Report
TGS Salt Identification Challenge

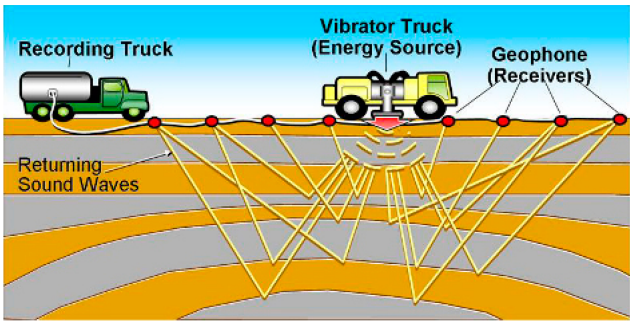
Nguyen Ngoc Hieu
University of Engineering and Technology
18020510@vnu.edu.vn

Nguyen Duc Duong
University of Engineering and Technology
18020386@vnu.edu.vn

Nguyen Duy Long
University of Engineering and Technology
18020790@vnu.edu.vn

1 Giới thiệu

Seismic image - Ảnh địa chất: là ảnh thu được sau khi xử lý tín hiệu sóng âm bị phản xạ bởi các cấu trúc địa chất, sử dụng một thiết bị thu nhận được gọi là geophones.



Hình 1: Mô tả quá trình thu nhận ảnh địa chất (nguồn: www.petroman.co)

Trong khi cho biết thông tin về ranh giới giữa các loại đá, ảnh địa chất không cho biết nhiều về cấu nội tại của chúng; một số loại đá có thể dễ dàng xác định được từ ảnh, một số khó hơn. Về lý thuyết, độ mạnh của phản xạ tỉ lệ thuận và trực tiếp với sự khác nhau về các tính chất vật lý giữa 2 bên đường ranh giới.

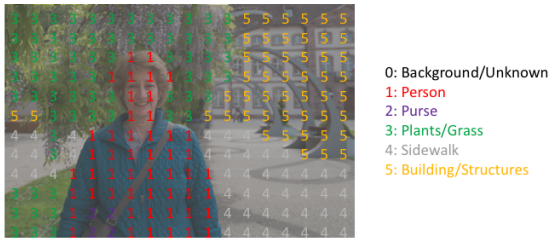
Một trong những yêu cầu khi phân tích ảnh địa chất là phát hiện các vùng chứa muối, việc phát hiện các hồ muối đóng vai trò quan trọng trong quá trình tìm kiếm các nguồn nhiên liệu đốt. Đó cũng là mục tiêu của cuộc thi này, từ một ảnh địa chất kích thước 101x101, nhiệm vụ là tìm ra và phân loại các pixel trong ảnh có thuộc hố muối hay không.

Một số tính chất của muối ảnh hưởng đến ảnh địa chất thu được:

- Khối lượng riêng thường là 2.14g/cc, thấp hơn hầu hết các loại đá xung quanh
- Tốc độ sóng âm khi đi qua lớp muối là 4.5 km/s, thường nhanh hơn các loại đá xung quanh tạo ra các sharp reflection trên bề mặt lớp muối. Vận tốc cao này cũng có thể gây nên một số vấn đề trên ảnh địa chất.
- Thường có tính chất vô định hình, nên muối có ít cấu trúc nội tại, do đó cũng có ít sự phản xạ tròng lòng lớp muối, trừ khi có cặn hay trầm tích.

2 Hướng tiếp cận

Yêu cầu của contest này tương ứng với bài toán semantic image segmentation trong computer vision. Với input là một ảnh, mục tiêu là xác định label tương ứng với mỗi pixel trong ảnh đó.

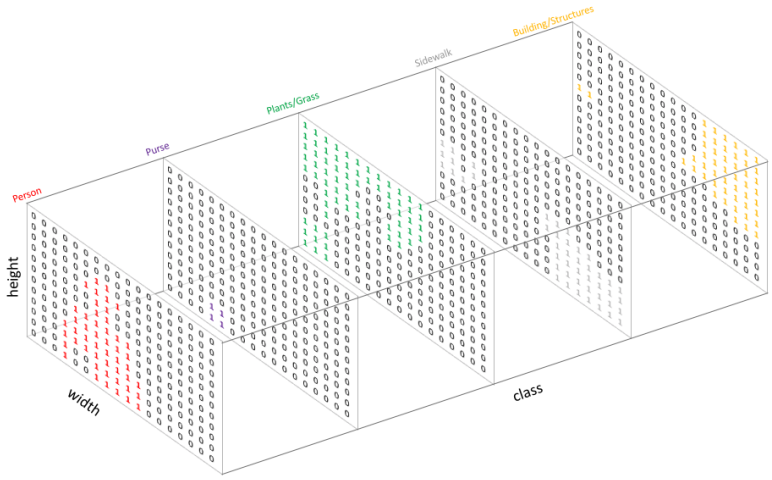


Hình 2: Semantic image segmentation

Trong bài toán này, sample trong training set thường gồm ảnh và segmentation map tương ứng:

- Mỗi ảnh được biểu diễn bởi 1 tensor 3 chiều kích thước $height \times width \times 3$ với ảnh màu hoặc $height \times width \times 1$ với ảnh xám.
- Segmentation map biểu diễn label tương ứng với mỗi pixel trong ảnh đầu vào, label này được xác định bởi một số nguyên. Segmentation map có kích thước $height \times width \times 1$.

Giống với bài toán classification, label của mỗi pixel có thể được encode bằng one hot encoding và thuật toán học sẽ cho ra một predict segmentation map P, kích thước $height \times width \times$ num class, với $P_{i,j}$ biểu diễn xác suất của pixel (i, j) ứng với mỗi class.

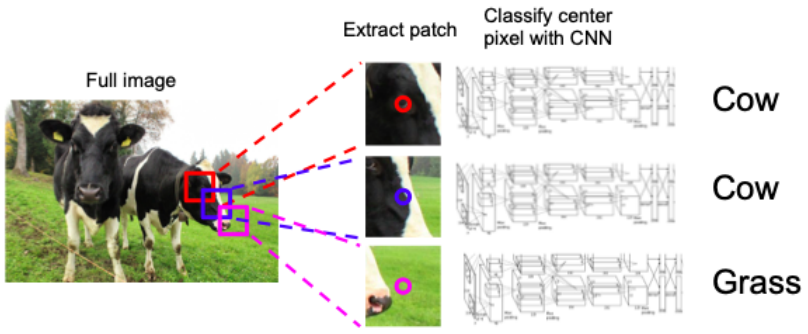


Hình 3: One-hot encoding trong bài toán segmentation

Sau đây là một số ý tưởng được dùng để giải quyết bài toán này.

2.1 Sliding window

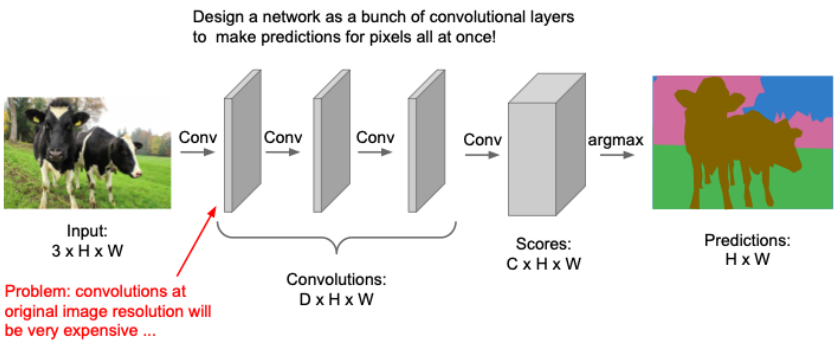
Đây là cách tiếp cận sử dụng một mạng CNN classification để gán nhãn cho một pixel bằng cách crop xung quanh pixel đó và thực hiện classification trên ảnh này. Có thể thấy điểm trừ của phương pháp này là hiệu quả tính toán. Với mỗi pixel trong ảnh ta lại cần chạy model CNN cho lận cận của nó và giữa các pixel không có sự chia sẻ trong việc tính toán mặc dù giao của lân cận của 2 pixel liền kề nhau là rất lơn. Hơn nữa thông tin được dùng để dự đoán nhãn cho mỗi pixel bị giới hạn bởi crop window của pixel đó, dẫn đến một số trường hợp thông tin này là không đủ để model thực hiện phân lớp.



Hình 4: Phương pháp sliding window. Nguồn: CS231n Stanford

2.2 Fully Convolutional 1

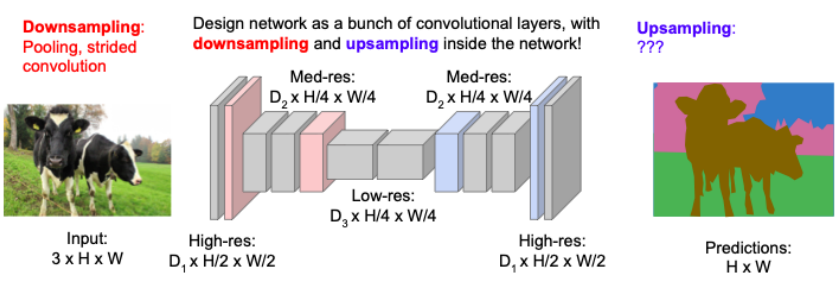
Phương pháp này sử dụng kiến trúc CNN với feature map giữa các convolution layer có kích thước bằng ảnh input. Dù đã giảm số lần chạy trên mỗi input về một lần duy nhất và thực hiện share computing giữa các pixel, tuy nhiên cách tiếp cận này vẫn yêu cầu lượng tài nguyên tính toán tương đối lớn do kích thước của feature map không giảm.



Hình 5: Phương pháp fully convolution. Nguồn: CS231n Stanford

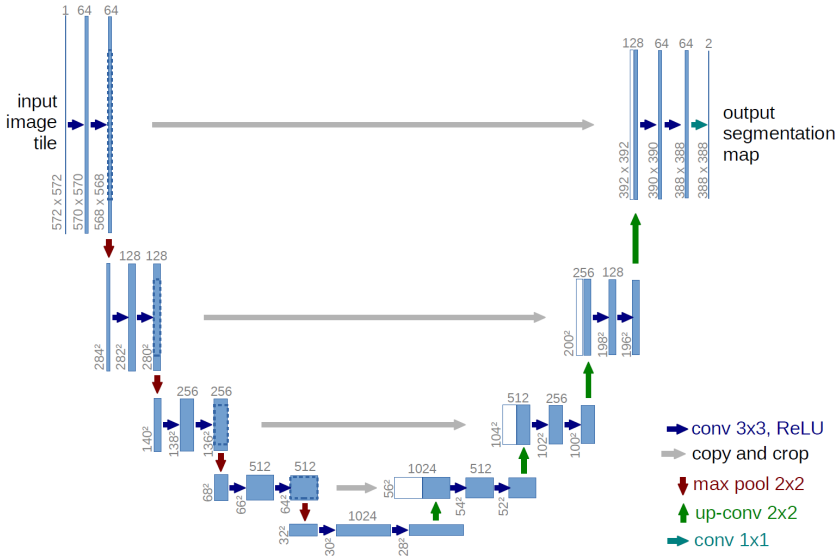
2.3 Fully Convolutional: Encoder - Decoder

Kế thừa ý tưởng từ phương pháp trên, cách tiếp cận này sử dụng mô hình encoder - decoder. Encoder sẽ thực hiện trích chọn các đặc trưng ở nhiều cấp độ từ ảnh input, downsample feature map và nén các thông tin trong ảnh vào các feature channel. Decoder sẽ sử dụng các thông tin này để xây dựng lại segmentation map có kích thước bằng ảnh đầu vào qua các bước upsampling. Bằng cách này, ta có thể giảm lượng tài nguyên yêu cầu do giảm kích thước feature map trong quá trình tính toán



Hình 6: Phương pháp fully convolution với encoder decoder. Nguồn: CS231n Stanford

Và để cải tiến phương pháp này, kiến trúc UNet đã được đề xuất bởi Olaf Ronneberger, Philipp Fischer và Thomas Brox, sử dụng các skip connection giữa encoder và decoder, kiến trúc này giúp decoder tận dụng các thông tin chi tiết về không gian ở các lớp encoder trong quá trình upsample feature map.



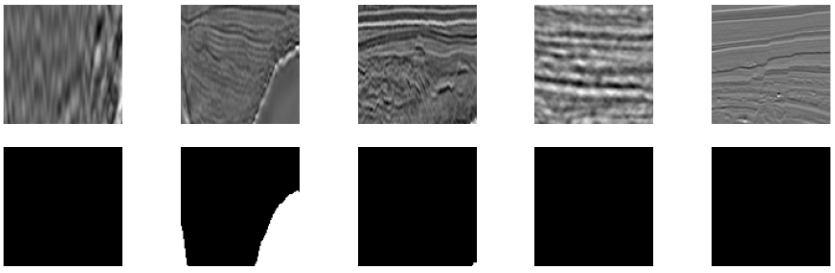
Hình 7: UNet architecture.
Nguồn: U-Net: Convolutional Networks for Biomedical Image Segmentation

Từ những tìm hiểu trên chúng tôi quyết định sử dụng kiến trúc UNet để giải quyết bài toán **TGS Salt Identification Challenge**. Chúng tôi dự định sẽ thử nghiệm với nhiều biến thể UNet với những backbone khác nhau.

3 Quá trình thực hiện

3.1 TGS Salt Identification Challenge Dataset

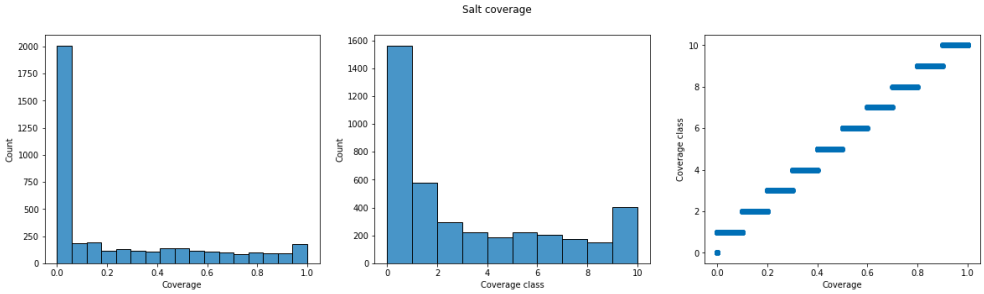
Dữ liệu gồm 4000 training sample (image, mask) và 18000 test sample (chỉ có image). Mỗi sample được định danh bằng **id**. Toàn bộ **id** sẽ được cung cấp kèm với độ sâu tại tại vị trí của ảnh trong file **depths.csv**. **id** của các sample thuộc training set được cung cấp trong file **train.csv**, ngoài ra file này còn bao gồm **rle_mask** là mask map của sample đã được mã hóa bằng running length encoding.



Hình 8: Một số ví dụ training sample

Tính độ bao phủ của muối cho mỗi sample trong tập training và phân coverage class cho mỗi sample từ độ bao phủ:

- 0% - class 0
- 0% < x <= 10% - class 1
- ...



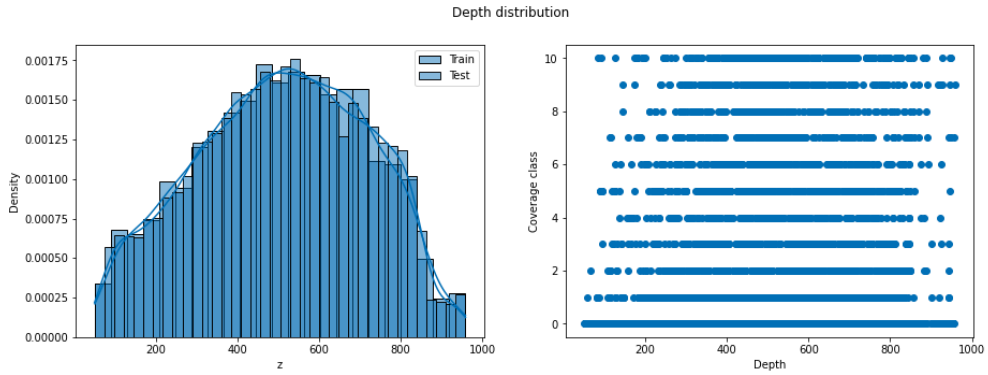
Hình 9: Một số biểu đồ thống kê độ phủ trên traning set. Từ trái qua phải (Biểu đồ 1) Độ bao phủ của muối trong ảnh. (Biểu đồ 2) Phân mỗi sample vào class tương ứng với độ phủ. (Biểu đồ 3) Phân bố của độ phủ trên mỗi class

Nhận xét:

- Có lượng lớn sample trong training set có nhãn rỗng, tức mask map chỉ gồm backgroun, cần lưu ý rằng theo tác giả của cuộc thi, các ảnh bị bao phủ hoàn toàn bởi muối sẽ không có nhãn (mask map đen hoàn toàn) do họ chỉ quan tâm đến phần viền của hố muối.
- Với các class còn lại phân bố của độ phủ tương đối đều

Tiếp theo, chúng tôi thực hiện phân tích phân bố độ sâu của các sample trong tập training set và test set. Nhận thấy:

- Từ biểu đồ 4 có thể thấy 2 phân bố độ sâu khá tương đồng giữa training set và test set. Ta kì vọng training set có khả năng biểu diễn tốt cho test set
- Biểu đồ 5 cho thấy phân bố độ sâu tương đối đều và tương đồng giữa các class. Có thể thông tin về độ sâu không cung cấp nhiều thông tin về độ phủ.



Hình 10: Từ trái qua phải (Biểu đồ 4) Phân bố của depth trong training set và test set. (Biểu đồ 5) Phân bố của depth ứng với mỗi coverage class.

3.2 Evaluation metric

Cuộc thi này sử dụng metric là mAP (mean Average Precision) với 10 ngưỡng IoU (Intersection over Union) $t = (0.50, 0.55, ..., 0.95)$

Với Y_i là ground truth segmentation map và $\hat{Y}_i$ là predict từ model cho sample i thuộc test dataset. Với mỗi ngưỡng t, Precision của sample i được tính theo công thức:

$$P(t, Y_i, \hat{Y}_i) = \begin{cases} 0, & \text{if } |Y_i| = 0 \text{ and } |\hat{Y}_i| > 0 \\ 0, & \text{if } |Y_i| > 0 \text{ and } |\hat{Y}_i| = 0 \\ 1, & \text{if } |Y_i| = 0 \text{ and } |\hat{Y}_i| = 0 \\ IoU(Y_i, \hat{Y}_i) > t, & \text{otherwise} \end{cases}$$

Sau đó trung bình của precision hay $AP(Y_i, \hat{Y}_i)$ là trung bình của $P(t, Y_i, \hat{Y}_i)$ trên các ngưỡng t.

$$AP(Y_i, \hat{Y}_i) = \frac{1}{10} \sum_{k=1}^{10} P(t_k, Y_i, \hat{Y}_i)$$

Cuối cùng, mAP là trung bình của AP trên toàn bộ sample trong test dataset. Với n là số sample trong test dataset, mAP được tính theo công thức:

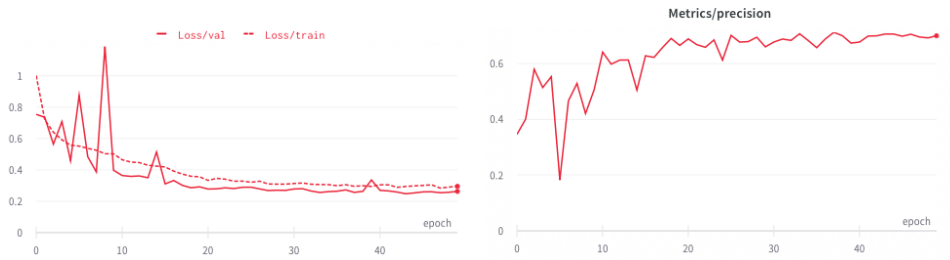
$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP(Y_i, \hat{Y}_i)$$

3.3 Baseline Implementation

Train dataset được chia làm 5 fold dựa theo độ sâu và được kiểm tra để đảm bảo phân bố của độ bao phủ của muối trong mỗi ảnh là như nhau. Với baseline, chúng tôi sử dụng fold 0 làm valid set, 4 fold còn lại được dùng làm training set. Training set được áp dụng một số phép augmentation như horizontal flip, horizontal sheer, random crop, rotate với góc nhỏ và thay đổi độ sáng với brightness limit 0.1 và contrast limit 0.08. Chúng tôi sử dụng thư viện albumentations cho augmentation. Sau đó, cả training set và validation set được resize thành kích thước 128×28 và đưa vào model.

Với baseline, chúng tôi sử dụng model UNet đơn giản với một ít thay đổi sau khi xem xét một số cài đặt như độ sâu của encoder và số filter sử dụng ở mỗi layer.

Binary cross entropy loss được sử dụng cùng Adam optimizer với lr=0.001 và train 50 epoch. Kết quả được thể hiện ở biểu đồ sau:



Hình 11: Loss và mAP sau khi training

Hình 12: Kết quả khi submit lên kaggle

3.4 Một số quan sát khi training

- Tăng kích thước ảnh đầu vào không làm tăng kết quả
- Đối với baseline model, tăng số filter (>16 với lớp convolution đầu tiên) không làm tăng kết quả, tương tự với tăng độ sâu

3.5 Data Augmentation

Để đưa ra lựa chọn cho data augmentation chúng tôi thực hiện một số thí nghiệm sử dụng baseline model sau đó quan sát kết quả thu được như sau:

Data Augmentation	Public Score	Private Score
HFlip (p=0.5)	0.74346	0.77564
HFlip + HSheer (-0.1, 0.1, p=0.5)	0.75829 (+1.49)	0.78938
HFlip + ShiftScaleRotate	0.76563	0.796
HFlip + Brightless + Contrast	0.7558	0.7884
All (+100 epoch)	0.77681	0.8043

3.6 Loss Function

Sự lựa chọn hàm mục tiêu hay loss function là cực kì quan trọng trong việc thiết kế một hệ thống dựa trên kiến trúc deep learning do nó xác định mục tiêu cho quá trình học của thuật toán. Trong bài toán semantic segmentation, các nhà nghiên cứu đã thử nghiệm rất nhiều hàm loss của nhiều domain khác nhau. Những hàm loss này có thể được phân vào 4 loại chình bao gồm: Dựa trên phân bố (Distribution-based), Dựa trên khu vực (Region-based), Dựa trên đường bao (Boundary base) và loại tổ hợp của các loại trên (Compounded).

Loại	**Hàm Loss**
Dựa trên phân bố	Binary Cross-Entropy Weighted Cross-Entropy Focal Loss
Dựa trên khu vực	Dice Loss Sensitivity-Specificity Loss Focal Tversky Loss Log-Cosh Dice Loss
Dựa trên đường bao	Hausdorff Distance loss Shape aware loss
Tổ hợp	Combo Loss Exponential Logarithmic Loss

Bảng 1: Các loại hàm loss trong bài toán semantic segmentation

3.6.1 Binary Cross-Entropy Loss

Cross-Entropy đo sự khác nhau giữa 2 phân bố xác suất của biến ngẫu nhiên cho trước. Nó thường xuyên được sử dụng như hàm mục tiêu của các bài toán phân lớp, và do segmentation là bài toán phân lớp ở mức độ điểm ảnh, do đó Cross-Entropy cũng có thể được sử dụng. Trong cuộc thi này, do chỉ có 2 nhãn nên ta sẽ quan tâm đến Binary Cross-Entropy, được định nghĩa như sau:

$$L_{BCE}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{1}$$

Với $\hat{y}$ là dự đoán được đưa ra bởi mô hình. Với bài toán segmentation ta cần tính L_{BCE} cho mỗi pixel sau đó lấy trung bình trên mỗi ảnh và lấy trung bình trên dataset.

3.6.2 Dice Loss

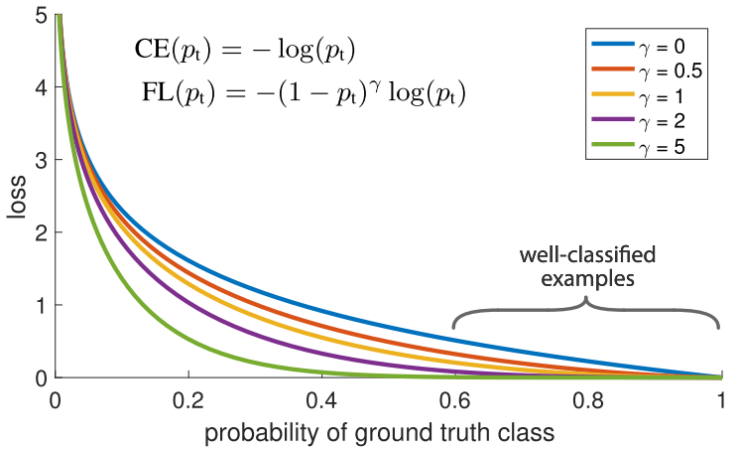
Hệ số tương quan Dice là độ đo được sử dụng rộng rãi trong cộng đồng thị giác máy để tính toán sự tương quan giữa 2 ảnh, nó cũng được sử dụng làm metric đánh giá kết quả của mô hình segmentation. Hệ số này được điều chỉnh để trở thành loss function được biết đến như Dice Loss:

$$DL(y, \hat{p}) = 1 - \frac{2y\hat{p}}{y + \hat{p} + 1} \tag{2}$$

Trong đó, 1 được thêm vào ở mẫu số để tránh trường hợp DL trở thành không xác định khi $y = \hat{p} = 0$.

3.6.3 Focal Loss

Trong bài toán Segmentation, Focal Loss được đưa ra để giải quyết trong trường hợp có sự mất cân bằng lớn giữa các foreground và background classes trong huấn luyện, chẳng hạn 1:1000. Focal loss có công thức khá đơn giản, cách tính Focal loss cho dự đoán có giá trị p_t và nhãn đúng là positive như sau: $FL(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t)$. Nhận thấy khi hệ số focusing $\gamma = 0$ Focal loss tương ứng Cross Entropy, đồ thị 13 biểu diễn Focal loss với khác nhau. Đối với các nhãn có số lượng lớn, giúp dễ học hơn thì trong quá trình huấn luyện, trọng số của các sample có nhãn này đóng góp vào loss sẽ nhỏ hơn. Điều này khiến mô hình tập trung vào các pixel khó học.



Hình 13: Đồ thị biểu diễn Focal Loss theo p_t. Có thể thấy với các dự đoán gần tốt (p_t gần với 1) thì Focal Loss phạt ít hơn Cross Entropy, như vậy sẽ giúp mô hình tập chung học để cải thiện các dự đoán chưa tốt hơn

3.6.4 Lovasz Hinge Loss

Lovasz Hinge Loss [6] cũng giống với Dice Loss, hàm loss này giúp tối ưu trực tiếp cho metric mIoU, tuy nhiên được áp dụng kĩ thuật lovasz để làm mượt nên dễ dàng hơn khi tối ưu bằng các thuật toán gradient. Đây cũng là hàm loss chúng tôi lựa chọn sử dụng cho mô hình cuối cùng.

Loss function	Public Score	Private Score
BCE	0.7311	0.7628
Dice Loss	0.7360	0.7686
Dice BCE Loss	0.7353	0.7680
Focal Loss	0.7401	0.7699
Lovasz Hinge Loss	0.7667	0.7960

Bảng 2: Kết quả thử nghiệm với nhiều loss function trên baseline model (không augmentation), được train với Adam optimizer, lr=1e-3. Có thể thấy Lovasz Hinge Loss cho kết quả rất tốt

3.7 Test Time Augmentation

Sau khi thêm augmentation là Horizontal Flip trong quá trình dự đoán, chúng tôi thu được kết quả trong bảng 3

3.8 Các kiến trúc

3.8.1 Mish Activation Function

Hàm kích hoạt không tuyến tính Mish [7] có công thức là $f(x) = x * \tanh(\ln(1+\exp(x)))$. Mish activation function có một số tính chất tốt của hàm activation đó là:

Loss function	Public Score	Private Score
BCE	0.7540	0.7852
Dice Loss	0.7464	0.7786
Dice BCE Loss	0.7541	0.7886
Focal Loss	0.7637	0.7965
Lovasz Hinge Loss	0.7737	0.8045

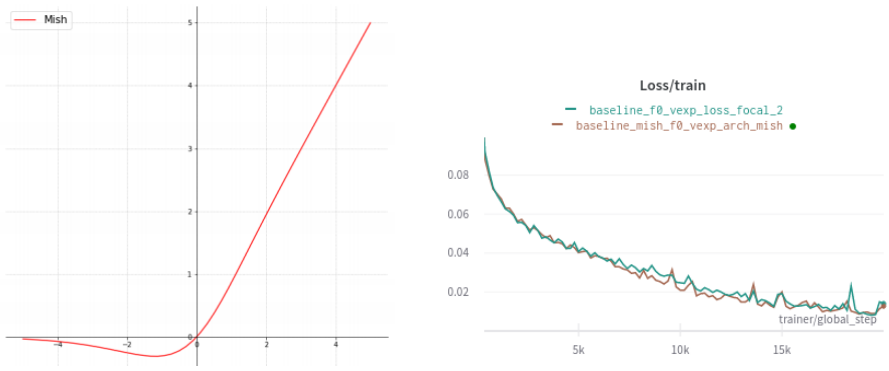
Bảng 3: Kết quả thử nghiệm ở bảng 2 sau khi sử dụng thêm TTA

1. Không có chặn trên: Hàm kích hoạt Mish có một ưu điểm đó là tránh được sự bão hòa và triệt tiêu gradents vốn là nhược điểm của hàm Sigmoid.

2. Có chặn dưới: Hàm kích hoạt Mish thể hiện được sự hiệu quả trong regularization.

3. Hàm không đơn điệu: Do hàm vẫn giữ lại các giá trị gradient âm nhỏ nên cho phép mạng học tốt hơn bằng việc cho gradient chạy trên miền âm.

4. Tính liên tục: Đạo hàm bậc nhất của hàm Mish liên tục trên toàn bộ miền giá trị, nhờ đó có được hiệu quả trong optimization và generalization.



Hình 14: Đồ thị hàm kích hoạt Mish 14a và train Focal Loss khi sử dụng Mish Activation và ReLu Activation Function 14b. Có thể thấy loss khi sử dụng Mish activation hơi thấp hơn

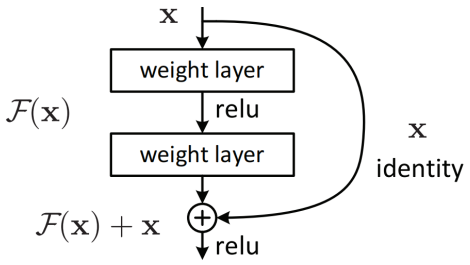
Sau khỉ thử thay hàm ReLu bằng Mish activation cho baseline, từ đồ thị 14 chúng tôi nhận thấy sử dụng Mish có thể tăng tốc độ hội tụ một lượng nhỏ. Tuy nhiên do sử dụng cài đặt không tối ưu cho CUDA và do có chứa hàm exponential, sử dụng Mish làm tăng thời gian huấn luyện một epoch (khoảng gần 1s với baseline). Chúng tôi sử dụng Mish Activation cho các bước tiếp theo.

3.8.2 Encoder

Resnet34

Như đã thảo luận trong báo cáo đầu tiên, sau khi thử nghiệm với mô hình baseline, chúng tôi cài đặt Unet với backbone là Resnet34.

Trước tiên ta cần tìm hiểu về mục đích sử dụng Resnet34 trong bài toán này. Trong một mô hình học sâu, nếu mạng đủ sâu thì sẽ xảy ra hiện tượng Vanish Gradient. Nguyên nhân của vấn đề này là do trong quá trình tiến hành traning với thuật toán lan truyền ngược Backpropagation, gradient tính toán được từ hàm mất mát sẽ tiền dần về 0 do việc tiến hành nhân liên tiếp các giá trị nhỏ hơn 1 với nhau và dẫn tới việc cập nhập các hệ số như weight và bias trong Gradient Descient không còn hiệu quả, làm giảm chất lượng học của các hệ số nerual network. Vì vậy Resnet đã ra đời với ý tưởng sử dụng các kết nối tắt đồng nhất để đi xuyên qua một hoặc nhiều lớp.



Hình 15: Kiến trúc cơ bản của Resnet

Mô tả một cách ngắn gọn kiến trúc trên của ResNet, gọi $H(x)$ là giá trị ánh xạ lý tường muốn được học. Ta định nghĩa các tầng xếp chồng phi tuyến tín trước khi cộng với giá trị x là $F(x) = H(x) - x$. Do đó ánh xạ mong muốn sẽ có thể được biến đổi thành $F(x) + x$. Cần lưu ý rằng ta đặt ra giả thuyết rằng việc tối ưu ánh xạ phần dư sẽ dễ dàng hơn là việc tối ưu ánh xạ gốc, vì trong thực tế nếu như một ánh xạ đồng nhất (identity mapping) là tối ưu thì việc đặt phần dư $F(x) = 0$ sẽ dễ dàng hơn việc khớp ánh xạ phần dư với các lớp phi tuyến.

Công thức $F(x) + x$ có thể đạt được bằng việc feedforward neural network với kết nối tắt như trong Hình 16. Các kết nối tắt đơn giản là thực hiện ánh xạ đồng nhất, kết quả của nó sẽ được cộng với kết quả của các lớp xếp chồng. Các kết nối tắt này không tạo thêm các tham số cũng như không tăng độ phức tạp tính toán. Do đó mạng vẫn có thể được train từ đầu đến cuối bằng SGD với thuật toán Backpropagation một cách hiệu quả.

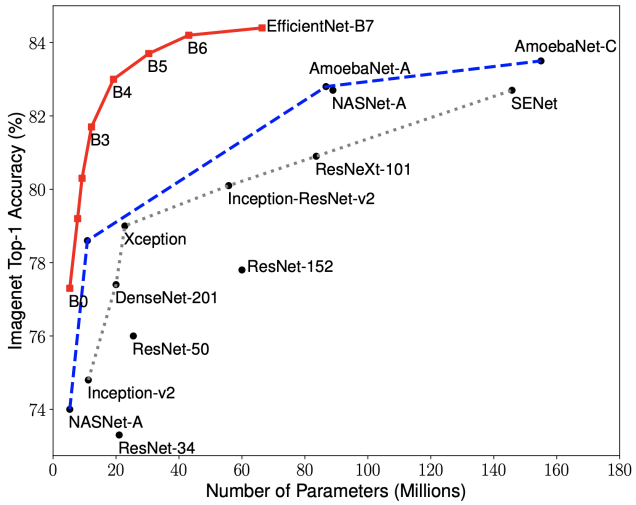
UEfficientNet

Mở rộng mô hình hay nói rõ hơn là quyết định cách tăng kích thước mô hình để mang lại độ chính xác tốt hơn là một vấn đề quan trọng của việc thiết kế và cài đặt CNN. Đây là quá trình tốn rất nhiều thời gian và tài nguyên để xử lý. Và để giải quyết cho vấn đề này thì EfficientNet là một trong những giải pháp. Kỹ thuật này cho phép tạo ra các mô hình với độ chính xác cao hơn đồng thời giảm đáng kể FLOPS (Floating-point Operations Per Second) và kích thước mô hình tổng thể.

Một mô hình CNN có thể được mở rộng theo 3 chiều: chiều sâu (depth), chiều rộng (width) và độ phân giải (resolution). Khó khăn của việc mở rộng này đó là 3 yếu tố kể trên phụ thuộc lẫn nhau và những giá trị của nó thay đổi bởi những ràng buộc tài nguyên khác nhau. Do đó, thường các phương pháp mở rộng sẽ chỉ mở rộng theo 1 trong 3 chiều. Việc mở rộng theo một chiều bất kì nào trong ba chiều kể trên đều sẽ cải thiện độ chính xác, nhưng độ chính xấc lải giảm bớt đi khi mô hình càng lớn. Và để có thể nhận được độ chính xác và hiệu quả cao hơn, điều quan trọng là cần phải cân bằng cả ba chiều - chiều sâu, chiều rộng, độ phân giải của mạng. Với EfficientNet, ta sẽ sử dụng phương pháp mở rộng riêng - Compound Scaling Method, sử dụng một hệ số ϕ để mở rộng đồng nhất cả chiều rộng, chiều sâu và độ phân giải:

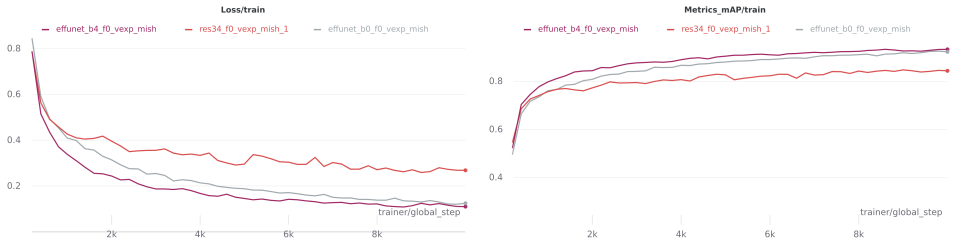
$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \end{aligned}$$

Trong đó, ϕ là hệ số do người dùng tự chỉ định. Hệ số ϕ này sẽ kiểm soát số tài nguyên có thể được dùng tiếp để mở rộng model. Còn α , β, γ là những hằng số theo thứ tự sẽ thể hiện cách đưa những tài nguyên thêm cho chiều sâu, chiều rộng và độ phân giải. α , β, γ có thể được xác định bằng tìm kiếm theo lưới (Grid Search) khi để giá trị $\phi = 1$ và tìm giá trị cho độ chính xác cao nhất. Sau khi xác định được các hằng số α , β, γ , ta có thể tăng dần ϕ để có thể có được mô hình mở rộng hơn với độ chính xác cao hơn. Từ đó ta có được các EfficentNet-B1 đến EfficientNet-B7, với số nguyên cuối ở tên tương ứng với giá trị của hệ số ϕ.



Hình 16: Kết quả của một số kiến trúc bao gồm Efficient Net và ResNet trên Imagenet

Kết quả thử nghiệm với Resnet34, EfficientNet B0 và EfficientNet B4 làm encoder

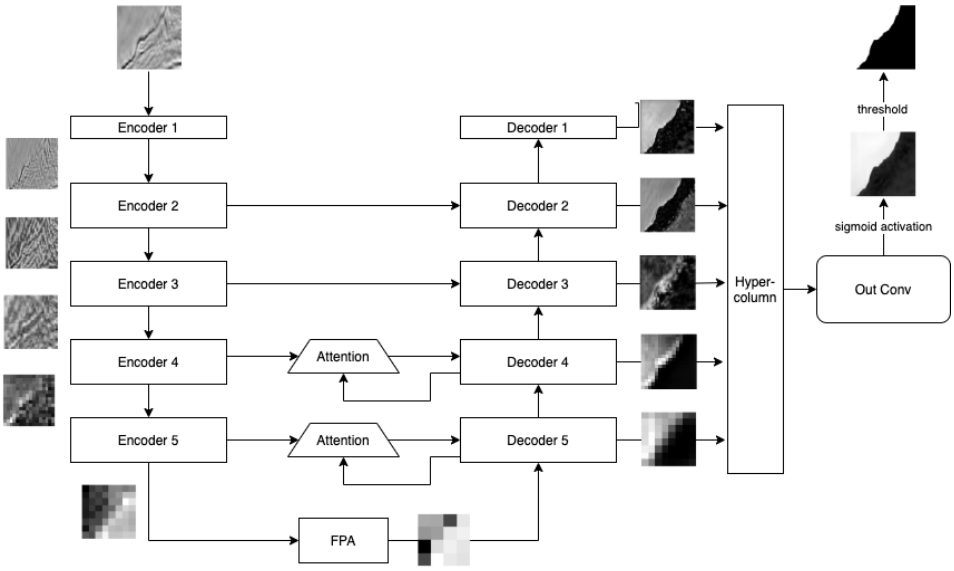


Hình 17: Loss và mAP trên tập huấn luyện khi thử nghiệm với resnet34, efficient net b0 và b4 làm encoder

Encoder	Public Score	Private Score	Number of parameters
ResNet34	0.8148	0.8466	30.4M
EfficientNet B0	0.8080	0.8290	6.6M
EfficientNet B4	0.8375	0.8544	22.2M

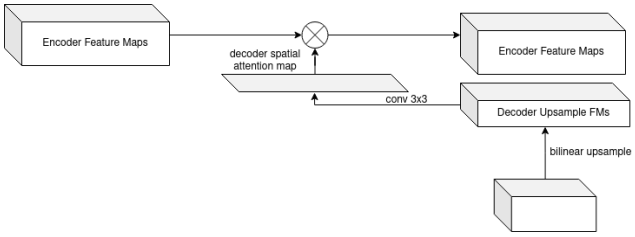
Bảng 4: Kết quả thử nghiệm các với pretrain encoder khác nhau trên kiến trúc Unet đơn giản. Do sử dụng ảnh input có kích thước 128x128, khá nhỏ so với kích thước input thường thấy của các pretrain model, chúng tôi giảm stride của lớp convolution đầu tiên xuống còn 1.

Dựa bào bảng 4, chúng tôi quyết định sử dụng EfficientNet B4 làm encoder. Mô hình cuối cùng chúng tôi sử dụng được miêu tả trong hình 18. Trong đó sử dụng hyper-column là concat của các feature map từ tất cả các decoder sau khi đã upsampling lên kích thước 128x128. Bằng cách này output convolution có thể cùng lúc sử dụng thông tin ở nhiều tầng decoder và gradient có thể nhanh chóng lan truyền trong mô hình. Ngoài ra còn có FPA hay Pyramid Attention Network là phương pháp attention trên nhiều kích thước khác nhau, chi tiết trong bài báo [8]



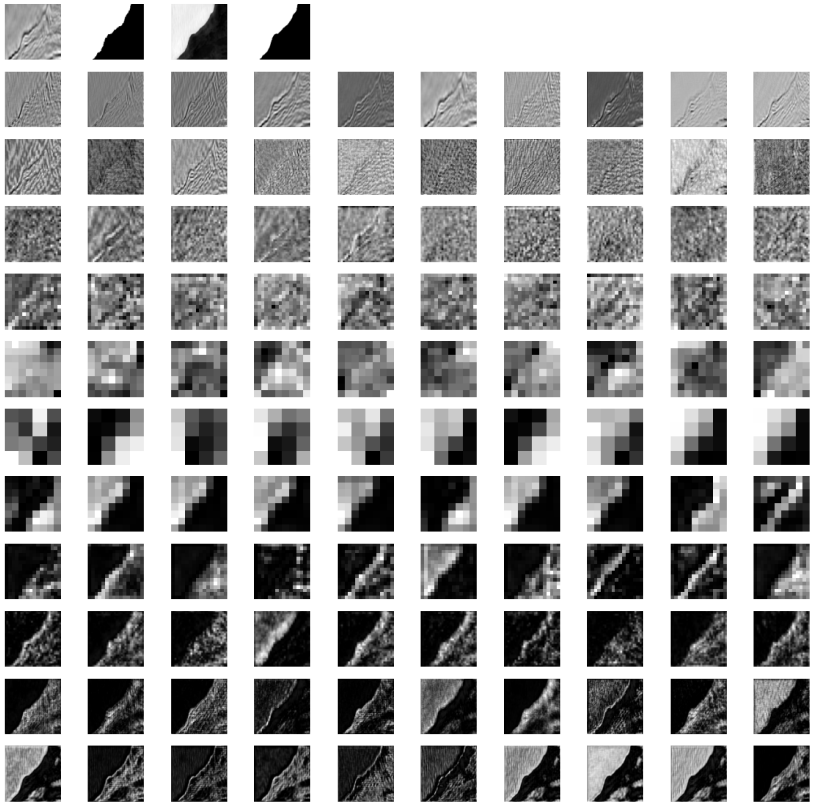
Hình 18: Mô hình được sử dụng.

3.9 Spatial Decoder-Encoder Attention

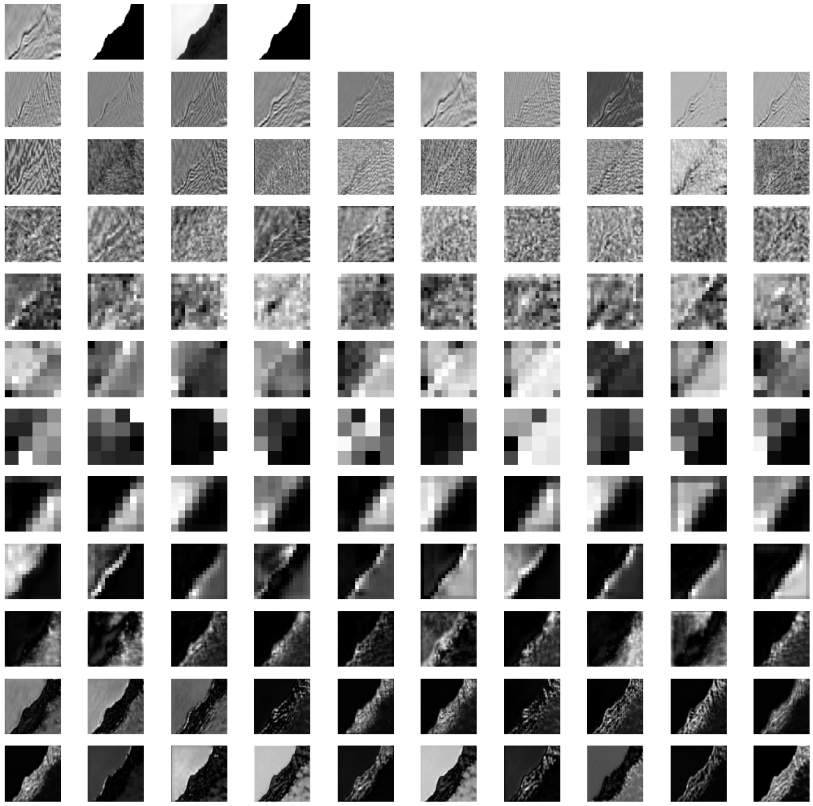


Hình 19: Mô hình Spatial Decoder-Encoder Attention được đề xuất. Bằng cách thực hiện một phép conv3x3 với dialation 2 có output channel là 1 trên decoder upsample feature map và đưa output này qua một hàm sigmoid, ta thu được một attention map của decoder. Cuối cùng nhân attention map này với output từ encoder tương ứng và concat kết quả với decoder upsample feature map.

Hình 20 biểu diễn feature map thu được khi không sử dụng spatial attention đề xuất trong 19, có thể thấy các feature map của decoder có chứa các vân (pattern) từ encoder rất rõ ràng. Hình 21 biểu diễn feature map thu được khi sử dụng spatial attention, bằng cách chỉ chú ý vào khu vực gần đường bao, mô hình vẫn có khả năng lấy thông tin từ encoder để làm rõ phần rìa nhưng không ảnh hưởng đến khu vực khác, có thể nhận thấy rõ điều này khi so sánh feature map của decoder2. Khi training chúng tôi quan sát thấy sử dụng spatial decoder cho kết quả validation mAP của các epoch đầu tiên cao hơn (từ 0.67 lêm 0.73). Tuy nhiên kết quả cuối cùng không bị ảnh hưởng đáng kể.



Hình 20: Các feature map khi không sử dụng spatial attention. Lần lượt từ trên xuống dưới là input, mask, sigmoid prediction, threshold prediction ở hàng 1, các hàng tiếp theo là feature map từ encoder1, encoder2, encoder3, encoder4, encoder5, center, decoder5, ..., decoder1



Hình 21: Các feature map khi sử dụng spatial attention. Lần lượt từ trên xuống dưới là input, mask, sigmoid prediction, threshold prediction ở hàng 1, các hàng tiếp theo là feature map từ encoder1, encoder2, encoder3, encoder4, encoder5, center, decoder5, ..., decoder1

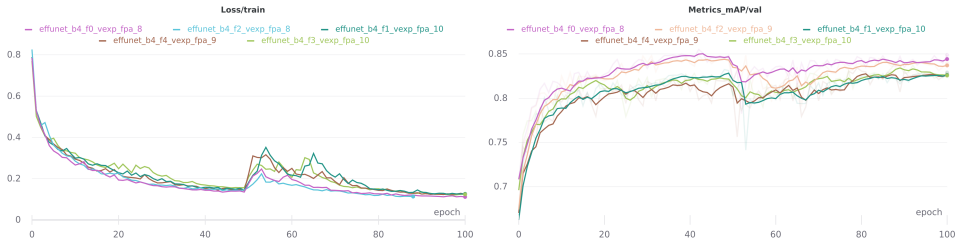
3.10 Quá trình training

Đầu tiên chúng tôi thử huấn luyện với AdamW optimizer, tuy gía trị loss giảm nhanh với các epoch đầu nhưng kết quả cuối cùng không cho validation mAP cao bằng sgd optimizer. Cuối cùng, chúng tôi sử dụng SGD optimizer với lr ban đầu là $1.5e^{-2}$, momentum là 0.9, weight decay $1e^{-4}$ và scheduler là CosineAnnealingWarmRestarts với một chu kì bằng 50 epoch và train với 100 epoch. Ở mỗi chu kì, chúng tôi lưu lại kết quả mAP tốt nhất trên validation set gọi là một snapshot. Ngoài ra, chúng tôi còn sử dụng Stochastic Weight Averaging lấy trung bình các tham số trong 5 epoch cuối, kĩ thuật này được cho là giúp generalization tốt hơn.

Như vậy sau khi train xong 5 fold, tổng cộng có 10 checkpoint model, kết quả cuối cùng được lấy trung bình từ 10 checkpoint này.

	Public Score	Private Score
Best single snapshot	0.8411	0.8639
Best single fold	0.8432	0.8676
All folds	0.8571	0.8747

Bảng 5: Bảng kết quả cuối cùng



Hình 22: Đồ thị kết quả training gồm training loss (bên trái) và validation mAP (bên phải). Các điểm bị nhô lên (xuống) ứng với bắt đầu chu kì mới của scheduler.

4 Kết luận

Trên đây là báo cáo của nhóm chúng tôi, chi tiết về model sau khi training xin xem notebook [3]. Chi tiết code cài đặt được public tại [9]. Kaggle kernel được dùng để huấn luyện có thể xem tại [2], pretrain model được lưu tại [10].

Tài liệu

[1] Notebook phân tích dataset - `https://colab.research.google.com/drive/1IFeFWRUt3CWzJ2GGarLKZ3mT-95CRfYS`

[2] Notebook training - `https://www.kaggle.com/ngueih/tgs-salt-training-eba96a`

[3] Phân tích kết quả training - `https://github.com/NNHieu/INT3405_TGSSalt/blob/master/notebooks/PostTraining.ipynb`

[4] CS231n: Convolutional Neural Networks for Visual Recognition Stanford - `http://cs231n.stanford.edu/`

[5] U-Net: Convolutional Networks for Biomedical Image Segmentation - Olaf Ronneberger, Philipp Fischer, Thomas Brox

[6] The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks - Maxim Berman, Amal Rannen Triki, Matthew B. Blaschko

[7] Mish: A Self Regularized Non-Monotonic Neural Activation Function - Diganta Misra

[8] Pyramid Attention Network for Semantic Segmentation - Hanchao Li, Pengfei Xiong, Jie An, Lingxue Wang

[9] `https://github.com/NNHieu/INT3405_TGSSalt`

[10] `https://drive.google.com/drive/folders/13ARazKIkSQokxZVQSI9D8JsnpubcBOHn?usp=sharing`