Multiple-Instance and Transfer Learning for Detecting Breast Cancer

Nathan Dass
Department of CS
Stanford University
`ndass@stanford.edu`

Kais Kudrolli
Department of EE
Stanford University
`kudrolli@stanford.edu`

David Liang
Department of CS
Stanford University
`dwliang@stanford.edu`

Abstract

In this paper, we perform binary image classification using convolutional neural networks (CNNs) on mammograms. We evaluate and compare a classical deep learning approach, a transfer learning approach, and a multiple-instance learning (MIL) approach to detecting cancer in mammograms from the DDSM dataset [1]. We propose two novel techniques and implement a number of other techniques to combine information from multiple views of the same patient into one prediction. After training on our transfer learning and MIL models, we find that MIL provides significant improvements over non-MIL techniques, and we are able to achieve a precision of 0.611, recall of 0.856, and an F1 score of 0.701 on our test set using image stacking.

1 Introduction

We investigate the problem of detecting breast cancer tumors in mammograms, X-ray images of the breast that can detect cancer tumors that cannot be felt [2]. Depending on which of our three approaches we are using, our input is either a single view or an entire patient case containing 4 different views of a mammogram. In all approaches, we pass the input to a CNN, which outputs whether there is a breast cancer tumor in the example. Breast cancer affects 1 in 8 women and is expected to cause 40,000 deaths in 2018, alone [3]. Early detection using mammograms can increase a patient's chance of survival to 93% [4], but it takes a trained radiologist to find breast cancer in a mammogram, and even then, there are false-positive and false-negative diagnoses [2]. Using CNNs to supplement a radiologist or make a diagnosis if a radiologist is not available can save many lives.

2 Related work

Recently, good progress has been made in applying deep learning techniques to breast cancer detection. Jain and Levy [5] were able to achieve 0.93 accuracy in classifying a tumor as malignant or benign. However, their task is different because our models are classifying all mammograms (some without cancer) as either malignant or not. Furthermore, they "pad" each cancerous region with more pixels to make it easier to identify. We opted not to do this as it requires manual identification of the interesting area in the mammogram, and we believe there is an opportunity here to apply deep learning to solve a more general problem on any mammogram. Wu et al. [6] applied deep multiple-instance learning to image classification but use embeddings to do bag-level multiple-instance learning, which we do not do. Ilse et al. [7] achieved an F1 score of 0.577 on a different breast cancer dataset with instance-level MIL techniques. Though we do not use attention mechanisms, this would be a fantastic feature to add given their F1 score with attention is 0.712 respectively. Ilse et al. do propose both mean and max bag-level MIL methods which we implement and analyze in our report, but we also propose two novel MIL techniques (stacking and stitching images) and compare them with previous MIL and non-MIL techniques.

3 Dataset

We are using the DDSM dataset [1], which is comprised of 2,555 cases, which each contain four images, each a different view of the mammogram of a patient. There are four classes: normal (688 cases), benign without callback (140 cases), benign (814 cases), and cancer (913 cases). The normal class represents cases that did not need additional work at least four years after the initial exam. The benign class represents cases that looked suspicious but were proven to be harmless. Similarly, the benign without callback class represents cases that had something suspicious that proved to be harmless, but an additional follow up was not done. Finally, the cancer class represents cases that were proven to be malignant. We combine the benign, normal, and benign without callback classes into the non-cancer class class for our binary classification task. Note this causes a class imbalance because there are roughly 4x more non-cancer images than cancer images.

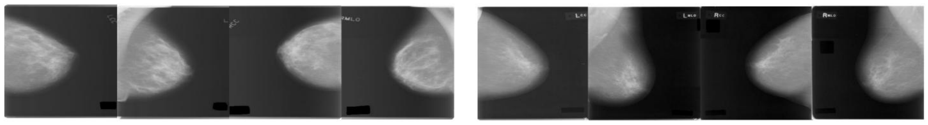
Figure 1: Four views from a benign case followed by four views from a cancer case.

With 10,220 total images, we started experiments with a 90-5-5 train/dev/test split. We split the data in two different ways: randomly splitting cases up, or randomly splitting but keeping cases together.

3.1 Data Preprocessing

Each image in the dataset was a grayscale image of varying sizes up to 7000x7000x1 pixels. To deal with the different input sizes, we reshaped all images to 299x299x1 pixels, because Inception-v3 [8] takes as input 299x299x3 images. Furthermore, we combined the normal, benign without callback, and benign classes into a single class to simplify the problem a little bit. This leaves us with a binary classification task: cancer or not cancer. We also preprocessed the data by normalizing the images by subtracting the mean pixel value over all train images from every image in train, dev, and test splits, and then dividing by the standard deviation from the pixel values. We additionally experimented with augmenting the data, and originally wrote scripts to preprocess the data before training the model, but found that it was quite expensive to store the augmentations that we wanted. In particular, we wanted to apply three rotations at 90, 180, and 270 degrees, up-down and left-right flips, and different crops to each view. This increased the size of our dataset to $\sim$70GB which was not only expensive to store but expensive to load, as well. As a result, we chose to do this augmentation randomly and on the fly.

4 Model Architectures

We mainly explored two types of model architectures: models that predicted the output based on a single view of a patient and models that predicted the output based on all of the views in a single case through MIL. Since the latter is closer to what doctors do, we expected MIL models to do better than non-MIL models. All of our models use binary cross-entropy loss because our task is binary image classification. We also added L2-regularization to all weights and filters, and used Xavier initialization throughout each of the models. To correct for the class imbalance between cancers and non-cancers we introduced a cancer class loss multiplier that scales the loss of incorrect predictions for images with cancer. Because there were roughly 4x more non-cancer images than cancer images, we set this value to 4, initially.

4.1 Baseline

For our baseline model, we fed in an input image through a few series of convolutional and max pool layers, flattened the output, passed the flattened output through a few fully connected layers, and predicted the class with a sigmoid activation function. Below is a visual depiction of what our baseline model looks like. We didn't expect this model to do great, but it would give us a good idea of how much room there is to improve performance.

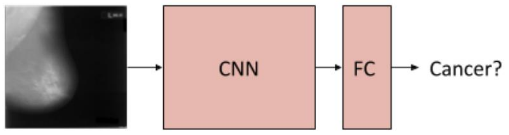

Figure 2: Simplified diagram of our baseline model.

4.2 Transfer Learning with Inception-v3

The other non-MIL model that we tried was Inception-v3, which we expected to do better than our baseline model as it is more powerful. After passing an image through Inception-v3, we added one or two fully-connected layers after the model to be trained on our input images. We ran two types of experiments when using Inception-v3: freezing the Inception-v3 part of the network and using the pre-trained weights just as an initialization.

4.3 Multiple Instance Learning

We chose the DDSM dataset because it provided multiple views for a single patient. Intuitively, a model should better detect whether or not a patient has cancer if it has more views as inputs.

A naive extension of our previous two models that would give one prediction per case would be to predict the probability of cancer for each individual view and then take the max or mean to get a final probability. The drawback is that the model treats each type of view the same as every other view. This approach is also beneficial in our case because not all of the views in a cancer case have the cancer in the view. After combining knowledge of all views in a case, we can design models that can take advantage of this extra information.

We explored three types of Multiple Instance Learning: stacking input views on top of each other, stitching input views together, and using a voting mechanism to decide the final output for a case. For all of these approaches, we used our baseline CNN architecture to get predictions. Additionally, we were able to use Inception-v3 when using a voting mechanism because the inputs to the network was the original 299x299 image size. While voting mechanisms have been explored in prior research, as far as we can tell, the stacking and stitch approaches are novel.

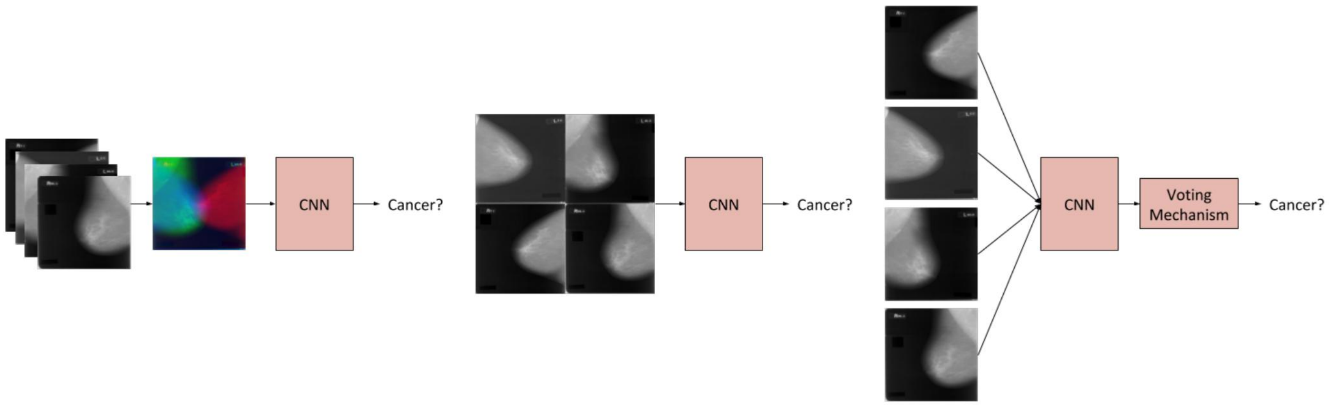

Figure 3: Three types of Multiple Instance Learning (from left to right): Stack, Stitch, and Vote.

4.3.1 Stack

Since the input images are all grayscale, the first MIL approach that we tried was stacking the views on top of each other in the third dimension. This has a similar effect of creating an RGBA image where each color is one view from a case. We hypothesized that this would work better than passing in each image one by one because single filters can learn information from all four views at once.

4.3.2 Stitch

An alternative approach to stacking is to concatenate the four views in a case in a 2x2 grid, making the network's input 598x598 instead of 299x299. The intuition behind this approach is that the network can apply the same filter to each of the four images but also combine information between each of the case images after a few layers have down-sampled the input and the feed-forward outputs from each original case are closer. For consistency, we keep the order that the images are stitched the same.

4.3.3 Vote

The last MIL approach we tried was voting: we obtained the predictions of each image, stacked the predictions on top of each other, and applied a voting mechanism to the four predictions to get a final one. We implemented three voting mechanisms: taking the maximum of the predictions, taking the mean of the predictions, and using a neural network to determine the final prediction. For the neural network voting mechanism, we simply added a fully connected layer of size 4 that took the 4 predictions as an input and reduced it to a single output. We thought that the neural network voting mechanism would have the best performance because it is analogous to a weighted vote where the network can determine which image is most important.

5 Experiments/Results/Discussion

5.1 Architecture/Hyperparameter Search

Hyperparameter	Transfer Model	MIL with stacked images
Optimizer	Adam(α, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$)	same
Learning rate, α	10^{-3}	10^{-6}
Cancer class loss multiplier	1	1.25
Mini-batch size	32	16

Table 1: Hyperparameters for our final models

We tried many sizes for the dense and convolutional layers of our initial non-MIL and MIL models. We trained all of our non-MIL models and MIL models using one base set of hyperparameters. After tuning the learning rate on all models to get the best performance on each, we observed that MIL performed better than non-MIL, and amongst the MIL models using stacked images provided the best performance. Accordingly, we tuned the stacked image model by performing a hyperparameter search. The final values for deep learning, transfer learning, and MIL models are shown in Table 1. We also found that lowering the learning rate improved performance. Finally, we introduced dropout and searched over values for the "keep probability" but found that dropout was unhelpful since we really did not have an overfitting problem.

5.2 Results

From our results in Table 2, we have found that MIL methods perform significantly better than non-MIL methods. The best transfer learning model (trainable, initialized with pre-trained weights) achieved a F1 score of 0.321 whereas the stack MIL model achieved the best F1 score of 0.698. This confirms our hypothesis that giving the model more information will allow it to make better predictions. Furthermore, we observe that the voting methods perform worse than the stack method because in one forward pass the model has access to all four cases views and can combine information in the CNN, unlike the vote methods. The stitch method performed worse than two of the vote methods and the stack method perhaps because it was hard to combine information from different views when they are separated into a grid. As a result, the network outputs are only combined when the forward pass outputs become small enough and either convolution/pool layers combine them or the dense layers combine the outputs.

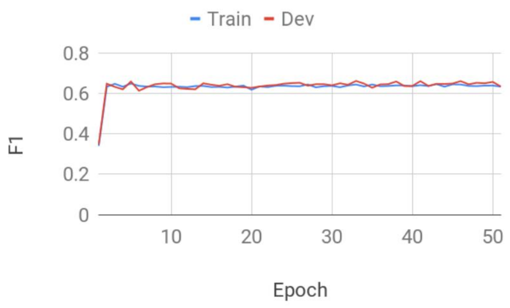

Figure 4: Train and Dev F1 Score for final model, MIL with stacked images

Model	Precision	Recall	F1 Score
Shallow (Baseline)	0.164	0.375	0.219
Transfer (Frozen)	**0.249**	0.503	0.314
Transfer (Trainable)	0.238	**0.551**	**0.321**
Stack	**0.611**	0.856	**0.701**
Stitch	0.528	0.714	0.602
Vote w/ Max	0.479	0.904	0.625
Vote w/ NN	0.492	**0.917**	0.637
Vote w/ NN + Transfer	0.524	0.514	0.514

Table 2: Precision, Recall, and F1 for all of our models measured on the test set

5.3 Discussion

Our initial cancer loss multiplier value of 4 was an over-correction and caused many non-cancers to be classified as cancer. This is evidenced by our high recall and a low precision, indicating a high false-positive rate. We think that with batch gradient descent, a multiplier of 4 would have might have worked well, but because mini-batches could have high variance in the number of cancer cases, the gradient could sometimes be overcompensating too much if there were too many cancer classes in the mini-batch. Eventually, we found that 1.25 was a good multiplier.

While transfer learning for non-MIL performed well compared to other non-MIL methods, transfer learning for MIL gave worse results in comparison to other MIL methods. We don't think this is due to not training for enough time as we were able to run these models for 300+ iterations. It is possible that there is simply not enough data to train such deep networks, and augmentation did not provide enough additional data.

We reduced the size of our images from up to 7000x7000 pixels to 299x299 pixels to comply with Inception-v3's input requirements. To allow for fair comparisons, we kept this size for the MIL methods. However, this may have caused a serious loss in information as the areas of interest are already quite small in the 7000x7000 pixel images. We tried running on bigger images, but it seemed to get much slower and we found it a more worthwhile investment of our time to find better hyperparameters, so we leave running MIL methods on larger images as future work.

6 Future Work

With more time, we first would use saliency maps to determine what parts of the image the network looks at when predicting cancer and see if the model focuses on different areas for different views. Second, we think image segmentation and/or attention mechanisms would help our network ignore large uninteresting regions in our images and hone in on regions that are important for determining if there is a malignant tumor. Third, we would try other MIL techniques like learning from low-dimensional image embeddings proposed by Ilse et al. [7]. Finally, we would run the model on higher resolution images to preserve more information from the original images. We believe that more information will improve the performance of our models.

7 Conclusion

We showed that MIL techniques provide significant improvements in model performance over transfer learning. We also proposed two new MIL techniques, stack and stitch, that provided better results than other MIL techniques. Class imbalance was a large factor in the performance of our models, and we compensated by introducing the cancer class loss multiplier. Our final best model used stacked images to combine multiple views of a mammogram to make a more informed prediction even if not all views contained cancer.

8 Contributions

- Nathan: Implemented splitting data into train, dev, and test sets, generating stitched images, calculating metrics, and all MIL voting models.
- Kais: Implemented loading data into Keras model, baseline, and Inception-v3, and ResNet models in Keras, data augmentation, MIL stitch model, and did the hyperparameter search.
- David: Implemented preprocessing DDSM data to convert it to a usable format, baseline and Inception-v3 models in Tensorflow, TensorFlow data iterator, a common architecture for our models to build off of, and MIL stack model.
- Everyone worked together to think of experiments to run, analyze results from experiments, and write the report.

References

[1] "U.S. Breast Cancer Statistics | Breastcancer.org." Breastcancer.org, 2018. [Online] Available: `https://www.breastcancer.org/symptoms/understand_bc/statistics`. [Accessed 14 Dec. 2018].

[2] "Mammograms." National Cancer Institute, 2016. [Online]. Available: `https://www.cancer.gov/types/breast/mammograms-fact-sheet`. [Accessed: 16-Dec-2018].

[3] "U.S. Breast Cancer Statistics." Breastcancer.org, 2018. [Online]. Available: `https://www.breastcancer.org/symptoms/understand_bc/statistics`. [Accessed: 16-Dec-2018].

[4] "Early Detection is Key." Carol Milgard Breast Center, 2018. [Online]. Available: `http://www.carolmilgardbreastcenter.org/early-detection`. [Accessed: 16-Dec-2018].

[5] "Breast Mass Classification from Mammograms using Deep Convolutional Neural Networks." Daniel Lévy, Arzav Jain, 2016. [Online]. Available: `https://arxiv.org/abs/1612.00542`. [Accessed: 16-Dec-2018].

[6] "Multiple-instance learning based decision neural networks for image retrieval and classification." Lei Zhou, Yu Zhao, Jie Yang, Qi Yu, Xun Xu, 2016. [Online]. Available: `https://dl.acm.org/citation.cfm?id=2839977`. [Accessed: 16-Dec-2018].

[7] "Attention-based Deep Multiple Instance Learning." Maximilian Ilse, Jakub M. Tomczak and Max Welling, 2018. [Online]. Available: `https://arxiv.org/abs/1802.04712`. [Accessed: 16-Dec-2018].

[8] "Rethinking the Inception Architecture for Computer Vision." Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens and Zbigniew Wojna, 2015. [Online]. Available: `https://arxiv.org/abs/1512.00567`. [Accessed: 16-Dec-2018].

[9] "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, 2015. [Online]. Available: `https://www.tensorflow.org/`.

[10] "Keras." Fraçois Chollet and others, 2015. [Online]. Available: `https://keras.io`.

GitHub

`https://github.com/ndass6/deep-mammogram`