Seminar in Deep Reinforcement Learning

Part I
Deep Learning and Neural Architecture

Zhao MA

25.02.2020

Deep Learning in **One** Slide

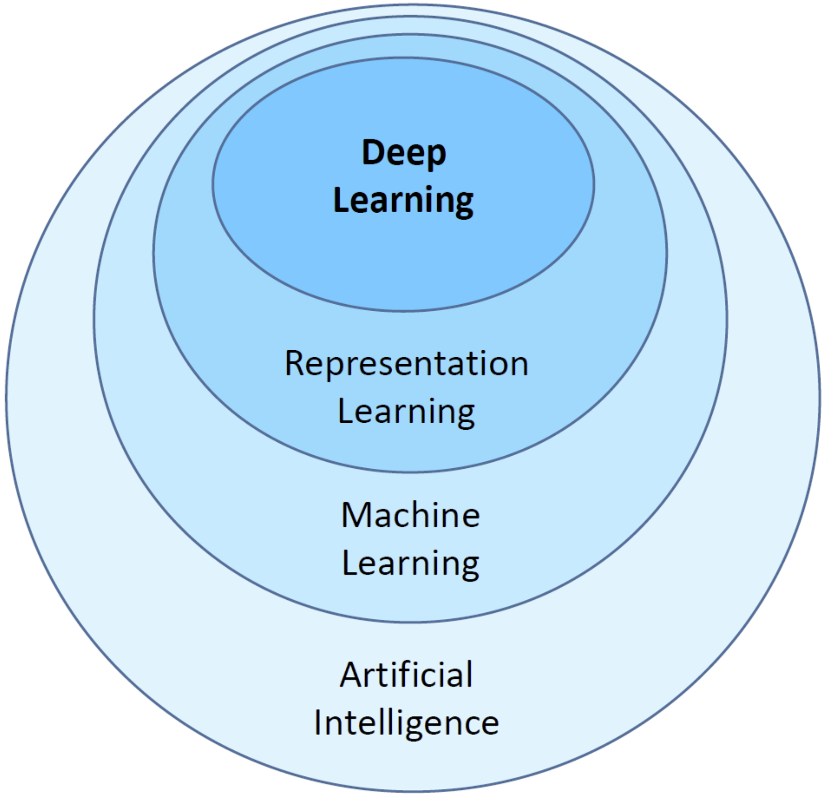
- **What is it:**
  Extract useful patterns from data.
- **How:**
  Neural network + optimization
- **How (Practical):**
  Python + TensorFlow & friends
- **Hard Part:**
  Good Questions + Good Data
- **Why now:**
  Data, hardware, community, tools, investment
- **Where do we stand?**
  Most big questions of intelligence have not been answered nor properly formulated

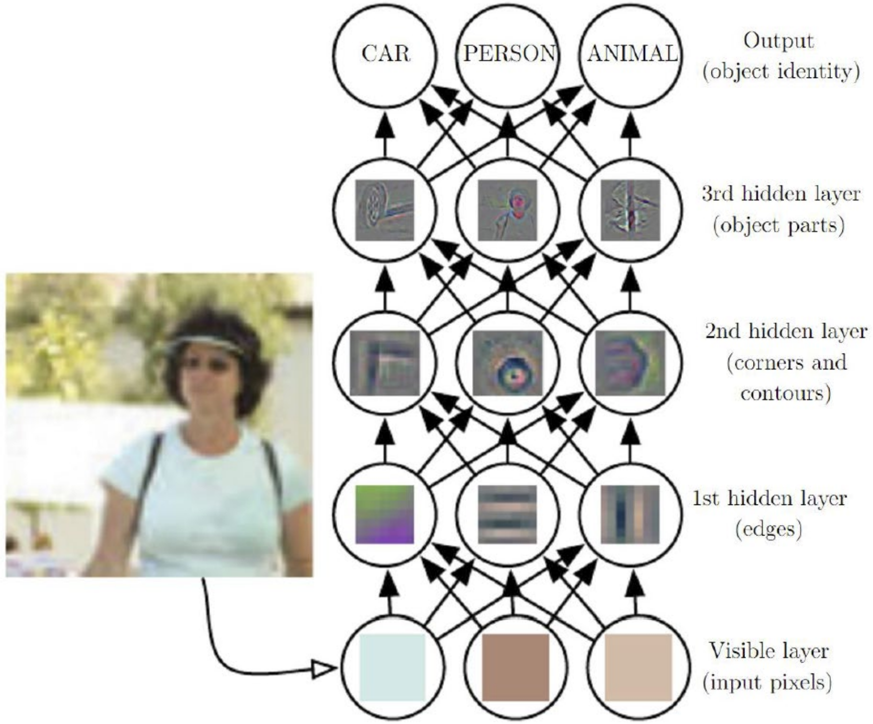
Exciting progress:

- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Cars: drivable area, lane keeping
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep RL

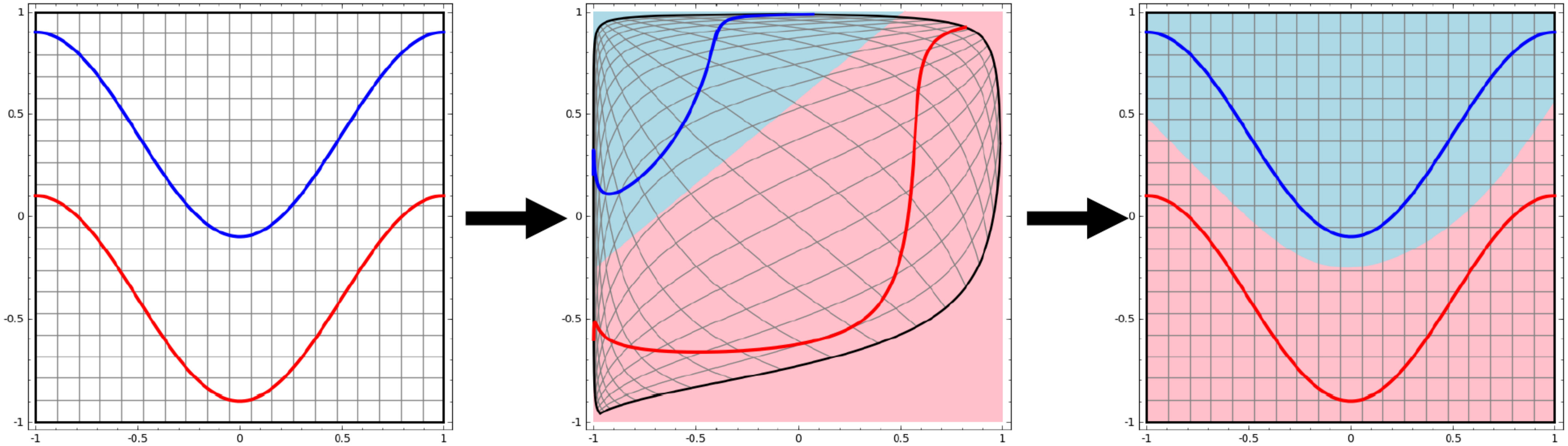
Deep Learning is **Representation Learning**

Representation Learning:
the automated formation of useful representations from data.



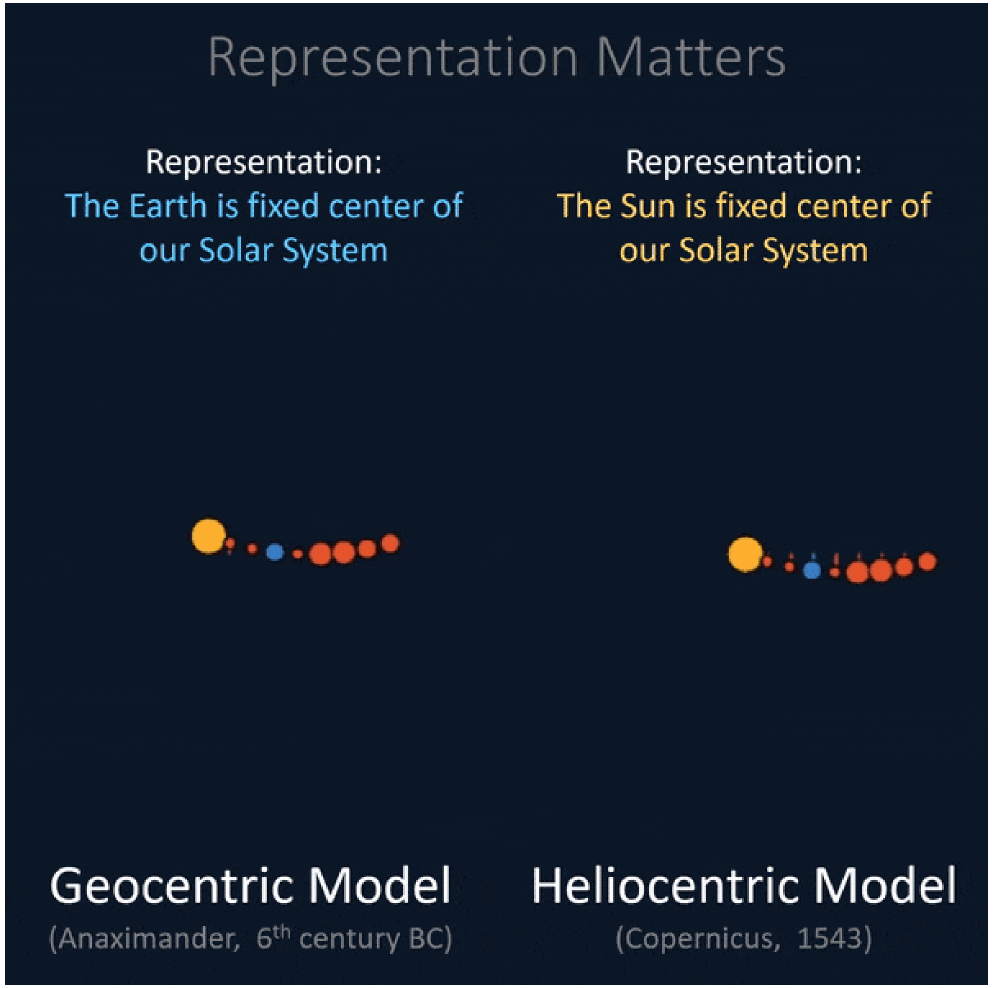


Representation Matters



Task: Draw a line to separate the **blue curve** and **red curve**

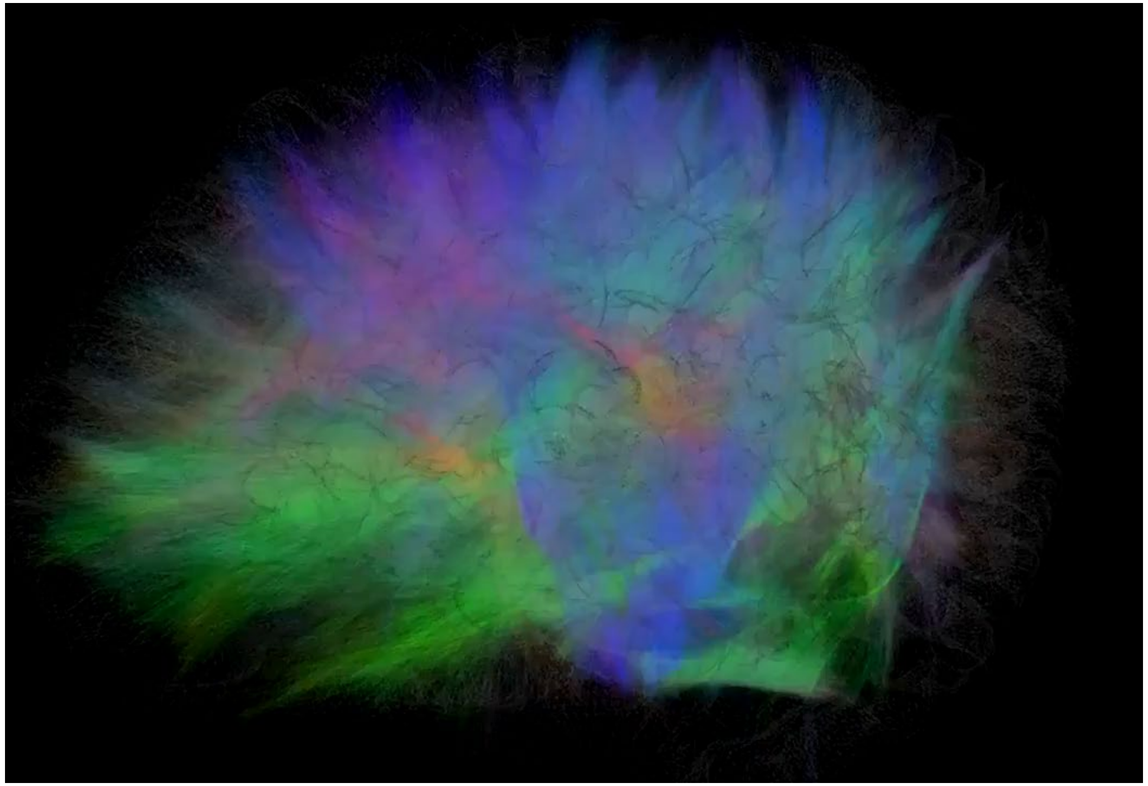
Representation Matters



How we represent the world can **make the complex appear simple both to us humans and to the machine learning models we build.**

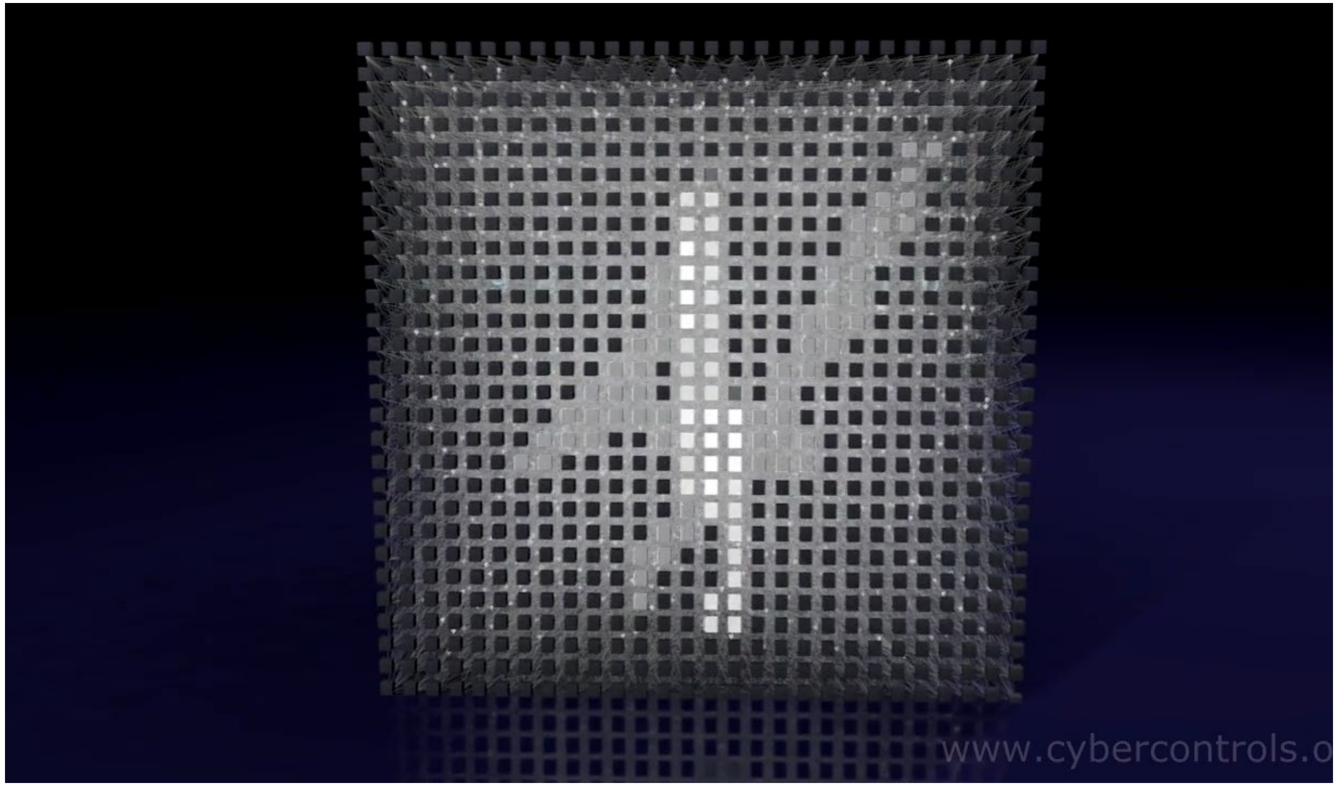
"AI began with an ancient wish to forge the gods."

-Pamela McCorduck, *Machines Who Think, 1979*



3% of the neurons and 0.0001% of the synapses in the brain.

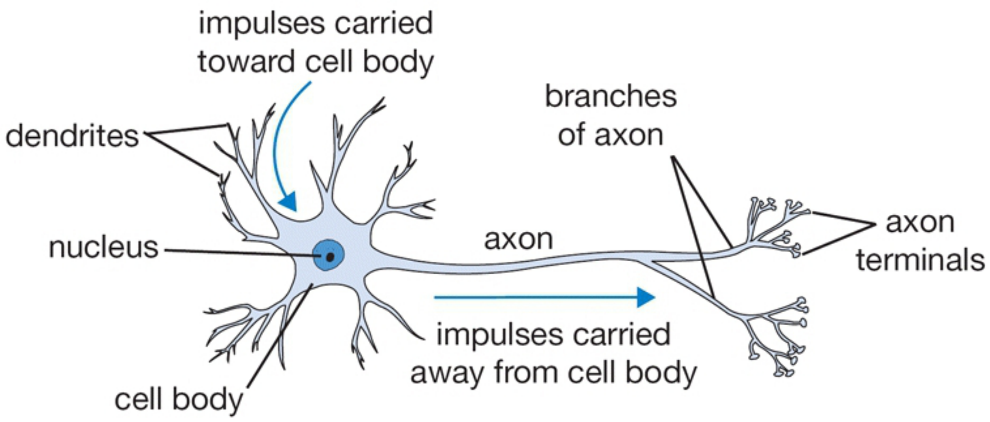
Thalamocortical systemvisualization via DigiCortexEngine.

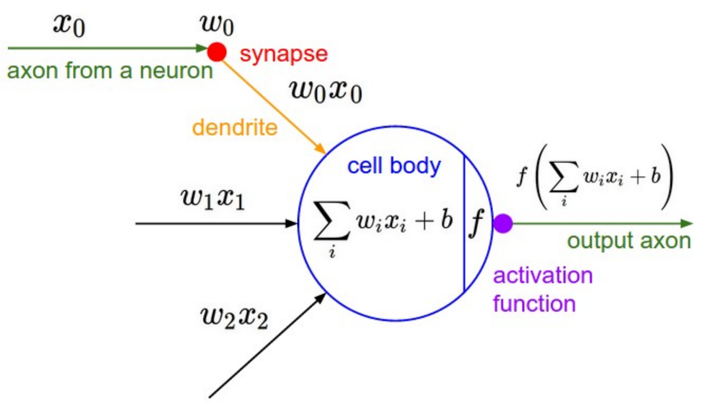


Visualization of MNIST dataset classification.

www.cybercontrols.org

Neuron: Biological Inspiration for Computation





Neuron: computational building block for the brain.

(Artificial) Neuron: computational building block for the "neural network"

Why does it work?

http://cs231n.github.io/neural-networks-1/

Content

Universal Approximation Theorem

Common DL Architectures

Selected NNs in details

CNN (+ResNet), RNN (+LSTM), Transformer (+Attention)

Deep Double Decent

Universal approximation theorem

“A **feed-forward** network with **a single hidden layer** containing a **finite number of neurons** can approximate continuous functions on compact subsets of $\mathbf{R}^n$, under mild assumptions on the activation function.”

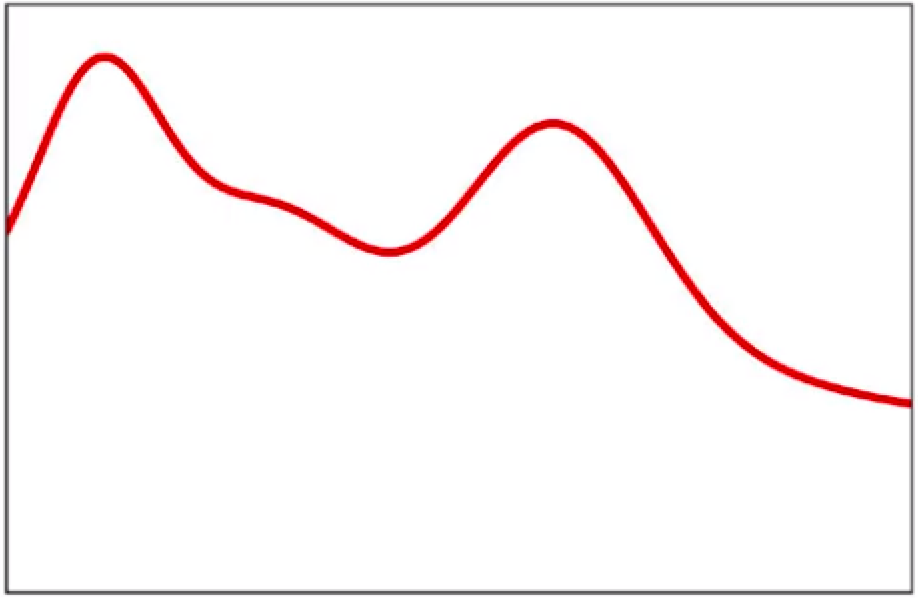
-- Universal Approximation Theorem, Wikipedia

Two caveats of “any function”:

1. “*approximation*” instead of “*exactly*”;
2. the *continuous* functions;

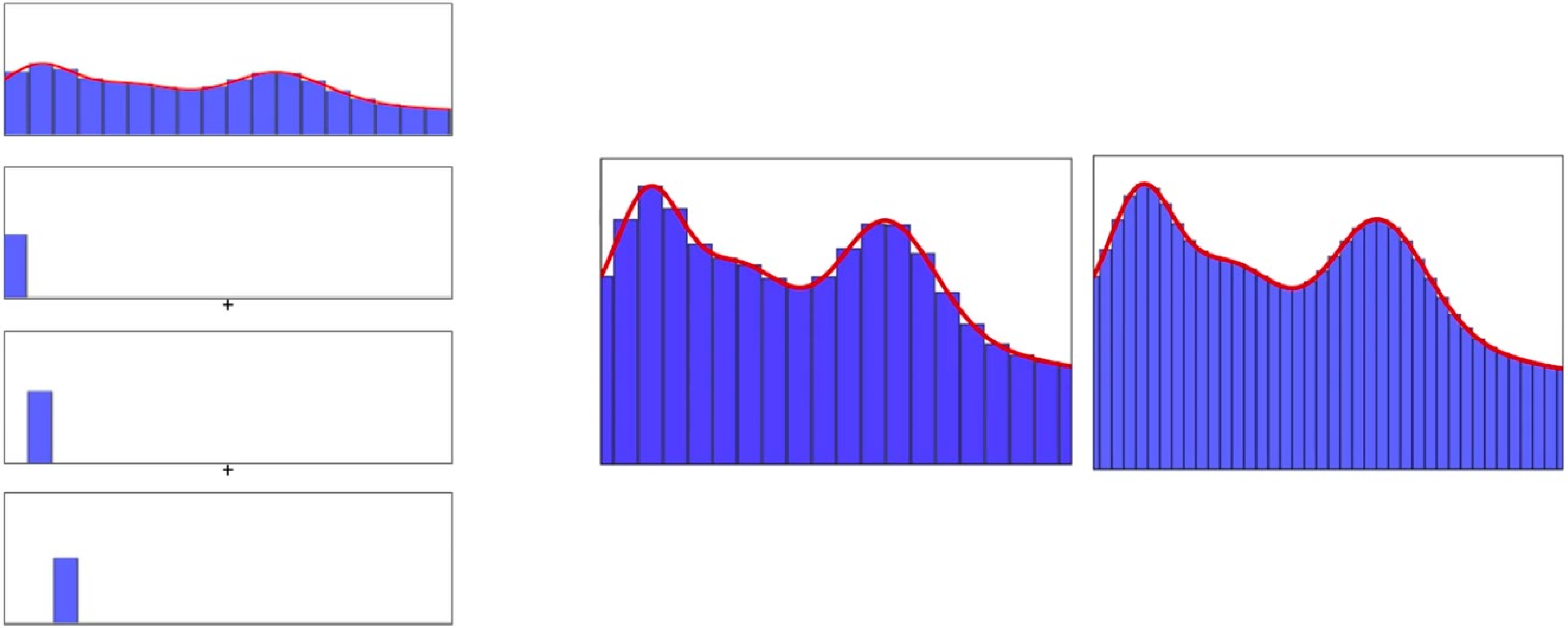
Lazy version:
*“A Neural Network can approximate **almost** any functions.”*

Universal approximation theorem

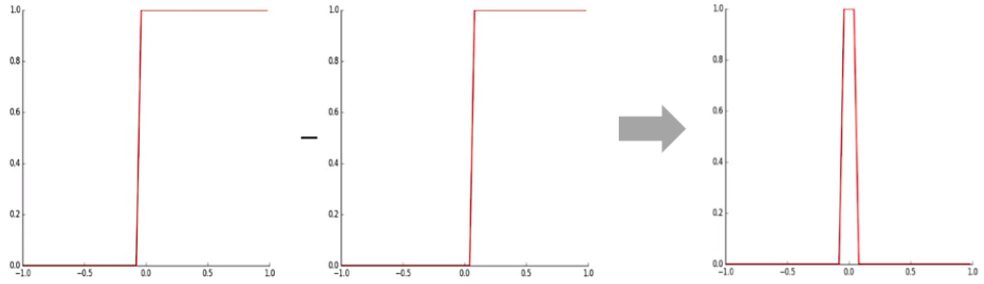


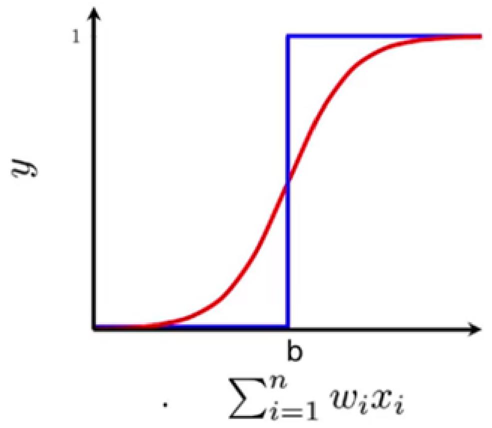
how to construct a neural network which approximates a function with just one input and one output

Universal approximation theorem

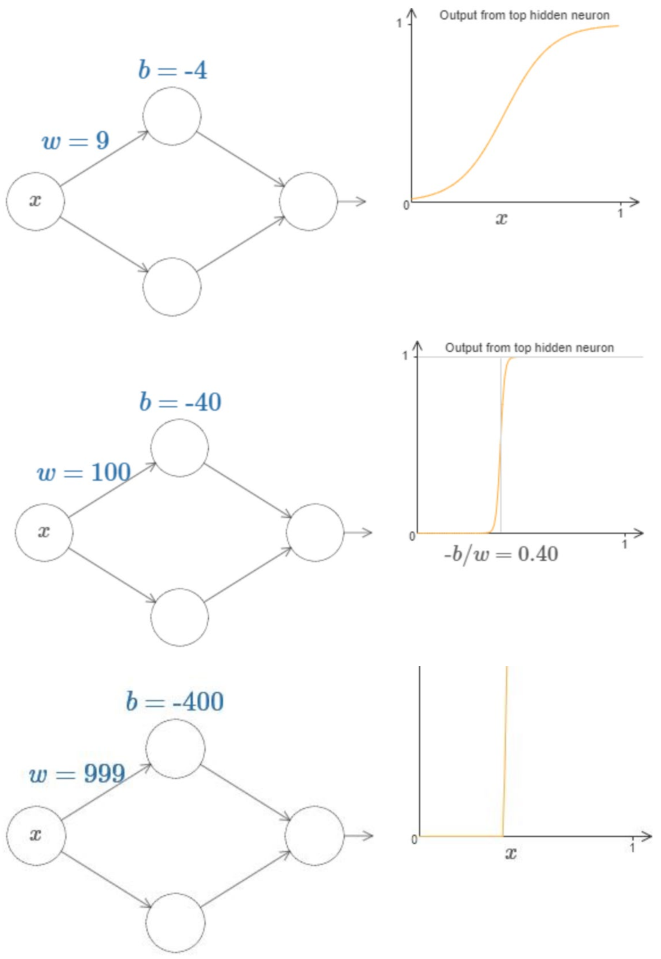


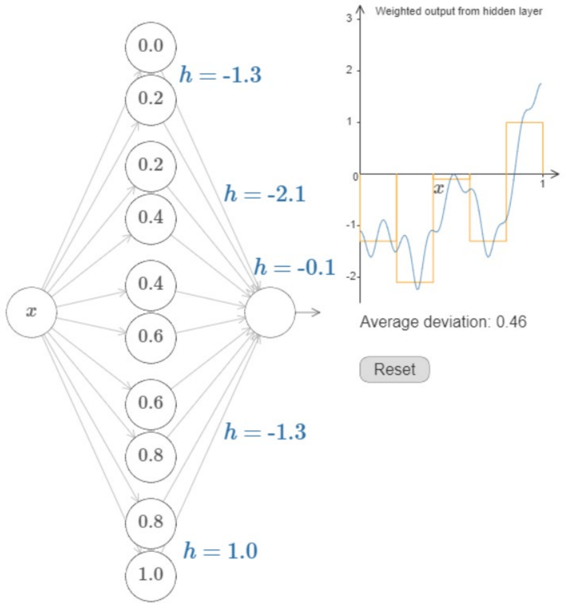
Universal approximation theorem



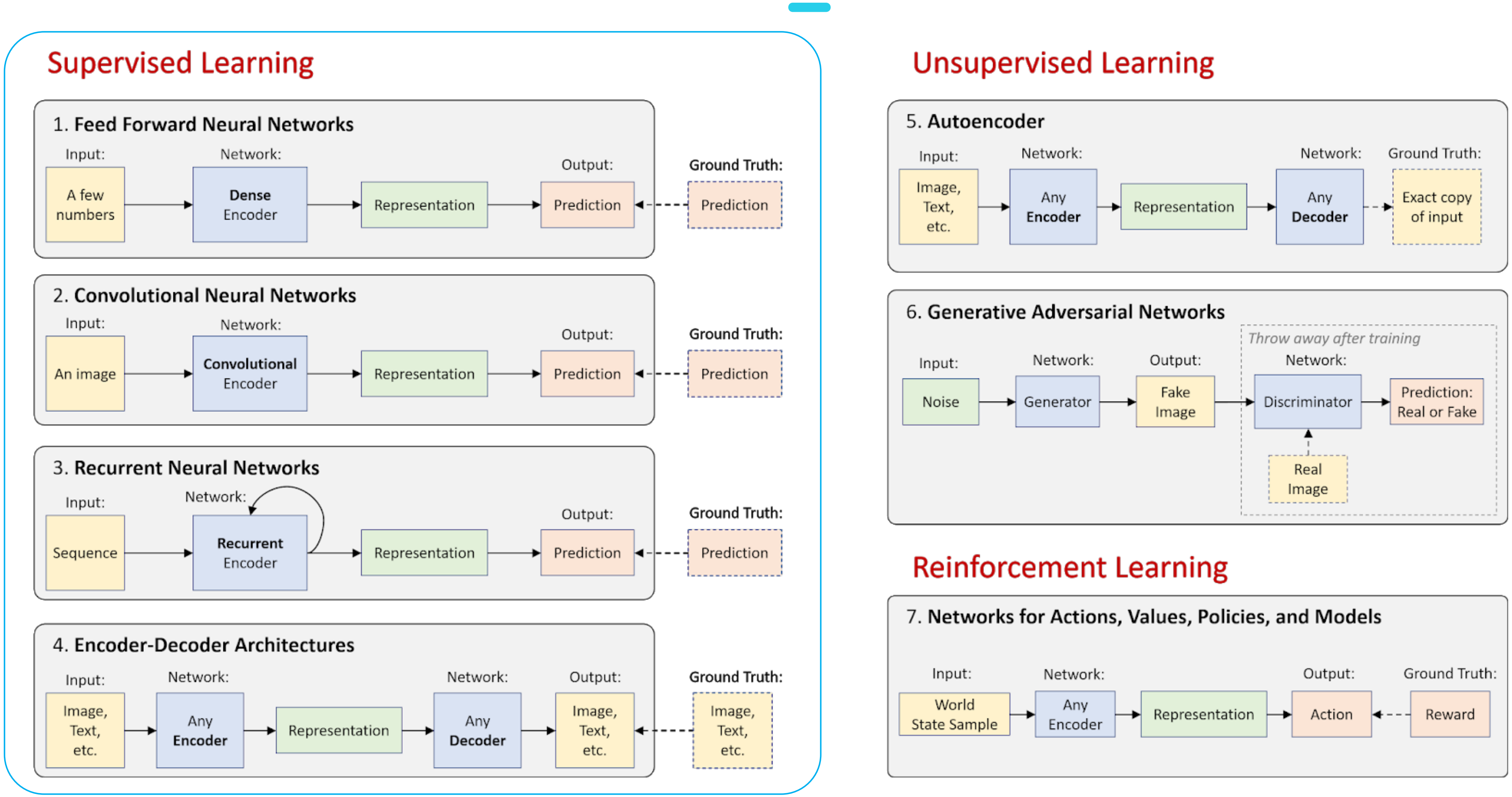


$$y = \frac{1}{1+e^{-(w^T x+b)}}$$

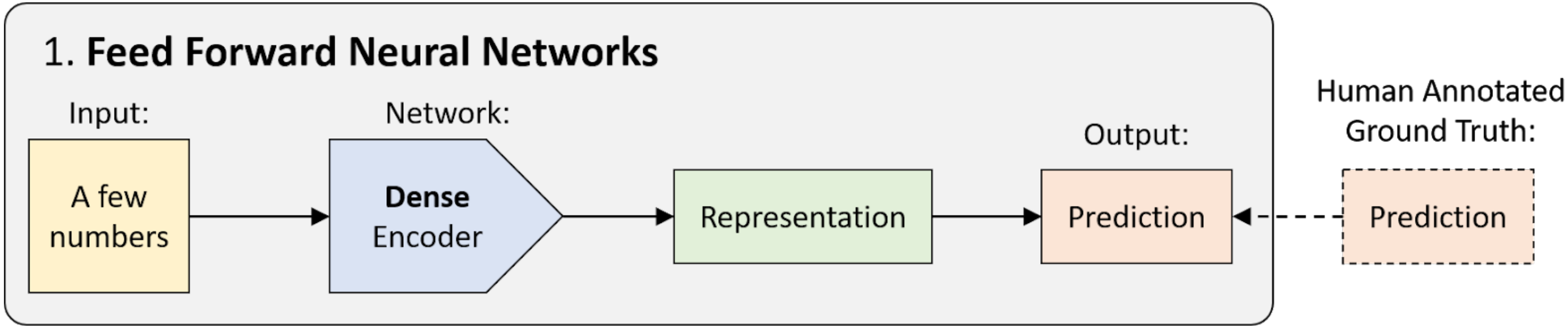




Common architecture of neural networks

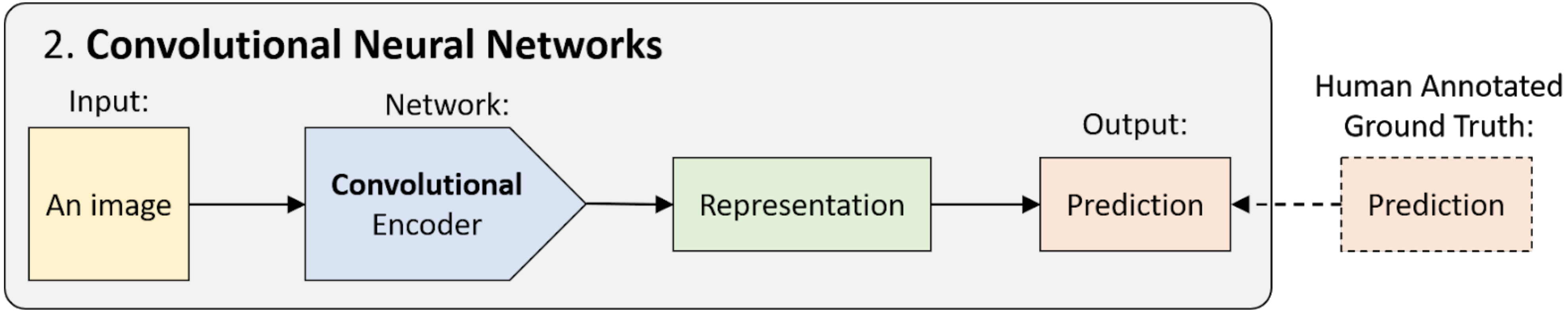


FFNNs



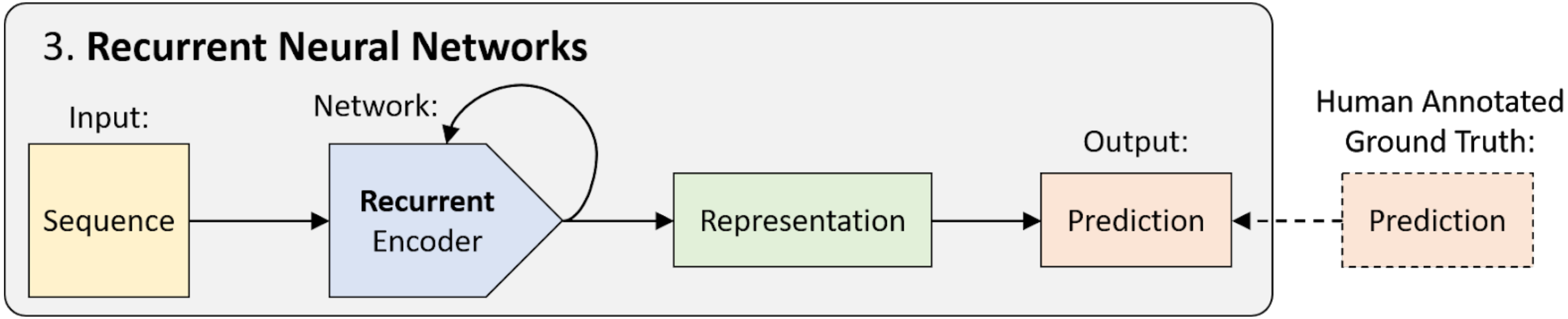
- dating back to 1940s;
- data passes from input to output in a **single pass** without any “state memory” of what came before.

CNNs



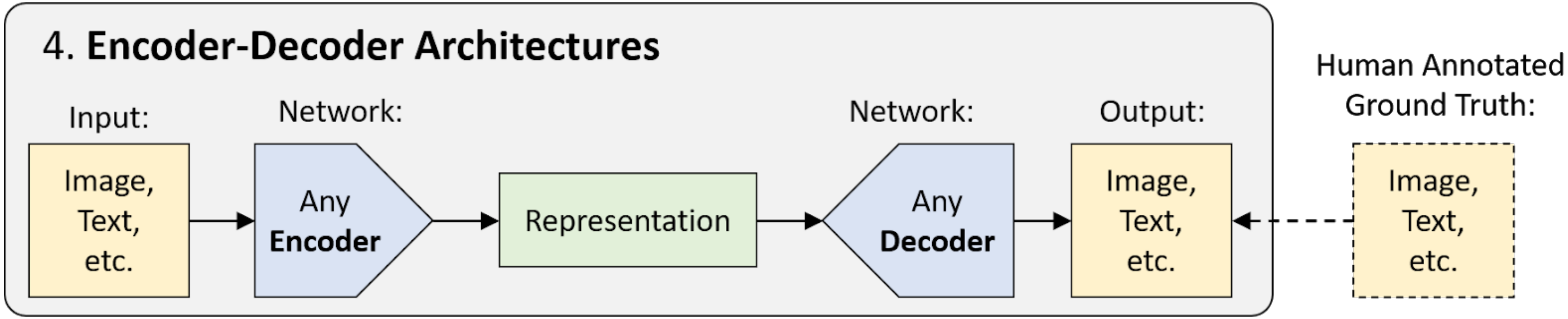
- Densely-connected layers + convolutional layers (convolutional encoder).
- Feed forward neural networks that use a **spatial-invariance** trick to efficiently learn **local patterns**; *(most commonly, in images)*

RNNs



- Have cycles and therefore have “state memory”;
- Can be unrolled in time to become feed forward networks where the weights are shared;
- CNN – weights shared across “space” v.s. RNN – weights shared across “time”; → sequential data

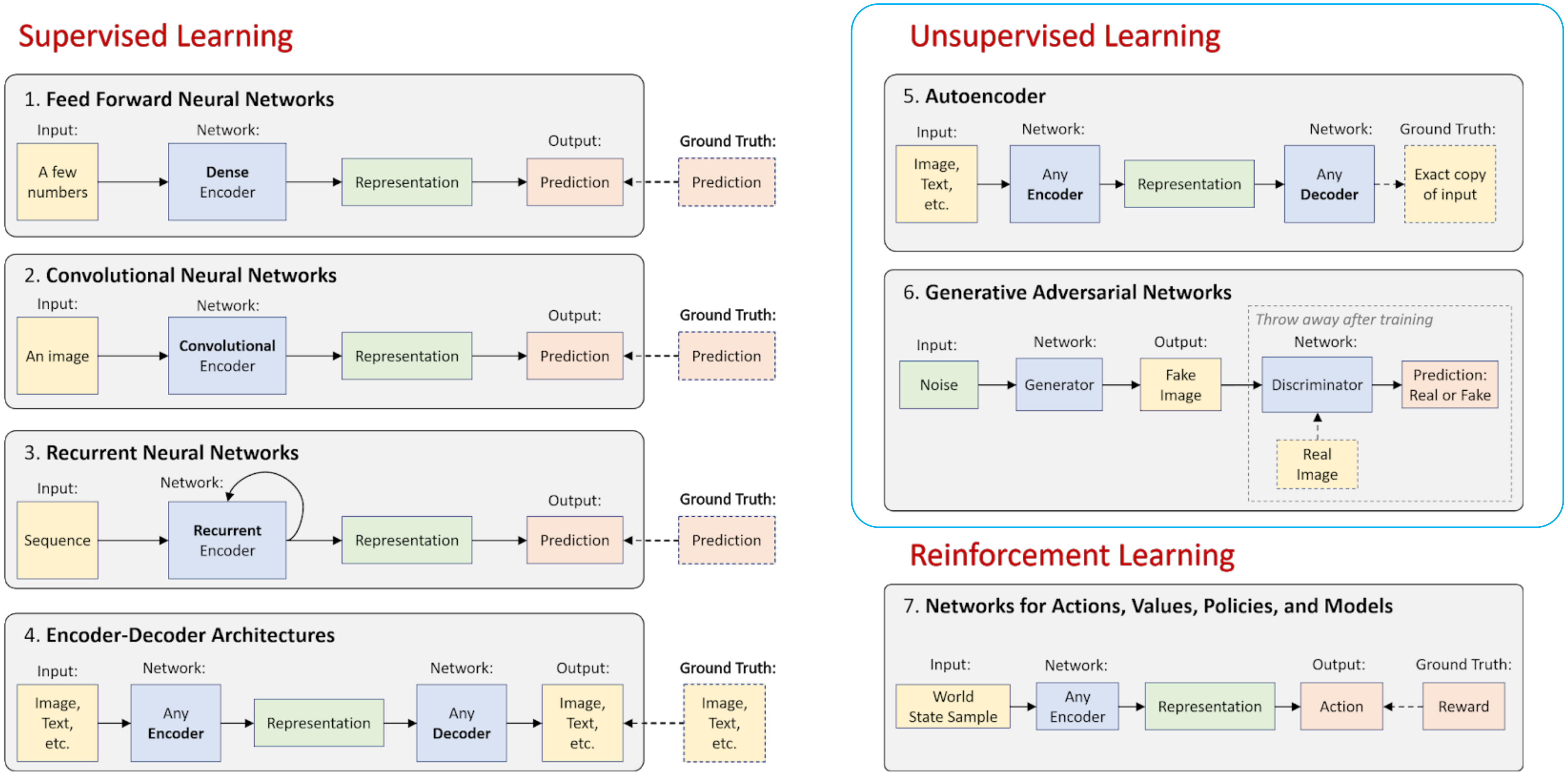
Encoder-Decoder Architectures



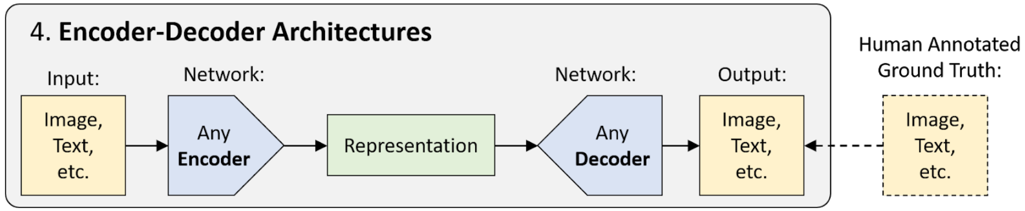
e.g. image caption: encoder-CNN, decoder-RNN;

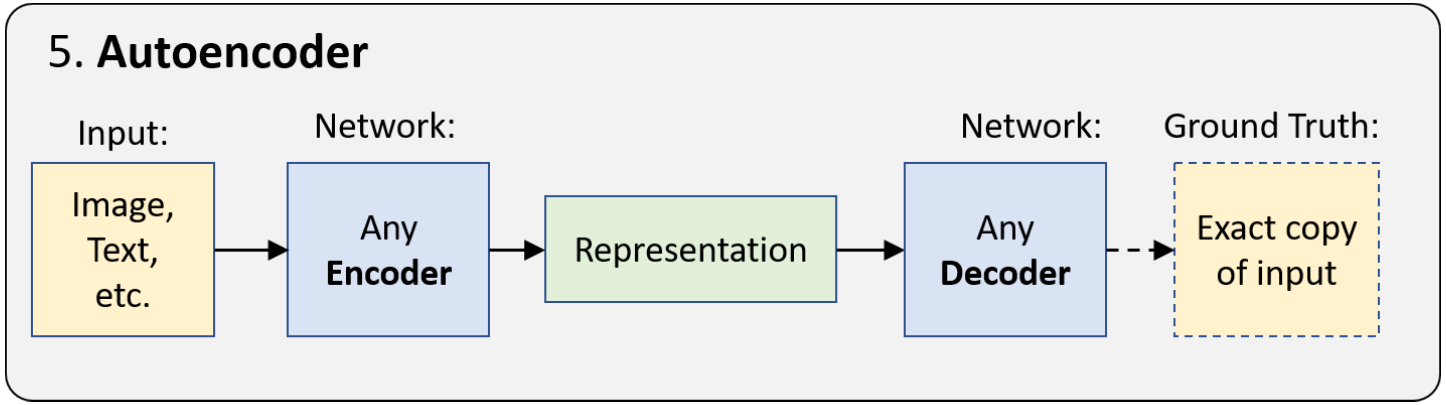
- FFNNs – dense encoder, CNNs – convolutional encoder, RNNs – recurrent encoder;
- Encoder: find patterns in raw data to form compact, useful representations;
- Decoder: generate high-resolution data from those representations.

Common architecture of neural networks



Autoencoders

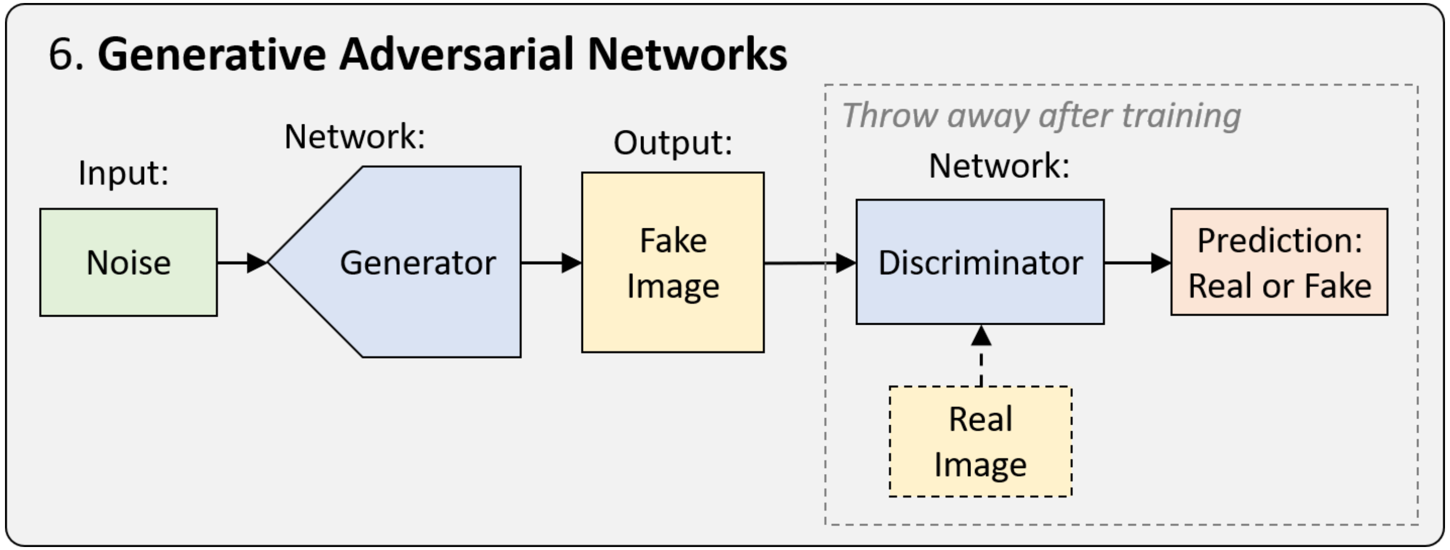




- **self-supervised**: the ground truth data comes from the input data, no human effort is required;
- Application: unsupervised embeddings, image denoising, etc.

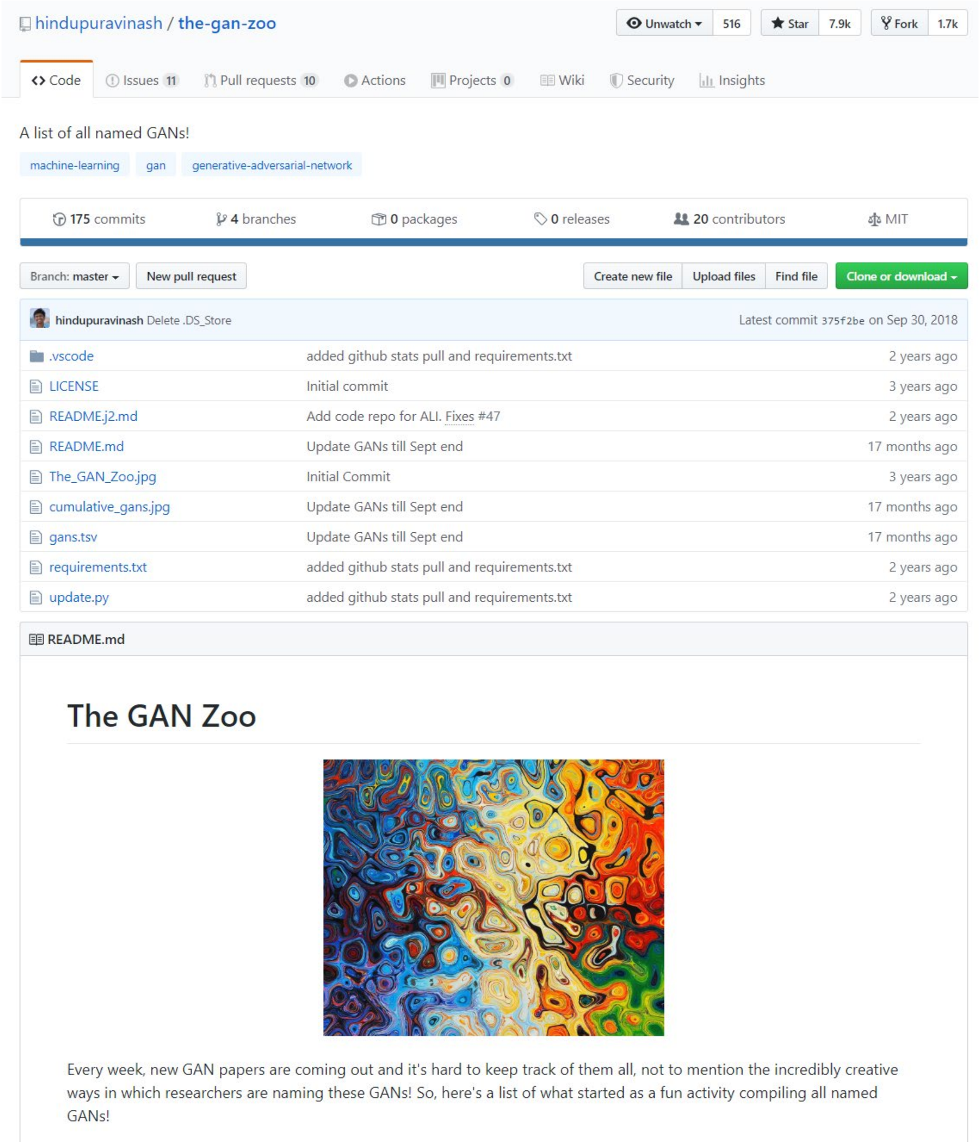
Generative Adversarial Networks (GANs)

a framework for training networks optimized for generating new realistic samples from a particular representation.



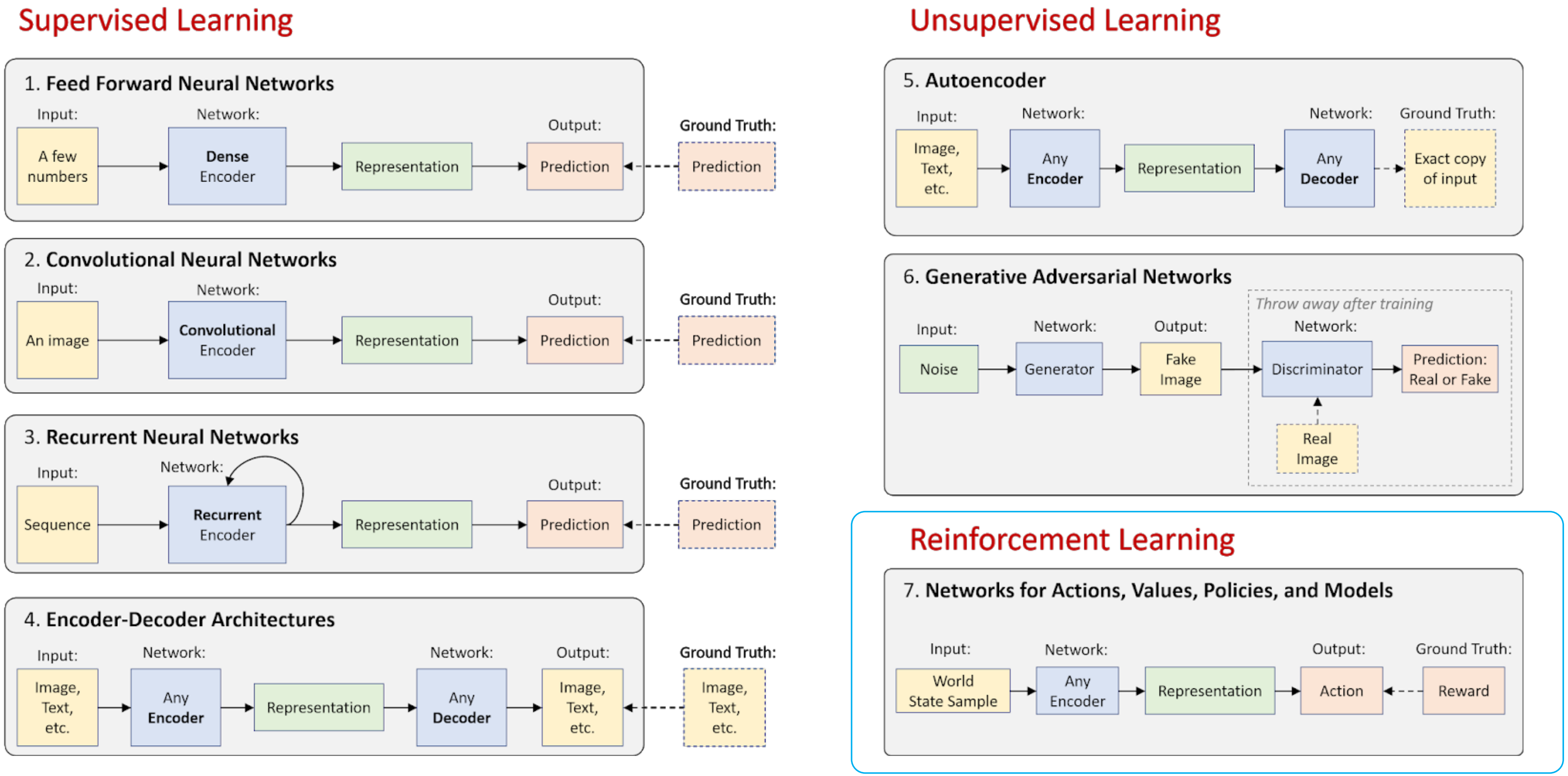
images generated by BigGAN.

The GAN Zoo

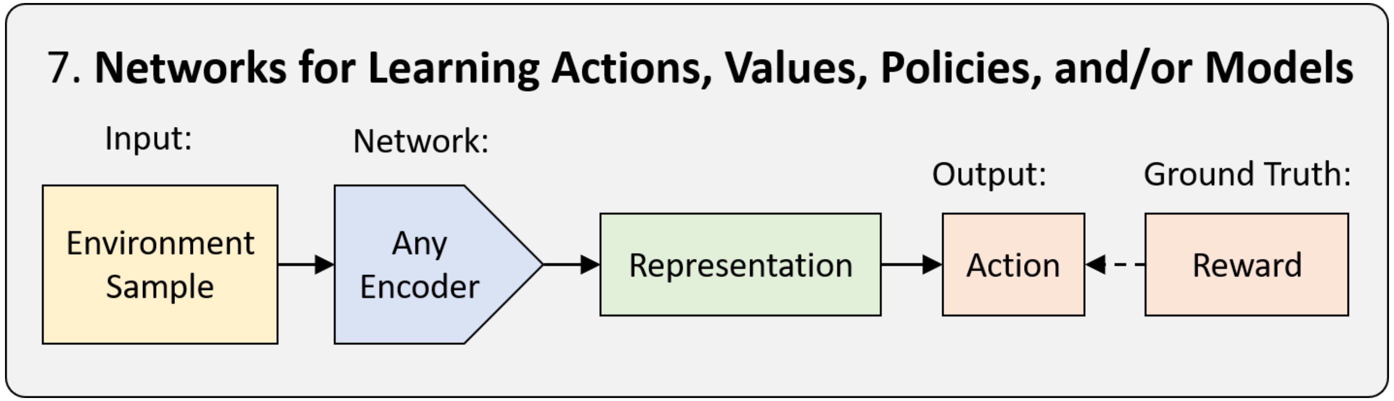


- 500+ different named GAN variations.

Common architecture of neural networks

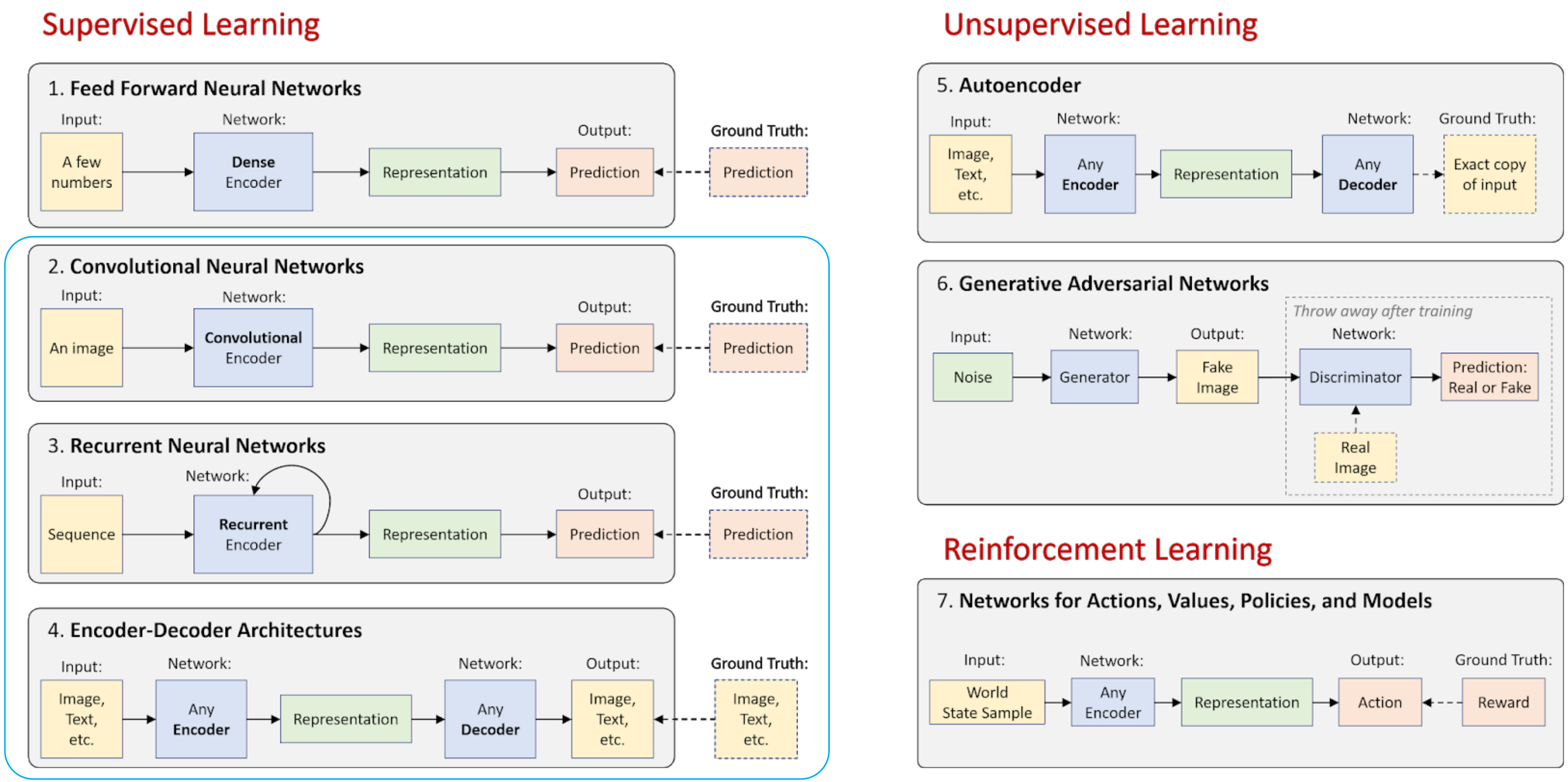


Deep Reinforcement Learning (Deep RL)

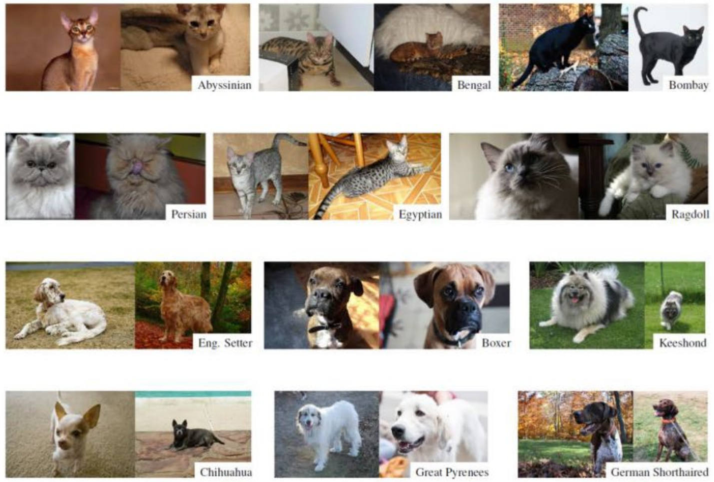


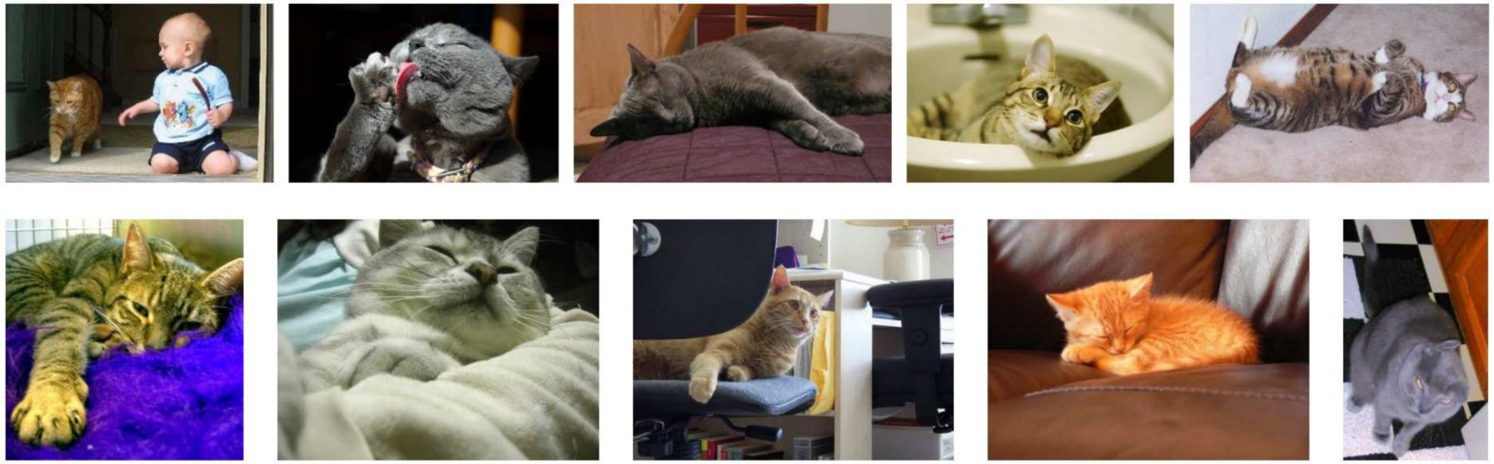
- Based on what the NN is tasked with learning:
  policy-based, **value-based**, and **model-based**;

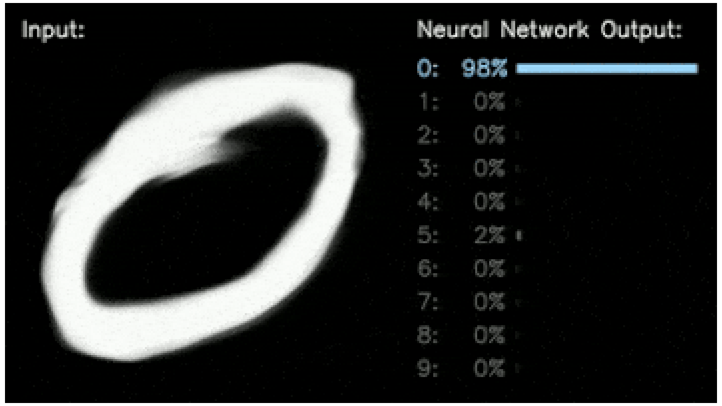
Selected NNs in Detail



Pure Perception is HARD

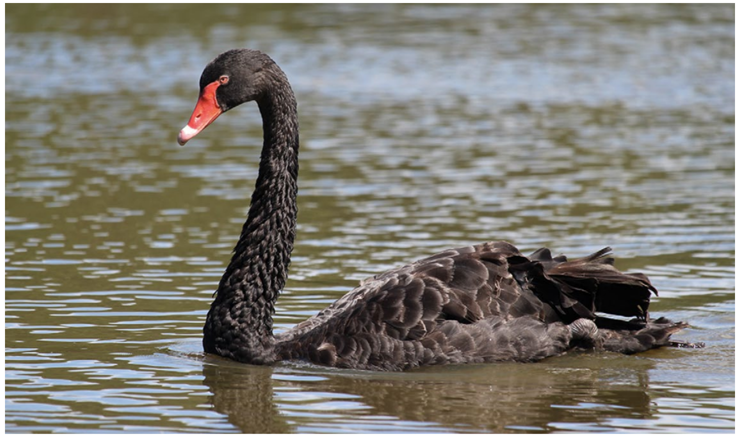


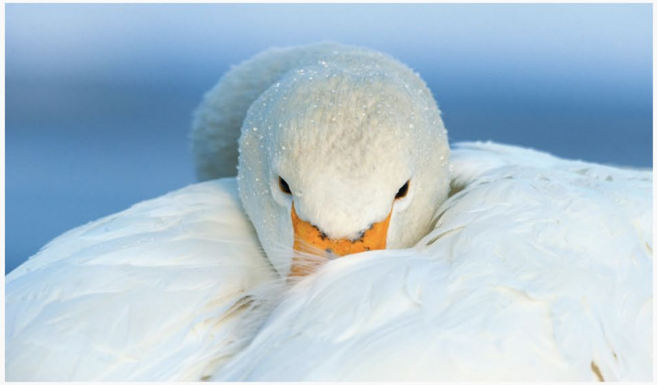




CNN is a revolutionary tool in the Computer Vision field.

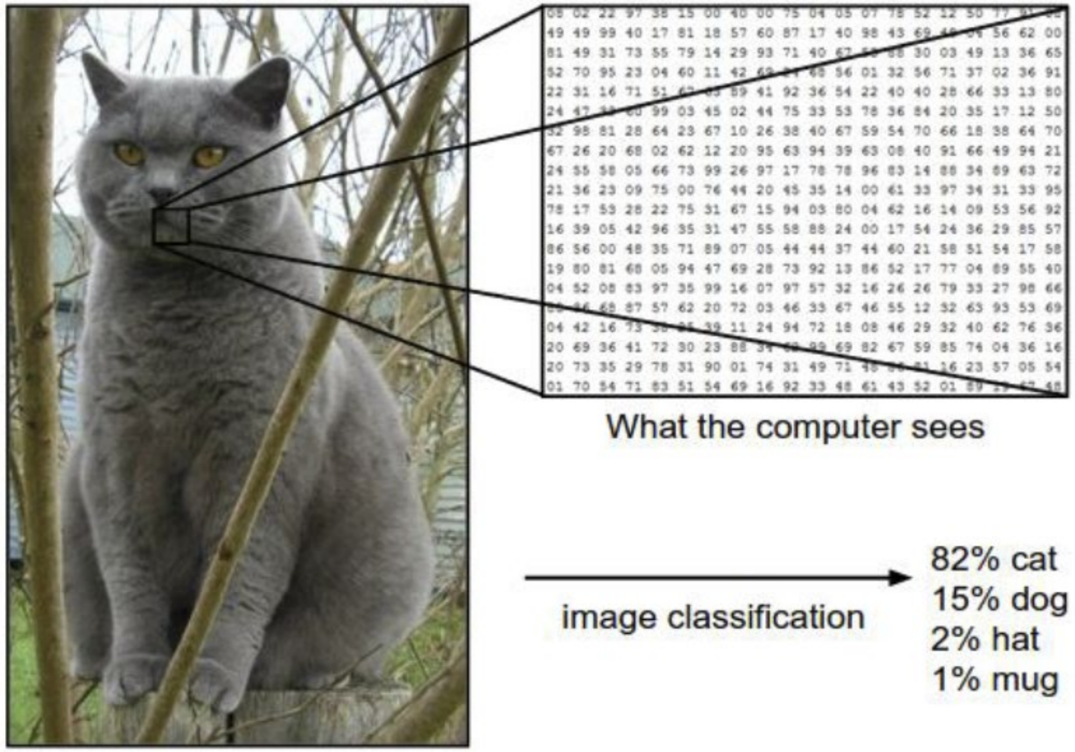
Image Understanding is HARD



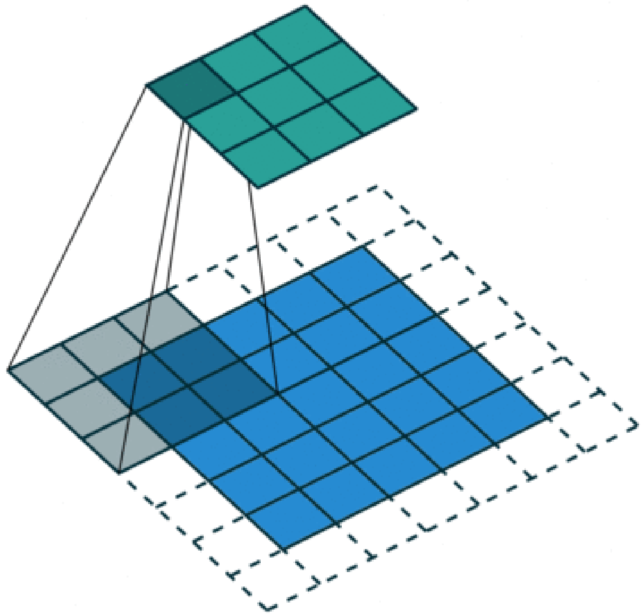


Man in swan tent photographing swans

CNN



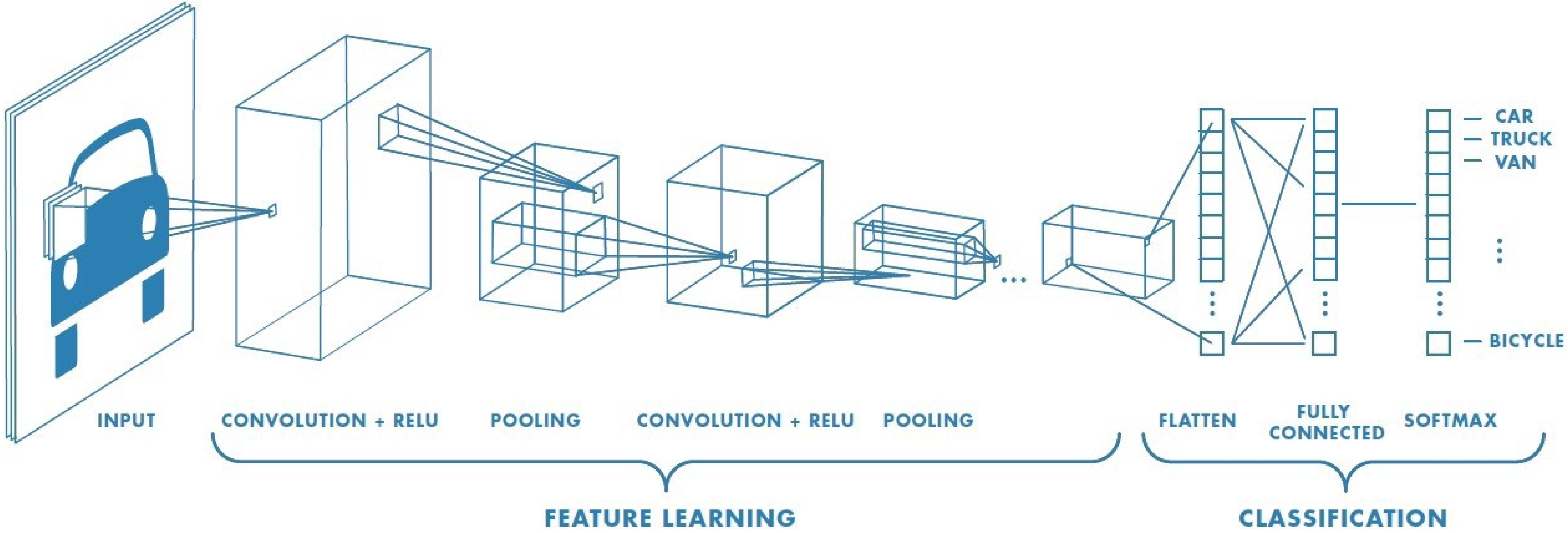
Images are Numbers



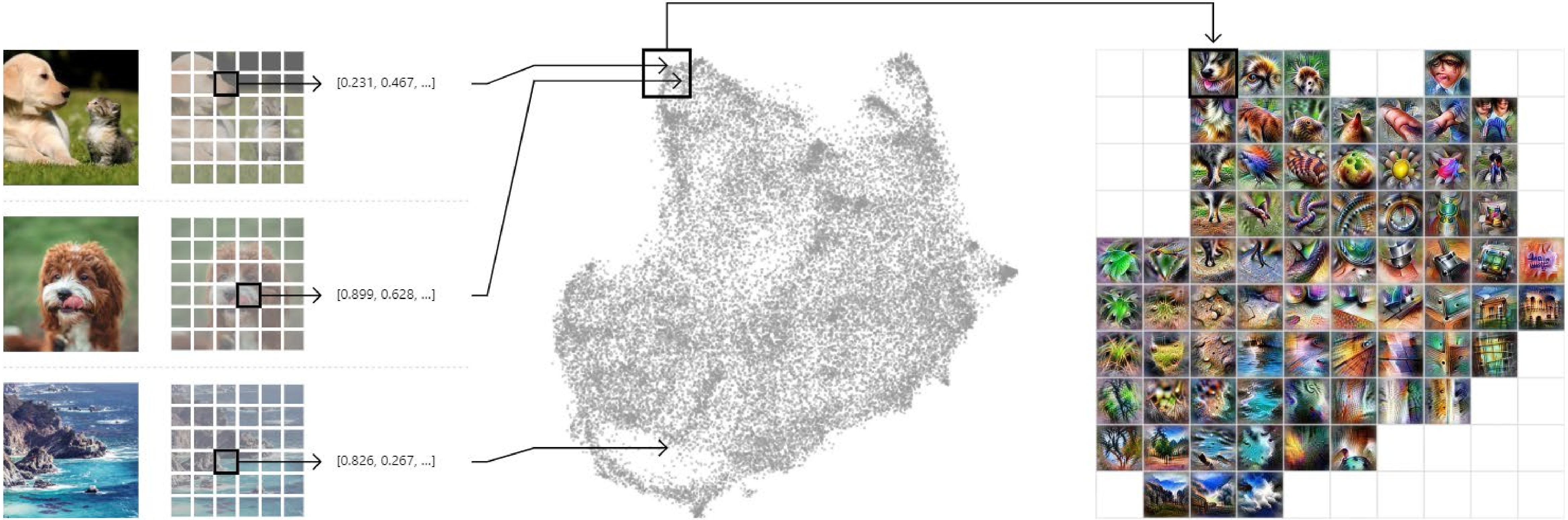
Convolutional filters:

take advantage of spatial invariance;

CNN


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

why CNN works



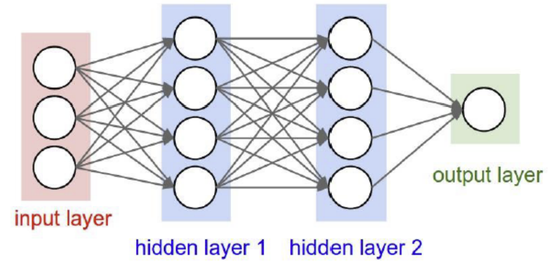
“Exploring Neural Networks with Activation Atlases”, distill, 2019

https://distill.pub/2019/activation-atlas/

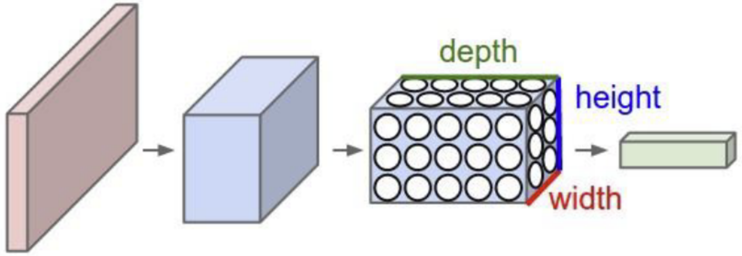
Why CNN over FFNN?

Convolutional Neural Networks

Regular neural network (fully connected):



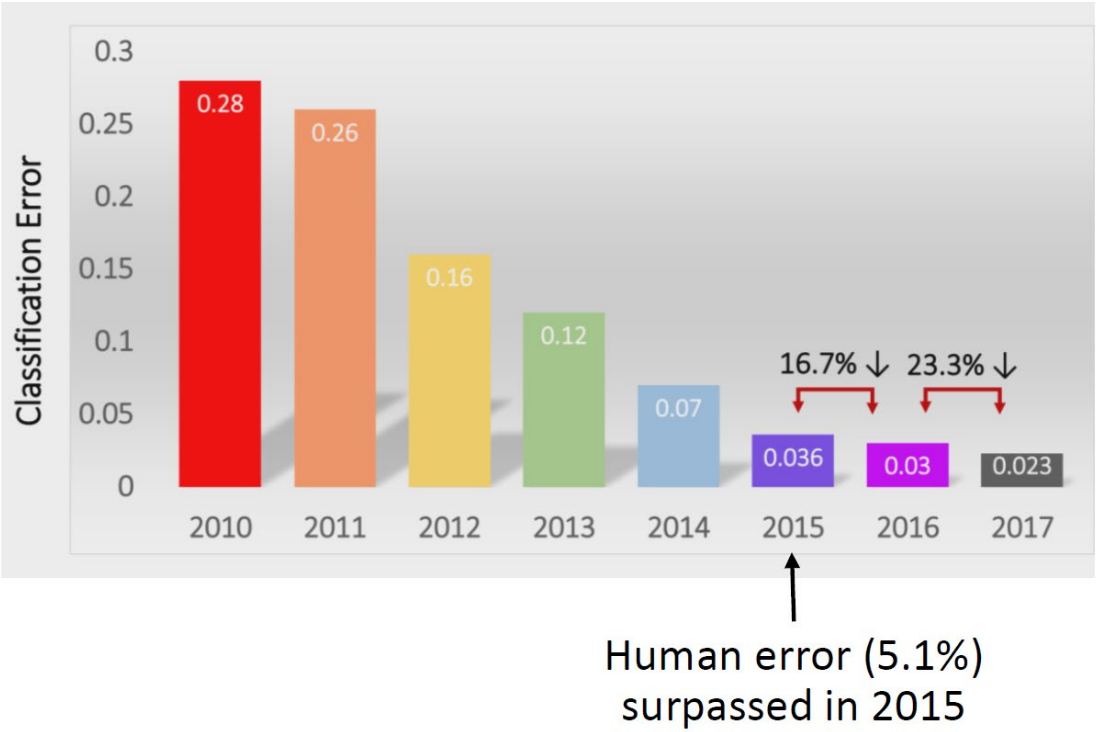
Convolutional neural network:



Each layer takes a 3d volume, produces 3d volume with some smooth function that may or may not have parameters.

- Spatial variant v.s. Spatial invariant;
- Scale ill v.s. Scale well;

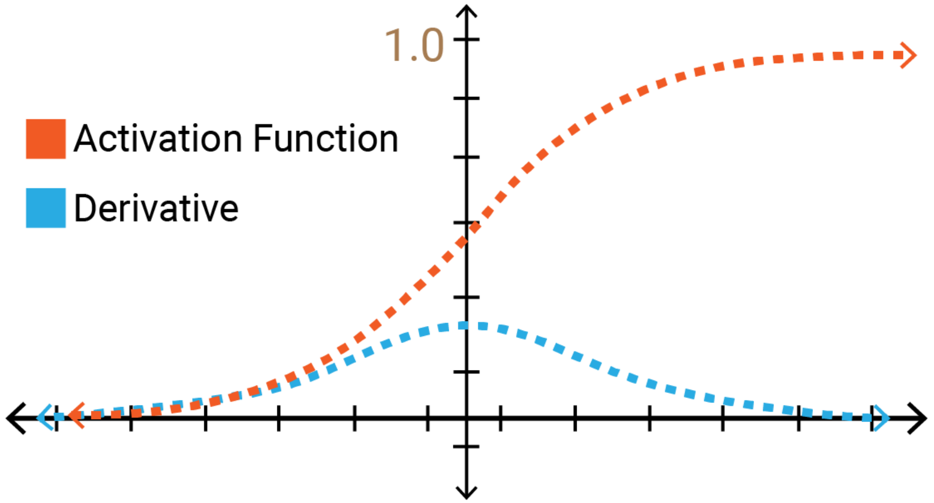
Classification: CNNs beat Human

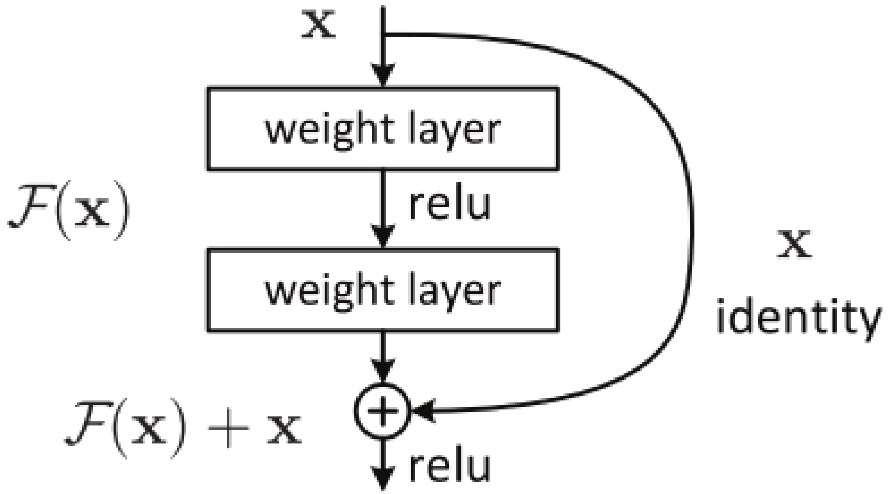


- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
  - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

ResNet (residual network)

- Is deeper the better? Vanishing Gradient!





The Residual Block

AlexNet:	8 layers
VGGNet:	16 layers
GoogLeNet:	22 layers

ResNet: 152 layers!

ResNet

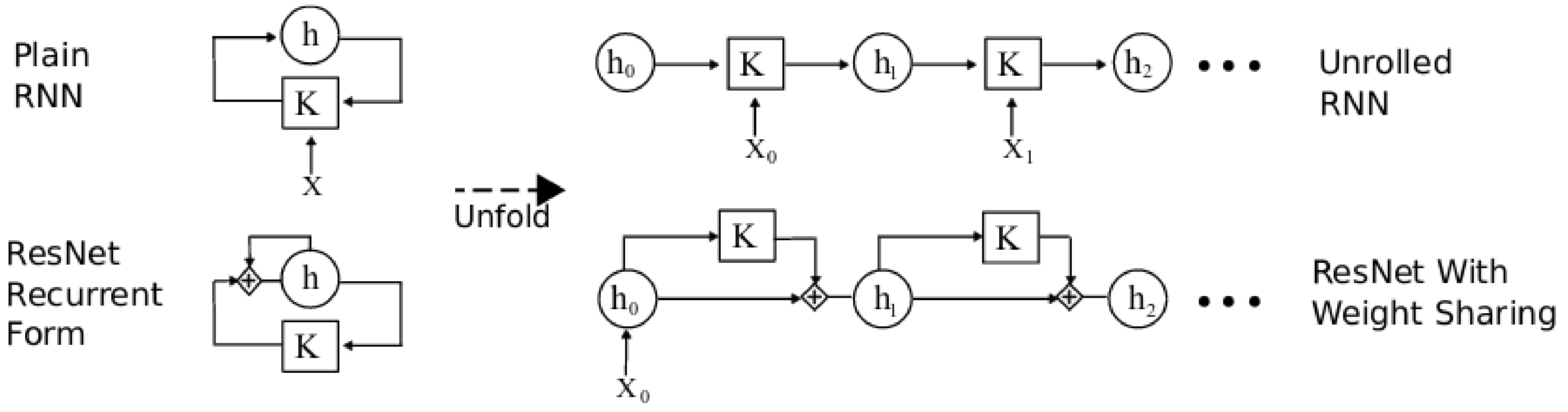
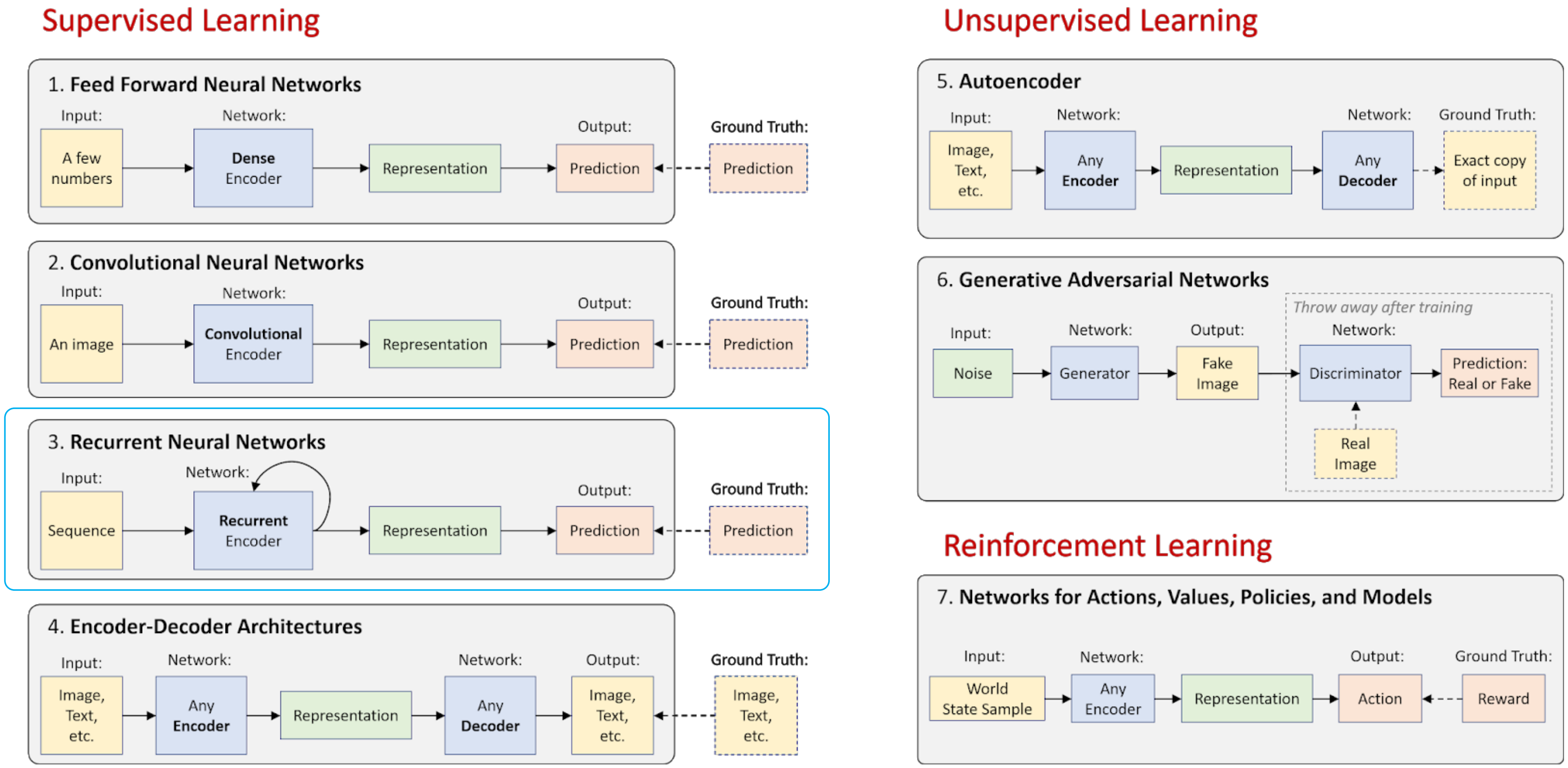


Figure 11: A ResNet can be reformulated into a recurrent form that is almost identical to a conventional RNN.

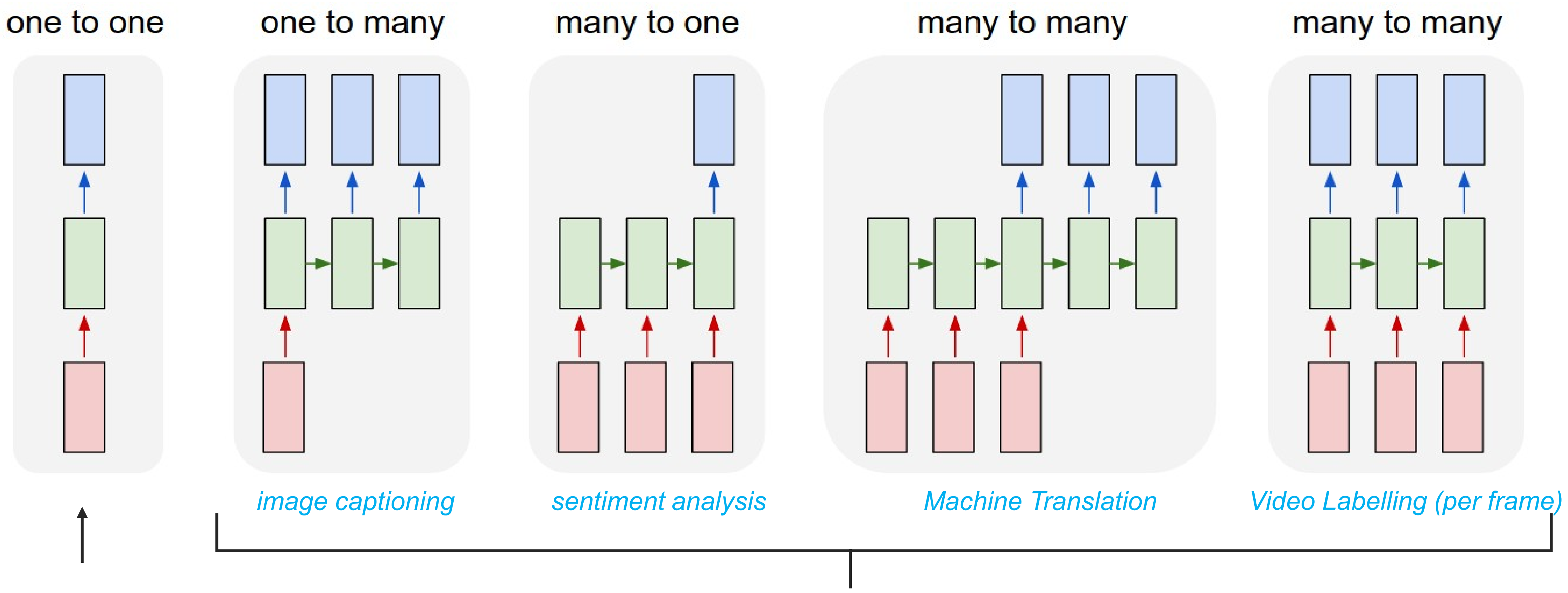
"Bridging the gaps between residual learning, recurrent neural networks and visual cortex.“ (2016)

- RNNs without the explicit time based construction;

Selected NNs in Detail



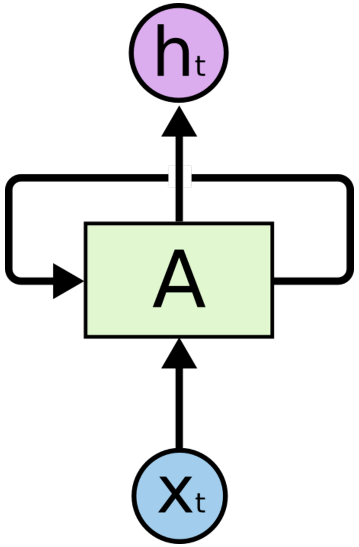
RNN (LSTM)



Vanilla NN

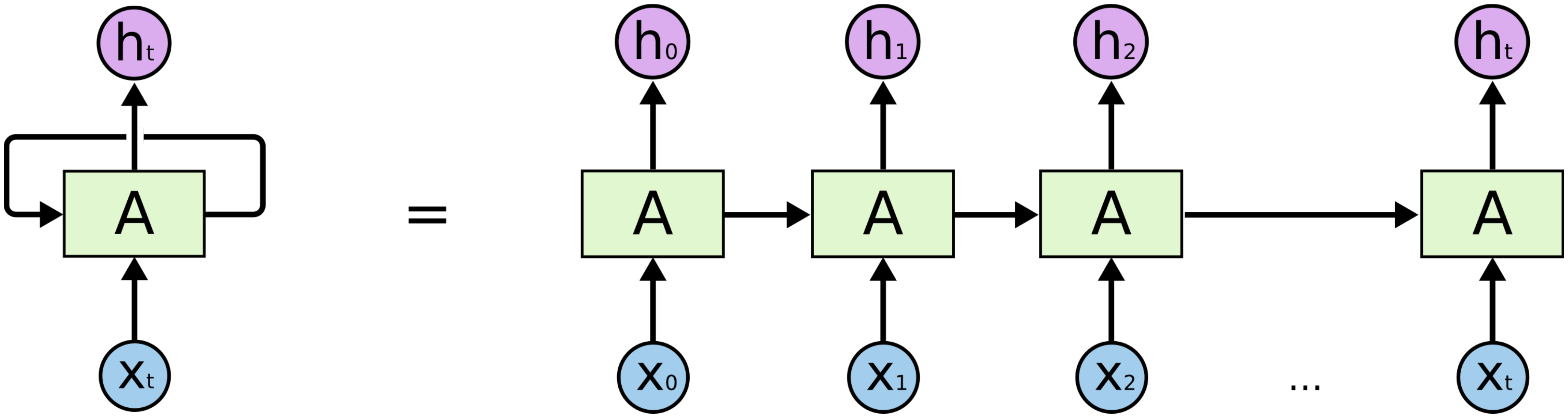
RNNs are amazing. But tricky to train.

RNN(LSTM)



Recurrent Neural Networks have loops.

RNN(LSTM)



An unrolled recurrent neural network.

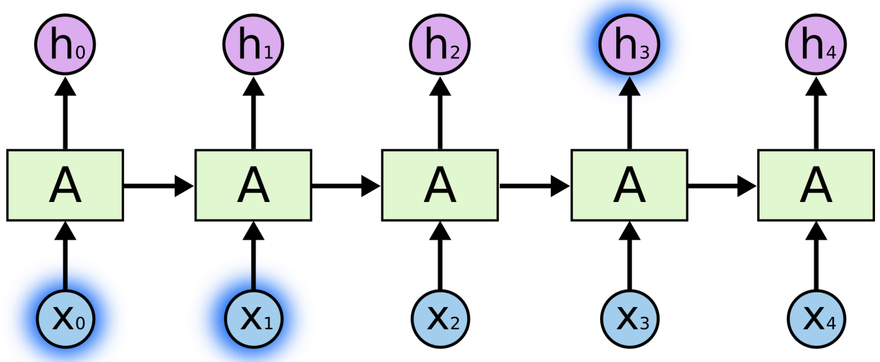
Input: (example: word of a sentence)
Hidden state: function of previous hidden state and new input
Output: (example: predict next word in the sentence)

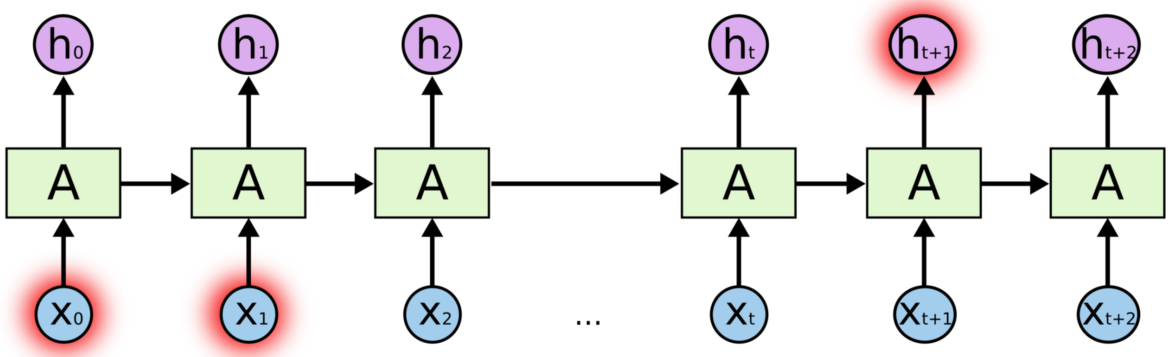
Long-Term Dependency

Context

"Bob is eating an **apple**."

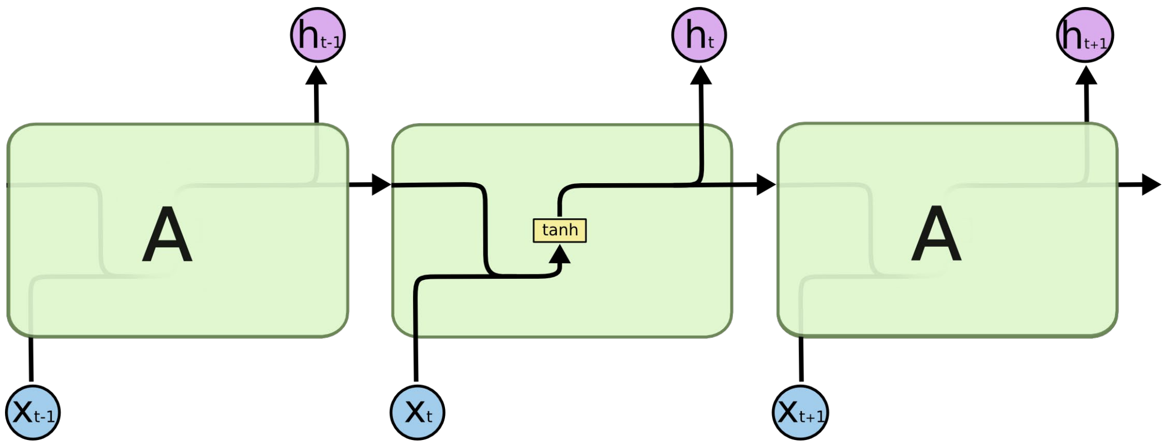
"*Bob likes apples.* He is hungry and decided to have a snack. So now he is eating an **apple**."

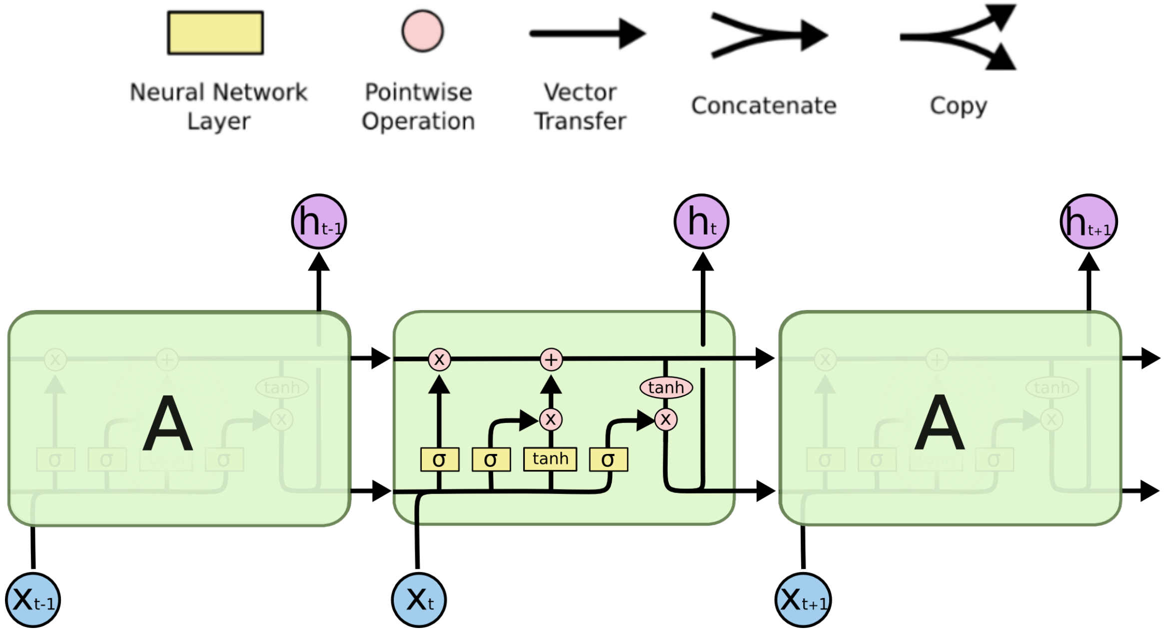




In theory, RNNs could learn this long-term dependencies.
In practice, it is difficult.

RNN v.s. Long short-term memory (LSTM)

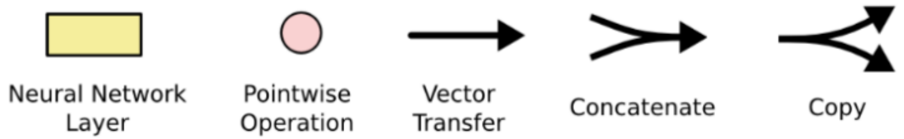


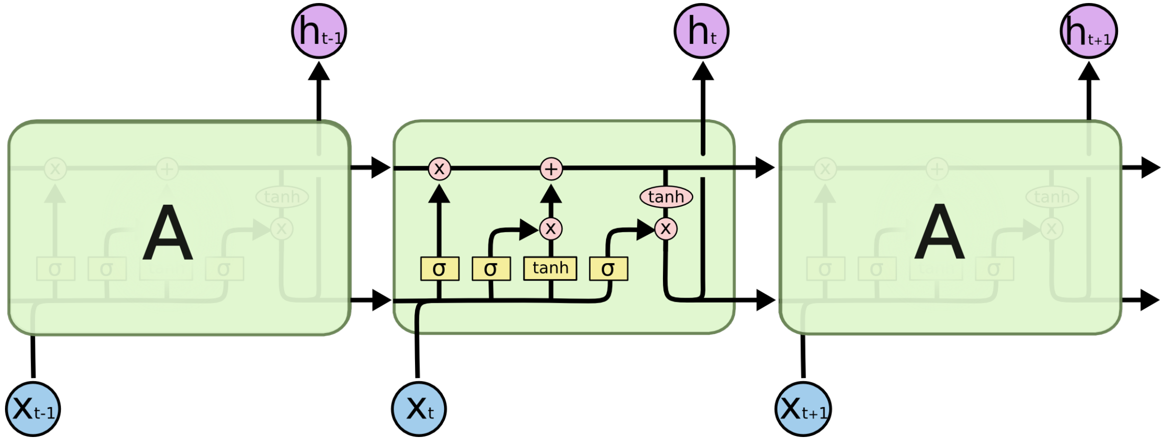


The repeating module in a standard RNN contains a single layer.

The repeating module in a standard LSTM contains four interacting layers.

LSTM: Pick What to Forget and What To Remember





Conveyer belt for **previous state** and **new data**:

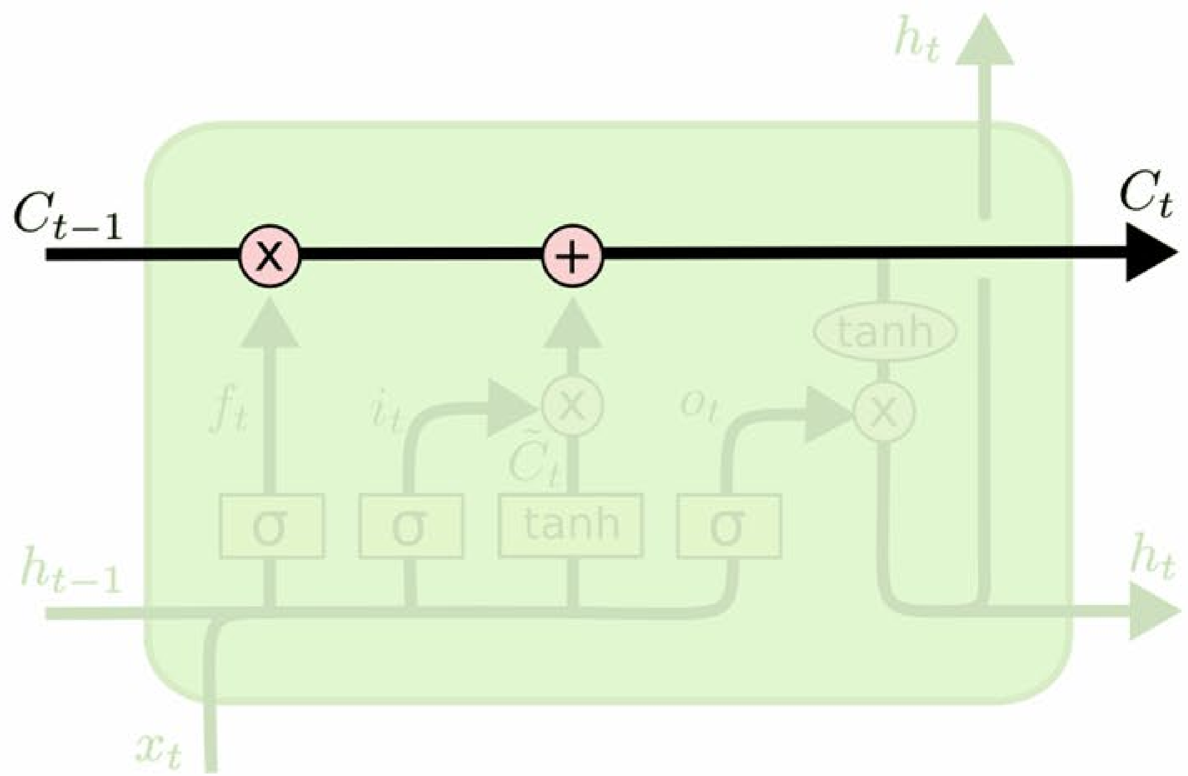
1. Decide what to forget (state)

2. Decide what to remember (state)

3. Decide what to output (if anything)

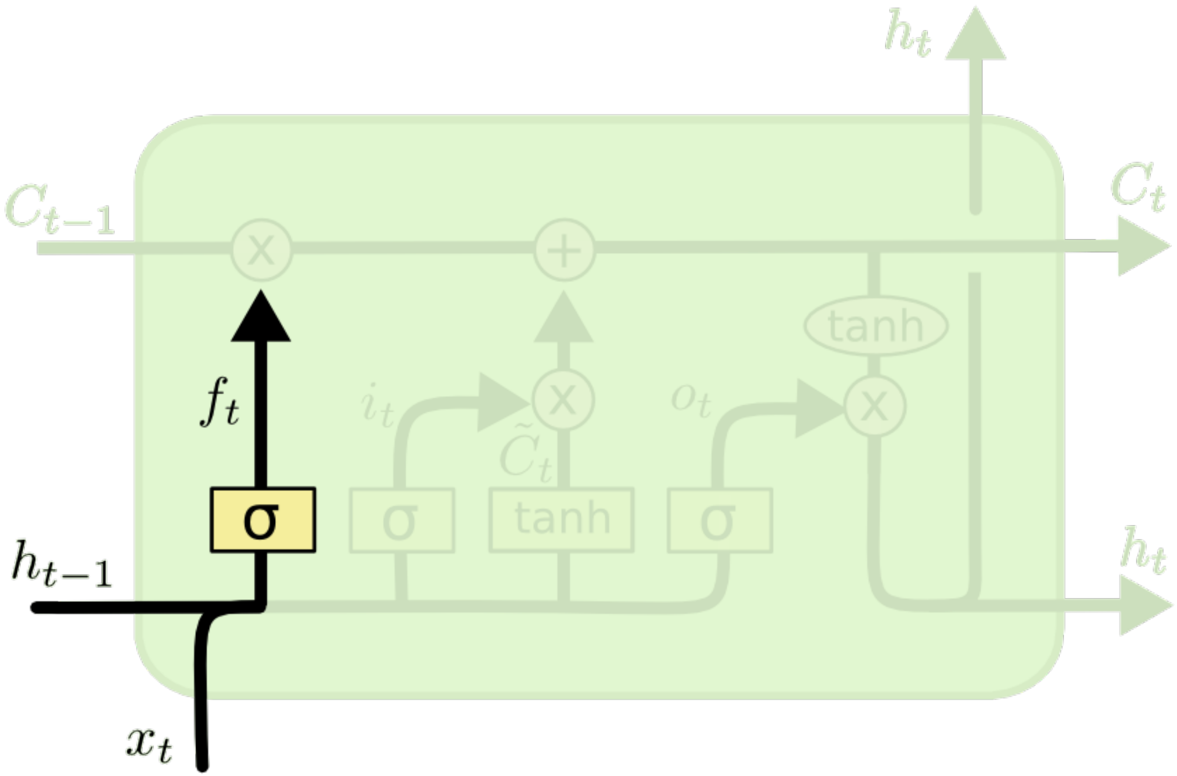
Bob and Alice are having lunch. Bob likes apples. Alice likes oranges. **She** is eating an **orange.**

LSTM Conveyer Belt



- State run through the cell
- 3 sigmoid layers output deciding which information is let through (~1) and which is not (~0)

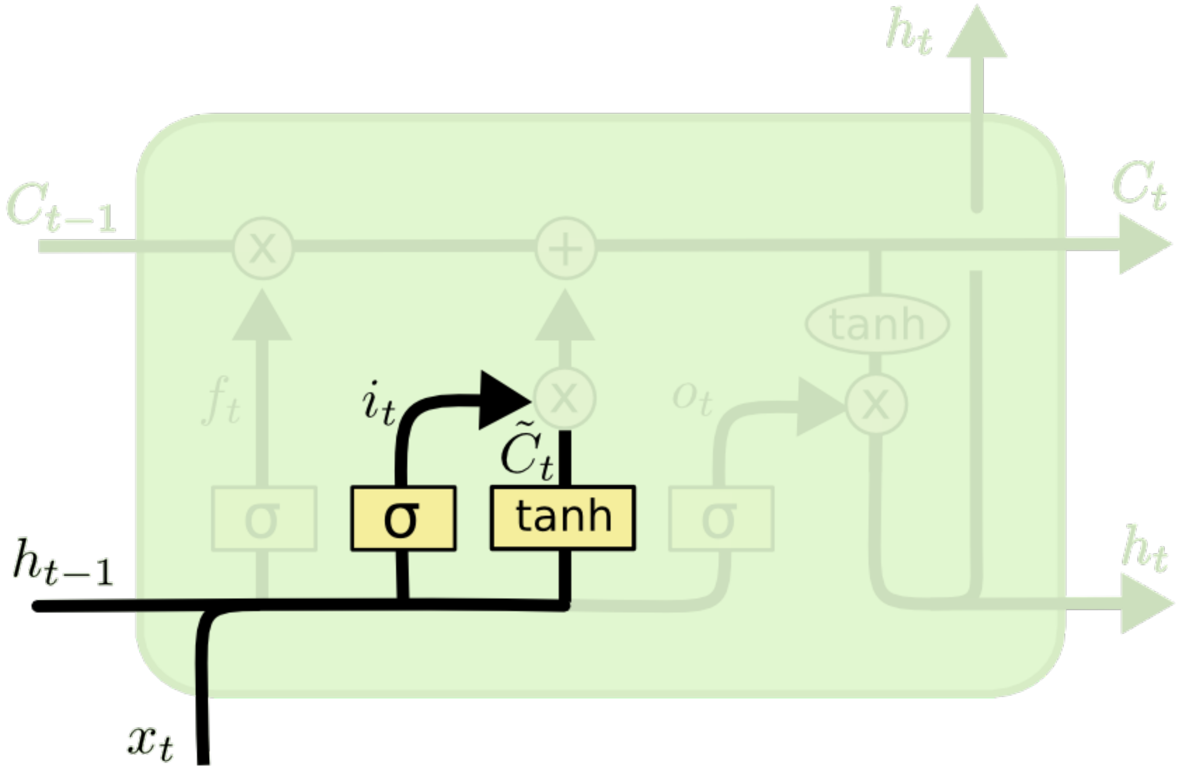
LSTM Conveyer Belt



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

Step 1: Decide what to forget / ignore

LSTM Conveyer Belt

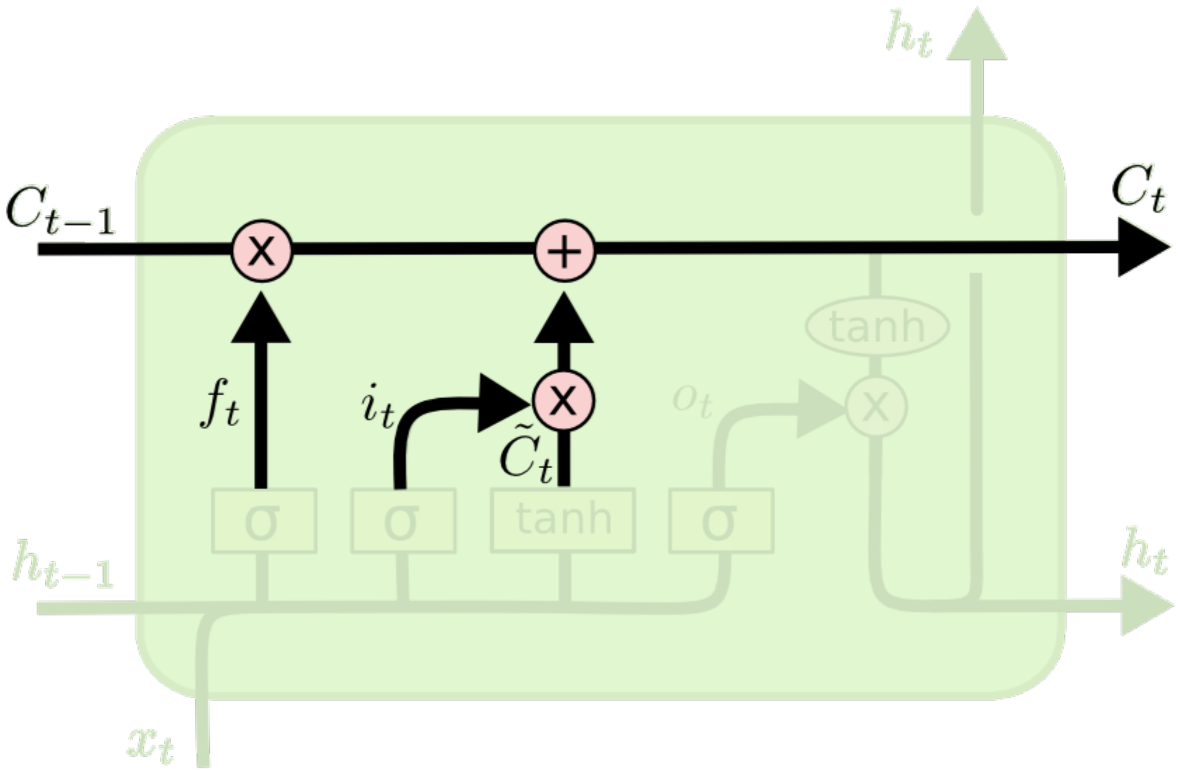


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

Step 2: Decide which state values to update (w/sigmoid) and what values to update with (w/ tanh)

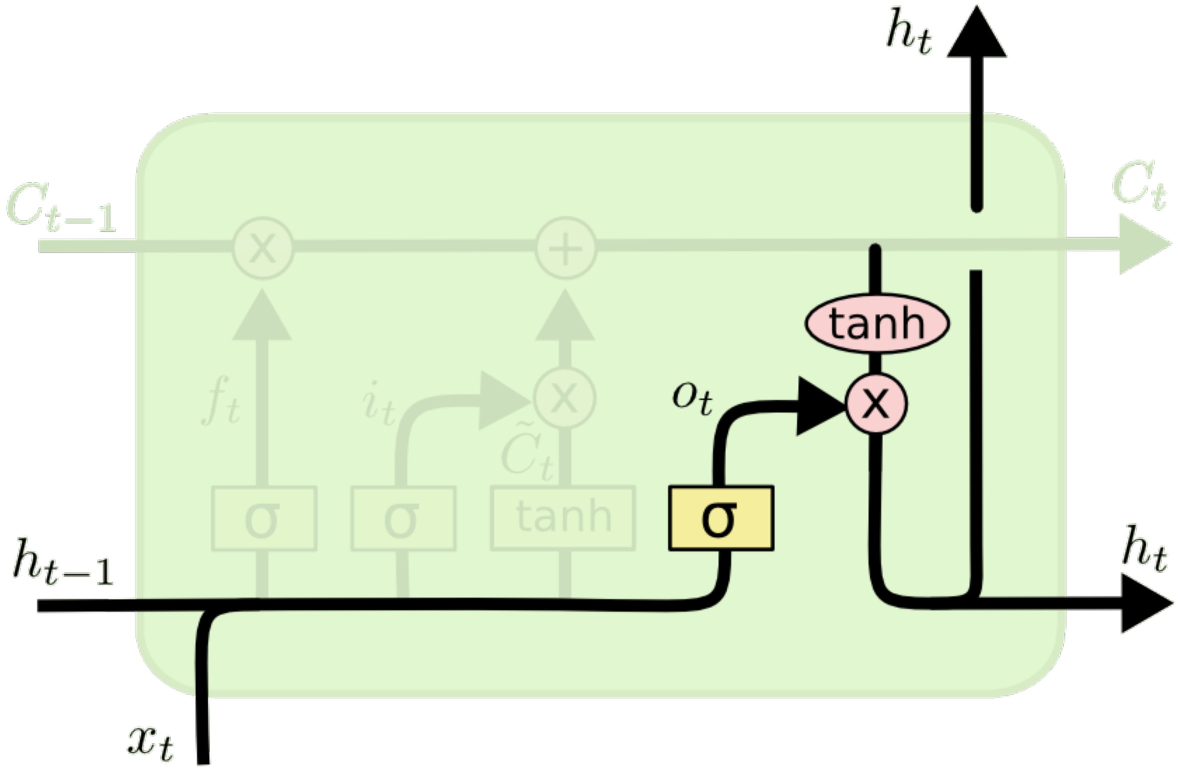
LSTM Conveyer Belt



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

Step 3: Perform the forgetting and the state update

LSTM Conveyer Belt

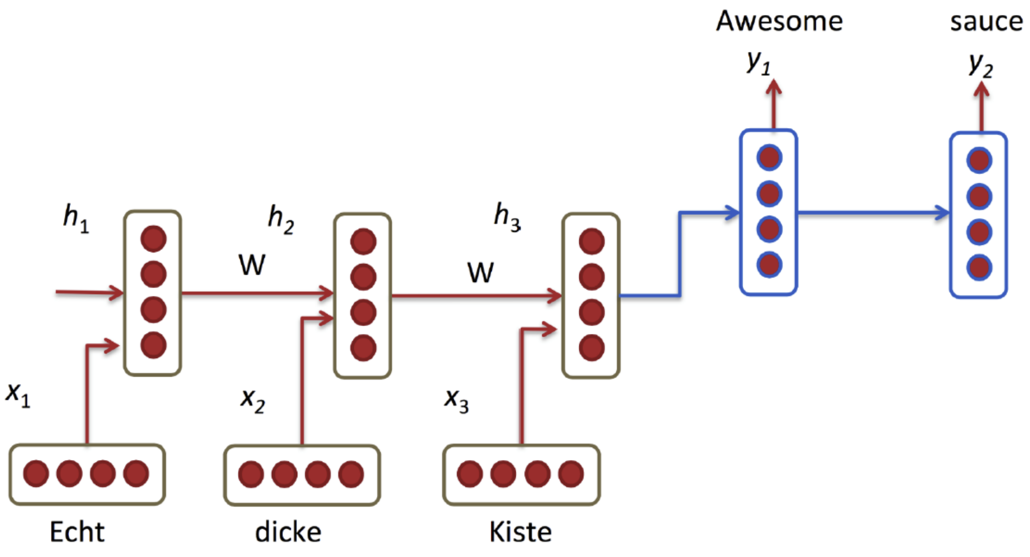


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

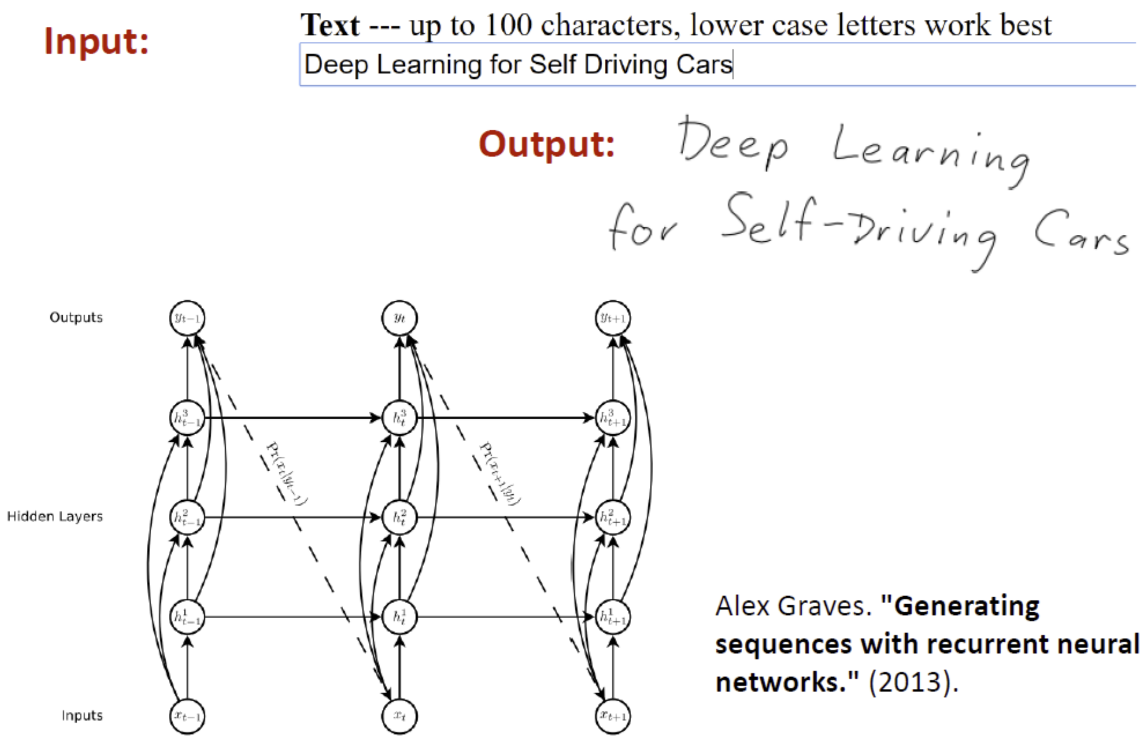
$$h_t = o_t * \tanh\left(C_t\right)$$

Step 4: Produce output with tanh [-1, 1] deciding the values and sigmoid [0, 1] deciding the filtering

Applications



Machine Translation

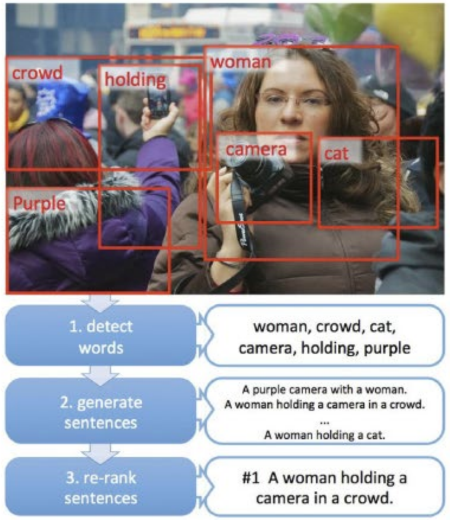


Alex Graves. "**Generating sequences with recurrent neural networks.**" (2013).

Handwriting Generation from Text

Applications

a man sitting on a couch with a dog
a man sitting on a chair with a dog in his lap



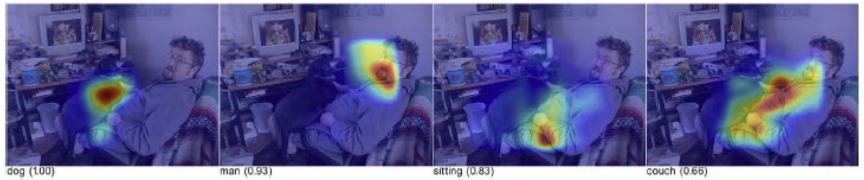
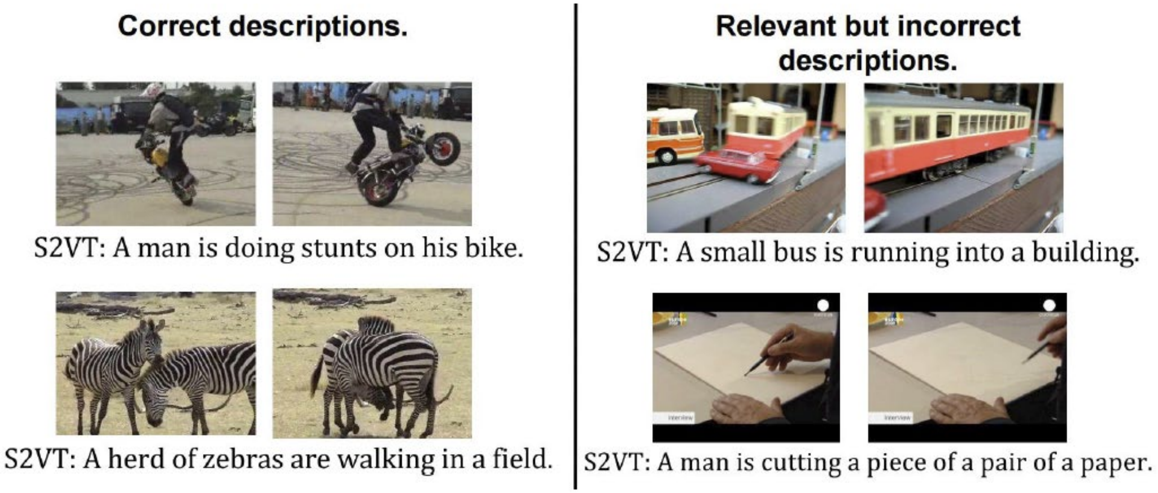
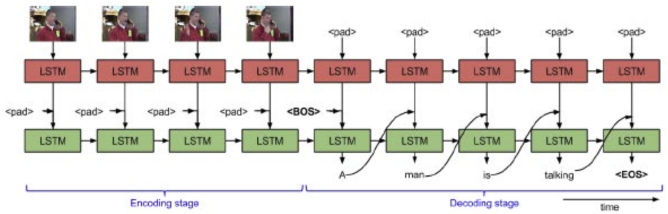


Image Caption Generation





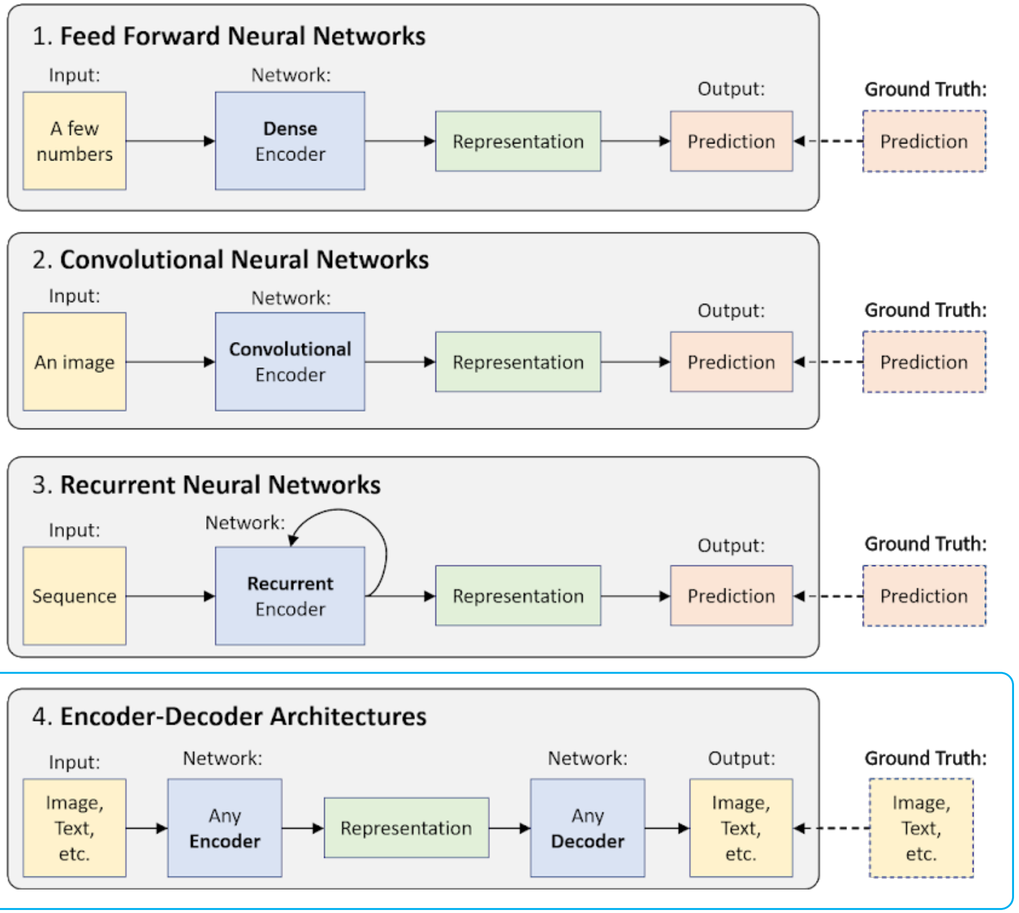
Venugopalan et al.
"Sequence to sequence-video to text." 2015.

Code: https://vsubhashini.github.io/s2vt.html

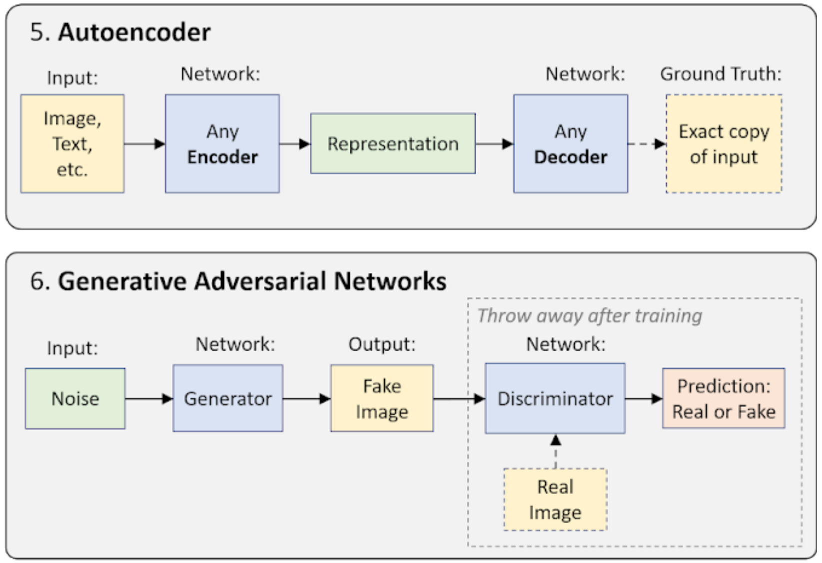
Video Description Generation

Selected NNs in Detail

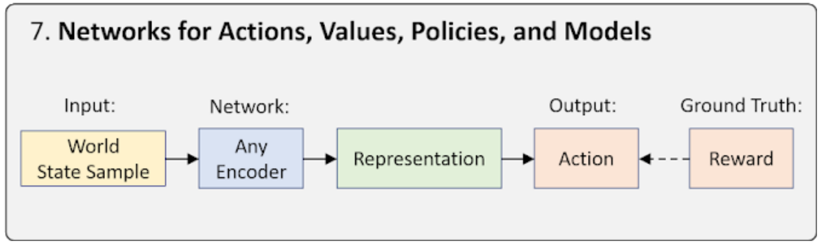
Supervised Learning



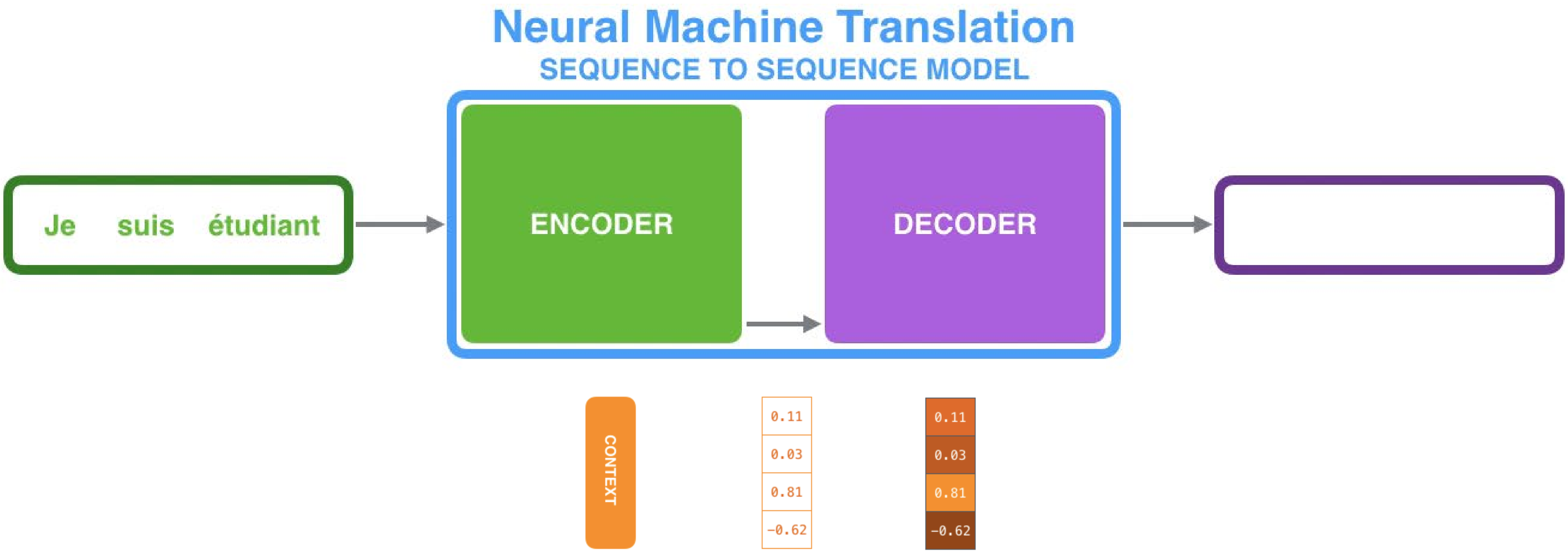
Unsupervised Learning



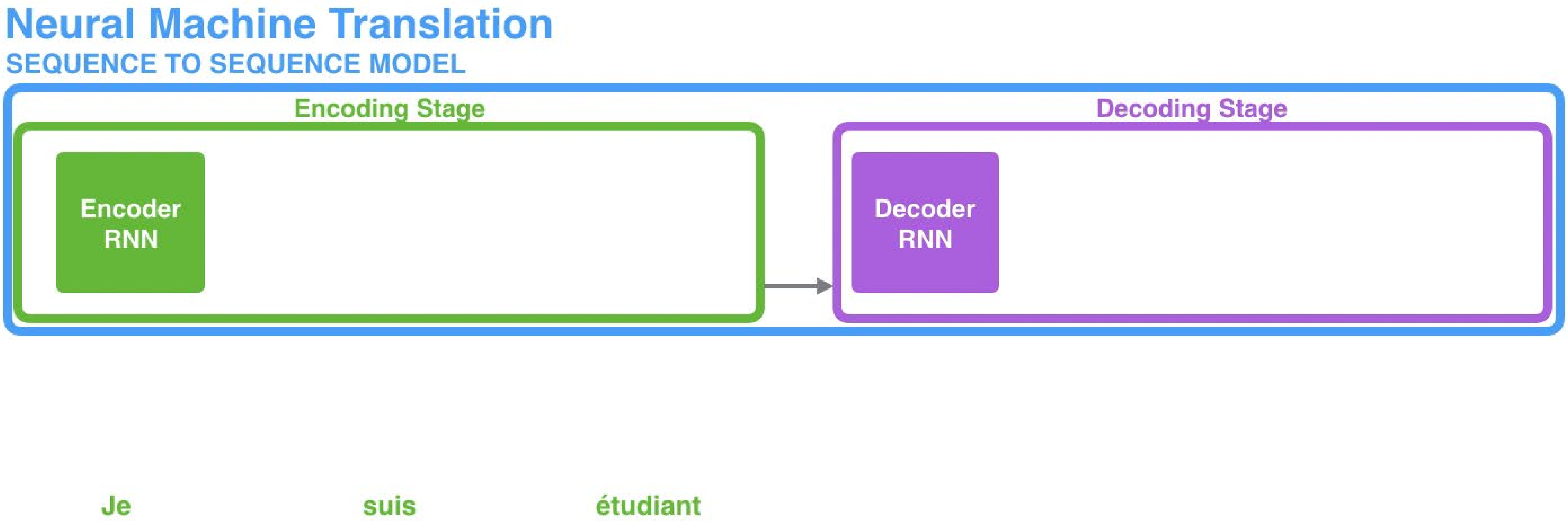
Reinforcement Learning



Seq2Seq: Encoder & Decoder

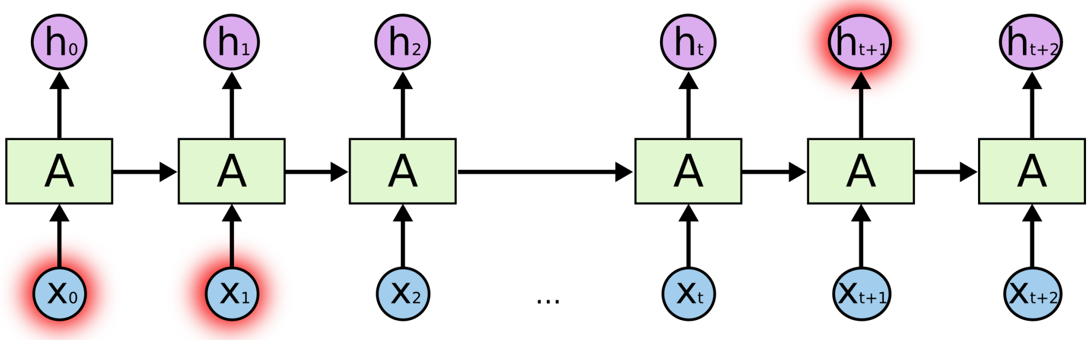


LSTM: Unrolled Process

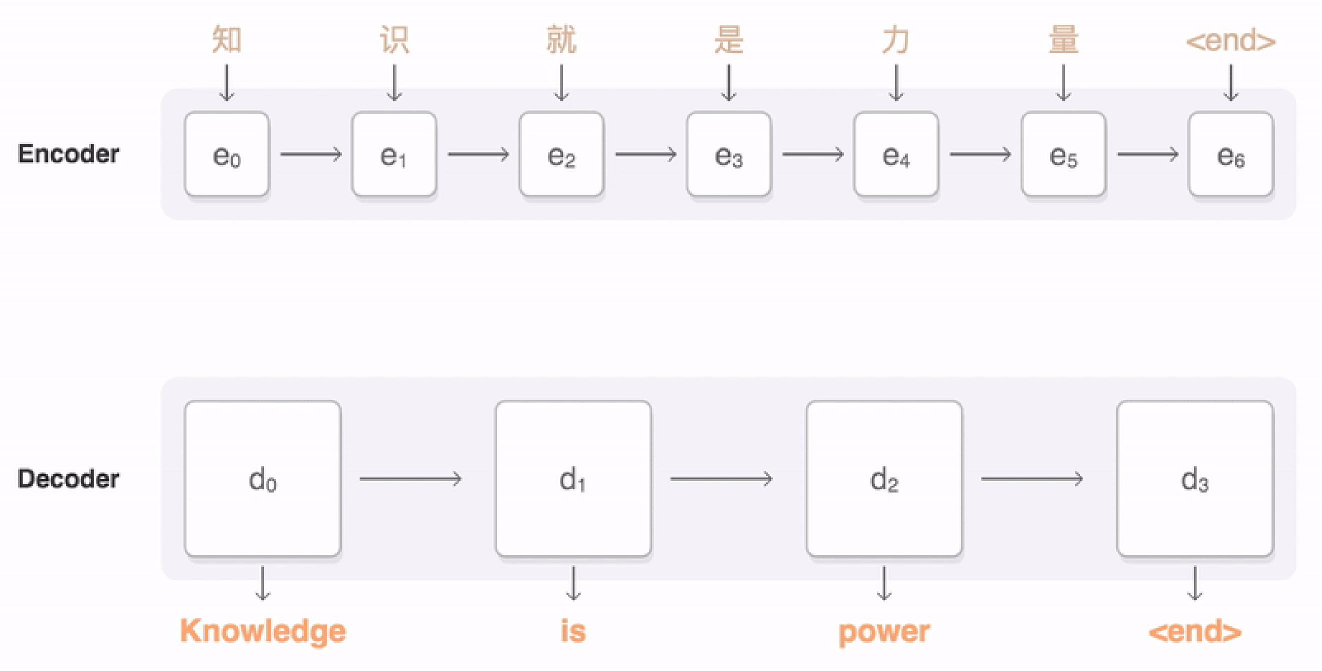


Attention is all you need

"*<u>Bob</u> likes <u>apples</u>.* He is hungry and decided to have a snack. So now he is eating an **apple**."

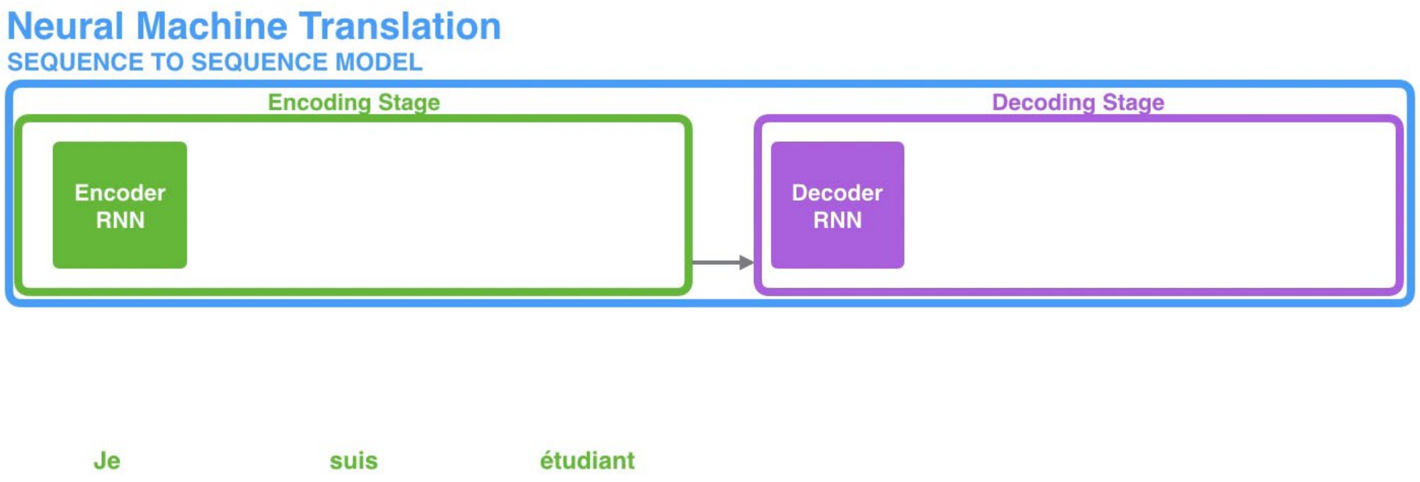


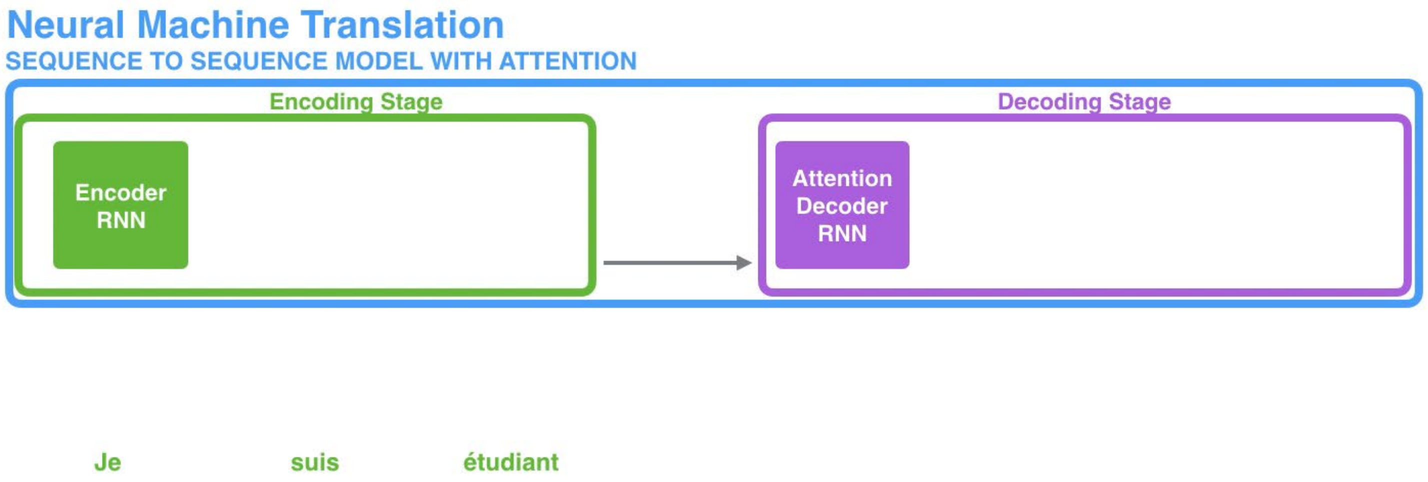
"*<u>Bob</u> likes <u>apples</u>.* He is hungry and decided to have a snack. Alice likes oranges and she is having lunch with Kate and Bob in the park. He is eating an **apple**."



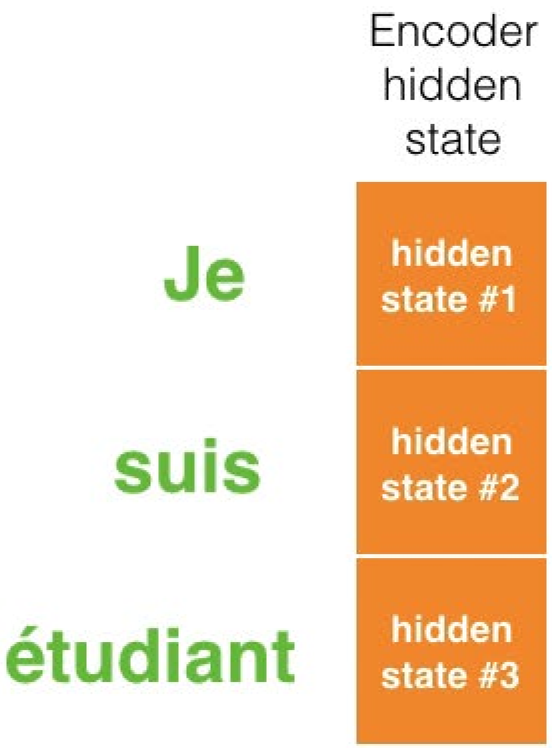
LSTMs does not solve the problem of RNNs completely:
when sentences are long, the model often forgets the content of distant positions in the sequence

LSTM v.s. Attention



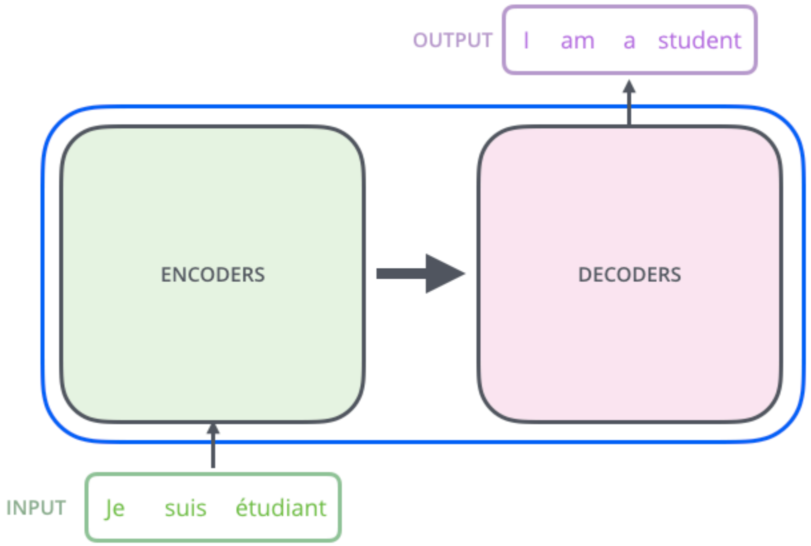


Attention


Encoder
hidden
state
Je
suis
étudiant
hidden
state #1
hidden
state #2
hidden
state #3

Transformer

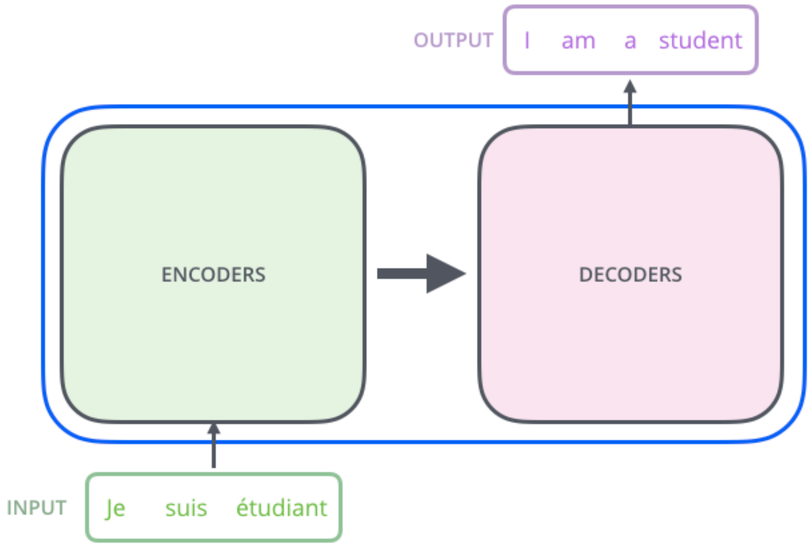
The Transformer – a model that uses **attention** to boost the speed with which these models can be trained.



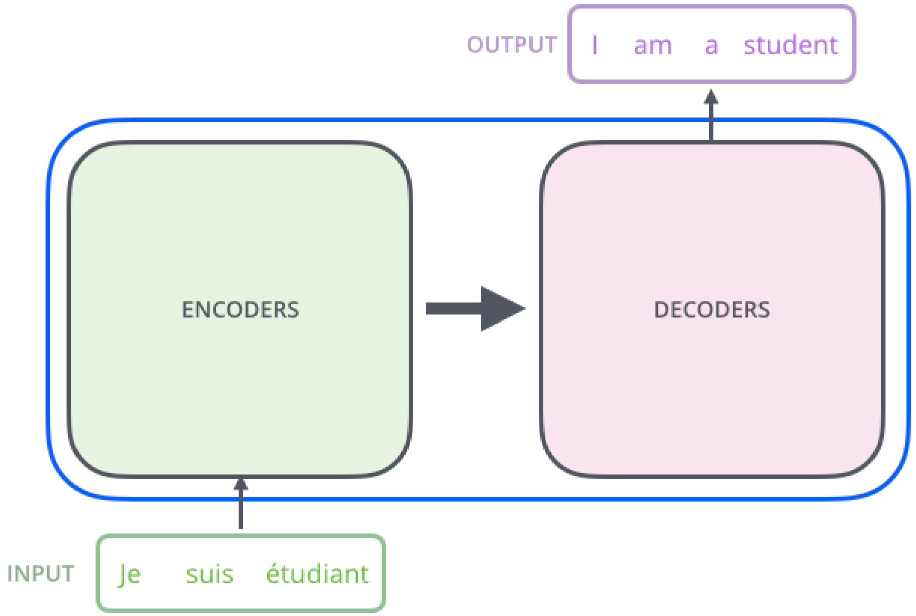
Transformer

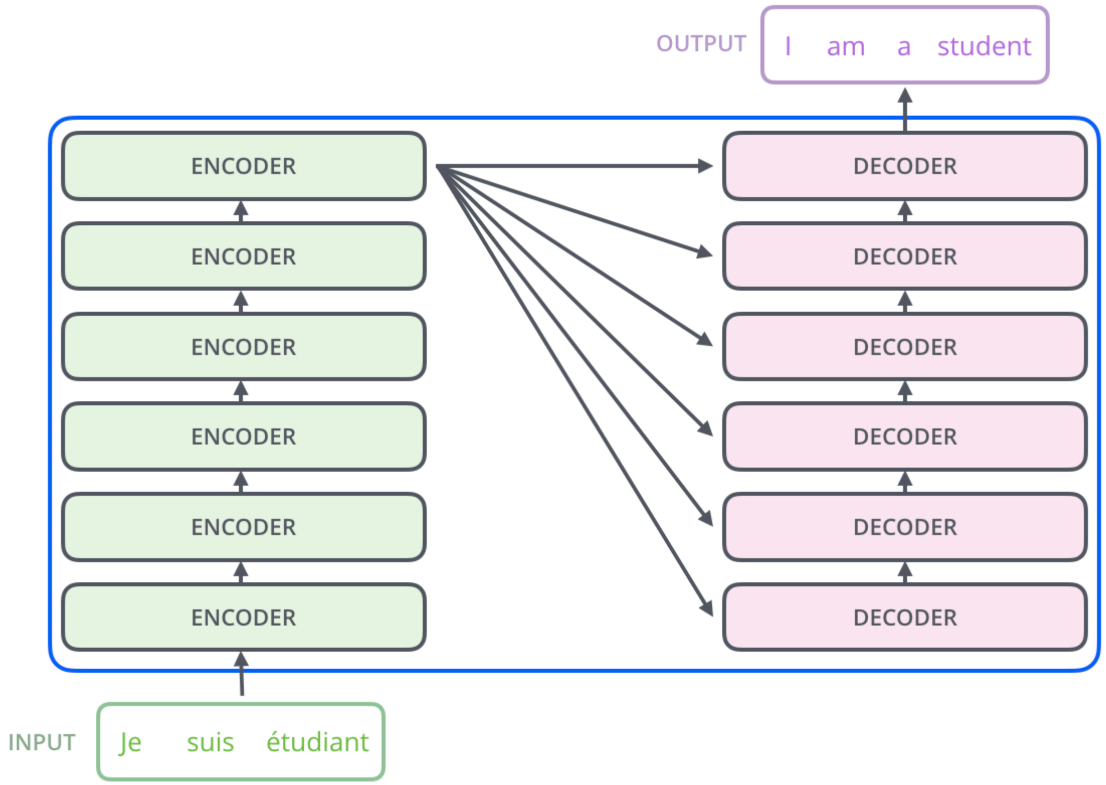
The Transformer – a model that uses **attention** to boost the speed with which these models can be trained.

Speed, accuracy, parallelization.

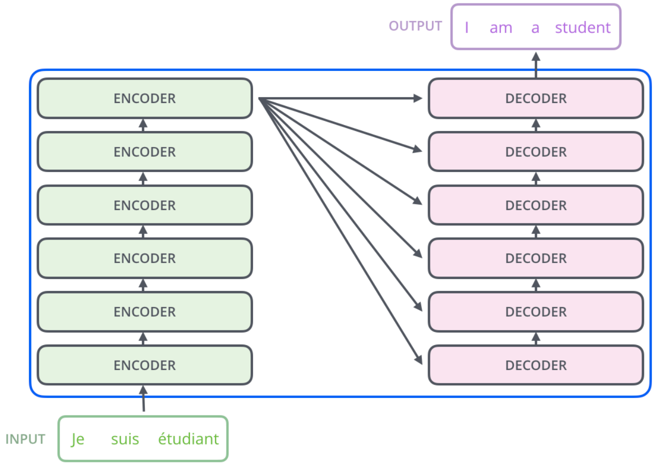


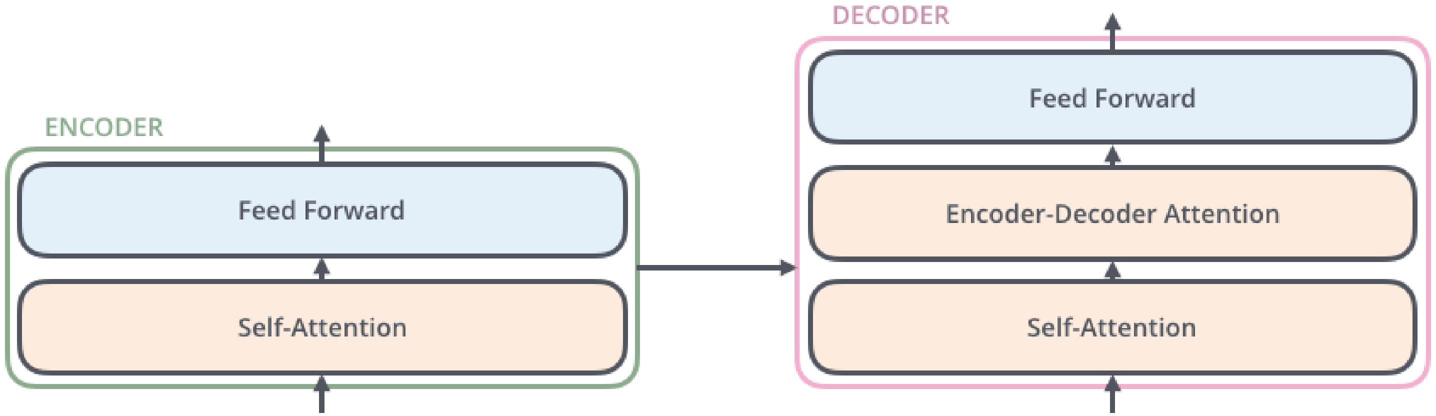
Transformer: more encoder & decoder



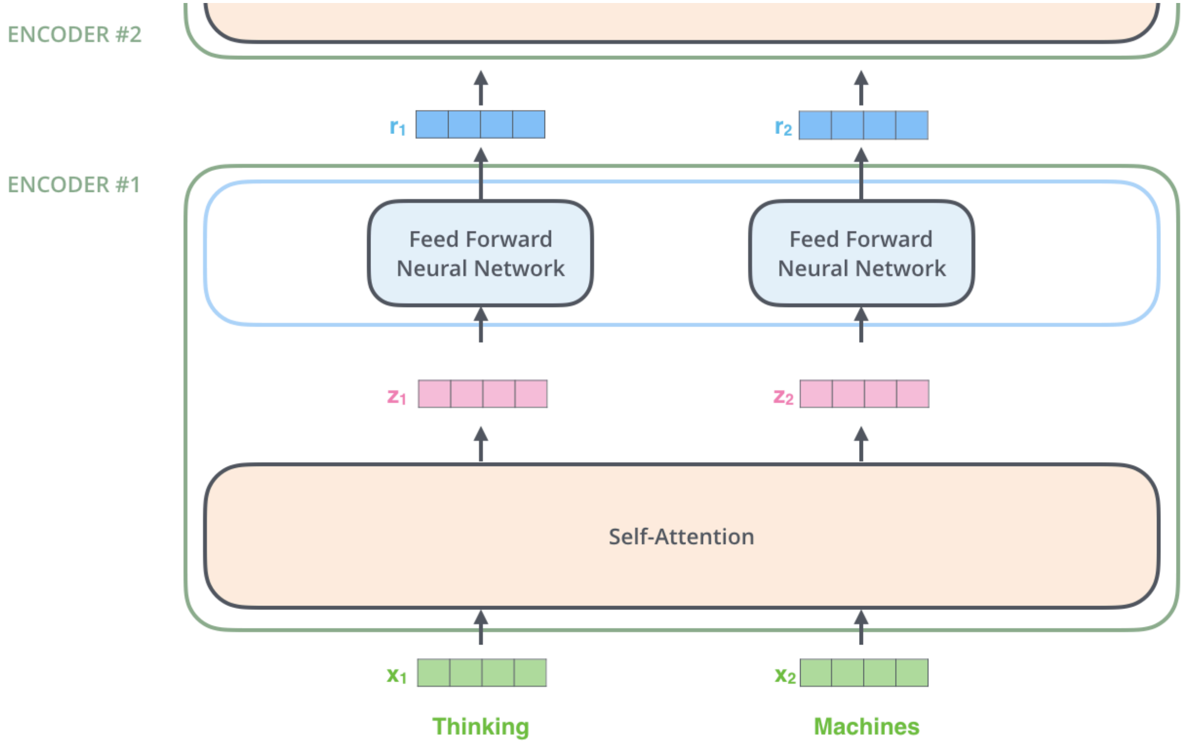


Transformer: more network



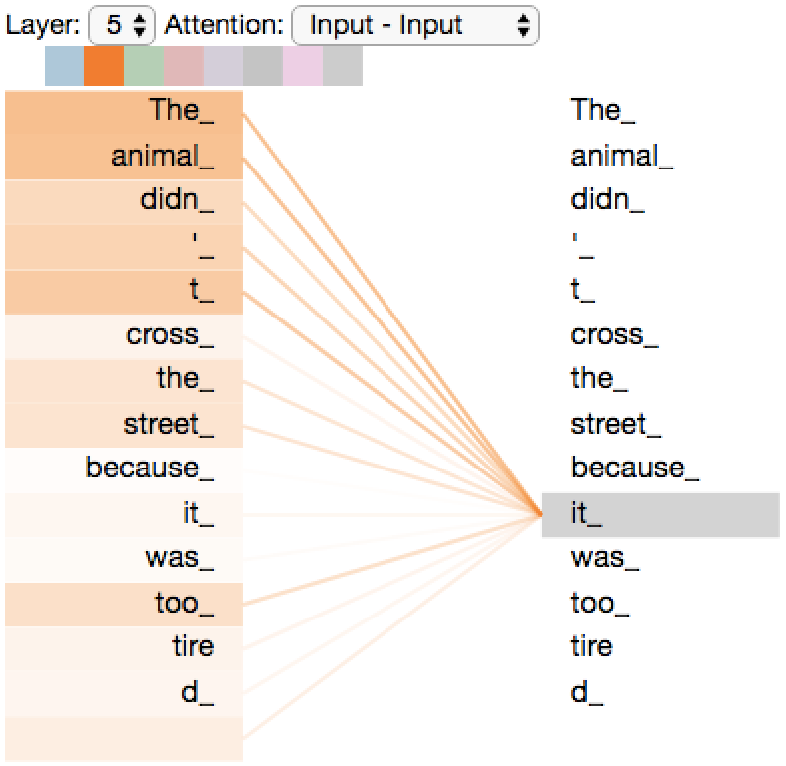


Transformer: Self-Attention



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

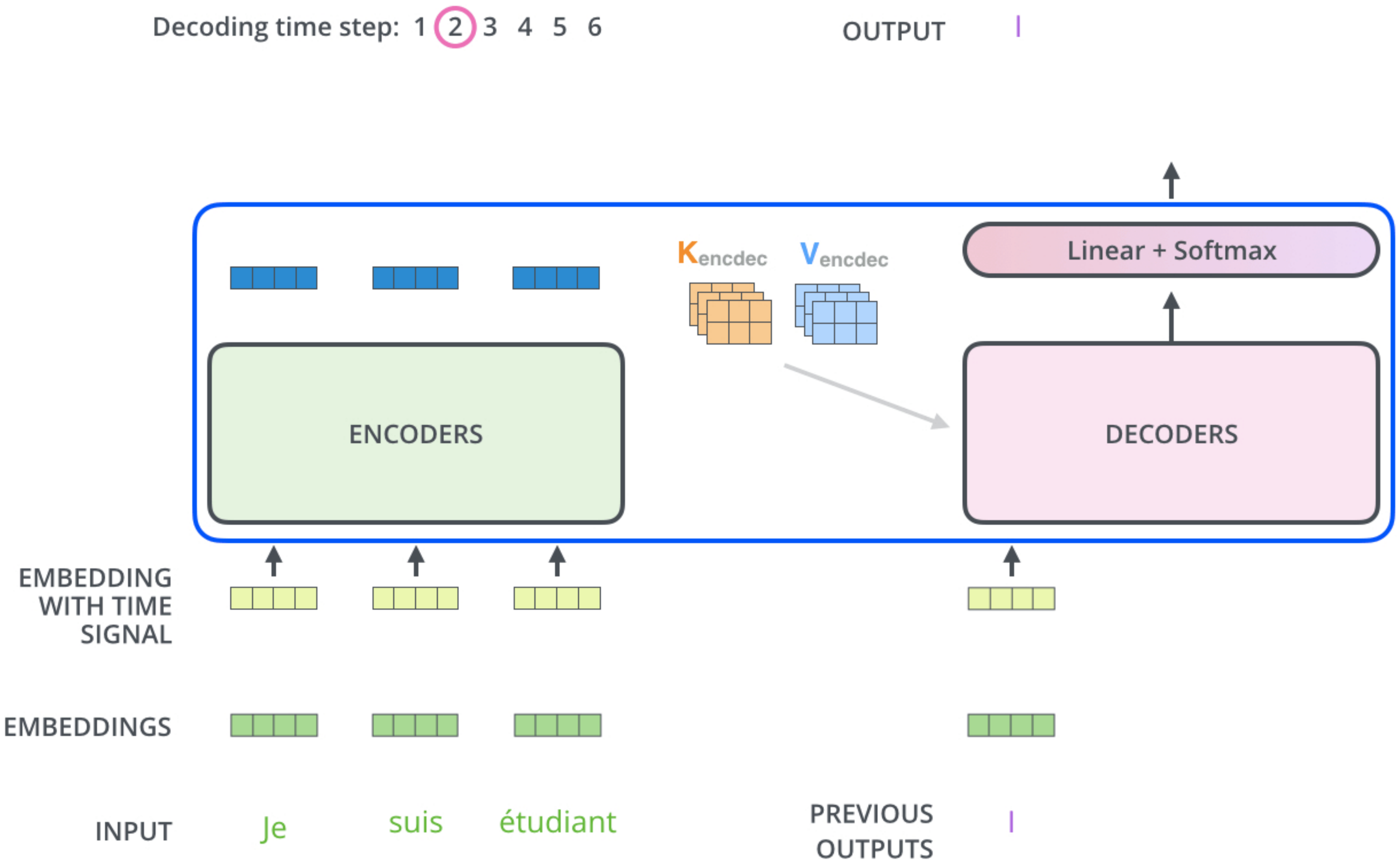
"The animal didn't cross the street because it was too tired"



When encoding "it" in encoder #5 (the top encoder in the stack):

part of the attention mechanism was focusing on "The Animal", and baked a part of its representation into the encoding of "it".

Transformer



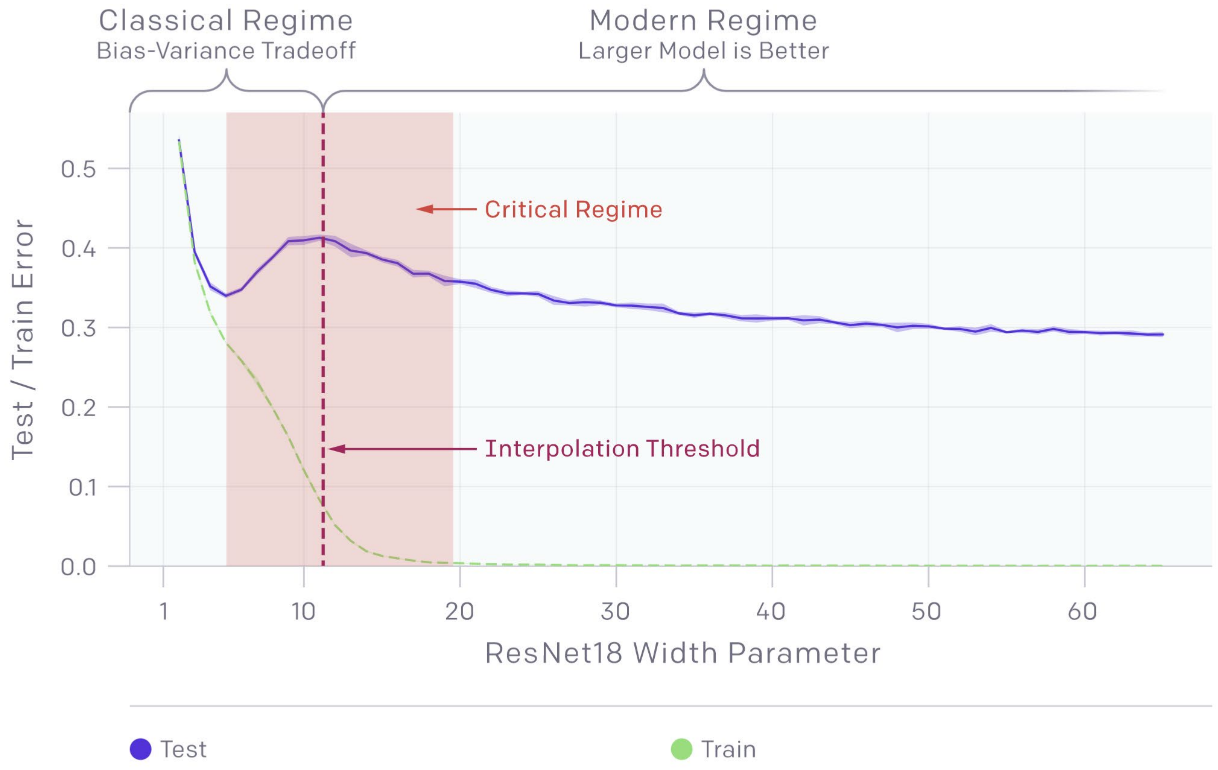
Deep Double Descent

"An effect occurs in CNNs, ResNets, and transformers: **performance first improves, then gets worse, and then improves again with increasing model size, data size, or training time.**"

Challenges “conventional wisdoms.”

- Bias-variance trade-off: “larger models are worse.”
- Modern NN: “larger models are better.”
- “early stopping” is sometimes good.

EMC and three regimes



Effective model complexity (EMC)
the maximum number of samples on which it can achieve close to zerotraining error.

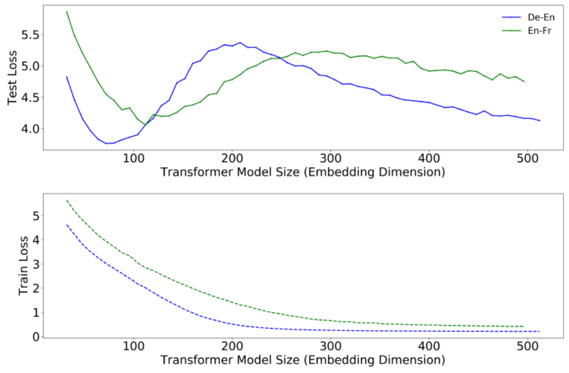
Under-paremeterized regime: $EMC(T) \ll n$:
any perturbation of T that increases its effective complexity will decrease the test error.

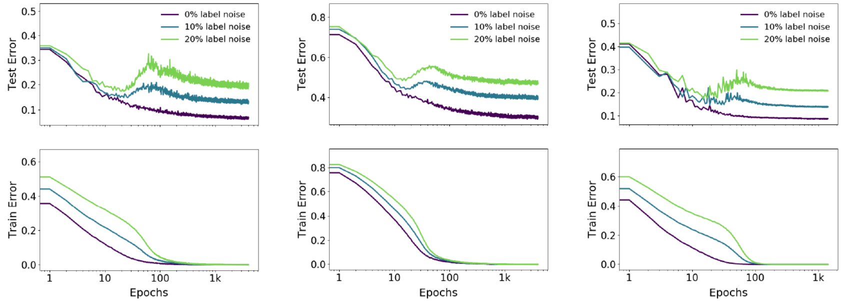
Over-parameterized regime $EMC(T) \gg n$:
any perturbation of T that increases its effective complexity will decrease the test error.

Critically parameterized regime $EMC(T) \approx n$:
a perturbation of T that increases its effective complexity might decrease or increase the test error.

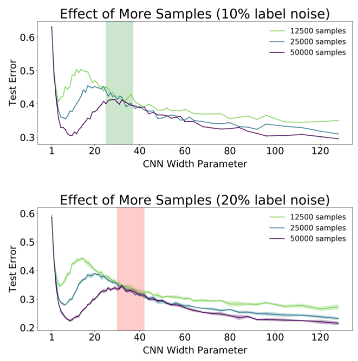
Deep Double Descent: A Stable Phenomenon

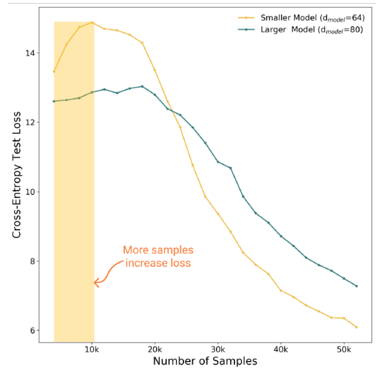
- Model-wise Double Descent
- Epoch-wise Double Descent
- Sample-wise Non-monotonicity





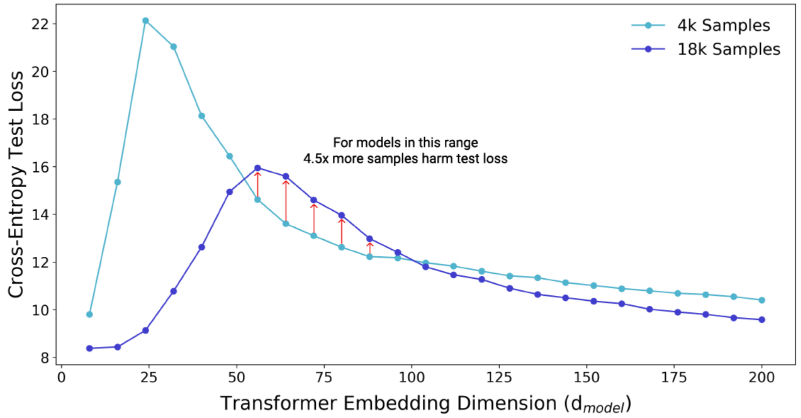
(a) ResNet18 on CIFAR10. (b) ResNet18 on CIFAR100. (c) 5-layer CNN on CIFAR 10.



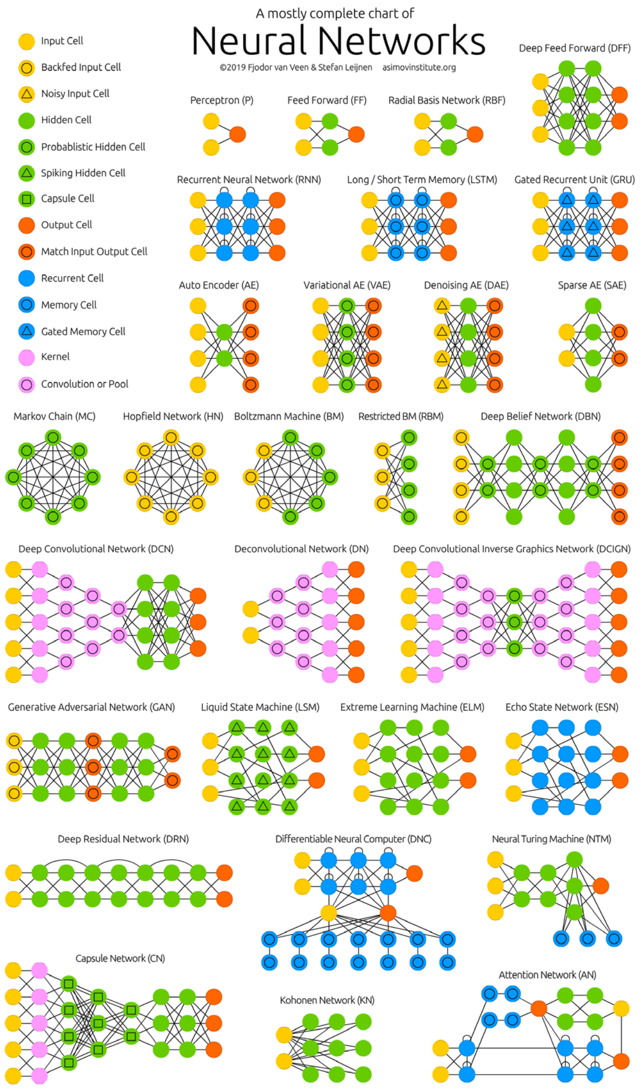


Take-away

- **Model behaves unexpectedly in transition regime**
- Training longer reverses overfitting
    - Double the training epoch is a technique in some task
- Bigger models are worse
- More data hurts



"While this behaviour appears to be fairly universal, we don't yet fully understand why it happens, and view further study of this phenomenon as an important research direction."



Reference

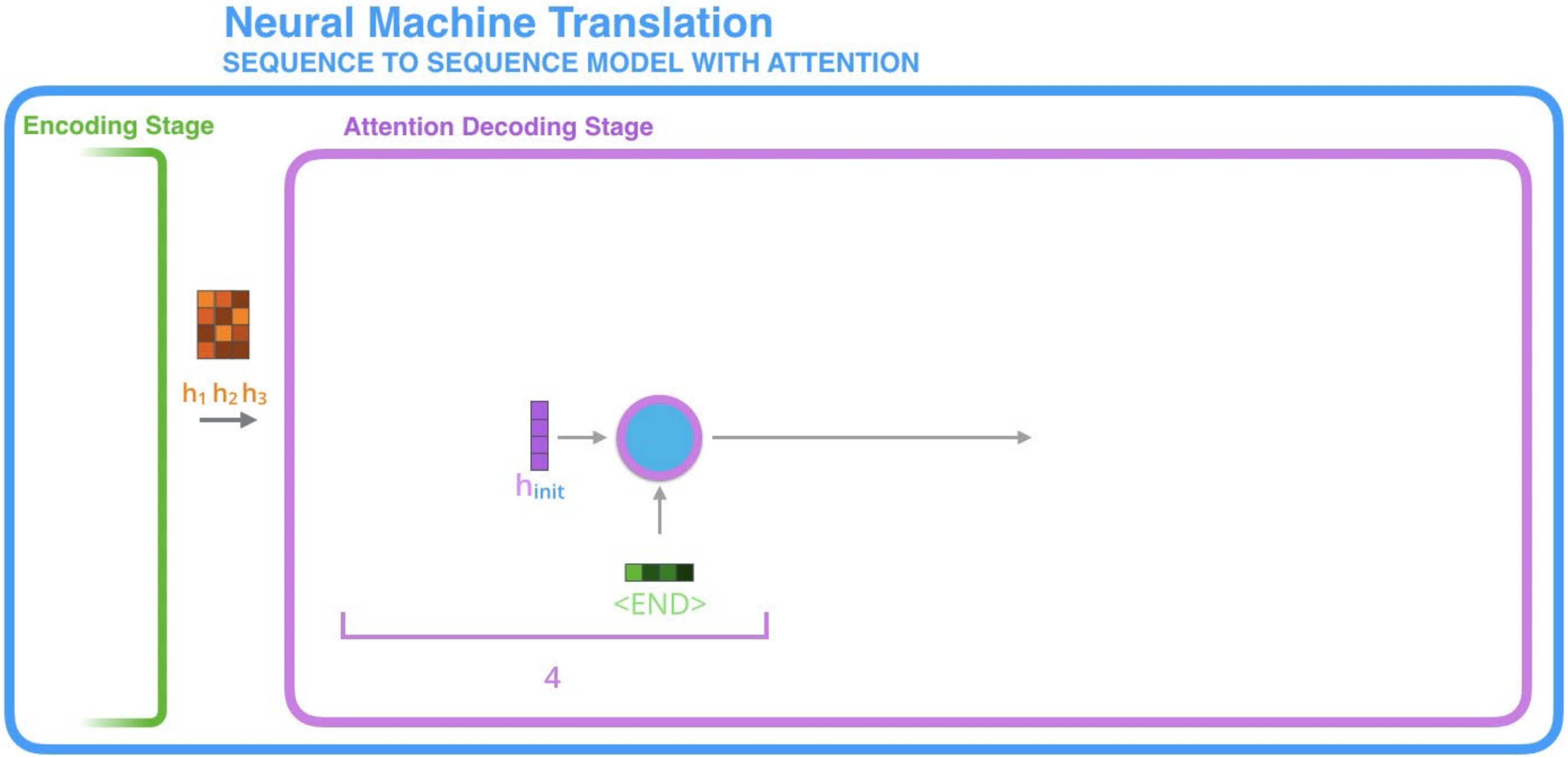
- https://deeplearning.mit.edu
- https://blog.tensorflow.org/2019/02/mit-deep-learning-basics-introduction-tensorflow.html
- http://cs231n.github.io/neural-networks-1/
- https://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/
- http://jalammar.github.io
- https://hackernoon.com/illustrative-proof-of-universal-approximation-theorem-5845c02822f6

- Liao, Qianli, and Tomaso Poggio. "Bridging the gaps between residual learning, recurrent neural networks and visual cortex." arXiv preprint arXiv:1604.03640 (2016).
- Nakkiran, Preetum, et al. "Deep double descent: Where bigger models and more data hurt." arXiv preprint arXiv:1912.02292 (2019).
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232.

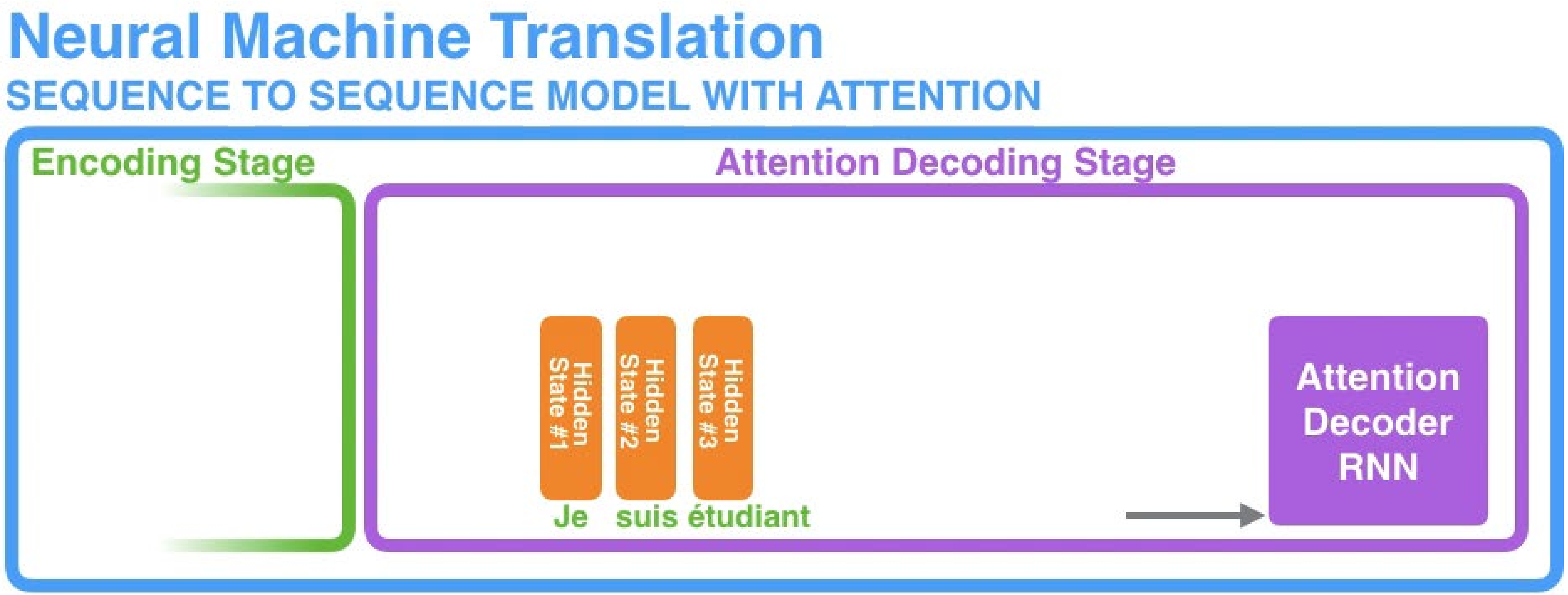
RNN → LSTM →**Attention** → Transformer

Attention at time step 4

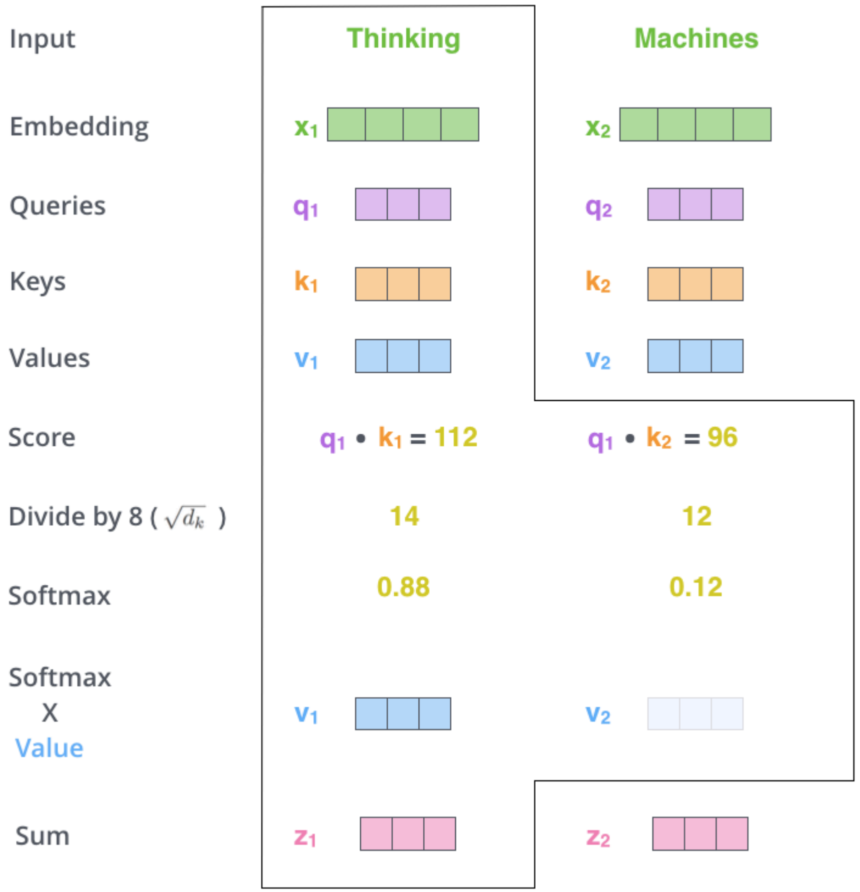
RNN → LSTM →**Attention** → Transformer

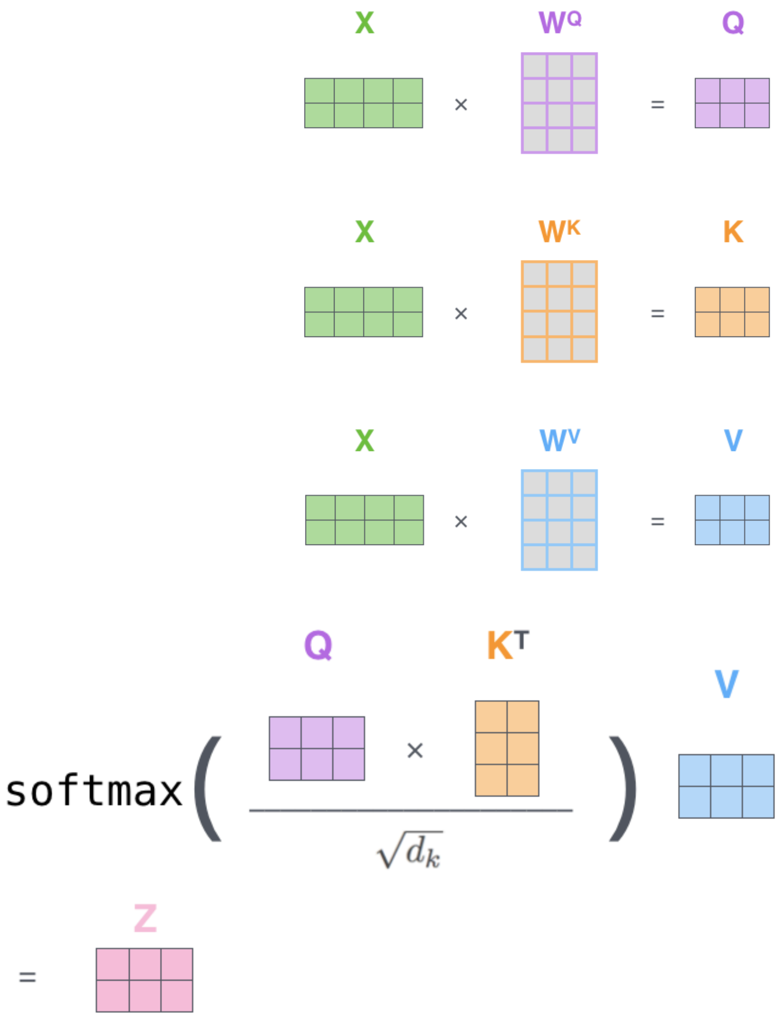


RNN → LSTM →**Attention** → Transformer



RNN → LSTM →Attention → **Transformer**





$$\mathrm{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$