Recurrent Neural Networks

Jean-Martin Albert

August 27, 2017

Training RNN's

The Basic RNN Cell

The Basic RNN Cell (code)

```
def basic_rnn_cell(input_tensor, state_tensor, output
  input_dimension = input_tensor.get_shape()[1]
  state_dimension = input_tensor.get_shape()[1]
  A_u = tf.Variable(shape=[input_dimension, output_di
  B_u = tf.Variable(shape=[state_dimension, output_di
  A_v = tf.Variable(shape=[input_dimension, state_dim
  B_v = tf.Variable(shape=[state_dimension, state_dim
  output_tensor = tf.relu(tf.matmul(input_tensor, A_u
                                    tf.matmul(state_tensor, B
  new_state_tensor = tf.tanh(tf.matmul(input_tensor,
                                       tf.matmul(state_tens
  return output_tensor, new_state_tensor
```

The LSTM Cell

The LSTM Cell (code)

```
def lstm_gate(input_tensor, previous_output, port_op)
  A = tf.Variable(shape=[N, L])
  B = tf.Variable(shape=[L, L])
  b = tf.Variable(shape=[L, L])
  x = tf.matmul(input_tensor, A)+ tf.matmul(previous_o
  return post_op(x)

def lstm_cell(input_tensor, output, state):
  F = lstm_gate(input_tensor, output, tf.sigmoid)
  I = lstm_gate(input_tensor, output, tf.sigmoid)
  O = lstm_gate(input_tensor, output, tf.sigmoid)
  S = lstm_gate(input_tensorm output, tf.tanh)
  new_state = tf.mul(output, F) + tf.mul(I, S)
  output = tf.mul(O, tf.tanh(new_state))
  return output, new_state
```

The Basic GRU Cell

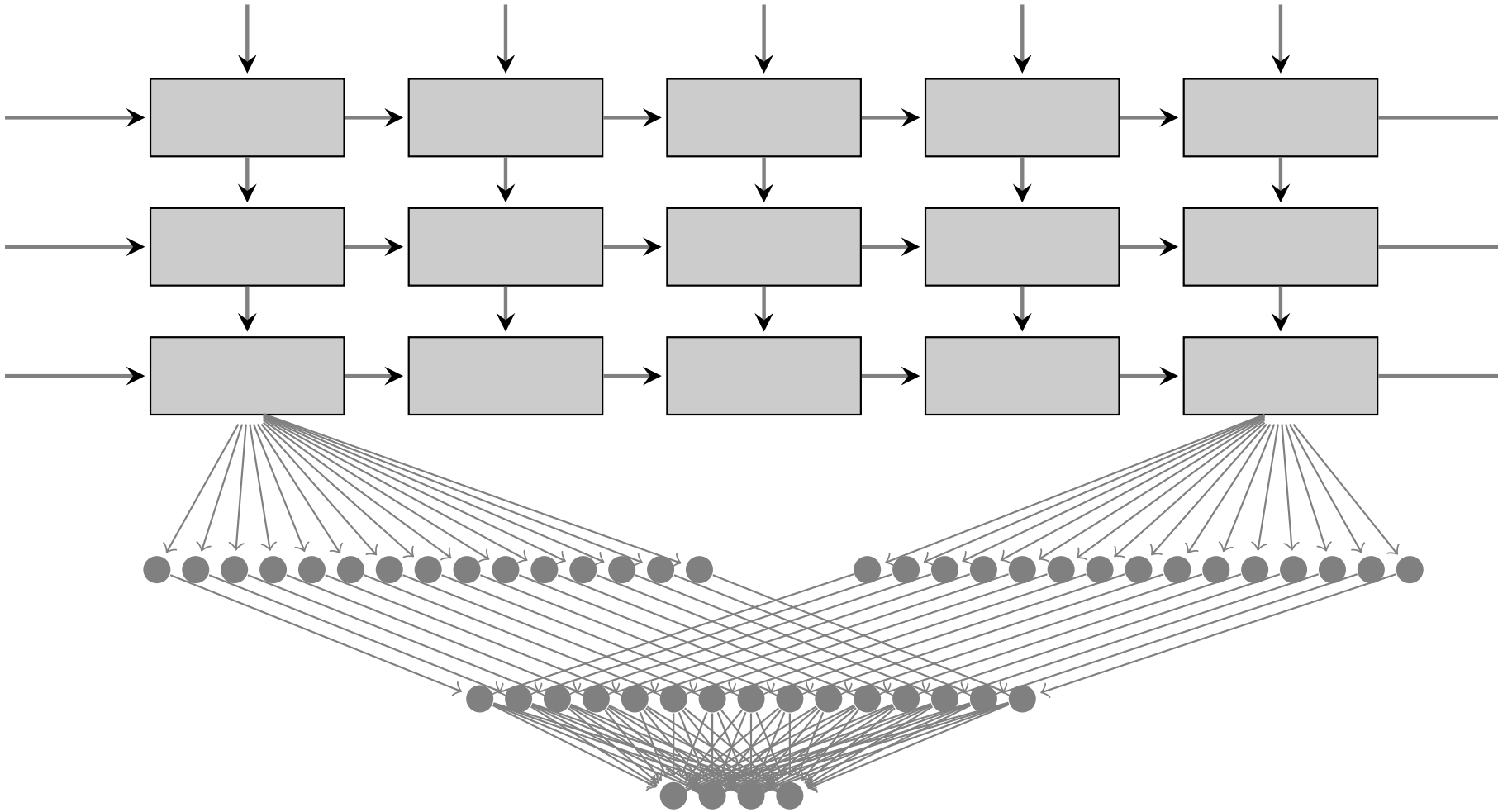
The GRU Cell (code)

```
def gru_gate(input_tensor, previous_output, port_op):
  A = tf.Variable(shape=[N, L])
  B = tf.Variable(shape=[L, L])
  b = tf.Variable(shape=[L, L])
  x = tf.matmul(input_tensor, A)+ tf.matmul(previous_o
  return post_op(x)

def gru_cell(input_tensor, output, state):
  U = gru_gate(input_tensor, output, tf.sigmoid)
  R = gru_gate(input_tensor, output, tf.sigmoid)
  O = gru_gate(input, tf.mul(R, output))
  return tf.mul(R, output) + tf.mul((1-R)O)
```

A many-to-one example

A many-to-one example (model architecture)

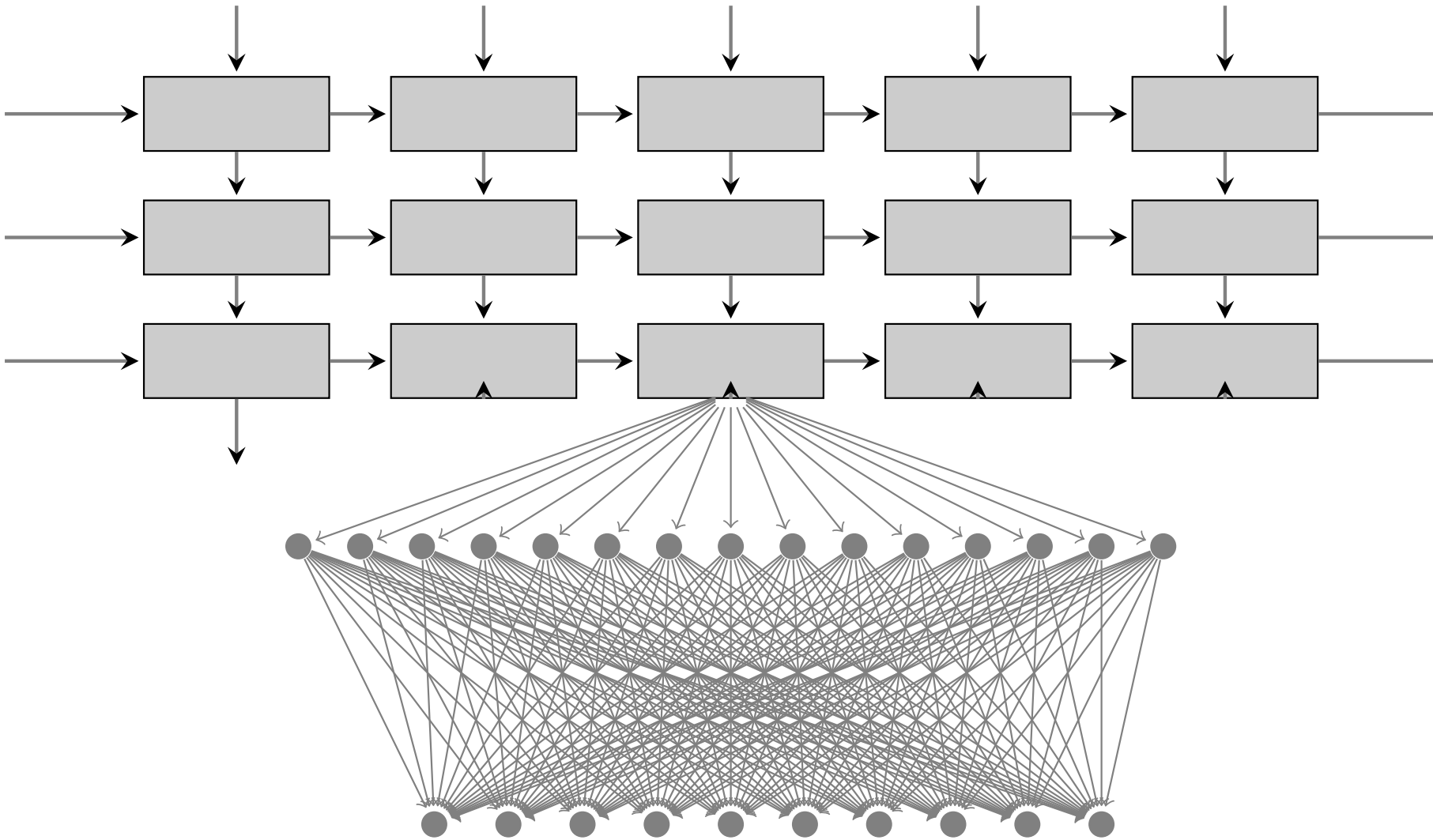


A many-to-one example (code)

```
SEQ_LENGTH = 256
E_DIM = 128
STATE_DIM = 512
NUM_CLASSES = 4
def inference():
    model_input = tf.placeholder('uint8', shape=[None
    _ = tf.one_hot(Globals.model_input, depth=E_DIM,
    _ = tf.reshape(_, [-1, SEQ_LENGTH, E_DIM])
    fw = multi_layer_rnn(N_LAYERS, STATE_DIM)
    bw = multi_layer_rnn(N_LAYERS, STATE_DIM)
    output, _ = tf.nn.bidirectional_dynamic_rnn(fw, bw
    fw_output = tf.reshape(output[0][:, -1:], [-1, ST
    bw_output = tf.reshape(output[1][:, :1], [-1, STA
    f = project(fw_output, E_DIM)
    b = project(bw_output, E_DIM)
    e = tf.add(f, b)
    Globals.model_output = project(e, NUM_CLASSES)
    Globals.prediction = tf.cast(tf.argmax(Globals.mo
    return Globals.model_input, Globals.model_output
```

A many-to-many example

A many-to-many example (model architecture)



A many-to-one example (code)

```
SEQ_LENGTH = 256
E_DIM = 128
STATE_DIM = 512
N_LAYERS = 3

def inference():
    model_input = tf.placeholder('uint8', shape=[None
    _ = tf.one_hot(Globals.model_input, depth=E_DIM, 
    encode = multi_layer_rnn(N_LAYERS, STATE_DIM)
    state_tuple = tuple(tf.unstack(Globals.initial_sta
    output, state = tf.nn.dynamic_rnn(encode, _,
                                      dtype=tf.float32
                                      initial_state=st

    output = tf.reshape(output, [-1, STATE_DIM])
    output = project(output, E_DIM)
    out = tf.cast(tf.argmax(output, 1), tf.uint8)
    out = tf.reshape(out, [-1, SEQ_LENGTH])
    Globals.generated_sequence = out
    Globals.generated_characters = tf.nn.softmax(outpu
    Globals.model_output = output
    Globals.state = state
```

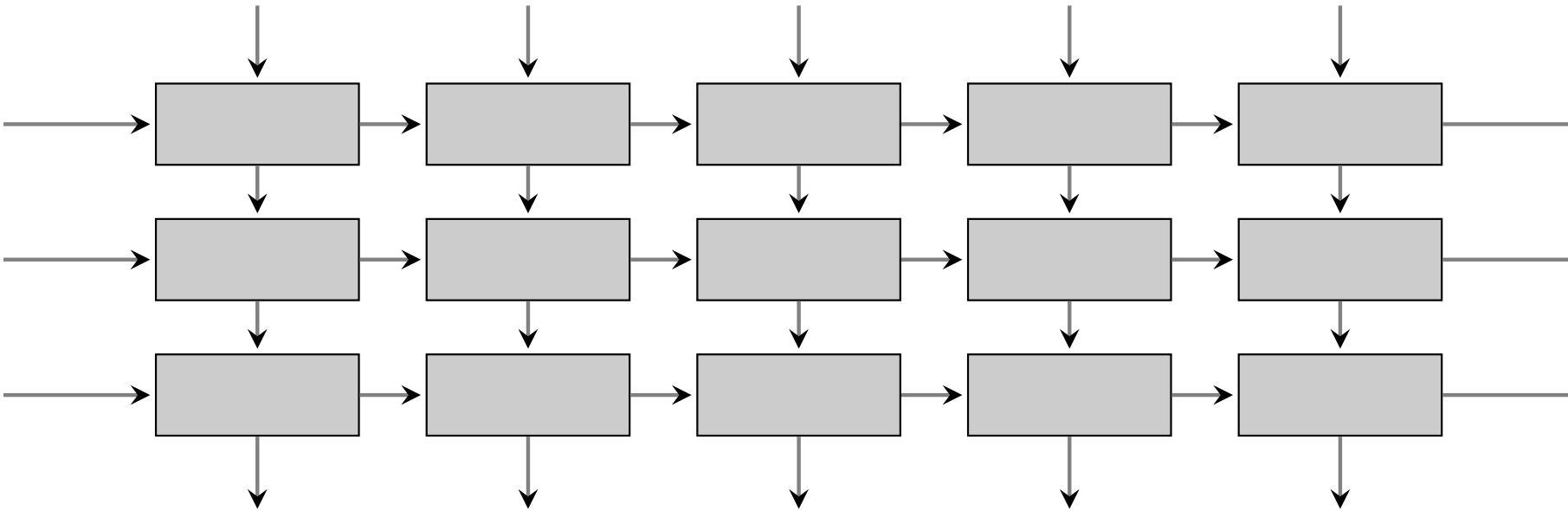
A many-to-many example (text generating)

```
def generate_text(length, session=None):
    generated_text = ''
    character = [[ord(' ')]]
    istate = np.zeros([N_LAYERS, 1, STATE_DIM])
    while len(generated_text) < length:
        feed_dict = {Globals.model_input: character,
                     Globals.initial_state: istate}
        next_char, state = session.run([Globals.genera
                                        Globals.state]
                                       feed_dict=feed_
        next_char = np.asarray(next_char).astype('floa
        next_char = next_char / next_char.sum()
        op = np.random.multinomial
        next_char_id = op(1, next_char.squeeze(), 1).a
        next_char_id = next_char_id if chr(next_char_i
                           string.printable else ord(" 
        generated_text += chr(next_char_id)
        character = [[next_char_id]]
        istate = state
    return generated_text
```

Many-to-one-to-many example

- A recurrent neural network that can sort sequences
- Two parts: an encoder, and a decoder
- The encoder encodes sequences into fixed length vectors
- The decoder transforms this vector into a sorted list of numbers.

Many-to-one-to-many example



Many-to-one-to-many example code

```
SEQ_LENGTH = 256
E_DIM = 128
STATE_DIM = 512
N_LAYERS = 4

def inference():
    model_input = tf.placeholder('uint8',
                                 shape=[None, SEQ_
    _ = tf.one_hot(model_input, depth=E_DIM, axis=
    _ = tf.reshape(_, [-1, SEQ_LENGTH, E_DIM])
    encode = multi_layer_rnn(N_LAYERS, STATE_DIM)
    OP = tf.nn.dynamic_rnn
    encoded_input, state = OP(encode,_,dtype=tf.fl
    Globals.encoder_output = state
    with tf.variable_scope('decoder'):
        training_decoder_input = tf.zeros_like(Glo
        _ = tf.one_hot(training_decoder_input, dep
        _ = tf.reshape(_, [-1, SEQ_LENGTH, E_DIM])
        decode = multi_layer_rnn(N_LAYERS, STATE_D
        decoded_output, state = tf.nn.dynamic_rnn(
```