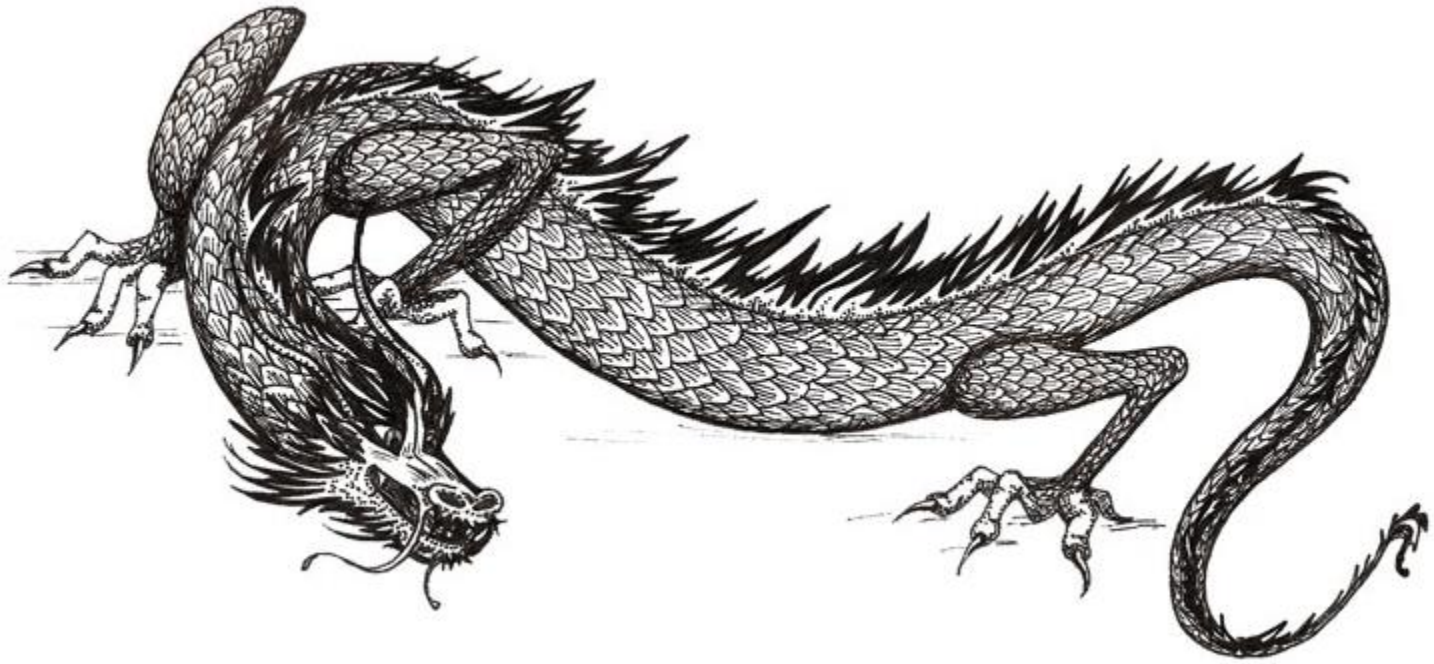
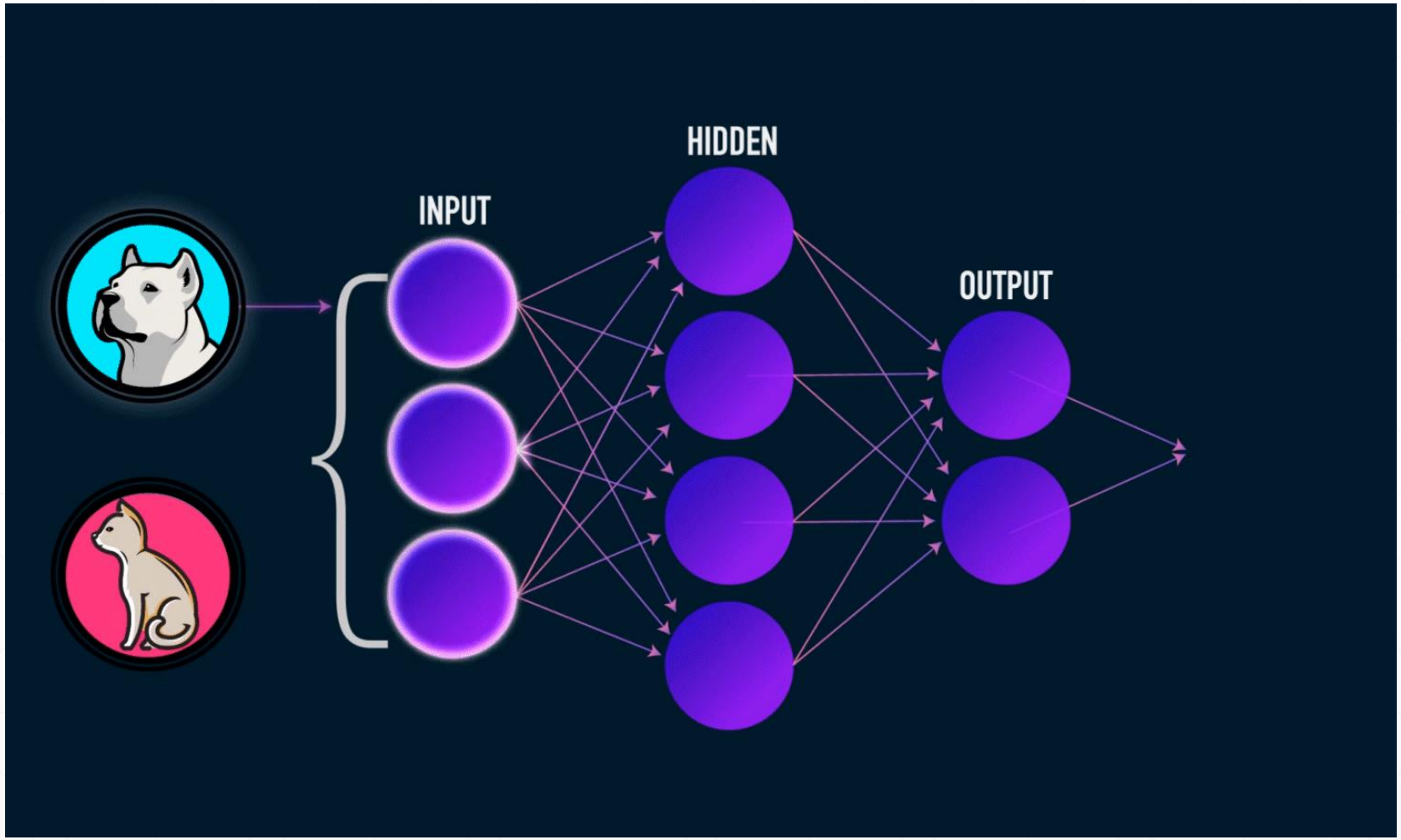


手写数字识别体验

主讲：龙良曲

Enjoy MNIST!



Step0. X and Y

```
import  tensorflow as tf
from    tensorflow.keras import datasets, layers, optimizers

(xs, ys),_ = datasets.mnist.load_data()
print('datasets:', xs.shape, ys.shape)

xs = tf.convert_to_tensor(xs, dtype=tf.float32) / 255.
db = tf.data.Dataset.from_tensor_slices((xs,ys))


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

Step0. X and Y

```
datasets: (60000, 28, 28) (60000,)
   ...
4971 (28, 28) tf.Tensor(1, shape=(), dtype=uint8) ()
4972 (28, 28) tf.Tensor(1, shape=(), dtype=uint8) ()
4973 (28, 28) tf.Tensor(9, shape=(), dtype=uint8) ()
4974 (28, 28) tf.Tensor(5, shape=(), dtype=uint8) ()
4975 (28, 28) tf.Tensor(9, shape=(), dtype=uint8) ()
4976 (28, 28) tf.Tensor(1, shape=(), dtype=uint8) ()
```

Step0. $out = relu\{relu\{relu[X@W_1 + b_1]@W_2 + b_2\}@W_3 + b_3\}$

```
model = keras.Sequential([
    layers.Dense(512, activation='relu'),
    layers.Dense(256, activation='relu'),
    layers.Dense(10)])

optimizer = optimizers.SGD(learning_rate=0.001)
```

Step1&2. Compute out&loss

```
        with tf.GradientTape() as tape:
            # [b, 28, 28] => [b, 784]
            x = tf.reshape(x, (-1, 28*28))
            # Step1. compute output
            # [b, 784] => [b, 10]
            out = model(x)
            # Step2. compute loss
            loss = tf.reduce_sum(tf.square(out - y)) / x.shape[0]
```

Step3.Compute gradient and optimize

```
        # Step3. optimize and update w1, w2, w3, b1, b2, b3
        grads = tape.gradient(loss, model.trainable_variables)
        # w' = w - lr * grad
        optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

Step4.Loop

```
def train_epoch(epoch):
    # Step4.loop
    for step, (x, y) in enumerate(train_dataset):
        with tf.GradientTape() as tape:
            # [b, 28, 28] => [b, 784]
            x = tf.reshape(x, (-1, 28*28))
            # Step1. compute output
            # [b, 784] => [b, 10]
            out = model(x)
            # Step2. compute loss
            loss = tf.reduce_sum(tf.square(out - y)) / x.shape[0]

        # Step3. optimize and update w1, w2, w3, b1, b2, b3
        grads = tape.gradient(loss, model.trainable_variables)
        # w' = w - lr * grad
        optimizer.apply_gradients(zip(grads, model.trainable_variables))

        if step  % 100 == 0:
            print(epoch, step, loss.numpy())
```

http://joshuaariza.com/jdi

下一课时

TensorFlow

Thank You.