1. Các thành phần trong CNN

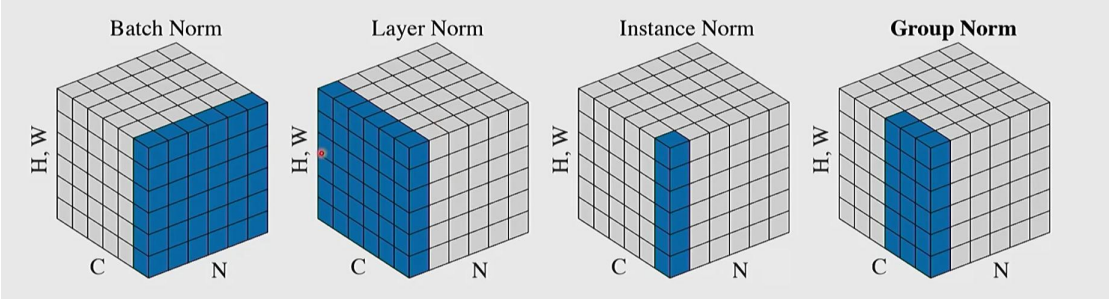
- Convolution layers
- Pooling layers
- Fully-Connected layer
- Normalization layer
- Dropout (thỉnh thoảng xài)
- Activation function (non - linear)

2. Normalization layer

Có nhiều dạng normalization layer và chúng thường để scale hoặc shift data bằng cách học các parameter cụ thể 2 bước sau:

- Normalize data input: Chuẩn hoá dữ liệu (thường làm cho mean ≈ 0, std ≈ 1 theo trục mà layer đó quy định)
- Scale/Shift using learned parameter: Sau khi chuẩn hoá xong, mạng sẽ dùng tham số học được:γ (gamma) để scale (nhân lên/thu nhỏ) và β (beta) để shift (cộng vào/dịch lên xuống)

Điểm khác nhau giữa các normalize là cách chọn data apply và cách tính statistic



C là số channel và N là data trong 1 batch

Vd: ở batch norm t đang tính normalize cho 1 channel của N data hoặc ở layer norm là đang tính cho C channel cho 1 data trong batch

3. Regularization: Drop out

Regularization là một phương pháp giúp model tránh bi overfitting -> để tránh overfitting trên CNN thì ta đơn giản tắt một vài neurons đi khi mỗi batch foward pass theo tỉ lệ dropping (thường là 0.5 hoặc 0.25)

Thường dropout dùng nhiều ở fully-connected/MLP

Tóm lại, khi dùng dropout, mỗi lần train đang huấn luyện một mạng con khác nhau, và tổng thể giống như đang train rất nhiều model, nhưng tất cả cùng dùng chung một bộ weight

Khi test thì không tắt neurons nào cả. Tuy nhiên phải scale lại vì train bị tắt bớt neuron nên output trung bình nhỏ đi, test bật hết nên phải scale để output test ≈ output trung bình lúc train.

4. Activation function

Phần này đã nói ở lecture trước -> file này chỉ nói nhược và ưu điểm và các activation được áp dụng

4.1. Sigmoid

Ưu điểm:Có ý nghĩa như xác suất → hay dùng ở output cho binary classification

Nhược điểm: Vanishing gradient khi x rất dương hoặc rất âm, sigmoid bão hòa (gần 1 hoặc gần 0) -> đạo hàm gần 0 -> gradient qua nhiều lớp nhỏ dần

-> chỉ nên xài cho output không nên dùng cho hidden layers

4.2. ReLU

Thường được hay xài cho CNN cổ điển

Ưu điểm: không bão hòa ở vùng dương, tính toán nhanh và rẻ, giúp hội tụ nhanh hơn sigmoid

Nhược điểm: giá trị mà dưới 0 thì chết nhanh hơn sigmoid

4.3. GELU hay gặp ở Transformer/ViT

Ưu điểm: giá trị quanh 0 (smooth gating): nhỏ thì cho qua ít, lớn thì cho qua nhiều

Nhược điểm: negative quá lớn thì gradient = 0

5. Case study

5.1. Case Study: VGGNet

Ý tưởng: Small filters, deeper networks: thay vì dùng kernel lớn (11×11, 7×7…), VGG chủ yếu dùng 3×3 conv và xếp chồng rất nhiều layer để tăng độ sâu

Tại sao dùng filter nhỏ (3×3)?

+ Receptive field tương đương

+ Sâu hơn -> nhiều phi tuyến hơn

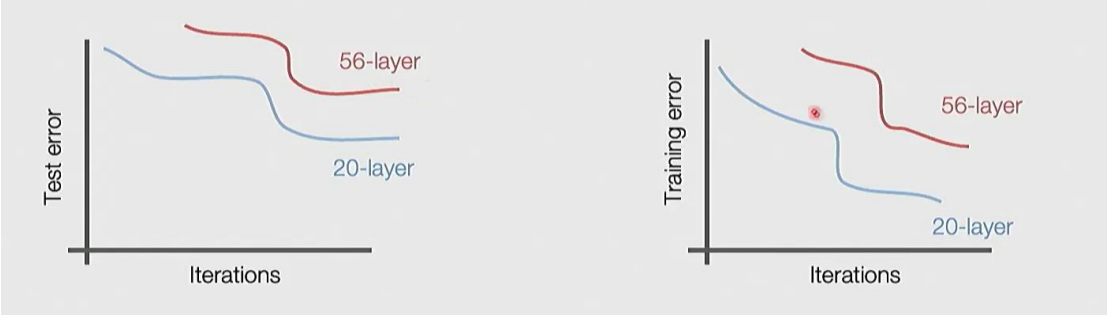
+ Ít tham số hơn

Vd: Giả sử mỗi layer có C kênh vào/ra để so sánh 1 conv 7×7: $7^2C^2=49C^2$ trong khi 3 conv 3×3: $3(3^2C^2)= 27C^2$

Tuy nhiên, VGG khá nặng compute/memory vì rất nhiều layer + FC lớn

5.2. Case Study: ResNet

Khi tiếp tục stack nhiều layer trên một CNN "plain" (không skip connection), mô hình 56-layer lại tệ hơn cả training error lẫn test error so với mô hình 20-layer.

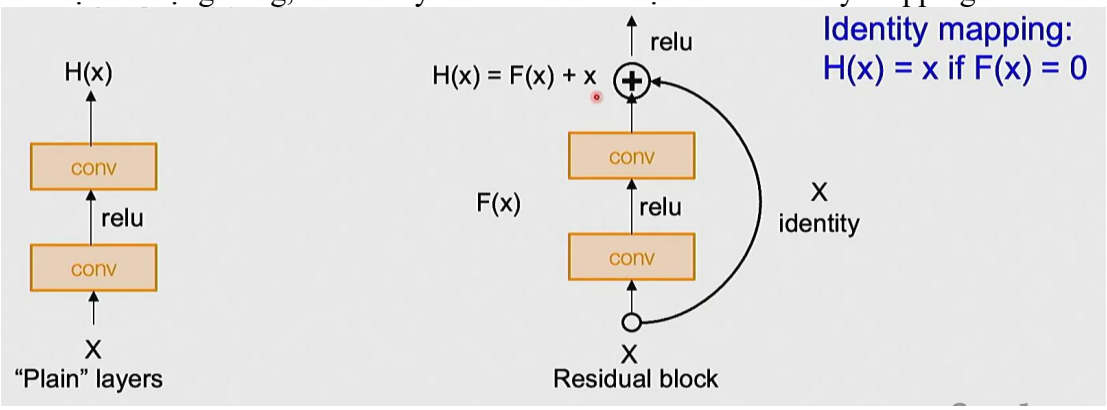


-> lý do cũng không phải do overfitting

Fact: mạng sâu có nhiều tham số hơn -> về lý thuyết representation power phải mạnh hơn.

Hypothesis: mạng sâu khó optimize hơn (khó học được nghiệm tốt).

Slide đặt câu hỏi: "Mạng sâu cần học gì để ít nhất bằng mạng nông?" -> copy các layer đã học từ mạng nông, và các layer thêm vào thì đặt thành identity mapping.



6. Weight initialization

Weight initialization là bước gán giá trị ban đầu cho các tham số của mạng (chủ yếu là weights WWW và bias bbb) trước khi bắt đầu train.

Nếu init "quá nhỏ" → activations tắt dần về 0

Nếu init "quá lớn" → activations nổ (blow up)

Có thể xài ReLU correction: $std = \sqrt{\frac{2}{D_{in}}}$

Với D_{in} là số input vào 1 neuron

7. Data Preprocessing

+ Trừ mean của kênh đó

+ Chia cho std của kênh đó

Lý do

+ Đưa dữ liệu về cùng thang đo -> train ổn định hơn

+ Giảm chuyện một kênh có giá trị lớn hơn hẳn -> gradient lệch

-> Hầu hết model xài cách này

8. Data Augmentation

Mỗi lần lấy 1 ảnh train, bạn **transform ảnh** rồi mới đưa vào CNN để tính loss. Như vậy từ 1 ảnh gốc, model thấy rất nhiều "phiên bản khác nhau" -> giống như dataset lớn hơn

+ Horizontal / Vertical flip: tùy những vật (vd con mèo thì có thể horizontal vì có dạng như vậy còn vertical thì ko nên vì ít có ảnh con mèo bị lật ngược)
+ Random crops + Resize: có Training
+ Color jitter: thường tăng sáng/ giảm sáng
+ Cutout: chọn vùng ngẫu nhiên trên ảnh và set về 0

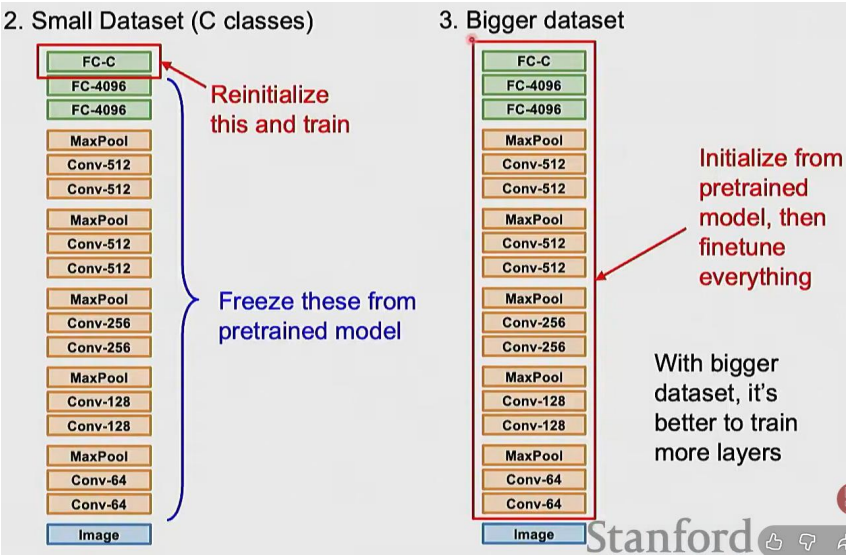
9. Transfer Learning

Transfer learning là cách lấy một model đã được train trước (pretrained) trên dataset lớn (thường ImageNet) để:

+ dùng làm feature extractor (trích đặc trưng)
+ fine-tune (tinh chỉnh) cho dataset khác

Có 2 kiểu transfer learning phổ biến:

+ Dataset nhỏ: chỉ thay layer cuối để phân loại, giữ lại backbone(ko đổi kể cả lúc train cho layer cuối)
+ Dataset lớn: fintune và train thêm với backbone



10. Hyperparameter Selection

Bước 1: Kiểm tra loss đầu tiên(cũng là kiểm tra coi code có bug không)

Bước 2: Overfit một sample nhỏ (Nếu không overfit nổi → thường là bug->có thể pipeline, normalize,… sai hoặc là chọn sai model)
Bước 3: Tìm Learning Rate (LR) làm loss giảm nhanh (để biết LR đó có ổn không)
Bước 4: Coarse grid: quét thô hyperparams, train 1–5 epochs -> để tìm vùng hyperparameter(LR, weight decay, dropout …) phù hợp với dataset
Bước 5: Refine grid, train lâu hơn -> lọc ra đươc khoảng hyperprameter tốt sau bước 4 thì train
Bước 6: Nhìn curve train/val để chẩn đoán
Bước 7: Quay lại bước 5 cho tới khi model tốt