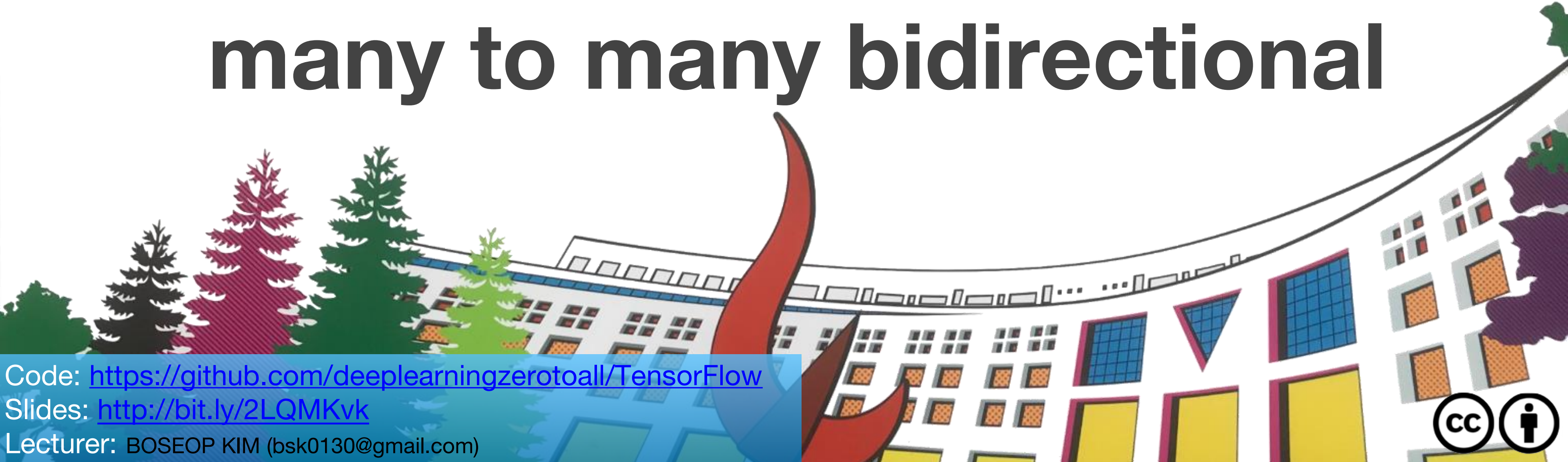
ML/DL for Everyone Season2

with TensorFlow

Lab 12-4
many to many bidirectional



Code: https://github.com/deeplearningzerotoall/TensorFlow

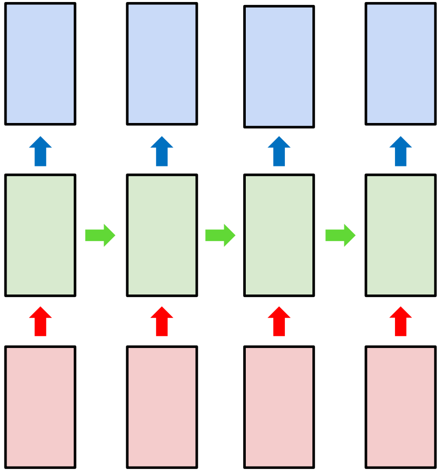
Slides: http://bit.ly/2LQMKvk

Lecturer: BOSEOP KIM (bsk0130@gmail.com)

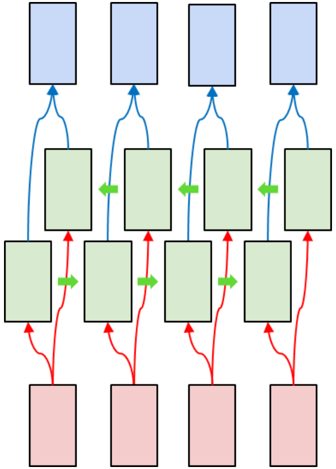
many to many bidirectional

- What is “bidirectional”?
- many to many bidirectional
- Example : part of speech tagging
    - Preparing dataset
    - Creating and training model
    - Checking performance

What is “bidirectional”?



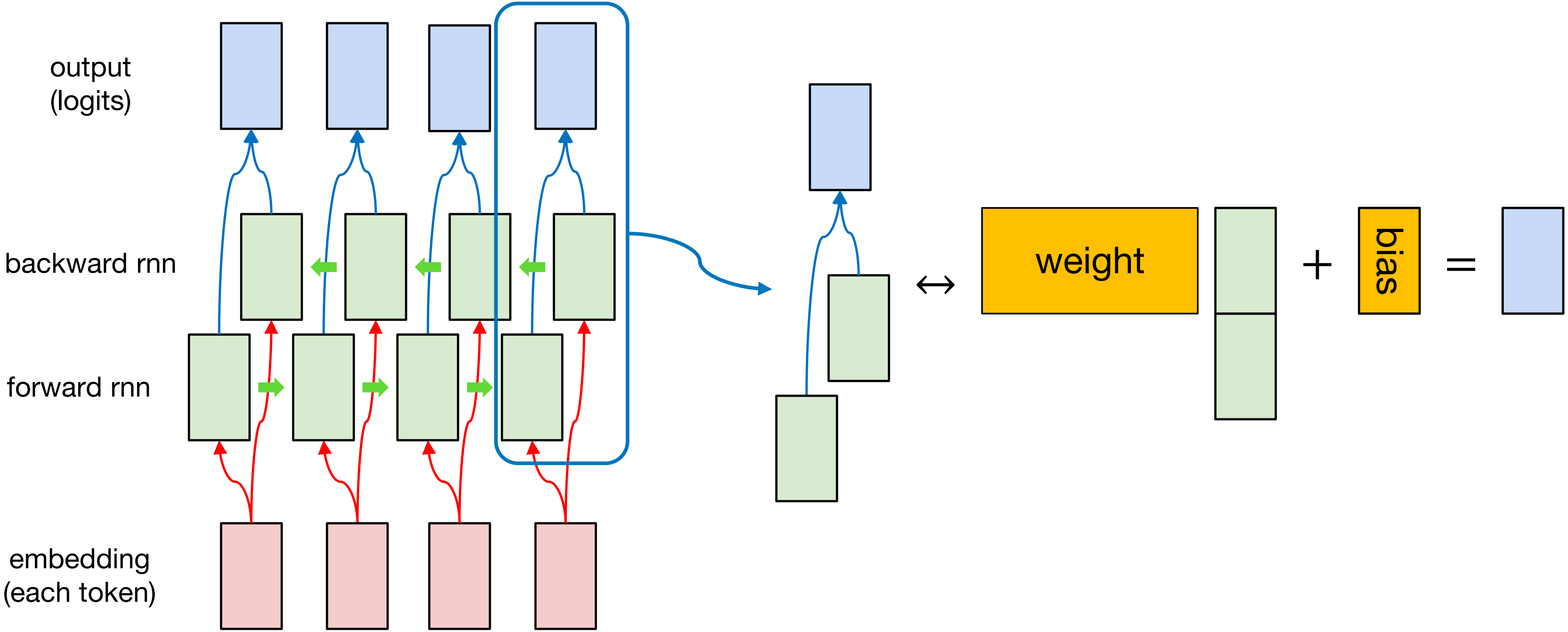
VS



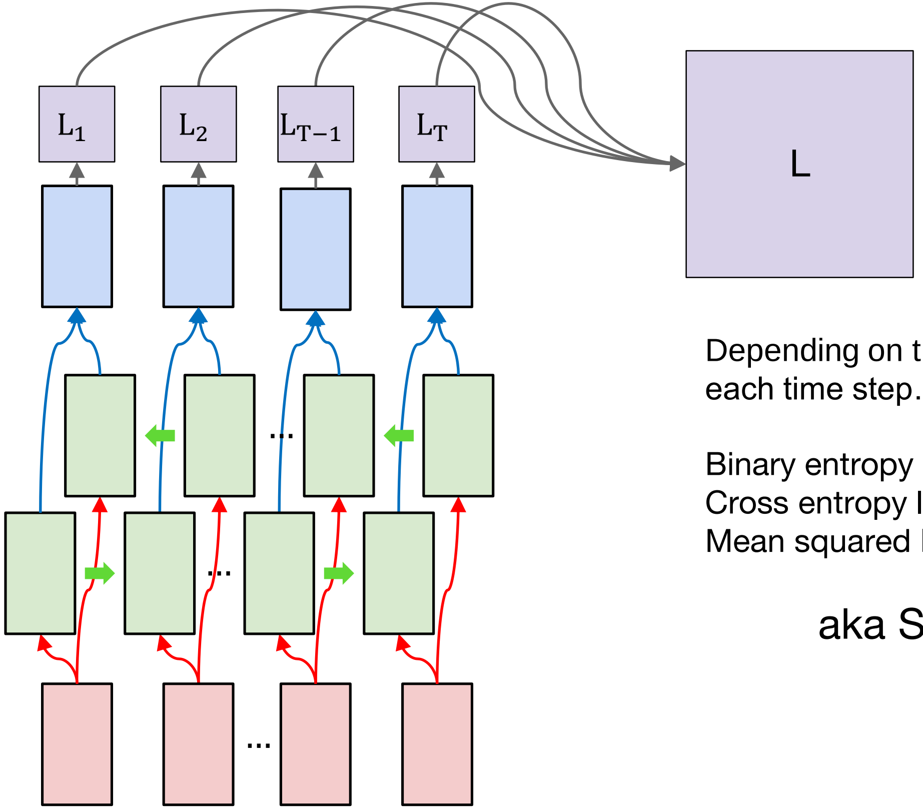
There is imbalance in the amount of information seen by the hidden states at different time steps.

There is balance in the amount of information seen by the hidden states at different time steps.

What is “bidirectional”?



many to many bidirectional



Depending on the task with regard to each time step…

Binary entropy loss (binary classifier)
Cross entropy loss (softmax classifier)
Mean squared loss (regression)

aka Sequence loss!

Example : part of speech tagging

Preparing dataset

```
# example data
sentences = [['I', 'feel', 'hungry'],
    ['tensorflow', 'is', 'very', 'difficult'],
    ['tensorflow', 'is', 'a', 'framework', 'for', 'deep', 'learning'],
    ['tensorflow', 'is', 'very', 'fast', 'changing']]
pos = [['pronoun', 'verb', 'adjective'],
    ['noun', 'verb', 'adverb', 'adjective'],
    ['noun', 'verb', 'determiner', 'noun', 'preposition', 'adjective', 'noun'],
    ['noun', 'verb', 'adverb', 'adjective', 'verb']]
```

```
# creating a token dictionary for word
word_list = sum(sentences, [])
word_list = sorted(set(word_list))
word_list = ['<pad>'] + word_list
word2idx = {word : idx for idx, word in enumerate(word_list)}
idx2word = {idx : word for idx, word in enumerate(word_list)}

print(word2idx)
print(idx2word)
print(len(idx2word))
```

```
{'<pad>': 0, 'I': 1, 'a': 2, 'changing': 3, 'deep': 4,
'difficult': 5, 'fast': 6, 'feel': 7, 'for': 8, 'framework':
9, 'hungry': 10, 'is': 11, 'learning': 12, 'tensorflow': 13,
'very': 14}
{0: '<pad>', 1: 'I', 2: 'a', 3: 'changing', 4: 'deep', 5:
'difficult', 6: 'fast', 7: 'feel', 8: 'for', 9: 'framework',
10: 'hungry', 11: 'is', 12: 'learning', 13: 'tensorflow', 14:
'very'}
15
```

```
# creating a token dictionary for part of speech
pos_list = sum(pos, [])
pos_list = sorted(set(pos_list))
pos_list = ['<pad>'] + pos_list
pos2idx = {pos : idx for idx, pos in enumerate(pos_list)}
idx2pos = {idx : pos for idx, pos in enumerate(pos_list)}

print(pos2idx)
print(idx2pos)
print(len(pos2idx))
```

```
{'<pad>': 0, 'adjective': 1, 'adverb': 2, 'determiner': 3,
'noun': 4, 'preposition': 5, 'pronoun': 6, 'verb': 7}
{0: '<pad>', 1: 'adjective', 2: 'adverb', 3: 'determiner',
4: 'noun', 5: 'preposition', 6: 'pronoun', 7: 'verb'}
8
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

Example : part of speech tagging

Preparing dataset

```
# converting sequence of tokens to sequence of indices
max_sequence = 10
x_data = list(map(lambda sentence : [word2idx.get(token) for token in sentence], sentences))
y_data = list(map(lambda sentence : [pos2idx.get(token) for token in sentence], pos))

# padding the sequence of indices
x_data = pad_sequences(sequences = x_data, maxlen = max_sequence, padding='post')
x_data_mask = ((x_data != 0) * 1).astype(np.float32)
x_data_len = list(map(lambda sentence : len(sentence), sentences))

y_data = pad_sequences(sequences = y_data, maxlen = max_sequence, padding='post')

# checking data
print(x_data, x_data_len)
print(x_data_mask)
print(y_data)
```

```
[[ 1  7 10  0  0  0  0  0  0  0]
 [13 11 14  5  0  0  0  0  0  0]
 [13 11  2  9  8  4 12  0  0  0]
 [13 11 14  6  3  0  0  0  0  0]] [3, 4, 7, 5]
[[1. 1. 1. 0. 0. 0. 0. 0. 0. 0.]

 [1. 1. 1. 1. 0. 0. 0. 0. 0. 0.]
 [1. 1. 1. 1. 1. 1. 1. 0. 0. 0.]
 [1. 1. 1. 1. 1. 0. 0. 0. 0. 0.]]
```

```
[[6 7 1 0 0 0 0 0 0 0]
 [4 7 2 1 0 0 0 0 0 0]
 [4 7 3 4 5 1 4 0 0 0]
 [4 7 2 1 7 0 0 0 0 0]]
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

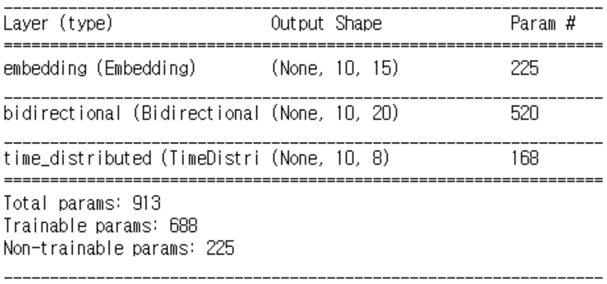
Example : part of speech tagging

Creating and training model

```
# creating bidirectional rnn for "many to many" sequence tagging
num_classes = len(pos2idx)
hidden_dim = 10

input_dim = len(word2idx)
output_dim = len(word2idx)
one_hot = np.eye(len(word2idx))

model = Sequential()
model.add(layers.InputLayer(input_shape=(max_sequence,)))
model.add(layers.Embedding(input_dim=input_dim, output_dim=output_dim, mask_zero=True,
                           trainable=False, input_length=max_sequence,
                           embeddings_initializer=keras.initializers.Constant(one_hot)))
model.add(layers.Bidirectional(keras.layers.SimpleRNN(units=hidden_dim, return_sequences=True)))
model.add(layers.TimeDistributed(keras.layers.Dense(units=num_classes)))
model.summary()
```



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 10, 15)            225
_________________________________________________________________
bidirectional (Bidirectional (None, 10, 20)            520
_________________________________________________________________
time_distributed (TimeDistri (None, 10, 8)             168
=================================================================
Total params: 913
Trainable params: 688
Non-trainable params: 225
_________________________________________________________________
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

Example : part of speech tagging

Creating and training model

```
# creating loss function
def loss_fn(model, x, y, x_len, max_sequence):
   masking = tf.sequence_mask(x_len, maxlen=max_sequence, dtype=tf.float32)
   valid_time_step = tf.cast(x_len,dtype=tf.float32)
   sequence_loss = tf.losses.sparse_softmax_cross_entropy(labels=y, logits=model(x),
                                                          reduction='none') * masking
   sequence_loss = tf.reduce_sum(sequence_loss, axis=-1) / valid_time_step
   sequence_loss = tf.reduce_mean(sequence_loss)
   return sequence_loss

# creating and optimizer
lr = 0.1
epochs = 30
batch_size = 2
opt = tf.train.AdamOptimizer(learning_rate = lr)

# generating data pipeline
tr_dataset = tf.data.Dataset.from_tensor_slices((x_data, y_data, x_data_len))
tr_dataset = tr_dataset.shuffle(buffer_size=4)
tr_dataset = tr_dataset.batch(batch_size = 2)

print(tr_dataset)
```

```
<BatchDataset shapes: ((?, 10), (?, 10), (?,)), types: (tf.int32, tf.int32,
tf.int32)>
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

Example : part of speech tagging

Creating and training model

```
# training
tr_loss_hist = []

for epoch in range(epochs):
   avg_tr_loss = 0
   tr_step = 0

   for x_mb, y_mb, x_mb_len in tr_dataset:
       with tf.GradientTape() as tape:
           tr_loss = loss_fn(model, x=x_mb, y=y_mb, x_len=x_mb_len, max_sequence=max_sequence)
       grads = tape.gradient(target=tr_loss, sources=model.variables)
       opt.apply_gradients(grads_and_vars=zip(grads, model.variables))
       avg_tr_loss += tr_loss
       tr_step += 1
   else:
       avg_tr_loss /= tr_step
       tr_loss_hist.append(avg_tr_loss)

   if (epoch + 1) % 5 == 0:
       print('epoch : {:3}, tr_loss : {:.3f}'.format(epoch + 1, avg_tr_loss))
```

```
epoch :   5, tr_loss : 0.052
epoch :  10, tr_loss : 0.002
epoch :  15, tr_loss : 0.000
epoch :  20, tr_loss : 0.000
epoch :  25, tr_loss : 0.000
epoch :  30, tr_loss : 0.000
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

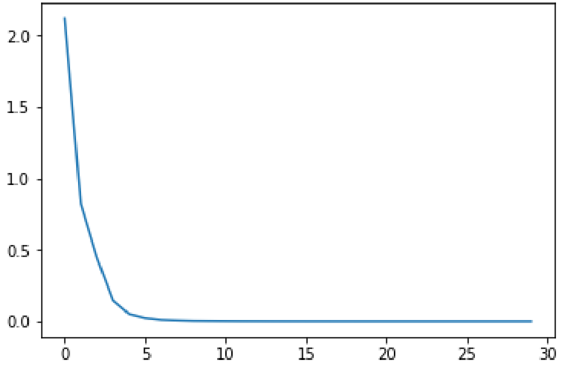
Example : part of speech tagging

Checking performance

```
yhat = model.predict(x_data)
yhat = np.argmax(yhat, axis=-1) * x_data_mask

pprint(list(map(lambda row : [idx2pos.get(elm) for elm in row],yhat.astype(np.int32).tolist())), width = 120)
pprint(pos)
```

```
[['pronoun', 'verb', 'adjective', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'],
['noun', 'verb', 'adverb', 'adjective', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>'],
['noun', 'verb', 'determiner', 'noun', 'preposition', 'adjective', 'noun', '<pad>', '<pad>',
'<pad>'],
['noun', 'verb', 'adverb', 'adjective', 'verb', '<pad>', '<pad>', '<pad>', '<pad>', '<pad>']]
[['pronoun', 'verb', 'adjective'],
['noun', 'verb', 'adverb', 'adjective'],
['noun', 'verb', 'determiner', 'noun', 'preposition', 'adjective', 'noun'],
['noun', 'verb', 'adverb', 'adjective', 'verb']]
```



https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-4-many-to-many-bidirectional-keras-eager.ipynb

What’s Next?

- sequence to sequence