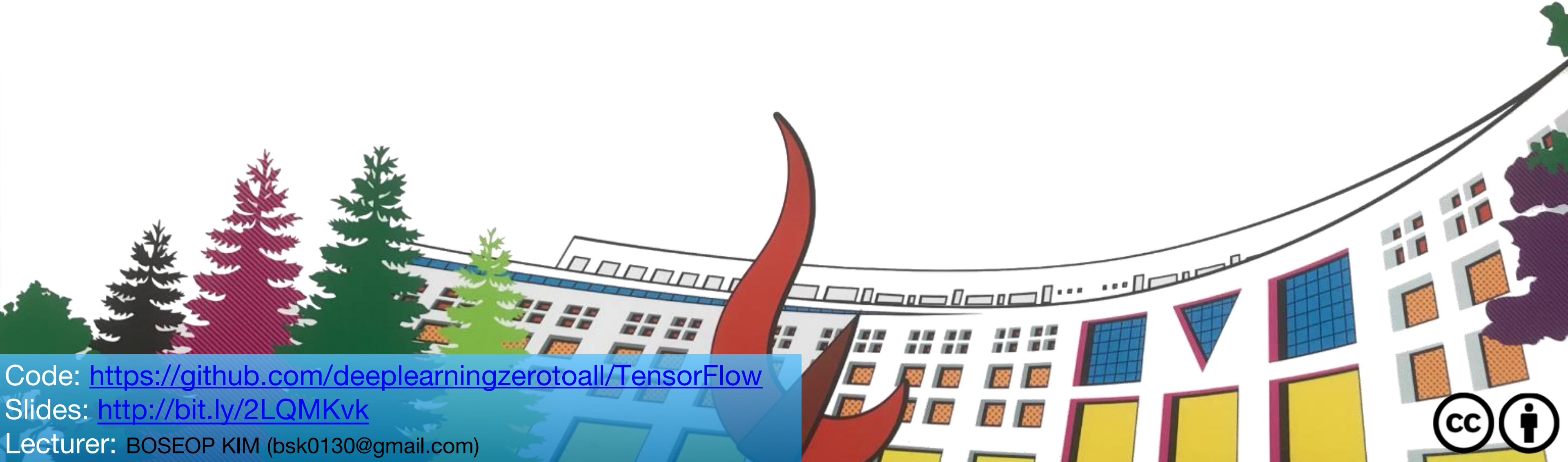
ML/DL for Everyone Season2

with TensorFlow

Lab 12-1 many to one

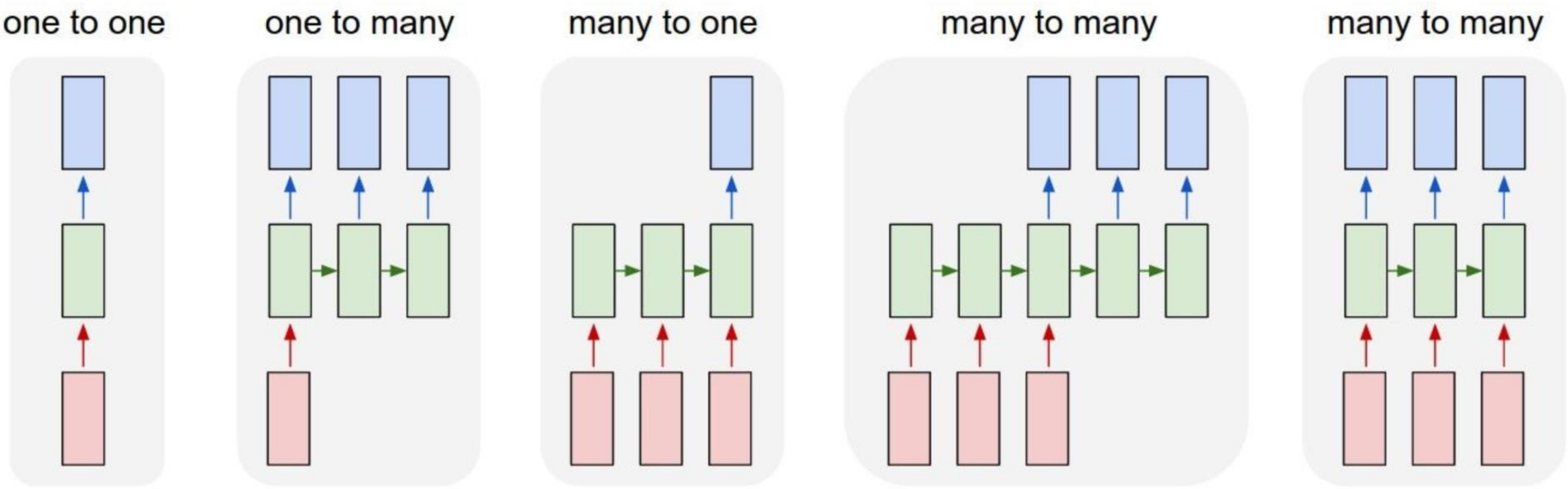


Code: https://github.com/deeplearningzerotoall/TensorFlow
Slides: http://bit.ly/2LQMKvk
Lecturer: BOSEOP KIM (bsk0130@gmail.com)

many to one

- Various usage of RNN
- What is “many to one”?
- Example : word sentiment classification
  - Preparing dataset
  - Creating and training model
  - Checking performance

Various usage of RNN



http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture10.pdf

What is “many to one”?

Sequence classification

eg. classify polarity of sentence

sequence : sentence, tokens : word

[‘This movie is good’] : 문법적으로 나눌 수 있는 기본적인 언어 단위

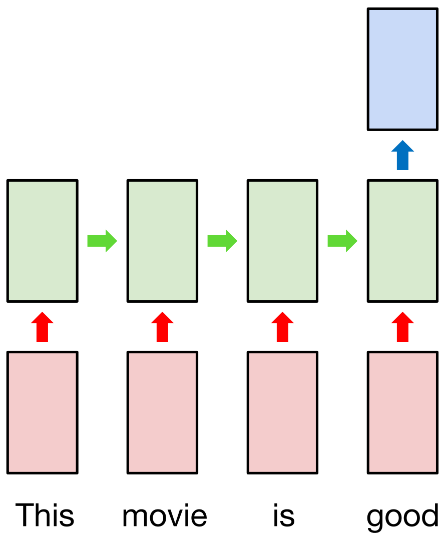
↓ *Tokenization*

[‘This’, ‘movie’, ‘is’, ‘good’]

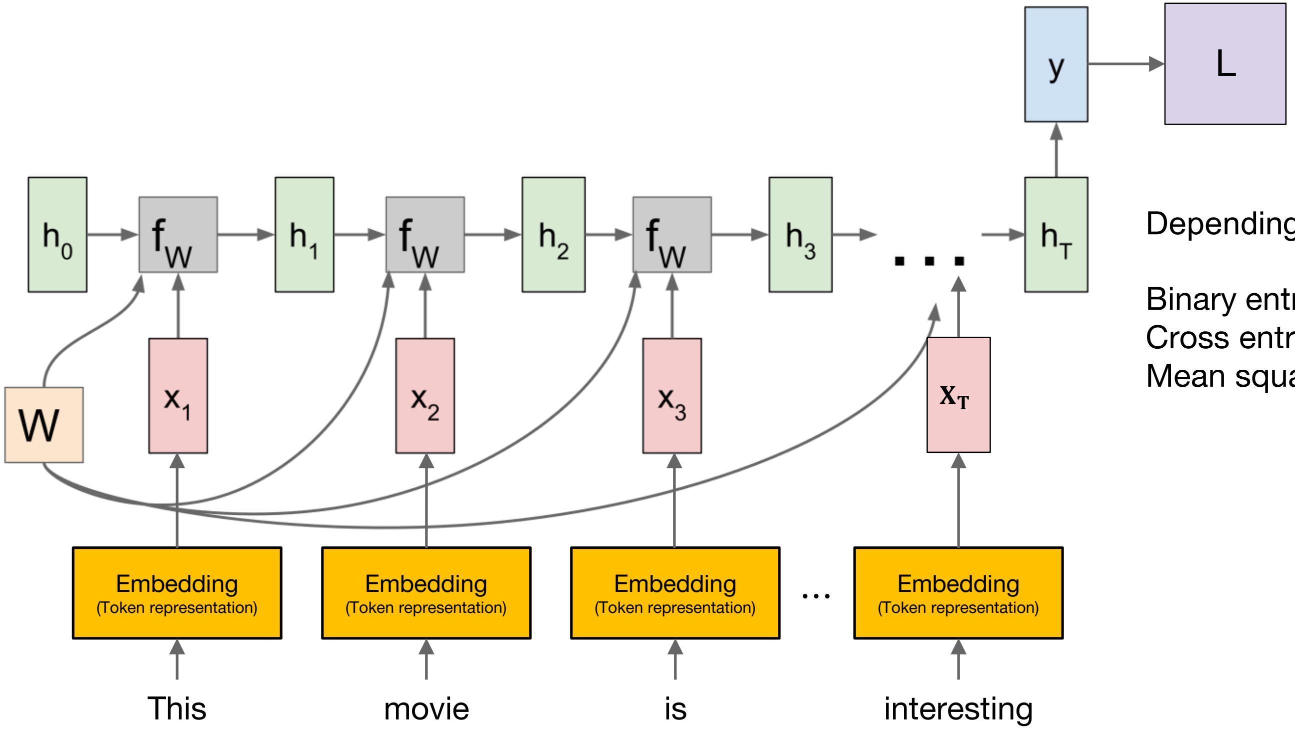
↓ *Classification*

Positive

Classification : Positive or negative?



What is “many to one”?



Depending on the task…

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

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture10.pdf

Example : word sentiment classification

Preparing dataset

```python
# example data
words = ['good', 'bad', 'worse', 'so good']
y_data = [1,0,0,1]

# creating a token dictionary
char_set = ['<pad>'] + sorted(list(set(''.join(words))))
idx2char = {idx : char for idx, char in enumerate(char_set)}
char2idx = {char : idx for idx, char in enumerate(char_set)}

print(char_set)
print(idx2char)
print(char2idx)
```

```
['<pad>', ' ', 'a', 'b', 'd', 'e', 'g', 'o', 'r', 's', 'w']
{0: '<pad>', 1: ' ', 2: 'a', 3: 'b', 4: 'd', 5: 'e', 6: 'g', 7: 'o', 8: 'r', 9: 's', 10: 'w'}
{'<pad>': 0, ' ': 1, 'a': 2, 'b': 3, 'd': 4, 'e': 5, 'g': 6, 'o': 7, 'r': 8, 's': 9, 'w': 10}
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

Example : word sentiment classification

Preparing dataset

```
# converting sequence of tokens to sequence of indices
x_data = list(map(lambda word : [char2idx.get(char) for char in word], words))
x_data_len = list(map(lambda word : len(word), x_data))

print(x_data)
print(x_data_len)

[[6, 7, 7, 4], [3, 2, 4], [10, 7, 8, 9, 5], [9, 7, 1, 6, 7, 7, 4]]
[4, 3, 5, 7]

# padding the sequence of indices
max_sequence = 10
x_data = pad_sequences(sequences = x_data, maxlen = max_sequence,
                       padding = 'post', truncating = 'post')

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

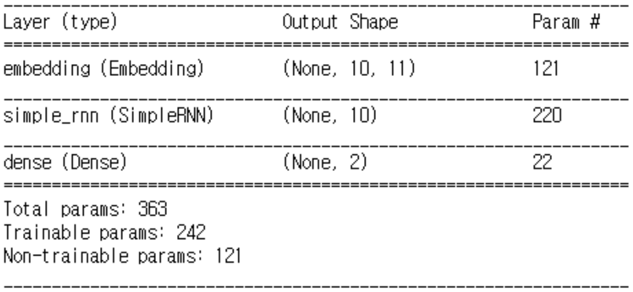
```
[[ 6  7  7  4  0  0  0  0  0  0]
 [ 3  2  4  0  0  0  0  0  0  0]
 [10  7  8  9  5  0  0  0  0  0]
 [ 9  7  1  6  7  7  4  0  0  0]]
[4, 3, 5, 7]
[1, 0, 0, 1]
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

Example : word sentiment classification

Creating and training model

```
# creating simple rnn for "many to one" classification
input_dim = len(char2idx)
output_dim = len(char2idx)
one_hot = np.eye(len(char2idx))
hidden_size = 10
num_classes = 2

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



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 10, 11)            121
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 10)                220
_________________________________________________________________
dense (Dense)                (None, 2)                 22
=================================================================
Total params: 363
Trainable params: 242
Non-trainable params: 121
_________________________________________________________________
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

Example : word sentiment classification

Creating and training model

```
# creating loss function
def loss_fn(model, x, y):
   return tf.losses.sparse_softmax_cross_entropy(labels=y, logits=model(x))

# creating an optimizer
lr = .01
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))
tr_dataset = tr_dataset.shuffle(buffer_size = 4)
tr_dataset = tr_dataset.batch(batch_size = batch_size)

print(tr_dataset)
```

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

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

Example : word sentiment classification

Creating and training model

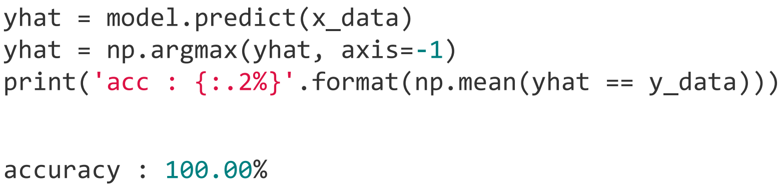
```
# training
tr_loss_hist = []

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

   for x_mb, y_mb in tr_dataset:
       with tf.GradientTape() as tape:
           tr_loss = loss_fn(model, x=x_mb, y=y_mb)
       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 : 0.156
epoch :  10, tr_loss : 0.031
epoch :  15, tr_loss : 0.009
epoch :  20, tr_loss : 0.005
epoch :  25, tr_loss : 0.003
epoch :  30, tr_loss : 0.003
```

https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

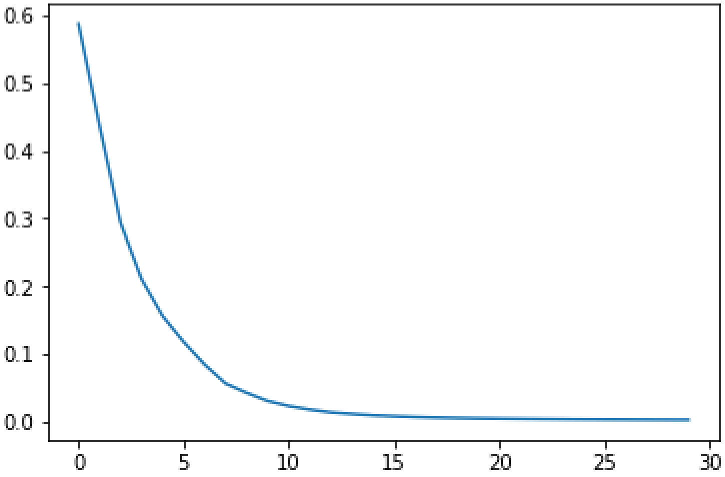
Example : word sentiment classification

Checking performance



```
yhat = model.predict(x_data)
yhat = np.argmax(yhat, axis=-1)
print('acc : {:.2%}'.format(np.mean(yhat == y_data)))
```

```
accuracy : 100.00%
```



https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-12-1-many-to-one-keras-eager.ipynb

What’s Next?

- many to one stacking