

Ghi chép bài giảng: Stanford CS231n

Lecture 6: CNN Architectures & Training Strategies

Ho Hong Phuc Nguyen

Ngày 12 tháng 2 năm 2026

Mục lục

1	Normalization Layers (Các lớp chuẩn hóa)	3
1.1	Cơ chế chung	3
2	Dropout (Kỹ thuật điều chỉnh)	3
2.1	Cơ chế hoạt động	3
2.2	Tại sao Dropout hiệu quả?	3
2.3	Vấn đề về tỷ lệ (Scaling)	3
3	Activation Functions (Hàm kích hoạt)	4
4	CNN Architectures: Case Studies	4
4.1	VGGNet	4
4.1.1	Tại sao lại dùng chồng các lớp 3x3?	4
4.2	ResNet	4
4.2.1	Giải pháp: Residual Learning (Học phần dư)	5
5	Weight Initialization (Khởi tạo trọng số)	5
5.1	Kaiming Initialization	5
6	Training Strategies & Data Augmentation	6
6.1	Data Augmentation (Tăng cường dữ liệu)	6
6.1.1	Các kỹ thuật cơ bản	6
6.1.2	ResNet's Scale Augmentation (Quy trình Resize & Crop 3 bước)	6
6.2	Test-Time Augmentation (TTA - Tăng cường lúc kiểm thử)	6
6.3	Transfer Learning (Học chuyển giao)	7
7	Hyperparameter Selection (Lựa chọn siêu tham số)	7

1 Normalization Layers (Các lớp chuẩn hóa)

Để mạng nơ-ron hội tụ nhanh và ổn định, việc chuẩn hóa dữ liệu đầu vào (Input) là chưa đủ. Chúng ta cần chuẩn hóa cả các đặc trưng (features) bên trong mạng.

1.1 Cơ chế chung

Hầu hết các lớp chuẩn hóa đều tuân theo quy trình 2 bước:

1. **Normalize:** Dựa dữ liệu về phân phối chuẩn tắc (mean = 0, std = 1) dựa trên thống kê μ và σ được tính toán từ dữ liệu hiện tại.

$$\hat{x} = \frac{x - \mu}{\sigma + \epsilon} \quad (1)$$

2. **Scale and Shift:** Áp dụng biến đổi tuyến tính với các tham số học được (learnable parameters) là γ (scale) và β (shift) để mạng có thể khôi phục lại đặc tính phân phối nếu cần thiết.

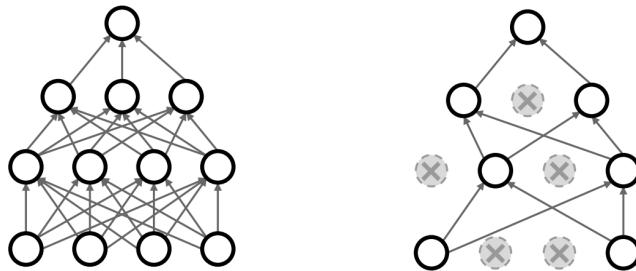
$$y = \gamma \hat{x} + \beta \quad (2)$$

2 Dropout (Kỹ thuật điều chỉnh)

2.1 Cơ chế hoạt động

Dropout là một lớp "Random" được chèn vào mạng để chống Overfitting (quá khớp).

- **Trong lúc Train:** Với mỗi lượt truyền tới (forward pass), mỗi nơ-ron sẽ bị tắt (set về 0) với xác suất p (thường là 0.5).
- **Trong lúc Test:** Sử dụng tất cả các nơ-ron (không tắt cái nào).



2.2 Tại sao Dropout hiệu quả?

1. **Tránh sự đồng thích nghi (Co-adaptation):** Mạng không thể dựa dẫm vào bất kỳ một đặc trưng cụ thể nào (ví dụ: chỉ nhìn vào "tai" để đoán "mèo") vì đặc trưng đó có thể bị tắt bất cứ lúc nào. Nó buộc mạng phải học các đặc trưng phân tán và dư thừa (redundant representations).
2. **Hiệu ứng Ensemble (Tổ hợp mô hình):** Mỗi lần dropout tạo ra một mạng con (sub-network) khác nhau. Quá trình train có thể được xem là đang huấn luyện song song 2^n mạng con chia sẻ trọng số.

2.3 Vấn đề về tỷ lệ (Scaling)

Nếu khi train ta tắt 50% nơ-ron, thì đầu ra $E[x]$ sẽ nhỏ hơn một nửa so với khi test (lúc bật 100%).

- **Cách cũ:** Nhân đầu ra với p lúc Test. (Phức tạp khi triển khai).
- **Inverted Dropout (Chuẩn hiện đại):** Chia cho p ngay lúc Train.

$$\text{Train: } x = \frac{x}{\text{prob}} \times \text{mask} \quad (3)$$

Điều này giúp giữ nguyên biên độ tín hiệu, lúc Test không cần làm gì thêm.

3 Activation Functions (Hàm kích hoạt)

Bài giảng nhấn mạnh sự chuyển dịch lịch sử:

- **Sigmoid:** Đã lỗi thời. Gặp vấn đề **Vanishing Gradient** (biến mất đạo hàm) ở hai đầu cực trị (khi slope ≈ 0).
- **ReLU:** Tiêu chuẩn hiện tại. Tính toán nhanh, không bị bão hòa ở miền dương. Nhược điểm: Dead ReLU ở miền âm.
- **GELU (Gaussian Error Linear Unit):** Biến thể mượt hơn của ReLU. Được dùng chủ yếu trong Transformers (như BERT, GPT) và các CNN hiện đại (ConvNeXt). Có hình dạng cong nhẹ ở gần gốc tọa độ thay vì gấp khúc như ReLU.

4 CNN Architectures: Case Studies

4.1 VGGNet

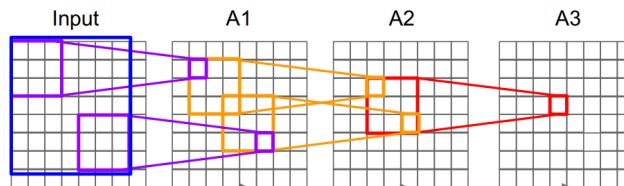
VGGNet đánh dấu bước ngoặt khi từ bỏ các filter kích thước lớn (11×11 , 7×7 trong AlexNet) để chỉ sử dụng toàn bộ filter kích thước nhỏ **3x3**.

4.1.1 Tại sao lại dùng chòng các lớp 3x3?

Tại sao VGG dùng 3 lớp Conv 3x3 thay vì 1 lớp Conv 7x7?

1. Receptive Field (Vùng cảm nhận) tương đương:

- 1 lớp 3x3 nhìn thấy vùng 3x3.
 - 2 lớp 3x3 chồng lên nhau nhìn thấy vùng 5x5.
 - 3 lớp 3x3 chồng lên nhau nhìn thấy vùng 7x7.
- \Rightarrow *Hiệu quả không gian là như nhau.*



2. **Tăng tính phi tuyến (More Non-linearity):** 3 lớp Conv nghĩa là có 3 lần đi qua hàm kích hoạt ReLU. Điều này giúp mạng học được các hàm phức tạp hơn nhiều so với 1 lớp tuyến tính duy nhất.

3. **Giảm số lượng tham số (Fewer Parameters):** Giả sử input và output đều có C kênh.

- 1 lớp 7×7 : $7 \times 7 \times C \times C = 49C^2$ tham số.
 - 3 lớp 3x3: $3 \times (3 \times 3 \times C \times C) = 27C^2$ tham số.
- \Rightarrow *Tiết kiệm được gần 45% tham số trong khi khả năng biểu diễn lại tốt hơn.*

4.2 ResNet

Trước ResNet, người ta tin rằng càng sâu càng tốt. Nhưng thực tế:

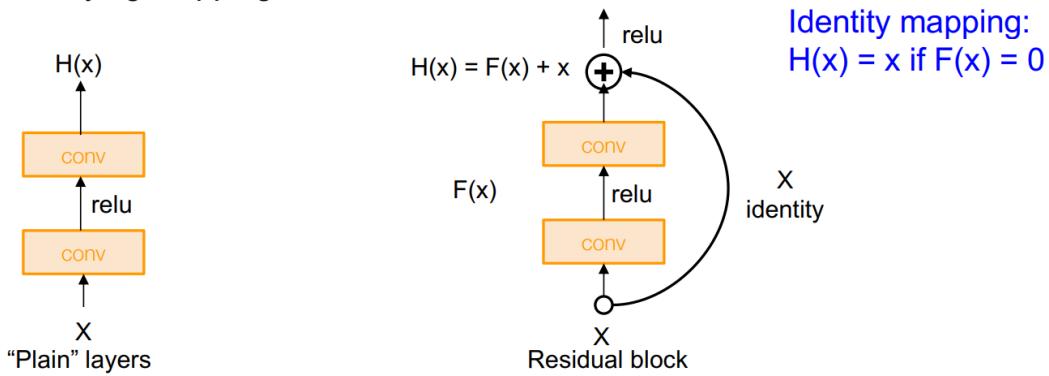
- **Vấn đề:** Mạng 56 lớp có lỗi (error) cao hơn mạng 20 lớp trên cả tập Test **VÀ** tập Train.
- **Kết luận:** Đây **không phải là Overfitting** (nếu overfitting thì train error phải thấp). Đây là vấn đề về **Optimization** (Tối ưu hóa). Mạng quá sâu khiến gradient khó truyền ngược về các lớp đầu, việc học hàm đồng nhất (identity mapping) trở nên khó khăn.

4.2.1 Giải pháp: Residual Learning (Học phần dư)

Thay vì cố gắng học trực tiếp hàm ánh xạ $H(x)$, ResNet thiết kế các block để học phần dư $F(x)$:

$$H(x) = F(x) + x \quad (4)$$

- x : Identity connection (kết nối tắt/đường tắt).
- $F(x)$: Các lớp Conv học sự sai khác (residual).

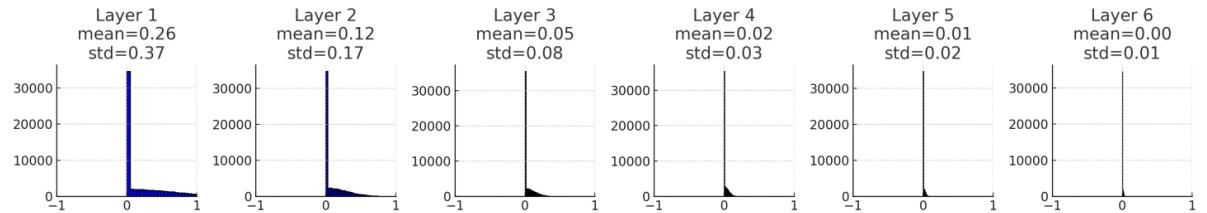


Tại sao nó hiệu quả? Nếu mạng muốn học hàm đồng nhất (Identity), nó chỉ cần đẩy trọng số của $F(x)$ về 0 (điều này dễ hơn nhiều so với việc khớp $H(x)$ thành Identity matrix). Gradient có thể "chạy thẳng" qua đường tắt x về các lớp trước mà không bị suy giảm (Gradient Highway).

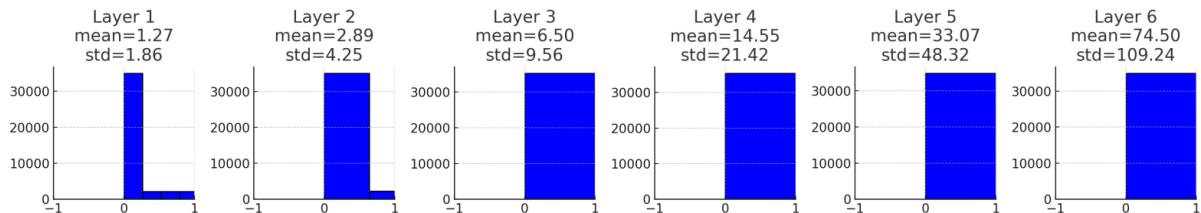
5 Weight Initialization (Khởi tạo trọng số)

Việc khởi tạo W ban đầu cực kỳ quan trọng.

- **Quá nhỏ (ví dụ 0.01):** Tín hiệu qua nhiều lớp sẽ tiến về 0 → Gradient biến mất → Không học được.



- **Quá lớn:** Tín hiệu phát nổ (Exploding) hoặc bị bão hòa (với Sigmoid/Tanh) → Gradient bằng 0 → Không học được.



5.1 Kaiming Initialization

Được thiết kế riêng cho hàm kích hoạt **ReLU** (do Kaiming He - tác giả ResNet đề xuất). Công thức khởi tạo dựa trên kích thước đầu vào D_{in} :

$$W \sim \mathcal{N}(0, \sqrt{\frac{2}{D_{in}}}) \quad (5)$$

Lý do có số 2: Vì ReLU loại bỏ một nửa tín hiệu (phần âm), nên ta cần nhân đôi phương sai (variance) để giữ nguyên cường độ tín hiệu qua các lớp.

6 Training Strategies & Data Augmentation

6.1 Data Augmentation (Tăng cường dữ liệu)

Biến đổi ảnh đầu vào để tạo ra dữ liệu mới nhưng vẫn giữ nguyên nhãn (label). Giúp mô hình khai quát hóa tốt hơn (Generalization).

6.1.1 Các kỹ thuật cơ bản

- **Horizontal Flips:** Lật ảnh theo chiều ngang (với xác suất 0.5).
- **Color Jitter:** Thay đổi ngẫu nhiên độ sáng (brightness), độ tương phản (contrast), và màu sắc (saturation).
- **CutOut:** Ngẫu nhiên chọn một vùng hình vuông trong ảnh và tô đen (masking). Giả lập tình huống vật thể bị che khuất (occlusion).

6.1.2 ResNet's Scale Augmentation (Quy trình Resize & Crop 3 bước)

Thay vì chỉ resize ảnh về một kích thước cố định rồi crop, các mạng hiện đại như ResNet sử dụng kỹ thuật **Scale Jittering** để giúp mô hình nhận diện vật thể ở nhiều kích thước khác nhau (Scale Invariance). Quy trình xử lý mỗi ảnh trong một Batch như sau:

1. Bước 1: Chọn kích thước ngẫu nhiên (Random Scale Selection)

- Chọn ngẫu nhiên một giá trị L trong khoảng [256, 480].
- Đây là kích thước mục tiêu cho cạnh ngắn nhất của ảnh.

2. Bước 2: Thay đổi kích thước ảnh (Resize Short Edge)

- Resize ảnh gốc sao cho cạnh ngắn nhất (shorter edge) bằng L .
- Cạnh còn lại được resize theo tỷ lệ tương ứng (aspect ratio preservation) để ảnh không bị méo.
- *Ví dụ:* Ảnh gốc 800×600 . Nếu chọn $L = 256$, ảnh sẽ được resize về $\approx 341 \times 256$.

3. Bước 3: Cắt ngẫu nhiên (Random Crop)

- Từ ảnh đã resize ở Bước 2, cắt ngẫu nhiên một vùng có kích thước cố định đầu vào của mạng (thường là 224×224).
- Kết hợp thêm lật ngang (Horizontal Flip) ngẫu nhiên.

Tại sao phải làm vậy? Cách này giúp mô hình "nhìn thấy" vật thể ở nhiều tỷ lệ khác nhau. Lúc thì vật thể rất to (khi L nhỏ), lúc thì vật thể nhỏ (khi L lớn). Điều này tốt hơn nhiều so với việc luôn resize về 256×256 rồi crop.

6.2 Test-Time Augmentation (TTA - Tăng cường lúc kiểm thử)

Trái ngược với Training (cần ngẫu nhiên), quá trình Testing cần sự ổn định (Deterministic). Tuy nhiên, để tối đa hóa độ chính xác (thường tăng thêm 1-2%), ta áp dụng TTA: đưa nhiều phiên bản của cùng một ảnh vào mạng và lấy trung bình kết quả.

• Standard 10-Crop Testing:

- Từ một ảnh, ta tạo ra 5 vùng crop: 4 góc + 1 trung tâm.
- Lật ngang 5 vùng đó → Tổng cộng 10 ảnh.
- Dưa cả 10 ảnh qua mạng, tính trung bình vector xác suất (softmax probability) để ra kết quả cuối cùng.

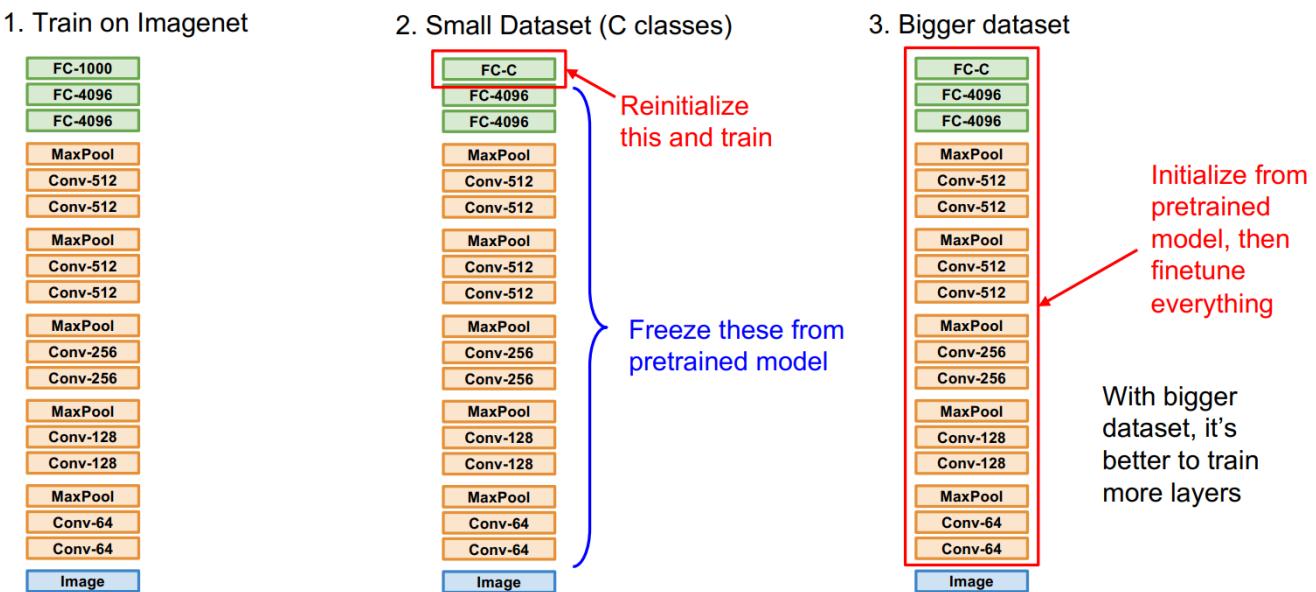
• Multi-scale Testing (Cách của ResNet):

- Không chỉ crop, ta còn resize ảnh gốc về nhiều kích thước khác nhau (ví dụ: {224, 256, 384, 480, 640}).

- Với mỗi kích thước, thực hiện 10-Crop hoặc trượt cửa sổ (sliding window) qua ảnh.
 - Lấy trung bình tất cả các dự đoán.
 - *Nhược điểm:* Tốn chi phí tính toán gấp nhiều lần (thường chỉ dùng trong các cuộc thi như Kaggle hoặc ImageNet Challenge để đua top, ít dùng trong ứng dụng thực tế thời gian thực).

6.3 Transfer Learning (Học chuyển giao)

Câu hỏi: "Không đủ data thì có train CNN được không?" Đáp án: Có, và rất hiệu quả. Chiến lược phụ thuộc vào lượng dữ liệu bạn có và sự tương đồng với dữ liệu gốc (thường là ImageNet).



Dữ liệu của bạn	Tương đồng ImageNet	Khác biệt ImageNet
Ít	Dùng Linear Classifier trên nền CNN đã đóng băng (Feature Extractor)	Thử Linear Classifier ở các lớp sớm hơn (do đặc trưng mức cao quá khác biệt)
Nhiều	Fine-tune toàn bộ mạng (khởi tạo từ pre-trained)	Fine-tune toàn bộ mạng hoặc Train from scratch

Bảng 1: Chiến lược Transfer Learning

7 Hyperparameter Selection (Lựa chọn siêu tham số)

Quy trình debug và tìm kiếm siêu tham số được thực hiện theo 6 bước tuần tự:

1. Step 1: Check Initial Loss (Kiểm tra Loss khởi tạo).

- Tắt regularization (đặt $\text{reg} = 0$).
 - Tính Loss ngay tại vòng lặp đầu tiên.
 - **Ví dụ:** Với Softmax classifier có C lớp, Loss ban đầu phải xấp xỉ $\ln(C)$ (ví dụ: 10 lớp thì loss ≈ 2.3). Nếu khác biệt quá lớn \rightarrow Code khởi tạo bị lỗi.

2. Step 2: Overfit a Small Sample (Cố tình làm Overfit).

- Lấy một tập dữ liệu cực nhỏ (ví dụ: 20 ảnh).
 - Tắt regularization, sử dụng kiến trúc mạng chuẩn.
 - Train thử: Loss phải giảm về 0 và Accuracy phải đạt 100% (trên tập train đó).

- Nếu không overfit được dữ liệu nhỏ này → Lỗi trong code (Backprop, Update rule) hoặc Learning Rate quá tê.

3. Step 3: Find a Learning Rate that makes Loss drop (Tìm LR để Loss bắt đầu giảm).

- Sử dụng toàn bộ dữ liệu training.
- Thử các Learning Rate (LR) khác nhau để tìm ra giá trị làm cho Loss giảm xuống nhanh chóng (trong vòng vài trăm vòng lặp đầu).
- Nếu Loss không đổi → LR quá thấp. Nếu Loss nổ (NaN/Infinity) → LR quá cao.

4. Step 4: Coarse Search (Tìm thô trên phạm vi rộng).

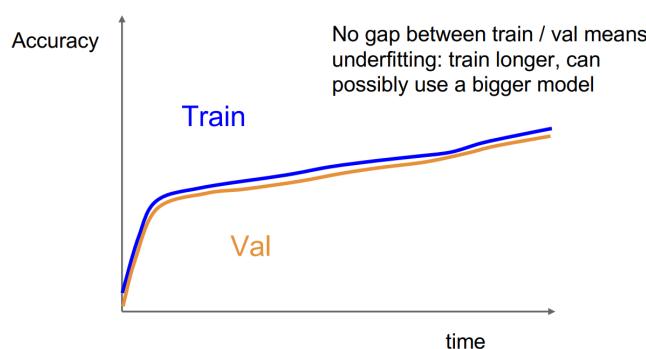
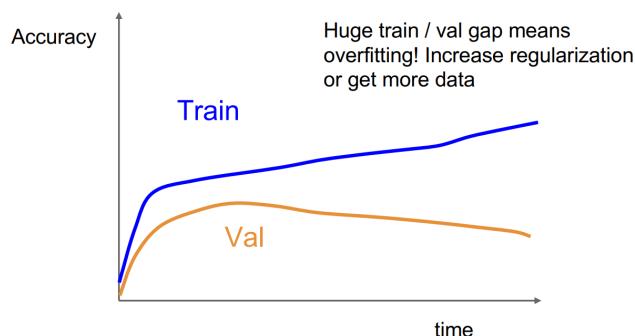
- Chọn các giá trị Hyperparameter (LR, Reg) cách xa nhau theo thang log (ví dụ: $10^{-1}, 10^{-2}, 10^{-3}, \dots$).
- Train trong thời gian ngắn (khoảng 1 - 5 epochs) để xem xu hướng giảm của Loss.
- Mục đích: Xác định khoảng giá trị tiềm năng, loại bỏ các vùng giá trị không hiệu quả.

5. Step 5: Fine Search (Tìm tinh & Thu hẹp phạm vi).

- Thu hẹp phạm vi tìm kiếm dựa trên kết quả tốt nhất ở Step 4.
- Train lâu hơn (Train longer) để đảm bảo mô hình hội tụ, tránh kết luận vội vàng khi Loss chưa ổn định.

6. Step 6: Check Loss & Accuracy (Kiểm tra kết quả cuối cùng).

- Vẽ đồ thị Loss và Accuracy trên cả tập Train và Val.
- Nếu khoảng cách giữa Train Acc và Val Acc quá lớn → Overfitting (cần tăng Regularization).
- Nếu cả hai đều thấp → Underfitting (cần model to lớn hơn hoặc train lâu hơn).
- Chọn bộ tham số có Val Accuracy cao nhất.



Lưu ý quan trọng: Khi thực hiện Step 4 và Step 5, nên sử dụng **Random Search** (lấy mẫu ngẫu nhiên trong khoảng log) thay vì Grid Search. Random Search giúp quét không gian tham số hiệu quả hơn, đặc biệt khi độ quan trọng của các siêu tham số là không đồng đều.

Random Search vs. Grid Search

*Random Search for Hyper-Parameter Optimization
Bergstra and Bengio, 2012*

