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

import tensorflow as tf
print(tf.__version__)
```

```
`%tensorflow_version` only switches the major version: `1.x` or `2.x`.
You set: `2.x  # Colab only.`. This will be interpreted as: `2.x`.


TensorFlow 2.x selected.
2.0.0-beta1
```

In [ ]:
```
# additional imports

import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout, GlobalMaxPo
from tensorflow.keras.models import 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)
```

```
x_train.shape: (50000, 32, 32, 3)
y_train.shape (50000,)
```

In [ ]:
```
# number of classes
K = len(set(y_train))
print("number of classes:", K)
```

```
number of classes: 10
```

In [ ]:
```
# Build the model using the functional API
def create_model():
  i = Input(shape=x_train[0].shape)

  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(K, activation='softmax')(x)

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

In [ ]:

```
strategy = tf.distribute.MirroredStrategy()
# strategy = tf.distribute.experimental.CentralStorageStrategy()
```

In [ ]:

```
print(f'Number of devices: {strategy.num_replicas_in_sync}')
```

```
Number of devices: 1
```

In [ ]:

```
with strategy.scope():
  model = create_model()

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

In [ ]:

```
# Fit
r = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5)
```

```
WARNING: Logging before flag parsing goes to stderr.
W0811 18:03:54.175649 140429139875584 deprecation.py:323] From /tensorflow-2.0.0b1/pytho
n3.6/tensorflow/python/keras/layers/normalization.py:457: add_dispatch_support.<locals>.
wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 1563 steps, validate on 313 steps
Epoch 1/15
1563/1563 [==============================] - 23s 14ms/step - loss: 1.3045 - accuracy: 0.
5541 - val_loss: 0.9532 - val_accuracy: 0.6640
Epoch 2/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.8394 - accuracy: 0.
7093 - val_loss: 0.8020 - val_accuracy: 0.7339
Epoch 3/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.6854 - accuracy: 0.
7643 - val_loss: 0.8026 - val_accuracy: 0.7286
Epoch 4/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.5819 - accuracy: 0.
8012 - val_loss: 0.6905 - val_accuracy: 0.7778
Epoch 5/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.4859 - accuracy: 0.
8322 - val_loss: 0.6967 - val_accuracy: 0.7767
Epoch 6/15
1563/1563 [==============================] - 15s 10ms/step - loss: 0.4108 - accuracy: 0.
8567 - val_loss: 0.6145 - val_accuracy: 0.8024
Epoch 7/15
```

```
1563/1563 [==============================] - 15s 10ms/step - loss: 0.3517 - accuracy: 0.
8768 - val_loss: 0.6851 - val_accuracy: 0.7975
Epoch 8/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.2934 - accuracy: 0.
8977 - val_loss: 0.6551 - val_accuracy: 0.8104
Epoch 9/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.2536 - accuracy: 0.
9132 - val_loss: 0.6276 - val_accuracy: 0.8251
Epoch 10/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.2205 - accuracy: 0.
9224 - val_loss: 0.6870 - val_accuracy: 0.8156
Epoch 11/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.1921 - accuracy: 0.
9327 - val_loss: 0.7153 - val_accuracy: 0.8161
Epoch 12/15
1563/1563 [==============================] - 15s 10ms/step - loss: 0.1732 - accuracy: 0.
9413 - val_loss: 0.6962 - val_accuracy: 0.8218
Epoch 13/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.1587 - accuracy: 0.
9458 - val_loss: 0.7655 - val_accuracy: 0.8200
Epoch 14/15
1563/1563 [==============================] - 16s 10ms/step - loss: 0.1512 - accuracy: 0.
9505 - val_loss: 0.8894 - val_accuracy: 0.8010
Epoch 15/15
1556/1563 [============================>.] - ETA: 0s - loss: 0.1332 - accuracy: 0.9550
W0811 18:07:53.185556 140432385251200 training_arrays.py:309] Your dataset ran out of da
ta; interrupting training. Make sure that your dataset can generate at least `steps_per_
epoch * epochs` batches (in this case, 23445 batches). You may need to use the repeat()
function when building your dataset.
```

In [ ]:

```
50000/391
```

Out[ ]:

```
127.8772378516624
```

In [ ]:

```
10000/79
```

Out[ ]:

```
126.58227848101266
```

In [ ]:

```
# Compare this to non-distributed training
model2 = create_model()
model2.compile(loss='sparse_categorical_crossentropy',
               optimizer='adam',
               metrics=['accuracy'])
r = model2.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=5)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
50000/50000 [==============================] - 25s 502us/sample - loss: 1.3957 - accurac
y: 0.5369 - val_loss: 1.0386 - val_accuracy: 0.6328
Epoch 2/5
50000/50000 [==============================] - 23s 465us/sample - loss: 0.8563 - accurac
y: 0.7029 - val_loss: 0.8461 - val_accuracy: 0.7063
Epoch 3/5
50000/50000 [==============================] - 24s 471us/sample - loss: 0.7013 - accurac
y: 0.7592 - val_loss: 0.8004 - val_accuracy: 0.7334
Epoch 4/5
50000/50000 [==============================] - 24s 473us/sample - loss: 0.5897 - accurac
y: 0.7959 - val_loss: 0.6652 - val_accuracy: 0.7788
Epoch 5/5
```

```
50000/50000 [==============================] - 24s 471us/sample - loss: 0.5054 - accurac
y: 0.8267 - val_loss: 0.6899 - val_accuracy: 0.7758
```