NHẬP MÔN TRÍ TUỆ NHÂN TẠO

Thành viên

Nguyễn Thế Phong

Trần Anh Minh

Dương Xuân Đức

Phùng Phúc Hậu

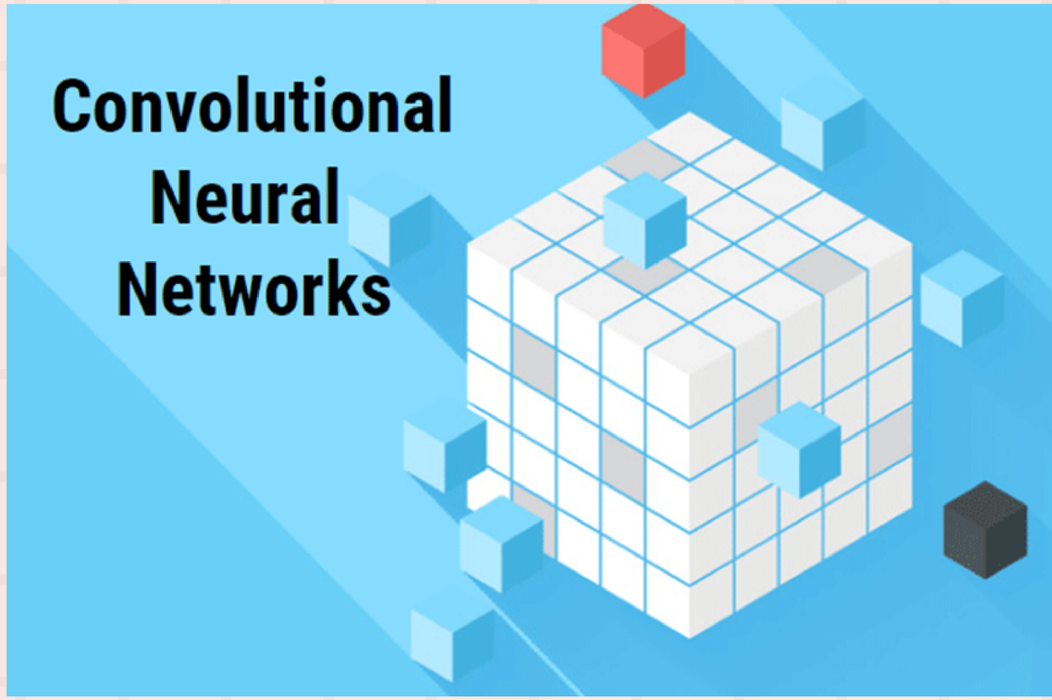
Đinh Khả Vy

NỘI DUNG

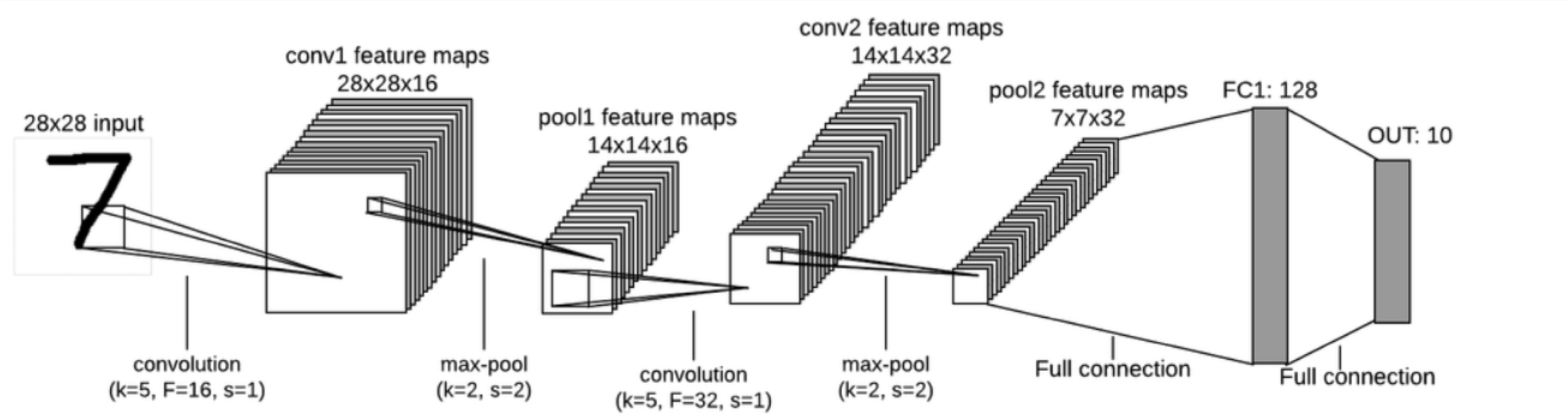
- Giới thiệu tổng quát đề tài
- Thuật toán CNN
- Model Simple CNN
- Model mini_Xception
- Kết quả

GIỚI THIỆU VỀ ĐỀ TÀI

Thuật toán CNN (Convolutional Neural Network)



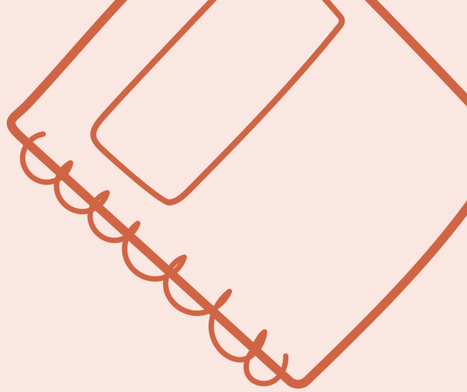
- Thuật toán CNN là một kiểu mạng nơ-ron nhân tạo được thiết kế đặc biệt để xử lý các dữ liệu dạng ảnh, âm thanh hoặc video.
- Điểm nổi bật của CNN là khả năng tự học được đặc tính của một bức ảnh, trong đó cấu trúc mạng được thiết kế để tìm ra các đặc tính này.



Các bước chính của thuật toán CNN bao gồm:

1. Convolution layer: Áp dụng các bộ lọc (filters) để quét và trích xuất các đặc tính từ ảnh đầu vào.
2. Activation layer: Sử dụng hàm kích hoạt để tính toán giá trị đầu ra của mỗi đặc tính.
3. Pooling layer: Giảm kích thước không gian của đầu ra từ các lớp trước đó và ngăn chặn overfitting.
4. Fully-connected layer: Học cách phân loại đối tượng dựa trên đặc tính đã trích xuất từ các lớp trước đó.

Model Simple CNN



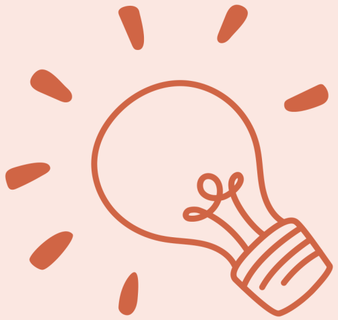
```
[ ]  def simple_CNN(input_shape, num_classes):
         model = Sequential()
         model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same',
                                 name='image_array', input_shape=input_shape))
         model.add(BatchNormalization())
         model.add(Convolution2D(filters=16, kernel_size=(7, 7), padding='same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
         model.add(Dropout(.5))

         model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
         model.add(BatchNormalization())
         model.add(Convolution2D(filters=32, kernel_size=(5, 5), padding='same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
         model.add(Dropout(.5))

         model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
         model.add(BatchNormalization())
         model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
         model.add(Dropout(.5))
```

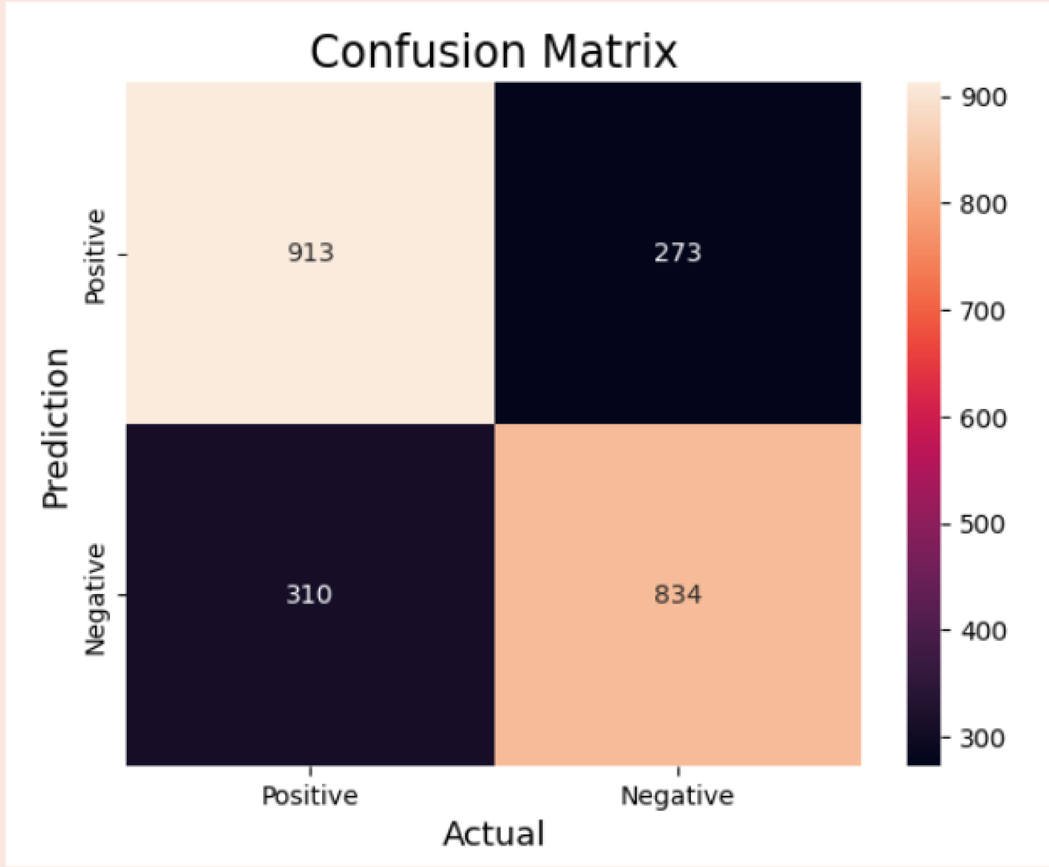
```
model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Convolution2D(filters=128, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(AveragePooling2D(pool_size=(2, 2), padding='same'))
model.add(Dropout(.5))

model.add(Convolution2D(filters=256, kernel_size=(3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Convolution2D(
    filters=num_classes, kernel_size=(3, 3), padding='same'))
model.add(GlobalAveragePooling2D())
model.add(Activation('softmax', name='predictions'))
return model
```

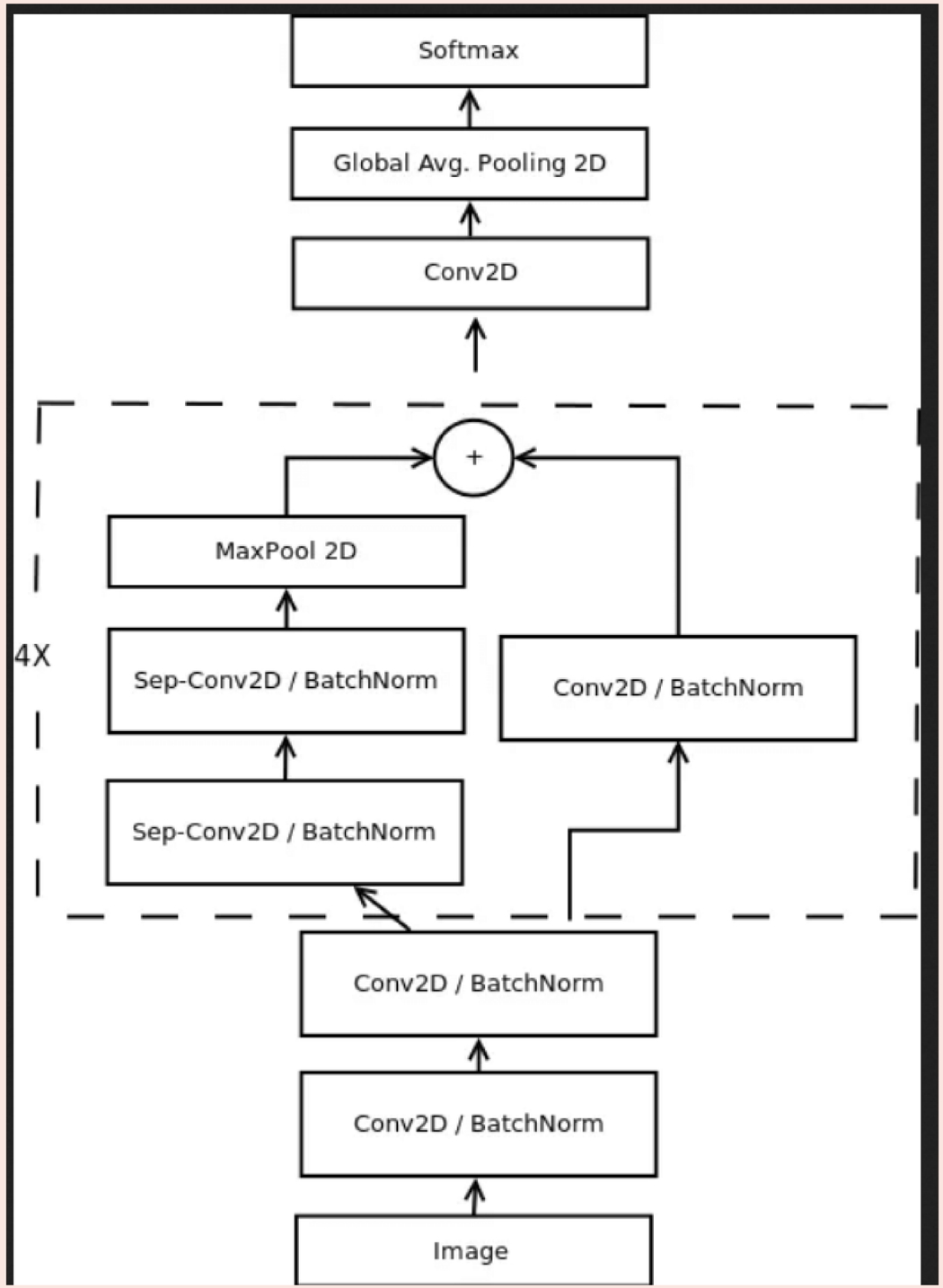


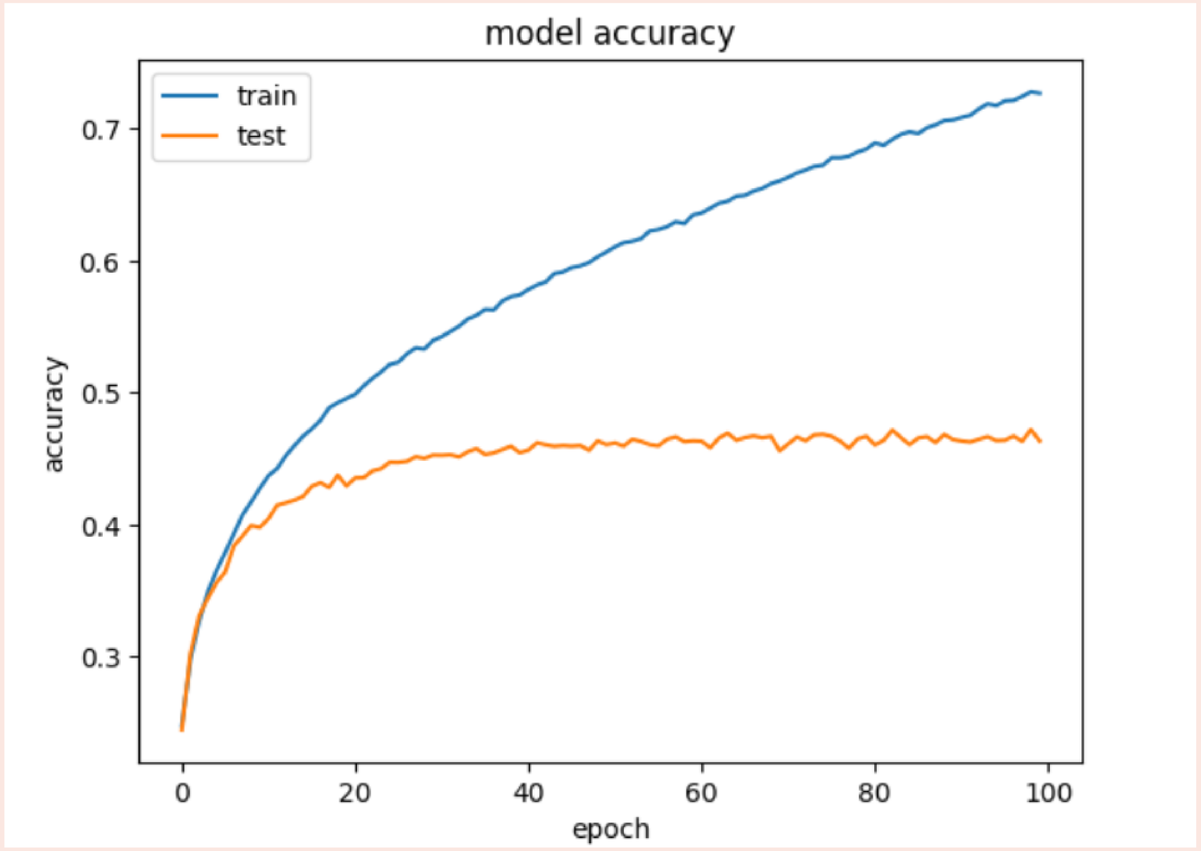
Đánh giá mô hình

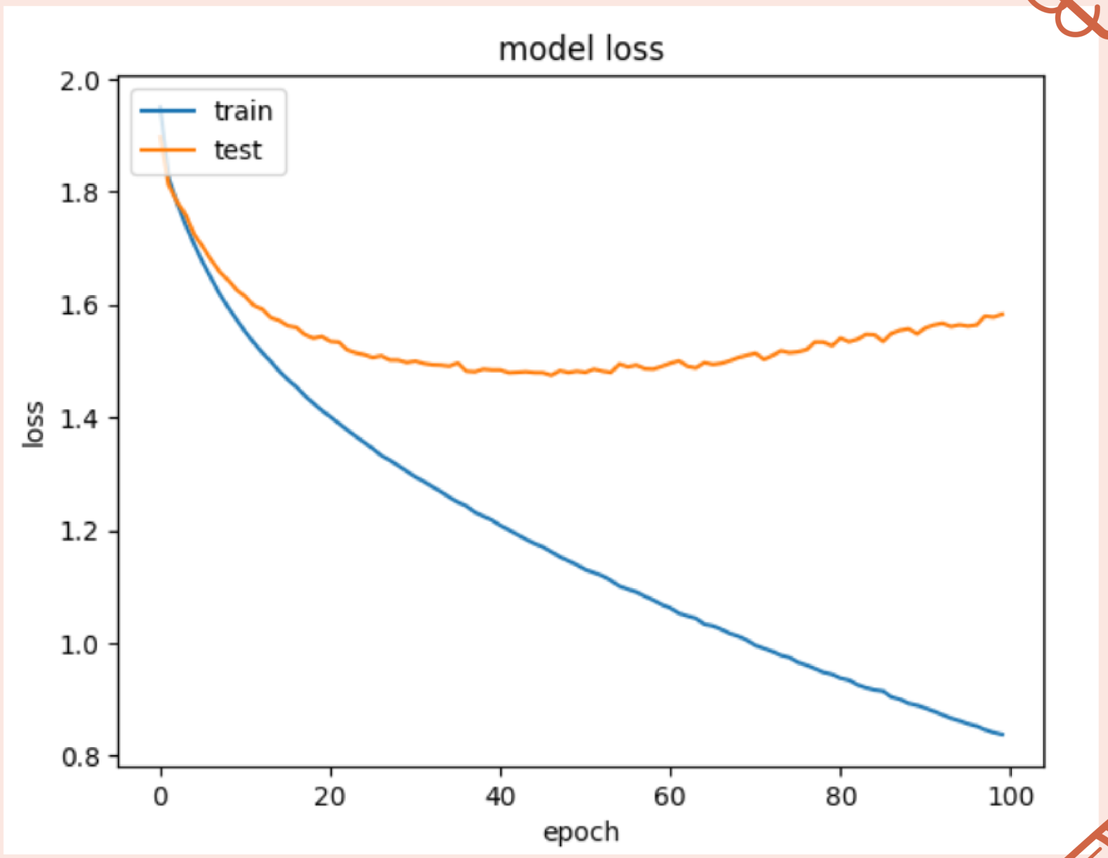
	precision	recall	f1-score	support
0	0.75	0.77	0.76	1186
1	0.75	0.73	0.74	1144
accuracy			0.75	2330
macro avg	0.75	0.75	0.75	2330
weighted avg	0.75	0.75	0.75	2330



Model mini_Xception

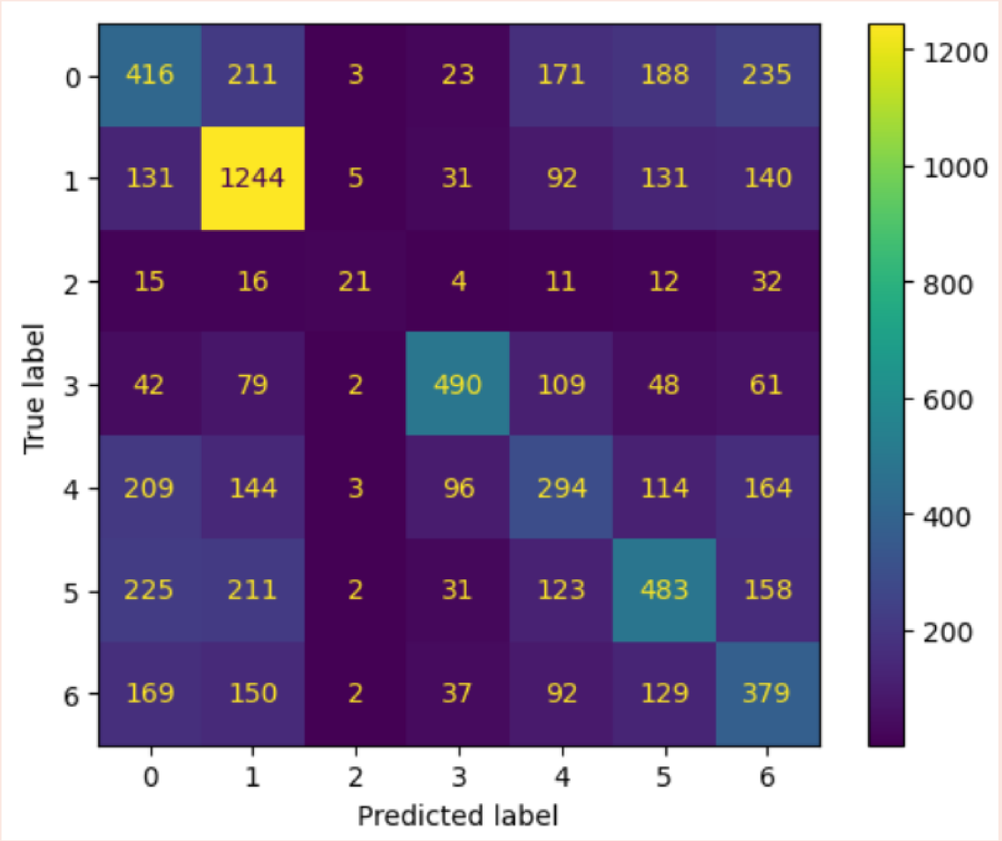

Softmax
Global Avg. Pooling 2D
Conv2D
+
MaxPool 2D
4X
Sep-Conv2D / BatchNorm
Conv2D / BatchNorm
Sep-Conv2D / BatchNorm
Conv2D / BatchNorm
Conv2D / BatchNorm
Image


model accuracy
train
test
accuracy
epoch
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100


model loss
train
test
loss
epoch
0.8
1.0
1.2
1.4
1.6
1.8
2.0
0
20
40
60
80
100

Đánh giá mô hình

	precision	recall	f1-score	support
0	0.34	0.33	0.34	1247
1	0.61	0.70	0.65	1774
2	0.55	0.19	0.28	111
3	0.69	0.59	0.64	831
4	0.33	0.29	0.31	1024
5	0.44	0.39	0.41	1233
6	0.32	0.40	0.36	958
accuracy			0.46	7178
macro avg	0.47	0.41	0.43	7178
weighted avg	0.46	0.46	0.46	7178



THANK YOU