CS769 Advanced NLP

Recurrent Neural Networks

Junjie Hu

UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Graham, Sergey, Chris

https://junjiehu.github.io/cs769-spring23/

Goals for Today

- Review: Sequential Data
- **Recurrent Neural Network**
- **Vanishing/Exploding Gradients & LSTM**
- Applications and Training Tricks of RNN
- Variants of RNN

NLP and Sequential Data

- NLP is full of sequential data
    - Words in sentences
    - Characters in words
    - Sentences in discourse
    - …

Long-distance Dependencies in Language

- Agreement in number, gender, etc.

He does not have very much confidence in **himself**.
She does not have very much confidence in **herself**.

- Selectional preference

The **reign** has lasted as long as the life of the **queen**.
The **rain** has lasted as long as the life of the **clouds**.

Can be Complicated!

- What is the referent of “it”?

The trophy would not fit in the brown suitcase because it was too **big**.

Trophy

The trophy would not fit in the brown suitcase because it was too **small**.

Suitcase

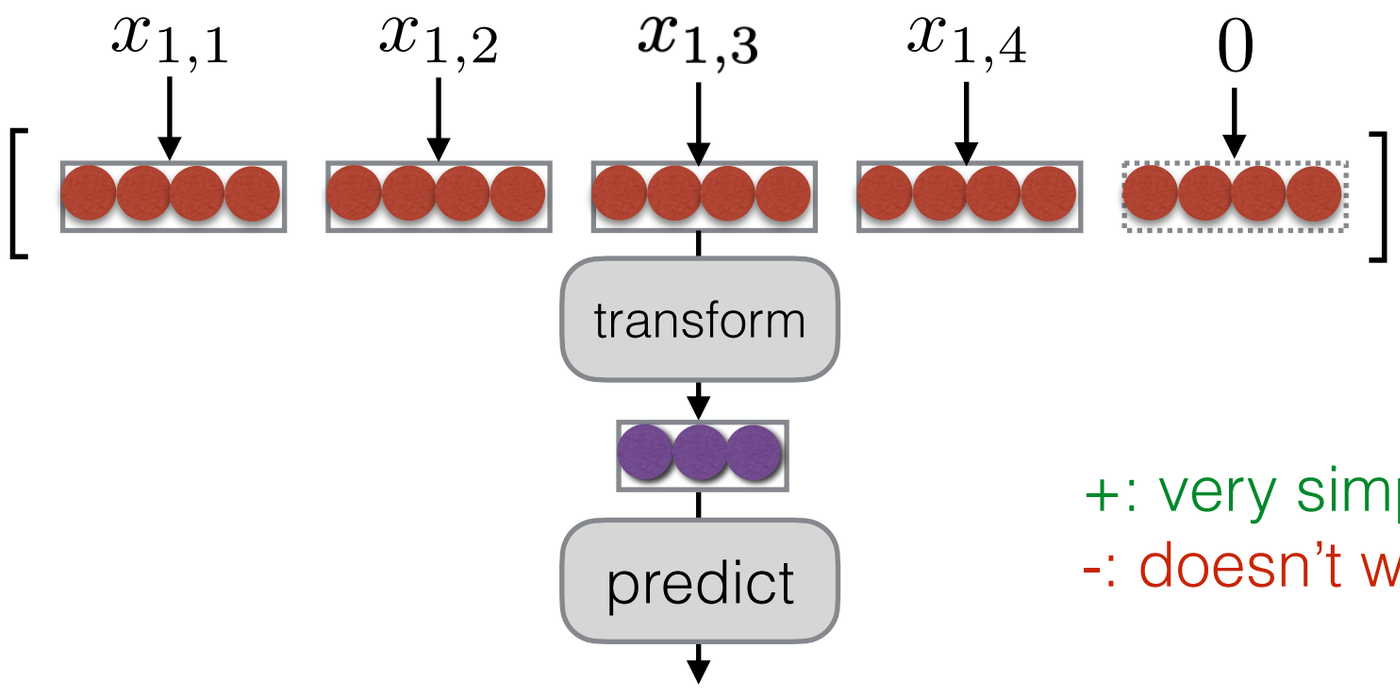
(from Winograd Schema Challenge: http://commonsensereasoning.org/winograd.html)

Recurrent Neural Networks

What if we have variable-size inputs?

$$x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4})$$
$$x_2 = (x_{2,1}, x_{2,2}, x_{2,3})$$
$$x_3 = (x_{3,1}, x_{3,2}, x_{3,3}, x_{3,4}, x_{3,5})$$

- Simple idea: zero-pad up to length of longest sequence
  $x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4}, 0)$



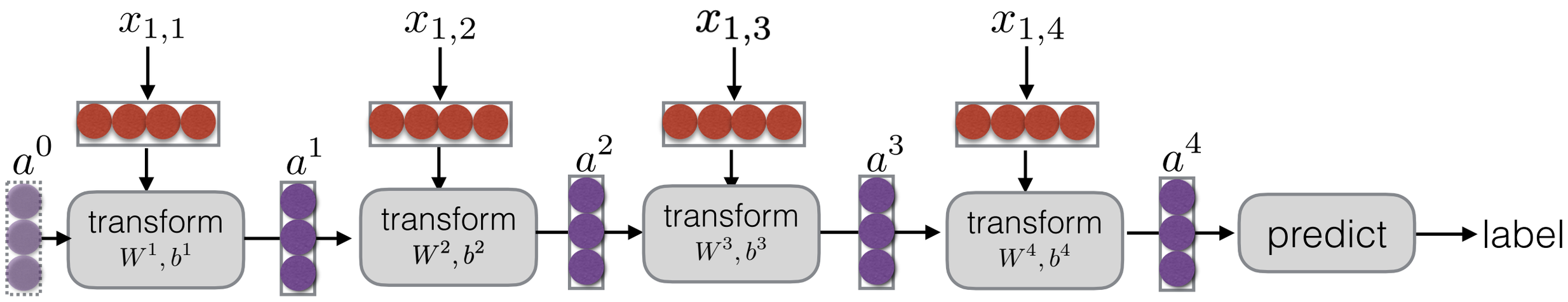
+: very simple and work in a pinch
-: doesn't work for very long sentences

Feedforward Networks

One input per layer?

$$x_1 = (x_{1,1}, x_{1,2}, x_{1,3}, x_{1,4})$$
$$x_2 = (x_{2,1}, x_{2,2}, x_{2,3})$$
$$x_3 = (x_{3,1}, x_{3,2}, x_{3,3}, x_{3,4}, x_{3,5})$$

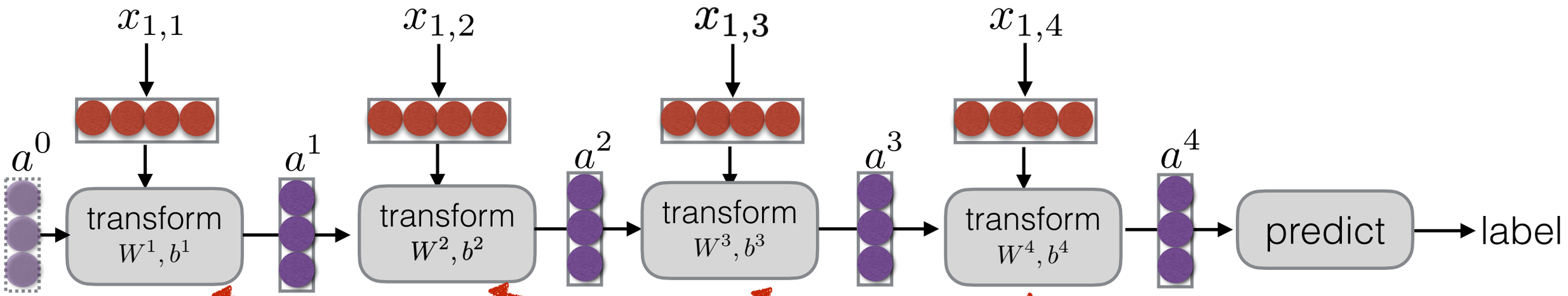
- Another idea: feed one word to each feedforward layer



$$\bar{a}^{\ell-1} = \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix} \qquad z^\ell = W^\ell \bar{a}^{\ell-1} + b^\ell \qquad a^\ell = \sigma(z^\ell)$$

+: do not need to pad sentences
+: each FF layer is much smaller than the giant FF layer for concatenated inputs
+: shorter sentence requires fewer layers
-: Obvious question: what happens to the long sentences?

Can we share weight matrics?



$$\bar{a}^{\ell-1} = \begin{bmatrix} a^{\ell-1} \\ x_{i,t} \end{bmatrix} \qquad z^\ell = W^\ell \bar{a}^{\ell-1} + b^\ell \qquad a^\ell = \sigma(z^\ell) \qquad W = W^\ell, b = b^\ell, \ \forall \ell$$

+: we can have as many “layers” as we want!

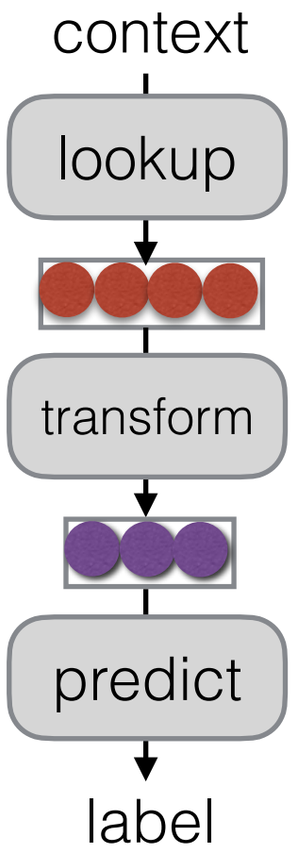
This is called a **recurrent** neural network (RNN) — “variable-depth” network.

Recurrent Neural Networks

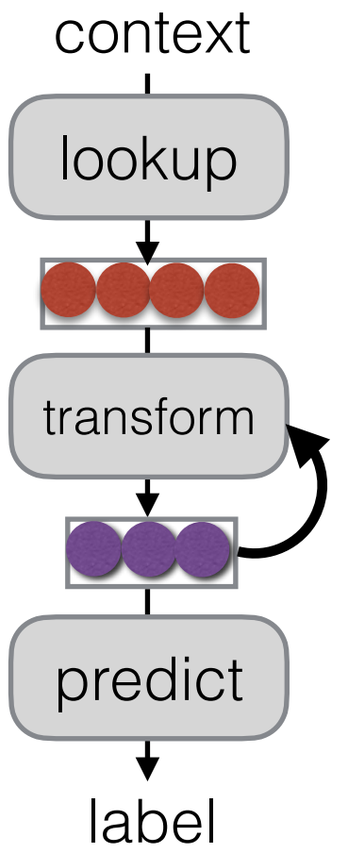
(Elman 1990)

- Tools to “remember” information along the “time dimension”
- RNNs are just NNs that share weights across layers, take an input each layer, and have a variable number of layers.

Feed-forward NN



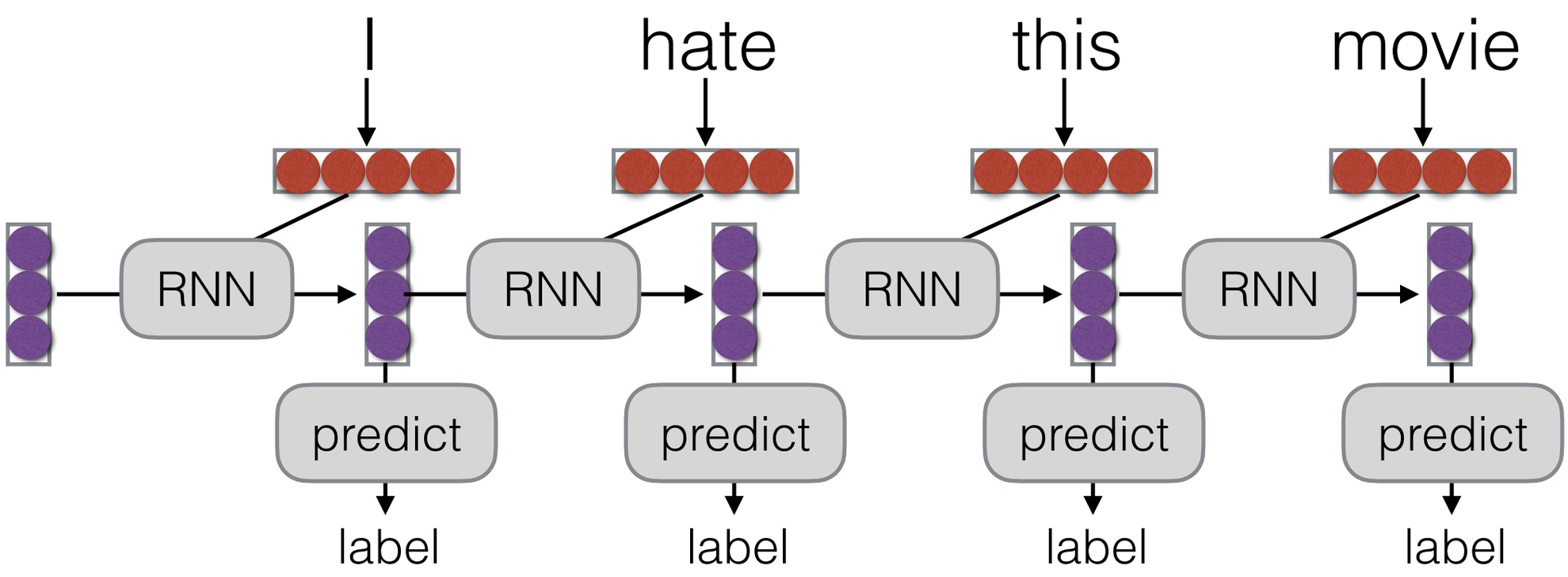
Recurrent NN



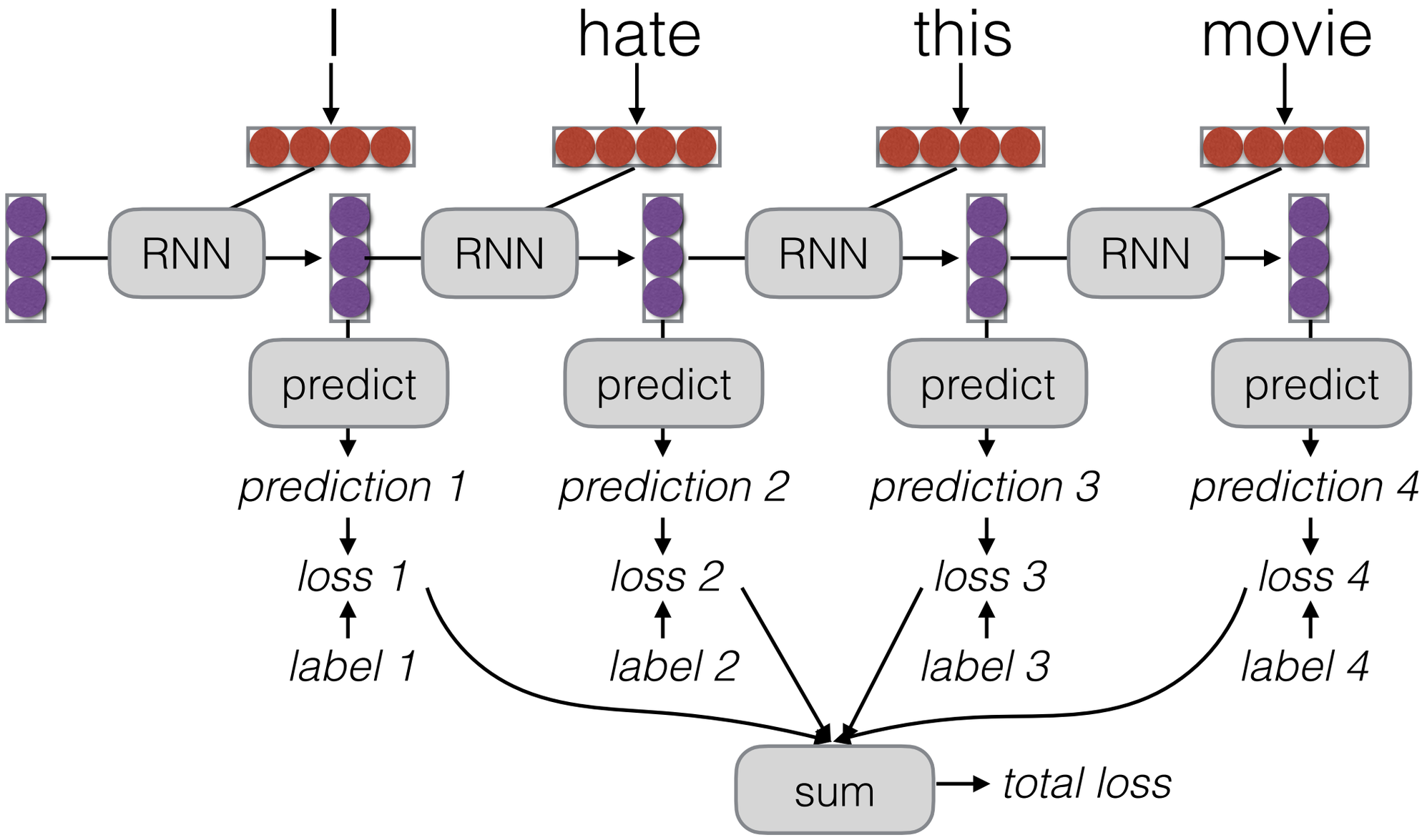
The layer gets its own “previous” value as inputs

Unrolling in Time

- What does processing a sequence look like?

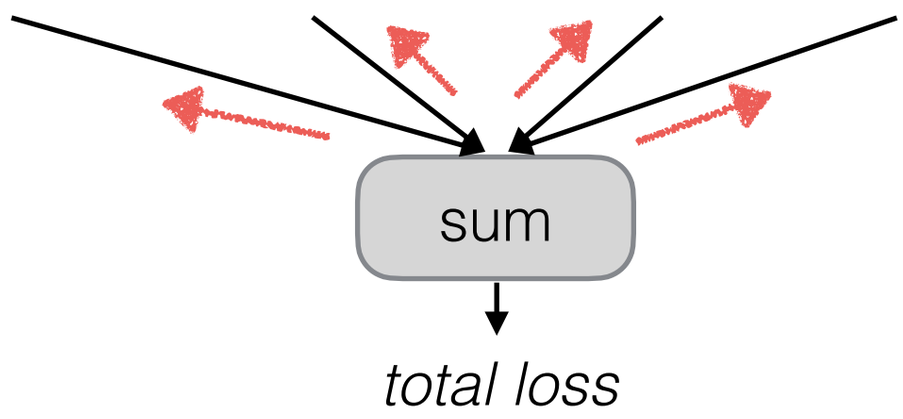


Training RNNs



RNN Training

- The unrolled graph is a well-formed directed acyclic graph (DAG)—we can run backprop

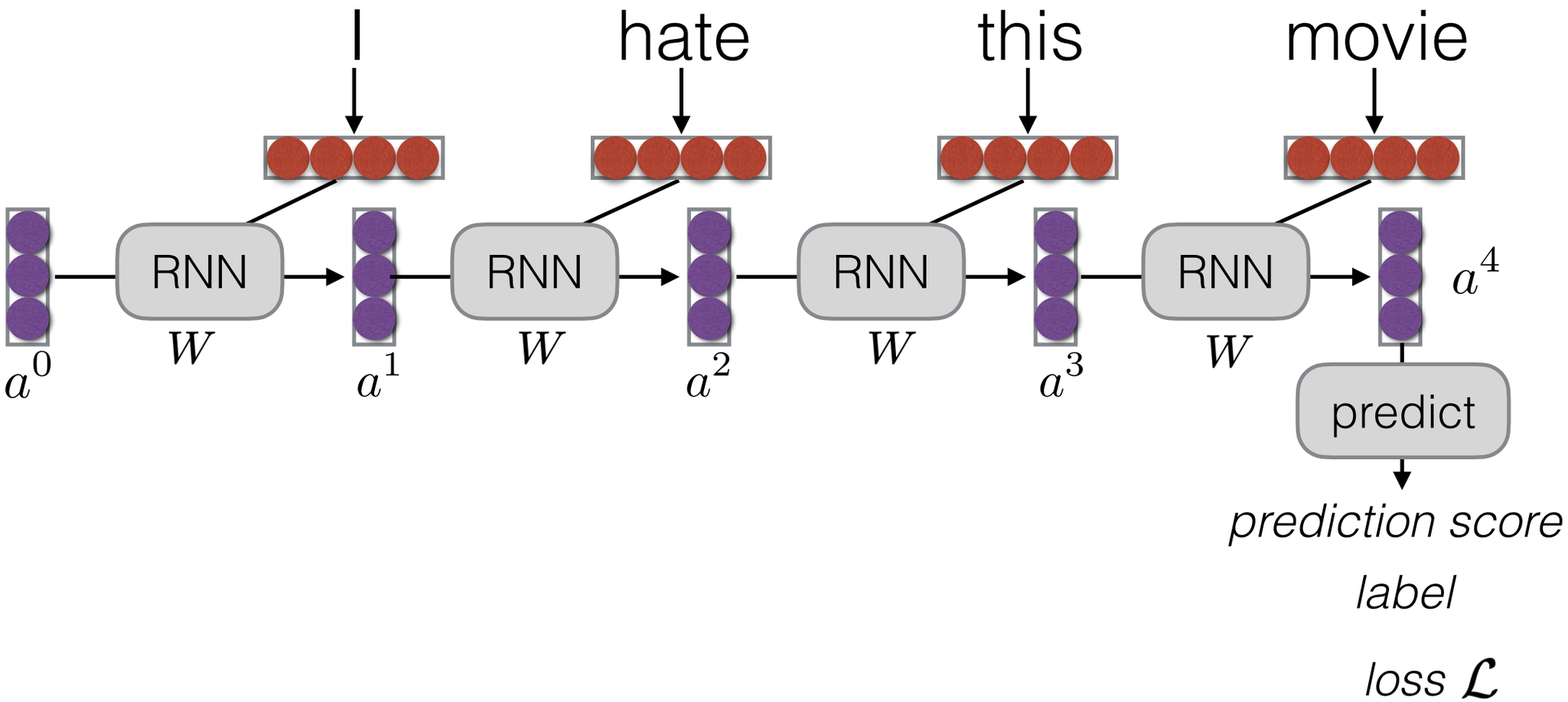


- Parameters are tied across time, derivatives are aggregated across all time steps

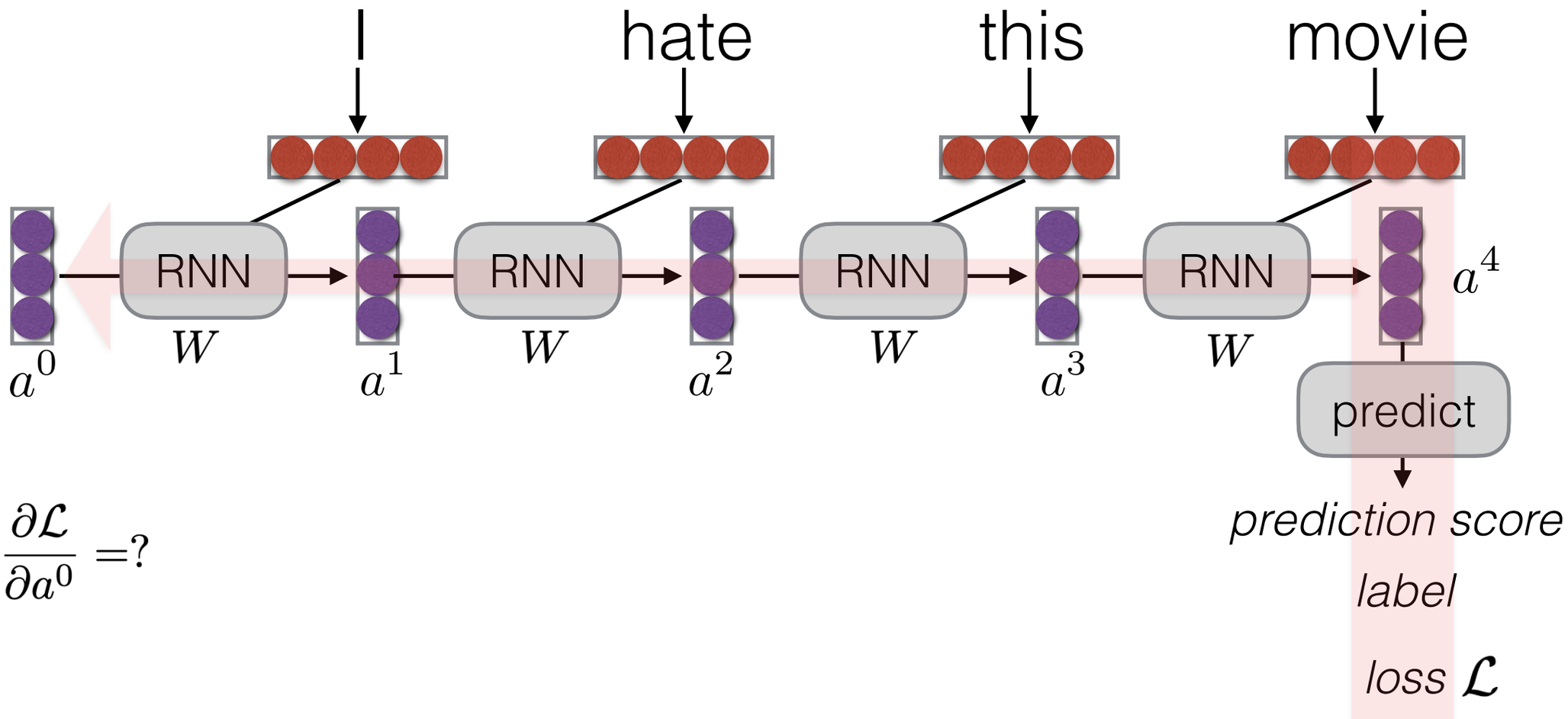
- This is historically called “backpropagation through time” (BPTT)

Forward pass

Parameters *W* are shared!

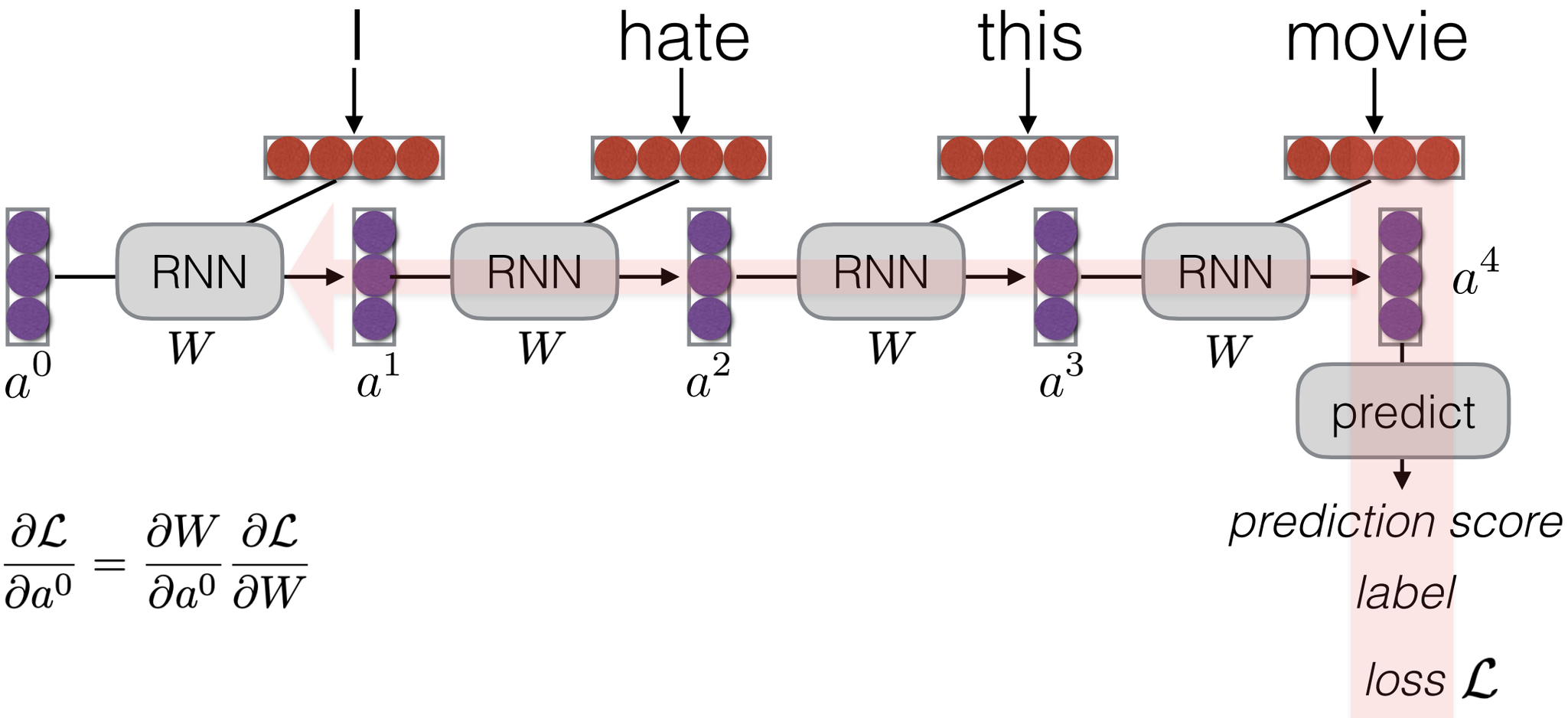


Backward pass



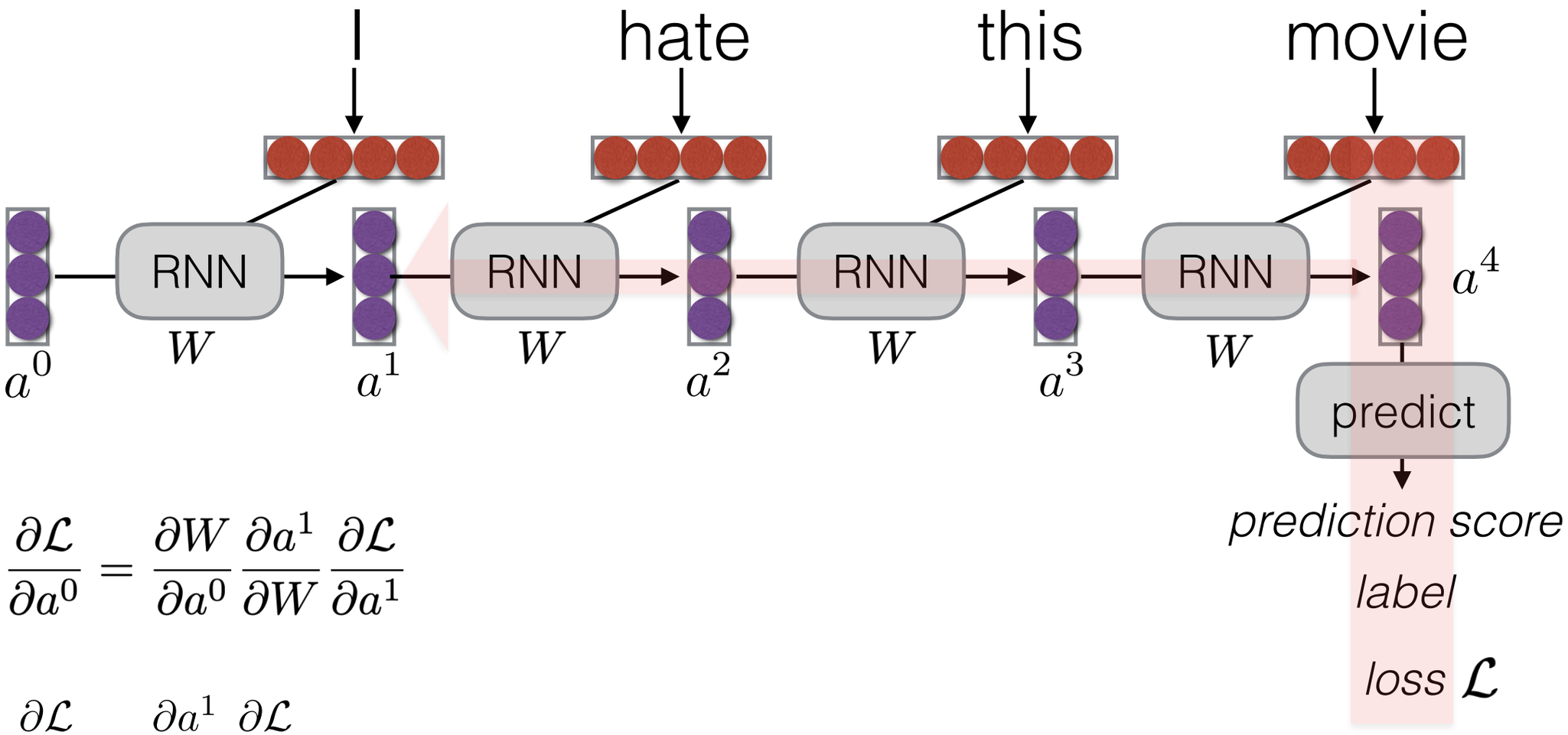
$$\frac{\partial \mathcal{L}}{\partial a^0} = ?$$

Backward pass



$$\frac{\partial \mathcal{L}}{\partial a^0} = \frac{\partial W}{\partial a^0} \frac{\partial \mathcal{L}}{\partial W}$$

Backward pass

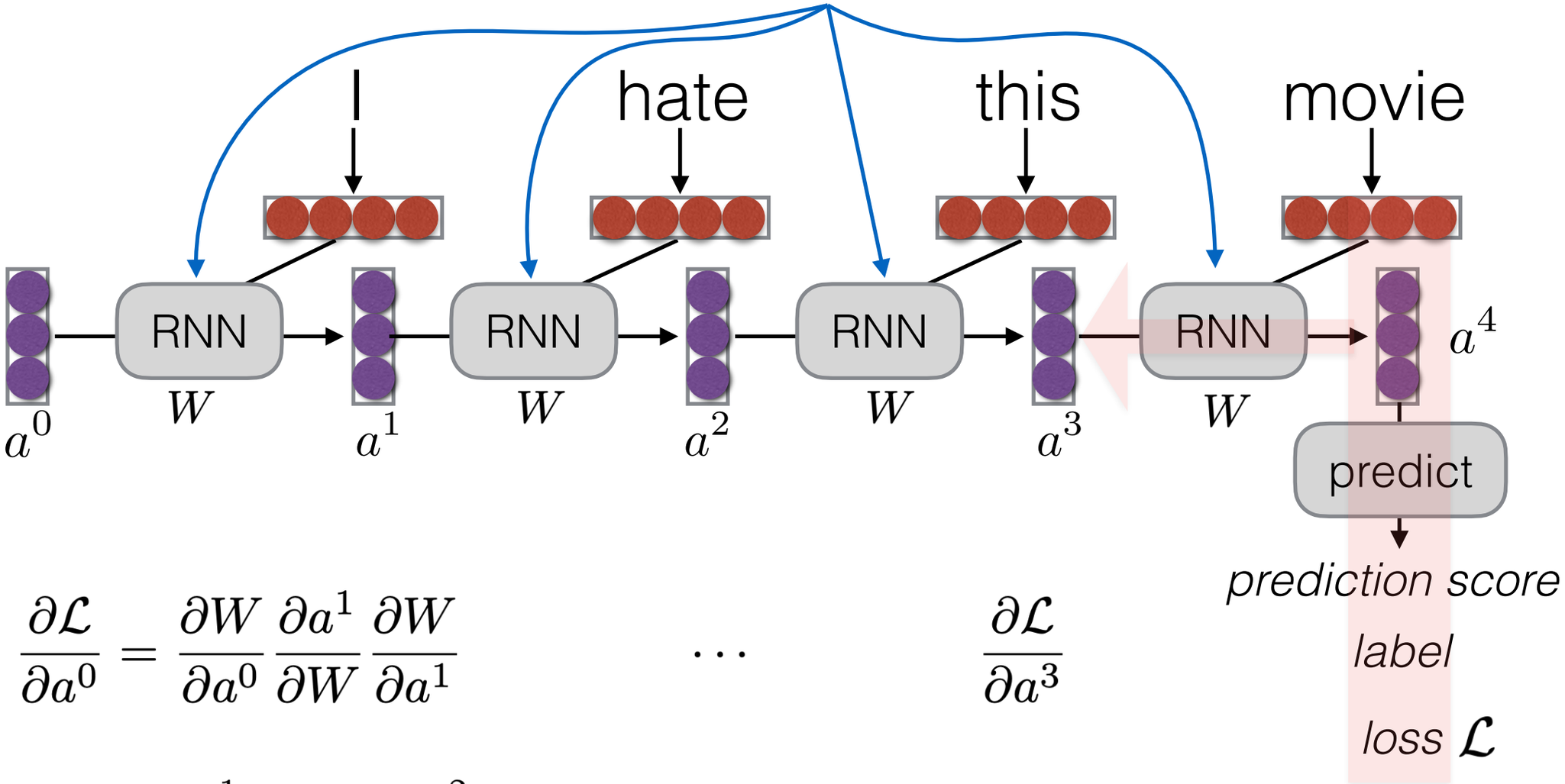


$$\frac{\partial \mathcal{L}}{\partial a^0} = \frac{\partial W}{\partial a^0} \frac{\partial a^1}{\partial W} \frac{\partial \mathcal{L}}{\partial a^1}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial a^1}{\partial W} \frac{\partial \mathcal{L}}{\partial a^1}$$

Backward pass

Parameters are shared! Derivatives are accumulated.



$$\frac{\partial \mathcal{L}}{\partial a^0} = \frac{\partial W}{\partial a^0}\frac{\partial a^1}{\partial W}\frac{\partial W}{\partial a^1} \quad \cdots \quad \frac{\partial \mathcal{L}}{\partial a^3}$$

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial a^1}{\partial W}\frac{\partial \mathcal{L}}{\partial a^1} + \frac{\partial a^2}{\partial W}\frac{\partial \mathcal{L}}{\partial a^2} + \cdots$$

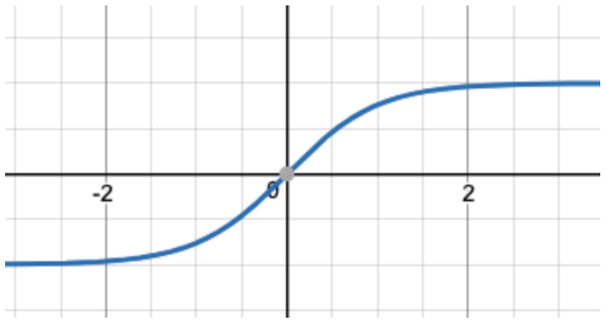
What happens if these gradients are small or large?

Vanishing/Exploding gradient: The gradient signal gets smaller (or larger) as it backpropagates further.

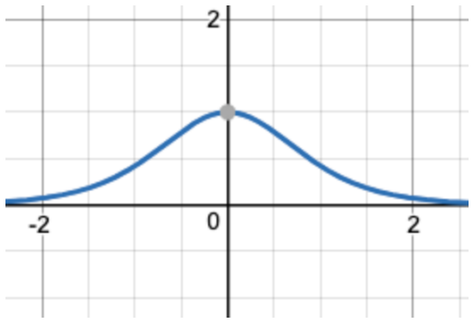
Vanishing/Exploding Gradients

Why is vanishing gradient a problem?

- Why? “Squashed” by **non-linearities** (e.g., tanh) or **multiplication of small weights** in matrices.



$$f(x) = \tanh(x)$$



$$\frac{\partial f}{\partial x} = 1 - \tanh(x)^2$$

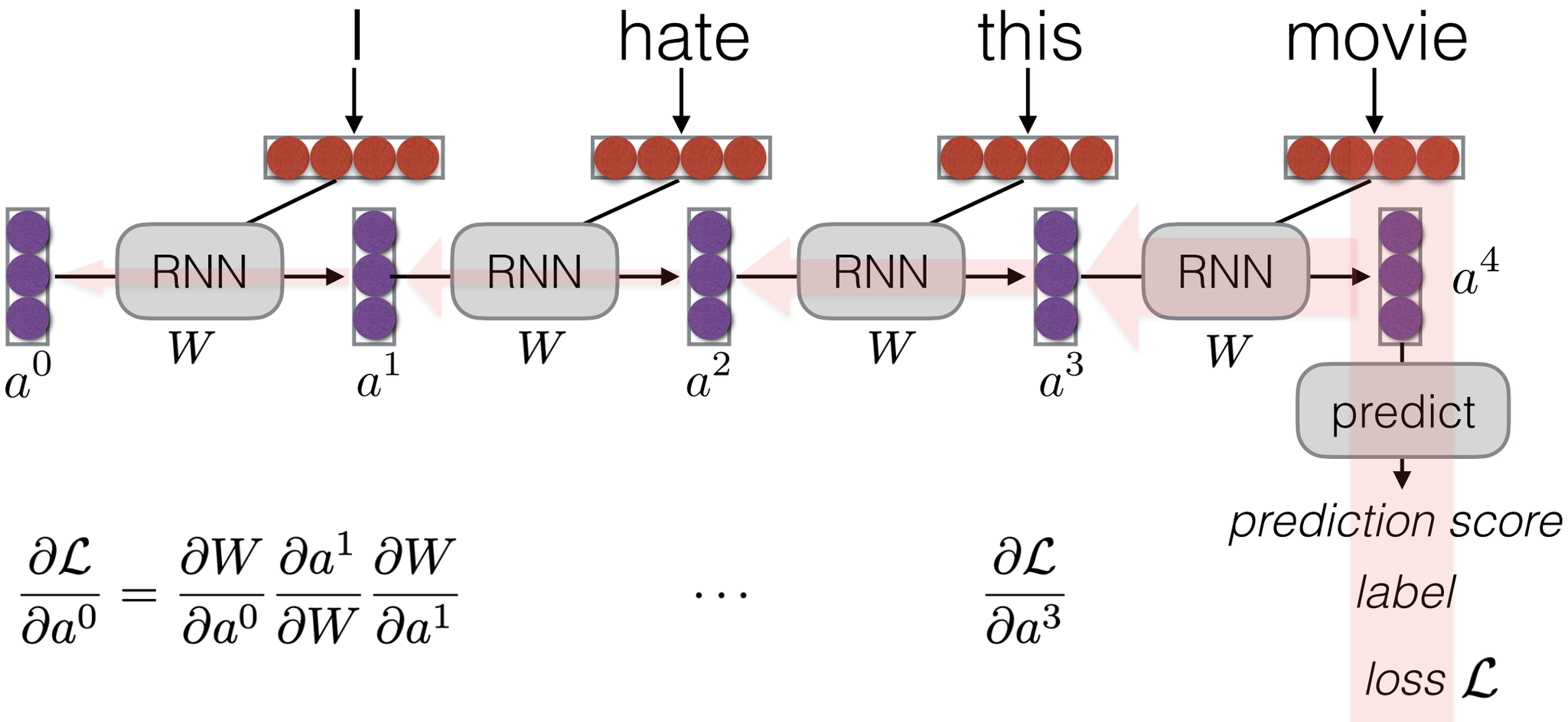
Easy to get derivatives less than 1

$$\frac{\partial \mathcal{L}}{\partial a^0} = \frac{\partial \mathcal{W}}{\partial a^0} \frac{\partial a^1}{\partial W} \frac{\partial W}{\partial a^1} \cdots$$

Small matrix multiplication

Why is vanishing gradient a problem?

- Why? "Squashed" by **non-linearities** (e.g., tanh) or **multiplication of small weights** in matrices.



$$\frac{\partial \mathcal{L}}{\partial a^0} = \frac{\partial W}{\partial a^0} \frac{\partial a^1}{\partial W} \frac{\partial W}{\partial a^1} \quad \cdots \quad \frac{\partial \mathcal{L}}{\partial a^3}$$

Gradient signal from far away is lost because it's much smaller than **gradient signal from close-by**.
So, model parameters are updated only with respect to **near effects**, not **long-term effects**.

A Solution:
Long Short-term Memory
(Hochreiter and Schmidhuber 1997)

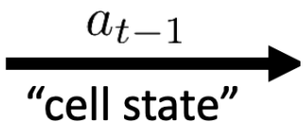
- **Basic idea:** make **additive** connections between time steps

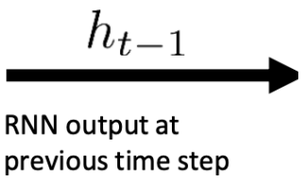
- Addition does not modify the last step's gradient, no vanishing

- Gates to control the information flow

Long short-term memory

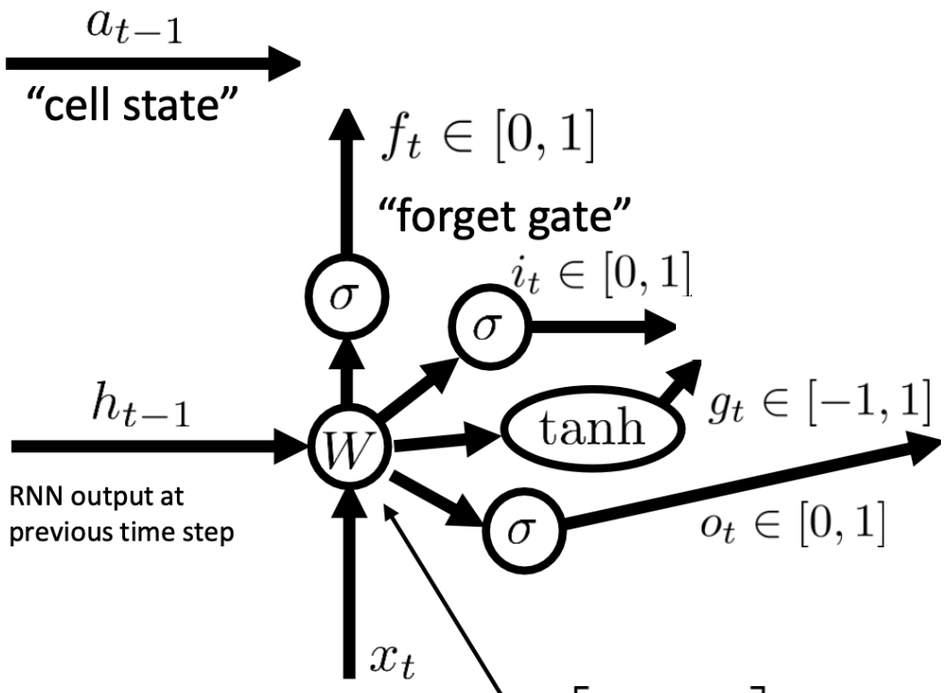
- In practice, LSTM works much better than a vanilla RNN
  - Long term memory cell: a_{t-1} <=> Short term memory state: h_{t-1}





Long short-term memory

- In practice, LSTM works much better than a vanilla RNN
  - Long term memory cell: a_{t-1} <=> Short term memory state: h_{t-1}
  - Forget gate f_t: how much memory do we want to forget
  - Input gate i_t: how much information do we add to the memory
  - g_t: compute new values we want to add to the memory cell
  - Output gate o_t : how much information do we reflect in the next state



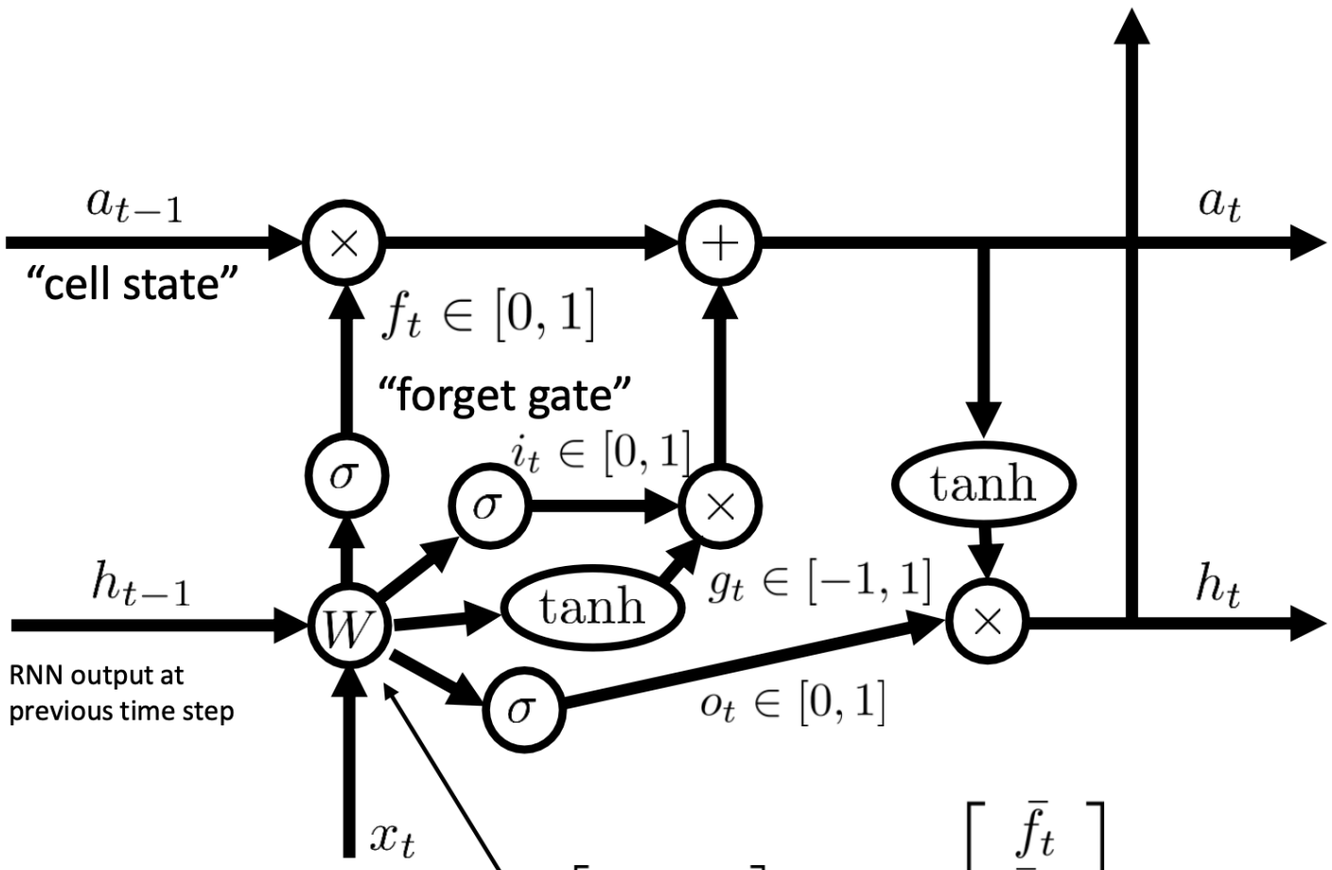
$$W \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} + b = \begin{bmatrix} \bar{f}_t \\ \bar{i}_t \\ \bar{g}_t \\ \bar{o}_t \end{bmatrix}$$

$$f_t = \sigma(\bar{f}_t)$$
$$i_t = \sigma(\bar{i}_t)$$
$$g_t = \tanh(\bar{g}_t)$$
$$o_t = \sigma(\bar{o}_t)$$
$$\sigma : \text{sigmoid}$$
$$f_t, i_t, g_t, o_t \in \mathbb{R}^d$$

$W \in \mathbb{R}^{4d \times 2d}$ in LSTM is 4x larger in dimensionality than $W \in \mathbb{R}^{d \times 2d}$ in RNN

Long short-term memory

- In practice, LSTM works much better than a vanilla RNN
  - Long term memory cell: a_{t-1} <=> Short term memory state: h_{t-1}
  - Forget gate f_t: how much memory do we want to forget
  - Input gate i_t: how much information do we add to the memory
  - g_t: compute new values we want to add to the memory cell
  - Output gate o_t : how much information do we reflect in the next state

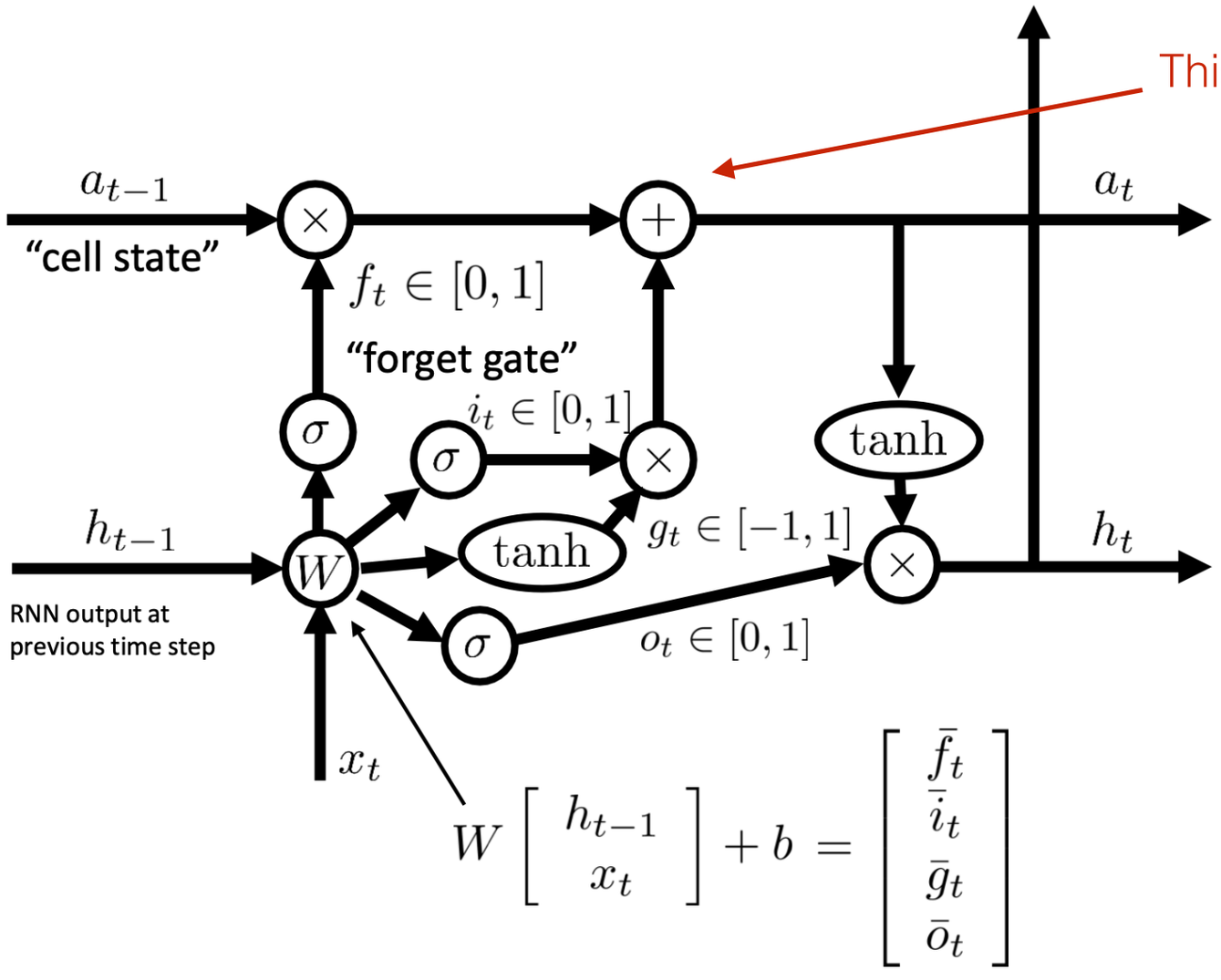


$$a_t = f_t \otimes a_{t-1} + i_t \otimes g_t$$

$$h_t = o_t \otimes \tanh(a_t)$$

where $\otimes$ is element-wise product

Long short-term memory



$$a_t = f_t \otimes a_{t-1} + i_t \otimes g_t$$

changes very little step to step!
"long term memory"

$$h_t = o_t \otimes \tanh(a_t)$$

changes all the time (multiplicative)!
"short term memory"

Why is exploding gradient a problem?

- If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}}{\partial \theta}$$

- This can cause **bad updates**, if we take too large a step and obtain weird and bad parameters.

- In the worst case, this will result in **Inf** or **NaN** error in the network (then you have to restart training from an earlier checkpoint).

Gradient clipping: solution for exploding gradient

- If the norm of the gradient is greater than some threshold, scale it down before applying SGD update

- Intuition: take a step in the same direction, but a smaller step

Algorithm 1 Pseudo-code for norm clipping

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$
end if

Aside: distributional shift (or exposure bias)

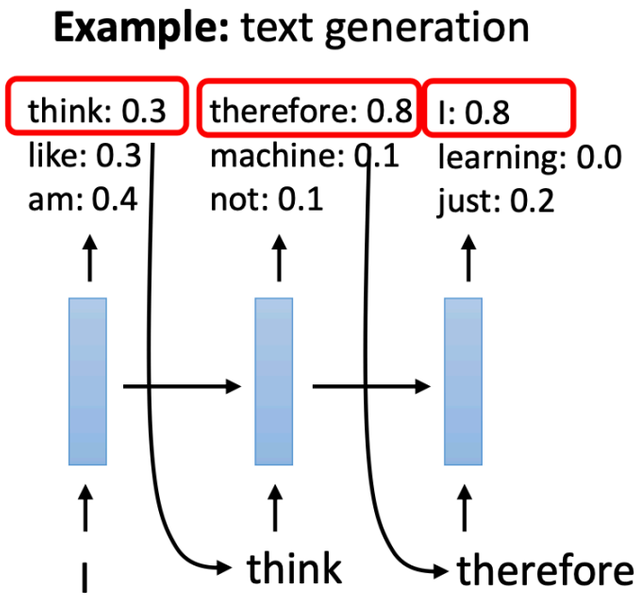
- During training, we feed ground-truth words as the context to predict the next word
- During testing, we feed the model's predicted words as the context to predict the next word

I think therefore I am

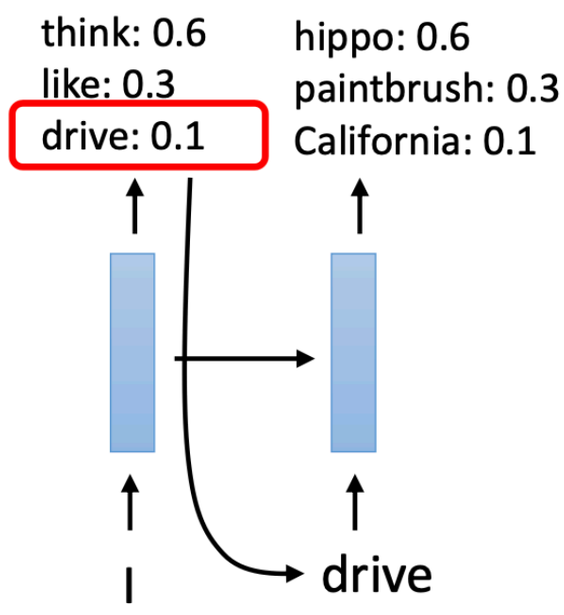
I like machine learning

I am not just a neural network

Training data



Training phrase

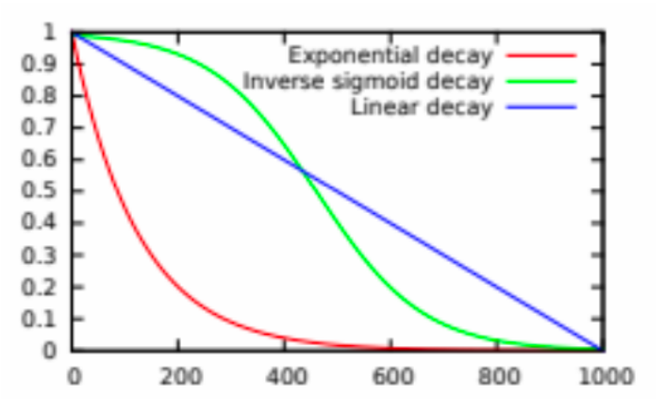


Testing phrase

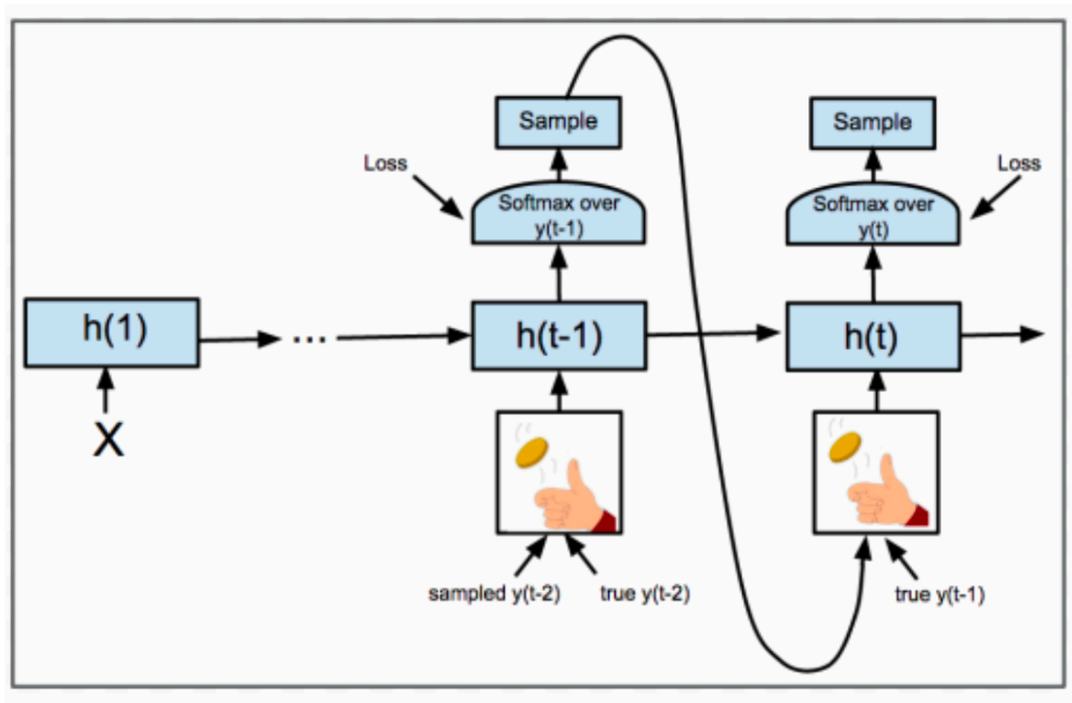
Distributional shift: input distribution shifts from true strings (at training) to predicted strings (at test time)
Even **one** random mistake can completely scramble the output!

Aside: scheduled sampling

- An old trick from reinforcement learning adapted to training RNNs
  - Start by feeding in ground-truth tokens as input, since model predictions are mostly nonsense.
  - In the middle of training, randomly decide whether to give the network a ground-truth token or its own previous prediction.
  - At the end of training, mostly feed in the model's own predictions, to mitigate distribution shift.



Schedules for probability of using ground-truth token

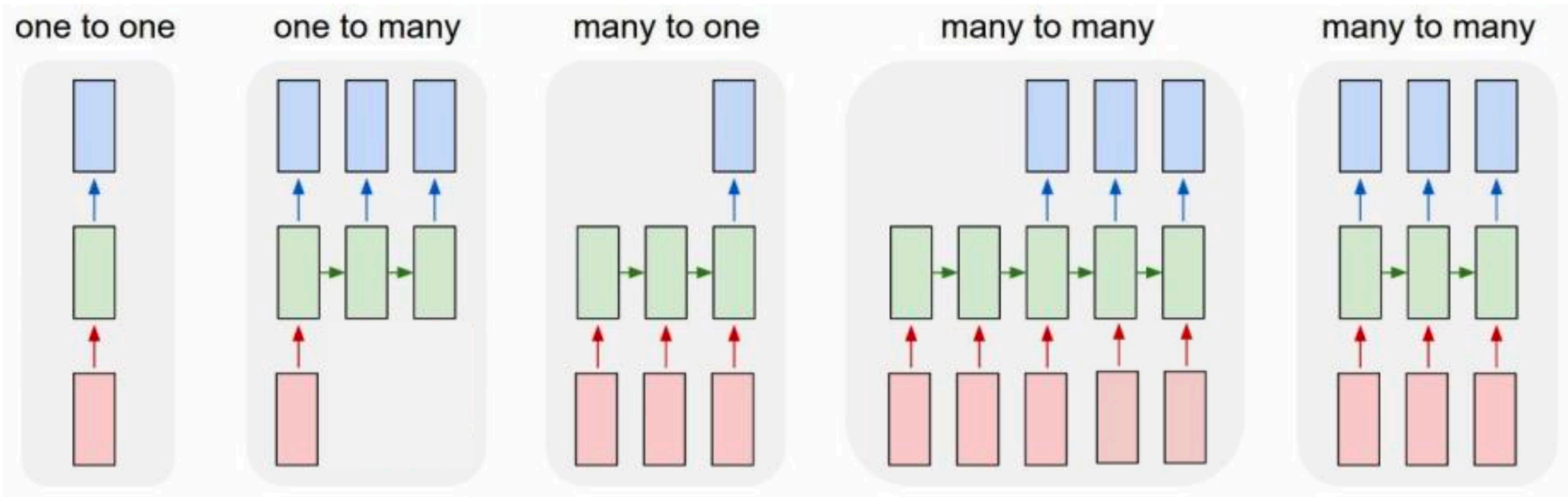


Applications of RNNs

What Can RNNs Do?

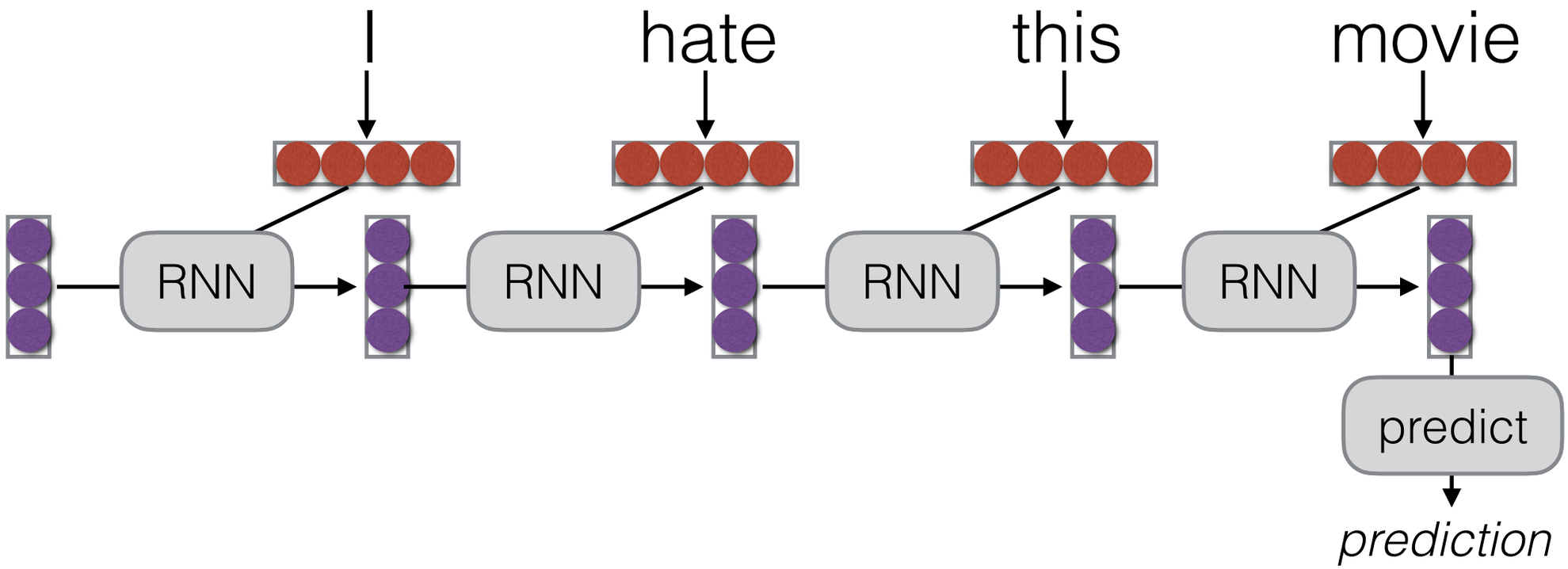
- Represent a sentence
  - Read whole sentence, make a prediction
- Represent a context within a sentence
  - Read context up until that point

Different ways to use RNNs



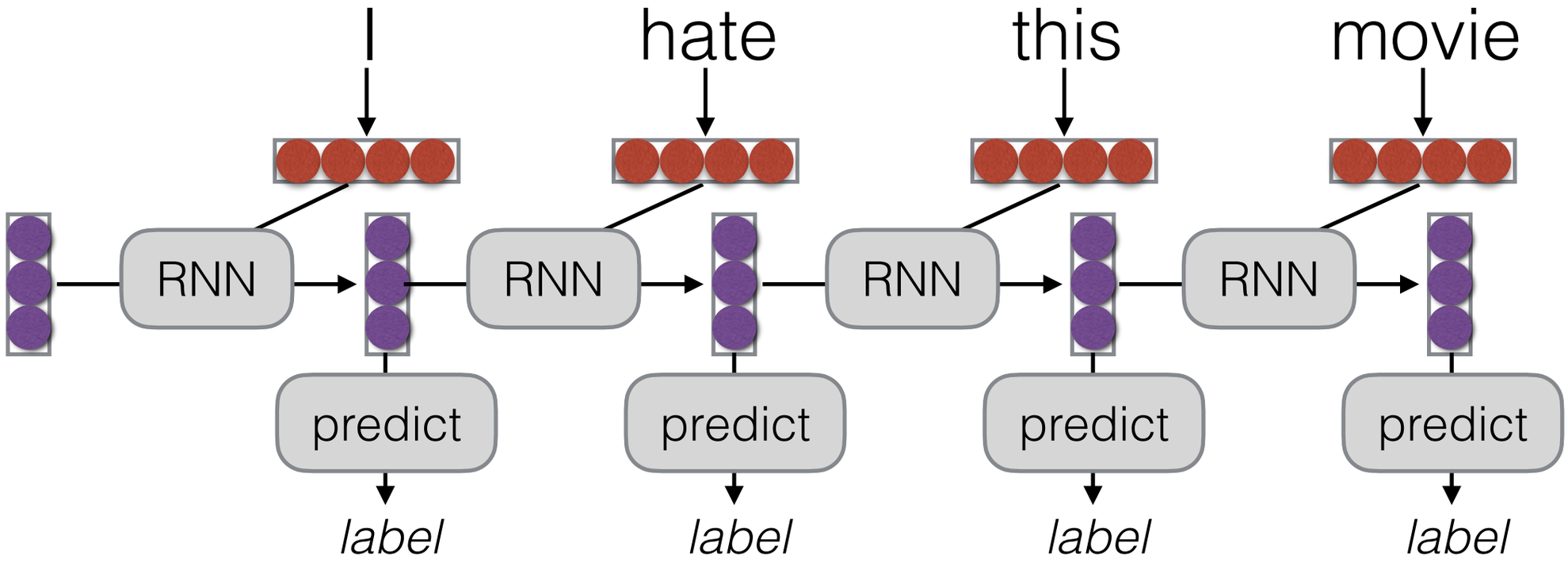
e.g., image captioning e.g., sentence classification e.g., machine translation e.g., sentence tagging (POS, NER)

Encoding Sentences



- Binary or multi-class prediction
- Sentence representation for retrieval, sentence comparison, etc.

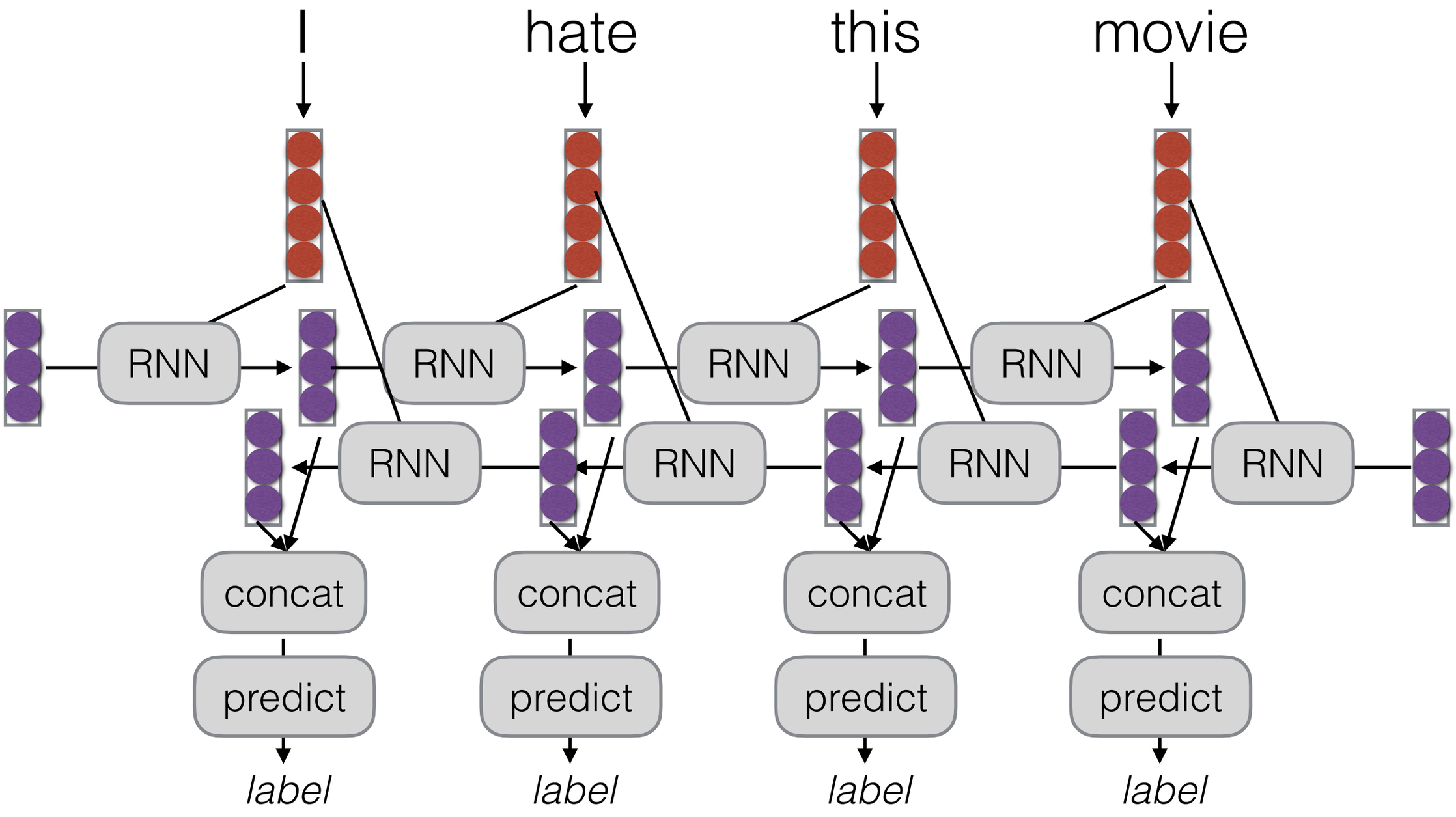
Representing Contexts



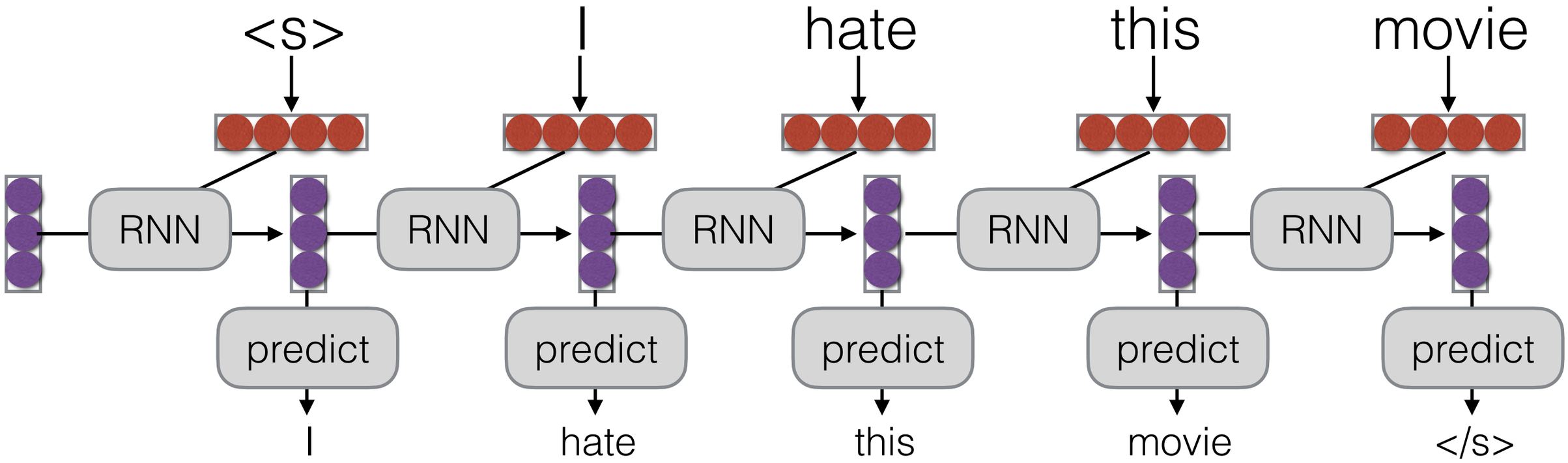
- Sequence labeling
- Calculating representations for parsing, etc.

Bi-RNNs

- A simple extension, run the RNN in both directions



e.g. Language Modeling



- Language modeling is like a tagging task, where each tag is the next word!

- Note: this is an autoregressive model

Understanding RNNs

What can LSTMs Learn? (1)

(Karpathy et al. 2015)

- Additive connections make single nodes surprisingly interpretable

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."
```

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;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
```

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;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

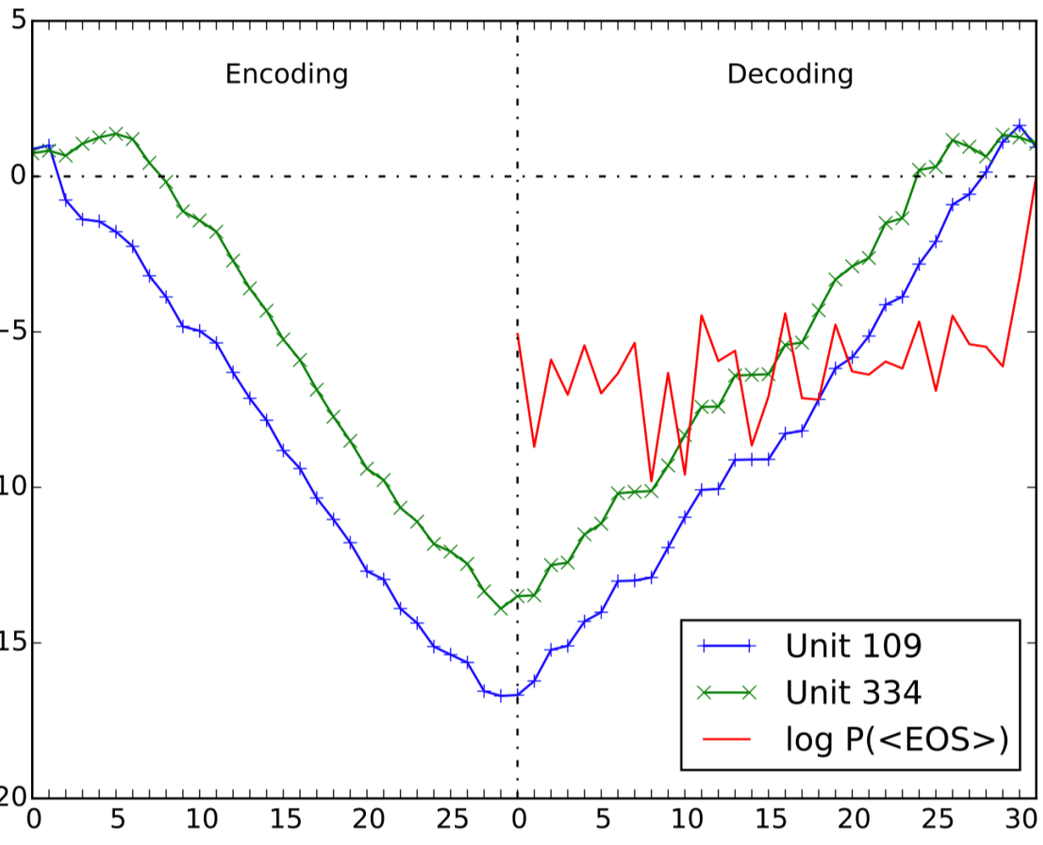
```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
   * defines the longest valid length.
   */
  if (len > PATH_MAX)
    return ERR_PTR(-ENAMETOOLONG);
  str = kmalloc(len + 1, GFP_KERNEL);
  if (unlikely(!str))
    return ERR_PTR(-ENOMEM);
  memcpy(str, *bufp, len);
  str[len] = 0;
  *bufp += len;
  *remain -= len;
  return str;
}
```

Text color corresponds to *tahn(c)*, where -1 is red and +1 is blue, where c is one element in the memory cell.

What can LSTMs Learn? (2)

(Shi et al. 2016, Radford et al. 2017)

Count length of sentence



Sentiment

25 August 2003 League of Extraordinary Gentlemen: Sean Connery is one of the all time greats and I have been a fan of his since the 1950's. I went to this movie because Sean Connery was the main actor. I had not read reviews or had any prior knowledge of the movie. The movie surprised me quite a bit. The scenery and sights were spectacular, but the plot was unreal to the point of being ridiculous. In my mind this was not one of his better movies it could be the worst. Why he chose to be in this movie is a mystery. For me, going to this movie was a waste of my time. I will continue to go to his movies and add his movies to my video collection. But I can't see wasting money to put this movie in my collection

I found this to be a charming adaptation, very lively and full of fun. With the exception of a couple of major errors, the cast is wonderful. I have to echo some of the earlier comments -- Chynna Phillips is horribly miscast as a teenager. At 27, she's just too old (and, yes, it DOES show), and lacks the singing "chops" for Broadway-style music. Vanessa Williams is a decent-enough singer and, for a non-dancer, she's adequate. However, she is NOT Latina, and her character definitely is. She's also very STRIDENT throughout, which gets tiresome. The girls of Sweet Apple's Conrad Birdie fan club really sparkle -- with special kudos to Brigitta Dau and Chiara Zanni. I also enjoyed Tyne Daly's performance, though I'm not generally a fan of her work. Finally, the dancing Shriners are a riot, especially the dorky three in the bar. The movie is suitable for the whole family, and I highly recommend it.

Notably, difficult for GRUs!

RNNs as Turing Machines
(Siegelmann and Sontag 1992)

- Real-valued recurrent nets can calculate arbitrary functions

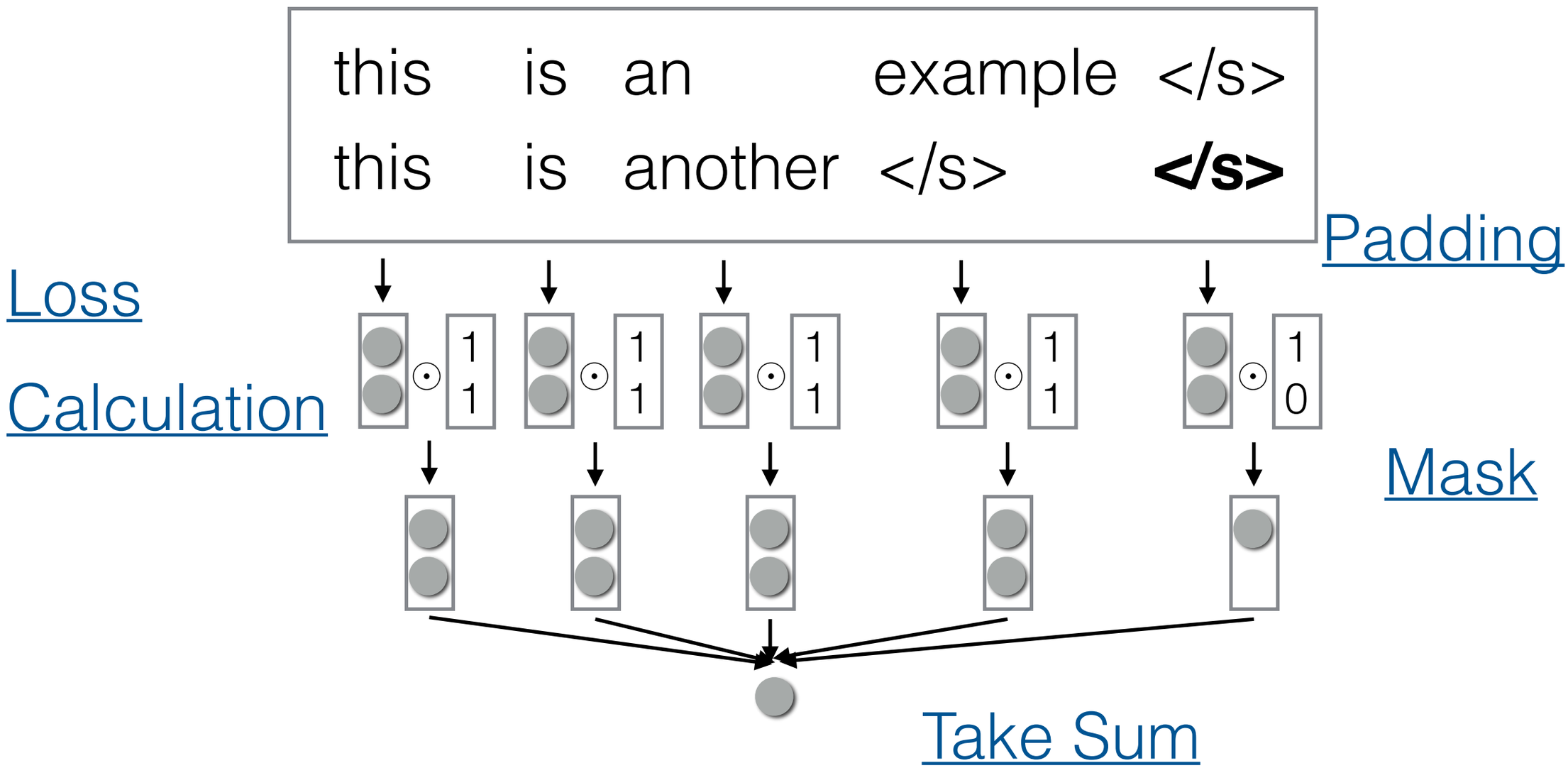
Weights	Recognition Power
integer	regular
rational	recursive
real	arbitrary

Efficiency Tricks

Handling Mini-batching

- Mini-batching makes things much faster!
- But mini-batching in RNNs is harder than in feed-forward networks
  - Each word depends on the previous word
  - Sequences are of various length

Mini-batching Method



Bucketing/Sorting

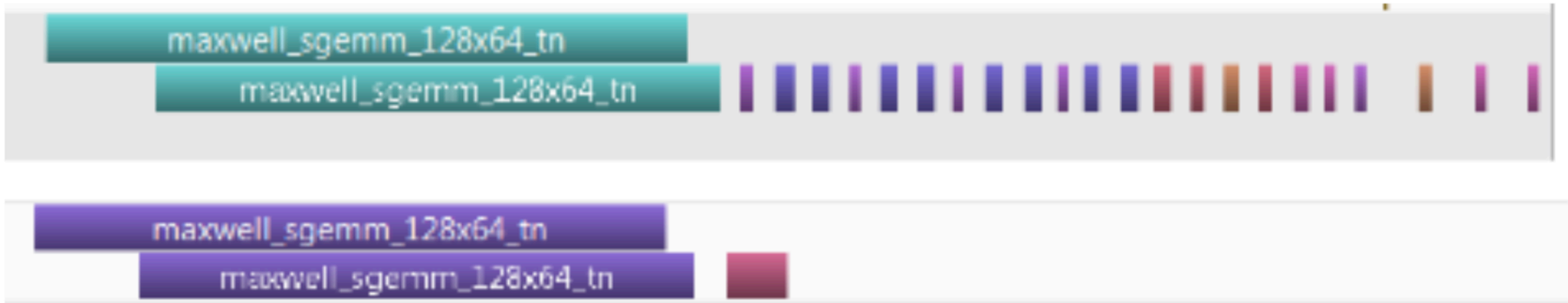
- If we use sentences of different lengths, too much padding and sorting can **result in decreased performance**

- To remedy this: **sort sentences** so similarly-lengthed sentences are in the same batch

Optimized Implementations of LSTMs

(Appleyard 2015)

- In simple implementation, still need one GPU call for each time step
- For some RNN variants (e.g. LSTM) efficient full-sequence computation supported by CuDNN



- **Basic process:** combine inputs into tensor, single GPU call
- **Downside:** significant loss of flexibility

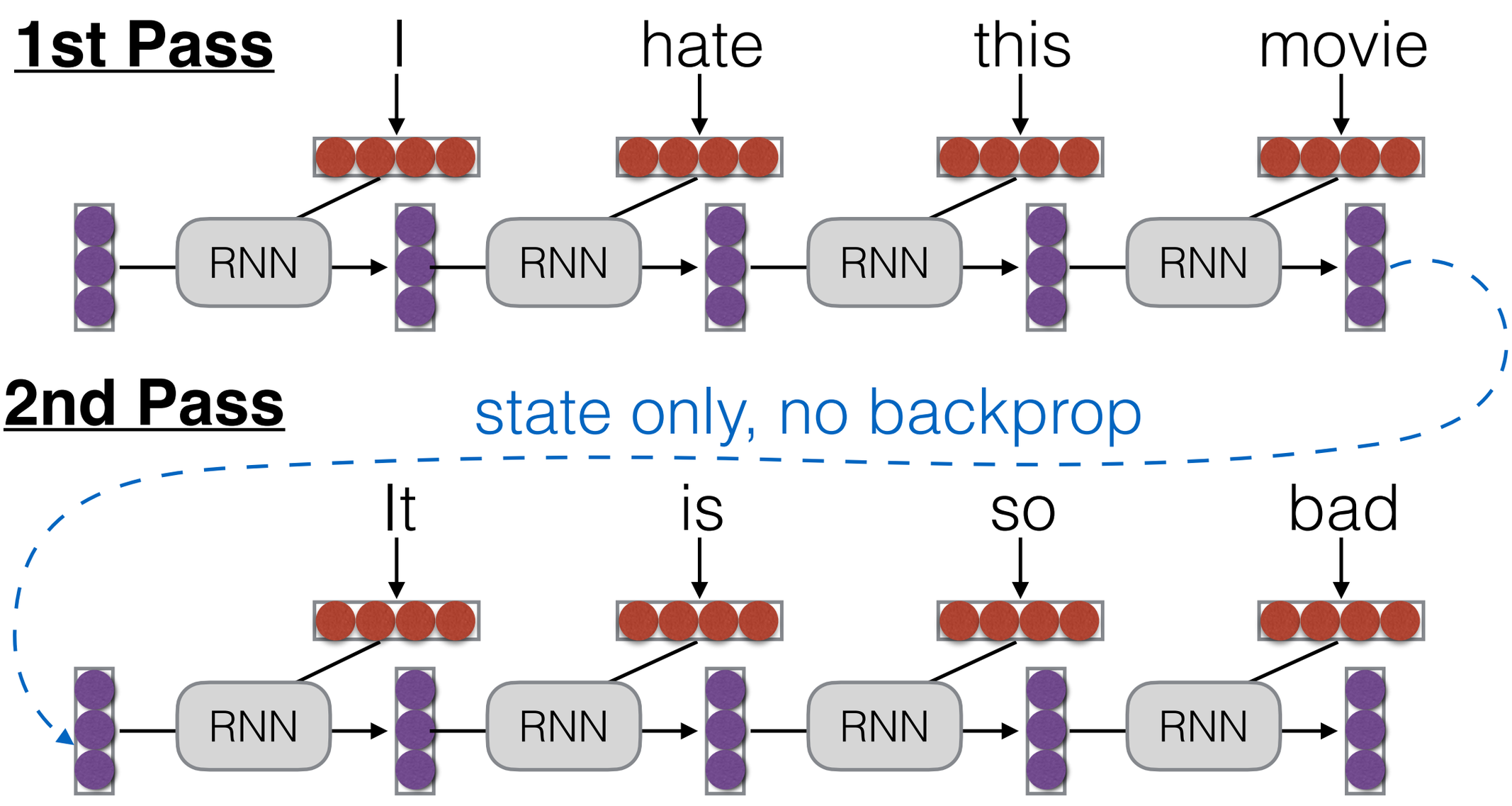
Handling Long Sequences

Handling Long Sequences

- Sometimes we would like to capture long-term dependencies over long sequences
- e.g. words in full documents
- However, this may not fit on (GPU) memory

Truncated BPTT

- Backprop over shorter segments, initialize w/ the state from the previous segment



RNN Variants

Gated Recurrent Units

(Cho et al. 2014)

- A simpler version that preserves the additive connections

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

Additive or Non-linear

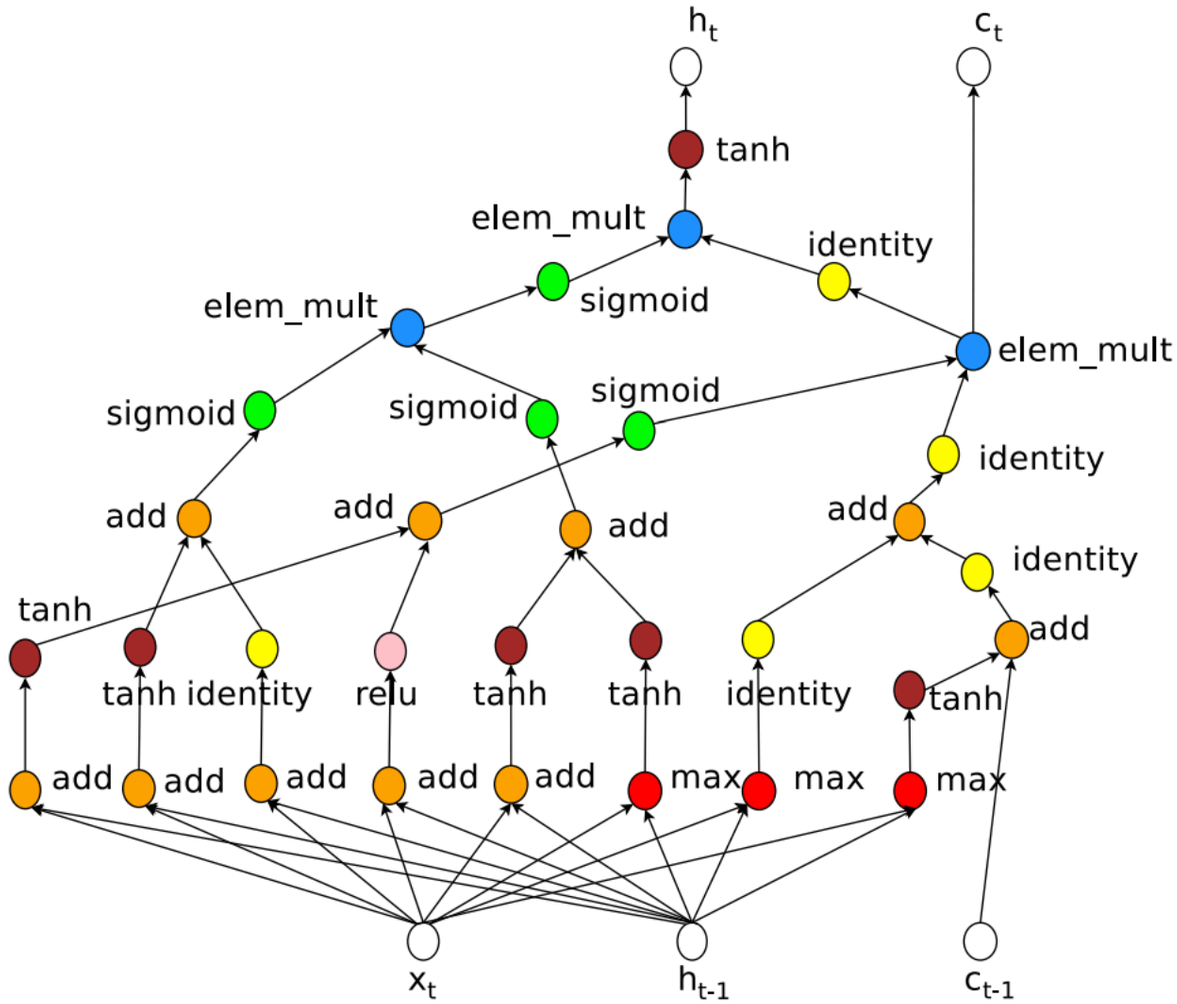
- **Note:** GRUs cannot do things like simply count

Extensive Architecture Search for LSTMs

(Greffen et al. 2015, Zoph and Le 2017)

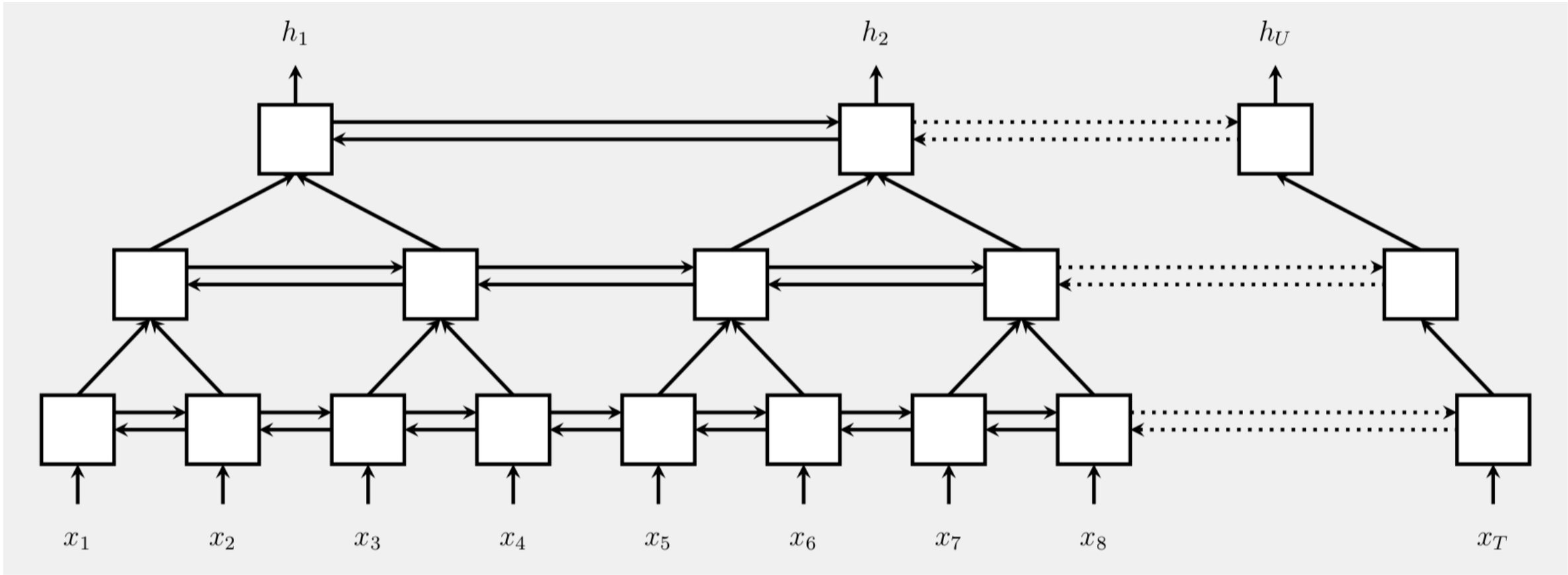
- Many different types of architectures tested for LSTMs (manually or automatically)

- **Conclusion:** basic LSTM quite good, other variants (e.g. coupled input/forget gates) reasonable



Multi-scale/Pyramidal RNN
(e.g. Chan et al. 2015)

- Have different RNNs of different scales



Types of Prediction
Recurrent Neural Nets
RNN Applications
Vanishing Gradients

Understanding RNNs
Efficiency Tricks
Handling Long Sequences
RNN Variants

Questions?