```
'''
ECE471, Selected Topics in Machine Learning – Assignment 4
Submit by Oct. 4, 10pm
tldr: Classify cifar10. Acheive performance similar to the state of the art.
Classify cifar100. Achieve a top-5 accuracy of 70%.

The architecture is adapted from the AllConv net presented in this paper:
   https://arxiv.org/pdf/1412.6806.pdf
And some portions of code were modified from this GitHub repo:
   https://github.com/MateLabs/All-Conv-Keras
My contributions:
   – Experimented with placement and probabilities of dropout layers (removed
     the first dropout layer, contrary to paper).
   – Experimented with optimizers (decided that SGD was still the best, but
     used Nesterov momentum and a different learning rate, contrary to paper).
   – Trained it for longer, contrary to GitHub repo.
'''

import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.datasets.cifar10 import load_data
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Data parameters
NUM_CLASSES = 10
HEIGHT = 32
WIDTH = 32
NUM_CHANNELS = 3

# Load data and split into train, val, test sets
(x_train, y_train), (x_test, y_test) = load_data()
(x_val, y_val), (x_test, y_test) = \
    (x_test[:5000], y_test[:5000]), (x_test[5000:], y_test[5000:])

# Normalize and reshape data and labels
x_train, x_val, x_test = \
    map(lambda x: (x / 255.0).reshape([-1, HEIGHT, WIDTH, NUM_CHANNELS]),
        [x_train, x_val, x_test])
y_train, y_val, y_test = \
    map(lambda y: keras.utils.to_categorical(y, NUM_CLASSES),
        [y_train, y_val, y_test])

datagen = ImageDataGenerator(rotation_range=0.1,
                             width_shift_range=0.1,
                             height_shift_range=0.1,
                             horizontal_flip=True)
datagen.fit(x_train)


# Hyperparameters
BATCH_SIZE = 128
NUM_EPOCHS = 200  # I feel bad about this...

model = keras.Sequential()

# Idk if I can wrap this architecture with a function (DRY) without making
# it less readable... Notice the exceptions to the patterns.
model.add(keras.layers.Conv2D(96, 3, activation='relu', padding='same',
                              input_shape=[32, 32, 3]))
model.add(keras.layers.Conv2D(96, 3, activation='relu', padding='same'))
model.add(keras.layers.Conv2D(96, 3, activation='relu', padding='same',
                              strides=2))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Conv2D(192, 3, activation='relu', padding='same'))
model.add(keras.layers.Conv2D(192, 3, activation='relu', padding='same'))
model.add(keras.layers.Conv2D(192, 3, activation='relu', padding='same',
                              strides=2))
```



```
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Conv2D(192, 3, activation='relu', padding='same'))
model.add(keras.layers.Conv2D(192, 1, activation='relu', padding='valid'))
model.add(keras.layers.Conv2D(10, 1, activation='relu', padding='valid'))

model.add(keras.layers.GlobalAveragePooling2D())
model.add(keras.layers.Dense(NUM_CLASSES,
                             activation=keras.activations.softmax))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.SGD(lr=0.01,
                                             decay=1e-6,
                                             momentum=0.9,
                                             nesterov=True),
              metrics=['accuracy'])

model.fit_generator(datagen.flow(x_train, y_train, batch_size=BATCH_SIZE),
                    epochs=NUM_EPOCHS,
                    verbose=1,
                    validation_data=[x_val, y_val])

loss, acc = model.evaluate(x_test, y_test, verbose=1)

print('Test loss:', loss)
print('Test accuracy:', acc)

''' Output, omitting Keras logs.
Test loss: 0.3725
Test accuracy: 0.9054
'''
```