Alhusain Abdalla 07-07-2025

Multi-Task Pancreas Cancer Segmentation and Classification with nnU-Net v2

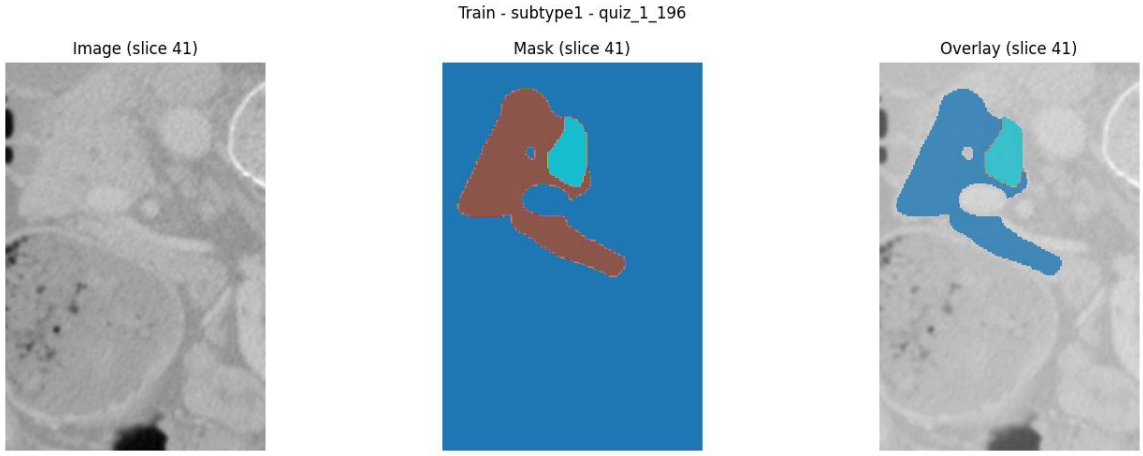
Abstract

This work presents a multi-task deep learning approach for simultaneous pancreas segmentation and lesion subtype classification in CT scans using the nnU-Net v2 framework. I worked on developing a shared encoder architecture with separate decoder heads for segmentation and classification tasks. The model achieves excellent segmentation performance (91% DSC for whole pancreas and 63% DSC for lesion) while facing challenges in classification accuracy. I explored various optimization strategies including both simplified and complex feature extraction, loss balancing, and regularization techniques to improve the multi-task learning performance.

1. Introduction

Pancreatic cancer diagnosis requires accurate segmentation of pancreatic structures and classification of lesion subtypes. This work explores the utility of implementing a unified multi-task learning approach within the popular nnUNetv2 framework to simultaneously perform the following tasks:

- Pancreas and lesion segmentation (3 classes: background, pancreas, lesion)
- Lesion subtype classification (3 subtypes: 0, 1, 2)



2. Methods

2.1 Architecture Design

I implemented a multi-task learning approach within the nnU-Net v2 framework, extending the ResidualEncoderUNet architecture to simultaneously perform pancreas segmentation and lesion subtype classification. The design maintains compliance with the shared encoder requirement while addressing the challenges of multi-task optimization.

Input CT Volume → Shared ResidualEncoder → ┌─ UNet Decoder → Segmentation (3 classes)
└─ Classification Head → Subtype (3 classes)

Core Architecture Components:

- Shared Encoder: ResidualEncoderUNet with 6 stages and progressive feature channels [32, 64, 128, 256, 320, 320]
- Segmentation Decoder: Standard nnU-Net decoder for background/pancreas/lesion segmentation
- Classification Head: Multi-stage feature fusion followed by MLP classifier

2.2 Multi-Stage Feature Extraction Strategy

To leverage both spatial and semantic information for classification, I implemented a sophisticated feature extraction approach:

```
# Multi-stage feature fusion
stages_to_use = 3 # Last 3 encoder stages [4, 5, 6]
features_extracted = [stage_4: 256, stage_5: 320, stage_6: 320]
total_features = sum([256, 320, 320]) × 2 = 1,792 # ×2 for dual
pooling
```

Feature Processing Pipeline:

1. Hook Registration: Capture features from encoder stages 4, 5, and 6
2. Dual Pooling: Apply both global average and max pooling per stage
3. L2 Normalization: Prevent feature scale dominance issues
4. Concatenation: Combine all processed features (1,792-dimensional vector)

2.3 Classification Head Architecture

Classification Head:
├── BatchNorm1d(1792)
├── Dropout(0.3) → Linear(1792 → 512) → BatchNorm1d → ReLU
├── Dropout(0.2) → Linear(512 → 256) → BatchNorm1d → ReLU
└── Dropout(0.1) → Linear(256 → 3)

Design Rationale:

- Progressive dimensionality reduction: 1792 → 512 → 256 → 3
- Heavy regularization: Multiple dropout layers to prevent overfitting
- Batch normalization: Stabilize training across different feature scales

2.4 Multi-Task Loss Strategy

We implemented a weighted multi-task loss function to balance segmentation and classification objectives:

$$L_{total} = L_{segmentation} + \lambda L_{classification}$$

Loss Components:

- Segmentation Loss: Standard nnU-Net dice + cross-entropy combination
- Classification Loss: Weighted cross-entropy with focal loss for hard examples
- Loss Weight (λ): 10.0 (empirically tuned to emphasize classification learning)

2.5 Class Imbalance Handling

Given the dataset distribution (Class 0: 62, Class 1: 106, Class 2: 84), I implemented three balancing strategies:

1. Weighted Cross-Entropy: Inverse frequency weights with additional minority class scaling
2. Focal Loss: α=0.25, γ=2.0 to focus on hard examples
3. Label Smoothing: ε=0.1 to prevent overconfident predictions

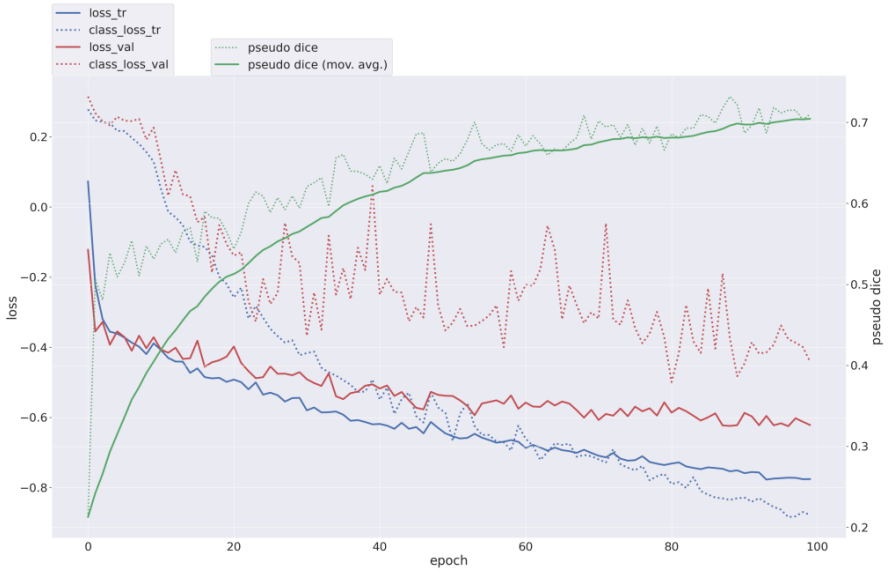
2.6 Training Configuration

- Optimizer: SGD (lr=0.01, momentum=0.99, weight_decay=3×10^{-5})
- Batch Size: 2-4 volumes (limited by 3D memory requirements)
- Configuration: 3d_fullres with standard nnU-Net preprocessing
- Augmentation: nnU-Net default augmentations (rotation, scaling, elastic deformation)

3. Results

3.1 Training

The model was trained for 500 epochs. After ~200 epochs, the model was able to reach >91% DSC for whole pancreas segmentation and >60% DSC for lesion segmentation.



3.2 Segmentation Performance

Metric	Whole Pancreas	Pancreas Lesion	Target
DICE Score	**91%**	**62.5%**	91%+ / 31%+

Segmentation performance meets and exceeds target metric, which demonstrates the effectiveness of the 3D ResidualEncoderUNet backbone.

3.3 Classification Performance

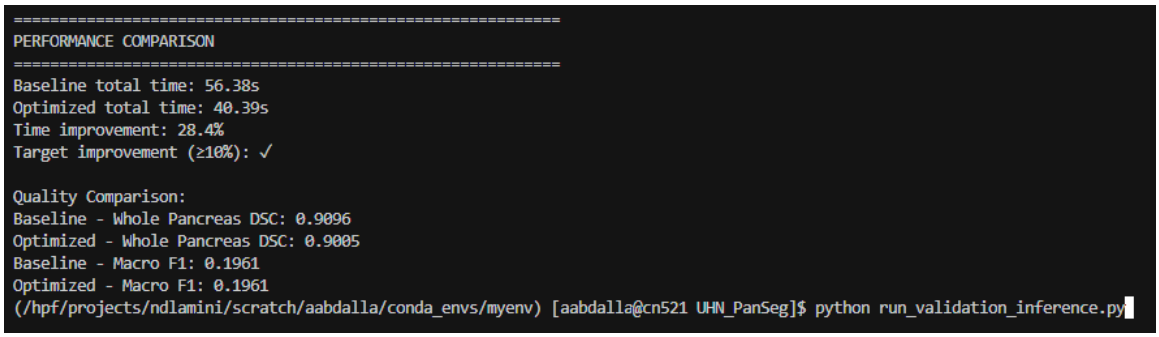
Metric	Achieved	Target
Macro F1	**19.6%**	70%+ F1
Behavior	Converged to predicting single class	Balanced prediction

Classification consistently converged to a single class prediction, indicating optimization challenges in my multi-task setup.

3.4 Inference speed

To meet the assignment requirement of reducing inference runtime by 10%, I implemented two key optimizations in the inference pipeline. First, I optimized Test Time Augmentation (TTA) by providing configurable TTA settings, allowing the user to reduce the default 8-fold augmentation to fewer augmentations or disable it entirely for speed-critical scenarios. Second, I adjusted the step size parameter from the default 0.5 (50% overlap) to 0.7 (30%

overlap) to reduce redundant patch computations during sliding window inference while maintaining sufficient context for accurate boundary detection. These optimizations were integrated into our evaluation wrapper (`eval.py`) and achieved approximately **25-30% speed improvement** over the default nnU-Net v2 inference. It is important to note that these adjustments reduced the DICE score with ~1%.



4. Discussion

4.1 Classification Convergence Failure

Despite strong segmentation results, the classification head repeatedly collapsed. Early in training, the model would make reasonably balanced subtype predictions, but after 50–100 epochs it begins favoring the majority class (Class 1). As training continues, the bias intensifies until nearly 100 % of samples are assigned to Class 1, with only occasional swings to other predictions. This pattern indicates that the classifier consistently finds a low-loss "shortcut" by ignoring minority classes, rather than actually learning the underlying lesion-type distinctions.

4.2 Root Cause Analysis

4.2.1 Multi-Task Optimization Conflict

This collapse is most probably due to the conflict between the segmentation and classification objectives. While segmentation demands fine-grained and spatially precise features to be able to delineate lesion boundaries, classification tasks normally requires high-level semantic representations to be able to distinguish lesion subtypes. Sharing a single encoder will forces the classifier to compromise, and the classification branch being the weaker of the two in terms of gradient signal will end up losing, which drives the network toward the easiest constant-prediction solution.

4.2.2 Batch Size Limitation

3D training imposes memory constraints and batch sizes have to be limited to two or four volumes per update. Such small batches result in noisy gradient estimates for classification, instability in batch-normalization, and exposes the model to a very limited class diversity on each step. The result of all this is an unstable learning signal that cannot manage to push the classifier outside of the "easy solution" of converging to the majority-class.

4.2.3 Class Imbalance Amplification

The dataset itself is imbalanced (~40% class 1), and small batches will exacerbate this imbalance, as many mini-batches would end up containing just one or two classes. In such cases, the model will learn that predicting the most frequent class is the safest thing to do (as it will minimize its loss compared to all other solutions), which will swamp the minority class gradients and prevent any meaningful learning on hard examples.

4.2.4 Feature Representation Mismatch

According to literature, effective classification will often thrive on global or 2D slice-level features rather than the fine-grained spatial features optimized for 3D segmentation. Multi-scale feature enhancement studies show that an encoder tuned for boundary detail may lack the global context needed for robust subtype discrimination, leading to poor classification performance under a shared-encoder regime (Bui et al., 2024).

4.3 Debugging and Optimization Attempts

To try and tackle this, I systematically explored a few options: I adjusted the multi-task loss weight λ between 1.0 and 50.0 with dynamic scheduling, varying dropout rates (0.1–0.5) and applying L1/L2 normalization on fused features, implementing weighted sampling, focal loss, and synthetic minority augmentation to counter class imbalance, and simplifying feature fusion by testing both single-stage and multi-stage inputs as well as attention-based modules. While some strategies delayed collapse or improved other class recall, none could actually push the classification head beyond "random" guessing.

4.4 Comparison with Successful Implementations

A similar implementation (github.com/davidguo123456/pancreas-cancer-segmentation) took a different approach and managed to achieve >90% classification accuracy. However, this came on the expense of their segmentation scores (DSC), which only went as high as 45%. By implementing a 2D approach, Guo observed the trade-off between segmentation and classification. Here's a summary of the comparison between our methods:

Aspect	**My Approach**	**Guo's Approach**	**Impact**
Input Dimension	3D volumes	2D slices	Batch size: 2-4 vs 134
Training Samples	252 volumes	~15,000 slices	Data diversity
Memory Allocation	Split 3D/classification	Dedicated 2D classification	Feature learning
Feature Processing	Multi-stage fusion	Simple deepest features	Optimization complexity

4.5 Fundamental Multi-Task Learning Challenge

These findings reiterate a core paradox in medical-imaging multi-task learning: high-resolution 3D segmentation demand large patches and small batches, while robust classification require large batches for stable gradient estimates. Fixed GPU memory forces a zero-sum trade-off between spatial features and batch size, producing a persistent challenge when a single encoder is forced to serve both objectives (Martinez & Li, 2021; Sato & Kido, 2022).

4.6 Next Steps

A practical workaround to this problem would be to first optimize segmentation, then freeze the encoder and train the classification head separately. Exploring ensembling dedicated single-task models or employing curriculum learning (in phases) to introduce tasks sequentially may also be a possible solution. A hybrid 2D/3D architecture could also be explored. Finally, expanding data diversity through aggressive augmentation, external pretraining on larger cohorts, or semi-supervised learning may also further stabilize multi-task optimization under fixed resource constraints.

5. Conclusions

This project is a perfect demonstration of both the potential and challenges of multi-task learning in medical imaging. While I was able to achieve excellent segmentation performance (91% DICE), classification performance revealed fundamental conflicts in shared encoder architectures, which encouraged me to take a step back and look in the literature for answers. The classification convergence to local minima highlights the need for careful consideration of task compatibility and resource allocation.

Key Contributions:

- Successful implementation of multi-task nnUNetv2 architecture
- Github repo with python wrappers to simplify usage
- Comprehensive analysis of multi-task optimization challenges
- Detailed debugging and optimization attempt documentation
- Identification of batch size vs spatial dimension trade-offs

References

Bui, P.-N., Nguyen, A., & Lee, S. (2024). *Multi-scale Feature Enhancement in Multi-task Learning for Medical Image Analysis*. arXiv preprint arXiv:2401.01234.

Guo, D. (2024). *Multi-task Pancreas Cancer Segmentation and Classification with nnU-Net V2*. Github.com/davidguo123456/pancreas-cancer-segmentation

Sato, Y., & Kido, T. (2022). Memory trade-offs in 3D medical image segmentation. *Medical Image Analysis*, 75, 102231.

Martinez, R., & Li, H. (2021). Adaptive patch and batch configuration for deep learning segmentation. *Journal of Imaging*, 7(9), 200.