Báo cáo Vision AI intern assignment

Họ và tên: Trần Phú Vinh

Exercise 1 – Image Classification

1) Dataset

Em sử dụng dataset “cats_vs_dogs” được cung cấp sẵn tại tfds của Tensorflow (https://www.tensorflow.org/datasets/catalog/cats_vs_dogs) . Dataset cung cấp tập train bao gồm 23262 ảnh chó và mèo. Do dataset có số lượng ảnh nhiều nên em không sử dụng augmentation để bổ sung ảnh. Em sử dụng 80% ảnh cho việc huấn luyện (train) và 20% ảnh sử dụng để đánh giá (val). Tỉ lệ các lớp được đảm bảo sau khi chia tập train val. Ảnh được resize thành kích thước 224 x 224 trước khi đưa vào mô hình. Nhãn sẽ được chuyển thành dạng one-hot. [1, 0] nếu ảnh thuộc lớp mèo, [0. 1] nếu ảnh thuộc lớp chó.

2) Model

Em sử dụng model ResNet50 với weight được huấn luyện trước (pre train) trên tập ImageNet. Hàm keras.applications.resnet.preprocess_input được dùng để đảm bảo ảnh mà mô hình sắp dùng giống như ảnh mà mô hình được huấn luyện trước. Lớp global average pooling được dùng để giảm chiều đặc trưng do ResNe50 rút trích, giúp giảm các tham số cần được học ở lớp dense, giúp giảm trường hợp overfitting. Do nhãn thuộc dạng one-hot, nên lớp dense cuối có 2 neuron với hàm kích hoạt là softmax. Mỗi neuron sẽ cho giá trị xác xuất lớp mà ảnh thuộc về. Để chuyển từ dự đoán dưới dạng xác xuất thành nhãn chó hoặc mèo, em sử dụng hàm argmax để chọn lớp có xác xuất cao nhất. Sử dụng 2 neuron thay vì 1 neuron + hàm kích hoạt sigmoid thì sẽ không cần điều chỉnh ngưỡng threshold (< 0.5 là mèo, >= 0.5 là chó). Ngoài ra, các tham số của backbone ResNet50 được freeze (không được cập nhật trong quá trình huấn luyện) giúp cho tham số của lớp dense làm quen với feature do ResNet50 trích xuất.

3) Training

Em sử dụng hàm tối ưu (optimizer) SGD với learning_rate = 0.0001, hàm mất mát là categorical cross entropy, metric sử dụng để đánh giá quá trình training là accuracy. Em sử dụng các callback sau: giảm learning_rate ngay tức khắc khi sau 3 epoch accuracy trên tập val không tăng 0.05; early stopping giúp kết thúc sớm quá trình huấn luyện nếu

accuracy trên tập val không tăng 0.05 sau 5 epoch; model checkpoint giúp lưu mô hình có accuracy trên tập val cao nhất.

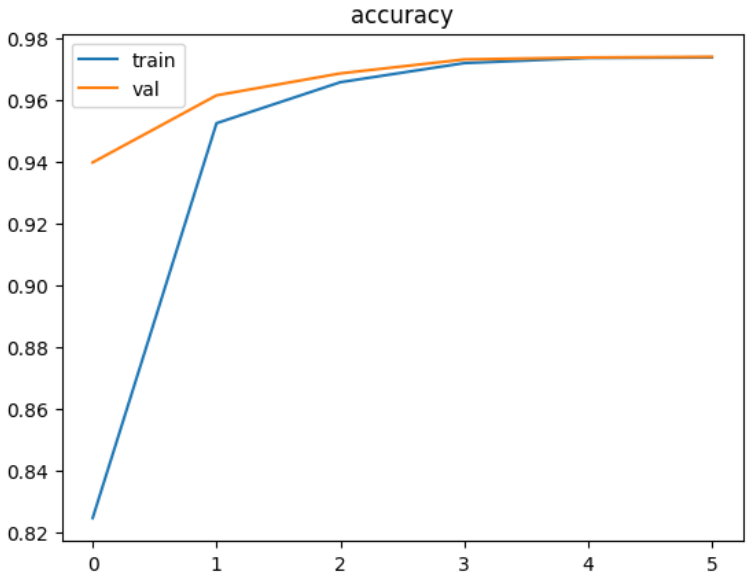
Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 224, 224, 3)	0	-
get_item (GetItem)	(None, 224, 224)	0	input_layer[0][0]
get_item_1 (GetItem)	(None, 224, 224)	0	input_layer[0][0]
get_item_2 (GetItem)	(None, 224, 224)	0	input_layer[0][0]
stack (Stack)	(None, 224, 224, 3)	0	get_item[0][0], get_item_1[0][0], get_item_2[0][0]
add (Add)	(None, 224, 224, 3)	0	stack[0][0]
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712	add[0][0]
global_average_poo… (GlobalAveragePool…	(None, 2048)	0	resnet50[0][0]
dense (Dense)	(None, 2)	4,098	global_average_p…

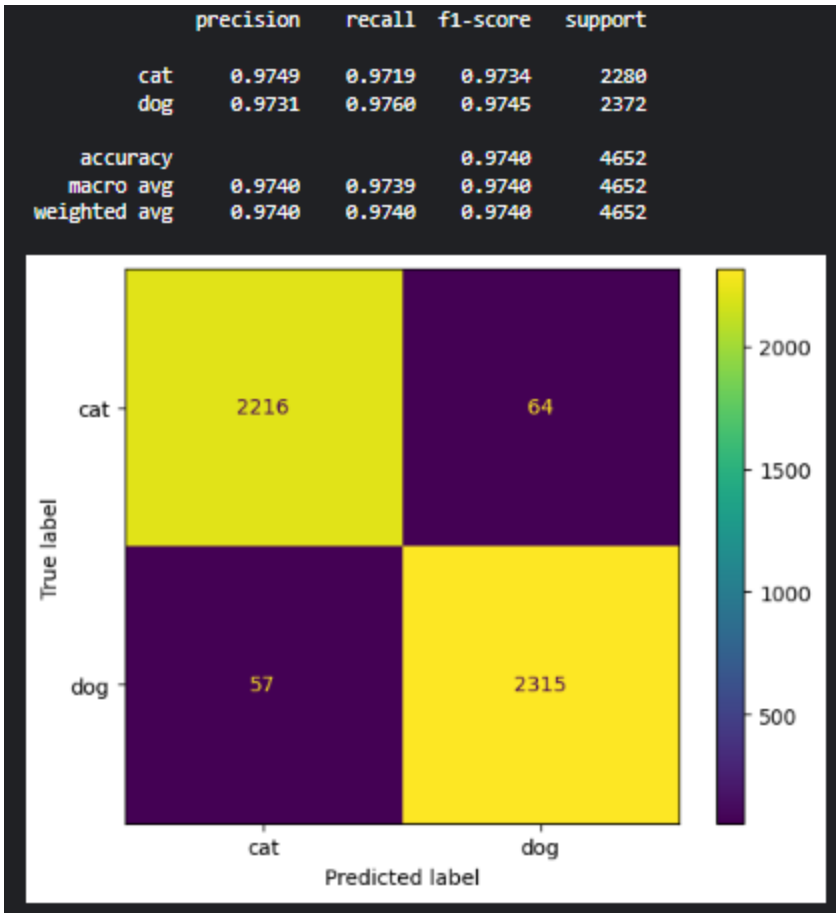
Các lớp của mô hình.

4) Result

Mô hình đạt 0.9740 accuracy trên tập val sau 6 epoch. Kết quả f1 đạt 0.9740 trên tập val. Tuy nhiên, mô hình vẫn còn một số nhầm lẫn giữa ảnh chó và mèo.



Accuracy trên tập val và train sau từng epoch.

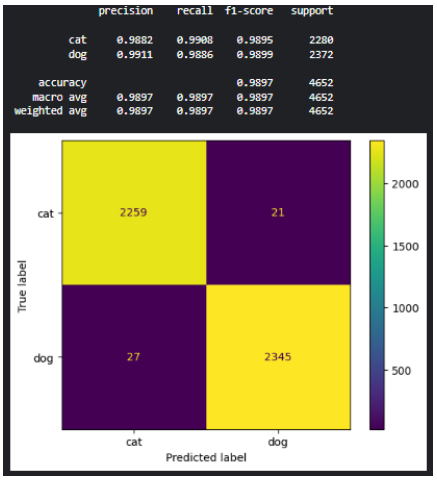


Kết quả precision, recall, f1 score và confusion matrix.

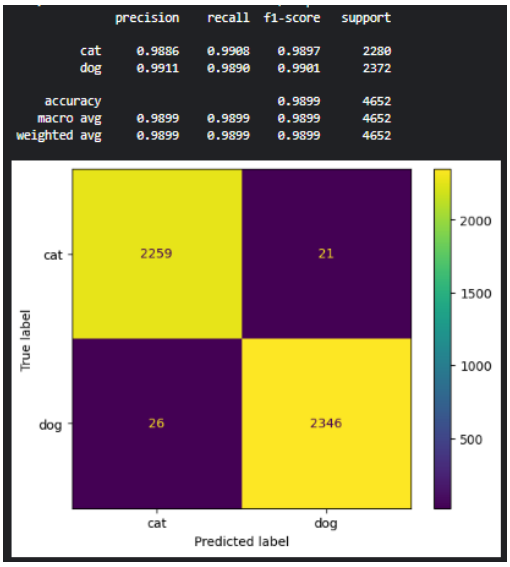
5) Improving result

Em sử dụng hai hàm mất mát là Adam và RMSprop để huấn luyện tiếp mô hình có kết quả trên. Cả hai hàm loss dùng learning_rate = 0.0001 và sử dụng các callback như trên.

- Adam giúp cải thiện kết quả accuracy trên val từ 0.9740 lên 0.9903.



- RMSprop giúp cải thiện kế quả accuracy trên val từ 0.9740 lên 0.9899.



Mặc dù kết quả accuracy có cải thiện khi sử dụng hàm loss Adam và RMSprop để huấn luyện mô hình tiếp nhưng mô hình vẫn còn nhầm lẫn một số ảnh chó mèo. Có thể phân tích các ảnh bị nhầm lẫn, xác định vùng ảnh nào ảnh hưởng đến kết quả dự đoán bằng Class Activation Map, unfreeze backbone ResNet50,... để giải quyết hết các trường hợp nhầm lẫn này.

Exercise 2 – Text to speech

1) Dataset

Sử dụng các bộ dữ liệu tts được phát âm bởi người Việt. Có thể tiến hành xử lý lọc âm thanh nền, lọc các âm thanh bị lỗi, khó nghe. Các đoạn ghi âm nên được thực hiện bởi nhiều người thuộc nhiều vùng miền (ít nhất là từ 3 miền Bắc, Trung, Nam) giúp cho giọng đọc tts đa dạng hơn.

2) Data preprocessing

Chọn lọc các giọng đọc có nhiều đoạn thu âm nhất để dảm bảo tính nhất quán của giọng tts. Tạo embedding của giọng đọc giúp nắm bắt đặc tính của từng giọng đọc qua đó có thể xử lý các điểm nhấn mạnh, ngắt quãng, ... => giọng đọc tts có cảm xúc tự nhiên hơn. Đối với văn bản có thể dùng tokenizer, âm thanh thì dùng log-mel spectrogram

3) Model

Có thể sử dụng model FastSpeech2Conformer hay SpeechT5 và fine tune hoặc train lại từ đầu trên dữ liệu vừa xử lý.

4) Pipeline inference

Có thể dùng OCR để trích xuất văn bản từ PDFs, sách, website,... => tokenizer để trích xuất token dùng làm input cho model tts -> sử dụng vocoder để chuyển spectrogram mà model vừa output sang âm thanh