



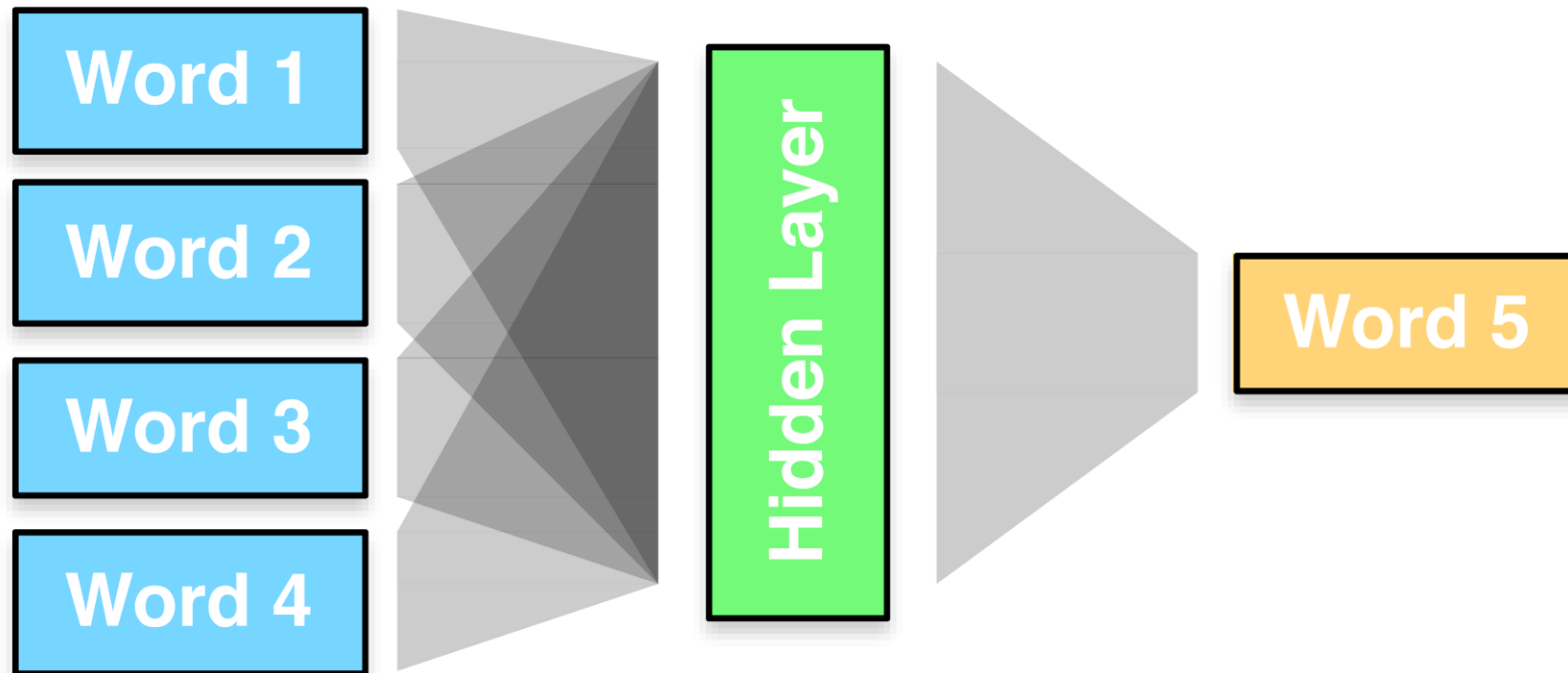
COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

Recurrent Language Models

CMSC 470

Marine Carpuat

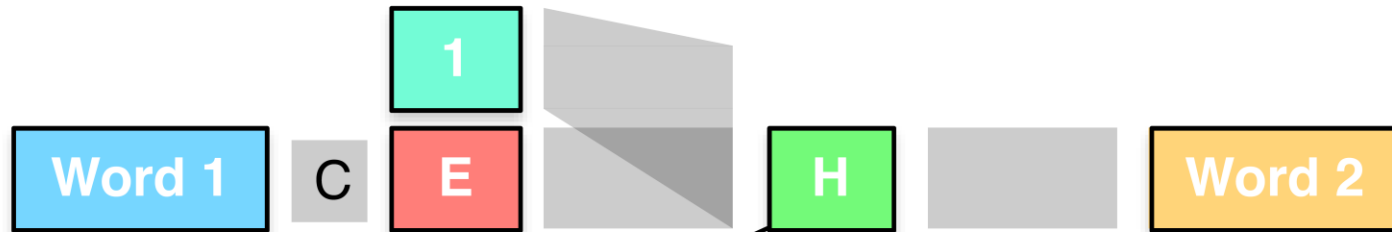
Toward a Neural Language Model



Count-based n-gram models vs. feedforward neural networks

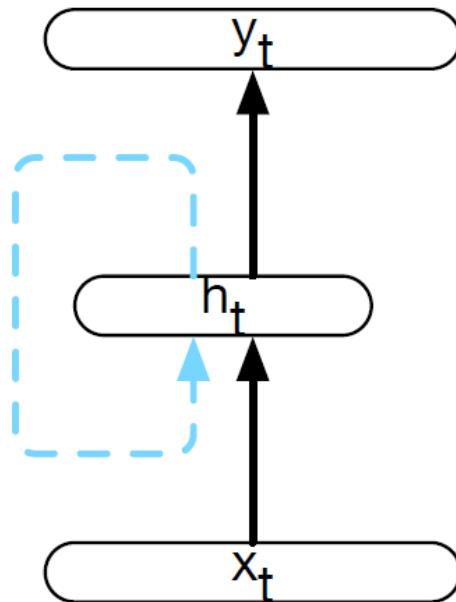
- Pros of feedforward neural LM
 - Word embeddings capture generalizations across word types
- Cons of feedforward neural LM
 - Closed vocabulary
 - Training/testing is more computationally expensive
- Weaknesses of both types of model
 - Only work well for word prediction if the test corpus looks like the training corpus
 - Only capture short distance context

Language Modeling with Recurrent Neural Networks



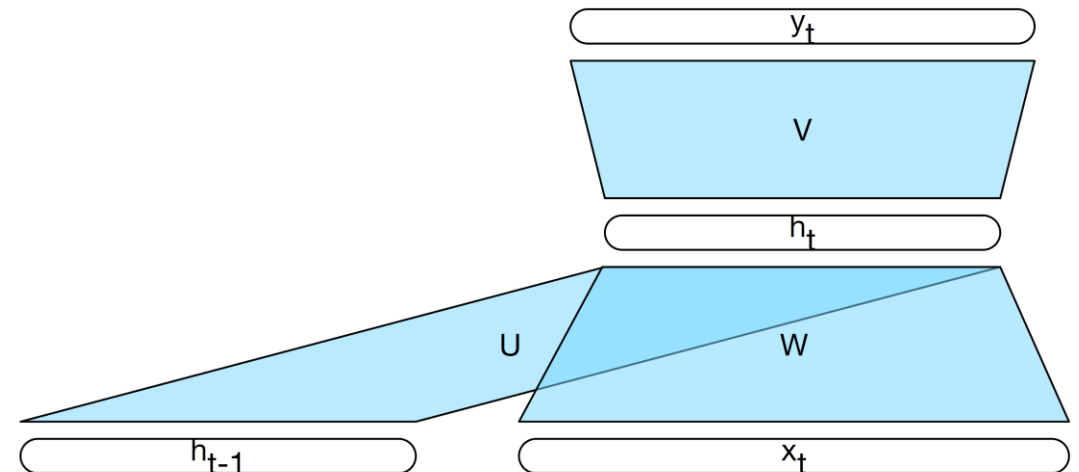
Recurrent Neural Networks (RNN)

The hidden layer includes a recurrent connection as part of its input

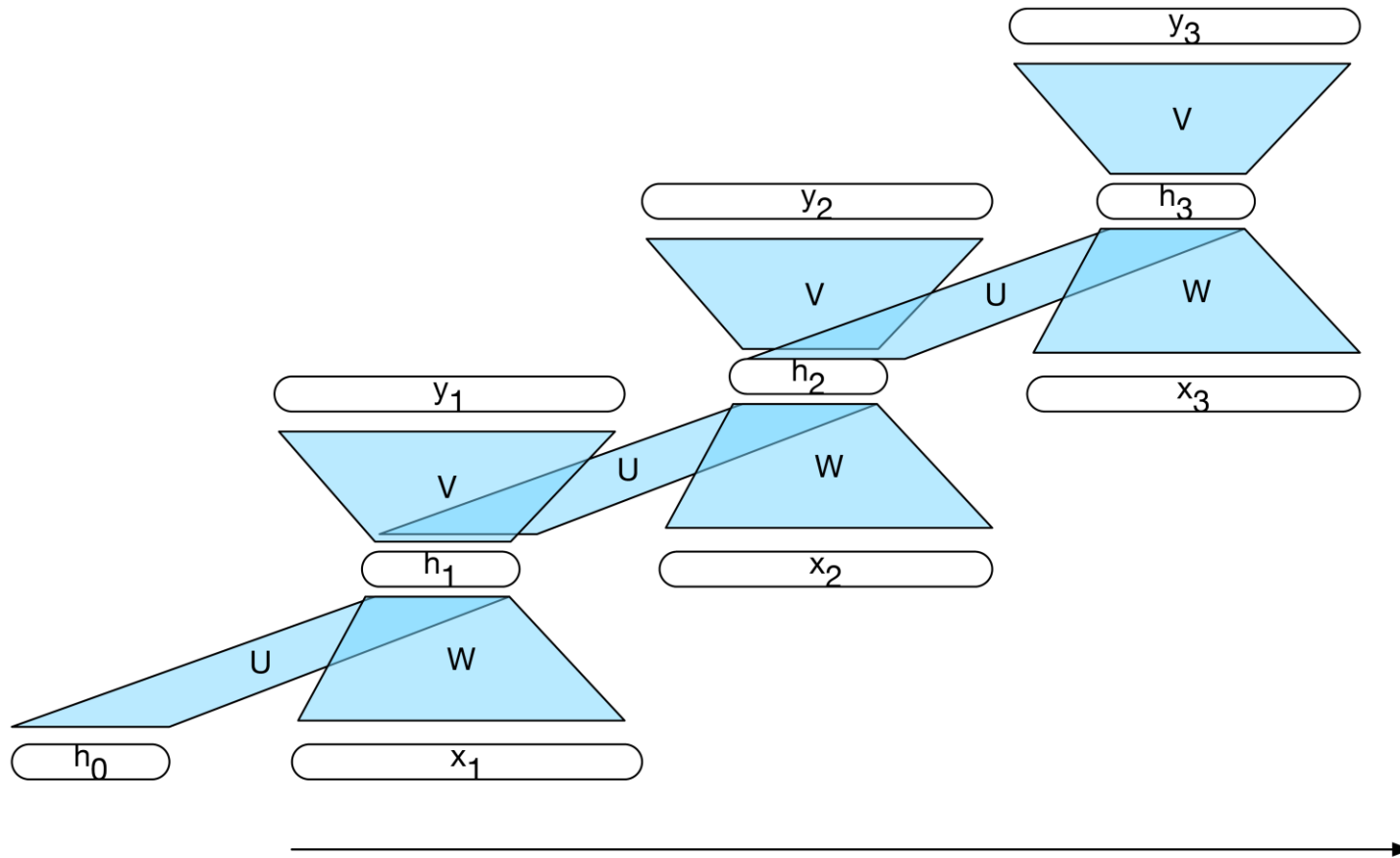


The hidden layer from the previous time step plays the role of memory, remembering earlier context

Unrolling the RNN over the time sequence as a feed-forward network



Unrolled RNN illustrated



weights U , V , W
are shared across
all timesteps

Prediction/Inference with RNNs

function FORWARDRNN(x , *network*) **returns** output sequence y

$h_0 \leftarrow 0$

for $i \leftarrow 1$ **to** LENGTH(x) **do**

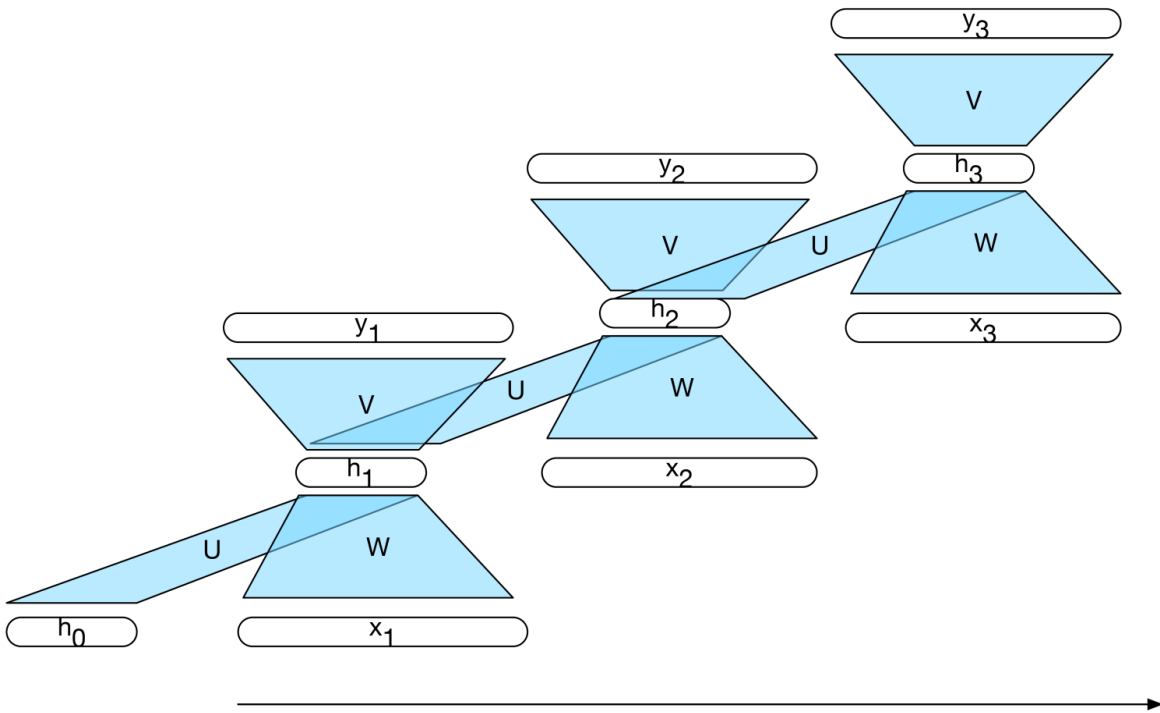
$h_i \leftarrow g(U h_{i-1} + W x_i)$

$y_i \leftarrow f(V h_i)$

return y

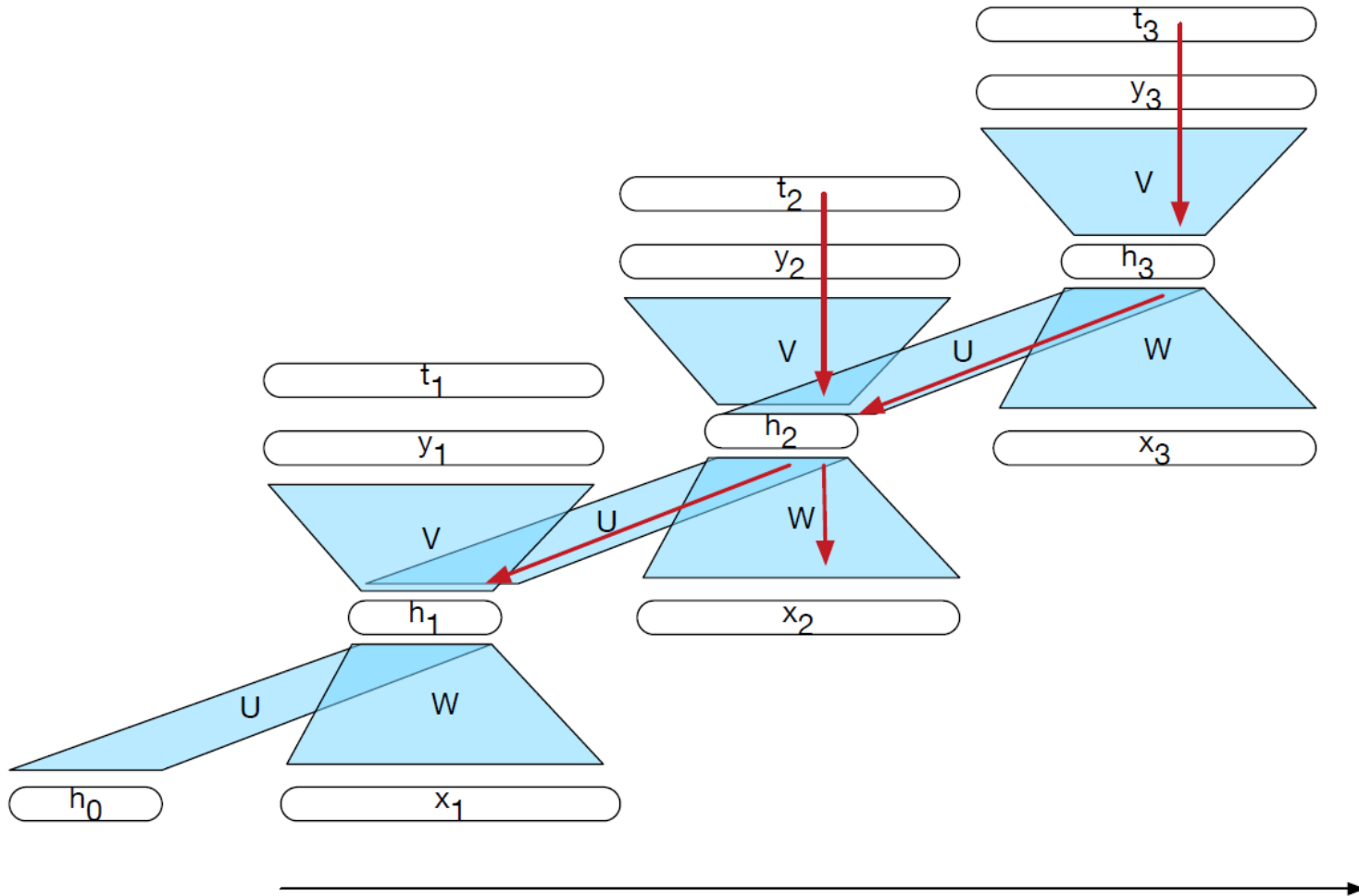
For language modeling, f = softmax function to provide normalized probability distribution over possible output classes

Training RNNs with backpropagation



- Training goal: estimate parameter values for U , V , W
- Use same loss as for feedforward language models
- Given unrolled network, run forward and backpropagation algorithms as usual

Training RNNs with backpropagation



Practical Training Issues: vanishing/exploding gradients

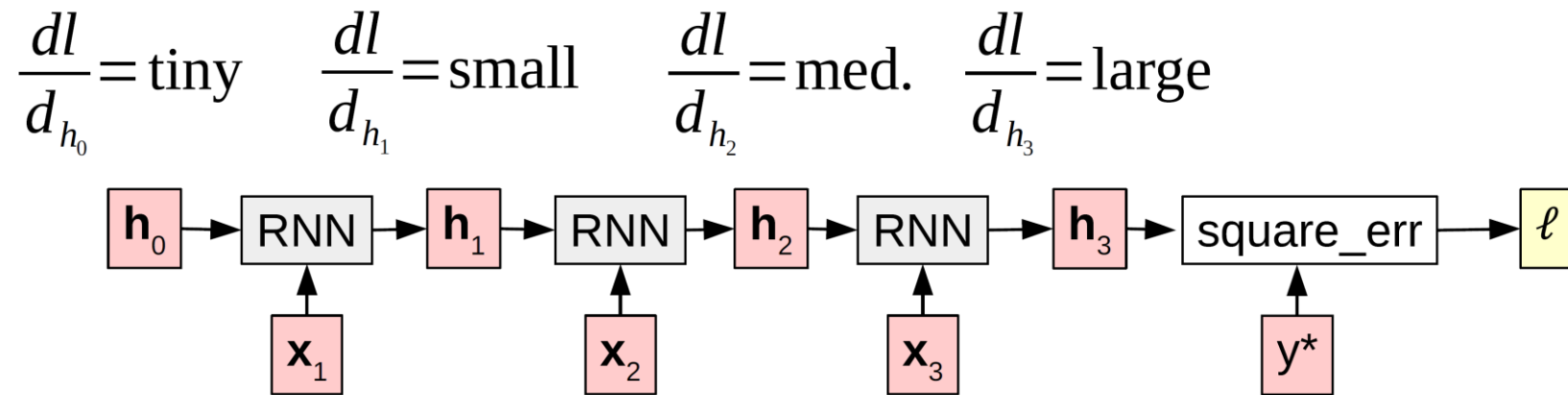
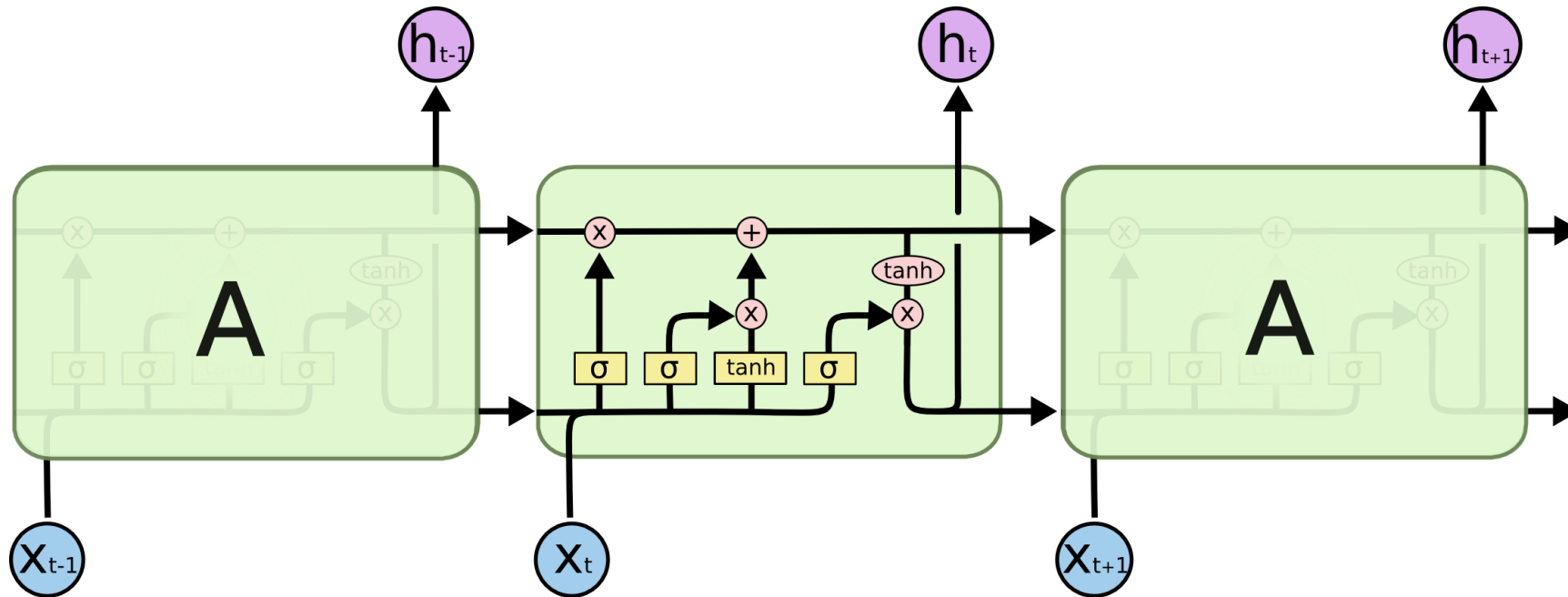


Figure 16: An example of the vanishing gradient problem.

Multiple ways to work around this problem:

- ReLU activations help
- Dedicated RNN architecture (Long Short Term Memory Networks)

Aside: Long Short Term Memory Networks



What do Recurrent Language Models Learn?

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 bridges 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."

What do Recurrent Language Models Learn?

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 robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
                           siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

What do Recurrent Language Models Learn?

- Parameters are hard to interpret, so we can gain insights by analyzing their output behavior instead
- Can capture (some) long-distance dependencies

After much economic progress over the years, the country **has**...

The country, which has made much economic progress over the years, still **has**...

Recurrent neural network language models

- Have all the strengths of feedforward language model
- And do a better job at modeling long distance context
- However
 - Training is trickier due to vanishing/exploding gradients
 - Performance on test sets is still sensitive to distance from training data