CSCE 636 Neural Networks (Deep Learning)

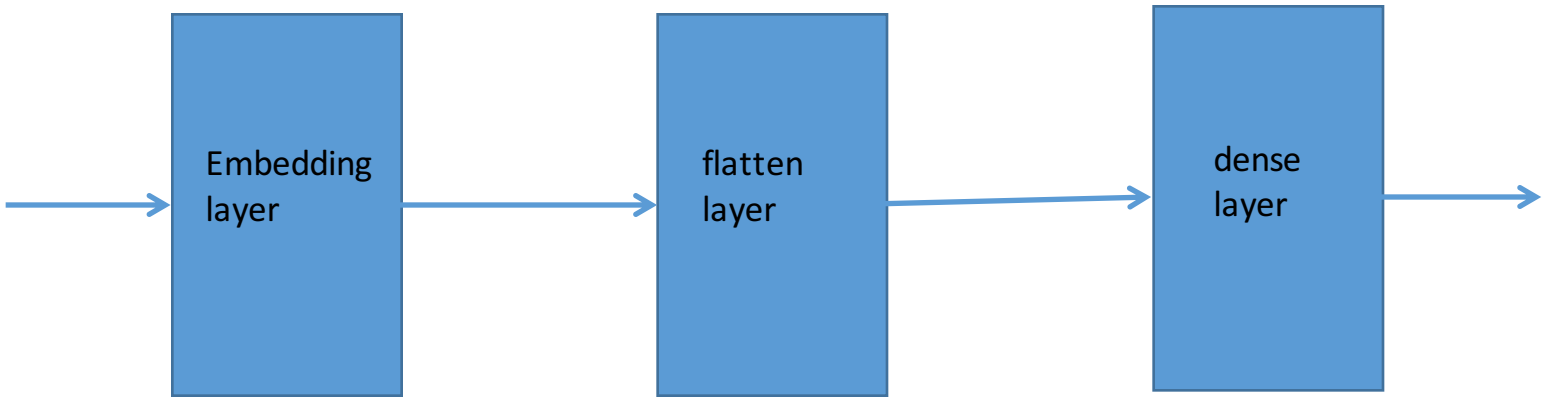
Lecture 9: Deep Learning for Text and Sequences (continued)

Anxiao (Andrew) Jiang

Embedding layer and Recurrent layer

Say we have N = 100 movie reviews as a batch.
Each movie review has maxlen = 20 words.
We build a network with an embedding layer to classify the reviews.

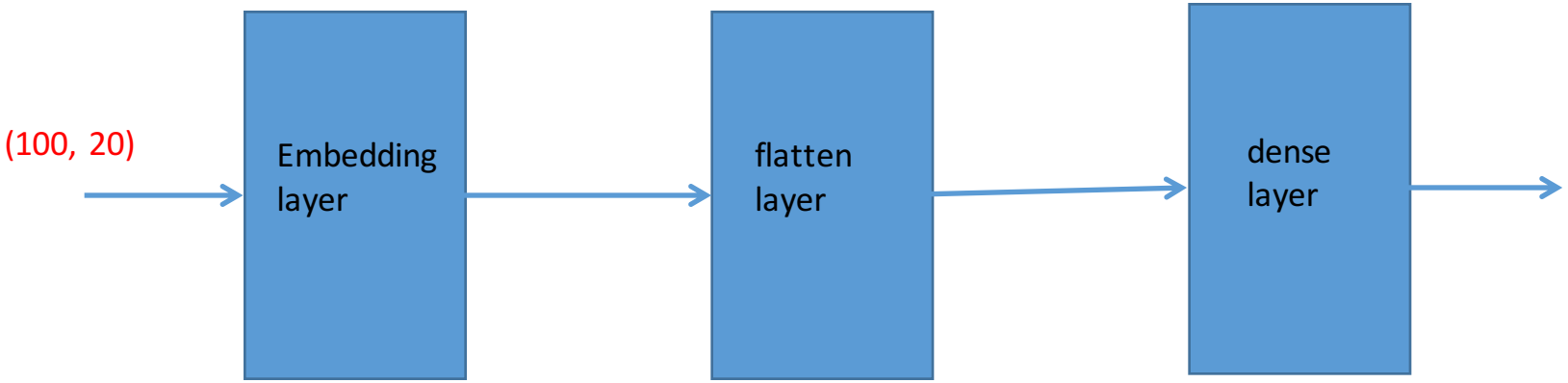
```
model = Sequential()
model.add(Embedding(10000, 8, input_length=maxlen))

model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```



Say we have N = 100 movie reviews as a batch.
Each movie review has maxlen = 20 words.
We build a network with an embedding layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 8, input_length=maxlen))

model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Shape of tensors



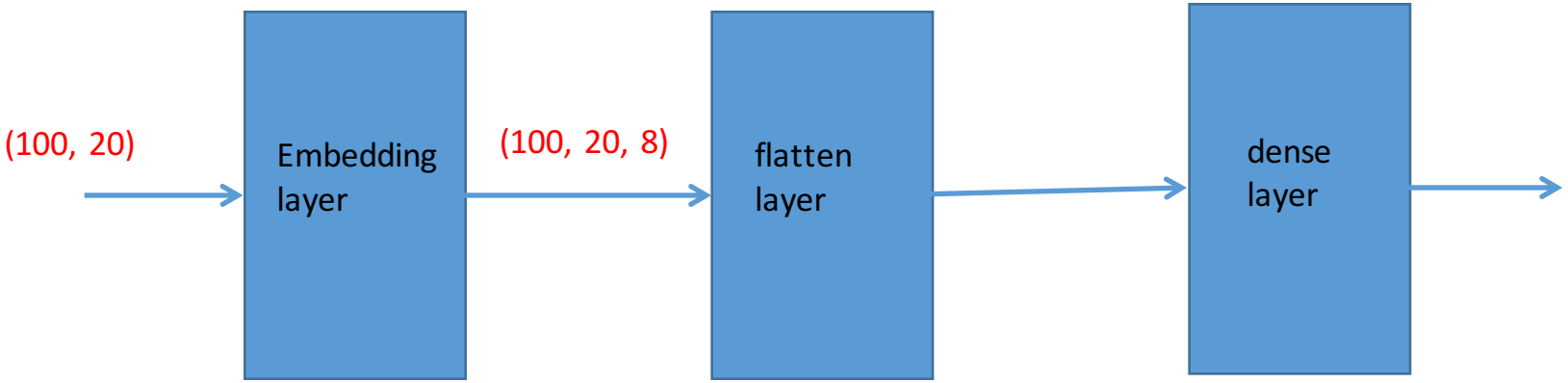
Say we have N = 100 movie reviews as a batch.
Each movie review has maxlen = 20 words.
We build a network with an embedding layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 8, input_length=maxlen))

model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Vocabulary size

Length of embedding vector

Shape of tensors



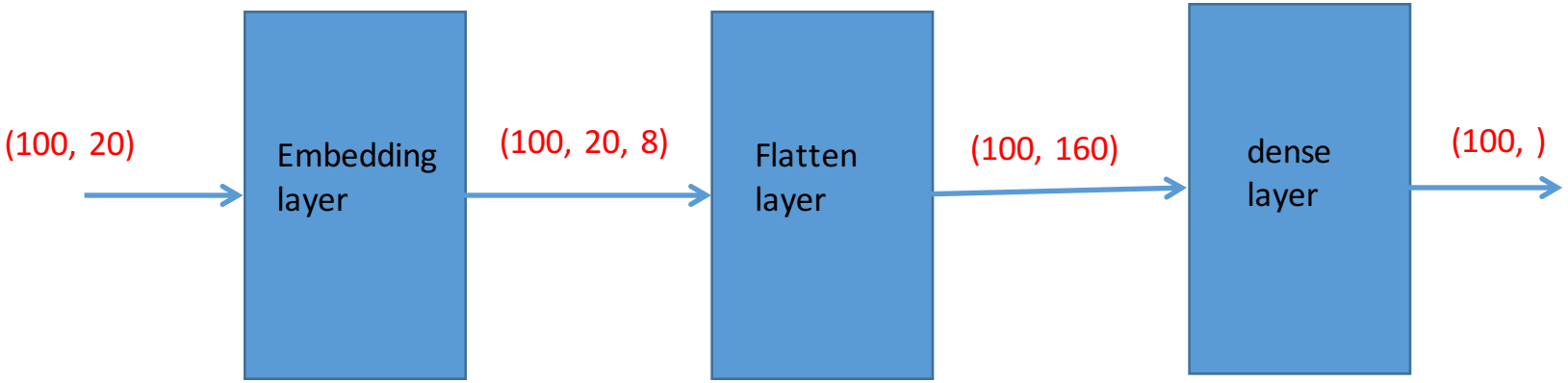
Say we have N = 100 movie reviews as a batch.
Each movie review has maxlen = 20 words.
We build a network with an embedding layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 8, input_length=maxlen))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
```

Vocabulary size

Length of embedding vector

Shape of tensors

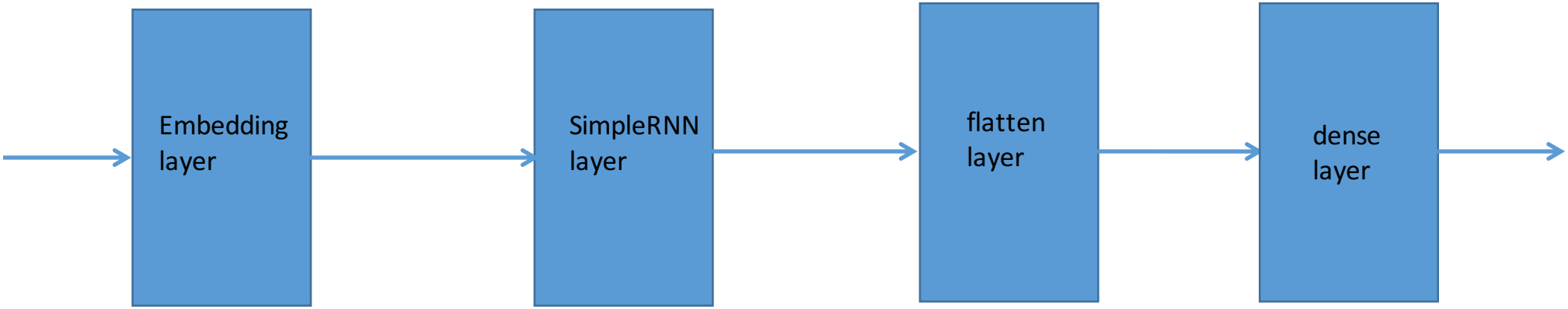


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

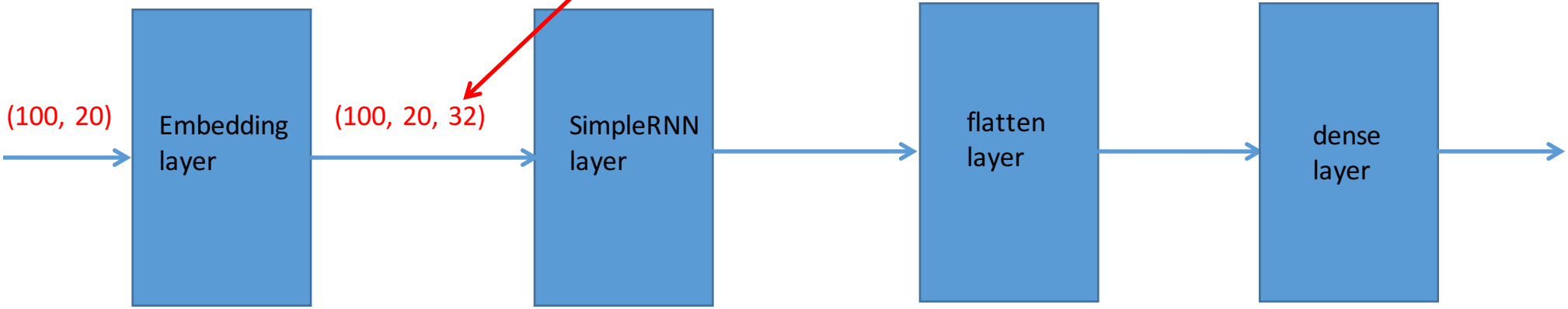
```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

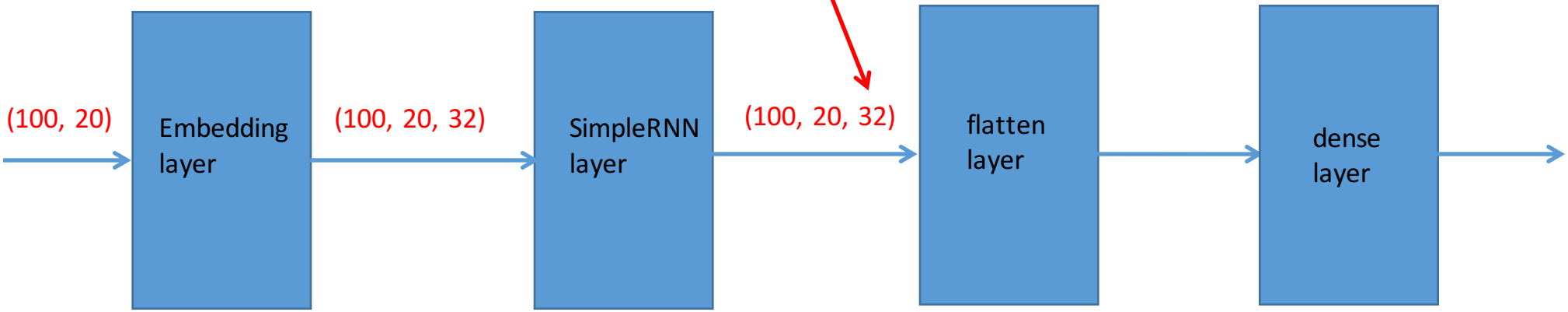
Shape of tensors



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Shape of tensors

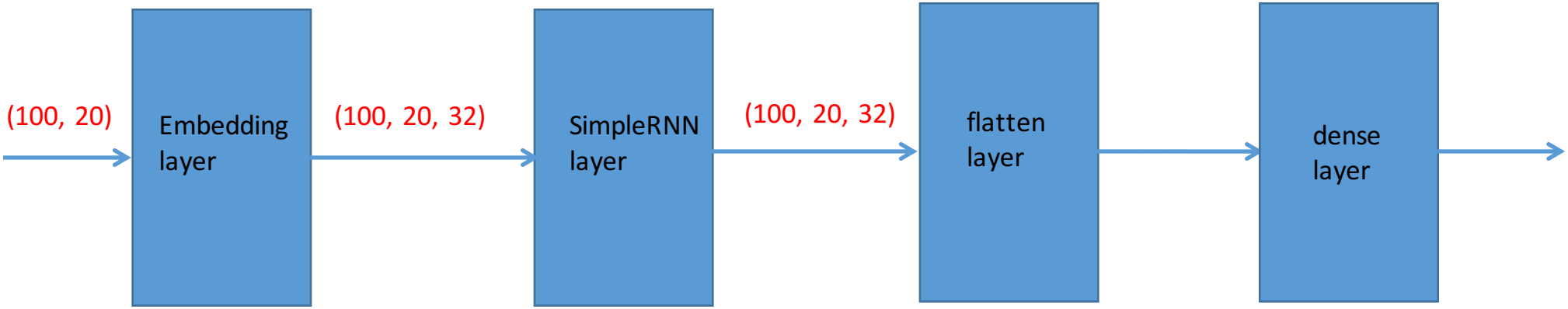


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Return the full sequence of outputs

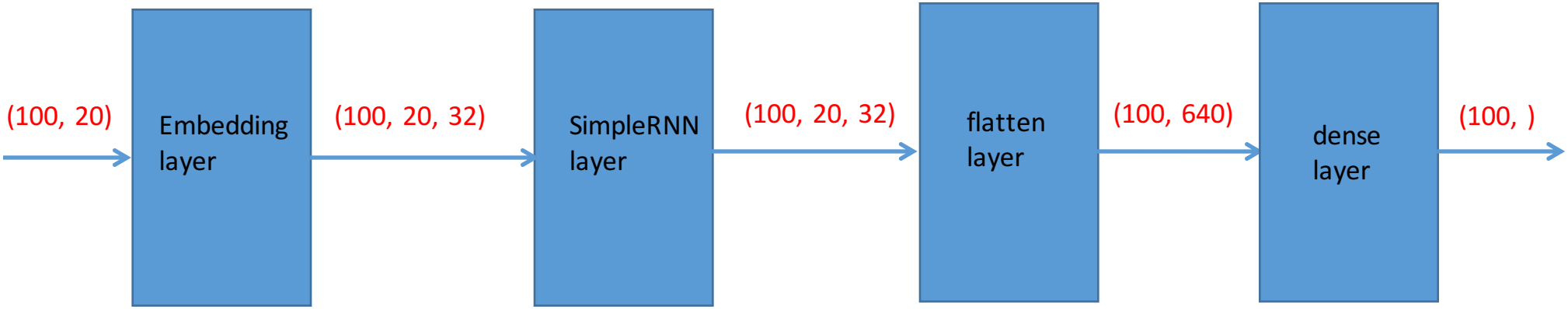
Shape of tensors



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(Flatten())

model.add(Dense(1, activation='sigmoid'))
```

Shape of tensors

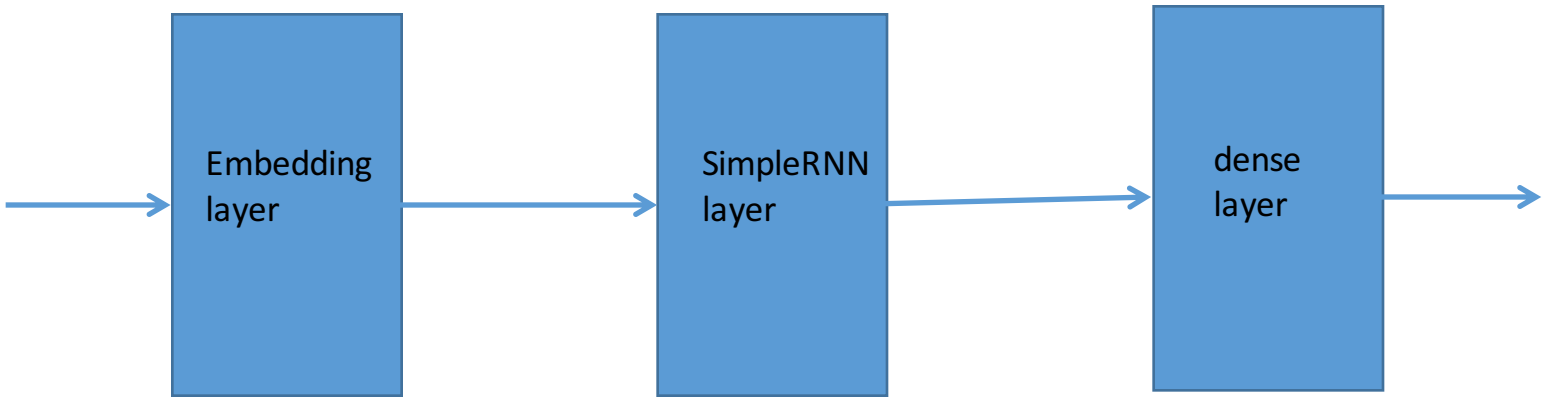


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

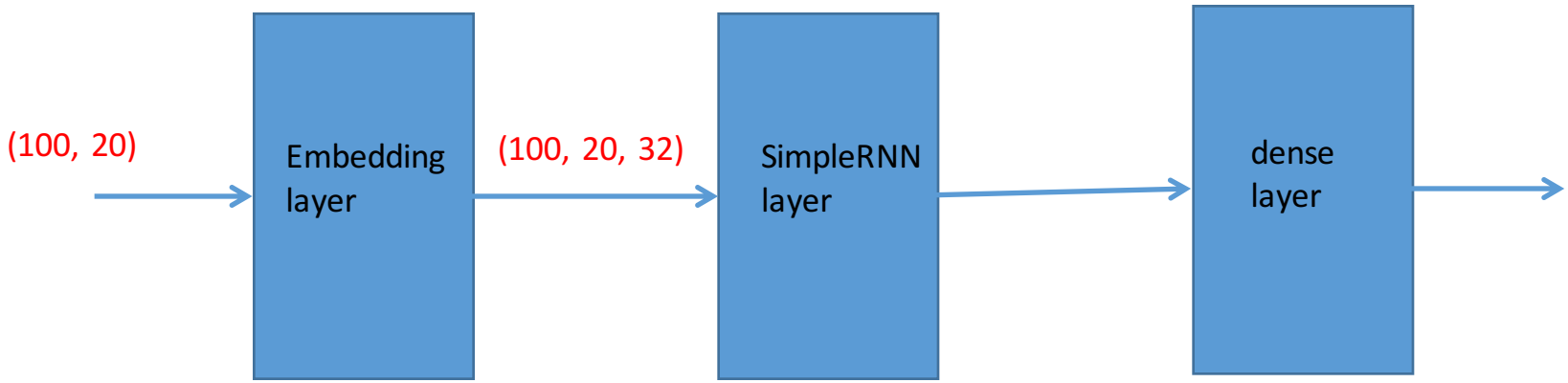
```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

Shape of tensors

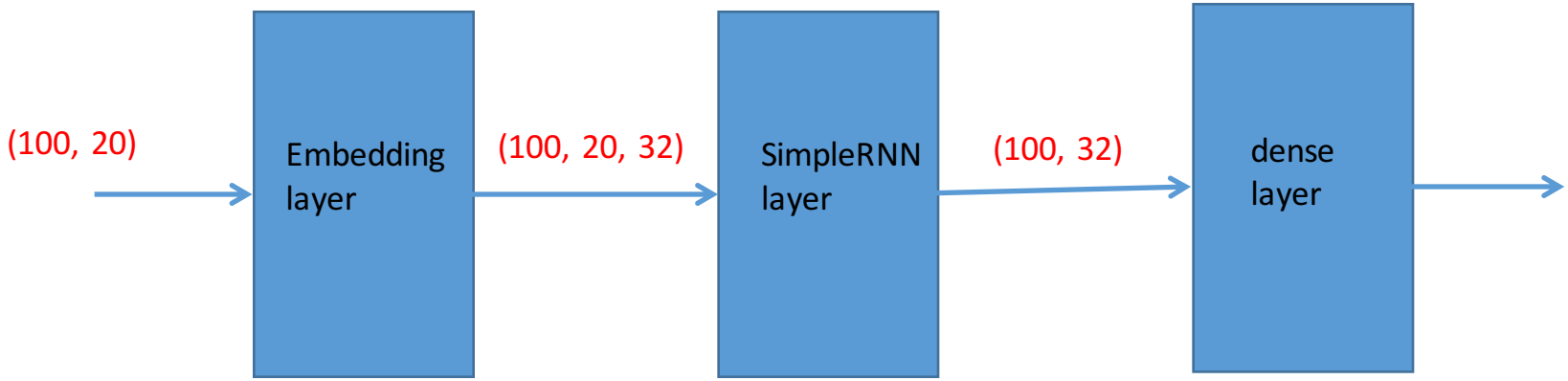


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

The default is to return only the last output of RNN

Shape of tensors

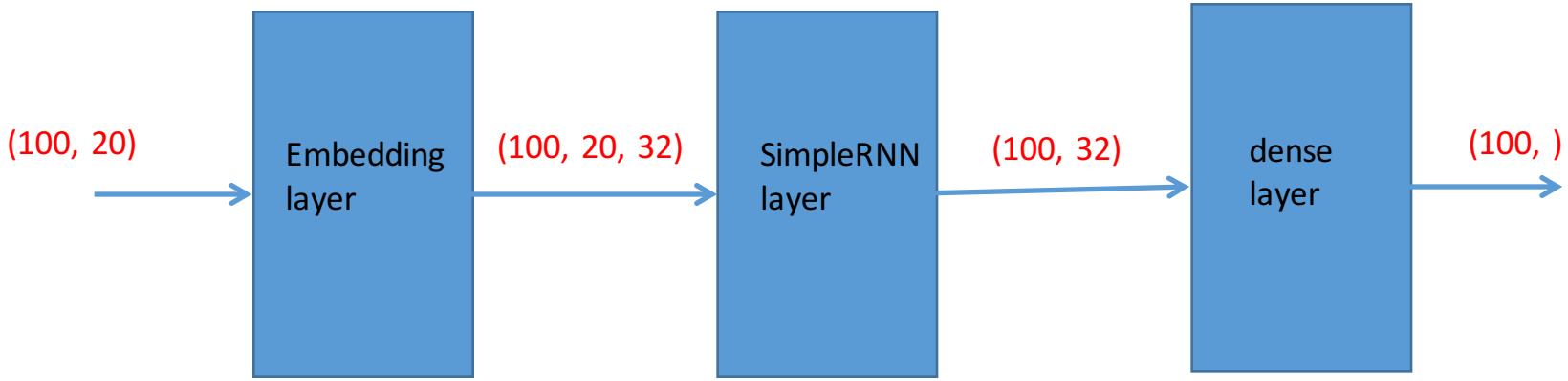


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a SimpleRNN layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

The default is to return only the last output of RNN

Shape of tensors

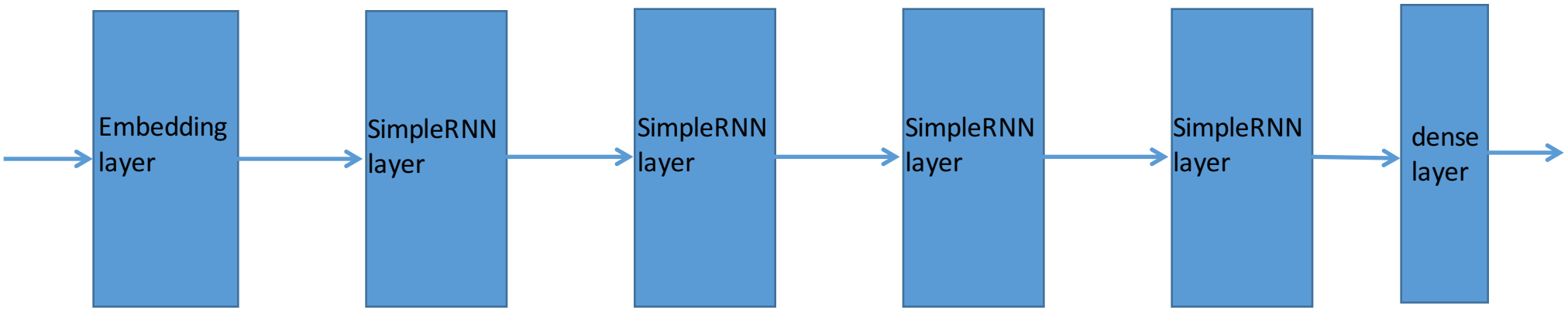


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

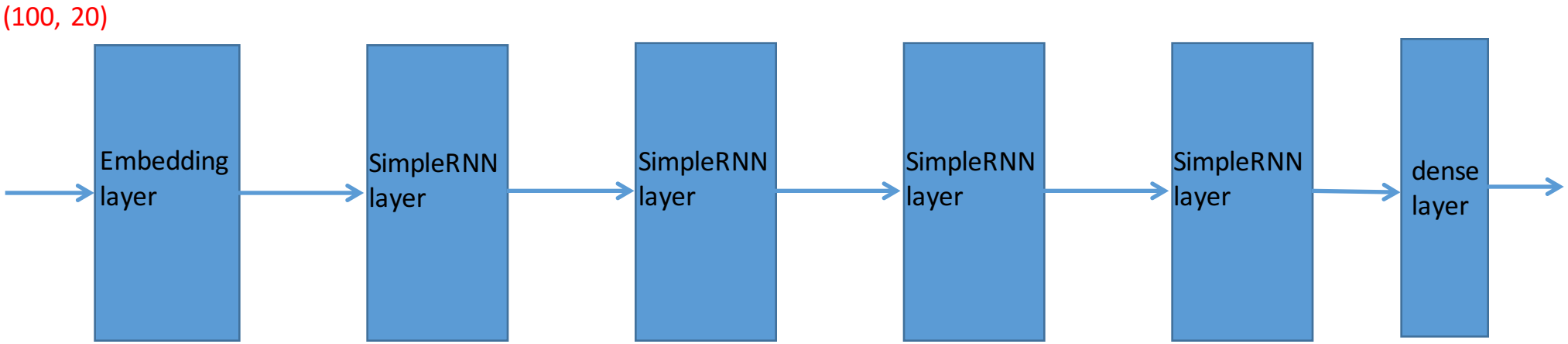
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



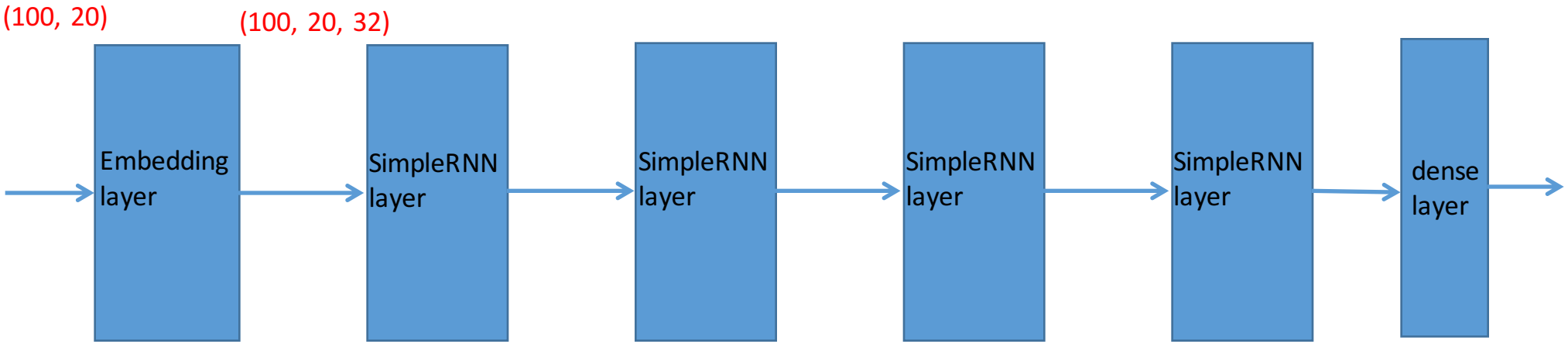
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



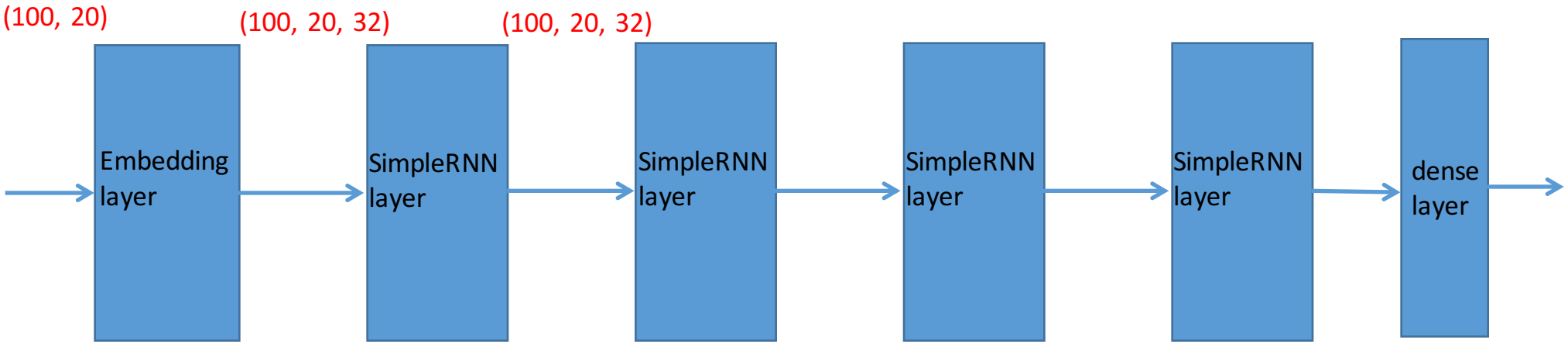
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



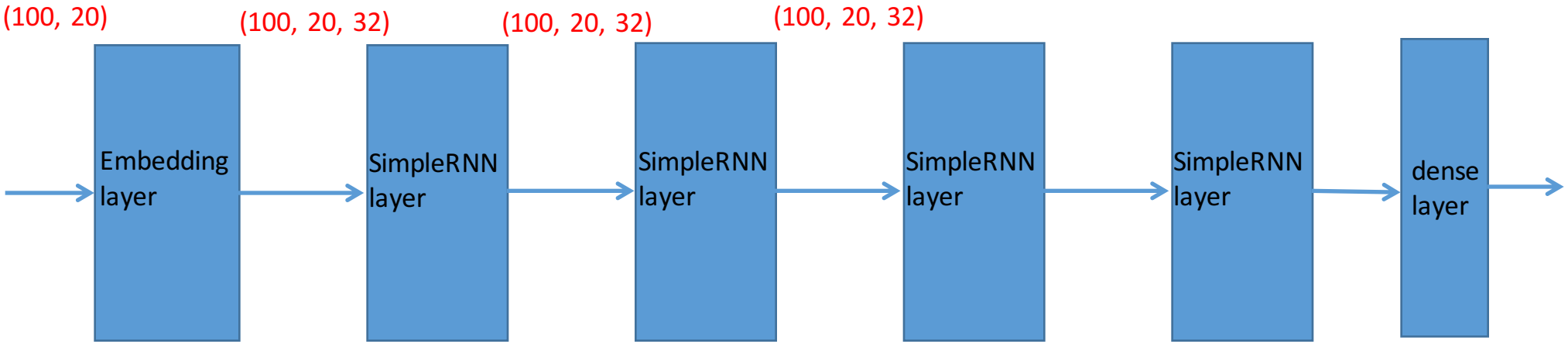
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



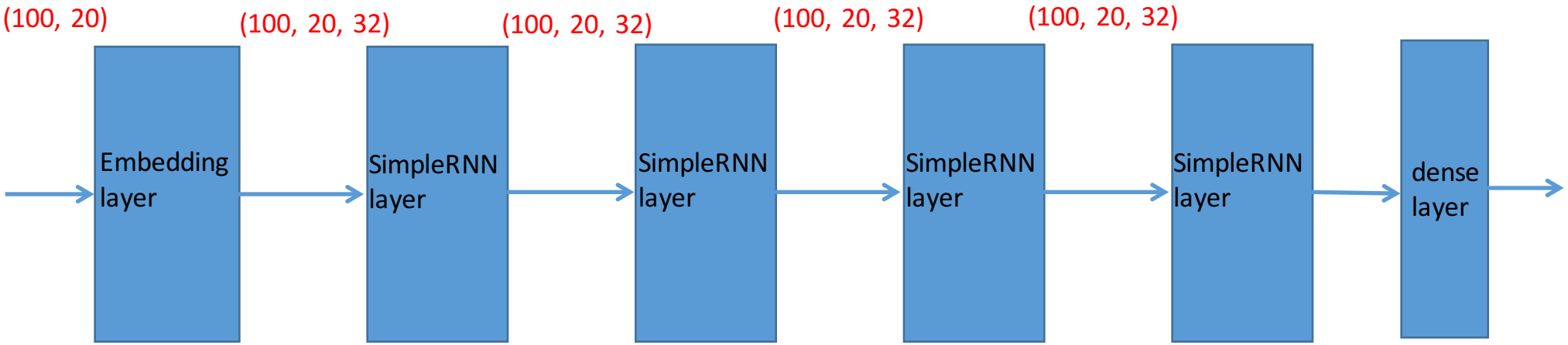
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



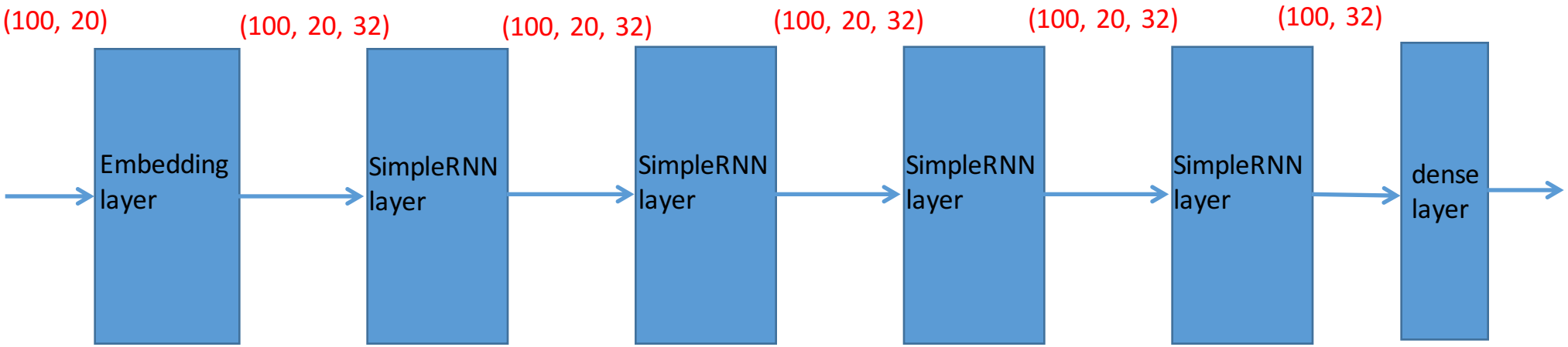
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



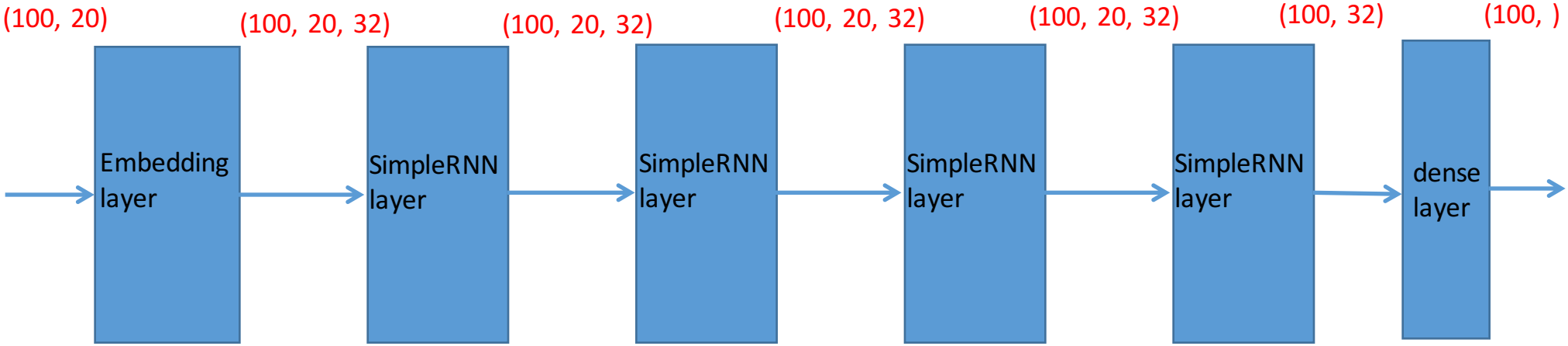
Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words. We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
We stack several RNN layers one after the other to classify the reviews.

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
```

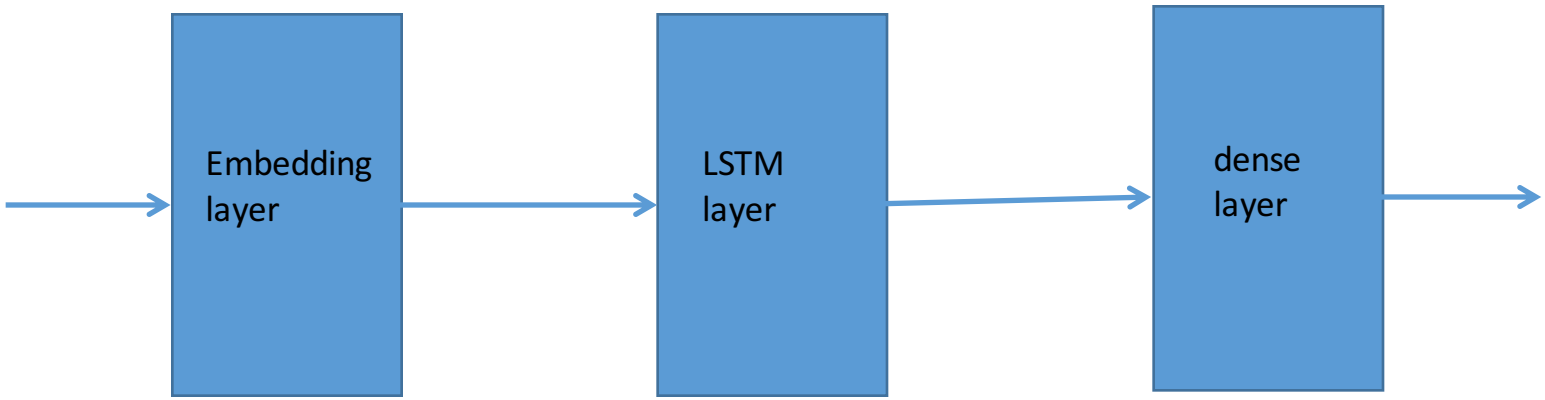


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

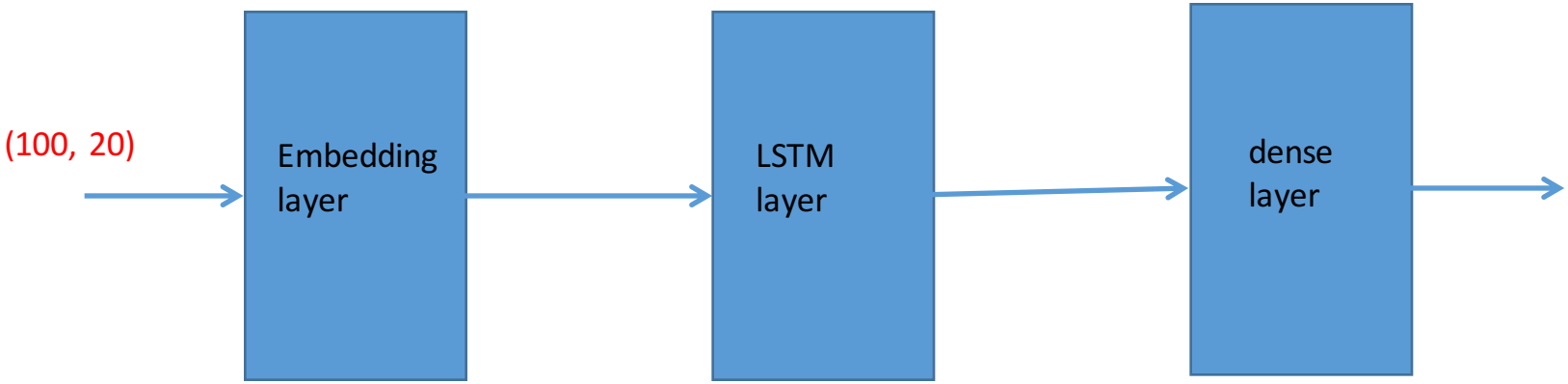
```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

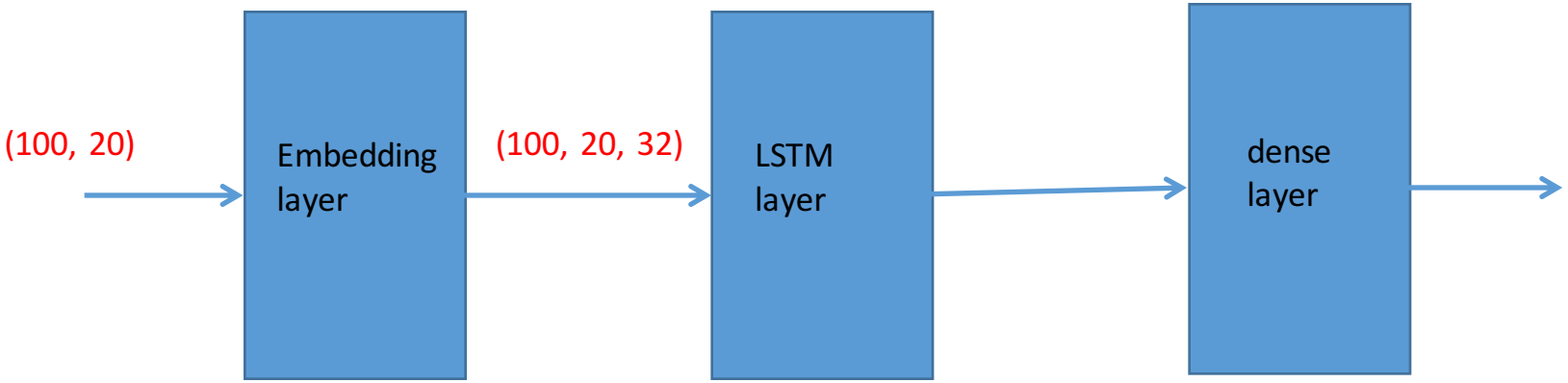
Shape of tensors



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

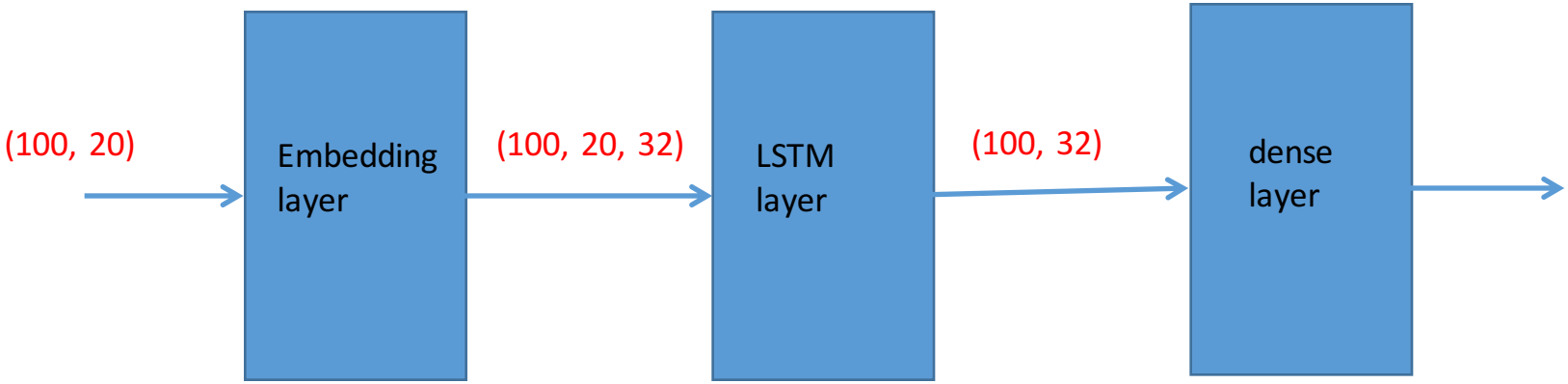
Shape of tensors



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

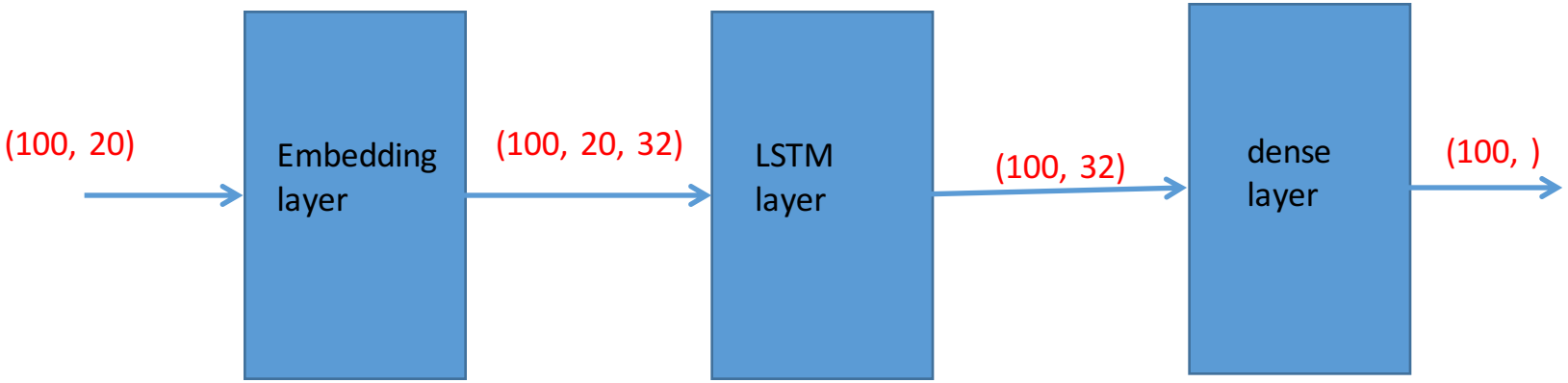
Shape of tensors



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with an embedding layer and a LSTM layer to classify the reviews.

```
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))
```

Shape of tensors

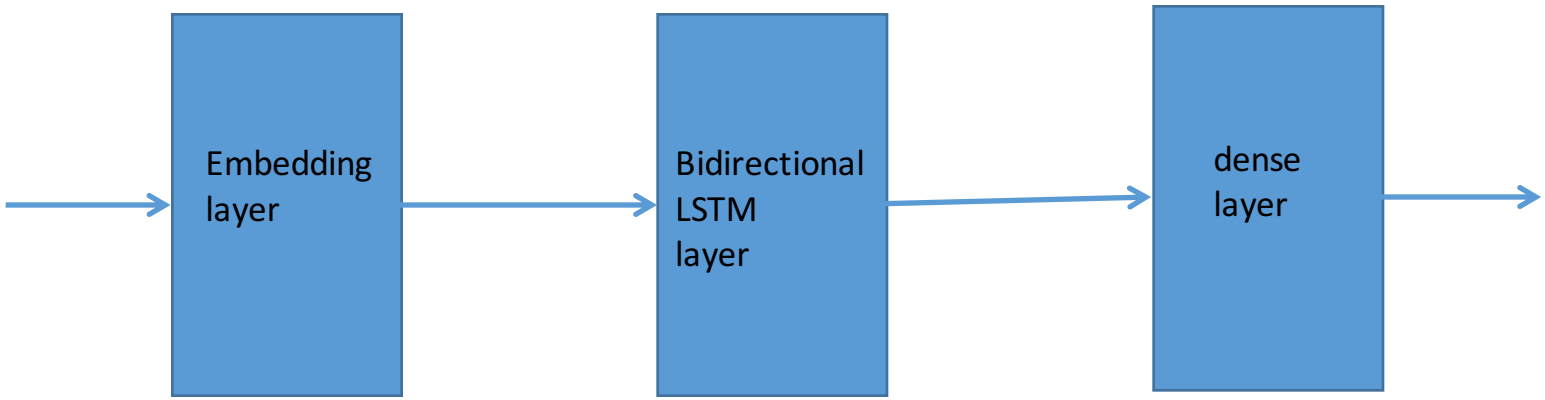


Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with a bidirectional LSTM layer to classify the reviews.

```
model = Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.Bidirectional(layers.LSTM(32)))
model.add(layers.Dense(1, activation='sigmoid'))
```

Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with a bidirectional LSTM layer to classify the reviews.

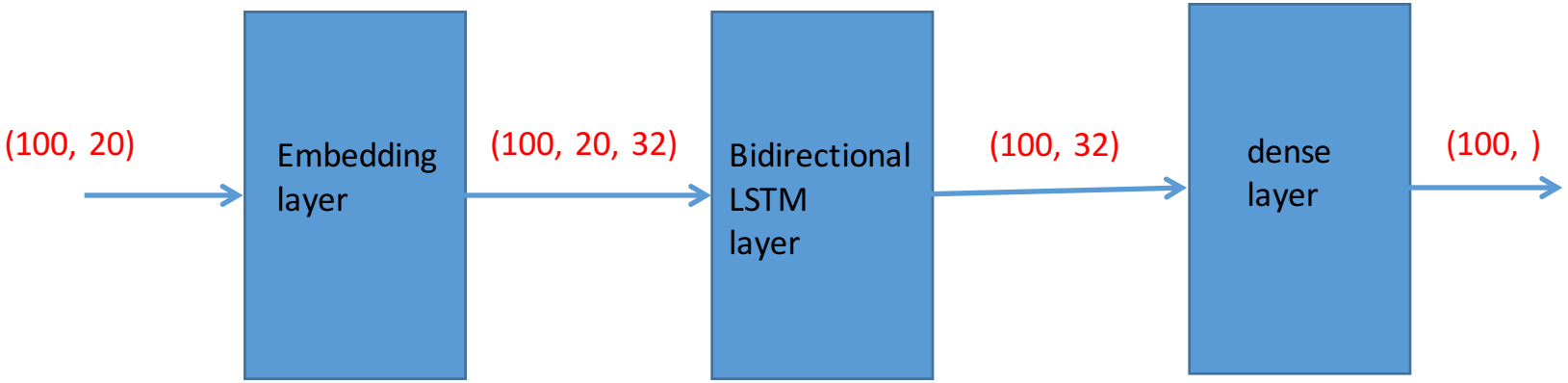
```
model = Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.Bidirectional(layers.LSTM(32)))
model.add(layers.Dense(1, activation='sigmoid'))
```



Say we have N=100 movie reviews as a batch. Each movie review has maxlen = 20 words.
The embedding layer has a vocabulary size of max_features = 1000 words.
We build a network with a bidirectional LSTM layer to classify the reviews.

```
model = Sequential()
model.add(layers.Embedding(max_features, 32))
model.add(layers.Bidirectional(layers.LSTM(32)))
model.add(layers.Dense(1, activation='sigmoid'))
```

Shape of tensors



Quick Summary

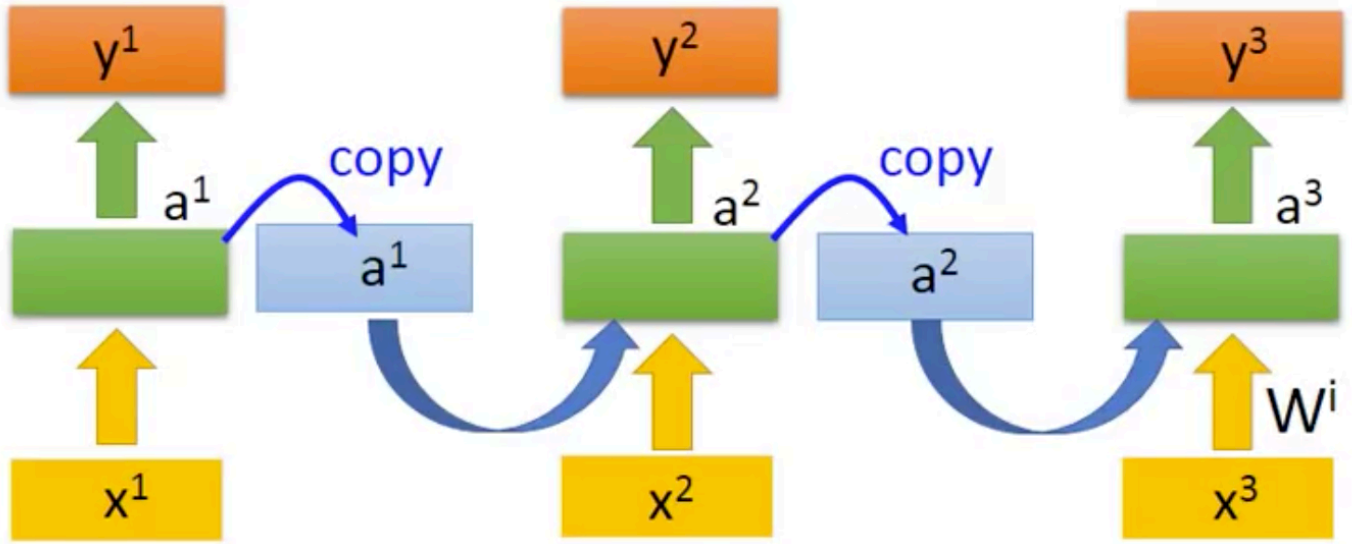
- From the outside, a RNN layer looks just like an ordinary layer.
- Inside the RNN layer, data are processed sequentially.

Why do people prefer LSTM / GRU to SimpleRNN?

The folllowing slides are based on the interesting lecture of Prof. Hung-yi Lee “Recurrent Neural Network”
https://www.youtube.com/watch?v=xCGidAeyS4M&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=30

How to Train an RNN

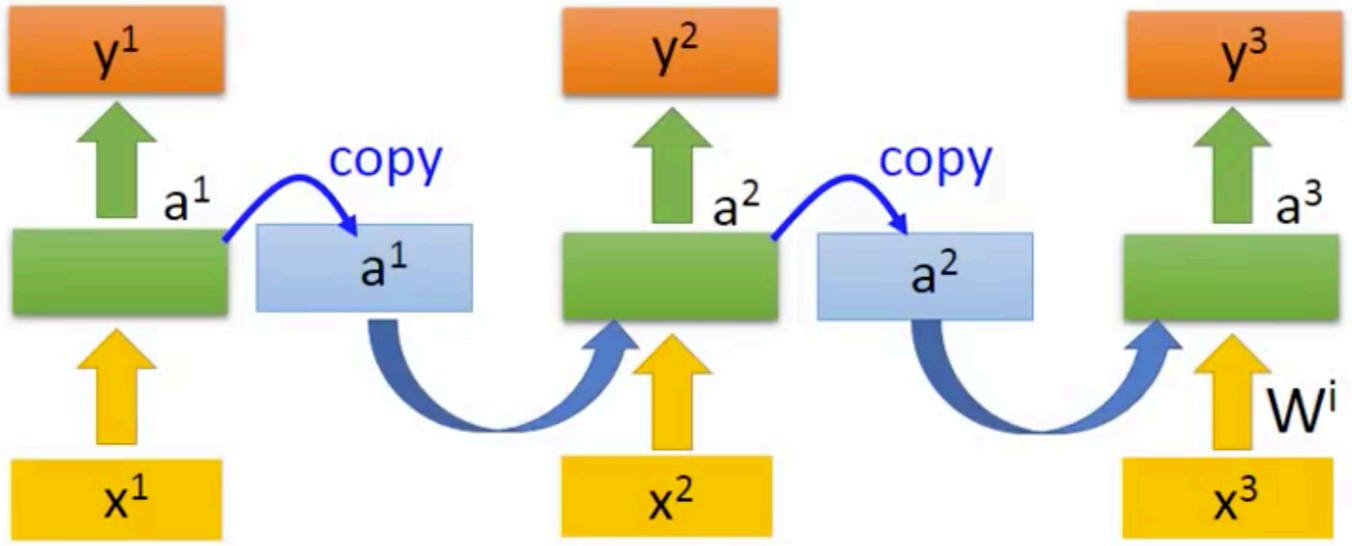
Expand the RNN to all N time steps.
Define a loss function based on the output.



How to Train an RNN

Expand the RNN to all N time steps.
Define a loss function based on the output.

Use Gradient Descent for Training: Back Propagation through time (BPTT)



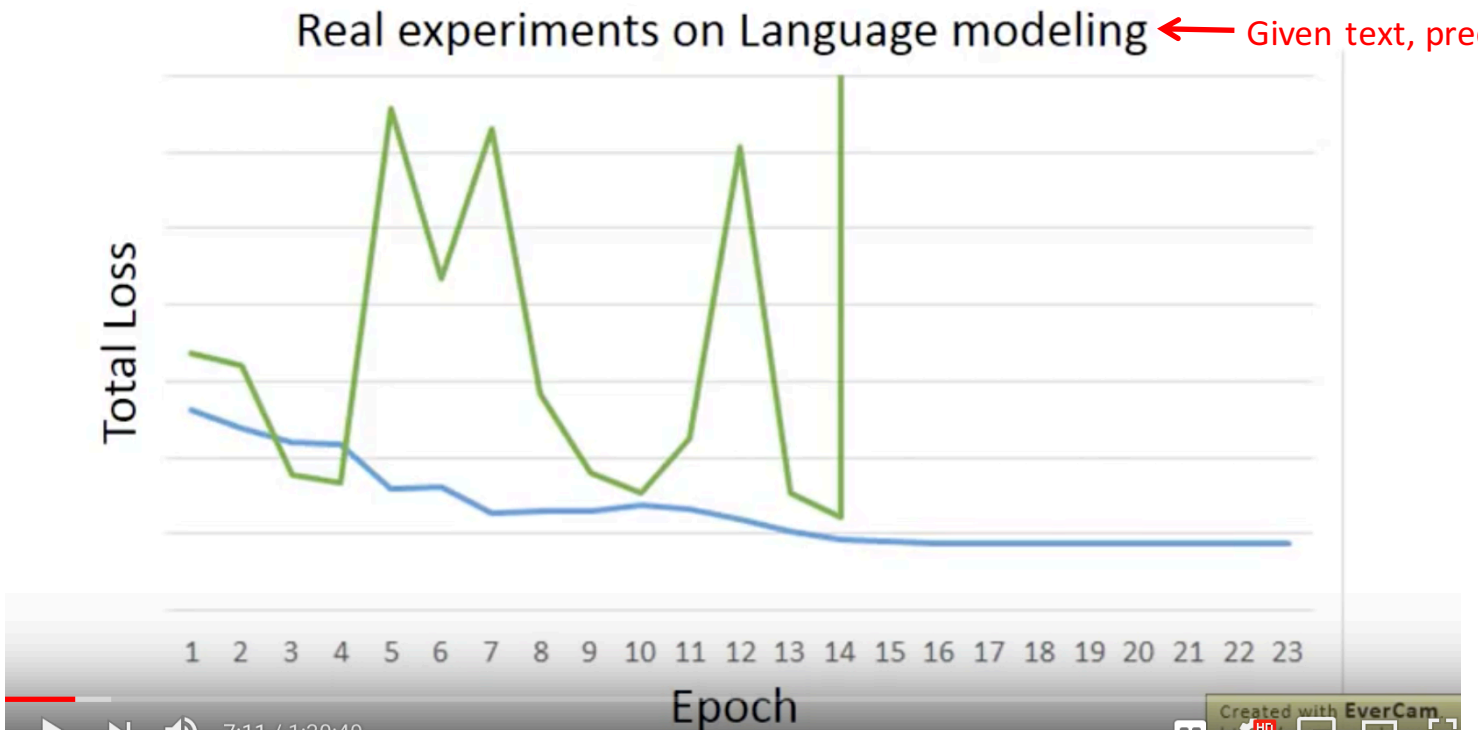
Unfortunately ……

- RNN-based network is not always easy to learn

感謝 曾柏翔 同學
提供實驗結果

Unfortunately ……

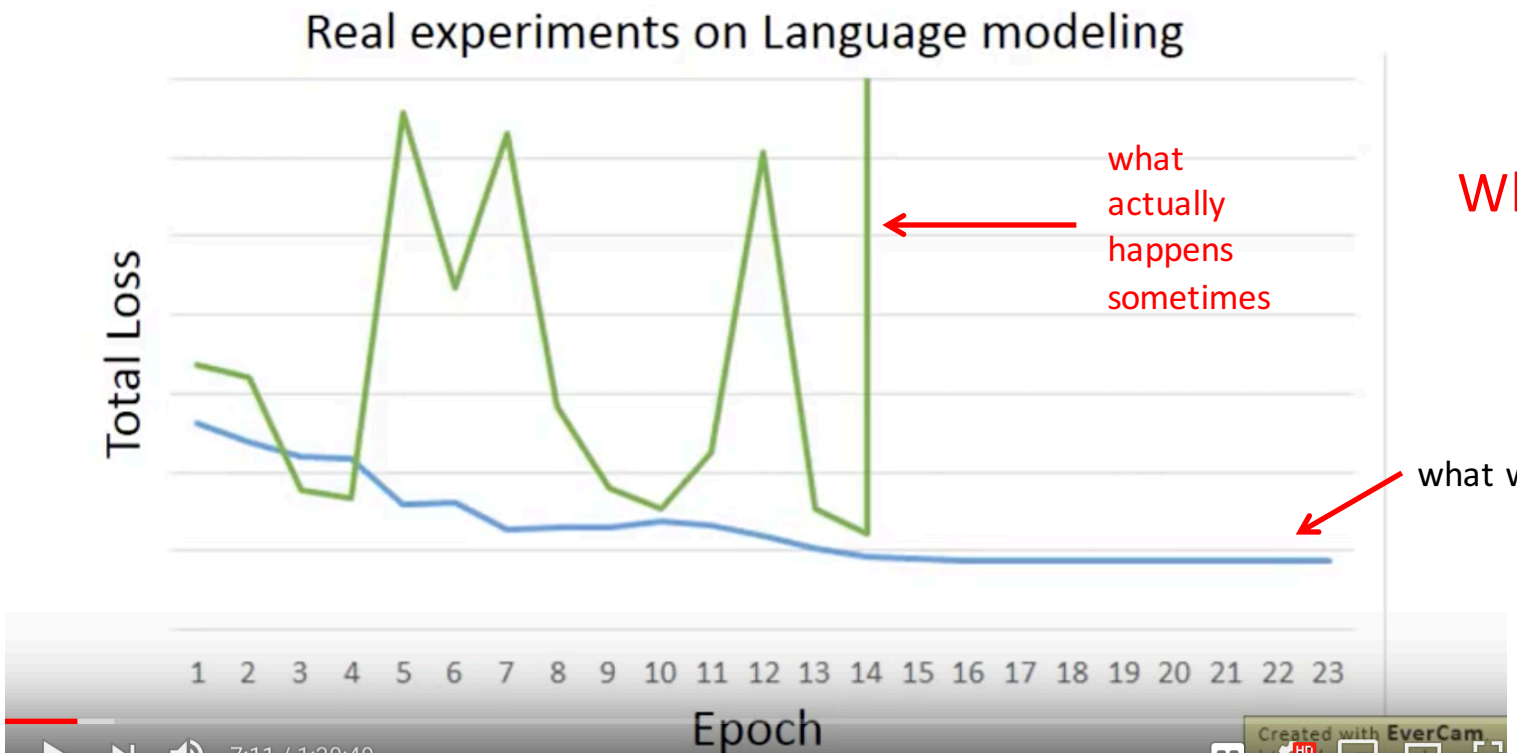
- RNN-based network is not always easy to learn



感謝 曾柏翔 同學
提供實驗結果

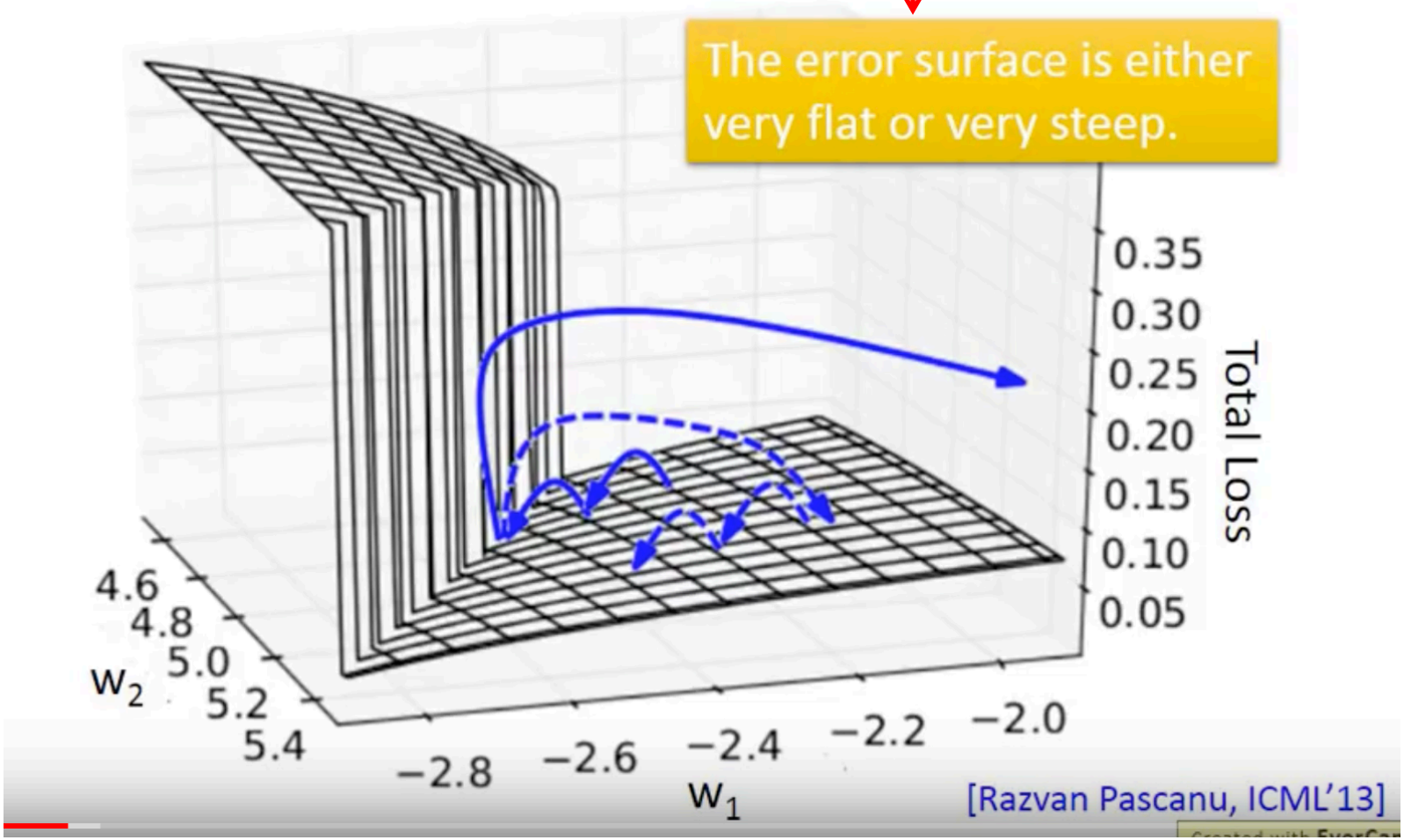
Unfortunately ……

- RNN-based network is not always easy to learn



Why?

The error surface is rough.

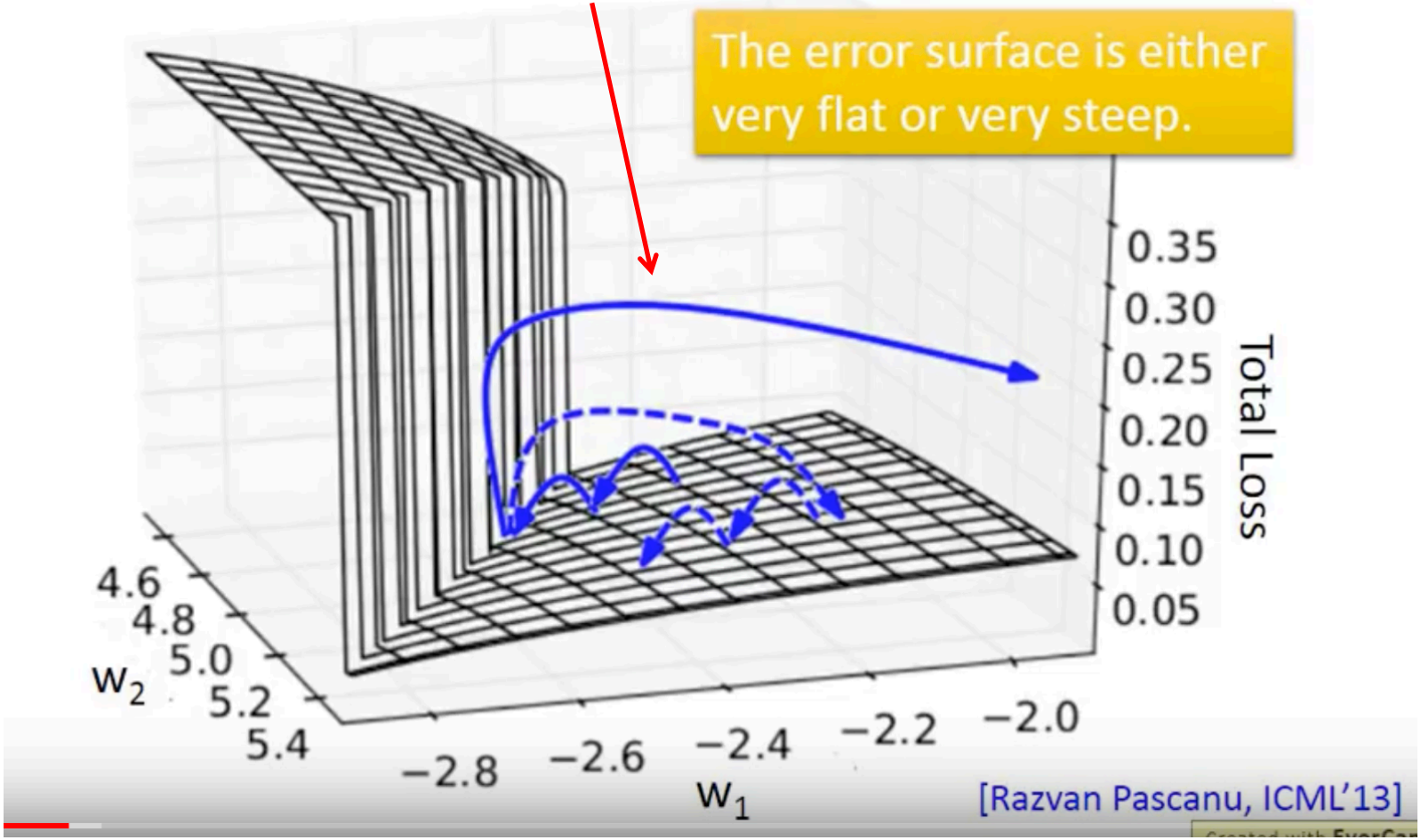


In commonly used gradient descent algorithms,
The learning rate is dynamically adjusted:
When the gradients are small, the algorithm makes the learning rate larger.
When the gradients are large, the algorithm makes the learning rate smaller.

But when the gradients change abruptly,
the algorithm does not know how to adjust the learning rate.
Also, the loss function's value can change abruptly.

The error surface is rough.

Large Learning Rate x Large Gradient ➔ very large change in weights

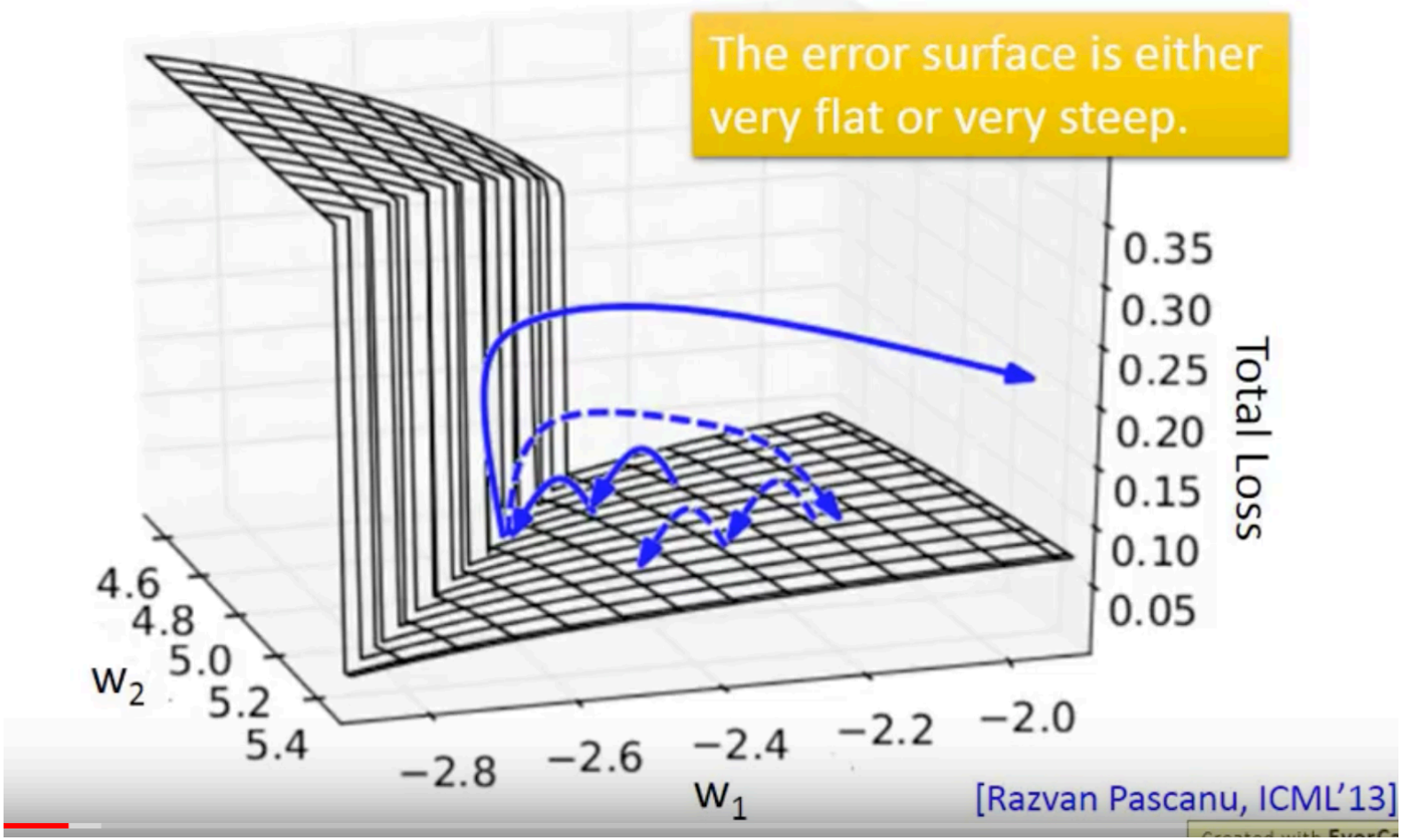


The error surface is rough.

Solution: Gradient Clipping (limit the maximum value for gradient)

Example:

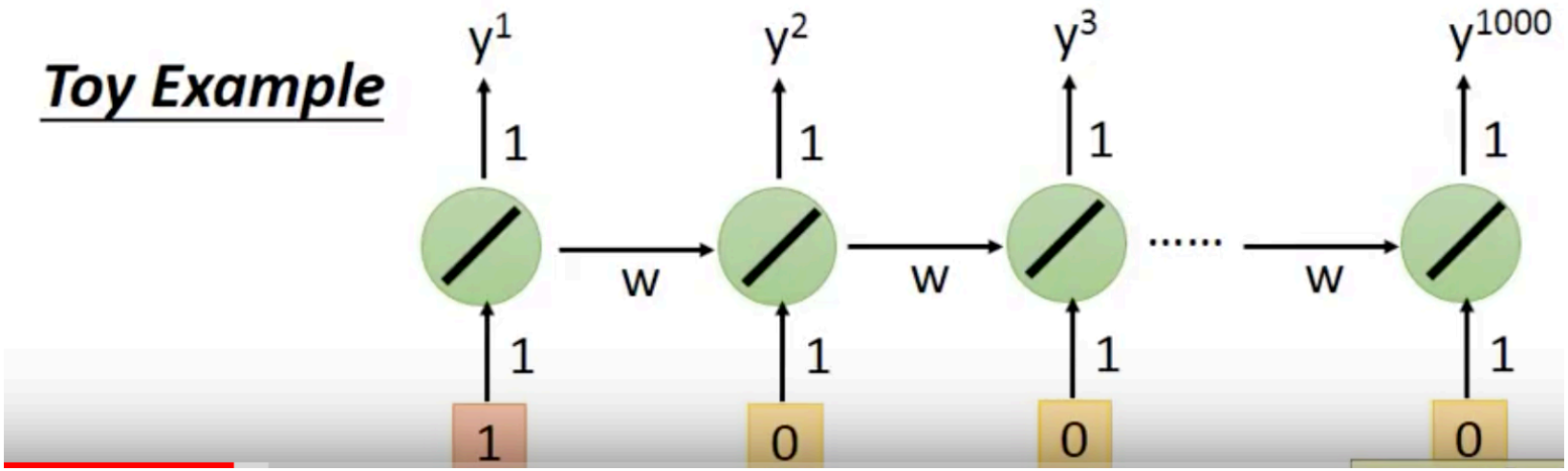
If gradient > 15,
let it be 15.



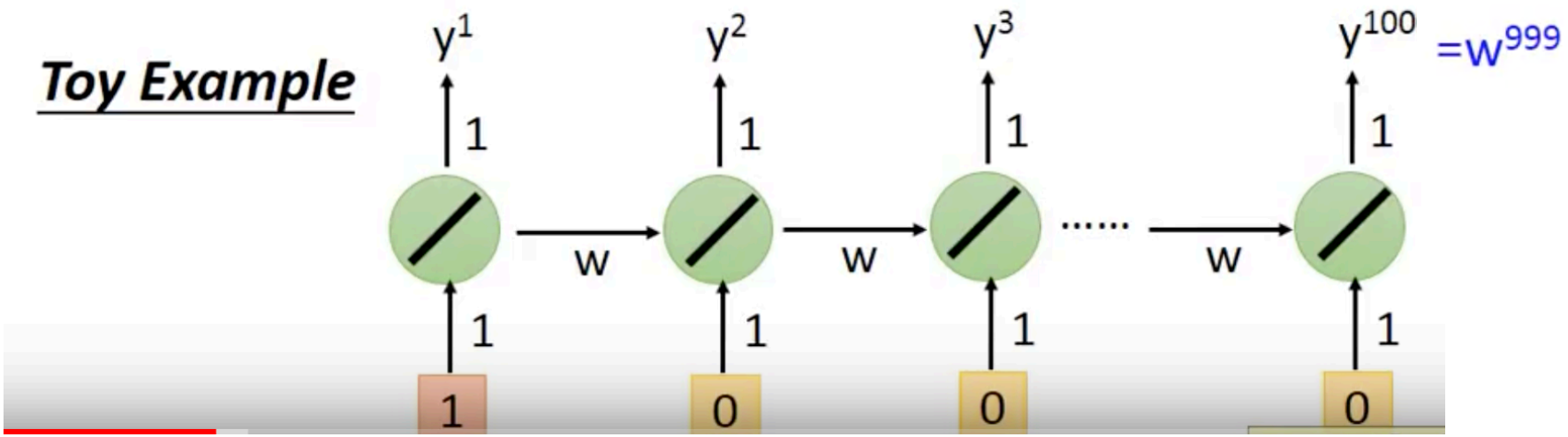
Why does RNN have rough error surfaces
(that is, very small and very large gradients)?

Why?

Consider a very simple RNN: just one neuron,
whose activation function is linear (namely, output = input),
bias = 0,
edge weight = 1 or w.



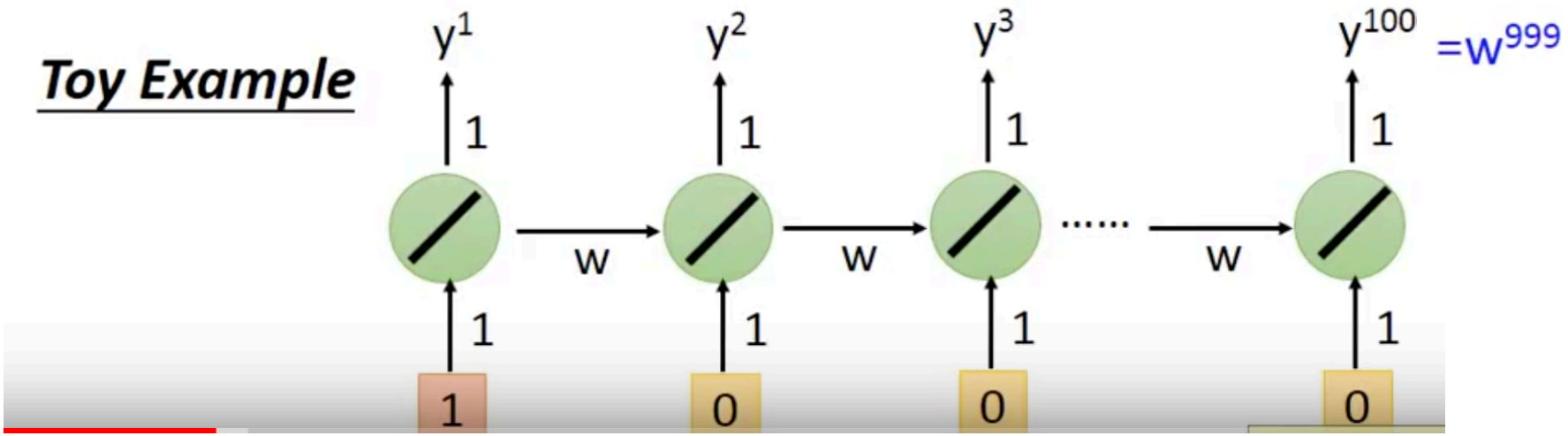
Why?



Why?

$w = 1 \rightarrow y^{1000} = 1$

$w = 1.01 \rightarrow y^{1000} \approx 20000$



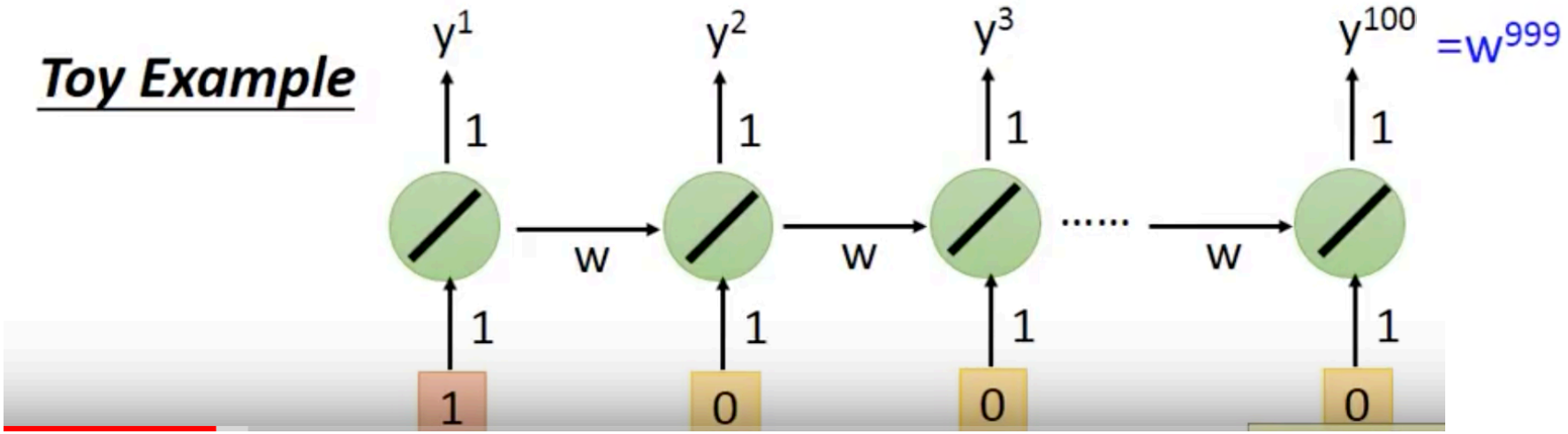
Why?

$w = 1 \rightarrow y^{1000} = 1$

$w = 1.01 \rightarrow y^{1000} \approx 20000$

Large $\partial L / \partial w$ → Small Learning rate?

Toy Example



Why?

$w = 1 \rightarrow y^{1000} = 1$

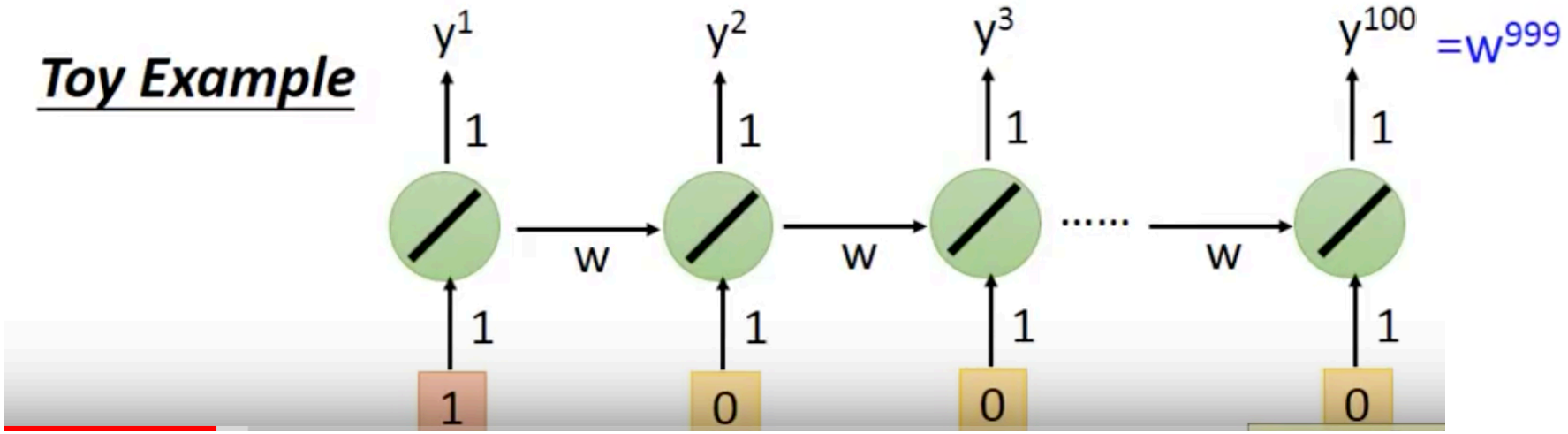
$w = 1.01 \rightarrow y^{1000} \approx 20000$

Large $\partial L / \partial w$ → Small Learning rate?

$w = 0.99 \rightarrow y^{1000} \approx 0$

$w = 0.01 \rightarrow y^{1000} \approx 0$

Toy Example



Why?

$w = 1 \rightarrow y^{1000} = 1$

$w = 1.01 \rightarrow y^{1000} \approx 20000$

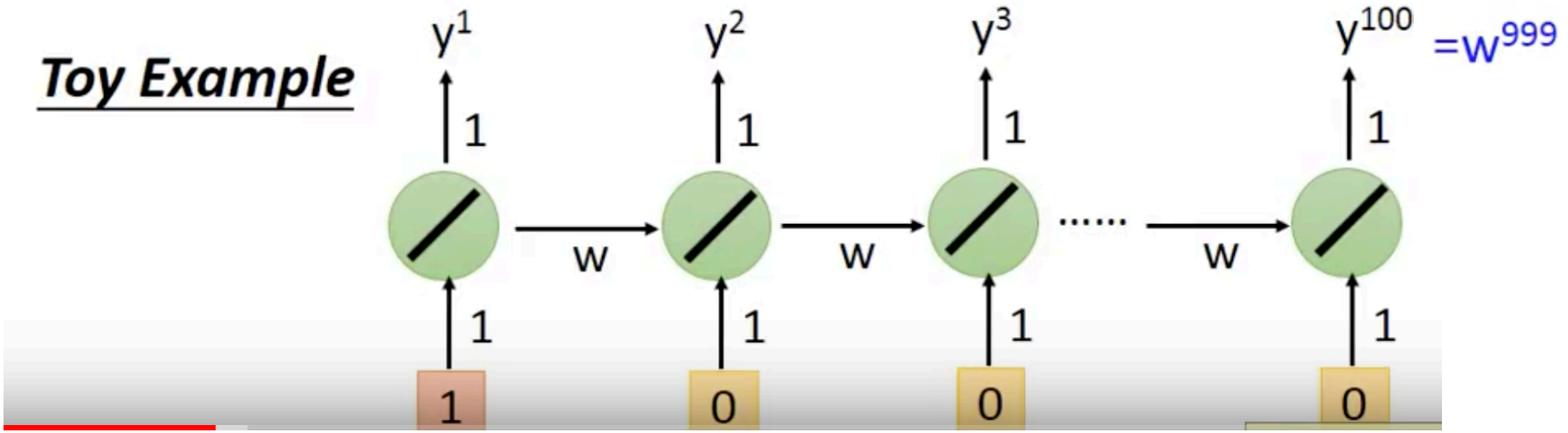
Large $\partial L/\partial w$ → Small Learning rate?

$w = 0.99 \rightarrow y^{1000} \approx 0$

$w = 0.01 \rightarrow y^{1000} \approx 0$

small $\partial L/\partial w$ → Large Learning rate?

Toy Example



Why?

Exploding Gradient

$w = 1 \rightarrow y^{1000} = 1$

$w = 1.01 \rightarrow y^{1000} \approx 20000$

Large $\partial L/\partial w$ → Small Learning rate?

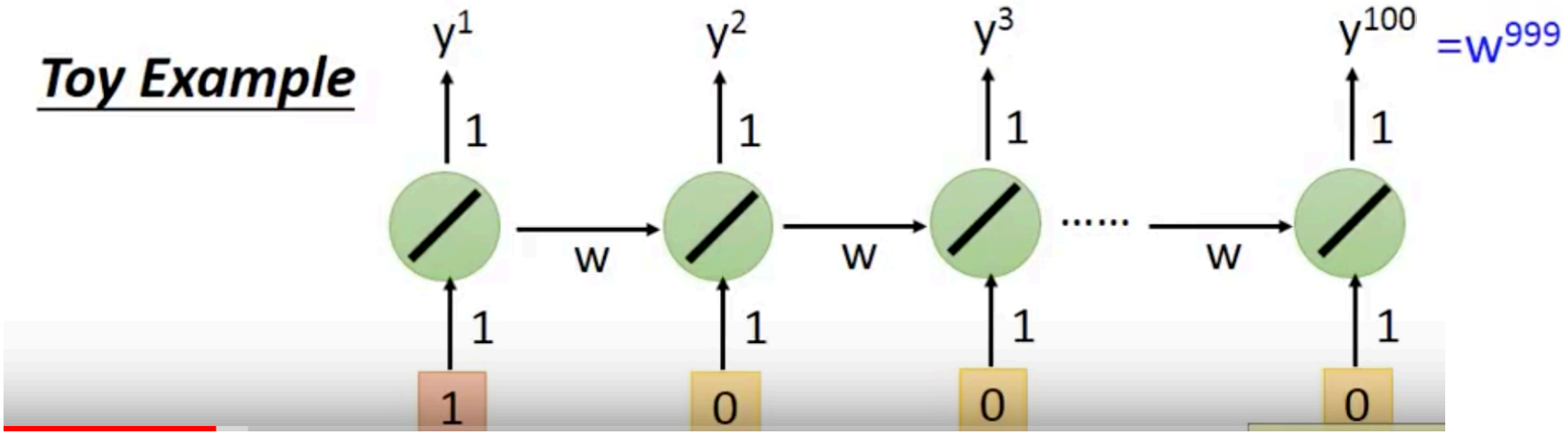
$w = 0.99 \rightarrow y^{1000} \approx 0$

$w = 0.01 \rightarrow y^{1000} \approx 0$

small $\partial L/\partial w$ → Large Learning rate?

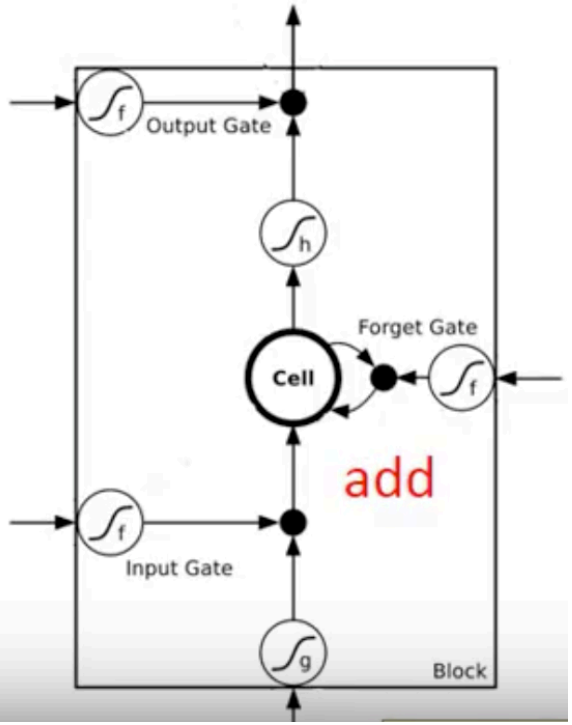
Vanishing Gradient

Toy Example



Helpful Techniques

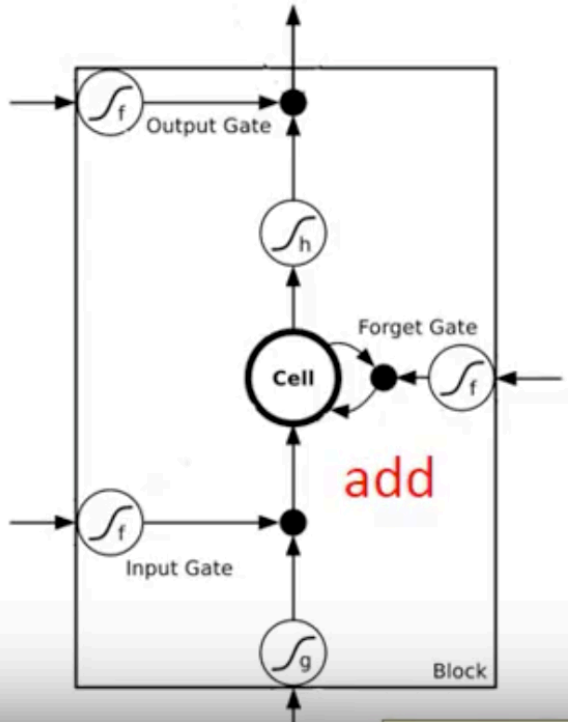
- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)



Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

Train LSTM with very small learning rates



Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)

Gated Recurrent Unit (GRU):
simpler than LSTM.

LSTM: 3 gates. GRU: 2 gates.

Intuition of GRU: control input gate and forget gate together.
When input gate is open, forget gate is closed and memory is erased.
When input gate is closes, forget gate is open and memory is kept.