Sequence Learning

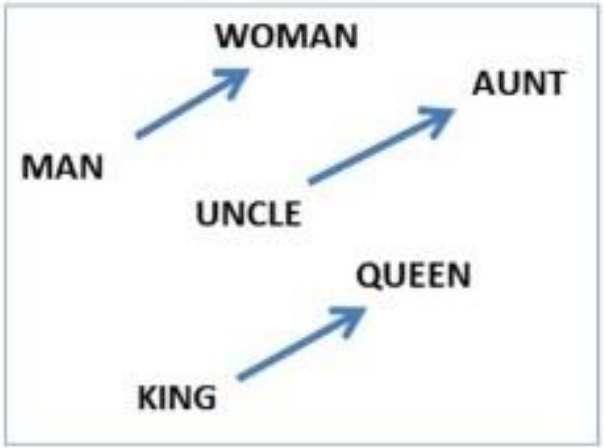
Grigory Sapunov  CTO / Intento

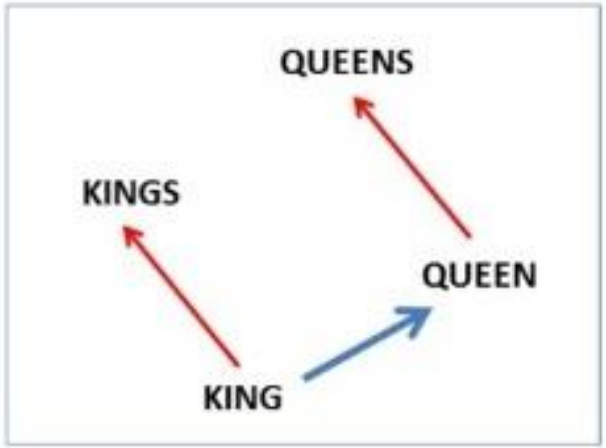
HSE / #12 / 10.12.2019

Multimodal Learning

Example: Semantic Spaces (word2vec, GloVe)

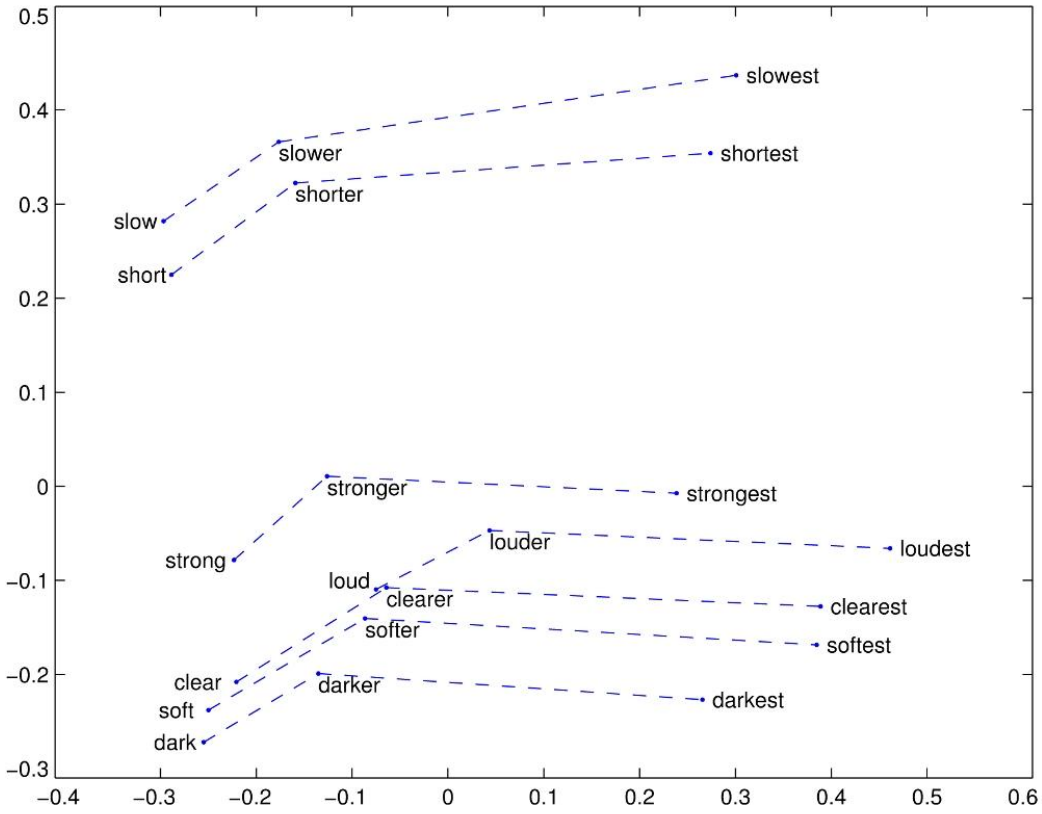
vector('king') - vector('man') + vector('woman') = vector('queen')





https://code.google.com/archive/p/word2vec/

Example: Semantic Spaces (word2vec, GloVe)



http://nlp.stanford.edu/projects/glove/

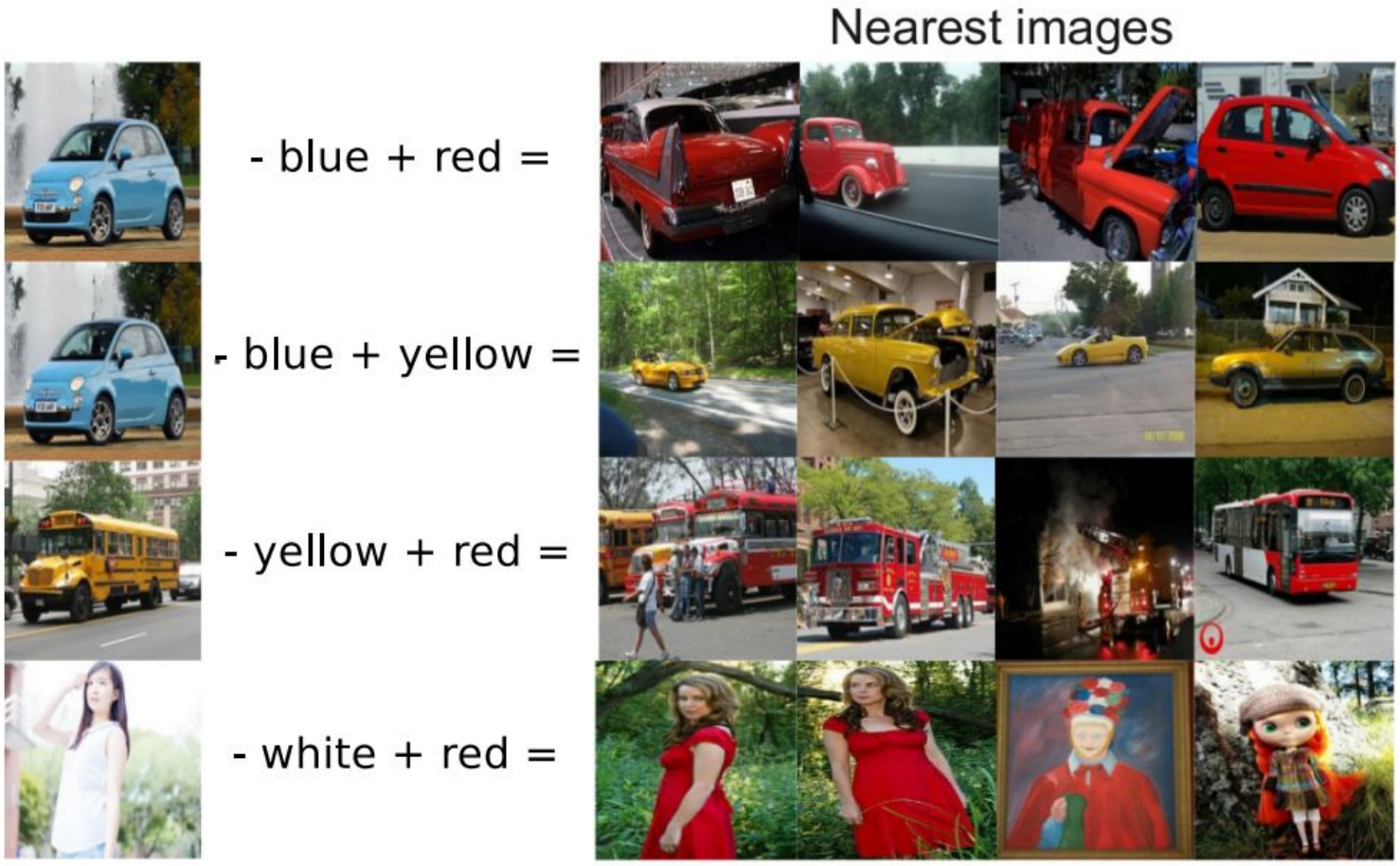
Encoding semantics

Using word2vec instead of word indexes allows you to better deal with the word meanings (e.g. no need to enumerate all synonyms because their vectors are already close to each other).

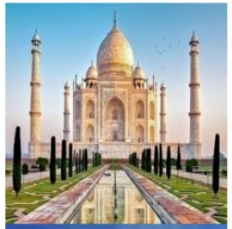
But the naive way to work with word2vec vectors still gives you a “bag of words” model, where phrases “The man killed the tiger” and “The tiger killed the man” are equal.

Need models which pay attention to the word ordering: paragraph2vec, sentence embeddings (using RNN/LSTM), even World2Vec (LeCunn @CVPR2015).

Example: More multi-modal learning



Nearest Images



- day + night =

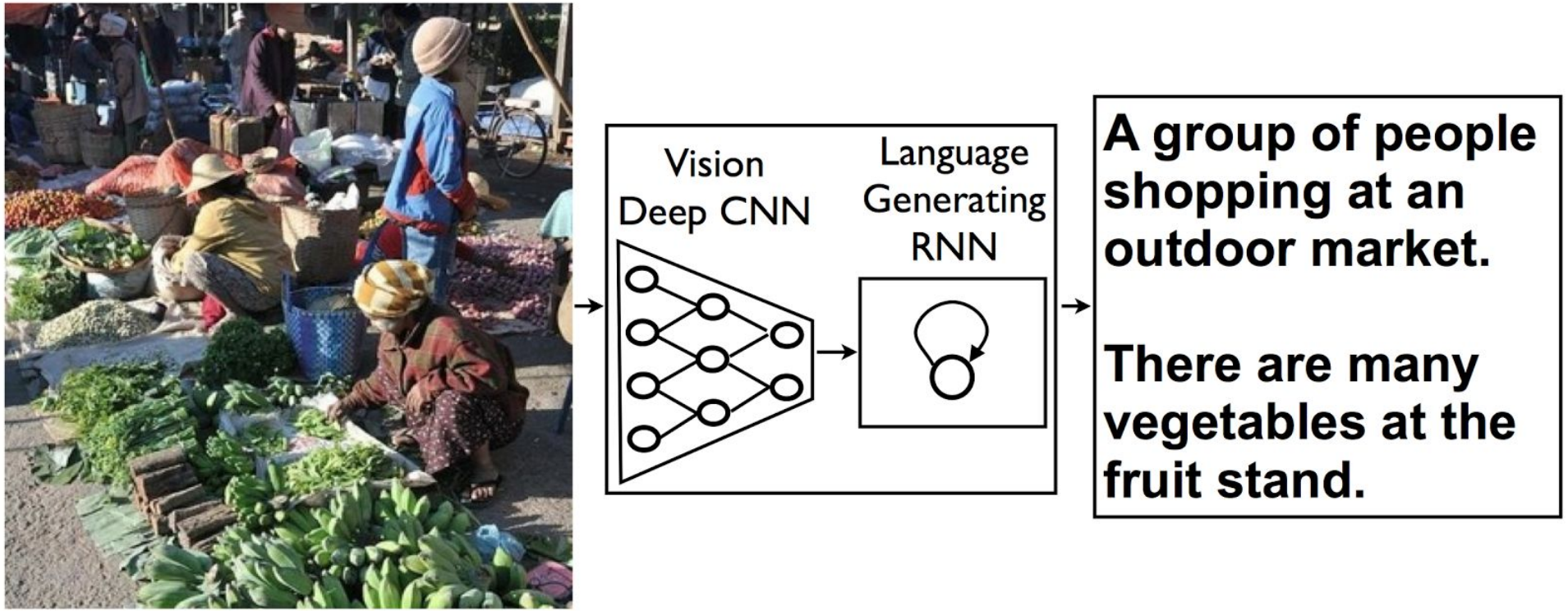
- flying + sailing =

- bowl + box =

- box + bowl =

(Kiros, Salakhutdinov, Zemel, TACL 2015)

Image caption generation



http://arxiv.org/abs/1411.4555 “Show and Tell: A Neural Image Caption Generator”

Human: "A group of men playing Frisbee in the park."
Computer model: "A group of young people playing a game of Frisbee."

Image description (image captioning)

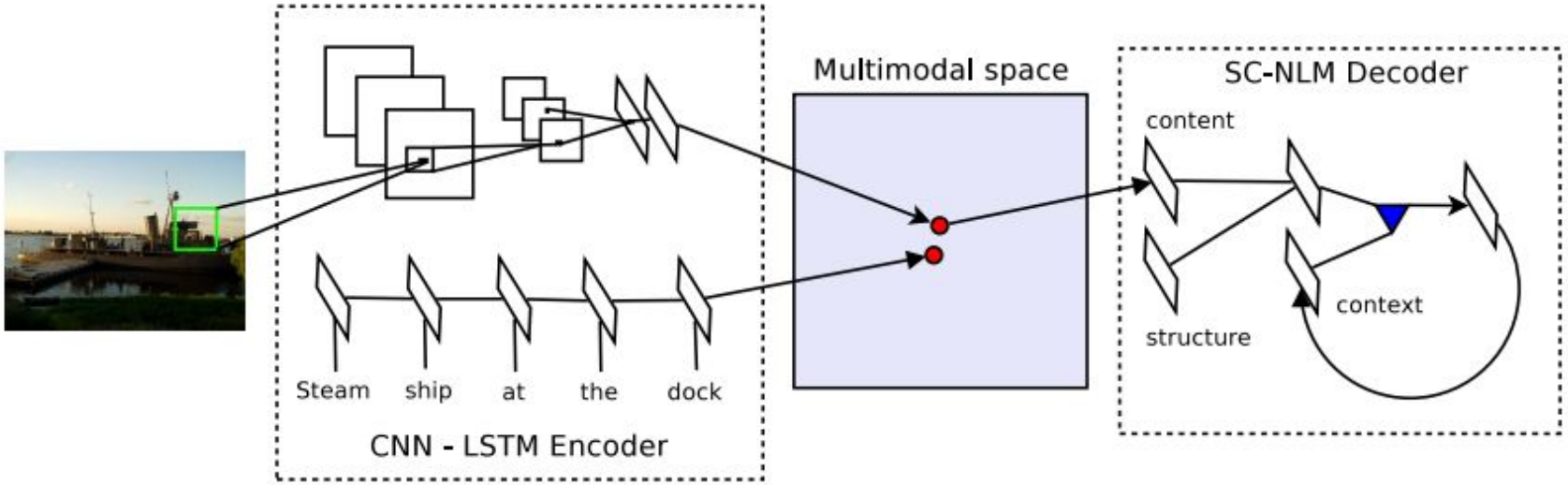
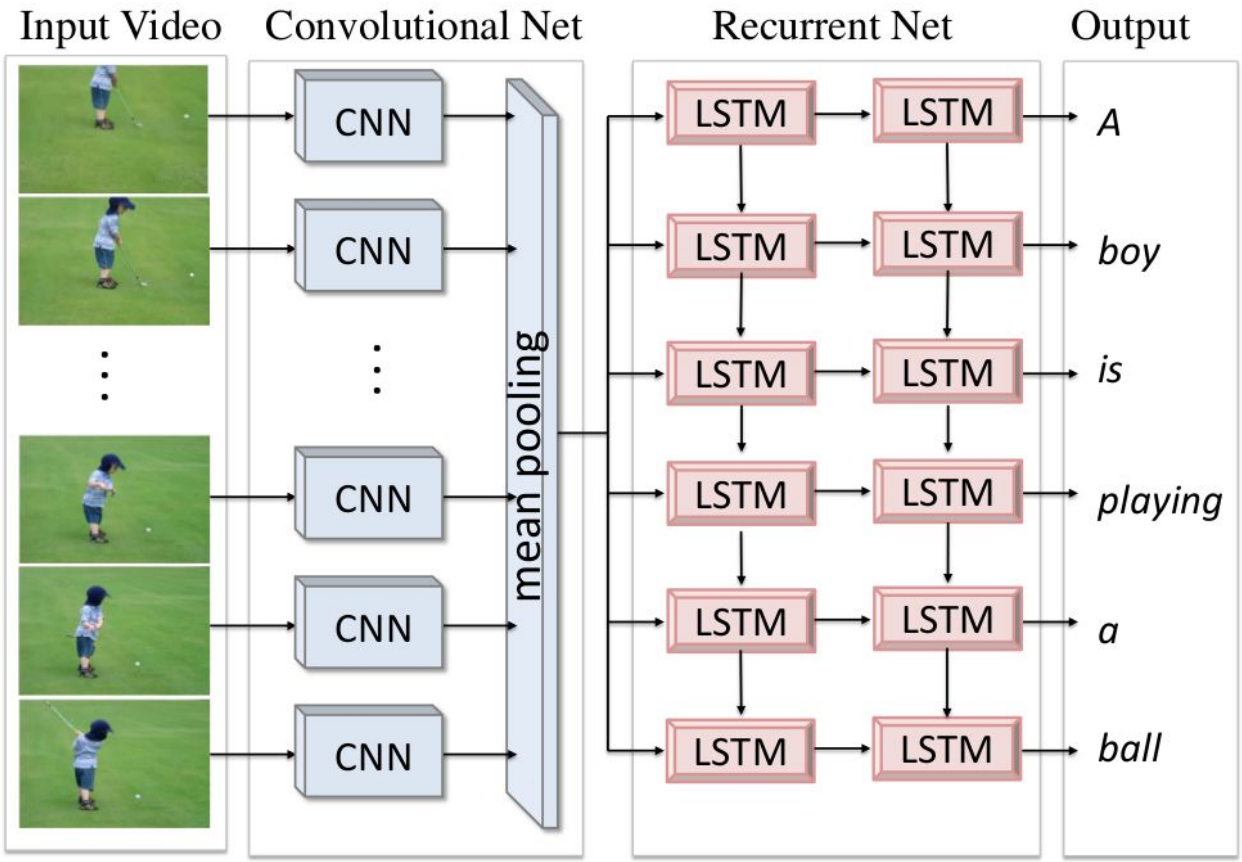


Figure 2: **Encoder:** A deep convolutional network (CNN) and long short-term memory recurrent network (LSTM) for learning a joint image-sentence embedding. **Decoder:** A new neural language model that combines structure and content vectors for generating words one at a time in sequence.

http://arxiv.org/abs/1411.2539 Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models

Video description

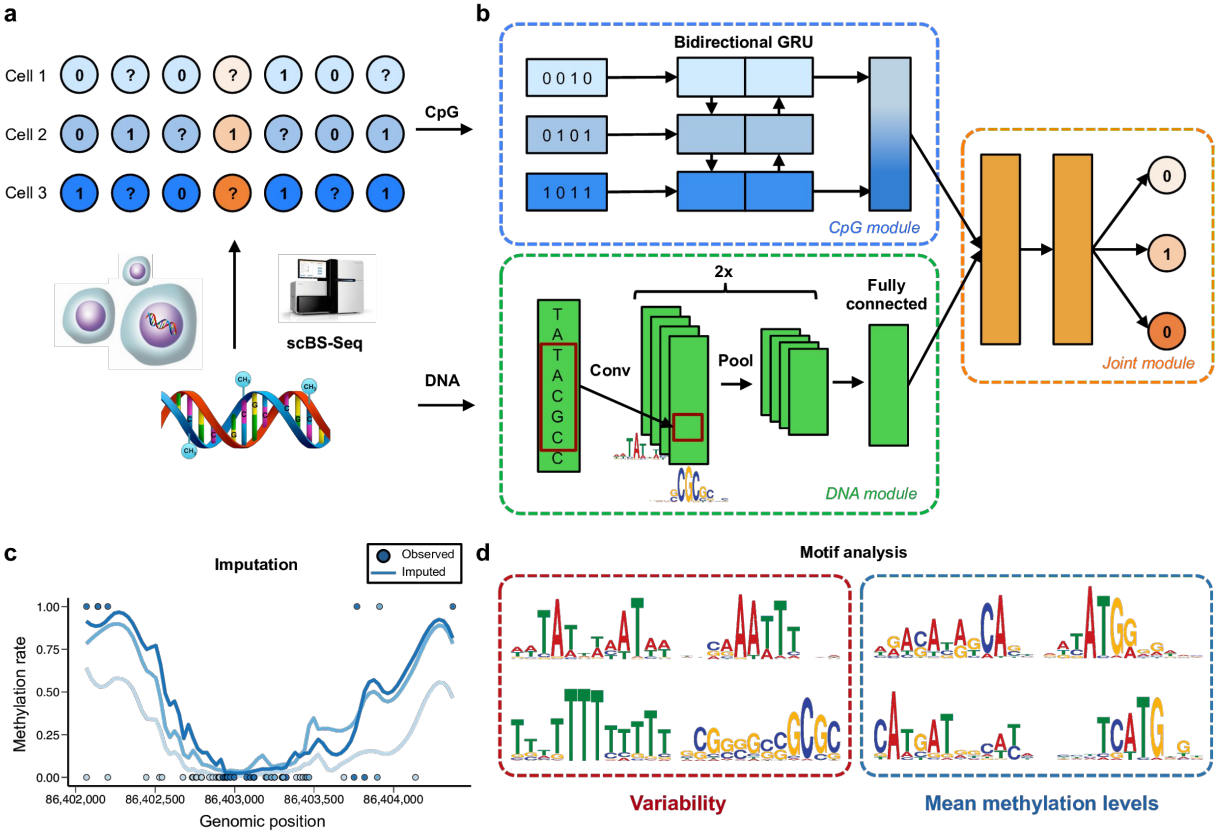


https://www.cs.utexas.edu/~vsub/

Example: Image generation by text

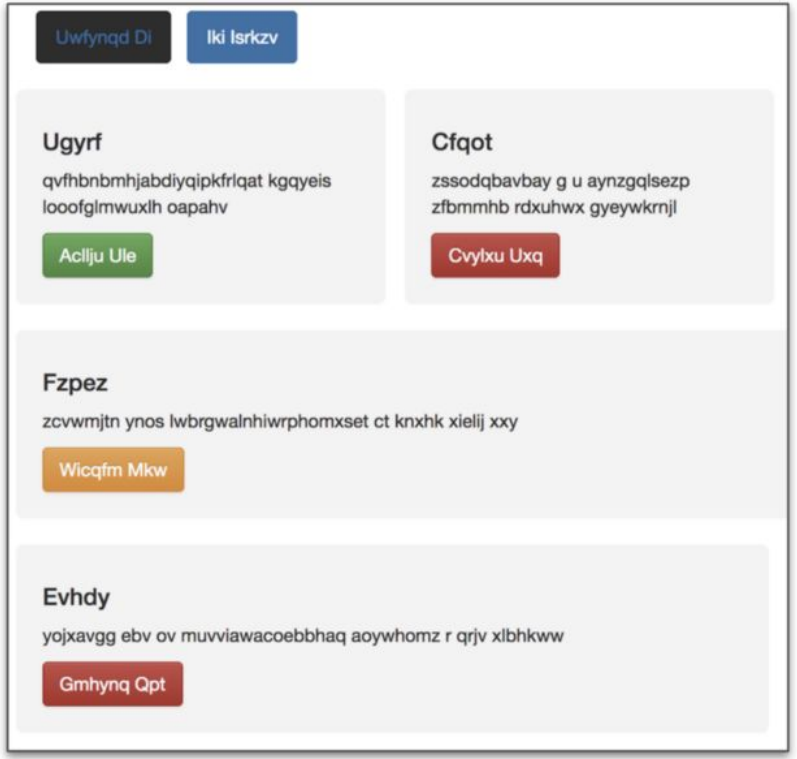
StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks, https://arxiv.org/abs/1612.03242

Mixing different bioinformatics signals



Accurate prediction of single-cell DNA methylation states using deep learning
http://biorxiv.org/content/early/2017/02/01/055715

SketchCode: Go from idea to HTML in 5 seconds



Automated front-end development using deep learning

https://blog.insightdatascience.com/automated-front-end-development-using-deep-learning-3169dd086e82

Example: Code generation by image

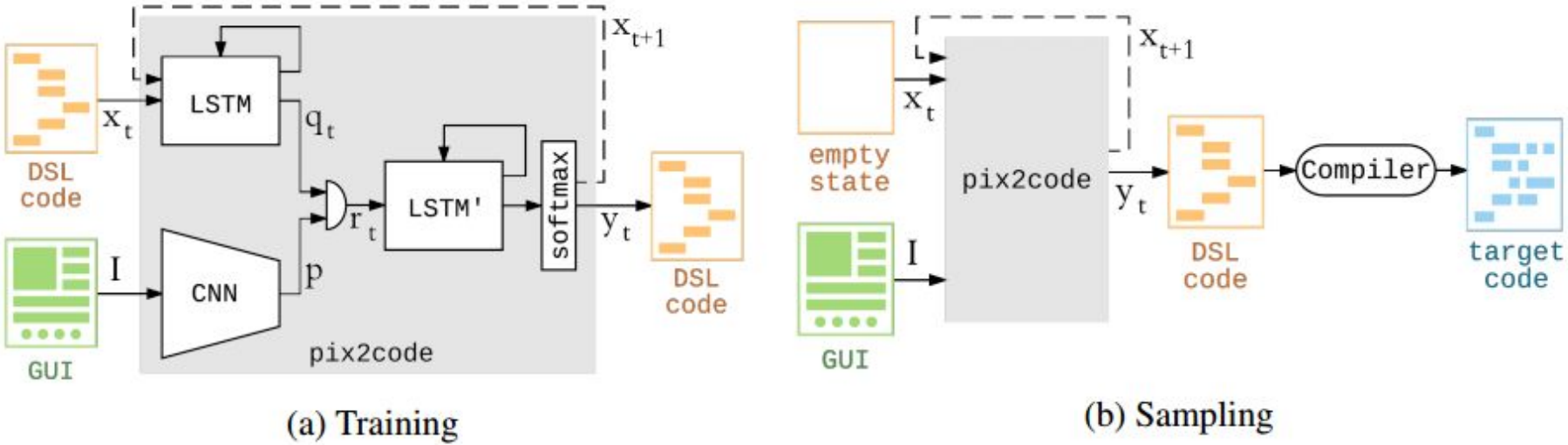
(a) iOS GUI screenshot

```
stack {
    row {
        label, switch
    }
    row {
        label, btn-add
    }
    row {
        label, slider, label
    }
    row {
        img, label
    }
}
footer {
    btn-more, btn-contact, btn-search, btn-download
}
```

(b) Code describing the GUI written in our DSL

pix2code: Generating Code from a Graphical User Interface Screenshot,
https://arxiv.org/abs/1705.07962

Example: Code generation by image



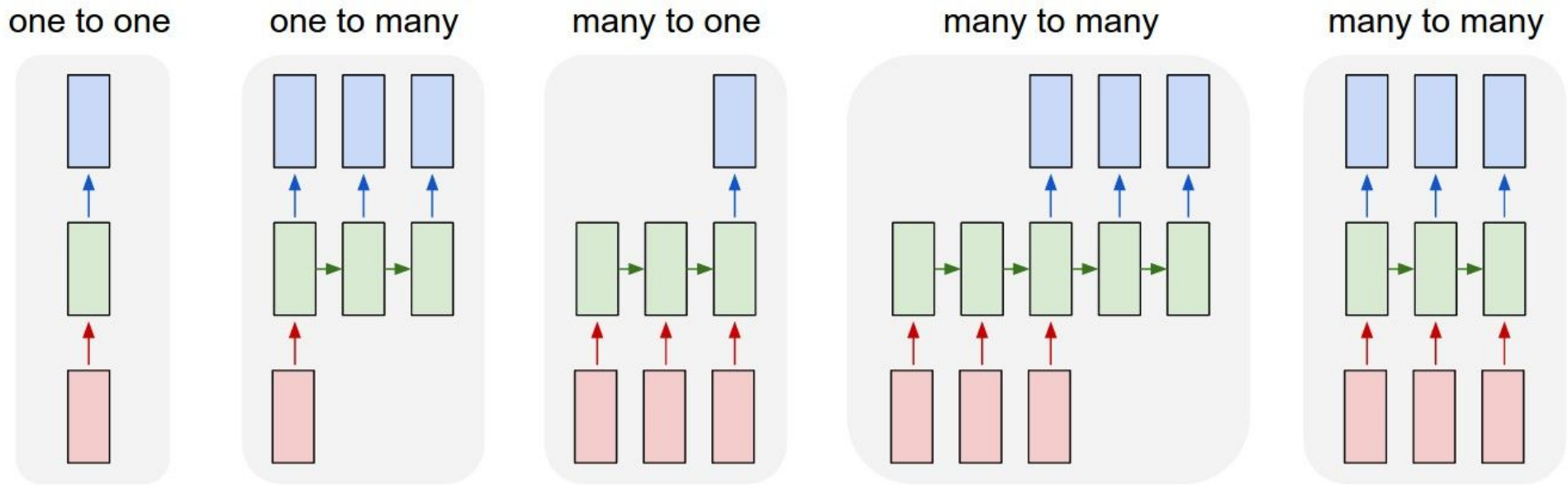
pix2code: Generating Code from a Graphical User Interface Screenshot,
https://arxiv.org/abs/1705.07962

Resources

- Image captioning TF example: https://www.tensorflow.org/tutorials/text/image_captioning

Sequence Learning / seq2seq

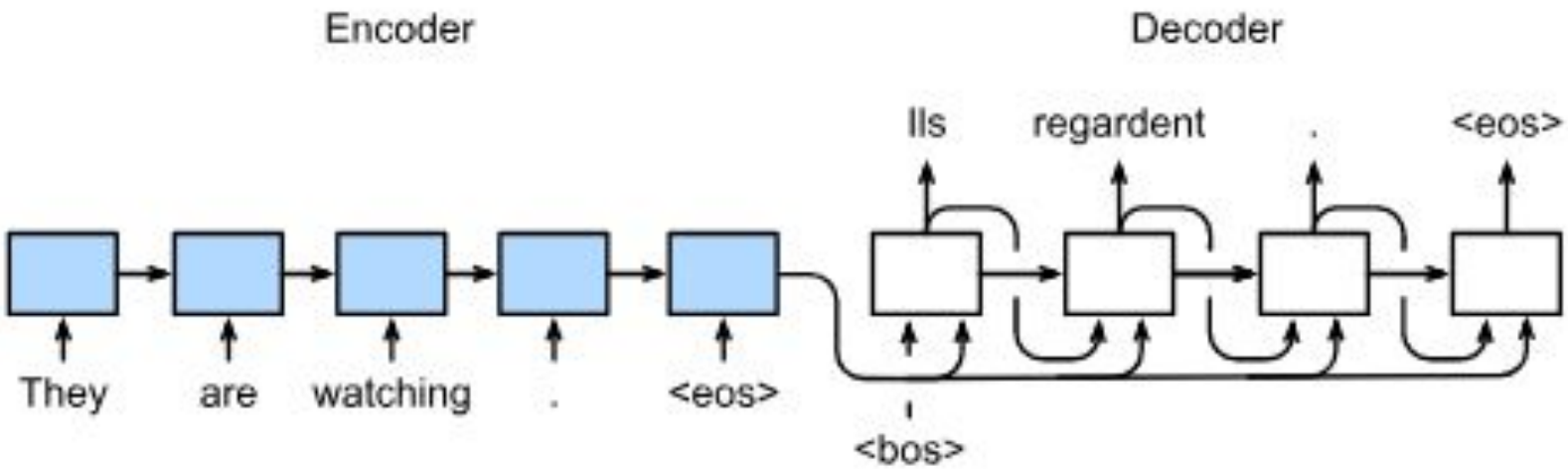
Sequence to Sequence Learning (seq2seq)



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Encoder-Decoder

Encoder-Decoder architecture



http://en.diveintodeeplearning.org/chapter_natural-language-processing/seq2seq.html

Encoder-Decoder architecture

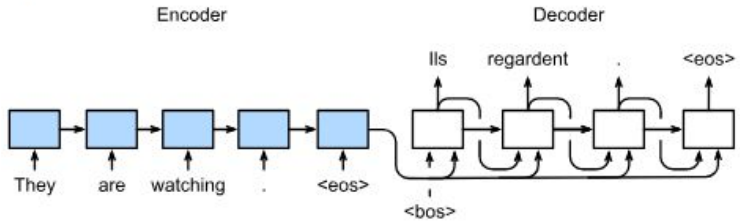
The encoder transforms the hidden state of each time step into **context variables** through custom function ***q***:

$$\boldsymbol{c} = q(\boldsymbol{h}_1, \ldots, \boldsymbol{h}_T)$$

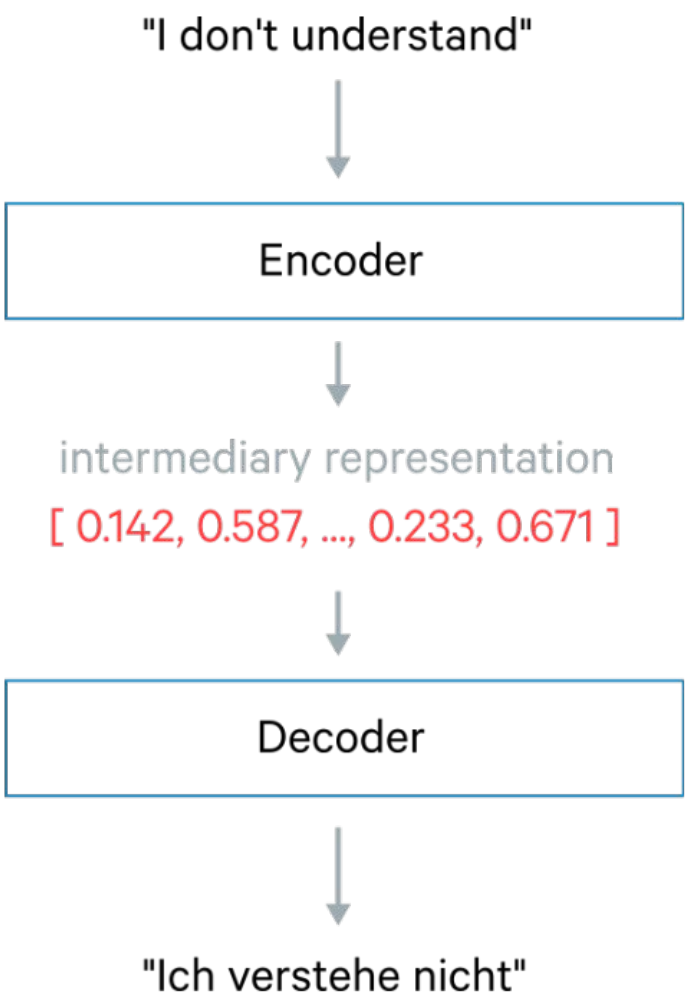
For example, when we select $q(\boldsymbol{h}_1,\ldots,\boldsymbol{h}_T)=\boldsymbol{h}_T$, the context variable is the hidden state of input sequence $\boldsymbol{h}_T$ for the final time step.

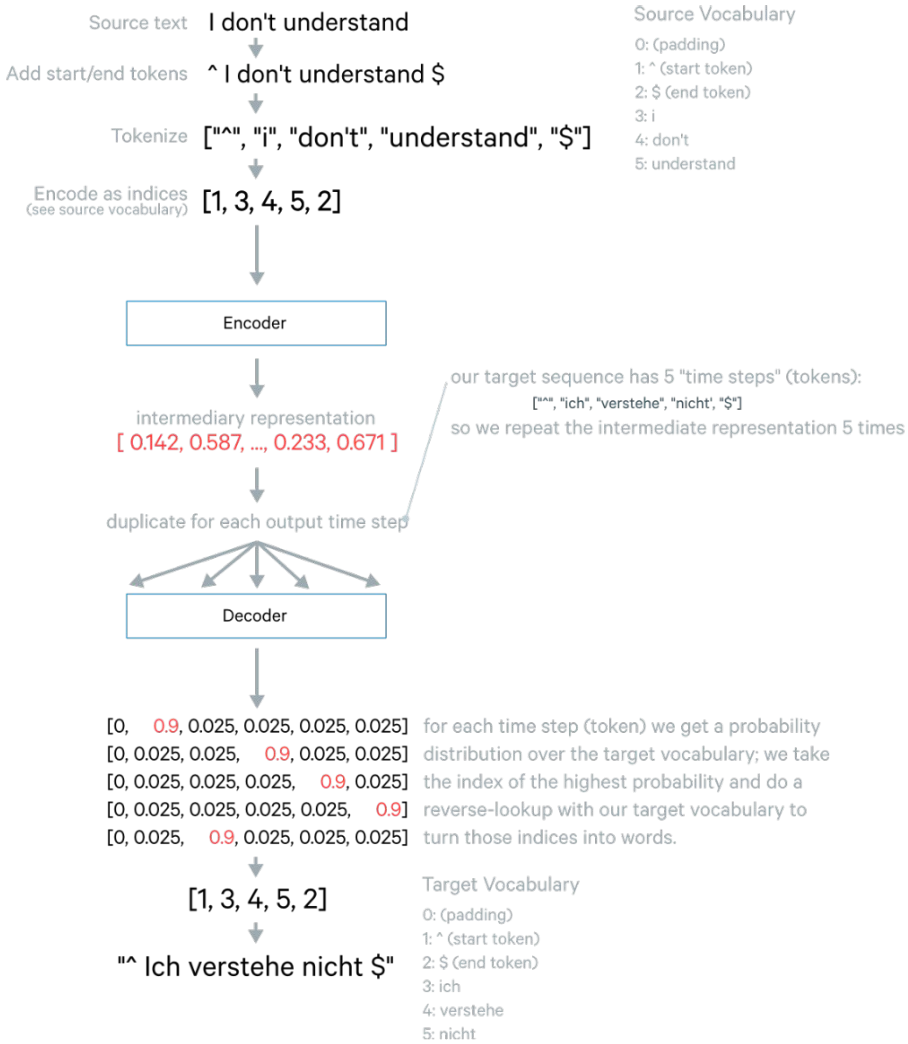
The transformation of the decoder's hidden layer:

$$\boldsymbol{s}_{t'} = g(y_{t'-1}, \boldsymbol{c}, \boldsymbol{s}_{t'-1})$$



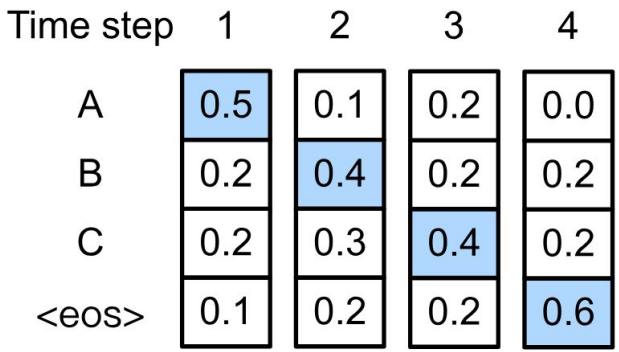
http://en.diveintodeeplearning.org/chapter_natural-language-processing/seq2seq.html


"I don't understand"
Encoder
intermediary representation
[0.142, 0.587, ..., 0.233, 0.671]
Decoder
"Ich verstehe nicht"

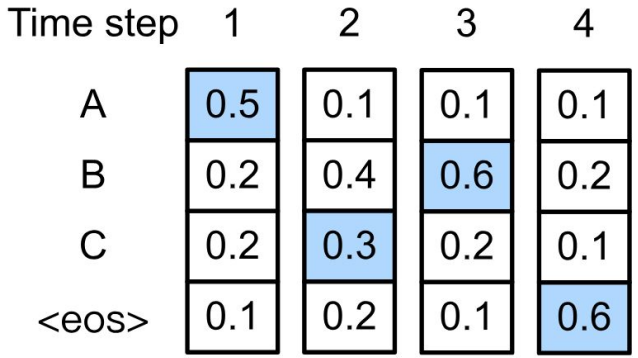

Source text
I don't understand
Add start/end tokens
^ I don't understand $
Tokenize
["^", "i", "don't", "understand", "$"]
Encode as indices
(see source vocabulary)
[1, 3, 4, 5, 2]
Source Vocabulary
0: (padding)
1: ^ (start token)
2: $ (end token)
3: i
4: don't
5: understand
Encoder
intermediary representation
[0.142, 0.587, ..., 0.233, 0.671]
duplicate for each output time step
our target sequence has 5 "time steps" (tokens):
["^", "ich", "verstehe", "nicht', "$"]
so we repeat the intermediate representation 5 times
Decoder
[0, 0.9, 0.025, 0.025, 0.025, 0.025]
[0, 0.025, 0.025, 0.9, 0.025, 0.025]
[0, 0.025, 0.025, 0.025, 0.9, 0.025]
[0, 0.025, 0.025, 0.025, 0.025, 0.9]
[0, 0.025, 0.9, 0.025, 0.025, 0.025]
for each time step (token) we get a probability distribution over the target vocabulary; we take the index of the highest probability and do a reverse-lookup with our target vocabulary to turn those indices into words.
[1, 3, 4, 5, 2]
"^ Ich verstehe nicht $"
Target Vocabulary
0: (padding)
1: ^ (start token)
2: $ (end token)
3: ich
4: verstehe
5: nicht

Decoding: Greedy search

Greedy search is not optimal. Exhaustive Search is not possible.



P = 0.5×0.4×0.4×0.6 = 0.048

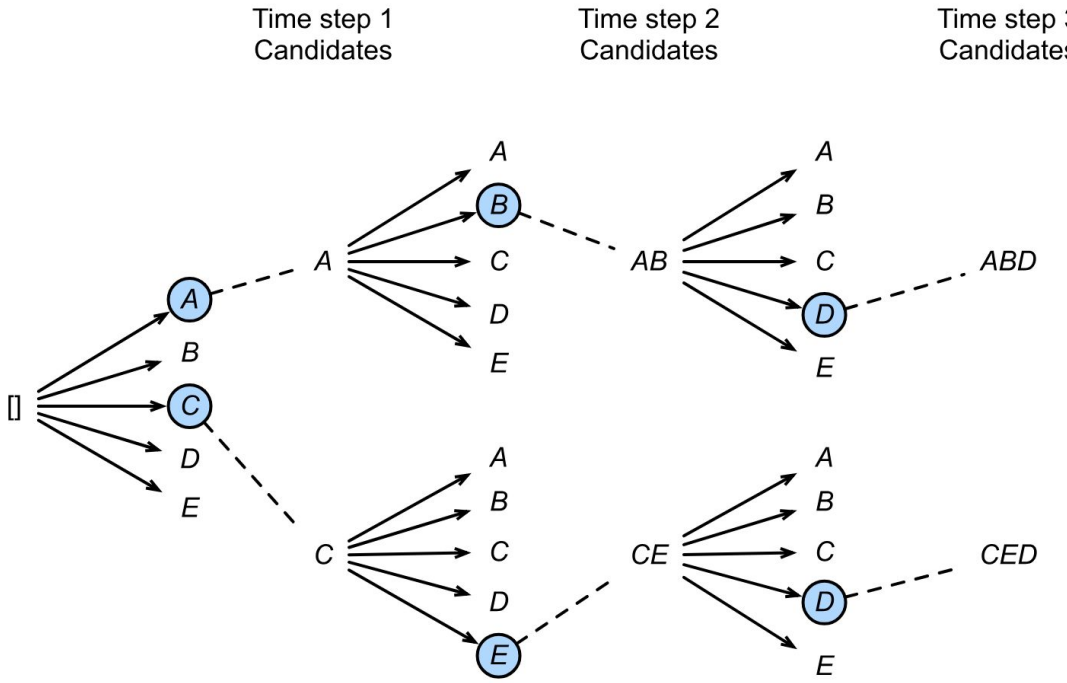


P = 0.5×0.3×0.6×0.6 = 0.054

http://en.diveintodeeplearning.org/chapter_natural-language-processing/beam-search.html

Decoding: Beam search

Beam search can give better solutions. Hyper-parameter: beam size.



http://en.diveintodeeplearning.org/chapter_natural-language-processing/beam-search.html

Neuraltalk2

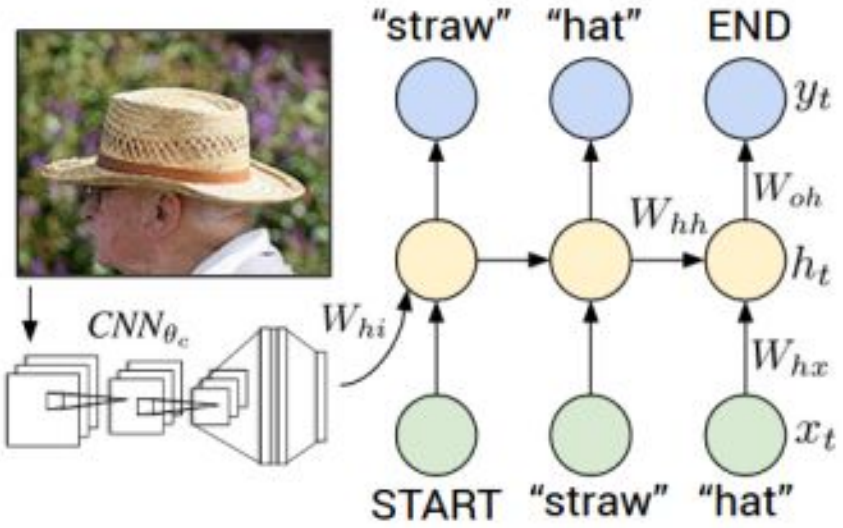


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

http://cs.stanford.edu/people/karpathy/deepimagesent/

Deep Visual-Semantic Alignments for Generating Image Descriptions

Neuraltalk2

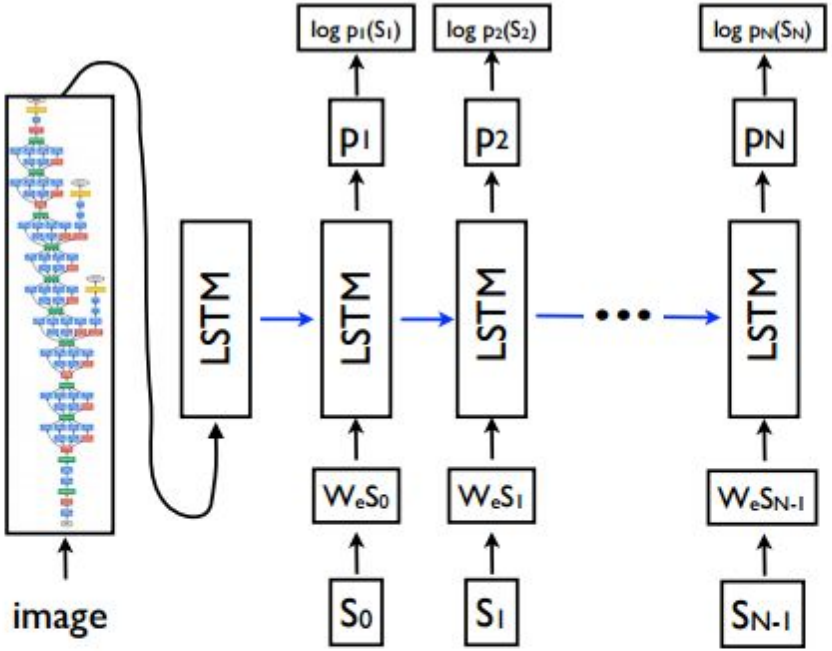


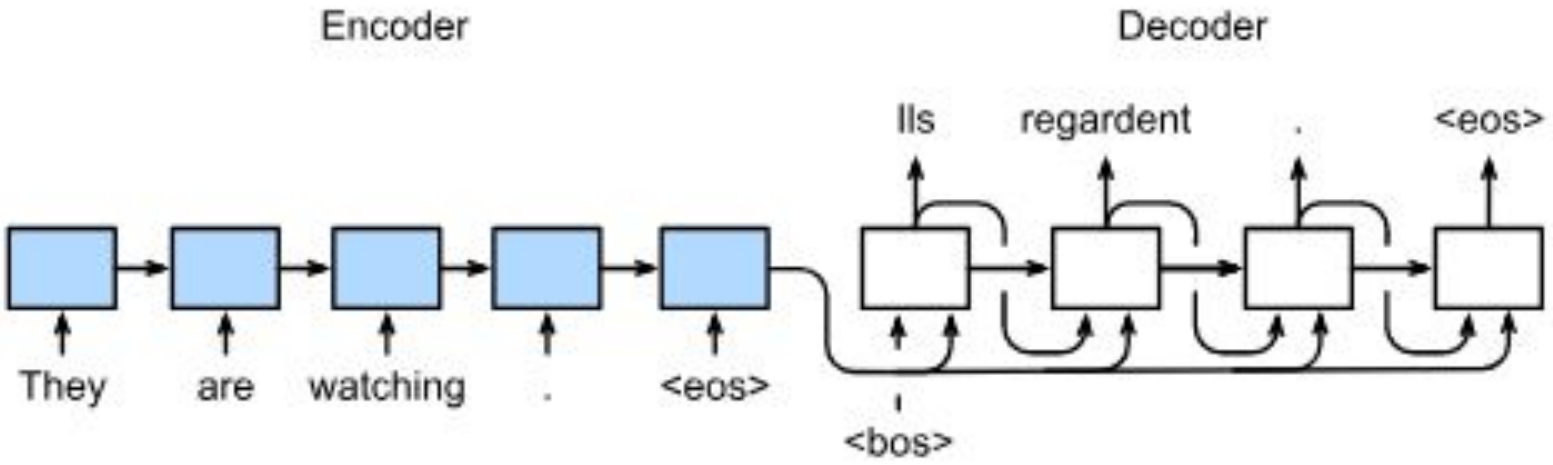
Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

https://arxiv.org/abs/1411.4555 Show and Tell: A Neural Image Caption Generator

Attention mechanisms

Encoder-Decoder shortcomings

Encoder-Decoder can be applied to N-to-M sequence, yet an Encoder reads and encodes a source sentence into a fixed-length vector. Is one hidden state really enough? A neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector.



Encoder-Decoder with Attention

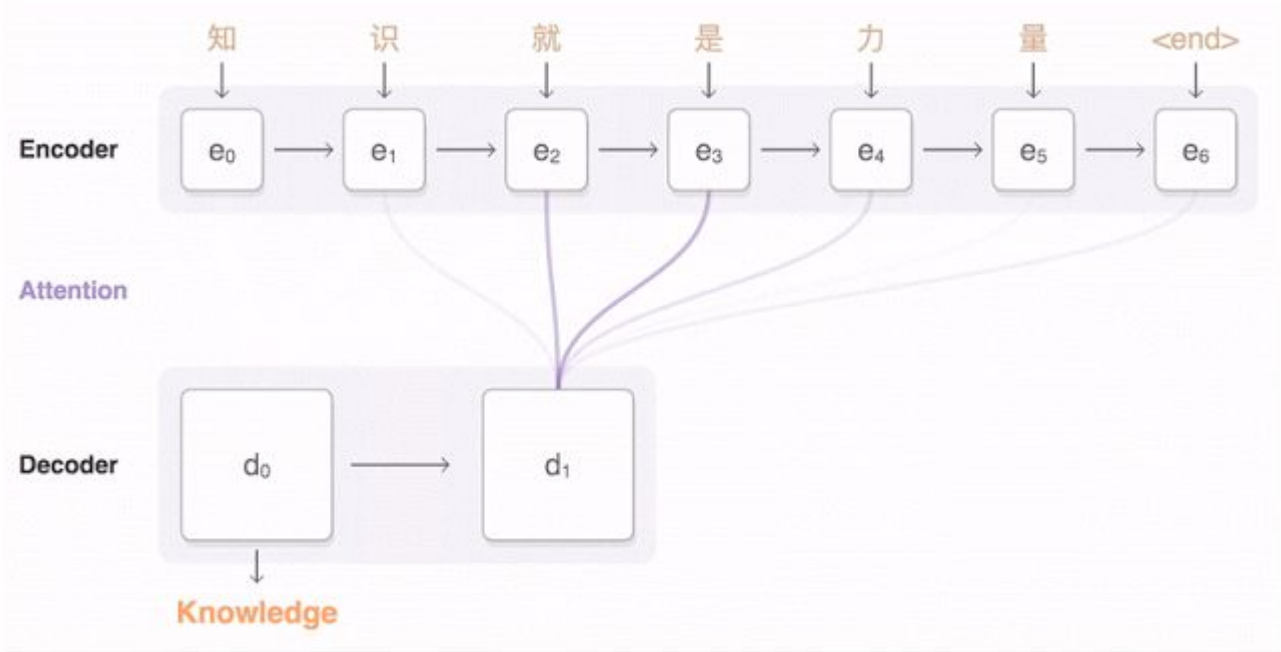
Attention Mechanism allows the decoder to attend to different parts of the source sentence at each step of the output generation.

Instead of encoding the input sequence into a single fixed context vector, we let the model learn how to generate a context vector for each output time step. That is we let the model learn what to attend based on the input sentence and what it has produced so far.

https://hackernoon.com/attention-mechanism-in-neural-network-30aaf5e39512

Encoder-Decoder with Attention

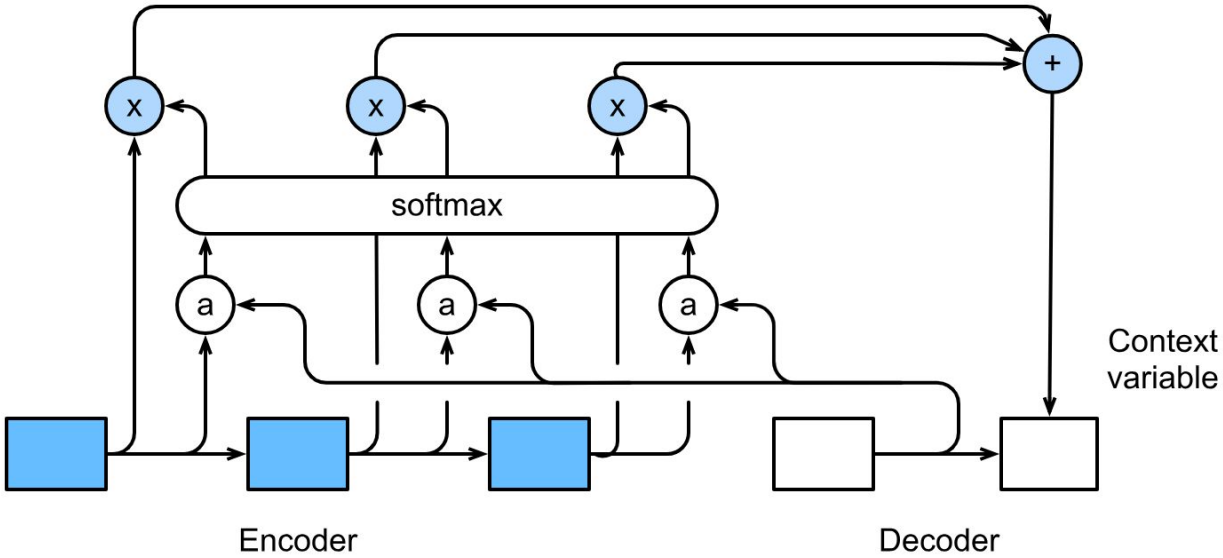
Attention Mechanism allows the decoder to attend to different parts of the source sentence at each step of the output generation.



https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

Attention Mechanism

The attention mechanism obtains the context variable by weighting the hidden state of all time steps of the encoder.

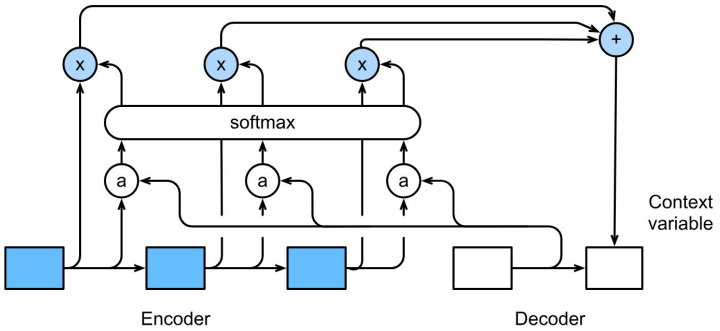


http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

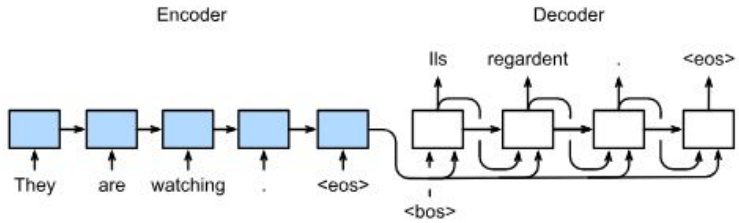
Now the hidden state of the decoder at time step ***t'*** can be rewritten as:

$$\boldsymbol{s}_{t'} = g(\boldsymbol{y}_{t'-1}, \boldsymbol{c}_{t'}, \boldsymbol{s}_{t'-1})$$



Compare to simple encoder-decoder:

$$\boldsymbol{s}_{t'} = g(y_{t'-1}, \boldsymbol{c}, \boldsymbol{s}_{t'-1})$$



http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

The context variable of the decoder at time step ***t'*** is the weighted average on all the hidden states of the encoder ($\boldsymbol{h}_t$).

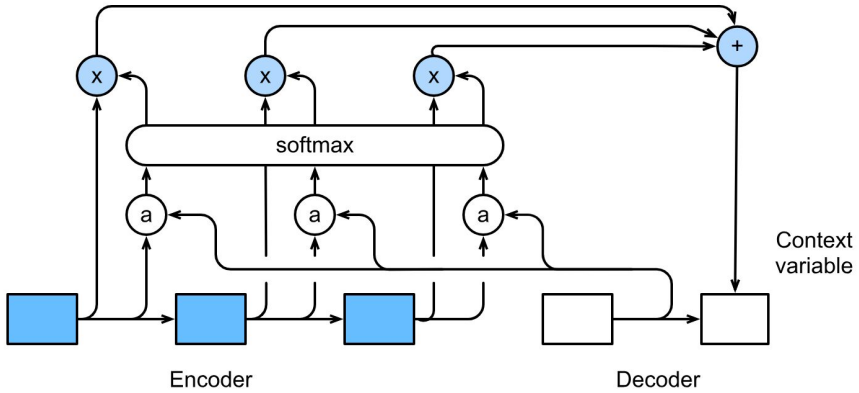
$\boldsymbol{s}_{t'-1}$: decoder's hidden state at that time step ***t'−1***

$$\boldsymbol{c}_{t'} = \sum_{t=1}^{T} \alpha_{t't} \boldsymbol{h}_t$$

$$\alpha_{t't} = \frac{\exp(e_{t't})}{\sum_{k=1}^{T} \exp(e_{t'k})}, \quad t = 1, \ldots, T$$

$$e_{t't} = a(\boldsymbol{s}_{t'-1}, \boldsymbol{h}_t)$$

$$a(\boldsymbol{s}, \boldsymbol{h}) = \boldsymbol{v}^\top \tanh(\boldsymbol{W}_s \boldsymbol{s} + \boldsymbol{W}_h \boldsymbol{h})$$



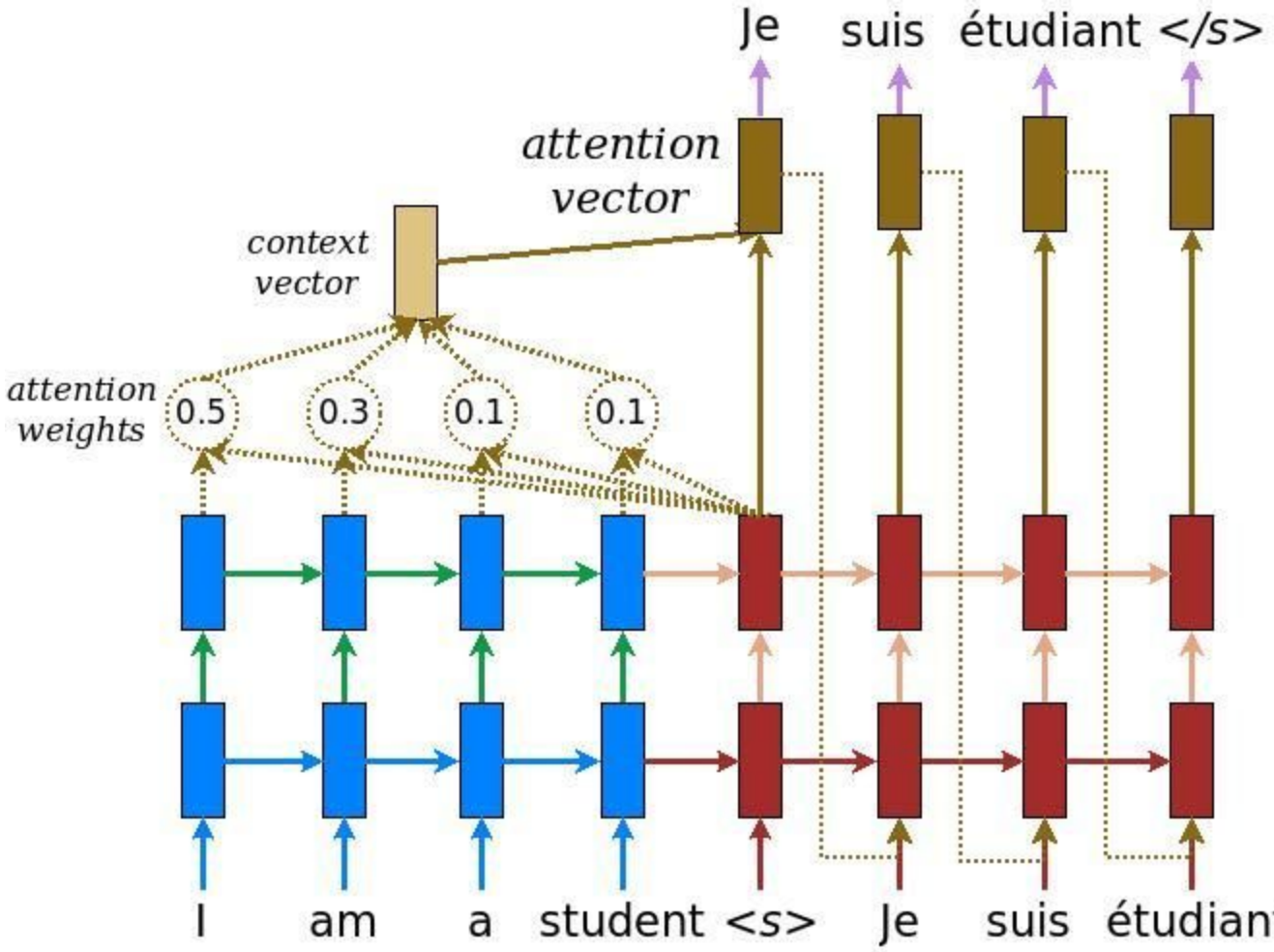
http://en.diveintodeeplearning.org/chapter_natural-language-processing/attention.html

Attention Mechanism

Attention is simply a vector, often the outputs of dense layer using softmax function.

It is just an interface, you could plug it anywhere you find it suitable.

Can use different functions: tahn, sigmoid, dot product, learned transformations



https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

Attention Mechanism

q is the query (decoder state) and **k** is the key (encoder states)

- **Multi-layer Perceptron** (Bahdanau et al. https://arxiv.org/abs/1409.0473)

  $$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_2^\intercal \tanh(W_1[\boldsymbol{q}; \boldsymbol{k}])$$

  Sometimes called ‘concat’.
- **Bilinear** (Luong et al. https://arxiv.org/abs/1508.04025)

  $$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal W \boldsymbol{k}$$
- **Dot Product** (Luong)

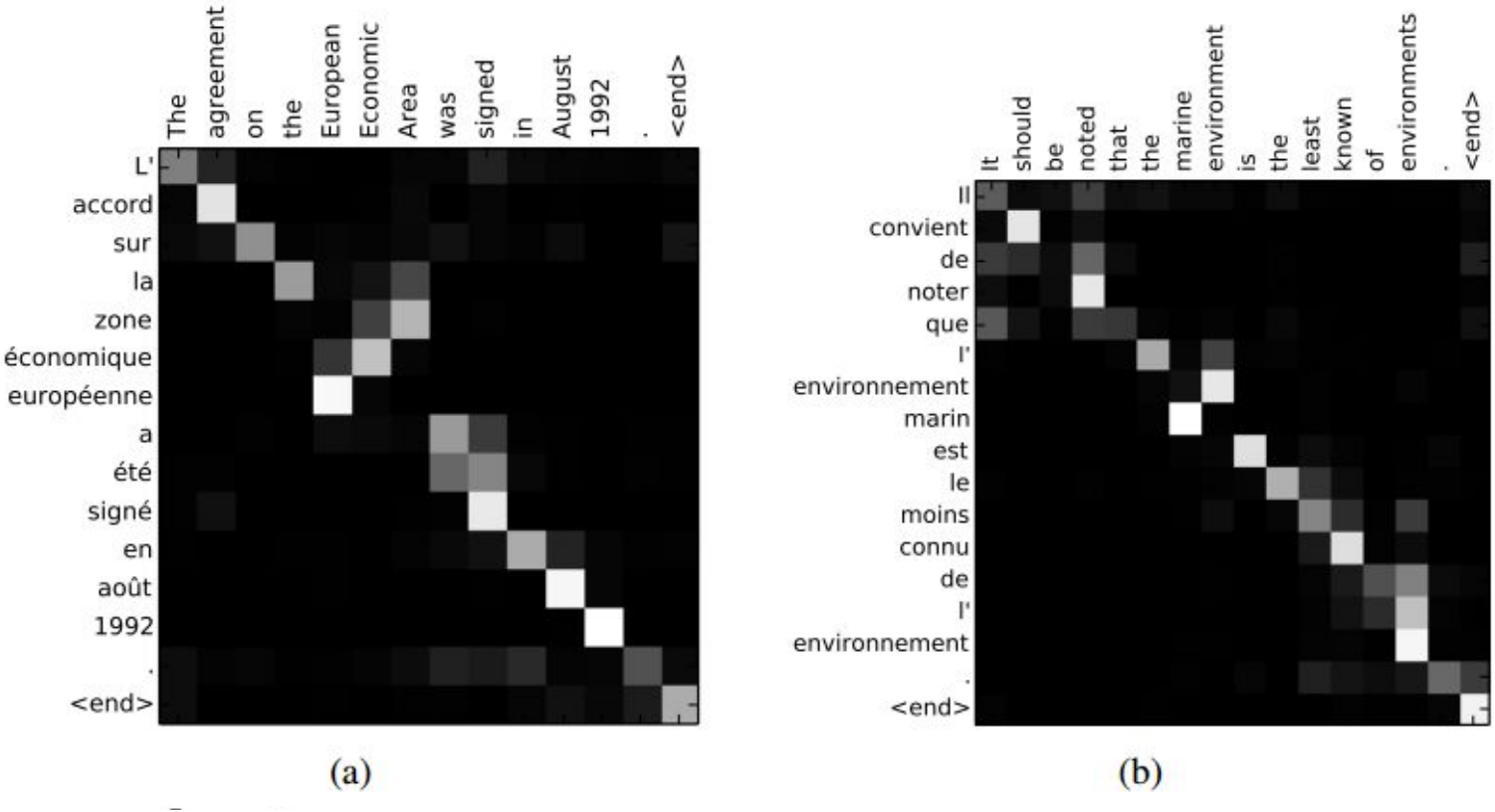
  $$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal \boldsymbol{k}$$
- **Scaled Dot Product** (Vaswani et al. https://arxiv.org/abs/1706.03762)

  $$a(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\intercal \boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}}$$

  Scale by size of the vector

http://phontron.com/class/nn4nlp2017/assets/slides/nn4nlp-09-attention.pdf

Visualizing RNN attention weights α_{ij} on MT



Neural Machine Translation by Jointly Learning to Align and Translate, https://arxiv.org/abs/1409.0473

Visualizing RNN attention heat maps on QA

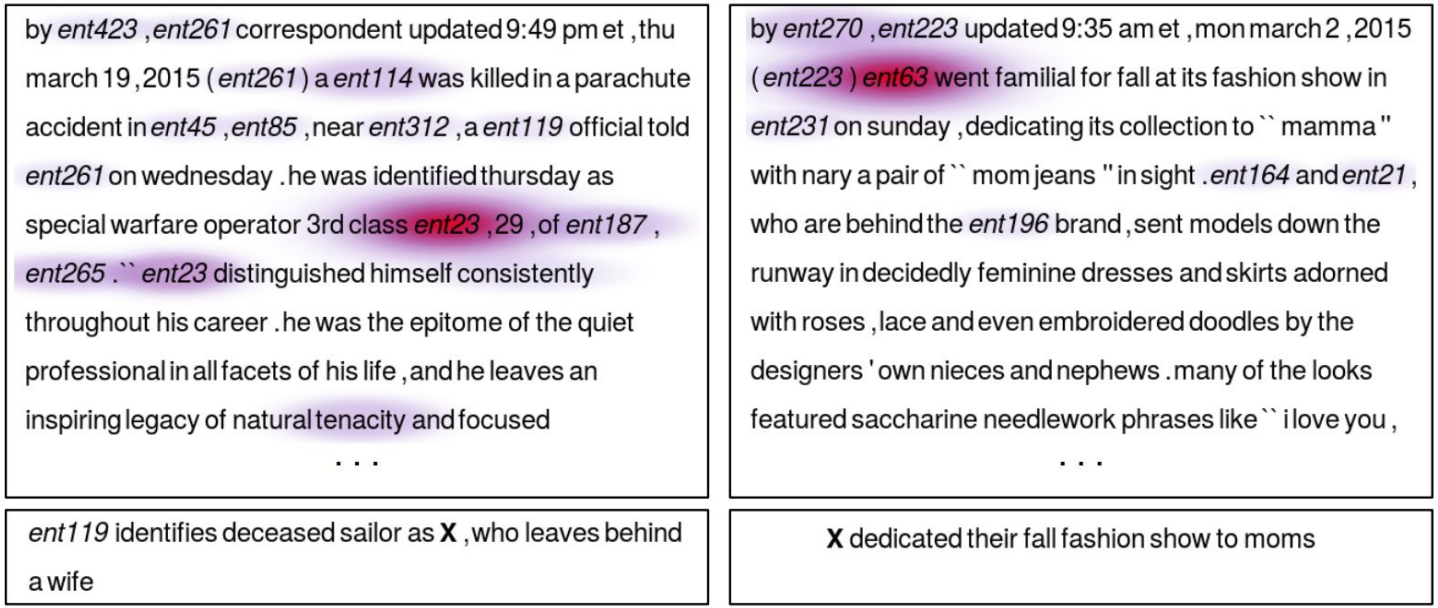
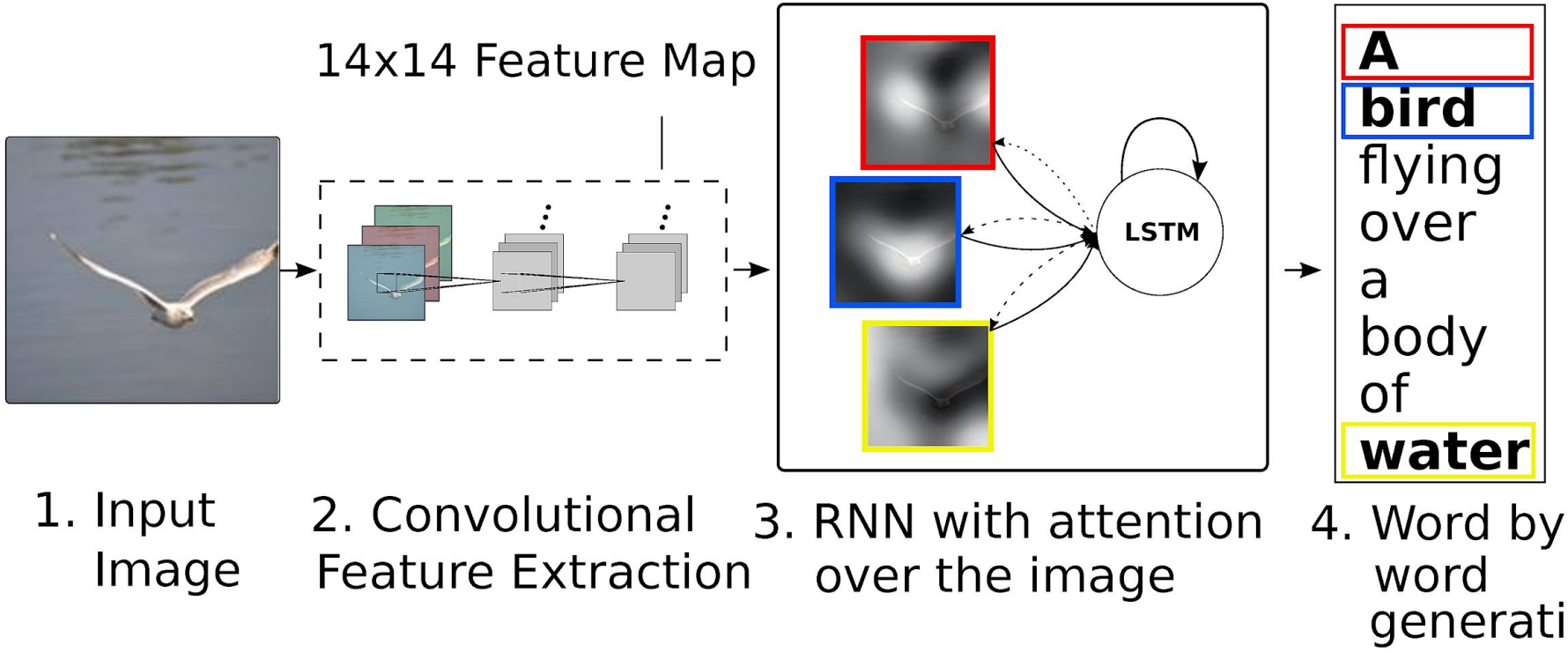


Figure 3: Attention heat maps from the Attentive Reader for two correctly answered validation set queries (the correct answers are *ent23* and *ent63*, respectively). Both examples require significant lexical generalisation and co-reference resolution in order to be answered correctly by a given model.

Teaching Machines to Read and Comprehend, https://arxiv.org/abs/1506.03340

CNN+RNN with Attention



http://kelvinxu.github.io/projects/capgen.html

CNN+RNN with Attention

A stop sign is on a road with a mountain in the background.

A woman is throwing a frisbee in a park.

http://kelvinxu.github.io/projects/capgen.html

Hard vs. Soft attention

Soft means differentiable, Hard means non-differentiable.

The mechanism described previously is called **Soft attention** because it is a fully differentiable deterministic mechanism that can be plugged into an existing system, and the gradients are propagated through the attention mechanism at the same time they are propagated through the rest of the network.

Hard attention is a stochastic process: instead of using all the hidden states as an input for the decoding, the system samples a hidden state y_i with the probabilities s_i. In order to propagate a gradient through this process, we estimate the gradient by Monte Carlo sampling. Alternatively you can use Reinforcement Learning approaches.

https://blog.heuritech.com/2016/01/20/attention-mechanism/
https://jhui.github.io/2017/03/15/Soft-and-hard-attention/

Self-attention (Intra-Attention)

Each element in the sentence attends to other elements. It gives context sensitive encodings.

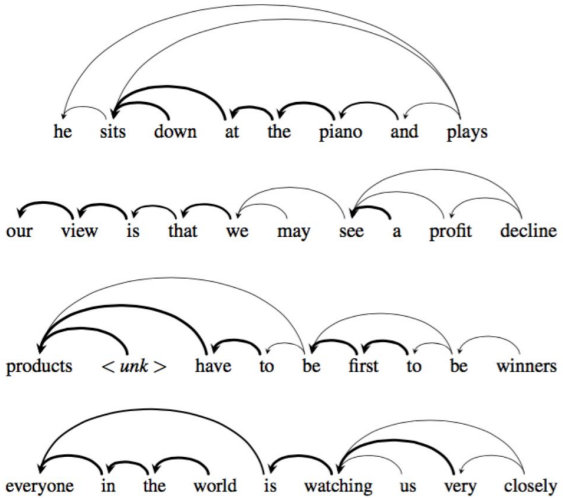


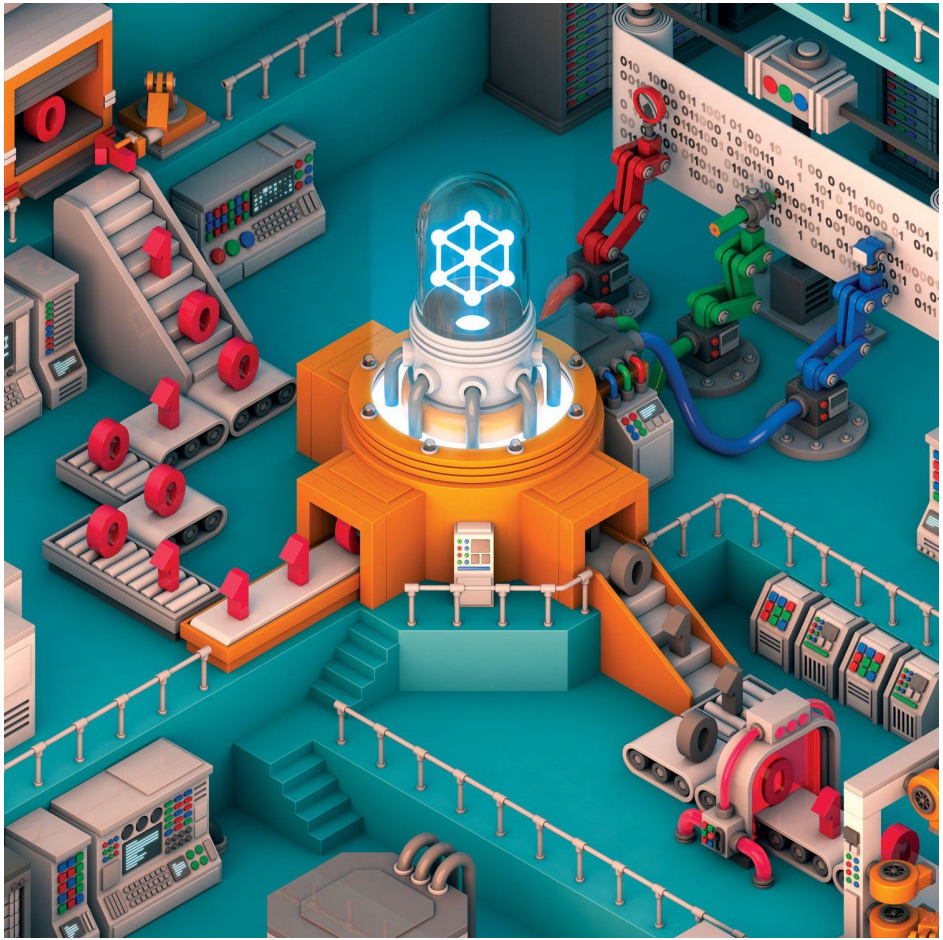
Figure 4: Examples of intra-attention (language modeling). Bold lines indicate higher attention scores. Arrows denote which word is being focused when attention is computed, but not the direction of the relation.

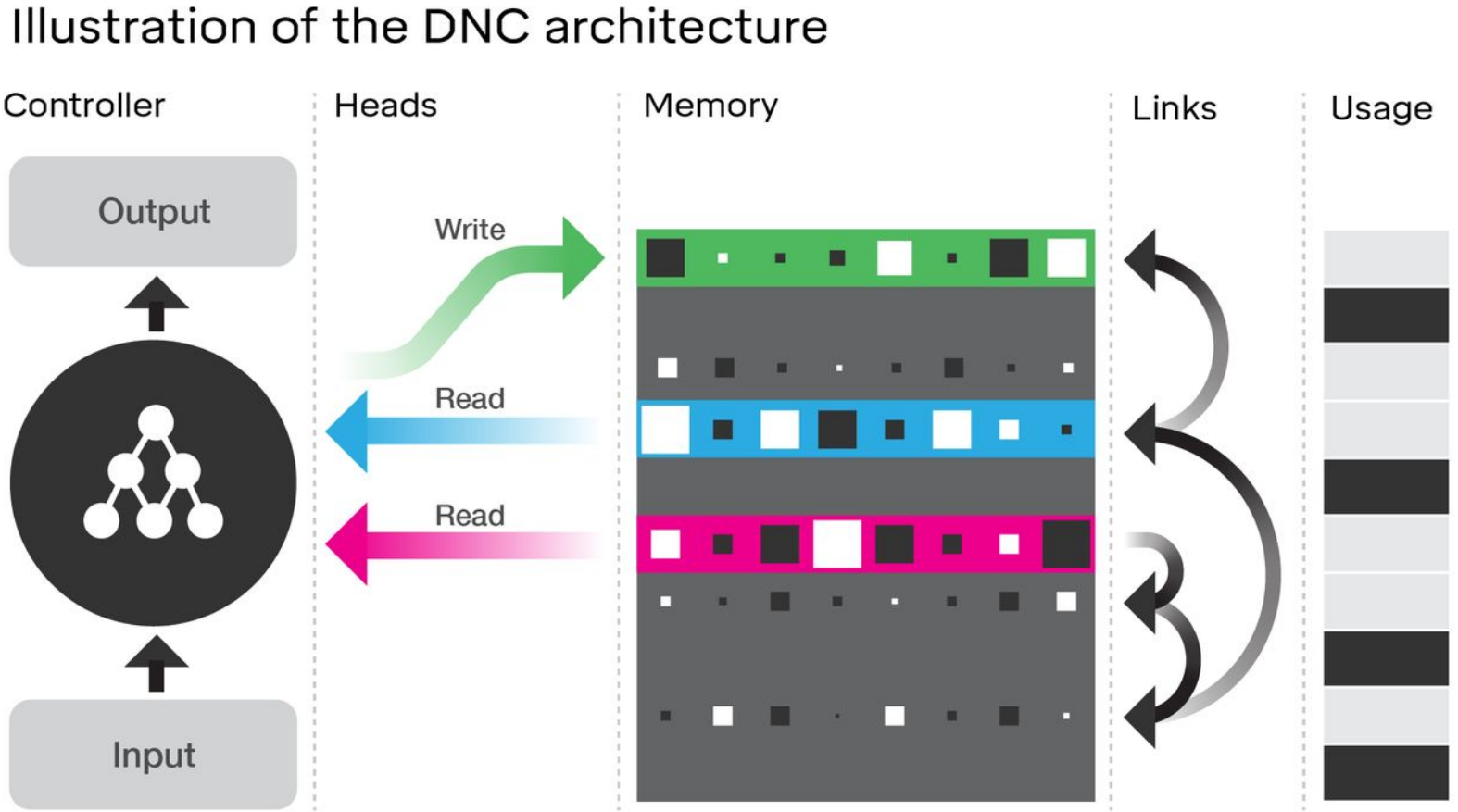
Long Short-Term Memory-Networks for Machine Reading, https://arxiv.org/abs/1601.06733

Augmented RNNs

More augmented RNNs

- Attentional Interfaces (Hard attention, Soft attention)
- Differentiable Memory (Neural Turing Machines, Differentiable neural computer, Hierarchical Attentive Memory, Memory Networks, ...)
- Adaptive Computation Time
- Differentiable Data Structures (structured memory: stack, list, queue, …)
- Differential Programming (Neural Programmer, Differentiable Functional Program Interpreters, …)
- ...





https://deepmind.com/blog/differentiable-neural-computers/

Resources

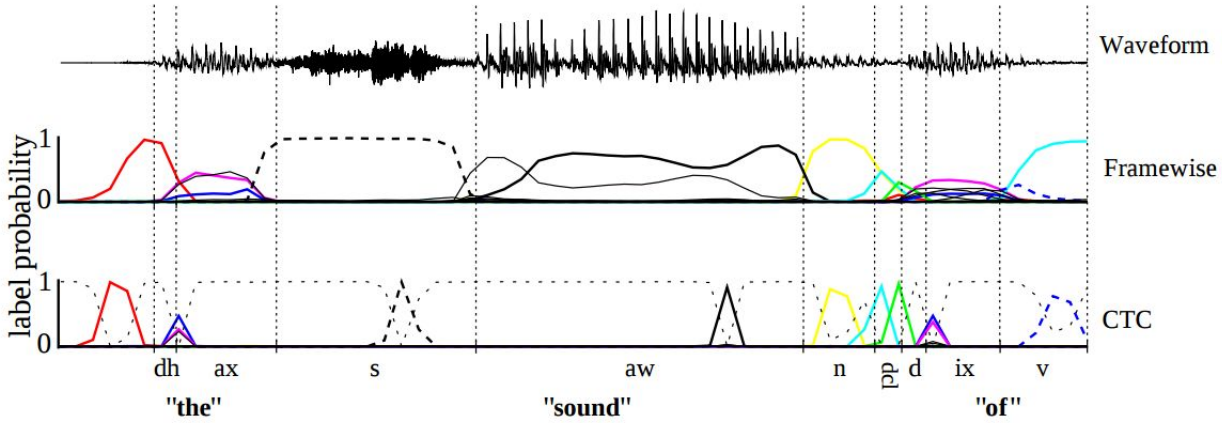
- Visualizing A Neural Machine Translation Model (Mechanics of Seq2seq Models With Attention) https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/
- Memory, attention, sequences https://towardsdatascience.com/memory-attention-sequences-37456d271992
- Attention? Attention! https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html
- Attention and Augmented Recurrent Neural Networks http://distill.pub/2016/augmented-rnns/
- https://mchromiak.github.io/articles/2017/Sep/01/Primer-NN/#attention-basis
- TF example: Neural machine translation with attention https://www.tensorflow.org/tutorials/text/nmt_with_attention

Connectionist Temporal Classification (CTC)

CTC (Connectionist Temporal Classification)

There are many tasks where the exact label placement is not important, only the order is important. E.g. speech recognition, handwriting recognition, licence plate recognition.

Special CTC output layer (Graves, Fernández, Gomez, Schmidhuber, 2006) was created for such tasks of temporal classification, where alignment between input frames and output labels is not known and is not required.



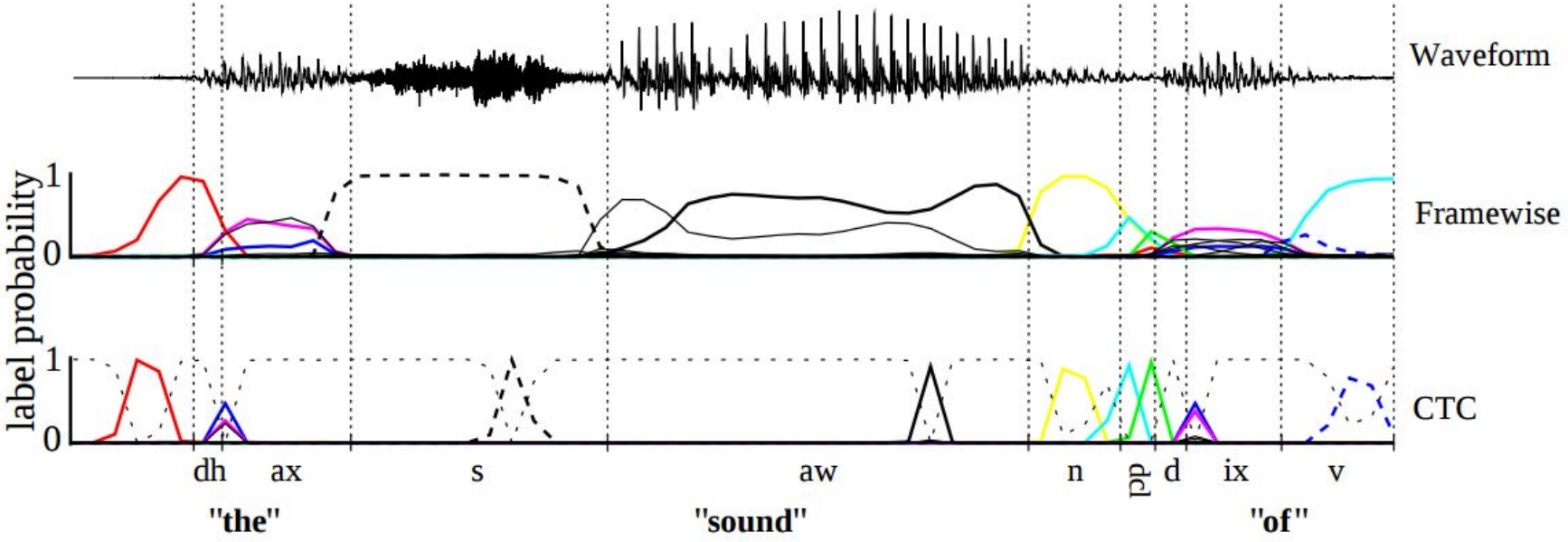
Another useful thing: CTC Output Layer

CTC (Connectionist Temporal Classification; Graves, Fernández, Gomez, Schmidhuber, 2006) was specifically designed for temporal classification tasks; that is, for sequence labelling problems where the alignment between the inputs and the target labels is unknown.

CTC models all aspects of the sequence with a single neural network, and does not require the network to be combined with a hidden Markov model. It also **does not require presegmented training data**, or external post-processing to extract the label sequence from the network outputs.

The CTC network **predicts only the sequence** of phonemes (typically as a series of spikes, separated by ‘blanks’, or null predictions), while the framewise network attempts to align them with the manual segmentation.

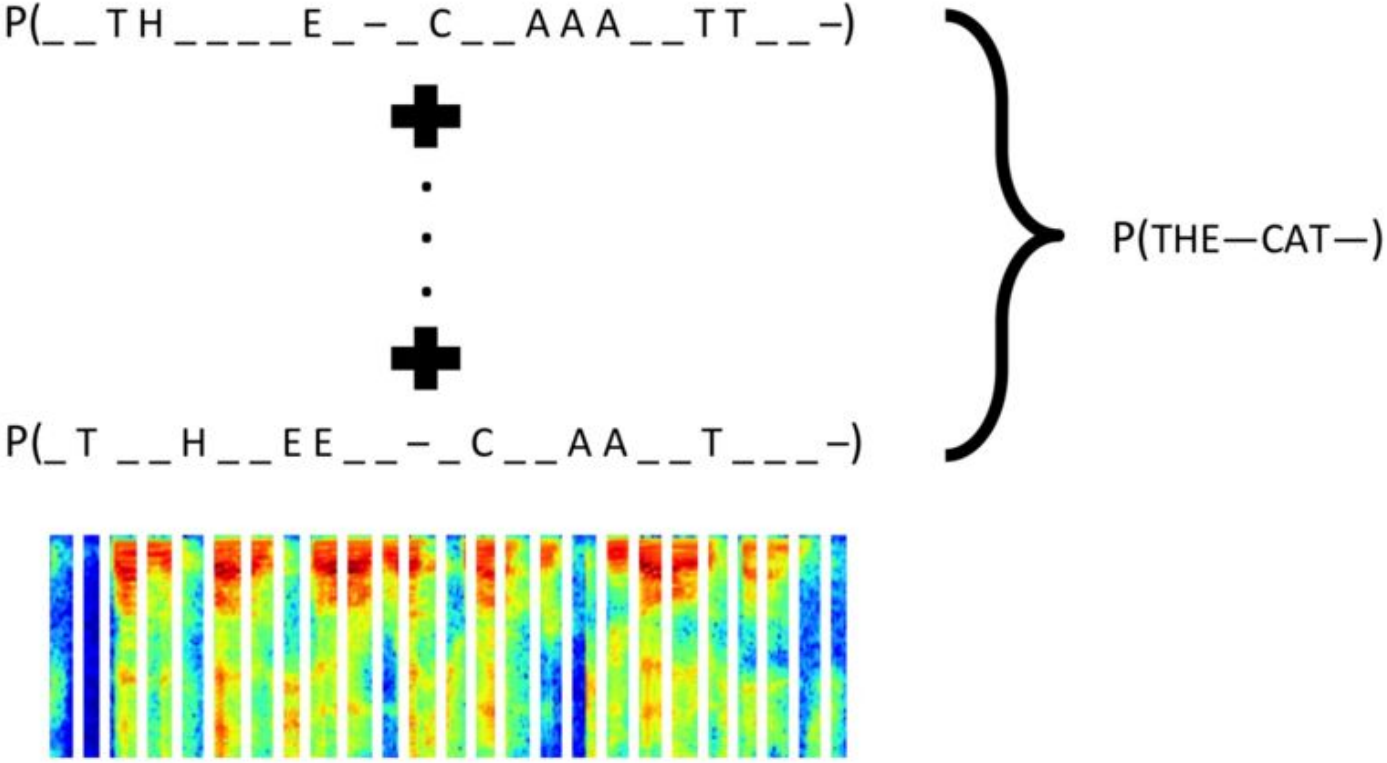
Example: CTC vs. Framewise classification



CTC

- Models the sequence. No further processing of recognized text required (e.g. HMM decoding).
- Goes from probabilities at each time-step to the probability of an output sequence.
- Introduces a special blank pseudo-character “-” (helps to solve duplicate-character problem)
  - If you have a sequence of phonemes/symbols “tttooooooooo”, what exactly does it mean? “to” or “too”?
  - Using blank symbols they can be distinguished: “--ttt-ooo-oooooo”
- Is a special loss function
  - calculated by summing up all scores of all possible alignments of the GT
- Decoding:
  - usually (but not only) best path (greedy) (“---ttt---oooooo-----oooooo----”)
  - then removing duplicate characters (“-t-o-o-”)
  - then removing all blanks from the path (“too”)

CTC (Connectionist Temporal Classification)



https://github.com/baidu-research/warp-ctc

Resources

- **Supervised Sequence Labelling with Recurrent Neural Networks (Multidimensional RNN, CTC and sequence labeling)**
  https://www.cs.toronto.edu/~graves/preprint.pdf
- **Sequence Modeling With CTC**
  https://distill.pub/2017/ctc
- CTC and Speech Recognition
  https://habrahabr.ru/company/google/blog/269747/
- Speech Recognition: You down with CTC?
  https://gab41.lab41.org/speech-recognition-you-down-with-ctc-8d3b558943f0
- https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
- https://www.youtube.com/watch?v=c86gfVGcvh4
- https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss

Non-RNN Sequence Learning

Encoder-Decoder: original architecture

Recurrent Encoder / Recurrent Decoder

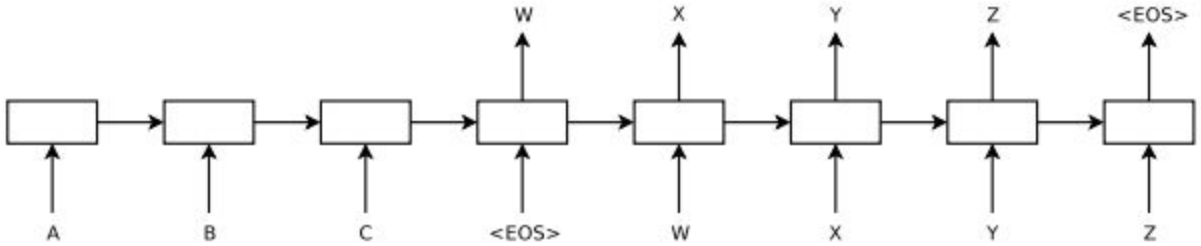


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence to Sequence Learning with Neural Networks, https://arxiv.org/abs/1409.3215

Case: Machine Translation

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

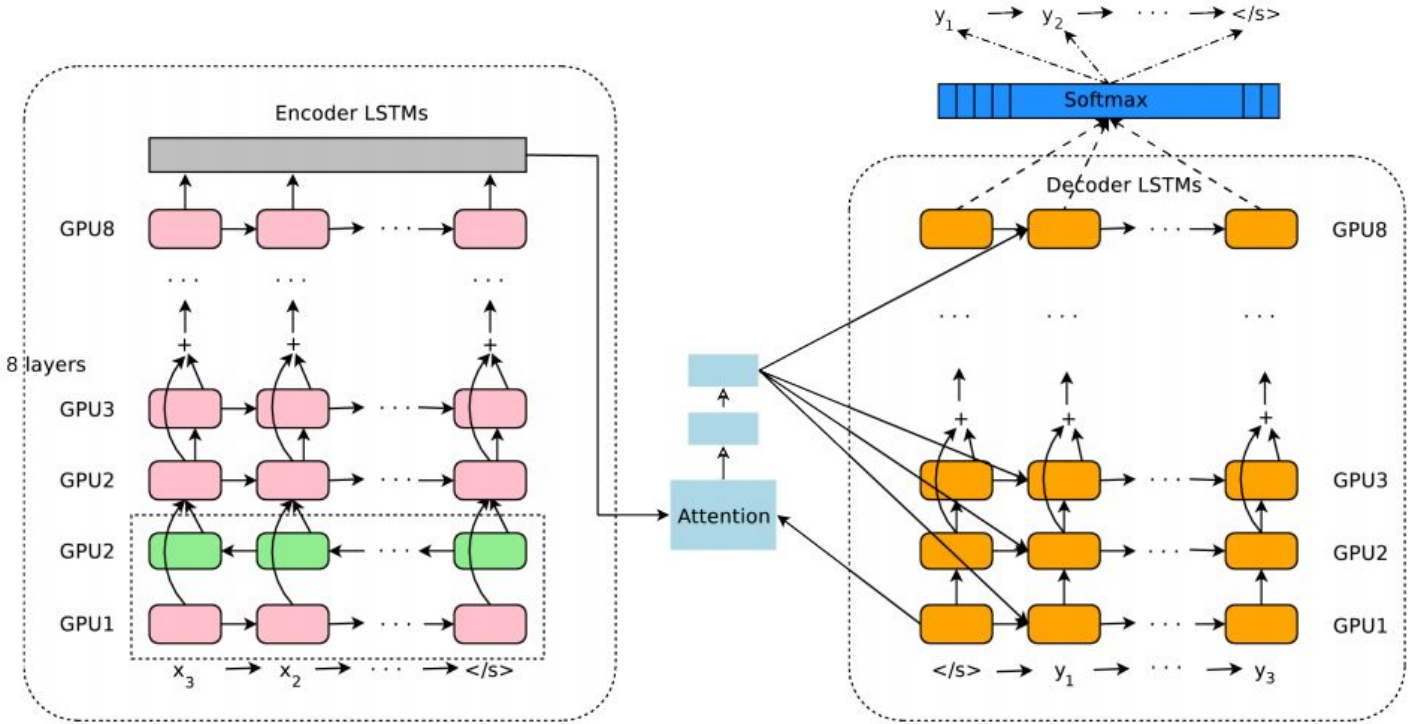
Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
State of the art [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	~45

Table 2: Methods that use neural networks together with an SMT system on the WMT'14 English to French test set (ntst14).

Sequence to Sequence Learning with Neural Networks, http://arxiv.org/abs/1409.3215

Encoder-Decoder: modern architecture



Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation,
https://arxiv.org/abs/1609.08144
Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation
https://arxiv.org/abs/1611.04558

Encoder-Decoder: character-level models

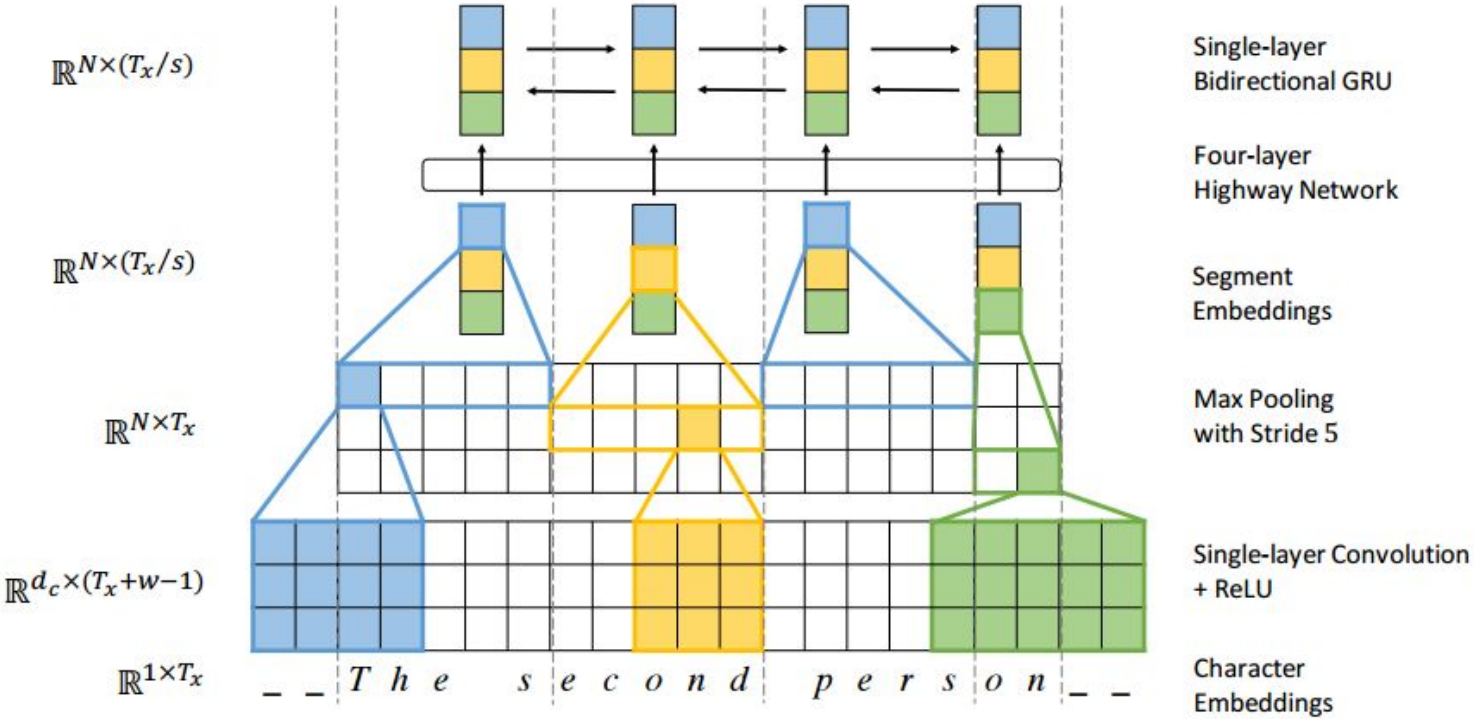


Figure 1: Encoder architecture schematics. Underscore denotes padding. A dotted vertical line delimits each segment.

Fully Character-Level Neural Machine Translation without Explicit Segmentation,
https://arxiv.org/abs/1610.03017

The Problem:
RNNs are slow.

The solution #1: Optimizing RNNs

"Pruning RNNs reduces the size of the model and can also help achieve significant inference time speed-up using sparse matrix multiply. Benchmarks show that using our technique model size can be reduced by 90% and speed-up is around 2× to 7×."

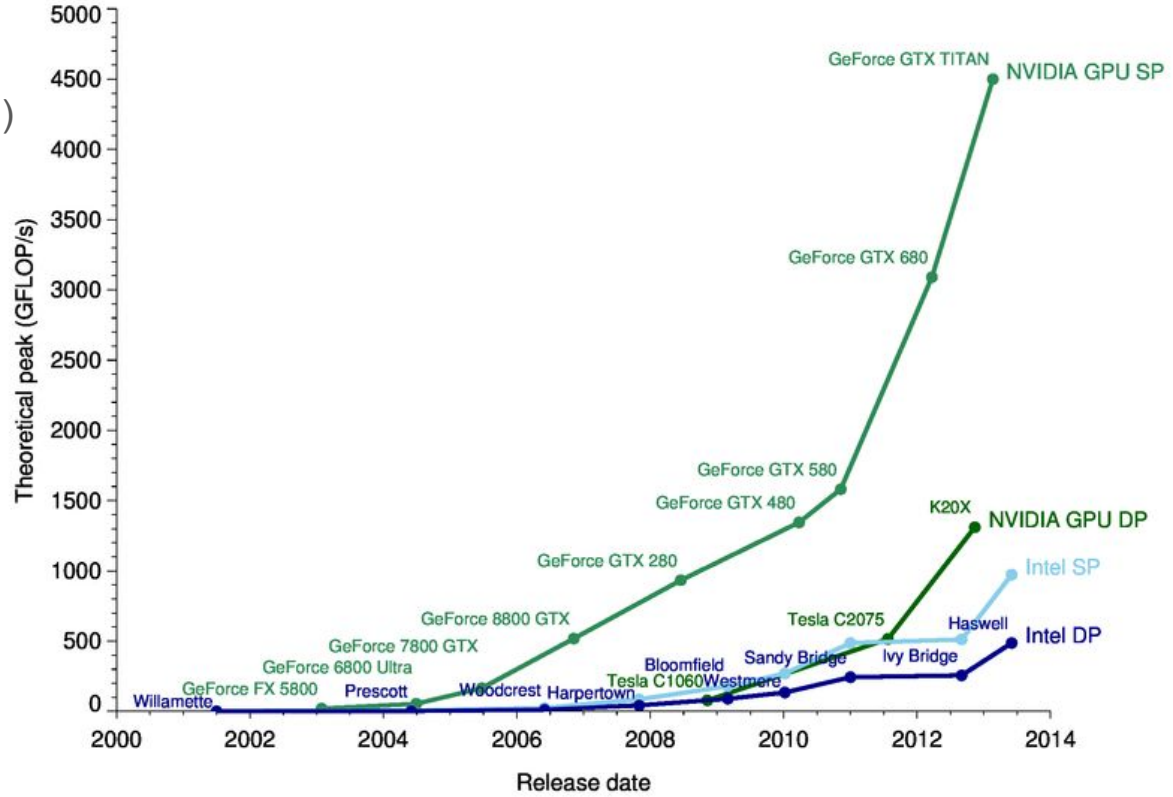
Table 3: GRU & bidirectional RNN model results

MODEL	# UNITS	CER	# PARAMS	RELATIVE PERF
RNN Dense Baseline	1760	10.67	67 million	0.0%
RNN Dense Small	704	14.50	11.6 million	-35.89%
RNN Dense Medium	2560	9.43	141 million	11.85%
RNN Sparse 1760	1760	12.88	8.3 million	-20.71%
RNN Sparse Medium	2560	**10.59**	11.1 million	0.75%
RNN Sparse Big	3072	**10.25**	16.7 million	3.95%
GRU Dense	2560	9.55	115 million	0.0%
GRU Sparse	2560	10.87	13 million	-13.82%
GRU Sparse Medium	3568	**9.76**	17.8 million	-2.20%

Exploring Sparsity in Recurrent Neural Networks, https://arxiv.org/abs/1704.05119

The solution #2: Better hardware

- Google TPU gen.2
  - 180 TFLOPS?
- NVIDIA DGX-1 (8*P100) ($129,000)
  - 170 TFLOPS (FP16)
  - 85 TFLOPS (FP32)
- NVIDIA Tesla V100
  - 15 TFLOPS (FP32)
  - 120 TFLOPS (Tensor Core)
- NVIDIA Tesla P100
  - 10.6 TFLOPS (FP32)
- NVIDIA GTX Titan X ($1000)
  - 11 TFLOPS (FP32)
- NVIDIA GTX 1080/1080 Ti ($700)
  - 8/11.3 TFLOPS (FP32)



The solution #2: Better hardware

Why this solution could be among the most interesting ones?

Current success of NNs (especially CNNs) is backed by a large amounts of data available _AND_ more powerful hardware (using the decades-old algorithms). We potentially could achieve the same performance in the past, but the learning process was just too slow (and we were too impatient).

The processor performance grows exponentially and in 5-10 years the available computing power can increase 1000x. There may appear computing units more suitable for RNN computations as well.

The situation could repeat. When the hardware will allow fast training of RNNs, we could achieve a new kind of results. Remember, RNNs are Turing complete. They are (potentially) much more powerful than feed-forward NNs.

Convolutional Sequence Learning

CNN encoder

Convolutional Encoder / Recurrent Decoder

Encoder	Words/s	BLEU
BiLSTM	139.7	22.4
Deep Conv. 6/3	187.9	23.1

(a) IWSLT'14 German-English generation speed on *tst2013* with beam size 10.

Encoder	Words/s	BLEU
2-layer BiLSTM	109.9	23.6
Deep Conv. 8/4	231.1	23.7
Deep Conv. 15/5	203.3	24.0

(b) WMT'15 English-German generation speed on *newstest2015* with beam size 5.

Table 3: Generation speed in source words per second on a single CPU core.

A Convolutional Encoder Model for Neural Machine Translation, https://arxiv.org/abs/1611.02344

CNN encoder

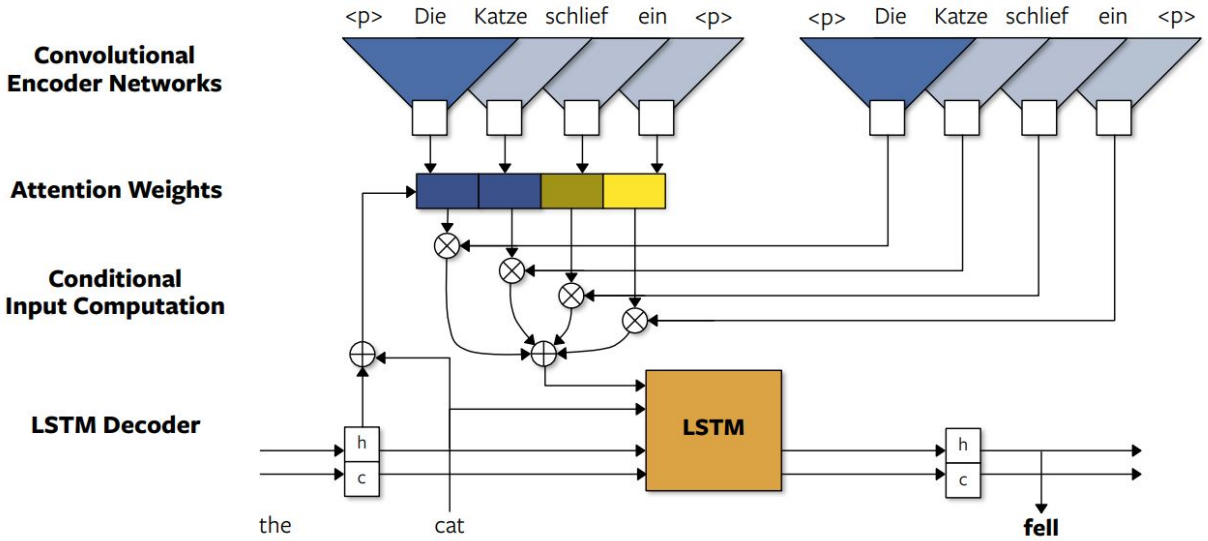


Figure 1: Neural machine translation model with single-layer convolutional encoder networks. CNN-a is on the left and CNN-c is at the right. Embedding layers are not shown.

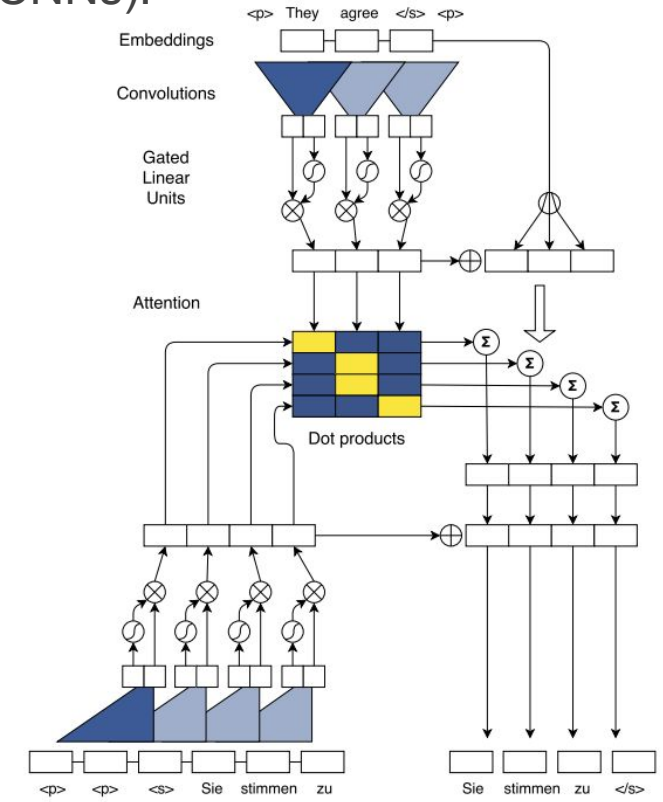
A Convolutional Encoder Model for Neural Machine Translation, https://arxiv.org/abs/1611.02344

CNN encoder + decoder

Actually no RNN here (Facebook AI Research loves CNNs).

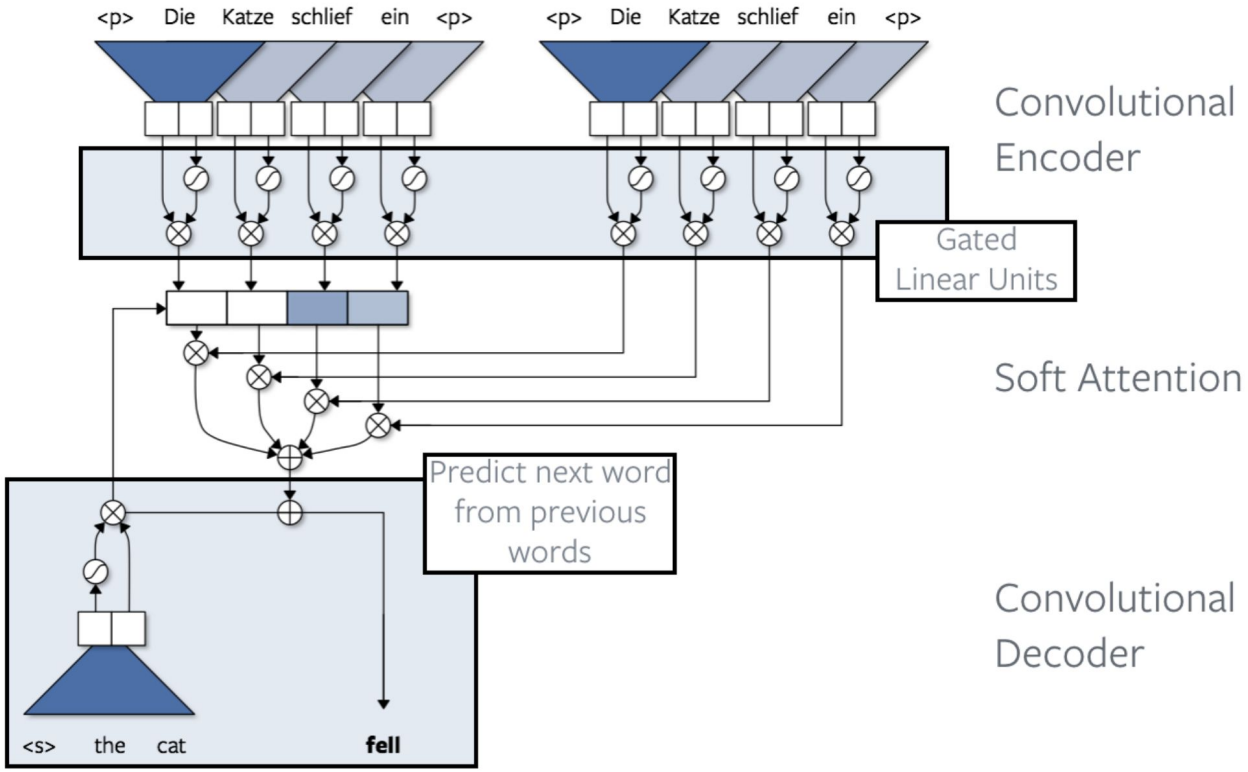
	BLEU	Time (s)
GNMT GPU (K80)	31.20	3,028
GNMT CPU 88 cores	31.20	1,322
GNMT TPU	31.21	384
ConvS2S GPU (K40) $b = 1$	33.45	327
ConvS2S GPU (M40) $b = 1$	33.45	221
ConvS2S GPU (GTX-1080ti) $b = 1$	33.45	142
ConvS2S CPU 48 cores $b = 1$	33.45	142
ConvS2S GPU (K40) $b = 5$	34.10	587
ConvS2S CPU 48 cores $b = 5$	34.10	482
ConvS2S GPU (M40) $b = 5$	34.10	406
ConvS2S GPU (GTX-1080ti) $b = 5$	34.10	256

Table 3. CPU and GPU generation speed in seconds on the development set of WMT'14 English-French. We show results for different beam sizes b. GNMT figures are taken from Wu et al. (2016). CPU speeds are not directly comparable because Wu et al. (2016) use a 88 core machine compared to our 48 core setup.



Convolutional Sequence to Sequence Learning, https://arxiv.org/abs/1705.03122

CNN encoder + decoder



https://aiukraine.com/wp-content/uploads/2017/10/2_1-Yarats.pdf

Resources

- https://github.com/facebookresearch/fairseq (old)
- https://github.com/pytorch/fairseq (new)

  - **Convolutional Neural Networks (CNN)**
    - Dauphin et al. (2017): Language Modeling with Gated Convolutional Networks
    - Gehring et al. (2017): Convolutional Sequence to Sequence Learning
    - Edunov et al. (2018): Classical Structured Prediction Losses for Sequence to Sequence Learning
    - ***New*** Fan et al. (2018): Hierarchical Neural Story Generation
  - **Long Short-Term Memory (LSTM) networks**
    - Luong et al. (2015): Effective Approaches to Attention-based Neural Machine Translation
    - Wiseman and Rush (2016): Sequence-to-Sequence Learning as Beam-Search Optimization
  - **Transformer (self-attention) networks**
    - Vaswani et al. (2017): Attention Is All You Need
    - ***New*** Ott et al. (2018): Scaling Neural Machine Translation
    - ***New*** Edunov et al. (2018): Understanding Back-Translation at Scale

Thanks!

grigory.sapunov@ieee.org
gs@inten.to