

# Advanced concepts in deep learning



**THOME Nicolas** – Prof. at SORBONNE University  
ISIR Lab, MLIA TEAM



Machine Learning &  
Deep Learning for  
Information Access

# Sequential models

## Up to now ...

- Fully connected neural nets, parameters learning and training tricks
- Convolutional neural networks (ConvNets)
- Generative models
- **Sequential models:** dealing with “ordered sets”
- 2 main state-of-the-art deep architectures
  - Recurrent Neural Networks (RNNs)
  - Attention models & transformers

# Outline

## I. Recurrent Neural Networks (RNNs)

- a) Vanilla RNNs
- b) RNN training
- c) RNN architectures
- d) Applications

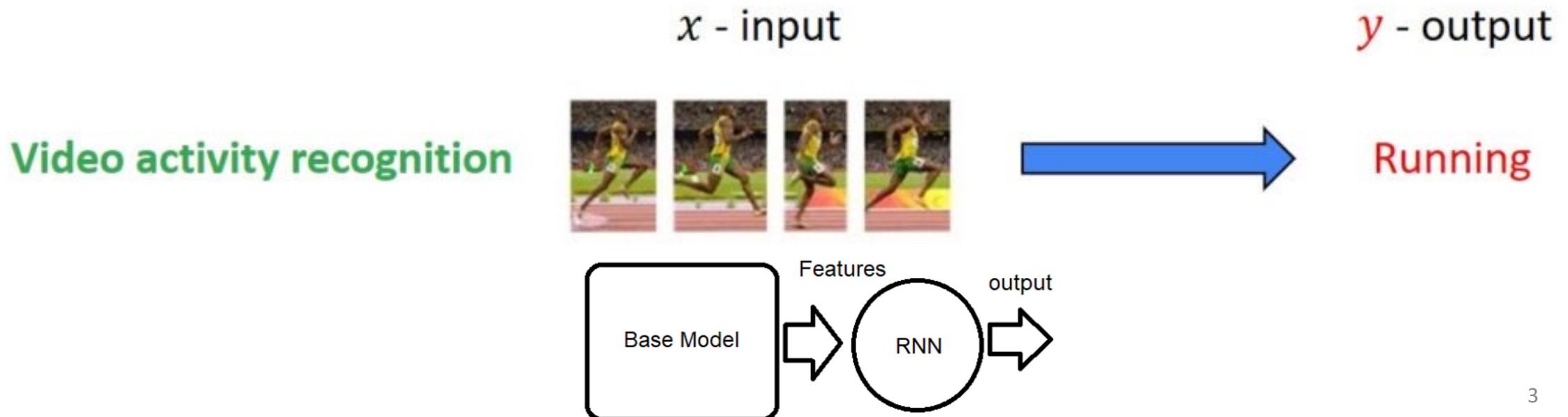
## II. Attention models & transformers

# Recurrent Neural Networks (RNNs)

- Manipulating sequences of “tokens”, *i.e.*, atomic elements
  - Ex tokens: characters/word in NLP

"Hello I love you"  $\longrightarrow$  "Hello", "I", "love", "you"

- Pixels/patches in images, features a time t for time series
- Video: sequences of frames / feature embeddings with CNNs



# Outline

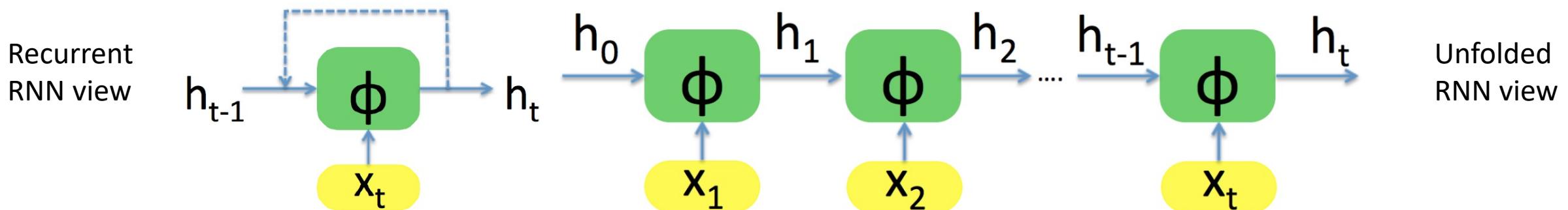
## I. Recurrent Neural Networks (RNNs)

- a) Vanilla RNN model
- b) RNN training
- c) RNN architectures
- d) Applications

## II. Attention models & transformers

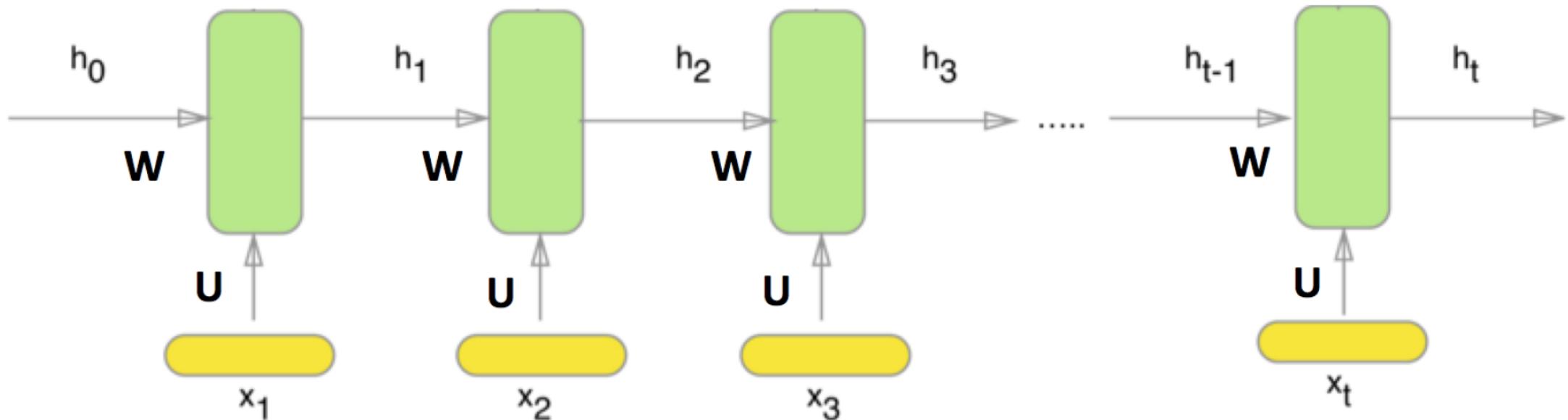
# Recurrent Neural Networks (RNNs) [1]

- Input sequence  $\{x_t\}_{t \in \{1; T\}}$ ,  $x_t \in \mathbb{R}^d$
- Internal RNN state  $\{h_t\}_{t \in \{1; T\}}$ ,  $h_t \in \mathbb{R}^l$
- **RNN Cell:**  $h_t = \phi_t(x_t, h_{t-1})$ 
  - Loop,  $h_t$  depends on current  $x_t$  and previous state  $h_{t-1}$ 
    - $h_t$ : **memory of the network**  $\Leftrightarrow$  **history up to time t**
  - In RNNs, function  $\phi_t = \phi$  shared across time



# Recurrent Neural Networks (RNNs)

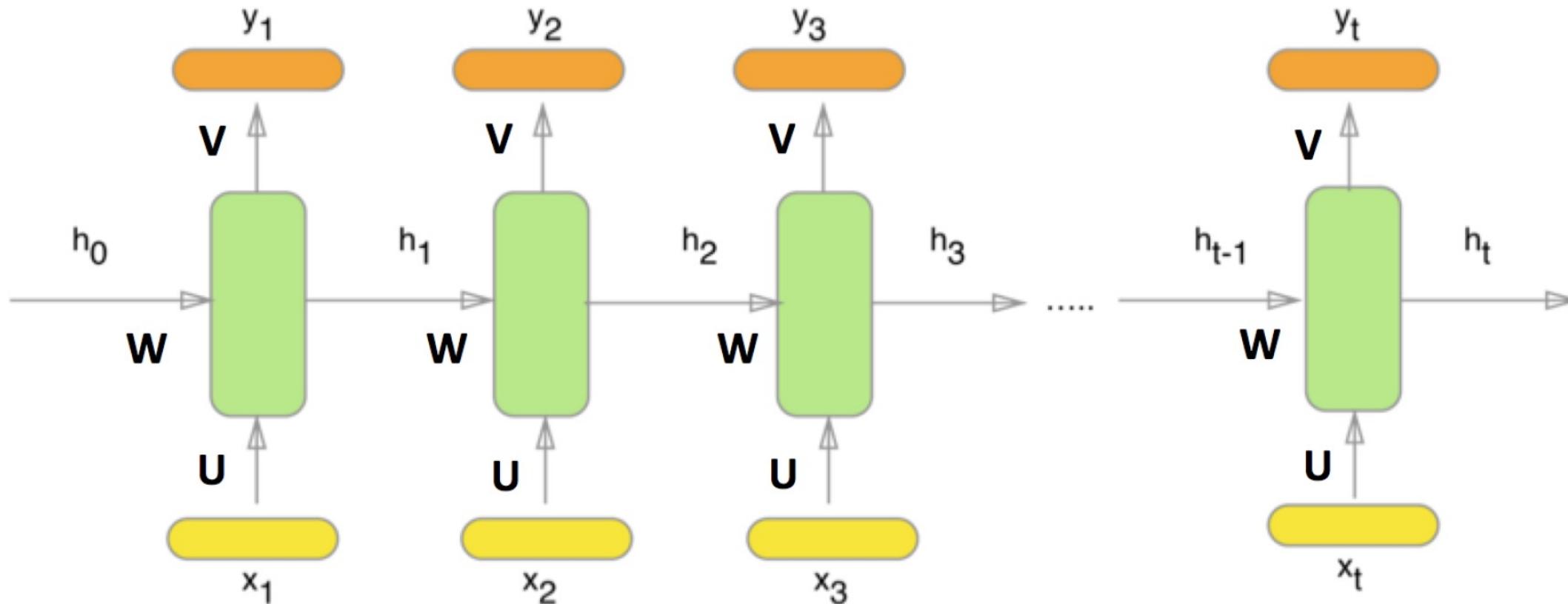
- **RNN Cell:**  $h_t = \phi(x_t, h_{t-1})$ 
  - $\phi$ : linear projection of  $x_t$  and  $h_{t-1}$ , i.e. fully connected layers
  - $h_t = f(Ux_t + Wh_{t-1} + b_h)$ 
    - $U$  matrix size  $I \times d$ ,  $W$  matrix size  $I \times I$  ( $b$  vector size  $I$ )
    - $f \leftarrow \tanh$  non-linearity



# Recurrent Neural Networks (RNNs)

- Depending on the task: prediction at each time step

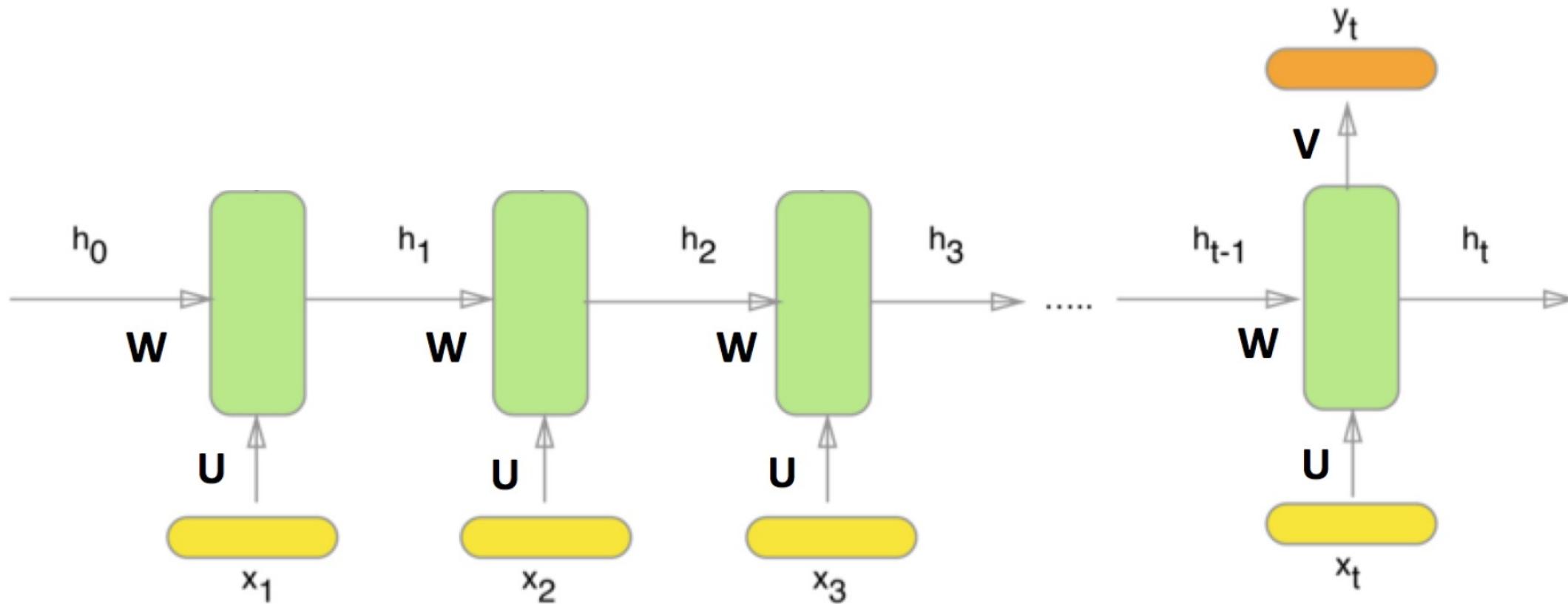
$$y_t = f'(\mathbf{V}h_t + \mathbf{b}_y) \quad \text{e.g., } f' \leftarrow \text{soft-max if } y_t \leftrightarrow \text{class probabilities}$$



# Recurrent Neural Networks (RNNs)

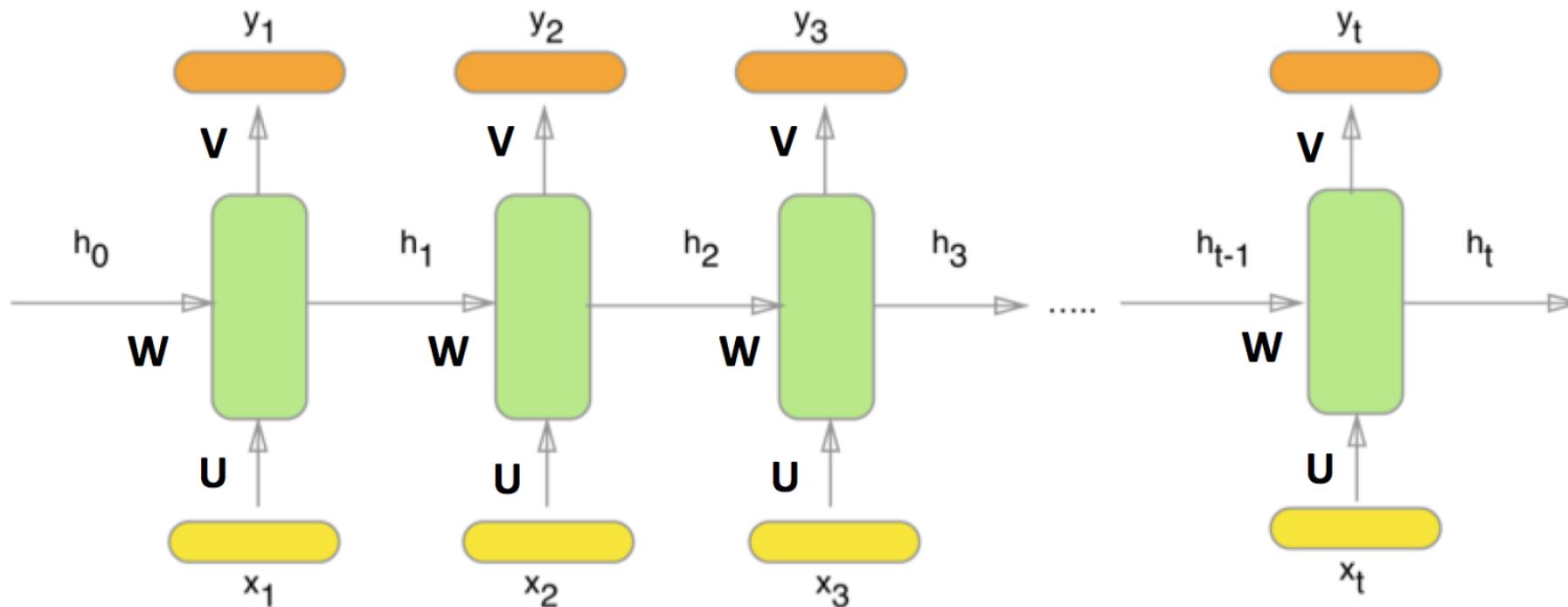
- Depending on the task: prediction at the last time step

$$y_t = f'(\mathbf{V}h_t + \mathbf{b}_y) \quad \text{e.g., } f' \leftarrow \text{soft-max if } y_t \leftrightarrow \text{class probabilities}$$



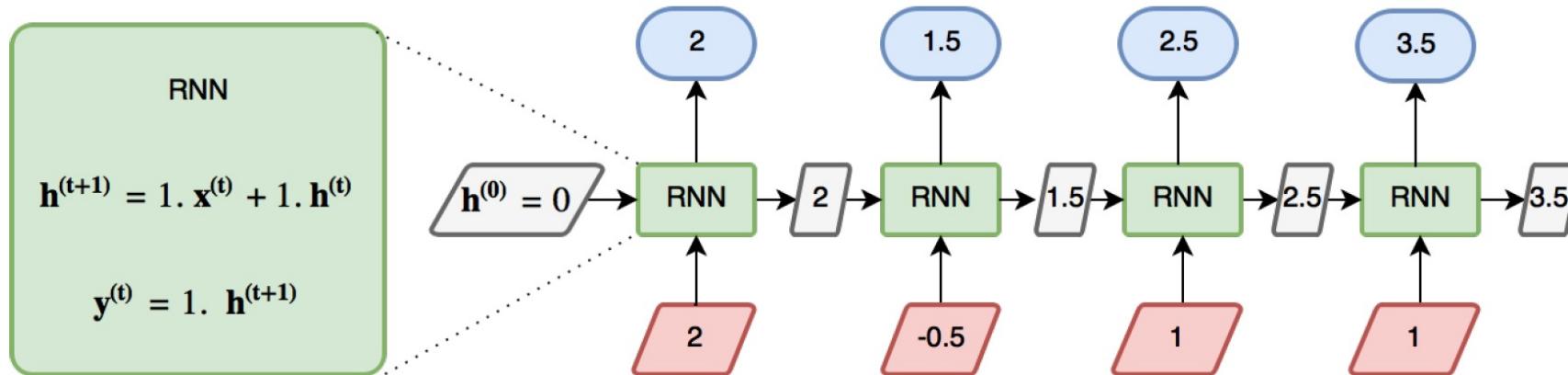
# RNNs expressiveness

- Recap: Feed-forward neural networks are universal function approximators [Cybenko, 1989]
- Expressibility of the mapping between  $\{x_t\}_{t \in \{1; T\}}$  and  $\{y_t\}_{t \in \{1; T\}}$ ?
  - RNNs are universal program approximators [Siegelmann and Sontag, 1995]
    - Can approximate any any computable function, *i.e.* Turing machine
  - RNNs can approximate any measurable sequence to sequence mapping [Hammer, 2000]

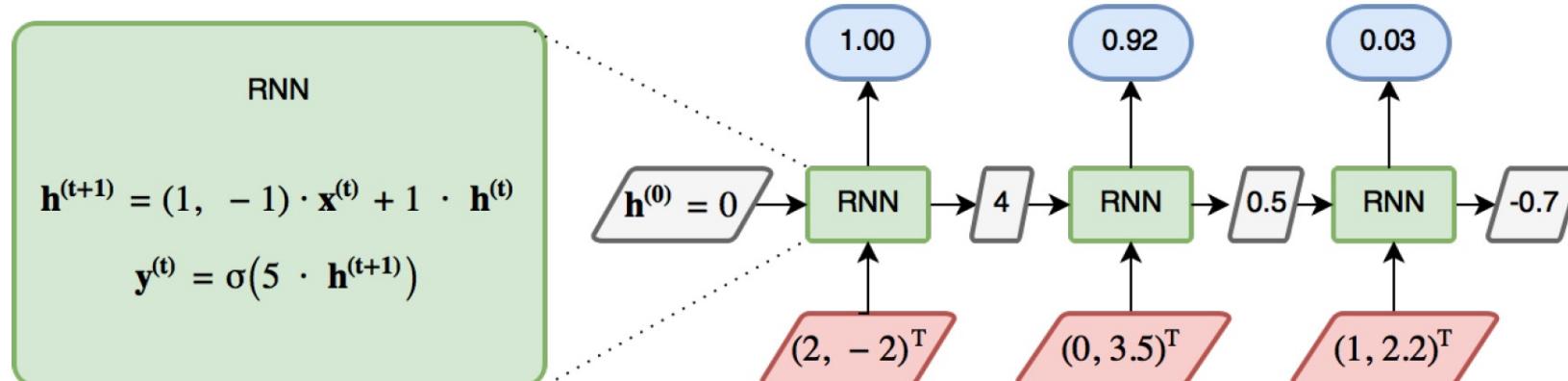


# RNNs expressiveness: examples

- Computing sum



- Determining if sum of 1<sup>st</sup> dimension values > than sum of 2<sup>nd</sup> dimension



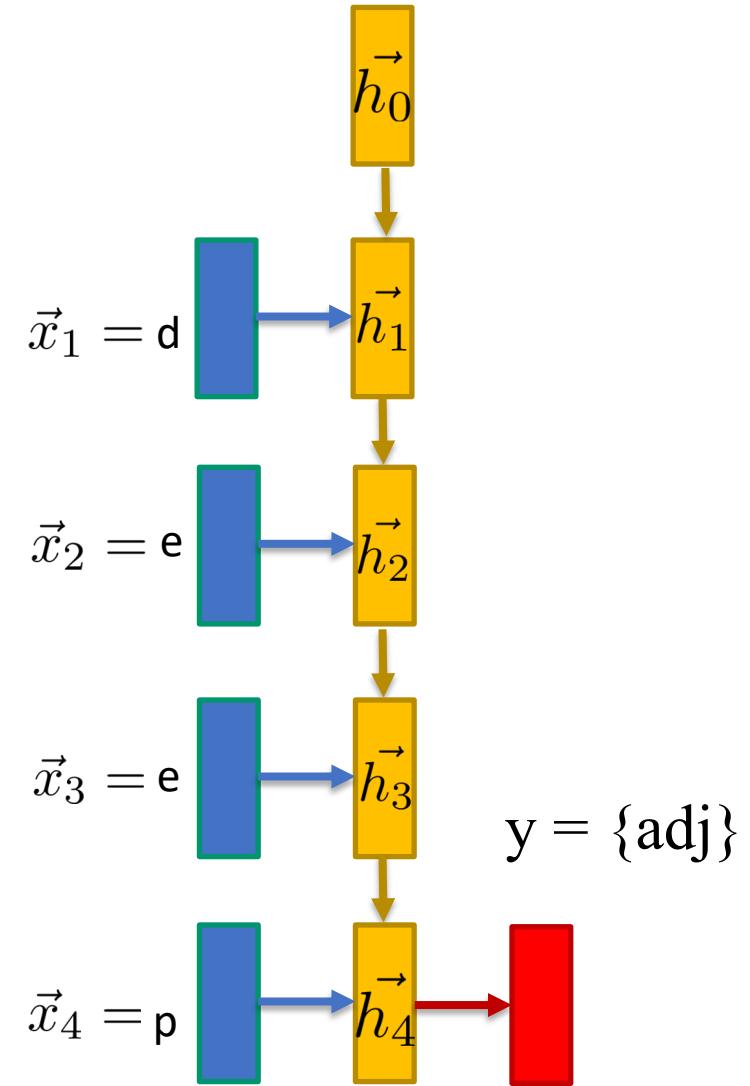
# Toy ex: grammatical analysis, 'deep' => adj

- Input  $x_t$ : character, one-hot encoding

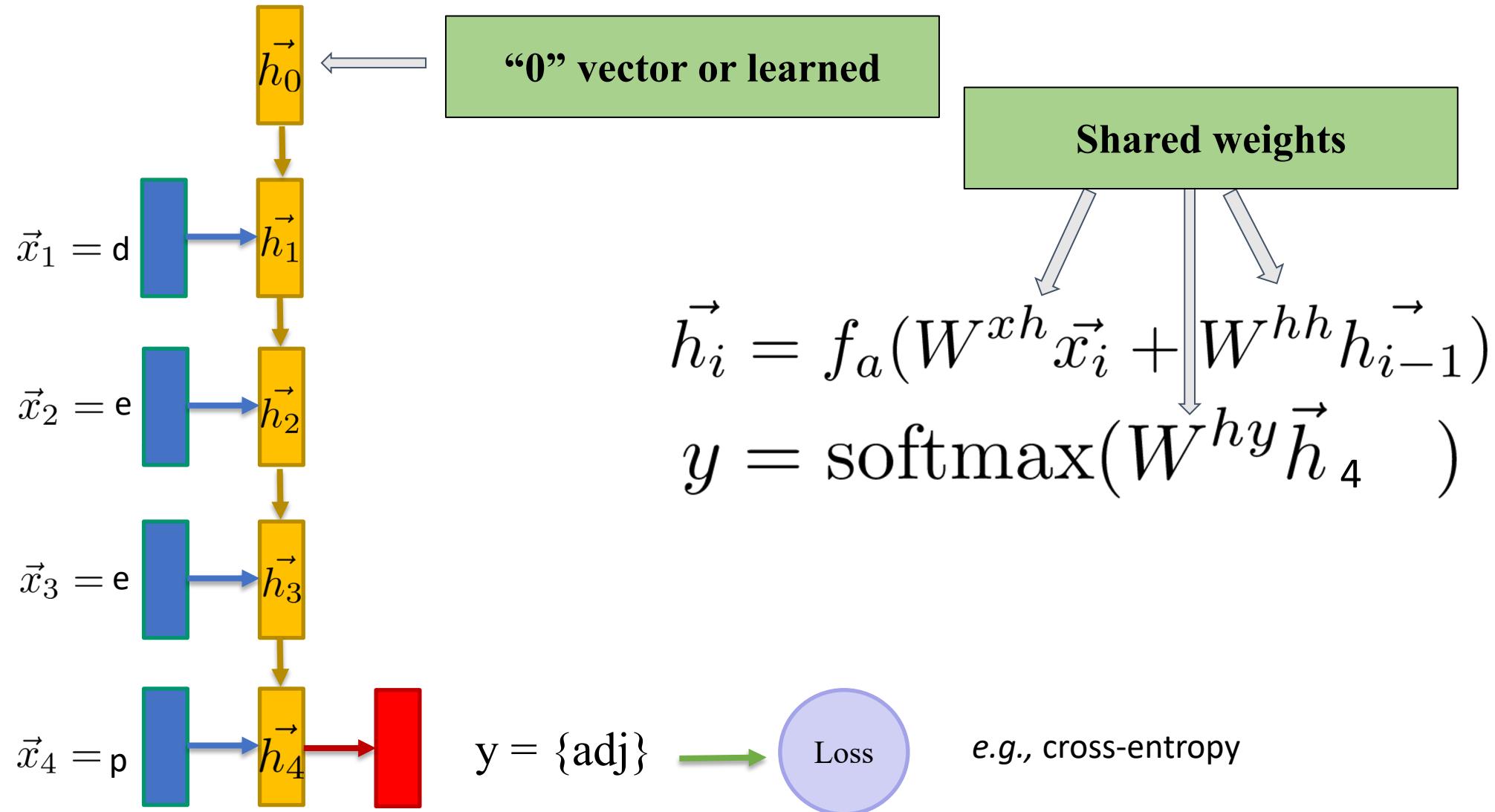
$$\begin{aligned} 'a' &= [1,0,0,\dots,0] \\ 'b' &= [0,1,0,\dots,0] \\ 'c' &= [0,0,1,\dots,0] \end{aligned} \quad \left. \right\} \in R^V$$

- Output  $y_t$ : grammatical class

$$\begin{aligned} \text{'verb'} &= [1,0,0,\dots,0] \\ \text{'noun'} &= [0,1,0,\dots,0] \\ \text{'adj'} &= [0,0,1,\dots,0] \end{aligned} \quad \left. \right\} \in R^M$$



# Ex: grammatical analysis, 'deep' => adj



# Outline

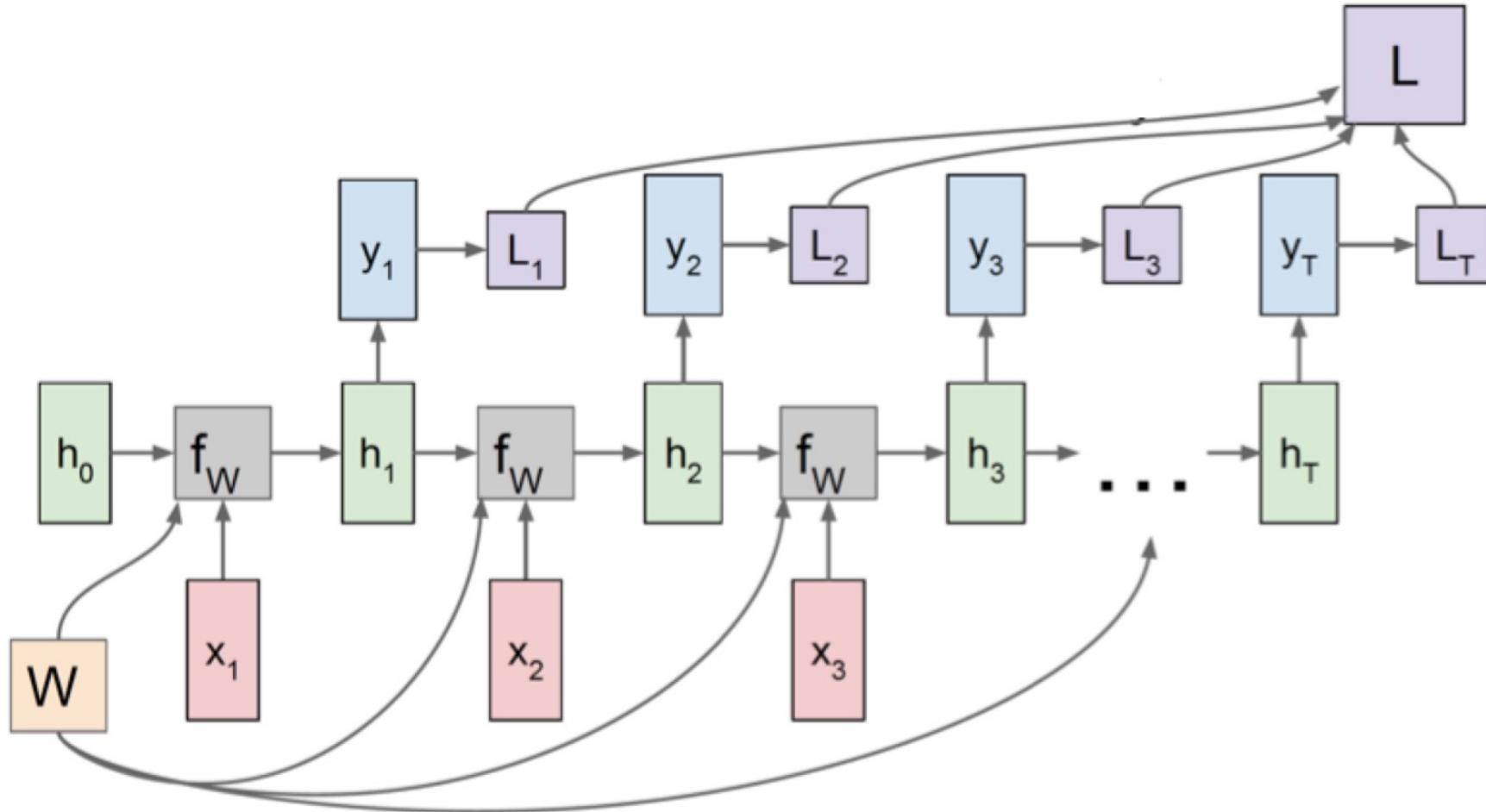
## I. Recurrent Neural Networks (RNNs)

- a) Vanilla RNNs
- b) RNN training
- c) RNN architectures
- d) Applications

## II. Attention models & transformers

# RNN training: formulation

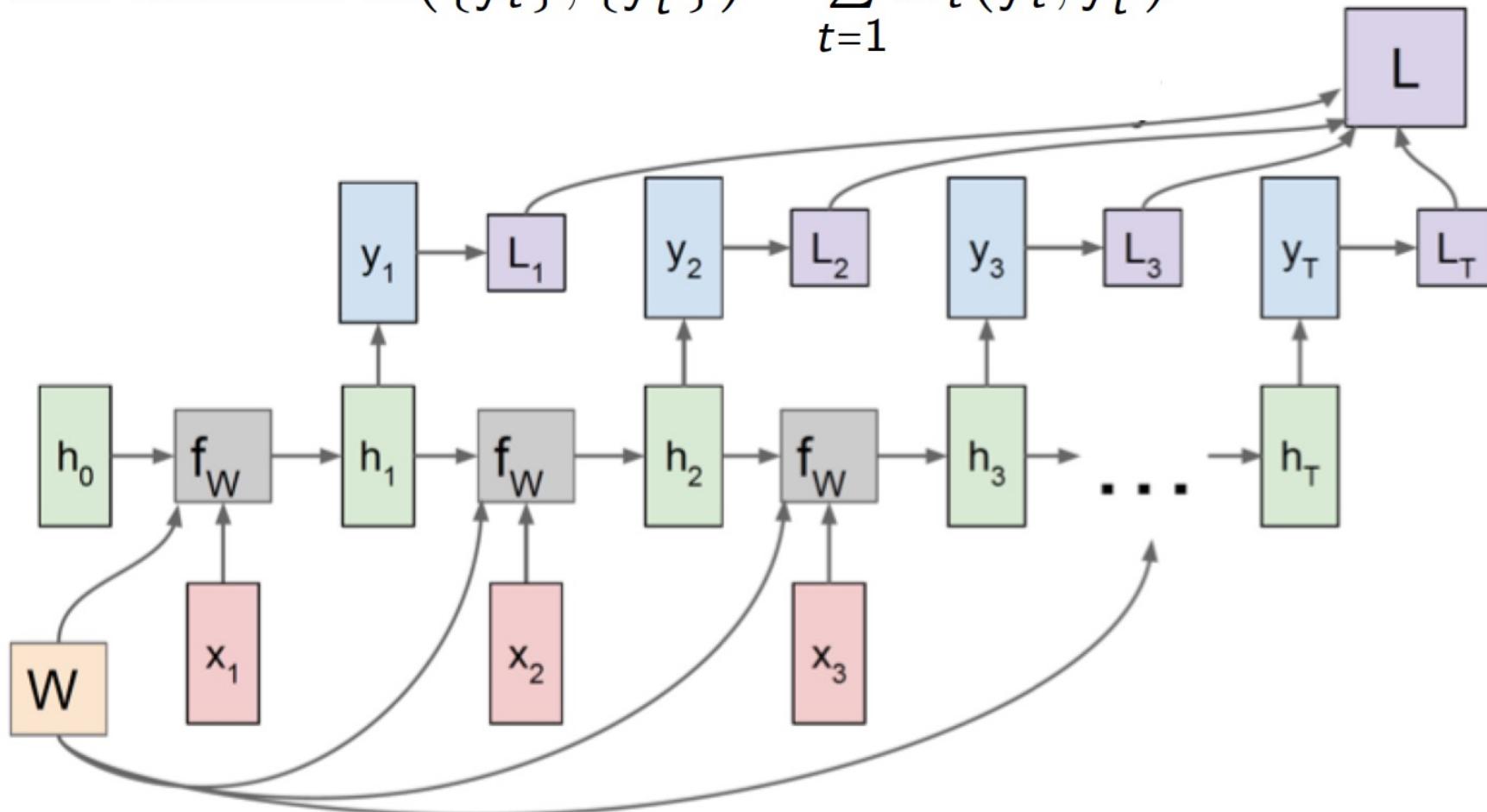
- Comparing output prediction  $\{y_t\}_{t \in \{1; T\}}$  with supervision  $\{y_t^*\}$ 
  - Task-dependent, e.g. only  $\{y_T^*\}$  in many-to-one



Credit: Fei-Fei

# RNN training: formulation

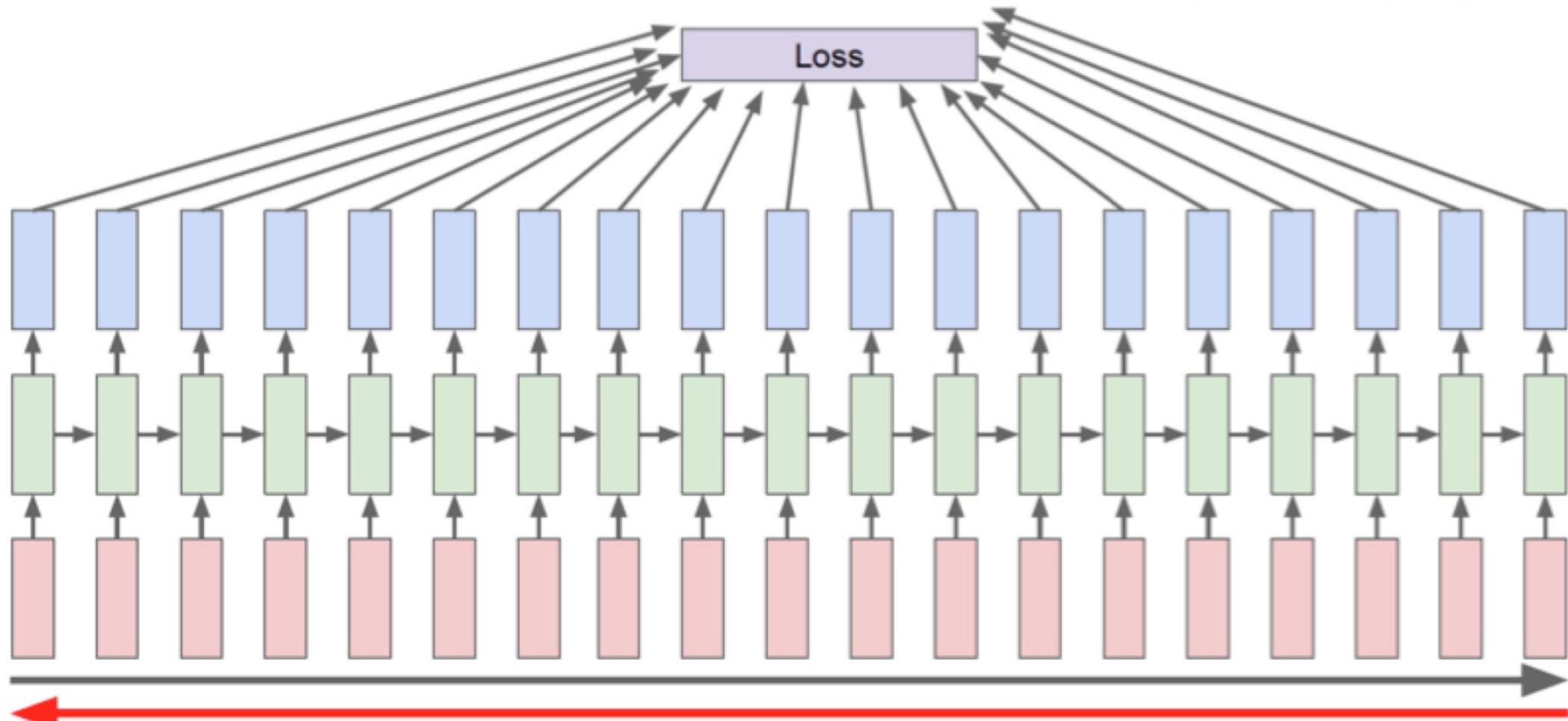
- Loss function at time  $t$ :  $\mathcal{L}_t(y_t, y_t^*)$ , e.g. cross-entropy (classification)
- Total loss function  $\mathcal{L}(\{y_t\}, \{y_t^*\}) = \sum_{t=1}^T \mathcal{L}_t(y_t, y_t^*)$



Credit: Fei-Fei

# Back-Propagation Through Time (BPTT)

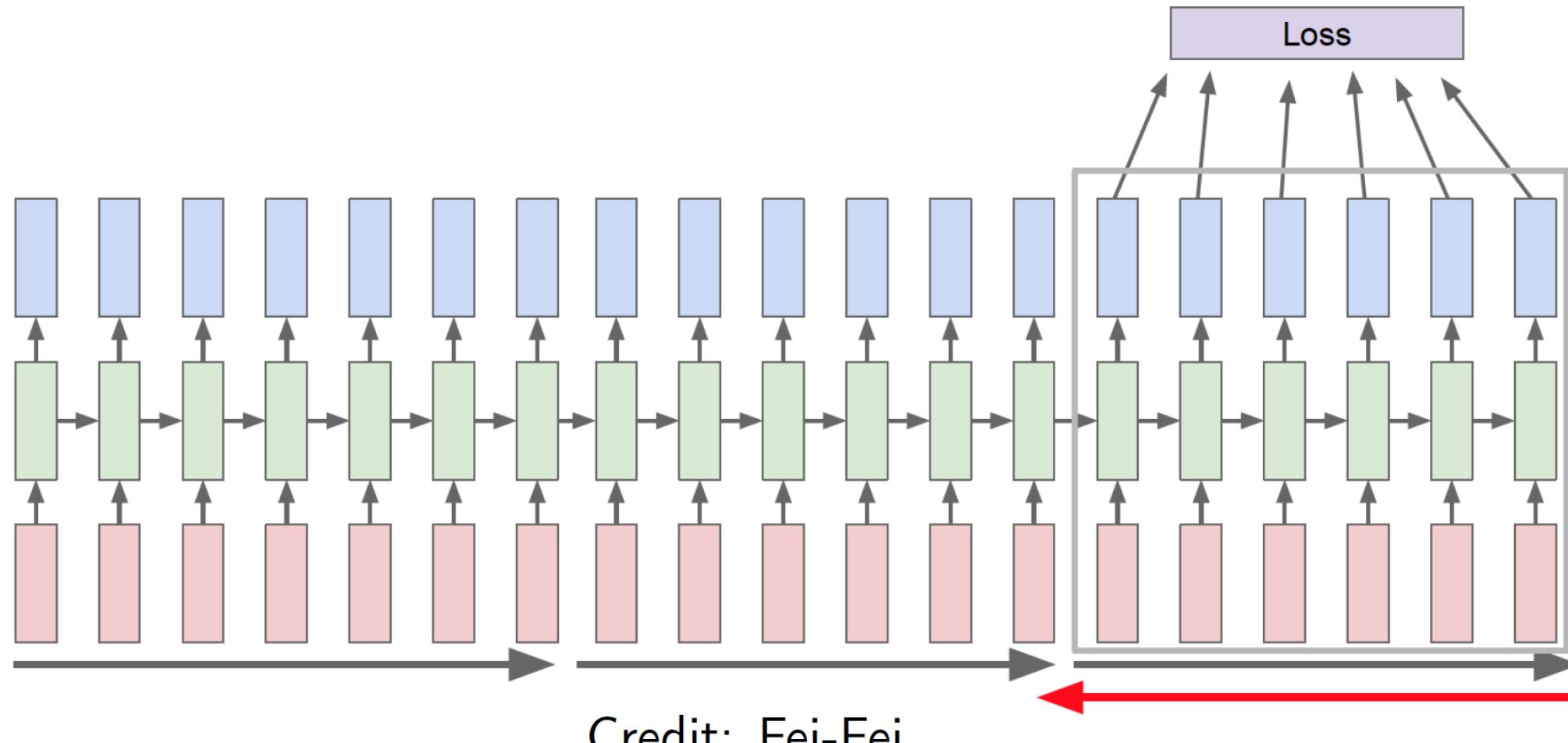
- Similar to standard back-prop, with time (sequence)  $\sim$  depth



Credit: Fei-Fei

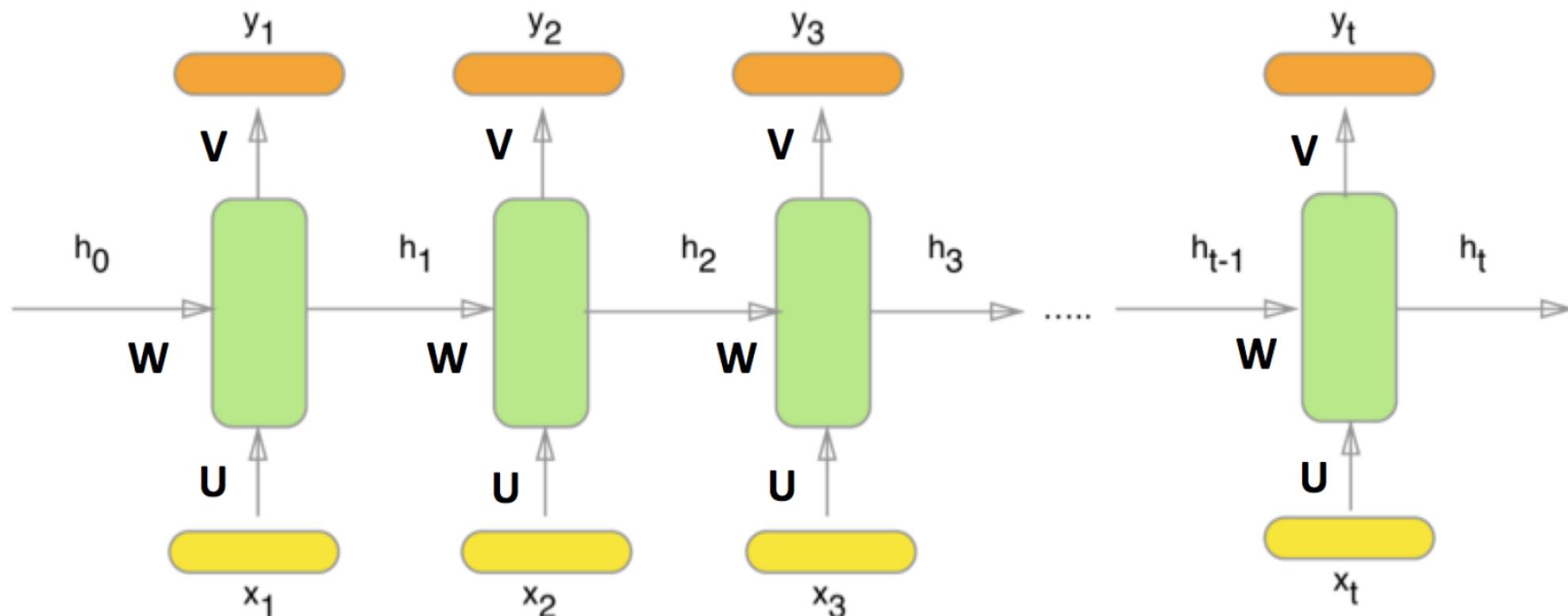
# Truncated BPTT

- Issue for large T => storage, + vanishing gradients issues (next)
- Truncated BPTT => BPTT on local temporal windows



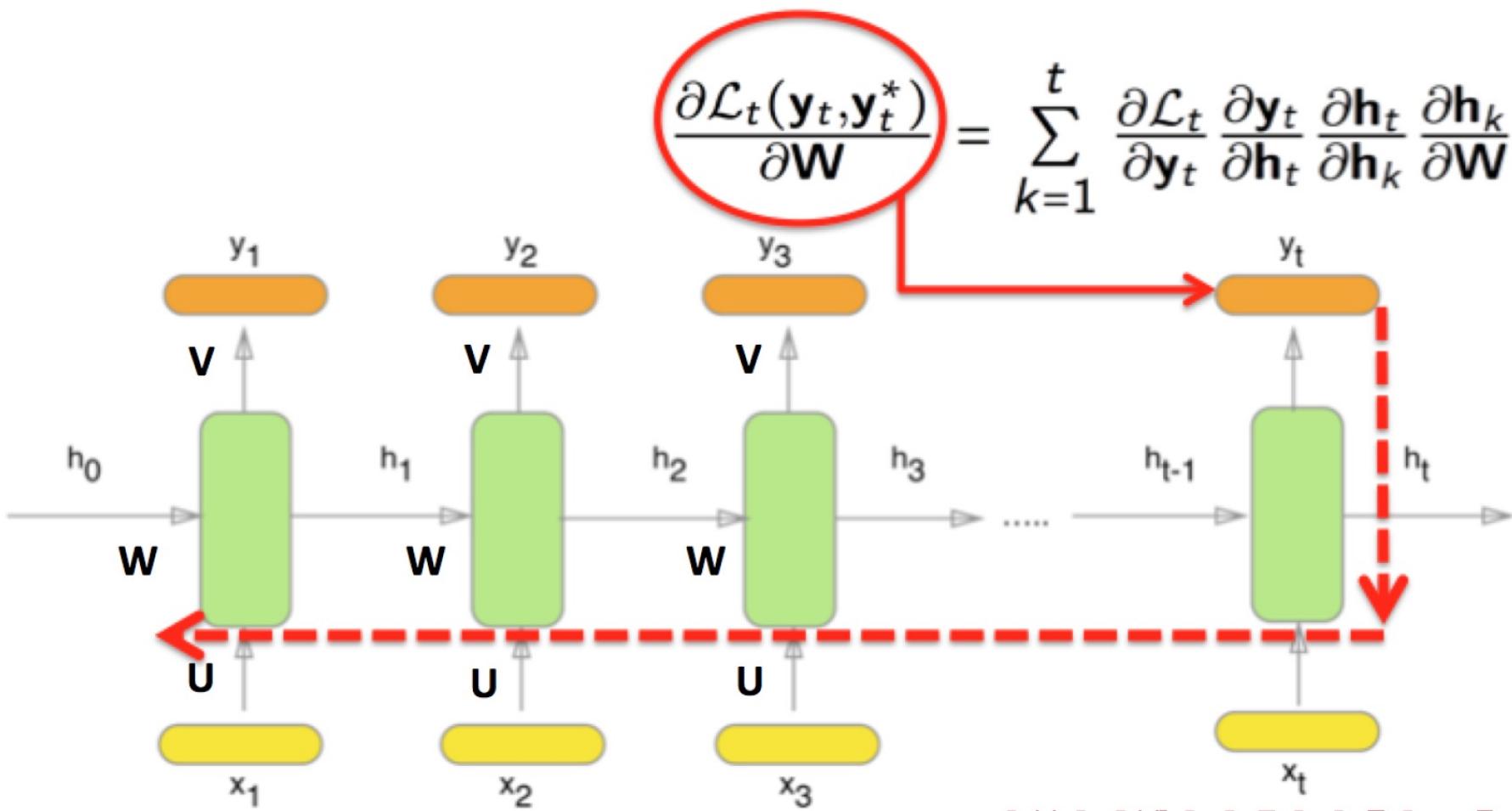
# BPTT: Gradient Computation

- **BPTT**: computing gradient  $\frac{\partial \mathcal{L}_t}{\partial W}$ ,  $\frac{\partial \mathcal{L}_t}{\partial U}$ ,  $\frac{\partial \mathcal{L}_t}{\partial V}$  (+biases)
- **Unfolded RNN**: same spirit as back-prop with fully connected networks (chain rule)
  - **BUT**: shared parameters  $W$ ,  $U$ ,  $V$  across time



# BPTT: Gradient Computation

- Shared parameters  $W, U, V$  across time  
⇒ gradients depend on the whole past history
- Ex: for  $W$ :  $\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{k=1}^t \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$



# BPTT: Gradient Computation

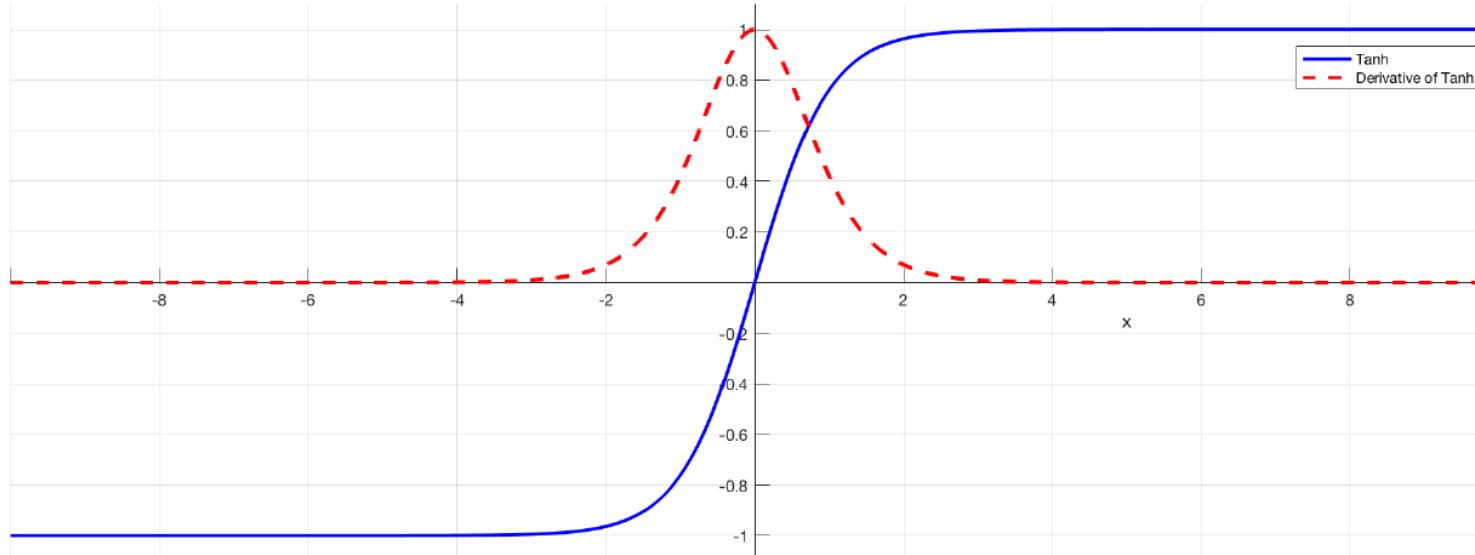
- $\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{k=1}^t \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}}$
- Chain rule (again):  $\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}}$
- $h_t = f(Ux_t + Wh_{t-1} + b_h)$ , e.g.  $f$  tanh
- Jacobian matrix  $\frac{\partial h_j}{\partial h_{j-1}} = W^T diag[f'(h_{j-1})]$

$$\Rightarrow \text{Analyzing } \boxed{\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^t W^T diag[f'(h_{j-1})] \right\|}$$

# BPTT: Exploding and Vanishing Gradients

$$\left\| \frac{\partial h_t}{\partial h_k} \right\| = \left\| \prod_{j=k+1}^t W^T \text{diag}[f'(h_{j-1})] \right\| \leq (\beta_w \beta_h)^{t-k}$$

- $\beta_h$  for activation ( $\tanh=1$ ,  $\text{sigmoid}=0.25$ ),  $\beta_w$  for  $W$  (largest eigenvalue)
  - $\beta_h \cdot \beta_w > 1 \Rightarrow$  exploding gradients
  - $\beta_h \cdot \beta_w < 1 \Rightarrow$  vanishing gradients
- True for any deep networks, exacerbated for RNNs



# Exploding and Vanishing Gradients: challenges & solutions

## Main challenge: modeling long-range dependencies

### Solution for exploding

- Regularization, e.g.,  $\|W\|_2$
- Simple common strategy: gradient clipping  
⇒ Exploding gradients relatively easy to detect and fix

### Solution for vanishing

- Use Truncated BPTT, but smaller range dependencies
- Using ReLU activation instead of tanh/sigmoid
- Specific architectures/models, e.g. GRU/LSTM

# Outline

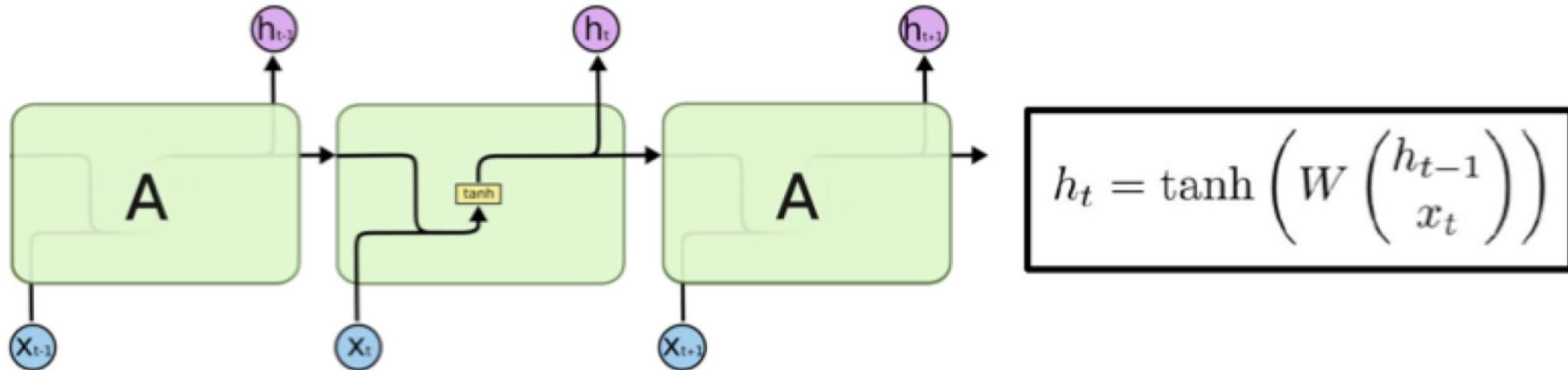
## I. Recurrent Neural Networks (RNNs)

- a) Vanilla RNNs
- b) RNN training
- c) Specific RNN architectures
- d) Applications

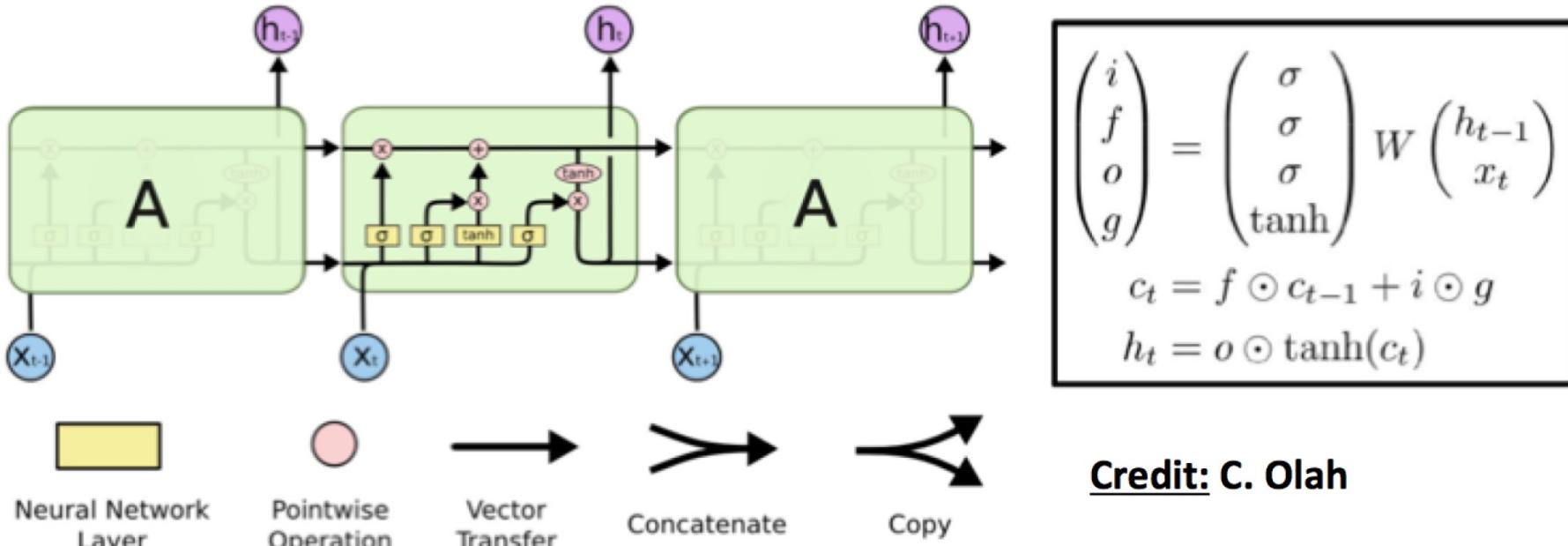
## II. Attention models & transformers

# Long Short-Term Memory (LSTM)

- Recap: Vanilla RNN cell

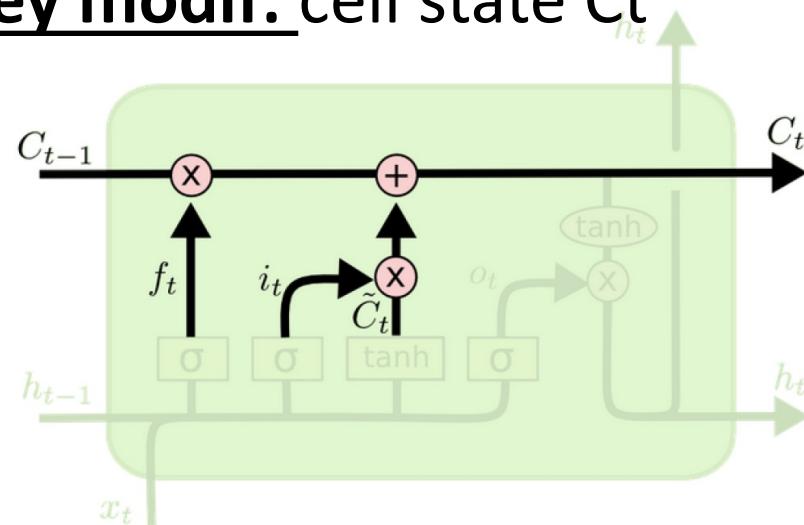


- Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997]



# Long Short-Term Memory (LSTM)

Key modif: cell state  $C_t$



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

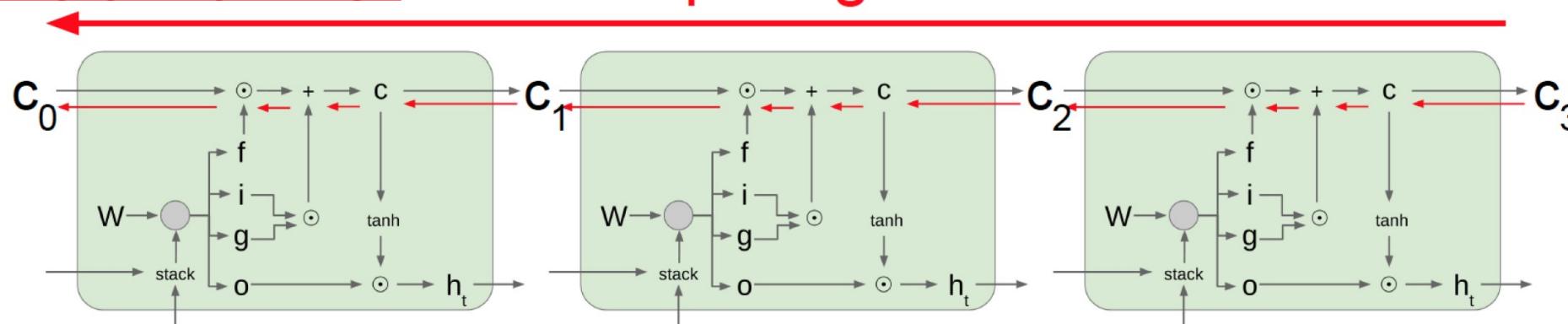
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Forget  
Input

- Only elementwise multiplication and addition, no matrix multiply by  $W$

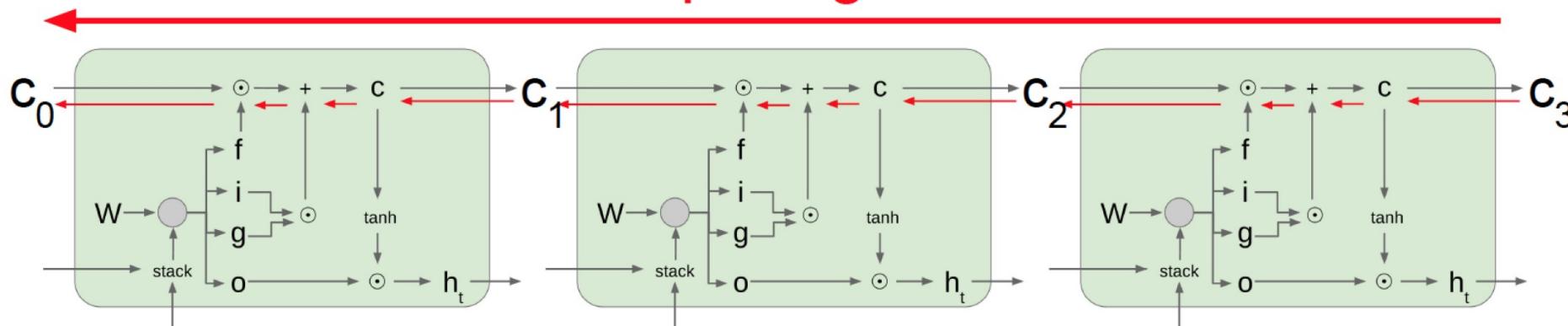
**Key property:** Uninterrupted gradient flow!



# Long Short-Term Memory (LSTM)

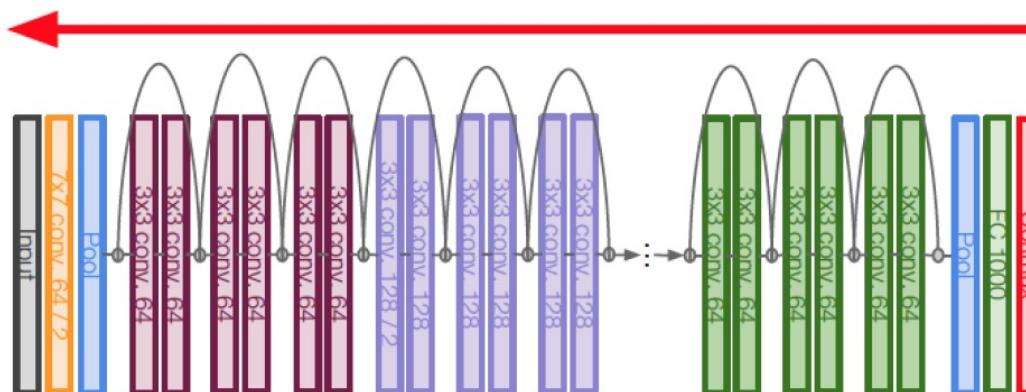
- Only elementwise multiplication and addition, no matrix multiply by W

**Key property: Uninterrupted gradient flow!**



**Sounds familiar?**

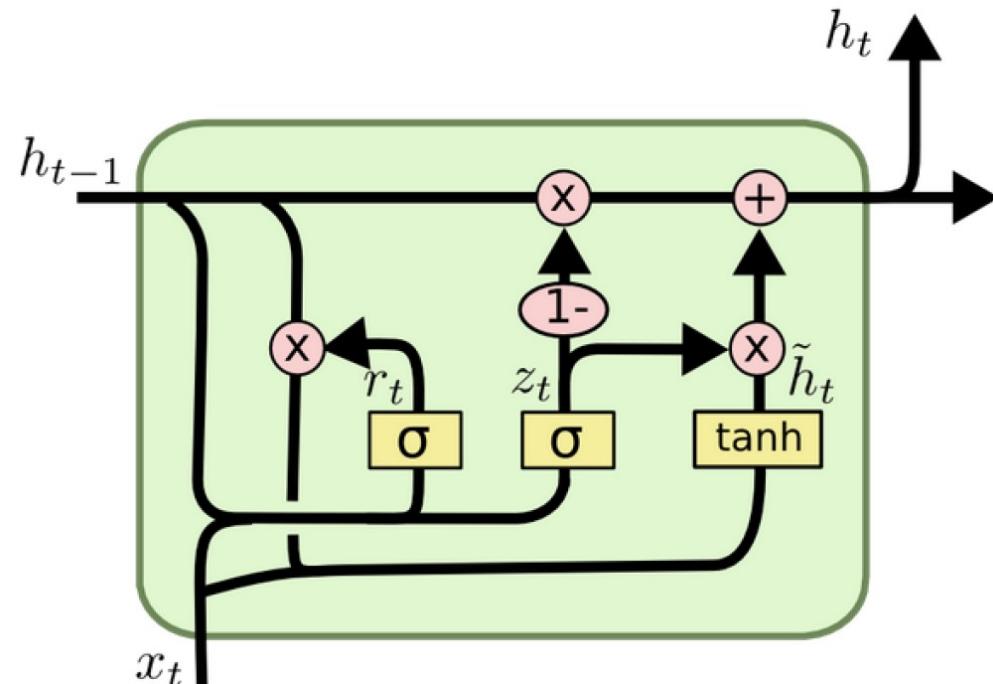
Similar to ResNets!



Credit: Fei-Fei

# Gated Recurrent Unit [Cho et al., 2014]

- LSTM popular variant: Gated Recurrent Unit (GRU) [Cho et al., 2014]
  - Combines forget and input gates into a single "update gate"
  - Merges the cell state and hidden state



$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$

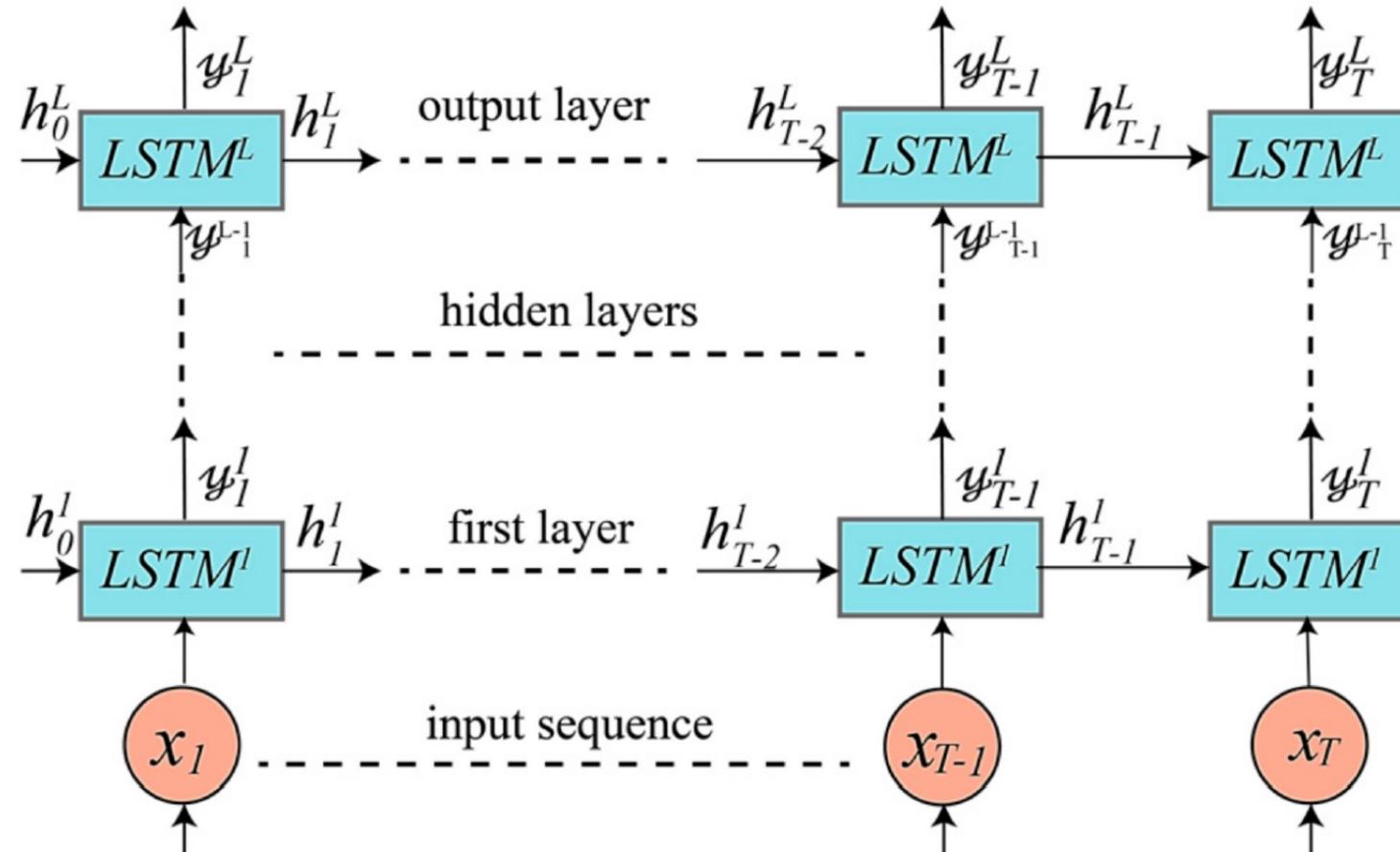
$$\tilde{h}_t = \tanh (W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- Simpler than LSTM, generally slightly inferior performances

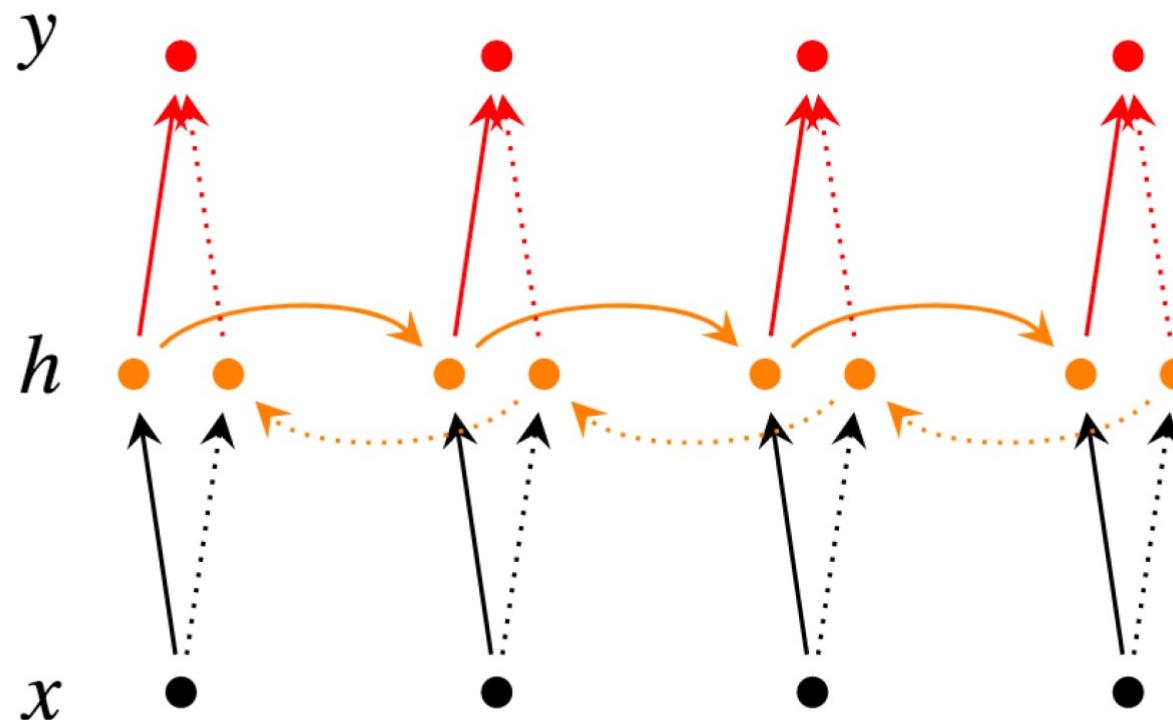
# Deep RNNs

- Stacking RNN/ LSTM layers  $\Rightarrow$  learning more complex features
- Deep LSTM: very powerful, especially when stacked and made even deeper and if you have lots and lots of data



# Bi-directionnal RNNs

- For classification, incorporate information from words both preceding and following



$$\begin{aligned}\vec{h}_t &= f(\vec{W}x_t + \vec{V}\vec{h}_{t-1} + \vec{b}) \\ \overleftarrow{h}_t &= f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b}) \\ y_t &= g(U[\vec{h}_t; \overleftarrow{h}_t] + c)\end{aligned}$$

$h = [\vec{h}; \overleftarrow{h}]$  now represents (summarizes) the past and future around a single token.

# Outline

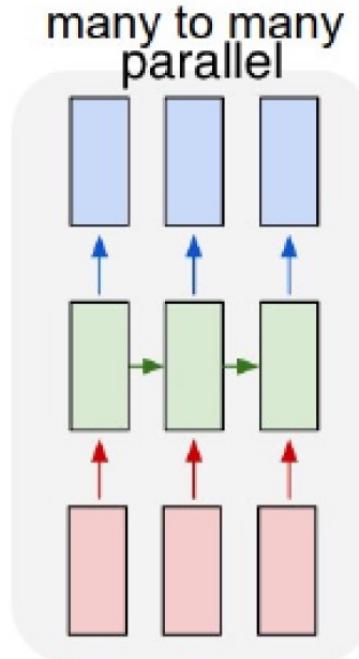
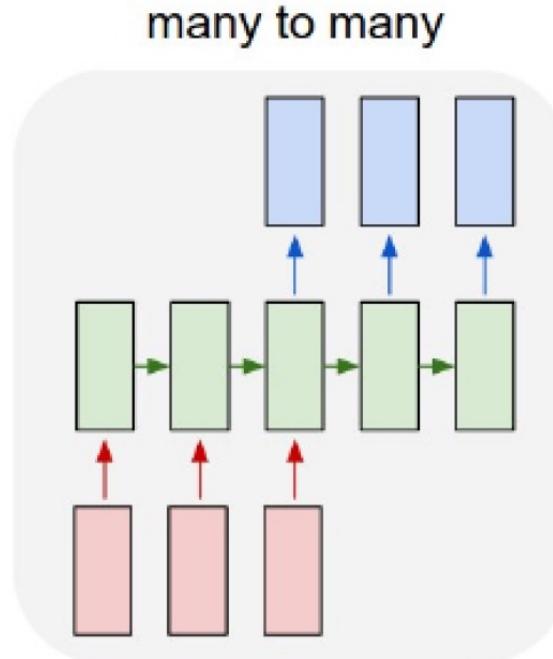
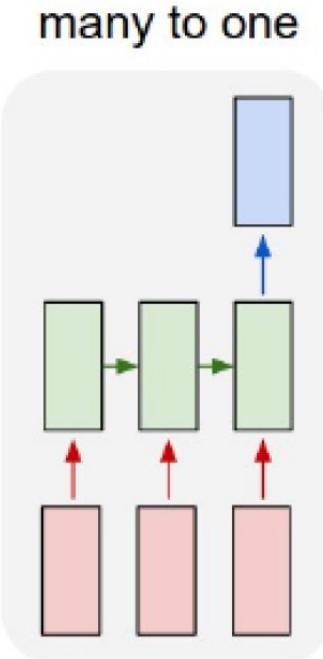
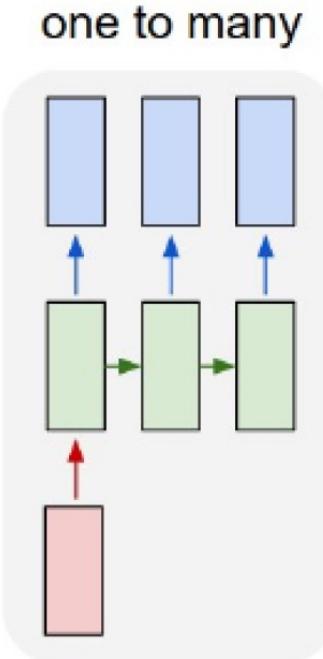
## I. Recurrent Neural Networks (RNNs)

- a) Vanilla RNNs
- b) RNN training
- c) Specific RNN architectures
- d) **RNN Applications**

## II. Attention models & transformers

# RNN applications

- RNN: mapping input sequence  $\{x_t\}_{t \in \{1; T\}}$  into  $\{y_t\}_{t \in \{1; T\}}$
- Different tasks  $\Leftrightarrow$  different mappings



Language models,  
Image captioning

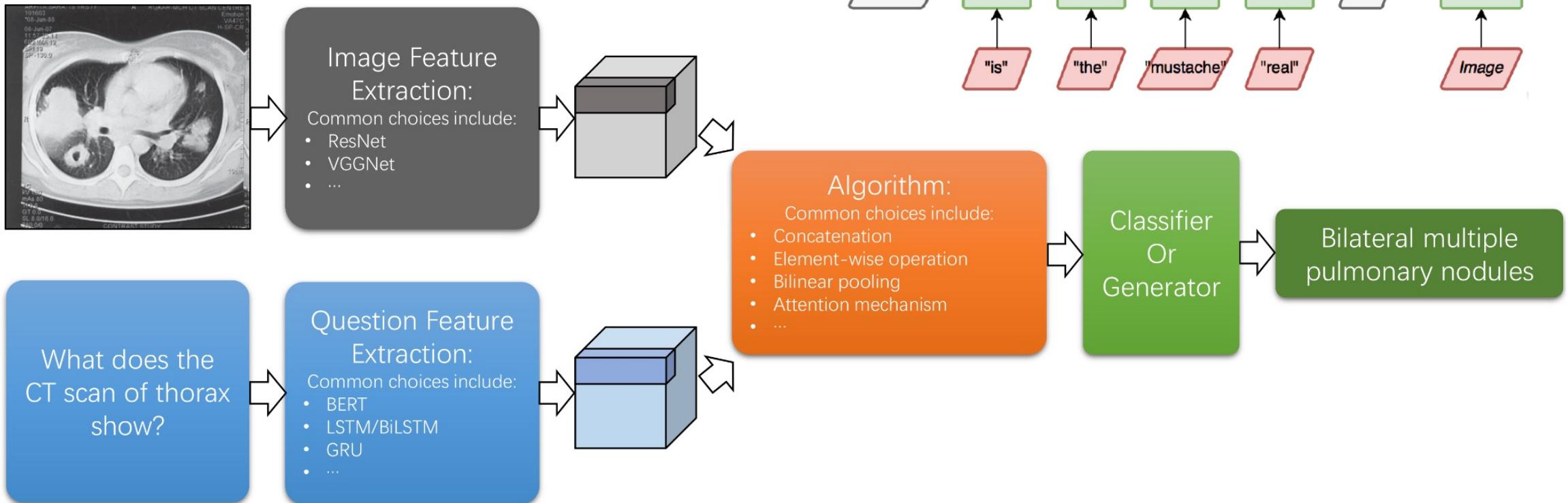
Sentiment classification,  
Visual Question  
Answering (VQA)

Translation, char  
nn

video  
classification

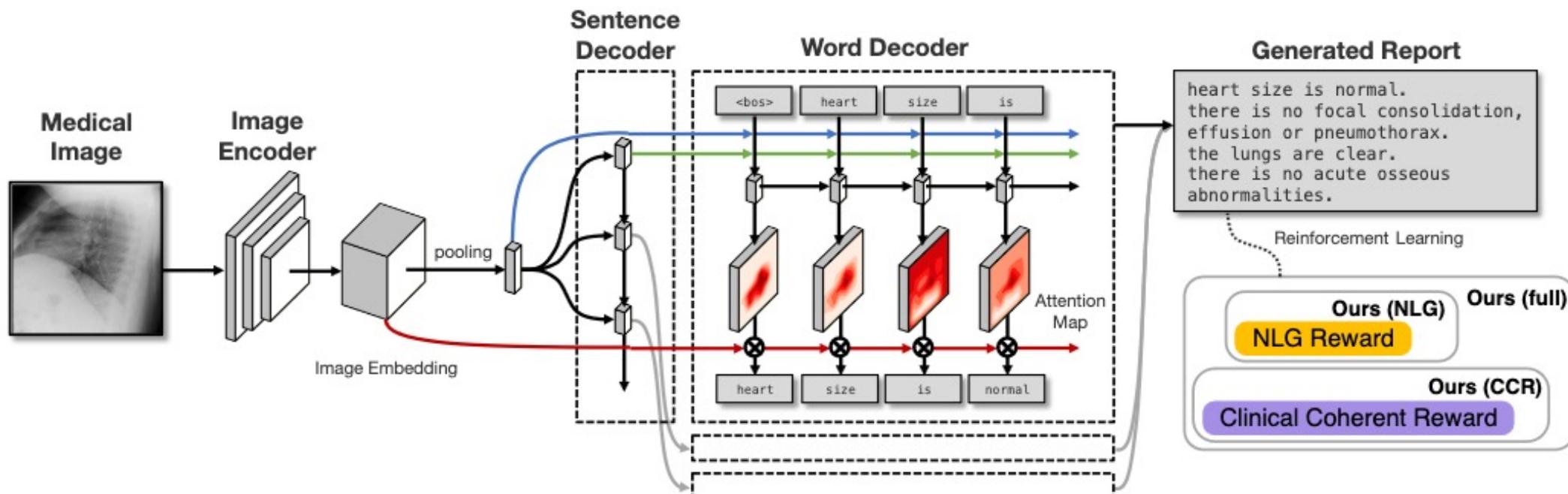
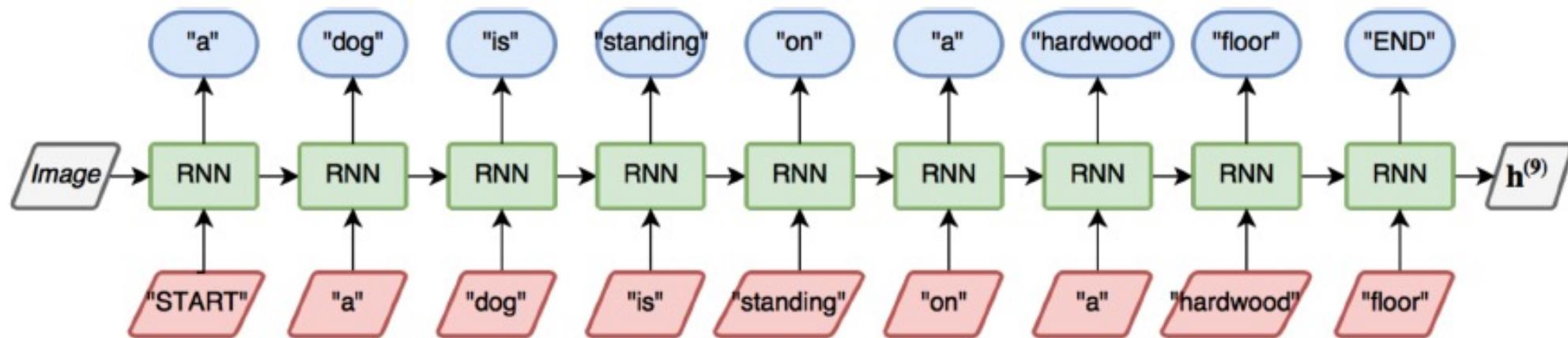
# Medical Visual Question answering (MVQA)

## General architecture



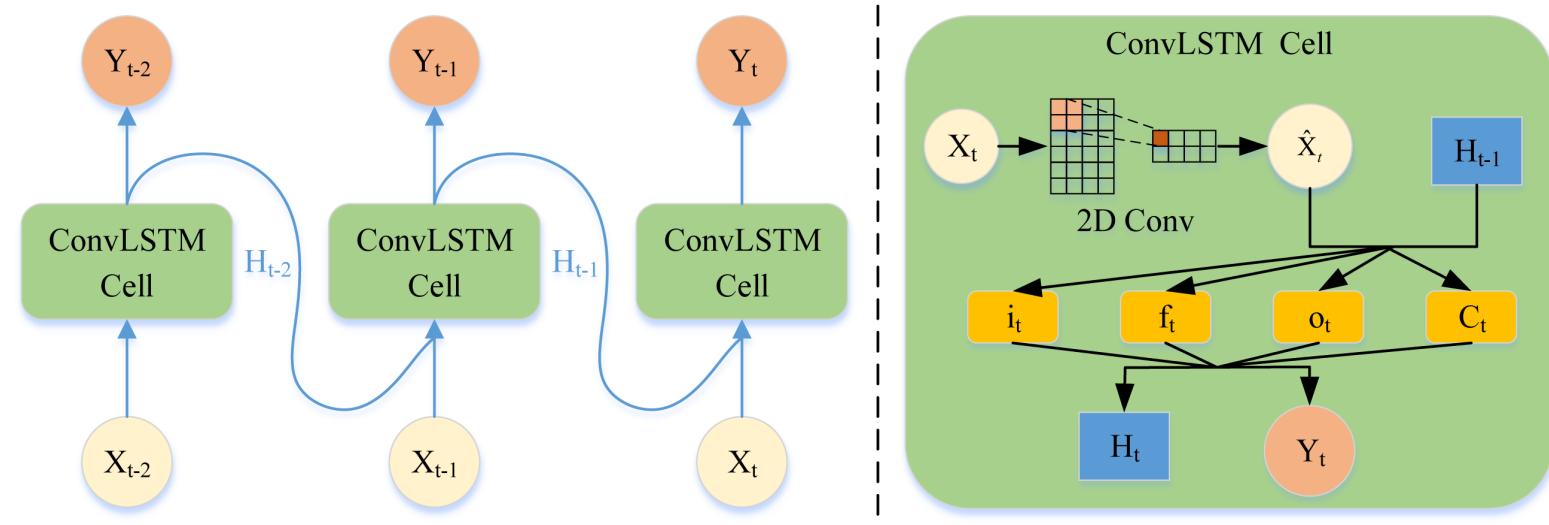
# Image Captioning: medical report generation

One-to-many  
mapping

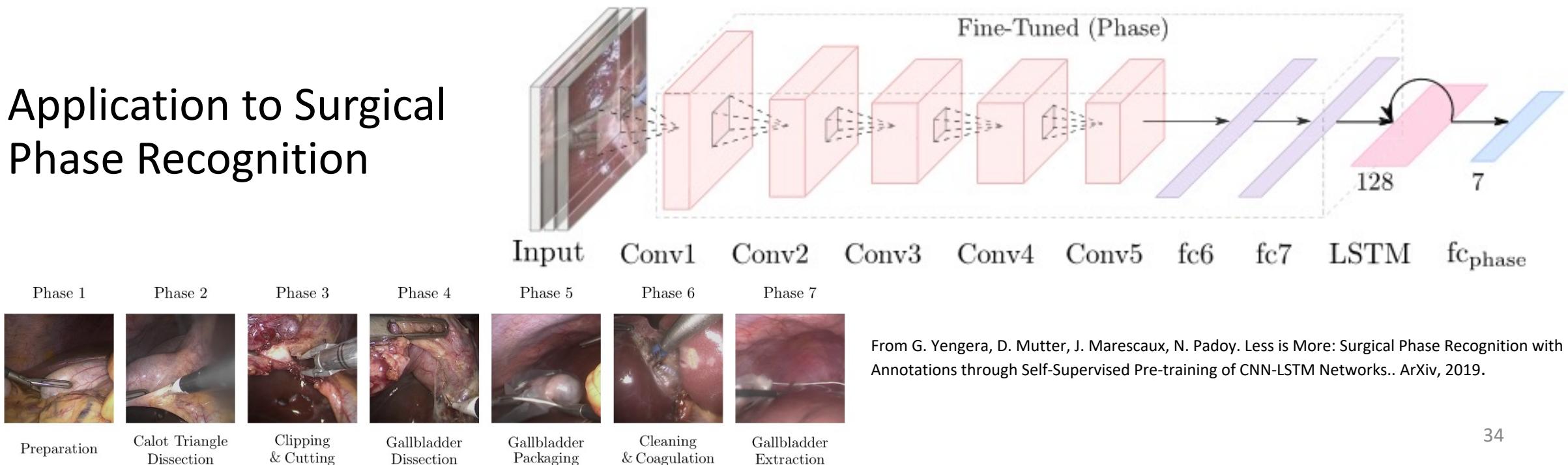


# Video classification

- ConvLSTM [Shi et. al. 2015]
  - generic module for video classification
  - Many-to-many mapping



- Application to Surgical Phase Recognition



# Outline

- I. Recurrent Neural Networks (RNNs)
- II. Attention models & transformers
  - a) Context
  - b) Transformer block
  - c) Medical image segmentation

# Transformers everywhere since 2017

NLP: BERT, GPT-3/4, Chat-GPT, etc

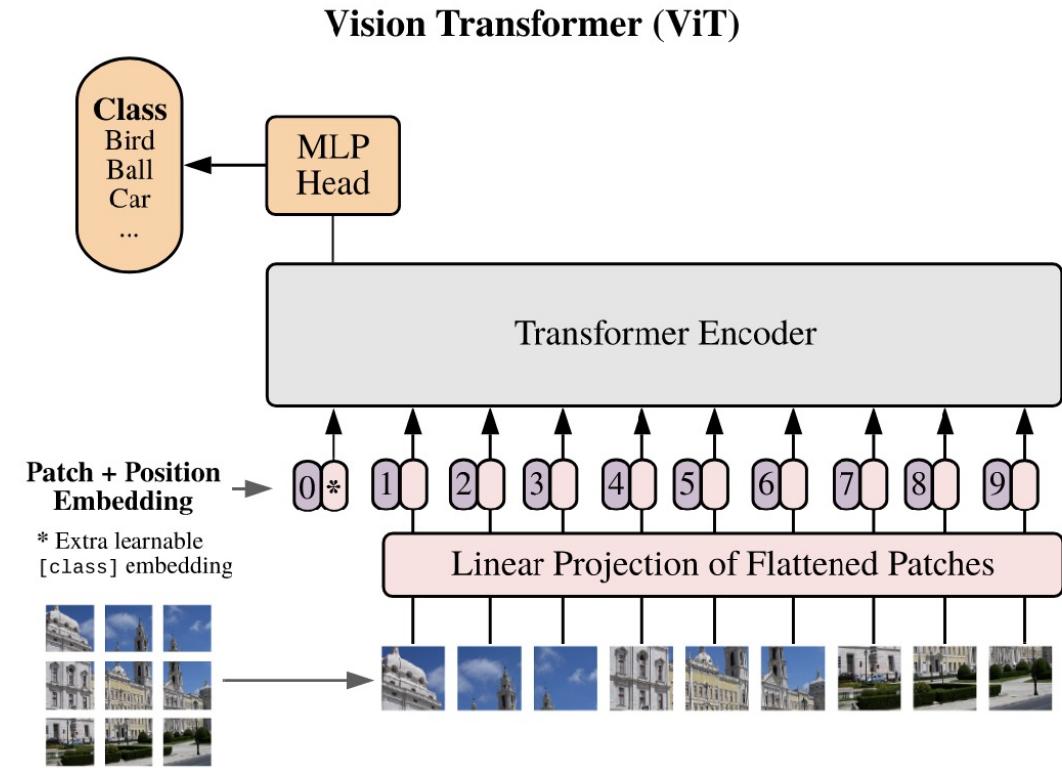
explain quantum theory to a child in the style of snoop dogg

Yo, little homie, let me break it down for you  
Quantum theory, it's a crazy thing, it's true  
See, the world is made up of tiny, tiny particles  
That are always moving, and changing, and actin' wild

These particles, they don't always behave like we think  
Sometimes they're waves, and sometimes they're points, it's a trip  
They can be in two places at once, and they can be in two states  
It's a mind-blowing concept, but that's the quantum fate

So don't be scared, little homie, it's all good  
Quantum theory, it's just the way the world works, it's understood  
It's a wild ride, but it's a cool one too  
Quantum theory, it's the way the world does what it do.

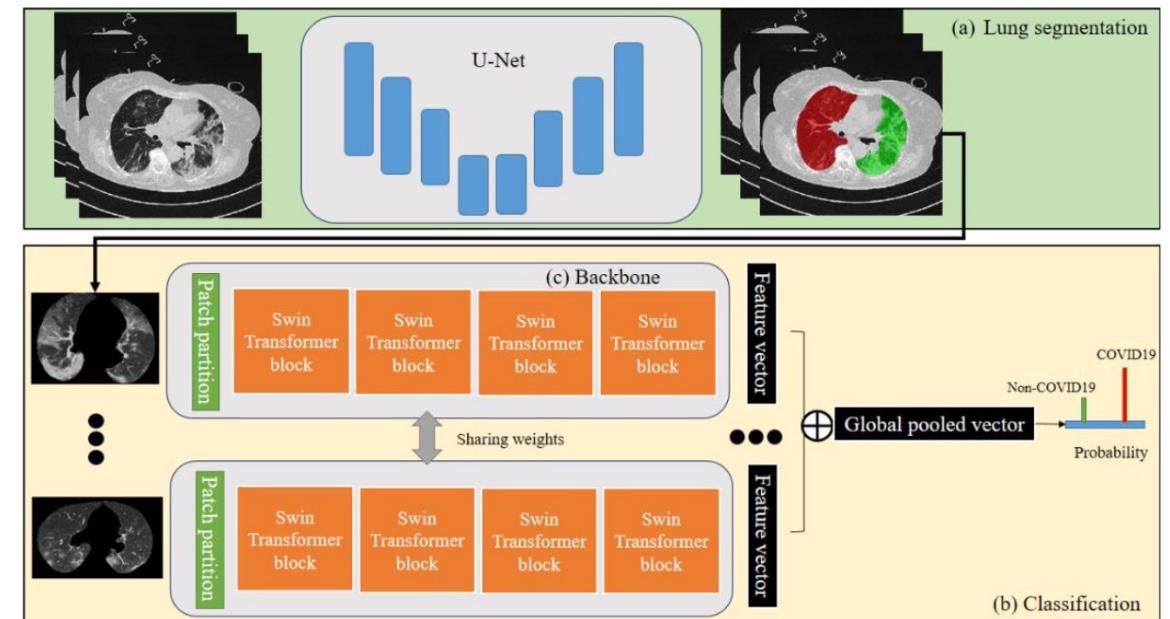
Vision since '21: Vision Image Transformer (ViT)



# Transformer in medical image analysis

## Used in various contexts and tasks

- Image classification, detection, e.g. COVID, Semantic segmentation
- Image Registration
- Image Generation
- Im-2-im translation

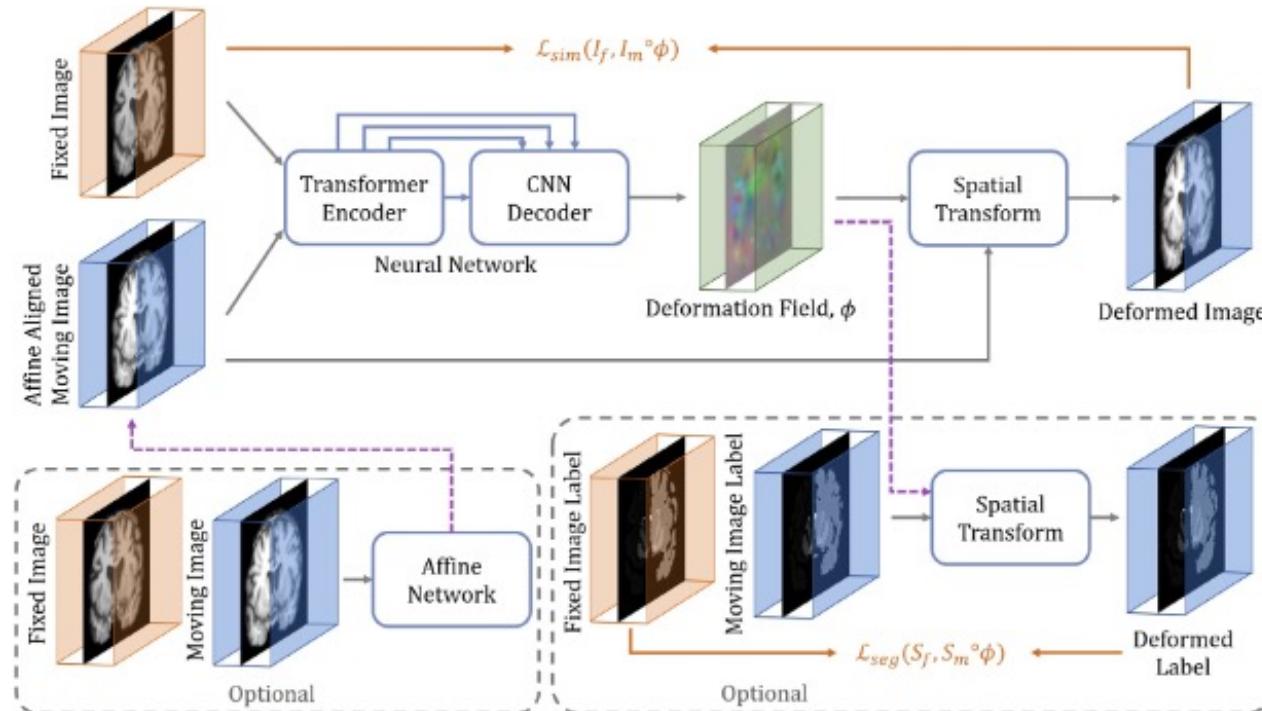


Zhang L, Wen Y. Micov19d: A transformer-based framework for covid19 classification in chest cts. arXiv, 2021.

# Transformer in medical image analysis

## Used in various contexts and tasks

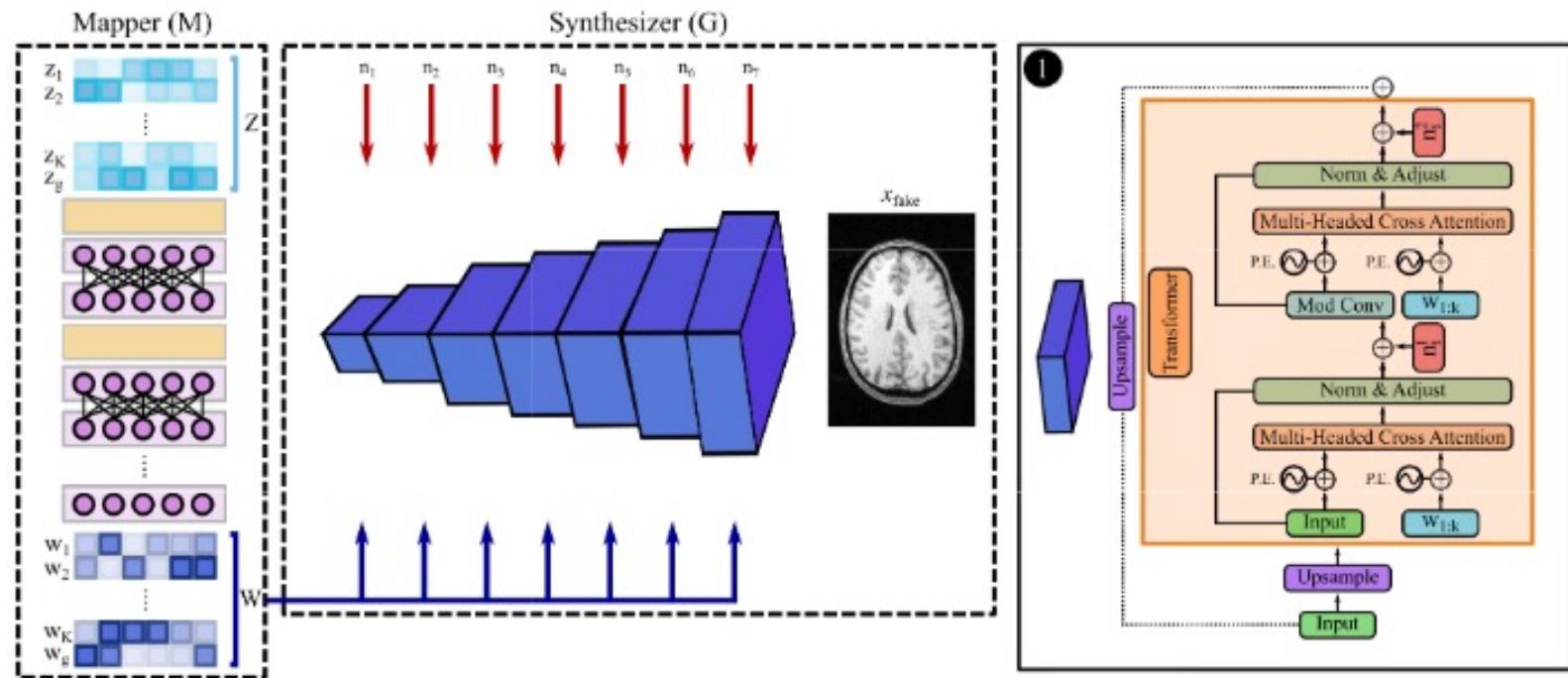
- Image classification, detection, e.g. COVID, Semantic segmentation
- Image Registration
- Image Generation
- Im-2-im translation



# Transformer in medical image analysis

## Used in various contexts and tasks

- Image classification, detection, e.g. COVID, semantic segmentation
- Image Registration
- Image Generation
- Im-2-im translation

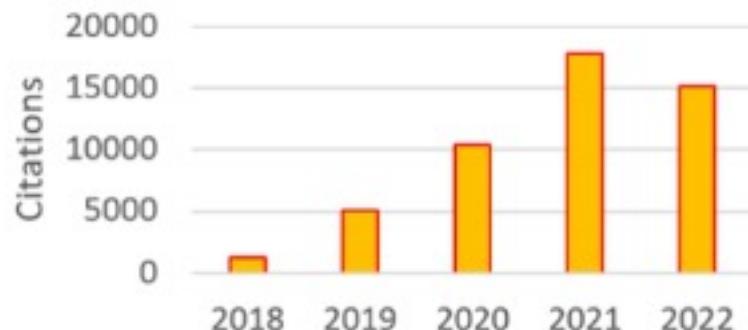


Korkmaz Y, Dar SU, Yurt M, et al. Unsupervised MRI reconstruction via zero-shot learned adversarial transformers. IEEE TMI, VOL. 41, NO. 7, JULY 2022

# Focus on this talk

- Paper on transformer every day...

(a) Citations of Transformer papers in recent years



- By no means exhaustive literature review

Intelligent Medicine 3 (2023) 59–78



Contents lists available at ScienceDirect

Intelligent Medicine

journal homepage: [www.elsevier.com/locate/imed](http://www.elsevier.com/locate/imed)



Review

Transformers in medical image analysis

Kelei He<sup>1,2,#</sup>, Chen Gan<sup>2,#</sup>, Zhuoyuan Li<sup>1,2,#</sup>, Islam Rekik<sup>3,4,#</sup>, Zihao Yin<sup>2</sup>, Wen Ji<sup>2</sup>, Yang Gao<sup>2,5</sup>, Qian Wang<sup>6,\*</sup>, Junfeng Zhang<sup>1,2,\*</sup>, Dinggang Shen<sup>6,7,8,\*</sup>

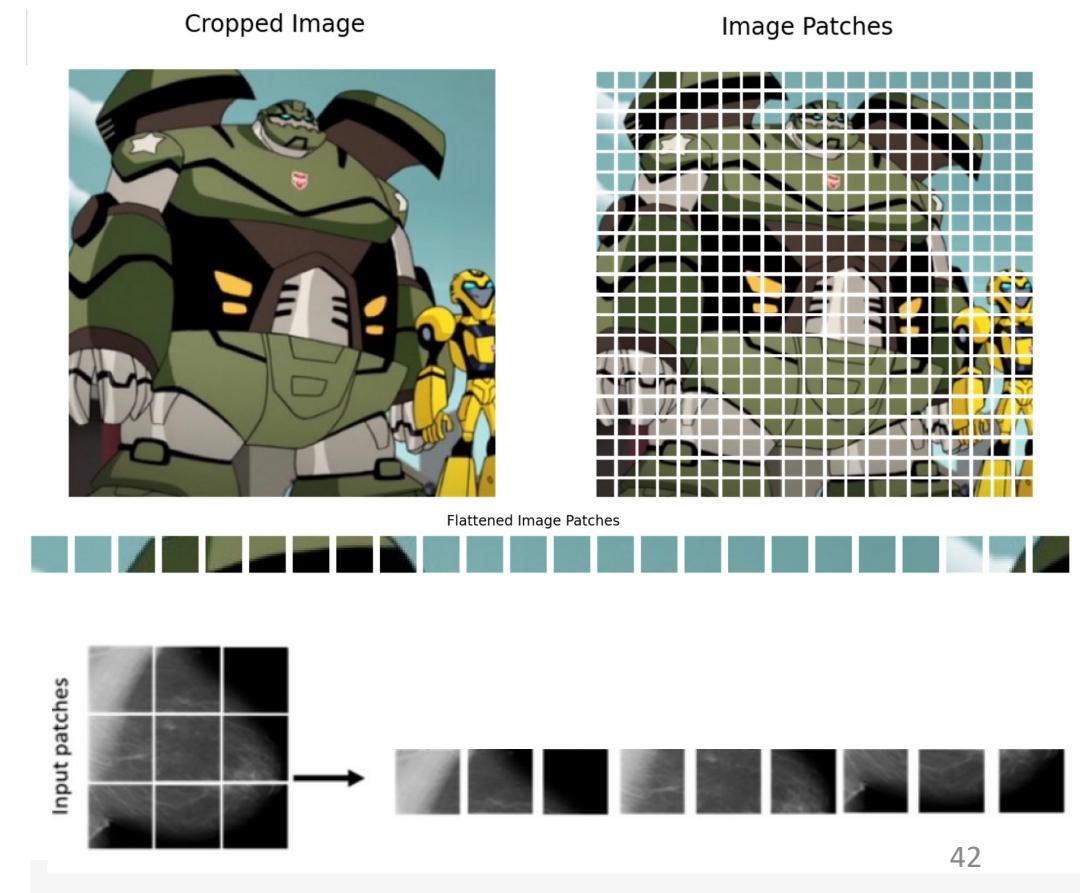
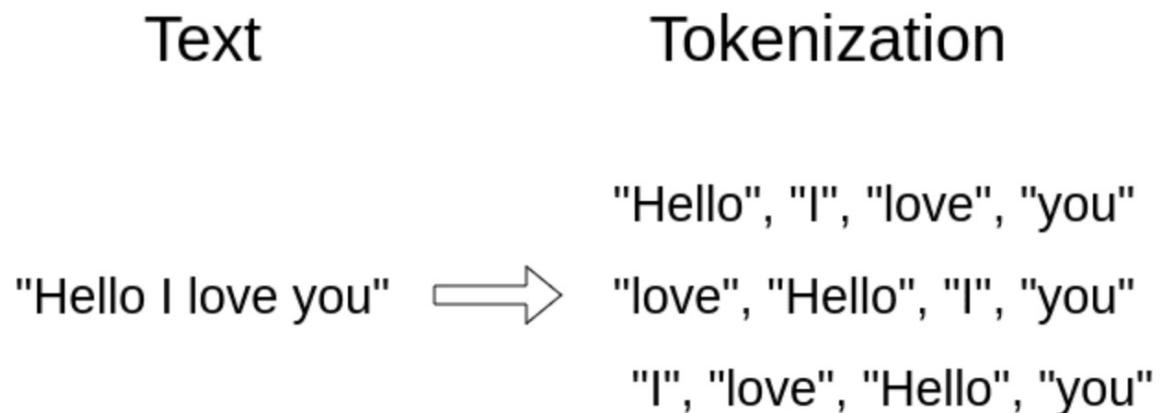


# Outline

- I. Recurrent Neural Networks (RNNs)
- II. Attention models & transformers
  - a) Context
  - b) **Transformer block**
  - c) Medical image segmentation

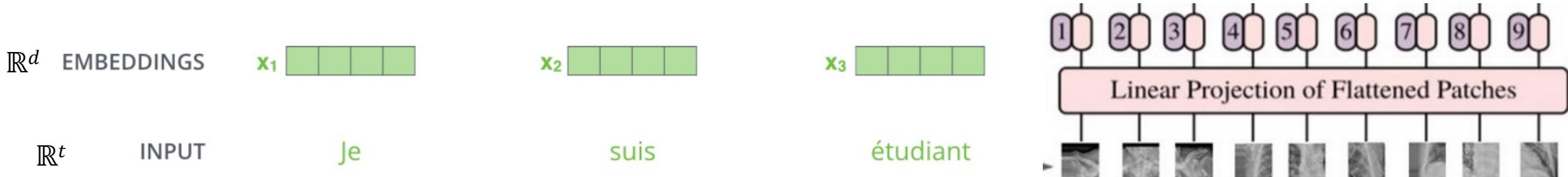
# From sequence to set

- A sequence of elements → a **set** of tokens, no order
  - Token: primitives, elementary elements of data
    - Text: token are e.g. words
    - Image: token are e.g. patches



# Input embedding

- Token: input vector in  $\mathbb{R}^t$ 
  - Word:  $t = |\mathcal{V}|$ ,  $\mathcal{V}$  vocabulary
  - Image patch:  $t = s^2$ , where  $s$  is the patch size
- Input embedding: linear projection  $\mathbb{R}^t \rightarrow \mathbb{R}^d : e_i = x_i W^e$



# Positional encoding

- Sequence → set of token:
  - Permutation invariant
  - Loosing structural information from data
- Recovering structure: **positional encoding (PE)**
  - Mapping token position  $t$  to a vector  $\mathbf{p}_t \in \mathbb{R}^d$
  - Seminal PE: sinusoidal

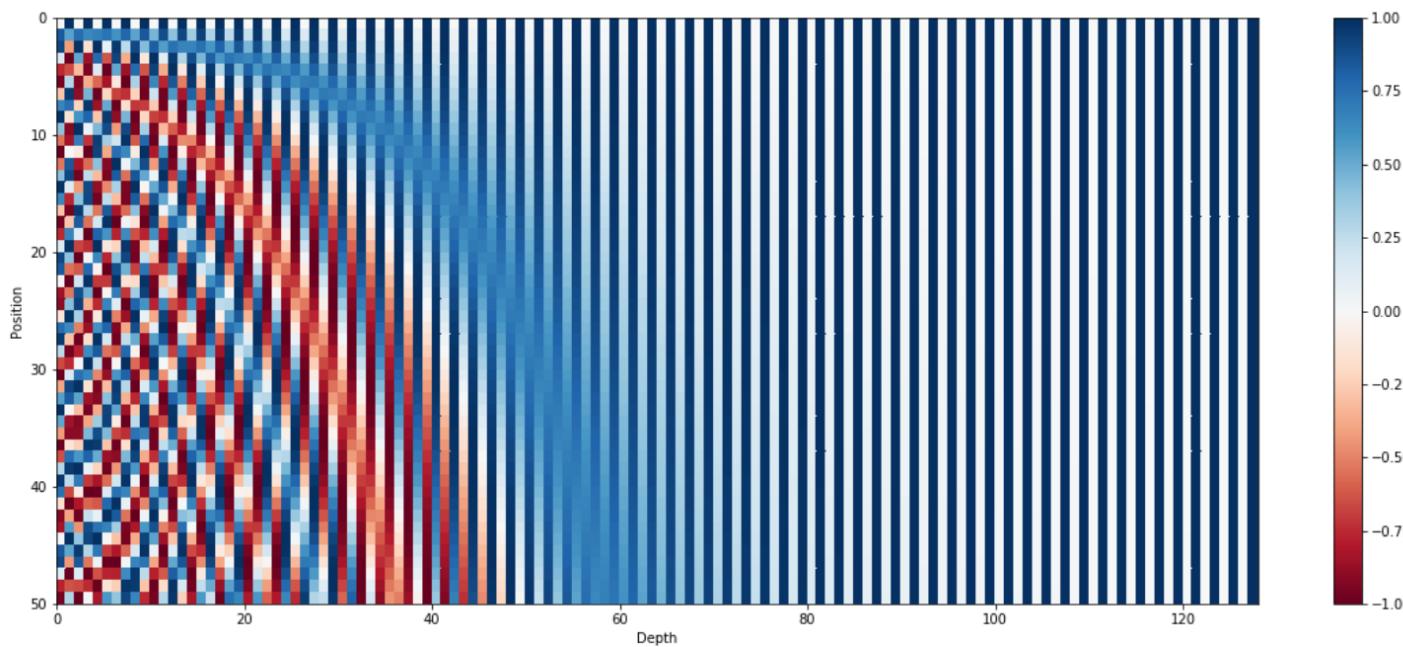
$$\vec{p}^{(i)} := \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k + 1 \end{cases}$$

$$\omega_k = \frac{1}{10000^{2k/d}}$$

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

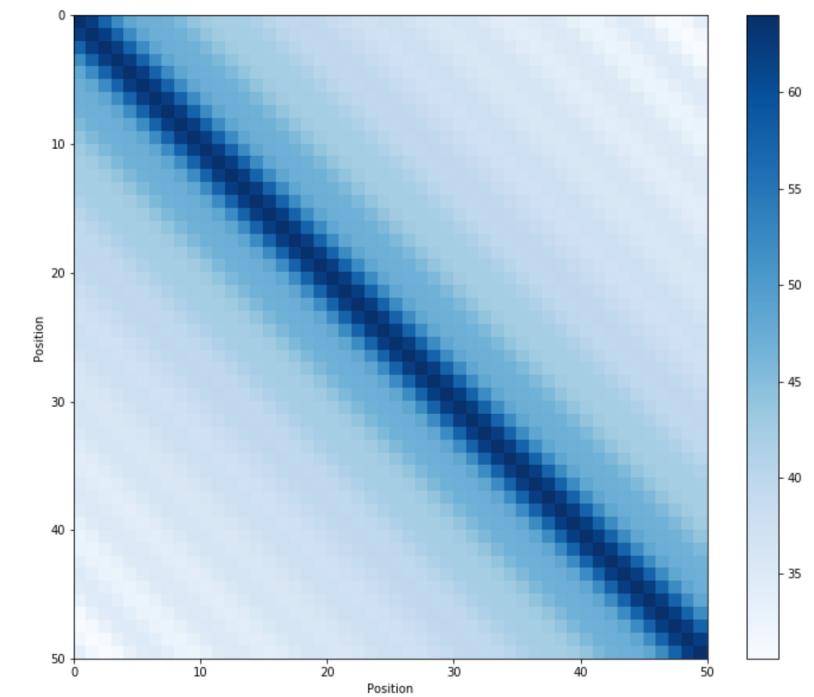
# Sinusoidal positional encoding

- Unique vector  $\mathbf{p}_t$  for each position  $t$
- $p_t(i) \in [-1;1]$ : natural normalization



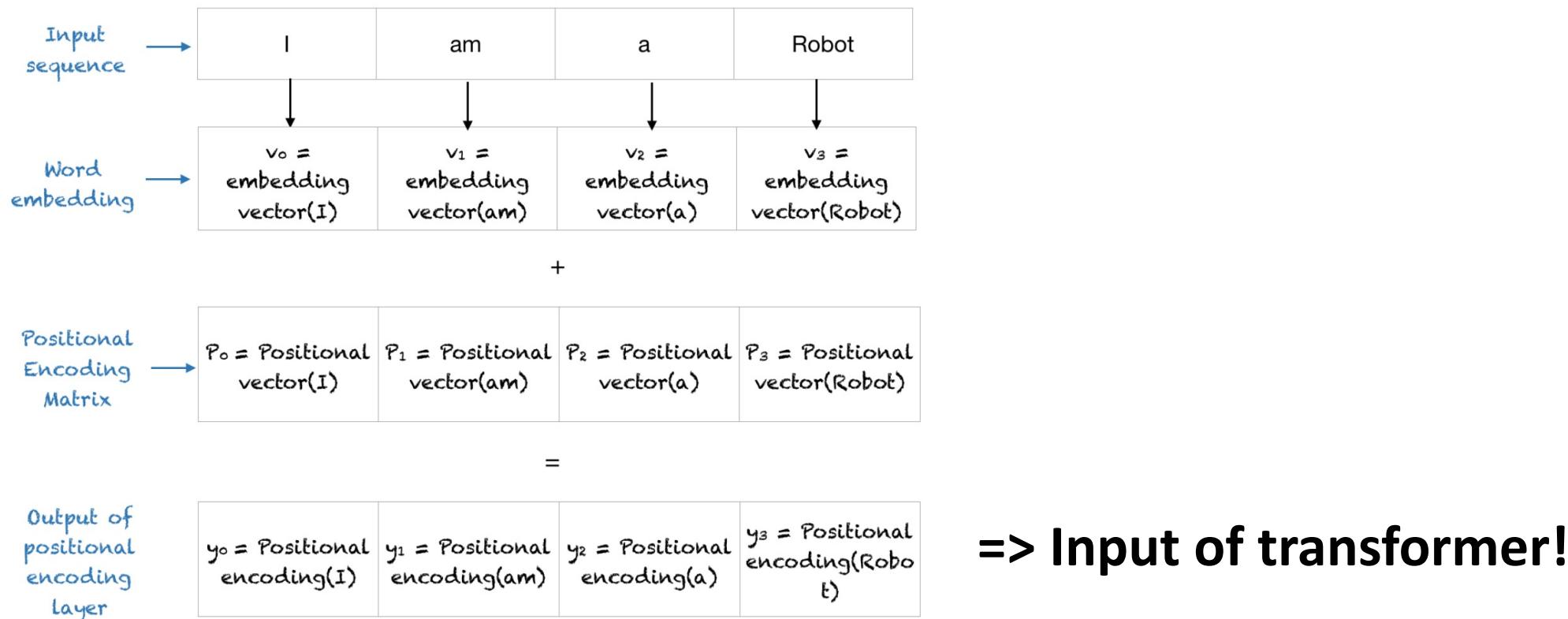
$d=128$ , max length of token set = 50

- Models relative position
- Positional similarity:  
$$K = P P^T$$

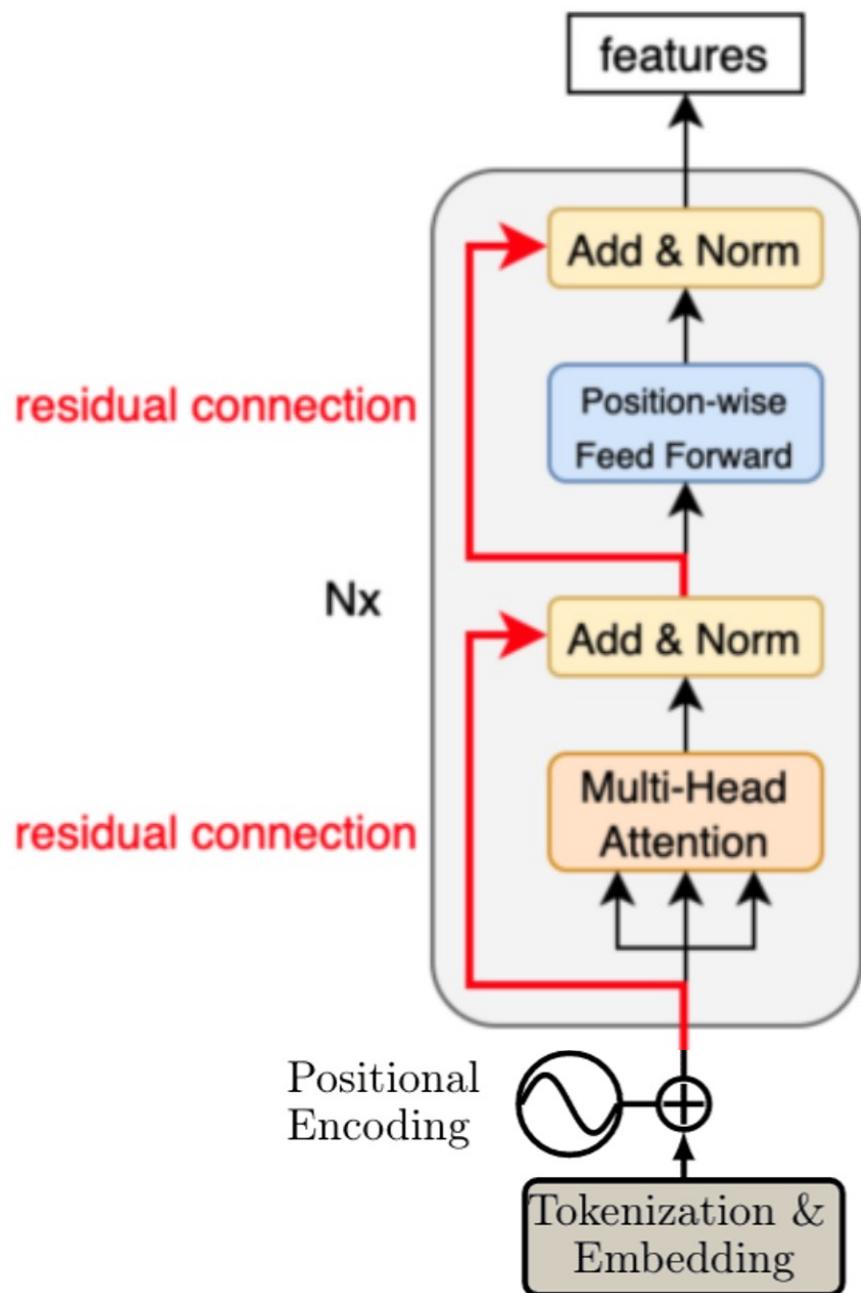


# Positional encoding

- Other possible encoding, can be learned
- Final embedding :



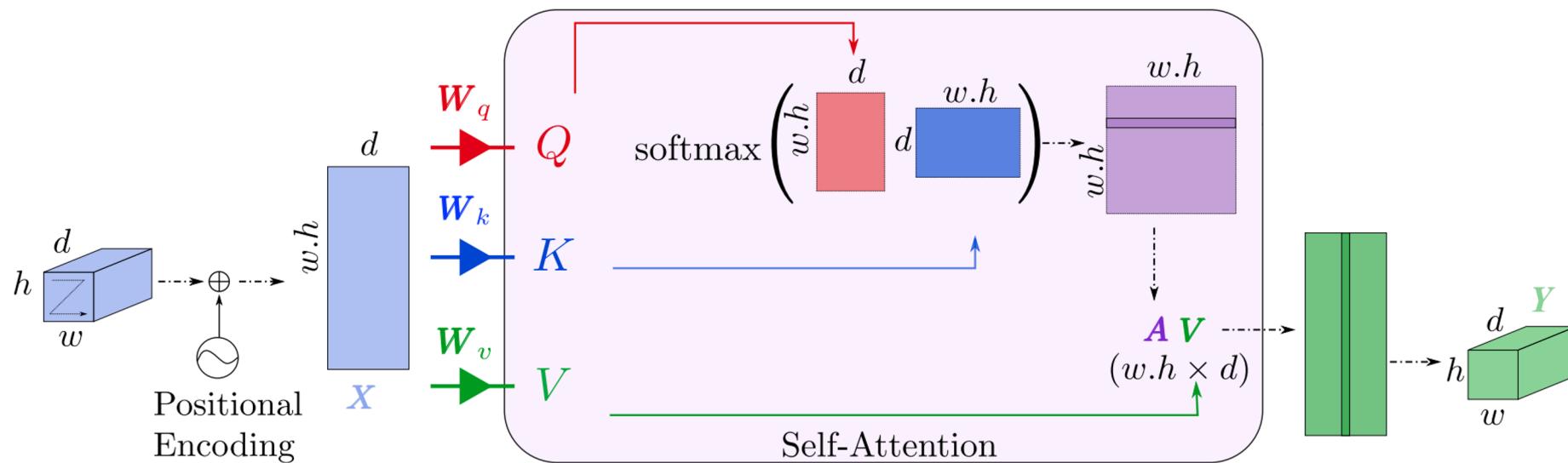
# Transformer [4] : the encoder



- A stack a  $N$  transformer blocks
  - Input a set of embedded tokens
  - Output: a set of re-embedded tokens

[4] Attention Is All You Need. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. NeurIPS 2017.

# Self-attention



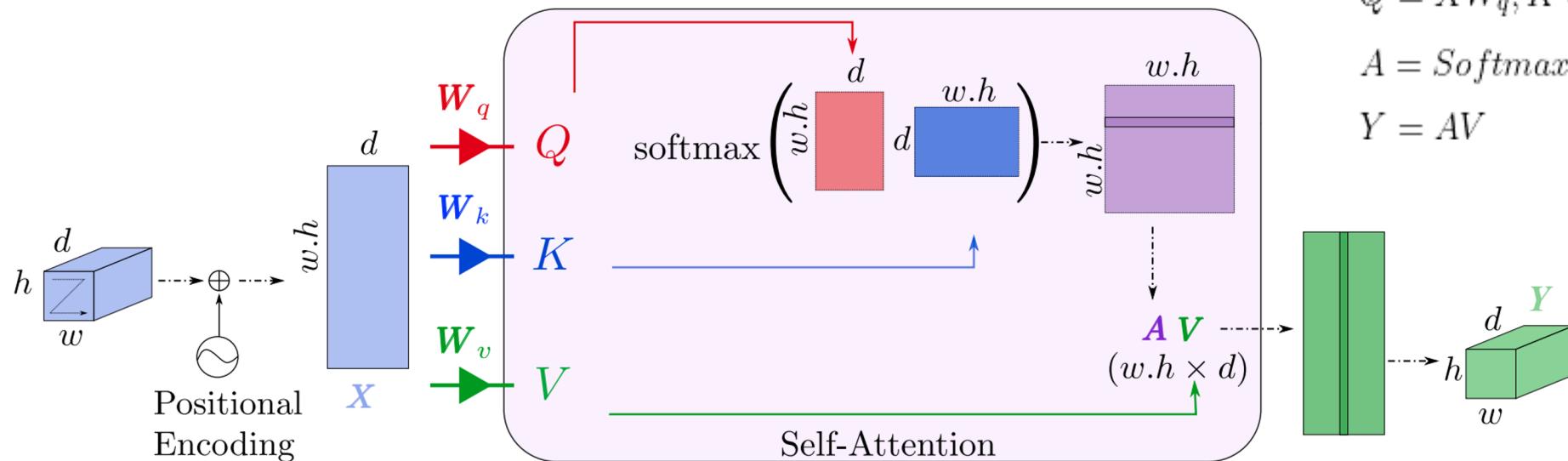
$$X \in \mathbb{R}^{wh \times d}, W_q \in \mathbb{R}^{d \times d}, W_k \in \mathbb{R}^{d \times d}, W_v \in \mathbb{R}^{d \times d}$$

$$Q = XW_q, K = XW_k, V = XW_v$$

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

$$Y = AV$$

# Self-attention: conclusion



$$X \in \mathbb{R}^{wh \times d}, W_q \in \mathbb{R}^{d \times d}, W_k \in \mathbb{R}^{d \times d}, W_v \in \mathbb{R}^{d \times d}$$

$$Q = XW_q, K = XW_k, V = XW_v$$

$$A = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)$$

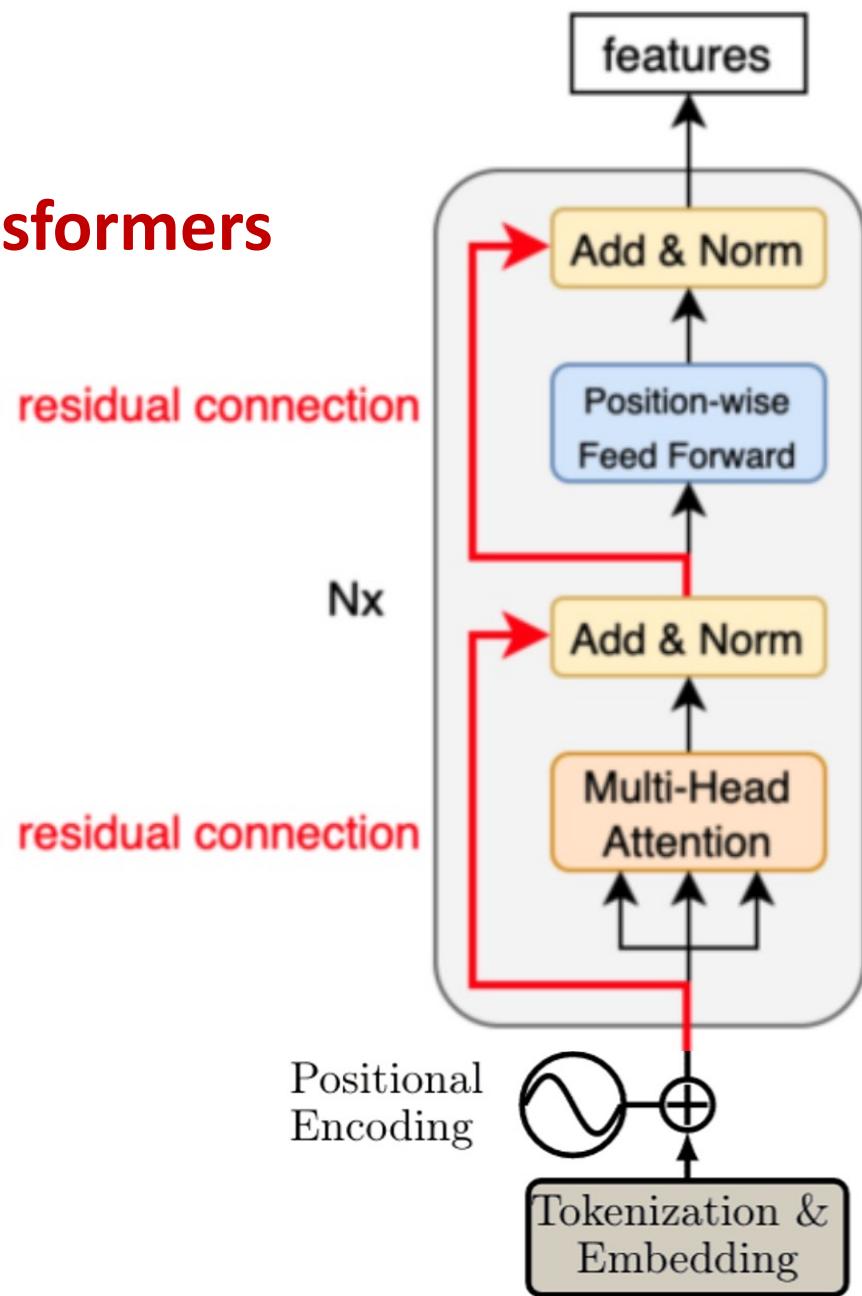
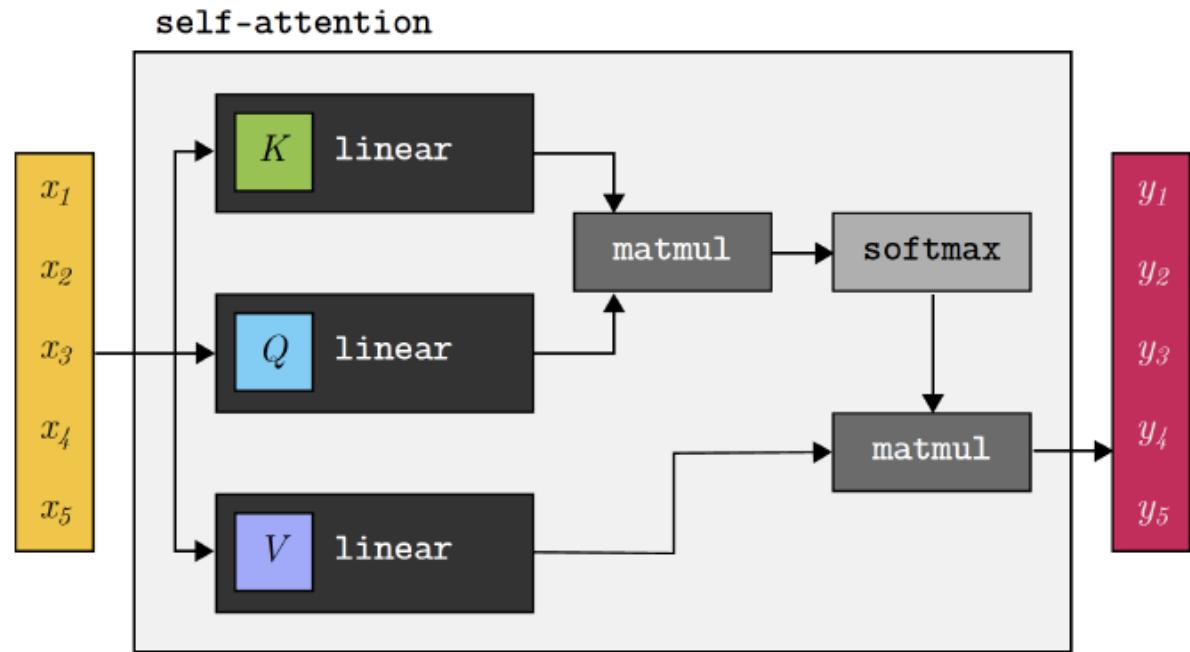
$$Y = AV$$

- Each token  $y_i$  in  $Y$ : computed a linear combination of  $v_i$ 
  - Enables to model **global interactions** between  $v_i$  tokens: full contextual information
  - $\neq$  ConvNets in vision, interactions limited by the size of the receptive field
  - $\neq$  RNNs for sequence processing, interactions limited by vanishing gradients
- **Self attention:  $O(N^2)$  complexity**
  - Expensive (or impossible) for large N

# Transformer: self attention

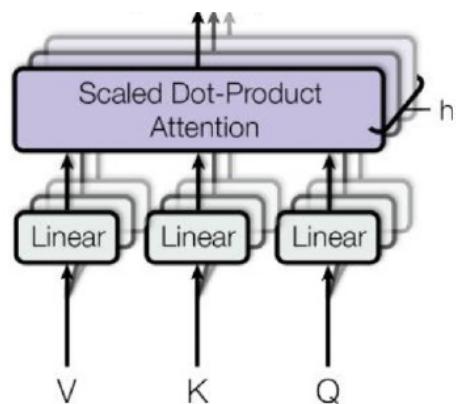
- **The most important and specific module in transformers**

- Project the input set into 3 sets
  - Query: sought info
  - Key: context elements
  - Value: retrieved



# Multi-headed attention

- High-level idea: multiple self-attention in parallel
- Each head: attend to different parts
- Combine the heads' outputs



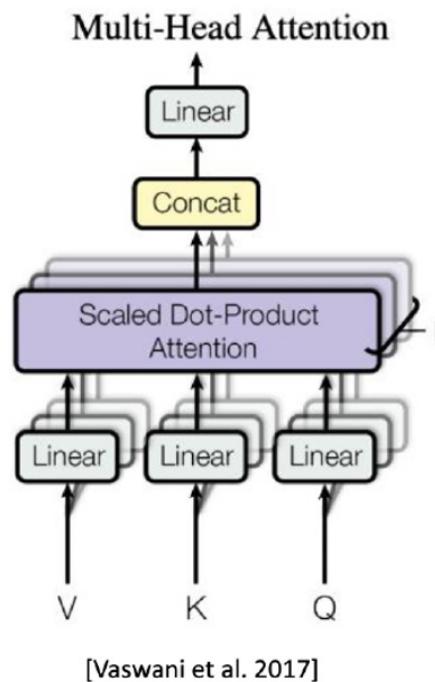
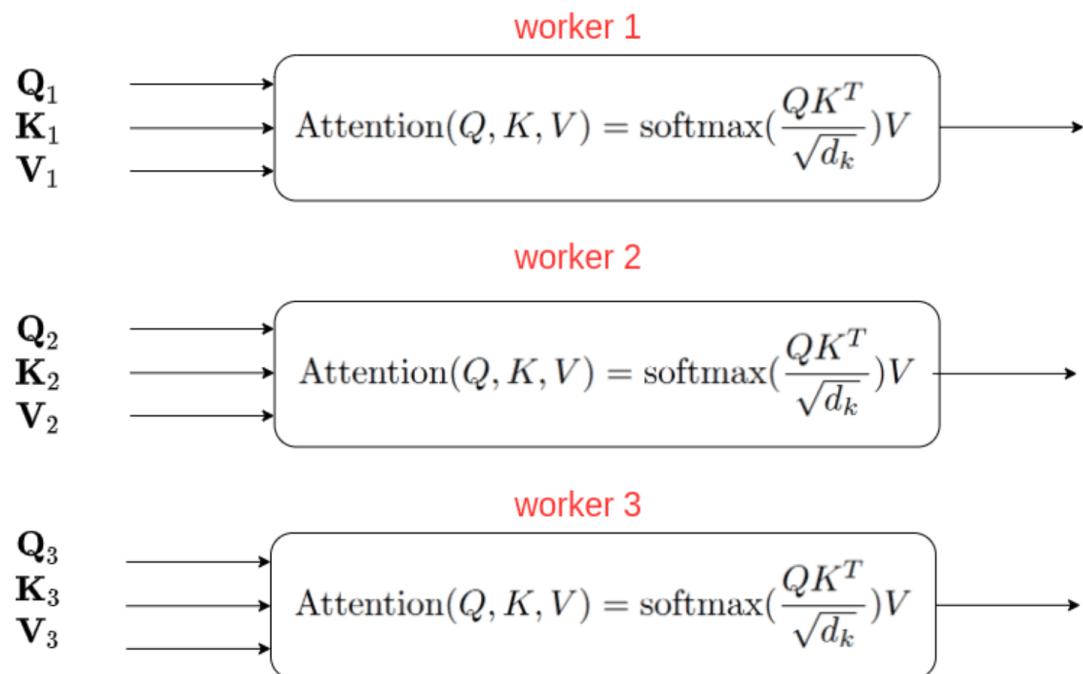
[Vaswani et al. 2017]



Wizards of the Coast, Artist: Todd Lockwood

Credit: Anna Goldie

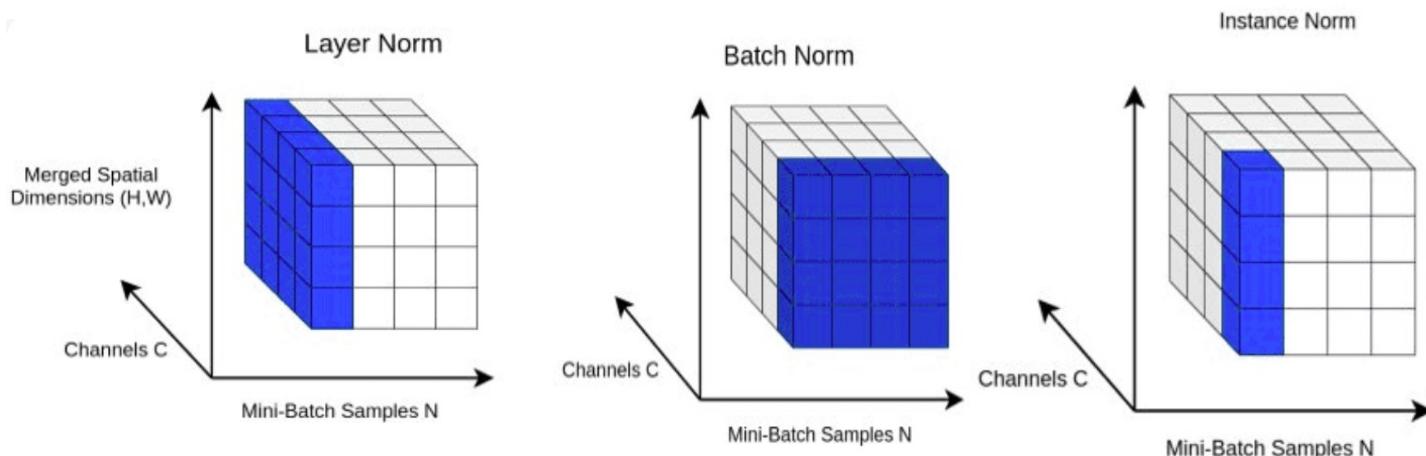
# Multi-headed attention



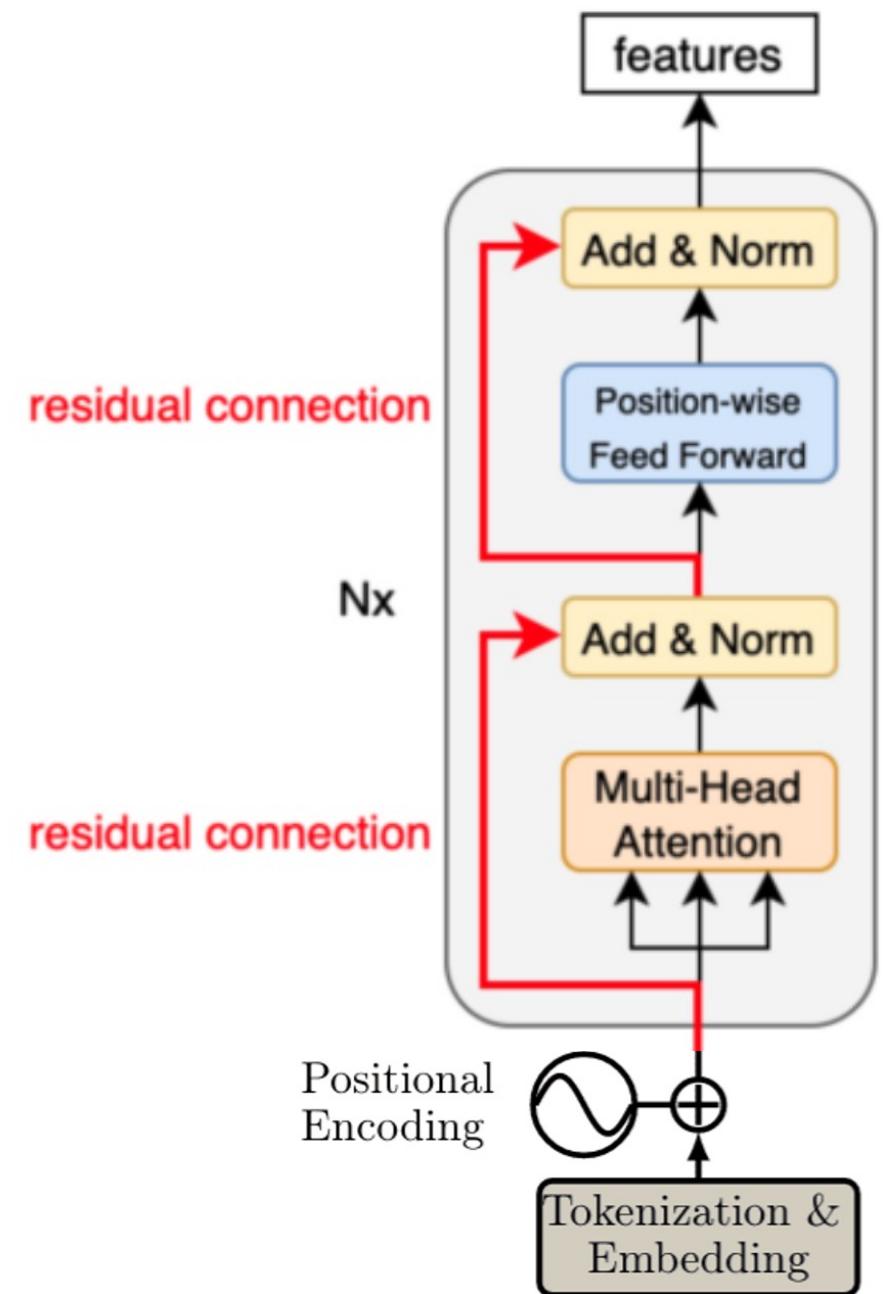
- Concatenate the heads' outputs
- Use a linear layer: desired output size

# Layer normalization

- Normalization on joint channel and spatial dimensions



- Stabilize training, faster convergence



# Layer normalization

- Normalization on joint channel and spatial dimensions

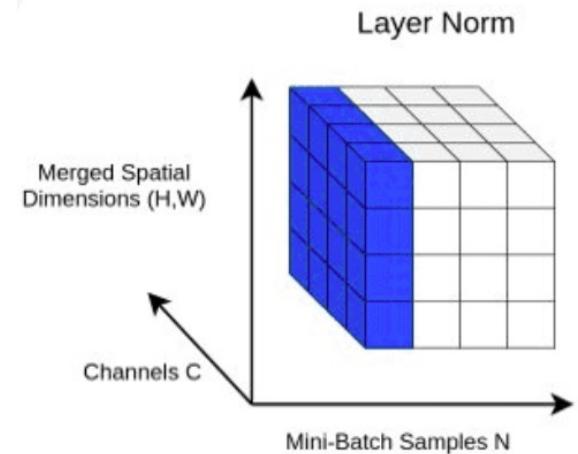
$$\mu_n = \frac{1}{K} \sum_{k=1}^K x_{nk}$$

$$\sigma_n^2 = \frac{1}{K} \sum_{k=1}^K (x_{nk} - \mu_n)^2$$

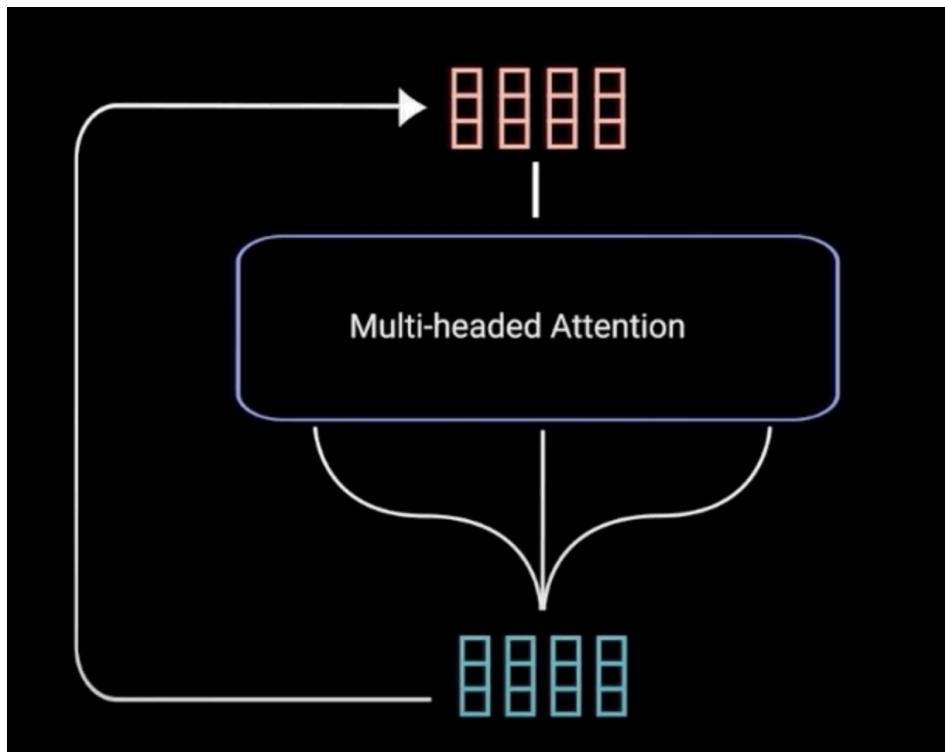
$\beta, \gamma$  learnable parameters

$$\hat{x}_{nk} = \frac{x_{nk} - \mu_n}{\sqrt{\sigma_n^2 + \epsilon}}, \hat{x}_{nk} \in R$$

$$\text{LN}_{\gamma, \beta}(x_n) = \gamma \hat{x}_n + \beta, x_n \in R^K$$



# Layer normalization + residual connections



LayerNorm( $\text{Input} + \text{Output}$ )

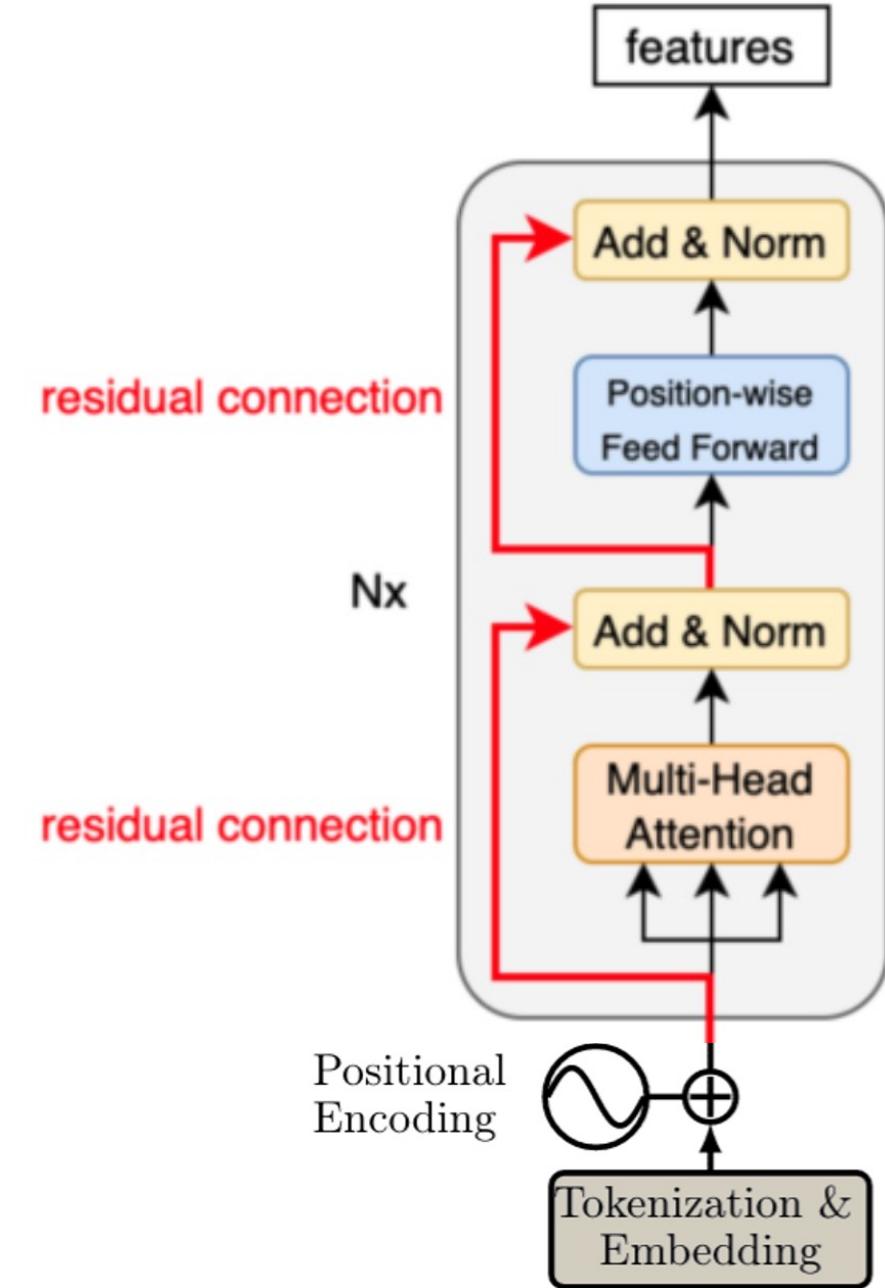
## Residual connections

- Better gradient flow (vanishing gradients)
- Leverage input encoding, e.g. PE

# Feed-Forward Network (FFN)

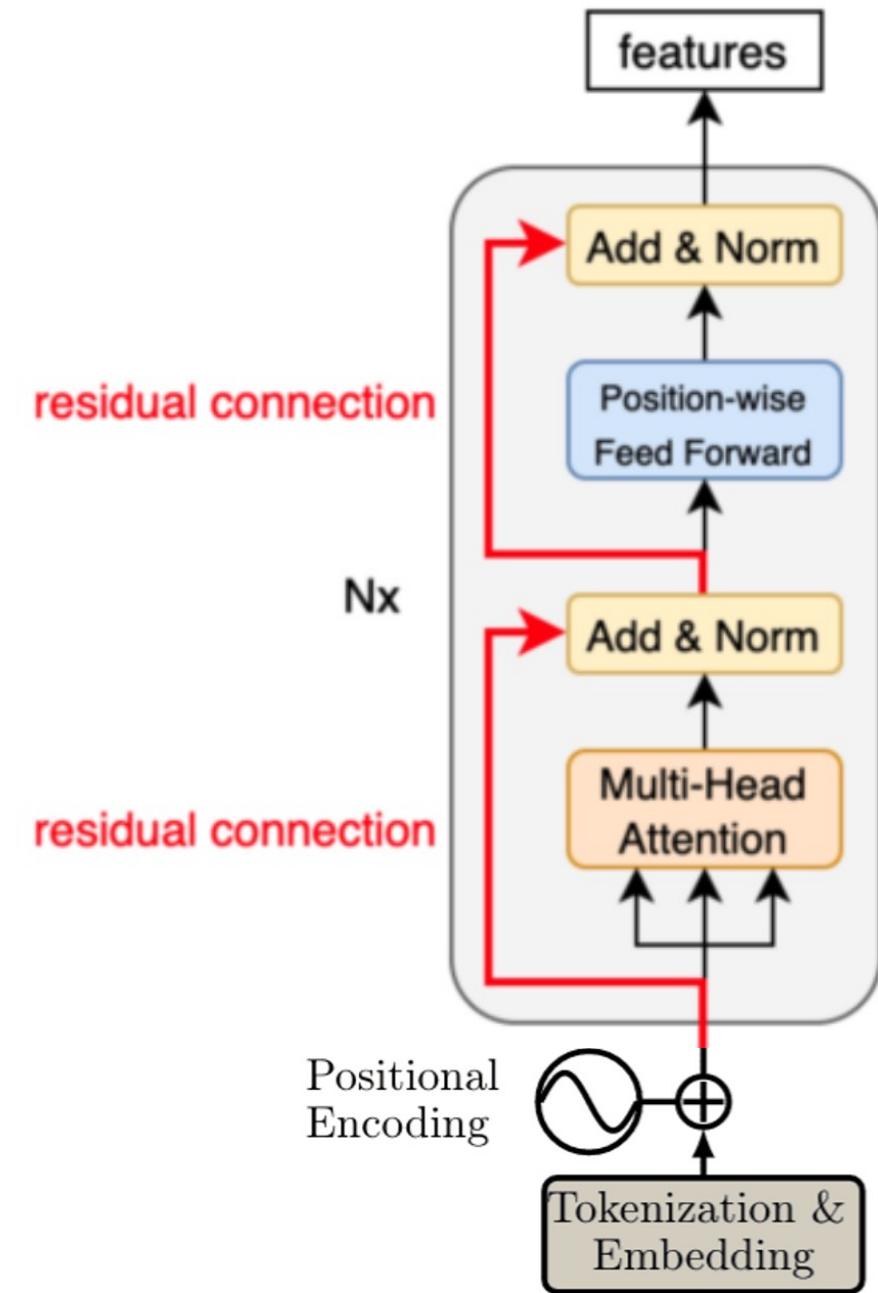
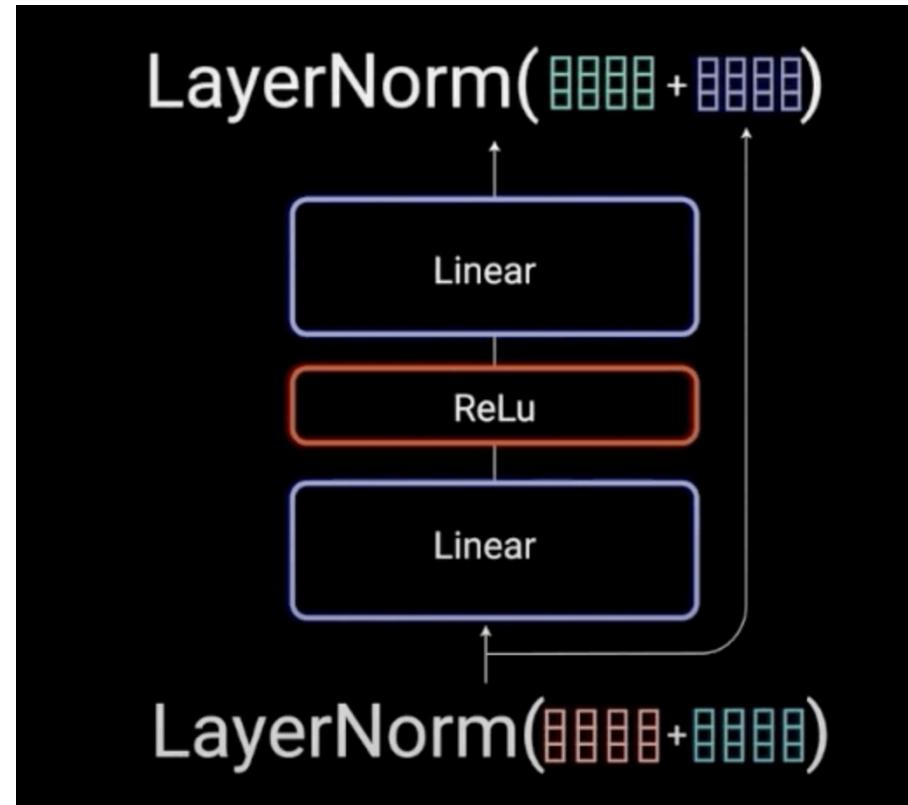
- Position-wise FFN: applied to each token separately and identically

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



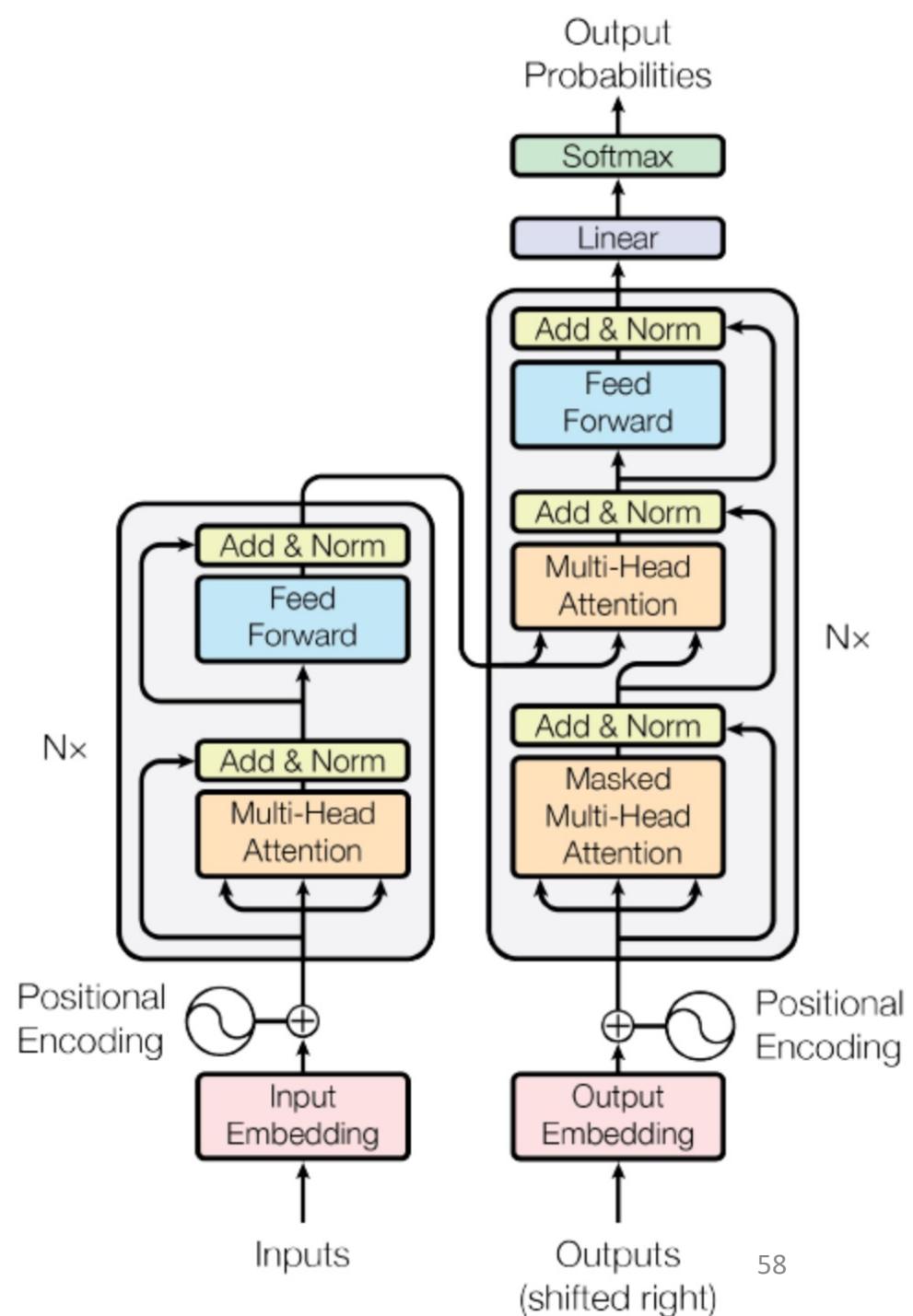
# FNN + residual Layer Norm

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$



# Transformer: conclusion

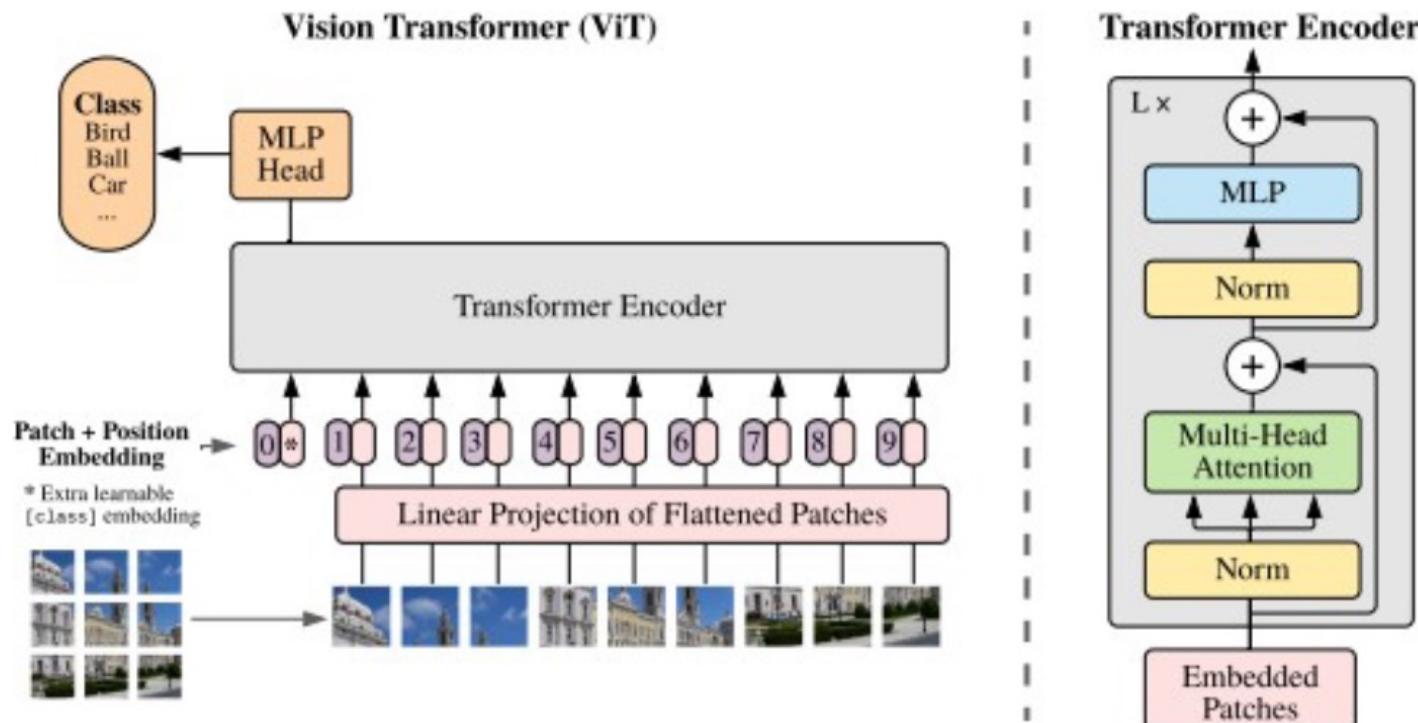
- Importance of attention: global interactions between tokens
- On the other hand relaxes inductive biases
  - e.g. ConvNets translation equivariant
    - vs transformers permutation equivariant
  - More flexibility to learn adequate mapping
  - Needs more data



# Outline

- I. Recurrent Neural Networks (RNNs)
- II. Attention models & transformers
  - a) Context
  - b) Transformer block
  - c) Medical image segmentation

# Vision Image Transformer (ViT) [2]



- Direct application of transformer's encoder for images
- Learned on JFT ( $300.10^6$  images)
- Extra learnable token: used for class prediction
  - “Learned” pooling wrt visual tokens

# Transformer in segmentation

Vanilla idea: use ViT encoder, add a classification layer for each patch (token) => SETR [9]

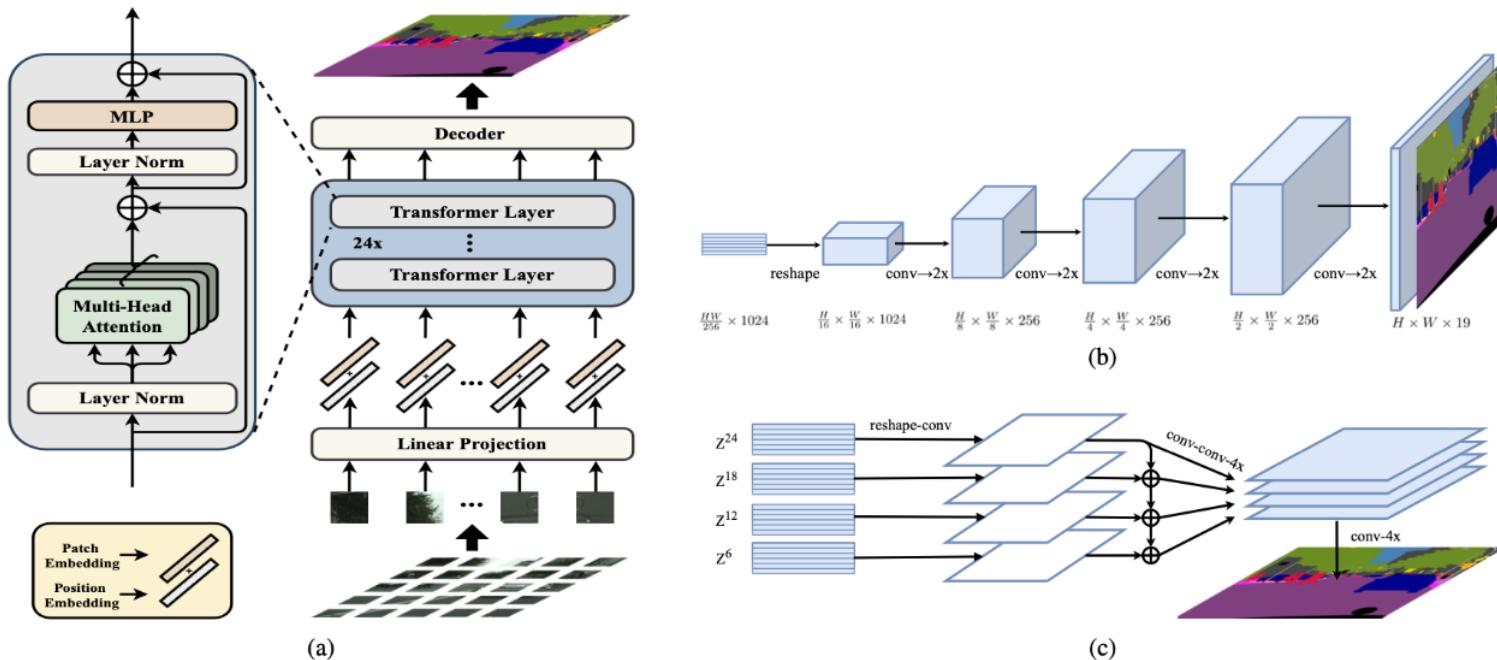


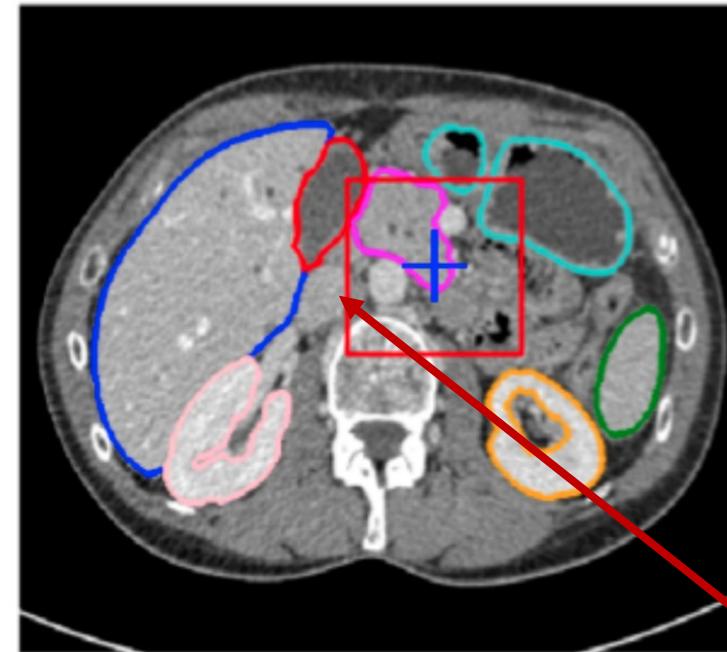
Figure 3: SETR architecture and its variants adapted from [5]. (a) SETR consists of a standard Transformer. (b) *SETR-PUP* with a progressive up-sampling design. (c) *SETR-MLA* with a multi-level feature aggregation.

# Transformer in medical image segmentation

## Organ segmentation

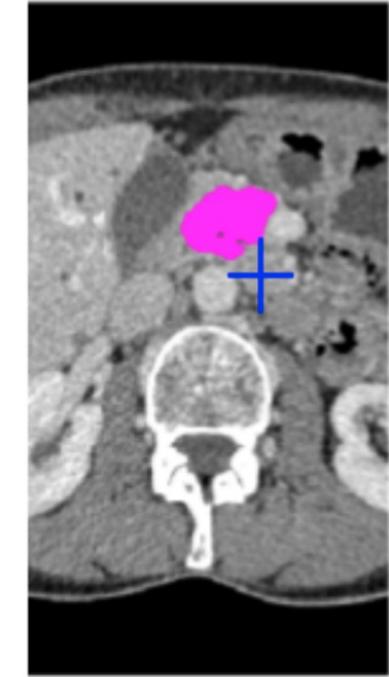


## Ex: Pancreas segmentation



a) Ground Truth

U-Net  
receptive  
field



c) U-Net

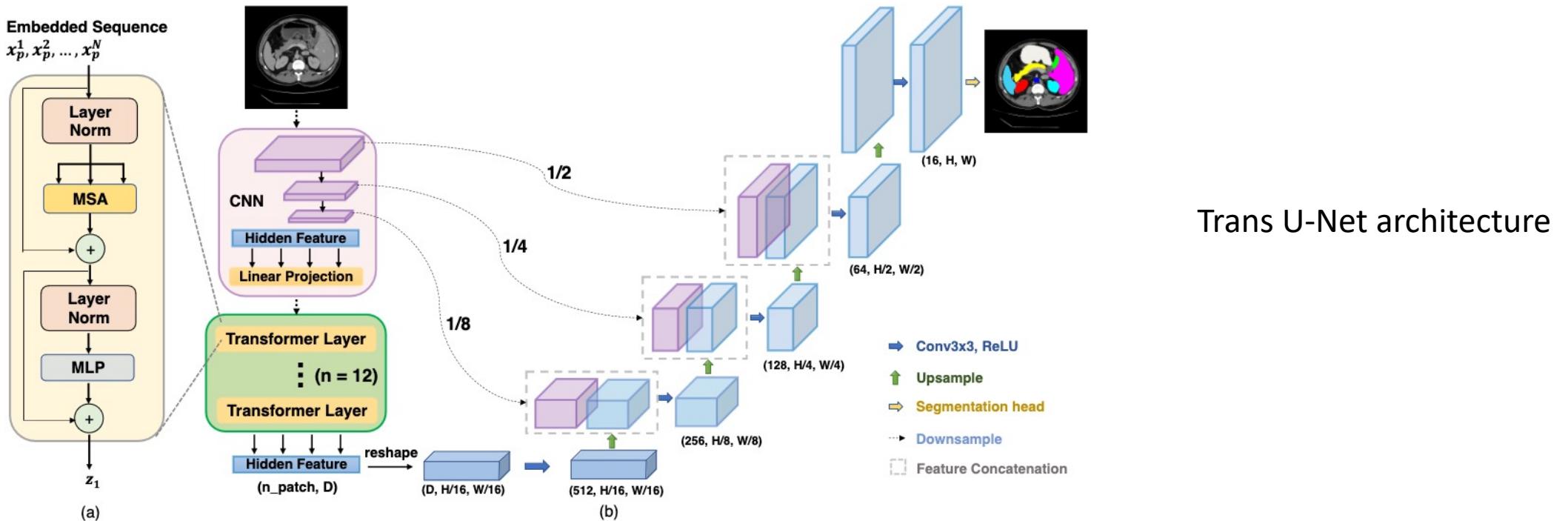
Long-range dependencies: crucial context in segmentation

Main challenge of transformers: self attention complexity -  $O(N^2)$

- Expensive (or impossible) for large N => critical for large 2D images, 3D volumes

# Hybrid conv / transformers architectures

- Trans U-Net [7], U-Transformer [8]: seminal works for using transformers in medical image segmentation
- Adding self-attention on U-Net's bottleneck
  - Inspired from non-local networks [9]



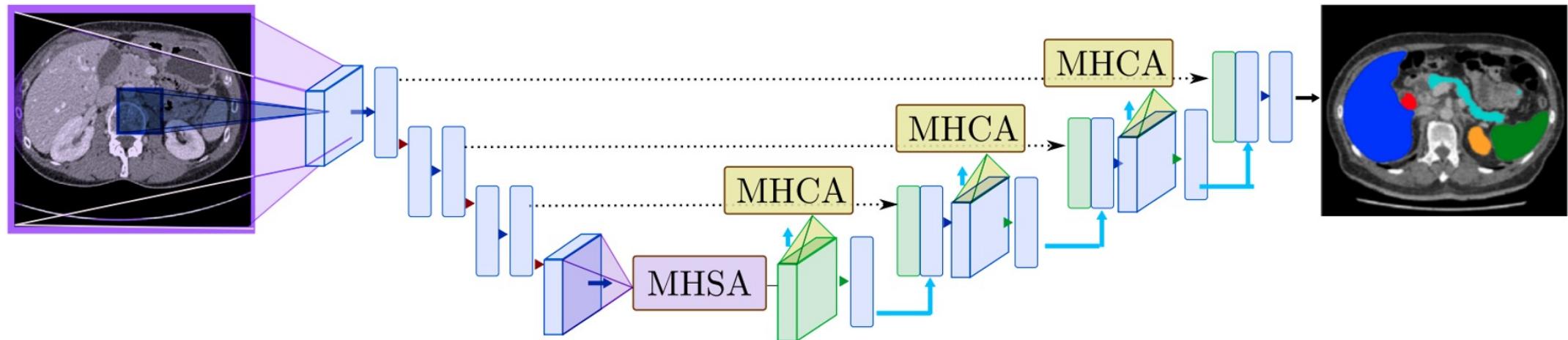
[7] TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation. J. Chen et.al. arXiv, Feb 2021.

[8] U-Net Transformer: Self and Cross Attention for Medical Image Segmentation. O. Petit, N. Thome, C. Rambour, L. Soler. arXiv, March 2021.

[9] Non-local Neural Networks. X. Wang, R. Girshick, A. Gupta, K. He. CVPR 2018.

# U-Transformer [8]

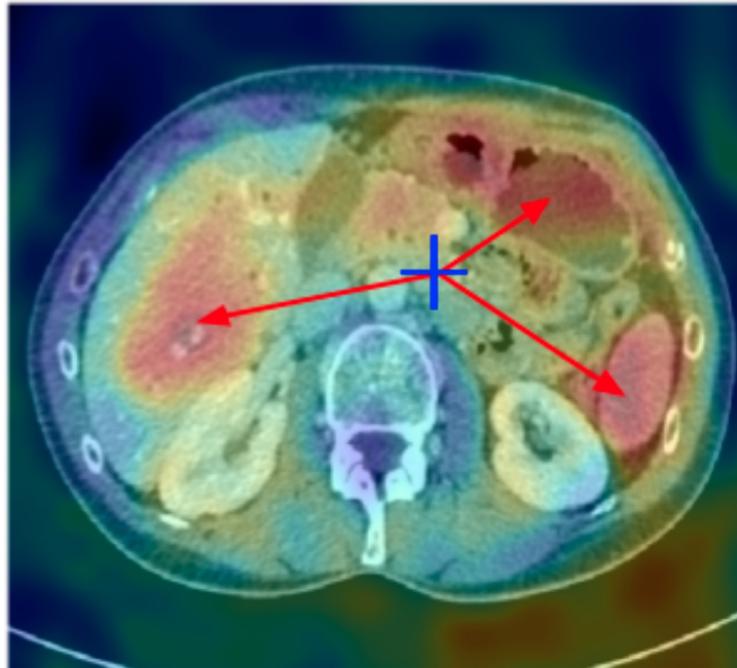
- **U-Transformer:** self and cross attention in medical image segmentation
  - Self-attention in bottleneck
  - Cross attention to improve super-resolution in skip connections



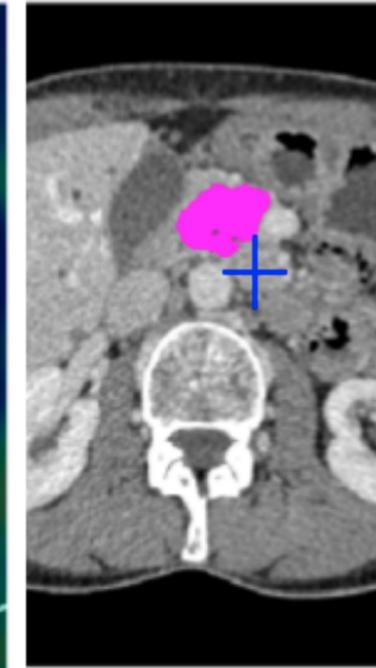
# U-Transformer illustration



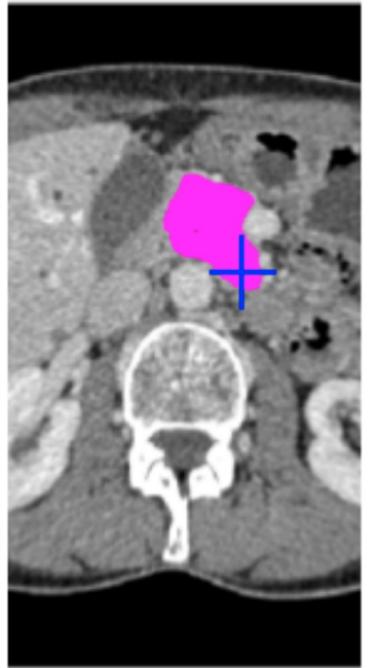
a) Ground Truth



b) Attention map



c) U-Net

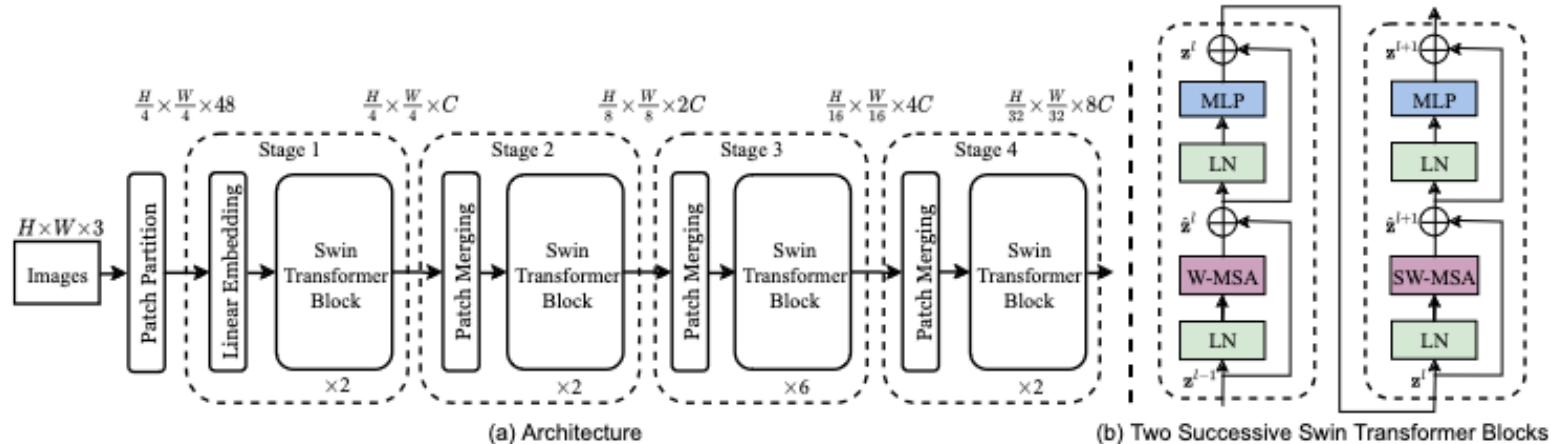
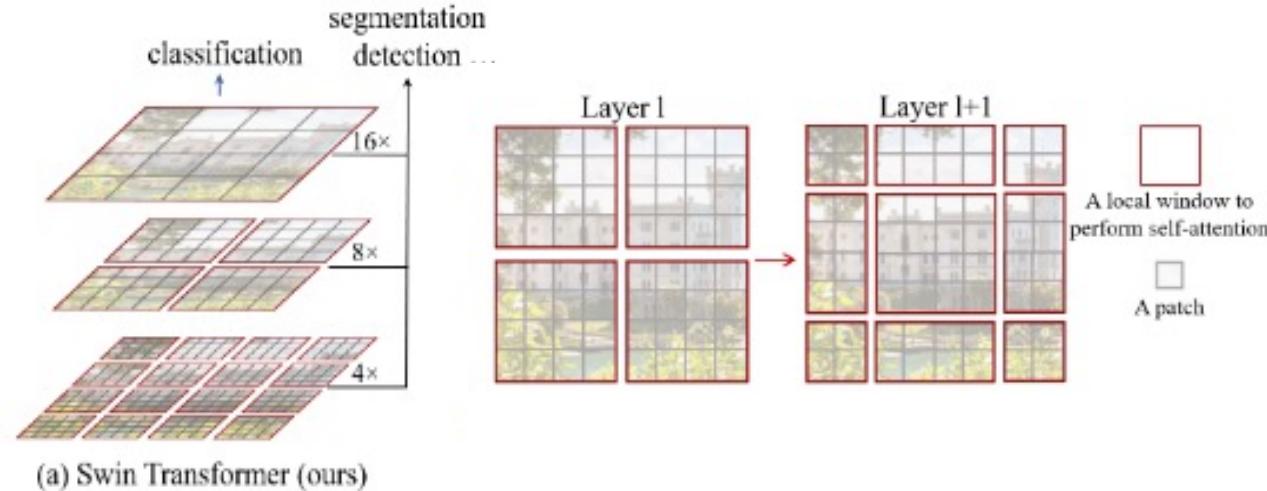


d) U-Transformer

# Full transformers in segmentation

## Motivation: breaking self-attention complexity

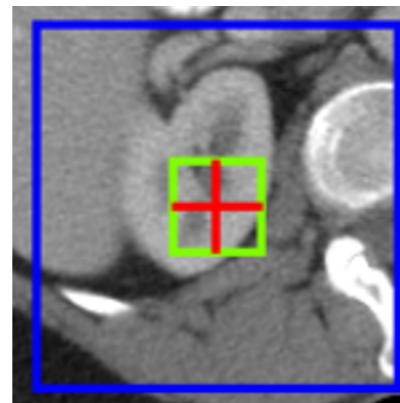
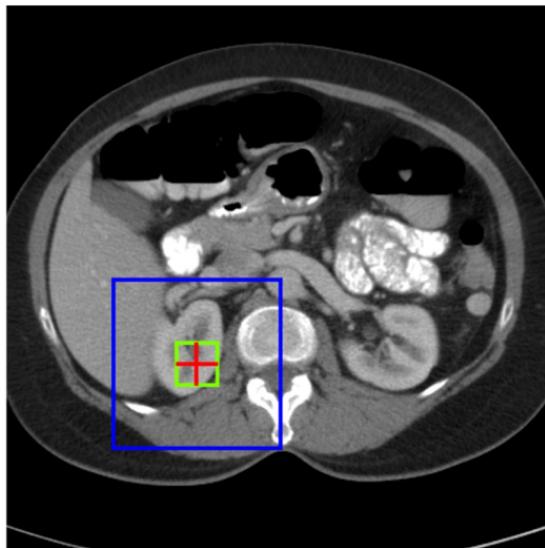
- Swin-Transformer [5]
  - Multi-resolution transformer
    - Local attention in lower-layers
      - Shifted windows at layers  $l/l+1$
    - Patch merging => larger receptive field



# 3D medical image segmentation

## Challenge: input volume size

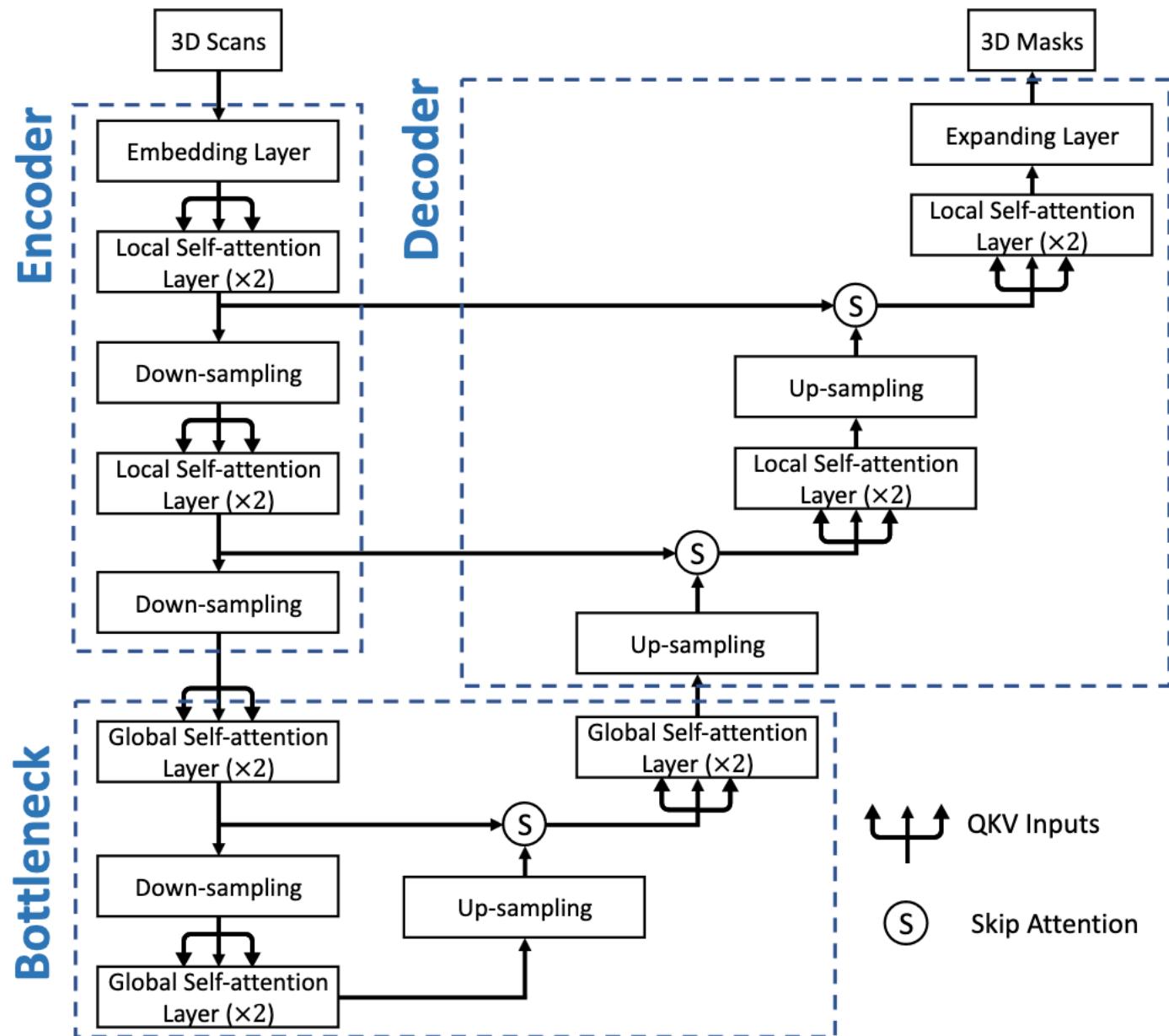
- Intractable memory requirements (180Gb for U-Net with image 512x512x256)
- **Common strategy:** train on 3 crops



- **Full context lost**
- **Even on patch: full context challenging!**

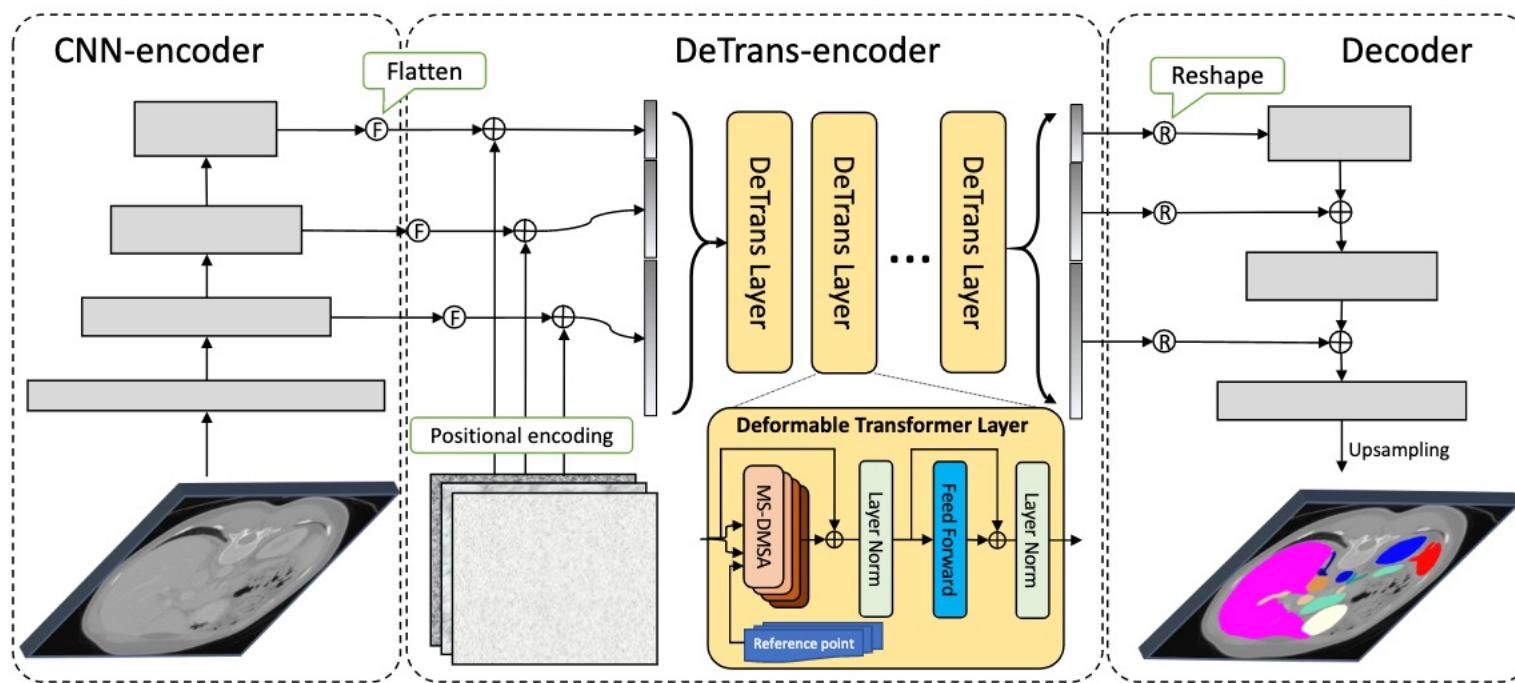
# nn-Former [11]

- Global self-attention in bottleneck
- Local self-attention in higher-resolution feature maps
  - ~ 3D Swin-Unet
- **No context beyond patch**
- **No global context in high-resolution maps**



# CoTr: Convolutional NN and Transformer [12]

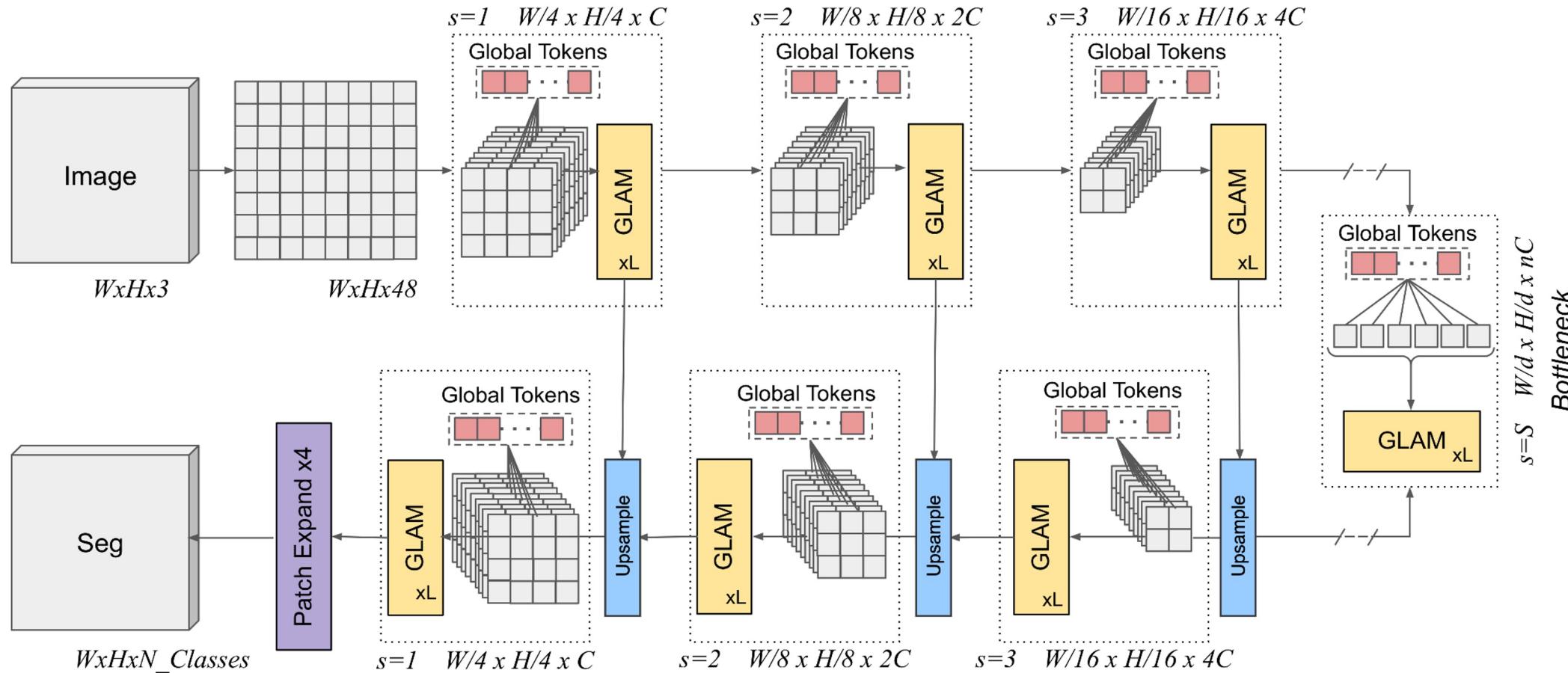
- **CoTr:** Conv encoder => flattened multi-scale feature
  - Deformable transformer encoder (DeTrans) in multi-res input
  - Several DeTrans layers, sent to conv decoder



[12] CoTr: Efficiently Bridging CNN and Transformer for 3D Medical Image Segmentation. Y. Xie, J. Zhang, C. Shen, Y. Xia. MICCAI 2021

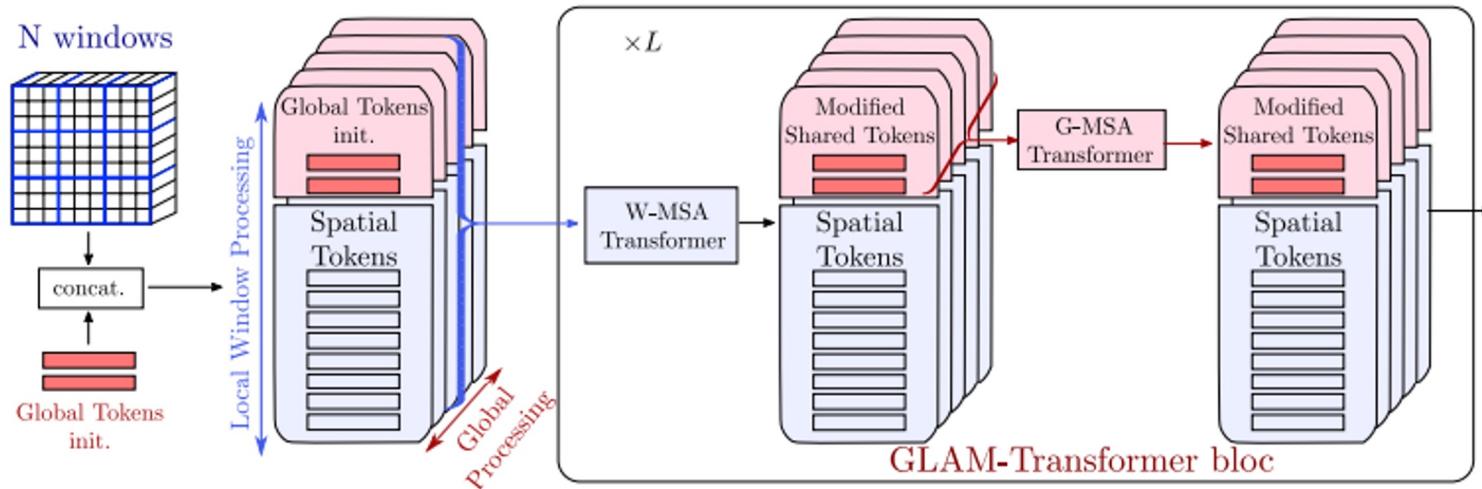
# Global attention in multi-resolution transformers (GLAM) [13]

- GLAM block in any multi-resolution architecture (e.g. Swin, nn-Former)



# GLAM block

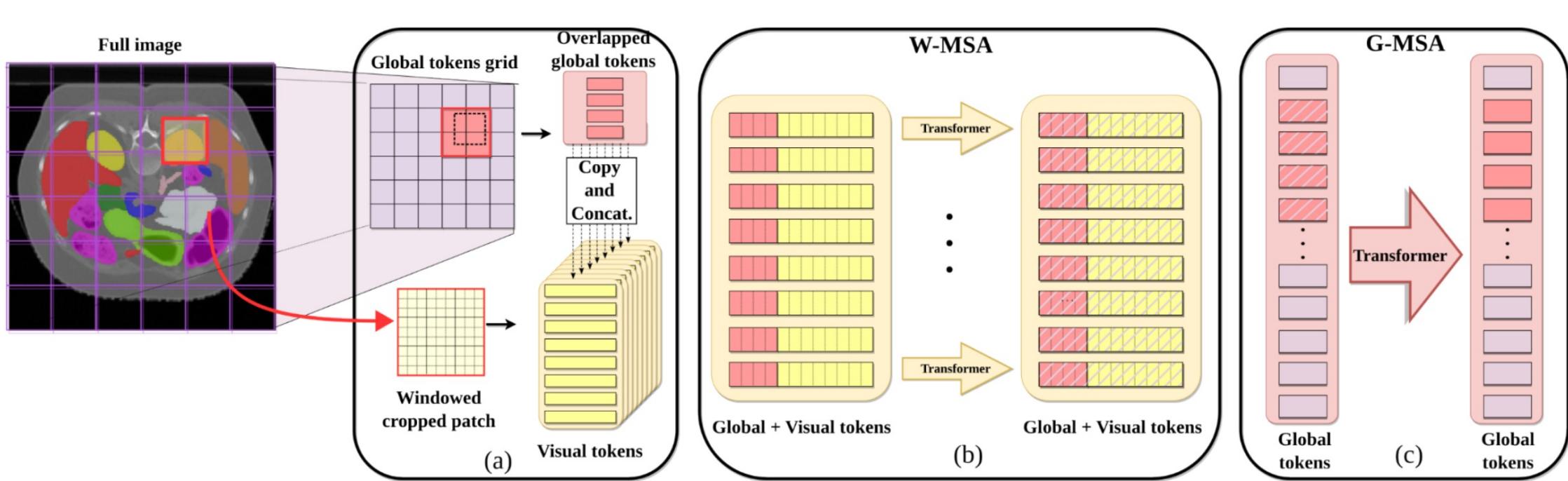
- Define learnable global tokens in each window, cf CLS in ViT
  - Window self-attention (W-MSA): attention between visual and global tokens
  - Global attention (G-MSA) between global token
- G-MSA: indirection between all visual tokens
  - Break computational complexity of full attention between visual token
  - But **enables full indirect interaction between them**



=> full attention even in high-resolution features!

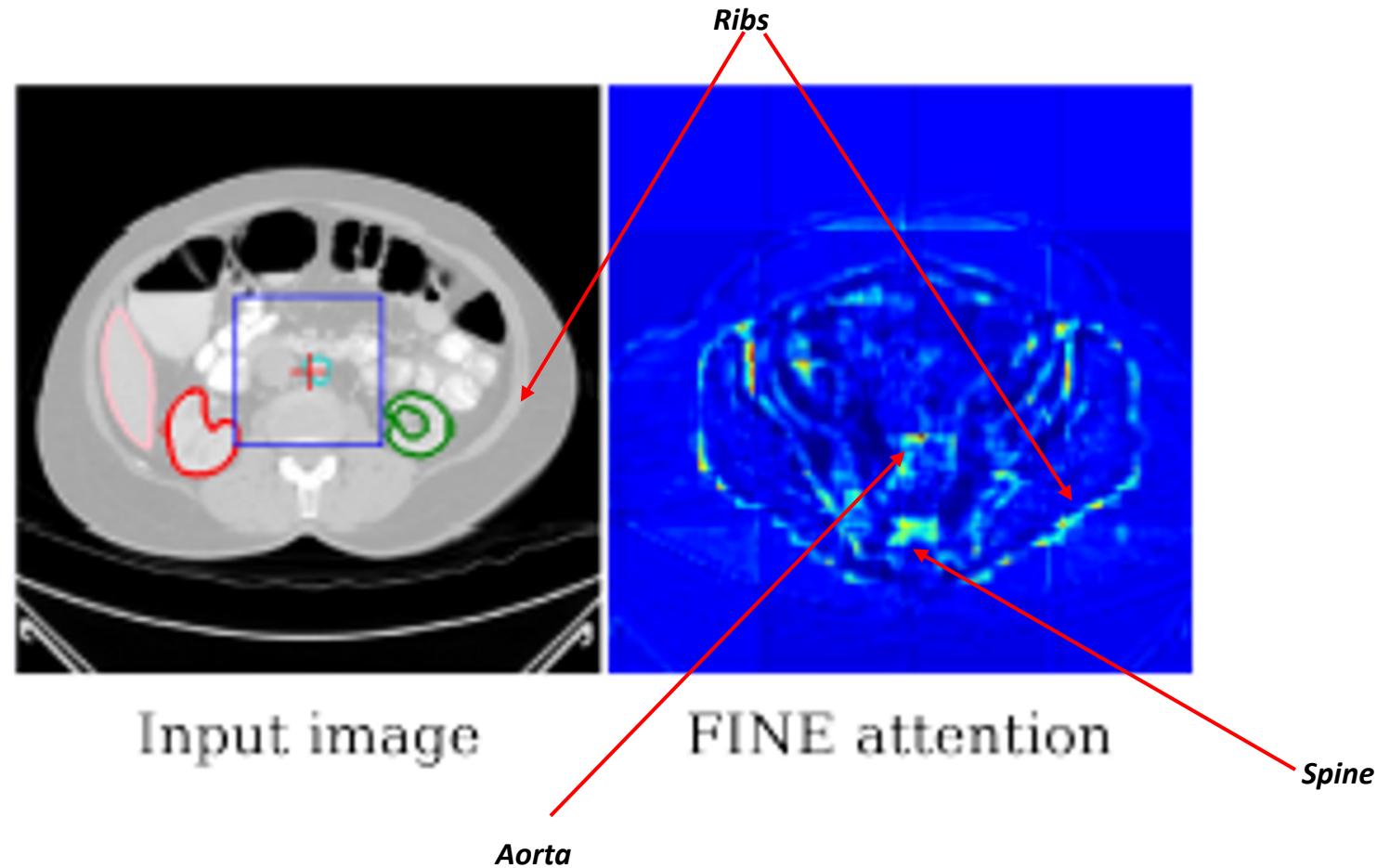
# FINE : Full resolution memory transformer [14]

- Extends GLAM for full context modelling in 3D segmentation
  - Global tokens for the full volume



=> (indirect) full interaction between all voxels!

# FINE results



Thank you for your attention!

Questions?