ĐẠI HỌC CÔNG NGHỆ THÔNG TIN – ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

BÁO CÁO ĐỒ ÁN

MÁY HỌC

ĐỀ TÀI

PHÁT HIỆN TAI NẠN GIAO THÔNG SỬ DỤNG MẠNG CNN KẾT HỢP SVM

Giảng viên hướng dẫn:

1. PGS-TS Lê Đình Duy
2. ThS Phạm Nguyễn Trường An

Nhóm báo cáo:

1. 19522492 – Huỳnh Thiện Tùng
2. 19522542 – Nguyễn Thành Vương
3. 19520954 – Lê Thị Thanh Thanh

TP HỒ CHÍ MINH, THÁNG 8 NĂM 2021

Mục lục

I. Tổng quan

1. Mô tả bài toán

Những năm gần đây, nước ta đã lắp đặt nhiều camera giám sát giao thông với mục đích xử lý vi phạm giao thông, cảnh báo tắc đường,… và một số thành phố có công khai phát trực tiếp tình trạng giao thông như TP Hồ Chí Minh (Cổng thông tin giao thông thành phố Hồ Chí Minh (hochiminhcity.gov.vn) và TP Đà Nẵng (Phát Triển Đà Nẵng - Xem Camera Trực Tuyến Đà Nẵng (0511.vn) và (45) Phát Triển Đà Nẵng - YouTube).

Từ những dữ liệu công khai đó, nhóm mong muốn phát triển mô hình phát hiện tai nạn giao thông dựa trên hình ảnh trích xuất từ camera giám sát trên cao với mục đích nhanh chóng phát hiện tai nạn giao thông để lực lượng chức năng có những xử lý kịp thời, tránh các trường hợp đáng tiếc.

Input:

1 hình ảnh trích xuất từ camera trên cao. (Ràng buộc: Trong hình phải có đường, có hoặc không có phương tiện giao thông, là địa điểm ở Việt Nam)

Ví dụ 1: Ví dụ về input

Output:

Hình ảnh input kèm dòng text:

- “accident” : nếu trong hình ảnh đầu vào có tai nạn.

- “non-accident”: nếu trong hình ảnh đầu vào không có tai nạn.

Ví dụ 2: Ví dụ về output

Ứng dụng hoàn chỉnh có thể được sử dụng trong hệ thống quản lý camera giao thông của các tỉnh thành ở VN với input là hình ảnh đến từ camera giám sát đó và output là đoạn text bao gồm thời gian, tuyến đường, địa điểm xảy ra tai nạn và hình ảnh lúc xảy ra tai nạn.

2. Mô tả dữ liệu

Bộ dữ liệu nhóm thu thập được bao gồm 4307 ảnh, trong đó có 2305 ảnh tai nạn và 2002 ảnh không tai nạn.

Dữ liệu được thu thập như sau:

- Các hình ảnh có tai nạn được thu thập qua các bài báo, trang mạng xã hội:
  + Báo Giao thông: https://www.baogiaothong.vn
  + Mạng xã hội giao thông: Giao thông Quốc gia (91.com.vn)
  + SOHA: Tin giao thông hôm nay, vu tai nan thảm khốc | soha.vn
  + Youtube: Bản Tin Tổ Lái - YouTube
- Các hình ảnh không có tai nạn được thu thập tại các trang:
  + Cổng thông tin giao thông TP Hồ Chí Minh: hochiminhcity.gov.vn
  + Camera trực tuyến Đà Nẵng: Phát Triển Đà Nẵng - 0511.vn
  + Hệ thống Camera giám sát giao thông thông minh: centic.vn

Hiện tại, ở Việt Nam, Có nhiều công ty, nhóm nghiên cứu đã làm đề tài này (như Trí Nam Group: Giao thông thông minh - ITS (trinam.com.vn)), tuy nhiên nhóm chưa tìm thấy những nghiên cứu công khai hay dữ liệu công khai nào có

liên quan. Đó cũng một phần lý do mà nhóm tự thu thập dữ liệu, ngoài ra, còn có những lý do sau:

- Dữ liệu phải đảm bảo đặt trong ngữ cảnh phù hợp (các ràng buộc trong phần mô tả bài toán)
- Qua quá trình thu thập dữ liệu, nhóm biết được những khó khăn và tầm quan trọng của việc này (thu thập và tiền xử lý dữ liệu tốn hơn 70% tổng thời gian thực hiện đề tài)
- Tiếp xúc nhiều với dữ liệu khiến nhóm có cái nhìn tổng quan hơn về dữ liệu, từ đó có thể tìm ra hướng giải quyết tốt hơn cho bài toán.

Những khó khăn của quá trình thu thập dữ liệu:

- Camera giao thông tại Việt Nam chỉ mới đi vào hoạt động vài năm trở lại đây nên hiện tại dữ liệu về tai nạn còn rất ít (Nhóm ước lượng thu thập được khoảng hơn 300 vụ tai nạn).
- Dữ liệu được thu thập từ nhiều nguồn, với rất nhiều dữ liệu không phù hợp ngữ cảnh nên nhóm phải thu thập thủ công, và xử lý dữ liệu cũng tốn nhiều thời gian.
- Dữ liệu được đăng tải theo hướng tin tức nên có nhiều ký hiệu đánh dấu vùng tai nạn (khoanh đỏ, thêm hiệu ứng,…), và chất lượng video rất thấp, với nhiều kích thước khác nhau.

II. Các nghiên cứu trước

Đây là một bài toán có tính thực tế cao nhưng dữ liệu còn hạn chế vì không được công khai dẫn đến không có nhiều nghiên cứu được tìm thấy. Ở Việt Nam, nhóm chưa tìm thấy những nghiên cứu công khai về đề tài này.

Dưới đây là nghiên cứu nhóm tham khảo về đề tài phân loại hình ảnh để phục vụ cho việc giải quyết bài toán của nhóm:

An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification (arxiv.org) của tác giả Abien Fred M. Agarap với các phiên bản:

[v1] Sun, 10 Dec 2017 14:50:28 UTC (486 KB)

[v2] Thu, 7 Feb 2019 06:25:08 UTC (484 KB)

Tóm tắt:

Bài báo này so sánh độ chính xác của model CNN và SVM trên cùng một tập dữ liệu. Kết quả chỉ ra rằng mô hình CNN-SVM có thể đạt được độ chính xác trên tập test là ~ 99,04% bằng cách sử dụng bộ dữ liệu MNIST còn CNN-Softmax có thể đạt được độ chính xác là ~ 99,23% . Cả hai model cũng được thử nghiệm trên tập dữ liệu Fashion-MNIST (được cho là tập dữ liệu phân loại hình ảnh khó hơn MNIST). Kết quả là CNN-SVM đạt độ chính xác kiểm tra ~ 90,72%, trong khi CNN-Softmax đạt độ chính xác kiểm tra ~ 91,86%.

Bộ dataset trong nghiên cứu này với dataset không được tiền xử lý, không chuẩn hóa hoặc giảm chiều dữ liệu.

Table 1: Dataset distribution for both MNIST and Fashion-MNIST.

Dataset	MNIST	Fashion-MNIST
Training	60,000	10,000
Testing	60,000	10,000

Both datasets were used as they were, with no preprocessing such as normalization or dimensionality reduction.

Kết quả quá trình thử nghiệm trên MNIST và Fashion-MNIST mang lại độ chính xác rất cao:

Table 3: Test accuracy of CNN-Softmax and CNN-SVM on image classification using MNIST[10] and Fashion-MNIST[13].

Dataset	CNN-Softmax	CNN-SVM
MNIST	99.23%	99.04%
Fashion-MNIST	91.86%	90.72%

Xét riêng model CNN-SVM, kết quả rất cao với độ chính xác 99.04% trên bộ dataset MNIST và 90.72% trên dataset Fashion-MNIST.

Ngoài ra nhóm còn nghiên cứu thêm về Transfer Learning để sử dụng cho đồ án này. Phần này sẽ được trình bày kỹ ở tại chương 4.

III. Xây dựng bộ dữ liệu

Với mục tiêu thu thập dữ liệu đúng ngữ cảnh đồ án, nhóm đã lên kế hoạch sưu tầm các vụ tai nạn giao thông được ghi lại từ camera tại Việt Nam. Dữ liệu thô nhất được sưu tầm từ các video bản tin thời sự liên quan đến tai nạn giao thông được trích xuất từ camera giao thông trên các tuyến đường trên cả nước.

Vấn đề khó khăn lớn nhất chính là góc quay từ camera. Để có được tấm ảnh như mong muốn của nhóm đặc ra như là: góc quay từ trên cao, không bị các tác nhân ngoại cảnh như là bị cây cối hoặc bất cứ vật gì che khuất tầm nhìn camera, ảnh phải có nội dung về giao thông đường bộ Việt Nam,…

Nhóm đã thu thập được hơn 300 vụ tai nạn giao thông từ các nguồn như bản tin thời sự, các video trên Youtube có nội dung tương tự. Sau đó nhóm sẽ cắt ảnh từng vụ tai nạn từ 5 – 8 ảnh theo trên từng khuôn hình khác nhau.

Bộ dataset có tổng cộng 4307 ảnh, trong đó:

Class accident	2305 ảnh
Class non-accident	2002 ảnh

Nhóm viết một đoạn script nhỏ để cắt video ra thành nhiều frame theo thời gian:

```python
import cv2


def video_to_images(inp, out):
    video_cap = cv2.VideoCapture(inp)
    success, image = video_cap.read()
    count = 0
    while success:
        video_cap.set(cv2.CAP_PROP_POS_MSEC,(count*1000))
        cv2.imwrite(out + "vid01_frame%d.jpg" % count, image)
        success, image = video_cap.read()
        print('Read a new frame: ', success)
        count += 1


inp = "video2 (8).mp4"
out = "video16/"
video_to_images(inp=inp, out=out)
```

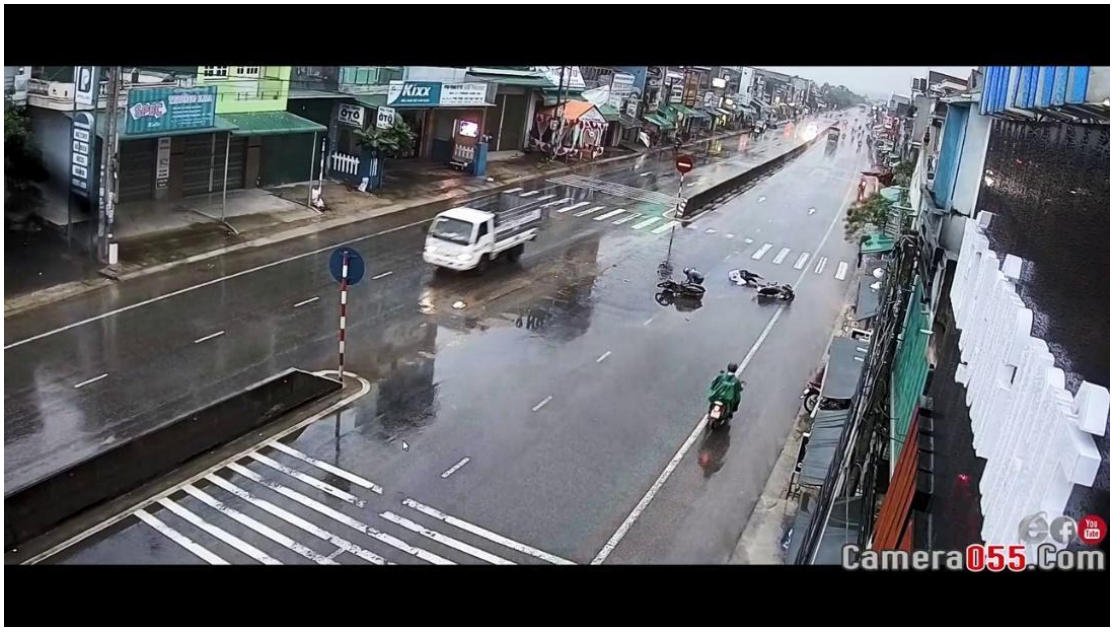
Vì hình ảnh được trích xuất từ camera giao thông hoặc lấy từ báo chí có chất lượng không cao, hình ảnh bị ảnh hưởng bởi ánh nắng, sương mù,… nên nhóm có thử nâng cao chất lượng hình ảnh bằng thuật toán Retinex. Tuy nhiên, kết quả không mấy khả quan trên bộ dữ liệu của nhóm.

Hình ảnh ban đầu:

Hình ảnh sau khi sử dụng Retinex:

Qua thử nghiệm một số hình ảnh chất lượng kém, nhận thấy hình ảnh không cải thiện mấy, và cũng mất nhiều thời gian (mỗi hình ảnh phải xử lý mất hơn 2 phút, và việc chọn 1 ảnh tốt nhất trong 3 ảnh cũng mất thời gian) nên nhóm giữ nguyên ảnh gốc để train.

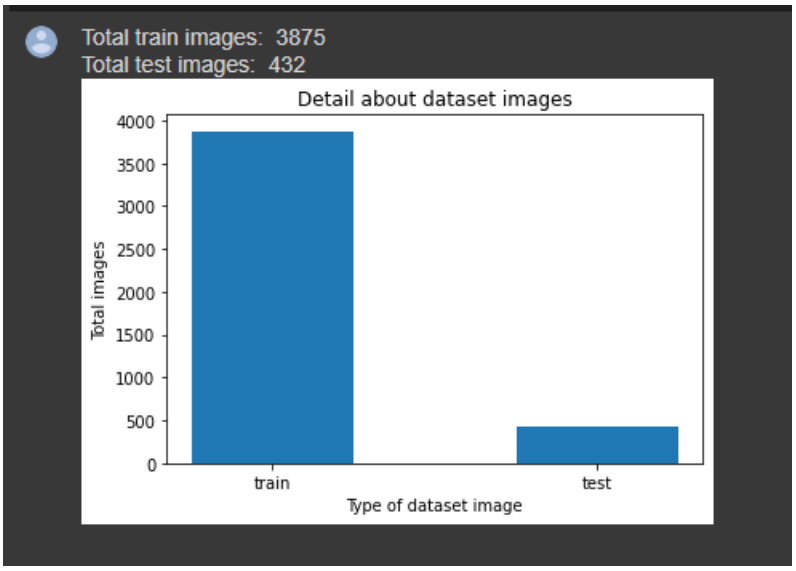
Dưới đây là một số hình ảnh trong bộ dữ liệu:



Ví dụ 3: Một số hình ảnh trong bộ dữ liệu

Về định dạng, 100% hình ảnh trong tập dữ đều có định dạng JPGE.

Tỉ lệ dataset cho mục train/test là 9/1 với 90% cho quá trình training (trong đó 80% làm tập train và 20% làm tập validation), 10% còn lại cho quá trình testing.



IV. Training và đánh giá model

Các mô hình sẽ dụng trong đồ án:

- Mô hình CNN-SVM.
- Mô hình CNN có Transfer Learning.

1. Mô hình CNN-SVM (không tăng cường dữ liệu)

a. Tổng quan mô hình

- Mô hình này áp dụng CNN để tận dụng khả năng trích xuất đặc trưng của ảnh thông qua các hidden layers kết hợp thuật toán SVM để phân loại lớp bằng cách thiết lập giá trị regulizer và hàm loss tại output layer trên tập dữ liệu không tăng cường.
- Thời gian train: 50 phút.
- Thời gian test: 17 phút.

b. Thiết lập mô hình

Đầu tiên, thiết lập đường dẫn của dataset và giới hạn ảnh đầu vào là 224x224 pixels, bởi đa số ảnh trong tập train của cả 2 class đều là ảnh từ camera giám sát giao thông chụp lại nên sẽ có kích thước khá lớn, vậy nên nếu ảnh đầu vào sẽ được resize lại thành ảnh 224 x 224 pixels. Việc giới hạn ảnh điểm ảnh sẽ

giúp quá trình train model diễn ra nhanh hơn. Ngoài ra nhóm còn thực hiện chuẩn hoá dữ liệu (Data Normalization) để đưa tất cả các giá trị của từng pixel về khoảng [0;1]. Các ảnh đầu vào đa số đều là ảnh màu (RGB) sẽ có hệ số từ 0-255 nhưng các giá trị trong khoảng này lại quá lớn cho model để model có thể xử lý. Bước chuẩn hoá dữ liệu sẽ giúp tất cả các ảnh dù là ảnh màu hay ảnh trắng đen sẽ được đưa về cùng 1 kiểu dữ liệu nằm trong khoảng [0;1].

```
data_root = ("/content/train/")
IMAGE_SHAPE = (224, 224) # (height, width) in no. of pixels
TRAINING_DATA_DIR = str(data_root)
datagen_kwargs = dict(rescale=1./255, validation_split=.20)
```

Trước khi bước vào quá trình train model, chúng ta phải thực hiện 1 bước rất quan trọng, đó chính là phân chia dữ liệu từ 1 tập dữ liệu lớn cụ thể là phân chia thành tập train và tập valid theo tỉ lệ 80/20 tương ứng thông qua thư viện ImageDataGenerator của Keras.

```
# Part 1 - Data Preprocessing
train_datagen = ImageDataGenerator(**datagen_kwargs)

# Preprocessing the Training set
training_set = train_datagen.flow_from_directory(TRAINING_DATA_DIR, subset="training",
                                                 shuffle=False, target_size=IMAGE_SHAPE,
                                                 class_mode='binary', batch_size=32)

# Preprocessing the Test set
valid_datagen = ImageDataGenerator(**datagen_kwargs)
test_set = valid_datagen.flow_from_directory(TRAINING_DATA_DIR, subset="validation",
                                             shuffle=False, target_size=IMAGE_SHAPE,
                                             class_mode='binary', batch_size=32)
```

Tiếp theo nhóm bắt đầu xây dựng mạng nơ ron tích chập (Convolutional neural networks - CNN) kết hợp với thuật toán Support Vector Machine (SVM) bằng cách thêm từng layer một cách nối tiếp nhau nhưng riêng output layer chúng ta không dùng hàm kích hoạt softmax, thay vào đó là hàm kích hoạt linear.

```
# Part 2 - Building the CNN

# Initialising the CNN
model = tf.keras.models.Sequential()

# Step 1 - Convolution
model.add(tf.keras.layers.Conv2D(filters=32,padding="same",kernel_size=3, activation='relu', strides=2, input_shape=[224, 224, 3]))

# Step 2 - Pooling
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Adding a second convolutional layer
model.add(tf.keras.layers.Conv2D(filters=32,padding='same',kernel_size=3, activation='relu'))
model.add(tf.keras.layers.MaxPool2D(pool_size=2, strides=2))

# Step 3 - Flattening
model.add(tf.keras.layers.Flatten())

# Step 4 - Full Connection
model.add(tf.keras.layers.Dense(units=128, activation='relu'))

# Step 5 - Output Layer
model.add(Dense(1, kernel_regularizer=tf.keras.regularizers.l2(0.001),activation='linear'))

model.summary()
```

Tổng quan lại mô hình CNN-SVM mà nhóm đã train:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 112, 112, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 56, 56, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 28, 28, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 128)               3211392
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129
=================================================================
Total params: 3,221,665
Trainable params: 3,221,665
Non-trainable params: 0
```

Mô hình train được tất cả 3,221,665 tham số và không có tham số nào không train được.

Sau khi đã khởi tạo thành công model CNN-SVM, nhóm định nghĩa hàm loss và thuật toán optimizer và bắt đầu huấn luyện cho model:

```
# Part 3 - Training the CNN

# Compiling the CNN
model.compile(optimizer='adam', loss='hinge', metrics=['acc'])

# Stop training if 'val_loss' stops improving for over 5 epochs <=> Fix Overfitting problem
early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,verbose=1)

# Save the best model to .h5 file
best_model = keras.callbacks.ModelCheckpoint(filepath='best_model.h5',monitor='val_loss',save_best_only=True,verbose=1)

r=model.fit(x = training_set, validation_data = test_set, epochs = 20, callbacks=[early_stopping,best_model])
```

Model được huấn luyện với 20 epochs và kết hợp cơ chế early stopping để kết thúc sớm hơn. Trong quá trình huấn luyện, nếu 5 epochs liên tiếp không cải thiện được giá trị của val_loss thì huấn luyện sẽ kết thúc ngay tại epoch đó vì nếu có huấn luyện tiếp cũng không cải thiện được kết quả, phí thời gian. Việc này giúp kiệm thời gian huấn luyện. Ngoài ra, nhóm còn thiết lập cơ chế lưu model tốt nhất qua mỗi epoch bằng cách kiểm tra giá trị val_loss có cải thiện không thay cho cơ chế mặc định là lấy model tại epoch cuối cùng để dự đoán.

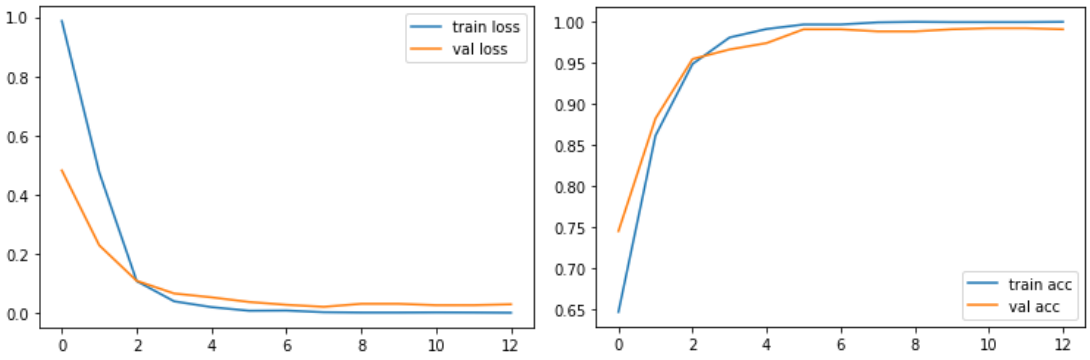
c. Kết quả

Sau khi đã huấn luyện và load model tốt nhất, nhóm đã đánh giá model trên tập valid và thu được độ chính xác **98,84%** và giá trị loss là **0.02**.

```
final_loss, final_accuracy = model.evaluate(test_set)
print("Final loss: {:.2f}".format(final_loss))
print("Final accuracy: {:.2f}%".format(final_accuracy * 100))
```

```
25/25 [==============================] - 11s 434ms/step - loss: 0.0211 - acc: 0.9884
Final loss: 0.02
Final accuracy: 98.84%
```

Để trực quan hơn, nhóm đã vẽ biểu đồ thể hiện độ chính xác và độ loss của model xuyên suốt quá trình huấn luyện cho đến khi xảy ra early stopping.



Qua 2 biểu đồ trên ta có thể nhận thấy:

- Xuyên suốt quá trình huấn luyện, giá trị train loss và val loss đều có xu hướng giảm xuống cho đến epoch thứ 9, tuy nhiên từ sau đó trở đi val_loss có xu hướng tăng giảm không đồng đều và giá trị val loss sau epoch thứ 9 đều cao hơn val loss tại epoch thứ 9 dẫn đến early stopping.
- Xuyên suốt quá trình huấn luyện, độ chênh lệch 2 cặp giá trị train loss – val loss và train acc - val acc là rất nhỏ.

Và cuối cùng là dự đoán của mô hình trên tập test:

```
Total correct prediction:  431
Total incorrect prediction:  1
```

Mô hình dự đoán đúng 431 ảnh trên tổng số 432 ảnh, đạt tỉ lệ chính xác **99,78%**.

Một số ảnh mà mô hình đã dự đoán:

Ảnh dự đoán sai:

2. Mô hình CNN-SVM (có tăng cường dữ liệu)

a. Tổng quan mô hình

- Sơ lược: Mô hình này áp dụng CNN để tận dụng khả năng trích xuất đặc trưng của ảnh thông qua các hidden layers kết hợp thuật toán SVM để phân loại lớp bằng cách thiết lập giá trị regulizer và hàm loss tại output layer trên tập dữ liệu tăng cường. Dữ liệu không tăng cường trên cả 2 tập train và valid, mà chỉ tăng cường trên tập train nhằm tăng kích thước tập train. Đây cũng chính là điểm khác biệt duy nhất khi thiết lập mô hình này với mô hình không tăng cường dữ liệu.
- Thời gian train: 1 tiếng 13 phút.
- Thời gian test: 17 phút.

b. Thiết lập mô hình

Nhóm tiến hành tăng cường dữ liệu trên tập train bằng các phương pháp như phóng to ngẫu nhiên, lật ảnh ngang,… Các tham số khác như rescale hay tỉ lệ chia tập train/valid đều hoàn toàn giống mô hình không tăng cường dữ liệu.

```
# Part 1 - Data Preprocessing
train_datagen = ImageDataGenerator(rescale = 1./255, validation_split=0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# Preprocessing the Training set
training_set = train_datagen.flow_from_directory(TRAINING_DATA_DIR, subset="training",
                                                 shuffle=False, target_size=IMAGE_SHAPE,
                                                 class_mode='binary', batch_size=32)

valid_datagen = ImageDataGenerator(**datagen_kwargs)
test_set = valid_datagen.flow_from_directory(TRAINING_DATA_DIR, subset="validation",
                                             shuffle=False, target_size=IMAGE_SHAPE,
                                             class_mode='binary', batch_size=32)
```

c. Kết quả

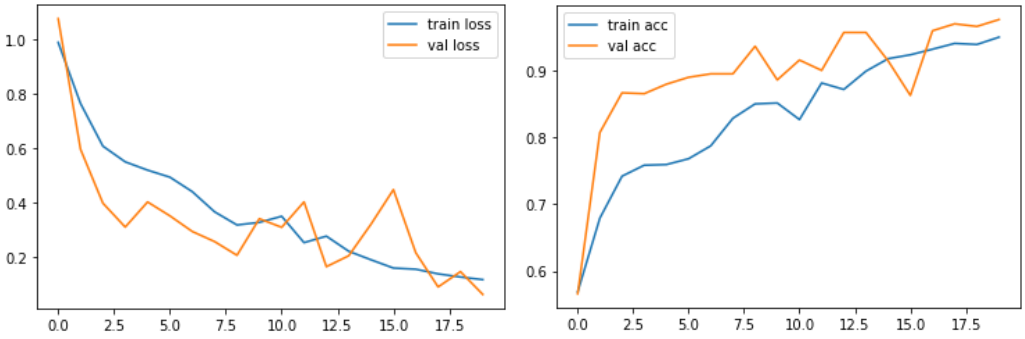
Sau khi đã thiết lập mô hình CNN-SVM, xây dựng các layers cho mô hình và huấn luyện mô hình. Nhóm load mô hình tốt nhất và đánh giá mô hình trên tập valid:

```
final_loss, final_accuracy = model.evaluate(test_set)
print("Final loss: {:.2f}".format(final_loss))
print("Final accuracy: {:.2f}%".format(final_accuracy * 100))
```

```
25/25 [==============================] - 11s 425ms/step - loss: 0.0605 - acc: 0.9767
Final loss: 0.06
Final accuracy: 97.67%
```

Mô hình sau khi đã tăng cường dữ liệu đạt độ chính xác **97,67%** và giá trị loss là **0.06.**

Trực quan quá trình training bằng đồ thị:



Qua 2 biểu đồ trên có thể nhận thấy:

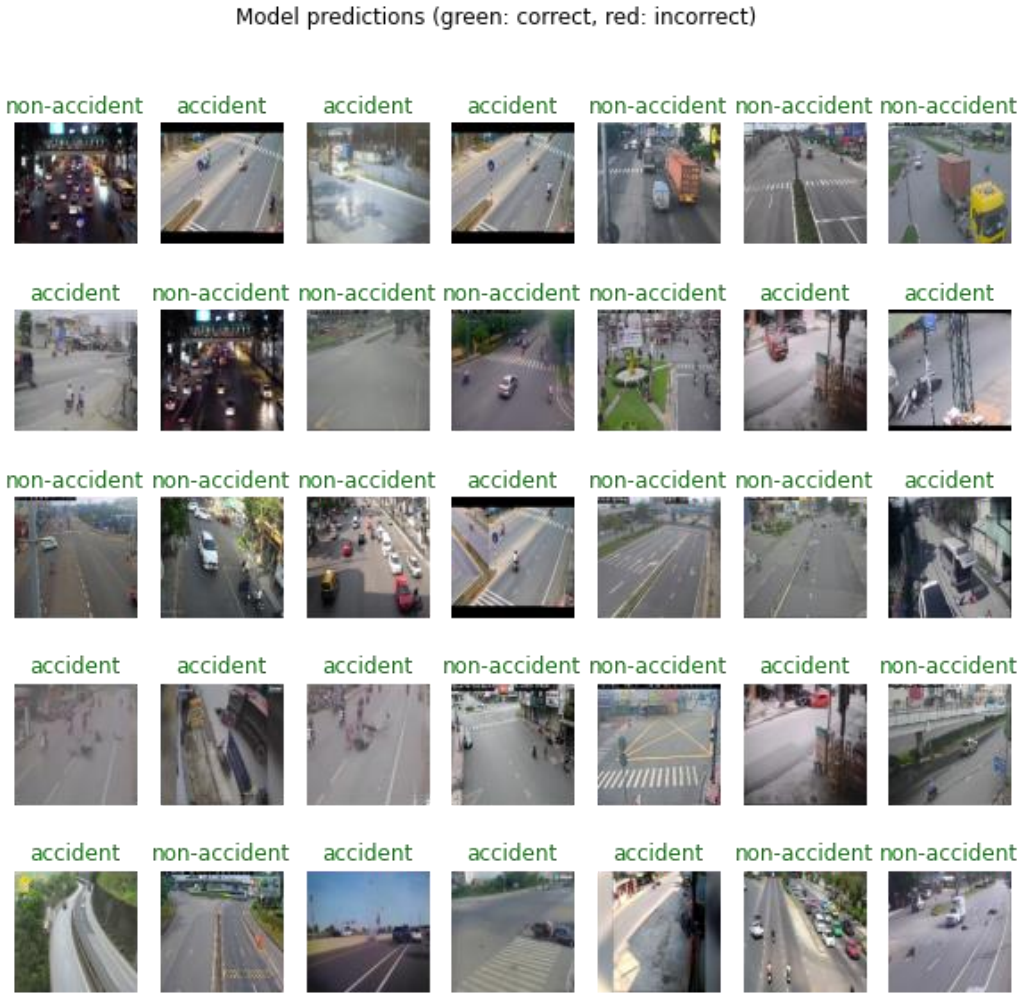
- Xuyên suốt quá trình huấn luyện, giá trị train loss và val loss đều giảm nhưng giá trị val loss rất biến động, tăng giảm bất thường theo đoạn kể từ epoch thứ 5 trở đi.
- Xuyên suốt quá trình huấn luyện, độ chênh lệch 2 cặp giá trị train loss – val loss và train acc - val acc là khá lớn.

Và cuối cùng là dự đoán của mô hình trên tập test:

```
Total correct prediction:  421
Total incorrect prediction:  11
```

Mô hình dự đoán đúng 421 tấm ảnh trên tổng số 432 tấm ảnh trong tập test. Đạt tỉ lệ chính xác **97,45%.**

Một số ảnh mà mô hình đã dự đoán:



Ảnh được model dự đoán sai:


@Newsflare
SOHA.VN
Cierre


2020-04-07 09:00:54
TikTok
@kynguyen444


www.0511.vn
2021-07-16 14:42:13


Nguyen Thi Dinh - Duong A 20-07-2021 Tue 16:11:55
TC LINE

3. Mô hình CNN có Transfer Learning

a. Tổng quan mô hình

- Việc tận dụng pre-trained model là một kỹ thuật hợp lý đối với bài toán của nhóm vì có thể tận dụng những gì mà model đã được huấn luyện sẵn theo từng task cụ thể. Ở đây nhóm sử dụng mô hình MobileNetV2 đã được pre-trained từ TensorFlow Hub (tfhub.dev).

- Pre-trained model này được huấn luyện trong bộ dữ liệu ImageNet (image-net.org) chứa 1000 class với hơn 1.2 triệu hình ảnh.
- Thời gian train: 35 phút
- Thời gian test: 15 phút (tính theo thời gian thực hiện việc đánh giá model và thử nghiệm trên tập test).

b. Thiết lập mô hình

```
import tensorflow_hub as hub
model = tf.keras.Sequential([
 hub.KerasLayer("https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4", output_shape=[1280],
                trainable=False), tf.keras.layers.Dropout(0.4), tf.keras.layers.Dense(train_generator.num_classes, activation='softmax')])
model.build([None, 224, 224, 3])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
keras_layer (KerasLayer)     (None, 1280)              2257984
_________________________________________________________________
dropout (Dropout)            (None, 1280)              0
_________________________________________________________________
dense (Dense)                (None, 2)                 2562
=================================================================
Total params: 2,260,546
Trainable params: 2,562
Non-trainable params: 2,257,984
_________________________________________________________________
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['acc'])
steps_per_epoch = np.ceil(train_generator.samples/train_generator.batch_size)
val_steps_per_epoch = np.ceil(valid_generator.samples/valid_generator.batch_size)

# Stop training if 'val_loss' stops improving for over 5 epochs <=> Fix Overfitting problem
early_stopping = keras.callbacks.EarlyStopping(monitor='val_loss',patience=5,verbose=1)

# Save the best model to .h5 file
best_model = keras.callbacks.ModelCheckpoint(filepath='best_model3.h5',monitor='val_loss',save_best_only=True,verbose=1)

hist = model.fit(train_generator, epochs=20, verbose=1, steps_per_epoch=steps_per_epoch,
          validation_data=valid_generator, validation_steps=val_steps_per_epoch, callbacks=[early_stopping,best_model]).history
```

Mô hình được huấn luyện trên tập training với 3101 ảnh, valid với 774 ảnh.

Số epoch cho training là 20 và có sử dụng cơ chế early stopping.

c. Kết quả

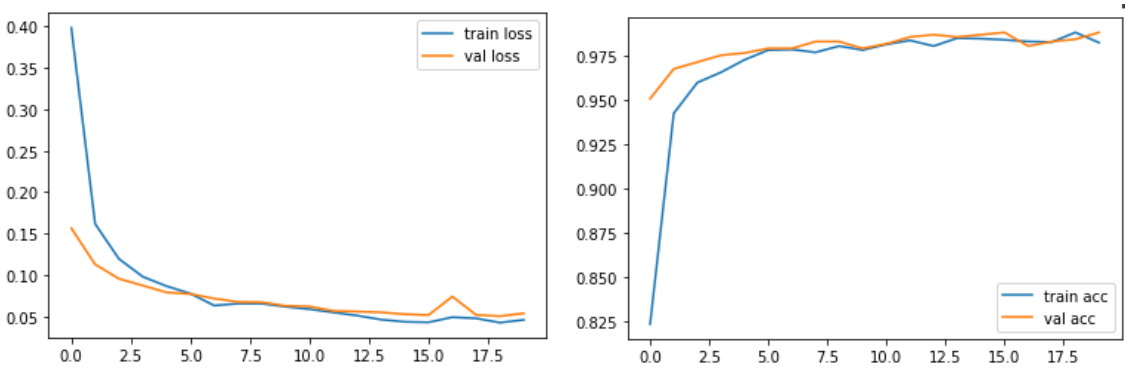
Mô hình được đánh giá với độ chính xác là **98.45%**, giá trị loss là **0.05**

Đánh giá mô hình trên tập valid

```
[ ]  final_loss, final_accuracy = model.evaluate(valid_generator, steps = val_steps_per_epoch)
     print("Final loss: {:.2f}".format(final_loss))
     print("Final accuracy: {:.2f}%".format(final_accuracy * 100))
```

```
25/25 [==============================] - 27s 1s/step - loss: 0.0507 - acc: 0.9845
Final loss: 0.05
Final accuracy: 98.45%
```

Biểu đồ về giá trị loss và accuracy trong suốt quá trình traning:



Nhận xét:

Giá trị training_loss và valid_loss mang chiều hướng giảm dần và sự chênh lệch giữa hai giá trị này là nhỏ. Xuyên suốt quá trình training, model đạt khoảng 98% độ chính xác trên tập validation.

Ảnh dự đoán sai:

4. Đánh giá

Thử nghiệm và đánh giá trên tập test:

Model	Accuracy	Loss
CNN-SVM	**0.9884**	**0.02**
CNN-SVM (có tăng cường dữ liệu)	**0.9767**	**0.06**
MobileNetV2 (Transfer learning)	**0.9845**	**0.05**

Nhóm dự đoán kết quả của mô hình CNN-SVM có tăng cường dữ liệu sẽ tốt hơn so với không tăng cường dữ liệu, và mô hình có transfer learning là đạt kết quả tốt nhất. Tuy nhiên, kết quả cuối cùng cho thấy CNN-SVM lại đạt độ chính xác cao nhất. Một số lý do có thể dẫn đến kết quả này:

- Việc tăng cường dữ liệu mà nhóm thử có thể đã làm mất thông tin quan trọng của tấm ảnh, nhìn qua các ảnh dự đoán sai có thể thấy tai nạn nằm ở trong góc, còn hình ảnh không tai nạn thì có chất lượng thấp, bị mờ bởi thời tiết.
- Dữ liệu của còn quá ít, và phần lớn lấy từ 1 nguồn, dẫn đến trường hợp overfitting.

V. Ứng dụng và hướng phát triển

Nhóm đã cho chạy mô hình CNN-SVM không tăng cường dữ liệu với input là video, dưới đây là kết quả:

- Video có tai nạn: accident.mp4 - Google Drive
- Video không có tai nạn: non_accident.mp4 - Google Drive

Kết quả cho thấy mô hình phân loại sai một vài frame trong video không có tai nạn. Và qua thử nghiệm vài video khác thì nhóm nhận thấy kết quả chưa thật sự tốt do bị overfitting. Chính vì vậy trong thực tế thì model này chưa đáp ứng được yêu cầu sử dụng.

Về hướng phát triển:

- Trong tương lai gần:
  - Tìm cách cải thiện chất lượng hình ảnh trong bộ dữ liệu.
  - Thử nghiệm thêm nhiều model khác để giải quyết bài toán.
  - Nghiên cứu thêm hướng giải quyết khác cho bài toán, chẳng hạn như đưa về bài toán nhỏ là phát hiện các trường hợp người ngã, xe ngã, xe lật,…để phát hiện tai nạn.
- Trong tương lai xa:
  - Thu thập thêm dữ liệu, lấy từ nhiều nguồn và tìm cách xử lý dữ liệu tốt hơn.
  - Nghiên cứu phát triển mô hình chạy được trên thời gian thực để triển khai mô hình trên ứng dụng thực tế.
  - Phát triển thêm mô hình phân loại mức độ nghiêm trọng của tai nạn giao thông để tăng tính ứng dụng.

Toàn bộ quá trình train và thao tác cũng như dữ liệu nhóm đã trình bay trên github: FinalProject · GitHub

VI. Các tài liệu tham khảo:

1. How To Implement Image Classification Using SVM In Convolution Neural Network - YouTube
2. Image classification | TensorFlow Core
3. How to do Image Classification on custom Dataset using TensorFlow | Codementor
4. Image classification with Convolution Neural Networks (CNN)with Keras | by Manas Narkar | Medium
5. Retinex (imagej.net)
6. Các phương pháp tránh Overfitting - Regularization, Dropout (viblo.asia)