“Loading Data From File”

```
import tensorflow.compat.v1 as tf
import numpy as np
tf.disable_v2_behavior()

# for reproducibility
tf.set_random_seed(777)

xy = np.loadtxt('data-test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

```
One drawback of “dtype”
Every number should be the same type
```

```
# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)
```

Numpy indexing, slicing, iterating

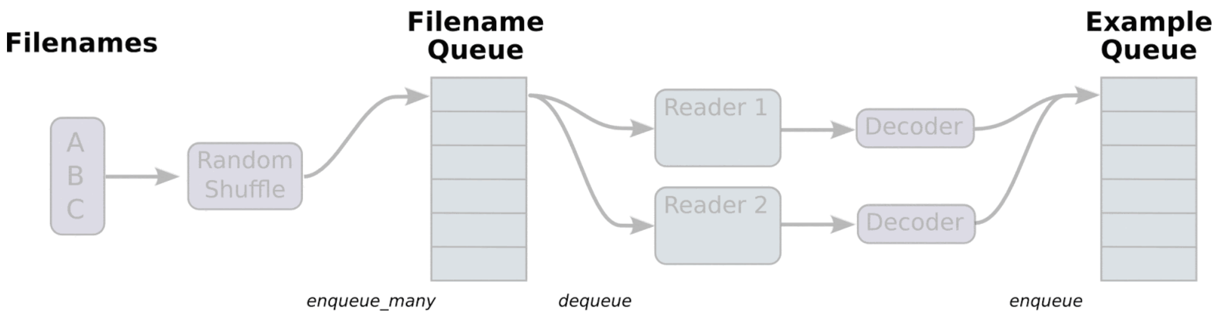
```
Ex) A = np.array([[1,2,3], [4,5,6], [7,8,9]])
A[ : 1]
→ array([ 2 , 5 , 8 ])
A[-1]
→ array([ 7 , 8 , 9 ])
A[-1, : ]
→ array([ 7 , 8 , 9 ])
A[-1, …]
→ array([ 7 , 8 , 9 ])
A[0:2, :]
→ array([ [ 1 , 2 , 3 ],
          [ 4 , 5 , 6 ] ])
```

1. A [X , Y]
→ X: row
→ Y: column
2. Colon = dot(“:” = “…”)

<What if there are lots of files?>

Queue Runners

1. Filenames   2. Defining Reader   3. Example Queue



```
1. filename_queue = tf.train.string_input_producer(
   ['data-01-test-score.csv', 'data-02-test-score.csv', ... ],
   shuffle=False, name='filename_queue')
```

```
2. reader = tf.TextLineReader()
key, value = reader.read(filename_queue)
```

```
3. record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)
```
The reason why we use [0.] → because we are using float

```
import tensorflow,compat.v1 as tf
import numpy as np
tf.disable_v2_behavior()

filename_queue = tf.train.string_input_producer(
   ['data-test-score.csv'], shuffle=False, name='filename_queue')

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)

# Default values, in case of empty columns. Also specifies the type of the
# decoded result.
record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)
```

```
If there are many rows…
→record_defaults = [ [0.] for row in range(n)
→Type the number you want in n
```

```
# collect batches of csv in
train_x_batch, train_y_batch = \
   tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
```

```
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)
```

If you want to use shuffle

<shuffle_batch>

```
# min_after_dequeue defines how big a buffer we will randomly sample
#   from -- bigger means better shuffling but slower start up and more
#   memory used.
# capacity must be larger than min_after_dequeue and the amount larger
#   determines the maximum we will prefetch.  Recommendation:
#   min_after_dequeue + (num_threads + a small safety margin) * batch_size

min_after_dequeue = 10000
capacity = min_after_dequeue + 3 * batch_size
example_batch, label_batch = tf.train.shuffle_batch(
   [example, label], batch_size=batch_size, capacity=capacity,
   min_after_dequeue=min_after_dequeue)
```

<Materials by>

-Sung Kim (Youtuber)

Code: https://github.com/hunkim/DeepLearningZeroToAll/