OCR实战

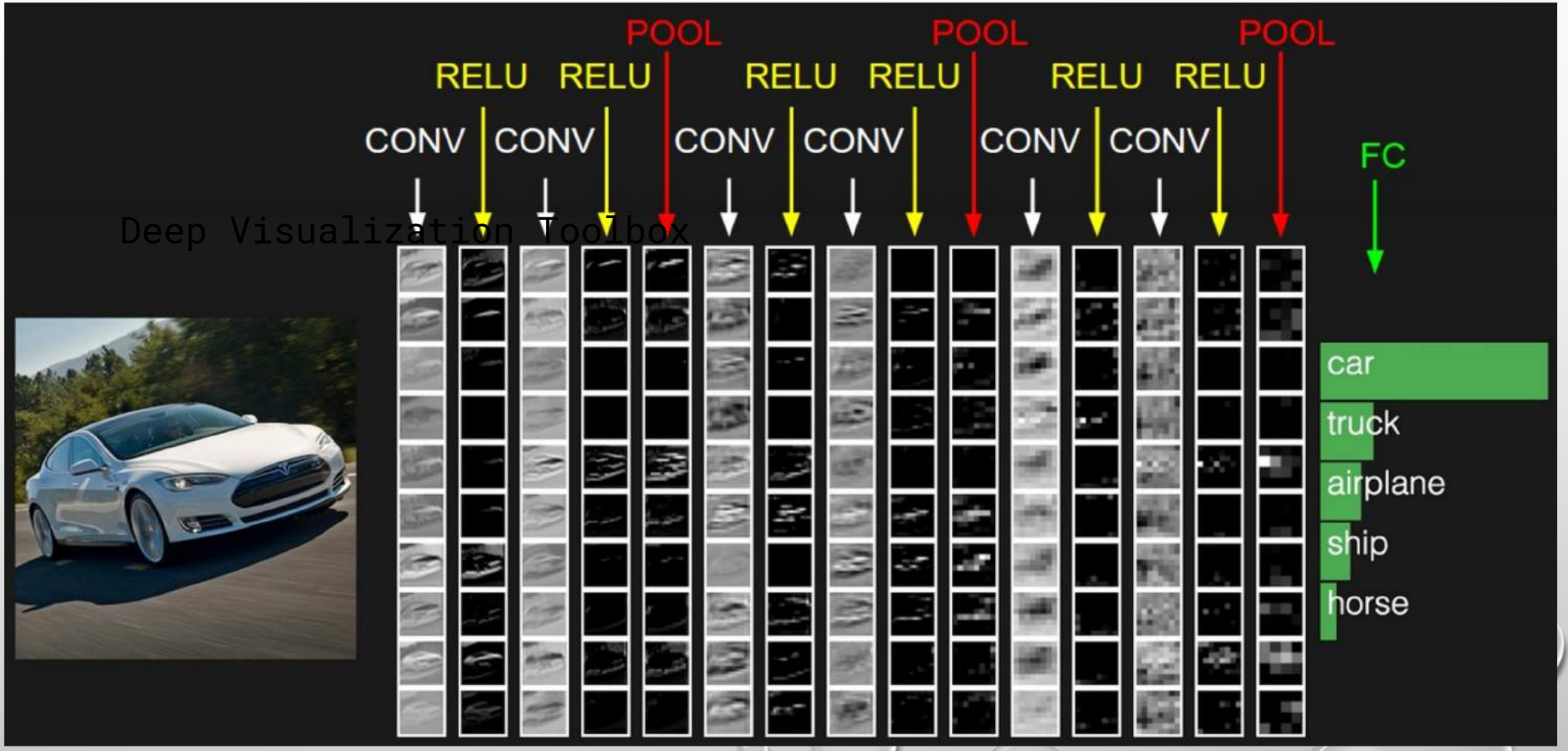
第三课 序列识别

第二课 序列识别

- RNN
  - LSTM
  - GRU
- CRNN
  - CTC
- RARE
  - ATTENTION

单字符识别

- CNN模型


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
Deep Visualization Toolbox
car
truck
airplane
ship
horse

单字符识别

- CNN模型

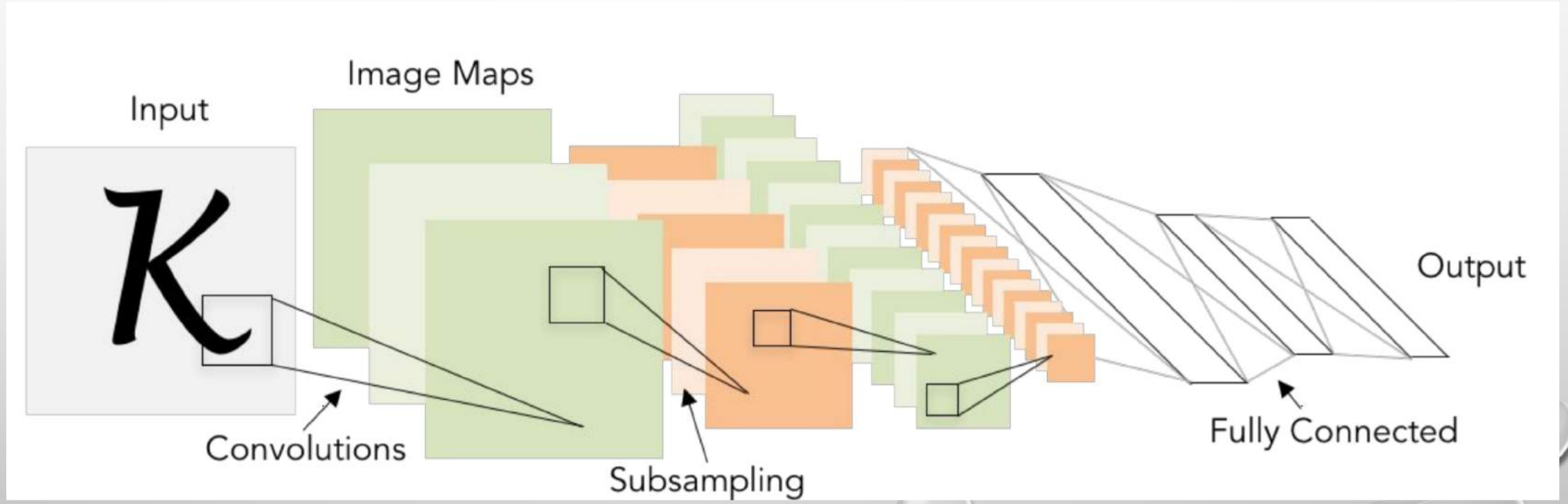


Image Maps
Input
Output
Convolutions
Subsampling
Fully Connected

- 字符分割

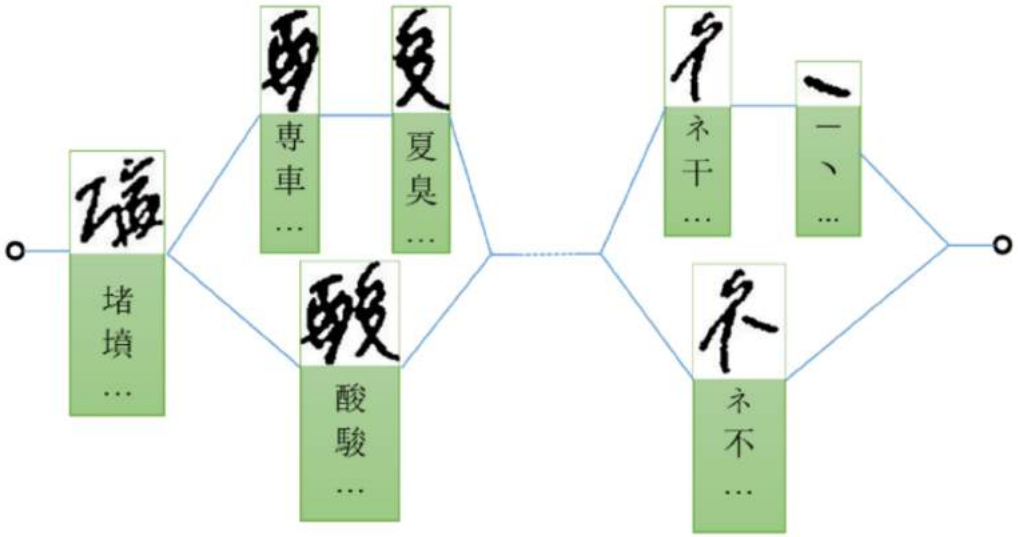
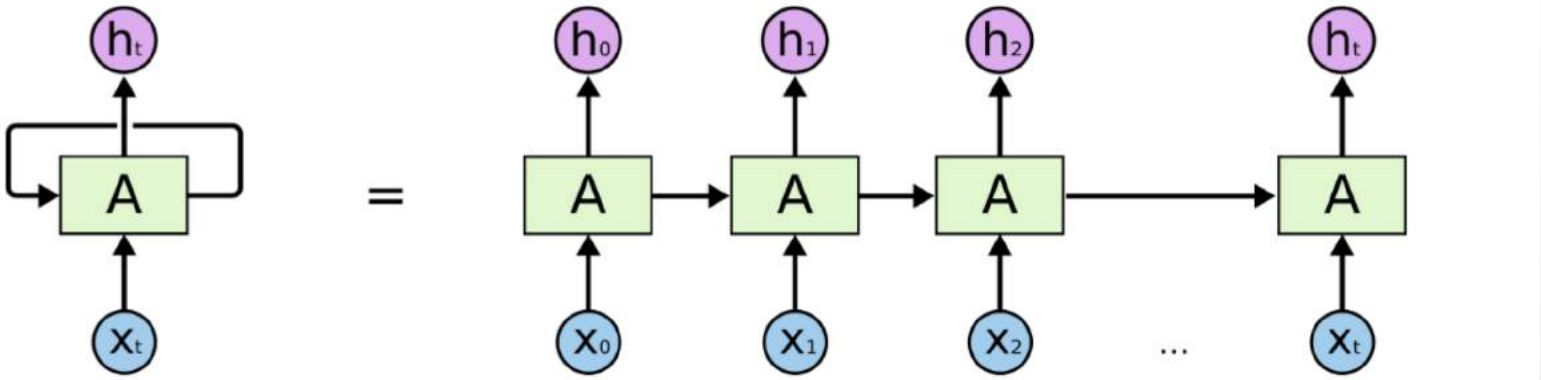


Fig. 11: Candidate lattice diagram.

第二课 序列识别

- RNN
  - LSTM
  - GRU
- CRNN
  - CTC
- RARE
  - ATTENTION

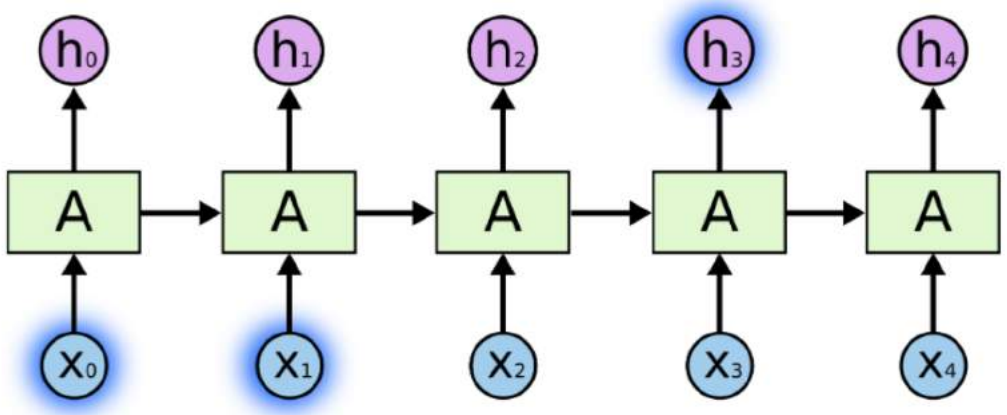
RNN



An unrolled recurrent neural network.

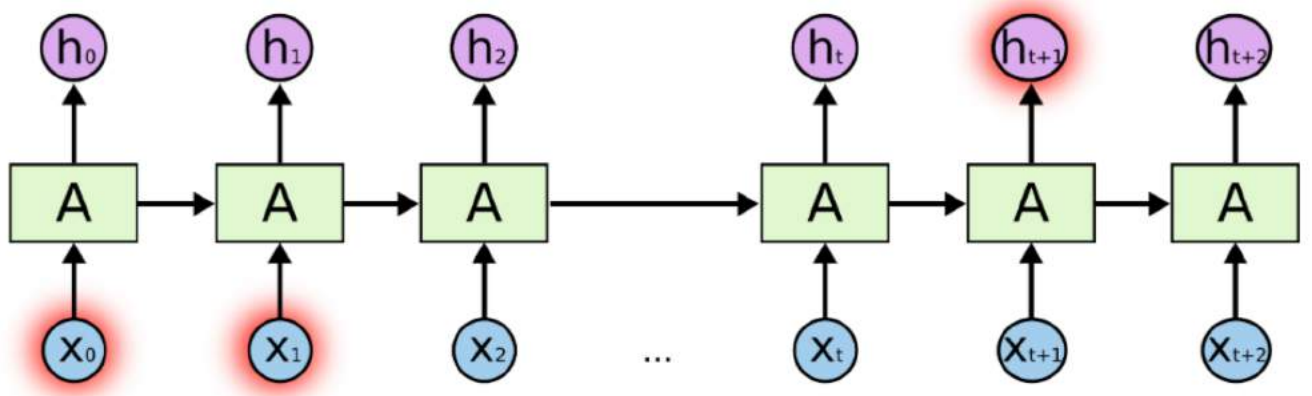
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN



"the clouds are in the *sky*,"

"I grew up in France... I speak fluent *French*."



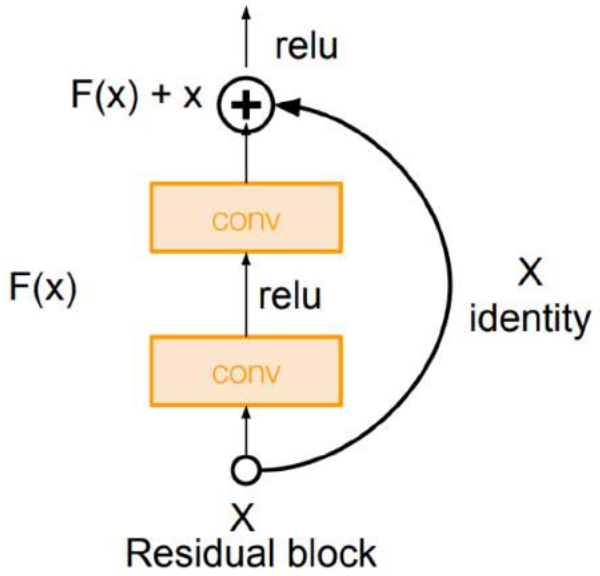
CNN

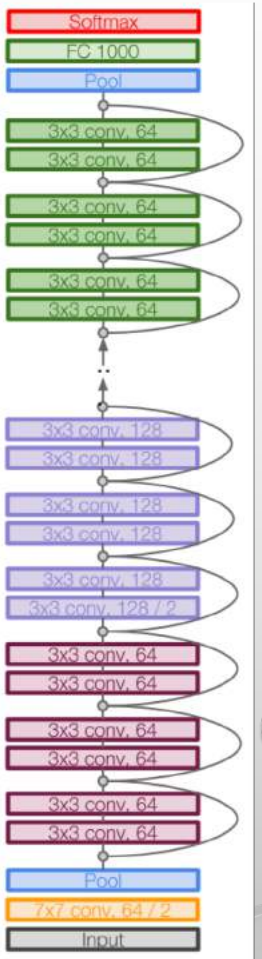
Case Study: ResNet

[He et al., 2015]

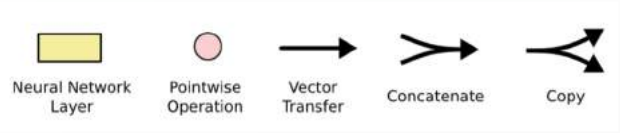
Very deep networks using residual connections

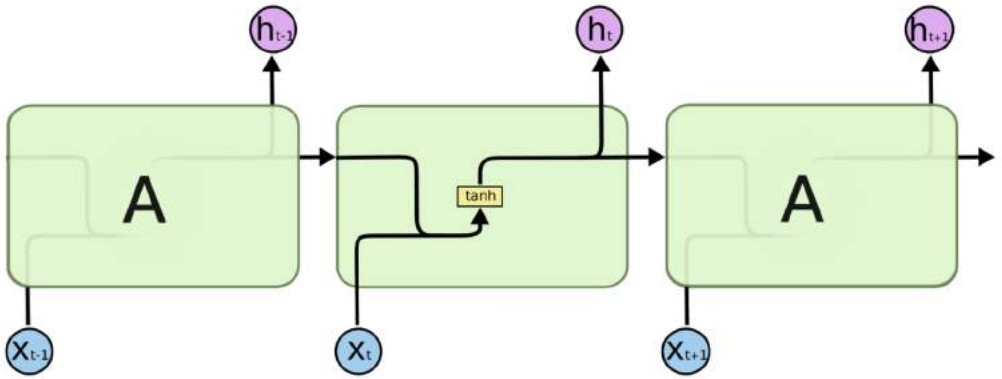
- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!





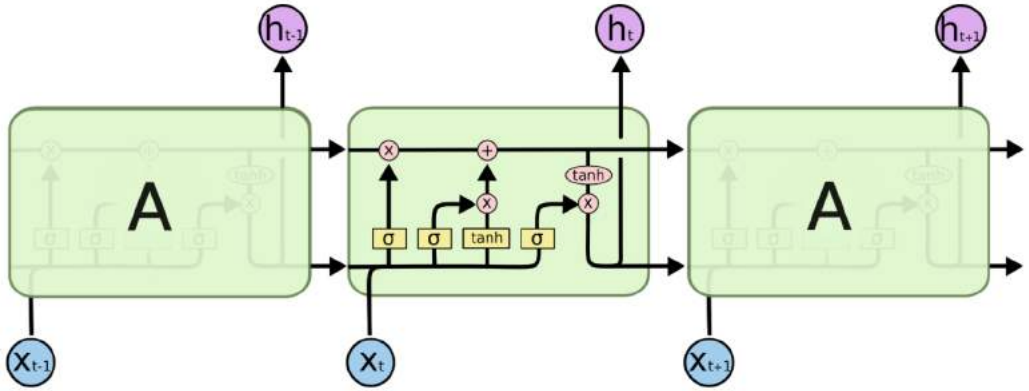
LSTM





The repeating module in a standard RNN contains a single layer.

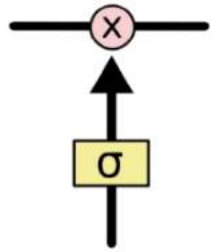
"the clouds are in the *sky*,"



The repeating module in an LSTM contains four interacting layers.

"I grew up in France... I speak fluent *French*."

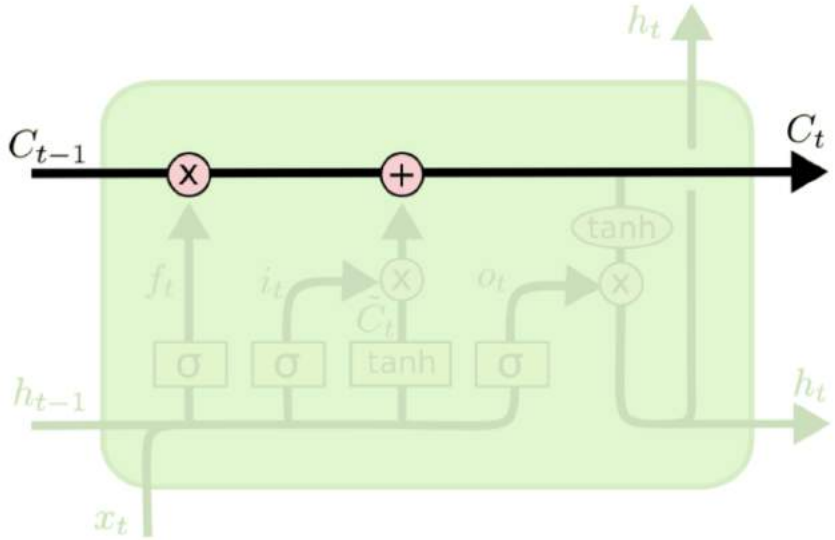
Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.



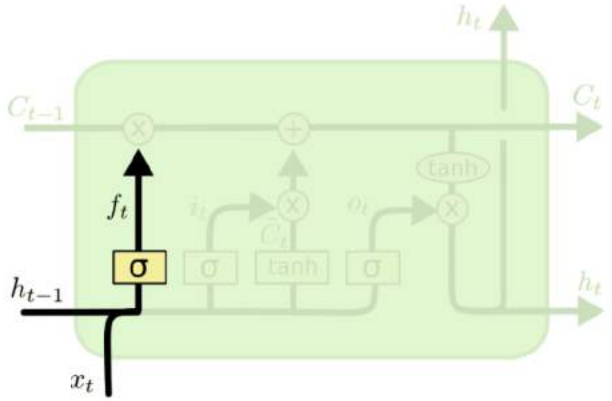
LSTM

The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.

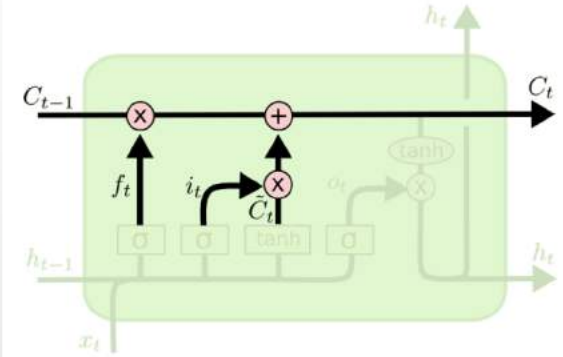
The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



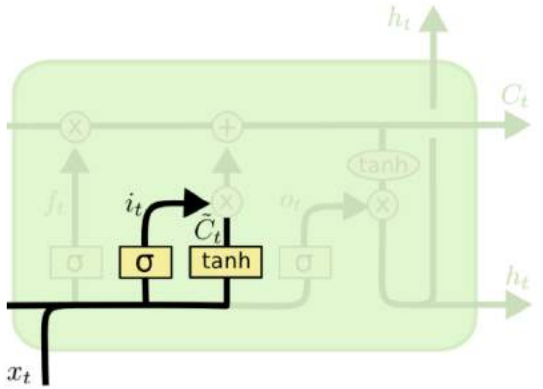
LSTM



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

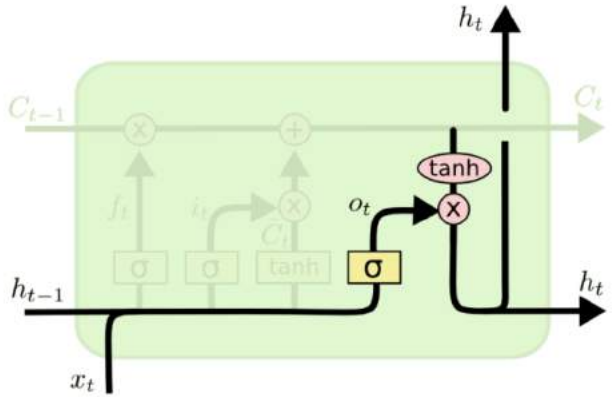


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

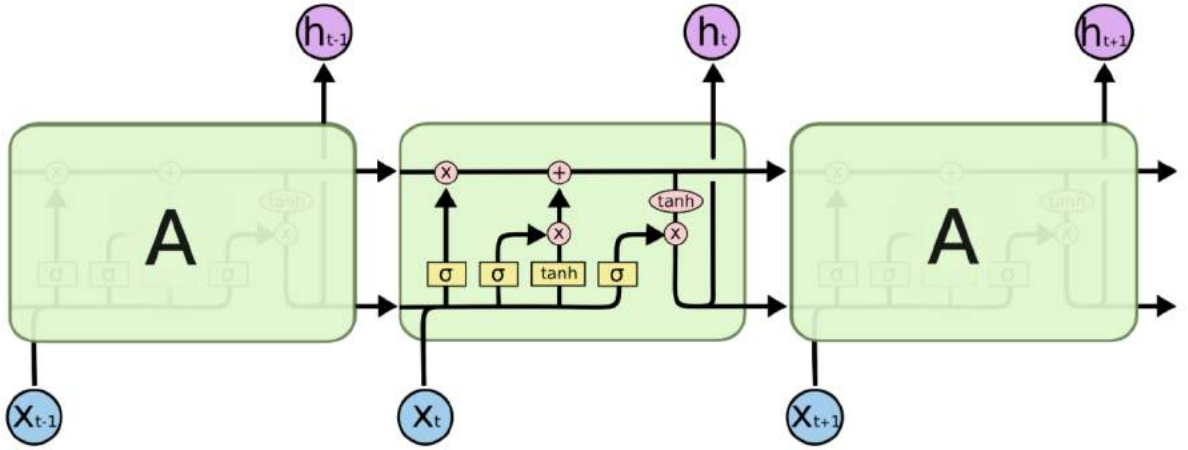
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

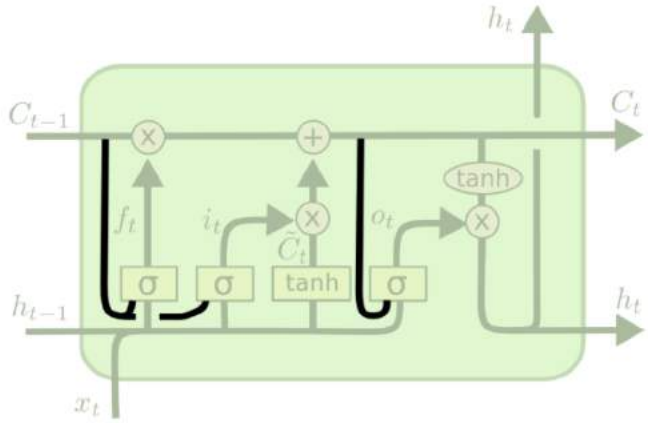
$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM



The repeating module in an LSTM contains four interacting layers.

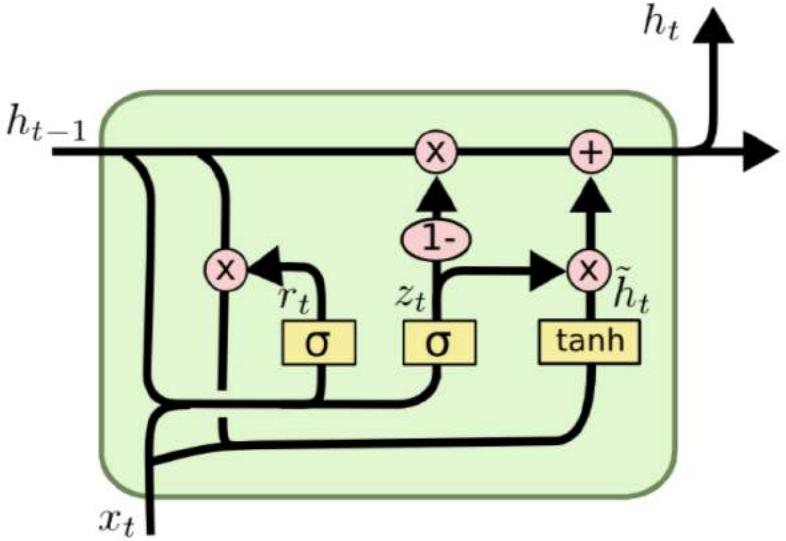
One popular LSTM variant, introduced by Gers & Schmidhuber (2000), is adding "peephole connections." This means that we let the gate layers look at the cell state.



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

GRU

A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes. The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.



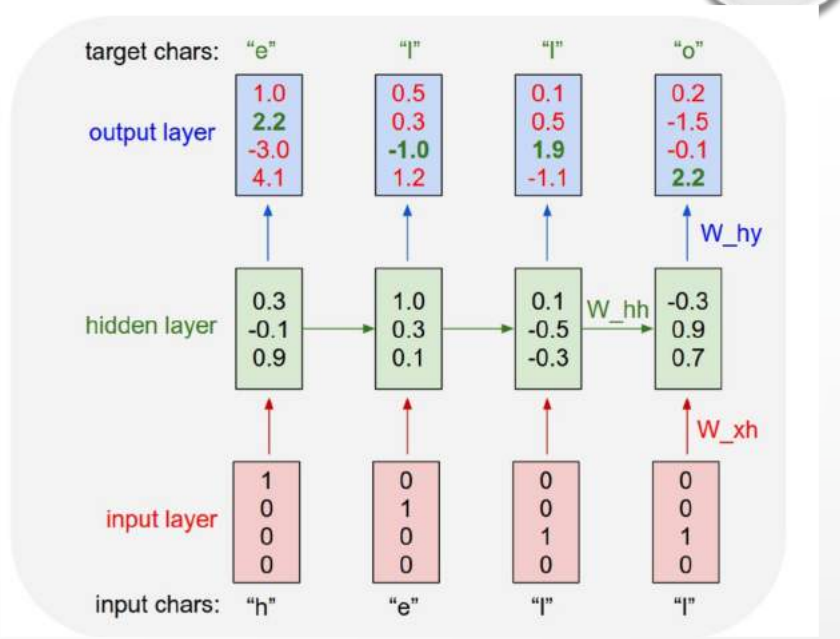
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

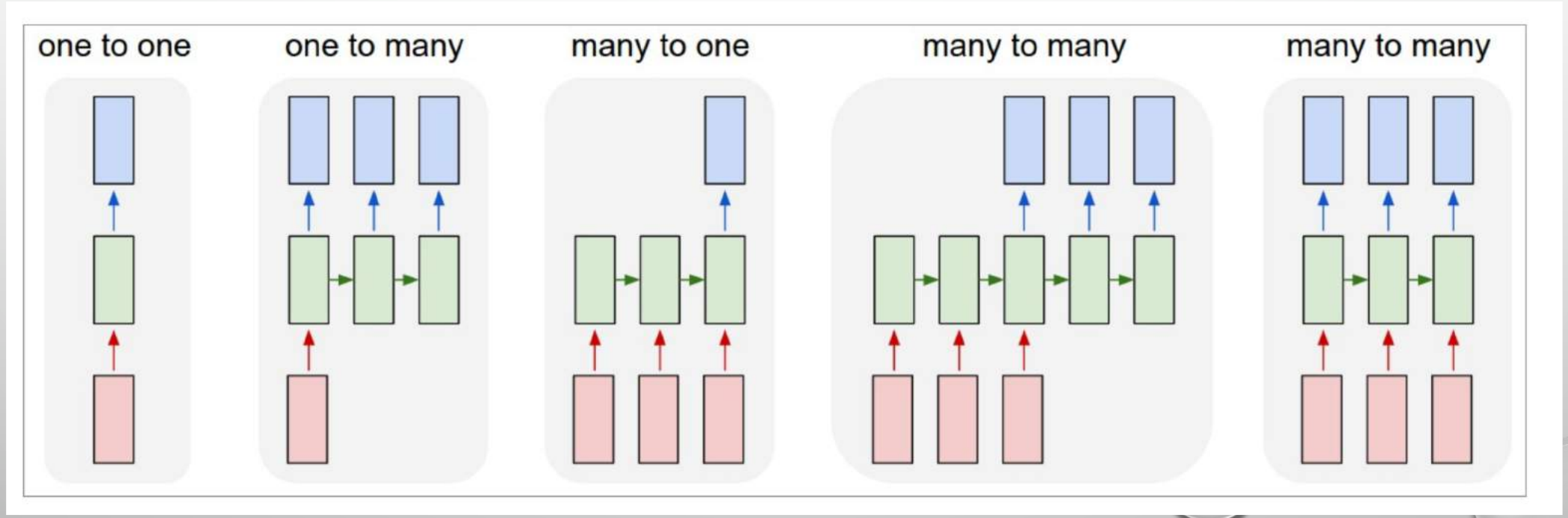
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

RNN

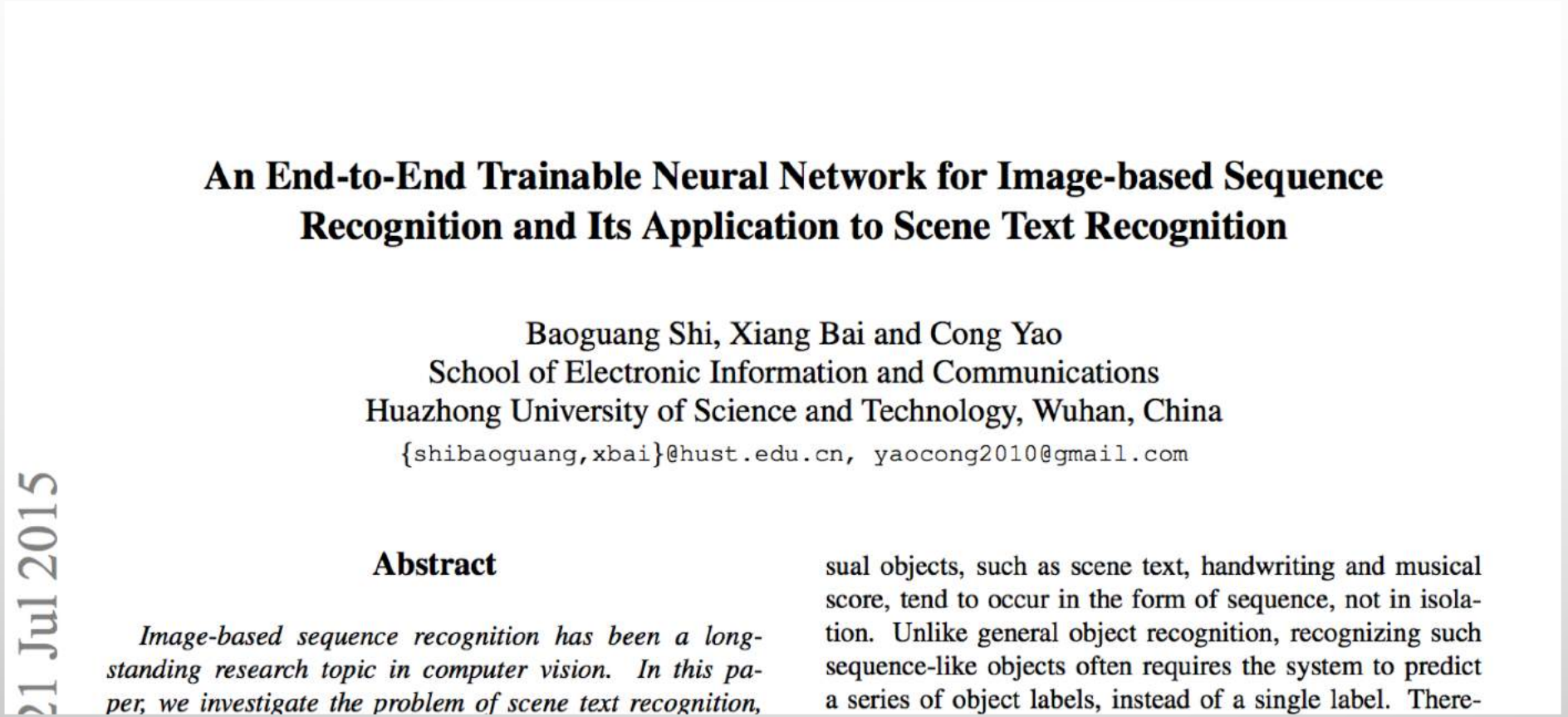

target chars: "e" "l" "l" "o"
output layer
1.0 2.2 -3.0 4.1
0.5 0.3 -1.0 1.2
0.1 0.5 1.9 -1.1
0.2 -1.5 -0.1 2.2
W_hy
hidden layer
0.3 -0.1 0.9
1.0 0.3 0.1
0.1 -0.5 -0.3
W_hh
-0.3 0.9 0.7
W_xh
input layer
1 0 0 0
0 1 0 0
0 0 1 0
0 0 1 0
input chars: "h" "e" "l" "l"


one to one
one to many
many to one
many to many
many to many

第二课 序列识别

- RNN
  - LSTM
  - GRU
- CRNN
  - CTC
- RARE
  - ATTENTION

CRNN



An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition

Baoguang Shi, Xiang Bai and Cong Yao
School of Electronic Information and Communications
Huazhong University of Science and Technology, Wuhan, China
{shibaoguang,xbai}@hust.edu.cn, yaocong2010@gmail.com

Abstract

Image-based sequence recognition has been a long-standing research topic in computer vision. In this paper, we investigate the problem of scene text recognition,

sual objects, such as scene text, handwriting and musical score, tend to occur in the form of sequence, not in isolation. Unlike general object recognition, recognizing such sequence-like objects often requires the system to predict a series of object labels, instead of a single label. There-

http://cn.arxiv.org/pdf/1507.05717v1

CRNN

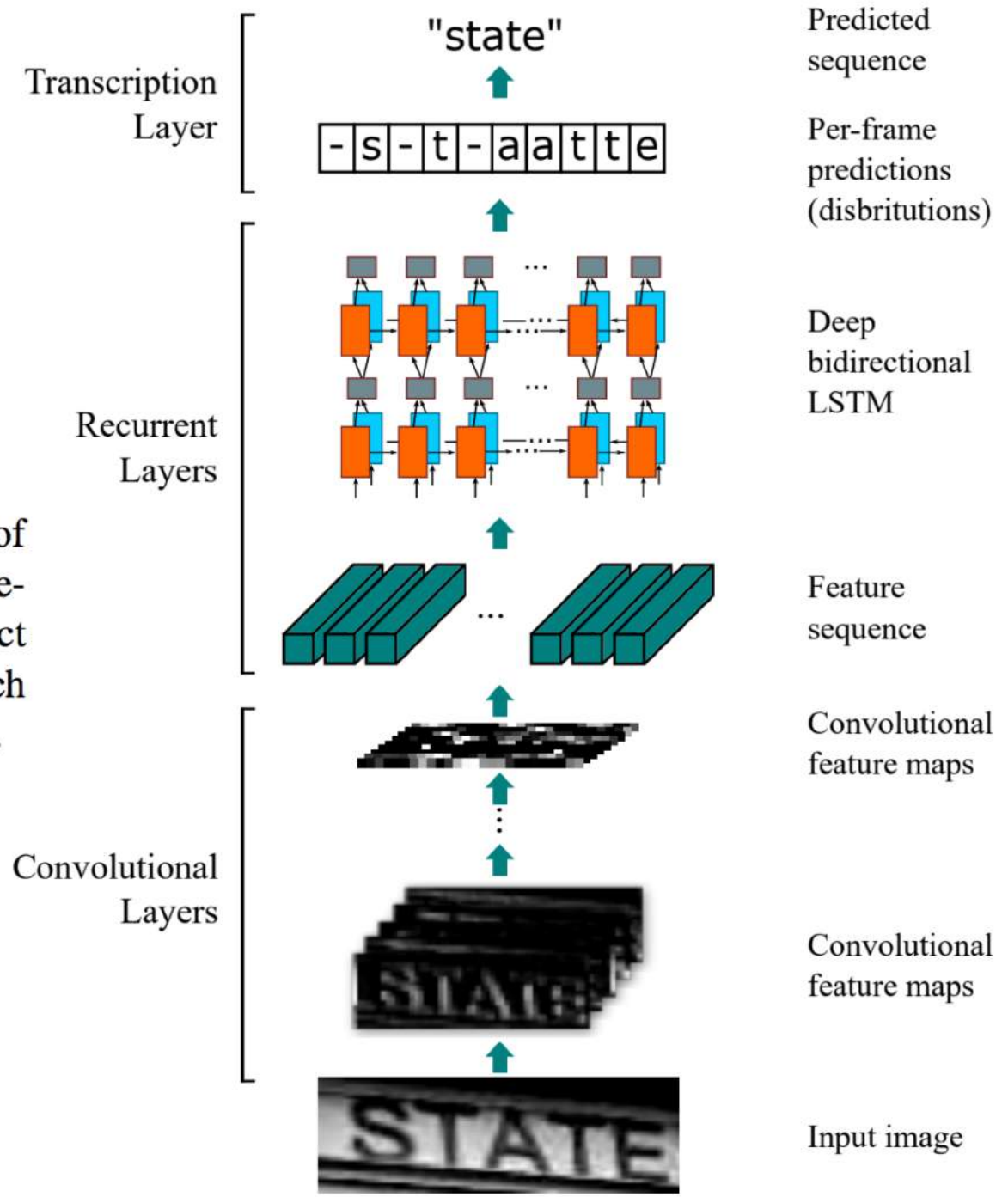
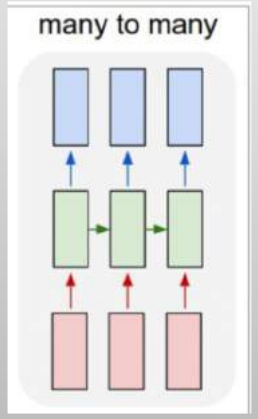
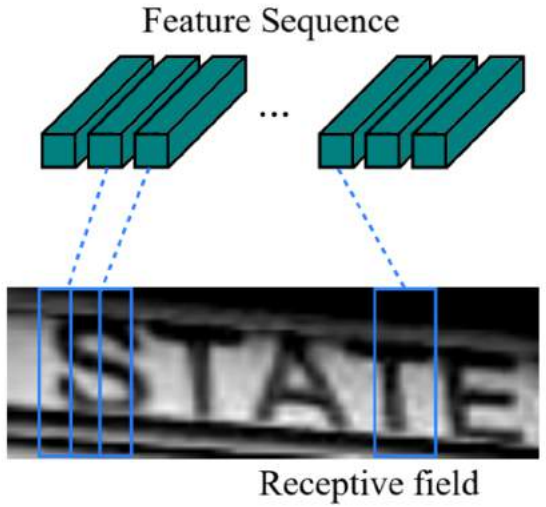


Figure 1. The network architecture. The architecture consists of three parts: 1) convolutional layers, which extract a feature sequence from the input image; 2) recurrent layers, which predict a label distribution for each frame; 3) transcription layer, which translates the per-frame predictions into the final label sequence.

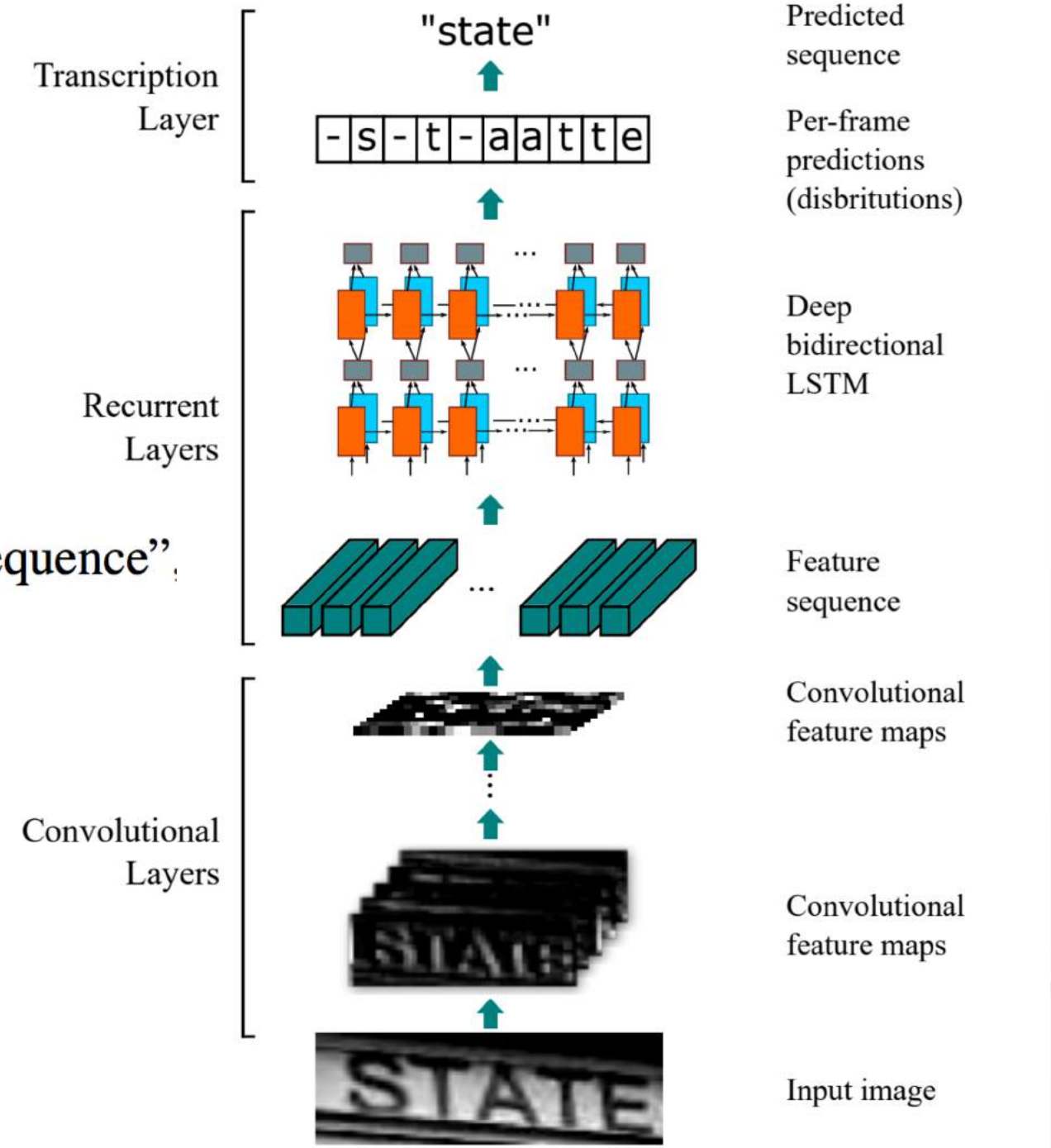


CRNN



“Map-to-Sequence”

Figure 2. The receptive field. Each vector in the extracted feature sequence is associated with a receptive field on the input image, and can be considered as the feature vector of that field.



CRNN

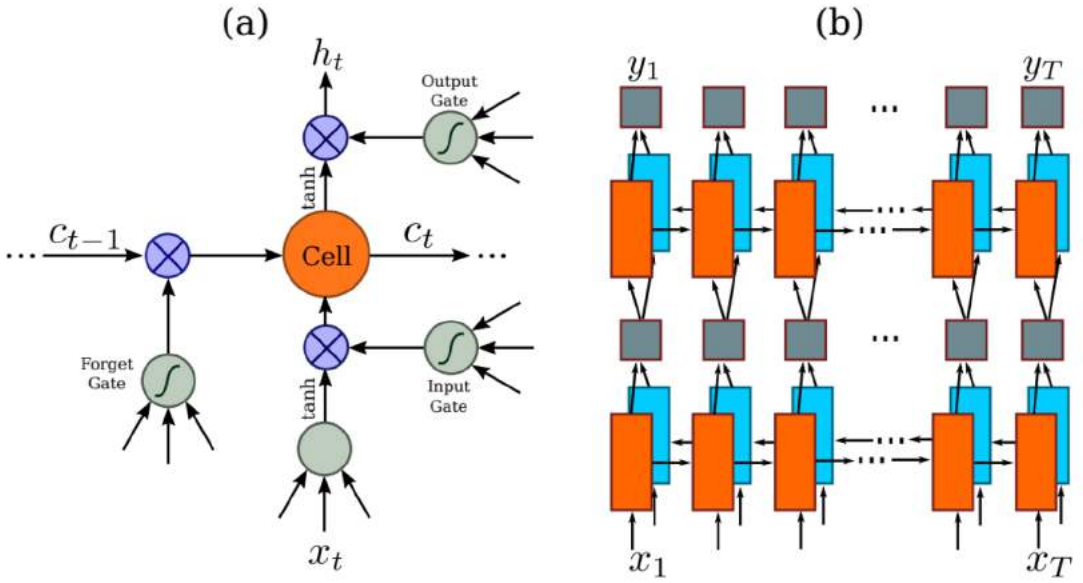
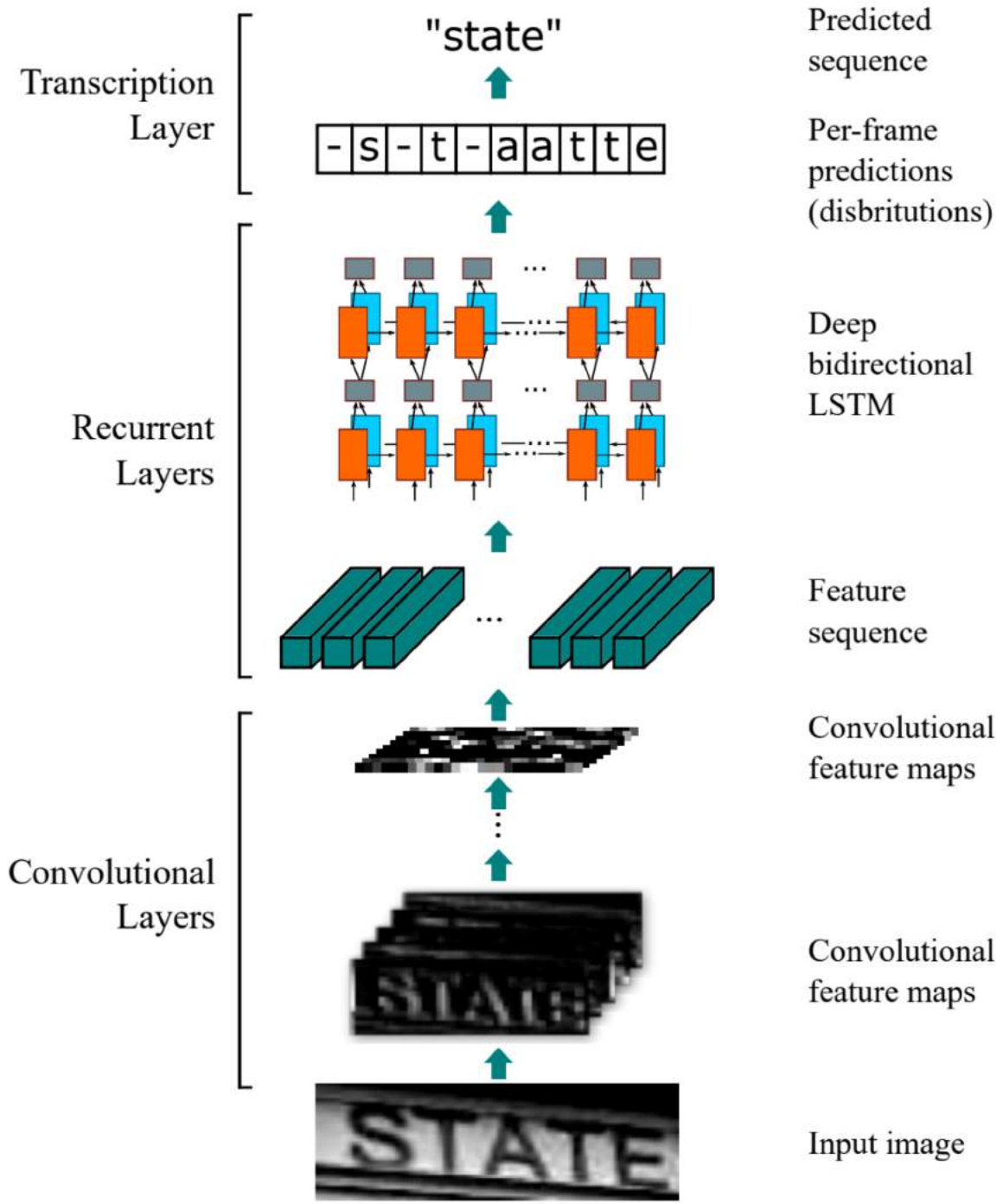


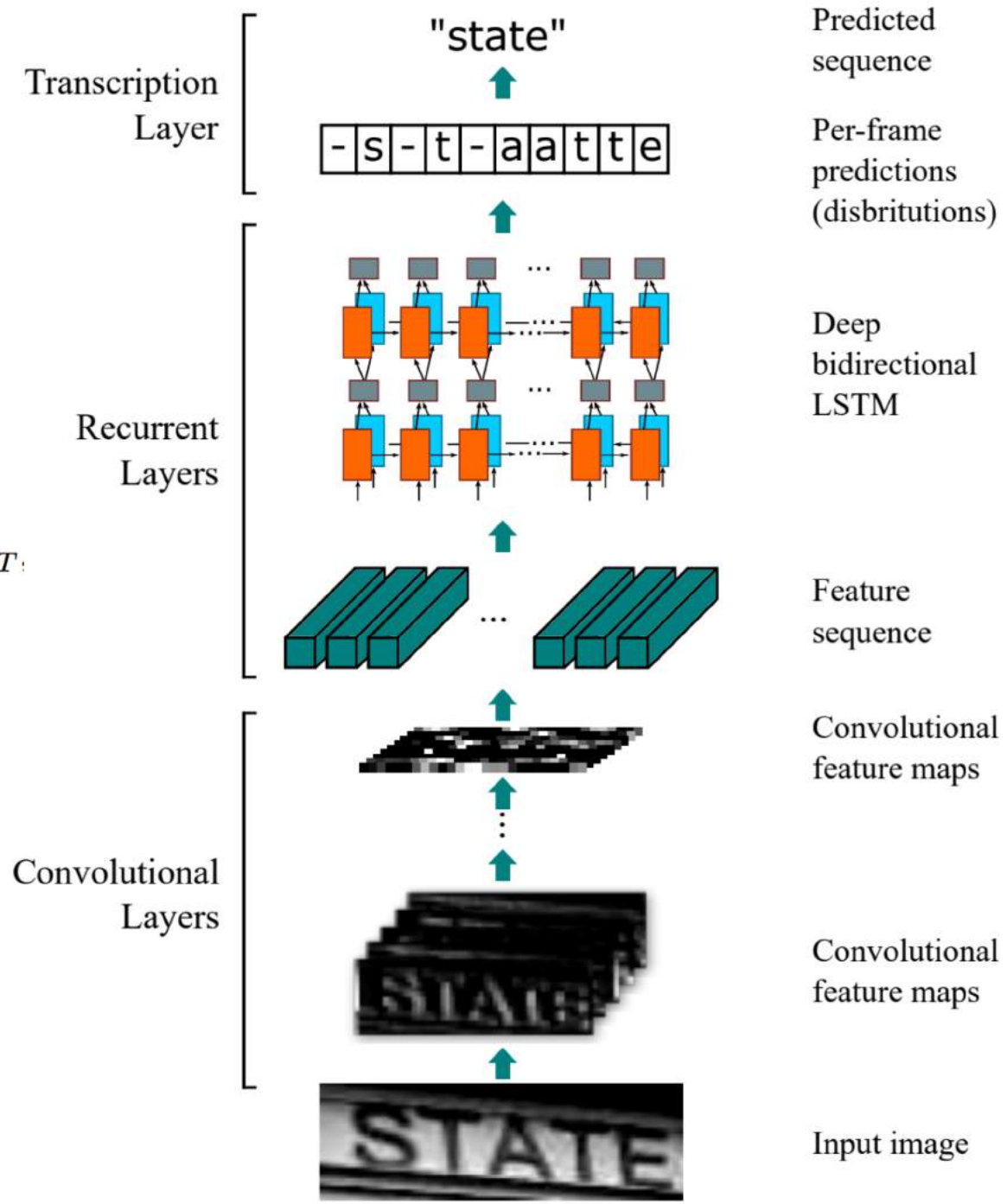
Figure 3. (a) The structure of a basic LSTM unit. An LSTM consists of a cell module and three gates, namely the input gate, the output gate and the forget gate. (b) The structure of deep bidirectional LSTM we use in our paper. Combining a forward (left to right) and a backward (right to left) LSTMs results in a bidirectional LSTM. Stacking multiple bidirectional LSTM results in a deep bidirectional LSTM.



CRNN

Connectionist Temporal Classification (CTC)

The probability is defined for label sequence **l** conditioned on the per-frame predictions $\mathbf{y} = y_1, \ldots, y_T$

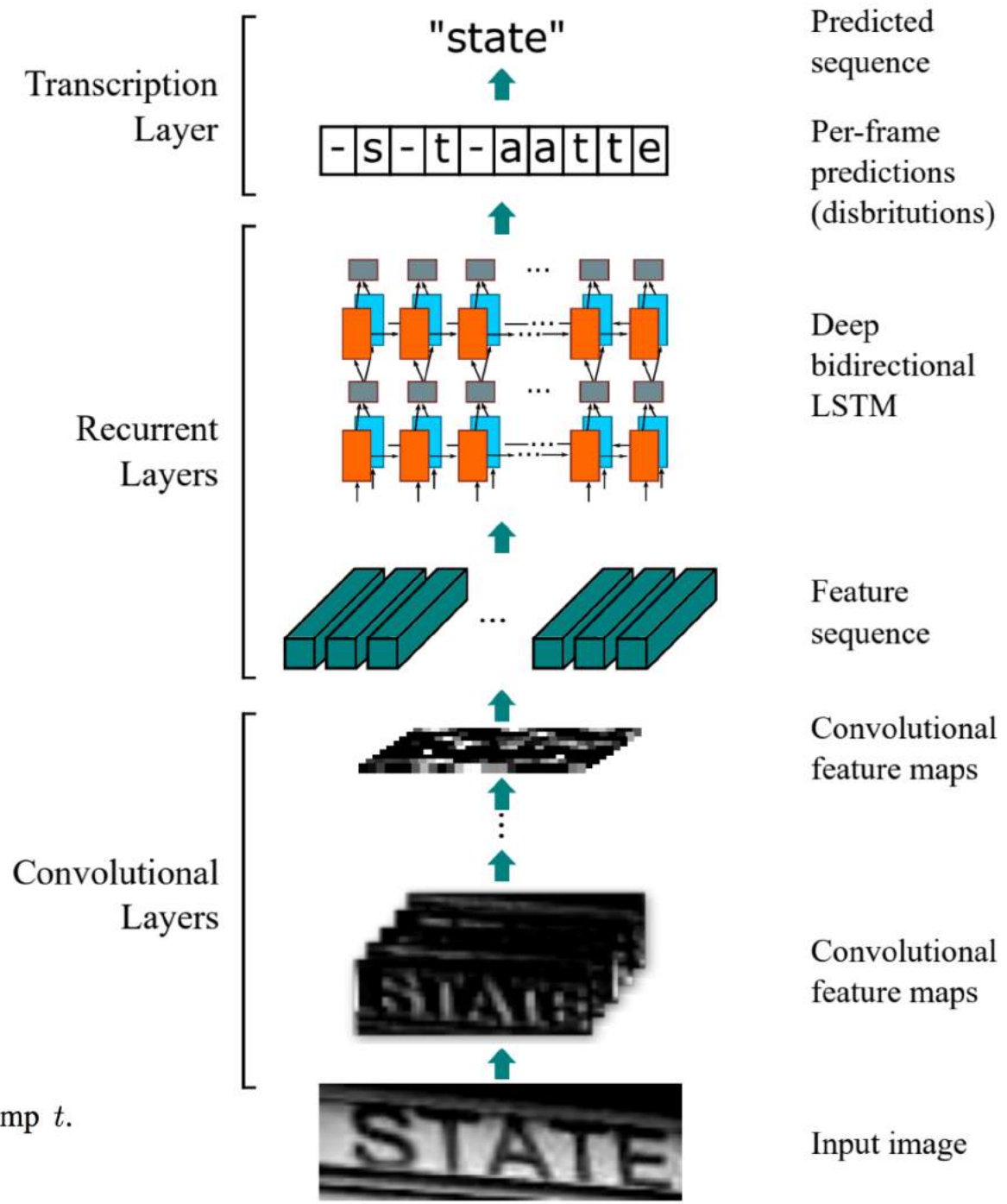


CRNN

The formulation of the conditional probability is briefly described as follows: The input is a sequence $\mathbf{y} = y_1, \ldots, y_T$ where T is the sequence length. Here, each $y_t \in \Re^{|\mathcal{L}'|}$ is a probability distribution over the set $\mathcal{L}' = \mathcal{L} \cup$, where $\mathcal{L}$ contains all labels in the task (*e.g.* all English characters), as well as a 'blank' label denoted by . A sequence-to-sequence mapping function $\mathcal{B}$ is defined on sequence $\boldsymbol{\pi} \in \mathcal{L}'^T$, where T is the length. $\mathcal{B}$ maps $\boldsymbol{\pi}$ onto $\mathbf{l}$ by firstly removing the repeated labels, then removing the 'blank's. For example, $\mathcal{B}$ maps "`--hh-e-l-ll-oo--`" ('–' represents 'blank') onto "`hello`". Then, the conditional probability is defined as the sum of probabilities of all $\boldsymbol{\pi}$ that are mapped by $\mathcal{B}$ onto $\mathbf{l}$:

$$p(\mathbf{l}|\mathbf{y}) = \sum_{\boldsymbol{\pi}:\mathcal{B}(\boldsymbol{\pi})=\mathbf{l}} p(\boldsymbol{\pi}|\mathbf{y}), \qquad (1)$$

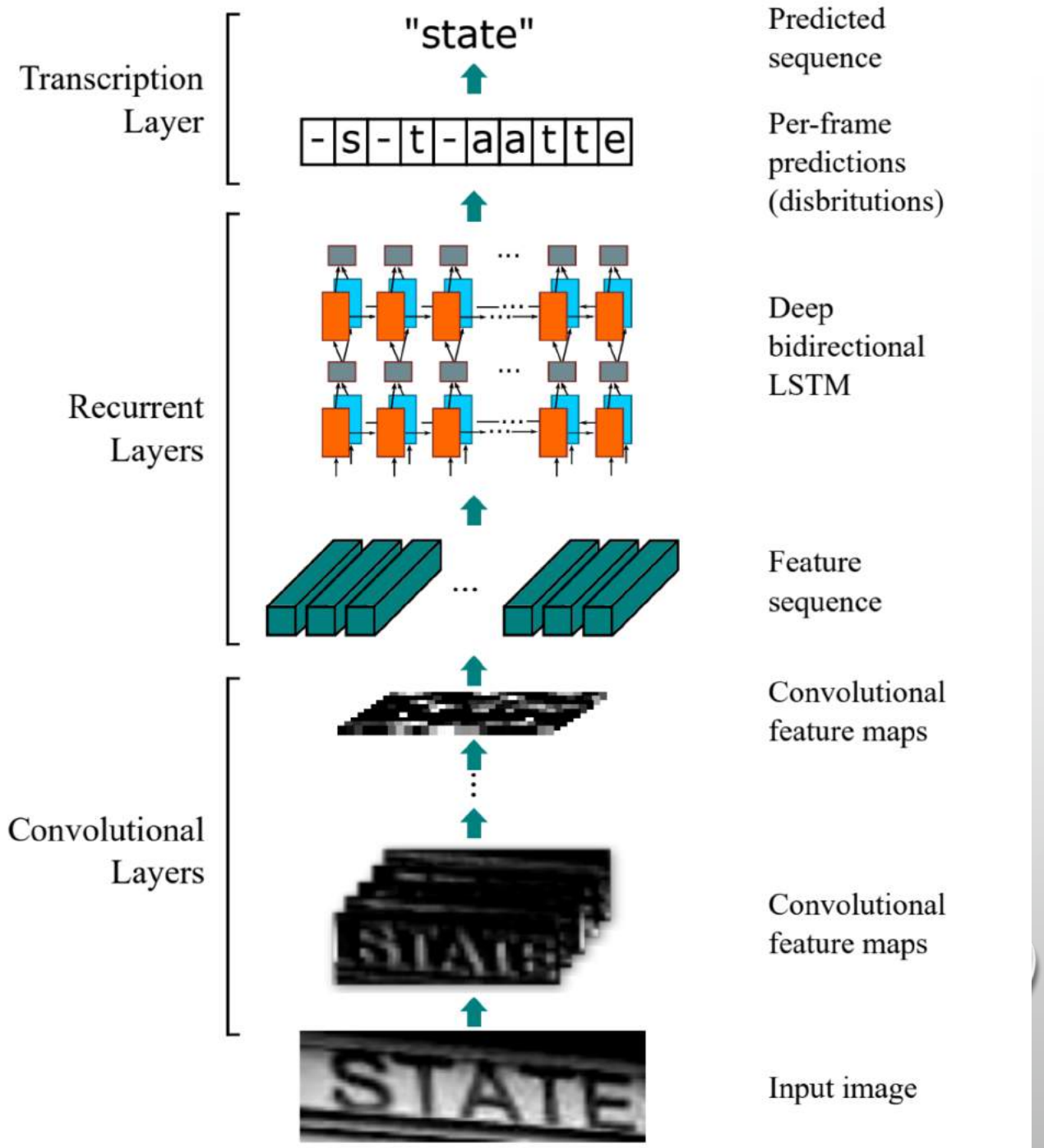
where the probability of $\boldsymbol{\pi}$ is defined as $p(\boldsymbol{\pi}|\mathbf{y}) = \prod_{t=1}^{T} y_{\pi_t}^t$, $y_{\pi_t}^t$ is the probability of having label π_t at time stamp t.



CRNN

2.3.2 Lexicon-free transcription

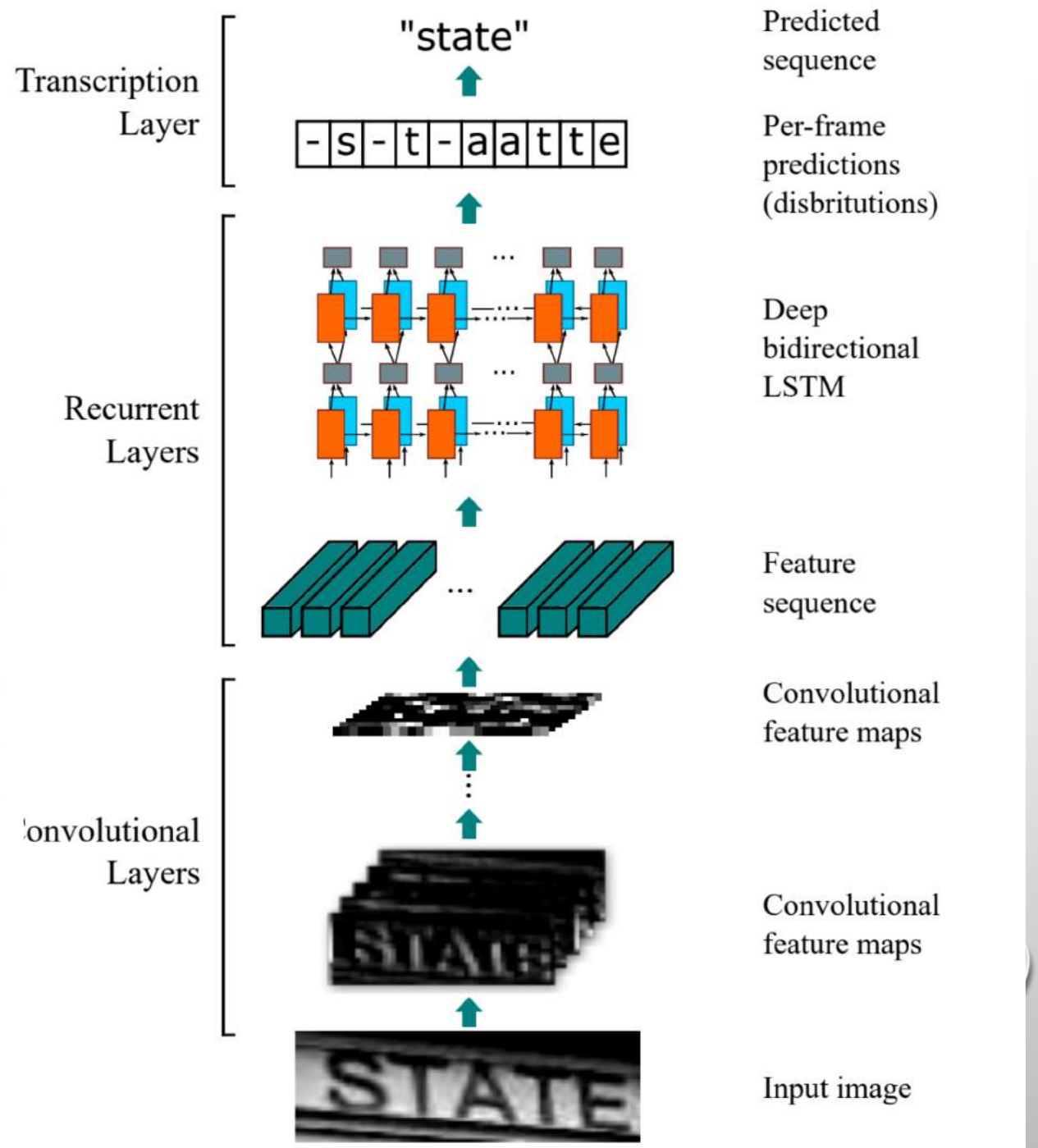
In this mode, the sequence $\mathbf{l}^*$ that has the highest probability as defined in Eq. 1 is taken as the prediction. Since there exists no tractable algorithm to precisely find the solution, we use the strategy adopted in [15]. The sequence $\mathbf{l}^*$ is approximately found by $\mathbf{l}^* \approx \mathcal{B}(\arg\max_{\boldsymbol{\pi}} p(\boldsymbol{\pi}|\mathbf{y}))$, *i.e.* taking the most probable label π_t at each time stamp t, and map the resulted sequence onto $\mathbf{l}^*$.



CRNN

2.3.3 Lexicon-based transcription

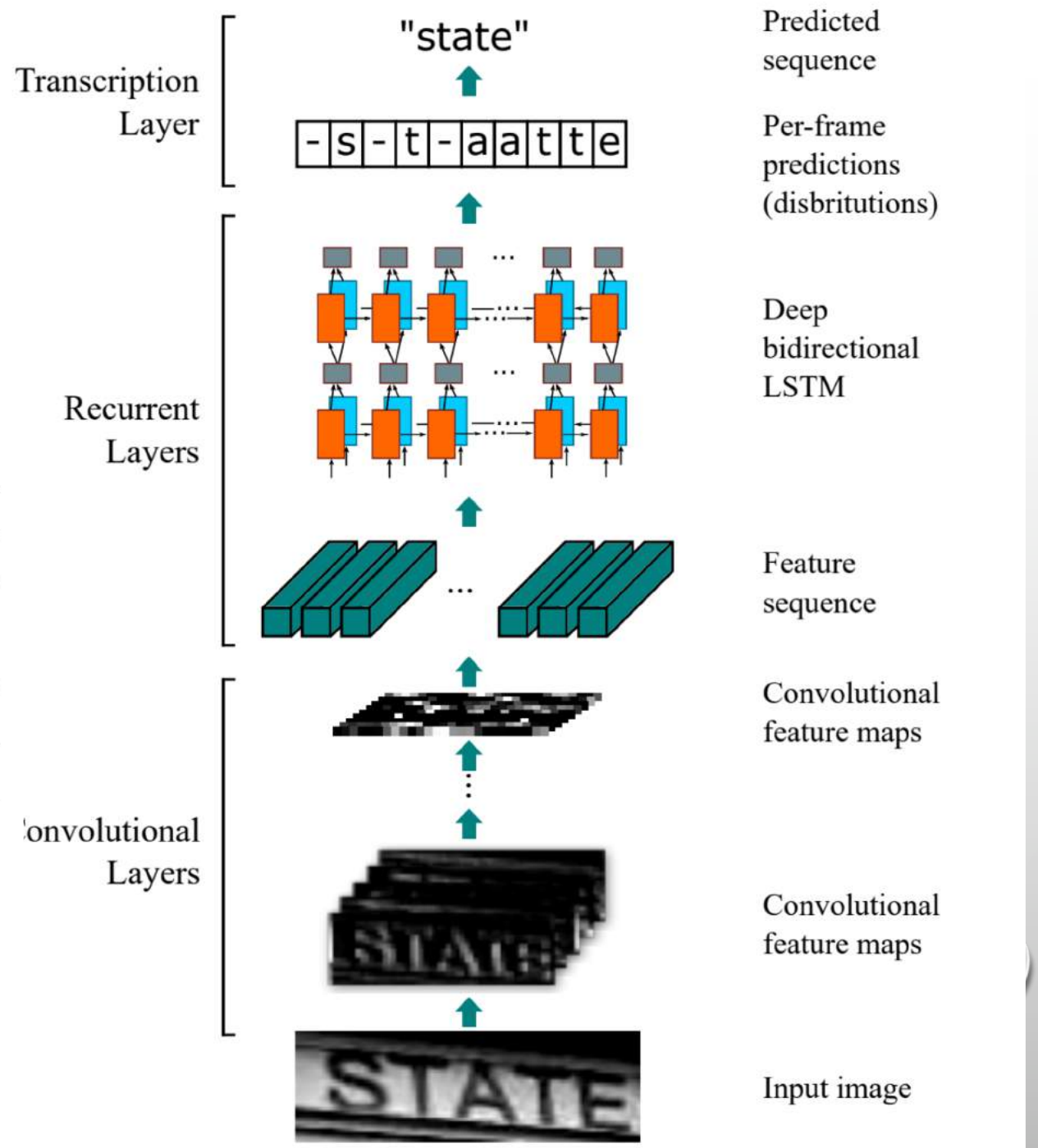
In lexicon-based mode, each test sample is associated with a lexicon $\mathcal{D}$. Basically, the label sequence is recognized by choosing the sequence in the lexicon that has highest conditional probability defined in Eq. 1, *i.e.* $\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{D}} p(\mathbf{l}|\mathbf{y})$. However, for large lexicons, *e.g.* the 50k-words Hunspell spell-checking dictionary [1], it would be very time-consuming to perform an exhaustive search over the lexicon



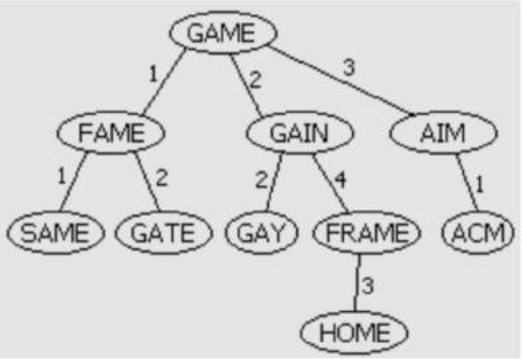
CRNN

2.3.3 Lexicon-based transcription

In lexicon-based mode, each test sample is associated with a lexicon $\mathcal{D}$. Basically, the label sequence is recognized by choosing the sequence in the lexicon that has highest conditional probability defined in Eq. 1, *i.e.* $\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{D}} p(\mathbf{l}|\mathbf{y})$. However, for large lexicons, *e.g.* the 50k-words Hunspell spell-checking dictionary [1], it would be very time-consuming to perform an exhaustive search over the lexicon



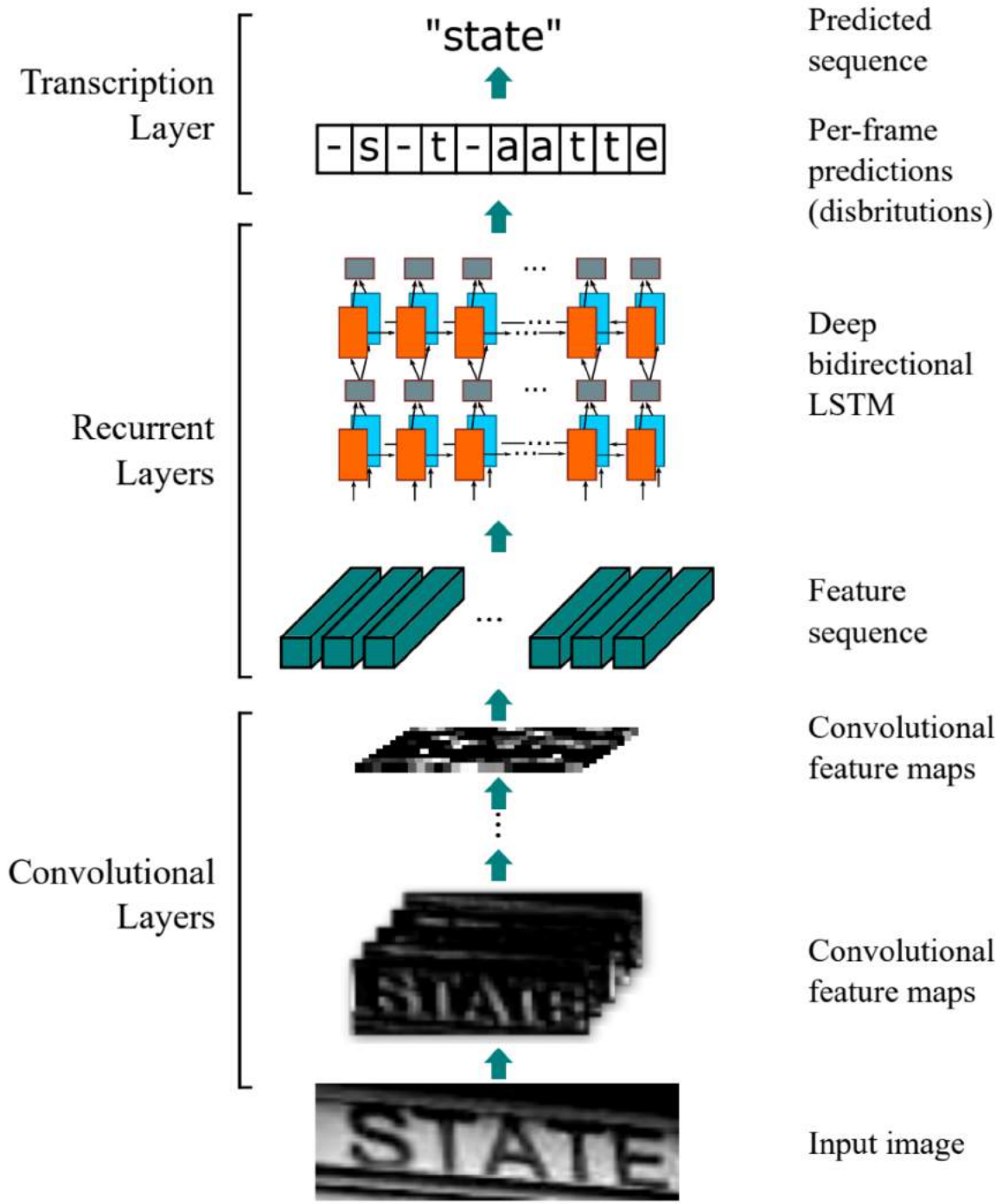
CRNN



To solve this problem, we observe that the label sequences predicted via lexicon-free transcription, described in 2.3.2, are often close to the ground-truth under the edit distance metric. This indicates that we can limit our search to the nearest-neighbor candidates $\mathcal{N}_\delta(\mathbf{l}')$, where δ is the maximal edit distance and $\mathbf{l}'$ is the sequence transcribed from $\mathbf{y}$ in lexicon-free mode:

$$\mathbf{l}^* = \arg\max_{\mathbf{l}\in\mathcal{N}_\delta(\mathbf{l}')} p(\mathbf{l}|\mathbf{y}). \quad (2)$$

The candidates $\mathcal{N}_\delta(\mathbf{l}')$ can be found efficiently with the BK-tree data structure [9], which is a metric tree specifically adapted to discrete metric spaces. The search time complexity of BK-tree is $O(\log|\mathcal{D}|)$, where $|\mathcal{D}|$ is the lexicon size.

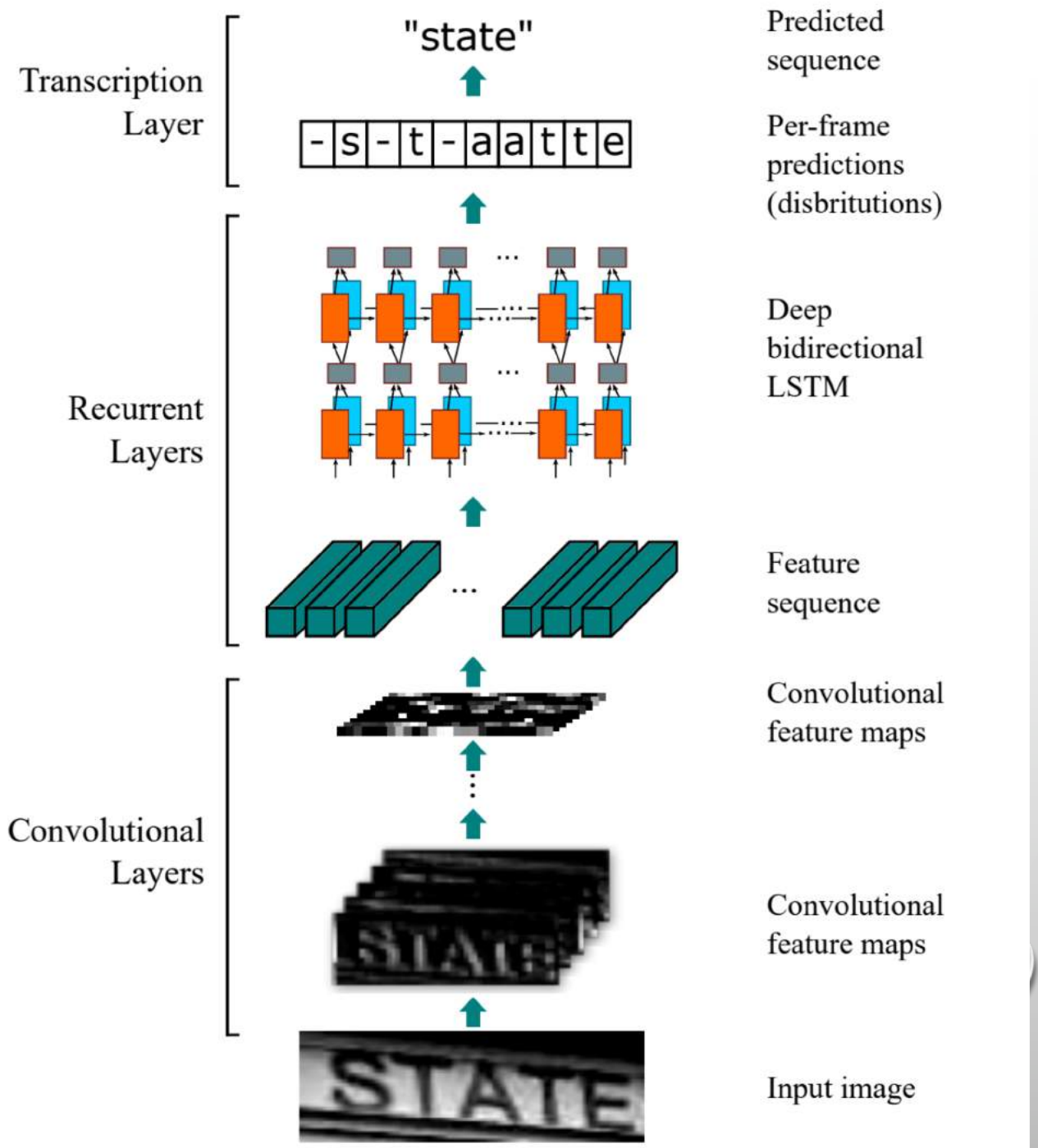


CRNN

2.4. Network Training

Denote the training dataset by $\mathcal{X} = \{I_i, \mathbf{l}_i\}_i$, where I_i is the training image and $\mathbf{l}_i$ is the ground truth label sequence. The objective is to minimize the negative log-likelihood of conditional probability of ground truth:

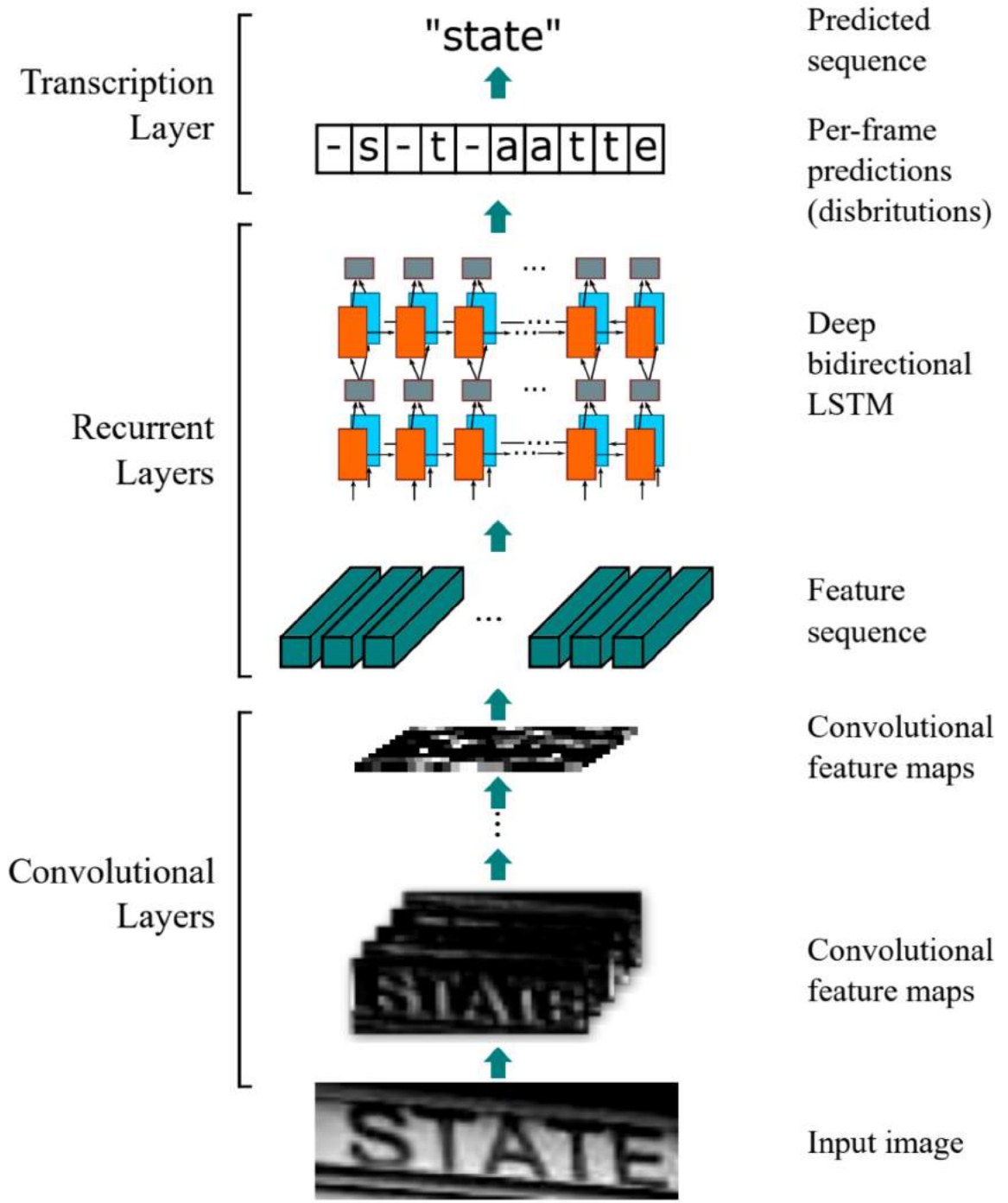
$$\mathcal{O} = -\sum_{I_i, \mathbf{l}_i \in \mathcal{X}} \log p(\mathbf{l}_i | \mathbf{y}_i), \tag{3}$$



CRNN

Table 1. Network configuration summary. The first row is the top layer. 'k', 's' and 'p' stand for kernel size, stride and padding size respectively

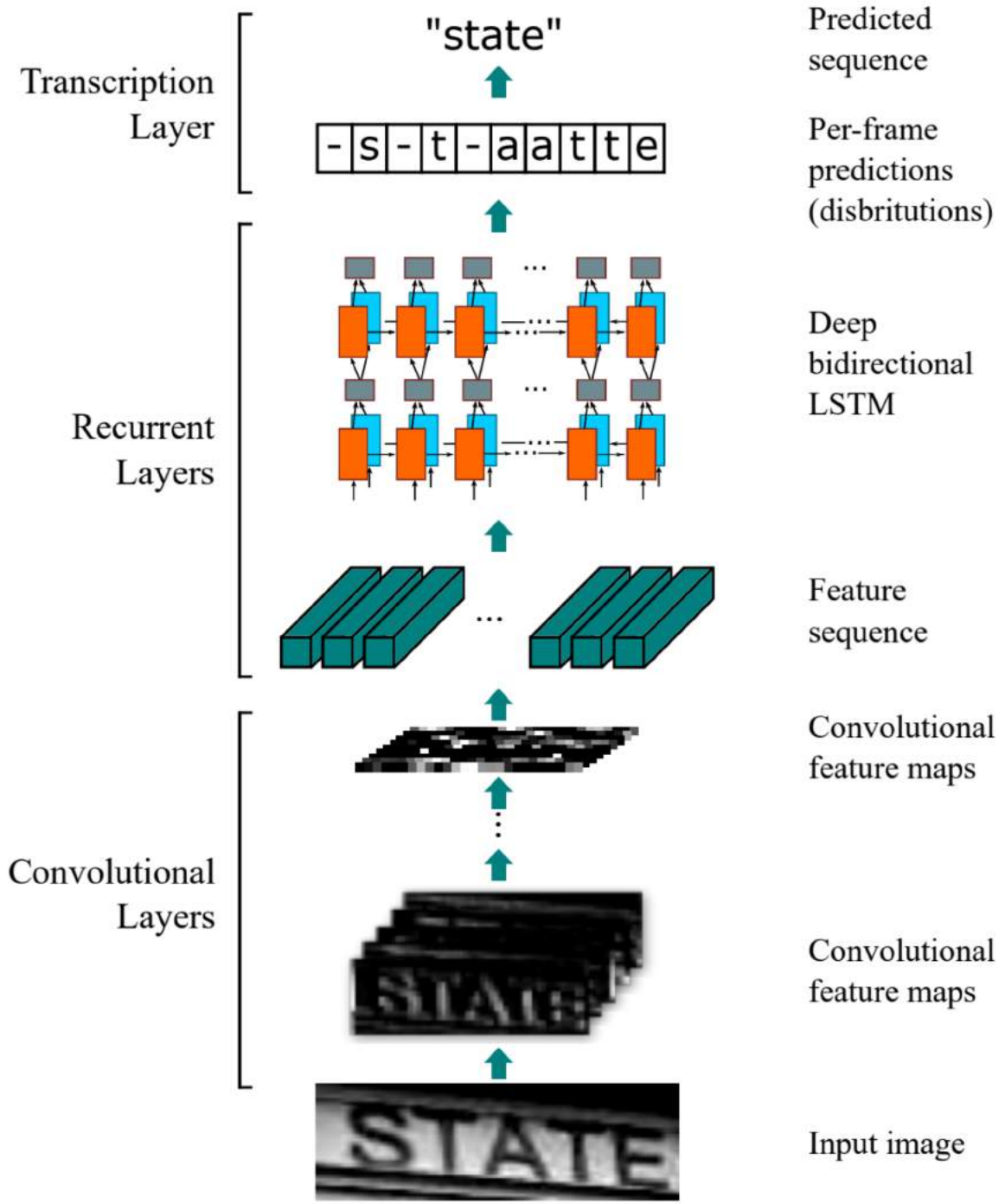
Type	Configurations
Transcription	-
Bidirectional-LSTM	#hidden units:256
Bidirectional-LSTM	#hidden units:256
Map-to-Sequence	-
Convolution	#maps:512, k:2 × 2, s:1, p:0
MaxPooling	Window:1 × 2, s:2
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
BatchNormalization	-
Convolution	#maps:512, k:3 × 3, s:1, p:1
MaxPooling	Window:1 × 2, s:2
Convolution	#maps:256, k:3 × 3, s:1, p:1
Convolution	#maps:256, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:128, k:3 × 3, s:1, p:1
MaxPooling	Window:2 × 2, s:2
Convolution	#maps:64, k:3 × 3, s:1, p:1
Input	$W \times 32$ gray-scale image



CRNN

Table 3. Comparison among various methods. Attributes for comparison include: 1) being end-to-end trainable (E2E Train); 2) using convolutional features that are directly learned from images rather than using hand-crafted ones (Conv Ftrs); 3) requiring no ground truth bounding boxes for characters during training (CharGT-Free); 4) not confined to a pre-defined dictionary (Unconstrained); 5) the model size (if an end-to-end trainable model is used), measured by the number of model parameters (Model Size, M stands for millions).

	E2E Train	Conv Ftrs	CharGT-Free	Unconstrained	Model Size
Wang *et al.* [34]	✗	✗	✗	✔	-
Mishra *et al.* [28]	✗	✗	✗	✗	-
Wang *et al.* [35]	✗	✔	✗	✔	-
Goel *et al.* [13]	✗	✗	✔	✗	-
Bissacco *et al.* [8]	✗	✗	✗	✔	-
Alsharif and Pineau [6]	✗	✔	✗	✔	-
Almazán *et al.* [5]	✗	✗	✔	✗	-
Yao *et al.* [36]	✗	✗	✗	✔	-
Rodrguez-Serrano *et al.* [30]	✗	✗	✔	✗	-
Jaderberg *et al.* [23]	✗	✔	✗	✔	-
Su and Lu [33]	✗	✗	✔	✔	-
Gordo [14]	✗	✗	✗	✗	-
Jaderberg *et al.* [22]	✔	✔	✔	✗	490M
Jaderberg *et al.* [21]	✔	✔	✔	✔	304M
CRNN	✔	✔	✔	✔	**8.3M**



Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks

Alex Graves
TU Munich, Germany
graves@in.tum.de

Jürgen Schmidhuber
IDSIA, Switzerland and TU Munich, Germany
juergen@idsia.ch

http://people.idsia.ch/~juergen/nips2009.pdf

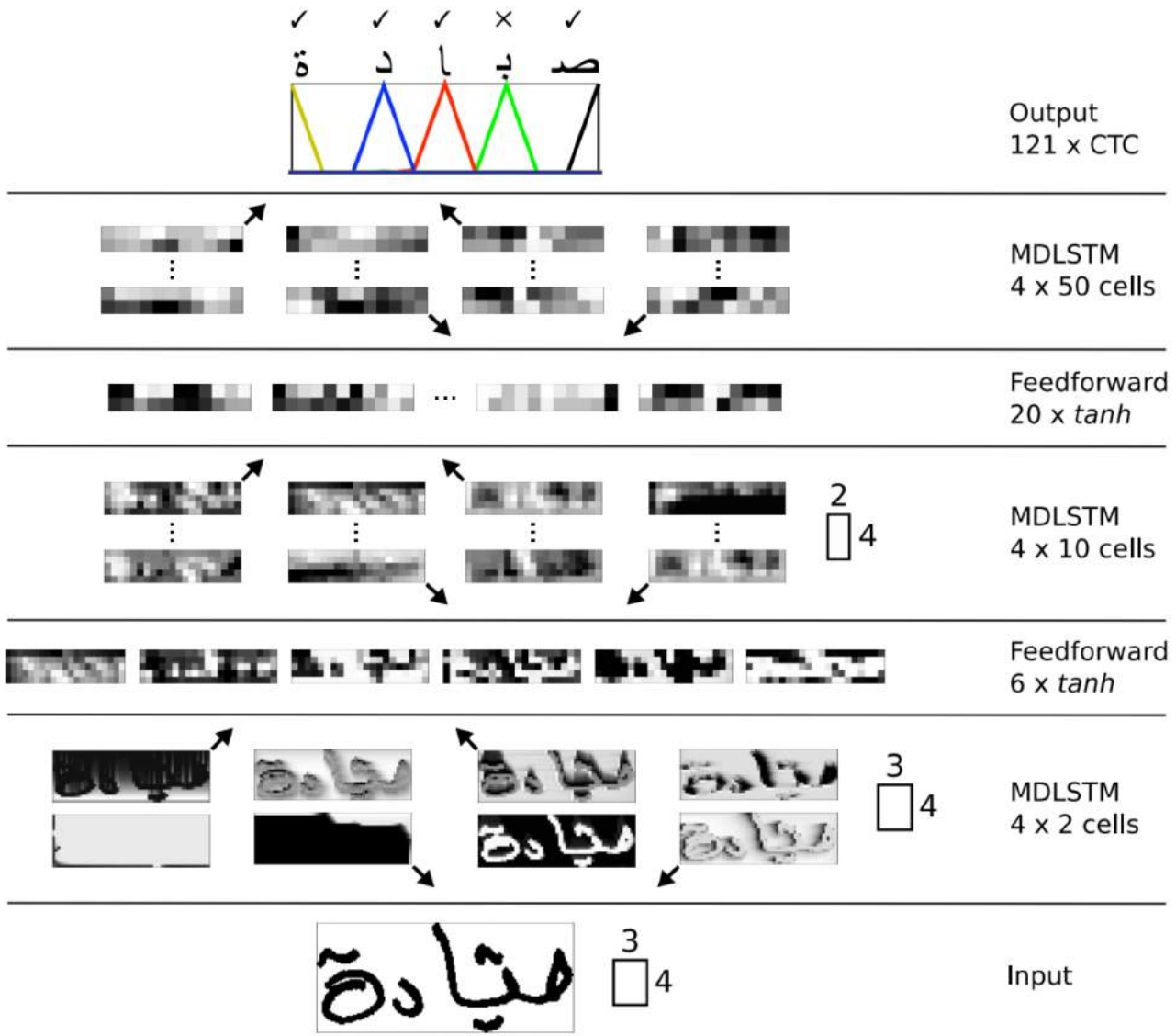


Figure 2: **The complete recognition system.** First the input image is collected into boxes 3 pixels wide and 4 pixels high which are then scanned by four MDLSTM layers. The activations of the cells in each layer are displayed separately, and the arrows in the corners indicates the scanning direction. Next the MDLSTM activations are gathered into 4 x 3 boxes and fed to a feedforward layer of *tanh* summation units. This process is repeated two more times, until the final MDLSTM activations are collapsed to a 1D sequence and transcribed by the CTC layer. In this case all characters are correctly labelled except the second last one, and the correct town name is chosen from the dictionary.

休息一下

- HTTPS://WWW.YOUTUBE.COM/WATCH?V=QPCCK1V1JO8&T=10S

第二课 序列识别

- RNN
  - LSTM
  - GRU
- CRNN
  - CTC
- RARE
  - ATTENTION

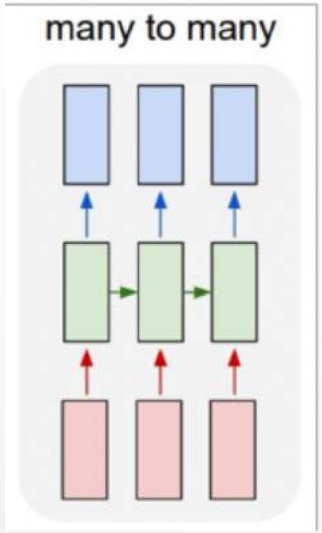
CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)

RNNs are powerful learners for sequences, but:

- Standard methods need pre-segmented training data
- Need for complex post-preprocessing

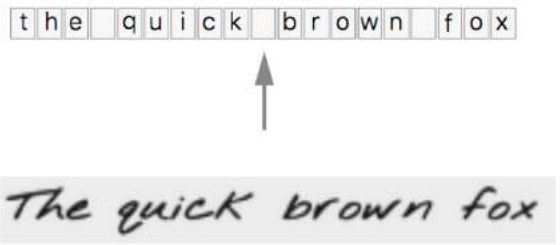
CTC solves this problem:

- Able to train RNNs using unsegmented training data
- Learns the segmentation automatically
- Provides directly usable output

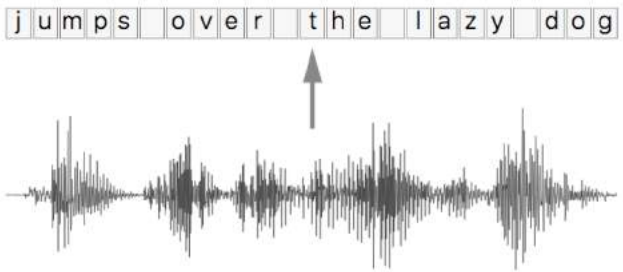


$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X).$$

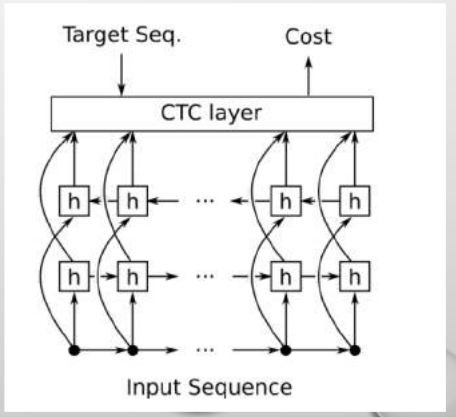
x_1	x_2	x_3	x_4	x_5	x_6	input (X)
c	c	a	a	a	t	alignment
c		a			t	output (Y)



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.

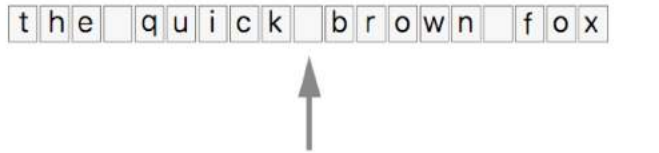


Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

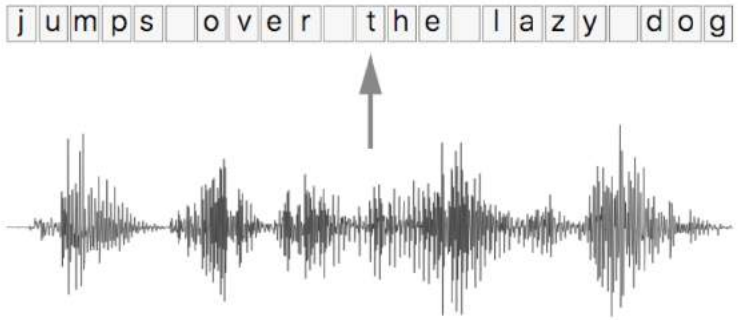


https://distill.pub/2017/ctc/

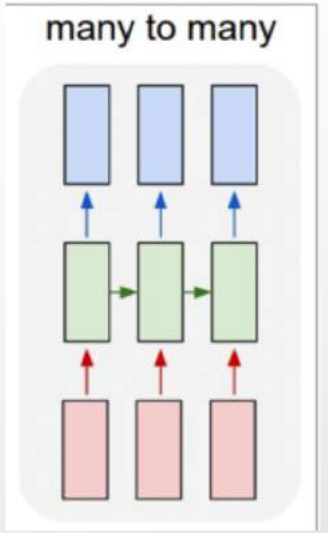
CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.



$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X).$$

x_1 x_2 x_3 x_4 x_5 x_6	input (X)
c c a a a t	alignment
c a t	output (Y)

This approach has two problems.

- Often, it doesn't make sense to force every input step to align to some output. In speech recognition, for example, the input can have stretches of silence with no corresponding output.
- We have no way to produce outputs with multiple characters in a row. Consider the alignment [h, h, e, l, l, l, o]. Collapsing repeats will produce "helo" instead of "hello".

https://distill.pub/2017/ctc/

CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X).$$

x_1 x_2 x_3 x_4 x_5 x_6	input (X)
c c a a a t	alignment
c a t	output (Y)

The alignments allowed by CTC are the same length as the input. We allow any alignment which maps to Y after merging repeats and removing ϵ tokens:

h h e ϵ ϵ l l l ϵ l l o

First, merge repeat characters.

h e ϵ l ϵ l o

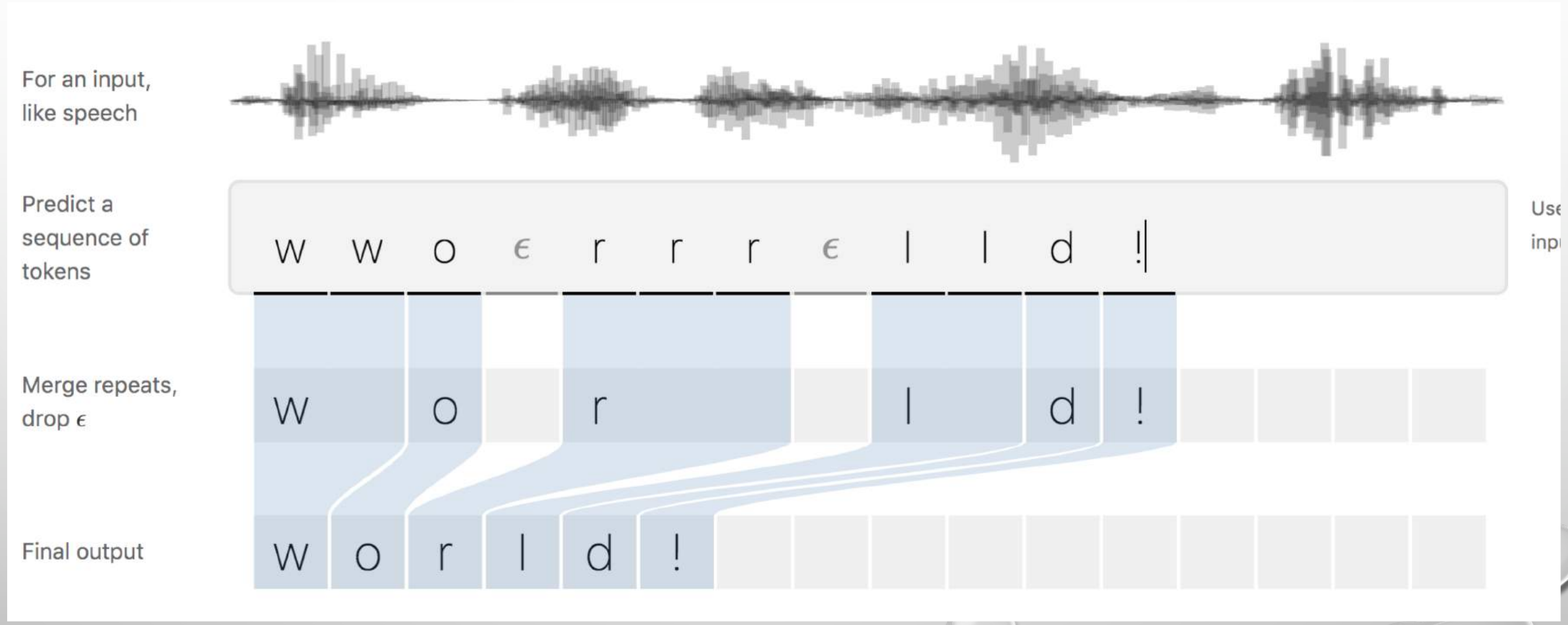
Then, remove any ϵ tokens.

h e l l o

The remaining characters are the output.

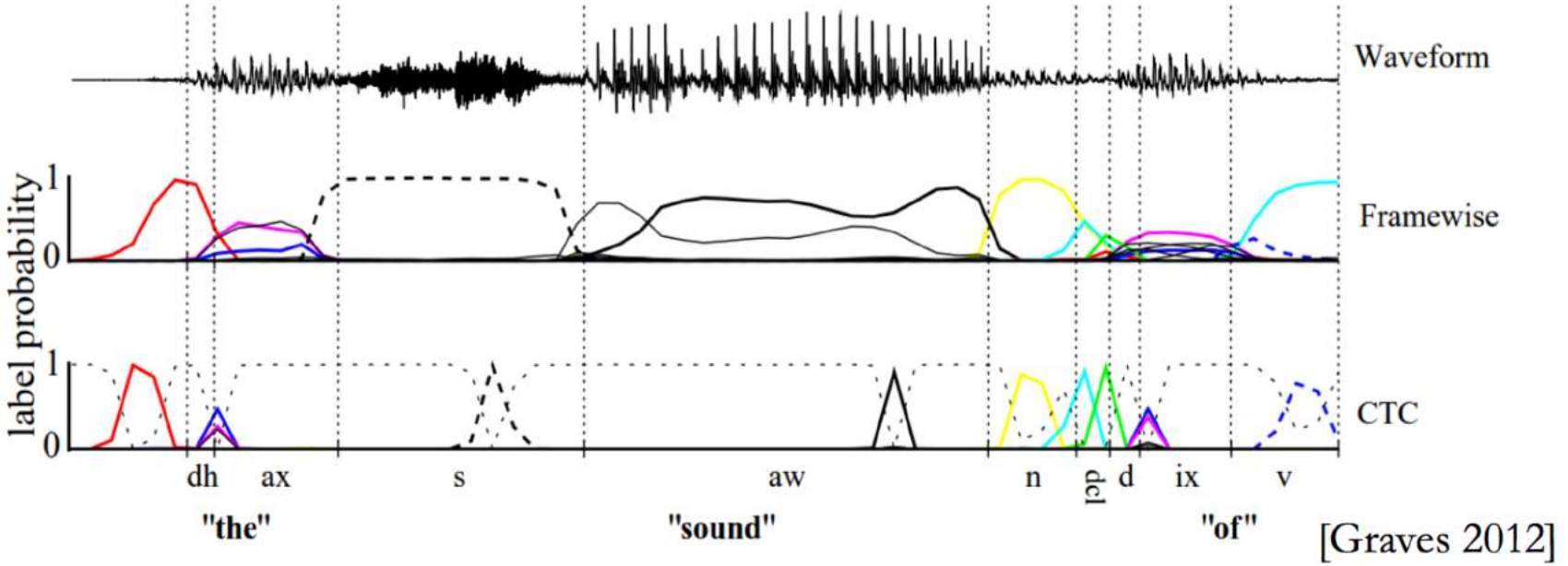
h e l l o

CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)



https://distill.pub/2017/ctc/

CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)



[Graves 2012]

- Sparse representations (spikes) appear
- Any modeling unit can be used: phone, syllable, word

CTC (CONNECTIONIST TEMPORAL CLASSIFICATION)

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X).$$

x_1 x_2 x_3 x_4 x_5 x_6	input (X)
c c a a a t	alignment
c a t	output (Y)

Let's go back to the output [c, a, t] with an input of length six. Here are a few more examples of valid and invalid alignments.

Valid Alignments	Invalid Alignments	
ϵ c c ϵ a t	c ϵ c ϵ a t	corresponds to Y = [c, c, a, t]
c c a a t t	c c a a t _	has length 5
c a ϵ ϵ ϵ t	c ϵ ϵ ϵ t t	missing the 'a'

CTC

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

Alex Graves[1] ALEX@IDSIA.CH
Santiago Fernández[1] SANTIAGO@IDSIA.CH
Faustino Gomez[1] TINO@IDSIA.CH
Jürgen Schmidhuber[1,2] JUERGEN@IDSIA.CH

[1] Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno-Lugano, Switzerland
[2] Technische Universität München (TUM), Boltzmannstr. 3, 85748 Garching, Munich, Germany

$$Y^* = \underset{Y}{\operatorname{argmax}}\ p(Y \mid X).$$

x_1	x_2	x_3	x_4	x_5	x_6	input (X)
c	c	a	a	a	t	alignment
c		a			t	output (Y)

Valid Alignments

ϵ	c	c	ϵ	a	t
c	c	a	a	t	t
c	a	ϵ	ϵ	ϵ	t

2.1. Label Error Rate

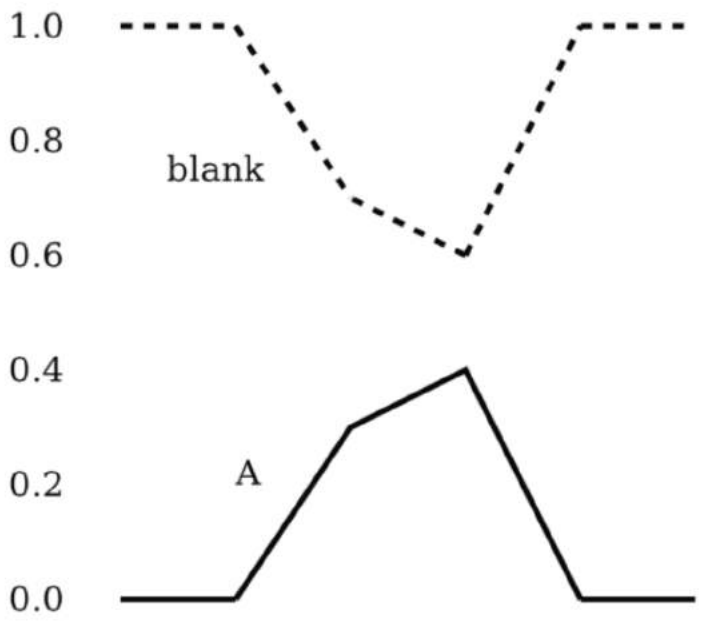
In this paper, we are interested in the following error measure: given a test set $S' \subset \mathcal{D}_{\mathcal{X}\times\mathcal{Z}}$ disjoint from S, define the *label error rate* (LER) of a temporal classifier h as the mean normalised edit distance between its classifications and the targets on S', i.e.

$$LER(h, S') = \frac{1}{|S'|} \sum_{(\mathbf{x},\mathbf{z})\in S'} \frac{ED(h(\mathbf{x}), \mathbf{z})}{|\mathbf{z}|} \qquad (1)$$

where $ED(\mathbf{p}, \mathbf{q})$ is the edit distance between two sequences $\mathbf{p}$ and $\mathbf{q}$ — i.e. the minimum number of insertions, substitutions and deletions required to change $\mathbf{p}$ into $\mathbf{q}$.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.75.6306&rep=rep1&type=pdf

Problem with Best Path Decoding



$$p(l=\text{blank}) = p(- -) = 0.7*0.6 = 0.42$$

$$p(l=A) = p(AA)+p(A-)+p(-A) = 0.3*0.4 + 0.3*0.6 + 0.7*0.4 = 0.58$$

Fig. 7.5 Problem with best path decoding. The single most probable path contains no labels, and best path decoding therefore outputs the labelling 'blank'. However the combined probabilities of the paths corresponding to the labelling 'A' is greater.

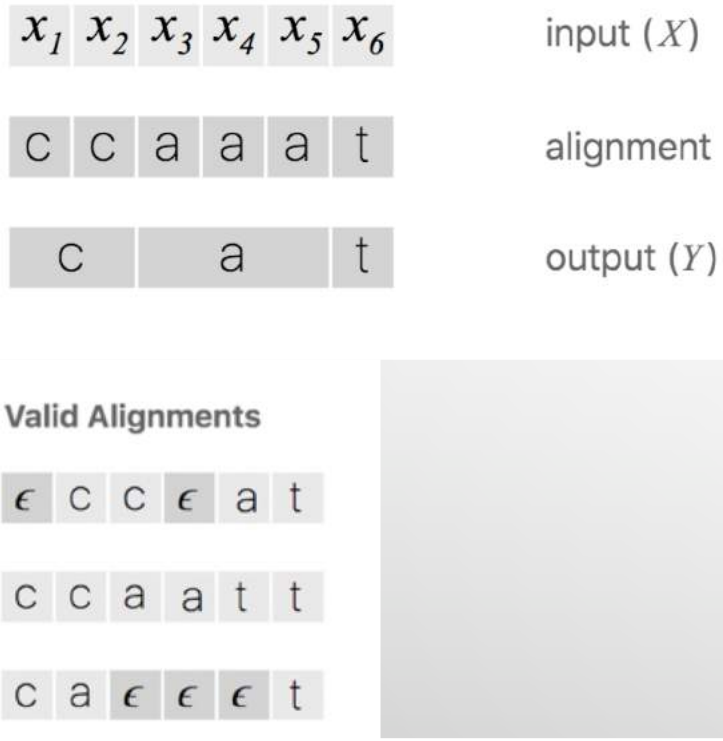
[Graves 2012]

CTC

More formally, for an input sequence $\mathbf{x}$ of length T, define a recurrent neural network with m inputs, n outputs and weight vector w as a continuous map $\mathcal{N}_w : (\mathbb{R}^m)^T \mapsto (\mathbb{R}^n)^T$. Let $\mathbf{y} = \mathcal{N}_w(\mathbf{x})$ be the sequence of network outputs, and denote by y_k^t the activation of output unit k at time t. Then y_k^t is interpreted as the probability of observing label k at time t, which defines a distribution over the set L'^T of length T sequences over the alphabet $L' = L \cup \{blank\}$:

$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \ \forall \pi \in L'^T. \qquad (2)$$

CTC



Given the above formulation, the output of the classifier should be the most probable labelling for the input sequence:

$$h(\mathbf{x}) = \arg\max_{\mathbf{l} \in L^{\leq T}} p(\mathbf{l}|\mathbf{x}).$$

The next step is to define a many-to-one map $\mathcal{B} : L'^T \mapsto L^{\leq T}$, where $L^{\leq T}$ is the set of possible labellings (i.e. the set of sequences of length less than or equal to T over the original label alphabet L). We do this by simply removing all blanks and repeated labels from the paths (e.g. $\mathcal{B}(a - ab-) = \mathcal{B}(-aa - -abb) = aab$). Intuitively, this corresponds to outputting a new label when the network switches from predicting no label to predicting a label, or from predicting one label to another (c.f. the CTC outputs in figure 1). Finally, we use $\mathcal{B}$ to define the conditional probability of a given labelling $\mathbf{l} \in L^{\leq T}$ as the sum of the probabilities of all the paths corresponding to it:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \qquad (3)$$

TRAINING
THE CTC FORWARD-BACKWARD ALGORITHM

CTC is a dynamic programming algorithm

For some sequence $\mathbf{q}$ of length r, denote by $\mathbf{q}_{1:p}$ and $\mathbf{q}_{r-p:r}$ its first and last p symbols respectively. Then for a labelling $\mathbf{l}$, define the forward variable $\alpha_t(s)$ to be the total probability of $\mathbf{l}_{1:s}$ at time t. i.e.

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}. \tag{5}$$

TRAINING
THE CTC FORWARD-BACKWARD ALGORITHM

To allow for blanks in the output paths, we consider a modified label sequence $\mathbf{l}'$, with blanks added to the beginning and the end and inserted between every pair of labels. The length of $\mathbf{l}'$ is therefore $2|\mathbf{l}| + 1$. In calculating the probabilities of prefixes of $\mathbf{l}'$ we allow all transitions between blank and non-blank labels, and also those between any pair of *distinct* non-blank labels. We allow all prefixes to start with either a blank (b) or the first symbol in $\mathbf{l}$ (l_1).

This gives us the following rules for initialisation

$$\alpha_1(1) = y_b^1$$
$$\alpha_1(2) = y_{\mathbf{l}_1}^1$$
$$\alpha_1(s) = 0, \ \forall s > 2$$

For some sequence $\mathbf{q}$ of length r, denote by $\mathbf{q}_{1:p}$ and $\mathbf{q}_{r-p:r}$ its first and last p symbols respectively. Then for a labelling $\mathbf{l}$, define the forward variable $\alpha_t(s)$ to be the total probability of $\mathbf{l}_{1:s}$ at time t. i.e.

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y_{\pi_{t'}}^{t'}. \qquad (5)$$

TRAINING
THE CTC FORWARD-BACKWARD ALGORITHM

output unit k at time t. Then y_k^t is interpreted as the probability of observing label k at time t, which defines

For some sequence $\mathbf{q}$ of length r, denote by $\mathbf{q}_{1:p}$ and $\mathbf{q}_{r-p:r}$ its first and last p symbols respectively. Then for a labelling $\mathbf{l}$, define the forward variable $\alpha_t(s)$ to be the total probability of $\mathbf{l}_{1:s}$ at time t. i.e.

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y_{\pi_{t'}}^{t'}. \tag{5}$$

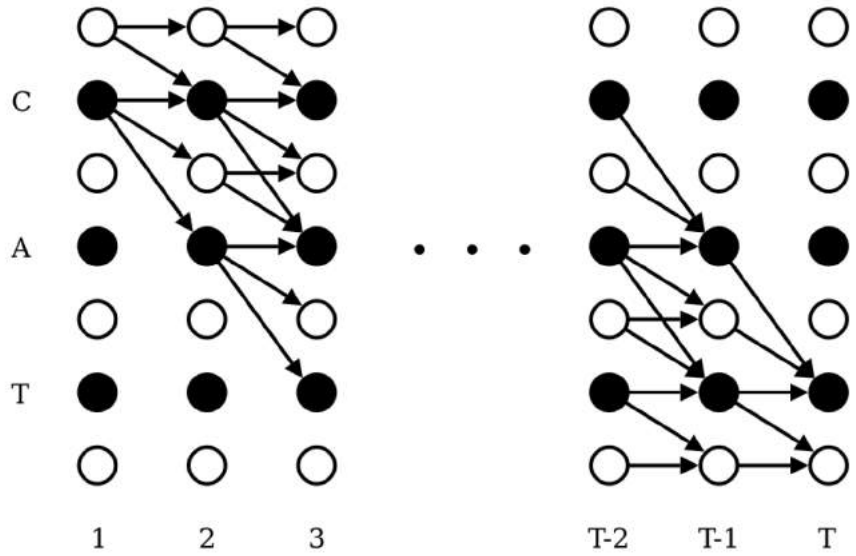
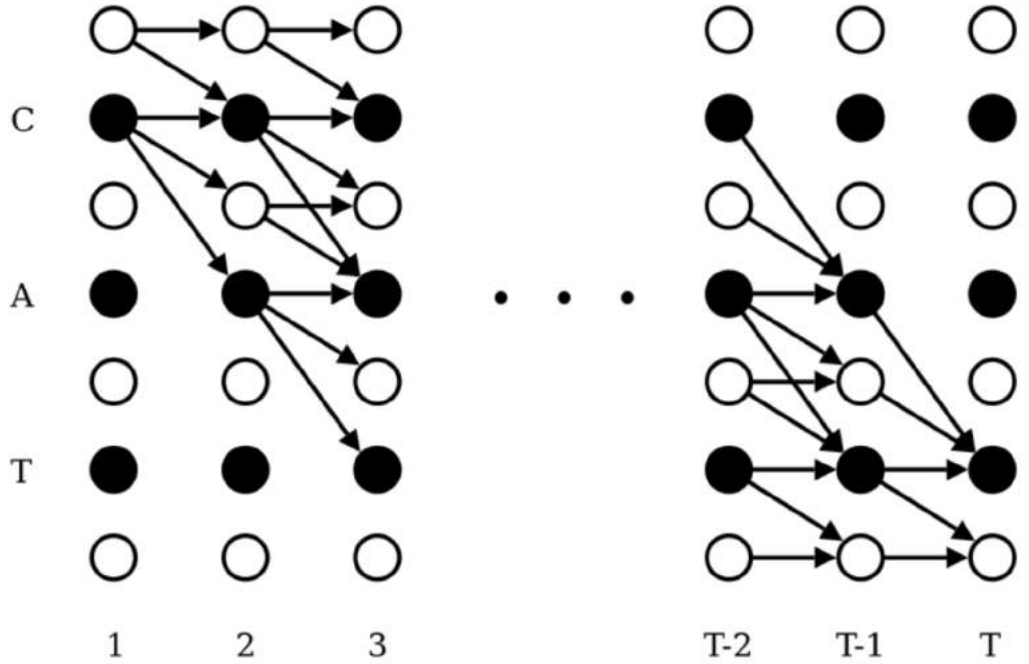


Figure 3. **illustration of the forward backward algorithm applied to the labelling 'CAT'.** Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

TRAINING



This gives us the following rules for initialisation

$$\alpha_1(1) = y^1_b$$
$$\alpha_1(2) = y^1_{\mathbf{l}_1}$$
$$\alpha_1(s) = 0, \ \forall s > 2$$

and recursion

$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{l'_s} & \text{if } l'_s = b \text{ or } l'_{s-2} = l'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{l'_s} & \text{otherwise} \end{cases} \quad (6)$$

where

$$\bar{\alpha}_t(s) \overset{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1). \quad (7)$$

output unit k at time t. Then y^t_k is interpreted as the probability of observing label k at time t, which defines

For some sequence $\mathbf{q}$ of length r, denote by $\mathbf{q}_{1:p}$ and $\mathbf{q}_{r-p:r}$ its first and last p symbols respectively. Then for a labelling $\mathbf{l}$, define the forward variable $\alpha_t(s)$ to be the total probability of $\mathbf{l}_{1:s}$ at time t. i.e.

$$\alpha_t(s) \overset{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}. \quad (5)$$

TRAINING
THE CTC FORWARD-BACKWARD ALGORITHM

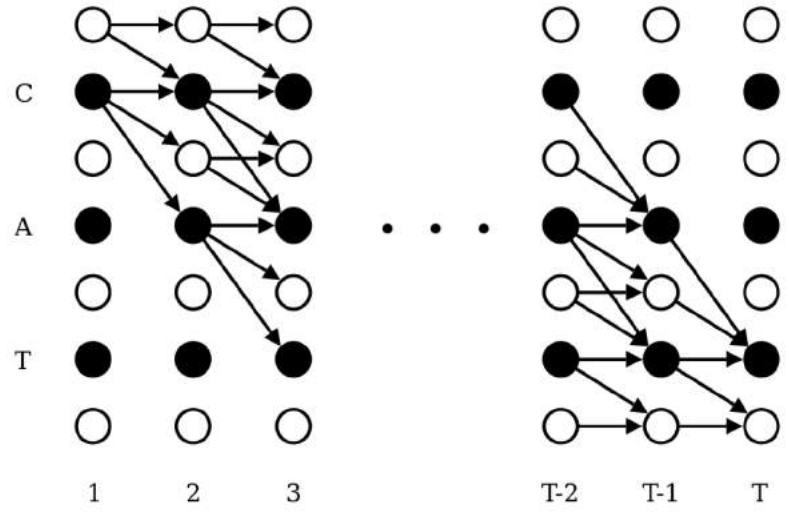


Figure 3. **illustration of the forward backward algorithm applied to the labelling 'CAT'**. Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

The probability of $\mathbf{l}$ is then the sum of the total probabilities of $\mathbf{l}'$ with and without the final blank at time T.

$$p(\mathbf{l}|\mathbf{x}) = \alpha_T(|\mathbf{l}'|) + \alpha_T(|\mathbf{l}'| - 1). \qquad (8)$$

TRAINING
THE CTC FORWARD-BACKWARD ALGORITHM

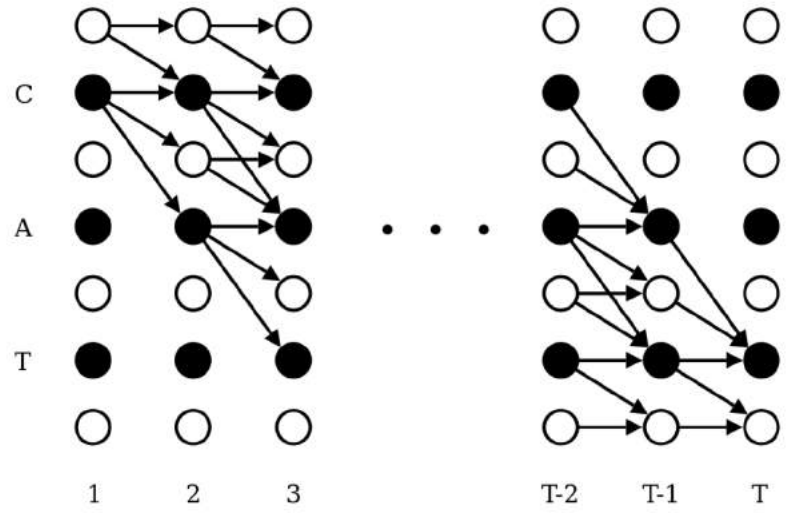


Figure 3. **illustration of the forward backward algorithm applied to the labelling 'CAT'**. Black circles represent labels, and white circles represent blanks. Arrows signify allowed transitions. Forward variables are updated in the direction of the arrows, and backward variables are updated against them.

Similarly, the backward variables $\beta_t(s)$ are defined as the total probability of $\mathbf{l}_{s:|\mathbf{l}|}$ at time t.

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}} \tag{9}$$

$$\begin{aligned} \beta_T(|\mathbf{l}'|) &= y^T_b \\ \beta_T(|\mathbf{l}'|-1) &= y^T_{\mathbf{l}_{|\mathbf{l}|}} \\ \beta_T(s) &= 0, \ \forall s < |\mathbf{l}'|-1 \end{aligned}$$

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y^t_{l'_s} & \text{if } l'_s = b \text{ or } l'_{s+2} = l'_s \\ \big(\bar{\beta}_t(s) + \beta_{t+1}(s+2)\big) y^t_{l'_s} & \text{otherwise} \end{cases} \tag{10}$$

where

$$\bar{\beta}_t(s) \stackrel{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1). \tag{11}$$

TRAINING

4.2. Maximum Likelihood Training

The aim of maximum likelihood training is to simultaneously maximise the log probabilities of all the correct classifications in the training set. In our case, this means minimising the following objective function:

$$O^{ML}(S, \mathcal{N}_w) = -\sum_{(\mathbf{x},\mathbf{z})\in S} ln\big(p(\mathbf{z}|\mathbf{x})\big) \qquad (12)$$

To train the network with gradient descent, we need to differentiate (12) with respect to the network outputs. Since the training examples are independent we can consider them separately:

$$\frac{\partial O^{ML}(\{(\mathbf{x},\mathbf{z})\}, \mathcal{N}_w)}{\partial y_k^t} = -\frac{\partial ln(p(\mathbf{z}|\mathbf{x}))}{\partial y_k^t} \qquad (13)$$

We now show how the algorithm of section 4.1 can be used to calculate (13).

TRAINING

The key point is that, for a labelling $\mathbf{l}$, the product of the forward and backward variables at a given s and t is the probability of all the paths corresponding to $\mathbf{l}$ that go through the symbol s at time t. More precisely, from (5) and (9) we have:

$$\alpha_t(s)\beta_t(s) = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = l_s}} y^t_{l_s} \prod_{t=1}^{T} y^t_{\pi_t}.$$

for a labelling $\mathbf{l}$, define the forward variable $\alpha_t(s)$ to be the total probability of $\mathbf{l}_{1:s}$ at time t. i.e.

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t}) = \mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}. \tag{5}$$

Similarly, the backward variables $\beta_t(s)$ are defined as the total probability of $\mathbf{l}_{s:|\mathbf{l}|}$ at time t.

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}} \tag{9}$$

Rearranging and substituting in from (2) gives

$$\frac{\alpha_t(s)\beta_t(s)}{y^t_{l_s}} = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = l_s}} p(\pi|\mathbf{x}).$$

TRAINING

Rearranging and substituting in from (2) gives

$$\frac{\alpha_t(s)\beta_t(s)}{y^t_{l_s}} = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = l_s}} p(\pi|\mathbf{x}).$$

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \tag{3}$$

From (3) we can see that this is the portion of the total probability $p(\mathbf{l}|\mathbf{x})$ due to those paths going through l_s at time t. We can therefore sum over all s and t to get:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{t=1}^{T} \sum_{s=1}^{|\mathbf{l}|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{l_s}}. \tag{14}$$

TRAINING

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x}). \tag{3}$$

From (3) we can see that this is the portion of the total probability $p(\mathbf{l}|\mathbf{x})$ due to those paths going through l_s at time t. We can therefore sum over all s and t to get:

$$p(\mathbf{l}|\mathbf{x}) = \sum_{t=1}^{T} \sum_{s=1}^{|\mathbf{l}|} \frac{\alpha_t(s)\beta_t(s)}{y_{l_s}^t}. \tag{14}$$

Rearranging and substituting in from (2) gives

$$\frac{\alpha_t(s)\beta_t(s)}{y_{l_s}^t} = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = l_s}} p(\pi|\mathbf{x}).$$

Because the network outputs are conditionally independent (section 3.1), we need only consider the paths going through label k at time t to get the partial derivatives of $p(\mathbf{l}|\mathbf{x})$ with respect to y_k^t. Noting that the same label may be repeated several times in a single labelling $\mathbf{l}$, we define the set of positions where label k occurs as $lab(\mathbf{l}, k) = \{s : l_s = k\}$, which may be empty. We then differentiate (14) to get:

$$\frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y_k^t} = -\frac{1}{{y_k^t}^2} \sum_{s \in lab(\mathbf{l},k)} \alpha_t(s)\beta_t(s). \tag{15}$$

TRAINING

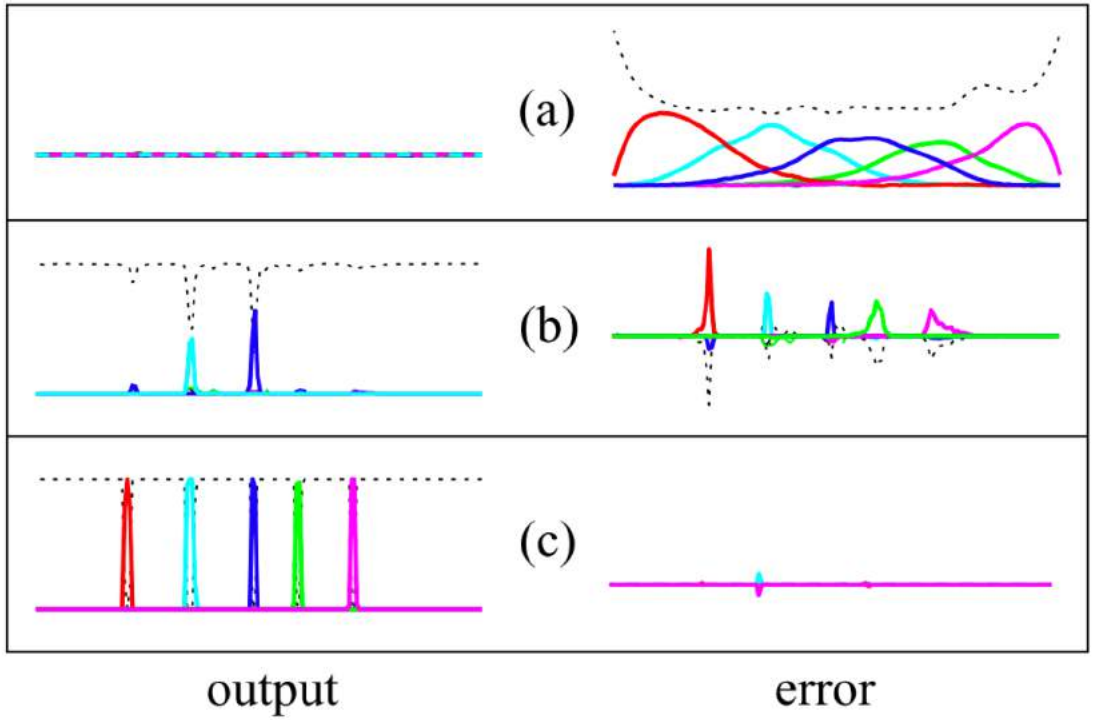
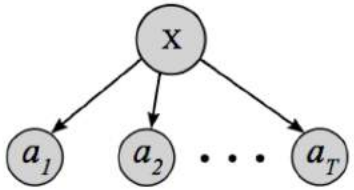


Figure 4. **Evolution of the CTC Error Signal During Training**. The left column shows the output activations for the same sequence at various stages of training (the dashed line is the 'blank' unit); the right column shows the corresponding error signals. Errors above the horizontal axis act to increase the corresponding output activation and those below act to decrease it. (a) Initially the network has small random weights, and the error is determined by the target sequence only. (b) The network begins to make predictions and the error localises around them. (c) The network strongly predicts the correct labelling and the error virtually disappears.

Conditional Independence

One of the most commonly cited shortcomings of CTC is the conditional independence assumption it makes.

The model assumes that every output is conditionally independent of the other outputs given the input. This is a bad assumption for many sequence to sequence problems.



Graphical model for CTC.

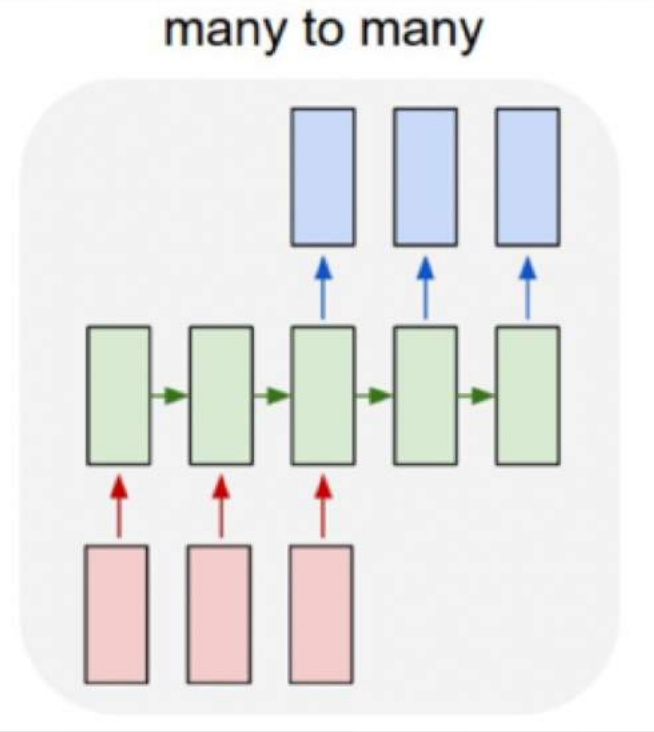
Say we had an audio clip of someone saying "triple A". [1] Another valid transcription could be "AAA". If the first letter of the predicted transcription is 'A', then the next letter should be 'A' with high probability and 'r' with low probability. The conditional independence assumption does not allow for this.

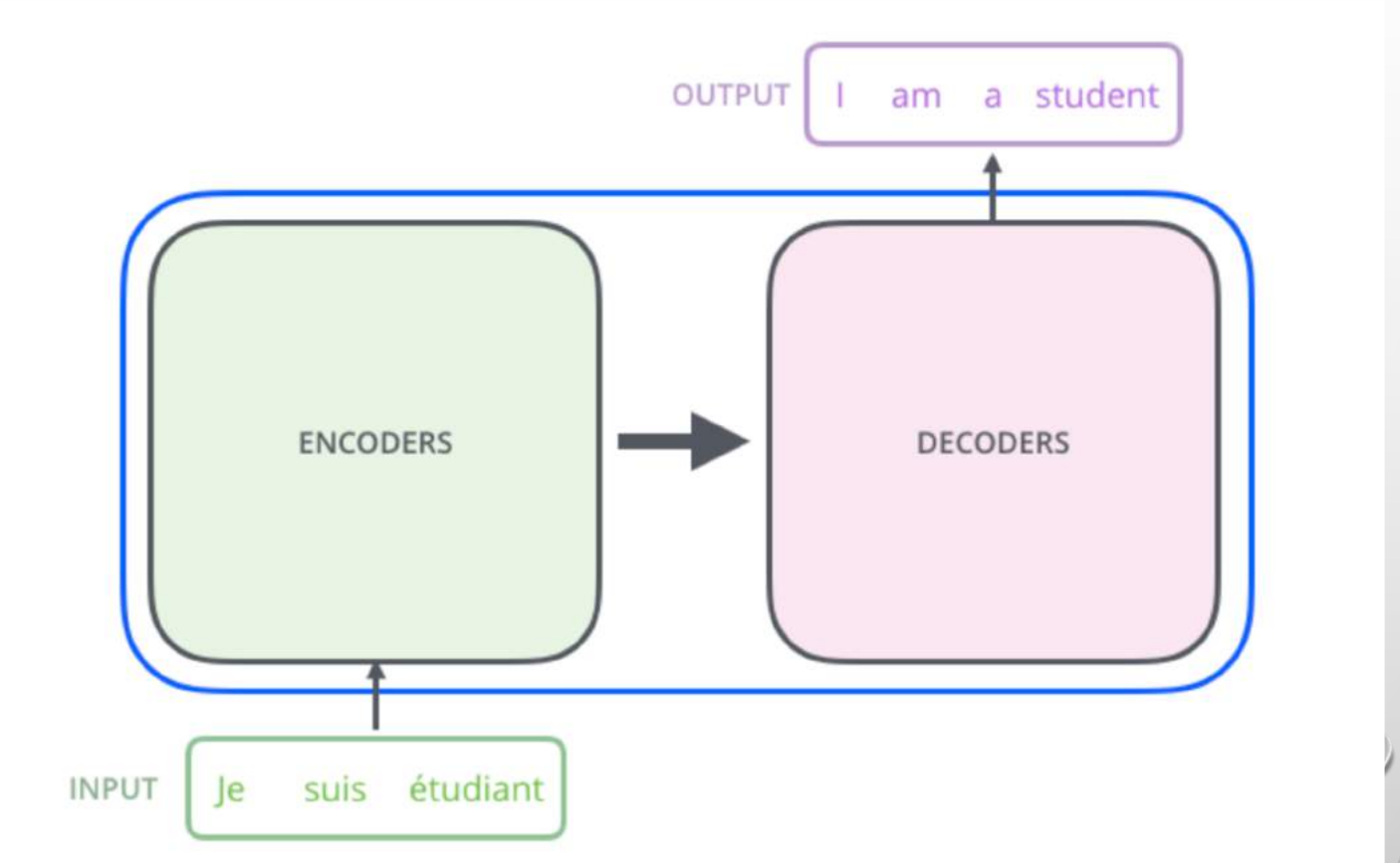
If we predict an 'A' as the first letter then the suffix 'AA' should get much more probability than 'riple A'. If we predict 't' first, the opposite should be true.

第二课 序列识别

- RNN
  - LSTM
  - GRU
- CRNN
  - CTC
- RARE
  - ATTENTION

ATTENTION





http://jalammar.github.io/illustrated-transformer/
https://arxiv.org/pdf/1706.03762.pdf

TRANSFORMER

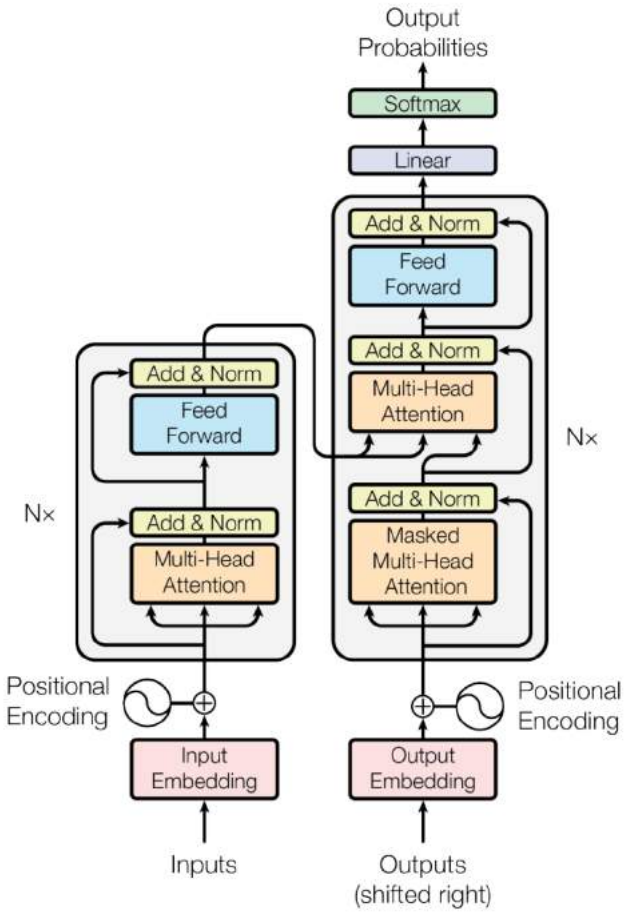
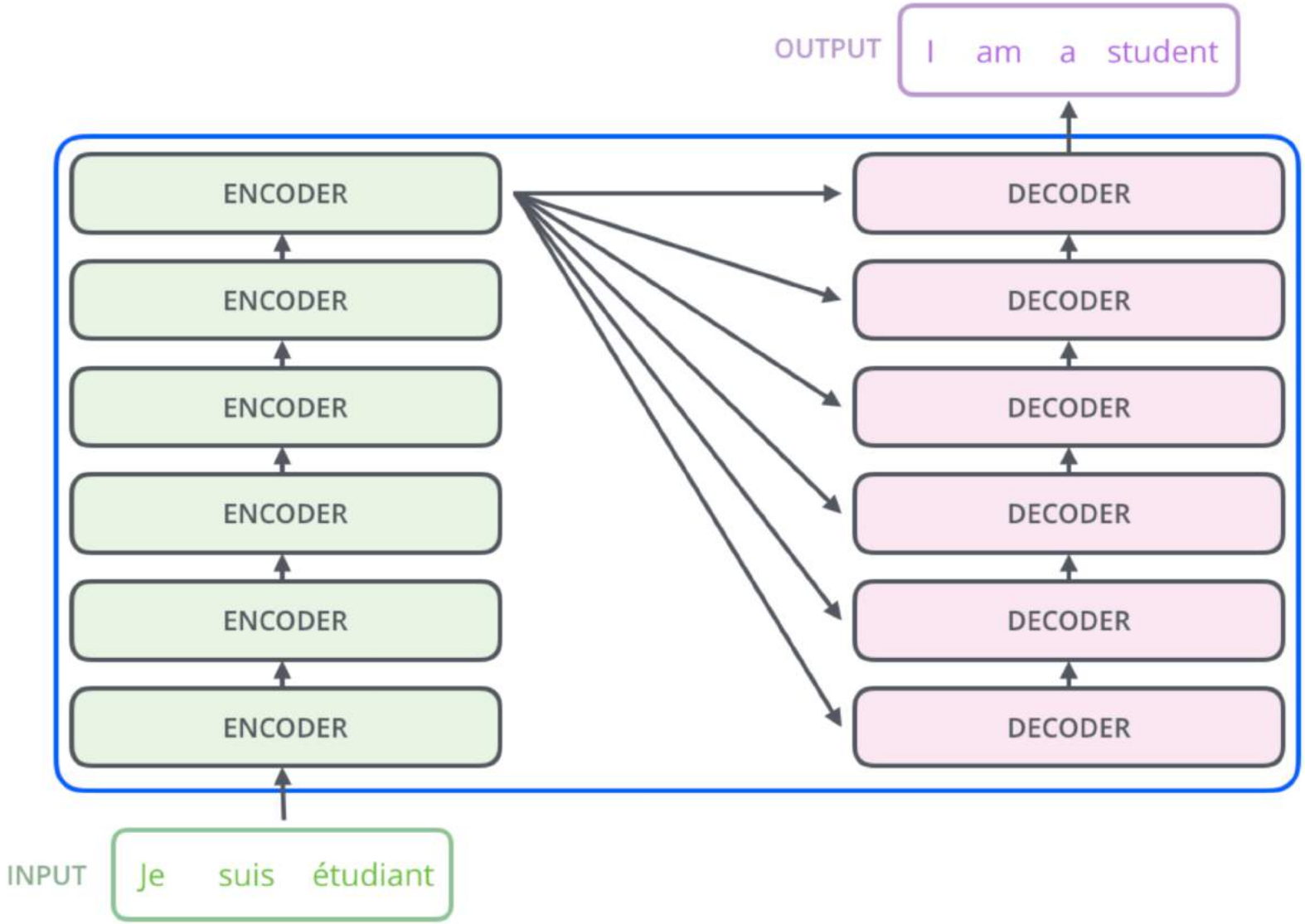
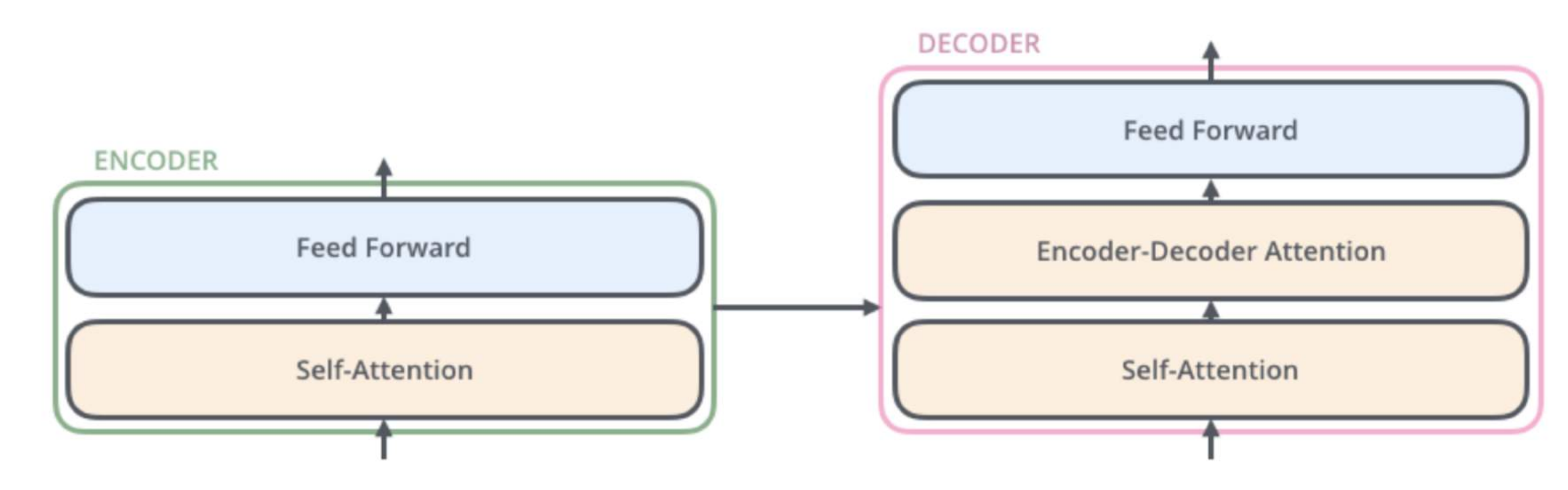


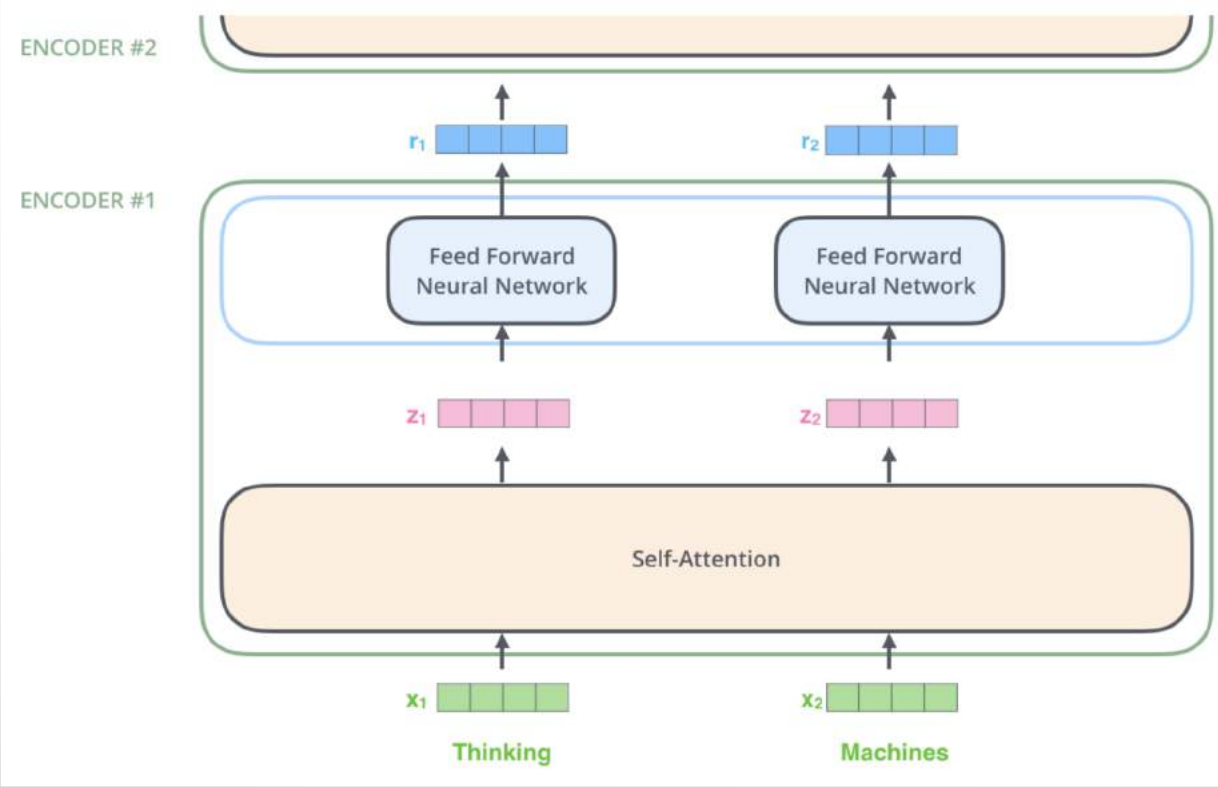
Figure 1: The Transformer - model architecture.



TRANSFORMER



TRANSFORMER

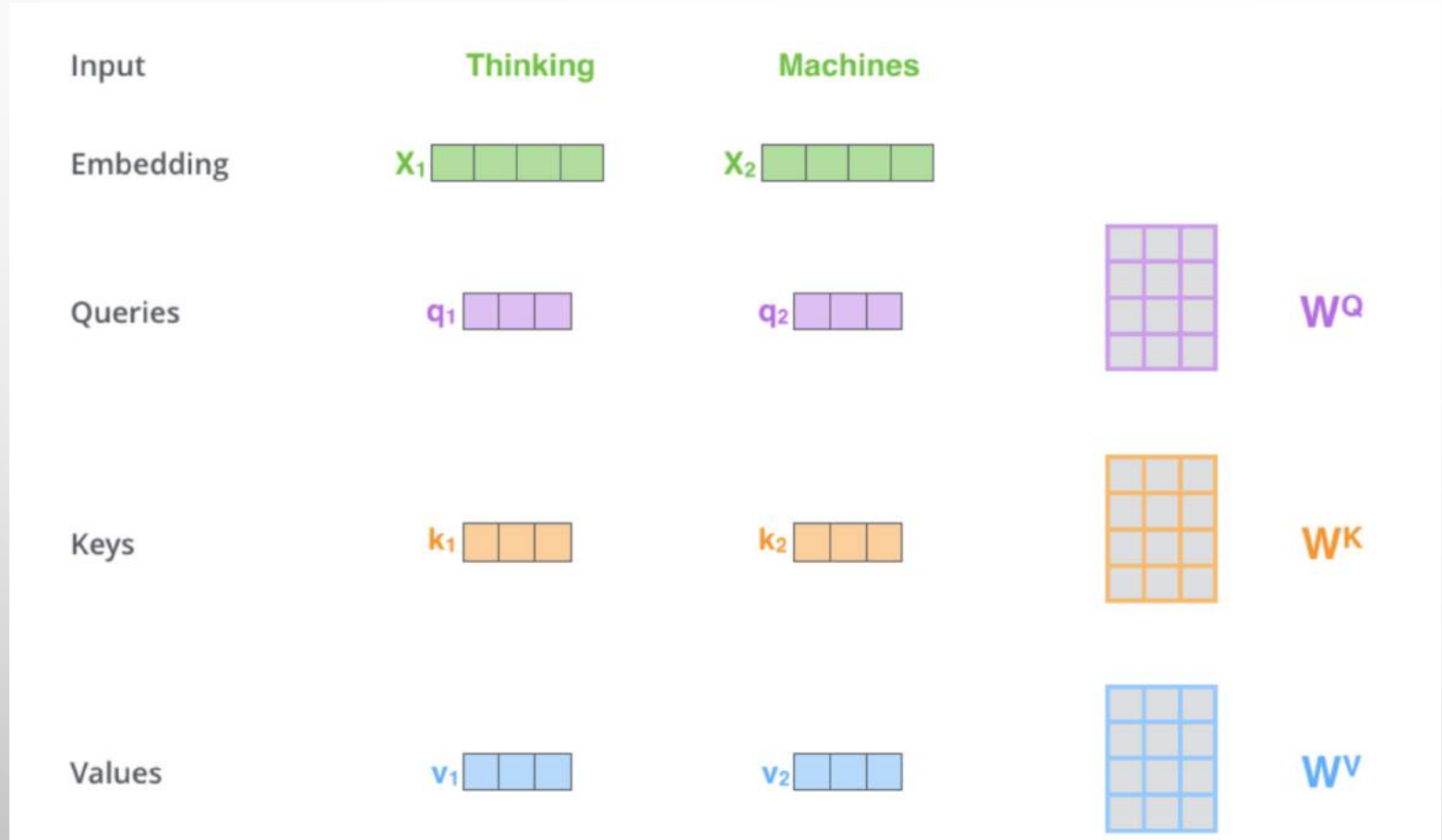


$$\text{FFN}(Z) = max(0, ZW_1 + b_1)W_2 + b_2$$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

TRANSFORMER SELF-ATTENTION IN DETAIL

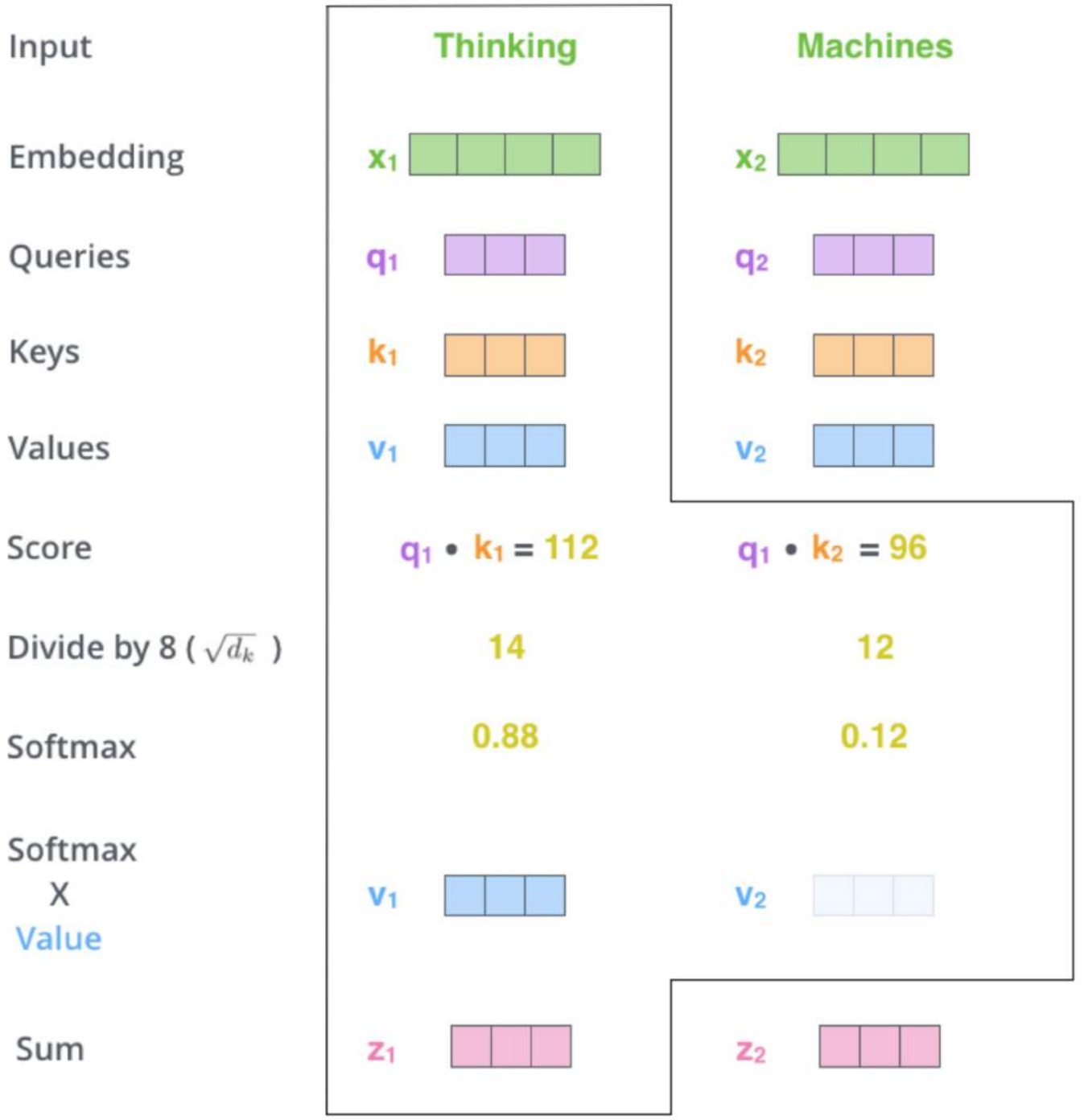
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



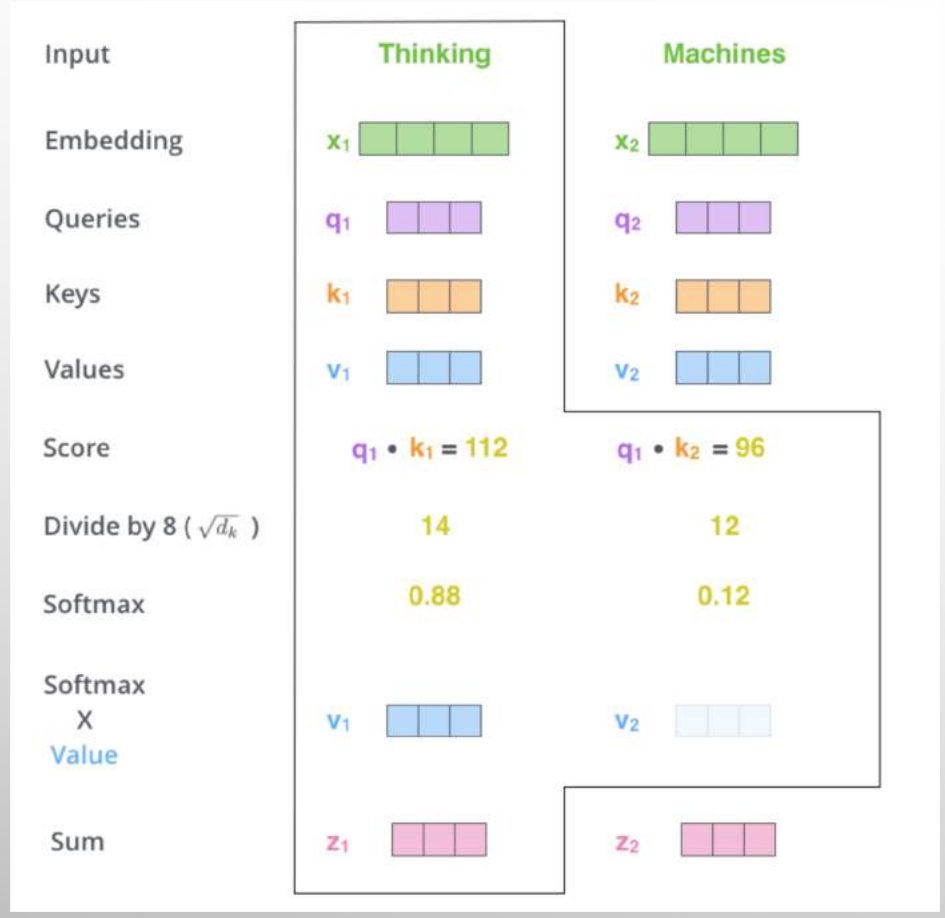
Multiplying x1 by the WQ weight matrix produces q1, the "query" vector associated with that word. We end up creating a "query", a "key", and a "value" projection of each word in the input sentence.

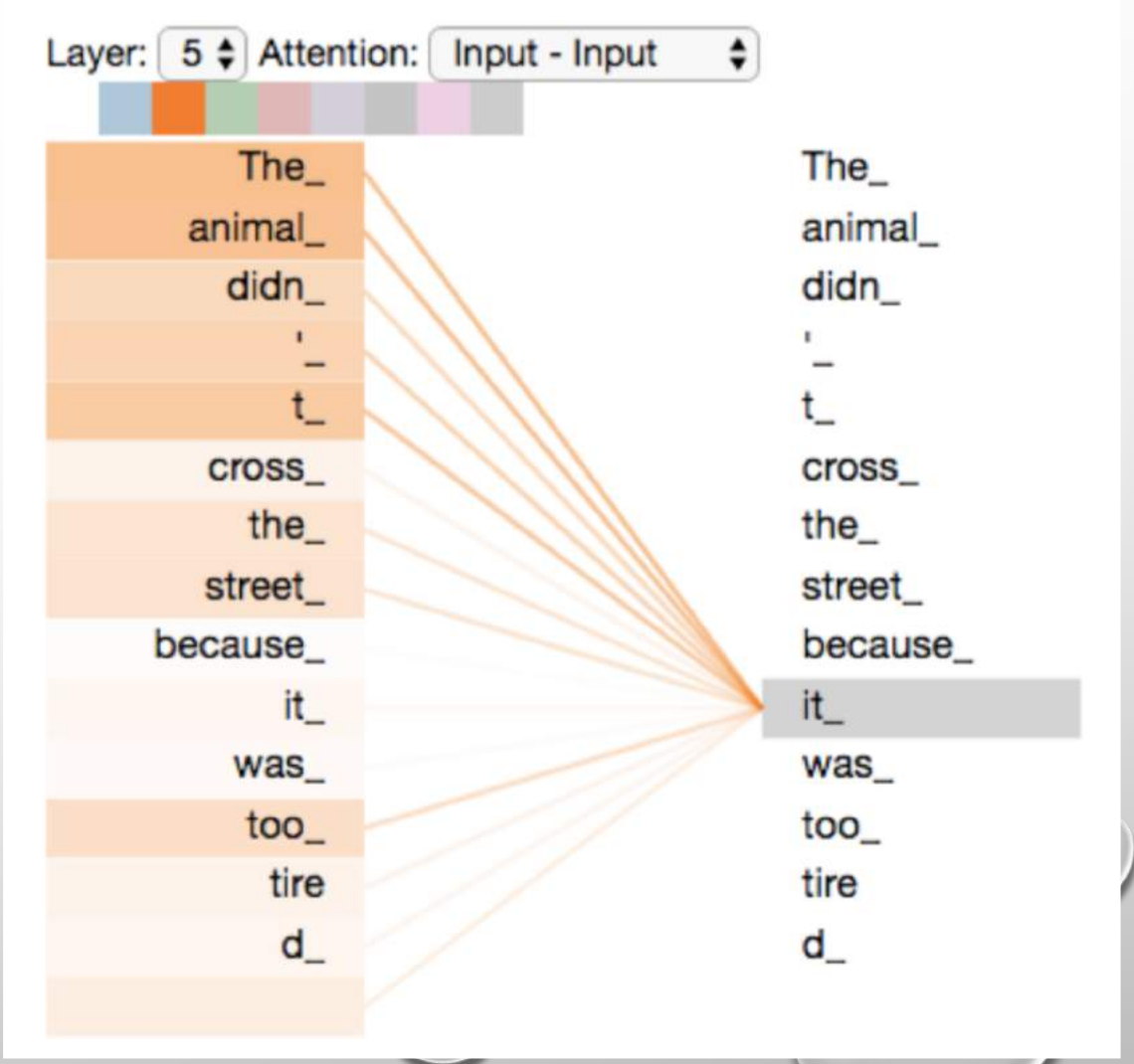
TRANSFORMER SELF-ATTENTION

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



TRANSFORMER SELF-ATTENTION





ATTENTION

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.

A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

RARE

Robust Scene Text Recognition with Automatic Rectification

Baoguang Shi, Xinggang Wang, Pengyuan Lyu, Cong Yao, Xiang Bai*
School of Electronic Information and Communications
Huazhong University of Science and Technology
shibaoguang@gmail.com, xbai@hust.edu.cn

Abstract

Recognizing text in natural images is a challenging task with many unsolved problems. Different from those in documents, words in natural images often possess irregular shapes, which are caused by perspective distortion, curved character placement, etc. We propose RARE (Robust

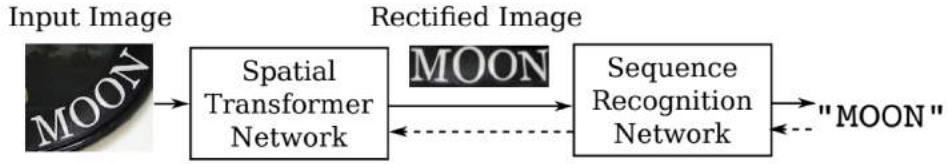


Figure 1. Schematic overview of RARE, which consists a spatial transformer network (STN) and a sequence recognition network (SRN). The STN transforms an input image to a rectified im-

https://arxiv.org/pdf/1603.03915.pdf

RARE

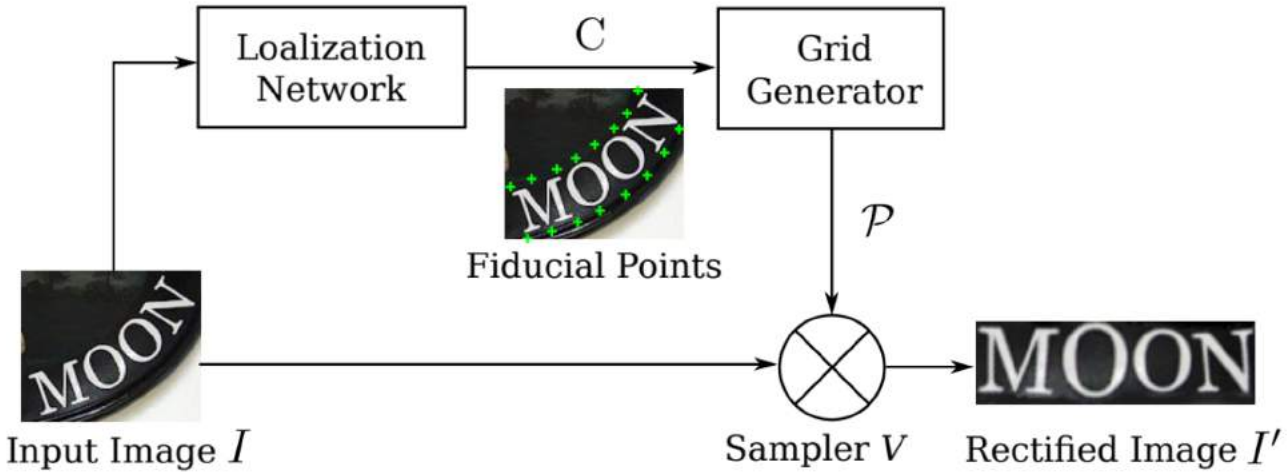


Figure 2. Structure of the STN. The localization network localizes a set of fiducial points $\mathbf{C}$, with which the grid generator generates a sampling grid $\mathcal{P}$. The sampler produces a rectified image I', given I and $\mathcal{P}$.

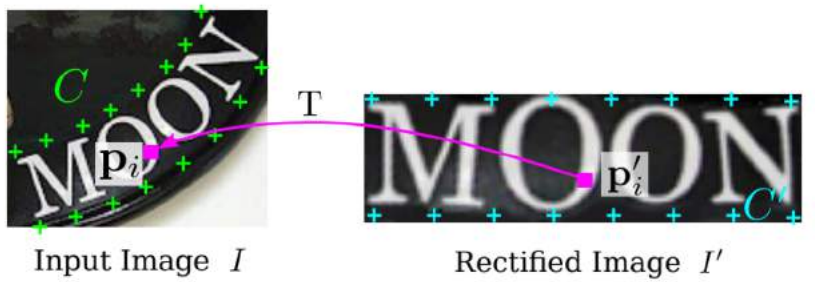


Figure 3. Fiducial points and the TPS transformation. Green markers on the left image are the fiducial points $\mathbf{C}$. Cyan markers on the right image are the base fiducial points $\mathbf{C}'$. The transformation $\mathbf{T}$ is represented by the pink arrow. For a point (x'_i, y'_i) on I', the transformation $\mathbf{T}$ finds the corresponding point (x_i, y_i) on I.

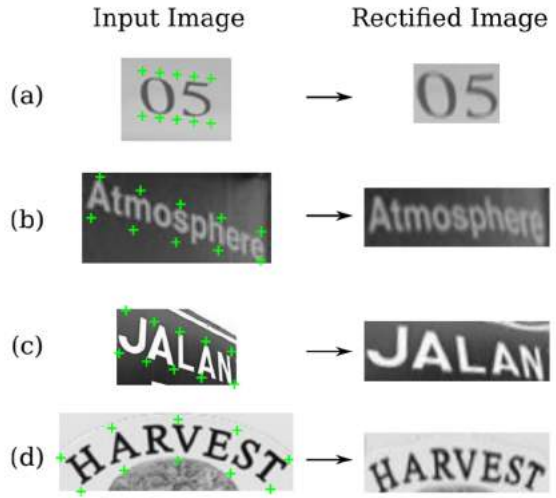


Figure 4. The STN rectifies images that contain several types of irregular text. Green markers are the predicted fiducial points on the input images. The STN can deal with several types of irregular text, including (a) loosely-bounded text; (b) multi-oriented text; (c) perspective text; (d) curved text.

RARE

The generation is a T-step process, at step t, the decoder computes a vector of attention weights $\boldsymbol{\alpha}_t \in \Re^L$ via the attention process described in [8]:

$$\boldsymbol{\alpha}_t = \text{Attend}(\mathbf{s}_{t-1}, \boldsymbol{\alpha}_{t-1}, \mathbf{h}), \qquad (6)$$

where $\mathbf{s}_{t-1}$ is the state variable of the GRU cell at the last step.

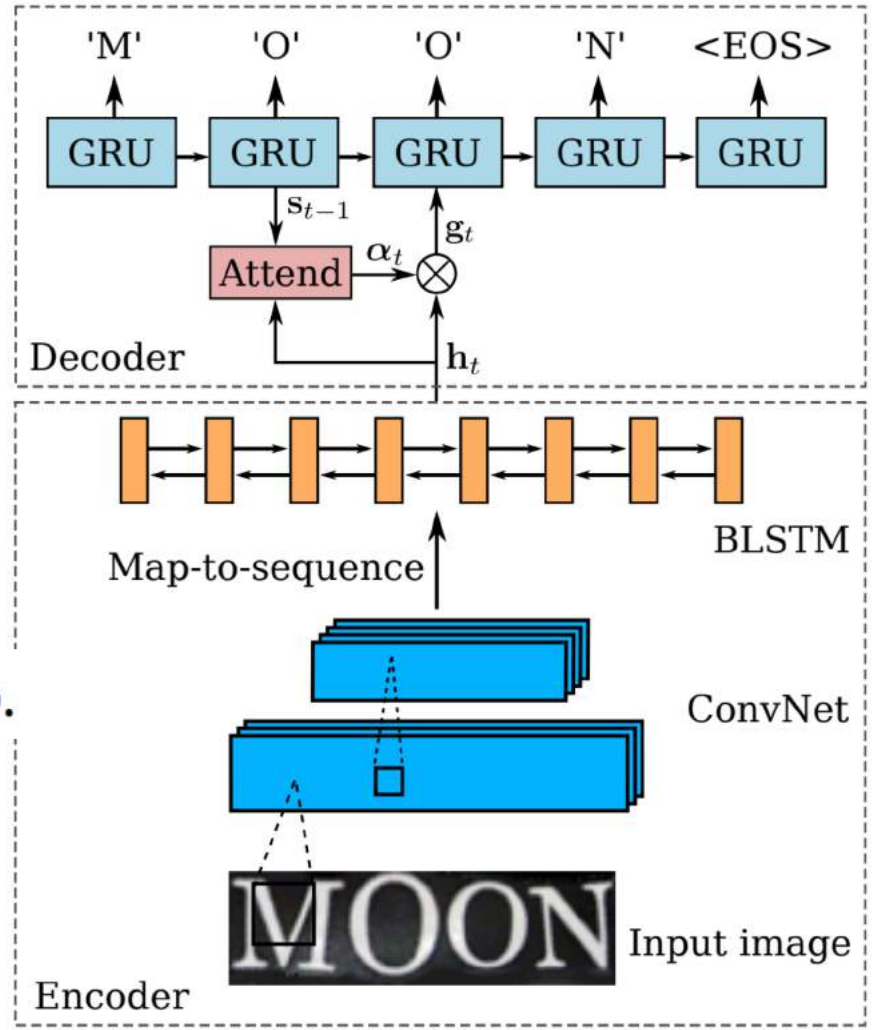


Figure 5. Structure of the SRN, which consists of an encoder and a decoder. The encoder uses several convolution layers (ConvNet) and a two-layer BLSTM network to extract a sequential representation ($\mathbf{h}$) for the input image. The decoder generates a character sequence (including the EOS token) conditioned on $\mathbf{h}$.

RARE

step. For $t = 1$, both $\mathbf{s}_0$ and $\boldsymbol{\alpha}_0$ are zero vectors. Then, a *glimpse* $\mathbf{g}_t$ is computed by linearly combining the vectors in $\mathbf{h}$: $\mathbf{g}_t = \sum_{i=1}^{L} \alpha_{ti}\mathbf{h}_i$. Since $\boldsymbol{\alpha}_t$ has non-negative values that sum to one, it effectively controls where the decoder focuses on.

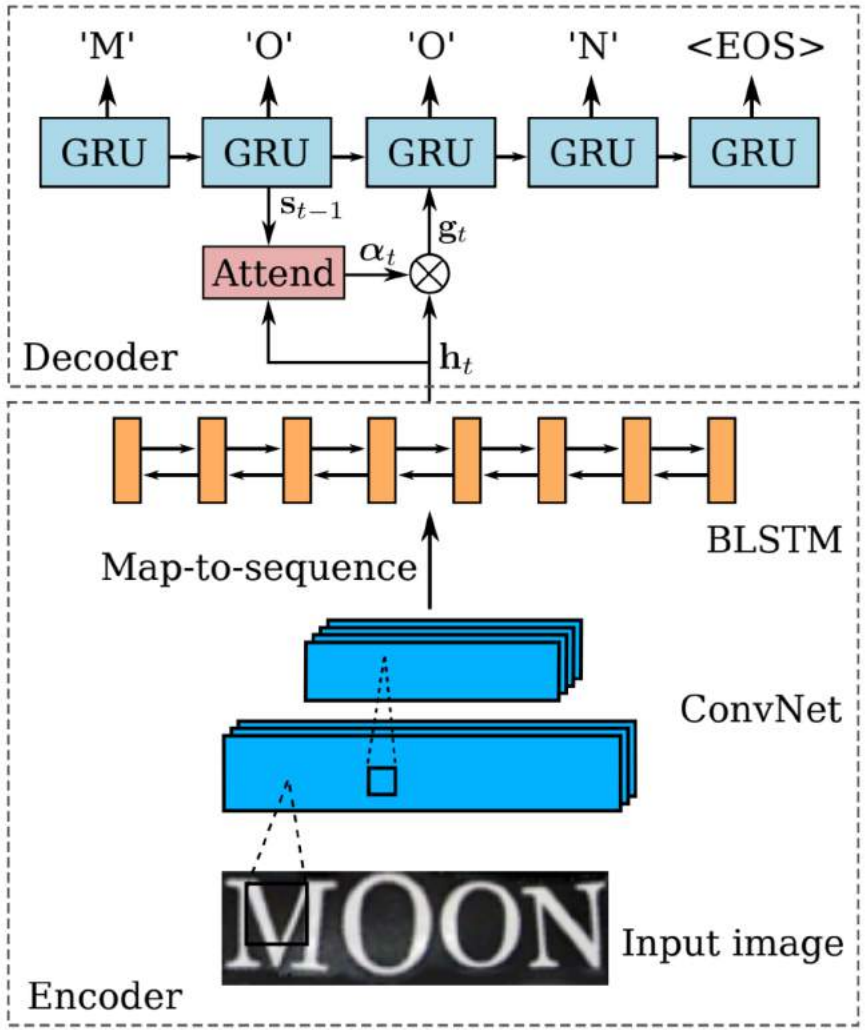


Figure 5. Structure of the SRN, which consists of an encoder and a decoder. The encoder uses several convolution layers (ConvNet) and a two-layer BLSTM network to extract a sequential representation ($\mathbf{h}$) for the input image. The decoder generates a character sequence (including the EOS token) conditioned on $\mathbf{h}$.

RARE

The state $\mathbf{s}_{t-1}$ is updated via the recurrent process of GRU [7, 8]:

$$\mathbf{s}_t = \mathrm{GRU}(l_{t-1}, \mathbf{g}_t, \mathbf{s}_{t-1}), \tag{7}$$

where l_{t-1} is the $(t-1)$-th ground-truth label in training, while in testing, it is the label predicted in the previous step, *i.e.* $\hat{l}_{t-1}$.

The probability distribution over the label space is estimated by:

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{W}^\intercal \mathbf{s}_t). \tag{8}$$

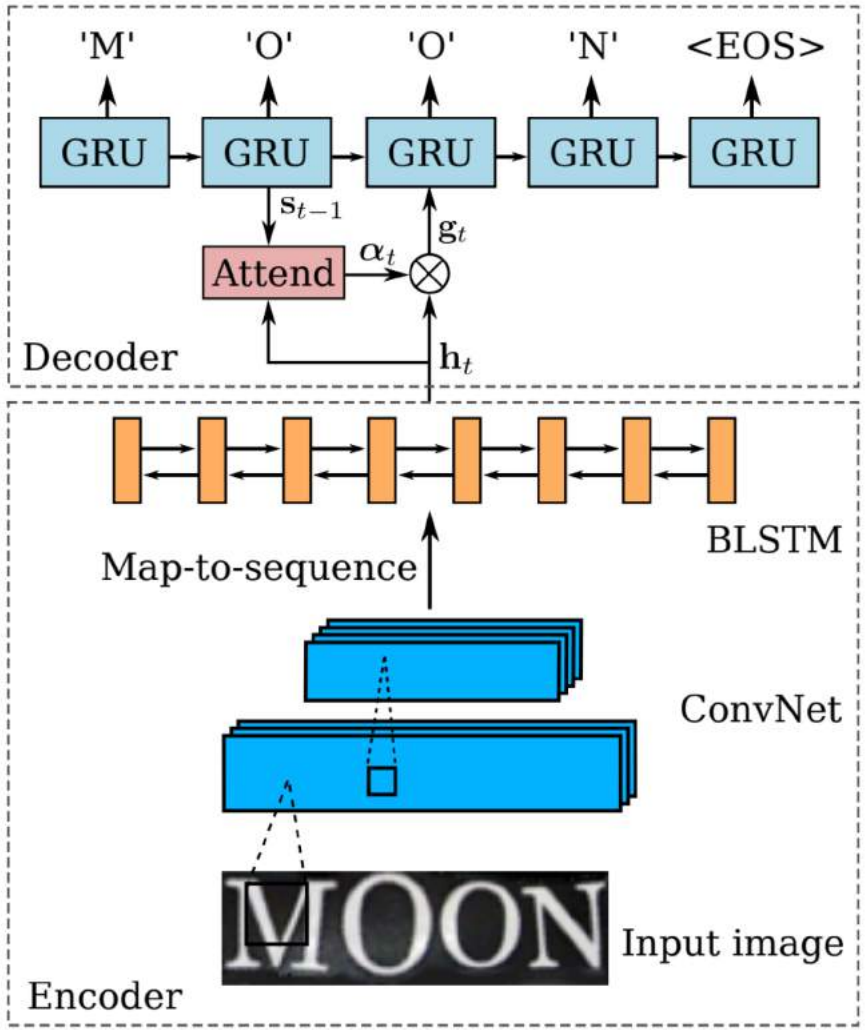


Figure 5. Structure of the SRN, which consists of an encoder and a decoder. The encoder uses several convolution layers (ConvNet) and a two-layer BLSTM network to extract a sequential representation ($\mathbf{h}$) for the input image. The decoder generates a character sequence (including the EOS token) conditioned on $\mathbf{h}$.

RARE

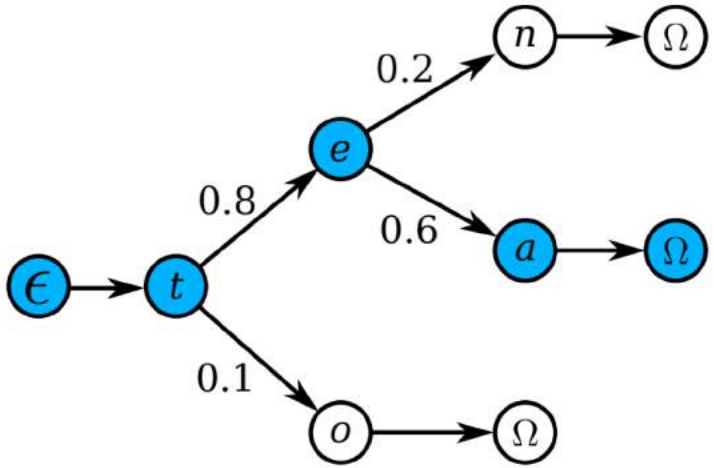


Figure 7. A prefix tree of three words: "ten", "tea", and "to". ϵ and Ω are the tree root and the EOS token respectively. The recognition starts from the tree root. At each step the posterior probabilities of all child nodes are computed. The child node with the highest probability is selected as the next node. The process iterates until a leaf node is reached. Numbers on the edges are the posterior probabilities. Blue nodes are the selected nodes. In this case, the predicted word is "tea".

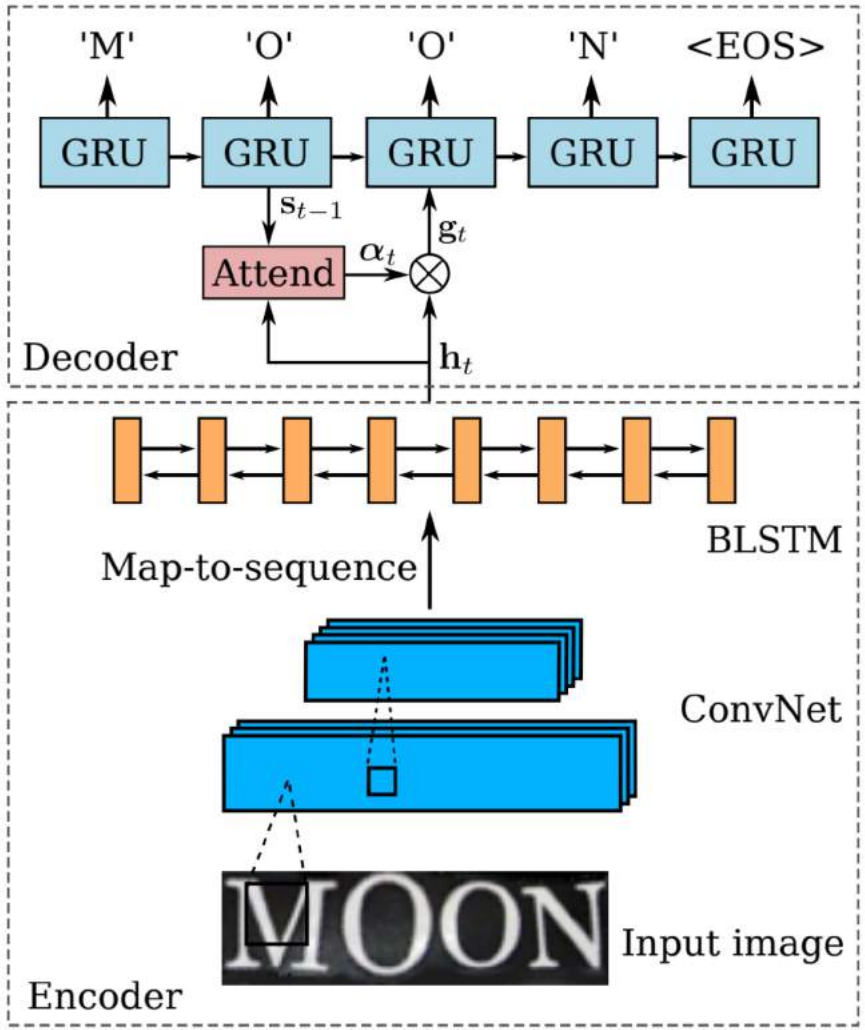


Figure 5. Structure of the SRN, which consists of an encoder and a decoder. The encoder uses several convolution layers (ConvNet) and a two-layer BLSTM network to extract a sequential representation ($\mathbf{h}$) for the input image. The decoder generates a character sequence (including the EOS token) conditioned on $\mathbf{h}$.

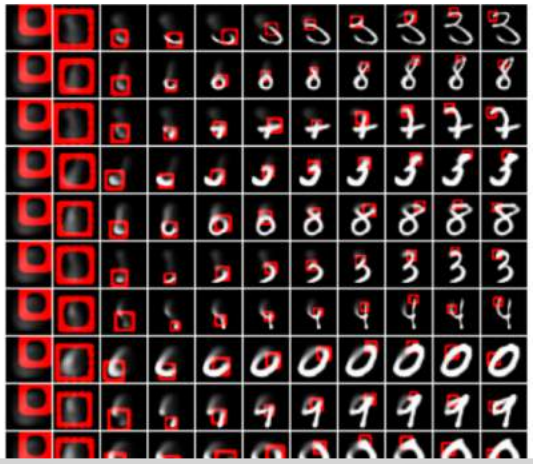
DRAW

DRAW: A Recurrent Neural Network For Image Generation

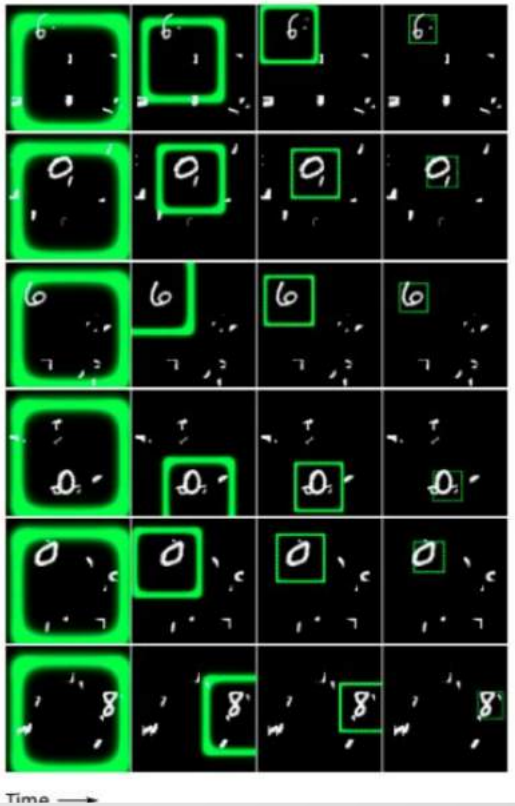
Karol Gregor KAROLG@GOOGLE.COM
Ivo Danihelka DANIHELKA@GOOGLE.COM
Alex Graves GRAVESA@GOOGLE.COM
Danilo Jimenez Rezende DANILOR@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Google DeepMind

Abstract

This paper introduces the *Deep Recurrent Attentive Writer* (DRAW) neural network architecture for image generation. DRAW networks combine a novel spatial attention mechanism that mimics the foveation of the human eye, with a sequential variational auto-encoding framework that allows for the iterative construction of complex images. The system substantially improves on the state of the art for generative models on MNIST, and, when trained on the Street View House Numbers dataset, it generates images that cannot be distinguished from real data with the naked eye.



Classify images by attending to arbitrary regions of the *input*



https://arxiv.org/pdf/1502.04623.pdf

作业

- CRNN
  - https://github.com/belval/crnn
  - https://github.com/bgshih/crnn

THANKS