Exploding and vanishing gradients
Solutions

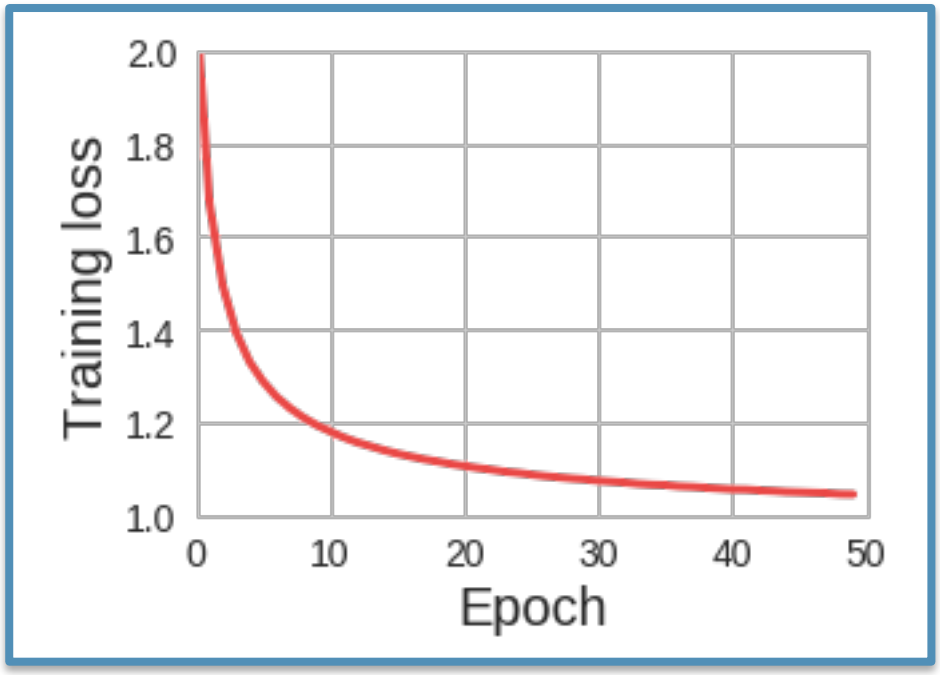
Previously on this week: exploding gradients

- Gradient norms may become very large or even NaNs in the worst case
- Exploding gradients make the learning process unstable

Exploding gradients: detection

Exploding gradients are easy to detect

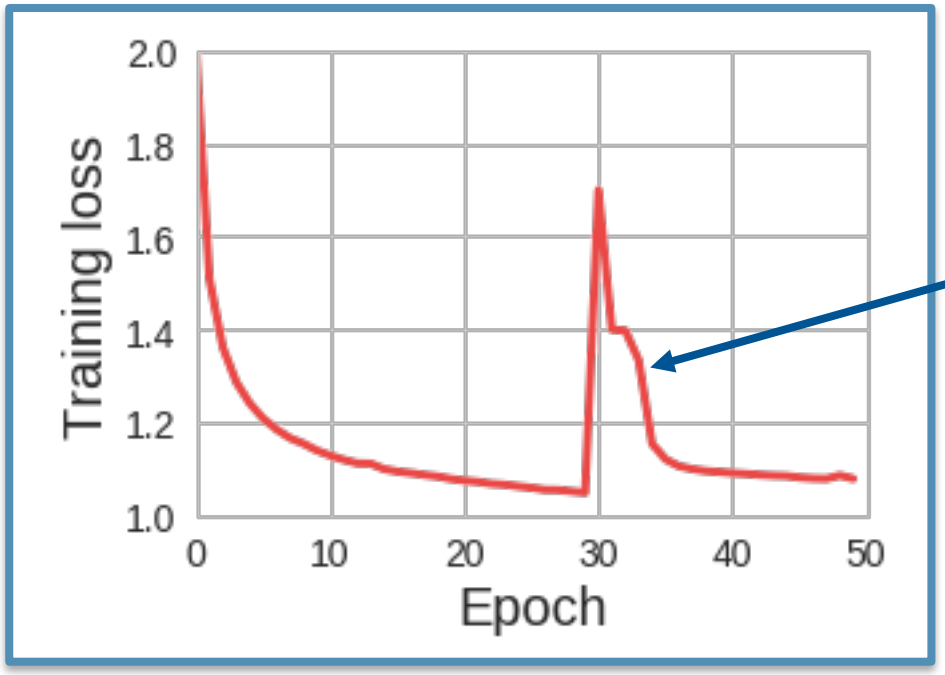
Stable learning curve



Exploding gradients: detection

Exploding gradients are easy to detect

Unstable learning curve



This is it!

If the gradients contain NaNs you end up with NaNs in the weights

Gradient clipping

Exploding gradients are easy to detect

Gradient $g = \frac{\partial L}{\partial \theta}$,θ - all the network parameters

If $||g|| >$ threshold:

$$g \leftarrow \frac{threshold}{\|g\|} g$$

Simple but still very effective!

Gradient clipping

Exploding gradients are easy to detect

Gradient $g = \frac{\partial L}{\partial \theta}$,θ - all the network parameters

If $||g|| >$ threshold:

$$g \leftarrow \frac{threshold}{\|g\|} g$$

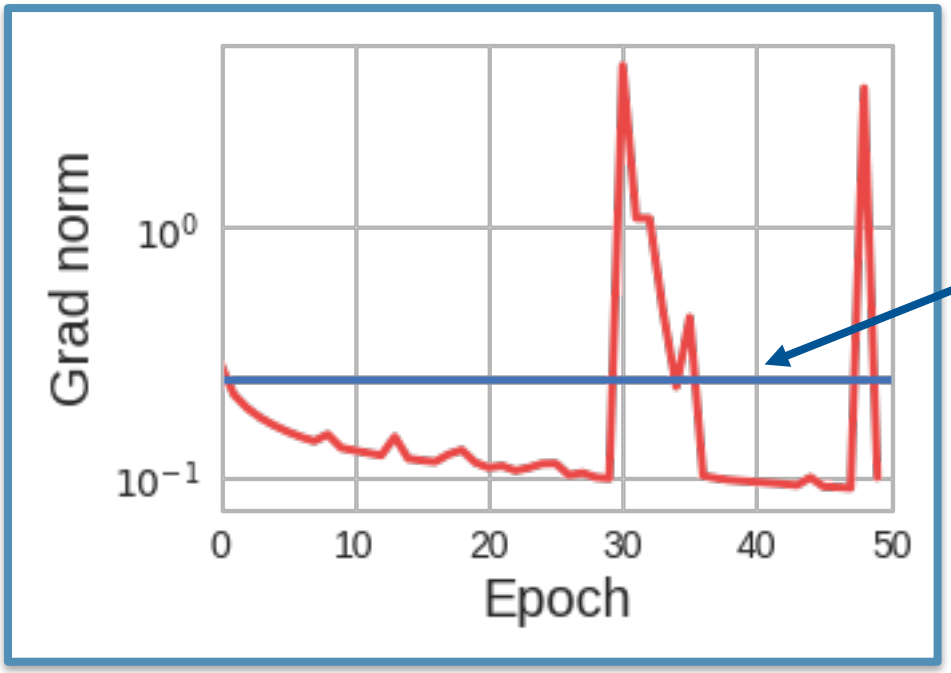
Simple but still very effective!

It is enough to clip $\frac{\partial h_t}{\partial h_{t-1}}$

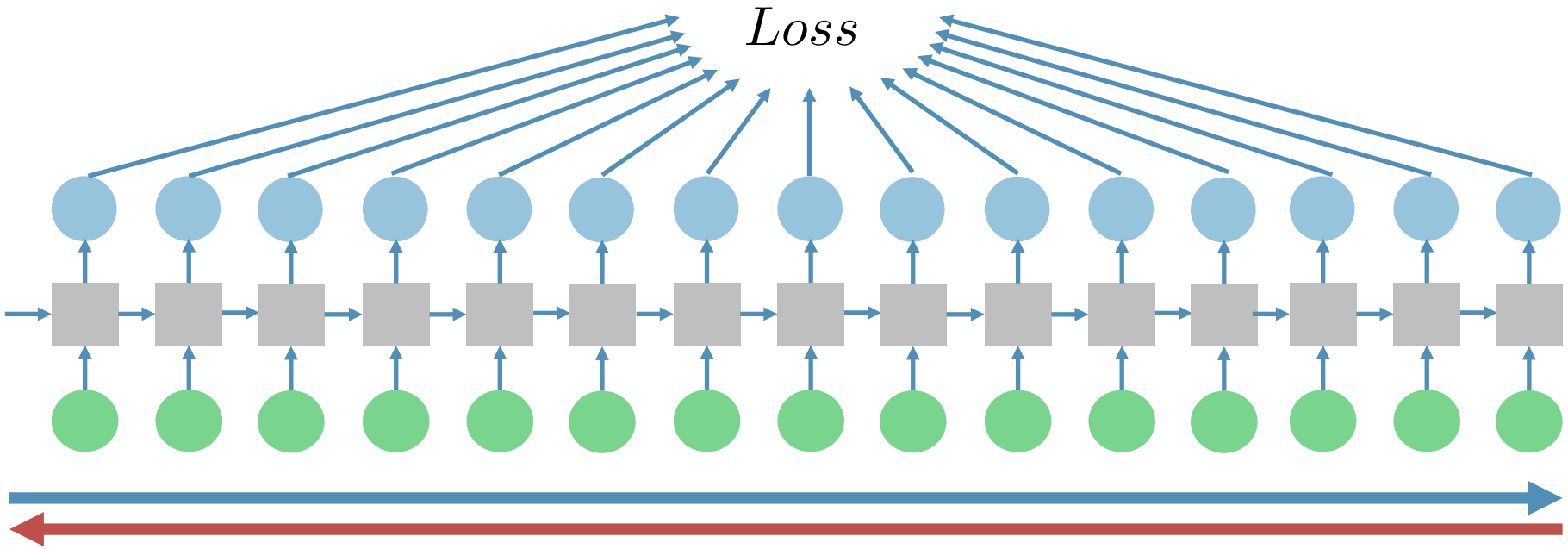
Gradient clipping: threshold

Choose the highest threshold which helps to overcome the exploding gradient problem

Curve of the gradient norm



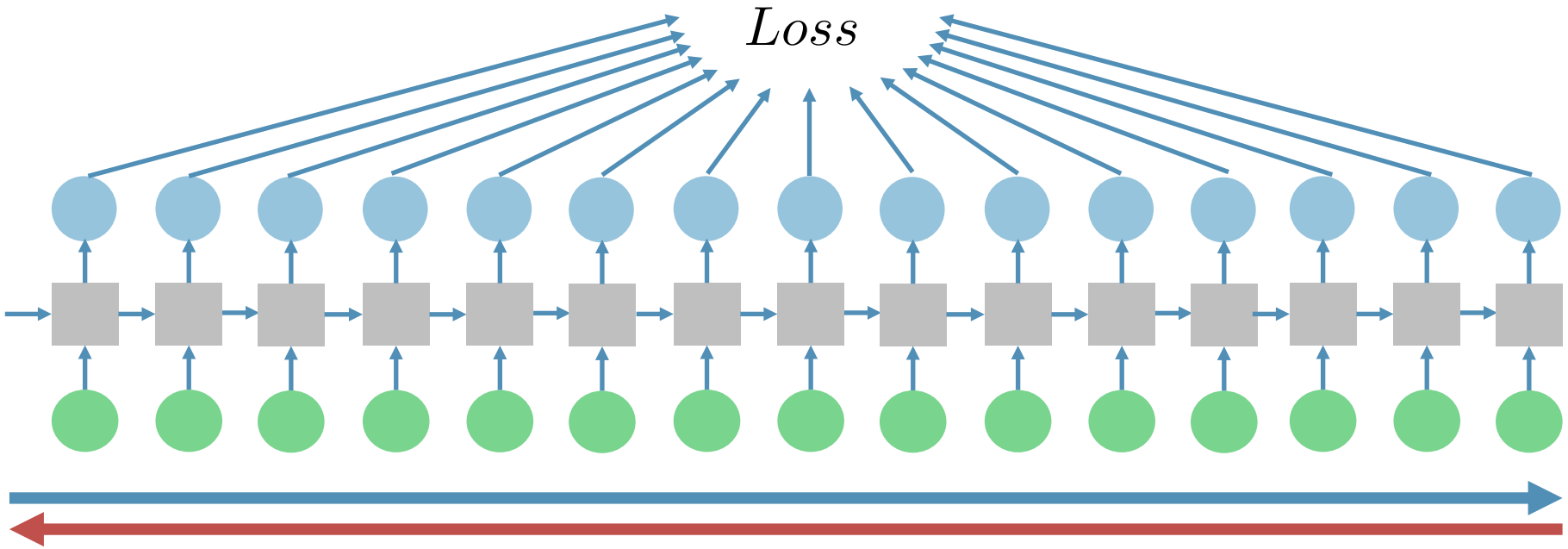
Previously on this week: BPTT



Forward pass through the entire sequence to compute the loss

Backward pass through the entire sequence to compute the gradient

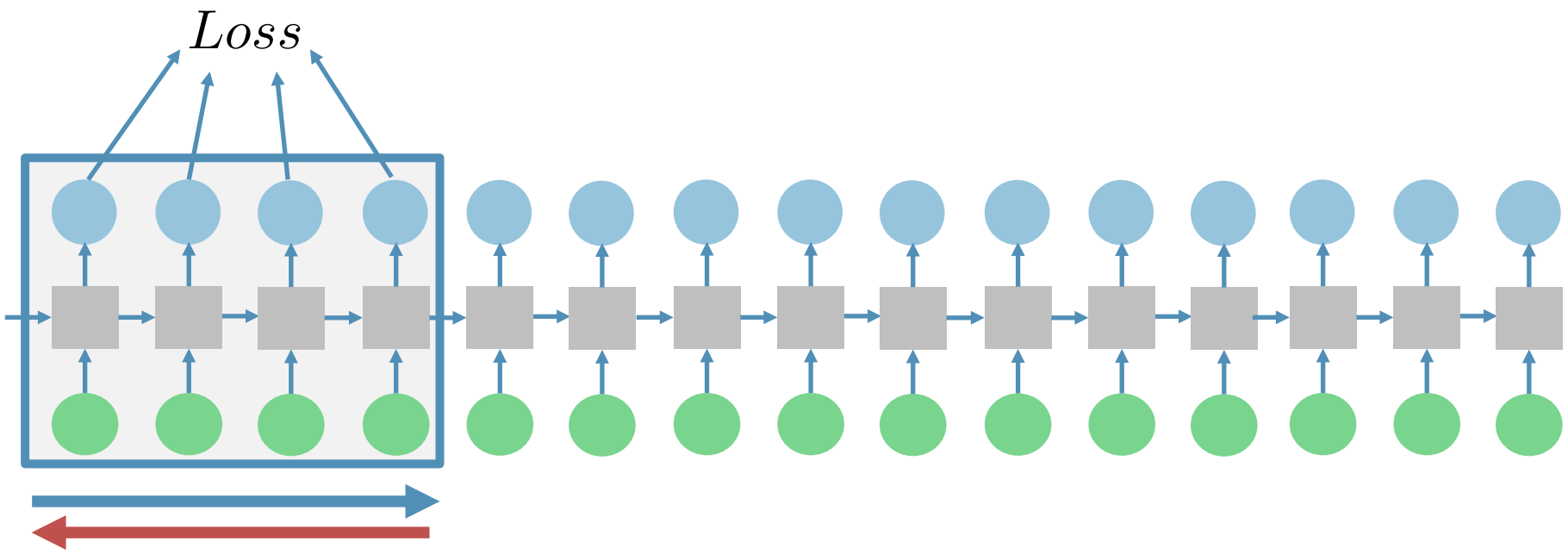
Previously on this week: BPTT



And what if we have very long training sequences?

Way too expensive + exploding gradients

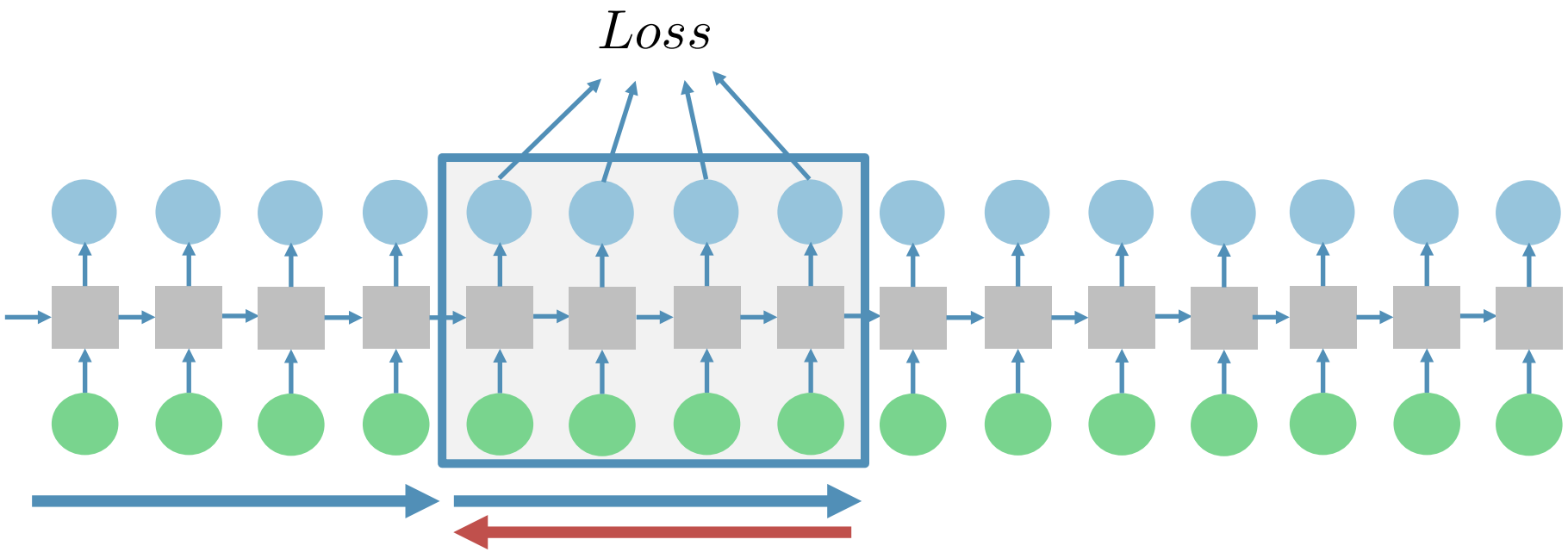
Truncated BPTT



Let's run forward and backward passes through the chunks of the sequence instead of the whole sequence.

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

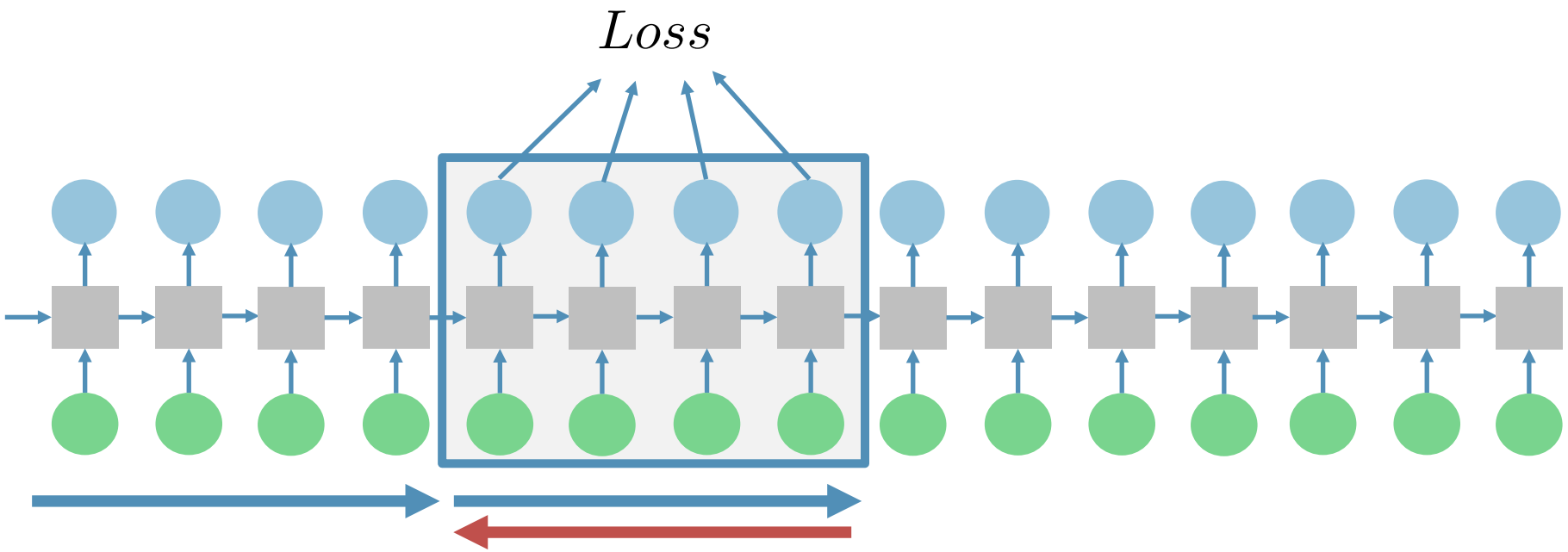
Truncated BPTT



Let's run forward and backward passes through the chunks of the sequence instead of the whole sequence.

Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

Truncated BPTT



Truncated BPTT is much faster but it doesn't come without a price! Dependencies longer than the chunk size don't affect the training but at least they still work at forward pass.

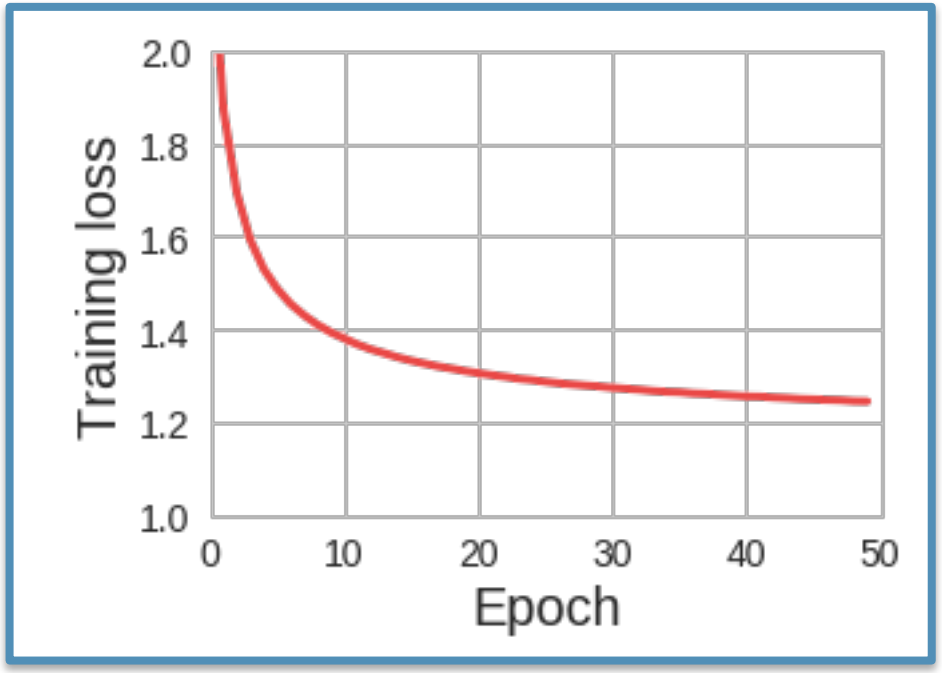
Previously on this week: vanishing gradients

- Contributions from faraway steps vanish and don't affect the training
- It is difficult to learn long-range dependencies

Vanishing gradients: detection

It is not clear how to detect vanishing gradients

Learning curve



Does the gradient vanish or the task is difficult?

Vanishing gradients: detection

It is not clear how to detect vanishing gradients

Gradient norm

$$\left\| \frac{\partial L_t}{\partial h_{t-100}} \right\|_2$$ is small

Does the gradient vanish or there are no long-range dependencies in the data?

Vanishing gradients: how to deal with them?

- LSTM, GRU
- ReLU activation function
- Initialization of the recurrent weight matrix
- Skip connections
- …

ReLU activation function

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial pr_t}\frac{\partial pr_t}{\partial h_{t-1}} = \boxed{diag(f_h'(pr_t))} \cdot W$$

Let's use the ReLU which is much more resistant to the vanishing gradient problem.

Initialization of the recurrent weight matrix

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial pr_t}\frac{\partial pr_t}{\partial h_{t-1}} = diag(f_h'(pr_t)) \cdot W$$

Q is orthogonal if $Q^T = Q^{-1}$ =>

$\prod_i Q_i$ doesn't explode or vanish

Initialization of the recurrent weight matrix

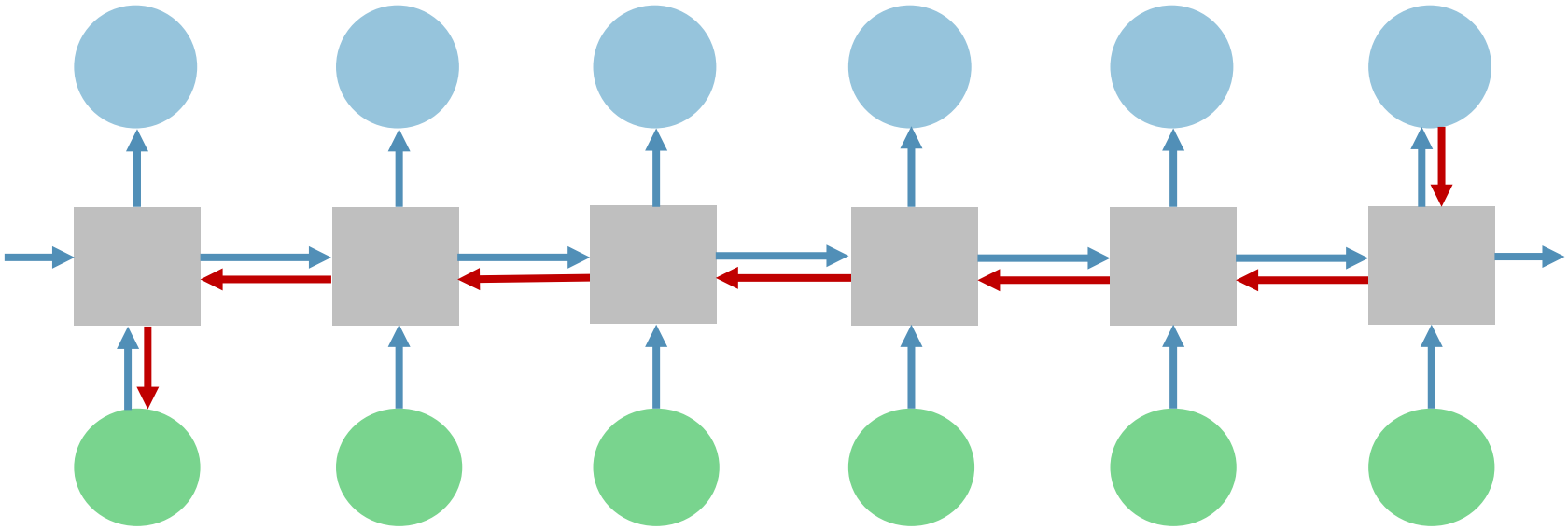
$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial pr_t}\frac{\partial pr_t}{\partial h_{t-1}} = diag(f_h'(pr_t)) \cdot \boxed{W}$$

Q is orthogonal if $Q^T = Q^{-1}$ =>

$\prod_i Q_i$ doesn't explode or vanish

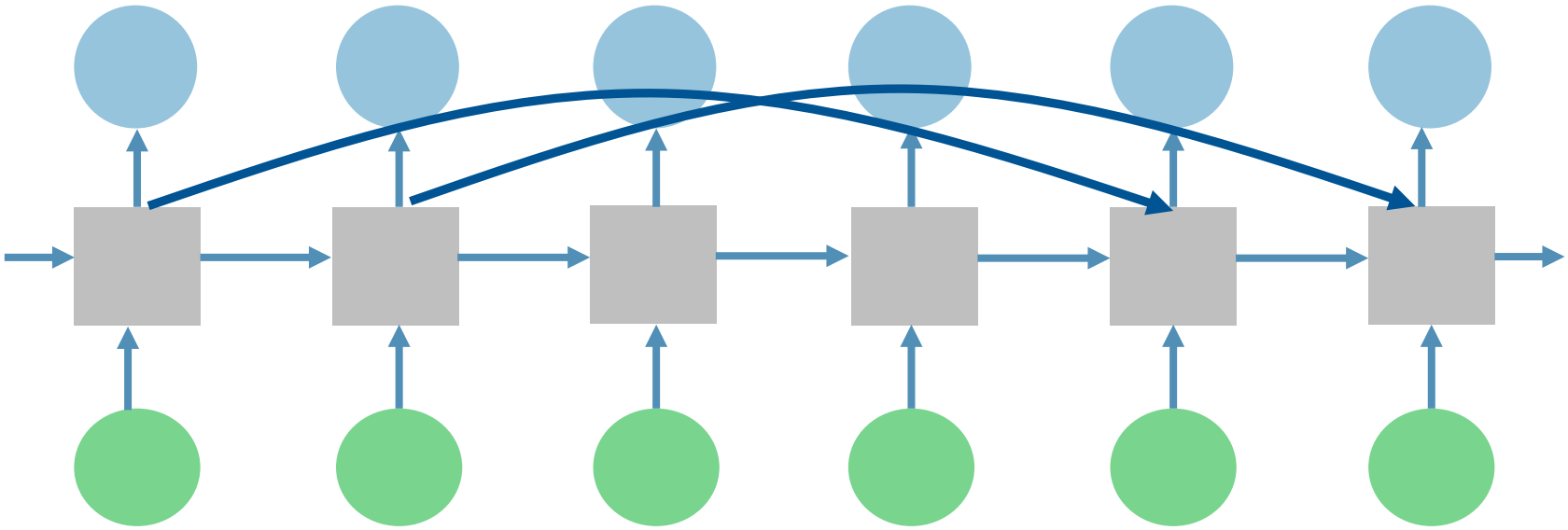
- Initialise W with an orthogonal matrix
- Use orthogonal W through the whole training

Skip connections



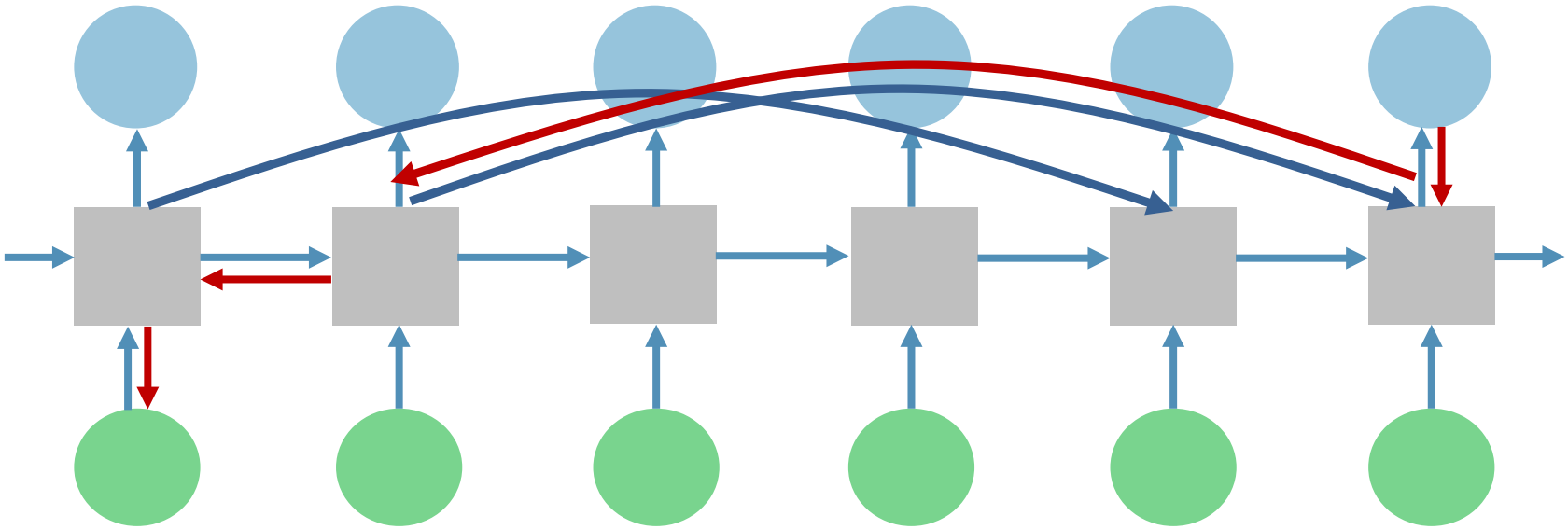
Very long ways for the gradients => vanishing gradients

Skip connections



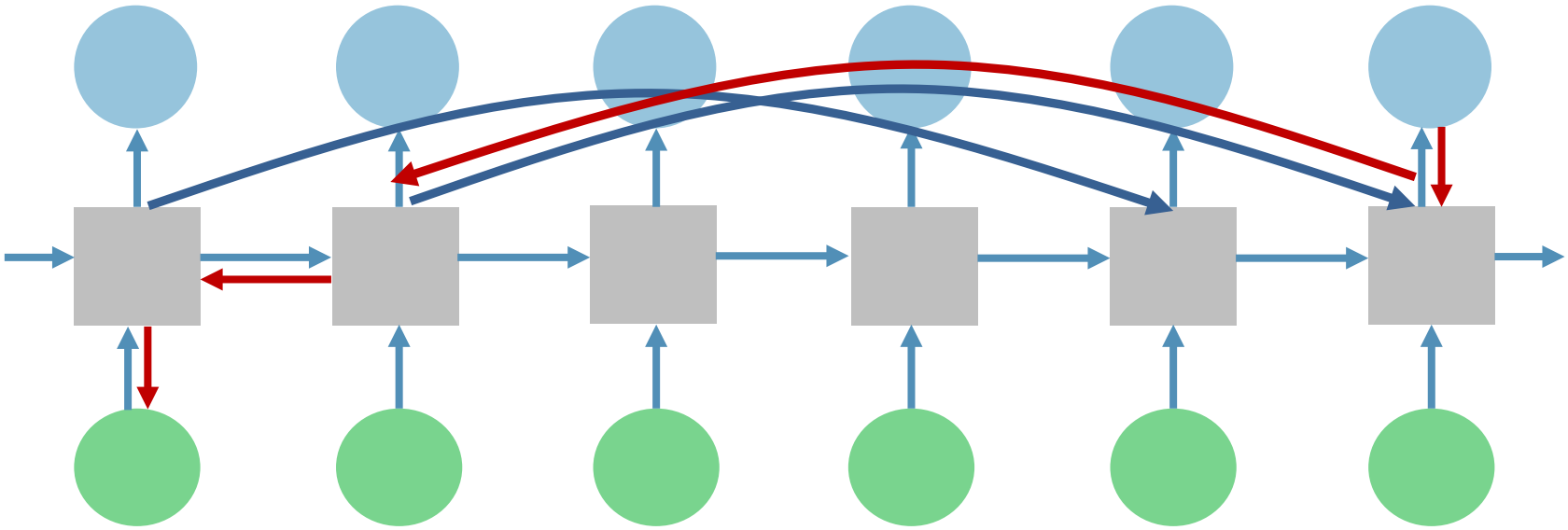
Let’s add shortcuts!

Skip connections



Add shortcuts => shorter ways for the gradients =>
learn longer dependencies

Skip connections



Add shortcuts => shorter ways for the gradients => learn longer dependencies

The idea is similar to the residual connections in the ResNet

Summary

- Exploding gradients are easy to detect but it is not clear how to detect vanishing gradients.
- Exploding gradients: gradient clipping and Truncated BPTT
- Vanishing gradients: ReLU nonlinearity, orthogonal initialisation of the recurrent weights, skip connections.

In the next video:

LSTM and GRU