In [ ]:
```
import tensorflow as tf
import numpy as np
import pandas as pd

from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout, \
  GlobalMaxPooling2D, MaxPooling2D, BatchNormalization
from tensorflow.keras.models import Model
```

In [ ]:
```
resolver = tf.distribute.cluster_resolver.TPUClusterResolver(tpu='')
tf.config.experimental_connect_to_cluster(resolver)
tf.tpu.experimental.initialize_tpu_system(resolver)
print("All devices: ", tf.config.list_logical_devices('TPU'))
```

```
All devices:  [LogicalDevice(name='/job:worker/replica:0/task:0/device:TPU:0', device_ty
pe='TPU'), LogicalDevice(name='/job:worker/replica:0/task:0/device:TPU:1', device_type
='TPU'), LogicalDevice(name='/job:worker/replica:0/task:0/device:TPU:2', device_type='TP
U'), LogicalDevice(name='/job:worker/replica:0/task:0/device:TPU:3', device_type='TPU'),
LogicalDevice(name='/job:worker/replica:0/task:0/device:TPU:4', device_type='TPU'), Logi
calDevice(name='/job:worker/replica:0/task:0/device:TPU:5', device_type='TPU'), LogicalD
evice(name='/job:worker/replica:0/task:0/device:TPU:6', device_type='TPU'), LogicalDevic
e(name='/job:worker/replica:0/task:0/device:TPU:7', device_type='TPU')]
```

In [ ]:
```
strategy = tf.distribute.TPUStrategy(resolver)
```

In [ ]:
```
# Note: model creation must be in strategy scope
# We will define the function now, but this code
# won't run outside the scope
```

In [ ]:
```
def create_model():
  i = Input(shape=(32, 32, 3))

  x = Conv2D(32, (3, 3), activation='relu', padding='same')(i)
  x = BatchNormalization()(x)
  x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = MaxPooling2D((2, 2))(x)
  x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = MaxPooling2D((2, 2))(x)
  x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
  x = BatchNormalization()(x)
  x = MaxPooling2D((2, 2))(x)

  x = Flatten()(x)
  x = Dropout(0.2)(x)
  x = Dense(1024, activation='relu')(x)
  x = Dropout(0.2)(x)
  x = Dense(10)(x)
```

```
  model = Model(i, x)
  return model
```

In [ ]:

```
# Load in the data
cifar10 = tf.keras.datasets.cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
y_train, y_test = y_train.flatten(), y_test.flatten()
print("x_train.shape:", x_train.shape)
print("y_train.shape", y_train.shape)
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
x_train.shape: (50000, 32, 32, 3)
y_train.shape (50000,)
```

In [ ]:

```
train_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train))
test_dataset = tf.data.Dataset.from_tensor_slices((x_test, y_test))
```

In [ ]:

```
with strategy.scope():
  model = create_model()
  model.compile(
      optimizer='adam',
      loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
      metrics=['sparse_categorical_accuracy'])

batch_size = 256

# reshuffle_each_iteration=None is default but is later set to True if None
# thus "True" is the actual default
train_dataset = train_dataset.shuffle(1000).batch(batch_size)
test_dataset = test_dataset.batch(batch_size)

model.fit(
    train_dataset,
    epochs=5,
    validation_data=test_dataset)
```

```
Epoch 1/5
196/196 [==============================] - 24s 77ms/step - loss: 1.6333 - sparse_categor
ical_accuracy: 0.4704 - val_loss: 3.3706 - val_sparse_categorical_accuracy: 0.1392
Epoch 2/5
196/196 [==============================] - 9s 44ms/step - loss: 0.9726 - sparse_categori
cal_accuracy: 0.6538 - val_loss: 3.2314 - val_sparse_categorical_accuracy: 0.2049
Epoch 3/5
196/196 [==============================] - 9s 44ms/step - loss: 0.7618 - sparse_categori
cal_accuracy: 0.7312 - val_loss: 0.9472 - val_sparse_categorical_accuracy: 0.6745
Epoch 4/5
196/196 [==============================] - 9s 44ms/step - loss: 0.6260 - sparse_categori
cal_accuracy: 0.7786 - val_loss: 0.7071 - val_sparse_categorical_accuracy: 0.7514
Epoch 5/5
196/196 [==============================] - 9s 44ms/step - loss: 0.5182 - sparse_categori
cal_accuracy: 0.8168 - val_loss: 0.6676 - val_sparse_categorical_accuracy: 0.7828
```

Out[ ]: <keras.callbacks.History at 0x7f256c228090>

In [ ]:
```
model.save('mymodel.h5')
```

In [ ]:
```
with strategy.scope():
  model = tf.keras.models.load_model('mymodel.h5')
  out = model.predict(x_test[:1])
  print(out)
```

In [ ]:
```
# Note: old bug
# https://www.kaggle.com/c/flower-classification-with-tpus/discussion/148615
```