```
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)])


# 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): 1
```

```
(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 [==============================] - 13s 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
704/704 [==============================] - 10s 15ms/step - loss: 0.9051 - accuracy: 0.6825 - val_loss: 0.8923 - val_accuracy: 0.6886
Epoch 4/30
704/704 [==============================] - 10s 14ms/step - loss: 0.7972 - accuracy: 0.7223 - val_loss: 0.8798 - val_accuracy: 0.7020
Epoch 5/30
704/704 [==============================] - 10s 15ms/step - loss: 0.7178 - accuracy: 0.7495 - val_loss: 0.7403 - val_accuracy: 0.7476
Epoch 6/30
704/704 [==============================] - 11s 16ms/step - loss: 0.6565 - accuracy: 0.7745 - val_loss: 0.7419 - val_accuracy: 0.7526
Epoch 7/30
704/704 [==============================] - 9s 13ms/step - loss: 0.6030 - accuracy: 0.7928 - val_loss: 0.6401 - val_accuracy: 0.7798
Epoch 8/30
704/704 [==============================] - 10s 15ms/step - loss: 0.5530 - accuracy: 0.8102 - val_loss: 0.5999 - val_accuracy: 0.8004
Epoch 9/30
704/704 [==============================] - 11s 15ms/step - loss: 0.5179 - accuracy: 0.8208 - val_loss: 0.5924 - val_accuracy: 0.7990
Epoch 10/30
704/704 [==============================] - 9s 13ms/step - loss: 0.4799 - accuracy: 0.8336 - val_loss: 0.6274 - val_accuracy: 0.7950
Epoch 11/30
704/704 [==============================] - 10s 14ms/step - loss: 0.4526 - accuracy: 0.8451 - val_loss: 0.5222 - val_accuracy: 0.8288
Epoch 12/30
704/704 [==============================] - 10s 14ms/step - loss: 0.4310 - accuracy: 0.8491 - val_loss: 0.5712 - val_accuracy: 0.8150
Epoch 13/30
704/704 [==============================] - 10s 14ms/step - loss: 0.4048 - accuracy: 0.8574 - val_loss: 0.5300 - val_accuracy: 0.8298
Epoch 14/30
704/704 [==============================] - 9s 13ms/step - loss: 0.3875 - accuracy: 0.8642 - val_loss: 0.5539 - val_accuracy: 0.8260
Epoch 15/30
704/704 [==============================] - 10s 14ms/step - loss: 0.3673 - accuracy: 0.8729 - val_loss: 0.5652 - val_accuracy: 0.8214
Epoch 16/30
704/704 [==============================] - 11s 15ms/step - loss: 0.3508 - accuracy: 0.8774 - val_loss: 0.5265 - val_accuracy: 0.8306
Epoch 17/30
704/704 [==============================] - 10s 15ms/step - loss: 0.3427 - accuracy: 0.8801 - val_loss: 0.5266 - val_accuracy: 0.8332
Epoch 18/30
704/704 [==============================] - 10s 14ms/step - loss: 0.3183 - accuracy: 0.8872 - val_loss: 0.5258 - val_accuracy: 0.8342
Epoch 19/30
704/704 [==============================] - 10s 14ms/step - loss: 0.3071 - accuracy: 0.8922 - val_loss: 0.5767 - val_accuracy: 0.8226
Epoch 20/30
```

```
704/704 [==============================] - 10s 15ms/step - loss: 0.2706 - accuracy: 0.9058 - val_loss: 0.5605 - val_accuracy: 0.8380
Epoch 24/30
704/704 [==============================] - 11s 15ms/step - loss: 0.2572 - accuracy: 0.9108 - val_loss: 0.5648 - val_accuracy: 0.8360
Epoch 25/30
704/704 [==============================] - 9s 13ms/step - loss: 0.2506 - accuracy: 0.9117 - val_loss: 0.5593 - val_accuracy: 0.8422
Epoch 26/30
704/704 [==============================] - 10s 15ms/step - loss: 0.2417 - accuracy: 0.9139 - val_loss: 0.5700 - val_accuracy: 0.8294
Epoch 27/30
704/704 [==============================] - 10s 14ms/step - loss: 0.2342 - accuracy: 0.9164 - val_loss: 0.5625 - val_accuracy: 0.8402
Epoch 28/30
704/704 [==============================] - 11s 15ms/step - loss: 0.2249 - accuracy: 0.9206 - val_loss: 0.5490 - val_accuracy: 0.8468
Epoch 29/30
704/704 [==============================] - 9s 13ms/step - loss: 0.2243 - accuracy: 0.9221 - val_loss: 0.5946 - val_accuracy: 0.8318
Epoch 30/30
704/704 [==============================] - 10s 15ms/step - loss: 0.2170 - accuracy: 0.9240 - val_loss: 0.5739 - val_accuracy: 0.8428
classifier trained in 366.14 seconds
```

```
import pandas as pd

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