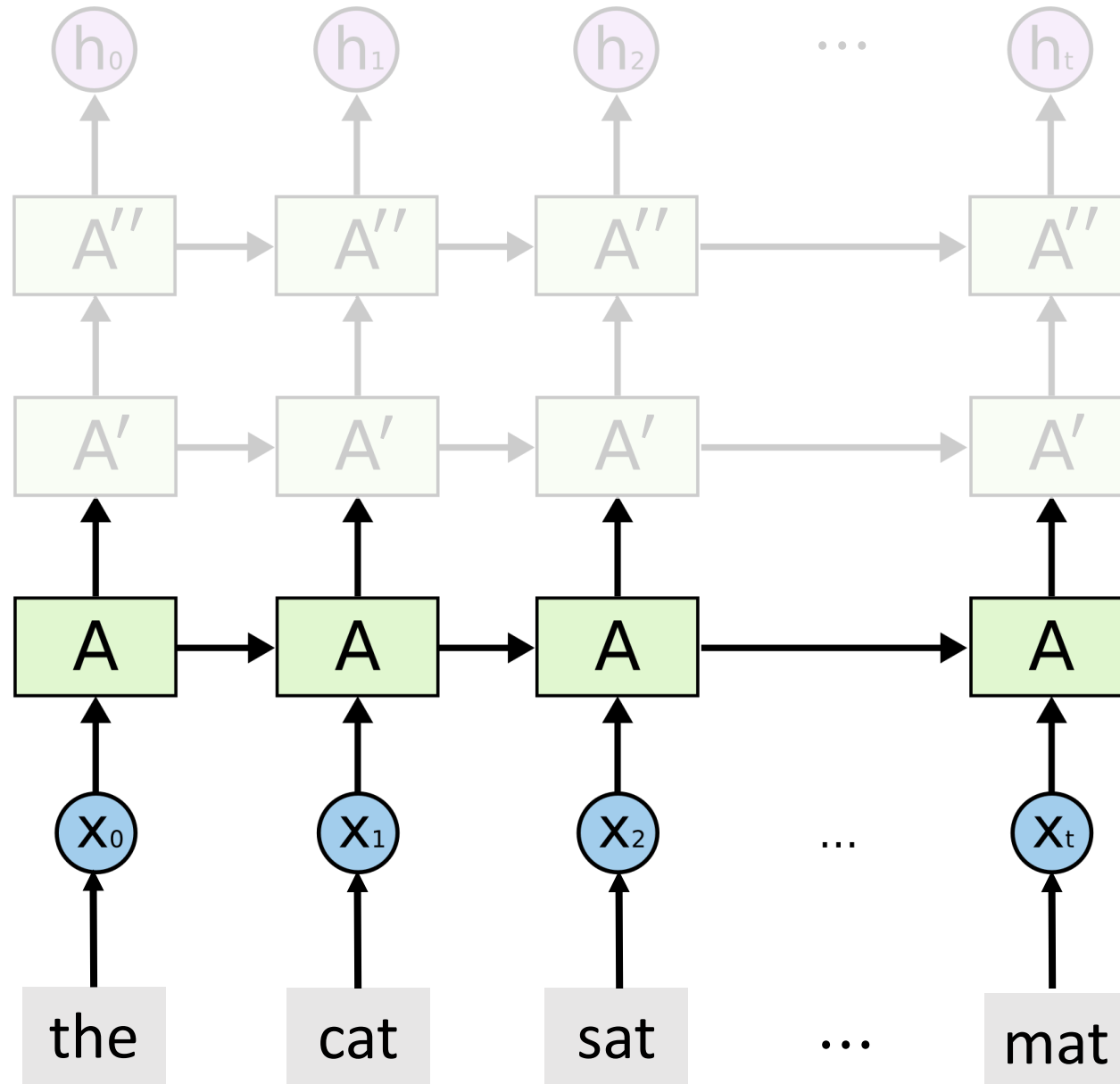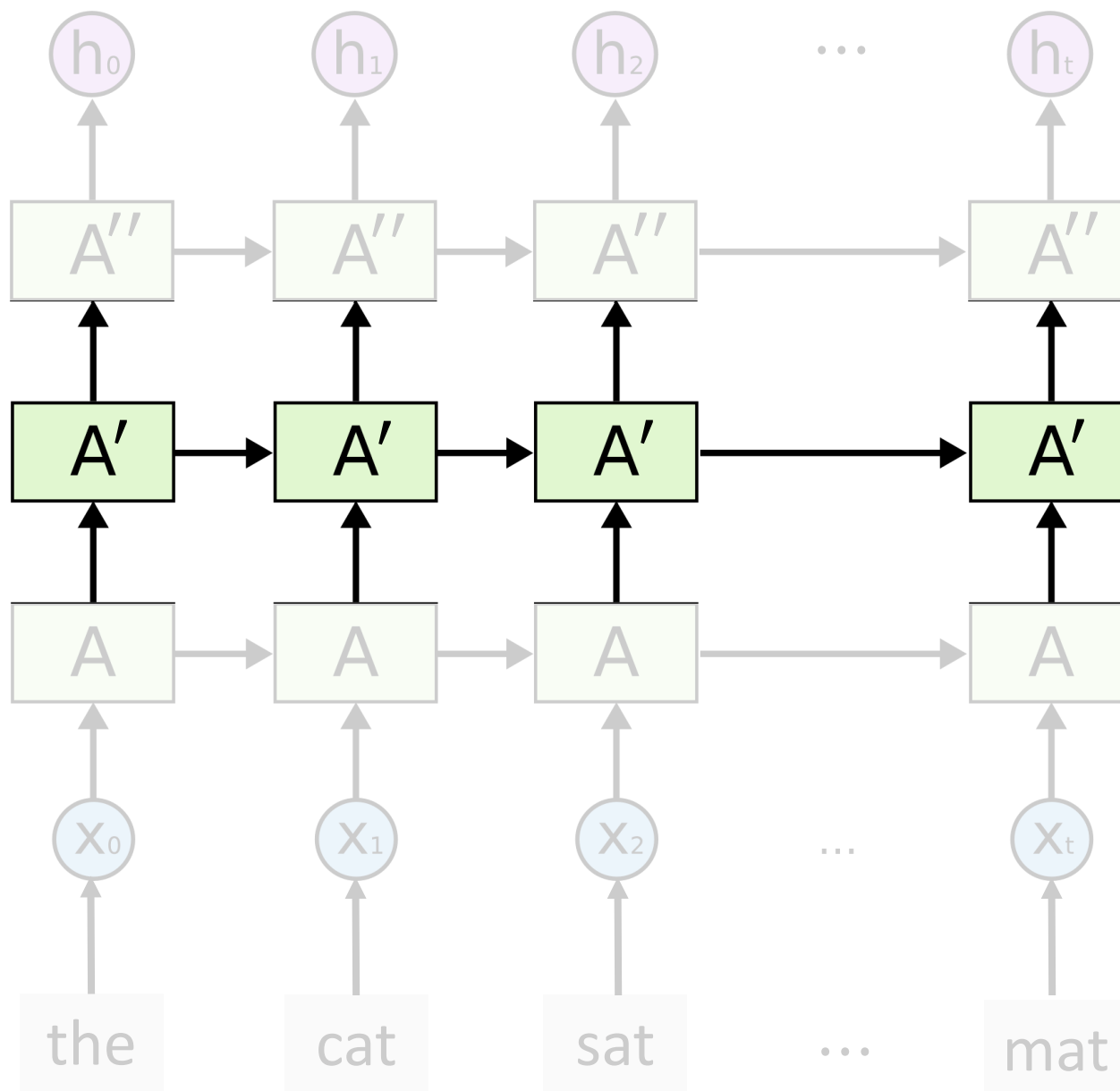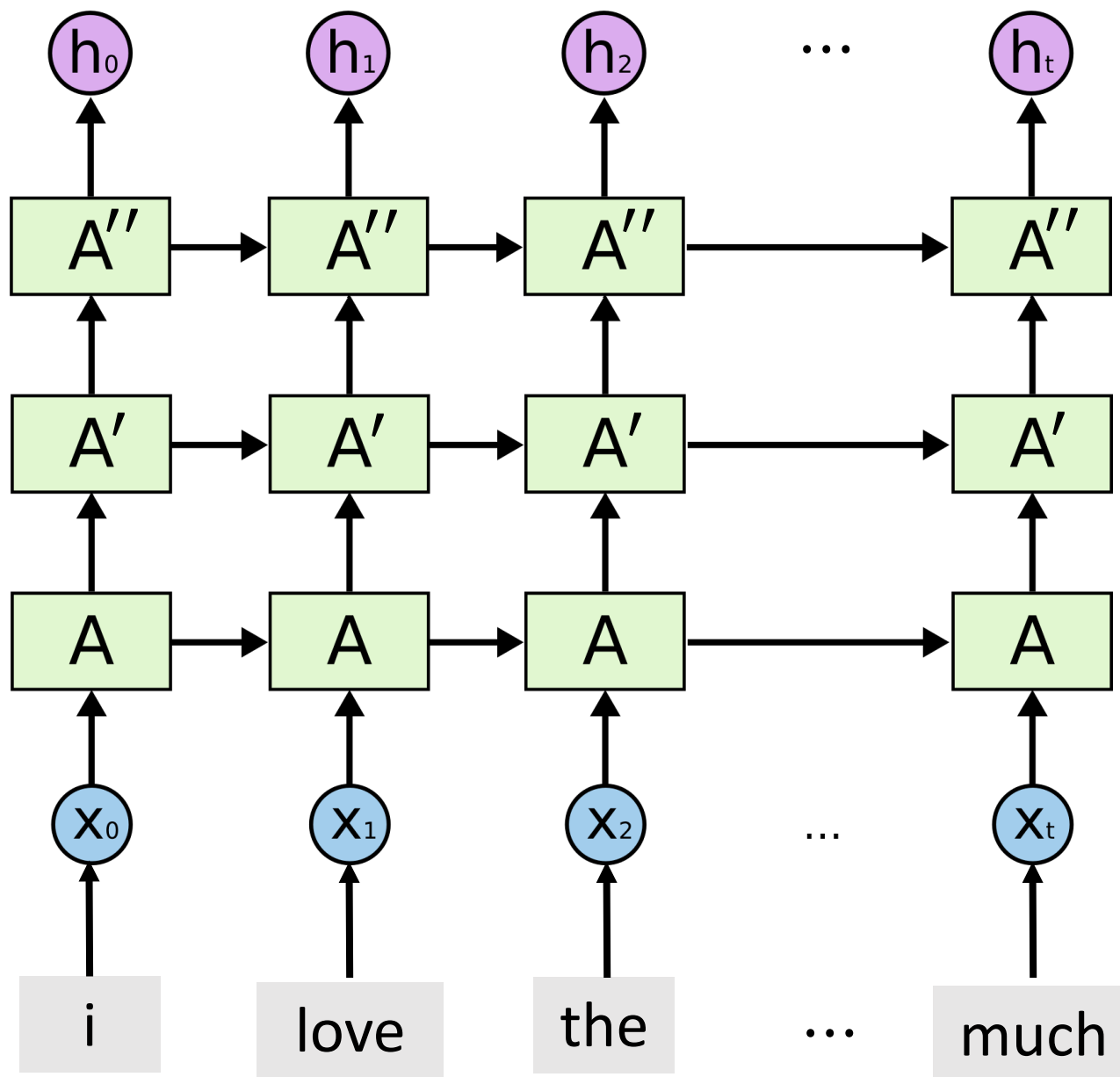# Stacked RNN

Stacked RNN

Stacked RNN

Stacked RNN

Stacked LSTM

```python
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'))
```

`model.summary()`

Stacked LSTM

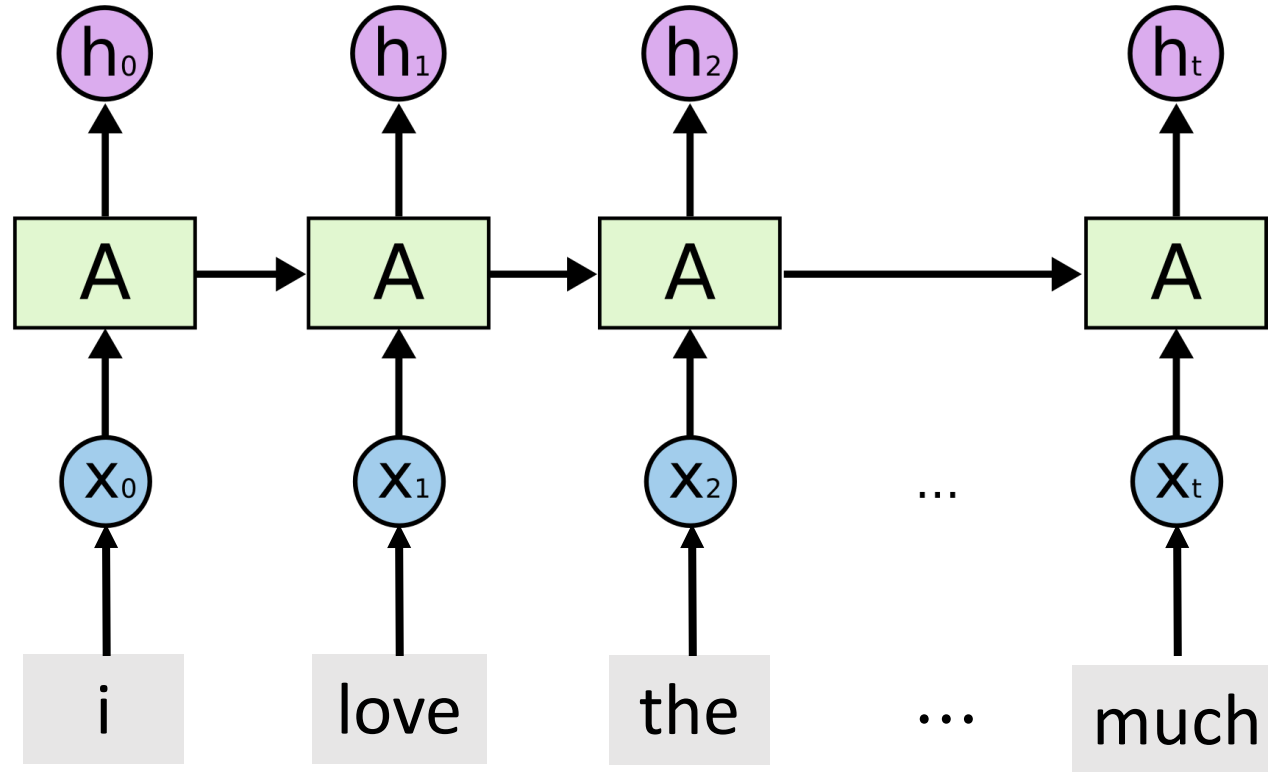| Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 500, 32)	320000
lstm_1 (LSTM)	(None, 500, 32)	8320
lstm_2 (LSTM)	(None, 500, 32)	8320
lstm_3 (LSTM)	(None, 32)	8320
dense_1 (Dense)	(None, 1)	33

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

Bidirectional RNN

Standard RNN

Bidirectional RNN
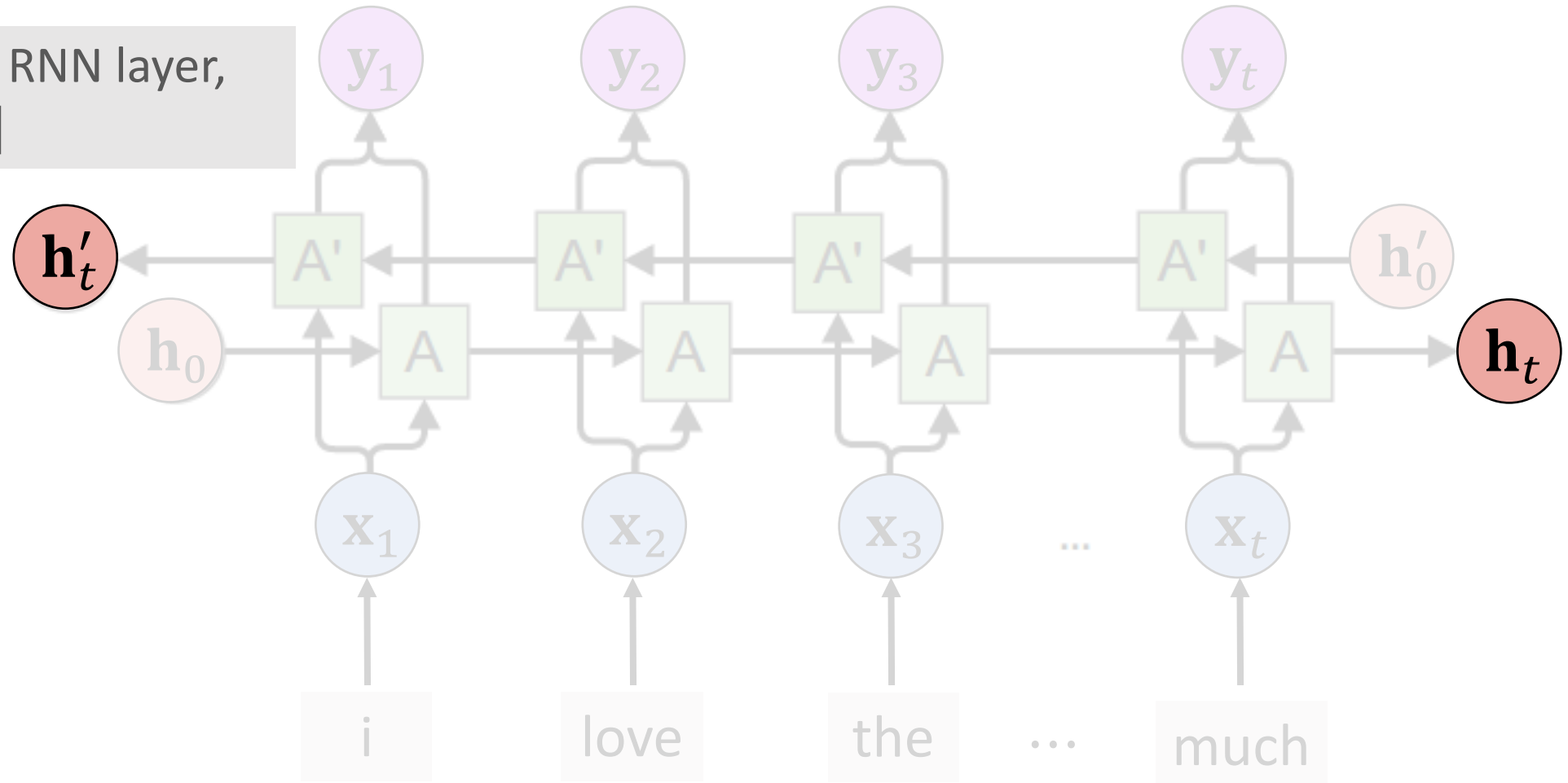
$\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$ $\mathbf{y}_t$

$\mathbf{h}'_t$ A' A' A' A' $\mathbf{h}'_0$

$\mathbf{h}_0$ A A A A $\mathbf{h}_t$

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\cdots$ $\mathbf{x}_t$

i love the $\cdots$ much
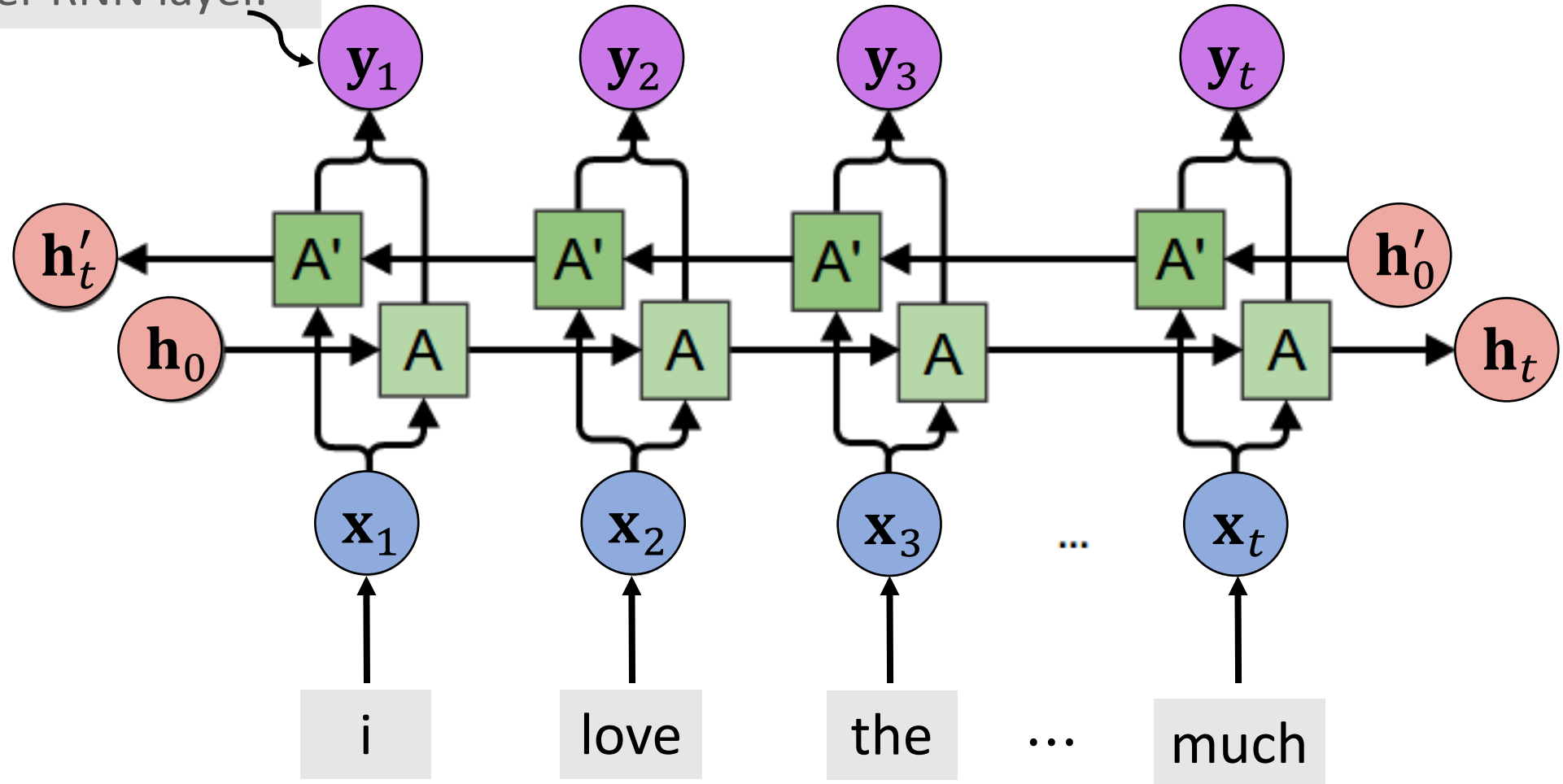
Bidirectional RNN

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

$\mathbf{y}_1$ $\mathbf{y}_2$ $\mathbf{y}_3$ $\mathbf{y}_t$

$\mathbf{h}'_t$ A' A' A' A' $\mathbf{h}'_0$

$\mathbf{h}_0$ A A A A $\mathbf{h}_t$

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\cdots$ $\mathbf{x}_t$

i love the ... much

6. Bi-LSTM

Bi-LSTM

```python
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()
```

Layer (type)   Output Shape   Param #

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

Bi-LSTM

```
model.summary()
```

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?

```
model.add(Bidirectional(LSTM(state_dim, return_sequences=False,
model.add(Dense(1, activation='sigmoid'
```

Observation: The embedding layer contributes most of the parameters!

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
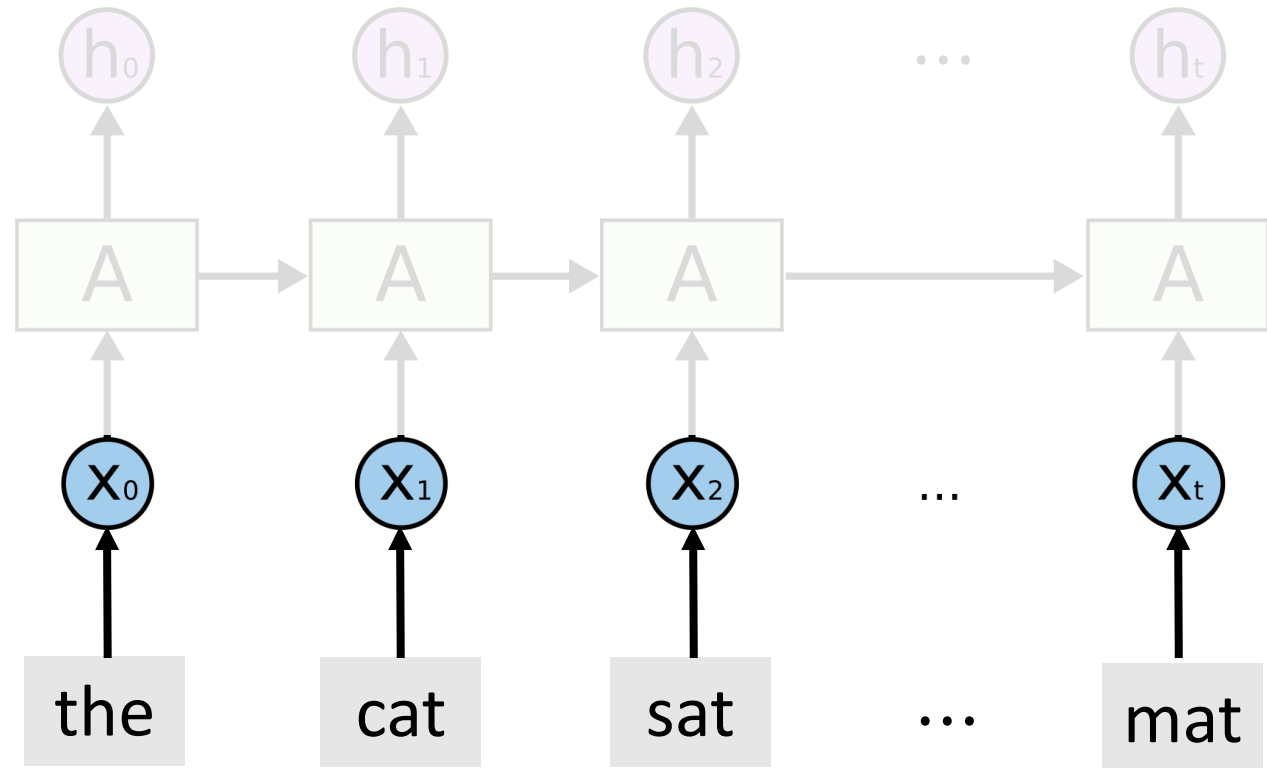
Pretrain the Embedding Layer

Step 1: Train a model on large dataset.
- Perhaps different problem.
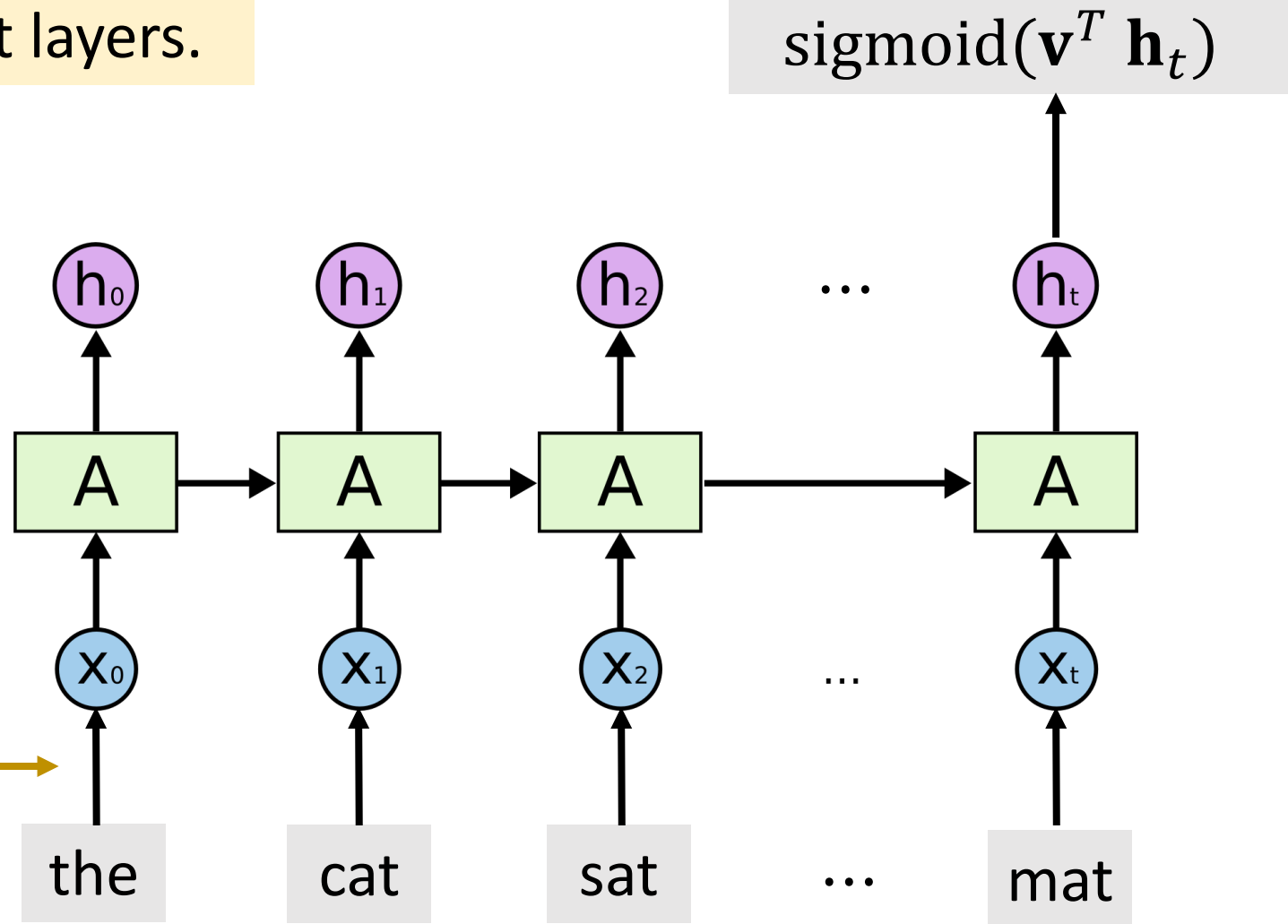- Perhaps different model.
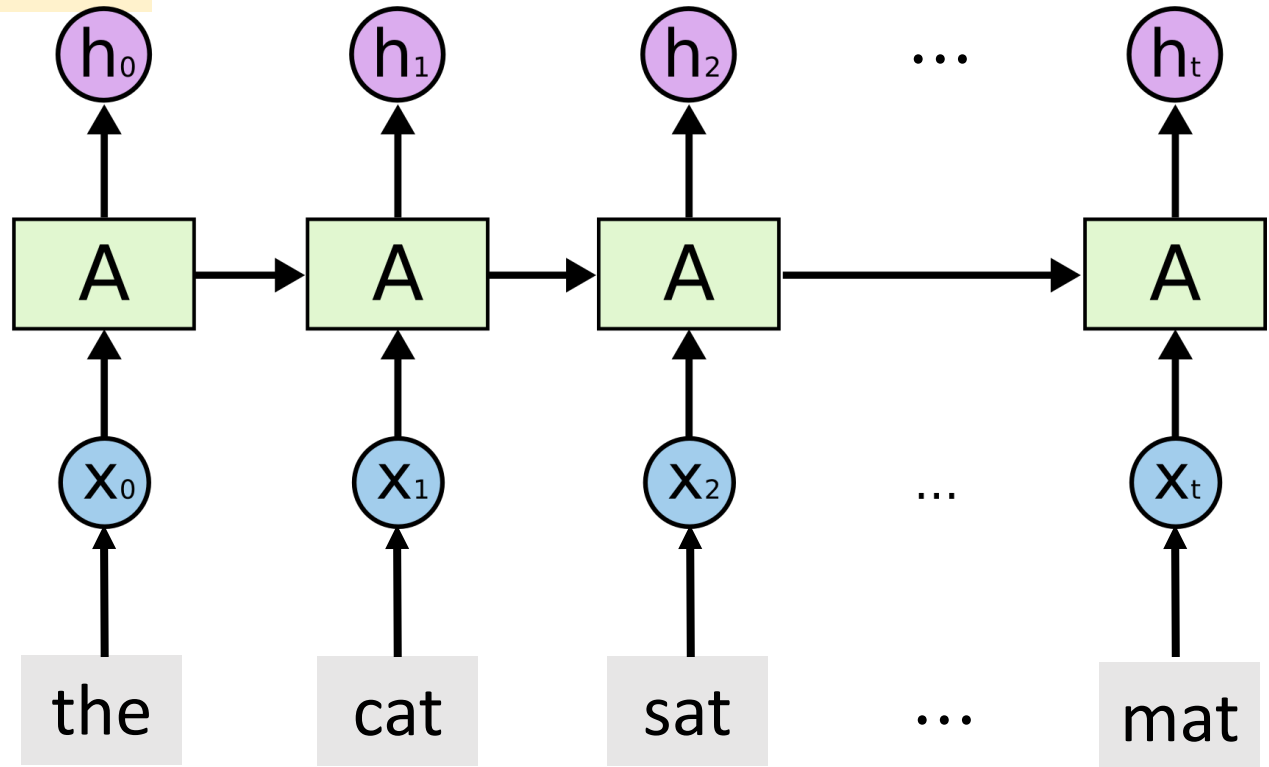
Pretrain the Embedding Layer

Pretrain the Embedding Layer

$$\text{sigmoid}(\mathbf{v}^T \mathbf{h}_t)$$

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

the   cat   sat   …   mat

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).

Thank you!