RIB FRACTURE DETECTION USING UNET3+ WITH 2.5D PATCHES AND POSITIONAL ENCODINGS

Diego Garcia Cerdas[1], Egoitz Gonzalez[1], Martijn van Raaphorst[1], Erencan Tatar[1], Luka Wong Chung[1]

[1]Artificial Intelligence MSc, Universiteit van Amsterdam, Amsterdam, The Netherlands

ABSTRACT

Rib fracture diagnosis is a common, but time consuming task. A lot of machine learning models already exist to help with this problem, however false positives remain a problem for these models. In our work we build upon a 2D Unet3+[1] model which alleviates this problem. We use a patch-wise method along with adding context to the input slice of the model. Additionally, we add positional encodings as an extra layer of context. In our experiments we found out that adding context helped the model perform better, while positional encodings showed slightly negative results on model performance.

1. INTRODUCTION

Detection of rib fractures in chest-abdomen CT scans is a common and essential task in clinical practice. However, spotting these fractures is a challenging and time-consuming task, requiring professionals to sift through multiple CT scans. Automatic detection with deep learning models provides a way to alleviate this challenge. For example, UNet [2] architectures have proven successful for many medical image segmentation tasks.

Nonetheless, a major issue is that these models sometimes flag fractures that are not there, leading to false positives. High recall is a good thing in medical settings, but too many false positives can mean less accuracy and the risk of overdiagnosing. Unet3+ [1] provides a solution by incorporating deep supervision and a classification-guided module.

In this work, we use modified UNet3+ model for automatic detection of rib fractures. We chose a 2D patch approach to make our models efficient on memory. However, this results in losing important structural information in the inputs. As a solution, we propose to improve the base model by incorporating local context in the patches, resulting in a 2.5D approach. Additionally, to compensate for the loss of global information we encode positional information about the patch coordinates and concatenate it to the input.

Our results show promising qualitative and quantitative results, as measured using intersection-over-union and false positive rate in the validation data. We observe a benefit in including local context information in our inputs. However, positional encodings showed no significant impact on the results. Additionally, we observe a discrepancy with testing data, measured with FROC score, likely induced by misaligned patch distributions.

The code for our implementation can be found at `https://github.com/diegogcerdas/rib-fracture`.

2. DATASET

The MICCAI 2020 RibFrac Challenge establishes a large-scale benchmark dataset for automatically detecting rib fractures from chest-abdomen CT scans. The dataset contains 420 scans for training, 80 for validation, and 120 for testing. Each scan comprises a varying number of 512×512 axial slices. Ground-truth annotations are provided for training and validation. These consist of pixel-level masks of rib fracture regions, plus classification labels. Testing results can be evaluated via submission to the challenge's platform. An example of these data is displayed in figure 1.

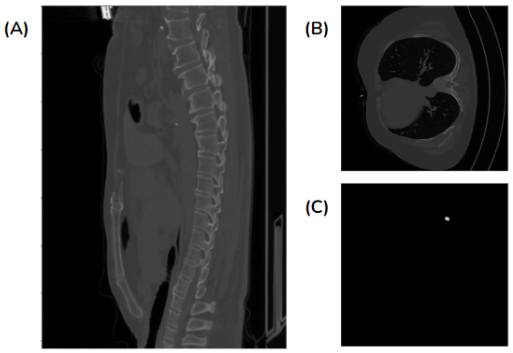

Fig. 1: Visualizations of example data. A) CT slice from the sagittal plane. B) CT slice from the axial plane. C) Fracture mask corresponding to slice in B.

We performed an extensive analysis of the data, in an attempt to derive heuristics for the hyperparameters of our preprocessing and model. We preferred this data-driven approach over a full hyperparameter search since the latter would be restrictive on our time and computing resources. Table 1 shows statistics of fracture size and scan height across the training set. Relevant quantities derived from these analyses are referenced throughout the report.

	Fracture size			Scan size
	size_x	size_y	size_z	size_z
mean	27.11	22.85	15.02	367.08
std	16.91	13.55	8.73	51.73
max	166	137	106	721
min	3	1	1	239

Table 1: Fracture and scan size statistics from data analysis.

2.1. Preprocessing and Data Sampling

An overview of our preprocessing pipeline is displayed in figure 3. We choose to use small patches as input to our model as opposed to whole scans or slices for three main reasons. First, it allows the model to better focus on ribs, which typically occupy a tiny fraction of the full slice. Second, it increases the effective size of the training dataset since we can extract multiple different samples for a single slice. Lastly, their lightweight nature permits a more flexible use of available memory.

Furthermore, we perform all preprocessing steps on-the-fly during data loading to avoid increasing the amount of storage space occupied by the data. In the following subsections, we describe these steps, as well as the two sampling methods we employ.

2.1.1. Preprocessing of image and label slices

As a first step, we load a complete raw slice for a given CT scan and remove soft tissue and air information, which are assumed irrelevant to the task of detecting rib fractures. To do so, we clip the pixel intensity values to a range between 100 and 2000. These values were found to successfully encapsulate bone regions. However, this range also preserves pixels belonging to the backplate over which the subjects were lying. To fix this, we define a cutoff height hyperparameter $h_{\text{cutoff}} = 450$ derived from our data analyses and use it to exclude these pixels during patch sampling.

Since we do not intend to perform multiclass classification, we turn the accompanying annotation data into a binary mask signaling the location of fractures in the image slice. We refer to this as the label slice.

2.1.2. Class-balanced random sampling of patches

Once we have isolated bone pixels in the slice, we proceed to extract a random small patch. To do so, we randomly select one of the bone pixel locations and use it as the center coordinate for cropping a patch (from the image slice, as well as the label slice). Patch size is chosen to be 64×64 pixels because it was found to encapsulate the most fractures. Here, we ignore all pixels with height $> h_{\text{cutoff}}$ to avoid sampling from the backplate. We also add a random offset in the $x-$ and $y-$direction from the center coordinate to better align the training and testing patch distributions (the latter is described in the following subsection).

To aid optimization, we require all training batches to be class-balanced, with classes defined as "fracture" and "non-fracture". To this end, we define a selection criterion for each of these classes. For fracture patches, we require a minimum proportion of 1% of pixels in the labels to be non-zero. In our case, this is ~ 40 pixels. For the non-fracture patches, we require all label pixels to be zero. Training batches are built in an iterative manner, adhering to these criteria. Due to the nature of our approach, each epoch presents a different set of sampled patches, with each epoch containing around 91 thousand examples.

Finally, we apply data transforms. Images patches are normalized with $\mu = 0.0268$ and $\sigma = 0.0841$. These values are estimated from a set of 500 thousand randomly sampled patches, in order to better adhere to the newly induced data distribution. Patches are resized to 128×128 pixels and randomly flipped horizontally and vertically.

2.1.3. Sliding-window sampling of patches

During testing, we need to provide predictions for every patch in a slice. Therefore, we cannot follow the approach of random patch sampling. Instead, we implement a sliding-window approach that extracts every single patch using a stride of 32 pixels (for half-overlapping windows). Therefore, every pixel gets at least one prediction, and predictions for pixels in overlapping regions are aggregated during post-processing.

3. METHOD

3.1. Model

We base our approach on deep learning models for medical image segmentation that work with 2D data. In contrast to 3D models, the former models ease restrictions on memory and model capacity. We also focus on models that tackle the false positive problem.

The following sections describe our chosen baseline model, UNet3+. In section 3.2, we describe our proposed improvements to this model.

3.1.1. Network architecture

Compared to the classic UNet[2] architecture, the UNet3+[1] model introduces the following features:

- **Full-Scale Skip-Connections**: Each decoder scale is connected to every encoder output from the same and higher scales.
- **Deep Supervision (DS)**: Each of the decoder outputs is directly supervised by the ground truth.

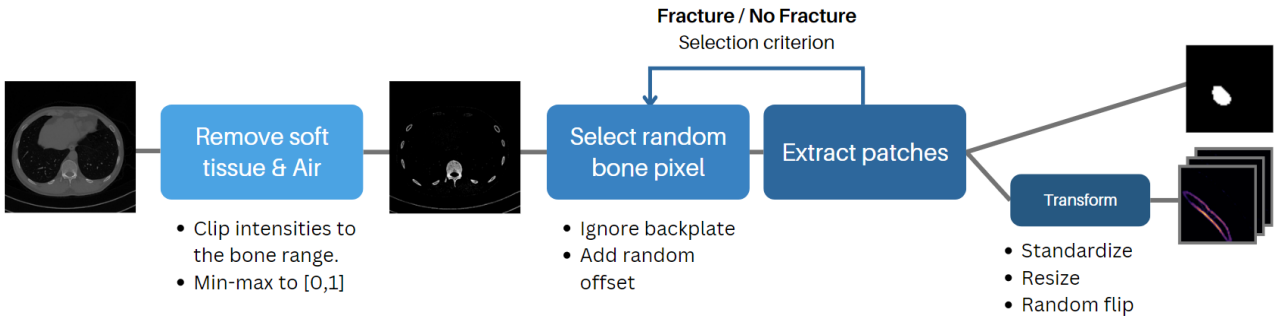

Fig. 2: Overview of the preprocessing pipeline and class-balanced random sampling of patches.

- **Classification Guided Module (CGM)**: This is an extra classification task, designed for predicting whether the input has a fracture or not. The CGM was introduced to address and alleviate the common problem of false positives in medical image segmentation.

3.1.2. Training Objective

Each of the decoder scale outputs is supervised by the ground truth segmentation mask. This supervision is done by computing a hybrid loss that compares predictions at each scale level to the ground truth. $\mathcal{L}_{hyb} = \mathcal{L}_{iou} + \mathcal{L}_{msssim} + \mathcal{L}_{foc}$ This hybrid loss contains the following components:

- **Intersection over Union (IoU) loss** calculates the intersection of the predicted patch and the ground truth. The lower the IoU loss, the better the bounding box surrounds the object of interest.
- **Multi-Scale Structural Similarity (MS-SSIM) loss** is a perceptually-motivated loss function used in image processing that measures the structural similarity between two images at multiple scales. Here, images are compared at various scales by applying a Gaussian kernel at each scale, allowing the function to capture both the luminance, contrast, and structural information of the images. The MS-SSIM value is calculated as the product of these comparisons at each scale, with a higher value indicating greater similarity.
- **Focal loss** addresses class imbalance during training in tasks like object detection by applying a modulating term to cross entropy loss in order to focus learning on hard misclassified examples

If the CGM module is used, then the output of the CGM branch is also supervised by the ground truth using Binary Cross-Entropy (BCE) loss between the class prediction and the true class, i.e. whether the ground truth segmentation mask has any fracture pixels.

3.2. Proposed Improvements

3.2.1. Context slices

Operating with 3D data can provide important structural information, but it is often too heavy on memory. As an alternative, one can concatenate adjacent 2D slices in order to introduce local context. We refer to this approach as 2.5D. The usefulness of the 2.5D approach relies on the ability of the model to detect correlations in the data between different slices while predicting a segmentation mask for the slice in the middle.

We control the size of this local context with a hyperparameter C. It defines the amount of layers to include in each direction (above and below) from the middle slice. Based on our data analysis as we saw in Table 1 (size z for fracture size), the best choice seems to be $C = 8$, since the size of the window $1 + 2 \times C = 17$ will cover most of the fractures.

3.2.2. Positional encodings

To compensate for the loss of global positional information when we focus on smaller patches, we add positional encodings as extra channel inputs appended to the previous context slices. Since they are independent, they will not conflict with spatial data. We used concatenation instead of addition since we thought addition might affect the prediction negatively.

We decided to encode the (x, y, z) coordinates of the location of the center of the patch within the scan. We analyzed the alignment of the data for scans with different heights. This showed that the relative shape of the rib cage was the same while the amount of slices differed a lot. We thus opted to consider the z coordinate to be relative to the scan size, which was done by normalizing the values.

The first naïve approach was to use a simple sin/cos wave as the encodings. However, they have a Nyquist frequency limit [3] given by half the amount of pixels of the target encoded image. Our final approach is to use the encoding used in transformers [4], which combines both sin/cos signals and

uses a wider range of frequencies. For simplicity, we choose to add a layer for each encoded value (x, y, z).

3.3. Postprocessing + Evaluation/Validation

While evaluation through slicing-window sampling gives a complete picture of how well our models perform, we found it to be time-prohibitive (evaluation per slice takes around 3 seconds). Therefore, we (approximately) validate our models after each training epoch using random patch sampling. As our validation metrics, we compute patch-wise intersection-over-union (IoU) and false positive rate (FPR).

For testing, we perform a complete evaluation through sliding windows. Overlapping windows are aggregated via their average prediction. We compute FROC scores by submitting them to the challenge's website. To comply with the submission format, we perform the following postprocessing steps: we run a connected-component analysis to identify individual fracture detections, and provide an average confidence for each of them.

4. EXPERIMENTS

In section 4.1, we describe the design and results of our exploratory round of experiments. These results motivate the choice of conditions for our final experiments, described in section 4.2, where we identify the effect of our proposed improvements on the baseline model. Finally, section 4.3 provides details on implementation and training settings.

4.1. Exploratory experiments

To justify our choice of conditions for the final experiments, we conducted an ablation study involving various combinations of the improvements we proposed for the baseline UNet3+ pipeline. Based on the insight gained from prior experiments that ran up to 15 epochs, we observed that by the third epoch, we could effectively identify models with superior performance. Due to both this observation and time constraints, we limited the training duration of all ablation study models to 3 epochs, ultimately selecting the top 3 models based on their lowest losses and optimal scores in the FPR and IoU metrics.

4.1.1. Ablation of network components

To investigate the effectiveness of the different UNet3+ components when using context, we trained the base UNet3+ without DS, with DS, and with DS + CGM, all using a context size of 8. For these experiments we did not use the focal loss. The results in table 3 show that the UNet3+ with DS but without the CGM module performs best, especially with respect to the false positive rate (Experiment 2).

4.1.2. Ablation of losses

Because we experienced some issues with the focal loss and the MS-SSIM loss during our first training round, we decided to investigate the effectiveness of using different combinations of losses to train the models. For both the UNet3+ model (DS, DS + CGM) we examined the effect of adding the MS-SSIM loss and the focal loss to the IoU loss. Again, we used a context size of 8 for all experiments. The optimal model resulting from this experiment coincides with the findings of the network components ablation experiment (Experiment 2).

4.1.3. Variation of context size

To investigate the effectiveness of our proposed method of using different context sizes, we trained the UNet3+ model (DS, DS + CGM) on context sizes of 0, 8 and 16. For this experiment, table 3 shows that the optimal model was the UNet3+ with DS and a context size of 0 (Experiment 8).

4.1.4. Variation of positional encoding

Our second contribution is the use of positional encoding. We investigated its effectiveness by comparing the results of the UNet3+ model (DS, DS + CGM) with and without the use of positional encoding. Again, we used context size 8 for these experiments. This experiment showed the effectiveness of positional encoding with the optimal model being the UNet3+ with DS and positional encoding (Experiment 12).

4.2. Final experiments

Our ablation study indicates that the optimal models for conducting the final three experiments are as follows: the UNet3+ with Deep Supervision and a context size of 8, the UNet3+ with Deep Supervision and a context size of 0, and the UNet3+ with Deep Supervision, a context size of 8, and the inclusion of positional encoding. In contrast to the ablation experiments, we trained the final experiments for 50 epochs each, further improving the optimization.

Table 2 shows the results of the abovementioned final experiments. Like the previous ablation experiments, we employed the IoU and FPR metrics for validation, finding them to be suitable indicators when qualitatively assessing the outputs generated by our models (refer to figure 3). While these metrics correlate with FROC metrics, the latter show significantly lower magnitudes, evidencing poor performance on the test set. We provide an interpretation of these results in the Discussions section.

4.3. Training settings and implementation

We train all models using a batch size of 32, learning rate 1e-4, and weight decay 1e-5 with RMSprop optimizer. We

Exp. name	Setup/Config		Results/Metrics				
	Context	PE	Train Loss↓	Val Loss↓	FPR↓	IOU↑	FROC↑
exp08			4.222	4.399	0.008265	0.5473	0.0033
exp02	8		3.691	3.937	0.005948	0.6181	0.0204
exp12	8	✓	3.770	3.991	0.006269	0.6073	0.0147

Table 2: Main experiments on *50* epochs. All experiments use the UNet3+ model with Deep Supervision and without CGM module. The segmentation loss only uses IoU+MSSSIM components of the hybrid loss. Context corresponds to the context size used; no context means 0. PE says whether positional encodings are used. FPR and IOU correspond to the false-positive-rate and intersection-over-union metrics, respectively.

use early stopping to preserve the best-performing checkpoint based on validation metrics.

We trained all models in the Snellius cluster, employing an NVIDIA A100 GPU and 18 CPUs. Our implementation[1] is based on the PyTorch Lightning library, and we use Weights and Biases for logging.

5. DISCUSSION

The original UNet3+ [1] paper uses a hybrid-loss consisting of the focal loss, proposed as a way to deal with class imbalance, a MSSSIM-loss to incorporate image details at different resolution scores, and the standard IoU loss. The ablation experiments showed that the focal loss does not help with the training and since we addressed the class imbalance issue by focusing on patches around bones, we decided to not include it in the final experiments.

Due to limitations time-wise as well as resource-wise, we were not able to explore and experiment with 3D models which are better performing and the current state-of-the-art models. For the same reason, we were also unable to perform a grid search for the optimal hyperparameters and thus decided to choose these based on our data analysis. Due to storage limitations, we were unable to store all the data and had to preprocess the data on the fly. Additionally, the sliding window validation approach is time-consuming, since doing it this way the program was opening and closing files a lot which is not feasible.

We focused on a 2D approach due to our limitations and decide to compensate the 3D information loss adding context and positional encodings. Experiments show that context was really helpful for the model, but the positional encodings containing patch global location information did not really benefit the model. This might be happening because of our design or implementation choice for the positional encodings. For future work it might be interesting to investigate conditioning a unet in other ways or using a different positional encoding design.

Regarding final results, we can clearly see that the computed FROC values are bad despite the training metrics being relatively good for the task in hand. The reason behind this might have been in the post-processing pipeline, when binarizing the results to match the submission format. When performing the connected component analysis we got 150 regions instead of the 2-5 we think we were supposed to get. This results might also be due to the domain shift (distributional shift) in the data that the model sees between training and testing.

6. CONCLUSION

Contrary to the initial hypothesis where a UNET3+ model with DS and CGM performs best with largest context size we found that the best performance was reached when using a model with only DS and a context size of 8 and without positional encodings. From what we see in the results we can conclude that giving context to the model did improve performance. Adding positional encodings did not add to the performance and even seems to make the model perform slightly worse. However, since we did not test different ways of adding the positional encodings we can not say that this is always the case.

7. REFERENCES

[1] H. Huang, L. Lin, R. Tong, H. Hu, Q. Zhang, Y. Iwamoto, X. Han, Y.-W. Chen, and J. Wu, "Unet 3+: A full-scale connected unet for medical image segmentation," 2020.

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[3] E. Robinson and D. Clark, "Sampling and the Nyquist frequency," *The Leading Edge*, vol. 10, no. 3, pp. 51–53, 03 1991. [Online]. Available: https://doi.org/10.1190/1.1436812

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[1] https://github.com/diegogcerdas/rib-fracture

Appendices

A. ABLATION STUDY

Exp. name	Setup/Config					Results/Metrics			
	DS	CGM	Context	Pos. Enc.	Seg. Loss	Train Loss↓	Val Loss↓	FPR↓	IOU ↑
exp01	✓		8		IoU	3.549	3.580	0.034	0.587
exp02	✓		8		IoU+MSSSIM	4.165	4.262	0.015	0.550
exp03	✓		8		IoU+MSSSIM+Focal	5.851	6.142	0.000	0.074
exp04	✓	✓	8		IoU	3.887	4.094	0.303	0.512
exp05	✓	✓	8		IoU+MSSSIM	5.078	5.279	0.295	0.515
exp06	✓	✓	8		IoU+MSSSIM+Focal	60.636	59.457	0.258	0.134
exp07			8		IoU+MSSSIM	0.035	0.046	0.011	0.442
exp08	✓				IoU+MSSSIM	4.457	4.526	0.017	0.493
exp09	✓	✓			IoU+MSSSIM	5.384	6.963	0.448	0.518
exp10	✓		16		IoU+MSSSIM	4.168	4.304	0.010	0.504
exp11	✓	✓	16		IoU+MSSSIM	5.088	5.299	0.279	0.503
exp12	✓		8	✓	IoU+MSSSIM	4.184	4.333	0.014	0.530
exp13	✓	✓	8	✓	IoU+MSSSIM	5.074	5.185	0.288	0.529

Table 3: Configuration and results for ablation experiments for *3* epochs. DS and CGM are whether the model includes Deep Supervision and the CGM module respectively. Context corresponds to the context size used; no context means 0. Pos. Enc. says whether positional encodings are used. Seg. Loss lists the components of the hybrid loss used for each experiment. FPR and IOU correspond to the false-positive-rate and intersection-over-union metrics over the validation set, respectively. The highlighted experiments are the main experiment configurations.

B. QUALITATIVE RESULTS

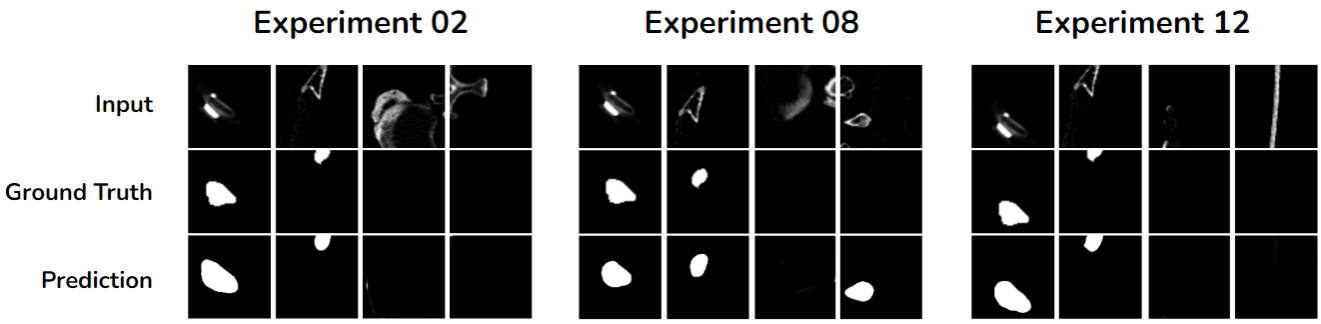

Fig. 3: Visualization of the patch predictions made by the model. On top is the input patch below that is the ground truth and the bottom row contains the predicition made by the model.