Skull Defect Detection Using CT Foundation Embeddings

Shawn Varma

Varma3211@gmail.com

March 2025

Abstract

Case prioritization in hospitals and the medical industry is an essential capability to ensure that patients get access to care in time sensitive scenarios. The ability to effectively triage incoming medical information at the point of upload reduces the demands on various members of the healthcare system to evaluate patient condition. In this case study I explored the application of such a system in the context of cranial skull defects to enable a custom implant design team to determine the best course of treatment to offer for neurosurgical and craniofacial applications. Using transfer learning from the CT Foundation Model [1], a result of 64% AUC on binary classification for defect presence was achieved with just 100 total samples of data. CT Foundation is trained on a large variety of medical applications so sourcing its embeddings enables classification even at small scales. The compression offered from training on CT scan embeddings, when compared to natively processing 3D scan volumes, allowed for fast training times and rapid iteration. Future work can expand the scale of the models with little computational overhead due to the use of embedding based workflows.

Methods/Results

The starting database of CT skull data was acquired from the AutoImplant 2020 research challenge/paper [2]. Contents included 100 anonymized healthy patient scans in NRRD format with thresholding set to bone Hounsfield unit ranges. The resulting skulls therefore only show bone shape but provide no information of the underlying soft tissue. Each skull was then artificially defected based on the code provided on the AutoImplant project page [3]. This procedure involves creating a cube or sphere with size and position randomly generated (within a user defined range) which is then subtracted in the scan volume to create an artificially defected skull. Scans were then filtered using connected component analysis to remove floating fragments leftover from the cut operations.

CT Foundation embeddings allow for training of models using scan and text pairs. A JSON file with class labels had to be created and have assignments for each study. To accomplish

this task efficiently, I wrote a simple script to sequentially fill out the image classes with corresponding values. The workflow was to run the script through Visual Studio Code's terminal and open each scan in 3D Slicer for review as shown below:

Figure 1: Sample terminal workflow steps and corresponding scan. Each scan is manually reviewed and the number for the relevant tags was input in the terminal which automatically sends it into the JSON containing scan labels.

From the two datasets of healthy and defected scans, studies were randomly selected for the final database of 100 scan + text pairs. At the start I wasn't sure of all the labels that would be necessary so I cast a wide net to capture details that may be of interest later. The initial labels are shown in the following table:

Defect Present	Location	Shape	Suggested Treatment	Additional Information
Yes	Left parietal region	Square	Implant necessary	N/A
No	Right parietal region	Rectangle	No intervention required	Scatter present on dental implants
	Left frontal region	Circle		Patient intubated
	Right frontal region	Oval		Skull and upper body scan
	Left temporal region	Triangle		Scatter present in right parietal area
	Right temporal region	Trapezoid		Scan cropped
	Left occipital region	Irregular		
	Right occipital region			
	Bifrontal region			
	Biparietal region			

Table 1: Initial labels based on defect presence, location, shape, suggested treatment, and additional information.

This labelling strategy provided several challenges. First, the number of defect locations is very high for the number of samples present. While it may be clinically relevant, some defects are not highly present in this dataset with a particular lack in the temporal defect locations or circular shaped defects. Splitting the sides for each bone region adds additional complexity which may be hard for the model to adequately learn. Shapes also have some redundancy such as the similarities between circle/oval and rectangle/square. The strategy for treatment is simple in the sense that if a defect is present then an implant will always be necessary so treating it as a learnable parameter may not be particularly helpful. Reflecting on this I decided to simplify the classes as shown below:

Defect Present	Defect Location	Defect Shape
Yes	Frontal	Square
No	Parietal	Trapezoid
	Occipital	Triangle / Irregular
		None

Table 2: Simplified labels

Final data distributions were also visualized after randomly seeding train, validation, and test datasets.

Figure 2: Final data distributions

To get the embedding data for each CT scan, I first had to upload the scans to the Google Cloud DICOM Store via Google Cloud SDK. The first challenge here was converting the NRRD files to the correct DICOM format. Initially the files were converted to a single file .dcm format

DICOM without any of the corresponding metadata of each scan attached. This failed when testing in the sample Google Colab notebook file [4] that was provided with the CT Foundation blogpost. This led me to printing the scan properties of the sample study within the notebook to learn more about the requirements. The file conversion was then updated correspondingly so that each scan ID contained each slice within a corresponding folder with correct metadata identifying the patient (for example patient A0001 whose folder contained slices 1 through 320). The debugging info from the end of the Colab document was useful for guiding the troubleshooting process.

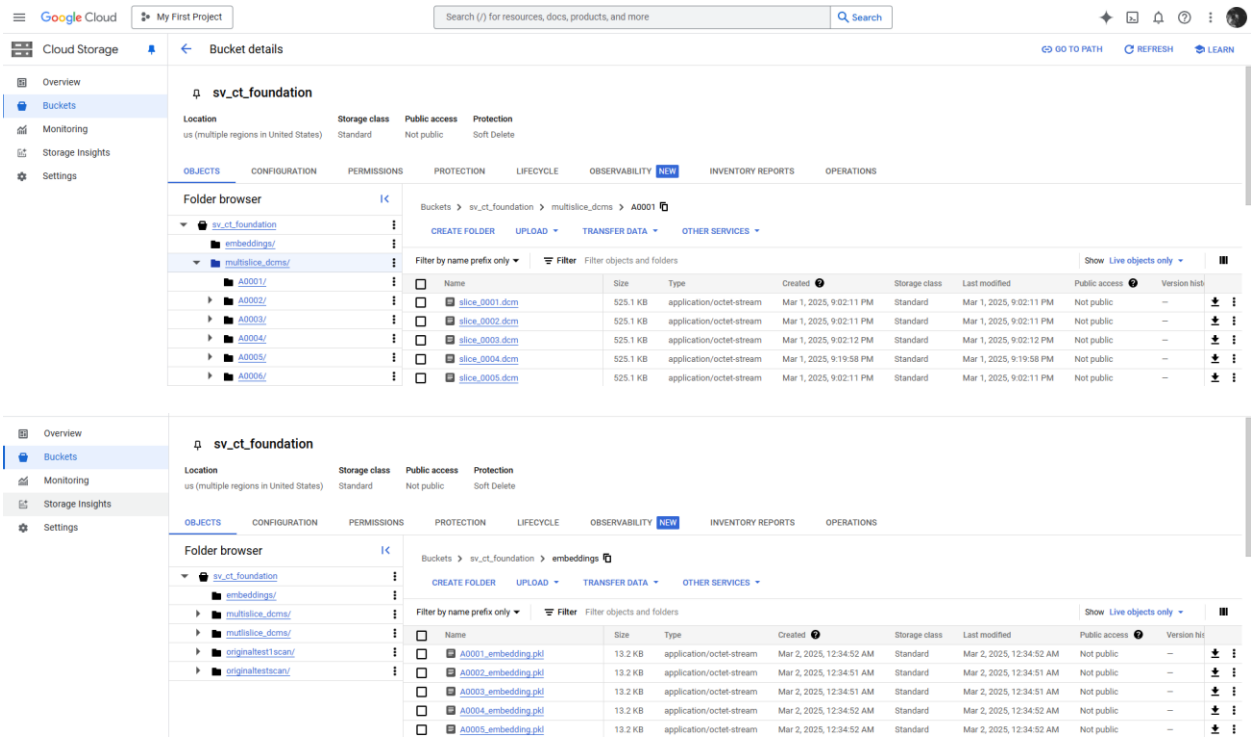

Figure 3: Cloud Storage Folders for DICOM studies and embedding .pkl files

Embeddings were then processed in two batches using the Colab notebook.

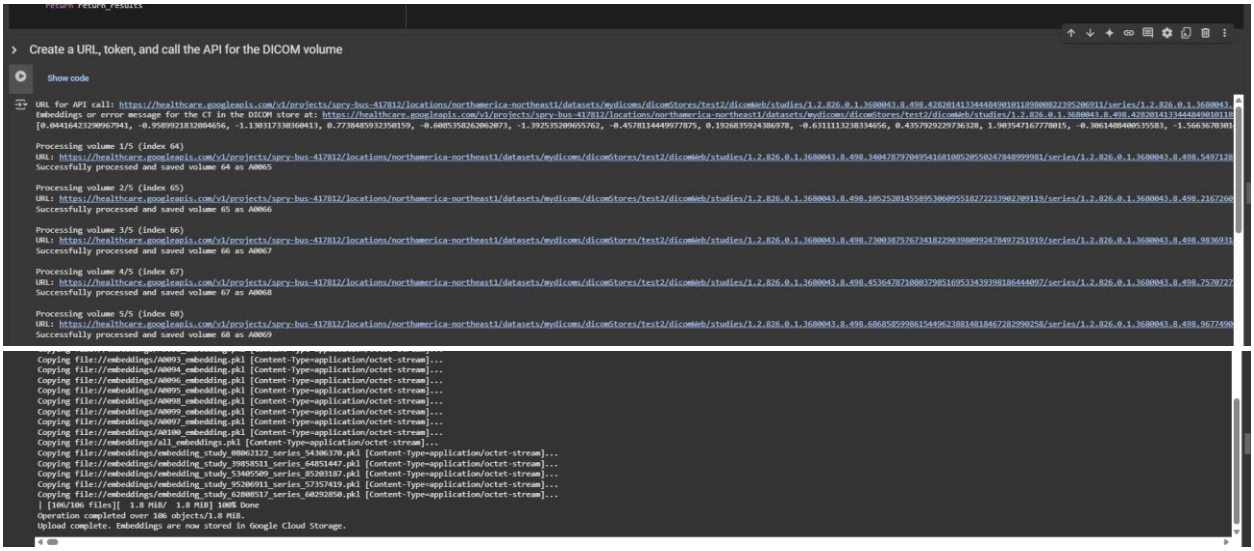

Figure 4: Successful runs from second embeddings batch generation

With embeddings in hand, the necessary files were ready for a model training run. After the initial test I opted to run a principal component analysis to better understand the contributions of the 1,408-dimensional vector.

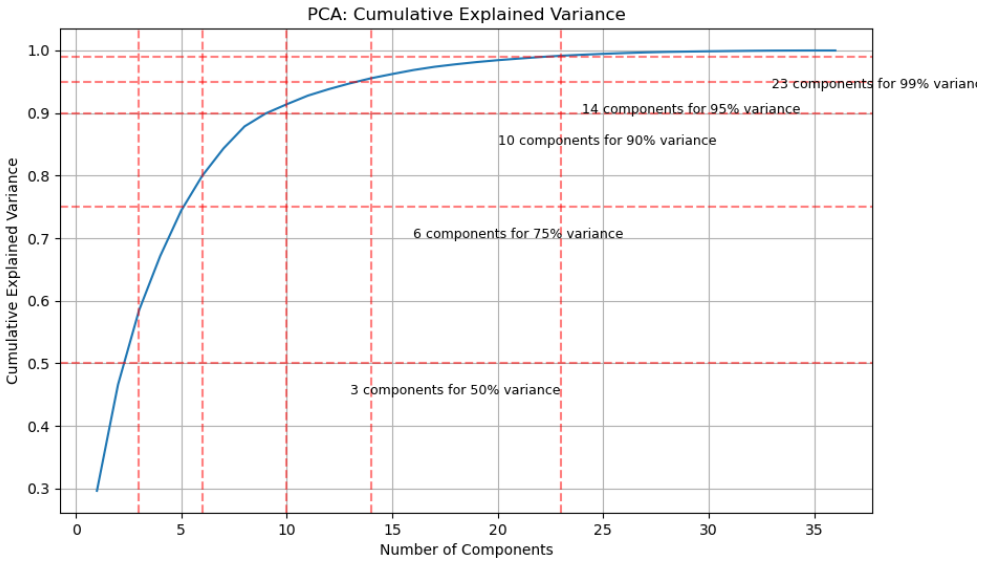

Figure 5: Labelled PCA visualization of embedding space variance

The PCA shows that there could potentially be redundancy in the contributions of the embedding vector components. I thought it may be useful to look for a dimension reduction given that my goal was to make the most of a limited dataset and I wanted to minimize the contribution of any unrelated features as potential sources of noise. Using mutual information, the 50 most relevant embeddings features were then selected and used for training. Initially I

trained a multilayer perceptron (MLP) based model but ran into issues with data size (training runs discussed later in the report) so I opted for a logistic regression based binary classifier. I also had the hypothesis that, given that the images were binary masks, there were fewer features available to use from the images.

Regularization strengths were selected via tuning and the optimal outputs were used in the final plots. Both L1 and L2 approaches resulted in approximately 61% AUC which is in line with the values expected from the CT Foundation AUC Plots at a 10^2 sample size scale.

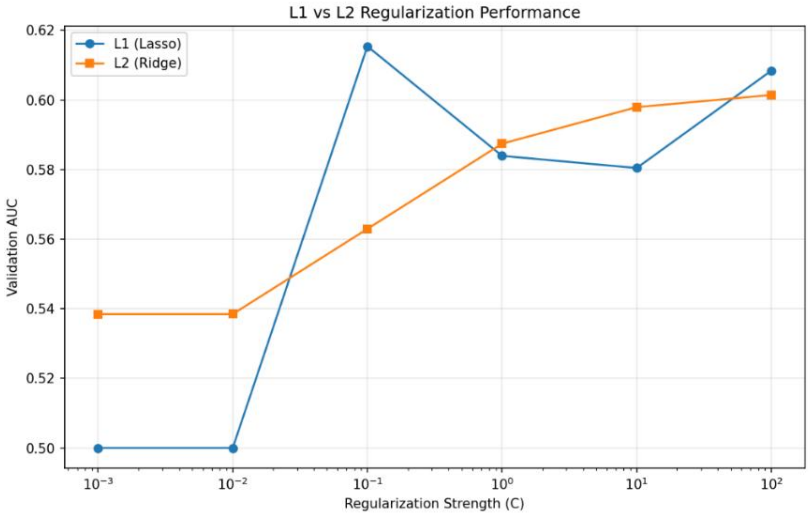

Figure 6: Improved L1/L2 performance at increasing regularization strength values

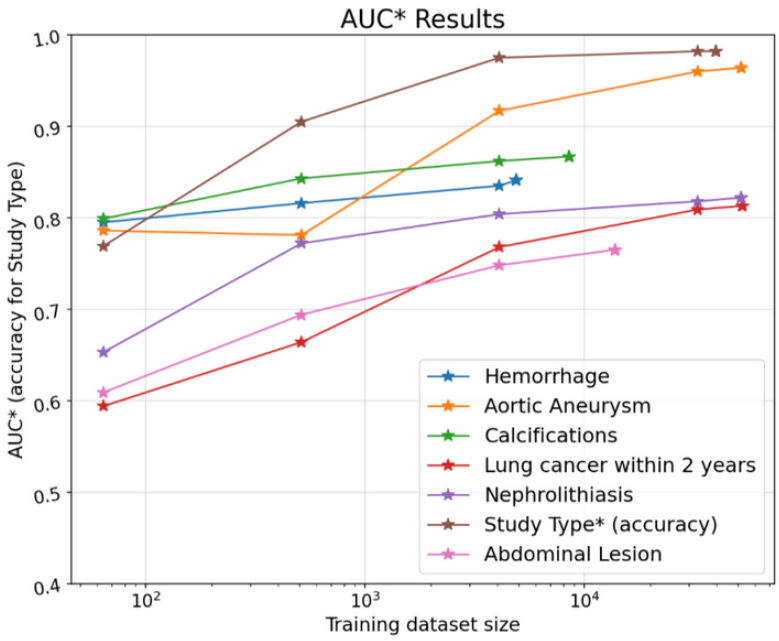

Figure 7: CT Foundation AUC versus dataset size plot from CT Foundation blogpost [1]

A confusion matrix was generated to further probe the models' results:

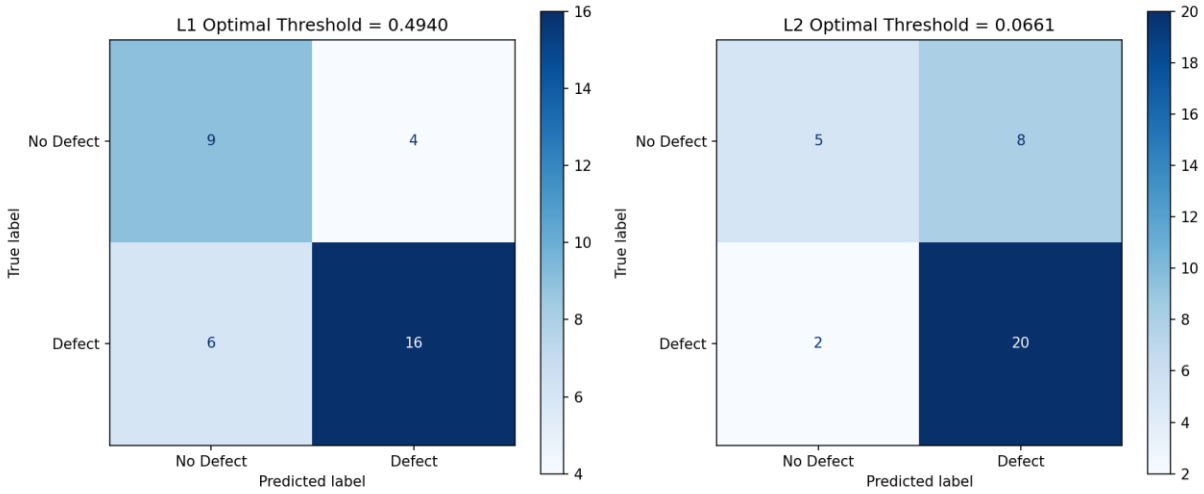

Figure 8: Confusion matrices showing predicted values vs true values occurrence frequency

The main takeaway from the confusion matrices is that each model can correctly guess the defect being present more consistently than it is able to identify that no defect is present. In a clinical setting a bias towards guessing that a defect is present would result in further review by a trained professional who could exclude incorrect classifications manually. As more data is added to the models, we can also reduce the rate that misidentifications occur (such as the orbital volume being mistaken for a nonstandard void in the skull). The alternative scenario of missing defects entirely would be more concerning but is happening at a lower rate in these models. The L2 model results in fewer misses for defect detection but requires more manual review by a trained professional due to the extra no defect present scans being passed.

To evaluate if dimension reduction was the right decision I ran a comparison using both embedding datasets with results shown below:

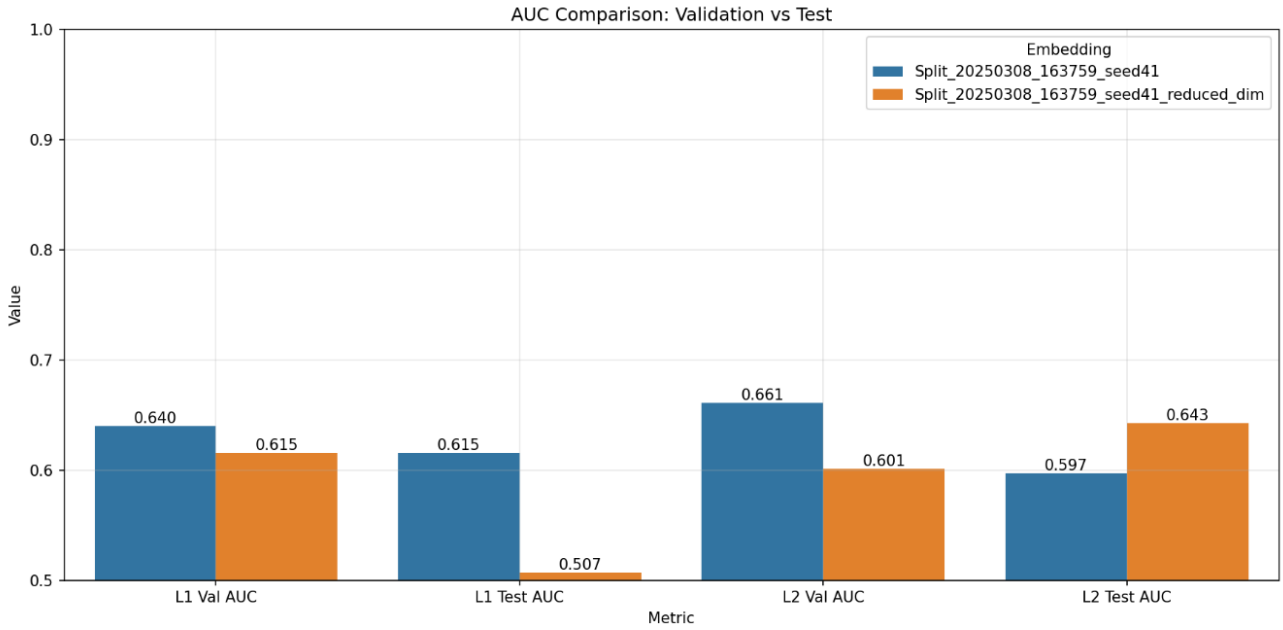

Figure 9: Validation and Test AUC values for L1 and L2 methods. The Y axis starts at 0.5 to show the amount of improvement over 50% random guessing.

Given these results it was found that the full set of embedding vectors did result in a better performing model overall. The only time this did not show to be the case was in the scenario of L2 Test AUC which suggests that the L2 approach generalized better with the dimensional reduction. L2 is more sensitive to outliers which could indicate that there were lower value outputs pulling down performance in the other scenarios. The smaller differences between L2 validation and test scores suggest better generalization (less overfitting) than with the L1 model.

Initial attempts at modelling scan classification used a neural network approach with an MLP for multitask evaluation. This used the 1,408-dimensional embedding vector in the input layer and passes it through subsequent hidden layers of 512 > 256 > 128 until the final output layer. Rectified linear units (ReLU) and batch normalization are used between layers to stabilize training. Training was run using the reduced dimension embeddings to benchmark against the initial training run. The results were surprising and indicate the model failed to generalize and instead learned to guess the most common result from the data.

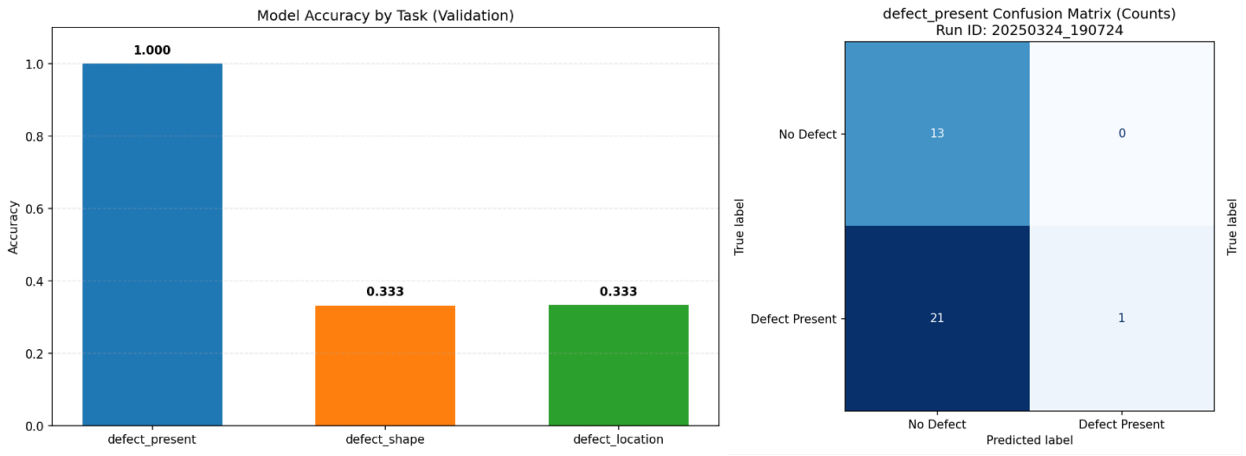

Figure 10: Validation performance of each task and confusion matrix values for reduced dimensions

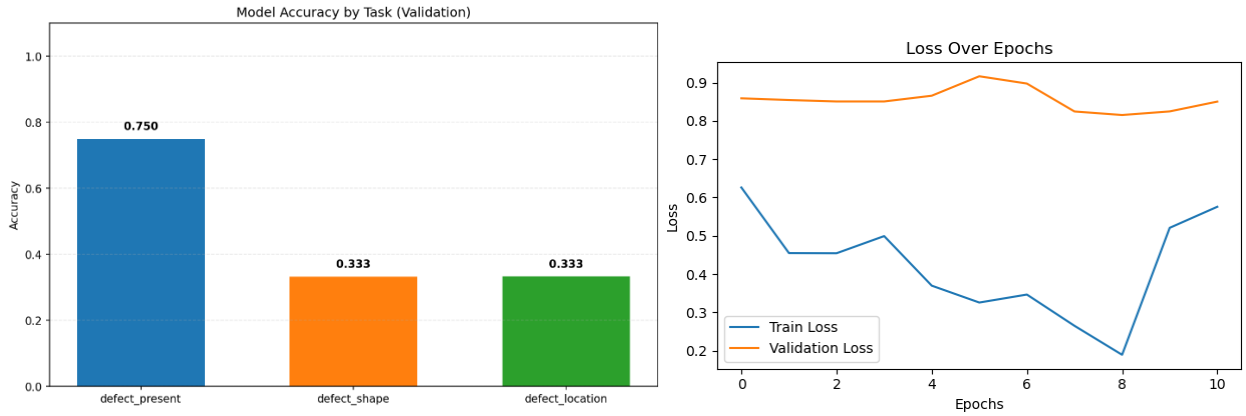

Figure 11: Validation performance of each task and loss plot for full dimensions model. The confusion matrix showed similar results so loss was shown instead to emphasize the misalignment between train and validation loss.

A perfect defect present accuracy value in the model validation results indicated a need for further inspection. The corresponding confusion matrix suggested that the model was learning to guess no defect present for all cases. The other two tasks performed below random guessing and were therefore ignored. Bias for avoiding false negatives could have occurred here as missing defects is worse than guessing a defect is not present. In practice this does not give a clinical user any actionable information that they could not otherwise get by reviewing the scan manually.

Results did not improve for multiclass hierarchical methods nor for k-means sampling. It is likely that these could show better results with a larger dataset. It was because of these results that I decided to focus only on defects being present as the task to focus on with the L1/L2 classifier methods.

Conclusion

The best performing models achieved AUCs between 0.6 and 0.64 which were consistent with the expected values provided in the original CT Foundation blog post for this dataset size. The overall experience of using the embeddings allowed for much lower computational requirements when training the model. This allowed for rapid iteration when making model adjustments and removed the need to source GPU capacity. Given the integration of multiple medical imaging applications with text, I could see this workflow being a great fit for many clinical applications in the future.

One such application I thought could make for a good case study is the current workflow that I've seen on the medical device side of the industry in custom implant design teams. Multiple patient specific implant teams exist in areas such as CMF, spine, and knee surgery where surgeons can send scan data to a design team for review. The design team needs to know the type of scan present so that they can correctly assign a corresponding product (if one exists for that application). Identifying defects present can tell design teams immediately whether they should assign a void filling solution, an augmentation solution, or a virtual resection solution before having an engineer manually review. With the ability to pre-check a scan's contents, the teams are given automatic broad case categorization of the images which could then be sent to a more traditional scripting system to verify parameters like slice thickness, slice spacing, and differentiating cone beam from medical grade studies. Overall, this provides a system for creating clinical image metadata that allows surgeons to get confirmation of case compatibility upfront. Sometimes patients are waiting at the hospital to hear back from design teams if their latest scans are compatible with existing regulatory approval processes, so they also gain from these time savings. When processing a large volume of clinical requests, prioritization and categorization are valuable candidates for automation and teams will continue to look for solutions to help make case management as efficient as possible.

References

1. Taking medical imaging embeddings 3D. Google Research Blog. https://research.google/blog/taking-medical-imaging-embeddings-3d/
2. Li, J., Pepe, A., Gsaxner, C., von Campe, G., & Egger, J. (2020). A baseline approach for AutoImplant: The MICCAI 2020 cranial implant design challenge. https://arxiv.org/abs/2006.12449
3. Li, J. (2020). AutoImplant: A baseline approach for the AutoImplant challenge. GitHub repository. https://github.com/Jianningli/autoimplant

4. Google Health. (2023). CT foundation demo. Google Colab. https://colab.research.google.com/github/Google-Health/imaging-research/blob/master/ct-foundation/CT_Foundation_Demo.ipynb