endo_Basic_Unet_mk1

May 11, 2025

[42]:
```
import os
import numpy as np
from numpy.lib.stride_tricks import as_strided
import time
import matplotlib.pyplot as plt
from scipy.spatial.distance import directed_hausdorff

import torch
from torch.utils.data import DataLoader
from torch.utils.data import Dataset
from torch.utils.data import DataLoader, random_split
from torch.optim.lr_scheduler import StepLR

from pytorch_lightning import LightningDataModule
from pytorch_lightning import LightningModule
from pytorch_lightning import Trainer
from pytorch_lightning.callbacks import LearningRateMonitor, ModelCheckpoint
from pytorch_lightning.callbacks import EarlyStopping
from pytorch_lightning.loggers import TensorBoardLogger

from sklearn.model_selection import train_test_split

from monai.networks.nets import BasicUNet
from monai.losses import DiceCELoss
from monai.metrics import DiceMetric, MeanIoU, HausdorffDistanceMetric
from monai.transforms import (
    AsDiscreted,
    Compose,
    Resized,
    EnsureChannelFirstd,
    LoadImaged,
    ScaleIntensityd,
    ToTensord,
    RandFlipd,
    RandZoomd,
    ToTensord,
    AsDiscreted,
    CenterSpatialCropd
)
```

[43]:
```
# Custom dataset class for pytorch compatibility
# https://pytorch.org/tutorials/beginner/data_loading_tutorial.html
class EndoVis2017Dataset(Dataset):
    def __init__(self, label_subdir=None):
        self.data = []

        if label_subdir is None:
            raise ValueError("You must specify a `label_subdir` for ground␣
 ↪truth masks (e.g., 'instrument_seg_composite').")

        self.root_dir = "C:/Users/dsumm/OneDrive/Documents/UMD ENPM Robotics␣
 ↪Files/BIOE658B (Intro to Medical Image Analysis)/Project/dataset/train/"
        self.label_subdir = label_subdir

        # Recursively walk through directory to find left frame image paths and␣
 ↪GT image paths
        for subdir, dirs, files in os.walk(self.root_dir):
            if 'left_frames' in subdir:
                #print("Hit!")
                for file in sorted(files):
                    if file.endswith(('.png', '.jpg', '.jpeg')):
                        img_path = os.path.join(subdir, file)
                        #print(img_path)

                        gt_root = subdir.replace('left_frames', 'ground_truth')
                        mask_path = os.path.join(gt_root, self.label_subdir,␣
 ↪file)

                        if os.path.exists(mask_path):
                            #print("Hit!")
                            self.data.append({"image": img_path, "label":␣
 ↪mask_path})    # Dictionary for MONAI compatability

        transforms_list = [
            LoadImaged(keys=["image", "label"]),                        # Loads␣
 ↪image data and metadata from file path dictionaries
            EnsureChannelFirstd(keys=["image", "label"]),               #␣
 ↪Adjust or add the channel dimension of input data to ensure channel_first␣
 ↪shape

            # Images are of nominal size 1280x1024 --> resizing for memory␣
 ↪efficiency
```

```
            CenterSpatialCropd(keys=["image", "label"], roi_size=(1024, 1280)),␣
↪           # Cropping background padding from images
            Resized(keys=["image", "label"], spatial_size=(256, 320)),          ␣
↪           # Imported images are of various sizes: standardize to 320,256

            # Apply data augmentation techniqes
            RandFlipd(keys=["image", "label"], prob=0.3, spatial_axis=1),       ␣
↪           # Horizontal axis flip imposed w/ 30% prob
            #RandRotate90d(keys=["image", "label"], prob=0.3, max_k=3),         ␣
↪            # Random 90° rotation imposed w/ 30% prob
            RandZoomd(keys=["image", "label"], prob=0.3, min_zoom=0.75,␣
↪max_zoom=1.25), # Zoom range (+/-25%) imposed w/ 30% prob
            #RandAdjustContrastd(keys=["image"], prob=0.3, gamma=(0.75, 1.25)),␣
↪            # Contrast variation (+/-25%) imposed w/ 30% prob

            ScaleIntensityd(keys=["image"]),                             # Scale␣
↪the intensity of input image to the value range 0-1
            ToTensord(keys=["image", "label"]),                          #␣
↪Ensure data is of tensor type for pytorch usage
        ]

        # Additional conditional transforms based on label_subdir
        if label_subdir == "binary_composite":
            transforms_list.append(AsDiscreted(keys=["label"], threshold=0.5)) ␣
↪          # Binary threshold for binary seg
        elif label_subdir == "part_seg_composite":
            transforms_list.append(AsDiscreted(keys=["label"], to_onehot=5))   ␣
↪          # 5 individual class labels for instrument independent part seg
        elif label_subdir == "instrument_seg_composite":
            transforms_list.append(AsDiscreted(keys=["label"], to_onehot=8))   ␣
↪           # 8 individual class labels for part independent instrument seg
        elif label_subdir == "instrument_part_seg_composite":
            transforms_list.append(AsDiscreted(keys=["label"], to_onehot=21))  ␣
↪           # 26 individual class labels for instrument & part seg

        # Imposing MONAI transforms
        # https://docs.monai.io/en/stable/transforms.html
        self.transform = Compose(transforms_list)

    def __len__(self):
        # Returns number of imported samples
        length = len(self.data)
        return length

    def __getitem__(self, idx):
        # Return transformed sample from the dataset as dictated by the index
        sample = self.data[idx]
        return self.transform(sample)
```

[44]:
```
class MONAIDataLoader(LightningDataModule):
    def __init__(self, dataset=None, batch_size: int = None, img_size: int =␣
 ↪None, dimensions:int = None):
        super().__init__()
        if dataset is None:
            raise ValueError("No dataset given!")
        else:
            self.dataset = dataset

        self.train, self.val = random_split(self.dataset, [int(len(self.
 ↪dataset) * 0.8), len(self.dataset) - int(len(self.dataset) * 0.8)])
        self.batch_size = batch_size
        #self.num_workers = 2
        self.pin_memory = True
        #self.persistent_workers = True
        print(f"Train dataset size: {len(self.train)}")
        print(f"Validation dataset size: {len(self.val)}")

    def setup(self, stage=None):
        # required by PyTorch Lightning
        pass

    def train_dataloader(self):
        return DataLoader(self.train, batch_size=self.batch_size,␣
 ↪pin_memory=self.pin_memory)

    def val_dataloader(self):
        return DataLoader(self.val, batch_size=self.batch_size, pin_memory=self.
 ↪pin_memory)

    #def predict_dataloader(self):
    #    return DataLoader(self.test, batch_size=self.batch_size,␣
 ↪num_workers=16)
```

[45]:
```
class basic_UNet_Train(LightningModule):
    def __init__(self, img_size=(1, 3, 256, 320), batch_size=1, lr=0.001,␣
 ↪num_classes=1):
        super().__init__()

        self.save_hyperparameters()
        self.num_classes = num_classes
        print("num_classes", self.num_classes, num_classes, self.hparams.
 ↪num_classes)
        self.example_input_array = [torch.zeros(self.hparams.img_size)]
        self.dice_metric = DiceMetric(include_background=True,␣
↪reduction="mean", ignore_empty=True)
        self.iou_metric = MeanIoU(include_background=True, reduction="mean",␣
↪ignore_empty=True)

        # Metric tracking
        self.dice_scores = []
        self.iou_scores = []

        # Defining MONAI Unet model paramters
        self.model = BasicUNet(spatial_dims=2,      # 2D image so spatial dims␣
↪= 2
                              in_channels=3,        # RGB input ultrasound image
                              out_channels=num_classes,      # Binary␣
↪segmentation mask output image
                              features= (32, 64, 128, 256, 512, 32),        #␣
↪standard Unet feature sizes (32, 32, 64, 128, 256, 32)
                              dropout=0.1)     # Dropout prob 10%

        # Using combined DICE and CE loss as loss function
        # Conditional loss function based on the number of classes
        if num_classes == 1:
            self.DICE_CE_Loss = DiceCELoss(
                include_background=False,  # Exclude background class
                sigmoid=True,  # Use softmax for multiclass segmentation
                softmax=False,  # Apply softmax for multiclass
                lambda_dice=1.0,  # Adjust the weight for Dice loss
                lambda_ce=1.0,  # Adjust the weight for Cross-Entropy loss
                reduction='mean'  # Use mean reduction
            )
        else:
            self.DICE_CE_Loss = DiceCELoss(
                include_background=False,  # Exclude background class
                sigmoid=False,  # Use softmax for multiclass segmentation
                softmax=True,  # Apply softmax for multiclass
                lambda_dice=1.0,  # Adjust the weight for Dice loss
                lambda_ce=1.0,  # Adjust the weight for Cross-Entropy loss
                reduction='mean'  # Use mean reduction
            )

        # Tracking losses for matplotlib
        self.train_losses = []
        self.val_losses = []

        # For storing images for the last epoch
        self.last_image = []
        self.last_pred = []
        self.last_mask = []
        self.logged_epochs = []

    # Passes model inputs through U-net to get output predictions
    def forward(self, inputs):
        outputs = self.model(inputs)
        return outputs

    def training_step(self, batch, batch_idx):
        # Gets labels for input and corresponding ground truth
        inputs, gt_input = self._prepare_batch(batch)

        # Call forward pass
        outputs = self.forward(inputs)

        # Compute DICE & CE loss based on current params
        loss = self.DICE_CE_Loss(outputs, gt_input)

        # Log DICE loss with PyTorch Lightning logger
        self.log(f"Train_Dice_CE_loss", loss, on_epoch=True, prog_bar=True)

        # Append train loss at the end of each epoch
        if batch_idx == len(batch) - 1:
            self.train_losses.append(loss.item())

        return loss

    def validation_step(self, batch, batch_idx):

        # Gets labels for input and corresponding ground truth
        inputs, gt_input = self._prepare_batch(batch)
        outputs = self.forward(inputs)
        loss = self.DICE_CE_Loss(outputs, gt_input)
        self.log("val_loss", loss, on_step=False, on_epoch=True, prog_bar=True)

        if self.hparams.num_classes == 1:
            probs = torch.sigmoid(outputs)
            preds = (probs > 0.5).float()
            # Ensure ground truth is binary (i.e., 0 or 1)
            gt_input = (gt_input > 0.5).float()  # Threshold the ground truth␣
↪if needed

            intersection = (preds * gt_input).sum()
            union = preds.sum() + gt_input.sum()
            bin_dice_score = 2.0 * intersection / (union + 1e-8)  # Avoid␣
↪division by zero
```

```
            # IoU score calculation for binary segmentation
            bin_iou_score = intersection / (union - intersection + 1e-8)  #␣
↪Avoid division by zero

            self.log("val_dice", bin_dice_score, on_step=False, on_epoch=True,␣
↪prog_bar=True)
            self.log("val_iou", bin_iou_score, on_step=False, on_epoch=True,␣
↪prog_bar=True)

        else:
            probs = torch.softmax(outputs, dim=1)
            preds = torch.nn.functional.one_hot(torch.argmax(probs, dim=1),␣
↪num_classes=self.num_classes)
            preds = preds.permute(0, 3, 1, 2).float()  # Shape: [B, C, H, W]

            self.dice_metric(y_pred=preds, y=gt_input)
            self.iou_metric(y_pred=preds, y=gt_input)

        if self.trainer.sanity_checking:
            return  # skip logging during sanity check

        # Append validation loss at the end of each epoch
        if batch_idx == len(batch) - 1:
            self.val_losses.append(loss.item())

            # For binary segmentation: apply sigmoid and threshold
            if self.hparams.num_classes == 1:
                outputs = torch.sigmoid(outputs)
                outputs = (outputs > 0.5).float()  # Convert probabilities to␣
↪binary mask
                self.dice_scores.append(bin_dice_score)
                self.iou_scores.append(bin_iou_score)

            # For multiclass segmentation: apply softmax
            else:
                outputs = torch.softmax(outputs, dim=1)  # Apply softmax for␣
↪multi-class outputs
                dice = self.dice_metric.aggregate()[0].item()
                #print("Dice", dice)
                iou = self.iou_metric.aggregate()[0].item()
                #print("IOU", iou)
                self.dice_metric.reset()
                self.iou_metric.reset()
                self.dice_scores.append(dice)
                self.iou_scores.append(iou)
                self.log("val_dice", dice, on_step=False, on_epoch=True,␣
↪prog_bar=True)
                self.log("val_iou", iou, on_step=False, on_epoch=True,␣
↪prog_bar=True)

            # Normalize and convert tensor to 3 channels (RGB) for visualization
            def process(last):
                # Detach from cpu to not interrupt training
                # https://stackoverflow.com/questions/63582590/
↪why-do-we-call-detach-before-calling-numpy-on-a-pytorch-tensor
                last = last[0].detach().cpu()

                # Min max normalization
                # https://www.codecademy.com/article/normalization
                last= (last - last.min()) / (last.max() - last.min() + 1e-8)

                # If grayscale, reshape last image to RGB for display by␣
↪replicating gray value twice
                # https://discuss.pytorch.org/t/convert-grayscale-images-to-rgb/
↪113422
                return last.repeat(3, 1, 1) if last.shape[0] == 1 else last

            current_epoch = self.current_epoch
            total_epochs = self.trainer.max_epochs
            print("TE", total_epochs)

            if current_epoch == 0 or current_epoch == total_epochs - 1 or␣
↪current_epoch == total_epochs // 2:
                self.last_image.append(process(inputs))
                self.last_pred.append(process(outputs))
                self.last_mask.append(process(gt_input))
                self.logged_epochs.append(current_epoch)
                print(f"Logged image from epoch {current_epoch}")

        return loss

    #def predict_step(self, batch, batch_idx, dataloader_idx=0):
    #    return self(batch['image'])

    def configure_optimizers(self):
        #set optimizer
        optimizer = torch.optim.AdamW(self.parameters(), lr=self.hparams.lr,␣
↪weight_decay=1e-4)
        scheduler = StepLR(optimizer, step_size=5, gamma=0.5)  # halve LR every␣
↪5 epochs
        return {
            'optimizer': optimizer,
            'lr_scheduler': {
                'scheduler': scheduler,
                'interval': 'epoch',
                'frequency': 1
            }
        }

    def _prepare_batch(self, batch):
        return batch['image'], batch['label']

    # Plot training and val losses when needed
    def plot_losses(self):
        min_len = min(len(self.train_losses), len(self.val_losses))
        epochs = range(1, min_len + 1)

        # Plotting training vs validation loss
        plt.figure(figsize=(10, 6))
        plt.plot(epochs, self.train_losses[:len(epochs)], label="Training␣
↪Loss", color='blue')
        plt.plot(epochs, self.val_losses[:len(epochs)], label="Validation␣
↪Loss", color='orange')
        plt.title("Training vs Validation Loss")
        plt.xlabel("Epochs")
        plt.ylabel("Loss")
        plt.legend()
        plt.show()

    def plot_metrics(self):
        epochs = range(1, len(self.dice_scores) + 1)

        # Convert to CPU floats if necessary
        dice = [d.cpu().item() if torch.is_tensor(d) else d for d in self.
↪dice_scores]
        iou = [i.cpu().item() if torch.is_tensor(i) else i for i in self.
↪iou_scores]

        plt.figure(figsize=(10, 6))
        plt.plot(epochs, dice, label='Dice Coefficient')
        plt.plot(epochs, iou, label='IoU')
        plt.xlabel("Epochs")
        plt.ylabel("Score")
        plt.title("Validation Metrics Over Time")
        plt.legend()
        plt.show()

    def plot_result_by_epoch(self):
        total_epochs = len(self.last_image)

        if total_epochs < 5:
            print(f"Only {total_epochs} epochs recorded, plotting all.")
            selected_epochs = list(range(total_epochs))
        else:
            print(f"{total_epochs} epochs recorded, bug in code.")

        for epoch_idx in selected_epochs:
            epoch_num = self.logged_epochs[epoch_idx] if hasattr(self,␣
↪"logged_epochs") else epoch_idx
            img = self.last_image[epoch_idx]
            pred = self.last_pred[epoch_idx]
            mask = self.last_mask[epoch_idx]

            fig, ax = plt.subplots(1, 3, figsize=(12, 4))

            ax[0].imshow(np.transpose(img.numpy(), (1, 2, 0)))
            ax[0].set_title(f"Epoch {epoch_num} - Image")
            ax[0].axis("off")

            if self.hparams.num_classes == 1:
                ax[1].imshow(np.transpose(pred.numpy(), (1, 2, 0)))
                ax[1].set_title(f"Epoch {epoch_num} - Prediction")
                ax[1].axis("off")

                ax[2].imshow(np.transpose(mask.numpy(), (1, 2, 0)))
                ax[2].set_title(f"Epoch {epoch_num} - Ground Truth")
                ax[2].axis("off")
            else:
                # Define the colormap and normalization
                num_classes = self.hparams.num_classes
                cmap = plt.get_cmap('viridis', num_classes)
                bounds = np.arange(num_classes + 1) - 0.5
                norm = plt.matplotlib.colors.BoundaryNorm(bounds, cmap.N)

                # Convert one-hot encoded predictions and masks to␣
↪single-channel class labels
                pred_mask = torch.argmax(pred, dim=0).cpu().numpy()
                true_mask = torch.argmax(mask, dim=0).cpu().numpy()

                # Apply consistent colormap and normalization
                im1 = ax[1].imshow(pred_mask, cmap=cmap, norm=norm)
                ax[1].set_title(f"Epoch {epoch_num} - Prediction")
                ax[1].axis("off")

                im2 = ax[2].imshow(true_mask, cmap=cmap, norm=norm)
                ax[2].set_title(f"Epoch {epoch_num} - Ground Truth")
                ax[2].axis("off")

                im_for_cbar = im1  # just need one mappable

                # Adjust layout to leave space at the bottom
                fig.subplots_adjust(bottom=0.25) # tweak this if labels get cut␣
↪off

                # Add a new axis below the plots for the colorbar
                cbar_ax = fig.add_axes([0.1, 0.1, 0.8, 0.10])  # [left, bottom,␣
↪width, height]
                cbar = fig.colorbar(im_for_cbar, cax=cbar_ax,␣
↪orientation='horizontal', ticks=np.arange(num_classes))

                # Add colorbar below the plots
                #cbar = fig.colorbar(im1, ax=ax.ravel().tolist(),␣
↪orientation='horizontal',
                        #ticks=np.arange(num_classes), pad=0.15, fraction=0.05)

                # Set class labels
                if num_classes == 5:
                    cbar.ax.set_xticklabels(['Background', 'Shaft', 'Wrist',␣
↪'Claspers', 'Probe'])
                elif num_classes == 8:
                    cbar.ax.set_xticklabels(['Background', 'Bipolar Forceps',␣
↪'Prograsp Forceps', 'Large Needle Driver',
                                            'Vessel Sealer', 'Grasping␣
↪Retractor', 'Monopolar Curved Scissors', 'Other'])

                    plt.setp(cbar.ax.get_xticklabels(), rotation=30,␣
↪ha="right", rotation_mode="anchor")
                elif num_classes == 21:
                    cbar.ax.set_xticklabels([
                        "Background",
                        "Bipolar Forceps Shaft", "Bipolar Forceps Wrist",␣
↪"Bipolar Forceps Claspers",
                        "Prograsp Forceps Shaft", "Prograsp Forceps Wrist",␣
↪"Prograsp Forceps Claspers",
                        "Large Needle Driver Shaft", "Large Needle Driver␣
↪Wrist", "Large Needle Driver Claspers",
                        "Vessel Sealer Shaft", "Vessel Sealer Wrist", "Vessel␣
↪Sealer Claspers",
                        "Grasping Retractor Shaft", "Grasping Retractor Wrist",␣
↪"Grasping Retractor Claspers",
                        "Monopolar Curved Scissors Shaft", "Monopolar Curved␣
↪Scissors Wrist", "Monopolar Curved Scissors Claspers",
                        "Other Probe","Other Probe"
                    ])
                    plt.setp(cbar.ax.get_xticklabels(), rotation=45,␣
↪ha="right", rotation_mode="anchor")

                cbar.set_label('Class ID')

        plt.show()
```

[46]:
```
# Generate datasets, loaders, and models for basic UNet
binary_endo_images = EndoVis2017Dataset(label_subdir='binary_composite')
part_seg_endo_images = EndoVis2017Dataset(label_subdir='part_seg_composite')
instr_seg_endo_images =␣
↪EndoVis2017Dataset(label_subdir='instrument_seg_composite')
part_instr_seg_endo_images =␣
↪EndoVis2017Dataset(label_subdir='instrument_part_seg_composite')

binary_endo_data = MONAIDataLoader(dataset=binary_endo_images, batch_size=10) ␣
↪# batch size should be divisible, ie. 50 images and bs 20 wont work
part_seg_endo_data = MONAIDataLoader(dataset=part_seg_endo_images,␣
↪batch_size=10)
instr_seg_endo_data = MONAIDataLoader(dataset=instr_seg_endo_images,␣
↪batch_size=10)
part_instr_seg_endo_data = MONAIDataLoader(dataset=part_instr_seg_endo_images,␣
↪batch_size=10)

binary_basic_UNet_model = basic_UNet_Train(num_classes=1)
part_seg_basic_UNet_model = basic_UNet_Train(num_classes=5)
instr_seg_basic_UNet_model = basic_UNet_Train(num_classes=8)
part_instr_seg_basic_UNet_model = basic_UNet_Train(num_classes=21)
```

```
Train dataset size: 1440
Validation dataset size: 360
Train dataset size: 1440
Validation dataset size: 360
Train dataset size: 1440
Validation dataset size: 360
Train dataset size: 1440
Validation dataset size: 360
num_classes 1 1 1
BasicUNet features: (32, 64, 128, 256, 512, 32).
num_classes 5 5 5
BasicUNet features: (32, 64, 128, 256, 512, 32).
num_classes 8 8 8
BasicUNet features: (32, 64, 128, 256, 512, 32).
```

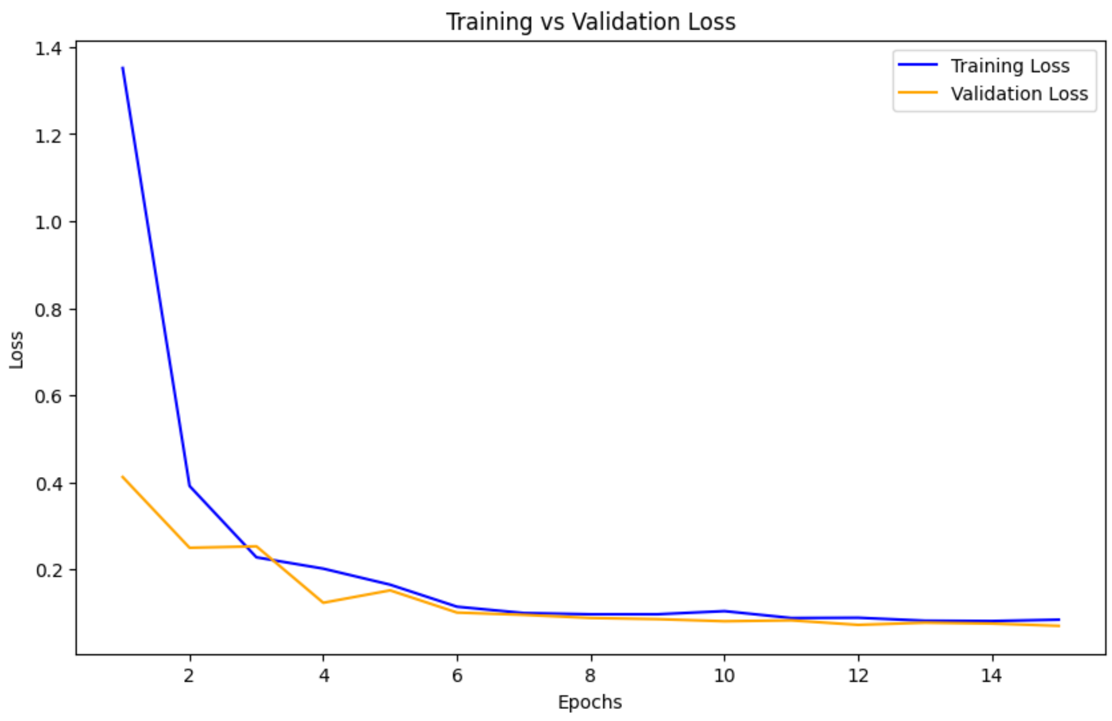
```
num_classes 21 21 21
BasicUNet features: (32, 64, 128, 256, 512, 32).
```

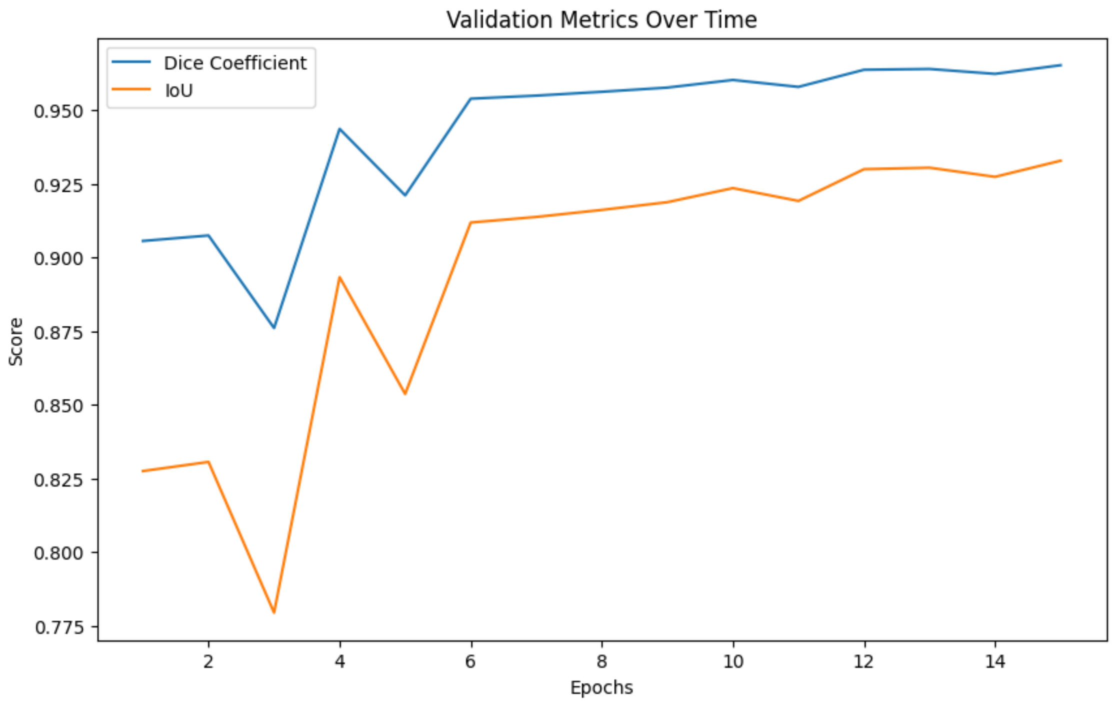
[47]:
```python
if __name__ == "__main__":

    logger = TensorBoardLogger("tb_logs", name="binary_seg")

    early_stop_callback = EarlyStopping(
        monitor="Train_Dice_CE_loss",           # metric name from self.log
        mode="min",                        # because lower loss is better
        patience=5,                        # epochs to wait before stopping
        verbose=True
    )

    checkpoint_callback = ModelCheckpoint(
        monitor="Train_Dice_CE_loss",
        mode="min",
        save_top_k=1,
        dirpath="checkpoints/",
        filename="best-part-seg-basic-unet",
    )

    trainer = Trainer(
        accelerator="gpu",
        max_epochs=15,
        #limit_train_batches=0.1,  # or 0.1 to use 10%
        logger=logger,
        callbacks=[early_stop_callback, checkpoint_callback],
    )

    start_train = time.time()
    trainer.fit(
        model=binary_basic_UNet_model,
        datamodule=binary_endo_data
    )
    end_train = time.time()
    print(f"Training time: {(end_train - start_train)/60:.2f} minutes")

    # Plot the overlaid training and val loss curves per epoch
    binary_basic_UNet_model.plot_losses()

    # Plot the IOU and DSC curves per epoch
    binary_basic_UNet_model.plot_metrics()

    # Plot images from last epoch
    binary_basic_UNet_model.plot_result_by_epoch()
```

```
GPU available: True (cuda), used: True
```

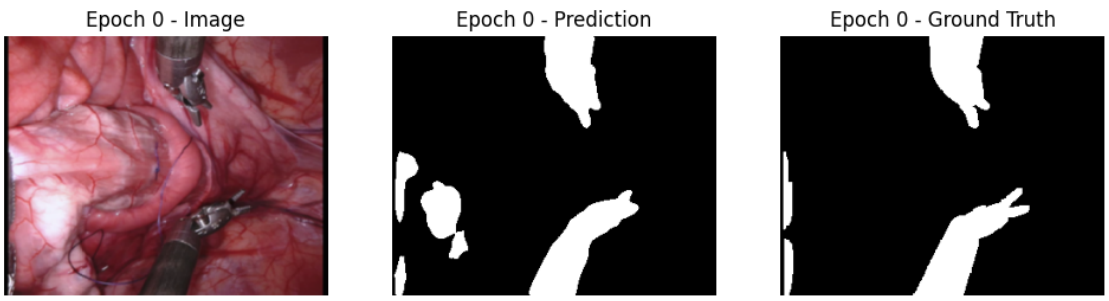
```
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
Checkpoint directory C:\Users\dsumm\OneDrive\Documents\UMD ENPM Robotics
Files\BIOE658B (Intro to Medical Image Analysis)\Project\code\checkpoints exists
and is not empty.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name         | Type       | Params | Mode  | In sizes         | Out sizes
--------------------------------------------------------------------------------
----------
0 | model        | BasicUNet  | 7.8 M  | train | [1, 3, 256, 320] | [1, 1, 256,
320]
1 | DICE_CE_Loss | DiceCELoss | 0      | train | ?                | ?
--------------------------------------------------------------------------------
----------
7.8 M     Trainable params
0         Non-trainable params
7.8 M     Total params
31.134    Total estimated model params size (MB)
143       Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

The 'val_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.

Sanity Checking DataLoader 0:  50%|        | 1/2 [00:00<00:00, 13.15it/s]

single channel prediction, `include_background=False` ignored.


The 'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.

Epoch 0: 100%|     | 144/144 [01:43<00:00,  1.39it/s, v_num=5,
Train_Dice_CE_loss_step=0.370]TE 15
Logged image from epoch 0
Epoch 0: 100%|     | 144/144 [02:00<00:00,  1.20it/s, v_num=5,
Train_Dice_CE_loss_step=0.370, val_loss=0.458, val_dice=0.881, val_iou=0.789,
Train_Dice_CE_loss_epoch=0.731]

Metric Train_Dice_CE_loss improved. New best score: 0.731

Epoch 1: 100%|     | 144/144 [01:50<00:00,  1.31it/s, v_num=5,
Train_Dice_CE_loss_step=0.204, val_loss=0.458, val_dice=0.881, val_iou=0.789,
Train_Dice_CE_loss_epoch=0.731]TE 15
Epoch 1: 100%|     | 144/144 [02:06<00:00,  1.14it/s, v_num=5,
Train_Dice_CE_loss_step=0.204, val_loss=0.282, val_dice=0.898, val_iou=0.817,
Train_Dice_CE_loss_epoch=0.362]

Metric Train_Dice_CE_loss improved by 0.369 >= min_delta = 0.0. New best score:
0.362

Epoch 2: 100%|      | 144/144 [01:31<00:00,  1.57it/s, v_num=5,
Train_Dice_CE_loss_step=0.184, val_loss=0.282, val_dice=0.898, val_iou=0.817,
Train_Dice_CE_loss_epoch=0.362]TE 15
Epoch 2: 100%|      | 144/144 [01:46<00:00,  1.35it/s, v_num=5,
Train_Dice_CE_loss_step=0.184, val_loss=0.271, val_dice=0.885, val_iou=0.796,
Train_Dice_CE_loss_epoch=0.252]

Metric Train_Dice_CE_loss improved by 0.111 >= min_delta = 0.0. New best score:
0.252

Epoch 3: 100%|      | 144/144 [01:30<00:00,  1.59it/s, v_num=5,
Train_Dice_CE_loss_step=0.117, val_loss=0.271, val_dice=0.885, val_iou=0.796,
Train_Dice_CE_loss_epoch=0.252]TE 15
Epoch 3: 100%|      | 144/144 [01:46<00:00,  1.36it/s, v_num=5,
Train_Dice_CE_loss_step=0.117, val_loss=0.185, val_dice=0.924, val_iou=0.859,
Train_Dice_CE_loss_epoch=0.206]

Metric Train_Dice_CE_loss improved by 0.046 >= min_delta = 0.0. New best score:
0.206

Epoch 4: 100%|      | 144/144 [01:31<00:00,  1.57it/s, v_num=5,
Train_Dice_CE_loss_step=0.128, val_loss=0.185, val_dice=0.924, val_iou=0.859,
Train_Dice_CE_loss_epoch=0.206]TE 15
Epoch 4: 100%|      | 144/144 [01:47<00:00,  1.34it/s, v_num=5,
Train_Dice_CE_loss_step=0.128, val_loss=0.183, val_dice=0.922, val_iou=0.855,
Train_Dice_CE_loss_epoch=0.177]

Metric Train_Dice_CE_loss improved by 0.028 >= min_delta = 0.0. New best score:
0.177

Epoch 5: 100%|      | 144/144 [01:35<00:00,  1.51it/s, v_num=5,
Train_Dice_CE_loss_step=0.0861, val_loss=0.183, val_dice=0.922, val_iou=0.855,
Train_Dice_CE_loss_epoch=0.177]TE 15
Epoch 5: 100%|      | 144/144 [01:52<00:00,  1.28it/s, v_num=5,
Train_Dice_CE_loss_step=0.0861, val_loss=0.142, val_dice=0.944, val_iou=0.894,
Train_Dice_CE_loss_epoch=0.161]

Metric Train_Dice_CE_loss improved by 0.016 >= min_delta = 0.0. New best score:
0.161

Epoch 6: 100%|      | 144/144 [01:42<00:00,  1.40it/s, v_num=5,
Train_Dice_CE_loss_step=0.079, val_loss=0.142, val_dice=0.944, val_iou=0.894,
Train_Dice_CE_loss_epoch=0.161] TE 15
Epoch 6: 100%|      | 144/144 [01:59<00:00,  1.21it/s, v_num=5,
Train_Dice_CE_loss_step=0.079, val_loss=0.130, val_dice=0.948, val_iou=0.902,
Train_Dice_CE_loss_epoch=0.143]
```

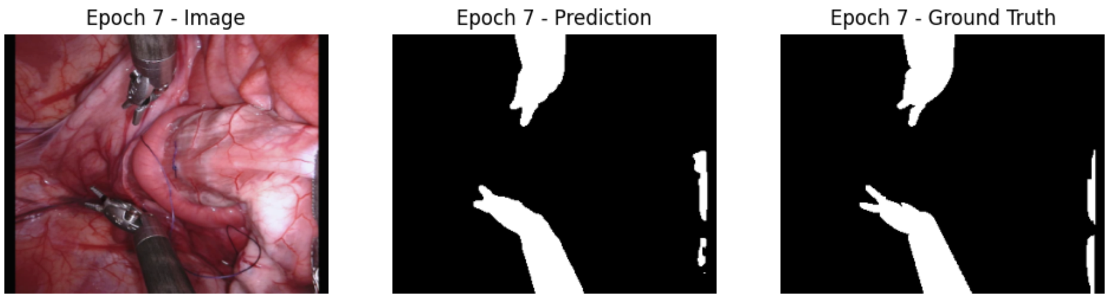
```
Metric Train_Dice_CE_loss improved by 0.018 >= min_delta = 0.0. New best score:
0.143

Epoch 7: 100%|      | 144/144 [01:41<00:00,  1.43it/s, v_num=5,
Train_Dice_CE_loss_step=0.0874, val_loss=0.130, val_dice=0.948, val_iou=0.902,
Train_Dice_CE_loss_epoch=0.143]TE 15
Logged image from epoch 7
Epoch 7: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=5,
Train_Dice_CE_loss_step=0.0874, val_loss=0.123, val_dice=0.950, val_iou=0.905,
Train_Dice_CE_loss_epoch=0.141]

Metric Train_Dice_CE_loss improved by 0.002 >= min_delta = 0.0. New best score:
0.141

Epoch 8: 100%|      | 144/144 [01:39<00:00,  1.45it/s, v_num=5,
Train_Dice_CE_loss_step=0.0855, val_loss=0.123, val_dice=0.950, val_iou=0.905,
Train_Dice_CE_loss_epoch=0.141]TE 15
Epoch 8: 100%|      | 144/144 [01:54<00:00,  1.25it/s, v_num=5,
Train_Dice_CE_loss_step=0.0855, val_loss=0.115, val_dice=0.954, val_iou=0.912,
Train_Dice_CE_loss_epoch=0.129]

Metric Train_Dice_CE_loss improved by 0.012 >= min_delta = 0.0. New best score:
0.129

Epoch 9: 100%|      | 144/144 [01:39<00:00,  1.45it/s, v_num=5,
Train_Dice_CE_loss_step=0.0776, val_loss=0.115, val_dice=0.954, val_iou=0.912,
Train_Dice_CE_loss_epoch=0.129]TE 15
Epoch 10: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=5,
Train_Dice_CE_loss_step=0.0722, val_loss=0.109, val_dice=0.956, val_iou=0.916,
Train_Dice_CE_loss_epoch=0.129]TE 15
Epoch 10: 100%|      | 144/144 [01:54<00:00,  1.26it/s, v_num=5,
Train_Dice_CE_loss_step=0.0722, val_loss=0.113, val_dice=0.954, val_iou=0.912,
Train_Dice_CE_loss_epoch=0.121]

Metric Train_Dice_CE_loss improved by 0.008 >= min_delta = 0.0. New best score:
0.121

Epoch 11: 100%|      | 144/144 [01:34<00:00,  1.52it/s, v_num=5,
Train_Dice_CE_loss_step=0.0867, val_loss=0.113, val_dice=0.954, val_iou=0.912,
Train_Dice_CE_loss_epoch=0.121]TE 15
Epoch 11: 100%|      | 144/144 [01:50<00:00,  1.31it/s, v_num=5,
Train_Dice_CE_loss_step=0.0867, val_loss=0.107, val_dice=0.956, val_iou=0.916,
Train_Dice_CE_loss_epoch=0.118]

Metric Train_Dice_CE_loss improved by 0.003 >= min_delta = 0.0. New best score:
0.118

Epoch 12: 100%|      | 144/144 [01:36<00:00,  1.49it/s, v_num=5,
Train_Dice_CE_loss_step=0.0848, val_loss=0.107, val_dice=0.956, val_iou=0.916,
Train_Dice_CE_loss_epoch=0.118]TE 15
Epoch 12: 100%|      | 144/144 [01:51<00:00,  1.29it/s, v_num=5,
Train_Dice_CE_loss_step=0.0848, val_loss=0.110, val_dice=0.956, val_iou=0.916,
Train_Dice_CE_loss_epoch=0.117]

Metric Train_Dice_CE_loss improved by 0.001 >= min_delta = 0.0. New best score:
0.117

Epoch 13: 100%|     | 144/144 [01:39<00:00,  1.45it/s, v_num=5,
Train_Dice_CE_loss_step=0.0725, val_loss=0.110, val_dice=0.956, val_iou=0.916,
Train_Dice_CE_loss_epoch=0.117]TE 15
Epoch 13: 100%|     | 144/144 [01:55<00:00,  1.24it/s, v_num=5,
Train_Dice_CE_loss_step=0.0725, val_loss=0.105, val_dice=0.958, val_iou=0.919,
Train_Dice_CE_loss_epoch=0.116]

Metric Train_Dice_CE_loss improved by 0.002 >= min_delta = 0.0. New best score:
0.116

Epoch 14: 100%|     | 144/144 [01:39<00:00,  1.45it/s, v_num=5,
Train_Dice_CE_loss_step=0.0758, val_loss=0.105, val_dice=0.958, val_iou=0.919,
Train_Dice_CE_loss_epoch=0.116]TE 15
Logged image from epoch 14
Epoch 14: 100%|     | 144/144 [01:55<00:00,  1.25it/s, v_num=5,
Train_Dice_CE_loss_step=0.0758, val_loss=0.102, val_dice=0.959, val_iou=0.921,
Train_Dice_CE_loss_epoch=0.111]

Metric Train_Dice_CE_loss improved by 0.004 >= min_delta = 0.0. New best score:
0.111
`Trainer.fit` stopped: `max_epochs=15` reached.

Epoch 14: 100%|     | 144/144 [01:55<00:00,  1.24it/s, v_num=5,
Train_Dice_CE_loss_step=0.0758, val_loss=0.102, val_dice=0.959, val_iou=0.921,
Train_Dice_CE_loss_epoch=0.111]
Training time: 28.69 minutes
```

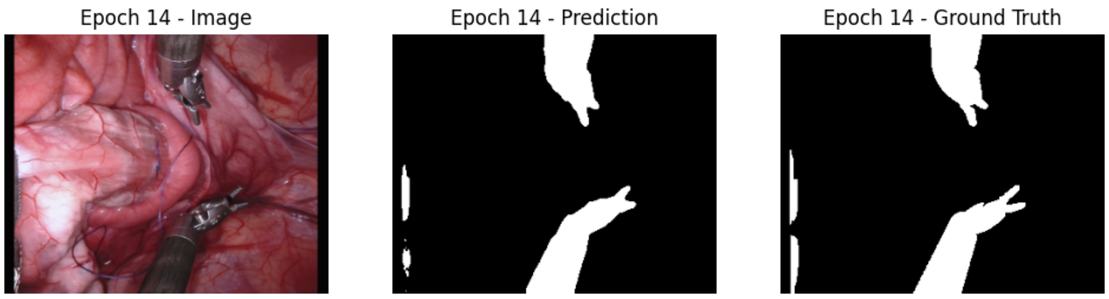




```
Only 3 epochs recorded, plotting all.
```





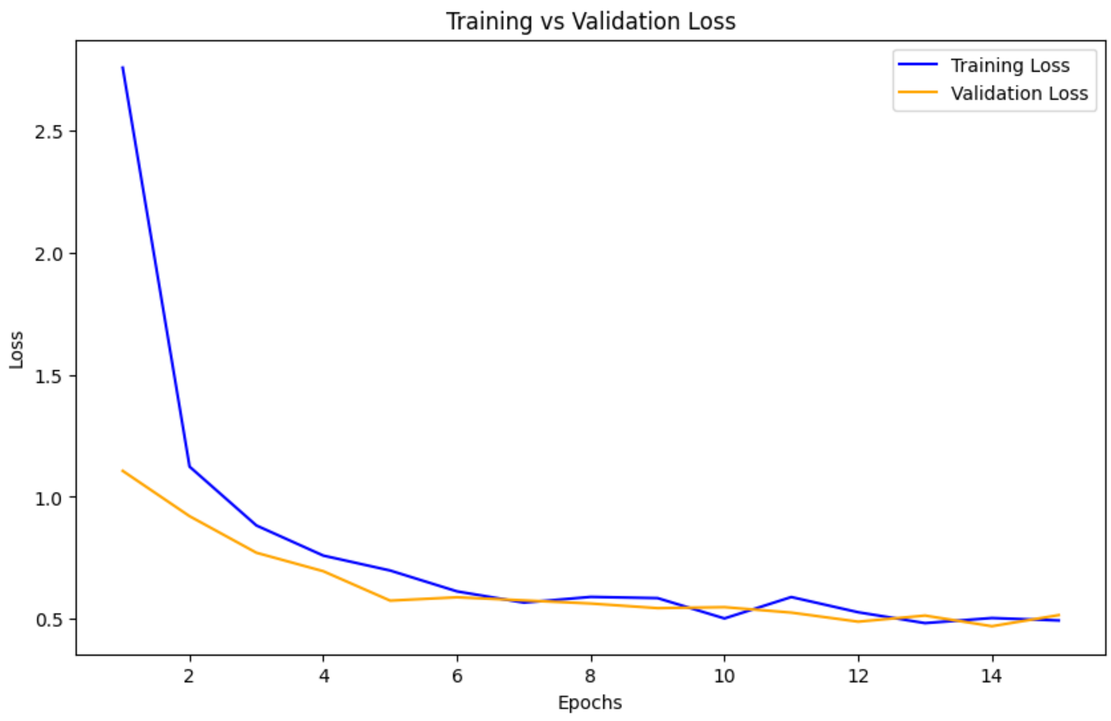


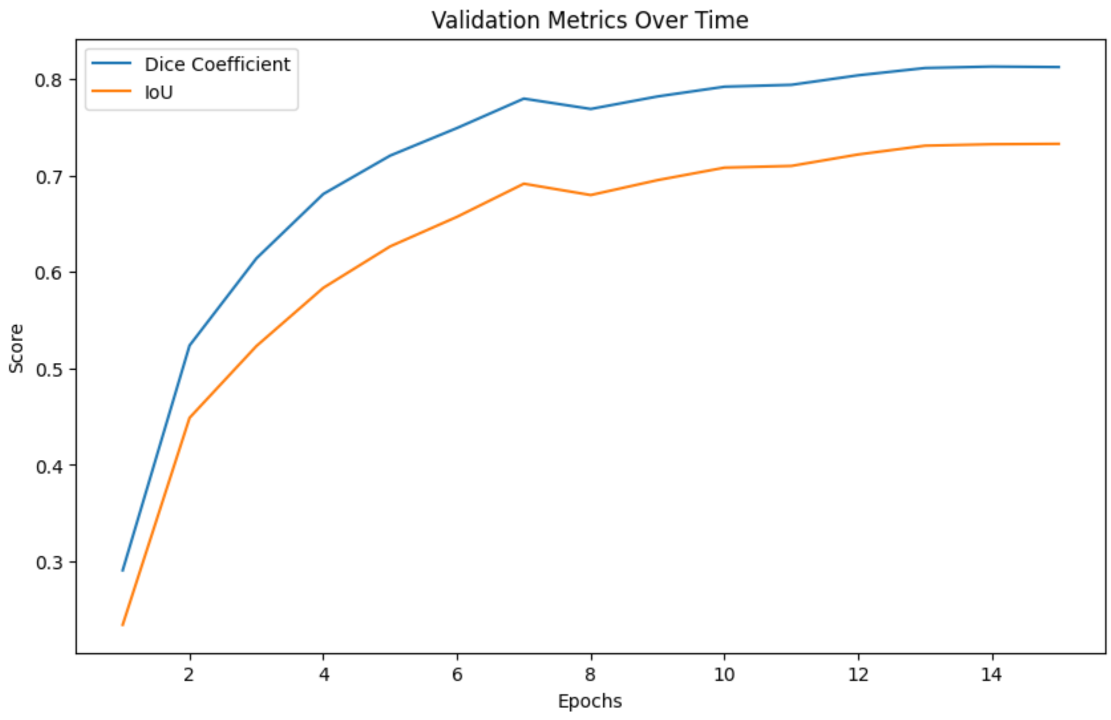
[48]:
```python
if __name__ == "__main__":

    logger = TensorBoardLogger("tb_logs", name="part_seg")

    early_stop_callback = EarlyStopping(
        monitor="Train_Dice_CE_loss",           # metric name from self.log
        mode="min",                        # because lower loss is better
        patience=5,                        # epochs to wait before stopping
        verbose=True
    )

    checkpoint_callback = ModelCheckpoint(
        monitor="Train_Dice_CE_loss",
        mode="min",
        save_top_k=1,
        dirpath="checkpoints/",
        filename="best-part-seg-basic-unet",
    )

    trainer = Trainer(
        accelerator="gpu",
        max_epochs=15,
        #limit_train_batches=0.1,  # or 0.1 to use 10%
        logger=logger,
        callbacks=[early_stop_callback, checkpoint_callback],
    )

    start_train = time.time()
    trainer.fit(
        model=part_seg_basic_UNet_model,
        datamodule=part_seg_endo_data
    )
    end_train = time.time()
    print(f"Training time: {(end_train - start_train)/60:.2f} minutes")

    # Plot the overlaid training and val loss curves per epoch
    part_seg_basic_UNet_model.plot_losses()

    # Plot the IOU and DSC curves per epoch
    part_seg_basic_UNet_model.plot_metrics()

    # Plot images from last epoch
    part_seg_basic_UNet_model.plot_result_by_epoch()
```

```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name          | Type        | Params | Mode  | In sizes          | Out sizes
--------------------------------------------------------------------------------
----------
0 | model         | BasicUNet   | 7.8 M  | train | [1, 3, 256, 320] | [1, 5, 256,
320]
1 | DICE_CE_Loss | DiceCELoss | 0      | train | ?                 | ?
--------------------------------------------------------------------------------
----------
7.8 M     Trainable params
0         Non-trainable params
7.8 M     Total params
```

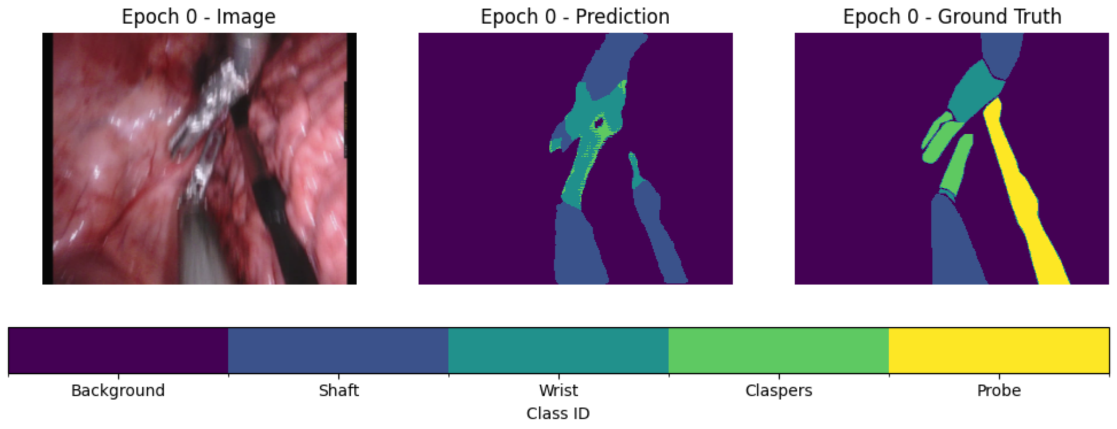
```
31.134    Total estimated model params size (MB)
143       Modules in train mode
0         Modules in eval mode

Epoch 0: 100%|      | 144/144 [01:40<00:00,  1.43it/s, v_num=6,
Train_Dice_CE_loss_step=1.120]TE 15
Logged image from epoch 0
Epoch 0: 100%|      | 144/144 [02:00<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=1.120, val_loss=1.130, val_dice=0.291, val_iou=0.234,
Train_Dice_CE_loss_epoch=1.660]

Metric Train_Dice_CE_loss improved. New best score: 1.664

Epoch 1: 100%|      | 144/144 [01:39<00:00,  1.45it/s, v_num=6,
Train_Dice_CE_loss_step=0.857, val_loss=1.130, val_dice=0.291, val_iou=0.234,
Train_Dice_CE_loss_epoch=1.660]TE 15
Epoch 1: 100%|      | 144/144 [01:58<00:00,  1.21it/s, v_num=6,
Train_Dice_CE_loss_step=0.857, val_loss=0.913, val_dice=0.524, val_iou=0.449,
Train_Dice_CE_loss_epoch=0.980]

Metric Train_Dice_CE_loss improved by 0.684 >= min_delta = 0.0. New best score:
0.980

Epoch 2: 100%|      | 144/144 [01:36<00:00,  1.49it/s, v_num=6,
Train_Dice_CE_loss_step=0.690, val_loss=0.913, val_dice=0.524, val_iou=0.449,
Train_Dice_CE_loss_epoch=0.980]TE 15
Epoch 2: 100%|      | 144/144 [01:55<00:00,  1.24it/s, v_num=6,
Train_Dice_CE_loss_step=0.690, val_loss=0.804, val_dice=0.614, val_iou=0.523,
Train_Dice_CE_loss_epoch=0.828]

Metric Train_Dice_CE_loss improved by 0.152 >= min_delta = 0.0. New best score:
0.828

Epoch 3: 100%|      | 144/144 [01:37<00:00,  1.47it/s, v_num=6,
Train_Dice_CE_loss_step=0.681, val_loss=0.804, val_dice=0.614, val_iou=0.523,
Train_Dice_CE_loss_epoch=0.828]TE 15
Epoch 3: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=6,
Train_Dice_CE_loss_step=0.681, val_loss=0.709, val_dice=0.681, val_iou=0.583,
Train_Dice_CE_loss_epoch=0.749]

Metric Train_Dice_CE_loss improved by 0.079 >= min_delta = 0.0. New best score:
0.749

Epoch 4: 100%|      | 144/144 [01:41<00:00,  1.42it/s, v_num=6,
Train_Dice_CE_loss_step=0.630, val_loss=0.709, val_dice=0.681, val_iou=0.583,
Train_Dice_CE_loss_epoch=0.749]TE 15
Epoch 4: 100%|      | 144/144 [02:00<00:00,  1.19it/s, v_num=6,
Train_Dice_CE_loss_step=0.630, val_loss=0.649, val_dice=0.721, val_iou=0.626,
Train_Dice_CE_loss_epoch=0.672]

Metric Train_Dice_CE_loss improved by 0.077 >= min_delta = 0.0. New best score:
0.672
```

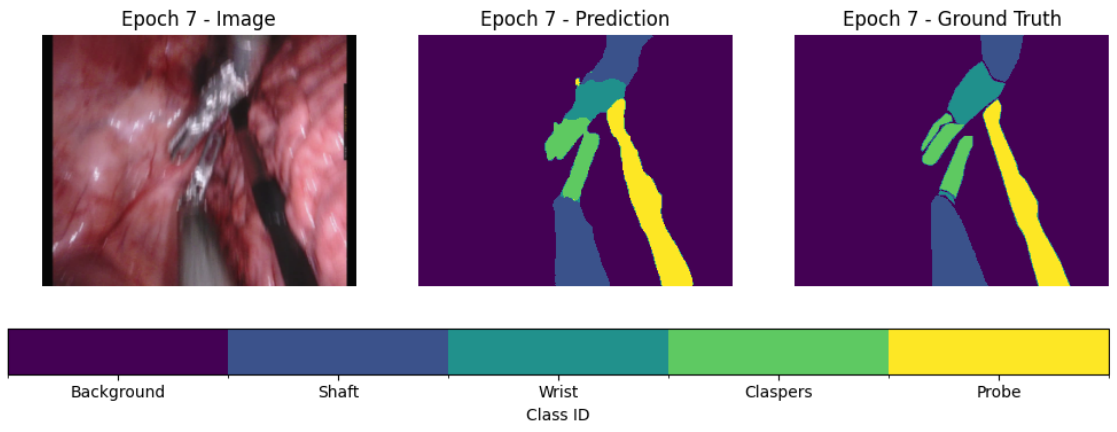
```
Epoch 5: 100%|     | 144/144 [01:38<00:00,  1.46it/s, v_num=6,
Train_Dice_CE_loss_step=0.512, val_loss=0.649, val_dice=0.721, val_iou=0.626,
Train_Dice_CE_loss_epoch=0.672]TE 15
Epoch 5: 100%|     | 144/144 [01:58<00:00,  1.22it/s, v_num=6,
Train_Dice_CE_loss_step=0.512, val_loss=0.596, val_dice=0.749, val_iou=0.657,
Train_Dice_CE_loss_epoch=0.615]

Metric Train_Dice_CE_loss improved by 0.057 >= min_delta = 0.0. New best score:
0.615

Epoch 6: 100%|     | 144/144 [01:38<00:00,  1.46it/s, v_num=6,
Train_Dice_CE_loss_step=0.504, val_loss=0.596, val_dice=0.749, val_iou=0.657,
Train_Dice_CE_loss_epoch=0.615]TE 15
Epoch 6: 100%|     | 144/144 [01:58<00:00,  1.22it/s, v_num=6,
Train_Dice_CE_loss_step=0.504, val_loss=0.610, val_dice=0.780, val_iou=0.691,
Train_Dice_CE_loss_epoch=0.582]

Metric Train_Dice_CE_loss improved by 0.033 >= min_delta = 0.0. New best score:
0.582

Epoch 7: 100%|     | 144/144 [01:38<00:00,  1.46it/s, v_num=6,
Train_Dice_CE_loss_step=0.523, val_loss=0.610, val_dice=0.780, val_iou=0.691,
Train_Dice_CE_loss_epoch=0.582]TE 15
Logged image from epoch 7
Epoch 7: 100%|     | 144/144 [01:58<00:00,  1.21it/s, v_num=6,
Train_Dice_CE_loss_step=0.523, val_loss=0.583, val_dice=0.769, val_iou=0.680,
Train_Dice_CE_loss_epoch=0.570]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
0.570

Epoch 8: 100%|     | 144/144 [01:39<00:00,  1.44it/s, v_num=6,
Train_Dice_CE_loss_step=0.458, val_loss=0.583, val_dice=0.769, val_iou=0.680,
Train_Dice_CE_loss_epoch=0.570]TE 15
Epoch 8: 100%|     | 144/144 [01:59<00:00,  1.21it/s, v_num=6,
Train_Dice_CE_loss_step=0.458, val_loss=0.565, val_dice=0.782, val_iou=0.695,
Train_Dice_CE_loss_epoch=0.553]

Metric Train_Dice_CE_loss improved by 0.017 >= min_delta = 0.0. New best score:
0.553

Epoch 9: 100%|     | 144/144 [01:39<00:00,  1.44it/s, v_num=6,
Train_Dice_CE_loss_step=0.435, val_loss=0.565, val_dice=0.782, val_iou=0.695,
Train_Dice_CE_loss_epoch=0.553]TE 15
Epoch 9: 100%|     | 144/144 [01:59<00:00,  1.21it/s, v_num=6,
Train_Dice_CE_loss_step=0.435, val_loss=0.567, val_dice=0.792, val_iou=0.708,
Train_Dice_CE_loss_epoch=0.547]

Metric Train_Dice_CE_loss improved by 0.006 >= min_delta = 0.0. New best score:
0.547

Epoch 10: 100%|     | 144/144 [01:39<00:00,  1.44it/s, v_num=6,
Train_Dice_CE_loss_step=0.422, val_loss=0.567, val_dice=0.792, val_iou=0.708,
Train_Dice_CE_loss_epoch=0.547]TE 15
Epoch 10: 100%|      | 144/144 [01:59<00:00,  1.21it/s, v_num=6,
Train_Dice_CE_loss_step=0.422, val_loss=0.537, val_dice=0.794, val_iou=0.710,
Train_Dice_CE_loss_epoch=0.523]

Metric Train_Dice_CE_loss improved by 0.024 >= min_delta = 0.0. New best score:
0.523

Epoch 11: 100%|      | 144/144 [01:40<00:00,  1.43it/s, v_num=6,
Train_Dice_CE_loss_step=0.443, val_loss=0.537, val_dice=0.794, val_iou=0.710,
Train_Dice_CE_loss_epoch=0.523]TE 15
Epoch 11: 100%|      | 144/144 [02:00<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=0.443, val_loss=0.522, val_dice=0.804, val_iou=0.722,
Train_Dice_CE_loss_epoch=0.510]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
0.510

Epoch 12: 100%|      | 144/144 [01:40<00:00,  1.44it/s, v_num=6,
Train_Dice_CE_loss_step=0.467, val_loss=0.522, val_dice=0.804, val_iou=0.722,
Train_Dice_CE_loss_epoch=0.510]TE 15
Epoch 12: 100%|      | 144/144 [01:59<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=0.467, val_loss=0.519, val_dice=0.811, val_iou=0.731,
Train_Dice_CE_loss_epoch=0.506]

Metric Train_Dice_CE_loss improved by 0.004 >= min_delta = 0.0. New best score:
0.506

Epoch 13: 100%|      | 144/144 [01:40<00:00,  1.43it/s, v_num=6,
Train_Dice_CE_loss_step=0.420, val_loss=0.519, val_dice=0.811, val_iou=0.731,
Train_Dice_CE_loss_epoch=0.506]TE 15
Epoch 13: 100%|      | 144/144 [01:59<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=0.420, val_loss=0.516, val_dice=0.813, val_iou=0.732,
Train_Dice_CE_loss_epoch=0.499]

Metric Train_Dice_CE_loss improved by 0.007 >= min_delta = 0.0. New best score:
0.499

Epoch 14: 100%|      | 144/144 [01:39<00:00,  1.44it/s, v_num=6,
Train_Dice_CE_loss_step=0.431, val_loss=0.516, val_dice=0.813, val_iou=0.732,
Train_Dice_CE_loss_epoch=0.499]TE 15
Logged image from epoch 14
Epoch 14: 100%|      | 144/144 [01:59<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=0.431, val_loss=0.517, val_dice=0.812, val_iou=0.733,
Train_Dice_CE_loss_epoch=0.495]

Metric Train_Dice_CE_loss improved by 0.003 >= min_delta = 0.0. New best score:
0.495
`Trainer.fit` stopped: `max_epochs=15` reached.

Epoch 14: 100%|      | 144/144 [01:59<00:00,  1.20it/s, v_num=6,
Train_Dice_CE_loss_step=0.431, val_loss=0.517, val_dice=0.812, val_iou=0.733,
Train_Dice_CE_loss_epoch=0.495]
Training time: 29.88 minutes
```

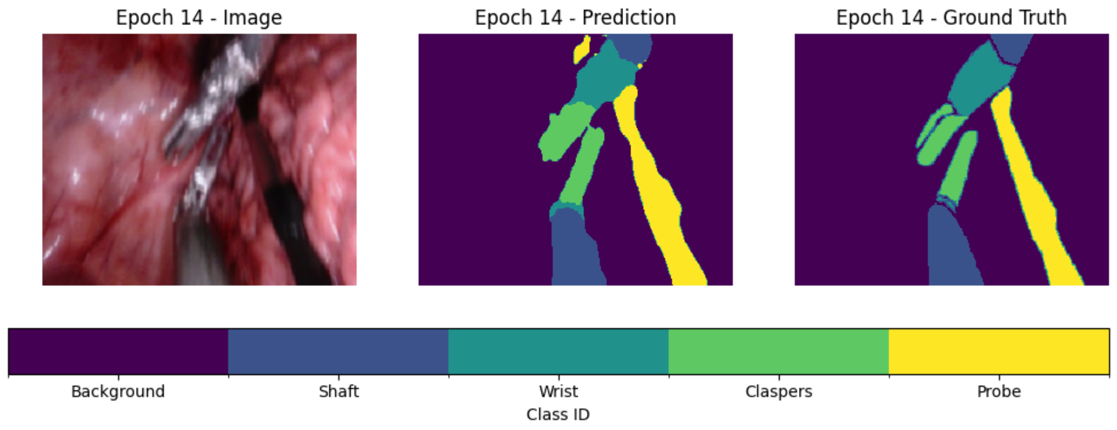




```
Only 3 epochs recorded, plotting all.
```







[49]:
```python
if __name__ == "__main__":

    logger = TensorBoardLogger("tb_logs", name="instrument_seg")

    early_stop_callback = EarlyStopping(
        monitor="Train_Dice_CE_loss",          # metric name from self.log
        mode="min",                      # because lower loss is better
        patience=5,                      # epochs to wait before stopping
        verbose=True
    )

    checkpoint_callback = ModelCheckpoint(
        monitor="Train_Dice_CE_loss",
        mode="min",
        save_top_k=1,
        dirpath="checkpoints/",
        filename="best-instrument-seg-basic-unet",
    )

    trainer = Trainer(
        accelerator="gpu",
        max_epochs=20,
        #limit_train_batches=0.1,  # or 0.1 to use 10%
        logger=logger,
        callbacks=[early_stop_callback, checkpoint_callback],
    )

    start_train = time.time()
    trainer.fit(
        model=instr_seg_basic_UNet_model,
        datamodule=instr_seg_endo_data
    )
    end_train = time.time()
    print(f"Training time: {(end_train - start_train)/60:.2f} minutes")

    # Plot the overlaid training and val loss curves per epoch
    instr_seg_basic_UNet_model.plot_losses()

    # Plot the IOU and DSC curves per epoch
    instr_seg_basic_UNet_model.plot_metrics()

    # Plot images from last epoch
    instr_seg_basic_UNet_model.plot_result_by_epoch()
```

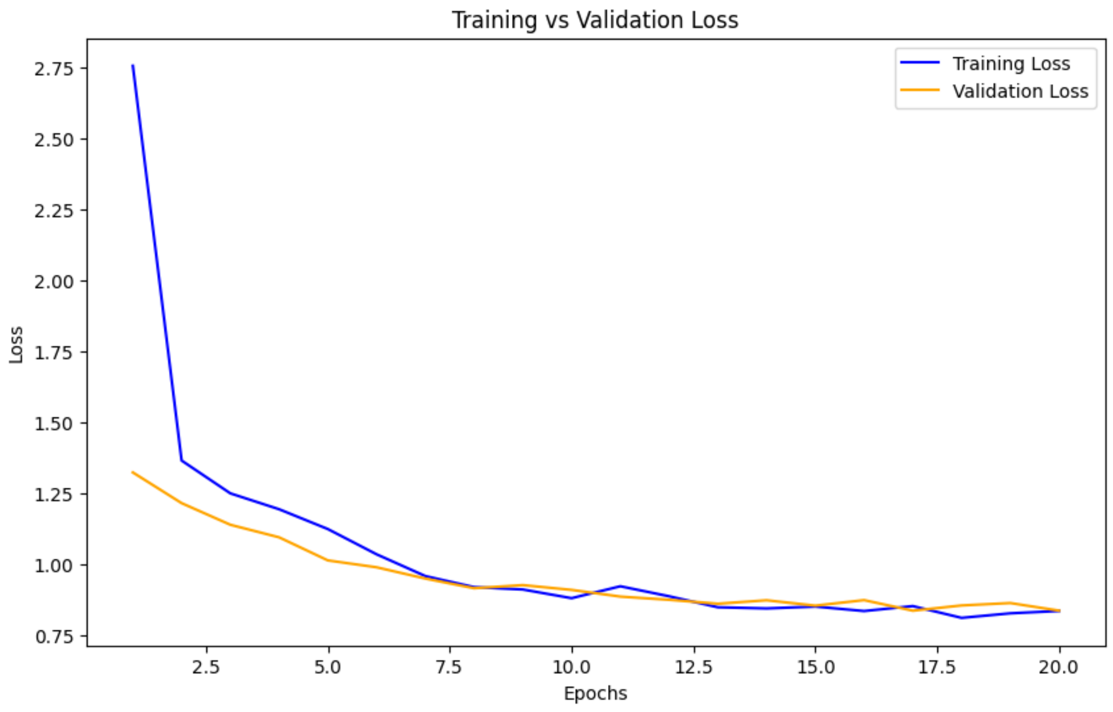
```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
```

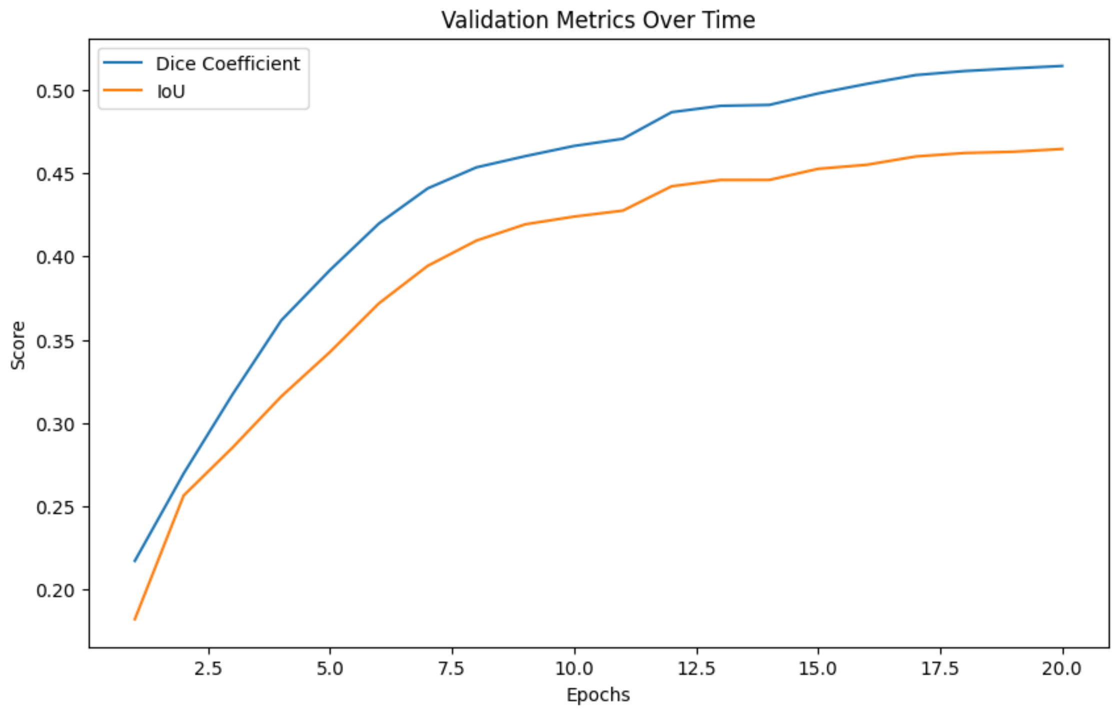
```
HPU available: False, using: 0 HPUs
Checkpoint directory C:\Users\dsumm\OneDrive\Documents\UMD ENPM Robotics
Files\BIOE658B (Intro to Medical Image Analysis)\Project\code\checkpoints exists
and is not empty.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name         | Type       | Params | Mode  | In sizes         | Out sizes
--------------------------------------------------------------------------------
----------
0 | model        | BasicUNet  | 7.8 M  | train | [1, 3, 256, 320] | [1, 8, 256,
320]
1 | DICE_CE_Loss | DiceCELoss | 0      | train | ?                | ?
--------------------------------------------------------------------------------
----------
7.8 M     Trainable params
0         Non-trainable params
7.8 M     Total params
31.135    Total estimated model params size (MB)
143       Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

The 'val_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.


The 'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.

Epoch 0: 100%|     | 144/144 [01:49<00:00,  1.32it/s, v_num=0,
Train_Dice_CE_loss_step=1.360]TE 20
Logged image from epoch 0
Epoch 0: 100%|     | 144/144 [02:11<00:00,  1.10it/s, v_num=0,
Train_Dice_CE_loss_step=1.360, val_loss=1.340, val_dice=0.217, val_iou=0.182,
Train_Dice_CE_loss_epoch=1.700]

Metric Train_Dice_CE_loss improved. New best score: 1.695

Epoch 1: 100%|     | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=1.280, val_loss=1.340, val_dice=0.217, val_iou=0.182,
Train_Dice_CE_loss_epoch=1.700]TE 20
Epoch 1: 100%|     | 144/144 [01:58<00:00,  1.21it/s, v_num=0,
Train_Dice_CE_loss_step=1.280, val_loss=1.240, val_dice=0.270, val_iou=0.256,
Train_Dice_CE_loss_epoch=1.300]

Metric Train_Dice_CE_loss improved by 0.395 >= min_delta = 0.0. New best score:
1.300
```

```
Epoch 2: 100%|     | 144/144 [01:39<00:00,  1.45it/s, v_num=0,
Train_Dice_CE_loss_step=1.210, val_loss=1.240, val_dice=0.270, val_iou=0.256,
Train_Dice_CE_loss_epoch=1.300]TE 20
Epoch 2: 100%|     | 144/144 [01:58<00:00,  1.21it/s, v_num=0,
Train_Dice_CE_loss_step=1.210, val_loss=1.180, val_dice=0.317, val_iou=0.285,
Train_Dice_CE_loss_epoch=1.220]

Metric Train_Dice_CE_loss improved by 0.081 >= min_delta = 0.0. New best score:
1.219

Epoch 3: 100%|     | 144/144 [01:39<00:00,  1.45it/s, v_num=0,
Train_Dice_CE_loss_step=1.170, val_loss=1.180, val_dice=0.317, val_iou=0.285,
Train_Dice_CE_loss_epoch=1.220]TE 20
Epoch 3: 100%|     | 144/144 [01:58<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=1.170, val_loss=1.120, val_dice=0.362, val_iou=0.316,
Train_Dice_CE_loss_epoch=1.160]

Metric Train_Dice_CE_loss improved by 0.055 >= min_delta = 0.0. New best score:
1.164

Epoch 4: 100%|     | 144/144 [01:34<00:00,  1.52it/s, v_num=0,
Train_Dice_CE_loss_step=1.080, val_loss=1.120, val_dice=0.362, val_iou=0.316,
Train_Dice_CE_loss_epoch=1.160]TE 20
Epoch 4: 100%|     | 144/144 [01:53<00:00,  1.26it/s, v_num=0,
Train_Dice_CE_loss_step=1.080, val_loss=1.040, val_dice=0.392, val_iou=0.343,
Train_Dice_CE_loss_epoch=1.110]

Metric Train_Dice_CE_loss improved by 0.056 >= min_delta = 0.0. New best score:
1.108

Epoch 5: 100%|     | 144/144 [01:35<00:00,  1.51it/s, v_num=0,
Train_Dice_CE_loss_step=1.020, val_loss=1.040, val_dice=0.392, val_iou=0.343,
Train_Dice_CE_loss_epoch=1.110]TE 20
Epoch 5: 100%|     | 144/144 [01:54<00:00,  1.26it/s, v_num=0,
Train_Dice_CE_loss_step=1.020, val_loss=0.996, val_dice=0.420, val_iou=0.372,
Train_Dice_CE_loss_epoch=1.030]

Metric Train_Dice_CE_loss improved by 0.079 >= min_delta = 0.0. New best score:
1.029

Epoch 6: 100%|     | 144/144 [01:36<00:00,  1.50it/s, v_num=0,
Train_Dice_CE_loss_step=1.000, val_loss=0.996, val_dice=0.420, val_iou=0.372,
Train_Dice_CE_loss_epoch=1.030]TE 20
Epoch 6: 100%|     | 144/144 [01:55<00:00,  1.25it/s, v_num=0,
Train_Dice_CE_loss_step=1.000, val_loss=0.958, val_dice=0.441, val_iou=0.394,
Train_Dice_CE_loss_epoch=0.989]

Metric Train_Dice_CE_loss improved by 0.041 >= min_delta = 0.0. New best score:
0.989

Epoch 7: 100%|     | 144/144 [01:36<00:00,  1.49it/s, v_num=0,
Train_Dice_CE_loss_step=1.010, val_loss=0.958, val_dice=0.441, val_iou=0.394,
Train_Dice_CE_loss_epoch=0.989]TE 20
```

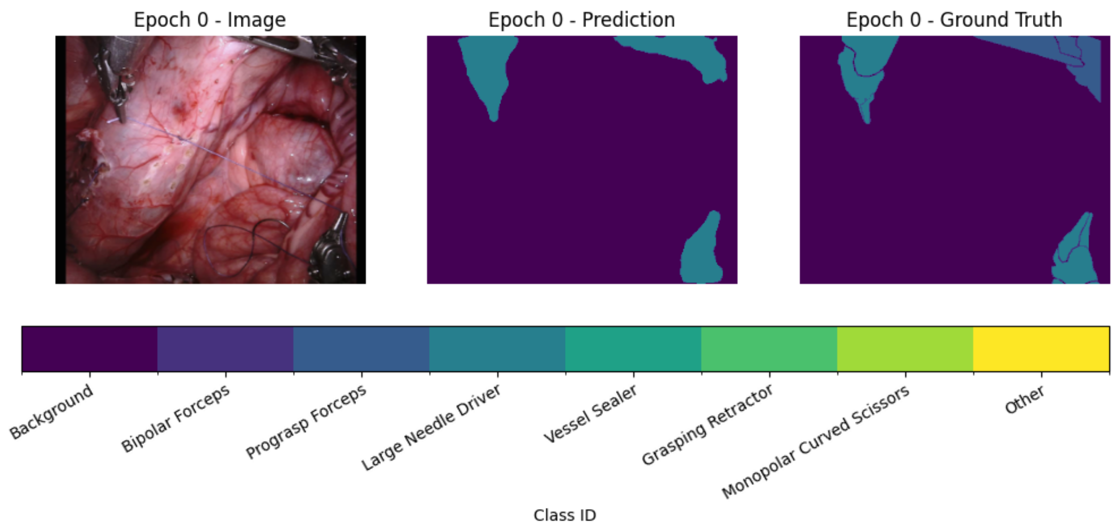
```
Epoch 7: 100%|      | 144/144 [01:55<00:00,  1.25it/s, v_num=0,
Train_Dice_CE_loss_step=1.010, val_loss=0.931, val_dice=0.453, val_iou=0.409,
Train_Dice_CE_loss_epoch=0.956]

Metric Train_Dice_CE_loss improved by 0.033 >= min_delta = 0.0. New best score:
0.956

Epoch 8: 100%|      | 144/144 [01:37<00:00,  1.47it/s, v_num=0,
Train_Dice_CE_loss_step=1.010, val_loss=0.931, val_dice=0.453, val_iou=0.409,
Train_Dice_CE_loss_epoch=0.956]TE 20
Epoch 8: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=1.010, val_loss=0.927, val_dice=0.460, val_iou=0.419,
Train_Dice_CE_loss_epoch=0.940]

Metric Train_Dice_CE_loss improved by 0.016 >= min_delta = 0.0. New best score:
0.940

Epoch 9: 100%|      | 144/144 [01:36<00:00,  1.49it/s, v_num=0,
Train_Dice_CE_loss_step=0.950, val_loss=0.927, val_dice=0.460, val_iou=0.419,
Train_Dice_CE_loss_epoch=0.940]TE 20
Epoch 9: 100%|      | 144/144 [01:55<00:00,  1.24it/s, v_num=0,
Train_Dice_CE_loss_step=0.950, val_loss=0.930, val_dice=0.466, val_iou=0.424,
Train_Dice_CE_loss_epoch=0.915]

Metric Train_Dice_CE_loss improved by 0.025 >= min_delta = 0.0. New best score:
0.915

Epoch 10: 100%|      | 144/144 [01:37<00:00,  1.48it/s, v_num=0,
Train_Dice_CE_loss_step=0.936, val_loss=0.930, val_dice=0.466, val_iou=0.424,
Train_Dice_CE_loss_epoch=0.915]TE 20
Logged image from epoch 10
Epoch 10: 100%|      | 144/144 [01:56<00:00,  1.23it/s, v_num=0,
Train_Dice_CE_loss_step=0.936, val_loss=0.893, val_dice=0.471, val_iou=0.427,
Train_Dice_CE_loss_epoch=0.890]

Metric Train_Dice_CE_loss improved by 0.025 >= min_delta = 0.0. New best score:
0.890

Epoch 11: 100%|      | 144/144 [01:37<00:00,  1.47it/s, v_num=0,
Train_Dice_CE_loss_step=0.904, val_loss=0.893, val_dice=0.471, val_iou=0.427,
Train_Dice_CE_loss_epoch=0.890]TE 20
Epoch 11: 100%|      | 144/144 [01:57<00:00,  1.23it/s, v_num=0,
Train_Dice_CE_loss_step=0.904, val_loss=0.883, val_dice=0.487, val_iou=0.442,
Train_Dice_CE_loss_epoch=0.877]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
0.877

Epoch 12: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=0.919, val_loss=0.883, val_dice=0.487, val_iou=0.442,
Train_Dice_CE_loss_epoch=0.877]TE 20
Epoch 12: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.919, val_loss=0.883, val_dice=0.490, val_iou=0.446,
Train_Dice_CE_loss_epoch=0.872]

Metric Train_Dice_CE_loss improved by 0.005 >= min_delta = 0.0. New best score:
0.872

Epoch 13: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=0.905, val_loss=0.883, val_dice=0.490, val_iou=0.446,
Train_Dice_CE_loss_epoch=0.872]TE 20
Epoch 13: 100%|      | 144/144 [01:58<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.905, val_loss=0.871, val_dice=0.491, val_iou=0.446,
Train_Dice_CE_loss_epoch=0.868]

Metric Train_Dice_CE_loss improved by 0.004 >= min_delta = 0.0. New best score:
0.868

Epoch 14: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=0.909, val_loss=0.871, val_dice=0.491, val_iou=0.446,
Train_Dice_CE_loss_epoch=0.868]TE 20
Epoch 14: 100%|      | 144/144 [01:58<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.909, val_loss=0.867, val_dice=0.498, val_iou=0.453,
Train_Dice_CE_loss_epoch=0.863]

Metric Train_Dice_CE_loss improved by 0.005 >= min_delta = 0.0. New best score:
0.863

Epoch 15: 100%|      | 144/144 [01:42<00:00,  1.40it/s, v_num=0,
Train_Dice_CE_loss_step=0.891, val_loss=0.867, val_dice=0.498, val_iou=0.453,
Train_Dice_CE_loss_epoch=0.863]TE 20
Epoch 15: 100%|      | 144/144 [02:03<00:00,  1.17it/s, v_num=0,
Train_Dice_CE_loss_step=0.891, val_loss=0.859, val_dice=0.504, val_iou=0.455,
Train_Dice_CE_loss_epoch=0.850]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
0.850

Epoch 16: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=0.875, val_loss=0.859, val_dice=0.504, val_iou=0.455,
Train_Dice_CE_loss_epoch=0.850]TE 20
Epoch 16: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.875, val_loss=0.856, val_dice=0.509, val_iou=0.460,
Train_Dice_CE_loss_epoch=0.846]

Metric Train_Dice_CE_loss improved by 0.004 >= min_delta = 0.0. New best score:
0.846

Epoch 17: 100%|      | 144/144 [01:38<00:00,  1.47it/s, v_num=0,
Train_Dice_CE_loss_step=0.893, val_loss=0.856, val_dice=0.509, val_iou=0.460,
Train_Dice_CE_loss_epoch=0.846]TE 20
Epoch 17: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.893, val_loss=0.855, val_dice=0.511, val_iou=0.462,
Train_Dice_CE_loss_epoch=0.842]
```

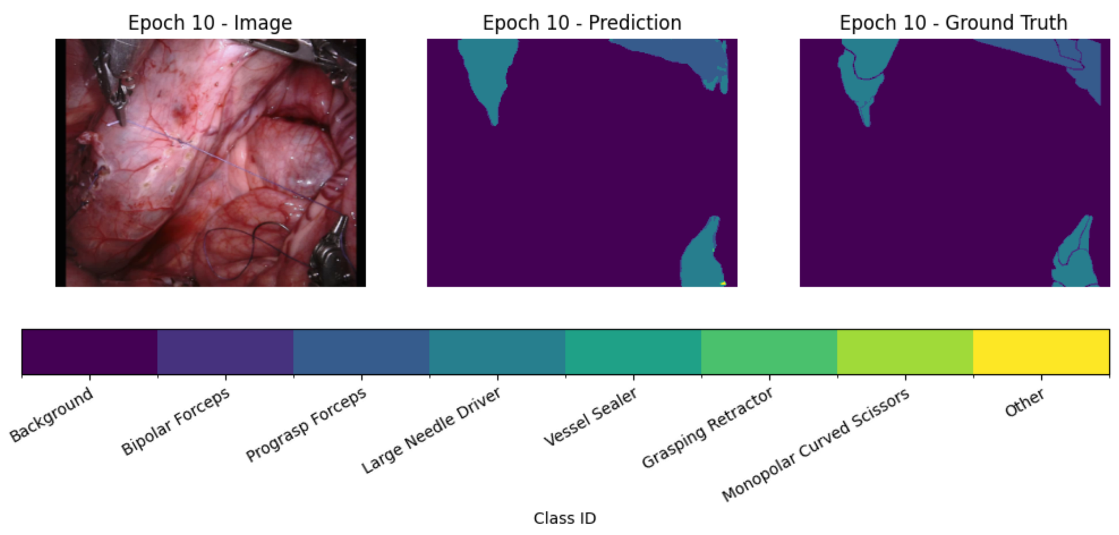
```
Metric Train_Dice_CE_loss improved by 0.004 >= min_delta = 0.0. New best score:
0.842

Epoch 18: 100%|      | 144/144 [01:38<00:00,  1.47it/s, v_num=0,
Train_Dice_CE_loss_step=0.865, val_loss=0.855, val_dice=0.511, val_iou=0.462,
Train_Dice_CE_loss_epoch=0.842]TE 20
Epoch 18: 100%|      | 144/144 [01:58<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.865, val_loss=0.850, val_dice=0.513, val_iou=0.463,
Train_Dice_CE_loss_epoch=0.838]

Metric Train_Dice_CE_loss improved by 0.003 >= min_delta = 0.0. New best score:
0.838

Epoch 19: 100%|      | 144/144 [01:38<00:00,  1.46it/s, v_num=0,
Train_Dice_CE_loss_step=0.874, val_loss=0.850, val_dice=0.513, val_iou=0.463,
Train_Dice_CE_loss_epoch=0.838]TE 20
Logged image from epoch 19
Epoch 19: 100%|      | 144/144 [01:57<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.874, val_loss=0.848, val_dice=0.514, val_iou=0.464,
Train_Dice_CE_loss_epoch=0.835]

Metric Train_Dice_CE_loss improved by 0.003 >= min_delta = 0.0. New best score:
0.835
`Trainer.fit` stopped: `max_epochs=20` reached.

Epoch 19: 100%|      | 144/144 [01:58<00:00,  1.22it/s, v_num=0,
Train_Dice_CE_loss_step=0.874, val_loss=0.848, val_dice=0.514, val_iou=0.464,
Train_Dice_CE_loss_epoch=0.835]
Training time: 39.52 minutes
```

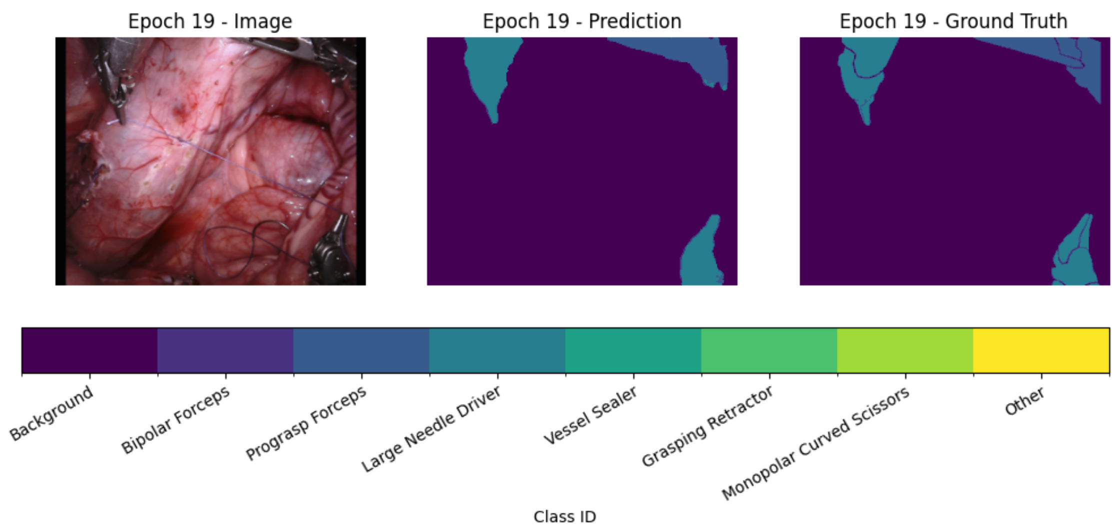




```
Only 3 epochs recorded, plotting all.
```







[50]:
```
if __name__ == "__main__":

    logger = TensorBoardLogger("tb_logs", name="part_instrument_seg")

    early_stop_callback = EarlyStopping(
        monitor="Train_Dice_CE_loss",           # metric name from self.log
        mode="min",                   # because lower loss is better
        patience=5,                   # epochs to wait before stopping
        verbose=True
    )
    checkpoint_callback = ModelCheckpoint(
        monitor="Train_Dice_CE_loss",
        mode="min",
        save_top_k=1,
        dirpath="checkpoints/",
        filename="best-part-instrument-seg-basic-unet",
    )

    trainer = Trainer(
        accelerator="gpu",
        max_epochs=20,
        #limit_train_batches=0.1,  # or 0.1 to use 10%
        logger=logger,
        callbacks=[early_stop_callback, checkpoint_callback],
    )

    start_train = time.time()
    trainer.fit(
        model=part_instr_seg_basic_UNet_model,
        datamodule=part_instr_seg_endo_data
    )
    end_train = time.time()
    print(f"Training time: {(end_train - start_train)/60:.2f} minutes")

    # Plot the overlaid training and val loss curves per epoch
    part_instr_seg_basic_UNet_model.plot_losses()

    # Plot the IOU and DSC curves per epoch
    part_instr_seg_basic_UNet_model.plot_metrics()

    # Plot images from last epoch
    part_instr_seg_basic_UNet_model.plot_result_by_epoch()
```

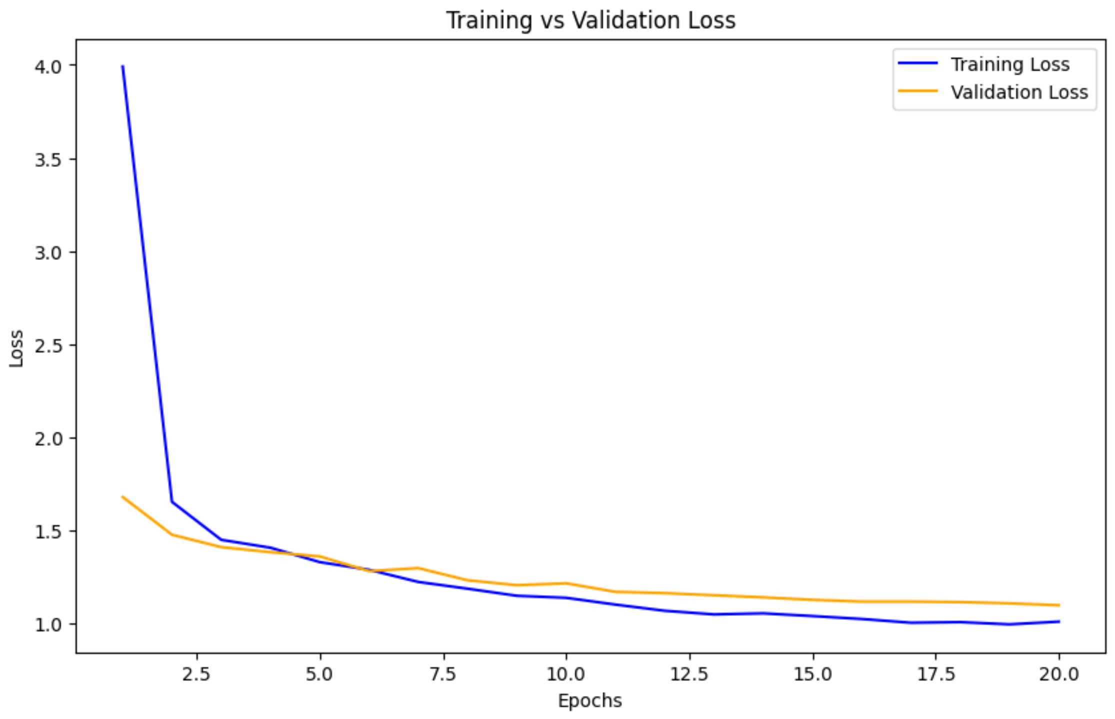
```
GPU available: True (cuda), used: True
TPU available: False, using: 0 TPU cores
HPU available: False, using: 0 HPUs
Checkpoint directory C:\Users\dsumm\OneDrive\Documents\UMD ENPM Robotics
Files\BIOE658B (Intro to Medical Image Analysis)\Project\code\checkpoints exists
and is not empty.
LOCAL_RANK: 0 - CUDA_VISIBLE_DEVICES: [0]

  | Name        | Type       | Params | Mode  | In sizes          | Out sizes
--------------------------------------------------------------------------------
-----------
0 | model       | BasicUNet  | 7.8 M  | train | [1, 3, 256, 320] | [1, 21, 256,
320]
1 | DICE_CE_Loss | DiceCELoss | 0      | train | ?                 | ?
--------------------------------------------------------------------------------
```

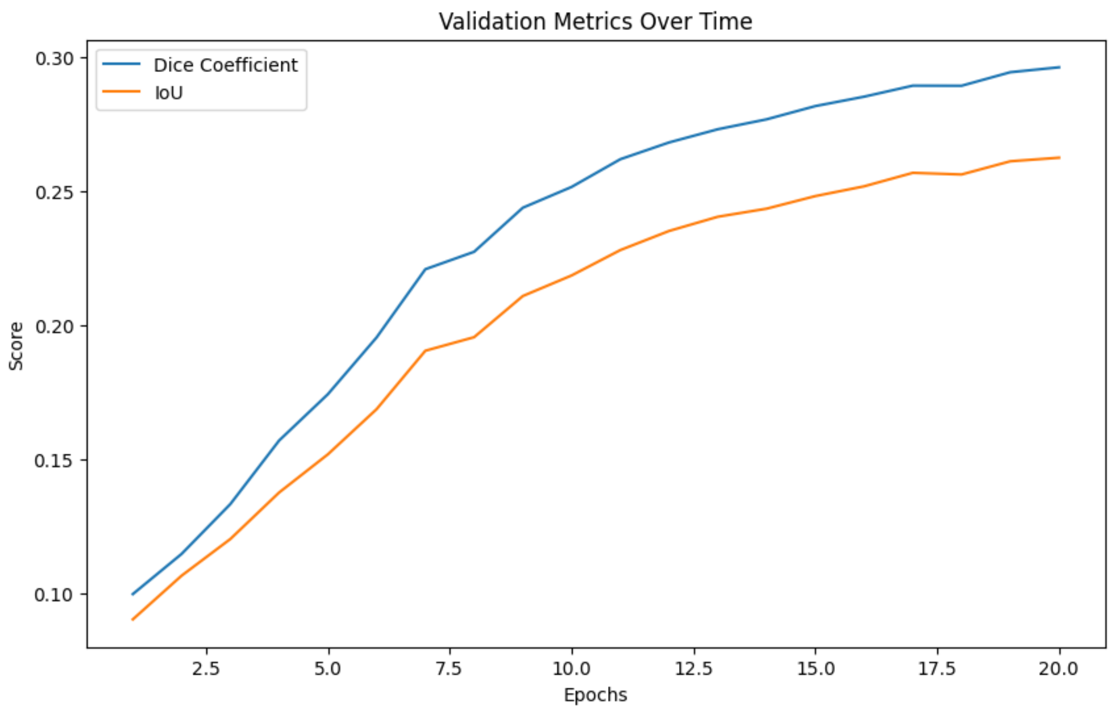
```
-----------
7.8 M     Trainable params
0         Non-trainable params
7.8 M     Total params
31.136    Total estimated model params size (MB)
143       Modules in train mode
0         Modules in eval mode

Sanity Checking: |          | 0/? [00:00<?, ?it/s]

The 'val_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.


The 'train_dataloader' does not have many workers which may be a bottleneck.
Consider increasing the value of the `num_workers` argument` to `num_workers=31`
in the `DataLoader` to improve performance.

Epoch 0: 100%|     | 144/144 [01:51<00:00,  1.29it/s, v_num=0,
Train_Dice_CE_loss_step=1.630]TE 20
Logged image from epoch 0
Epoch 0: 100%|     | 144/144 [02:15<00:00,  1.06it/s, v_num=0,
Train_Dice_CE_loss_step=1.630, val_loss=1.610, val_dice=0.0997, val_iou=0.0903,
Train_Dice_CE_loss_epoch=2.410]

Metric Train_Dice_CE_loss improved. New best score: 2.411

Epoch 1: 100%|     | 144/144 [01:41<00:00,  1.41it/s, v_num=0,
Train_Dice_CE_loss_step=1.420, val_loss=1.610, val_dice=0.0997, val_iou=0.0903,
Train_Dice_CE_loss_epoch=2.410]TE 20
Epoch 1: 100%|     | 144/144 [02:04<00:00,  1.16it/s, v_num=0,
Train_Dice_CE_loss_step=1.420, val_loss=1.400, val_dice=0.115, val_iou=0.107,
Train_Dice_CE_loss_epoch=1.470]

Metric Train_Dice_CE_loss improved by 0.943 >= min_delta = 0.0. New best score:
1.469

Epoch 2: 100%|     | 144/144 [01:41<00:00,  1.41it/s, v_num=0,
Train_Dice_CE_loss_step=1.380, val_loss=1.400, val_dice=0.115, val_iou=0.107,
Train_Dice_CE_loss_epoch=1.470]TE 20
Epoch 2: 100%|     | 144/144 [02:04<00:00,  1.15it/s, v_num=0,
Train_Dice_CE_loss_step=1.380, val_loss=1.330, val_dice=0.133, val_iou=0.120,
Train_Dice_CE_loss_epoch=1.360]

Metric Train_Dice_CE_loss improved by 0.106 >= min_delta = 0.0. New best score:
1.363

Epoch 3: 100%|     | 144/144 [01:42<00:00,  1.40it/s, v_num=0,
Train_Dice_CE_loss_step=1.290, val_loss=1.330, val_dice=0.133, val_iou=0.120,
Train_Dice_CE_loss_epoch=1.360]TE 20
Epoch 3: 100%|     | 144/144 [02:04<00:00,  1.16it/s, v_num=0,
Train_Dice_CE_loss_step=1.290, val_loss=1.290, val_dice=0.157, val_iou=0.138,
Train_Dice_CE_loss_epoch=1.310]

Metric Train_Dice_CE_loss improved by 0.053 >= min_delta = 0.0. New best score:
1.310

Epoch 4: 100%|      | 144/144 [01:41<00:00,  1.42it/s, v_num=0,
Train_Dice_CE_loss_step=1.260, val_loss=1.290, val_dice=0.157, val_iou=0.138,
Train_Dice_CE_loss_epoch=1.310]TE 20
Epoch 4: 100%|      | 144/144 [02:03<00:00,  1.17it/s, v_num=0,
Train_Dice_CE_loss_step=1.260, val_loss=1.260, val_dice=0.174, val_iou=0.152,
Train_Dice_CE_loss_epoch=1.270]

Metric Train_Dice_CE_loss improved by 0.037 >= min_delta = 0.0. New best score:
1.273

Epoch 5: 100%|      | 144/144 [01:55<00:00,  1.25it/s, v_num=0,
Train_Dice_CE_loss_step=1.220, val_loss=1.260, val_dice=0.174, val_iou=0.152,
Train_Dice_CE_loss_epoch=1.270]TE 20
Epoch 5: 100%|      | 144/144 [02:20<00:00,  1.02it/s, v_num=0,
Train_Dice_CE_loss_step=1.220, val_loss=1.190, val_dice=0.196, val_iou=0.169,
Train_Dice_CE_loss_epoch=1.220]

Metric Train_Dice_CE_loss improved by 0.058 >= min_delta = 0.0. New best score:
1.215

Epoch 6: 100%|      | 144/144 [01:45<00:00,  1.37it/s, v_num=0,
Train_Dice_CE_loss_step=1.280, val_loss=1.190, val_dice=0.196, val_iou=0.169,
Train_Dice_CE_loss_epoch=1.220]TE 20
Epoch 6: 100%|      | 144/144 [02:07<00:00,  1.13it/s, v_num=0,
Train_Dice_CE_loss_step=1.280, val_loss=1.180, val_dice=0.221, val_iou=0.191,
Train_Dice_CE_loss_epoch=1.190]

Metric Train_Dice_CE_loss improved by 0.029 >= min_delta = 0.0. New best score:
1.186

Epoch 7: 100%|      | 144/144 [01:40<00:00,  1.43it/s, v_num=0,
Train_Dice_CE_loss_step=1.160, val_loss=1.180, val_dice=0.221, val_iou=0.191,
Train_Dice_CE_loss_epoch=1.190]TE 20
Epoch 7: 100%|      | 144/144 [02:02<00:00,  1.17it/s, v_num=0,
Train_Dice_CE_loss_step=1.160, val_loss=1.140, val_dice=0.228, val_iou=0.196,
Train_Dice_CE_loss_epoch=1.160]

Metric Train_Dice_CE_loss improved by 0.027 >= min_delta = 0.0. New best score:
1.159

Epoch 8: 100%|      | 144/144 [01:40<00:00,  1.44it/s, v_num=0,
Train_Dice_CE_loss_step=1.160, val_loss=1.140, val_dice=0.228, val_iou=0.196,
Train_Dice_CE_loss_epoch=1.160]TE 20
Epoch 8: 100%|      | 144/144 [02:01<00:00,  1.19it/s, v_num=0,
Train_Dice_CE_loss_step=1.160, val_loss=1.120, val_dice=0.244, val_iou=0.211,
Train_Dice_CE_loss_epoch=1.140]
```

```
Metric Train_Dice_CE_loss improved by 0.024 >= min_delta = 0.0. New best score:
1.136

Epoch 9: 100%|     | 144/144 [01:40<00:00,  1.44it/s, v_num=0,
Train_Dice_CE_loss_step=1.100, val_loss=1.120, val_dice=0.244, val_iou=0.211,
Train_Dice_CE_loss_epoch=1.140]TE 20
Epoch 9: 100%|     | 144/144 [02:02<00:00,  1.18it/s, v_num=0,
Train_Dice_CE_loss_step=1.100, val_loss=1.100, val_dice=0.252, val_iou=0.219,
Train_Dice_CE_loss_epoch=1.110]

Metric Train_Dice_CE_loss improved by 0.027 >= min_delta = 0.0. New best score:
1.108

Epoch 10: 100%|     | 144/144 [01:41<00:00,  1.42it/s, v_num=0,
Train_Dice_CE_loss_step=1.090, val_loss=1.100, val_dice=0.252, val_iou=0.219,
Train_Dice_CE_loss_epoch=1.110]TE 20
Logged image from epoch 10
Epoch 10: 100%|     | 144/144 [02:03<00:00,  1.17it/s, v_num=0,
Train_Dice_CE_loss_step=1.090, val_loss=1.080, val_dice=0.262, val_iou=0.228,
Train_Dice_CE_loss_epoch=1.080]

Metric Train_Dice_CE_loss improved by 0.031 >= min_delta = 0.0. New best score:
1.077

Epoch 11: 100%|     | 144/144 [01:40<00:00,  1.43it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.080, val_dice=0.262, val_iou=0.228,
Train_Dice_CE_loss_epoch=1.080]TE 20
Epoch 11: 100%|     | 144/144 [02:01<00:00,  1.18it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.060, val_dice=0.268, val_iou=0.235,
Train_Dice_CE_loss_epoch=1.070]

Metric Train_Dice_CE_loss improved by 0.010 >= min_delta = 0.0. New best score:
1.067

Epoch 12: 100%|     | 144/144 [01:40<00:00,  1.43it/s, v_num=0,
Train_Dice_CE_loss_step=1.060, val_loss=1.060, val_dice=0.268, val_iou=0.235,
Train_Dice_CE_loss_epoch=1.070]TE 20
Epoch 12: 100%|     | 144/144 [02:01<00:00,  1.19it/s, v_num=0,
Train_Dice_CE_loss_step=1.060, val_loss=1.050, val_dice=0.273, val_iou=0.241,
Train_Dice_CE_loss_epoch=1.050]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
1.054

Epoch 13: 100%|     | 144/144 [01:32<00:00,  1.55it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.050, val_dice=0.273, val_iou=0.241,
Train_Dice_CE_loss_epoch=1.050]TE 20
Epoch 13: 100%|     | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.050, val_dice=0.277, val_iou=0.244,
Train_Dice_CE_loss_epoch=1.050]

Metric Train_Dice_CE_loss improved by 0.009 >= min_delta = 0.0. New best score:
1.045
```

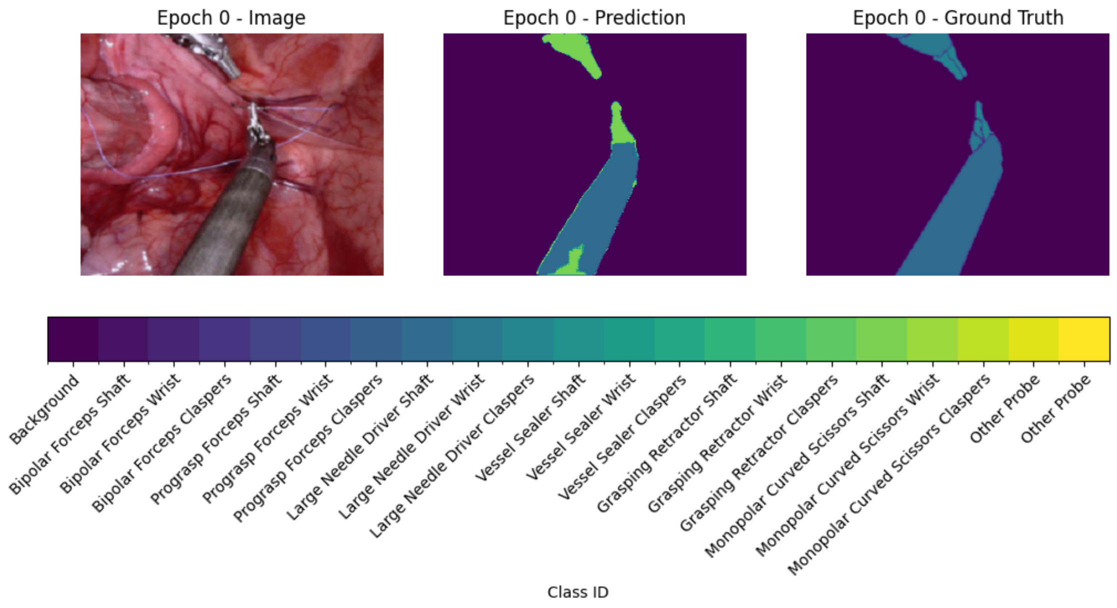
```
Epoch 14: 100%|      | 144/144 [01:32<00:00,  1.56it/s, v_num=0,
Train_Dice_CE_loss_step=1.060, val_loss=1.050, val_dice=0.277, val_iou=0.244,
Train_Dice_CE_loss_epoch=1.050]TE 20
Epoch 14: 100%|      | 144/144 [01:52<00:00,  1.28it/s, v_num=0,
Train_Dice_CE_loss_step=1.060, val_loss=1.040, val_dice=0.282, val_iou=0.248,
Train_Dice_CE_loss_epoch=1.040]

Metric Train_Dice_CE_loss improved by 0.009 >= min_delta = 0.0. New best score:
1.037

Epoch 15: 100%|      | 144/144 [01:32<00:00,  1.55it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.040, val_dice=0.282, val_iou=0.248,
Train_Dice_CE_loss_epoch=1.040]TE 20
Epoch 15: 100%|      | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.070, val_loss=1.030, val_dice=0.285, val_iou=0.252,
Train_Dice_CE_loss_epoch=1.020]

Metric Train_Dice_CE_loss improved by 0.013 >= min_delta = 0.0. New best score:
1.024

Epoch 16: 100%|      | 144/144 [01:32<00:00,  1.55it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.030, val_dice=0.285, val_iou=0.252,
Train_Dice_CE_loss_epoch=1.020]TE 20
Epoch 16: 100%|      | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.030, val_dice=0.290, val_iou=0.257,
Train_Dice_CE_loss_epoch=1.020]

Metric Train_Dice_CE_loss improved by 0.005 >= min_delta = 0.0. New best score:
1.019

Epoch 17: 100%|      | 144/144 [01:32<00:00,  1.56it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.030, val_dice=0.290, val_iou=0.257,
Train_Dice_CE_loss_epoch=1.020]TE 20
Epoch 17: 100%|      | 144/144 [01:52<00:00,  1.28it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.010, val_dice=0.290, val_iou=0.256,
Train_Dice_CE_loss_epoch=1.010]

Metric Train_Dice_CE_loss improved by 0.007 >= min_delta = 0.0. New best score:
1.012

Epoch 18: 100%|      | 144/144 [01:32<00:00,  1.55it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.010, val_dice=0.290, val_iou=0.256,
Train_Dice_CE_loss_epoch=1.010]TE 20
Epoch 18: 100%|      | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.020, val_dice=0.295, val_iou=0.261,
Train_Dice_CE_loss_epoch=1.010]

Metric Train_Dice_CE_loss improved by 0.003 >= min_delta = 0.0. New best score:
1.009

Epoch 19: 100%|      | 144/144 [01:32<00:00,  1.55it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.020, val_dice=0.295, val_iou=0.261,
Train_Dice_CE_loss_epoch=1.010]TE 20
```

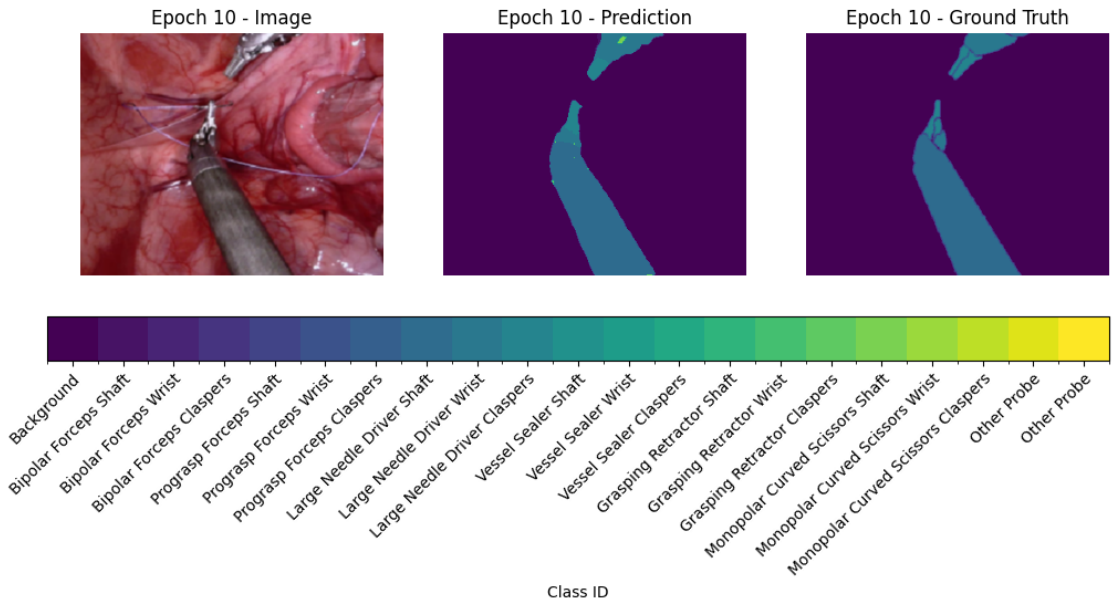
```
Logged image from epoch 19
Epoch 19: 100%|      | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.020, val_dice=0.296, val_iou=0.263,
Train_Dice_CE_loss_epoch=1.010]

Metric Train_Dice_CE_loss improved by 0.002 >= min_delta = 0.0. New best score:
1.007
`Trainer.fit` stopped: `max_epochs=20` reached.

Epoch 19: 100%|      | 144/144 [01:53<00:00,  1.27it/s, v_num=0,
Train_Dice_CE_loss_step=1.030, val_loss=1.020, val_dice=0.296, val_iou=0.263,
Train_Dice_CE_loss_epoch=1.010]
Training time: 40.55 minutes
```

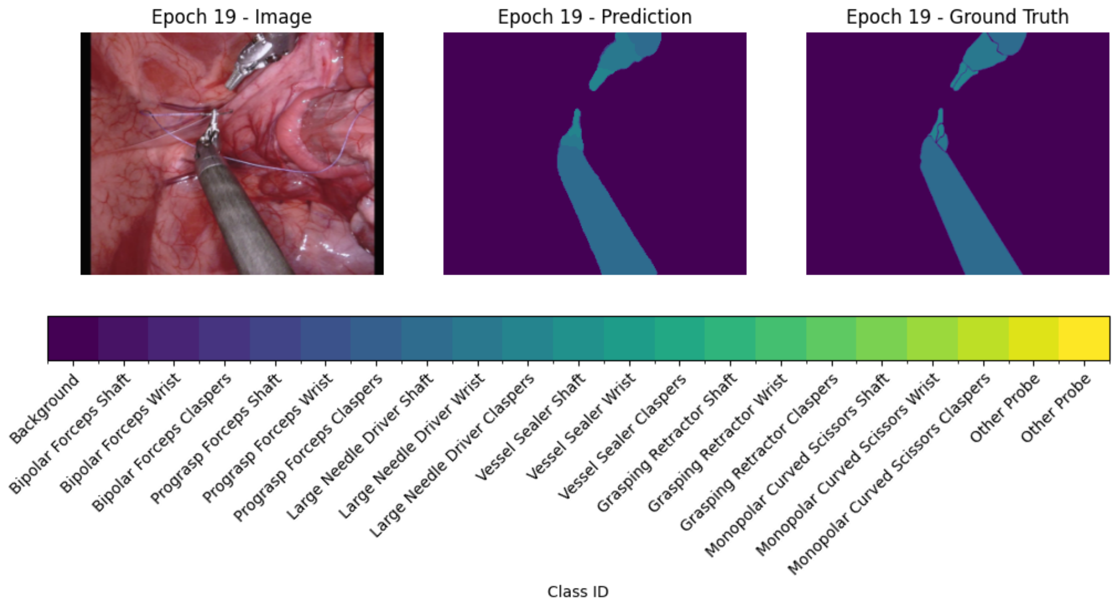




```
Only 3 epochs recorded, plotting all.
```







[52]:
```
os.makedirs('basic_UNet_final_models', exist_ok=True)

# Define file names with paths
binary_basic_UNet_model_filename = 'basicUNetmodels/binary_basic_UNet_model.pth'
part_seg_basic_UNet_model_filename = 'basicUNetmodels/part_seg_basic_UNet_model.
 ↪pth'
```

```
instr_seg_basic_UNet_model_filename = 'basicUNetmodels/
 ↪instr_seg_basic_UNet_model.pth'
part_instr_seg_basic_UNet_model_filename = 'basicUNetmodels/
 ↪part_instr_seg_basic_UNet_model.pth'

# Save the model parameters
torch.save(binary_basic_UNet_model.state_dict(),␣
 ↪binary_basic_UNet_model_filename)
torch.save(part_seg_basic_UNet_model.state_dict(),␣
 ↪part_seg_basic_UNet_model_filename)
torch.save(instr_seg_basic_UNet_model.state_dict(),␣
 ↪instr_seg_basic_UNet_model_filename)
torch.save(part_instr_seg_basic_UNet_model.state_dict(),␣
 ↪part_instr_seg_basic_UNet_model_filename)

print("Models saved in the 'basic_UNet_final_models' directory!")
```

```
Models saved in the 'basic_UNet_final_models' directory!
```