End-to-end Training for Whole Image Breast Cancer Diagnosis using An All Convolutional Design

Li Shen
Icahn School of Medicine at Mount Sinai
New York, NY 10029
li.shen@mssm.edu, shenli.sam@gmail.com

Abstract

We develop an end-to-end training algorithm for whole-image breast cancer diagnosis based on mammograms. It has the advantage of training a deep learning model without relying on cancer lesion annotations. Our approach is implemented using an all convolutional design that is simple yet provides superior performance in comparison with the previous methods. With modest model averaging, our best models achieve an AUC score of 0.91 on the DDSM data and 0.96 on the INbreast data. We also demonstrate that a trained model can be easily transferred from one database to another with different color profiles using only a small amount of training data.

Code and model availability: https://github.com/lishen/end2end-all-conv

1 Introduction

With the rapid advancement in machine learning and especially, deep learning in recent years, there is a keen interest in the medical imaging community to apply these techniques to cancer screening. Recently, a group of researchers, along with the Sage Bionetworks and the DREAM community organized a challenge for competing teams to develop algorithms to improve breast cancer diagnosis using a large database of digital mammograms (DM) [1]. The main purpose of the challenge is to predict the probability that a patient will develop cancers within 12 months based on mammographic images. The final results of the competitive phase showed that machine learning methods can be used to obtain very high accuracy in breast cancer screening [2], with the top performing team achieving the area under the receiver operating characteristic curve (AUC) score of 0.87. We also participated in the challenge [3] and obtained a reasonably good result which ranked #12 on the leaderboard [4]. However, much can still be done to further improve the result.

Mammography based breast cancer diagnosis is a challenging problem and faces two dilemmas that distinguish it from other typical image classification problems, such as the ImageNet challenge [5]. The first dilemma is caused by large image size and small sample size. A mammogram is typically in the size of 4000x3000 (height by width) pixels while the cancerous lesion or the region of interest (ROI) can be as small as 100x100 pixels. Resizing a large mammogram to 224x224 – a common choice used by the image classification filed – will likely make the ROI hard to detect and/or classify. If we use large image size it will inevitably increase a model's parameter space and therefore demands more samples for training. However, most databases to date have a sample size in the order of thousands of images or less. For a survey, see [6]. The DM challenge boasts a large database with more than 640,000 images from over 86,000 women but its dataset is highly imbalanced: with only 2000 or so positive cases in the public train set, which significantly reduces the effective sample size. Therefore, directly training a classification model on such a dataset from scratch is unlikely to lead to good performance.

The second dilemma is caused by the availability of ROI annotations. When ROI annotations are provided, the diagnostic problem can be easily casted as an object detection task, which has already been well studied in the computer vision field. For example, the region-based convolutional neural network (R-CNN) approach [7] and its variants [8]–[10] can be applied here. However, many databases including the DM challenge do not contain ROI annotations. If we train an R-CNN model based on a database with known ROIs, it will not be straightforward to transfer that model onto another database **without** ROIs due to color profile mismatch.

Obtaining ROI annotations requires expertise in breast cancer image reading and therefore is difficult for most machine learning researchers to do. This implies we will need to train a whole image classifier based on image-level labels only and then have to face the first dilemma as discussed above.

Our entry to the DM challenge used an approach that was inspired by the winning method in the Camelyon16 challenge [11]. Briefly, a patch classifier was trained on a public dataset with pixel-level ROI annotations and used to scan an image to produce a so called probabilistic heatmap. Each pixel of the heatmap represents the three probabilities of the corresponding patch being background, benign or malignant. Regional properties, such as max intensity, area and diameter, were extracted from binarized heatmaps and fed into a gradient boosting tree classifier for whole image classification. We achieved reasonable results with this method on the public dataset but observed a decrease in performance when applied the trained model to the DM challenge data. This is mainly because the two datasets have different color profiles. Because this method heavily relies on the patch classifier to tell the malignancy of local regions on a mammogram, and the patch classifier cannot be finetuned without ROI annotations, it is not straightforward to transfer this whole image classifier to the DM challenge data. We needed an end-to-end training strategy that can train a model based on only image-level labels. During the final stage of the challenge, we realized the end-to-end training can be achieved by using the convolutional property to change the input size from patch to whole image, create larger feature maps, and then add additional convolutional layers on top to produce the final labels. We discussed this possibility in our write-up [3]. However, we were not able to implement this strategy in the challenge due to time constraint. After the final results came out, we found that a similar idea was employed by the top-performing team [12].

The purpose of this work is to continue investigating this line of methods by testing different configurations of patch classifiers and the top layers for whole image breast cancer diagnosis using public databases. We will focus more on the network structures than different augmentation or ensemble tricks to boost the performance. In particular, we find that adding convolutional layers on top of the patch classifier without using the patch classifier's probabilistic output not only simplifies the design but also provides better performance. We argue that is because the probabilistic output is merely a nonlinear transformation that squashes output values into the [0, 1] range, which may impede information flow in back-propagations.

The manuscript is organized as follows. In Section 2, we will study the performance of different network structures to convert from a patch classifier into a whole image classifier based on a public database with ROI annotations. In Section 3, we will finetune the whole image classifiers on another public database to prove its transferability **without** using ROI annotations. In Section 4, we evaluate the performance using model ensembles. In Section 5, we provide some discussion of the results. We hope, by virtue of this study, we can provide further insights into this promising method for whole image classification for the medical imaging community.

2 End-to-end training for whole image classification on DDSM

2.1 Setup and processing of the dataset

The Digital Database for Screening Mammography (DDSM) [13] is the largest public database for mammograms. It contains digitized images from scanned films but uses a lossless-JPEG format that is obsolete. We use a modernized version of the database called CBIS-DDSM [14] which contains images that have already been converted into the standard DICOM format. Our downloaded dataset on Mar 21, 2017 from CBIS-DDSM's website contains the data for 1249 cases or 2584 mammograms, which represents a subset of the original DDSM database. It includes the cranial cardo (CC) and media lateral oblique (MLO) views for most of the screened breasts. Using both views for prediction shall provide better result than using each view separately. However, we will treat each view as a standalone image in this study due to the limitation in sample size. Our purpose is to predict the malignancy status for an entire mammogram. We perform an 85-15 split on the dataset into train and test sets based on the cases. For the train set, we further set aside 10% of the cases as the validation set. The total numbers of images in the train, validation and test sets are: 1903, 199 and 376, respectively.

The DDSM dataset contains the pixel-level annotation for the ROIs and their pathology confirmed labeling: benign or malignant. It further contains the type of each ROI, such as calcification or mass. Most mammograms contain only one ROI while a small portion contain more than one ROIs. We first convert all mammograms into PNG format and resize them into 1152x896. We then create several datasets by sampling image patches from ROIs and background regions. All patches have the same size of 224x224. The first dataset (referred as s1) is from patches that are centered on each ROI and a random background patch from the same mammogram. The second dataset (s10) is from 10 sampled patches around each ROI with a

Table 1: Test accuracies of the patch classifiers using the Resnet50 and VGG16/19 structures. #Epochs indicates the epoch when the best validation accuracy has been reached.

Model	**Pretrained**	**#Patches per image**	**Accuracy**	**#Epochs**
Resnet50	N	1	0.97	198
Resnet50	Y	1	0.99	99
Resnet50	N	10	0.63	24
Resnet50	Y	10	0.89	39
VGG16	Y	10	0.84	25
VGG19	Y	10	0.79	15
Resnet50	Y	30	0.91	23
VGG16	Y	30	0.86	22
VGG19	Y	30	0.89	24

minimum overlapping ratio of 0.9 and the same number of background patches from the same mammogram. The third dataset (s30) is created the same way as the second dataset but we increase the number of patches to 30. According to the annotations of the ROIs, all patches are classified into five different categories: background, calcification-benign, calcification-malignant, mass-benign and mass-malignant.

To make it easier to convert a patch classifier into a whole image classifier, we calculate the pixel-wise mean for the mammograms on the train set and use this value for pixel-wise mean centering for both patch and whole image training. No other preprocessing is applied. To compensate for the lack of sample size, we use data augmentation on-the-fly for both patch and whole image training. The followings are the random image transformations we use: horizontal and vertical flips, rotation in [-25, 25] degrees, shear in [-0.2, 0.2] radians, zoom in [0.8, 1.2] ratio and channel shift in [-20, 20] pixel values.

2.2 Development of patch classifiers

To train a classifier to classify a patch into the five classes, we use three popular convolutional network structures: the 50-layer residual network (Resnet50) [15] and the 16- and 19-layer VGG networks (VGG16 and VGG19) [16]. We further consider the networks with their weights pretrained on the ImageNet database against the randomly initialized ones. Pretraining may help in speedup learning as well as improving generalization of the networks in comparison with random initialization. When finetuning a pretrained network, we note that the bottom layers represent primitive features that tend to be invariant across datasets. While the top layers represent higher representations that are more related to the final labels and therefore need to be trained more aggressively. This demands a higher learning rate for the top layers than the bottom layers. Learning rate multiplier is a feature that allows one to adjust the learning rate for each layer separately but is not available in Keras [17] – the computational framework that we use for this work. Therefore, we develop a 3-stage training strategy that freeze the parameter learning for all layers except the last one and progressively unfreeze the parameter learning from top to bottom and decrease the learning rate at the same time. The details of this training strategy is as follows:

1. Set learning rate to 1e-3 and train the last layer for 3 epochs.
2. Set learning rate to 1e-4, unfreeze the top layers and train for 10 epochs, where the top layer number is set to 46 for Resnet50, 11 for VGG16 and 13 for VGG19.
3. Set learning rate to 1e-5, unfreeze all layers and train for 37 epochs.

In the above, an epoch is defined as a sweep through the train set. The total number of epochs is 50 and early stopping is used if training does not improve the validation loss. The above epoch values are the same for the s10 and s30 datasets. For the s1 dataset, we increase the total number of epochs to 200 and adjust the value for each stage accordingly. For randomly initialized networks, we use a learning rate of 1e-3. Stochastic gradient descent (SGD) is used to optimize the networks with Adam [18] as the optimizer and the batch size is set as 32. We also adjust the sample weights within a batch to keep the classes balanced.

We evaluate the models using test accuracy of the five classes. The results are summarized in Table 1. On the s1 dataset, both randomly initialized and pretrained Resnet50 models achieve very high accuracies but the pretrained network converges much faster: cutting down the number of epochs by half. On the s10 dataset,

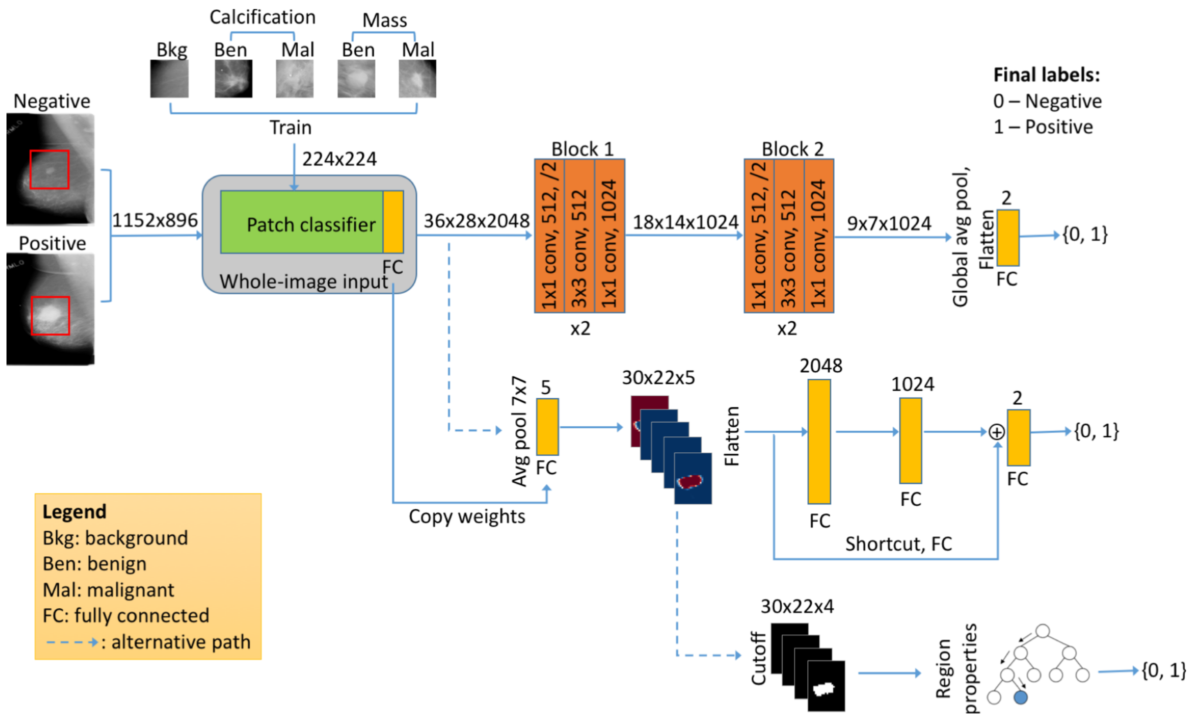

Figure 1: The deep learning structure for converting a patch classifier into a whole image classifier by adding convolutional layers on top. Two alternative strategies are also presented: one is to add probabilistic heatmaps and FC layers on top and the other is to add a random forest classifier on top of the heatmap.

the pretrained Resnet50 performs much better than the randomly initialized Resnet50: a 0.26 difference in test accuracy. We therefore conclude that pretraining can help us train networks faster and produce better models. We use pretrained networks for the rest of the study. On both s10 and s30 datasets, Resnet50 outperforms VGG16 and VGG19. VGG16 performs better than VGG19 on the s10 dataset but the order is reversed on the s30 dataset.

2.3 Conversion from patch to whole image classifiers

We now convert the patch classifiers into whole image classifiers by testing many different configurations (Fig. 1). This allows us to train classifiers based on whole-image labels, without relying on ROI annotations. It is done by first altering the input size from 224x224 to 1152x896, which proportionally increases the feature map size for every convolutional layer. For Resnet50, an input size of 1152x896 implies the feature map size at the last convolutional layer becomes 36x28. If we take 7x7 average pooling, we will have a feature map of 30x22. We can then copy the softmax output layer from the patch classifier to add on top of this average pooling layer and get probabilistic outputs for the five classes for each of the 30x22 elements, i.e. producing a probabilistic heatmap. If we add a random forest classifier on top of the heatmap, that will give the solution of our entry in the challenge [3]. Alternatively, we can add a few fully connected (FC) layers and a shortcut connection similar to what the top performing team did [12]. Although the addition of the probabilistic heatmap feels natural when converting from a patch into a whole image classifier, we find it to be unnecessary and may even impede the training (For some analysis, see Appendix 1). Indeed, the probabilistic heatmap may become a bottleneck that prevents the gradients from flowing backward, which causes the patch classifier layers to train slowly when used in whole image training.

From the above analysis, it looks like a better strategy is to get rid of the probabilistic heatmap completely and add convolutional layers directly on top of the last convolutional output of the patch classifier. For Resnet50, we just need to add two additional residual blocks each with a stride of 2 to reduce the feature map size to 9x7 before the global average pooling and softmax output. Like section 2.2, we develop a 2-stage training strategy to avoid unlearning the important features at the bottom layers. After some trial-and-errors, we decide to use smaller learning rates than usual to prevent the training from diverging. The details of the 2-stage training are as follows:

Table 2: Test AUC scores of the whole image classifiers using the Resnet50 as patch classifiers. #Epochs indicates the epoch when the best validation score has been reached.

Add residual blocks on top				
#Patches per image	**Block1**	**Block2**	**AUC**	**#Epochs**
1	[512-512-2048] x 1	[512-512-2048] x 1	0.63	35
10	[512-512-2048] x 1	[512-512-2048] x 1	0.85	20
10	[512-512-1024] x 2	[512-512-1024] x 2	0.86	25
10	[256-256-256] x 1	[128-128-128] x 1	0.84	25
10	[256-256-256] x 3	[128-128-128] x 3	0.83	17
10	[256-256-512] x 3	[128-128-256] x 3	0.84	48
30	[512-512-1024] x 2	[512-512-1024] x 2	0.81	49
30	[256-256-256] x 1	[128-128-128] x 1	0.84	40
Add heatmap and FC layers on top				
	Heatmap pool size	**FC1 / FC2**		
10	5x5	64 / 32	0.67	28
10	2x2	512 / 256	0.68	47
10	1x1	2048 / 1024	0.74	43
Probabilistic heatmap + random forest classifier				
10	#trees=500, max depth=9, min samples split=300		0.73	

Table 3: Test AUC scores of the whole image classifiers using the VGG16 (s10) and VGG19 (s30) as patch classifiers. #Epochs indicates the epoch when the best validation score has been reached. $^{\$}$Result obtained from extended model training (See text for more details).

Add VGG/residual blocks on top (* is residual block)				
#Patches per image	**Block1**	**Block2**	**AUC**	**#Epochs**
10	512 x 3	512 x 3	0.71	47
10	512 x 1	512 x 1	0.83	44
10	256 x 3	128 x 3	0.78	30
10	256 x 1	128 x 1	0.80	35
10	128 x 1	64 x 1	0.82	46
10	*[512-512-1024] x 2	*[512-512-1024] x 2	0.81, $^{\$}$0.85	46
30	512 x 1	512 x 1	0.75	15
30	128 x 1	64 x 1	0.75	1
Add heatmap and FC layers on top				
	Heatmap pool size	**FC1 / FC2**		
10	5x5	64 / 32	0.66	26
10	2x2	512 / 256	0.71	27
10	1x1	2048 / 1024	0.78	50

1. Set learning rate to 1e-4, weight decay to 0.001 and train the newly added top layers for 30 epochs.
2. Set learning rate to 1e-6, weight decay to 0.01 and train all layers for 20 epochs.

Due to memory constraint, we use a small batch size of 2 for whole image training. The other parameters are the same as the patch classifier training.

We evaluate different models using the AUC scores on the test set. We first test the conversion based on the Resnet50 patch classifiers. The results are summarized in Table 2. In the original residual network design [15], the authors increased the dimension for each residual block from the previous block to compensate for

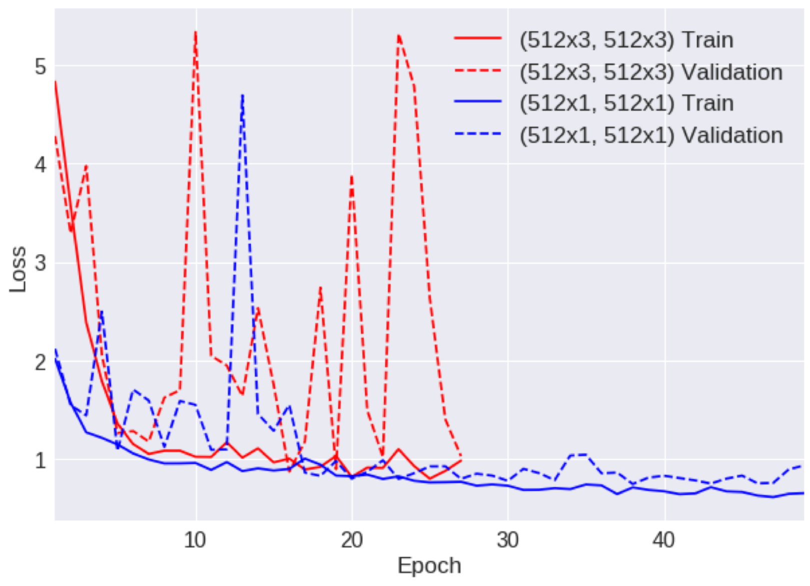

Figure 2: Train and validation loss curves of two VGG structures: one is more complex than the other and suffers from overfitting.

the reduction in feature map size. This avoids creating “bottlenecks” in computation. However, we are not able to do the same here due to memory constraint. In our first test, we use two residual blocks with the same dimension and a bottleneck design (see [15]) of [512-512-2048] without repeating the residual units. This gives an AUC score of 0.85 for the Resnet50 trained on the s10 set and 0.63 for the Resnet50 trained on the s1 set, a large 0.22 discrepancy in score. We hypothesize that because the s10 set is 10x larger than the s1 set, it contains much more information about the variations of benign and malignant ROIs and their neighborhood regions. This information helps a lot in training a whole image classifier to locate and classify cancer-related regions for diagnosis. We then focus only on patch classifiers trained on the s10 and s30 sets for the rest of the study. We further vary the design of the residual blocks by reducing the dimension of the last layer in each residual unit to 1024 and repeating each residual unit twice, i.e. two [512-512-1024]x2 blocks. This design slightly increases the score to 0.86. We also reduce the dimensions of the first and second blocks and find the scores to drop by only 0.02-0.03. Therefore, we conclude that the dimensions for the newly added top layers are not critical to the performance of the whole image classifiers.

We then test the conversion based on the VGG16 patch classifier trained on the s10 set. The newly added VGG blocks all use 3x3 convolutions with batch normalization (BN). The results are summarized in Table 3. We find that the VGG structure is more likely to suffer from overfitting than the residual networks. However, this can be alleviated by reducing the model complexity of the newly added VGG blocks. To illustrate this, we plot the train and validation losses for two VGG configurations during training (Fig. 2): one is two VGG blocks with the same dimensions of 512 and 3x repetitions and the other has the same dimensions but no repetition; the first VGG structure contains more convolutional layers and therefore has higher complexity than the second one. It can be seen that the first VGG network has difficulty in identifying a good local minimum and suffers badly from overfitting. While the second VGG network has smoother loss curves and smaller differences between train and validation losses. We further reduce the dimensions of the VGG blocks and find the scores to decrease by only a small margin. This is in line with the results on the Resnet50 based models. Overall, the Resnet50 based whole image classifiers perform better than the VGG16 based ones (Tables 2 & 3). In addition, the Resnet50 based models seem to achieve the best validation score earlier than the VGG16 based models. Lastly, we add two residual blocks on top of the VGG16 patch classifiers, creating a “hybrid” model, and also get a good score of 0.81.

Encouraged by the performance leap going from the s1 to the s10 set, we choose two residual configurations and two VGG configurations and add them on top of the Resnet50 and the VGG19 patch classifiers trained on the s30 set. We expect to obtain even further performance gains. Surprisingly, the performance decreases for both Resnet50 and VGG19 based models (Tables 2 & 3). Indeed, overfitting happens for the VGG19 based model training at early stage and the validation loss stops improving further. It is obvious that a good patch classifier is critical to the performance of a whole image classifier. However, merely increasing the number of sampled patches does not necessarily lead to better whole image classifiers. We leave to future work to study

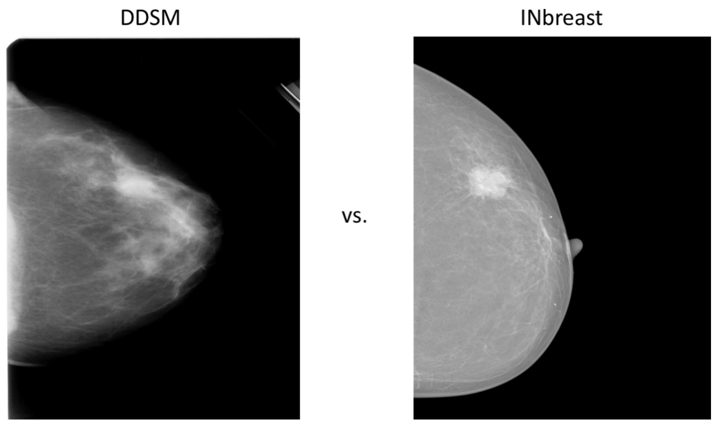

Figure 3: Comparison of two example mammograms from DDSM and INbreast.

how to sample patches more efficiently and effectively to help building better image classifiers.

In [12], Yaroslav used a different design that adds a probabilistic heatmap on top of the patch classifier and followed by a few FC layers and a shortcut connection. We do the same here for the Resnet50 and VGG16 patch classifiers trained on the s10 set. We choose from a few different filter sizes for the max pooling layer after the heatmap and follow the same idea of [12], gradually decrease the size of the FC layers until output. We also add a shortcut between the heatmap and the output, implemented by an FC layer. The results are summarized in Tables 2 & 3. First, we find the max pooling layer to be destructive to the heatmap features and harm the scores. Second, the image classifiers from this design generally underperform the image classifiers from the all convolutional design. We note that our implementation is not an exact replicate of the work in [12]. They have used a modified VGG network by using 6 VGG blocks instead of 5 and adding BN layers for each convolutional layer, both of which may help improving their results. However, because we use pretrained networks with fixed structures in this study, it is not straightforward to modify our networks to be exactly the same as theirs. Nonetheless, we have shown that our all convolutional design is superior to adding probabilistic heatmap and FC layers on top, based on the same patch classifier at the bottom. We believe the top performing team's score can be further improved if they adopt our all convolutional design.

Finally, we test the same strategy as [3], [11]: set a cutoff to binarize the heatmaps; extract regional features; and train a random forest classifier based on the features. The Resnet50 trained on the s10 set is used as the patch classifier. We use four different cutoffs – 0.3, 0.5, 0.7, 0.9 and combine the features. We use a random forest classifier with 500 trees. This gives a test score of 0.73, which seems to be in line with our score of 0.65 in the challenge. There are two major reasons for the increase of the score in this study: 1. the patch classifier in this study has been given more training time and produces better accuracy; 2. the train and test sets are both from the same database. Clearly, this method performs worse than the end-to-end trained deep convolutional networks.

3 Transfer learning for whole image classification on INbreast

3.1 Setup and processing of the dataset

The INbreast [6] dataset is another public database for mammograms. It is more recent than the DDSM and contains full-field digital images as opposed to digitized images from films. These images have different color profiles from the images of DDSM, which can be visually confirmed by looking at two example images from the two databases (Fig. 3). Therefore, this is an excellent source to test the transferability of a whole image classifier from one database to another. The INbreast database contains 115 cases and 410 mammograms including both CC and MLO views. It includes the BI-RADS readings for the images but lacks biopsy confirmation. Therefore, we manually assign all images with BI-RADS readings of 1 and 2 as negative samples; 4 and 5 as positive samples; and ignore BI-RADS readings of 3. This excludes 12 cases or 23 mammograms from further analysis. We note that the categorization based on the BI-RADS readings makes the INbreast dataset inherently easier to classify than the DDSM dataset. This is because two mammograms with different BI-RADS readings are already visually discernible according to radiologists, which biases the labels. We perform a 70-30 split on the dataset into train and validation sets based on the cases. The total numbers of images in the train and validation sets are: 280 and 107, respectively.

Table 4: Transfer learning efficiency with different train set sizes.

#Patients	**#Images**	**Resnet50**	**VGG16**	**VGG-Resnet Hybrid**
20	79	0.78	0.87	0.89
30	117	0.78	0.90	0.90
40	159	0.82	0.90	0.93
50	199	0.80	0.93	0.93
60	239	0.84	0.95	0.91

We use exactly the same processing steps on the INbreast images as the DDSM images.

3.2 Effectiveness and efficiency of transfer learning

To prove that the whole image classifiers can be transferred onto the INbreast database without using the ROI annotations, we directly finetune the whole images classifiers on the train set and evaluate the model performance using validation AUC scores. Two best models – the Resnet50 + two [512-512-1024]x2 residual blocks and the VGG16 + two 512x1 VGG blocks – are used for transfer learning. We set the learning rate at 1e-6 with the Adam optimizer, the number of epochs at 200 and the weight decay to be 0.01. The finetuned Resnet50 based model achieves a score of 0.84. Surprisingly, the finetuned VGG16 based model achieves a score of 0.92, better than the Resnet50 based models. Yaroslav argued in [12] that the VGG structure is better suited than the residual structure for whole image classification because the residual networks reduce the feature map sizes too aggressively to damage the ROI features at the first few layers. To validate that, we use the hybrid model from last section, which has the VGG16 patch classifier at the bottom and two residual blocks on top, to finetune on the INbreast dataset. This hybrid model achieves a very high score of **0.95**. This proves that the bad performance of the Resnet50 based models are not due to the inefficacy of the residual blocks on top but rather the destructive properties of the first few layers of Resnet50. However, if the VGG structure is indeed better than the residual structure at the bottom layers, how does Resnet50 beat VGG16/19 on the DDSM data? We reason that is because the VGG networks need to train longer than the residual networks to reach their full potential. This is in line with the observation that the VGG16 based whole image classifiers achieve the best validation scores on DDSM at later stage than the Resnet50 based ones (Tables 2 & 3). Our choice of 50 epochs on DDSM is mainly driven by computational resource limitation. The residual networks are powered by BN and shortcuts to speed up training and are able to converge better than the VGG networks within the 50 epochs. To prove that, we perform another run of model training for the VGG16 + residual blocks hybrid model on the DDSM data with 200 additional epochs. The model improves the score from 0.81 to 0.85 (Table 3), which is as good as the best Resnet50 based model. We did not perform additional training for the other VGG16 based models due to computational constraint.

Finally, we test the efficiency of the transfer learning in terms of the training data required. We sample a subset with 20, 30, …, 60 cases from the train set for finetuning and evaluate the model performance on the same validation set (Table 4). With as little as 20 cases or 79 images, the VGG16 based and the hybrid models can achieve scores of 0.87 and 0.89, respectively. The scores seem to quickly saturate as we increase the train set size. We hypothesize that the "hard" part of the learning is to recognize the textures of the benign and malignant ROIs while the "easy" part is to adjust to different color profiles. This quick adjustment can be a huge advantage for the whole image classification based methods. In future works, a whole image classifier can be finetuned to make predictions on another database with only a small amount of whole-image labeled training data. This greatly reduce the burden of train set construction.

4 Performance of model ensembles of whole image classifiers

We use inference-level augmentation by doing horizontal and vertical flips to create four predictions and take an average. On DDSM, three best performing models: Resnet50 with two [512-512-1024]x2 residual blocks, VGG16 with two 512x1 VGG blocks and VGG16 with two [512-512-1024]x2 residual blocks on top (hybrid) are selected. The augmented predictions for the three models improve AUC scores from 0.86→0.88, 0.83→0.86 and 0.85→0.88, respectively. The average of the three augmented models gives an AUC score of **0.91**.

With augmented prediction on the INbreast data, the VGG16 based model improves the score from 0.92→0.94 and the hybrid model improves the score from 0.95→0.96. The average of the two augmented models gives a score of

0.96. Including the Resnet50 based model in model averaging does not improve the score.

5 Discussion

We have shown that accurate whole-image breast cancer diagnosis can be achieved with a deep learning model trained in an end-to-end fashion that is independent from ROI annotations. The network can be based on an all convolutional design that is simple yet powerful. It can be seen that high-resolution mammograms are critical to the accuracy of the diagnosis. However, large image size can easily lead to an explosion of memory requirement. If more GPU memory becomes available in the future, we shall return to this problem and train our models with larger image sizes or even use the original resolution without downsizing. This will provide much more details of the ROIs and can potentially improve the performance.

The decrease of the scores from the s10 to the s30 set is a surprise. Yet it indicates the intricacy of training a whole image classifier. More research is needed to make the whole image training more robust against divergence and overfitting, especially when the train set is not large. Patch sampling can also be made more efficient by focusing more on the difficult cases than the easy ones.

Our result supports Yaroslav's argument [12] that the VGG networks are more suitable than the residual networks for breast cancer prediction based on mammograms. However, we also demonstrate the superiority of the residual networks to the VGG networks in several aspects. The only problem with the residual structure is the first few layers that may destroy the fine details of the ROIs. In future works, we can modify the original residual network to make the first few layers less aggressive in reducing the feature map sizes. That shall lead to improved performance for the residual networks.

Computational environment

The research in this study is carried out on a Linux workstation with 8 CPU cores and a single NVIDIA Quadro M4000 GPU with 8GB memory. The deep learning framework is Keras 2 with Tensorflow as the backend.

References

[1] A. D. Trister, D. S. M. Buist, and C. I. Lee, "Will Machine Learning Tip the Balance in Breast Cancer Screening?," *JAMA Oncol*, May 2017.

[2] "The Digital Mammography DREAM Challenge - Final Results." [Online]. Available: https://www.synapse.org/#!Synapse:syn4224222/wiki/401763. [Accessed: 23-Aug-2017].

[3] L. Shen, "Breast cancer diagnosis using deep residual nets and transfer learning," 2017. [Online]. Available: https://www.synapse.org/#!Synapse:syn9773182/wiki/426912. [Accessed: 23-Aug-2017].

[4] "The Digital Mammography DREAM Challenge - Final Ranking of Validation Round." [Online]. Available: https://www.synapse.org/#!Synapse:syn4224222/wiki/434546. [Accessed: 23-Aug-2017].

[5] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[6] I. C. Moreira, I. Amaral, I. Domingues, A. Cardoso, M. J. Cardoso, and J. S. Cardoso, "INbreast: Toward a Full-field Digital Mammographic Database," *Academic Radiology*, vol. 19, no. 2, pp. 236–248, Feb. 2012.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, 2014, pp. 580–587.

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[10] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks," *arXiv:1605.06409 [cs]*, May 2016.

[11] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," *arXiv:1606.05718 [cs, q-bio]*, Jun. 2016.

[12] Yaroslav Nikulin, "DM Challenge Yaroslav Nikulin (Therapixel)." [Online]. Available: https://www.synapse.org/#!Synapse:syn9773040/wiki/426908. [Accessed: 23-Aug-2017].

[13] Michael Heath, Kevin Bowyer, Daniel Kopans, Richard Moore, and W. Philip Kegelmeyer, "The Digital Database for Screening Mammography," in *Proceedings of the Fifth International Workshop on Digital Mammography*, 2001, pp. 212–218.

[14] R. S. Lee, F. Gimenez, A. Hoogi, and D. Rubin, "Curated Breast Imaging Subset of DDSM," *The Cancer Imaging Archive*, 2016.
[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015.
[16] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Sep. 2014.
[17] F. Chollet and others, *Keras*. GitHub, 2015.
[18] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014.

Appendix 1

To understand why the probabilistic heatmap may impede whole image training, let's perform some analysis on a hypothetical network structure. For brevity, we assume there is only one neuron per layer since for multiple neurons, we can simply add up their derivatives. Therefore, we can ignore the average pooling layer and reduce the softmax function to the logistic sigmoid function. Let's assume we have an input of x and the corresponding target t, the following are the sequence of transformations the input will go through:

- Patch classifier from input to the last convolutional layer: $w = f(x)$, parametrized by weights and biases Θ.
- Patch classifier output: $y = g(w)$. When the output is a probability, $g(w) = \frac{1}{1+\exp(-w)}$. If we want to turn off the probabilistic output, set $g(w) = w$.
- The output of the top layers added to the patch classifier: $z = h(y)$.
- The loss of the input: $e = l(z, t)$.

During training, we want to update the parameters Θ of the patch classifier. Without loss of generality, for any parameter $\theta \in \Theta$, we have the following by applying the train rule:

$$\frac{\partial e}{\partial \theta} = \frac{\partial e}{\partial z} \cdot \frac{\partial z}{\partial y} \cdot \boldsymbol{\frac{\partial y}{\partial w}} \cdot \frac{\partial w}{\partial \theta}$$

where

$$\frac{\partial y}{\partial w} = \begin{cases} 1, & g(w) = w \\ \dfrac{1}{[1+\exp(w)][1+\exp(-w)]}, & g(w) = \dfrac{1}{1+\exp(-w)} \end{cases}$$

therefore, the derivative of $\frac{\partial y}{\partial w}$ is always a constant when there is no probabilistic output. But when the probabilistic output is used, the derivative approaches zero when the convolutional output w becomes large. This can be easily seen by

$$\lim_{w \to +\infty \text{ or } w \to -\infty} \frac{1}{[1+\exp(w)][1+\exp(-w)]} = 0$$

therefore, the probabilistic heatmap may become a bottleneck that prevents the gradients from flowing backward, which causes the patch classifier layers to train slowly.