BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO BÀI TẬP LỚN

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI:

DỰ ĐOÁN COVID-19 BẰNG ẢNH X-QUANG SỬ DỤNG MÔ HÌNH HỌC SÂU CNN

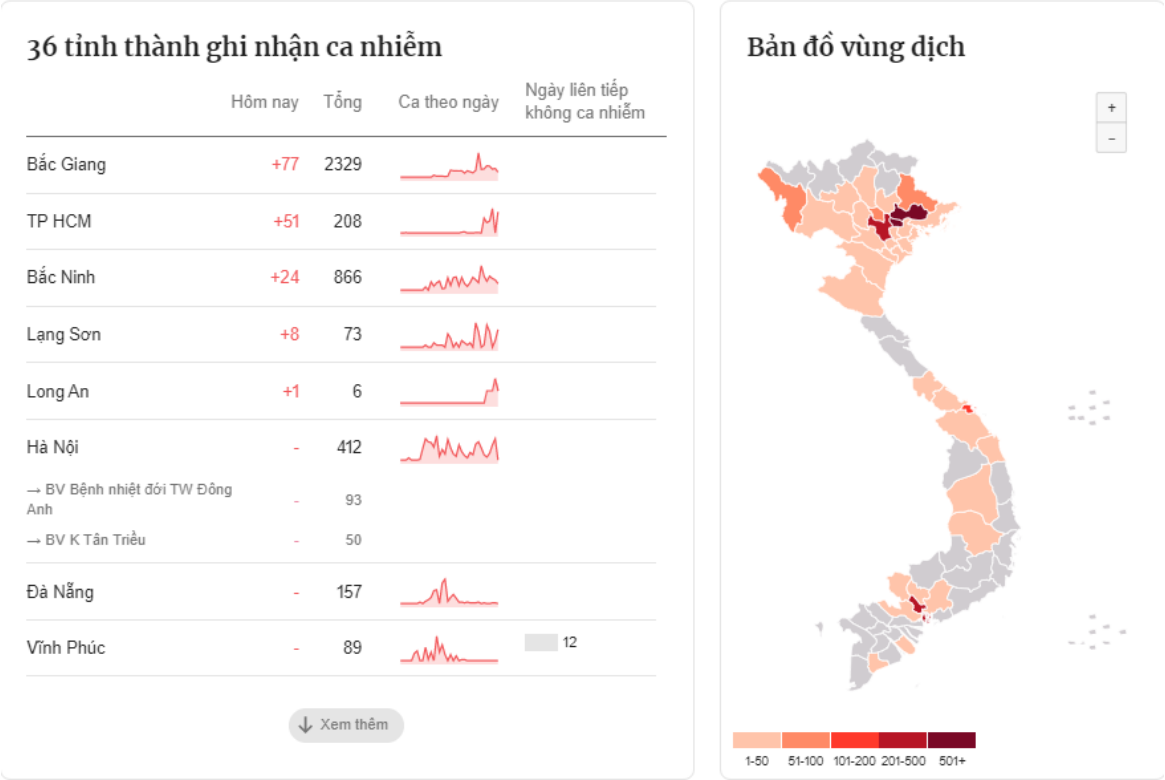
Giảng viên	**Đào Thị Thúy Quỳnh**
Nhóm môn học	**7**
Sinh viên thực hiện	**Nguyễn Văn Sang - B18DCCN508**
	Tô Văn Hải - B18DCCN200
	Đào Quang Công - B18DCCN057

Hà Nội - 2021

Mục lục:

1. Vấn đề Covid-19

Đại dịch Covid-19 có thể nói là một vấn đề thời sự nóng nhất trong hơn 1 năm trở lại đây, với số lượng người mắc bệnh ở thời điểm viết báo cáo (ngày 01/06/2021) là 171,491,204 ca mắc và 3,565,891 ca tử vong. Tại Việt Nam, đợt dịch lần thứ 4 cũng đang diễn biến phức tạp với hơn 4000 ca mắc mới. Từ đó đặt ra yêu cầu lớn trong việc xét nghiệm và chẩn đoán nhanh Covid-19.



Nguồn: VnExpress

Hai phương pháp xét nghiệm Covid-19 hiện nay là xét nghiệm bằng phương pháp Real-time PCR và thực hiện test nhanh bằng bộ Kit xét nghiệm. Tuy nhiên việc test nhanh lại không mang đến độ chính xác quá cao, còn xét nghiệm bằng PCR tuy cho kết quả tốt hơn nhưng thời gian xét nghiệm lại lâu dẫn đến khó khăn cho nhà chức trách trong việc đưa người đi cách ly, khoanh vùng và dập dịch. Hơn nữa, trên thế giới cũng như Việt Nam, hiện tượng âm tính giả đối với người mắc Covid-19 sau một thời gian điều trị là có khả năng, dẫn đến một nhu cầu cấp bách có thêm nhiều các phương pháp đơn giản, nhanh chóng và rẻ tiền hơn để chẩn đoán và sàng lọc Covid-19.

Các nghiên cứu gần đây tên thế giới đã chỉ ra rằng Covid-19 có thể được phát hiện bằng cách sử dụng ảnh X quang. Hình ảnh chụp X quang phổi của người

nhiễm bệnh có thể được sử dụng để phát hiện tổn thương ở phổi liên quan đến Covid-19.

- *Nhóm các nhà khoa học Trung Quốc cho rằng phần lớn bệnh nhân dương tính với Covid-19 có kết quả chụp X quang bất thường như độ mờ kính nền và bất thường hai bên. Họ đã phát triển một mô hình dựa trên mạng học sâu CNN có thể phát hiện Covid-19 với độ chính xác cao, cung cấp khả năng chuẩn đoán nhanh và đáng tin cậy.*
- *Viện công nghệ Massachusetts(MIT) đã nghiên cứu ứng dụng AI phát hiện Covid thông qua tiếng ho của người bệnh. Việc áp dụng công nghệ đã giúp con người có khả năng phát hiện sớm được Covid-19 để từ đó có thể đưa ra được những quyết định đúng đắn để ngăn chặn sự lây lan của đại dịch.*
- *Ở Việt Nam, với diễn biến phức tạp của dịch Covid, VinBrain đã liên kết với Bộ Y tế thực hiện đề tài "Nghiên cứu ứng dụng trí tuệ nhân tạo để sử dụng ảnh X-quang phổi trong hỗ trợ chuẩn đoán Covid-19" và được tích hợp vào DrAid. Ứng dụng này có khả năng phát hiện nhanh các dấu hiện bất thường trong vòng 5 giây kết hợp cùng xét nghiệm PCR từ độ nâng cao độ chính xác và giảm thiểu tình trạng âm tính giả. Bệnh nhân nghi ngờ mắc Covid-19 sẽ được xét nghiệm PCR và chụp ảnh X quang để AI đưa ra kết quả. Trường hợp AI có kết quả dương tính thì bệnh nhân cần được cách ly ngay, đồng thời sẽ dựa trên kế quả xét nghiệm PCR để đưa ra các quyết định điều trị chính xác. Ứng dụng này đã được tập đoàn Vin chia sẻ với mã nguồn mở để cộng đồng và doanh nghiệp trong lĩnh vực liên quan cùng chung tay nghiên cứu các phải pháp tốt hơn cho việc dự đoán Covid-19.*

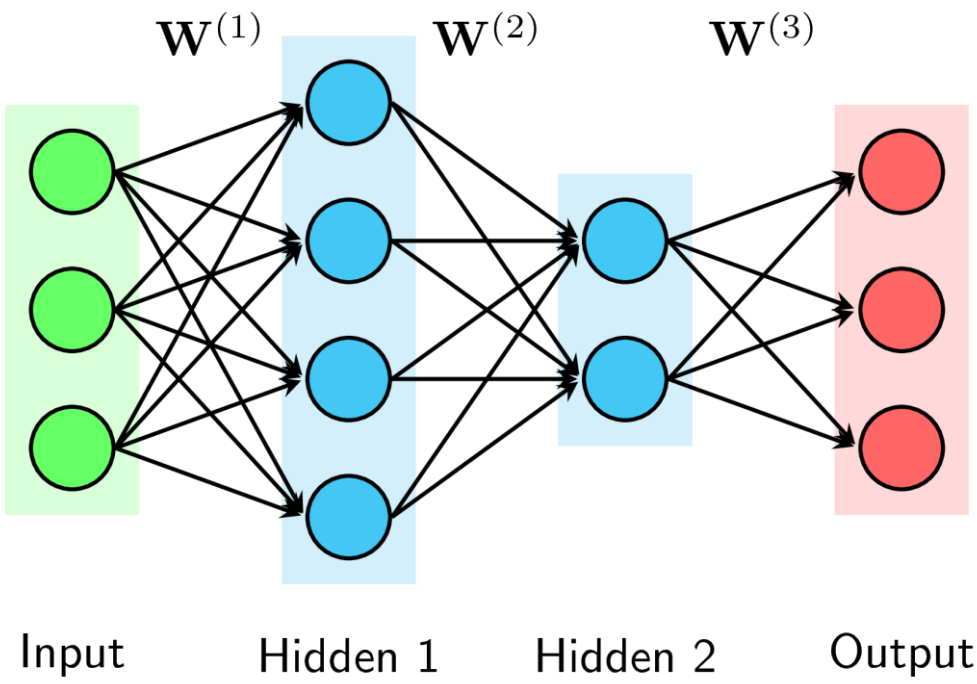
Thông qua việc tìm hiểu về các ứng dụng trí tuệ nhân tạo trong việc dự đoán Covid-19, nguồn dữ liệu được đóng góp từ các bệnh viện trên thế giới ở Kaggle:

"*https://www.kaggle.com/tawsifurrahman/covid19-radiography-database*"
và các kiến thức về học sâu chúng em học được trong thời gian qua, nhóm chúng em đã xây dựng mô hình CNN để đưa ra kết quả dự đoán Covid-19 và các bệnh có liên quan khác và triển khai mô hình đó trên ứng dụng web được trình bày ở các phần sau.

2. Các vấn đề toán học cơ bản trong mô hình học sâu Convolution Neural Network (CNN).

2.1. Mạng neural đa tầng Multi-layer Perceptron

Một mạng neural đa tầng Multi-layer Perceptron là một mạng gồm các layer xếp liên tiếp nhau, mỗi layer có các node giống như các noron trong mạng thần kinh con người.



Với lớp đầu tiên gọi Input Layer, cuối cùng đầu ra là Output Layer, ở giữa có các Hidden Layer. Các node trong các layer khác nhau được nối với nhau bởi các trọng số w_{ij} (thể hiện trọng số nối từ node thứ i của layer trước với node thứ j của layer sau). Ngoài ra ở mỗi layer có thêm hệ số điều chỉnh b gọi là bias thường không được vẽ trên hình. Tất cả các node và bias của layer trước đều được tổ hợp tuyến tính với các trọng số tương ứng để tạo ra các giá trị node của layer sau, mạng với các layer kết nối đầy đủ như vậy gọi là Fully Connected Layer.

Trong mô hình neural đa tầng, sau khi tính tổng:

$$z_j = \sum a_i * w_{ij} + b_j$$

tại mỗi node z_j của layer fully connected, các tổng này tại mỗi node sẽ được đưa qua hàm kích hoạt **ReLu** là hàm $y = f(x) = \max(0, x)$ trả về 0 nếu x âm và trả về x nếu x dương. Hàm kích hoạt này đã được chứng minh là hiệu quả hơn và đơn giản cho việc tính đạo hàm hơn so với các hàm kích hoạt khác như

Sigmoid hay tanh. Hàm kích hoạt cho đầu ra là hàm **Softmax** để chuyển các giá trị đầu ra về số dương và tính dưới xác suất mỗi ảnh bị gán vào từng nhãn.

Trong model chúng em tạo ra lớp Input sau khi được duỗi ảnh (Flatten) có 7*7*64=3136 node, lớp Ouput có 3 node ứng với 3 nhãn cho dữ liệu lần lượt là Covid (bị nhiễm Covid-19), Normal (phổi người bình thường) và Viral Pneumonia (bệnh viêm phổi thông thường), có 1 Hidden layer 64 node. Tất cả đều là các fully connected layer.

```
model.add(Flatten())

model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

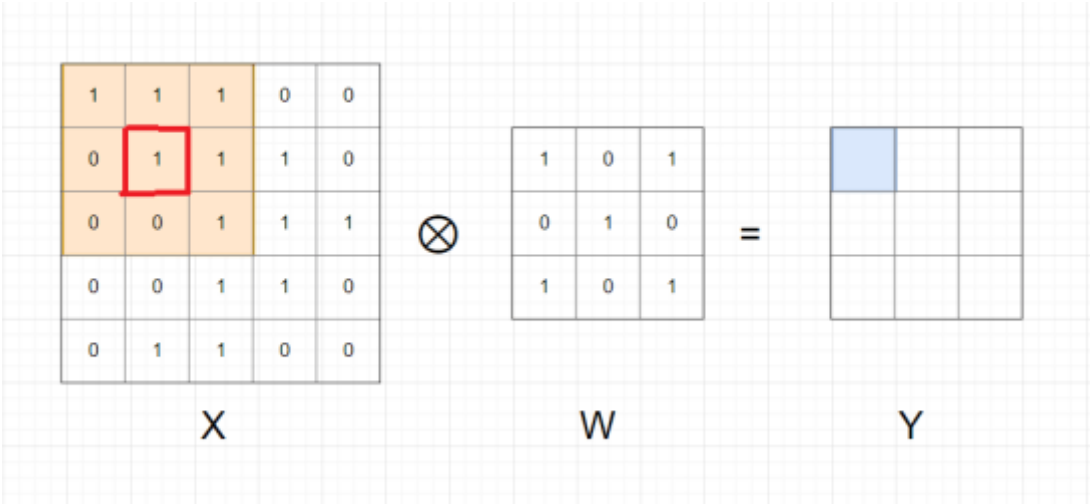
2.2. Xử lý ảnh với tích chập (Convolution)

Mỗi bức ảnh được hiển thị trên màn hình là một ma trận điểm ảnh được gọi là các pixel. Với ảnh màu RGB, mỗi pixel là một vecto trong không gian 3 chiều (r, g, b), trong đó r, g, b là các số nguyên 8 bit chạy từ 0 đến 255 thể hiện tỉ lệ pha trộn của 3 màu cơ bản là red, green và blue. Từ đó, số tổ hợp màu tạo ra lên tới 256^3 = 16 777 216 màu. Đối với ảnh xám (gray), mỗi pixel chỉ là một số nguyên 8 bit như một vecto 1 chiều biểu thị độ xám của điểm ảnh đó (với 0 là đen nhất và 255 là trắng nhất).

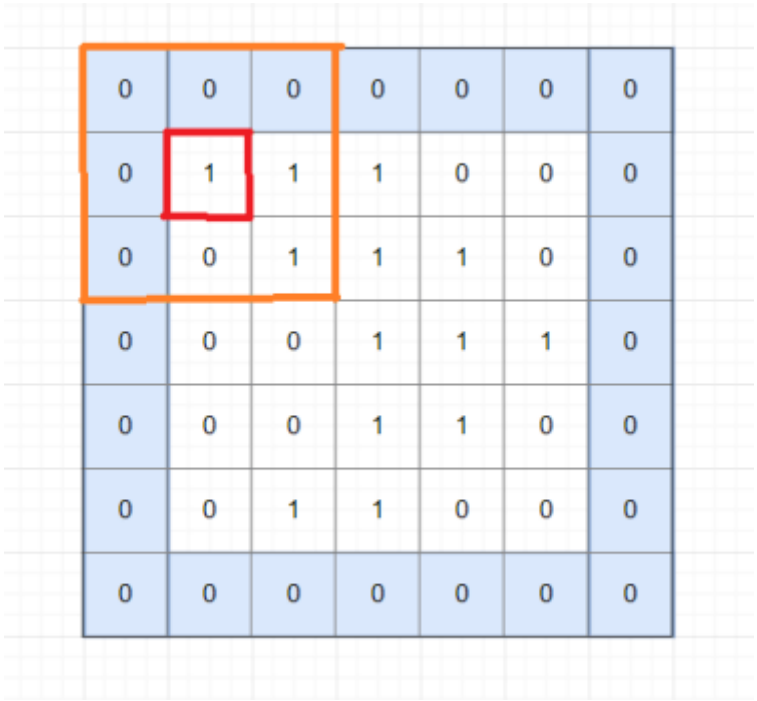
Convolution là phép tích chập một ma trận điểm ảnh với một kernel. Kernel là một ma trận vuông W kích thước k*k với k là số lẻ 1, 3, 5, 7,… Phép tích chập được định nghĩa như sau:

$$Y = X \otimes W$$

Mỗi phần tử của ma trận kết quả Y được tính bằng cách lấy ra một ma trận con trong ma trận điểm ảnh X có kích thước bằng W, rồi nhân lần lượt từng phần tử của ma trận con đó với phần tử tương ứng của kernel W, và cộng các kết quả lại thành tổng $\sum x_{ij} * w_{ij}$ với i, j chạy từ 1 đến k. Tổng này được ghi vào ma trận Y. Lần lượt dịch kernel W theo từng bước nhảy (stride), ta sẽ điền hết được kết quả là một ma trận Y.



Để ma trận Y có cùng kích cỡ với X, ta cần tạo thêm padding (đệm lót) cho ma trận X tùy vào kích thước của kernel. Ví dụ với ma trận kernel kích thước 3*3, thì padding = 1, với sự thêm vào của các dòng toàn 0 và cột toàn 0 vào 4 phía biên của ma trận X. Như vậy khi di chuyển kernel X với bước nhảy stride=1 từ trái sang phải và từ trên xuống dưới, ta sẽ thu được ma trận Y cùng cỡ với X.



Ý nghĩa của phép tính Convolution:

Phép tích chập với kernel sẽ trích xuất ra các đặc trưng (feature) của ảnh. Với mỗi kernel W khác nhau sẽ cho ra kết quả với phép tích chập có ý nghĩa trích rút các đặc trưng khác nhau. Ví dụ muốn trích rút ra các biên ảnh, ta có thể sử dụng 1 số các dạng kernel như sau:

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

Kernel trên là ma trận xác định biên trung tâm (đánh giá tính trội của điểm ảnh trung tâm với 8 điểm ảnh xung quanh).

$$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \qquad \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix}$$

2 kernel trên là kernel xác định biên dọc và biên ngang (đánh giá tính trội của các điểm ảnh bên trái so với bên phải, hoặc tính trội của điểm ảnh trên so với điểm ảnh dưới).

Trong mô hình có sử dụng bộ dữ liệu ảnh X-ray trên Kaggle là ảnh xám với kích thước 299*299. Ảnh sẽ được resize về kích thước 224*224, rồi đưa vào các layer tích chập xử lý với 64 kernel 3*3 mỗi bước, mỗi lần tích chập model được add thêm 1 layer **MaxPooling** 2*2 để giảm kích thước của ảnh bằng phép gộp (pool) 4 điểm ảnh thành 1 điểm ảnh có trị số cao nhất đại diện cho 4 điểm ảnh đó.

```
model = Sequential()
model.add(tf.keras.Input(shape=(size, size, 1)))

model.add(tf.keras.layers.Lambda(lambda x: x/255.0))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
```

Sau 5 lớp Convolution và MaxPooling, ảnh còn lại kích thước 7*7 được duỗi bằng hàm Flatten để đưa vào các Fully Connected Layer trong mạng Neural Network.

2.3. Tối ưu hàm mất mát (Loss function) với Gradient Descent

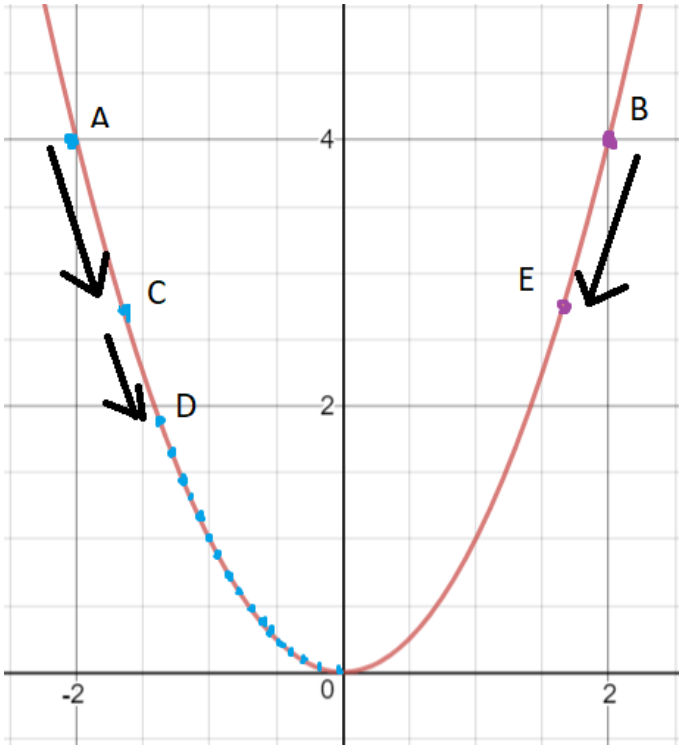
Trong Machine Learning cũng như Deep Learning, chúng ta luôn cố gắng tạo ra những mô hình có thể mô tả một cách gần đúng hay xấp xỉ nhất với mô hình thực tế. Việc đánh giá độ tốt của mô hình như thế dựa vào việc giảm thiểu hàm mất mát – là một hàm mang tính thống kê. Ví dụ xét cho 1 tập dữ liệu nhị phân:

$$L = y_{(i)} * log(y'_{(i)}) + (1 - y_{(i)}) log(1 - y'_{(i)})$$

Với $y(i)$ là giá trị nhãn thực tế cho điểm dữ liệu thứ i, $y'(i)$ là giá trị dự đoán của mô hình. Hàm L như trên được gọi là cross entropy, nó được sử dụng rộng rãi do tối ưu hơn hàm tính khoảng cách theo công thức khoảng cách Euclide thông thường.

Để tối ưu hóa (giảm thiểu) hàm mất mát trên, chúng ta thường dùng một phương pháp điển hình gọi là **Gradient Descent**. Bằng việc tính đạo hàm của hàm L trên, tìm cách thay đổi các trọng số w để đạo hàm của L với các trọng số trở về gần 0 nhất có thể (vì khi đạo hàm bằng 0 đối với các hàm lồi, thì giá trị của nó là minimum).

Tư tưởng của Gradient Descent gần giống với thuật toán leo đồi. Theo hình vẽ dưới, xuất phát từ 1 điểm nào đó, nếu đạo hàm tại điểm đang xét là âm (như điểm A, C, D - ở bên trái), chứng tỏ điểm tối ưu (đạo hàm bằng 0) đang nằm bên phải: $x^* > x_D > x_C > x_A$. Ngược lại với điểm B, E có đạo hàm dương, điểm tối ưu sẽ bên nằm bên trái.



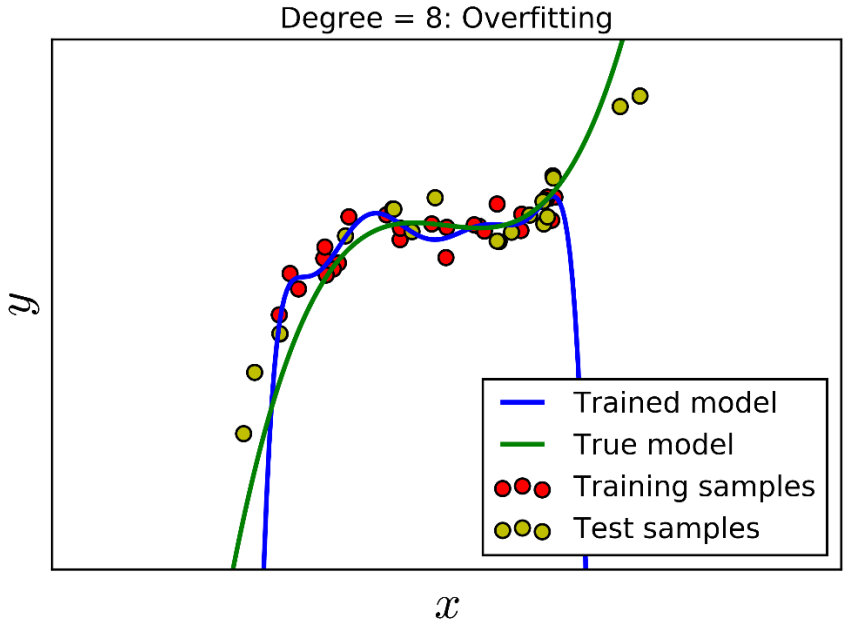
Từ đó, ta chỉ cần xác định đạo hàm tại điểm đang xét là âm hay dương, ta sẽ dịch x sang trái (giảm x) hoặc sang phải (tăng x) cho phù hợp. Khoảng cách dịch đó của x người ta gọi là *learning rate – tốc độc học*. Việc chọn learning rate phù hợp cùng rất quan trọng. Nếu learning rate quá lớn, có thể sẽ khiến đến một bước nào đó bước nhảy sẽ khiến x nhảy qua nhảy lại điểm tối ưu mà không thể chạm gần tới x*. Nếu learning rate quá nhỏ, có thể x sẽ rơi vào những cực trị địa phương hoặc việc tối ưu hóa model trở nên rất chậm.

Trong model của chúng em có sử dụng tối ưu bằng ‘Adam’ – 1 dạng biến thể được chứng minh là tối ưu hơn Gradient Descent. Chúng em đã chỉnh lại tốc độ học learning rate = 0.0002 cho phù hợp với model của mình. Ngoài ra để đánh giá độ chính xác của model với các nhãn của dữ liệu, chúng em dùng metric = [‘accuracy’].

```
model.compile(loss='categorical_crossentropy',
              optimizer=tf.keras.optimizers.Adam(learning_rate=0.0002),
              metrics=['accuracy'])
```

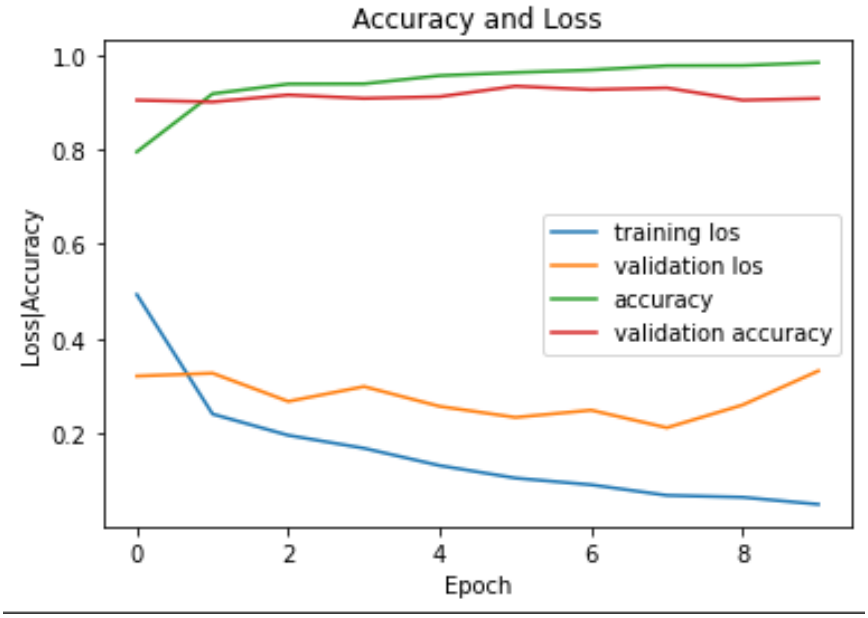
2.4. Overfitting và cách tối ưu hóa mô hình cho dữ liệu

Trong toán học, việc xác định một đa thức bậc không vượt quá n để nó đi qua tất cả n điểm (x_1, y_1), (x_2, y_2),…, (x_n, y_n) là việc đi xác định **Đa thức nội suy Lagrange**. Tuy nhiên, đa thức nội suy Lagrange đó có thể chỉ đúng ở trên tập n điểm cho trước. Khi có thêm các điểm dữ liệu khác hoặc thêm các điểm nhiễu, thì n điểm cho trước lại không chắc có thể đại diện cho toàn bộ tập dữ liệu. Khi đó, đa thức mà chúng ta tìm được sẽ rất *fit* với n điểm dữ liệu trong tập training, nhưng khi thử với tập validation hoặc tập test, thì sai số bắt đầu lớn và trở lên mất kiểm soát – hiện tượng đó người ta gọi là **Overfitting**. Chúng ta có thể nhìn hình vẽ dưới là một ví dụ minh chứng cho việc đa thức nội suy tìm được có thể rất tốt với tập train, nhưng lại sai khác lớn với tập validation và test.



Các điểm màu đỏ thuộc tập training được mô hình fit với đường màu xanh dương rất tốt. Tuy nhiên nó lại không tốt với tập test là các điểm màu vàng, chứng tỏ tập training không đại diện cho toàn bộ dữ liệu và từ đó xảy ra overfitting – quá khớp. Điều này khiến cho mô hình sau khi training không xấp xỉ với mô hình đúng thực tế.

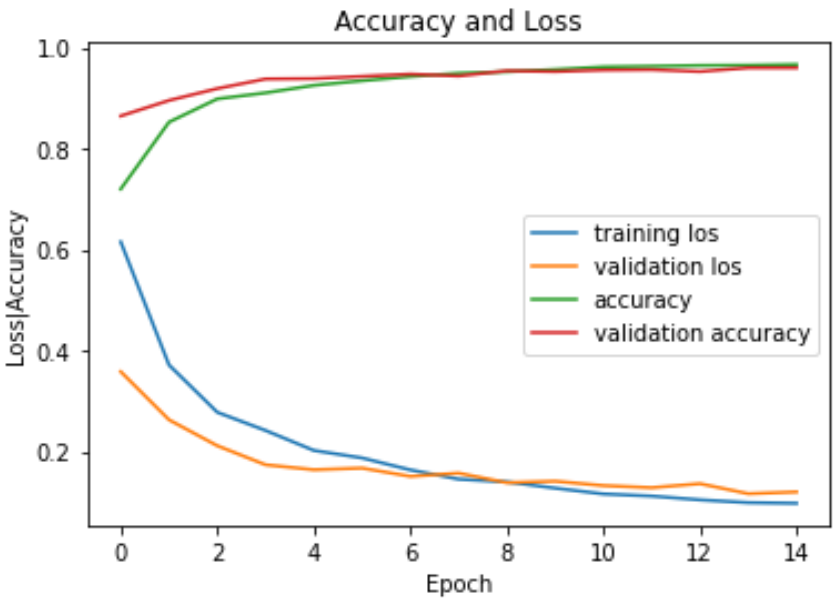
Trong quá trình train model của chúng em cũng xảy ra hiện tượng overfitting như trên. Các đường loss và accuracy của tập train và tập validation bắt đầu tách xa nhau ra sau số lượng epoch lớn:



Cách chúng em đã chọn để giảm thiểu overfitting với mô hình của chúng em là:

- Kỹ thuật Early Stopping: Với việc chọn số lần training model là số epoch thích hợp, dựa vào đồ thị của Loss và Accuracy như trên, chúng em sẽ chọn số epoch vừa đủ để 2 đường đồ thị trên và 2 đường đồ thị dưới gần sát vào nhau trước khi nó bắt đầu xảy ra overfitting ở các epoch sau.
- Dùng thêm các layer Dropout: là việc tắt ngẫu nhiên có tỉ lệ (thường rate chọn trong khoảng 0.2 đến 0.5) các units trong mạng neural network – tức là cho các unit có giá trị bằng 0 trước khi tính toán lan truyền ngược (backpropagation) trong mạng. Việc này làm giảm đi khối lượng các tham số cần tính toán, đồng thời cũng giảm hiện tượng overfitting với tập dữ liệu training.

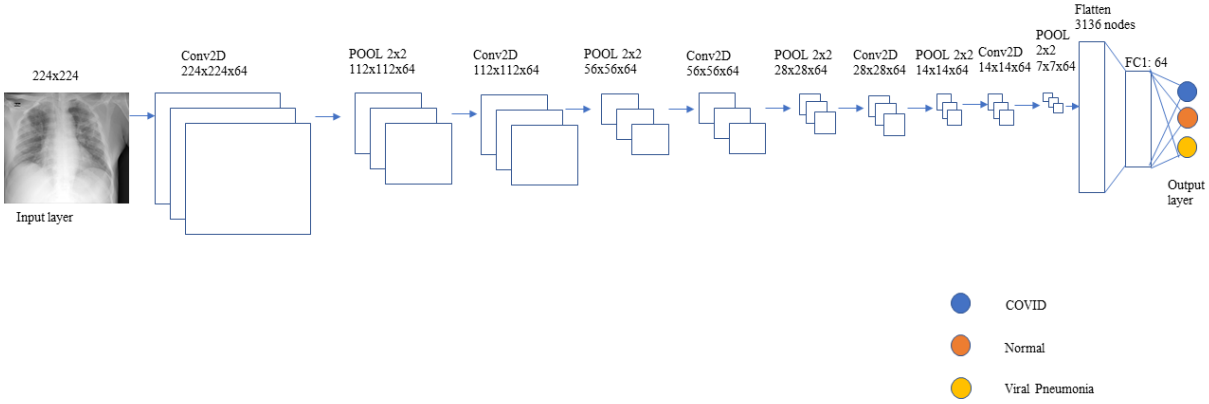
  Kết quả sau khi xử lý, chúng em chọn **epoch = 15**, và **2 layer Dropout có rate = 0.5** thu được kết quả model sau khi train như sau:



3. Đánh giá mô hình đã sử dụng và thử nghiệm chạy trên local host.

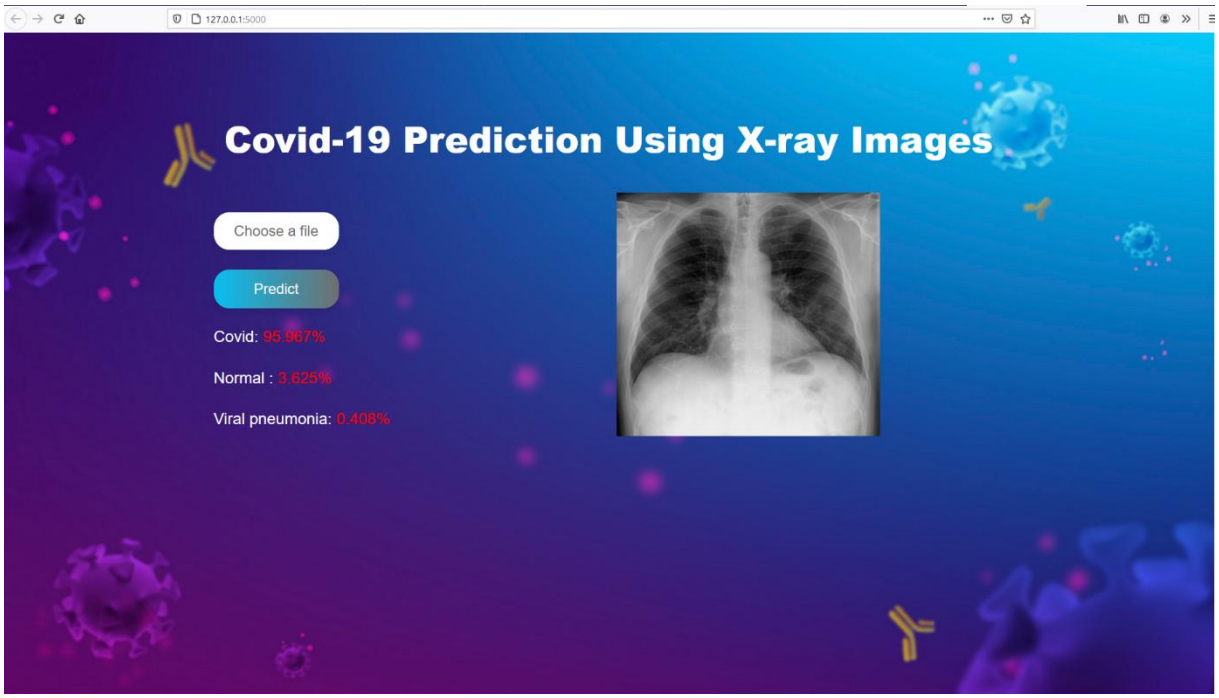
Tổng quan mô hình chúng em đã sử dụng:

Dữ liệu đầu vào bao gồm 3 nhãn Covid, Normal, Viral Pneumonia với số lượng 1000 ảnh mỗi loại. Tập dữ liệu được chia theo tỉ lệ 20% test, 16% validation và còn lại là tập train.



Mô hình trên sau khi được train có test score với độ chính xác 94,5% trên tập 600 ảnh test.

Sau khi viết giao diện front-end cho web và thiết lập back-end để lấy dữ liệu ảnh đưa vào model và in ra kết quả. Kết quả sau khi model dự đoán là xác suất ứng với các nhãn Covid, Normal, Viral Pneumonia. Dựa vào xác suất đó, chúng ta có thể sàng lọc người bệnh và đưa ra phương pháp để xử lý với từng trường hợp và mức độ.



4. Tài liệu tham khảo và lời cảm ơn.

Bài báo cáo có tham khảo kiến thức và sử dụng một số hình ảnh, dữ liệu tại các nguồn:

- https://machinelearningcoban.com/ - Blog machine learning cơ bản
- https://nttuan8.com/ - Blog deep learning cơ bản
- https://d2l.aivivn.com/ - Đắm mình vào học sâu
- https://www.kaggle.com/

Ngoài ra, nhóm chúng em cũng xin chân thành cảm ơn sự trợ giúp của bạn **Vũ Mạnh Quang** trong việc viết front-end thiết kế giao diện cho web.