ANIMATING STILL IMAGES USING GANS AND CNNS

Ammar Vora
Student #1007668677
ammar.vora@mail.utoronto.ca

Brett Yang
Student #1007951101
bretteng.yang@mail.utoronto.ca

Tyler Yan
Student #1008088542
ty.yan@mail.utoronto.ca

Shivansh Singh
Student #1008125832
shivansh.singh@mail.utoronto.ca

ABSTRACT

With the new advancements in deep learning that allow people to create visual art with text prompts or original photos, we have decided to explore the applications of image generation. Our project consists of converting an image into a realistic video in a fully automatic manner. We target photos with birds in a flying or stationary position, where we'll output a video of the bird flying. In this report, we go over our background research, data collection and processing, primary model, baseline model, quantitative and qualitative results, and ethical considerations. An in-depth look at the challenges we faced designing our models and processing our data and the results produced by our model will be discussed later in the report as well. —-Total Pages: 9

Link to GitHub repository: https://github.com/Ammar-V/APS360

1 INTRODUCTION

Our team has decided to create a model that is capable of receiving a still image of an object and predicting the movement of the objects in the photo to produce a short video animation. The project has many applications such as helping individuals relive history or personal memories in a dynamic and engaging manner, empowering artists to bring their photos or paintings to life and adding a new level of creativity to artwork. As a proof-of-concept that validates this model, we scoped our project down to landscape images with birds as the subject of focus to animate. Flying patterns of birds are known to be quite complex based on a study from Rennes University in France [1]. The animation should consist of a combination of random yet fluid movements that encompass the natural motion of a bird. A model that can successfully animate a bird flying in an image shows if such a network is feasible for other similar objects and scenery.

Deep learning is a suitable approach for many reasons. Firstly, it allows for end-to-end learning which is important for this specific project because the program needs to be adaptable and flexible, and converting images to videos can't consistently be done with straightforward rules or heuristics. Secondly, predicting an object's motion and understanding the content and context of a still image requires advanced pattern recognition which can be achieved with deep learning models such as convolutional neural networks (CNNs) [2]. Finally, our project requires generating many realistic images that resemble a real photo and capturing complex patterns and designs which can be accomplished with machine learning models like Generative Adversarial Networks (GANs) [3].

2 ILLUSTRATION

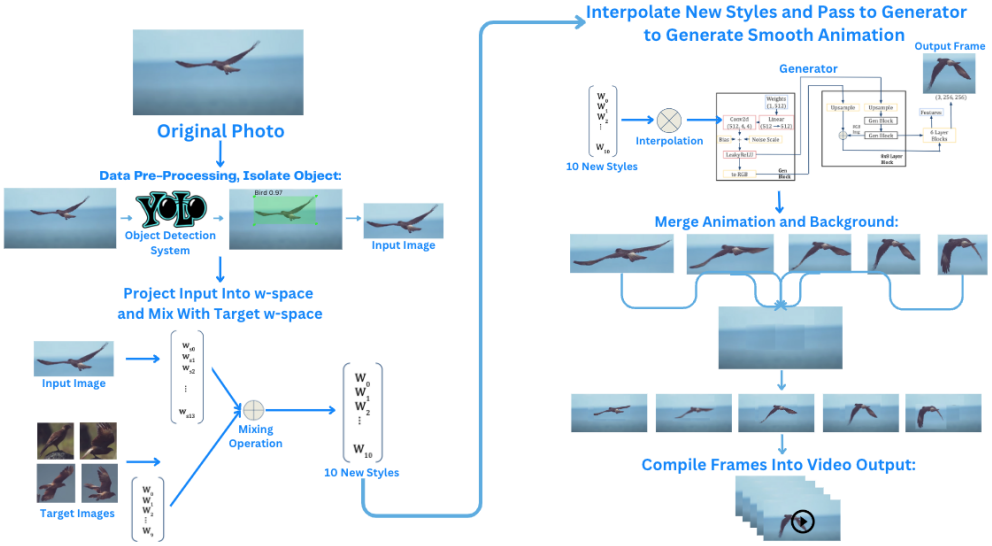

Figure 1: A flowchart illustrating the project's overall design

3 BACKGROUND AND RELATED WORK

The first study, **Animating Pictures with Eulerian Motion Fields**, by the University of Washington students and a Facebook employee, focuses on turning static images into looping animations using natural motion patterns like water, smoke, fire, or clouds [4]. While we're concentrating on bird animations, their work uses an image-to-image translation network to capture motion from online videos of natural scenes. This motion is then used to create realistic animations through a process called deep warping. This technique warps original images to achieve lifelike movement, something we could apply to our project for realistic bird motions.

The second piece of research found was **Controllable Image-to-Video Translation: A Case Study on Facial Expression Generation**, a paper that goes into the study of image-to-video translation with a particular focus on facial expressions [5]. A photo of a face is selected and then they allow the user to select the length of the video and the expression to be shown. The team designed a neural network architecture that is composed of a base and residual encoder, a decoder to generate new frames, and two discriminators for the purpose of generative adversarial training. They employed the Convolution-BatchNorm-ReLu layers in these modules. When training, their team tested the accuracy of their results by comparing generated images to actual video frames, achieving realistic results. In fact, about 50% of their output was labelled as real by Amazon Mechanical Turk workers.

The next piece of research was the paper, **Polymorphic-GAN: Generating Aligned Samples across Multiple Domains with Learned Morph Maps**, done by 5 members of the NVIDIA Research team and looked at a new type of GAN, a Polymorphic GAN which was able to generate images from multiple different domains [6]. The paper proposed that the Polymorphic GAN different objects shared common features between domains. An example of this would be human and animal faces; while the faces shared fundamental geometric differences, there were many features that were still common between the two such as having two eyes, a mouth, and a nose between the eyes and mouth. The same idea of common features can be applied to our project to help our model predict the features of different species of birds.

The fourth paper, **Understanding Object Dynamics For Interactive Image-To-Video Synthesis**, explores the effect of locally manipulating a static scene and its impact on the global articulations of objects [7]. The method involves training a generative model using videos of moving objects without any explicit knowledge of the underlying manipulation. The model learns to predict natural

object dynamics in response to user interaction, specifically focusing on the deformation caused by the local poking of a pixel. The model is not limited to specific object categories and can be transfer learned to represent the dynamics to novel, unseen object instances.

The final paper **Future Video Synthesis With Object Motion Prediction**, focuses on predicting future video frames based on a sequence of continuous video frames from the past. Similar to our proposed method, rather than directly synthesizing images, the method aims to comprehend the complex dynamics of the scene by separating the background scene from the moving objects [8]. It predicts the future appearance of scene components by employing non-rigid deformation for the background and affine transformation for the moving objects. These anticipated appearances are then combined to generate a coherent video of the future. Experimental results on the Cityscapes and KITTI datasets demonstrate that this model surpasses the state-of-the-art in terms of both visual quality and accuracy.

4 Data Processing

For our dataset, we started off by finding videos of birds flying from YouTube. For our model, we needed massive amounts of data (frames) to train since our model is a Generative Adversarial Network (GAN).

We wrote a simple Python script to split the videos into separate frames to use for our model. GANs require high-quality data with minimal noise, so we had to clean this dataset before training by using YOLOv8's object detection model to crop the frames around the birds (Figure 2). This allowed the model to train only on the isolated bird's flying motion and to ignore any background noise and other objects in the frame. Some frames had multiple birds so we had to discard these images. Another issue we faced was low-resolution images which we had to manually remove from the dataset.

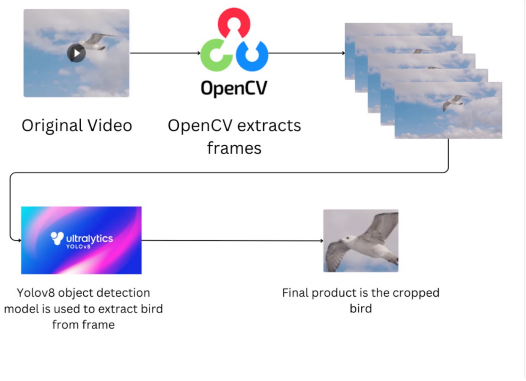

Figure 2: Data collection and processing pipeline

Figure 3: CUB200 dataset images

After we had finished the first round of data preprocessing we had around 10000 frames for our model to train on. Although this seems like a large dataset, our model still wasn't able to train properly. So, we decided to find other sources of data to increase the size of our dataset. We found the CUB200 dataset on Kaggle which was perfect for use case [9]. This dataset had frames of singular birds on the ground and in the air (Figure 3). This dataset consisted of only single birds in the frame so we could freely run YOLOv8 to crop the image around the bird. After compiling and cleaning data from these sources we ended up with around 18000 frames for our model. We decided to use 90 percent of the images for training and 10 percent for testing (validation isn't necessary for GANs so we did not need include a validation dataset).

5 Architecture

For our final model, we took inspiration from the PyTorch implementation of the StyleGAN2-ada model by NVLabs [10; 11]. Due to the size and complexity of their model implementation, we were

unable to train it because of our resources and time constraints. Instead, we used concepts from our baseline model like our training method, size of convolution kernels, activation functions, etc. and made significant architecture/hyperparameters changes to their implementation, while keeping the idea of using blocks as in the StyleGAN2-ada network. By merging together these ideas, we were able to successfully train a complex GAN with the limited resources we had. The final architecture is described below.

The model consists of the mapping network which is used to map a latent vector to an intermediate latent space by normalizing the vector and using two linear fully connected layers and the LeakyReLU activation function.

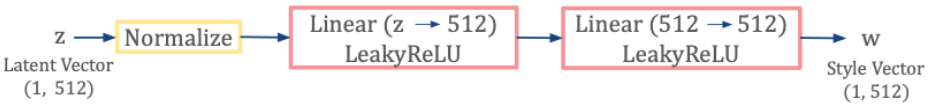

Figure 4: StyleGAN2 mapping network

The Generator starts with a learned constant of features and has an initial Gen Block and a series of seven Layer Blocks consisting of convolutional and linear layers and the LeakyReLU activation function to output a RBG image and feature map. The feature map resolution is doubled at each block and the image is scaled up and summed at each block to output the final 3x256x256 image. The Upsample function scales the image up by two using the bilinear algorithm and smooths each feature channel. The toRGB function uses a 1x1 convolution to change the channel size from any feature size to three.

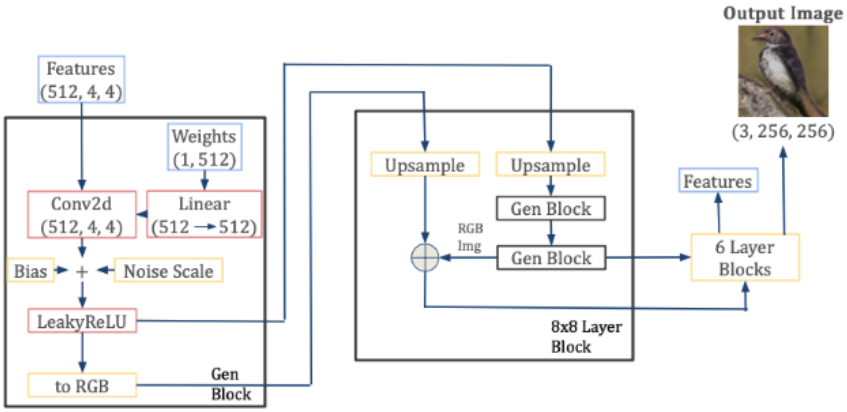

Figure 5: StyleGAN2 generator architecture

The Discriminator first transforms the image to a feature map of the same resolution and then runs it through a series of five Layer Blocks with convolutional layers with residual connections. The resolution is down-sampled by two using bilinear interpolation and smoothed out at each block while doubling the number of features. After the layer blocks, the mean of all minibatch standard deviations for each feature is appended to the feature map as an extra feature. Finally, the data is flattened after a convolutional layer and then goes through two linear layers to produce a prediction on whether the image is real or fake.

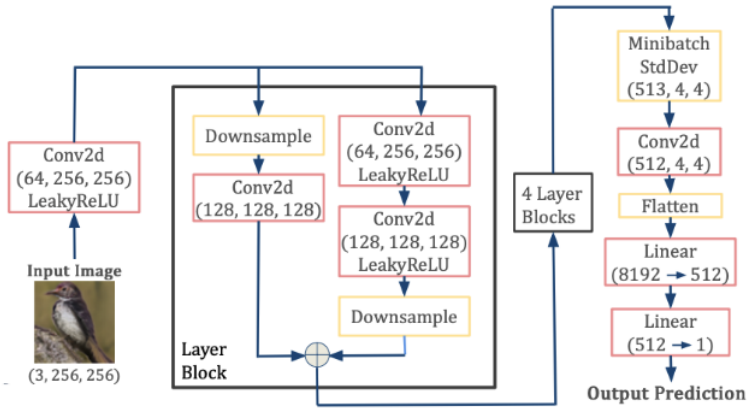

Figure 6: StyleGAN2 discriminator architecture

5.1 Training the Primary Model with Adaptive Data Augmentation

When training the primary model, a method introduced by NVLabs called Adaptive Discriminator Augmentation (ADA) was used [10]. This method was specifically created to address issues while training GANs such as mode collapse and instability, and most importantly, eliminated the need for over 100k training images to successfully train a GAN [10]. While training GANs, the Discriminator often overfits and makes it extremely challenging for the Generator to learn the data set distribution. To address this, a selected set of augmentations are applied to all inputs seen by the Discriminator. The probability of applying each augmentation is controlled by a value, p. To make this adaptive, Discriminator overfitting is measured after each epoch via a metric $r_t = E[sign(D_{train})]$, where D_{train} is the output of the Discriminator on the training set and $E[\cdot]$ is the mean, and p is proportionally changed based on it.

5.2 Animation Pipeline

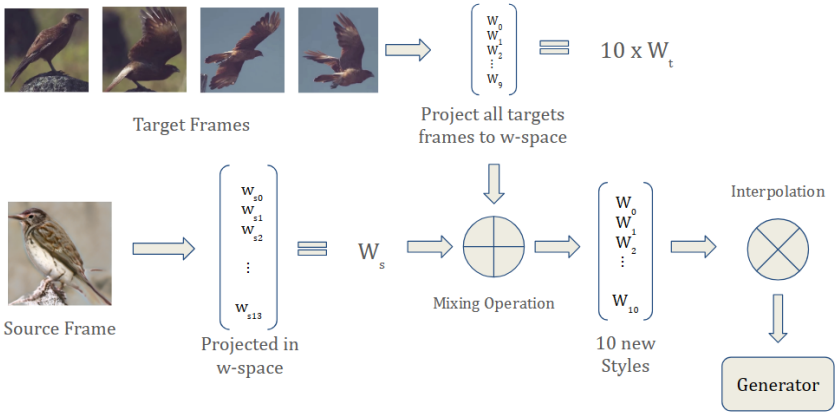

Figure 7: The animation pipeline

The final step in our pipeline is to take an input bird image and generate a video/animation such that the specific input bird looks like it is flying. To do so, the input bird image is translated to different poses using a pre-collected set of 10 target frames of a bird flying. This image translation is done through style mixing and relies on our trained GAN on our processed dataset. Style mixing uses the output of the Mapping Network, also known as style vectors, which are then passed into

the generator to produce an image. Once a GAN is trained, it is able to map different species of birds in different positions in its latent (mapping) space. These style vectors control features in the image such as bird pose, shape, color, and texture. When two different style vectors are selectively combined and passed through the Generator, the output is a bird that contains features from both its inputs. Using this idea, the source style vectors are combined with target style vectors to produce a new image that showcases the input bird in a different pose. This process is performed with all 10 target frames, and after interpolating the vectors in the latent space, we are able to generate over 300 frames that portray a smooth animation of a bird in flight.

6 Baseline Model

The baseline model we decided to use was a Deep Convolutional Generative Adversarial Network or DCGAN for short. We chose this model because our problem requires us to create similar images to an input image, and we believe a properly trained GAN would be able to generate such images. The GAN consists of two main components, a generator and a discriminator. Generator loss is updated by passing generated images with real labels into the discriminator. Discriminator loss is updated by taking the average loss of passing real images with real labels and fake images with fake labels into the discriminator. Both the generator and discriminator use batch normalization and the LeakyReLU function inbetween layers to improve accuracy, to speed up training, and to avoid the dying ReLU problem. As seen below in Figure 8, our generator contains 5 transpose convolutional layers to upsample a vector containing random noise to an image, and our discriminator contains 5 convolutional layers to break down an image to predict whether the image is real or fake.

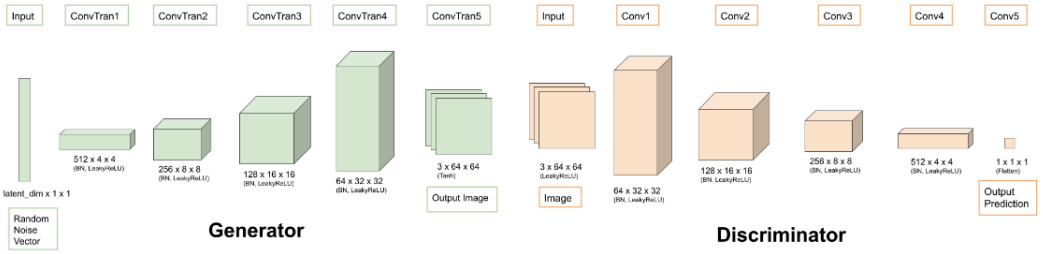

Figure 8: Deep convolutional generative adversarial network architecture

7 Quantitative Results

From research and training, we found that the standard min-max loss function doesn't always properly represent how well a GAN is performing, so instead we used the Fréchet Inception Distance (FID) score [12]. This score is based on the distance between calculated feature vectors for two data sets of real and generated images, and it is used as a metric to quantify the realism of images generated instead of seeing how well the generator fools the discriminator. The lower the FID score, the more realistic the generated image is.

At the beginning of training we started off with a FID score of 149.24, but after tuning our hyperparameters, we were able to bring it down to 23.06. To achieve these results we trained our model for 160 epochs (800 ticks) which took over two days. Due to our team's slower GPUs, we were not able to test as many different hyperparameters and achieve values closer to the state-of-the-art FID results of 2.93.

8 Qualitative Results

To put the FID scores from before into perspective, below are images produced by the generator corresponding to specific FID scores. The images are produced by the generator using style vectors as illustrated in Section 5.

These four results were from four models with different hyperparameters. As seen from the examples below, as the hyperparameters were tuned the FID score lowered and the generator was able to produce images that better resembled birds.

The model works very well on images with simple and single-coloured backgrounds, but when the model tries to reproduce bird photos that have branches, leaves, or other more complex objects and backgrounds the model has more trouble producing clear and coherent photos.

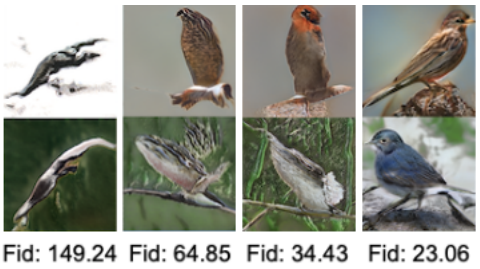

Figure 9: Images generated by models with differing FID scores

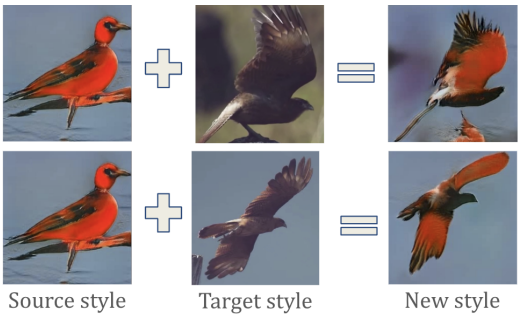

Figure 10: Visual results created from the style mixing operator

In order to create a smooth animation where the bird goes from a sitting to flying motion, we incorporated style mixing, where we swapped the style vectors of 2 different images. This allowed us to swap the major features of the images, which allowed us to mix the sitting and flying features of the two different images. As seen from the results in Figure 10, the style mixing was quite successful. In the two rows, we can see that target styles were able to successfully take features from the source style and create an image that combined both the source and target style features. This demonstrates that the model is able to extract features from birds in a variety of settings and combine them with our team's desired features, bird shape and pose. The style mixing results are also corroborated by the model's low FID score. The model with the best FID score was able to produce the most photorealistic style mixed images.

Figure 11: Randomly generated samples of birds from our network

9 Evaluating Model on New Data

To make sure our model has not overfit to our training data we need to see how our model behaves with new unseen data. To do so, a test set of images are projected into the style space, to see how well our model can represent an unseen image in its latent space. The projected vectors are then passed through the generator, giving us the generated data set. After getting the test and generated data set, a FID score is calculated, showcasing our model's performance on new data. The test set contains new images we've found online and photos we've taken ourselves. The top row in Figure 12 shows examples from our test set that contain images of birds we personally took of in different environments and poses. To fully evaluate our model's performance, the generated images are passed through the animation pipeline to generate videos of the test birds flying.

With the calculated FID score value of 25.88 and the similarity of the generated data to the test data, it is clear our model is able to effectively project new images into the latent space while accounting

Figure 12: Top row: real, bottom row: generated.

Figure 13: Source image passed through the animation pipeline. Four frames have been sampled at spaced intervals from the video.

for the pose and color of the bird. Comparing the test set videos to the results from our training data and to videos of real birds in flight, we can see that our test set performs very well and fulfills our goal of simulating the complex animation of a bird flying from a single picture of a bird (Figure 12 and 13). Due to our test set containing a broad range of environments, poses, and specific bird features (color, texture, size), our results are able to fully portray our models ability to extract all important features from the input while altering the pose of the bird to simulate a flying motion (Figure 13). Video files of our test results can be found in our ***visuals folder*** in our GitHub repository.

10 Discussion

10.1 Does our model perform well?

There are several layers to the discussion of whether our model is performing well. First, let us analyze just our trained neural network. Based on the FID score of 23.06, we can conclude that our model is performing decently well, however, it is easily surpassed by NVLab's state-of-the-art 2.92 FID score. Although, this is not surprising at all considering that GANs are extremely challenging to train on new datasets. Moreover, our models were not trained to their full extent, due to resources and time constraints. Researchers often use multiple high-end GPUs with large amounts of memory while training, whereas we had access to an NVIDIA RTX 3060 Laptop GPU with only 6GB of memory. As such, a model would take us six days to train completely compared to one day for NVLab researchers. They also have the ability to run multiple models in parallel as they have access to multiple computers with state-of-the-art GPUs.

In terms of qualitative results, our model has learned the data distribution quite well. This can be seen in Figure 11 by how detailed some of the aspects of the birds are. The network has clearly learned fine-grain details of a bird such as eyes, beaks, and feet, as well as larger scale distinctions such as the background and the colour difference between the belly and the wings. To further understand these results, we constructed a confusion matrix showcasing cosine similarities between the latent vectors of several generated birds.

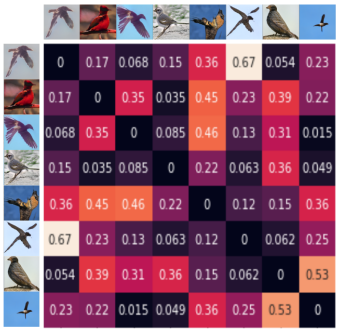

Figure 14: Confusion matrix showcasing cosine similarities between latent vectors. Diagonal has been adjusted to be 0.0.

As you can see, birds in similar poses, species, and environments, have higher cosine similarities than the latter. Since our end goal is to animate a bird in different poses, it is crucial to have a network that can make clear distinctions between birds with different features. For example, a network that is unable to make these distinctions would map a red sitting bird to the same flying bird, as it would for a blue sitting bird. This is because it has not fully separated the different features of a bird in its 512-dimensional latent space. Since our model is able to represent these features really well in its latent/style space, it can be used to produce believable flying animations (Figure 13).

10.2 Lessons Learned and Future Work

For our pipeline, we used the Adaptive Discriminator Augmentation (ADA) method since we did not have the resources to train a vanilla GAN with over 100k images. However, even though ADA allowed us to successfully train a GAN with just 18k images, there are some cases where our model produces images with unusual artifacts such as the last image in Figure 11. This can be explained by further analyzing the data used to train the model, and exploring the underlying effects of using a smaller data set. Although our data set consists of diverse bird species in a multitude of backgrounds, the birds in the images are mostly facing left or right. As such, our model occasionally fails to represent a bird facing the camera, resulting in an unnatural, double headed bird (Figure 11). This showcases a limitation of training GANs with the ADA method: networks are unable to represent the complete 3D model of an object. To address this, future improvements would include using conditional GANs where more information is passed along to the model. This can be in the form of text descriptions, or physical features on the bird's body such as the location of wings, beak, and legs.

11 Ethical Considerations

Our model is primarily designed to animate birds, which inherently carries minimal ethical concerns. However, it is important to acknowledge that if our model is intentionally modified or used for malicious purposes, it can give rise to various ethical considerations.

One ethical concern to take into consideration are deepfakes. Deepfakes can be used to create convincing videos or audio recordings of individuals saying or doing things they never actually did. This ability to fabricate content raises concerns about the spread of false information, as deepfakes can be used to manipulate public opinion, create hoaxes, or damage reputations. This can be especially dangerous in the case of political manipulation as deepfakes can be exploited for political purposes by creating fake videos or speeches of politicians, which can be used to manipulate public sentiment, influence elections, or incite social unrest. So, we need to take these into account while working on these deep learning models.

Another major ethical concern that needs to be taken into consideration is the violation of privacy. Images inputted into the model, whether for training or by a user, may infringe on an individual's privacy if they contain any personal or sensitive information. These images may contain personally identifiable information such as faces and personal financial information. Furthermore, the input of these images into the model may create predictions that could be used for various purposes which further violate an individual's privacy.

REFERENCES

[1] G. Ruaux, S. Lumineau, and E. de Margerie, "The development of flight behaviours in birds," Proceedings. Biological sciences, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7329042/ .

[2] What Is a Convolutional Neural Network? — 3 things you need to know - MATLAB amp; Simulink, https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html .

[3] J. Brownlee, "18 impressive applications of generative adversarial networks (Gans)," MachineLearningMastery.com, https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/ .

[4] Animating Pictures with Eulerian Motion Fields. (n.d.). https://eulerian.cs.washington.edu/

[5] Controllable image-to-video translation: A case study on facial ... (n.d.). https://arxiv.org/pdf/1808.02992.pdf

[6] Generating aligned samples across multiple domains with learned morph maps. Polymorphic-GAN. (n.d.). https://nv-tlabs.github.io/PMGAN/

[7] Abstract. Understanding Object Dynamics for Interactive Image-To-Video Synthesis. (n.d.). https://compvis.github.io/interactive-image2video-synthesis/

[8] IEEE Xplore. (n.d.-b). https://ieeexplore.ieee.org/Xplore/dynhome.jsp

[9] Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). "The Caltech-UCSD Birds-200-2011 Dataset." California Institute of Technology. https://www.vision.caltech.edu/visipedia/CUB-200-2011.html

[10] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila, "Training Generative Adversarial Networks with Limited Data," arXiv preprint arXiv:2006.06676, Jun. 2020.

[11] NVlabs, "StyleGAN2-ADA-PyTorch." [Online]. Available: https://github.com/NVlabs/stylegan2-ada-pytorch.

[12] Machine Learning Mastery. "How to Implement the Frechet Inception Distance (FID) from Scratch." Machine Learning Mastery. https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/. Accessed: August 10, 2023.