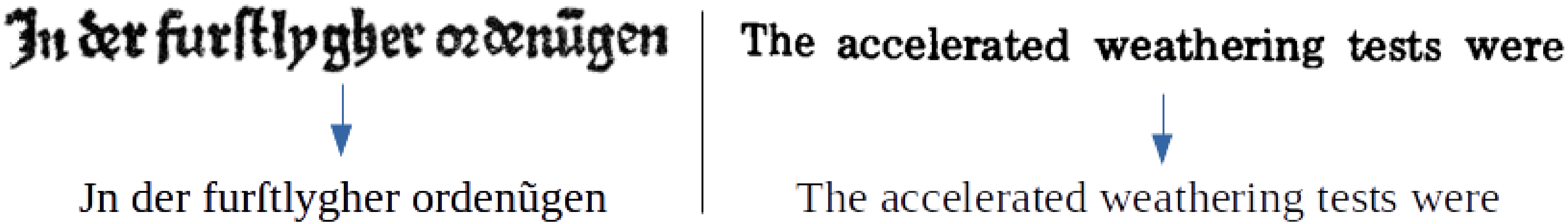
Sequence2Sequence

CTC Algorithm in OCR and speech recognition

Table of contents

Sequence2Sequence

- Example OCR: Image of a line in a text

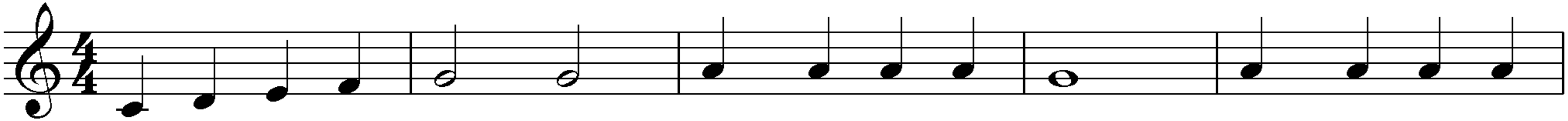


Jn der furſtlygher ordenũgen

The accelerated weathering tests were

Sequence2Sequence

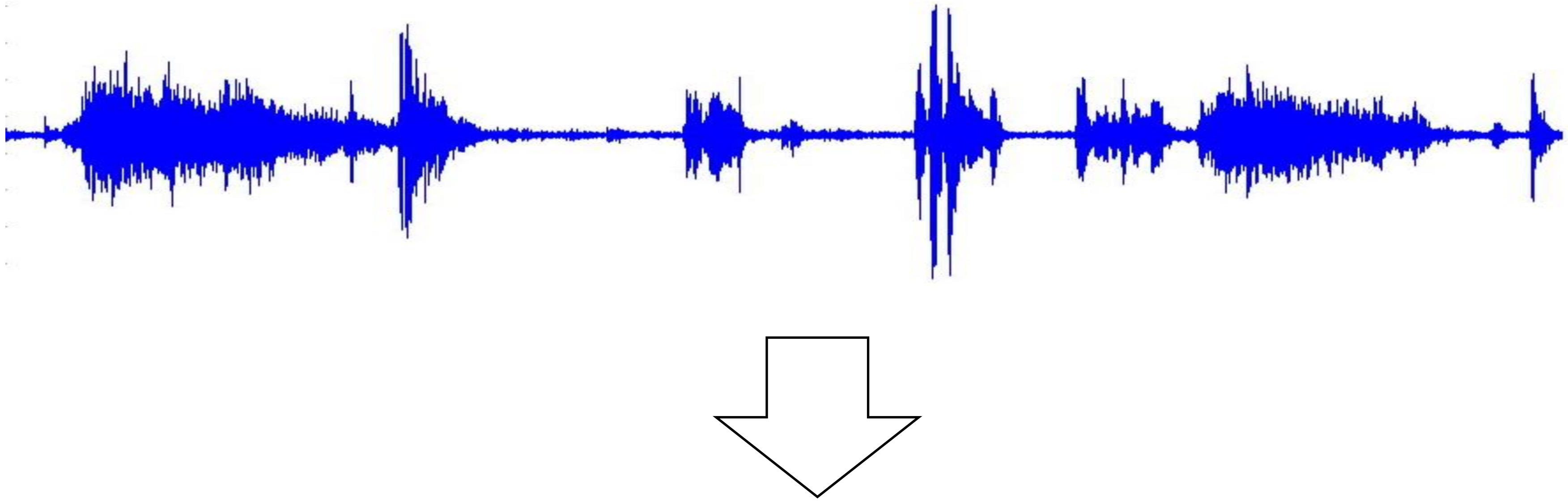
- Example OMR: Image of a line in sheet music



CDEF|G G |AAAA|G |AAAA|

Sequence2Sequence

- Example Speech Recognition: Audio signal in phonemes/letters



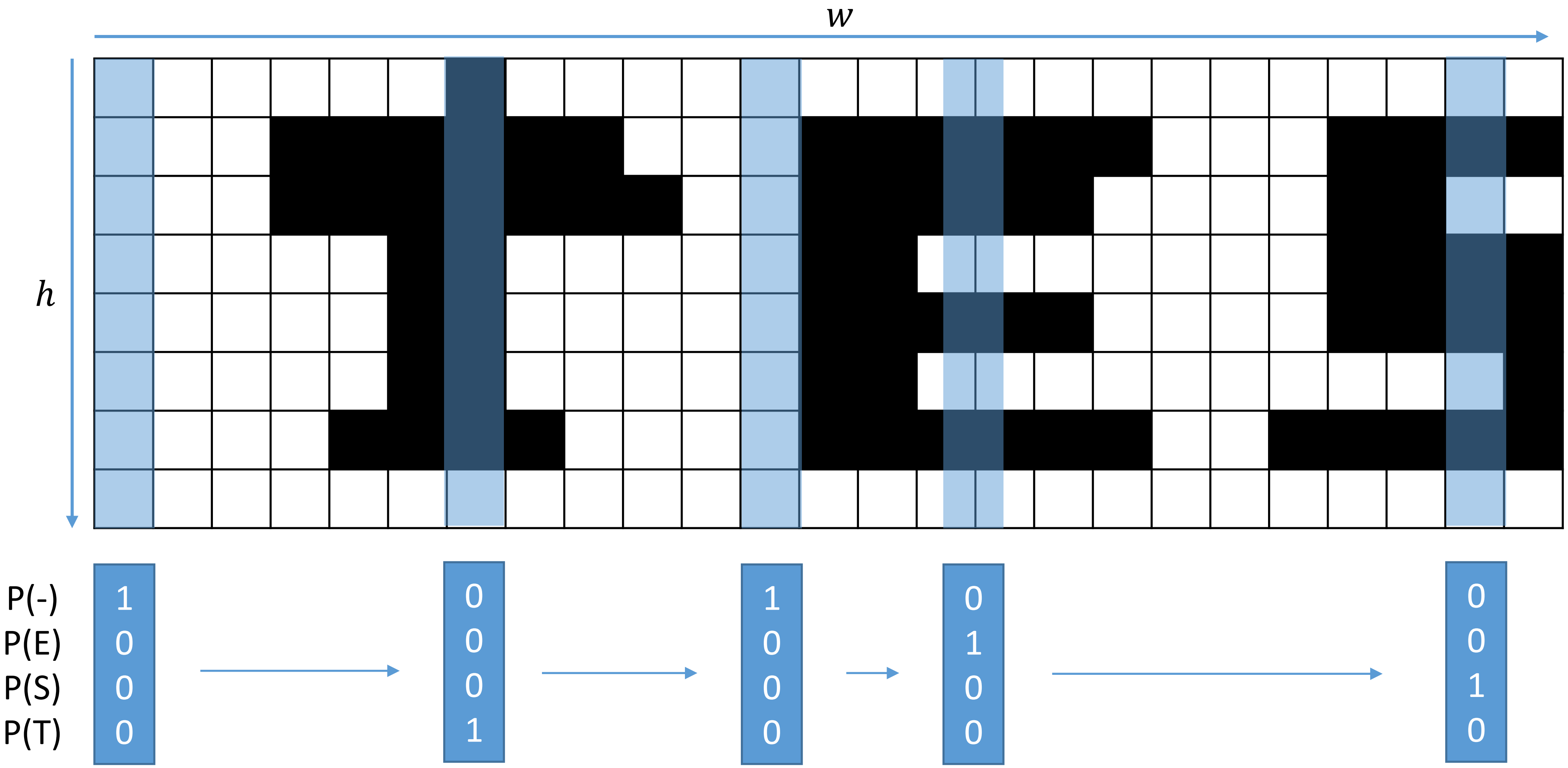
A goose is running around the house

Sequence2Sequence

Problem formulation:

- Input: Sequence of vectors:
  - OCR/OMR: Sequence of pixel columns (over the x-axis/image width)
  - Speech recognition: Sequence of amplitudes/frequencies (over time)
- Output: Sequence of probabilities over classes/labels = alphabet:
  - OCR/OMR: Letters/numbers/notes
  - Speech recognition: phonemes (letters also possible)

Sequence2Sequence: OCR



Dimensions:
$w \times h$, or
$T \times F$ (Time, Features)

Dimensions:
$T \times L$ (Time, Output Size)

Sequence2Sequence: OCR

- Input image (Pixel): Interpret width as „time“ and height as „features“
- Neuronal Network, e.g. LSTM
- Output sequence over probabilities of characters of the alphabat (matrix of „time“ times alphabet size)

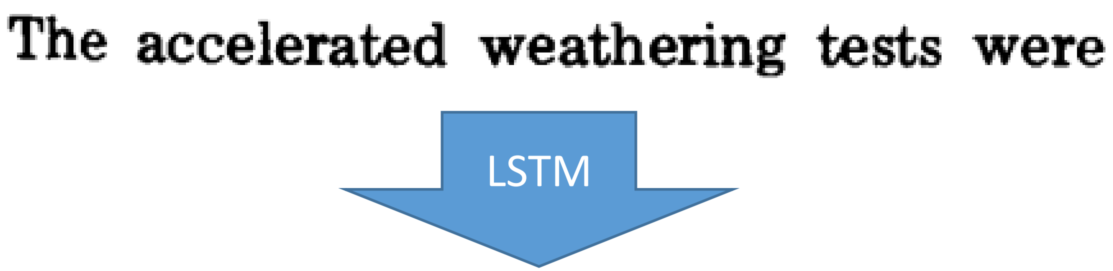
The accelerated weathering tests were

LSTM

T	0	1	2	3
P(-)	1	1	0	…
P(a)	0	0	0	…
P(b)	0	0	0	…
P(c)	0	0	0	…
P(d)	0	0	0	…
P(e)	0	0	0	…
…	…	…	…	…

Sequence2Sequence: OCR

- Alphabet consists of all possible characters (letters/numbers/special characters) and a „blank“ (=nothing) label („-“ on the right)
- The label L is the „internal ID“ or the index within the output matrix



The accelerated weathering tests were

L	T	0	1	2	3
0	P(-)	1	1	0	…
1	P(a)	0	0	0	…
2	P(b)	0	0	0	…
3	P(c)	0	0	0	…
4	P(d)	0	0	0	…
5	P(e)	0	0	0	…
…	…	…	…	…	…

Sequence2Sequence: OCR

Open questions

- How to decode?
    - Determine most probable sequence from output matrix
- How to train?
    - Pairs and format of training data (image/GT)
    - Which loss function?

Main problem:

- Input sequence is longer than output sequence

Decode

Determine/approximate most probable sequence

Sequence2Sequence: OCR

- Example Output

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.7	0.4	0.1	0.6	0

Sequence2Sequence: OCR

- Example Output with argmax

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.7	0.4	0.1	0.6	0
amax	**-**	**a**	**a**	**-**	**f**	**f**	**-**	**f**	**e**

Sequence2Sequence: OCR

- Example Output with argmax:
  1. Argmax: -aa-ff-fe
  2. Remove duplicate characters: -a-f-fe
  3. Remove blanks: affe
- This decoder is also called **Greedy-Decode**
- Alternative: **Beam-Search** (more later)

Sequence2Sequence: OCR

- Example Output

	0	1	2	3	4	5	6	7	8
P(-)	0	0.1	0.4	0.8	0.3	0.9	0.9	0.2	0.01
P(a)	0	0.2	0	0.15	0	0	0	0	0
P(b)	0.4	0.1	0	0	0.2	0	0	0	0
P(c)	0	0.1	0	0	0.2	0	0	0	0
P(d)	0	0.1	0.1	0.05	0	0	0.1	0	0
P(e)	0	0.1	0.1	0	0.1	0	0	0.8	0.99
P(f)	0.6	0.3	0	0	0.2	0.1	0	0	0

Sequence2Sequence: OCR

- Example Output

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
P(-)	0	0.1	0.4	0.8	0.3	0.9	0.9	0.2	0.01
P(a)	0	0.2	0	0.15	0	0	0	0	0
P(b)	0.4	0.1	0	0	0.2	0	0	0	0
P(c)	0	0.1	0	0	0.2	0	0	0	0
P(d)	0	0.1	0.1	0.05	0	0	0.1	0	0
P(e)	0	0.3	0.1	0	0.1	0	0	0.8	0.99
P(f)	0.6	0.1	0	0	0.2	0.1	0	0	0
amax	**f**	**e**	**-**	**-**	**-**	**-**	**-**	**e**	**e**

→ fee

Training

Loss function

Sequence2Sequence: Training, Loss

- Given input $x = (x_1, x_2, \dots, x_T) \in \mathcal{X} = \mathbb{R}^{F \times T}$(image) and
- Alphabet L, Alphabet with blank L'
- Target output $z = (z_1, z_2, \dots, z_U) \in \mathcal{Z} = L^{\leq T}$(Label sequence)
- Output sequence is shorter than input sequence ($U \leq T$)
- Goal: Classification (e.g. with NN): $h: \mathcal{X} \longmapsto \mathcal{Z}$
- Error (Accuracy): Number of false characters: Label Error Rate (LER) on test data S:

$$LER(h, S) = \frac{1}{Z} \sum_{(x,z) \in S} ED(h(x), z)$$

Z is the total number of characters in S, i.e. $Z = \sum_{z \in S} |z|$

ED = Edit-Distanz

Sequence2Sequence: Training, Loss

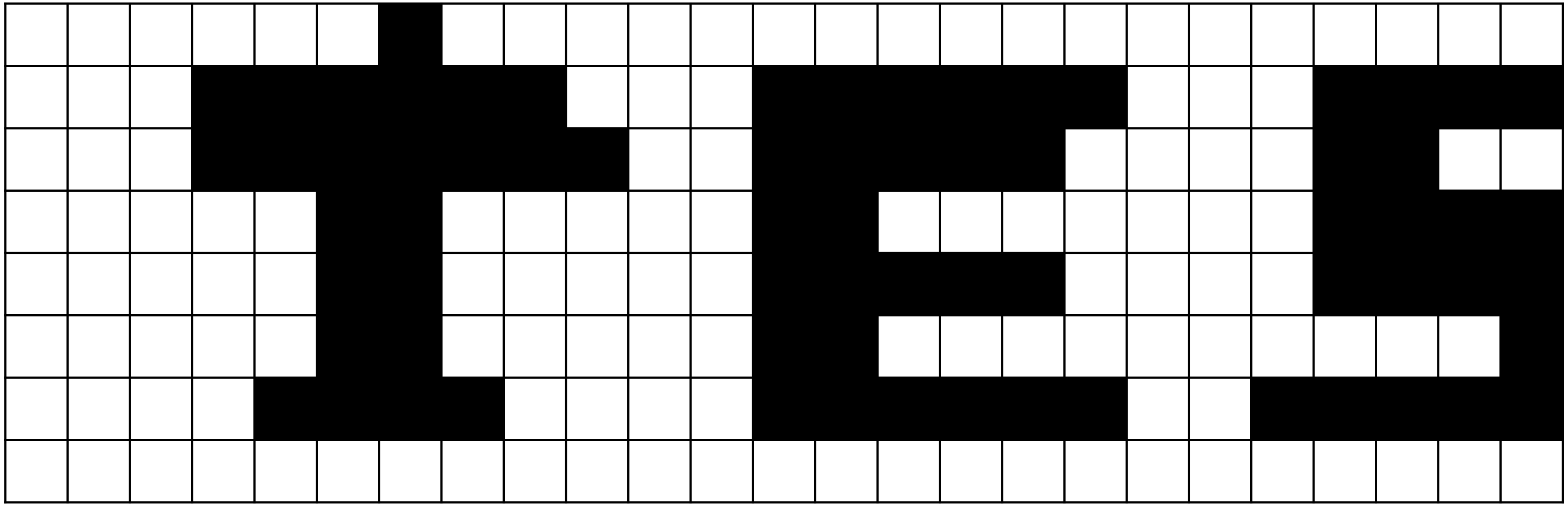
- Given input $x = (x_1, x_2, \dots, x_T) \in \mathcal{X} = \mathbb{R}^{F \times T}$ (image) and
- Alphabet L, Alphabet with blank L'
- Target output $z = (z_1, z_2, \dots, z_U) \in \mathcal{Z} = L^{\leq T}$ (Label sequence)
- Output sequence is shorter than input sequence ($U \leq T$)

First look at

- Easy case: $T = U$

Sequence2Sequence: Training, Loss

- I.e. we know the labels at every target position



-	-	-	T	T	T	T	T	T	T	-	-	E	E	E	E	E	E	-	-	S	S	S	S	S

Sequence2Sequence: Training, Loss

- NN output: probability y_k^t at time t, of label k
- Probability of a sequence π (of labels)

$$P(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t , \qquad \forall \pi \in \mathcal{Z}$$

- Decoding (highest probability of all possible sequences)

$$\underset{\pi}{\operatorname{argmax}}\, P(\pi|x)$$

Sequence2Sequence: Training, Loss

- Loss (Negative-Log-Likelihood)

$$L = -\log P(z|x) = -\log \prod_{t=1}^{T} y_{\pi_t}^{t} = -\sum_{t=1}^{T} \log y_{\pi_t}^{t}$$

  - Corresponds to maximizing total probability
  - Analogous to simple classification
(however in that case without the sum over t)

Sequence2Sequence: Training, Loss

- Creating a GT which assigns a label to every time step is very costly
- Better/in general: Only the true (short, normalized) sequence is known
- The Neural Network learns assigning the labels itself!
- I.e. complex case: $U \leq T$
- New loss function: CTC loss, which considers this case

Training

CTC Loss Function

Labellings $\mathcal{B}$

- Mapping $\mathcal{B}: L'^{T} \mapsto L^{\leq T}$
  - Transformation to reduced representation, e.g.
  - $\mathcal{B}(a - ab -) = B(-aa - -aabb) = aab$

- Inverse function $\mathcal{B}^{-1}: L^{\leq T} \mapsto \{L'^{T}\}$
  - Compute all paths π (of length T), that produce input l
  - $\mathcal{B}^{-1}(aab) = \{a - ab-, \dots, aa - ab, \dots, \dots, \dots\}$

- We already used this function during decoding!

Labellings $\mathcal{B}$: Examples/Exercise

- Sei $T = 5$
- Gilt:

$\mathcal{B}(a - ab -) = ab$	No → aab
$\mathcal{B}(babba) = baba$	Yes
$\mathcal{B}(-cat -) = -cat -$	No → cat, no blanks allowed in output
$\mathcal{B}(- - - - -) =$	Yes
$-abbb \in \mathcal{B}^{-1}(ab)$	Yes
$ab - b \in \mathcal{B}^{-1}(abb)$	No, sequence only has length 4, 5 required
$-a - bb \in \mathcal{B}^{-1}(abb)$	No, since $\mathcal{B}(-a - bb) = ab$
$-d - b - \in \mathcal{B}^{-1}(ab)$	No, since $\mathcal{B}(-d - b -) = db \neq ab$

Probability of a sequence l

- Mapping $\mathcal{B}: L'^T \mapsto L^{\leq T}$
- Inverse function $\mathcal{B}^{-1}$ returns all paths (length T) which fit l.

- Probability of a sequence $l \in L^{\leq T}$:

$$P(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} P(\pi|x)$$

  - Sum over all paths π, which l can stem from
  - Corresponds to total probability, since it is a sum over single probabilities

Probability of a path $\pi \in \mathcal{B}^{-1}(l)$

- Probability of a path π is still the product of network outputs
- Each path has length T (one output value for each time step)

$$P(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^{t}$$

Example

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

- $l = \text{affe}, \mathcal{B}^{-1}(l) = \{----\text{af}-\text{fe}, ---\text{aaf}-\text{fe}, \ldots, \text{aaaff}-\text{f}-\text{e}, \ldots, \text{af}-\text{feeeee}\}$
- $P(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} P(\pi|x)$:
  - $P(----\text{af}-\text{fe}) = 0.9 \cdot 0.4 \cdot 0.1 \cdot 0.8 \cdot 0.1 \cdot 0.4 \cdot 0.9 \cdot 0.6 \cdot 0.99$
  - $P(\text{af}-\text{feeeee})=?$

Example

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

- $l = \text{ade}, \mathcal{B}^{-1}(l) =?$

Summary

- Input $x = (x_1, x_2, \dots, x_T) \in \mathcal{X} = \mathbb{R}^{F \times T}$ (Image)
- Alphabet L, Alphabet with blank L'
- Target output $z = (z_1, z_2, \dots, z_U) \in \mathcal{Z} = L^{\leq T}$(Label sequence)
- NN output: Probability y_k^t at time t, of label k
- Mapping $\mathcal{B}: L'^T \mapsto L^{\leq T}$ (Transformation to reduced output)
- Probability of a sequence $l \in L^{\leq T}$: $P(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} P(\pi|x)$
- Probability of a path $\pi \in L'^T$: $P(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t$

Agenda: Training and Loss function

- **Formulate** loss function :
  - What should be optimized?
  - How is it calculated?
- **Simplify** loss function :
  - Very costly „single computations“
  - Dynamic programming
- **Derive** loss function :
  - Returns the necessary backpropagation update

What should be optimized?

- Given is a Ground Truth pair (x, z) consisting of input x (image) and the target sequence $z \in L^{\leq T}$
  - E.g. ( The accelerated weathering The accelerated weathering) with
  - Dim. $T \times F = 200 \times 40, \;\; 27$
- The neural network (e.g. LSTM) computing the probability y_k^t at time step $t \in 0, \dots, T$ for the label class $k \in L = \{-abcde \dots ABCDE \dots 01234 \dots\}$
- Optimization goal:

$$\text{Maximize } P(z|x) \rightarrow 1\text{, or NLL:}$$

$$\text{Minimize: } \boxed{-\log P(z|x) \rightarrow 0}$$

How is it calculated?

- Dataset $D = \{(x_0, z_0), (x_1, z_1), (x_N, z_N)\}$ with several examples: Not just one, but all examples need to be classified correctly
- Loss/Optimizer (here Gradient Descent)

$$L(D) = \sum_{n=0}^{N} \ell(x_n, z_n)\,, \qquad \ell(x, z) = -\log P(z|x)$$

Probability of a sequence is defined by the sum of all its paths

$$P(z|x) = \sum_{\pi \in \mathcal{B}^{-1}(z)} P(\pi|x)\,, P(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^{t}$$

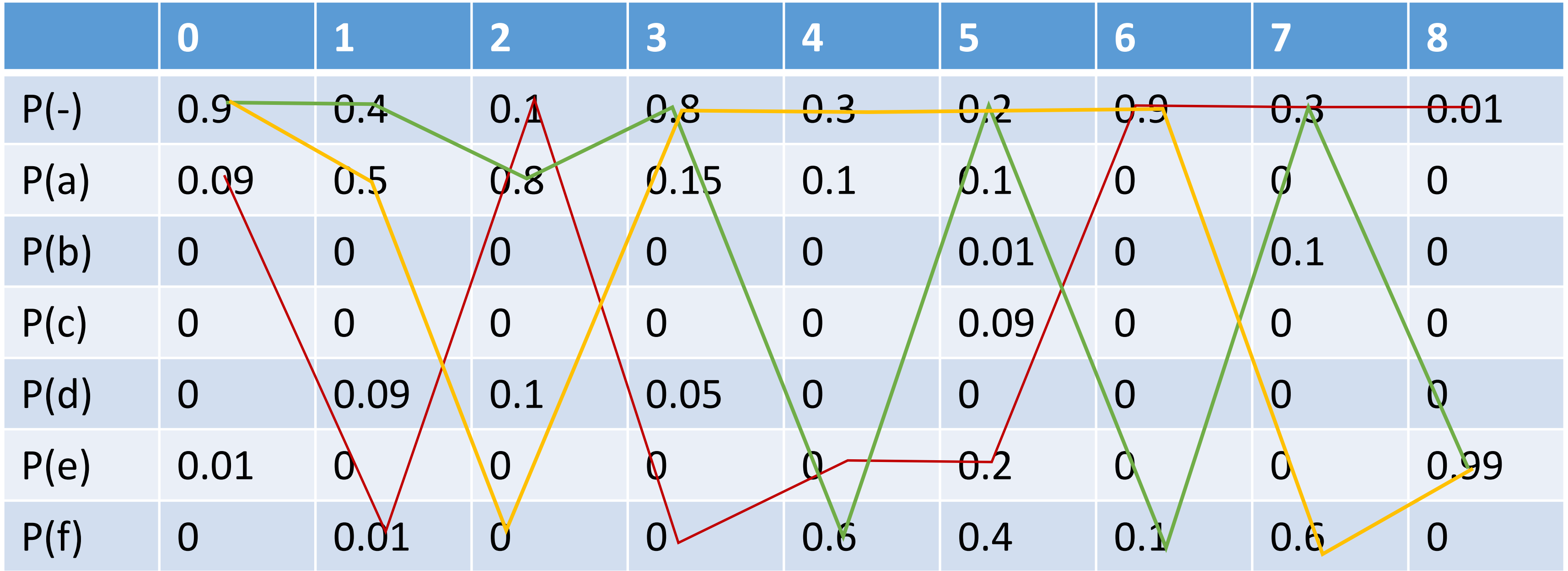
$$\Rightarrow L(D) = -\sum_{n=0}^{N} \log \sum_{\pi \in \mathcal{B}^{-1}(z)} \prod_{t=1}^{T} y_{\pi_t}^{t}$$

Very costly „single computations“

$$P(z|x) = \underbrace{\sum_{\pi \in \mathcal{B}^{-1}(z)}}_{} P(\pi|x)$$

This sum is extremely huge
(since it has ALL possible paths)

Familiar example for $l = affe$



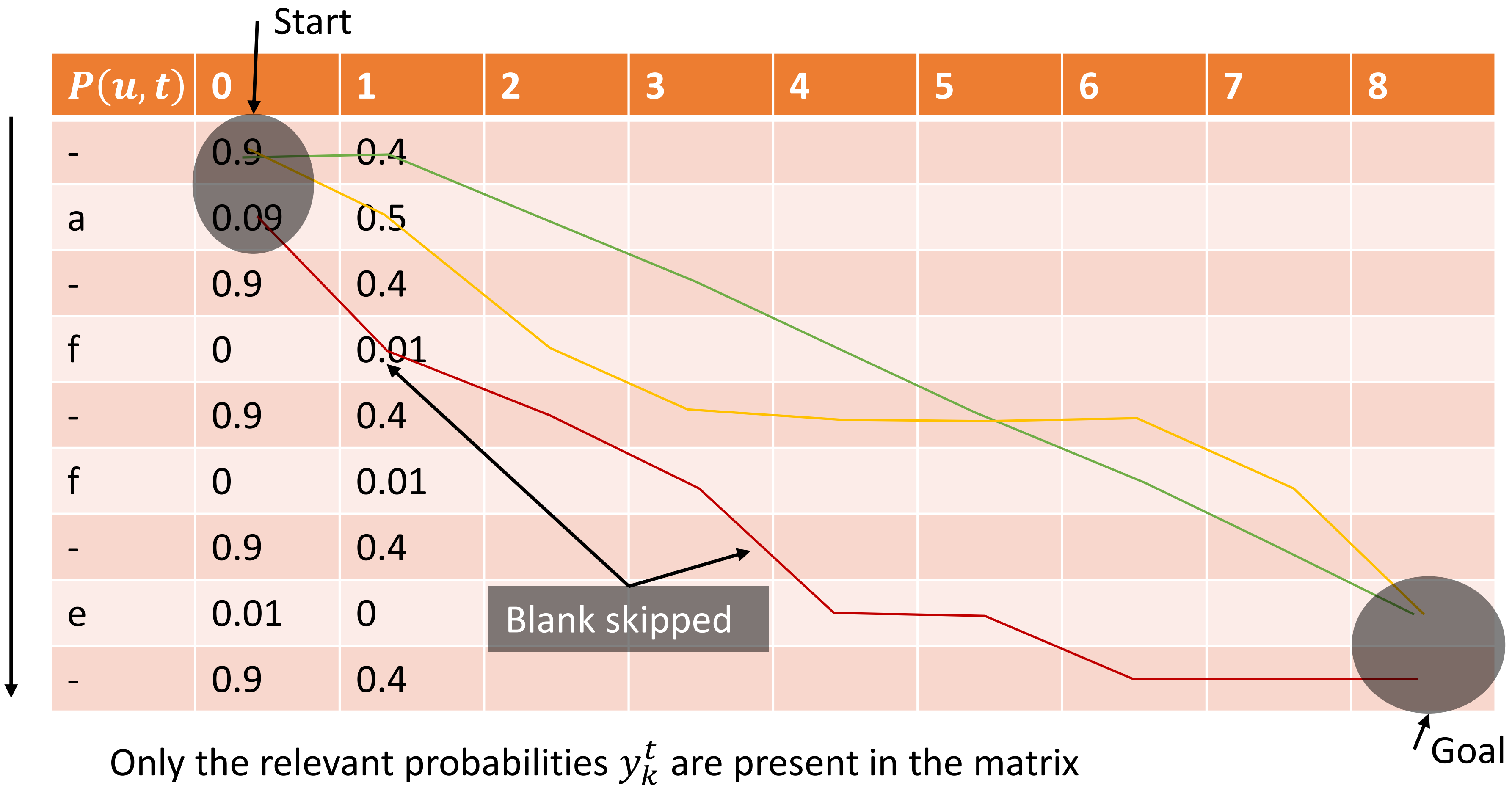
	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

Look at all paths and
sum the probabilities…

Alternative path representation

for $l = affe$:

- Blank beginning/end, between every character
- Paths can only go to the right or down
- No letters can be skipped, but blanks can
- Where are the paths now?



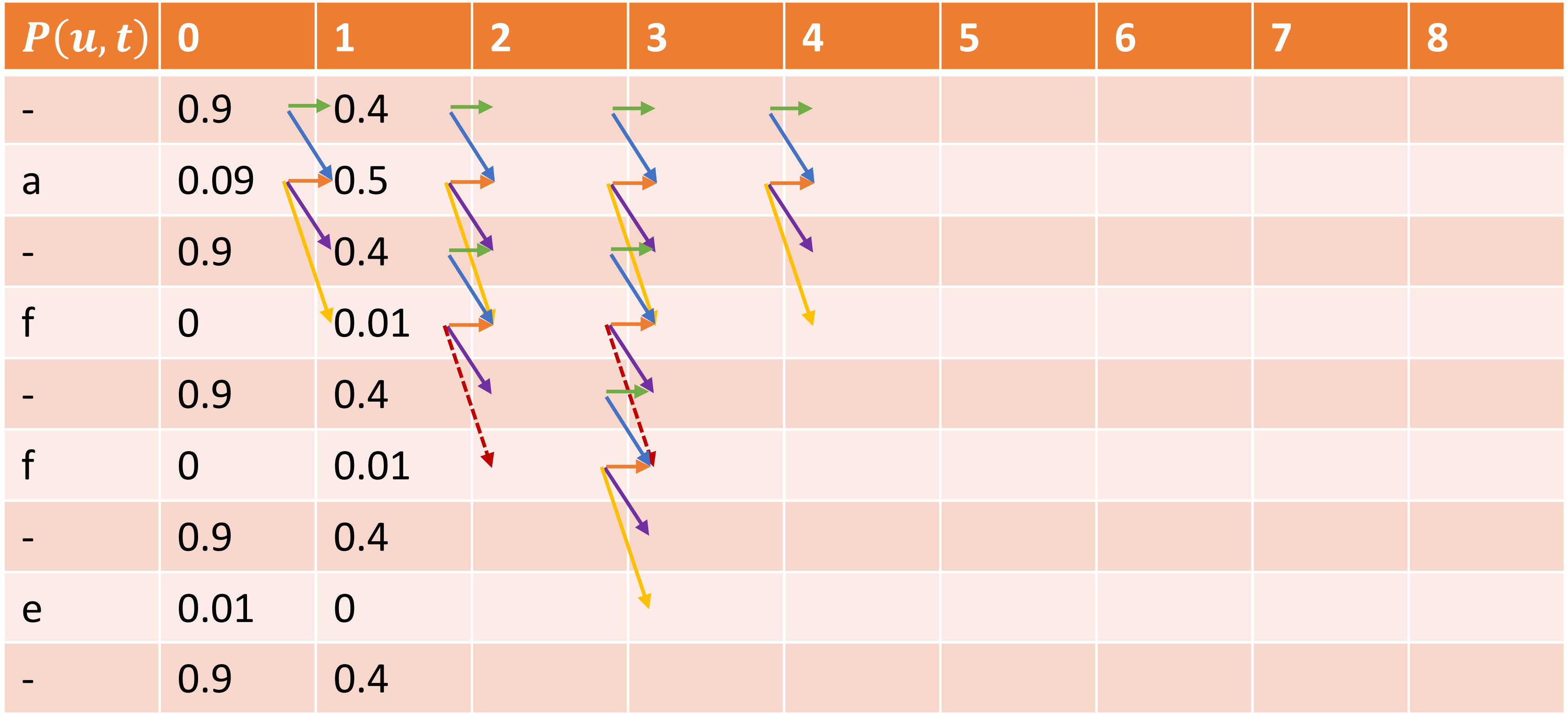
$P(u,t)$	0	1	2	3	4	5	6	7	8
-	0.9	0.4							
a	0.09	0.5							
-	0.9	0.4							
f	0	0.01							
-	0.9	0.4							
f	0	0.01							
-	0.9	0.4							
e	0.01	0							
-	0.9	0.4							

Only the relevant probabilities y_k^t are present in the matrix

Alternative path representation

Allowed transitions:

- Blank → Blank
- Blank → next letter
- Letter → same letter
- Letter → next blank
- Letter → next letter (if not identical)



$P(u,t)$	0	1	2	3	4	5	6	7	8
-	0.9	0.4							
a	0.09	0.5							
-	0.9	0.4							
f	0	0.01							
-	0.9	0.4							
f	0	0.01							
-	0.9	0.4							
e	0.01	0							
-	0.9	0.4							

No path can go here (otherwise letters will be skipped)

Sub path probabilities

- $\alpha(0,0) = y^0_-$
- $\alpha(1,0) = y^0_a$
- $\alpha(0,1) = \alpha(0,0) \cdot y^1_-$
- $\alpha(1,1) = \alpha(0,0) \cdot y^1_a + \alpha(1,0) \cdot y^1_a$

	0	1
P(-)	0.9	0.4
P(a)	0.09	0.5
P(b)	0	0
P(c)	0	0
P(d)	0	0.09
P(e)	0.01	0
P(f)	0	0.01

t →

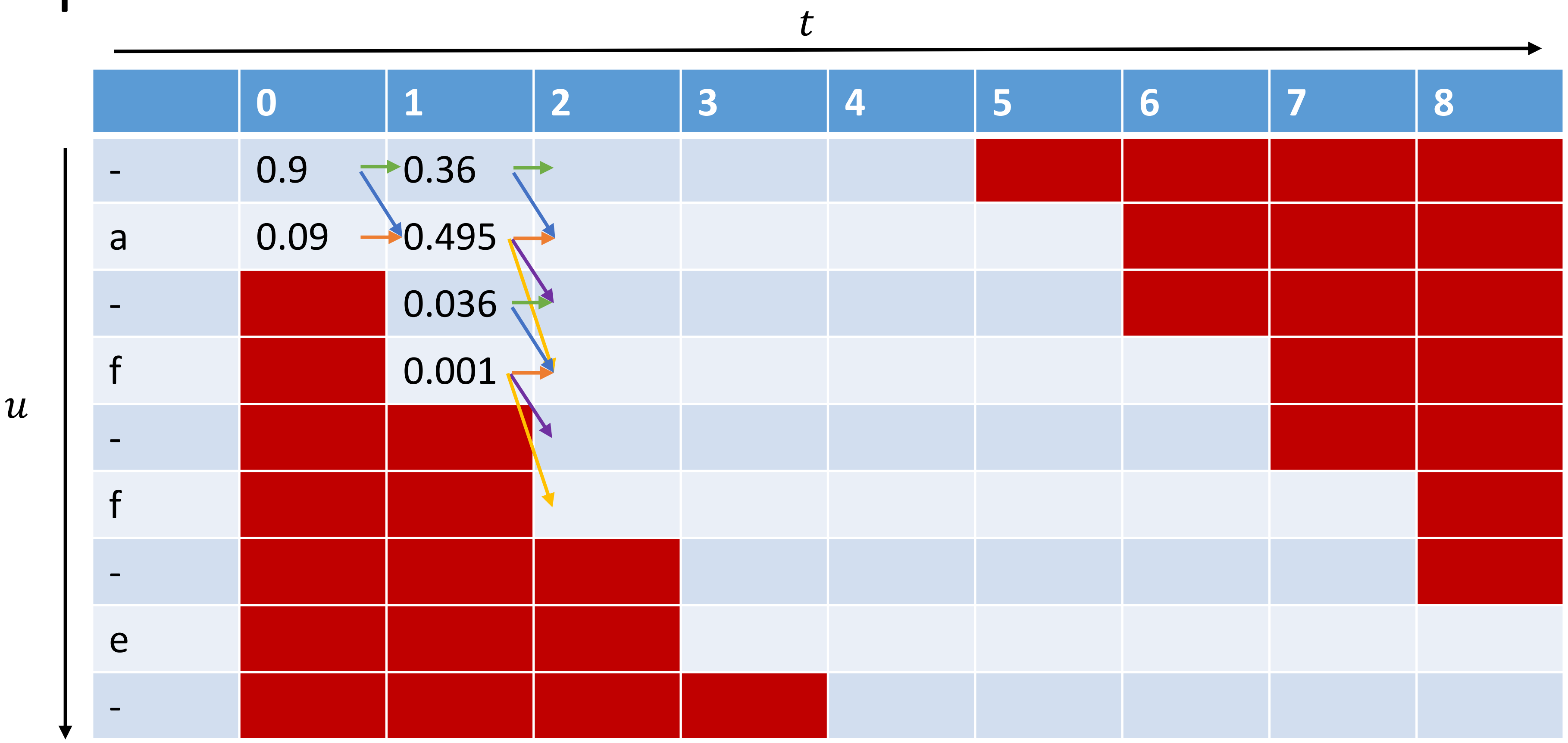
u ↓

$\alpha(u,t)$	0	1	2	3	4	5	6	7	8
-	0.9	0.36				🟥	🟥	🟥	🟥
a	0.09	0.495					🟥	🟥	🟥
-	🟥	0.036					🟥	🟥	🟥
f	🟥	0.001						🟥	🟥
-	🟥	🟥						🟥	🟥
f	🟥	🟥							🟥
-	🟥	🟥	🟥						🟥
e	🟥	🟥	🟥						
-	🟥	🟥	🟥	🟥					

Now: Every row shows the probability $\boldsymbol{\alpha(u,t)}$ an, that any path passes through the cell from the left

Sub path probabilities

	1	2
P(-)	0.4	0.1
P(a)	0.5	0.8
P(b)	0	0
P(c)	0	0
P(d)	0.09	0.1
P(e)	0	0
P(f)	0.01	0

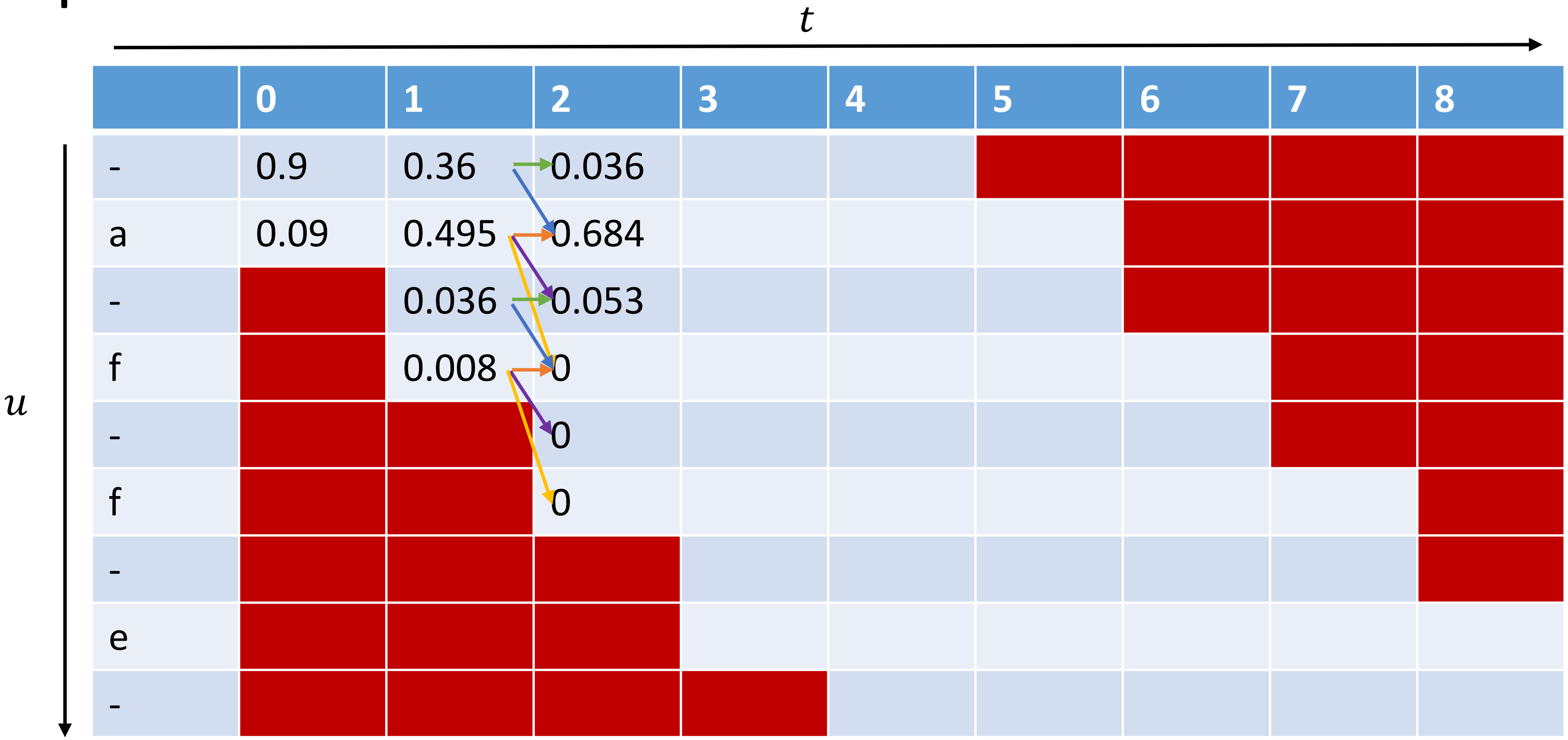


	0	1	2	3	4	5	6	7	8
-	0.9	0.36				🟥	🟥	🟥	🟥
a	0.09	0.495					🟥	🟥	🟥
-	🟥	0.036					🟥	🟥	🟥
f	🟥	0.001						🟥	🟥
-	🟥	🟥						🟥	🟥
f	🟥	🟥							🟥
-	🟥	🟥	🟥						🟥
e	🟥	🟥	🟥						
-	🟥	🟥	🟥	🟥					

Now: Every row shows the probability $\boldsymbol{\alpha(u,t)}$ an, that any path passes through the cell from the left

Sub path probabilities

	1	2
P(-)	0.4	0.1
P(a)	0.5	0.8
P(b)	0	0
P(c)	0	0
P(d)	0.09	0.1
P(e)	0	0
P(f)	0.01	0

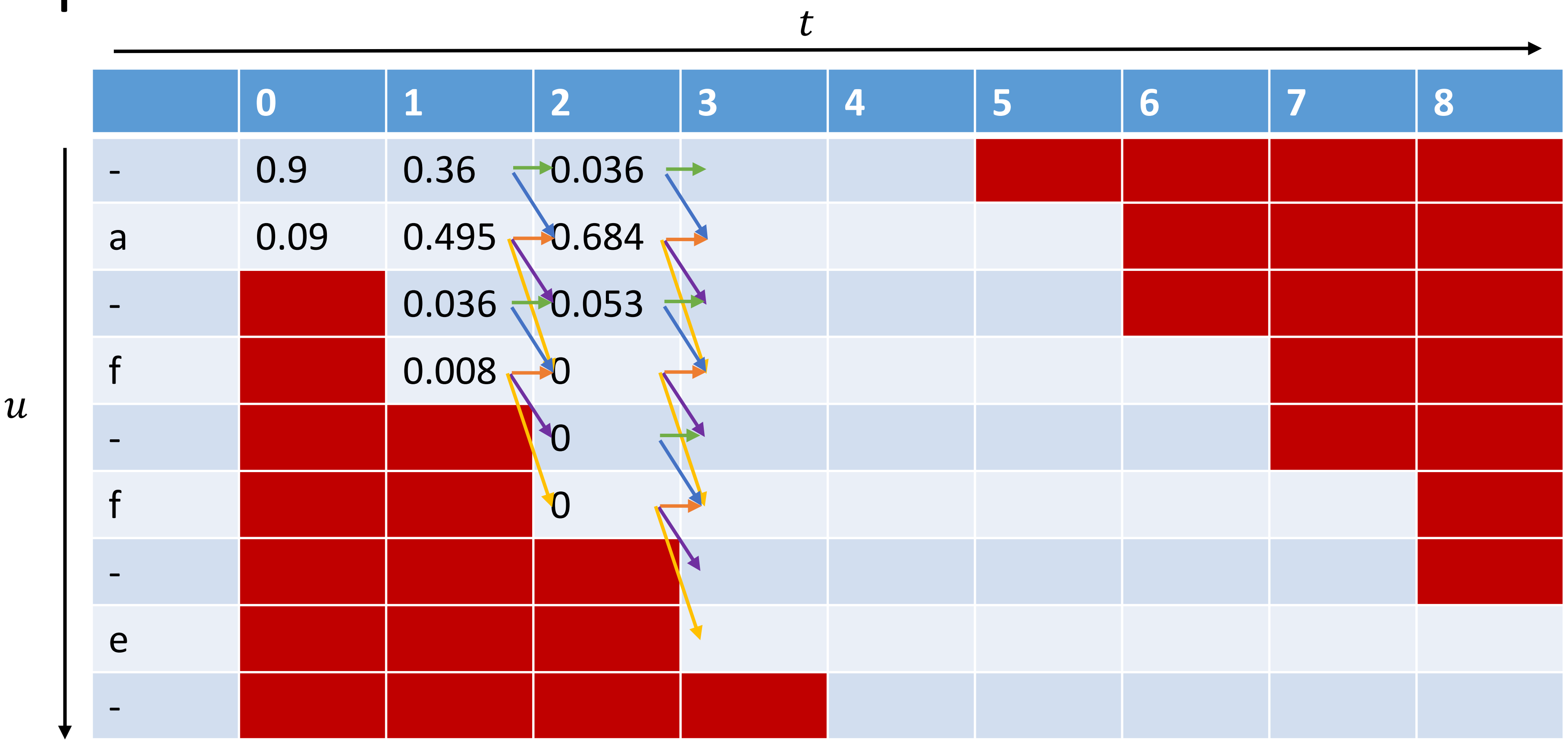


	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036			🟥	🟥	🟥	🟥
a	0.09	0.495	0.684				🟥	🟥	🟥
-	🟥	0.036	0.053				🟥	🟥	🟥
f	🟥	0.008	0					🟥	🟥
-	🟥	🟥	0					🟥	🟥
f	🟥	🟥	0						🟥
-	🟥	🟥	🟥						🟥
e	🟥	🟥	🟥						
-	🟥	🟥	🟥	🟥					

Now: Every row shows the probability $\boldsymbol{\alpha}(\boldsymbol{u}, \boldsymbol{t})$ an, that any path passes through the cell from the left

Sub path probabilities

	2	3
P(-)	0.1	0.8
P(a)	0.8	0.15
P(b)	0	0
P(c)	0	0
P(d)	0.1	0.05
P(e)	0	0
P(f)	0	0



	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036			🟥	🟥	🟥	🟥
a	0.09	0.495	0.684				🟥	🟥	🟥
-	🟥	0.036	0.053				🟥	🟥	🟥
f	🟥	0.008	0					🟥	🟥
-	🟥	🟥	0					🟥	🟥
f	🟥	🟥	0						🟥
-	🟥	🟥	🟥						🟥
e	🟥	🟥	🟥						
-	🟥	🟥	🟥	🟥					

Now: Every row shows the probability $\boldsymbol{\alpha(u,t)}$ an, that any path passes through the cell from the left

Sub path probabilities

	7	8
P(-)	0.3	0.01
P(a)	0	0
P(b)	0.1	0
P(c)	0	0
P(d)	0	0
P(e)	0	0.99
P(f)	0.6	0

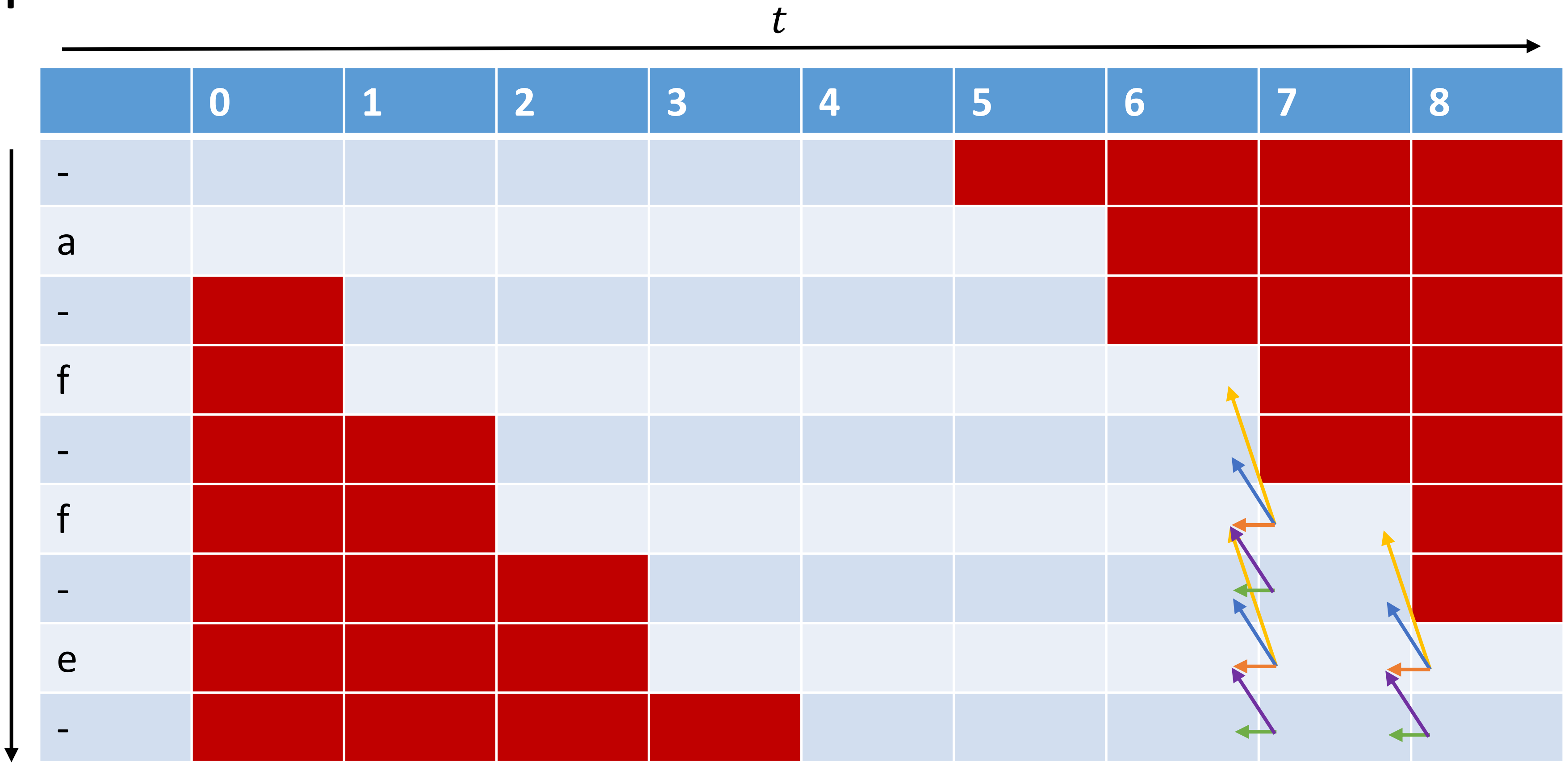
t

u

	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

Now: Every row shows the probability $\boldsymbol{\alpha}(\boldsymbol{u},\boldsymbol{t})$ an, that any path passes through the cell from the left

Sub path probabilities: Backwards!



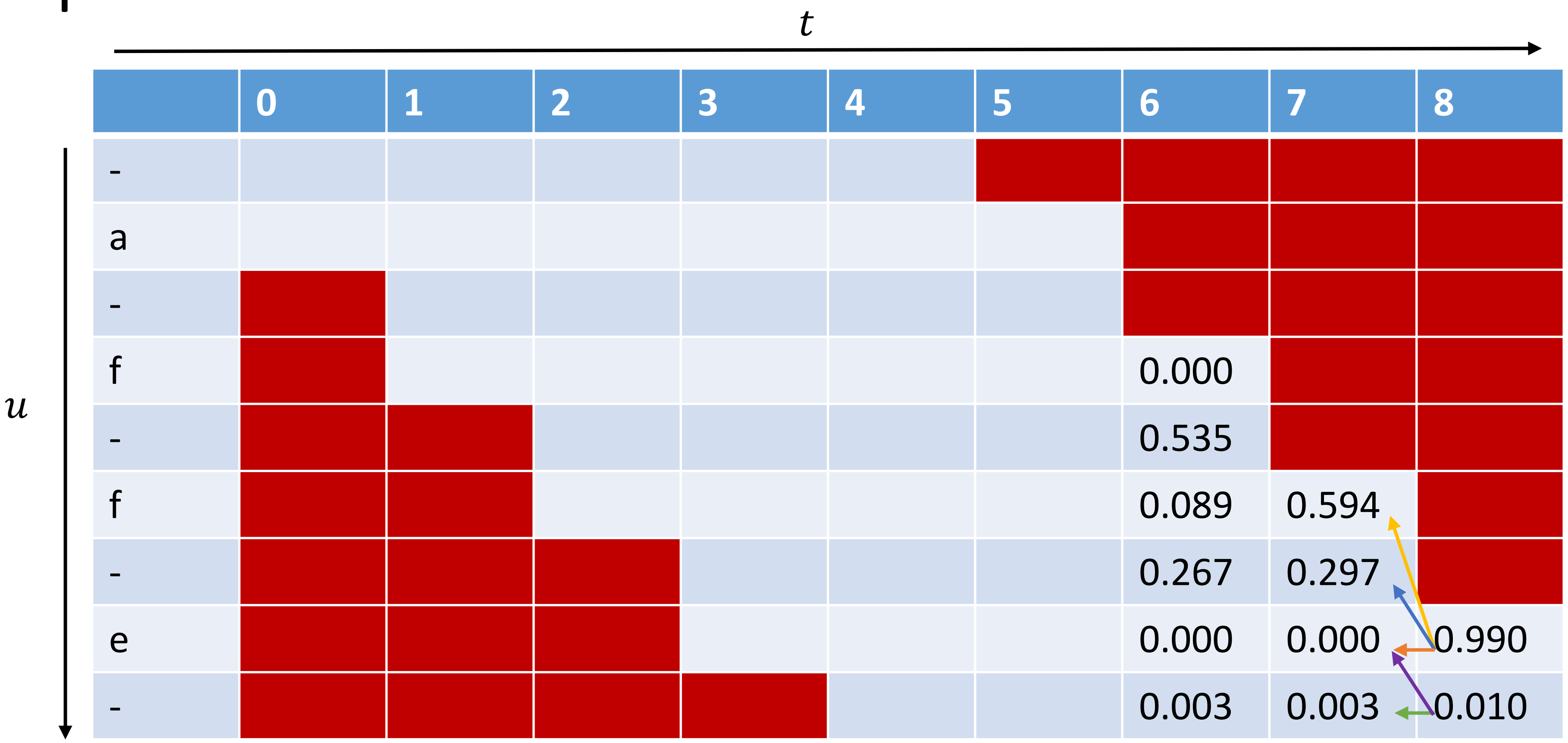
	0	1	2	3	4	5	6	7	8
-						🟥	🟥	🟥	🟥
a							🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥							🟥	🟥
-	🟥	🟥						🟥	🟥
f	🟥	🟥							🟥
-	🟥	🟥	🟥						🟥
e	🟥	🟥	🟥						
-	🟥	🟥	🟥	🟥					

Erlaubte Übergänge:
- Blank → Blank
- Blank → nächster Buchstabe
- Buchstabe → gleicher Buchstabe
- Buchstabe → nächstes Blank
- Buchstabe → nächster Buchstabe (wenn nicht identisch)

Now: Every row shows the probability $\boldsymbol{\beta}(\boldsymbol{u}, \boldsymbol{t})$ an, that any path passes through the cell from the right

Sub path probabilities: Backwards!

	6	7
P(-)	0.9	0.3
P(a)	0	0
P(b)	0	0.1
P(c)	0	0
P(d)	0	0
P(e)	0	0
P(f)	0.1	0.6



	0	1	2	3	4	5	6	7	8
-						🟥	🟥	🟥	🟥
a							🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.003	0.003	0.010

Now: Every row shows the probability $\boldsymbol{\beta(u,t)}$ an, that any path passes through the cell from the right

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.003	0.003	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

- Total probability of „affe“: $\alpha(7,8) + \alpha(8,8)$, but also $\beta(0,0) + \beta(1,0)$
- Result here: 24.3%

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.003	0.003	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

More general: The sum of probabilities of all paths is

> The probability that a path will pass from the left times the prob. that it will pass from the right

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	🟩 0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	🟩 0

β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.003	0.003	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

$$P(l|x) = \sum_{u} \frac{\alpha(u,t)\beta(u,t)}{y^{t}_{l(u)}}, \qquad \forall t$$

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.003	0.003	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

At $t = 6$:

$$P(affe|x) = \frac{0.030 \cdot 0.000}{0.1} + \frac{0.0383 \cdot 0.535}{0.9}$$

$$+ \frac{0.017 \cdot 0.089}{0.1} + 0 + 0 + 0 = \underline{0.243}$$

Summary

- Computing the total probability of a sequence l is defined as:

$$P(l|x) = \sum_{\pi \in \mathcal{B}^{-1}(l)} P(\pi|x)$$

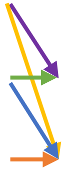
- With the forward and backward variables α and β this can be reshaped to

$$P(l|x) = \sum_{u} \frac{\alpha(u,t)\beta(u,t)}{y^{t}_{l'_u}}, \qquad \forall t,$$

where u follows $l' = (-, l_0, -, l_1, -, \dots)$

Summary

- The forward and backward variables α and β are computed with the above formular
- For $\alpha(u,t)$ it is:
  - Starting condition: $\alpha(0,0) = y^0_{l'_0},\ \alpha(1,0) = y^0_{l'_1},\ \alpha(u,0) = 0\ (\text{else})$
  - Recursion: $\alpha(u,t) =$



$$y^t_{l'_u} \cdot \begin{cases} \alpha(u,t-1) + \alpha(u-1,t-1), & l'_u = - \text{ or } l'_u = l'_{u-2} \\ \alpha(u,t-1) + \alpha(u-1,t-1) + \alpha(u-2,t-1), & else \end{cases}$$

Don‘t skip if blank

Don‘t skip if two identical characters

- Analogous for $\beta(u,t)$

Backpropagation: Loss derivative

- Use **Gradient Descent** as usual for training
- Loss function L must be derived by the weights W over the network output y, which is:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y_k^t} \frac{\partial y_k^t}{\partial W}$$

Required derivative of the loss (at time t for label k)

Derivative of the deepest network layer

Backpropagation: Loss derivative

- We need:

$$\frac{\partial L}{\partial y_k^t} = \frac{\partial}{\partial y} \sum_{n=0}^{N} \ell(x_n, z_n), \qquad \ell(x, l) = -\log P(l|x)$$

- So we are actually searching for:

$$\frac{\partial \ell(x,l)}{\partial y_k^t} = -\frac{1}{P(l|x)} \cdot \frac{\partial P(l|x)}{\partial y_k^t} = -\frac{1}{P(l|x)} \cdot \frac{\partial}{\partial y_k^t} \sum_u \frac{\alpha(u,t)\beta(u,t)}{y_{l'_u}^t}$$

- Only sums, when $l'(u) = k$:

$$\frac{\partial P(l|x)}{\partial y_k^t} = -\frac{1}{{y_k^t}^2} \cdot \sum_{u \,|\, l'_u = k} \alpha(u,t)\beta(u,t)$$

Backpropagation: Loss derivative

- Hence:

$$\frac{\partial \ell(x,l)}{\partial y_k^t} = \underbrace{\frac{1}{P(l|x)}}\cdot \frac{1}{y_k^t}\cdot \underbrace{\frac{1}{y_k^t}\cdot \sum_{u \,|\, l'_u = k} \alpha(u,t)\beta(u,t)}$$

Normalization of path probabilities by the total probability of the path

Probability of all paths, which pass through label k at time t

„Error of label probabilities“

Normalization of label probability

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.001	0.001	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

$$\frac{\partial \ell(x,l)}{\partial y_f^6} = -\frac{1}{P(affe|x)} \cdot \frac{1}{y_f^6} \cdot \frac{1}{y_f^6} \cdot \sum_{u\,|\,l'_u=f} \alpha(u,6)\beta(u,6)$$

$$= -\frac{1}{0.243} \cdot \frac{1}{0.1} \cdot \frac{1}{0.1} \cdot \left(\alpha(3,6)\beta(3,6) + \alpha(5,6)\beta(5,6)\right)$$

$$= -\frac{1}{0.243} \cdot \frac{1}{0.1} \cdot \frac{1}{0.1} \cdot (0.03 \cdot 0 + 0.017 \cdot 0.089)$$

$$= 0.623$$

α	0	1	2	3	4	5	6	7	8
-	0.9	0.36	0.036	0.029	0.009	🟥	🟥	🟥	🟥
a	0.09	0.495	0.684	0.108	0.014	0.002	🟥	🟥	🟥
-	🟥	0.036	0.053	0.590	0.209	0.045	🟥	🟥	🟥
f	🟥	0.008	0	0	0.419	0.257	0.030	🟥	🟥
-	🟥	🟥	0	0	0	0.168	0.383	🟥	🟥
f	🟥	🟥	0	0	0	0	0.017	0.240	🟥
-	🟥	🟥	🟥	0	0	0	0	0.005	🟥
e	🟥	🟥	🟥	0	0	0	0	0	0.243
-	🟥	🟥	🟥	🟥	0	0	0	0	0

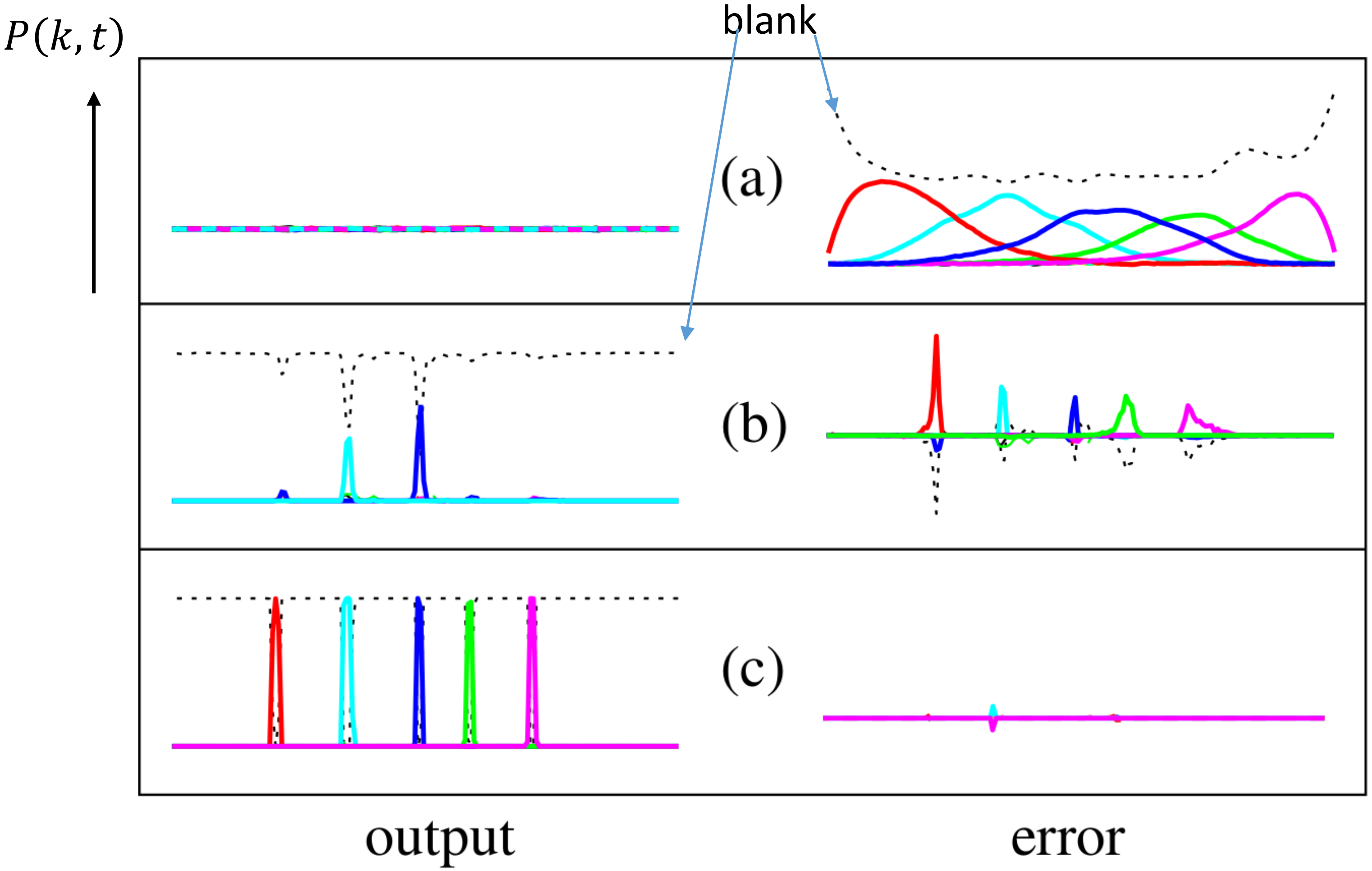
β	0	1	2	3	4	5	6	7	8
-	🟩					🟥	🟥	🟥	🟥
a	🟩						🟥	🟥	🟥
-	🟥						🟥	🟥	🟥
f	🟥						0.000	🟥	🟥
-	🟥	🟥					0.535	🟥	🟥
f	🟥	🟥					0.089	0.594	🟥
-	🟥	🟥	🟥				0.267	0.297	🟥
e	🟥	🟥	🟥				0.000	0.000	0.990
-	🟥	🟥	🟥	🟥			0.001	0.001	0.010

	0	1	2	3	4	5	6	7	8
P(-)	0.9	0.4	0.1	0.8	0.3	0.2	0.9	0.3	0.01
P(a)	0.09	0.5	0.8	0.15	0.1	0.1	0	0	0
P(b)	0	0	0	0	0	0.01	0	0.1	0
P(c)	0	0	0	0	0	0.09	0	0	0
P(d)	0	0.09	0.1	0.05	0	0	0	0	0
P(e)	0.01	0	0	0	0	0.2	0	0	0.99
P(f)	0	0.01	0	0	0.6	0.4	0.1	0.6	0

$$\frac{\partial \ell(x,l)}{\partial y_{-}^{6}} = -\frac{1}{P(affe|x)} \cdot \frac{1}{y_{-}^{6}} \cdot \frac{1}{y_{-}^{6}} \cdot \sum_{u \,\mid\, l'_u = -} \alpha(u,6)\beta(u,6)$$

$$= -\frac{1}{0.243} \cdot \frac{1}{0.9} \cdot \frac{1}{0.9} \cdot \big(\alpha(0,6)\beta(0,6) + \alpha(2,6)\beta(2,6) + \alpha(4,6)\beta(4,6) + \alpha(6,6)\beta(6,6) + \alpha(8,6)\beta(8,6)\big)$$

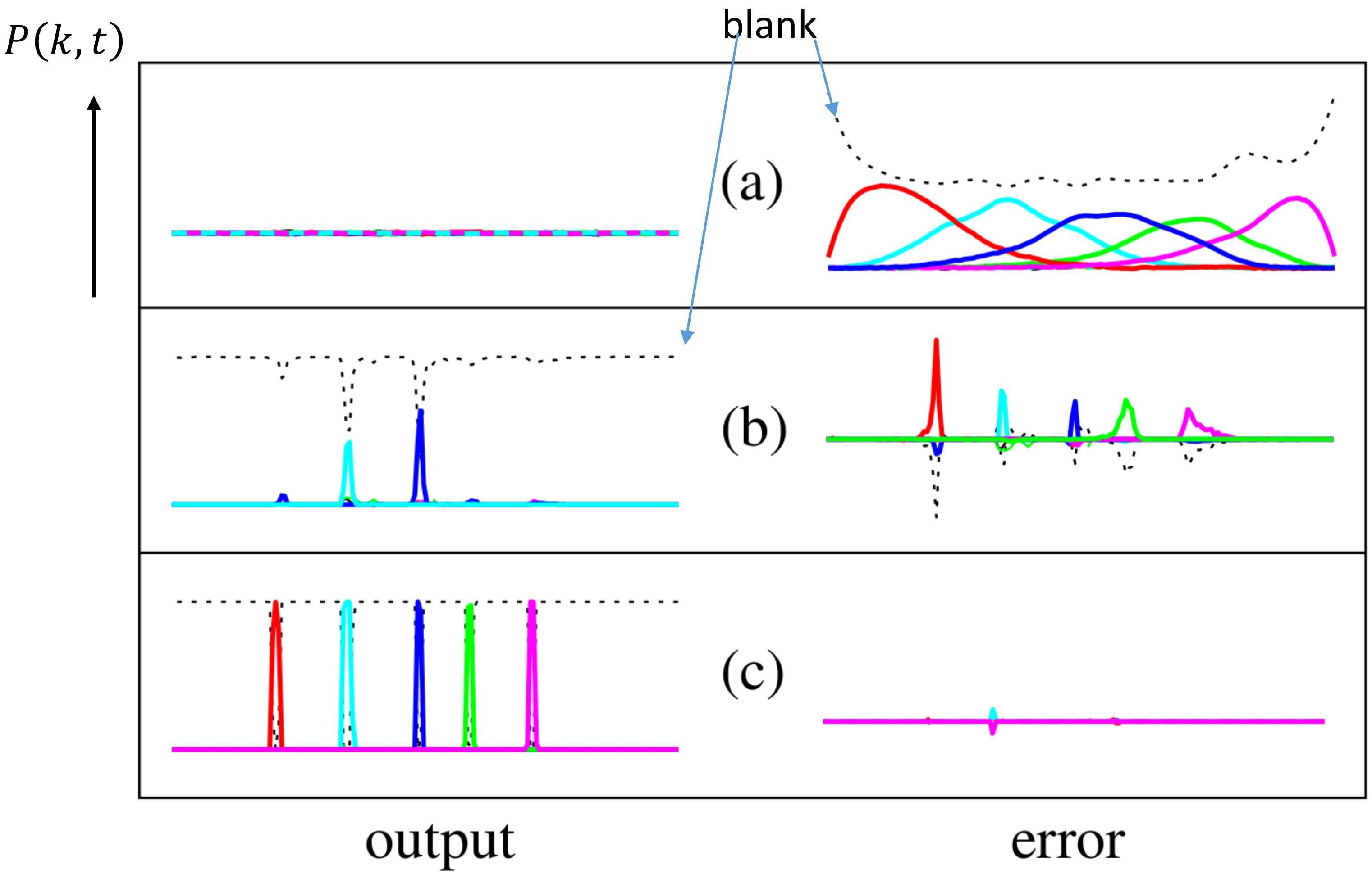
Error and label probabilities



a) At training start:

- all probabilities are (approximately) the same (noise),
- the error is split (sequence abcde) over the entire time span

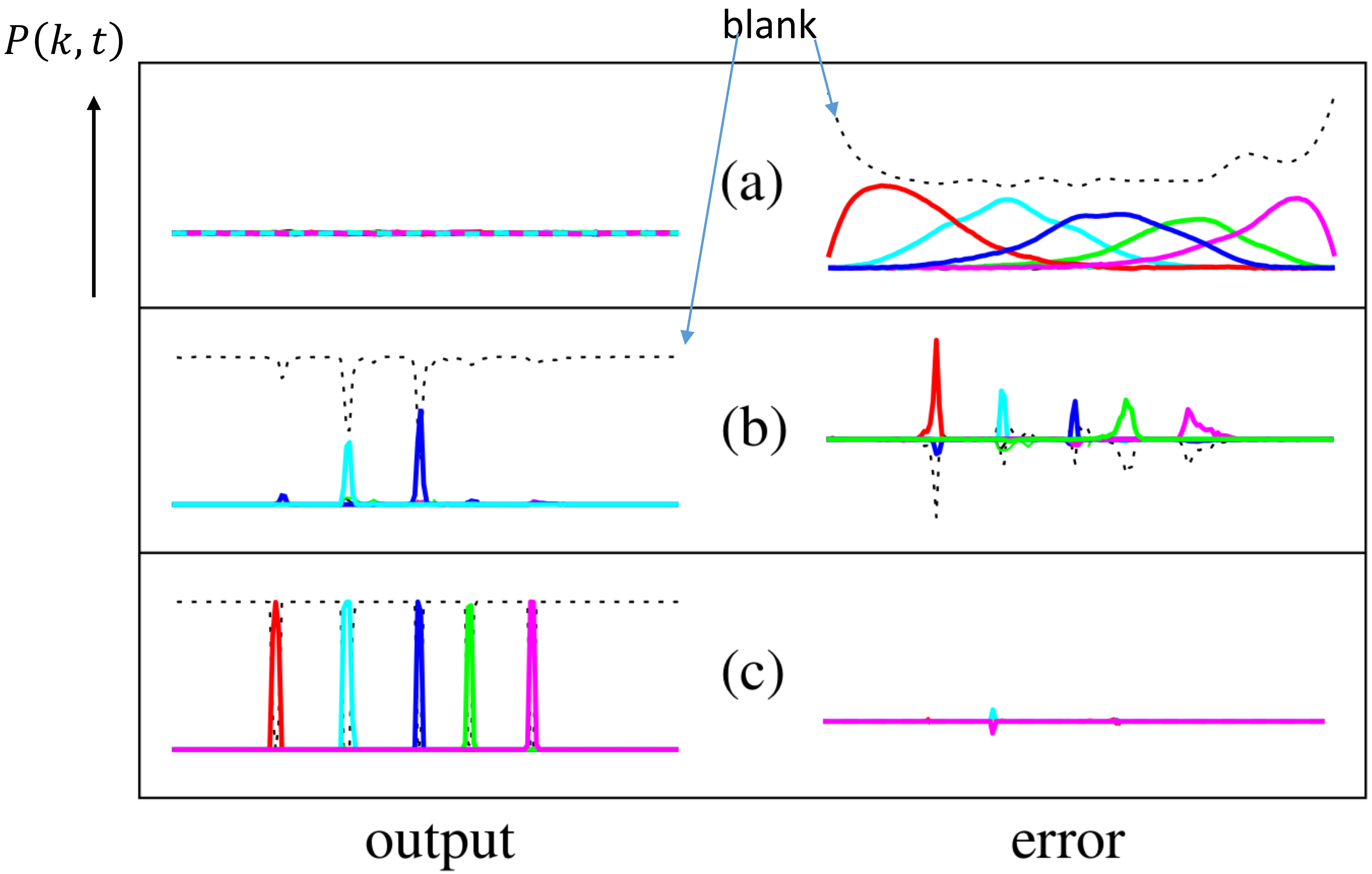
Error and label probabilities



b) During training (very early):

- Predicting blank makes sense (most common class)
- Singular labels are learned at distinct places as peaks (error and label probability)

Error and label probabilities



c) At training end

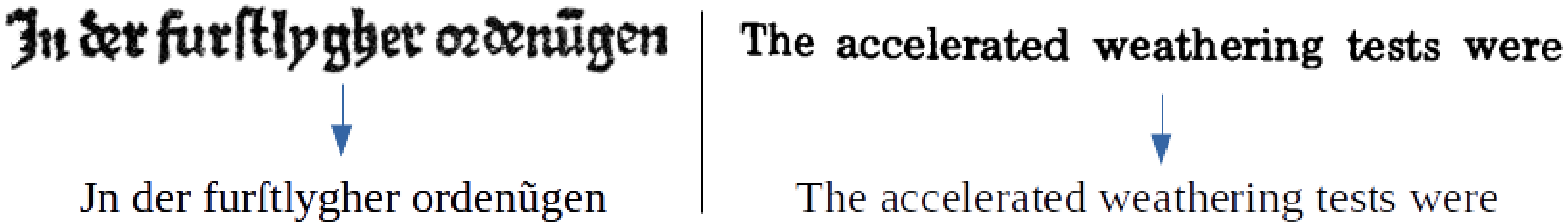
- Blank stays main class
- Label sequence is learned correctly as single peaks
- The error is becoming ever smaller

Summary

- The presented loss is called CTC Loss:

  Connectionist Temporal Classification
- It is used, when the input is a sequence of time of length T and the output is an unaligned label sequence of length $\leq T$
- Decoding the probability matrix to a label sequence is a problem that can be defined separately, but standard CTC greedy decoder (argmax and using $\mathcal{B}$=remove double labels and blanks) already returns good results
- Alternative decoders use beam search and consider transition probabilities (e.g. language models/dictionary).

Application OCR: Calamari

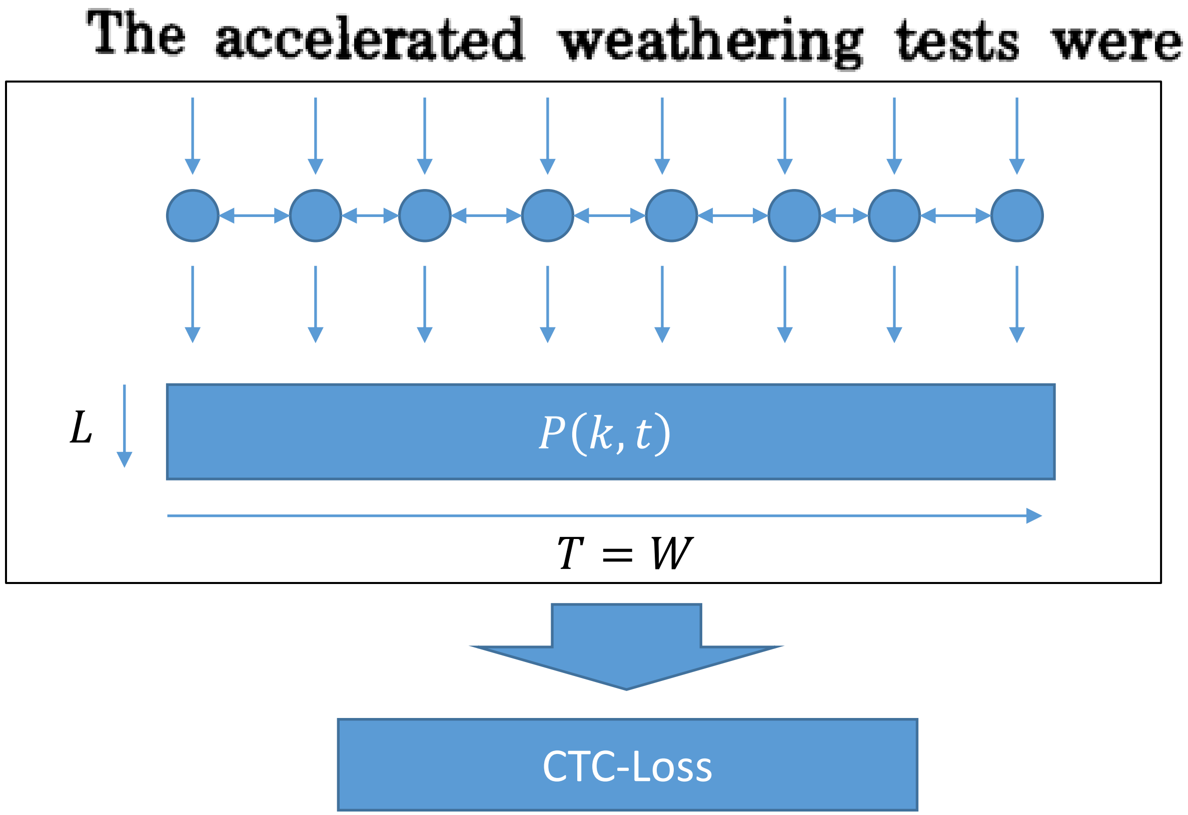

Jn der furſtlygher ordenũgen

The accelerated weathering tests were

- Input: Image of a line, Output: character sequence
- Neural Network (LSTM/CNN) for computing the probability matrix
- CTC-Loss with „time“ as the width of the image

Application OCR: Calamari

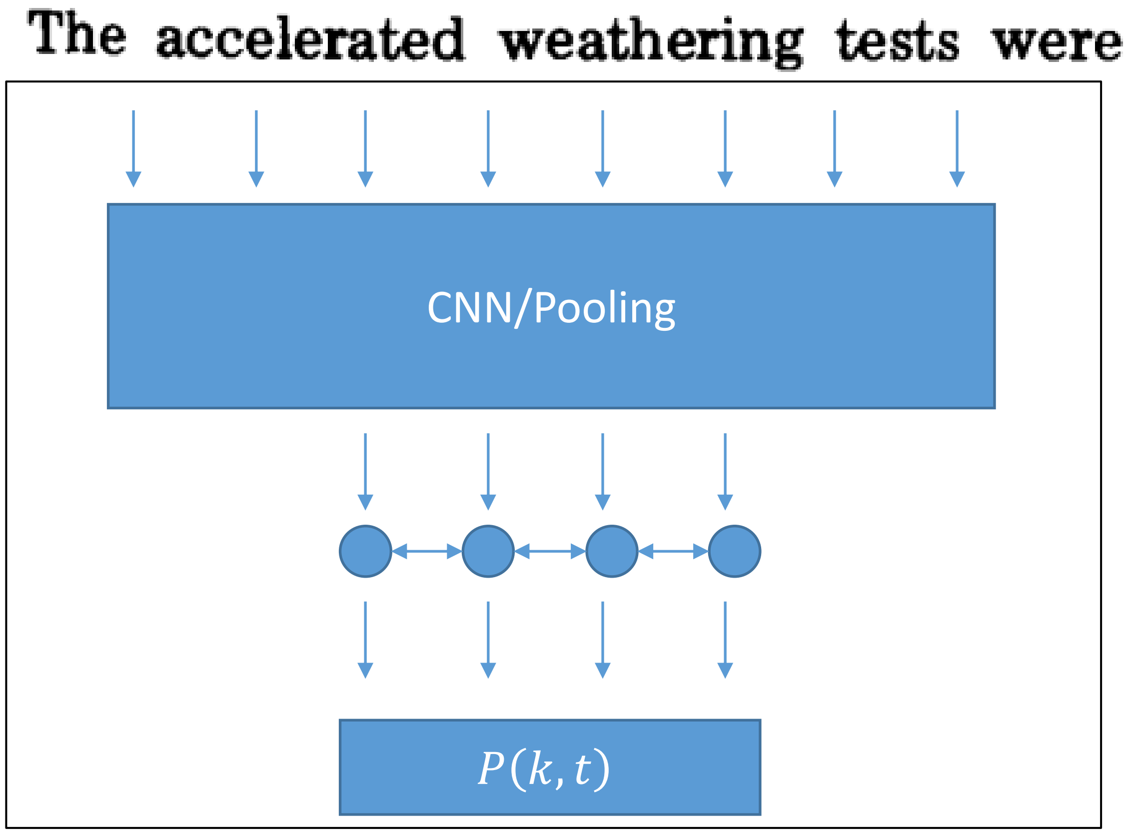
- Easiest network: LSTM with CTCloss



- Bidirektional LSTM with h hidden nodes ($\rightarrow \mathrm{T} \times 2h$)
- Fully Connected with L nodes ($\rightarrow T \times L$)
- Softmax ($\rightarrow T \times L$)

Application OCR: Calamari

- Complex Network: CNN-LSTM-Hybrid with CTC loss



- CNN pooling with 2 times 2x2 Pooling and last layer with C filters ($\rightarrow \frac{W}{4} \times \frac{H}{4} \times C$)
- Reshaping/Transpose ($\rightarrow \frac{W}{4} \times \frac{H}{4} \cdot C, T = \frac{W}{4}$)
- Bidirektional LSTM with h hidden nodes ($\rightarrow T \times 2h$)
- Fully Connected with L nodes ($\rightarrow T \times L$)
- Softmax ($\rightarrow T \times L$)

Probabilities in „Log“ space

- Problem: When computing α and β many probabilities $(\geq 0, \leq 1)$ are multiplied
- Consequence:
  - Very small numbers are generated (e.g. $0.5^{80} = 8.3 \cdot 10^{-25}$)
  - Problems with decimal accuracy aries
- Logarithm helps!
  - Changes product to sum!
  - $\log a \cdot \log b = \log(a + b)$
  - E.g. $\log 0.5^{80} = -55$

Probabilities in „Log“ space

- Now don‘t work with P (probabilities) anymore, but always with $\log P$:
  - From all $\log y_k^t = {y'}_k^t, \log \alpha = \alpha', \log \beta = \beta'$
  - From recursion:

$$\alpha(u,t) = y_{l'_u}^t \cdot \begin{cases} \alpha(s,t-1) + \alpha(s-1,t-1), & l'_u = - \text{ or } l'_u = l'_{u-2} \\ \alpha(s,t-1) + \alpha(s-1,t-1) + \alpha(s-2,t-1), & \text{else} \end{cases}$$

$$\alpha'(u,t) = {y'}_{l'_u}^t + \begin{cases} \alpha'(s,t-1) \oplus \alpha'(s-1,t-1), & l'_u = - \text{ or } l'_u = l'_{u-2} \\ \alpha'(s,t-1) \oplus \alpha'(s-1,t-1) \oplus \alpha'(s-2,t-1), & \text{else} \end{cases}$$

  - $\oplus$ is the LogSumExp LSE :
    - $\alpha'_1 \oplus \alpha'_2 \equiv \log(\alpha_1 + \alpha_2) = \log(\exp \alpha'_1 + \exp \alpha'_2) = \text{LSE}(\alpha'_1, \alpha'_2)$

Probabilities in „Log“ space

Questions for comprehension:

1. Can positive values appear in log space? Why?

   No, see 2.

2. Hence, what is the range of values of log space?

   $[-\infty, 0]$ (Range of values of logarithm for numbers <= 1)

3. What value does a probability need in normal space, to reach the smallest value?

   0

Probabilities in „Log“ space

4. What is the LSE-formula, if a term has this value and how should it be implemented?

   $\log(\exp -\infty + \exp \alpha') = \log(0 + \exp \alpha) = \alpha'$, implement as an „if“

5. What is the behaviour of addition of log-probabilities, i.e. multiplying in normal space?

   $-\infty + \alpha' = -\infty$, in normal space: $0 \cdot \alpha = 0$

Probabilities in „Log“ space

6. Probabilities often come from softmax. Is it useful to combine the CTC loss with softmax in log space?

Yes, softmax uses the exponential function, which is also numerically „unstable“. Analogous to Cross Entropy Loss official implementations always use log space and softmax is combined with the loss:

$$\log \frac{e^{x_i}}{\sum_j e^{x_j}} = x_i - \log \sum_j e^{x_j} = x_i - \text{LSE}\,(x)$$

Outlook

- Exercise: CTC algorithm
  - Architecture in pseudo code
  - Questions for comprehension
  - Compute forward and backward variables
- Next lecture
  - Application: Medical Image processing
  - X-ray Classification
  - Polyp detection in gastroenterology