**I. Mô hình tổng quát**

**1. Lý do chọn đề tài**

[<https://arxiv.org/pdf/2011.03149.pdf>]

[<https://arxiv.org/pdf/2008.12603.pdf>]

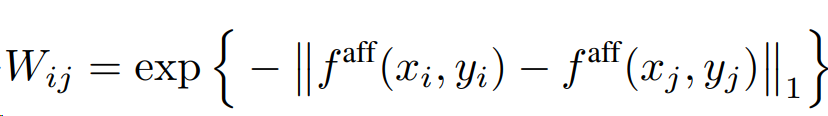
- Cần biết số lượng cá, kích thước cá,... để đánh giá tổng quan môi trường nước.

- Đo đạc bằng thước, xuống dưới nước chụp ảnh mất thời gian và công sức.

- Tạo label đầy đủ mất thời gian, công sức, đắt đỏ mà có thể không chính xác; label bằng point, từ point viền lên 1 con cá (point-level supervision, weakly-supervised segmentation).

[cái này văn vẻ sau =(((((((]

**2. Affinity Matrix**

Ma trận quan hệ (affinity matrix) hay còn gọi là ma trận tương đồng (similarity matrix), là một công cụ thống kê có vai trò biểu diễn sự tương đồng, “thân cận” của các điểm dữ liệu. Sự “thân cận” ở đây có thể hiểu theo nghĩa gần giống với khoảng cách, ngoại trừ việc nó không thỏa mãn các tính chất của một metric. Cụ thể hơn, hai điểm dữ liệu càng giống nhau thì mức độ thân cận của chúng càng gần 1 thay vì 0. Hai ví dụ điển hình cho độ đo này có thể kể đến hệ số tương đồng cosine và Jaccard. Một cách hiểu khác cho độ đo này đó là xác suất mà hai điểm dữ liệu có sự tương đồng với nhau. Một cách phổ biến để tính affinity giữa hai điểm dữ liệu:

Đối với đồ thị, ma trận affinity là một ma trận vuông cấp n với n là số đỉnh của ma trận, mỗi giá trị ở hàng i cột j tượng trưng cho độ tương đồng giữa đỉnh i và đỉnh j, do vậy ma trận affinity trong trường hợp này là một ma trận đối xứng và xác định không âm. Đối với các bài toán về Computer Vision, mỗi bức ảnh mà một đồ thị có trọng số với mỗi pixel là 1 đỉnh và trọng số trên mỗi cạnh biểu thị sự tương đồng giữa mỗi pixel tùy theo đặc thù của bài toán, các trọng số này sẽ hình thành một ma trận affinity với số cấp là số pixel của bức ảnh. Đối với image segmentation, ma trận affinity nên cho thấy sự tương đồng về ngữ nghĩa (semantic-level similarities) giữa mỗi pixel.

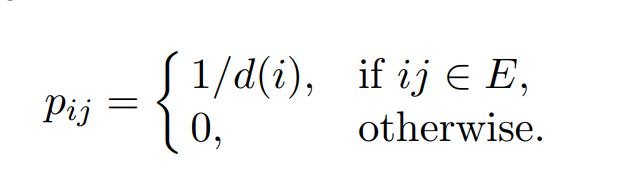
Trong image segmentation, phương pháp affinity được đưa vào cuối model dự đoán để tinh chỉnh lại ảnh phân vùng nhằm tăng độ chính xác ở biên của vật thể. Một trong những cách tinh chỉnh đó là sử dụng Random Walk để khuyến khích các pixel có độ tương thích cao được gán cùng một label.

**3. Random Walk**

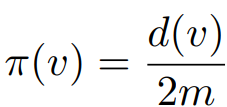
Cho một đồ thị và một tập đỉnh R, ta chọn một đỉnh để cho vào R, sau đó ta chọn ngẫu nhiên một đỉnh thuộc lân cận đỉnh đó để bổ sung vào R, dãy các đỉnh theo thứ tự bổ sung như vậy được gọi là Random walk.

Tiếp theo, cho G=(V,E) là một đồ thị liên thông với n đỉnh và m cạnh. Xét một random walk trên G, ta xuất phát tại đỉnh nếu ở bước thứ t ta đi đến đỉnh thì ta đi đến một trong những lân cận của với xác suất . Ta ký hiệu là phân phối xác suất của :

Ký hiệu M = ( với:



Gọi là ma trận đỉnh kề của G và D là ma trận chéo với , như vậy . Như vậy ta có thể biểu diễn phân phối xác suất tại thời điểm như sau:  và . Hơn nữa, nếu xuất phát tại đỉnh i thì xác suất mà ta đi đến đỉnh j tại thời điểm t chính là thành phần nằm ở hàng i cột j của ma trận .

Một tính chất quan trọng đó là nếu G không phải đồ thị hai phía thì phân phối xác suất của hội tụ về phân phối bất động (stationary distribution) với  khi t tiến ta vô cùng , tức .

**4. LCFCN loss function**

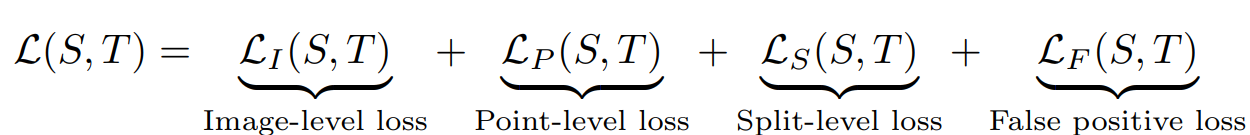
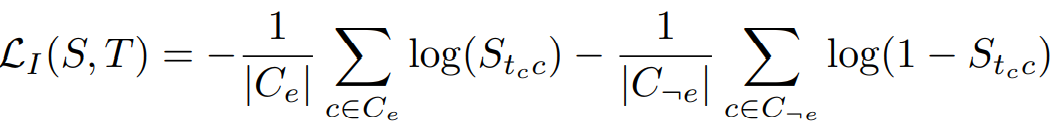
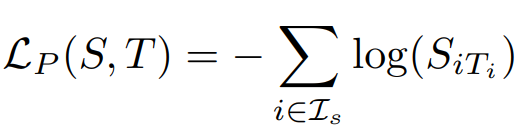


Image-level loss: Có vai trò giúp mô hình dự đoán các vật thể có trong ảnh



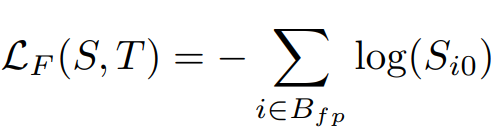
Point-level loss: Có vai trò giúp mô hình khoanh vùng vị trí của vật thể bằng một nhóm các pixel (blob) nằm trong phần vật thể được phân đoạn.





Split-level loss: Ngăn mô hình không dự đoán các blobs nằm trên nhiều vật thể một lúc.

False positive level loss: Ngăn không cho mô hình dự đoán blob không nằm trong vật thể nào.



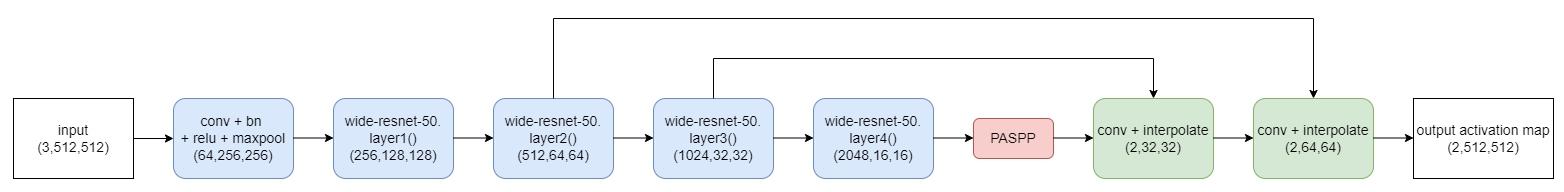
**II. Mô hình đề xuất**

**1. Mô hình đề xuất**

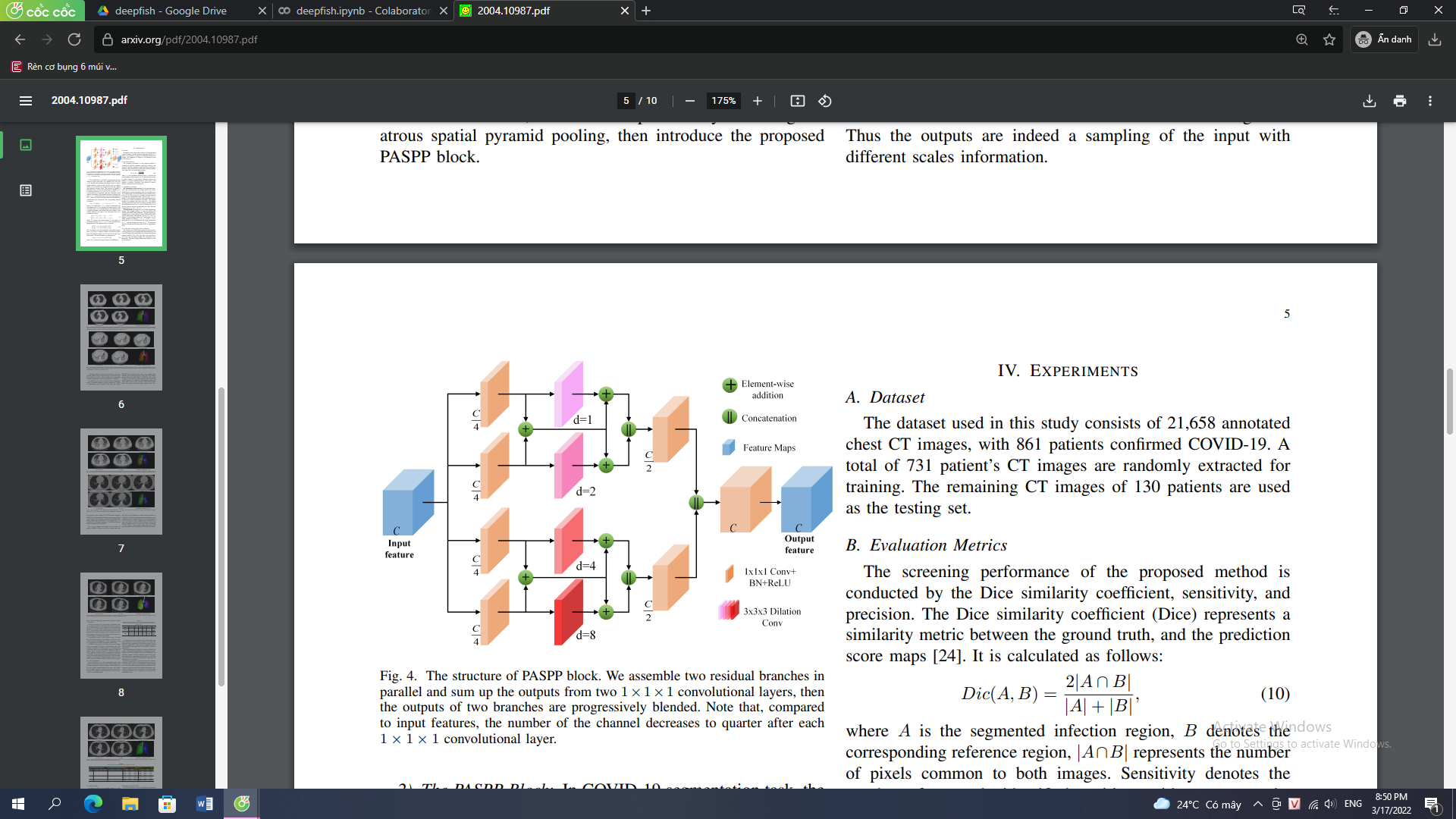
[https://www.kaggle.com/phcli2792000/medt-cell-paspp-global-4-layer]

FCN pretrain trên wide-resnet-50; có khối PASPP (lấy từ bài bạn Phúc); tùy chọn affinity True/False. Khác hẳn với tác giả dùng FCN pretrain trên VGG16.





[https://arxiv.org/pdf/2004.10987.pdf]



**2. Hàm loss đề xuất**

trong paper chỉ ghi mỗi LCFCN; còn trong code chính của tác giả thì là:

- Sau khi em đọc kĩ paper thì em thấy thực chất paper này nó dựng 1 cái stage làm pseudo-mask label từ các label bằng point; sau khi qua stage này rồi thì stage sau giống kiểu fully-supervised thích dùng model nào train thì dùng. Cơ mà riêng khoản in hẳn ra tất cả các ảnh thì cực khó vì code này người ta viết theo thư viện người ta tự làm với nhau nên bọn em chắc chỉ làm nổi 1 stage thôi ạ =(((( nên cái thành phần nó không thể có trong stage đầu tiên được.

- Loss không dùng Tversky hay Tversky-Kahneman được vì class imbalance nặng, đại lượng TN quá lớn làm cho loss quá nhanh hội tụ, mà hội tụ nhanh thì model sẽ dễ bị đứng ở điểm cực trị địa phương.

**III. Phân tích thiết kế cài đặt thử nghiệm**

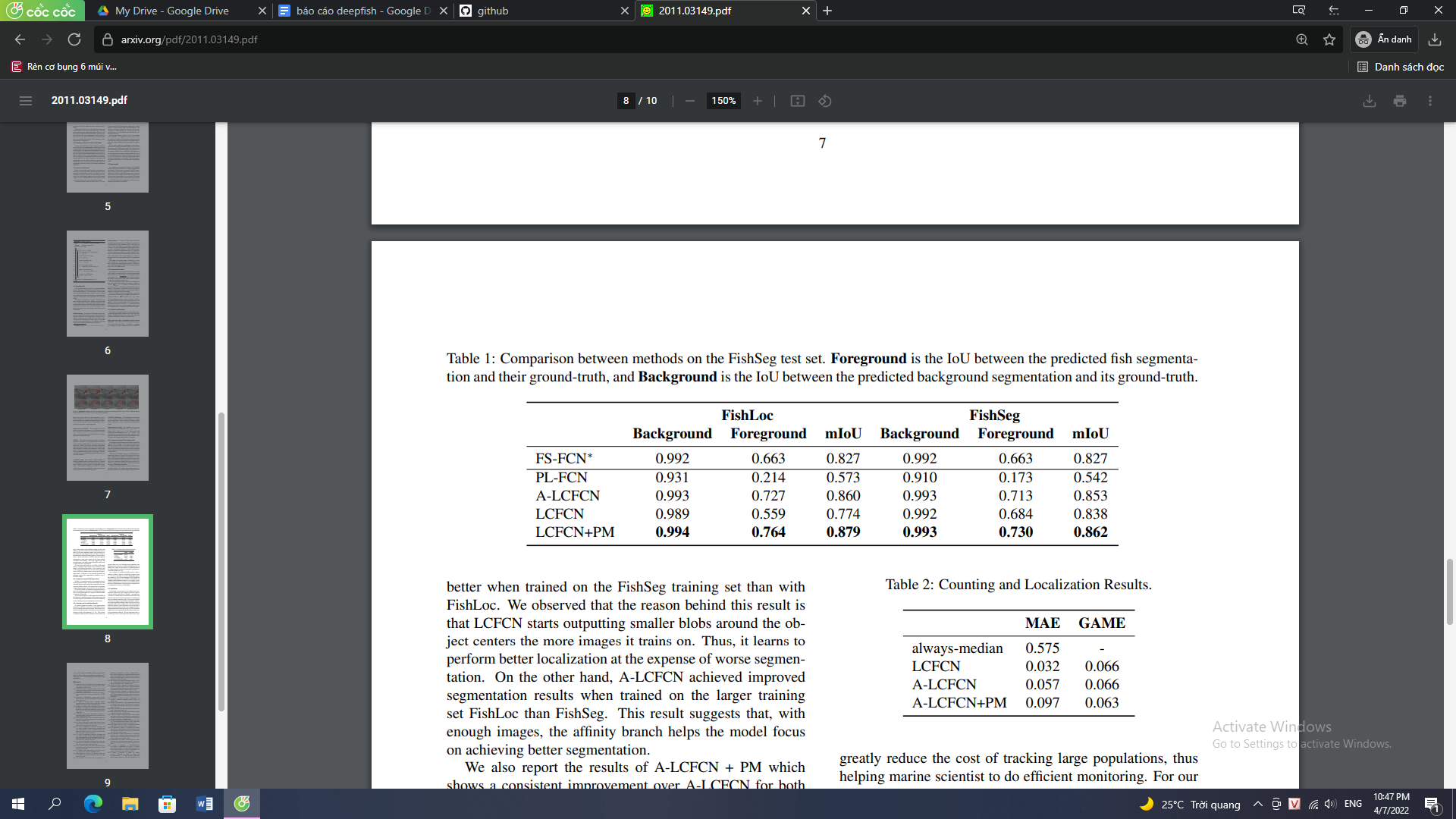
- Train với optimizer nadam, lr=1e-5. Model này cực nhạy với lr.

- Train trên train data của fishloc và fishseg; test trên test data của fishseg.

- Contribution so với bài gốc:

* train bằng optimizer nadam ≠ adam
* proposed model = fcn8 + wide\_resnet50 + paspp ≠ fcn8 + vgg16
* loss = focalCE + point-level + lcfcn ≠ lcfcn

- Kết quả author viết trong bài báo:



Trong bài bọn em w/o aff ≃ LCFCN còn w/ aff ≃ A-LCFCN. Cơ mà bọn em không hiểu tại sao nó cao hơn mà trong bài gốc author nó cũng không nói đến môi trường dùng. Nên bọn em mong muốn thầy cô có nhận xét về chọn số liệu nào, lược bỏ cắt gọn cái nào cho Báo cáo SVNCKH còn cái nào cho conference mới ạ. Còn phần toán em sẽ có trách nhiệm chi tiết với phần Dũng viết ạ.

**1. w/o aff**

| Model | loss | IoU\_class0  (background) | IoU\_class1  (fish) | IoU\_avg | test\_mae | test\_game |
| --- | --- | --- | --- | --- | --- | --- |
| Attention-UNet | CE | 0.9757 | 0.0004 | 0.4880 | 68.7688 | 17.4233 |
| FocalCE | 0.9781 | 4.6935 | 0.4890 | 1.5806 | 0.5537 |
| Point-level | 0.4685 | 0.0192 | 0.2438 | ∞ | ∞ |
| Tversky loss | 0.9211 | 0.0208 | 0.4710 | 4.4462 | 1.3239 |
| Dice loss | 0.8570 | 0.0210 | 0.4390 | 3.0376 | 0.9529 |
| FCN-ResNet50 | CE | 0.9966 | 0.8499 | 0.9233 | 0.0752 | 0.0430 |
| FocalCE | 0.9966 | 0.8516 | 0.9241 | 0.0752 | 0.0349 |
| Point-level | 0.9793 | 0.4708 | 0.7251 | 0.2419 | 0.1787 |
| Tversky loss | 0.9926 | 0.6713 | 0.8319 | 0.1075 | 0.0914 |
| Dice loss | 0.9927 | 0.6918 | 0.8422 | 0.1129 | 0.0739 |
| FCN-wide  -ResNet50 | CE | 0.9968 | 0.8566 | 0.9267 | 0.0806 | 0.0551 |
| FocalCE | 0.9967 | 0.8516 | 0.9241 | 0.0806 | 0.0497 |
| Point-level | 0.9720 | 0.3523 | 0.6622 | 0.7688 | 0.3239 |
| Tversky loss | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Dice loss | 0.9925 | 0.6895 | 0.8410 | 0.0914 | 0.0847 |
| FCN8-VGG16 | CE | 0.9970 | 0.8672 | 0.9321 | 0.1720 | 0.0618 |
| FocalCE | 0.9967 | 0.8616 | 0.9292 | 0.1237 | 0.0578 |
| Point-level | 0.9724 | 0.3824 | 0.6774 | 0.0538 | 0.1694 |
| Tversky loss | 0.9943 | 0.7444 | 0.8694 | 0.1075 | 0.0618 |
| Dice loss | 0.9952 | 0.7970 | 0.8961 | 0.0484 | 0.0470 |
| FCN8 | CE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| FocalCE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Point-level | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Tversky loss | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Dice loss | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Medical-  Transformer | CE | 0.9735 | 0.0057 | 0.4896 | ∞ | ∞ |
| FocalCE | 0.9737 | 0.0094 | 0.4915 | ∞ | ∞ |
| Point-level | 0.3020 | 0.0188 | 0.1604 | ∞ | ∞ |
| Tversky loss | 0.4948 | 0.0300 | 0.2624 | ∞ | ∞ |
| Dice loss | 0.4976 | 0.0304 | 0.2640 | ∞ | ∞ |
| PASPP-FCN  -wide-ResNet50 | CE | 0.9967 | 0.8550 | 0.9259 | 0.1075 | 0.0456 |
| FocalCE | 0.9971 | 0.8713 | 0.9342 | 0.1129 | 0.0524 |
| Point-level | 0.9633 | 0.3411 | 0.6522 | 0.1129 | 0.1738 |
| Tversky loss | 0.9939 | 0.7430 | 0.8684 | 0.0860 | 0.0752 |
| Dice loss | 0.9950 | 0.7966 | 0.8958 | 0.0860 | 0.0618 |
| ResUNet | CE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| FocalCE | 0.9910 | 0.6125 | 0.8018 | 0.2366 | 0.1156 |
| Point-level | 0.9910 | 0.6125 | 0.8018 | 0.2366 | 0.1156 |
| Tversky loss | 0.9904 | 0.6318 | 0.8111 | 0.2580 | 0.1478 |
| Dice loss | 0.9902 | 0.6372 | 0.8137 | 0.2258 | 0.1183 |
| Swin-UNet | CE | 0.9816 | 0.1397 | 0.5607 | 1.2581 | 0.3414 |
| FocalCE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Point-level | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Tversky loss | 0.8377 | 0.0400 | 0.4389 | ∞ | ∞ |
| Dice loss | 0.9048 | 0.1120 | 0.5084 | ∞ | ∞ |
| Transformer  -UNet | CE | 0.9540 | 0.0381 | 0.4961 | ∞ | ∞ |
| FocalCE | 0.9602 | 0.0278 | 0.4940 | ∞ | ∞ |
| Point-level | 0.6669 | 0.0180 | 0.3424 | ∞ | ∞ |
| Tversky loss | 0.6069 | 0.0183 | 0.3126 | ∞ | ∞ |
| Dice loss | 0.5793 | 0.0226 | 0.3010 | ∞ | ∞ |
| U-Net | CE | 0.9779 | 0.0 | 0.4890 | ∞ | ∞ |
| FocalCE | 0.9784 | 0.0 | 0.4892 | ∞ | ∞ |
| Point-level | 0.5211 | 0.0135 | 0.2673 | ∞ | ∞ |
| Tversky loss | 0.7259 | 0.0287 | 0.3772 | ∞ | ∞ |
| Dice loss | 0.7965 | 0.0436 | 0.4200 | ∞ | ∞ |

**2. w/ aff**

| Model | loss | IoU\_class0  (background) | IoU\_class1  (fish) | IoU\_avg | test\_mae | test\_game |
| --- | --- | --- | --- | --- | --- | --- |
| Attention-UNet | CE | 0.9787 | 0.0001 | 0.4894 | ∞ | 0.5376 |
| FocalCE | 0.9722 | 0.0007 | 0.4865 | ∞ | ∞ |
| Tversky loss | 0.8560 | 0.0394 | 0.4477 | ∞ | ∞ |
| Dice loss | 0.9787 | 0.0001 | 0.4894 | ∞ | 0.5376 |
| FCN-ResNet50 | CE | 0.9969 | 0.8614 | 0.9292 | 0.1183 | 0.0618 |
| FocalCE | 0.9970 | 0.8637 | 0.9303 | 0.1237 | 0.0497 |
| Tversky loss | 0.9952 | 0.7881 | 0.8917 | 0.0645 | 0.0618 |
| Dice loss | 0.9949 | 0.7798 | 0.8874 | 0.0439 | 0.0578 |
| FCN-wide  -ResNet50 | CE | 0.9969 | 0.8664 | 0.9317 | 0.1290 | 0.0484 |
| FocalCE | 0.9971 | 0.8717 | 0.9344 | 0.1183 | 0.0538 |
| Tversky loss | 0.9946 | 0.7659 | 0.8803 | 0.0439 | 0.0551 |
| Dice loss | 0.9935 | 0.7274 | 0.8604 | 0.0753 | 0.0753 |
| FCN8-VGG16 | CE | 0.9969 | 0.8596 | 0.9282 | 0.1075 | 0.0537 |
| FocalCE | 0.9972 | 0.8773 | 0.9373 | 0.2150 | 0.0725 |
| Tversky loss | 0.9957 | 0.8071 | 0.9014 | 0.0860 | 0.0484 |
| Dice loss | 0.9962 | 0.8315 | 0.9139 | 0.1667 | 0.0685 |
| FCN8 | CE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| FocalCE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Tversky loss | - | - | - | - | - |
| Dice loss | - | - | - | - | - |
| Medical-  Transformer | CE | 0.9676 | 0.0090 | 0.4883 | ∞ | ∞ |
| FocalCE | 0.9738 | 0.0119 | 0.4928 | ∞ | ∞ |
| Tversky loss | 0.5247 | 0.0306 | 0.2777 | ∞ | ∞ |
| Dice loss | 0.5209 | 0.0310 | 0.2760 | ∞ | ∞ |
| PASPP-FCN  -wide-ResNet50 | CE | 0.9974 | 0.8823 | 0.9398 | 0.1505 | 0.0565 |
| FocalCE | 0.9975 | 0.8888 | 0.9431 | 0.2688 | 0.0914 |
| Tversky loss | 0.9821 | 0.4950 | 0.7386 | 0.5108 | 0.1922 |
| Dice loss | 0.9821 | 0.4950 | 0.7386 | 0.5108 | 0.1922 |
| ResUNet | CE | 0.9921 | 0.6719 | 0.8320 | 0.7204 | 0.2366 |
| FocalCE | 0.9913 | 0.6624 | 0.8269 | 0.7097 | 0.2258 |
| Tversky loss | 0.9908 | 0.6362 | 0.8135 | 0.1720 | 0.1102 |
| Dice loss | 0.9903 | 0.6538 | 0.8222 | 0.3387 | 0.1546 |
| Swin-UNet | CE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| FocalCE | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Tversky loss | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Dice loss | 0.9792 | 0.0 | 0.4896 | 0.6129 | 0.1532 |
| Transformer  -UNet | CE | 0.9048 | 0.0238 | 0.4641 | ∞ | ∞ |
| FocalCE | 0.9444 | 0.0432 | 0.4938 | ∞ | ∞ |
| Tversky loss | 0.6606 | 0.0177 | 0.3391 | ∞ | ∞ |
| Dice loss | 0.4861 | 0.0222 | 0.2541 | ∞ | ∞ |
| U-Net | CE | 0.9776 | 0.0 | 0.4889 | ∞ | ∞ |
| FocalCE | 0.9783 | 0.0 | 0.4892 | ∞ | ∞ |
| Tversky loss | 0.8630 | 0.0203 | 0.4417 | ∞ | ∞ |
| Dice loss | 0.7125 | 0.0269 | 0.3697 | ∞ | ∞ |

**IV. Các task còn lại**

**1. Localize**

**V. Vẽ qualitative**

Mô hình được huấn luyện trong 300 epoches trên 2 tập FishLoc và FishSeg, được kiểm tra trên FishSeg và hình vẽ kết quả được minh họa trên FishLoc.

| Ảnh nhãn | FCN8  -ResNet50 | FCN8-VGG16 | FCN8-wide  -ResNet50 | Mô hình  đề xuất |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |
| --- |

| Ảnh nhãn | FCN8  -ResNet50 | FCN8-VGG16 | FCN8-wide  -ResNet50 | Mô hình  đề xuất |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |