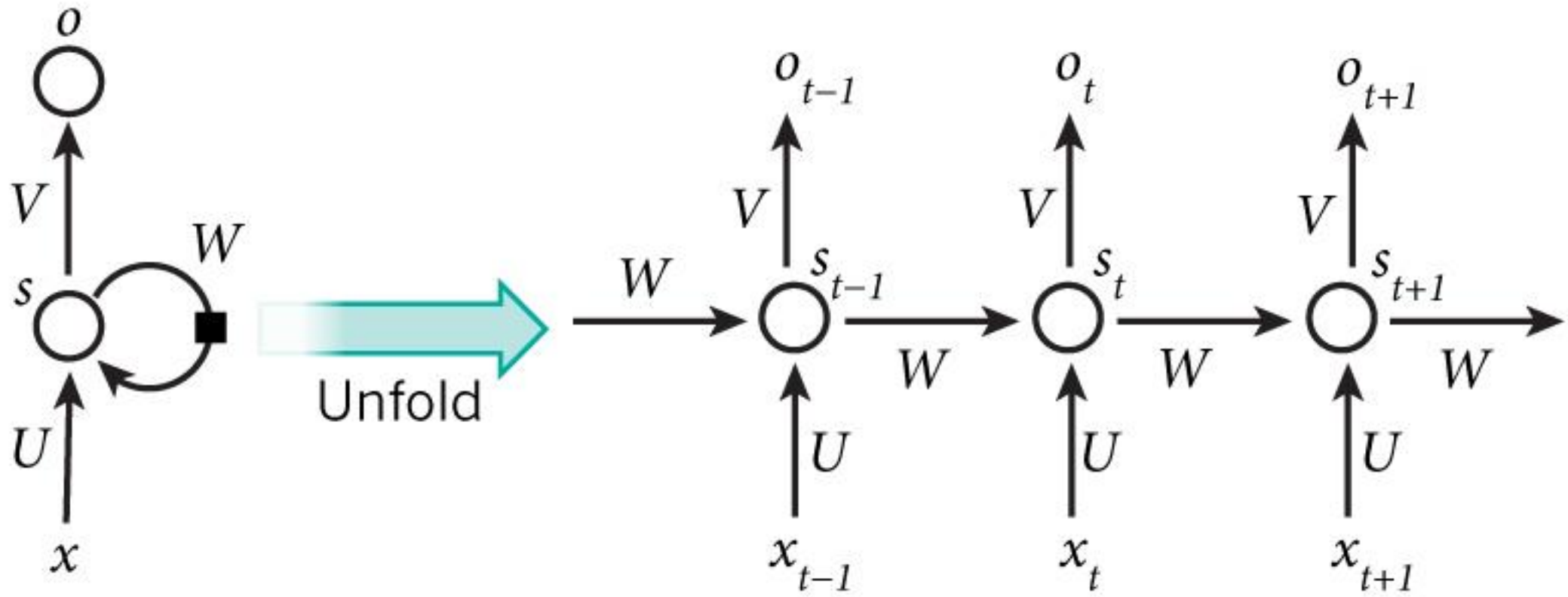
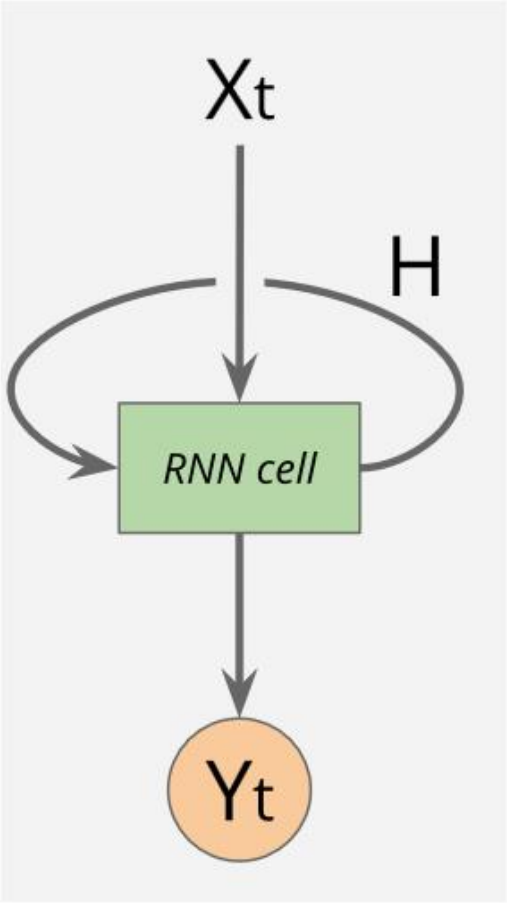
Recurrent Neural Network

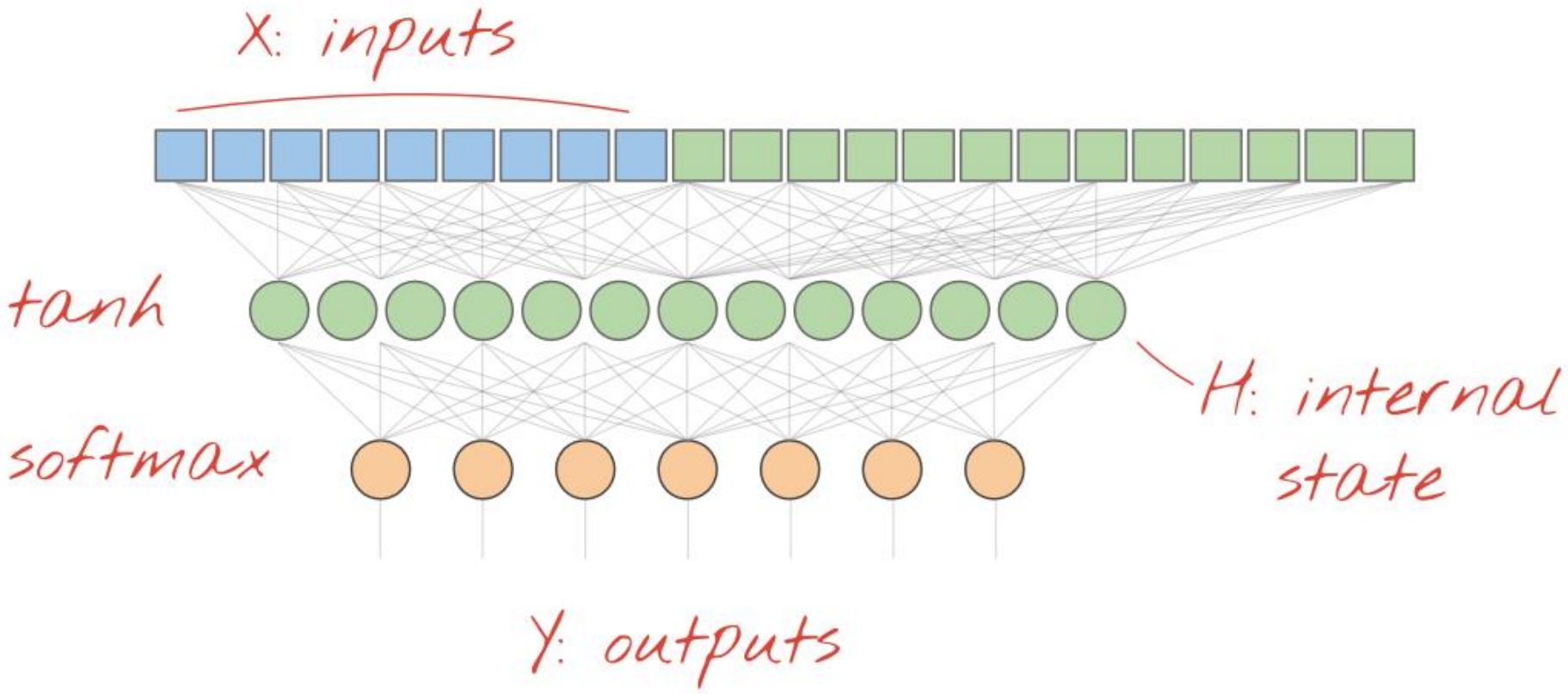
By Comdet Phaudphut, fb.com/comdet, comdet.p@gmail.com

What is Recurrent Neural Network

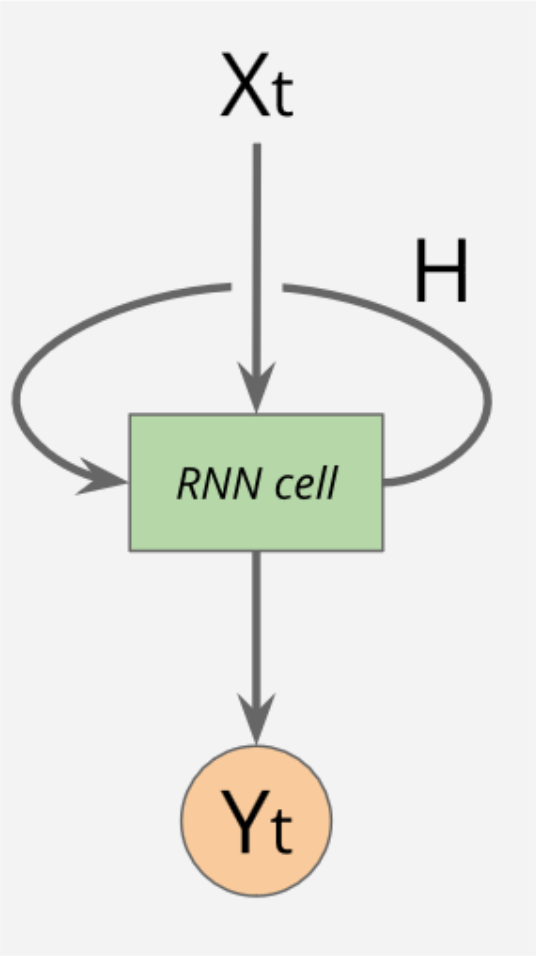


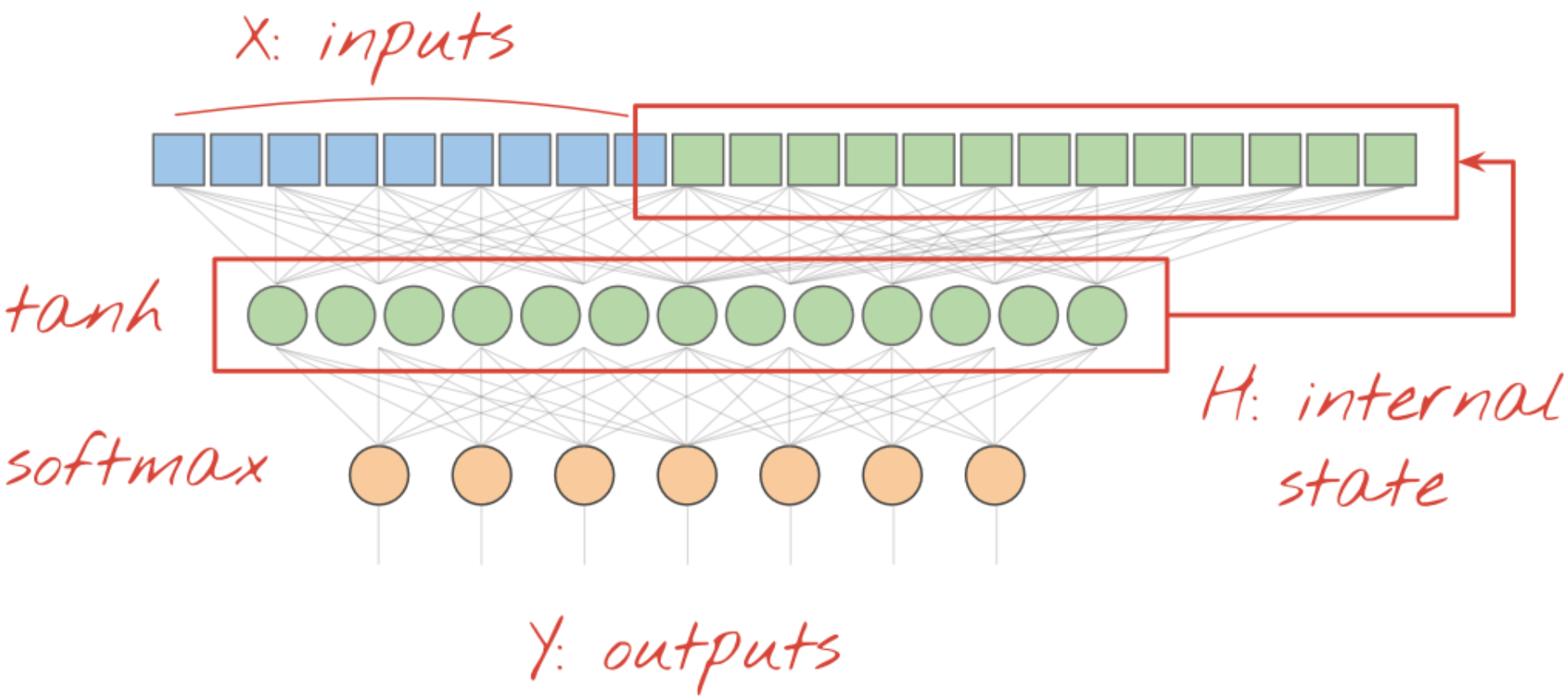
Inside RNN


Xt
H
RNN cell
Yt

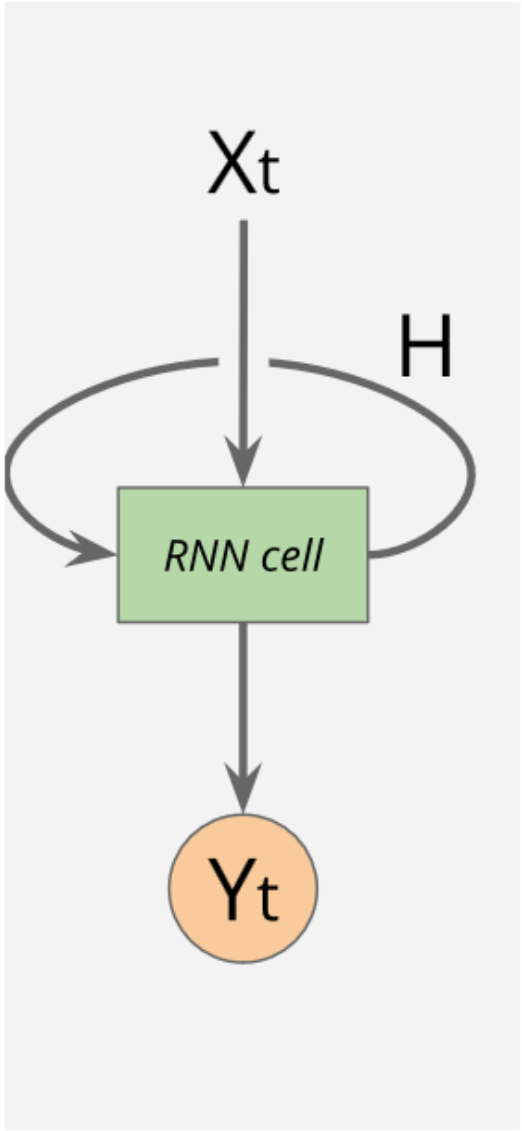

X: inputs
tanh
softmax
H: internal state
Y: outputs

Inside RNN


Xt
H
RNN cell
Yt


X: inputs
tanh
softmax
H: internal state
Y: outputs

Inside RNN

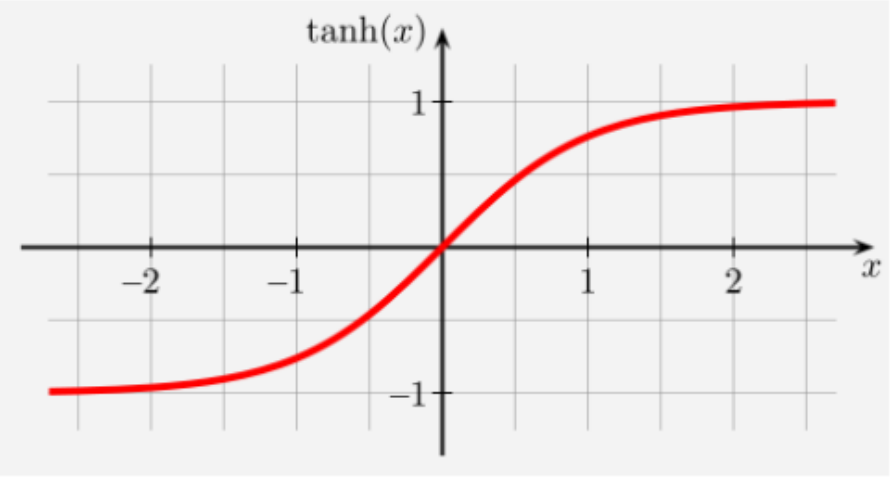


concatenation

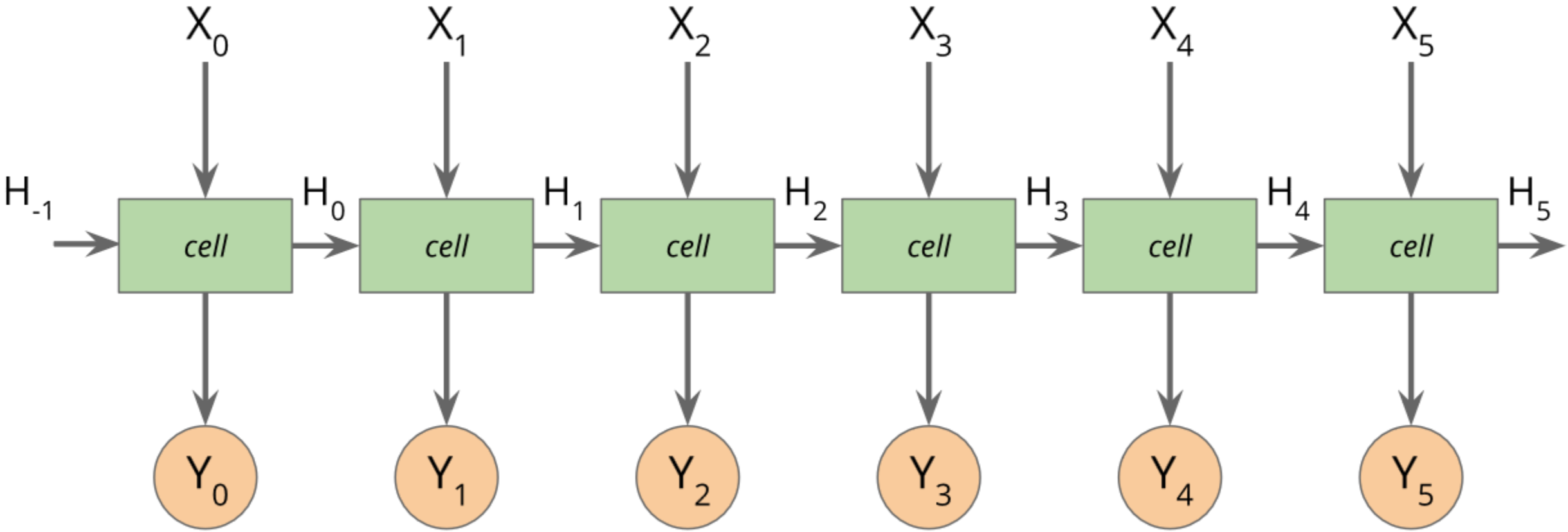
$$X = X_t \mid H_{t-1}$$

$$H_t = \tanh(X.W_H + b_H)$$

$$Y_t = \text{softmax}(H_t.W + b)$$



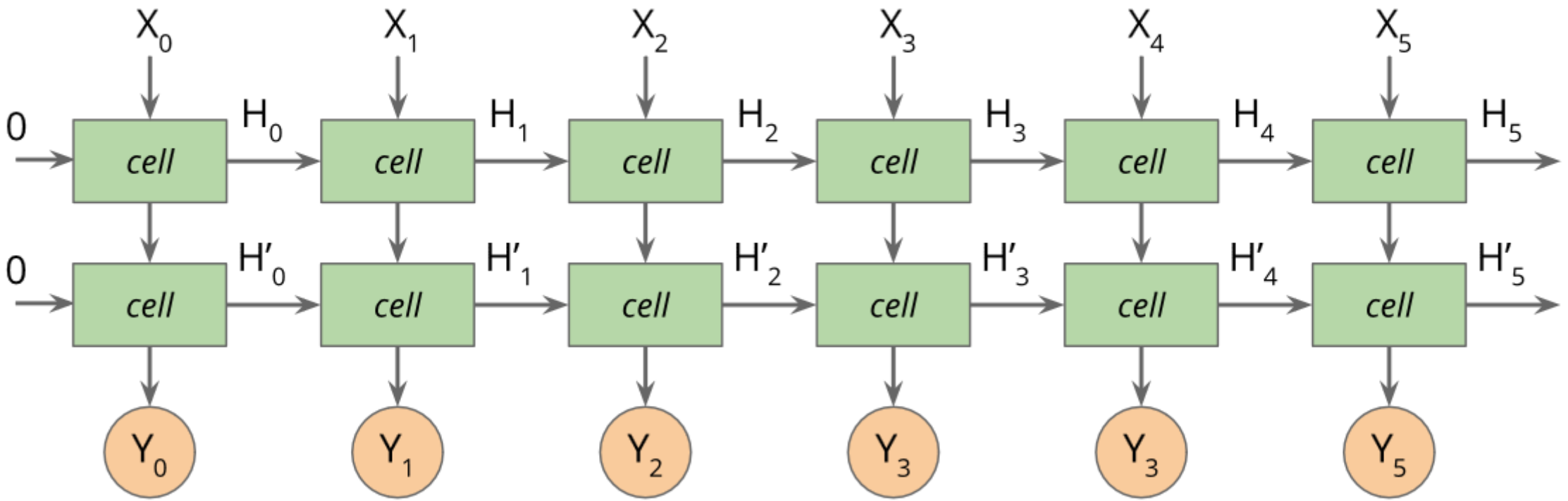
TRAINING RNN



The same weights and biases shared across iterations

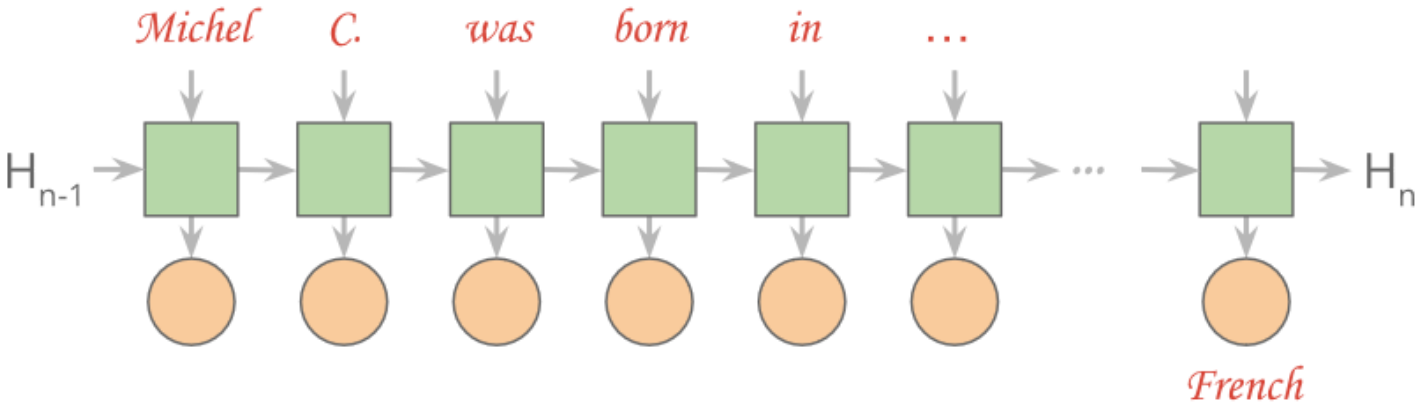
TRAINING RNN

L: number of layers



Long context problem

Michel C. was born in Paris, France. He is married and has three children. He received a M.S. in neurosciences from the University Pierre & Marie Curie and the Ecole Normale Supérieure in 1987, and and then spent most of his career in Switzerland, at the Ecole Polytechnique de Lausanne. He specialized in child and adolescent psychiatry and his first field of research was severe mood disorders in adolescent, topic of his PhD in neurosciences (2002). His mother tongue is ?????

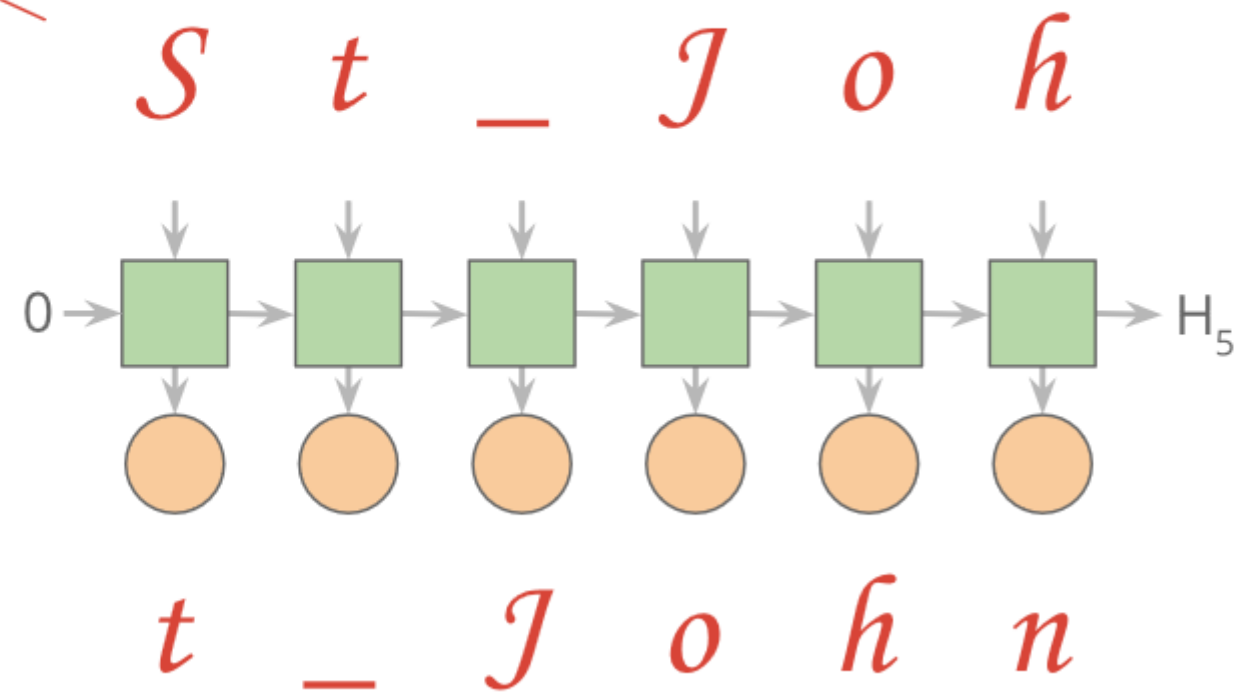


Short context → English, German, Russian, French ...

Long context → Problems...

NLP and RNN

Characters,
one-hot encoded

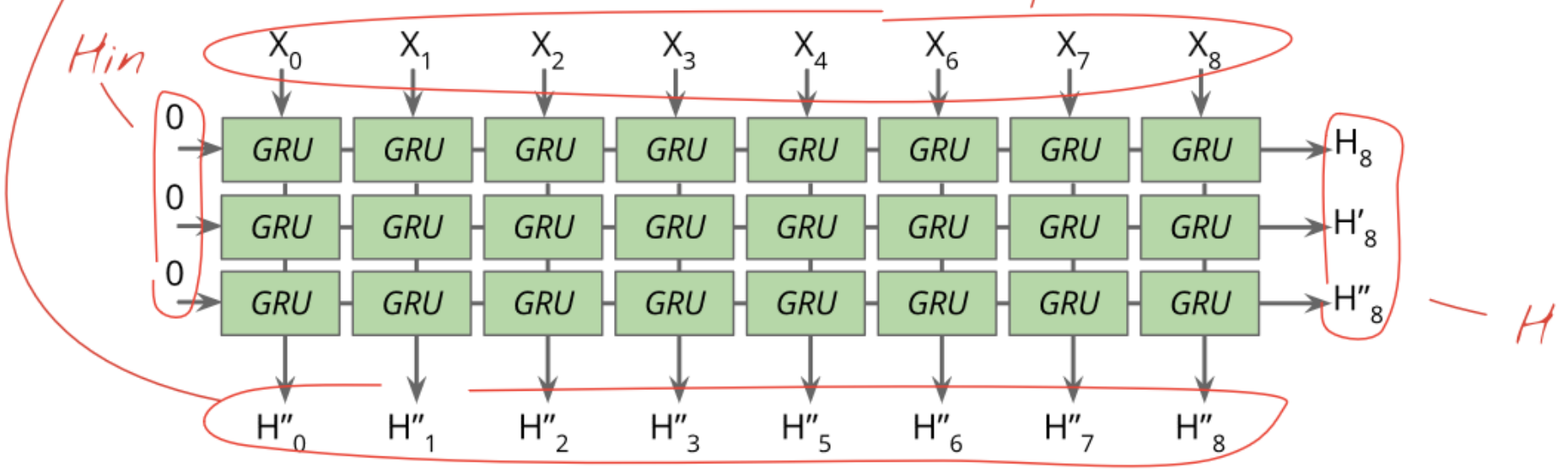


character-based

NLP and RNN

```
cell  = tf.nn.rnn_cell.GRUCell(CELLSIZE)
mcell = tf.nn.rnn_cell.MultiRNNCell([cell]*NLAYERS, state_is_tuple=False)
Hr, H = tf.nn.dynamic_rnn(mcell, X, initial_state=Hin)
```

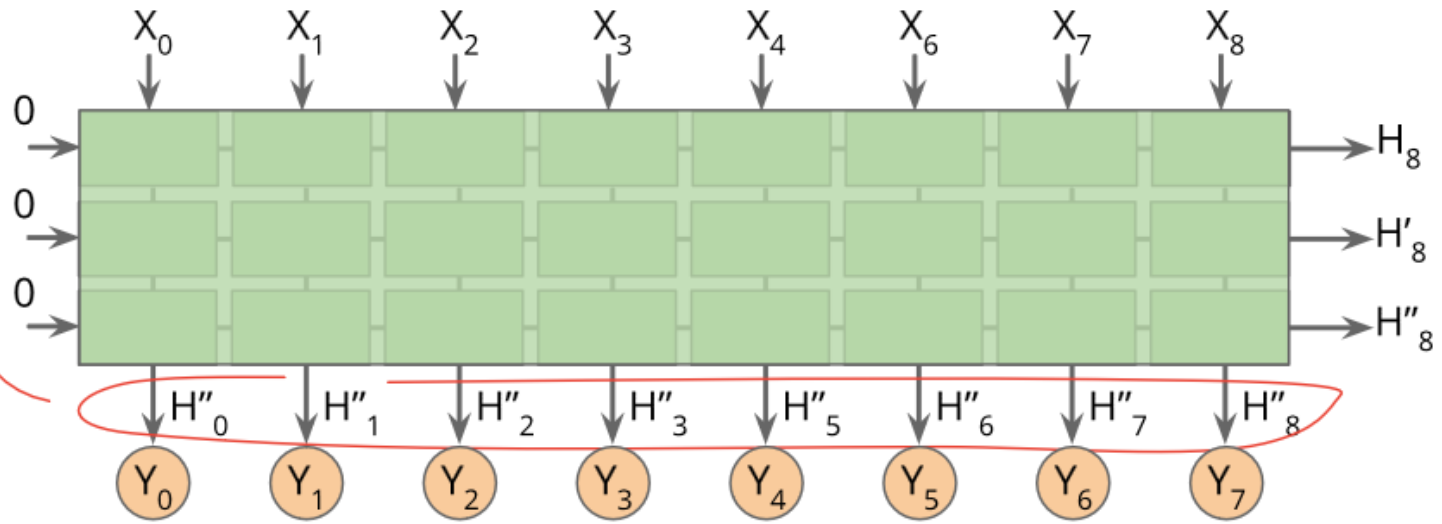
defines weights and biases internally




```
ALPHASIZE =  98
CELLSIZE  = 512
NLAYERS   =   3
SEQLEN    =  30
```

NLP and RNN

```
# Hr                                                    [ BATCHSIZE, SEQLEN, CELLSIZE ]
Hf = tf.reshape(Hr, [-1, CELLSIZE])                     [ BATCHSIZE x SEQLEN, CELLSIZE ]
Ylogits = layers.linear(Hf, ALPHASIZE)                  [ BATCHSIZE x SEQLEN, ALPHASIZE ]
Y = tf.nn.softmax(Ylogits)                              [ BATCHSIZE x SEQLEN, ALPHASIZE ]
```



Tip: handle sequence and batch elements the same

```
loss = tf.nn.softmax_cross_entropy_with_logits(Ylogits, Y_)
```

```
ALPHASIZE =  98
 CELLSIZE = 512
  NLAYERS =   3
   SEQLEN =  30
```

PYTHON CODE

0.03
epochs

```
diassts_=  =tlns==eti.s=tessn_((
   sie_s_nts_ens=  dondtnenroe dnar  taonte
srst anttntoilonttiteaen

detrtstinsenoaolsesnesoairt(

   arssserleeeerltrdlesssoeeslslrlslie(e
drnnaleeretteaelreesioe niennoarens
dssnstssaorns sreeoeslrteasntotnnai(ar
dsopelntederlalesdanserl
lts(sitae(e)
```

PYTHON CODE

0.1
epochs

Python
Keywords

```
with
self.essors_sigeater(output_dits_allss,
                                    self._train.
for sampated to than ubtexsormations.

expeddions = np.randim(natched_collection,
ranger, mang_ops, samplering)

def assestErrorume_gens(assignex) as
and(sampled_veases):
eved.
```

PYTHON CODE

0.4 epochs

```
def testGiddenSelfBeShareMecress(self):
    with self.test_session() as sess:
        tat = tf.contrib.matrix.cast_column_variable([1, 1], [0, 1, 1], [1, 7]],
        [[1, 1, 1]].file(file, line_state_will_file))
        with self.test_session():
            self.assertAllEqual(1, 1.ex6)
        self.assertEqual(output_graph_def is_output_tensors_op(
            tf.pro_context_name.sqrt(sess)

def test_shape(self):
    res = values=value_rns[0].eval())

    def tempDimpleSeriesGredicsIothasedWouthAverageData(self):
        self._testDirector(self):
        self._test_inv3_size = 5
    with tf.train.ConvolutioBailLors_startswith("save_dir_context.PutIsprint().eval())
        return tf.contrib.learn.RUCISLCCS:

# Check the orfloating so that the nimesting object mumputable othersifier.
# dense_keys.tokens_prefix/statch_size of the input1 tensors.

@property
```

Correct use of colons:

Wrong ([]) nesting

Hallucinated function names

PYTHON CODE

12 epochs

```
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in [0.1, 2.0, 3.0]]
```

Recites Apache license

Tensorflow tips!

```
def __init__(self, expected):
 return np.array([[0, 0, 0], [0, 0, 0]])
 self.assertAllEqual(tf.placeholder(tf.float32, shape=(3, 3)),(shape, prior.pack(),
                                                               tf.float32))

 for keys in tensor_list:
   return np.array([[0, 0, 0]]).astype(np.float32)

 # Check that we have both scalar tensor for being invalid to a vector of 1 indicating
 # the total loss of the same shape as the shape of the tensor.
 sharded_weights = [[0.0, 1.0]]
 # Create the string op to apply gradient terms that also batch.
 # The original any operation as a code when we should alw infer to the session case.
```

Correct triple ([]) nesting