Generative Editing via Convolutional Obscuring (GECO): A Generative Adversarial Network for MRI de-artifacting

Bryce Allen Bagley [a 1], Sergei Petrov [a 2], Ge Cheng[3], Matei Armanasu[4], Nancy Fischbein, MD[5], Bin Jiang, MD, PhD[5], Michael Iv, MD[5], Eric Tranvinh, MD[5], Michael Zeineh, MD, PhD[5], and Olivier Gevaert, PhD[b6,7]

[1]MD program, Stanford University
[2]Department of Energy Resources Engineering, Stanford University
[3]Gevaert Lab, Stanford University
[4]Department of Statistics, Stanford University
[5]Department of Radiology, Stanford University
[6]Stanford Center for Biomedical Informatics Research (BMIR), Department of Medicine, Stanford University
[7]Department of Biomedical Data Science, Stanford University

Abstract

Magnetic resonance imaging (MRI) is the dominant diagnostic technique to non-invasively image the brain, and deep learning has proven a powerful tool for analyzing these images. However, machine learning models trained on such MRI data have empirically shown an ability to detect complex and invisible artifacts, such as which type of machine a scan was taken from to a high degree of accuracy. Such artifacts are potentially invisible to the human eye, but can be identified by machine learning systems, leading them to focus on irrelevant features rather than scientifically and/or medically useful ones. For example, machine learning systems can often "shortcut" past the actual features researchers would like to detect and utilize separate spurious correlations to make predictions. Several such undesired features have been reported to interfere with cross-institutional medical imaging deep learning research, and more are likely to be identified as time goes on. Here, we develop a method capable of removing these spurious correlations in an unsupervised manner, leveraging generative techniques to produce images which maintain image quality while learning how to remove technical artifacts. Generative Adversarial Networks are a class of deep learning architectures which have shown impressive efficacy in image generation and editing tasks, and our work builds upon this success. Here, we propose Generative Editing via Convolutional Obscuring (GECO), a Generative Adverserial Network for MRI deartifacting. GECO is based on a CycleGAN, a GAN architecture designed for image-to-image translation that is transforming an input image into a new image with one or more desirable properties. By formulating the CycleGAN loss as a two-player game with a regularization term and incentivizing the generator to erase spurious correlations the original image quality can be better preserved. Beginning with classifiers trained on original images to identify images based on artifacts of interest, GECO reduced the classifiers' ability to detect these spurious correlations from 97% down to a difference which is nearly equal to a classifier making purely random guesses. We also observe over 98% structural similarity between the original and deartifacted images, indicating the preservation of the vast majority of non-spurious information contained in the original images. In addition to solving the known problem of avoiding artifacts from scanner type, this method opens the door to potentially removing many other types of spurious correlations from medical images.
Software availability: https://github.com/gevaertlab/GECO.

1 Introduction

In the field of medical imaging there is a strong desire to train AI models for disease and pathology detection, and there is great excitement regarding the dream of a fully autonomous end-to-end medical system to aid radiologists,

[a]Co-first authors
[b]Corresponding author: ogevaert@stanford.edu

pathologists, and other physicians. Magnetic Resonance Imaging (MRI) is a powerful modality which requires no radiation exposure yet still provides valuable clinical information to medical professionals, particularly with brain pathologies given the difficulty of imaging within the skull using X-rays. The complexity of medical scans and their information-richness have made them a key target of machine learning research within medicine, and impressive successes have repeatedly been made possible thanks to the past decade's advancements in deep learning. Methods such as Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs), autoencoders, and other deep learning models can infer properties of images which a human eye would struggle to notice - or in some cases be incapable of detecting. However, when machine learning models are trained on medical images (MRI or otherwise) they often encounter issues from spurious correlations. For example, it has been known for years that AI models easily memorize features of scans such as the particular model of machine used to take the scan. Because of this, models often fail to generalize beyond the training dataset[1]. Similar issues arise when a distributional shifts occurs in real-world data, which is known to occur over time in many medical contexts. The specific artifacts used by models to predict machine type and manufacturer are generally poorly understood and often invisible to the naked eye, so removing them from images is a complex task. Here, we propose, GECO, a method for automatically removing said artifacts, by leveraging generative deep learning techniques to 'harmonize' an input image. We develop a generator network that is capable of modifying an input MRI image to prevent a classifier from being able to identify the machine manufacturer corresponding to medical images, while also not affecting the quality of the image. For this task, we employ a combined loss function for the generator network, which uses a latent space of uncorrected MRI scans, to encode an incentive to maintain the features of the input image while also applying the GAN technique of a competing classifier and generator to train both networks.

GECO takes in a MRI image and outputs via a generative network a 'deartifacted' version of the original image. This deartifacted product retains virtually all of the structure of the initial scan, but eliminates or significantly reduces any identifying marks that would allow a deep learning model to determine the spurious correlation from the original image. Our approach for doing so is to use a GAN, which is trained to take raw MRI scans and remove as much as possible these spurious correlations. Note that this is not the same process as removing scanning artifacts such as zipper or herringbone artifacts that affect image quality, but rather the intent is to remove features that are likely invisble to the human eye and impact a classifier's ability to categorize the image. These 'marks' in question are thus encoded in complex correlations which human users cannot identify, but which deep learning techniques often show an ability to recognize. Our results focus on image artefacts corresponding to different MRI manufacturers and field strengths. In each case, across the training of these GANs we reduce the classification accuracy from nearly perfect down to a level essentially equivalent to random guessing on the part of the classifier.

Examining related studies in this area, the work of Zech et al. [1] serves as one of the primary inspirations for this project, and we set out to resolve some of the relevant issues via a state-of-the-art deep learning solution. Zech et al provided an overview of the difficulties in training classifiers on radiological data, and discussed potential reasons for the observed generalization gaps. Despite the difficulties of spurious correlations, there is much promise in the use of machine learning for radiology. Pranav Rajpurkar et al. [2] and Varun Gulshan et al. [3], achieved performance similar to radiologists on specified tasks. The latter case saw at least some successful generalization between institutions' patient populations - though it is not unsurprising that some cases would successfully generalize despite artifacts. Work in this field indicates that CNNs have the capacity to attain a near-expert level of performance in the absence of spurious correlations, and our focus here is on reducing data leakage to enable better generalization in future work in the field. For example, Sebastian Voort et al. [4] developed an MRI image pre-processing pipeline capable of detecting metadata in a scan using purely the image data alone, and we use their architecture as a baseline for our classification task.

The GECO generator is based upon the U-net architecture now widespread in image-related deep learning tasks. Developed by Ronnenberger et al.[5], a U-net has proven extremely effective for image segmentation and a variety of other problems, especially in the medical machine learning literature. While a U-net forms the generator, the overall structure of our GAN is inspired by Jun-Yan Zhu et al.'s [6] CycleGAN. This structure proved to be an excellent approach for GANs with a latent space of domain images. Our pixelwise loss has similarities to the identity loss term included in CycleGAN, and our experiments also bear out the stabilizing effect they propose in their paper.

In contrast with the work done by Zhicheng Yan et al. [7] on photo editing via neural networks which performed generic edits, our architecture for image editing is a one-shot, fully convolutional process that has no reliance on segmentation maps or other learning tasks in order to generate the edited image. Jun-Yan Zhu et al. [8] also published a generative approach to photo editing, but this was again in the context of generic image editing and their method produces divergence between the input and generated images which are not desirable for de-artifacting. On the related

task of image inpainting, Deepak Pathak et al. [9] saw some success at generating blocked out data from general images, though once again encountered the traditional difficulties observed with generative networks such as blurriness in high-detail areas.

In summary, related approaches to image editing required some level of manual defining of desired changes, which is highly problematic in the case of artifacts with unknown and complex distributions. Our work overcomes this difficulty by directly detecting the artifacts' distributions and editing them out all within a single, unsupervised architecture which we call Generative Editing via Convolutional Obscuring, or "GECO". In this paper, we demonstrate that GECO is a powerful approach for removing unwanted information from images with complex distributions. Specifically, we succeed in drastically reducing a classifiers' ability to detect spurious correlations in MRI images, and further demonstrate that GECO is similarly capable of "fooling" expert neuro-radiologists.

2 Methods

2.1 Technical Approach

Our process begins with a large collection of MRI images, whose metadata is known to us but not visible to humans in the images themselves. Two pieces of metadata were selected on the basis of their complex impact on the images themselves: scanner manufacturer and scanner field strength. MRI machines have designs which vary between manufacturers[1] and play into the physics of the scanners, and additionally the strength of the magnetic field generated by a scanner significantly impacts the images produced. For classifying by manufacturer, on the basis of data availability we selected four companies: GE Medical Systems, Siemens, Philips Medical Systems, or Imaging Biometrics. While most of the categories studied in this paper correspond to actual manufacturers and their proprietary software which is run on the respective types of machines, Imaging Biometrics is in fact a software company. Their software is run on various machines, and the presence of artifacts traceable to this company despite the machines not being specifically theirs makes its inclusion in our research reasonable. In this paper we use "manufacturer" to mean the label which is found in the "manufacturer" entry for an image's DICOM file (a standard format in radiology). For field strengths, most commercial models today operate at either 1.5 Tesla (T) or 3T, and so these were the natural classes for the network to operate with in the field strength experiments. While not all images have precisely one of these strengths, the margins by which any differ from the two classes is proportionately quite minimal. It is important to observe that this is a classification problem rather than a real/fake detection problem as is more common in GAN training systems. We assessed only one of the two classification problems - manufacturer and field strength - in a given experiment, so separate results are provided for each.

The generator network is given only the original MRI scan itself, and outputs a 'deartifacted' scan which retains as much as feasible the initial features of the image. The generator is a convolutional network with six convolutional layers, as well as batchnorm layers and leaky ReLU activation functions. We use a Tanh function as the final activation to squeeze the output between -1 and 1. The classifier network follows the structure used by DeepDicomSort[4], employing a similar set of six convolutional layers, along with two dense layers on top for generating the final machine-type classification scores. Our core architecture is shown in Figure 4 from Appendix B.

The structure of the losses shown in Appendix B pit the generator and classifier networks against one another such that improving the loss of one worsens the loss of the other, with the goal that this teaches the generator network a strong representation of what 'deartifacted' images look like without having to provide examples ourselves. We define our learning target for this problem by developing a generator loss with two distinct components: the first is a sum of squared pixelwise errors between the input image and the output image, and the second is a cross-entropy style loss applied to the output of our classifier network. This 'inverse' cross-entropy loss is low when the classifier outputs a uniform probability distribution, and goes to infinity as any one class probability drops to zero (note the similarities to the TripleGAN structure [10]). Note, as this portion of the loss is our own formulation, we provide a contour plot for the two-class case in Appendix B to aid in conceptualization. The generator loss can be expressed as follows, where m is the total number of pixels in an image, c is the number of different classes, Gen.$(\cdot)$ is the output of the generator on a given input, Clas.$(\cdot)$ is the output of the classifier on a given input, and MSE is the mean-squared-error:

[1] There is also variance between different models from the same manufacturer, but a larger dataset would be needed to properly study de-artifacting at this more fine-grained level.

$$L_{gen.} = \frac{-1}{c}\sum_{i=1}^{c} log(Clas.(Gen.(X))) + \beta\sum_{i=1}^{m} MSE(X_i, Gen.(X)_i). \tag{1}$$

As part of our experiments, we also explored choosing class 0 (GE Medical Systems) arbitrarily as the 'default' class, with the loss structured to encourage images that the classifier labels as this default. This reformulation is given by:

$$\tilde{L}_{gen.} = CE(Clas.(Gen.(X)), \vec{0}) + \beta\sum_{i=1}^{m} MSE(X_i, Gen.(X)_i). \tag{2}$$

Note the beta term associated with the MSE component of the loss, which we use as a hyperparameter to tune the importance of the generator's two tasks. We interpret this loss as being split into a classification and a discrimination component, replacing the discriminator network from TripleGAN [10] with discriminatory loss in our case for simplicity and stability. The classifier network is only used during training and evaluation time, and takes in as input a MRI scan from the generator. The classifier outputs a predicted class label which is compared to the true label to generate the cross-entropy loss that is used to update the classifier. The loss that guides the classifier training process is calculated as follows:

$$L_{disc.} = CE(Y, Clas.(Gen.(X))) \tag{3}$$

The final product of this process is the trained generator network, which should be able to deartifact an MRI scan given only the raw unpadded image, as all the learnable weights are in fully convolutional layers that do not care about the pixel size of the image.

In this work we considered the following architecture variants: a core generator/classifier pair, a max-pooling classifier, a three-player architecture, a residual generator and a multi-residual block generator. Further we explore four variants in addition to our core architecture.

2.1.1 Variant Classification Architectures

In terms of the classifying part of our architecture, we explore two other variants to observe how they affect performance on this classification task. We begin with a discussion of the variants on the classifier network, specifically including a max-pooling layer after all convolutional layers, and introducing a discriminator network to explore three-player architectures. For the max-pool classifier, the pooling layer guarantees that the dense head of the classifier will always receive N inputs where N is the channel size of the final convolutional layer. This allows the classifier to be scale invariant, so long as the features of the image are not vastly larger or smaller than the images on which the classifier is trained.

In the case of the three-player architecture, we explore the benefits and drawbacks of using a learned discriminator network in place of the pixelwise MSE loss, resulting in a GAN structure with three distinct networks. In this structure the generator modifies the input images, and the generator output is fed to both a discriminator network that attempts to guess whether or not the image is unmodified or output from the generator and a classifier network that attempts to guess which type of machine the image was taken from. The discriminator is also fed real images from the dataset as a separate phase of training in each iteration. As a result, the discriminator and classifier are evaluated using binary or standard cross entropy as relevant, and the generator is evaluated using a combination of our own loss function on the classifier's output, and the reverse of the binary cross-entropy loss on the discriminator's output. That is our generator loss becomes (where BCE stands for binary cross-entropy)

$$L_{trip.gen.} = \tilde{L}_{gen.} + BCE(Clas.(Gen.(X)), \vec{0}) \tag{4}$$

(where $\tilde{L}_{gen.}$, described in Equation (5), is used to tackle a variant on our initial problem). Thus, the generator attempts to both fool the classifier into being unable to predict the class of the output image, while also producing 'realistic' images as evaluated by the discriminator. A diagram of our variant three-player structure is shown in Appendix D.

2.1.2 Variant Generator Architectures

In combination with our classification architectures, we also explore variants on the architecture of our generator with the goal of improving image quality and the model's ability to fool the classifier. The primary technique we explore here is the concept of a residual generator, that is learning the edits to make to an image instead of the raw image itself. To do this, we construct the generator as normal with all convolutional layers, but after generating the final output we scale it down by a learnable parameter α and add it to the initial input. Mathematically, we represent the output X' of the generator on an image X as:

$$X' = X + \alpha * Gen.(X) \tag{5}$$

With this structure, we allow the generator to decide the magnitude of the changes required to fool the classifier, while not requiring it to learn the identity mapping that optimizes the MSE component of the loss. We also explore a multi-residual block structure, with three residual blocks of two convolutional layers each. At the end of each block, we add the values from the start of the block to the output, testing this residual structure to see if it provides benefits in a generative context. While the original ResNet paper performs this operation in the context of significantly deeper networks tuned for classification, we believe that it is worth exploring these options due to the naturally residual nature of the problem the generator is being asked to solve–that is, applying minor edits to the image while maintaining most of the structure.

The generator architecture which proved most effective was a modified U-net [11]. A U-net is a modification of the normal ResNet structure, where instead of periodic blocks the architecture forms a sort of "U" or "V" shape, with layers arranged along the curve of the eponymous letter and residual connections crossing the middle of the letter shape. That is, the first layer connects to the second, but also to the last. The second layer connects to the third layer and the second-to-last layer, and so on until the middle layer which has no residual connections. While in recent years U-nets have seen widespread use for image segmentation problems in biomedical applications of machine learning [11, 12, 13, 14, 15, 16], their use in this paper is a novel application.

2.2 Baseline Method

There are no examples we were able to find in literature of relevant generative 'editing' networks that we are able to use a baseline comparison. Instead, we use a classification network as our baseline for performance on this task and compare our post-learning classification results to show the effectiveness of our generative network. To that end, we use the DeepDicomSort [4] architecture and apply transfer learning to replacing the dense head with a set of weights trained to classify the manufacturer of the machine used to take the scan instead of the type of scan being taken. We use the weights hosted on the project's github page, replacing the weight matrix of the first dense layer with a 14400x1024 matrix instead of a 1601x1024 matrix, and replace the second dense layer with a four-neuron classifier instead of the eight-neuron classifier due to the difference in target label size. We then train the model for three epochs on the padded 512x512 unmodified MRI scans, freezing all weights in the convolutional backbone. As an output, we return the predicted class label of the input MRI scan and train with a cross-entropy loss. This set of inputs, outputs, and losses is equivalent to our own classifier with the exception that the baseline method is trained purely on the unmodified MRI scans instead of the generator output. We obtain a loss of 0.0325, and a test accuracy of 0.9884, showing that even with a short duration of training, it is possible to achieve near-perfect classification on this task and to place greater context on the final results of our generator.

2.3 Dataset

We obtained images in the DICOM format as well as manufacturer information from the Cancer Imaging Archive (TCIA). We use the Brain Tumor Progression and TCGA-LGG datasets. Our preprocessing steps are inspired by DeepDicomSort's [4] handling of the same dataset. As a first step we retrieve necessary information such as patient, study date, series ID and manufacturer from the DICOM header. To save space and accelerate the training time, we then convert DICOM format images to the NIfTI format, which stores less extra information within the image file. The brain images might have different orientation so we adjust output direction and origin. Particularly, we focus on an Axial orientation. After constructing the whole 3D brain from a set of DICOM files, we slice the brain horizontally into NIfTI slices (Figure 1e). We only keep meaningful images that have fewer than 90% of values set to pure black or pure white, as a heuristic to remove slices of the very top or very bottom of the brain. Before training, we zero

pad all images to have the size as the largest image of the set (which in our case is 512x512), preferring to preserve the raw data as opposed to methods like interpolation and zero-centering all the training data. In some cases, due to memory limitations, we instead trained on a smaller image size of 350x350 or a reduced dataset of 5000 images. The final dataset contains 21987 slices and consists of four classes: GE Medical Systems, Siemens, Philips Medical Systems, or Imaging Biometrics. The class distributions are relatively balanced (ranging from 22% to 28%), and we follow a 70/20/10 rule to split the training/validation/testing sets. Due to the size of our dataset and the importance of maintaining every piece of semantic information from an image, we perform no other preprocessing or augmentation steps.

2.4 Hyperparameters

For our hyperparameter tuning process, we selected three variables to explore: generator learning rate, classifier learning rate, and beta (our tuning parameter for the two-component generator loss). We started our sweep for the classifier and generator around 5×10^{-3} as this was near the value used in the DeepDicomSort classifier and it had empirically shown strong results. We tested log space sweeps of approximately seven values per parameter, and achieved the best performance with the combined parameter set of 5×10^{-4} for the learning rate of the generator and the classifier, and a beta value of 0.02.

We observe that when training the classifier alone, memory space allowed us to use a batch size of 16 instead of a batch size of 8 as was necessary when training with both the generator and classifier together. This explains the factor of two difference observed between the initial classifier learning rate and the full model learning rate, as the inverse relationship from Smith and Le's work [17] on batch size and learning rate follows. We also note that the generator has a high sensitivity when modifying beta, with high values of the parameter resulting in a strong classifier that is unable to be fooled by the generator, while low values of the parameter quickly lead to a decay in image quality past a threshold point (examples shown in Appendix C). All training was performed using the Adam optimizer due to its empirically proven performance on a large range of deep learning tasks.

2.5 Metrics of GECO Performance

We evaluate GECO using three performance metrics: Classification Accuracy, Structural Similarity Index Measure (SSIM), and Peak Signal-to-Noise Ratio (PSNR). The first metric quantifies the ability of the generator to deartifact an image in terms of how often it is capable of fooling a classifier trained on the output distribution, while the second and third metrics quantify the ability of the generator to maintain the integrity of the input image throughout the de-artifacting process. Classification accuracy is calculated via running the generator on our test set, and feeding the generated images into the corresponding classifier to achieve class predictions. These predictions are then compared against the true labels of the images fed into the generator, and we return the simple ratio of correct guesses to total images. Since we have four classes, the 'optimal' accuracy we expect is 0.25, representing effectively random guessing.

SSIM is calculated by computing similarity between an image before and after it is processed by the generator in terms of luminance, contrast, and structure. These three sub-components are then multiplied together to return the SSIM value, which we desire to bring close to 1 as we hope to only perform minor edits to the image that do not significantly change any of these metrics. Finally, PSNR is calculated by comparing the square of the maximum value of the input image to the mean-squared error between the input image and the generated image. This metric is generally reported in terms of 'decibels' on a log scale, with a potentially infinite PSNR if the two images are identical.

3 Results

3.1 Assessment and Comparison of GAN Structures

We explored several structures for the GECO network based on a review of the techniques used for generative modelling and methods for improving accuracy and robustness. Our results in Table 1 on the core architecture show high SSIM and PSNR reflecting visual clarity in the generated images as seen in Figure 1, and an accuracy of 40% from the classifier showing a large dip in performance compared to the pretrained model. We note that in failed experiments such as hyperparameter choices which led to mode collapse, poor visual fidelity, or restrained editing,

the final classifiers retained approximately 96% accuracy. This indicates that the quantiative results showing GECO's efficacy do not reflect a lack of opportunity for the model to learn to classify these images. If the generator fails to train successfully for one of the defined and detectable issues listed above, the classification accuracy remains at the baseline of approximately 96%, but when the generator trains successfully the classifier's accuracy is reduced significantly.

Next, we tested two residual networks, both the multi-residual and full residual, and we observe similar results to the core generator but with slightly to moderately degraded performance in terms of image quality and deartifacting efficacy (Table 1). Numerically, these are a SSIM of 0.918, PSNR of 18.71, and classification accuracy of $\sim$44% in the full residual GAN (simply referred to as "Residual GAN" in Table 1), and SSIM of 0.933, PSNR of 19.93, and classification accuracy of $\sim$51% in the Multi-Residual GAN. In addition to the residual generators, we tested a three-player architecture that uses a discriminator network that classifies images as real or generated in addition to the standard classifier in order to teach the generator to create deartifacted images. This architecture frequently suffered from a mode collapse and instability during training. With the three-player architecture, we observed a slight improvement moving to two iterations of the classifier instead of one per generator iteration, however in the process we observed that the accuracy of the classifier during training did not increase as much compared to a one for one training regime. For comparison, the two iteration classifier achieved a maximum accuracy of 42% during training, while the one iteration classifier achieved a maximum accuracy of 47% during training. We also tested using three iterations for the classifier, but the SSIM decayed beneath the benchmark of 0.95 where image quality began to degrade.

From these results and the data in Table 1, we conclude that both the core architecture and three-player formulations provide similar results with proper training. However, due to the comparative robustness of each model and the additional complexity required in training three models, we select the core architecture as our optimal structure.

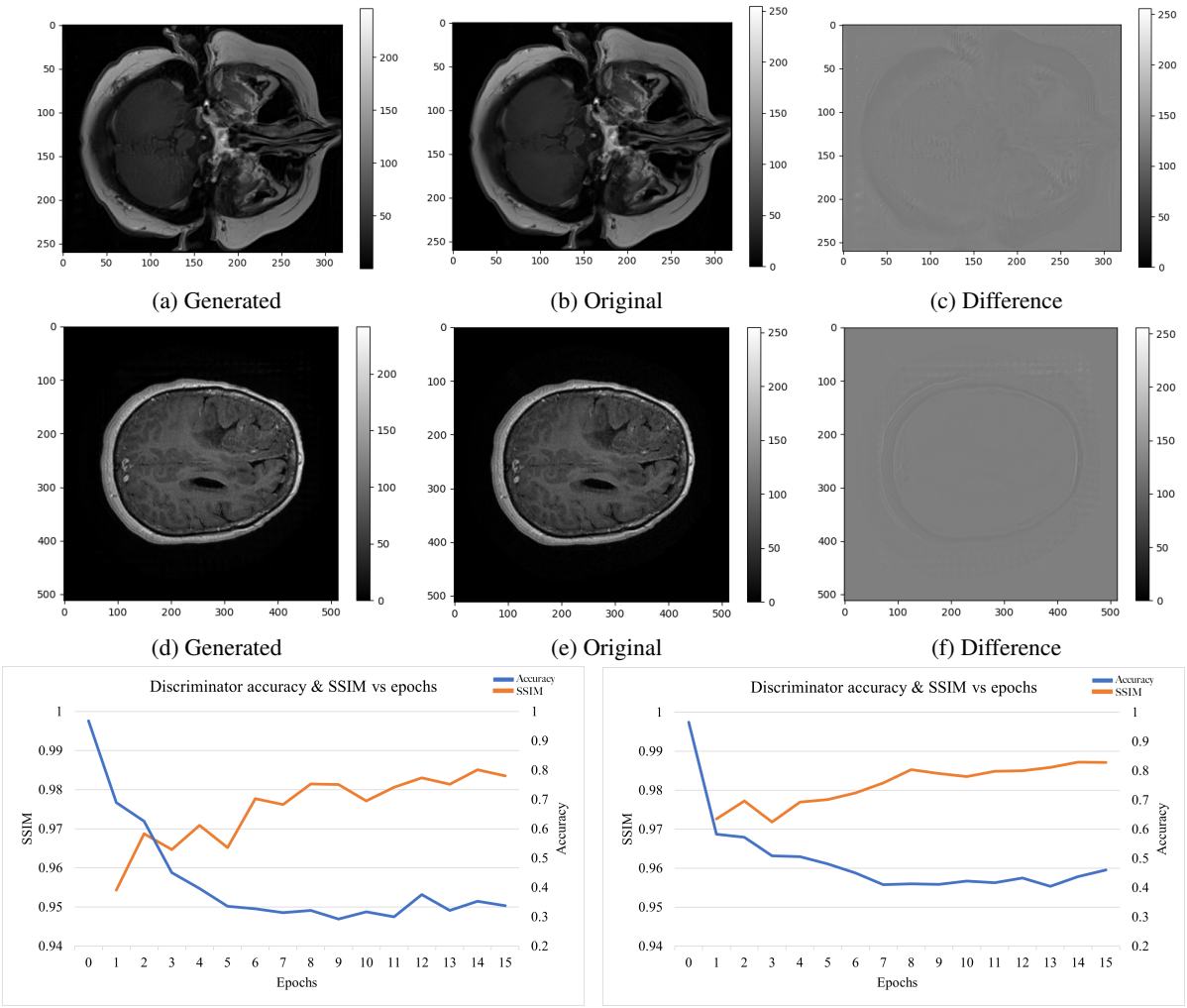
3.2 Evaluation of GECO's Efficacy in Hampering Classifiers

Next, we focus on analyzing the performance of the core architecture across both manufacturer-specific and field strength-specific experiments. In Figure 1 we show the SSIM and accuracy while training the model for both field strength and manufacturer, showing an increase in SSIM and a decrease in accuracy essentially equalling a random classifier for manufacturer (Figure 1(g)) and field strength (Figure 1(h)). In Figure 2, we observe that the distribution of predicted labels becomes normalized across the true labels for both field strength and manufacturer. Finally, Table 2 provides indication of the ability of this model to perform well across several modalities of imaging.

We also explore methods to confirm the quality of our results and the behaviors of our models. We plot visualizations of the edits made by our generator network in terms of the difference between the input image and the generated image. We observe that these edits are in general very minor adjustments as we expect, and the effect appears to be most prominent around the skull (matching with what we observe from the saliency map). Using Figure 1f as an example, we also observe minor adjustments within the brain itself following along the edges of the image, but the strongest effects are in regions that are unlikely to change the medical information of the image.

Experiment	Metrics		
	SSIM	PSNR	Acc.
Core GAN	0.975	37.75	0.401
Residual GAN	0.918	18.71	0.442
Multi-Residual GAN	0.933	19.93	0.514
Three Player GAN (1 iter)	0.982	32.58	0.396
Three Player GAN (2 iter)	0.981	33.27	0.378
Three Player GAN, no MSE	0.756	17.09	0.552

Table 1: Metrics on various successful architectures, with the best performance in each metric highlighted in dark blue, and the top three performances in each metric highlighted in light blue. SSIM is the "structural similarity index measure", PSNR is the peak signal-to-noise ratio, and "Acc." is the accuracy achieved by the classifier on images which had been run through GECO's obscuring generator, all as discussed in Section 2.5. Note: More information on three-player architectures can be found in Appendix D.

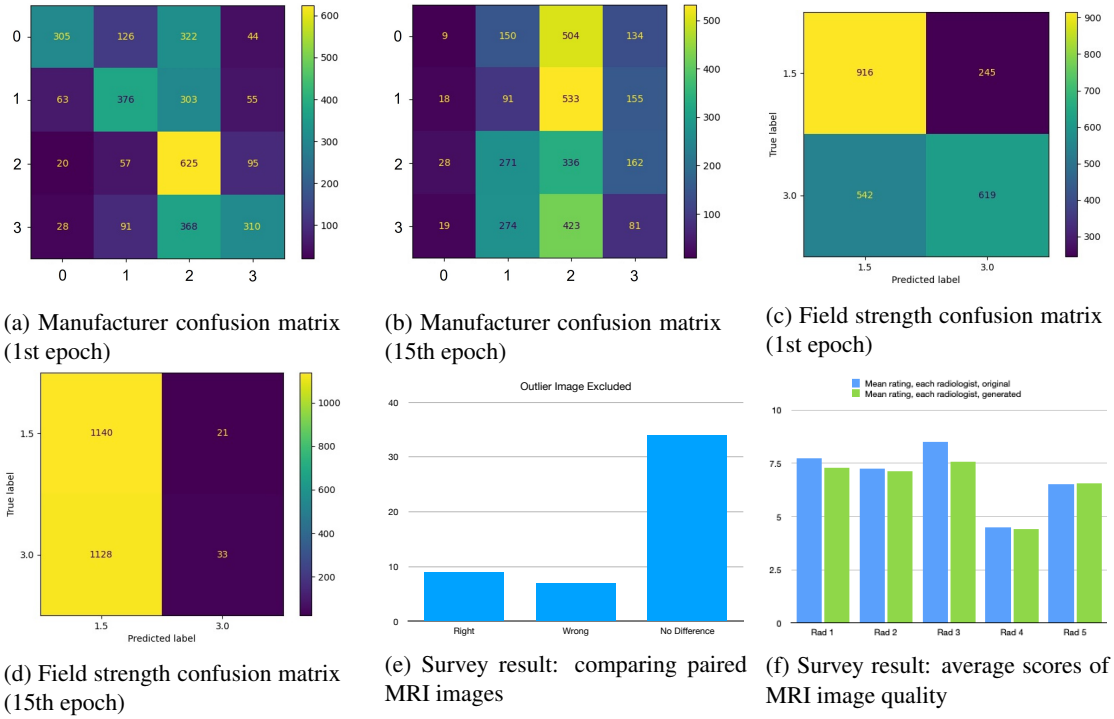


(a) Generated (b) Original (c) Difference

(d) Generated (e) Original (f) Difference

(g) (h)

Figure 1: Visualization of two MRI scan slices being passed through GECO, and plots of performance metrics over the course of training GECO. (a), (b), and (c): Results of training a generator to eliminate the difference between images of different manufacturers (top) and field strengths (middle). Here (a) is the original image, (b) is the edited image taken from the GECO generator after 15 epochs of training, and (c) is a grayscale map of the pixel-wise differences between the original and generated (edited) image. (d), (e), and (f): The same progression as (a) through (c), but with a different input image as an additional example. (g) and (h) Change of accuracy and SSIM over epochs of training. Orange is the SSIM and is shown on the left vertical axis of each figure, and blue is the classifier accuracy shown on the right-hand vertical axis in each. Metrics are averaged over 5 independent training runs. In (g) the generator is trained to eliminate the difference between four manufacturer classes (of note: one "manufacturer" class found in the DICOM files for some MRI images is "Imaging Biometrics". This is a software system rather than physical machines, but for the purposes of this paper is an equally valid class.). The best metric values achieved when evaluating the 5 trained GECO networks on a test dataset were 0.986 for SSIM and 0.256 for accuracy after epoch 15. Averaged over the final 5 epochs the metric values are 0.337 for the accuracy and 0.983 for the SSIM. In (h) the generator is trained to eliminate the difference between field strengths (1.5T and 3T). Averaged over final 5 epochs the metric values are 0.431 for the accuracy and 0.986 for the SSIM. In the case of field strength this was essentially equivalent to the values for the individual training runs.



(a) Manufacturer confusion matrix (1st epoch)

(b) Manufacturer confusion matrix (15th epoch)

(c) Field strength confusion matrix (1st epoch)

(d) Field strength confusion matrix (15th epoch)

(e) Survey result: comparing paired MRI images

(f) Survey result: average scores of MRI image quality

Figure 2: Confusion matrices demonstrating machine learning classifiers cannot effectively distinguish between original and GECO-edited generated MRI images, and plots of results from surveying experts which demonstrate they are similarly unable to effectively distinguish between original and generated images. (a) and (b) Manufacturer confusion matrices, respectively for epochs 1 and 15. (c) and (d) Field strength classifier confusion matrices, respectively for epochs 1 and 15. For both cases, we observe that after training each true label has a similar distribution of predicted labels. The classes above for the manufacturers are: 0 – GE Medical Systems, 1 – Imaging Biometrics, 2 – Philips, 3 - Siemens, all corresponding to how they appear in the DICOM files. (e) Summary of survey results when a group of 5 radiologists were asked to compare paired MRI images and the results of those same images being fed through a GECO trained to remove manufacturer artefacts. The rediologists could select from "Left", "Right", and "No Difference". Because image pairs were presented with the original and generated in random order with respect to left and right, we have translated the labels into "Right" and "Wrong" to show the results over 10 pairs. (f) Summary of additional results from the same survey as (e), now showing differences in average scores of MRI image quality, in a separate section of the survey with half original and half GECO-edited images. Radiologists scored image quality on a scale from 1-10 with 1 the worst and 10 the best. Results are shown radiologist by radiologist, to account for variances in the baselines for different survey respondents. For example, Radiologist 4 ("Rad 4")'s average score for both original and edited images is less than 2/3 that of all others'.

3.3 Expert Evaluation of GECO Performance by Radiologists

During the training process, we observed that when SSIM dips below approximately 0.95 we begin to see irregularities in the generated image that are clear to the naked eye. Conversely for SSIM values at or above that estimated threshold the two images appear virtually indistinct. While this level of analysis was suitable for the iterative process of model experimentation and construction, the final say in the success of our model's ability to maintain the semantically meaningful portions of a scan must be provided by trained radiologists. Thus, we collected data via survey asking experts to determine which scan they preferred between the unmodified and GECO-processed images as well as asking five neuro-radiologists to compare and score the visual quality of both types of images. We show our results in Figure 2.

From Figure 2e we observe that the large majority of cases show no significant preference between the provided images, and that in the cases where a preference is exhibited it is largely random which of the two images is preferred. Note that we refer to the original image as 'right' and the generated image as 'wrong' as we assume that our generator

has not actually improved the image quality in any meaningful way, since it was not trained on any such task during this process.

This data is further backed up by Figure 2f, from which we observe that when asked to rate the quality of MRI scans on a scale of 1-10 our generated images achieved very nearly the same scores on average as unprocessed images. There is an overall very minor trend of generated images being perceived as slightly lower quality, but aside from radiologist 3 the scale of the effect is negligible. Thus, we conclude that our model has succeeded at its task of removing identifying artifacts without significantly impacting the image's value as a diagnostic tool.

4 Discussion

In this work, we show that GECO is able to de-artifact MRI brain images by achieving a drop in classifier accuracy from ∼97% to ∼26% on the task of MRI manufacturer identification while improving the visual quality of images as measured by SSIM and PSNR. Similarly, for MRI field strength identification, GECO took the accuracy of a classifier from 97% to approximately 50% - and for that problem this was achieved in only a few epochs of training. This faster learning likely due to the fact that solving a two-class problem requires less data than a four-class problem in the case of removing manufacturer-specific artifacts. The magnitude of these artifact detection accuracy drops, indicates the generator network's strong efficacy in removing artifacts from scans, and in fact the final result is nearly equivalent to a classifier that is randomly guessing. Our architecture is particularly exciting given this level of efficacy was achieved with a relatively small number of convolutional layers in the model.

Applying GECO to similar problems of complex artifacts in CT and X-ray images are clear directions for follow-up research, and a more ambitious extension would be treating certain subtle physiological features as "artifacts". For example, there may be cases where researchers would like to study one physiological or clinical feature, but a separate gene representing a spurious correlation produces subtle variation in brain architecture which are not visible to the naked eye yet still confound the efficacy of computer vision methods. In such a situation it may be possible to use GECO or a variant thereof to reduce the amount of information a machine learning system could possibly extract from those spurious correlation of variants of the gene, enabling the ML methods to more effectively learn the systems or processes which researchers and clinicians are interested in. Similar applications could be found in studying patients where separate disorders correlate with the feature(s) being studied, such as early-stage diabetic damage changing the appearance and physiology of organs subtly. In short, minimizing the extent to which machine learning methods can "shortcut" past the actual features of interest is likely to have broad applications in medical imaging.

Though here we have demonstrated GECO's ability to clear spurious correlations from medical imaging data for computer vision applications, there is great potential for applications of this method to solve similar problems with other data types. We expect that a broader range of potential applications may exist beyond medical imaging. This approach of generatively filtering spurious information, or future approaches based upon it, could not only be used for computer vision applications in various fields, but also in natural language processing. For example, the work of Koppel et al. [18] and Rocha et al. [19] have demonstrated that NLP systems can accurately match individuals to even short pieces of text. With a generator modified for text data, a modification to GECO could potentially obscure the identities of authors. Further applications could potentially include difference-in-differences analyses (among other approaches) for causal inference in clinical trials and social sciences by aiding in accounting for externalities with

Modality	Accuracy	SSIM	Number of images			
			Siemens	Philips	Imaging Biometrics	GE Medical Systems
FLAIR	0.101	0.983	145	39	120	143
OTHER	0.210	0.986	243	359	333	453
T1 POST	0.351	0.990	83	49	0	22
T1 PRE	0.347	0.984	71	71	232	47
T2	0.305	0.988	186	85	112	132
T2 FLAIR	0.292	0.981	2	22	0	0

Table 2: Metrics obtained on the manufacturer artifacts removal task reported by modalities. The number of samples from each manufacturer are given for reference.

complex and/or unknown distributions - especially in data like satellite or medical images.

To summarize, the approach at the core of GECO has potential for a setting where there is benefit in the precisely targeted elimination of specific correlates whose labels are known, yet whose distributions are not. It's strength lies in achieving high efficacy without excessive complexity. The GECO system's sole objective is to pass on as much data to downstream systems as possible while 'erasing' information which developers and users of other ML systems would like ignored. This 'restricts' downstream systems from detecting spurious correlations, and thus 'frees' them to learn the true distributions of interest.

Competing Interests

Michael Zeineh reports receiving funding from GE. All other authors report no conflict of interest.

References

[1] John R. Zech, Marcus A. Badgeley, Manway Liu, Anthony B. Costa, Joseph J. Titano, and Eric K. Oermann. Confounding variables can degrade generalization performance of radiological deep learning models. *CoRR*, abs/1807.00431, 2018. 2, 15

[2] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Yi Ding, Aarti Bagul, Curtis Langlotz, Katie S. Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *CoRR*, abs/1711.05225, 2017. 2

[3] Varun Gulshan, Lily Peng, Marc Coram, Martin C. Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, Ramasamy Kim, Rajiv Raman, Philip C. Nelson, Jessica L. Mega, and Dale R. Webster. Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. *JAMA*, 316(22):2402–2410, 12 2016. 2

[4] Sebastian Voort, Marion Smits, and Stefan Klein. Deepdicomsort: An automatic sorting algorithm for brain magnetic resonance imaging data. *Neuroinformatics*, 19, 01 2021. 2, 3, 5

[5] U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015. 2

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020. 2

[7] Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic photo adjustment using deep learning. *CoRR*, abs/1412.7725, 2014. 2

[8] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. Generative visual manipulation on the natural image manifold. *CoRR*, abs/1609.03552, 2016. 2

[9] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. 3

[10] Chongxuan Li, Kun Xu, Jun Zhu, and Bo Zhang. Triple generative adversarial nets, 2017. 3, 4

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, 2015. 5

[12] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, 2016. 5

[13] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A.W.M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A survey on deep learning in medical image analysis. *Medical Image Analysis*, 42:60–88, 2017. 5

[14] Fabian Isensee, Jens Petersen, André Klein, David Zimmerer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Köhler, Tobias Norajitra, Sebastian J. Wirkert, and Klaus H. Maier-Hein. nnu-net: Self-adapting framework for u-net-based medical image segmentation. 2018. 5

[15] Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. 2018. 5

[16] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. International Workshop on Deep Learning in Medical Image Analysis, 2018. 5

[17] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. *CoRR*, abs/1710.06451, 2017. 6

[18] Moshe Koppel and Yaron Winter. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 2013. 10

[19] Anderson Rocha, Walter J. Scheirer, Christopher W. Forstall, Forstall Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne R. B. Carvalho, and Efstathios Stamatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, 2017. 10

[20] Clark K;Vendt B;Smith K;Freymann J;Kirby J;Koppel P;Moore S;Phillips S;Maffitt D;Pringle M;Tarbox L;Prior F. The cancer imaging archive (tcia): maintaining and operating a public information repository. *Journal of digital imaging*, 26:1045–1057, 12 2013. 21

[21] KM Schmainda and M Prah. Data from brain-tumor-progression. *The Cancer Imaging Archive*, 2018. 21

[22] N. Pedano, A. E. Flanders, L. Scarpace, T. Mikkelsen J. M. Eschbacher, B. Hermes, V. Sisneros, J. Barnholtz, and Q. Ostrom. Radiology data from the cancer genome atlas low grade glioma [tcga-lgg] collection. *The Cancer Imaging Archive*, 2016. 21

[23] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 21

[24] 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, September 2020. 21

[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. 21

[26] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2:e453, 6 2014. 21

[27] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com. 21

[28] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. 21

[29] 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. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 21

[30] Nibabel. *Neuroimaging in Python - NiBabel 3.2.1 100.g9fdf5e3e documentation*. 21

[31] Ziv Yaniv, Bradley C. Lowekamp, Hans J. Johnson, and Richard Beare. Simpleitk image-analysis notebooks: a collaborative environment for education and reproducible research, Sep 2019. 21

[32] Darcy Mason, Scaramallion, Rhaxton, Mrbean-Bremen, Jonathan Suever, Vanessasaurus, Guillaume Lemaitre, Dimitri Papadopoulos Orfanos, Aditya Panchal, Alex Rothberg, and et al. pydicom, Aug 2020. 21

A Extended Data Figure 1: Background Information

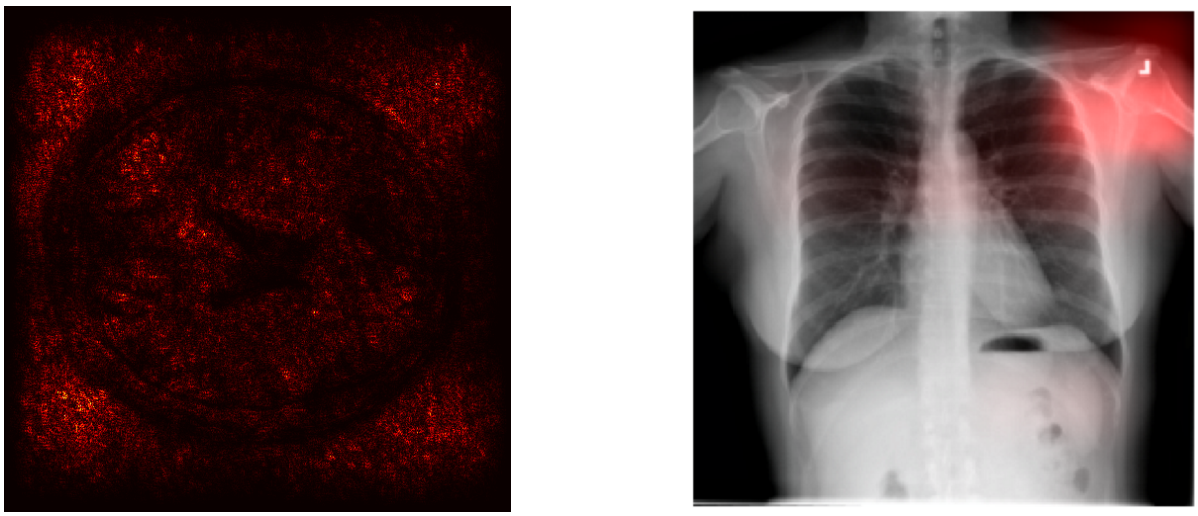


Figure 3: Extended Data Figure 1: Background Information. At left: saliency map taken from our baseline classifier. Note the attention paid to areas outside of the skull (central ball-shaped region). At right: saliency map taken from John Zech [1]. Here the classifier picks up on a tag visible in the CT scan.

B Extended Data Figure 2: Model Architecture

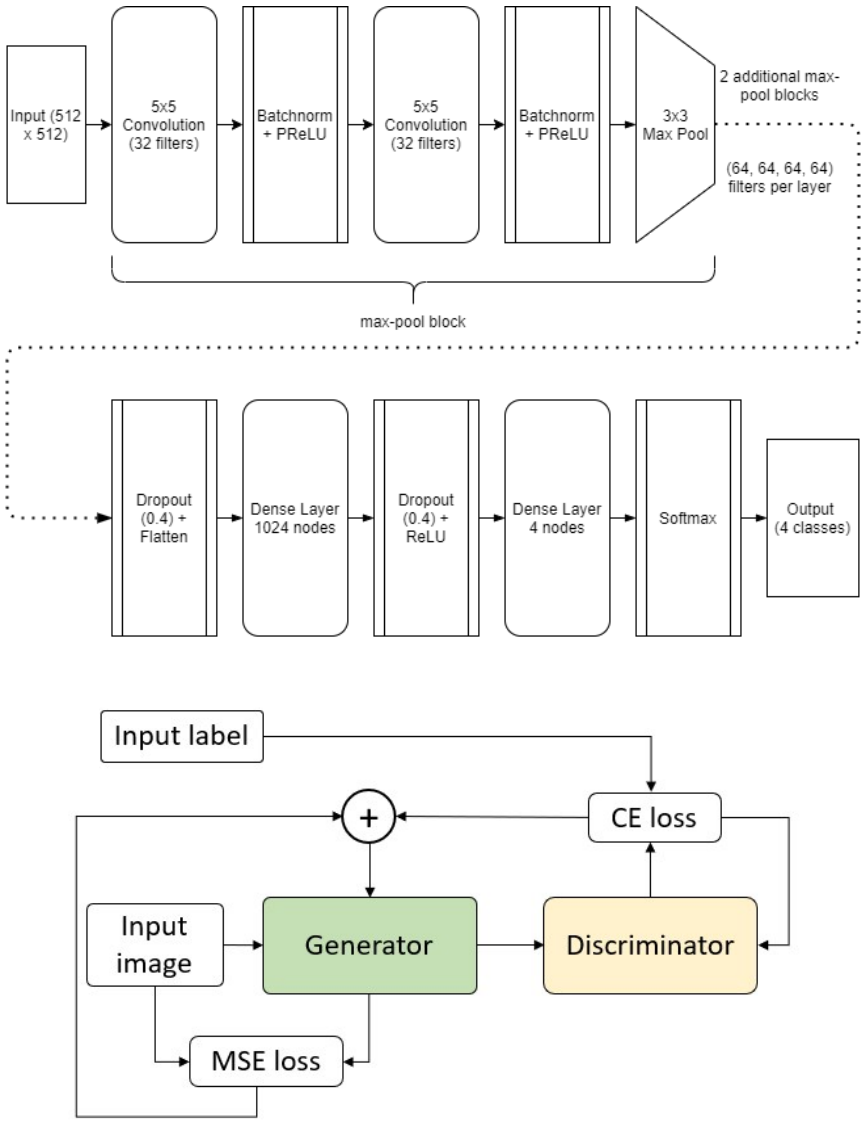
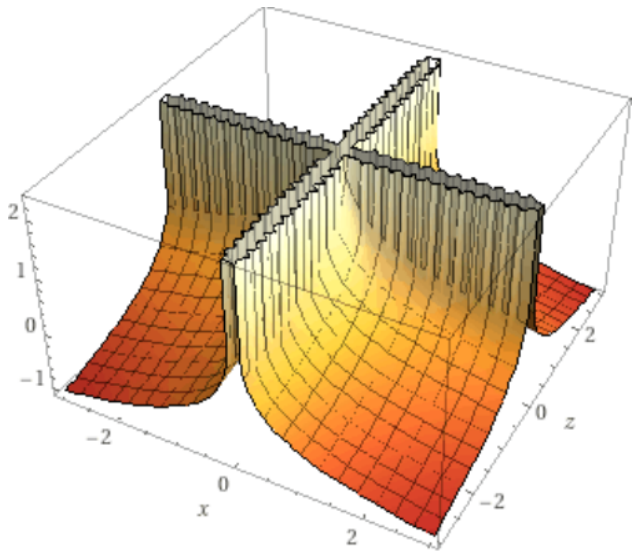


Figure 4: Extended Data Figure 2: Model Architecture. Top: layerwise architecture of our classifier, architecture initially from DeepDicomSort. Bottom: Depiction of the general structure for our GAN architecture.



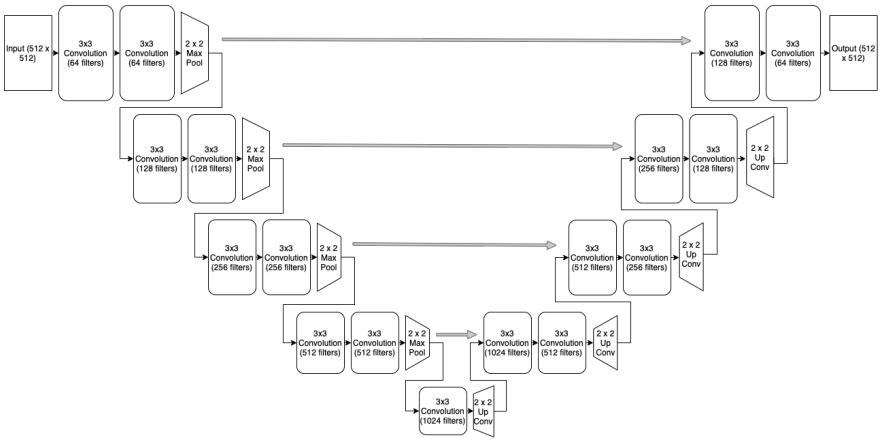


Figure 5: Top: two-class example of our inverse cross-entropy loss structure. At bottom: layerwise architecture of our generator. Note that the final convolutional layer uses a different activation function in order to squeeze the output to the expected range.

C Extended Data Figure 3: Sample Results

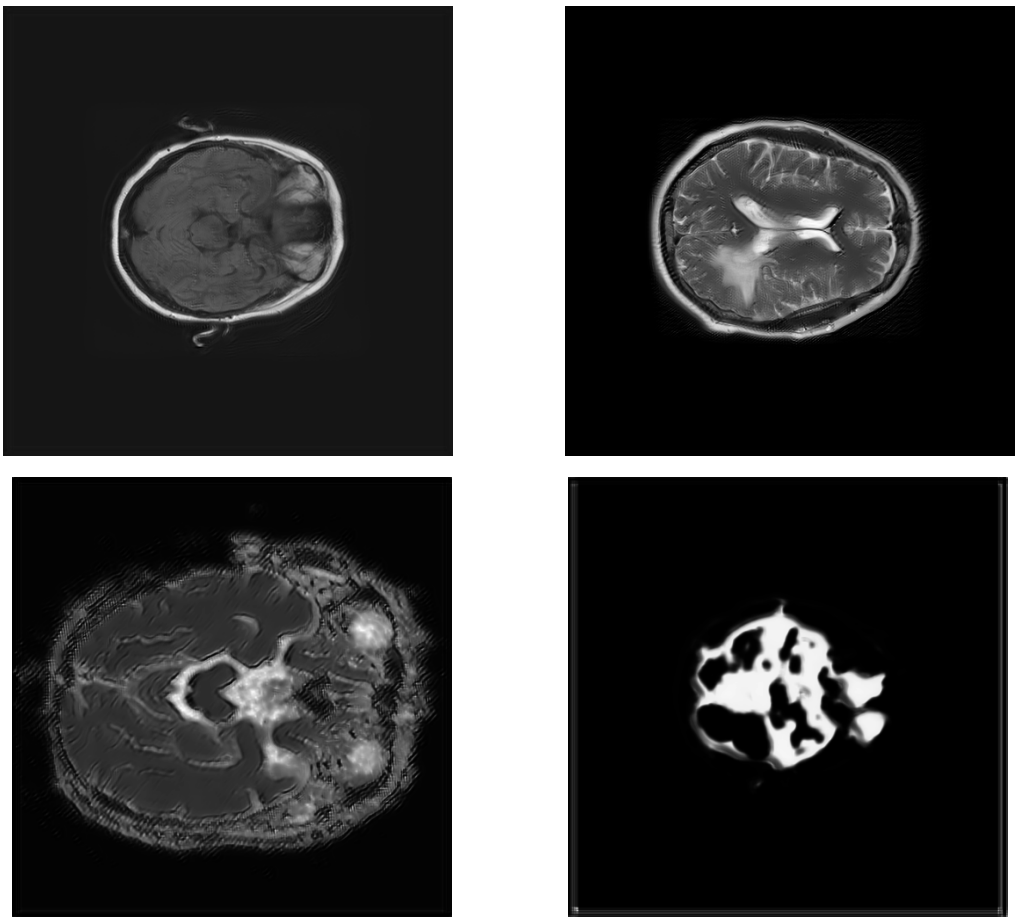


Figure 6: Extended Data Figure 3: Sample Results. At top left: image taken from resnet generator after 30 epochs of training. At top right: sample image from generator with beta = 0.007. Note the fringes around the exterior of the skull, introduced by the generator. At bottom left: sample image from generator with beta = 0.004. We see the fringe effect of the generator stronger here, as beta falls. At bottom right: sample image from generator with beta = 1.2e-4. At this point image quality decays entirely, highlighting the importance of the MSE loss term.

D Extended Data Figure 4: Three-player Results

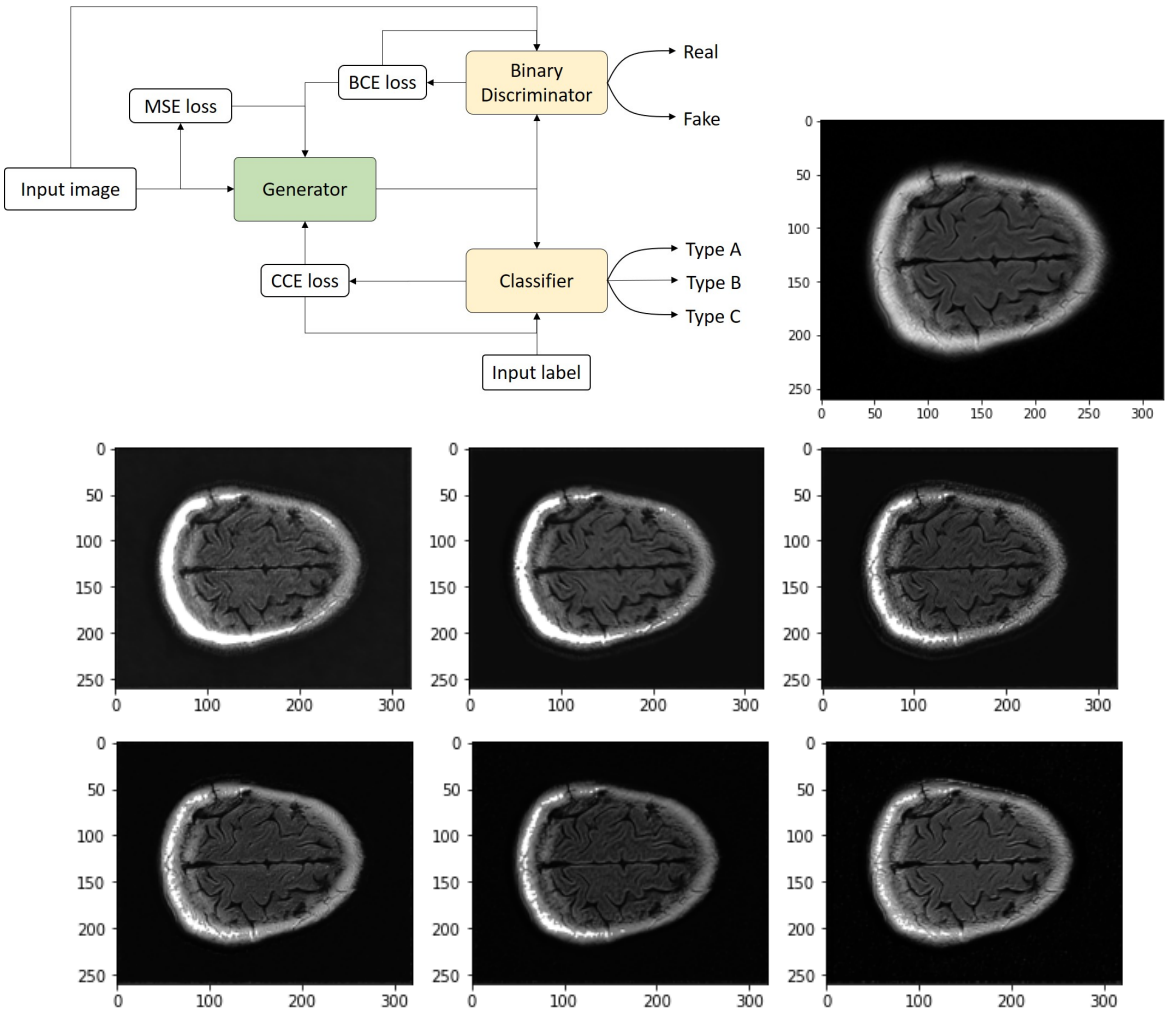
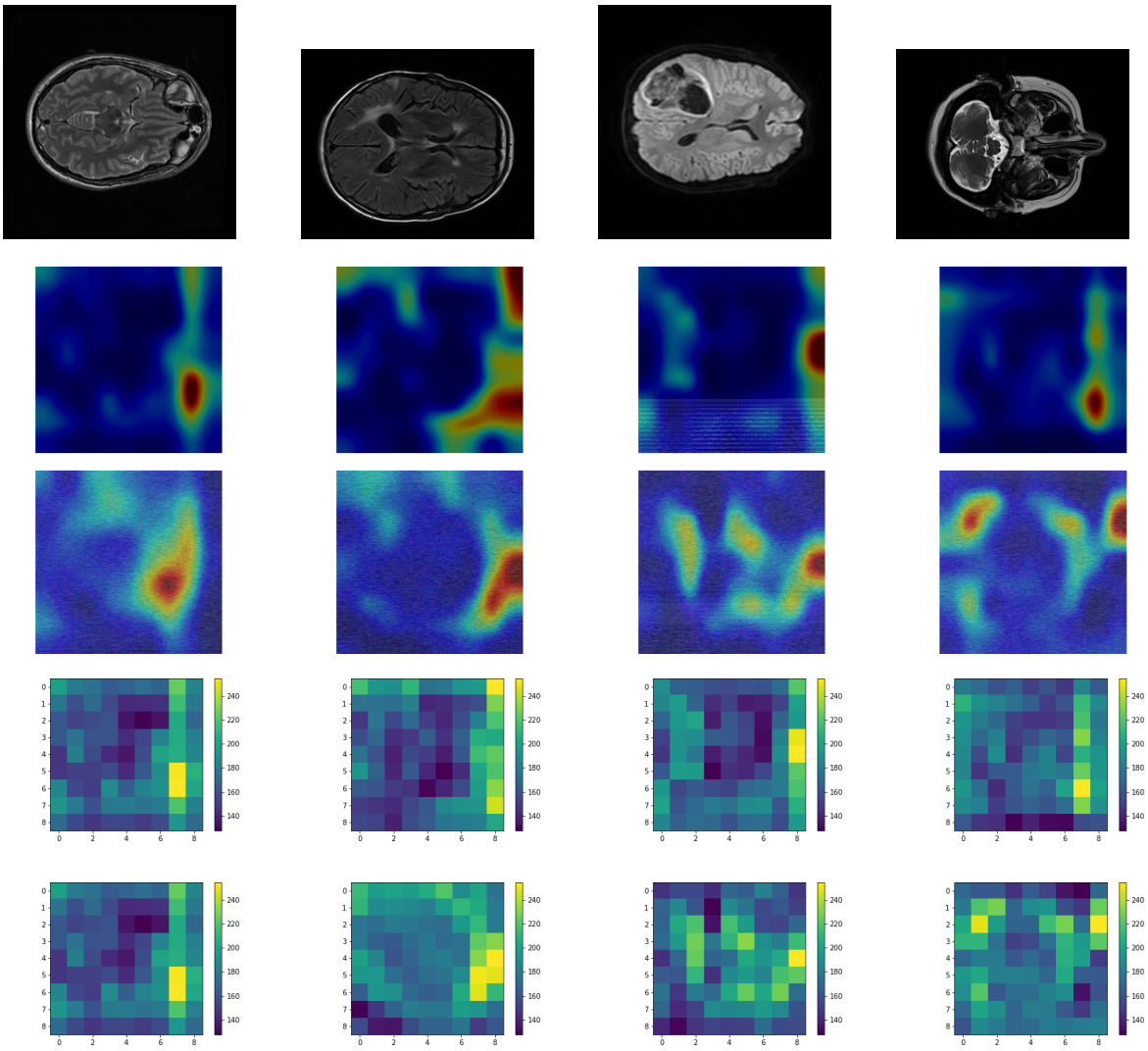


Figure 7: Extended Data Figure 4: Three-player Results. At top left: three-player architecture schematic. At top right: reference unmodified image for panels below. At bottom: series of six generated images taken over 30 epochs of training the three-player network, with 2 classifier steps per generator epoch. Epoch increases from left to right, top to bottom.

E Extended Data Figure 5: Class Activation Mapping



(i) GE (j) Imaging Bio. (k) Philips (l) Siemens

Figure 8: Extended Data Figure 5: Class Activation Mapping. At top: raw images used to generate the heat maps in the following panels. Image labels are as follows: GE: TCGA-HT-7605, Imaging Bio.: PGBM-009, Philips: TCGA-CS-6666, Siemens: TCGA-FG-A713. At middle: class activation maps of a pretrained classifier overlaid onto MRI scan on four machine type samples. Top row shows the CAM for raw images, bottom row for generated images. Note the change in location of activations. At bottom: class activation maps/Heat Map on four machine type samples. Top row shows the heat map for raw images, bottom row shows the heat map for generated images. The order is the same between all panels.

F Contributions and Acknowledgements

In developing this project, we used public weights from DeepDicomSort as well as their image preprocessing pipeline as a basis for our own work. We also appreciate all the consultation received from Stanford CS231N teaching team: Ranjay Krishna, Rachel Gardner, Chris Waites and Danfei Xu. We also made use of GPU resources from the Stanford Sherlock cluster which one of the team members had access to. Three Tesla V100 32Gb GPUs were used to run the experiments, machines with less powerful hardware were problematic to train our models on. Some of the experiments were run on Google Cloud Tesla K80 GPU.

G Data Sets and Packages

Data: The Cancer Imaging Archive [20], Brain Tumor Progression [21], TCGA-LGG [22]

Packages: Pytorch [23], Numpy [24], Scikit-learn [25], Scikit-image [26] Weight and biases [27], , Matplotlib [28], Tensorflow [29], Nibabel [30], SimpleITK [31], Pydicom [32]