

Phân tích hành vi model

☑ Trắc nghiệm 7 ☑ Thực hành 5 🔗 Chạy trên Kaggle ⬇ PDF bài

Hai model cùng đạt accuracy cao trên tập test, nhưng khi đổi sang ảnh từ camera khác thì một model sai gần hết. Lý do có thể là model đã học một dấu hiệu tình cờ đi cùng nhãn, ví dụ một vật ở góc ảnh, thay vì học chính vật cần nhận ra. Bài này học cách kiểm model dựa vào thông tin gì: thay đổi một yếu tố của ảnh rồi đo kết quả đổi ra sao, sau đó dùng các phương pháp tô vùng ảnh ảnh hưởng tới dự đoán.

MỤC TIÊU

- So hai model trên từng mẫu và từng nhóm dữ liệu.
- Thiết kế thí nghiệm kiểm model có dùng dấu hiệu tình cờ không.
- Dùng occlusion, saliency, Grad-CAM và biết giới hạn của chúng.

Điểm tổng che khuất điều gì?

Accuracy là một trung bình. Một model có thể đúng ở ảnh sáng và sai ở ảnh tối, đúng người chụp cũ và sai người chụp mới. Trước khi nhìn heatmap, ta giữ từng prediction cùng ID, nhãn, score và các thuộc tính hợp lệ như camera hoặc kích thước vật thể.

Chia lỗi thành nhóm giúp đặt câu hỏi cụ thể: sai vì nhầm lớp, thiếu vùng nhỏ, background khác hay preprocessing? Trong mỗi nhóm, đọc nhiều chỉ số chứ không chỉ một: hai nhóm có cùng recall vẫn có thể có precision khác hẳn khi tỉ lệ lớp dương của chúng khác nhau. Ghi số mẫu cạnh mỗi chỉ số; một nhóm vài ảnh chỉ cho một gợi ý.

Nếu so A và B, cần cùng mẫu và cùng metric. Bảng chéo đúng/sai theo ID có bốn ô: cả hai đúng, chỉ A đúng, chỉ B đúng, cả hai sai. Hai ô giữa mới chứa thông tin phân biệt hai model, nên hai score tổng bằng nhau vẫn có thể che việc A đúng ở mẫu B sai và ngược lại.

Ô cả hai sai là nơi nên xem ảnh trước: đôi model chưa cứu được chúng, nhiều khi vì nhãn sai hoặc ảnh thiếu thông tin.

HỎI NHANH

Trên cùng 100 ảnh validation, A và B đều đúng 90 ảnh. Có 6 ảnh A đúng còn B sai. Có bao nhiêu ảnh cả hai cùng sai?

A 0

B 4

C 6

D 10

	B đúng	B sai
A đúng	84	6
A sai	6	4

hai ô giữa phân biệt hai model

ô cả hai sai: xem ảnh trước

A và B cùng đúng 90 trên 100 ảnh

A và B cùng đúng 90 ảnh, nhưng mỗi model đúng riêng 6 ảnh.

Chỉ xem ảnh sai được chọn theo cảm giác để bỏ sót một nhóm. Lấy mẫu đúng/sai theo tiêu chí đã ghi, kiểm cả những ảnh model tự tin nhưng sai. Dữ liệu chụp theo nhóm còn có phụ thuộc; 100 frame gần nhau không tương đương 100 buổi độc lập.

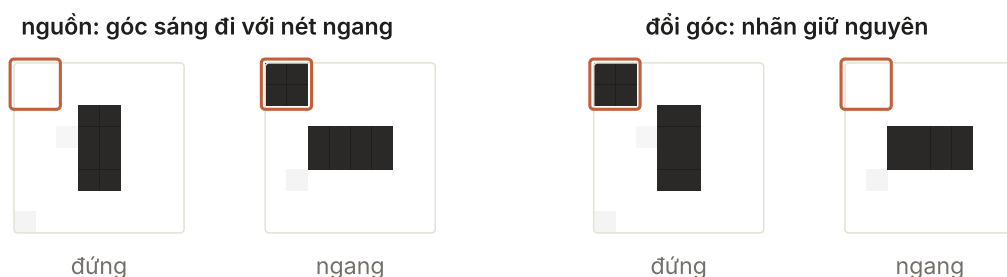
HỎI NHANH

Model sai 300 ảnh validation, nhưng ta chỉ đủ thời gian xem kỹ 30 ảnh. Chọn 30 ảnh nào thì bức tranh về lỗi đáng tin nhất?

- A 30 ảnh sai đầu tiên theo tên tệp, vì thứ tự tên tệp coi như ngẫu nhiên
- B 30 ảnh có heatmap rõ nhất, vì dễ giải thích cho cả đội
- C 30 ảnh có score sát ngưỡng 0.5, vì đó là những ảnh khó nhất
- D Lấy ngẫu nhiên trong từng nhóm, thêm vài ảnh sai mà score rất cao

Một đường tắt được tạo để ta kiểm rõ

Ta quay lại ảnh nét: nhãn 0 là đứng, 1 là ngang. Trong tập nguồn, ảnh ngang tình cờ có ô góc sáng, ảnh đứng có ô góc tối. Model có thể dùng góc thay vì nét. Trên validation cùng quy luật, cả hai cách đều đúng; thay đổi dấu góc sẽ tách chúng.



khung cam: ô góc 2x2 mà model A đọc; model B chỉ đọc vùng nét ở giữa

Ở tập nguồn, ô góc sáng luôn đi với nét ngang; tập kiểm đảo ô góc.

Ta tự thiết kế dữ liệu để biết quy tắc nhãn. Trong ảnh thật, xóa nền có thể vô tình xóa bóng hoặc viền vật thể; việc giữ nhãn phải được kiểm, không tự tuyên bố.

Python

```

import numpy as np
from sklearn.linear_model import LogisticRegression

vertical = np.zeros((8, 8))
vertical[2:6, 3:5] = 1
horizontal = np.zeros((8, 8))
horizontal[3:5, 2:6] = 1
orientation_filter = (horizontal - vertical) / 4

def make_images(count, seed, reverse_corner=False):
    rng = np.random.default_rng(seed)
    labels = np.arange(count) % 2
    images = np.zeros((count, 8, 8))
    for index, label in enumerate(labels):
        images[index] = vertical if label == 0 else horizontal
        marker = 1 - label if reverse_corner else label
        images[index, :2, :2] = marker
    images += rng.normal(0, 0.02, images.shape)
    return np.clip(images, 0, 1), labels

def features(images):
    corner = images[:, :2, :2].mean(axis=(1, 2))
    shape = (images * orientation_filter).sum(axis=(1, 2))
    return corner[:, None], shape[:, None]

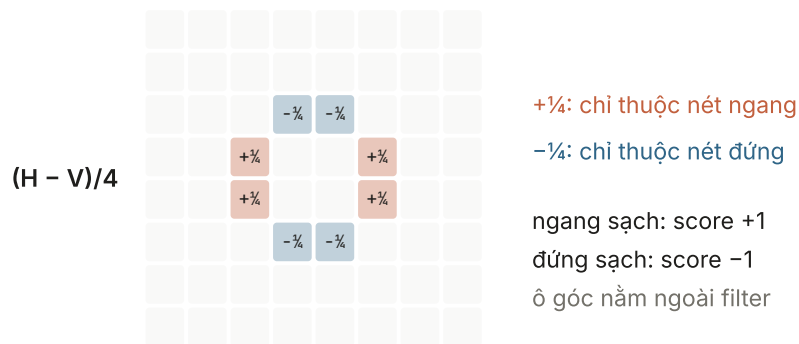
train_images, train_labels = make_images(80, 1)
normal_images, labels = make_images(40, 2)
changed_images, changed_labels = make_images(40, 2, reverse_corner=True)
train_corner, train_shape = features(train_images)
model_a = LogisticRegression().fit(train_corner, train_labels)
model_b = LogisticRegression().fit(train_shape, train_labels)
print("same labels after intervention:", np.array_equal(labels, changed_labels))
print("shape features preserved:",
      np.allclose(features(normal_images)[1], features(changed_images)[1]))

```

Model A chỉ thấy độ sáng trung bình của ô góc, model B chỉ thấy shape score (điểm hướng nét); cả hai là logistic regression một đặc trưng. Một CNN thật tự chọn đặc trưng nên ta không biết trước nó dùng gì; ở đây ta biết trước, nhờ vậy kiểm được phép đánh giá có phát hiện đúng điều đã cài hay không.

Shape filter nằm ở giữa ảnh, không đọc góc; hai dòng True xác nhận phép đổi góc giữ nhãn và giữ nguyên tín hiệu hướng.

Ví dụ: shape score bằng tổng ảnh nhân $(H - V)/4$. Template ngang và template đứng mỗi cái 8 ô, chung nhau 4 ô ở giữa. Filter vì thế có bốn ô $+1/4$ (chỉ thuộc nét ngang), bốn ô $-1/4$ (chỉ thuộc nét đứng), còn lại bằng 0. Với nét ngang sạch, bốn ô $+1/4$ đều sáng và bốn ô $-1/4$ đều tối, nên score bằng $+1$; nét đứng cho -1 . Dấu góc nằm ngoài filter nên không góp vào score.



Filter của model B chỉ có trọng số trên vùng nét, không có ở ô góc.

HỎI NHANH

Trong code, nét ngang chiếm [3:5, 2:6]. Dịch nó xuống một hàng thành [4:6, 2:6], ảnh sạch, dấu góc giữ nguyên. Shape score bằng bao nhiêu?

- | | |
|---|------|
| A | +1 |
| B | +0.5 |
| C | 0 |
| D | -1 |

So hai model bằng cùng một can thiệp

Ta đo accuracy ở ảnh nguồn bình thường và ảnh đổi góc. Cùng seed nhiều, cùng hướng nét, chỉ đổi nội dung dấu góc, nên khác biệt có một nguyên nhân được kiểm soát trong bộ dữ liệu này.

```
def predict_both(images):
    corner, shape = features(images)
    return model_a.predict(corner), model_b.predict(shape)

for name, images in [("normal", normal_images), ("changed corner", changed_images)]:
    pred_a, pred_b = predict_both(images)
    print(name, "A/B accuracy:",
          round(np.mean(pred_a == labels), 3),
          round(np.mean(pred_b == labels), 3))

pred_a, pred_b = predict_both(changed_images)
print("A correct, B wrong:", int(np.sum((pred_a == labels) & (pred_b != labels))))
print("B correct, A wrong:", int(np.sum((pred_b == labels) & (pred_a != labels))))
```

Trên normal, cả hai đúng; trên changed corner, A thất bại còn B giữ kết quả. Hai dòng đếm theo từng ảnh cho thấy mọi khác biệt đi một chiều: 40 ảnh B đúng mà A sai, không ảnh nào ngược lại. Ta có bằng chứng A phụ thuộc dấu góc trong dữ liệu đã viết.

Chưa suy ra được B bền với mọi camera: B dùng filter ở vị trí cố định, nên nét dịch chỗ hoặc đổi kích thước có thể làm shape score mất ý nghĩa. Mục “Tìm lỗi có hệ thống” đo điều này.

Trên model thật, một can thiệp hợp lệ có thể là thay background, che watermark hoặc đổi sáng trong mức hợp lý. Ta bắt đầu bằng giả thuyết, giữ nhãn, chạy cùng mẫu trước/sau và ghi tỉ lệ đổi nhãn cùng mức đổi score. Nếu can thiệp làm ảnh xa phân phối thật, kết quả nói về điều kiện ấy, chưa trực tiếp nói về sử dụng thường ngày.

HỎI NHANH

Trên tập `changed corner`, A có accuracy 0.0 chứ không phải khoảng 0.5. Con số này nói lên điều gì?

- A Quyết định của A đi theo dấu góc trên mọi ảnh
- B A đoán gần như ngẫu nhiên khi gặp ảnh lạ
- C Nhãn của tập `changed corner` đã bị đảo nhầm
- D A học được hướng nét, nhưng theo chiều ngược lại

HỎI NHANH

Nếu sinh `changed_images` bằng seed nhiều 3 thay vì seed 2 như `normal_images`, phép so trước và sau mất đi điều gì?

- A Không mất gì, vì nhiều chỉ có độ lệch chuẩn 0.02
- B Nhãn của hai tập không còn khớp nhau theo từng ID
- C Model B không còn dự đoán được trên ảnh mới
- D Mất phép so cặp, vì mỗi cặp khác nhau cả nhiều lần dấu góc

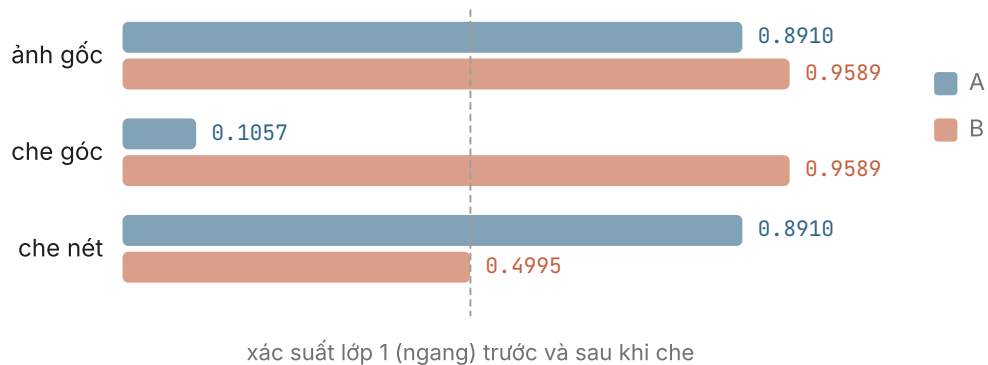
Occlusion: che vùng rồi xem score đổi

Occlusion thay một vùng ảnh bằng giá trị nền hoặc mẫu khác, chạy lại model và xem score của **cùng lớp** thay đổi. Nó gần với thử bỏ một đặc trưng: nếu score giảm nhiều, vùng ấy có ảnh hưởng theo phép thay đã dùng.

Ta chọn một ảnh ngang, lớp 1. Che góc và che vùng nét riêng. Code đọc xác suất của lớp 1 cho cả hai model, cả trước lẫn sau khi che.

Python

```
example = normal_images[1].copy()
without_corner = example.copy()
without_corner[:2, :2] = 0
without_shape = example.copy()
without_shape[2:6, 2:6] = 0
batch = np.stack([example, without_corner, without_shape])
corner, shape = features(batch)
score_a = model_a.predict_proba(corner)[: , 1]
score_b = model_b.predict_proba(shape)[: , 1]
print("rows: original, cover corner, cover shape")
print("A probability of class 1:", score_a.round(4).tolist())
print("B probability of class 1:", score_b.round(4).tolist())
```



Che góc làm A đổi hẳn dự đoán; che nét làm B đổi dự đoán.

Nhìn dòng A khi che góc ($0.891 \rightarrow 0.1057$) và dòng B khi che nét ($0.9589 \rightarrow 0.4995$). Mỗi model chỉ phản ứng với vùng nó đọc; che vùng còn lại không đổi score đến chữ số thứ tư. Che nét đồng thời bỏ đối tượng quyết định nhãn, nên ảnh ấy không phải một mẫu validation giữ nguyên nhãn; nó chỉ phục vụ đo độ phụ thuộc của score.

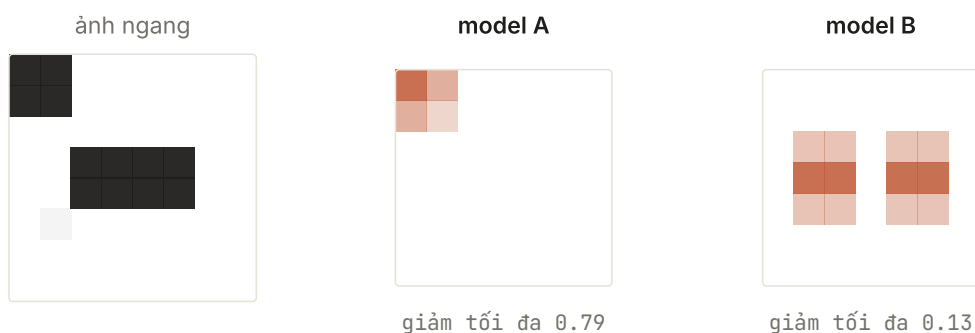
HỎI NHANH

Che góc làm A đổi argmax từ lớp 1 sang lớp 0. Vì sao ta vẫn đọc xác suất của lớp 1, cả trước lẫn sau khi che?

- A Để model không thể đổi nhãn dự đoán sau khi che
- B Để so cùng một đại lượng trước và sau khi che
- C Vì xác suất hai lớp luôn bằng nhau sau khi che
- D Vì lớp argmax mới luôn có score thấp hơn lớp cũ

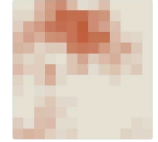
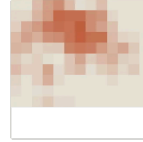
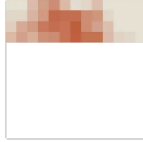
Chọn màu che ảnh hưởng kết quả. Ô đen trong ảnh rất sáng có thể tự là tín hiệu lạ. Muốn kết luận vững hơn, thử nhiều baseline, nhiều kích thước cửa sổ và nhóm ảnh; kiểm score có giảm vì vùng cần thiết hay vì tạo nhiễu ngoài phân phối. Một cửa sổ đo tác động khi mọi vùng khác giữ nguyên, nên khi nghi model có hai tín hiệu thay thế nhau, che cả tổ hợp vùng.

Trên ảnh lớn, occlusion map trượt cửa sổ khắp ảnh, mỗi vị trí một lượt forward. Chi phí tăng theo số vị trí nhân số ảnh; gộp các ảnh đã che thành batch như code trên, và chỉ chạy trên một mẫu ảnh chọn theo tiêu chí.



mỗi ô: score lớp 1 giảm bao nhiêu khi che cửa sổ 2x2 bắt đầu ở đó

Bản đồ occlusion: A chỉ nhảy với ô góc, B nhảy với vùng nét.



che ô hàng 1, cột 5:
 $p(\text{"espresso"}) = 0.50$ (gốc 0.85)

che ô hàng 4, cột 13:
 $p(\text{"espresso"}) = 0.90$ (gốc 0.85)

che ô hàng 10, cột 13:
 $p(\text{"espresso"}) = 0.93$ (gốc 0.85)

vùng che làm $p(\text{"espresso"})$
giảm mạnh nhất được giữ sáng:
mặt tách cà phê

Che từng vùng ảnh: vùng nào làm xác suất lớp giảm nhiều thì càng cam.

HỎI NHANH

Ảnh 224×224 , cửa sổ che 16×16 , bước trượt 8 theo cả hai chiều, không padding. Không kể ảnh gốc, một ảnh cần bao nhiêu lượt forward để vẽ occlusion map?

- A 196
- B 729
- C 784
- D 50 176

Saliency và Grad-CAM thêm góc nhìn nào?

Saliency dùng gradient của score lớp theo input để đo nhạy cảm cục bộ: thay một pixel rất nhỏ có thể đổi score thế nào. Occlusion cần một lượt forward cho mỗi vùng; saliency chỉ cần một lượt forward và một lượt backward cho cả ảnh.

Cái giá là nó chỉ nhìn thay đổi vô cùng nhỏ: gradient lớn không luôn nghĩa pixel đó là nguyên nhân ngữ nghĩa quan trọng, và gradient có thể bão hòa hoặc nhạy với noise. Qua ReLU hay max pooling, gradient bằng 0 ở các nhánh không được chọn, nên một vùng có ích vẫn có thể có saliency bằng 0.

Hai model của bài đọc pixel bằng phép tính viết lại được trong torch, nên autograd tính saliency trực tiếp. Ta lấy gradient của logit (score trước sigmoid) theo 64 pixel của ảnh ngang dùng ở mục occlusion, đo phần gradient nằm ở ô góc, rồi so với map của một ảnh đúng.

```

import torch

shape_filter = torch.tensor(orientation_filter)
readers = {
    "A": (model_a, lambda pixels: pixels[:2, :2].mean()),
    "B": (model_b, lambda pixels: (pixels * shape_filter).sum()),
}

def saliency(name, image, use_probability=False):
    model, read_feature = readers[name]
    pixels = torch.tensor(image, requires_grad=True)
    score = (float(model.coef_[0, 0]) * read_feature(pixels)
             + float(model.intercept_[0]))
    if use_probability:
        score = torch.sigmoid(score)
    score.backward()
    return pixels.grad.abs().numpy()

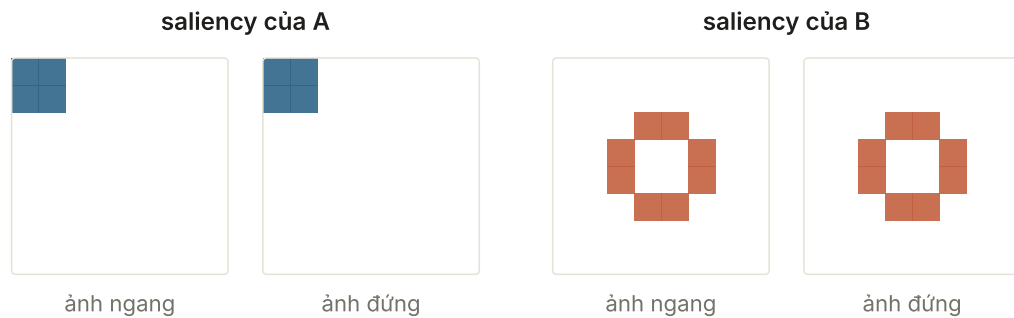
corner = np.zeros((8, 8), dtype=bool)
corner[:2, :2] = True
for name in ["A", "B"]:
    horizontal_map = saliency(name, normal_images[1])
    vertical_map = saliency(name, normal_images[0])
    print(name, "corner:", round(horizontal_map[corner].sum() / horizontal_map.sum(), 3),
          "| max:", round(horizontal_map.max(), 4),
          "| same on vertical:", np.allclose(horizontal_map, vertical_map))
print("B max grad of probability:", round(saliency("B", normal_images[1], True).max(), 4))

```

A dồn toàn bộ gradient vào ô góc, B không có chút nào ở đó: khớp với occlusion. Số 1.0688 tính tay được: logit của A là hệ số của A nhân trung bình bốn ô góc, nên đạo hàm theo mỗi ô bằng hệ số ấy chia 4.

Cột `same on vertical` cho thấy giới hạn: với model tuyến tính theo pixel, gradient chính là trọng số, giống hệt nhau cho ảnh ngang và ảnh đứng. Saliency nói model nhạy ở đâu, chưa nói ảnh này có gì ở đó.

Dòng cuối là bảo hòa: lấy gradient của xác suất thay cho logit, mọi giá trị nhân thêm $p(1 - p)$ với $p = 0.9589$, nên map nhỏ đi khoảng 25 lần đúng ở ảnh model rất tự tin.



model tuyến tính: gradient chính là trọng số, giống nhau trên mọi ảnh

Saliency: A dồn vào ô góc, B dồn vào vùng nét.

HỎI NHANH

Saliency của B trên ảnh ngang và trên ảnh đứng trùng nhau từng ô. Điều đó cho biết gì?

- A Model tuyến tính có gradient không phụ thuộc ảnh
- B Hai ảnh có nội dung giống nhau ở vùng giữa
- C Autograd chưa bật nên map trả về giá trị mặc định
- D B đã học đúng hướng nét trên mọi ảnh có thể gặp

Grad-CAM dùng feature map ở một lớp convolution thay cho pixel. Với score s_c và feature map A^k , lấy trung bình gradient theo không gian để có trọng số α_k , rồi cộng các map:

$$\alpha_k = \frac{1}{HW} \sum_{i,j} \frac{\partial s_c}{\partial A_{i,j}^k}, \quad M_c = \text{ReLU} \left(\sum_k \alpha_k A^k \right).$$

k là kênh, i, j là vị trí. Kênh có α_k âm làm giảm score lớp c , và ReLU chỉ giữ những vùng đẩy score lên.

Chọn lớp nào để tính là một quyết định: lớp nông giữ chi tiết vị trí nhưng feature còn là cạnh, góc; lớp conv cuối mang ngữ nghĩa của lớp nhưng với ảnh 224×224 có thể chỉ còn

7×7 ô, mỗi ô ứng với một vùng 32×32 pixel. Phóng map lên kích thước ảnh làm nó mờ, không làm nó chính xác hơn.

Grad-CAM tổng quát hóa CAM gốc. CAM chỉ dùng được khi mạng kết thúc bằng global average pooling rồi một lớp linear: map của lớp c khi đó là $\sum_k w_{c,k} A^k$ với w là trọng số của lớp linear. Grad-CAM thay $w_{c,k}$ bằng α_k lấy từ gradient nên dùng được với kiến trúc bất kỳ, không phải sửa và train lại mạng.

Trên CNN thật, Grad-CAM giữ A^k và gradient theo A^k ở lớp đã chọn bằng hook; [bài báo Grad-CAM](#) mô tả phương pháp, và Captum có sẵn cài đặt.

HỎI NHANH

Lớp được chọn có hai kênh 2×2 . Kênh $A^1 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$ có mọi gradient bằng 0.5. Kênh $A^2 = \begin{bmatrix} 0 & 3 \\ 1 & 0 \end{bmatrix}$ có gradient $\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$. Map Grad-CAM M_c bằng bao nhiêu?

A $\begin{bmatrix} 0.5 & 1.5 \\ 1.5 & 0.5 \end{bmatrix}$

B $\begin{bmatrix} 0.5 & 0 \\ 1 & 0.5 \end{bmatrix}$

C $\begin{bmatrix} 0.5 & 0 \\ 0.5 & 0.5 \end{bmatrix}$

D $\begin{bmatrix} 0.5 & -1.5 \\ 0.5 & 0.5 \end{bmatrix}$

$A^1 \cdot \alpha_1 = 0.5$	$A^2 \cdot \alpha_2 = -0.5$	M_c
1 0	0 3	0.5 0
2 1	1 0	0.5 0.5

$\xrightarrow{\text{ReLU}}$

α = trung bình gradient theo không gian

$$0.5 \cdot A^1 - 0.5 \cdot A^2 = (0.5, -1.5; 0.5, 0.5) \rightarrow \text{ReLU}$$

Grad-CAM trên lưới nhỏ: ô góc trên phải ra -1.5, ReLU đưa về 0.

Heatmap tạo một giả thuyết để kiểm, chẳng hạn model nhìn watermark. Nó chưa chứng minh nhân quả. Có thể kiểm bằng che watermark giữ đối tượng, thay vị trí hoặc làm thử nghiệm nhiều ảnh. Với Transformer, attention map cũng không tự thay thế can thiệp này.

Tìm lỗi có hệ thống

Một kết quả tốt ở một phép thử chưa nói gì về các phép thử khác. Với ứng dụng nét phấn, ta lập ma trận kiểm: mỗi hàng là một điều kiện (dấu góc, vị trí nét, độ dày, sáng tối), mỗi cột một model, mỗi ô là accuracy trên cùng 40 ảnh gốc. Hàm `shift_strokes` dịch phần ảnh từ hàng 2, cột 2 trở đi xuống dưới (hàng trên quay vòng lên trên, ở đó chỉ có nhiễu), giữ nguyên ô góc, để mỗi hàng chỉ đổi một yếu tố.

Python

```
def shift_strokes(images, rows):
    shifted = images.copy()
    shifted[:, 2:, 2:] = np.roll(images[:, 2:, 2:], rows, axis=1)
    return shifted

conditions = {
    "normal": normal_images,
    "changed corner": changed_images,
    "shift 1 row": shift_strokes(normal_images, 1),
    "shift 2 rows": shift_strokes(normal_images, 2),
}

print("condition      A      B")
for name, images in conditions.items():
    pred_a, pred_b = predict_both(images)
    print(f"{name:15s} {np.mean(pred_a == labels):.3f} {np.mean(pred_b == labels):.3f}")
```

Ma trận cho thấy mặt còn lại của bức tranh. A không hề bị ảnh hưởng bởi dịch nét vì chỉ nhìn góc; B rơi xuống 0.775 rồi 0.5. Dịch một hàng đưa shape score của ảnh ngang về khoảng 0, nên với ảnh ngang B quyết định theo nhiễu; dịch hai hàng làm score âm và B gọi mọi ảnh ngang là đúng.

Mỗi model thắng ở một hàng, nên không ô nào cho phép nói “model bền” chung chung. Những điều kiện chưa có hàng, như nét dày hơn hay sáng tối, vẫn là điều chưa biết; điểm tổng và phân tích theo nhóm cũng cần thiết.

	A (đọc góc)	B (đọc nét)	
normal	1.000	1.000	mỗi model thắng ở một hàng khác
changed corner	0.000	1.000	
shift 1 row	1.000	0.775	
shift 2 rows	1.000	0.500	

A sụp khi đổi ô góc; B giữ được nhưng tụt khi nét dịch chỗ.

HỎI NHANH

Hãy đoán trước một hàng chưa có trong ma trận: dịch nét hai hàng và đổi dấu góc cùng lúc. Accuracy của A và B lần lượt là bao nhiêu?

A A 1.0, B 0.5

B A 0.0, B 1.0

C A 0.5, B 0.5

D A 0.0, B 0.5

Một dạng tìm lỗi khác là ảnh adversarial (ảnh đối kháng): thay đổi nhỏ có chủ đích làm model đổi quyết định. Fast Gradient Sign Method (FGSM) đi một bước $\delta = \epsilon \text{sign}(\nabla_{\mathbf{x}} L)$, thỏa ràng buộc mỗi pixel đổi không quá ϵ , tức $\|\delta\|_{\infty} \leq \epsilon$. Với logit tuyến tính $z = \mathbf{w}^T \mathbf{x} + b$, bước ngược dấu $\mathbf{w}$ làm z giảm đúng $\epsilon \|\mathbf{w}\|_1$: mỗi pixel đổi rất ít nhưng mọi pixel cùng đẩy một chiều. Ảnh càng nhiều pixel có trọng số, tổng lực đẩy càng lớn.

Trong nhiệm vụ “tìm ảnh làm model sai”, tránh chỉ tối ưu một mẫu rồi gọi đó là đánh giá tổng quát. Xác nhận nhãn, giữ ngân sách truy vấn nếu có, lưu input chính xác và seed.

Sau mỗi bước, chiếu nhiều về giới hạn đề cho và về miền pixel hợp lệ, rồi kiểm lại model còn sai không. Đọc đề cụ thể trước khi truy vấn hoặc dùng dữ liệu ngoài.

Đặt phép che đúng chỗ

HỎI NHANH

Pipeline đưa ảnh trong $[0, 1]$ qua `Normalize` theo mean $(0.485, 0.456, 0.406)$ và std của ImageNet, rồi vào model. Code occlusion gán 0 vào vùng che **sau** bước `Normalize`. Vùng che thực chất có màu gì?

- A Màu trung bình của ImageNet, một màu xám hơi ấm
- B Màu đen, vì giá trị 0 luôn là pixel tối nhất
- C Màu trắng, vì `Normalize` đảo dấu giá trị pixel
- D Không xác định được, vì `Normalize` là phép phi tuyến

gán 0 trước `Normalize`



gán 0 sau `Normalize`



sau `Normalize`, 0 ứng với mean ImageNet:

(124, 116, 104)

che đen phải gán ≈ -2.1 ở kênh R

Gán 0 trước `Normalize` cho ô đen; gán 0 sau `Normalize` cho ô màu trung bình.

Khi đi thi

- Lưu prediction theo ID từ baseline để so model mới trên từng ảnh.

- Mỗi lần thử chỉ đổi một yếu tố, đo bằng cùng scorer trên cùng ảnh.
- Heatmap dùng để chọn thí nghiệm, không thay cho bằng chứng.

Tóm tắt

- Điểm tổng che khuất lỗi theo nhóm; so hai model bằng bảng chéo đúng/sai trên cùng mẫu.
- Kiểm một giả thuyết bằng can thiệp: đổi đúng một yếu tố, giữ nhãn, chạy cùng mẫu trước và sau.
- Occlusion đo tác động của việc che một vùng; saliency đo độ nhạy cục bộ; Grad-CAM cho bản đồ thô ở một lớp conv. Heatmap chỉ là giả thuyết, chưa phải bằng chứng.
- Ma trận kiểm cho thấy một model có thể thắng ở điều kiện này và thua ở điều kiện khác.

Kiểm tra cuối bài

CÂU 1

Một model phát hiện ảnh lỗi đoán ngẫu nhiên. Nó không nhìn ảnh, cứ mỗi ảnh gọi “lỗi” với xác suất $1/2$. Camera X có 50% ảnh lỗi thật, camera Y có 10%. Đại lượng nào **khác nhau** giữa hai camera?

- A Recall, tức tỉ lệ ảnh lỗi thật được gọi là lỗi
- B False positive rate, tức tỉ lệ ảnh tốt nhưng bị gọi là lỗi
- C Precision, tức tỉ lệ lỗi thật trong các ảnh bị gọi là lỗi
- D Tỉ lệ ảnh bị model gọi là lỗi trên toàn nhóm

PHÒNG THEO CMU 10-601 · Exam 2 Practice · Q5.3

CÂU 2

Làm occlusion trên một ảnh. Che riêng vùng nét thì score của lớp đúng giảm chưa tới 0.02. Che riêng watermark cũng vậy. Nhưng che cả hai cùng lúc thì score rơi từ 0.95 xuống 0.10. Kết quả này cho thấy gì?

- A Model không dùng vùng nào trong hai vùng đã che
- B Hai vùng thay thế được cho nhau, chỉ cần còn một vùng là giữ được score
- C Che cả hai làm ảnh quá lạ, nên có thể bỏ con số này đi
- D Phải che lần lượt từng pixel thì score mới có ý nghĩa

CÂU 3

Logit $z = \mathbf{w}^\top \mathbf{x} + b$, trong đó $\mathbf{w}$ có 16 phần tử bằng ± 0.5 , còn lại bằng 0. Nhiễu thỏa $\|\delta\|_\infty \leq 0.1$ và không chạm biên miền pixel. z giảm được nhiều nhất bao nhiêu?

A 0.05

B 0.2

C 0.8

D 1.6

CÂU 4

Giữ nguyên nét và nhãn thật, chỉ thay watermark, thì score của lớp đúng giảm mạnh trên 200 ảnh validation. Dữ liệu này cho phép kết luận điều gì?

A Mọi lỗi của model trên test đều do watermark gây ra

B Model khác cùng kiến trúc cũng sẽ phụ thuộc watermark

C Watermark là nguyên nhân khiến ảnh có nhãn "ngang"

D Model này phụ thuộc watermark trên loại ảnh đã kiểm

CÂU 5

Một mạng một chiều gồm một filter convolution cho ra $z_1, \dots, z_d$, các giá trị khác nhau, rồi $g = \sigma(\max_j z_j)$. Ảnh có pattern ở hai vị trí: z_3 lớn nhất, z_7 lớn nhì. Saliency theo z cho kết quả nào?

A Chỉ $\partial g / \partial z_3$ khác 0, và bằng $g(1 - g)$

B Chỉ $\partial g / \partial z_3$ khác 0, và bằng 1

C $\partial g / \partial z_3$ và $\partial g / \partial z_7$ cùng khác 0

D $\partial g / \partial z_i = g(1 - g)$ với mọi i

PHÒNG THEO MIT 6.036 · Final, Spring 2022 · Q8(d)

CÂU 6

Đề phân loại hình tròn, tam giác và chữ nhật trên nền trắng. Vị trí, cỡ, góc xoay và màu của hình đều ngẫu nhiên, và chỉ có 200 ảnh có nhãn. Model đạt macro-F1 0.90 trên validation, nhưng hầu hết lỗi là ảnh hình màu đỏ bị gọi là tam giác. Thí nghiệm nào kiểm nhanh nhất giả thuyết “model dùng màu”?

A Vẽ Grad-CAM cho vài ảnh tam giác đỏ và xem vùng sáng

B Đổi màu hình bằng code, giữ hình dạng, rồi so dự đoán theo ID

C Train thêm epoch rồi so lại macro-F1 trên validation

D Bỏ kênh R khỏi mọi ảnh rồi train lại model từ đầu cho chắc

PHÒNG THEO USA-NA-AIO 2026 · Vòng 2 · Ngày 2 · Problem 4

CÂU 7

Ảnh xe trong train luôn chụp ngang, đầu xe quay sang phải, cùng kích thước và vị trí. Một đội dùng lớp không chia sẻ trọng số: mỗi vị trí có một bộ filter riêng, giống template cố định của model B. Cần kiểm gì trước khi tin model này?

- A Không cần kiểm, vì lớp không chia sẻ trọng số luôn tốt hơn convolution
- B Không cần kiểm, cứ đổi sang convolution vì nó luôn tốt hơn
- C Kiểm xem test có giữ vị trí và hướng như train không, bằng cách thử dịch và lật
- D Tăng số epoch để model tự học bất biến với phép dịch

PHÒNG THEO CMU 10-601 · Exam 3 Practice · Q1.4(b)

CÂU 8

Heatmap sáng ở watermark trên vài ảnh. Thí nghiệm nào kiểm giả thuyết “model dùng watermark” tốt nhất?

- A Xem thêm heatmap của 20 ảnh khác và đếm số ảnh sáng ở watermark
- B Chọn ảnh có heatmap rõ nhất để đưa vào báo cáo cuối
- C Che toàn bộ ảnh rồi coi mức giảm là tác động của watermark
- D Che hoặc thay watermark trên nhiều ảnh, rồi so score của cùng một lớp

CÂU 9

Có 100 frame liên tiếp từ cùng một video, và 100 ảnh từ 100 buổi chụp khác nhau. Vì sao không thể coi hai tập này như nhau, cùng là 100 quan sát độc lập?

- A Các frame cùng video có chung cảnh và ánh sáng, nên phụ thuộc nhau
- B Số frame quá nhiều nên metric không tính được trên tập video
- C `model.eval()` làm các frame video độc lập với nhau
- D Shuffle ID trước khi chấm sẽ làm mất mọi tương quan giữa các ảnh

CÂU 10

Đề cho phép mỗi pixel đổi tối đa $8/255$. Ta tìm được một ảnh làm model sai, nhưng có pixel lệch $12/255$ so với ảnh gốc. Phải xử lý thế nào?

- A Nộp luôn, vì model đã sai và độ lệch rất nhỏ, khó thấy bằng mắt
- B Chiếu nhiều về trong giới hạn, rồi kiểm lại xem model còn sai không
- C Giữ ảnh nhưng đổi nhãn để model bị tính là dự đoán sai
- D Nới giới hạn lên $12/255$ cho mọi ảnh để kết quả nhất quán

Đọc thêm

- Selvaraju và cộng sự — [Grad-CAM](#): xem gradient trên feature map và giới hạn độ phân giải.
- [PyTorch — Adversarial Example Generation](#): đọc ví dụ có giới hạn thay đổi, kiểm nhãn và gradient.
- [Captum — Occlusion](#): đối chiếu baseline, sliding window và target khi áp dụng occlusion cho model thật.