Faculty of Mathematics, Physics and Informatics
Comenius University Bratislava

Neural Networks

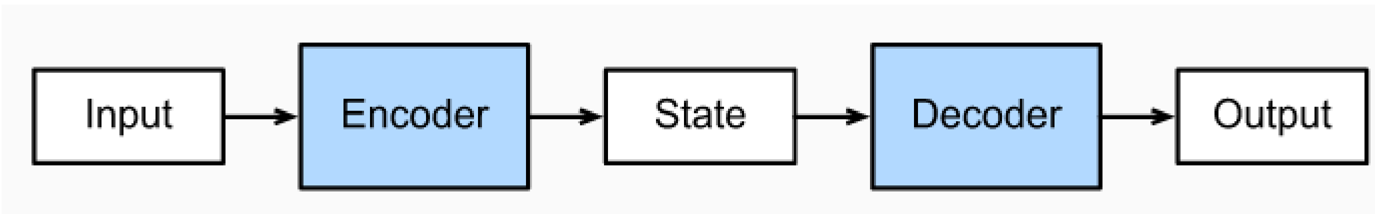
Lecture 11

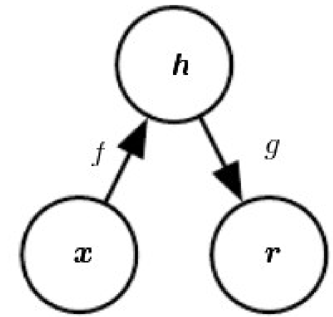
Autoencoders, gated recurrent models and transformers

Igor Farkaš

2024

Autoencoders





- Encoder-decoder architecture = NN that is trained to attempt to copy its input to its output
- We focus on simpler case – a spatial mapping (no time involved)
- Encoder $\boldsymbol{h} = f(\boldsymbol{x})$, decoder $\boldsymbol{r} = g(\boldsymbol{h}) = g(f(\boldsymbol{x}))$ yields reconstruction
- $\dim(\boldsymbol{x}) = \dim(\boldsymbol{r}) > \dim(\boldsymbol{h})$ → bottleneck
- imperfect reconstruction crucial (due to bottleneck)
- AE can also be stochastic: $p_{\text{encoder}}(\boldsymbol{h} \mid \boldsymbol{x})$ and $p_{\text{decoder}}(\boldsymbol{x} \mid \boldsymbol{h})$, leading to generative models

Purpose

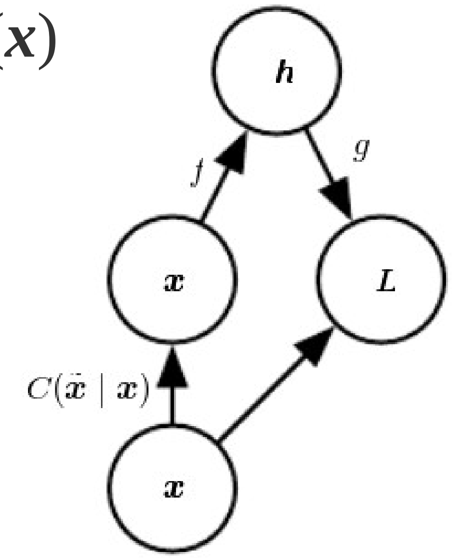
- Autoencoders – used for dimensionality reduction, since 1980s (LeCun, 1987; Bourlard & Kamp, 1988)
- undercomplete AE, i.e. If $\dim(\boldsymbol{h}) < \dim(\boldsymbol{x})$ → bottleneck
  - captures the most salient features of the training data
- Self-supervised training to minimize loss function $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$
- if linear and $Loss$ = MSE, then → PCA,
- nonlinear AE is a more powerful generalization
- overcomplete AE, i.e. $\dim(\boldsymbol{h}) > \dim(\boldsymbol{x})$ interesting only...
- … if regularized, in order to learn data distribution (in latent space)
- Interesting properties at hidden layer: sparsity, small derivatives of the representation, robustness

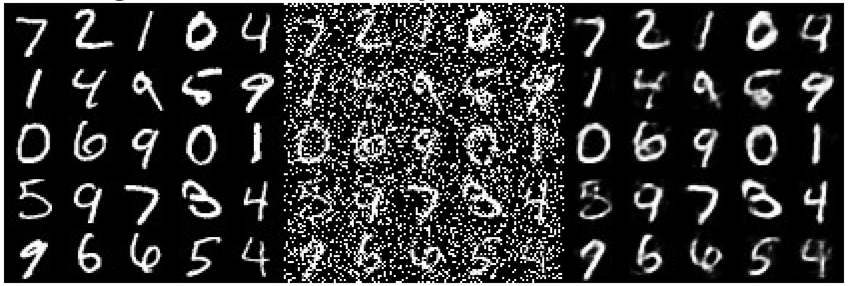
Sparse autoencoders

- Trained to minimize $L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + P(\boldsymbol{h})$, ($P$ = sparsity penalty)
- typically used to learn features for another task (such as classification)
- e.g. $P(\boldsymbol{h}) = \lambda \Sigma_i |h_i|$
- using ReLU activation function also enforces sparsity
- Probabilistic interpretation: learn generative model $p_{\text{model}}(\boldsymbol{x} \mid \boldsymbol{h})$ that best explains observed data (by latent variables)
- Alternative: $L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + P(\boldsymbol{h},\boldsymbol{x})$, where
- $P(\boldsymbol{h},\boldsymbol{x}) = \lambda \Sigma_i \|\nabla_{\boldsymbol{x}} h_i\|^2$ → contractive autoencoder

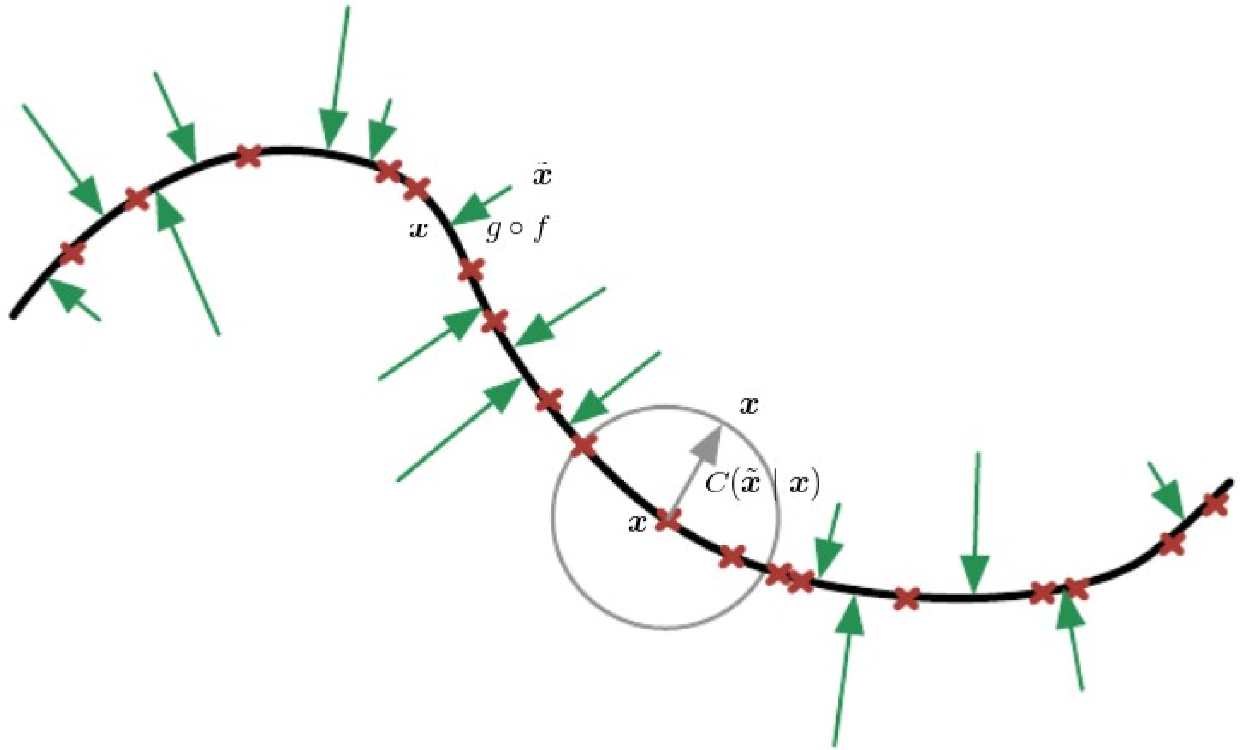
Denoising autoencoders

- Based on changing the reconstruction error term of the cost function (rather than adding penalty term)
- Minimizes $L(\boldsymbol{x}, \boldsymbol{g}(f(\boldsymbol{x}')))$, where $\boldsymbol{x}'$ is noisy version of input $\boldsymbol{x}$
- implicitly forced to learn the structure of data $p_{\text{data}}(\boldsymbol{x})$
- Introduces corruption process $C(\boldsymbol{x}' \mid \boldsymbol{x})$
- DAE learns reconstruction distrib. $p_{\text{reconstruct}}(\boldsymbol{x} \mid \boldsymbol{x}')$
- … from training pairs $\{\boldsymbol{x}', \boldsymbol{x}\}$
- can by trained by SGD as any feedforward NN





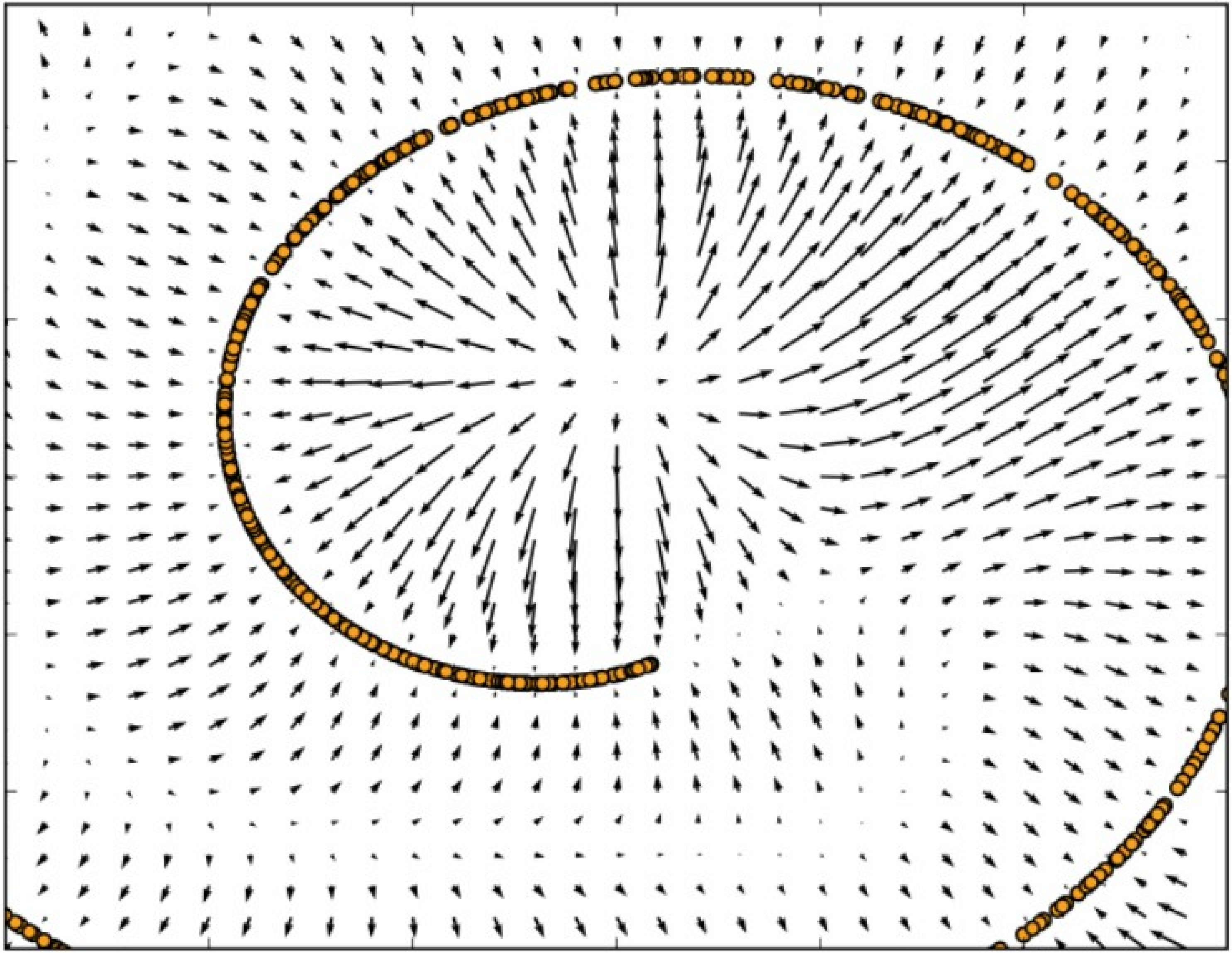
Graphical interpretation of DAE learning



- data $\boldsymbol{x}$ assumed to lie on a low-dim. manifold **M** (black curve)
- noisy inputs $\boldsymbol{x}'$ represent departures from **M**
- DAE learns a vector field (green arrows): $g(f(\boldsymbol{x})) - \boldsymbol{x}$
- projections onto the manifold

(Goodfellow et al., 2015)

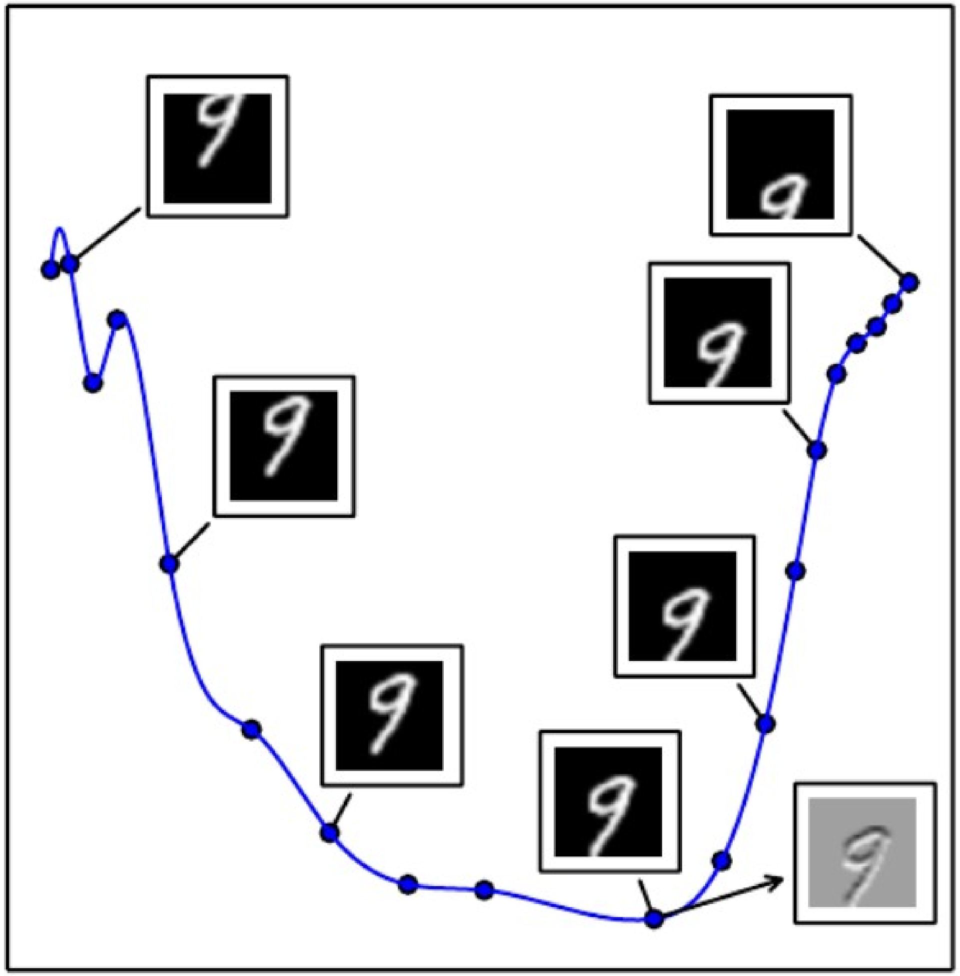
Example: 2D → 1D



(Alain & Bengio, 2013)

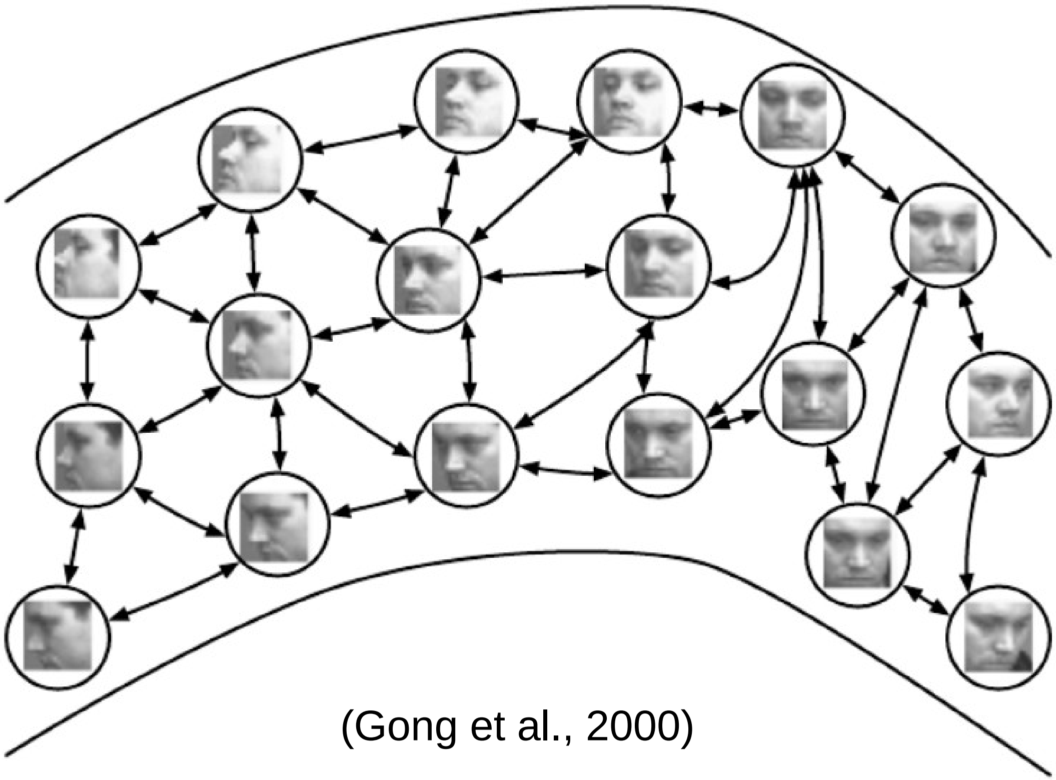
Manifold learning with autoencoder

- 1D example in 784-dim. space
- vertically translated images → a coordinate along **M**
- **M** projected in 2D (via PCA)
- Each node is associated with a tangent plane that spans the directions of variations associated with difference vectors between the example and its neighbors
- shown example (bottom right)



(Goodfellow et al., 2015)

2D example with manifold of faces

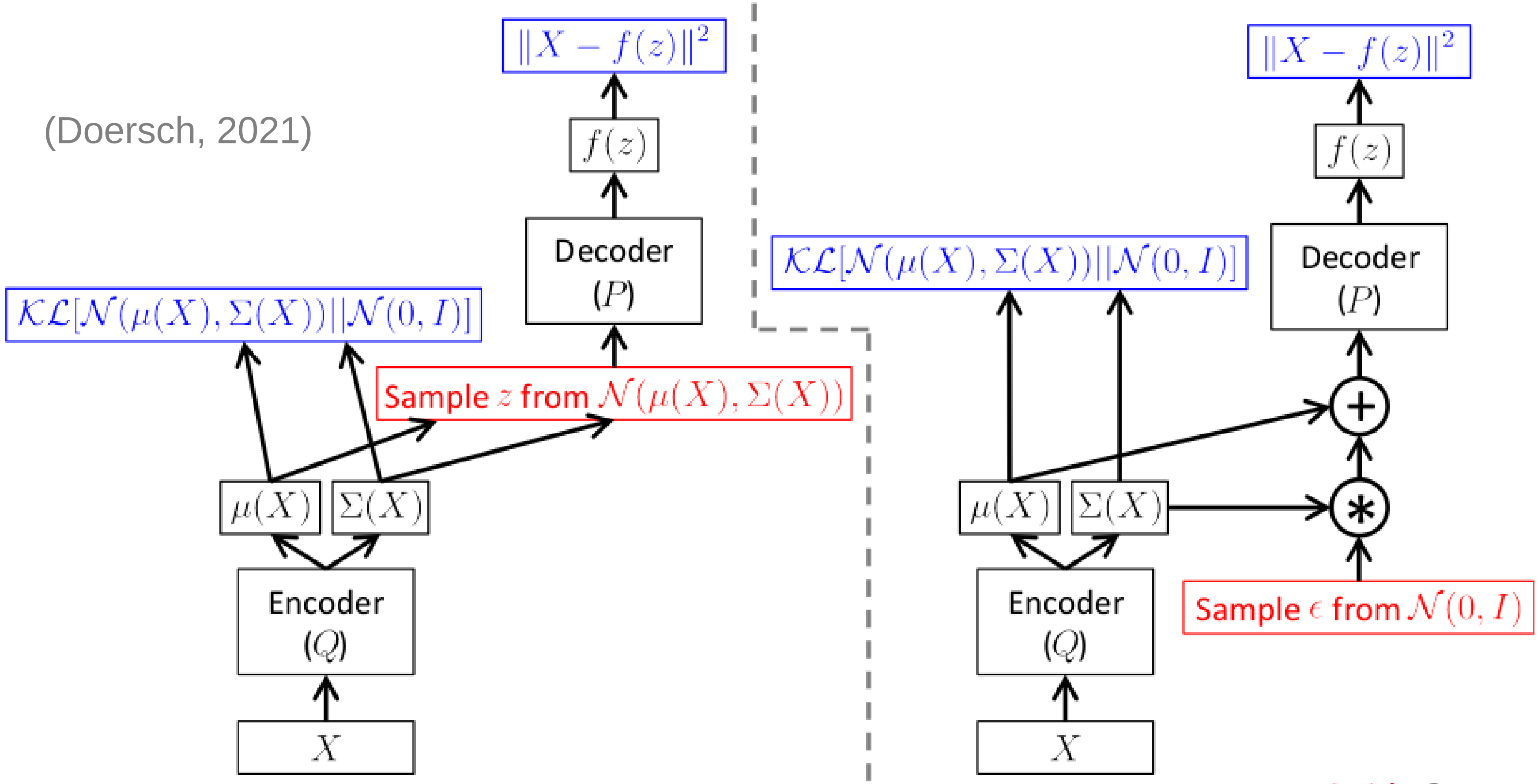


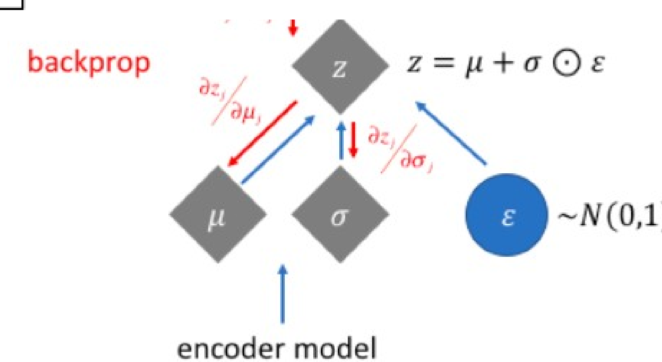
(Gong et al., 2000)

- Unsupervised learning of manifold (embedding) based on a (nonparametric) nearest neighbors graph
- Generalization to new examples possible via interpolation for dense graphs

Variational autoencoder (VAE)

(Doersch, 2021)

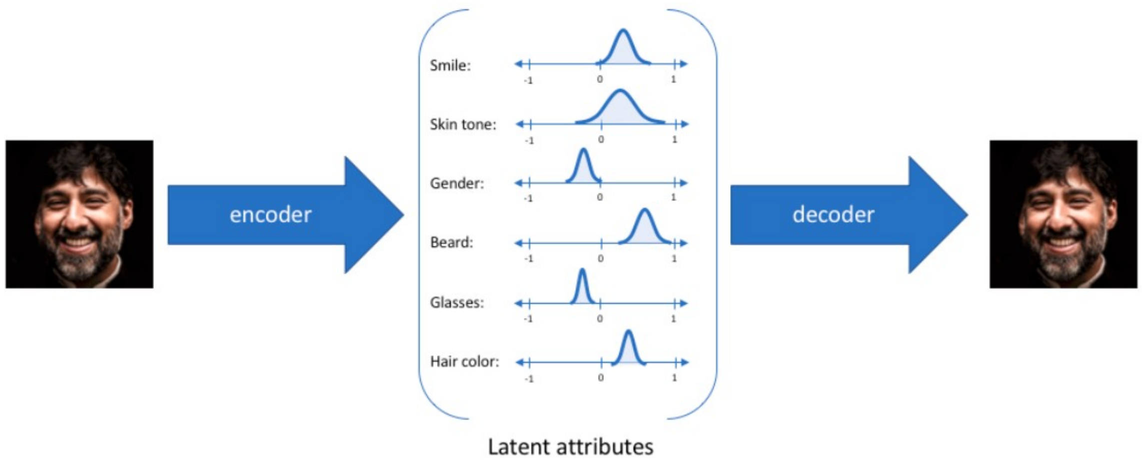


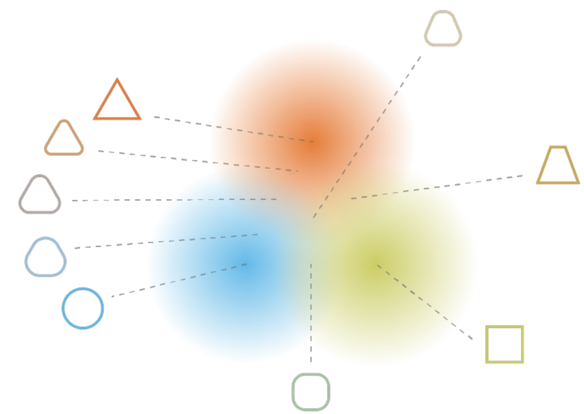


- Uses a reparametrization trick (right), which allows gradient propagation and controlled generation of samples
- conditional VAE possible

(Kingma et al., 2014)

VAE properties





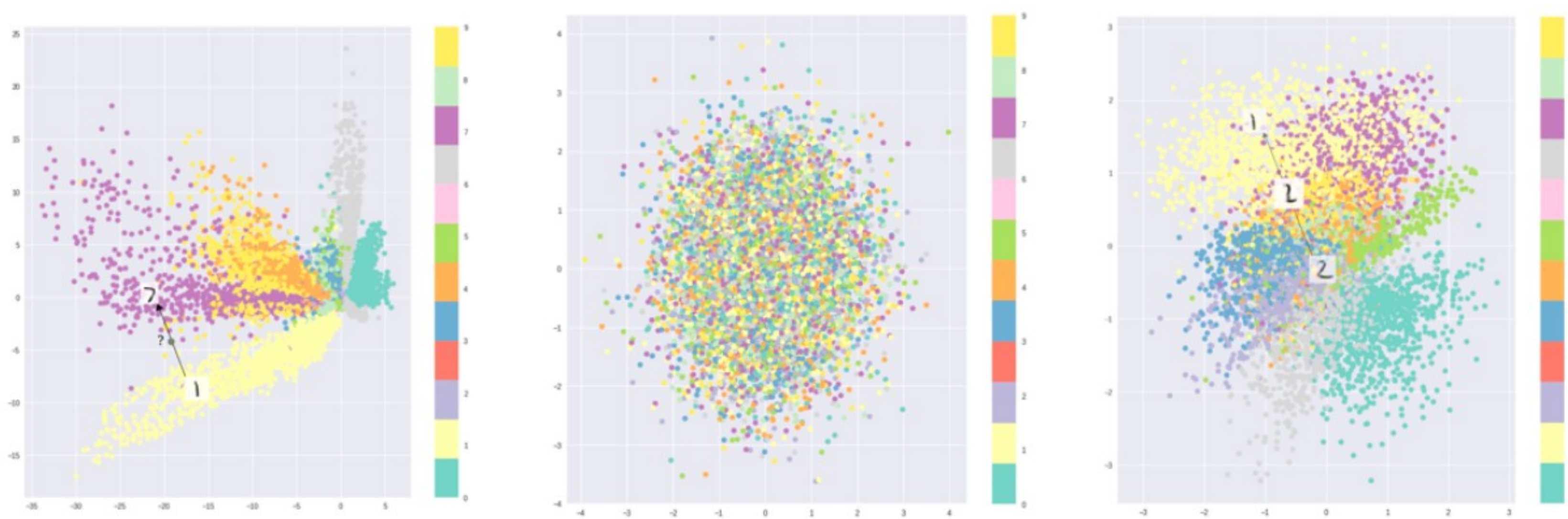
(Rocca,2019)

Latent space representation

Only reconstruction loss

Only KL divergence

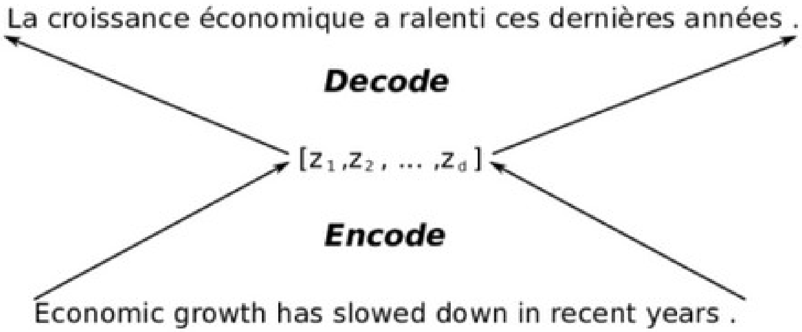
Combination



https://www.jeremyjordan.me/variational-autoencoders/

Applications of autoencoders

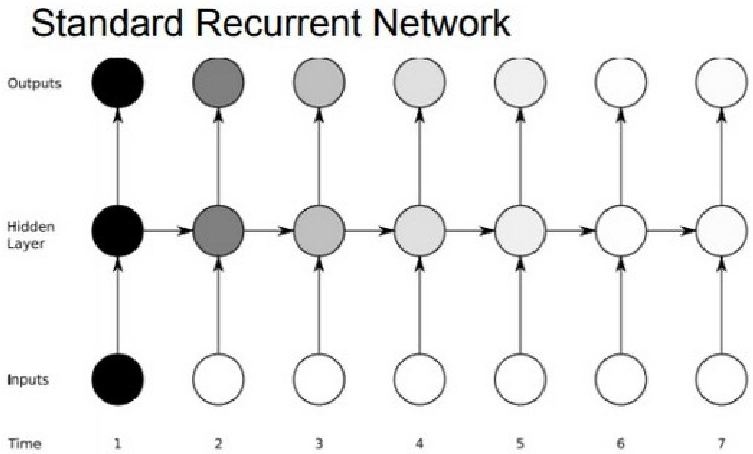
- Explicit dim. reduction for subsequent classification – reduces error (also less memory and runtime)
- can be applied recursively (hierarchically)
- Information retrieval – task of finding entries in a database that resemble (are relevant for) a query entry
  - entries mapped to binary low-dim. hash codes (fast search)
  - entries with the same or slightly different codes (a few bits flipped) retrieved → semantic hashing
  - sigmoid units used in encoding function (forced to saturate)
  - technique used for text and images
- machine translation

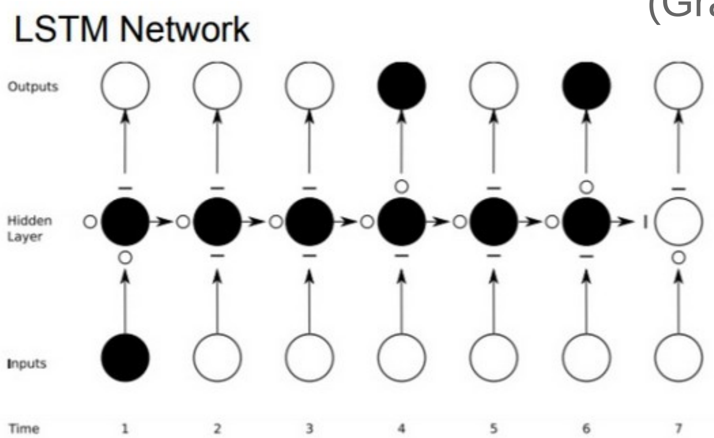


Recurrent NN models with gated units

- Help preserve long-term dependencies (via gradient learning)
- Two models will be mentioned: GRU (Cho et al, 2014) – simpler, LSTM (Hochreiter & Schmidhuber, 2007) – more complex
- New components:
  - memory cell (to capture long-term dependencies)
  - skipping irrelevant inputs (in latent space)
  - resetting (internal state representation)

(Graves et al, 2013)





GRU – Gating the hidden state

minibatches (of size *n*)

$\mathbf{X}_t\ [n \times d]$ (examples $\times$ dimension)

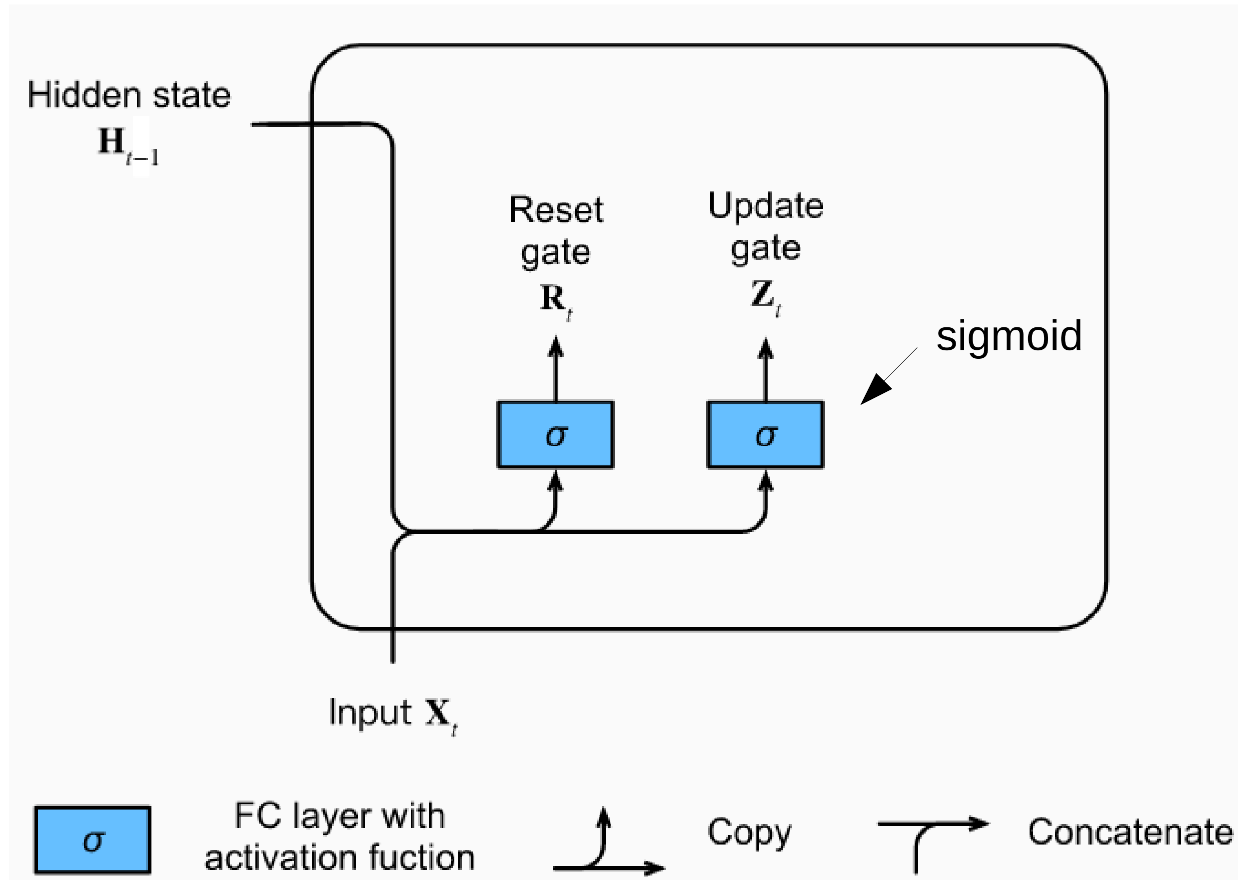
$\mathbf{H}_{t\text{-}1}\ [n \times h]$

$\mathbf{R}_t\ ,\ \mathbf{Z}_t\ [n \times h]$

$\mathbf{W}_{xr}\ ,\mathbf{W}_{xz}\ [d \times h]$

$\mathbf{W}_{hr}\ ,\mathbf{W}_{hz}\ [h \times h]$

$\boldsymbol{b}_t\ ,\ \boldsymbol{b}_z\ [1 \times h]$

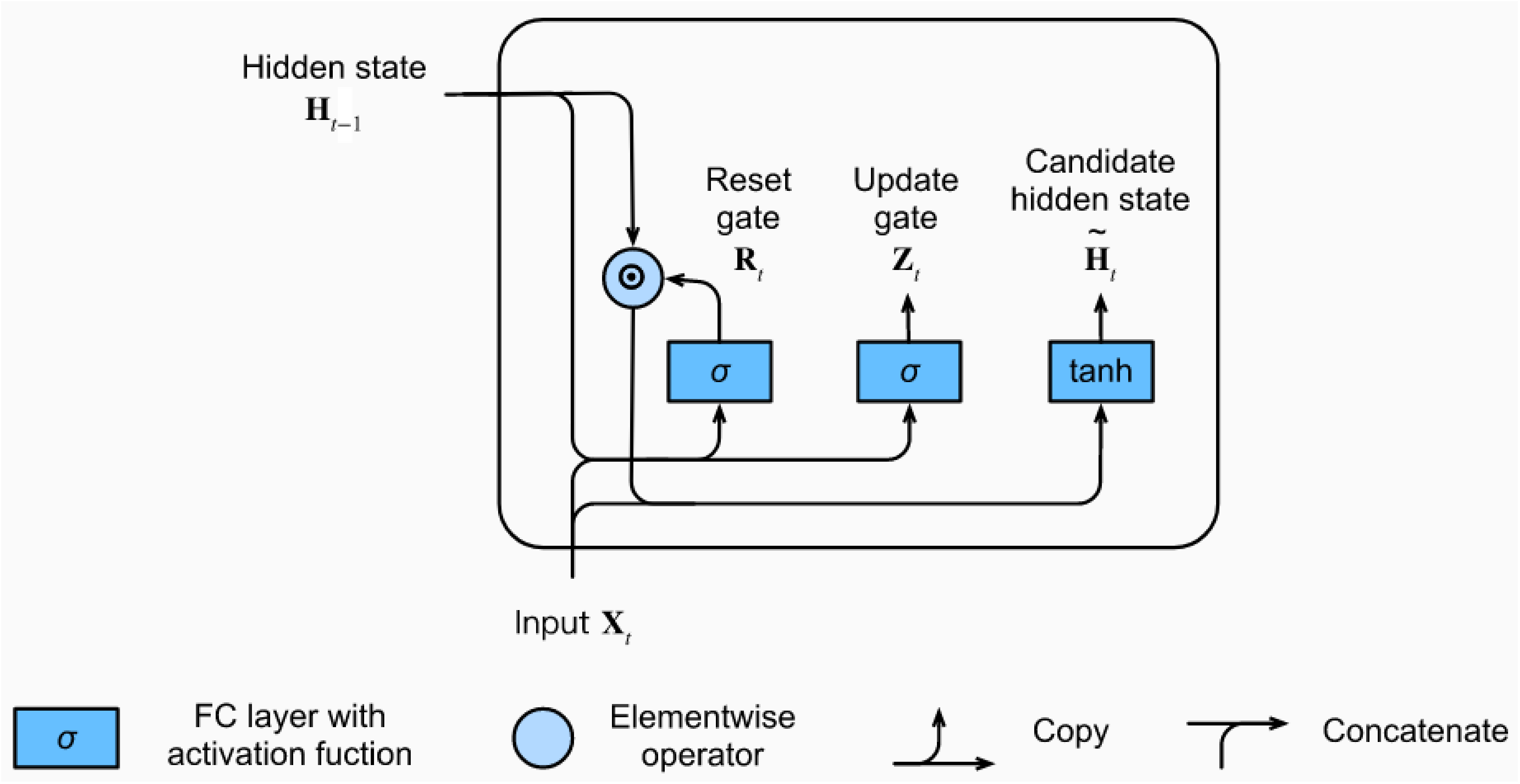


(Cho et al., 2014)

$$\mathbf{R}_t = \sigma(\mathbf{X}_t\ \mathbf{W}_{xr} + \mathbf{H}_{t\text{-}1}\ \mathbf{W}_{hr} + \boldsymbol{b}_r)$$
$$\mathbf{Z}_t = \sigma(\mathbf{X}_t\ \mathbf{W}_{xz} + \mathbf{H}_{t\text{-}1}\ \mathbf{W}_{hz} + \boldsymbol{b}_z)$$

(Zhang et al, 2020)

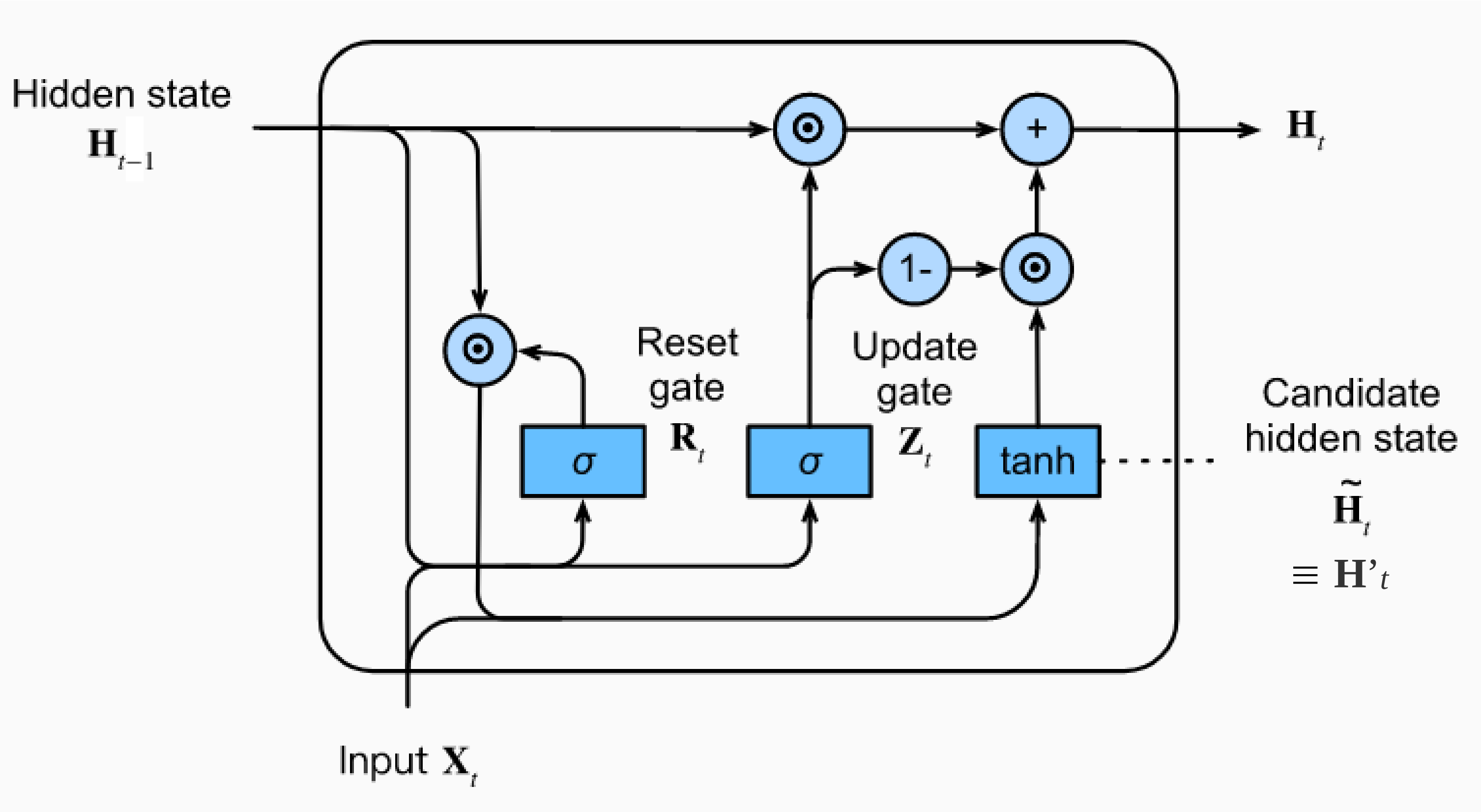
GRU – Reset gates in action



$$\mathbf{H}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{R}_t \odot \mathbf{H}_{t\text{-}1} \mathbf{W}_{hh} + \boldsymbol{b}_h)$$

- helps capture short-term dependencies in time series

GRU – Update gates in action



$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t\text{-}1} + (1 - \mathbf{Z}_t) \odot \mathbf{H}'_t$$

- help capture long-term dependencies in time series

LSTM's gated memory cells

- inspired by logic gates of a computer
- 3 gates controls the behavior of the memory cell (latent state)
- output gate – controls when to read from the cell
- input gate – controls when to read data into the cell
- forget gate – controls when to reset the contents of the cell
- In addition, LSTM introduces a memory cell ($\mathbf{C}$)
  - having the same shape as latent state ($\mathbf{H}$)
  - providing additional information
- GRU is simpler: has a single mechanism for input and forgetting

LSTM's three gates

minibatches (of size n)

$\mathbf{X}_t$ $[n \times d]$ (examples $\times$ dimension)

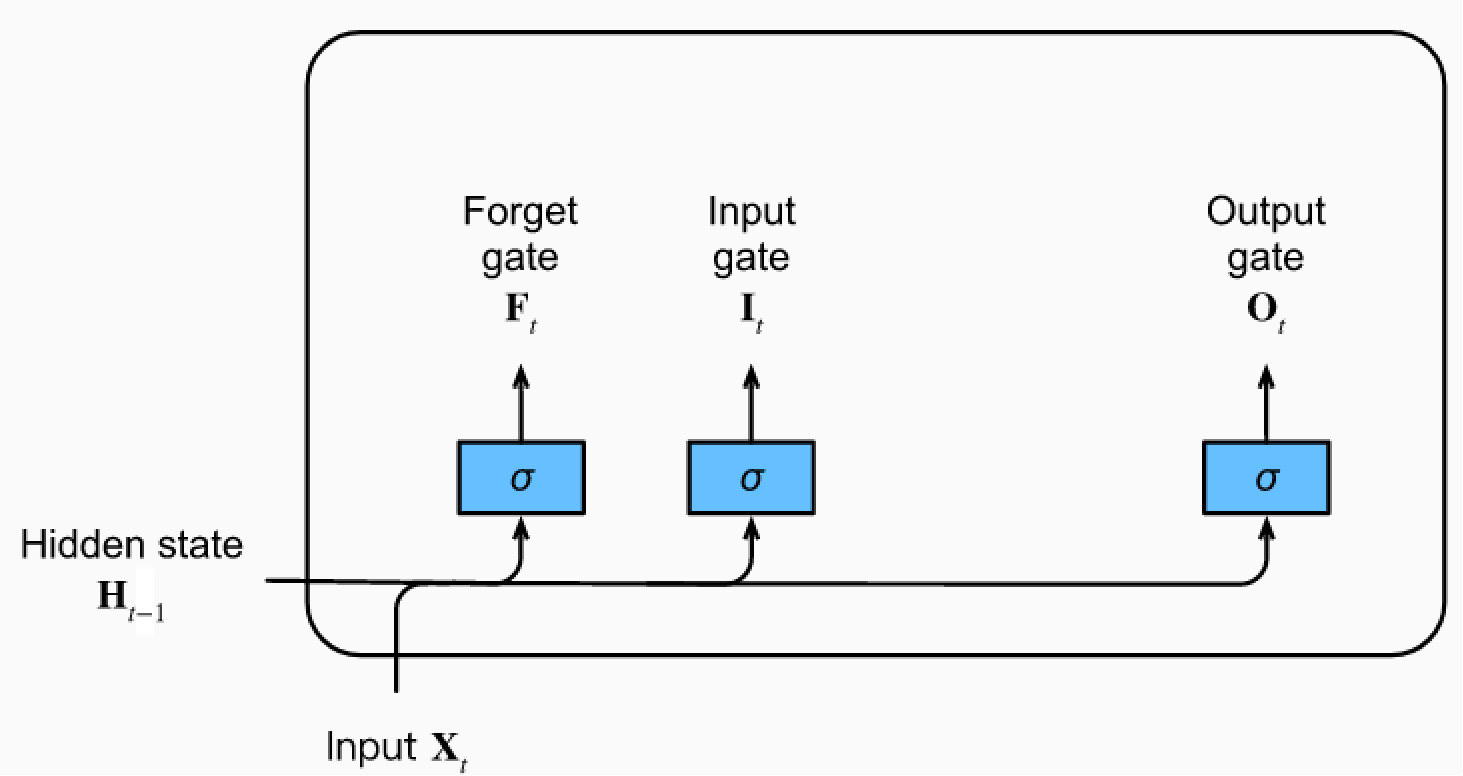
$\mathbf{H}_{t\text{-}1}$ $[n \times h]$

$\mathbf{R}_t$, $\mathbf{Z}_t$ $[n \times h]$

$\mathbf{W}_{xi}$,$\mathbf{W}_{xf}$,$\mathbf{W}_{xo}$ $[d \times h]$

$\mathbf{W}_{hi}$,$\mathbf{W}_{hf}$,$\mathbf{W}_{ho}$ $[h \times h]$

$\boldsymbol{b}_i$, $\boldsymbol{b}_f$, $\boldsymbol{b}_o$ $[1 \times h]$



$$\mathbf{I}_t = \sigma(\mathbf{X}_t \, \mathbf{W}_{xi} + \mathbf{H}_{t\text{-}1} \, \mathbf{W}_{hi} + \boldsymbol{b}_i)$$
$$\mathbf{F}_t = \sigma(\mathbf{X}_t \, \mathbf{W}_{xf} + \mathbf{H}_{t\text{-}1} \, \mathbf{W}_{hf} + \boldsymbol{b}_f)$$
$$\mathbf{O}_t = \sigma(\mathbf{X}_t \, \mathbf{W}_{xo} + \mathbf{H}_{t\text{-}1} \, \mathbf{W}_{ho} + \boldsymbol{b}_o)$$

(Hochreiter & Schmidhuber, 1997)

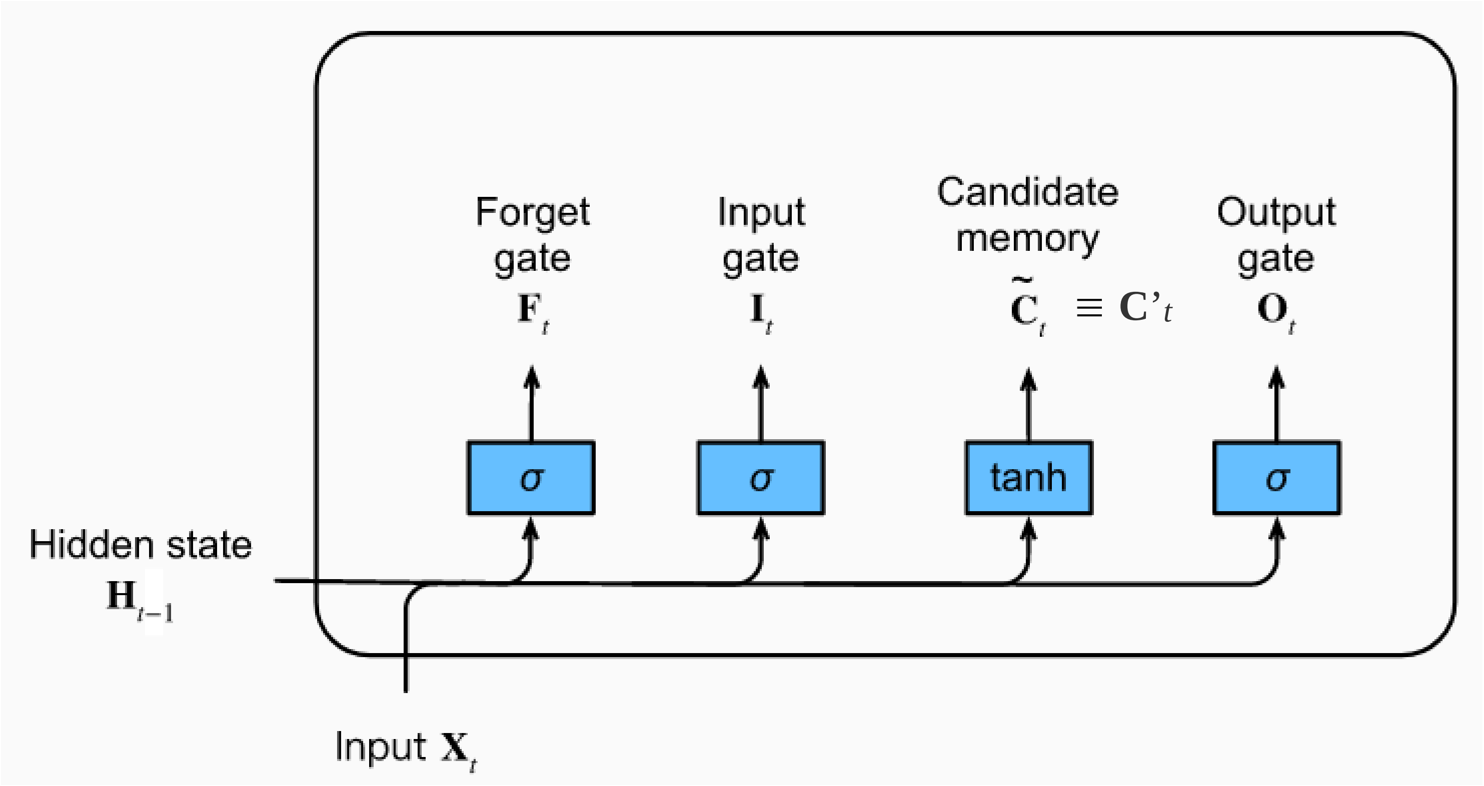
Candidate memory cell

$\mathbf{W}_{xc}$ $[d \times h]$

$\mathbf{W}_{hc}$ $[h \times h]$

$\boldsymbol{B}_c$ $[1 \times h]$

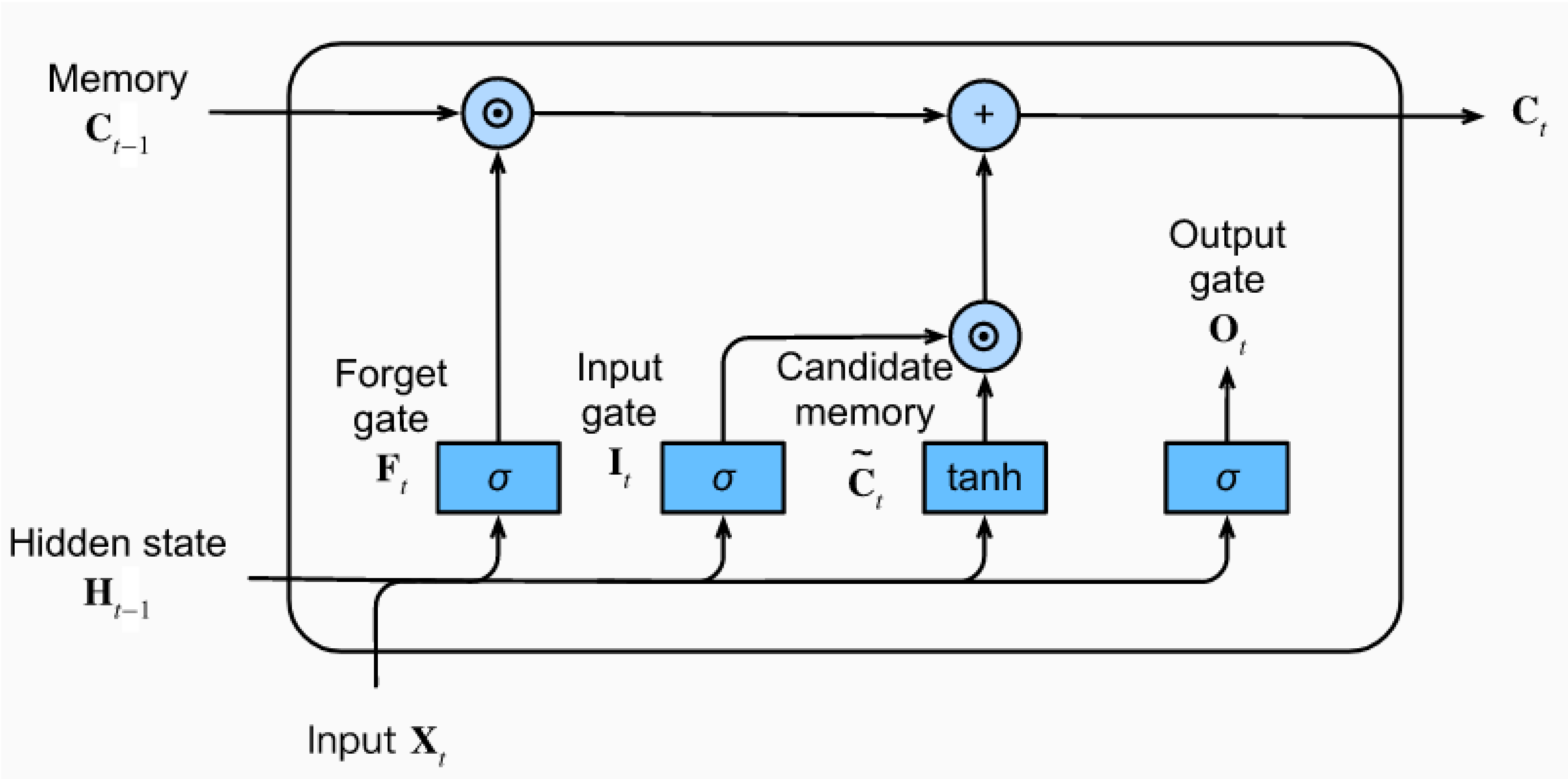
$\mathbf{C'}_t$ $[n \times h]$



$$\mathbf{C'}_t = \tanh(\mathbf{X}_t\mathbf{W}_{xc} + \mathbf{H}_{t\text{-}1}\mathbf{W}_{hc} + \boldsymbol{b}_c$$

computation similar to the 3 gates described above, but using a tanh function

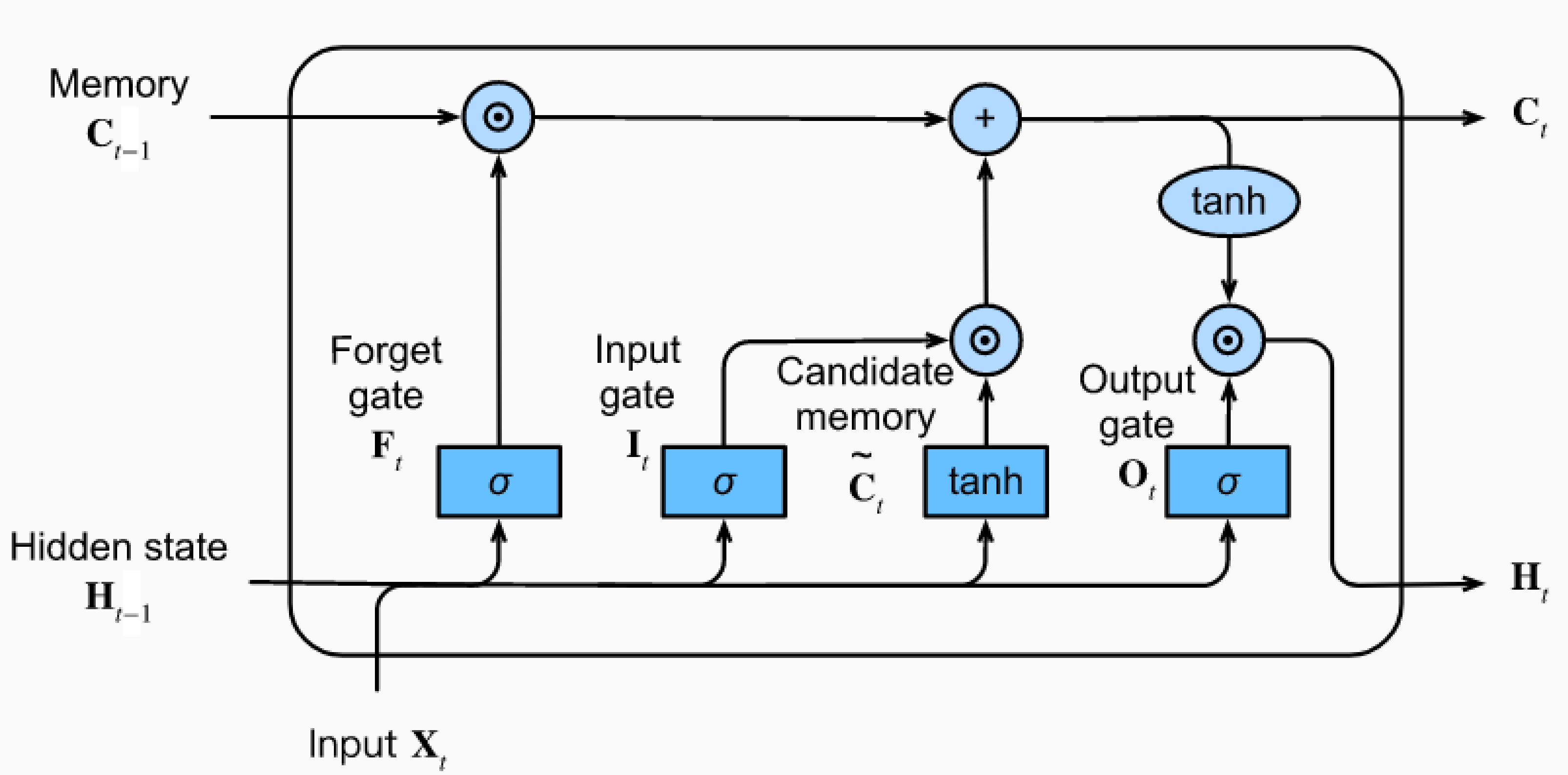
Memory cell



LSTM has 2 parameters: $\mathbf{I}_t$ governs how much of new data we take via $\mathbf{C'}_t$ and $\mathbf{F}_t$ determines how much of the old memory content $\mathbf{C}_{t-1}$ we retain.

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \mathbf{C'}_t$$

Hidden states



Hidden state is $\mathbf{O}_t$-gated version of the tanh of the memory cell:

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t)$$

Complete LSTM dynamics

Input gates

$$\boldsymbol{g}(t)=\sigma(\mathbf{U}^{\text{inp}}\boldsymbol{x}(t)+\mathbf{W}^{\text{inp}}\boldsymbol{h}(t-1)+\mathbf{b}^{\text{inp}})$$

Forget gates

$$\boldsymbol{f}(t)=\sigma(\mathbf{U}^{\text{fgt}}\boldsymbol{x}(t)+\mathbf{W}^{\text{fgt}}\boldsymbol{h}(t-1)+\boldsymbol{b}^{\text{fgt}})$$

Memory cell state

$$\boldsymbol{s}(t)=\boldsymbol{f}(t)\odot\boldsymbol{s}(t-1)+\boldsymbol{g}(t)\odot\sigma(\mathbf{U}^{\text{fgt}}\boldsymbol{x}(t)+\mathbf{W}^{\text{fgt}}\boldsymbol{h}(t-1)+\boldsymbol{b})$$

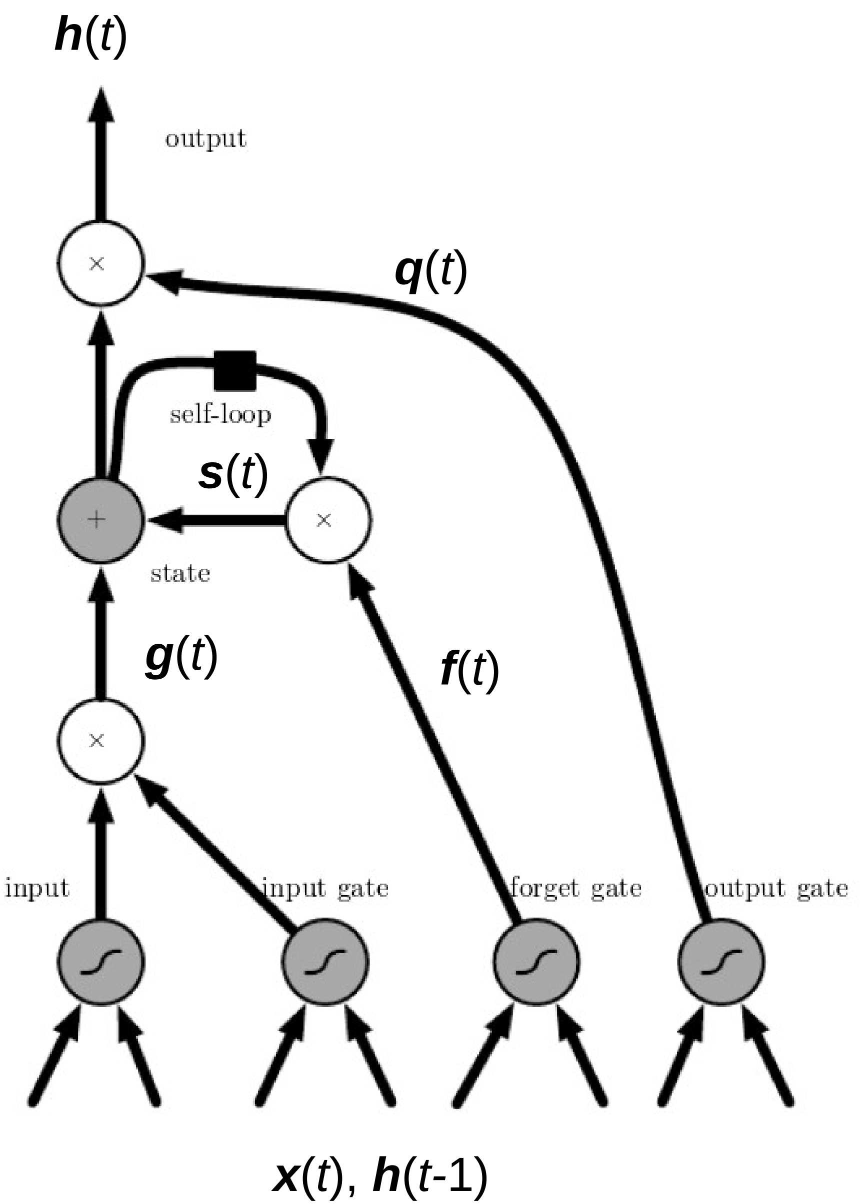
Output gates

$$\boldsymbol{q}(t)=\sigma(\mathbf{U}^{\text{out}}\boldsymbol{x}(t)+\mathbf{W}^{\text{out}}\boldsymbol{h}(t-1)+\boldsymbol{b}^{\text{out}})$$

LSTM state output

$$\boldsymbol{h}(t)=\tanh(\boldsymbol{s}(t))\odot\boldsymbol{q}(t)$$

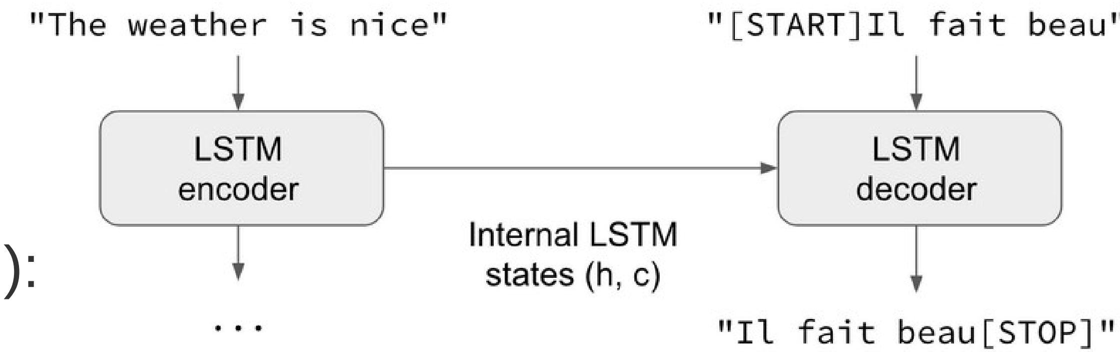
$\odot$ = element-wise multiplication



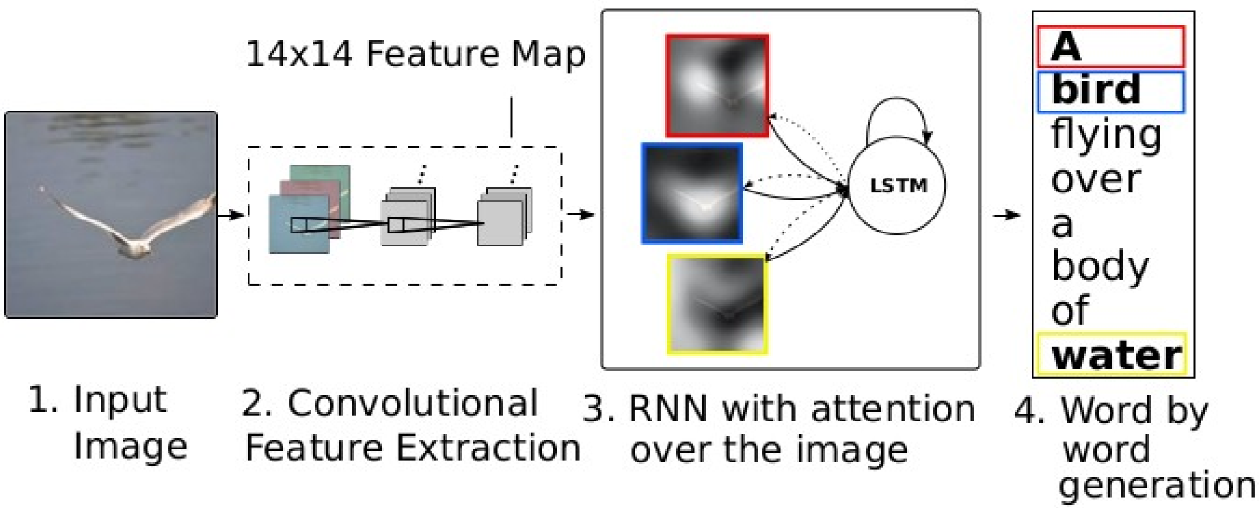
(Goodfellow et al, 2015) - using different symbols

Applications of LSTM

- unconstrained handwriting recognition
- speech recognition
- music generation
- parsing (PoS tagging)
- machine translation (seq2seq):
- image captioning
- …
- new:
  - attention mech.
  - bidirect. models



(Sutskever et al, 2014)



(Xu et al, 2016)

LSTM summary

- Using trained gates, it introduces self-loops to produce paths where the gradient can flow for long (neither exploding nor vanishing)
- the time scale of integration can be changed dynamically
- the cell state is the core of the LSTM, controlled by the gates
- Trainable with various methods, e.g. SGD, 2^{nd} order methods, Nesterov gradient, …
- Various variants found useful, clipping the gradient, e.g. element-wise (Mikolov, 2012); or by L2 norm (Pascanu et al, 2013).
- Can be combined with autoencoders

Attention mechanism and transformers

- Transformer – a new category of NN models (successor of CNNs and RNNs) (Vaswani et al., 2017)
- Attention – the core idea behind the transformers, originated in the NLP context of sequence-to-sequence applications, like machine translation (Bahdanau et al., 2014).
- Transformers are currently widely used in various AI domains:
  - in NLP – Transformed-based pretrained models (BERT, RoBERTa,...) – fine-tuned to concrete language tasks
  - speech recognition, reinforcement learning tasks
  - vision tasks (image recognition, object detection, semantic segmentation,…)

Queries, keys and values

- Consider the database $D = \{(\boldsymbol{k}_1, \boldsymbol{v}_1),(\boldsymbol{k}_2, \boldsymbol{v}_2),\ldots,(\boldsymbol{k}_m, \boldsymbol{v}_m)\}$, of key-value pairs. For a query $\boldsymbol{q}$, we can define attention as

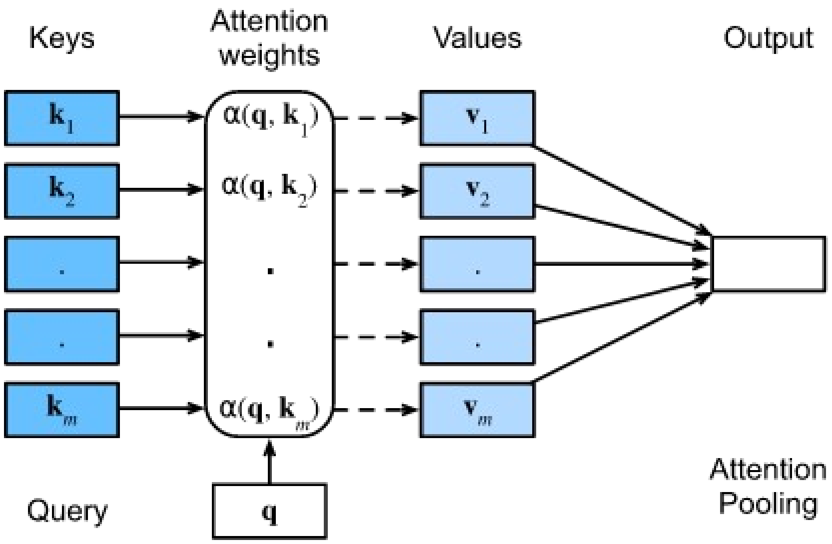
$$Attention(\boldsymbol{q}, D) = \sum_{i=1}^{m} \alpha(\boldsymbol{q}, \boldsymbol{k}_i)\boldsymbol{v}_i$$

  where $\alpha(\boldsymbol{q}, \boldsymbol{k}_i) \in R$ are scalar attention weights.

- Attention pooling: the attention over *D* generates a linear combination of values in *D.*

  $\alpha_1 + \ldots + \alpha_m = 1$ and all $\alpha_i \geq 0$ => convex combination

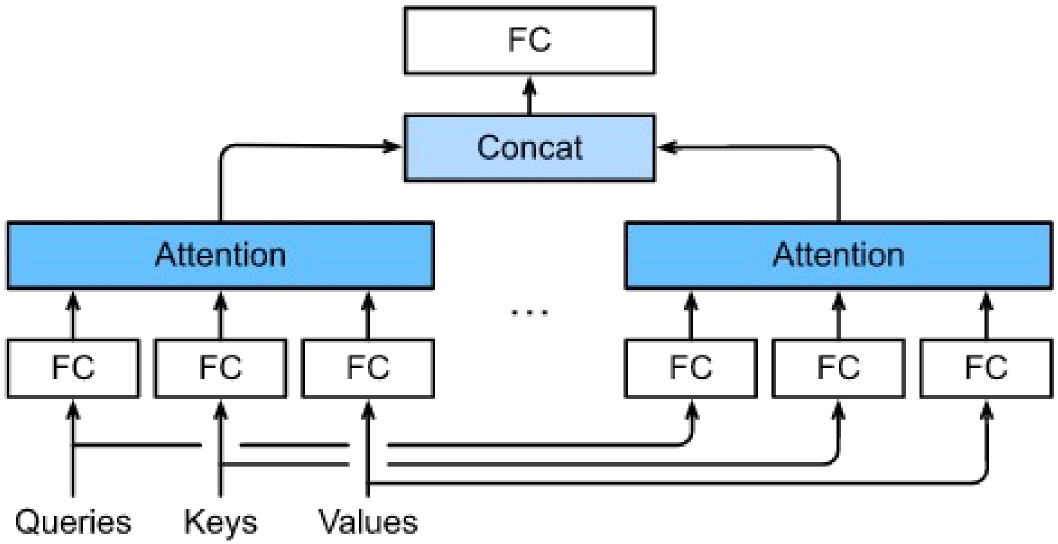
$$\alpha(\boldsymbol{q}, \boldsymbol{k}_i) = softmax\left(\frac{(\boldsymbol{q}^T \boldsymbol{k}_i)}{\sqrt{d}}\right)$$



(Zhang et al, 2020)

Multi-head attention

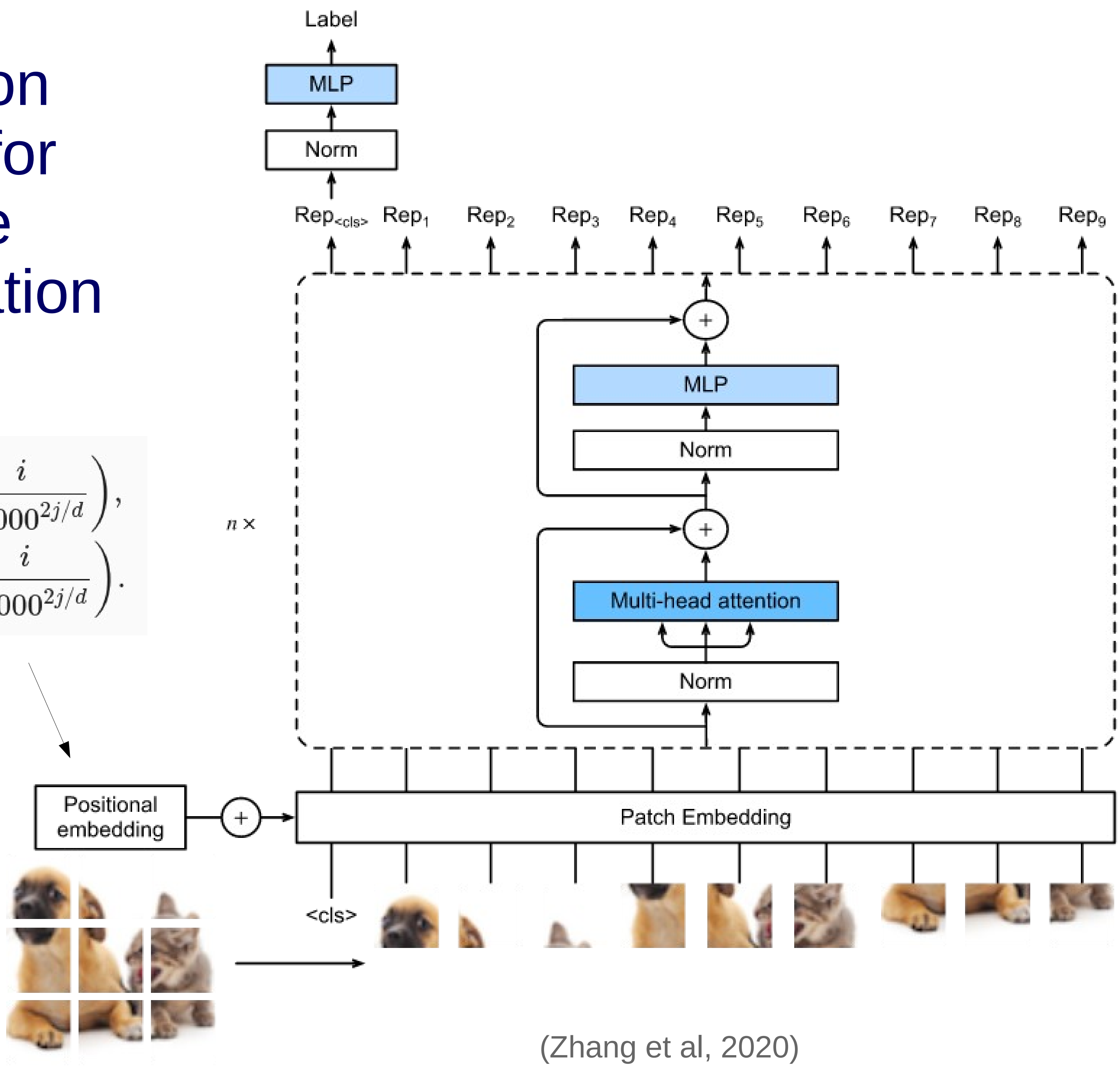
- given the same set of *queries*, *kyes*, and *values* it may be useful to combine knowledge e.g. capturing dependencies of various ranges (shorter, longer) within a sequence (→ different representation subspaces)
- Each head $\boldsymbol{h}_i = f(\mathbf{W}_i^{(q)}\boldsymbol{q}, \mathbf{W}_i^{(k)}\boldsymbol{k}, \mathbf{W}_i^{(v)}\boldsymbol{v})$
- $\mathbf{W}_i^{(x)}$ = learnable parameters



(Zhang et al, 2020)

Attention model for image classification

$$p_{i,2j} = \sin\left(\frac{i}{10000^{2j/d}}\right),$$
$$p_{i,2j+1} = \cos\left(\frac{i}{10000^{2j/d}}\right).$$



(Zhang et al, 2020)