IRV2_on_GZ2_v4

April 30, 2022

[ ]:
```
import os
import pandas as pd
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import zipfile
import io
from PIL import Image
import matplotlib.pyplot as plt
from skimage.transform import resize
from tensorflow import keras
from tensorflow.keras.models import Model, load_model, Sequential
from tensorflow.keras.layers import Input, Dense, Conv2D, Flatten
from tensorflow.keras.optimizers import SGD, Adam
from keras.applications.inception_resnet_v2 import InceptionResNetV2 as ␣
 ↪PretrainedModel,preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,␣
 ↪array_to_img, img_to_array, load_img
from tensorflow.keras.callbacks import ModelCheckpoint, Callback, EarlyStopping
```

[ ]:
```
# zippath = '/content/drive/MyDrive/Major_Project/GZ-2/archive.zip'
# z = zipfile.ZipFile(zippath)
# imgname = 'images_gz2/images/233063.jpg'
# im = Image.open(io.BytesIO(z.read(imgname)))
# im_list = np.asarray(im)
# plt.imshow(im_list)
# plt.show()

# z.close()
```

[ ]:
```
# plt.figure(figsize=(16,4))
# for i in range(3):
#     plt.subplot(1,3,i+1)
#     plt.imshow(im_list[:,:,i])
#     plt.colorbar()
# plt.show()
```

[ ]:
```
# imgname = 'images_gz2/images/233063.jpg'
# img = load_img(imgname)
# data = img_to_array(img)
# samples = np.expand_dims(data, 0)
```

[ ]:
```
# def visualiseAugmentation(datagen):
#   it = datagen.flow(samples, batch_size=1)
#   plt.figure(figsize=(15,15))
#   for i in range(9):
#     plt.subplot(330 + 1 + i)
#     batch = it.next()
#     image = batch[0].astype('uint8')
#     plt.imshow(image)
#   plt.show()
```

[ ]:
```
# widthShift = ImageDataGenerator(width_shift_range=[-200,200])
# visualiseAugmentation(widthShift)
```

[ ]:
```
# zoomRange = ImageDataGenerator(zoom_range=[0.4, 0.7])
# visualiseAugmentation(zoomRange)
```

[ ]:
```
# rotation_range = ImageDataGenerator(rotation_range=90)
# visualiseAugmentation(rotation_range)
```

[ ]:
```
# shear_range = ImageDataGenerator(shear_range=0.7)
# visualiseAugmentation(shear_range)
```

[ ]:
```
def append_ext(fn):
    '''
    This function is used to take the GalaxyID from the CSV and append .jpg to␣
 ↪it in order to denote the image names.
    '''
    return fn + ".jpg"


traindf = pd.read_csv('D:/OneDrive/Major Project/HybridModel_37Classes/
 ↪GZ_2_Processed_classes.csv')

traindf["id"] = traindf['GalaxyID'].astype(str).apply(append_ext)
```

[ ]:
```
classes = [
    'Class1.1', 'Class1.2', 'Class1.3', 'Class2.1', 'Class2.2', 'Class3.1',
    'Class3.2', 'Class4.1', 'Class4.2', 'Class5.1', 'Class5.2', 'Class5.3',
    'Class5.4', 'Class6.1', 'Class6.2', 'Class7.1', 'Class7.2', 'Class7.3',
    'Class8.1', 'Class8.2', 'Class8.3', 'Class8.4', 'Class8.5', 'Class8.6',
    'Class8.7', 'Class9.1', 'Class9.2', 'Class9.3', 'Class10.1', 'Class10.2',
    'Class10.3', 'Class11.1', 'Class11.2', 'Class11.3', 'Class11.4',
```

```
    'Class11.5', 'Class11.6'
]
```

[ ]:
```
datagenerator = ImageDataGenerator(
    fill_mode='nearest',
    cval=0,
    rescale=1/255,
    rotation_range=25,
    shear_range=0.2,
    width_shift_range=[0.1, 0.15],
    height_shift_range=[0.1, 0.15],
    horizontal_flip=True,
    vertical_flip=True,
    zoom_range=[0.4, 0.7],
    validation_split=0.025)
```

[ ]:
```
train_generator = datagenerator.flow_from_dataframe(
    dataframe=traindf,
    directory="D:/Rahul Noronha/Shared Folder/Eighth Semester/Major Project/
 ↪Data/images",
    x_col="id",
    y_col=classes,
    subset="training",
    batch_size=64,
    seed=123,
    shuffle=True,
    class_mode="raw",
    target_size=(299, 299))

validation_generator = datagenerator.flow_from_dataframe(
    dataframe=traindf,
    directory="D:/Rahul Noronha/Shared Folder/Eighth Semester/Major Project/
 ↪Data/images",
    x_col="id",
    y_col=classes,
    subset="validation",
    batch_size=16,
    seed=123,
    shuffle=True,
    class_mode="raw",
    target_size=(299, 299))

STEP_SIZE_TRAIN = train_generator.n // train_generator.batch_size
STEP_SIZE_VALID = validation_generator.n // validation_generator.batch_size
```

```
D:\anaconda\envs\python37majorproject\lib\site-
packages\keras_preprocessing\image\dataframe_iterator.py:282: UserWarning: Found
```

```
108 invalid image filename(s) in x_col="id". These filename(s) will be ignored.
  .format(n_invalid, x_col)

Found 198632 validated image filenames.
Found 5093 validated image filenames.
```

[ ]:
```python
import os
import re
import sys
import time
import numpy as np
from typing import Any, List, Tuple, Union
from tensorflow.keras.datasets import mnist
from tensorflow.keras import backend as K
import tensorflow as tf
import tensorflow.keras
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, \
  LearningRateScheduler, ModelCheckpoint
from tensorflow.keras import regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.models import load_model
import pickle
```

[ ]:
```python
def generate_output_dir(outdir, run_desc):
    prev_run_dirs = []
    if os.path.isdir(outdir):
        prev_run_dirs = [x for x in os.listdir(outdir) if os.path.isdir(\
            os.path.join(outdir, x))]
    prev_run_ids = [re.match(r'^\d+', x) for x in prev_run_dirs]
    prev_run_ids = [int(x.group()) for x in prev_run_ids if x is not None]
    cur_run_id = max(prev_run_ids, default=-1) + 1
    run_dir = os.path.join(outdir, f'{cur_run_id:05d}-{run_desc}')
    assert not os.path.exists(run_dir)
    os.makedirs(run_dir)
    return run_dir

# From StyleGAN2
class Logger(object):
    """Redirect stderr to stdout, optionally print stdout to a file, and
    optionally force flushing on both stdout and the file."""

    def __init__(self, file_name: str = None, file_mode: str = "w", \
                 should_flush: bool = True):
        self.file = None
```

```
        if file_name is not None:
            self.file = open(file_name, file_mode)

        self.should_flush = should_flush
        self.stdout = sys.stdout
        self.stderr = sys.stderr

        sys.stdout = self
        sys.stderr = self

    def __enter__(self) -> "Logger":
        return self

    def __exit__(self, exc_type: Any, exc_value: Any, \
                 traceback: Any) -> None:
        self.close()

    def write(self, text: str) -> None:
        """Write text to stdout (and a file) and optionally flush."""
        if len(text) == 0:
            return

        if self.file is not None:
            self.file.write(text)

        self.stdout.write(text)

        if self.should_flush:
            self.flush()

    def flush(self) -> None:
        """Flush written text to both stdout and a file, if open."""
        if self.file is not None:
            self.file.flush()

        self.stdout.flush()

    def close(self) -> None:
        """Flush, close possible files, and remove
            stdout/stderr mirroring."""
        self.flush()

        # if using multiple loggers, prevent closing in wrong order
        if sys.stdout is self:
            sys.stdout = self.stdout
        if sys.stderr is self:
            sys.stderr = self.stderr
```

```
        if self.file is not None:
            self.file.close()
```

[ ]:
```
outdir = "D:/OneDrive/Major Project/HybridModel_37Classes/params/"
run_desc = "test-train"
batch_size = 128
num_classes = len(classes)

run_dir = generate_output_dir(outdir, run_desc)
print(f"Results saved to: {run_dir}")
```

```
Results saved to: D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train
```

[ ]:
```
class MyModelCheckpoint(ModelCheckpoint):
  def __init__(self, *args, **kwargs):
    super().__init__(*args, **kwargs)

  def on_epoch_end(self, epoch, logs):
    super().on_epoch_end(epoch,logs)\

    # Also save the optimizer state
    filepath = self._get_file_path(epoch, logs=logs, batch=2)
    filepath = filepath.rsplit( ".", 1 )[ 0 ]
    filepath += ".pkl"

    with open(filepath, 'wb') as fp:
      pickle.dump(
        {
          'opt': hybridModel.optimizer.get_config(),
          'epoch': epoch+1
         # Add additional keys if you need to store more values
        }, fp, protocol=pickle.HIGHEST_PROTOCOL)
    print('\nEpoch %05d: saving optimizer to %s' % (epoch + 1, filepath))
```

[ ]:
```
def step_decay_schedule(initial_lr=1e-3, decay_factor=0.75, step_size=10):
    def schedule(epoch):
        return initial_lr * (decay_factor ** np.floor(epoch/step_size))
    return LearningRateScheduler(schedule)
```

[ ]:
```
# from tensorflow.keras.applications import DenseNet121, VGG16, ResNet50V2,␣
 ↪MobileNetV2, EfficientNetB0, Xception

img_shape = (299, 299, 3)
num_classes = len(classes)
```

```
def build_model(img_shape, num_classes):
    hybridModel = Sequential()

    pretrained_model = PretrainedModel(
            input_shape = img_shape,
            weights = 'imagenet',
            include_top = False

    )
    for layer in pretrained_model.layers:
            layer.trainable=False

    hybridModel.add(pretrained_model)
    hybridModel.add(Flatten())
    hybridModel.add(Dense(len(classes), activation='softmax'))
    optimizer = keras.optimizers.Adam()
    hybridModel.compile(optimizer, loss='mse', metrics=["accuracy"])
    return hybridModel

def train_model(hybridModel, initial_epoch=0, max_epochs=10):
    start_time = time.time()

    checkpoint_cb = MyModelCheckpoint(
        os.path.join(run_dir, 'model-{epoch:02d}-{val_loss:.2f}.hdf5'),
        monitor='val_loss',verbose=1)

    lr_sched_cb = step_decay_schedule(initial_lr=2.9874e-19, decay_factor=0.75,␣
 ↪\
                                      step_size=9)
    cb = [checkpoint_cb, lr_sched_cb]

    hist = hybridModel.fit(
    train_generator,
    steps_per_epoch=STEP_SIZE_TRAIN,
    validation_data=validation_generator,
    validation_steps=STEP_SIZE_VALID,
    epochs=max_epochs,
    initial_epoch = initial_epoch,
    callbacks=cb)
```

[ ]:
```
# with Logger(os.path.join(run_dir, 'log.txt')):
#   hybridModel = build_model(img_shape, num_classes)
#   train_model(hybridModel)
```

[ ]:
```
# !ls '/content/drive/MyDrive/Major Project/Galaxy_Morphology/Data/GalaxyZoo2/
 ↪model/params'
```

[ ]:
```
MODEL_PATH = 'D:/OneDrive/Major Project/HybridModel_37Classes/params/
 ↪00006-test-train/model-31-0.15.hdf5'
OPT_PATH = 'D:/OneDrive/Major Project/HybridModel_37Classes/params/
 ↪00006-test-train/model-31-0.15.pkl'
```

[ ]:
```
def load_model_data(model_path, opt_path):
    model = load_model(model_path)
    with open(opt_path, 'rb') as fp:
      d = pickle.load(fp)
      epoch = d['epoch']
      opt = d['opt']
      return epoch, model, opt

epoch, hybridModel, opt = load_model_data(MODEL_PATH, OPT_PATH)
hybridModel.compile(optimizer=tf.keras.optimizers.Adam.from_config(opt),␣
 ↪loss='mse', metrics=["accuracy"])
with Logger(os.path.join(run_dir, 'log.txt')):
  train_model(hybridModel, initial_epoch=epoch, max_epochs=100)
```

```
Epoch 32/100
3103/3103 [==============================] - ETA: 0s - loss: 0.1562 - accuracy:
0.3715
Epoch 32: saving model to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-32-0.15.hdf5

Epoch 00032: saving optimizer to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-32-0.15.pkl
3103/3103 [==============================] - 7385s 2s/step - loss: 0.1562 -
accuracy: 0.3715 - val_loss: 0.1484 - val_accuracy: 0.2824 - lr: 1.2603e-19
Epoch 33/100
3103/3103 [==============================] - ETA: 0s - loss: 0.1562 - accuracy:
0.3695
Epoch 33: saving model to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-33-0.15.hdf5

Epoch 00033: saving optimizer to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-33-0.15.pkl
3103/3103 [==============================] - 4606s 1s/step - loss: 0.1562 -
accuracy: 0.3695 - val_loss: 0.1484 - val_accuracy: 0.2856 - lr: 1.2603e-19
Epoch 34/100
3103/3103 [==============================] - ETA: 0s - loss: 0.1562 - accuracy:
0.3702
Epoch 34: saving model to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-34-0.15.hdf5

Epoch 00034: saving optimizer to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-34-0.15.pkl
```

```
3103/3103 [==============================] - 4042s 1s/step - loss: 0.1562 -
accuracy: 0.3702 - val_loss: 0.1484 - val_accuracy: 0.2913 - lr: 1.2603e-19
Epoch 35/100
3103/3103 [==============================] - ETA: 0s - loss: 0.1562 - accuracy:
0.3721
Epoch 35: saving model to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-35-0.15.hdf5

Epoch 00035: saving optimizer to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-35-0.15.pkl
3103/3103 [==============================] - 7497s 2s/step - loss: 0.1562 -
accuracy: 0.3721 - val_loss: 0.1484 - val_accuracy: 0.2816 - lr: 1.2603e-19
Epoch 36/100
3103/3103 [==============================] - ETA: 0s - loss: 0.1562 - accuracy:
0.3721
Epoch 36: saving model to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-36-0.15.hdf5

Epoch 00036: saving optimizer to D:/OneDrive/Major
Project/HybridModel_37Classes/params/00007-test-train\model-36-0.15.pkl
3103/3103 [==============================] - 5476s 2s/step - loss: 0.1562 -
accuracy: 0.3721 - val_loss: 0.1484 - val_accuracy: 0.2836 - lr: 1.2603e-19
Epoch 37/100
  13/3103 […] - ETA: 2:36:39 - loss: 0.1572 -
accuracy: 0.3690
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_13912\3742729547.py in <module>
     10 hybridModel.compile(optimizer=tf.keras.optimizers.Adam.from_config(opt)
 ↪loss='mse', metrics=["accuracy"])
     11 with Logger(os.path.join(run_dir, 'log.txt')):
---> 12   train_model(hybridModel, initial_epoch=epoch, max_epochs=100)

~\AppData\Local\Temp\ipykernel_13912\1175396569.py in train_model(hybridModel,
 ↪initial_epoch, max_epochs)
     41     epochs=max_epochs,
     42     initial_epoch = initial_epoch,
---> 43     callbacks=cb)
     44

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\keras\utils\traceback_u
 ↪py in error_handler(*args, **kwargs)
     62     filtered_tb = None
     63     try:
---> 64       return fn(*args, **kwargs)
```

```
     65     except Exception as e:  # pylint: disable=broad-except
     66       filtered_tb = _process_traceback_frames(e.__traceback__)

D:\anaconda\envs\python37majorproject\lib\site-packages\keras\engine\training.py␣
 ↪in fit(self, x, y, batch_size, epochs, verbose, callbacks, validation_split,␣
 ↪validation_data, shuffle, class_weight, sample_weight, initial_epoch,␣
 ↪steps_per_epoch, validation_steps, validation_batch_size, validation_freq,␣
 ↪max_queue_size, workers, use_multiprocessing)
   1382                 _r=1):
   1383               callbacks.on_train_batch_begin(step)
-> 1384               tmp_logs = self.train_function(iterator)
   1385               if data_handler.should_sync:
   1386                 context.async_wait()

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\util\traceback_util
 ↪py in error_handler(*args, **kwargs)
    148     filtered_tb = None
    149     try:
--> 150       return fn(*args, **kwargs)
    151     except Exception as e:
    152       filtered_tb = _process_traceback_frames(e.__traceback__)

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in __call__(self, *args, **kwds)
    913
    914       with OptionalXlaContext(self._jit_compile):
--> 915         result = self._call(*args, **kwds)
    916
    917       new_tracing_count = self.experimental_get_tracing_count()

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\def_function
 ↪py in _call(self, *args, **kwds)
    945       # In this case we have created variables on the first call, so we
 ↪run the
    946       # defunned version which is guaranteed to never create variables.
--> 947       return self._stateless_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    948     elif self._stateful_fn is not None:
    949       # Release the lock early so that multiple threads can perform the
 ↪call

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\function.
 ↪py in __call__(self, *args, **kwargs)
   2955        filtered_flat_args) = self._maybe_define_function(args, kwargs)
   2956     return graph_function._call_flat(
```

```
-> 2957         filtered_flat_args, captured_inputs=graph_function.
 ↪captured_inputs)  # pylint: disable=protected-access
  2958
  2959   @property

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\function.
 ↪py in _call_flat(self, args, captured_inputs, cancellation_manager)
  1852       # No tape is watching; skip to running the function.
  1853       return self._build_call_outputs(self._inference_function.call(
-> 1854           ctx, args, cancellation_manager=cancellation_manager))
  1855     forward_backward = self._select_forward_and_backward_functions(
  1856         args,

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\function.
 ↪py in call(self, ctx, args, cancellation_manager)
   502               inputs=args,
   503               attrs=attrs,
--> 504               ctx=ctx)
   505         else:
   506           outputs = execute.execute_with_cancellation(

D:
 ↪\anaconda\envs\python37majorproject\lib\site-packages\tensorflow\python\eager\execute.
 ↪py in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
    53     ctx.ensure_initialized()
    54     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name,␣
 ↪op_name,
---> 55                                         inputs, attrs, num_outputs)
    56   except core._NotOkStatusException as e:
    57     if name is not None:

KeyboardInterrupt:
```