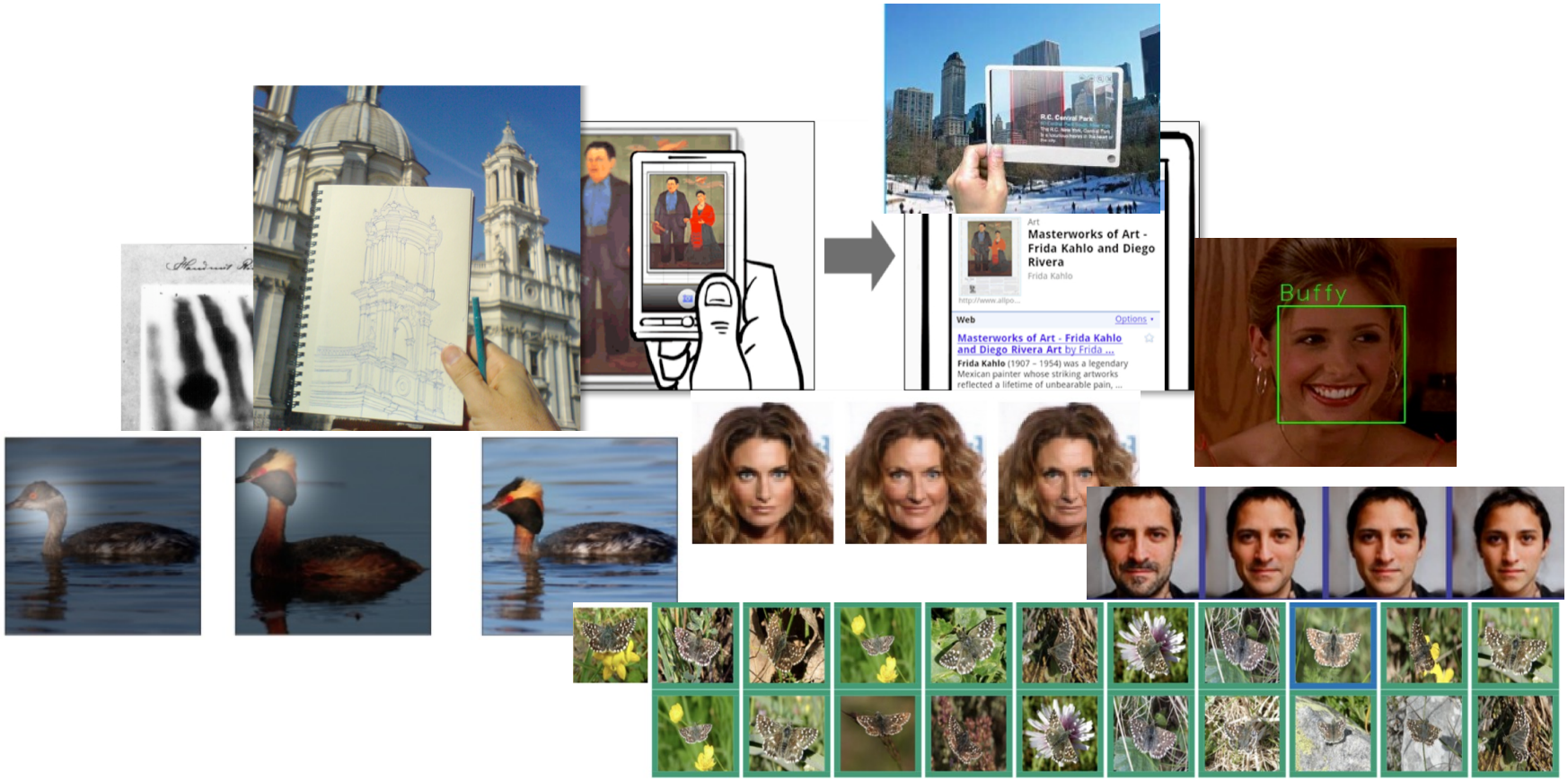


COURS RDFIA deep Image

Matthieu Cord
Sorbonne University

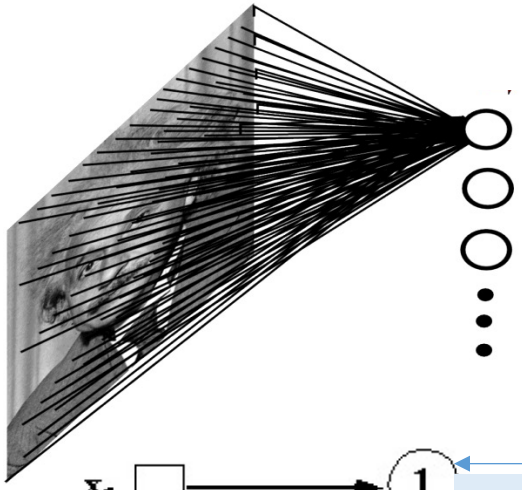
Course Outline – Week timeline

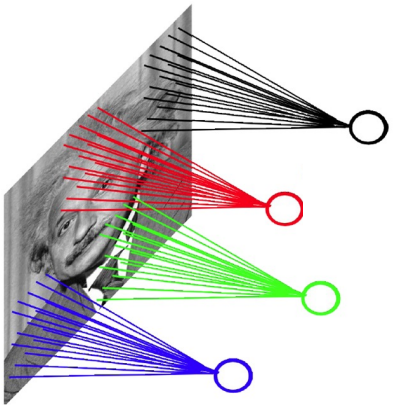
1. Computer Vision basics: Visual (local) feature detection and description, Bag of Word Image representation
2. Supervised learning: Introduction to Neural Networks (NNs)
3. Machine Learning basics: Risk, Classification, Datasets, benchmarks and evaluation, Linear classification (SVM)
4. Convolutional Nets for visual classification
5. Large deep convnets
6. Beyond ImageNet
7. Transfer learning and domain adaptation
8. **Attention and Vision Transformers**
9. Generative models with GANs
10. Generative models with conditional GANs
11. Control

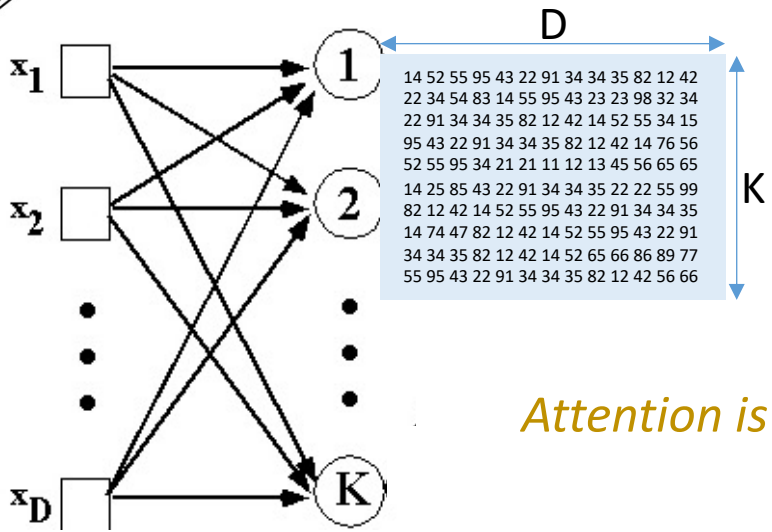
12/14 Bayesian deep learning

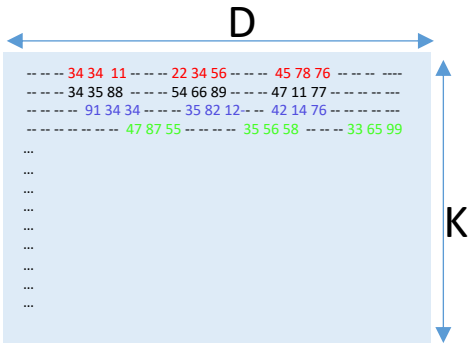
Attention process in ConvNets

In ConvNets what information is shared between pixels (or features) in one block? => *2D spatial locality (typically 3x3) => attention is done locally*







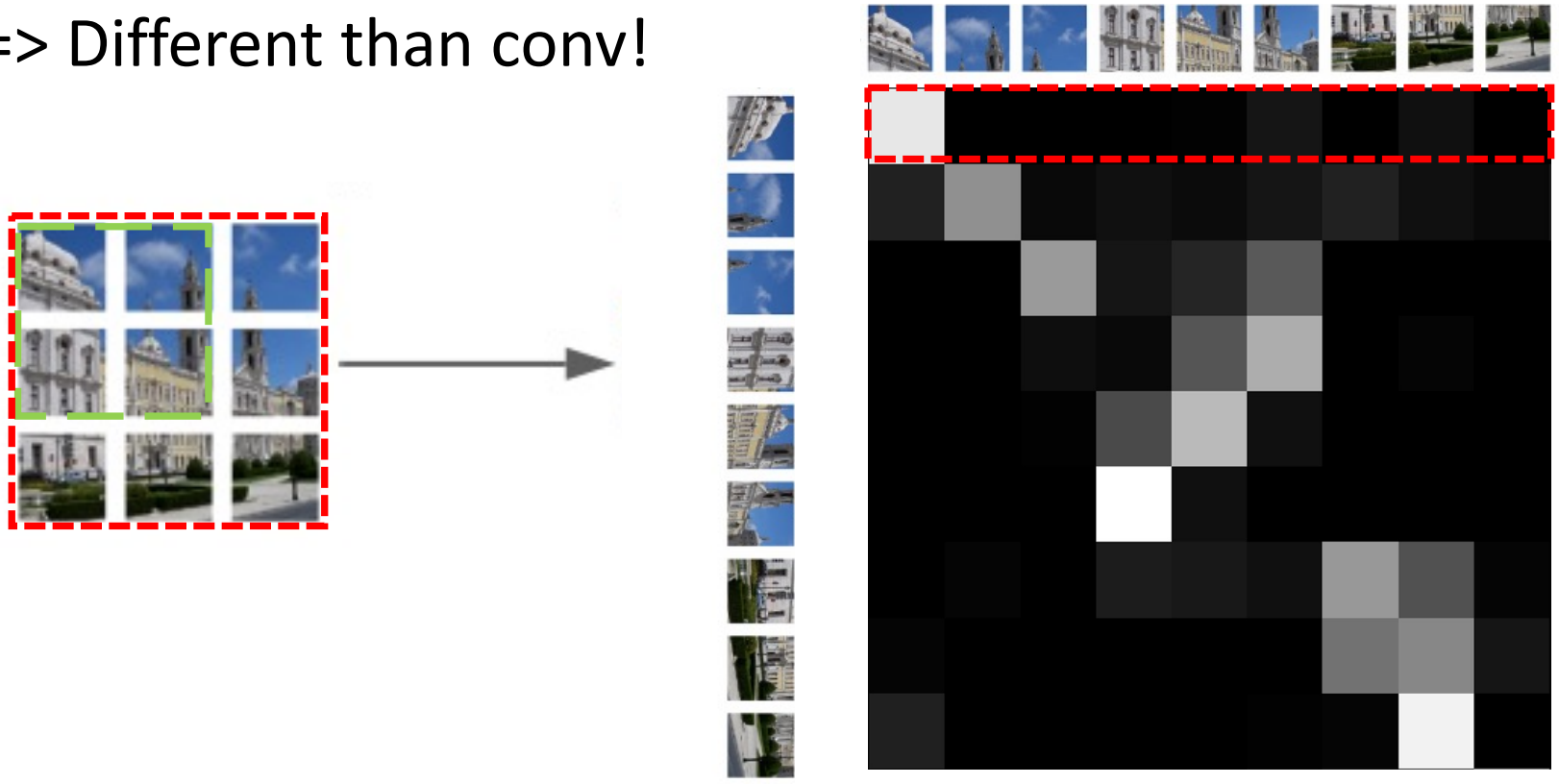


Attention is done locally=> less local after many layers

Global (Self) attention

How to build a deep architecture with ~~local~~ global attention inside? Meaning that one patch may interact with all others!

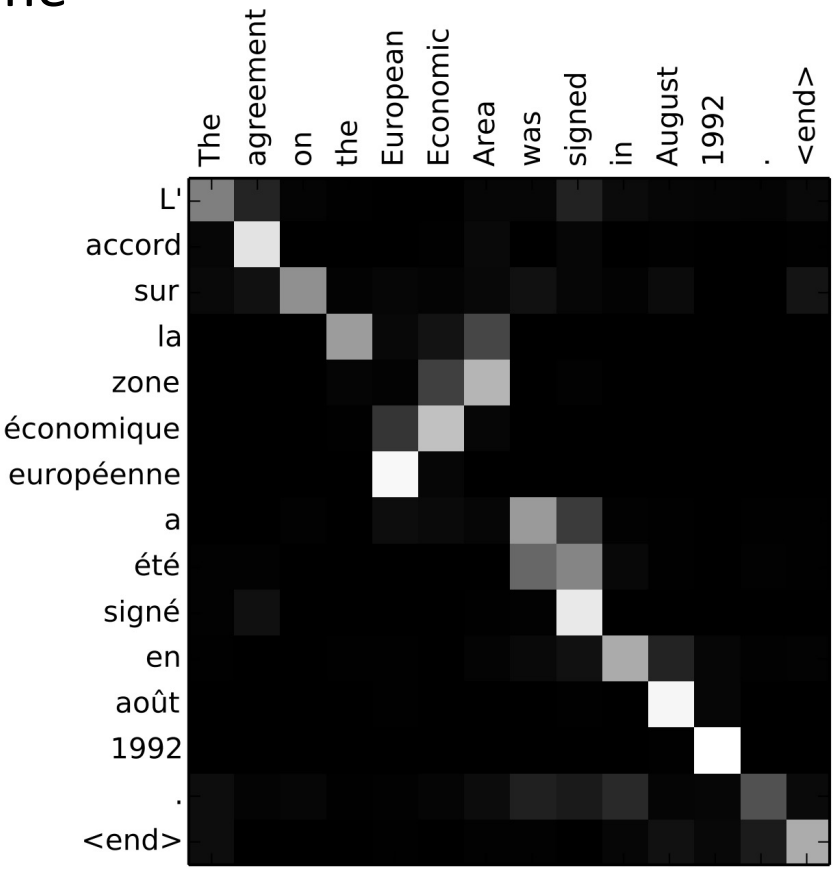
=> Different than conv!



Let’s see what they do in Natural Language Processing (NLP):

Attention between words in Machine translation process:

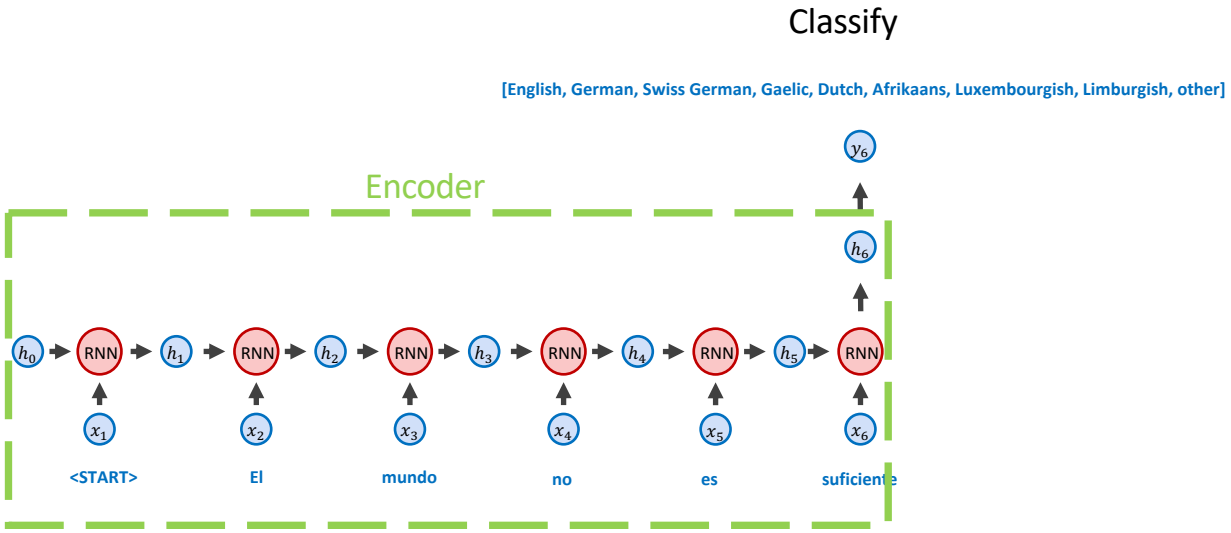
- Computing of weights
- Use them to compute feat.



Attention process in Natural Language Processing (NLP)

Attention between words

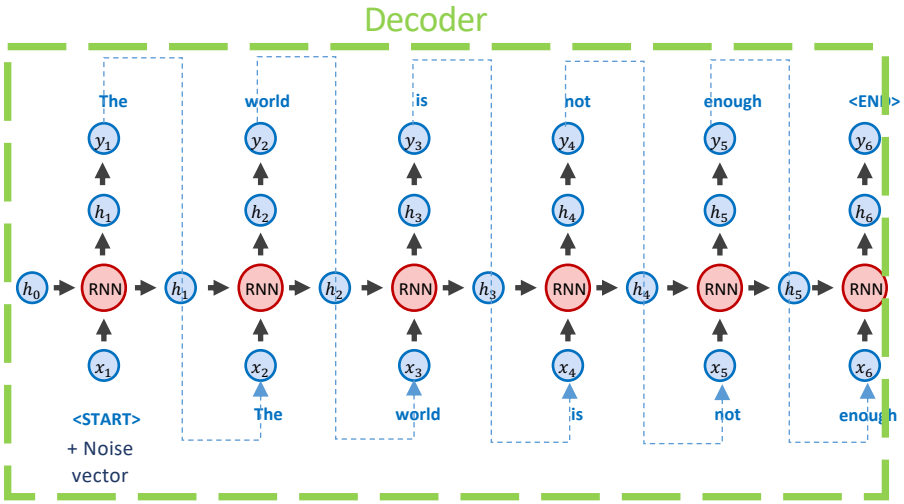
Basic models: Recurrent neural Nets (RNNs) as encoders => classification



Attention process in NLP

Attention between words

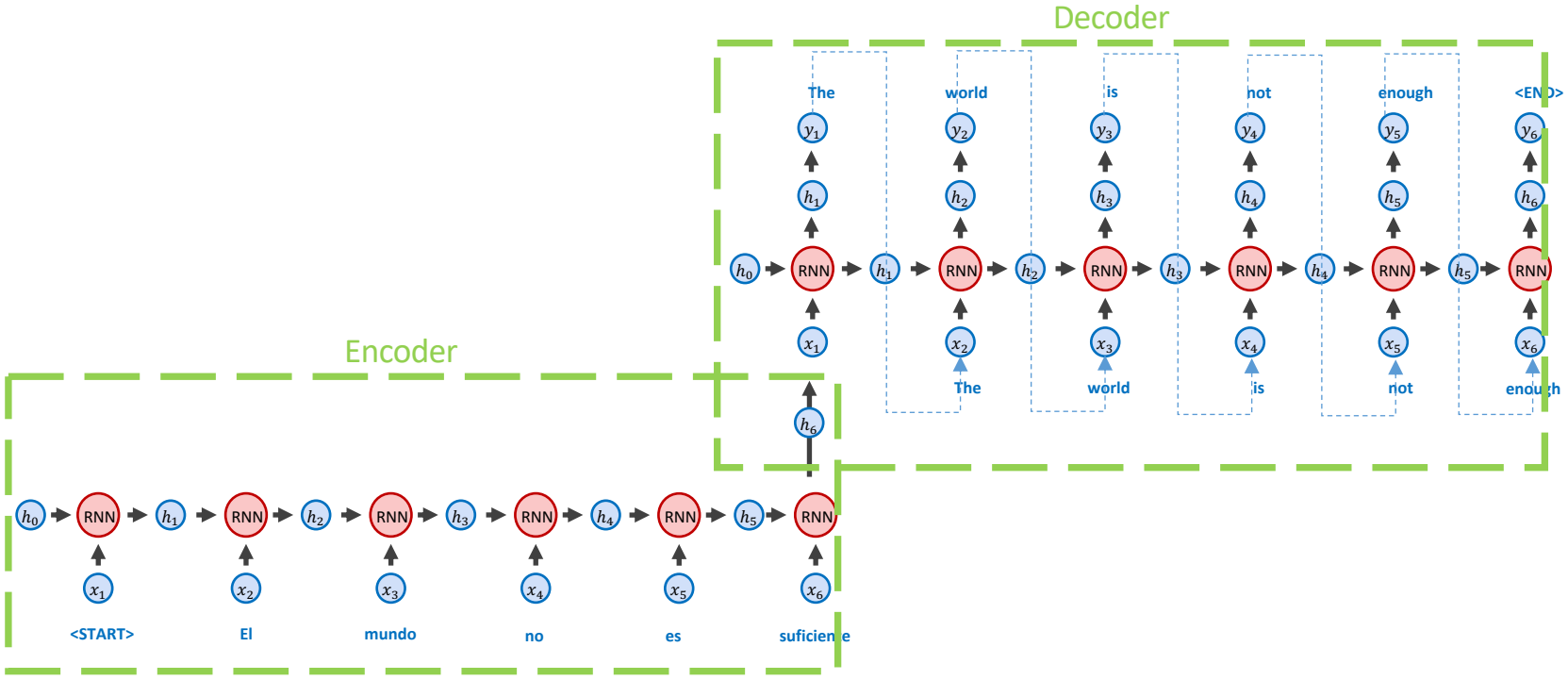
Basic models: Recurrent neural Nets (RNNs) as decoders => text generation



Attention process in NLP

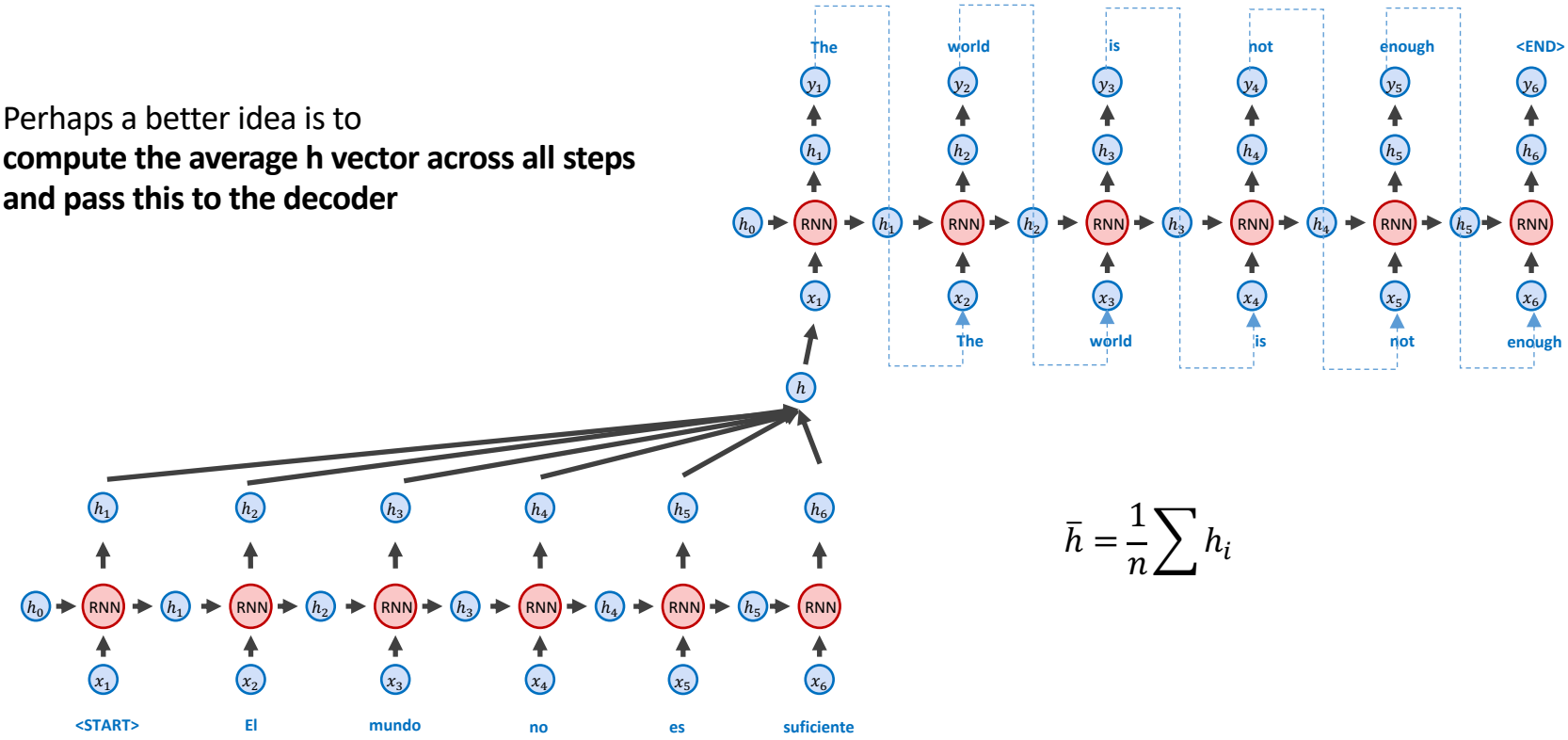
Attention between words/language

Basic models: Cross-attention for language translation: Seq2Seq -- RNNs2RNNs



Attention process in NLP

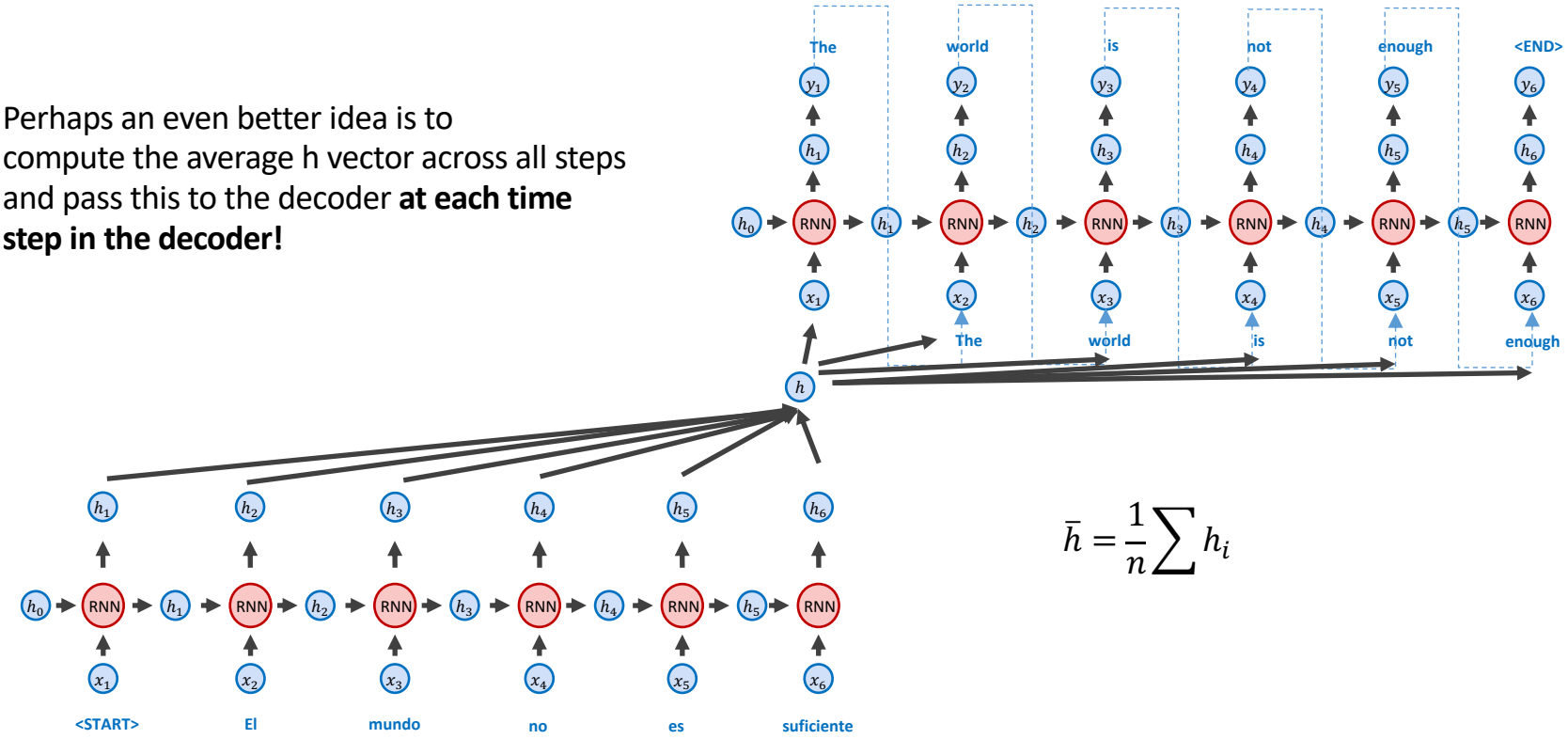
Perhaps a better idea is to
compute the average h vector across all steps and pass this to the decoder



$$\bar{h} = \frac{1}{n}\sum h_i$$

Attention process in NLP

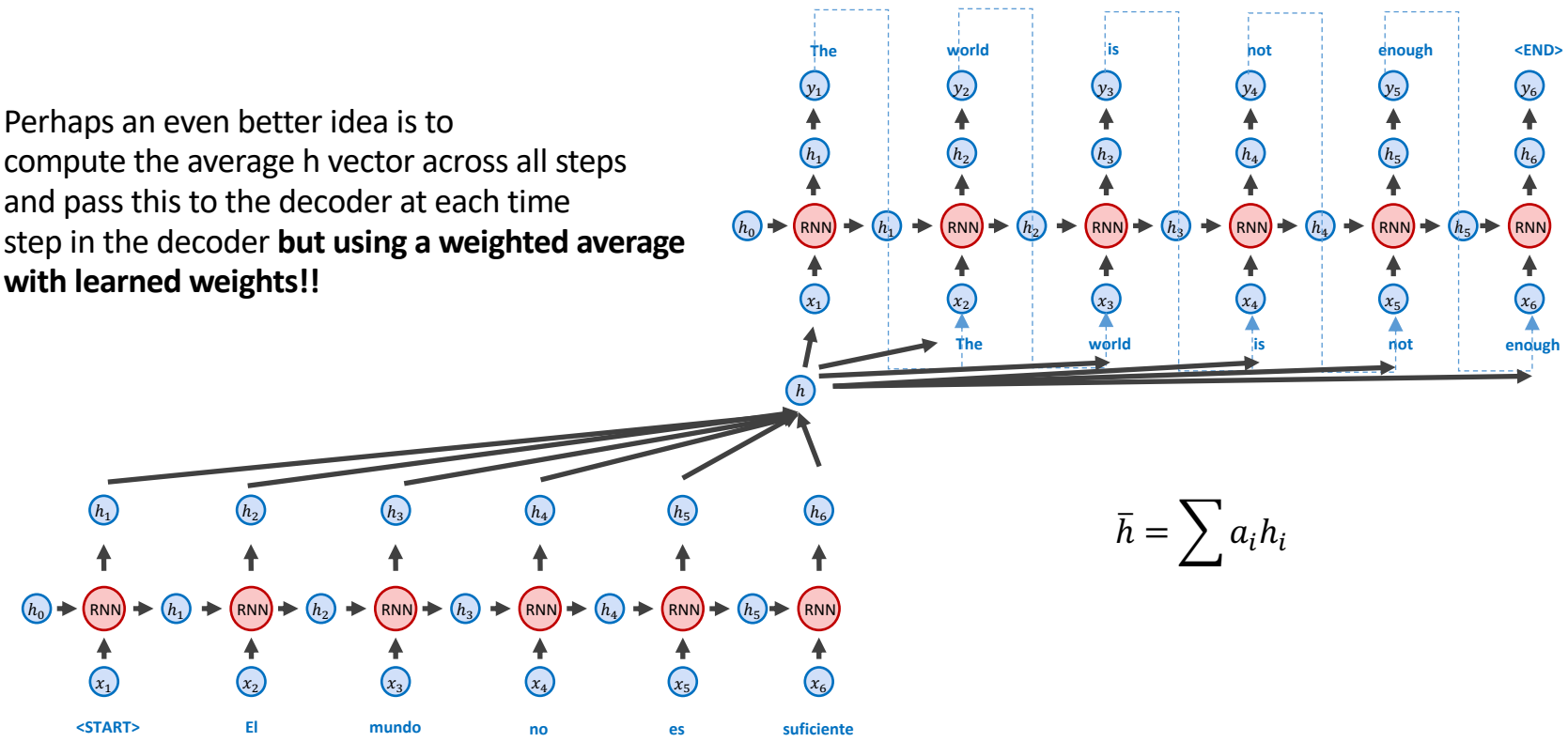
Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder **at each time step in the decoder!**



$$\bar{h} = \frac{1}{n}\sum h_i$$

Attention process in NLP

Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder at each time step in the decoder **but using a weighted average with learned weights!!**



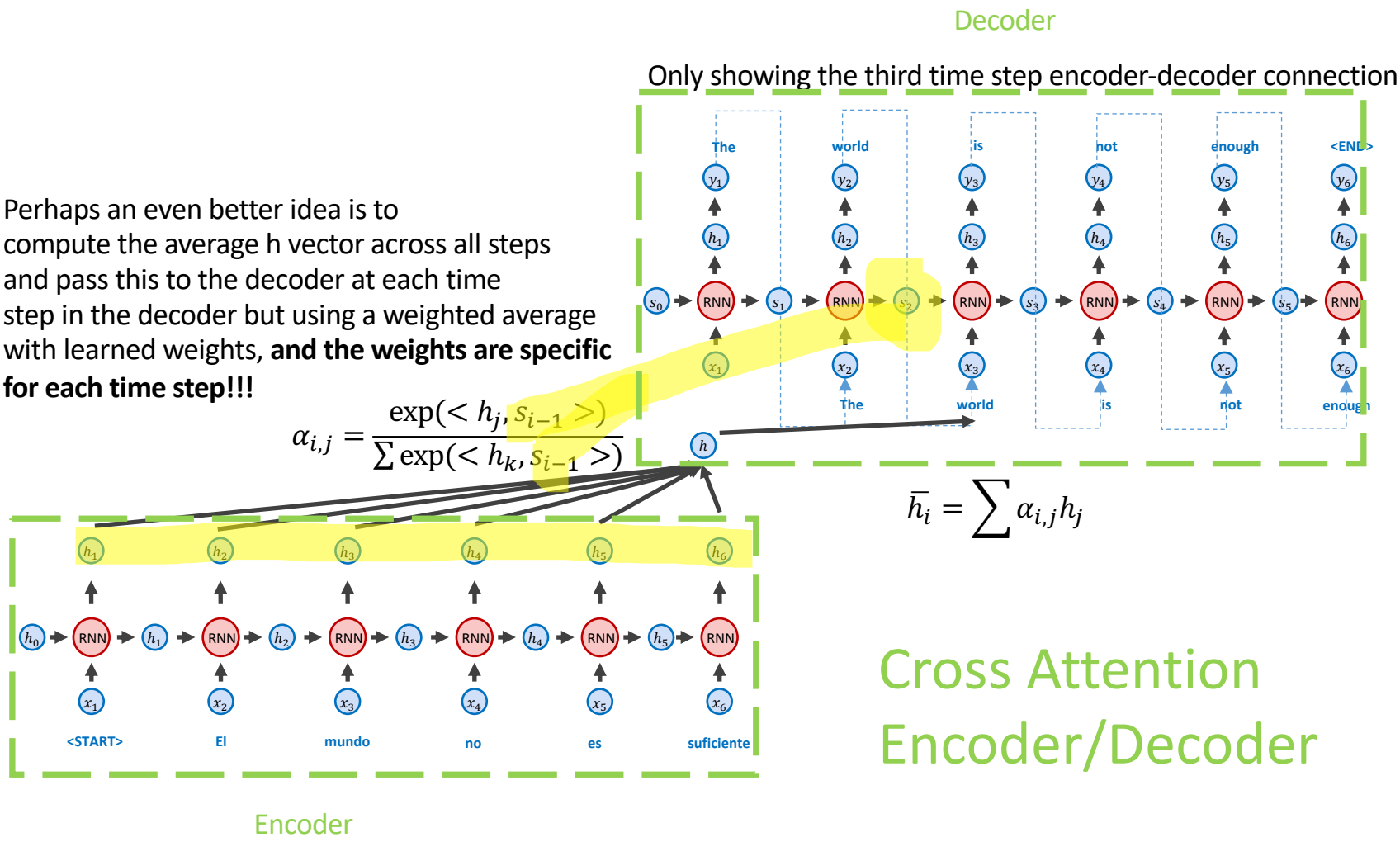
$$\bar{h} = \sum a_i h_i$$

Attention process in NLP

Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder at each time step in the decoder but using a weighted average with learned weights, **and the weights are specific for each time step!!!**

$$\alpha_{i,j} = \frac{\exp(< h_j, s_{i-1} >)}{\sum \exp(< h_k, s_{i-1} >)}$$

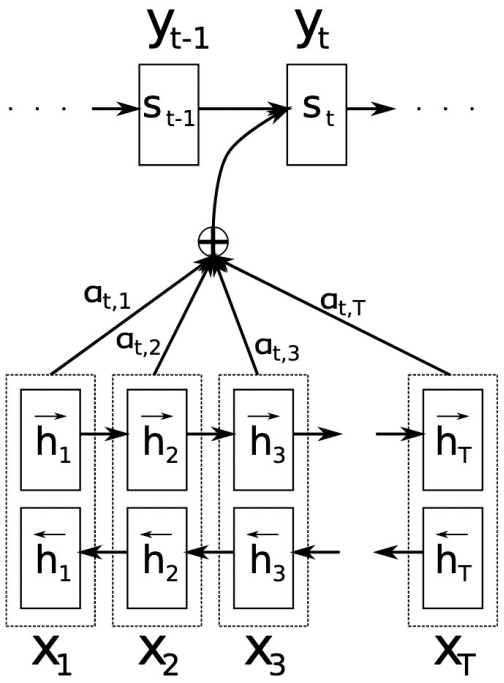
$$\bar{h}_i = \sum \alpha_{i,j} h_j$$



Cross Attention Encoder/Decoder

Attention process in NLP

[Neural Machine Translation By Jointly Learning To Align And Translate, Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio, ICLR 2015]



$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

Cross Attention for decoder:

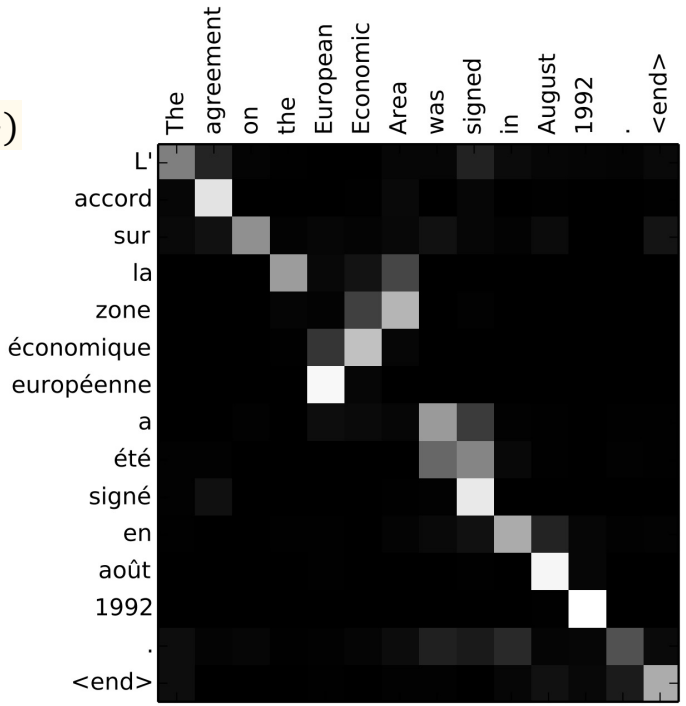
(before $\bar{h}_i = \sum \alpha_{i,j} h_j$)

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

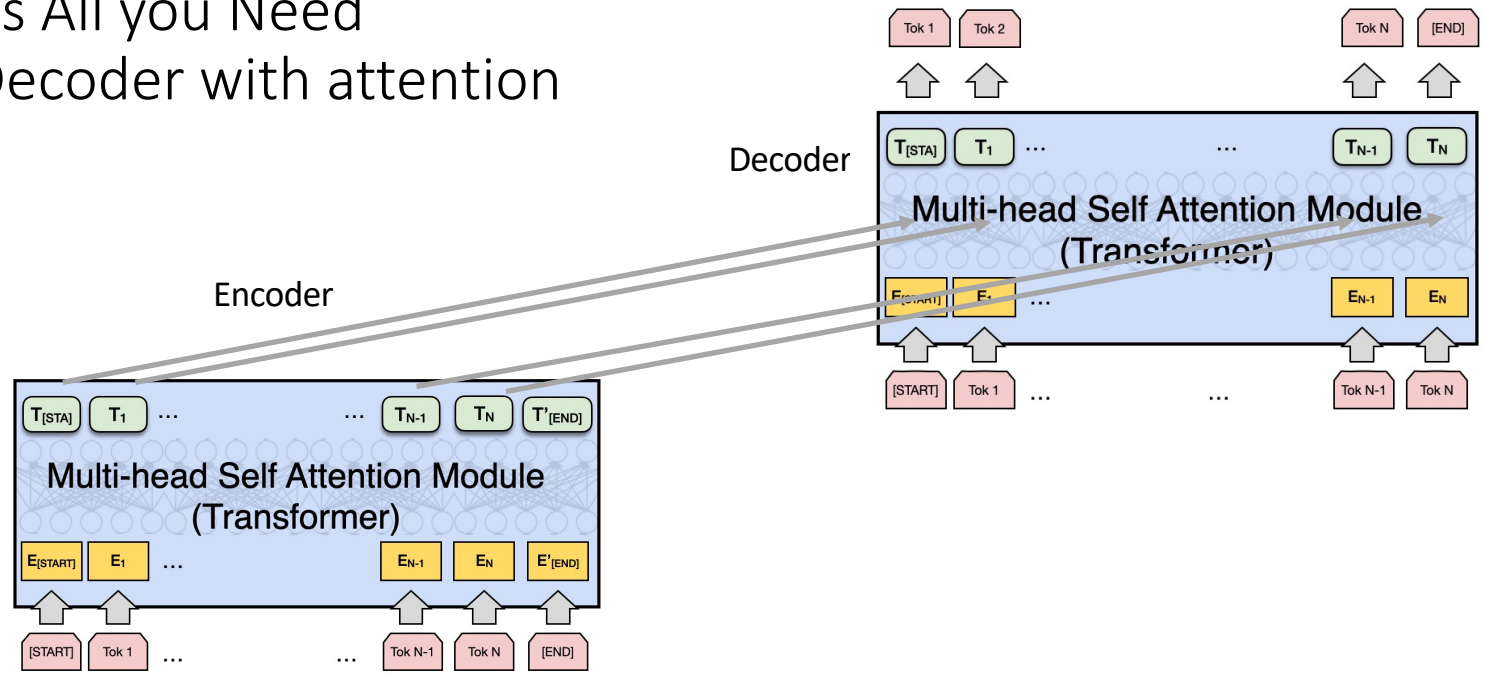
Weights



a() is an MLP instead of a dot pdt $< h_j, s_{i-1} >$

Attention process in NLP

Transformer architecture:
no RNNs
Attention is All you Need
Encoder/Decoder with attention

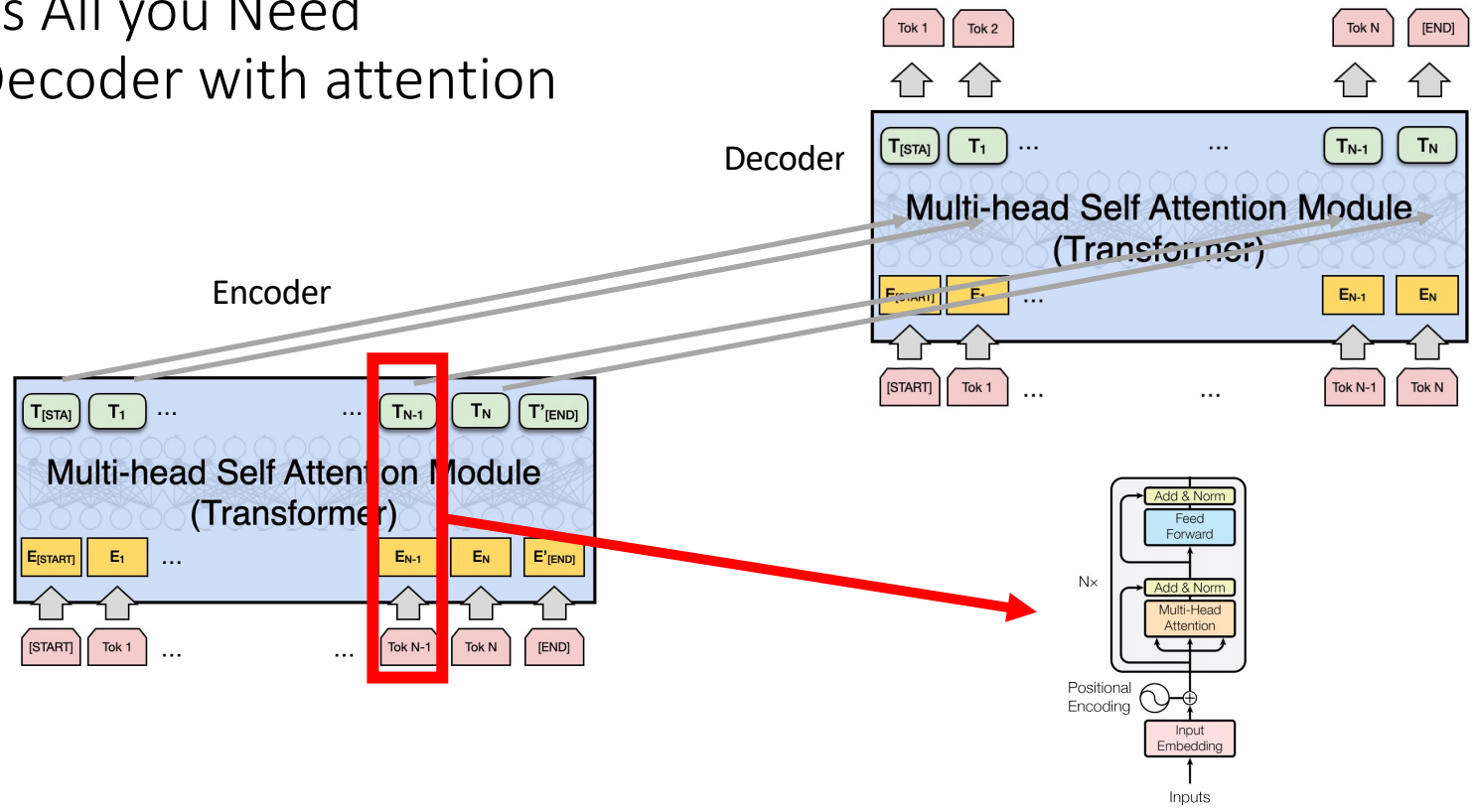


Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]

[Vaswani et al. Attention is all you need] https://arxiv.org/abs/1706.03762

Attention process in NLP

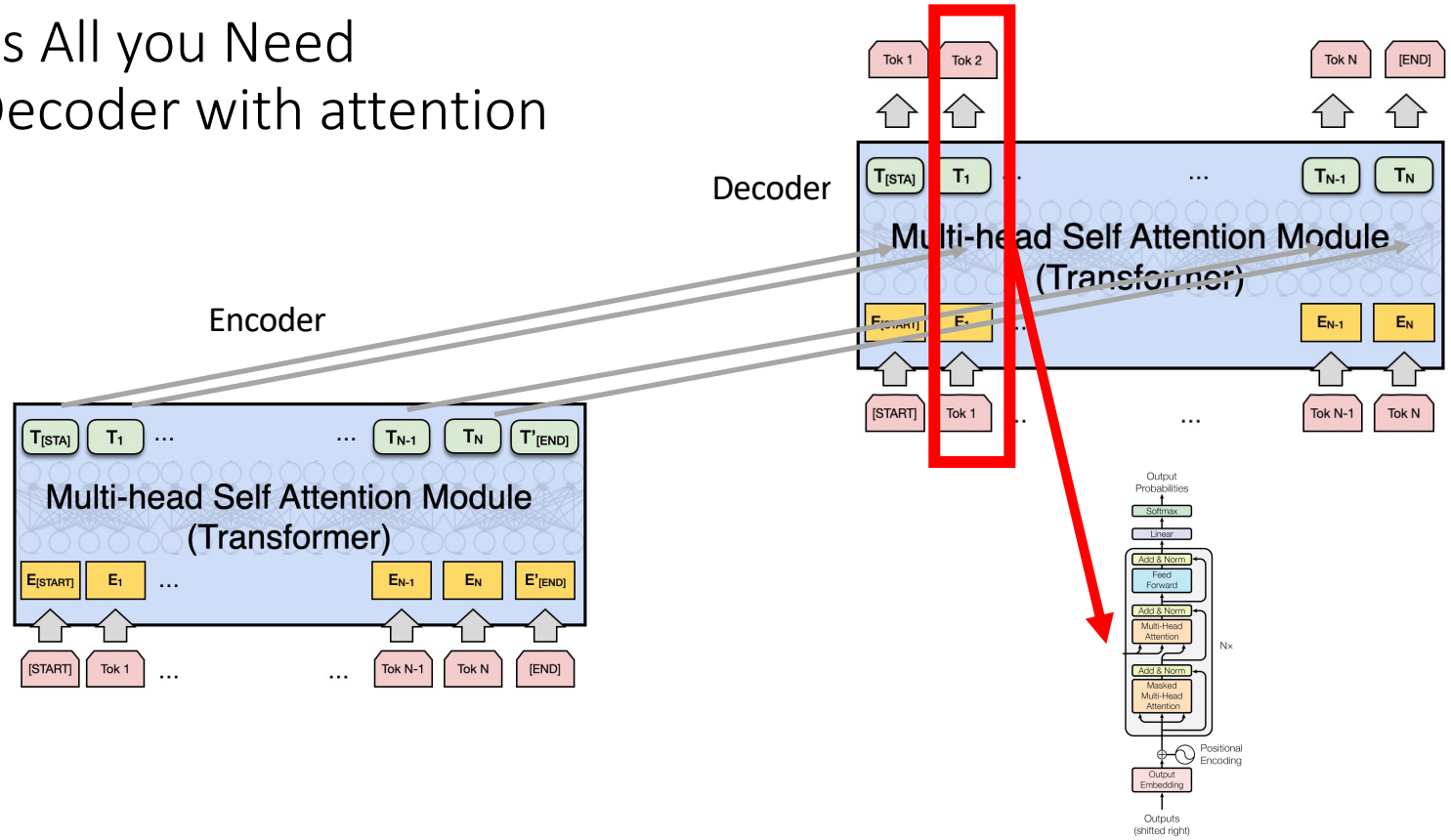
Transformer architecture:
no RNNs
Attention is All you Need
Encoder/Decoder with attention



Self Attention Module

Attention process in NLP

Transformer architecture:
no RNNs
Attention is All you Need
Encoder/Decoder with attention

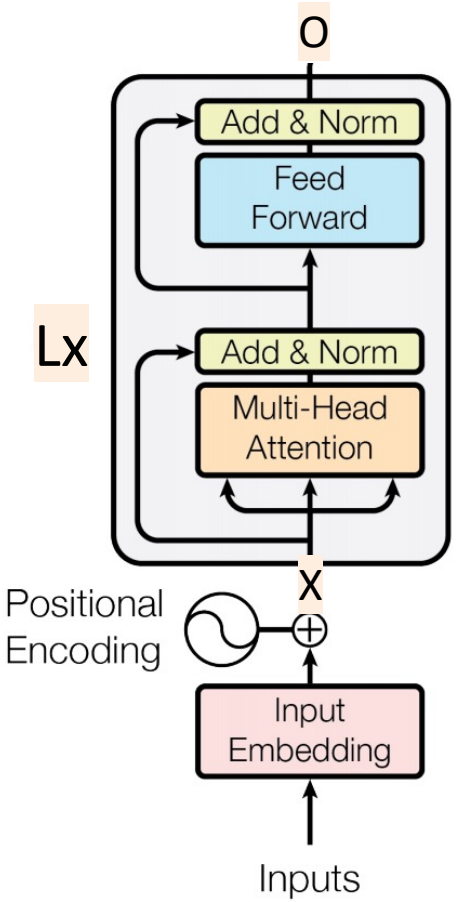


Cross Attention Module

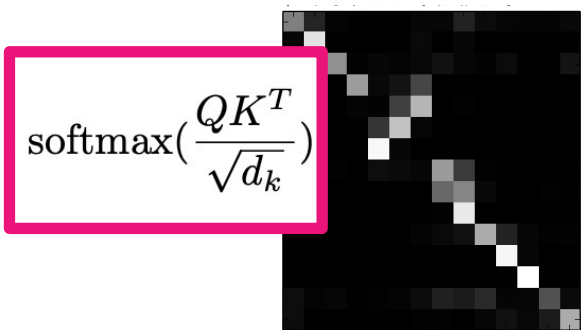
Attention process in NLP

Transformer architecture:
Encoder with attention
(Same principle of self/cross attention)
Detail of the **Self Attention Module (SA):**

X, Q, V, K,O, projections …
Details in course …



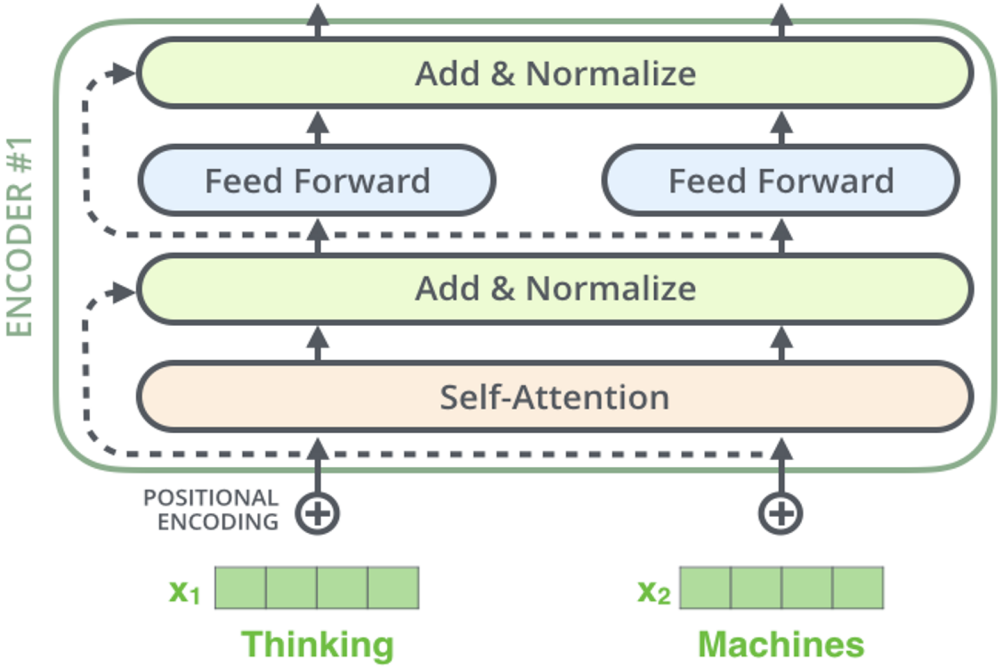
$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Attention process in NLP

Transformer architecture:

Encoder: Self-Attention all together MLP alone



Attention process in NLP

Multi-head Attention: Do not settle for just one set of attention weights.

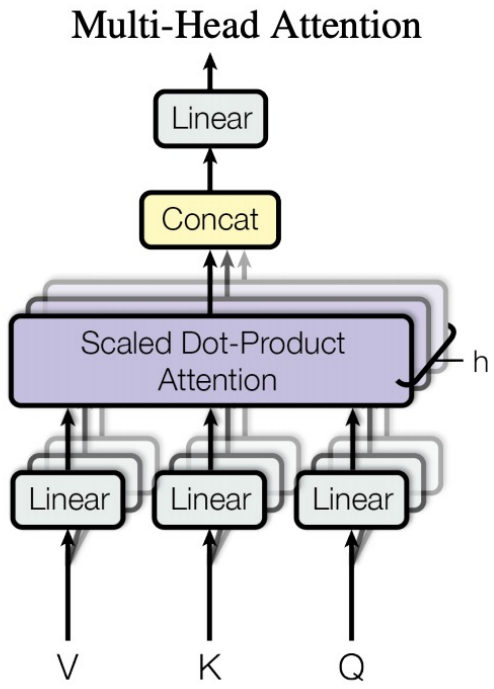
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

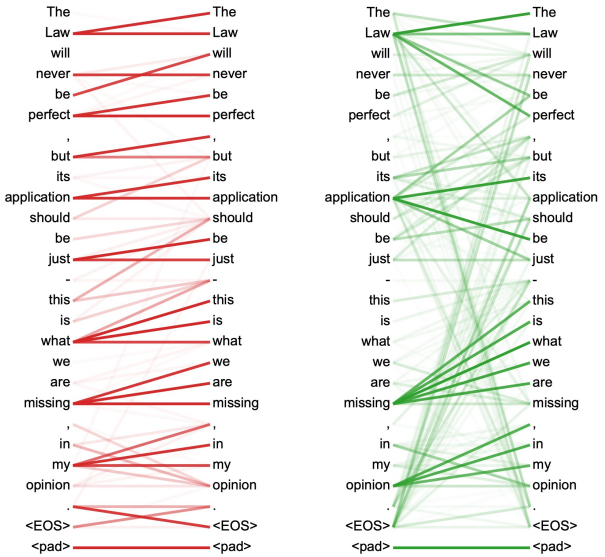
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Where the projections are parameter matrices

$$W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}, W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$$

$$W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$$





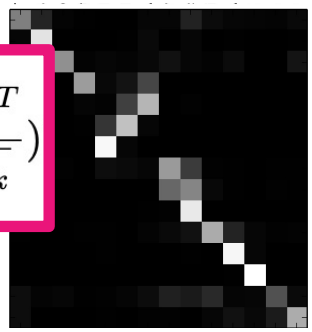
=> MSA: Multi-head Self Attention Module

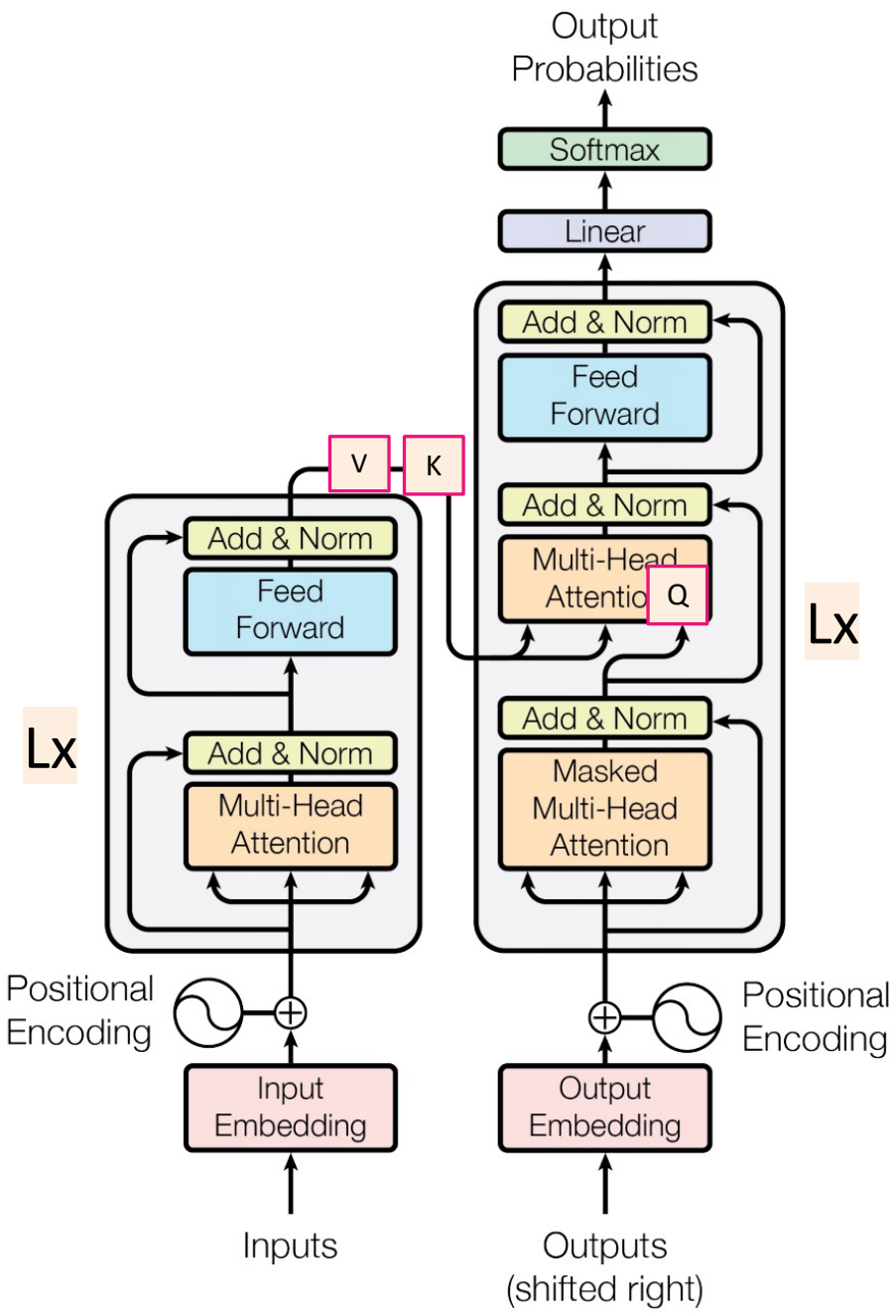
Attention process in NLP

Transformer architecture:
Decoder with attention
Same principle than self attention
Except that **K, V come from the Encoder and Q from the decoder**

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

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



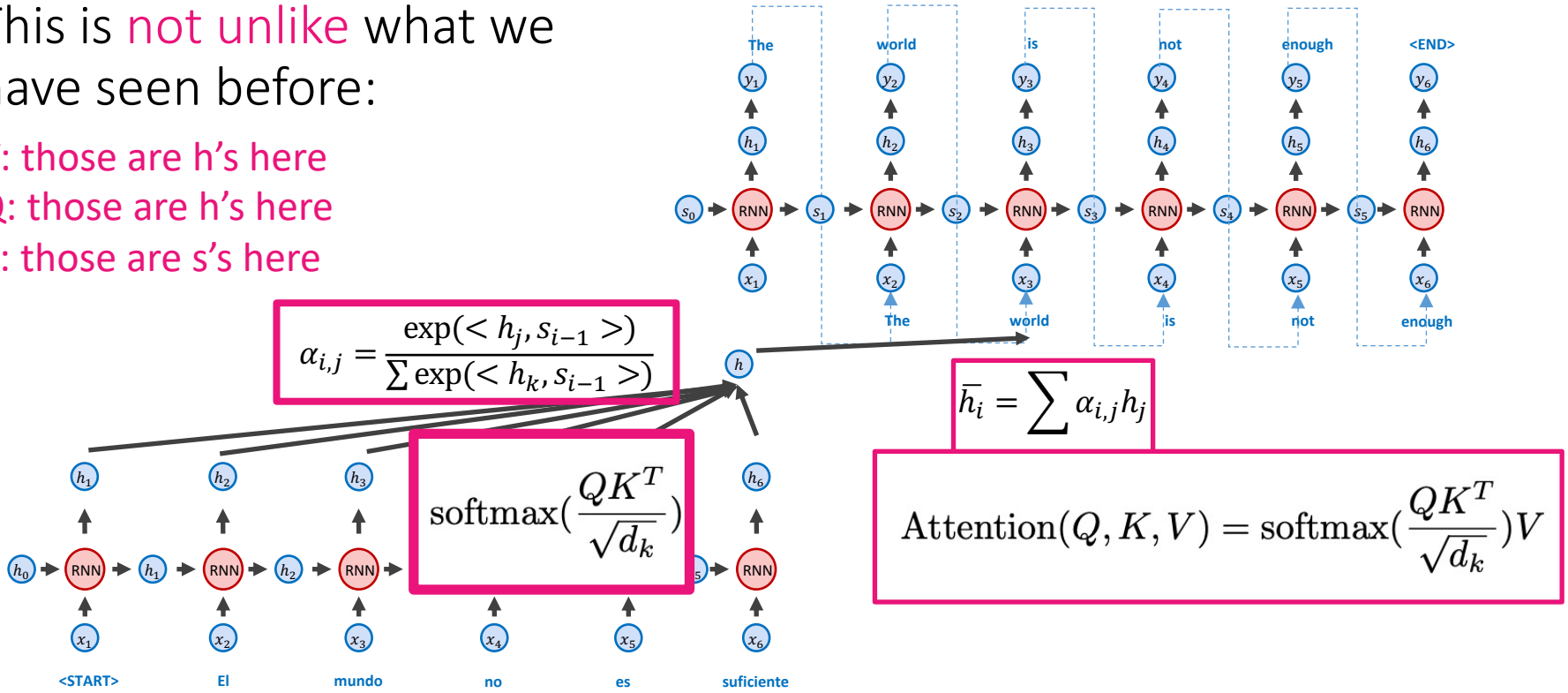


Attention process in NLP

Transformer architecture

This is not unlike what we have seen before:

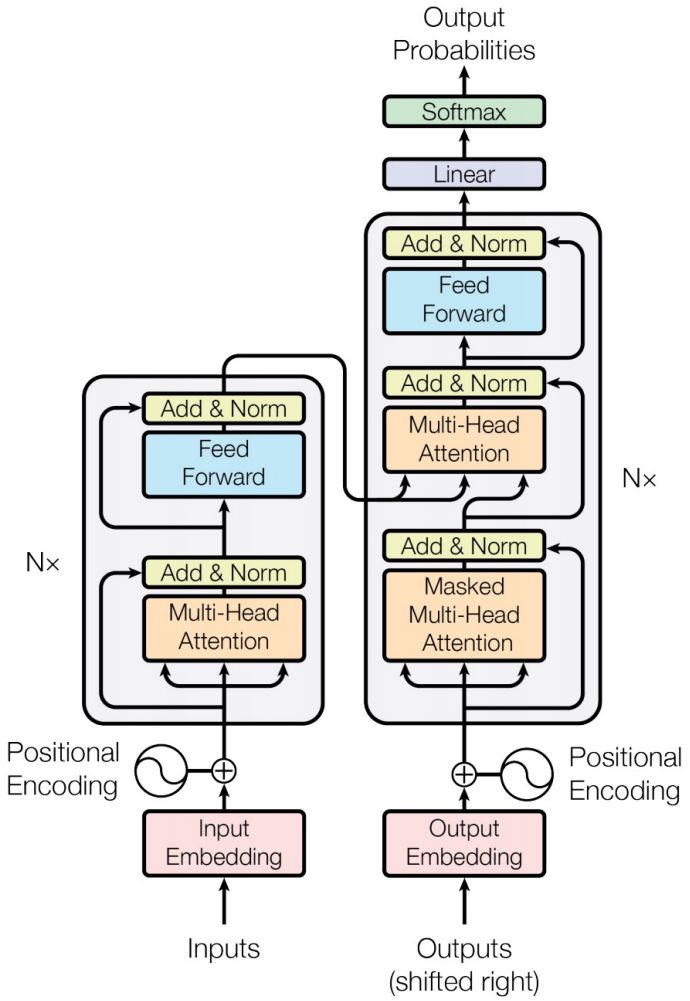
V: those are h's here
Q: those are h's here
K: those are s's here



Attention process in NLP

Position encoding:
We can lose track of position since we are aggregating across all locations
Adding a position code for each patch

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



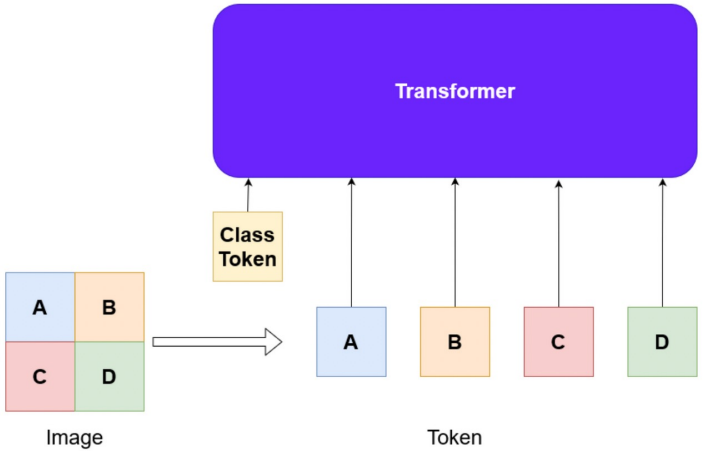
Attention process in Vision

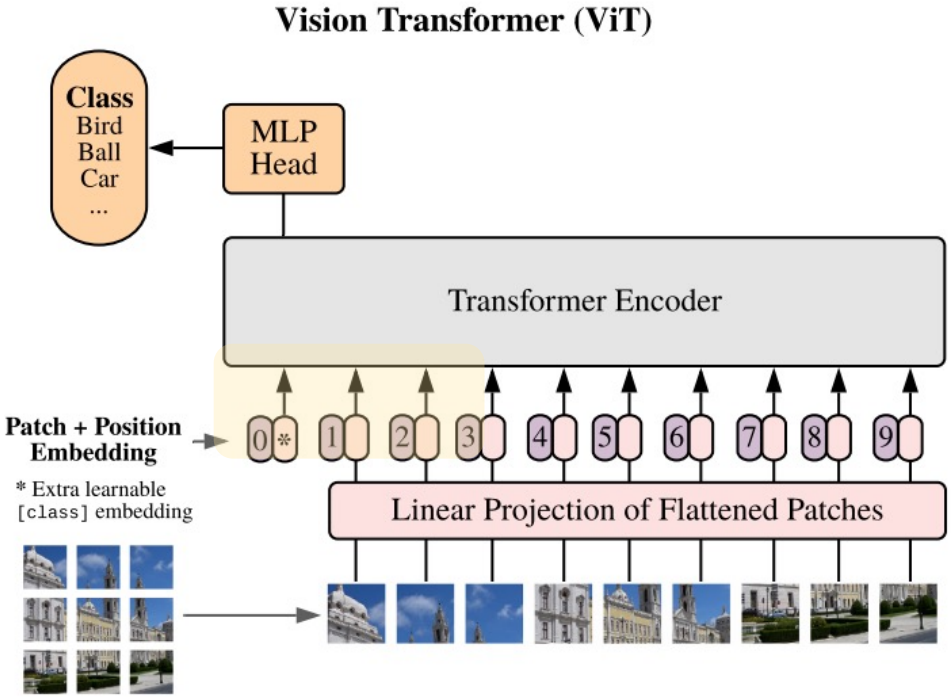
(credit slide to Jeamin Jong)

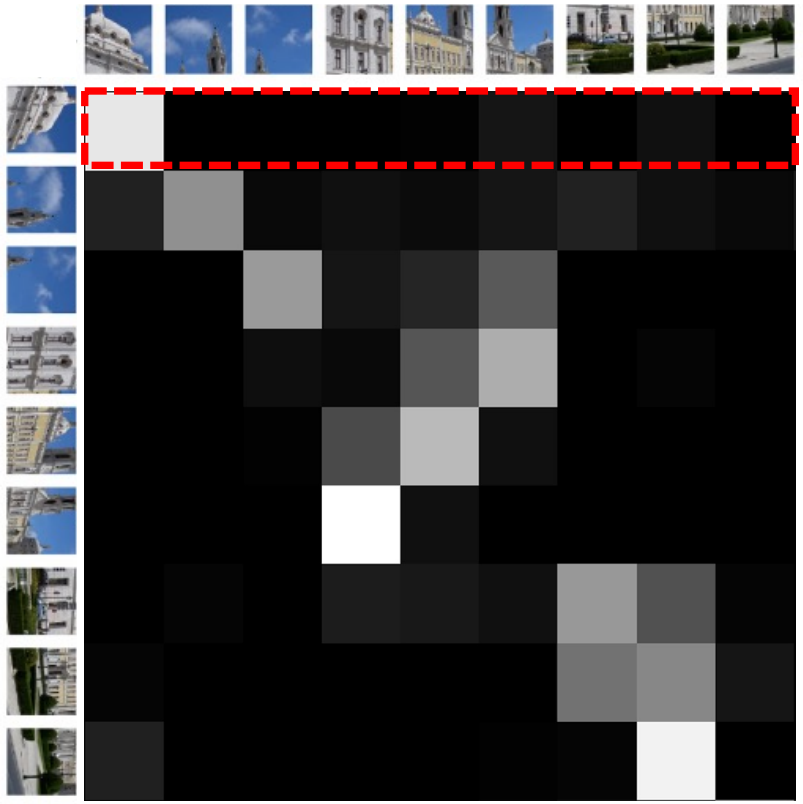
Is it possible to mimic this attention-based architecture for vision processing?

Yes! **ViT** (Vision image Transformers) architecture

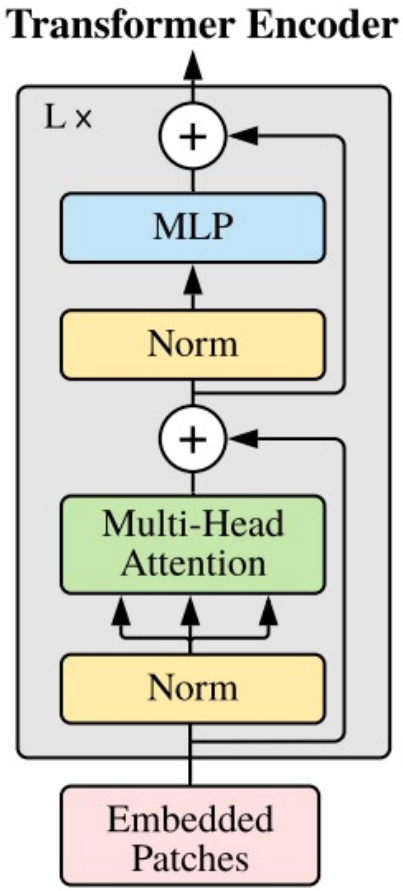
=> Self attention encoder modules

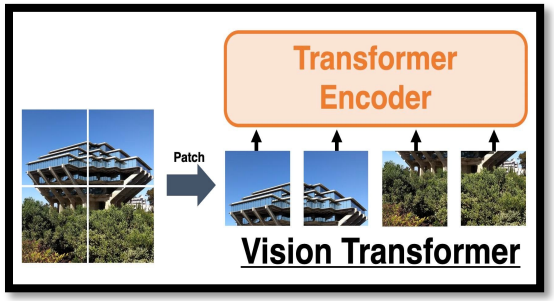


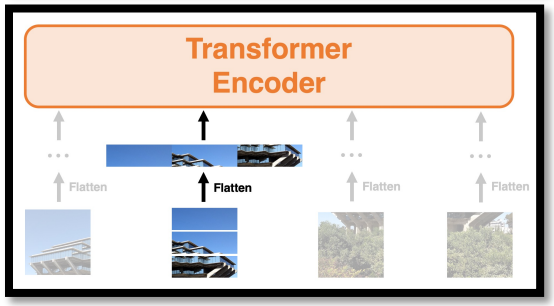




ViT - Transformer Encoder







$$x \in \mathbb{R}^{H \times W \times C}$$

$$x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$$

$$N = HW/P^2$$

CLS token

$$\mathbf{z}_0 = [\mathbf{x}_{\text{class}};\ \mathbf{x}_p^1\mathbf{E};\ \mathbf{x}_p^2\mathbf{E};\ \cdots\ ;\ \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}, \qquad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D},\ \mathbf{E}_{pos} \in \mathbb{R}^{(N+1) \times D}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \qquad \ell = 1 \ldots L$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \qquad \ell = 1 \ldots L$$

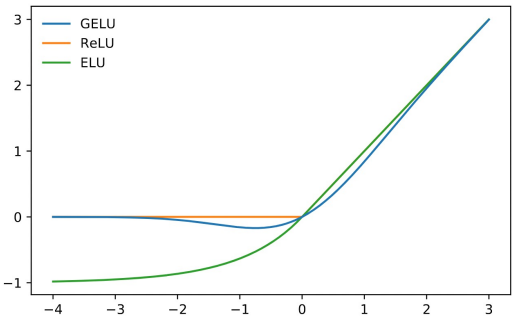
$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0)$$

[class=CLS] token: a learnable embedding to the sequence of embedded patches

Layernorm (LN) before every block, and residual connections after every block

MLP: two layers with a GELU non-linearity

Hybrid Architecture : Raw image patches --> Feature map of a CNN

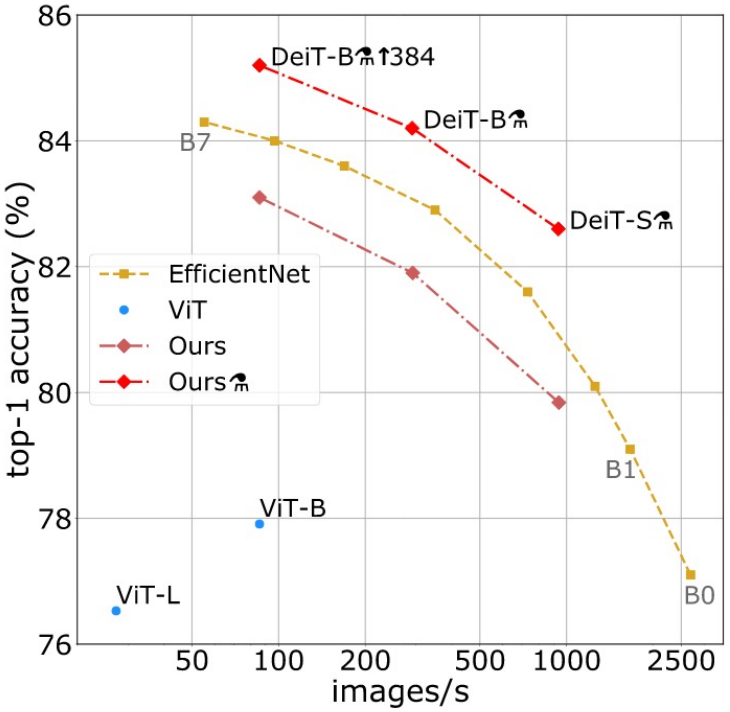


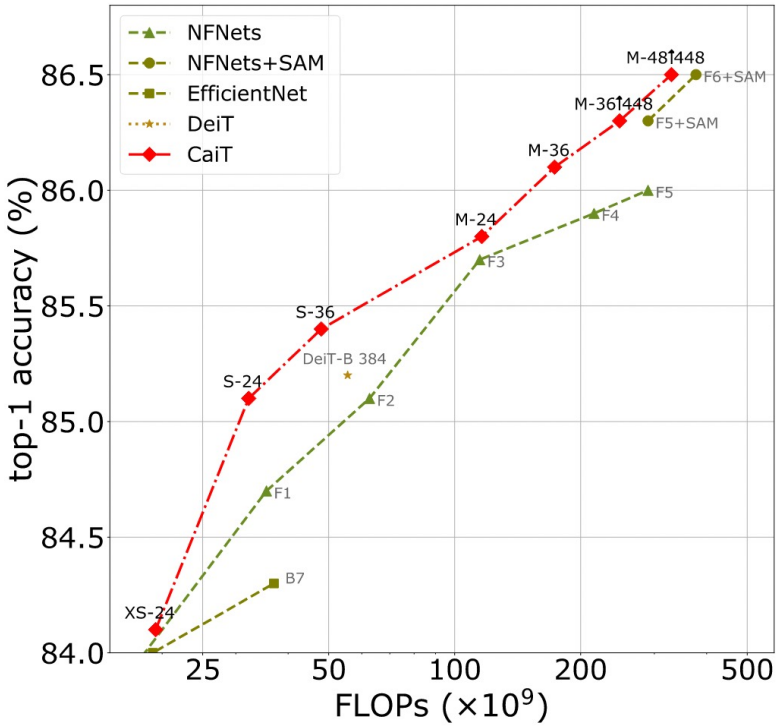
ViT - Transformer Encoder

Expe with ViT like transformers

State-of-the-art performance on ImageNet1k!

Many tricks to be discussed further





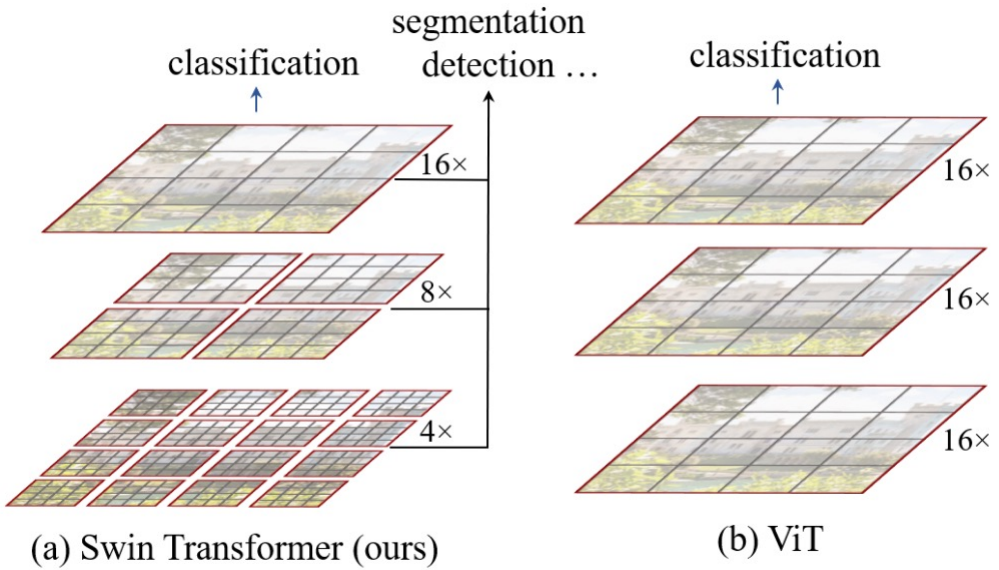
ViT - Transformer Encoder

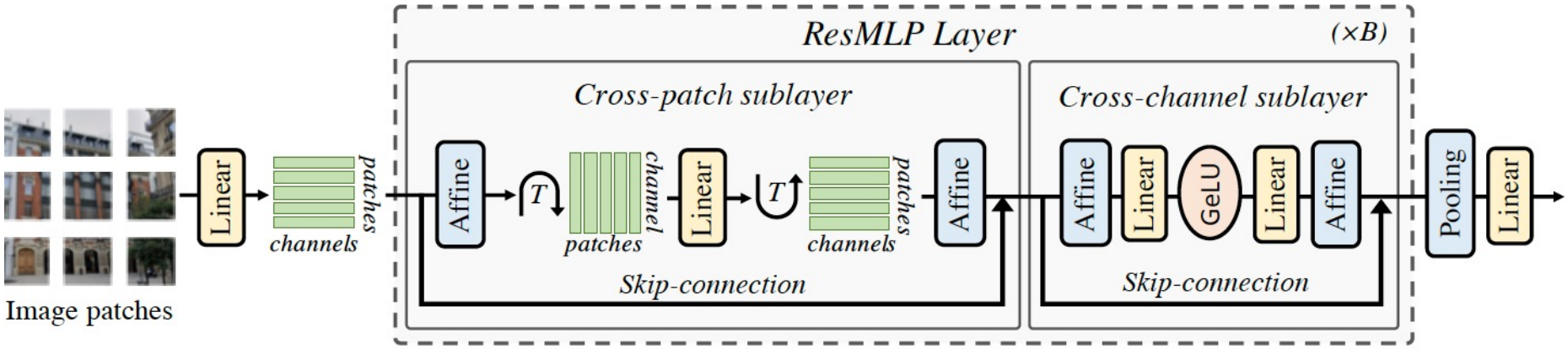
Many kinds of hybrid archi after conv/transformers introduction

Ex:

Swin Transformers

ResMLP: Feedforward networks for image classification with data-efficient training





Beyond transformers …

Perceiver IOA General Architecture for Structured Inputs & Outputs