Differential Data Augmentation Techniques for Medical Imaging Classification Tasks (DDSM Mammography)

Ashish Vinodkumar
MIDS
Duke University
`ashish.vinodkumar@duke.edu`

Pranav Manjunath
MIDS
Duke University
`pranav.manjunath@duke.edu`

Tommy Tseng
MIDS
Duke University
`tommy.tseng@duke.edu`

Abstract

In this paper, we explore various data augmentation techniques such as Gaussian Blur, Gaussian Noise, Random Rotation, and Color Jitters, on the Digital Database for Screening Mammography (DDSM) dataset using a VGG-16 model architecture. We learn that data augmentation in general boosts model performance and reduces overfitting. Specifically, Gaussian Blur showcased to have the highest testing accuracy, and Random Rotation followed by Gaussian Blur had the highest Recall score. When plotting the ROC curves, Gaussian Blur had the highest AUC score when compared to the other models. As a result, our analysis showed that given we are dealing with Medical Imaging DDSM data, Gaussian Blur is the superior data augmentation technique across the model validation metrics.

1 Background

In the area of deep learning, data pre-processing is an important stage that ensures readiness of data for model training. Particularly in medical imaging applications, one of the important pre-processing techniques that is commonly applied is called data augmentation. Data augmentation is particularly helpful when the amount of images available is scarce and hard to be collected. By applying different transformations to the original images, data augmentation creates a more sufficient volume of training data, which helps to prevent model over-fitting to the limited training data.

For this project, the team aims to follow an earlier work on comparing different augmentation techniques, applied on a medical imaging dataset, by their testing accuracy and ROC curves.[1] Specifically, the team focuses on four augmentation techniques: Gaussian blur, Gaussian noise, Random rotations, Color jittering. The team shows that some augmentation techniques are able to extract medical image statistics more effectively than others, which will lead to better predictive performance in the corresponding models.

2 Method

2.1 Data Pre-processing

We obtained our dataset from Kaggle.[2] This dataset had the images in a compressed "tfrecords" format. The team explored various techniques to parse the numerous tfrecords files and we found one such approach that involved using tensorflow to open, parse, and convert the images into a numpy array from Ananta Raj.[3] Upon parsing the dataset as a numpy array, we were able to stratify the dataset into 1000 non-cancerous (negative) and 1000 cancerous (positive) images from the Digital Database for Screening Mammography (DDSM). These 2000 images were used in baseline training and testing purposes along with the data augmentation experiments. To be compatible with the model's input dimension, the original images were resized to 224 by 224 pixels. Finally, the dataset was split into training and testing set based on a 80-20 percent ratio. (1600 training, 400 testing images)

2.2 Model Architecture and Hyperparameters

We followed the study by Hussain et al. (2018), and used VGG-16 as our model architecture. This VGG-16 model architecture allowed us to create the baseline model and all 4 data augmentation experiments as outlined in the paper. For all of these 4 augmentation models, the learning rate was set to 1e-4, L2 regularization was set to 1e-7 and dropout parameter p was set to 0.5.

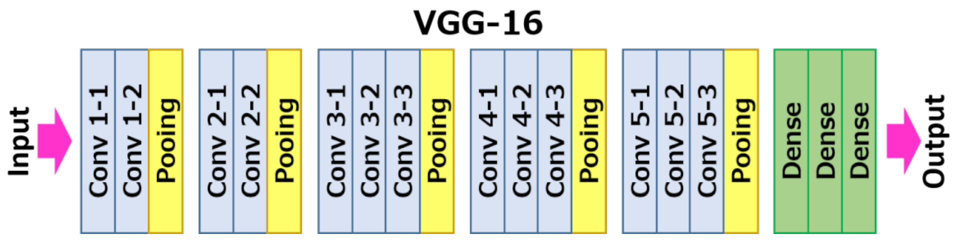


Figure 1: VGG 16 Model Architecture

2.3 Data Augmentation

To compare the effect of different data augmentation techniques on model's predictive performance, we used a baseline model that was only trained on all the 1600 unaugmented images to compare with the other four models. For each augmentation technique, we have explored at least 3 distinct values for each hyperparameter of the given augmentation technique to ensure the best experimental result from each technique is used in the final comparison.

2.3.1 Gaussian Blur

A Gaussian blur, also known as Gaussian smoothing, is the result of blurring an image by a Gaussian function. Each pixel in the image is considered independently, and its pixel value changed depending on its own value, and those of its surroundings, based on a filter matrix called a kernel.[4] The kernel consists of a rectangular array of numbers that follow a Gaussian distribution. A Gaussian blur is applied by convolving the image with a Gaussian function presented below.

$$G(x,y) = \frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where, x and y specific the delta from the center pixel. The values from this function will create the convolution matrix / kernel that we'll apply to every pixel in the original image. Below shows two images from the dataset, the one on left represents the normal image while the one on the right represents an image with a Gaussian Blur of Kernel Size = 91 applied to it.

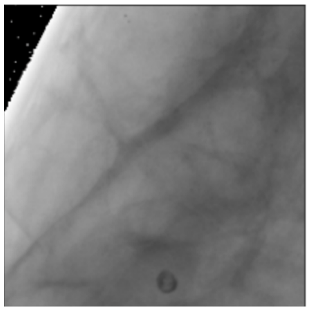


Figure 2: No Gaussian Blur

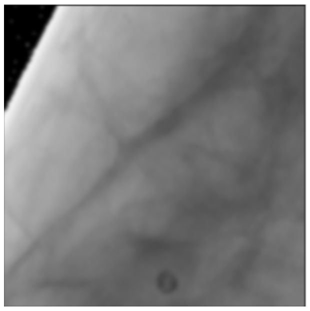


Figure 3: Gaussian Blur Kernel Size = 91

2.3.2 Gaussian Noise

Gaussian Noise utilizes a normal probability density distribution, also called as a Gaussian Distribution, where a random Gaussian function defined by a mean and standard deviation offset is added to the original image which generates this noise.[5] The noise can only take on values that are Gaussian distributed. The image on the left represents the original image, while the image on the right represents Gaussian noise with mean 0.2 and standard deviation of 0.09 is applied. To help with the Gaussian noise data augmentation transformation, a custom transformation class is created following the steps detailed in.[6]

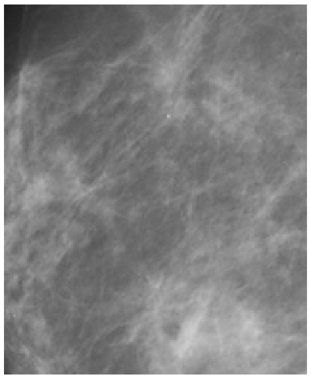


Figure 4: Without Gaussian Noise

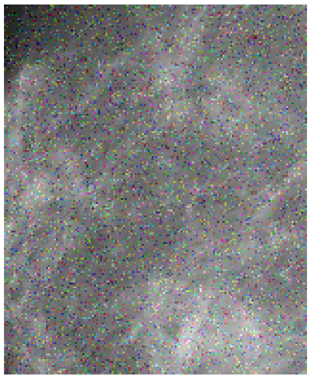


Figure 5: With Gaussian Noise

2.3.3 Random Rotation

Random Rotation is a Pytorch transformation where the image is rotated by user specified angle.[7] The figure below on the left shows the normal image while the figure on the right shows the image rotated by an angle of 90 degrees.

Figure 6: No Rotation

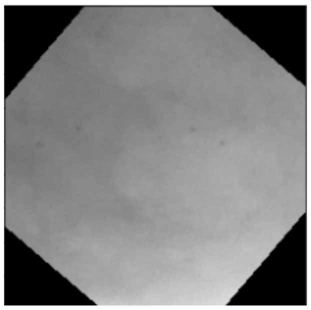


Figure 7: 90 Degree Rotation

2.3.4 Color Jitter (Contrast)

We focused mainly changing the contrast levels as a part of color jitter data augmentation. Changing the contrast in the images is another readily-available Pytorch transformation package where the contrast of the image is increased by the user specified range.[7] The figure below on the left shows the normal image while the figure on the right shows the image with increased contrast.

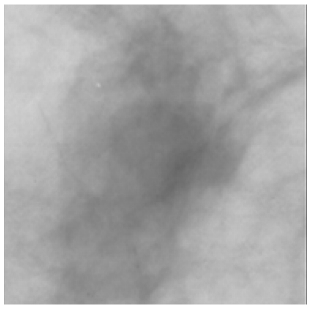


Figure 8: No Contrast

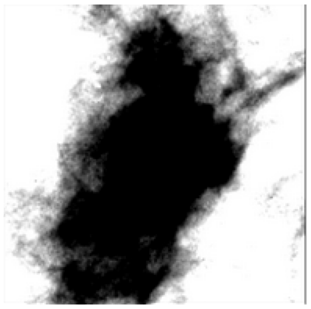


Figure 9: Contrast=10

2.4 Training and Evaluation

In terms of epoch number for model training, each model was trained on 50 epochs with each batch size equals to 64 images. During each epoch, the random shuffle for the training loader function was set to $True$ to ensure the augmented and original images were blended in each batch of images. After each training epoch, each model was tested with the batch size of 64 images. Once all 50 epochs of testing is finished, the highest testing accuracy is saved for comparison purpose.

For the hyperparameter used in each data augmentation technique, at least 3 different values were explored, and the one with highest testing accuracy was chosen as the final hyperparameter for that specific augmentation technique. Below are the final hyperparameters chosen for each augmentation technique:

$$GaussianNoise : Mean = 0.2, StdDev. = 0.09$$

$$GaussianBlur : KernelSize = 17$$

$$RandomRotations : DegreeofRotation = 10$$

$$ColorJittering : Contrast = 10$$

3 Experiment results

For a standard comparison, we have used testing accuracy as the metric of comparison. When comparing all classification metrics, it can be seen that all data augmentations have performed better than baseline model, indicating that adding data augmentation to the training dataset helps improve model performance.

ROC Curves are used to evaluate the model performance of a binary classification problem. ROC curves are a graphical way the connection/trade-off between clinical sensitivity and specificity for every possible cut-off for a test or a combination of tests. Looking at the ROC curve with AUC values in Figure 10, we naturally observe that the baseline model has the lowest AUC score of 76.87%, and that Gaussian Blur with an AUC of 92.88% is the best. Specifically in the ROC curve, better performing models are grouped on the top-left quadrant of the graph, and above the 50% random-guess AUC score which is represented by the black diagonal line.

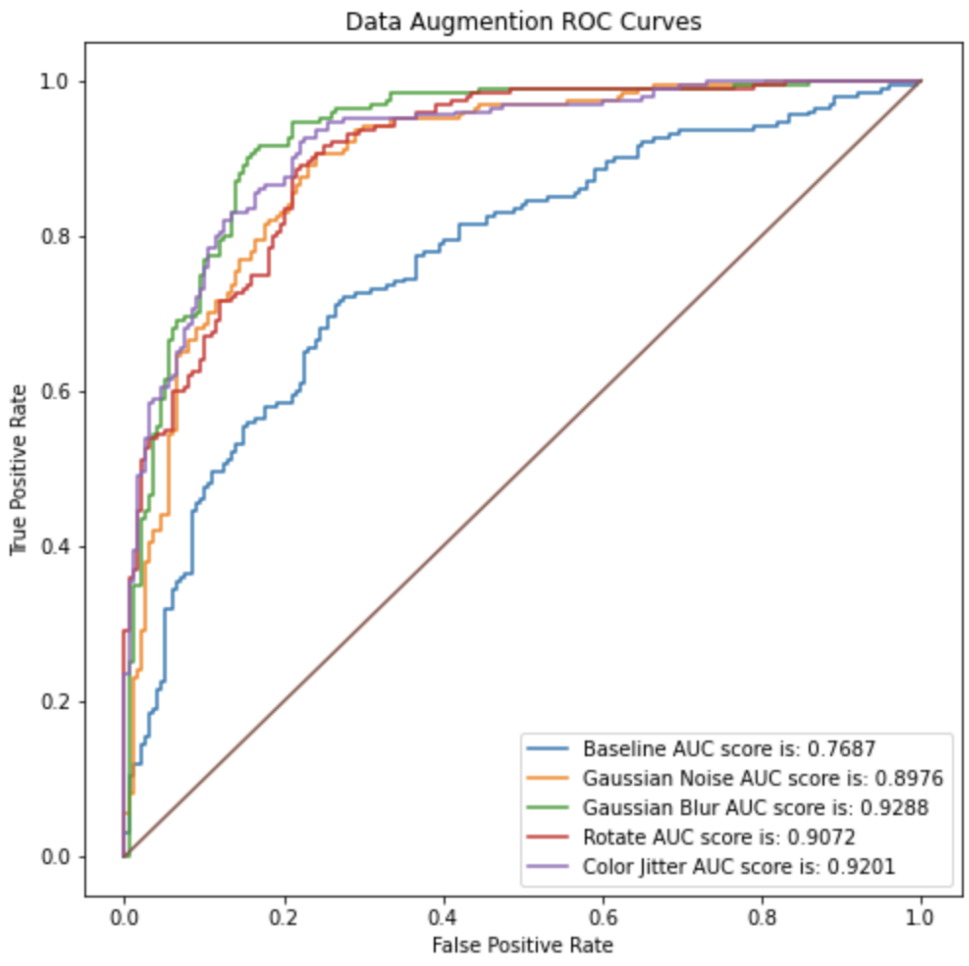
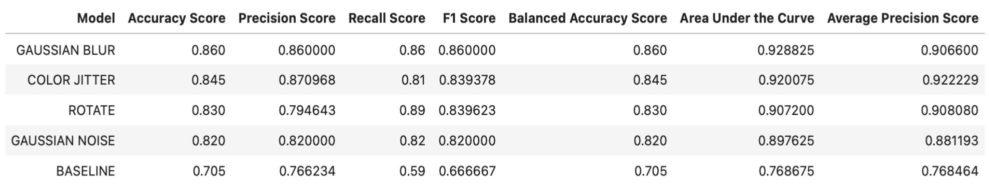


Figure 10: ROC Curves

In the Appendix section under the Additional Experiment Results sub-section, a Precision-Recall graph is plotted. Precision-Recall score is used to measure the trade-off between precision and recall. We similarly observe that models that are in the top-right quadrant have better Precision-Recall scores. In the PR curve, we learn that Color Jitters has the highest Average Precision score, followed by Rotation, and Gaussian Blur.

As shown in Figure 11 below, the augmentation technique with the highest testing accuracy score to the lowest is as follows: Gaussian Blur (86.0%), Color Jitter, Random Rotation, Gaussian Noise, and Baseline (70.5%).



Model	Accuracy Score	Precision Score	Recall Score	F1 Score	Balanced Accuracy Score	Area Under the Curve	Average Precision Score
GAUSSIAN BLUR	0.860	0.860000	0.86	0.860000	0.860	0.928825	0.906600
COLOR JITTER	0.845	0.870968	0.81	0.839378	0.845	0.920075	0.922229
ROTATE	0.830	0.794643	0.89	0.839623	0.830	0.907200	0.908080
GAUSSIAN NOISE	0.820	0.820000	0.82	0.820000	0.820	0.897625	0.881193
BASELINE	0.705	0.766234	0.59	0.666667	0.705	0.768675	0.768464

Figure 11: Overall Testing Results

Furthermore, in the context of healthcare application, we would also want to maximize the recall and minimize the type-2 error. In such case, the augmentation technique with highest recall score to the lowest is as the following: Random Rotation (89.0%), Gaussian Blur, Gaussian Noise, Color Jitter, and Baseline (59.0%). As a result Gaussian Blur is the superior data augmentation technique across both testing accuracy and recall score.

4 Conclusions

Having compared the VGG-16 model architecture driven models with 4 data augmentation techniques: Gaussian Blur, Gaussian Noise, Random Rotation, and Color Jitters, we observe that Gaussian Blur is the most effective data augmentation technique as it has the best scores for majority of classification metrics such as testing accuracy, AUC, and F1 Score. It also has the 2nd best recall score which is critical given the medical domain and the high price for false-negatives. As a result, Gaussian Blur is the most effective augmentation technique from our analysis with the DDSM dataset.

5 References

1. Hussain,Zeshan,et al.‘‘(PDF)Differential Data Augmentation Techniques for Medical Imaging Classification Tasks.’’ ResearchGate, Apr. 2018, https://www.researchgate.net/publication/325532618_Differential_Data_Augmentation_Techniques_for_Medical_Imaging_Classification_Tasks.

2. Scuccimarra, Eric A. ‘‘DDSM Mammography.’’ Kaggle, 3 July 2018, https://www.kaggle.com/skooch/ddsm-mammography.

3. Raj, Ananta. ‘‘DDSM Breast Cancer VGG-19 Features Extraction.’’ Kaggle, 15 Apr.2021, https://www.kaggle.com/vortexkol/ddsm-breast-cancer-vgg-19-features-extraction.

4. ‘‘Gaussian Blur.’’ Wikipedia, Wikimedia Foundation, 15 Oct. 2021, https://en.wikipedia.org/wiki/Gaussian blur.

5. Swain, Anisha.‘‘Noise in Digital Image Processing.’’ Medium,Image Vision, 9 Aug.2020, https://medium.com/image-vision/noise-in-digital-image-processing-55357c9fab71.

6. "How to Add Noise to MNIST Dataset When Using Pytorch." PyTorch Forums, 1 Nov. 2019, https://discuss.pytorch.org/t/how-to-add-noise-to-mnist-dataset-when-using-pytorch/59745.

7. ‘‘Torchvision.transforms.’’Torchvision.transforms-Torchvision 0.11.0 Documentation, https://pytorch.org/vision/stable/transforms.html.

A Timeline and task allocation

Week 1 (Nov 1st):

- Refine project idea and create project proposal. (Everyone)
- Everyone brainstorms project idea and pitches at least 1 project of interest.

Week 2 (Nov 8th):

- Read through model architecture details of identified research paper, along with understanding various data augmentation techniques. (Everyone)
- Everyone picks 2-3 data augmentation techniques to research.

Week 3 (Nov 15th):

- Extract dataset corresponding to paper and perform necessary data preprocessing (Everyone)
- Everyone works in parallel to explore various techniques to extract tfrecords medical imaging data.

Week 4 (Nov 22nd):

- Build baseline model and alternative models to compare performance. (Everyone)
- Ashish: Baseline and Gaussian Noise Model
- Pranav: Gaussian Blur and Random Rotation
- Tommy: Color Jitter (Including exploring contrast)

Week 5 (Nov 29th):

- Draft final project paper (Everyone)
- Ashish: Baseline and Gaussian Noise Model - Model Architecture and Results Section
- Pranav: Gaussian Blur and Random Rotation - Model Architecture and Results Section
- Tommy: Color Jitter (Including exploring contrast) - Model Architecture and Results Section

Week 6 (Dec 6th):

- Refine and submit paper and work on Poster Presentation. (Everyone)
- Collective effort where everyone works on the poster presentation.

B Implementation detail

Please refer to the README in the code zip file for steps to gather data, executive code, and generate model results.

C Additional Experiment Results

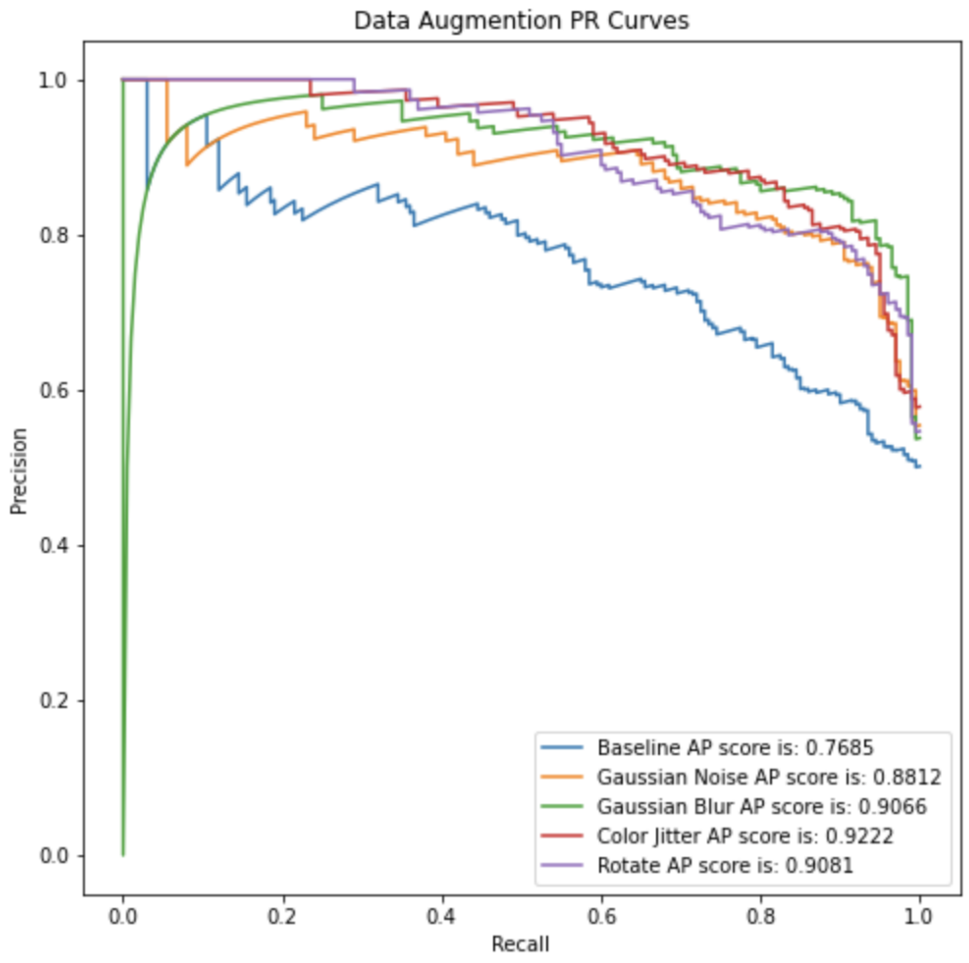


Figure 12: PR Curves