Lecture

Speech and Audio Signal Processing

Lecture 12: Hidden Markov Models (HMM) and Speech Recognition, Part II

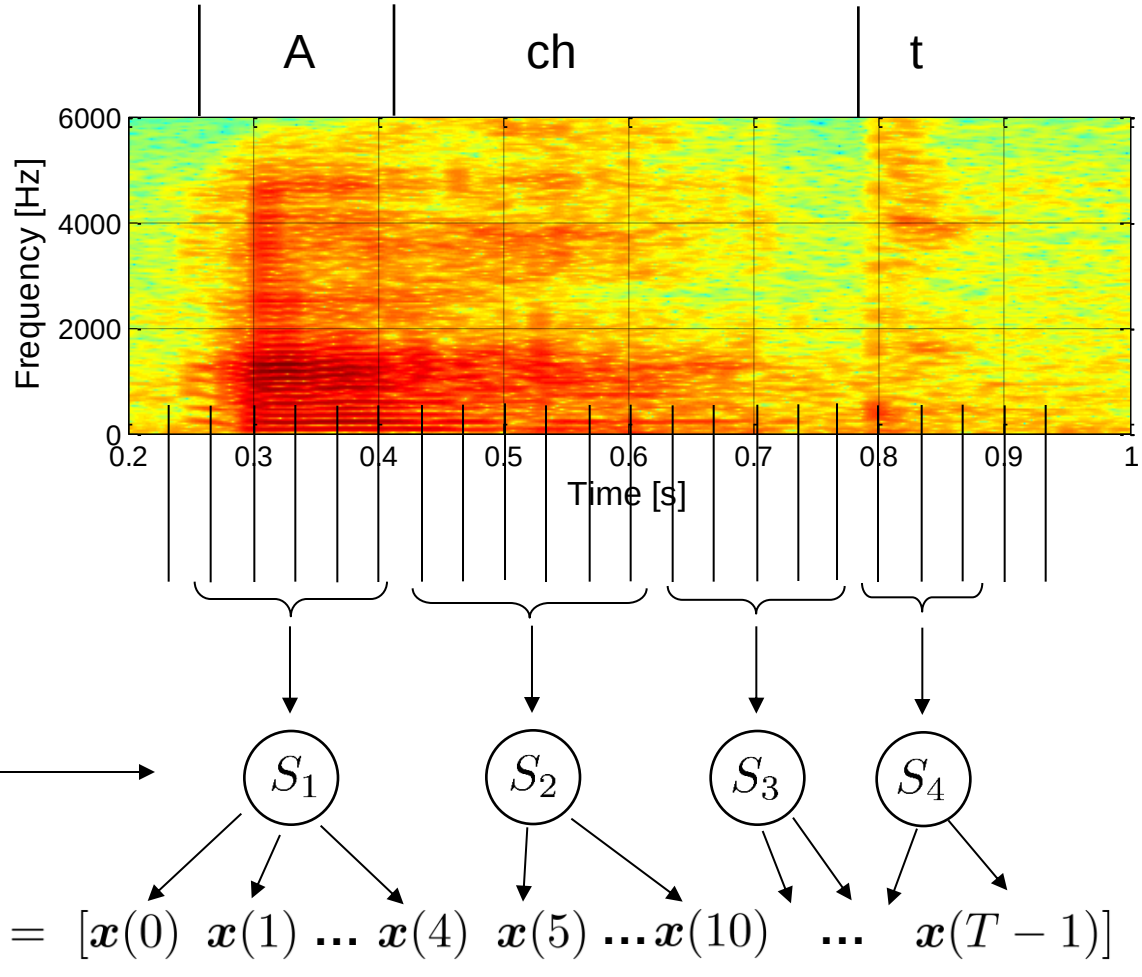
TECHNISCHE UNIVERSITÄT DARMSTADT

Content

- Repetition
- The three basic problems of HMMs
  - Evaluation problem
  - Decoding problem
  - Model parameter estimation problem
- Model parameter estimation problem:
  - Introduction of backward probability
  - Estimation of the transition probabilities
  - Estimation of the observation probabilities.
- Application of speech recognition systems
- DNN based speech recognition

Principle of speech recognition

- **Procedure:**

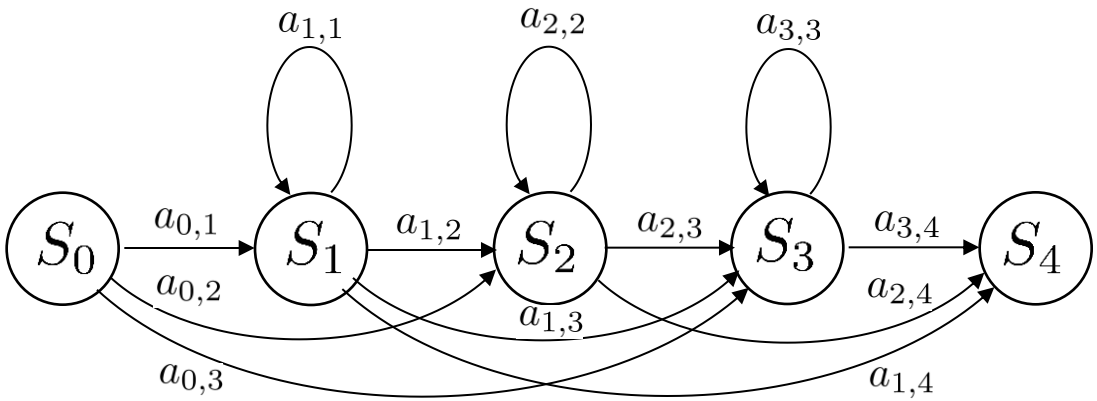


HMM: The general definition

- **The transition probabilities:**

  Modeling the statistical dependencies of consecutive feature vectors.

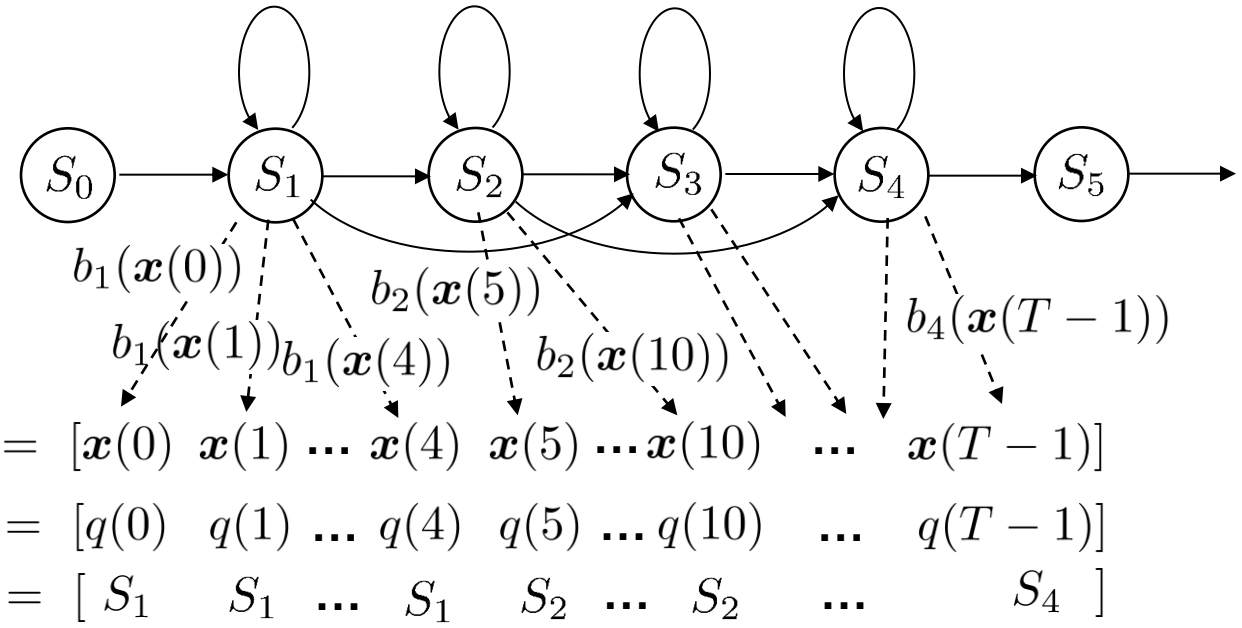
$$a_{i,j} = p\left(q(n) = S_j \middle| q(n-1) = S_i\right)$$



HMM: The general definition

- **The observation probabilities:**

$$b_j(\boldsymbol{x}(n)) = p\left(\boldsymbol{x}(n)\middle|q(n) = S_j\right)$$



The three basic problems of HMMs

TECHNISCHE UNIVERSITÄT DARMSTADT

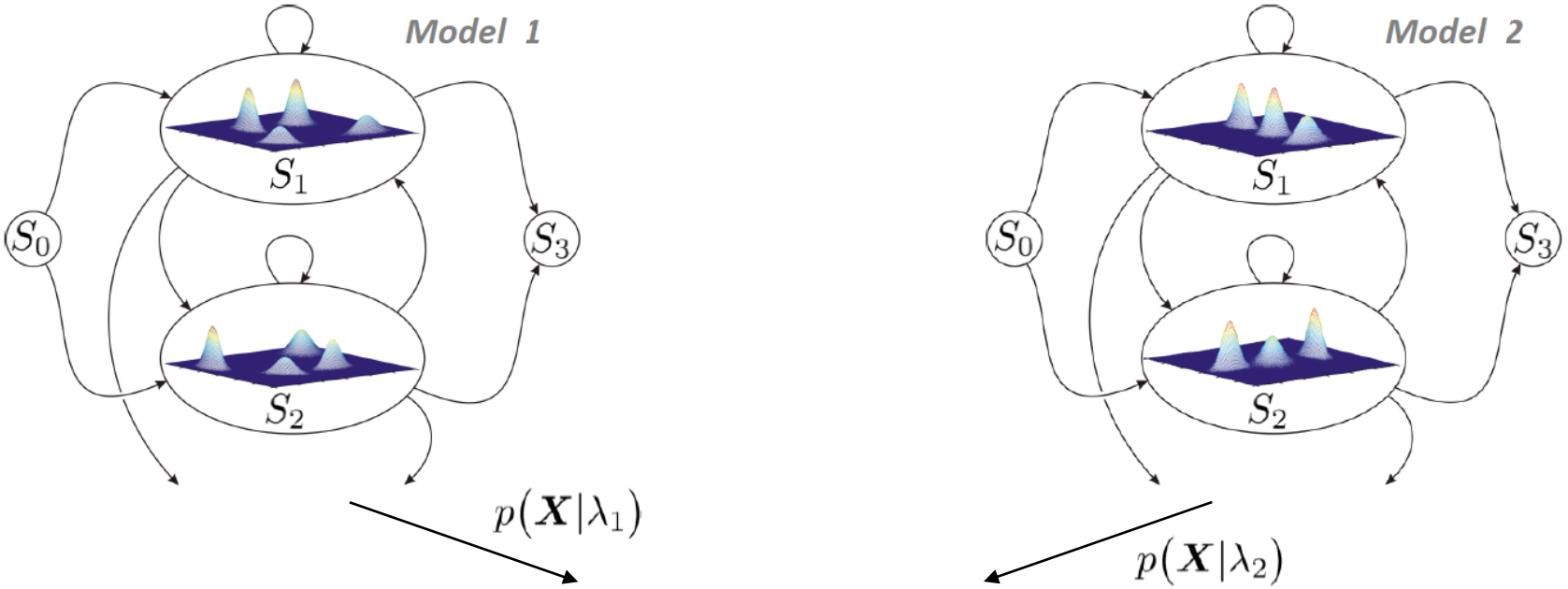
- **Evaluation problem:**
  - Estimate the probability $p(\boldsymbol{X}|\lambda)$ that a hidden Markov model has generated an observed sequence $\boldsymbol{X}$.
  - The Markov model parameters $a_{i,j}$ and $b_j(\boldsymbol{x}(n))$ are combined by λ.
- **Decoding problem:**
  - Estimate the „correct“, i.e. most probable, hidden state sequence:

$$\hat{\boldsymbol{q}} = [S_0, \hat{q}(1), \hat{q}(2), \ldots, \hat{q}(T-2), S_{N-1}]^{\mathrm{T}}$$

    given the observed sequence $\boldsymbol{X}$.
- **Model parameter estimation:**
  - Adjustment or training of the hidden Markov models based on training data.

Evaluation problem

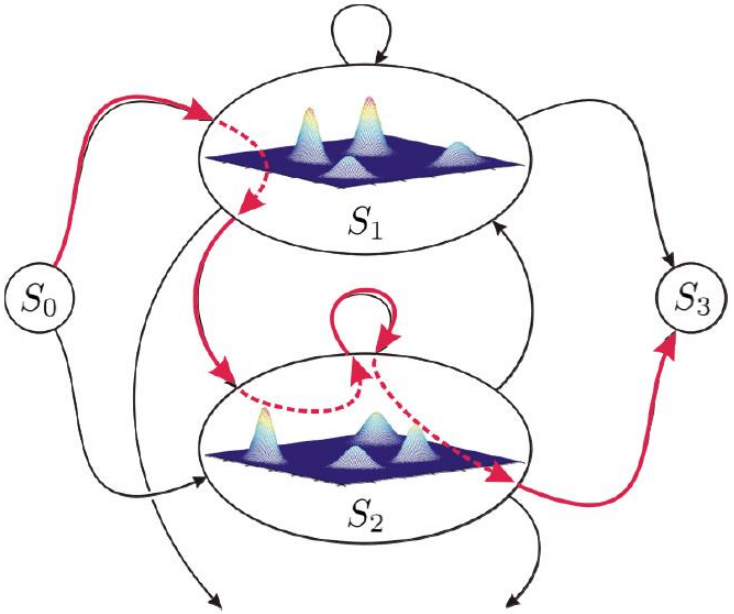
- **Evaluation problem:**
  - Determine the probability that a Markov model has generated an observed data sequence. => Determine the most probable model.



Observed sequence: $\boldsymbol{X} = [\boldsymbol{x}(0), \boldsymbol{x}(1), \ldots, \boldsymbol{x}(N-1)]$

Decoding problem

- **Decoding problem:**
  - Determine the state sequence which has generated the observed sequence with the highest probability



$$\hat{\boldsymbol{q}} = \underset{\boldsymbol{q}_j}{\operatorname{argmax}} \left\{ p(\boldsymbol{q}_j, \boldsymbol{X}|\lambda) \right\}$$

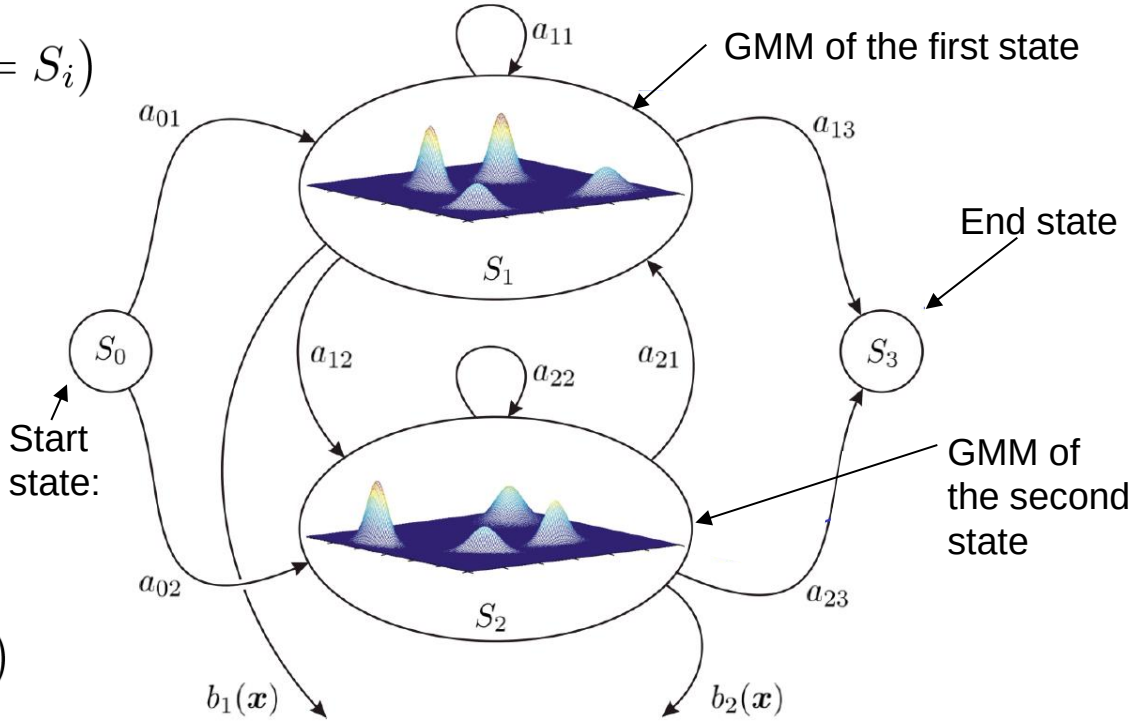
Observed sequence: $\boldsymbol{X} = \left[\boldsymbol{x}(0), \boldsymbol{x}(1), \boldsymbol{x}(2)\right]$

Model parameter estimation

- The transition probabilities:

$$a_{i,j} = p\left(q(n) = S_j \middle| q(n-1) = S_i\right)$$

$$\boldsymbol{A} = \begin{Bmatrix} 0 & a_{0,1} & a_{0,2} & a_{0,3} & 0 \\ 0 & a_{1,1} & a_{1,2} & a_{1,3} & a_{1,4} \\ 0 & 0 & a_{2,2} & a_{2,3} & a_{2,4} \\ 0 & 0 & 0 & a_{3,3} & a_{3,4} \\ 0 & 0 & 0 & 0 & 0 \end{Bmatrix}$$



- The observation probabilities:

$$b_j(\boldsymbol{x}(n)) = p\left(\boldsymbol{x}(n) \middle| q(n) = S_j\right)$$

modeled by Gaussian mixture models:

$$b_j(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_{j,k}\, \mathcal{N}(\boldsymbol{x} | \boldsymbol{\mu}_{j,k}, \boldsymbol{\Sigma}_{j,k})$$

Model parameter estimation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Target of the model parameter estimation:
  - Maximize the probability for the **training** sequence $\boldsymbol{X}$ given the model parameter

    $$p(\boldsymbol{X}|\lambda) \longrightarrow \max$$

  - where λ describes the HMM parameters.
  - Comparable to GMM, also for the HMM parameter estimation, an iteration is developed based on an initial model. The iteration is – as typically – terminated when a given optimization criterion has been achieved or when a maximum number of iterations has been reached.
  - Typically, only achieving local maxima cannot be avoided.
  - The iteration described in the following is known as **“Baum-Welch” algorithm** or **“forward-backward” algorithm**.

Model parameter estimation

- Backward probability
  - Comparable to the iterative calculation of the forward probability:

$$f_i(n) = p(\boldsymbol{X}^{(n)},\, q(n) = S_i|\lambda)$$

$$f_i(n) = \left[\sum_{j=1}^{N-2} f_j(n-1)\, a_{j,i}\right] b_i(\boldsymbol{x}(n))$$

$$\boldsymbol{f}(n) = A^{\mathrm{T}}\, \boldsymbol{f}(n-1) * \boldsymbol{b}(\boldsymbol{x}(n))$$

element-wise multiplication

  - the backward probability is defined:

$$r_i(n) = p(\boldsymbol{X}_{(n+1)},\, q(n) = S_i|\lambda)$$

  - where the observed sequence $\boldsymbol{X}_{(n)}$ describes all observations starting at time index *n* until the end of the sequence:

$$\boldsymbol{X}_{(n)} = [\boldsymbol{x}(n),\, \boldsymbol{x}(n+1),\, \ldots,\, \boldsymbol{x}(T-1)]$$

  - An iterative calculation is possible:

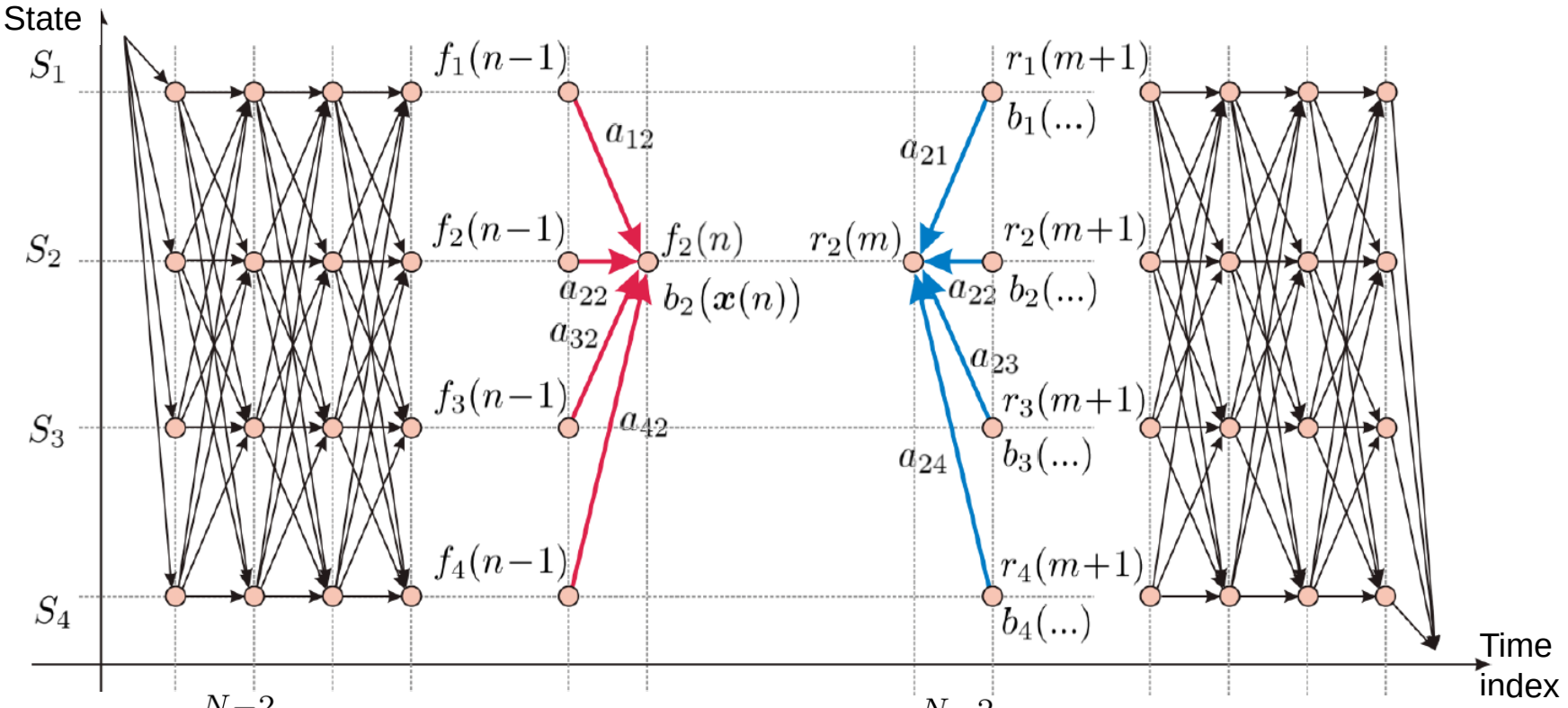
$$r_i(n) = \sum_{j=1}^{N-2} r_j(n+1)\, b_j(\boldsymbol{x}(n+1))\, a_{i,j}$$

Initialization:

$$r_i(T) = a_{i,N}$$

Model parameter estimation

- Forward and Backward probability:



$$f_i(n) = \sum_{j=1}^{N-2} f_j(n-1)\, a_{j,i}\, b_i(\boldsymbol{x}(n)) \qquad r_i(m) = \sum_{j=1}^{N-2} r_j(m+1)\, b_j(\boldsymbol{x}(m+1))\, a_{i,j}$$

Model parameter estimation

- State probability:
  - Based on the **forward and backward probability** the probability that the model is in state *i* at the time index *n* having observed the **complete** observation sequence:

$$\gamma_i(n) = p(q(n) = S_i|\boldsymbol{X}, \lambda) = \frac{p(q(n) = S_i,\, \boldsymbol{X}|\lambda)}{p(\boldsymbol{X}|\lambda)}$$

$$= \frac{p(q(n) = S_i,\, \boldsymbol{X}^{(n)}|\lambda)\, p(q(n) = S_i,\, \boldsymbol{X}_{(n+1)}|\lambda)}{p(\boldsymbol{X}|\lambda)}$$

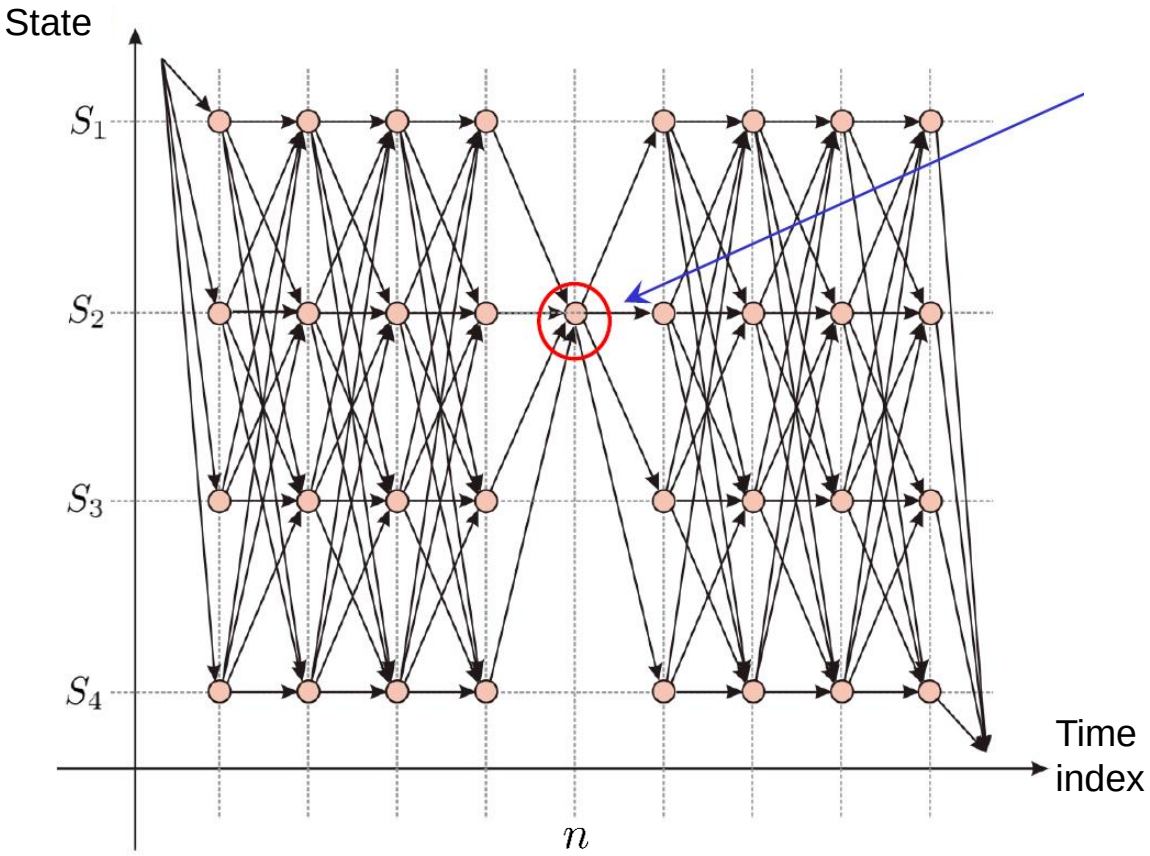
$$= \frac{f_i(n)\, r_i(n)}{p(\boldsymbol{X}|\lambda)} \qquad \text{with:} \quad \sum_{i=1}^{N-2} \gamma_i(n) = 1$$

  - The normalization can be determined based on the forward or backward probability:

$$p(\boldsymbol{X}|\lambda) = \sum_{j=1}^{N-2} f_j(T-1)\, a_{j,N-1} = \sum_{j=1}^{N-2} r_j(0)\, b_j(\boldsymbol{x}(0))\, a_{0,j}$$

Model parameter estimation

- State probability:



The Markov model is in state S_i at the time index *n*

$$p(q(n) = S_i,\, \boldsymbol{X}|\lambda) = f_i(n)\, r_i(n)$$

Model parameter estimation

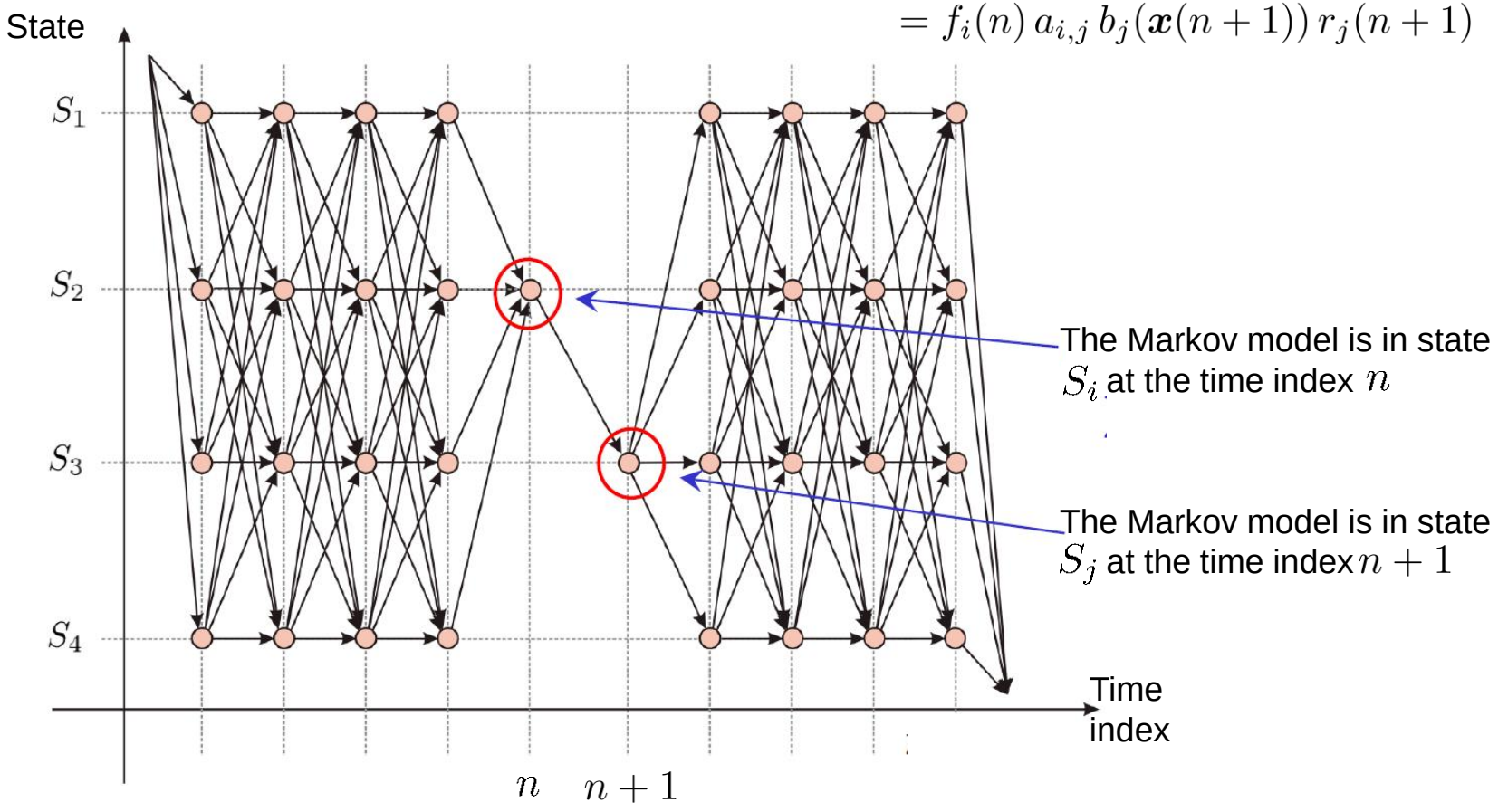
- Transition probability:
  - Based on the forward and backward probability one can also calculate the probability that the Markov model changes from state i to state j at the time index n:

$$\begin{aligned}
\xi_{i,j}(n) &= p(q(n) = S_i, q(n+1) = S_j | \boldsymbol{X}, \lambda) \\
&= \frac{p(q(n) = S_i, q(n+1) = S_j, \boldsymbol{X} | \lambda)}{p(\boldsymbol{X} | \lambda)} \\
&= \frac{f_i(n)\, a_{i,j}\, b_j(\boldsymbol{x}(n+1))\, r_j(n+1)}{p(\boldsymbol{X} | \lambda)}
\end{aligned}$$

Model parameter estimation

- Transition probability:

$$p(q(n) = S_i, q(n+1) = S_j, \boldsymbol{X}|\lambda)$$
$$= f_i(n)\, a_{i,j}\, b_j(\boldsymbol{x}(n+1))\, r_j(n+1)$$



Model parameter estimation

- **Estimation of the transition probabilities:**
  - The transition probabilities are updated (iteration step) as follows:

aij = p(q(n) = Sj | q(n-1) = Si

P (q(N) = Sj, q(n-1) = Si)

$$a_{i,j} = \frac{\sum_{n=0}^{T-1} \xi_{i,j}(n)}{\sum_{n=0}^{T-1} \gamma_i(n)} \quad i,j \in \{1, N-2\}$$

$$a_{0,j} = \gamma_j(0)$$

Expected mean number of transitions between the state i and j.

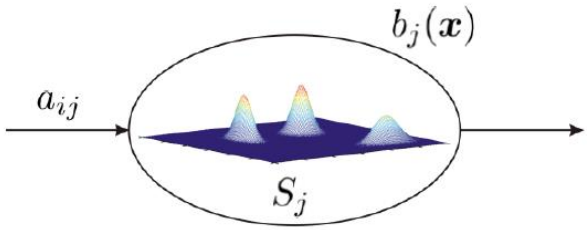
Expected mean number of transitions starting at state i.

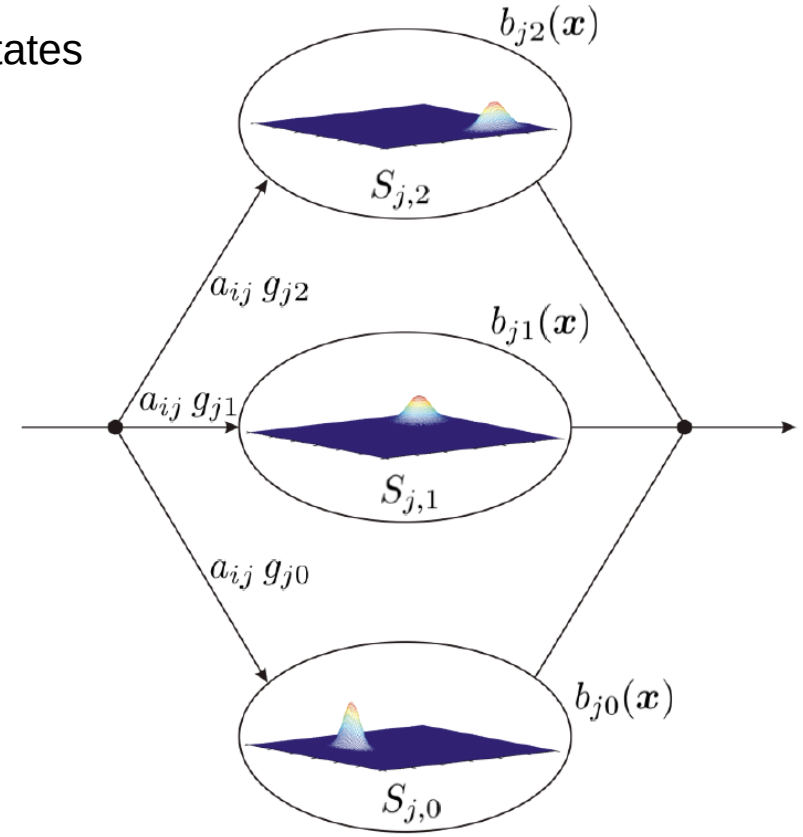
P(q(n-1) = Si

  - The parameter updates should be calculated for several observations $\boldsymbol{X}$ and mean values (average over the observations).

Model parameter estimation

- Observation probability (GMM models):
  - Decomposition of one state with several Gaussians into several states with one Gaussian each:





$$b_j(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_{jk} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$$

$$= \sum_{k=0}^{K-1} g_{jk}\, b_{jk}(\boldsymbol{x})$$

Model parameter estimation

TECHNISCHE UNIVERSITÄT DARMSTADT

- Observation probability (GMM models):
  - Comparable to the update procedure of the transition probabilities, the individual observation probabilities can be determined:

$$\zeta_{i,j,k}(n) \;=\; p(q(n)=S_i, q(n+1)=S_j, \boldsymbol{x}(n+1) \mapsto \mathcal{N}_{jk} | \boldsymbol{X}, \lambda)$$

Probability that a transition between the states i and j occurred and that the k-th Gaussian was mainly involved ("allocation" value)

  - An expression with the forward and backward probabilities is possible:

$$\zeta_{i,j,k}(n) = \frac{f_i(n)\, a_{i,j}\, g_{jk}\, b_{jk}(\boldsymbol{x}(n+1))\, r_j(n+1)}{p(\boldsymbol{X}|\lambda)}$$

  - Relation to the transition probabilities:

$$\xi_{i,j}(n) = \sum_{k=0}^{K-1} \zeta_{i,j,k}(n)$$

with: $\xi_{i,j}(n) = \dfrac{f_i(n)\, a_{i,j}\, b_j(\boldsymbol{x}(n+1))\, r_j(n+1)}{p(\boldsymbol{X}|\lambda)}$

$$b_j(\boldsymbol{x}) = \sum_{k=0}^{K-1} g_{jk}\, b_{jk}(\boldsymbol{x})$$

Model parameter estimation

- Observation probability (GMM models):
  - Summing over all (start) states i, one obtains the probability that the model is in state j at the time index n AND the k-th Gaussian generated the observation value.

$$\zeta_{j,k}(n) = \sum_{i=1}^{N-1} \zeta_{i,j,k}(n)$$

„affiliation value for state *j*“

$$= \frac{\sum_{i=1}^{N-1} f_i(n)\, a_{i,j}\, g_{jk}\, b_{jk}(\boldsymbol{x}(n+1))\, r_j(n+1)}{p(\boldsymbol{X}|\lambda)}$$

  - Relation to the state probabilities:

$$\gamma_j(n) = \sum_{k=0}^{K-1} \zeta_{j,k}(n)$$

After the sum: Probability that the model is in state j at the time index n

  - Having the “affiliation” values, comparable to the GMM iteration procedure, the Gaussian parameters can be updated.

Model parameter estimation

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Estimation of the observation probabilities** (GMM models), Final iteration:
  - Gaussian probability model:

$$b_j(\boldsymbol{x}(n)) = \sum_{k=0}^{K-1} g_{jk}\, \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk})$$

  - Adaptation of the weights:

$$g_{jk} = \frac{\sum_{n=0}^{T-1} \zeta_{jk}(n)}{\sum_{n=0}^{T-1} \gamma_j(n)}$$

Mean frequency that the *k*-th Gaussian generated the observation value at the state *j*

Mean frequency that the model is in state *j*

  - Adaptation of the covariance matrices:

$$\boldsymbol{\Sigma}_{jk} = \frac{\sum_{n=0}^{T-1} \zeta_{jk}(n)\, \left[\boldsymbol{x}(n) - \boldsymbol{\mu}_{jk}\right] \left[\boldsymbol{x}(n) - \boldsymbol{\mu}_{jk}\right]^{\mathrm{T}}}{\sum_{n=0}^{T-1} \zeta_{jk}(n)}$$

  - Adaptation of the means:

$$\boldsymbol{\mu}_{jk} = \frac{\sum_{n=0}^{T-1} \zeta_{jk}(n)\boldsymbol{x}(n)}{\sum_{n=0}^{T-1} \zeta_{jk}(n)}$$

Model parameter estimation

TECHNISCHE UNIVERSITÄT DARMSTADT

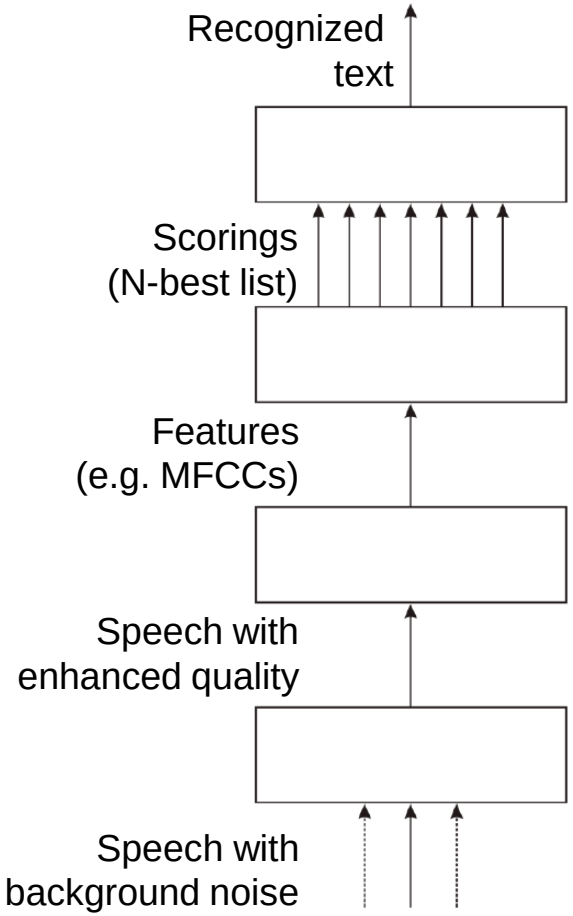
- Initialization of HMMs:
  - First, one fixes the number of states and the topology (i.e. allowed and forbidden transitions).
  - For each state, first, only one Gaussian models the probability distribution.
  - During the training procedure, continuously the number of Gaussians is increased, e.g. by splitting:

$$
\begin{aligned}
g &\rightarrow g_0 = \frac{g}{2},\ g_1 = \frac{g}{2} \\
\boldsymbol{\mu} &\rightarrow \boldsymbol{\mu}_0 = \boldsymbol{\mu} + 0.2\sqrt{\mathrm{diag}\{\boldsymbol{\Sigma}\}},\ \boldsymbol{\mu}_1 = \boldsymbol{\mu} - 0.2\sqrt{\mathrm{diag}\{\boldsymbol{\Sigma}\}} \\
\boldsymbol{\Sigma} &\rightarrow \boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma},\ \boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}
\end{aligned}
$$

  - This procedure is repeated as long as the probability which generates the training observation sequences is no more significantly increased or the maximum number of parameters has been reached.

Speech recognition: Applying HMMs

Basis principle of speech recognition



- **Decision:**
  - Determines the best entry based on additional a-priori knowledge (word probabilities).
- **Classification:**
  - For each activated word class an evaluation is performed.
  - As a result, first the N best scores are given to the output.
- **Feature extraction:**
  - Data compression.
  - Extraction of relevant parameters.
- **Pre-processing:**
  - Beamforming
  - Noise reduction and echo cancellation

Types of speech recognition systems

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Single word recognition:**
  - Single words or commands.
  - The command words are spoken separately (with pauses).
- **Key word recognition:**
  - A single word or word sequence in a arbitrary utterance.
  - When the key word has been recognized, a specific speech recognition system is started.
  - The key word recognizer – as activation tool – is an alternative to the push-to-talk button one often finds in cars.
- **Combined word recognizer:**
  - Sequence of continuously spoken words in a small vocabulary.
- **Continuous speech recognizer:**
  - Full, fluently spoken sentences.

Types of speech recognition systems

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Speaker dependent systems:**
  - Such systems are trained for speakers in a dedicated training phase.
  - The training phase to this speaker may be rather long, dependent on the desired vocabulary of the recognizer and the desired quality.
- **Speaker independent systems:**
  - No (speaker specific) training phase is necessary.
  - In order to obtain a reasonable detection quality, a large training basis has to be provided.
- **Speaker adaptive systems:**
  - One starts the application with a universal model, which then continuously adapts to a specific speaker.

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Recognition criterion:**
  - Given an observation sequence $\boldsymbol{X}$, the word (single word recognizer) or the word sequence $\widehat{W}$ (continuous word recognizer) should be selected with the highest a-posteriori probability within the possible (allowed) words or word sequences V:

$$\widehat{W} = \arg\max_{W \in V} \{p(W|\boldsymbol{X})\}$$

  - Using the Bayes rule, one can note this expression as follows:

$$\widehat{W} = \arg\max_{W \in V} \left\{ \frac{p(\boldsymbol{X}|W)\, p(W)}{p(\boldsymbol{X})} \right\}$$

  - As the probability for the feature sequence does not influence the maximization, the previous expression is similar to the following:

$$\widehat{W} = \arg\max_{W \in V} \{p(\boldsymbol{X}|W)\, p(W)\}$$

- **Recognition criterion:**
  - Function to optimize:

$$\widehat{W} = \arg\max_{W \in V} \{p(\boldsymbol{X}|W)\, p(W)\}$$

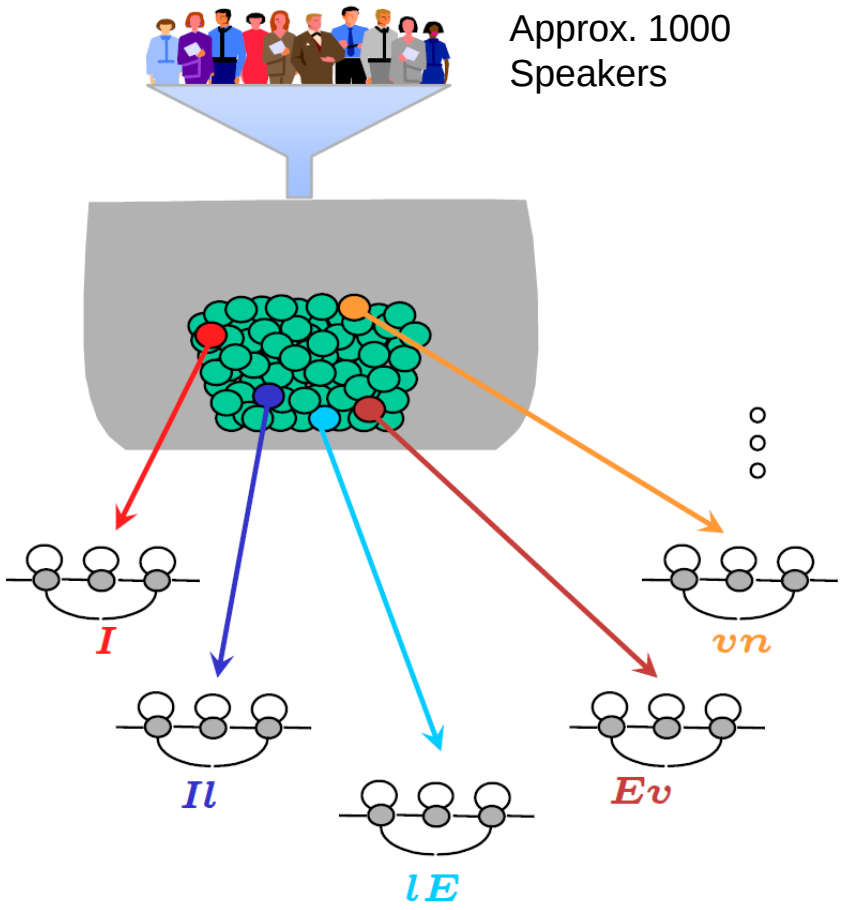
  - The probability $p(\boldsymbol{X}|W)$ gives the probability for the observed feature sequence $\boldsymbol{X}$ under the condition that the word / word sequence W has been uttered.
  - For the model of these conditional probabilities, HMMs have proven to be appropriate models. The HMM models an **acoustic model.**

  - The probability $p(W)$ is the a-priori probability of the word / word sequence. This probability is independent of the observed feature sequence and contains prior knowledge about the words / word sequences W. E.g., some are more probable than others. This part of the optimization criterion is known as **speech model.**

Basic units of Hidden Markov models

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Acoustic modeling:**
  - One extracts – based on a large speech data basis (e.g., 1000 speakers with approx. 1h speech material for each speaker) acoustic **basic units** and trains HMMs for each.
  - The speech material has to be annotated in order to find the appropriate speech samples.
  - Such basic units can be phonemes, phoneme pairs or triples of phonemes.
  - Additionally, often some HMMs are trained for important (entire) words (i.e. digits).
  - Per language, there are approximately 50 phonemes.
  - When using phoneme pairs or triples the number of phoneme groups increases significantly to 50^2 or 50^3.

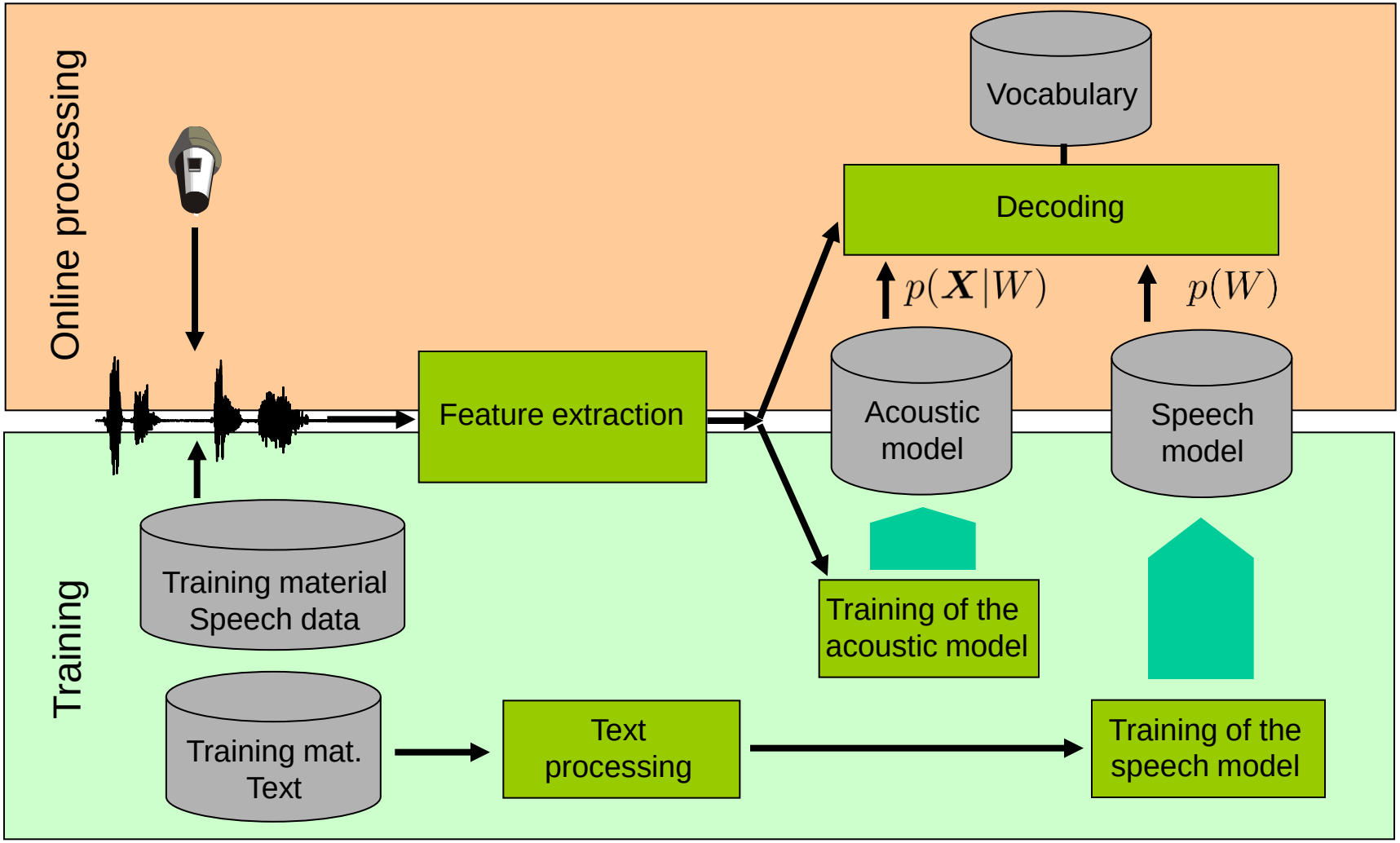


Basic units of Hidden Markov models

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Acoustic modeling:**
  - The concatenation of the phoneme models to word models or word sequence models based on the basic unit models (s. prov. slide) can be done rather easily and can be done also online during the operation of the speech recognizer.
  - Example: Using a speech dialog system, based on the generated question by the dialog machine, the system can adjust / adapt the allowed words / word sequences to the possible cluster of answers V .
  - This allows to keep the vocabulary for each recognition step rather small, which leads to a reduced number of errors, but nevertheless generates a flexible recognizer.
  - This is especially important for request to data bases.

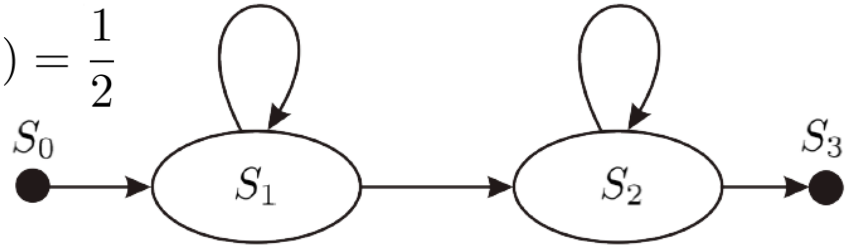
Training vs. online processing of a speech recognizer



Connecting HMMs

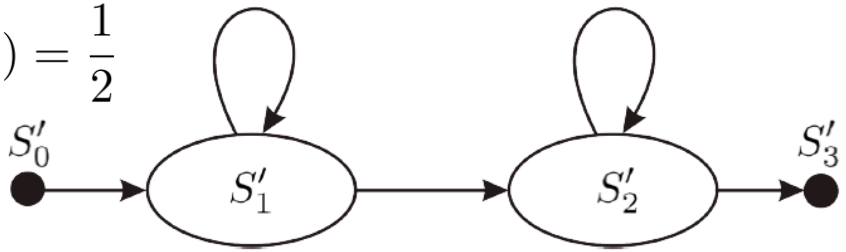
- **Parallel connection of HMMs (basic units to words):**
  - For the parallel connection of HMMs, the transition probabilities and the a-priori probabilities of the single HMM have to be connected.
  - Example of the parallel connection of two left – right models.
    This may by necessary to connect two phones of the same phoneme, e.g. voiced and unvoiced “s”, or different pronunciation of “r”.

$$p(H_1) = \frac{1}{2}$$



$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$p(H_2) = \frac{1}{2}$$

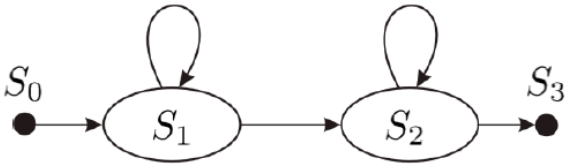


$$\boldsymbol{A}' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Connecting HMMs

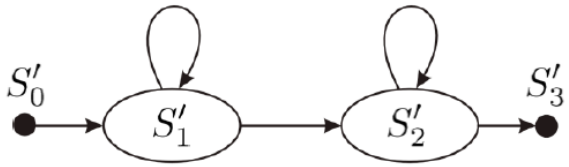
- **Parallel connection of HMMs (basic units to words):**
  - Example of the parallel connection of two left – right models.

$$p(H_1) = \boxed{\frac{1}{2}}$$



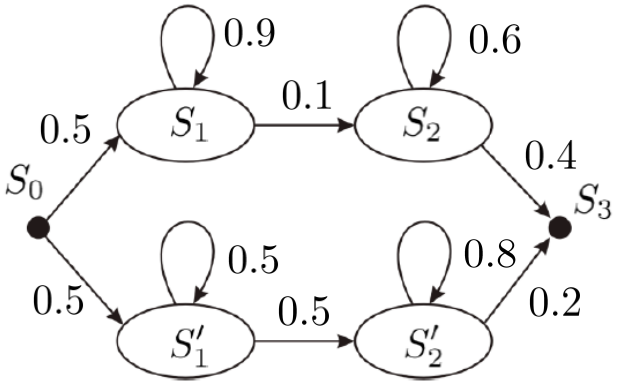
$$\boldsymbol{A} = \begin{bmatrix} 0 & \boxed{1} & \boxed{0} & 0 \\ 0 & \boxed{0.9} & \boxed{0.1} & \boxed{0} \\ 0 & \boxed{0} & \boxed{0.6} & \boxed{0.4} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$p(H_2) = \boxed{\frac{1}{2}}$$



$$\boldsymbol{A}' = \begin{bmatrix} 0 & \boxed{1} & \boxed{0} & \boxed{0} \\ 0 & \boxed{0.5} & \boxed{0.5} & \boxed{0} \\ 0 & \boxed{0} & \boxed{0.8} & \boxed{0.2} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

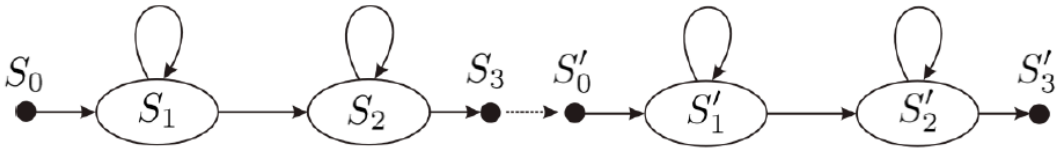
$$\boldsymbol{A}_{\text{par}} = \begin{array}{c} \begin{matrix} S_0 & S_1 & S_2 & S'_1 & S'_2 & S_3 \end{matrix} \\ \begin{bmatrix} 0 & \boxed{0.5} & \boxed{0} & \boxed{0.5} & \boxed{0} & \boxed{0} \\ 0 & \boxed{0.9} & \boxed{0.1} & \boxed{0} & \boxed{0} & \boxed{0} \\ 0 & \boxed{0} & \boxed{0.6} & \boxed{0} & \boxed{0} & \boxed{0.4} \\ 0 & 0 & 0 & \boxed{0.5} & \boxed{0.5} & \boxed{0} \\ 0 & 0 & 0 & \boxed{0} & \boxed{0.8} & \boxed{0.2} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$



Connecting HMMs

- **Serial connection of HMMs (basic units to words):**
  - Example of the serial connection of two left – right models, e.g. two basic units to one word.

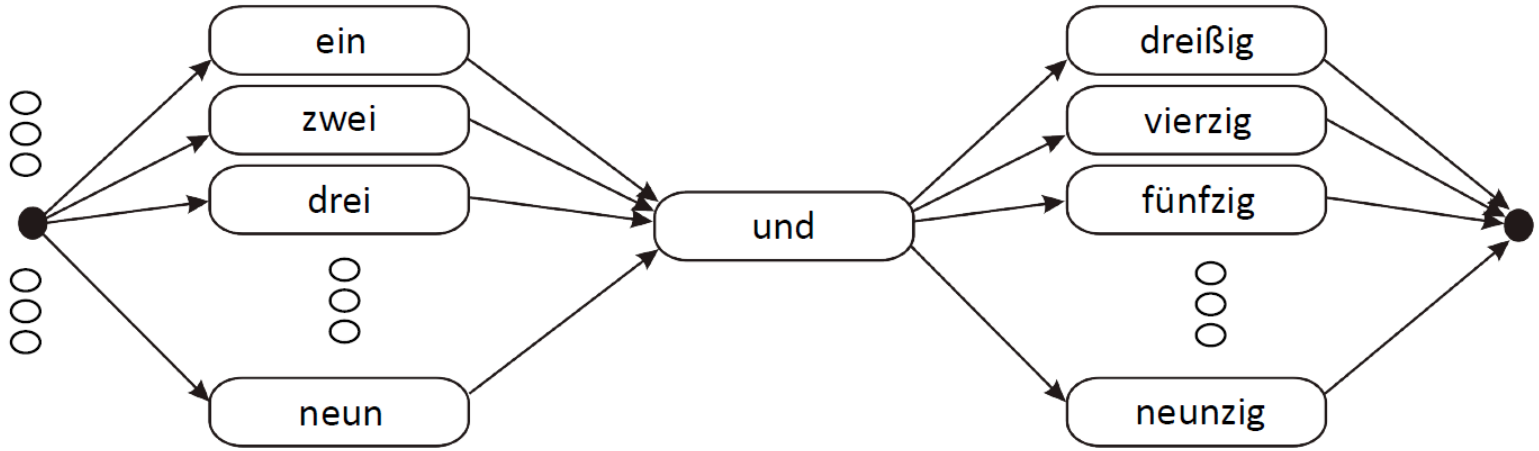
$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.6 & 0.4 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{A}' = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\boldsymbol{A}_{\text{ser}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Connecting HMMs

- **Generating an active vocabulary (e.g., the recognizer expects a double-digit number)**
  - The HMMs should be concatenated rather efficiently.
  - Example:



DNN based speech recognition

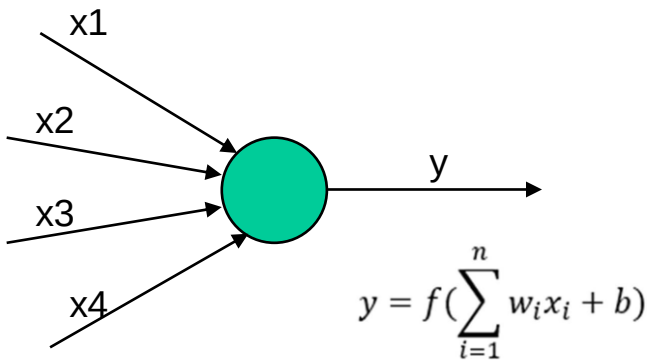
TECHNISCHE UNIVERSITÄT DARMSTADT

- **DNN processing**
  - Deep neural networks (DNN) are a machine learning approach for non-linear decision or regression tasks.
  - There are several basic structures of DNN processing which became popular in the last years for different kinds of applications.
  - The “scientific” aspect is lower since the design of DNN processing is based on many trial-and-error procedures.

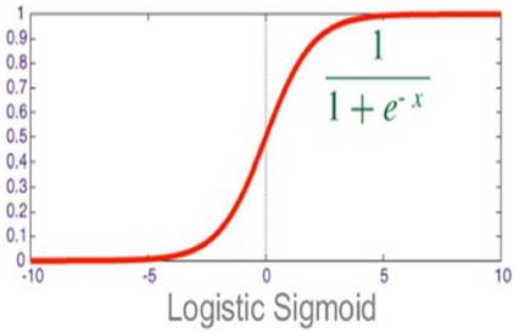
- **DNN processing for speech recognition**
  - Many different procedures
  - Here we will have a look at two:
    - DNN processing for modeling the observation probabilities in HMMs
    - DNN processing as end-to-end sequence: Connectionist Temporal Classification

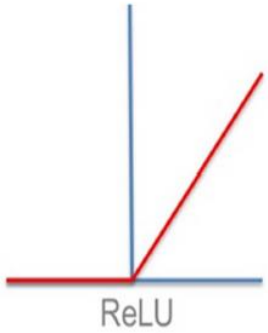
DNN – Basic concept

- The Neuron as basis unit of a DNN:
  Weighted combination a non-linear mapping to the output.



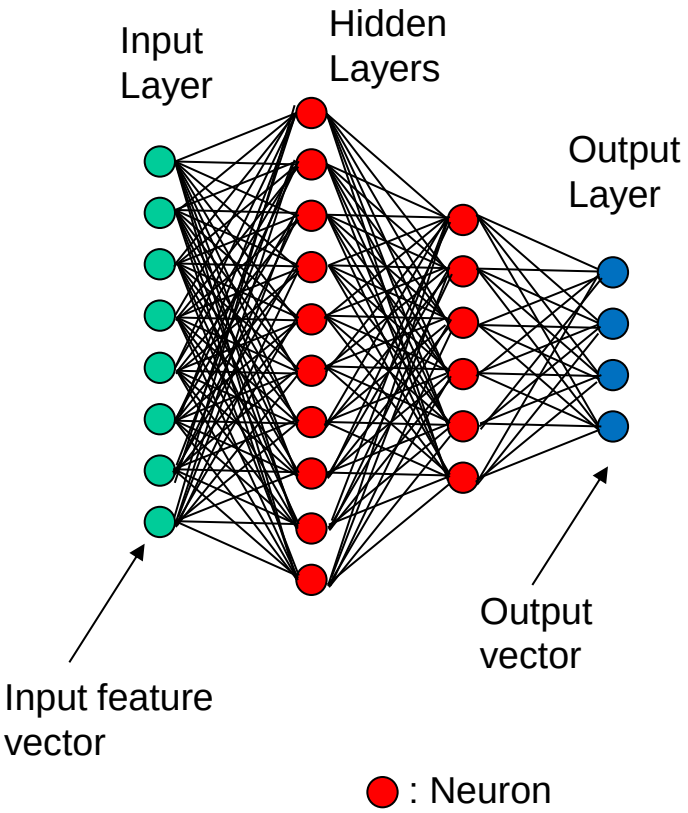
- Different non-linear functions f() may be used.
  Most applies ones: Sigmoid or ReLU (rectified linear unit)





DNN – different concepts

- The MultiLayer Perceptron (MLP):

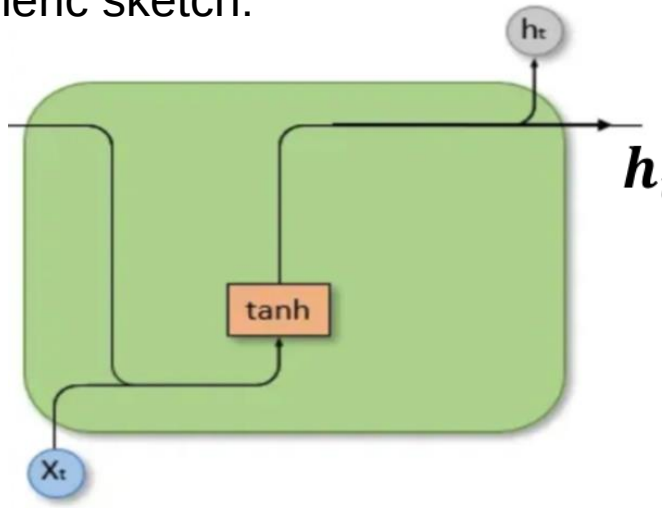


MLP:
Each neuron is connected to all neurons of the previous layer:
=> **Feedforward and unidirectional DNN.**

MLP training:
The weights and the bias values are trained with training data based on stochastic gradient descent.
Overfitting needs to be avoided. E.g. by dropout (randomly removing some neurons during training). This avoids “co-adaptation of neurons”.

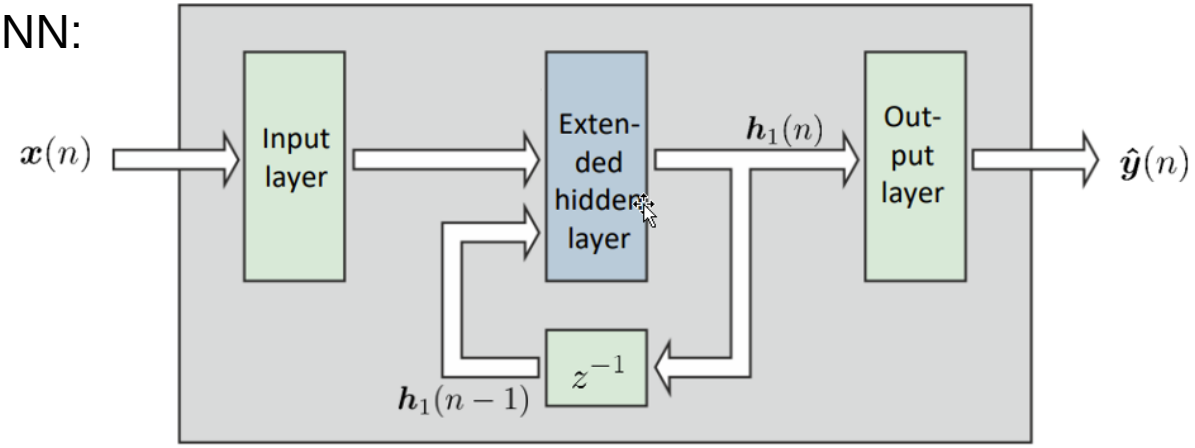
RNN – Recurrent Neural Networks

- Generic sketch:



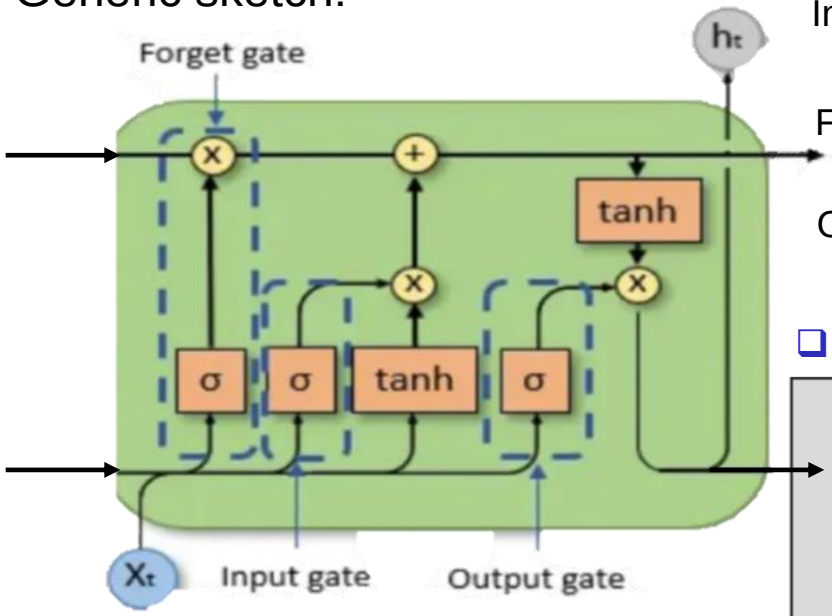
$$\boldsymbol{h}_t(n) = \tanh(\boldsymbol{W}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b})$$

- Integration in a DNN:



LSTM – Long Short-Term Memory Networks

- Generic sketch:



Input gate:

$$\boldsymbol{i}_t(n) = \sigma(\boldsymbol{W}_{\text{in}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{in}})$$

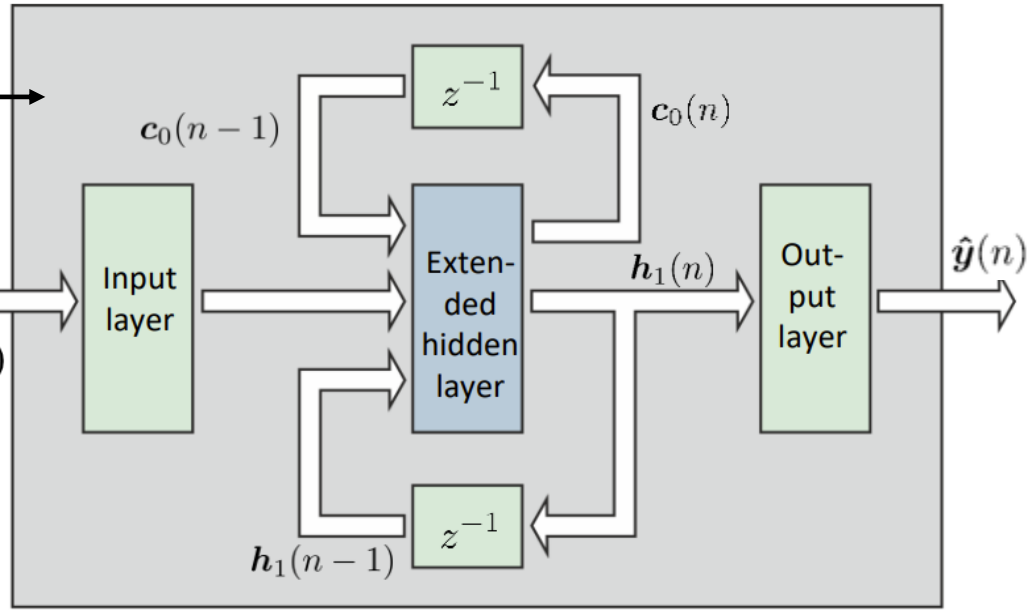
Forget gate:

$$\boldsymbol{f}_t(n) = \sigma(\boldsymbol{W}_{\text{in}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{for}})$$

Output gate:

$$\boldsymbol{o}_t(n) = \sigma(\boldsymbol{W}_{\text{out}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{out}})$$

- Integration in a DNN:



Cell state update:

$$\bar{\boldsymbol{c}}_t(n) = \tanh(\boldsymbol{W}_{\text{c}}[\boldsymbol{x}^T(n), \boldsymbol{h}_t(n-1)^T]^T + \boldsymbol{b}_{\text{c}})$$

$$\boldsymbol{c}_t(n) = \text{diag}\{\boldsymbol{f}_t(n)\}\boldsymbol{c}_t(n-1) + \text{diag}\{\boldsymbol{i}_t(n)\}\,\bar{\boldsymbol{c}}_t(n)$$

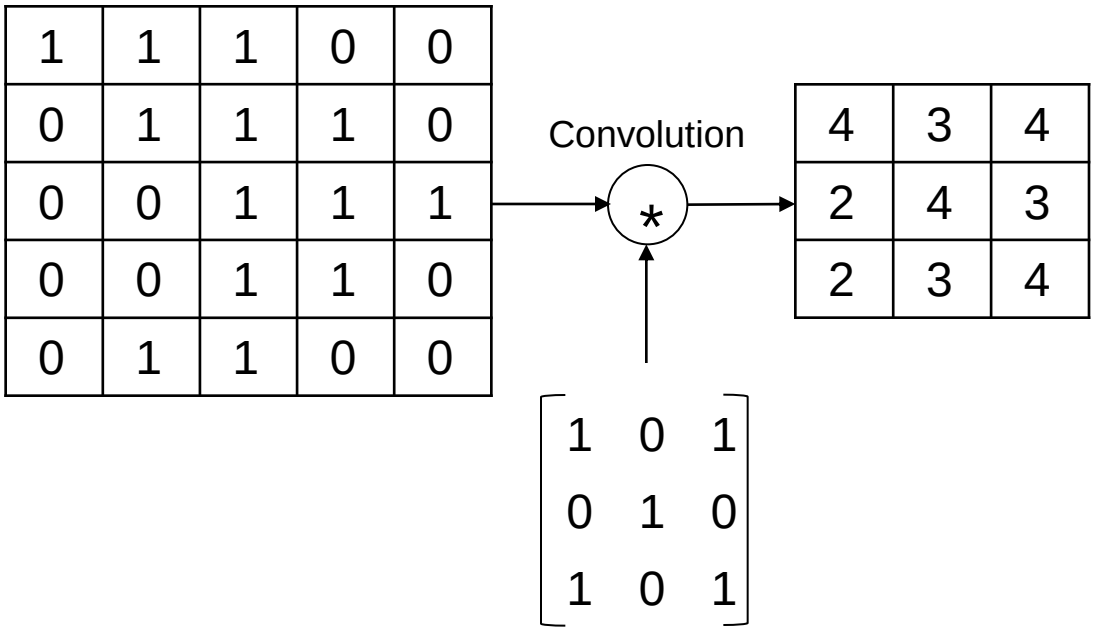
Hidden state update:

$$\boldsymbol{h}_t(n) = \text{diag}\{\tanh(\boldsymbol{c}_t(n)\}\boldsymbol{o}_t(n)$$

DNN – different concepts

- The Convolutive Neural Network (CNN):

Example of the application of a convolutional layer:



DNN – different concepts

- Pooling layer (as part of CNNs):

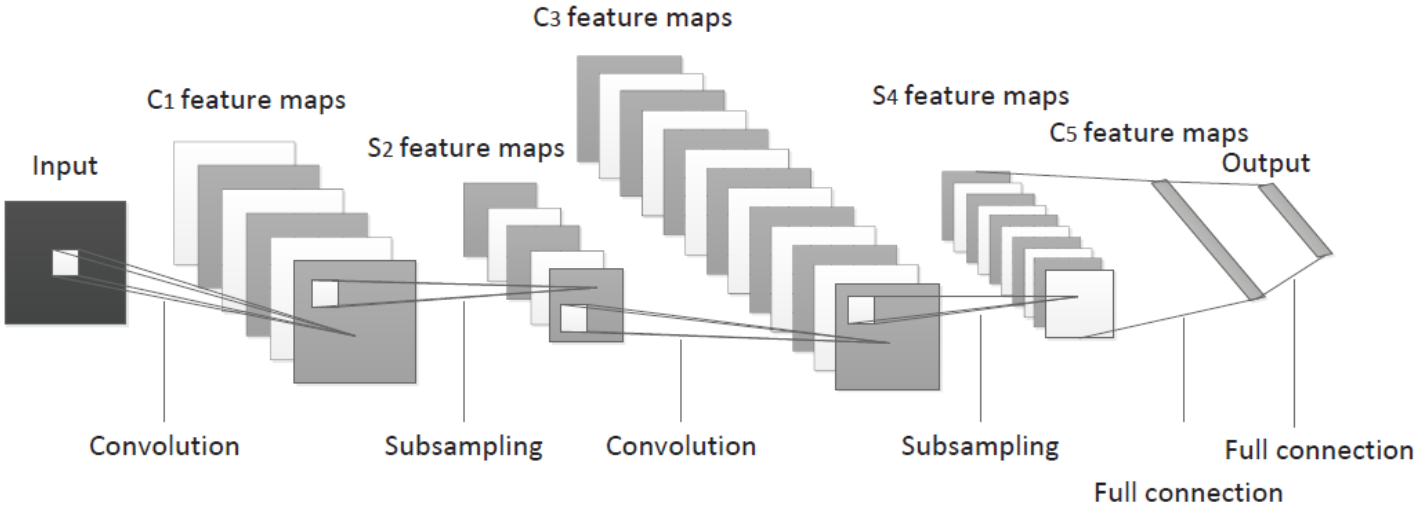
Example of the application of a pooling layer (non-linear „max“ – function):

1	3	5	1
5	2	6	7
3	5	2	3
1	7	1	4

2x2 max pooling

5	7
7	4

DNN – An example of convolution and pooling layers

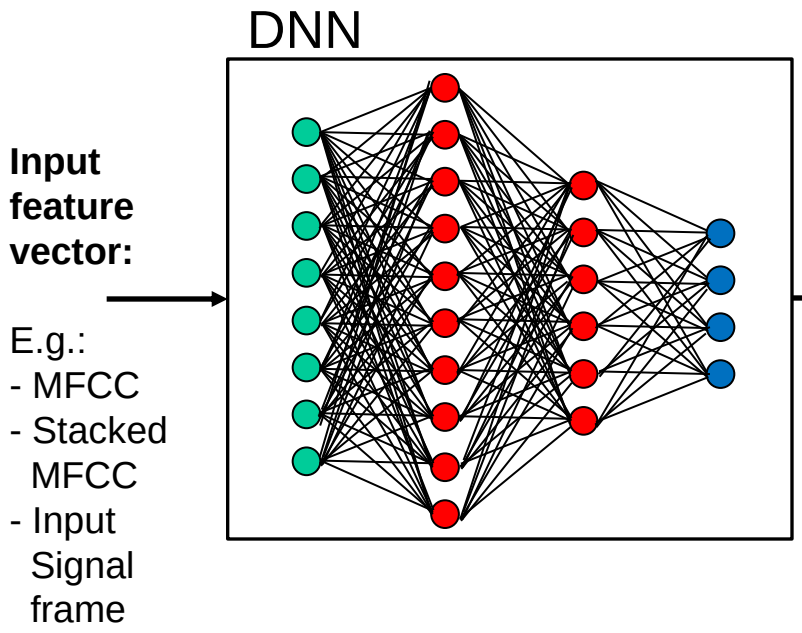


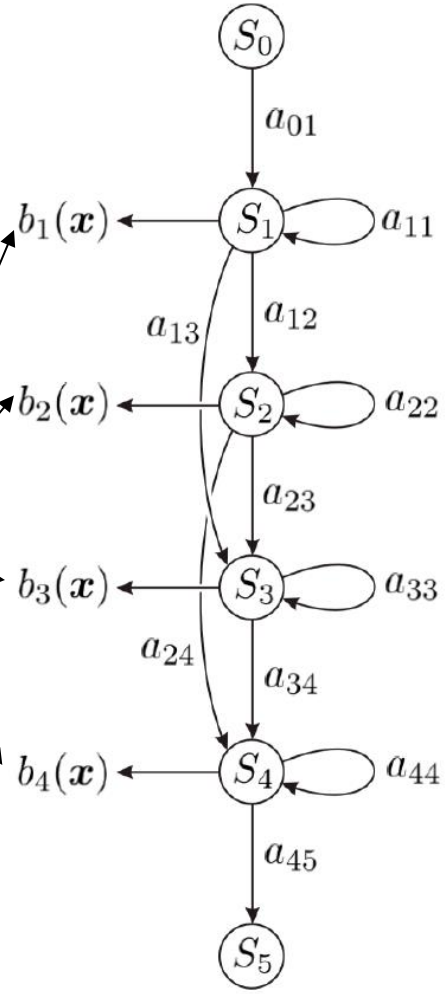
Ref: Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, Fuad E. Alsaadi, A survey of deep neural network architectures and their applications,Neurocomputing, Volume 234, 2017, Pages 11-26,

DNN – HMM Structure

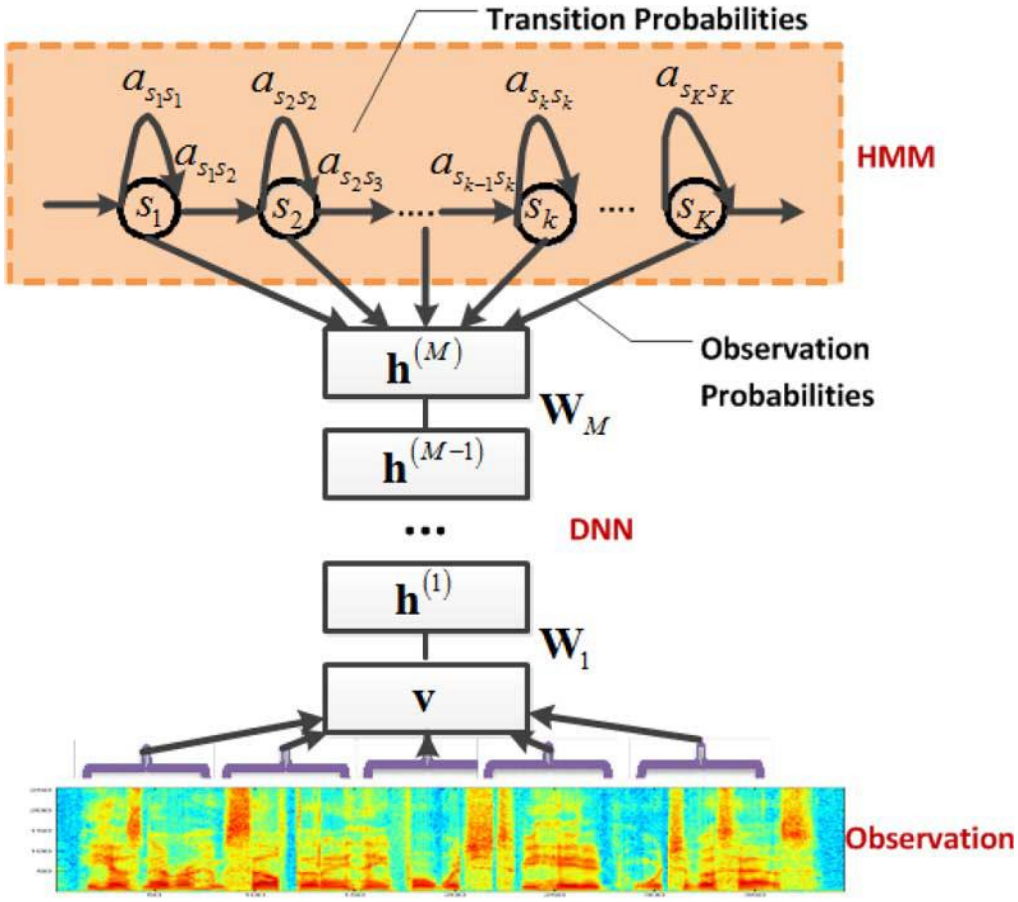
DNN modeling the observation probabilities

- Based on a trained DNN the observation probabilities can be calculated for an input (feature) vector.
- DNN-HMM structure
  (against the known GMM-HMM structure).





DNN modeling the observation probabilities



- Diagram of a hybrid architecture employing a deep neural network.
  The HMM models the sequential property of the speech signal, and the DNN models the scaled observation likelihood of all the senones (tied tri-phone states).
  The same DNN is replicated over different points in time.

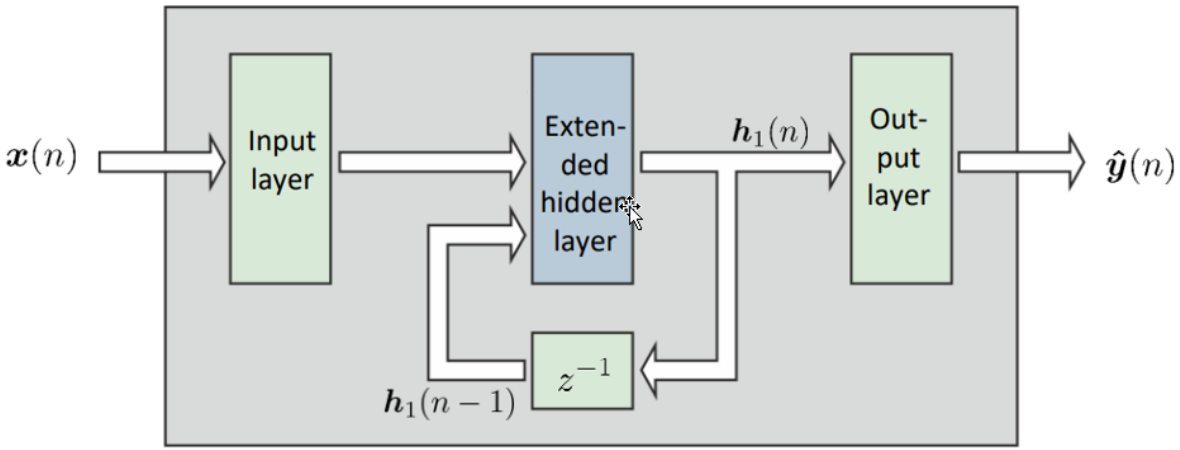
Ref: G. E. Dahl, D. Yu, L. Deng and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 20, no. 1, pp. 30-42, Jan. 2012

TECHNISCHE UNIVERSITÄT DARMSTADT

End-to-end model:
Connectionist Temporal Classification (CTC)

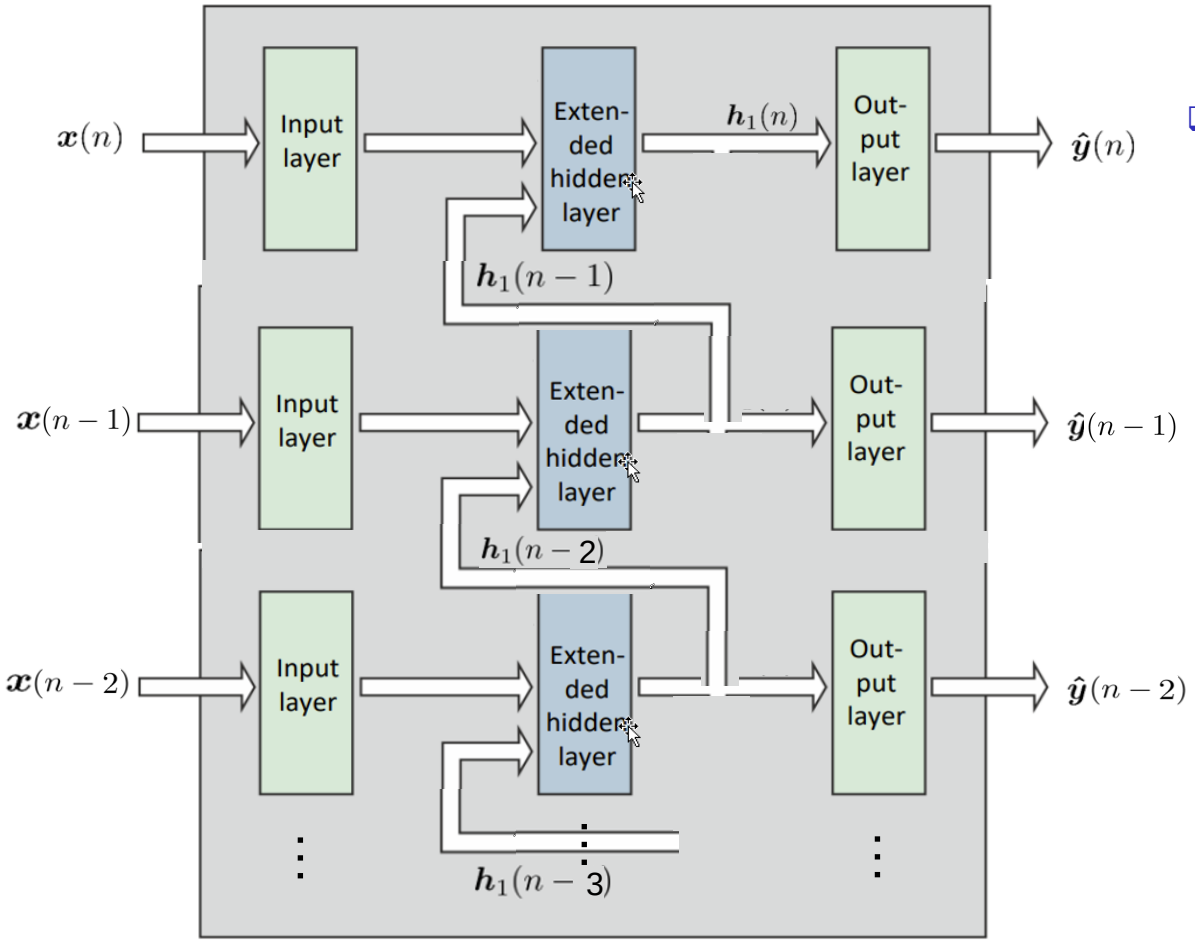
Fast decoding, efficient, practical relevance

RNNs: From uni-directional to bi-directional concepts

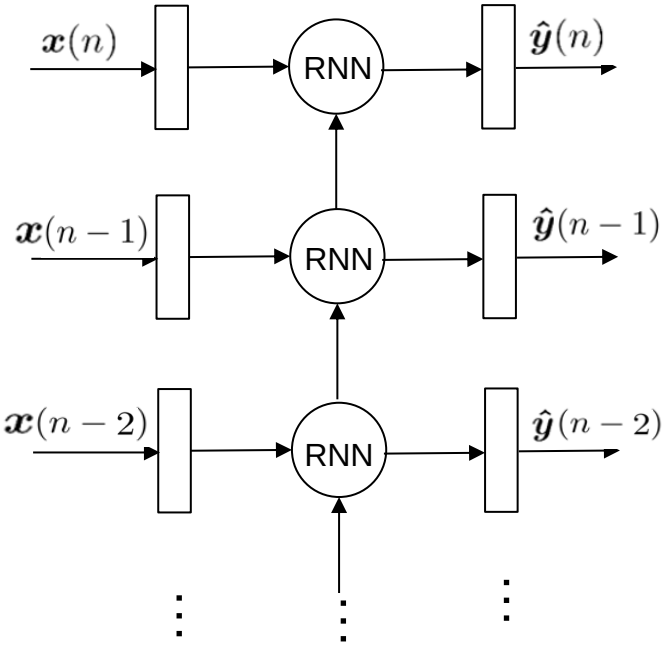


RNNs: From uni-directional to bi-directional concepts

- Showing the time history:

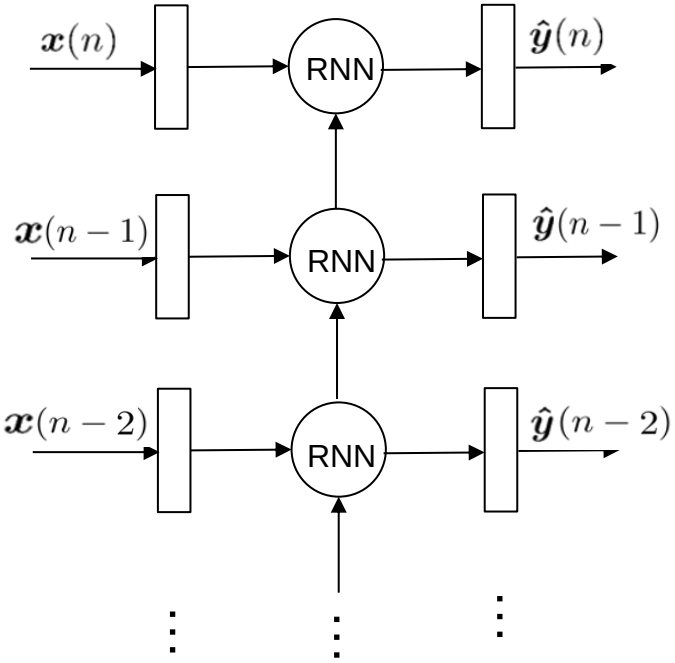


- Simplified version:

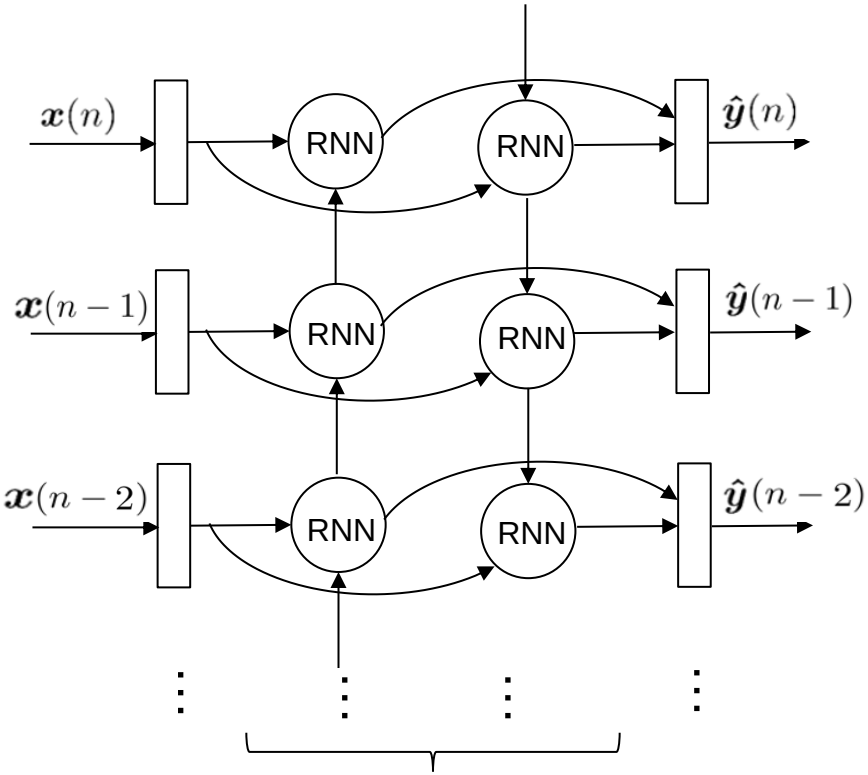


RNNs: From uni-directional to bi-directional concepts

- Simplified version of a uni-directional RNN:



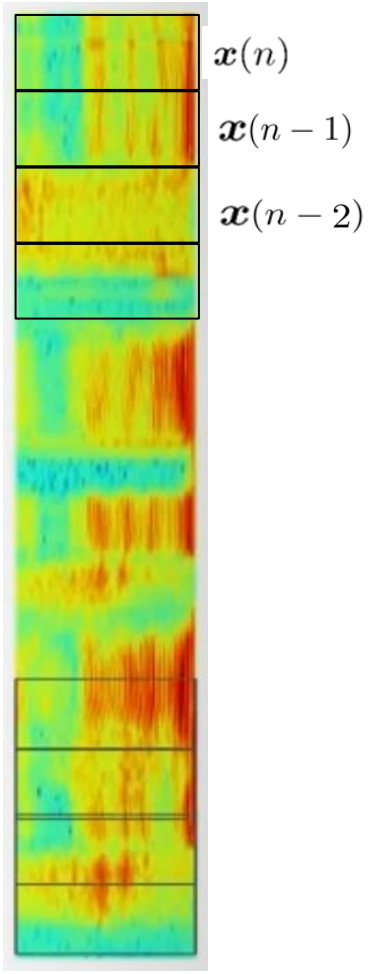
- Simplified version of a bi-directional RNN:

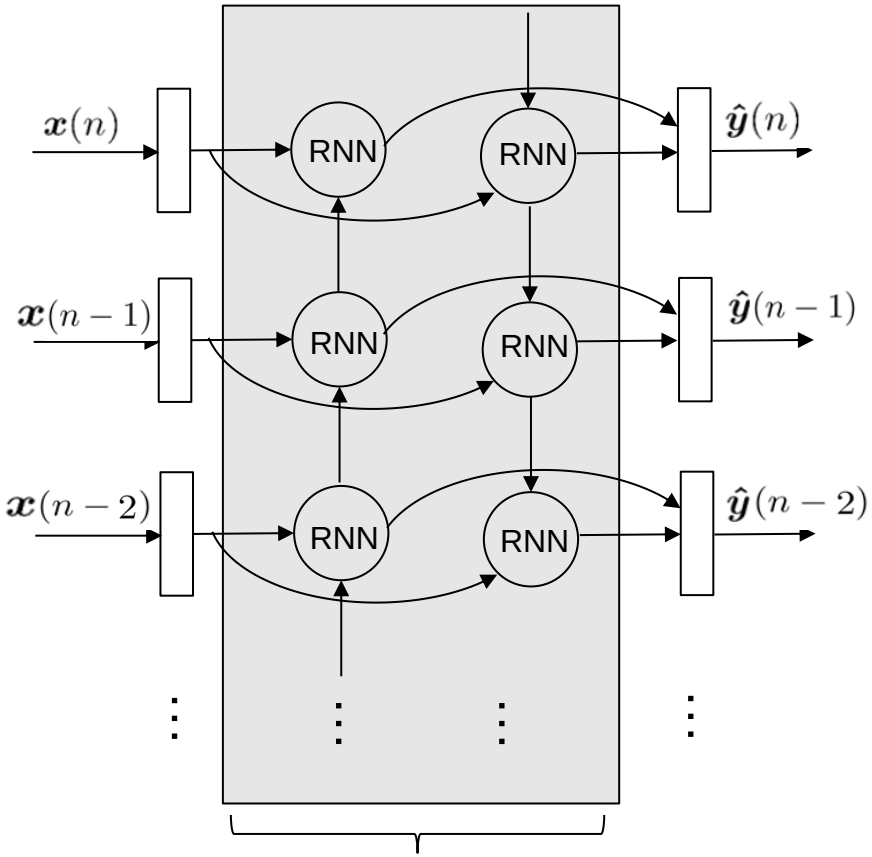


One bi-directional RNN layer

RNN Concept

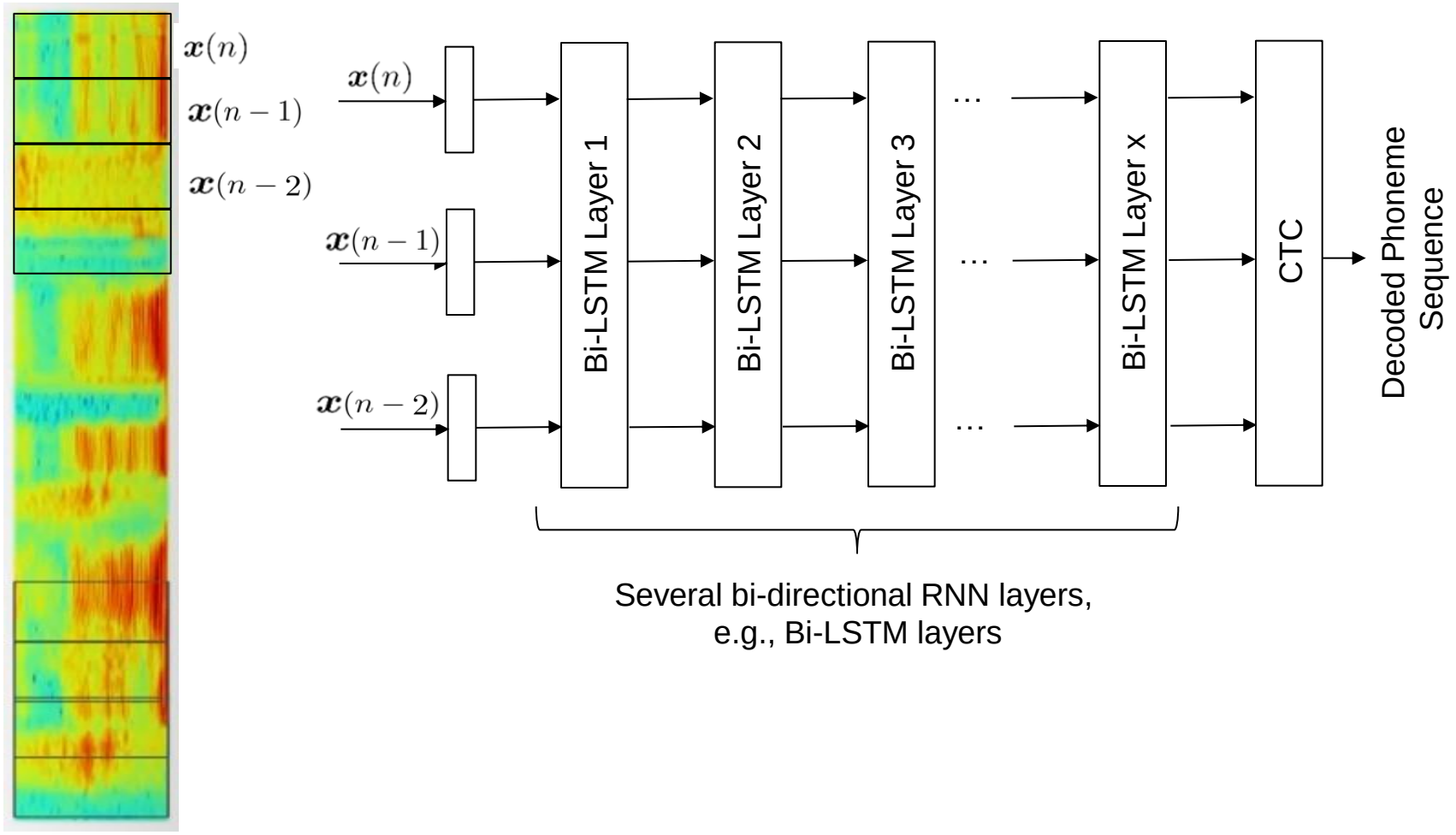
Using the MFCC feature vectors:





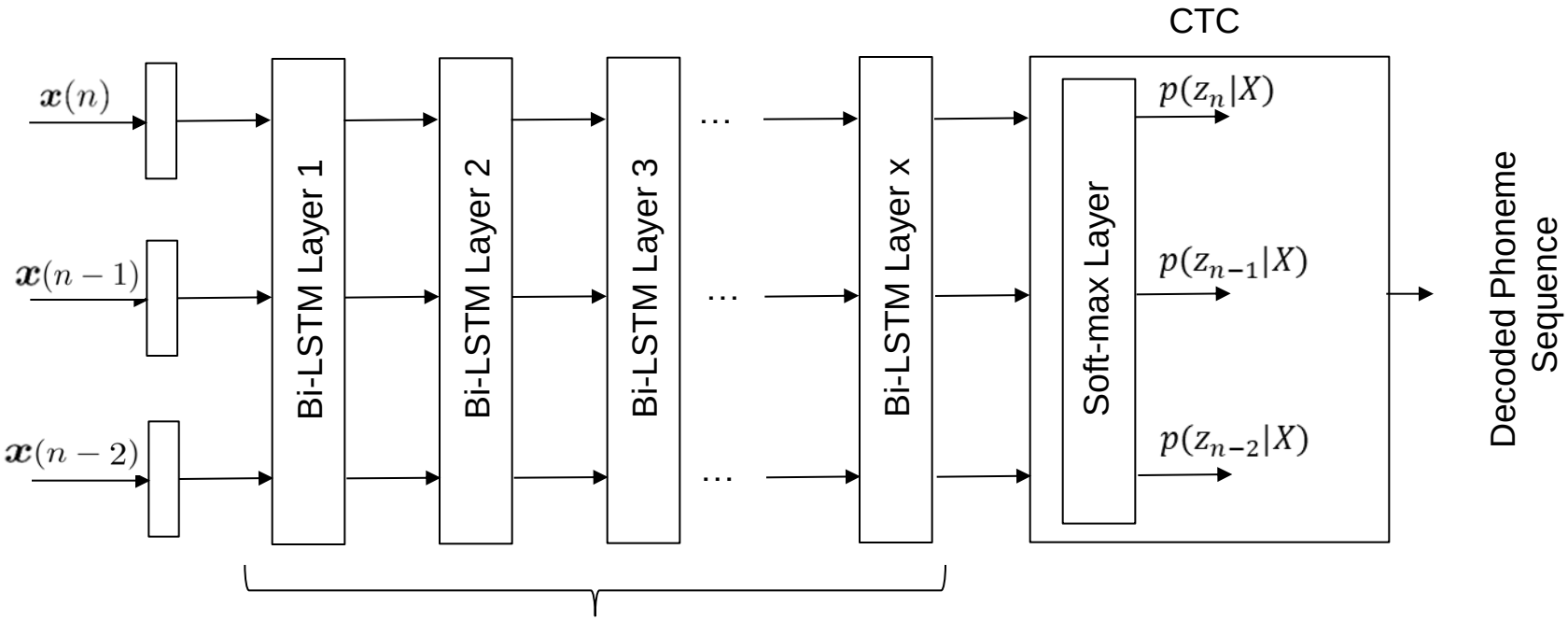
One bi-directional RNN layer

RNN + Connectionist Temporal Classification (CTC)



Connectionist Temporal Classification (CTC)

- CTC consists of a soft-max layer and a decoding problem.
- The Soft-max layer provides the probabilities for all phonemes (vocabulary scoring) on top of the phonemes there is also the probability for a blank <b> calculated (equivalent to a phoneme pause).



Connectionist Temporal Classification (CTC)

- The probability of the entire phoneme sequence is the product of all output phonemes:

$$p(Z|X) = \prod_{n=0}^{N-1} p(z_n|X)$$

- This is done under the assumption that there are no token dependencies, i.e., independence assumption:

Pros: Simple and fast model
Can be combined with a language model

Cons: Price on performance

Connectionist Temporal Classification (CTC)

- The output sequence is as long as the feature sequence.
- There is a shortening necessary to come to a typical phoneme sequence which models words.

- If we want to calculate the probability for a certain word, we need to sum up all the probabilities for output sequences which could model this word.

$$p(W|X) = \sum_{Z \in f^{-1}(W)} p(Z|X)$$

with:

$$p(Z|X) = \prod_{n=0}^{N-1} p(z_n|X)$$

$$p(W|X) = \sum_{Z \in f^{-1}(W)} \prod_{n=0}^{N-1} p(z_n|X)$$

Connectionist Temporal Classification (CTC)

- $p(W|X)$ is the sum of the probabilities of all Z that reduce to W:

$$p(W|X) = \sum_{Z \in f^{-1}(W)} p(Z|X)$$

- Example for the word CAT (based on a sequency length of 6):

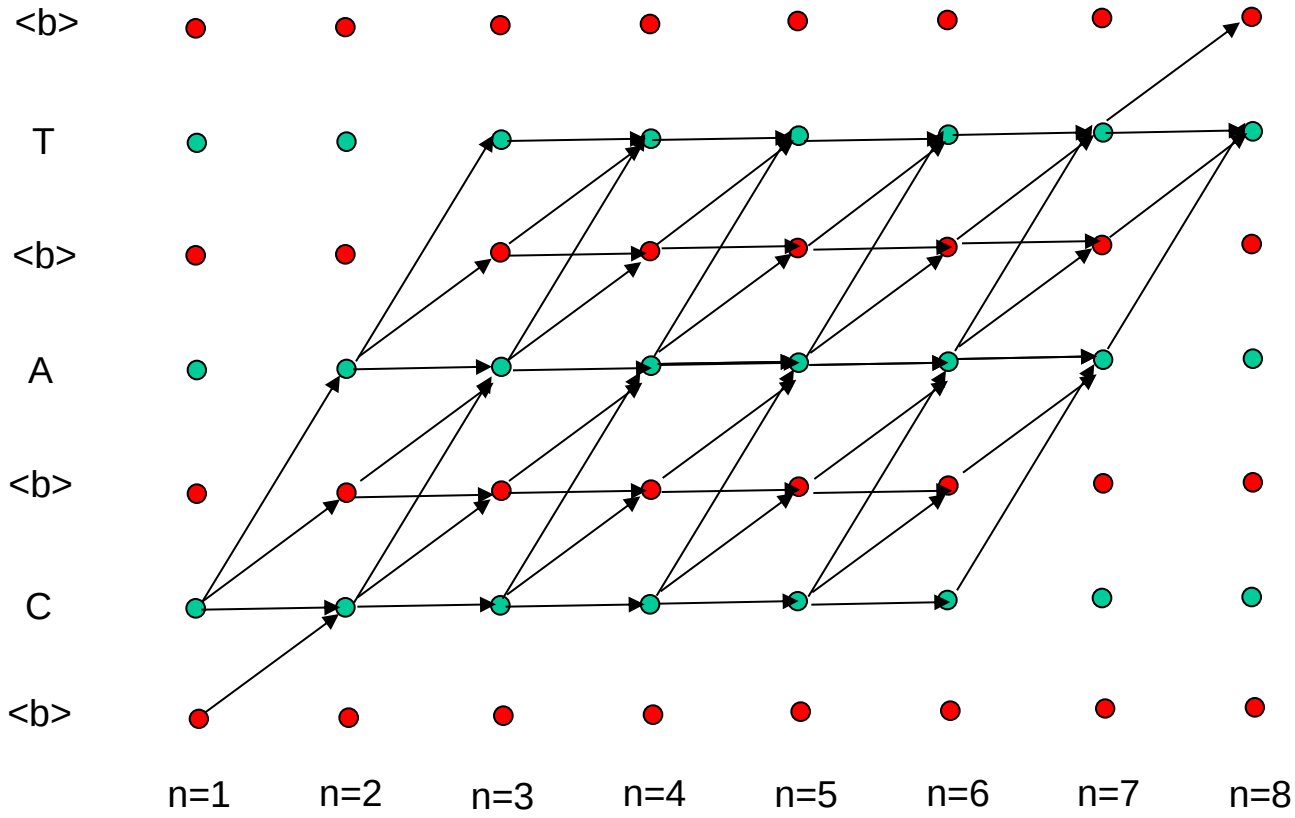
Z = C <b> A <b> T <b> => CAT
= C C <b> A <b> T => CAT
= <b> C A <b> T <b> => CAT
= ….

Removing all blanks <b> and repeats results in CAT for all of the given sequences.

- How can we efficiently calculate the probability for all paths?

Connectionist Temporal Classification (CTC)

- How to efficiently calculate the probability sum of all the possible sequences?:
  => Use the known “Trellis” concept.



Procedure according to the forward and/or backward algorithm:

Only two start / end options for each word/ phoneme sequence.

Known Concept:
Sum the probabilities of the paths arriving at each node.
Allows to keep the complexity of long path handling low.

Greedy search

TECHNISCHE UNIVERSITÄT DARMSTADT

Repetition:

- $p(W|X)$ is the sum of the probabilities of all Z that reduce to W:

$$p(W|X) = \sum_{Z \in f^{-1}(W)} p(Z|X)$$

$$\widehat{W} = \underset{W}{\operatorname{argmax}}\, p(W|X)$$

Greedy search

Greedy search:

- Two step simplification:
  - 1) Select the path with the highest probability

$$\hat{Z} = \underset{Z}{\operatorname{argmax}}\, p(Z|X) \quad \Longrightarrow \quad \widehat{W}: \text{corresponding } to\ \hat{Z} \text{ with all <b> and repeats removed}$$

  - 2) Chose for each output the phoneme with the highest probability.

$$\hat{Z} = \underset{Z}{\operatorname{argmax}}\, p(Z|X)$$

$$\hat{Z} \approx \left(\underset{z_1}{\operatorname{argmax}}\, p(Z|X), \underset{z_2}{\operatorname{argmax}}\, p(Z|X), \underset{z_3}{\operatorname{argmax}}\, p(Z|X), \ \ldots \right)$$

Using the conditional independence assumption: $p(Z|X) = \prod_{n=0}^{N-1} p(z_n|X)$

$$\hat{Z} \approx \left(\underset{z_1}{\operatorname{argmax}}\, p(z_1|X), \underset{z_2}{\operatorname{argmax}}\, p(z_2|X), \underset{z_3}{\operatorname{argmax}}\, p(z_3|X), \ \ldots \right)$$

Greedy search

Greedy search:

- Only one path selected by:

$$\hat{Z} \approx \left(\underset{z_1}{\operatorname{argmax}}\, p(z_1|X), \underset{z_2}{\operatorname{argmax}}\, p(z_2|X), \underset{z_3}{\operatorname{argmax}}\, p(z_3|X), \ \ldots \right)$$

- Resulting in one sequence of Z, e.g.,

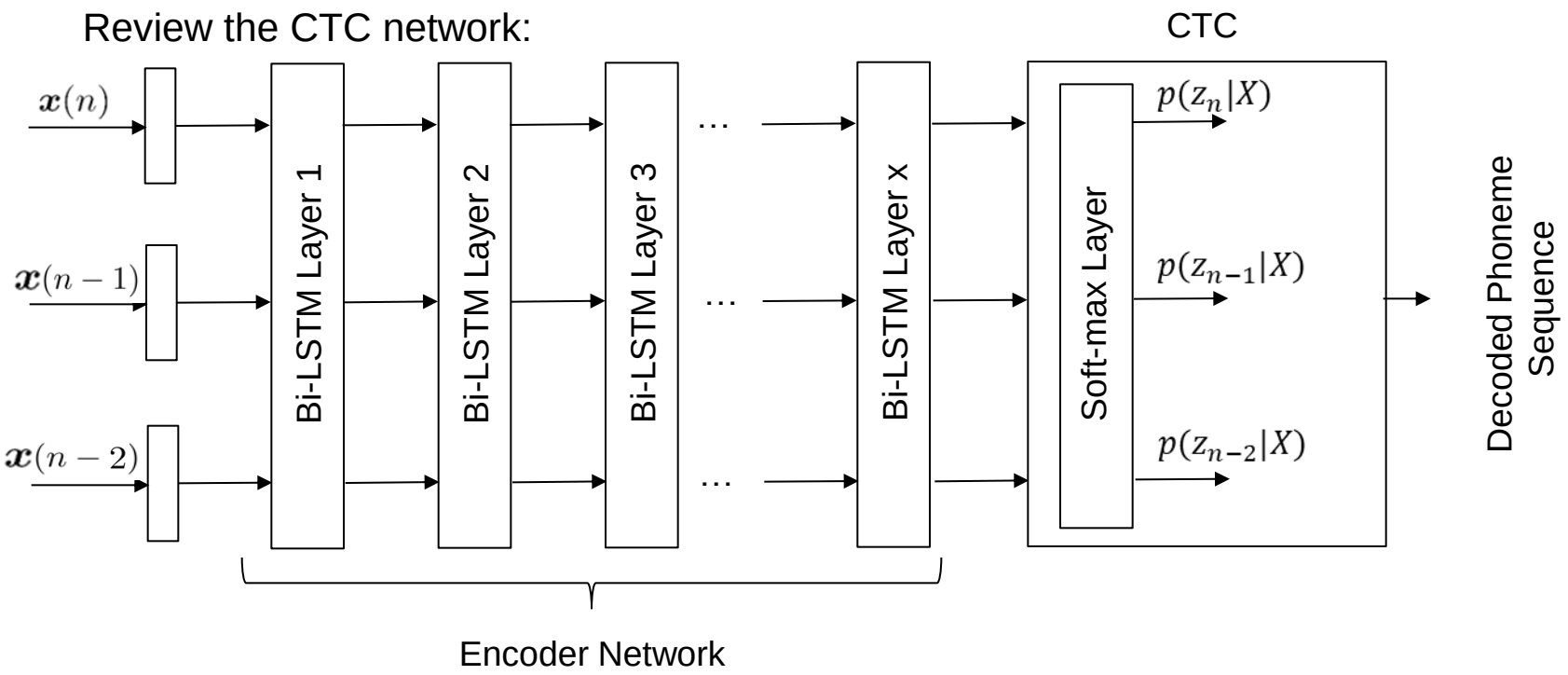
$\hat{Z}$ => CAA<b>T $\Longrightarrow$ $\hat{W}$ => CAT

which can directly converted to the word estimate by removing all <b> and repeats.

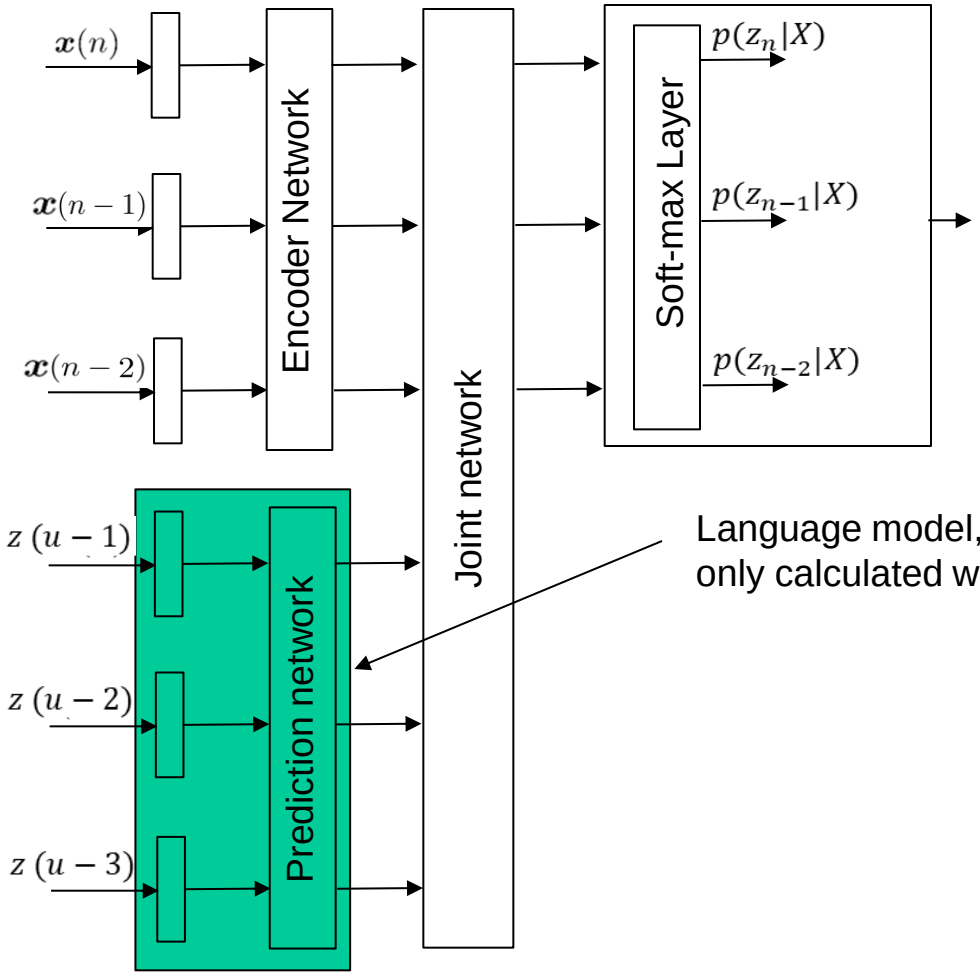
Combination with a language model

Language model combination

- Target: Increase accuracy



Combination with a language model



Summary

- Repetition
- The three basic problems of HMMs
  - Evaluation problem
  - Decoding problem
  - Model parameter estimation problem
- Model parameter estimation problem:
  - Introduction of backward probability
  - Estimation of the transition probabilities
  - Estimation of the observation probabilities.
- Application of speech recognition systems
- DNN based speech recognition

References

TECHNISCHE UNIVERSITÄT DARMSTADT

Speech recognition:

[1] B. Pfister, T. Kaufman: *Sprachverarbeitung*, Springer, 2008

[2] C. M. Bishop: *Pattern Recognition and Machine Learning,* Springer, 2006

[3] G. Ruske: *Automatische Spracherkennung –Methoden der Klassifikation und Merkmalsextraktion*, Oldenbourg, 1988

[4] S. Euler: *Grundkurs Spracherkennung*, Vieweg, 2006
=> good explanations and a complete demo system to free download

[5] D. Fohr, O. Mella, I. Illina. New Paradigm in Speech Recognition: Deep Neural Networks. IEEE International Conference on Information Systems and Economic Intelligence, Apr 2017,Marrakech, Marocco

CTC:

[6] https://www.youtube.com/watch?v=3MjIkWxXigM

[7] https://www.youtube.com/watch?v=sR6_bZ6VkAg

[8] https://www.youtube.com/watch?v=00GHpnRTAQE

[9] H. Sak, et. al.: “Learning acoustic frame labeling for speech recognition with recurrent neural networks” IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), 2015