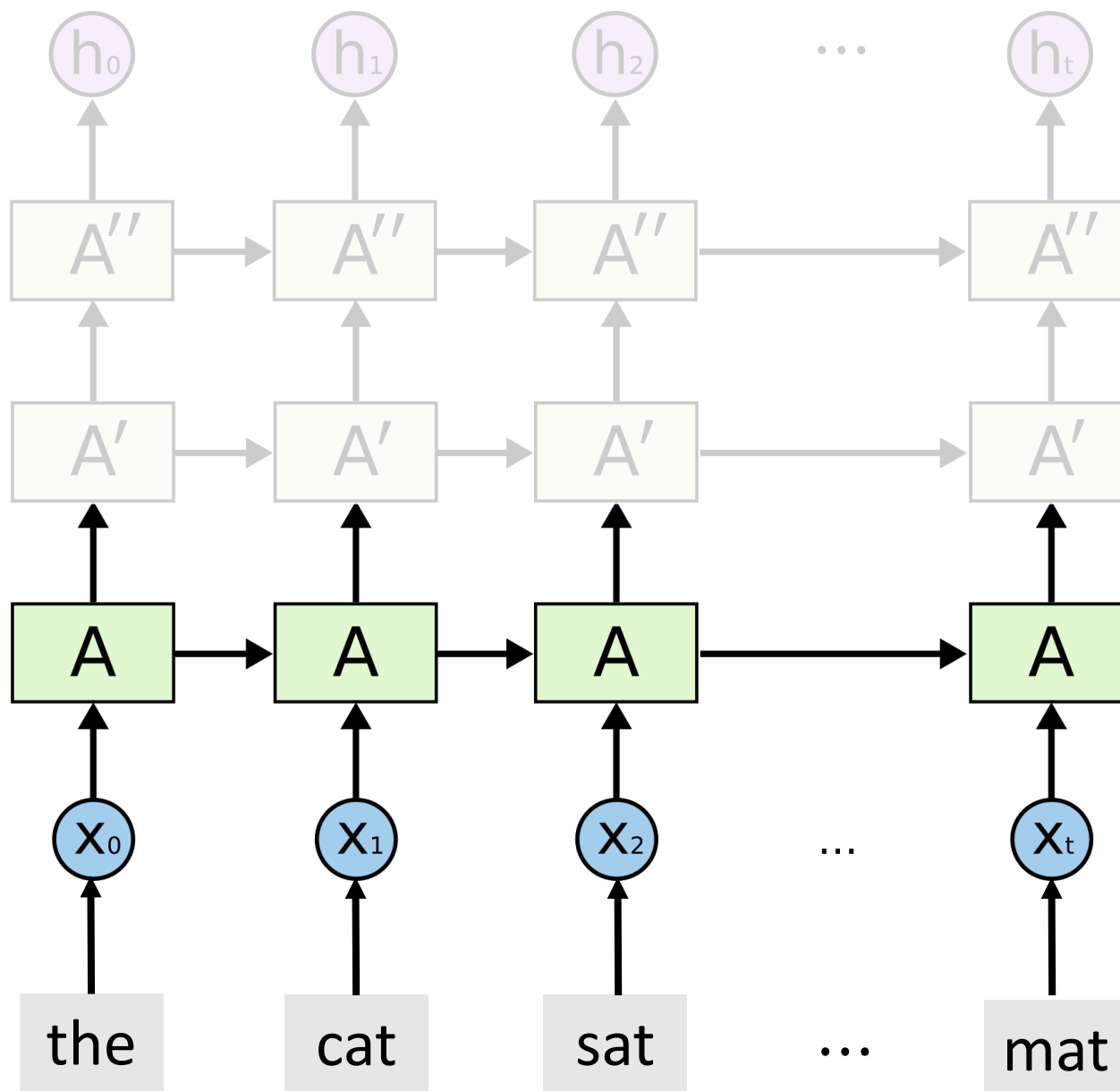


Making RNNs More Effective

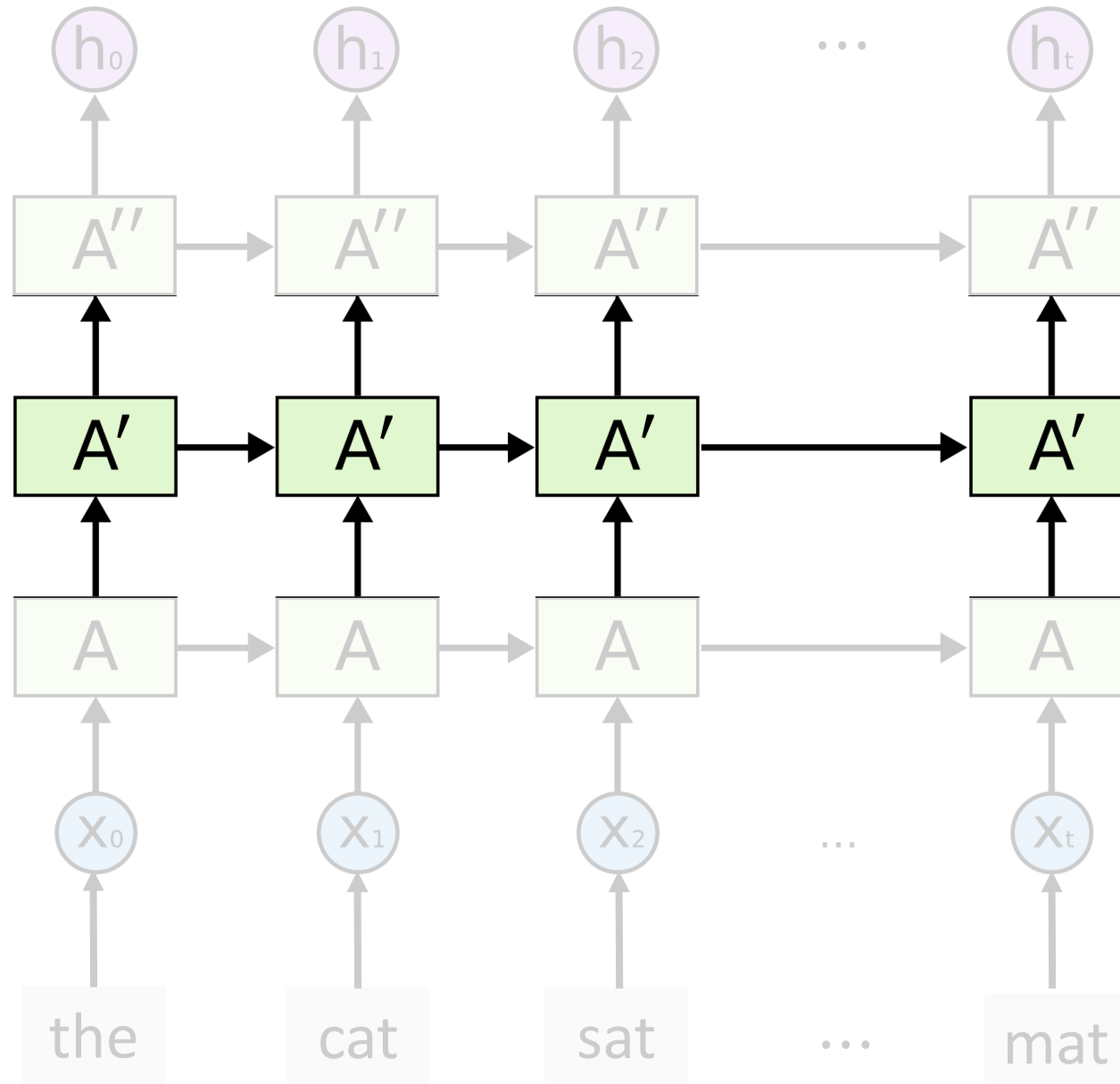
Shusen Wang

Stacked RNN

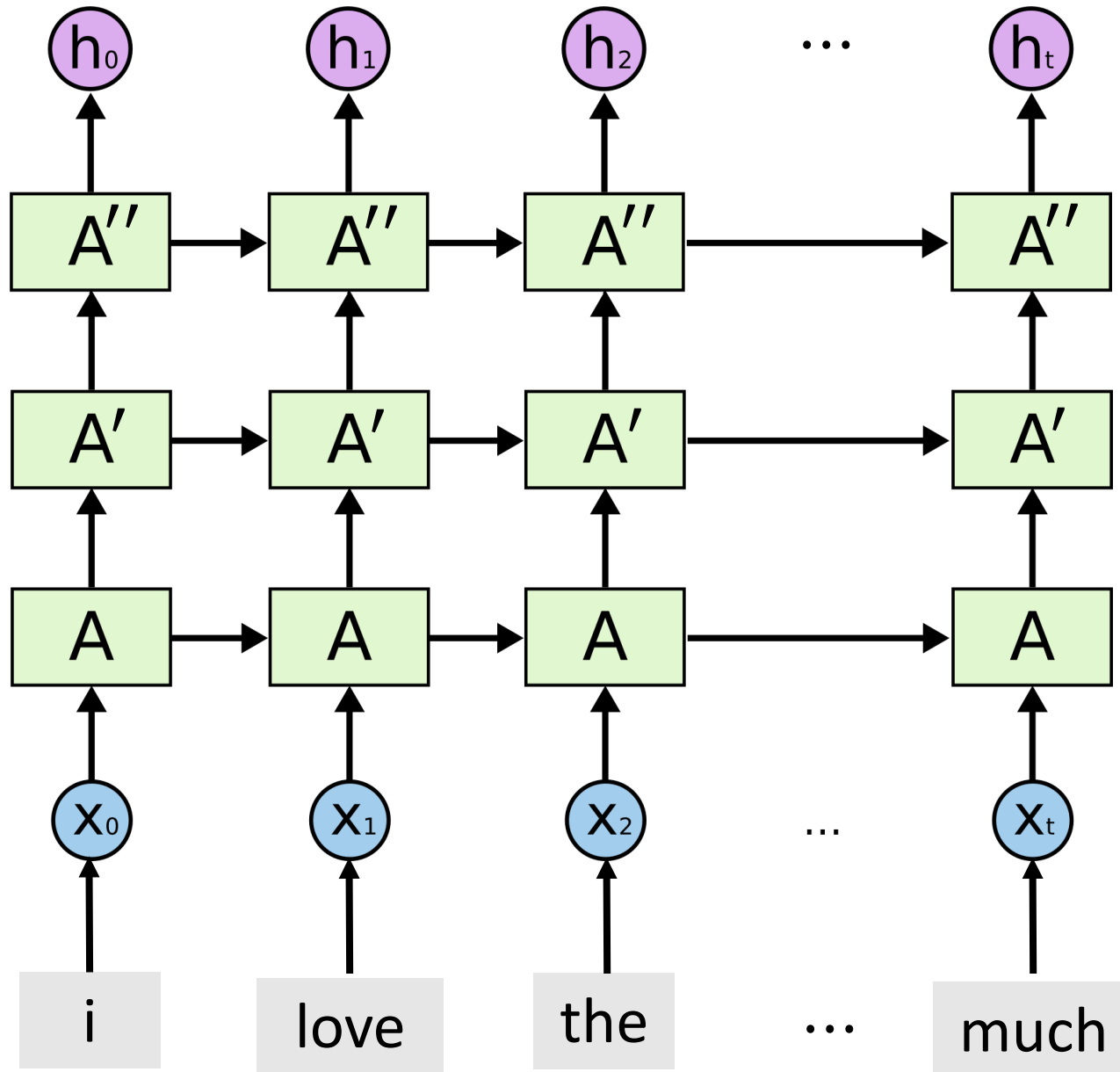
Stacked RNN



Stacked RNN



Stacked RNN



Stacked LSTM

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(LSTM(state_dim, return_sequences=True, dropout=0.2))
model.add(LSTM(state_dim, return_sequences=True, dropout=0.2))
model.add(LSTM(state_dim, return_sequences=False, dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

Stacked LSTM

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 500, 32)	320000
<hr/>		
lstm_1 (LSTM)	(None, 500, 32)	8320
<hr/>		
lstm_2 (LSTM)	(None, 500, 32)	8320
<hr/>		
lstm_3 (LSTM)	(None, 32)	8320
<hr/>		
dense_1 (Dense)	(None, 1)	33
=====		

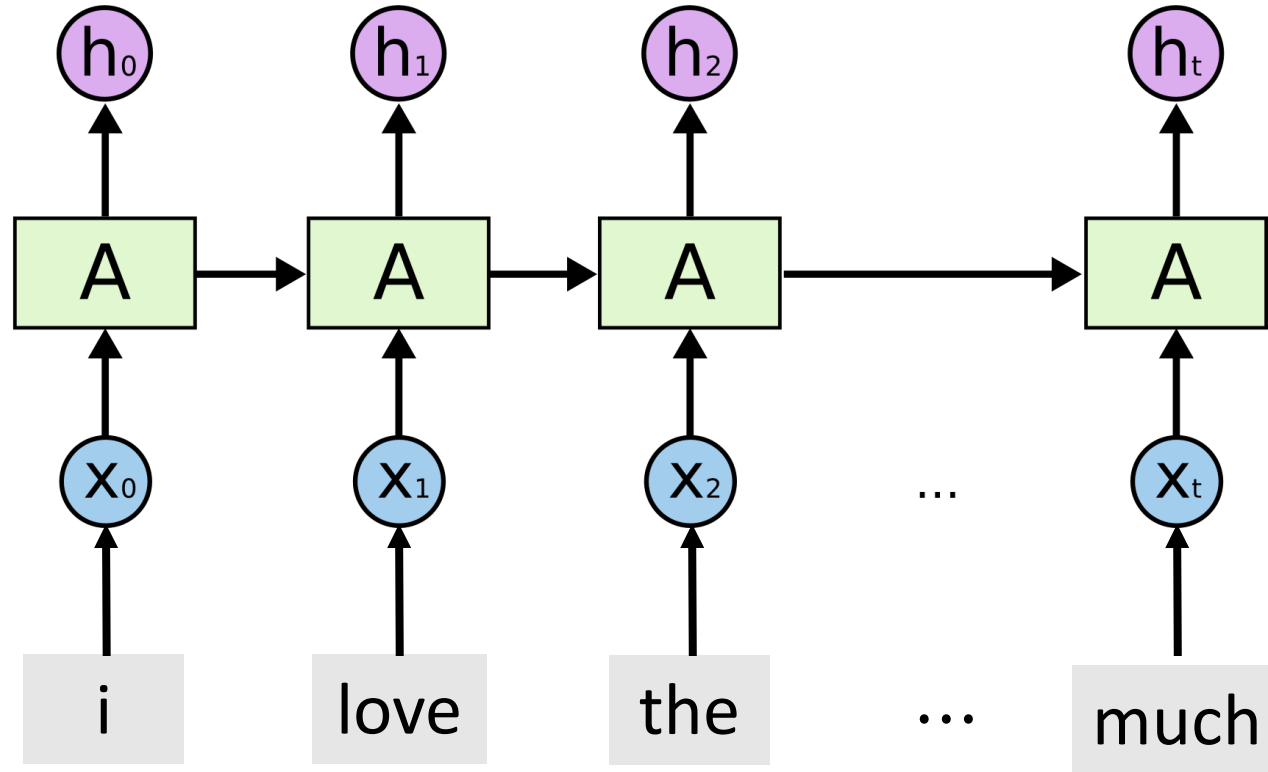
Total params: 344,993

Trainable params: 344,993

Non-trainable params: 0

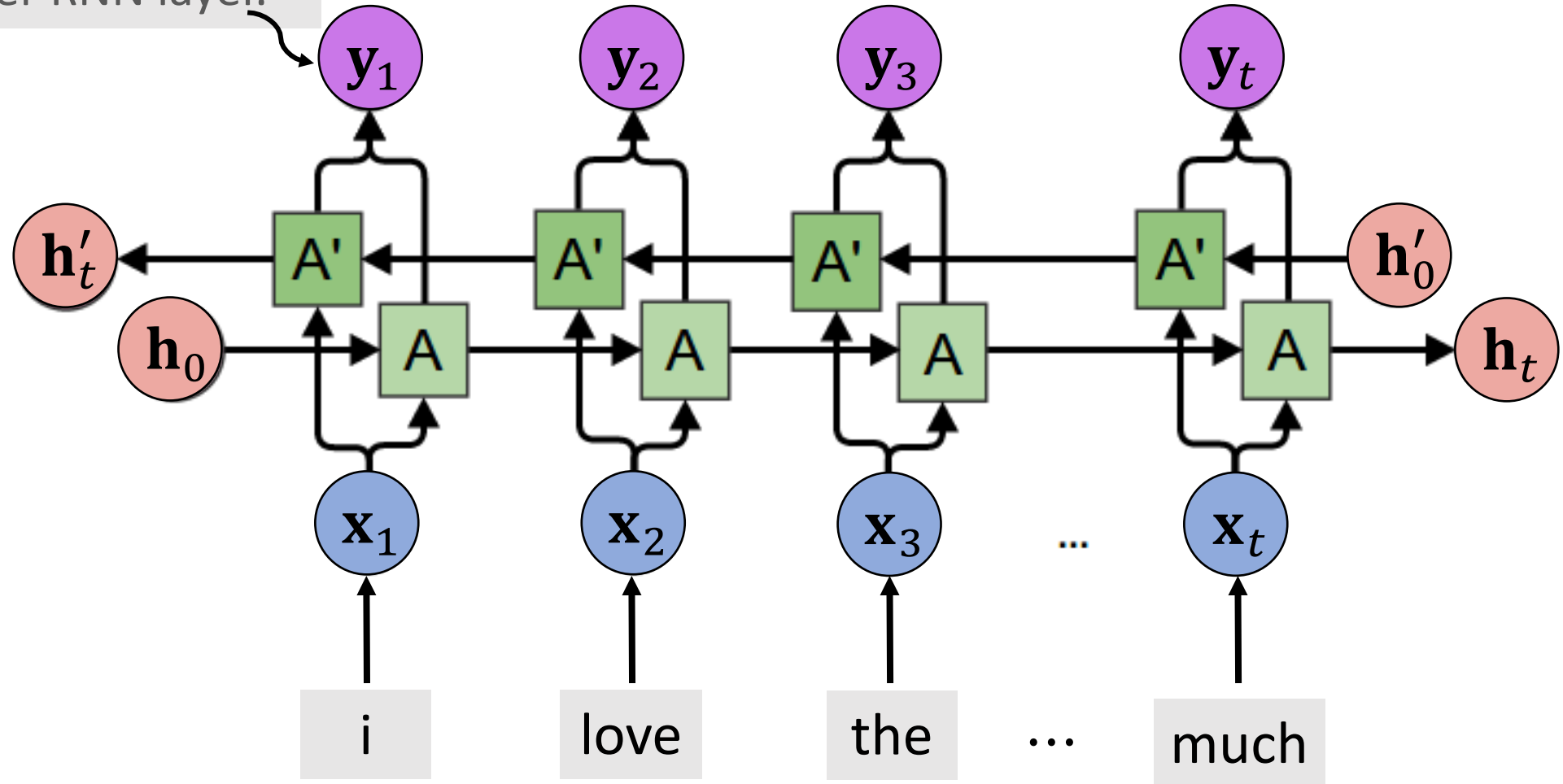
Bidirectional RNN

Standard RNN



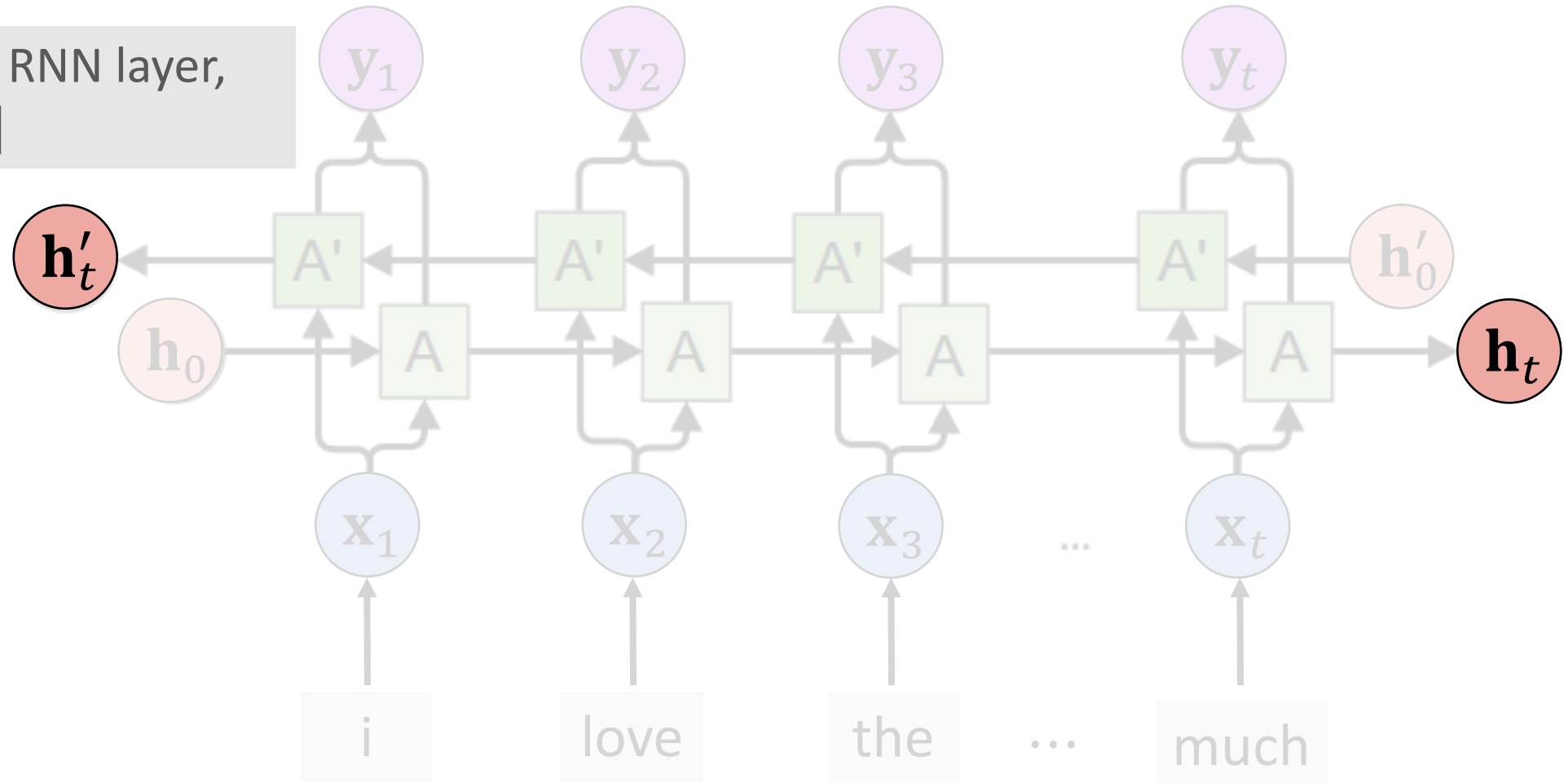
Bidirectional RNN

- Stack of 2 states.
- Passed to the upper RNN layer.



Bidirectional RNN

If there is no upper RNN layer,
then return $[\mathbf{h}_t, \mathbf{h}'_t]$



双向LSTM或者RNN效果总是比普通的好，
原因在于LSTM或RNN总是会遗忘之前的输入。

Bi-LSTM

```
from keras.models import Sequential
from keras.layers import LSTM, Embedding, Dense, Bidirectional

vocabulary = 10000
embedding_dim = 32
word_num = 500
state_dim = 32

model = Sequential()
model.add(Embedding(vocabulary, embedding_dim, input_length=word_num))
model.add(Bidirectional(LSTM(state_dim, return_sequences=False, dropout=0.2)))
model.add(Dense(1, activation='sigmoid'))

model.summary()
```

Bi-LSTM

Layer (type)	Output Shape	Param #
=====		
embedding_1 (Embedding)	(None, 500, 32)	320000
bidirectional_1	(Bidirection (None, 64)	16640
dense_1 (Dense)	(None, 1)	65
=====		
Total params: 336,705		
Trainable params: 336,705		
Non-trainable params: 0		

Pretrain

Why pretraining?

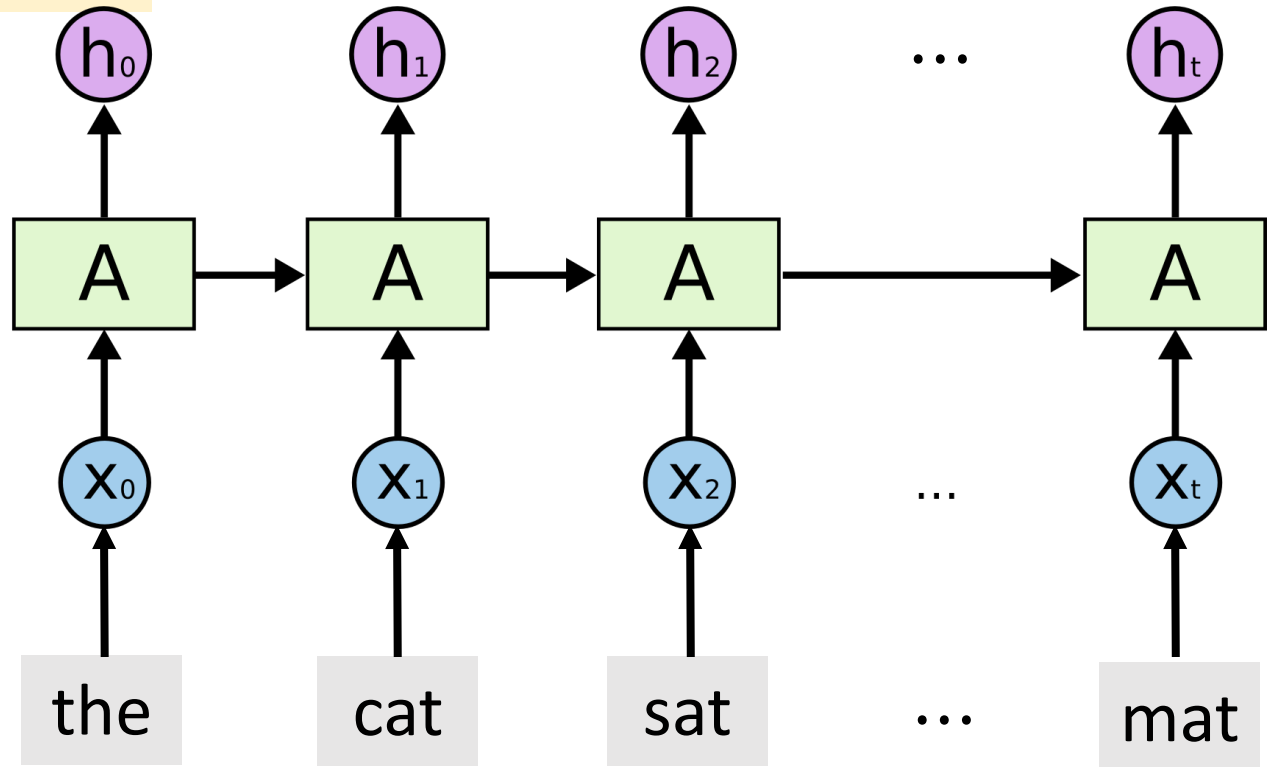
Observation: The **embedding layer** contributes most of the parameters!

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
bidirectional_1 (Bidirectional)	(None, 64)	16640
dense_1 (Dense)	(None, 1)	65
=====		
Total params: 336,705		
Trainable params: 336,705		
Non-trainable params: 0		

Pretrain the Embedding Layer

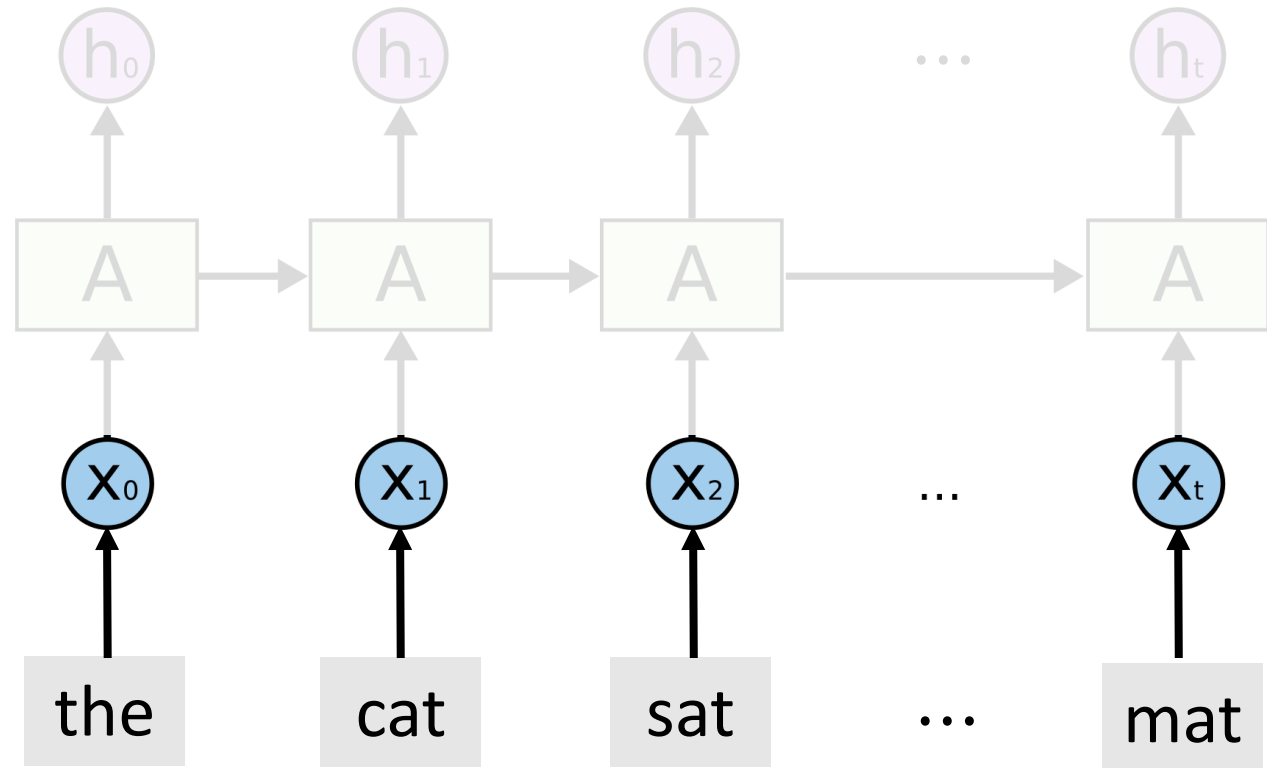
Step 1: Train a model on large dataset.

- Perhaps different problem.
- Perhaps different model.



Pretrain the Embedding Layer

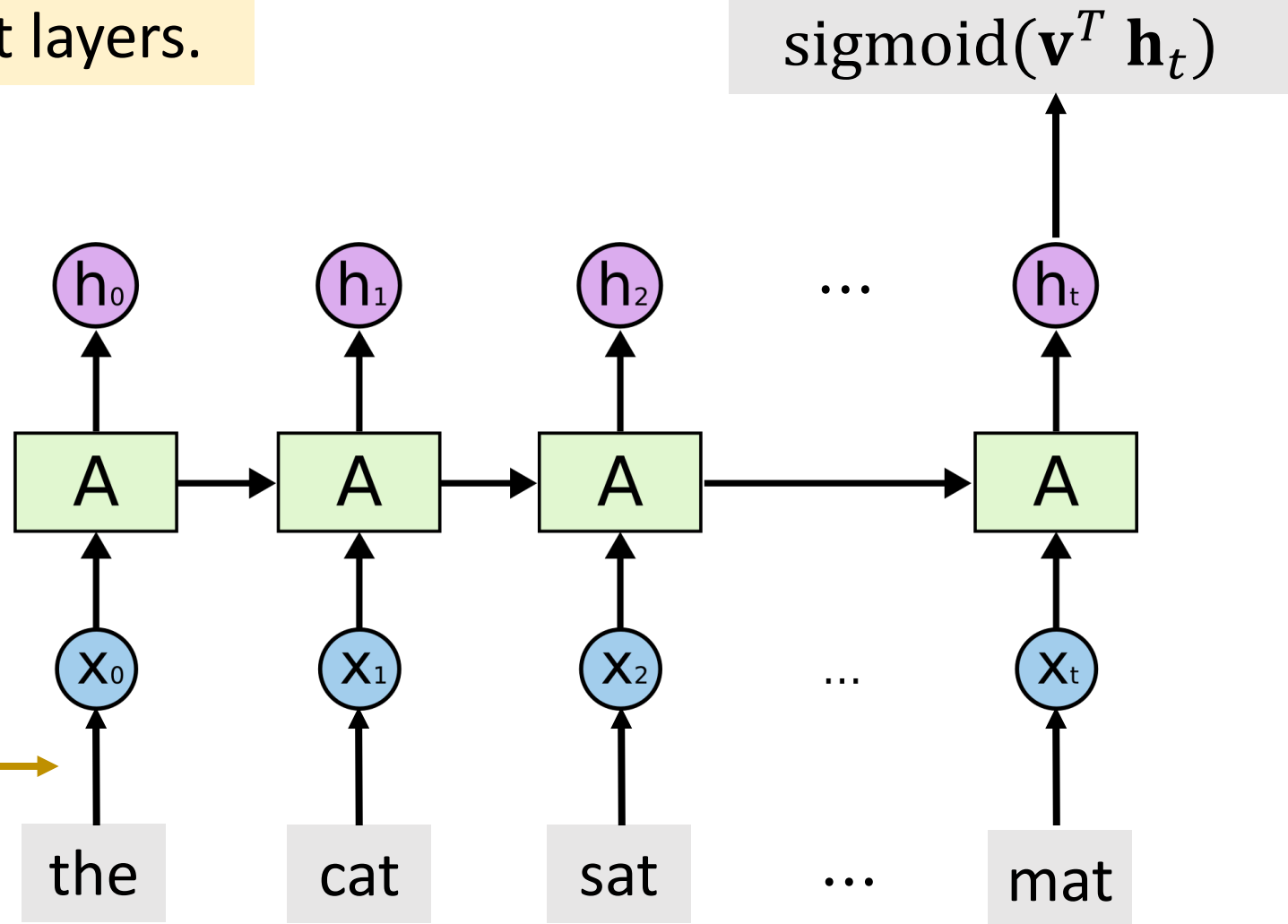
Step 2: Keep only the embedding layer.



Pretrain the Embedding Layer

Step 3: Train new LSTM and output layers.

Set the embedding layer **non-trainable**.



Summary

Summary

- SimpleRNN and LSTM are two kinds of RNNs; always use **LSTM** instead of **SimpleRNN**.
- Use **Bi-RNN** instead of RNN whenever possible.
- **Stacked RNN** may be better than a single RNN layer (if n is big).
 n 是训练数据大小
- **Pretrain** the embedding layer (if n is small).

Thank you!