Sketch to Animation

GitHub Repo

Bhavya Bellannagari (uni: bb3109, github: bhavyab13), **Malini Kundamal** (uni: mk4329, github: nuclear_pasta02), **Em Sieler** (uni: eas2287, github: emsieler), **& Raaid Elmelegy** (uni: re2473, github: relmelegy)

Table of Contents

Introduction: Project Goals and Relevance
Background
Dataset Explanation
Model Development & Results
Discussion
Limitations
Future Work & Conclusion
References

Introduction: Project Goals and Relevance

The goal of this project was to develop a Sketch to Animation tool that converts sketches into images and then animates those images. This project was split into two sub-projects: Sketch to Image and Image to Animation. Sketch to Image used GANs, while Image to Animation used a diffusion model (Animate-Anything). Our goal was to streamline the animation creation process, providing artists with a quick and efficient tool to bring their sketches to life.

This project seeks to help artists reduce the time between simple sketches and final images or animations. Furthermore, it can increase creativity and productivity for artists across industries. Lastly, it can generate more versions of an image from a sketch. For example, for character sketches, it can lead to more inclusive characters or even personalized characters based on individual preferences.

Background

We began by surveying recent developments in sketch to image generation from the following three papers: Picture that Sketch: Photorealistic Image Generation from Abstract Sketches, SketchInverter: Multi-Class Sketch-Based Image Generation via GAN Inversion, and Sketch to Image Using GANs. We ultimately decided to use the last paper, Sketch to Image Using GANs. The first paper, Picture that Sketch, started with "an abstract, deformed, ordinary sketch from untrained amateurs," differing from the usual edgemap input used in many similar papers. This model also used a StyleGAN trained on images only. SketchInverter used GAN Inversion, and did not rely on sketch-photo pairs, but sought to abstract the process (given any sketch from any category, generate an image). Finally, the paper we based our project on, Sketch to Image Using GANs, used conditional GAN in order to provide additional information in the form of labels to the model.

Dataset:

We used the SketchyCOCO Dataset, which includes 400 total sketches and 400 total images for 4 fruits (apple, strawberry, banana, and pear). Data augmentation techniques like random scaling and flips were performed to increase the dataset to 10,000 images per fruit. We originally intended to use the Sketchy Database, which contained more variation in image categories (animals, vehicles, etc), but as the model struggled with learning complex sketch/image pairs like animals, we decided to switch to fruits for simplicity. We focused on our results for the strawberry and apple datasets for this project.

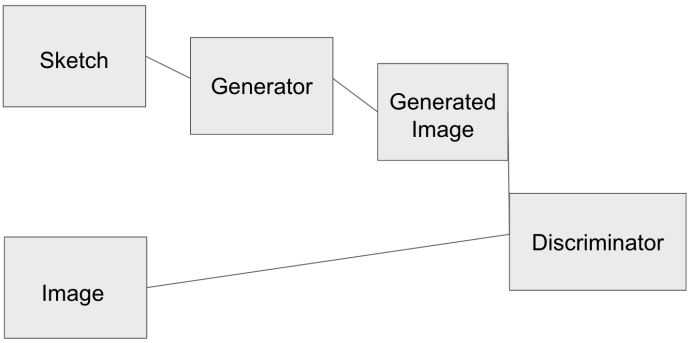
Examples of sketch and corresponding image from Dataset:

strawberry sketch	strawberry image	apple sketch	apple image

Model Development & Results

Image to Sketch GAN Model Diagram:

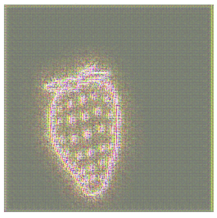
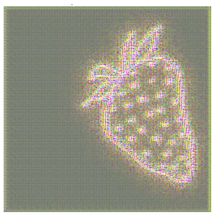
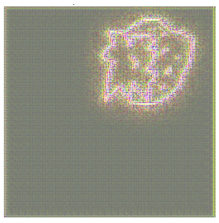


Sketch to Image GAN V1:

Explanation:

For the first version of the GAN we started with a U-Net architecture based generator, which had an encoder portion to process the sketches and extract features from them and a decoder portion to construct the fake images. For the encoder, we used LeakyReLU and reduced the spatial dimensions of the input image while progressively increasing the number of features. For the decoder, we used deconvolutional layers, followed by batch normalization and ReLU activation functions and increased the spatial dimensions of the feature maps.

Results:

Epoch 25	Epoch 50	Epoch 75
		

Evaluation:

	Strawberry

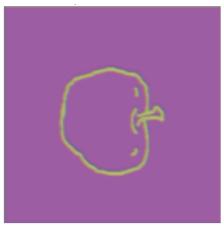
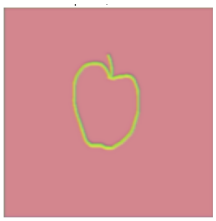
Epoch	Generator Loss	Discriminator Loss
25	72.3893	0.1385
50	72.8592	0.0253
75	79.8464	0.0101

Sketch to Image GAN V2:

Explanation:

As you can see, in Sketch to Image GAN V1, the generator loss (around 70) is quite larger than the discriminator loss (around 0.01), indicating that the GAN is imbalanced, as the discriminator learned which images were fake and which were real too quickly, while the generator could not learn how to produce images that could fool the discriminator. To improve these results, we introduced skip connections to help the model construct more detailed images.

Results:

fruit	Epoch 25	Epoch 50	Epoch 75
strawberry			
apple			

Evaluation:

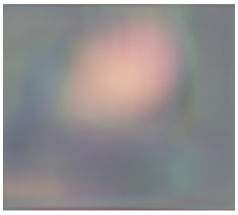
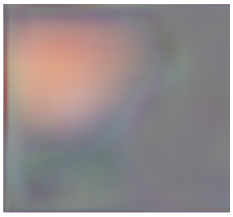
	Strawberry		Apple	
Epoch	**Generator Loss**	**Discriminator Loss**	**Generator Loss**	**Discriminator Loss**
25	69.3103	0.3353	67.1492	0.3952
50	54.6254	0.1658	56.7438	0.1934
75	53.1748	0.1209	54.8266	0.1476

Sketch to Image GAN V3:

Explanation:

As you can see, in Sketch to Image GAN V2, the generator loss (around 55) is still quite larger than the discriminator loss (around 0.01), indicating that the GAN is imbalanced, as the discriminator learned which images were fake and which were real too quickly, while the generator could not learn how to produce images that could fool the discriminator. To improve these results, we changed the architecture of the model by adding a final deconvolutional layer with batch normalization and ReLU.

Results:

fruit	Epoch 25	Epoch 50	Epoch 75
strawberry			
apple			

Evaluation:

	Strawberry		Apple	
Epoch	**Generator Loss**	**Discriminator Loss**	**Generator Loss**	**Discriminator Loss**
25	6.1274	0.1648	6.3589	0.1752
50	5.9743	0.1539	6.1509	0.1687
75	5.8350	0.1357	5.9843	0.1469

Sketch to Image GAN Performance:

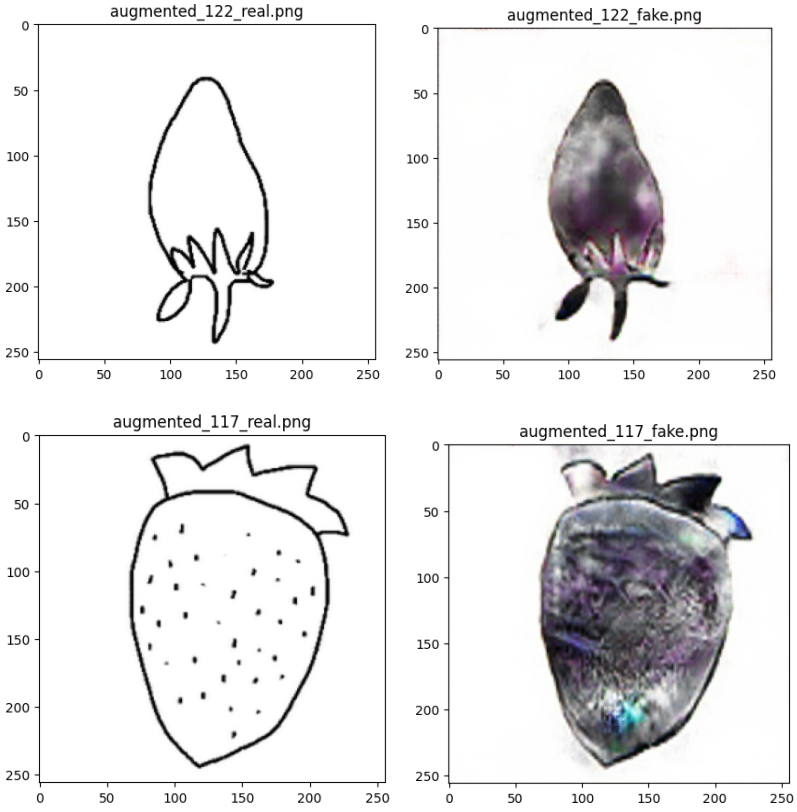
Although the loss improved quite a bit from V2 to V3 of the GAN, the results were not ideal. Even though we used a GPU, it took around 2 hours to only train for 10 epochs. Thus, every time we changed the model architecture, it took several hours to run. We tried improving V3 of the

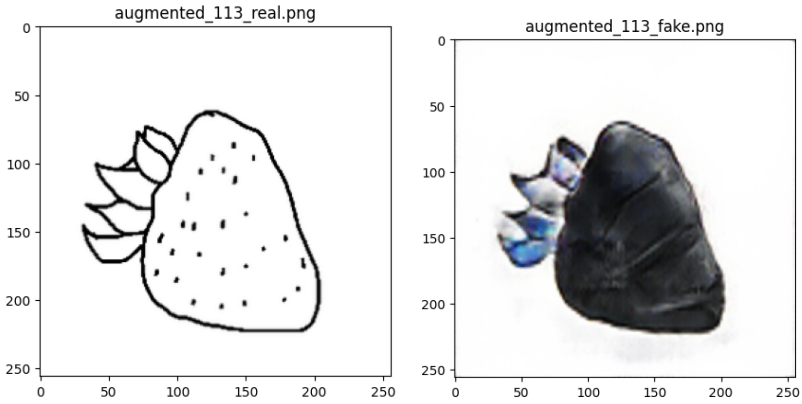
model with a different number of layers and activation functions, but ultimately, the results did not improve much.

SOTA GAN: Pix2Pix

Explanation: We used the official implementation of the paper Image-to-Image Translation with Conditional Adversarial Networks to compare to our model. The pre-trained model edges2shoes was chosen as it seemed to apply most closely to our use case (edges to photos), and the other models provided were unrelated to our application (Labels to Street Scene, Aerial to Maps, etc). The model was deployed out of the box, and below is a sample of its results.

Results:





Evaluation:
If it was trained on our dataset, the model's performance might improve. Since the models provided were trained on maps, shoes, and other non-fruit related images, the qualitative performance was not perfect, although it seemed to outperform our model from a human perspective of realism. Our model did better in learning the color of the images, while the pre-trained model was better at shape and overall translation to photo, although the color and details were lacking.

Image to Video

Two different models were tested to convert the image to video. The first is AnimateDiff, and the second is Animate Anything. The results and discussion for both are provided below.

AnimateDiff

The AnimateDiffPipeline is a part of the diffusers library from Hugging Face. It is designed to create animations from single images based on textual prompts.
This project focused on improving the AnimateDiffPipeline's performance in animating the process of strawberries rotting. Initially, the model struggled with accurately depicting this process due to the lack of training on such prompts. I had to fine-tune the model to enhance its performance, particularly focusing on the Motion Adapter component, using a dataset created from videos of strawberries rotting.

When fine-tuning, the adapter's weights are adjusted using new data, which in this project was the strawberry rotting videos. This process involves training the adapter to recognize changes associated with strawberries rotting, including textural softening and fungus. Once fine-tuned, the Motion Adapter feeds its predictions into the diffusion model. These predictions guide the model in generating each subsequent frame, ensuring that the animation reflects the described transformation accurately.

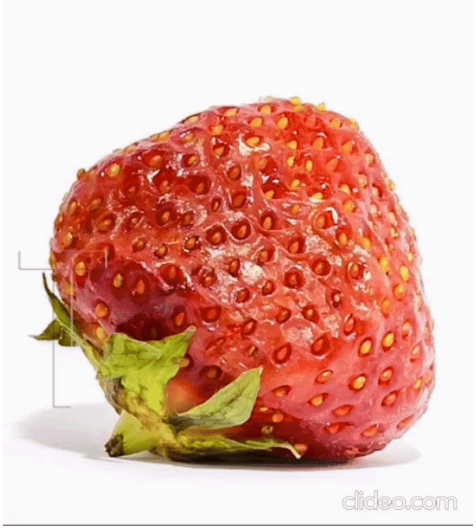
Results from pre-trained Animate Diff Model:
Prompt: Strawberry rotting

Input image:	Result:
	

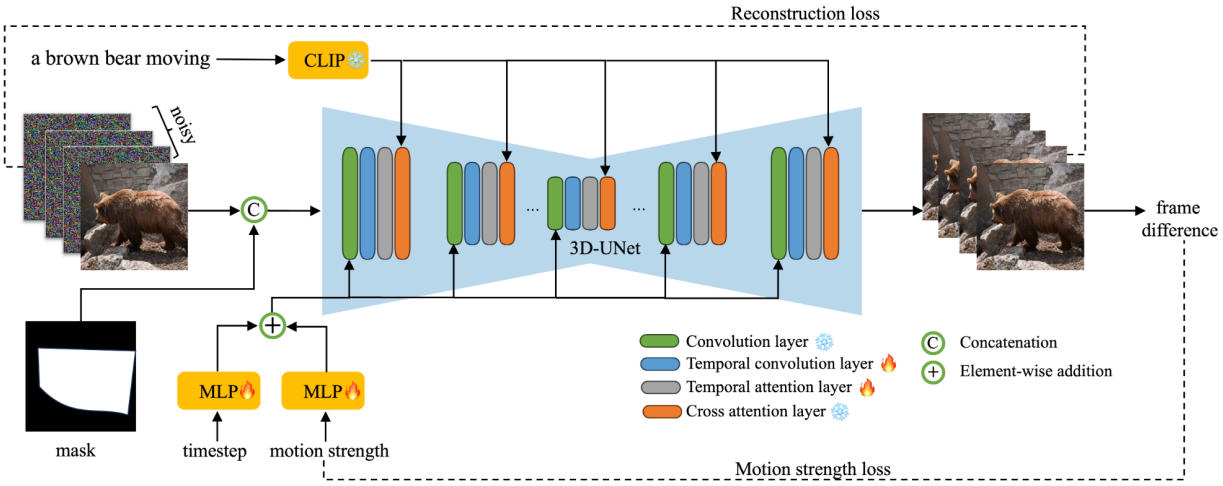
Finetuning AnimateDiff:

To enhance the AnimateDiffPipeline's ability to accurately animate the process of strawberries rotting, I created a specialized dataset tailored for this task. This dataset was constructed from a selection of videos sourced from YouTube, showcasing the gradual decay of strawberries over time. A series of steps were used in order to make the dataset ready to use. Using the ffmpeg tool, each video was converted into individual frames. The frames were resized to 128x128 pixels to ensure uniformity and to reduce computational load during model training. For each video, a corresponding JSON file was created containing annotations for each video. Each extracted frame was then preprocessed to fit the input requirements of the AnimateDiffPipeline. Then, this dataset was used to train the adapter to make better strawberry rotting videos.

Example of training data:



Animate Anything



Model Architecture

The AnimateAnything model aims to generate dynamic visual content from static images. It leverages the motion prior of video diffusion models and introduces two novel guidance mechanisms: motion area masks and motion strength. These features allow for precise control of movable areas and motion speed, enhancing the alignment between animated elements and the prompting text.

Key Contributions

- Motion Area Masks: Guides the animation by specifying which regions in the static image are movable.

- Motion Strength: Controls the speed of motion for each region to achieve fine-grained animation.
- 3D U-Net Architecture: Employs a 3D U-Net-based video diffusion model for high-quality image-to-video generation.

Architecture

- The overall architecture of the AnimateAnything model integrates motion area masks, motion strength guidance, and prompt text into a 3D U-Net video diffusion model to produce high-quality animated sequences from static images.
- Training Process

Reference Image Encoding: The static reference image is encoded into a latent representation using a VAE, serving as the initial frame for the generated video.

- Motion Area Masks and Strength:
- Masks: Specify which regions should move.
- Strength: Defines the speed of movement.

Video Diffusion Model: A 3D U-Net-based model predicts subsequent frames iteratively, starting from the initial encoded frame.

- Noise Prediction Loss: The model is trained with a combination of noise prediction loss and motion strength loss.
- Guidance Composition: Integrates guidance from motion masks, strength, and text prompts to control the animation.

Implementation Details

- Setup: Involves installing Miniconda, cloning the Sketch-to-Video and Animate-Anything repositories, and creating a new conda environment for the animation software.
- Model Preparation: After setting up and activating the environment, all required Python libraries are installed. The Animate-Anything model files are downloaded and organized in a structured directory.
- Training and Inference: Runs the Animate-Anything model with specific configurations. The model uses a static image and a text prompt to generate animations aligned with the descriptions.

Observations and Insights from Model Testing

The AnimateAnything model was tested using a variety of prompts paired with specific images to evaluate its capabilities and limitations in generating dynamic visual content from static images. Below are the images and corresponding prompts that were fed to the model, along with observations from the outcomes:

Animate Anything Results:

- Saturn (Goya): The prompt "Monster biting woman's head off" was used with Goya's painting to test the model's ability to handle complex interactions within classic art scenes.
- Cats (Abstract Painting): An abstract painting featuring cats was paired with the prompt "Four cats wagging tail," which allowed testing of the model's ability to synchronize similar actions across multiple figures within an abstract setting.
- Dali (Salvador Dali): Using a photograph of Salvador Dali, the prompt "Man moves his long mustache" focused on the model's precision in animating subtle, localized movements in a realistic portrait.
- Gizmo (Character from Gremlins): The prompt "Gizmo waving his arm" involved a well-known fictional character, testing the model's effectiveness in translating characteristic gestures into animated movements.
- Gremlin (From Gremlins): For the prompt "Gremlin laughing hysterically," the model was challenged to animate complex facial expressions that convincingly portrayed emotions, specifically laughter.

Full Model Results:

- Dog: Using a sketch of a dog and the prompt: "A barking dog" the pix2pix model produced what looked like a painting of a dog and the AnimateAnything model failed to produce the desired results.

General Observations

- Appearance of Random Objects: Some animations included the appearance of random objects not directly related to the prompts. This suggests limitations in the video model's class recognition abilities, indicating a need for broader class recognition capabilities within the model. Enhancing this aspect would better accommodate a wider range of animation prompts and help minimize the generation of irrelevant visual elements.
- Influence of Image Detail and Complexity: The tests demonstrated that the quality of the model's output is significantly influenced by the level of detail in the image, the nature of the artwork, and the complexity of the requested animation.

These insights highlight the importance of ongoing refinement in the model's ability to handle diverse artistic styles and complex animations effectively. The testing phase has shown that while the model is generally capable of producing high-quality animations from static images, there is substantial room for improvement, particularly in handling diverse and intricate animation scenarios. We learned recently about a database "ImageNet-Sketch" that could possibly help with the image model.

Results:
Prompt: Strawberry rotting

Starting image	Animation 1	Animation 2
		

Discussion and Limitations

One significant limitation we encountered was the quality of the generated images using our non pretrained GAN. One plausible explanation for this result is the insufficient size of our training dataset. While we initially started with a dataset of 100 images, we attempted to augment it to 10,000 images to enhance model performance. However, this extensive augmentation may have introduced noise and inconsistencies, ultimately impacting the quality of the generated images. Furthermore, the diversity and complexity of the sketches within our dataset could have posed additional challenges, as the sketches varied in style and detail, making it difficult for the model to generalize effectively. Another explanation might be that our dataset contained sketches and images, but no corresponding sketch-to-image pairs, which might have made it more difficult for the model to learn the relationship between sketches and images.

Another limitation we encountered was the computational resources available for training. Although we used a GPU, it took around 2 hours to only train for 10 epochs. Thus, every time we changed the model architecture, we faced significant time constraints and delays in experimentation and fine-tuning. This prolonged iteration cycle hampered our ability to explore a wide range of architectural variations and hyperparameters effectively, limiting our capacity to optimize the model's performance efficiently. As a result, the process of refining and improving the model was more time-consuming than anticipated, often impeding our progress.

Another limitation was the GPU out of memory for testing out the fine-tuned model. Because of this, we were unable to get the final results of videos for the strawberry rotting with the finetuned version of the model.

Conclusion and Future Work

In addition to improving our model, for future iterations of the Sketch to Animation projects, we can also add additional features. For example, we could incorporate semantic understanding techniques using NLP to recognize objects, scenes, and gestures within sketches, facilitating more accurate image generation and animation. Furthermore, we can also implement interactive editing features that allow users to refine generated images and animations directly within the tool's interface, enabling greater control and customization. Lastly, we can explore style transfer and integrate style transfer algorithms to enable users to apply different artistic styles to their sketches and animations, expanding creative possibilities for animators and artists.

References

Koley, Subhadeep, et al. "Picture that sketch: Photorealistic image generation from abstract sketches." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023.

Ramy, Ahmed & Barakat, Prof & Barakat, Nahla. (2022). Sketch to Image Using Generative Adversarial Networks (GAN). 10.13140/RG.2.2.16797.38889.

Z. An, J. Yu, R. Liu, C. Wang and Q. Yu, "SketchInverter: Multi-Class Sketch-Based Image Generation via GAN Inversion," 2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).

[1]

Z. Dai *et al.*, “AnimateAnything: Fine-Grained Open Domain Image Animation with Motion Guidance.” arXiv, Dec. 04, 2023. Accessed: Apr. 22, 2024. [Online]. Available: http://arxiv.org/abs/2311.12886

Guo, Y., Yang, C., Rao, A., Liang, Z., Wang, Y., Qiao, Y., Agrawala, M., Lin, D., & Dai, B. (2023). AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning. *ArXiv*. /abs/2307.04725

https://github.com/guoyww/AnimateDiff?tab=readme-ov-file