Making RNNs More Effective

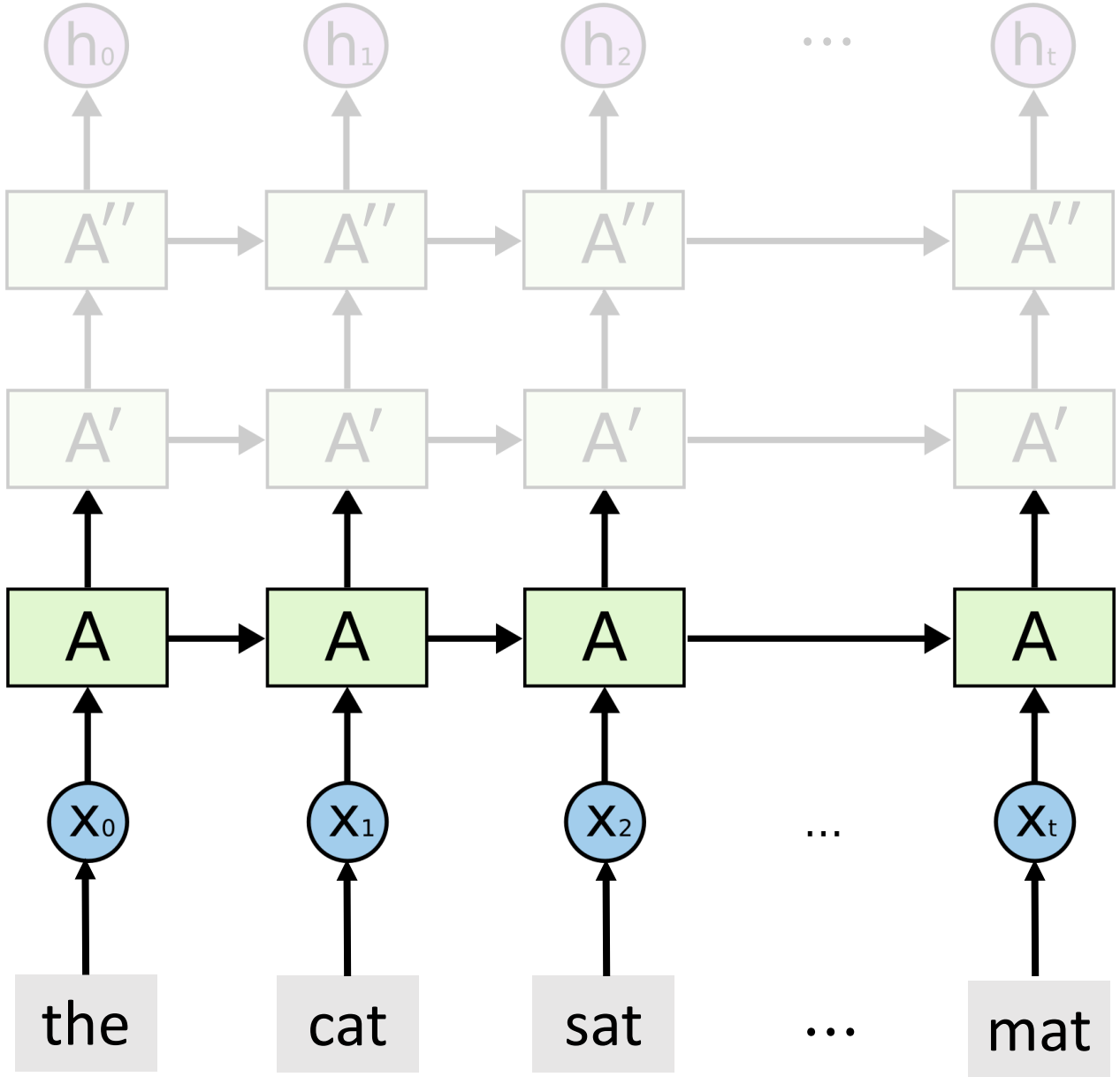
使rnn更有效

Shusen Wang

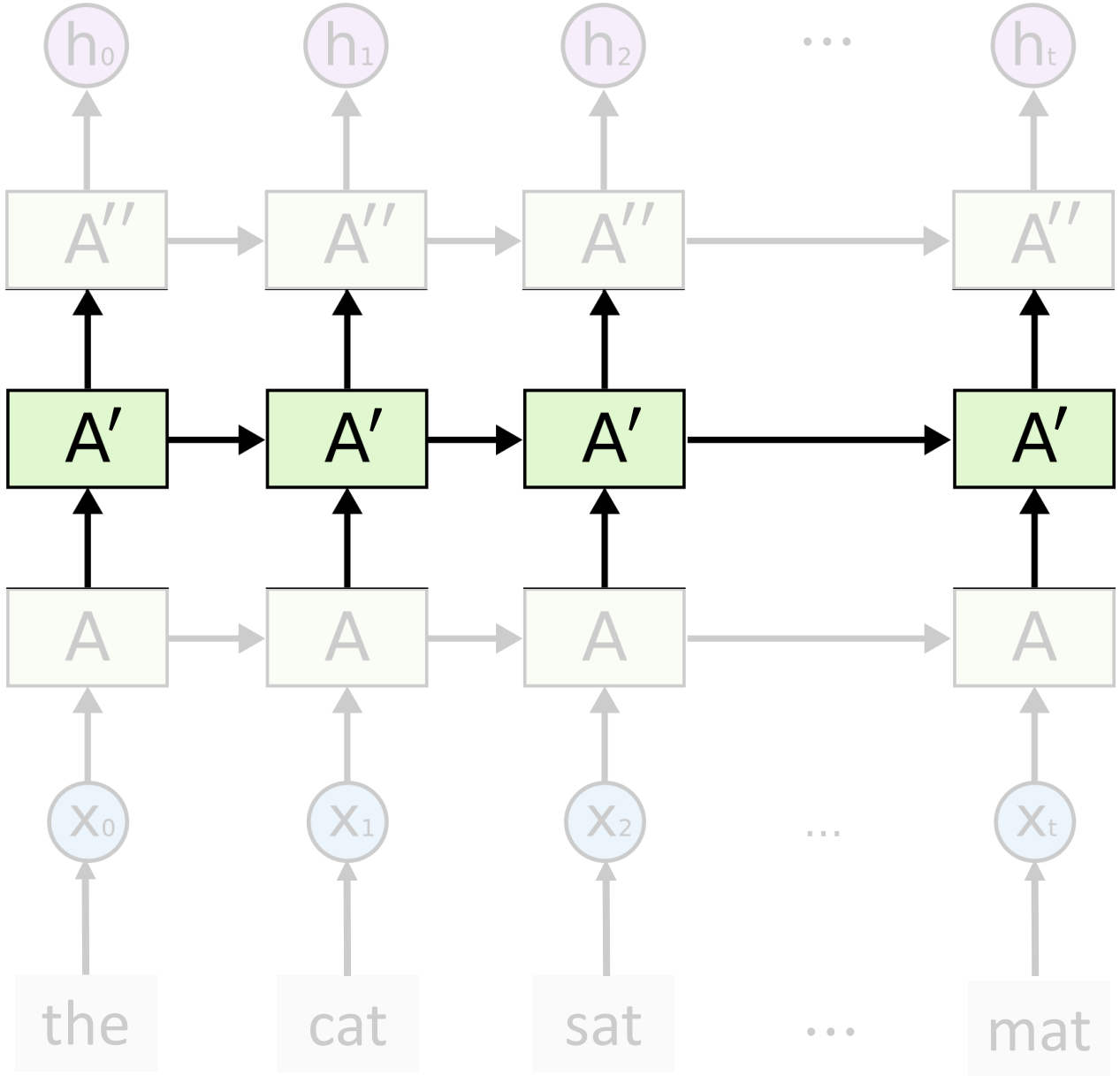
Stacked RNN

Stacked RNN

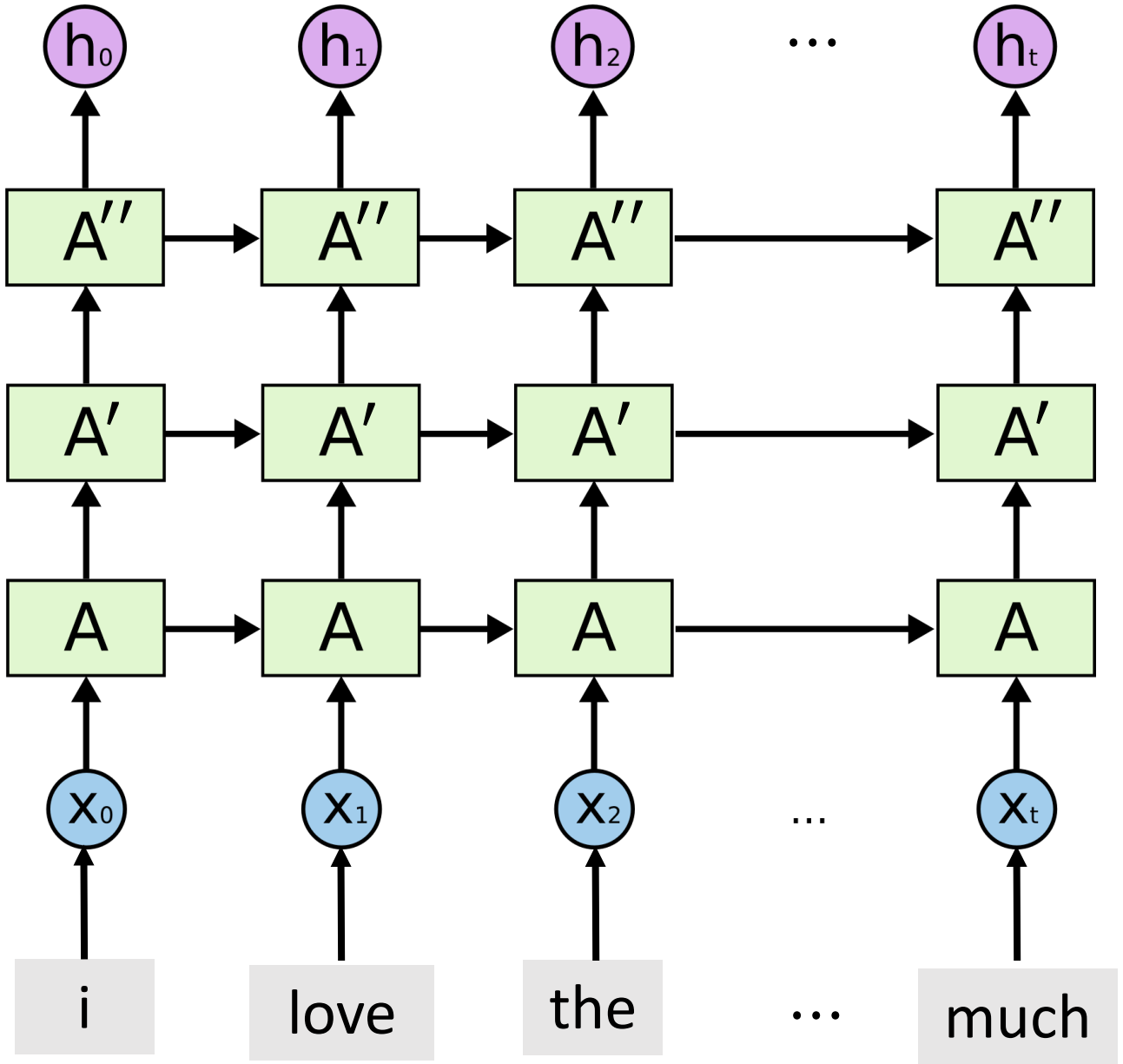
堆叠RNN



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'))
```

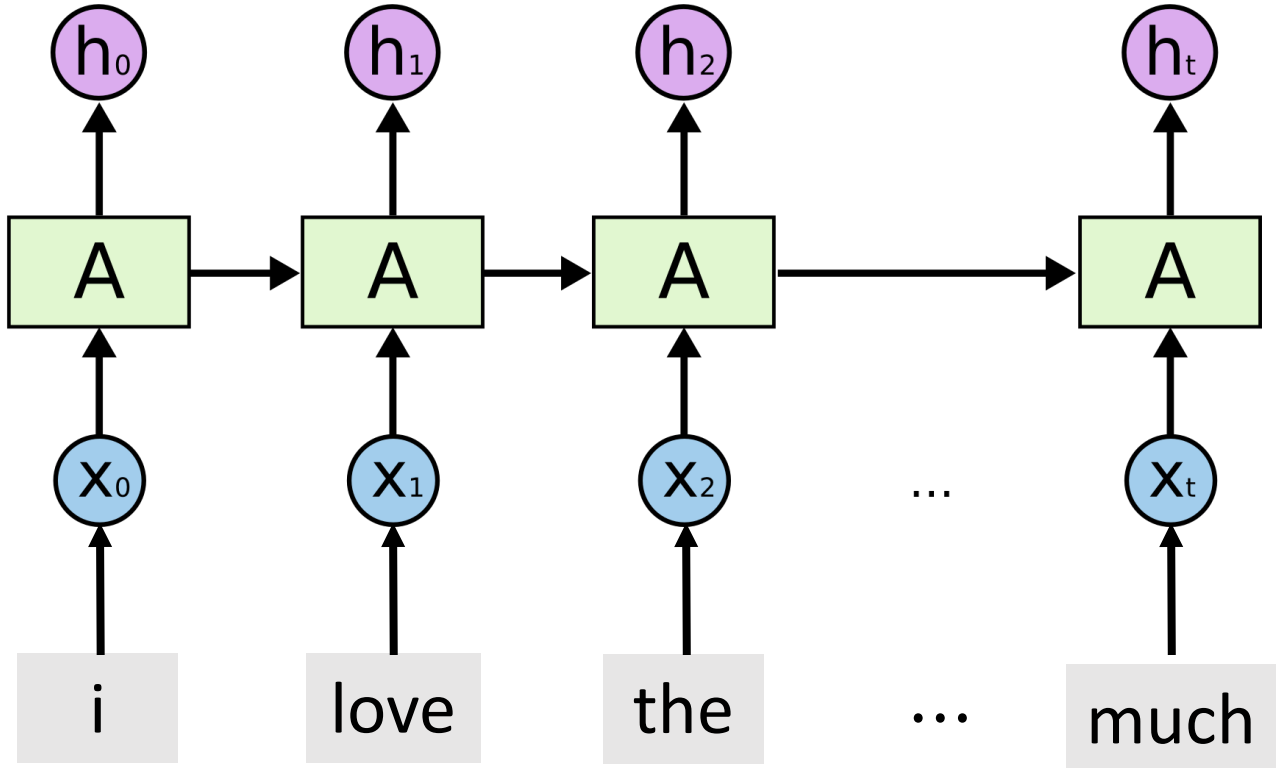
Stacked LSTM

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

Total params: 344,993
Trainable params: 344,993
Non-trainable params: 0

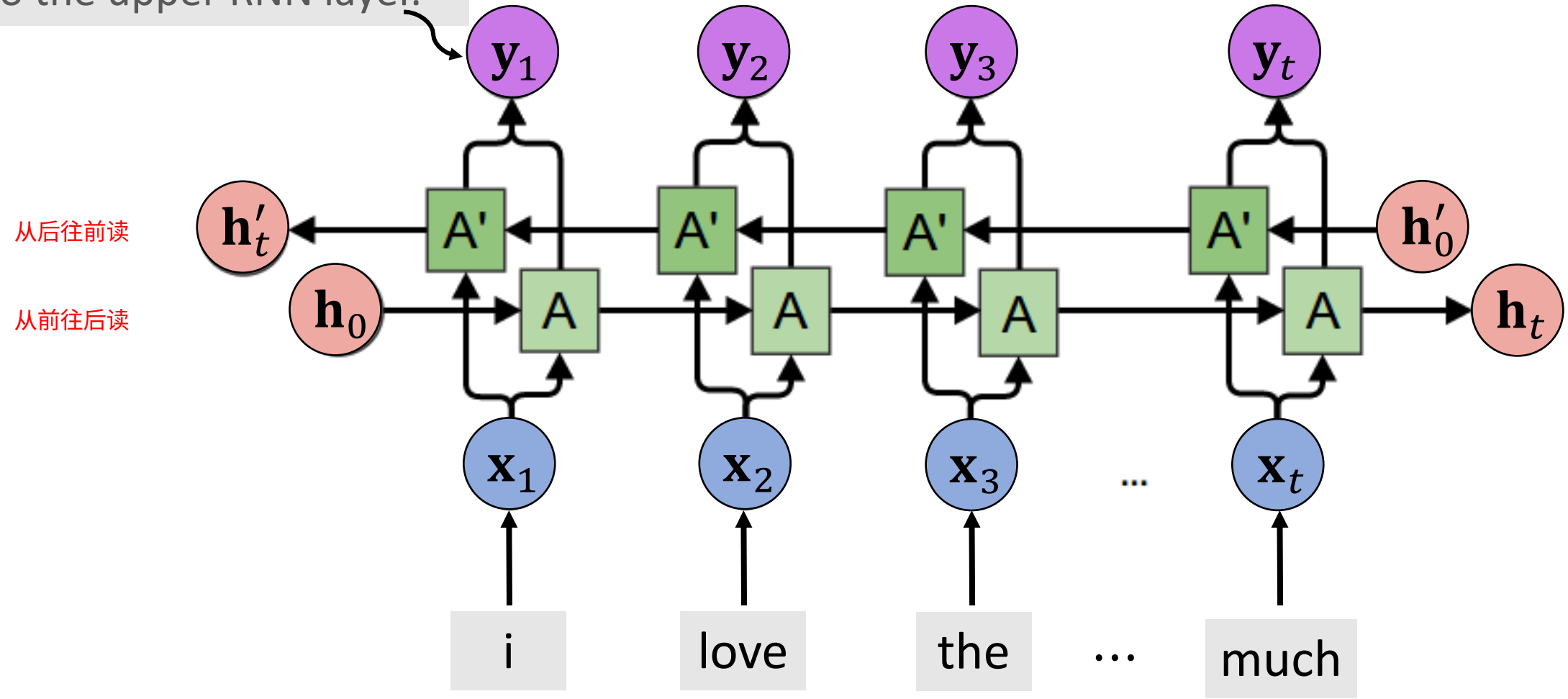
Bidirectional RNN

Standard RNN



Bidirectional RNN 双向RNN

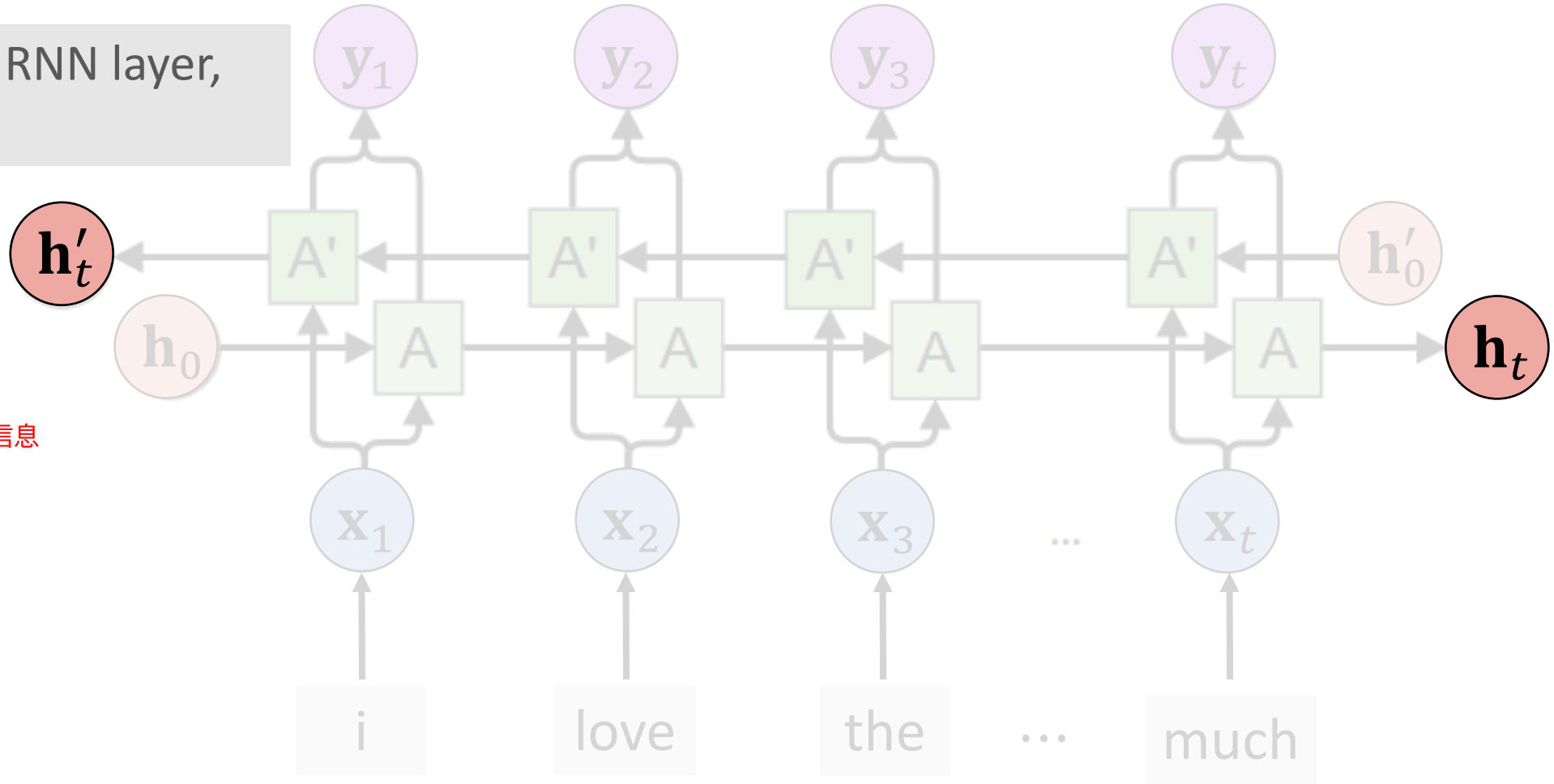
- Stack of 2 states.
- Passed to the upper RNN layer.



Bidirectional RNN

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

concat



RNN 会遗忘一些信息
Bi RNN 能尽量保留 较为久远的信息

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

单向：8320

Total params: 336,705
Trainable params: 336,705
Non-trainable params: 0

Pretrain

Why pretraining?

Observation: The embedding layer contributes most of the parameters!

观察到：嵌入层贡献了大部分的参数！

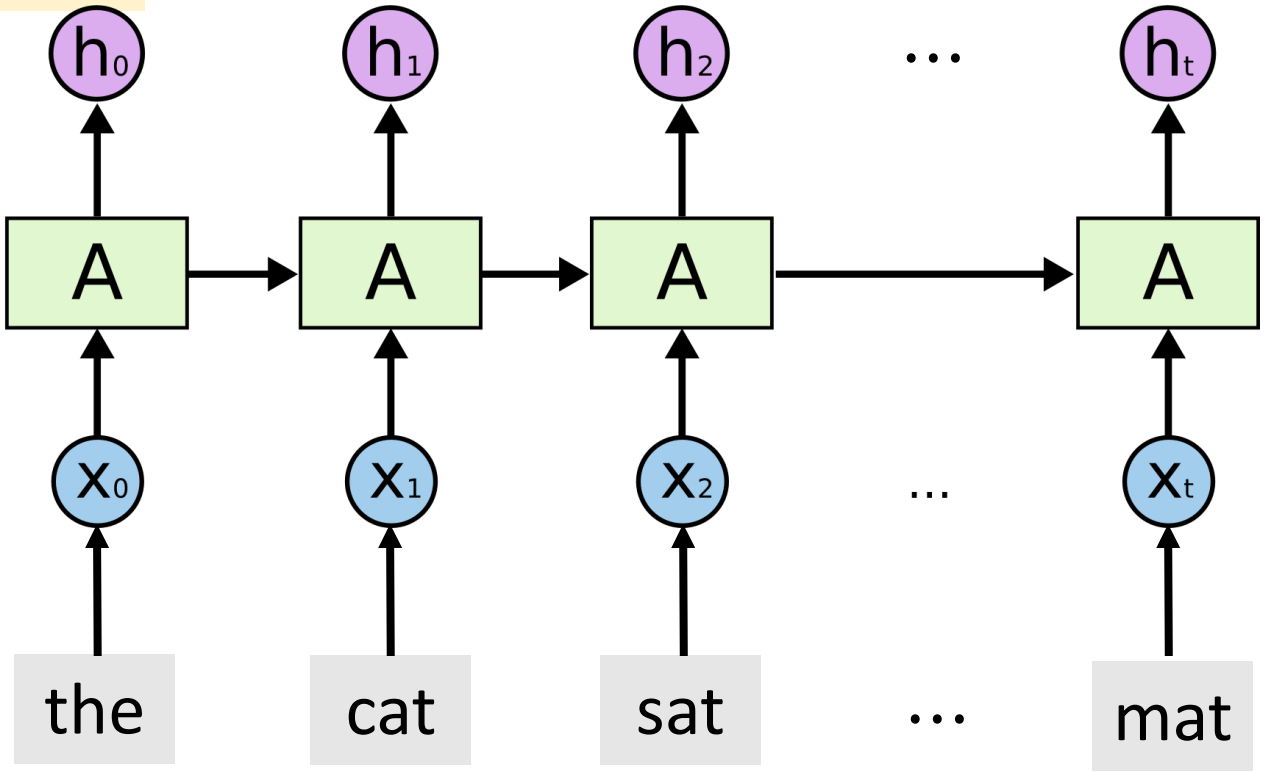
```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           320000
_________________________________________________________________
bidirectional_1 (Bidirection (None, 64)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

Pretrain the Embedding Layer

Step 1: Train a model on large dataset.

- Perhaps different problem.
- Perhaps different model.

步骤1：在大数据集上训练一个模型。
?也许有不同的问题。（尽量是相同内容的数据）
?也许是不同的模型。

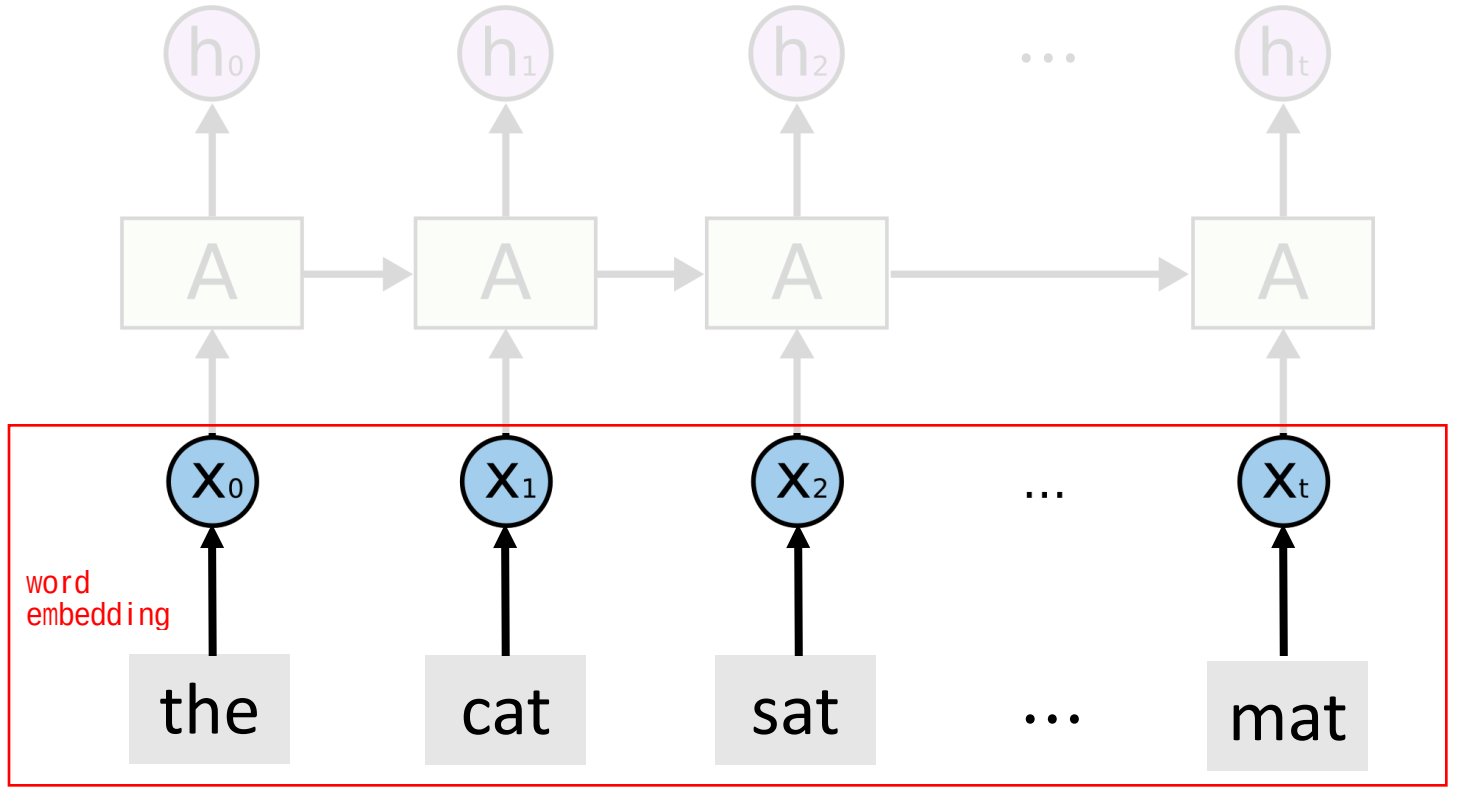


Pretrain the Embedding Layer

Step 2: Keep only the embedding layer.

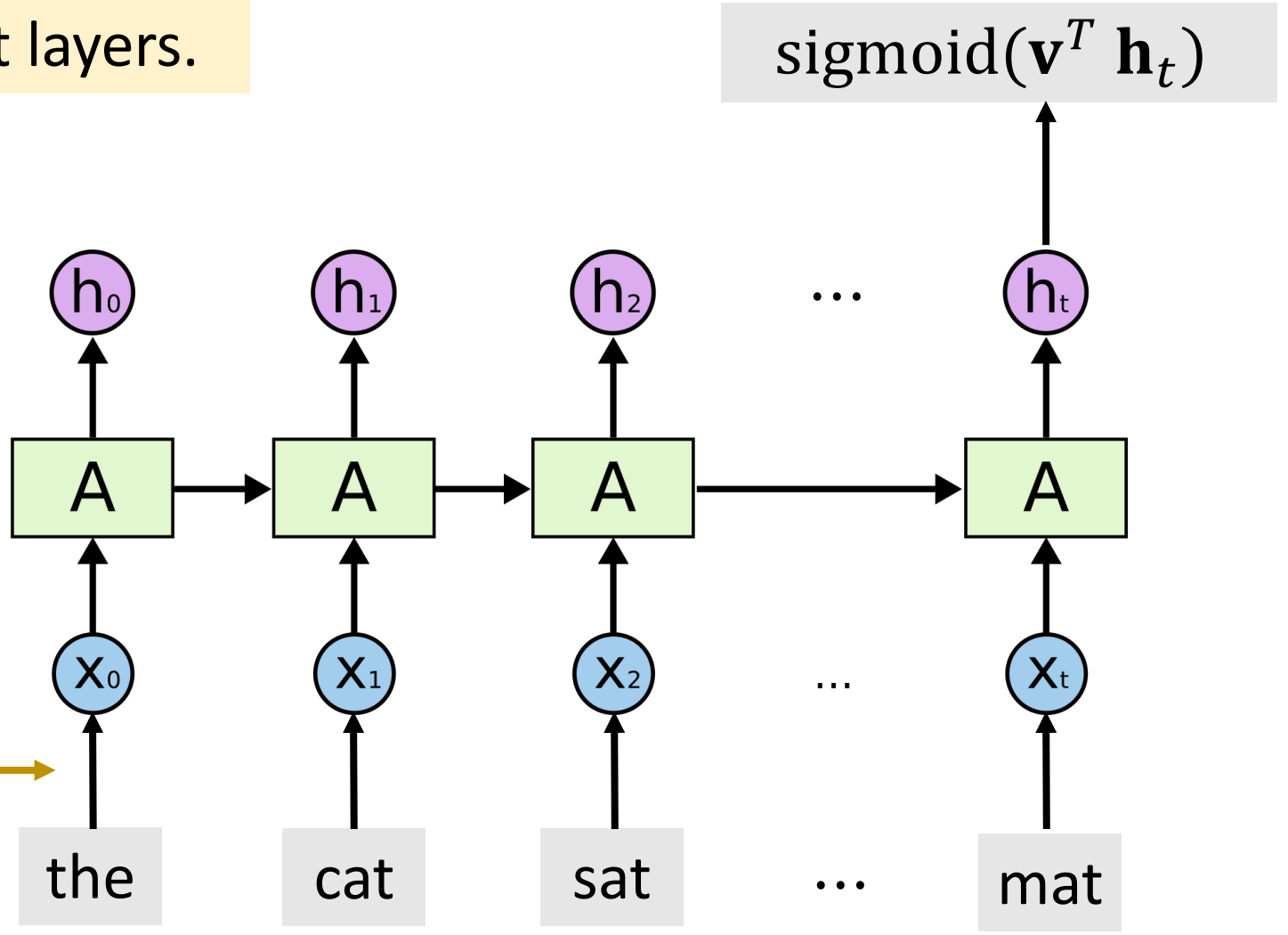
只保留嵌入层

保留



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).
- Pretrain the embedding layer (if n is small). 预训练嵌入层(如果n都是很小的)。

Thank you!