Multi_2_Conv

January 15, 2024

[1]:
```
#!pip install tensorflow
import pickle

import numpy as np
from matplotlib import pyplot as plt
from pandas import read_csv
```

[2]:
```
print("Loading data...")
training_file = './Data/train.p'

sign_names = read_csv("./Data/signname.csv").values[:, 1]

with open(training_file, mode='rb') as f:
    train = pickle.load(f)
images_train, labels_train = train['features'], train['labels']

for i in range(len(labels_train)):

    # replace hardik with shardul
    if labels_train[i] < 9:
        labels_train[i] = 0
    elif labels_train[i] >= 9:
        labels_train[i] = 1
```

```
Loading data...
```

[3]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D,␣
 ↪Dropout

# Assuming your image dimensions and channels
height = 32  # example height
width = 32   # example width
channels = 3  # RGB channels
```

```
# Define the new model
model = Sequential([
    # First Convolutional Layer with 32 filters, a 3x3 kernel size, 'same'␣
 ↪padding, and ReLU activation
    Conv2D(32, (3, 3), padding='same', activation='relu', input_shape=(height,␣
 ↪width, channels)),

    # MaxPooling to downsample the output of the first Convolutional Layer
    MaxPooling2D((2, 2)),

    # Second Convolutional Layer with 64 filters, a 3x3 kernel size, 'same'␣
 ↪padding, and ReLU activation
    Conv2D(64, (3, 3), padding='same', activation='relu'),

    # MaxPooling to downsample the output of the second Convolutional Layer
    MaxPooling2D((2, 2)),

    # Third Convolutional Layer with 128 filters, a 3x3 kernel size, 'same'␣
 ↪padding, and ReLU activation
    Conv2D(128, (3, 3), padding='same', activation='relu'),

    # MaxPooling to downsample the output of the third Convolutional Layer
    MaxPooling2D((2, 2)),

    # Additional Dropout layer after the third Convolutional Layer
    Dropout(0.3),

    # Flatten layer to convert the 2D output of the convolutional layers into a␣
 ↪1D array
    Flatten(),

    # First Dense (fully connected) layer with 128 units and ReLU activation
    Dense(128, activation='relu'),

    # Dropout layer with 50% dropout rate for regularization
    Dropout(0.5),

    # Second Dense layer with 64 units and ReLU activation
    Dense(64, activation='relu'),

    # Output layer for binary classification using sigmoid activation
    Dense(1, activation='sigmoid')
])

# Compile the model
```

```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])

print("Finished building model...")
```

```
2023-12-12 11:42:16.340677: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

```
Finished building model…
```

```
2023-12-12 11:42:27.062985: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 AVX512F FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-12-12 11:42:27.066378: I
tensorflow/core/common_runtime/process_util.cc:146] Creating new thread pool
with default inter op setting: 2. Tune using inter_op_parallelism_threads for
best performance.
```

[4]:
```
validation_file = './Data/valid.p'

with open(validation_file, mode='rb') as f:
    valid = pickle.load(f)
images_valid, labels_valid = valid['features'], valid['labels']

for i in range(len(labels_valid)):

    # replace hardik with shardul
    if labels_valid[i] < 9:
        labels_valid[i] = 0
    elif labels_valid[i] >= 9:
        labels_valid[i] = 1
```

[ ]:
```
# model.fit(images_train, labels_train, epochs=10,␣
 ↪validation_data=(images_valid, labels_valid))
model.fit(images_train, labels_train, epochs=1, validation_data=(images_valid,␣
 ↪labels_valid))
```

```
Epoch 1/2
  79/1088 [=>…] - ETA: 4:32 - loss: 0.0142 -
accuracy: 0.9953
```

[ ]:
```
# Define the file name for saving the model
model_filename = 'Convolution_Model_Saved'

# Save the model to a file
model.save(model_filename)
```

[ ]:
```
test_file = './Data/test.p'

with open(test_file, mode='rb') as f:
    test = pickle.load(f)
images_test, labels_test = test['features'], test['labels']

for i in range(len(labels_test)):

    # replace hardik with shardul
    if labels_test[i] < 9:
        labels_test[i] = 0
    elif labels_test[i] >= 9:
        labels_test[i] = 1
```

[ ]:
```
test_loss, test_accuracy = model.evaluate(images_test, labels_test)
```