FISHIAL.AI

ENABLING FISHIAL RECOGNITION™

Advancements in Fish Segmentation: A Mask R-CNN Approach

Utilizing Point Rend Mask Head for Enhanced Precision

Traditional Approaches

1. **Manual Annotation** - Traditional methods often relied on manual annotation, where experts manually delineated fish boundaries in images.
    a. Pros: Provides high-quality annotations tailored to specific research needs.
    b. Cons: Time-consuming, labor-intensive, and not scalable to large datasets. Subject to human error and inter-annotator variability.

2. **Basic Computer Vision Techniques** - Basic computer vision techniques such as thresholding, edge detection, and morphological operations were commonly used for fish segmentation.
    a. Pros: Simplicity and ease of implementation.
    b. Cons: Time-consuming, labor-intensive, and not scalable to large datasets. Subject to human error and inter-annotator variability.

3. **Semi-Automated Algorithms GrabCut** - GrabCut's main goal is to separate foreground objects from the background of an image, but it also requires human interaction to refine the object and background regions.

Dataset

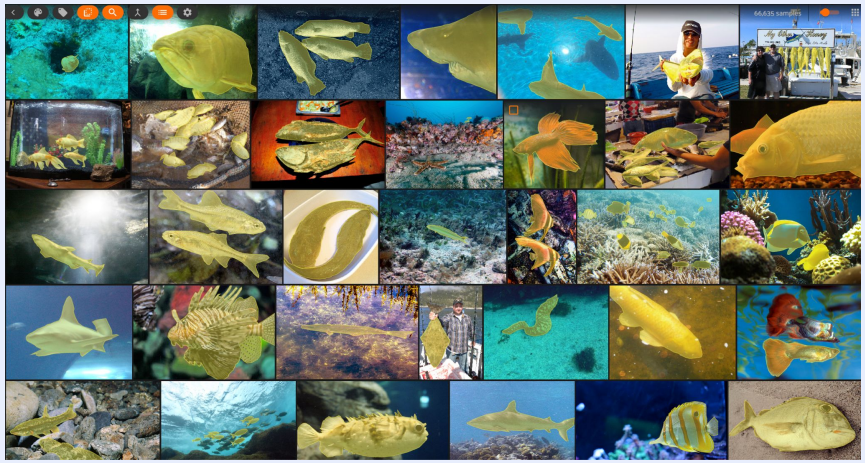
The latest version V8 of the segmentation training dataset includes:

1. 98658 Total train images
2. 6135 Total validation images

The latest version of the classification and training dataset includes:

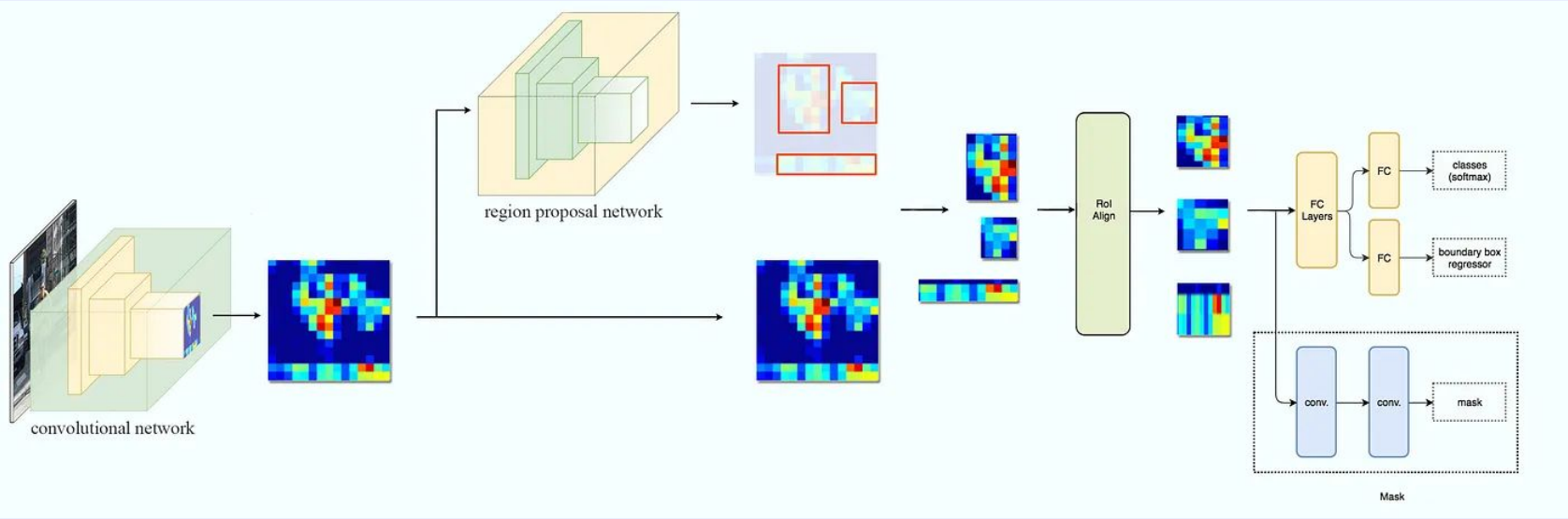
1. 52669 Total train images.
2. 17321 Total validation images.
3. 427 Number of different classes.
4. 250 Maximum number of images for one class (Sphyraena barracuda).
5. 40 Minimum number of images for one class (Opisthonema oglinum)

Dataset



Example images in the classification and segmentation dataset

Introduction to Mask R-CNN (Deprecated)

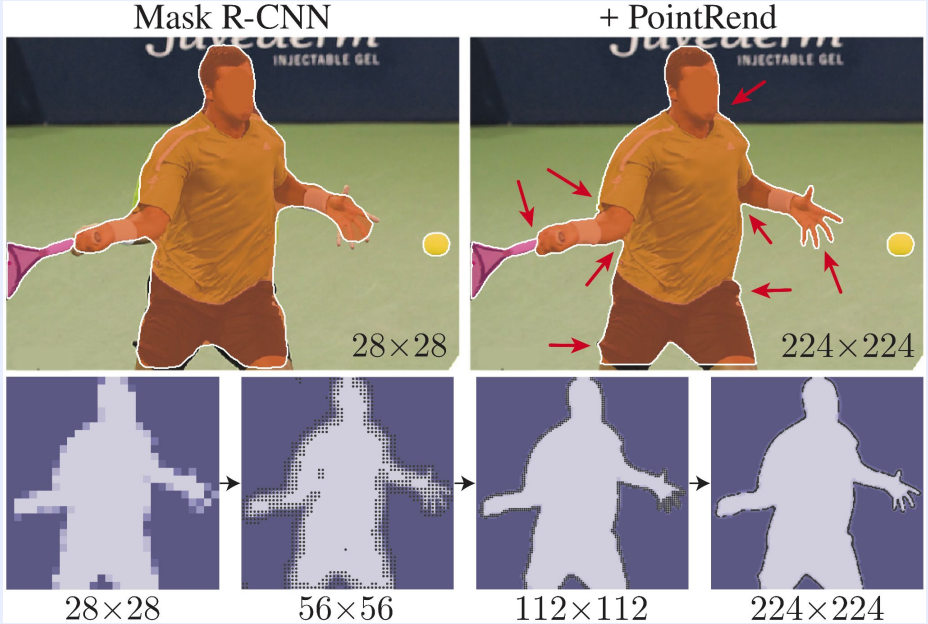


[src] https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272

Architecture	Mask R-CNN
Backbone	Feature Pyramid Network Res Net 50
Mask Head	*Point Rend Mask Head*

Point Rend Mask Head (Deprecated)

The standard implementation of **MaskRCNN** includes the **Mask Head** module, at the output of which we obtain a logit mask of **28x28** values, thus for large-sized images important information will be lost and the polygons will not be accurate, and increasing the logit mask entails a significant increase in the model's memory occupied and processing time. To do this, it was replaced with the **"Point Rend Mask Head"** module, which allows you to calculate an increased logit mask without resource-intensive operations.



Parameters of Model and Training Results (Deprecated)

1. Input image size: 1024x1024 pixels
2. Training Configuration:
    a. AMP: True
    b. Optimizer: Adam
    c. Learning rate: 0.0028
    d. Batch size: 32
    e. Number of epochs: 100
3. Data Augmentation:
    a. Copy-Paste
    b. Horizontal flip
    c. Vertical flip
    d. Random rotation
    e. Random scale jitter

1. Loss Function:
    a. Combined loss: Sum of classification, bounding box regression, and mask segmentation losses
2. Point Rend Mask Head:
    a. Iterations: 4
    b. Number of points: 14x14 grid
3. Evaluation Metrics:
    - Mean Average Precision (mAP)

Mean Average Precision (mAP)

AP	AP50	AP75	APs	APm	APl
82.504	96.742	94.727	13.283	58.029	84.540

Fish Detection Using YOLO v12

State-of-the-Art Model: YOLOv12 is the latest iteration in the YOLO (You Only Look Once) series, known for its speed and accuracy in object detection tasks.

Advantages:

- **Speed**: YOLOv12 operates in real-time, making it ideal for video processing.
- **Accuracy**: The model has been fine-tuned to improve detection performance, especially in complex environments like underwater settings.
- **Lightweight**: YOLOv12 has been optimized for deployment on medium-power devices, ensuring that it can be used on-site, even with limited computational resources.

Model Architecture

- **Backbone**: YOLOv12 uses a convolutional neural network (CNN) as its backbone, which extracts features from the input images.
- **Neck**: A series of layers designed to further refine features and enhance the model's ability to detect small objects, which is critical in underwater imagery.
- **Head**: The detection head outputs bounding boxes, class probabilities, and confidence scores, enabling precise fish localization and classification.

Fish Detection Using YOLO v12 (Training Process)

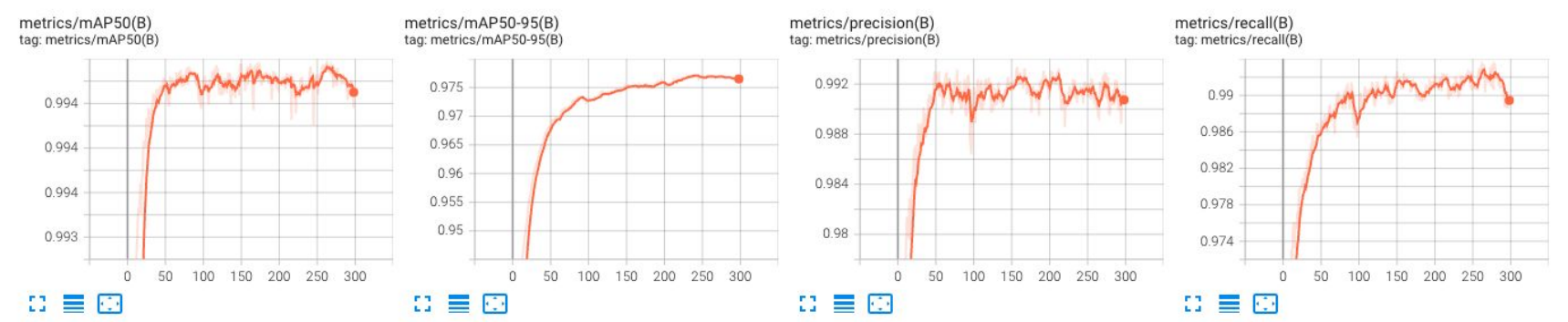
Training: The model was trained on a high-performance GPU RTX 4090. The loss function used was a combination of objectness loss, bounding box regression loss, and classification loss.

Hyperparameters:

- **Learning Rate**: Set to an optimal value for smooth convergence. **default value 0.01**
- **Batch Size**: Chosen to balance between memory usage and model performance. **default value 16**
- **Augmentation Strategies**: Employed to simulate different underwater conditions and increase generalization.

Fish Detection Using YOLO v12 (Model Performance and Evaluation)

- **Accuracy Metrics**: The model was evaluated using standard metrics such as Precision, Recall, and mAP (mean Average Precision).
- **Results**:
  - **Precision**
  - **Recall**
  - **mAP**



Fish Detection Using YOLO v12 (Model Performance and Evaluation)

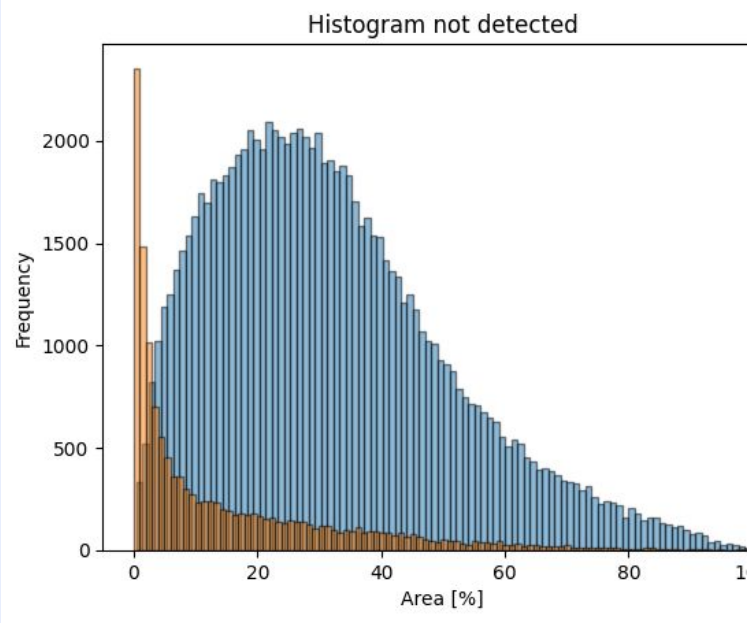
The histogram you provided shows the distribution of detected and not detected fish based on the area of the fish's bounding box as a percentage of the total image area.

Interpretation of the Histogram:

- **X-Axis (Area [%]):** This axis represents the ratio of the fish's bounding box area to the entire image area, expressed as a percentage. The values range from 0% to 100%, indicating the proportion of the image occupied by the fish.
- **Y-Axis (Frequency):** This axis indicates the frequency of occurrences for each area percentage, showing how often fish with specific area ratios are detected or not detected.

Key Observations:

1. **High Frequency of Small Area Fish (Not Detected):**
   - The brown bars on the left side of the histogram (near 0% area) show a high frequency of fish that are not detected. This suggests that fish occupying a very small portion of the image are often missed by the detection algorithm.
2. **Detected Fish Distribution (Blue Bars):**
   - The blue bars represent the frequency of detected fish. The histogram shows that as the area percentage increases (fish occupy a larger portion of the image), the frequency of detected fish also increases.
   - The peak detection frequency occurs between 10% and 30% area, where fish are more likely to be detected. After this peak, the frequency gradually decreases as the area percentage continues to rise.
3. **Overlap of Detected and Not Detected Fish:**
   - There is an overlap between the detected and not detected fish in the lower area percentages (up to around 20%), indicating that smaller fish might be inconsistently detected depending on their exact size and perhaps other factors like image quality or occlusion.



Fish Object Segmentation Using FPN with ResNet18 Backbone

Model Overview

- **Model Type**: We are using an object segmentation model with a Feature Pyramid Network (FPN) architecture, paired with a ResNet18 backbone.
- **Advantages**:
    - **FPN**: FPN is known for its ability to leverage multi-scale feature maps, which is particularly useful for segmenting objects of varying sizes, like fish in underwater images.
    - **ResNet18 Backbone**: A lightweight and efficient convolutional neural network, ResNet18 provides the model with strong feature extraction capabilities while maintaining computational efficiency.
    - **Output**: The model outputs binary masks that segment fish from the background, allowing us to crop bounding boxes around the fish.

Dataset and Preprocessing

- **Dataset**: A large dataset images containing various species of fish was used. The dataset was annotated with pixel-level masks for each fish instance.
- **Preprocessing**:
    - **Image Resizing**: All images were resized to 416x416 pixels to maintain uniform input dimensions for the model.
    - **Data Augmentation**: Techniques like flipping, rotation, and color adjustments were applied to increase the diversity of the training data and improve model robustness.
    - **Normalization**: Pixel values were normalized to standardize the input data, which helps in faster and more stable training.

Model Architecture

Encoder:

- **Type**: Feature Pyramid Network (FPN).
- **Backbone**: ResNet18 is used as the backbone for the encoder, responsible for extracting feature maps at different scales from the input image.

Decoder:

- The FPN structure effectively combines these multi-scale features to generate a detailed segmentation mask.
- The model outputs a single-channel binary mask (out_classes = 1) representing the fish in the image.
-

Training Process

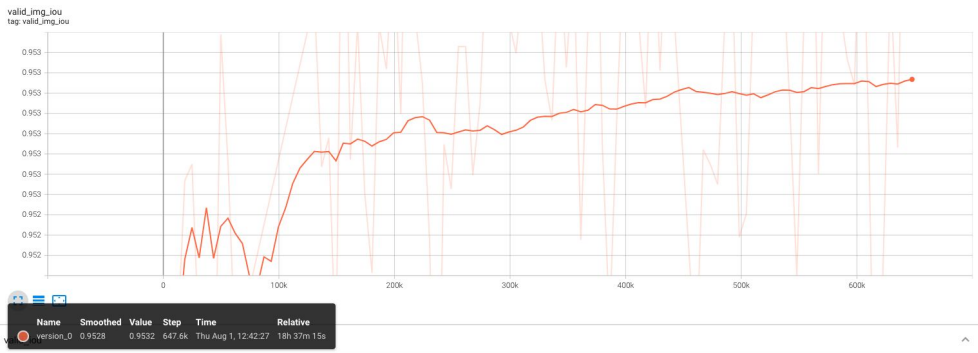
Training Setup:

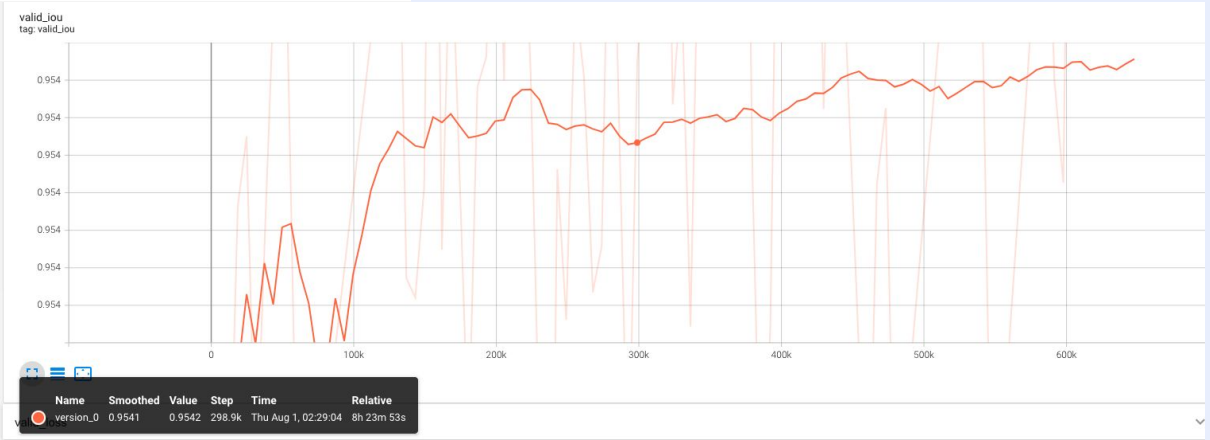
- **In Channels**: The model takes 3-channel RGB images as input (in_channels = 3).
- **Loss Function**: Dice loss was used to optimize the segmentation accuracy.
- **Hyperparameters**:
    - **Learning Rate**: self-configuring learning rate
    - **Batch Size**: 16
- **Epochs**: max 1000, with early stopping for a 20 epochs
- **Precision**: 16 bit

Model Performance and Evaluation

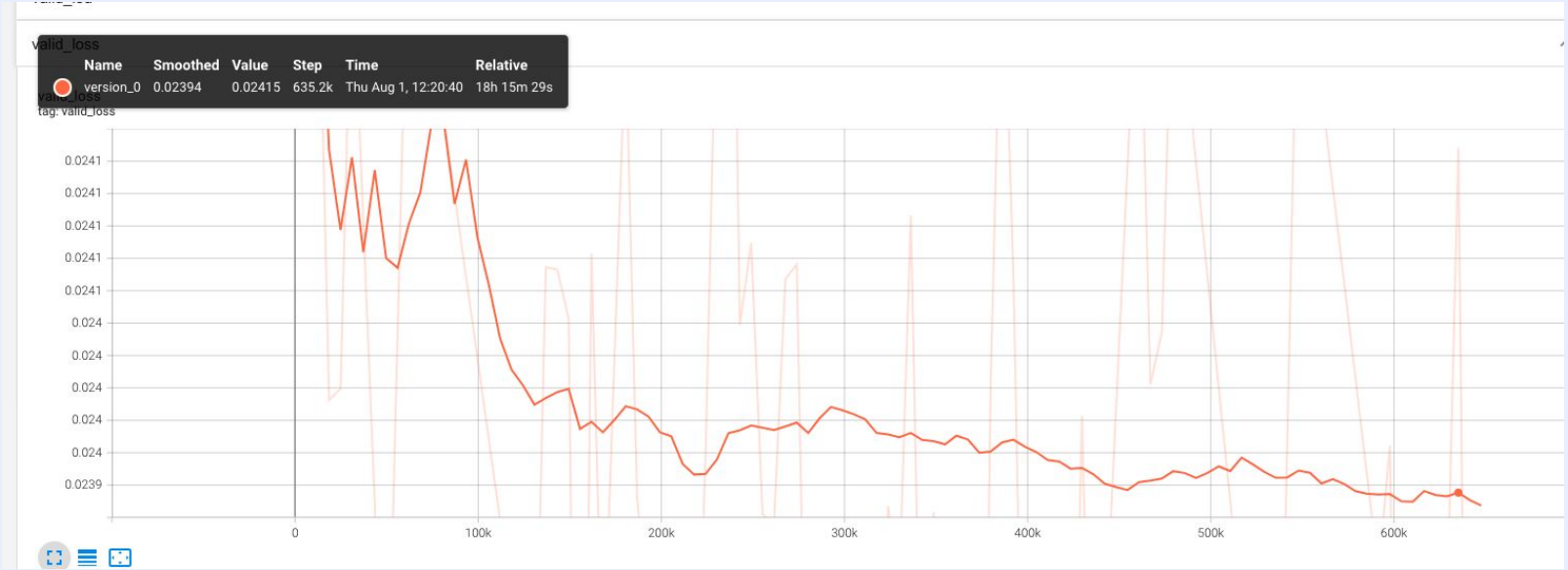
- **Evaluation Metrics**:
  - **IoU (Intersection over Union)**: Used to measure the overlap between predicted segmentation masks and ground truth masks, indicating the accuracy of fish segmentation.
  - **Dice Coefficient**: Another metric to evaluate the similarity between the predicted masks and ground truth, with a focus on overall segmentation quality.
- **Results**:
  - **IoU Score**: The model achieved high IoU scores, indicating accurate fish localization.
  - **Dice Coefficient**: The high Dice scores demonstrate the model's effectiveness in segmenting fish even in challenging underwater conditions.

average IOU on image and over full dataset

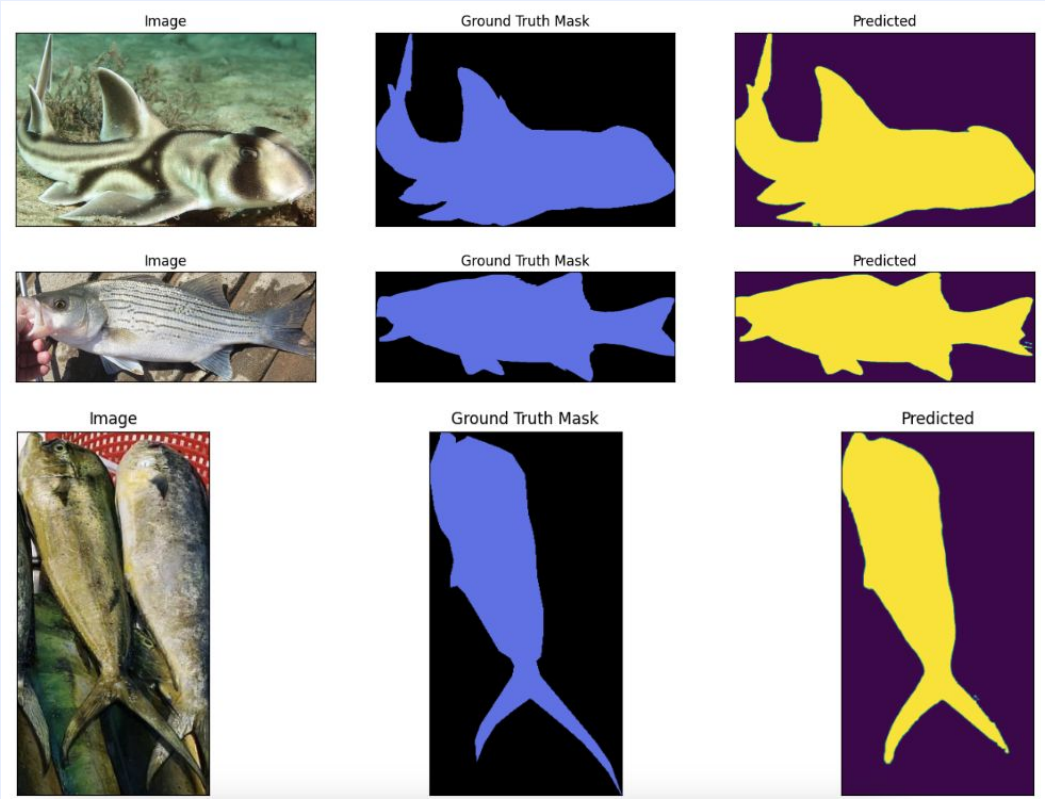




DICE LOSS on validation dataset



Example results



Fish Classification

The model based on a **Vision Transformer (ViT)** architecture, specifically designed for **deep metric learning**. Its primary purpose is to generate highly discriminative feature embeddings from input images. This type of model is commonly used for tasks like face recognition, person re-identification, or large-scale image retrieval, where the goal is to measure the similarity between images.

The model's key features include:

- A **ViT backbone** for powerful feature extraction.
- A custom **attention-based pooling mechanism** that aggregates patch features intelligently, rather than simply using the standard CLS token.
- An **ArcFace head** for training, which is a state-of-the-art loss function that promotes high intra-class compactness and inter-class separation.
- Multiple outputs for versatility: a normalized embedding for similarity tasks, logits for classification, and a visual attention map for interpretability.

Input Data

Input Data (Image) The model receives an image input with the shape [B, C, H, W], where (image size 224x224):

- B is the batch size (number of images).
- C is the number of image channels (e.g., 3 for RGB).
- H and W are the height and width of the image.

Vision Transformer Backbone

Vision Transformer Backbone The image is processed by a pretrained Vision Transformer (ViT) model, such as beitv2_base_patch16_224.in1k_ft_in22k_in1k from the timm library.

The image is divided into patches and encoded into vector embeddings.

The backbone outputs a sequence of patch embeddings with dimensions [B, Num_Patches, Embed_Dim].

ViTAttentionPooling

ViTAttentionPooling The patch tokens undergo an Attention Pooling layer. This layer employs a small trainable neural network, comprising linear layers and a Tanh activation function, to compute attention weights for each patch.

Attention weights are normalized using softmax.

A weighted sum of the patches is then calculated, resulting in an aggregated (pooled) embedding for each image with dimensions [B, Embed_Dim].

Embedding

Embedding Fully Connected Layers The aggregated embedding passes through a linear layer to transform it into an embedding of the desired dimension set to 512.

Optionally, Batch Normalization and Dropout may be applied.

The output is an embedding vector of size [B, Embedding_Dim].

Embedding Normalization The embeddings are normalized using the L2 norm, making them suitable for cosine similarity computations in subsequent search or classification tasks.

ArcFace Head

ArcFace Head Normalized embeddings are fed into an ArcFace head:

Cosine similarity is computed between embeddings and learnable class centers.

During training, an angular margin is added for the correct class, enhancing class separability.

The resulting values are scaled by a factor s.

Output

Model Output The model returns three outputs (as needed):

- Normalized embeddings: Used for nearest neighbor search or cluster visualization.
- Logits (or class probabilities): Enable image classification.
- Attention Map: Used to visualize which image regions the model considers most informative for decision-making.

Training Results

Overall classification accuracy:
Class num: 640

Average **F1-score**: 0.936

Thank you