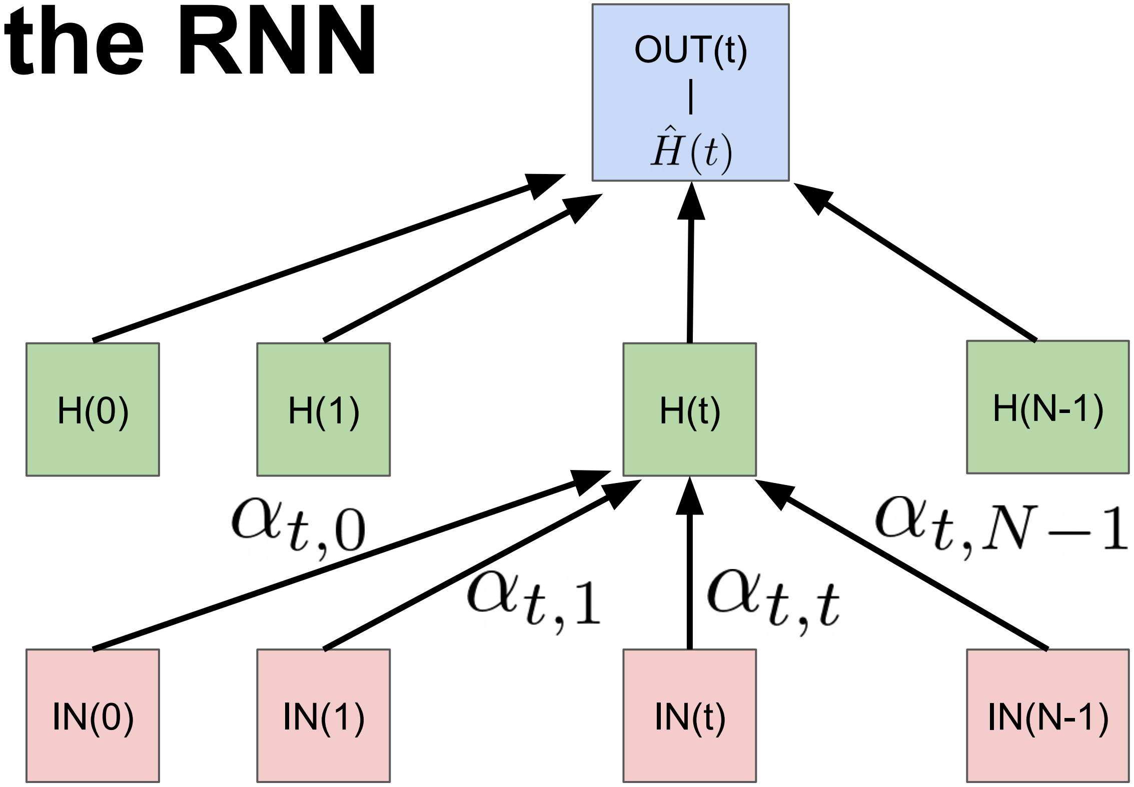
Each latent step has access to all inputs simultaneously



Re-Examining the RNN

Each latent step has access to all inputs simultaneously

- Blend of all inputs
- Implemented using Multi-Headed Attention



Transformers

Do we need two different layers here? Details are hidden in the two layers of Attention:

- Decoder: Keys capture the sequence of outputs decoded so far
- (new) Encoder: Keys will just be the inputs

Tool: Position Encoding

- With our RNN encoder, the relative positions of the input token are captured in the latent representation
- Likewise, with our decoder, relative positions of the output words were captured in the blended latent representation & the output

By replacing our RNN encoder with MH Attention, we lose an encoding of position

Position Encoding

Goals for a positional encoding:

- Want the network to be able to reason about absolute position in the sequence of an input token
- Also want the network to be able to reason about the relative position of two input tokens
- Should make use of finite values, even when the sequences are long

Positional Embeddings

Approach: for each token, translate its integer position in the sequence (t) into a vector

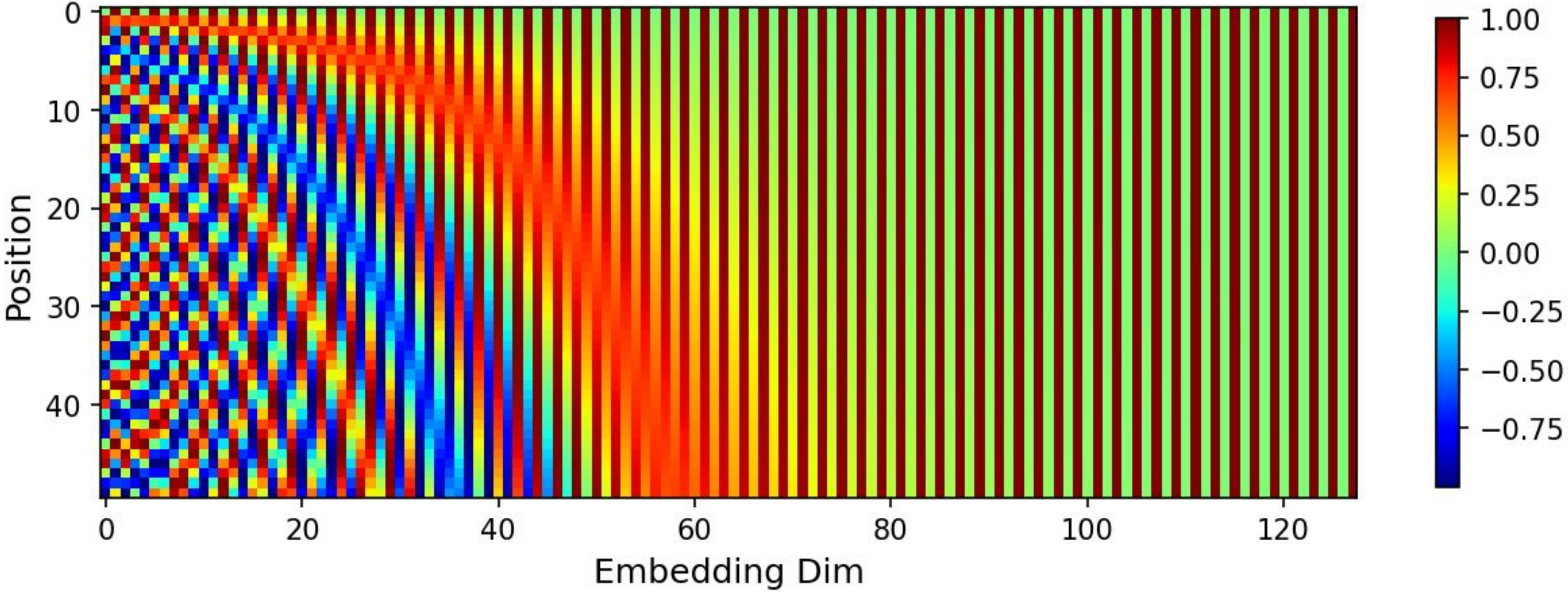
$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

where:

$$w_k = \frac{1}{10000^{(2\,k/d)}}$$

Vaswani et al. (2017)

Embedding Example



Positional Embeddings

Details:

- d is selected to match the token embedding size
- #positions is the maximum “sentence” length
- 10,000 is selected so that the different positional encodings are distinct
- Each element falls within +/- 1

$$\vec{p_t} = \begin{bmatrix} \sin(\omega_1 . t) \\ \cos(\omega_1 . t) \\ \\ \sin(\omega_2 . t) \\ \cos(\omega_2 . t) \\ \\ \vdots \\ \\ \sin(\omega_{d/2} . t) \\ \cos(\omega_{d/2} . t) \end{bmatrix}_{d \times 1}$$

Positional Embeddings

Key property: easy to compute the relative difference between two positions

- Consider the encoding of two positions: p_t and $p_{t+\Delta t}$
- For any t and Δt, they are related through a fixed linear transformation: $p_{t+\Delta t} = D(\Delta t)\, p_t$
- Where:
  - $D(\Delta t)$ is a fixed tridiagonal matrix that is only a function of Δt !

Positional Embeddings: Implications

Two tokens that are separated by a fixed distance (Δt) share the same $D(\Delta t)$

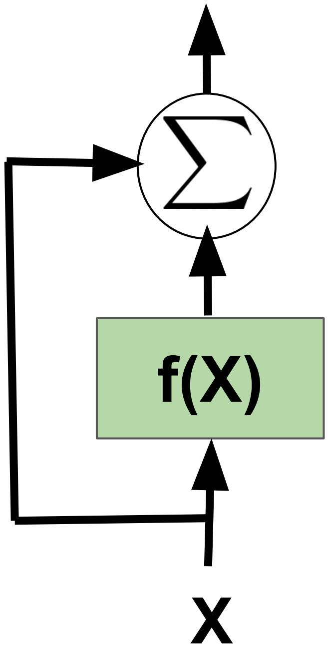
- Relative positions between tokens are really easy to compute by our network
- The phrase “they are” are the same positional difference, no matter their absolute location
- The phrase “are they” has a distinct difference
- Allows for generalization to sequence lengths that are different than what the network is trained with

Next Tool: Skip Connections

- Shapes of X and f(X) are the same
- Error propagation through f(): it can be hard to find a gradient
- Error propagation through the skip is trivial

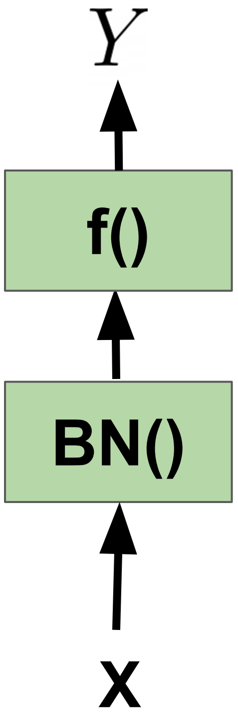
-> Even when there are many of these modules stacked on top of each other, there is an easy gradient to find

$$Y = X + f(X)$$



Final Tool: Batch Normalization

- Statistics of X over a large batch *can* be anything
  - Assume that we fall within a Normal dist
- BN() scales and translates each element of X so that the inputs to f() fall within N(0,1)
- This means that the **net inputs** to f() are more likely to fall within the dynamic range of the non-linearity within f()



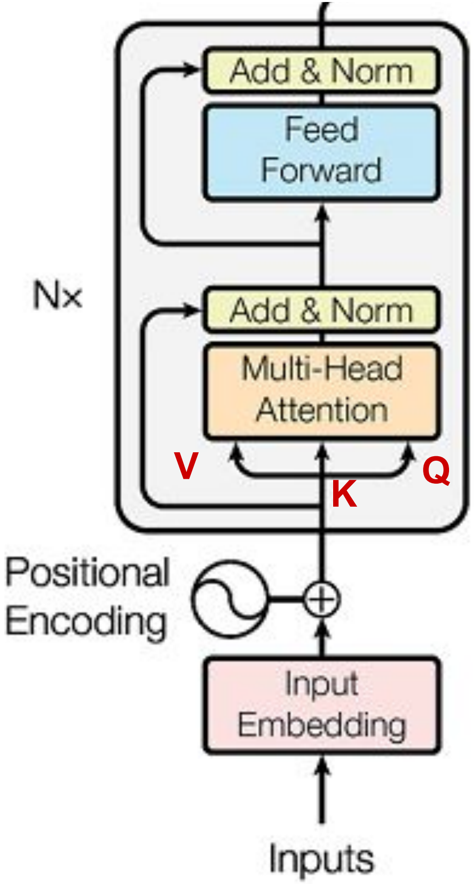
-> Much less likely to vanish the gradient

Putting All the Pieces Together

- Input: encoded sequence of tokens (N x d)
- Encoder: use Attention to create a sequence of "hyper-tokens" (also N x d), each of which captures some subset of the token sequence
  - Computed in parallel
- Decoder: use Attention to "read out" one token at a time
  - Combines the latent representation of the encoder with what has already been read out

Transformer: Encoder

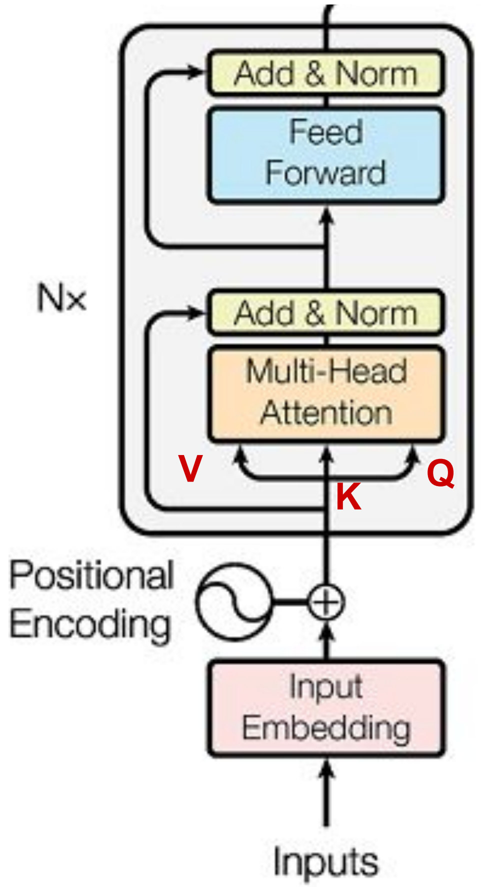
- Embedding of the inputs (N x d) is the same shape as the positional encoding for each position
- MH Attention creates multiple combinations of the input tokens
  - V, K and Q are all the same!
  - Called ***self-attention***
- Feed Forward is some learned function that is applied to each of the hyper-tokens



Vaswani et al. (2017)

Transformer: Encoder

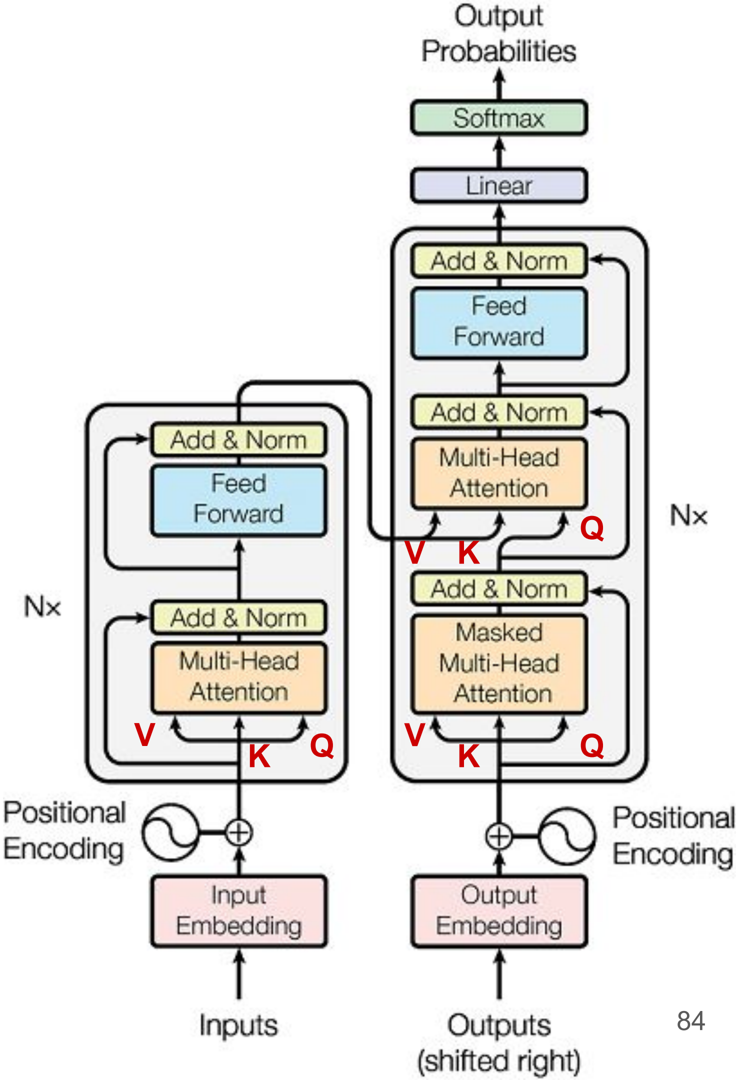
- Skip connections + Normalization: avoid vanishing gradient
- Shape stays the same at each stage
- Stack multiple modules on top of each other (for Vaswani et al., they use N=6)



Vaswani et al. (2017)

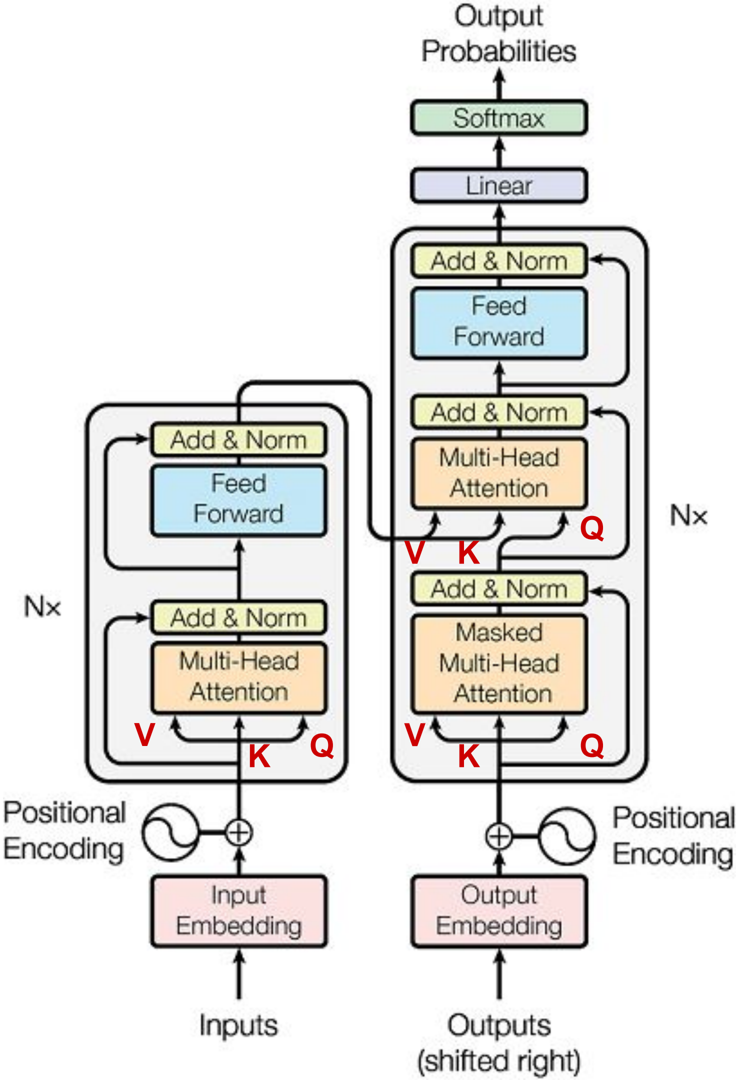
Transformer: Decoder

- MH Attention 1: Hyper-token rep of output sentence
  - V, K and Q
  - Mask avoid “look ahead”
- MH Attention 2: Integrate input
  - V, K from Encoder
  - K from Decoder



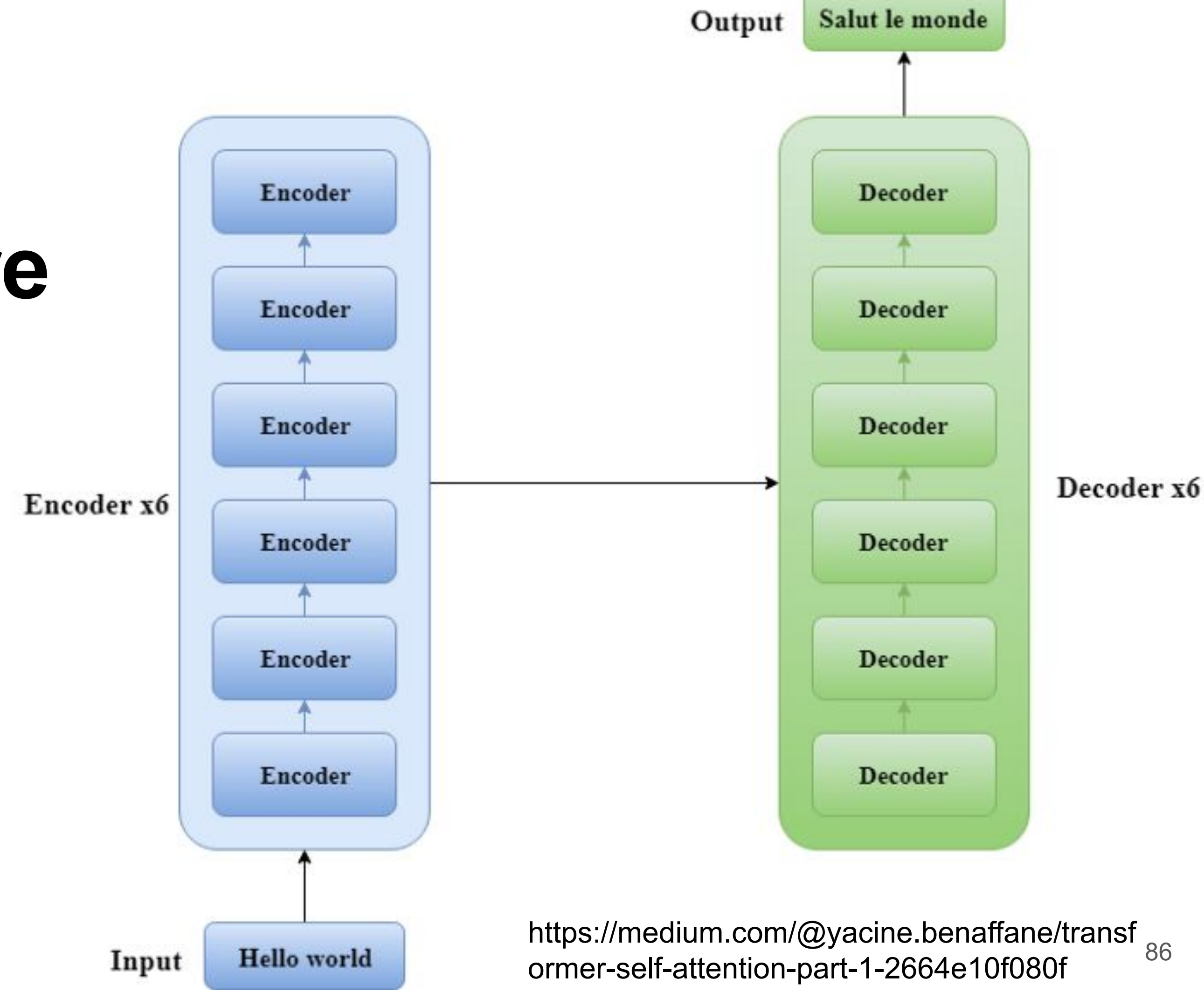
Transformer: Decoder

- Stack multiple modules
- Final stages:
  - Linear transform computes scores for every possible output token
  - Softmax: probability of emitting a given token



Full Architecture

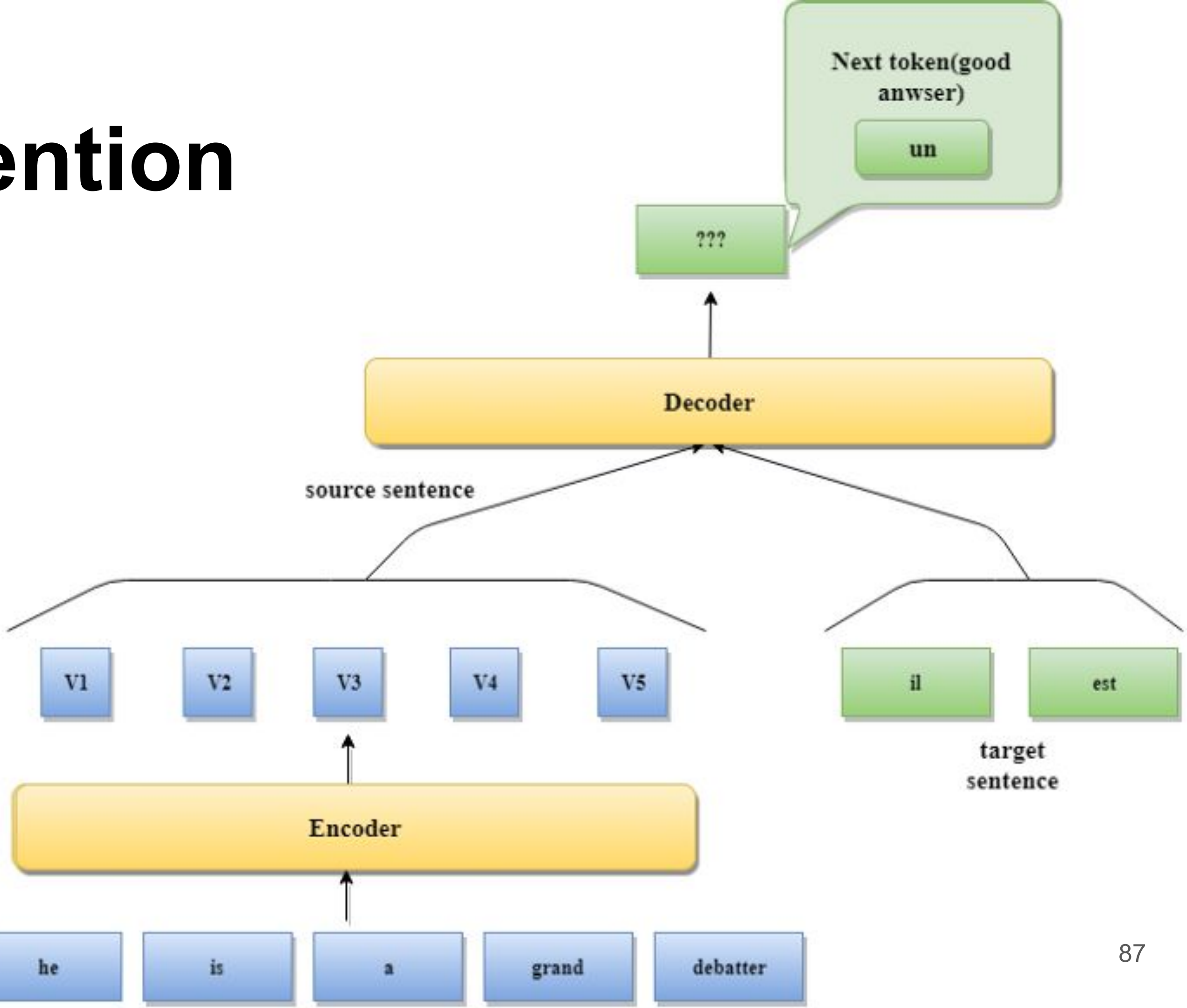
Note: output from the top of the encoder stack is the input to each of the decoder layers



https://medium.com/@yacine.benaffane/transformer-self-attention-part-1-2664e10f080f

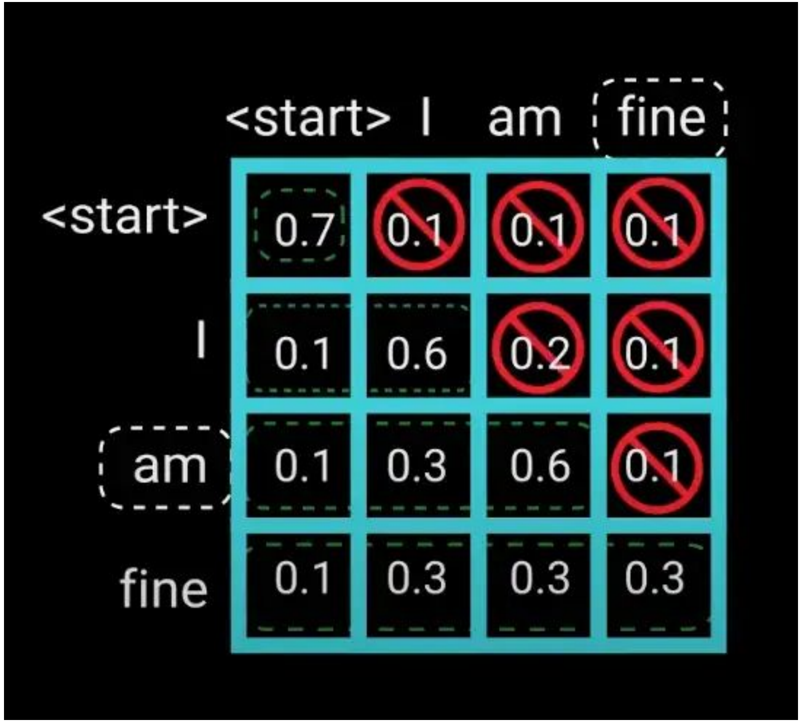
Masked Attention

- Decoder only produces output one token at a time
- For a token at time t, we do not yet know output tokens t+1 …

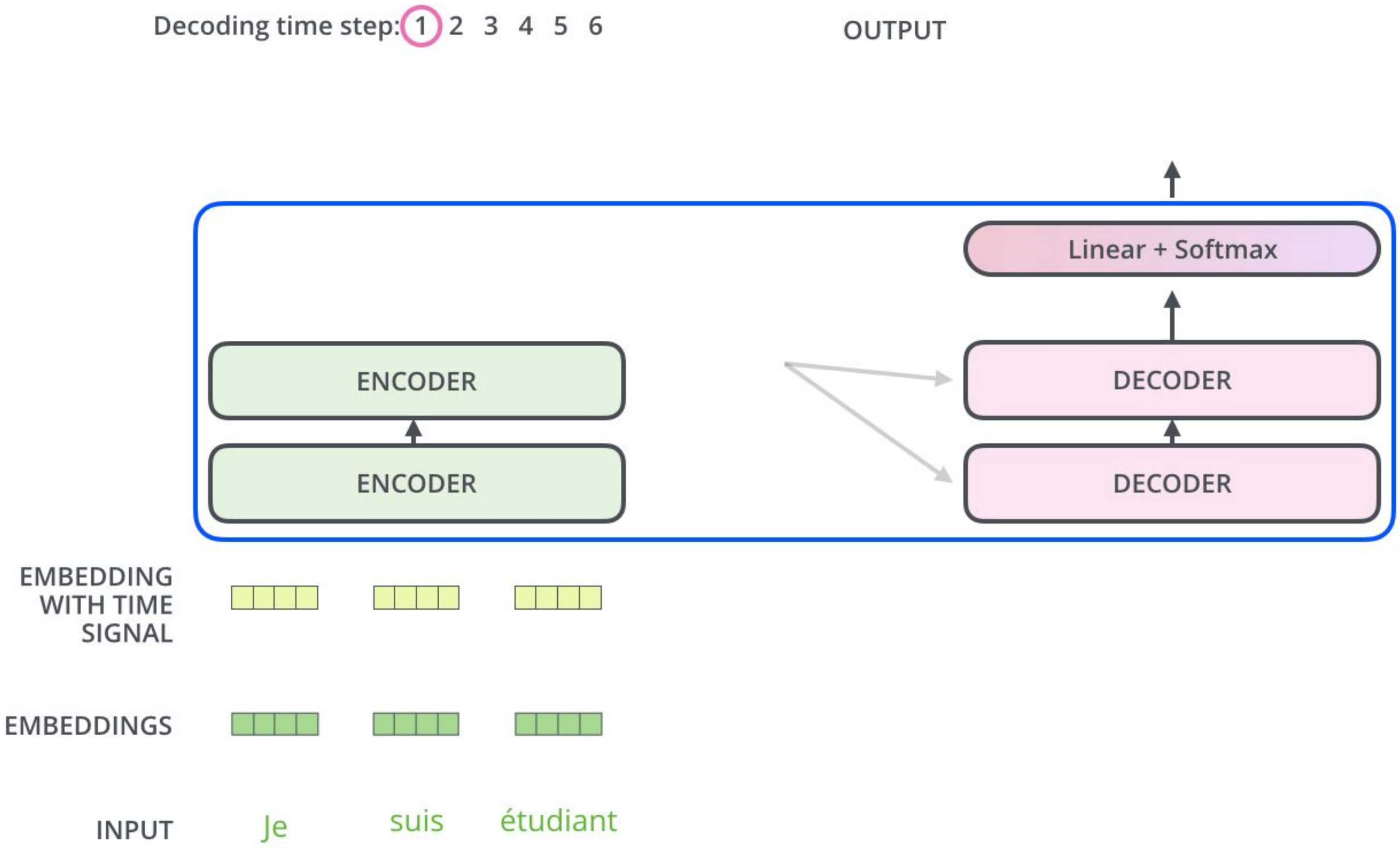


Masked Attention

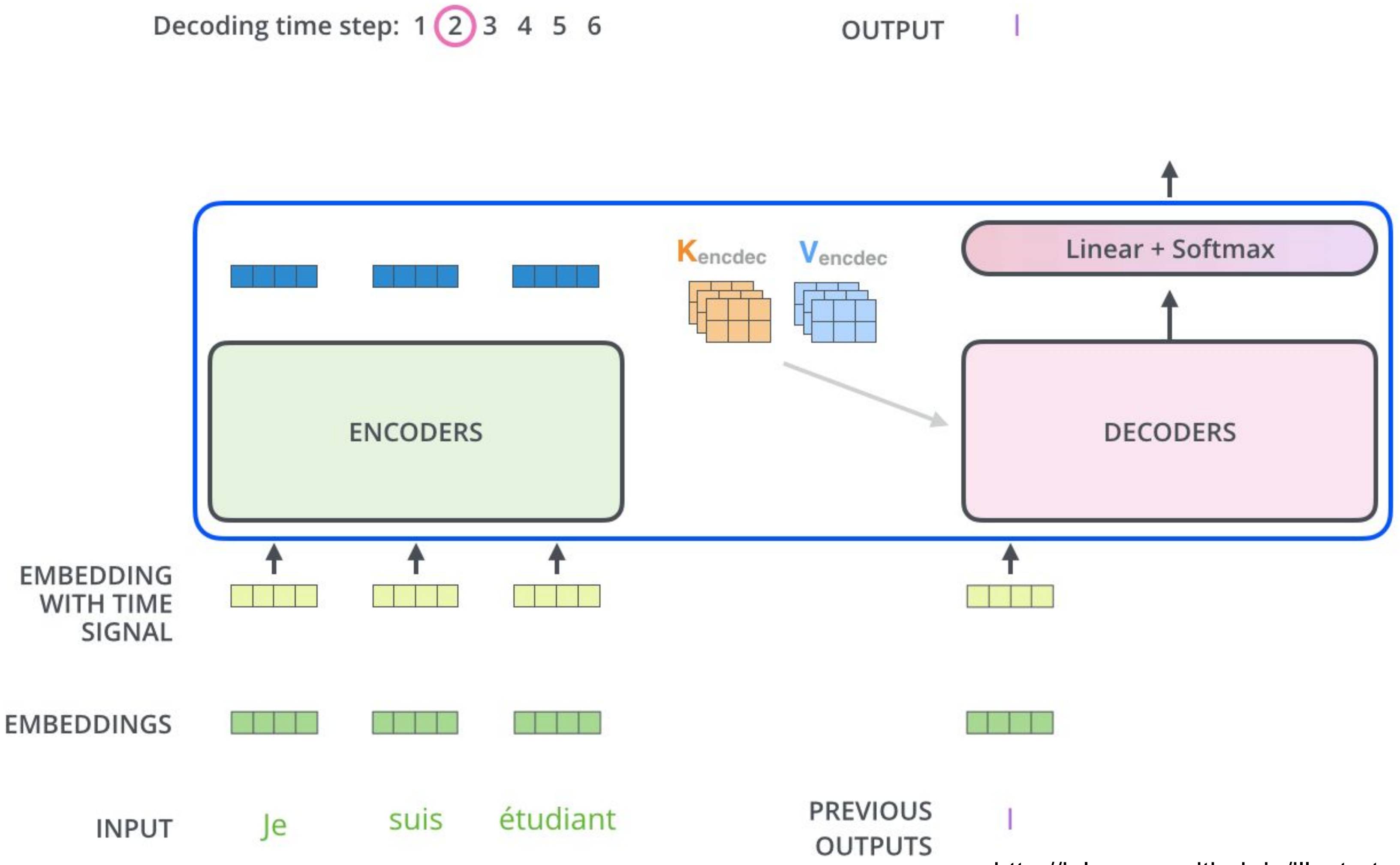
- Alphas for future tokens are set to zero
- Decoder input token “fine” cannot be used as context (the query) while selecting the output token “fine”



https://towardsdatascience.com/illustrated-guide-to-transformers-step-by-step-explanation-f74876522bc0



http://jalammar.github.io/illustrated-transformer/



http://jalammar.github.io/illustrated-transformer/

Training Process

Simple case:

- Training is done with a large corpus of paired sentences/paragraphs/more across two languages
- Cost function: cross-entropy
- Although all of the true output tokens are known ahead of time, **Masked Attention** is still used so that the model does not learn to rely on future information

Training Process

These models are data hungry. Many variations for addressing this with examples from a single language

- Self-supervised pre-training of the encoder model (BERT; Devlin et al., 2019)
- Judging similarity of sentences
- Predicting next sentence/paragraph/other

Also: multilingual training outperforms bilingual models

Uses of Text-Based Transformers

- Language translation
- Generating text given small prompts
- Question answering systems
- Text summarization
- …

Transformers

- Transformers are one class of generative models
- Fundamentally, they are about sampling from a conditional distribution $p(y|x)$, where x and y are composed of smaller, similarly-structured objects
    - Objects have some spatial or temporal relationship
    - Often gridded

Transformers: Extensions

x and y can come from different domains

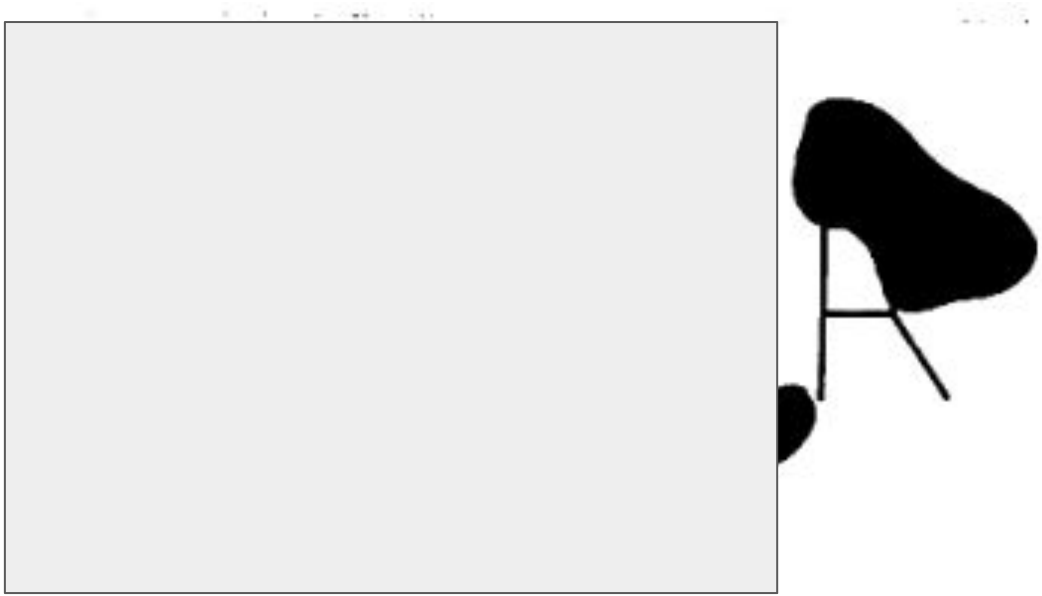
- Text to image
- Text to movie
- Image to text
- Image to semantic segmentation
- Image to repaired image

Gentle? Introduction to Attention and Transformers III

Andrew H. Fagg
Symbiotic Computing Laboratory
University of Oklahoma

A Challenge

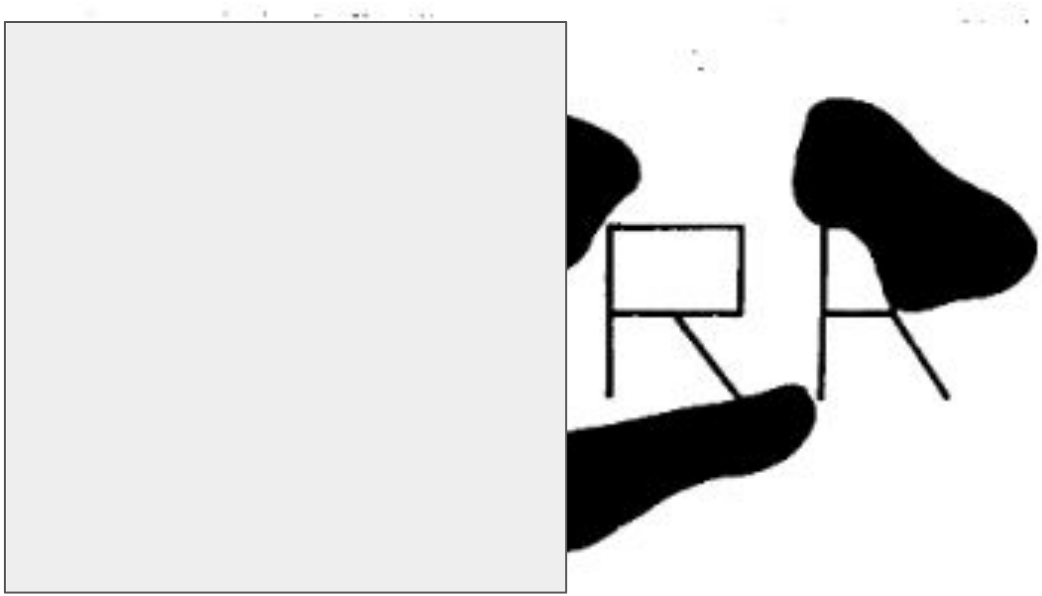
What is this letter?



McClelland & Rumelhart (1981)

A Challenge

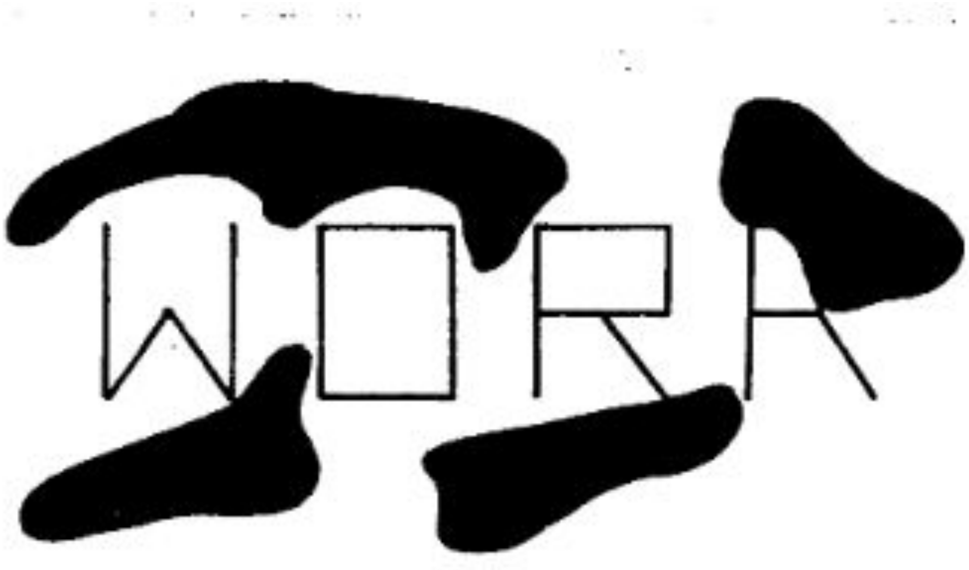
What is this letter?



McClelland & Rumelhart (1981)

A Challenge

What is this letter?

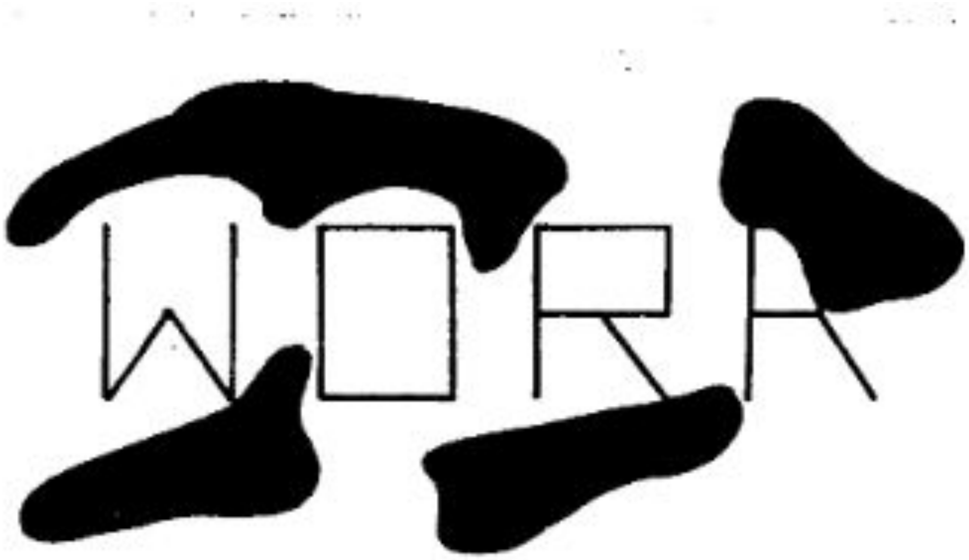


McClelland & Rumelhart (1981)

A Challenge

What is this letter?

We need the context of the entire sequence of letters to properly interpret the last letter



McClelland & Rumelhart (1981)

Review: Our Data Context

- Most general sense: we are working with some regular sampling of ‘objects,’ each of which is described by some feature vector
  - Sequences of words/tokens
  - Sequences of images or parts of images
- Our models must be able to reason (potentially) about all elements of the sequence as it is producing its predictions

I am going to continue to use the word ***token*** to mean the representation of a single object

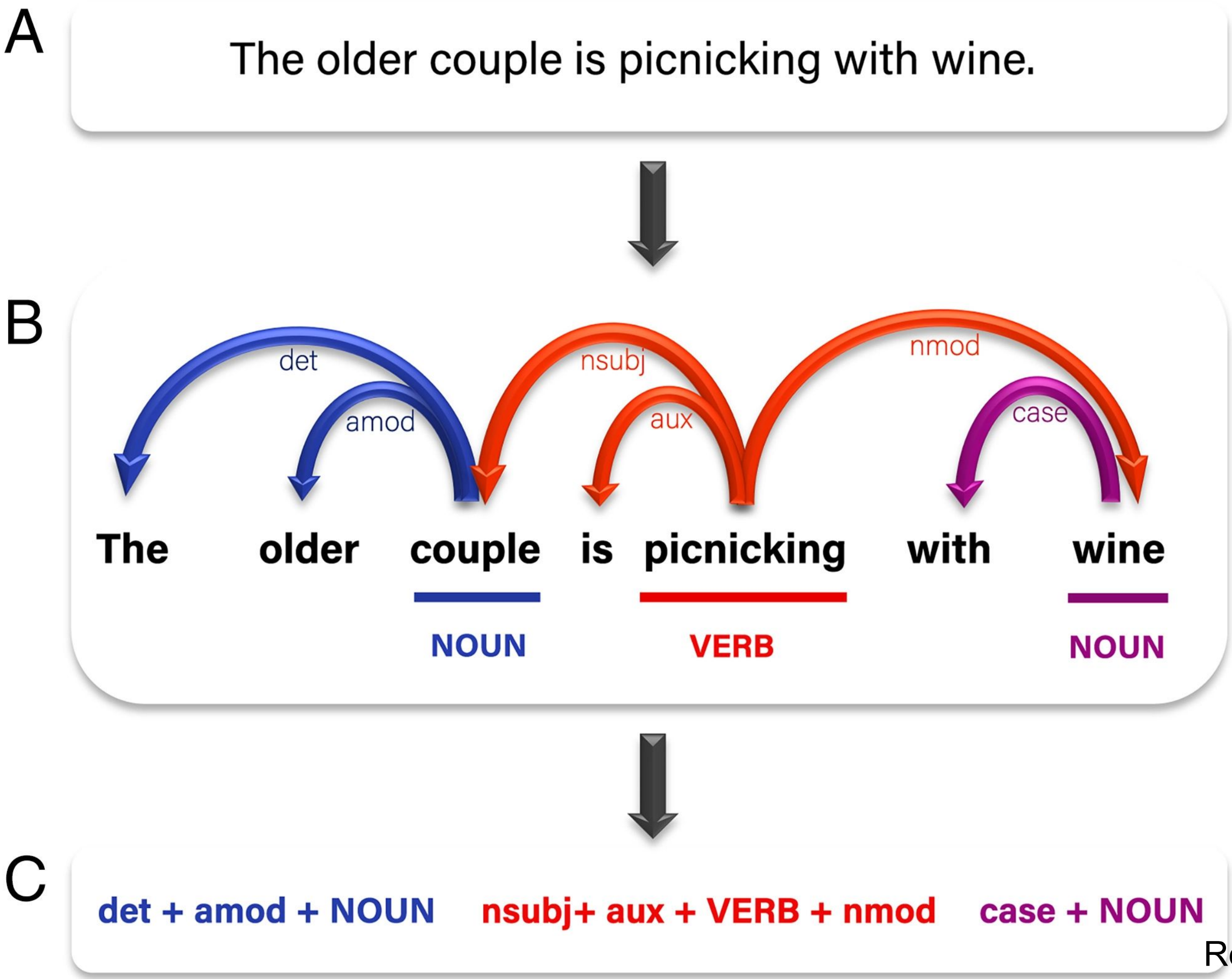
Attention

While the model is processing one token in this sequence, it often must use the context of the other tokens in the sequence to properly interpret it

- The word ‘it’ in the middle of a sentence is ambiguous
- An eye-like shape in an image can be interpreted in different ways
- An updraft has different meanings, depending on other, nearby, high-level features

Attention

- Attention enables the model to bring in information from other parts of the sequence
- It is selective as to what information is used
- For sentences, can think of the model as ‘decorating’ a word with richer, context-specific information
- Multi-headed attention provides multiple types of decoration for a single input element



Rezaii et al. (2022)

Position Embedding

- Judging how one token should decorate another token depends on their relative positions
- Positional embedding re-encodes the index position to a vector. Key properties:
  - All elements of the embedding fall within +/- 1
  - Difference of position between two tokens is a linear operation

Transformers

Transformers are generative models: they produce samples from some conditional distribution $p(y|x)$

- A sentence in German (y) given a sentence in English (x)
- A sentence given a previous sentence
- The completion of an incomplete or corrupted image
- The generation of a sequence of images given a sequence of non-image variables

Transformers

Key property: we have a sequence of input and output tokens. As we are sampling from $p(y|x)$:

- The tokens that are generated must be consistent with one another
- Hard to do all at once with long sentences or large images
- Transformers (and RNNs) solve this problem by generating one token at a time

Transformers

Transformers solve this problem by generating one token at a time

- With this type of model, what has already been generated is important context for the next token to generate
- Transformers use a combination of the encoded input sequence and the encoded outputs up to time step ***t*** to decide what the next token(s) should be

Plan for Today

Transformers for 2D data

- Image recognition (encoder only)
- Image generator (encoder + decoder)

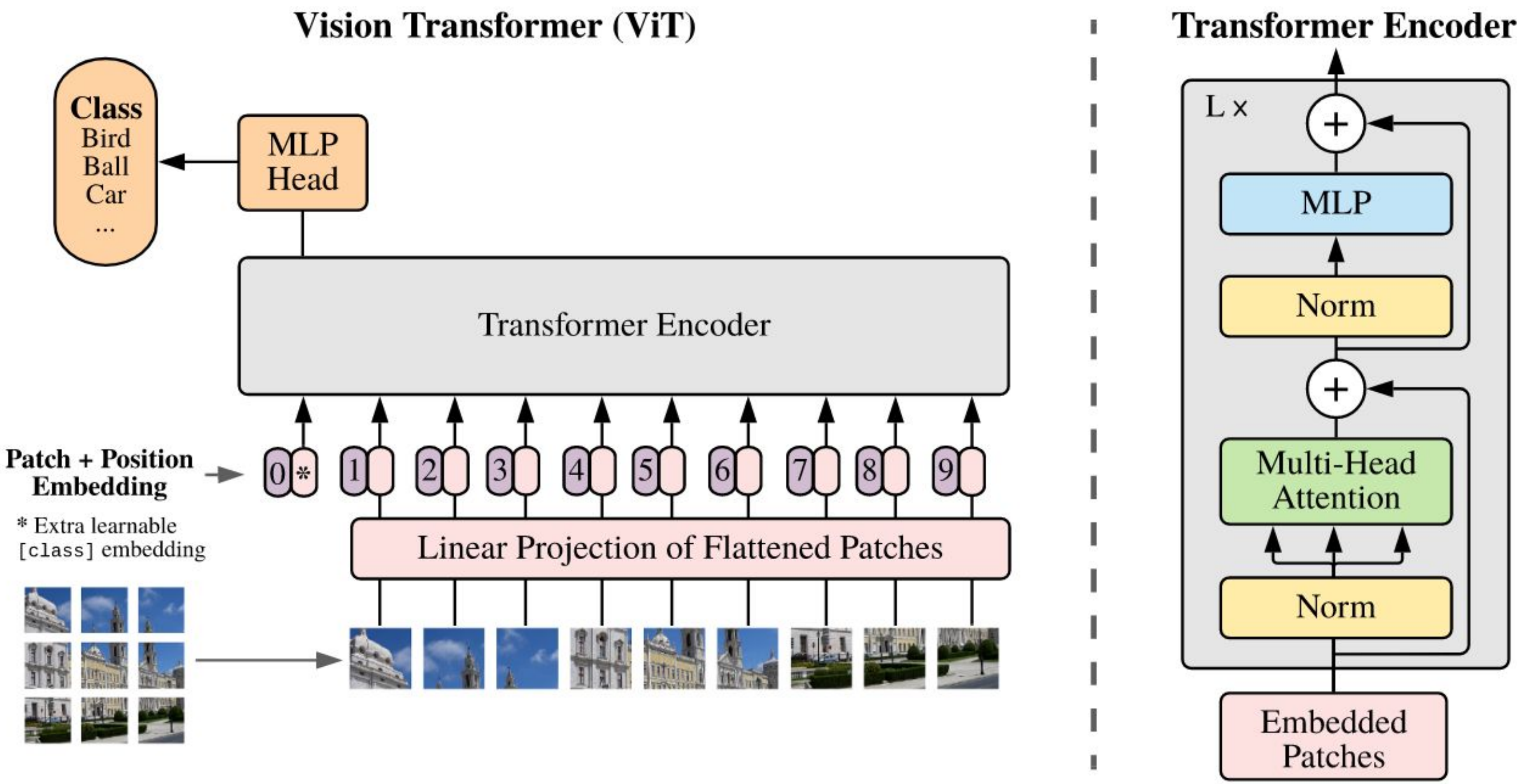
Image Recognition with Transformers

Ramachandran et al. (2019):

- Spatial convolutions can only integrate information from small neighborhoods
- But want to recognize spatial details that potentially cover the entire image
- Can Attention be used to replace convolutions?

Attention with Images

- Want to be able to integrate information from all corners of the image, but to do so in a computationally feasible way
- Proposal:
  - Cut the input image into a grid of image patches
  - The individual ‘tokens’ for attention are the image patches or are derived from them
  - Within a patch: processing is done with a fully-connected layer
  - Across patches: Attention layers

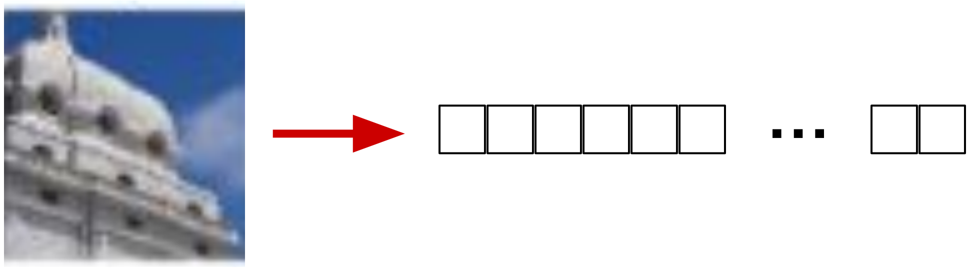


Dosovitskiy, et al. (2021)

Within a Patch

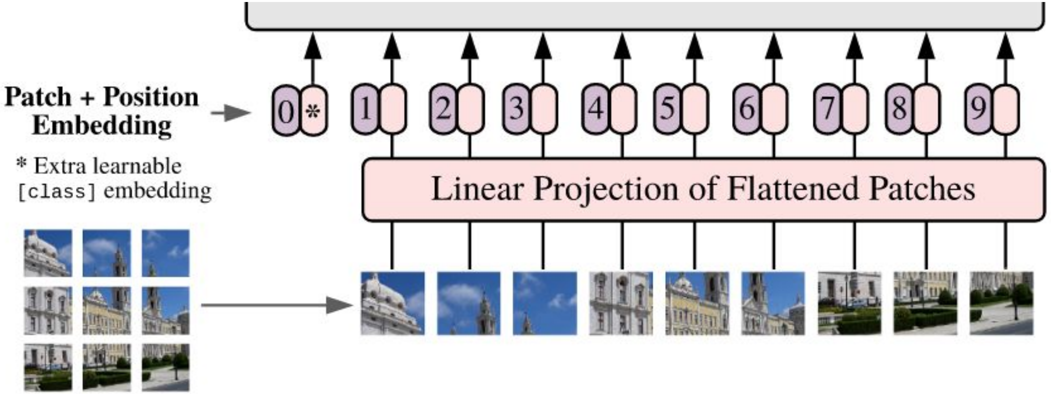
Extreme approach: pixels in a patch are flattened into an embedding vector

(d = #pixels x #channels):

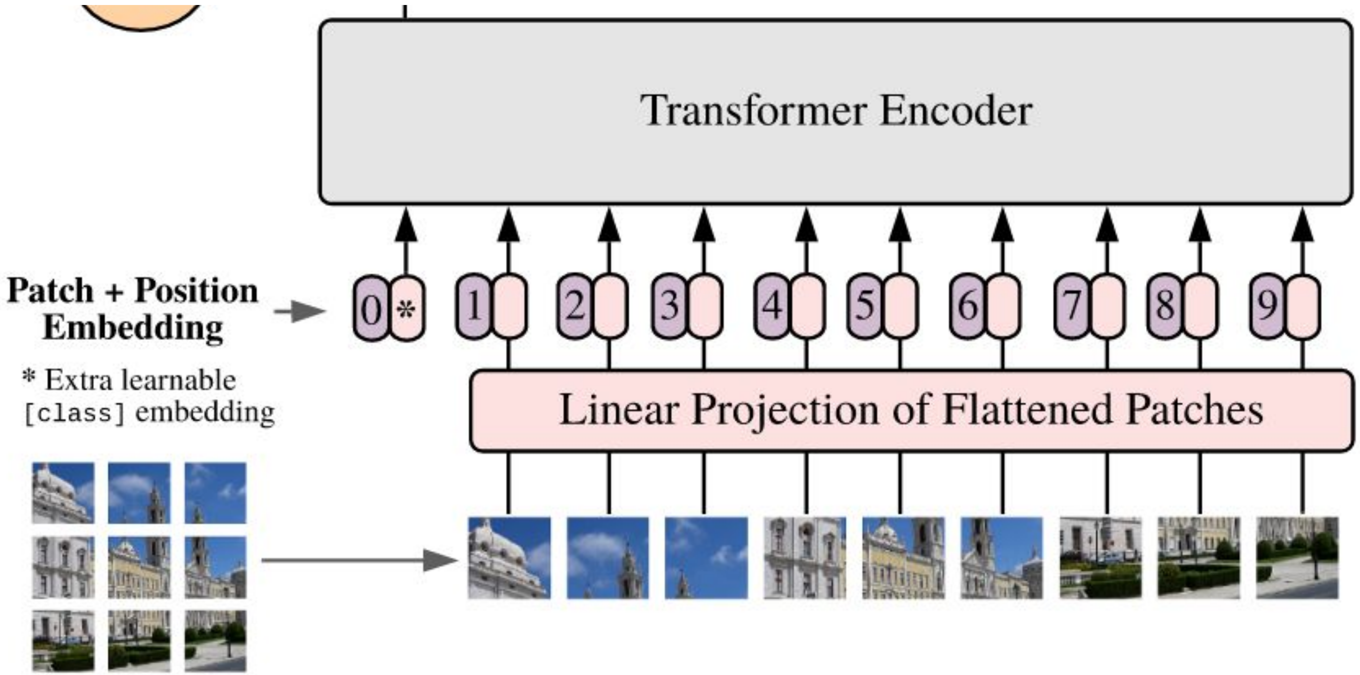


Pre-Processing

- Patches can undergo some transformation beforehand
- Each patch is transformed in the same way
- Position embedding captures 2D position of each patch in the original image



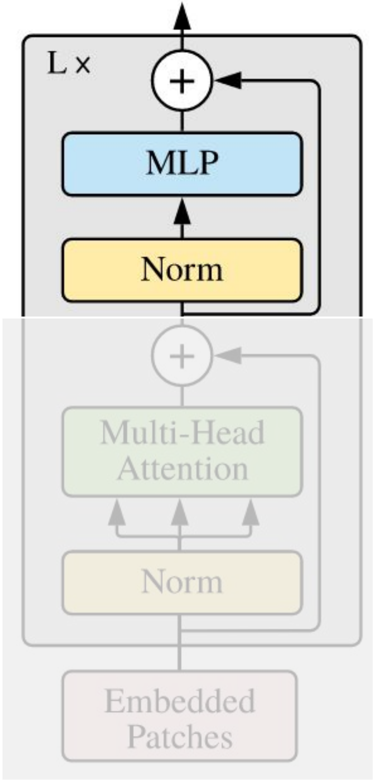
Transformer Encoder



Transformer Encoder Module

Latter stage:

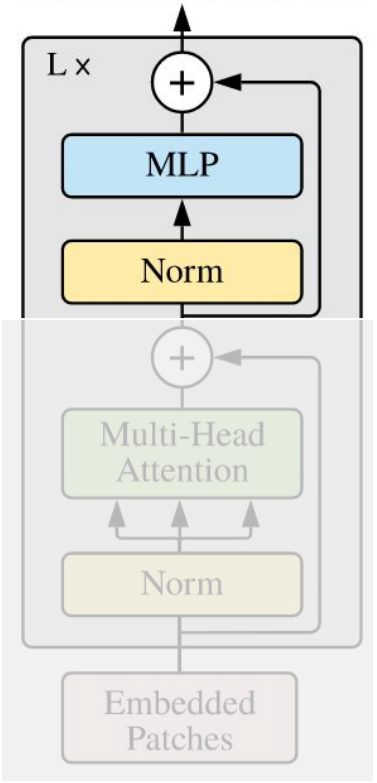
- Transforms a single patch into an output patch of the same dimensionality through a fully-connected layer (MLP)
- Each pixel in the patch influences every other pixel in the output patch



Transformer Encoder Module

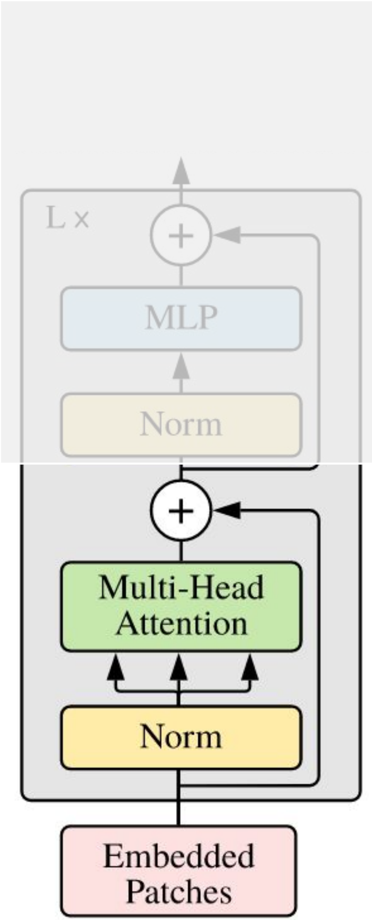
- A fully connected layer is far more expressive than a convolutional layer
- This comes at the cost of more parameters
- But, if the patches are small, then this is not a huge increase over convolution

Note: each of the patches are processed with same fully connected network



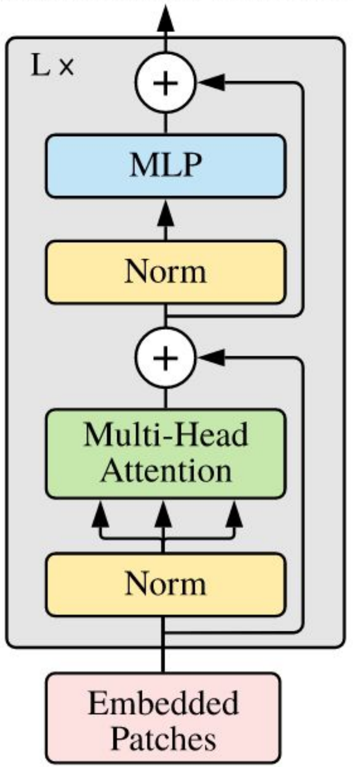
Attention Across Patches

- Multi-Headed Self Attention: allows the interpretation of one image patch to be influenced by other patches
- This influence is implemented as a weighted blend of the patch with other patches (these are our attentional alphas, again!)
- Skip connection ensures that the current patch is maintained to some degree



Transformer Encoder Modules

- Output shape is identical to the input shape
- Multi-Headed Attention: convolution-like operation, but with a reach across the entire image
- MLP: dense processing at a pixel scale within each patch separately



Attention vs Convolution

- Convolution only allows a local neighborhood of pixels to influence the corresponding output pixel
    - This transformation is the same for all offsets
- Attention potentially allows all patches to influence all output patches
    - This influence varies depending on the match of the key/query match

Note: there is a scale difference here (pixels vs patches)

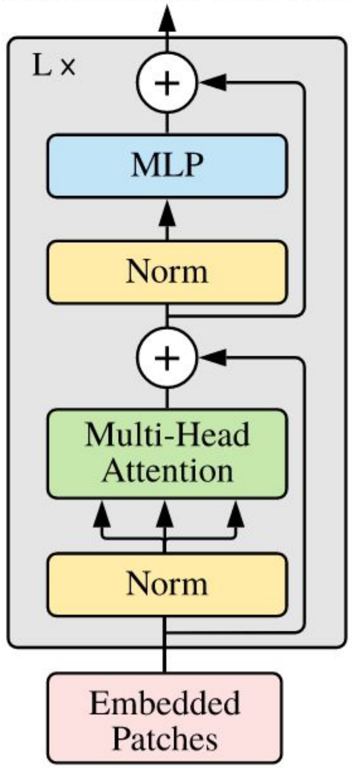
Attention vs Convolution

- Pure Attention approach requires that all patches can attend to all other patches
  - This requires N^2 comparisons & a lot of parameters
- Can reduce the complexity of the model by only allowing comparisons with a limited neighborhood of patches
- Think of this as a compromise between pure Attention and convolution

Attention vs Convolution

Stacking multiple Attention modules on top of each other

- Allows for building more abstract representations the higher we go
- Can compensate for limited Attention
  - One patch may not be able to attend to another patch directly, but it can do so across multiple layers of modules



Positional Embedding

Multiple options for implementation:

- None
- 1 D: absolute position of the patches
- 2 D: absolute row/col position of the patches (learned)
- 2 D-relative: relative row/col of two patches (learned)

Empirically: positional embedding is helpful over None

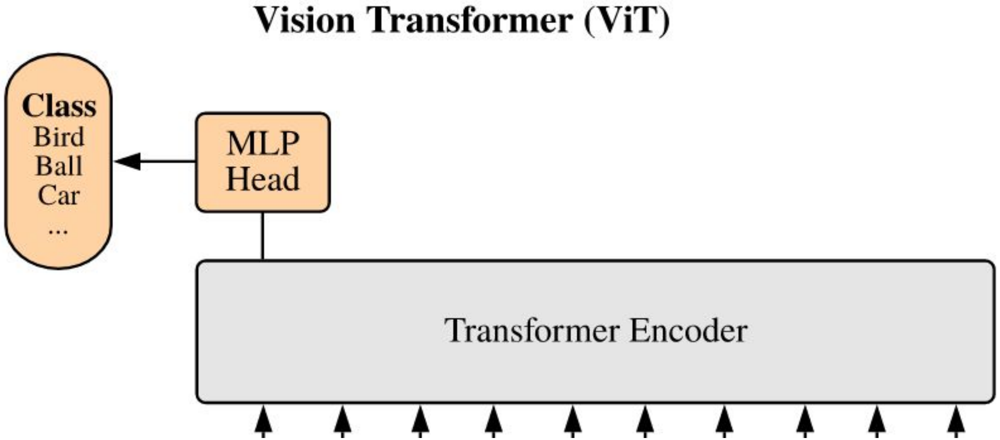


Image Recognition

Final stage:

- Combine evidence across the patches
- Compute class probabilities (via softmax)

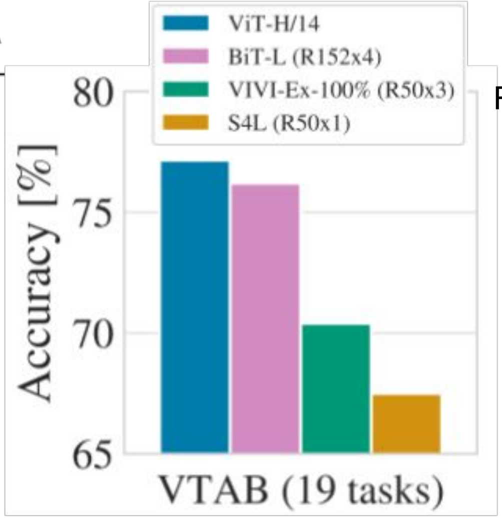
https://ai.googleblog.com/2020/12/transformers-for-image-recognition-at.html

Training details

- Typical loss: cross-entropy
- These methods are very data hungry
  - ImageNet is really small by these standards
  - A lot of work has gone into using bigger data sets for training or for pre-training the models

Results: Collection of Image Recog Tasks

	ViT-H	Previous SOTA
ImageNet	88.55	88.5
ImageNet-ReaL	90.72	90.55
Cifar-10	99.50	99.37
Cifar-100	94.55	93.51
Pets	97.56	96.62
Flowers	99.68	99.63



Visual Transformer

ResNet

ResNet (co-training)

Supervised + Semi-Supervised Learning

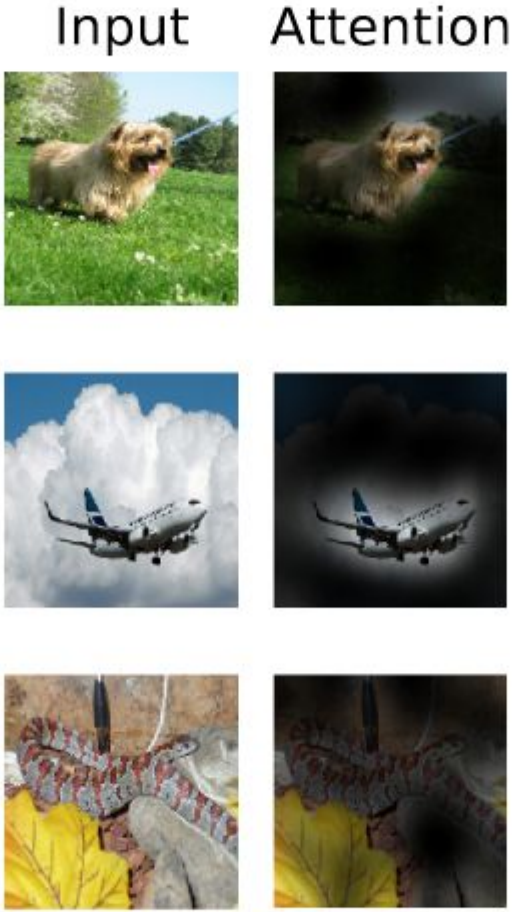
Requires fewer training compute cycles than SOTA CNNs

Dosovitskiy (2021), Houlsby & Weissenborn (2020)

Attention XAI

Aggregate Attention across multiple layers

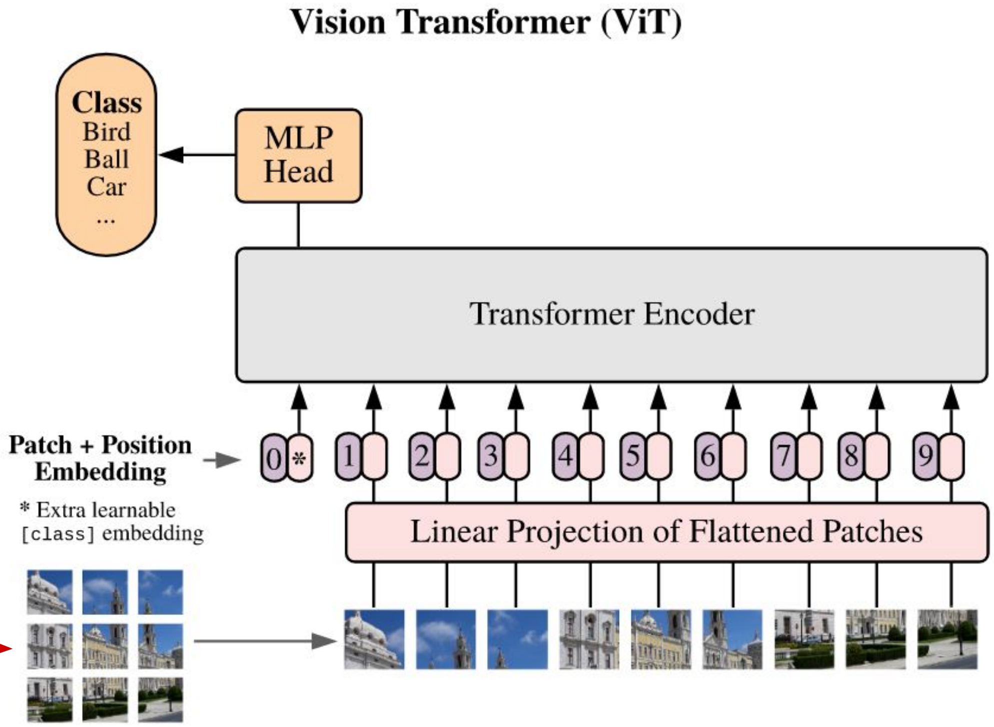
- Approach feels a lot like Grad-CAM



Hybrid CNN / Attention Approaches

- Early layers are just about learning primitive feature detectors
- Convolution can do this with fewer parameters than Attention
- Ramachandran et al. (2019): Use a CNN as a pre-processing step to the Attention layers

Hybrid CNN / Attention Approaches



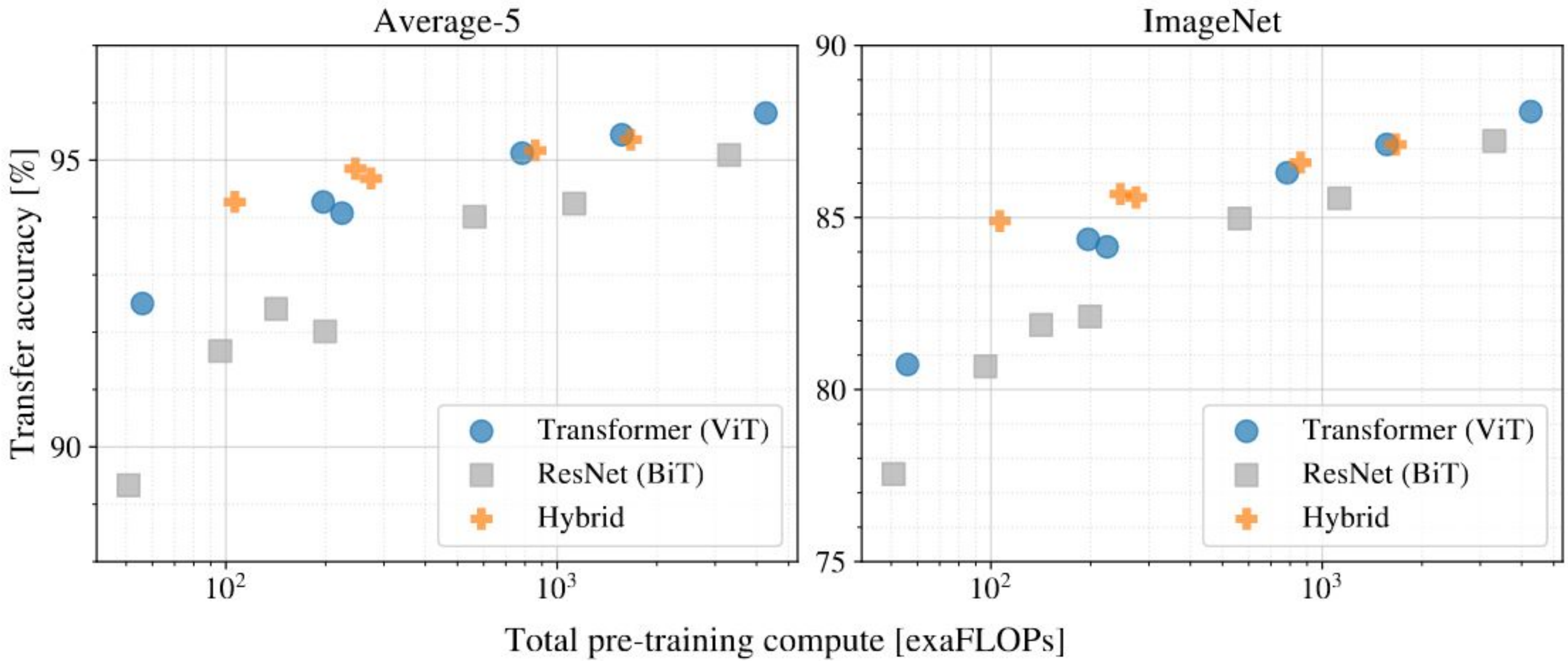
Multi-Layer CNN

Hybrid CNN / Attention Approaches

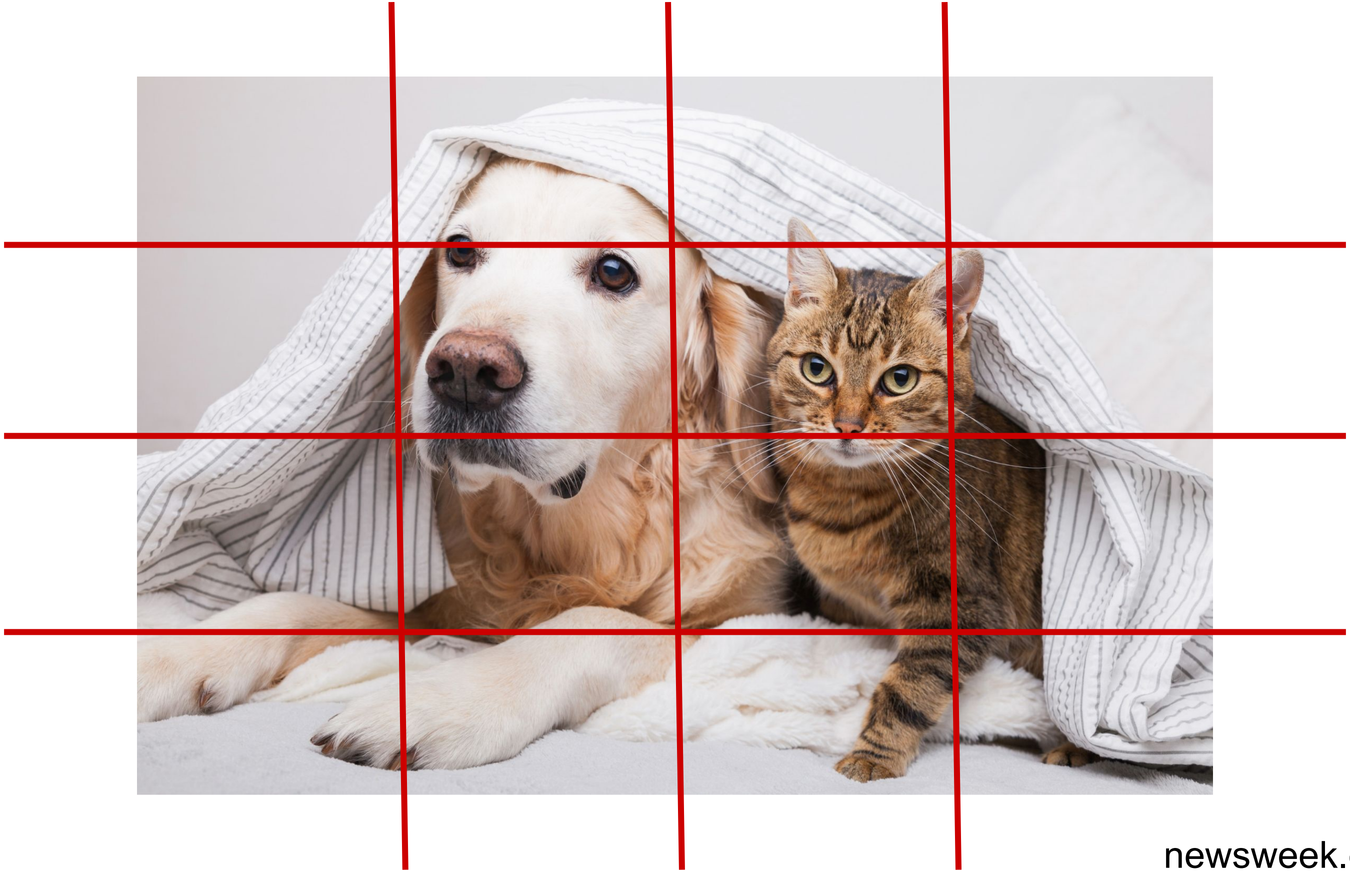
Conv Groups	Attention Groups	FLOPS (B)	Params (M)	Top-1 Acc. (%)
-	1, 2, 3, 4	7.0	18.0	80.2
1	2, 3, 4	7.3	18.1	**80.7**
1, 2	3, 4	7.5	18.5	**80.7**
1, 2, 3	4	8.0	20.8	80.2
1, 2, 3, 4	-	8.2	25.6	79.5
2, 3, 4	1	7.9	25.5	79.7
3, 4	1, 2	7.8	25.0	79.6
4	1, 2, 3	7.2	22.7	79.9

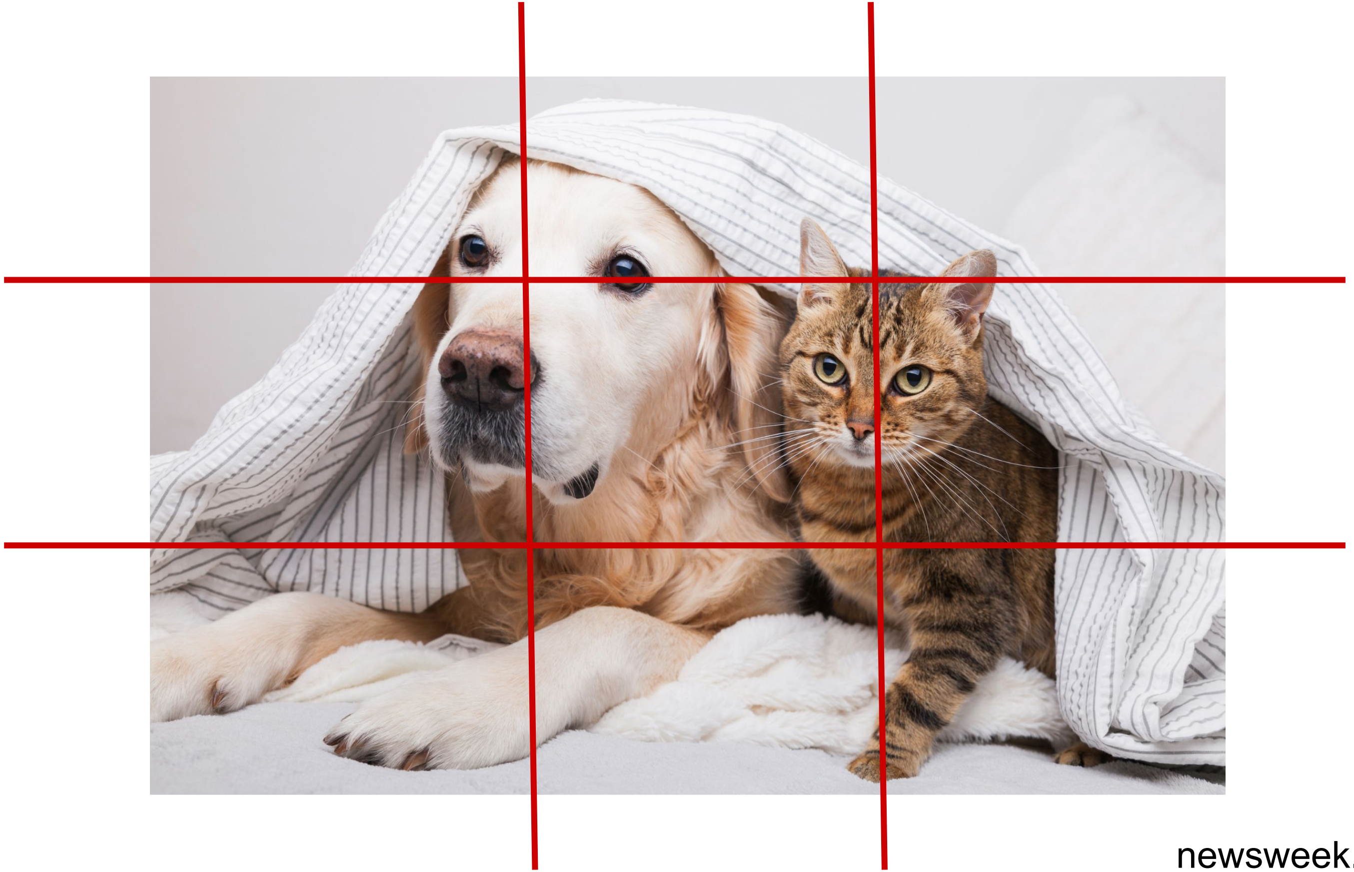
Ramachandran et al. (2019)

Transfer Learning Advantages

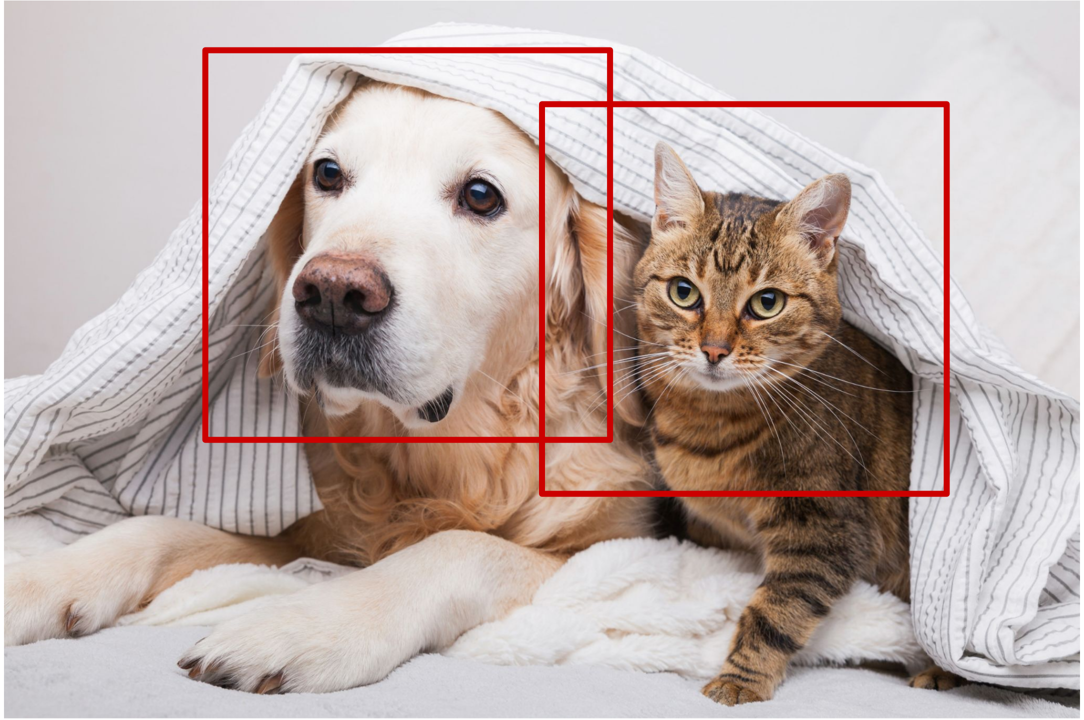


Is a Regular Gridding Appropriate?





Region of Interest Preprocessing



newsweek.com

Region of Interest Preprocessing

- Identify RoIs
  - These become our image patches
- CNN to extract high-level representation of each patch
  - Class label?
  - Embedding vector?
- Attention to construct higher-level representations of the patches

E.g.: “A dog sitting next to a cat”

Generative Models

Goal: produce images

- Conditioned on some input
  - Could be another image
- Generated image should be self-consistent

Generative Models

Can be used for:

- Image clean-up
- Inpainting / Outpainting
- Upsampling
- Production of (fake) in-distribution samples for training other models
- Text to image
- Prediction of future video frames
- …

Transformer Image Decoder

- Producing a full, self-consistent image in one shot is challenging (pdf is high-dimensional and complex)
- Transformer approach:
  - Produce one piece of the image at a time
  - Can then condition the next piece on the pieces that have already been generated
  - Easier to produce self-consistent images

Transformer Image Decoder

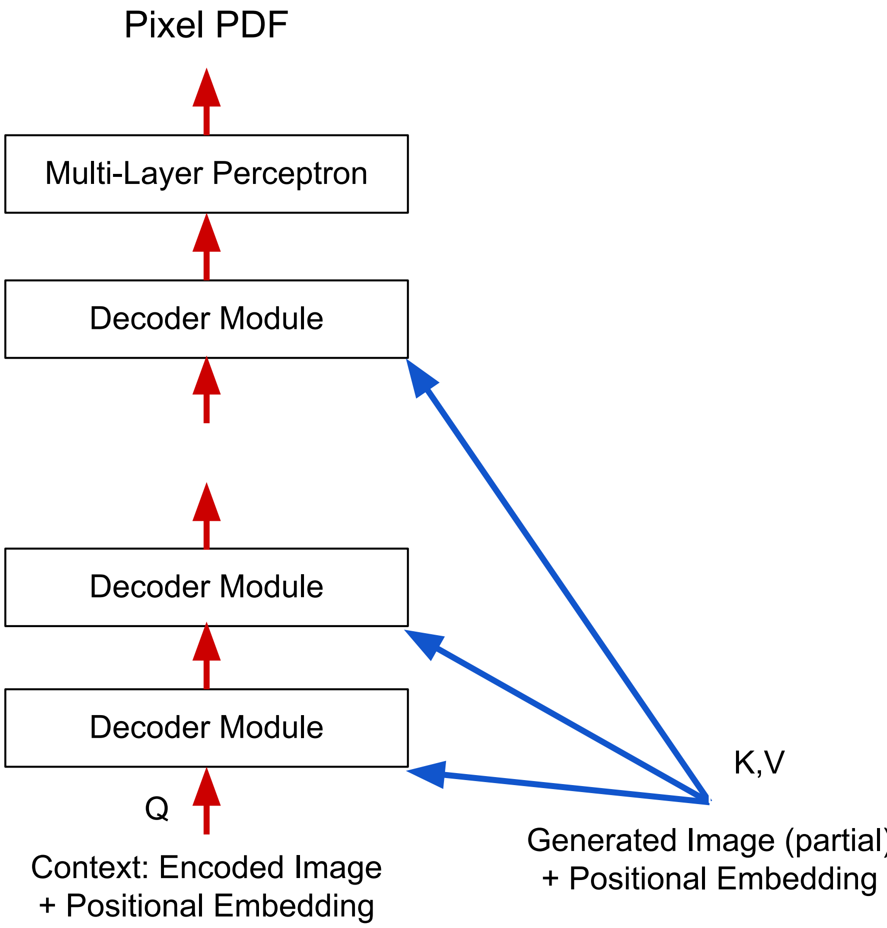
Parmar et al. (2018):

- Following PixelRNN architecture (van den Oord, 2016): produce pixels one at a time
- Pixels are sampled from a distribution that is conditioned on some external context + the pixels generated so far
- External context: from an **image encoder**, text encoder or other source

Full Decoder

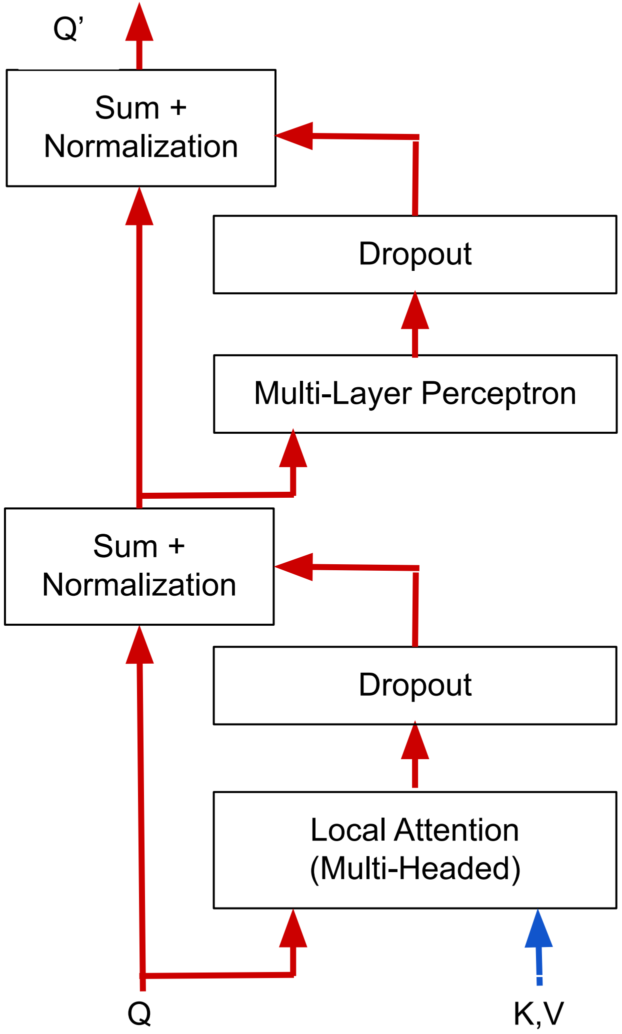
Similar in structure to our text decoders. But:

- Q: Encoder pixel
- K, V: Partially generated image



Decoder Modules

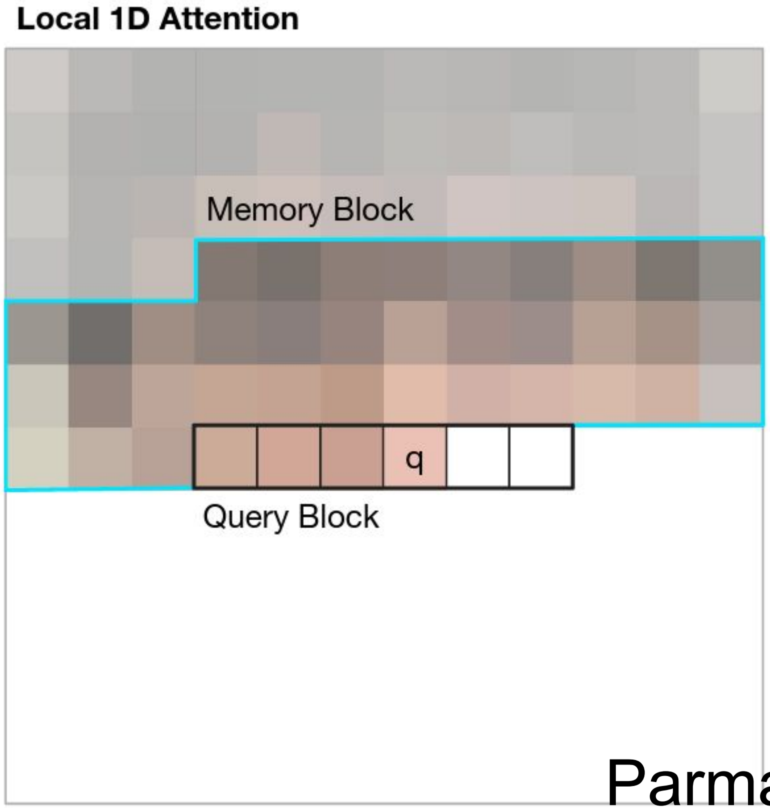
- Similar in structure to our text decoders. But:
  - Q: Encoder pixel
  - K, V: Partially generated image
- Local Attention: only attend to a subset of pixels
- Skip connections



Local Attention

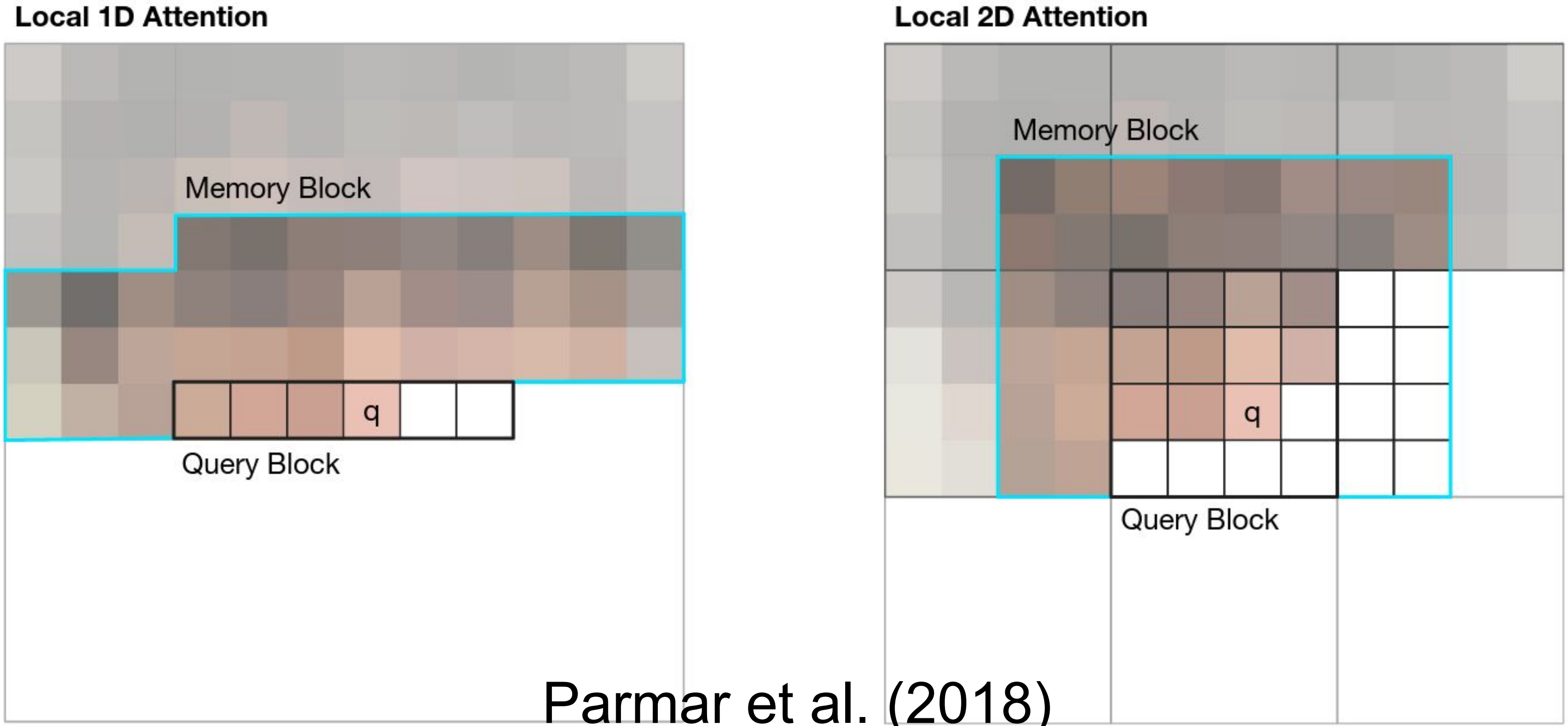
- **Masked Attention**: attend only to pixels that have already been generated so far
- However, it is not feasible to allow a pixel to be able to attend to all other generated pixels
- **Local Attention**: further restrict attention to some local neighborhood
- Two varieties: 1D and 2D Attention

Local Attention



Parmar et al. (2018)

Local Attention



Parmar et al. (2018)

Decoder Output

- As a function of the contextual input and what pixel values have already been produced, network outputs a representation of the likelihood over possible pixel values
- Model samples from this distribution & adds the new pixel to what has been generated
- Process is repeated until the full image emerges

Decoder Output

Two possibilities for representing the output pdf:

- Categorical probability distribution (van den Oord, 2016): represent probability for each pixel value combination (3x256 parameters/pixel)
- Discretized mixture of logistics (Salimans et al., 2017): Gaussian-style mixture distribution (100 parameters/pixel)

Training

- For the results today:
  - Inputs: a modified/corrupted version of an image + (optionally) a class label
  - Outputs: the original image
- Maximize likelihood of each pixel variable:

$$LL = \sum_{i=0}^{3\times r\times c-1} \log p(x_i|x_0 ... x_{i-1})$$

Image Completion

Input: partial image

Output: generated image

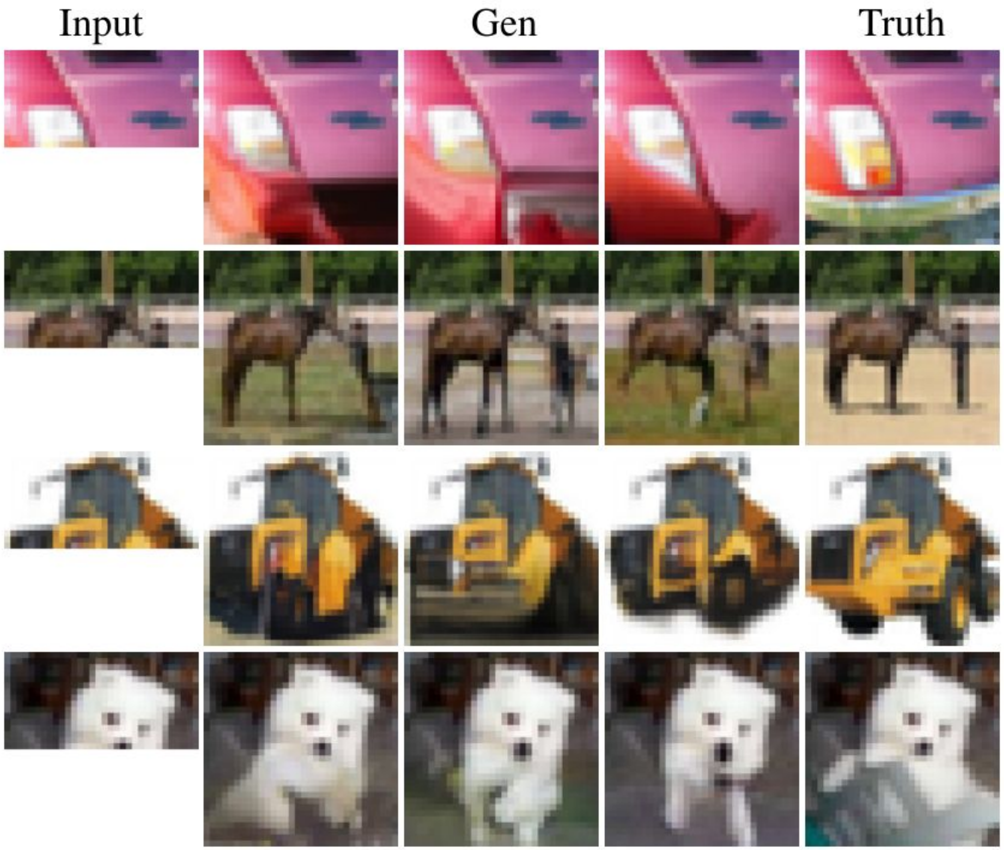


Image Generation

Input: image class

Output: 32x32 image

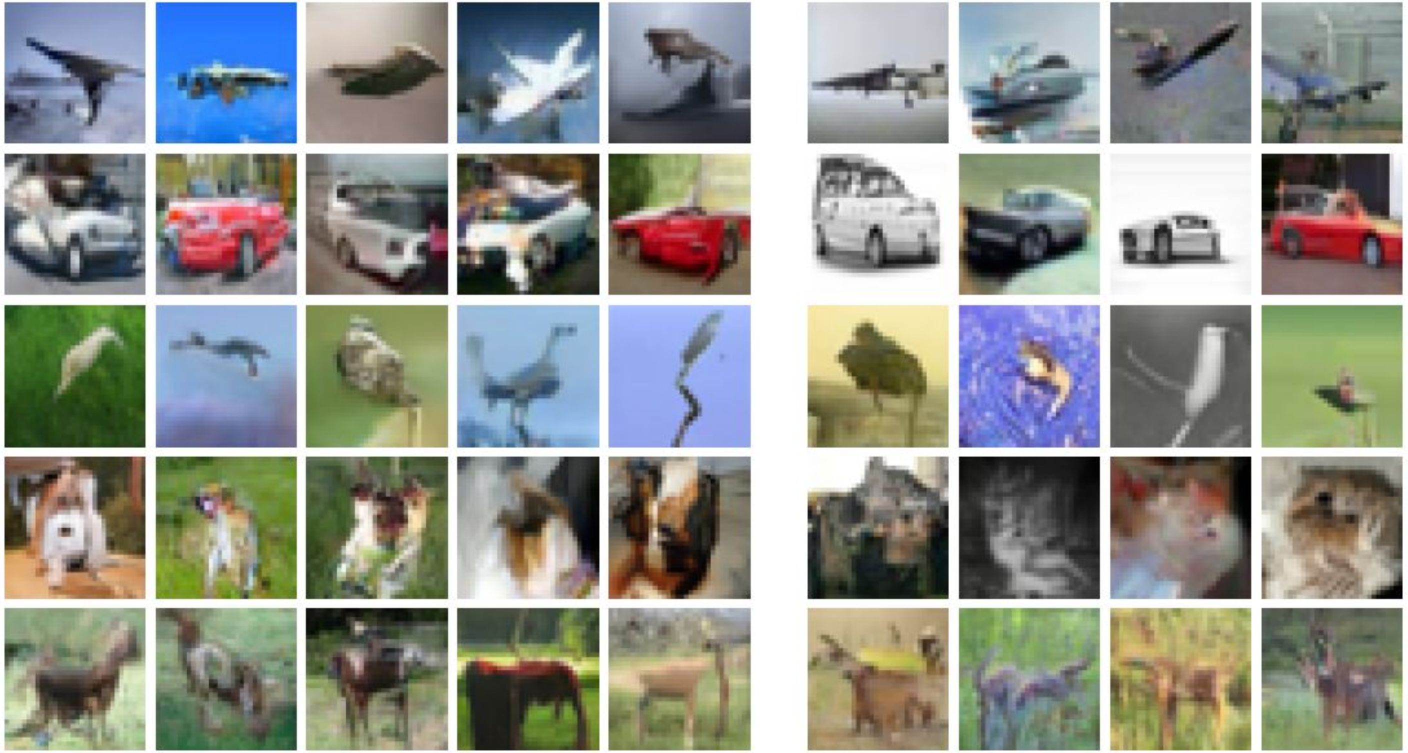
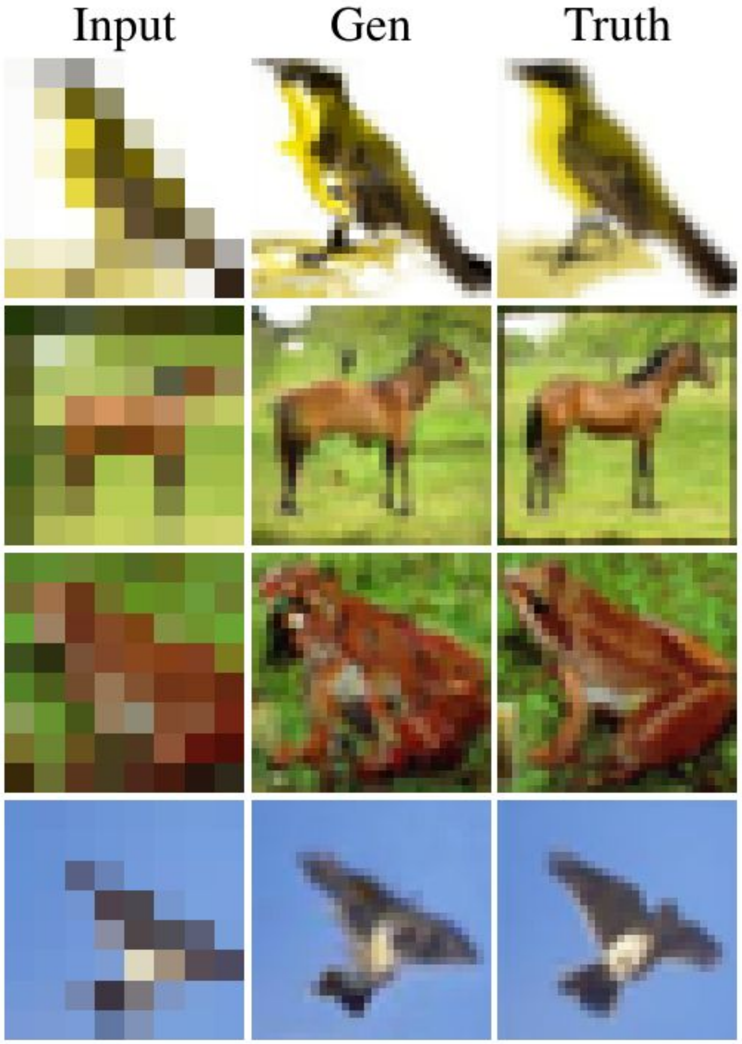


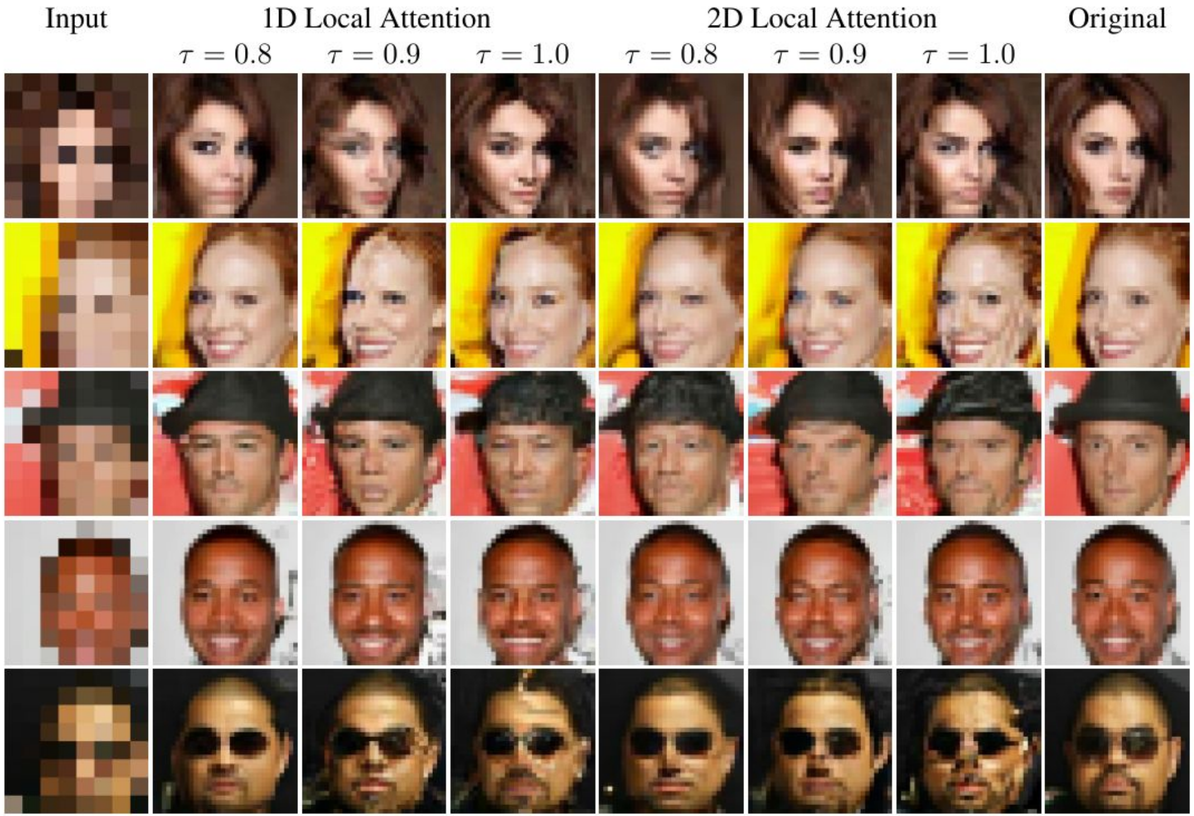
Image Upsampling

- Input: 8x8 image
- Output: 32x32 image



Super-Resolution

Tau controls the entropy in the output pixel selection step



Visual Self-Attention

Self-Attention can supplant convolutional modules

- Even with local Attention, receptive fields are larger
- Can also implement more complex transformations
- The cost is an increased number of parameters and the need for larger training data sets

Decoder

Explicitly represent the pdf of the next pixel color:

- Conditioned on a contextual input and the output pixels that have already been selected
- Makes it easy to take many samples from the distribution
- Cost is that images are generated one pixel (or one channel) at a time

Moving Forward

Encoder side:

- 3D and 4D data: session 4
- Hamid Kamangir (Wednesday)

Decoder side:

- Dealing with higher densities of pixels
- Representing conditional pdfs over larger image regions
- Session 5 tutorial: John Schreck (post AMS)

References

- Vaswani et al. (2017) Attention is All You Need: https://arxiv.org/abs/1706.03762
- Alammar Blog Post: http://jalammar.github.io/illustrated-transformer/
- Kazemnejad Blog Post: https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

References

- Visual Transformers for classification: https://arxiv.org/abs/2010.11929
- Image Transformers (generators): https://arxiv.org/abs/1802.05751