

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ thông tin và Truyền thông



Bài Tập Lớn Học Máy

Áp dụng học sâu vào bài toán dự đoán độ lệch ảnh

Giảng viên hướng dẫn: ThS. Ngô Văn Linh

Nhóm sinh viên thực hiện:

Nguyễn Tử Toàn Lợi	- 20162569
Nguyễn Trọng Đức	- 20161113
Lê Thanh Tuấn	- 20164350
Lưu Ngọc Thành	- 20163712

Hà Nội, tháng 6 năm 2020

1. Giới thiệu về đề tài

Bài toán dự đoán độ lệch ảnh là một trong những bài toán tiền xử lý quan trọng trong một số tác vụ xử lý ảnh. Đặc biệt là các bài toán OCR (Optical Character Recognition), các document phải được căn chỉnh thẳng trước khi xử lý. Ngoài ra, căn chỉnh độ lệch của ảnh cũng giúp cho những photographer, designer làm việc dễ dàng hơn khi chụp ảnh, chỉnh sửa ảnh.

Cách tiếp cận phổ biến hiện nay thường sử dụng các kỹ thuật xử lý hình ảnh để tìm kiếm các cạnh ngang, dọc trong hình ảnh và sử dụng chúng để xoay theo cách sao cho các cạnh đó được căn chỉnh hoàn toàn với đường chân trời sau khi hiệu chỉnh. Cách tiếp cận này hoạt động rất tốt đối với dữ liệu là tài liệu, văn bản hoặc ảnh chụp đường phố, phong cảnh ... Tuy nhiên với những ảnh chụp thông thường, nhiều ảnh không có những đặc trưng như cạnh ngang, dọc, đường chân trời để áp dụng cách xử lý trên. Trường hợp khác là khi ảnh hoàn toàn bị xoay ngược (trường hợp ảnh bị xoay 180 độ), đường chân trời, các cạnh ngang, dọc có thể bị nhận diện sai, làm cho ảnh hoàn toàn bị xoay ngược.

Trong khi đó, các thuật toán học sâu, đặc biệt là các thuật toán về thị giác máy tính đang được quan tâm và phát triển mạnh. Các mô hình với hiệu năng và độ chính xác cao liên tục được đề xuất. Do đó, nhóm em quyết định chọn đề tài môn Học Máy là **dự đoán độ lệch ảnh**, sử dụng các pretrain model để xây dựng 1 mô hình học sâu tích chập, với dữ liệu là các hình ảnh chung, chụp nhiều đối tượng khác nhau trong các môi trường khác nhau.

2. Các khái niệm cơ bản

2.1 Bài toán classification và regression

Bài toán phân lớp (classification) là quá trình phân lớp một đối tượng dữ liệu vào một hay nhiều lớp đã cho trước nhờ một mô hình phân lớp (model), số lượng lớp là hữu hạn. Mô hình này được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có gán nhãn (hay còn gọi là tập huấn luyện). Quá trình phân lớp là quá trình gán nhãn cho đối tượng dữ liệu.

Như vậy, nhiệm vụ của bài toán phân lớp là cần tìm một mô hình phân lớp để khi có dữ liệu mới thì có thể xác định được dữ liệu đó thuộc vào phân lớp nào. Có nhiều bài toán phân lớp dữ liệu như phân lớp nhị phân (binary), phân lớp đa lớp (multi class), phân lớp đa trị.

Bài toán phân lớp nhị phân là bài toán gán nhãn dữ liệu cho đối tượng vào một trong hai lớp khác nhau. Bài toán phân lớp đa lớp là quá trình phân lớp dữ liệu với số lượng lớp lớn hơn hai. Như vậy với từng dữ liệu chúng ta phải xem xét và phân lớp chúng vào những lớp khác nhau chứ không phải là hai lớp như bài toán phân lớp nhị phân. Và thực chất bài toán phân lớp nhị phân là một bài toán đặt biệt của phân lớp đa lớp.

Một ví dụ của bài toán phân lớp là phân loại mức độ tín dụng của một người là tốt hay xấu dựa vào lịch sử giao dịch, thu nhập,.. của người đó trong quá khứ và hiện tại.

Bài toán hồi quy (regression) là bài toán dự đoán giá trị của một đối tượng dữ liệu, giá trị ở đây là một số thay vì một lớp như bài toán phân lớp. Thông thường các thuật toán regression sẽ trả về câu trả lời là một giá trị thực. Kết quả sẽ gồm rất nhiều số thập phân hay thậm chí mang giá trị âm. Thông thường các giá trị âm sẽ được chuyển thành số 0 và các giá trị thập phân sẽ được làm tròn đến giá trị gần nhất.

Một ví dụ của bài toán hồi quy là dự đoán nhiệt độ cao nhất hay thấp nhất trong những ngày tiếp theo, dựa vào các thông số khí tượng đo được.

2.2 Xử lý ảnh và thị giác máy tính

Xử lý ảnh (image processing) là các thuật toán nhằm chuyển đổi một ảnh gốc thành ảnh khác. Các bài toán xử lý ảnh bao gồm: giảm nhiễu, khôi phục ảnh, tìm cạnh, nén ảnh, nâng cao tương phản,

Thị giác máy tính (computer vision) là một lĩnh vực trong Artificial Intelligence và Computer Science (Trí tuệ nhân tạo và Khoa học máy tính) nhằm giúp máy tính có được khả năng nhìn và hiểu giống như con người. Thị giác máy tính bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh và, nói chung là dữ liệu đa chiều từ thế giới thực để cho ra các thông tin số hoặc biểu tượng.

Các bài toán về thị giác máy tính bao gồm: phát hiện vật thể, nhận dạng, phân loại, phân vùng ngữ cảnh, theo vết vật thể, ...

2.3 Deep Learning

Deep learning đã và đang là một chủ đề AI được bàn luận sôi nổi. Deep learning được bắt nguồn từ thuật toán Neural network vốn xuất phát chỉ là một ngành nhỏ của machine learning. Tuy nhiên theo thời gian, với sự phát triển của các chip xử lý tính toán song song như GPU, TPU thì Deep Learning đang ngày càng phát triển mạnh mẽ và giải quyết được nhiều bài toán với kết quả vượt trội hơn hẳn so với Machine Learning cơ bản. Các kiến trúc mạng Deep Learning phổ biến hiện nay là CNN, RNN, LSTM, Transformer,.. Hiện nay deep learning đang được sử dụng để giải quyết các bài toán về xử lý ngôn ngữ tự nhiên, nhận dạng giọng nói và thị giác máy tính, ..

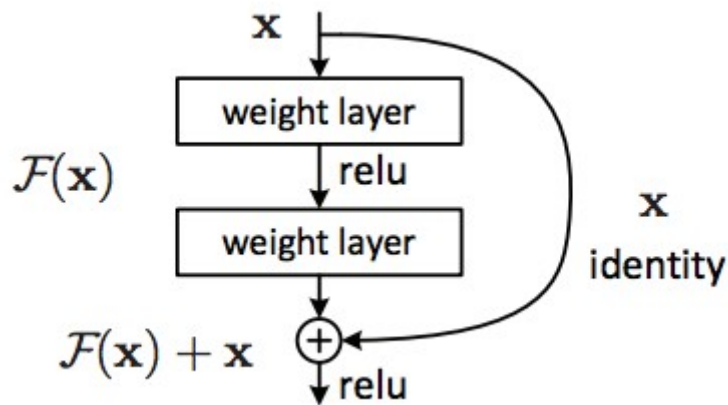
3. Kiến thức liên quan

3.1 ResNet50

Resnet 50 là 1 kiến trúc mạng CNN phục vụ cho bài toán classification. Resnet là viết tắt của: Residual network, 50 là số layer của mạng.

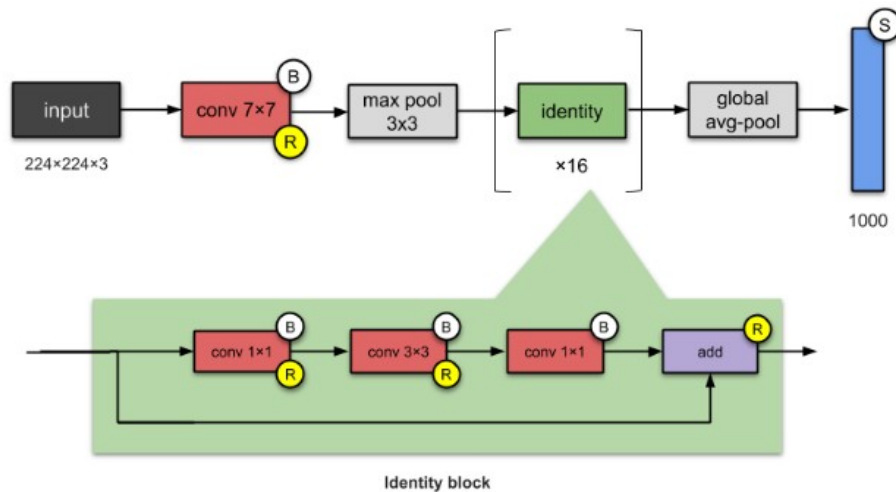
Khi train các mô hình Deep CNN (số lượng layers lớn, số lượng param lớn), ta thường gặp phải vấn đề về vanishing gradient hoặc exploding gradient. Thực tế cho thấy khi số lượng layer trong CNN model tăng, độ chính xác của mô hình cũng tăng theo, tuy nhiên khi tăng số layers quá lớn (>50 layers) thì độ chính xác lại bị giảm đi.

Giải pháp mà ResNet đưa ra là sử dụng kết nối tắt đồng nhất để xuyên qua một hay nhiều lớp. Một khối như vậy được gọi là một Residual Block (Identity block)



Residual Block

Mấu chốt của Residual block là cứ sau 2 layer, ta cộng input với output : $F(x) + x$. Với Residual block, ta hoàn toàn có thể train các mô hình CNN có kích thước và độ phức tạp lớn hơn mà không lo bị exploding/vanishing gradient. Resnet là một mạng CNN bao gồm nhiều Residual block nhỏ tạo thành. Dưới đây là cấu trúc đơn giản hóa của Resnet50



Kiến trúc Resnet50

3.2 Category Cross entropy

Cross entropy giữa hai phân phối p và q được định nghĩa là:

$$H(\mathbf{p}, \mathbf{q}) = \mathbf{E}_{\mathbf{p}}[-\log \mathbf{q}]$$

với p và q rời rạc thì ta có :

$$H(\mathbf{p}, \mathbf{q}) = - \sum_{i=1}^C p_i \log q_i \quad (1)$$

Cross entropy loss đo lường hiệu suất của mô hình phân loại có đầu ra là giá trị xác suất trong khoảng từ 0 đến 1. Cross entropy loss tăng khi xác suất dự đoán khác biệt nhiều so với nhãn thực tế. Xác suất dự đoán vào nhãn đúng (true label) càng thấp thì loss sẽ càng cao. Một mô hình hoàn hảo sẽ có một mất log là 0.

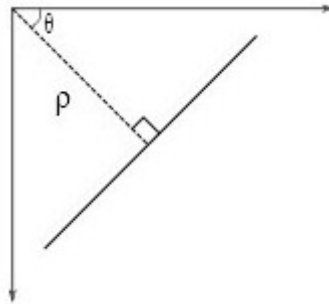
Hàm loss cross entropy có nhiều ưu điểm hơn so với các hàm loss thông thường như MSE và MAE bởi vì cross entropy có xu hướng làm giá trị loss lớn hơn các hàm loss trên khi p và q ở xa. Như vậy có thể thấy hàm cross entropy sẽ hội tụ nhanh hơn các hàm loss thông thường.

3.3 HoughLineP

HoughLineP, là một phương pháp để trích xuất đường thẳng có trong ảnh. Trong OpenCV có HoughCircles để phát hiện các đường tròn, đường elip. Còn HoughLine dùng để phát hiện các đường thẳng có trong ảnh và HoughLineP cũng là một phương pháp trích xuất đường thẳng nhưng đơn giản hơn nhiều so với HoughLine.

HoughLine là một thuật toán tìm đường thẳng có trong ảnh mặc dù nó có bị đứt đoạn. Trong hình học thì đường thẳng có phương trình: $y = m \cdot x + c$, hoặc

là $\rho = x.\cos\theta + y.\sin\theta$, với ρ là khoảng cách từ đường thẳng tới gốc tọa độ, còn θ là góc giữa trục hoành và đoạn thẳng ngắn nhất nối tới gốc tọa độ.



Hình vẽ hệ trục tọa độ trong OpenCV

Có 2 cách để xác định 1 đường thẳng, người ta chọn cách thứ 2 vì chỉ cần 2 tham số (ρ , θ) tiện cho tính toán hơn. Vì θ được tính bằng radian nên có giá trị trong khoảng $[0;\pi]$ (hay là $[0;3.14]$). Khi θ bằng 0 hoặc bằng 3.14 thì đường thẳng dựng đứng, còn θ bằng 1.57 ($\pi/2$) là nằm ngang.

Cách hoạt động của thuật toán:

Ban đầu ảnh đầu vào sẽ được đưa về dạng ảnh xám, sau đấy sử dụng các thuật toán lọc biên để xác định các đường biên có trong ảnh.

Mỗi đường thẳng được xác định bởi 2 giá trị (p, θ). Trong đấy p là khoảng cách giữa đường thẳng tới gốc tọa độ, θ là góc tạo bởi đường thẳng p (đường thẳng ngắn nhất) và trục hoành. (θ thuộc từ 0 đến 180 độ)

Vì θ thuộc từ 0 đến 180 độ, nếu ta muốn xét các đường thẳng chỉ lệch nhau 1 độ thì sẽ có 180 đường thẳng có thể thử, và mỗi đường thẳng không chỉ được đặc tả bởi θ mà còn được đặc tả bởi p . Vì p là khoảng cách từ điểm O tới đường thẳng nên p có giá trị max là bằng đường chéo của ảnh (diagonal length). Vì p có thể âm nên khoảng cách p nằm trong khoảng $[- \text{diagonal length}, \text{diagonal length}]$. Vậy tổng số p có thể thử là $2 * \text{int}(\text{diagonal_length}/p) + 1$ cách.

Từ các thông tin trên ta xác định ma trận thống kê X (khởi tạo giá trị đầu là 0) có kích thước là : số trường hợp θ * số trường hợp p . Tiếp theo ta cần phải duyệt trên mỗi pixel cạnh xem có thể có bao nhiêu phương trình đường thẳng (xác định cặp số (θ, p)) đi qua được nó.

Ý tưởng cho việc xác định được đường thẳng là cứ mỗi pixel cạnh (x, y), ta tính được giá trị p tương ứng lần lượt cho các giá trị θ (180 giá trị θ). Ví dụ : Giả sử có 4000 pixel được cho là cạnh bởi giải thuật lọc biên, thì ta sẽ có 4000 cặp giá trị (x, y) => được biểu diễn bằng ma trận 2 chiều $[4000, 2]$. Sau đấy với mỗi cặp (x, y) ta muốn thử θ từ 0 độ đến 180 độ vào phương trình đường thẳng $p = x\cos\theta + y\sin\theta$ để tính ra 180 giá trị p . Từ đấy ta thu được một ma trận C chứa các giá trị p có kích thước $[4000, 180]$.

Bây giờ ta chỉ cần duyệt mỗi phần tử trên ma trận C để "vote" vào ma trận thống kê X bằng cách xác định cặp (p, θ) phù hợp để +1 đơn vị .

Sau khi duyệt xong để vote vào ma trận X thì ta lọc theo ngưỡng xác định trước để giữ lại các đường thẳng có trong ảnh. Lọc theo ngưỡng trên ma trận X ta sẽ thu được các index của θ và index của p . Từ các giá trị index \Rightarrow tìm được các cặp (p, θ) tương ứng để đưa ra được các line tương ứng.

4. Dữ liệu

Nhóm sử dụng 2 bộ dữ liệu khác nhau (google street và COCO) để train các model khác nhau. Mục đích của việc dùng nhiều bộ dữ liệu là để cho thấy sự khác biệt với cách tiếp cận thuần xử lý ảnh (HoughLine). Bộ dữ liệu google street chụp các ảnh đường phố, nhà cửa sẽ dễ dàng hơn nhiều cho cách tiếp cận xử lý ảnh vì có các đặc trưng như đường chân trời, cây cối. Trong khi đó, bộ dữ liệu COCO chụp nhiều đối tượng hơn và không có các đặc trưng trên, sẽ gây khó khăn hơn.

4.1 Google Street

Tập dữ liệu Google Street View hơn 62 nghìn ảnh Google Street View chất lượng cao. Các hình ảnh được chụp được lấy từ trung tâm thành phố và các khu vực lân cận bang Pittsburgh, PA; Orlando, FL và một phần Manhattan. Các đối tượng trong image google street view thường là hình ảnh tòa nhà, cây cối, sự vật, ...

Bộ dữ liệu sử dụng được dùng trong bài toán được lấy từ Google Street View với hơn 7916 ảnh trong đây nhóm sử dụng 6916 ảnh để train và 1000 ảnh được dùng làm tập test

Có thể tải bộ dữ liệu Google street tại địa chỉ :

https://www.crcv.ucf.edu/data/GMCP_Geolocalization/

4.2 COCO

COCO là một bộ dữ liệu ảnh đã được gán nhãn và phân đoạn cho các đối tượng có trong ảnh. Hình ảnh trong COCO được lấy từ các cảnh hàng ngày. Datasets COCO gồm 91 loại đối tượng, là người, xe đạp, xe hơi, xe máy, thuyền, máy bay, đồ vật, cây cối, ...

Bộ dữ liệu sử dụng được dùng trong bài toán này là dữ liệu COCO năm 2017 với hơn 40 nghìn ảnh, trong đó nhóm đã sử dụng hơn 36000 ảnh để train và 4000 ảnh để test.

Có thể tải bộ dữ liệu COCO tại địa chỉ : <http://cocodataset.org/#home>

5. Kiến trúc chương trình

5.1 Hướng tiếp cận

Bài toán có thể tiếp cận theo cả hướng classification và regression. Với hướng tiếp cận classification, bài toán sẽ trở thành phân loại ảnh đã xoay vào 360 lớp tương ứng với 360 độ. Hướng tiếp cận regression sẽ dự đoán giá trị xoay của bức ảnh từ 0-360 độ, hoặc 0-1 (nếu được chuẩn hóa).

Vấn đề của cách tiếp cận regression :

Khi tiếp cận bài toán theo hướng regression, mô hình sẽ dự đoán giá trị xoay của bức ảnh. Tuy nhiên, các loss function cơ bản thường sử dụng trong bài toán regression như Mean Squared Error, Mean Absolute Error sẽ không thể sử dụng trong bài toán này vì giá trị cần dự đoán là độ xoay của ảnh. Giá trị 360 độ và 0 độ là tương đương nhau, nhưng sử dụng các loss function đã nêu sẽ tạo nên những giá trị lỗi rất lớn. Khi sử dụng các loss function đã nêu, mô hình sẽ không thể mô hình hóa được tính chất của dữ liệu.

Các cách tiếp cận dựa trên classification

Do vấn đề trên, nhóm quyết định sử dụng cách tiếp cận vấn đề bằng cách giải quyết bài toán classification, phân loại ảnh input vào các class biểu thị độ lệch của ảnh.

Khi tiếp cận theo hướng classification, các class sẽ được hiểu như không liên quan đến nhau. Xác suất dự đoán sẽ có dạng phân phối chuẩn xung quanh điểm dự đoán. Từ đó tránh được việc phải giải quyết bài toán trực tọa độ hình tròn.

Nhóm đưa ra 2 cách tiếp cận. Cách tiếp cận thứ nhất là phân loại trực tiếp ảnh input vào 360 class tương đương với 360 độ. Cách tiếp cận thứ hai là chia thành 2 quá trình, quá trình thứ nhất là dự đoán ảnh input vào 4 góc phần tư [(0-90), (90-180), (180-270), (270-360)], sau đó xoay ảnh về khoảng (0-90), quá trình thứ 2 là phân loại ảnh vào 90 class với các góc lệch tương ứng.

5.2 Các bước thực hiện

5.2.1 Tiền xử lý dữ liệu

Data augment

Data augmentation là 1 kỹ thuật trong deep learning làm tăng cường dữ liệu cho mô hình.

Các kỹ thuật data augmentation cơ bản cho deep learning:

- Original (Ảnh gốc): Không biến đổi gì cả
- Flip (Lật): lật theo chiều dọc, ngang miễn sao ý nghĩa của ảnh (label) được giữ nguyên hoặc suy ra được.
- Random crop (Cắt ngẫu nhiên): cắt ngẫu nhiên một phần của bức ảnh.

- Color shift (Chuyển đổi màu): Chuyển đổi màu của bức ảnh bằng cách thêm giá trị vào 3 kênh màu RGB. Việc này liên quan tới ảnh chụp đôi khi bị nhiễu --> màu bị ảnh hưởng.
- Noise addition (Thêm nhiễu): Thêm nhiễu vào bức ảnh. Một số loại nhiễu như: nhiễu ngẫu nhiên, nhiễu có mẫu, nhiễu cộng, nhiễu nhân, nhiễu do nén ảnh, nhiễu mờ do chụp không lấy nét, nhiễu mờ do chuyển động ...
- Information loss (Mất thông tin): Một phần của bức hình bị mất.
- Contrast change (Đổi độ tương phản): thay độ tương phản của bức hình, độ bão hòa
- Geometry based: xoay, lật, scale, padding, bóp hình, biến dạng hình ...
- Whether: thêm tác dụng của thời tiết như mưa, tuyết, sương mờ, ...
- Transitional: dịch chuyển ảnh sang trái phải.
- Dùng gan để tạo thêm ảnh (cẩn thận khi sử dụng, có thể làm xấu đi tình hình).

Trong bài toán cụ thể, nhóm đã thực hiện data augment với Flip, Random Crop, Transitional cho bộ dữ liệu.

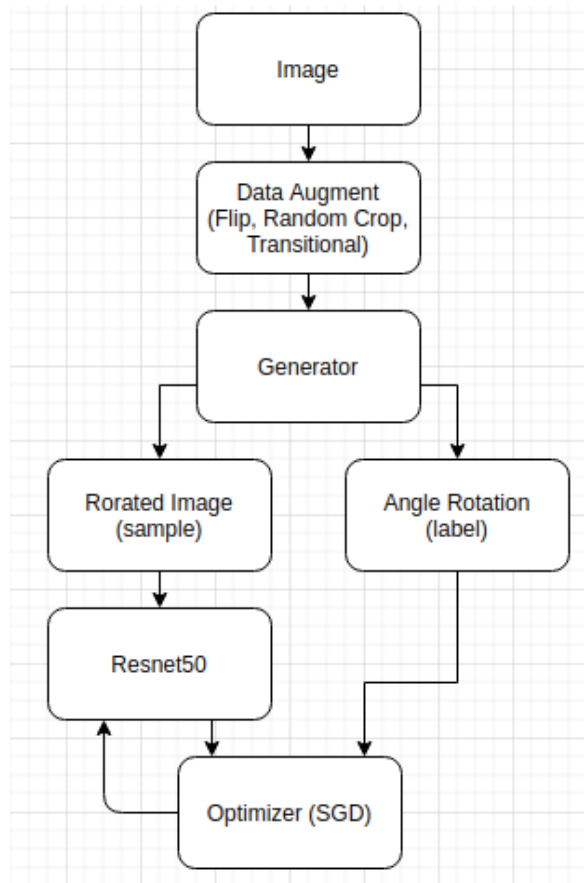
Generator

Từ bộ dữ liệu, định nghĩa 1 class Iterator để có thể chuyển dữ liệu từ ảnh thông thường sang dữ liệu vector.

Với bài toán trên, nhóm sử dụng cách tiếp cận one hot, tức là 1 ảnh có nhãn xoay ban đầu là a độ, thì sẽ được chuyển thành 1 vector 360 chiều với giá trị 1 ở vị trí a, còn lại mang giá trị 0.

Nhóm sử dụng thư viện open-cv để đọc ảnh. Và thực hiện xoay ảnh trước trước khi resize ảnh, việc này mục đích tránh tình huống nội suy khi xoay ảnh với độ phân giải thấp.

5.2.2 Cách tiếp cận 1



Xây dựng mô hình classification bằng thư viện Keras

Ảnh input được biến đổi sau khi đi qua phần Data Augment và được xoay 1 góc ngẫu nhiên. Góc ngẫu nhiên này được biến đổi thành dạng one-hot code và lưu lại làm label cho quá trình training.

Input được reshape thành [224, 224, 3]. Model ResNet50 được dùng như 1 feature extraction. Trọng số của ResNet50 là trọng số đã pretrain với bộ dữ liệu imageNet. Layer fully connected 360 unit tương ứng với 360 class. Softmax activation được sử dụng để tính xác suất rơi vào từng class của ảnh input.

Phương pháp tối ưu là SGD (Stochastic gradient descent) và loss function là category cross entropy.

5.2.3 Cách tiếp cận 2

Với cách tiếp cận thứ nhất, bài toán có 360 class, model phải dự đoán 1 vector 360 chiều.

Việc làm này có thể khiến model chưa đạt được hiệu quả nhất định. Do đó, nhóm tiến hành thử nghiệm phương án chia thành 2 quá trình.

Quá trình 1: tạo model thứ nhất (sau sẽ gọi là model_1) dự đoán bức ảnh đang xoay ở góc phần tư thứ mấy.

Quá trình 2: tạo model (sau sẽ gọi là model_2) dự đoán bức ảnh xoay ở 90 độ.

Như vậy, từ 1 bức ảnh được xoay với độ từ 0-360, model_1 sẽ dự đoán bức ảnh thuộc góc phần tư thứ mấy.

- Nếu là góc phần tư thứ nhất - tức góc xoay của bức ảnh trong khoảng: 0-90. Ta giữ nguyên sau đó dùng model_2 dự đoán ra output.

- Nếu là góc phần tư thứ hai - tức góc xoay của bức ảnh trong khoảng: 90-180. Ta thực hiện xoay bức ảnh 1 góc 90 độ theo chiều ngược chiều kim đồng hồ, sau đó dùng model_2 dự đoán kết quả, output sẽ bằng kết quả dự đoán + 90.

- Nếu là góc phần tư thứ ba - tức góc xoay của bức ảnh trong khoảng: 180-270. Ta thực hiện xoay bức ảnh 1 góc 180 độ theo chiều ngược chiều kim đồng hồ, sau đó dùng model_2 dự đoán kết quả, output sẽ bằng kết quả dự đoán + 180.

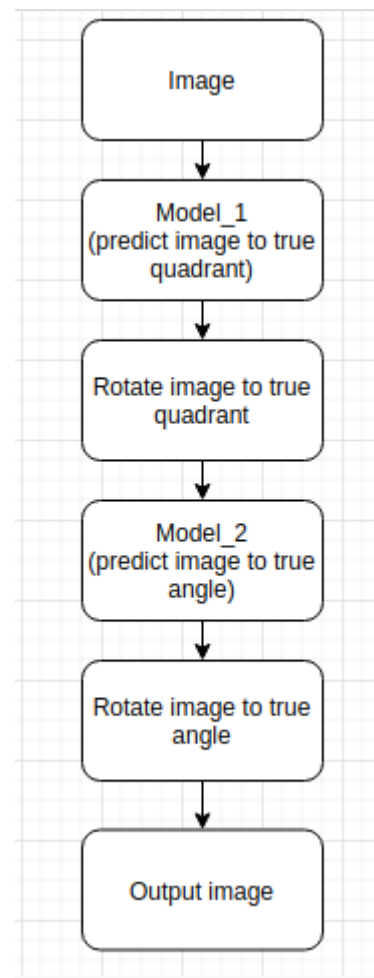
- Nếu là góc phần tư thứ tư - tức góc xoay của bức ảnh trong khoảng: 270-360. Ta thực hiện xoay bức ảnh 1 góc 90 độ theo chiều ngược chiều kim đồng hồ, sau đó dùng model_2 dự đoán kết quả, output sẽ bằng kết quả dự đoán + 270.

Model_1:

- Optimizer: SGD
- Loss function: category cross entropy.

Model_2:

- Optimizer: SGD
- Loss function: category cross entropy.



5.2.4 Tạo bộ dữ liệu test

Hình ảnh trong bộ dữ liệu COCO không phải lúc nào cũng được căn chỉnh chính xác. Một số hình ảnh được chụp với góc nghiêng, trở thành nhiễu trong quá trình test. Một số ảnh được chụp từ trên xuống, những ảnh này không thể được định hướng chính xác nên cũng gây sai lệch kết quả thử nghiệm. Những ảnh này nên được loại bỏ để có kết quả test chính xác nhất.

Để đánh giá chính xác hiệu suất của phương pháp và để so sánh với baseline, nhóm đã thực hiện chọn thủ công một tập hợp gồm 1000 ảnh chuẩn từ bộ dữ liệu để làm dữ liệu test. Đối với những hình ảnh này, nhóm đảm bảo được định hướng chính xác. Loại bỏ những trường hợp ảnh nghiêng hoặc chụp từ trên xuống.

Tập test được chia làm 2 phần easy và hard. Những ảnh easy là những ảnh chứa các đường dọc, ngang, ví dụ như ảnh các tòa nhà, ảnh chứa đường chân

trời, tường hoặc cửa. Những ảnh hard là những ảnh không có tính chất trên, như ảnh đồ ăn hoặc động vật ...

Tổng cộng có 300 hình ảnh thuộc tập easy và 700 ảnh thuộc tập hard. Việc chia này giúp đánh giá hiệu quả của mô hình so với những cách tiếp cận dựa trên xử lý ảnh thuần túy. Cách tiếp cận dựa trên xử lý ảnh sẽ dự đoán chính xác những ảnh thuộc tập easy nhưng có thể không dự đoán đúng những ảnh thuộc tập hard.

Bộ ảnh test sẽ đi qua hàm sinh dữ liệu (generator) đã trình bày bên trên để tạo bộ dữ liệu test.

6. Kết quả thực nghiệm

6.1 Google Street

6.1.1 Kết quả

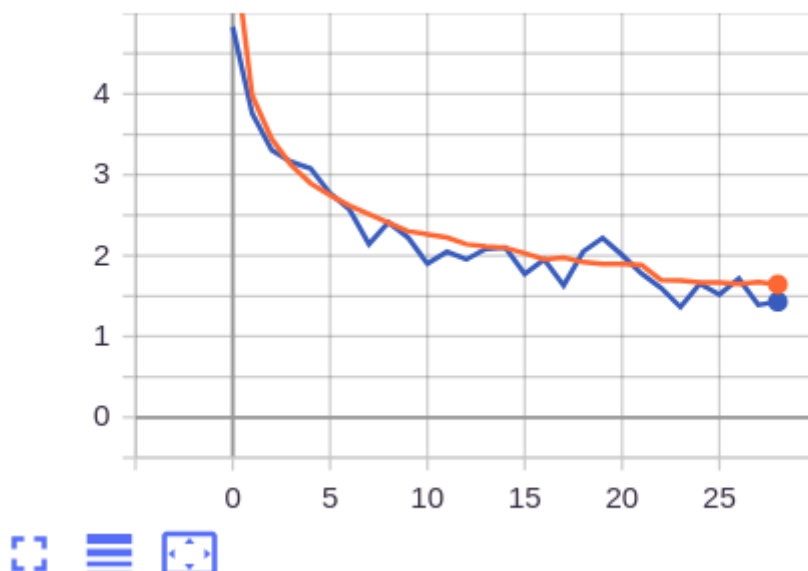
Model được đào tạo trên Google Colab với 7916 ảnh

- Epochs = 29, batchsize = 64
- Learning rate: 0.001
- Train angle error: 1.9797 độ
- Tổng thời gian train : 3 giờ

Kết quả test với 1000 ảnh :

- Test angle error: 2.2571 độ

epoch_loss



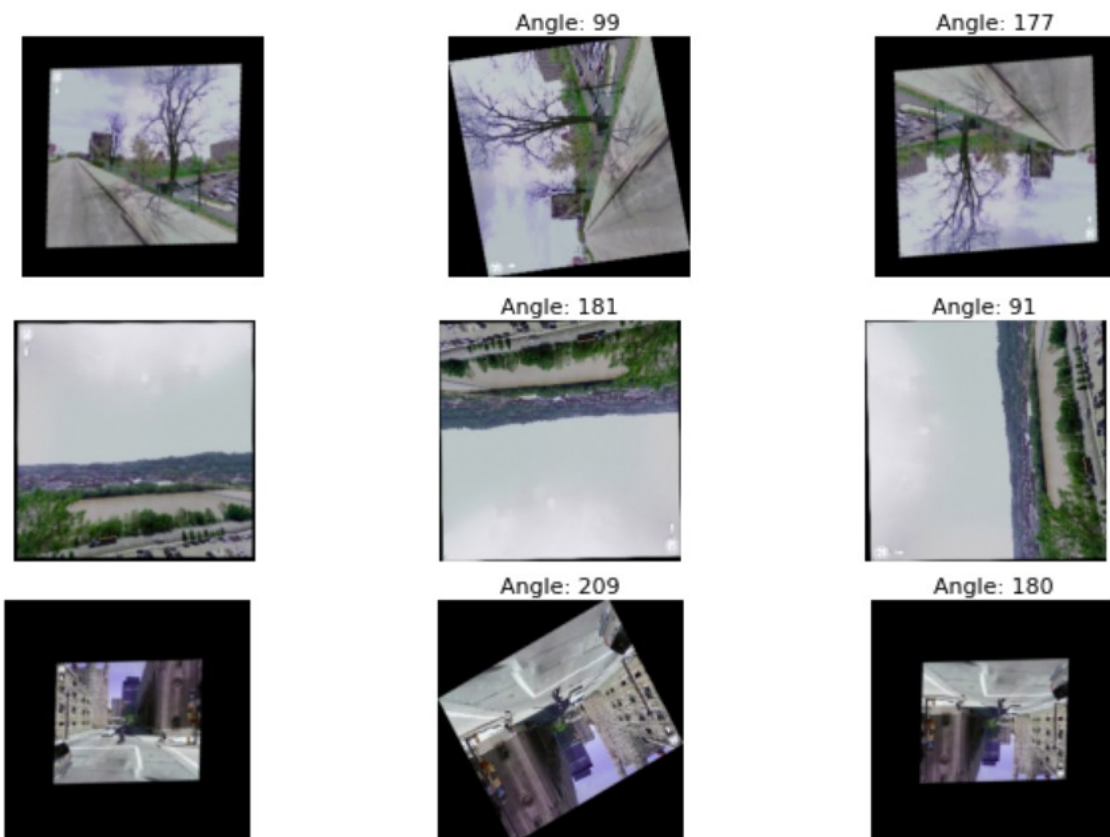


Đường màu xanh là loss của tập train

Đường màu cam là loss của tập validation

6.1.2 Những ảnh có angle loss cao

Ghi chú: 3 ảnh ở 3 cột theo thứ tự là ảnh gốc, ảnh đã bị xoay sai để đưa vào model và ảnh do model xoay lại





6.2 COCO – Cách tiếp cận 1

6.2.1 Kết quả

Model được train trên google colab.

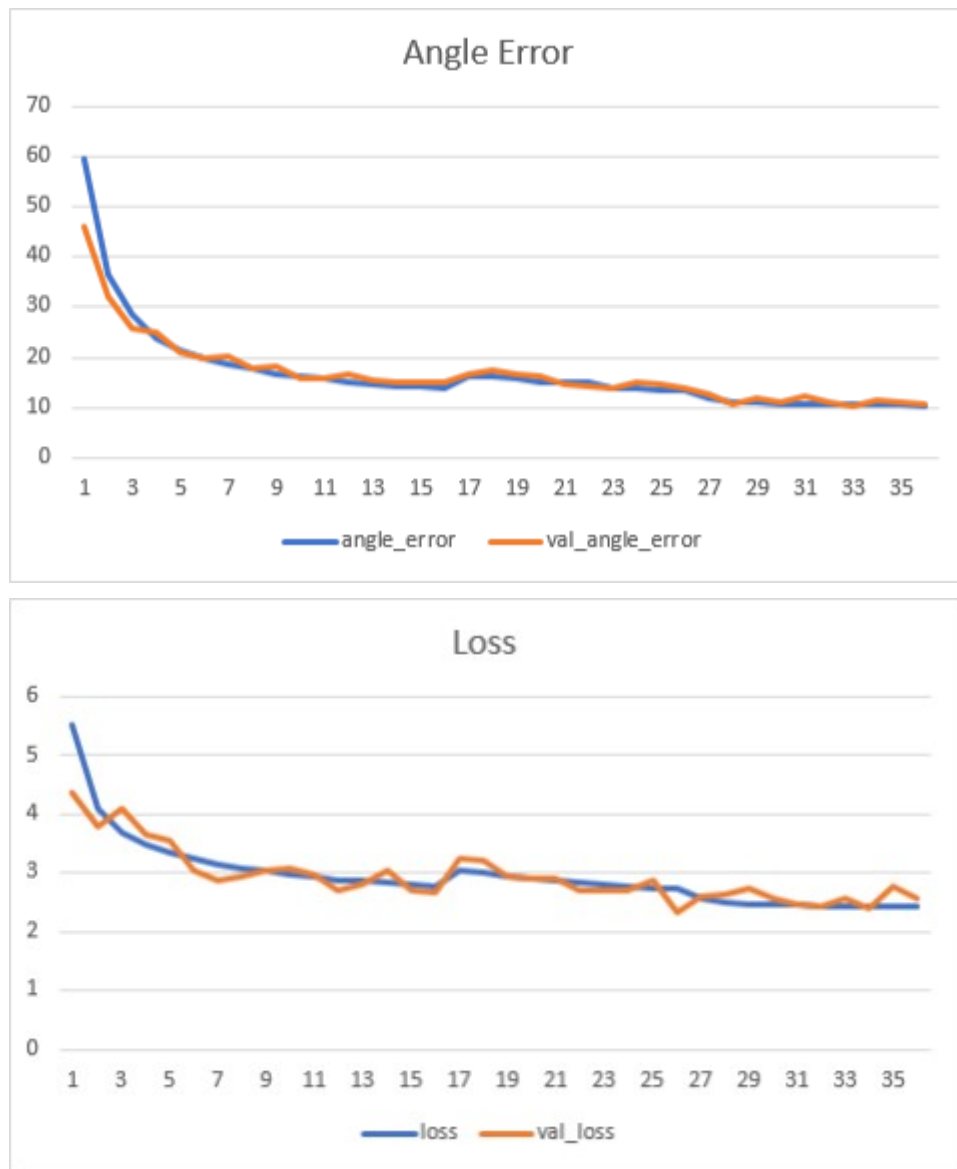
- Epochs = 37, batchsize = 64
- Thời gian mỗi epoch : 1867s
- Tổng thời gian train : Hơn 19h

Kết quả test với 4067 ảnh :

- Test loss: 2.3351807594299316
- Test angle error: 9.974504470825195
- 313 ảnh sau dự đoán có độ lệch so với ảnh gốc > 20 độ





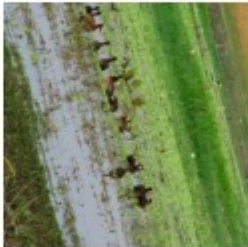



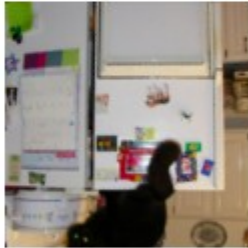

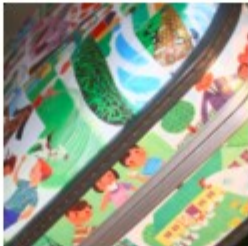
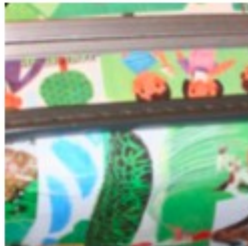
Kết quả test với tập dataset 1000 lọc sẵn :

- 700 ảnh easy:
 - Angle error: 7.81
 - Loss: 2.05
- 300 ảnh hard:
 - Angle error: 9.11
 - Loss: 2.21



6.2.2 Những ảnh có angle loss cao

Test model với 50 ảnh ngẫu nhiên, có 4 ảnh có độ lệch lớn hơn 20 độ.

Original	Rotated Angle: 167	Corrected Angle: 90
		
		
		
		

Có thể thấy, những trường hợp sai phần lớn là vì mô hình xác định sai góc phần tư của ảnh (do độ lệch các ảnh này sau khi xoay xấp xỉ bội của 90). Để cải thiện vấn đề này, ta có thể nghĩ đến phương án train 1 mô hình riêng hoặc dùng những phương pháp khác để xác định góc phần tư của ảnh.

6.3 COCO – Cách tiếp cận 2

6.3.1 Kết quả

Kết quả cho task 1:

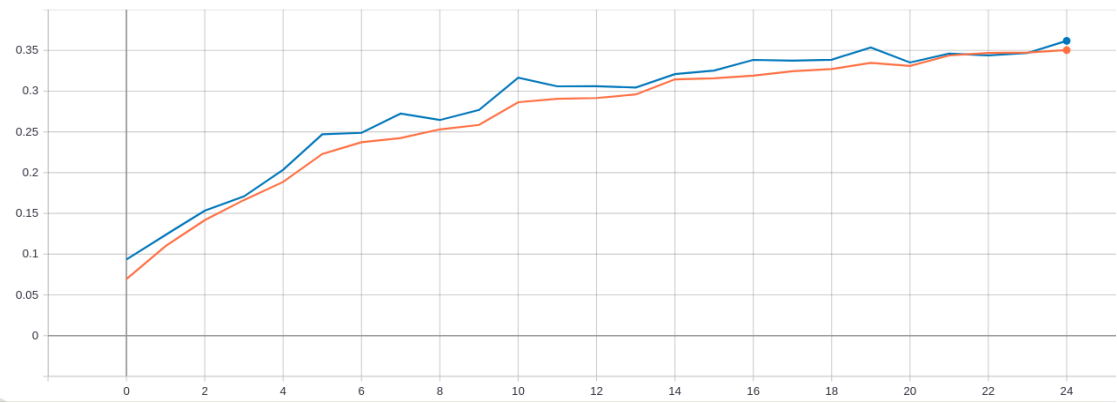
- Epochs: 22 epochs
- Learning_rate: $1e-4 \rightarrow 1e-6$
- Batch_size: 64
- Time to train: 3h14m



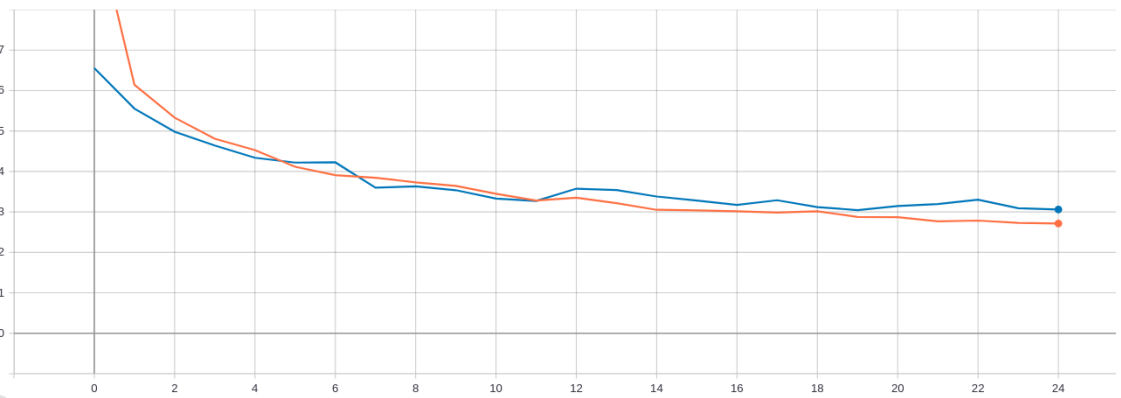
Kết quả cho task 2:

- Epochs: 24 epochs
- Learning_rate: $1e-4 \rightarrow 1e-6$
- Batch_size: 64
- Time to train: 3h34m

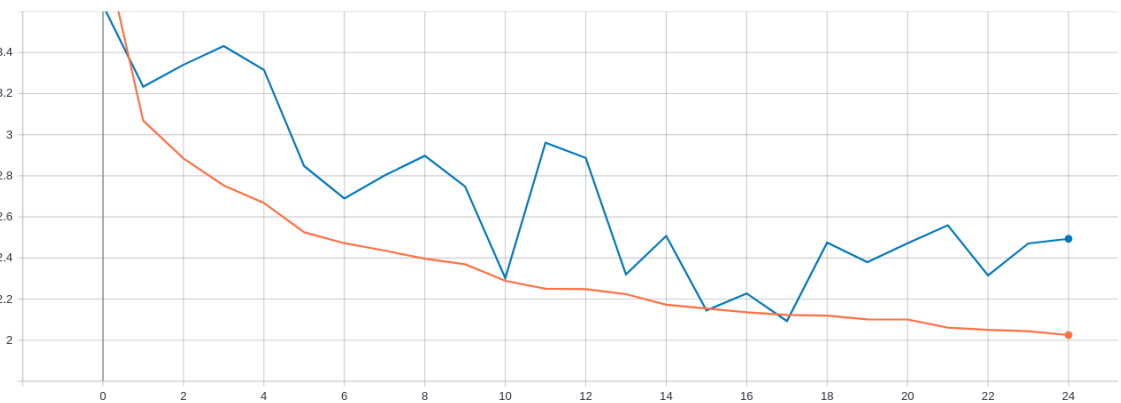
epoch_accuracy



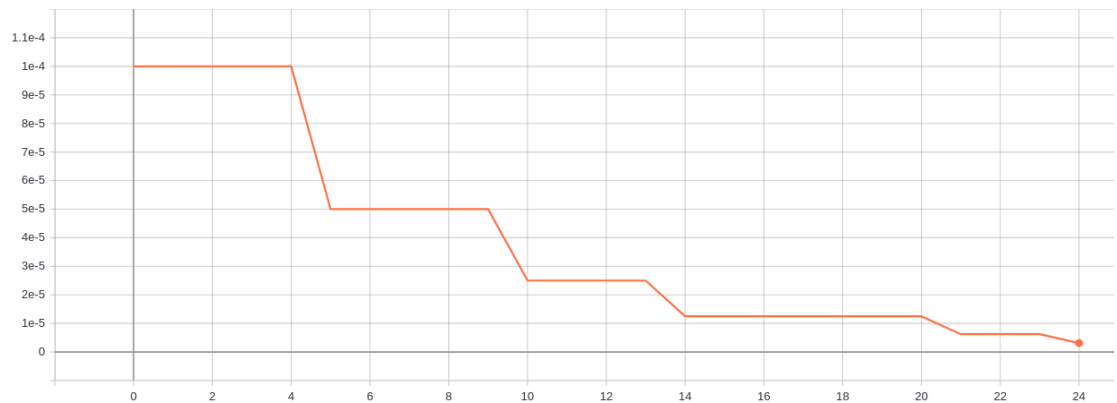
epoch_angle_error



epoch_loss



epoch_lr



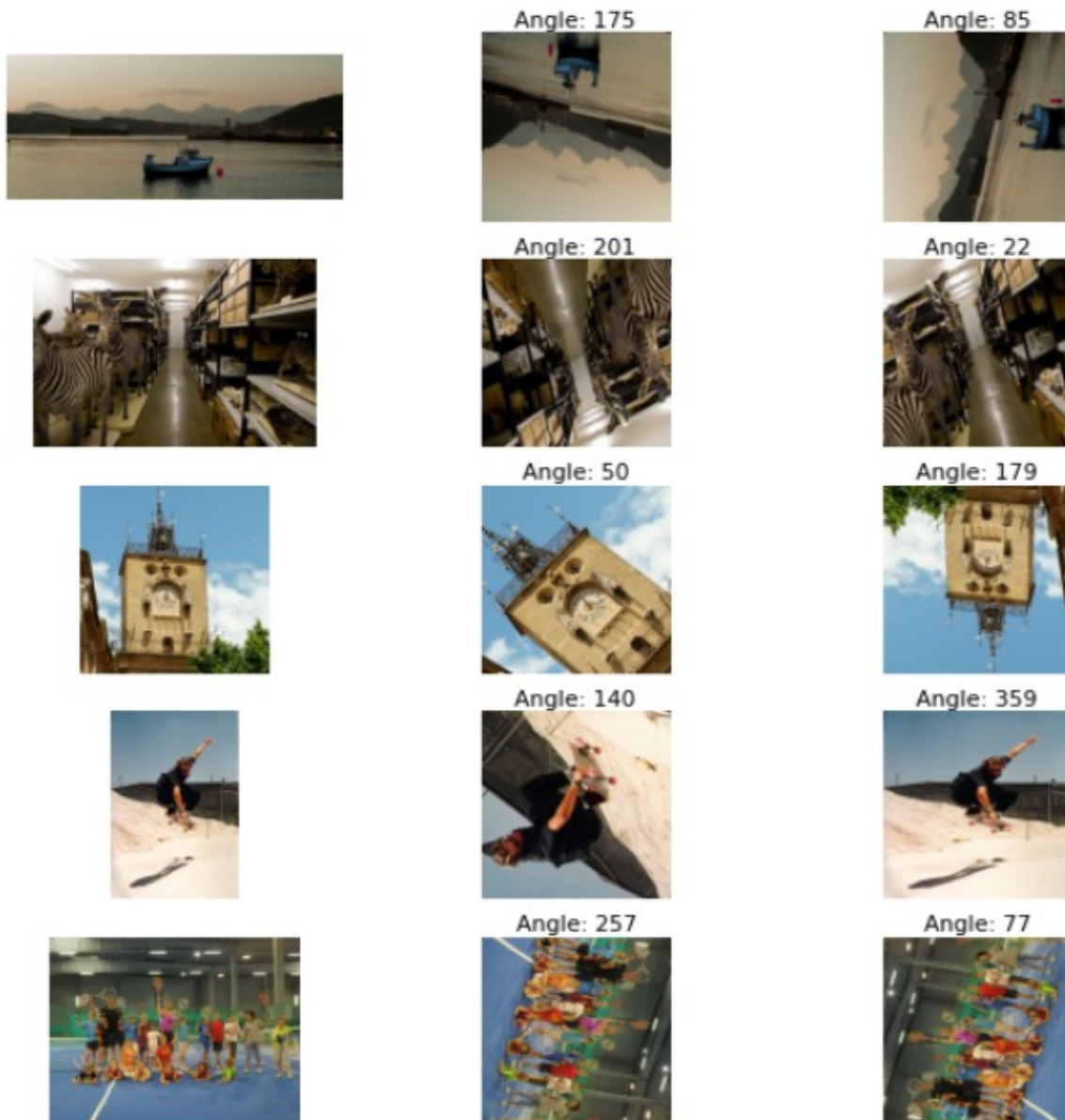
Kết hợp cả 2 task:

- Đánh giá tập test 4067 ảnh :
- Angle error: 34.5

Kết quả test với tập dataset 1000 lọc sẵn :

- 700 ảnh easy : Angle error: 35.9.
- 300 ảnh hard : Angle error: 37.3

6.3.2 Một số ảnh có loss cao:



=> Nhận xét :

Đối với ý tưởng chia ra 2 model để xử lý các giai đoạn như này, có vấn đề lớn như sau:

Độ chính xác của model 1 là ~90% không đủ cao để có thể làm tốt nhiệm vụ xác định góc phần tư. Do đó chỉ cần model 1 dự đoán góc phần tư sai, thì ngay lập tức model 2 sẽ dự đoán rất lệch.

Như vậy, để xử lý vấn đề dự đoán sai góc phần tư của model, ta cần phải dùng 1 phương pháp khác hoặc 1 mô hình khác chính xác hơn

6.4 So sánh với HoughLineP

Test 2 bộ ảnh hard và easy với phương pháp HoughLineP, nhóm thu được kết quả như sau :

- Angle Error của 300 ảnh easy : 87 độ
- Angle Error của 700 ảnh hard : 91 độ

Độ lệch của phương pháp HoughLineP là rất lớn. Trong những trường hợp ảnh input bị xoay > 90 độ, phương pháp này hoàn toàn ra kết quả sai. Nghĩa là phương pháp này không thể thực hiện với những ảnh có góc xoay lớn là những ảnh không có đường thẳng (cây cối, đường chân trời ...).

Do đó có thể thấy, cách tiếp cận bài toán xoay ảnh bằng áp dụng các mô hình học sâu đem lại hiệu quả tốt hơn nhiều với baseline sử dụng HoughlineP

7. Tài liệu tham khảo

1. Image Orientation Estimation with Convolutional Networks :
https://lmb.informatik.uni-freiburg.de/Publications/2015/FDB15/image_orientation.pdf
2. Correcting Image Orientation Using Convolutional Neural Networks :
<https://d4n.st.github.io/2017/01/12/image-orientation/>
3. Illustrated: 10 CNN Architectures :
<https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d>
4. Hough Line Transform :
https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html
5. Deep Residual Learning for Image Recognition :
<https://arxiv.org/abs/1512.03385>
6. Understanding Categorical Cross-Entropy Loss :
https://gombbru.github.io/2018/05/23/cross_entropy_loss/
7. Residual blocks — Building blocks of ResNet :
<https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec>
8. Skip connections and Residual blocks :
<https://kharshit.github.io/blog/2018/09/07/skip-connections-and-residual-blocks>
9. Giới thiệu mạng ResNet : <https://viblo.asia/p/gioi-thieu-mang-resnet-vyDZOa7R5wj>
10. Softmax Regression – Machine learning cơ bản :
<https://machinelearningcoban.com/2017/02/17/softmax/>