Report: Deep Learning for Automatic Pancreas Cancer Segmentation and Classification in CT Scans

Prepared by:
Muhua (Brianna) Zhao

Prepared for:
M31

Submission Date:
August 15th, 2025

Table of Content

1. Introduction

This assessment explores the multi-task extension of nnUNetv2 for pancreas CT images, enabling simultaneous pancrease/lesion segmentation and three-way subtype classification. We utilize a de-identified pancreatic CT image dataset for joint learning: each 3D case includes voxel-level labels and case-level subtypes. The dataset is organized in the nnUNetv2 [1], [2] format and undergoes standardized preprocessing. Development and experiments were conducted in a Google Colab environment. The model is based on the residual encoder U-Net ("ResEncUNet-M"), sharing a common encoder across two specific tasks: a decoder for segmentation and a classification head based on encoder features, calibrated using empirical priors. Inference follows the nnUNetv2 framework, and performance for pancreatic/lesion classification is reported using Dice, while subtype classification performance is reported using accuracy and macro F1.

2. Methods

2.1. Dataset

In this assessment, a de-identified pancreas CT dataset specifically designed for multi-task learning involving segmentation and classification of pancreatic cancer was utilized. Each 3D volume was pre-cropped to regions of interest (ROIs) to reduce computational load and enable training on Google Colab. Ground truth annotations include three-class semantic labels: background (0), pancreas (1), and pancreatic lesion (2). Additionally, each case was labeled with one of three lesion subtypes for classification tasks (Subtype 0, 1, 2). The dataset was split into training, validation, and testing sets. The training set comprised 252 cases, while the validation set included 36 cases, distributed evenly across the three subtypes. Only images were provided for the test set.

2.2. Data Preprocessing and Augmentation

The dataset was converted to the standardized nnU-Net v2 format following the framework's conventions. Images were renamed using the pattern case_XXXX_0000.nii.gz for training images and case_XXXX.nii.gz for corresponding segmentation masks. A comprehensive dataset configuration file (dataset.json) was generated specifying the 3D tensor format, channel information, and label mappings. The nnU-Net v2 preprocessing pipeline was employed, which includes [1]:

1) Intensity normalization: CT-specific normalization using the framework's CTNormalization scheme.
2) Resampling: Automatic determination of target spacing and patch sizes based on dataset characteristics.
3) Configuration planning: The nnUNetPlannerResEncM planner was used to generate optimized network configurations

The preprocessing resulted in two configurations: 2D configuration (Patch size [128, 192], batch size 134) and 3D fullres configuration (Patch size [64, 128, 192], batch size 2).

2.3. Network Architecture

The baseline model was derived from the nnUNetTrainer with the ResEncUNet-M configuration, which provides a medium-capacity 3D U-Net architecture with a residual encoder. To support multi-task learning, a modified trainer (MultiTaskTrainer) that shares the encoder between two task-specific heads was implemented:

1) Segmentation Head: Retained from the original nnU-Net design, this head outputs a three-channel segmentation map (background, pancreas, lesion) and is trained using a soft Dice + Cross Entropy loss function.
2) Classification Head: We introduced a parallel head composed of a global average pooling (GAP) layer followed by a fully connected layer with three output neurons for subtype classification. The classification head operates on the encoded feature volume prior to upsampling and outputs a softmax probability over the three subtypes. Cross-entropy loss was used for training this head.

2.4. Multi-task Loss Formulation

The overall training objective combines segmentation and classification losses as follows:

$$L_{total} = L_{seg} + \lambda(t) \cdot L_{cls}$$

Where L_{seg}: soft Dice + Cross Entropy loss on voxel-level masks; L_{cls}: class-balanced Cross Entropy with label smoothing (ε = 0.05) and logit adjustment based on empirical subtype priors; $\lambda(t)$: a dynamic balancing weight linearly ramped from 0.10 to 0.55 over the first 25 epochs.

2.5. Classification Calibration and Label Mapping

Subtype labels were loaded from a custom CSV file mapping each case ID to a subtype class. During training, class weights were computed from inverse frequency counts and applied in the Cross Entropy loss. To further calibrate predictions, subtype priors were transformed into log space and used to adjust logits before applying the softmax layer. This logit adjustment improves generalization and reduces overconfidence in dominant classes.

2.6. Training Procedure

Training was conducted on Colab NVIDIA T4 GPU using nnUNetv2's modular trainer pipeline. Key hyperparameters included:

1) Optimizer: SGD with Nesterov momentum
2) Learning rate: Polynomial decay starting at 0.01

3) Batch size: Dynamically selected based on GPU memory
4) Epochs: 200 (with early stopping once validation loss plateaued)
5) AMP: Enabled after 6 warm-up iterations to avoid NaN gradients
6) Gradient clipping: Applied with a max norm of 12

Data augmentations followed the nnUNetv2 defaults, including random affine transforms, Gaussian noise, and intensity scaling. A custom training and validation routine (train_step and validation_step) was also implemented to support multi-task supervision. During each step:

1) Segmentation predictions are compared with targets using soft Dice + CE loss.
2) Subtype predictions are adjusted using log priors and masked to exclude missing labels.
3) A dynamic weighting schedule balances classification and segmentation objectives.
4) If a non-finite loss is detected (NaN/Inf), the forward and backward passes are re-executed in full-precision (FP32) as a fallback.
5) Classification accuracy is logged per epoch, and the best-performing classification head is checkpointed separately for reproducibility.

2.7. Inference and Evaluation

To perform inference on the test and validation datasets, we used the trained MultiTask nnUNetv2 model implemented in PyTorch. The input images were first resampled to the target spacing defined in the planning stage and normalized using z-score normalization. For each case, a sliding window inference strategy was applied with overlap-based patch stitching, and predictions were generated for both the segmentation masks and classification subtype logits. The segmentation output was obtained by applying an argmax over the softmax probabilities, while subtype classification was derived from the logits using logit adjustment calibrated with the empirical class prior. The best-performing classification head, saved separately during training, was reloaded prior to inference to ensure optimal classification performance.

Segmentation performance was evaluated using the mean Dice similarity coefficient computed per class (pancreas and lesion) across the validation set [3]. Classification performance was assessed using accuracy and macro-averaged F1-score, along with a confusion matrix to capture inter-class misclassifications [3]. In cases where class imbalance was observed in predictions, logit adjustment and τ-calibration were applied to the classification head during inference to improve robustness. All metrics were computed on the native dataset structure, with per-case and summary results saved for further analysis.

3. Results

Evaluation was performed on the trained multi-task nnUNetv2 model on the held-out validation set (36 cases. For each case, images were resampled to the target spacing specified in plans.json, z-score normalized, and inferred with sliding-window prediction and Gaussian-weighted blending. Predicted labelmaps were restored to the original grid for scoring. Subtype predictions were obtained from the

classification head using logit adjustment with the empirical class prior ($\tau = 1.0$). The classification head weights saved during training were reloaded when available.

Table 1: Segmentation Performance - Mean Dice similarity coefficient (per case, averaged across the set)

Whole Pancreas	0.4564
Pancreas Lesion	0.1890

Table 2: Classification Performance (on the same set, the subtype classifier)

Accuracy	0.3333
Macro-F1	0.1667

Per-case metrics are provided in submission_outputs/eval_val/validation_per_case.csv, and a summary table in submission_outputs/eval_val/validation_summary.csv. The confusion matrix is available at submission_outputs/eval_val/cls_confusion_matrix.csv.

Despite aligning inference to the training spacing and applying prior-corrected logits, segmentation performance remains modest, particularly for lesions, and the classifier predictions were effectively dominated by a single subtype, yielding chance-level accuracy and low macro-F1. These outcomes are consistent with a classifier that is prior-biased and under-trained on the available labels. Further analysis and targeted fine-tuning are warranted.

4. Discussion

As can be seen from the results in Tables 1 and 2, the overall performance was modest for both tasks. For segmentation, lesion Dice is particularly low, which is expected when the target is small and heterogeneous: small lesions are easily missed by patch sampling and are disproportionately penalized by Dice. The inference recipe remained conservative: single model, no test-time augmentation or ensembling, and standard sliding-window stitching. Even with spacing aligned to the planning stage and Gaussian blending, limited patch overlap can leave boundary artifacts, and our simple per-case z-score normalization may not compensate for inter-scan intensity variation. I also disabled deep supervision and did not apply post-processing (e.g., connected-component filtering), both of which commonly yield incremental gains.

Classification underperformed more severely and, in prior runs, collapsed to predicting the same subtype for nearly all cases. This behavior is consistent with several interacting factors: (i) task imbalance during joint training (the classification loss was ramped in slowly relative to a strong segmentation objective), (ii) the initial setting that blocked gradient flow from the classification head into the shared encoder (cls_stopgrad_through_encoder=True), limiting the head's ability to shape features, and (iii) fragility of ROI-conditioned pooling—our masked GeM pooling relies on the segmentation logits [4]; when the mask is weak or the foreground is tiny, the classifier effectively reduces to near-global pooling and defaults to the majority class. I applied the same logit adjustment at inference as used in training, but the

near-constant predictions persisted, suggesting the limitation lies in how the head was trained rather than a pure inference mismatch. Class imbalance in available labels and any mismatch between case identifiers and the subtype mapping during training would further weaken effective supervision.

A practical limitation throughout was the compute units in Google Colab. Single-GPU VRAM constrained batch size and patch overlap, as well as restricted wall-time limited total epochs, hyperparameter exploration, and training. These constraints might favor a fast, reproducible pipeline but typically sacrifice peak accuracy, particularly for small-structure segmentation and weakly supervised classification.

5. Future Work

To address the limitations observed in segmentation and classification performance, several future research directions are proposed. A key issue is the imbalance between the two tasks during joint training. The classifier tends to predict a single dominant subtype, indicating insufficient supervision or dominance of the segmentation loss function. Future training strategies could incorporate task balancing techniques, such as dynamic loss function reweighting (e.g., GradNorm or uncertainty-based scaling), or staged fine-tuning, where the classification head is further trained after the segmentation task has converged.

Data imbalance may also be a primary cause of poor classification results. Subtype-aware sampling strategies, synthetic minority class oversampling, or customized data augmentation processes tailored to underrepresented categories can help mitigate this effect. Additionally, combining more advanced data augmentation techniques (e.g., elastic deformation or intensity perturbation) may improve lesion segmentation, especially in challenging anatomical contexts.

Given the limited classification supervision, semi-supervised methods may offer additional benefits. For example, utilizing unlabeled data through pseudo-labeling, consistency training, or teacher-student frameworks can provide a scalable approach to improving the classification head. Finally, current methods for subtype prediction using logistic regression adjustments may still produce outputs with poor calibration effects. Calibration methods may help improve the reliability and interpretability of predictions.

References

[1] F. Isensee, P. F. Jaeger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation," *Nat. Methods*, vol. 18, no. 2, pp. 203–211, Feb. 2021, doi: 10.1038/s41592-020-01008-z.

[2] F. Isensee, P. F. Jäger, S. A. A. Kohl, J. Petersen, and K. H. Maier-Hein, "Automated Design of Deep Learning Methods for Biomedical Image Segmentation," 2019, doi: 10.48550/ARXIV.1904.08128.

[3] L. Maier-Hein *et al.*, "Metrics reloaded: recommendations for image analysis validation," *Nat. Methods*, vol. 21, no. 2, pp. 195–212, Feb. 2024, doi: 10.1038/s41592-023-02151-z.

[4] F. Radenović, G. Tolias, and O. Chum, "Fine-tuning CNN Image Retrieval with No Human Annotation," Jul. 10, 2018, *arXiv*: arXiv:1711.02512. doi: 10.48550/arXiv.1711.02512.

Appendix: GitHub Link

Code is released on GitHub: https://github.com/briannazhao/nnUNetv2_quiz_MultiTask