Multi_1

January 15, 2024

[18]:

```
#!pip install tensorflow
```

[12]:

```
tf.get_logger().setLevel('ERROR')
```

[14]:

```
import pickle
import numpy as np
from matplotlib import pyplot as plt
from pandas import read_csv
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense

# Load data
print("Loading data...")
#training_file = './Data/train.p'
training_file = 'train.p'

sign_names = read_csv("signname.csv").values[:, 1]
# 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']

# Filter only labels 0-8
mask_0_to_8 = labels_train <= 8
images_train_filtered = images_train[mask_0_to_8]
labels_train_filtered = labels_train[mask_0_to_8]

# Create a model for multi-class classification
model_multi_class = Sequential([
    Flatten(input_shape=(32, 32, 3)),  # Flatten the input

    Dense(128, activation='relu'),  # First fully connected layer
    Dense(64, activation='relu'),   # Second fully connected layer
    # The final Dense layer with 9 neurons for 9 classes (0 to 8) and softmax␣
 ↪activation for multi-class classification
    Dense(9, activation='softmax')  # Output layer for multi-class␣
 ↪classification (9 classes)
])

# Compile the model for multi-class classification
model_multi_class.compile(optimizer='adam',␣
 ↪loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Training the model
model_multi_class.fit(images_train_filtered, labels_train_filtered, epochs=10,␣
 ↪validation_split=0.2)

# Now you can proceed with evaluation or prediction on test/validation data
```

```
Loading data…
Epoch 1/10
294/294 [==============================] - 2s 4ms/step - loss: 13.6204 -
accuracy: 0.3643 - val_loss: 59.6374 - val_accuracy: 0.0774
Epoch 2/10
294/294 [==============================] - 1s 3ms/step - loss: 2.3437 -
accuracy: 0.5729 - val_loss: 75.2995 - val_accuracy: 0.0957
Epoch 3/10
294/294 [==============================] - 1s 3ms/step - loss: 1.9573 -
accuracy: 0.6327 - val_loss: 70.0488 - val_accuracy: 0.0876
Epoch 4/10
294/294 [==============================] - 1s 3ms/step - loss: 1.8119 -
accuracy: 0.6710 - val_loss: 60.4847 - val_accuracy: 0.0616
Epoch 5/10
294/294 [==============================] - 1s 3ms/step - loss: 0.9175 -
accuracy: 0.7701 - val_loss: 58.2926 - val_accuracy: 0.0948
Epoch 6/10
294/294 [==============================] - 1s 3ms/step - loss: 1.0400 -
accuracy: 0.7510 - val_loss: 56.9321 - val_accuracy: 0.0846
Epoch 7/10
294/294 [==============================] - 1s 3ms/step - loss: 0.7837 -
accuracy: 0.7944 - val_loss: 59.5798 - val_accuracy: 0.1008
Epoch 8/10
294/294 [==============================] - 1s 3ms/step - loss: 0.9120 -
accuracy: 0.7809 - val_loss: 67.3559 - val_accuracy: 0.0710
Epoch 9/10
294/294 [==============================] - 1s 4ms/step - loss: 0.6722 -
accuracy: 0.8200 - val_loss: 59.8849 - val_accuracy: 0.0578
Epoch 10/10
294/294 [==============================] - 1s 4ms/step - loss: 0.6515 -
accuracy: 0.8179 - val_loss: 53.0430 - val_accuracy: 0.1267
```

[14]:

```
<keras.src.callbacks.History at 0x7f358b0c9e10>
```

[16]:

```
# Load test data
#test_file = './Data/test.p'
test_file = 'test.p'

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

# Filter test data for labels 0-8
mask_test_0_to_8 = labels_test <= 8
images_test_filtered = images_test[mask_test_0_to_8]
labels_test_filtered = labels_test[mask_test_0_to_8]

# Evaluate the model on test data
test_loss, test_accuracy = model_multi_class.evaluate(images_test_filtered,␣
  ↪labels_test_filtered)
print(f"Test Accuracy: {test_accuracy}")
```

```
135/135 [==============================] - 0s 2ms/step - loss: 10.9780 -
accuracy: 0.6329
Test Accuracy: 0.6328703761100769
```

[21]:

```
model_multi_class.save('Experiment_2_Saved_Model')
```

[ ]:

```

```