# Generating sequences with **Recurrent Neural Networks**

Computational Neuroscience Seminar

Tarasco Pier Paolo

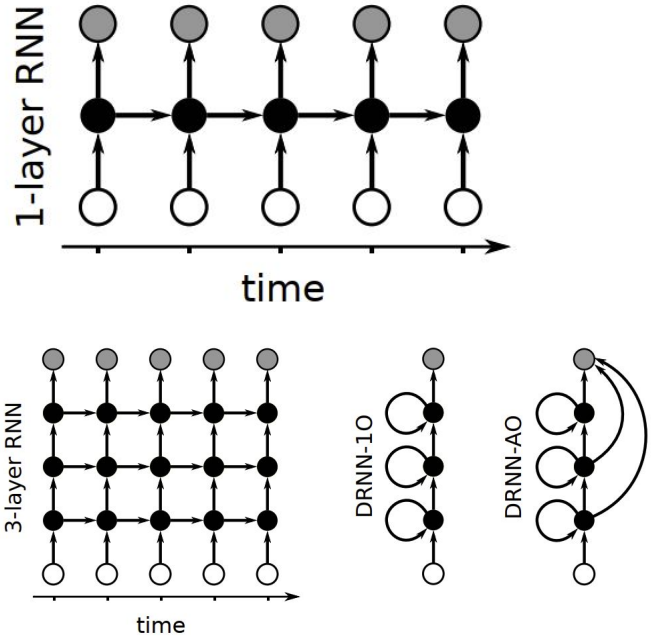
# Which problems are we trying to solve?

- Our aim is to solve tasks that require a form of memory
- Usually this problems have to deal with sequences where we must remember what we have processed earlier in order to make the current prediction
- We'll mainly focus on tasks dealing with text or more generally with sequences.

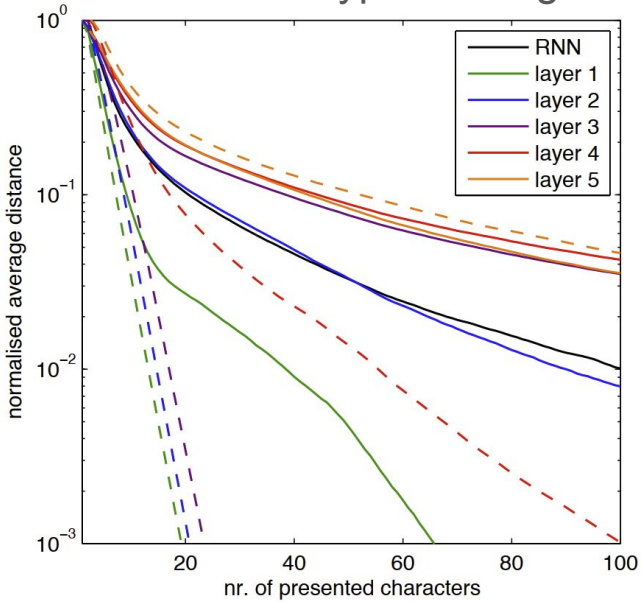
# Why sequences? Lot of interesting and important problems

- Health Care: Help with processing of ECG/MRI
- Text to Audio and Images to Audio to improve screen readers for the blinds
- … and many others

# RNN and Deep-RNN [1][2]




- A deep RNN allows the model to process the sequence at different time-scales
- Experiment: Run the network on two identical sequences but after 100 steps introduce a random typo in one of them.
  Record the next hidden states of both sequences and measure the Euclidean distance between them → We can see that the deepest layers maintains the typo for longer.



# Training difficulties

In practise the RNN has difficulties in learning long-range dependencies due to the Gradient vanishing / exploding problem. [3]

To solve this issue we will analyze different architectures that ease the task of a long-term memory, in detail:

- Long Short Term Memory (LSTM)
- Multiplicative RNN (MRNN) trained with an HF optimizer

Also, to ease the training we usually use output connections at each layer to facilitate the training.

# Text Generation at character level with RNN

Tries to overcome the difficulties of training RNN by:

1. Using an HF Optimizer
2. Dynamically adapt the hidden weight recurrent matrix to the current input → **Multiplicative RNN**

Classic RNN

for $t = 1$ to $T$:

$$
\begin{aligned}
h_t &= \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \\
o_t &= W_{oh}h_t + b_o
\end{aligned}
$$

A more powerful method for the input $x_t$ to influence the hidden state dynamics is to determine the entire $W_{hh}$ based on the current input.

$$
\begin{aligned}
h_t &= \tanh\left(W_{hx}x_t + W_{hh}^{(x_t)}h_{t-1} + b_h\right) \\
o_t &= W_{oh}h_t + b_o
\end{aligned}
$$

# The multiplicative RNN [4]

We can easily store the matrices in a Tensor, where M is the dimension of the input $x_t$

$$W_{hh}^{(x_t)} = \sum_{m=1}^{M} x_t^{(m)} W_{hh}^{(m)}$$

However, in practice, it's unfeasible to store this Tensor since, for example, if we want to use an RNN with 1000 hidden units and an input dimension of 50 the tensor would be of size 1000x50x50.

We can factorize $W_{hh}$ into 3 matrices: $W_{fx}$, $W_{hf}$ and $W_{fh}$

$$W_{hh}^{(x_t)} = W_{hf} \cdot \mathrm{diag}(W_{fx} x_t) \cdot W_{fh}$$

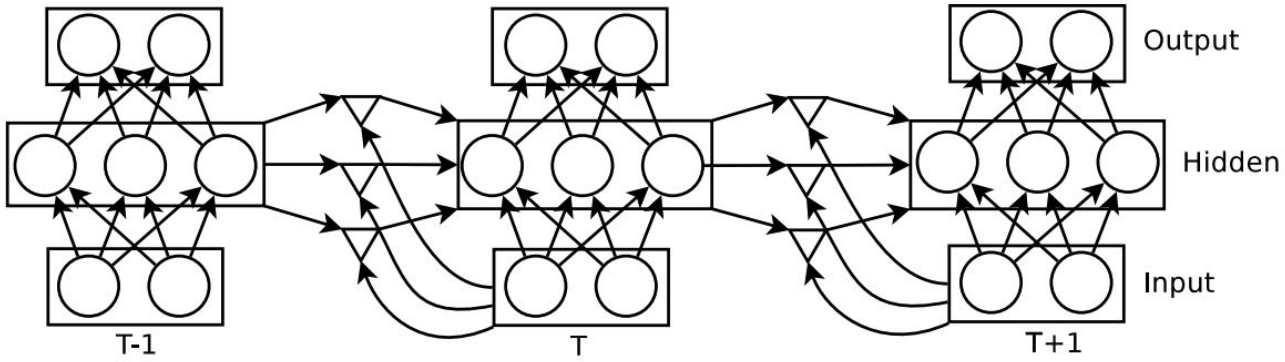
# MRNN Dynamics

As the dimensionality of $W_{fx}x_t$ gets larger, the factorization becomes as expressive as the original Tensor.

$$W_{hh}^{(x_t)} = W_{hf} \cdot \mathrm{diag}(W_{fx}x_t) \cdot W_{fh}$$

$$\begin{aligned} h_t &= \tanh\left(W_{hx}x_t + W_{hh}^{(x_t)}h_{t-1} + b_h\right) \\ o_t &= W_{oh}h_t + b_o \end{aligned}$$

$$\begin{aligned} f_t &= \mathrm{diag}(W_{fx}x_t) \cdot W_{fh}h_{t-1} \\ h_t &= \tanh(W_{hf}f_t + W_{hx}x_t) \\ o_t &= W_{oh}h_t + b_o \end{aligned}$$




The MRNN Architecture, the triangles are the factors $f_t$

# Is the MRNN better than the classic RNN?

- Compare an RNN with 500 hidden units with an MRNN with 350 hidden units and 350 factors on the Machine Learning dataset (a corpus of papers of Machine Learning, ~100MB of text)

- Even if the **MRNN** has slightly fewer parameters, it **achieved better performance** at **1.56 BPC** while the RNN reached 1.65 BPC.

- bits per character (BPC) = $-\log_2 P(x_{t+1}|y_t)$ = Avg. loss

# Difficulties of learning the multiplicative units

In the MRNN the weight from hidden unit i to hidden units j parametrized by the character c is given by:

$$W_{ij}^{(c)} = \sum_f W_{if} W_{fc} W_{fj}$$

- This product of parameters can make gradient descent learning hard → Mitigates by using an HF optimizer (Conjugate gradient method).

- To facilitate the training we also truncate the sequences splitting a large text into many shorter sequences → Easier to parallelize

# Other experiments and samples

## Wikipedia dataset

The meaning of life is the tradition of the ancient human reproduction: it is less favorable to the good boy for when to remove her bigger. In the show's agreement unanimously resurfaced. The wild pasteured with consistent street forests were incorporated by the 15th century BE. In 1996 the primary rapford undergoes an effort that the reserve conditioning, written into Jewish cities, sleepers to incorporate the .St Eurasia that activates the population. Mar??a Nationale, Kelli, Zedlat-Dukastoe, Florendon, Ptu's thought is. To adapt in most parts of North America, the dynamic fairy Dan please believes, the free speech are much related to the

## New York Times articles

while he was giving attention to the second advantage of school building a 2-for-2 stool killed by the Cultures saddled with a half-suit defending the Bharatiya Fernall 's office . Ms . Claire Parters will also have a history temple for him to raise jobs until naked Prodiena to paint baseball partners , provided people to ride both of Manhattan in 1978 , but what was largely directed to China in 1946 , focusing on the trademark period is the sailboat yesterday and comments on whom they obtain overheard within the 120th anniversary , where many civil rights defined , officials said early that forms , " said Bernard J. Marco Jr. of Pennsylvania , was monitoring New York

## Machine Learning papers

Recurrent network with the Stiefel information for logistic regression methods Along with either of the algorithms previously (two or more skewprecision) is more similar to the model with the same average mismatched graph. Though this task is to be studied under the reward transform, such as (c) and (C) from the training set, based on target activities for articles a ? 2(6) and (4.3). The PHDPic (PDB) matrix of cav'va using the three relevant information contains for tieming measurements. Moreover, because of the therap tor, the aim is to improve the score to the best patch randomly, but for each initially four data sets. As shown in Figure 11, it is more than 100 steps, we used ?? \to \infty with 1000

# LSTM and Attention as Mixture of Gaussians [5]

- Design tasks where the model has to use a memory of the past in order to achieve good performance.

Tasks analyzed:

1) Text Prediction
2) Handwriting generation
3) Text to Handwriting

Note: we have both differences in the output domain (discrete vs continuous) and also a conditioned sequence generation problem (Task 3)

# LSTM architecture

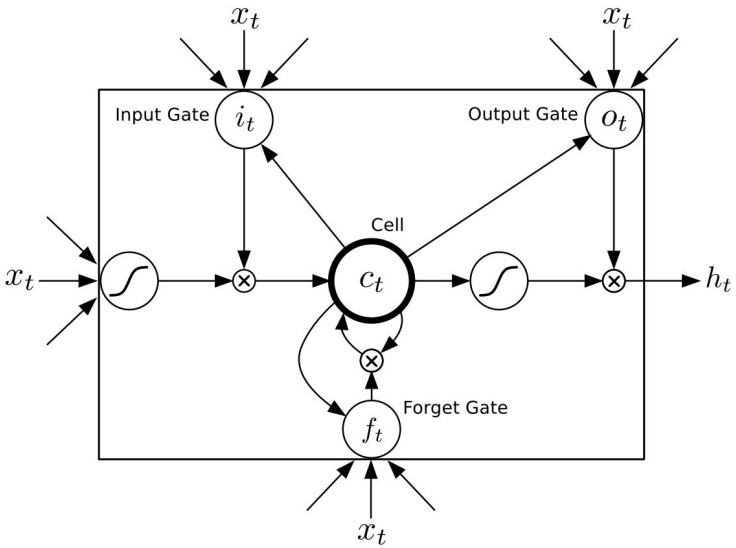
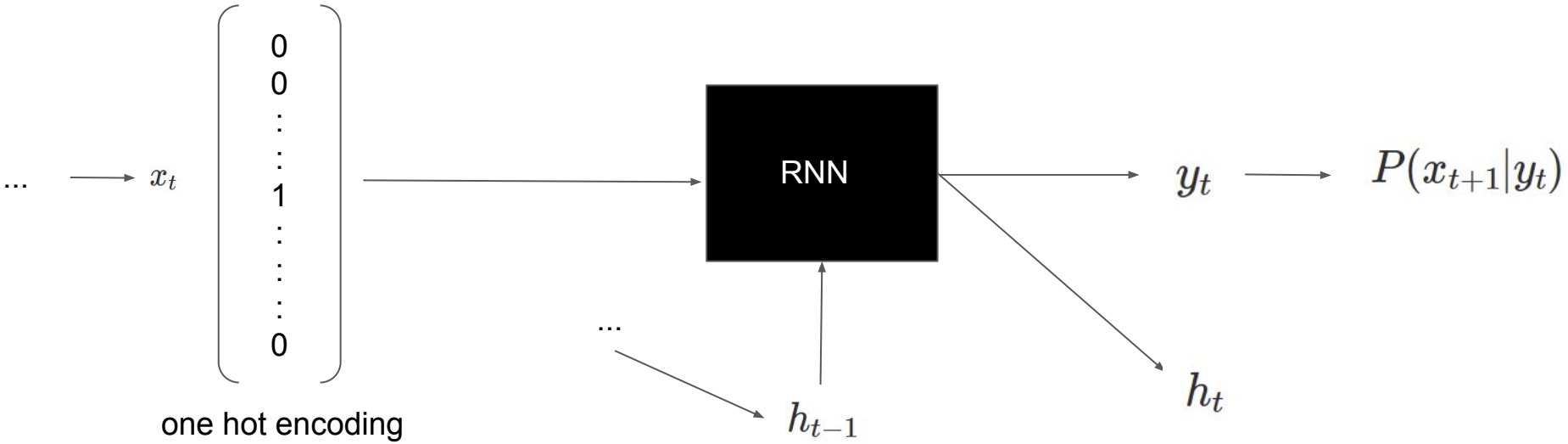



Figure 2: **Long Short-term Memory Cell**

- Introduction of a memory cell and multiplicative gates to protect the cell content from perturbation by irrelevant inputs.
- Facilitates learning due to multiple paths from the cell state at time **t** to the cell state at time **t-1**

# First task: Text Generation

- Text data is **discrete** → We have K classes and the output layer defines a probability distribution over them.



# Which set of classes to use?

- The most natural choice is to associate each word to a vector
- K, therefore, is the size of the vocabulary → Can be problematic as usually $|V| > 100000$ → **Increases the number of parameters** of the model and **requires more training data** since we have to look at a single word in various contexts.
- But, the most constraining disadvantage of representing words is that we lose the ability to generate novel words or numbers since we cannot represent an infinite number of classes.
- **Solution**: Operate at the **character level** since it allows for more flexibility

# Task 1: Penn Treebank Dataset

- Text from the Wall Street Journal containing a little over 1,000,000 words
- Vocabulary limited to the top 10,000 most used words in the dataset

- Compares performance of **LSTM** operating at **word level vs char level**
- **The same number of hidden recurrent units (1000)**
- **Char-LSTM**: input and output of size 49 → approx. **4.3M weights**
- **Word-LSTM**: input and output of size 10,000 → approx. **54M weights**

- Unfair comparison since the word level LSTM has about **10x** the number of parameters however both networks were easily able to overfit the training data → Not clear if the char-LSTM would have benefited from more weights

# Dynamic Evaluation and Regularization

- To evaluate the performance of the model we compare the classical evaluation with the **dynamic evaluation** which allows the network to adapt its weights as it is being evaluated.

To regularize the models we used two types of regularization:

1. **Weight noise:** After training the model we retrain the model having modified the final weights with random noise.
2. **Adaptive weight noise**: After training the model and having applied weight noise we re-train the model this time injecting a noise which variance has been learned.

- The noise is added to the weights of the trained unregularized network instead of training from scratch with regularization because the former is faster.

# Results and comparison with classic RNN

Table 1: **Penn Treebank Test Set Results.** 'BPC' is bits-per-character. 'Error' is next-step classification error rate, for either characters or words.

| Input | Regularisation | Dynamic | BPC | Perplexity | Error (%) | Epochs |
|---|---|---|---|---|---|---|
| Char | none | no | 1.32 | 167 | 28.5 | 9 |
| char | none | yes | 1.29 | 148 | 28.0 | 9 |
| char | weight noise | no | 1.27 | 140 | 27.4 | 25 |
| char | weight noise | yes | 1.24 | 124 | 26.9 | 25 |
| char | adapt. wt. noise | no | 1.26 | 133 | 27.4 | 26 |
| char | adapt. wt. noise | yes | 1.24 | 122 | 26.9 | 26 |
| word | none | no | 1.27 | 138 | 77.8 | 11 |
| word | none | yes | 1.25 | 126 | 76.9 | 11 |
| word | weight noise | no | 1.25 | 126 | 76.9 | 14 |
| word | weight noise | yes | 1.23 | 117 | 76.2 | 14 |

BPC = Avg. loss

Perplexity = 2^(5.6 * BPC) where 5.6 is the avg. word length

- Previous Perplexity was 124.7 using a Word-level RNN with 200-400 hidden units [6]

# Task 2: Handwriting generation

- Aim: Generate handwriting
- Change in output domain from discrete to continuous
- Input data corresponding to a list of triplets (x1, x2, x3) where **x1** and **x2** are **real values** describing the pen displacement while **x3** is a **binary value** which indicates whether the pen was lifted after this sample.
- Avg. letter contains 25 sample of (x1, x2, x3)
- Avg. line composed of 700 samples, each line composing a sequence.

# How to model the output $P(x_{t+1}|y_t)$ for real values

- The model outputs $y_t$ are used to parametrize a mixture distribution

Formally the Input and Output vector consists of:

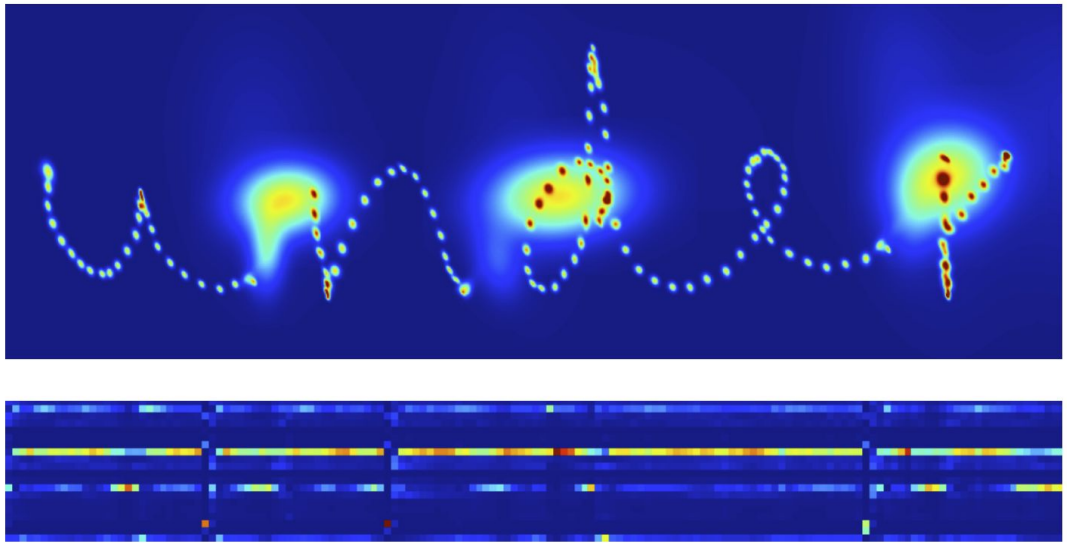
$$x_t \in \mathbb{R} \times \mathbb{R} \times \{0, 1\} \tag{15}$$

$$y_t = \left(e_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^M\right) \tag{16}$$

The probability density $\Pr(x_{t+1}|y_t)$ of the next input $x_{t+1}$ given the output vector $y_t$ is defined as follows:

$$\Pr(x_{t+1}|y_t) = \sum_{j=1}^{M} \pi_t^j \, \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} e_t & \text{if } (x_{t+1})_3 = 1 \\ 1 - e_t & \text{otherwise} \end{cases} \tag{23}$$

# Mixture density outputs and dataset details



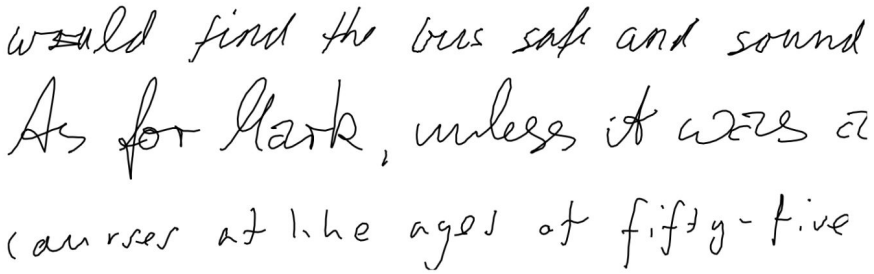
- IAM-OnDB dataset
- Train set ~5k lines
- Val set ~3k lines
- Test ~4k lines

The top heatmap shows the sequence of probability distributions for the predicted pen locations. Note that the blobs get larger at the end of the stroke since we do not know where the next letters begins.
The bottom heatmap shows the mixture weights.

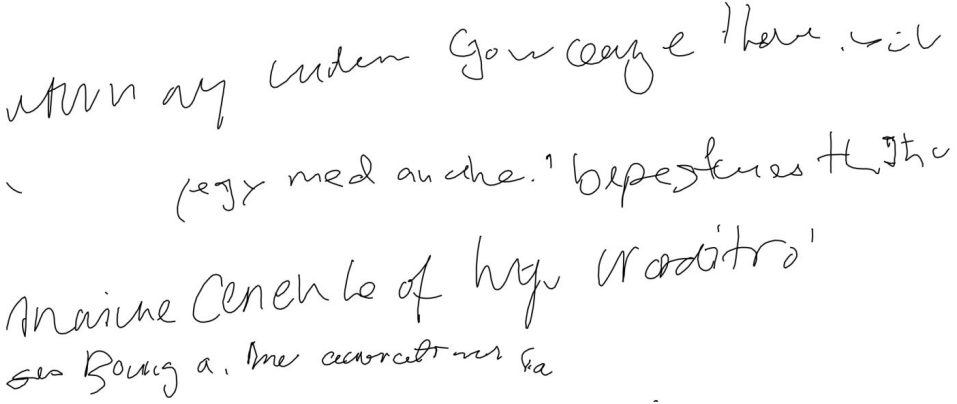
# Model info and Samples

- 20 mixture components for the offsets → 120 mixture parameters

Training data example:



Model prediction sample:



# Task 3: Text → Handwriting

Different problem than the one before since we want to **condition** the network result.

Difficulties arise from the different lengths that the sequences have, the network has to learn both to **align** and to translate.

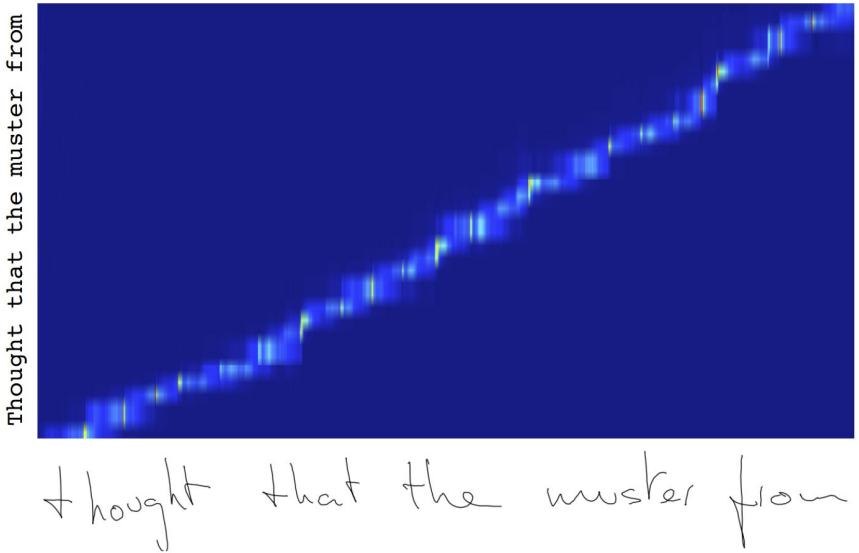
Idea: Let the network **learn the alignment** by letting the network choose what to focus → Convolve the text with a ‘soft window’ and feed the result to the LSTM as an extra input. The parameters of this window are themself output by the network at prediction time.

# Attention as a mixture of gaussians

$$\phi(t,u) = \sum_{k=1}^{K} \alpha_t^k \exp\left(-\beta_t^k \left(\kappa_t^k - u\right)^2\right)$$

$$w_t = \sum_{u=1}^{U} \phi(t,u) c_u$$

Visualizing $\phi(t,u)$




- **c** is the input text of length U
- **t** indices the current input fed into the netv
- $\phi(t,u)$ is the window weight of $c_u$ defined as a mixture of K gaussians
- $w_t$ is a discrete convolution between $\phi(t,u)$ and $c_u$

# Results

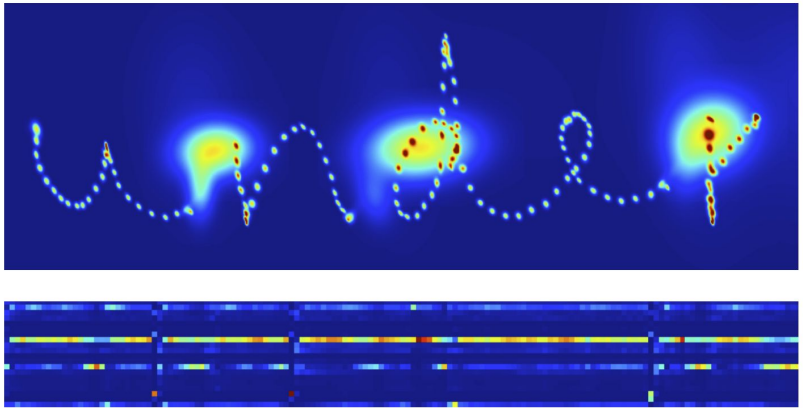
Same dataset as Handwriting generation: IAM-OnDB and a similar architecture in order to make comparisons.

3 hidden layers of 400 LSTM cells each 20 bivariate gaussians at the output layer + (new) Mixture of **10 gaussians** for the window parameters.

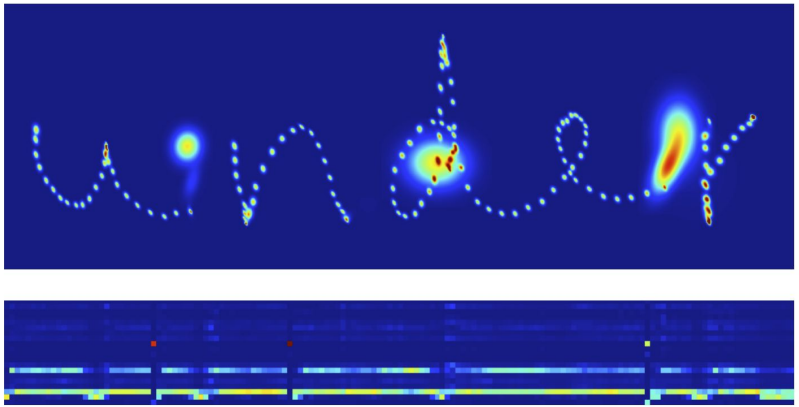
| Network | Regularisation | Log-Loss | SSE |
|---|---|---|---|
| 1 layer | none | -1025.7 | 0.40 |
| 3 layer | none | -1041.0 | 0.41 |
| 3 layer | adaptive weight noise | -1057.7 | 0.41 |

Current results

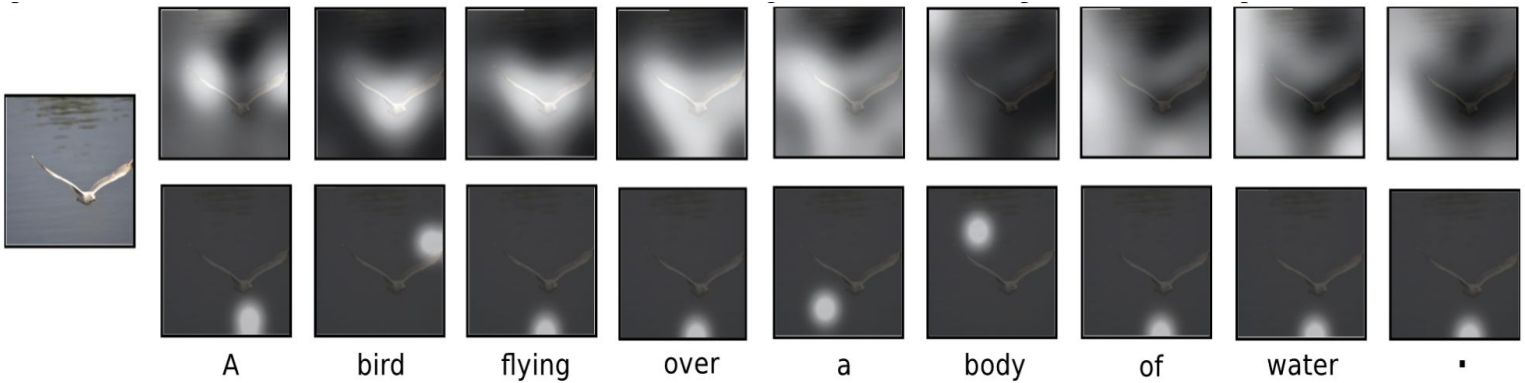
| Regularisation | Log-Loss | SSE |
|---|---|---|
| none | -1096.9 | 0.23 |
| adaptive weight noise | -1128.2 | 0.23 |



Current results



# Attention applied to different domains [7]





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.

- Also useful as a way to debug the model

# New methods for sequence transduction: Attention is all you need [8]

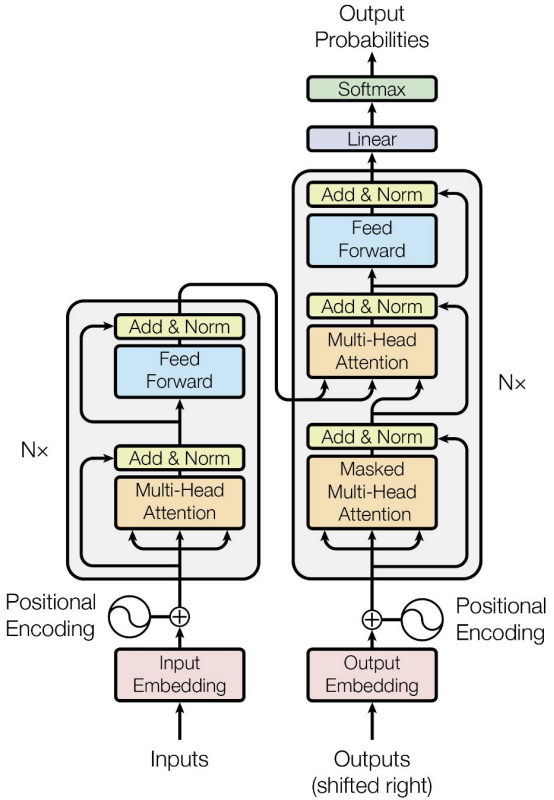
- Drops entirely the recurrent module
- Relies only on self-attention: Compute a fixed representation of a word influenced by all the other words in the sequence.

Encoder:

$$x_1, \ldots, x_2, \ldots x_n \quad \rightarrow \mathbf{z} = \quad z_1, \ldots, z_2, \ldots z_n$$

Decoder:

Given **z** the decoder generates $y_1, \ldots, y_2, \ldots y_n$ one element at a time



# How does it work?

- In short, we use the attention mechanism to produce a new representation for a word influenced by its context.
- More formally, for every word in the input sequence we recompute the representation using self-attention and then by passing this representation into a MLP to get the final one. → This operation is repeated N times and can be easily parallelized for different words.
- The decoder, also uses self-attention but it’s limited to words up to the one that is currently decoding (enforce causality).
- Finally, the decoder combines the word representation produced by the encoder with the representation of the previous word to produce the output.

# OpenAI GPT: Generative Pre-trained Transformers [9][10]

GPT-2 is a Transformer-based model trained to predict the next-word in 40GB of internet data (includes many different domains).

What makes it special is the scale at which it operates: 1.5 billions parameters scaled to 175 billions with GPT-3.

GPT-3 sample https://openai.com/blog/better-language-models/#sample1

| | |
|---|---|
| SYSTEM PROMPT (HUMAN-WRITTEN) | *In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.* |
| MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES) | The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.<br><br>Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.<br><br>Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow. |

# Wrap-up and Conclusions

- How and Why to use recurrent neural networks for sequences
- Why do we go Deep in recurrent neural network
- Training not so obvious → LSTM and MRNN
- Attention mechanism to condition the model
- Transformers

- The model architecture has a lot of influence on the performance our model can achieve
- Not an easy catch-all solution since we have multiple problems going from the predictive distribution to the learning/stability issues and the network topology.
- Recent results on models using the Transformers architecture are giving very good performance on a variety of tasks → Influence of the architecture

# References:


[1] Pascanu, R., Gulcehre, C., Cho, K., & Bengio, Y. (2014). How to construct deep recurrent neural networks. In Proceedings of the Second International Conference on Learning Representations (ICLR 2014)

[2] Hermans, Michiel and Schrauwen, Benjamin. (2021). Training and Analysing Deep Recurrent Neural Networks. In Advances in Neural Information Processing Systems

[3] Hochreiter, S., Bengio, Y., Frasconi, P. & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer & J. F. Kolen (ed.), A Field Guide to Dynamical Recurrent Neural Networks . IEEE Press

[4] Ilya Sutskever, James Martens, and Geoffrey Hinton. (2011). Generating text with recurrent neural networks. In Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11). Omnipress, Madison, WI, USA, 1017–1024.

[5] Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. ArXiv, abs/1308.0850.

[6] MIKOLOV, Tomáš. Statistical Language Models Based On Neural Networks. Brno, (2012). Ph.D. Thesis. Brno University of Technology, Faculty of Information Technology. 2012-10-02. Supervised by Černocký Jan. Available from: https://www.fit.vut.cz/study/phd-thesis/283/

[7] Xu, K. et al. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Proceedings of the 32nd International Conference on Machine Learning, in Proceedings of Machine Learning Research 37:2048-2057 Available from http://proceedings.mlr.press/v37/xuc15.html

[8] Ashish Vaswani et al. (2017). Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[9] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners.

[10] Brown et al. (2020). Language Models are Few-Shot Learners. In Advances in Neural Information Processing Systems

# Thank you for your attention!