Medical Image Dataset Rebalancing via Conditional Deep Convolutional Generative Adversarial Networks (cDCGANs)

Adam Sun
Stanford University
450 Jane Stanford Way, Stanford CA, 94305
adsun@stanford.edu

Kevin Li
Stanford University
450 Jane Stanford Way, Stanford CA, 94305
kevinli8@stanford.edu

Abstract

Training deep learning models on imbalanced data sets is difficult in practice because models tend to be biased towards predicting the majority class [21]. This poses a particular challenge in the medical field, not only because the minority class may often be a disease we are trying to diagnose, but also because of the high risk nature of medical artificial intelligence (AI). In this paper we propose to rebalance imbalanced datasets by oversampling using Generative Adversarial Networks (GANs) [11]. We intentionally create an imbalanced dataset consisting of chest x-ray images labeled normal or pneumonia, with pneumonia images being the minority class. In particular, we consider Conditional Deep Convolutional Generative Adversarial Networks (cDCGANs), Deep Convolutional Generative Adversarial Networks (DCGANs) [26] conditioned on class labels [23]. For cDCGANs, we propose a new parameter α that indicates the probability of a randomly generated class label of 0. We then use these GANs to generate new synthetic pneumonia images. We train a ResNet18 model [15] on the new rebalanced datasets and compare their results to baseline, as well as some more common machine learning methods like undersampling, oversampling, class weights, and compare precision, recall/sensitivity, specificity, and the F1 score.

Our results show that DCGAN and cDCGAN-based oversampling methods can outperform class weights and classic resampling methods in terms of F1 score, and outperform the baseline across the board, making GANs a valid method to rebalance datasets. Additionally, we found that our proposed α hyperparameter can potentially be used to tweak the sensitivity and specificity of the ResNet18 model.

1. Introduction

Use of artificial intelligence (AI) in the medical field is becoming increasingly popular. However, it is frequently the case that in many medical image datasets, healthy patient data tends to have higher prevalence, while diseased data is rare. As a result, models trained on imbalanced data can often become biased, resulting in high false negative rates [33]. This can lead to medical conditions being left undetected, which leads to negative real world consequences, causing models to be less reliable. It is thus important to improve the model's performance despite the imbalance. Go-to approaches to deal with imbalanced datasets include oversampling, undersampling, class weighting and SMOTE [3]. However, these approaches can cause overfitting and may not be directly applicable for medical image datasets.

One method to improve the model's performance and reduce bias involves data augmentation [29] to create more samples of the minority class. However, this data augmentation takes the form of image flipping, rotation, scaling, and others, which still involve the same image.

We propose to train Generative Adversarial Networks (GANs) [11] on the minority class in order to generate diverse assorted minority class examples as a data augmentation step.

The GANs are trained on 128×128 grayscale chest X-rays. Then, the generator is saved and used to take a random noise vector as input and generate synthetic chest x-ray images as output. After using the synthetic chest x-ray images to rebalance the dataset, we compare these different methods with a ResNet18 [15] model. This model takes chest x-ray images as input, then performs binary classification, outputting whether each chest x-ray is healthy or has pneumonia.

2. Related Work

There are two main approaches when dealing with imbalanced datasets — modifying the loss function or modifying the dataset itself.

A very common method for **modifying the loss function** is class weighting of the loss function [19]. These weights are calculated by

$$w_j = \frac{n}{c \cdot n_j}$$

where w are the class weights, n is the total number of samples, n_j is the number of samples of class j, and c is the total number of classes. In a binary classification task, the binary cross entropy loss can then be given by

$$L = \frac{1}{N}\sum_{i=1}^{N} -w_0(y_i \cdot \log \hat{y}_i) - w_1((1-y_i) \cdot \log(1-\hat{y}_i))$$

This weighting can cause the minority class to have a bigger class weight than the majority class and thus a bigger impact on the loss function, compensating for the lack of instances of the minority class.

Cost sensitive learning is a generalization of class weights based approach— each misclassification is assigned a different cost based on a cost matrix [18]. For example, the cost of a false negative may be higher than that of false positive. One weakness of this method is that it might take a lot of effort to find a good cost matrix.

Another technique, Label-Distribution-Aware Margin loss (LDAM), proposed in [5], is similar to class weights and cost sensitive learning [18] in that a single example training loss depends on the label of the training example and how it is classified by the model. LDAM is a specially constructed loss function that is designed to maximize the margins from the minority class examples to the decision boundary, allowing to focus on minority examples close to the decision boundary that are prone to misclassification.

A frequent approach taken to **modify the dataset** is oversampling of the dataset via image data augmentation [29], which is a technique that is utilized in many state-of-the-art applications to generate more data at training time. This can involve height and width translations to encourage translation invariance for model predictions, and rotations for rotational invariance. Cropping is used to encourage model to learn to classify correctly even when looking at only a part of the image. The advantage of such a technique is that it is easy to generate lots of new training images. However in the medical context, a disadvantage is that data augmentation still preserves the overall structure of each individual medical image. This may prove to be an obstacle if there is a small number of patients, as augmentation may not be able to make the model robust to different manifestations of the disease.

SMOTE [3] and SMOTE-like methods like BorderlineSMOTE [12] and ADASYN [14] are common machine learning methods that oversample the minority class by generating synthetic data. In SMOTE methods, to oversample based on a minority class example x, we consider the k nearest minority class neighbors (where k is a hyperparameter), take a random such neighbor x', and generate a random synthetic example from the line segment connecting x and x' by randomly choosing a point on the segment. We continue oversampling via this process for random x until we sample enough synthetic examples as needed. BorderlineSMOTE and ADASYN are modifications of SMOTE where only a subset of the minority examples x are sampled, namely samples on the borderline and samples according to a density distribution, respectively. While SMOTE-based methods may be useful for numerical data, it is not practical for image purposes, since individually generating each pixel will not result in visually useful images.

Generating synthetic minority class examples with GANs is a relatively new problem [27]. The advantage of the GAN approach compared to simple oversampling or slightly more involved methods like SMOTE [3], is that GANs are desirable for their ability to represent complex data and generate realistic synthetic images from a training dataset. This means that GANs can theoretically help convert a scarce, discrete training dataset into a better, continuous dataset via nonlinear interpolation [27]. This can be incredibly desirable for our medical purposes, where the features in a chest x-ray indicative of pneumonia can be interpolated into many different variations that may not be sufficiently represented in our dataset.

CTGAN [32] was designed as a way to model tabular data with a mix of discrete and continuous columns. It uses a conditional generator to learn complex distributions in columns. CTGAN also is able to generate synthetic data with a specific discrete value. However, our approach concerns image data, a very different form of data.

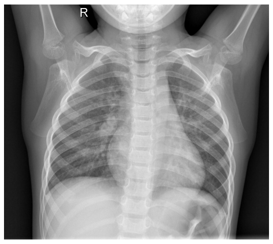
Since there are naturally not many instances of the minority class in image datasets, it may not practical to train a GAN with such a small amount of images. Balancing GAN (BAGAN) [22] addresses this fact by training on the whole dataset instead, then applying class conditioning for the generation of minority class images. BAGAN seeks to generate images that represent the minority class, are not repetitive, and are not already found in the training set. It uses an autoencoder based initialization strategy to help the model learn class-conditioning in the latent space.

Our approach explores oversampling the training dataset with a Conditional Deep Convolutional Generative Adversarial Network (cDCGAN) [26] [23] [4]. This deep-learning-based approach should be theoretically more advantageous over the more traditional class weighting and augmentation approaches due to the new and unique images generated. We train on the entire dataset, similar to the approach of BAGAN. However, while BAGAN generates latent vectors with a class-conditional latent vector generator to feed into the generator, we feed in the true labels along with randomly generated latent vectors into the generator. BAGAN's discriminator predicts the original class of each latent vector along with whether it was real or fake, while our approach solely predicts whether it was real or fake. Additionally, our approach does not use an autoencoder based initialization strategy.

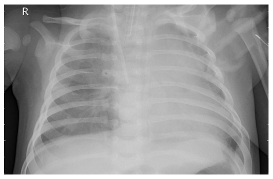
3. Dataset

We base our dataset on Kermany et. al's dataset of pediatric chest x-rays from Guangzhou [17]. In this dataset, there are 3875 pneumonia images and 1341 normal images. For our train dataset, we supplement the pneumonia dataset with 3044 healthy chest X-rays from the NIH's ChestX-ray8 dataset [31]. We had previously used images from [7] as a supplement, but had later discovered that the data was badly curated—in particular there were some CT-scans mixed in with the X-rays, so this caused us to use the NIH data instead.

In order to create a data imbalance where normal images were the majority class, we randomly deleted training examples from the pneumonia class so that we were left with 4385 normal examples and 1460 pneumonia examples. This allows our dataset to have 25% of chest x-rays with pneumonia and 75% of images that are healthy. Note that the test dataset is untouched and completely originates from the pneumonia dataset to provide an unbiased evaluation of model performance, with 234 normal images and 390 pneumonia images. Similarly the validation data we take from the training set is balanced to provide unbiased evaluation.



(a) Healthy X-ray



(b) Pneumonia X-ray

Figure 1. Images from the dataset. Note the presence of white pixels in a diffuse interstitial pattern indicative of viral pneumonia in the rightmost x-ray [17].

The image data is converted to grayscale and resized to 128×128. Then, to make the model more robust to translations and rotations, we perform data augmentation in the form of random horizontal flips, rotations, and cropping. We also normalize the images to ensure that each image has a similar distribution of pixel intensities before being fed into the ResNet18 model.

4. Methods

4.1. Model and Training

We use a Pytorch [24] ResNet18 model [15] based off of [30], with sigmoid activation for the final layer binary cross entropy loss. We change the number of nodes in our fully connected layer to 2 for our binary classification task. Utilizing deeper, more powerful models like ResNet50 [15] is prone to overfitting due to our relatively small dataset, and in preliminary experiments, this was confirmed when a large gap between train and test accuracy was observed.

As a baseline method, we trained a ResNet18 from scratch. We use a batch size of 32, and learning rate of 4e−3, with an Adam optimizer [20] with standard decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We also train for 5 epochs and save and evaluate only the model with the best validation accuracy to further prevent overfitting.

4.2. Resampling Methods

The classic methods for dealing with imbalanced datasets are resampling methods, namely undersampling and oversampling [27]. Undersampling involves randomly removing majority class examples, oversampling involves randomly duplicating minority class examples. The goal of each method is to make the number of training examples of each class the same to prevent bias. However, these methods are not desirable in practice – undersampling throws away useful data, whereas oversampling may cause the model to overfit since it duplicates data.

4.3. Generative Adversarial Networks

Generative Adversarial Networks [11] are state-of-the-art deep learning models that are used to learn the underlying distribution of complex data and that can be used to thus generate photo-realistic images after being trained on a dataset. These images can then be used to oversample the training set. The architecture consists of a generator network, whose task is to generate realistic images that fool the discriminator network, whose task is to distinguish between real and fake images.

The generator network learns the mapping from random noise to data space as $G(z;\theta_g)$, where G is a neural net. Meanwhile the discriminator learns a neural net $D(x;\theta_d)$, which outputs probability that x actually is real data. Thus if x is drawn from p_{data} the discriminator tries to maximize $D(x;\theta_d)$, equivalent to maximizing $\log D(x;\theta_d)$. Generating a fake example involves sampling $z \sim p_z(z)$ from noise, then applying $G(z;\theta_g)$. We would like to thus minimize $D(G(z;\theta_g);\theta_d)$, which is thus equivalent to maximizing $\log(1 - D(G(z;\theta_g);\theta_d)$. Putting this all together the discriminator networks wants to learn θ_d to maximize

$$\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x;\theta_d)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z;\theta_g);\theta_d))]$$

Now the generator and discriminator are playing a minimax zero-sum game with each other, so this means that the generator wants to learn θ_g to minimize the above expression. Thus the objective function ends up being

$$\min_{\theta_g}\max_{\theta_d}[\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x;\theta_d)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z;\theta_g);\theta_d))]]$$

4.4. Deep Convolutional GANs

Deep Convolutional Generative Adversarial Networks (DCGANs) [26] are a class of GANs that use all-convolutional layers for unsupervised learning. DCGANs are able to deal with larger and more complex data sets after modifying convolutional GANs with strided convolution, batch normalization, ReLU activation, and other architectural constraints. This allows for DCGANs to become more stable compared to vanilla GANs and learn good representations of the images for generative modeling.

The discriminator in our DCGAN model consists of 5 convolutional layers of kernel size 5 and stride 2 alternating with LeakyReLU activation with an alpha of 0.2, which is then flattened and fed into a 1-node fully connected layer with sigmoid activation. In the generator, the noise vector is fed through a fully-connected layer then reshaped into an (8,8,128) feature map. This is fed through 4 transpose convolutional layers of kernel size 4 and stride 2 followed by a single regular convolutional layer of kernel size 5 and tanh activation [2].

We train this model on the pneumonia images in the dataset for 160 epochs with a batch size of 128. We use an Adam optimizer [20] with learning rate $2e-3$ and decay rates $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Images were normalized to [0,1].

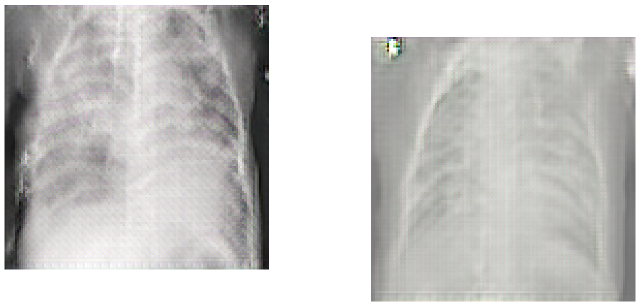
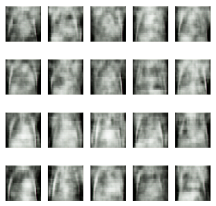
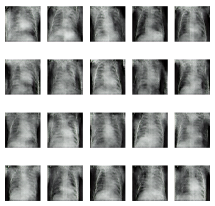


Figure 2. Fake Pneumonia X-rays generated by DCGAN. While they may look realistic, the white cloudiness indicative of pneumonia seems quite excessive, especially on the image to the right. Also notice the artifact on the top left of the rightmost image.


(a) DCGAN-generated images at epoch 20


(b) DCGAN-generated images at epoch 60

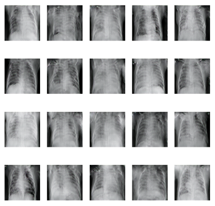

(c) DCGAN-generated images at epoch 160.

Figure 3. Images generated during different phases of the training process by DCGAN. The images start off quite pixelated, but gradually become more defined and realistic.

The images generated by the DCGAN seen in Figures 2 and 3 seem promising. Most of them bear semblances to real chest x-rays — however, some seem unrealistic. There is an excessive cloudiness in many images. While this is a feature indicative of pneumonia, it is obviously not found naturally, even to the untrained eye. We inferred that this is due to the DCGAN being trained on a relatively small dataset of only pneumonia chest x-rays. As a result, it was not exposed to healthy chest x-rays, which probably caused it to have trouble generating realistic pneumonia images.

4.5. Conditional GANs and Conditional Deep Convolutional GANs

Conditional GANs (CGANs) [23] are a conditional version of GANs which conditions the generator and discriminator on additional information y (usually a label, which is what our implementation does). This way, the objective function becomes

$$\min_{\theta_g}\max_{\theta_d}[\mathbb{E}_{x\sim p_{\text{data}}}[\log D(x|y;\theta_d)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z|y;\theta_g)|y;\theta_d))]]$$

The labels y are combined with randomly generated latent points $p_z(z)$ before being fed into the generator. This means the generator is trying to generate data while incorporating information y, causing the generated result to be conditioned on y. The discriminator also receives this label information y, which it uses to help discriminate between real

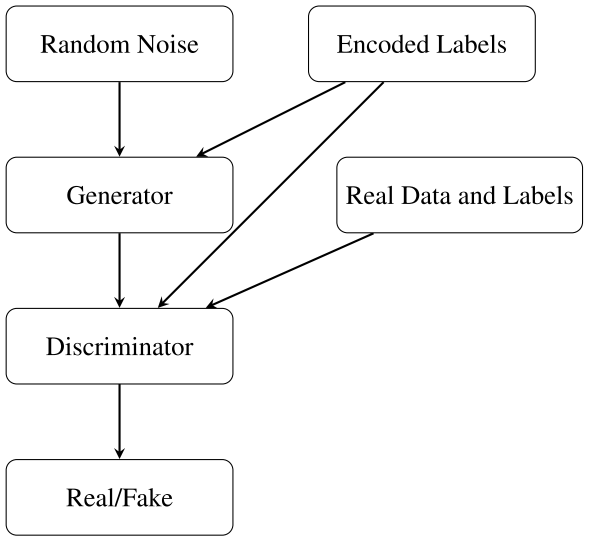
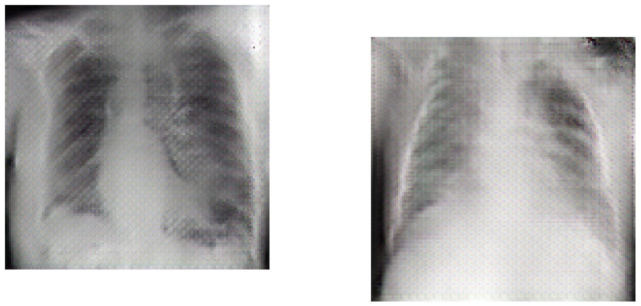


Figure 4. The overall architecture of a CGAN [23]. Randomly-generated labels are encoded with an embedding layer and then concatenated with random noise and fed into the generator. These encoded labels are also used to train the discriminator, which takes as input the generated data and encoded labels, as well the real data and labels, to distinguish between fake and real data.

and fake images that it knows the generator might be generating based off of y [23]. These CGANs can thus be useful for modeling complex multi-modal data distributions, as described in [23]. By training on the entire dataset and conditioning on labels, a CGAN should theoretically be able to learn the differences between healthy and pneumonia x-rays. This can improve quality of generated images over the DCGAN approach, which was not exposed to the healthy chest x-rays.

To modify our DCGAN architecture into a conditional DCGAN (cDCGAN), we randomly generate labels and pass them through an embedding layer to encode them into unique 50-element vectors. Then, these encoded labels are concatenated with the random latent points generated and inputted into the DCGAN generator. The encoded labels are also concatenated with the images in the DCGAN discriminator, as shown in Figure 4 [4].

Since the dataset used to train the cDCGAN is imbalanced, we explore the process of generation of random labels. We compare the default equal probability approach (cDCGAN) with a weighted approach we propose (weighted cDCGAN). The default approach used in cDCGAN is to generate labels of 0 and 1 equally with 0.5 probability. Our weighted cDCGAN approach has a hyperparameter α, representing the probability that label 0 (healthy) is generated. Label 1 will thus have a probability of generation of $(1 - \alpha)$. The default approach has $\alpha = 0.5$. However, this probability is not reflective of the dataset, which consists of 75% healthy images and 25% pneumonia images. By tuning the probability that these random labels are generated, we can influence the resulting generated images. So, we experiment with $\alpha = 0.25$ and $\alpha = 0.75$.



(a) A healthy chest x-ray generated by cDCGAN.

(b) A pneumonia chest x-ray generated by cDCGAN

Figure 5. Chest X-rays generated by cDCGAN. cDCGAN seems to be able to learn the general differences between healthy and pneumonia chest x-rays, as demonstrated in the slight cloudiness in the right image. The pneumonia image seems less extreme than the images generated by DCGAN, as a result of the class conditioning and being exposed to the entire dataset.

We train the cDCGAN model on the entire dataset and labels for 48 epochs with a batch size of 128. We use an Adam optimizer [20] with learning rate $2e - 3$ and decay rates $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We note that cDCGAN converges in less epochs than DCGAN due to the much bigger dataset. Images were normalized to [0,1].

5. Experiments

5.1. Methods and Metrics

We experiment with DCGAN, cDCGAN, and weighted cDCGAN to generate synthetic minority data to oversample the minority data. After training each different architecture, we save the model and then randomly generate latent points to use the trained generator model to oversample and generate new pneumonia images, which we then add to the imbalanced dataset in order to rebalance the dataset to a 50-50 split. A potential advantage of this approach is the ability to generate unlimited images by repeatedly feeding randomly generated latent points into the generator, allowing for repeated generation.

Our DCGAN implementation is based on [2], found at (https://www.kaggle.com/code/djibybalde/dcgan-keras-chest-x-ray-images), and our cDCGAN is inspired by [4], found at (https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/). We adapted the DCGAN code to add conditionality and the α parameter, and tuned hyperparameters to get the new model to converge. Our ResNet18 model is based on [30], found at

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{Recall/Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$
$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 6. Metric calculations. TP, FP, TN, FN denote the number of true positives, false positives, true negatives, and false negatives, respectively.

(https://www.pluralsight.com/guides/introduction-to-resnet). We adapted the ResNet18 code to adapt it to binary classification, to process imbalanced datasets, include image augmentation and normalization, as well as calculate metrics and generate heatmaps.

When training the ResNet18 model, we used a batch size of 32. We experimented with a few learning rates and found that `4e-3` was the optimal learning rate for the ResNet18 model.

Since we are working with imbalanced data, we do not focus on accuracy for our main metric. Instead, we use metrics like precision and recall, sensitivity and specificity (which are important in the medical context particularly) and F1 score (to measure general model performance). The metrics were computed with the help of scikit-learn library [25].

5.2. Results

Since our dataset is imbalanced, training from scratch can easily lead to bias. We compare training from scratch and training with pre-trained weights on ImageNet [8] from the PyTorch model zoo [24]. We explore the value of initializing our ResNet18 with pretrained ImageNet [8] weights.

As shown by Figure 6, initializing with pretrained ImageNet weights improves the performance of our model. Since ImageNet is a large dataset with a large number of classes, the weights contain features that can be transferrable to a wide variety of tasks. The baseline has very low precision and outstandingly high recall, which means that the model returns many results, but most are incorrect [25]. On the other hand, initializing with the pretrained model has a more balanced precision and recall, causing it to have a higher F1 score. So, we initialize with ImageNet pretrained weights for the rest of the experiments, as it results in a less biased model.

Now, we compare the baseline performance of the model along with class weights, undersampling and oversampling, DCGAN, as well as weighted cDCGAN with $\alpha =$

	Precision	Recall/ Sensitivity	Specificity	F1
Baseline	0.543	**0.927**	0.780	0.685
Pretrained	**0.730**	0.842	**0.850**	**0.782**

Figure 7. Comparison of test evaluation metrics for baseline ResNet18 and baseline model pretrained on ImageNet.

	Precision	Recall/ Sensitivity	Specificity	F1
Baseline	0.730	0.842	0.850	0.782
Class Weights	0.859	0.855	0.915	0.857
Under-sampling	0.709	0.865	0.843	0.779
Over-sampling	0.863	0.845	0.917	0.854
DCGAN	0.821	**0.910**	0.898	0.863
cDCGAN $\alpha = 0.50$	**0.893**	0.857	0.934	**0.874**
cDCGAN $\alpha = 0.25$	0.710	0.706	**0.949**	0.803
cDCGAN $\alpha = 0.75$	0.821	0.901	0.898	0.859

Figure 8. Comparison of baseline ResNet18 with class weights, undersampling, oversampling, DCGAN, as well as cDCGAN methods. Note that the data augmentation scheme detailed in Section 3 of this paper was utilized for all of these methods.

$0.25, 0.5, 0.75$ in Figure 8.

Based on the results, we conclude that GAN-based approaches are a valid way to oversample an imbalanced training dataset. DCGAN and the cDCGAN variants were all able to outperform the baseline in almost all metrics. In particular, both our DCGAN and cDCGAN ($\alpha = 0.5$) approaches outperform class weights, oversampling, and undersampling (the most commonly used methods) in terms of F1 score.

Our cDCGAN with $\alpha = 0.5$ achieves the highest F1 score out of all the methods, although by a very narrow margin. Interestingly, using $\alpha = 0.25$ slightly improves specificity at the cost of precision and recall/sensitivity, while using $\alpha = 0.75$ slightly improves sensitivity at the cost of precision and specificity. This means that this hyperparameter can be tweaked according to the specific task at hand. Another observation of note is the fact that the cDCGAN with $\alpha = 0.75$ seems to have very similar metrics to the DCGAN, suggesting that generating labels according to the true distribution of classes essentially negates the conditional nature of the model.

Interestingly, DCGAN achieves the highest sensitivity and cDCGAN ($\alpha = 0.5$) achieves the highest specificity out of all methods. This means that the dataset oversampled with DCGAN will produce much less false negative results, and one with cDCGAN will produce much less false positive results, making DCGAN more desirable if we want to rule out disease, and cDCGAN more desirable if we want to rule in disease. This unveils the possibility of using both DCGAN and cDCGAN in conjunction to create an even better dataset, a possible future path to explore.

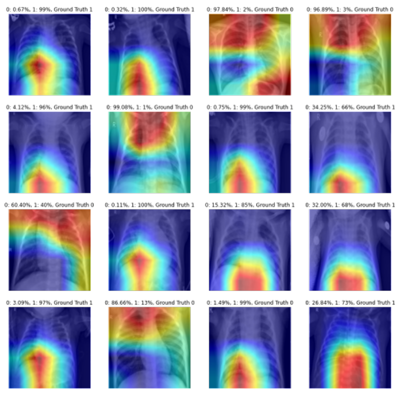
One important note is the stochastic nature of model training. Due to time constraints, models were only trained once and results recorded. Despite random seeding and using the same ImageNet weight initialization for all the models, due to the narrow margins between each method, there exists the possibility that these results occurred purely by chance.

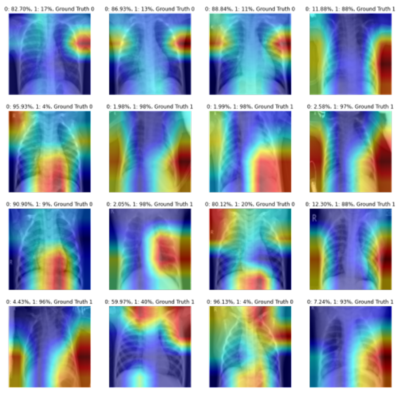
5.3. Visualization

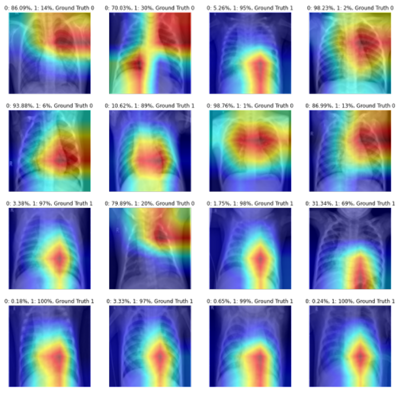
Zhou et al proposed Class Activation Maps [34] as a way to visualize where and with what intensity a CNN model is looking at an image, but it trades off model complexity and performance. Grad-CAM [28], on the other hand, does not modify the network and uses the gradients flowing into the final convolutional layer in order to produce a map that localizes and highlights important regions in the image indicative of a certain class. This is especially important and useful in the medical field. Firstly, it can be used as a gauge of model relevance. If the Grad-CAM indicates that the model is looking at somewhere other than the lung cavity to make predictions, we know that the model is not a valid predictor of pneumonia. Secondly, using Grad-CAM can be helpful to help doctors localize the disease. Since Grad-CAMs highlight specific regions of an image, an experienced radiologist can then easily confirm these locations. This increased model transparency and explainability thus makes Grad-CAM a great way to gauge the quality of our different oversampled datasets.

We generated Grad-CAM images using [10] for the baseline, DCGAN, and cDCGAN($\alpha = 0.5$) methods, using the highest-scoring category to generate each image.

Upon examination of the class activation maps, we can see that the DCGAN is definitely learning irrelevant features. In particular, the splotches on the left and right of the chest x-ray is indicative that it is not actually looking at the lungs, where pneumonia is found. The baseline method faces a similar problem, with the model commonly looking at the abdomen instead of the lungs. cDCGAN does a bit better in this respect, as the highlighted portions of the Grad-CAM tends to be more varied and generally lie in the chest cavity.


(a) Grad-CAMs generated for a random test batch for the baseline method.


(b) Grad-CAMs generated for a random test batch for DCGAN method.


(c) Grad-CAMs generated for a random test batch for cDCGAN ($\alpha = 0.5$) method.

6. Conclusion

In this work, we explored the usage of Deep Convolutional Generative Adversarial Networks (DCGANs) and conditional Deep Convolutional Generative Adversarial Networks (cDCGANs) for oversampling of the minority class in imbalanced medical image datasets. Furthermore, we proposed a new α hyperparameter for cDCGANs that can potentially be used to tune the sensitivity and specificity of the model.

We found that we found that our DCGAN and cCDGAN based approaches outperform standard approaches in terms of F1 score, despite potentially underperforming in other metrics. However, despite DCGAN performing well in metrics, we note that the Grad-CAMs generated indicate that the ResNet-18 model is not learning relevant features for pneumonia diagnosis. The dataset oversampled with cDCGAN does not face this issue as much, which we attribute to more realistically generated images as a result of the conditional component of the network. Nevertheless, we believe that our approach has the potential to become a reliable way to oversample an imbalanced image dataset.

In the future, we would like to tune the GAN models more carefully, and train models multiple times to compensate for the stochastic nature of model training. The bad Grad-CAMs could likely be the result of a bad dataset, so curating a more diverse imbalanced dataset should also be a future step. We may offer our GAN-generated images to an actual experienced radiologist for review for usefulness, relevance, and realisticness, resulting in a better annotated dataset. More time and computational power may admit the ability to test out and compare our results to some other approaches highlighted in the related work section, including BAGAN [22], as well as cost sensitive learning [18], active learning [9], and LDAM [5]. We also want to explore how well our model architecture transfers to other medical image datasets. The methods in this paper may prove useful for other cancers like skin cancer, as well as any image dataset in general.

7. Group Member Contributions

7.1. Adam Sun

Idea for project. Performed literature review on different GAN models and techniques dealing with imbalanced dataset. Wrote code to load data and create imbalanced dataset. Implemented ResNet18 in PyTorch, implemented code for loading data and training Resnet18 model. Implemented, experimented, and trained all GAN models on Amazon Web Services (AWS), and ran experiments comparing Resnet18 performance on each GAN-oversampled dataset. Used grad-CAM package to generate heatmaps. Scribe for milestone and final report, especially the Methods and Experiments sections.

7.2. Kevin Li

Idea for project. Reviewed papers and verified understanding of concepts. Implemented code for preprocessing and managing dataset for creating imbalance. Ran early tests on ResNet50 that were overfitting. Ran baseline and basic (non-GAN) tests. Implemented code, ran all experiments and compiled all results for CS229 portion of the project, and incorporated results into the CS231N paper for comparison. Scribe and reviewer for milestone and final report, in particular contributed to the math-heavy portions of the report, managing citations, $\LaTeX$ typesetting, and general formatting. Poster developer and printer.

7.3. CS229 Overlap

The CS229 portion of the project, which is being completed by the second author of this paper, Kevin Li, and Taran Kota (`tkota@stanford.edu`), examines more classical methods for resolving data imbalance, including methods mentioned before, like oversampling, undersampling, standard image data augmentation involving flipping, rotating, and cropping images, class weights, and SMOTE-like methods: SMOTE, BorderlineSMOTE, and ADASYN. All of this is done without the use of pre-learned model weights. Certain methods like undersampling, oversampling, data augmentation, and class weights are done in the CS231N portion of the project but with pre-learned weights from ImageNet, and the GAN work is unique to the CS231N portion. Taran's work does not overlap with any work or research presented in this paper at all.

7.4. Additional Acknowledgements

For the ResNet18 model, we built off of the code base [30]. The DCGAN code was based off of [2], and the cDCGAN code was inspired by [4]. We also acknowledge the wonderful open-source PyTorch library [24], Tensorflow [1], Keras [6], and other libraries like NumPy [13] (for managing data), Scikit-Learn [25] (for computing metrics), Matplotlib [16] (for plotting and visualizing images), and Jacob Gildenblat's Pytorch Library for CAM methods [10] (for class activation maps).

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensor-

Flow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 8

[2] Djiby Balde. Dcgan-keras (chest x-ray images). https://www.kaggle.com/code/djibybalde/dcgan-keras-chest-x-ray-images, 2020. 4, 5, 8

[3] Kevin W. Bowyer, Nitesh V. Chawla, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *CoRR*, abs/1106.1813, 2011. 1, 2

[4] Jason Brownlee. How to develop a conditional gan (cgan) from scratch. https://machinelearningmastery.com/how-to-develop-a-conditional-generative-adversarial-network-from-scratch/, 2019. 2, 5, 8

[5] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garneet, editors, *Neural Information Processing Systems*, 2019. 2, 8

[6] Francois Chollet et al. Keras, 2015. 8

[7] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection. In *ArXiv*, 2020. 3

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[9] Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: Active learning in imbalanced data classification. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, page 127–136, New York, NY, USA, 2007. Association for Computing Machinery. 8

[10] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021. 7, 8

[11] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2672–2680, Cambridge, MA, USA, 2014. MIT Press. 1, 3

[12] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *International Conference on Intelligent Computing*, volume 3644. Springer, 2005. 2

[13] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. 8

[14] Haibo He, Yang Bai, Edwardo A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE Joint Conference on Neural Networks*, pages 1322–1328, 2008. 2

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 3

[16] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 8

[17] Daniel S. Kermany et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. In *Cell*, volume 172, pages 1122–1131, 2018. 3

[18] Salman H. Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A. Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3573–3587, 2018. 2, 8

[19] Gary King and Langche Zeng. Logistic regression in rare events data. *Political Analysis*, 9:137 – 163, 2001. 2

[20] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 3, 4, 5

[21] Bartosz Krawczyk. Learning from imbalanced data: Open challenges and future directions. In *Progress in Artificial Intelligence*, volume 5, pages 221–232. Open Access, 2016. 1

[22] Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and A. Cristiano I. Malossi. BAGAN: data augmentation with balancing GAN. *CoRR*, abs/1803.09655, 2018. 2, 8

[23] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014. 1, 2, 4, 5

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 3, 6, 8

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 6, 8

[26] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 1, 2, 4

[27] Vignesh Sampath, Iñaki Maurtua, Juan José Aguilar Martín, and Aitor Gutierrez. A survey on generative adversarial

networks for imbalance problems in computer vision tasks. *Journal of Big Data*, 8(1), Jan. 2021. 2, 3

[28] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. 7

[29] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. In *Journal of Big Data*, volume 6. Springer Nature, 2019. 1, 2

[30] Gaurav Singhal. Transfer learning with resnet in pytorch. `https://www.pluralsight.com/guides/introduction-to-resnet`, 2020. 3, 5, 8

[31] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *CoRR*, abs/1705.02315, 2017. 3

[32] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garneet, editors, *Neural Information Processing Systems*, 2019. 2

[33] Yang Zhao, Zoie Shui-Yee Wong, and Kwok Leung Tsui. A framework of rebalancing imbalanced healthcare data for rare events' classification: A case of look-alike sound-alike mix-up incident detection. *Journal of Healthcare Engineering*, 2018:1–11, 2018. 1

[34] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016. 7