Breast Cancer Segmentation: A Deep Learning Approach

Han Lin Aung
Department of Computer Science
Stanford University
Stanford, CA, USA
hanlaung@stanford.edu

Hugo Valdivia
Department of Computer Science
Stanford University
Stanford, CA, USA
hugov65@stanford.edu

Sajana Weerawardhena
Department of Computer Science
Stanford University
Stanford, CA, USA
sajana@stanford.edu

Abstract—**Mammograms have been widely used as a first step in detecting whether a patient has breast cancer. However, to precisely pinpoint the location of the tumor, doctors still have been manually labeling the data themselves. Instead, we propose an automated segmentation of the breast cancer tumor on mammograms. We experimented with a variety of machine learning techniques, such as logistic regression, while focusing on deep learning approaches. We focused on implementing and expanding upon well-known deep learning architectures, such as Mask-RCNN and UNet that have shown impressive results from image segmentation tasks, especially on medical images.**

I. Introduction

Breast cancer is one of the leading causes of death and the most common cancer for women worldwide; in 2018, there were more than 2 million new diagnoses [1]. In the medical field, there exist quite a few technologies for diagnosis and analysis of data from mammograms and MRI. Inspecting tumors for mammogram, however, has been a highly manual task for radiologists. In this project, we experiment and implement different machine learning techniques to tackle this problem. in particular, given an input mammography image, we use logistic regression, and a plethora of deep learning architectures to produce a mask image over the tumors present in that image.

II. Related Work

In hospitals, a lot of the mammogram data on the precise location and area of the tumor have been labeled by trained professionals, specifically radiologists. However, recently, there have been multiple approaches in the space of automatic image segmentation on breast cancer tumors. There have been more traditional computer vision based methods such as the use of edge-based segmentation techniques. However, they are limited by blurring and are highly sensitive to noise and intensity values (difference) in data. More notably, there have been quite a few impressive results through deep learning approaches in image segmentation. Mask-RCNN, an object instance segmentation model, detects the object in the image while producing a mask of the image, has been considered as a highly generalizable network architecture [2]. In the medical data space itself, there have been a rising number of architectures that have shown promising results. UNet, in particular, has shown state-of-the-art results for segmentation tasks in the space of segmentation of medical imaging data [3]. [4] develops a fully convolutional neural network based on the UNet along with very careful image preprocessing and augmentation and is able to produce impressive results on automatic detection of brain tumor.

More specifically in the space of breast cancer tumor segmentation using mammograms, [5] uses generative adversarial networks (GANs) to produce realistic binary masks to segment the breast tumor, which claim to have an overall accuracy of 80 percent. [6] takes another approach on detection on mammogram images by doing transfer learning on well-known deep learning architectures such as AlexNet and GoogleNet which have yielded impressive results in other imaging tasks, namely classification. [6] claims to have outperformed trained radiologist based on the GoogLeNet architecture without extensive use of features from domain knowledge.

III. Dataset and Features

We've drawn data from a variety of sources.

Our first source of data is the MIAS MiniMammographic Database [7]. This is a collection of 322 mammograms, each of dimension 1024 x 1024. Of the 322, some 119 images had either a malignant or benign tumor. For each of these images, we have an x, y, radius triple that denotes the center of the tumor and the radius of the tumor. From this data, we've generated mask images that highlight the regions of interest, in order to create an end-to-end segmentation task.

Our second source of data is the CBIS-DDSM dataset [8], which features 2,620 scanned film mammography studies with over 10,000 dicom images. The dataset consists of images with abnormalities along with the masks of the location of the tumor. However, the images do not have a standardized size, so we preprocess the dataset to standardize the images to be 1024 x 1024 pixels before feeding into our models.

For training our models, we create a 60-20-20 percent split on train, validation and test sets respectively.

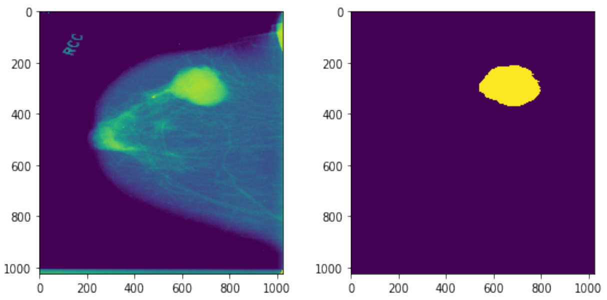

Fig. 1. Image and corresponding mask from the DDSM dataset

IV. Methods

A. Logistic Regression

We implemented logistic regression, which is a linear classifier using the sigmoid function. We use logistic regression to classify each pixel in an image as 0 (non-tumor) or 1 (tumor). As a linear classifier, it is less powerful than deep learning models, but it serves as baseline model for our approaches.

B. Feed-forward neural network

We can think of the logistic regression algorithm equivalently as training a single layer shallow feed-forward network with a single sigmoid output; however, the hypothesis class grows as we extend the architecture with additional layers and different activation functions. For each layer l, this neural network is governed by the following equations:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

where $W^{[l]}$, $b^{[l]}$, and $g^{[l]}$ denote the weight matrix, bias, and activation function at that layer, respectively, and $a^{[0]}$ is our input image x.

In order to train such a network, we need to define a loss function that quantifies how unhappy we are with our predictions. For our particular problem, the natural first choice was that of binary cross entropy loss, defined as:

$$CE(y, \hat{y}) = \sum_{i=1}^{M} y_i \log(\hat{y}_i)$$

where $M = 2$ in this case. By computing the gradients with respect to the parameters in the backpropagation algorithm, a network is then able to update its weights and biases to improve its predictions, and this framework also applies to the following neural network models.

C. Shallow convolutional neural network

The shallow convolutional neural network we implemented consists of a sequence of Convolutional layers, max pool layers, droput, and dense layers along with ReLU and softmax (the last layer) as our activation functions.

Every network layer in the convolutional neural network (CNN) can be seen as a detector of a certain set of features, with the initial layers detecting coarser and larger features while the later layers refining on top of the originally detected features. Shallow convolutional neural networks are less computationally expensive to train compared to deep convolutional neural networks (with more hidden layers) and produce fairly reasonable results but they tend not to be the state-of-the-art models. In CNN, filters, usually smaller than the input space, are weights represented as matrices that will be applied over the inputs at each layer. The values of the filters will be learned during the training.

The convolution layer is used to extract the initial high level features from an image and is thus the initial layer in our model. The Max pooling layer reduces the dimensionality of the parameters and computing power by taking the maximum element over a grid across the input. Dropout has been a popular technique to reduce overfitting by zeroing out certain input values with some probability p.

The ReLU non-linearity activation function $max(0, x)$ provides a reduced likelihood of vanishing gradients during the training of the model. The softmax activation function is implemented in the output layer: $\sigma(z)_j = \frac{e^{z_j}}{\sum_{i=1}^{K} e^{z_k}}$ where K=2 represents the number of classes. The softmax is an appropriate function for our classification task since it will then produce a probability for each of the two classes (tumor vs non-tumor).

D. UNet

UNet is one of the more recent convolutional neural network models that has had great success in the image segmentation space, especially with medical data. The UNet consists of convolution, max pooling, ReLU activations, and up-sampling layers. The up-sampling layer uses transposed convolution filter which associates each input value with a grid of other values (intuitively the backwards of a convolution). The UNet first downsamples the image and then later up-samples the feature maps (inputs to the layer). The number of feature maps, which are the features extracted by applying the filters, will increase throughout the downsampling process. This will help capture the context of the input image and provide coarser features. Afterwards, these feature maps, with the context features of the input image, will then continue on the process of up-sampling which will then provide information on more precise localization to get an accurate binary mask of our data.

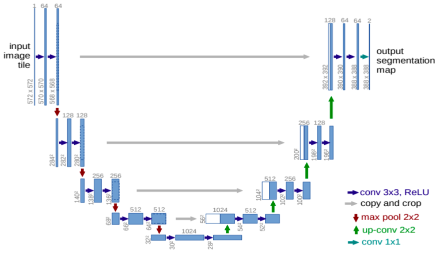

Fig. 2. Original UNet architecture

E. Patch classifier: Transfer learning

We also explored transfer learning by looking into models with pretrained weights. Specifically, we used a model that is pretrained on classifying whether a certain patch of an image is a benign calcification, malignant calcification, benign tumor, malignant tumor or simply the background. The pretrained model has been trained on patches of whole images. We attempted with a variety of patch classifiers trained on similar datasets, but primarily focused on the Yaroslav Net patch classifier that has been trained on similar mammogram images and has shown one of the best results in the classification task [9]. The YaroslavNet consists of max-pooling, fully connected layers and shortcut conenctions, which are connections that skip one or more layers. Shortcut-connections alleviate the degradation problem of reduction in accuracy in deeper layers of the network after the model reaches a local minima. We were hoping to use the initial weights from this patch classifier to fine-tune the model so that we can reach convergence faster with our CBIS-DDSM/MIAS dataset.

More specifically, for our architecture, the last layer was removed and we added two more Dense layers to predict whether each pixel in the whole image will be considered as a tumor or non-tumor.

F. Mask R-CNN: Transfer learning

While we applied the above CNN architectures to the task of semantic segmentation, the Mask R-CNN [10] architecture tackles the instance segmentation problem; that is, given an input image, we wish to find all of the instances of a particular class. It boils down to first performing object detection to create bounding boxes for each object and then performing semantic segmentation on each of these bounding boxes [11]. Now, the Mask R-CNN architecture can be broken down into two parts. First, there is a backbone architecture (e.g., VGG16, ResNet/NeXt) to extract features and a network head for the actual instance segmentation task [10].

To train the model, we first loaded in the weights of the model that is pretrained on Mask R-CNN from the COCO dataset, which is a large scale object detection and segmentation dataset. We also experimented with the weights from ImageNet, a sizable natural images dataset. We then retrained the model with the mammogram dataset using the pretrained weights so that the model has already learned some initial features and that convergence will be faster.

V. Experiments/Results/Discussion

A. Preliminary Experiments

Using only the MIAS dataset, we tried to use the raw data: the x, y, radius triples given. Because these were the output labels we had to predict, we framed the task as a regression problem in which we had to predict these three separate real numbers. With this in mind, we tried a Feed-Forward NN architecture with 5 layers of (128, 64, 32, 16, 3) hidden units respectively, where at each stage we used the ReLU nonlinearity. We tried to use squared mean error, absolute error, and the logcosh loss functions; however, after training, we used our model to predict our validation set and dug through the predictions. We found that the model outputs [0.0, 0.0, 0.0] for our entire validation set with each of these choices. After attempting to change the optimizer (Adam, RMSProp, SGD), learning rates (by powers of 10 away from the recommended Keras default values), and increasing the number of layers, our hypothesis is that this is because our architecture was not fit to handle these large images and could not learn the relationship between our data and the output. After all, we were expecting a feed forward network to learn solely from a flattened list of over a million features (1024 x 1024 pixels); this experiment thus motivated our following approach.

B. Semantic Segmentation

With the preliminary results in mind, we aimed to train several convolutional neural networks for our task. Since convolutional neural networks have been able to produce state-of-the-art results for imaging data, we have decided to move onto this approach. For this, we built masks for the MIAS dataset and made use of the CBIS-DDSM dataset. Our masks were the same size as the image and were composed of 1s at the pixels that were part of tumors and 0s at the other pixels. With a pixel-wise binary cross-entropy as our loss function, we hoped that CNN architectures could take in our input image and learn to produce the corresponding mask. While binary cross entropy was our primary loss, we judged our model on two other metrics: epoch to epoch, we looked at the mean Intersection Over Union, and we looked at our mask predictions on a subset of our validation set.

We started with a small CNN architecture that we designed and trained from scratch. (5 by 5 kernel, 16 filters, 5 by 5 kernel, 8 filters, Max pool 50x50, Dropout 0.25, 5 by 5 kernel, 16 filters, 2 dense layers and a Softmax). However even with an extensive hyper-parameter search, we could not get our model to predict anything other than 0s. Our instinct was that the model was not deep enough to effectively map the input to the mask. Hence, we decided to use a UNet - a model that has been shown to be very effective in brain tumor segmentation. We used a standard UNET with no extra dense layers: this choice was largely because our image output size was 1024 by 1024 and hence would require training a huge hidden layer. Once again, even with an extensive hyper-parameter search, we could not get our model to predict anything other than 0s.

C. Modified Loss Function

In response to the local minimum convergence, we decide to use a weighted binary (pixel-wise) cross entropy function. We realized that because the images were 1024 by 1024 and the average tumor was close less than 100 pixels (in radius) in our dataset, by weighting incorrect predictions on tumor pixels high, and vice-versa for non tumor pixels, we would encourage our model to predict the tumor class more. We experimented with a range of weight values. Our initial instincts were to use 1:10 or 1:100 weights for the two classes but these did little to impact our bottom line. The metric we utilized to judge

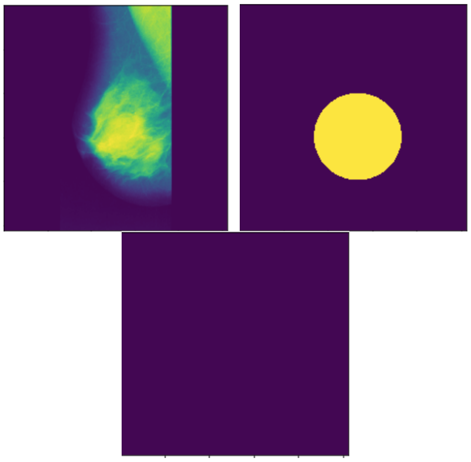

Fig. 3. Example of a Mammogram, Mask and our prediction

our models was, as mentioned before, mean Intersection over Union; however, IOU is a poor indicator unfortunately of how our model performs as it weights both classes equally. Still, a high IOU was a strong indicator for us during training that our model had tended towards predicting no tumor on every pixel, because a 100% IOU on non tumor skewed the mean. The graph (Fig. 4) shows how the mean IOU increased to different weights (1:10, 1:1 million, 1:1.5 million). Note in particular how the mean IOU for 1:1 million (as seen at 1 epoch) and 1:1.5 million (as seen at 0.5 epoch) kept the IOU from tending towards 95% and above more than the 1:10 weighting but still failed to ultimately stop our model from converging to the predicting all 0s (after each epoch we predicted our model on the train data again to see if predicts the correct mask; they were all 0s).

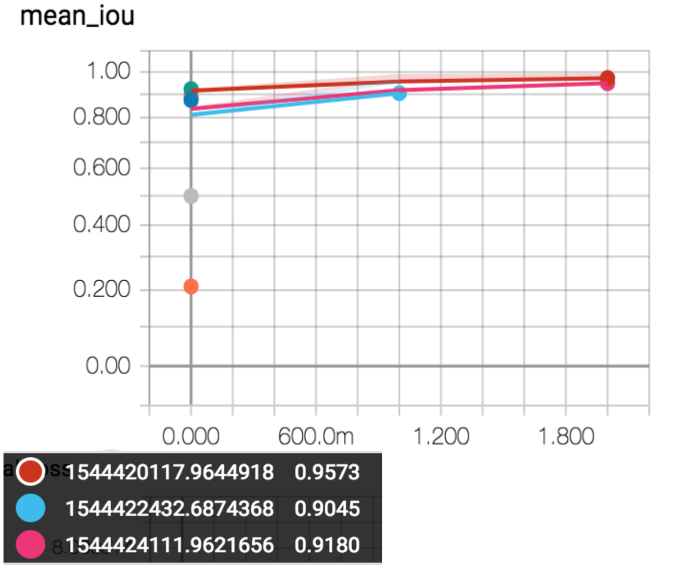

Fig. 4. Mean IOU; Key: Red- 1:10 weighting, Pink:1:1000000, Blue: 1:1500000

D. Data Augmentation and Sampling

Following this, we decided to oversample tumor data into our training set. We did this by first identifying tumors that above average large masks and then we sampled more of them into our training set. In the process of doing so, we also augmented the data (rotational), using Keras' ImageDataGenerator class [12]. Our hope was that this would help offset the imbalance in the data and help our models perform better. This approach too failed to produce models that predicted any mask at all. We responded by just isolating the larger tumor data that had more round/circular masks, and trying to over fit our model on this train data. The model once again converged to not predicting a mask. Our hypothesis is this is because even the largest tumors were less than 200 px radius (the maximum was 198 px).

Following the failure of our UNet to over fit just the large tumor data, we instinctively tried to add dense layers at the end of the UNet. We thought that having more dense layers would improve the bias of our model. However given our output images were 1024 by 1024 grid, the final dense layer had to have 1024 x 1024 hidden units, which required far too many parameters to tune for our computational resources.

E. Data vs the Model

We then started to look into what the issue may be, whether it is our implementation of the model or the data itself. Hence, we started our debugging process by giving it a separate imaging task of predicting the edges around the whole breast instead of the tumor. The models that we have were able to accomplish this task with considerable accuracy (this was done in a more of a qualitative manner).

Hence, we then decided to rule out our implementation of the model and then moved into looking into the data itself. We randomly sampled about 100 images to make sure that the data from the mmamogram image corresponds to that of the mask. We could do this only more on a surface level manner since the tumor location does not necessarily depend on the intensity or difference in intensity of the pixel values, and thus requires a trained professional to really determine the validity of the data.

Given the issue we have, we then hypothesized that the the issue may be due to the insufficient amount of data to learn such a task. To further debug the issue, we decided to try the approach of transfer learning. We used a model with pretrained weights of a patch classifier that was also trained on similar mammogram images (such as the InBreast dataset). We attempted different patch classifiers, but primarily tested using the YaroslavNet [9]. However, after feeding in our data, the model still converged to a local minima of not predicting a mask. Hence, we decided to move onto trying a similar approach with transfer learning but on a different architecture that has previously shown impressive results in segmentation tasks.

F. R-CNN

The Mask R-CNN architecture gave us our best prediction; one such prediction on the validation set is seen in the figure below.

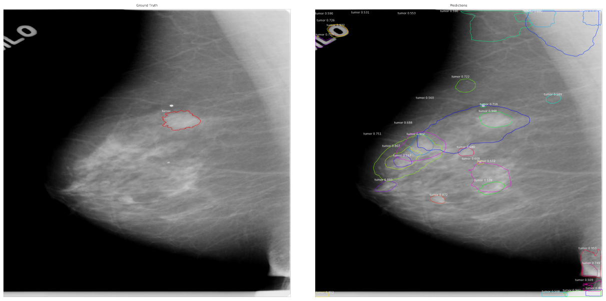

Fig. 5. Ground truth and corresponding prediction from Mask R-CNN model

Since our task has changed to instance segmentation, we report our metric of performance as AP, as in [2]. This measure takes the average over multiple IOU thresholds for which we consider positive matches; for instance, AP[0.50:0.05:0.95] is the average over the 10 IOU thresholds from 0.5 all the way to 0.95 with step size 0.05 [13]. For the single example we showed above, we can have a curve that plots precision over recall over a single IOU threshold values (in this case, 0.5) as in the Figure below.

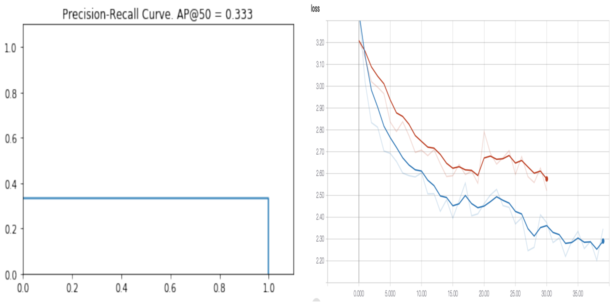

Fig. 6. Precision against recall curve; Loss across epochs (Red denotes ImageNet weights and Blue denotes COCO weights)

We then take an average over all the classes and over all threshold values to compute a mean AP score. Taking the mean AP over 50 images in our validation set, we get a score of 0.0786, which while better than the predictions we got using our previous model, does highlight how difficult our problem is. These predictions came after training the Mask R-CNN architecture from [14] and resuming training from their COCO pre-trained weights. We also tried to use their ImageNet pre-trained weights but those results were worse. We can see the loss stabilizing in the following image.

VI. Conclusion/Future Work

Our best-performing model using transfer learning on Mask R-CNN is able to predict non-zero values pixelwise but is predicting multiple masks that do not consist of the tumor for each image.

Given the plethora of models that we tried and the convergence of all the models to local minima, we now believe that the imbalanced nature of the data has a huge impact on our models. In particular, our models are not able to differentiate effectively between tissue structures in the breast and identify correct masks. We recognize that this is in part because breast cancer tumors form in a range of tissue- example Fatty vs dense tissue- which look very similar to the surrounding tissue. Furthermore, the tumors can look very different depending on whether it is a calcification or a mass, or benign or malignant. We believe that the optimal way forward is to further preprocess our mammography data. One approach is to focus on a simpler task of focusing one specific type of tumor. Some methods we are expecting to utilize for more preprocessing of the data include using k - nearest neighbours to develop regions of interest encoding on our data and the use of Countourlet Transform to denoise our data.

VII. Contributions

Overall, all three of us worked together collaboratively and had an equal share of work across all stages of this project, from preprocessing the data, experimenting with different algorithms, and finishing up the report and poster.

Code:

https://github.com/HanLinAung88/CS229-Tumor-Segmentation

References

[1] Breast cancer statistics. https://www.wcrf.org/dietandcancer/cancer-trends/breast-cancer-statistics. Accessed: 2018-12-18.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[4] Hao Dong, Guang Yang, Fangde Liu, Yuanhan Mo, and Yike Guo. Automatic brain tumor detection and segmentation using u-net based fully convolutional networks. *CoRR*, abs/1705.03820, 2017.

[5] Vivek Kumar Singh, Hatem A Rashwan, Santiago Romani, Farhan Akram, Nidhi Pandey, Md Sarker, Mostafa Kamal, Adel Saleh, Meritexell Arenas, Miguel Arquez, et al. Breast mass segmentation and shape classification in mammograms using deep neural networks. *arXiv preprint arXiv:1809.01687*, 2018.

[6] Alexander Rakhlin, Alexey Shvets, Vladimir Iglovikov, and Alexandr A. Kalinin. Deep convolutional neural networks for breast cancer histology image analysis. *CoRR*, abs/1802.00752, 2018.

[7] Parker J. Dance D. Astley S. Hutt I. Boggis C. Ricketts I. et al. Suckling, J. Mammographic image analysis society (mias) database v1.21. https://www.repository.cam.ac.uk/handle/1810/250394, 2015.

[8] Rebecca Sawyer Lee, Francisco Gimenez, Assaf Hoogi, and Daniel Rubin. Cancer imaging archive wiki, 2016.

[9] Li Shen. End-to-end training for breast cancer diagnosis using deep all convolutional networks. https://github.com/lishen/end2end-all-conv, 2017.

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[11] Mask r-cnn explained. https://becominghuman.ai/mask-r-cnn-explained-7f82bec890e3. Accessed: 2018-12-18.

[12] François Chollet et al. Keras. https://keras.io, 2015.

[13] map (mean average precision) for object detection. https://medium.com/@jonathan$_hui/map - mean - average - precision - for - object - detection - 45c121a31173. Accessed : 2018 - 12 - 18.$

[14] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. https://github.com/matterport/Mask_RCNN, 2017.

[15] 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.