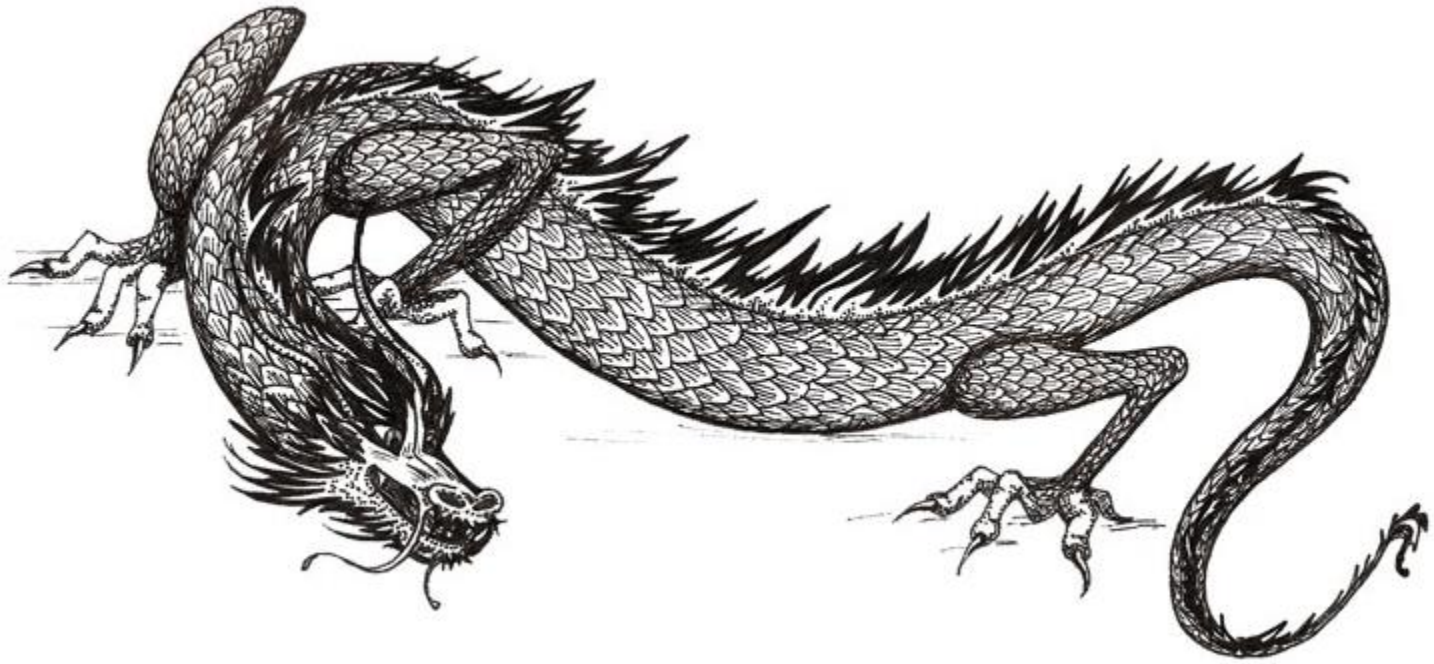

Compile&Fit

主讲：龙良曲

Outline

- Compile
- Fit
- Evaluate
- Predict

Individual loss and optimize

```
    with tf.GradientTape() as tape:
        # [b, 28, 28] => [b, 784]
        x = tf.reshape(x, (-1, 28*28))
        # [b, 784] => [b, 10]
        out = network(x)
        # [b] => [b, 10]
        y_onehot = tf.one_hot(y, depth=10)
        # [b]
        loss = tf.reduce_mean(tf.losses.categorical_crossentropy(y_onehot, out,
from_logits=True))

    grads = tape.gradient(loss, network.trainable_variables)
    optimizer.apply_gradients(zip(grads, network.trainable_variables))
```

Now

```
network.compile(optimizer=optimizers.Adam(lr=0.01),
        loss=tf.losses.CategoricalCrossentropy(from_logits=True),
        metrics=['accuracy']
    )
```

Individual epoch and step

```
for epoch in range(epochs):

    for step, (x, y) in enumerate(db):
        ....
```

Now

```
network.compile(optimizer=optimizers.Adam(lr=0.01),
        loss=tf.losses.CategoricalCrossentropy(from_logits=True),
        metrics=['accuracy']
    )

network.fit(db, epochs=10)
```

Standard Progressbar

```
ully opened dynamic library libcublas.so.10.0
4690/4690 [==============================] - 41s 9ms/step - loss: 0.1085
Epoch 2/10
4690/4690 [==============================] - 41s 9ms/step - loss: 0.0597
Epoch 3/10
4690/4690 [==============================] - 41s 9ms/step - loss: 0.0448
Epoch 4/10
4690/4690 [==============================] - 41s 9ms/step - loss: 0.0417
Epoch 5/10
4690/4690 [==============================] - 40s 8ms/step - loss: 0.0513
Epoch 6/10
4690/4690 [==============================] - 40s 9ms/step - loss: 0.1354
Epoch 7/10
4690/4690 [==============================] - 41s 9ms/step - loss: 0.1730
Epoch 8/10
4690/4690 [==============================] - 41s 9ms/step - loss: 0.2567
Epoch 9/10
4690/4690 [==============================] - 39s 8ms/step - loss: 0.3887
```

Individual evaluation

```
    # evaluate
    if step % 500 == 0:
        total, total_correct = 0., 0

        for step, (x, y) in enumerate(ds_val):
            # [b, 28, 28] => [b, 784]
            x = tf.reshape(x, (-1, 28*28))
            # [b, 784] => [b, 10]
            out = network(x)
            # [b, 10] => [b]
            pred = tf.argmax(out, axis=1)
            pred = tf.cast(pred, dtype=tf.int32)
            # bool type
            correct = tf.equal(pred, y)
            # bool tensor => int tensor => numpy
            total_correct += tf.reduce_sum(tf.cast(correct,
dtype=tf.int32)).numpy()
            total += x.shape[0]

        print(step, 'Evaluate Acc:', total_correct/total)
```

Now

```
network.compile(optimizer=optimizers.Adam(lr=0.01),
        loss=tf.losses.CategoricalCrossentropy(from_logits=True),
        metrics=['accuracy']
    )

network.fit(db, epochs=10, validation_data=ds_val,
            validation_steps=2)
```

freq

Evaluation

```
469/469 [==============================] - 5s 10ms/step - loss: 0.2794 - accuracy: 0.8408 - val_loss
: 0.0864 - val_accuracy: 0.9805
Epoch 2/10
469/469 [==============================] - 4s 8ms/step - loss: 0.1378 - accuracy: 0.9590 - val_loss:
 0.0637 - val_accuracy: 0.9805
Epoch 3/10
469/469 [==============================] - 4s 8ms/step - loss: 0.1077 - accuracy: 0.9693 - val_loss:
 0.0801 - val_accuracy: 0.9805
Epoch 4/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0941 - accuracy: 0.9729 - val_loss:
 0.0915 - val_accuracy: 0.9688
Epoch 5/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0833 - accuracy: 0.9775 - val_loss:
 0.1232 - val_accuracy: 0.9609
Epoch 6/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0793 - accuracy: 0.9789 - val_loss:
 0.0678 - val_accuracy: 0.9766
Epoch 7/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0741 - accuracy: 0.9810 - val_loss:
 0.0360 - val_accuracy: 0.9883
Epoch 8/10
```

Test

```
network.compile(optimizer=optimizers.Adam(lr=0.01),
        loss=tf.losses.CategoricalCrossentropy(from_logits=True),
        metrics=['accuracy']
    )

network.fit(db, epochs=10, validation_data=ds_val,
              validation_steps=2)

network.evaluate(ds_val)
```

Test

```
Epoch 8/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0710 - accuracy: 0.9807 - val_loss:
 0.1060 - val_accuracy: 0.9805
Epoch 9/10
469/469 [==============================] - 4s 8ms/step - loss: 0.0680 - accuracy: 0.9821 - val_loss:
 0.0927 - val_accuracy: 0.9766
Epoch 10/10
469/469 [==============================] - 4s 9ms/step - loss: 0.0594 - accuracy: 0.9845 - val_loss:
 0.0428 - val_accuracy: 0.9922
79/79 [==============================] - 0s 6ms/step - loss: 0.1205 - accuracy: 0.9757
```

Predict

```
sample = next(iter(ds_val))
x = sample[0]
y = sample[1] # one-hot
pred = network.predict(x) # [b, 10]
# convert back to number
y = tf.argmax(y, axis=1)
pred = tf.argmax(pred, axis=1)

print(pred)
print(y)
```

下一课时

自定义层

Thank You.