COSC4337_RNN(Bidirectional)

[1]:
```
# Bi-directional Recurrent Neural Network.

#A Bi-directional Recurrent Neural Network (LSTM) implementation example using
#TensorFlow library. This example is using the MNIST database of handwritten
#digits (http://yann.lecun.com/exdb/mnist/)

from __future__ import print_function

import tensorflow as tf
from tensorflow.contrib import rnn
import numpy as np

# Import MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

'''
To classify images using a bidirectional recurrent neural network, we consider
every image row as a sequence of pixels. Because MNIST image shape is 28*28px,
we will then handle 28 sequences of 28 steps for every sample.
'''

# Training Parameters
learning_rate = 0.001
training_steps = 10000
batch_size = 128
display_step = 200

# Network Parameters
num_input = 28 # MNIST data input (img shape: 28*28)
timesteps = 28 # timesteps
num_hidden = 128 # hidden layer num of features
num_classes = 10 # MNIST total classes (0-9 digits)

# tf Graph input
X = tf.placeholder("float", [None, timesteps, num_input])
Y = tf.placeholder("float", [None, num_classes])
```

```
# Define weights
weights = {
    # Hidden layer weights => 2*n_hidden because of forward + backward cells
    'out': tf.Variable(tf.random_normal([2*num_hidden, num_classes]))
}
biases = {
    'out': tf.Variable(tf.random_normal([num_classes]))
}


def BiRNN(x, weights, biases):

    # Prepare data shape to match `rnn` function requirements
    # Current data input shape: (batch_size, timesteps, n_input)
    # Required shape: 'timesteps' tensors list of shape (batch_size, num_input)

    # Unstack to get a list of 'timesteps' tensors of shape (batch_size,␣
 ↪num_input)
    x = tf.unstack(x, timesteps, 1)

    # Define lstm cells with tensorflow
    # Forward direction cell
    lstm_fw_cell = rnn.BasicLSTMCell(num_hidden, forget_bias=1.0)
    # Backward direction cell
    lstm_bw_cell = rnn.BasicLSTMCell(num_hidden, forget_bias=1.0)

    # Get lstm cell output
    try:
        outputs, _, _ = rnn.static_bidirectional_rnn(lstm_fw_cell,␣
 ↪lstm_bw_cell, x,
                                             dtype=tf.float32)
    except Exception: # Old TensorFlow version only returns outputs not states
        outputs = rnn.static_bidirectional_rnn(lstm_fw_cell, lstm_bw_cell, x,
                                       dtype=tf.float32)

    # Linear activation, using rnn inner loop last output
    return tf.matmul(outputs[-1], weights['out']) + biases['out']

logits = BiRNN(X, weights, biases)
prediction = tf.nn.softmax(logits)

# Define loss and optimizer
loss_op = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(
    logits=logits, labels=Y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=learning_rate)
train_op = optimizer.minimize(loss_op)
```

```
# Evaluate model (with test logits, for dropout to be disabled)
correct_pred = tf.equal(tf.argmax(prediction, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()

# Start training
with tf.Session() as sess:

    # Run the initializer
    sess.run(init)

    for step in range(1, training_steps+1):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        # Reshape data to get 28 seq of 28 elements
        batch_x = batch_x.reshape((batch_size, timesteps, num_input))
        # Run optimization op (backprop)
        sess.run(train_op, feed_dict={X: batch_x, Y: batch_y})
        if step % display_step == 0 or step == 1:
            # Calculate batch loss and accuracy
            loss, acc = sess.run([loss_op, accuracy], feed_dict={X: batch_x,
                                                                 Y: batch_y})
            print("Step " + str(step) + ", Minibatch Loss= " + \
                  "{:.4f}".format(loss) + ", Training Accuracy= " + \
                  "{:.3f}".format(acc))

    print("Optimization Finished!")

    # Calculate accuracy for 128 mnist test images
    test_len = 128
    test_data = mnist.test.images[:test_len].reshape((-1, timesteps, num_input))
    test_label = mnist.test.labels[:test_len]
    print("Testing Accuracy:", \
        sess.run(accuracy, feed_dict={X: test_data, Y: test_label}))
```

```
WARNING:tensorflow:From <ipython-input-1-50316c39f4de>:23: read_data_sets (from
tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:260:
maybe_download (from tensorflow.contrib.learn.python.learn.datasets.base) is
deprecated and will be removed in a future version.
Instructions for updating:
```

```
Please write your own downloading logic.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:262:
extract_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-images-idx3-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:267:
extract_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting /tmp/data/train-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:110:
dense_to_one_hot (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.one_hot on tensors.
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:290:
DataSet.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
WARNING:tensorflow:From <ipython-input-1-50316c39f4de>:68:
BasicLSTMCell.__init__ (from tensorflow.python.ops.rnn_cell_impl) is deprecated
and will be removed in a future version.
Instructions for updating:
This class is equivalent as tf.keras.layers.LSTMCell, and will be replaced by
that in Tensorflow 2.0.
WARNING:tensorflow:From <ipython-input-1-50316c39f4de>:75:
static_bidirectional_rnn (from tensorflow.python.ops.rnn) is deprecated and will
be removed in a future version.
Instructions for updating:
Please use `keras.layers.Bidirectional(keras.layers.RNN(cell, unroll=True))`,
which is equivalent to this API
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\python\ops\rnn.py:1610: static_rnn (from
tensorflow.python.ops.rnn) is deprecated and will be removed in a future
version.
Instructions for updating:
Please use `keras.layers.RNN(cell, unroll=True)`, which is equivalent to this
```

```
API
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\python\ops\rnn_cell_impl.py:735: Layer.add_variable
(from tensorflow.python.keras.engine.base_layer) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use `layer.add_weight` method instead.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\python\ops\rnn_cell_impl.py:739: calling Zeros.__init__
(from tensorflow.python.ops.init_ops) with dtype is deprecated and will be
removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
WARNING:tensorflow:From <ipython-input-1-50316c39f4de>:88:
softmax_cross_entropy_with_logits (from tensorflow.python.ops.nn_ops) is
deprecated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.

Step 1, Minibatch Loss= 2.7045, Training Accuracy= 0.086
Step 200, Minibatch Loss= 2.1934, Training Accuracy= 0.242
Step 400, Minibatch Loss= 1.9797, Training Accuracy= 0.383
Step 600, Minibatch Loss= 1.8861, Training Accuracy= 0.336
Step 800, Minibatch Loss= 1.7479, Training Accuracy= 0.477
Step 1000, Minibatch Loss= 1.6571, Training Accuracy= 0.406
Step 1200, Minibatch Loss= 1.5380, Training Accuracy= 0.477
Step 1400, Minibatch Loss= 1.4221, Training Accuracy= 0.555
Step 1600, Minibatch Loss= 1.4308, Training Accuracy= 0.539
Step 1800, Minibatch Loss= 1.2546, Training Accuracy= 0.617
Step 2000, Minibatch Loss= 1.3031, Training Accuracy= 0.570
Step 2200, Minibatch Loss= 1.3055, Training Accuracy= 0.594
Step 2400, Minibatch Loss= 1.1822, Training Accuracy= 0.633
Step 2600, Minibatch Loss= 1.2128, Training Accuracy= 0.578
Step 2800, Minibatch Loss= 1.0383, Training Accuracy= 0.688
Step 3000, Minibatch Loss= 1.0445, Training Accuracy= 0.664
Step 3200, Minibatch Loss= 0.8434, Training Accuracy= 0.750
Step 3400, Minibatch Loss= 0.8758, Training Accuracy= 0.750
Step 3600, Minibatch Loss= 1.0009, Training Accuracy= 0.672
Step 3800, Minibatch Loss= 1.0180, Training Accuracy= 0.703
Step 4000, Minibatch Loss= 0.9698, Training Accuracy= 0.688
Step 4200, Minibatch Loss= 0.9274, Training Accuracy= 0.727
Step 4400, Minibatch Loss= 0.9543, Training Accuracy= 0.672
Step 4600, Minibatch Loss= 0.7526, Training Accuracy= 0.766
```

```
Step 4800, Minibatch Loss= 0.8189, Training Accuracy= 0.734
Step 5000, Minibatch Loss= 0.7277, Training Accuracy= 0.781
Step 5200, Minibatch Loss= 0.6788, Training Accuracy= 0.797
Step 5400, Minibatch Loss= 0.7726, Training Accuracy= 0.727
Step 5600, Minibatch Loss= 0.7228, Training Accuracy= 0.805
Step 5800, Minibatch Loss= 0.6554, Training Accuracy= 0.781
Step 6000, Minibatch Loss= 0.6252, Training Accuracy= 0.836
Step 6200, Minibatch Loss= 0.7421, Training Accuracy= 0.797
Step 6400, Minibatch Loss= 0.6767, Training Accuracy= 0.781
Step 6600, Minibatch Loss= 0.7394, Training Accuracy= 0.766
Step 6800, Minibatch Loss= 0.6370, Training Accuracy= 0.781
Step 7000, Minibatch Loss= 0.6883, Training Accuracy= 0.773
Step 7200, Minibatch Loss= 0.5601, Training Accuracy= 0.805
Step 7400, Minibatch Loss= 0.7177, Training Accuracy= 0.781
Step 7600, Minibatch Loss= 0.6282, Training Accuracy= 0.820
Step 7800, Minibatch Loss= 0.6140, Training Accuracy= 0.805
Step 8000, Minibatch Loss= 0.6752, Training Accuracy= 0.781
Step 8200, Minibatch Loss= 0.4360, Training Accuracy= 0.891
Step 8400, Minibatch Loss= 0.5190, Training Accuracy= 0.883
Step 8600, Minibatch Loss= 0.5665, Training Accuracy= 0.820
Step 8800, Minibatch Loss= 0.4193, Training Accuracy= 0.875
Step 9000, Minibatch Loss= 0.6339, Training Accuracy= 0.812
Step 9200, Minibatch Loss= 0.3394, Training Accuracy= 0.906
Step 9400, Minibatch Loss= 0.4839, Training Accuracy= 0.844
Step 9600, Minibatch Loss= 0.3176, Training Accuracy= 0.914
Step 9800, Minibatch Loss= 0.4112, Training Accuracy= 0.883
Step 10000, Minibatch Loss= 0.5333, Training Accuracy= 0.867
Optimization Finished!
Testing Accuracy: 0.8984375
```

[ ]: