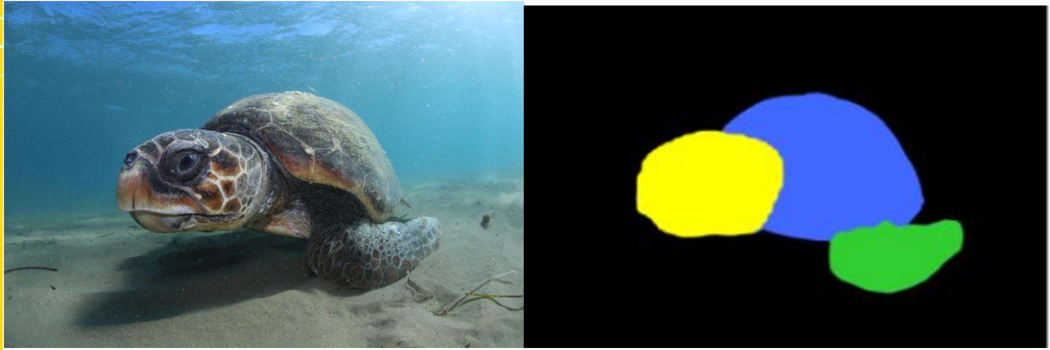
Sea Turtle Segmentation



Born Winner

Zihang Cheng (z5502944)
Yitong Li (z5504073)
Jinghan Wang (z5286124)
Xianghui Jiang (z5468921)
Xinyi Wu (z5509160)

UNSW SYDNEY

Introduction

- Goal of the Project
  - Developing and comparing various computer vision methods
  - Wildlife research
- Methods and Evaluation
  - Six different methods
  - IoU and Accuracy

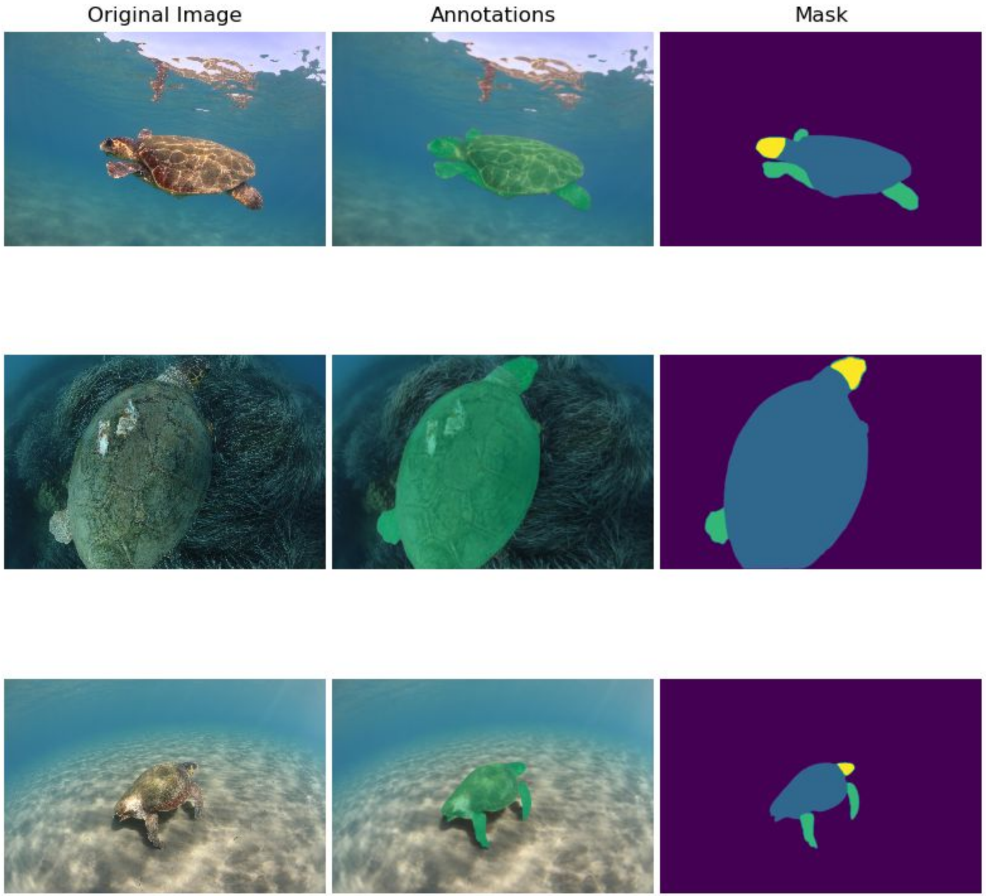
Dataset

Original Dataset：

- SeaTurtleID2022 dataset
- 8,729 high-resolution images
- 438 unique sea turtles

After splitting the dataset：

- Training Set: 5,303 images
- Validation Set: 1,118 images
- Testing Set: 2,308 images



Method 1 —— Mask R-CNN

- Model structure

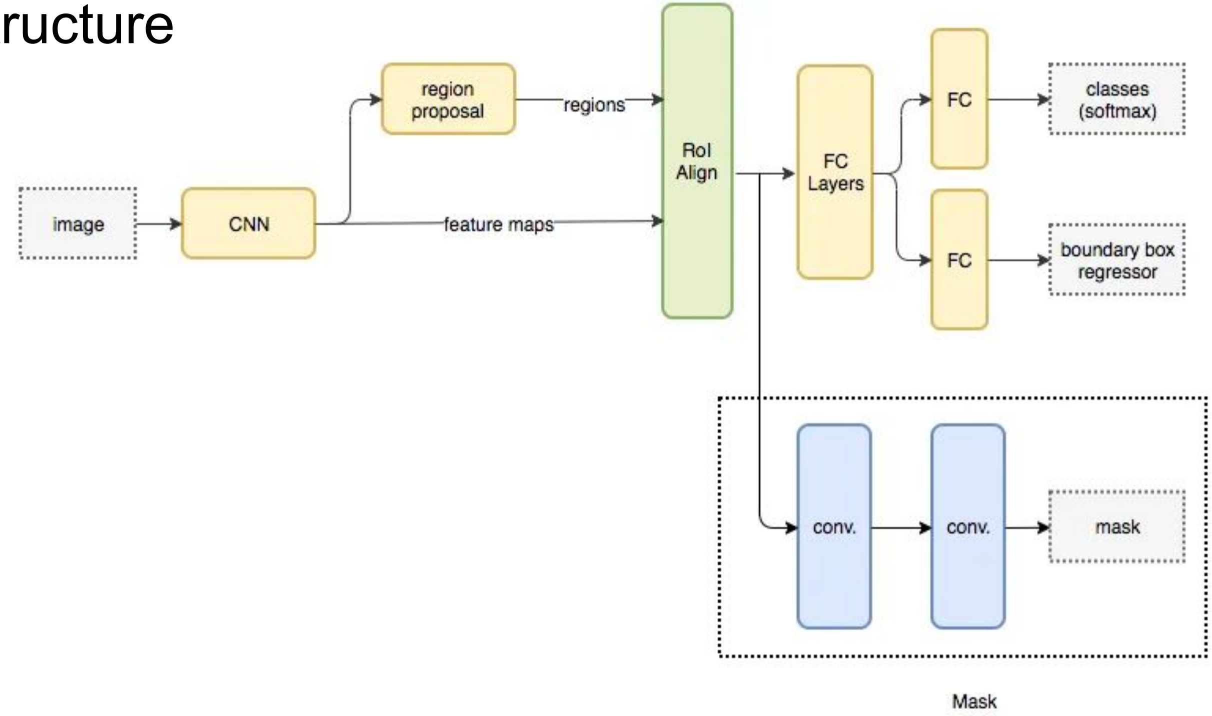


Image source: https://jonathan-hui.medium.com/image-segmentation-with-mask-r-cnn-ebe6d793272

Mask R-CNN

- Code Implementation
  - Backbone

    ResNet-50 and FPN (Feature Pyramid Network)
  - Classifier and Bounding Box

    Using FastRCNNPredictor
  - Mask Predictor

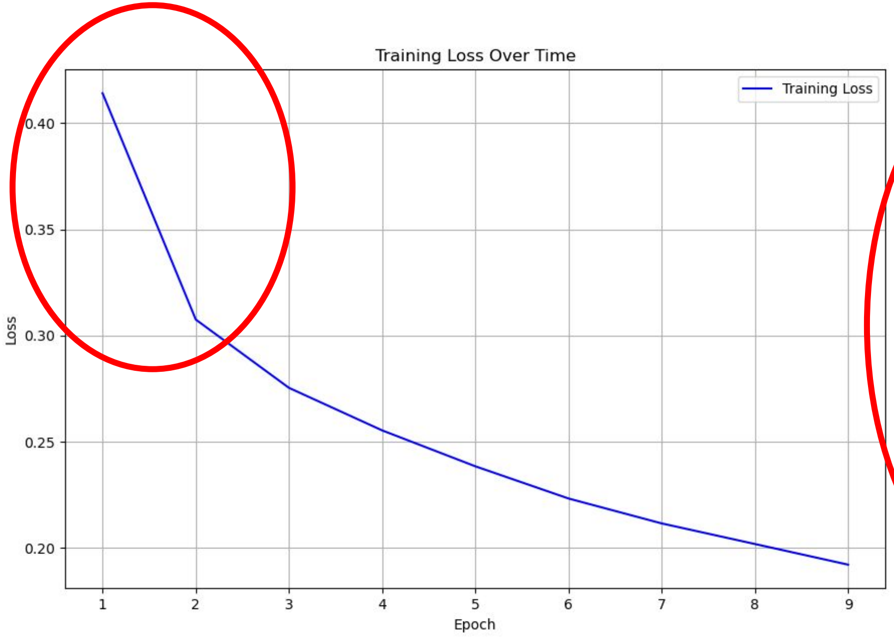
    Using MaskRCNNPredictor
  - Data augmentation
  - Gradient clipping
  - Loss monitoring

```python
def get_model_instance_segmentation(num_classes, pretrained=True):
    # Loading the base model
    model = maskrcnn_resnet50_fpn(pretrained=pretrained)

    # Modifying the classifier
    in_features = model.roi_heads.box_predictor.cls_score.in_features
    model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)

    # Modifying the mask predictor
    in_features_mask = model.roi_heads.mask_predictor.conv5_mask.in_channels
    hidden_layer = 256
    model.roi_heads.mask_predictor = MaskRCNNPredictor(
        in_features_mask, hidden_layer, num_classes
    )

    return model
```

Results of Mask R-CNN

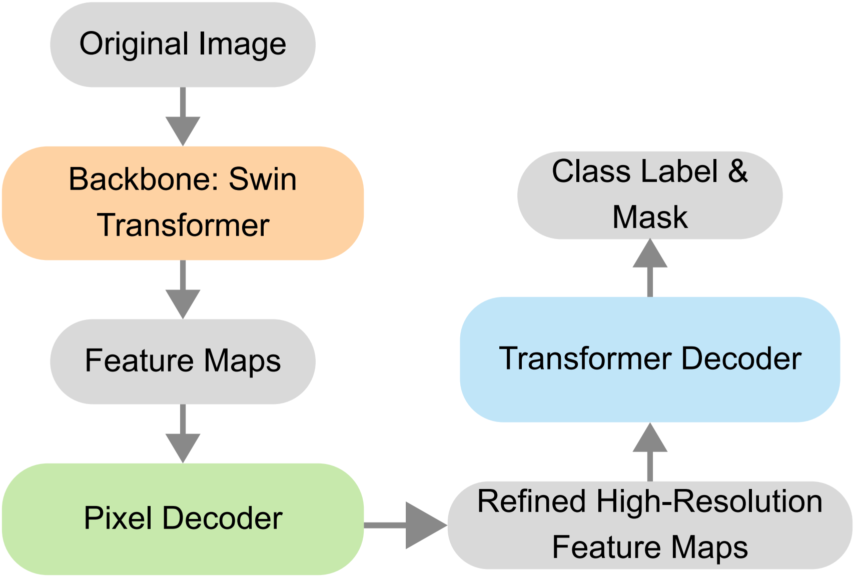
```
Test Results:
Mean Metrics per Category:
  turtle:
    Mean IoU: 0.9455
    Mean Dice Coefficient: 0.9707
    Mean Accuracy: 0.9764
  flipper:
    Mean IoU: 0.8495
    Mean Dice Coefficient: 0.9066
    Mean Accuracy: 0.9580
  head:
    Mean IoU: 0.8362
    Mean Dice Coefficient: 0.8973
    Mean Accuracy: 0.9146

Average Test Metrics:
  Average Test IoU: 0.8771
  Average Test Dice Coefficient: 0.9249
  Average Test Accuracy: 0.9497
```

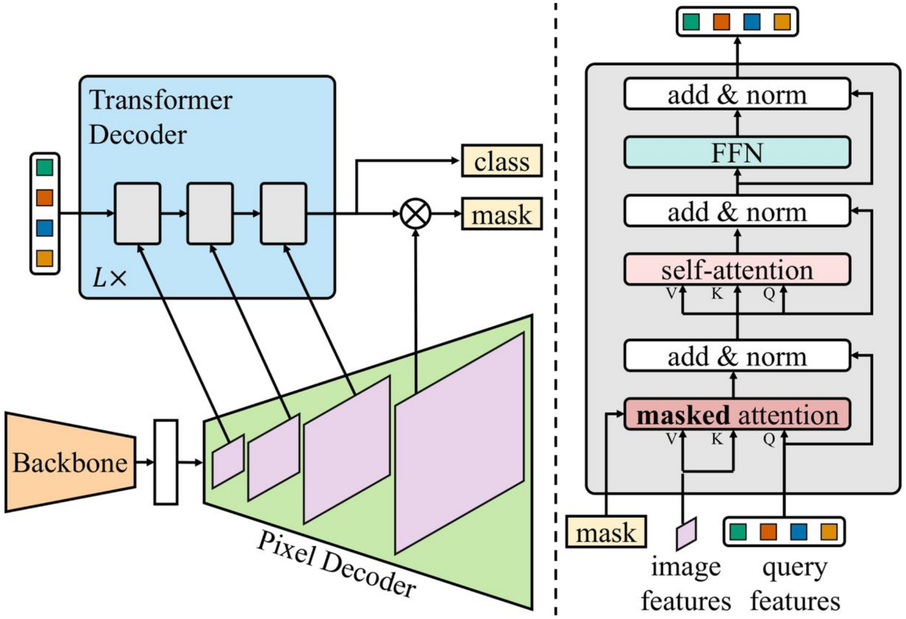
Results of Mask R-CNN



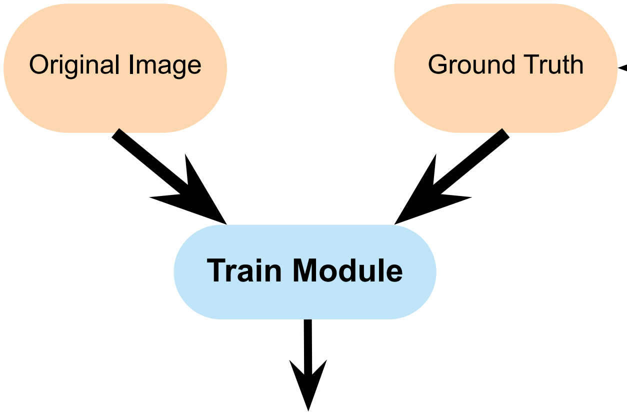
Method 2 —— Mask2Former+Swin Transformer

Structure





Mask2Former+Swin Transformer




```python
train_ids=[]
val_ids=[]
test_ids=[]
for i,row in data.iterrows():
    if row['split_open']=='train':
        train_ids.append(row['id'])
    elif row['split_open']=='valid':
        val_ids.append(row['id'])
    elif row['split_open']=='test':
        test_ids.append(row['id'])
print(f"TRAIN: {len(train_ids)}, VAL: {len(val_ids)}, TEST: {len(test_ids)}")
```

```
TRAIN: 5303, VAL: 1118, TEST: 2308
```

Split IDs to Different Lists

ID

```python
NUM_CLASSES=4
ALL_CLASSES=['backdrop','turtle','head','flipper']
LABEL_COLORS_LIST=np.array([
    [0,0,0],
    [255,255,0],
    [0,0,255],
    [255,0,0]
])
def generate_mask(img_id):
    coco=COCO("/kaggle/input/seaturtleid2022/turtles-data/data/annotations.json")
    catids=coco.getCatIds()
    annids=coco.getAnnIds(imgIds=img_id, catIds=catids, iscrowd=None)
    anns=coco.loadAnns(annids)
    mask=np.zeros((coco.imgs[img_id]['height'],coco.imgs[img_id]['width']),dtype=int)
    for j in anns:
        putmask=coco.annToMask(j)
        mask=np.maximum(mask,putmask*j['category_id'])
    return mask
```

CLASSES & LABEL_COLOR_LIST

Mask2Former+Swin Transformer

PARAMETERS

Parameters	Purpose	Value	Effect	Apply Value/Not Apply	Methods to Mitigate Negative Impacts
Batch_size	Controls the number of samples processed before model updates, affecting **memory usage and generalization**.	16	Capture **finer details**, improving generalization	16	Use **more num_workers** to accelerate the process.
		32	Use **more memory**,and converge more **slowly**		
num_workers	Control the **num of threads** to load data in parallel that affects the **efficiency**.	1	Load data **slowly**, useful for **smaller dataset**.	4	To balance memory usage and speed, **release memory** after each batch with torch.cuda.empty_cache() and gc.collect() to improve efficiency.
		4	Load **faster** but use more memory.		

```
train_data_loader = DataLoader(
    train_dataset,
    batch_size=batch_size,
    drop_last=False,
    num_workers=4,
    shuffle=True,
    collate_fn=collate_func
)
```

```
del pixel_values
del mask_labels
del class_labels
del pixel_mask
del outputs
gc.collect()
torch.cuda.empty_cache()
```

Mask2Former+Swin Transformer

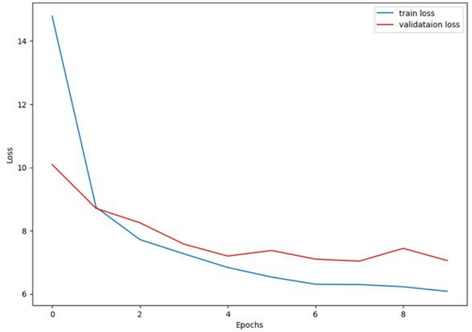
PARAMETERS

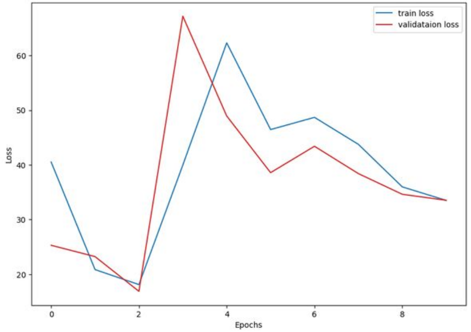
Parameters	Purpose	Value	Effect	Apply Value/Not Apply	Methods to Mitigate Negative Impacts
Weight_Decay	Control **overfitting** by applying a regularization term that penalizes large weights.	0.0001	Suppress learning **fine-grained features**	0.0001	**Use Data Augmentation** to prevent overfitting and improve generalization.
Learning_rate	Determines the step size during gradient descent, impacting convergence **speed and stability**.	0.001	Cause **unstable** or poor convergence	0.0001	**Add epochs** to train, input more features and choose a suitable **learning_rate**.
		0.0001	Approach the **optimal solution** stably, and provide better model performance		

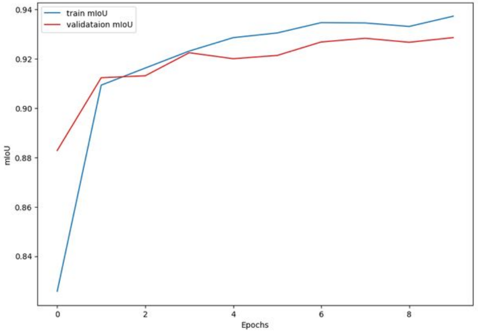
Results

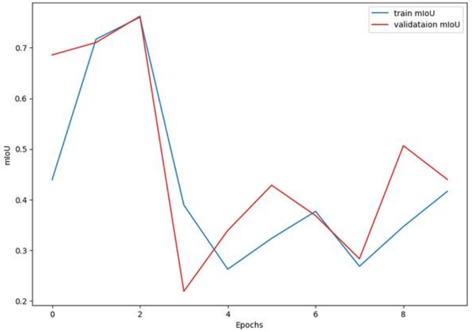
lr=0.0001

lr=0.001







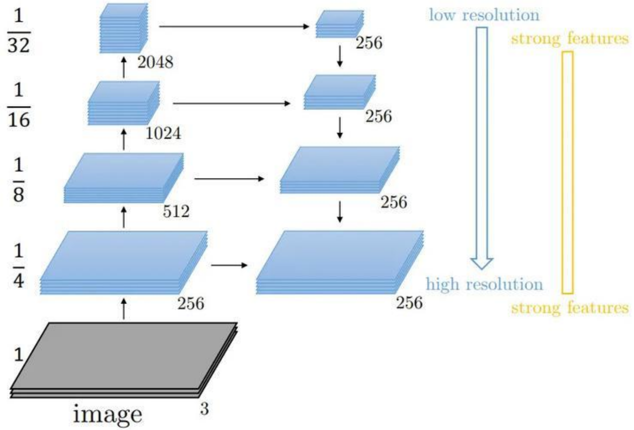



```
Mean IoU: 0.9104818343361768
Mean accuracy: 0.9502254109612451

Mean IoU of backdrop: 0.994851595678680З
Mean Accuracy of backdrop: 0.9966539178603651

Mean IoU of turtle: 0.9329208801331753
Mean Accuracy of turtle: 0.9742190345788008

Mean IoU of head: 0.861053333626903
Mean Accuracy of head: 0.928746372108177l

Mean IoU of flipper: 0.853101527905949З
Mean Accuracy of flipper: 0.9012823192976392
```

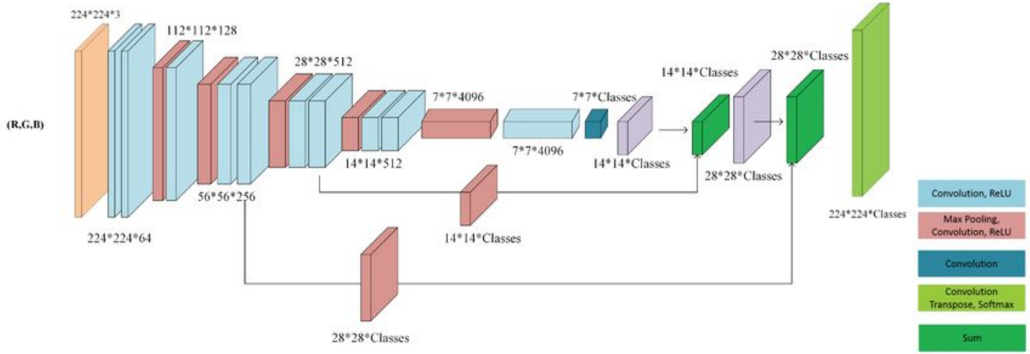
Method 3,4 —— FPN / FCN + Resnet

FPN

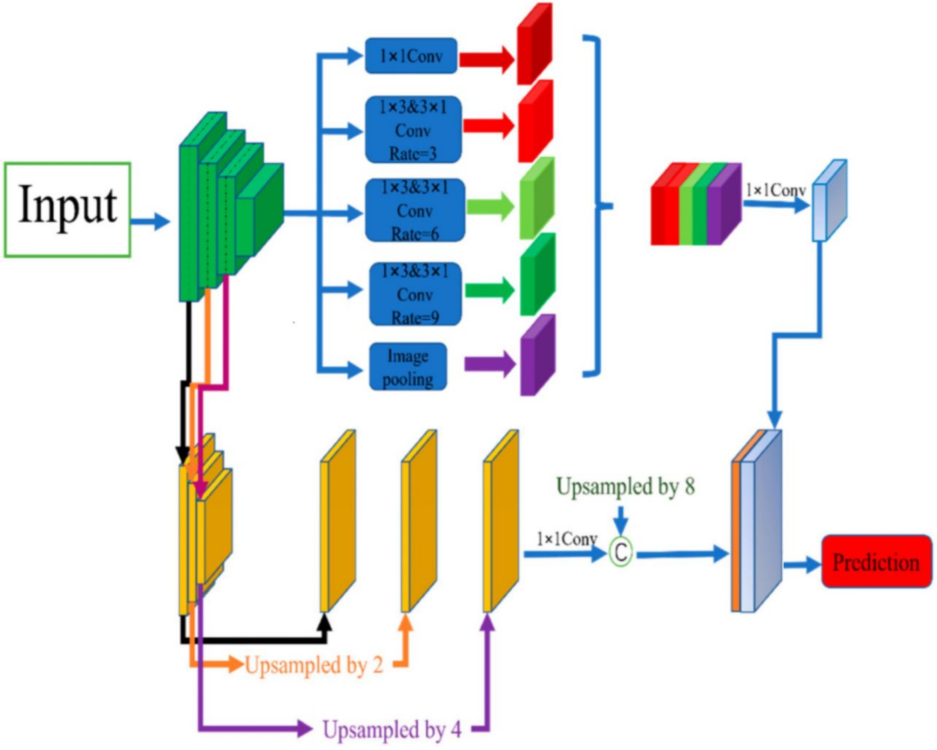
Feature Pyramid Network (FPN) [3]



FCN



Method5——Deeplabv3



1. Backbone:ResNet-50

2.ASPP:

a. a 1x1 convolution

b.3 voids convolution(different expansion rates)

c.A global average pooling layer

3.Output

Dataset Loader and preprocessing

```
[6]: coco = COCO('/content/drive/MyDrive/annotations.json')
     ds = load_dataset("EnmmmmOvO/SeaTurtleID2022")['train']

     # color:           backgtound     turtle       flipper        head
     COLOR = np.array([[0, 0, 0], [255, 0, 0],[0, 255, 0], [0, 0, 255]])
     SPLIT = 'split_closed'
     device = 'cuda' if torch.cuda.is_available() else 'cpu'
     iou_metric = evaluate.load("mean_iou")
     TRAIN_EPOCHS = 20
```

```
loading annotations into memory...
Done (t=7.06s)
creating index...
index created!
README.md:   0%|          | 0.00/656 [00:00<?, ?B/s]
train-00000-of-00004.parquet:   0%|          | 0.00/458M [00:00<?, ?B/s]
train-00001-of-00004.parquet:   0%|          | 0.00/483M [00:00<?, ?B/s]
train-00002-of-00004.parquet:   0%|          | 0.00/432M [00:00<?, ?B/s]
train-00003-of-00004.parquet:   0%|          | 0.00/382M [00:00<?, ?B/s]
Generating train split:   0%|          | 0/8729 [00:00<?, ? examples/s]
Downloading builder script:   0%|          | 0.00/12.9k [00:00<?, ?B/s]
```

```
[7]: train_dataset = ds.filter(lambda x: x[SPLIT] == 'train')
     valid_dataset = ds.filter(lambda x: x[SPLIT] == 'valid')
     test_dataset = ds.filter(lambda x: x[SPLIT] == 'test')
```

```
Filter:   0%|          | 0/8729 [00:00<?, ? examples/s]
Filter:   0%|          | 0/8729 [00:00<?, ? examples/s]
Filter:   0%|          | 0/8729 [00:00<?, ? examples/s]
```

```
[8]: def get_mask(id: str | int):
         id = int(id)
         anns = coco.loadAnns(coco.getAnnIds(imgIds=id, catIds=coco.getCatIds(), iscrowd=None))
         mask = np.zeros((coco.imgs[id]['height'], coco.imgs[id]['width']), dtype=int)
         for ann in anns:
             submask = coco.annToMask(ann)
             mask = np.maximum(mask, submask * ann['category_id'])
         return mask
```

```
[26]: class TurtleDataset(Dataset):
          def __init__(self, dataset, transform=None, rotate=None):
              self.dataset = dataset
              self.transform = transform
              self.rotate = rotate

          def __len__(self):
              return len(self.dataset)

          def __getitem__(self, index):
              id = self.dataset[index]['id']
              image = np.array(self.dataset[index]['image'])
              mask = np.array(get_mask(id))

              if self.transform:
                  augmented = self.transform(image=image, mask=mask)
                  image = augmented['image']
                  mask = augmented['mask']

              if self.rotate:
                  image = torch.tensor(image).permute(2, 0, 1)

              return image, mask
```

Apply the same data enhancement to the picture and the mask

```
[12]: train_transform = A.Compose([
          A.Resize(256, 256, always_apply=True),
          A.HorizontalFlip(p=0.4),
          A.RandomBrightnessContrast(p=0.15),
          A.Rotate(limit=20),
          A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
      ], is_check_shapes=False, additional_targets={'mask': 'mask'})

      image_transform = A.Compose([
          A.Resize(256, 256, always_apply=True),
          A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),
      ], is_check_shapes=False)
```

Dataset Loader and preprocessing

Model

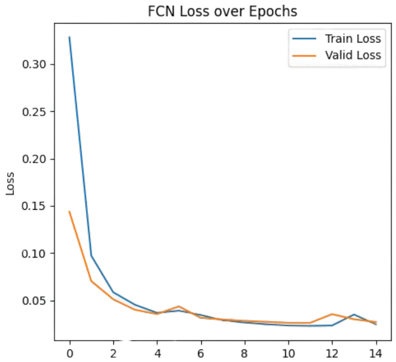
```
model = fcn_resnet101(weights=FCN_ResNet101_Weights.DEFAULT).to(device)
model.classifier[4] = nn.Conv2d(512, 4, kernel_size=1)
model = model.to(device)
```

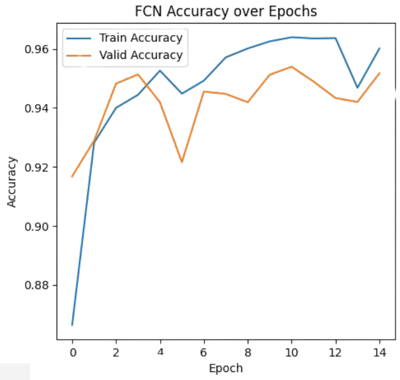
```
model = FPN(encoder_name="resnet101", encoder_weights="imagenet", in_channels=3, classes=4).to(device)
```

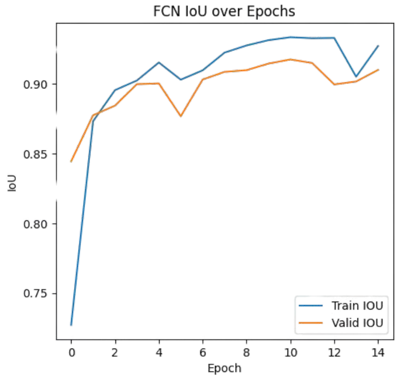
```
optimizer = optim.AdamW(model.parameters(), lr=1e-4, weight_decay=1e-5)
scheduler = ReduceLROnPlateau(optimizer, mode='min', factor=0.1, patience=5, verbose=True)
criterion = torch.nn.CrossEntropyLoss()
```

Optimizer

Results

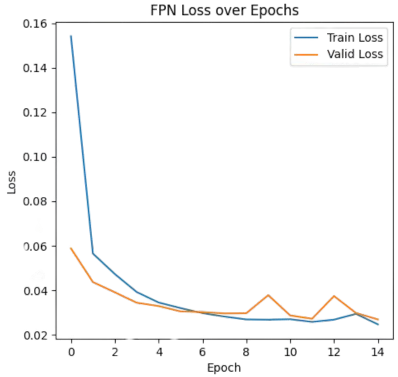


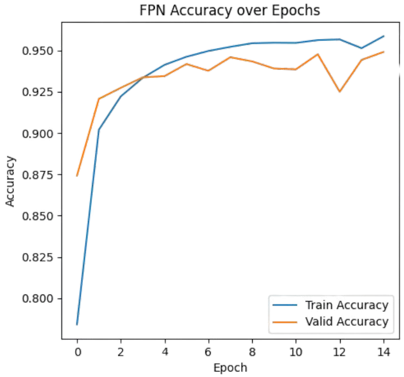


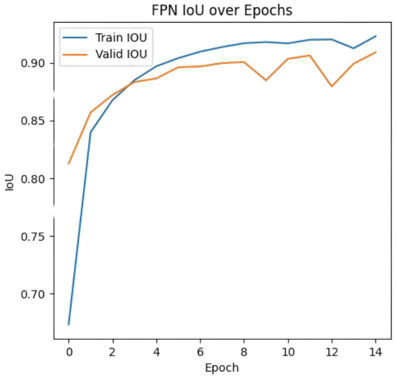



```
FCN (Resnet101)
  Mean IoU: 0.9104
  Mean Accuracy: 0.9488

  Mean IoU of backdrop: 0.9937
  Mean Accuracy of backdrop: 0.9970

  Mean IoU of turtle: 0.9287
  Mean Accuracy of turtle: 0.9647

  Mean IoU of flipper: 0.8447
  Mean Accuracy of flipper: 0.9144

  Mean IoU of head: 0.8746
  Mean Accuracy of head: 0.9190
```

```
FPN (Resnet152):
  Mean IoU: 0.9004
  Mean Accuracy: 0.9434

  Mean IoU of backdrop: 0.9926
  Mean Accuracy of backdrop: 0.9962

  Mean IoU of turtle: 0.9202
  Mean Accuracy of turtle: 0.9653

  Mean IoU of flipper: 0.8216
  Mean Accuracy of flipper: 0.8897

  Mean IoU of head: 0.8670
  Mean Accuracy of head: 0.9223
```







Pre-training

```python
image_transform = transforms.Compose([
    transforms.Resize((512, 512)),
    transforms.ToTensor()
])

mask_transform = transforms.Compose([
    transforms.Resize((512, 512)),
    transforms.ToTensor(),
    transforms.Lambda(lambda x: (x * 255).long())
])
```

a. resize Image and Mask
b. Images converted to tensors

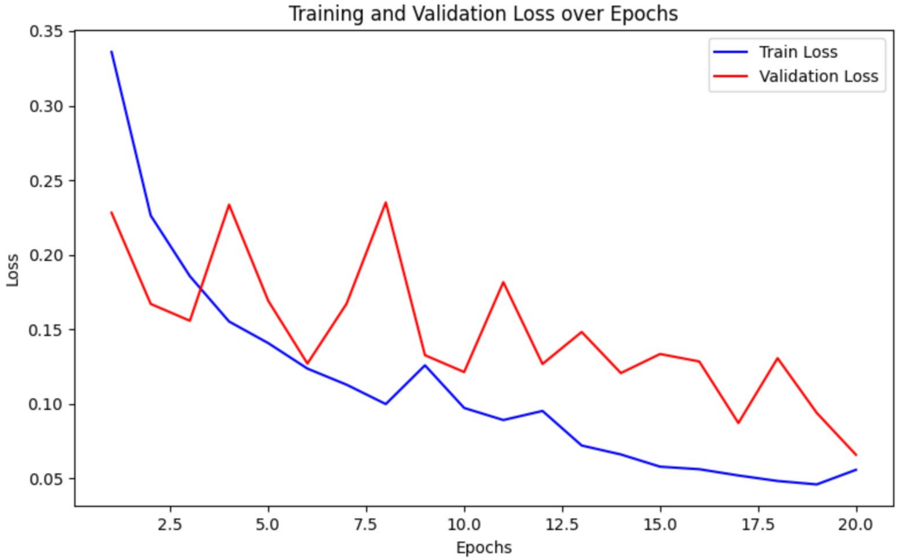
Core code

2. Read data, extract the mask

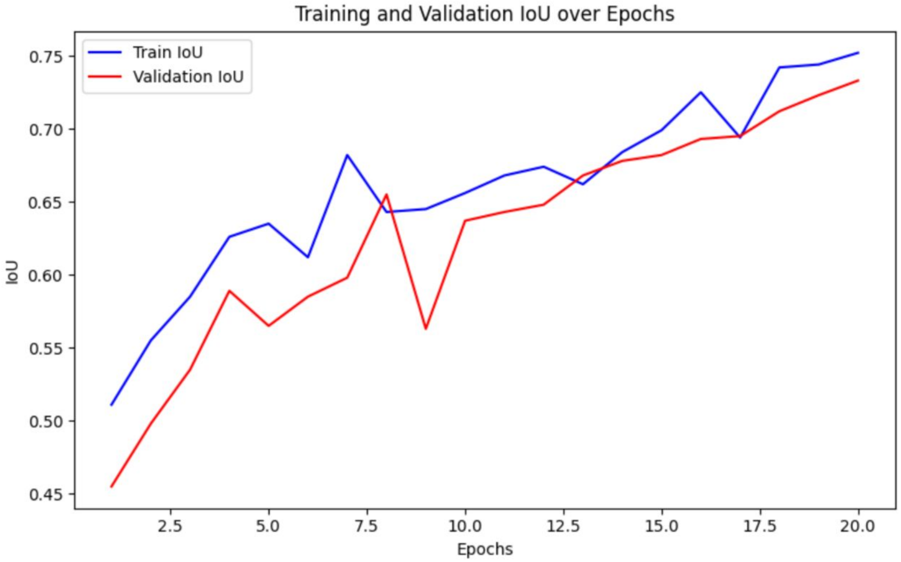
```python
class TurtleDataset(Dataset):
    def __init__(self, image_dir, coco, types, image_transform=image_transform, mask_transform=mask_transform):
        self.image_dir = image_dir
        self.coco = coco
        self.image_transform = image_transform
        self.mask_transform = mask_transform
        self.images = [img for img in coco.imgs.values() if data_type_mapping.get(img['file_name']) == types]

    def _process(self, image_id, image):
        mask = np.zeros((image.size[1], image.size[0]), dtype=np.uint8)
        categories = [(1, 1), (2, 2), (3, 3)]  # (category_id, label)

        for cat_id, label in categories:
            ann_ids = coco.getAnnIds(imgIds=image_id, catIds=cat_id, iscrowd=None)
            anns = coco.loadAnns(ann_ids)
            for ann in anns:
                mask_ann = coco.annToMask(ann)
                mask = np.maximum(mask, mask_ann * label)

        return mask
```

3. classification header= 4 categories (3 categories + 1 background class)

Results



Training and Validation Loss over Epochs
Train Loss
Validation Loss
Loss
Epochs
0.35
0.30
0.25
0.20
0.15
0.10
0.05
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



Training and Validation IoU over Epochs
Train IoU
Validation IoU
IoU
Epochs
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

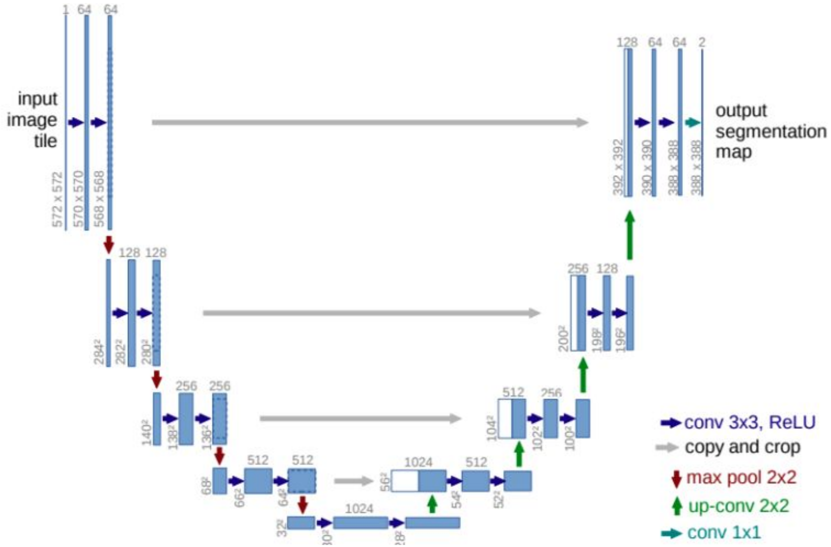
Results

```
Test Results:
Mean Metrics per Category:
turtle:
  Mean IoU: 0.7936
  Mean Dice Coefficient: 0.8696
  Mean Accuracy: 0.8933
flipper:
  Mean IoU: 0.6219
  Mean Dice Coefficient: 0.7261
  Mean Accuracy: 0.7274
head:
  Mean IoU: 0.6220
  Mean Dice Coefficient: 0.7113
  Mean Accuracy: 0.7009

Average Test Metrics:
Average Test IoU: 0.7563
Average Test Dice Coefficient: 0.8252
Average Test Accuracy: 0.8289
```

Method6 - UNet

- UNet from Scratch using PyTorch

Breif describe: The U-Net is a convolutional neural network architecture designed primarily.

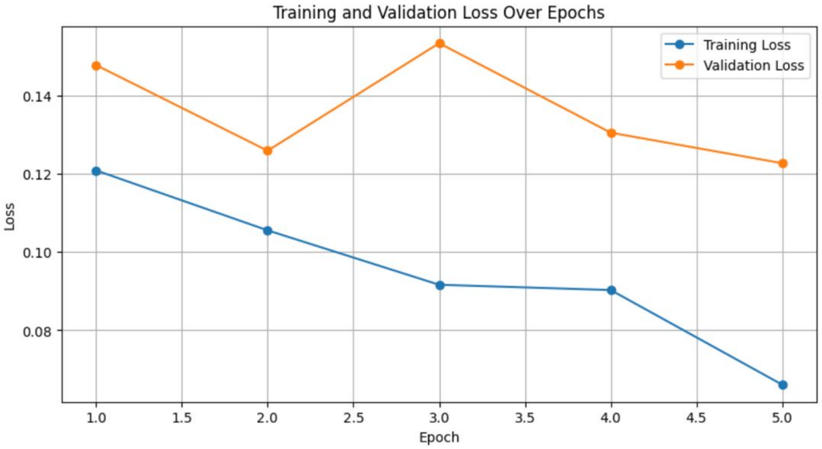


- UNet from Scratch using PyTorch

core code

```
import segmentation_models_pytorch as smp
import torch.optim as optim
import torch.nn as nn

# Define the U-Net model with a pre-trained ResNet34 backbone
model = smp.Unet(
    encoder_name="resnet50",
    encoder_weights="imagenet",     # Use ImageNet pre-trained weights
    in_channels=3,                  # Input channels (RGB)
    classes=4,                      # Output classes (background, head, body, flippers,
    activation=None             # No softmax applied here; handled by CrossEntropyLoss
)


criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)  # Adam optimizer
```

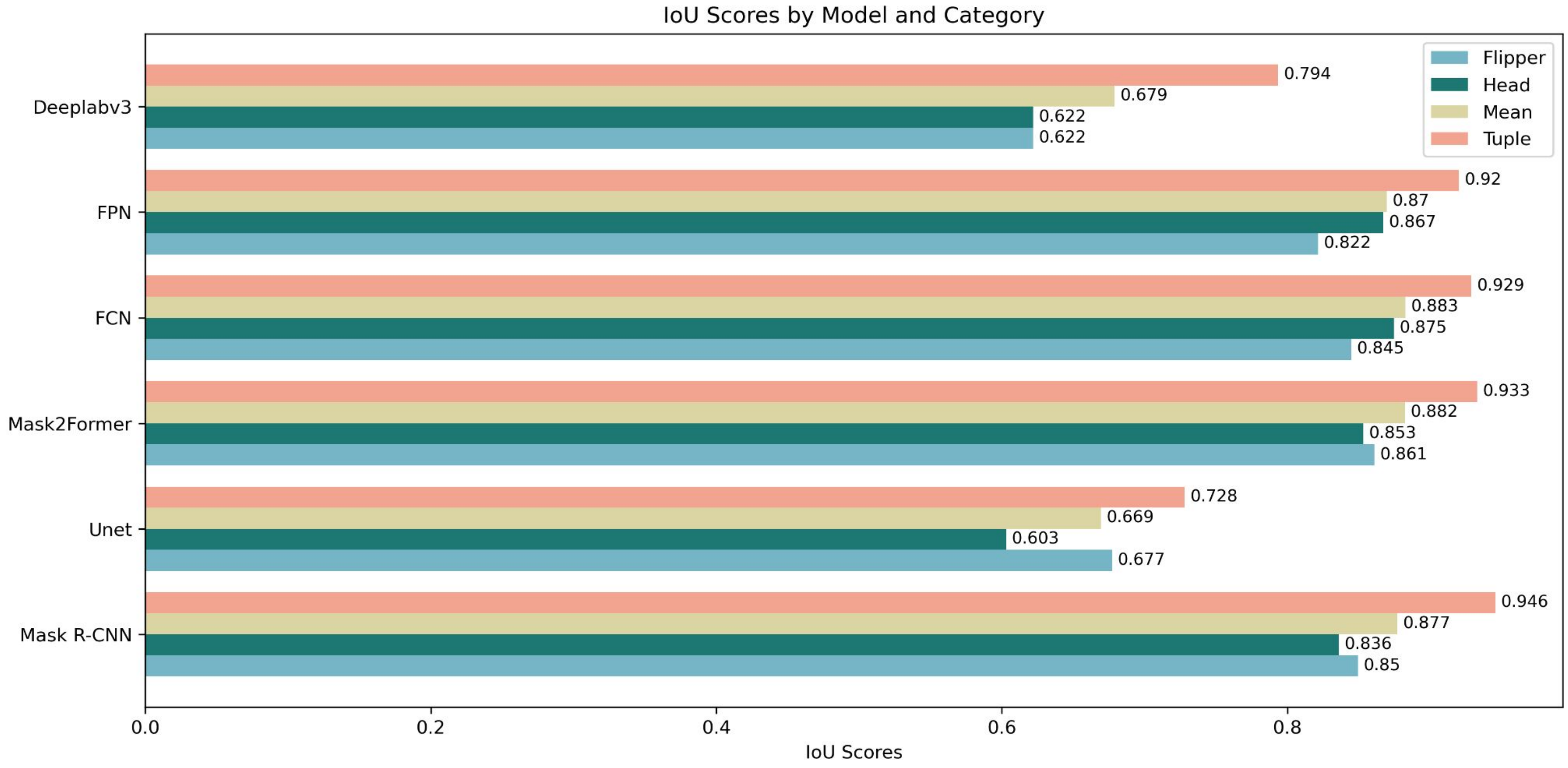
Unet-Results




```
Intersection over Union (IoU):
IoU for turtle: 0.728
IoU for flippers: 0.677
IoU for head: 0.603
Overall mean IoU (mIoU): 0.669
```

Presenter: Zihang Cheng (z5502944)

Discussion



Presenter: Zihang Cheng (z5502944)

Conclusion

Mask R-CNN	precise object instance segmentation.
Mask2Former+Swin Transformer	complex and detail-rich segmentation
FCN	Real-time segmentation
DeepLab V3	capturing large-scale objects and background information in extensive datasets
U-Net	Small datasets and large, regular-shaped targets.
FPN	multi-scale segmentation