Multi_2_Conv_F1

January 15, 2024

[128]:

```python
import pickle

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

[129]:

```python
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…
```

[131]:

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Conv2D, MaxPooling2D,␣
  ↪Dropout
from sklearn.metrics import f1_score

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

# Define a function to calculate F1 score
```

```
def f1_metric(y_true, y_pred):
    return tf.py_function(f1_score, (y_true, y_pred > 0.5), tf.float64)
# 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', f1_metric])
```

[132]:

```
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
```

[133]:

```
history = model.fit(images_train, labels_train, epochs=10,␣
 ↪validation_data=(images_valid, labels_valid))
```

```
Epoch 1/10
1088/1088 [==============================] - 2s 2ms/step - loss: 2.1003 -
accuracy: 0.8670 - val_loss: 0.1769 - val_accuracy: 0.9447
Epoch 2/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2695 -
accuracy: 0.9274 - val_loss: 0.5872 - val_accuracy: 0.8342
Epoch 3/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2211 -
accuracy: 0.9365 - val_loss: 0.2082 - val_accuracy: 0.9138
Epoch 4/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.1824 -
accuracy: 0.9372 - val_loss: 0.1667 - val_accuracy: 0.9465
Epoch 5/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2222 -
accuracy: 0.9290 - val_loss: 0.1960 - val_accuracy: 0.9200
Epoch 6/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2282 -
accuracy: 0.9040 - val_loss: 0.2889 - val_accuracy: 0.9136
Epoch 7/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2142 -
accuracy: 0.9366 - val_loss: 0.1943 - val_accuracy: 0.9401
Epoch 8/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2313 -
accuracy: 0.9254 - val_loss: 0.1696 - val_accuracy: 0.9336
Epoch 9/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2362 -
accuracy: 0.9243 - val_loss: 0.2462 - val_accuracy: 0.9211
```

```
Epoch 10/10
1088/1088 [==============================] - 2s 2ms/step - loss: 0.2248 -
accuracy: 0.9297 - val_loss: 0.1538 - val_accuracy: 0.9458
```

```
[133]: <keras.src.callbacks.History at 0x286629040>
```

```
[ ]: # Define the file name for saving the model
     model_filename = 'Convolution_Model_F1_Ex_1'

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

```
[134]: 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
```

```
[ ]: import matplotlib.pyplot as plt

     # Train the model and store the training history in a variable named "history"

     # Extract accuracy values
     train_accuracy = history.history['accuracy']
     val_accuracy = history.history['val_accuracy']

     # Plot accuracy
     plt.figure(figsize=(8, 6))
     epochs = range(1, len(train_accuracy) + 1)
     plt.plot(epochs, train_accuracy, 'b', label='Training Accuracy')
     plt.plot(epochs, val_accuracy, 'r', label='Validation Accuracy')
     plt.title('Training and Validation Accuracy')
     plt.xlabel('Epochs')
     plt.ylabel('Accuracy')
     plt.legend()
     plt.show()
```

```
[135]: test_loss, test_accuracy, f1_score  = model.evaluate(images_test, labels_test)
```

```
395/395 [==============================] - 0s 753us/step - loss: 0.1589 -
```

```
accuracy: 0.9452
```