DEEP LEARNING

An introduction to deep learning

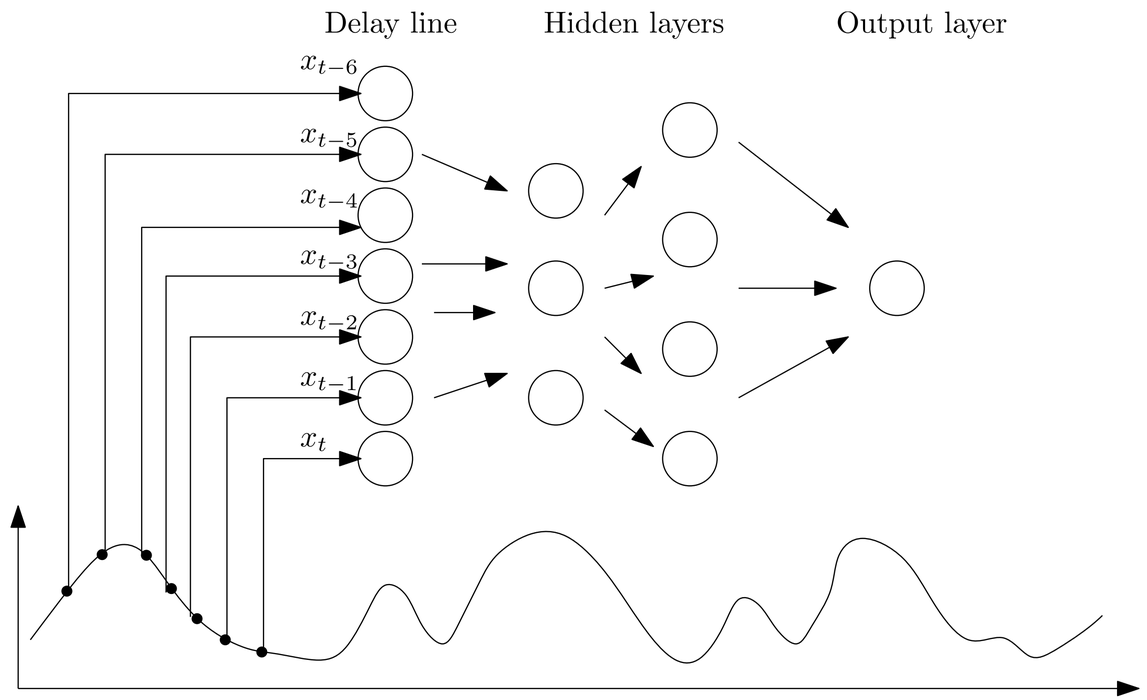
Jeremy Fix

January 10, 2021

DEALING WITH SEQUENTIAL DATA

SPATIALISING THE TIME : TDNN

Time delay neural networks as introduced in (Waibel, Hanazawa, Hinton, Shikano, & Lang, 1989) spatializes the time:



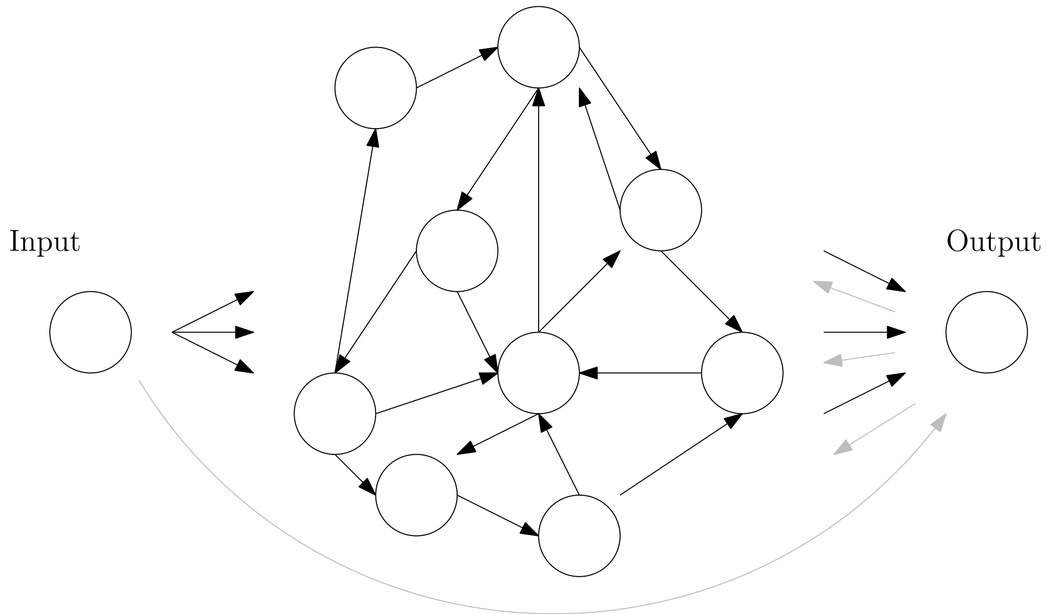
Time delay neural network

But : which size of the time window ? Must the history size always be the same ? Do we need the data over the whole time span ? How to share computations in time instead of using distinct weights per time instant?

Feedforward neural networks can still be efficient for processing sequential data, e.g. Gated ConvNet (Dauphin, Fan, Auli, & Grangier, 2017), Transformers, …

ARCHITECTURE OF A RNN

Introduced by (Elman, 1990).



Recurrent neural network

Weight matrices :

- W^{in} input to hidden
- W^{h} hidden to hidden
- W^{out} hidden to output
- W^{back} outputs to hidden

$$h(t) = f(W^{in}x(t) + W^{h}h(t-1) + W^{back}y(t-1) +$$
$$y(t) = g(W^{out}h(t) + b_y)$$

The hidden to hidden weight matrix W_h is repeatedly applied.

Named **Elman networks** if $W^{back} = 0$, and **Jordan networks** if $W^{h} = 0$. Elman networks with a random fixed W^{h} are called **Echo State networks**.

MANY TO ONE, ONE TO MANY, MANY TO MANY

The inputs and outputs can be of variable ($1 \rightarrow T_x, 1 \rightarrow T_y$) and arbitrary sizes ($T_x \neq T_y$).

- **Many to one** example : language model, sentiment analysis : multiclass sequence classification $T_y = 1$:
  - $['this', 'movie', 'was', 'fantastic'] \mapsto 1$
  - $['you', 'should', 'not', 'spend', 'even', 'a', 'single', 'penny', 'watching', 'this', 'movie'] \mapsto 0$
- **Many to many** example : Neural Machine Translation
  - What is the most likely EN translation of “La croissance économique s’est ralentie ces dernières années.” : “Economic growth has slowed down in recent years.”
- **One to many** example: image captioning, language model with probabilistic sampling

start : ‘LA JUMENT ET’

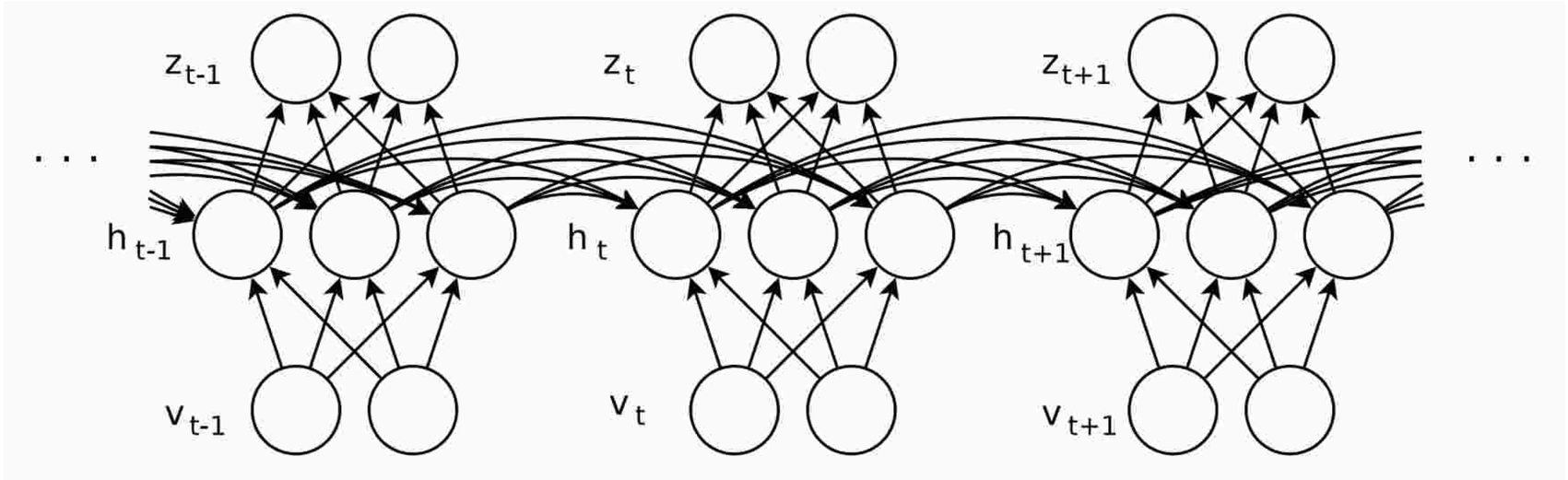
LA JUMENT ET LE RAT
ET L’huiller craignait les gens d’une mise un vers atteint:
Va c’est d’être indigne de Vénus d’aller pressez l’ame
D’une mais, dit-il, un plongeant l’avertion :
Son échangé vous refusiez-vous

start : ‘LA JUMENT ET’

LA JUMENT ET LE BULÉE
[Ésope]
Comme à part craindre déjà cet honneur à couvrir jamais
Et ses mélonces, condition tempérament.
L’autre honne alla vie.
Je ne saurais pas que d’un moutons.
Que ce choix, coquet, g

TRAINING WITH (TRUNCATED)-BPTT

Idea: unfold in time the computational graph and perform **reverse mode** differentiation (Werbos, 1990).



Backpropagation through time. Image from (Sutskever, 2013)

You must be training on truncated series to prevent a computational burden.

You can also perform **forward mode** differentiation (**Real time recurrent learning** RTTL (Williams & Peng, 1990)) with online adaptation as the inputs/targets comes in but this is computationally expensive.

TRAINING A RNN CAN BE HARD

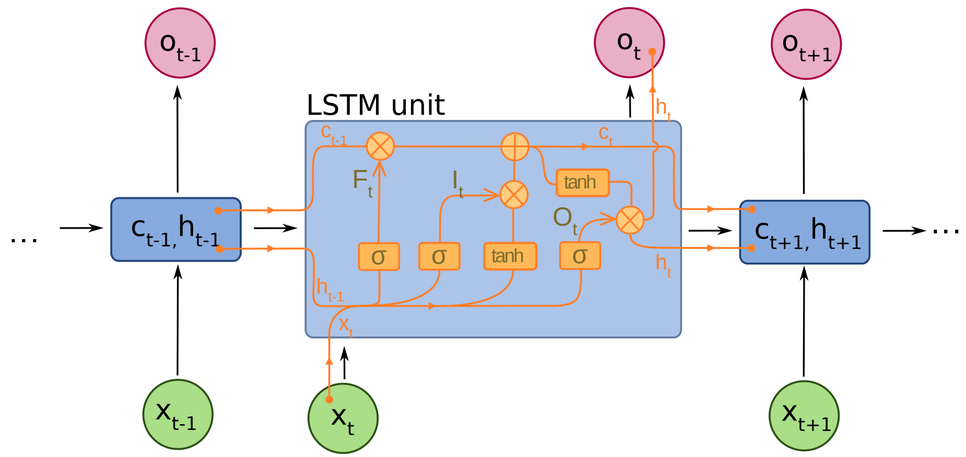
Unrolled in time, RNN appears as very deep networks $\rightarrow$ vanishing/exploding gradient

- Initialization strategies :
  - LSTM : high forget bias to favor a remember by default (Gers, Schmidhuber, & Cummins, 2000)
  - orthogonal weight initialization (Henaff, Szlam, & LeCun, 2016), to keep the hidden activities normalized $\|Wh\|_2^2 = h^T W^T W h = h^T h = \|h\|_2^2$, see also (Arjovsky, Shah, & Bengio, 2016)
  - identity recurrent weight initialization to favor a copy as-is by default for RNN (Le, Jaitly, & Hinton, 2015)
- Architecture :
  - in-lieu of Batch Normalization : Layer normalization (Ba, Kiros, & Hinton, 2016); statistics are computed independently per sample over the whole layer
- Training :
  - gradient clipping (Pascanu, Mikolov, & Bengio, 2013) seems to be frequently used,
  - activation clipping is sometimes considered (Hannun et al., 2014),
- Regularization:
  - Noise (Kam-Chuen Jim, Giles, & Horne, 1996), (Graves, Mohamed, & Hinton, 2013)
  - Naive dropout is not an option (LSTM/GRU memory cell states could be masked)
  - ZoneOut (Krueger et al., 2017), rnnDrop (Moon, Choi, Lee, & Song, 2015), ..

MEMORY CELLS

LONG-SHORT TERM MEMORY (LSTM)

RNNs have difficulties learning long range dependencies. The LSTM (Hochreiter & Schmidhuber, 1997) introduces **memory cells** to address that problem.



LSTM memory cell. Image from Wikipedia

Equations:

$$
\begin{aligned}
I_t &= \sigma(W_i^x x_t + W_i^h h_{t-1} + b_i) && \in [0,1], \text{Input ga} \\
F_t &= \sigma(W_f^x x_t + W_f^h h_{t-1} + b_f) && \in [0,1], \text{Forget g} \\
O_t &= \sigma(W_o^x x_t + W_o^h h_{t-1} + b_o) && \in [0,1], \text{Output g} \\
n_t &= \tanh(W_n^x x_t + W_n^h h_{t-1} + b_z) && \text{unit's input} \\
c_t &= F_t \odot c_{t-1} + I_t \odot n_t && \text{cell update} \\
h_t &= O_t \odot \tanh(c_t) && \text{unit's output}
\end{aligned}
$$

Peepholes may connect the c_t to **their** gates.

The next layers integrate what is exposed by the cells, i.e. the unit's output h_t, not c_t.

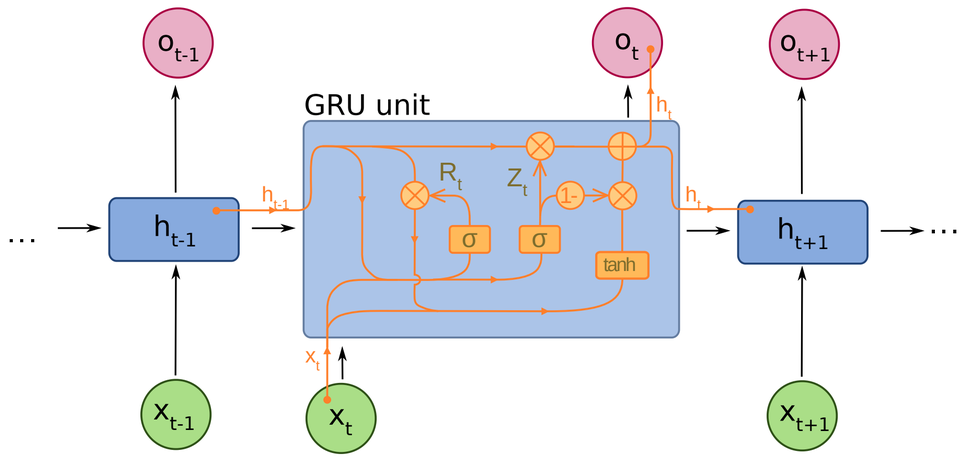
If $F_t = 1, I_t = 0$, the cell state c_t is unmodified. This is called the *constant error carrousel*.

The forget gate is introduced in (Gers et al., 2000). Variants have been investigated in a search space odyssey (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2017).

See also (Le et al., 2015) which reconsiders using ReLU in LSTM given appropriate initialization of the recurrent weights to the identity to be copy by default mode.

GATED RECURRENT UNITS (GRU)

The GRU is introduced as an alternative, simpler model than LSTM. Introduced in (Cho et al., 2014).



GRU memory cell. Image from Wikipedia

Equations:

$$R_t = \sigma(W_i^x x_t + W_i^h h_{t-1} + b_i) \text{ Reset gate}$$
$$Z_t = \sigma(W_z^x x_t + W_z^h h_{t-1} + b_z) \text{ Update gate}$$
$$n_t = \tanh(W_n^x x_t + b_{nx} + R_t \odot (W_n^h h_{t-1} + b_{nh}))$$
$$h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot n_t$$

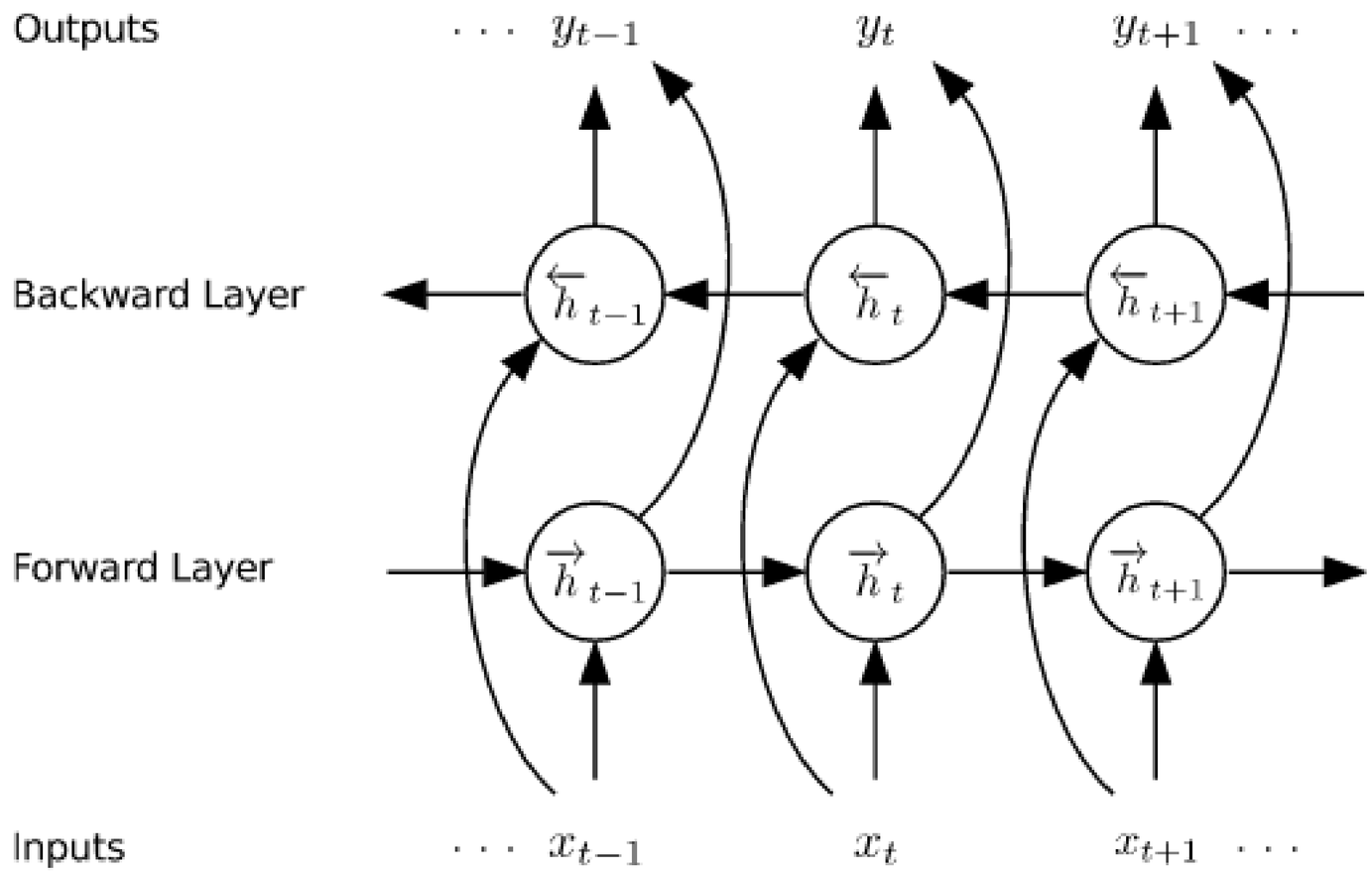
If $Z_t = 1$, the cell state h_t is not modified. If $Z_t = 0$ and $R_t = 1$, it is updated in one step.

Compared to LSTM, a GRU cell :

- unconditionally exposes its hidden state (there is no private cell state c_t)
- the hidden state is reset by getting $R_t = 0$

BIDIRECTIONAL RNN/LSTM/GRU

Idea Both past and future contexts can sometimes be required for classification at the current time step; e.g. when you speak, past and future phonemes influence the way you pronounce the current one.
Introduced in (Schuster & Paliwal, 1997})



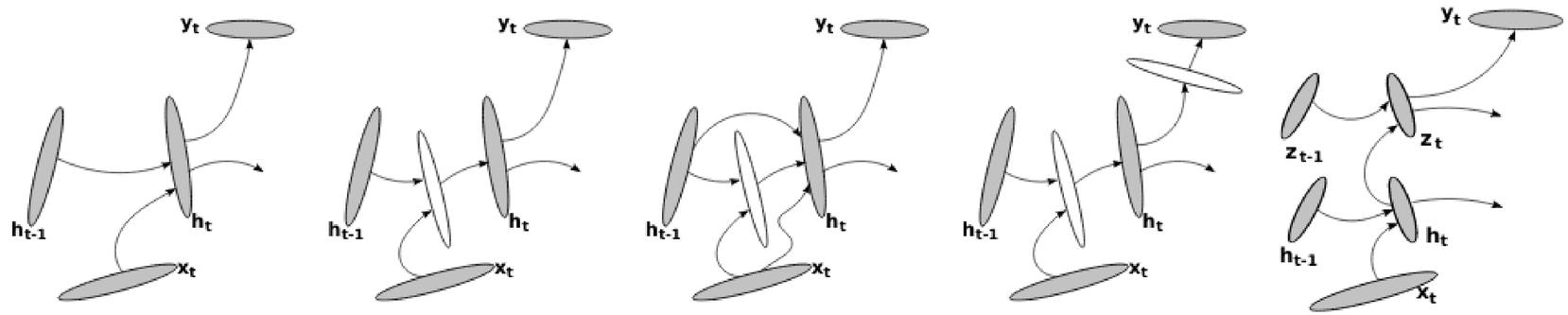
Bidirectionnal RNN. Image from (Graves et al., 2013)

DEEP RNNS

While RNN are fundamentally deep neural networks, they can still benefit from being stacked : this allows the layers to operate at increasing time scales. The lower layers can change their content at a higher rate than the higher layers.

- (Graves et al., 2013): Phoneme classification with stacked bidirectionnal LSTMs
- (Sutskever, Vinyals, & Le, 2014) : Machine translation with stacked unidirectionnal LSTMs (Seq2Seq)

In a stacked RNN, you can concatenate consecutive hidden states before feeding in the next RNN layer, e.g. Listen, Attend and Spell encoder ($\rightarrow$ downscale time)



(a) RNN (b) DT-RNN (b*) DT(S)-RNN (c) DOT-RNN (d) Stacked RNN

Deep RNN variants. Image from (Pascanu, Dauphin, Ganguli, & Bengio, 2014)

Other variants for introducing depth in RNN is explored in (Pascanu et al., 2014). For example, the transition function from h_{t-1} to h_t is not deep, even in stacked RNNs but is deep in DT-RNN.

DEFINING RNN IN PYTORCH

Stacked bidirectional LSTM, documentation

with discrete inputs (words, characters, …) of the same time length, one prediction per time step.

```
import torch
import torch.nn as nn

seq_len = 51
batch_size = 32
vocab_size = 10
embedding_dim = 128
hidden_size = 256
```

You can provide an initial state to the call function of the LSTM, in which case, you must take out the LSTM from the nn.Sequential, by default $\overrightarrow{h}_0 = \overleftarrow{h}_0 = \overleftarrow{c}_0 = \overrightarrow{c}_0 = 0$). You could learn these initial hidden states (to bias the first operations of your rnn).

All the weights and biases and initialized from LeCun like initialization $\mathcal{U}(-\sqrt{k}, \sqrt{k}), k = \frac{1}{\text{hidden\_size}}$

Some authors (Gers et al., 2000) suggest to favor either long-term dependencies or short-term dependencies by setting the bias of the forget gate accordingly ("Learning to forget", $F_{t=0} = 1$ to remember everything by default).

See the lab work for specificities on representing variable sized sequences with pytorch PackedSequences.

DIGGING INTO PYTORCH CODE

How do you know how to access these weights ? See the doc

CUSTOM INITIALIZATION

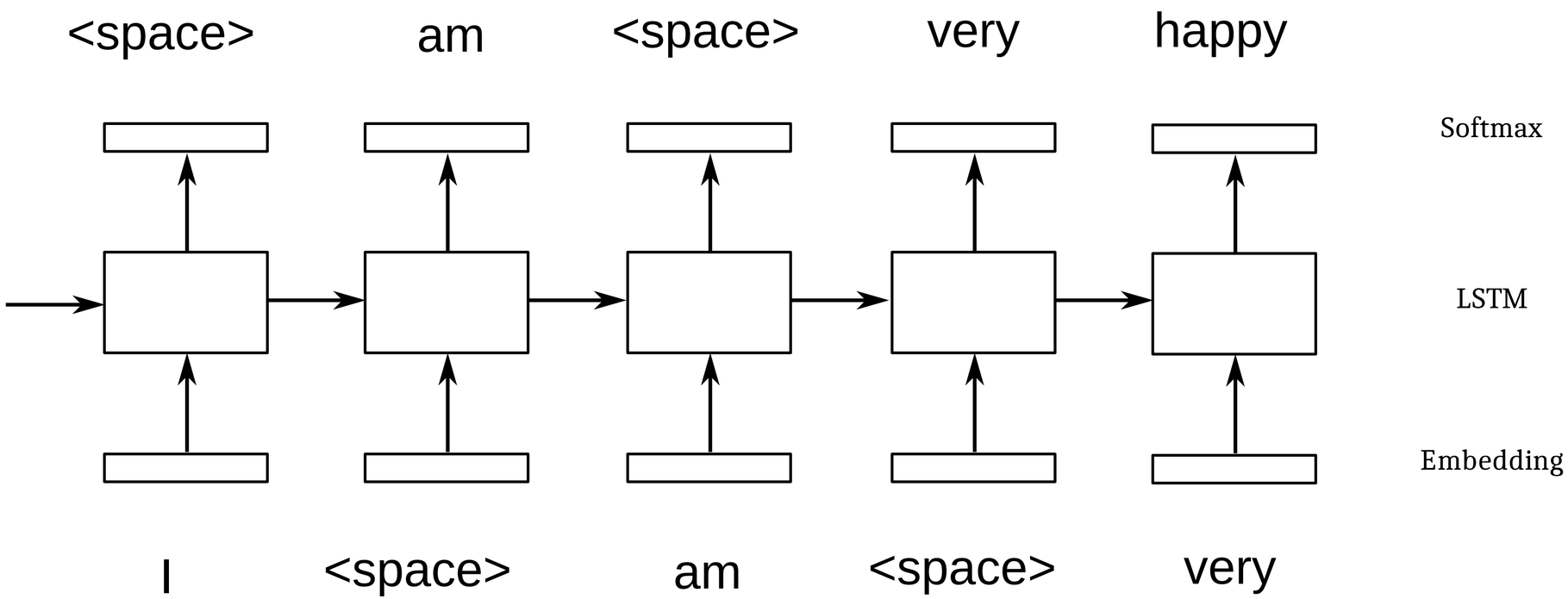
```
num_layers=3

rnn = nn.LSTM(input_size=embedding_dim,
              hidden_size=hidden_size,
              num_layers=num_layers,
              bidirectional=True)

    # Initialize to high forget gate bias
```

The ordering of the weights/biases are inputg/forgetg/cell/outputg.

LANGUAGE MODEL : AN EXAMPLE OF ALIGNED SEQUENCES OF IDENTICAL SIZES

CHARACTER LEVEL LANGUAGE MODEL (CHAR-RNN)

Problem given fixed length chunks of sentences, predict the next word/character : $p(x_T|x_0, x_1, \ldots, x_{T-1})$



Example RNN language model

Many to many during training (teacher forcing (Williams & Peng, 1990)) but many to one for inference.

A language model can be used, e.g., to constrain the decoding of a network outputting sentences (e.g. in speech-to-text or captioning tasks)

See also The unreasonnable effectiveness of recurrent neural networks and (Sutskever, Martens, & Hinton, 2011).

TRAINING AND SAMPLING FROM THE CHARACTER RNN

Example on “Les fabulistes”

- Vocabulary of 105 elements : {‘\t’: 0, ‘\n’: 1, ’ ‘: 2,’!‘: 3,’“‘: 4, "’”: 5, ‘(’: 6, ‘)’: 7, ‘*’: 8, ‘+’: 9, ‘,’: 10, ‘-’: 11, ‘.’: 12, ‘/’: 13, ‘0’: 14, ‘1’: 15, ‘4’: 16, ‘5’: 17, ‘6’: 18, ‘8’: 19, ‘9’: 20, ‘:’: 21, ‘;’: 22, ‘<’: 23, ‘>’: 24, ‘?’: 25, ‘A’: 26, ‘B’: 27, ‘C’: 28, ‘D’: 29, ‘E’: 30, ‘F’: 31, ‘G’: 32, ‘H’: 33, ‘I’: 34, ‘J’: 35, ‘L’: 36, ‘M’: 37, ‘N’: 38, ‘O’: 39, ‘P’: 40, ‘Q’: 41, ‘R’: 42, ‘S’: 43, ‘T’: 44, ‘U’: 45, ‘V’: 46, ‘W’: 47, ‘X’: 48, ‘Y’: 49, ‘Z’: 50, ‘\[’: 51, …}
- Dataset size : 10.048 non overlapping chunks of length 60.
- Example samples :
  - Input : [2,71,67,54,70,57,10,2,56,95,71…65,57,66,72,2,70,57,55,60,57]

    " sobre, dé…ment reche"
  - Output [71,67,54,70,57,10,2,56,95,71,61…57,66,72,2,70,57,55,60,57,70]

    “sobre, dés…ent recher”
- Network : Embedding(64) , 2$\times$ LSTM(64), 2$\times$LinearRelu(128), LinearSoftmax(105), 111.657 parameters

Note we use **uni**-directional LSTM. With **bi**-directionnal LSTM, the problem is easily solved by using the backward LSTM only.

- Loss : cross-entropy averaged over batch_size $\times$ seq_len
- Training: Adam(0.01), learning rate halved every 10 steps, gradient clipping (5) (not sure it helped though)

SAMPLING FROM THE CHARACTER RNN

- After 30 epochs, validation loss of 1.45 and validation accuracy of 56%.
- To sample from the language model, you can provide it with some context sentence, e.g. [‘L’, ‘A’, ’ ‘, ’G’, ‘R’,‘E’,‘N’,‘O’,‘U’, ‘I’, ‘L’, ‘L’, ‘E’, ’ ’]

Sample of 200 chars after init

LA GRENOUILLE y
-Ô)asZYc5[h+IÉë?8—>Y.bèqp;ÎzÇÇ<)!|f]Lt+«-u
XûoÜ:!ïgVùb|Ceü9ùÈ«à
6)ZàÀçBJi)X:ZÛdzxQ8PcviV]O]xPX,Înc.è’Pâs:X;ûfjBâ?X
*ç’ED’fSOl*Z(È’È1SnjàvPïLoUÊêàDgùO9z8eJûRYJ?Yg*
Uâp|jCbû—HxBràZBMZÛPCGuR’]ÀiÊÂSBF4D),û

Sample 1 of 200 chars after 30 epochs

LA GRENOUILLE ET MOURE ET LA RENARDIER
Quel Grâce tout mon ambassade est pris.
L’un pourtant rare,
D’une première
Qu’à partout tout en mon nommée et quelques
fleuris ;
Vous n’oserions les Fermerois, les heurs la

Note the upper case after the line breaks, the uppercase title, the quite existing words. The text does not make much sense but it is generated character by character !

Sample 2 of 200 chars (from the same model as before)

LA GRENOUILLE D’INDÉTES
[Phèdre]
Tout faire force belle, commune,
Et des arts qui, derris vôtre gouverne a rond d’une partage conclut sous besort qu’il plaît du lui dit Portune
comme un Heurant enlever bien homme,

More on language modeling (metrics, models, …) in the Deep NLP lecture of Joel Legrand.

GENERATING TEXT DESCRIPTIONS FROM IMAGES (ONE TO MANY)

IMAGE CAPTIONING

Problem Given an image, generate a textual description of it.

Example datasets : Coco captions, Flickr8k, Flickr30k

The man at bat readies to swing at the pitch while the umpire looks on.

A large bus sitting next to a very tall building.

Example of captioning samples from MSCoco Image captioning 2015

Some of the first entries : (Vinyals, Toshev, Bengio, & Erhan, 2015), (Xu et al., 2016)

Difficulty: object detection with their relationship

SHOW AND TELL

Idea Use a pre-trained CNN for image embedding plugged into a RNN for generating (decoding) the sequence of words. Introduced in (Vinyals et al., 2015).

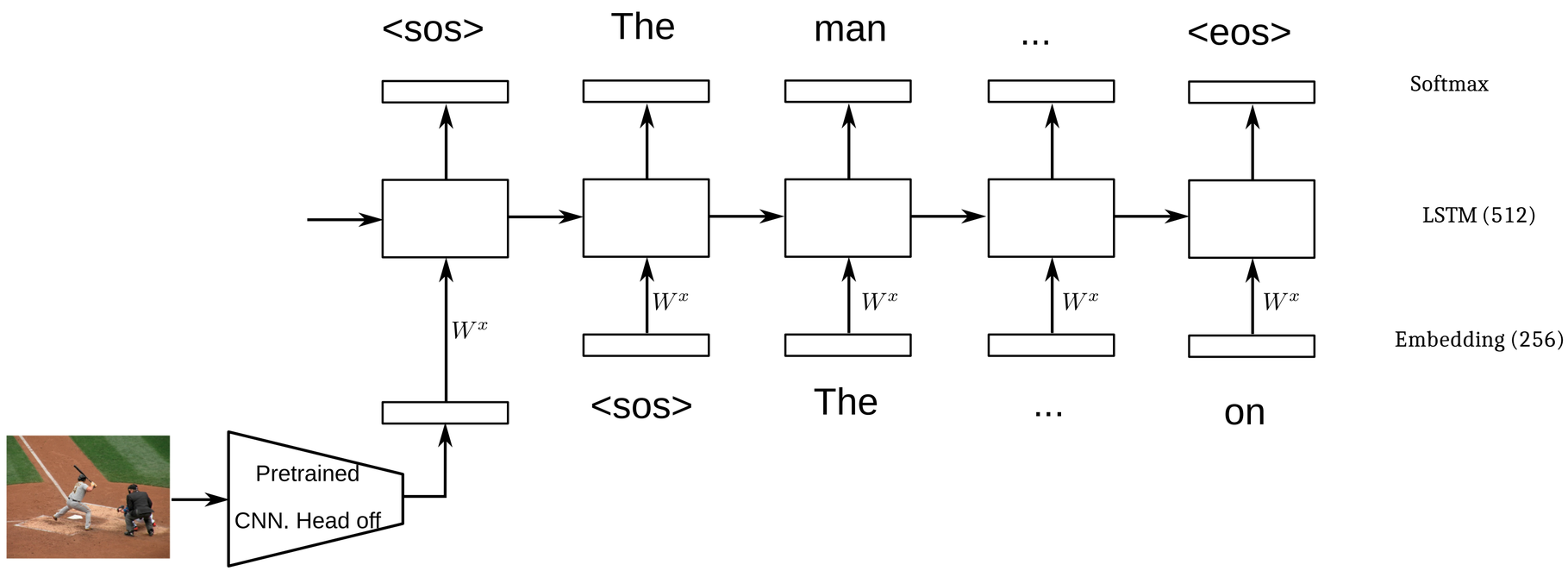
Learn a model maximizing :

$$p(S_0 S_1 S_2 .. S_T | I, \theta) = p(S_0 | I, \theta) \prod_{j>0} p(S_j | S_0 S_1 ... S_{j-1}, I, \theta)$$

i.e. minimizing $-\log(p(S_0 S_1 S_2 .. S_T | I, \theta)) = -\sum_j \log(p(S_j | S_0 ... S_{j-1}, I, \theta))$

Inspired by the Seq2Seq approach successful in machine translation (more on this later), they proposed an encoder-decoder model to *translate an image to a sentence*

SHOW AND TELL



Show and tell architecture

Training ingredients :

- GoogleNet CNN pretrained on ImageNet
- words embeddings randomly initialized (pretraining on a large news corpus did not help)
- embedding of size 512
- LSTM with 512 cells
- Stochastic gradient descent, no momentum,
- training the LSTM with frozen CNN then finetuning the whole. Too early training end-to-end fails
- scheduled sampling (otherwise, divergence between teacher forcing training and inference performances)

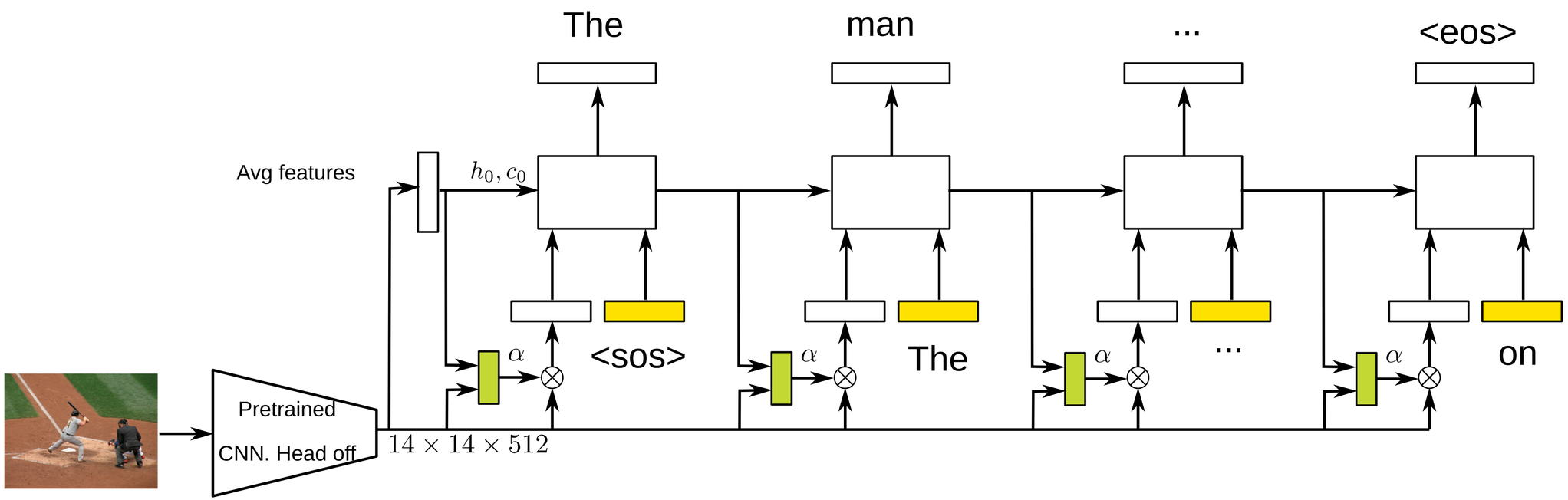
Introducing the visual convolutional features at every step did not help.

Inference :

- Decoding by beam search (beam size 20), then reduced beam size to 3 yielded, unexpectedly, better results

SHOW ATTEND AND TELL

Idea Allow the RNN to filter out/focus on CNN features during generation using an attention mechanism (Bahdanau, Cho, & Bengio, 2015). Introduced in (Xu et al., 2016). Link to theano source code



Show attend and tell architecture with soft attention. The alphas are normalized to sum to 1 (softmax).

Training:

- resnet CNN (head off)
- vocabulary of 10000 words
- Embedding (100), LSTM(1000),
- RMSProp(0.1),
- dropout for h_0, c_0, ,
- Early stopping on the BLUE score

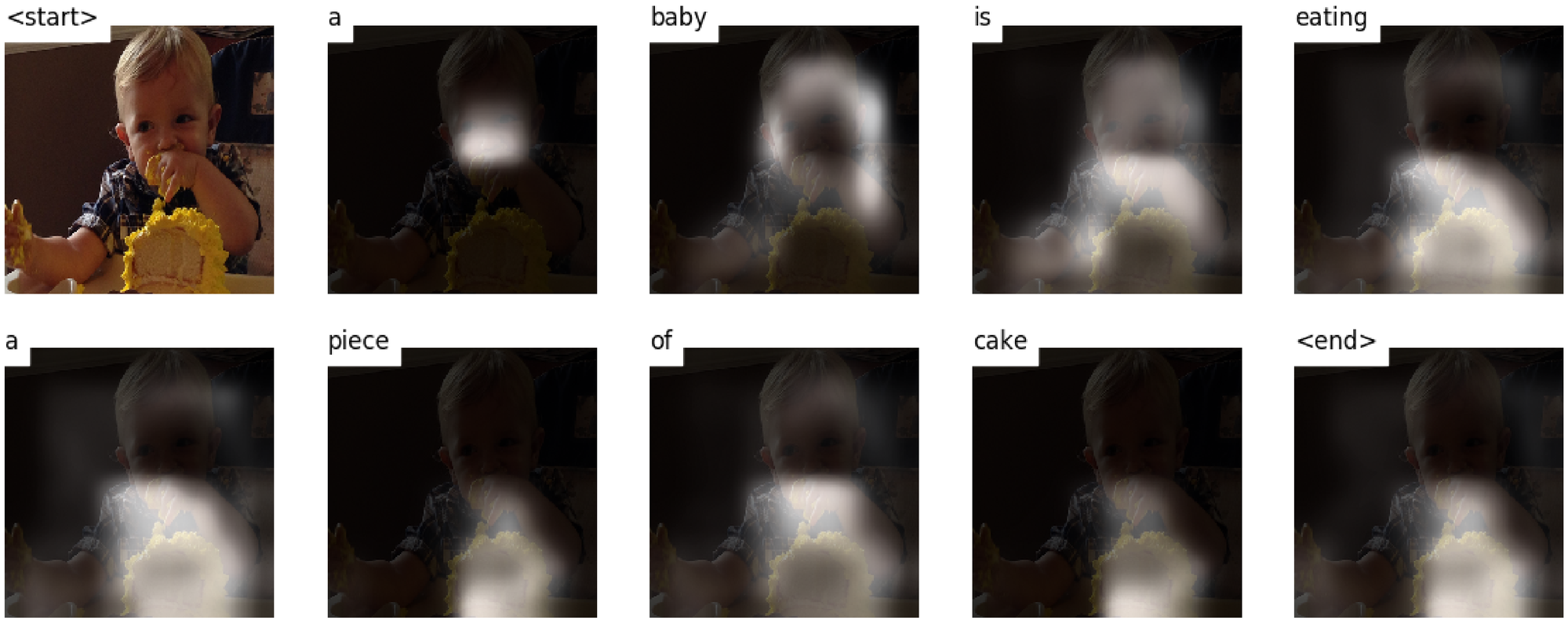
Double stochastic attention :

- by construction $\sum_i \alpha_{t,i} = 1$
- regularization $\lambda \sum_{loc}(1 - \sum_t \alpha_{t,loc})^2$ to enforce the model to pay equal attention to all the locations, norm in time for every location.

Inference:

- Decoding by beam search

SHOW ATTEND AND TELL



Example of caption with the attentional mask. Image from this nice tutorial and pytorch implementation

DEALING WITH VARIABLE SIZE UNALIGNED INPUT/OUTPUT SEQUENCES

WHERE IS THE PROBLEM

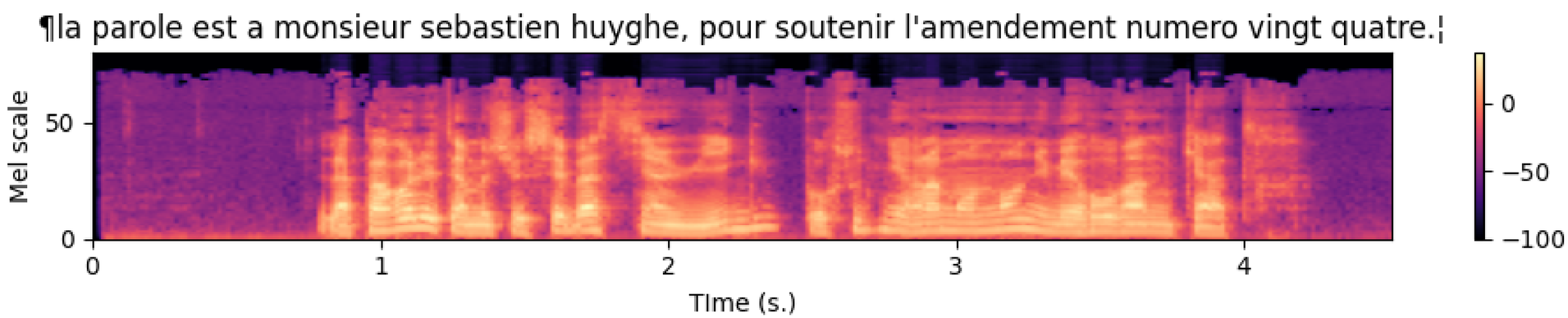
Problem In tasks such as Machine Translation (MT) or Automatic Speech Recognition (ASR), input sequences get mapped to output sequences, both can be of arbitrary sizes.

Machine translation :

The proposal will not now be implemented

Les propositions ne seront pas mises en application maintenant

Automatic speech recognition



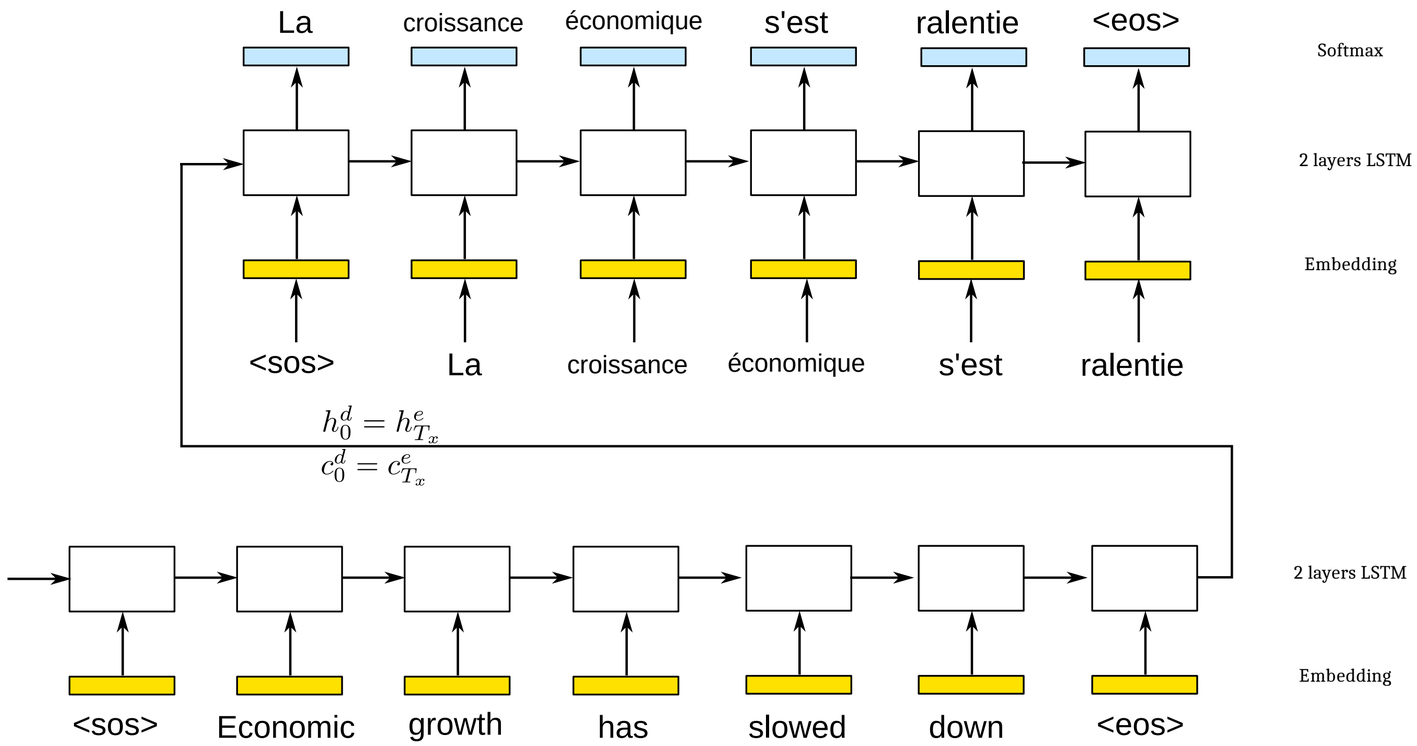
A mel-spectrogram with its expected transcript. The spectrogram is sampled at

The **alignment** can be difficult to explicit. Contrary to the language model, we may not know easily when to output what.

WHEN THE ALIGNMENT IS MISSING : SEQ2SEQ

ENCODER / DECODER ARCHITECTURES

Idea Encode/Compress the input sequence to a hidden state and decode/decompress the output sequence from there. Introduced in (Cho et al., 2014) for ranking translations and (Sutskever et al., 2014) for generating translations (NMT).



Seq2Seq architecture for Neural Machine Translation

Architecture :

- 4 layers LSTM$(1000,$ $\mathcal{U}(-0.08, 0.08))$, Embeddings$(1000)$
- Vocabularies (in:160.000, out: 80.000)
- SGD(0.7), halved every half epoch after 5 epochs. Trained for 7.5 epochs. Batch(128)
- gradient clipping 5
- 10 days on 8 GPUs

The input sentence is fed in reverse order.

Beam search decoding. Teacher forcing for training but see also Scheduled sampling or Professor Forcing.

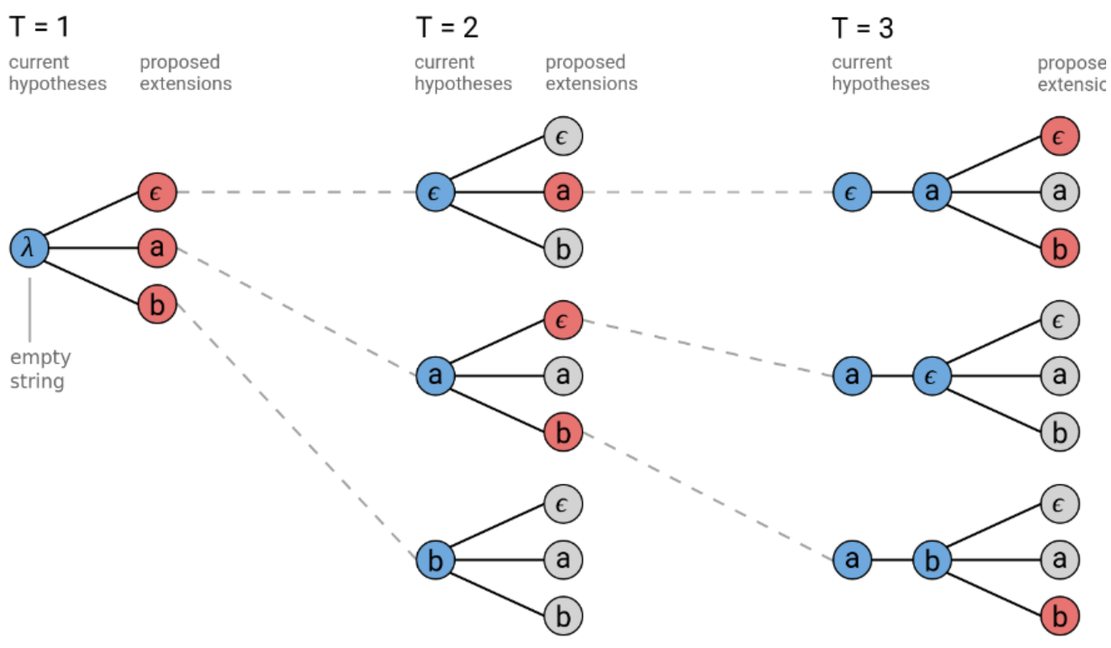
See Cho's blog post. See this implementation in pytorch

DECODING WITH BEAM SEARCH

To get the most likely translation, you need to estimate

$$p(y|x) = p(y_0|x, \theta) \prod_t p(y_t|y_0 \dots y_{t-1} x \theta)$$

But the probability distribution over the labels is dependent on the previously generated label (which feeds the input for the next step) $\rightarrow$ approximate search by maintaining a set of B candidates.



Beam search decoding with beam size B. Image from distill.pub

≡ See also the modified beam search scoring of GNMT (Wu et al., 2016).

WHEN THE ALIGNMENT IS MISSING : CTC

CONNECTIONIST TEMPORAL CLASSIFICATION (CTC)

Idea For problems with the output sequence length T_y is smaller than the input sequence T_x, allow a **blank** character. Introduced in (Graves, Fernández, Gomez, & Schmidhuber, 2006)

CTC collapsing. Illustration from distill.pub

The collapsing many-to-one mapping $\mathcal{B}$ removes the duplicates and then the blanks.

The CTC networks learn from all the possible alignments of X with Y by adding the extra-blank character. Allows to learn from **unsegmented** sequences !

See also alternatives of the **blank** character in (Collobert, Puhrsch, & Synnaeve, 2016).

CTC TRAINING : CTC LOSS

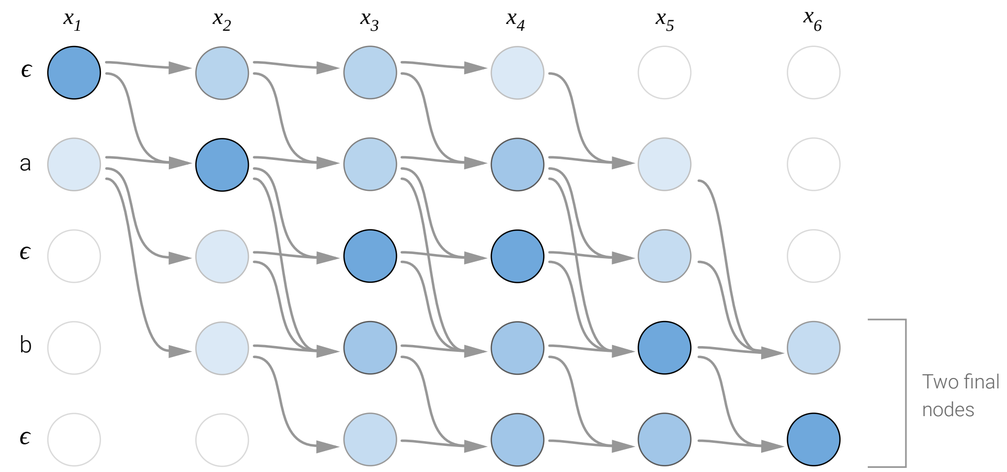
- Step: extend your model to output a **blank** character.

- Use the CTC Loss which is estimating the probability of a labeling by marginalizing over all the possible alignments. Assuming conditional independence of the outputs :

$$p(Y|X) = \sum_{\pi} p(\pi|x)$$
$$= \sum_{\pi} \prod_{t} p(\pi_t|x)$$

No need to sum over the possibly large number of paths π, it can be computed recursively.

Graphical representation from distill.pub



CTC cost efficient computation

Recursively compute $\alpha_{s,t}$ the probability assigned by the model at time t to the subsequence (extended with the blank) $y_{1:s}$

You end up with a computational graph through which the gradient can propagate.

CTC DECODING BY COMBINING THE ALTERNATIVES

Problem During inference, given an input x, what is the most probable **collapsed** labeling ? This is intractable.

Solution 1: best path decoding by selecting, at each time step, the output with the highest probability assigned by your model

$$\hat{y}(x) = \mathcal{B}(\text{argmax}_{\pi} p(\pi|x,\theta)) = \mathcal{B}(\text{argmax}_{\pi} \prod_{t} p(\pi_t|x,\theta))$$

But the same labeling can have many alignments and the probability can be spiky on one bad alignment.

Solution 2: beam search decoding taking care of the **blank** character (multiple paths may collapse to the same final labeling)

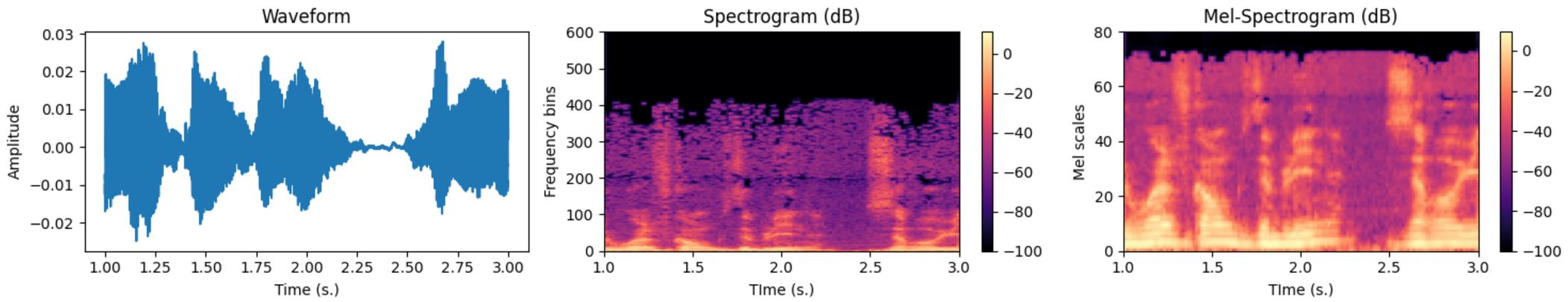
Possibility to introduce a language model to bias the decoding in favor of plausible words. See (Hannun et al., 2014) :

$$\text{argmax}_{y}(p(y|x)p_{LM}(y)^{\alpha}\text{wordcount}^{\beta}(y))$$

CTC EXAMPLE ON VOICE RECOGNITION

Problem Given a waveform, produce the transcript.

Example datasets : Librispeech (English, 1000 hours, Aligned), TED (English, 450 hours, Aligned), Mozilla common voice (Multi language, 2000 hours in English, 600 hours in French, unaligned)



Preprocessing of the waveform with spectrogram using STFT(win_size=25ms, win_step=15ms), with the transcript “Rue Wolfgang Doeblin, zéro huit, six cents Givet”

Note: you can contribute the open shared common voice dataset in one of the **60 languages** by either recording or validating (Ardila et al., 2020)!

Example model : end-to-end trainable Baidu DeepSpeech (v1,v2) (Hannun et al., 2014),(Amodei et al., 2015). See also the implementation of Mozilla DeepSpeech v2.

Note some authors introduced end-to-end trainable networks from the raw waveforms (Zeghidour, Usunier, Synnaeve, Collobert, & Dupoux, 2018).

DEEPSPEECH : ASR WITH CONV-BIGRU-CTC

Introduced in (Amodei et al., 2015) on English and Mandarin.

The English architecture involves :

- mel spectrograms
- 3 2D-convolutions in time and frequency with cliped ReLu
- 7 bidirectional GRU(1280)
- 1 FC with Batch Norm
- CTC loss, decoding with beam size 500

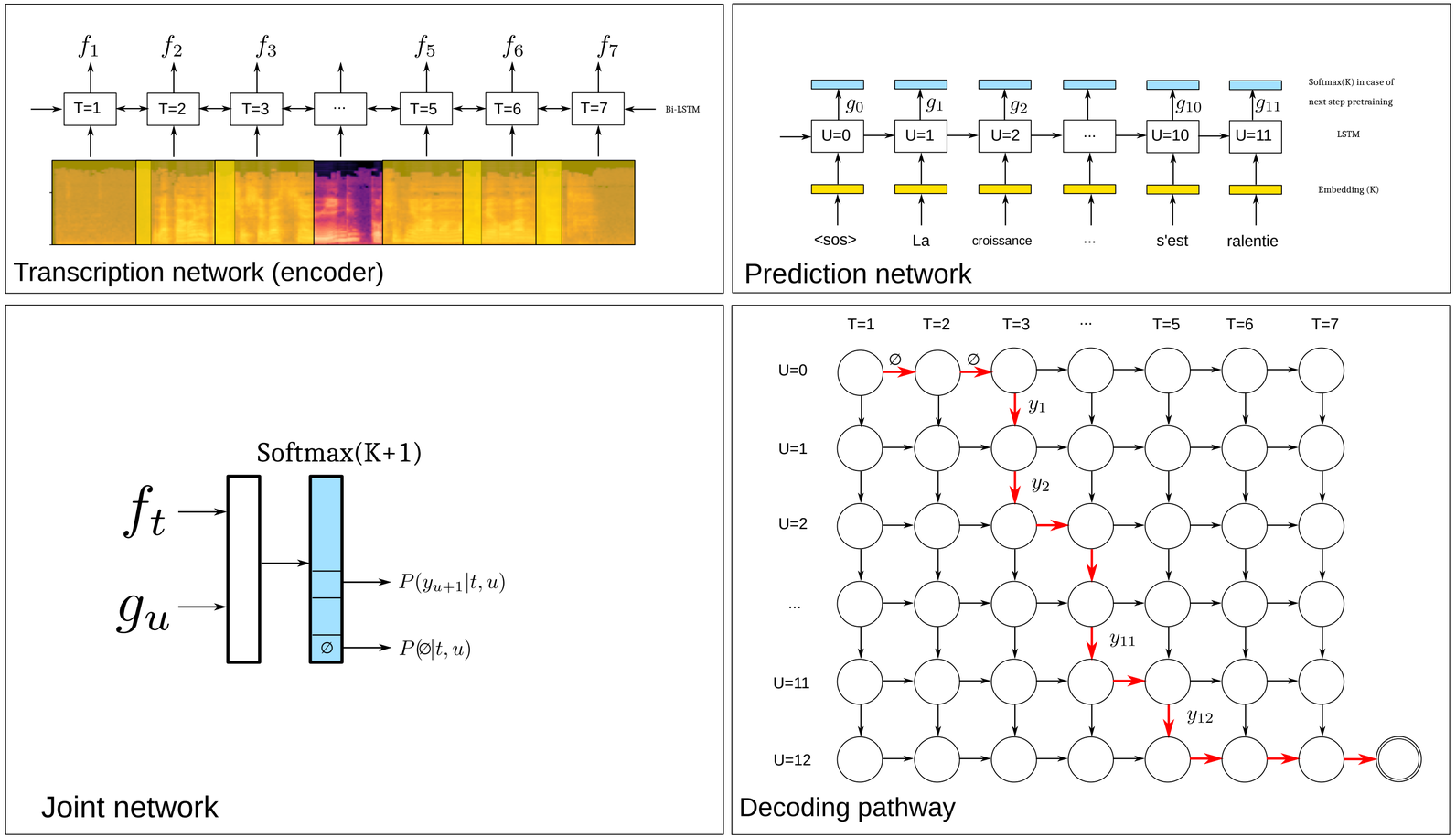
35 M. parameters

The training :

- dataset sizes up from 120hours up to 12.000 hours
- SGD with Nesterov momentum (0.99)
- gradient clipping to 400
- learning rate ($\approx 1e-4$), downscaled by 0.8 at every epoch
- during the 1st epoch, the samples are ordered by increasing duration (SortaGrad)
- data augmentation with noise added to the speech

CTC EXTENSION WITH SEQUENCE TRANSDUCTION (RNN-T)

Idea (Graves, 2012), (Graves et al., 2013) extended CTC to 1) cope with any T_y 2) make the prediction y_t dependent on previously generated outputs. Can produce from 0 to N output tokens per input time step.



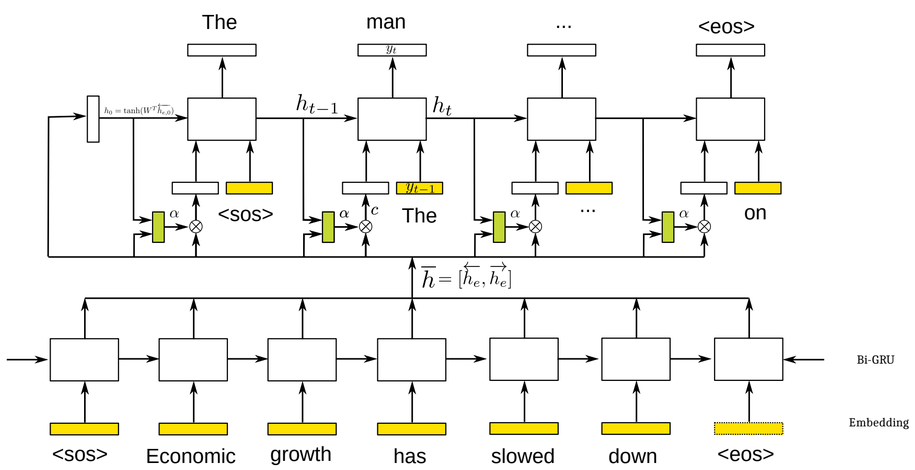
RNN-Transducer architecture

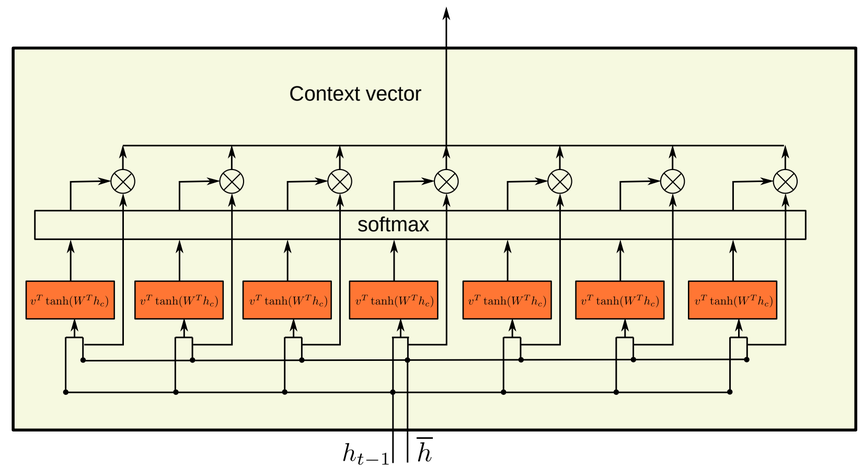
Can work online (stream based) contrary to seq2seq which encodes the complete input sequence (He et al., 2018).

ATTENTION BASED ENCODER-DECODER

GLOBAL ATTENTION

Idea Seq2Seq models are required to compress all the input sequence to a single hidden state which is challenged for long input sequences. What if the decoder could focus on part of the hidden states of the encoder ? Introduced in (Bahdanau et al., 2015).





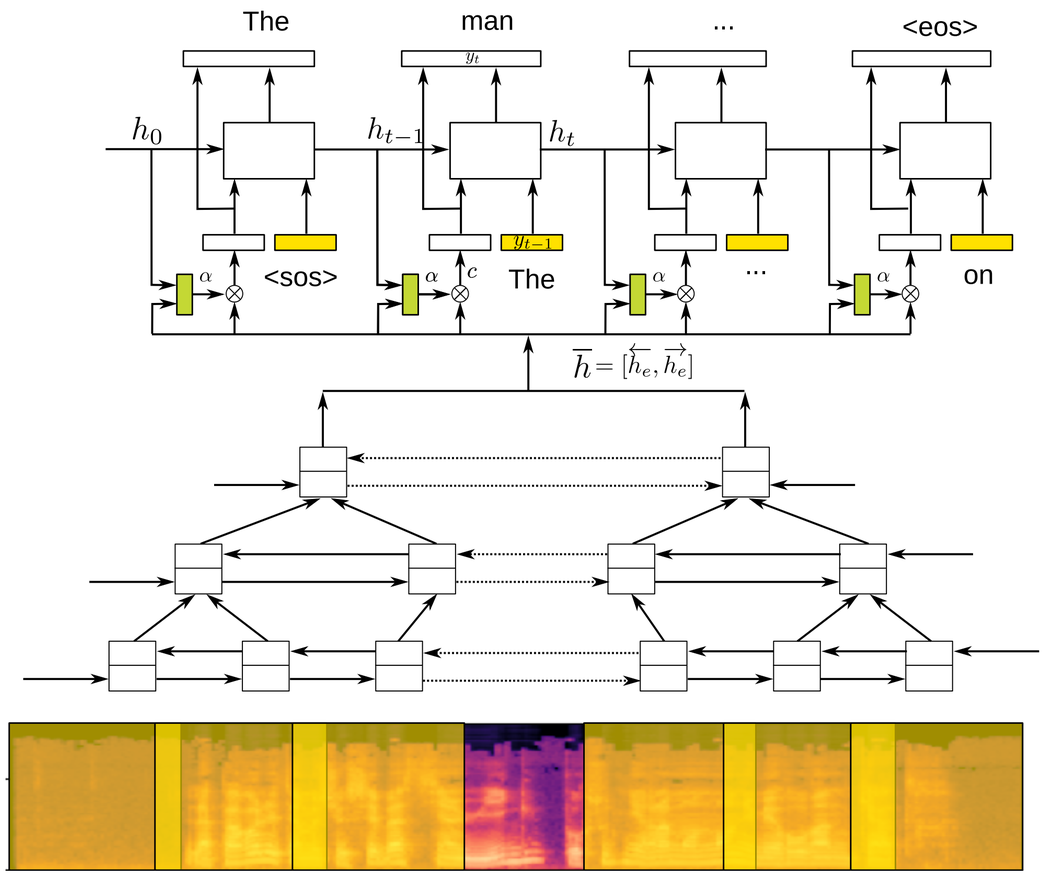
Bahdanau soft-alignement with global attention

Soft attention models the expected input alignments allowing to translate the output token at time t. It seeks to align the input w.r.t. the output.

See also distill.pub augmented-rnns

VOICE RECOGNITION / ASR : LISTEN ATTEND AND SPELL

Idea Apply the seq2seq encoder/decoder with soft attention on speech recognition (Chan, Jaitly, Le, & Vinyals, 2015),(Chiu et al., 2018)



Architecture

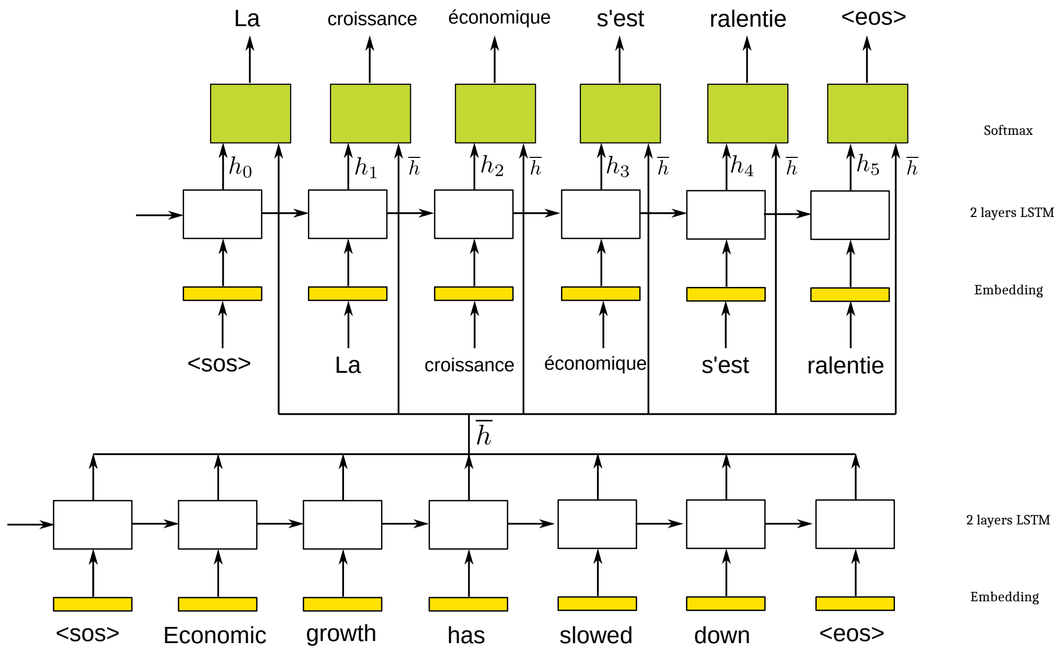
- log 80-mel spectrogram
- 5 (bi)-LSTM encoder
- 2 LSTM decoder
- distributed training,
- scheduled sampling
- label smoothing
- 4-head attention (multiple queries are computed from h_i)
- both the queries and keys are computed with a MLP

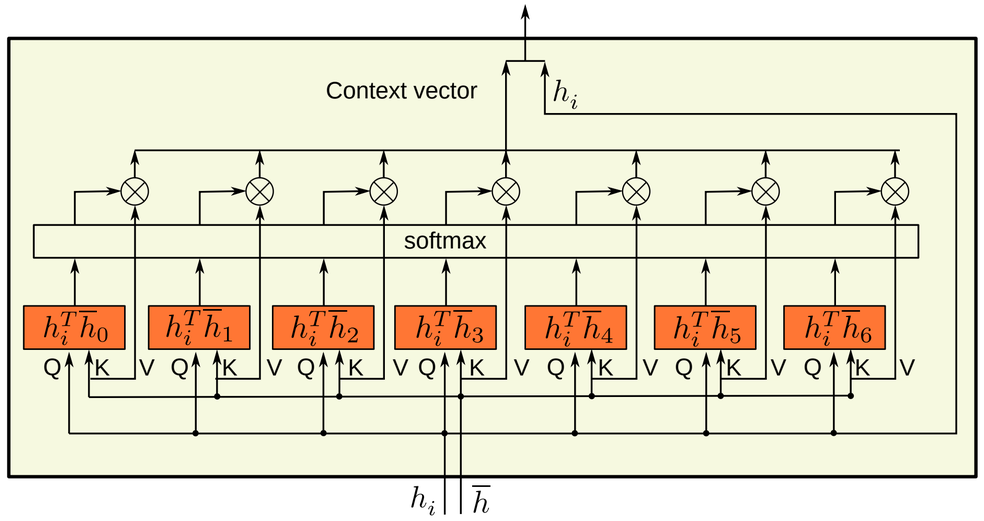
For Neural Machine Translation, see this series on attention based approaches. See also Google Neural Machine Translation (Wu et al., 2016)

CONTENT BASED ATTENTION

In (Luong, Pham, & Manning, 2015), several forms of attention have been explored. Their design does not feed the result of attention into the update of the decoder state.

The decoder hidden state is used a query to select part of the encoder hidden states. For multilayer encoder/decoder, they used only the states of the last layers.





Global attention with the Dot-Product attention.

More generally, you could score by matching a transformed query $Q = W_q^T h_i$ and a transformed key $K = W_k^T \overline{h_j}$, with $\text{score} = Q^T K = h_i^T W_{qk} \overline{h_j}$

ALL YOU NEED IS ATTENTION

Idea Feedforward self-attended encoder/decoder. Every input sequence element is encoded with its own self-attended context. Introduced in (Vaswani et al., 2017).

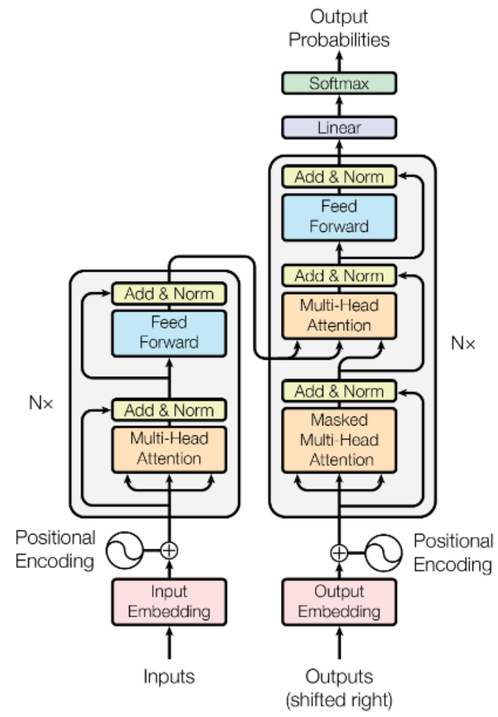
Encoder :

- feedforward
- word + position embedding
- self-attention : every word branch has a :
  - query : to compute its context
  - key : to match against the queries
  - value : the content to propagate

Fixed size position encoding ($\cos$, $\sin$).

Decoder:

- feedforward
- word + position embedding
- no time propagation of context
- access to all the encoder features
- input : previous token and position



Transformer network for sequence processing. Image from (Vaswani et al., 2017)

The idea of feedforward networks for sequence processing and position encoding is also used in (Gehring, Auli, Grangier, Yarats, & Dauphin, 2017).

BIBLIOGRAPHY

REFERENCES

Rather check the full online document references.pdf

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., … Zhu, Z. (2015). Deep Speech 2: End-to-End Speech Recognition in English and Mandarin. *arXiv:1512.02595 [Cs]*. Retrieved from http://arxiv.org/abs/1512.02595

Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., … Weber, G. (2020). Common Voice: A Massively-Multilingual Speech Corpus. *arXiv:1912.06670 [Cs]*. Retrieved from http://arxiv.org/abs/1912.06670

Arjovsky, M., Shah, A., & Bengio, Y. (2016). Unitary Evolution Recurrent Neural Networks. *arXiv:1511.06464 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1511.06464

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. *arXiv:1607.06450 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1607.06450

Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In.

Chan, W., Jaitly, N., Le, Q. V., & Vinyals, O. (2015). Listen, Attend and Spell. *arXiv:1508.01211 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1508.01211

Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., … Bacchiani, M. (2018). State-of-the-art Speech Recognition With Sequence-to-Sequence Models. *arXiv:1712.01769 [Cs, Eess, Stat]*. Retrieved from http://arxiv.org/abs/1712.01769

Cho, K., Merrienboer, B. van, Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1406.1078

Collobert, R., Puhrsch, C., & Synnaeve, G. (2016). Wav2Letter: An End-to-End ConvNet-based Speech Recognition System. *arXiv:1609.03193 [Cs]*. Retrieved from http://arxiv.org/abs/1609.03193

Dauphin, Y. N., Fan, A., Auli, M., & Grangier, D. (2017). Language modeling with gated convolutional networks. Retrieved from http://arxiv.org/abs/1612.08083

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional Sequence to Sequence Learning. *arXiv:1705.03122 [Cs]*. Retrieved from http://arxiv.org/abs/1705.03122

Gers, F. A., Schmidhuber, J. A., & Cummins, F. A. (2000). Learning to forget: Continual prediction with lstm. *Neural Comput.*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Graves, A. (2012). Sequence Transduction with Recurrent Neural Networks. *arXiv:1211.3711 [Cs, Stat]*. Retrieved from http://arxiv.org/abs/1211.3711

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on machine learning* (pp. 369–376). New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1143844.1143891

Graves, A., Mohamed, A.-r., & Hinton, G. (2013). Speech Recognition with Deep Recurrent Neural Networks. *arXiv:1303.5778 [Cs]*. Retrieved from http://arxiv.org/abs/1303.5778

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/TNNLS.2016.2582924

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., … Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition. *arXiv:1412.5567 [Cs]*. Retrieved from http://arxiv.org/abs/1412.5567

He, Y., Sainath, T. N., Prabhavalkar, R., McGraw, I., Alvarez, R., Zhao, D., … Gruenstein, A. (2018). Streaming End-to-end Speech Recognition For Mobile Devices. *arXiv:1811.06621 [Cs]*. Retrieved from http://arxiv.org/abs/1811.06621

Henaff, M., Szlam, A., & LeCun, Y. (2016). Recurrent Orthogonal Networks and Long-Memory Tasks, 9.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Kam-Chuen Jim, Giles, C. L., & Horne, B. G. (1996). An analysis of noise in recurrent neural networks: Convergence and generalization. *IEEE Transactions on Neural Networks*, *7*(6), 1424–1438. https://doi.org/10.1109/72.548170

Krueger, D., Maharaj, T., Kramár, J., Pezeshki, M., Ballas, N., Ke, N. R., … Pal, C. (2017). ZONEOUT: REGULARIZING RNNS BY RANDOMLY PRESERVING HIDDEN ACTIVATIONS, 11.

Le, Q. V., Jaitly, N., & Hinton, G. E. (2015). A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *arXiv:1504.00941 [Cs]*. Retrieved from http://arxiv.org/abs/1504.00941

Luong, M.-T., Pham, H., & Manning, C. D. (2015). Effective Approaches to Attention-based Neural Machine Translation. *arXiv:1508.04025 [Cs]*. Retrieved from http://arxiv.org/abs/1508.04025

Moon, T., Choi, H., Lee, H., & Song, I. (2015). RNNDROP: A novel dropout for rnns in asr. In *2015 ieee workshop on automatic speech recognition and understanding (asru)* (pp. 65–70). https://doi.org/10.1109/ASRU.2015.7404775

Pascanu, R., Dauphin, Y. N., Ganguli, S., & Bengio, Y. (2014). On the saddle point problem for non-convex optimization. *arXiv:1405.4604 [Cs]*. Retrieved from http://arxiv.org/abs/1405.4604

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30thInternational Conference on Machine Learning* (p. 9).

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. https://doi.org/10.1109/78.650093

Sutskever, I. (2013). *Training recurrent neural networks* (PhD thesis). University of Toronto, CAN.

Sutskever, I., Martens, J., & Hinton, G. (2011). Generating Text with Recurrent Neural Networks, 8.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *arXiv:1409.3215 [Cs]*. Retrieved from http://arxiv.org/abs/1409.3215

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is All you Need, 11.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(4), 652–663. https://doi.org/10.1109/TPAMI.2016.2587640

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., & Lang, K. J. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *37*(3), 328–339. https://doi.org/10.1109/29.21701

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560. Retrieved from https://www.bibsonomy.org/bibtex/25ea3485ce75778e802cd8466cd7ffa69/joachimagne

Williams, R. J., & Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, *2*(4), 490–501. https://doi.org/10.1162/neco.1990.2.4.490

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144 [Cs]*. Retrieved from http://arxiv.org/abs/1609.08144

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., … Bengio, Y. (2016). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv:1502.03044 [Cs]*. Retrieved from http://arxiv.org/abs/1502.03044

Zeghidour, N., Usunier, N., Synnaeve, G., Collobert, R., & Dupoux, E. (2018). End-to-End Speech Recognition from the Raw Waveform. In *Interspeech 2018* (pp. 781–785). ISCA. https://doi.org/10.21437/Interspeech.2018-2414