ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - TIN HỌC

PHÂN TÍCH XỬ LÝ ẢNH

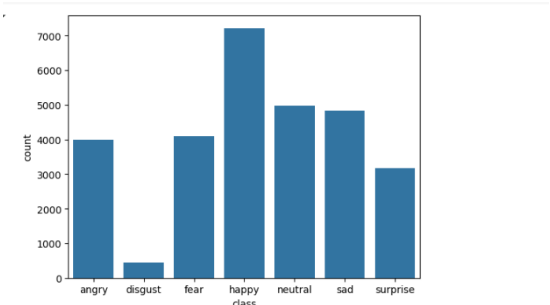
NHẬN DIỆN BIỂU CẢM KHUÔN MẶT BẰNG CÁCH SỬ DỤNG CNN

1 Sử dụng mô hình CNN để nhận diện cảm xúc khuôn mặt

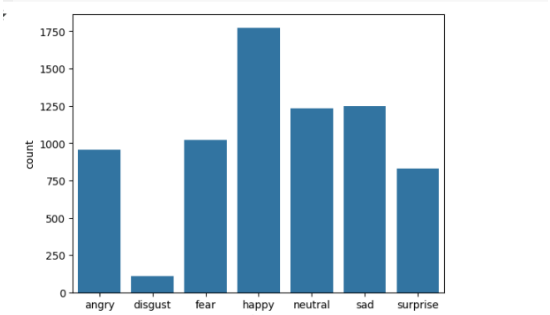
1.1 Tập dữ liệu:

Ở phần này, nhóm đã chia lại dữ liệu vào các mục có chứa sẵn lable và chia ra thành 2 tập train và test với tỉ lệ là 80-20.

Dữ liệu được mô tả nhưu hình dưới:



Hình 1: *Train data*

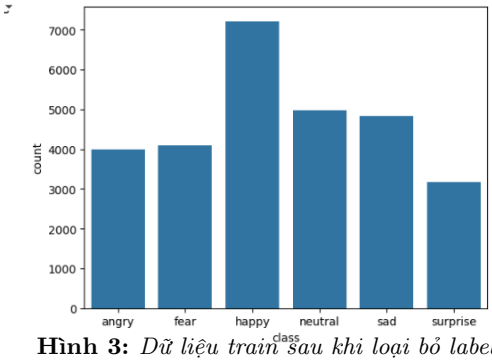


Hình 2: *Test data*

Ta tiến hành chia lại tập train thành tập train và tạp validation như sau:

```
] # splitting the training dataframe into train and val
  df_train, df_val = train_test_split(df, test_size=0.30, random_state=0)
```

Nhận xét: Tập dữ liệu có chứa dữ liệu label Disgust rất nhỏ, việc huấn luyện hay dự đoán với tập đó là rất khó, vì vậy để tránh mất cân bằng dữ liệu, nhóm đã quyết định bỏ label đó và chỉ huấn luyện trên 6 label còn lại.



Hình 3: *Dữ liệu train sau khi loại bỏ label disgust*

1.2 Xây dựng mô hình CNN:

Listing 1: Grid Search with Decision Tree

```
        # Initialize the CNN model
model = Sequential()

# Add Convolutional layers
model.add(Conv2D(32, (3, 3),padding='same', input_shape=(IMG_DIM, IMG_DIM, 1),
    activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3),padding='same', activation='relu',name='last_conv')
    )
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())

# Flatten the output before feeding into the fully connected layers
model.add(Flatten())

# Add Dense layers for classification
model.add(Dense(128, activation='relu'))
model.add(Dense(6, activation='softmax'))  # Output layer with 6 units for 6
    classes and softmax activation for multi-class classification

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
    accuracy'])

# Display the model summary
model.summary()
```

Ta có số Parameter sau khi xây dựng mô hình như sau:

```
Total params: 279,110 (1.06 MB)
Trainable params: 278,534 (1.06 MB)
Non-trainable params: 576 (2.25 KB)
```

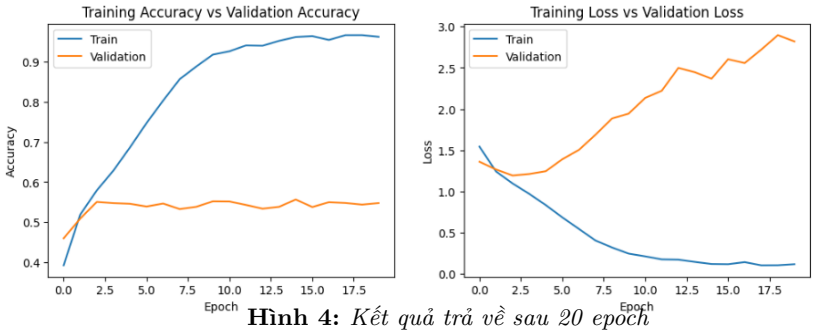
1.3 Tiến hành train dữ liệu:

```
from tensorflow.keras.callbacks import ModelCheckpoint

# make a check point that saves the best epoch
checkpoint_path = '/content/drive/My Drive/PTXLA/XLA/my_cnn_model.weights.h5'
checkpoint = ModelCheckpoint(checkpoint_path,
                             save_weights_only=True,
                             save_best_only=True,
                             verbose=1,
                             mode='min')

```

```python
# train the model in 20 epochs
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=20,
    callbacks=[checkpoint],
    shuffle=True
)
```



Hình 4: *Kết quả trả về sau 20 epoch*

2 kết quả và đánh giá mô hình

Kết quả mô hình rả về cho thấy độ chính xác của tập train cao, còn tập Validation nằm trong khoảng 60%. Vậy độ chính xác tập thực nghiệm là tầm 60%

```python
# evaluate the model on the testing images
print(checkpoint_path)
model.load_weights(checkpoint_path)
model.evaluate(train_gen)
```

```
/content/drive/My Drive/PTXLA/XLA/my_cnn_model.weights.h5
619/619 ━━━━━━━━━━━━━━━━━━━━ 66s 107ms/step - accuracy: 0.6585 - loss: 0.9133
[0.9149859547615051, 0.6552647352218628]
```

Hình 5: *Kết quả hàm mất mát và độ chính xác của tập train*

```python
# evaluate the model on the testing images
print(checkpoint_path)
model.load_weights(checkpoint_path)
model.evaluate(test_gen)
```

```
/content/drive/My Drive/PTXLA/XLA/my_cnn_model.weights.h5
221/221 ━━━━━━━━━━━━━━━━━━━━ 21s 95ms/step - accuracy: 0.4962 - loss: 1.2642
[1.188363790512085, 0.5469081401824951]
```

Hình 6: *Kết quả hàm mất mát và độ chính xác của tập test*

Kết quả cho thấy độ chính xác trên tập test vào khoảng 55%

2.1 Tiến hành kiểm tra mô hình:

Lấy 10 mẫu ngẫu nhiên trong tập test và tiên shanhf kiểm tra, ta thu được kết quả sau:

Nhận thấy kết quả đúng khá nhiều (7/10), các dự đoán sai có thể thấy được là do khuôn mặt này khó nhận dạng hơn để có thể dự đoán chính xác.

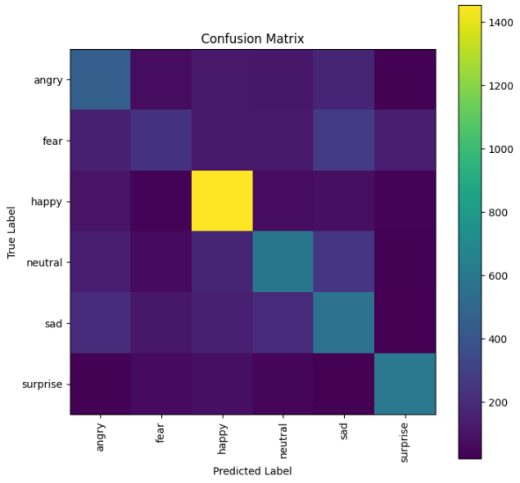
Classification Report

	precision	recall	f1-score	support
angry	0.42	0.47	0.45	958
fear	0.40	0.22	0.29	1024
happy	0.70	0.82	0.75	1774
neutral	0.51	0.47	0.49	1233
sad	0.42	0.45	0.43	1247
surprise	0.70	0.71	0.70	831
accuracy			0.55	7067
macro avg	0.52	0.52	0.52	7067
weighted avg	0.53	0.55	0.54	7067

Dựa vào bảng trên, ta có thể thấy được độ chính xác của từng cảm xúc. Với Happy là cảm xúc được dự đoán đúng nhiều nhất (Lý do là từ biểu đồ ban đầu ta có thể thấy được số mẫu trong Happy nhiều hơn so với số mẫu còn lại).

2.2 Confusion Matrix

Confusion Matrix



3 Một số nhận định về kết quả:

3.1 Hiệu suất mô hình

- Mô hình CNN đã đạt được độ chính xác tốt hơn các mô hình trước đó (MLP, Logistic Regression, Decision Tree, KNN). Độ chính xác trên tập test đạt khoảng 55-60%, cao hơn so với kết quả từ các phương pháp khác.
- Mặc dù có tiến bộ đáng kể, độ chính xác vẫn còn hạn chế, đặc biệt đối với các nhãn có ít mẫu dữ liệu.

3.2 Phân bố dữ liệu và ảnh hưởng đến dự đoán

- Cảm xúc "Happy" đạt độ chính xác cao nhất, điều này có thể được lý giải bởi số lượng mẫu trong lớp này lớn hơn các lớp khác, giúp mô hình học tốt hơn đặc điểm của nhãn này.
- Một số cảm xúc có ít dữ liệu hơn (như "Disgust") dẫn đến hiệu suất kém hoặc không khả thi để dự đoán.

3.3 Confusion Matrix

- Từ Confusion Matrix, dễ dàng nhận thấy mô hình gặp khó khăn trong việc phân biệt giữa các cảm xúc có đặc điểm biểu cảm khuôn mặt tương tự nhau.
- Một số nhãn thường bị nhầm lẫn với nhãn khác, cho thấy mô hình có thể chưa đủ phức tạp để nắm bắt sự khác biệt tinh tế trong dữ liệu.

3.4 Hạn chế của mô hình

- Dữ liệu không cân bằng (số lượng mẫu giữa các nhãn khác nhau), hoặc là bị overfitting là một yếu tố lớn ảnh hưởng đến hiệu suất.
- Mặc dù mô hình CNN phù hợp với dữ liệu hình ảnh, kết quả hiện tại cho thấy cần cải thiện thêm để tối ưu hóa hiệu suất.

3.5 Đề xuất cải thiện

- **Tăng cường dữ liệu:** Áp dụng các kỹ thuật như tăng cường dữ liệu (*data augmentation*) để tạo thêm mẫu cho các nhãn có ít dữ liệu.
- **Điều chỉnh kiến trúc mô hình:** Sử dụng các kiến trúc phức tạp hơn như ResNet, Inception hoặc VGG để cải thiện khả năng học đặc trưng.
- **Xử lý mất cân bằng dữ liệu:** Áp dụng các phương pháp như trọng số lớp (*class weighting*) hoặc tái lấy mẫu (*oversampling/undersampling*).
- **Tăng số lượng epoch:** Huấn luyện mô hình trong nhiều epoch hơn, kết hợp với kỹ thuật giảm tốc độ học (*learning rate scheduling*) để cải thiện khả năng tối ưu.

4 Hướng phát triển

Hướng phát triển của đề tài này là có thể phân tích các đoạn video, để nhận diện được cảm xúc của chủ thể trong video đó.

5 Tài liệu tham khảo