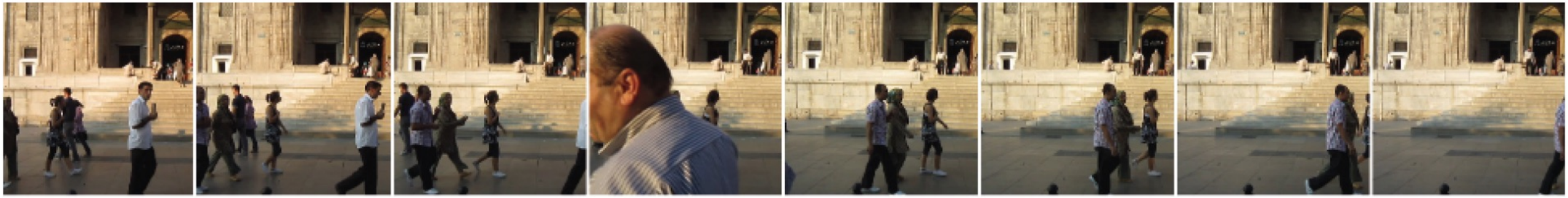
Computer Vision

Sequences and Attention Models

Sequences



time

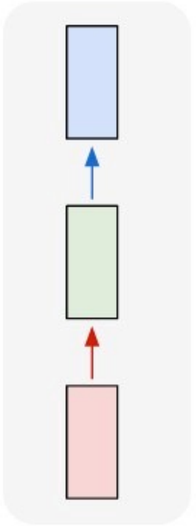
“An”, “evening”, “stroll”, “through”, “a”, “city”, “square”

time

time

Sequence Modeling

one to one

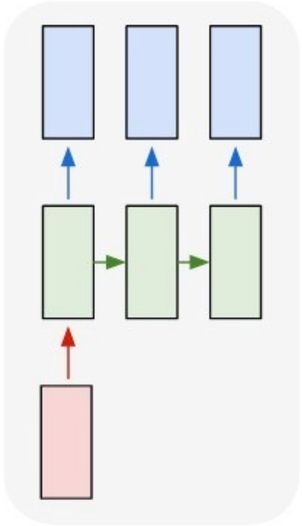


Input: No sequence

Output: No sequence

Examples: “standard” classification/ regression problems

one to many

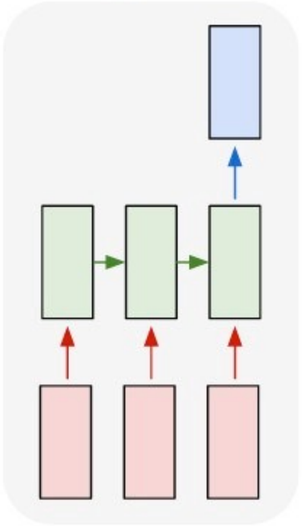


Input: No sequence

Output: Sequence

Examples: Image captioning

many to one

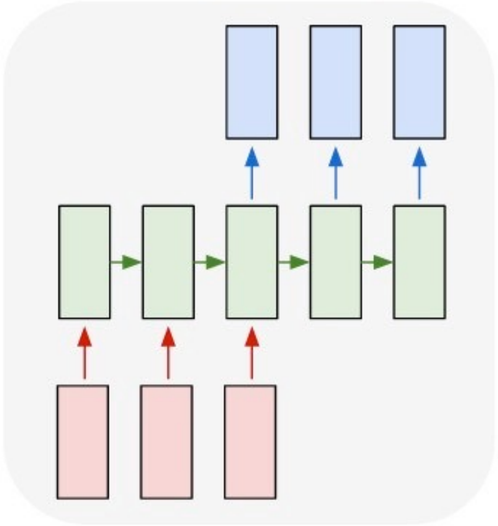


Input: Sequence

Output: No sequence

Examples: sentence classification, multiple-choice question answering

many to many

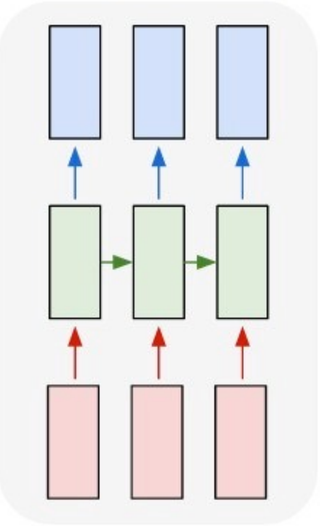


Input: Sequence

Output: Sequence

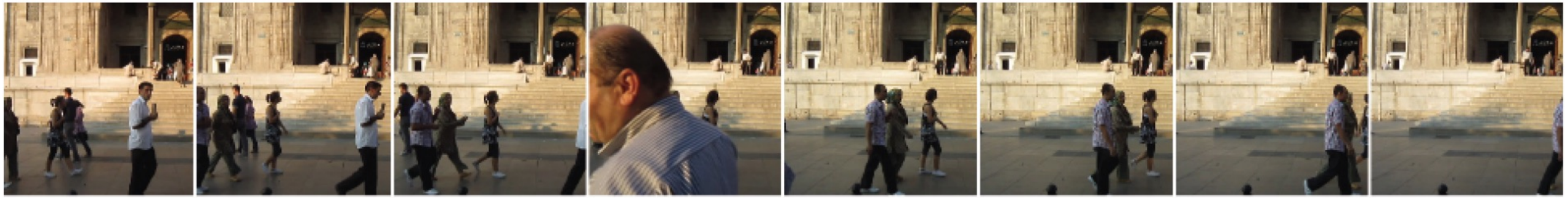
Examples: machine translation, video captioning, open-ended question answering, video question answering

many to many

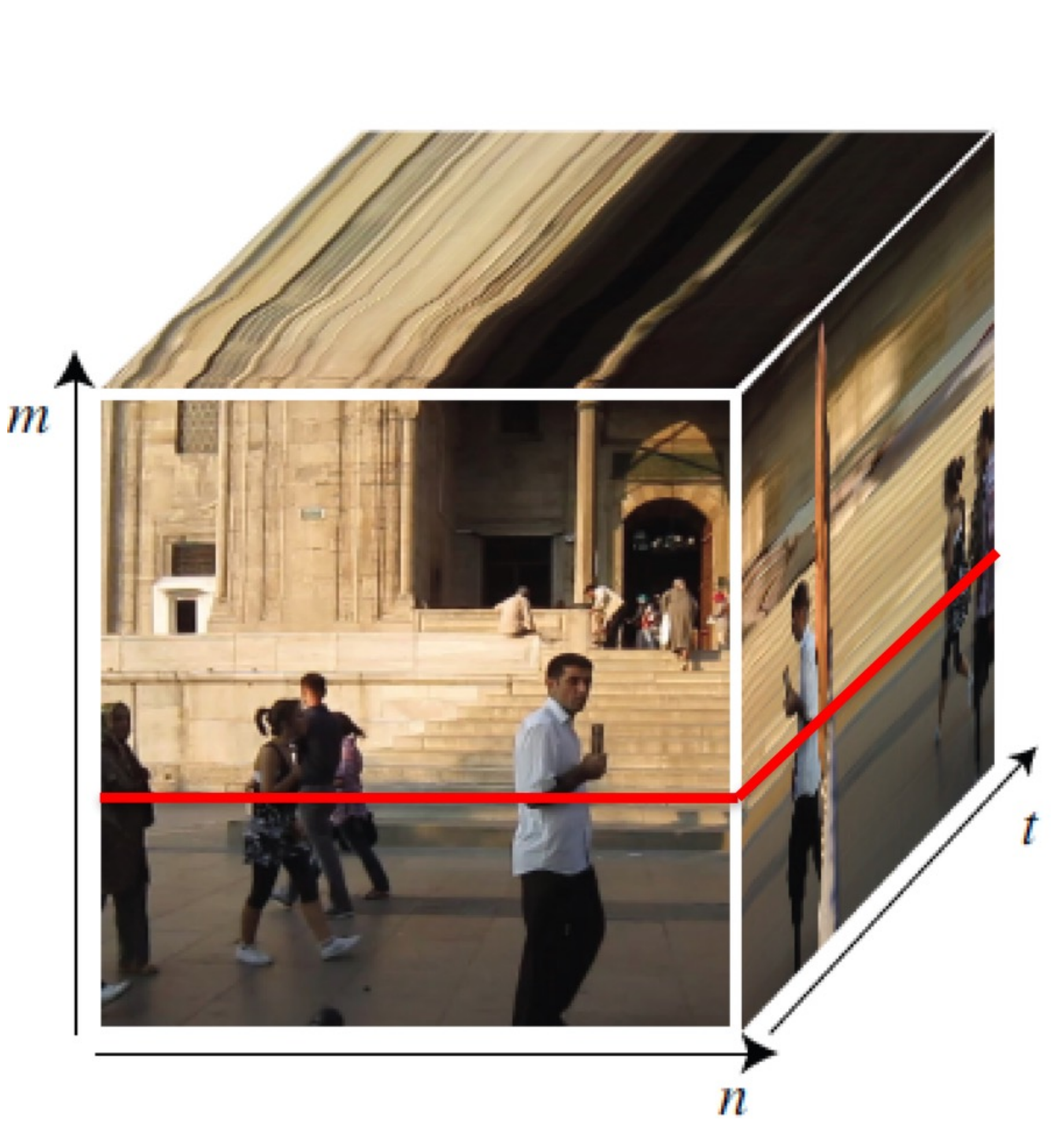


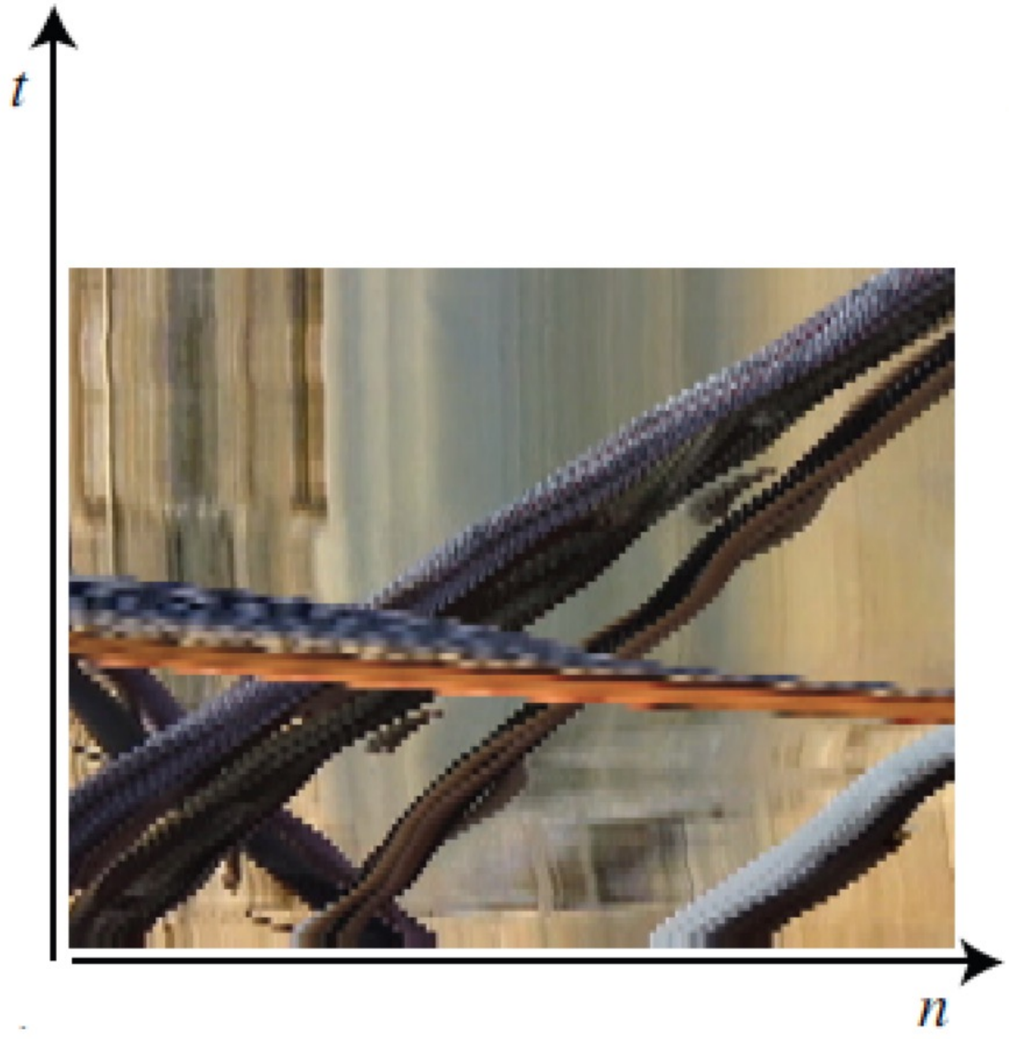
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

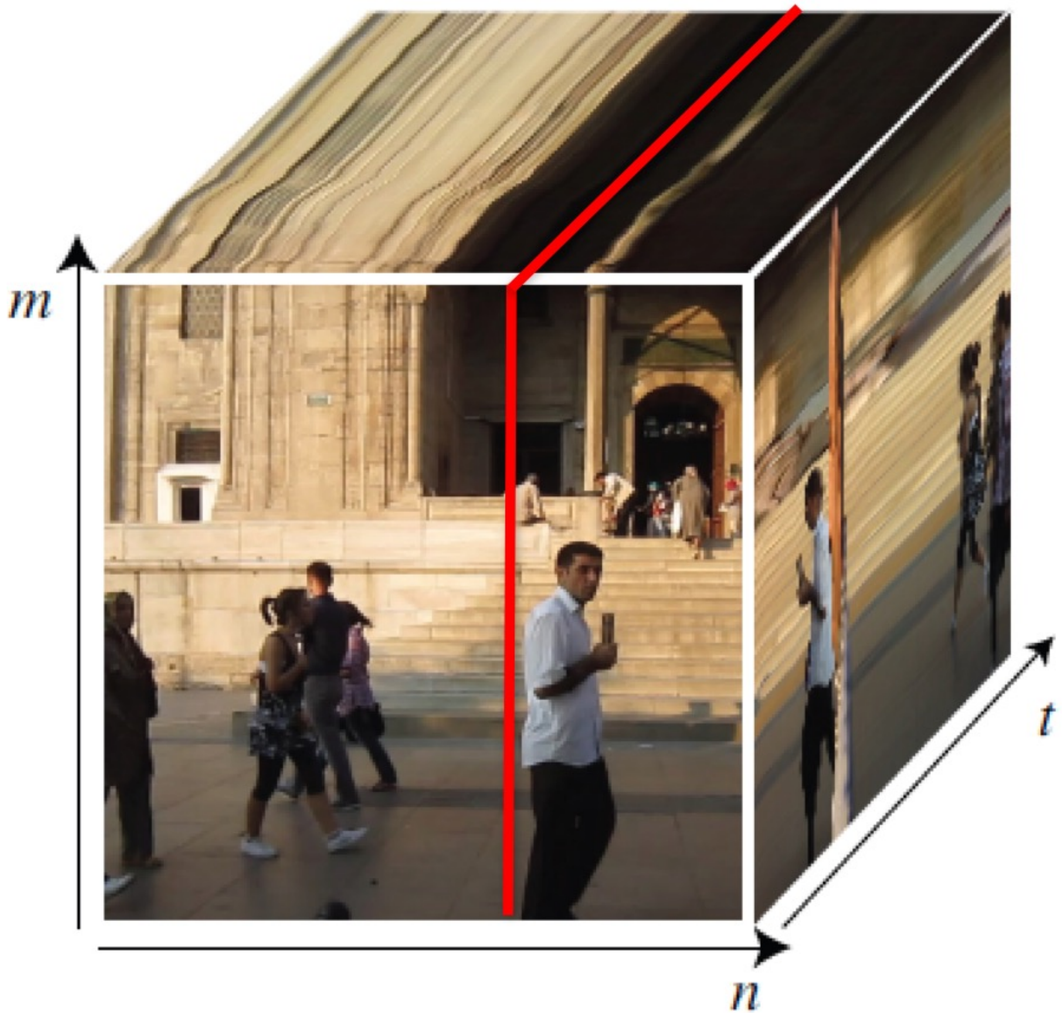
Credit: Dhruv Batra, Andrej Karpathy



Bill Freeman and Phillip Isola

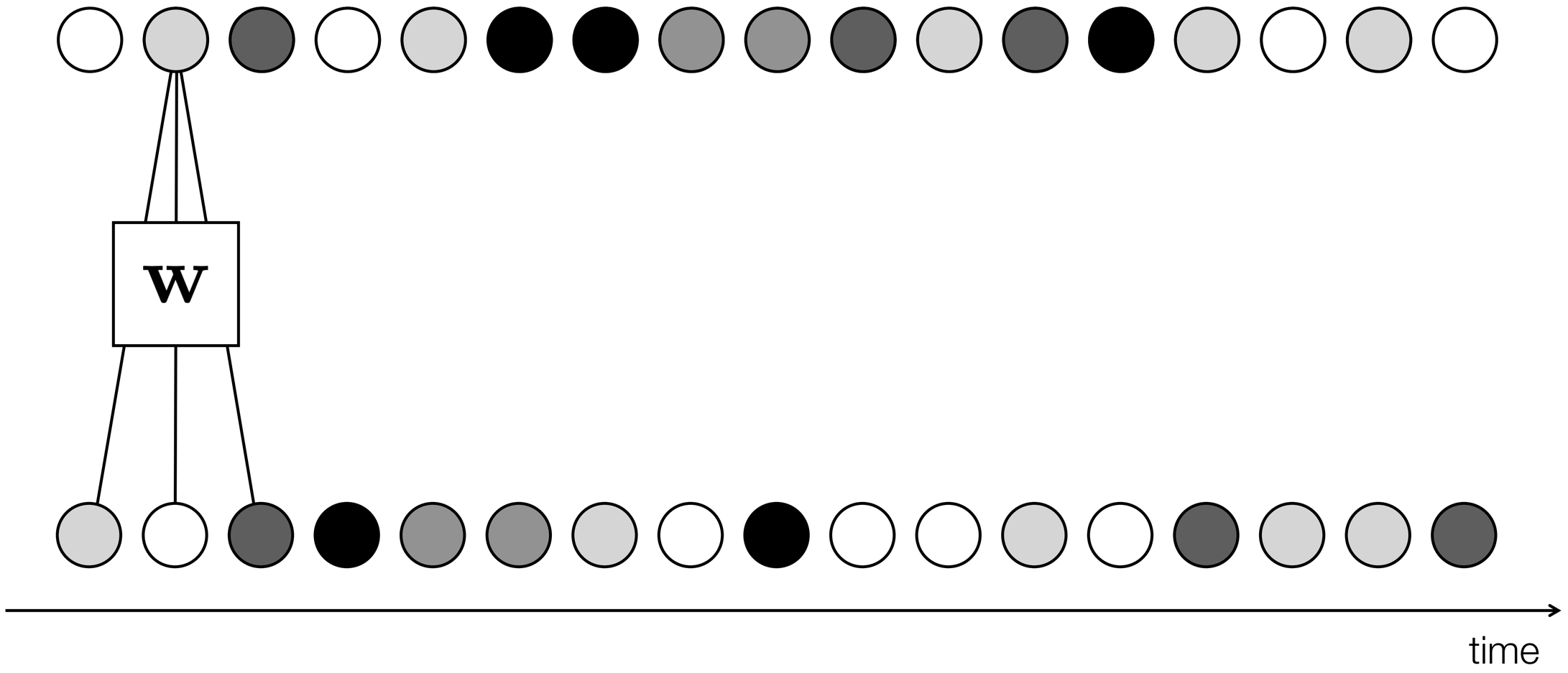

m
n
t

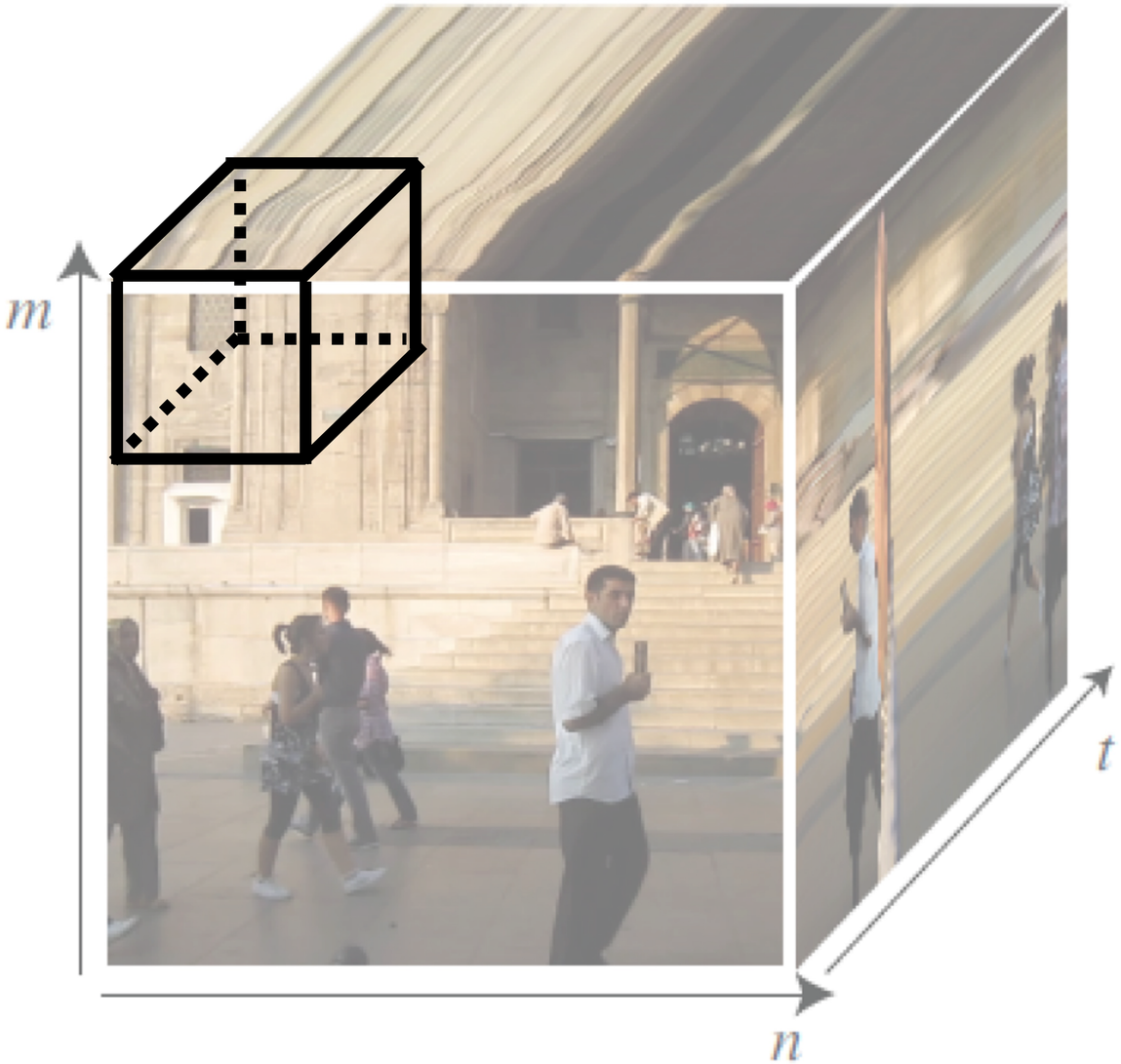

t
n

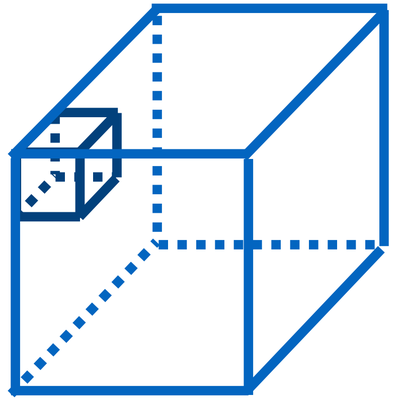


Bill Freeman and Phillip Isola

Convolutions in time



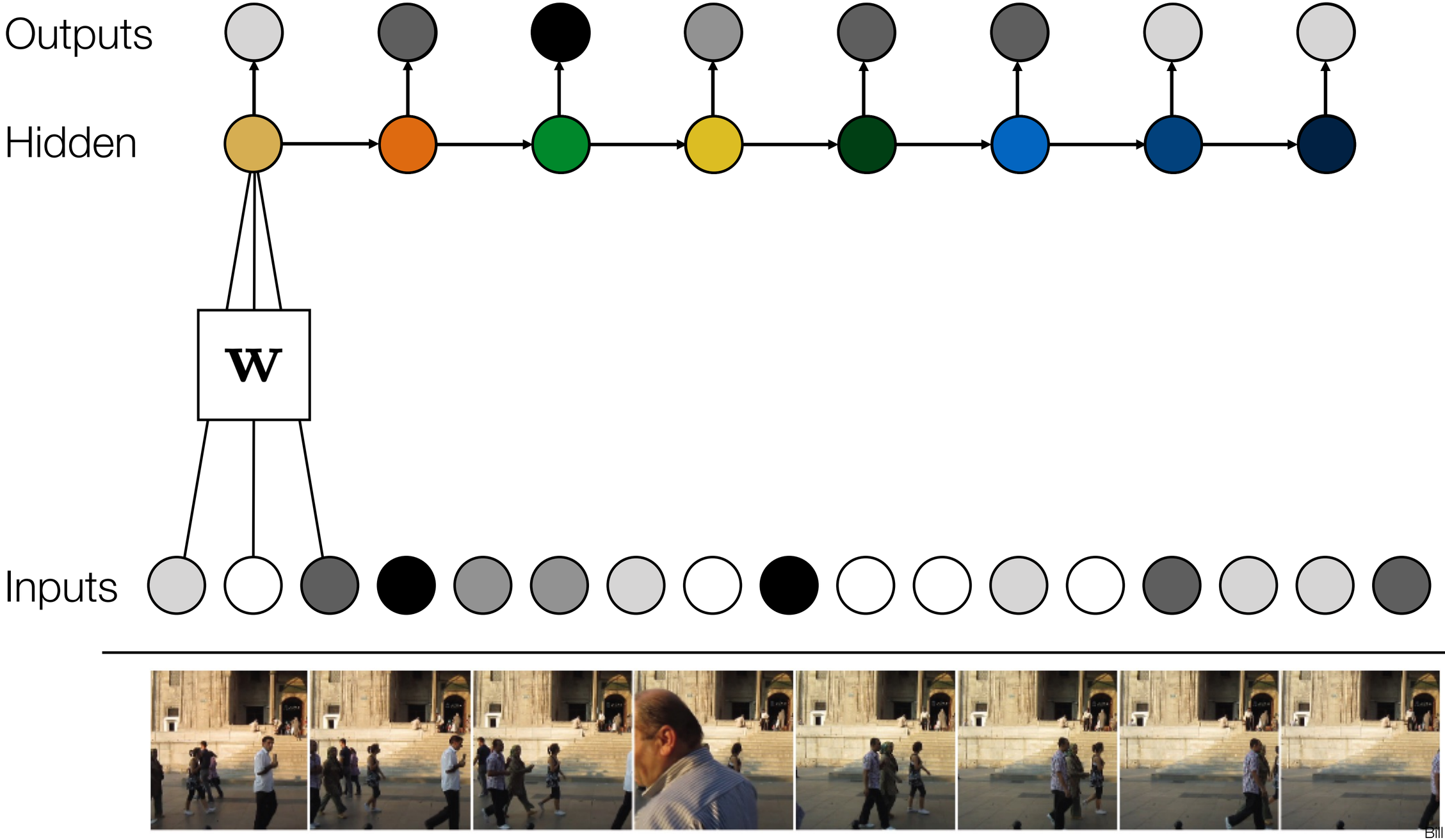


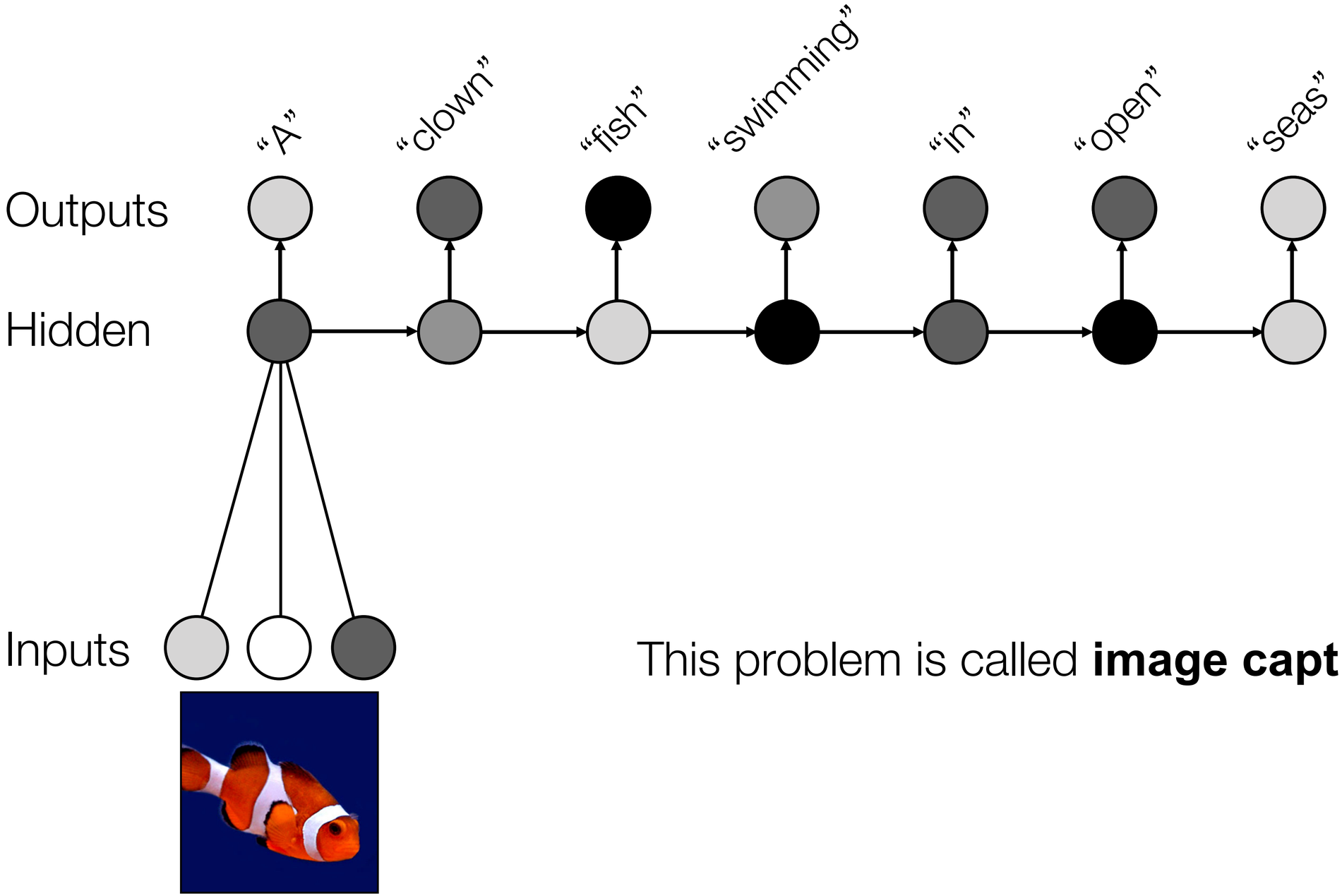


Potential problems?

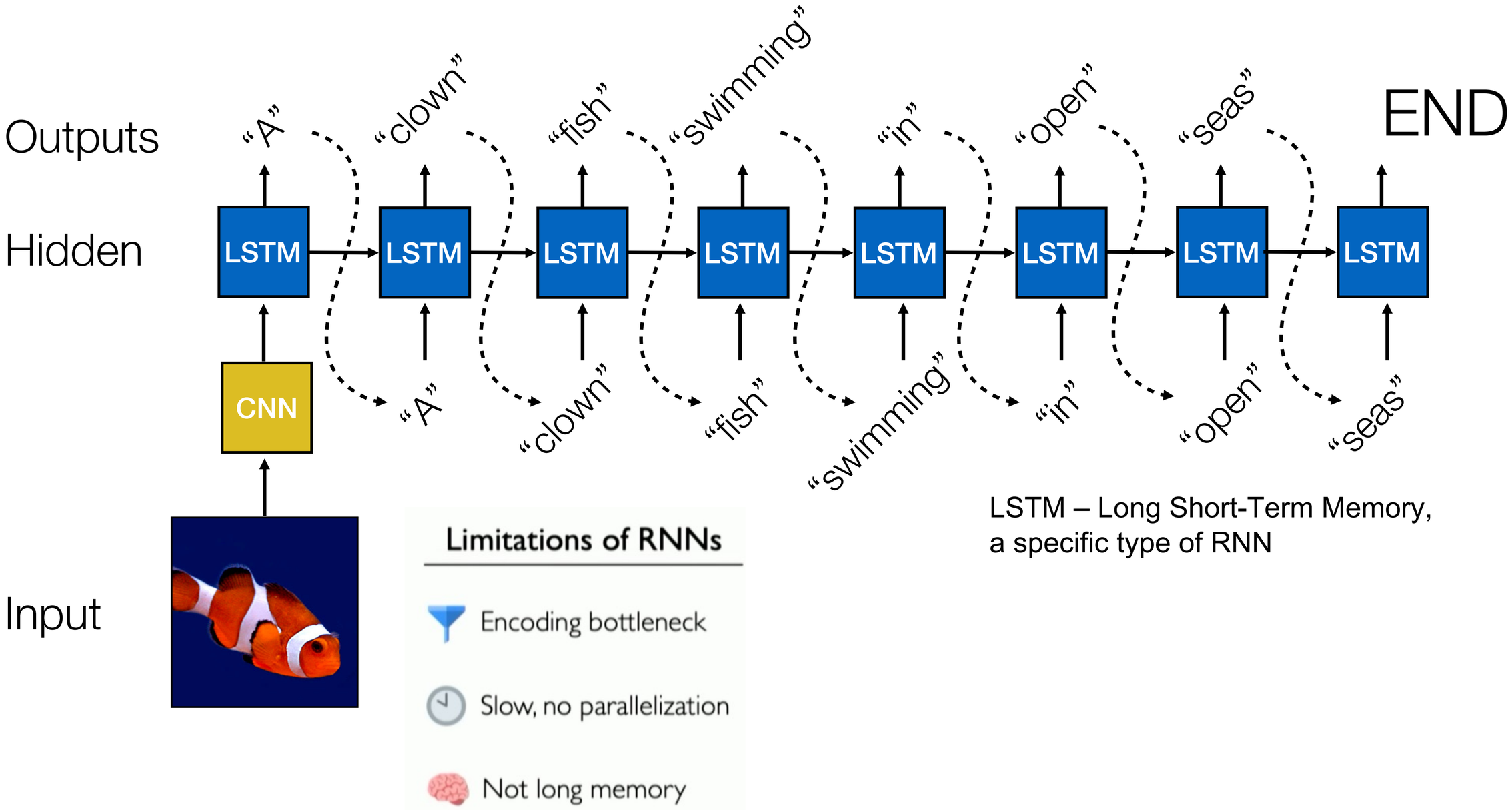
No memory.

Recurrent Neural Networks (RNNs)





This problem is called **image captioning**

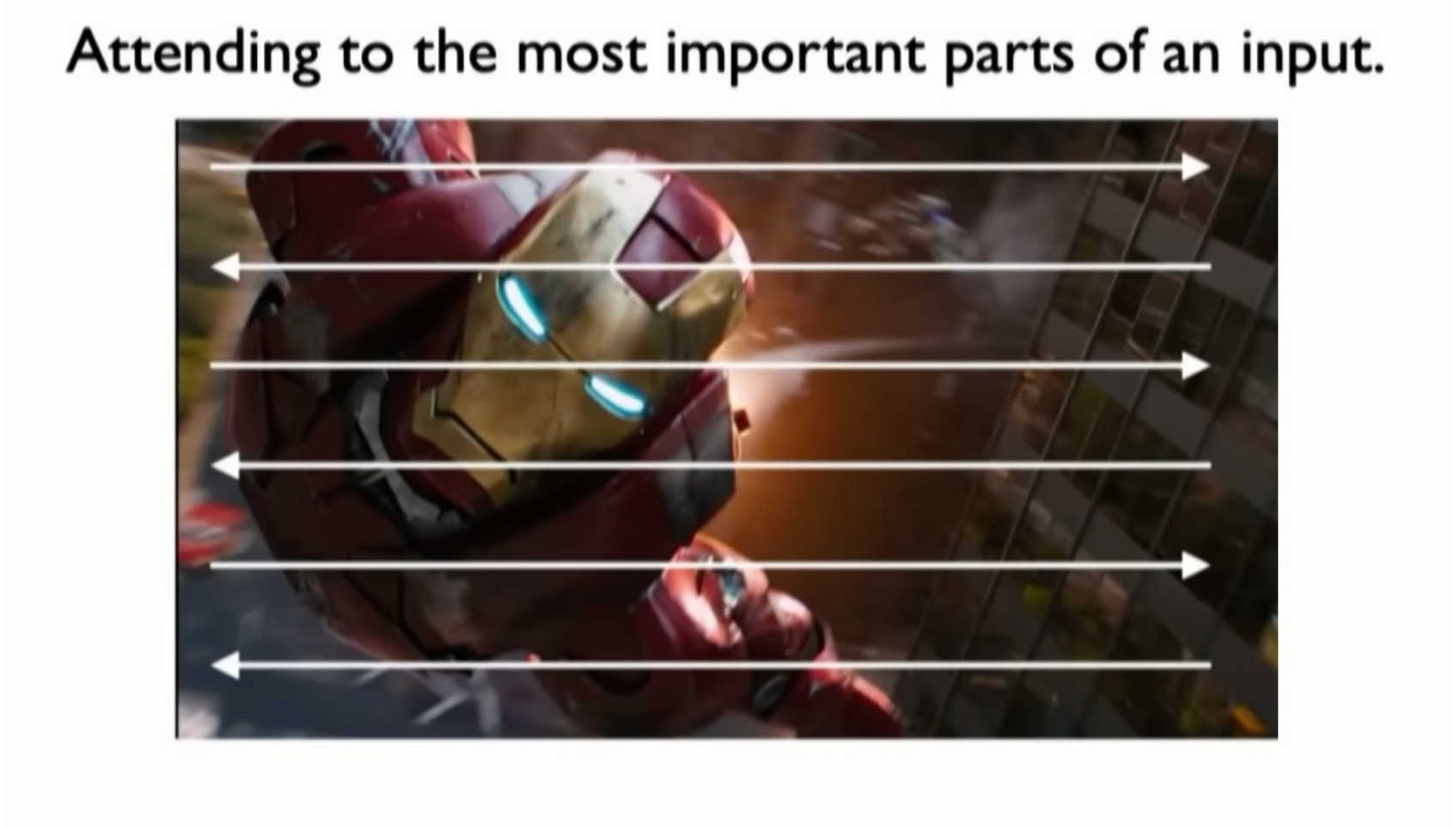


LSTM – Long Short-Term Memory,
a specific type of RNN

Limitations of RNNs

- Encoding bottleneck
- Slow, no parallelization
- Not long memory

Self-attention - Intuition



adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention - Intuition

Attending to the most important parts of an input.

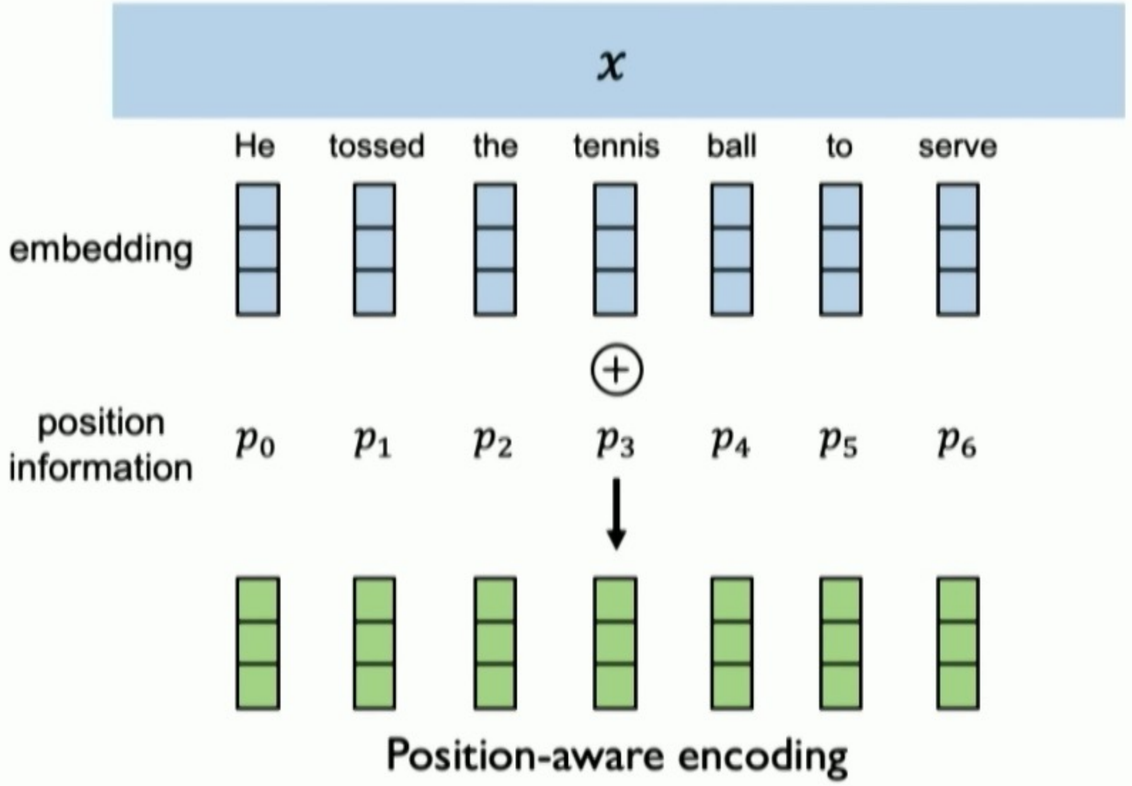
1. Identify which parts to attend to
2. Extract the features with high attention

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



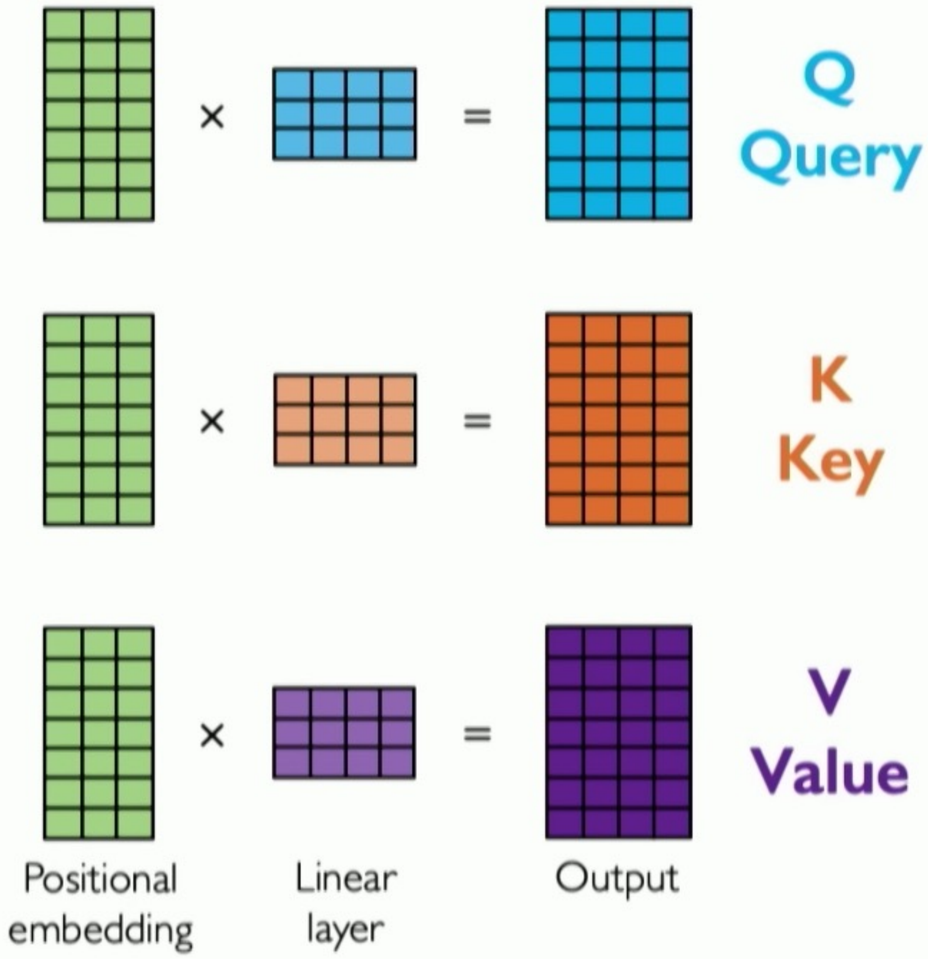
Data is fed in all at once! Need to encode position information to understand order.

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**



adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

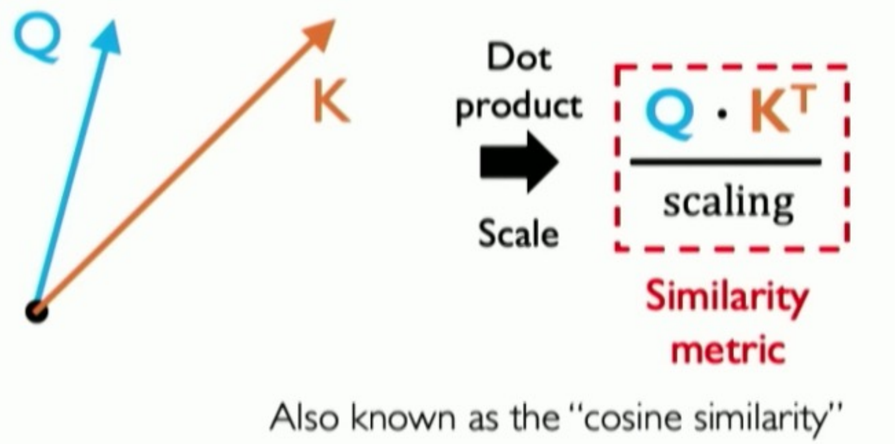
Self-attention

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention score: compute pairwise similarity between each query and key

How to compute similarity between two sets of features?



Also known as the "cosine similarity"

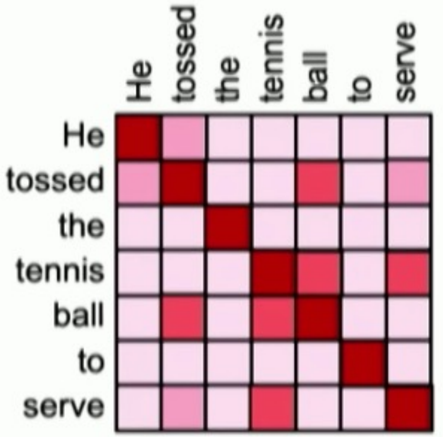
adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract features with high attention

Attention weighting: where to attend to! How similar is the key to the query?



$$softmax\left(\frac{Q \cdot K^T}{scaling}\right)$$

Attention weighting

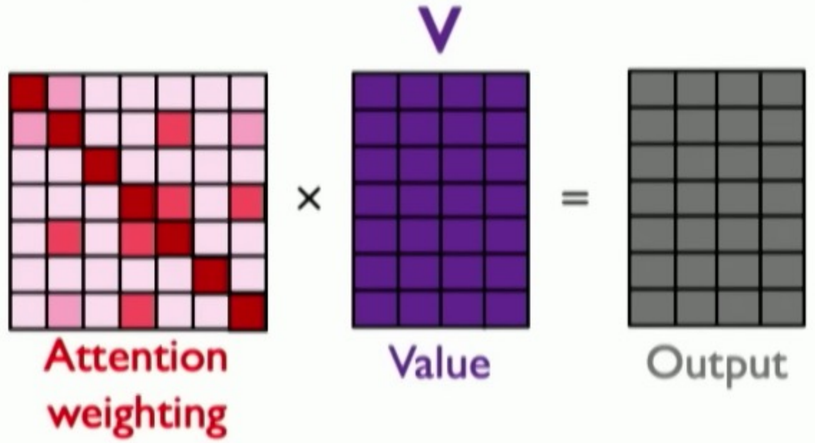
adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention

Goal: identify and attend to most important features in input.

1. Encode **position** information
2. Extract **query, key, value** for search
3. Compute **attention weighting**
4. Extract **features with high attention**

Last step: self-attend to extract features

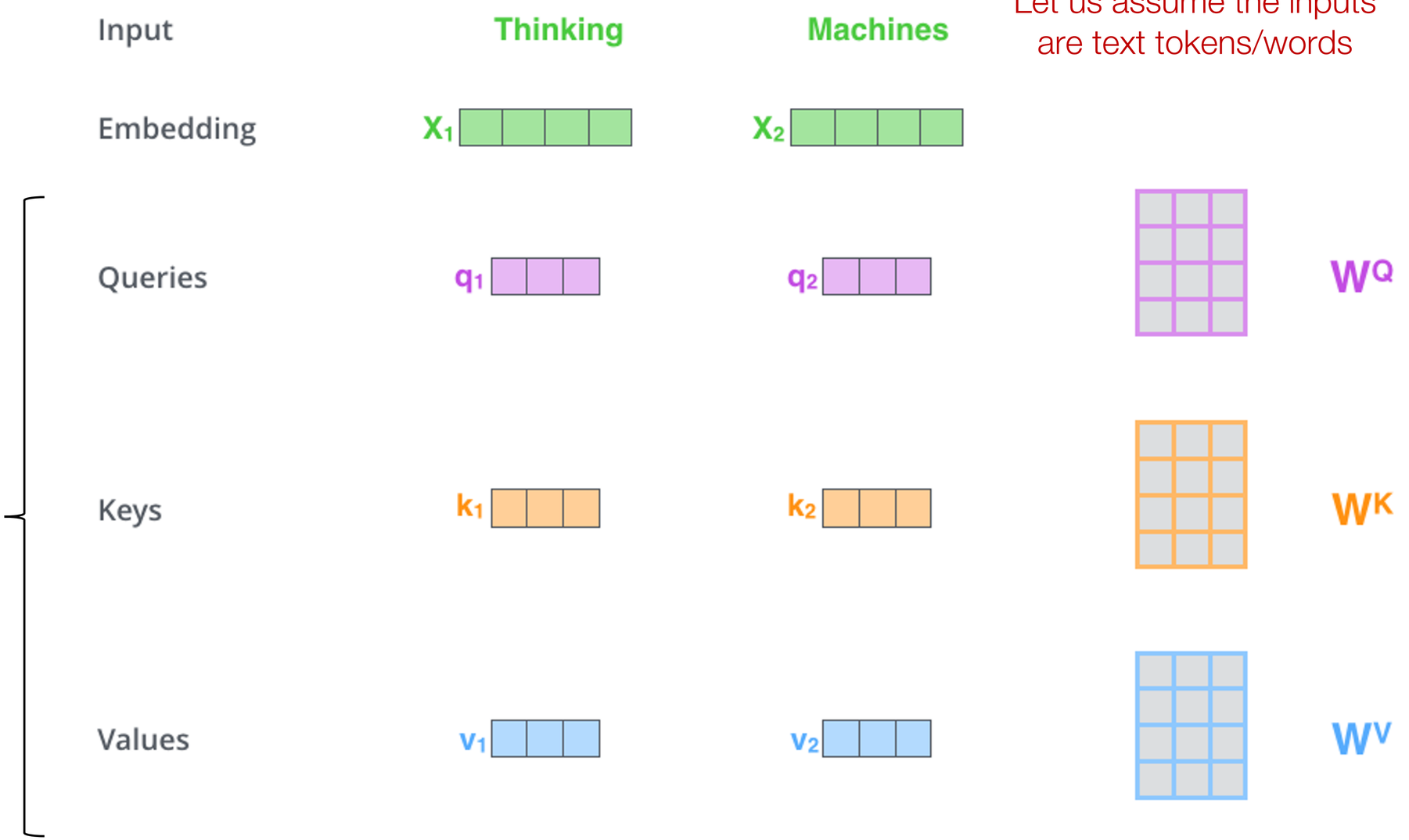


$$softmax\left(\frac{Q \cdot K^T}{scaling}\right) \cdot V = A(Q, K, V)$$

adapted from MIT 6.S191 Alexander Amini and Ava Soleimany

Self-attention – Example

Let us assume the inputs are text tokens/words

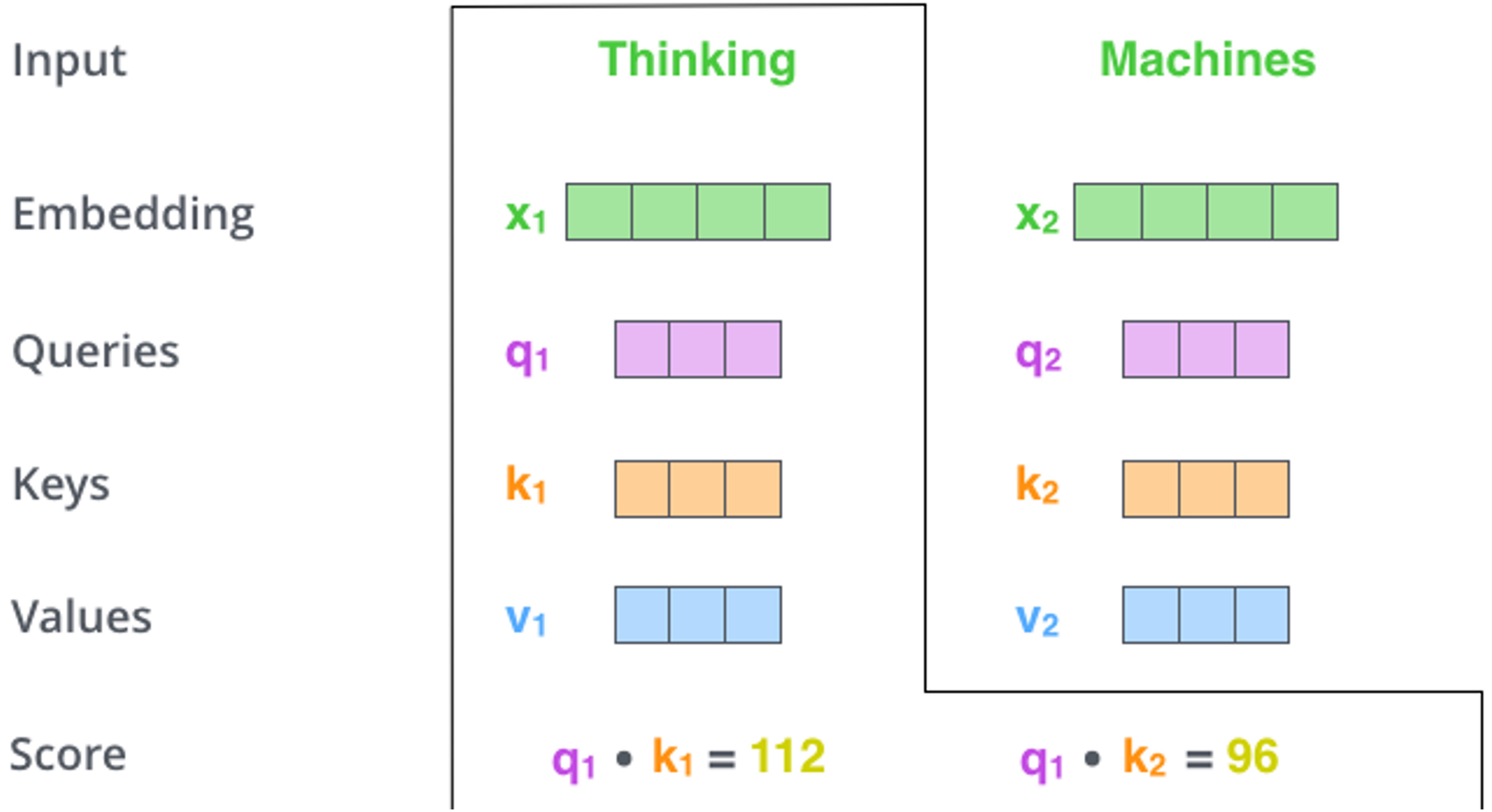


First we create 3 vectors by multiplying input embedding (1x512) x_i with three matrices (64x512):

$q_i = x_i W^Q$
$K_i = x_i W^K$
$V_i = x_i W^V$

Source: Ming Li

Self-attention – Example

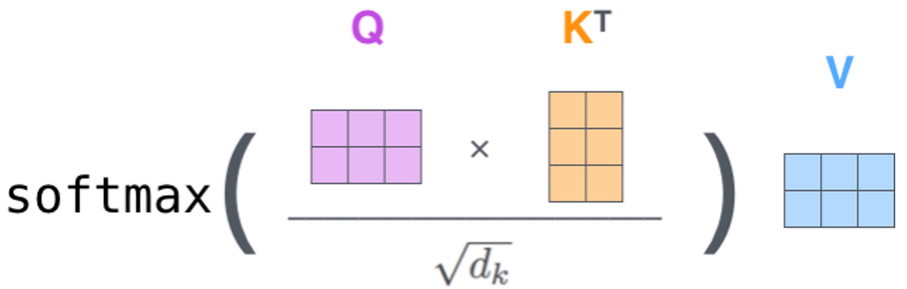


Now we need to calculate the **attention score** to determine how much focus to place on other parts of the input.

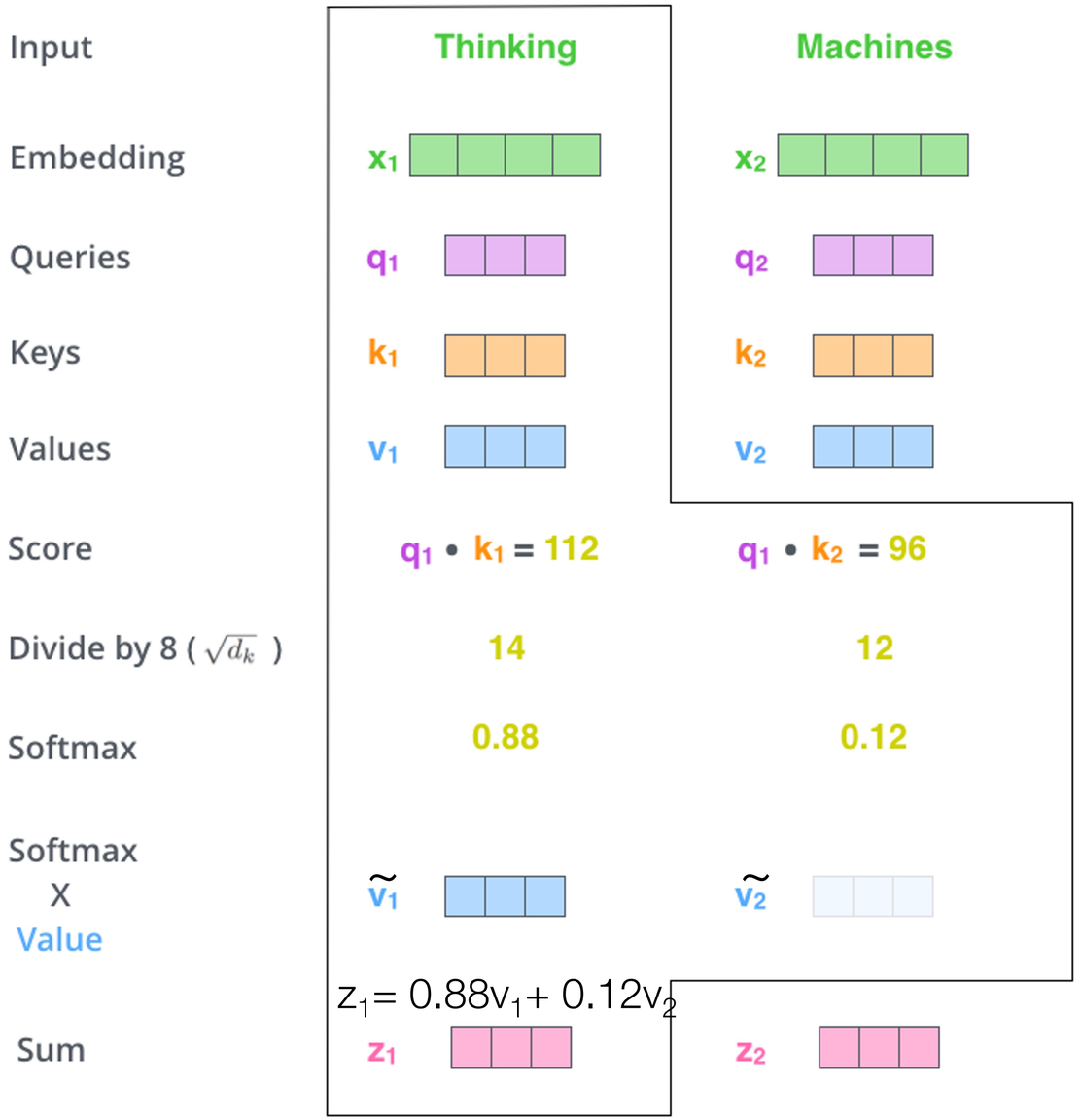
Source: Ming Li

Self-attention – Example

Considering scaled dot-product attention:



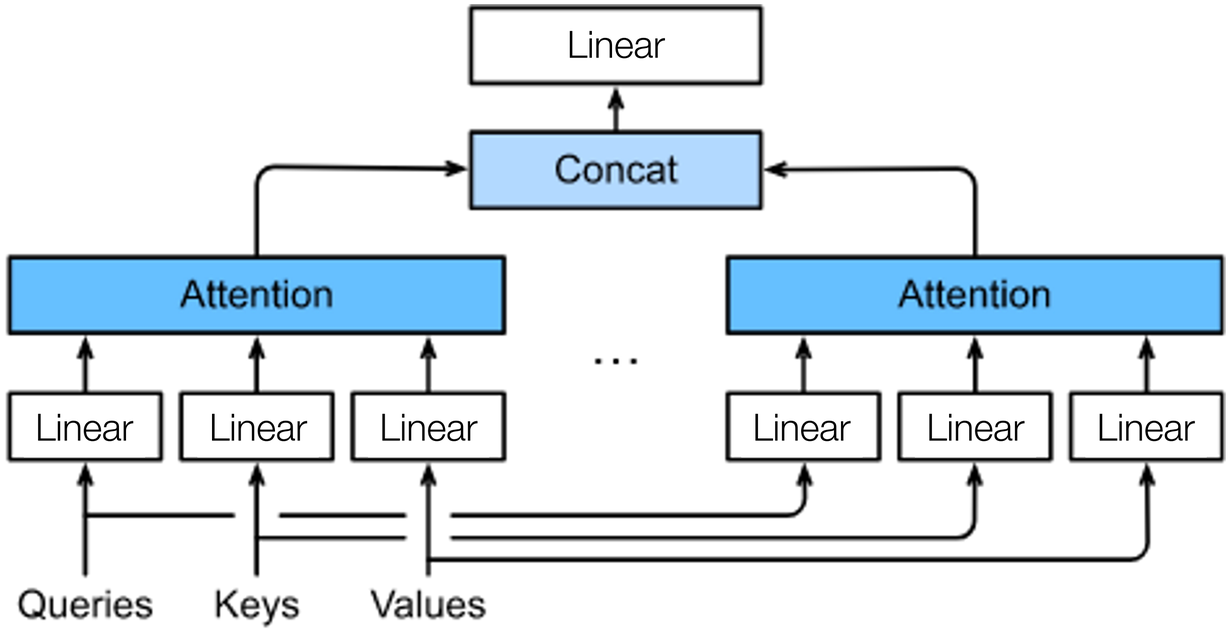
d_k=64 is dimension of key vector



Source: Ming Li

Multi-head attention

- Expands the model's ability to focus on different positions and gives the attention layer multiple "representation subspaces"



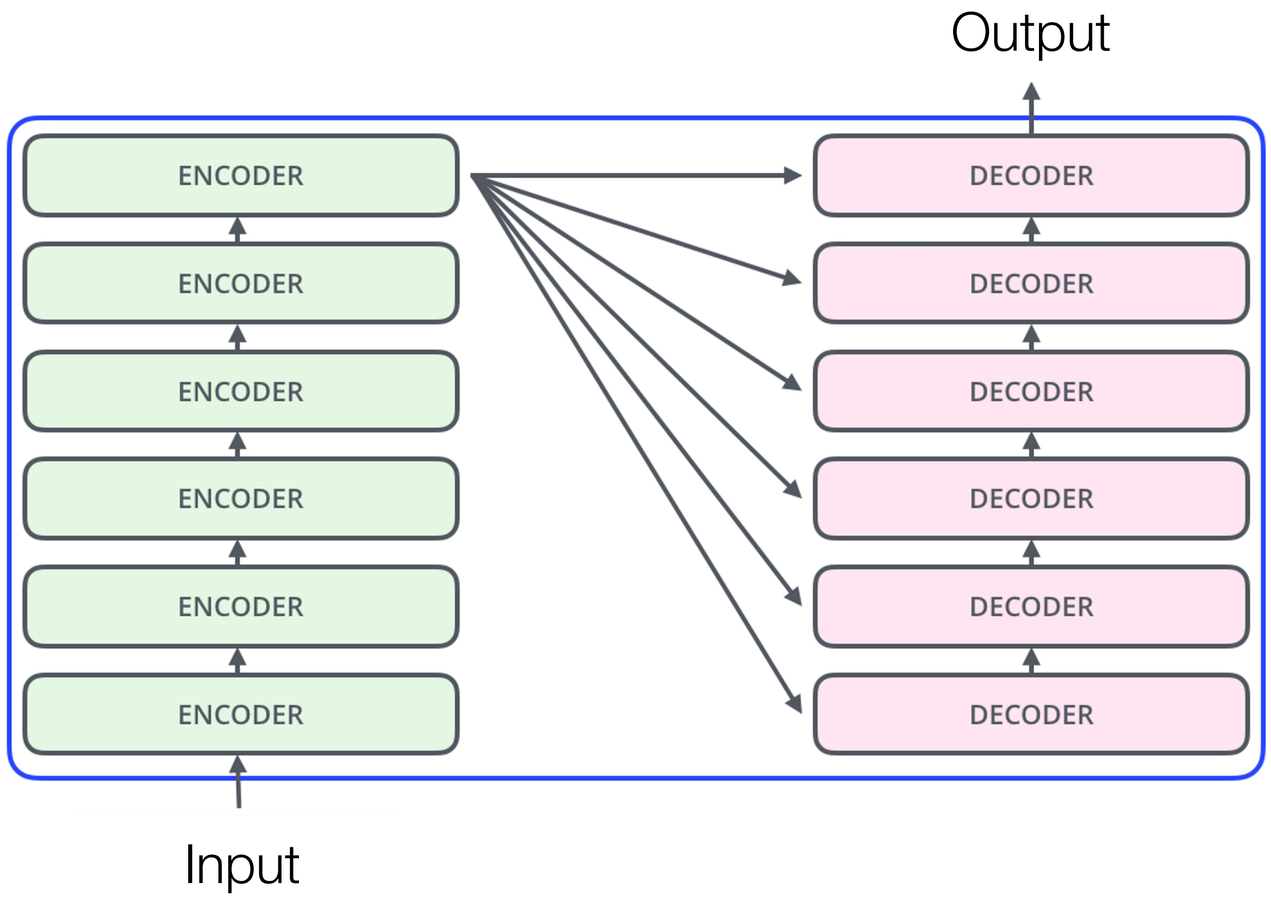
1. queries, keys, and values can be **transformed** with h independently learned linear projections
2. the h projected queries, keys, and values are fed into **attention pooling in parallel**
3. attention pooling outputs are **concatenated and transformed** with another learned linear projection to produce the final output

Vaswani et al. Attention Is All You Need
NeurIPS 2017

Source: https://d2l.ai/chapter_attention-mechanisms

Transformer

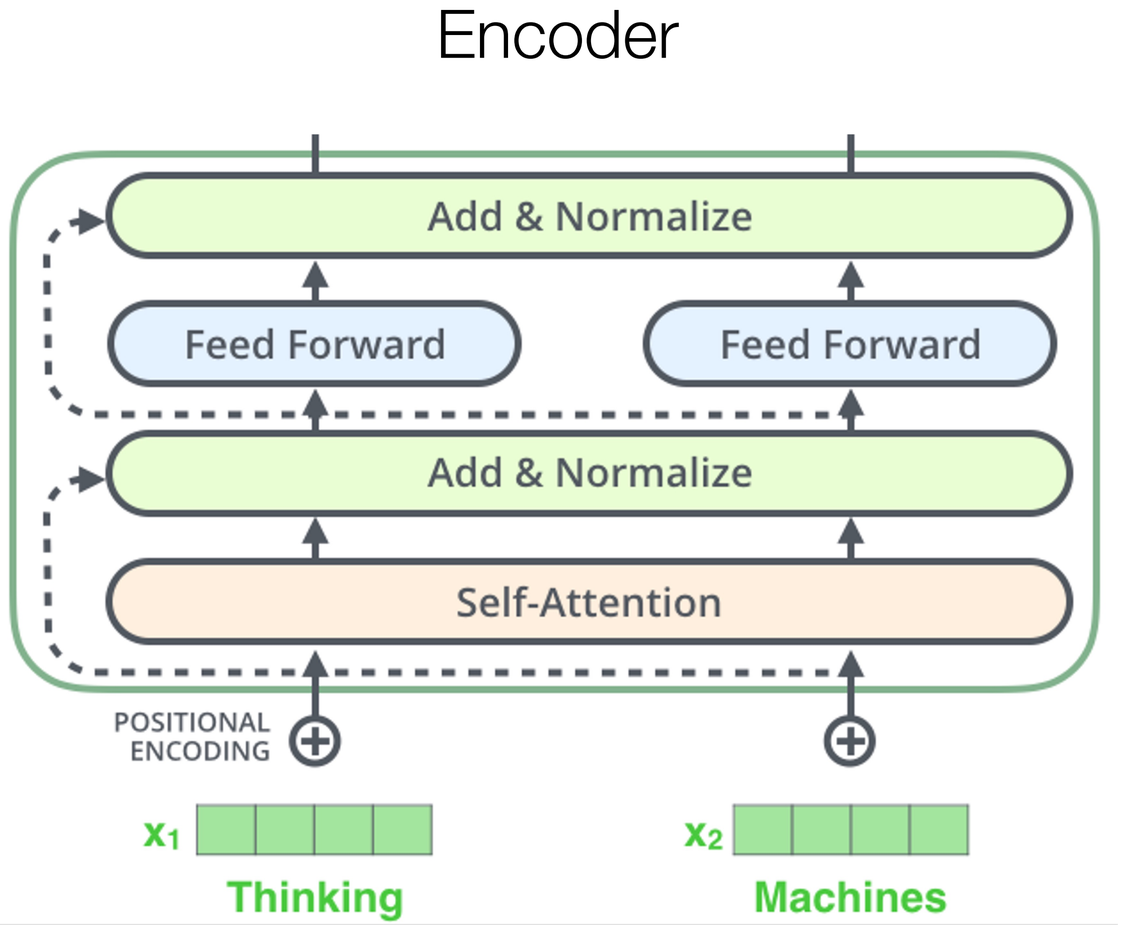
- The transformer adopts an **encoder-decoder architecture**
- Consists of a stack of encoder blocks and a stack of decoder blocks
- The encoder output is given to each decoder block



Adapted from Ming Li

Transformer

- Each encoder consists essentially of a **self-attention layer** and a **feed forward network** (only one hidden layer)
- The feed forward network is independent for each input → can be **parallelized**
- Add & normalize layers are also added



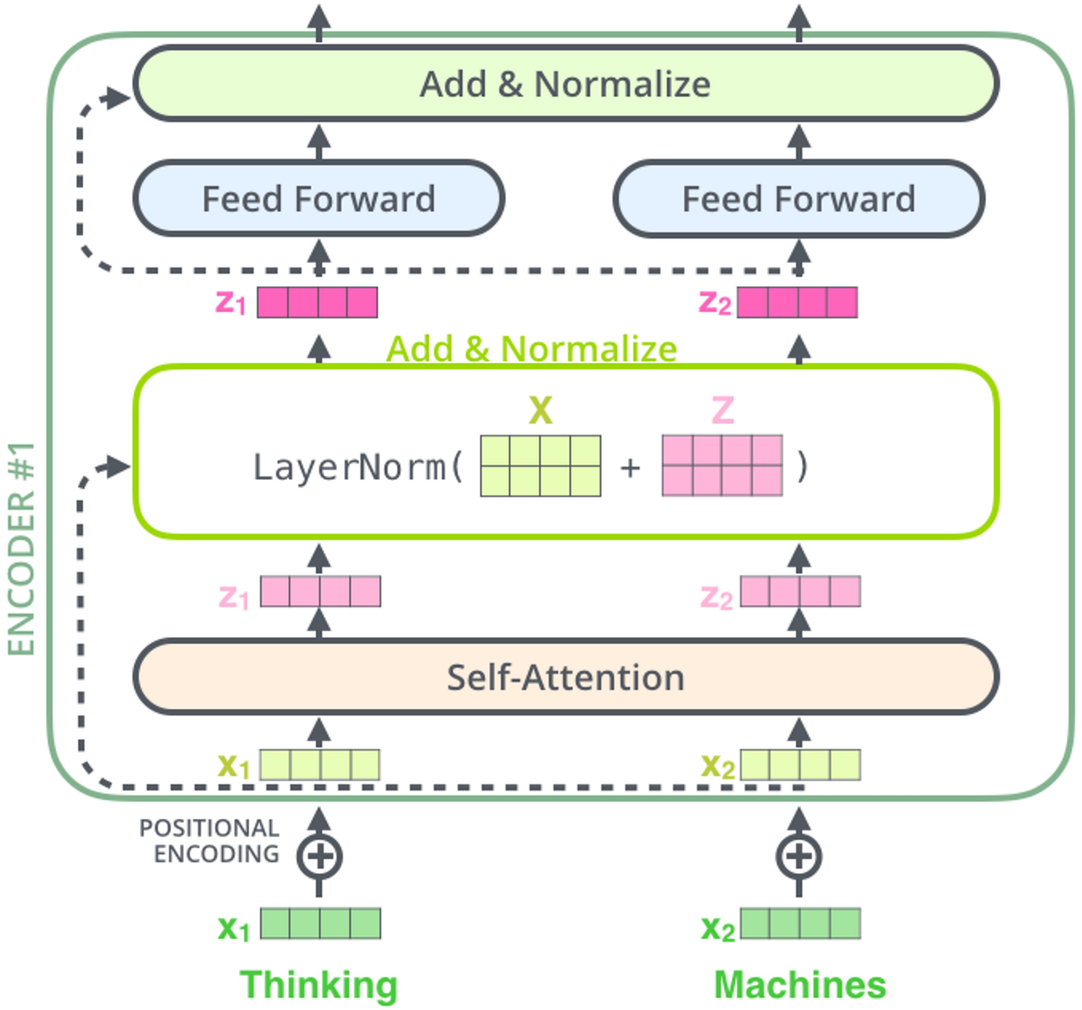
Adapted from Ming Li

Transformer

Add & Normalize

Both the **residual connections** and the **layer normalization** in the transformer are important for training a very deep model.

Layer normalization normalizes the inputs **across the features**.

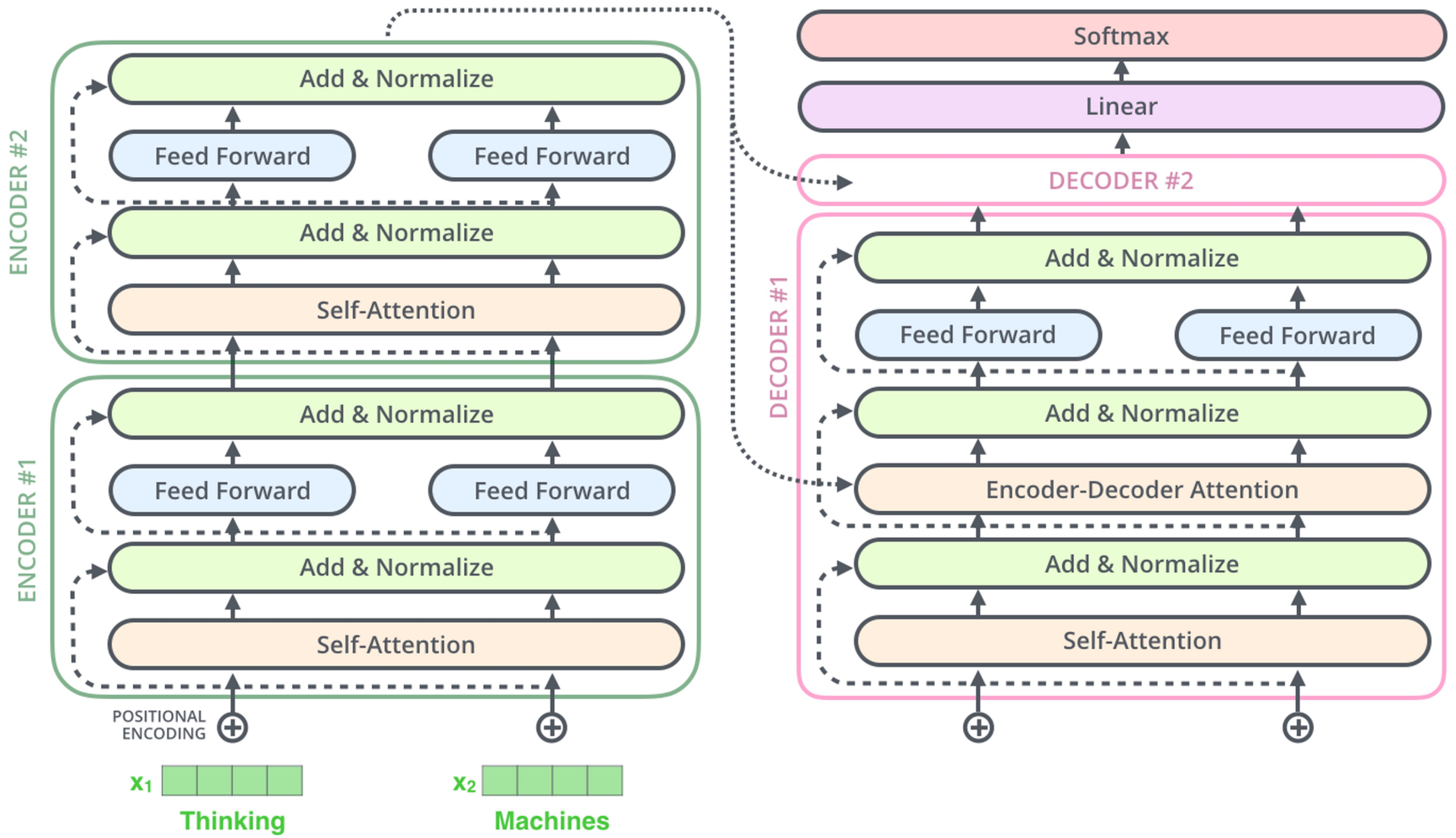


Adapted from Ming Li

Transformer

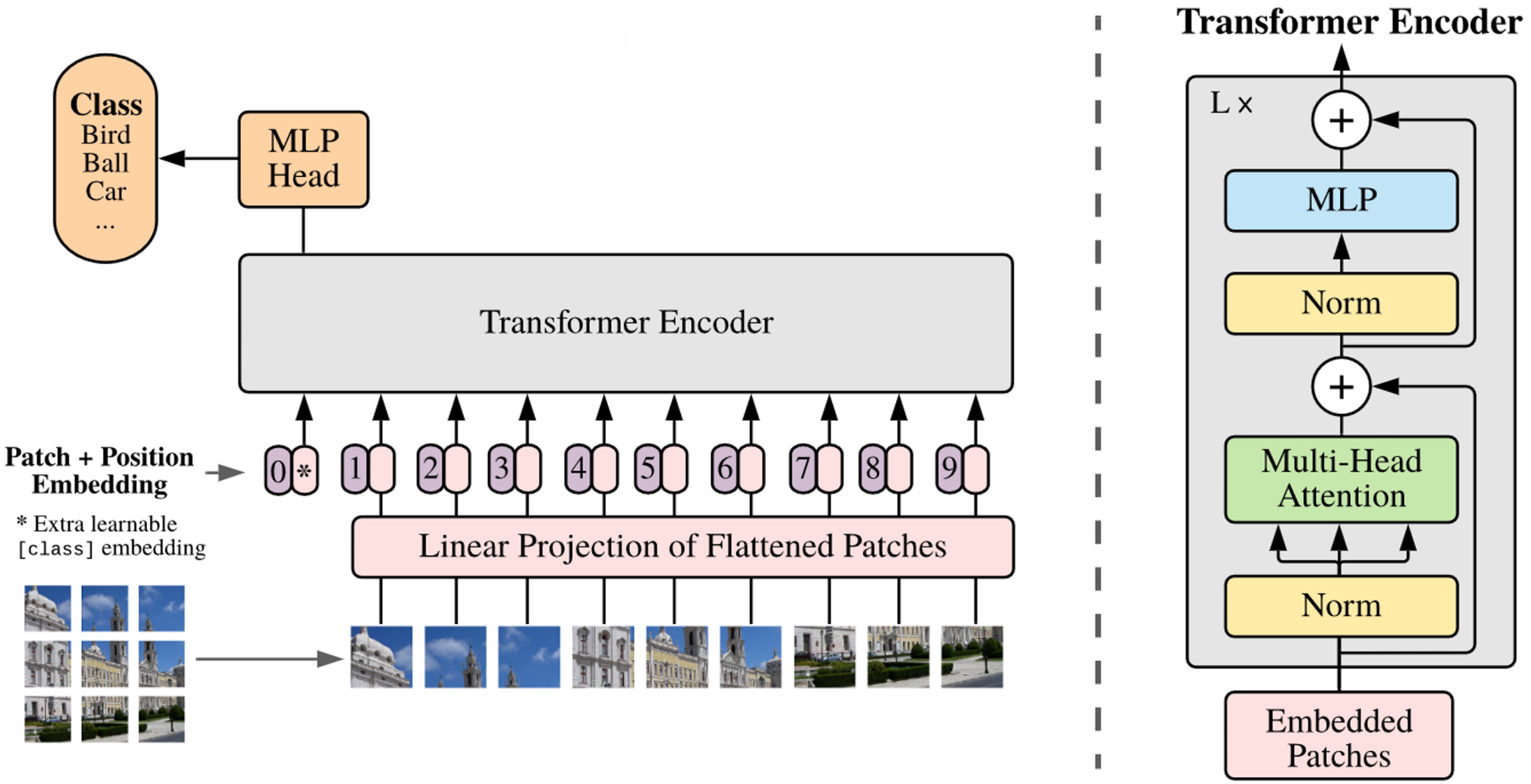
The encoder-decoder attention is similar to self attention, except it uses K, V from the top of encoder output, and its own Q

Complete transformer



Adapted from Ming Li

Vision Transformer (ViT)



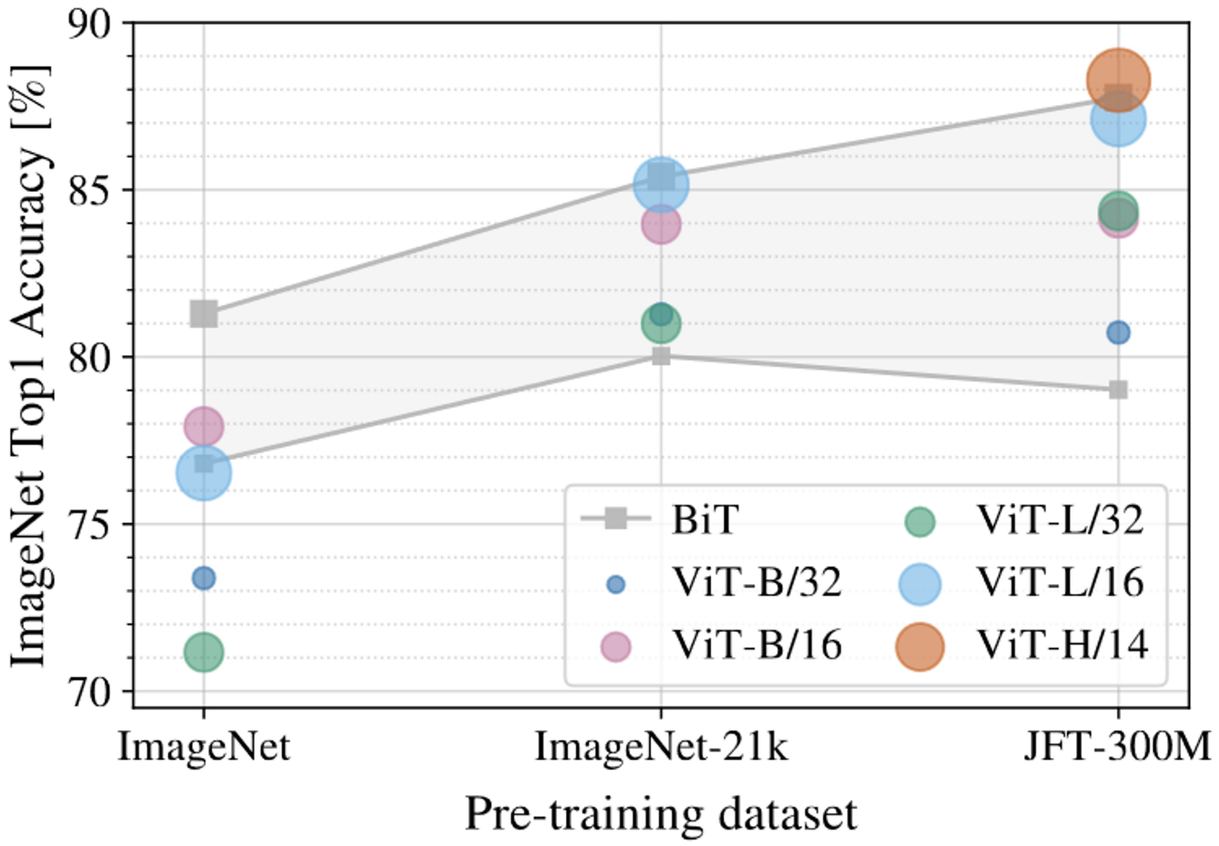
Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, arXiv 2020

Vision Transformer (ViT)

Model	Layers	Hidden size D	MLP size	Heads	Params
ViT-Base	12	768	3072	12	86M
ViT-Large	24	1024	4096	16	307M
ViT-Huge	32	1280	5120	16	632M

	ViT -JFT (ViT-H/14)	ViT -JFT (ViT-L/16)	ViT -I21K (ViT-L/16)	BiT-L (ResNet152x4)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k

Vision Transformer (ViT)

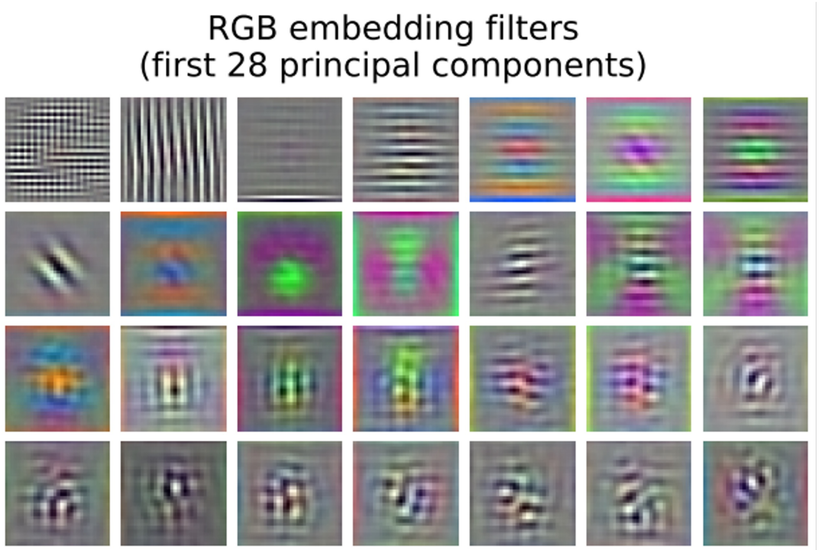


When trained on mid-sized datasets such as ImageNet, such models yield modest accuracies of a few percentage points below ResNets of comparable size. This seemingly discouraging outcome maybe expected: Transformers **lack some of the inductive biases inherent to CNNs**, such as translation equivariance and locality, and therefore **do not generalize well when trained on insufficient amounts of data.**

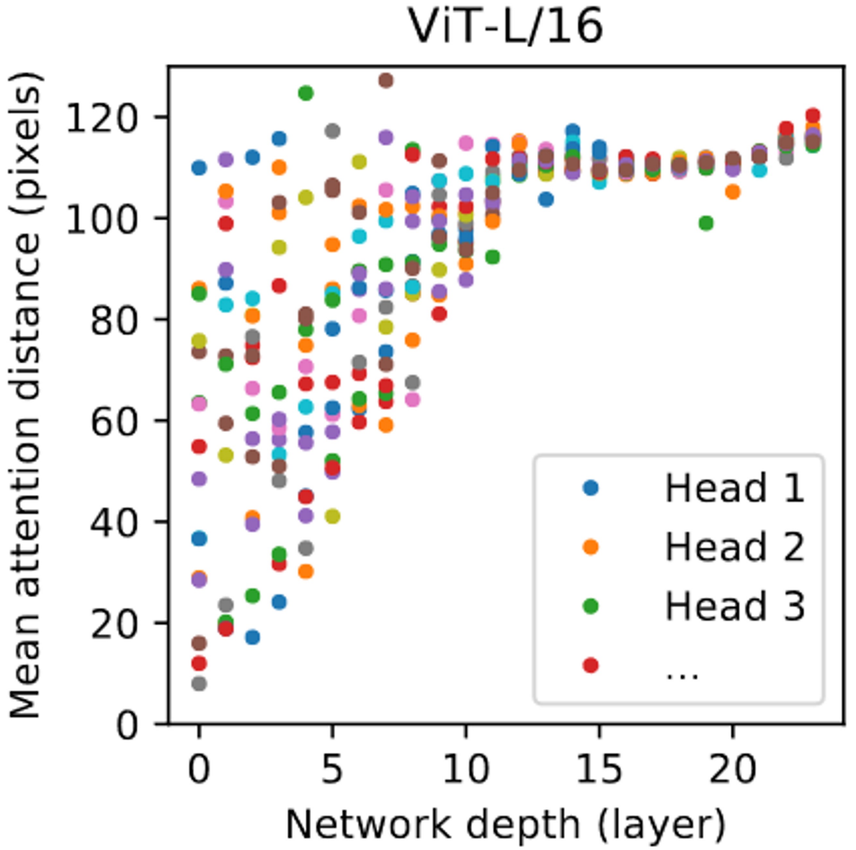
However, the picture changes if the models are trained on larger datasets (14M-300M images). We find that large scale training trumps inductive bias.

Dosovitskiy et al.

Vision Transformer (ViT)

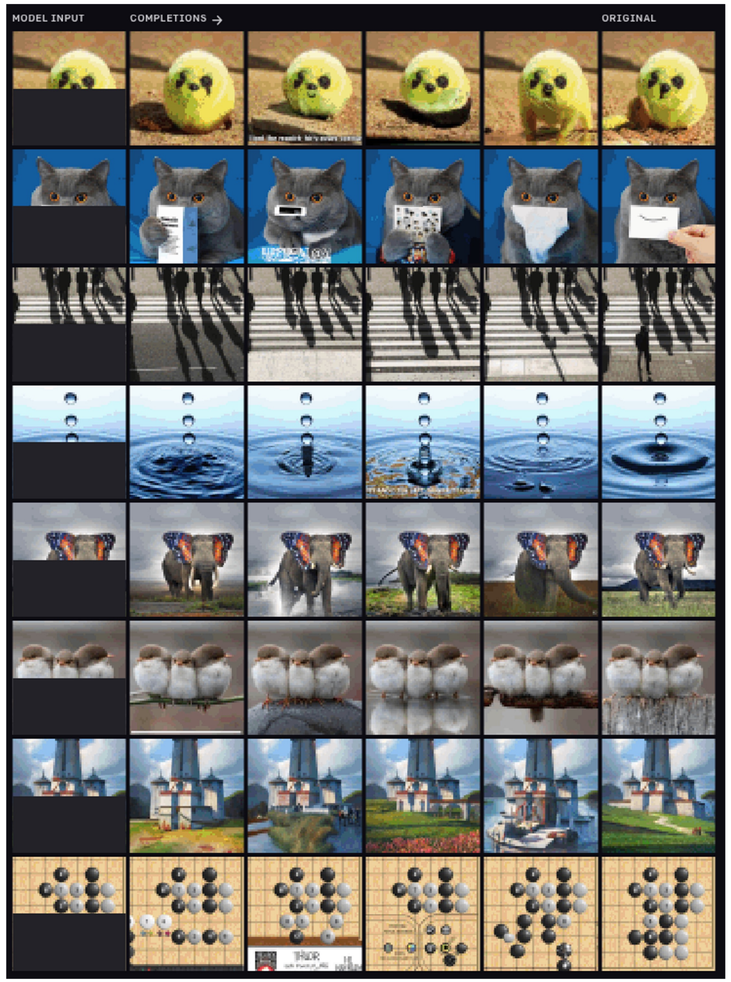


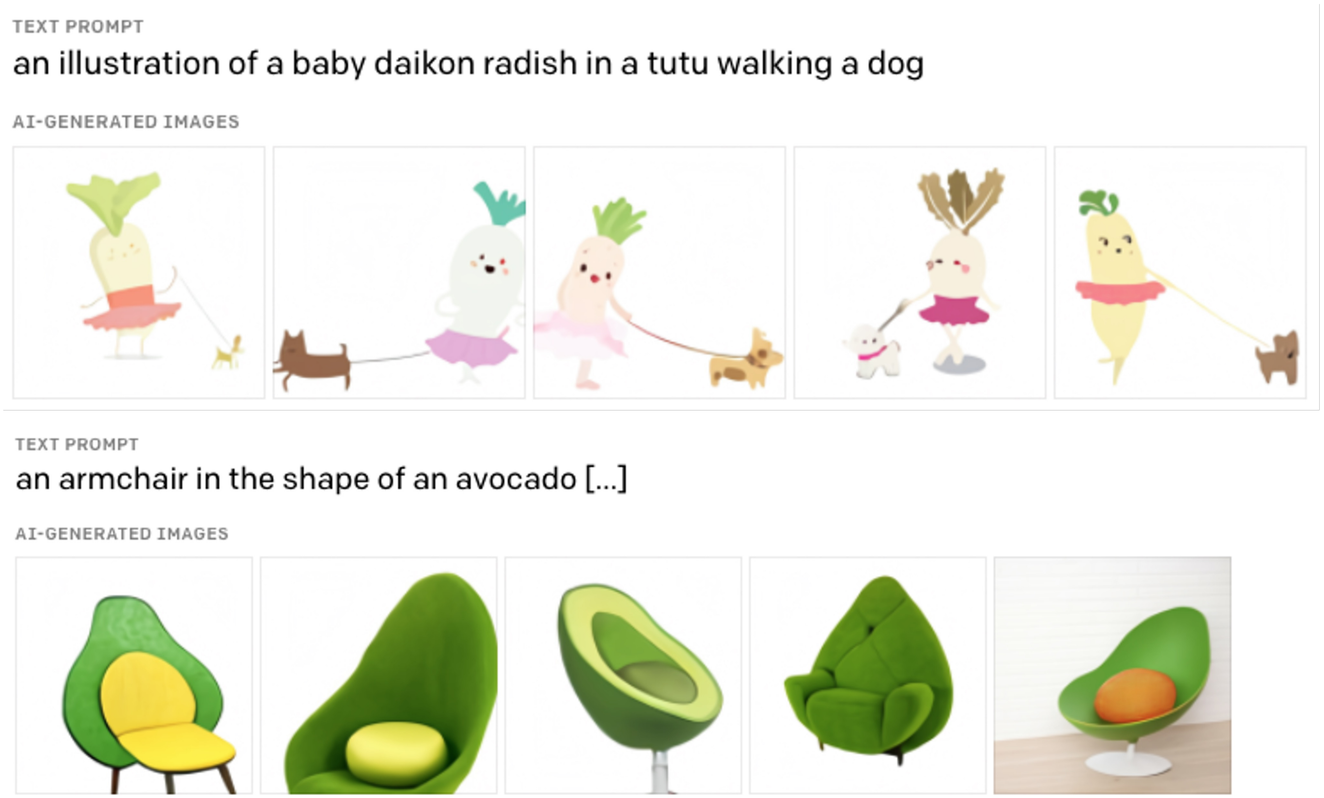
Filters of the initial linear embedding of RGB values of ViT-L/32



Some heads attend to most of the image already in the lowest layers → ability to integrate information globally

Image GPT and DALL-E





Chen et al. Generative Pretraining from Pixels, ICML 2020
Ramesh et al. Zero-Shot Text-to-Image Generation, arXiv 2021

Images source: Open.ai blog