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

Báo cáo đồ án

Môn: Cơ sở trí tuệ nhân tạo

PHÂN LOẠI ẢNH

MÃ MÔN HỌC

CSC14003

Thành phố Hồ Chí Minh – 2021

MỤC LỤC

1. Tổng quan 3
2. Phân biệt 3 loại training set, validation set, test set: 3
3. Mô tả cách phát hiện và ngăn chặn overfitting và underfitting problem: 3
3.1 Overfitting (quá khớp): 3
3.2 Underfitting: 5
4. Model: 5
5. Ưu điểm và nhược điểm của model 7
6. Trình bày kết quả nhận được và giải thích 7
7. Idea for improvement: 8
8. Hướng dẫn chạy code 9
9. Quá trình chạy dữ liệu 12
10. Reference 14

1. Tổng quan

MSSV	HỌ VÀ TÊN	EMAIL
19127392	Tô Gia Hảo	19127392@student.hcmus.edu.vn
19127525	Nguyễn Thanh Quân	19127525@student.hcmus.edu.vn
19127625	Lâm Chí Văn	19127625@student.hcmus.edu.vn

2. Phân biệt 3 loại training set, validation set, test set:

Training set

Training set: là tập dữ liệu có kích thước lớn, dùng để training cho máy học. Đây chính là tập dữ liệu máy dùng để học và rút trích được những đặc điểm quan trọng để ghi nhớ lại. Tập training set sẽ gồm 2 phần:

- Input: sẽ là những dữ liệu đầu vào
- Output: sẽ là những kết quả tương ứng với tập input

Test set

Test set: là tập dữ liệu dùng để kiểm thử **sau** quá trình huấn luyện. Một mô hình machine learning sau khi được huấn luyện, sẽ cần phải được kiểm chứng xem nó có đạt hiểu quả không. Đây là một tập hợp các giá trị input và được sử dụng để kiểm tra độ chính xác của các mô hình machine learning sau khi được training và đưa ra Ouput tương ứng với mỗi input. Khác với Training set, Testing set chỉ gồm các giá trị input mà không có các giá trị output.

Validation set

Validation test: cũng giống như tập training set, nó cũng bao gồm các cặp giá trị input và ouput tương ứng. Nhưng nó lại khác training set ở chỗ, nó được sử dụng để kiểm thử độ chính xác của mô hình máy học trong quá trình huấn luyện và được sử dụng để kiểm thử **trong** quá trình huấn luyện

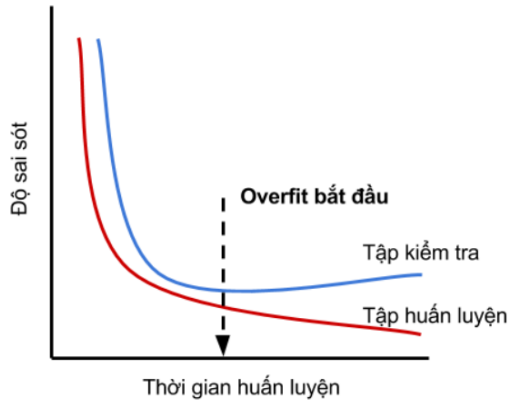
3. Mô tả cách phát hiện và ngăn chặn overfitting và underfitting problem:

3.1 Overfitting (quá khớp):

Khi mô hình của bạn được training tốt hơn nhiều so với tập validation, nó sẽ ghi nhớ các ví dụ huấn luyện riêng lẻ để mở rộng. Bằng cách đó, nó có thể dự đoán dữ liệu đào tạo của bạn rất tốt, nhưng không khái quát được vấn đề thực tế và do đó thất bại trên các ví dụ không nhìn thấy. Việc khớp dữ liệu đào tạo quá tốt được gọi là Overfitting.

Phát hiện overfitting

Để phát hiện overfitting, ta cần theo dõi **learning curve**, một biểu đồ thể hiện sự biến động của $\mathcal{L}_{D_{train}}$ và $\mathcal{L}_{D_{test}}$ trong suốt thời gian huấn luyện. Cứ sau một khoảng thời gian, ta ghi lại giá trị của $\mathcal{L}_{D_{train}}$ và $\mathcal{L}_{D_{test}}$ và vẽ biểu đồ của chúng theo thời gian, ta được learning curve.



- Nếu ta áp dụng một phương pháp tối ưu hàm số hiệu quả, $\mathcal{L}_{D_{train}}$ (đường đỏ) sẽ giảm theo thời gian.
- Ngược lại$\mathcal{L}_{D_{test}}$ (đường xanh) không phải lúc nào cũng giảm. Nếu model bị overfitting, đến một lúc nào đó, gía trị này bắt đầu tăng trở lại.
- Thời điểm mà $\mathcal{L}_{D_{test}}$ bắt đầu có xu hướng tăng được xem thời điểm bắt đầu overfitting. Vì sao? Vì sau đó, việc huấn luyện sẽ làm model dự đoán ngày càng tốt hơn trên training set, nhưng lại cho sai sót ngày càng nhiều trên test set.

Cách để ngăn việc Overfitting:

- ***Gather more data (Bổ sung nhiều dữ liệu hơn):***

Dữ liệu ít là 1 trong trong những nguyên nhân khiến model bị overfitting. Vì vậy chúng ta cần tăng thêm dữ liệu để tăng độ đa dạng, phong phú của dữ liệu (tức là giảm variance). Một số phương pháp tăng dữ liệu :

+ ***Thu thập thêm dữ liệu*** *:* chúng ta phải craw thêm dữ liệu hay tới thực tiễn để thu thập, quay video, chụp ảnh,...Tuy nhiên trong nhiều trường hợp thì việc thu thập thêm dữ liệu là infeasible nên phương pháp này không được khuyến khích.

+ ***Data Augmentation*** : Augmentation là 1 phương thức tăng thêm dữ liệu từ dữ liệu có sẵn bằng cách rotation, flip, scale, skew,... images. Phương pháp này được sử dụng rất phổ biến trong xử lý ảnh cho Deep learning.

+ ***GAN (Generative Adverserial Network)*** là mô hình học không giám sát dùng để sinh dữ liệu từ nhiễu (noise)

- ***Simple model (Mô hình đơn hơn):***

model của bạn quá sâu, phức tạp (chẳng hạn như nhiều layer, node) trong khi chỉ có chút xíu dữ liệu, dùng Một mô hình kém mạnh hơn có thể không có khả năng ghi nhớ nhiều ví dụ training, nhưng rất có thể đủ tốt để giải quyết nhiệm vụ của bạn. Nếu bạn sử dụng mạng neutral, hãy thử các lớp nhỏ hơn và ít hơn.

- ***Regularization(Quy định hóa):***

regularization là 1 kĩ thuật tránh overfitting bằng cách thêm vào hàm loss 1 đại lượng **lamda. f(weight)** => Tối ưu model (giảm hàm loss) => giảm weight => mô hình bớt phức tạp => tránh overfitting.

- ***Use Dropout:***

Dropout là kĩ thuật giúp tránh overfitting cũng gần giống như regularization bằng cách bỏ đi random p% node của layer => giúp cho mô hình bớt phức tạp (p thuộc [0.2, 0.5]) .

- ***Early stoping:***

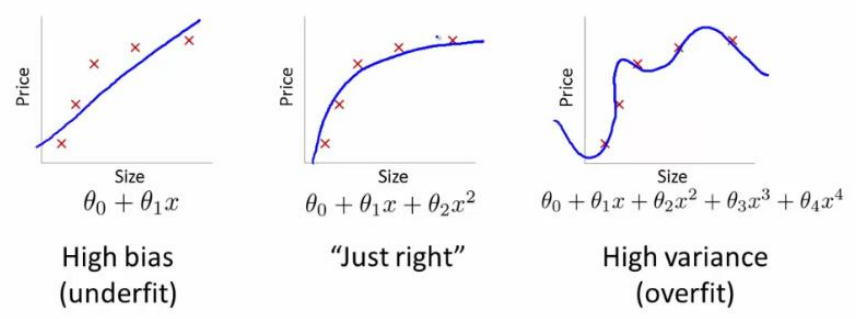
Khi training model thì không phải lúc nào (hàm mất mats) loss của tập train và tập test cũng đồng thời giảm, tới một epoch nào đó thì loss của tập train sẽ tiếp tục giảm nhưng loss của tập test không giảm mà tăng trở lại => Đó là hiện tượng overfitting. Vì vậy để ngăn chặn nó, thì ngay tại thời điểm đó người ta sẽ dừng việc training (vì để chương trình tiếp tục training thì cũng không cải thiện được gì mà lại tốn tài nguyên).

3.2 Underfitting (chưa khớp):

Underfitting (chưa khớp) là hiện tượng khi mô hình xây dựng chưa có độ chính xác cao trong tập dữ liệu huấn luyện cũng như tổng quát hóa với tổng thể dữ liệu. Khi hiện tượng Underfitting xảy ra, mô hình đó sẽ không phải là tốt với bất kì bộ dữ liệu nào trong vấn đề đang nhắc tới.

Phát hiện underfitting

Một cách để phát hiện tình huống như vậy là sử dụng phương pháp tiếp cận phương sai lệch, có thể được biểu diễn như sau:



Mô hình gặp vấn đề underfitting khi nó có bias cao.

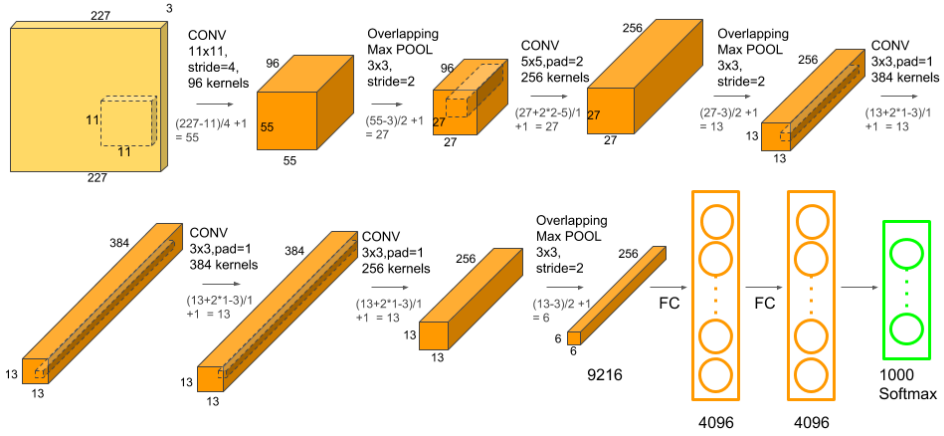
Cách để ngăn chặn vấn đề Underfitting:

Nhiều dữ liệu hơn nói chung sẽ không giúp ích được gì. Trên thực tế, nó có khả năng làm tăng lỗi training. Do đó chúng ta nên tăng thêm nhiều tính năng. Vì điều đó mở rộng không gian giả thuyết. Điều này bao gồm việc tạo ra các tính năng mới từ các tính năng hiện có. Tương tự như vậy, nhiều tham số hơn cũng có thể mở rộng không gian giả thuyết.

4. Model:

Model được nhóm sử dụng là **model AlexNet**

AlexNet là một mô hình CNN cơ bản, đơn giản và hiệu quả. Trong đó chủ yếu bao gồm các giai đoạn xếp tầng, như là lớp convolution, lớp pooling, lớp rectified linear unit (ReLU) và lớp fully connected như hình bên dưới



Kiến trúc AlexNet

Cấu trúc các lớp

Kiến trúc AlexNet bao gồm 5 lớp convolutional 5 và 3 lớp fully connected. Bên cạnh đó thực hiện ReLu và chuẩn hóa được thực hiện sau mỗi lớp convolutional.

Những convolution layer (hay còn gọi với tên khác là các filter) rút trích các thông tin hữu ích trong các bức ảnh. Trong một convolution layer bất kỳ thường bao gồm nhiều kernel có cùng kích thước. Ví dụ như convolution layer đầu tiên của AlexNet chứa 96 kernel có kích thước 11x11x3. Thông thường thì width và height của một kernel bằng nhau, và độ sâu (depth) thường bằng số lượng kênh màu.

Convolutional 1 và convolution 2 kết nối với nhau qua một Overlapping Max Pooling ở giữa. Tương tự như vậy giữa convolution 2 và convolution 3. Convolutional 3, convolution 4, convolution 5 kết nối trực tiếp với nhau, không thông qua trung gian. Convolutional 5 kết nối fully connected layter 1 thông qua một Overlapping Max pooling, tiếp theo mà một fully connected layter nữa. Và cuối cùng là một bộ phân lớp softmax với 1000 lớp nhãn

Overlapping Max Pooling

Max Pooling layer thường được sử dụng để giảm chiều rộng và chiều dài của một tensor nhưng vẫn giữ nguyên chiều sâu. sử dụng pooling có kích thước 3x3 và bước nhảy là 2 giữa các pooling. Nghĩa là giữa pooling này và pooling khác sẽ overlapping với nhau 1 pixel. Các thí nghiệm thực tế đã chứng minh rằng việc sử dụng overlapping giữa các pooling giúp giảm độ lỗi top-1 error 0.4% và top-5 error là 0.3% khi so với việc sử dụng pooling có kích thước 2x2 và bước nhảy 2 (vector output của cả hai đều có số chiều bằng nhau).

ReLU Nonlinearity

Sử dụng ReLU mô hình deep CNN sẽ huấn luyện nhanh hơn so với viêc sử dụng tanh hoặc sigmoid.

Công thức của ReLU là: f(X) = max(0,x)

Normalization

Sau khi sử dụng ReLU, hàm sẽ không có range như hàm tanh và sigmoid, vì vậy nên sử dụng chuẩn hóa.

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

Normalization

Dropout

Với gần 60 triệu tham số trong tập huấn luyện, việc overfitting xảy ra là điều dễ hiểu. Để giải quyết thì sử dụng kỹ thuật dropout. Kỹ thuật này khá đơn giản, một neural sẽ có xác suất bị loại khỏi mô hình là 0.5. Khi một neural bị loại khỏi mô hình, nó sẽ không được tham qia vào quá trình lan truyền tiến hoặc lan truyền ngược. Cho nên, mỗi giá trị input sẽ đi qua một kiến trúc mạng khác nhau. Trong quá trình test, toàn bộ network được sử dụng, không có dropout, tuy nhiên, giá trị output sẽ scaled bởi tham số 0.5 tương ứng với những neural không sử dụng trong quá trình trainning. Với việc sử dụng dropout, chúng ta sẽ tăng gấp đôi lần lặp cần thiết để đạt được độ hội tụ, nhưng khi không sử dụng dropout, mạng AlexNet rất dễ bị overfitting.

Vì sao chọn:

Vì mô hình này có độ chính xác khá cao 91.613%

5. Ưu điểm và nhược điểm của model

Advantages

- AlexNet là CNN đầu tiên sử dụng GPU để train, nên nó train rất nhanh.
- Sử dụng ReLU mang tới 2 lợi thế đó là: không giới hạn output như các hàm activation khác, điều này có nghĩa là không có quá nhiều loss function
- Đầu ra không có giá trị âm. Điều này có nghĩa là nó sẽ cải thiện tốc độ train model bởi vì không phải tất cả perceptron đều hoạt động

Disadvantages

- So với các model mới hơn thì, Model này không có hiệu quả bằng bởi vì nó phải học đặc điểm từ các tấm hình.
- Sử dụng nhiều thời gian hơn các model sau này trong việc train để đạt được kết quả accuracy cao hơn

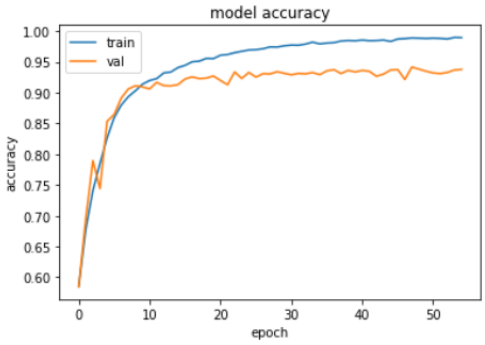
6. Trình bày kết quả nhận được và giải thích

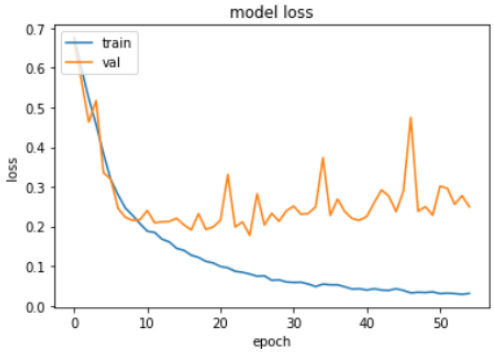
Độ chính xác: 93.375%

Model: "sequential"

Layer (type)	Output Shape	Param #
conv1 (Conv2D)	(None, 227, 227, 64)	1792
conv2 (Conv2D)	(None, 55, 55, 96)	743520
maxpool1 (MaxPooling2D)	(None, 27, 27, 96)	0
conv3 (Conv2D)	(None, 27, 27, 256)	614656
maxpool2 (MaxPooling2D)	(None, 13, 13, 256)	0
conv4 (Conv2D)	(None, 13, 13, 384)	885120
conv5 (Conv2D)	(None, 13, 13, 384)	1327488
conv6 (Conv2D)	(None, 13, 13, 256)	884992
maxpool3 (MaxPooling2D)	(None, 6, 6, 256)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 4096)	37752832
dense_1 (Dense)	(None, 4096)	16781312
dense_2 (Dense)	(None, 1000)	4097000
dense_3 (Dense)	(None, 1)	1001

===
Total params: 63,089,713
Trainable params: 63,089,713
Non-trainable params: 0





Đạt được kết quả vậy vì máy đã được train qua 20 epochs và sử dụng nhiều hình ảnh được generate dựa vào các ảnh góc, từ đó máy có nhiều góc nhìn đa dạng để tăng tỉ lệ nhận dạng

7. Idea for improvement:

Cải thiện bằng cách cải thiện hyper-parameter

- Tối ưu Learning Rate (LR) → Giảm dần learing rate sau mỗi epoch
- Batch Size → Dùng max batch size mà GPU có thể xử lí ở bài làm sử dụng batch size = 32
- Tăng độ phức tạp của model → tăng thêm nhiều lớp

Cải thiện bằng cách điều chỉnh lại dữ liệu

- Tăng kích thước của input shape → ở bài bọn em đã tăng từ 128,128,3 lên 227,227,3
- Xoay ảnh ngẫu nhiêu
- Trượt ảnh ngẫu nhiên
- Lật ngang-dọc

Cải thiện bằng cách cải thiện model

- Chỉnh lại model với tập dữ liệu con, bỏ một số mẫu dữ liệu cho một số lớp dữ liệu được lấy mẫu quá mức.
- Train lại model

8. Các file cần thiết:

final_model.h5:
https://studenthcmusedu-my.sharepoint.com/:u:/g/personal/19127525_student_hcmus_edu_vn/EUfrX0QDy7lErWMnXYSKNU8BrmgPLHXxLRQnztb4UJtJgg?e=VeU3mQ

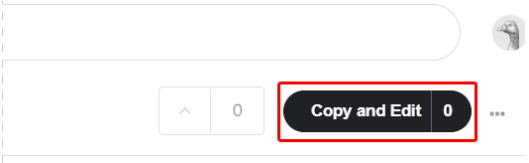
dogs-vs-cats.zip:
https://www.kaggle.com/c/dogs-vs-cats/data
hoặc
https://studenthcmusedu-my.sharepoint.com/:u:/g/personal/19127525_student_hcmus_edu_vn/EYip60GjDlpOkjaReE33d88BK5TEoVWqem0UydHD3oehhA?e=5rnDCi

9. Hướng dẫn chạy code

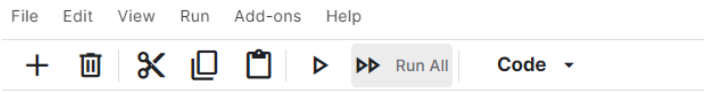
Cách 1:

Bước 1: Truy cập đường link: https://www.kaggle.com/tquntv/19127392-19127525-19127625
(Sử dụng trực tiếp project của nhóm bọn em trên kaggle)

Bước 2: Chọn **Copy and Edit** ở góc bên phải



Bước 3: Chọn **Run All**



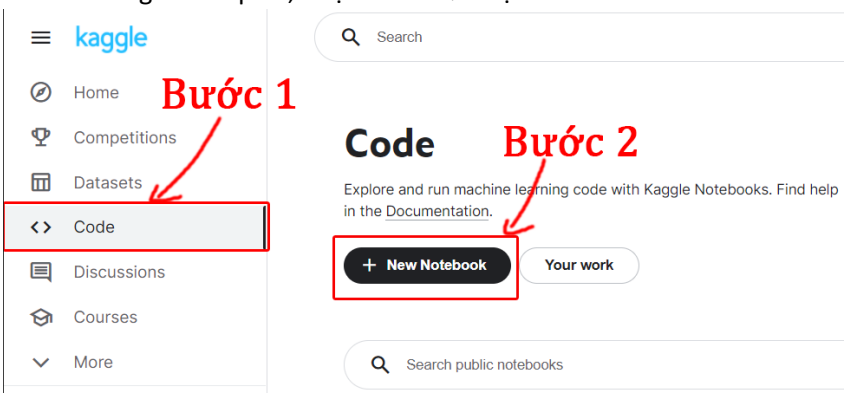
Cách 2:

Bước 1: Truy cập trang web: https://www.kaggle.com/#

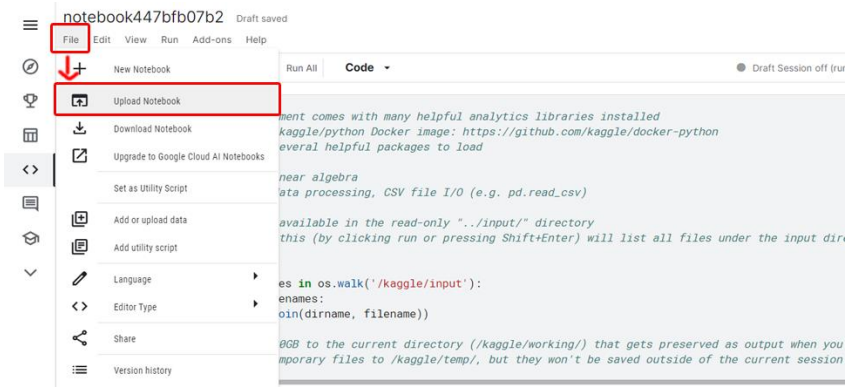
Đăng nhập:

- Nếu không có tài khoản, chọn **Register** để đăng kí
- Chọn **Sign In** để đăng nhập

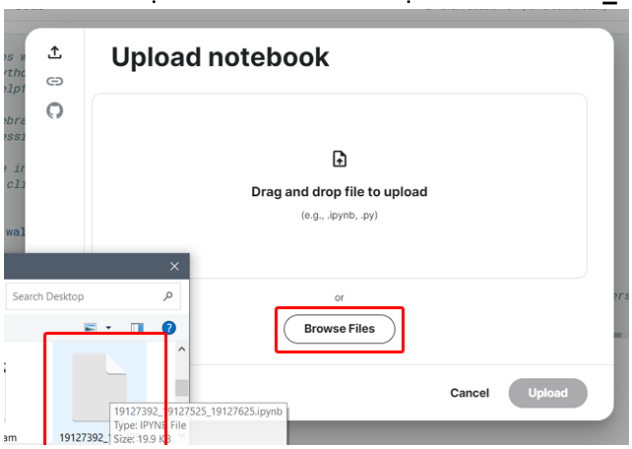
Bước 2: ở góc bên phải, chọn **Code** → chọn **New Notebook**



Bước 3: Chọn **File** → **Upload Notebook**



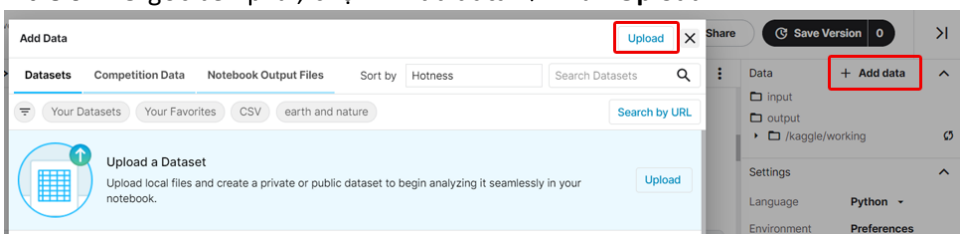
Bước 4: Chọn **Browse Files** → chọn **File 19127392_19127525_19127625.ipynb** → Nhấn **Upload**



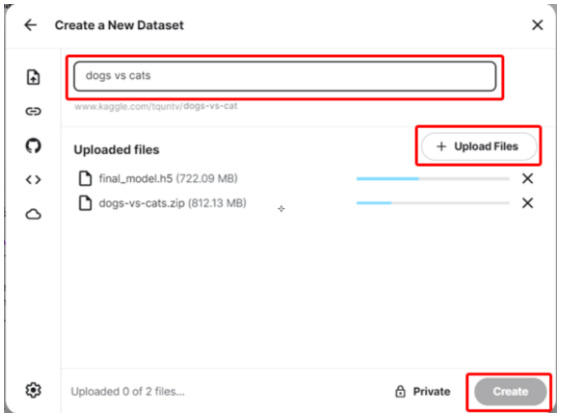
Bước 5: Upoad File

Cách 1:

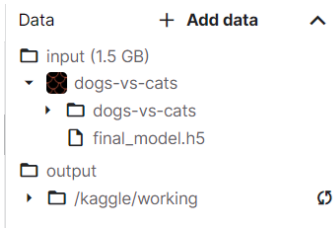
Bước 1: Ở góc bên phải, chọn **+ Add data** → nhấn **Upload**



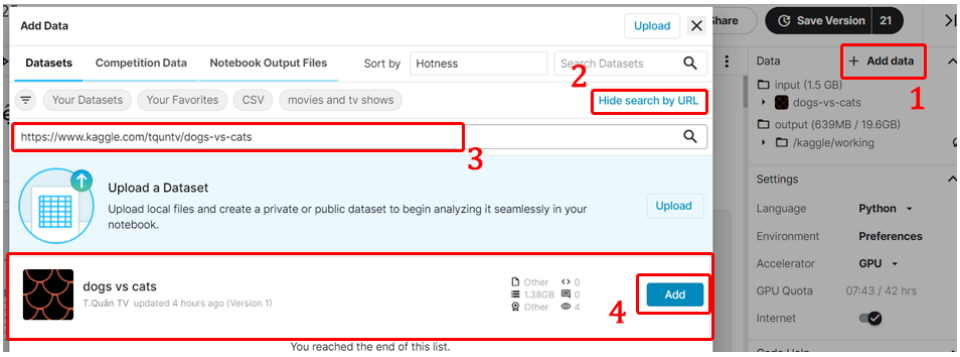
Bước 2: Nhấn **Browse File** → chọn 2 file **final_model.h5** và **dogs-vs-cats.zip**→ đặt tên data set: **dogs vs cats** → sau khi 2 file Upload xong → nhấn **Create**



Hoàn tất, trong Input sẽ hiện như trên

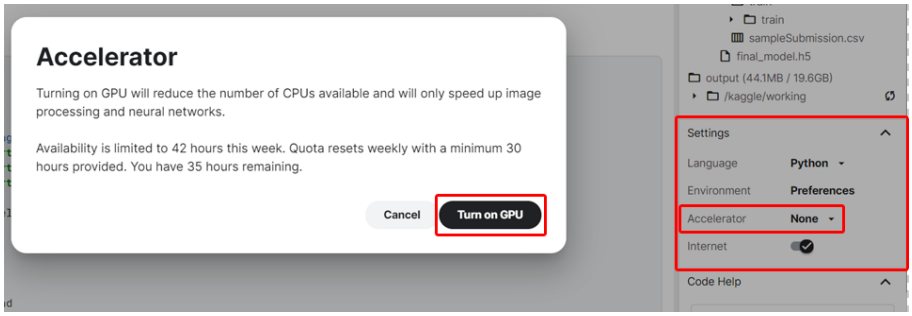


Cách 2: Chọn **+ Add data** → Nếu ở ô thứ 2 bên dưới có chữ **Hide search by URL** → khỏi nhấn, nếu có chữ **Search by URL** → nhấn → ở ô nhập URL(3) → dán vào đường dẫn https://www.kaggle.com/tquntv/dogs-vs-cats → Kaggle sẽ load dataset online của nhóm và hiện như ở bên dưới → nhấn **Add**

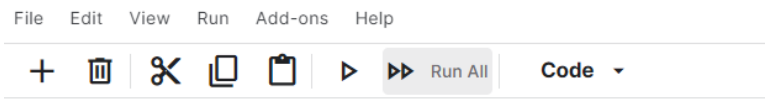


Bước 6: Mở GPU

Chọn Tab **Settings** bên phải → vào **None** (trong Accelerator) → Chọn **GPU** → Chọn **Turn on GPU**



Bước 7: Chọn **Run All**



10. Quá trình chạy dữ liệu

Sau khi nhấn **Run All,** kaggle sẽ tiến hành chạy hết tất cả các Cell

Trước tiên máy sẽ chạy cell **Tạo cây thư mục**


```
[2]:
def train_validate_test_split(data, train_percent=.6, validate_percent=.2, seed=None):
    np.random.seed(seed)
    perm = np.random.permutation(np.arange(len(data)))
    m = len(data)
    train_end = int(train_percent * m)
    validate_end = int(validate_percent * m) + train_end

    train = np.array(data)[:train_end].copy()
    validate = np.array(data)[train_end:validate_end].copy()
    test = np.array(data)[validate_end:-2].copy()

    return train, validate, test
```

Sau khi chạy xong, cell này sẽ thông báo như này, quá trình chạy có thể kéo dài từ 30 giây đến 1 phút

```
Create subdirectory.......Done!
Cat: 12500
Dog: 12500
Classify directory........Done!
```

Tiếp đến chạy cell **Data Augment**


```
[3]:
# Generating images for the Test set
print('Generating images.........', end='')
train_datagen = ImageDataGenerator(rescale=1.0/255.0,
                                   width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)

val_datagen = ImageDataGenerator(rescale=1.0/255.0)
test_datagen = ImageDataGenerator(rescale=1.0/255.0)

# Creating training set
train_it = train_datagen.flow_from_directory(data_dir +'/train/',
                                             class_mode='binary', batch_size=batchSize, target_size=(length, width))
val_it = val_datagen.flow_from_directory(data_dir +'/val/',
                                         class_mode='binary', batch_size=batchSize, target_size=(length, width))

# Creating the Test set
test_it = test_datagen.flow_from_directory(data_dir +'/test/',
                                           class_mode='binary', batch_size=batchSize, target_size=(length, width))

print('Done!',end='\n')
```

```
Generating images.........Found 15000 images belonging to 2 classes.
Found 5000 images belonging to 2 classes.
Found 4996 images belonging to 2 classes.
Done!
```

Chạy cell **Load model** để nạp model đã train dở (nếu trước có train dở)

Load model

- Nạp model đang được train dở từ trước
- File model được train dở: **./checkpoint.h5**

+ Code + Markdown

[5]:
```
# kiểm tra nếu model đã train dở từ trước thì tiến hành nạp và chạy
if os.path.exists(check_point_path) == True:
    print('Model loading....',end='')
    # nạp model đã chạy dở từ lần trước
    model = load_model(check_point_path)
    print('Done!',end='\n')
else:
    print('Model had not trained before')
```

```
Model had not trained before
```

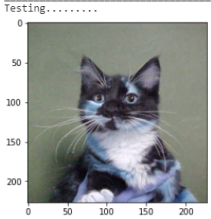
Kế đến là cell **(Main) Test model đã train từ trước**, ở đây sẽ tự động nạp model final_model.h5 đã train sẵn từ trước, chạy, hiển thị kết quả và dự đoán 1 số ảnh random trong mục ./data/test

(Main) **Test model đã train từ trước**

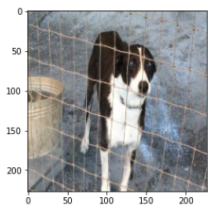
- Các ảnh test được đặt trong mục **./data/test**
- Hàm này lấy ngẫu nhiên **numtest** chó, mèo ra test

[12]:
```
print('Model loading.......',end='')
model = load_model('../input/dogs-vs-cats/final_model.h5')
print('Done!',end='\n')

evaluateModel(test_it)

model.summary()

test(model,'./data/test',3)
```

```
Model loading.......Done!
Evaluate model.......
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1877: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate`, which supports generators.
  warnings.warn('`Model.evaluate_generator` is deprecated and '
> accuracy: 93.375
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1 (Conv2D)               (None, 227, 227, 64)      1792
_________________________________________________________________
conv2 (Conv2D)               (None, 55, 55, 96)        743520
_________________________________________________________________
maxpool1 (MaxPooling2D)      (None, 27, 27, 96)        0
_________________________________________________________________
conv3 (Conv2D)               (None, 27, 27, 256)       614656
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 1001
=================================================================
Total params: 63,089,713
Trainable params: 63,089,713
Non-trainable params: 0
_________________________________________________________________
Testing.........
```




```
Predicted:Cat
```




```
Predicted: Dog
```

Sau cùng là cell **(Main) Train và chạy lại từ đầu**, để thầy có thể xem được quá trình train 55 epoch

(Main) **Train và chạy lại từ đầu**

+ Code + Markdown

```
model = defineModel()
# kiểm tra nếu model đã train dở từ trước thì tiến hành nạp và chạy
if os.path.exists(check_point_path) == True:
    print('Model loading.....Done!')
    # nạp model đã chạy dở từ lần trước
    model = load_model(check_point_path)
else:
    print('Model had not trained before')


history = trainModel(model,train_it, val_it, epoch)

saveModel('final_model.h5')

evaluateModel(test_it)

summarize_diagnostics(history)

model.summary()

test(model,'./data/test',3)
```

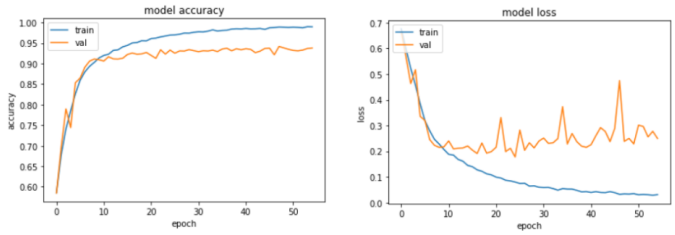
```
Defining model.....Done!
Model had not trained before
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/55
469/469 [==============================] - 218s 459ms/step - loss: 0.7243 - accuracy: 0.5404 - val_loss: 0.6305 - val_accuracy: 0.6556

Epoch 00001: val_accuracy improved from -inf to 0.65560, saving model to ./checkpoint.h5
Epoch 2/55
469/469 [==============================] - 216s 459ms/step - loss: 0.6051 - accuracy: 0.6710 - val_loss: 0.5554 - val_accuracy: 0.7098
```

Ở cell cuối cùng, không có chạy code, nó chỉ show **biểu đồ quá trình train 55 epoch**

```
Epoch 00010: val_accuracy did not improve from 0.90280
Epoch 11/55
469/469 [==============================] - 212s 452ms/step - loss: 0.1975 - accuracy: 0.9191 - val_loss: 0.2736 - val_accuracy: 0.8800

Epoch 00011: val_accuracy did not improve from 0.90280
Epoch 12/55
469/469 [==============================] - 211s 450ms/step - loss: 0.1860 - accuracy: 0.9232 - val_loss: 0.2132 - val_accuracy: 0.9136

Epoch 00012: val_accuracy improved from 0.90280 to 0.91360, saving model to ./checkpoint.h5
Epoch 13/55
149/469 [========>.....................] - ETA: 2:13 - loss: 0.1873 - accuracy: 0.9133
```

Kết quả tóm tắt lịch sử train

Đây là lịch sử sau khi train 55 epoch



11. Reference

AlexNet:
https://www.phamduytung.com/blog/2018-06-15-understanding-alexnet/

AlexNet Architecture:
https://www.kaggle.com/blurredmachine/alexnet-architecture-a-complete-guide

Avance + Disavance of AlexNet:
https://tejasmohanayyar.medium.com/a-practical-experiment-for-comparing-lenet-alexnet-vgg-and-resnet-models-with-their-advantages-d932fb7c7d17

Training, Test, Validate:
https://www.mrtech.vn/2018/11/training-set-testing-set-validation-set.html

Overfitting, Underfitting:

https://cuonglv1109.blogspot.com/2018/11/overfitting-la-gi.html
https://khanh-personal.gitbook.io/ml-book-vn/chapter1/overfitting
https://viblo.asia/p/cac-phuong-phap-tranh-overfitting-gDVK24AmlLj

Improve performance:
https://towardsdatascience.com/the-quest-of-higher-accuracy-for-cnn-models-42df5d731faf

Classification using AlexNet:
https://vitorgabo.medium.com/cats-and-dogs-classification-using-alexnet-5fc52b1cc1a6

Call back:
https://miai.vn/2020/09/05/keras-callbacks-tro-thu-dac-luc-khi-train-models/

Save và load model:
https://tek4.vn/cach-luu-va-load-lai-mot-mo-hinh-hoc-sau-keras-co-ban