Detecting Breast Cancer in screening Mammography images

Rajneesh Tiwari
Georgia Institute of Technology
rtiwari37@gatech.edu

Leena Kim
Georgia Institute of Technology
lkim93@gatech.edu

Abstract

One of the most prevalent cancers worldwide is breast cancer. Reducing breast cancer deaths requires early detection and treatment. Presently, screening mammography systems are expensive to operate since early diagnosis of breast cancer necessitates the expertise of highly trained human observers. With the looming shortage of radiologists, it becomes imperative to apply automated Machine Learning methods to enhance detection and improve access.

To this end, we conduct multiple experiments on breast cancer mammograms to establish a robust baseline of performance and propose methods that work well with this dataset.

In this report, we leverage datasets from Kaggle "RSNA Screening Mammography Breast Cancer Detection" competition and document our experiments, comprising experimental setup parameters such as image generation methods for JPEG 2000 format, image augmentations, CNN vs Transformer architectures, optimizers, single vs multi view models etc. We also compare our results to existing state-of-the-art methods in this domain.

1. Introduction, Background and Motivation

Cancer is life threatening disease with estimated 10 million deaths and more than 19 million cases reported worldwide in 2020 [1]. For women, Breast Cancer is fifth most fatal reason of death with almost 685,000 deaths and 2.3 million new cases in 2020 alone [1].

A vast majority of breast masses are benign, or non-cancerous, that typically lead to fibroids, pain, region thickening, or lumps. Similarly, most breast tumors show no sign when their size is small and can be easily treated if detected early in the stages.

Factors that typically lead to breast cancer include family history, excessive body weight, diet, alcohol, smoking, environmental factors, and other risk factors such as night employment. Breast cancer initially spreads slowly, but it eventually affects other bodily parts. [2]

There are multiple diagnosis routes for detecting breast tumors, these include breast mammography, magnetic resonance imaging (MRI), ultrasound etc. Mammography remains the most common diagnosis method especially at early stages of detection [2]. A mammogram is an X-ray picture of a Breast, and it is used by doctors to identify early signs of breast cancer. It can be used to detect breast cancer up to 3 years before it can be felt [3].

Radiologists typically use Computer Aided Detection (CAD) to understand mammography images. This involves using handcrafted features to identify regions that appear distinct from the normal tissue. Then the radiologists decide whether these areas might be cancerous or not.

Recent advances in Deep Learning based methods typically can improve upon any handcrafted features for tasks such as image classification, object detection etc. This makes Deep Learning algorithms such as CNNs and Transformers natural candidates that can do well in Breast Cancer Detection using mammography images. To this end, applying recent advances in Deep Learning is essential to automate breast cancer detection and can significantly improve detection rates and lower the requirements of highly trained radiologists.

Over the years, many deep learning-based approaches have been applied to Breast Cancer Mammography images. In 2019, Dina A. Ragab et al. [4] applied Deep CNN model AlexNet model with a SVM head to predict benign vs cancerous images. They achieve an impressive AUC of 0.94 and accuracy of 87.2%. [4]

Another approach proposed by Li Shen et al. [5] leverages ROI heatmaps and patch classifier models to construct an 'end-to-end' deep learning approach. The key idea here is to pretrain an image detection model on publicly available breast cancer datasets within ROI annotations. Further, they first construct a patch image classifier using initialization from detection model, and then this model is finetuned on whole image classification tasks. They leverage VGG and ResNet models for their experiments. On the CBIS-DDSM dataset, they report an AUC of 0.90 and on the INbreast dataset an AUC of 0.98.

In this report, our main contribution is exploring various image preprocessing techniques, network architectures, optimizers, auxiliary task modelling, LR schedulers, image augmentations and a novel self-distillation network. Further, we explore the impact of image conversion (JPEG 2000 to PNG) on the model performance. Along with this,

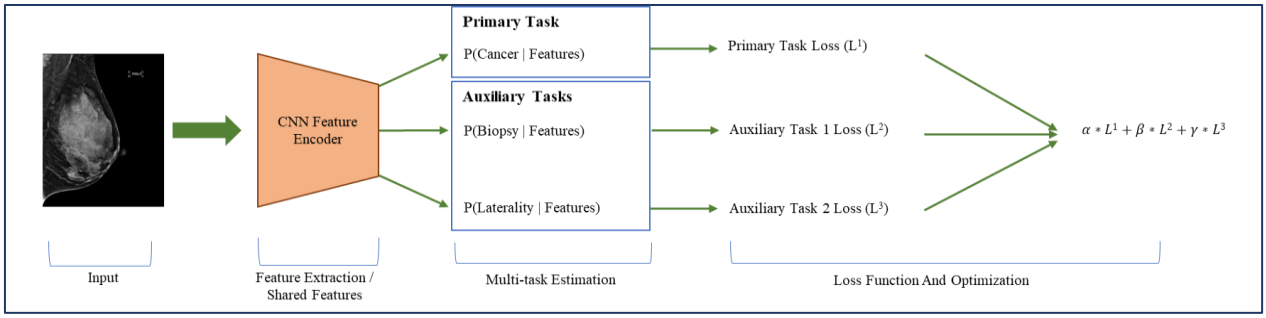

Figure 1: Architecture for MTL Model

we explore impact of single view vs multi-view deep learning models.

1.1. Dataset Details

We use dataset from Kaggle "RSNA Screening Mammography Breast Cancer Detection" competition [6]. We are provided with mammogram DICOM images. There are roughly 12,000 patients supplying 54,706 images in the training dataset. The dataset primarily belongs to North American population base and hence might not reflect the screening patterns in other geographies.

The dataset consists of patient_ids and their corresponding cancer diagnosis label. It also contains metadata such as site id where diagnosis was conducted, id of the machine, image laterality, image vie, age of the patient, breast implant indicator, biopsy indicator and a few others.

Patient_id	Laterality	View	Age	Cancer
10006	0	1	61	0
10006	0	5	61	0
10006	1	5	61	0
10011	0	1	59	1

Table 1. Sample Snapshot of Tabular Metadata

As opposed to most open-source datasets such as INbreast or CBIS-DDSM, this dataset is highly imbalanced with only 2% of images being cancer positive. This implies specialized handling of minority class and choice of loss function could be key. Furthermore, the average age of a cancerous patient (63 years) is higher than the average age of non-cancerous patient (58 years).

2. Approach

We utilize many standard image classification approaches to solve this classification problem. Specifically, we design a robust baseline and overlay improvements on top of this baseline. Later in our experiments, we break away from this baseline++ paradigm and try altogether new architectures such as Multi-task learning networks (MTL) [7] and self-distillation networks [8].

The key idea behind MTL [7] was to effectively utilize the tabular metadata as auxiliary labels to improve the overall model performance. We felt that the metadata held some very important details about the patient, and simultaneously learning to predict these parameters along with the cancer label could help the model perform better on the main task.

Further, we also observed that our baseline++ models had sufficient model capacity but were not showing drastic improvements while training. Our hypothesis is that the lack of positive labels was hampering the model's performance. Hence, we wanted to be more aggressive with our training and designed a self-distillation model [8] to force the model to learn effective features as quickly as possible.

We anticipated a variety of issues stemming from the lack of positive class labels and lossy image compression while converting DICOM to PNGs. Specifically, we hypothesized that our models will not learn effective class discrimination if we did not have specific protocols to address it.

We were able to take multiple steps to counter class imbalance, as our experiments will show gradual but significant improvements as we move from baseline to other implementations. In fact, even our baseline model worked reasonably well considering the class imbalance in the dataset.

To address the lossy image compression issue, we thought of utilizing 16-bit images (instead of 8 bit images) in our training, but were unable to do so as the pipeline became compute heavy to allow for effective training on our GPU hardware.

Our approaches hence focus more on finding better training paradigms and not on improving input image quality.

2.1. DL Framework & Pretrained Models

We used Pytorch as Deep learning framework of choice for this study.

Most of our pretrained models were sourced from the

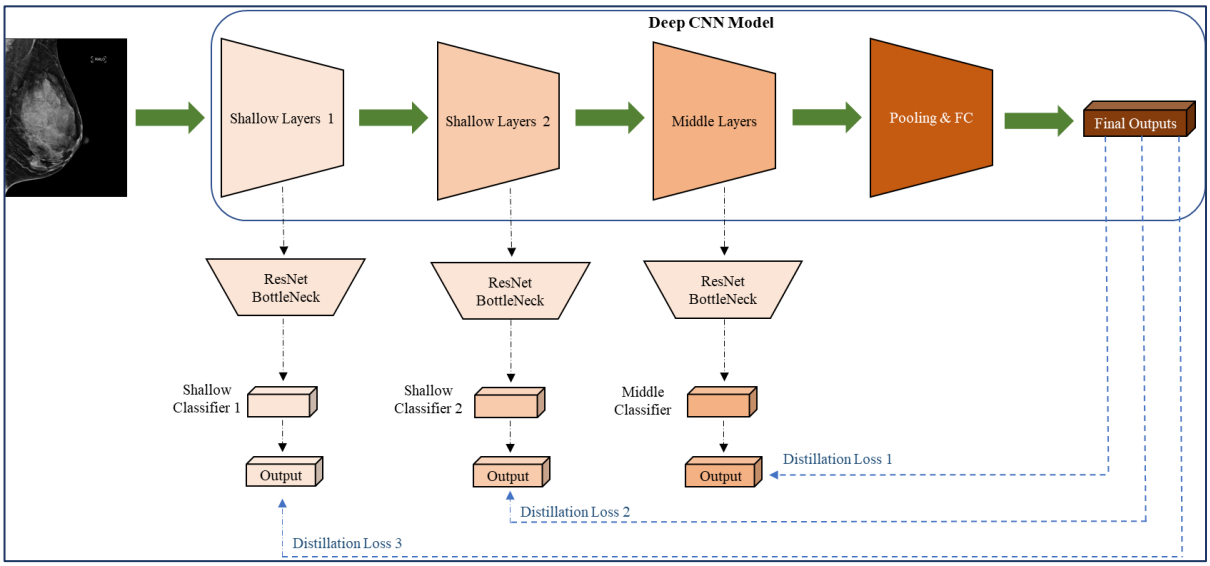

Figure 2: Architecture for Self-Distillation Model

robust timm [10] library. For the most part, we leveraged the models in a standard way, i.e., attaching a classifier head in-place of the pretrained classifier head.

For self-distillation models, we utilize specific feature-map hooks provided in the library utils to capture feature maps at varying depths.

We also tried leveraging the recently released transformer based NextVit [11] model from ByteDance but were unable to train it due to lack of compute resources.

2.2. Pre-processing: DICOM to PNG

We are provided DICOM images from the Kaggle competition. There were 2 types of DICOM images that were provided, j2k format and non-j2k file format.

For j2k file format, we use pydicom library to read the files and nvjpeg2k to decode it. For the non-j2k file format, we use dicomsdl library to process the file.

Since the raw images are typically high resolution in order of 4000 X 5000, we resized all the images to smaller sizes. In essence, we reduced all image sizes to $1/4^{th}$ the original to keep the same aspect ratio while keeping the image sizes manageable for training neural networks. Any further resizing is done as part of augmentation pipeline.

We also tried experimenting with two different image sizes formats - 8-bit PNG and 16-bit PNG. However, the 16-bit version was taking too long to train, hence, we dropped the idea of using it. We think that training with 16-bit images would yield even better results as 16 bit is less lossy vs 8-bit images.

2.3. Validation Setup

The data contains multiple images from the same patient, hence, it is essential to have all same patients in the same folds and prevent patient level leakage while setting up the validation set.

For this, our validation setup is based on StratifiedGroupKFold where the patient_id is the group variable, and we stratify against the target variable as well to account for class imbalance. For most experiments we use 4-fold setup to report out of fold performance.

2.4. Metric

The competition uses probabilistic F1 score as the evaluation metric of choice. This metric is an extension of the traditional F-score that accepts probabilities instead of binary classifications.

$$pF_1 = 2\frac{pPrecision \cdot pRecall}{pPrecision + pRecall}$$

where:

$$pPrecision = \frac{pTP}{pTP + pFP}$$

$$pRecall = \frac{pTP}{TP + FN}$$

2.5. Multi-task learning

To make models more robust to input noise, class imbalance, data efficient and perform better on primary task, we apply the concept of multi-task learning (MTL) [7] to the problem at hand.

Specifically, we utilize the multiple image level auxiliary fields as depicted in Table 1. to formulate auxiliary target labels such as Laterality, Biopsy, Implant etc.

Fig 1 shows the end-to-end architecture of such a Multitask Network formulation. We explore multiple variations of auxiliary targets and loss function combinations.

The feature encoder in the MTL is a typical Convolutional Neural Network such as EfficientNet, ResNet etc; this encoder block is shared across all tasks. The extracted features are fed to task specific classification/regression heads, which then make predictions for each task.

2.6. Self-Distillation

To further boost the baseline model's performance, we leverage the key idea of self-distillation in a neural network [8].

Self-distillation method is different from conventional knowledge distillation methods where the knowledge of a larger teacher model is transferred to a small student model, self-distillation method enables knowledge transfer within the model, from more complex deeper layers to simpler shallow layers. [8]

Self-distillation typically achieves overall better model performance when compared to other distillation methods. The key idea in self-distillation is to push the initial layers to learn as complex representations as the later layers, this makes the overall network much stronger.

Our implementation of self-distillation also introduces utilizing BottleNeck blocks from ResNet [9], this leads to better overall performance and helps with overfitting.

Figure 2 depicts the network architecture for a self-distilling model.

2.7. Augmentations

Image Augmentation is a very effective technique to improve generalization capabilities of deep learning models. We also employ and explore multiple augmentations during this study. We use existing augmentations in Albumentations [12] package for these purposes.

Typically, we used affine transformations such as shift, scale, rotate, flip along with others such as random brightness, random contrast, and optical/grid distortions.

3. Training Details

3.1. Crop ROI & Pectoral Muscle Removal

Across most images in this project, it was observed that a large area consists of black pixels with no pertinent information present in those. We wanted to crop out the exact breast area before feeding to the model as this would ensure less redundancy and would allow us to use even larger image sizes.

To do this, we leverage OpenCV library to find regions of non-empty pixels, then we extract out the region with highest area which in almost all cases corresponded to breast area.

Lastly, we apply series of OpenCV processing operations including canny edge detection, Sobel filters, Hough line detection to cut out the pectoral muscle area.

3.2. Image Normalization

As a standard training process, all image inputs to the models were normalized with zero mean and unit variance to enable stable training.

3.3. Loss Functions

For all binary classification tasks (i.e. primary or auxiliary tasks or distillation tasks), Binary Cross Entropy with logits loss was used. For all multi-class classification tasks cross entropy loss was used.

For the Multitask model, all losses (primary vs average auxiliary loss) are weighted in 4:1 ratio in favor of primary target loss.

For self-distillation model, all distillation losses are Binary Cross Entropy with logits loss with equal weights to all components.

3.4. Mixed Precision/FP 16 Training/Inference

Since we only had access to a single 4090 GPU/Colab Environment and the training data was huge, we leveraged fp16 training regimen to be able to train with larger model, larger batch sizes, and larger image sizes. Mixed precision effectively allowed us to increase our batch size by a factor of 2 in most cases.

3.5. Hardware

All the training and inference was done on either a NVIDIA 4090 GPU or Google Colab.

3.6. Code Repository

All the data preparation, training and inference codes are provided at the following Github Link - https://github.com/Rajneesh-Tiwari/CS-7643-Deep-Learning

4. Experiments and Results

We conducted a total of 8 experiments starting with a simple classification baseline. Experiments were designed to probe effects of parameters such as larger image sizes, larger models, augmentations, pre-processing, multi-task methods, and self-distillation models.

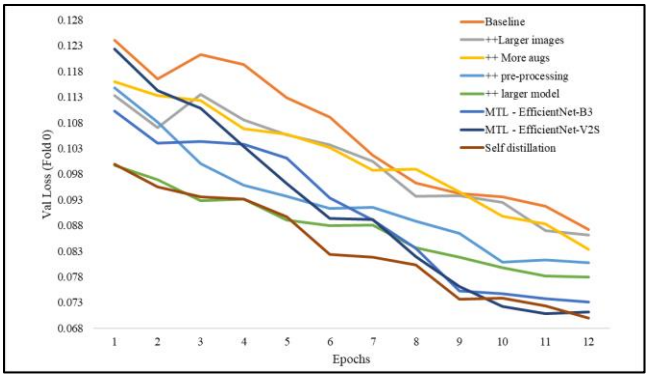

Figure 3: Validation BCE Loss for Baseline Derived Models (Fold 0 only)

Model	Train Loss	Val Loss	pf1 Metric
Baseline	0.070	0.087	0.2639
++ Larger images	0.064	0.086	0.3057
++ Extra augmentations	0.066	0.083	0.3291
++ Pre-processing	0.063	0.081	0.3353
++ Larger model	0.062	0.078	0.3814
Multitask - EfficientNet-B3	1.090	0.073	0.3934
Multitask - EfficientNet-V2S	1.227	0.071	0.3985
Self-Distillation - EfficientNet-V2S	0.305	0.070	0.4189

Table 2: Training & Validation loss for Fold 0; pf1 metric is calculated across all 4 folds. Train loss for MLT and Self Distillation are not directly comparable to Baseline numbers.

4.1. Baseline

We started with a simple classification baseline of EfficientNet B4 model. We choose EfficientNet model as starting point since it provides the best balance between performance and model complexity on industry benchmark datasets such as ImageNet and CIFAR [10].

In the baseline model we use 1024 X 1024 size input. We use minimal augmentations i.e., ShiftScaleRotate and Horizontal Flips only.

We use a CosineAnnealing LR schedular with starting LR of 4e-4. The optimizer used was lookahead-RAdam.

As provided in Table 2 the baseline model secures a 4-fold pf1 score of 0.264 and provides a great starting point for comparison. This baseline is heavily overfitted by examining the training vs validation BCE loss.

4.2. Baseline w/ larger images (++ Larger images)

Building on top of the EfficientNet-B4 baseline, we evaluate the same pipeline on a larger sized image i.e., 1536 X 960. The height and width are chosen to keep the aspect ratio like that of original DICOM images.

This was done to examine the impact of image sizes on the outputs. Further, as noted earlier, the input PNG images are heavily downscaled and lossy compressed version of original DICOM images, hence, it was intuitive to use the largest possible size subject to hardware constraints.

As provided in Table 2, the baseline w/ larger images model secures a 4-fold pf1 score of 0.305. This proves our hypothesis that larger images will have substantially better models. We also observe that this model was overfitted wrt. BCE loss (Fig. 3).

4.3. ++ Larger images w/ extra augmentations, pre-processing, and larger models

To address the issue of overfitting in the previous models, we increase the augmentations to our training cycles. We keep the original augmentations and include new ones such as Vertical Flip, Optical Distortion/Grid Distortion, Affine, Coarse Dropout, and Cutout. As a result, we see significant improvement in model's generalization ability to the validation set and it scores a pf1 of 0.329 (Table 2.) on the out of fold validation set.

Further, in the original images, there are a lot of black areas surrounding the actual breast area. These black areas provide no useful information to the model and end up only taking up space. We leverage OpenCV to crop out breast ROI, pectoral muscles and resize to 1536 X 960. This version of model scores a pf1 of 0.335 (Table 2) on the out of fold validation set.

Lastly, larger models work better than smaller models as they have higher model complexity. This is especially true in large data paradigms such as the one we are solving.

We replaced the baseline EfficientNet-B4 model with EfficientNet-B6 model. This gave us our best model so far with an OOF pf1 of 0.381 (Table 2).

All models in this section were trained at 1536 X 960 image size, LR was set 4e-4, with Weight Decay of 1e-8. LookAhead-RAdam optimizer with Cosine-Annealing LR Scheduler was used. We trained these models for 12 epochs.

4.4. Multi-task models

We utilize the other metadata features Laterality, Biopsy, View, and Implant to create auxiliary targets. We exploit the general idea that multitask models typically train better than single task models and can exploit relationship between features.

We use minimal augmentations and train on image sizes 1536 X 960. Other hyperparameters such as Learning Rate is set at 4e-4, weight decay at 1e-8. Optimization used LookAhead-RAdam optimizer with OneCycle LR Scheduler. We trained these models for 12 epochs.

Using EfficientNet-B3 as backbone MTL model achieves validation pf1 of 0.394, while with EfficientNet-V2S as backbone it achieves slightly higher validation pf1 of 0.398.

All the input features except View are binary. For all binary features Pytorch's BCE With Logists Loss was used while for View feature Pytorch's Cross Entropy loss was used.

The loss for backpropagation is calculated as below:

$$L_{aux} = (L_{Laterality} + L_{Biopsy} + L_{View} + L_{Impant}) /4$$
$$L_{Final} = (1 - \alpha) * L_{aux} + \alpha * L_{Cancer})$$
$$Where\ \alpha\ was\ set\ =\ 0.8$$

4.5. Self-distillation Models

We implemented a novel self-distillation model by utilizing a pretrained model and extracting feature maps at various depths of the model. In this implementation, we extract feature maps from 2 hooks within the model.

These feature maps are then passed through ResNet bottleneck, pooling, Dropout layers before being fed to the fully connected layer which makes auxiliary predictions. The distillation losses are calculated based on these auxiliary predictions. To avoid overfitting, we apply regular dropouts to the feature map outputs. The loss for this model is calculated as below:

$$L_{distilled_1} = L(DistilledLogit1, target)$$
$$L_{distilled_2} = L(DistilledLogit2, target)$$
$$L_{final} = L_{Cancer} + L_{distilled1} + L_{distilled_2}$$

The idea was to improve the model quality by forcing the first few layers to be as effective as deeper layers.

We use high-augmentations settings, train on image sizes 1536 X 960, learning Rate at 1e-4, weight decay at 1e-6. Optimization used Adam optimizer with OneCycle Scheduler. We trained this model for 12 epochs.

This model with an EfficientNet-V2S backbone has the best performance amongst all others with a validation pf1 of 0.419.

4.6. Comparison with Benchmarks

The benchmark datasets such as INBreast and CBIS-DDSM are significantly different from the dataset used in this study. This dataset has significantly higher-class imbalance and is based only on North American population sample.

However, we report that our best model on this dataset provides AUC ~0.85. On the benchmark datasets, the best models have AUC ~0.9+.

Even though it's not a like-like comparison, we believe our models would be able to achieve AUC of 0.9+ on INBreast/CBIS-DDSM if we trained and finetuned on those datasets. This requires a separate study and is not part of this report.

4.7. Project Success

We planned on measuring success based on quality of our ideas, implementation, and rigor in experimentation. We couldn't do all the experiments we thought of due to computation limitations (eg: NextVit Model, Transformers). However, by setting up and executing complex ideas such as DICOM to PNG conversion, Mixed Precision/FP16 training and inference, MLT and self-distillation methods, we feel we have successfully achieved what we wanted to.

4.8. Summary of Contributions

All team members contributed equally to this project. The below table (Table 3.) depicts the specific work division and summarizes individual's contribution to this project:

Student	Contributed Aspects	Details
Rajneesh Tiwari (rtiwari37@gatech.edu)	1. Implemented entire data preparation (DICOM-PNG) pipeline from scratch. 2. Implemented pectoral muscle removal algorithm (see: baseline_larger_images_moreaugs_preprocessing.ipynb) 3. Implemented multitask and self-distill model codes from scratch. 4. This includes dataloaders, loss functions, training & inference code. 5. Tuned parameters for weights of loss functions in MTL and self-distill models. 6. Tuned augmentations, LR, Optimizers, Schedulers	Implemented the following notebooks: 1. rsna2022-resize-8bit.ipynb 2. Multitask_efficientnet_b3.ipynb 3. Multitask_efficientnetv2_s.ipynb 4. self_distill_tf_efficientnetv2_s.ipynb
Leena Kim (lkim93@gatech.edu)	1. Implemented data loader, loss function, mixed precision training, mixed precision validation, and inference codes in baseline, and 4 other baseline derived models. 2. Identified overfitting in baseline and added augmentation experiments to improve performance. 3. Identified ways to improve inputs with black area removal pre-processing (see: baseline_larger_images_moreaugs_preprocessing.ipynb) 4. Tuned hyperparameters such as image sizes, augmentations, LR, Schedulers, Optimizer.	Implemented the following notebooks: 1. baseline.ipynb, 2. baseline_larger_images_moreaugs.ipynb, 3. baseline_larger_images_moreaugs_preprocessing.ipynb, 4. improved_baseline_larger_images_moreaugs_preprocessing.ipynb

Table 3: Contributions of team members.

References

[1] Hyuna Sung, Jacques Ferlay, ME, Rebecca L. Siegel, Mathieu Laversanne, Isabelle Soerjomataram, Ahmedin Jemal, Freddie Bray. Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. CA Cancer J. Clin. 2021;71:209–249. doi: 10.3322/caac.21660

[2] https://www.cancer.org/content/dam/cancer-org/research

[3] https://www.cdc.gov/cancer/breast/basic_info/mammograms.htm

[4] Dina A. Ragab, Maha Sharkas, Stephen Marshall, Jinchang Ren. Breast cancer detection using deep convolutional neural networks and support vector machines. PeerJ. 2019;7:e6201. doi: 10.7717/peerj.6201

[5] Li Shen, Laurie R. Margolies, Joseph H. Rothstein, Eugene Fluder, Russell McBride, Weiva Sieh. Deep learning to improve breast cancer detection on screening mammography. Sci. Rep. 2019;9:1–12. doi: 10.1038/s41598-019-48995-4

[6] https://www.kaggle.com/competitions/rsna-breast-cancer-detection/overview

[7] Michael Crawshaw. Multi-Task Learning with Deep Neural Networks: A Survey. arXiv preprint arXiv:2009.09796, 2020.

[8] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-Distillation: Towards Efficient and Compact Neural Networks. In IEEE Transactions on Pattern Analysis And Machine Intelligence, VOL. 44, NO. 8, AUGUST 2022.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[10] Ross Wightman. PyTorch Image Models, 2019. doi: 10.5281/zenodo.4414861

[11] Li, Jiashi and Xia, Xin and Li, Wei and Li, Huixia and Wang, Xing and Xiao, Xuefeng and Wang, Rui and Zheng, Min and Pan, Xin. Next-ViT: Next Generation Vision Transformer for Efficient Deployment in Realistic Industrial Scenarios. arXiv preprint arXiv: arXiv:2207.05501, 2022

[12] Buslaev, Alexander and Iglovikov, Vladimir I. and Khvedchenya, Eugene and Parinov, Alex and Druzhinin, Mikhail and Kalinin, Alexandr A. Albumentations: Fast and Flexible Image Augmentations, 2020. doi: 10.3390/info11020125