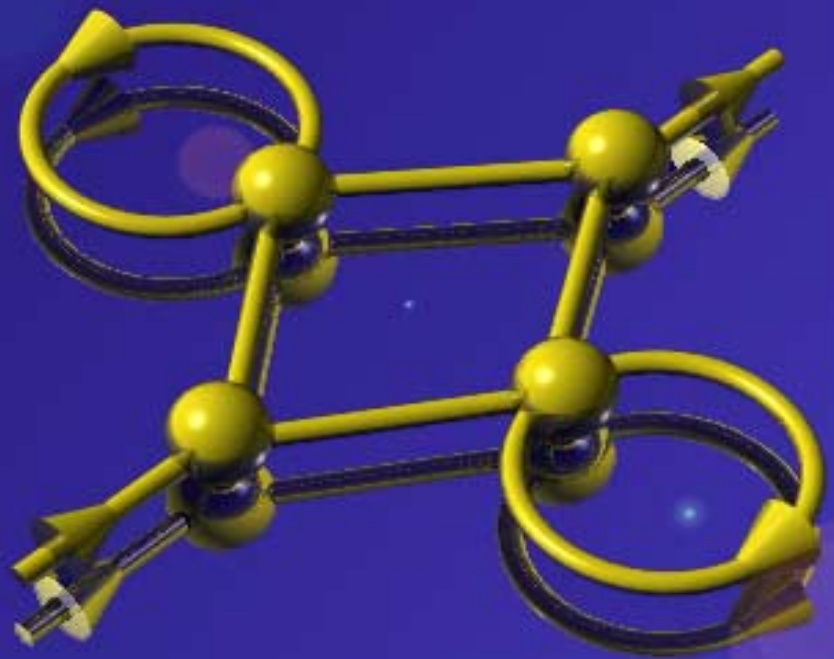
Automatic Speech Recognition

Using Recurrent Neural Networks

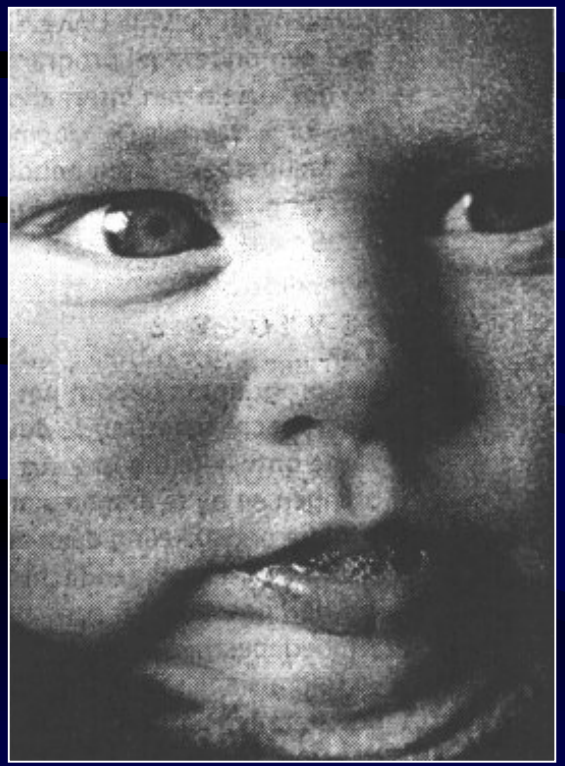


Daan Nollen

8 januari 1998

TUDelft
Delft University of Technology
Faculty of Information Technology and Systems
Knowledge-Based Systems

“Baby’s van 7 maanden onderscheiden al wetten in taalgebruik”



- Abstract rules versus statistical probabilities
- Example “meter”

“Er staat 30 graden op de *meter*”

“De breedte van de weg is hier 2 *meter*”

- Context very important
  - sentence (grammer)
  - word (syntax)
  - phoneme

Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Problem definition:

Create an ASR using only ANNs

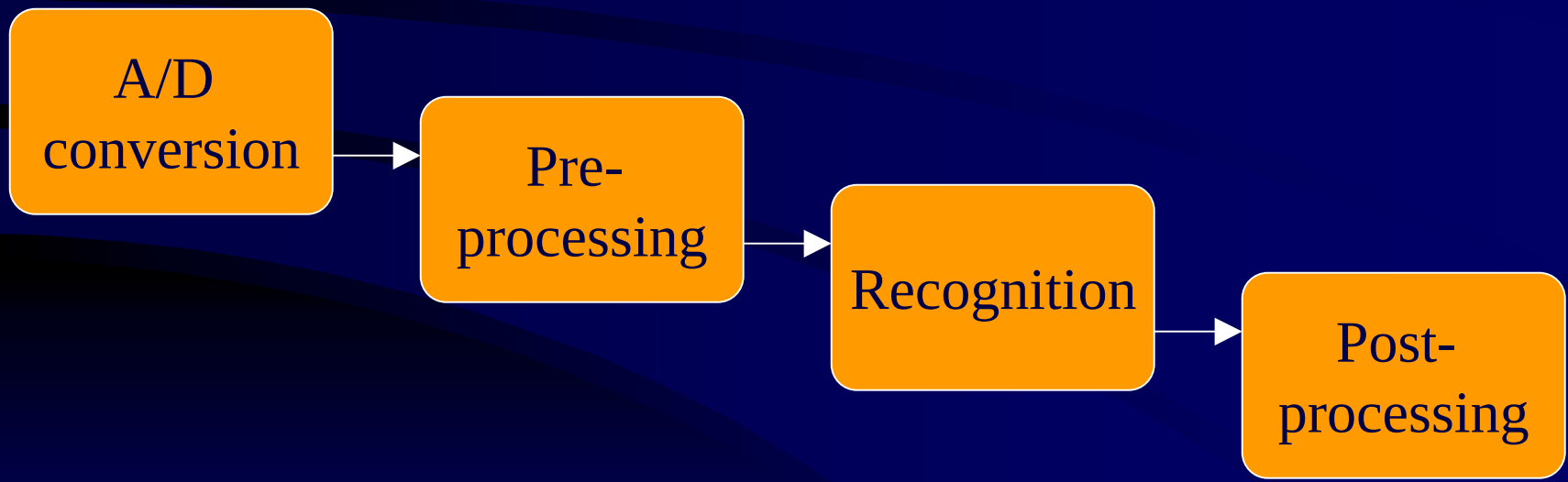
- Study Recnet ASR and train Recogniser
- Design and implement an ANN workbench
- Design and implement an ANN phoneme postprocessor
- Design and implement an ANN word postprocessor

Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Automatic Speech Recognition (ASR)

- ASR contains 4 phases



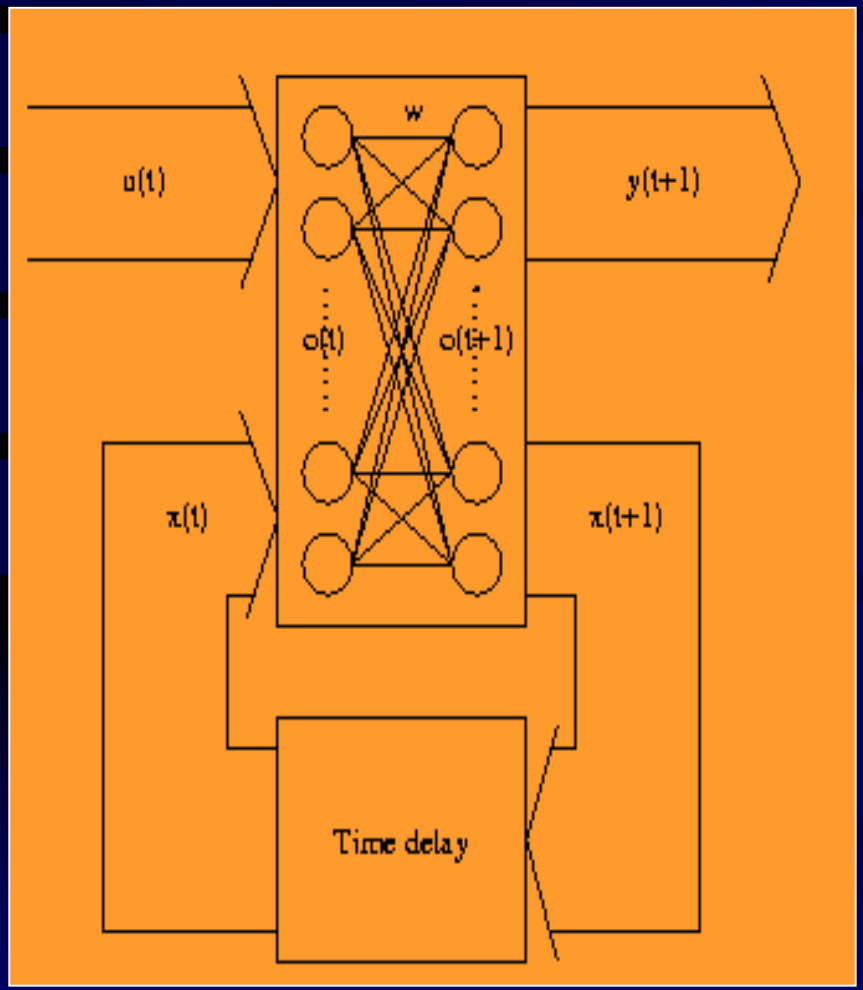
Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Recnet ASR

- TIMIT Speech Corpus US-American samples
- Hamming windows 32ms with 16ms overlap
- Preprocessor based on Fast Fourier Transformation

Recnet Recogniser



- *u(t)* external input (output preprocessor)
- *y(t+1)* external output (phoneme probabilities)
- *x(t)* internal input
- *x(t+1)* internal output

$x(0) = 0$

$x(1) = f(u(0), x(0))$

$x(2) = f(u(1), x(1))$
$= f(u(1), f(u(0), x(0)))$

Performance Recnet Recogniser

- Output Recnet probability vector

/eh/ = 0.1
/ih/ = 0.5
/ao/ = 0.3
/ae/ = 0.1

- 65 % phonemes correctly labelled
- Network trained on parallel nCUBE2

Training Recnet on the nCUBE2

- Per trainingcycle over 78 billion floating point calculations
- Training time SUN 144 mhz 700 hours (1 month)
- nCUBE2 (32 processors) processing time 70 hours

Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Recnet Postprocessor

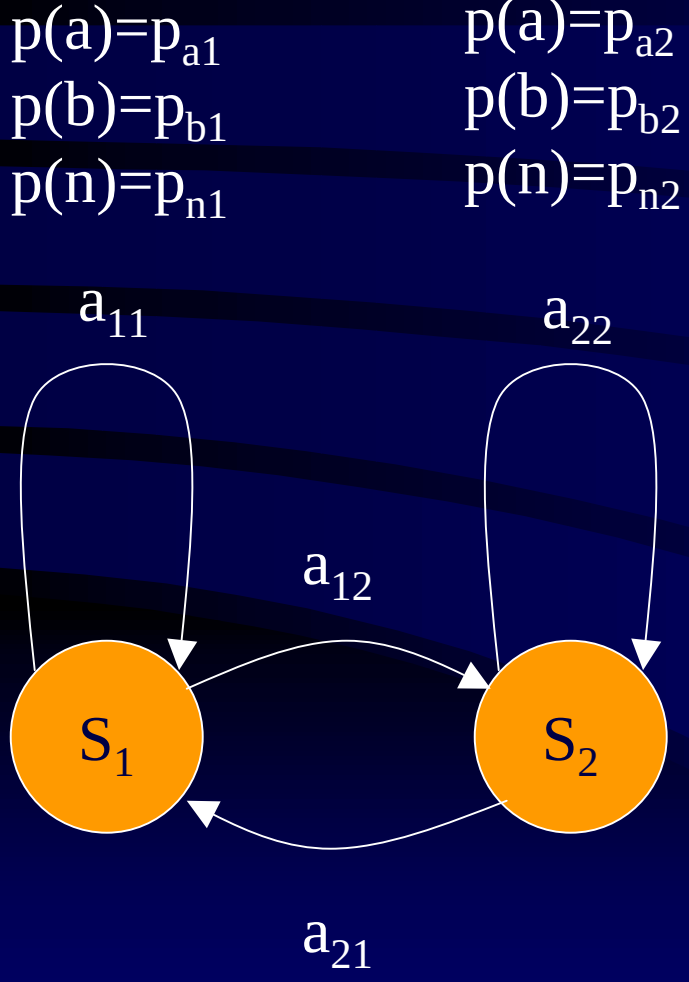
- Perfect recogniser

/h/ /h/ /h/ /eh/ /eh/ /eh /eh/ /l/ /l/ /ow/ /ow /ow/
/h/ /eh/ /l/ /ow/

- Recnet recogniser

/h/ /k/ /h/ /eh/ /ah/ /eh /eh/ /l/ /l/ /ow/ /ow /h/

Hidden Markov Models (HMMs)



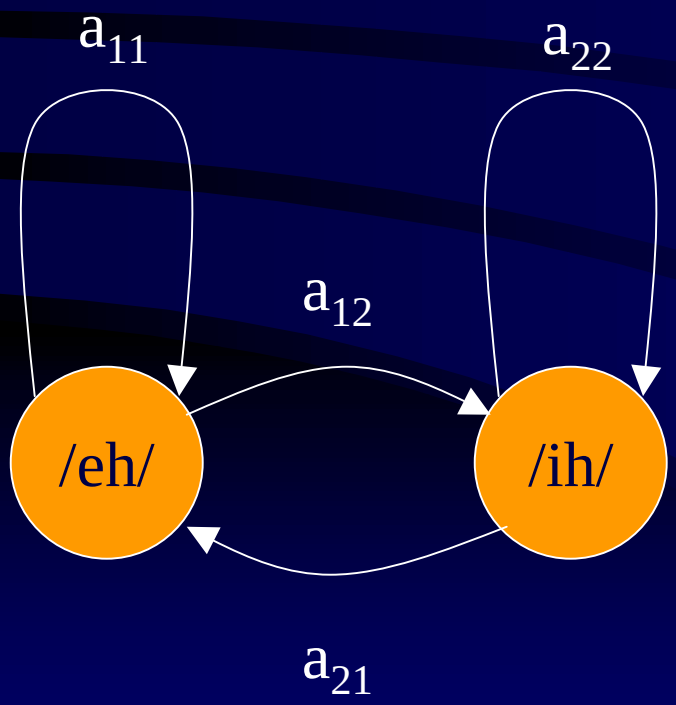
- Model stochastic processes
- Maximum a posteriori state sequence determined by

$$Q^* = \arg\max_Q \prod_{t=1}^{T} \Pr(q_t \mid q_{t-1})\, p(u_t \mid q_t)$$

- Most likely path by *Viterbi* algorithm

Recnet Postprocessing HMM

$p(/eh/)=y_{/eh/}$ $p(/eh/)=y_{/eh/}$
$p(/ih/)=y_{/ih/}$ $p(/ih/)=y_{/ih/}$
$p(/ao/)=y_{/ao/}$ $p(/ao/)=y_{/ao/}$



- Phonemes represented by 62 states
- Transitions trained on correct TIMIT labels
- Output Recnet phoneme recogniser determines output probabilities
- Very suitable for smoothing task

Scoring a Postprocessor

- Number of frames fixed, but number of phonemes not

/a/ /a/ /b/ /b/	/a/ /b/
/a/ /c/ /b/ /b/	/a/ /c/ /b/

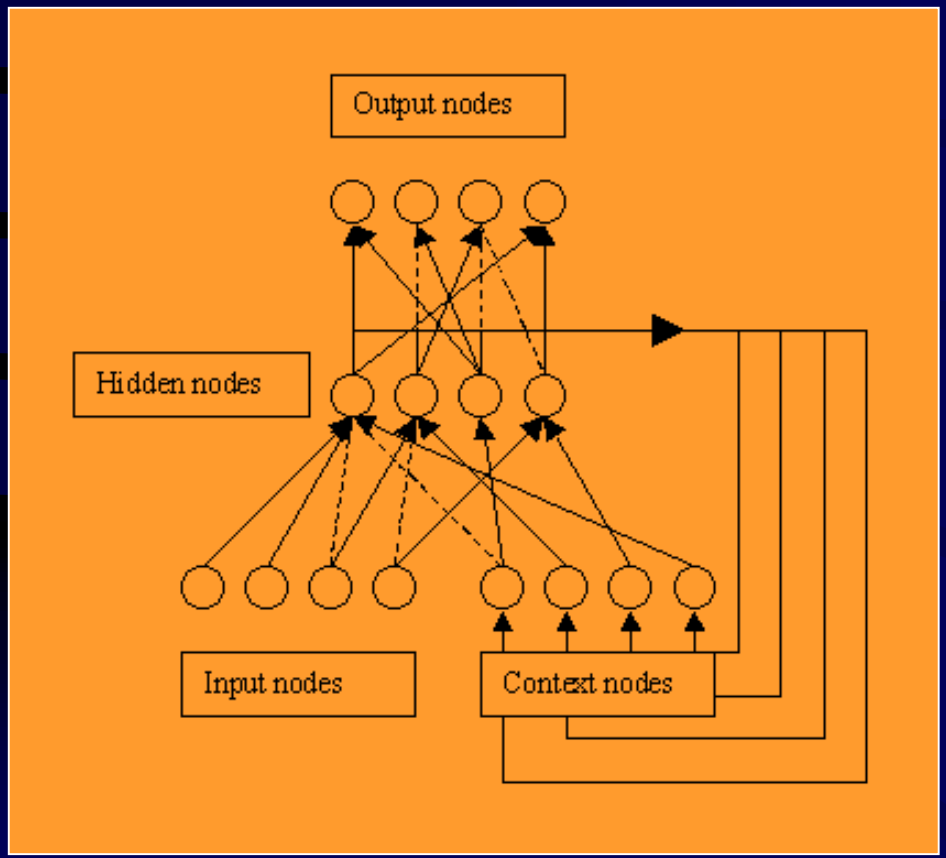
- Three types of errors
  - insertion /a/ /c/ /b/ /b/ → /a/ /c/ /b/
  - deletion /a/ /a/ /a/ /a/ → /a/ ...
  - substitution /a/ /a/ /c/ /c/ → /a/ /c/
- Optimal allignment

Scoring Recnet Postprocessing HMM

- No postprocessing vs Recnet HMM
- Removing repetitions
- Applying scoring algorithm

	Nothing	Recnet HMM
Correct	**79.9%**	**72.8%**
Insertion	**46.1%**	**3.7%**
Deletion	**17.6%**	**21.2%**
Substitution	**2.5%**	**6.0%**
Total errors	**66.2%**	**30.9%**

Elman RNN Phoneme Postprocessor(1)



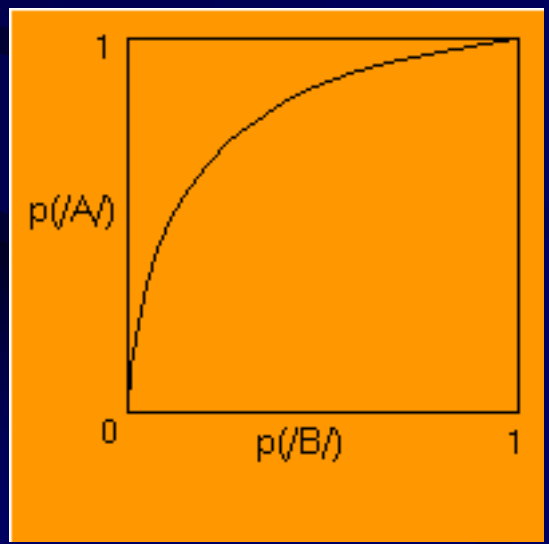
- Context helps to smooth the signal
- Probability vectors input
- Most likely phoneme output

RNN Phoneme Postprocessor(2)

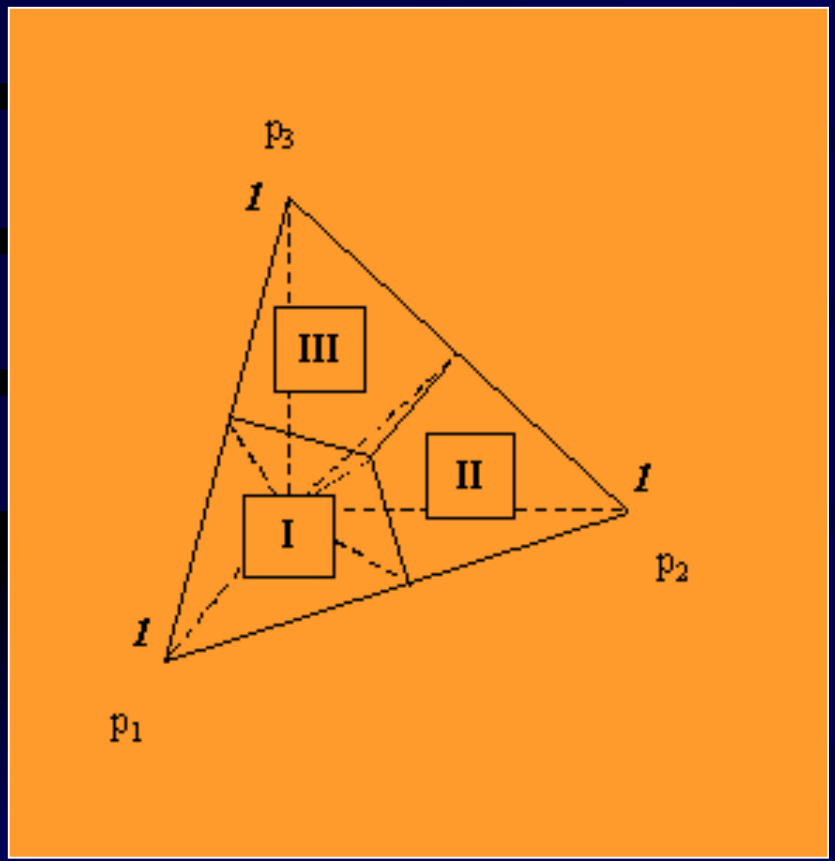
- Calculates conditional probability

$$\text{phon}_{\text{most likely (i)}} = \underset{0 \le i \le 61}{\text{argmax}}(p(\text{phon}_i \mid \text{input, context}))$$

- Probability changes through time



Training RNN Phoneme Postprocessor(1)



- Highest probability determines region
- 35 % in incorrect region
- Training dilemma:
  - Perfect data, context not used
  - Output Recnet, 35 % errors in trainingdata
- Mixing trainingset needed

Training RNN Phoneme Postprocessor(2)

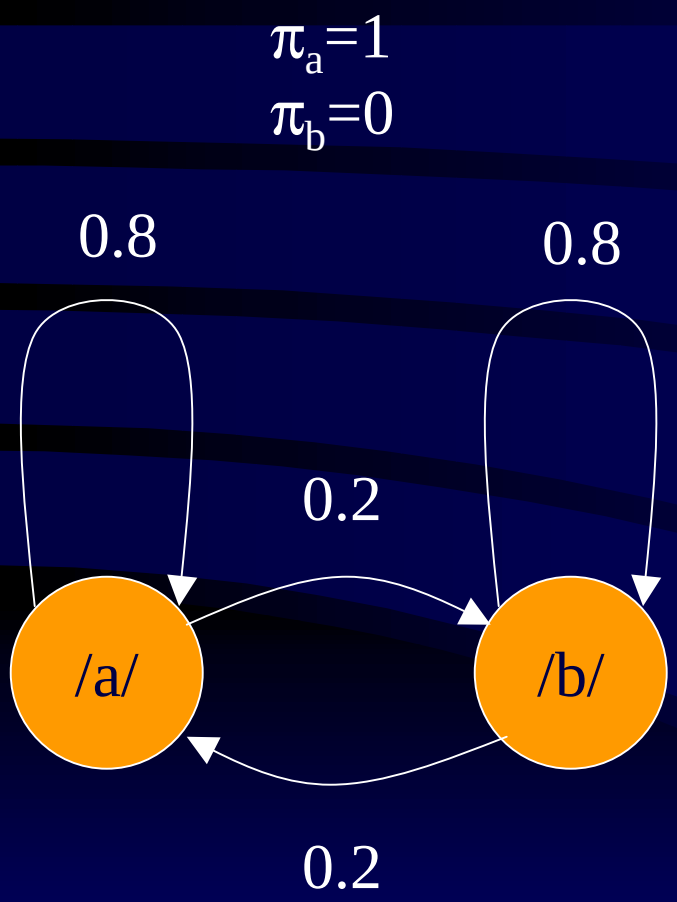
- Two mixing methods applied
  - I : Trainingset = norm(α Recnet + (1- α) Perfect prob)
  - II: Trainingset = norm(Recnet with $p(phn_{correct})=1$)

	Method I	Method II	Recnet HMM
Correct	74.8%	75.1%	72.8%
Insertion	23.5%	21.9%	3.7%
Deletion	21.4%	21.1%	21.2%
Substitution	3.8%	3.8%	6.0%
Total errors	48.7%	46.8%	30.9%

Conclusion RNN Phoneme Postprocessor

- Reduces 50% insertion errors
- Unfair competition with HMM
  - Elman RNN uses previous frames (context)
  - HMM uses preceding and successive frames

Example HMM Postprocessing



Frame 1: p(/a/)=0.1
p(/b/)=0.9

p(/a/, /a/) = 1·0.8·0.1 = 0.08
p(/a/, /b/) = 1·0.2·0.9 = 0.18

Frame 2: p(/a/)=0.9
p(/b/)=0.1

p(/a/, /a/, /a/) = 0.08·0.8·0.9 = 0.0576
p(/a/, /a/, /b/) = 0.08·0.2·0.1 = 0.0016
p(/a/, /b/, /a/) = 0.18·0.2·0.9 = 0.0324
p(/a/, /b/, /b/) = 0.18·0.8·0.1 = 0.0144

Conclusion RNN Phoneme Postprocessor

- Reduces 50% insertion errors
- Unfair competition with HMM
  - Elman RNN uses previous frames (context)
  - HMM uses preceding and succeeding frames
- PPRNN works real-time

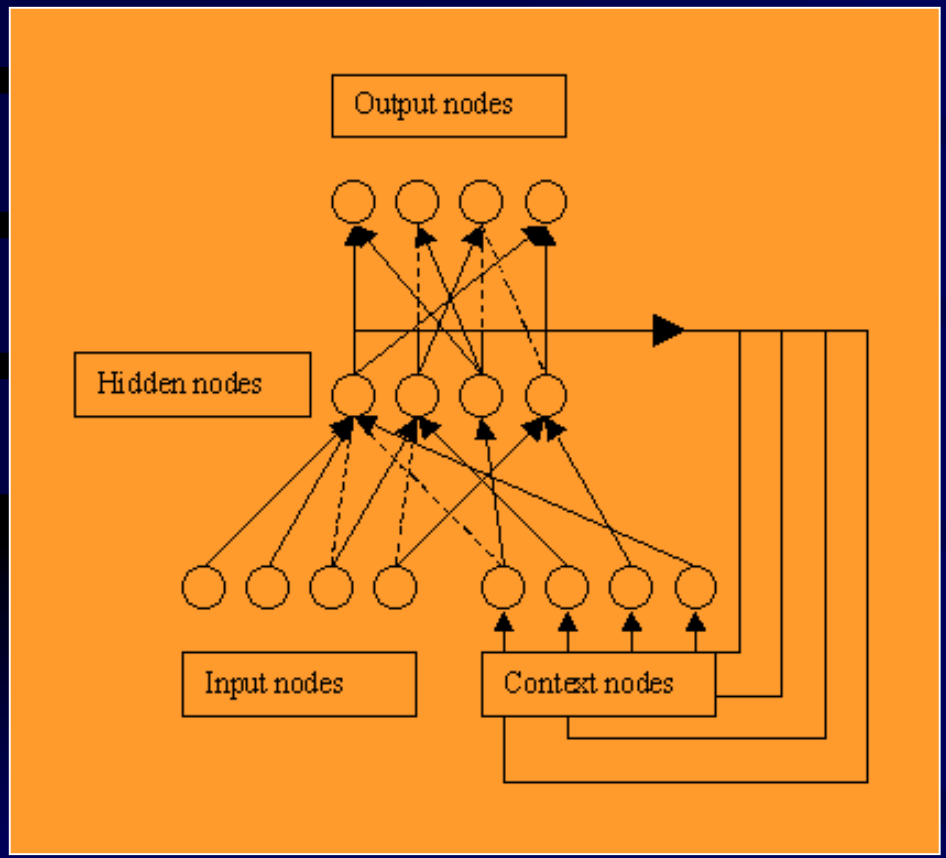
Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Word postprocessing

- Output phoneme postprocessor continuous stream of phonemes
- Segmentation needed to convert phonemes into words
- Elman Phoneme Prediction RNN (PPRNN) can segment stream and correct errors

Elman Phoneme Prediction RNN (PPRNN)

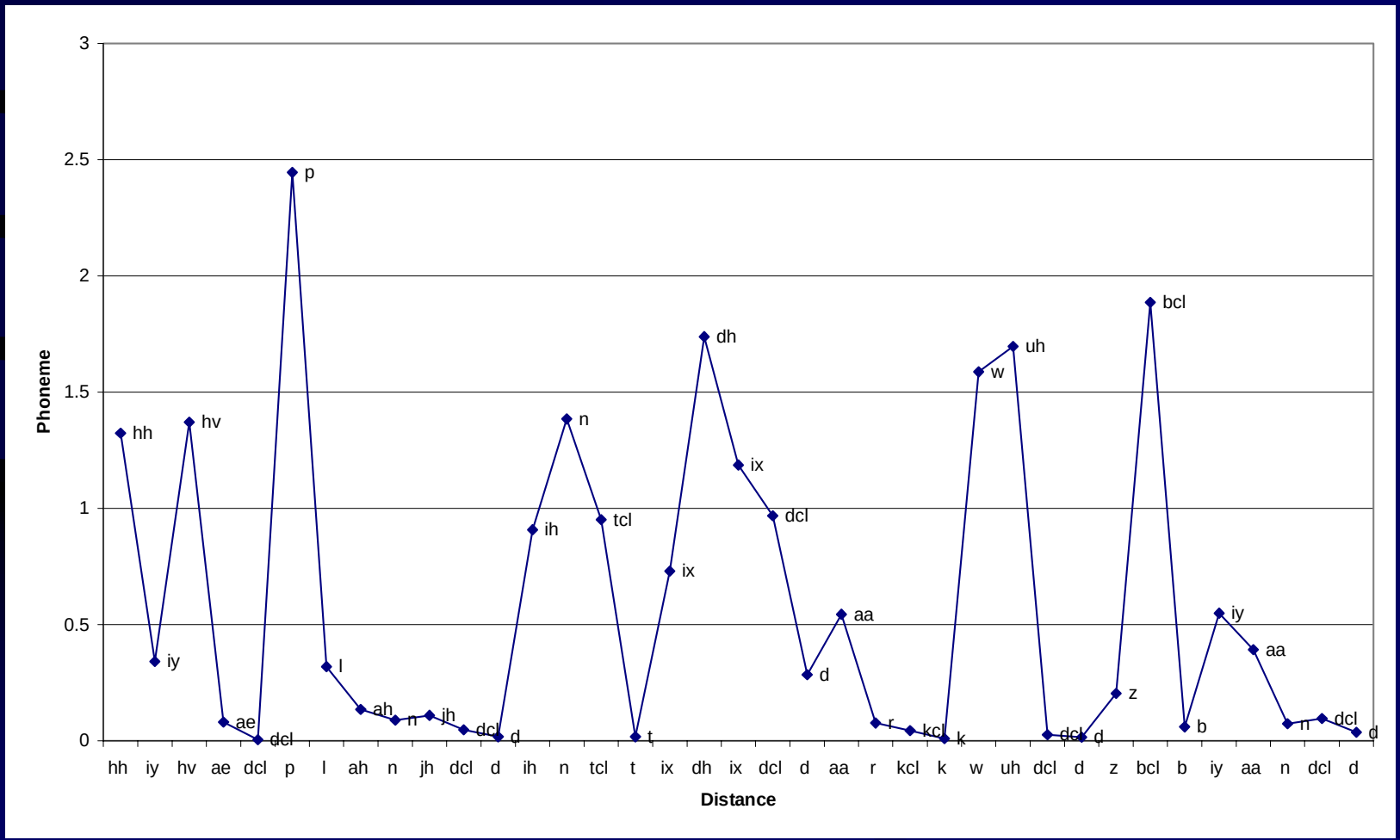


- Network trained to predict next phoneme in coninuous stream
- *Squared error* S_e determined by

$$S_e = \sum_{i=0}^{61} (y_i - z_i)^2$$

Testing PPRNN

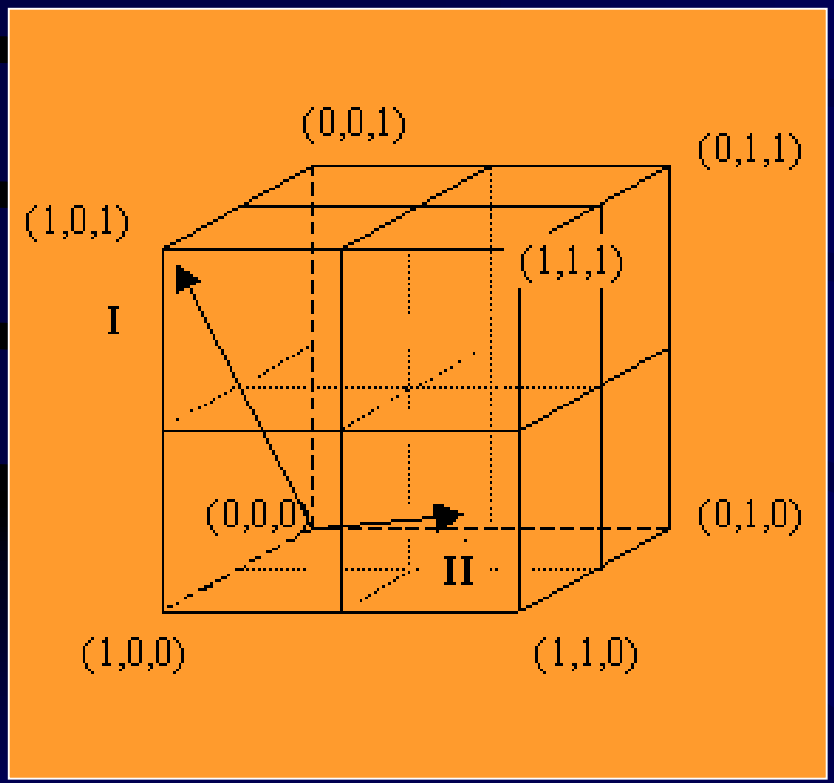
“hh iy - hv ae dcl - p l ah n jh dcl d - ih n tcl t ix - dh ix - dcl d aa r kcl k - w uh dcl d z - bcl b iy aa n dcl d”



Performance PPRNN parser

- Two error types
  - insertion error helloworld → he-llo-world
  - deletion error helloworld → helloworld
- Performance parser complete testset
  - parsing errors : 22.9 %

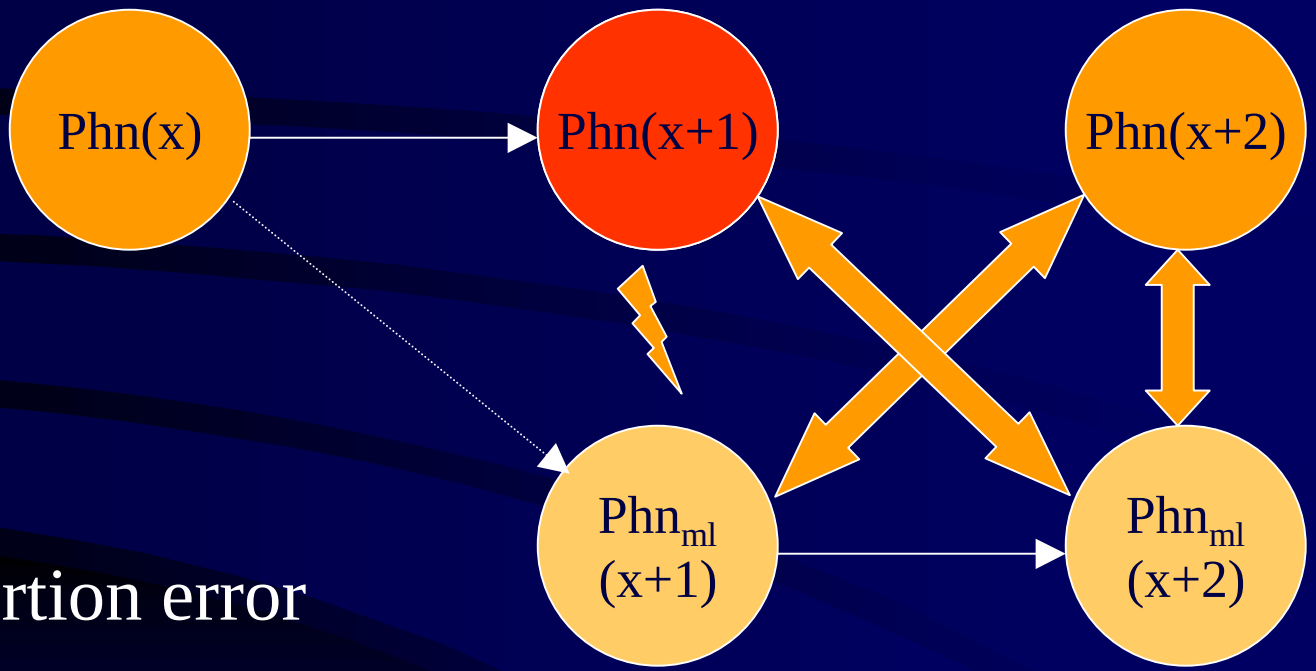
Error detection using PPRNN



- Prediction forms vector in phoneme space
- Squared error to closest cornerpoint *squared error to most likely* $S_{\text{e mostlikely}}$
- Error indicated by

$$E_{\text{indication}} = \frac{S_e}{S_{\text{e mostlikely}}}$$

Error correction using PPRNN



- Insertion error
- Deletion error
- Substitution error

Performance Correction PPRNN

- Error rate reduced 8.6 %

	Recnet HMM	HMM + PPRNN
Correct	72.9%	80.7%
Insertion	4.1%	3.3%
Deletion	20.9%	14.8%
Substitution	6.2%	4.5%
Total errors	31.2%	22.6%

Contents

- Problem definition
- Automatic Speech Recognition (ASR)
- Recnet ASR
- Phoneme postprocessing
- Word postprocessing
- Conclusions and recommendations

Conclusions

Possible to create an ANN ASR

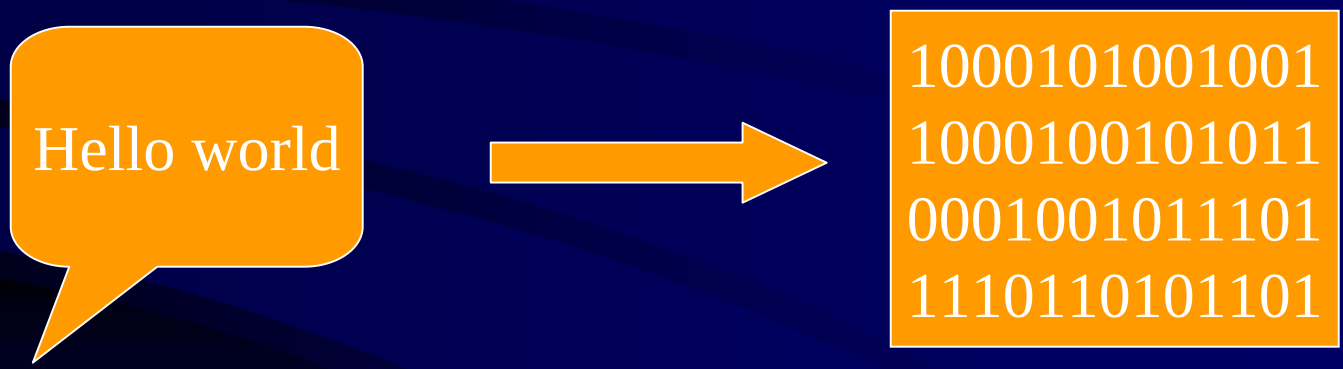
- Recnet
  - documented and trained
- Implementation RecBench
- ANN phoneme postprocessor
  - Promising performance
- ANN word postprocessor
  - Parsing 80 % correct
  - Reducing error rate 9 %

Recommendations

- ANN phoneme postprocessor
  - different mixing techniques
  - Increase framerate
  - More advanced scoring algorithm
- ANN word postprocessor
  - Results of increase in vocabulary
- Phoneme to Word conversion
  - Autoassociative ANN
- Hybrid Time Delay ANN/ RNN

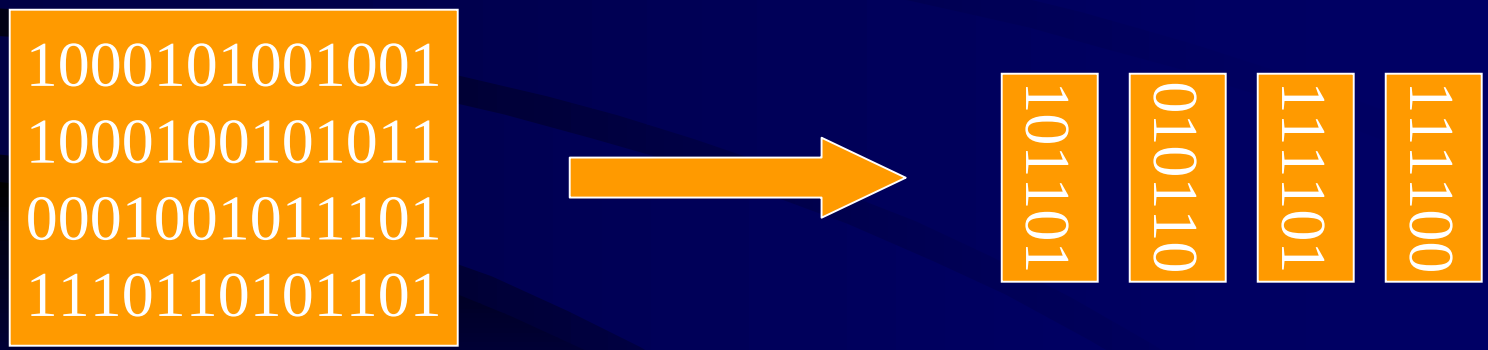
A/D Conversion

- Sampling the signal



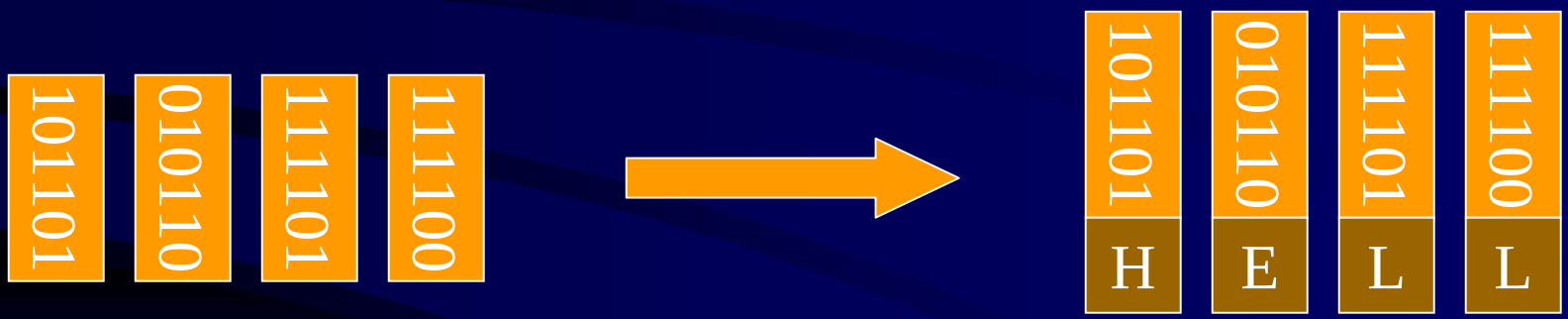
Preprocessing

- Spliting signal into frames
- Extracting features



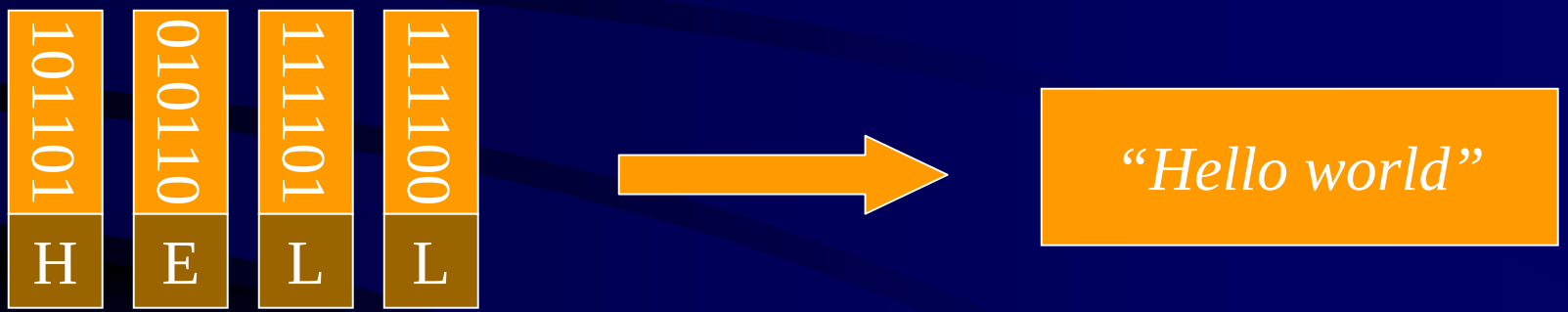
Recognition

- Classification of frames

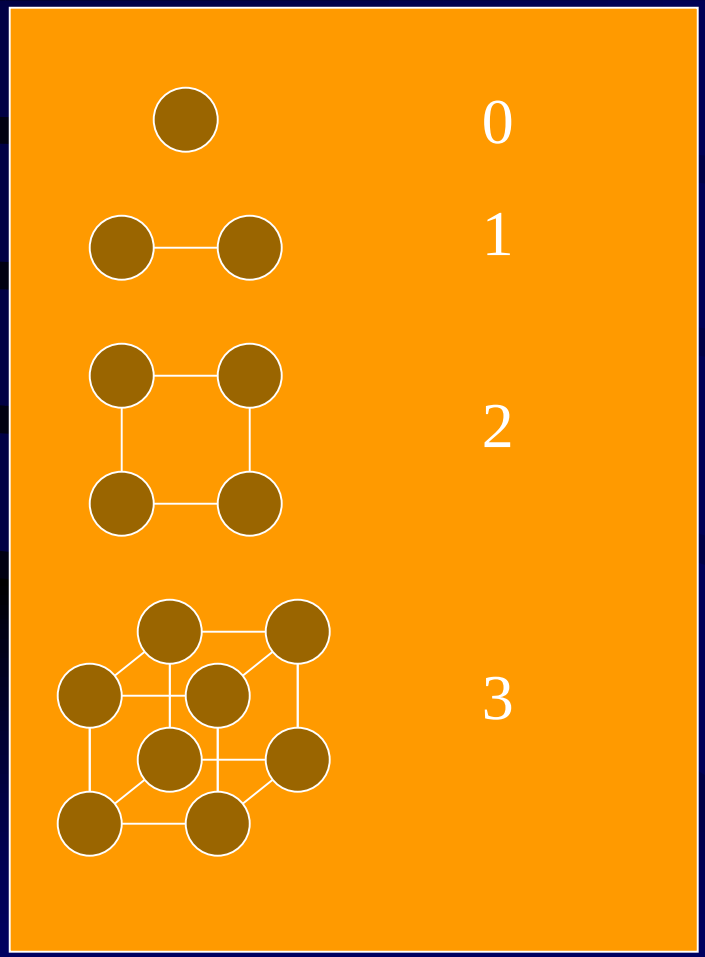


Postprocessing

- Reconstruction of the signal

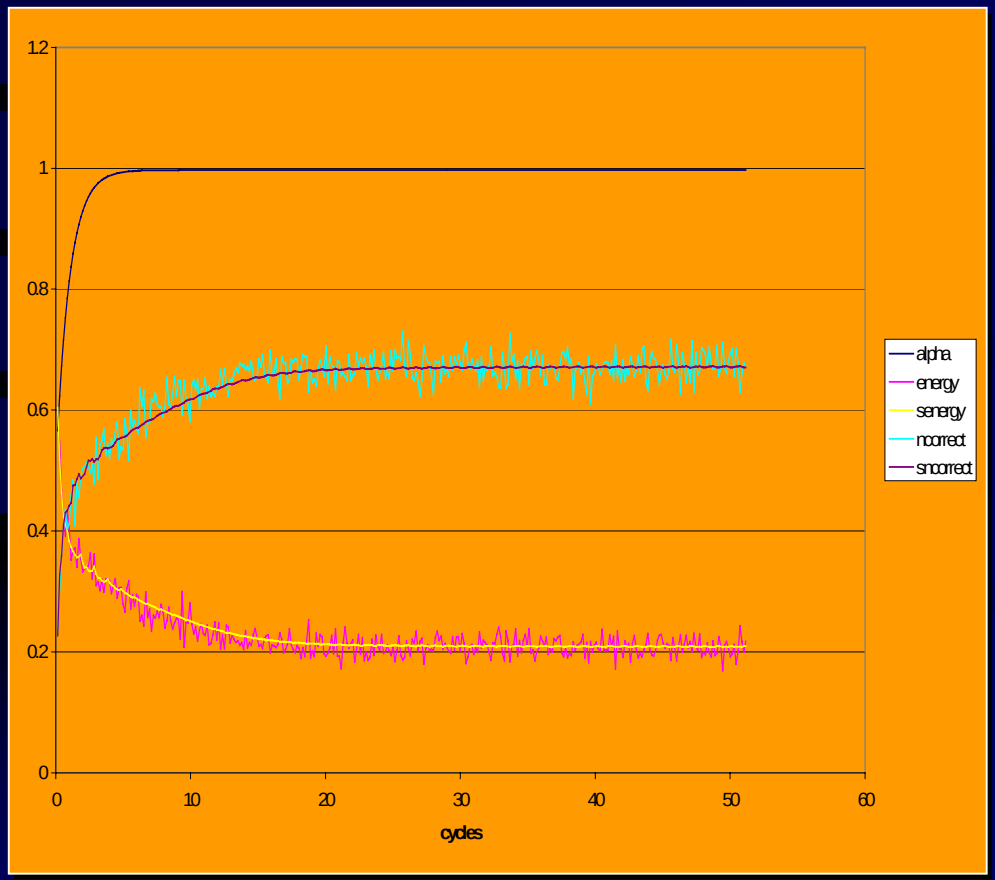


Training Recnet on the nCUBE2



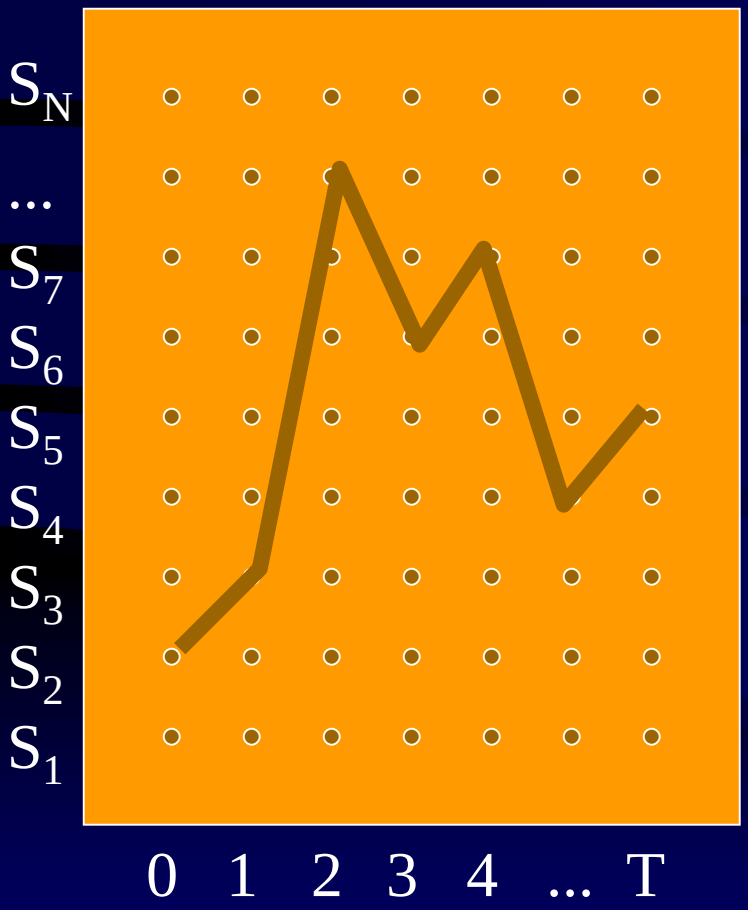
- Training RNN computational intensive
- Trained using Back-propagation Through Time
- nCUBE2 hypercube architecture
- 32 processors used during training

Training results nCUBE2



- Training time on nCUBE2 50 hours
- Performance trained Recnet 65.0 % correct classified phonemes

Viterbi algorithm



- A posteriori most likely path
- Recursive algorithm with *O(t)* is the observation

$$\delta_j(0) = \pi_j$$

$$\delta_j(t) = \max_{1 \le i \le N} (\delta_i(t-1) a_{ij}) b_j(O(t))$$

Elman Character prediction RNN (2)

- Trained on words
  "Many years a boy and girl lived by the sea. They played happily"
- During training words picked randomly from vocabulary
- During parsing *squared error* S_e is determined by

$$S_e = \sum_{i=0}^{61} (y_i - z_i)^2$$

  with y_i is correct output of node i and z_i is the real output of node i
- S_e is expected to decrease within a word

Elman Character prediction RNN (3)