Medical Imaging and Applications (MAIA)
E-Health Final Project
Image Preparation for Machine Learning

January 11, 2018

Vu Hoang Minh
Yeman Brhane Hagos

1 INTRODUCTION

Machine Learning applies machine learning algorithms to learn from data. It is critical that you feed them the right data for the problem you want to solve. Even if you have good data, you need to make sure that it is on a useful scale, format and even that essential features are included. To get a better result, it is recommended to look through your data, see the quality and quantity of data, work to improve the quality of images and/or apply augmentation to increase your dataset or remove unwanted images. The process for getting data ready for a machine learning algorithm can be summarized in three steps [1]:

- Data collection and selection
- Image preprocessing
- Image augmentation

1.1 Data collection and selection

This step is about selecting a subset of all available images that you will work with. We have a strong desire to include all available data to train our algorithm but, this might not always be true. We have to consider data that are needed to tackle our problem. Therefore, you have to know what data you have, what data is not available that you want to have it, exclude the data that you do not need to address the problem under consideration.

1.2 Image preprocessing

After selecting the appropriate images, we need to think how we are going to use our data with the proposed technique. The type of the preprocessing to be applied strongly depends on the type of the data and the machine learning tool you chose. The most commonly applied image preprocessing in machine learning and specifically to deep learning includes image contract enhancement, sampling and image denoising. The data already collected might not be in the format that is suitable for your machine learning algorithm. Here format refers to the data type of the images or image normalization. Medical imaging devices are susceptible to noise [2]. Image denoising is the process to remove the noise from the image naturally corrupted by the noise. In addition to that, there may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. Thus you need to sample some more appropriated data from the images.

Another most important steps in deep learning and other machine learning are scaling and patch generation. The gathered data might have different size, so it is needed to scale to the same shape before feeding to your learning algorithm. When we train with deep learning we have to consider memory consumption as we have limited GPU memory, and we have to use a batch of images to have a more generalized model. Thus, for 3D volume images and pathology images, it is not possible to train whole volume or whole slide with mini-batch size. In this case, we have to extract patches and apply reconstruction to get the final result. Moreover, extracting patches is used to increase training data as annotated medical image are scarce.

1.3 Image Augmentation

Image augmentation refers to applying deformation or transformation on the available dataset. By augmenting the images, you are training the network not to overfit your dataset with regards to the type of augmentation. For example, rotating an image in various angles will make the model to be invariant to rotation of the objects in the images. The most commonly applied augmentation techniques are zooming, scaling, rotation or adding random noise. So although new information is not added into the network, the synthetic data augmentation added into the network can both improve the results attained from the network and allow for training with less data, which is useful in medical application. If there are features extracted before applying augmenting the image and if the sensitive to the deformation applied, the same deformation should be applied.However, it is important to note that augmentation is only useful when semantically correct.

Although image preprocessing, patch extraction, and image contrast enhancement are determinant when we apply machine learning specifically deep learning in medical image processing and analysis, to the best of our knowledge, there is no a software for extracting patches, and applying image augmentation with different ground truth types(landmark file for example .xml files, ground truth or mask images). The main goal this project is to develop a software that contains the most commonly used image preprocessing and augmentation techniques to apply machine learning and deep learning in a medical image taking into consideration different ground truth types.

This report is organized as follows. In section two, we briefly explain datasets used to test our software. In section three we describe our software component and how to use the software. Result and discussion are presented in section four. Finally, project management and conclusion are presented.

2 DATASET

To test our software we have used two datasets. The first dataset is INBREAST mammogram dataset images [3]. This dataset contains 2D DICOM images of normal mammograms, and mammograms with masses, mammograms with calcifications, architectural distortions, asymmetries, and images with multiple findings. Ground truth of these findings is provided in Extensible Markup Language (XML) format with many information included. We have used the following information for patch extraction and image augmentation:

- Tag <key>NumberOfROIs</key> followed by an integer that indicates the number of annotations present in the image;
- Tag <key> Center</key> followed by <string>(x, y and z))</string>, a the coordinates of the point in the centre of the ROI(consider it if it is specified, however in the datasets given it is empty)

- Tag <key>Name</key> followed by the type of finding (mass, calcification, distortion, speculated region); for example, <string>Calcification</string>, this shows Calcification abnormality.
- For each ROI, a list of contour points is presented between the tags <array> and </array>.

We have also used IBSR18 dataset(from Medical Image Segmentation and Applications course project dataset) to test 3D, and 2.5D patches extraction. It is a collection of brain 3D MRI scans image dataset.

3 IMPLEMENTATION

3.1 Libraries Used

Medical images come in a different format, and we need different software and libraries to read and apply medical image processing and analysis. Thus, we have used many existing libraries to make our software able to read and processes different image formats. Our software is developed in python, and we have used the following libraries:

- **Pydicom:** It is pure python package for working with DICOM files such as medical images, and radiotherapy objects. We have used it to read DICOM files to our software and write DICOM file as an output.[`http://pydicom.readthedocs.io/en/stable/`]
- **OpenCV-Python:** This library was used in data augmentation to generate transformation matrix and apply the affine transformation on an image.
- **NiBabel:** It offers both high-level format-independent access to neuroimages, as well as an API with various levels of format-specific access to all available information in a particular file format. We have used it to read .nii images formats.
- **PyQt5:** Qt is a set of C++ libraries and development tools that includes platform independent abstractions for graphical user interfaces(GUI), and many other interesting features that are not needed in our software, and it it is the most commonly used application framework to develop cross-platform applications with GUI. PyQt5 is a comprehensive set of Python bindings for Qt v5. We have used PyQt5 to design our GUI.
- **Qimage2ndarray:** It is a python extension for conversion between QImages and numpy ndarrays (in both directions). We have used it for visualization to convert a 2D or 3D numpy array into QImage to display it in PyQt QLable object.

3.2 Implementation and Software Components

As can be seen in figure 1 the software contains five main components:

- Patch Extraction
- Image Augmentation
- Visualization

– Contrast Enhancement Techniques and

– Status bar

To make easy for debugging and integration of additional functionality, we have used python object-oriented programming, where implementation for patch extraction, image augmentation, and image contrast enhancement was done in a separate package and classes with methods to implement the different tasks. This makes our implementation modular and easily extensible. The status bar was used to visualize the status of a requested job, and the remaining components will be discussed in detail in next sections.

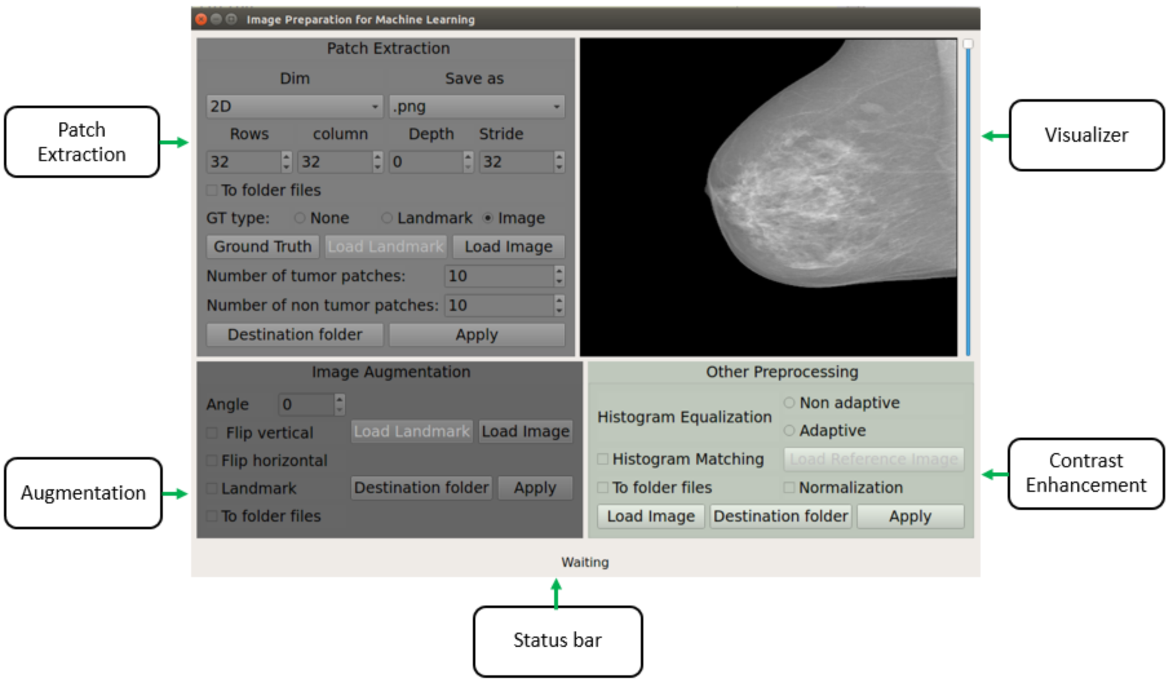

Dim: 2D
Save as: .png
Rows: 32
column: 32
Depth: 0
Stride: 32
GT type: None Landmark [✓ Image]
Number of tumor patches: 10
Number of non tumor patches: 10
Angle: 0
Histogram Equalization: Non adaptive / Adaptive
Waiting

Figure 1: Software components

3.2.1 Patch Extraction

Users can extract 2D, 2.5D or 3D patches. In addition to that user should specify the *row, column, depth*(for 3D patch)dimension, and *extraction step or stride*. The default values for row, column, depth, and stride are 32, 32, 0, and 32 respectively. It can be applied to a single file or to batch of files in a folder(*check to folder files checkbox*).If *To folder files* checkbox (figure 1) is checked, the buttons will allow you to select folders and patches will be extracted from all images in the folder. The patch saving file format can be chosen by the user and the available image saving formats are $f.png$ and *.npy* files. our software enables to extract patches in three different scenarios depending on the ground truth available:

1. When a user has only an image without any ground truth (GT type set to *None*); here, the user only need to upload an image and specify patch detail. Our software creates

a Region of Interest (ROI) image with threshold zero. This helps to make sure patches are extracted from the region which has meaningful information.

2. When user have a ground truth image(1 *GT type* set to *Image*); Here the user need to input image and a ground truth image. The image can be ground truth image or ROI image from which patches will be extracted. This helps to remove patches from background region only.

3. When the user have landmarks as a ground truth *(1 GT type set to Landmark)*; the landmarks should be in Extensible Markup Language (XML) format, and should follow the format of [3]. The annotations were saved in XML format with a well-defined structure and header information. To extract the patches contour points of the recorded abnormalities were fetched from the XML file and to center point was computed by averaging the x and y coordinate of the points. To read, process/or modify information from the XML file we have used ElementTree XML API.

   In addition to patch details, the user should also input the number of tumors and nontumor patches that he/she wants to extract. If the number of landmarks is larger than the requested number of patches, random selection is applied. In case the user request more patches than available landmarks, it will provide only patches extracted from the center of the contours available.

When the available ground truth is landmark, users might be also interested in extracting from the normal part of the image. Center points for normal patches (patches without any abnormalities) was computed using equation (1).

$$\underset{p}{\operatorname{argmax}} \, \|\mathbf{L} - \mathbf{P}\|_2 \tag{1}$$

where **P** is a point inside ROI part of the image and L is a set of landmark center points computed above to extract abnormal patches. Therefore a point is selected as a center of normal patches point if the Euclidean distance from the point to **L** is greater than a threshold, and from the available point, points with maximum distance will be selected. The threshold value was set to longest patch dimension. Possible improvement can be done by including the shape and the area of the abnormalities when selecting non-tumor patches.

3.2.2 Image Augmentation

The need of image augmentation was explained in the introduction part. In our software, we have included rotation and flipping as augmentation techniques. In some case in addition to images, we might have files which contain landmark points in the image on which the same transformation should be applied. The overall workflow of image augmentation through rotation is displayed in figure 2. Initially, the rotation angle is taken from which transformation(**T**) matrix is computed. Then, the image is transformed by applying T on it. If the image has a landmark associated with it, the same transformation will be applied to numeric data that will be affected upon application of the transformation. For example, if there are a center location of tumors or contour points, the same transformation will be applied to them. To generate the transformation, we have used ***cv2.getRotationMatrix2D(...)*** form OpenCV library, however, this module it will cut, the image after rotation. To solve this problem, we have computed the new dimension of the output image as a function of input image dimension and rotation angle. We have also modified the translation part of the transformation matrix generated to take in to account the translation by adding the difference between new center and input image center coordinate points.

If *To folder files* checkbox (figure 1) is checked, the button will allow you to select folders and the selected augmentation will be applied to all images in the folders. To apply augmentation for images and landmarks, the images and landmarks should be kept in a separate directory. The user should also provide destination folder, to save augmented image. To avoid overwriting, images when user choose more than one augmentation type, the name of the augmented image will be followed by the angle of rotation.

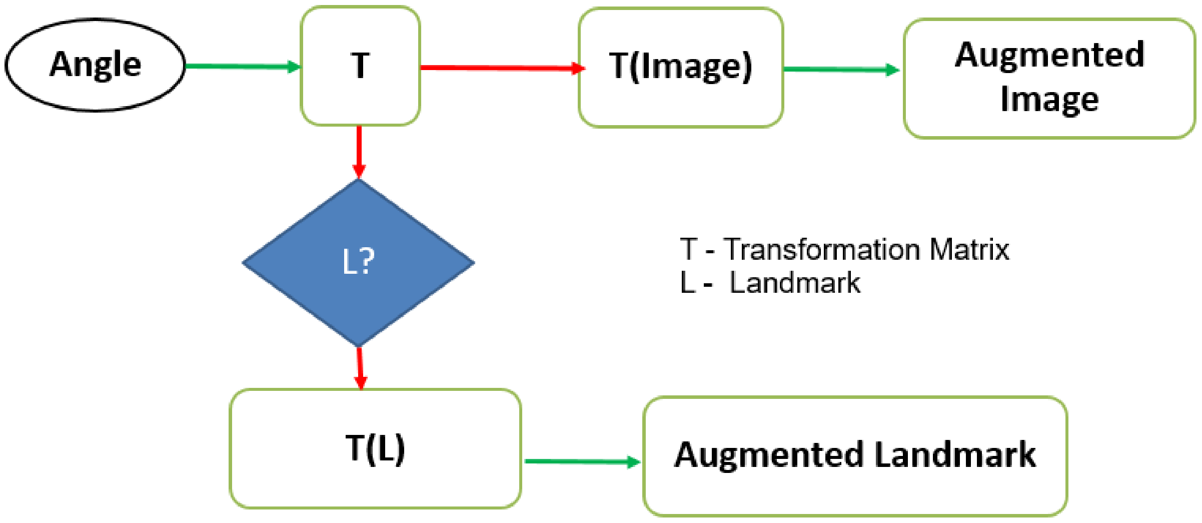


Figure 2: Image and landmark augmentation

3.2.3 Visualization

QLabl widget of PyQt5 was used to visualize images. As we can not directly visualize a numpy array, we used Qimage2ndarray library to convert it to QImages, a format suitable for QLabel. Users can visualize 2D and 3D images. For 3D images, the visualization is done slice by slice, and when the volume is loaded, the central slice is displayed by default. The other slice can be visualized by changing the slider value which is found the left side of the visualizer as can be seen in figure 1. During visualization, the size of the image will be resized to QLabl size.

3.2.4 Contrast Enhancement Techniques

Contrast enhancement methods are extensively used as preprocessing when we apply deep learning on medical image segmentation, classification, and registration. In our software, we have included and non-adaptive histogram equalization, histogram matching, and normalization. During histogram matching, we need a template image, and we have provided a separate button for it. If *To folder files* checkbox (other pre-processing section figure 1) is checked, the button will allow you to select folders, and the selected techniques will be applied to all images in the folders. You have to provide also destination folder to save preprocessed images. You can apply simultaneously all methods, and the file name will be the original file name followed by the type of pre-processing applied.

4 Result and Discussion

In this section, sample results generated will be discussed for the main components of the software; patch extraction, image augmentation and image contrast enhancement.

Figure 3 shows an input image with Calcification and Mass abnormalities and extracted patches from our software. The extracted patches are grouped according to their type of abnormalities, and normal patches are extracted outside the abnormal area of the mammogram. Normally, if a folder which contains images and/or ground truth was selected, a separate folder will be created for each image in the folder, and for each type of patches.

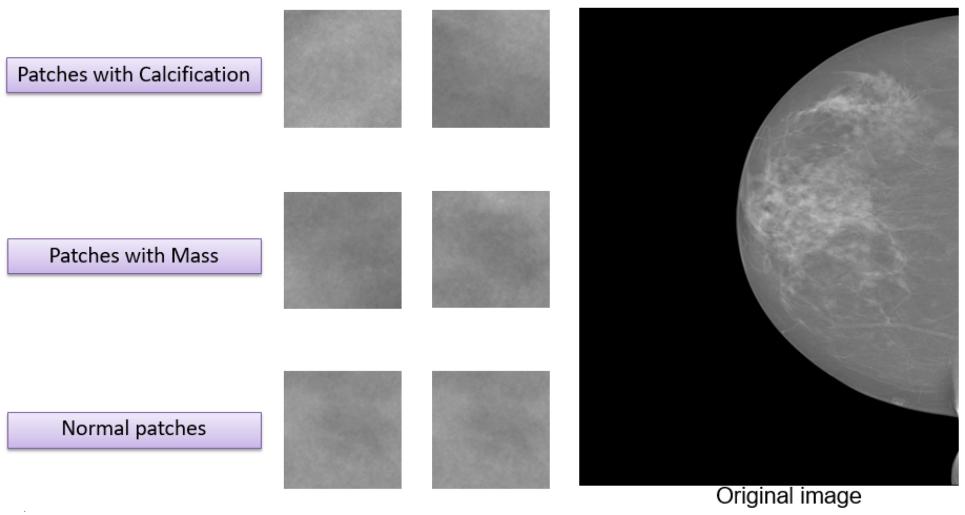


Figure 3: Patch extraction from an image with Calcification, Mass abnormalities

Sample images obtained by applying image contrast enhancement are depicted in figure 4. The figure contains input images, template image for histogram matching, and output images obtained using histogram matching and histogram equalization. However, histogram equalization and matching results do not look suitable for this type of images.

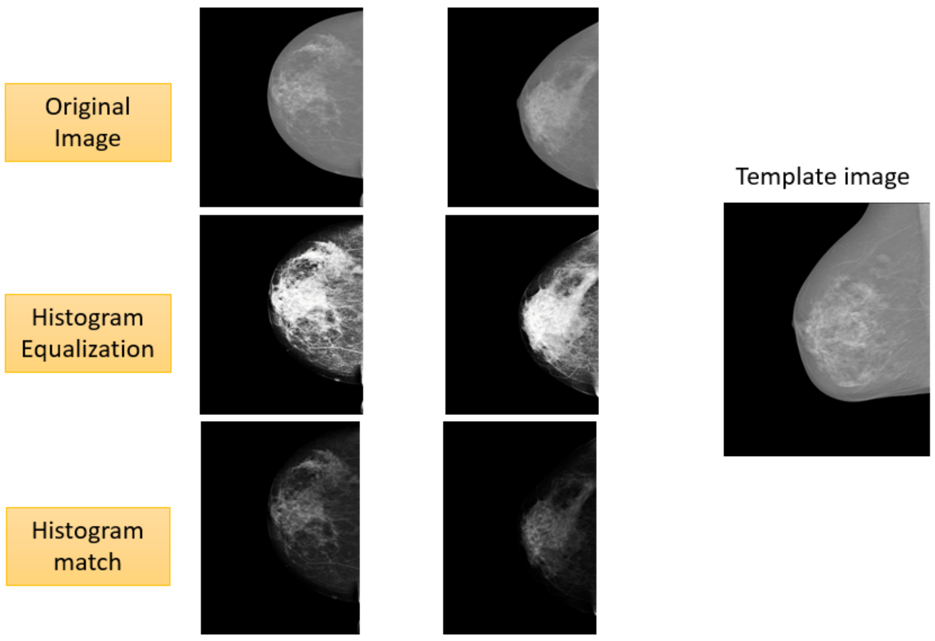
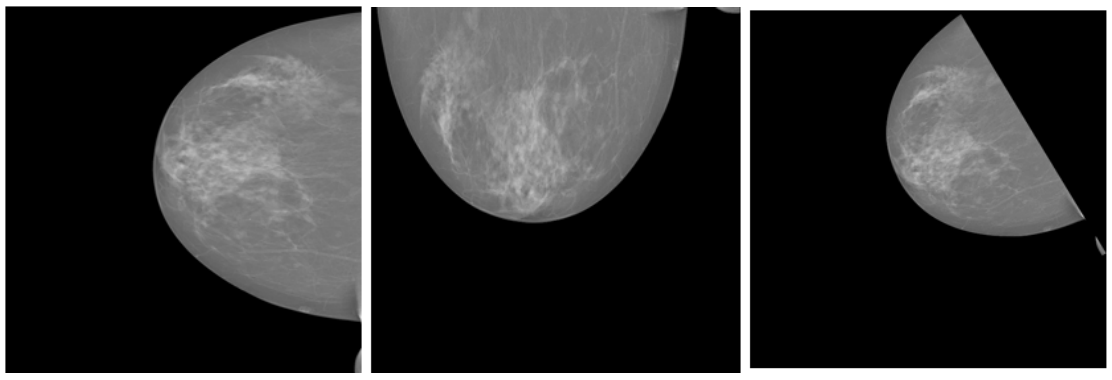


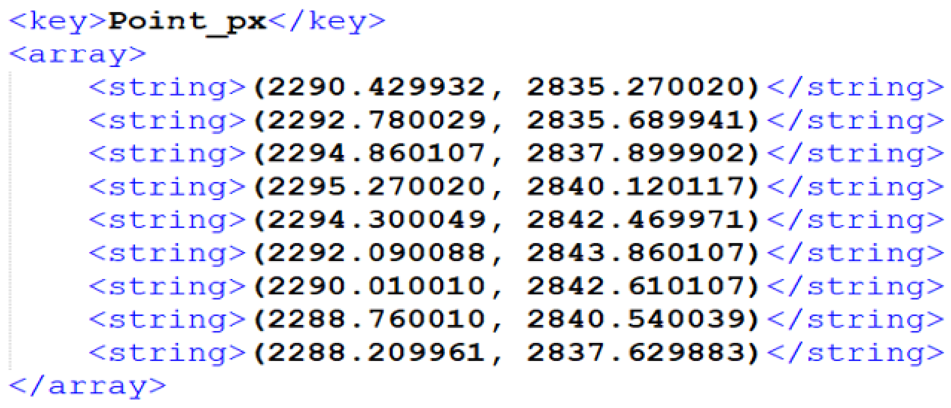
Figure 4: Contrast enhancement techniques



(a) Original image (a) Rotated by 90° (a) Rotated 30°

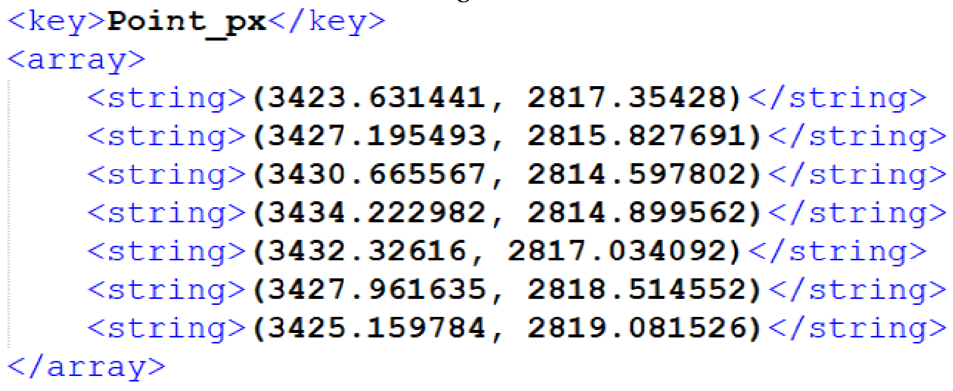
Figure 5: Input and augmented images

Our software also has image augmentation module; an image augmentation can be applied to images and landmarks from an XML file. Figure 5 presents input, and rotated images. The first image shows a rotated image, while (b) and (c) shows images rotated by 90° and 30° respectively. Augmentation was applied to both image and landmark points in an input XML file. Contour point of a sample Calcification abnormality in mammogram image before and after transformation applied are depicted in figure 6. Figure 6a shows original contour points, while 6b is rotated contour points with rotation angle of 30°. Users can extract patches or any features from the augmented images and XML file. Thus we need to apply rotation to any fields in the XML file that are not invariant to rotation. A field like an area is not affected by the augmentation as we are applying simple rotation and flipping while the central location and contour location are transformed according to the input angle from the user.



```xml
<key>Point_px</key>
<array>
    <string>(2290.429932, 2835.270020)</string>
    <string>(2292.780029, 2835.689941)</string>
    <string>(2294.860107, 2837.899902)</string>
    <string>(2295.270020, 2840.120117)</string>
    <string>(2294.300049, 2842.469971)</string>
    <string>(2292.090088, 2843.860107)</string>
    <string>(2290.010010, 2842.610107)</string>
    <string>(2288.760010, 2840.540039)</string>
    <string>(2288.209961, 2837.629883)</string>
</array>
```

(a) Original XML



```xml
<key>Point_px</key>
<array>
    <string>(3423.631441, 2817.35428)</string>
    <string>(3427.195493, 2815.827691)</string>
    <string>(3430.665567, 2814.597802)</string>
    <string>(3434.222982, 2814.899562)</string>
    <string>(3432.32616, 2817.034092)</string>
    <string>(3427.961635, 2818.514552)</string>
    <string>(3425.159784, 2819.081526)</string>
</array>
```

(a) XML file after rotation

Figure 6: Input and augmented XML using 30° augmentation angle

5 Project Management

As the selection of the project title was open, we have proposed a project titled *Image Preparation for Machine Learning* and we send a short description of it on December 7, 2017. After the the approval, we scheduled a proper plan to meet the deadline within the given time. The schedule of our meetings and an internal progress report is given in the following Table 1. All the deadlines were decided by mutually discussing and selecting the best approach scenario. An internal progress report was also maintained which was updated daily based on the work done or problem faced and that needs to be solved. We shared also a sharelatex to write report.

Table 1: Internal progress report

Name	**Work Assigned**	**Problems Faced**	**Next step**
		Week 1	
Both	Project topic selection	No	Waiting for approval
Both	Search for libraries	No	Decide GUI layout
		Week 2	
Yeman	Patch Extraction and Visualizer	Parsing XML files	Fix bugs
Minh	Image Augmentation	Apply augmentation for XML	Fix bugs
		Week 3	
Yeman	Contrast enhancement	No	Update report
Minh	Contrast enhancement	No	Update report
		Week 4	
Both	Prepare presentation, and report	No	Finalize codes
Both	Code commenting	No	Finalize report

6 Conclusion

We have developed a software that contains the most commonly used image preparation techniques to apply machine learning and deep learning for images using python programming language. The software contains patch extraction, image augmentation, and image contrast enhancement modules. The software is able to extract patches from images with ground truth (XML file of landmarks or mask or ROI image) and without ground truth, and apply augmentation for images and landmark points in an XML file.

The possible future work is to include more image augmentation techniques and other preprocessing methods like image denoising approaches.

References

[1] Jason Brownlee. How to prepare data for machine learning, 2013.

[2] Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*, pages 241–246. IEEE, 2016.

[3] Inês C Moreira, Igor Amaral, Inês Domingues, António Cardoso, Maria João Cardoso, and Jaime S Cardoso. Inbreast: toward a full-field digital mammographic database. *Academic radiology*, 19(2):236–248, 2012.