Imaging-based Prediction of Malignant Transformation of Inverted Papilloma using Machine Learning

Diana Voevodsky
Stanford University
dinsuhin@stanford.edu

Andrew Hojel
Stanford University
ahojel@stanford.edu

Oscar O'Rahilly
Stanford University
oscarfco@stanford.edu

Abstract

Sinonasal inverted papilloma (IP) are typically benign tumors that can be identified through imaging. However, it has the potential to undergo malignant transformation to squamous cell carcinoma (IP-SCC), which is often challenging to identify preoperatively. In this study, we aim to develop a machine learning model to predict the malignant transformation of IP-SCC from IP based on magnetic resonance imaging (MRI). This retrospective review included 194 patients from two institutions. With positive pathology reports, they have a history of or were treated for IP. After performing image preprocessing and dataset augmentation, and experimenting with transfer learning, a 3D convolutional neural network (CNN) was trained to classify preoperative MRIs as IP (n=329) or IP-SCC (n=117) scans. Our best model achieves an overall accuracy of 77.9% and an ROC-AUC of 0.74, correctly classifying 66.7% of the true positive scans, and 81.5% of the true negative scans.

1. Introduction

Inverted papillomas (IP) are benign tumors that form in the posterior nasal cavities. Although IPs are benign, they have a 7-10% chance of converting to squamous cell carcinoma (IP-SCC), which are the IP's malignant counterparts. Even if the papilloma is successfully removed, there is a 15% chance of a recurrent tumor. It is currently difficult to detect the malignancy of IPs preoperatively, which can result in a massively different type of surgical resection. The ability to preoperatively diagnose IP-SCCs would allow for more strategic operative planning and improved patient counseling, as a cancerous tumor resection could require more extensive surgery, such as resection of the eye or brain, as well as therapies such as chemotherapy. In order to address this problem, we designed a 3D CNN which takes a patient's MRI as an input, and outputs an accurate prediction of the malignancy of their IP. We have obtained our data from Dr. Zara Patel's lab at Stanford Medicine, and from her partner lab at UPenn School of Medicine. This data consists of 446 total MRIs, of which 117 are IP-SCC and 329 are IP. Given these limiting numbers, we perform data augmentation prior to feeding the scans to our model in order to extract maximal information from each scan. Our 3D CNN consists of four convolutional layers, an adaptive average pooling layer, and a dropout layer. In addition, we separately perform transfer learning on DenseNet161 in an attempt to use its robust parameters to better extract information from our MRIs.

2. Related Work

The motivation and medical backing for this project is rooted in the paper *"Imaging predictors for malignant transformation of inverted papilloma: Imaging Predictors of IP Trans- formation"* [23], written by Dr. Zara Patel, the Director of Endoscopic Skull Base Surgery at Stanford, who advised us on this project. Published in 2018, this study outlines the process for using radiographic imaging as predictors for the malignant transformation of IPs. The two types of scans typically ordered for the target patients of this study are CT scans and MRI scans. In the paper, Dr. Patel outlines features that have been considered in the past when trying to identify a malignant transformation of an IP. Rather than attempt to extract these specific features, we instead use a deep learning approach.

Our baseline 3D CNN is heavily based on the model developed by Zunair, et al [7], used to predict the presence of tuberculosis given a patient's CT scan. We used Singh, et al [22] survey paper to explore various methods of increasing the model's accuracy, which discusses recent innovative approaches in the medical deep learning space. In particular, Singh, et al references several pre-processing techniques that we use such as normalization across the images, which was initially proposed by Ioffe, et al [9] and later built upon by Bjorck, et al [10]. In addition, Castro, et el propose the use of elastic deformations for augmenting images for breast cancer mass detection, which we applied to our neural MRI images in one of our models [4]. In general, these

techniques have proven to be very effective and are now widely used in the realm of Computer Vision.

Given the imbalance and limiting size of our dataset, the loss function used by our model was chosen carefully and methodically. Although similar deep learning tasks often use Binary Cross Entropy Loss [14], we needed to account for the heavy imbalance in the number of positive and negative scans in our dataset. Therefore, we use Weighted BCE loss, proposed by Yaoshiang, et al [24] in order to address this type of dataset breakdown. We also experimented with the use of Focal Loss, proposed by Lin, et al [16] specifically intended for our data conditions and often used in state-of-the-art medical image classification tasks. However, Focal Loss heavily unperformed with respect to Weighted BCE Loss in our case.

Our initial inspiration for transfer learning came from Kanghan, et al [13], who use it to detect the presence of Alzheimer's in brain MRI scans. The idea of using transfer learning to classify medical images is a relatively popular one since lack of data is often a barrier in such classification tasks. However, given the uniqueness and specificity of each such task, building off of previously learned weights and features has proven to be challenging. Recently, several studies have successfully used transfer learning to achieve impressive results in classifying MRI scans. For instance, Wacker, et al [11] build on ResNet34 [12] in order to perform brain tumor segmentation. Although we were able to achieve an overall accuracy of 70.90% and an ROC-AUC of score of 0.60 using transfer learning on DenseNet161 [5], it still underperformed with respect to our independent model, which achieved an overall accuracy of 77.9% and an ROC-AUC of score of 0.74.

3. Methods

3.1. Baseline Model

3D convolutional neural networks are able to extract both spatial and spectral features from volumetric data, which in our case consists of medical scans. Our 3D CNN takes in image stacks of dimension (128 x 128 x 64), and output a confidence percentage of the binary prediction.

We use four convolutional layers, each with kernels of dimension (3 x 3 x 3). The first two layers have 64 filters, the third has 128 and the final layer has 256 filters. In our model we use ReLU as our activation function. We experimented with using leaky ReLU; however, we determined that the model performed better when just using standard ReLU.

The outputs of the activation function are then processed through a Max pooling layer with dimension (2 x 2 x 2). We then apply batch normalization to standardize the outputs of each layer before they are inputted to the next. After our convolutional layers, our data is passed through an adaptive average pooling layer. This pooling layer helps to combat small shifts in the data. In our final steps we apply two dense layers, which are fully connected neural network layers. In between these two dense layers we apply dropout, with p = 0.3. Finally, we pass our data through a sigmoid function and use the ouput to make our predictions.

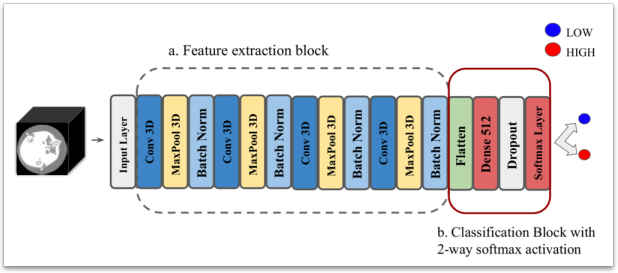

Figure 1: Model Architecture

3.2. Loss Function

Our baseline model uses Binary Cross-Entropy Loss. However, we replace it with Weighted Binary Cross-Entropy Loss (Weighted BCE Loss) in many of the others models due to the imbalanced nature of our dataset. Weighted BCE Loss very closely resembles conventional Binary Cross-Entropy loss, but has an added weighting factor $\alpha \in [0, 1]$ for class 1 and $1 - \alpha$ for class -1. We set α to be the inverse class frequency. The loss function can be written out as:

$$\text{CE}(p_t) = -\alpha_t \log(p_t)$$

where

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases}$$

Here, p is the model's estimated probability for the class with label $y = 1$.

One of our biggest challenges in developing a successful model is the small size and imbalanced nature of our dataset. A way in which we are attempting to combat this is through our choice of loss function. Weighted Focal loss is a loss function, based off the cross entropy loss, proposed by Tsung-Yi Lin, that is designed to address the scenario where there is an extreme imbalance between the classes of training images in a dataset [16].

The focal loss is defined as follows.

$$FL(p_t) = -(1 - p_t)^{\gamma} CE(p_t).$$

The factor that is modulating the original Cross entropy loss, $(1 - p_t)^{\gamma}$, is controlled by a tunable parameter, $\gamma \in [0, 5]$. When an example is misclassified and p_t is small, the modulating factor is near 1 and the loss is unaffected.

As $p_t \rightarrow 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted. The focusing parameter γ smoothly adjusts the rate at which easy examples are downweighted. Intuitively, the result of scaling the Cross Entropy Loss in this way down-weights the contribution of easy examples during training and focuses the model on hard examples.

Within our own model, we have implemented both our own version, and a built-in version from the `kornia` package of the Focal Loss with focusing parameter $\gamma = 2$; however, it is under-performing with respect to Weighted Binary Cross Entropy Loss.

3.3. Transfer Learning

Transfer learning is an exciting learning technique that leverages the use of large pretrained models to aid with a variety of different learning tasks. When it comes to computer vision classificiation, we can use state of the art image classification models, like ResNet and AlexNet, to aid with image classification tasks. What makes transfer training so powerful and widely used within the computer vision community is that we can use these pretrained models to classify images that they weren't even trained, so in our case MRIs.

When deciding how to integrate a pretrained model into our classification task, we had a few things to consider. Firstly, we had to decide which pretrained model to use. Secondly, we needed to determine what sort of architecture to add to the pretrained model in order for it to work with our specific classification task.

Pretrained Model: We settled on the DenseNet161 model, notably due to its exceptional performance to parameter ratio. DenseNet161 uses only around half the number of parameters as other state of the art models like ResNet, whilst simultaneously providing impressive classification accuracy. The result of using DenseNet161 over other models is improved computational efficiency and increased training speed.

Added Layers: Most of the pretrained models available for public use, such as DenseNet161 and ResNet, are all trained on 2-dimensional images. As our input data is 3-dimensional, passing through batches of MRI images into a the readily available pretrained models is not possible. To work around this, we experimented with three different architectures, leveraging the pretrained DenseNet161 model at its core, that would be make 3D classification using DenseNet161 possible.

As each MRI image consists of 64 (128x128) 2D images 'slices', our first design began with taking each of the 64 slices and passing them seperately through DenseNet161. We did not alter the architecture of DenseNet161, so the output of feeding in each slice was a (1 x 1000) vector. After we obtained all 64 outputs from DenseNet161, we concatenated all of them together and fed it into a seperate simple model consisting of a final linear layer that took the 64,000 long vector and produced a single output. This single output was then passed through a sigmoid function to produce a final classification. For backpropogation, we kept DenseNet161 frozen and backpropogated the loss only through the added simple model. Unfortunaely, however, this architecture was not able to learn.

For our next attempt, instead of feeding each slice separately into DenseNet161, we decided to add a layer to the start of DenseNet161 that would transform the image size from (1 x 3 x 64 x 128 x 128) to (64 x 3 x 128 x 128). This way, we did not have to define a seperate model and could simply replace the final classification layer of DenseNet161 to ouput 1 class score, rather than 1000. In our backpropogation, like before, we froze the DenseNet161 layers and simply performed backprop on our added first layer and classifier layer. Once again, however, this model was also not able to learn even after 5 epochs.

For our final iteration, we decided to go back to revisit our implementation of our first architecture: passing each slice through DenseNet161 and concatenating the 64 outputs. Instead of feeding this output through the same basic model as before, we decided to remove the final classification layer of DenseNet161 and replace it with a linear layer going from (1 x 2208) to (1 x 100). We then pass this output into a new model where we stack all outputs into of DenseNet161 into a (6400 x 1) vector, apply a ReLU activation function followed by a dropout layer (with $p = 0.3$) and then process it through a final linear and sigmoid layer (see figure 2). This more integrated architecture combined with a new non-linearity layer proved successful as we finally saw signs of learning.

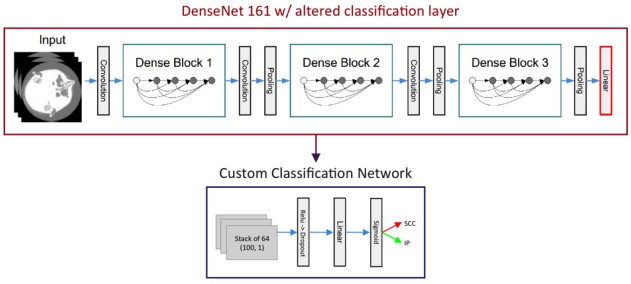

Figure 2: Transfer Training Model Architecture

4. Dataset and Features

Our dataset consists of anonymized medical scans in the form of dcm files. Each folder contains a series of images representing the scan at a specific time step. When sequentially stacked, these images make up the full 3D scan. Currently, we have roughly 446 total scans, of which 117 are IP-SCC+ and 329 are IP-SCC-.

Set	# Scans	IP-SCC-	IP-SCC+
Train	251	190	61
Validation	109	74	35
Test	86	65	21

Table 1: Dataset Breakdown

The stack size and image dimensions vary per scan, so we perform several pre-processing steps in order to obtain uniform data that can be fed to the model. First, we normalize the volume and then standardize the number of slices of each scan. In order to resize the image and extrapolate along the z-axis, we use the zoom function built into the `scipy` package. Setting our desired slice depth to be 64, and our desired width and height to be 128 pixels, we pass the calculated depth, width, and height factors for each scan to the scipy zoom function, which then returns the resized image.

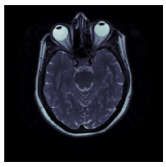

Figure 3: Example MRI Scan Slice

The full data set contains scans from various angles and orientations. We therefore manually sorted through each scan and filtered for those with a vertical orientation (exhibited above) since those outnumbered the rest.

5. Experiments and Results

5.1. Metric

Final reported metrics for each of our models are taken from evaluating the test set. We evaluate our model on two metrics:

- **Accuracy**: We calculate positive, negative, and general accuracy. Positive accuracy indicates the fraction of IP-SCC+ scans that are correctly classified by our model. Negative accuracy indicates the fraction of IP-SCC- scans that are correctly classified by our model. General accuracy indicates the fraction of all scans that are correctly classified by our model.
- **ROC-AUC**: This metric provides an aggregate measure of performance across all possible classification thresholds and is often used in medical image classification tasks. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. The closer ROC-AUC is to 1, the better the model is at distinguishing between classes. When the ROC-AUC score is 0.5, it indicates that the model cannot distinguish between the positive and negative class (either randomly guessing each class or predicting one class for all the data points).

5.2. Hyperparameters

Our learning rate for all the models was 1e-3 (we attempted tuning this, but the results were largely unaffected). We used ADAM optimizer for all of our models but experimented with RMSProp with transfer learning because ADAM optimized too quickly and would fluctuate later in learning, so we hoped RMSProp would provide a smoother decrease of loss. Yet, final model performance was largely unaffected.

For all of our models using our custom architecture we used a batch size of 4 (limited by GPU memory) except for All-Net-Elastic which used a batch size of 8 (because the data augmentation used less memory). All of the transfer trained models used a batch size of 1 because of memory limitations.

In addition to this, we attempted to use Weights & Biases hyperparameter sweep to tune our models, but struggled with Google Collab timing out in the middle of sweeps.

5.3. Model Breakdown

5.3.1 Baseline Model

The baseline model for this project consists of using our CNN architecture explained in the technical approach section using Binary Cross Entropy Loss. The baseline model is trained on preprocessed MRI scans from only our Stanford dataset, which consists of only 222 scans (155 training and 67 validation).

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5

Table 2: Baseline Results

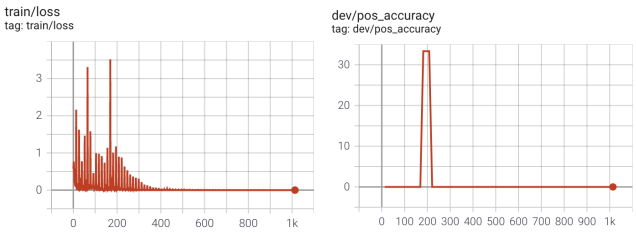

Figure 4: Baseline Positive Accuracy

5.3.2 Weighted BCE & Data Augmentation

Our first experiment was switching the loss function to Weighted Binary Cross Entropy loss given the imblanaced nature of our dataset (significantly more IP-SCC- scans than IP-SCC+ scans). In addition, we added data augmentation because of the small size of our original training set. We added three different types of data augmentation: random rotations, left-right flip, and random noise. For random rotations, we choose an rotate between -20 degrees and positive 20 degrees randomly. For left-right flip, we flipped the image across the x-axis. For random noise we added noise from a Gaussian distribution with mean of 0 and standard deviation of 0.2. Using the data augmentation increased the size of our training set by four times resulting in 620 training scans (in the baseline we had 155 training scans).

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
Weighted-Aug	222	4.76 %	96.92%	74.42 %	0.51

Table 3: Weighted-Aug Results

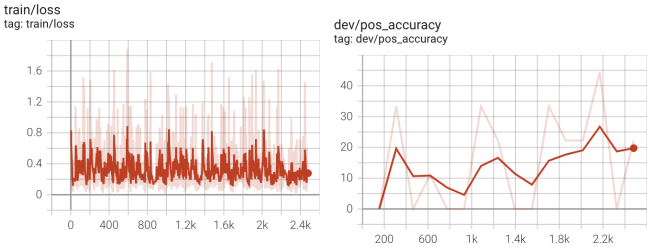

Figure 5: Weighted-Aug Positive Accuracy

5.3.3 Focal Loss

Our next experiment was implementing focal loss, as described in the *Methods* section above. In addition, we were able to collect more data from our research partners at UPenn, which increased our training and validation sets to 360 total scans. Nevertheless, replacing Weighted BCE with focal loss lead to significantly worse results (as displayed in the Table 9). The performance of the model did not change when we moved from our implementation of focal loss to the implementation from the `kornia` package.

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
Focal-Net	360	0%	100%	75.58%	0.5

Table 4: Focal-Net Results

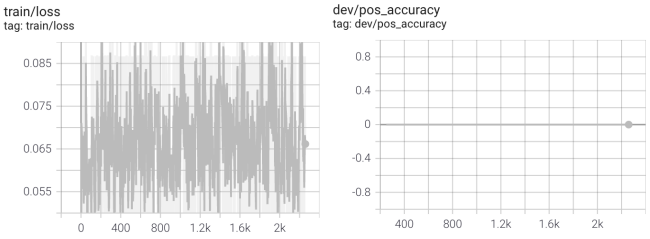

Figure 6: Focal-Net Positive Accuracy

5.3.4 Additional Data & Weighted BCE & Data Augmentation (All-Net)

In this experiment, we re-ran the *Weighted-Aug* Model, but with our new UPenn Data incorporated. This is our best performing model.

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
All-Net	360	66.7%	81.5%	77.9%	0.74

Table 5: All-Net Results

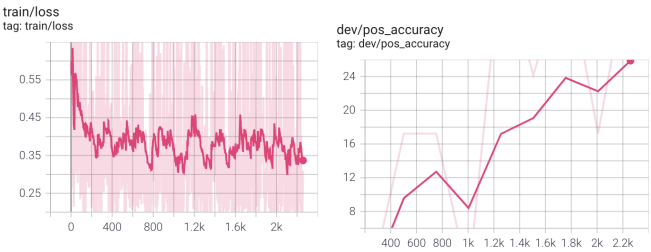

Figure 7: All-Net Positive Accuracy

5.3.5 All-Net Double Dimensions

In an attempt to extract more information from each scan, we doubled the number of filters for all the convolutional layers and doubled the size of all the linear layers in the All-Net model.

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
Large All-Net	360	38.1%	78.5%	68.6%	0.58

Table 6: Large All-Net Results

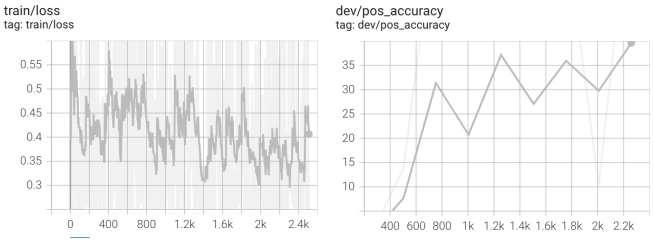

Figure 8: Large All-Net Positive Accuracy

5.3.6 All-Net Half Dimensions

Similar to the above model, this model is the same as All-Net, except each dimension is halved (in the same manner as All-Net Double Dimensions). This model achieved the best positive accuracy (95%), but not the best negative or overall accuracy. Yet, the ROC-AUC value was extremely close to the ROC-AUC value of All-Net.

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
Small All-Net	360	95.24%	50.77%	61.63%	0.73

Table 7: Small All-Net Results

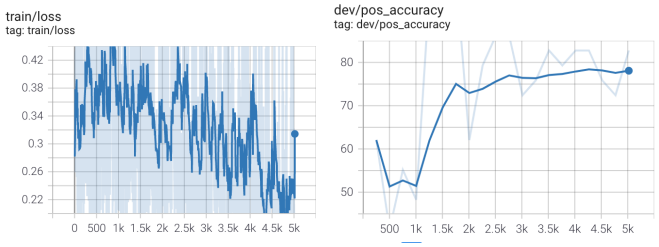

Figure 9: Small All-Net Positive Accuracy

5.3.7 All-Net-Elastic

In this model, we implemented a new data augmentation method using Random Elastic Deformation, which is taken from the `torchio` package and inspired by Castro, et al [4]. In addition, we increased the batch size from 4 to 8. We were not able to increase the size past 8 given GPU memory limitations.

5.3.8 Transfer Learning

In order to compensate for our limited data, we experimented with implementing transfer learning, as described in the *Methods* section. We modified the final linear layer and added a small classifier network onto DenseNet161 in order to produce a binary class prediction. Although this implementation was successful, it was still outperformed by All-Net.

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
All-Net-Elastic	360	80.95%	69.23%	72.09%	0.75

Table 8: All-Net-Elastic Results

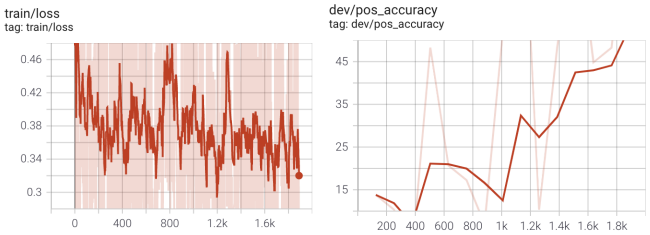

Figure 10: All-Net-Elastic Positive Accuracy

Model	# Scans	Pos-Acc	Neg-Acc	Overall-Acc	ROC-AUC
Baseline	222	0%	100%	75.58%	0.5
Weighted-Aug	360	38.1%	81.5%	70.9%	0.6

Table 9: Transfer Learning Results

6. Discussion

Ultimately, our three best performing models were All-Net, Small All-Net, and All-Net-Elastic. All-Net had the best overall accuracy of 77.9% and nearly the best ROC-AUC with 0.74. Small All-Net had the best positive accuracy of 95.24% and maintained an ROC-AUC of 0.73. All-Net-Elastic had the best ROC-AUC of 0.75 and maintained an overall accuracy of 72%. Each model has varying strengths which could each be useful in practice.

6.1. Baseline

This experiment exposed the initial challenges that our classification task faced given the small and imbalanced nature of our dataset. With conventional BCE Loss and no data augmentation, our model was not able to achieve any positive accuracy. In other words, in order to minimize its loss, it classified every scan as negative.

6.2. Weighted BCE & Data Augmentation

Implementing Weighted BCE and adding the data augmentation described above increased our positive accuracy to 4.75%. Although this is only a small improvement over

the Baseline model, it demonstrates that data augmentation and Weighted BCE Loss allowed the model to begin learning (evidenced by correctly classifying a positive scan). Yet, it is clear that this model would be unable to achieve acceptable results.

6.3. Focal Loss

Despite the overwhelmingly positive evidence in support of focal loss, presented by Tsung-Yi Lin [16], as a solution for severely imbalanced datasets, we were not able to achieve any positive classification accuracy. We attempted using both our own implementation of focal loss and the version implemented in the `kornia` package in addition with significant tuning of the α and γ hyperparameters. We were unable to pinpoint the reason for focal loss's poor performance, thus it merits further investigation and experimentation. As a result, for all the future models we decided to continue using Weighted BCE as our loss function.

6.4. All-Net

As was mentioned in *Results*, All-Net was our best performing model, with an overall classification accuracy of 77.9%. Whilst this may not sound like a substantial increase in accuracy from our original Weighted-Aug model (with an accuracy of 74.42%), when we look at the breakdown of positive and negative classification accuracies, we see an extreme improvement in the model's positive classification ability, from 4.76% to 66.7%. In fact, All-Net received nearly the highest ROC-AUC of 0.74, displaying its ability at distinguishing between the positive and negative classes. This is extremely important when considering we are working on a medical classification task, and a reliable positive classification rate is crucial for any sort of real-world deployment. The addition of new data in conjunction with data augmentation and Weighted BCE loss (which demonstrated slight learning in Weighted-Aug) significantly improved performance on positive scans.

When analyzing the saliency map produce by All-Net for the SCC scan, we notice that there is the most gradient flow occurs in between and on the two sinonasal cavities (where IP and SCC occur). In addition, there is more gradient flow on the side of the brain with the tumor. It is clear that the model has learned to search the proper areas.

6.5. Large All-Net

Our experimentation with Large All-Net was an attempt to determine if we could extract more information and increase our model's expressivity. Despite doubling the number of filters for all the convolutional layers and doubling the size of all linear layers, Large All-Net returned somewhat underwhelming results, with an overall accuracy of 68.6%. This is over a 9% dip in accuracy when compared to the 'simpler' All-Net variant. Furthermore, the positive

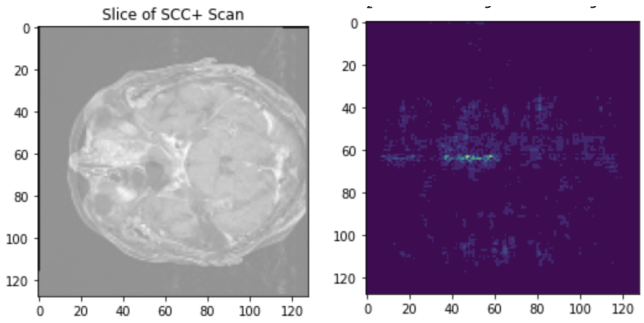

Figure 11: All-Net Saliency Map

accuracy saw an even steeper decline of nearly 30% (from 66.7% to 38.1%). We hypothesize two reasons for the substantial difference in accuracies. The first is perhaps a case of overfitting. Unfortunately, we did not record the training accuracy for this particular model, but it makes sense that doubling the number of filter in the convolutional layers and the size of all linear layers could cause the model to overfit to the training inputs. The second reason is that our small dataset does not provide enough information for a larger model to learn well, thus the increase in size would be unwarranted without the addition of more data.

6.6. Small All-Net

Small All-Net provided perhaps some of our most interesting results. Its overall classification accuracy is by far the lowest out of all our models. At 61.6%, Small All-Net's overall accuracy is lower than any other model by nearly 7% (Large All-Net), and nearly 14% lower than our baseline model. However, if we turn our attention to the positive accuracy of 95.24%, we see that Small All-Net significantly outperformed every other model. Furthermore, its ROC-AUC score (0.73) is one of the best out of all the networks, only underperforming All-Net by 0.01. Returning to the significance of real-world applicability, the small All-Net architecture shows great promise.

Turning our attention to the saliency map displayed below (figure 12), we can clearly see a region of high intensity towards the left of the image - the areas in which the tumor is located. This is very encouraging as it reassures us that the model is identifying the relevant regions when making its classification. That being said, there is also a general medium-level intensity in the whole image, even in regions outside the actual brain scan. This makes sense due to the fact the Small All-Net model contains the fewest number of convolutional filters and linear layer sizes, meaning it is not able to capture as much information as the other All-Net models. This increased gradient flow may also explain the lower negative sample accuracy (caused by more false

positive classifications).

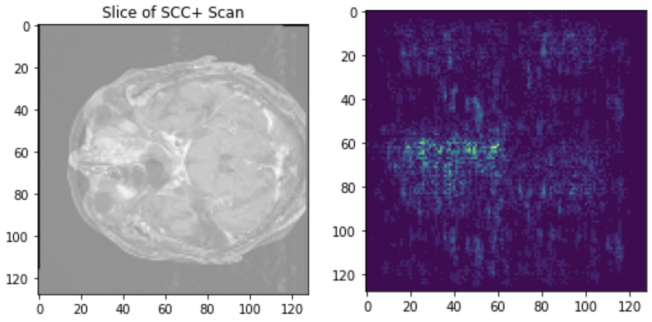

Figure 12: Small All-Net Saliency Map

6.7. All-Net-Elastic

All-Net-Elastic was another of our better performing models. In fact, in terms of its ability to correctly distinguish between positive and negative examples, it was our best performer. This was confirmed by its ROC-AUC score of 0.75, the highest out of all models. The positive accuracy was the second highest, only beaten by Small All-Net, at 80.95%. Its Saliency map was also very positive, with a distinct intense region along the sinonasal region where the tumor is present, and extremely minimal activity along the boundary of the MRI slice not part of the actual brain scan. This model has a good balance of high gradient flow in the region where the tumors occur without as much gradient in other regions, which helps explains its good performance and high ROC-AUC value.

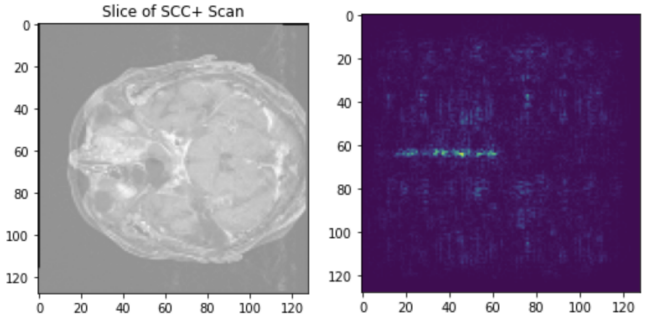

Figure 13: All-Net-Elastic Saliency Map

6.8. Transfer Learning

Interestingly, the results obtained through our transfer learning techniques, underperformed compared with our other successful CNN models; however, it was certainly not disappointing. Its negative accuracy of 81.5% was amongst the highest of any models. Although, the positive accuracy of 38.1% was amongst some of the lowest. In a medical context, where positive classification accuracy is most important, our transfer learning model is far off from a deployable, real-word solution. One reason we suspect this to be the case is due to the type of input data we are passing into DenseNet161 for feature extraction. While transfer training is certainly able to classify images not seen in its training set, perhaps the format of the MRI scans we used were not preprocessed in such a way that lent itself to be successfully used by DenseNet161. Another reason is perhaps in the simplicity of our final appended classification section. In future iterations it would be interesting to experiment with denser final layers to see the affects on performance. With regards to our two Transfer architectures that were not able to learn, we suspect that it may be in part due to the lack of any non-linearity in the final classification layer. Our classification problem may certainly not have a linear decision boundary and so the lack of non-linearities may prohibit any sort of successful learning.

7. Conclusion

Throughout this paper we have explored multiple different approaches and variations at classifying IP and SCC tumors. The poor performance / lack of learning of the early models demonstrated how large of a role the imbalanced nature of the dataset and the small size of the dataset would play in building the best possible architecture. With the addition of more data, various augmentation methods, and better loss functions we achieved much better performance. Interestingly, our top models each had unique benefits / downsides. With All-Net we see the highest accuracy (77.8%), but other models had significantly higher positive accuracy than 66.7%. Small All-Net had fantastic positive accuracy (95.24%) but significantly worse negative accuracy (50.77%). Finally, All-Net-Elastic fell somewhere in the middle with positive (80.95%) and negative (69.23%) accuracies in between the other top performing model and had the best ROC-AUC (0.75). Choosing the best model out of these three is difficult because it depends on what we are optimizing for (between aggressive positive classifications or high overall accuracy). Moving forward, we are very excited about continuing to build out our custom model (specifically experiment with model ensembles of our top models). Yet, we believe that transfer training poses the highest potential for gain with better image preprocessing and attempting to use different networks.

8. Contributions

This project was completed with the help of Dr. Zara Patel, the Director of Endoscopic Skull Base Surgery at Stanford Hospital, who provided us with her current research

in the field concerning inverted papillomas and squamous cell carcinoma, as well as advised the project throughout the course of the quarter. In addition, we were fortunate enough to work with two members of her research team, George Liu and Angela Yang, who helped us acquire and better understand the data. In terms of work breakdown, Dina and Andrew focused on building the custom model architecture and experimenting with various additions / changes. Oscar took the lead on investigating and implementing the transfer training models.

9. Acknowledgements

We based our custom model off of a model implemented in a Keras tutorial titled 3D Image Classification from CT Scans. Yet, it is worth noting that we did not use any code from it given that we used PyTorch and the provided implementation used Keras.

We based our own implementation of Focal Loss off of code found in a GitHub article titled What is Focal Loss and when should you use it?.

In addition to this, we used `kornia` [21] for a different implementation of focal loss and `torchio` [20] for an implementation of random elastic deformation that was used in All-Net-Elastic. Other relevant packages that we used are the following: PyTorch [18], pydicom [17], scikit-learn [19], DenseNet161 [8], numpy [6], OpenCV [3], Tensorboard [1], Weights and Biases [2], and CS231N Course Notes / Lecture Matieral [15].

10. Appendices

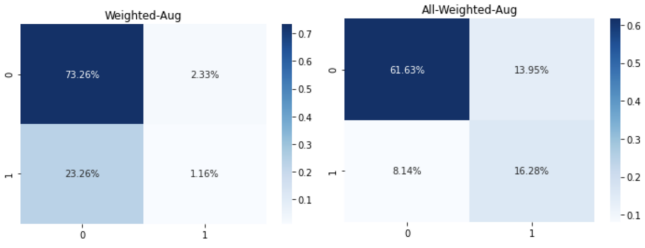

Figure 14: Weighted-Aug and All-Net Confusion Matrices

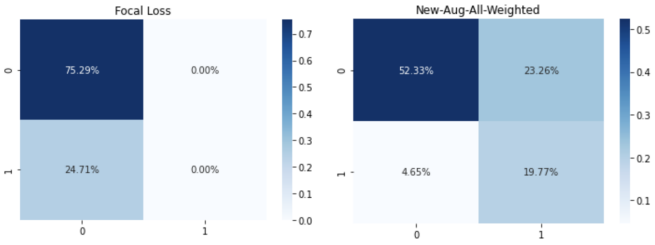

Figure 15: Focal Loss and Elastic-All-Net Confusion Matrices

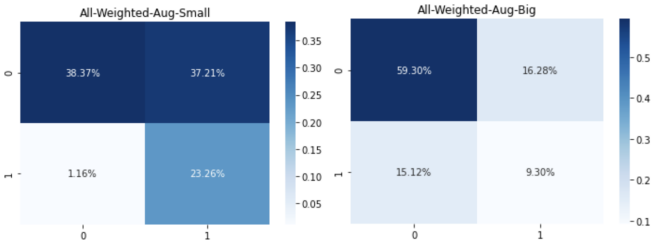

Figure 16: Small All-Net and Big All-Net Confusion Matrices

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 9

[2] L. Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 9

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 9

[4] E. Castro, J. S. Cardoso, and J. C. Pereira. Elastic deformations for data augmentation in breast cancer mass detection. In *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 230–234, 2018. 1, 6

[5] L. v. d. M. K. Q. W. Gao Huang, Zhuang Liu. Densely connected convolutional networks, 2018. 2

[6] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 9

[7] N. M. Hasib Zunair, Aimon Rahman and J. Cohen. Uniformizing techniques to process ct scans with 3d cnns for tuberculosis prediction. 1

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017. 9

[9] C. Ioffe, S.; Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 1

[10] B. S. K. Q. W. Johan Bjorck, Carla Gomes. Understanding batch normalization, 2018. 1

[11] M. L. Jonas Wacker and J. E. V. Nascimento. Transfer learning for brain tumor segmentation, 2020. 2

[12] S. R. J. S. Kaiming He, Xiangyu Zhang. Deep residual learning for image recognition, 2015. 2

[13] K. W. K. W.-S. K. a. I.-S. Kanghan Oh, Young-Chul Chung. Classification and visualization of alzheimer's disease using volumetric convolutional neural network and transfer learning, 2019. 2

[14] W. M. C. Katarzyna Janocha. On loss functions for deep neural networks in classification, 2017. 2

[15] F.-F. Li, A. Karpathy, and J. Johnson. Cs231n: Convolutional neural networks for visual recognition 2016. 9

[16] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection, 2018. 2, 7

[17] D. Mason. Su-e-t-33: pydicom: an open source dicom library. *Medical Physics*, 38(6Part10):3493–3493, 2011. 9

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 9

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 9

[20] F. Pérez-García, R. Sparks, and S. Ourselin. TorchIO: a Python library for efficient loading, preprocessing, augmentation and patch-based sampling of medical images in deep learning. *arXiv:2003.04696 [cs, eess, stat]*, Mar. 2020. arXiv: 2003.04696. 9

[21] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. R. Bradski. Kornia: an open source differentiable computer vision library for pytorch. *CoRR*, abs/1910.02190, 2019. 9

[22] S. G. H. G. P.-s. P. Satya Singh, Lipo Wang and B. Gulyas. 3d deep learning on medical images: A review. 2020. 1

[23] C. Yan, C. Tong, M. Penta, V. Patel, J. Palmer, N. Adappa, J. Nayak, P. Hwang, and Z. Patel. Imaging predictors for malignant transformation of inverted papilloma: Imaging predictors of ip transformation. *The Laryngoscope*, 129, 12 2018. 1

[24] S. W. Yaoshiang Ho. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling, 2019. 2