common-task-1-electron-photon-classification

March 26, 2024

[ ]:
```
pip install --upgrade --quiet tensorflow
```

```
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
import h5py
import requests
import numpy as np
import matplotlib.pyplot as plt
import os
import PIL
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.metrics import roc_auc_score
from tensorflow.keras.optimizers import Adam
```

[ ]:
```
photon_url = "https://cernbox.cern.ch/remote.php/dav/public-files/
 ↪AtBT8y4MiQYFcgc/SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5"
photon_filename = "SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5"

electron_url = "https://cernbox.cern.ch/remote.php/dav/public-files/
 ↪FbXw3V4XNyYB3oA/SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5"
electron_filename = "SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5"

urls, filenames = [photon_url, electron_url], [photon_filename,␣
 ↪electron_filename]

for i in range(2):
    response = requests.get(urls[i])
    with open(filenames[i], 'wb') as f:
        f.write(response.content)
```

[ ]:
```
with h5py.File(photon_filename, 'r') as f_photon:
    X_photon, y_photon = f_photon['X'][:], f_photon['y'][:]

with h5py.File(electron_filename, 'r') as f_electron:
    X_electron, y_electron = f_electron['X'][:], f_electron['y'][:]

X_combined = np.concatenate((X_photon, X_electron), axis=0)
y_combined = np.concatenate((y_photon, y_electron), axis=0)

print("Shape of combined data array:", X_combined.shape)
print("Shape of combined target array:", y_combined.shape)
```

```
Shape of combined data array: (498000, 32, 32, 2)
Shape of combined target array: (498000,)
```

[ ]:
```
energy_channel = X_combined[:, :, :, 0]
time_channel = X_combined[:, :, :, 1]

# Create a new array by stacking the two channels along with a zeros array for␣
 ↪the third channel
X_combined_3channel = np.stack([energy_channel, time_channel, np.
 ↪zeros_like(time_channel)], axis=-1)
```

[ ]:
```
x_train, x_test, y_train, y_test = train_test_split(X_combined, y_combined,␣
 ↪test_size=0.2, shuffle=True, random_state=42, stratify=y_combined)
```

[ ]:
```
y_train[0]
```

[ ]:
```
0.0
```

[ ]:
```
x_train.shape
```

[ ]:
```
(398400, 32, 32, 2)
```

[ ]:
```
del X_combined_3channel
del y_combined
del energy_channel
del time_channel
del X_photon
del y_photon
del X_electron
del y_electron
```

[ ]:
```
model = Sequential()

resnet50_model = keras.applications.ResNet50(
    include_top=True,
```

```
    weights=None,
    input_shape=(32, 32, 2),
    pooling='max',
    classes=1,
    classifier_activation="sigmoid",
)
# for layer in resnet50_model.layers:
#     layer.trainable = False

model.add(resnet50_model)
# model.add(Flatten())
# model.add(Dense(512, activation='relu'))
# model.add(Dropout(0.5))
# model.add(Dense(1, activation='sigmoid'))

model.summary()
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/applications/resnet.py:125:
UserWarning: This model usually expects 1 or 3 input channels. However, it was
passed an input_shape with 2 input channels.
  input_shape = imagenet_utils.obtain_input_shape(
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	?	23,586,625

Total params: 23,586,625 (89.98 MB)

Trainable params: 23,533,505 (89.77 MB)

Non-trainable params: 53,120 (207.50 KB)

[ ]:
```
model.compile(loss="binary_crossentropy", optimizer=Adam(learning_rate=0.001),␣
  ↪metrics=["accuracy"])
```

[ ]:
```
# rm -rf /kaggle/working/model_output
```

[ ]:
```
output_dir = 'model_output/resnet50_maxpool_adam001'

modelcheckpoint = ModelCheckpoint(filepath=output_dir+"/weights.{epoch:02d}.
  ↪keras")
```

```
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

callback = keras.callbacks.EarlyStopping(
    monitor="val_loss",
    patience=3,
    verbose=1,
    restore_best_weights=True,
)
```

[ ]:
```
epochs = 50
batch_size = 32

history = model.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(x_test, y_test),
    callbacks=[callback, modelcheckpoint]
)
```

```
Epoch 1/50

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1711103069.426778     881 service.cc:145] XLA service 0x7f8414002570
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1711103069.426822     881 service.cc:153]   StreamExecutor device
(0): Tesla P100-PCIE-16GB, Compute Capability 6.0

    3/12450            6:48 33ms/step -
accuracy: 0.4740 - loss: 1.8538

I0000 00:00:1711103089.079931     881 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

12450/12450            416s
29ms/step - accuracy: 0.5651 - loss: 0.7258 - val_accuracy: 0.6027 - val_loss:
0.6663
Epoch 2/50
12450/12450            343s
28ms/step - accuracy: 0.6050 - loss: 0.6702 - val_accuracy: 0.6134 - val_loss:
0.6592
Epoch 3/50
12450/12450            343s
```

```
28ms/step - accuracy: 0.6331 - loss: 0.6456 - val_accuracy: 0.6649 - val_loss:
0.6316
Epoch 4/50
12450/12450             345s
28ms/step - accuracy: 0.6610 - loss: 0.6226 - val_accuracy: 0.6728 - val_loss:
0.6117
Epoch 5/50
12450/12450             341s
27ms/step - accuracy: 0.6822 - loss: 0.6028 - val_accuracy: 0.6820 - val_loss:
0.6078
Epoch 6/50
12450/12450             336s
27ms/step - accuracy: 0.6966 - loss: 0.5877 - val_accuracy: 0.6975 - val_loss:
0.5835
Epoch 7/50
12450/12450             338s
27ms/step - accuracy: 0.7056 - loss: 0.5783 - val_accuracy: 0.7092 - val_loss:
0.5720
Epoch 8/50
12450/12450             340s
27ms/step - accuracy: 0.7087 - loss: 0.5733 - val_accuracy: 0.7072 - val_loss:
0.5751
Epoch 9/50
12450/12450             340s
27ms/step - accuracy: 0.7122 - loss: 0.5686 - val_accuracy: 0.6910 - val_loss:
0.5904
Epoch 10/50
12450/12450             341s
27ms/step - accuracy: 0.7142 - loss: 0.5668 - val_accuracy: 0.7141 - val_loss:
0.5670
Epoch 11/50
12450/12450             343s
28ms/step - accuracy: 0.7171 - loss: 0.5629 - val_accuracy: 0.7129 - val_loss:
0.5670
Epoch 12/50
12450/12450             339s
27ms/step - accuracy: 0.7204 - loss: 0.5592 - val_accuracy: 0.7163 - val_loss:
0.5651
Epoch 13/50
12450/12450             359s
29ms/step - accuracy: 0.7223 - loss: 0.5560 - val_accuracy: 0.7184 - val_loss:
0.5612
Epoch 14/50
12450/12450             362s
29ms/step - accuracy: 0.7212 - loss: 0.5564 - val_accuracy: 0.7160 - val_loss:
0.5608
Epoch 15/50
12450/12450             363s
```

```
29ms/step - accuracy: 0.7236 - loss: 0.5534 - val_accuracy: 0.6699 - val_loss:
0.6014
Epoch 16/50
12450/12450             368s
30ms/step - accuracy: 0.7269 - loss: 0.5506 - val_accuracy: 0.7175 - val_loss:
0.5643
Epoch 17/50
12450/12450             352s
28ms/step - accuracy: 0.7268 - loss: 0.5496 - val_accuracy: 0.7212 - val_loss:
0.5569
Epoch 18/50
12450/12450             342s
27ms/step - accuracy: 0.7277 - loss: 0.5488 - val_accuracy: 0.7227 - val_loss:
0.5552
Epoch 19/50
12450/12450             342s
27ms/step - accuracy: 0.7293 - loss: 0.5466 - val_accuracy: 0.5319 - val_loss:
0.8686
Epoch 20/50
12450/12450             341s
27ms/step - accuracy: 0.7310 - loss: 0.5447 - val_accuracy: 0.7200 - val_loss:
0.5588
Epoch 21/50
12450/12450             342s
27ms/step - accuracy: 0.7324 - loss: 0.5422 - val_accuracy: 0.6662 - val_loss:
0.6207
Epoch 21: early stopping
Restoring model weights from the end of the best epoch: 18.
```

[ ]:
```
model.save("trainedResNet50.keras")
```

[ ]:
```
x = keras.models.load_model('trainedResNet50.keras')
```

```
/opt/conda/lib/python3.10/site-packages/keras/src/saving/saving_lib.py:396:
UserWarning: Skipping variable loading for optimizer 'rmsprop', because it has
216 variables whereas the saved optimizer has 431 variables.
  trackable.load_own_variables(weights_store.get(inner_path))
```

[ ]:
```
x.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(32, 1)	23,586,625

Total params: 47,120,132 (179.75 MB)

Trainable params: 23,533,505 (89.77 MB)

Non-trainable params: 53,120 (207.50 KB)

Optimizer params: 23,533,507 (89.77 MB)

[ ]:
```
import os
import subprocess
from IPython.display import FileLink, display

def download_file(path, download_file_name):
    os.chdir('/kaggle/working/')
    zip_name = f"/kaggle/working/{download_file_name}.zip"
    command = f"zip {zip_name} {path} -r"
    result = subprocess.run(command, shell=True, capture_output=True, text=True)
    if result.returncode != 0:
        print("Unable to run zip command!")
        print(result.stderr)
        return
    display(FileLink(f'{download_file_name}.zip'))
```

[ ]:
```
download_file('/kaggle/working/trainedResNet50.keras', 'trainedResNet50.keras')
```

/kaggle/working/trainedResNet50.keras.zip

[ ]:
```
model = Sequential()

resnet50_model = keras.applications.ResNet50(
    include_top=True,
    weights=None,
    input_shape=(32, 32, 2),
    pooling='avg',
    classes=1,
    classifier_activation="sigmoid",
)
# for layer in resnet50_model.layers:
#     layer.trainable = False

model.add(resnet50_model)
# model.add(Flatten())
# model.add(Dense(512, activation='relu'))
```

```
# model.add(Dropout(0.5))
# model.add(Dense(1, activation='sigmoid'))

model.summary()
epochs = 50
batch_size = 32

history = model.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    verbose=1,
    validation_data=(x_test, y_test),
    callbacks=[callback, modelcheckpoint]
)
```

```
Epoch 1/50

WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
I0000 00:00:1711099400.521850     362 service.cc:145] XLA service 0x7e0c9c005fc0
initialized for platform CUDA (this does not guarantee that XLA will be used).
Devices:
I0000 00:00:1711099400.521893     362 service.cc:153]   StreamExecutor device
(0): Tesla P100-PCIE-16GB, Compute Capability 6.0

    3/12450            6:37 32ms/step -
accuracy: 0.4549 - loss: 1.7112

I0000 00:00:1711099419.214896     362 device_compiler.h:188] Compiled cluster
using XLA!  This line is logged at most once for the lifetime of the process.

12450/12450            400s
28ms/step - accuracy: 0.5691 - loss: 0.7245 - val_accuracy: 0.5965 - val_loss:
0.6680
Epoch 2/50
12450/12450            335s
27ms/step - accuracy: 0.6134 - loss: 0.6711 - val_accuracy: 0.6056 - val_loss:
0.6691
Epoch 3/50
12450/12450            333s
27ms/step - accuracy: 0.6672 - loss: 0.6235 - val_accuracy: 0.6711 - val_loss:
0.6168
Epoch 4/50
12450/12450            333s
27ms/step - accuracy: 0.6861 - loss: 0.6033 - val_accuracy: 0.6932 - val_loss:
0.5926
Epoch 5/50
```

```
12450/12450              336s
27ms/step - accuracy: 0.6966 - loss: 0.5894 - val_accuracy: 0.6884 - val_loss:
0.6562
Epoch 6/50
12450/12450              337s
27ms/step - accuracy: 0.7010 - loss: 0.5837 - val_accuracy: 0.7055 - val_loss:
0.5942
Epoch 7/50
12450/12450              337s
27ms/step - accuracy: 0.7049 - loss: 0.5800 - val_accuracy: 0.6794 - val_loss:
1.0661
Epoch 7: early stopping
Restoring model weights from the end of the best epoch: 4.
```

[ ]:
```
y_train.shape
```

[ ]:
```
(398400,)
```

[ ]:
```
x_train.shape
```

[ ]:
```
(398400, 32, 32, 2)
```

[ ]: