DTU Course 02456 Deep learning
3 Recurrent neural networks

Ole Winther

Dept for Applied Mathematics and Computer Science
Technical University of Denmark (DTU)

August 31, 2016

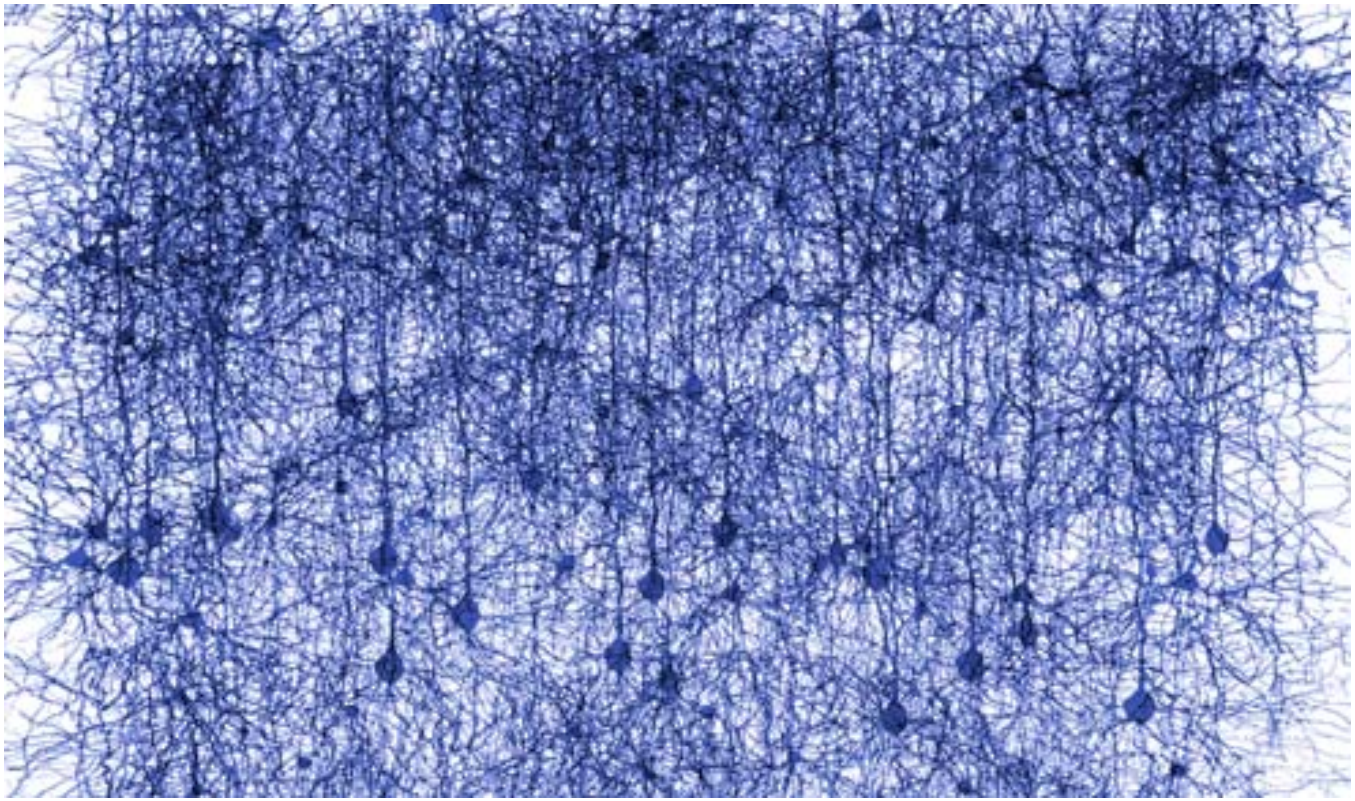
Objectives of lectures week 3

- P1: Recurrent neural networks
- what they can be used for and
- P2: how they are trained.
- P3: One example from my group's research:
  - Biological sequence analysis with attention mechanism
- P4: Recap supervised learning

Part 1:
Recurrent neural networks

Neural networks (NNs)

- Feedforward neural networks (FFNNs)
- Convolutional neural networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Auto-encoders (AE)

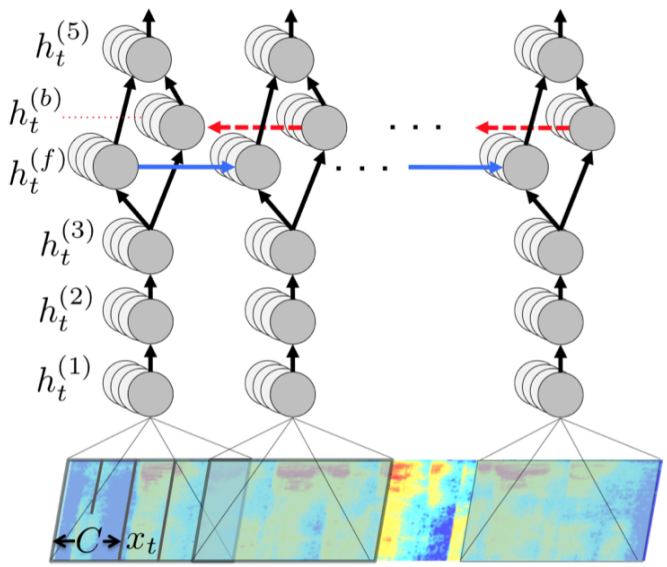


Recurrent neural networks – DeepSpeech

DeepSpeech: Scaling up end-to-end speech recognition

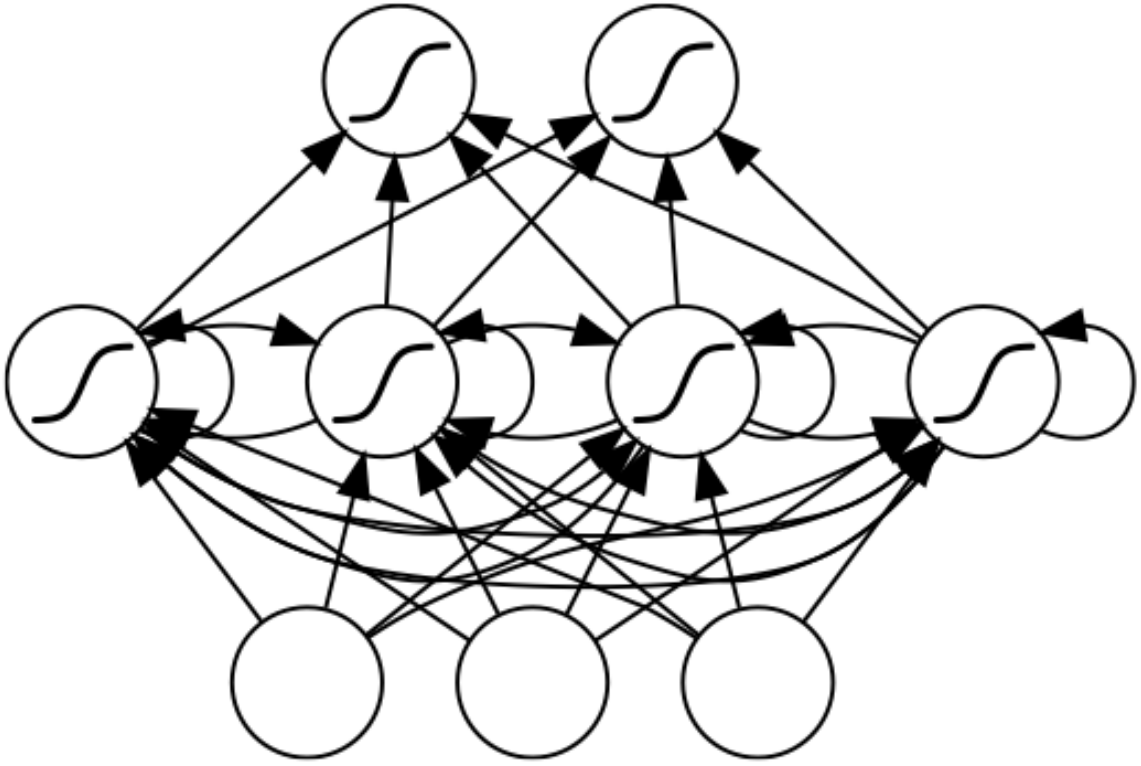
Awni Hannun,* Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, Andrew Y. Ng

Baidu Research – Silicon Valley AI Lab



Deep speech 2: ∼human level performance + realtime server

Recurrent neural networks

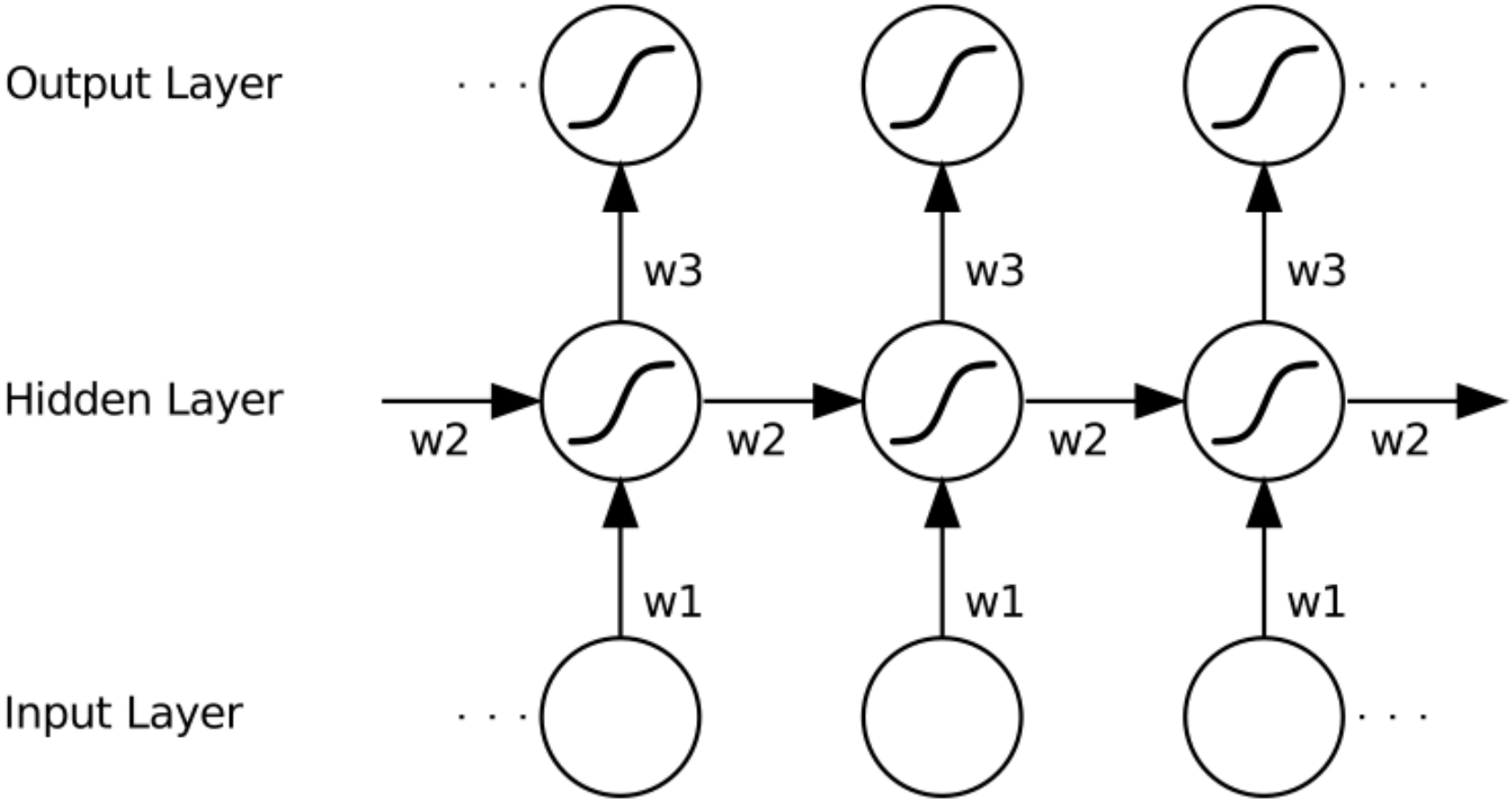


$$\mathbf{h}_t^{(1)} = f_1\left(W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}\mathbf{h}_{t-1}^{(1)}\right)$$

$$\mathbf{h}_t^{(2)} = f_2\left(W^{(2)}\mathbf{h}_t^{(1)}\right)$$

www.cs.toronto.edu/~graves/preprint.pdf

Recurrent neural networks unrolled

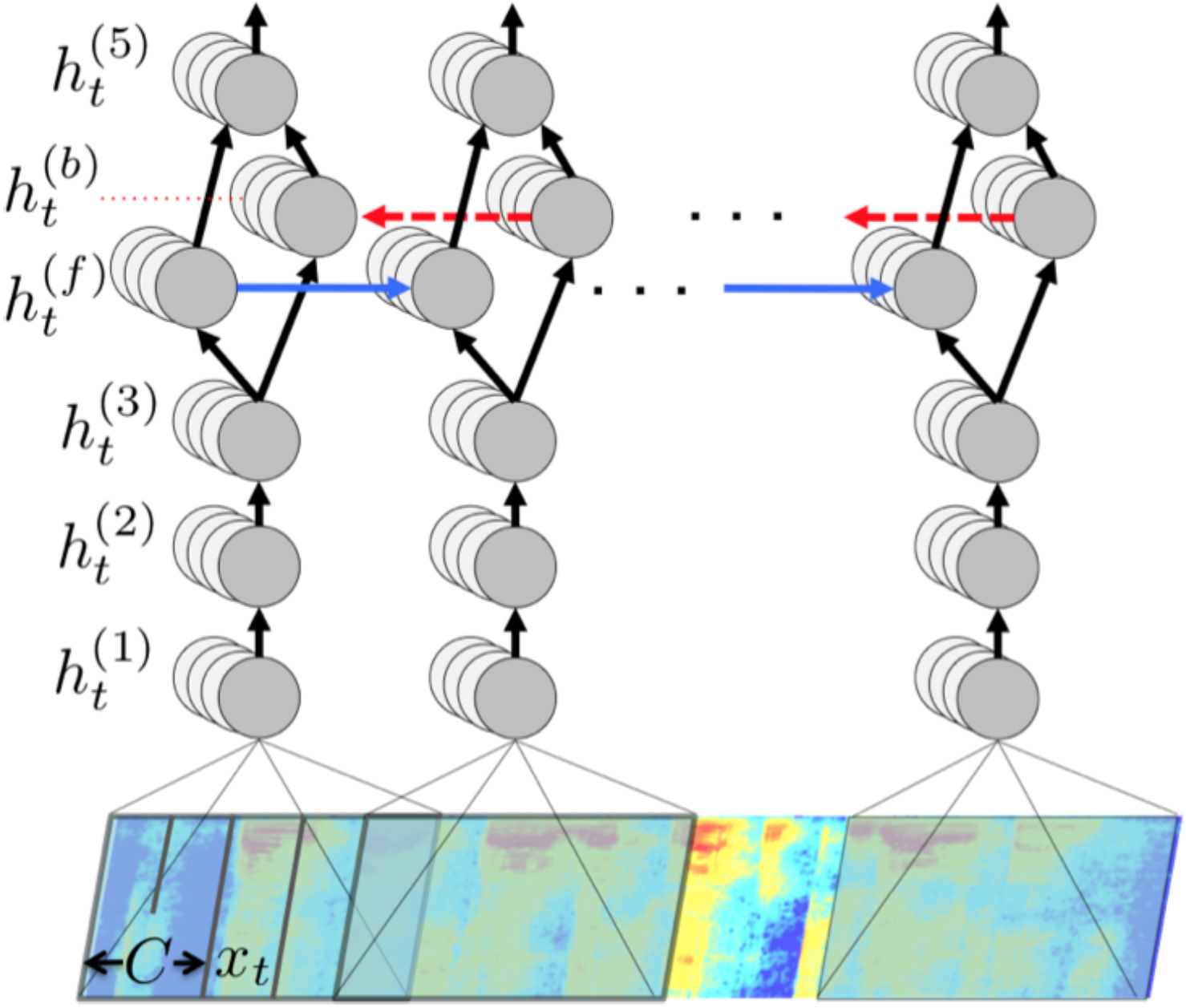


$$\mathbf{h}_t^{(1)} = f_1\left(W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}\mathbf{h}_{t-1}^{(1)}\right)$$

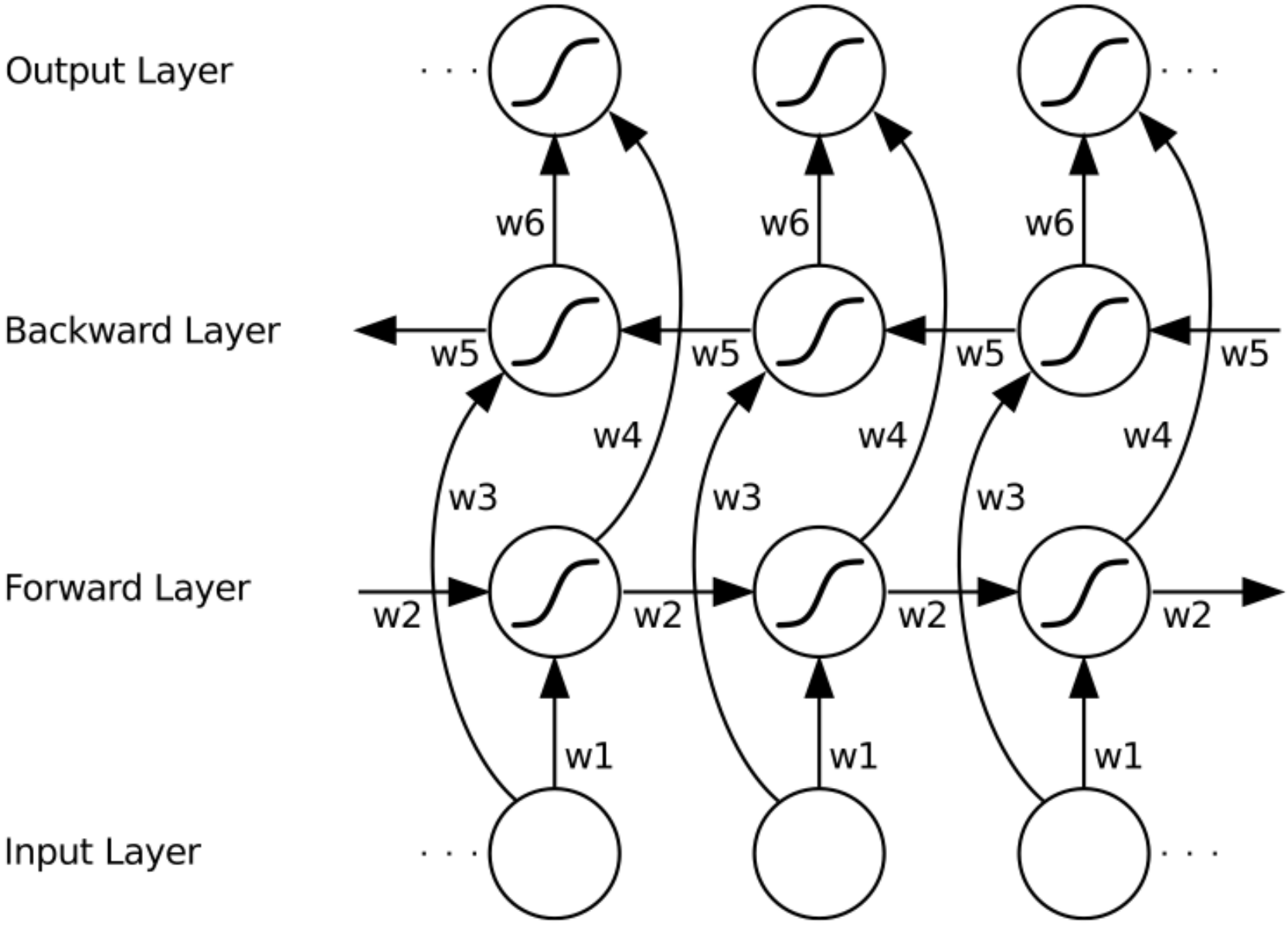
$$\mathbf{h}_t^{(2)} = f_2\left(W^{(2)}\mathbf{h}_t^{(1)}\right)$$

www.cs.toronto.edu/~graves/preprint.pdf

Bidirectional recurrent neural networks



Recurrent neural networks unrolled bidirectional



www.cs.toronto.edu/~graves/preprint.pdf

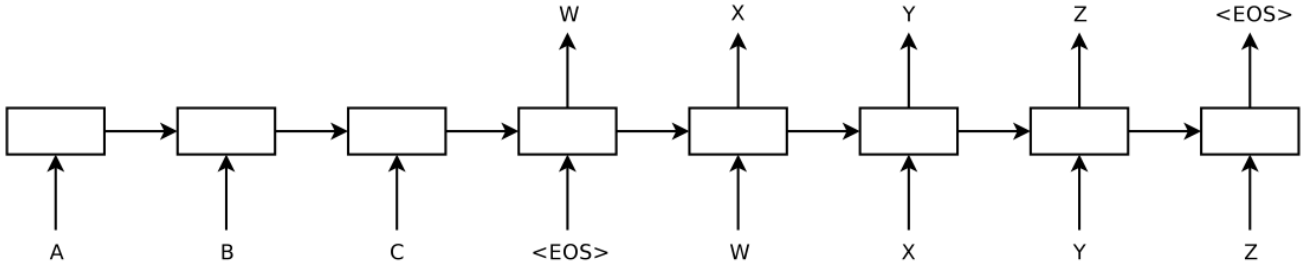
Encoder-decoder - machine translation

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com



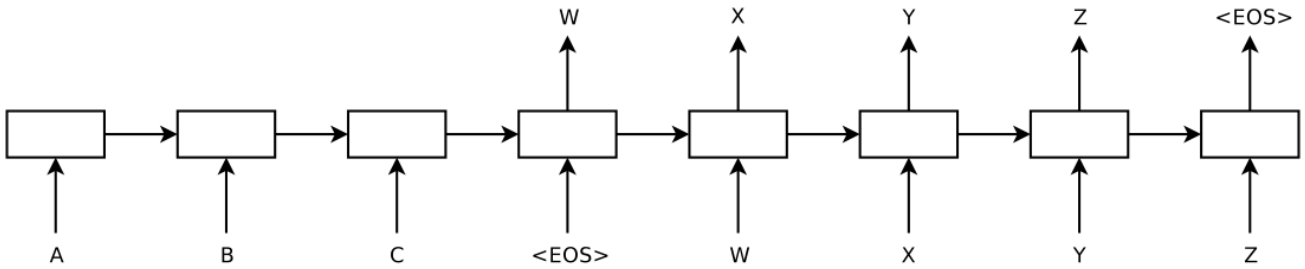
Encoder-decoder - machine translation

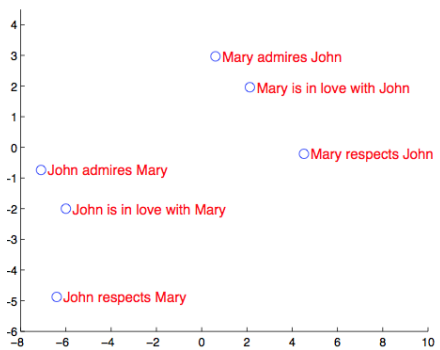
Sequence to Sequence Learning with Neural Networks

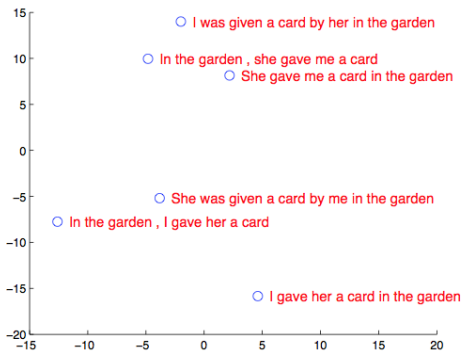
Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

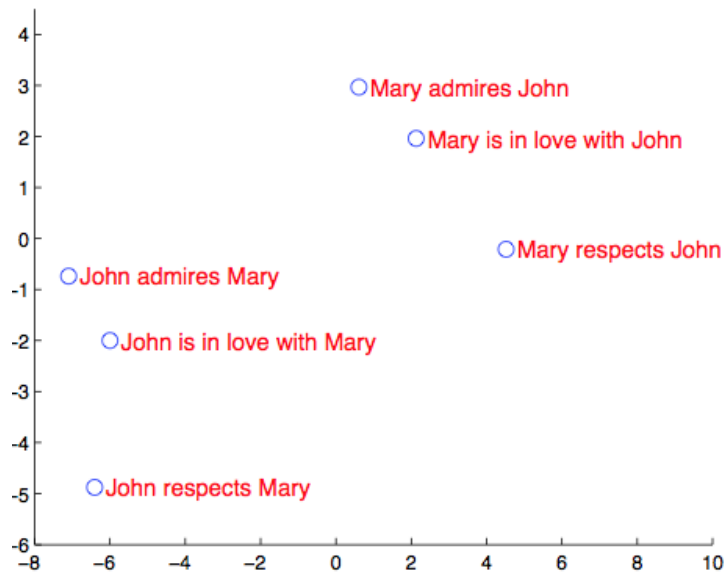
Quoc V. Le
Google
qvl@google.com

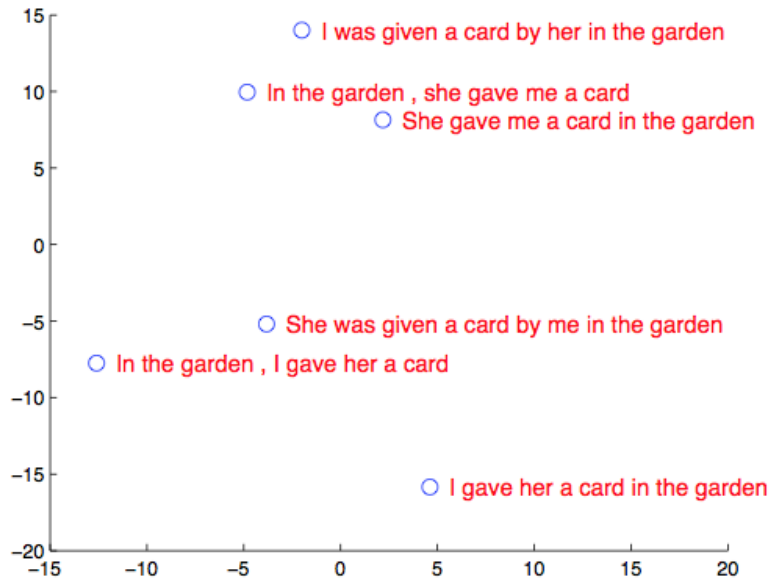






Machine translation - visualising latent space





Python interpretation with LSTM

Learning to Execute

Wojciech Zaremba WOJ.ZAREMBA@GMAIL.COM
Google & New York University
Ilya Sutskever ILYASU@GOOGLE.COM
Google

Python interpretation with LSTM

Learning to Execute

Wojciech Zaremba WOJ.ZAREMBA@GMAIL.COM
Google & New York University
Ilya Sutskever ILYASU@GOOGLE.COM
Google

Input:

```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```

Target: 25011.
Model prediction: 23011.

Input:

```
c=445
d=(c-4223)
for x in range(1):
  d+=5272
print((8942 if d<3749 else 2951))
```

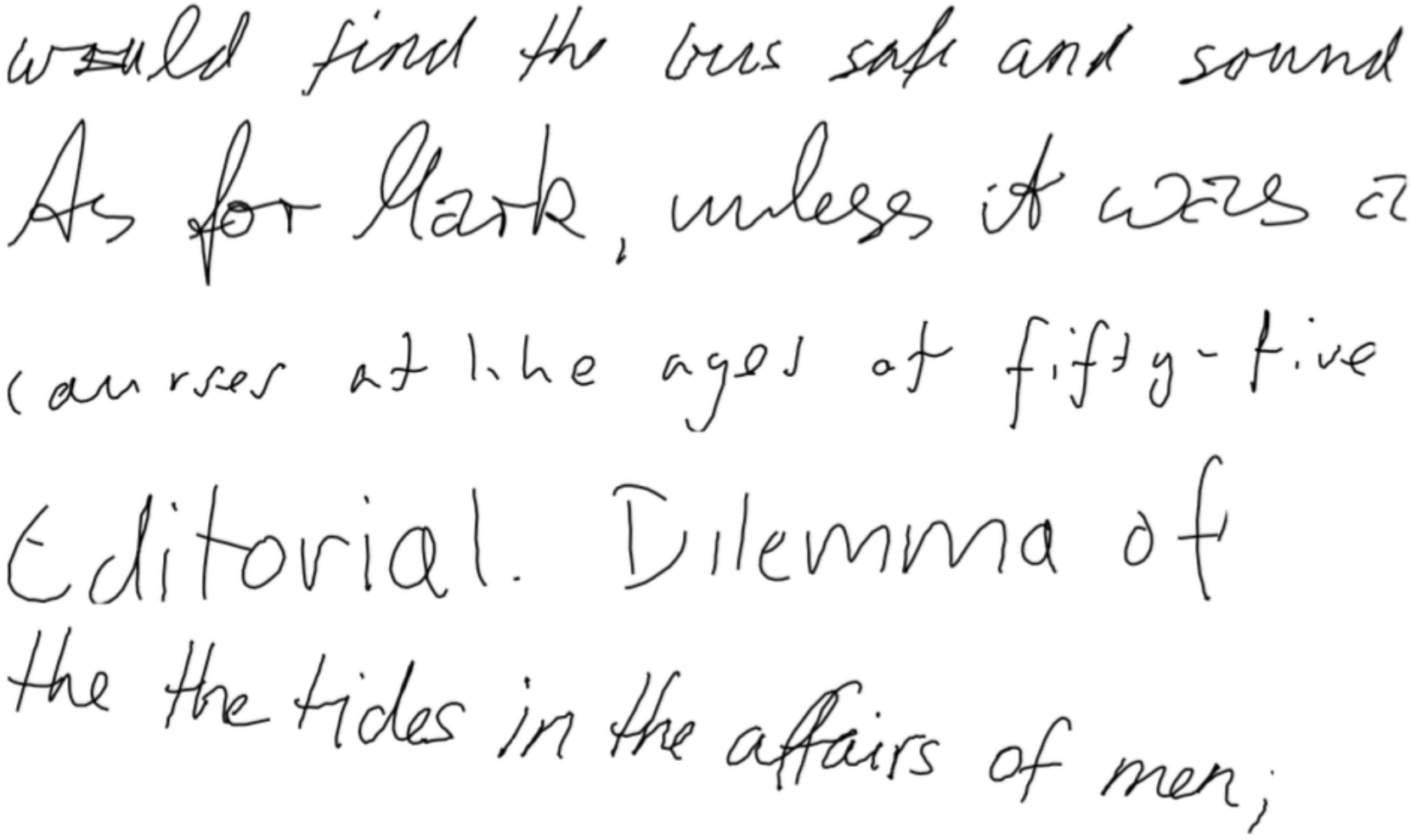
Target: 8942.
Model prediction: 8942.

Dialogue system

Machine: *hi*
Human: *hello*
Machine: *could you please let me know what are the operating systems you are using ?*
Human: *i am using linux*
Machine: *ok it was in the ssl vpn on which you stated it was broken*
Machine: *please check if you are logged in to corp network*
Machine: *what is the error that you are running please*
Human: *i am seeing an error related to vpn*
Machine: *what is the error message that you are getting when connecting to vpn using network connect ?*
Human: *connection refused or something like that*
Machine: *may i know the version of network connect you connect ?*
Human: *i am not sure i know that*
Machine: *is the network connect prompting for any user certificate*
Machine: *?*
Human: *yes*
Machine: *are you at home or at work at the moment ?*
Machine: *office ?*
Human: *i am at home*
Machine: *try this*
Machine: *goto <URL> page and enter your name and password*

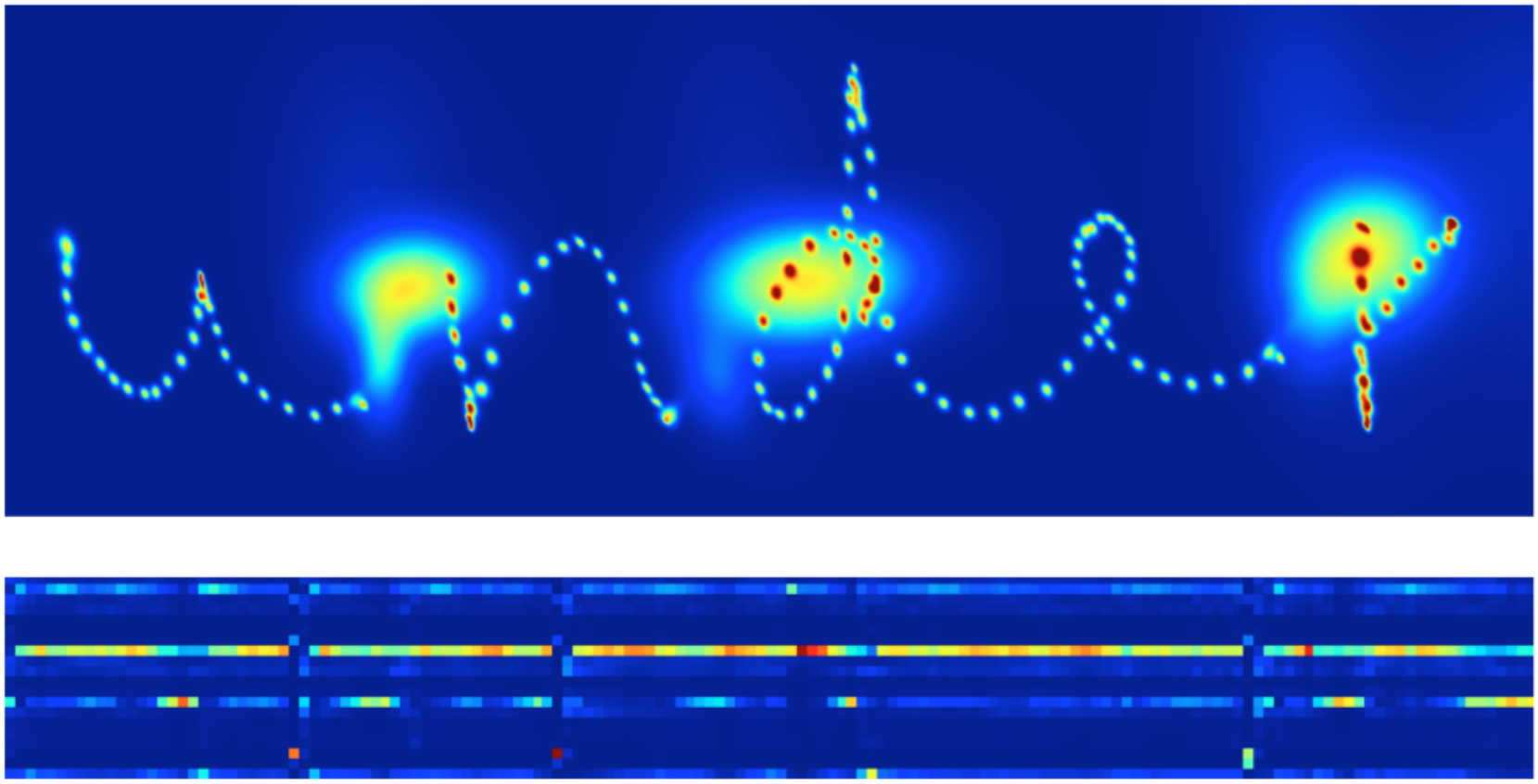
Many thanks to Pyry for sharing slides!

Generating sequences - training data $(x(t), y(t))$



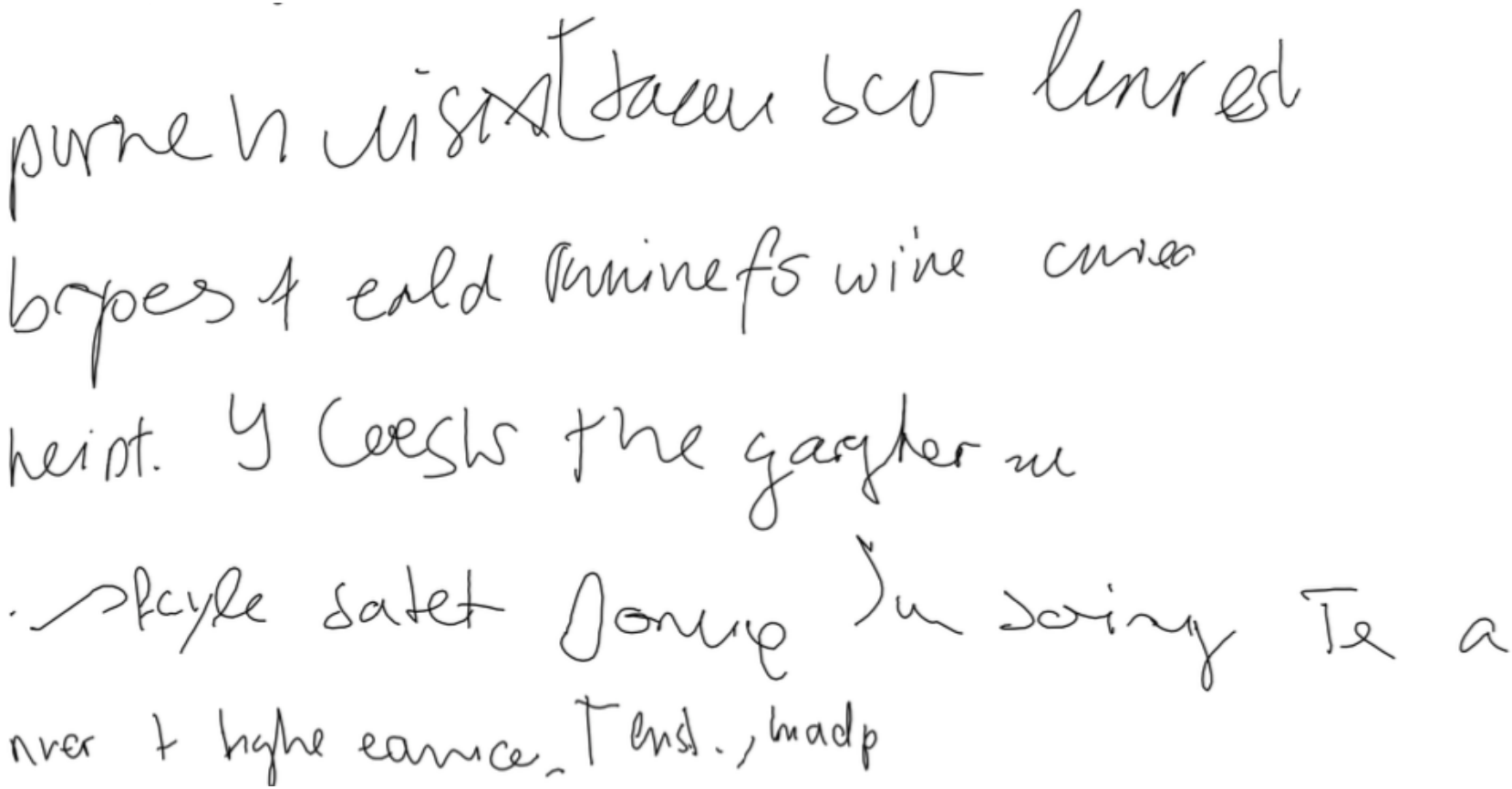
Alex Graves, Generating Sequences With Recurrent Neural Networks

Generating sequences - uncertainty of pen position



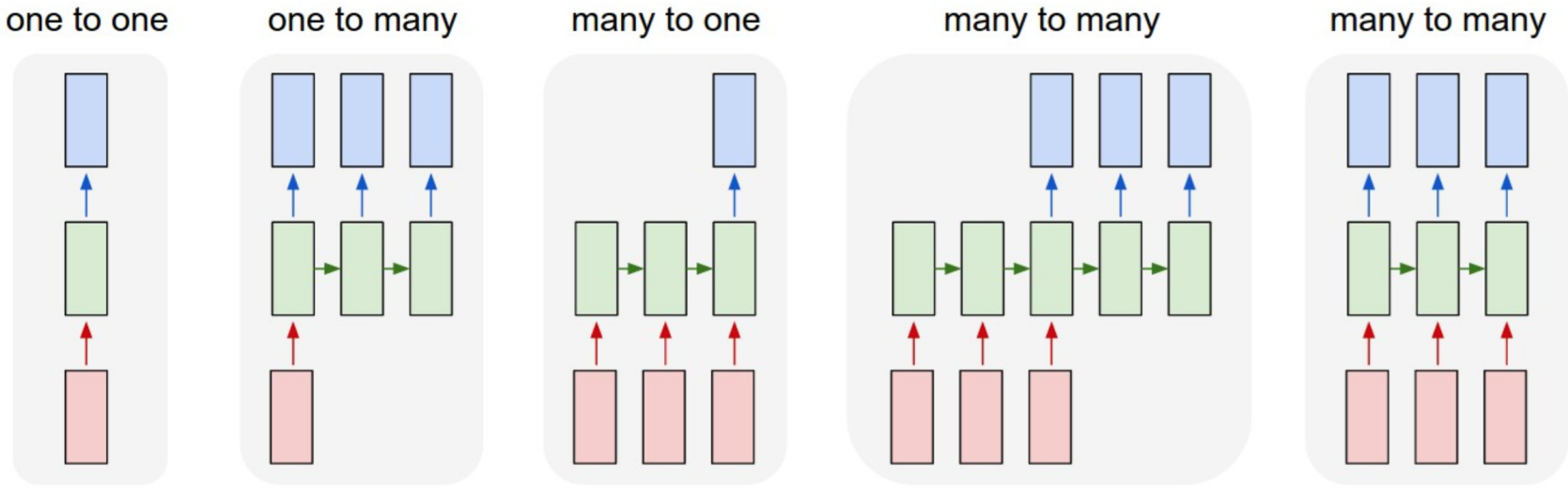
Alex Graves, Generating Sequences With Recurrent Neural Networks

Generating sequences - dreaming up new



Alex Graves, Generating Sequences With Recurrent Neural Networks
Demo

Overview recurrent achitectures



- Vanilla mode, no RNN.
- E.g. image classification

- Sequence output
- E.g. image captioning

- Sequence input
- E.g. sentiment analysis

- Sequence input and output (*encoder-decoder*, *sequence-to-sequence*)
- E.g. translation, question answering

- Synced sequence input and output
- E.g. label each video frame

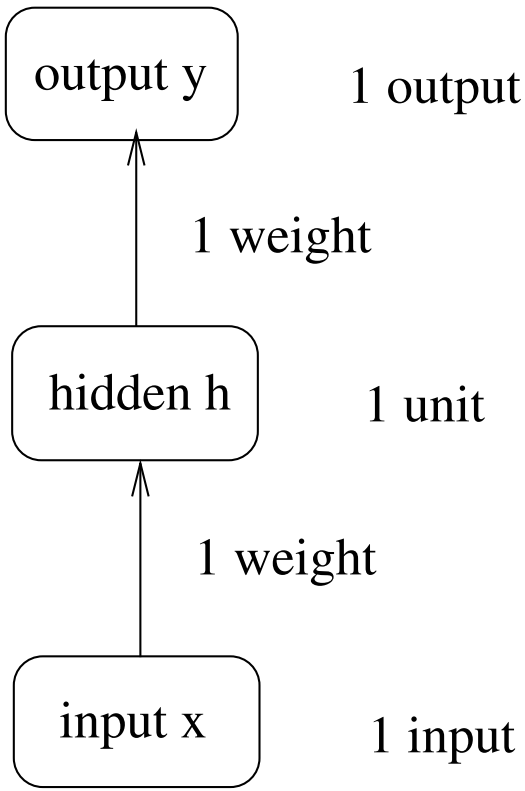
From Andrej Karpathy blog: The Unreasonable Effectiveness of Recurrent Neural Networks

Part 2:

RNN training Backpropagation through time

Recap backpropagation (Linnainmaa, 1970)

Computing gradients in a network.



- First with scalars. Use chain rule:

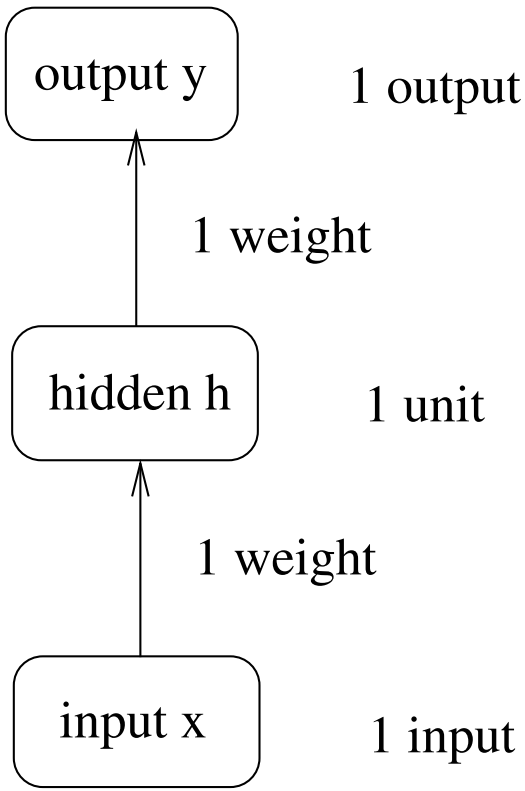
$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial w_2}$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial w_1}$$

- Chain rule: $\frac{\partial h^{(2)}}{\partial x} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial x}$

Recap backpropagation (Linnainmaa, 1970)

Computing gradients in a network.



- First with scalars. Use chain rule:

$$\frac{\partial C}{\partial w_2} = \frac{\partial C}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial w_2}$$

$$\frac{\partial C}{\partial w_1} = \frac{\partial C}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial w_1}$$

- Chain rule: $\frac{\partial h^{(2)}}{\partial x} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial x}$

Back-propagation through time

- RNN definition:

$$\mathbf{h}_t^{(1)} = f_1\left(W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}\mathbf{h}_{t-1}^{(1)}\right)$$

$$\mathbf{h}_t^{(2)} = f_2\left(W^{(2)}\mathbf{h}_t^{(1)}\right)$$

Back-propagation through time

- RNN definition:

$$\mathbf{h}_t^{(1)} = f_1\left(W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}\mathbf{h}_{t-1}^{(1)}\right)$$
$$\mathbf{h}_t^{(2)} = f_2\left(W^{(2)}\mathbf{h}_t^{(1)}\right)$$

- RNN weight derivatives:

$$\frac{\partial C}{\partial W^{(1,\rightarrow)}} = \sum_t \frac{\partial C}{\partial \mathbf{h}_t^{(1)}} \frac{\partial \mathbf{h}_t^{(1)}}{\partial W^{(1,\rightarrow)}}$$
$$\frac{\partial C}{\partial W^{(2)}} = \sum_t \frac{\partial C}{\partial \mathbf{h}_t^{(2)}} \frac{\partial \mathbf{h}_t^{(2)}}{\partial W^{(2)}}$$

Back-propagation through time

- RNN definition:

$$\mathbf{h}_t^{(1)} = f_1\left(W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}\mathbf{h}_{t-1}^{(1)}\right)$$
$$\mathbf{h}_t^{(2)} = f_2\left(W^{(2)}\mathbf{h}_t^{(1)}\right)$$

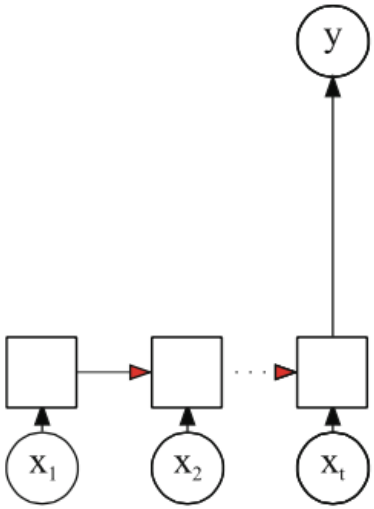
- RNN weight derivatives:

$$\frac{\partial C}{\partial W^{(1,\rightarrow)}} = \sum_t \frac{\partial C}{\partial \mathbf{h}_t^{(1)}} \frac{\partial \mathbf{h}_t^{(1)}}{\partial W^{(1,\rightarrow)}}$$
$$\frac{\partial C}{\partial W^{(2)}} = \sum_t \frac{\partial C}{\partial \mathbf{h}_t^{(2)}} \frac{\partial \mathbf{h}_t^{(2)}}{\partial W^{(2)}}$$

- Back-propagation through time:

$$\frac{\partial C}{\partial \mathbf{h}_t^{(1)}} = \frac{\partial C}{\partial \mathbf{h}_t^{(2)}} \frac{\partial \mathbf{h}_t^{(2)}}{\partial \mathbf{h}_t^{(1)}} + \frac{\partial C}{\partial \mathbf{h}_{t+1}^{(1)}} \frac{\partial \mathbf{h}_{t+1}^{(1)}}{\partial \mathbf{h}_t^{(1)}}$$

Gradients vanishing and exploding

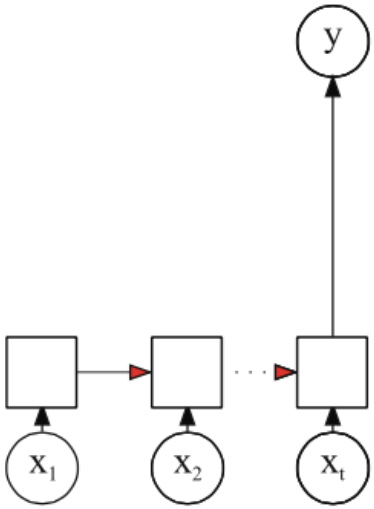


- Back-propagation through time:

$$\frac{\partial C}{\partial \mathbf{h}_t^{(1)}} = \frac{\partial C}{\partial \mathbf{h}_t^{(2)}} \frac{\partial \mathbf{h}_t^{(2)}}{\partial \mathbf{h}_t^{(1)}} + \frac{\partial C}{\partial \mathbf{h}_{t+1}^{(1)}} \frac{\partial \mathbf{h}_{t+1}^{(1)}}{\partial \mathbf{h}_t^{(1)}}$$

- Scalars $H = K = 1$ and length T
- $\frac{\partial C}{\partial h_t^{(2)}} = 0$ for $t < T$

Gradients vanishing and exploding



- Back-propagation through time:

$$\frac{\partial C}{\partial \mathbf{h}_t^{(1)}} = \frac{\partial C}{\partial \mathbf{h}_t^{(2)}} \frac{\partial \mathbf{h}_t^{(2)}}{\partial \mathbf{h}_t^{(1)}} + \frac{\partial C}{\partial \mathbf{h}_{t+1}^{(1)}} \frac{\partial \mathbf{h}_{t+1}^{(1)}}{\partial \mathbf{h}_t^{(1)}}$$

- Scalars $H = K = 1$ and length T
- $\frac{\partial C}{\partial h_t^{(2)}} = 0$ for $t < T$

- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_T^{(1)}} = \frac{\partial C}{\partial h_T^{(2)}} f_2' \left(W^{(2)} h_T^{(1)} \right) W^{(2)}$$

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

Gradients vanishing and exploding continued

- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

Gradients vanishing and exploding continued

- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

- Vanishing gradient: $|f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}| \ll 1$ - examples
- $|W^{(\rightarrow)}| \ll 1$

Gradients vanishing and exploding continued

- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

- Vanishing gradient: $|f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}| \ll 1$ - examples
- $|W^{(\rightarrow)}| \ll 1$
- $f(a) = \tanh(a)$, $f'(a) = 1 - f^2(a) \ll 1$ for $|a| \geq 3$.

Gradients vanishing and exploding continued

- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

- Vanishing gradient: $|f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}| \ll 1$ - examples
- $|W^{(\rightarrow)}| \ll 1$
- $f(a) = \tanh(a)$, $f'(a) = 1 - f^2(a) \ll 1$ for $|a| \geq 3$.
- Exploding gradient: $|W^{(\rightarrow)} f'(a_t^{(1)})| > 1$ - example
- $f(a) = \text{relu}(a) = \max(0, a)$, $f'(a) = (a > 0)$, $a_t^{(1)} > 0$ and $|W^{(\rightarrow)}| > 1$.

Gradients vanishing and exploding continued

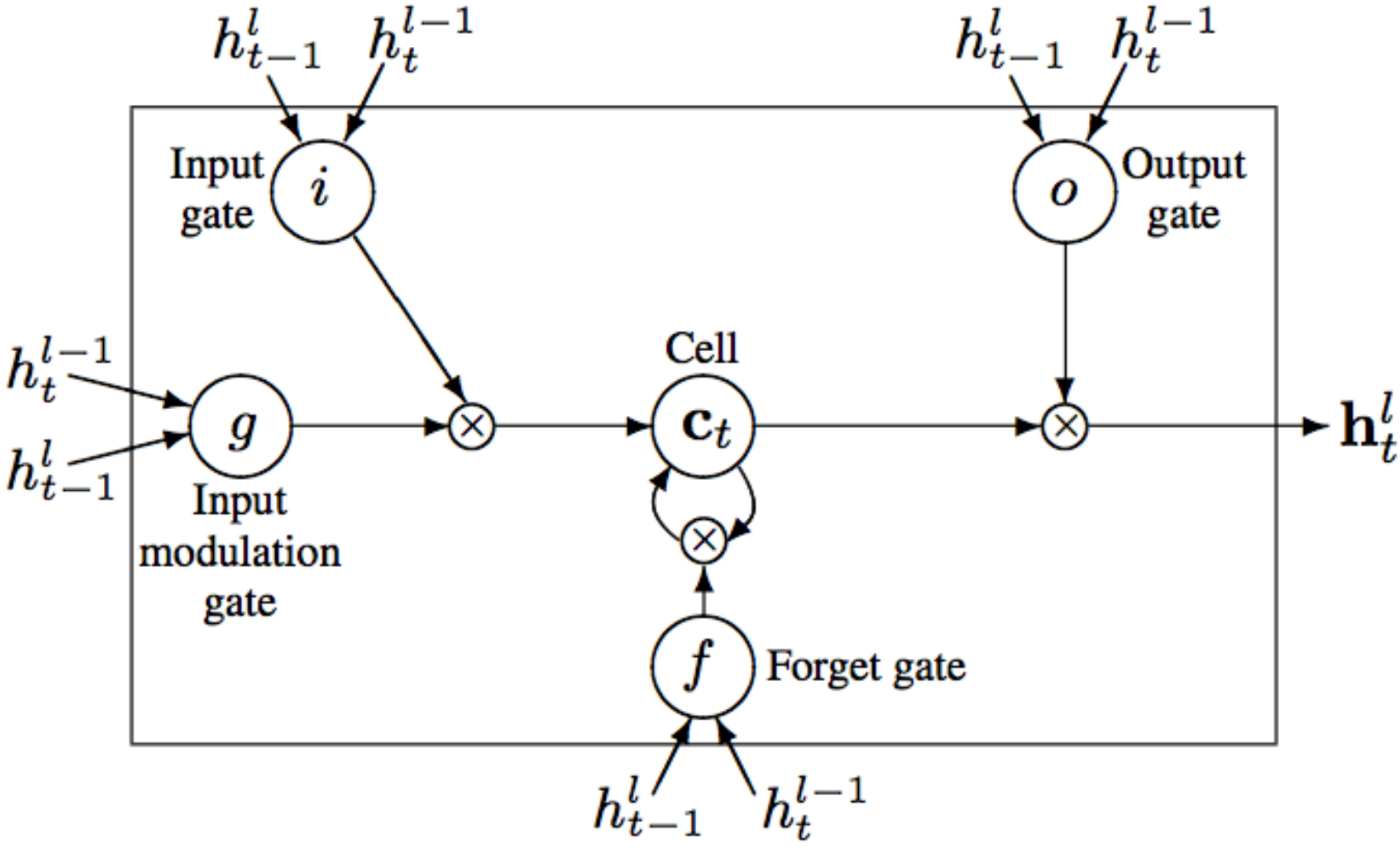
- Simplified back-propagation through time - ($t < T$):

$$\frac{\partial C}{\partial h_t^{(1)}} = \frac{\partial C}{\partial h_{t+1}^{(1)}} f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}$$

$$a_t^{(1)} \equiv W^{(1)} x_t + W^{(\rightarrow)} h_{t-1}^{(1)}$$

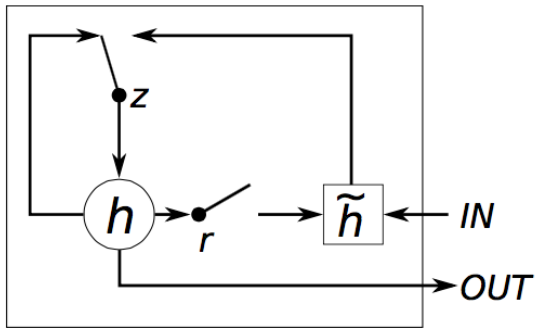
- Vanishing gradient: $|f_1' \left(a_t^{(1)} \right) W^{(\rightarrow)}| \ll 1$ - examples
- $|W^{(\rightarrow)}| \ll 1$
- $f(a) = \tanh(a)$, $f'(a) = 1 - f^2(a) \ll 1$ for $|a| \geq 3$.
- Exploding gradient: $|W^{(\rightarrow)} f'(a_t^{(1)})| > 1$ - example
- $f(a) = \text{relu}(a) = \max(0, a)$, $f'(a) = (a > 0)$, $a_t^{(1)} > 0$ and $|W^{(\rightarrow)}| > 1$.
- Partial solution: gated units!

Long short term memory cells



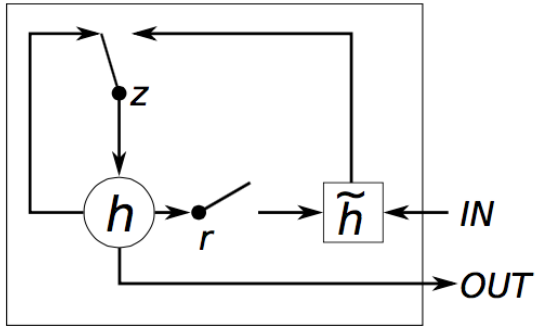
Hochreiter and Schmidhuber, 1997.

Gated Feedback Recurrent Neural Networks



- Chung et al, 2015

Gated Feedback Recurrent Neural Networks

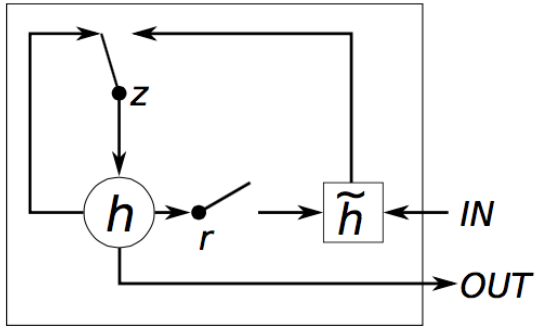


- Chung et al, 2015

$$\mathbf{r}_t^{(1)} = \sigma\left(U^{(1)}\mathbf{x}_t + U^{(\rightarrow)}\mathbf{a}_{t-1}^{(1)}\right)$$
$$\mathbf{z}_t^{(1)} = \sigma\left(V^{(1)}\mathbf{x}_t + V^{(\rightarrow)}\mathbf{a}_{t-1}^{(1)}\right)$$
$$\tilde{\mathbf{a}}_t^{(1)} = W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}(\mathbf{r}_t^{(1)} \circ \mathbf{a}_{t-1}^{(1)})$$
$$\mathbf{a}_t^{(1)} = (1 - \mathbf{z}_t^{(1)}) \circ \mathbf{a}_{t-1}^{(1)} + \mathbf{z}_t^{(1)} \circ f_1(\tilde{\mathbf{a}}_t^{(1)})$$

- $\sigma()$ is the logistic function $\in [0, 1]$.

Gated Feedback Recurrent Neural Networks

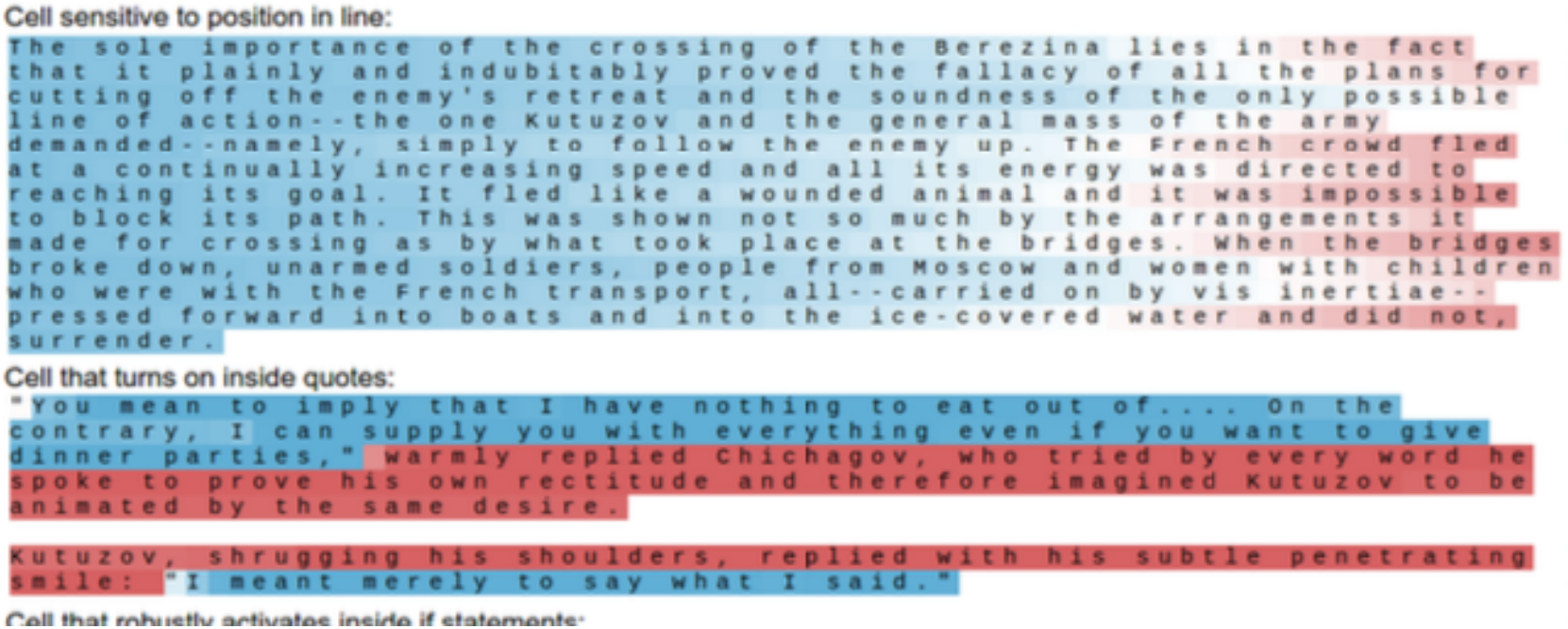


- Chung et al, 2015

$$\mathbf{r}_t^{(1)} = \sigma\left(U^{(1)}\mathbf{x}_t + U^{(\rightarrow)}\mathbf{a}_{t-1}^{(1)}\right)$$
$$\mathbf{z}_t^{(1)} = \sigma\left(V^{(1)}\mathbf{x}_t + V^{(\rightarrow)}\mathbf{a}_{t-1}^{(1)}\right)$$
$$\tilde{\mathbf{a}}_t^{(1)} = W^{(1)}\mathbf{x}_t + W^{(\rightarrow)}(\mathbf{r}_t^{(1)} \circ \mathbf{a}_{t-1}^{(1)})$$
$$\mathbf{a}_t^{(1)} = (1 - \mathbf{z}_t^{(1)}) \circ \mathbf{a}_{t-1}^{(1)} + \mathbf{z}_t^{(1)} \circ f_1(\tilde{\mathbf{a}}_t^{(1)})$$

- $\sigma()$ is the logistic function $\in [0, 1]$.
- Note that if $(\mathbf{z}_t^{(1)})_i = 0$ then $(\mathbf{a}_t^{(1)})_i = (\mathbf{a}_{t-1}^{(1)})_i$

Inside and LSTM cell



Karpathy et al, 2016

Inside and LSTM cell 2

Cell that turns on inside comments and quotes:

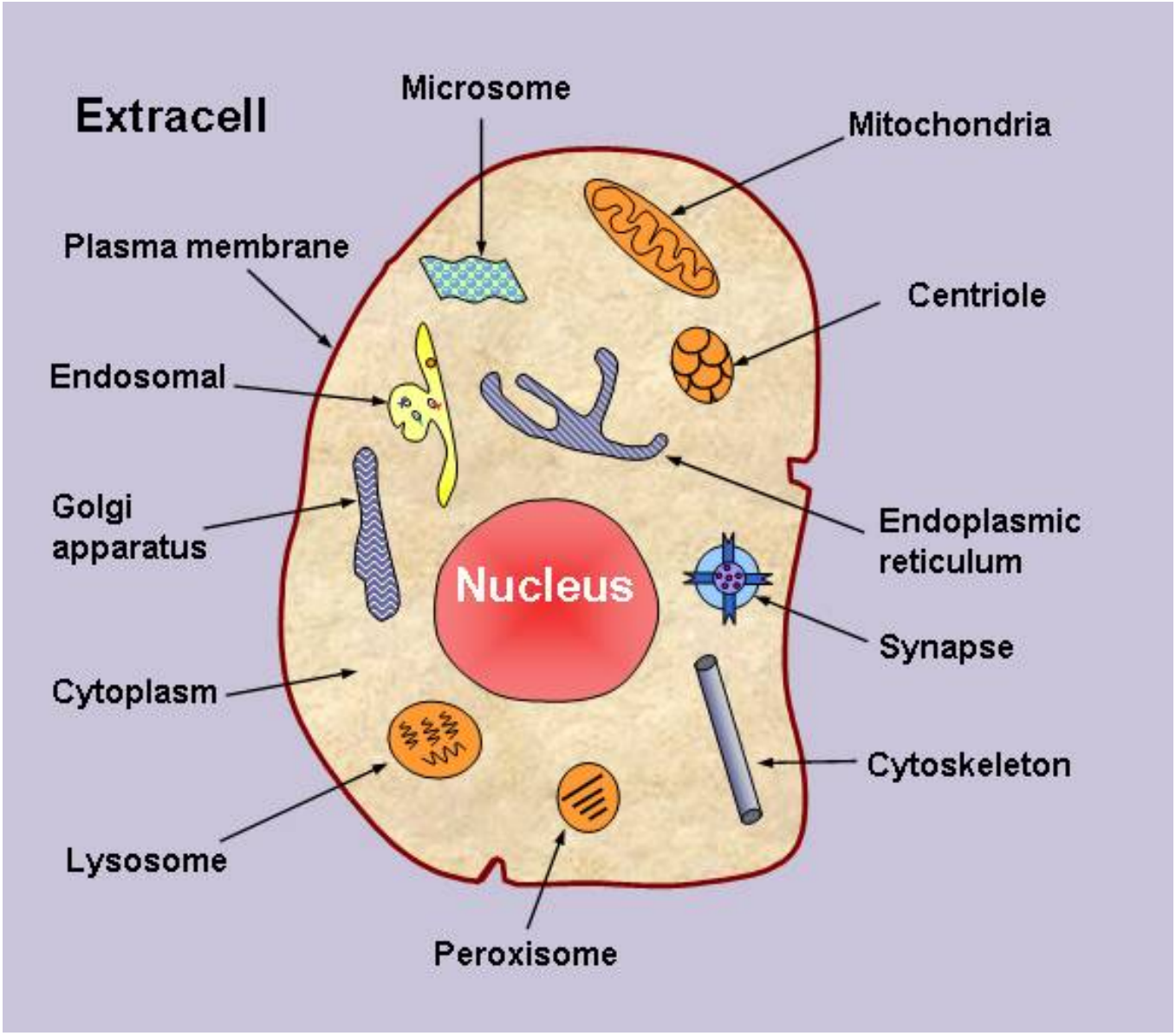
```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
           struct audit_field *sf)
{
  int ret = 0;
  char *lsm_str;
  /* our own copy of lsm_str */
  lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
  if (unlikely(!lsm_str))
    return -ENOMEM;
  df->lsm_str = lsm_str;
  /* our own (refreshed) copy of lsm_rule */
  ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                 (void **)&df->lsm_rule);
  /* Keep currently invalid fields around in case they
   * become valid after a policy reload. */
  if (ret == -EINVAL) {
    pr_warn("audit rule for LSM \'%s\' is invalid\n",
      df->lsm_str);
    ret = 0;
  }
  return ret;
}
```

Karpathy et al, 2016

Part 3:

Deep learning for biological sequence analysis

Sub-cellular localization



Sub-cellular localization machine learning set-up

- Use (one-hot encoded) protein sequence as input

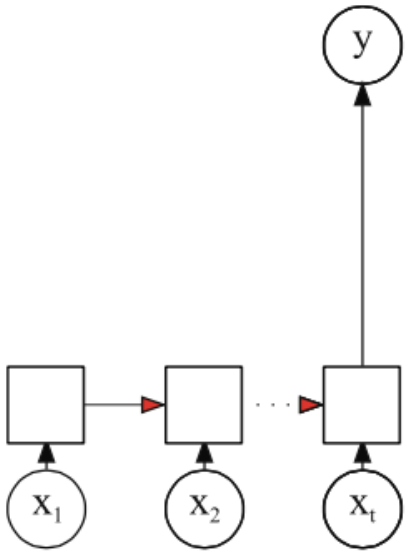
```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Sub-cellular localization machine learning set-up

- Use (one-hot encoded) protein sequence as input

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

- 11 localization classes softmax output
- RNN architecture - needs to memorize to the end!

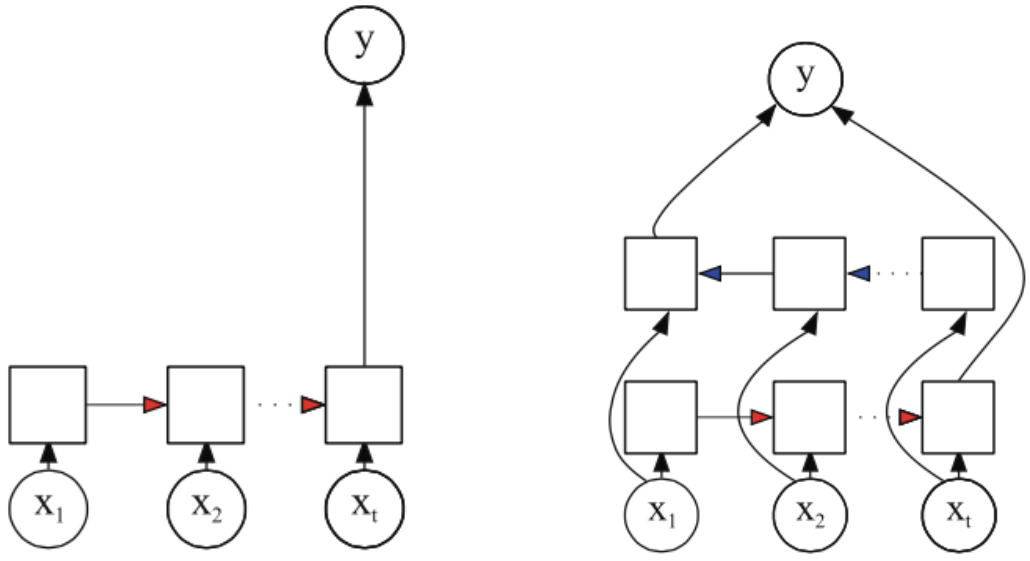


Sub-cellular localization machine learning set-up

- Use (one-hot encoded) protein sequence as input

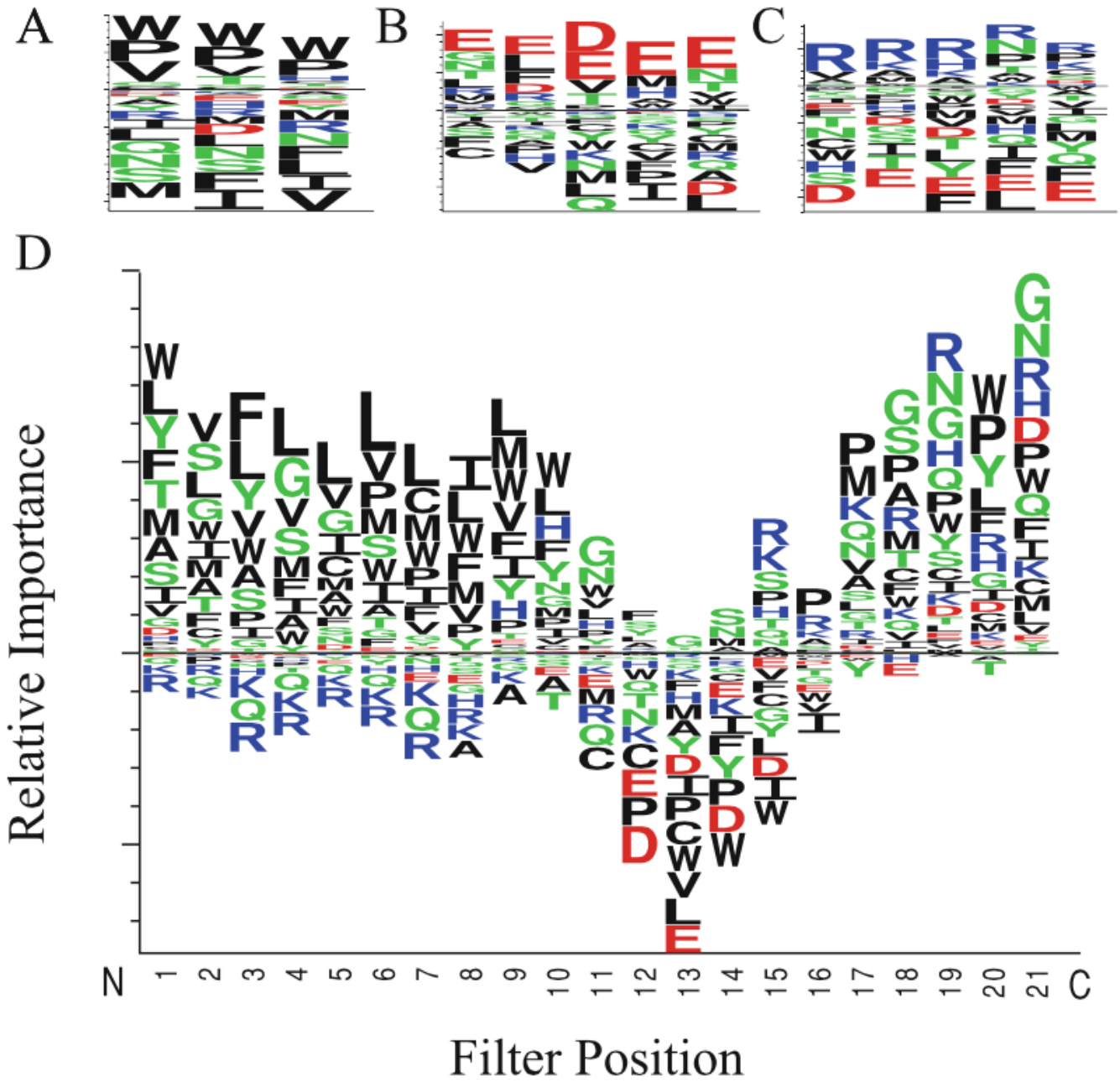
```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

- 11 localization classes softmax output
- RNN architecture - needs to memorize to the end!



- First layer is a convolutional layer
- Sonderby et. al. `http://arxiv.org/abs/1503.01919`

Visualizing learned convolutions



Advanced feature - attention

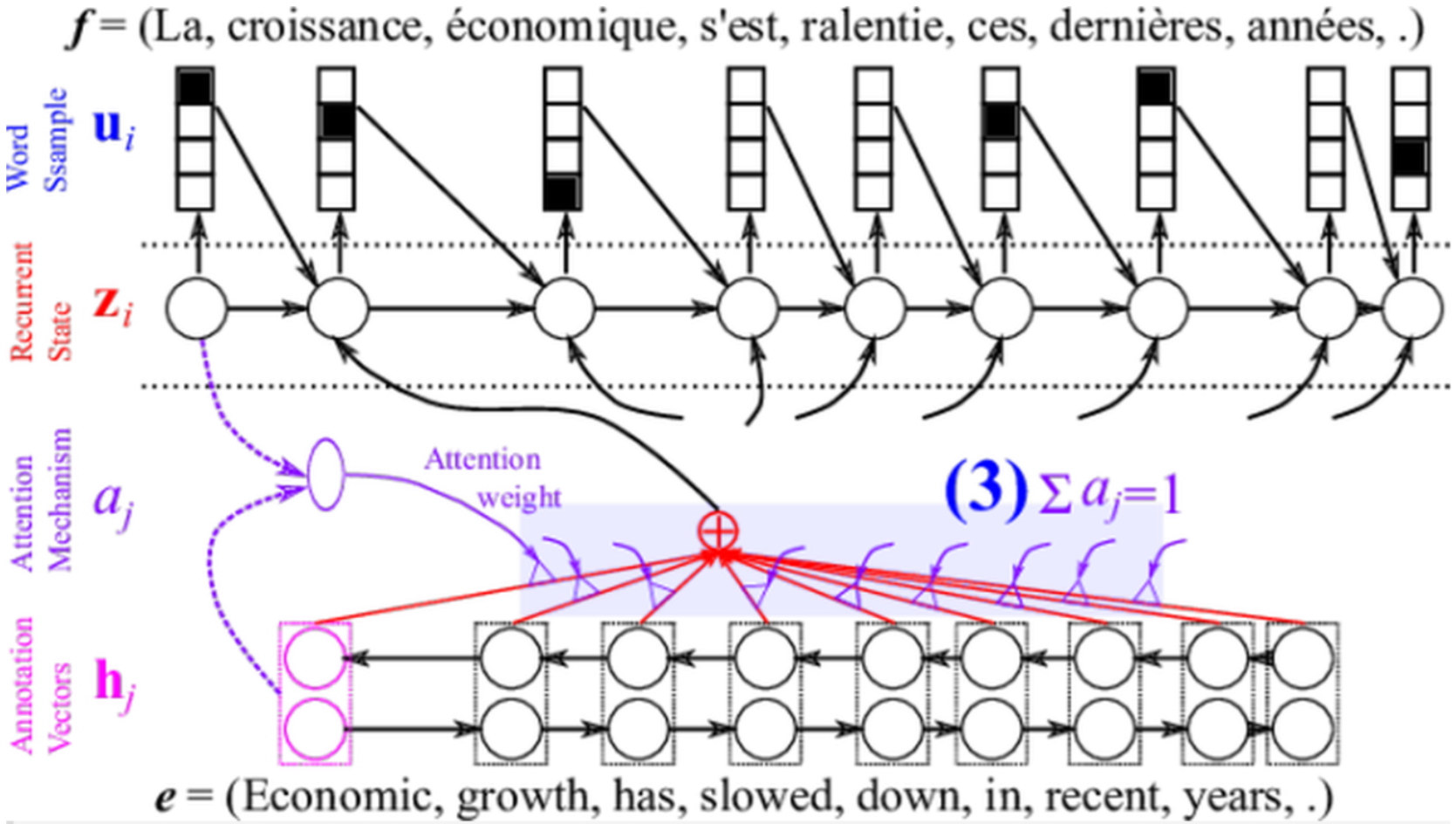


Figure 5. The relevance scores returned by the attention mechanism are normalized to sum to 1, which helps us interpret them as probabilities. From this probabilistic perspective, we compute the expectation of the annotation vectors under this distribution.

Advanced feature - attention

by *ent423* , *ent261* correspondent updated 9:49 pm et , thu march 19 , 2015 (*ent261*) a *ent114* was killed in a parachute accident in *ent45* , *ent85* , near *ent312* , a *ent119* official told *ent261* on wednesday . he was identified thursday as special warfare operator 3rd class *ent23* , 29 , of *ent187* , *ent265* . `` *ent23* distinguished himself consistently throughout his career . he was the epitome of the quiet professional in all facets of his life , and he leaves an inspiring legacy of natural tenacity and focused

. . .

ent119 identifies deceased sailor as **X** , who leaves behind a wife

A woman is throwing a frisbee in a park.

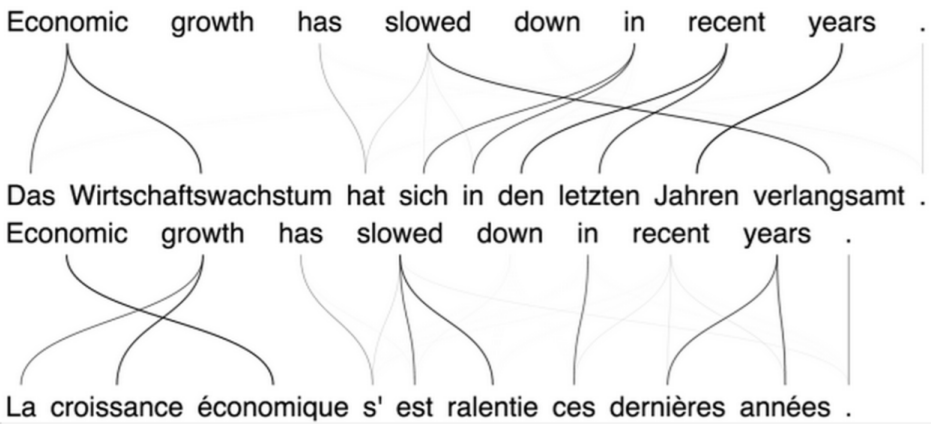
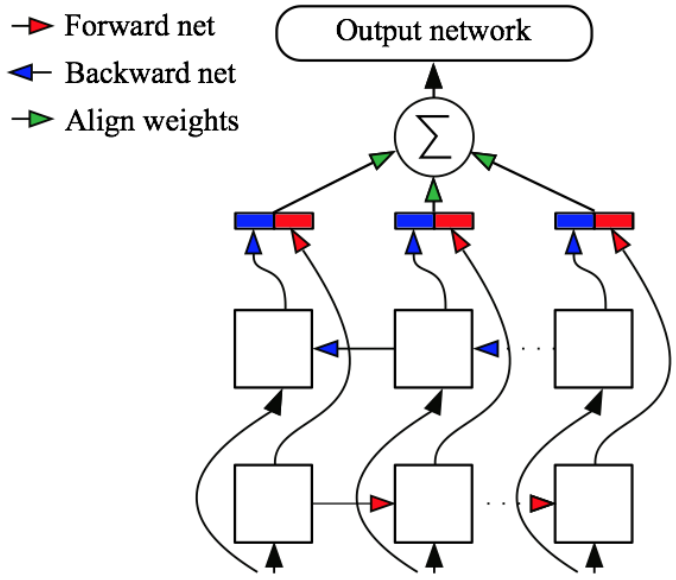


Figure 6. Sample translations made by the neural machine translation model with the soft-attention mechanism. Edge thicknesses represent the attention weights found by the attention model.

Attention mechanism

- Attention mechanism (adapted from Bahdanau et al, 2014)

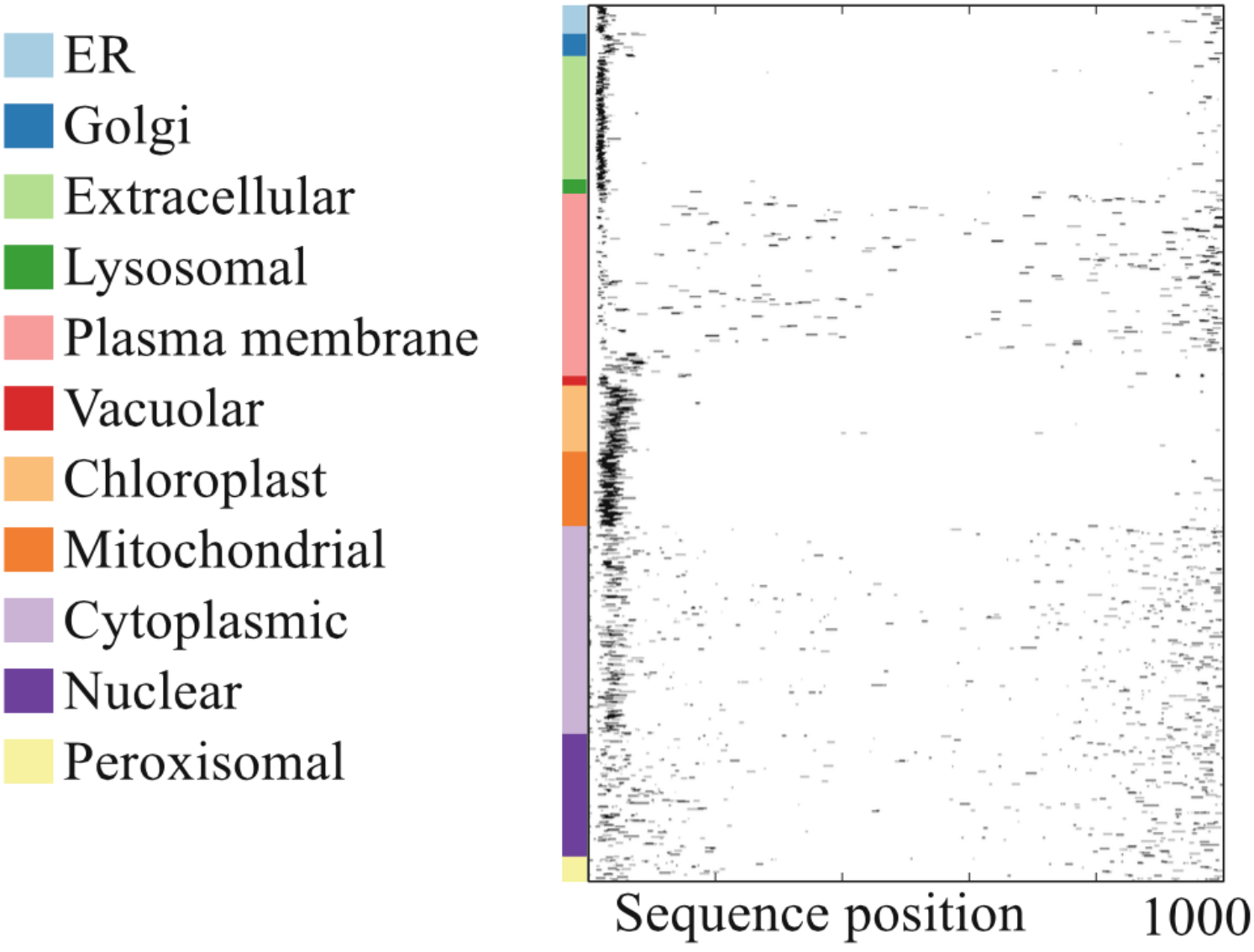


- Compute context vector c, $\dim(c) = \dim(h_t)$, $h_t = \begin{bmatrix} h_t^{(\mathrm{f})} \\ h_t^{(\mathrm{b})} \end{bmatrix}$

$$c = \sum_{t=1}^{T_x} a_t h_t \qquad a_t = \frac{\exp(f(h_t; W))}{\sum_{t'=1}^{T_x} \exp(f(h_{t'}; W))}$$

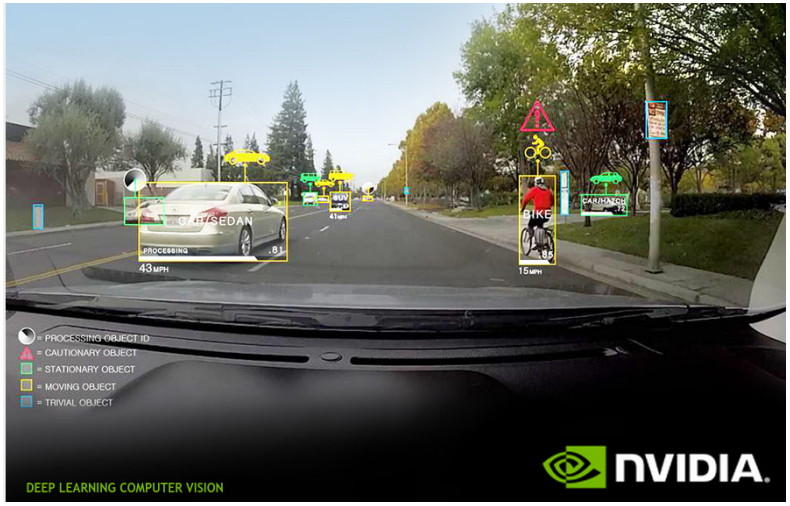
that are fed into feed-forward network.

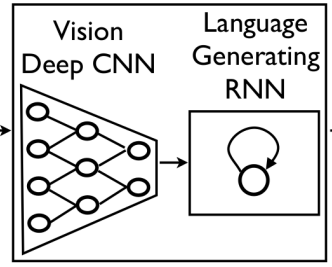
Where does the network pay attention?



Part 4:
Supervised learning recap

Deep learning is changing the world





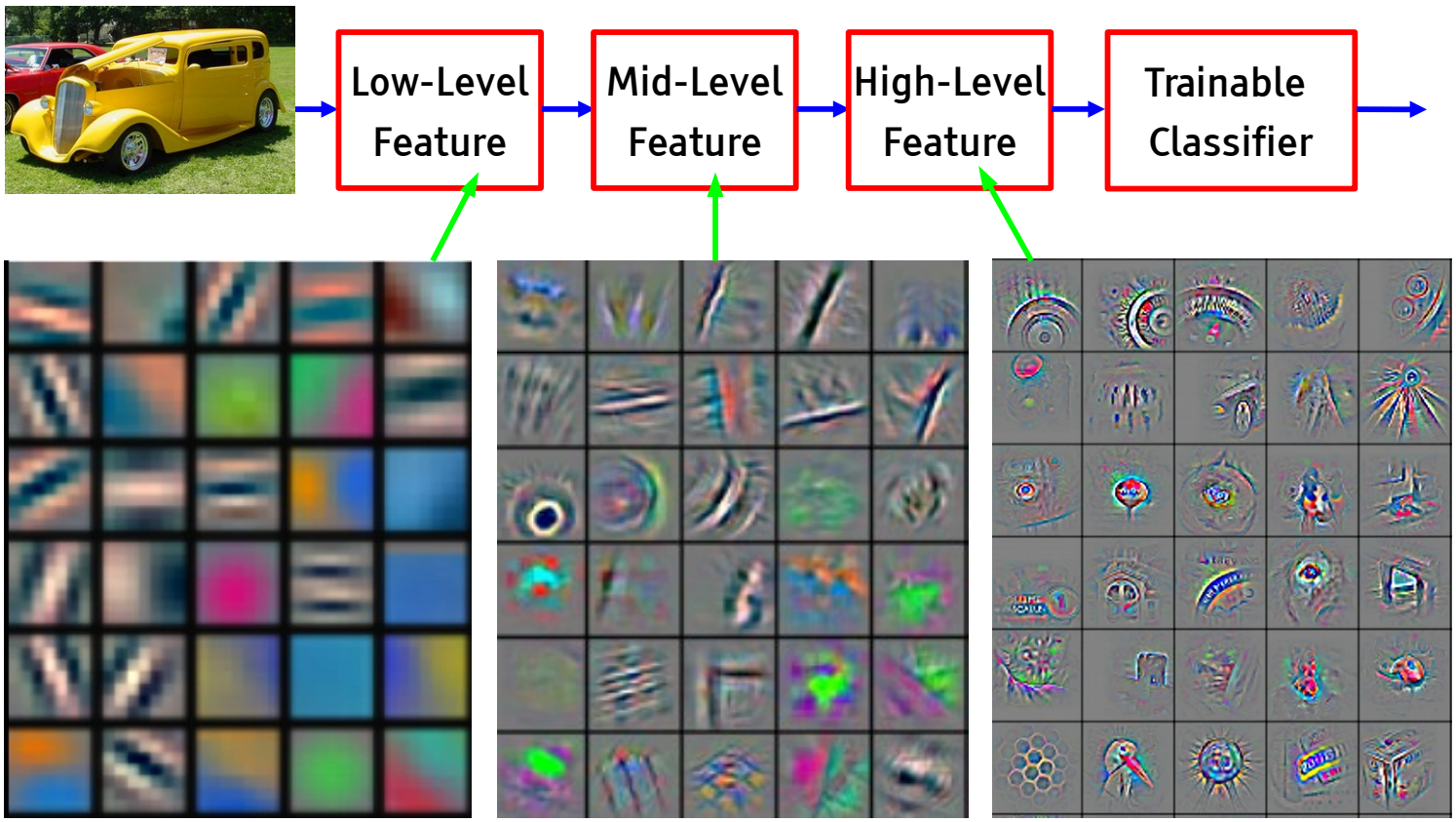
A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.

Deep Learning = Learning Hierarchical Representations

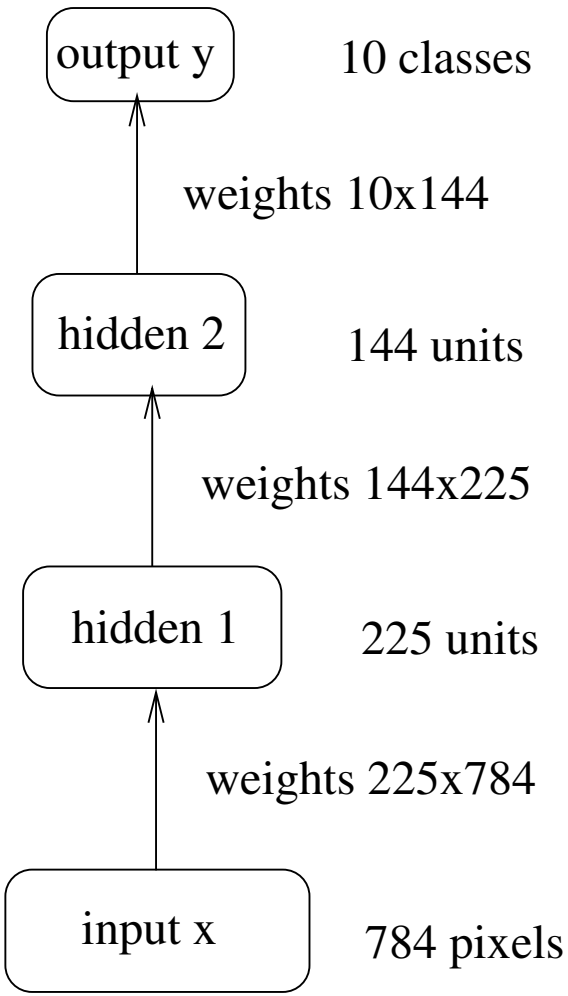
Y LeCun

- It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Example Network



$$\mathbf{h}^{(3)} = \text{softmax}(\mathbf{W}^{(3)}\mathbf{h}^{(2)} + \mathbf{b}^{(3)})$$

$$\mathbf{h}^{(2)} = \text{relu}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)})$$

$$\mathbf{h}^{(1)} = \text{relu}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\text{softmax}(\mathbf{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$\text{relu}(z) = max(0, z)$$

Training criterion

Find parameters

$$\theta = \{\mathbf{W}^{(L)}, \mathbf{b}^{(L)}\}$$

that minimize expected negative log-likelihood:

$$C = \mathbb{E}_{\text{data}} \left[-\log P(\mathbf{y}|\mathbf{x}, \theta) \right].$$

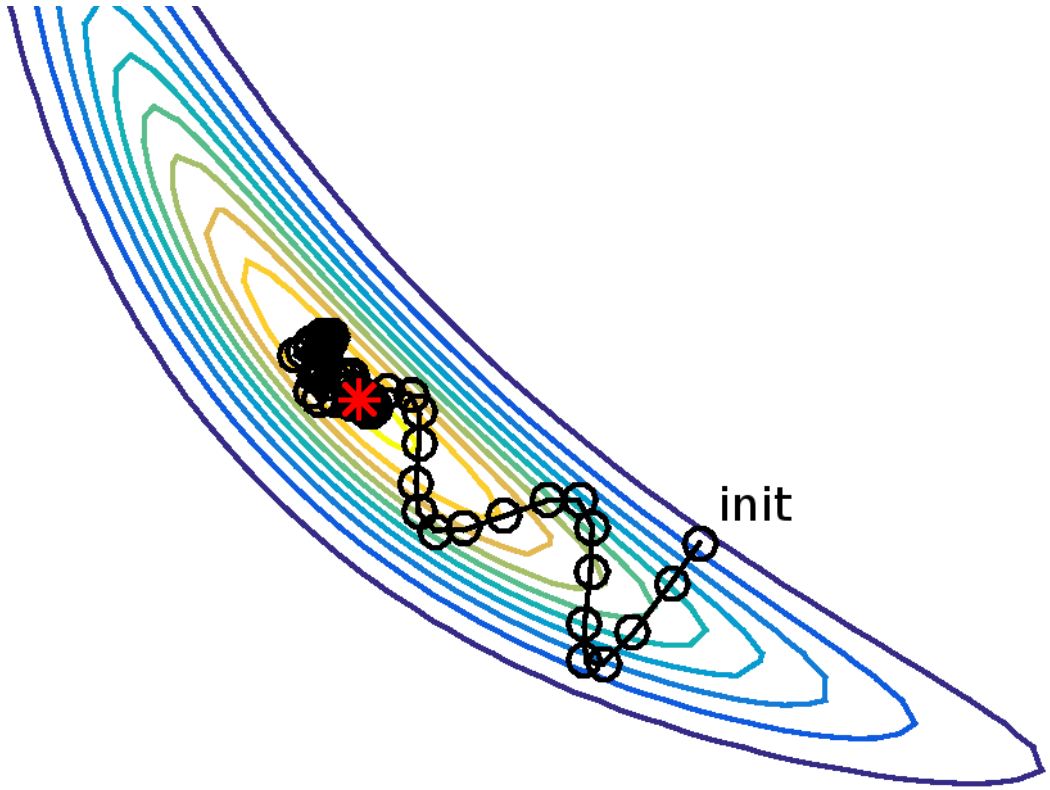
$\Rightarrow$ Learning becomes optimization.

Backpropagation

- For computing the gradient $\mathbf{g} = \nabla_{\boldsymbol{\theta}} C(\theta) = \begin{pmatrix} \frac{\partial C}{\partial \theta_1} \\ \vdots \\ \frac{\partial C}{\partial \theta_n} \end{pmatrix}$
  - **Chain rule** for derivatives
  - **Dynamic programming** avoids exponential complexity
- Modern libraries do automatic derivations

Momentum method, mini-batch training

$$\mathbf{m}_{k+1} = \alpha \mathbf{m}_k - \eta_k \mathbf{g}_k$$
$$\theta_{k+1} = \theta_k + \mathbf{m}_{k+1}$$



Use only a small batch of training data at once, noisy estimate of gradient $\mathbf{g}$ is ok.

References

- https://github.com/kjw0612/awesome-rnn
- http://arxiv.org/pdf/1507.01273.pdf
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://arxiv.org/pdf/1211.5063.pdf
- http://arxiv.org/abs/1506.02078
- http://devblogs.nvidia.com/parallelforall/introduction-neural-machine-translation-gpus-part-3/
- http://arxiv.org/pdf/1506.03340.pdf
- http://arxiv.org/abs/1502.03044
- http://arxiv.org/abs/1410.3916
- http://arxiv.org/abs/1410.5401
- http://arxiv.org/abs/1410.4615
- http://arxiv.org/pdf/1506.05869.pdf

Thanks!
Ole Winther