Deep Learning

Lecture 8 – Deep time series models 1

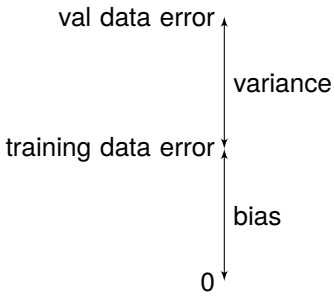
UPPSALA
UNIVERSITET

Carl Andersson
Department of Information Technology
Uppsala University

carl.andersson@it.uu.se

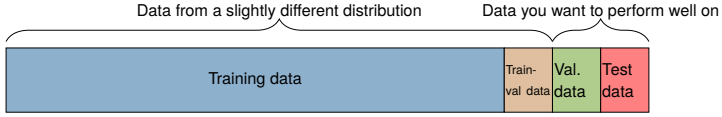
Summary Lecture 7 (I/III) - Training/validation/test data split + mismatched training data

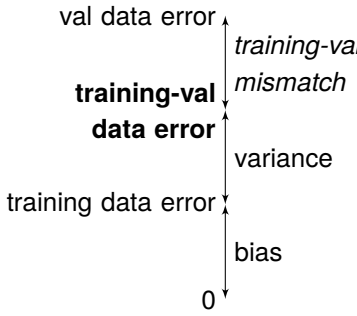
Split data into training, validation and test data.



- **Training data**: Used for **training** the model
- **Validation data**: Used for **chosing between models** for **optimizing hyperparameters**
- **Test data**: Used for **evaulating** the model

Summary Lecture 7 (I/III) - Training/validation/test data split + mismatched training data





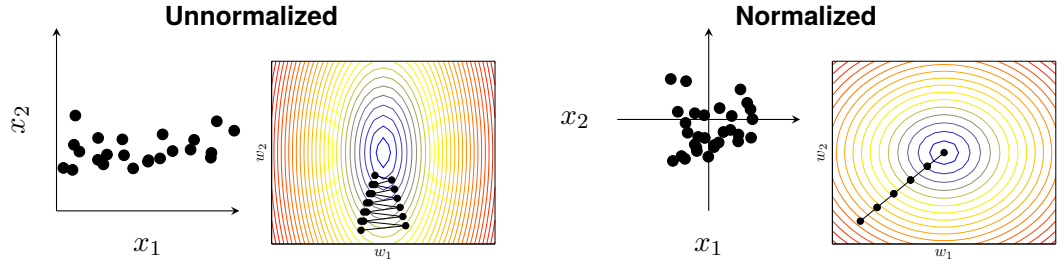
If training data comes from different distribution

- From your training data create a separate **training-val data**
- Training-val data should have the *same* distribution as the training data.
- You do *not* train on the training-val data, only on training data.

Summary Lecture 7 (II/III)
Batch/input normalization training inputs

Input normalization

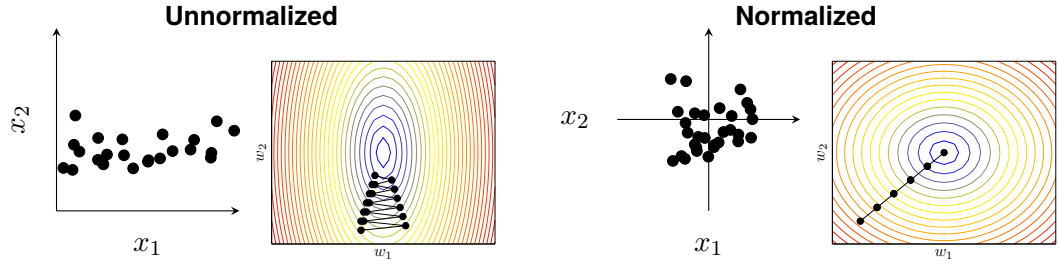
Normalize all input data *before* training.



Summary Lecture 7 (II/III)
Batch/input normalization training inputs

Input normalization

Normalize all input data *before* training.



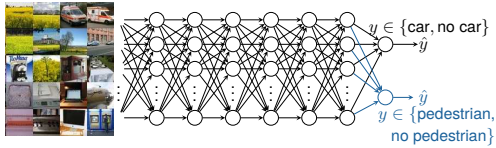
Batch normalization

Normalize each mini-batch in each layer *during* training.

- Makes the **loss landscape more smooth** and easier to optimize.
- Can use **higher learning rate** without getting problem with vanishing or exploding gradients.
- Has a slight **regularizing effect**

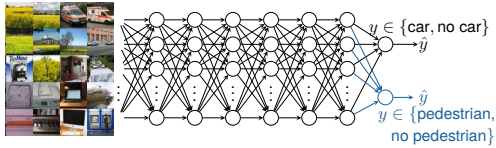
Summary Lecture 7 (III/III) - Learning from multiple tasks

Multitask learning: Train one network to solve multiple tasks



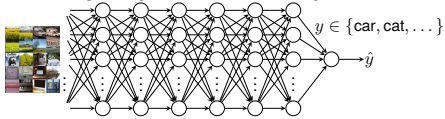
Summary Lecture 7 (III/III) - Learning from multiple tasks

Multitask learning: Train one network to solve multiple tasks

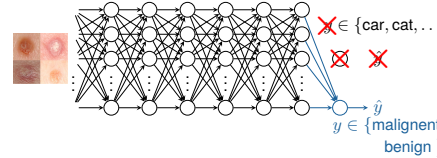


Transfer learning: Fine-tune a network trained on a different task than the one that you are interested in.

Task A: Image recognition $\approx 10\,000\,000$ images

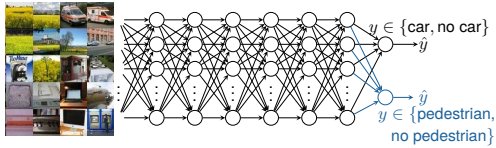


Task B: Skin cancer detection $\approx 100\,000$ images

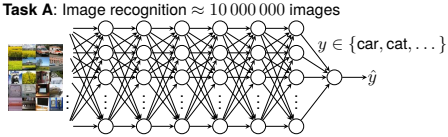


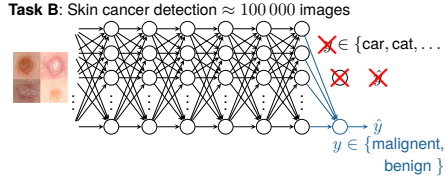
Summary Lecture 7 (III/III) - Learning from multiple tasks

Multitask learning: Train one network to solve multiple tasks



Transfer learning: Fine-tune a network trained on a different task than the one that you are interested in.

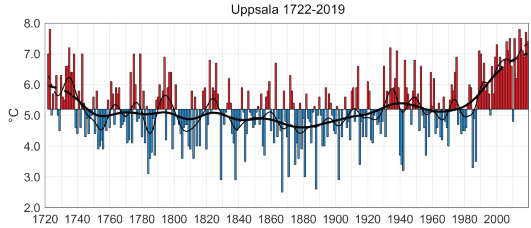




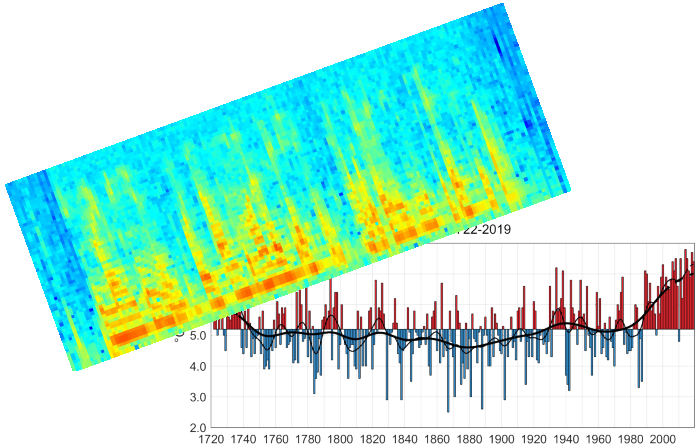
Self-supervised learning: Generate supervised tasks from *unlabeled* data and pretrain on these supervised tasks.

The sequential problem

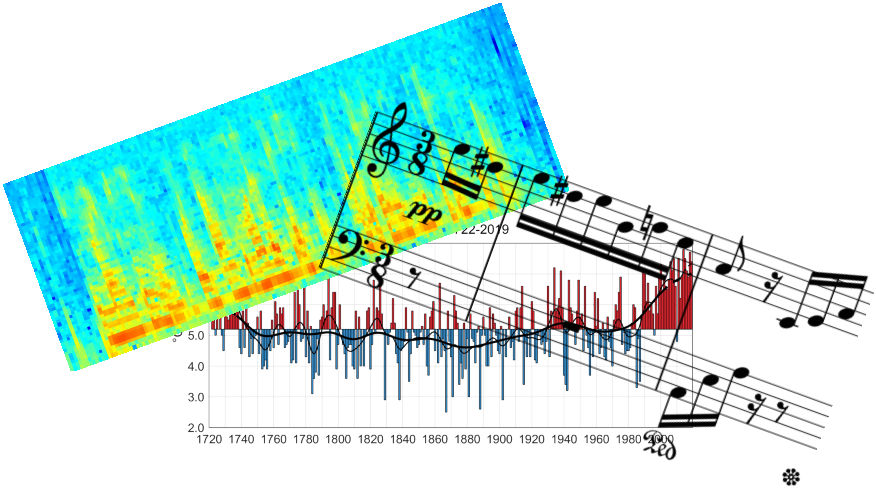
The sequential problem



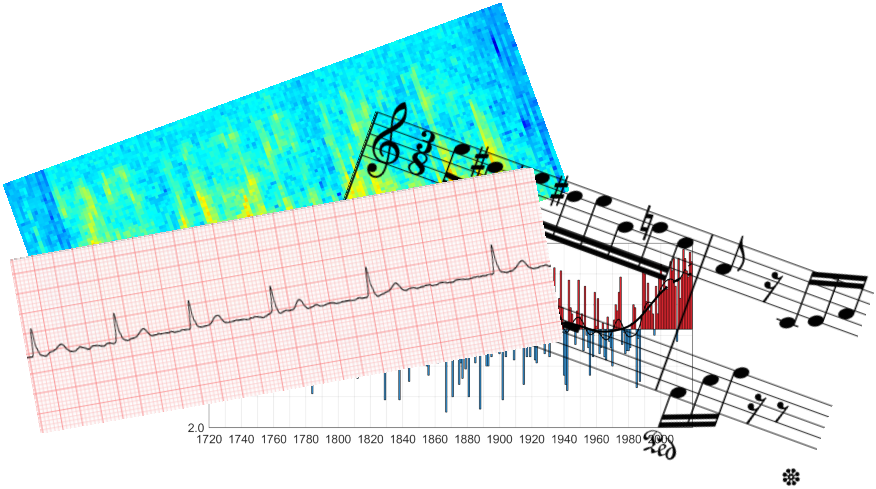
The sequential problem



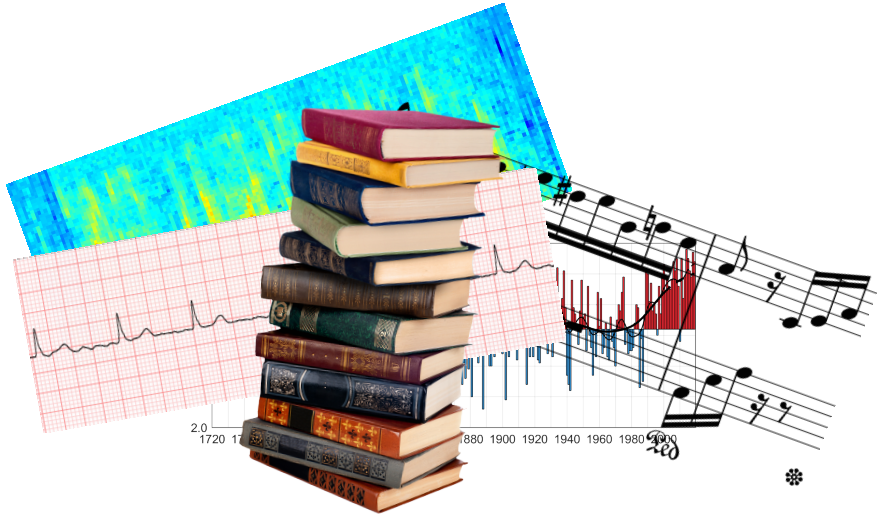
The sequential problem



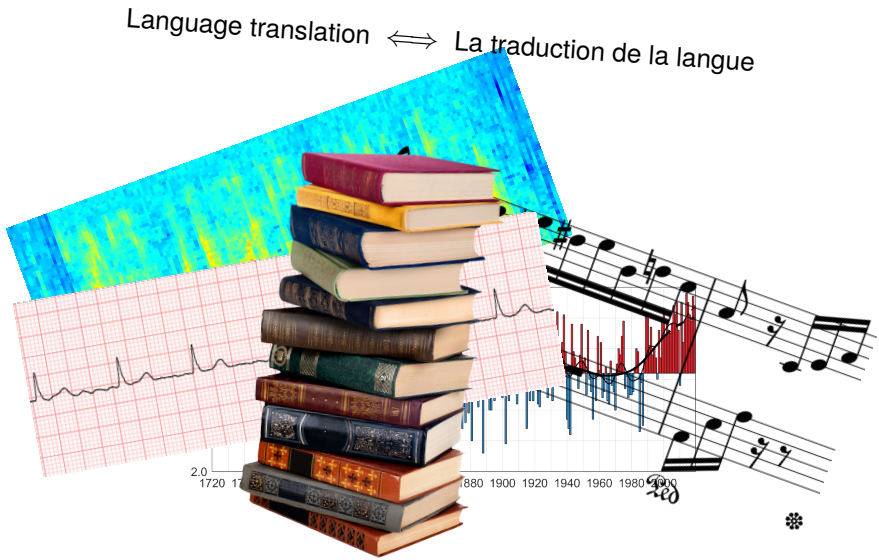
The sequential problem



The sequential problem



The sequential problem

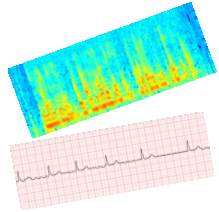


The sequential problem

The sequential problem: Inputs & Outputs

A sequential model can be used on:

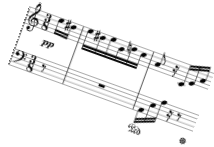
- Sequential input, $x_{1:N}$



The sequential problem: Inputs & Outputs

A sequential model can be used on:

- Sequential input, $x_{1:N}$
- Sequential output, $y_{1:T}$



The sequential problem: Inputs & Outputs

Language translation $\iff$ La traduction de la langue

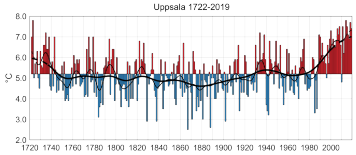
A sequential model can be used on:

- Sequential input, $x_{1:N}$
- Sequential output, $y_{1:T}$
- In any combination

Sequential classification/regression

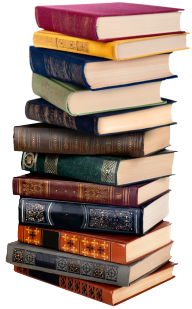
- The entire sequence is used to produce a single output
- The causality of input is not limiting the model
- $p(y|x_{1:T})$

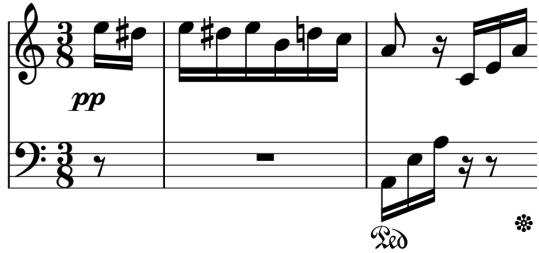
System modeling, filtering & reinforcement learning



- Predictions are made sequentially
- Future inputs are unknown, causality constrained
- Closed loop $p(y_{1:T}|x_{1:T}) = \prod_t p(y_t|y_{1:t-1}, x_{1:t})$
- or open loop $p(y_{1:T}|x_{1:T}) = \prod_t p(y_t|x_{1:t})$

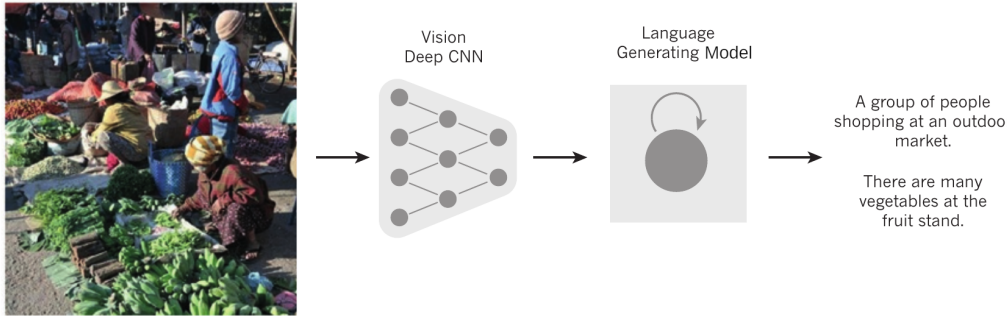
Generative models (Unsupervised)





- Predictions are made sequentially
- $p(y_{1:T}) = \prod_t p(y_t|y_{1:t-1})$

Conditional generative models



©Springer Nature: Deep learning, LeCun et al.

- Generate conditioned on some arbitrary input
- $p(y_{1:T}|x) = \prod_t p(y_t|y_{1:t-1}, x)$

Conditional generative models

Language translation $\iff$ La traduction de la langue

- The conditional data is also sequential
- $p(y_{1:T}|x_{1:N}) = \prod_t p(y_t|y_{1:t-1}, x_{1:N})$

The sequential problem: Recap

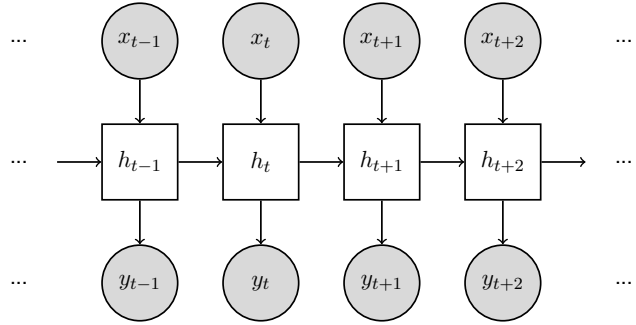
- Classification & regression
- System modeling
- Generative (which can be conditioned)

Sequential models: Overview

- Recurrent Neural Networks (RNN)
  - Long-Short Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Convolution Neural Network
  - Temporal Convolutional Network (TCN)
- Transformer

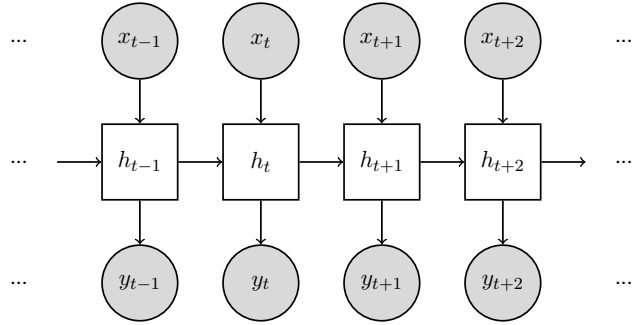
Recurrent neural network

Recurrent neural network



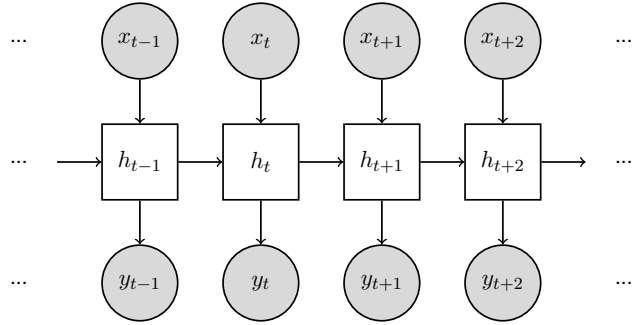
- Non-linear state space model

Recurrent neural network



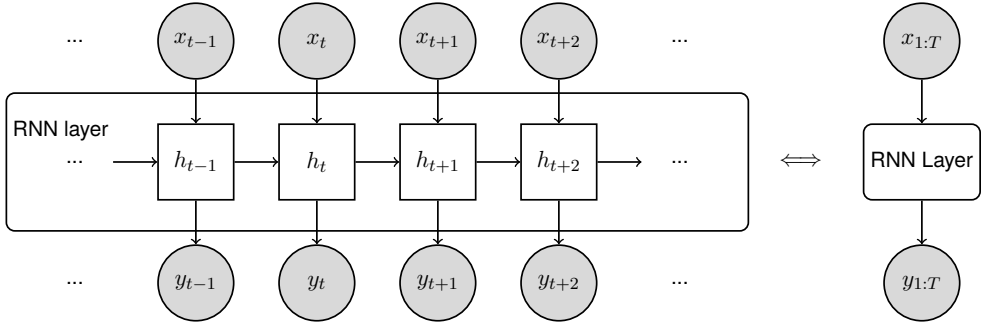
- Non-linear state space model
- State h_t, gathers information from all previous inputs, $x_{1:t}$

Recurrent neural network



- Non-linear state space model
- State h_t, gathers information from all previous inputs, $x_{1:t}$
- The incoming arrows corresponds to neural networks

Recurrent neural network



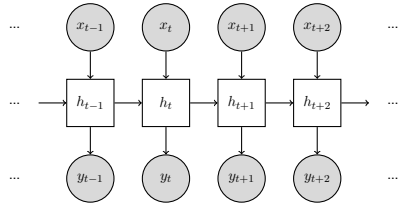
- Non-linear state space model
- State h_t, gathers information from all previous inputs, $x_{1:t}$
- The incoming arrows corresponds to neural networks

Recurrent neural network: Elman

A basic RNN is the Elman network,

$$h_t = f(h_{t-1}, x_t)$$
$$y_t = g(h_t)$$

where both f and g are neural networks.

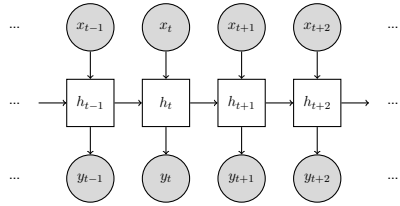


Recurrent neural network: Elman

A basic RNN is the Elman network,

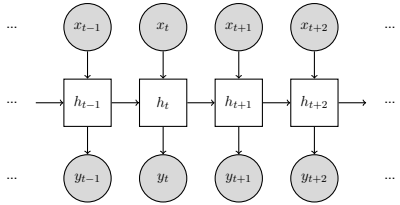
$$h_t = f(h_{t-1}, x_t)$$
$$y_t = g(h_t)$$

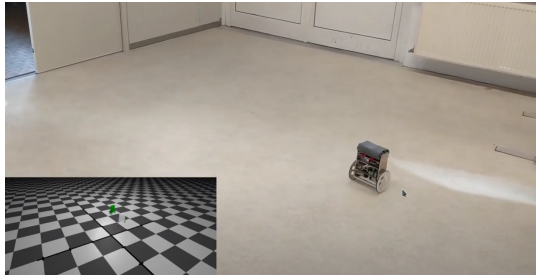
where both f and g are neural networks.



$$f(h_{t-1}, x_t) = \sigma(W_h h_{t-1} + W_x x_t + b_h)$$
$$g(h_t) = \sigma(W_y h_t + b_y)$$

Recurrent neural network: Example

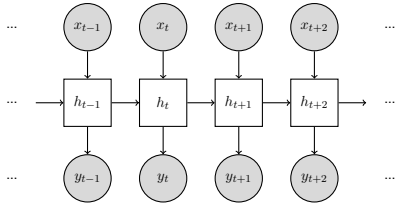


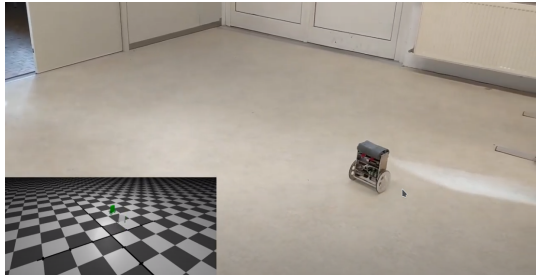


https://youtu.be/MwVZgRJSnXg

- We want to predict the torque for the motors, i.e. y_t is the torque applied this time step.

Recurrent neural network: Example

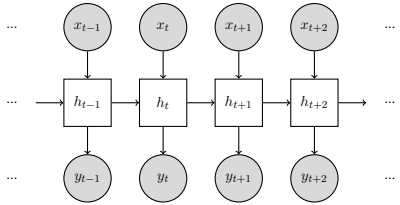


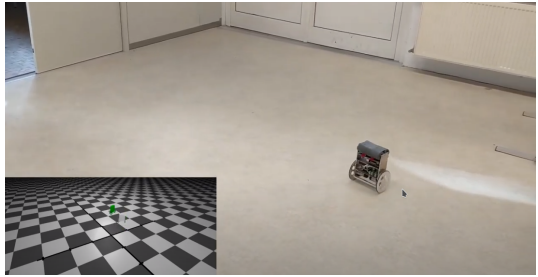


https://youtu.be/MwVZgRJSnXg

- We want to predict the torque for the motors, i.e. y_t is the torque applied this time step.
- The inputs is angular velocity, x_t, for each time step

Recurrent neural network: Example

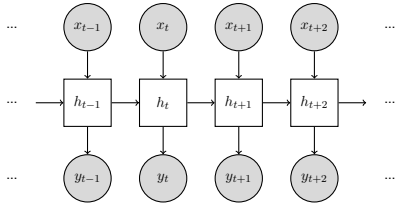


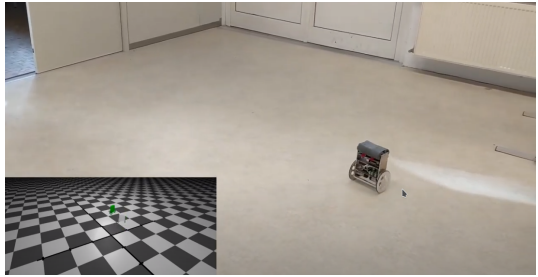


https://youtu.be/MwVZgRJSnXg

- We want to predict the torque for the motors, i.e. y_t is the torque applied this time step.
- The inputs is angular velocity, x_t, for each time step
- System modeling, $p(y_t|x_{1:t}, y_{1:t-1}) = p(y_t|h_{t-1}, x_t, y_{t-1})$

Recurrent neural network: Example

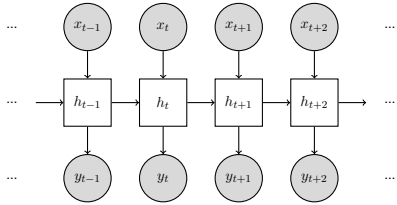




https://youtu.be/MwVZgRJSnXg

- We want to predict the torque for the motors, i.e. y_t is the torque applied this time step.
- The inputs is angular velocity, x_t, for each time step
- System modeling, $p(y_t|x_{1:t}, y_{1:t-1}) = p(y_t|h_{t-1}, x_t, y_{t-1})$
- The state contains integrated features, e.g. current vertical angle, how far the robot has traveled etc. These features are learned! (Not always interpretable)

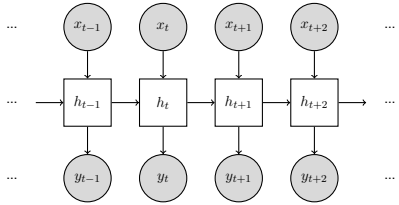
Recurrent neural network: Example 2



https://youtu.be/ynPWOMzossI

- (Uncoditioned) Generative model

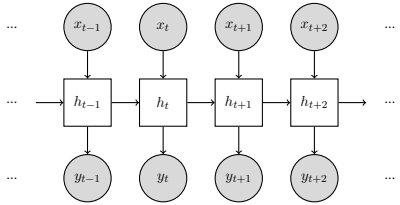
Recurrent neural network: Example 2



https://youtu.be/ynPWOMzossI

- (Uncoditioned) Generative model
- The input is the same as the output last step i.e. $x_t = y_{t-1}$

Recurrent neural network: Example 2



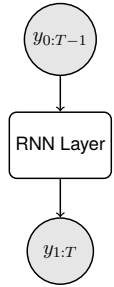
https://youtu.be/ynPWOMzossI

- (Uncoditioned) Generative model
- The input is the same as the output last step i.e. $x_t = y_{t-1}$
- The output is which key is pressed down

Generative: Training vs generating

Training

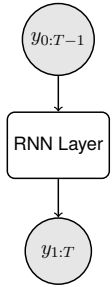
- Given a training sequence $y_{1:T}$
- **Teacher forcing** $x_{1:T} = y_{0:T-1}$
- y_0 can be chosen arbitrary, e.g. put it to zero



Generative: Training vs generating

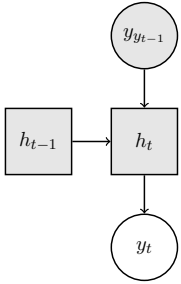
Training

- Given a training sequence $y_{1:T}$
- **Teacher forcing** $x_{1:T} = y_{0:T-1}$
- y_0 can be chosen arbitrary, e.g. put it to zero



Generating

- **Predict/sample** y_t and use it to update $h_{t+1} = f(h_t, y_t)$
- Classification — Bernoulli/Softmax distribution
- Regression — Normal (Need to fit a variance also)



Vanishing/Exploding gradient problem

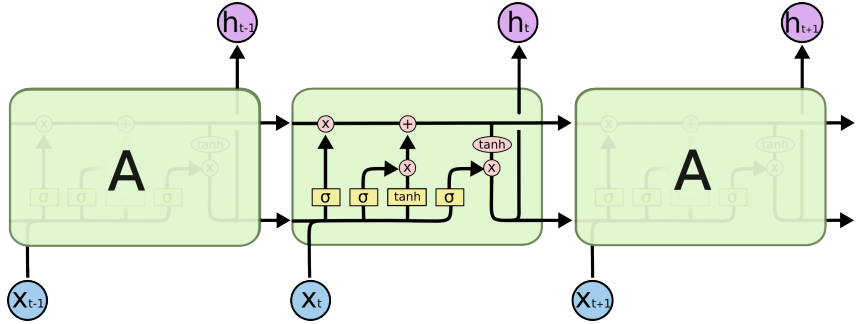
As we apply this function over and over again information gets lost,

$$h_t = f(f(\cdots f(h_{t-n}, x_{t-n+1}) \cdots, x_{t-1}), x_t)$$

R. Pascanu et Al, **On the difficulty of training Recurrent Neural Networks**

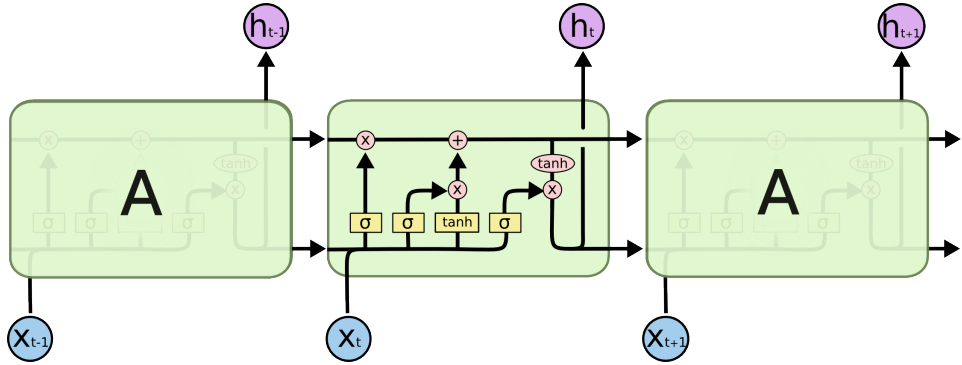
Gating & residual connections

- A residual connection: without operations
- Gates: that can turn of influence



Long-Short Term Memory

- Designed to propagate gradients better, make them more useful for tasks that require longer memory
- In essence a variant as the Elman RNN but f is chosen differently
- 2 different states c_t and h_t



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-Short Term Memory

- c_t is the long memory and h_t is the output from the previous step
- f_t is the forget gate, i_t is the incoming (new information) gate, o_t is the output gate

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c)$$
$$h_t = o_t \odot \tanh(c_t)$$

Long-Short Term Memory

- c_t is the long memory and h_t is the output from the previous step
- f_t is the forget gate, i_t is the incoming (new information) gate, o_t is the output gate

$$
\begin{aligned}
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} x_t + W_{ch} h_{t-1} + b_c) \\
h_t &= o_t \odot \tanh(c_t) \\
f_t &= \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f) \\
i_t &= \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i) \\
o_t &= \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o)
\end{aligned}
$$

Long-Short Term Memory

- c_t is the long memory and h_t is the output from the previous step
- f_t is the forget gate, i_t is the incoming (new information) gate, o_t is the output gate
- y_t can be model with another Neural Network, $y_t = g(h_t)$

$$
\begin{aligned}
c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_{cx} x_t + W_{ch} h_{t-1} + b_c) \\
h_t &= o_t \odot \tanh(c_t) \\
f_t &= \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f) \\
i_t &= \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i) \\
o_t &= \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o)
\end{aligned}
$$

UPPSALA UNIVERSITET

Example: Generating text

- Goal: A model that can generate text similar to a given dataset
- Data: A large corpus of text, i.e. a collection of books, linux source code, etc.
- Model assumption: Each unique character is represented with one hot encoding

Example: Generating text

- Goal: A model that can generate text similar to a given dataset
- Data: A large corpus of text, i.e. a collection of books, linux source code, etc.
- Model assumption: Each unique character is represented with one hot encoding

Generative model (Unsupervised)

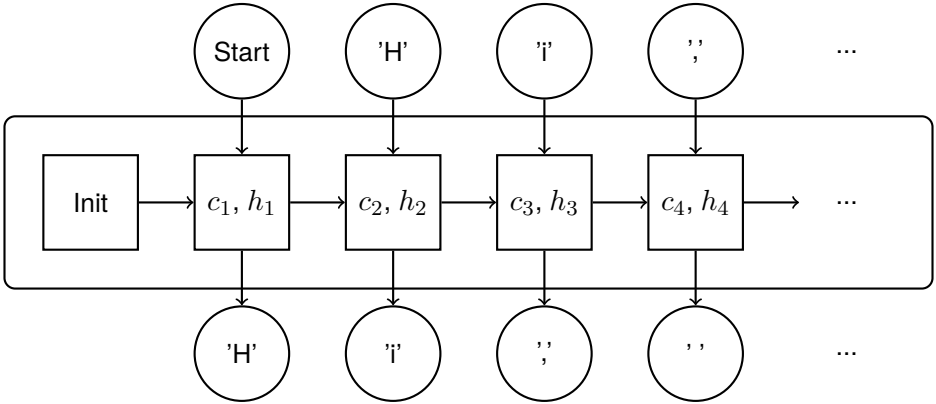
Example: Generating text

- Goal: A model that can generate text similar to a given dataset
- Data: A large corpus of text, i.e. a collection of books, linux source code, etc.
- Model assumption: Each unique character is represented with one hot encoding

Generative model (Unsupervised)

The last step output is used as input, $x_t = y_{t-1}$

Example: Generating text



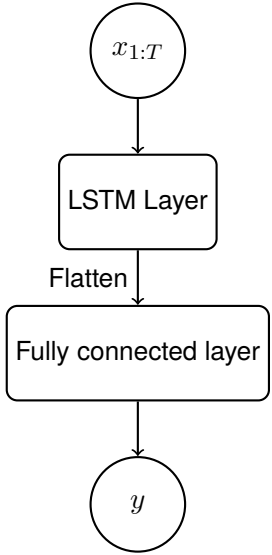
Example: Generating text

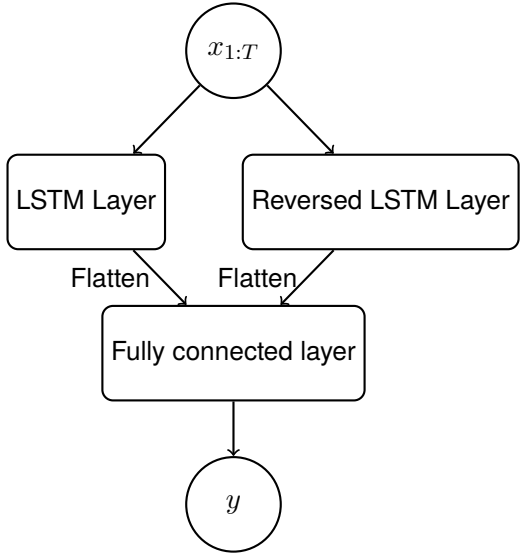
Example from a model trained on works by Shakespeare

> *PANDARUS: Alas, I think he shall be come approached and the day When little srain would be attain'd into being never fed, And who is but a chain and subjects of his death, I should not sleep.*
> *Second Senator: They are away this miseries, produced upon my soul, Breaking and strongly should be buried, when I perish The earth and thoughts of many states.*
> *DUKE VINCENTIO: Well, your wit is in the care of side and that.*
> *Second Lord: They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the wars.*
> *Clown: Come, sir, I will make did behold your worship.*
> *VIOLA: I'll drink it.*

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

Recurrent neural network: Classification





Temporal convolutional network

Convolution

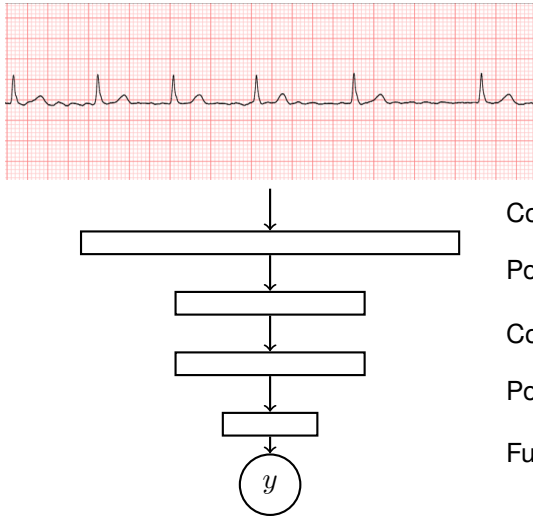
Many sequences problems share properties with images

- Scale and translation invariance
- Features from locally correlated inputs

Example: Classification

- ECG measurement 10 seconds long 400 Hz
- Single lead, i.e. one probe
- $x \in \mathbb{R}^{4000 \times 1}$
- Predict if there is an abnormality

Example: Classification



A. Riberio et al. **Automatic diagnosis of the 12-lead ECG using a deep neural network** *Nature Communications*

Temporal Convolutional Network

Can we use convolutions to generate sequences or system modeling?

Temporal Convolutional Network

Can we use convolutions to generate sequences or system modeling?

- We need to offset the convolutions
- We need to incorporate pooling

Temporal Convolutional Network

Can we use convolutions to generate sequences or system modeling?

- We need to offset the convolutions
- We need to incorporate pooling

Temporal convolutional networks (+ diluted convolutions)

Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Can be implemented as a convolution (if we are careful with our padding) and be stacked!

UPPSALA UNIVERSITET

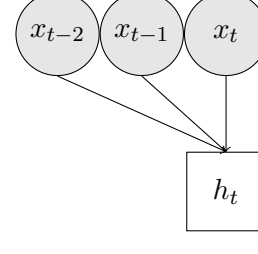
Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Can be implemented as a convolution (if we are careful with our padding) and be stacked!



UPPSALA UNIVERSITET

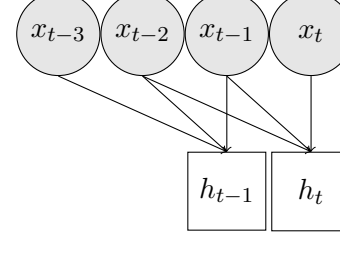
Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Can be implemented as a convolution (if we are careful with our padding) and be stacked!



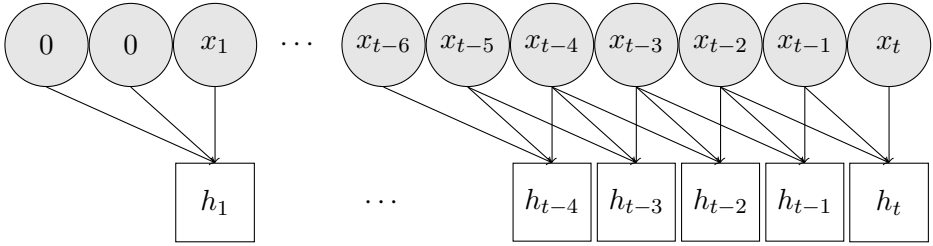
Temporal Convolutional Network

Autoregressive model instead of hidden units

Base building block

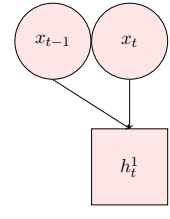
$$h_t = f(x_{t-k:t}) = \sigma(W x_{t-k:t} + b)$$

Can be implemented as a convolution (if we are careful with our padding) and be stacked!



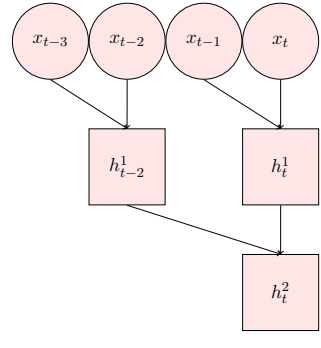
UPPSALA UNIVERSITET

Temporal convolutional network: Dilution

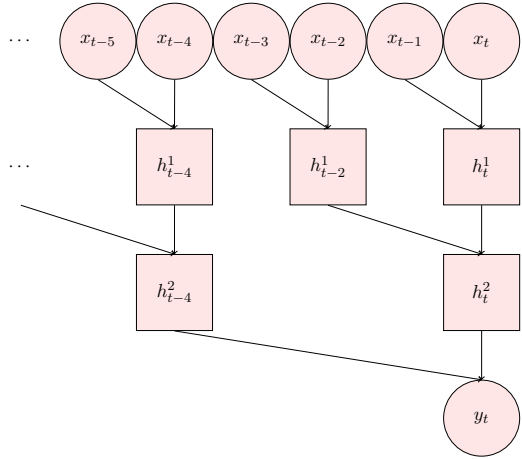


UPPSALA UNIVERSITET

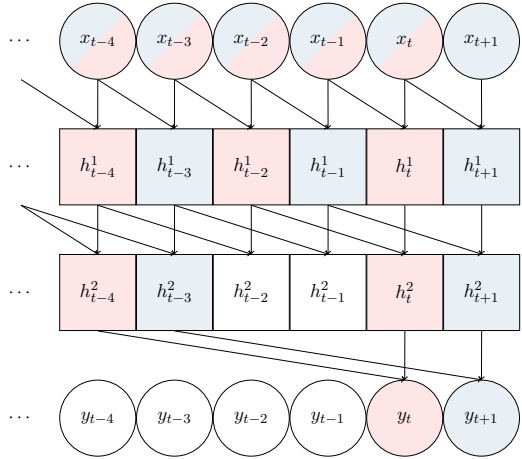
Temporal convolutional network: Dilution



Temporal convolutional network: Dilution



Temporal convolutional network: Dilution



TCN vs RNN

TCN

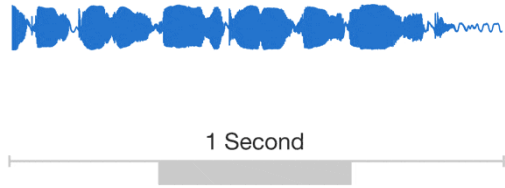
- Easy to train
- Can use design ideas from convolutional neural network

RNN

- Harder to train
- Fast at generating sequence

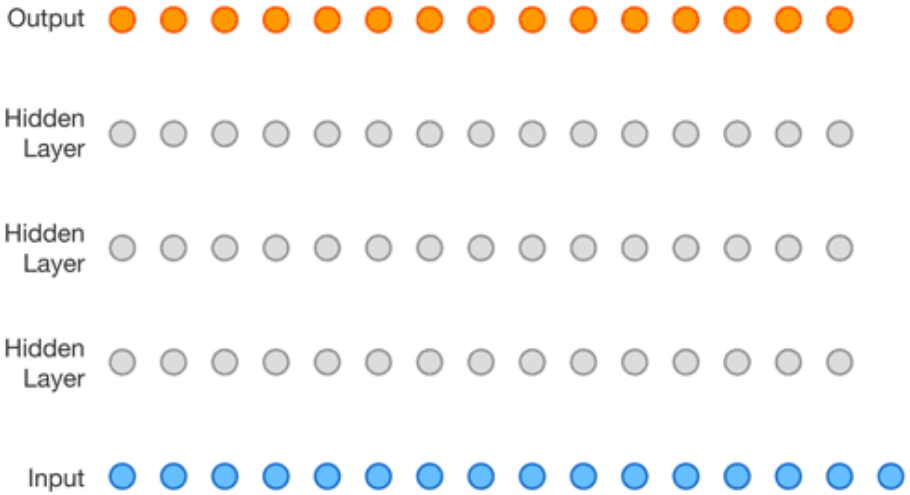
Example: Generating speech

- Generative, $\prod_t p(y_t|y_{1:t-1})$
- 5 seconds long sequences 16 000 Hz
- $y \in \mathbb{R}^{80000\times 1}$



A. van der Oord et al, **Wavnet: A Generative model for audio**, *arxiv: 1609.03499*

Example: Generating speech



A. van der Oord et al, **Wavnet: A Generative model for audio**, *arxiv: 1609.03499*

UPPSALA UNIVERSITET

Key Concepts from lecture 8

Identify our problem: Generative, classification, modeling

Causality: Naturally oredered data

Sequential models: RNN, LSTM, TCN

Reuse ideas: Residual connections, stacking layers, gating

Further reads/links

- **Generating muisc/speech:**
  https://medium.com/artists-and-machine-intelligence/neural-nets-for-generating-music-f46dffac21c0
- **The unreasonable effectiveness of RNN :**
  http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- **Understanding LSTMs:**
  https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- **TCN with more details**
  https://medium.com/unit8-machine-learning-publication/temporal-convolutional-networks-and-forecasting-5ce1b6e97ce4