International Journal of Computing and Digital Systems
2025, VOL. 17, NO.1, 1-12

http://dx.doi.org/10.12785/ijcds/1571000399

Incorporating Transfer Learning Strategy for improving Semantic Segmentation of Epizootic Ulcerative Syndrome Disease Using Deep Learning Model

Anbang[1] **and Gede Putra Kusuma**[2]

[1,2]*Computer Science Department, BINUS Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480*

Received 22 February 2024, Revised 20 October 2024, Accepted 25 October 2024

Abstract: Automated fish disease detection can eliminate the need for manual labor and provides earlier detection of fish disease such as EUS (Epizootic Ulcerative Syndrome) before it further spreads throughout the water. One of the problems that is faced on implementing a semantic segmentation fish disease detection system is the limited size of the semantic segmentation dataset. On the other hand, classification datasets for fish disease detections are more common and available in larger sizes, which cannot be used in segmentation tasks directly since it lacks the necessary label for such tasks. In this paper, we propose a training strategy based on transfer learning to learn from both ImageNet and classification dataset before being trained on the segmentation dataset. Specifically, we first train the ImageNet pre-trained VGG16 and ResNet50 on a classification task, then we transfer the weights into a semantic segmentation architectures such as U-Net and SegNet, and finally train the segmentation network on a segmentation task. We introduce two different modified U-Net architectures to allow the respective pre-trained VGG16 and ResNet50 weights to be transferred into the architecture. We used a classification dataset containing 304 images of fish diseases for classification task and a segmentation dataset containing 25 images of EUS-affected fishes for the segmentation task. The proposed training strategy is then compared with alternative training strategies such as training VGG16 and ResNet50 on ImageNet alone or classification dataset alone. When applied to SegNet and U-Net, the proposed training strategy surpasses their respective architecture trained on ImageNet or classification dataset alone. Between these two architectures with all compared training strategies, the U-Net+VGG16 architecture trained with our proposed training strategy achieves the best performance with validation and testing mIoU of 57.80% and 60.43%, respectively. The training code is available at https://github.com/RealOfficialTurf/FishDiseaseSegmentation.

Keywords: Fish Disease Detection, Semantic Segmentation, Transfer Learning, U-Net Model, SegNet Model

1. Introduction

Due to the geography of the country, fish has become one of the major sources of income in Indonesia. According to BPS, fish production has increased from 15.24-16.12 million tons in year 2017-2018 [1].

Like many other animals, most fish are susceptible to fish diseases. One of the diseases is EUS (Epizootic Ulcerative Syndrome), which is caused by *Aphanomyces invadans*. This disease is easily identified by the red spots or ulcers that appear on the fish body, hence this disease is also known as Red Spot disease. Fish that are affected by EUS will lose appetite, thus consuming less feed and growing slower. The affected fish may also die, with a mortality rate of around 20-80% [2]. Furthermore, EUS can spread from the infected fish to other healthy fishes in the same body of water. Therefore, early detection of fish disease can potentially prevent further infections and mortalities in the affected fishpond.

One conventional approach for detecting fish diseases is to manually monitor the fish. A person observes the fish in the tank and notices any anomaly visible on the fish skin. However, this approach is time-consuming and requires an individual that is capable of identifying fish diseases.

Computer vision is a study that tries to mimic the human's capability of recognizing images through the use of a computer. While traditional computer vision methods rely on algorithm selected to extract features from images such as edge detection, more recent computer vision methods employ deep learning neural network to automatically extract features from images, without the need to manually

E-mail address: anbang@binus.ac.id, inegara@binus.edu

select certain features.

There exists many works that are aimed at detecting fish diseases through image classification from hand-crafted features [3][4] and image classification using deep neural network [5][6][7]. There are also other works that also aims to detect fish disease through image segmentation [8][9], although image segmentation approaches are far less common than image classification approaches.

One of the problems faced by the previous work [8] was the limited sample size of the semantic segmentation dataset, which resulted in their model performing worse on testing data split. Semantic segmentation datasets are annotated with pixel-level labels. That is, the label is applied to each individual pixel by creating a mask of different classes in the image. Due to the complexity and level of expertise required to annotate pixel-level labels, they are more costly to annotate. Thus, they are less common and available in smaller sample sizes.

One common method used to overcome the dataset small sample sizes is to employ data augmentation. While data augmentation can increase sample sizes and reduce overfitting, the synthetic data created from data augmentation is limited in variety. As such, the performance improvement gained from data augmentation may be limited.

On the other hand, classification datasets are more common and available in larger sample sizes. In contrast to segmentation datasets, classification datasets are annotated with image-level labels. That is, the label is applied to the whole image rather than the individual pixels. image-level labels are easier to annotate than pixel-level labels. However, image-level datasets are incompatible with segmentation models which produce pixel-level outputs and cannot be directly used in segmentation tasks. Therefore, incorporating the features from the more abundant image-level datasets into segmentation models can be seen as a challenge.

An approach for a model to learn both image-level and pixel-level features is to build an architecture with one shared encoder and two outputs, one for classification, and another for segmentation. The shared encoder learns from input data by summing the losses from the respective label output. This approach is also known as Multi-task learning. Several works have proposed Multi-task learning architectures for classification and segmentation tasks [10][11][12]. Such architectures are more complicated to build.

On the other hand, there are several works that have applied transfer learning based on ImageNet for semantic segmentation problems [13][14][15]. Transfer learning is a method of reusing the knowledge on a different yet related domain to the target domain. This is done by transferring the weights from a pre-trained network to another compatible network. The network is then trained with the dataset related to the target domain.

The idea of this work stems from our observation that most previous works related to transfer learning and semantic segmentation problems make use of ImageNet pre-trained networks, which are classification networks on their own. By training the classification network on an image-level dataset to create a pre-trained network, it is possible to incorporate features from image-level datasets into segmentation models. The pre-trained network can then be transferred into a segmentation network to be trained on semantic segmentation dataset, allowing for better performance in segmentation tasks. Furthermore, ImageNet pre-trained network weights can be transferred into the classification model to also incorporate ImageNet features, which could further improve the model performance.

It should be noted that the idea of multi-step transfer learning is not completely new and has previously been explored by at least one study [16]. In their work, they proposed a two-stage transfer learning method, where an ImageNet pre-trained FCN-8 network is transferred into a FCN-8 network for training in a broad-scale segmentation dataset, then transferred into another FCN-8 network for training in narrow-scale segmentation dataset.

Therefore, this work proposes a training strategy based on two-stage transfer learning in the context of fish disease semantic segmentation to overcome the limited segmentation dataset sample size previously faced by [8]. Note that the difference between the proposed method used in this work and [16] lie in the tasks that the network is trained on. Our method trains the network on classification task in the first stage and segmentation task in the second stage, while their method trains the network on segmentation tasks in both stages. Following the previous work, we use the same fish EUS segmentation dataset as [8] and compare our results with their results. The main contributions made in this work are as follows:

1) A proposed training strategy involving training the network on classification data to overcome the scarce segmentation data. The proposed training strategy allows semantic segmentation models to be trained with both segmentation data and classification data without the need of a complicated architecture.
2) A comparison of the effectiveness between our proposed training strategy, which uses both ImageNet and fish disease classification dataset, and the alternative training strategies, which only uses either ImageNet alone or fish disease classification dataset alone.

The rest of the work is structured as follows. In Section 2, various related works for fish disease detection and transfer learning are presented. The proposed training strategy and network architectures used on this work are shown in Section 3 as well as the datasets used. Section 4 presents the comparison and results of various training strategies including our proposed training strategy. We discuss our

findings based on the results we presented in Section 5. Finally, we conclude our findings in Section 6.

2. Related Works

A. Fish Disease Detection

Chakravorty et al. [9] was one of the earliest that suggested the use of semantic segmentation to detect fish disease from fish images. Their work implemented a system to segment diseased areas in images of fish using PCA (Principal Component Analysis) and k-Means clustering. The dataset they use are collected from taking images from real fish and identified by human expert, but the sample size of the dataset is not described in their work. The results indicate that the implemented algorithm can work with above 90% accuracy. The authors suggested that more sophisticated approaches such as neural networks or SVM (Support Vector Machines) should be used for this problem. Continuing from their work, Rachman [8] proposed and compared four different deep learning semantic segmentation models with backbone networks for detecting EUS. They used a dataset of 26 images of fish that contracted EUS, which are augmented to increase the sample size to 1056 images. Although the performance was heavily affected by the limited dataset, the U-Net with ResNet50 backbone network achieved the highest test score out of the four tested models with an mIoU (mean Intersection over Union) of 59.33%.

Ahmed et al. [3] proposed a method of classifying infected salmon fish using SVM. Their work utilized statistical features and GLCM (Grey-Level Co-occurrence Matrix) as the features to be extracted from image from salmon, and SVM to classify the image based on the extracted features. They used a dataset of 266 images containing infected and healthy salmon fishes, which are augmented to increase the sample size into 1326 images. The highest accuracy obtained by the SVM classifier is 94.12% with Area Under ROC (Receiver Operating Characteristic) Curve value of 96.71%. Similarly, Mia et al. [4] proposed an expert system based on feature extraction of statistical feature and GLCM to detect fish diseases. Unlike the previous work, their work experimented with eight different machine learning algorithms, among of them are SVM. Their work also uses 10-fold cross-validation technique to evaluate the models. They used a dataset of 485 images containing fish with EUS and Tail Fin Rot diseases as well as healthy fishes. The Random Forest model performed the best with 88.87% accuracy and an Area Under Curve of 89.71%, while SVM achieved 77.04% accuracy with an Area Under Curve of 78.25%.

Waleed [17] proposed a system based on Raspberry Pi to recognize different fish diseases using the deep learning approach. Their work experimented with training and testing combinations between four different CNN (Convolutional Neural Network) models and three different color spaces. They used a dataset of 45 images collected online, which are preprocessed and augmented to increase the sample size into 2400 images. The best performing combination of model and colorspace, AlexNet with XYZ color space, achieved 99.04% accuracy in recognizing the disease of the diseased fish. Gupta et al. [5] proposed a modified CNN model based on VGG16 for classifying lice and wound on salmon fish. They captured 239 images from live fishes in a fish tank to create their dataset, which are then preprocessed and augmented to 3289 images. The proposed CNN model, VGG16, and VGG19 are trained and tested in their work. The proposed CNN model reached 96.7% accuracy in detecting salmon fish diseases, which is 3.89% and 5.5% higher than VGG19 and VGG16, respectively. Y.P. Huang and Khabusi [6] proposes an architecture based on attention mechanism, multilayer fusion, and online sequential extreme learning machine to classify five different fish diseases. Their work experimented with training and testing various CNN models as well as their proposed models. The dataset used in this work is composed of 649 images collected online, which are preprocessed and augmented to 5165 images. Their proposed architecture reached 94.28% accuracy in classifying various fish diseases, which is 2.24% higher than the second best performing model, VGG19.

B. Transfer Learning

Pravitasari et al. [13] proposed the UNet-VGG16 architecture and applied the transfer learning method for segmentation of brain tumor. The proposed architecture UNet-VGG16 was modified so that the encoder resembles the VGG16 network and the decoder to match the encoder layers. Imad et al. [15] proposes the use of transfer learning to semantically segment 3D objects from LiDAR (Light Detection and Ranging) data. The proposed network, which shape was inspired by the U-Net architecture, receives the pre-trained MobileNetV2 weights before being trained on pre-processed bird-eye view images from raw point clouds. Sakurai et.al. [16] proposed a two-step transfer learning for semantic segmentation of plants. The FCN-8s network was first pre-trained on ImageNet, then trained on a broader plant dataset, and lastly trained on a narrower plant dataset.

3. Materials and Methods

A. Proposed Training Strategy

The training strategy we propose is as follows. First, we obtain the ImageNet pre-trained network. As the network was previously trained on a different dataset, the number of classes on the network output will differ from the number of classes on our dataset. Therefore, the classifier layer in the pre-trained network is replaced with a new classifier layer with the matching classes in our classification dataset. The pre-trained network is then trained on the classification task with the fish disease classification dataset until the validation loss is at minimum. The pre-trained weights on the network are not frozen and allowed to change during training.

After the pre-trained network has been trained on the classification dataset, we transfer the network weights from the pre-trained network into the compatible layers of the

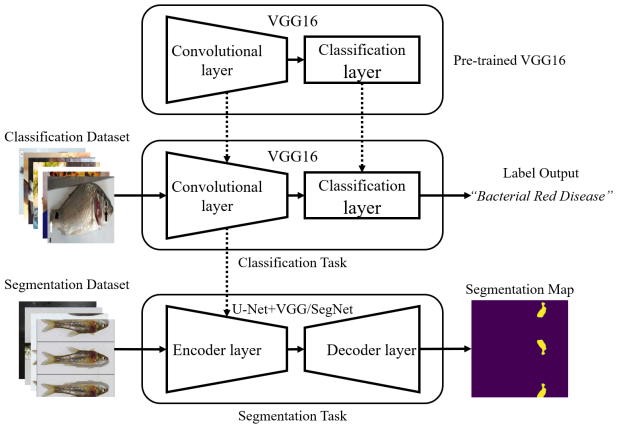

Figure 1. Our proposed training strategy.

segmentation network. This segmentation network is then trained on a segmentation task with the segmentation dataset until the validation loss is at minimum. The transferred weights on the network are not frozen and allowed to change during training. Figure 1 shows the process of our training strategy.

B. Network Architectures

To evaluate our proposed training strategy, we chose VGG16 as the classification network along with SegNet and U-Net to be evaluated as the segmentation networks. The VGG16 network was primarily chosen due to the network being easy to adapt and transfer into the chosen segmentation networks. Additionally, we evaluated ResNet50 as the classification network with U-Net to further test the proposed training strategy on different networks.

The U-Net architecture is composed of two sections, the encoder layers, and the decoder layers. Except for the last encoder layer, the encoder layers are connected to the respective decoder layers using skip connections. However, the VGG16 weights cannot be directly transferred into the U-Net architecture as the encoder layers do not match the VGG16 network. Therefore, we modify the encoder layers to adapt the VGG16 convolutional layers. This modified architecture will be referred to as U-Net+VGG throughout this paper. Unlike the architecture proposed in [13], we do not modify the decoder layers to match the encoder layers. The U-Net+VGG architecture is illustrated in Figure 2. In the figure, the number above the boxes indicate the number of channels of each layer in the same block, while the number at the leftmost side indicates the layer width and height for all the layers in the same row.

The ResNet50 network modification of U-Net is also tested. To adapt the encoder, we forward the outputs of the first convolutional layer and residual blocks into the respective skip connections. Due to the stride convolution in the first convolution layer in the ResNet50 network, the output width and height of the first skip connection is halved, resulting in the decoder layers outputting smaller output than the input image. In order to match the output size with the input size, we upsample the output with a linear upsampler to double the width and height.

The SegNet architecture resembles the U-Net architecture with the main difference in the skip connection, which only transfers max-pool indices into the respective decoder layers rather than feature maps [18]. Another difference of SegNet architecture is the use of VGG16 and inverted VGG16 networks in both encoder and decoder layers, respectively. Since the encoder layers of the SegNet architecture already matches with the VGG16 network, no modifications were necessary. We directly transfer the VGG16 weights into the encoder layers.

All architectures are adjusted to receive images with an input size of 448×448 to preserve the pixel sizing between datasets. Batch normalization is applied after every convolutional layer and before the ReLU (Rectified Linear Unit) layer.

C. Datasets

1) Classification Dataset

The classification dataset is a fish disease classification dataset obtained from Kaggle [19]. This dataset contains 460 images of fish that contracted various fish diseases, as well as pictures of healthy fish. The images in this dataset are labeled by being put into different folders, where each folder represents a class, and the folder name represents its class name. The dataset contains seven classes, one class of it represents healthy fish and the rest of the classes representing six different fish diseases.

We discovered that the dataset contains several duplicated images within the same classes and between different classes. Additionally, the dataset overlaps with the segmentation dataset. To prevent leaking testing data to the model, image deduplication is performed. We utilize a Python library to easily detect potentially duplicate images within the dataset and the segmentation dataset. The detected potentially duplicate images are then manually selected by a human to eliminate the possibility of false positives. The filtered duplicate images are removed from the dataset, prioritizing eliminating duplicate images from the largest class. This image deduplication process resulted in 304 images left in the dataset. The deduplicated dataset is split into training data, validating data, and testing data. Table I shows the composition of each data splits produced from the deduplicated dataset.

To match the segmentation network's input size, image pre-processing was applied to the entire dataset by resizing and stretching every image to 448×448 pixels.

2) Segmentation Dataset

The segmentation dataset is a fish EUS segmentation dataset obtained from Roboflow [20] that were used on the previous work [8]. This dataset contains 26 images of fish that contracted EUS. The images in this dataset are labeled

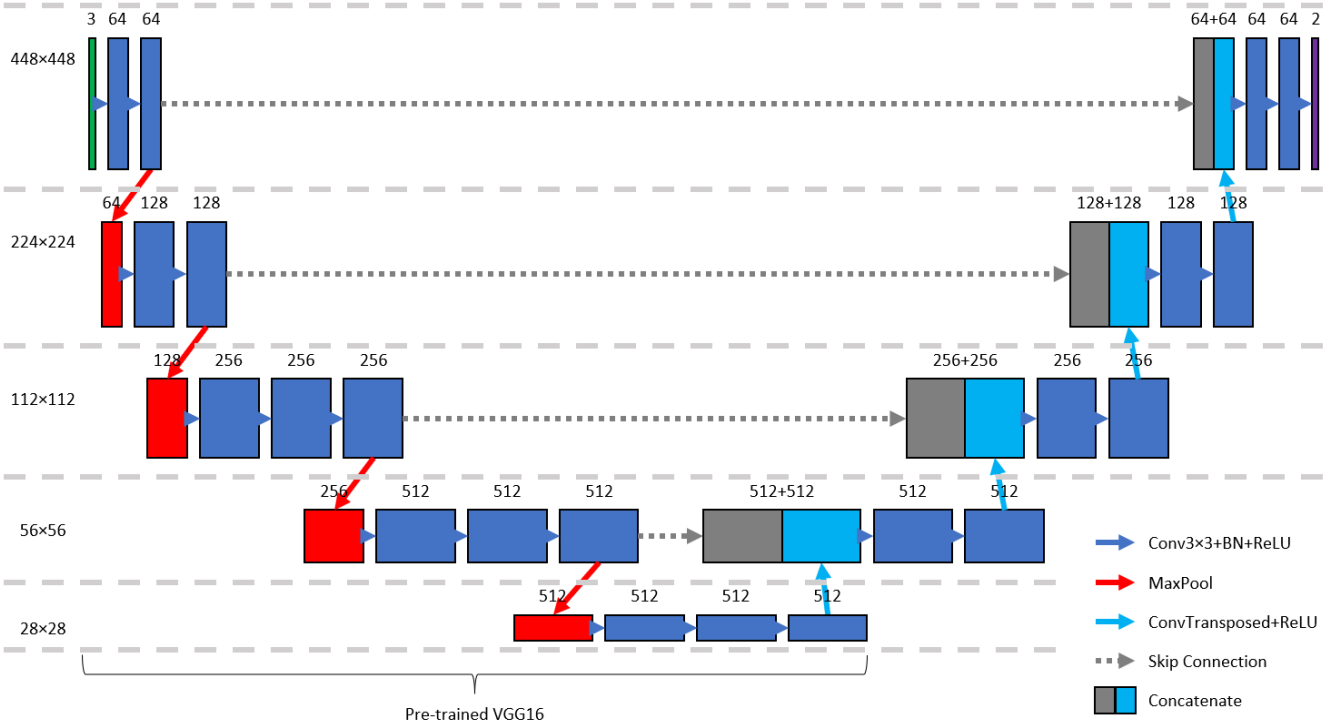

Figure 2. The U-Net+VGG architecture

TABLE I. The classification dataset composition

Class	Total	Train	Test	Valid
Bacterial diseases - Aeromoniasis	33	23	5	5
Bacterial gill disease	29	20	5	4
Bacterial Red disease	22	15	4	3
Fungal diseases . Saprolegniasis	34	24	5	5
Healthy Fish	128	90	19	19
Parasitic diseases	30	21	5	4
Viral diseases White tail disease	28	20	4	4
Total	304	213	47	44

by their respective segmentation masks that are included in the dataset. The dataset contains one class of EUS disease and background class, totaling in two classes.

Similar to the classification dataset, we also deduplicate the image using the same method applied to the classification dataset, resulting in 25 images left in the dataset. The dataset is then split into training data, validating data, and testing data containing 15, 5, and 5 images respectively.

The segmentation dataset contains images with varying sizes. The simple method of stretching the image to fit might skew the segmentation dataset in favor of images with wider aspect ratio and thus cannot be used. Instead, the images are pre-processed by rescaling the image equally in width and height, while scaling the image to have the same new width. We chose the new width to be fixed at the median width of the segmentation dataset of 374 pixels.

Next, we employ a method to extend the height of the image to increase the number of features in the image. The method involves extending the original image by mirroring. The segmentation masks are also extended the same way as the image, increasing the mask area. This method is only applied to the training data, as validation and testing data should be kept unaltered. Therefore, for validation and testing image data, we employ a different method in which the image is extended by padding with black pixels. To further illustrate the method used, an example is given in Figure 3.

The resulting image size of this dataset is 374×374 pixels. To fit the image with the model input size, we further resize and stretch the images to 448×448.

D. Data Augmentation

During the training process in both classification and segmentation tasks, image augmentations are applied to images right before being fed into the network. The image augmentations are only applied to the training data splits. The validation and testing data splits do not receive any image augmentation at all. Image augmentations are applied to all training strategies equally.

Following the previous work of Rachman [8], we adopt their augmentation methods and improve upon these

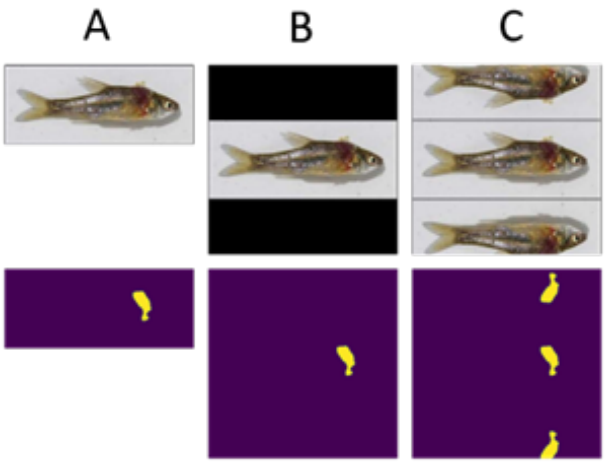

Figure 3. An example of the image processing method used for squaring images. A: Original image and mask. B: Padding with black pixels. C: Mirroring the original image

methods. We introduce batch-level augmentations such as CutMix [21], MixUp [22], and Mosaic [23], as we found out during testing that these augmentation techniques have an effect in improving the model performance. Batch-level augmentation techniques differ from usual augmentation techniques in that they are applied to a batch of images rather than single images. This allows for the augmented image to contain other images within the batch, increasing the variety of synthetic data. The image augmentations used for this work are as follows, in the order of application.

1) Random choice. We apply different augmentation methods depending on the tasks. In classification tasks, the image is applied one of the following augmentations with equal probability: CutMix, MixUp, or none. In segmentation tasks, the image is applied either a Mosaic augmentation or no augmentation, also with equal probability.
2) Gaussian noise. The image is added by a noise image generated from zero-mean gaussian noise with standard deviation of 0.0125.
3) Blur. The image is applied a gaussian blur with a random kernel size of 9 px, 5 px, or 0 px (no blurring).
4) Horizontal and Vertical flipping. The image is flipped on the horizontal and vertical axis, with equal flipping probability on each axis.
5) Color shifting. The brightness and saturation of the image is shifted by a random value between -25% and 25%.
6) Elastic deformations. The image is deformed by warping the image elastically [24].
7) Random rotation. The image is rotated to a random value between -15° to 15°.

E. Evaluation

To help in evaluating the proposed training strategy, we first introduce the training strategies used. The training strategy used to train a model is indicated by a suffix in the model's name. The four suffixes and the training strategies associated are as follows.

1) *No suffix*: The model is initialized with random weights and trained.
2) -I: The model is loaded with the weights from the respective classification model trained with ImageNet-1K and trained.
3) -C: The model is loaded with the weights from the respective classification model with no suffix and trained.
4) -IC: The model is loaded with the weights from the respective classification model with -P suffix and trained. This is our proposed training strategy.

We also introduce and train classification networks that will be used for transfer learning. The networks are as follows.

1) VGG16: VGG16 network initialized with random weights and trained on classification dataset.
2) VGG16-P: VGG16 network initialized with ImageNet-1K VGG16 weights and trained on classification dataset.
3) ResNet50: ResNet50 network initialized with random weights and trained on classification dataset.
4) ResNet50-P: ResNet50 network initialized with ImageNet-1K ResNet50 weights and trained on classification dataset.

Lastly, we experiment on different semantic segmentation architectures to see the effects of our proposed training strategy on different architectures. The semantic segmentation architectures that will be evaluated in this work are as follows.

1) U-Net: Unmodified U-Net architecture.
2) U-Net+VGG16: Modified U-Net architecture with encoder layers replaced by VGG16.
3) U-Net+ResNet50: Modified U-Net architecture with encoder layers replaced by ResNet50.
4) SegNet: Unmodified SegNet architecture, with encoder layers already containing VGG16.

To measure the improvement between training strategies, we use mIoU as a metric. The metric mIoU is widely used for evaluating the performance of semantic segmentation models. The mIoU is calculated by taking the statistical mean of all the IoU (Intersection over Union) values across all the classes (excluding the background class) and across all the images in the data split. Note that since there is only one non-background class in our dataset, we only compute the mean across all the images. The IoU of a class of an image is computed by the following equation,

$$IoU = \frac{P \cap G}{P \cup G} = \frac{TP}{TP + FP + FN} \quad (1)$$

Where P is the predicted pixels in the image that belong to the class, and G is the ground truth pixels in the image that belong to the class. TP, FP, and FN stand for true positive, false positive, and false negative, respectively.

4. Results

A. Experimental Setup

The training setup was implemented using PyTorch 2.3.1 library and written in Python programming language. The pre-trained VGG16 and ResNet50 models were obtained from PyTorch Hub. The data augmentation pipeline was implemented with TorchVision library that comes with PyTorch itself. The training is performed on a single Nvidia RTX 3080 GPU with 10 GB of Video RAM (Random Access Memory).

All random weights are initialized using Kaiming initialization with normal distribution [25]. Each training strategy was repeated 8 times with the same data split distributions, producing 8 models per training strategy. Out of the 8 models produced, the model with the best validation mIoU score is chosen to represent the result for the training strategy. The chosen models are then checked with testing data to produce the testing mIoU scores, which shows the actual model performance on unseen data.

B. Classification Task

We train all four classification networks with the SGD (Stochastic Gradient Descent) optimizer with 0.9 momentum. The learning rate is fixed at 0.0001 with no weight decay for both VGG16 and VGG16-P. From testing, we observed that ResNet trained better with weight decay. Therefore, for ResNet50 and ResNet50-P, the learning rate is fixed at 0.001 and weight decay is set at 0.0001. The batch size for training is set at 8 to allow for training to fit entirely in GPU. Weighted cross-entropy loss was used for calculating losses, with each loss weight for each class set to the mean class output frequency divided by individual class output frequency. This is done to address the class imbalance over-representing the healthy fish class in the classification dataset. Additionally, we apply an early stopping technique in which the networks are trained until there are no improvements in validation loss for the last 100 epochs. For both VGG16-P and ResNet50-P, the networks are trained until there are no validation loss improvements in the last 10 epochs instead of 100 epochs, since both networks have been pre-trained and converged quicker. Table II shows the performance results of VGG16 networks on the classification dataset.

C. Segmentation Task

Similar to the classification task, we use SGD optimizer to train all the segmentation networks. We base the learning rate and momentum from the paper [24] and search for the optimal learning rate. We found that a fixed learning rate of 0.0003 and momentum of 0.99 is optimal for training U-Net+VGG16 model, and so we apply these training hyperparameters for U-Net, U-Net+VGG16, and U-Net+ResNet50. We do not include weight decay in training, as from our testing we discovered that adding weight decay causes the model training to worsen. For SegNet, we base the learning rate and momentum from the paper [18] and search for the optimal learning rate. We found that a fixed learning rate of 0.01 and momentum of 0.9 is optimal for training SegNet. In order to eliminate variations in training, batch size is set to the highest number that accommodates the largest model while still allowing for training to fit entirely in GPU. We set the batch size to 5 for all models. Focal Tversky loss was used for calculating losses, as we found out that training with Focal Tversky loss gives better result than training with Dice loss. We also apply early stopping technique to segmentation task as well, in which the training strategies are trained until there are no improvements in validation loss for the last 50 epochs.

Table III shows the performance results for each training strategy employed at the end of training, with the performance calculated at the epoch when the model achieves its lowest validation score. The number of epoch when the model achieved its lowest validation mIoU score is shown in the table.

TABLE II. Accuracy scores of classification networks tested across all data splits in the classification dataset

Classification Networks	Epoch	Train	Valid	Test
VGG16	87	68.08	34.04	45.45
VGG16-P	73	88.26	61.70	61.36
ResNet50	40	39.91	34.04	40.91
ResNet50-P	50	98.51	68.09	63.64

TABLE III. mIoU scores of segmentation networks tested across all data splits in the segmentation dataset

Segmentation Networks	Epoch	Train	Valid	Test
U-Net	305	49.39	48.13	52.77
U-Net+VGG16	159	46.59	46.74	48.92
U-Net+VGG16-C	265	49.67	46.26	52.87
U-Net+VGG16-I	206	62.68	**59.99**	57.89
U-Net+VGG16-IC	197	**63.76**	57.80	**60.43**
U-Net+ResNet50	265	51.46	46.91	43.69
U-Net+ResNet50-C	506	48.95	47.53	46.13
U-Net+ResNet50-I	458	**54.62**	50.42	52.28
U-Net+ResNet50-IC	172	51.52	**50.93**	**54.48**
SegNet	215	68.69	48.51	51.26
SegNet-C	243	63.79	44.38	51.39
SegNet-I	264	**74.98**	**61.73**	58.22
SegNet-IC	252	73.91	61.25	**58.71**

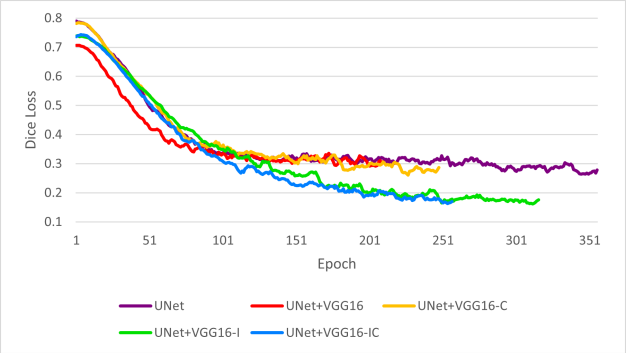

Figure 4. Training loss curves of the U-Net+VGG16 training strategies

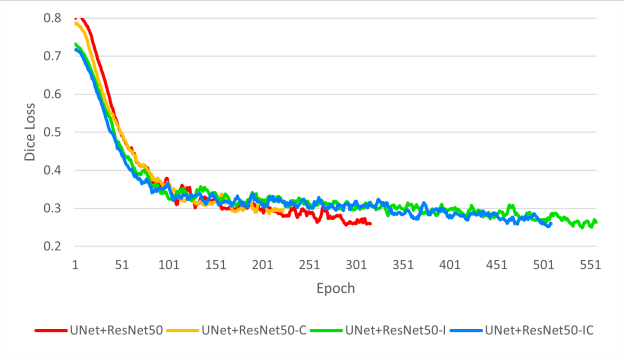

Figure 5. Training loss curves of the U-Net+ResNet50 training strategies

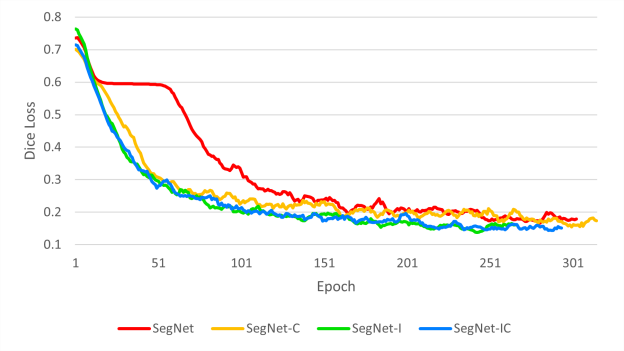

Figure 6. Training loss curves of the SegNet training strategies

From the results shown, all the models with the -IC prefix achieved the highest testing mIoU score in their respective architectures. The U-Net+VGG16-IC performed the best among other architectures with a testing mIou score of 60.43%. SegNet-IC performed very well and achieved second in testing with a mIoU score of 58.71%. While the U-Net+ResNet50-IC did not perform as good as most of the other models, it still manages to outperform the other model in its architecture, with a testing mIoU score of 54.48%. This suggests that our proposed training strategy, indicated with the -IC prefix, can improve the model's performance.

We show the training loss curves for each training strategy in their respective architecture to show how the training progresses in each training strategy. For clarity, all the training loss curves we show in this section are smoothed using a moving average calculated from the training loss scores from the 6 last epochs. The training loss curve shows how the loss of the model converges. Note that some models have curves that end at shorter epochs as these models were cut short in their training due to the early stopping technique we applied.

Figure 4 shows the training loss curves of various training strategies applied to the U-Net+VGG16 architecture. The curve clearly shows that U-Net+VGG16-I and U-Net+VGG16-IC converges the best among the other training strategies. Between these two training strategies, U-Net+VGG16-IC converges slightly better than U-Net+VGG16-I. This might suggest that the additional training from classification data causes the training to converge better.

Figure 5 shows the training loss curves of various training strategies applied to the U-Net+ResNet50 architecture. It appears that U-Net+ResNet50 converges faster than the other training strategies, despite achieving the worst testing score. We suspect this might be caused by the model overfitting due to the scarcity of segmentation data. U-Net+ResNet50-C also converges as fast as U-Net+ResNet50 but only performs slightly better, possibly due to the same reason as U-Net+ResNet50. Lastly, U-Net+ResNet50-IC converges slightly better and has a better testing score than U-Net+ResNet50-I.

Figure 6 shows the training loss curves of various training strategies applied to the SegNet architecture. SegNet training progress appears to have plateaued initially as the model was having difficulty in converging before it eventually started converging normally. While SegNet-IC performed slightly better than SegNet-IC, both models appear to converge nearly the same.

To further validate the results, we also show two difficult segmentation cases, image A and B, in Figure 7. The areas marked in yellow show the EUS-diseased area, while the background class is marked in purple. As seen in the segmentation masks, some of the models, most notably U-Net+VGG16 and U-Net+ResNet50, performed poorly and produced splotchy segmentation masks as these models struggle to differentiate between the fish skin features and the fish ulcers. However, the models prefixed with -I and -IC performed well and produced cleaner masks. The models prefixed with -IC performed better than the -I models in their respective architecture and produced even cleaner segmentation masks.

5. Discussion

We trained VGG16 and ResNet50 classification networks on a fish disease classification dataset to acquire the

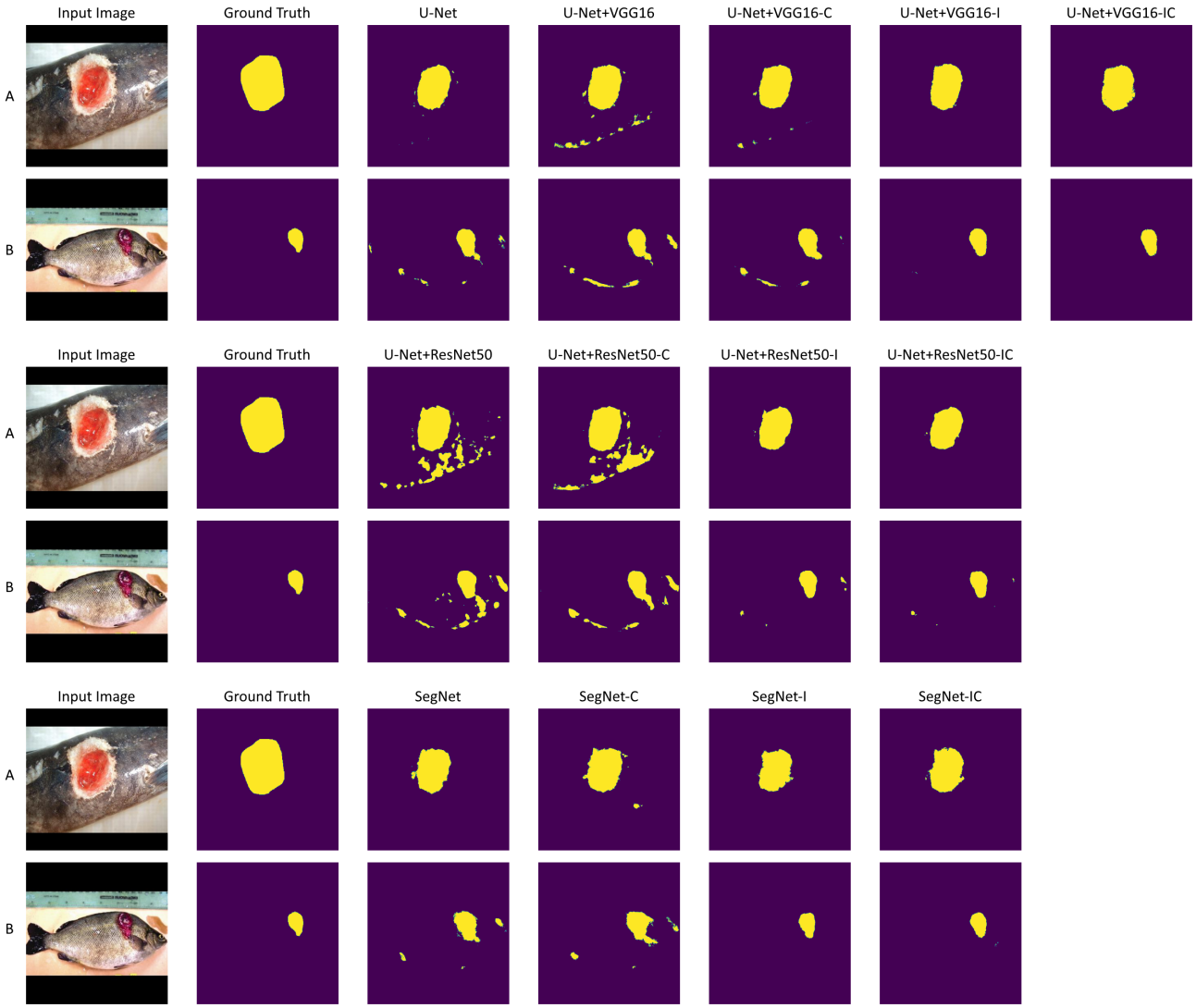

Figure 7. Comparison of segmentation maps produced by each training strategies

weights for testing different training strategies. We used U-Net, U-Net with VGG16, U-Net with ResNet50, and SegNet architectures for segmentation network. We then formulate 13 different training strategies, with varying methods of training, to compare between our proposed training method with alternative training methods. The training strategies are trained and compared their training results to each other training strategies. We showed the training loss curves for different strategies to see how the model loss converges during training. We showed the mIoU scores of each training strategy in three different data splits.

The modified U-Net with VGG architecture achieves a slightly worse performance compared to the unmodified U-Net architecture, only achieving a testing mIoU score of 48.92%. The convolutional layers of VGG16 are closely similar to the encoder layers in U-Net, with the only differences being that an extra block is added to layer 3 to 5, and the number of channels in the final layer being half compared to the U-Net ones. This might suggest that the reduction in the number of channels in the final layer has negatively affected the performance of U-Net+VGG16.

The training strategy that we propose, which makes use of both ImageNet and classification dataset, are also compared with alternative strategies which only make use of either ImageNet or classification dataset. This was done to confirm our findings that the inclusion of both ImageNet and classification dataset improves the performance of the network more than either ImageNet or classification dataset alone.

TABLE IV. mIoU scores comparison between our work and Rachman's work

Segmentation Networks	Epoch	Train	Valid	Test
U-Net	305	49.39	48.13	52.77
U-Net+VGG16	159	46.59	46.74	48.92
U-Net+VGG16-C	265	49.67	46.26	52.87
U-Net+VGG16-I	206	62.68	**59.99**	57.89
U-Net+VGG16-IC	197	**63.76**	57.80	**60.43**
U-Net+ResNet50	265	51.46	46.91	43.69
U-Net+ResNet50-C	506	48.95	47.53	46.13
U-Net+ResNet50-I	458	**54.62**	50.42	52.28
U-Net+ResNet50-IC	172	51.52	**50.93**	**54.48**
SegNet	215	68.69	48.51	51.26
SegNet-C	243	63.79	44.38	51.39
SegNet-I	264	**74.98**	**61.73**	58.22
SegNet-IC	252	73.91	61.25	**58.71**
fcn_32_mobilenet	-	87.28	57.97	52.82
resnet50_unet	-	89.64	59.74	**59.33**
mobilenet_segnet	-	93.77	**65.35**	53.69
mobilenet_unet	-	**98.75**	62.33	55.33

We evaluated the training strategies on both U-Net and SegNet to study the effectiveness of our training strategy in two different semantic segmentation architectures. Furthermore, we also applied our proposed training strategy onto a different classification network such as ResNet to study the effectiveness of our training strategy on a different network. As we have shown in the results section, the models that were trained with our proposed training strategy achieved the highest testing mIoU score in their respective architectures. This suggests that the proposed training strategy can be applied to various classification networks and semantic segmentation architectures to improve the model performance.

We compare our model results with [8] in Table IV. Both our work and their work uses the same segmentation dataset in [20]. In their work, the dataset was split from 26 images into 20, 4, and 2 images before various augmentation methods were used to increase the number of images into 960, 64, and 32 images for training data, validating data, and testing data respectively. In contrast to our work, we deduplicate the dataset from 26 images into 25 images before splitting it, and data augmentations are only applied to the training data split during training phase instead of being applied to all data splits during image pre-processing phase. The reason that we do not apply augmentation in the validation and testing data is because the models should be evaluated on natural data and not synthetic data. We also split the dataset in favor of the validation and testing data splits, resulting in 15, 5, and 5 images for training data, validating data, and testing data, respectively.

As seen on the comparison table, their best performing model is resnet50_unet with a testing mIoU score of 59.33%, which has a similar architecture to ours named U-Net+ResNet50. However, even with our proposed training strategy, our model performs worse than theirs. We believe that this might be caused by the difference in the method of sampling and splitting the dataset between both their and our works. Their data splits are composed of 20, 4, and 2 images while ours are composed of 15, 5, and 5 images for training data, validating data, and testing data, respectively. Furthermore, our work also makes use of data deduplication, pre-processing the image with aspect ratios preserved, and data augmentation applied only on training data rather than the entire dataset. Despite the difference between the methods used on both works, our model U-Net+VGG16-IC managed to perform better than their best model resnet50_unet with a testing mIoU score of 60.43%.

Some of our models such as U-Net+VGG16-C, U-Net+VGG16 and U-Net appear to have testing scores that are higher than their training scores. While it is abnormal for a model to achieve higher scores in testing than in training, we do not think that this anomaly is caused by a fluke, as we repeated each training strategy 8 times and found the score to be consistent between repetitions. Instead, we suspect that this is caused by the poor distribution of our dataset, which was caused by the limited sample size of the dataset. Despite our best efforts, we were unable to procure larger fish disease semantic segmentation public datasets online, as the other datasets found online appear to be either poorly annotated or lacking documentation. Nevertheless, the other models that we evaluated appear normal and achieve reasonable training and testing scores.

This work has shown that our proposed training strategy, which was closely based on the two-stage transfer learning method by [16], is effective in improving the performance of the fish disease semantic segmentation model. Despite the limited number of samples in segmentation dataset this work is facing, our proposed training strategy that makes use of both pre-trained network and classification dataset still managed to perform better than alternative training strategies that only either make use of pre-trained network or classification dataset. We have also tested the training strategies on different architectures, and the results also shows that each architecture trained with our proposed training strategy performed better than alternative training strategies. Therefore, our work suggests that the two-stage transfer learning method can be more effective than the conventional one-stage transfer learning method, even when applied at various architectures.

6. Conclusion and Future work

We have proposed a training strategy based on transfer learning to improve the semantic fish EUS segmentation. The proposed training strategy involves training the classification network on both ImageNet and classification dataset before being transferred into the segmentation architecture. To evaluate the proposed training strategy, we chose VGG16 and ResNet50 as the classification network along with U-

Net and SegNet as the segmentation architectures. Two datasets were used for the experiment, one for classification task, and another for segmentation task. We modified the architecture to include a VGG16 and ResNet50 encoder layer in order for the pre-trained weights to be transferred. To compare and show the effectiveness of the proposed training strategy, we train the proposed training strategy along with the alternative training strategies.

The results between the training strategies are then compared. The models that were trained with our proposed training strategy performed better than the alternative training strategies in their respective architecture classes, which shows the effectiveness of our proposed training strategy. The best performing training strategy is U-Net+VGG16-IC with validation and testing mIoU score of 57.80% and 60.43%, respectively.

In the future, more recent semantic segmentation models can be researched to improve the segmentation quality of fish diseases. We will also look into exploring object detection networks and region-level labeled datasets that are less difficult to annotate than pixel-level labeled datasets.

References

[1] BPS, "Produksi perikanan budidaya menurut komoditas utama," Tech. Rep., 2022.

[2] L. O. Haslan, *Buku Saku Hama dan Penyakit Ikan*, 2017.

[3] M. S. Ahmed, T. T. Aurpa, and M. A. K. Azad, "Fish disease detection using image based machine learning technique in aquaculture," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 5170–5182, 9 2022.

[4] M. J. Mia, B. R. Mahmud, M. S. Sadad, A. H. Asad, and R. Hossain, "An in-depth automated approach for fish disease recognition," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7174–7183, 10 2022.

[5] A. Gupta, E. Bringsdal, K. M. Knausgård, and M. Goodwin, "Accurate wound and lice detection in atlantic salmon fish using a convolutional neural network," *Fishes*, vol. 7, no. 6, p. 345, 11 2022.

[6] Y.-P. Huang and S. P. Khabusi, "A cnn-oselm multi-layer fusion network with attention mechanism for fish disease recognition in aquaculture," *IEEE Access*, vol. 11, no. June, pp. 58 729–58 744, 2023.

[7] N. Hasan, S. Ibrahim, and A. Azlan, "Fish diseases detection using convolutional neural network (cnn)," *Int. J. Nonlinear Anal. Appl*, vol. 13, no. November 2021, pp. 2008–6822, 2022.

[8] F. Rachman, M. Akbar, and E. Putera, "Fish disease detection of epizootic ulcerative syndrome using deep learning image processing technique," vol. 8, no. 1. The International Institute of Knowledge Management, 1 2023, pp. 23–34.

[9] H. Chakravorty and R. Paul, "Image processing technique to detect fish disease," *International Journal of Computer Science & Security*, vol. 9, no. 2, pp. 121–131, 2015.

[10] A. Amyar, R. Modzelewski, H. Li, and S. Ruan, "Multi-task deep learning based ct imaging analysis for covid-19 pneumonia: Classification and segmentation," *Computers in Biology and Medicine*, vol. 126, no. October, p. 104037, 11 2020.

[11] S. Hong, J. Oh, H. Lee, and B. Han, "Learning transferrable knowledge for semantic segmentation with deep convolutional neural network," vol. 2016-Decem. IEEE, 6 2016, pp. 3204–3212.

[12] Y. Sun, S. Liao, C. Gao, C. Xie, F. Yang, Y. Zhao, and A. Sagata, "Weakly supervised instance segmentation based on two-stage transfer learning," *IEEE Access*, vol. 8, pp. 24 135–24 144, 2020.

[13] A. A. Pravitasari, N. Iriawan, M. Almuhayar, T. Azmi, I. Irhamah, K. Fithriasari, S. W. Purnami, and W. Ferriastuti, "Unet-vgg16 with transfer learning for mri-based brain tumor segmentation," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 3, p. 1310, 6 2020.

[14] S. Baressi Šegota, I. Lorencin, K. Smolić, N. Anđelić, D. Markić, V. Mrzljak, D. Štifanić, J. Musulin, J. Španjol, and Z. Car, "Semantic segmentation of urinary bladder cancer masses from ct images: A transfer learning approach," *Biology*, vol. 10, no. 11, p. 1134, 11 2021.

[15] M. Imad, O. Doukhi, and D.-J. Lee, "Transfer learning based semantic segmentation for 3d object detection from point cloud," *Sensors*, vol. 21, no. 12, p. 3964, 6 2021.

[16] S. Sakurai, H. Uchiyama, A. Shimada, D. Arita, and R.-i. Taniguchi, "Two-step transfer learning for semantic plant segmentation," vol. 2018-Janua, no. Icpram. SCITEPRESS - Science and Technology Publications, 2018, pp. 332–339.

[17] A. Waleed, H. Medhat, M. Esmail, K. Osama, R. Samy, and T. M. Ghanim, "Automatic recognition of fish diseases in fish farms." IEEE, 12 2019, pp. 201–206.

[18] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 11 2015.

[19] S. Biswas, "Freshwater fish disease aquaculture in south asia dataset," 2022.

[20] Roboflow, "Fish disease 2 dataset," *Roboflow Universe*, 8 2022, visited on 2024-02-21.

[21] S. Yun, D. Han, S. Chun, S. J. Oh, Y. Yoo, and J. Choe, "Cutmix: Regularization strategy to train strong classifiers with localizable features," vol. 2019-Octob. IEEE, 10 2019, pp. 6022–6031.

[22] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, pp. 1–13, 10 2017.

[23] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 4 2020.

[24] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *IEEE Access*, vol. 9, pp. 16 591–16 603, 5 2015.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers:

Surpassing human-level performance on imagenet classification," vol. 2015 Inter. IEEE, 12 2015, pp. 1026–1034.