

Ungraded Lab: Lambda Layer

This lab will show how you can define custom layers with the [Lambda](#) layer. You can either use [lambda functions](#) within the Lambda layer or define a custom function that the Lambda layer will call. Let's get started!

Imports

```
In [ ]: try:
        # %tensorflow_version only exists in Colab.
        %tensorflow_version 2.x
    except Exception:
        pass

    import tensorflow as tf
    from tensorflow.keras import backend as K
```

Prepare the Dataset

```
In [ ]: mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Build the Model

Here, we'll use a Lambda layer to define a custom layer in our network. We're using a lambda function to get the absolute value of the layer input.

```
In [ ]: model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128),
    tf.keras.layers.Lambda(Lambda x: tf.abs(x)),
    tf.keras.layers.Dense(10, activation='softmax')
])
```

```
In [ ]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Another way to use the Lambda layer is to pass in a function defined outside the model. The code below shows how a custom ReLU function is used as a custom layer in the model.

```
In [ ]: def my_relu(x):
        return K.maximum(-0.1, x)

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128),
    tf.keras.layers.Lambda(my_relu),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```