SÂU HỌC THUẬT TOÁN

Báo cáo

Kiến trúc U-Net

Lĩnh vực: Computer Vision

Instructor : **Hồ Nguyễn Phú**

Student 1 : **Hồ Nguyễn Phú**

---o0o---

Mục lục

UNet – Segmentation

Hồ Nguyễn Phú

30 tháng 10, 2025

I. Ôn lại về mạng CNN:

Dữ liệu hình ảnh – khác với dữ liệu có cấu trúc dạng bảng, hay dữ liệu dạng chuỗi, không thể được xử lý trực tiếp thông qua các lớp Fully Connected (còn gọi là Dense Layer, Linear, hay cả mạng gọi là Multilayer Perceptron etc). Nguyên nhân:

+ Fully connected yêu cầu làm phẳng ảnh (Flatten), khiến các đặc trưng không gian bị mất đi.
+ Hai tính chất quan trọng của hình ảnh không được xử lý thông qua Multilayer Perceptron: spatial locality và translation invariance (tính cục bộ không gian và bất biến dịch chuyển; nguồn).
+ Fully connected với thông số $nn.Linear(3,\ 64)$ về cơ bản là một ma trận 3×64, tham số nhiều, không thể "tái sử dụng" và số lượng các phép tính toán rất lớn.

Để giải quyết vấn đề trên, kiến trúc CNN ra đời – với lớp tích chập (convolution).

1. Lịch sử phát triển mạng:

a) Giai đoạn thai nghén ý tưởng: (nguồn)

- **[1998] – LeNet-5**: Yann Lecun công bố mạng LeNet–5 cho tác vụ phân loại chữ số viết tay (MNIST). Mạng CNN vẫn chưa bùng nổ do hạn chế về tính toán và dữ liệu.
- **[2009] – ImageNet**: Bộ dữ liệu ImageNet được giới thiệu (1 triệu tấm ảnh với 1000 class).

b) Điểm đột phá: từ AlexNet đến ResNet

- **[2012] – AlexNet**: Krizhevsky và đồng sự công bố mạng AlexNet, vượt xa về độ chính xác so với các phương pháp Computer Vision truyền thống.
- **[2014] – VGG**: Sử dụng các bộ lọc nhỏ hơn (3×3), tăng độ sâu.
- **[2014] – GoogLeNet (Inception v1)**: Sử dụng khối Inception (kết hợp các kích thước bộ lọc khác nhau: 1×1 , 3×3 và 5×5).

- **[2015] – ResNet**: Đề xuất khối Residual và khối Bottleneck – tận dụng skip connection để giải quyết vanishing gradient (tác giả gọi là *degradation*).

c) Các mô hình sau đó:

- **[Feb 2016] – SqueezeNet**: Giảm số lượng tham số rất nhiều (dưới 0.5MB; thông qua kiến trúc và quantization), vẫn giữ accuracy cạnh tranh với AlexNet.
- **[Aug 2016] – DenseNet**: Mỗi layer kết nối tới tất cả layer trước nó bằng concat, giúp tái sử dụng feature, gradient flow tốt hơn.
- **[Nov 2016] – ResNext**: Giữ cấu trúc skip connection, nhưng thêm ***cardinality*** – số lượng đường nhánh song song trong mỗi khối – để tăng khả năng biểu diễn theo chiều rộng thay vì chiều sâu.
- **[Apr 2017] – MobileNet (V1)**: Nhẹ hơn, sử dụng depth-wise separable convolution (3×3 depth–wise từng kênh, sau đó 1×1 point–wise trộn các kênh) để giảm tính toán & parameters.
- **[Jul 2017] – ShuffleNet (v1)**: Sử dụng pointwise kết hợp shuffle channel để trộn thông tin giữa các nhóm trong group convolution (xem lại mối liên hệ với depthwise separable convolution).
- **[2019] – EfficientNet**: Dùng cách scaling kết hợp: depth + width + resolution một cách cân bằng; baseline architecture được tìm bằng neural architecture search.

2. Các thành phần của mạng CNN:

Một mạng CNN cho tác vụ phân loại (classification) thường bao gồm hai phần: feature extractor (backbone) và classifier.

- **Feature extractor (backbone)**: bao gồm các lớp Convolution, Activation, Pooling, và Normalization hoặc Drop–out. Tùy vào kiến trúc, ở cuối backbone sẽ có lớp Average Pooling hoặc Flatten.

  Spatial locality được xử lý tốt thông qua lớp Convolution; trong khi translation invariance được giải quyết bởi lớp Pooling.

- **Classifier**: các lớp Fully Connected.

3. Tính chất lớp tích chập:

Theo sách Deep Learning (nguồn), mạng CNN tận dụng các điểm mạnh sau: tương tác thưa (sparse interaction), chia sẻ tham số (parameter sharing) và equivariance to translation (phân biệt với translation invariant).

Nguồn đọc thêm: Springer Open; Spiderum; VNPT AI; Viblo về Scaling Up Your Kernel to 31x31; bài viết Facebook của Khánh Đình Phạm.

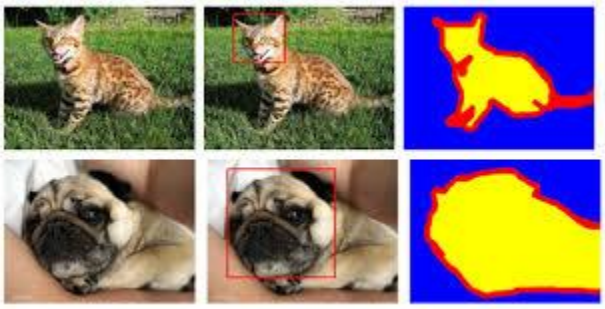
4. **Các siêu tham số một khối CNN:**

Các siêu tham số có thể điều chỉnh trong lớp tích chập (Pytorch) như sau: Kernel size, Padding, Stride, Dilation, Group.

II. Tác vụ segmentation:

Phân đoạn hình ảnh (image segmentation) có thể hiểu nôm na là sự phân loại từng pixel – khác với classification phân loại trên cả bức ảnh, hay object detection thực hiện khoanh vùng một object thông qua bounding box.

Segmentation cho phép xác định vị trí chính xác của đối tượng cũng như ranh giới của chúng. Do đó, thuật toán được ứng dụng rộng rãi trong nhiều lĩnh vực: ảnh y tế, xe tự hành, ảnh vệ tinh, nông nghiệp, cảnh báo cháy (Pham Dinh Khanh).



Hình 1. Object classification, object detection và segmentation (Github).

Dựa vào số lượng và loại thông tin, ta phân thành 3 nhóm tác vụ segmentation: semantic, instance và papoptic.

- ***Semantic***: phân loại các pixel thành những lớp ngữ nghĩa; không có sự tách biệt giữa các instance (đối tượng).
- **Instance**: phân loại các pixel thành các danh mục trên cơ sở các instance với nhãn cụ thể.
- **Panoptic**: kết hợp giữa hai tác vụ trên (ranh giới mỗi đối tượng trong ảnh được tách biệt và danh tính đối tượng được dự đoán).

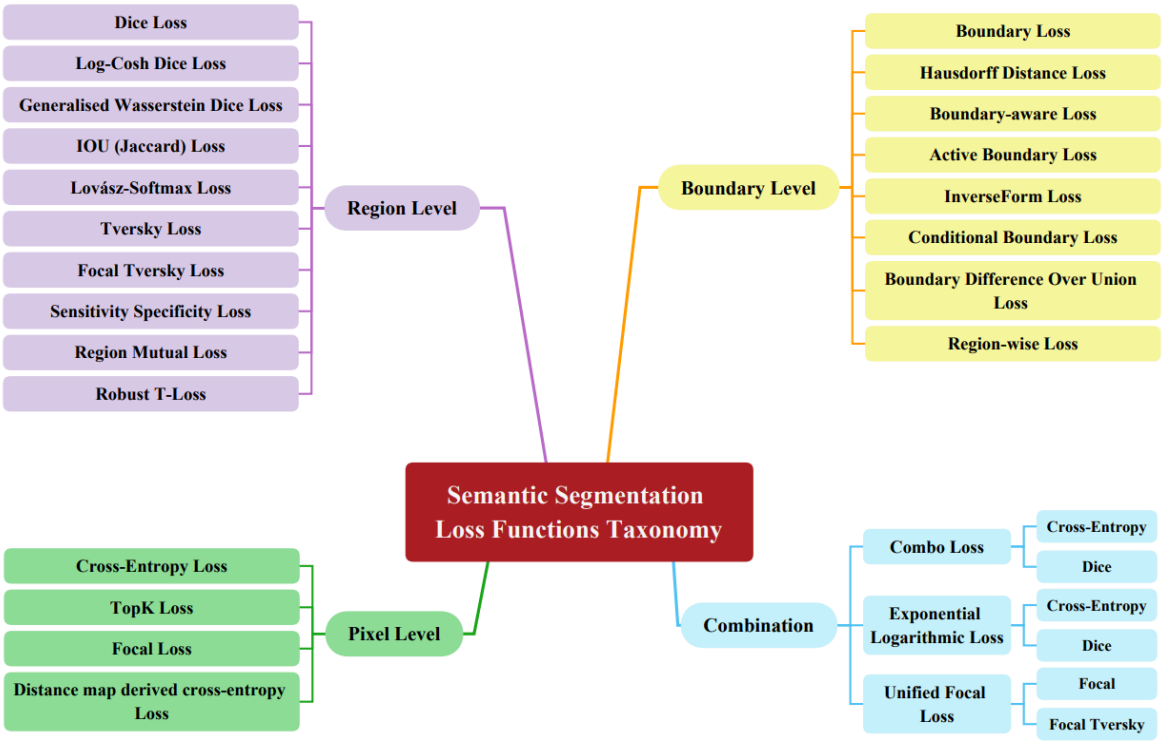
III. Các hàm loss segmentation:

Các paper survey (và Loss) sử dụng làm tài liệu tham khảo:

- Segmentation Loss Odyssey;
- A survey of loss functions for semantic segmentation; (Log–Cosh Dice Loss)
- Loss Functions in the Era of Semantic Segmentation: A Survey and Outlook

Ta phân thành **4 nhóm loss segmentation**:

- **Pixel level (Distributed–based)**: hướng đến độ chính xác *fine grain trên từng pixel*.
- **Boundary level**: hướng đến độ chính xác *ranh giới*.
- **Region level**: hướng đến độ chính xác *toàn cục*.
- **Combination (Compounded-based)**: kết hợp các loại loss trên.



Hình 2. Mindmap các hàm loss chính (Paper).

1. **Pixel level:**

a) **Cross–Entropy Loss:**

Hàm Cross–Entropy Loss được tính như sau:

$$L_{CE} = \sum_{n=1}^{N} \sum_{c=1}^{C} - g_c log(p_c)$$

Trong đó:

- t là vector nhãn từng pixel dưới dạng one–hot encoding.
- $\hat{y}$ là vector các giá trị xác suất dự đoán. Các xác suất $\hat{y}_c$ với c không gắn với nhãn sẽ bằng 0 do one–hot encoding t_c bằng 0 tại mọi vị trí trừ nhãn.

Cross–Entropy Loss chưa xét đến class imbalance (khi background chiếm diện tích lớn hơn nhiều foreground). Người ta sử dụng thêm Weighted Cross–Entropy Loss (trong đó class có số lượng mẫu ít hơn sẽ có weight lớn hơn):

$$L_{WCE} = \sum_{n=1}^{N} [w \times \sum_{c=1}^{C} - g_c log(p_c)]$$

Note – còn một số biến thể của Cross–Entropy nhưng chưa đề cập do:

- Trong paper **A survey of loss functions for semantic segmentation** thì Weighted Cross–Entropy Loss được dùng cho dạng Binary; trong **Loss Functions in the Era of Semantic Segmentation: A Survey and Outlook** thì được dùng dưới dạng tổng quát, nhưng paper thứ hai có vẻ thiếu chính xác ở thuật ngữ và notation toán học.
- Ngoài ra, cách biểu diễn Balanced Cross–Entropy trong paper **A survey of loss functions for semantic segmentation** và bài viết của **Phạm Đình Khánh Blog** lại khá khác nhau (***cần bàn luận thêm***).

b) **TopK Loss:**

Về cơ bản là Cross–Entropy Loss, tuy nhiên chỉ xét đến top k% những pixel với xác suất dự đoán thấp nhất. Thông qua TopK Loss, mô hình chỉ học từ những mẫu khó nhất:

$$L_{TopK}(y, t) = \sum_{n \in K} \sum_{c=1}^{K} - g_c log(p_c)$$

c) Focal Loss: (Phạm Đình Khánh Blog)

Để xử lý tốt hơn các class khó, ta thêm hệ số $\alpha \times (1 - p_t)^{\gamma}$ Cross–Entropy Loss để tạo thành Focal Loss:

$$L_{Focal} = \frac{-1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} [\alpha \times (1 - p_c)^{\gamma} \times g_c log\,(p_c)]$$

Trong đó:

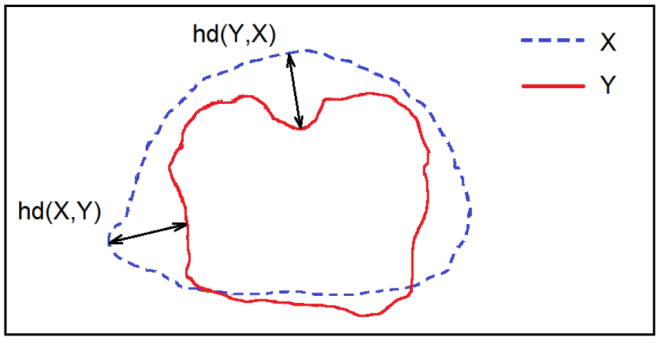
- α : thường là hệ số nghịch đảo của tần suất xuất hiện của class – class xuất hiện càng nhiều thì trọng số càng nhỏ. Tức: $\alpha = \frac{1}{f_c + \epsilon}$ với ϵ để tránh chia cho 0.
- $(1 - p_c)^{\gamma}$ vừa ảnh hưởng đồng thời lên loss và gradient descent theo hướng tập trung vào class khó phân biệt.

2. **Boundary level:** bao gồm Hausdorff Distance, Shape aware loss và Distance map loss penalty.

Note: Paper Distance Map Loss Penalty Term for Semantic Segmentation ghi rằng công trình của họ nhằm hướng tới phân loại đường biên tốt hơn, nhưng paper A survey of loss functions for semantic segmentation lại phân loại Distance map loss penalty thành distribution–based.

a) Hausdorff Distance:

Hausdorff distance (ngữ cảnh Segmentation) là khoảng cách xa nhất có thể giữa 1 điểm thuộc đường biên đối tượng này tới đường biên đối tượng còn lại (biến thể từ định nghĩa trong trang blog của ikemen96).



Hình 3. Minh họa khoảng cách Hausdorff giữa tập con X và tập con Y (Paper).

Do độ phức tạp tính toán, Hausdorff Distance không được tính trực tiếp (naive) mà thường sử dụng các cách xấp xỉ như **distance transform**. Hausdorff Distance có implementation HD95 cũng được sử dụng trong tác vụ phân đoạn ảnh y tế (như paper **High-Resolution Swin Transformer for Automatic Medical Image Segmentation** sử dụng HD95 cùng Dice Score làm metric đánh giá).

Xem thêm: **paper Reducing the Hausdorff Distance in Medical Image Segmentation with Convolutional Neural Networks**; **Trang blog về các loại distance của ikemen96**; và **Metric HD95 cheatsheet**.

b) Shape aware loss:

Trong khi Cross–Entropy tính loss dựa trên từng pixel, Shape aware Loss tính toán khoảng cách Euclid giữa **điểm trên curve của dự đoán** và **ground truth**, và sử dụng nó làm một hệ số của hàm Cross–Entropy Loss.

Công thức tính Shape aware Loss như sau (nguồn từ **A survey of loss functions for semantic segmentation**):

$$E_i = D(\hat{C}, C_{GT})$$

$$L_{shape-aware} = -\sum_i CE(y, \hat{y}) - \sum iE_i CE(y, \hat{y})$$

Xem thêm: **Boundary-aware Instance Segmentation**.

c) Distance map loss penalty:

Distance map loss penalty được ra đời với mục đích phân đoạn chính xác đường biên của vật thể (accurate segmentation of object boundaries). Theo bài báo **Distance Map Loss Penalty Term for Semantic Segmentation**, Công thức tính:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N}(1+\Phi) \odot \sum_{j=1}^{K} -y_j \log \hat{y_j}$$

- $\sum_{j=1}^{K} - y_j log\hat{y}_j$: là hàm Cross–Entropy Loss bình thường.
- $(1 + \Phi)$: Distance map (cộng thêm 1 để hạn chế vanishing gradient)
- $\odot$: tích Hadamard – hay tích Element–wise (nhân từng cặp phần tử tương ứng của hai ma trận).

Xem thêm: **Distance Map Loss Penalty Term for Semantic Segmentation**.

3. Region level:

a) Dice Loss:

Dice coefficient được sử dụng rộng rãi như một metric để tính toán độ giống nhau giữa hai bức ảnh. Vào năm 2017, Dice Loss ra đời để tối ưu trực tiếp chỉ số đánh giá này:

$$L_{Dice} = 1 - 2\frac{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c p_i^c}{\sum_{i=1}^{N}\sum_{c=1}^{C}(g_i^c + p_i^c)}$$

Để ổn định số học ta có thể cộng 1 vào cả tử và mẫu số như sau (tuy nhiên, phương pháp trên chỉ xuất hiện trong paper **A survey of loss functions for semantic segmentation**, không thấy xuất hiện trong paper gốc và cả 2 paper survey còn lại):

$$L_{Dice} = 1 - 2 \times \frac{\sum_{i=1}^{N}\sum_{c=1}^{C} (g_i^c p_i^c) + 1}{\sum_{i=1}^{N}\sum_{c=1}^{C}(g_i^c + p_i^c) + 1}$$

Bài báo Generalized Overlap Measures for Evaluation and Validation in Medical Image Analysis cũng đề xuất hàm Generalized Dice Loss (GDL) cho multi–class, trong đó mỗi class sẽ có một trọng số riêng. Trọng số này được tính dựa trên nghịch đảo của tần suất xuất hiện label, trong đó $w_c = \frac{1}{(\sum_{i=1}^{N} g_i^c)^2}$. Công thức tính:

$$L_{Dice} = 1 - 2\frac{\sum_{c=1}^{C} w_c \sum_{i=1}^{N} g_i^c p_i^c}{\sum_{c=1}^{C} w_c \sum_{i=1}^{N}(g_i^c + p_i^c)}$$

Trong mô hình **V–Net** (Milletari et al., 2016), tác giả sử dụng biến thể hàm Dice Loss sau để xử lý dữ liệu 3D:

$$L_{Dice} = 2 \times \frac{\sum_{i=1}^{N}\sum_{c=1}^{C} g_c p_c}{\sum_{i=1}^{N}\sum_{c=1}^{C} g_c^2 + \sum_{i=1}^{N}\sum_{c=1}^{C} p_c^2}$$

b) IoU Loss (Jaccard Loss):

$$\text{Intersection over Union} = \frac{|Y \cap T|}{|Y \cup T|}$$

IoU Loss có bắt nguồn từ IoU index và được tính theo công thức sau:

$$L_{Tversky} = \frac{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c p_i^c}{\sum_{i=1}^{N}\sum_{c=1}^{C} (g_i^c + p_i^c - g_i^c p_i^c)}$$

c) Tversky Loss:

Tương tự Focal Loss, Tversky Loss cũng bắt nguồn từ chỉ số Tversky index (đánh giá similarity giữa hai tập dữ liệu):

$$\text{Tversky index} = \frac{|Y \cap T|}{|Y \cap T| + \alpha|Y \backslash T| + \beta|Y \backslash T|}$$

α và β là trọng số cho trường hợp False Negative và False Positive.

+ $\alpha = \beta = 0.5 \rightarrow$ Ta có Dice Coefficient
+ $\alpha = \beta = 1 \rightarrow$ Ta có IoU (Jaccard index)

Từ đây, ta có hàm Tversky Loss như sau:

$$L_{Tversky} = \frac{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c p_i^c}{\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c p_i^c + \alpha\sum_{i=1}^{N}\sum_{c=1}^{C} (1-g_i^c) p_i^c + \beta\sum_{i=1}^{N}\sum_{c=1}^{C} g_i^c (1-p_i^c)}$$

d) Focal Tversky Loss:

Hàm Focal Tversky Loss áp dụng ý tưởng của Focal Loss để tập trung vào các case khó với xác suất dự đoán thấp.

$$L_{FTL} = (1 - L_{Tversky})^{\frac{1}{\gamma}}$$

Trong đó, γ trong khoảng $[1, 3]$.

e) Log–Cosh Dice Loss:

Biến thể mượt hơn của Dice Loss. Công thức tính không quá phức tạp:

$$L_{lc-dce} = log(cosh(L_{Dice}))$$

Trong đó, $cosh(x) = \frac{e^{x} - e^{-x}}{2}$.

4. Combination:

Kết hợp nhiều Loss lại với nhau để tận dụng thế mạnh và hạn chế điểm yếu từng hàm Loss riêng lẻ. Một số hàm Loss kết hợp thường thấy:

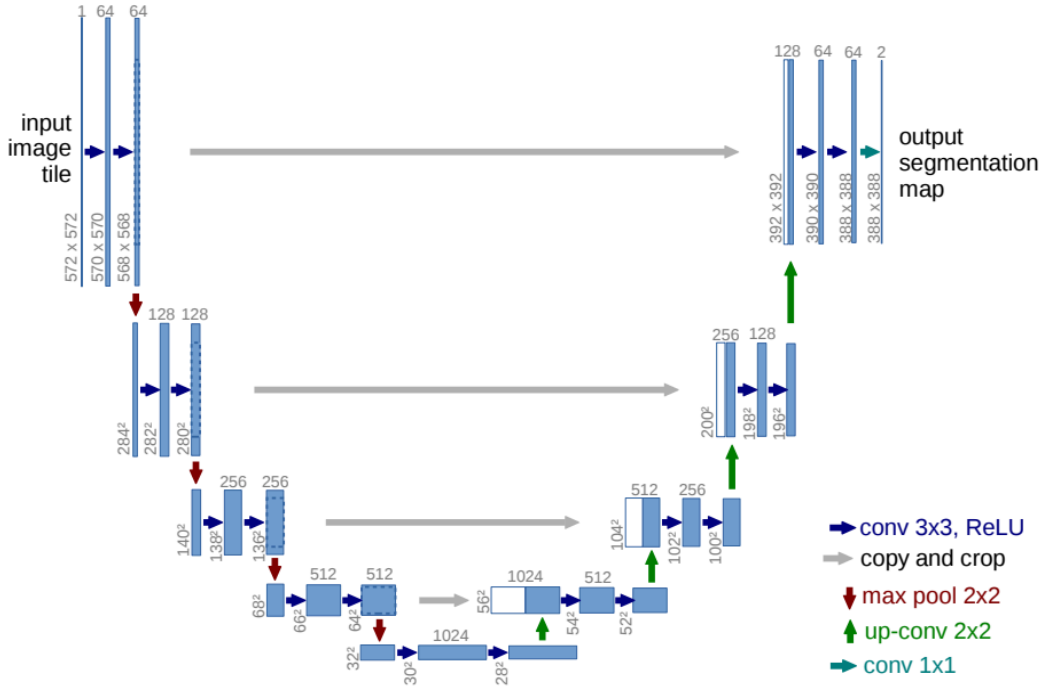
- **Combo Loss** (WCE và Dice Loss);
- **Exponential Logarithmic Loss** (Exponential Log Dice và Exponential Log WCE);
- **Unified Focal Loss** (Focal Loss và Focal Tversky Loss).

5. Xem thêm:

- Một số loại Loss khác: Sensitivity Specificity Loss; Region Mutual Loss; etc.
- Xem Table II của paper **A survey of loss functions for semantic segmentation** (Log–Cosh Dice Loss) để biết use case từng loại loss.
- ***Implementation của Loss***: **SegLossOdyssey** hoặc của paper **Log–Cosh**.

IV. Kiến trúc UNet:

UNet là một kiến trúc CNN (Fully Convolution) được thiết kế cho tác vụ phân đoạn ảnh. Ý tưởng chính của UNet: ***<u>một khối encoder</u>*** tạo ra các bản đồ đặc trưng giàu ngữ nghĩa và ***<u>một khối decoder</u>*** dựa trên các bản đồ đặc trưng đó, kết hợp với thông tin từ ảnh gốc để output một tensor với kích thước không gian (rộng, dài, sâu) như ảnh gốc, nhưng đã được phân đoạn (classify từng pixel).



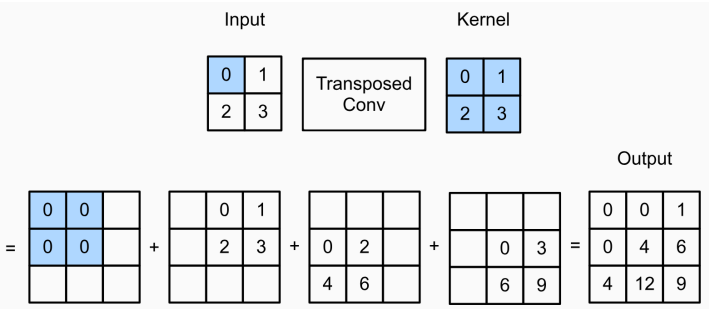
Hình 4. Kiến trúc mạng UNet (<u>Paper UNet</u>).

1. Kiến thức nền:

a) Upsampling:

Trong phần lớn tác vụ computer vision (Classification, Detection), mục tiêu của các lớp Convolution là tạo ra những bản đồ đặc trưng giàu ngữ nghĩa (với kích thước không gian nhỏ nhưng số kênh / channel lớn). Các lớp này gọi là Downsampling. Tuy nhiên, với các tác vụ Image2Image như sinh ảnh, hay segmentation, thì ta cần các lớp giúp chuyển từ đặc trưng giàu ngữ nghĩa thành một ảnh mới (kích thước không gian bằng ảnh gốc, số kênh tùy thuộc tác vụ). Các lớp này gọi là Upsampling – trái ngược với Downsampling (tham khảo **<u>Blog của Phạm Đình Khánh</u>**).

Trong mạng UNet, lớp tích chập chuyển vị (Transposed Convolution) được sử dụng để tăng kích thước ảnh, chuyển từ các bản đồ đặc trưng giàu ngữ nghĩa kết hợp với Skip connection để hình thành map segmentation. Cách lớp tích chập chuyển vị hoạt động như sau:



Hình 5. Tích chập chuyển vị (d2l.ai).

b) Skip connection:

Khi thực hiện Downsampling, ta không tránh khỏi việc mất thông tin. Điều này là do các lớp Convolution và Pooling liên tục làm giảm kích thước không gian và "nén" thông tin lại (e.g., giữ lại những thông tin "nổi trội nhất" với Max Pooling). Do đó, để chuyển từ bản đồ đặc rưng giàu ngữ nghĩa sang ảnh với kích thước gốc, ta cần tận dụng lại thông tin ảnh từ các lớp trước đó, thông qua Skip connection.

2. **Encoder:** convolution + pooling, trích xuất đặc trưng.

Đầu vào khối Encoder (Downsampling) là một ảnh với kích thước $1 \times 572 \times 572$. Thông qua các phép tích chập (Convolution), mạng UNet thực hiện Downsampling:

Block 1:

Không gian: $(572^2) \rightarrow (570^2) \rightarrow (568^2)$

Số kênh: $(1) \rightarrow (64) \rightarrow (64)$

Pooling 1: $(568^2) \rightarrow (284^2)$

Block 2:

Không gian: $(284^2) \rightarrow (282^2) \rightarrow (280^2)$

Số kênh: $(64) \rightarrow (128) \rightarrow (128)$

Pooling 2: $(280^2) \rightarrow (140^2)$

Block 3:

Không gian: $(140^2) \rightarrow (138^2) \rightarrow (136^2)$

Số kênh: $(128) \rightarrow (256) \rightarrow (256)$

Pooling 3: $(136^2) \rightarrow (68^2)$

Block 4:

Không gian: $(68^2) \rightarrow (66^2) \rightarrow (64^2)$

Số kênh: $(256) \rightarrow (512) \rightarrow (512)$

Pooling 4: $(64^2) \rightarrow .(32^2)$

Block 5:

Không gian: $(32^2) \rightarrow (30^2) \rightarrow (28^2)$

Số kênh: $(512) \rightarrow (1024) \rightarrow (1024)$

Hình 6. Thay đổi kích thước qua các lớp trong khối Encoder.

Như vậy, từ đầu vào $1 \times 572 \times 572$ ta nhận được đầu ra $512 \times 28 \times 28$ (tức 512 bản đồ đặc trưng – mỗi bản đồ đặc trưng có kích thước 28×28).

3. **Decoder:** up-convolution + skip connection từ encoder → khôi phục spatial detail

Khối Decoder thực hiện "giải mã" các bản đồ đặc trưng để đưa nó về ảnh với kích thước lớn hơn. Trong mạng UNet, kích thước đầu vào của khối Decoder này là $512 \times 28 \times 28$ và đầu ra là $2 \times 388 \times 388$ (trong đó 2 là số channel / số class). Kích thước đầu ra có giảm một chút so với đầu vào.

Tuy nhiên, khác với Encoder, đầu vào của một khối Decoder (dùng Transposed Convolution) sẽ là concat giữa ***bản đồ đặc trưng được upsample*** và ***bản đồ đặc trưng từ khối Encoder tương ứng*** (có kích thước gần bằng). Ta gọi phương pháp này là Skip connection.

V. Các biến thể kiến trúc UNet:

1. UNet++:

- **Ý tưởng**:
    + Thêm nhiều skip connection phức tạp hơn (Nested Dense Convolutional Block), giúp khớp thông tin encoder–decoder tốt hơn.
    + Tận dụng Deep Supervision (*Xem thêm*).
- **Github**: https://github.com/4uiiurz1/pytorch-nested-unet.
- **Bài báo**: *UNet++: A Nested U-Net Architecture for Medical Image Segmentation*.

2. Attention UNet: thêm attention gate, tập trung vào vùng quan trọng

- **Ý tưởng**:
    + Sử dụng các Attention Gate (Sigmoid) để filter các feature propagated qua Skip connection.
    + Khác với Self–Attention trong Transformers.
- **Github**: https://github.com/ozan-oktay/Attention-Gated-Networks (từ bài báo gốc).
- **Bài báo**: *Attention U-Net: Learning Where to Look for the Pancreas*.

3. Residual UNet:

- **Ý tưởng**:
    + *Sử dụng Recurrent Convolutional Neural Network (RCNN)*: trong đó một bộ lọc tích chập tại mỗi lớp sẽ được dùng lại và xử lý ảnh nhiều lần.
    + Có biến thể sử dụng Residual từ ResNet (để tạo thành RRCNN).
    + Không có block RNN được sử dụng.
- **Github**: https://github.com/navamikairanda/R2U-Net.
- **Bài báo**: *Recurrent Residual Convolutional Neural Network based on U-Net (R2U-Net) for Medical Image Segmentation*.

4. **UNet kết hợp Transformer:**

- **Ý tưởng**:

    Paper 1: Áp dụng *Self–Attention cho khối cuối của Encoder* và *Cross–Attention cho các Skip connection* để lọc bỏ non–semantic features.

    Paper 2: Áp dụng Transformers Encoder sau khối Encoder của UNet; với Decoder thì có Transformers Decoder và vector "organ queries." Xử lý cả ảnh 2D và 3D.

- **Github**: https://github.com/Beckschen/TransUNet (thư viện 3D đã mất tích).
- **Bài báo**:

    Paper 1: *U-Net Transformer: Self and Cross Attention for Medical Image Segmentation*.

    Paper 2: *TransUNet: Rethinking the U-Net architecture design for medical image segmentation through the lens of transformers*.

5. **nnUNet**:

- **Ý tưởng**: Framework auto-config, tự điều chỉnh hyperparameter cho các dataset khác nhau.
- **Bài báo**: *nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation*.
- **Github**: https://github.com/MIC-DKFZ/nnUNet (gốc).

6. **3D UNet**:

- **Ý tưởng**: Mở rộng sang dữ liệu 3D (MRI, CT), thay pooling/conv bằng 3D.
- **Bài báo**: *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*.
- **Github**: https://github.com/wolny/pytorch-3dunet.

VI. Thực hành Pytorch: