```
import tensorflow as tf

#le code suivant divise le premier GPU (GPU 0) en 4 GPU -périphériques- virtuels. Chacun 2Gio de RAM.
#les instructions suivantes doivent être effectuées juste après l'importation du module tensorflow.

physical_gpus = tf.config.experimental.list_physical_devices("GPU")
tf.config.experimental.set_virtual_device_configuration(
    physical_gpus[0],
    [tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048),
     tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048),
     tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048),
     tf.config.experimental.VirtualDeviceConfiguration(memory_limit=2048)])
```

```
# imports commun
import numpy as np
import os

# pour rendre stable l'exécution relativement aux nombres aléatoire générés.
np.random.seed(42)

# pour une meilleure visibilité des figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)
import os

from tensorflow import keras
```

```
strategy = tf.distribute.MirroredStrategy()
print('Nombre de périphériques (GPU): {}'.format(strategy.num_replicas_in_sync))
```

```
Nombre de périphériques (GPU): 4
```

```
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 3s 0us/step
```

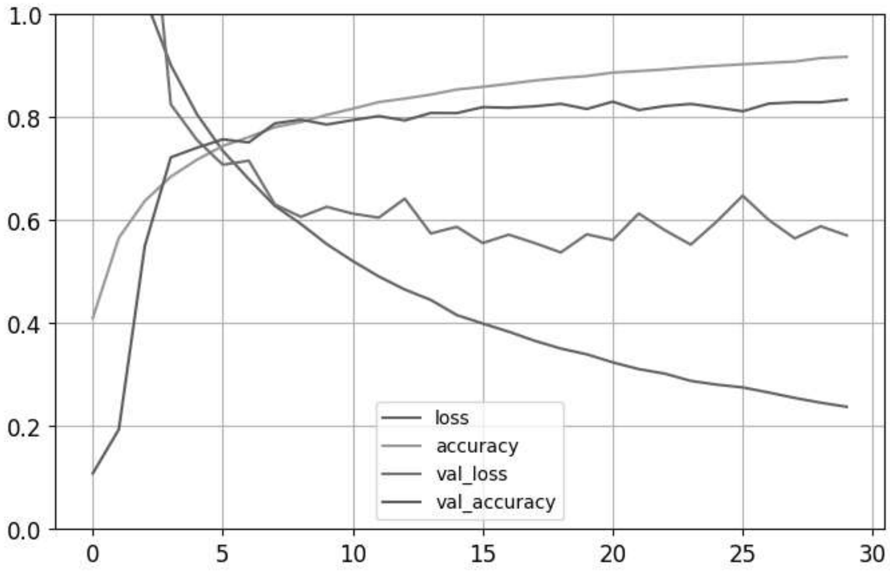
```
num_train_examples = x_train.shape[0]
num_test_examples = x_test[0]

BUFFER_SIZE = 1000

x_valid, x_train = x_train [:5000], x_train[5000:]
y_valid, y_train = y_train [:5000], y_train[5000:]

BATCH_SIZE_PER_REPLICA = 64
BATCH_SIZE = BATCH_SIZE_PER_REPLICA * strategy.num_replicas_in_sync

def scale(image, label):
  image = tf.cast(image, tf.float32)
  image /= 255

  return image, label

train_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train)).map(scale).cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
valid_dataset = tf.data.Dataset.from_tensor_slices((x_valid, y_valid)).map(scale).batch(BATCH_SIZE)
eval_dataset = tf.data.Dataset.from_tensor_slices((x_test, y_test)).map(scale).batch(BATCH_SIZE)
```

```
with strategy.scope():
  model = tf.keras.Sequential([
      tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(32, 32, 3),padding='same'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Conv2D(64, (3,3), activation='relu',padding='same'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Conv2D(128, (3,3), activation='relu',padding='same'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
      tf.keras.layers.BatchNormalization(),
      tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Flatten(),
      tf.keras.layers.Dense(128, activation='relu'),
      tf.keras.layers.Dropout(0.2),
      tf.keras.layers.Dense(10,activation='softmax'),
  ])

  model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                optimizer=tf.keras.optimizers.Adam(),
                metrics=['accuracy'])


EPOCHS = 30

from time import time
t0 = time()

history = model.fit(train_dataset, epochs=EPOCHS,validation_data=valid_dataset)
tt = time() - t0
print("classifier trained in {} seconds".format(round(tt,3)))
```

```
Epoch 3/30
176/176 [==============================] - 11s 60ms/step - loss: 1.0324 - accuracy: 0.6374 - val_loss: 1.3500 - val_accuracy: 0.5502
Epoch 4/30
176/176 [==============================] - 10s 58ms/step - loss: 0.9011 - accuracy: 0.6844 - val_loss: 0.8249 - val_accuracy: 0.7220
Epoch 5/30
176/176 [==============================] - 11s 61ms/step - loss: 0.8061 - accuracy: 0.7173 - val_loss: 0.7562 - val_accuracy: 0.7406
Epoch 6/30
176/176 [==============================] - 11s 61ms/step - loss: 0.7347 - accuracy: 0.7442 - val_loss: 0.7077 - val_accuracy: 0.7570
Epoch 7/30
176/176 [==============================] - 9s 53ms/step - loss: 0.6797 - accuracy: 0.7614 - val_loss: 0.7155 - val_accuracy: 0.7510
Epoch 8/30
176/176 [==============================] - 10s 58ms/step - loss: 0.6279 - accuracy: 0.7807 - val_loss: 0.6306 - val_accuracy: 0.7880
Epoch 9/30
176/176 [==============================] - 10s 58ms/step - loss: 0.5928 - accuracy: 0.7906 - val_loss: 0.6064 - val_accuracy: 0.7952
Epoch 10/30
176/176 [==============================] - 11s 60ms/step - loss: 0.5534 - accuracy: 0.8046 - val_loss: 0.6260 - val_accuracy: 0.7858
Epoch 11/30
176/176 [==============================] - 10s 56ms/step - loss: 0.5206 - accuracy: 0.8166 - val_loss: 0.6126 - val_accuracy: 0.7944
Epoch 12/30
176/176 [==============================] - 10s 54ms/step - loss: 0.4907 - accuracy: 0.8294 - val_loss: 0.6049 - val_accuracy: 0.8022
Epoch 13/30
176/176 [==============================] - 11s 60ms/step - loss: 0.4653 - accuracy: 0.8362 - val_loss: 0.6418 - val_accuracy: 0.7938
Epoch 14/30
176/176 [==============================] - 10s 58ms/step - loss: 0.4451 - accuracy: 0.8439 - val_loss: 0.5742 - val_accuracy: 0.8082
Epoch 15/30
176/176 [==============================] - 10s 56ms/step - loss: 0.4157 - accuracy: 0.8538 - val_loss: 0.5871 - val_accuracy: 0.8080
Epoch 16/30
176/176 [==============================] - 10s 56ms/step - loss: 0.3992 - accuracy: 0.8590 - val_loss: 0.5556 - val_accuracy: 0.8196
Epoch 17/30
176/176 [==============================] - 10s 58ms/step - loss: 0.3833 - accuracy: 0.8649 - val_loss: 0.5719 - val_accuracy: 0.8184
Epoch 18/30
176/176 [==============================] - 10s 58ms/step - loss: 0.3657 - accuracy: 0.8713 - val_loss: 0.5555 - val_accuracy: 0.8212
Epoch 19/30
176/176 [==============================] - 10s 56ms/step - loss: 0.3506 - accuracy: 0.8761 - val_loss: 0.5374 - val_accuracy: 0.8260
Epoch 20/30
```

```
176/176 [==============================] - 10s 57ms/step - loss: 0.3020 - accuracy: 0.8929 - val_loss: 0.5806 - val_accuracy: 0.8216
Epoch 24/30
176/176 [==============================] - 10s 58ms/step - loss: 0.2878 - accuracy: 0.8970 - val_loss: 0.5525 - val_accuracy: 0.8258
Epoch 25/30
176/176 [==============================] - 11s 60ms/step - loss: 0.2805 - accuracy: 0.8999 - val_loss: 0.5974 - val_accuracy: 0.8188
Epoch 26/30
176/176 [==============================] - 10s 59ms/step - loss: 0.2752 - accuracy: 0.9028 - val_loss: 0.6475 - val_accuracy: 0.8116
Epoch 27/30
176/176 [==============================] - 9s 52ms/step - loss: 0.2652 - accuracy: 0.9055 - val_loss: 0.6005 - val_accuracy: 0.8264
Epoch 28/30
176/176 [==============================] - 11s 60ms/step - loss: 0.2547 - accuracy: 0.9082 - val_loss: 0.5647 - val_accuracy: 0.8290
Epoch 29/30
176/176 [==============================] - 10s 58ms/step - loss: 0.2455 - accuracy: 0.9149 - val_loss: 0.5881 - val_accuracy: 0.8290
Epoch 30/30
176/176 [==============================] - 10s 58ms/step - loss: 0.2375 - accuracy: 0.9173 - val_loss: 0.5705 - val_accuracy: 0.8342
classifier trained in 378.386 seconds
```

```
import pandas as pd

pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```


loss
accuracy
val_loss
val_accuracy