**TÓM TẮT QUÁ TRÌNH TRAIN MODEL - CLASSIFIER**

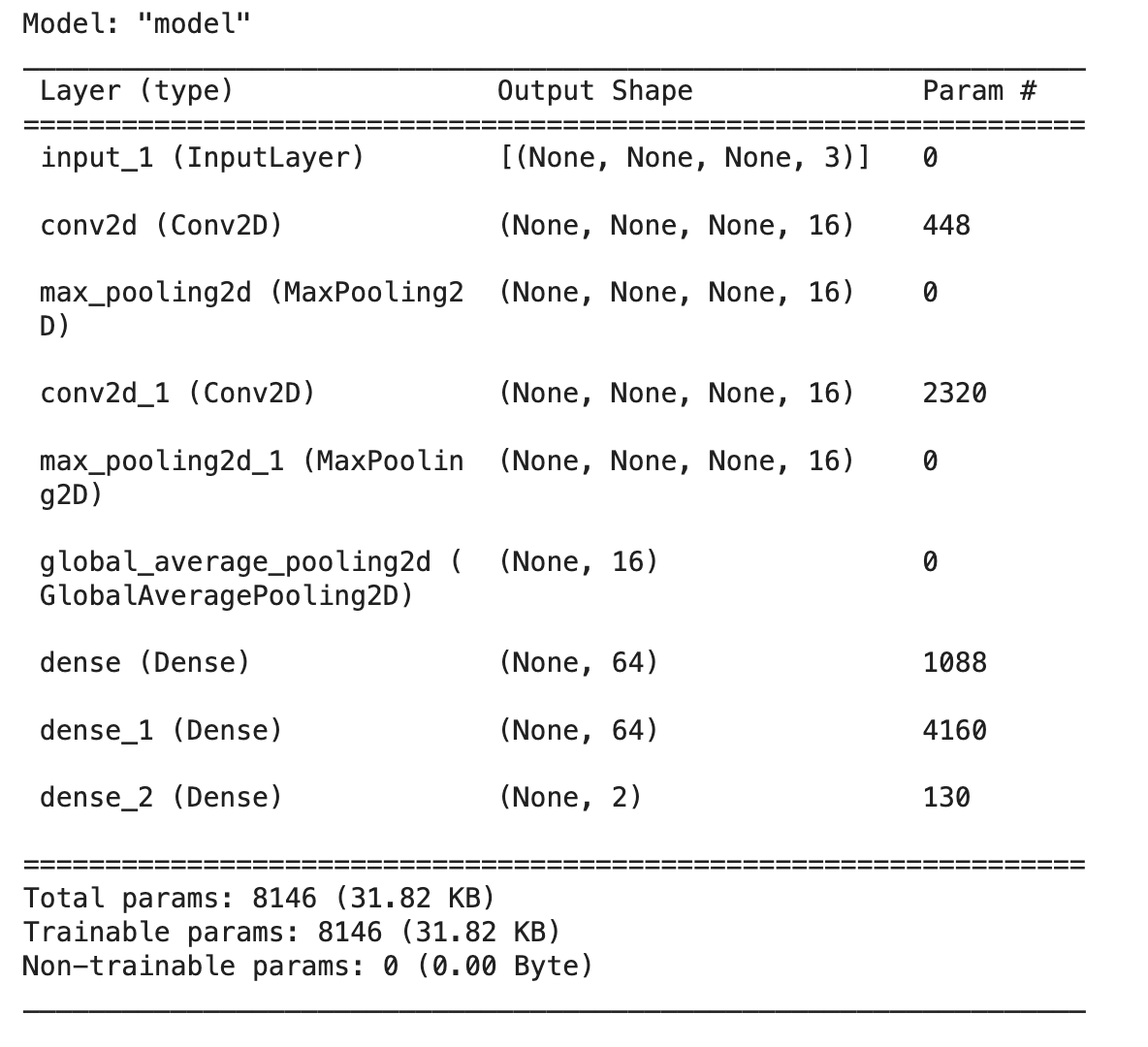
1. **Dataset:** Tập data đã được chia sẵn Train - Test Set. Mỗi folder bao gồm các file ảnh và một file CSV chứa tên ảnh, kích cỡ và Label của ảnh đó (bao gồm 2 label “Motor" và “Bycycle")
2. **Preprocess Data:**

* Lấy từ file csv và các link ảnh, chia ra các tập X\_train, X\_test, y\_train, y\_test
* X\_train có shape (625, 416, 416, 3) (chế độ màu RGB)
* Biểu diễn một số dữ liệu mẫu
* Áp dụng Label Encoder và One-hot Encoding cho y\_train và y\_test để có thể chuyển sang vector, áp dụng được cho quá trình Training (khi predict sẽ ra một vector 2 phần tử là float, sau đó chúng ta sẽ chọn argmax và đảo ngược LabelEncoder để ra kết quả dự đoán gốc)

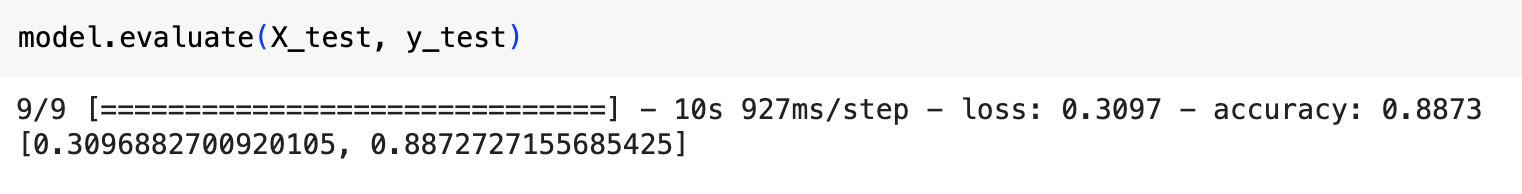
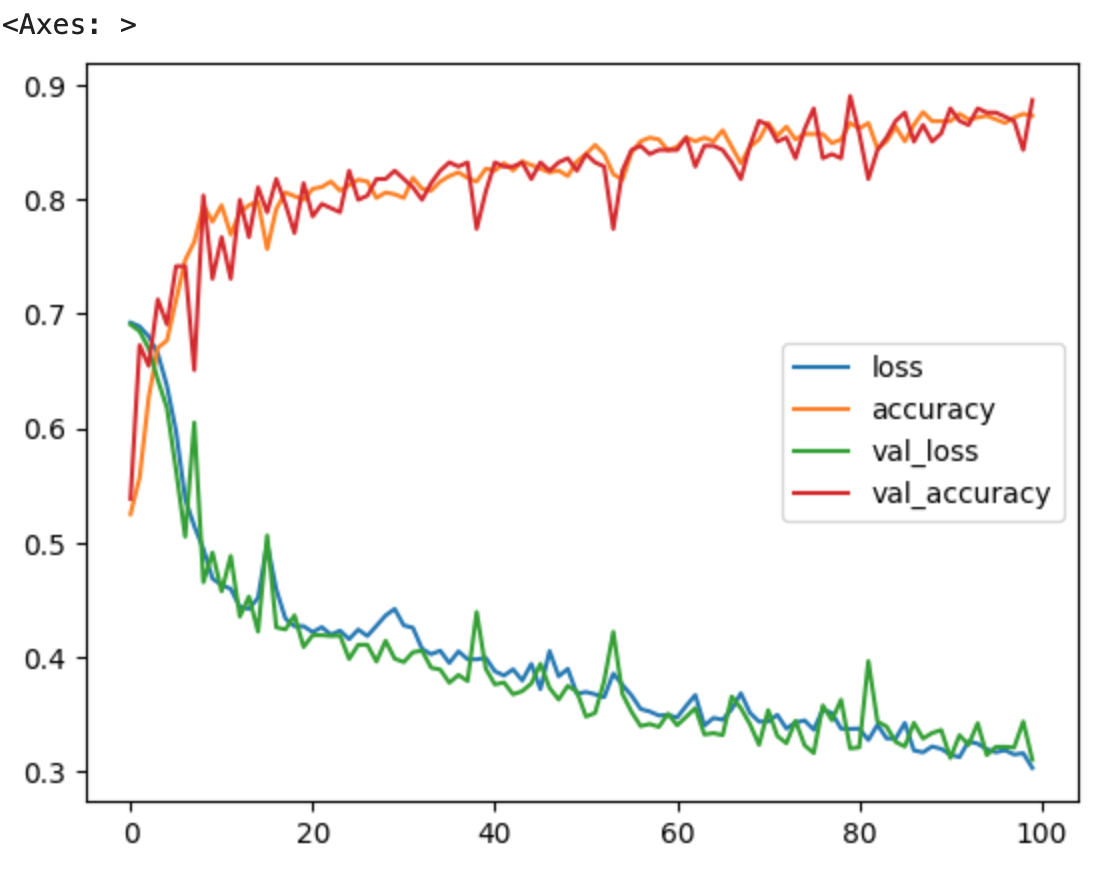
1. **Training Process:**

* Tạo model với Neural Networks và CNNs với Keras bao gồm các lớp:
* Đầu vào input là khối (None, None, 3): kích thước không cố định
* Lớp đầu tiên là lớp tích chập 2 chiều Conv2D với filter = 16 (16 lần chuyển toàn bộ hình ảnh), kích thước hạt nhân kenel\_size = 3x3.
* Sau đó, sử dụng Maxpool2D để giảm kích thước ảnh xuống một nửa.
* Tương tự ta lại sử dụng Conv2D với filter = 32 và Maxpool2D để có được 32 tính năng, kích thước tiếp tục giảm một nửa.
* Tiếp theo sử dụng GlobalAveragePooling2D để tính toán trung bình trên 2 chiều để đưa ra duy nhất 32 tính năng cuối cùng.
* Cuối cùng là tạo ra mạng lưới thần kinh 2 lớp với 64 tế bào.
* Đầu ra chỉ xuất ra 1 giá trị với activation= ‘sigmoid’
* Sử dụng hàm loss là BinaryCrossentropy vì đây chỉ cần phân loại 2 object
* Hàm Optimizers sử dụng Adam và số epoch là 100

=> Hình ảnh model.summary() với X\_train



1. **Evaluate Model:**

* Sử dùng model.evaluate được độ chính xác
* 
* Dùng DataFrame để có thể biểu diễn loss và accuracy của model
* Visualize các thông số để đánh giá tốt hơn: nhận thấy hàm loss giảm dần, nhanh sau mỗi epoch

1. **Predict:**

* Load model đã được lưu trong Google Drive
* Lấy link ảnh từ google và dùng thư viện request, OpenCV, Numpy để chuyển đổi ảnh sang chế độ màu RGB và numpy array, sau đó chuyển sang float để có thể lấy làm input để predict
* Sử dụng model để dự đoán, sau đó dùng dùng argmax và inverse\_transform để đảo ngược quá trình LabelEncoder => output của hình ảnh là “Bycycle" hay “Motor"