```
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-1-84542532fca6> in <module>
----> 1 import tensorflow as tf
      2 import matplotlib.pyplot as plt

14 frames

/usr/local/lib/python3.7/dist-packages/jax/_src/lax/lax.py in naryop(result_dtype,
accepted_dtypes, name)
   1591 def naryop(result_dtype, accepted_dtypes, name):
   1592   dtype_rule = partial(naryop_dtype_rule, result_dtype, accepted_dtypes,
name)
-> 1593   shape_rule = partial(broadcasting_shape_rule, name)
   1594   weak_type_rule = partial(_naryop_weak_type_rule, name)
   1595   prim = standard_primitive(shape_rule, dtype_rule, name,

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

CIFAR-10

```
dataset = tf.keras.datasets.cifar10
```

```
(x_train, y_train), (x_test, y_test) = dataset.load_data()
```

```
x_train.shape
```

```
x_train, x_test = x_train/255, x_test/255
```

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), input_shape=(32,32,3), activation="relu"),
    tf.keras.layers.MaxPool2D((2,2)),
    tf.keras.layers.Conv2D(64, (2,2)),
    tf.keras.layers.MaxPool2D((2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation="relu"),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation="sigmoid")
    ])
```

```
sgd = tf.keras.optimizers.SGD(learning_rate=0.01,momentum=0.01)
```

```
model.compile(optimizer=sgd, loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logit
```

```
history = model.fit(x_train, y_train, validation_data=(x_test, y_test), batch_size=2000, e
```

Loss

```
    plt.figure()
    plt.plot(range(400), history.history["loss"], label="Loss")
    plt.plot(range(400), history.history["val_loss"], label="Validation loss")
    plt.xlabel("Epochs")
    plt.ylabel("Losses")
    plt.legend()
    plt.show()
```

Accuracy

```
plt.figure()
plt.plot(range(400), history.history["accuracy"], label="Accuracy")
plt.plot(range(400), history.history["val_accuracy"], label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()
plt.show()
```