Carrying out error analysis:

việc đánh giá error rất quan trọng trong việc cải thiện model, bằng việc chia làm các loại “sai”

VD: nhận diện mèo

các tiêu chí đánh giá error: ảnh mờ, là chó, là hổ thì

với mỗi các tiêu chí trên sẽ có số lượng ảnh lỗi theo đó thống kê lại thì ta sẽ thấy được

loại nào đang lỗi nhiều nhất,… từ đó có thể đưa ra chiến lược phù hợp

Build your first system quickly, then iterate:

build model đầu tiên thật nhanh, rồi sau đó cải thiện sau (lazy type =))

Training and testing on different distributions:

nếu tập test và traing đến từ nhiều nguồn

VD: nhận diện mèo

10.000 ảnh từ mobile người dùng nên chất lượng kém, mờ

200.000 ảnh crawl từ web thì độ phân giải cao, rất art

Cách 1: mix lại với nhau và chia làm train/dev/test như bình thường

cách này có flaws: tập dev sẽ nhiều phần của art, do vậy phần mobile sẽ bị lu mờ

mà mình lại làm app nhận diện mèo cho người dùng

do vậy target sẽ thành ra focus vào ảnh nghệ thuật – rõ ràng là sai vl

-> KHÔNG DÙNG

Cách 2: tập train 205.000 ảnh bao gồm 200.000 ảnh từ web và 5.000 ảnh từ mobile

tập dev/test gồm 2.500 ảnh CHỈ từ mobile KHÔNG có ảnh từ web

cách này thì target của mình vs mobile được thỏa mãn

flaws: tập train, dev ,test đến từ các nguồn khác nhau

-> có thể sai một xíu nhưng tổng thể và lâu dài là đúng hướng

data mismatch: hiện tượng tập train-dev vs dev-test bị lệch, chứng tỏ data chưa phù hợp

việc này thường do data đến từ nghiều nguồn, cách giải quyết như sau:

lấy thêm data (of course)

nghiên cứu xem vì sao lại sai và đánh giá thêm

lấy thêm data VD: nhận diện giọng nói

giọng nói + nhiễu (car noise, vọng âm) = data mới

có vấn đề có thể xảy ra là giọng nói bị overfit vs car noise

Transfer Learning

Graphical user interface, text, application

Description automatically generated

Multi-task Learning

Chart

Description automatically generated with low confidence

When Multi-task Learning

Graphical user interface, text, application, Word

Description automatically generated

end-to-end learning: trên thực tế không phải lúc nào cũng map trực tiếp từ x -> y được, hình dưới là một ví dụ

Graphical user interface, application, Word

Description automatically generated

khi có ít data thì phải làm thành nhiều giai đoạn, nhưng ta luôn mong muốn train đc 1 mạng đủ lớn để map trực tiếp

để làm đc điều này đòi hỏi một lượng siêu lớn các data, thì mạng mới hoạt động tốt được

Nhưng cũng có những trường hợp công việc lại cần phải tách ra:

data của 2 công việc là độc lập, và số lượng siêu nhiều, hoặc lệch nhau

Diagram

Description automatically generated

lấy ví dụ nhận diện khuông mặt = nhận diện mặt + định danh mặt

việc nhận diện mặt đã cần rất nhiều data mà mục đích khác định nhanh

việc định danh có thể cần ít hơn rất nhiều so với nhận diện

mà lại muống làm 2 công việc thì đòi hỏi ít nhất phải = vs số để nhận diện khuôn mặt

=> việc gộp lại có khi lại làm mất thời gian hơn, không dùng “end-to-end”

Machine Translation

có thể dùng “end-to-end”

trong hệ thống thực tế cần trade-off khi dùng “end-to-end”, để dùng được “end-to-end” đòi hỏi rất nhiều thời gian và số lượng cho data, và nhiều use case là bất khả thi do nhiều lí bo, privacy, bảo mật, nhạy cảm, ...

do vậy các phương pháp truyền thống tuy có phức tạp hơn như: chia nhỏ task làm nhiều công đoạn

VD: nhận diện khuôn mặt = nhận mặt + định danh sẽ dễ triển khai cũng như hợp lí hơn

Whether to use end-to-end deep learning

Graphical user interface, text, application

Description automatically generated

ý là khi có ít data thì phải tự bịa, sáng tạo thêm các bước trung gian, còn khi có nhiều thì để mạng nó tự tìm ra trọng số