Face Anonymization

Molly Shillabeer, Adam Hirani, Michael Amorim, Daniella Ruisendaal

1. Motivation

There has recently been a push for more online privacy in government regulation, and many countries are introducing legislation to protect their citizens' privacy. For example, the General Data Protection Regulation in the European Union (GDPR) came into effect in 2018 and requires consent from individuals for the use of personal data that can identify them. When producing a dataset with hundreds of thousands or millions of images, getting consent from everyone in those images would be a very time-consuming task. To be more efficient, there is a need for AI researchers to have image datasets that include people's faces without revealing their identities, so they can continue working on computer vision tasks while complying with regulations. Face anonymization can be used to accomplish this task by using artificial intelligence to alter people's faces and make them unrecognizable, thereby complying with GDPR because the data is no longer identifiable. This task has an enormous social and economic impact. Socially, face anonymization protects people's privacy, making it safer for people to use the internet without worrying about their images being used without consent. Economically, face anonymization makes it possible for researchers to comply with new government restrictions, allowing research and development to continue.

2. Problem Description

Privacy is the main goal of face anonymization. Preserving peoples' privacy is, as explained above, becoming an increasing issue for the public. The goal of these models is to create realistic faces that cannot be identified when compared to the original image. They can struggle to be accurate in doing that when certain things such as irregular poses, complex backgrounds, or high occlusion are present in the images. The computational cost must be considered when deciding which model to use – the lower the better for large data sets. A limiter to the widespread use of face anonymizers is having to find advanced enough models to use for each scenario they're needed for that do their job well. The models we are covering in this paper are just some of the ones that we have found to do a great job at this.

3. Contribution

3.1 Talk and Slides

For the *Talk and Slides* portion of the project we assigned each section to a distinct group member. The group discussed the main talking points for the particular section, and this member was responsible for displaying and presenting the information. The idea behind this was to minimize video cuts from one person to another to maintain consistency. For the model and implementation section, we all completed this individually.

Molly Shillabeer	Adam Hirani	Michael Armorim	Daniella Ruisendaal
● Intro/Agenda ● Motivation ● Conclusion ● StarGAN	● Existing Work ● DCGAN	● Datasets ● ALAE	● Problem Description ● Deep Privacy

3.2 Written Report

For the *Written Report* portion of the project we distributed the work in a similar manner to the *Talk and Slides* portion since we found it worked well for our group. Again, we discussed the main talking points for the group slides and then assigned someone as the designated writer. For the *Related Work* and *Results and Discussion* sections, we completed these individually and assigned one group member to assemble and clean up the content. We assigned the similar topics from the previous assignment to the same members since they already had a deep understanding of it. For the implementation portion, we all completed this individually.

Molly Shillabeer	Adam Hirani	Michael Armorim	Daniella Ruisendaal
● Motivation ● Future Work/ Conclusion ● StarGAN	● Contribution ● DCGAN	● Datasets ● ALAE	● Problem Description ● Deep Privacy

4. Related Work

The majority of our models held up pretty well against the other models from papers that we have reviewed in our literature review submissions. There is some overlap between each person's chosen works, however, the comparisons of results are different. The first paper we reviewed introduced StarGAN, a general adversarial network created for efficient multi-domain image-to-image translation (Choi et al., 2017). Image-to-image (I2I) translation involves changing one aspect of an image, for example, altering a person's hair colour or facial expression. StarGAN provides an efficient way to perform multiple translations at once by using only 1 model for any number of translations. The datasets used were CelebA and RaFD. Both contain images of human faces with labels for various binary attributes, like 'brown hair' or 'happy' for each image. These attributes made up the set of possible translations the model could perform. StarGAN is very good at multi-domain translation, creating high quality fake images with realistic faces while maintaining identifying facial features. The main limitation of this method is that it creates lower quality images when 3 or more translations are done at once. In this case, the resulting images can have blurring, gray backgrounds, and unrealistic faces. As well, the datasets used only include high-quality headshots from a limited number of angles, so this model is likely to struggle with any occlusions blocking the face or extreme poses. Lastly, since there is a constraint on possible target attributes, there is not much fine-grain control over the generated images, which may limit the model's usability. The model was evaluated using a panel of about 100 judges who compared four I2I models and selected the best fake image for about 40 comparisons. StarGAN won for each category of translation (hair colour, gender, age, and combinations of these). For the 3-domain translation of hair colour, gender, and age, StarGAN received 52.2% of votes while the next best model only got 20.3% of votes. Molly modified this model for use in face anonymization by combining four translations into one image, effectively obscuring the identities of faces.

The next paper covered is the DCGAN which tends to generate poor results. Altering the code to reduce the image quality reduced the quality of the results drastically. The two articles/models to compare to are: CIAGAN and GAN. The third article/model was created to detect deepfake images so it will not be included in the comparison. In terms of strengths, the DCGAN model runs drastically faster than both the CIAGAN and GAN models. This comes with drawbacks as the DCGAN model has issues producing accurate reproductions of human

faces. The CIAGAN and GAN models take a long amount of time to train (with little computational power) but generate an accurate image of a human face. The CIAGAN paper used CelebA, MOTS, LFW as the training datasets, while the GAN paper crawled Flikr for images of faces. For training the DCGAN model, CelebA was used as it is a smaller dataset with no super high-resolution images. This is one of the other reasons why the CIAGAN and GAN works were able to create more accurate results.

The third paper we reviewed introduced an autoencoder. This autoencoder was built off a novel autoencoder architecture by modifying the original GAN paradigm. Latent space then is also assumed to be the same between 2 sets of nodes. This architecture is split up into an encoder and a generator where the encoder maps the data from the input onto space characterized by the latent distribution. The generator then maps the latent codes onto a space that can be described as data distribution. This approach to the architecture is called Adversarial Latent AutoEncoder (ALAE) and later this paper builds on that to make StyleALAE. The ALAE was used to build another autoencoder that uses a StyleGAN-based generator. At every layer the encoder is designed symmetrically to ensure that the same style of information is extracted. This is done by inserting Instance Normalization layers, which are used to get instance averages and deviation on every channel This model is then trained on Flikr-Faces-High-Quality dataset to generate accurate images of human faces and starts at 4x4 resolution and builds up to 1024x1024 resolution. This is done by training the model on 147 epochs where 18 of them are at 1024x1024 resolution input size. Between growing the size there are 500k images used during the transition and then another 500k used for training stabilization. This form of training and the model allows the StyleALAE to generate almost perfect new faces where it is impossible to tell the original picture.

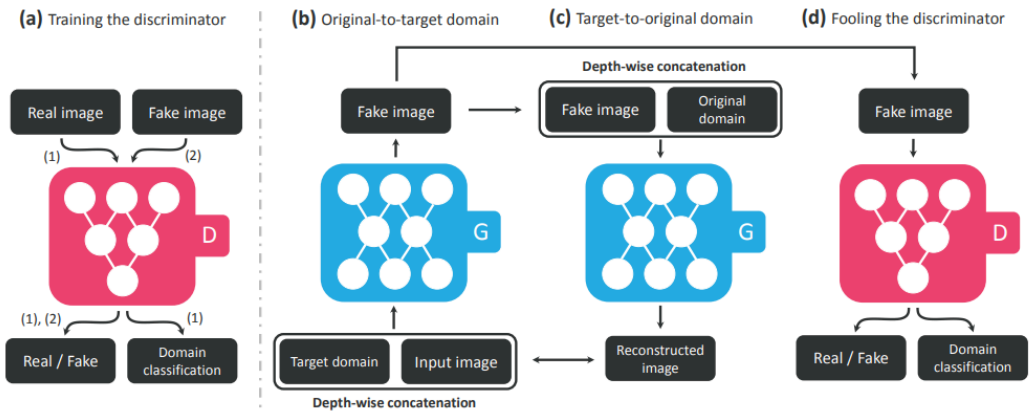
The fourth and final paper that we examined was for the DeepPrivacy model created by researchers at the Norwegian University of Science and Technology. DeepPrivacy is a Conditional Generative Adversarial Network, or CGAN. It was created as a novel architecture for “automatically anonymiz[ing] faces in images while retaining the original data distribution” (Hukkelas et al., 2019). This model was designed to run on FDF (Flickr Diverse Dataset), but the dataset we used was the CelebA dataset and it works just as well using this dataset. The biggest limitation of this dataset is the time it takes to run. It can take over 2 minutes to process just 10 images, so it is understandable why such a large dataset like FDF took 17 days to run on 2 high-end GPUs and a subset of CelebA’s dataset of approximately 1,500 images took over 9 hours to run on one GPU. Since my literature review was not on the same topic as this project, I will not be comparing the model I chose to any of the paper’s I discussed in that review. Instead, I will compare it with papers relevant to the model. This model can be compared to the GAN created by Ren *et al*. and the non-conditional GAN created by Karras *et al*. The GAN was created to anonymize images by altering every pixel of the original image, meanwhile DeepPrivacy’s CGAN can anonymize an image without ever seeing the original, making it much more privacy-secure than the GAN. The GAN was 94.75% accurate meanwhile the CGAN was 99.3% accurate. The non-conditional GAN was used heavily in the creation of DeepPrivacy, so it can be expected to have similar features and results. They both use an equalized earning rate and progressive growing generator and discriminator. The quality of the non-conditional GAN’s results was “generally high compared to earlier work on GANs” (Karras *et al*., 2018), but no specific numbers were given in this paper.

5. Dataset

There are many datasets that can be used in creating face anonymization algorithms. Some of these datasets are FFHQ, CelebA-HQ, VGGFace2, JAFFE, LFW, as well as CelebA, which is the dataset we chose. CelebA seemed like the best choice for us to train our models for many reasons. One of those is the resolution of the photos. FFHQ and CelebA-HQ are both high-quality photo datasets with images of 1024x1024 resolution. Meanwhile, the images in CelebA are only 178x218 resolution which allows quicker training and the ability to process larger batches. However, we sacrificed the better accuracy we would have gotten if we trained the models on better-quality photos. CelebA also had a large number of images compared to other datasets. Other datasets had around 70,000 images or less but CelebA has 202,599 images. A few datasets have millions of images, like VGGFace2, but this seemed excessive considering our computational limits. All images in CelebA are partially preprocessed with 40 binary attribute annotations and a large amount of diversity in race and gender. It contains 202,599 images of 10,177 celebrities with large pose variations and background clutter. Few images have occlusions blocking the face - which later became an area in which our models struggled.

6. Implementation

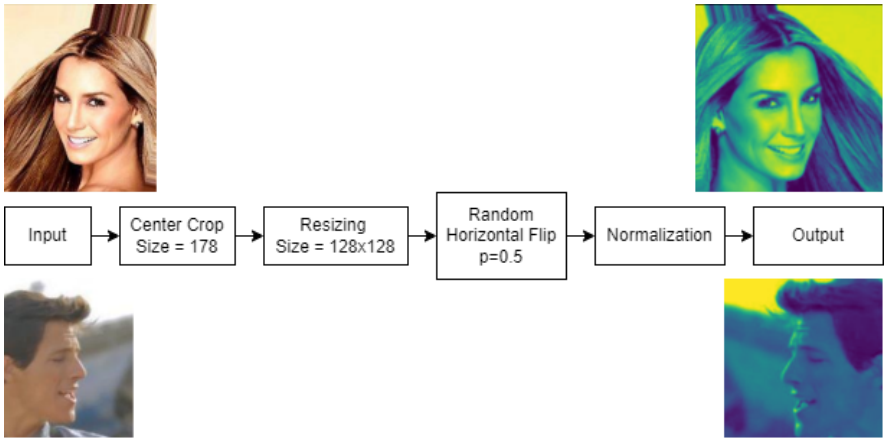
6.1 StarGAN - Molly Shillabeer



This model consists of two main blocks: a generator and a discriminator. The discriminator learns to distinguish real images from fake ones and penalizes the generator when it creates unrealistic images. The generator learns to create fake images using feedback from the discriminator, making the results more realistic. The generator is made of two convolutional layers, followed by 6 residual blocks, ending with two transpose convolutional layers. Each residual block contains two convolutional layers with skip connections, helping to protect the network from degradation. Every layer in the generator uses ReLU for activation. The discriminator is connected to the end of the generator and consists of 6 convolutional layers, each using leaky ReLU for activation. The kernels used in each convolutional layer vary in size, stride, and padding. While training, the target attributes are randomly selected for each batch, preventing overfitting for any one translation. The generator is updated after 5 discriminator updates, using Adam optimization for gradient descent.

I've modified this model for use in face anonymization by combining multiple translations into one photo, rather than creating a new image for each translation. I modified the testing phase so that for each photo, four attributes are flipped: age, gender, nose, and chin. I chose these properties because they are effective at making people unrecognizable when changed. So, if an image is tagged as having a young woman with a pointy nose and double chin in it, the model will alter that image to be an old man without a pointy nose or double chin. Since the model struggled with objects blocking the face, I reduced the initial kernel size from 7 to 5 and the padding from 3 to 2 to keep more detail in the early training phases. I've also reduced the maximum number of iterations from 200,000 to 21,000 for efficiency, as the original model would have taken 2 weeks to run on my hardware.

6.1.1 Data Preprocessing



The CelebA dataset is already mostly processed, with bounding boxes and attribute labels, so there was not much preprocessing to be done. The images were cropped so that the faces are centered, then they were resized to 128 x 128 pixels. Some randomly-selected images were flipped horizontally to augment the dataset, making the model more robust by providing more possible poses. Lastly, the images are converted to tensors with a possible range of 0-1 instead of 0-255 and normalized to have a mean of 0.5 and standard deviation of 0.5 to reduce noise in the input images.

6.1.2 Experimental Setup

This model was trained and tested on a Windows computer with one NVIDIA GPU. Training took 24 hours with this setup. The code was implemented with Python using VS Code and the

software packages used were PyTorch, Torchvision, and Numpy. PyTorch and Torchvision were used for data preprocessing, model implementation, training, and testing. Numpy was used for matrix manipulation, like setting the initial values in the kernels for convolution.

6.1.3 Training/Test/Validation

The model was trained with 21,000 iterations and this is the only stopping criteria. 2000 randomly selected images from the CelebA dataset were used for testing, and the remaining 200,599 images in the dataset were used for training. The results of testing were validated by visually inspecting the generated images compared to their original ones. The criteria I used for deciding when the images were good enough were checking that the majority of fake images were not identifiable, looked like real people, and were of high quality.

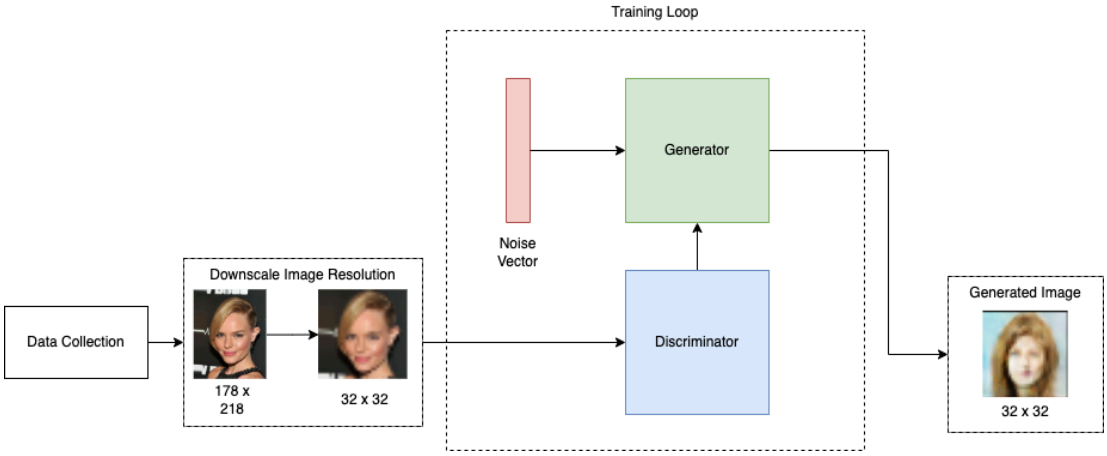
Training parameters and why they were chosen:

Parameter	Value	Reasoning
Batch Size	16	A batch size of 16 is suggested to be used when training a DCGAN model (Radford et al., 2015). This value seemed to be sufficient and still allowed the code to execute quicky
# of Colour Channels	3	The number of colour channels is set to 3 for colour images (RGB).
Image Size	128	An image size of 128 x 128 was used so that the resulting images wouldn't be overly pixelated.
Learning Rate	0.0001	A learning rate of 0.0001 with decay after 10,000 iterations was used. Reducing the learning rate near the end of training represents fine-tuning the image and results in better final images.
Beta 1, Beta 2	0.5, 0.999	The beta hyperparameters are used for Adam optimization. They control how important previous iterations are when performing gradient descent. Decreasing beta 1 to 0.5 slows down learning. They are multiplied together, so setting beta 2 to 0.999 prevents learning from slowing down too much.

6.2 DCGAN - Adam Hirani

This model was designed to generate new images using a batch of training data. In order to make the model compatible with the CelebA dataset, I modified it. The original model supported images with a resolution of 64x64 pixels, but I reduced the resolution of the dataset to 32x32 pixels and altered the image generator and discriminator to operate at this lower resolution. The design of this model is simple, as it utilizes both an image generator and an image discriminator. The generator employs five fractional-strided convolution layers to transform a high-level representation into a 64x64 colour image. The discriminator has been trained for image classification tasks and takes a 64x64 colour image as input, returning a scalar vector of extracted features. When combined, these components generate a new human face image.

6.2.1 Data Preprocessing



First the training data is collected and is downscaled from 128 x 128 to 32 x 32. It is then fed into the training loop where the model training takes place. The generator accepts a random noise vector as input and generates a picture, while the discriminator analyzes the quality of the resulting image and sends feedback to the generator to help it perform better. This approach is repeated until the generator creates pictures that the discriminator finds difficult to differentiate from genuine photos.

6.2.2 Experimental Setup

The computer specifications for this training model was run on a MacBook Pro with an M1 processor and 16GB of RAM. The M1 chip is equipped with a *Neural Engine* which helps boost performance when training a machine-learning model with no GPU. The code implementation format is a Jupyter Notebook which allowed me to use Google Colab for testing as well as my own computer with VSCODE and a local Jupyter server. For the code, 3 main libraries were used: Torch, Torchvision and Numpy.

6.2.3 Training/Test/Validation

Training Arguments		
Parameter	**Value**	**Reasoning**
Batch Size	128	A batch size of 128 is suggested to be used when training a DCGAN model (Radford et al., 2015). This value seemed to be sufficient and still allowed the code to execute quickly.
Image Size	32	An image size of 32 x32 was used so that this model would be able to run quickly on a computer that had less computing power.
Epochs	60	Training for 60 epochs seemed to produce good results in a shorter amount of time. When increasing the epochs higher than 60 the model reaches a point of diminishing returns.
Learning Rate	0.002	The paper stated that they found a learning rate of 0.001 to be too high so they used 0.002 instead (Radford et al., 2015). Both options were tested however, 0.002 produced a slightly better result.

Beta	0.5	The beta parameter is used for the Adam optimizers. Again, the paper describes that they tested different values and 0.5 seemed to stabilize the training (Radford et al., 2015).
NC (# of colour channels)	3	Number of colour channels is set to 3 so that it can handle colour images.

For this model, the stopping criteria was just when the number of epochs was reached. This model does not do any sort of validation. In the original paper they mention that on the CIFAR-10 dataset the model achieves 82.8% accuracy. For this they used a validation set of 10k images and used 1000 images for training (Radford et al., 2015). For the case of my implementation of the DCGAN, after making my extensions to the code it is able to utilize the entire CelebA dataset for training. For accuracy, it produces an image that is quite unlike a face but some distinct features are present. Most of the time, the created faces resemble actual faces but are usually incredibly distorted. Additionally, sometimes a picture would be produced that had no resemblance to a real face. This is because the facial characteristics it produced were uneven, such as the person's eyes, which were noticeably different. Also, it frequently creates a randomized background that is hazy and indistinct. The main cause of this issue with producing realistic faces is because I downscaled the training photos' quality too much, to the point that there was just insufficient information to do so.

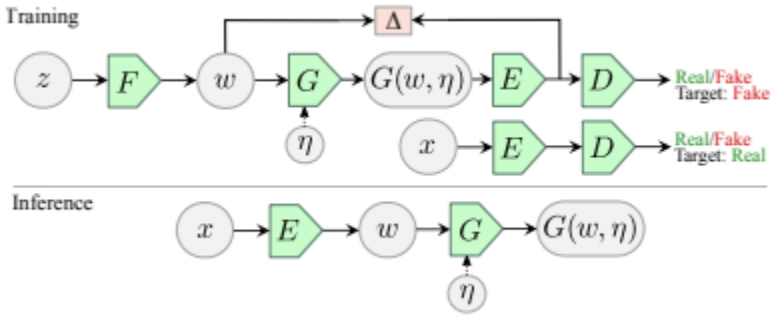
6.3 ALAE - Michael Amorim

This part is based on the paper and was told this is fine by Professor Farhana since the code was not running on my local machine.

This model was built to generate new pictures where the facial features are different from the original. I tweaked the code to run on less images of certain resolutions. Initially it was coded with a batch size at 128x128 resolution and 64x64 resolution when training on 1 GPU, but I changed to stop after it ran at 32x32. This was done so that it would train faster and still train effectively on my local machine. Later I lowered the learning rate to get more accurate results, initially it was 0.002 and then lowered it to 0.0015 as the base learning rate as well as lowering the learning rate for the different resolution files. Was unable to get the results to this though as the code would not run on my local machine.

6.3.1 Data Preprocessing

The dataset that was used for training the model came preprocessed and all of the images were 1024x1024 resolution. The images were then cropped based on the dataset that was used to train the model. Next the images were passed through an autoencoder, but this encoder is also part of the model so the data wasn't really preprocessed through it.



6.3.2 Experimental Setup

Cannot really go into much detail on the system that was used for the results due to the paper not describing what system they used. The code implementation though is multiple python files where there is one you can call and it will train the model and then create an interactive demo. The modules that this paper uses are imageio, numpy, scipy, torch, torchvision, sklearn, matplotlib and some other ones. Imageio is used to prepare some of the images to be used in the model. Numpy, scipy, sklearn and torch are all used for multiple different purposes such as setting up the values to actually create the new facial generations. Matplotlib is then used to plot out the results and accuracy of the model.

6.3.3 Training/Test/Validation

Training Arguments		
Parameter	**Value**	**Reasoning**
Batch Size	Different batch sizes for different resolutions and sizes	The model was trained on multiple different image resolutions where they slowly grew up to 1024x1024. Due to this multiple different batch sizes were used based on what the current resolution of photos were. For my modified code I changed the batch size for images to be (128, 128, 64, 32) where the size was 4x4, 8x8, 16x16, and 32x32 respectively.
Image Size	Different sizes used	As mentioned above different sized images were used in training. Starting at 4x4 then grew to 32x32
Epochs	60	The initial training epochs for this was 100 but I lowered it to 60 as the results were already almost perfect at 100 but the training time was taking too long.
Learning Rate	0.0015	The paper used a learning rate of 0.002 but that learning rate was for 100 epochs so to combat the lower epochs I modified it to 0.0015 to still get good results.
Beta	0.995	The beta parameter is used for the Adam optimizers.
# of colour channels	3	Number of colour channels is set to 3 for RGB to work

The stopping condition for this model was a set amount of epochs. Below you can see a table from the original paper showing the FID score as well as the PPL and how this new model compares to existing models already. The model was trained on a total training time of 10,000,000 if it were to be measured in number of images. This is also less training time than the StyleGAN model where it used 25,000,000 images to train. The training between sizes was done by growing the resolution with 500,000 training samples during the transition and then another 500,000 images were used after the transition as training stabilization. Once the final resolution of 1024x1024 was achieved the model was trained for 18 epochs for a total of 1,000,000 images. This resulted in a model with better results than models that do the same thing as well as a shorter training time in terms of number of images used.

	FID	PPL full
PGGAN [23]	**8.03**	229.2
GLOW [27]	68.93	219.6
PIONEER [16]	39.17	155.2
Balanced PIONEER [17]	25.25	146.2
StyleALAE (ours)	19.21	**33.29**

Table 5: Comparison of FID and PPL scores for CelebA-HQ images at 256×256 (lower is better). FID is based on 50,000 generated samples compared to training samples.

6.4 Deep Privacy - Daniella Ruisendaal

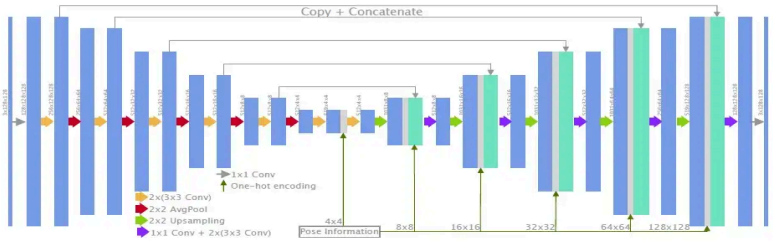
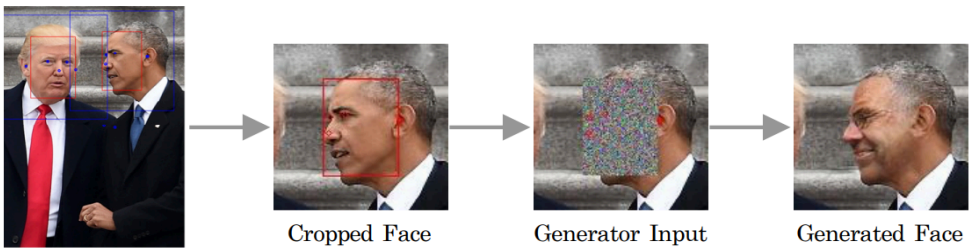


Fig. 3: **Generator Architecture** for 128 × 128 resolution. Each convolutional layer is followed by pixel normalization [12] and LeakyReLU($\alpha = 0.2$). After each upsampling layer, we concatenate the upsampled output with pose information and the corresponding skip connection.

The architecture for this model consists of a generator and a discriminator, as most other architectures do. The discriminator is used to train the generator on what would constitute a realistic vs unrealistic image. Both the generator and discriminator use progressive growing and therefore don't have a set number of layers or blocks. For the generator, each progressive increase in resolution results in 2 3x3 convolutional layers being added to the start of the encoder and the end of the decoder. Each convolutional layer uses a LeakyReLU for activation. In order to include information from the background of the images, a U-Net architecture is used that generates images with a 128x128 resolution. The discriminator architecture includes the image's background information as a *conditional* input at the start of the discriminator, which means there are 6 channels for the input image. The higher resolution layers are treated as residual blocks to mimic the skip-connections present in the generator and the weights for these increase linearly from 0 to 1. DeepPrivacy also uses an equalized learning rate rather than weight initialization. The weights used begin with a trivial initialization and are then dynamically scaled during runtime. This ensures the dynamic range (and therefore the learning speed) is the same across all weights.

6.4.1 Data Preprocessing



The images in the CelebA dataset are already set up with attribute labels and bounding boxes, but the DeepPrivacy model extends the pre-processing to add its own bounding boxes to replace

all the face pixels with a static-looking image instead of facial features. This extra pre-processing is what makes this model different from others in that the generator will never see the original image when creating the anonymized image. Each face in the image is cropped into a quadratic image and the smallest possible bounding box that surrounds the face is added, which then gets resized to the target resolution and the pixels within the bounding box are shifted to the range of -1 to 1, which is what gives the image the static-looking box over each face. This static-looking image is what is given to the generator as input.

6.4.2 Experimental Setup

This model was tested and trained on a Windows computer with an Intel Iris Plus GPU. The training was done on 1,500 images as doing the full 200,000 image dataset caused multiple issues for the computer. The training on the 1,500 images took over 9 hours to complete. The code was implemented in Python through VS Code. There were many software packages that this model used, but some of the most important ones include torch, torchvision, numpy, and cv2. The first two were used to preprocess the data, for the initialization and implementation of the model, and for training and testing. Numpy was used for setting initial values for the batches, anonymized images, and final anonymized faces. Cv2 was used to access the images and save the output to the correct directories with the correct names.

6.4.3 Training/Test/Validation

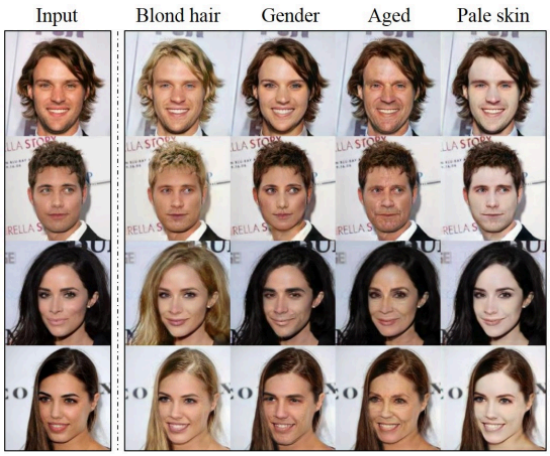
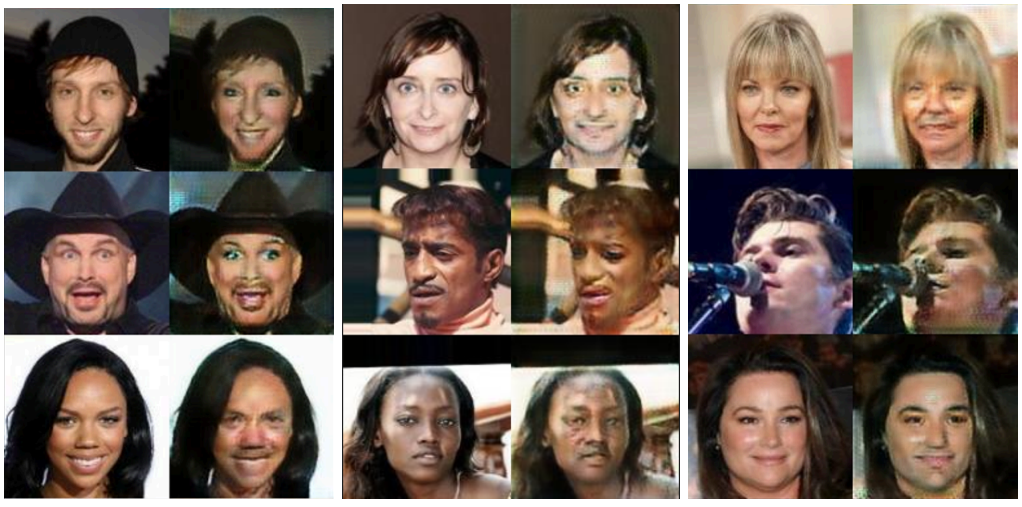
Parameter	Value	Reasoning
Batch Size	Variable	The batch size varies from 256 down to 48, halving itself as the resolution increases from the progressive growing aspect of this model.
Image Size	Variable - 128 original output	The image size (resolution) increases from 8x8 up to 128x128, doubling as the generator works. The final value is 128 to keep the results from being overly pixelated.
Learning Rate	0.00175	An equalized learning rate of 0.00175 with the Adam optimizer a decay of 0.5(batch size/10^4). I adjusted this learning rate to 0.005 to make up for the fewer images given to it
Beta 1, Beta 2	0, 0.99	The beta hyperparameters are used for Adam optimization. This model uses the same beta values as the Karras *et al*. paper does, with values of 0 and 0.99. The two beta values are multiplied together, so having beta 2 be 0.999 will help in preventing the learning from slowing down too much.
Colour channels	6	There is background information included in the discriminator input, so this model ends up with 6 channels instead of just the usual 3.

The paper associated with this model never mentioned any testing information (how the data was split for testing, training, and validating, how they determined accuracy, stop conditions etc.), however it is mentioned that the model was trained with 40 million images, with a transition and stabilization phase of 1.2 million images for each expansion of the network - the expansions being the resolution and batch size increases due to progressive growing. The code for the model

also didn't have any specific training information and due to the use of progressive growing, most of the parameters are set based on mathematical equations and function calls that cannot be altered without also changing a majority of the code. The adjustments I made to the code therefore include altering the maximum image size parameter, along with the learning rate and number of input channels. After I made these changes, I tested the model on approximately 1,500 images from the CelebA dataset and validated them by visually comparing the outputs to not only the input images, but also the original unaltered code outputs. The validation criteria I used were if the output images were identifiable as the people in the original image, whether the outputs would pass as being images of a real person, and how high quality the outputs were (on their own and in comparison to the input quality and original output quality).

7. Results and Discussion

7.1 Results

Model Results	
StarGAN (SOTA)	
StarGAN	

DCGAN	
StyleALAE	
Deep Privacy	

7.2 Discussion

StarGAN: StarGAN was reasonably successful at face anonymization. The fake images are fully anonymized and are not identifiable when compared to the before images. However, the resulting images are fairly low quality, and often don't look like realistic human beings. The model is good at handling occlusions, especially microphones blocking the face. This could be due to the nature of the dataset, as it consists of celebrities and many images are from performances, so there are a lot of pictures with microphones. The unrealistic images are likely due to too many translations being done at once, so the noise introduced by each translation gets amplified by each other. Due

to the model's attempt to change the gender of each image, some of the men look like they have makeup on in the after images and many of the women have grey upper lips, which I believe is the model attempting to put facial hair on them. Compared to the SOTA models, StarGAN produced lower quality images, but did a much better job at anonymization than the original StarGAN model. This is because the purpose of the two are different, as our implementation is focused on making faces unidentifiable while the StarGAN proposed by Choi et al. is focused on maintaining identifying features and only changing selected attributes.

DCGAN: Most of the time, the faces produced look like faces, but they are severely warped. Sometimes a picture was created that had no resemblance to a real face. This is related to the uneven facial traits it produced, such as the person's eyes being drastically different. It also produces somewhat randomized backgrounds that are blurry and incomprehensible. The primary component of this difficulty with constructing correct faces is because I downscaled the training image resolution too far, to the point that there was simply not enough data to generate a realistic face.

ALAE: Most of the time, the generated results from this model looked like real faces. They were not warped and looked realistic when the dataset started with 1024x1024 resolution images. In all cases the results resembled a real face, and most of the time you could not make out the original face due to many facial features being changed. The problem with this method was that complex backgrounds would always become blurry. Even if the background started out blurry, the features of the background would change. For example, blurry spots of light could be removed completely or have their colour changed. Other than the blurriness, the results from the model were very good at generating a new face by changing the facial features of original images.

DeepPrivacy: The results of this model are overall very good. DeepPrivacy successfully removes all privacy-sensitive information from faces – even when it fails to make those faces look realistic, it still manages to make them unrecognizable. The bounding boxes around the faces are as tight as they can be, which limits any odd changes being made to anything around the face and the key points are correctly marked on those images which do not pose significant challenges for the model. The model does have failure cases, though. These cases are the ones for which the person or people in the image are in an irregular pose, images in which there is a lot of occlusion (like if there is an object blocking part of the face), and images in which the background is complex.

8. Conclusion and Future Work

As a summary of each model and their results, StarGAN was good at anonymization but often produced unrealistic faces that would not be acceptable for a database. DCGAN struggled with reproducing facial features due to the small (32x32) image size used, there simply was not enough room to recreate detailed features. ALAE was almost perfect at facial anonymization, but did have some issues when the face was completely sideways. Other than that the new faces generated were anonymized to the point where you could not tell the original face. DeepPrivacy was good at anonymization but worked the best when faces were facing forwards, struggled with extreme poses, and could alter facial expressions. In all, the best results we produced from our modified models were from Molly’s implementation of StarGAN. It produced the highest quality results with the most success in anonymization. The best overall results including SOTA models was StyleALAE since they are high quality and very realistic. None of our implementations were as successful as the SOTA due to limitations in compute power and time needed to train these models.

The main challenge we faced was a lack of compute power. Most of the SOTA models we looked at used multiple powerful GPUs and took weeks to train, which were resources we did not have access to. This could be addressed to improve results by dedicating more time to training, using cloud computing solutions for more processing power, or further simplifying our models which could sacrifice the quality of our results. The other challenge faced was with Michael's model, ALAE, which would not run on his computer, so the information presented for this model is based on the paper that introduced it. We could further improve our results by using a more diverse dataset with more variety in poses, occlusions, and image resolution. This variety would prevent our models from overfitting to perfectly framed images, making them more usable in a real-world setting.

9. References

Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., & Choo, J. (2017). *StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation*. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2018.00916

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014, June 10). *Generative Adversarial Networks*. arXiv.org. Retrieved December 6, 2022, from https://arxiv.org/abs/1406.2661

Hukkelas, H., Mester, R., & Lindseth, F. (2019). *DeepPrivacy: A Generative Adversarial Network for Face Anonymization.* https://arxiv.org/pdf/1909.04538.pdf

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. https://openreview.net/pdf?id=Hk99zCeAb

Maximov, M., Elezi, I., & Leal-Taixé, L. (2020). CIAGAN: Conditional Identity Anonymization Generative Adversarial Networks. https://arxiv.org/abs/2005.09544

Pidhorskyi, S., Adjeroh, D. A., & Doretto, G. (2020). Adversarial latent autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* https://openaccess.thecvf.com/content_CVPR_2020/papers/Pidhorskyi_Adversarial_Latent_Autoencoders_CVPR_2020_paper.pdf

Radford, A., Metz, L., & Chintala, S. (2016). *Unsupervised representation learning with deep convolutional generative Adversarial*. https://arxiv.org/abs/1511.06434

Ren, Z., Jae Lee, Y., & Ryoo, M. (2018). *Learning to Anonymize Faces for Privacy Preserving Action Detection*. https://arxiv.org/pdf/1803.11556.pdf