

# C1\_W3\_Lab\_1\_lambda-layer

December 15, 2022

## 0.1 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!

## 0.2 Imports

```
[1]: 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
```

## 0.3 Prepare the Dataset

```
[2]: 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
```

Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>  
11493376/11490434 [=====] - 0s 0us/step

## 0.4 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.

```
[3]: 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')
])
```

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

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

Train on 60000 samples

Epoch 1/5

60000/60000 [=====] - 5s 81us/sample - loss: 0.2223 - accuracy: 0.9368

Epoch 2/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0935 - accuracy: 0.9716

Epoch 3/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0640 - accuracy: 0.9804

Epoch 4/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0496 - accuracy: 0.9842

Epoch 5/5

60000/60000 [=====] - 5s 77us/sample - loss: 0.0349 - accuracy: 0.9887

10000/10000 [=====] - 0s 37us/sample - loss: 0.0763 - accuracy: 0.9771

```
[4]: [0.07627426998066948, 0.9771]
```

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.

```
[5]: 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)
```

Train on 60000 samples

Epoch 1/5

60000/60000 [=====] - 5s 80us/sample - loss: 0.2615 -  
accuracy: 0.9254

Epoch 2/5

60000/60000 [=====] - 5s 77us/sample - loss: 0.1108 -  
accuracy: 0.9671

Epoch 3/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0764 -  
accuracy: 0.9770

Epoch 4/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0566 -  
accuracy: 0.9823

Epoch 5/5

60000/60000 [=====] - 5s 78us/sample - loss: 0.0427 -  
accuracy: 0.9866

10000/10000 [=====] - 0s 31us/sample - loss: 0.0710 -  
accuracy: 0.9784

[5]: [0.0709538705676794, 0.9784]

[ ]:

[ ]:

[ ]: