



LUND UNIVERSITY

FMAN-45: Machine Learning

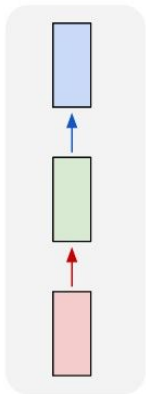
Lecture 8:

Recurrent Neural Networks

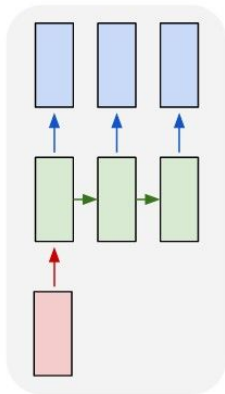
David Nilsson, Cristian Sminchisescu

RNNs

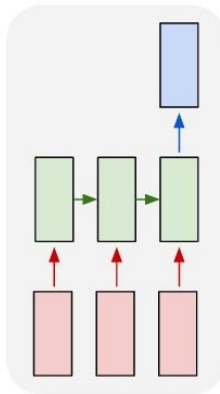
one to one



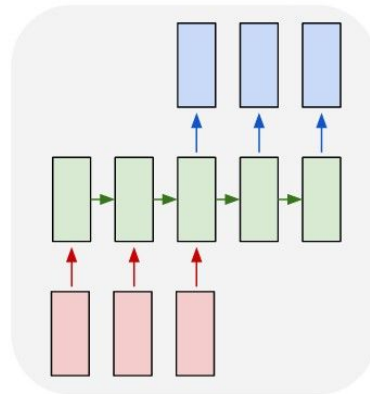
one to many



many to one



many to many



many to many

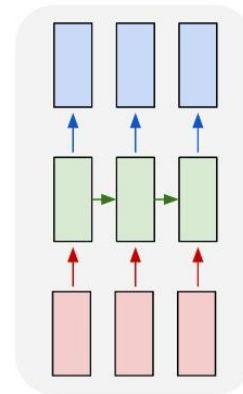
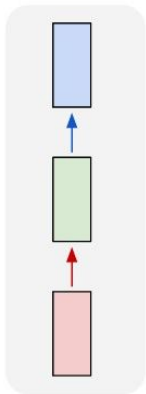


Image -> class

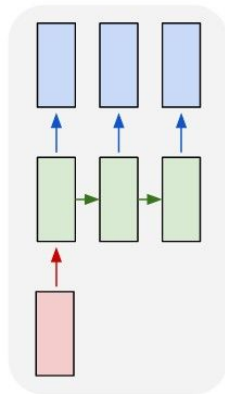
<http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

RNNs

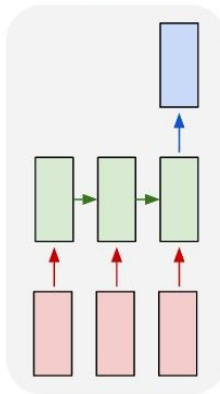
one to one



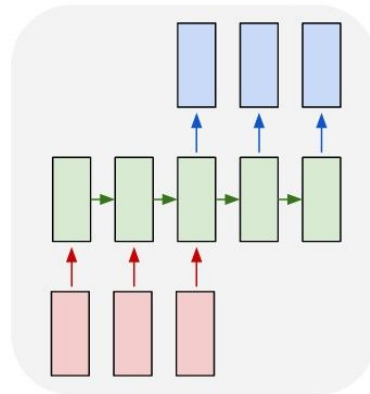
one to many



many to one



many to many



many to many

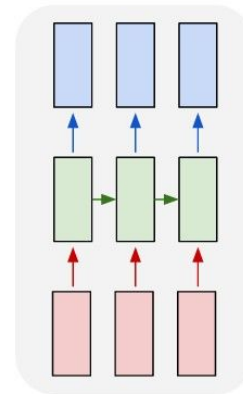
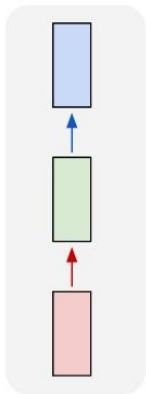


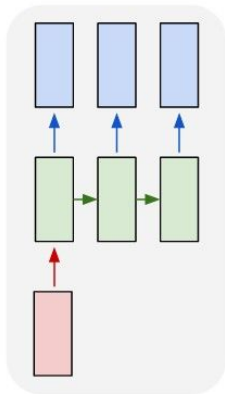
Image -> caption
Output one word at a time

RNNs

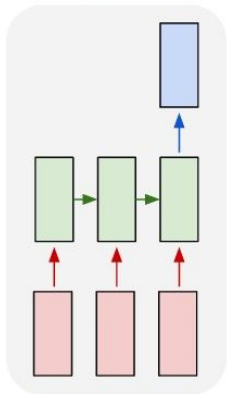
one to one



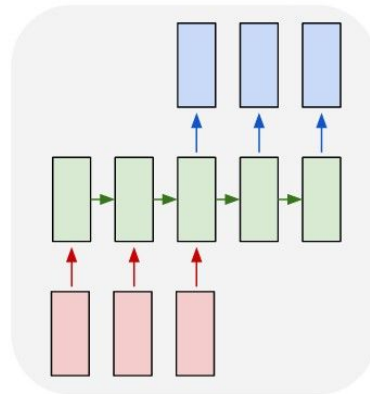
one to many



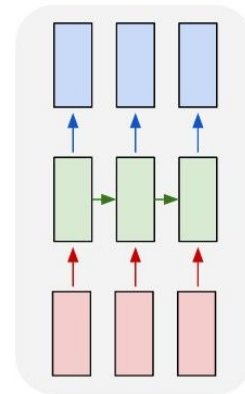
many to one



many to many



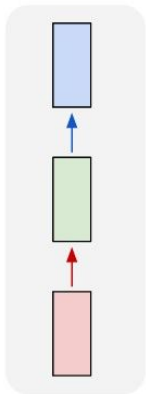
many to many



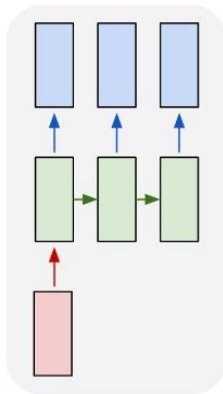
Video -> action (e.g. run, walk, jump, ...)

RNNs

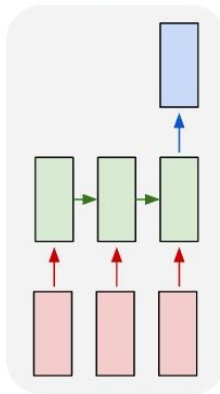
one to one



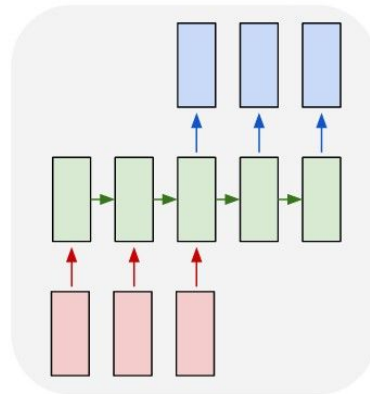
one to many



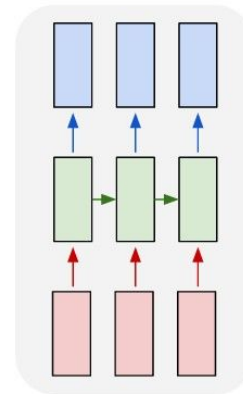
many to one



many to many



many to many

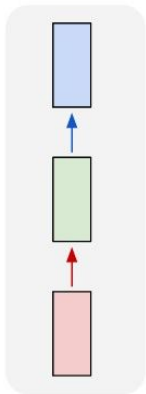


Machine translation

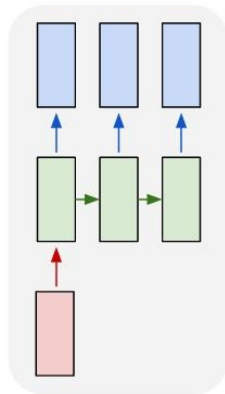
Sequence of words \rightarrow sequence of words

RNNs

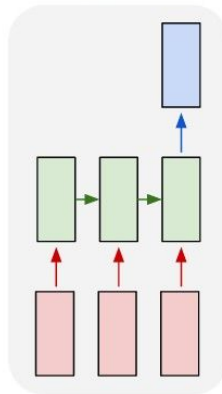
one to one



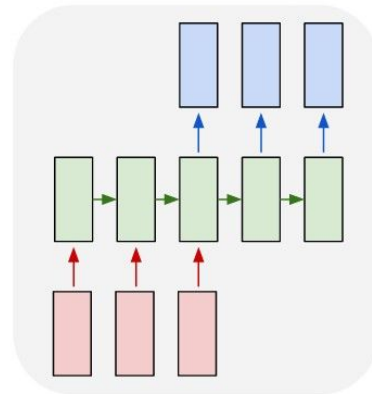
one to many



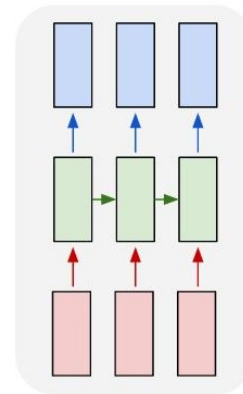
many to one



many to many



many to many



Video segmentation

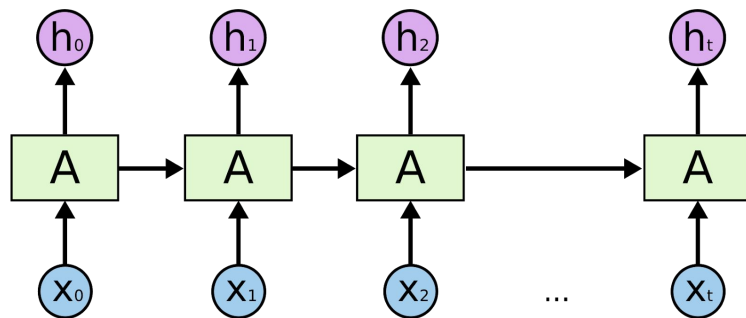
RNN

Process sequential data.

The RNN stores a state vector with relevant features.

Every time an input is processed, update the state vector.

The weights of the recurrent units are shared through time. There is only one set of parameters to learn.

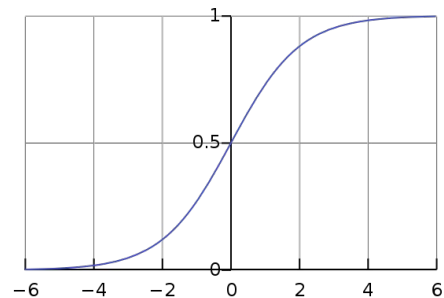


<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

RNNs - Gating

We want the hidden states to turn on and off during processing.

Multiply with 0/1, computed adaptively using the data, is the obvious modelling of how to do it. The problem is that all neural networks layers must be differentiable. We use the sigmoid function followed by element-wise multiplication to model gateings.



Sigmoid function

RNNs - Gating

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridge broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

RNNs - Gating

Cell that turns on inside comments and quotes:

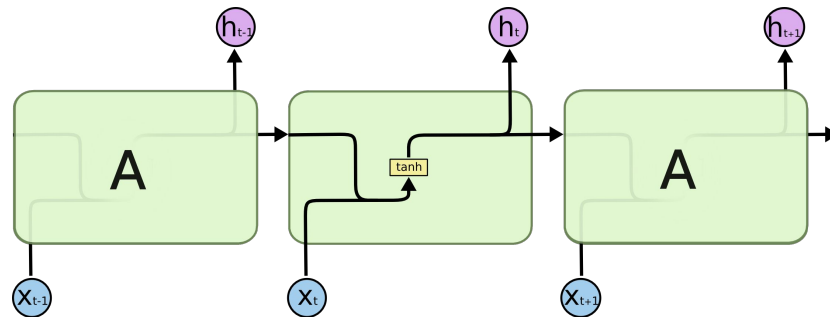
```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                     struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, 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;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

RNNs

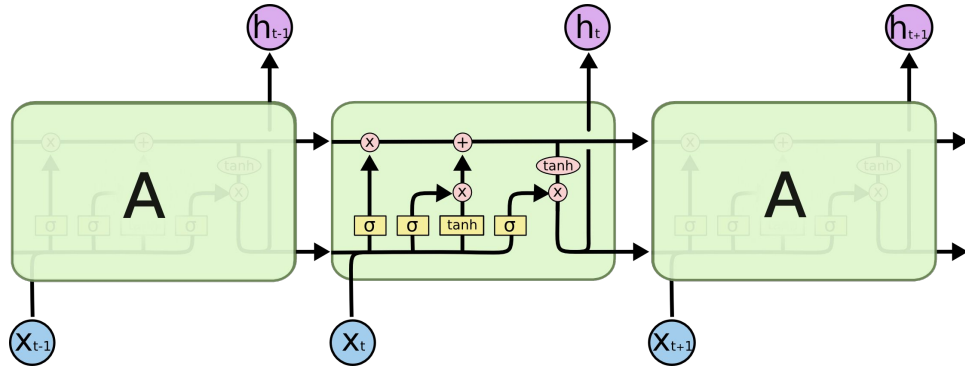
For every timestep the hidden states pass through a tanh non-linearity. This makes the optimization problem very hard and it will not learn long term dependencies.



$$h_t = \tanh(W h_{t-1} + U x_t + b)$$

$$y_t = \text{softmax}(V h_t + c)$$

LSTM - Long Short-Term Memory

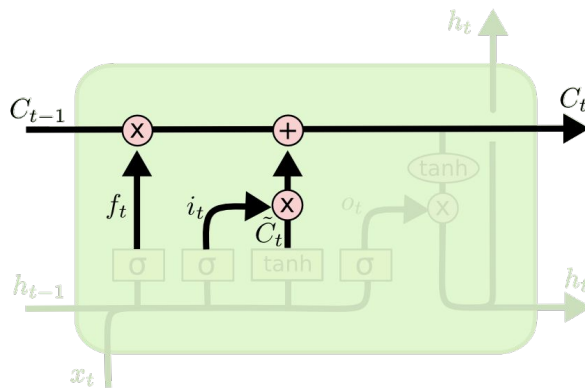


LSTM

Memory state.

The reason why the gradients are stable and why an LSTM can learn long-term dependencies.

We forget what should be forgotten and add what should be added.

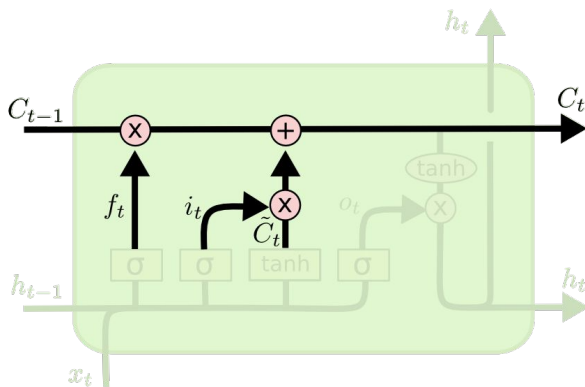


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM

If the state is unchanged, the mapping is an identity mapping and the same errors are back-propagated.

The formula holds for one hidden state. It holds element-wise for many hidden states.



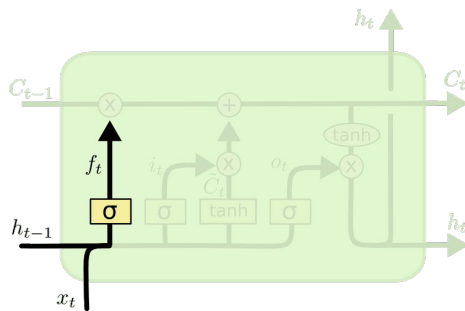
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$\frac{\partial L}{\partial C_{t-1}} = \frac{\partial L}{\partial C_t} f_t$$
$$\left(f_t = 1 \implies \frac{\partial L}{\partial C_{t-1}} = \frac{\partial L}{\partial C_t} \right)$$

LSTM

Forget gate.

Example: We have a feature that increases the longer it has been since a line break. When a line break is obtained in x_t , then the feature is reset to 0.



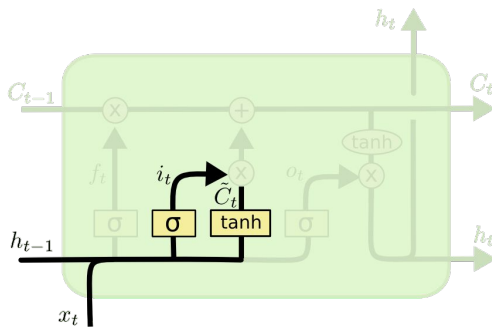
$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM

Input gate and candidate memory state.

Given new information in x_t , the memory state C_t should be updated.

Example: If x_t is a dot, then the memory state should contain information that we are about to start a new sentence.



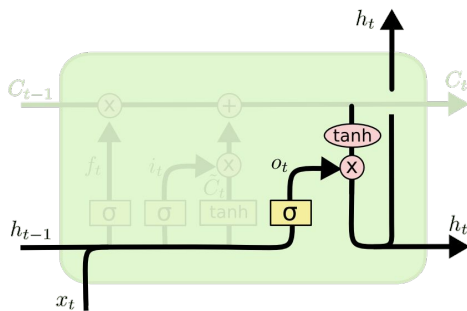
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

LSTM

Output gate.

The memory state C_t and the hidden state h_t have different time scales. The hidden states are relevant for short term processing and the memory gate stores relevant long-term information.

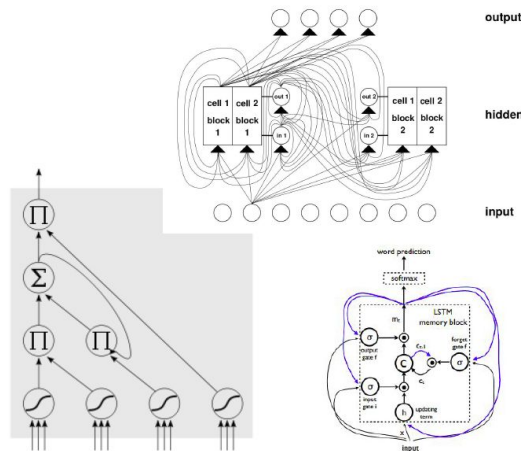
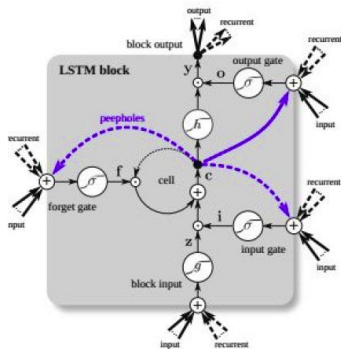
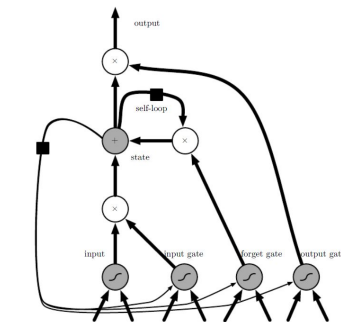
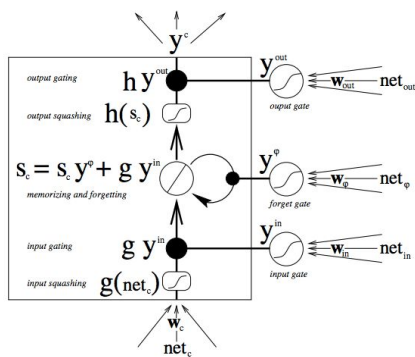


$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$

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

LSTM

Be aware of crazy LSTM diagrams. See the [webpage](#) where the figures in the previous slides are taken from for a very good description.



RNNs

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

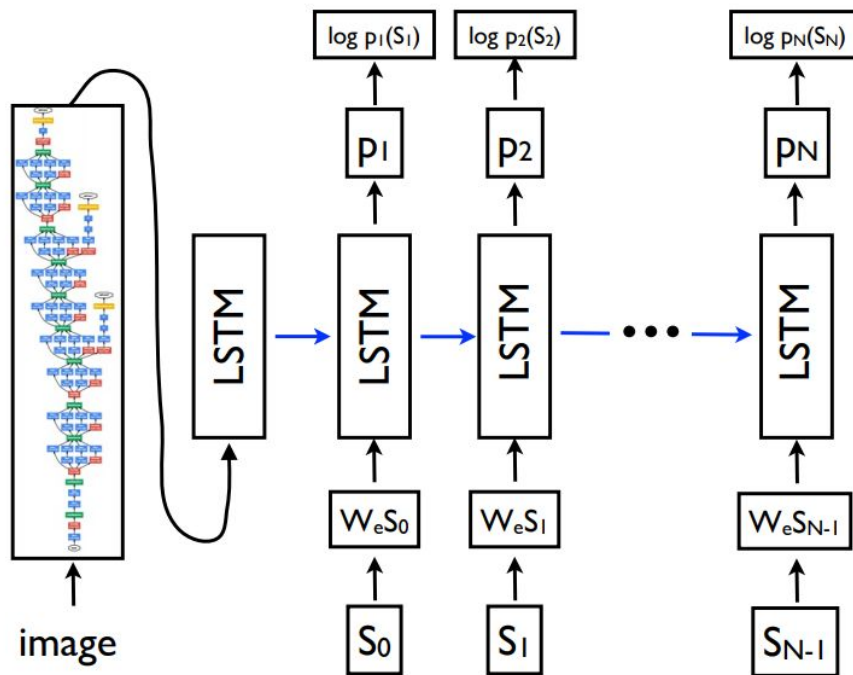
Somewhat related to the image

Unrelated to the image

RNNs

Process the image through a deep CNN to get a 512-dimensional feature vector, used to initialize the state vector.

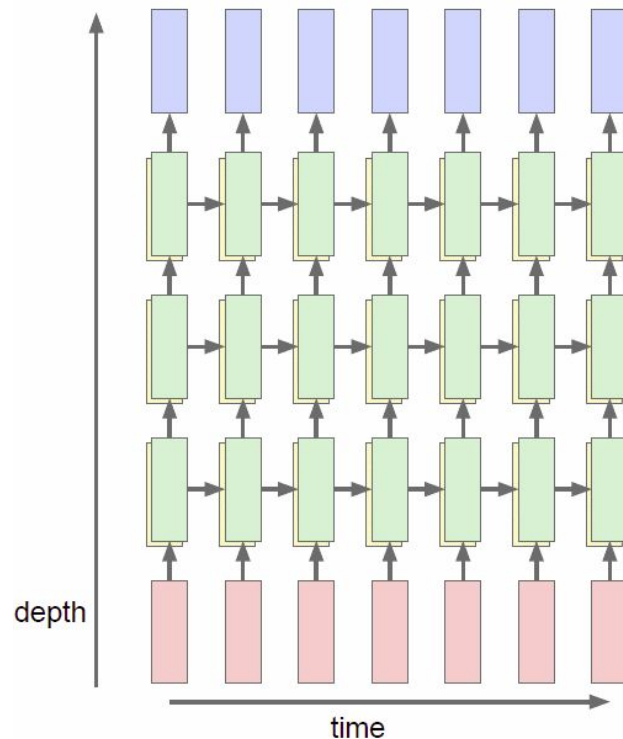
Send in one word embedding at a time, process it through the LSTM, compute/sample the next word.



Stack LSTMs

We can make RNNs that are arbitrarily deep.

Notice similar dependencies on inputs and hidden states from the same layer.



Stack LSTMs

Google's translation system.

Process the word embedding in a sequential manner to get an encoding of the sentence to translate.

Send the encoding to a decoder that outputs the translation.

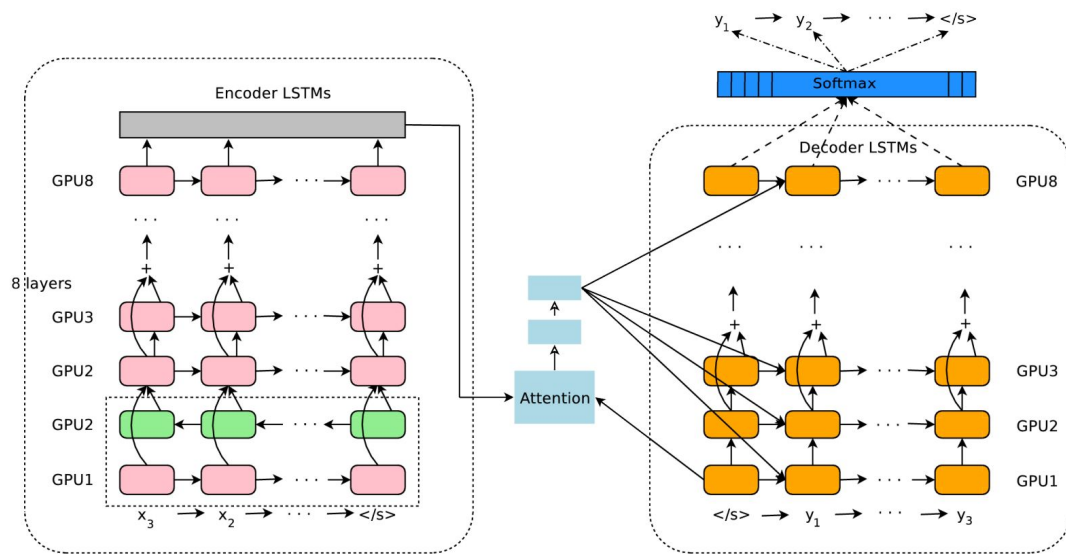
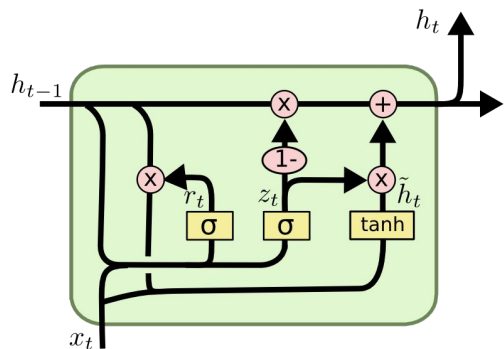


Figure 1: The model architecture of GNMT, Google's Neural Machine Translation system. On the left

<https://arxiv.org/pdf/1609.08144.pdf>

"it takes around 6 days to train a basic model using 96 NVIDIA K80 GPUs"

GRU - Gated Recurrent Unit



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

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

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

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

Assignment 3

Derive and implement forward and backward passes for fully connected layers, relu-layers and softmax-losses. Be very careful with indices!

For problem 1-4, start with the provided formulas. Do not use a Jacobian approach where it is not applicable, or preferably not at all. The Jacobian is defined for a mapping from a vector to a vector and not for mappings from matrices to matrices. $dA/dB=?$

Implement gradient descent with momentum.

Train and design neural networks for image classification.

Readings

Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning, MIT Press 2016
(online at <http://www.deeplearningbook.org/>)

CNNs

- Section 9.1-9.3 in the deep learning book

LSTMs

- <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Section 10.10 in the deep learning book