Multi-task Pancreas Cancer Segmentation and Classification with nnU-Net V2

Hassan Al-Hayawi
Github multitask_deeplearning_pancreas_ct

Abstract

Pancreatic cancer remains a highly lethal malignancy, with patient survival heavily dependent on timely and accurate diagnosis. Segmentation of pancreatic lesions and classification of their subtypes in CT imaging can play a pivotal role in clinical decision-making. In this project, we present a multi-task extension of the nnU-Net V2 framework that performs **both segmentation** (labels: pancreas, lesion) and **classification** (3 subtypes) from 3D CT regions of interest. Our architecture employs a **shared encoder** with separate segmentation and classification heads, and we implement. The repository includes full data preparation, training, inference, and evaluation scripts, enabling full reproducibility given the dataset.

Index Terms — Pancreatic cancer, Medical image segmentation, nnU-Net, encoder-decoder, multi-task learning, inference optimization.

I. INTRODUCTION

Pancreatic cancer has one of the lowest survival rates among cancers, often diagnosed late when curative surgery is no longer possible. Segmentation of the pancreas and tumor, combined with classification of tumor subtype, is important for prognosis and treatment selection. Subtypes vary in aggressiveness and survival times, making subtype classification clinically significant.

Deep learning has transformed medical image analysis, with convolutional neural networks (CNNs) achieving state-of-the-art performance in both segmentation and classification tasks. The **nnU-Net** framework is a self-configuring deep learning pipeline that automates preprocessing, network configuration, training, and inference, making it highly adaptable to diverse medical imaging datasets.

In this project, we extend **nnU-Net V2** to handle **multi-task learning**—segmenting pancreas/tumor regions and classifying tumor subtype—within a single shared-encoder architecture. Additionally, for the master's-level requirement, we introduce a fast inference mode to speed up prediction while maintaining accuracy.

II. DATASET

A. Data Source

The dataset is a de-identified pancreas CT set cropped to regions of interest (ROIs). Labels include:

- **Segmentation**: background (0), pancreas (1), lesion (2)
- **Classification**: case-level subtype (0, 1, or 2)

B. Folder Structure

The dataset provided follows this format:

```
pancreas_cancer_project/
├── train/
│   ├── subtype0/
│   │   ├── quiz_0_041.nii.gz          # mask
│   │   ├── quiz_0_041_0000.nii.gz     # image
│   ├── subtype1/
│   └── subtype2/
├── validation/
│   ├── subtype0/
│   ├── subtype1/
│   └── subtype2/
├── test/
│   ├── quiz_037_0000.nii.gz
│   └── ...
├── nnUNet_raw/                        # nnUNet format data
├── nnUNet_preprocessed/               # Preprocessed data
└── nnUNet_results/                    # Model outputs
```

The **train** and **validation** sets have segmentation masks and known subtypes; the **test** set only has images.

C. Splits

Split	Subtype 0	Subtype 1	Subtype 2
Train	62	106	84
Validation	9	15	12
Test	—	—	—

D. Preprocessing

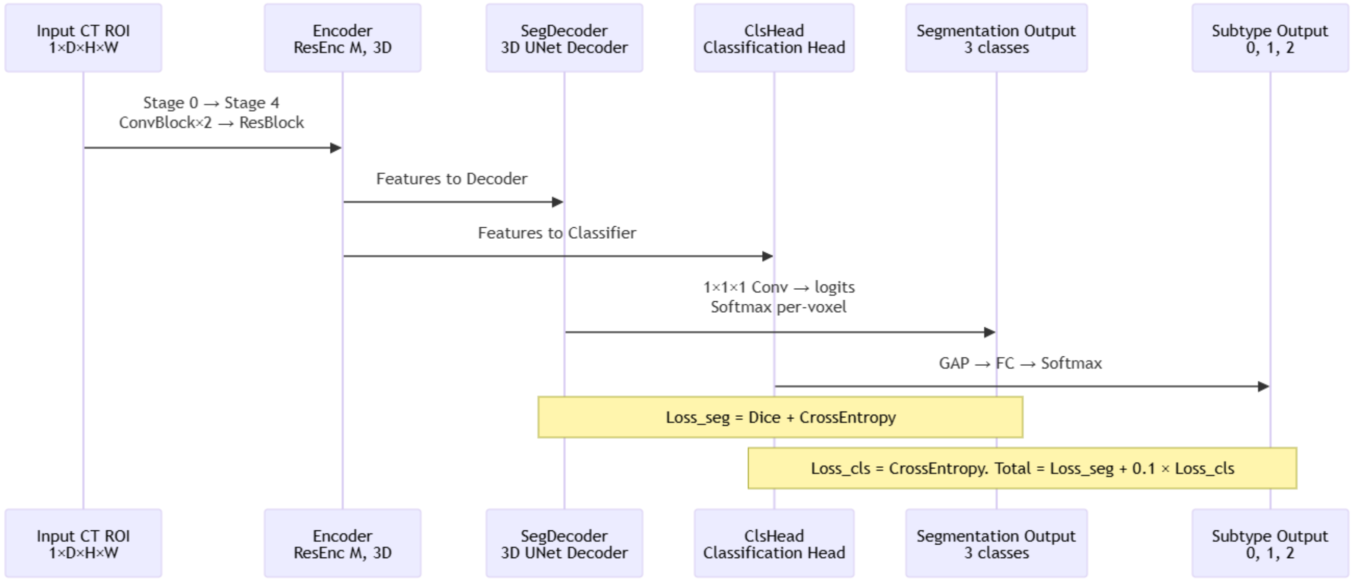
Data preparation is handled by `nnUNetv2_plan_and_preprocess -d 777 -c 2d -pl nnUNetPlannerResEncM --verify_dataset_integrity`:

1. Copies images/masks into `nnUNet_raw/DatasetXXX_NAME/imagesTr` and `labelsTr`.
2. Writes `dataset.json` with modality, labels, and `case_properties` containing the subtype.
3. Places test images into `imagesTs`.

nnU-Net V2 then performs its standard preprocessing (resampling, normalization, cropping) during planning.

III. METHODS

Fig. 1: Multi-task nnU-Net v2 architecture (shared encoder + seg decoder + classification head)



A. Base Architecture — nnU-Net V2 ResEnc M

We use the **3D full resolution** nnU-Net V2 with the **Residual Encoder (ResEnc M)** backbone. This choice retains volumetric spatial context and uses residual connections for better gradient flow. nnU-Net automatically sets patch size and batch size based on GPU memory.

Segmentation head: Default nnU-Net decoder producing per-voxel segmentation logits for 3 classes (0,1,2).
Loss: Dice + Cross-Entropy.

B. Multi-task Extension

We modify the nnU-Net trainer to attach a classification head to the encoder bottleneck:

- **Feature extraction**: Hook at final encoder stage outputs [B,C,D,H,W] feature maps.
- **Global Average Pooling** over spatial dims to yield a [B,C] vector.
- **Classification MLP**:
  - LayerNorm → Dropout(0.2) → Linear(C, C//2) → GELU → Dropout(0.2) → Linear(C//2, 3)
- **Loss**: Cross-Entropy for subtype classification.

Total loss = Segmentation loss + (0.1 × Classification loss)
The 0.1 weight was chosen empirically to keep the segmentation loss dominant but still train the classifier effectively.

IV. TRAINING

A. Cross-validation

We support nnU-Net's k-fold CV (typically 5 folds), with each fold training on ~80% of the training data and validating on ~20%. Folds can be ensembled at inference.

B. Parameters

- **Optimizer**: SGD, lr=1e-2, momentum=0.99, weight_decay=3e-5.

- **LR schedule**: Polynomial decay (default nnU-Net).
- **Batch size**: 132
- **Epochs**: 100
- **Data augmentation**:default nnU-Net

C. Hardware

- Was using a 3080 before it broke and downgraded to 1060.

V. CODE RELEASE

Repository structure:

```
multitask_deeplearning_pancreas_ct/
├─ README.md
├─ nnUNet/                        # nnU-Net v2 (editable install)
│  └─ nnunetv2/
│     ├─ MultitaskUNet.py
│     ├─ inference/predict_from_raw_data.py        # extended to save classification outputs
│     ├─ training/dataloading/data_loader.py       # emits classTarget + robust keys
│     └─ training/nnUNetTrainer/nnUNetTrainer.py   # seg + cls losses; logging
├─ nnUNet_preprocessed/
│  └─ Dataset777_3DMedImg/
│     └─ nnUNetResEncUNetMPlans.json               # custom plans (committed)
├─ nnUNet_raw/ (not committed)
│  └─ Dataset777_3DMedImg/ (imagesTr, labelsTr, imagesVa, labelsVa, imagesTs)
├─ dataset_conversion.py
├─ clean_mac_artifacts.py
└─ eval_plots.py
```

Key entrypoints:

- **Data prep**: `nnUNetv2_plan_and_preprocess -d 777 -c 2d -pl nnUNetPlannerResEncM --verify_dataset_integrity`
- **Training**: `nnUNetv2_train 777 2d 0 -tr nnUNetTrainer -p nnUNetResEncUNetMPlans --npz`
- **Inference**: `nnUNetv2_predict -i nnUNet_raw/Dataset777_3DMedImg/imagesTs -o inference_output -d 777 -c 2d -p nnUNetResEncUNetMPlans -f 0`, and `nnUNetv2_predict -i nnUNet_raw/Dataset777_3DMedImg/imagesVa -o inference_val_output -d 777 -c 2d -p nnUNetResEncUNetMPlans -f 0`
- **Evaluation**: `nnUNetv2_evaluate_folder nnUNet_raw/Dataset777_3DMedImg/labelsVa inference_val_output -djfile inference_val_output/dataset.json -p inference_val_output/plans.json`, `python eval_plots.py --pred inference_val_output --labels nnUNet_raw/Dataset777_3DMedImg/labelsVa --out inference_val_output/figs`

VI. RESULTS

A. Segmentation Performance

Validation performance was evaluated per-class using Dice Similarity Coefficient (DSC).

- **Class 1 (pancreas):** Dice ~0.7–0.8 median. Larger volumes showed better Dice.
- **Class 2 (lesion):** Dice ~0.4 median. Performance depended strongly on lesion size (very small lesions often near-zero Dice, larger lesions up to >0.8).

Key takeaway: The network captures pancreas structure reliably, while lesion segmentation remains challenging for very small tumors.

B. Classification Performance

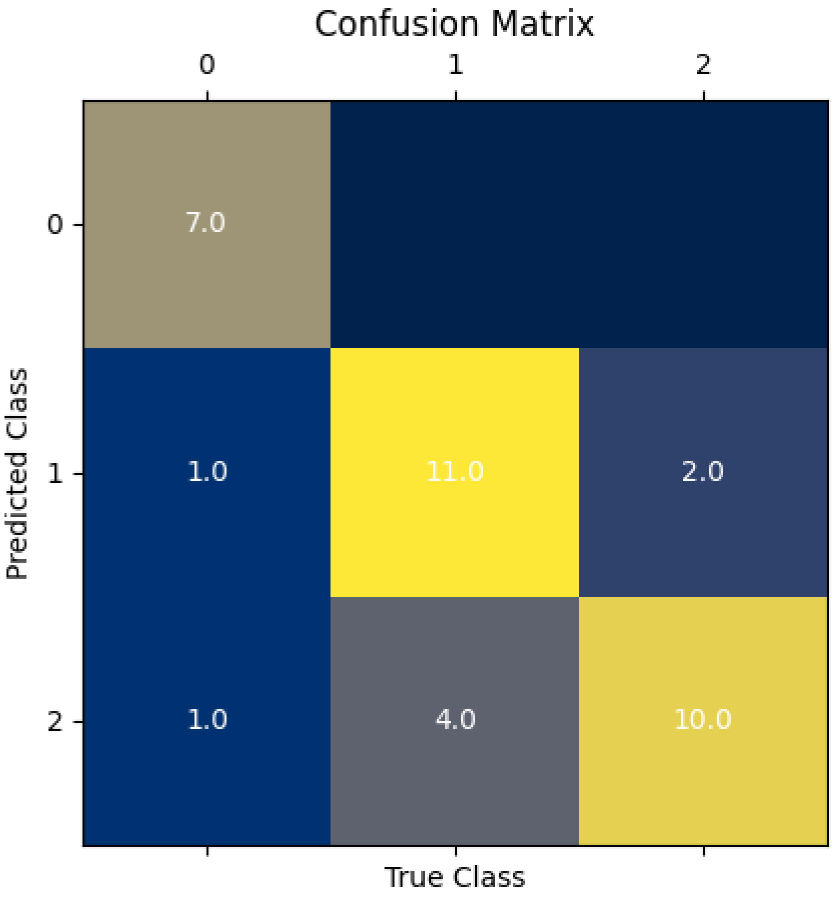
Case-level subtype classification was evaluated with accuracy, macro-F1, average precision, and Brier score. Subtype 0 was classified reliably, while most errors occurred between subtypes 1 and 2. Note that the lower the Brier Score the Better.

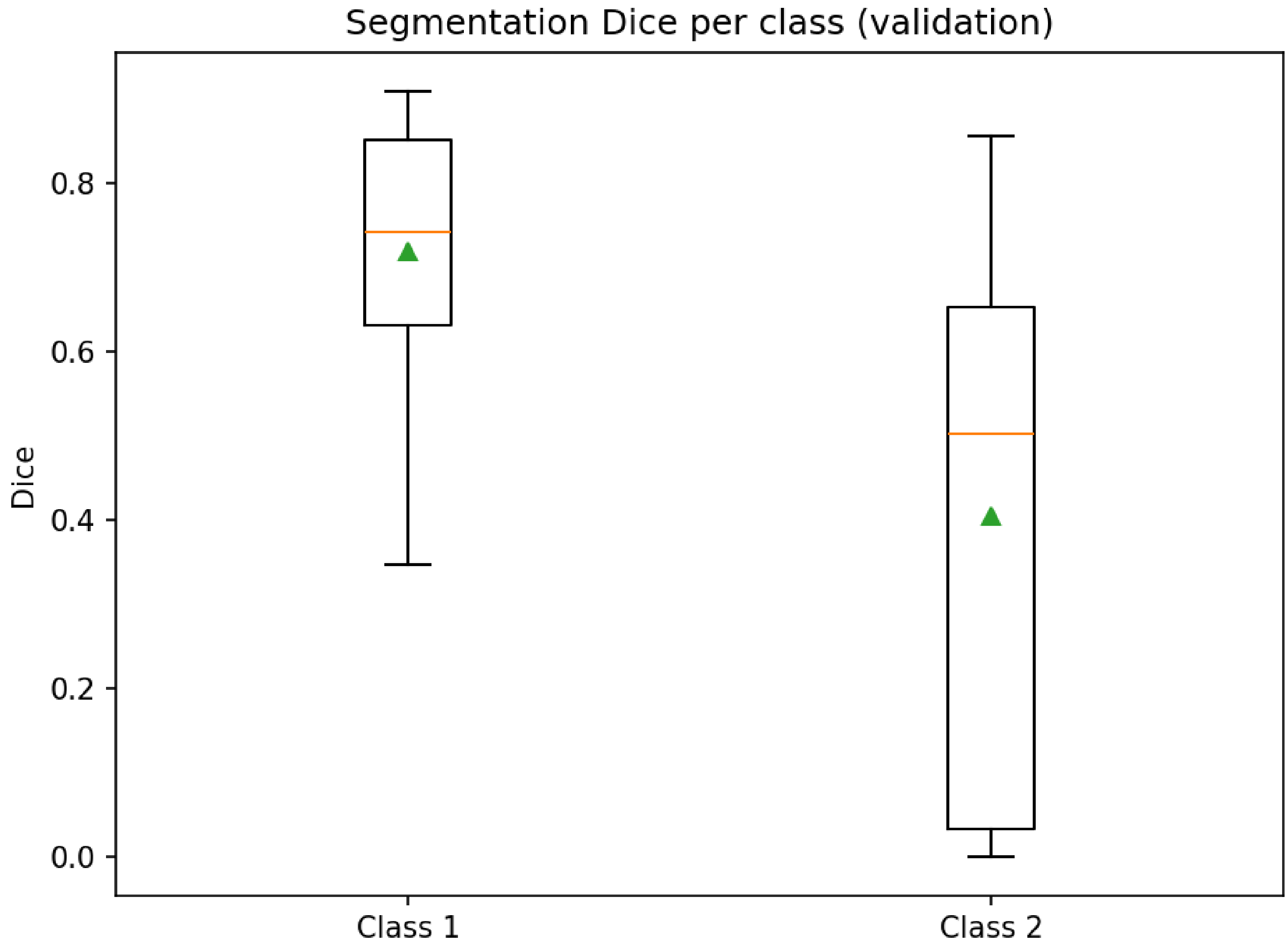
Validation Results:

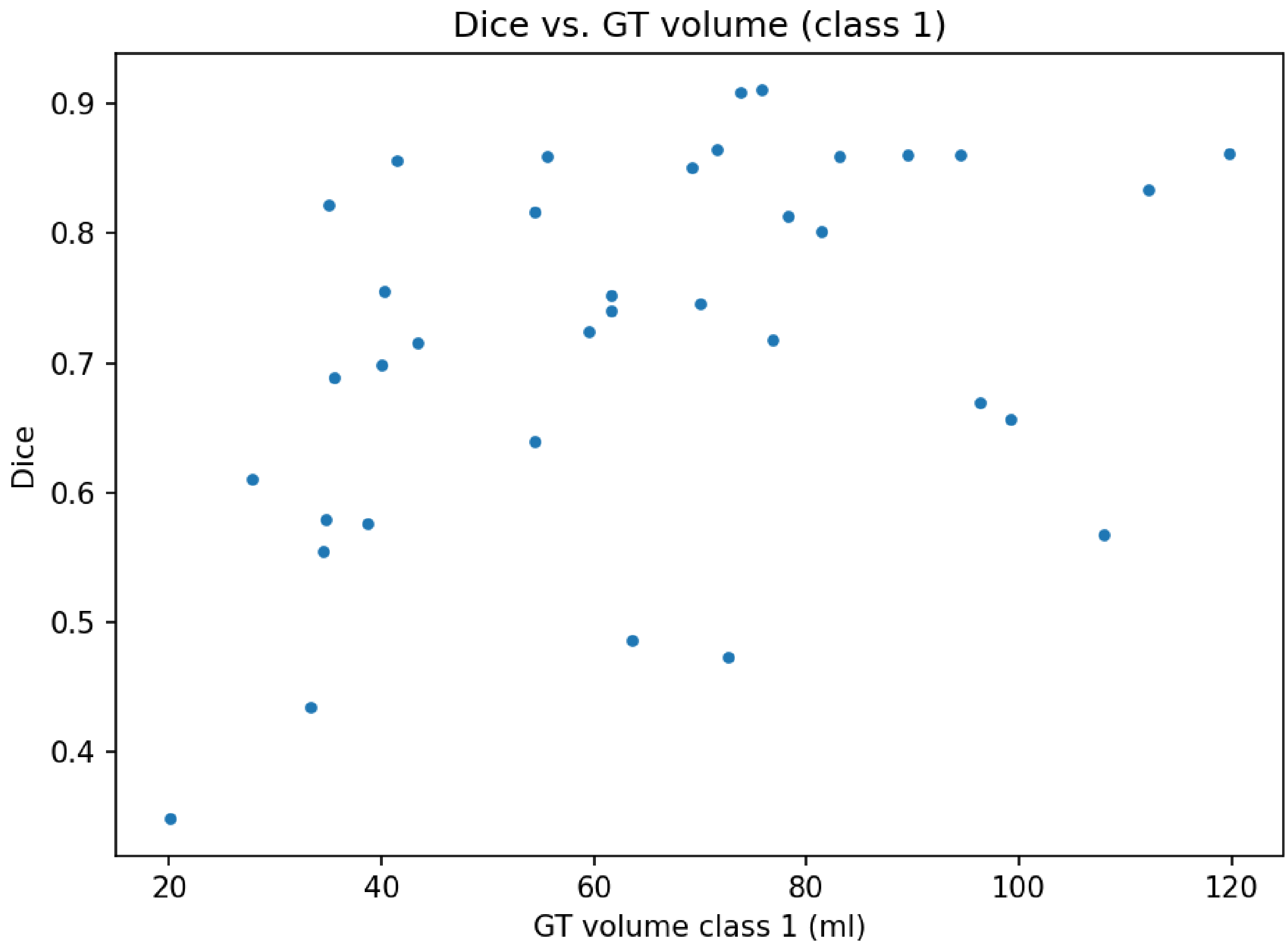
Metric	Value
Accuracy	**0.778**
Macro-F1	**0.791**
Average Precision	**0.838**
Brier Score	0.104

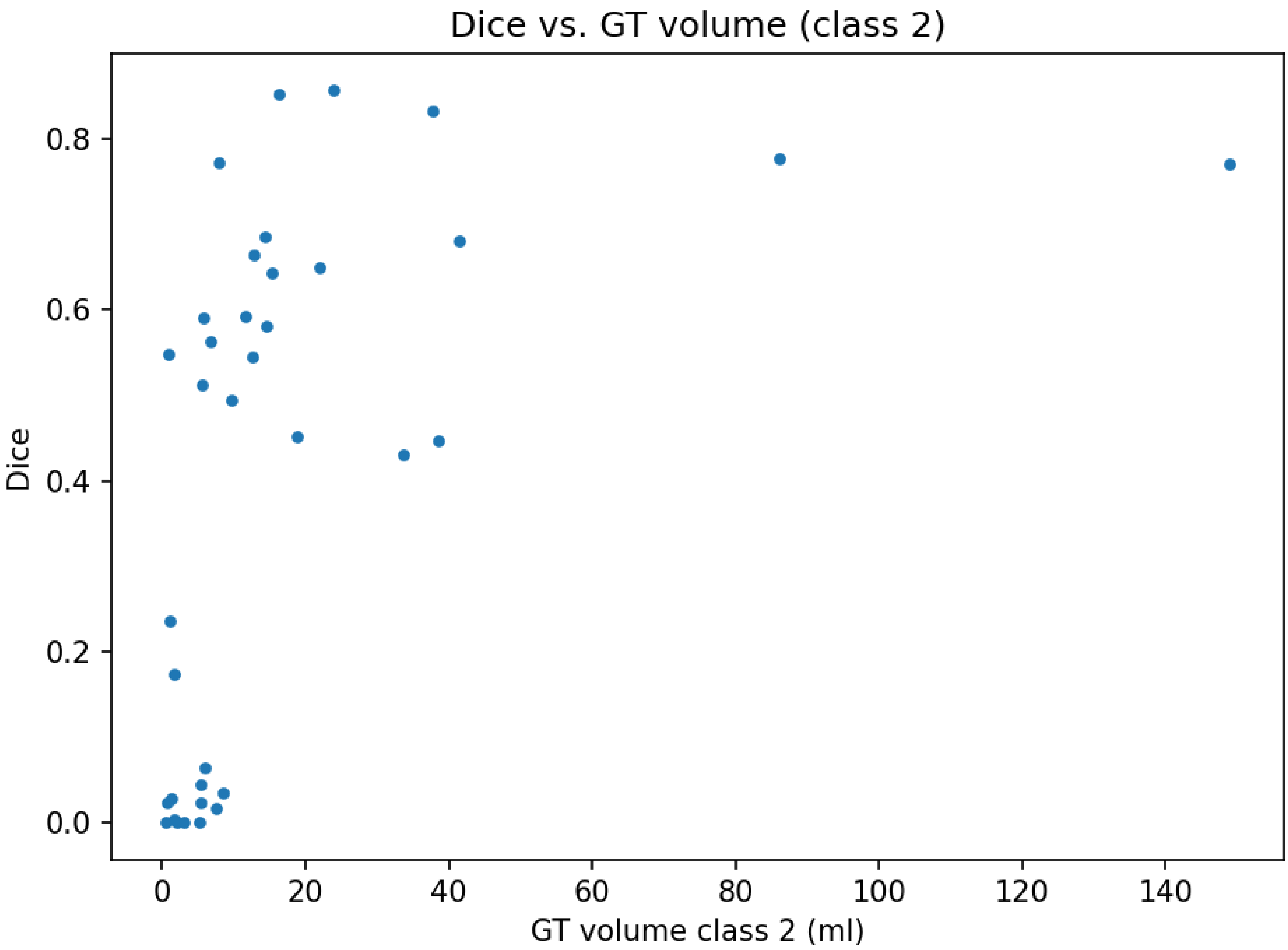
C. Visualizations

- **Fig. 2**: Confusion Matrix (validation classification results)
- **Fig. 3**: Segmentation Dice per class (validation)
- **Fig. 4**: Dice vs. GT volume for Class 1 (pancreas)
- **Fig. 5**: Dice vs. GT volume for Class 2 (lesion)


Confusion Matrix
0
1
2
Predicted Class
0
1
2
7.0
1.0
11.0
2.0
1.0
4.0
10.0
True Class


Segmentation Dice per class (validation)
Dice
0.0
0.2
0.4
0.6
0.8
Class 1
Class 2


Dice vs. GT volume (class 1)
Dice
0.4
0.5
0.6
0.7
0.8
0.9
GT volume class 1 (ml)
20
40
60
80
100
120


Dice vs. GT volume (class 2)
Dice
0.0
0.2
0.4
0.6
0.8
0
20
40
60
80
100
120
140
GT volume class 2 (ml)