ELEC-C5220
Lecture 10:
Computational cost in Deep Learning

Machine learning in information technology

Aalto University
School of Electrical Engineering

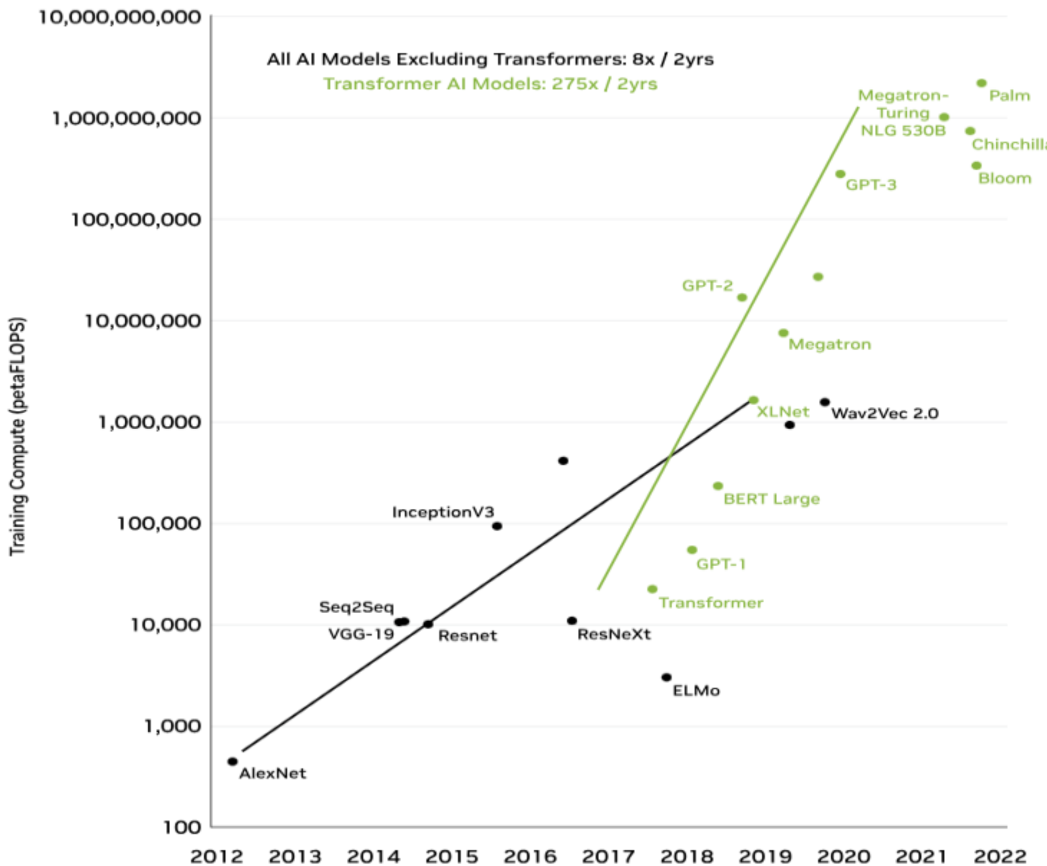
Lauri Juvela

21.3.2024

Lecture 10 content

- **Computation in deep learning models**
  - Parameter counting
  - Operation counting – FLOPs and MACs
  - Parallel and sequential computation
- **Recap on model architectures**
  - Linear layers, fully connected networks (MLPs)
  - Convolution networks
  - Recurrent networks
  - Attention

Computational resources in AI



Quantifying compute cost

Theoretical: depends on assumptions, not implementation

- Traditional big-O complexity analysis
- Parameter counting
- Operation counting (FLOPs and MACs)

Empirical: depends on specific implementation and hardware

- Profiling
- Wall-clock CPU/GPU hours (or years depending on the scale)
- Energy use kWh

Parameter counting

- How many floating-point parameters does and NN model have?
- Parameters are usually tensors, need to count tensor sizes
- Useful proxy for computational complexity, easy to calculate
- Parameter count is sometimes the same as operation count, but not always
- RNNs, Convolutions and Attention-based models share parameters over time

Operation counting – FLOPs and MACs

- **FLOPs – Floating point operations**
    - Scalar multiplication and addition cost ~FLOP
    - Division is more expensive, depends on implementation
    - Simple elementwise non-linearity cost ~FLOP
    - Exponentials are more expensive, (incl. tanh and sigmoids)

Operation counting – FLOPs and MACs

- **MACs - Multiply and accumulate operations**
  - Many processors can multiply and accumulate in a single processor cycle
  - Many DSP applications (i.e., filtering) rely on MAC operations
  - Matrix multiplication is pure MAC

OP counting: matrix multiplication

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ae + be \\ cf + df \end{bmatrix}$$

- **2 x 2 matrix dot product with 2 x 1 vector**
- **How many multiplications? (FLOPs)**
- **How many additions? (FLOPs)**
- **How many MACs?**

OP counting: matrix multiplication

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$$

$$y_i = \sum_j a_{i,j} x_j + b_i$$

$\mathbf{y} \in \mathbb{R}^M$ $\quad$ $\mathbf{A} \in \mathbb{R}^{M \times N}$

$\mathbf{b} \in \mathbb{R}^M$ $\quad$ $\mathbf{x} \in \mathbb{R}^N$

- **Linear layer in neural net (actually Affine)**
- **How many multiplications? (FLOPs)**
- **How many additions? (FLOPs)**
- **How many MACs?**

Tensor parameter counts

- **Scalar – 0D Tensor** $x \in \mathbb{R}$ $()$
- **Vector – 1D Tensor** $\mathbf{x} \in \mathbb{R}^D$ (D)
- **Matrix – 2D Tensor** $\mathbf{X} \in \mathbb{R}^{N \times M}$ (N, M)
- **Parameter count of a tensor variable is the product of its dimensions**

Tensor parameter counts

- **3D Tensor, 1D Convolution kernel**

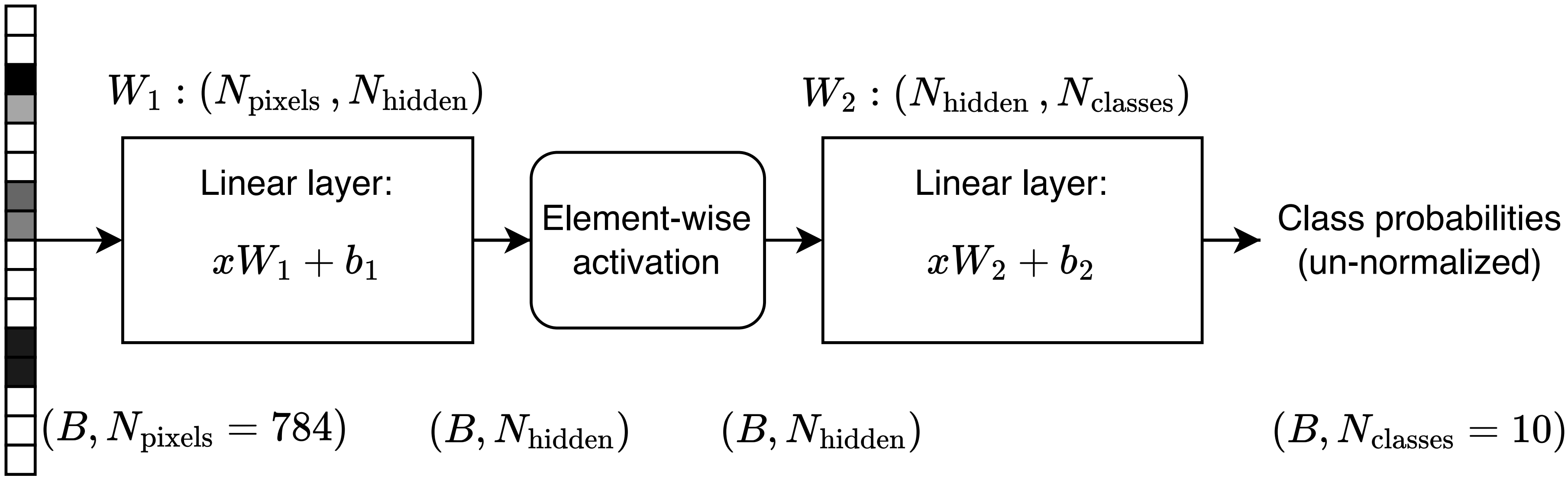
$$x \in \mathbb{R}^{(C_{\text{out}} \times C_{\text{in}} \times K)} \qquad (C_{\text{out}}, C_{\text{in}}, K)$$

- **4D Tensor, color images**

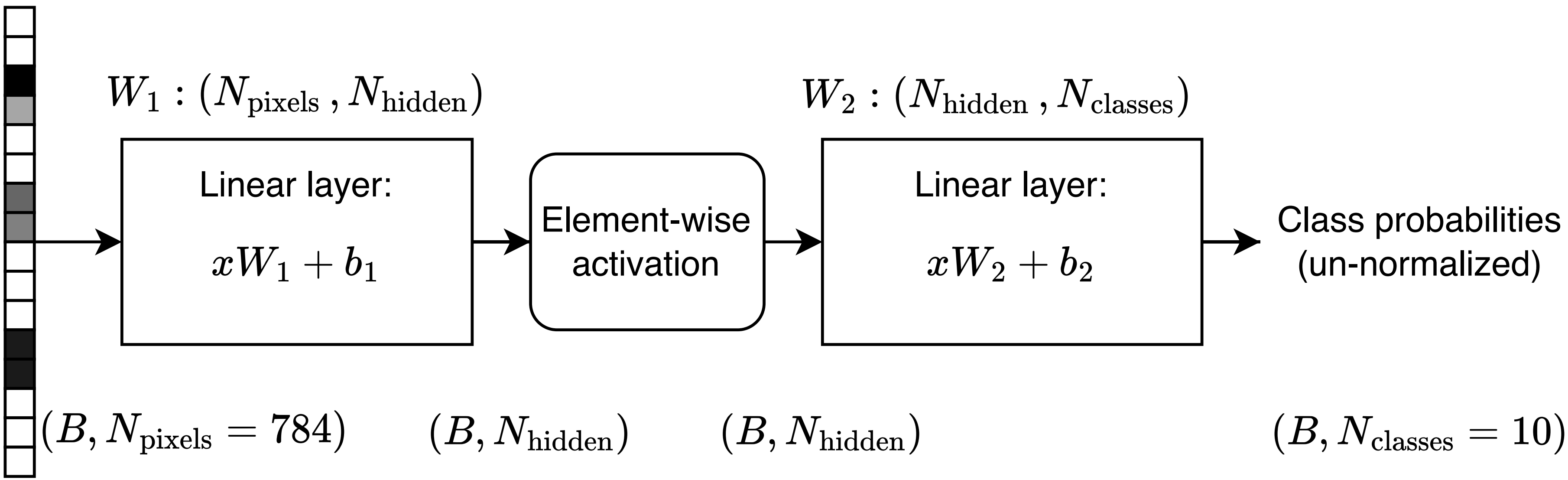
$$x \in \mathbb{R}^{(C_{\text{out}} \times C_{\text{in}} \times H \times W)} \quad (C_{\text{out}}, C_{\text{in}}, H, W)$$

- **Parameter count of a tensor variable is the product of its dimensions**

DNN Classifier for MNIST digits

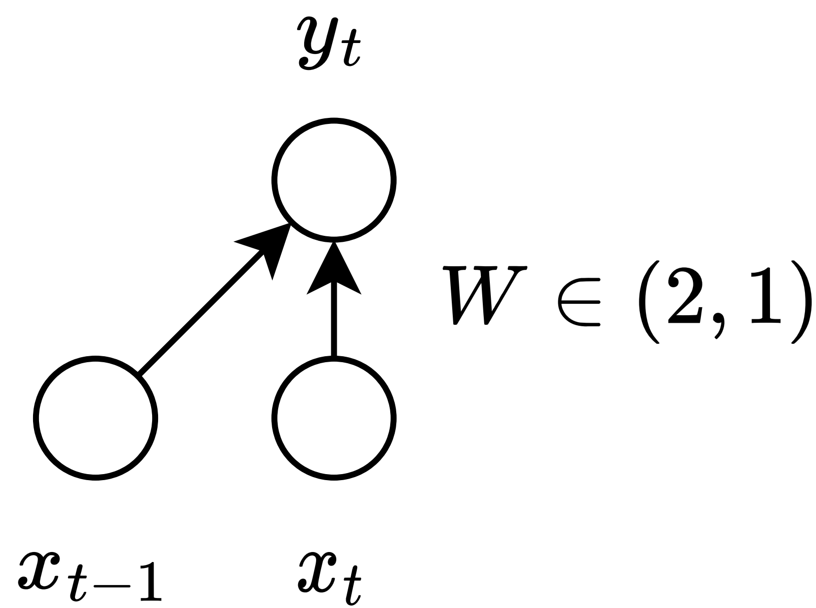


DNN Classifier –
How many parameters?



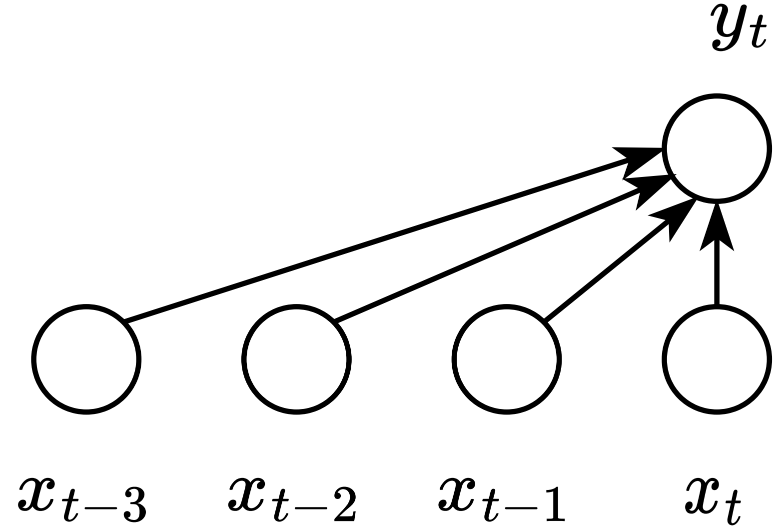
Minimal convolution net

- At each time-step, the output depends on the input values at current and previous time-steps
- Same dependency for all time values: weight sharing across time



Convolution is filtering

- Input dimension – 4 time steps
- Output dimension – 1 time step
- Complexity: filter length x input length MACs
- Typically filters are much shorter than input sequences!

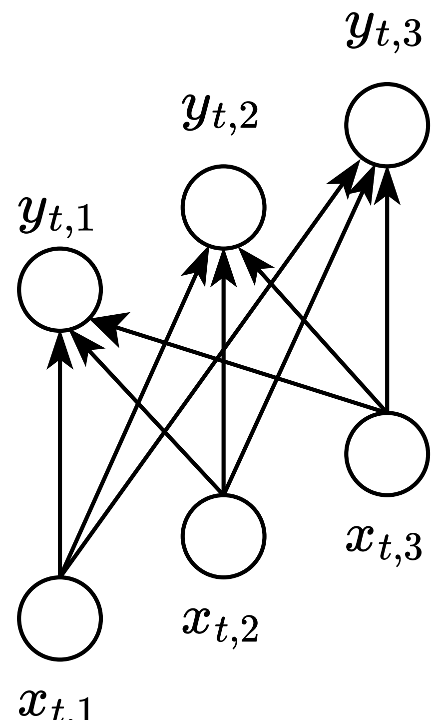


$$W \in (4,1)$$

$$y_t = \sum_{i=0}^{3} W_{i,0} x_{t-i}$$

Convolution is fully connected

- **Channels in CNNs are fully connected**
- **Kernel width = 1**
- **Input dim. = input channels**
- **Output dim. = output channels**
- **Complexity: prod(W.shape) * T**

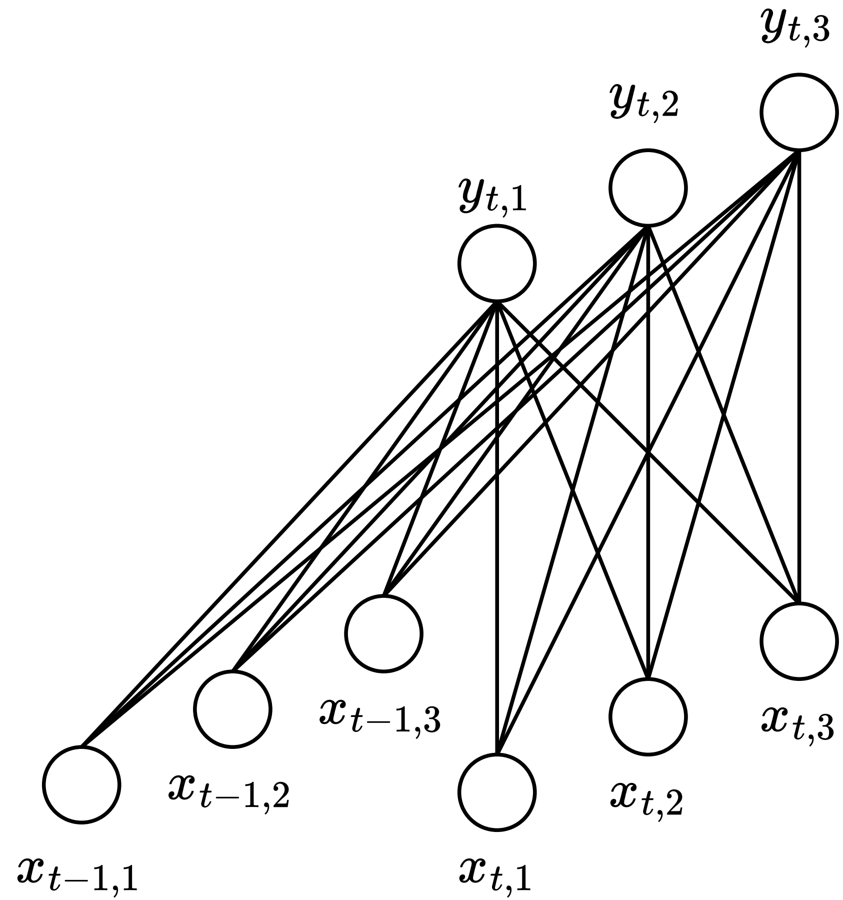


$$W \in (3, 3)$$

$$y_{t,j} = \sum_{i=1}^{3} W_{j,i} x_{t,i}$$

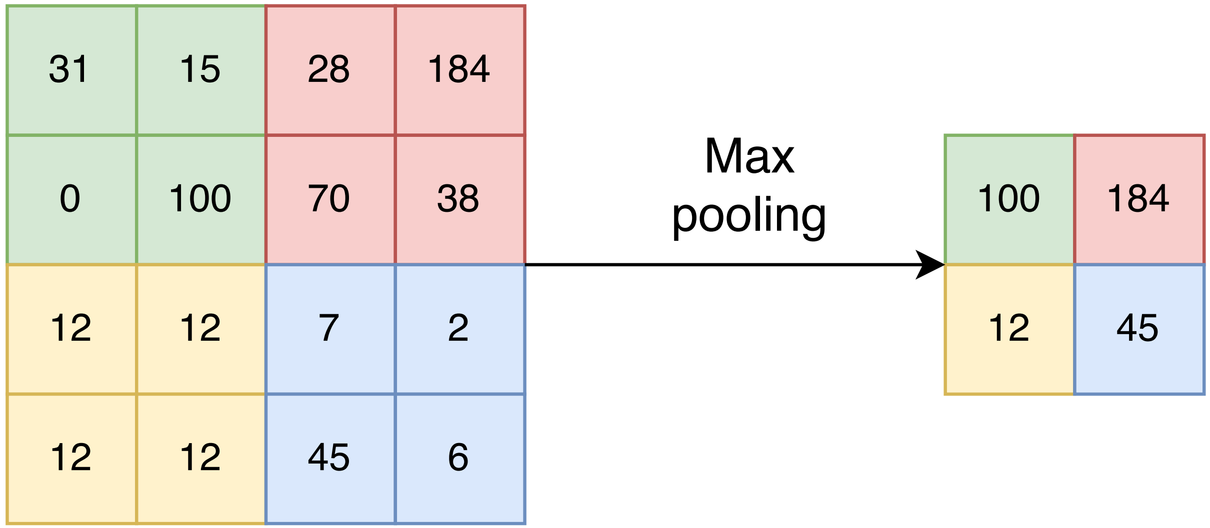
Convolution layer

- **Fully connected over channels**
- **Fully connected over kernel width in time**
- **Apply the compute output values for the whole sequence**
- **Complexity: prod(W.shape) * T**



$$W \in (3, 3, 2)$$

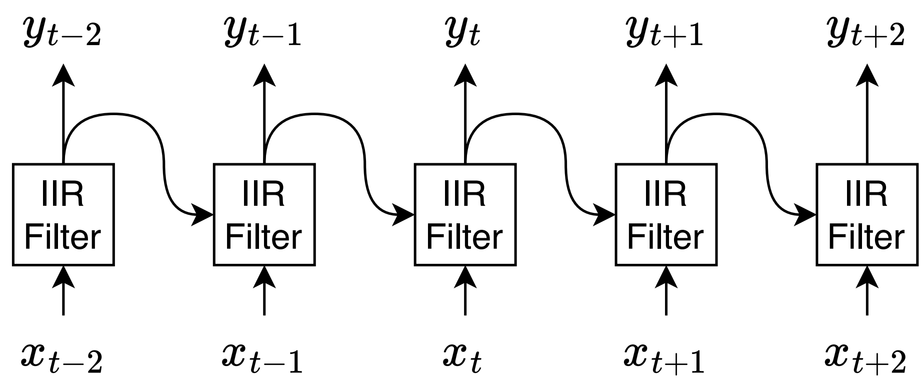
Max pooling and strided ops



- **Sliding window size (2, 2)**
- **Stride determines the downsampling factor, (2,2) in this case**
- **Complexity**

First order IIR unrolled in time

- For each time step, the filter output depends on the current input and previous state of the filter
- Apply the same operation on every time step (weight sharing)



Recurrent Neural Networks

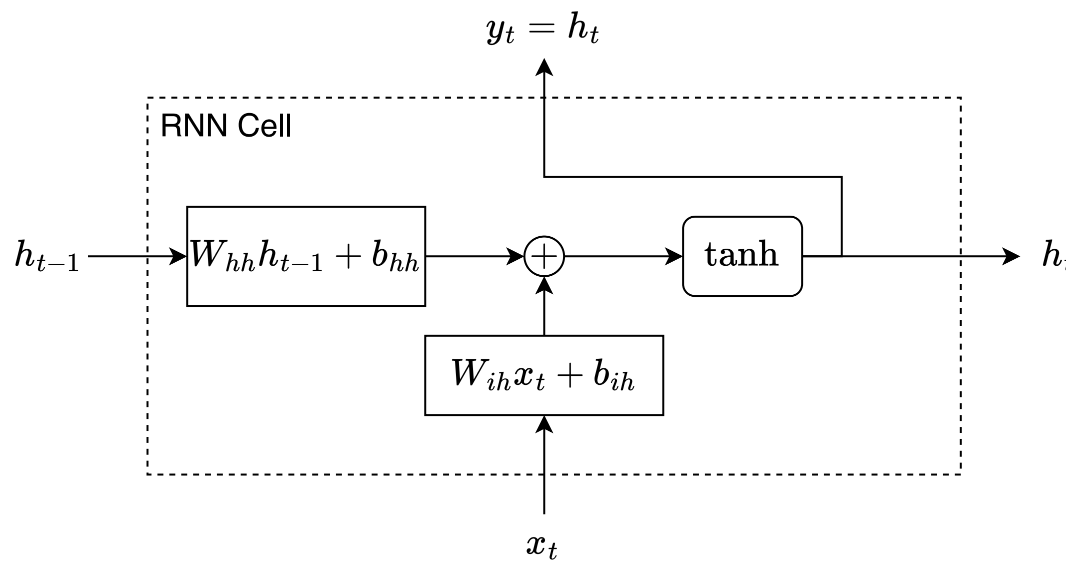
- Neural networks designed for time series processing
- A non-linear analogue of multi-channel first order IIR filters
- RNN output at each time step depends on the current input, the previous state of the RNN (and the network parameters)

$$h_t = f(x_t, h_{t-1}; \theta)$$

Elman RNN

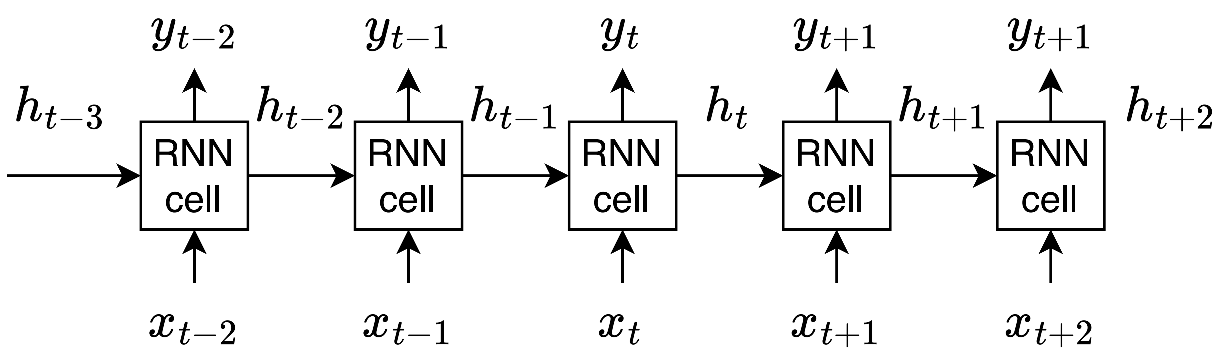
- **Two matrix multiplications per time-step**
- **Complexity: (I x H + H x H) * T**
- **Ignore biases?**
- **Ignore activations?**

$$h_t = \tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh})$$

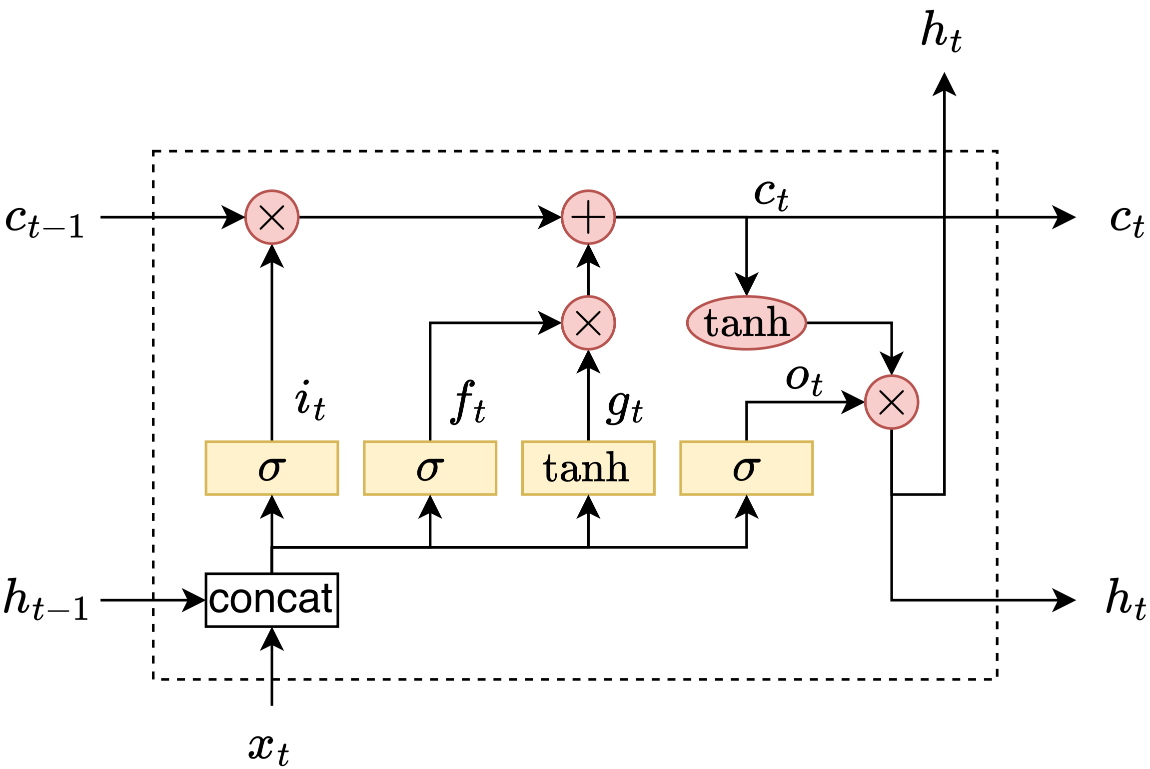


Unrolled RNNs

- **Forward pass requires sequential left-to-right processing**
- **Backward pass requires sequential right-to-left processing, aka backpropagation through time (BPTT)**
- **Forward and backward complexity is usually similar (focus on forward)**



Long Short Term Memory (LSTM)



$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$

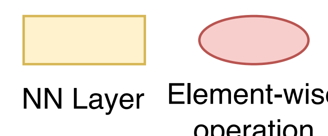
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf})$$

$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$

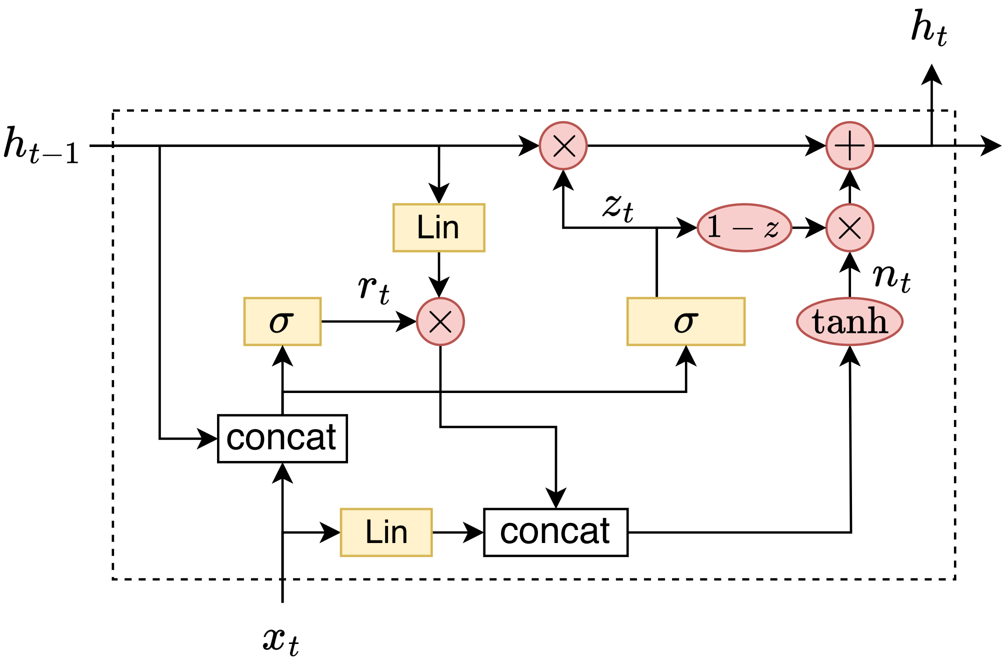
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh(c_t)$$



Aalto University
School of Electrical
Engineering

Gated Recurrent Unit (GRU)



$$r_t = \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr})$$

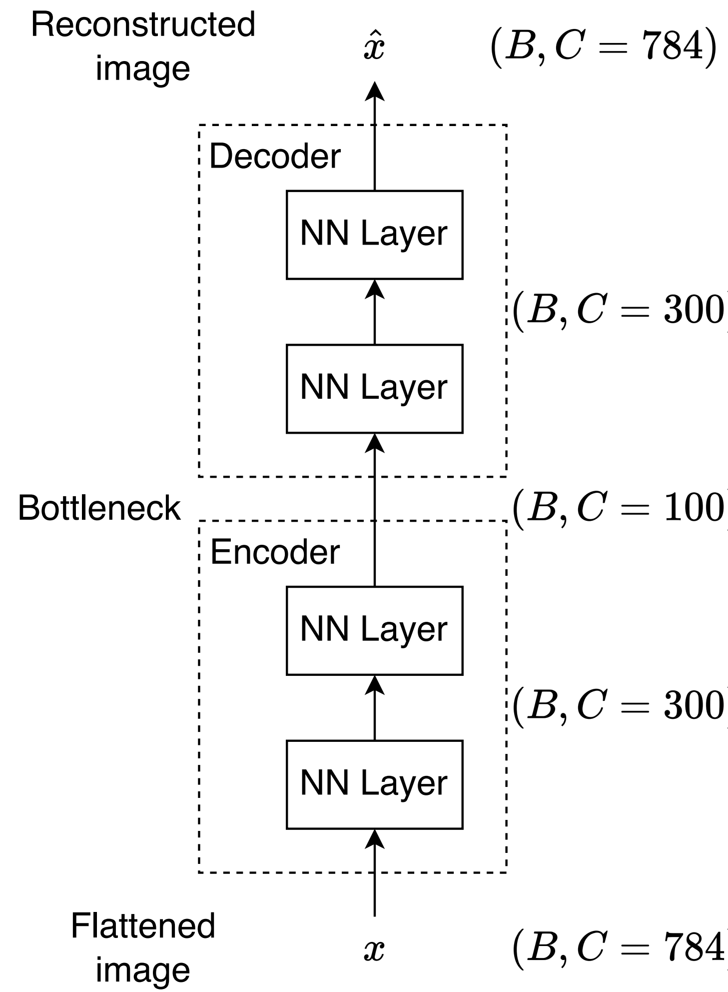
$$n_t = \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn}))$$

$$z_t = \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz})$$

$$h_t = (1 - z_t) \odot n_t + z_t \odot h_{t-1}$$

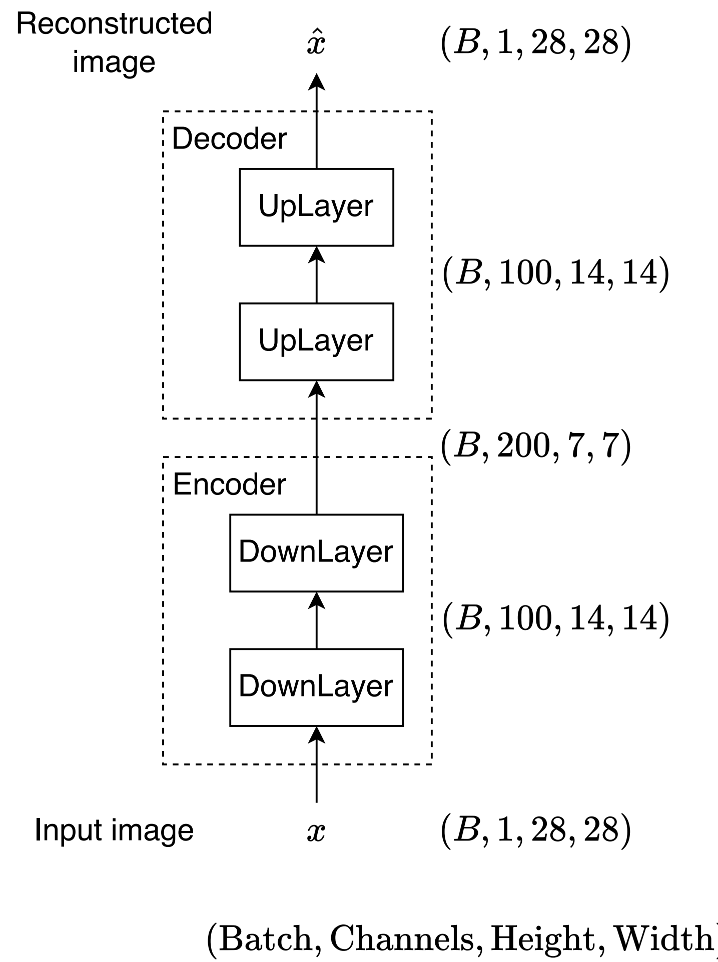
DNN Autoencoder

- **Data compression with neural networks**
- **Encoder reduces data dimensionality**
- **Decoder maps back to orignal data dimension**
- **Fully connected net: parameter count and computation match**



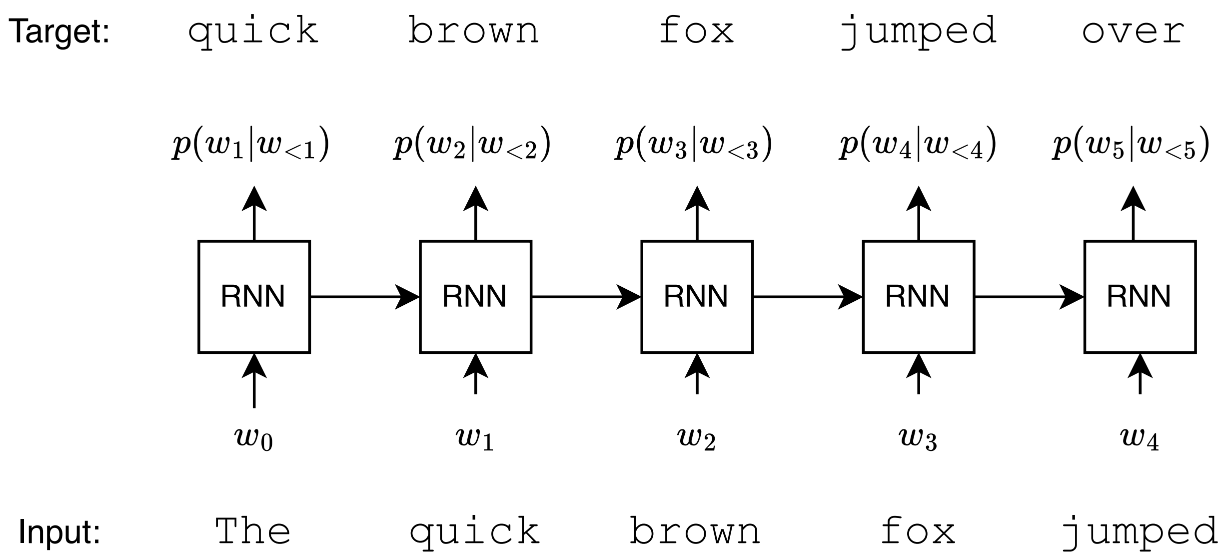
CNN Autoencoder

- **Encoder applies spatial dimensionality reduction by downsampling**
- **Decoder reconstructs the spatial dimensions by upsampling**
- **Convolution net - weight sharing over time**
- **Parameter count & FLOPS vs fully connected?**

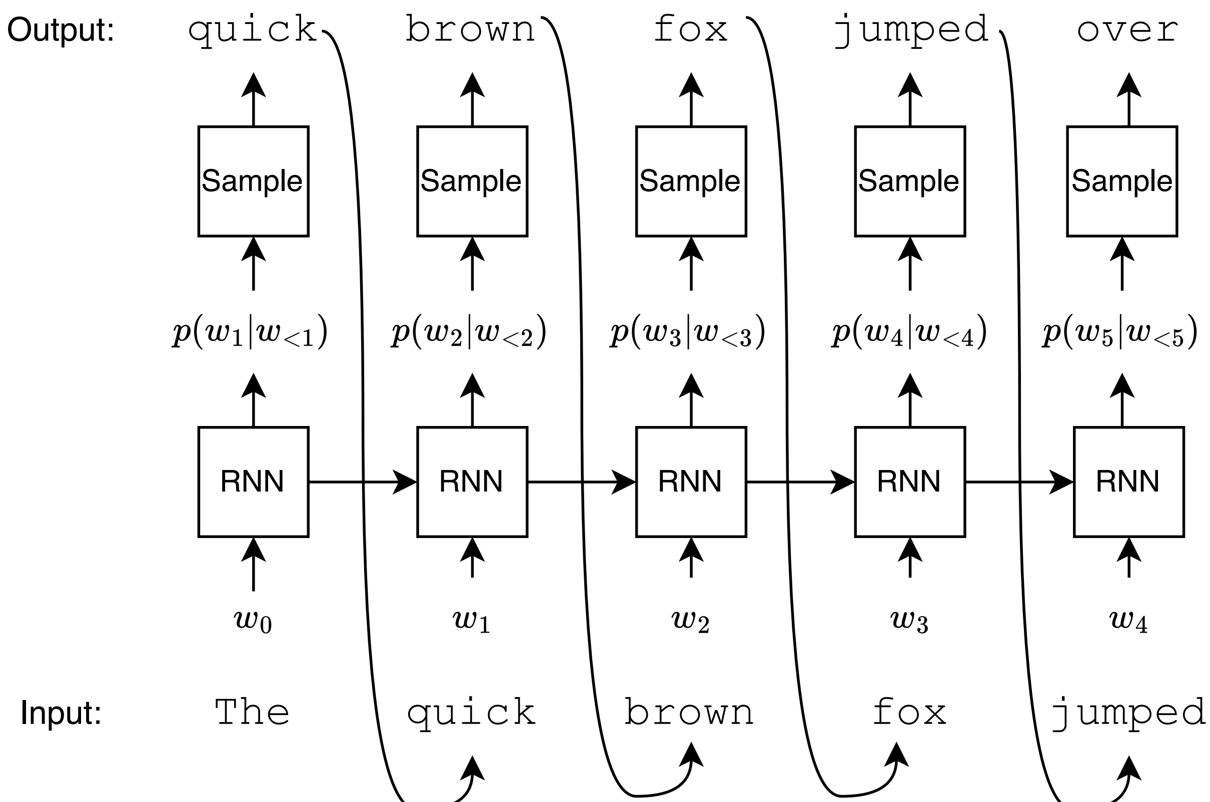


Aalto University
School of Electrical
Engineering

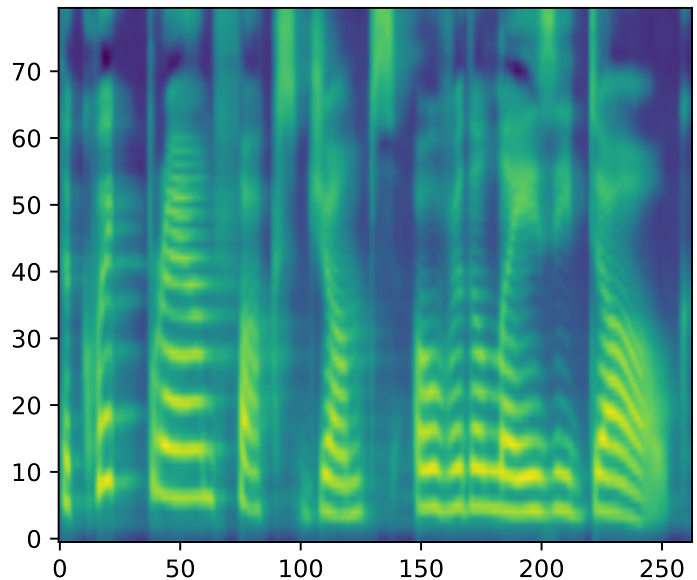
Language model training (full sequence is known)

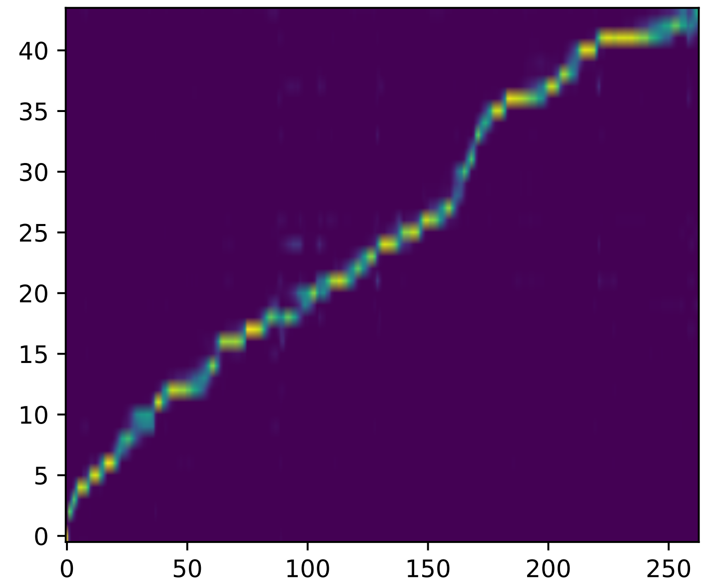


Autoregressive sampling



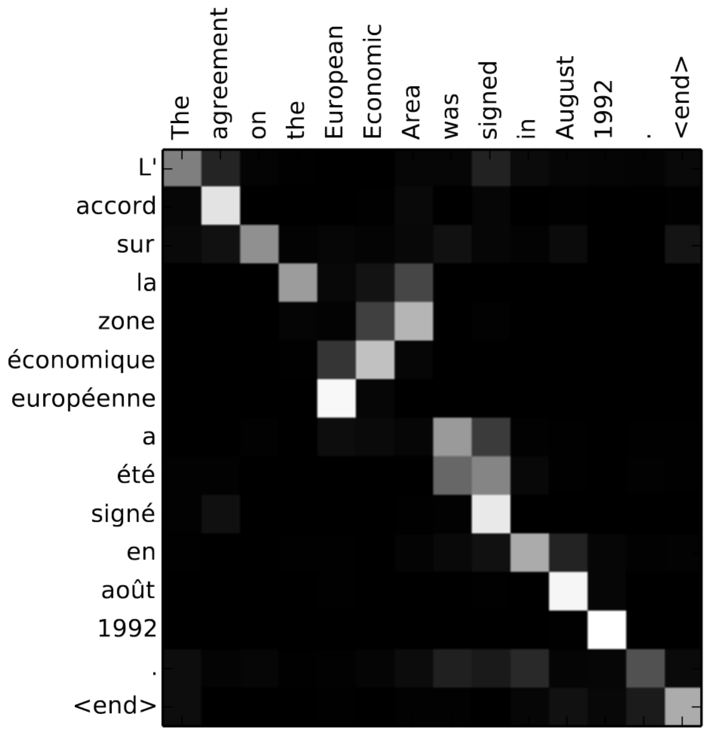
Generated mel-spectrogram and attention plot

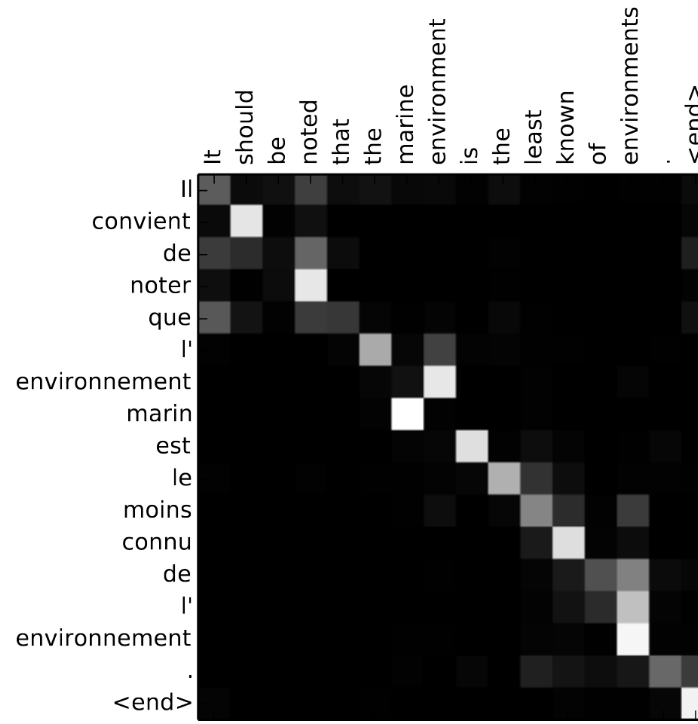




"The quick brown fox jumps over the lazy dog."

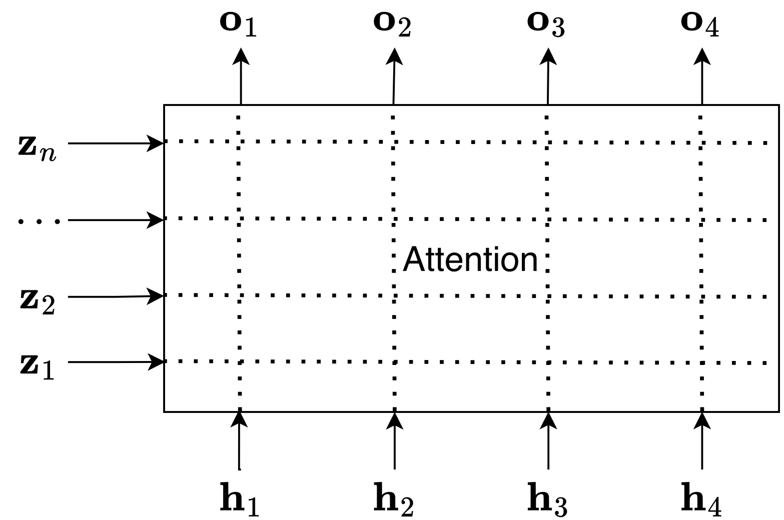
Attention weights visualised (machine translation example)





Attention

- **No parameters! Attention is calculated on activations**
- **Dot product of two d-dim vectors for each time-step pairing**
- **N time-steps on the “cross” sequence**
- **M time-steps on the “self” sequence**
- **Total operations: D x N x M**



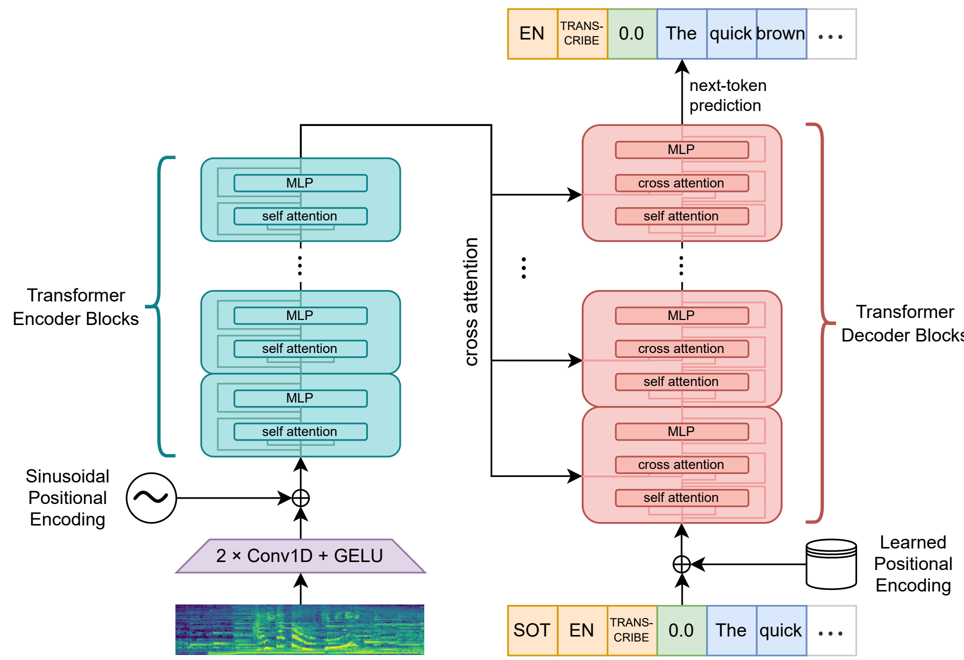
$$\mathbf{z}_j, \mathbf{h}_i \in \mathbb{R}^d$$

$$\alpha_{i,j} = \frac{\exp(\mathbf{z}_j^T \mathbf{h}_i / \sqrt{d})}{\sum_{j'=1}^{n} \exp(\mathbf{z}_{j'}^T \mathbf{h}_i / \sqrt{d})}$$

$$\mathbf{o}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{z_j}$$

Tracking FLOPs in complex systems

- Elementary modules know their complexity for given input size
- Complex modules can ask their submodules for FLOPS counts
- Forward pass already has a suitable recursive logic for FLOPs counting, just include the count in return



Complexity vs. parameter count

Model	Complexity	Parameter count	Time scaling
DNN	$T \times I \times H$	$T \times I \times H$	Fixed
CNN (1D)	$T \times H \times I \times K$	$H \times I \times K$	T
CNN (2D)	$T \times H \times I \times K_\mathrm{h} \times K_\mathrm{w}$	$H \times I \times K_\mathrm{h} \times K_\mathrm{w}$	T
RNN	$T \times (I \times H + H \times H)$	$I \times H + H \times H$	T
Attention	$T_\mathrm{self} \times T_\mathrm{cross} \times H$ $+I \times H + H \times I$	$I \times H + H \times I$	$T_\mathrm{self} \times T_\mathrm{cross}$

T = time, H = hidden channels,
I = input channels, K = kernel width

Lecture 10 summary

- **Computation in deep learning models**
  - Parameter counting
  - Operation counting – FLOPs and MACs
  - Parallel and sequential computation
- **Recap on model architectures**
  - Linear layers, fully connected networks (MLPs)
  - Convolution networks
  - Recurrent networks
  - Attention