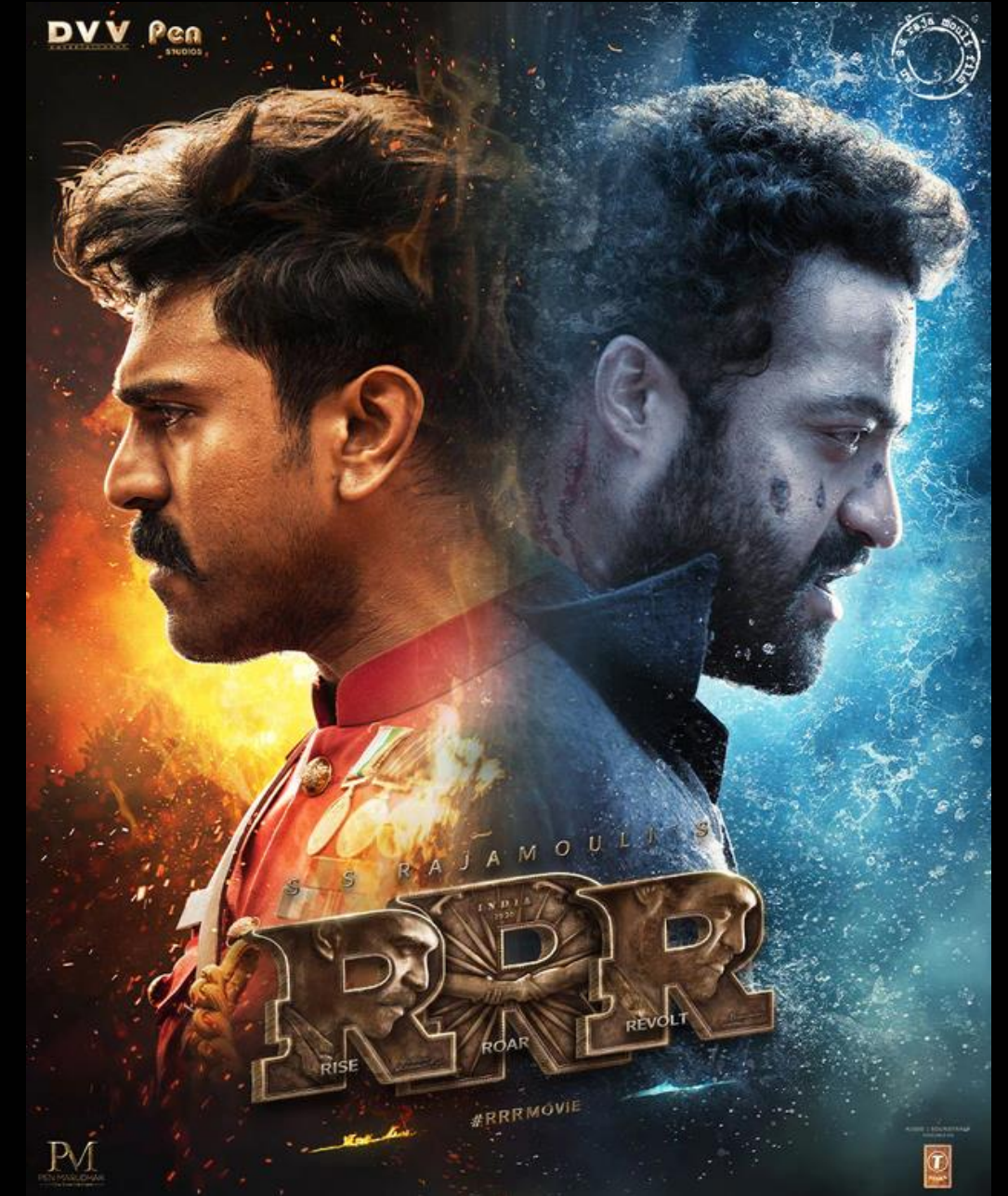
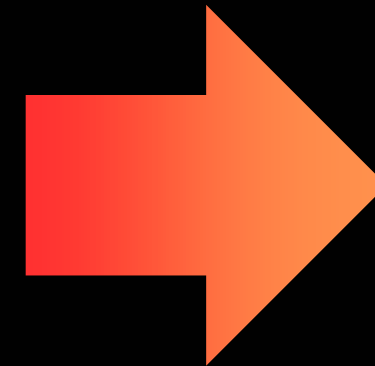
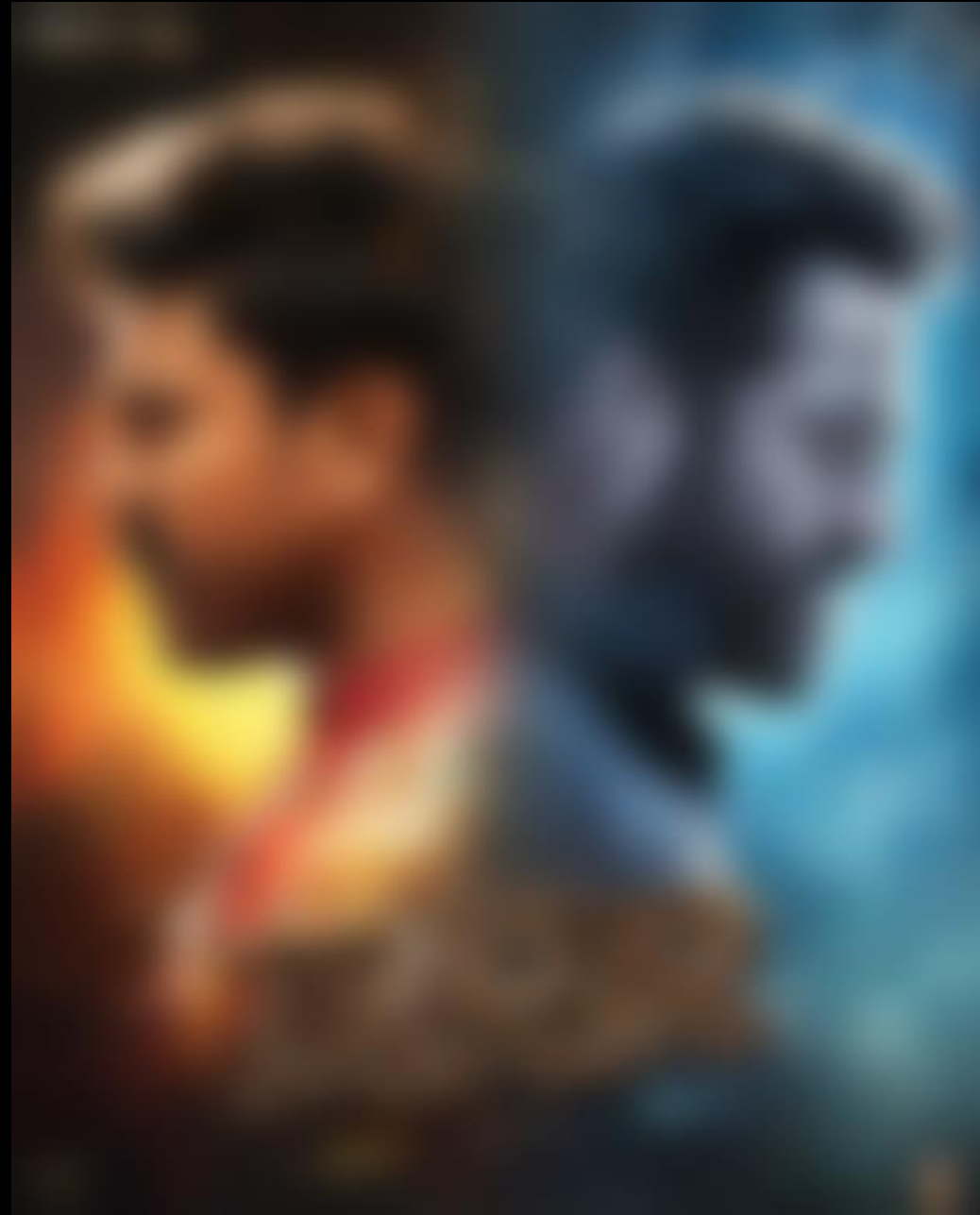


BDA Experience Project

# Image Deblurring Using GAN

## Team RRR

Nikhil Anne  
Srivatsav Busi  
Sahithi Nallani  
Lokesh Paineni  
Hemanth Kumar Enuguri





# Table of Contents

01

Objective

06

AWS S3

02

Real world examples

07

AWS Lambda

03

Architecture

08

ML approach (GAN)

04

Data Source

09

Demo Time

05

ETL

10

Limitations & Learnings



# OBJECTIVE

Enhancing the quality and utility of blurred images from multiple street views and Developing an image deblurring model that is more accurate and realistic than existing models using Generative Adversarial Network (GAN)





# Real World Applications

## Forensic Analysis:

- Enhance images captured from surveillance cameras
- Identify critical details (individuals, vehicles) in criminal investigations
- Useful for Law enforcement, Forensic experts & Judicial System



## Improved Object Recognition:

- Detection of road signs, traffic lights, objects useful for Autonomous Driving
- Identify Infrastructure, road conditions, public facilities for Urban Planning
- Recognize landmarks, provide precise direction and location information useful for Navigation



# Real World Applications

## Property Appraisal & Real Estate:

- Beneficial for Real Estate Marketing, virtual property tours
- Provide accurate property representation for buyers/sellers
- Useful for market research, property inspection & renovation



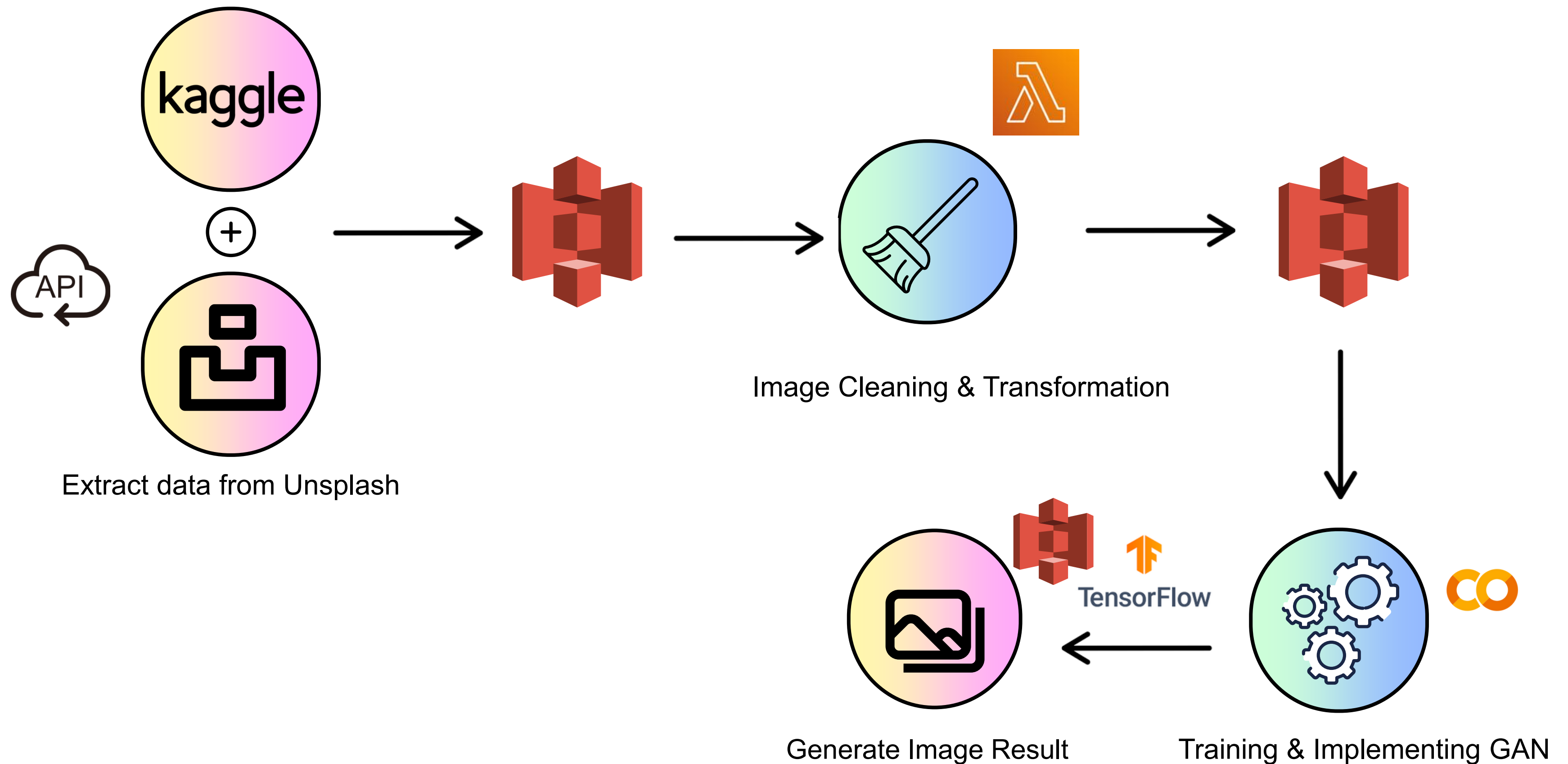
## VR, AR & Gaming:

- Realistic immersive virtual tour experience
- Urban exploration games & training simulations
- Used in AR games to interact with real-world environments





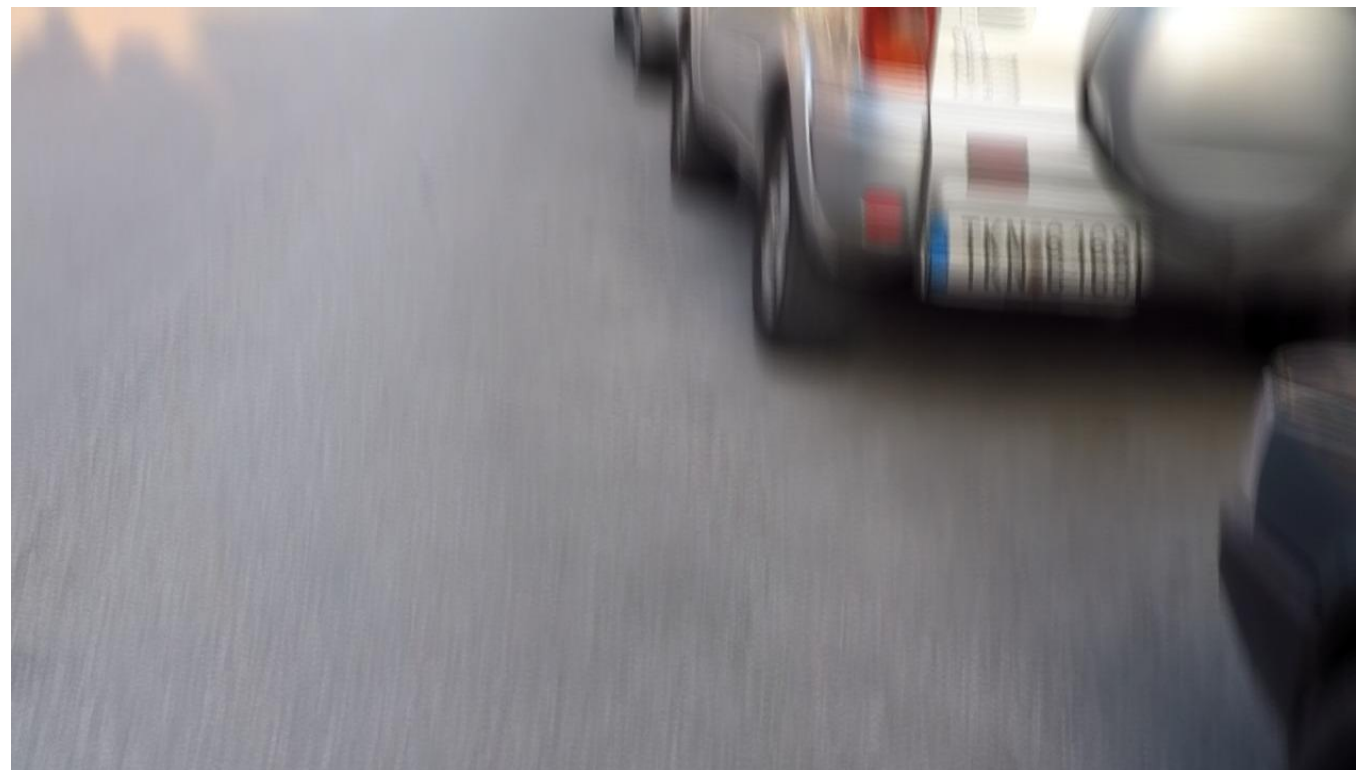
# Architecture



# Data Source - A



- The GoPro dataset is taken from Kaggle which contains blurry and sharp image pairs, collected from GoPro cameras. It was released in 2017 and is widely used for training and evaluating deblurring algorithms.
- The dataset contains 9,214 image pairs.



**Label – A Blur**



**Label – B Sharp**

# Data Source - B



Import required  
Libraries (Boto 3)



Set API keys and  
AWS credentials



Define function  
to get sharp and  
blur image pair



Calls the  
Unsplash API

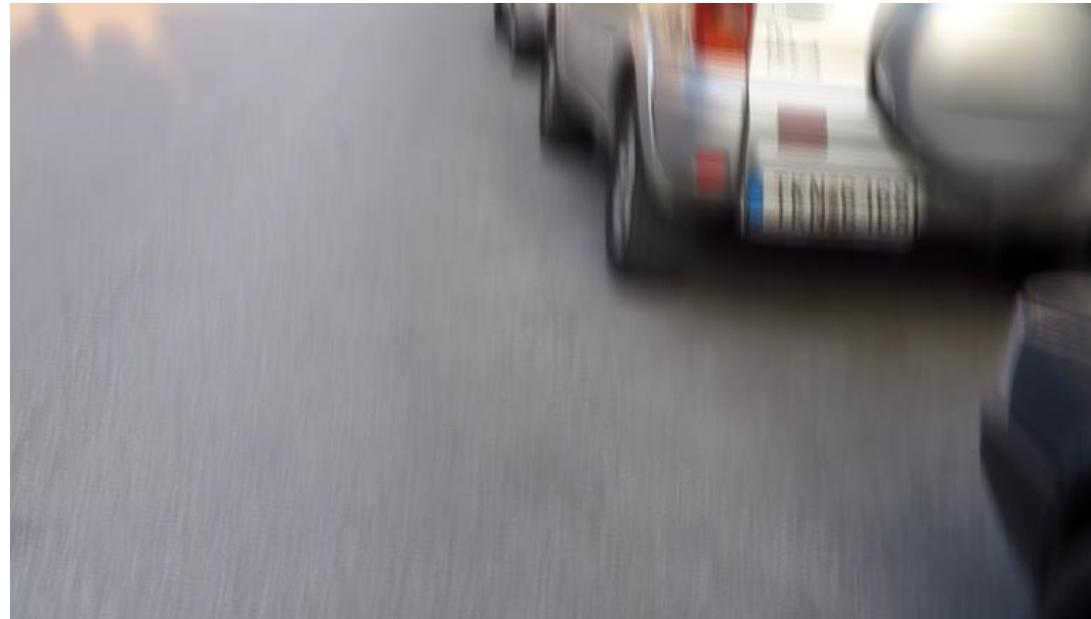


Upload to rrr-  
sourcedump





# SAMPLE IMAGES





# ETL

## EXTRACT

The data is extracted from Unsplash using an API

## TRANSFORM

Source data is cleaned and transformed using AWS Lambda to standardize the image data.(Configure event source to trigger lambda)

## LOAD

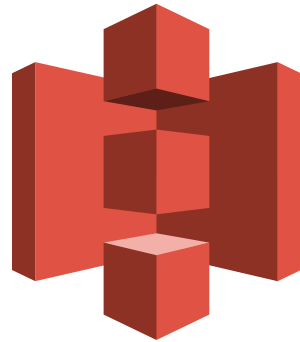
The processed data is loaded to AWS S3 bucket



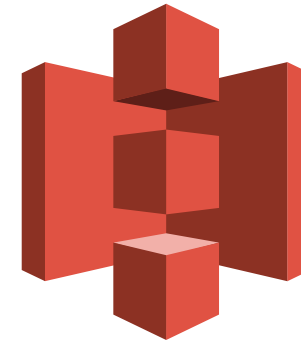


# Source Bucket , Resized Bucket & Output Bucket

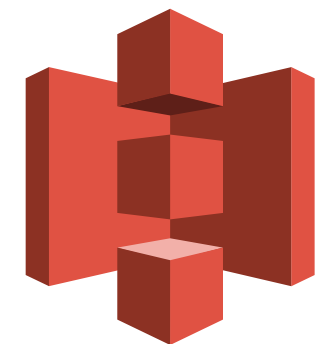
- Once the data is extracted it is stored into **Source dump** bucket.
- After cleaning and transforming the images, we upload into new bucket named **Preprocessed dump**.
- The final outputs received from GAN model is stored into other bucket name **Deblurred Image dump**.



**Source dump**

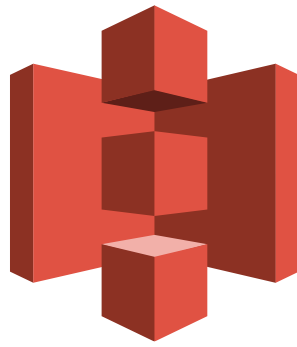


**Preprocessed  
dump**

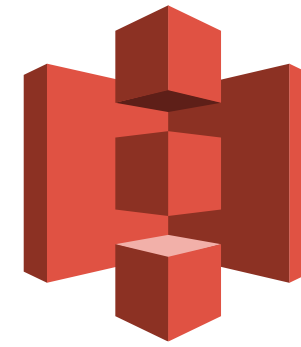


**Deblurred Image dump**





Source dump



Preprocessed dump



Deblurred Image dump

aws

Services

Search

[Alt+S]

Global

HemanthEnuguri

EC2

General purpose buckets (6)

Info

Copy ARN

Empty

Delete

Create bucket

Buckets are containers for data stored in S3. [Learn more](#)

Find buckets by name

< 1 >

|  | Name  | AWS Region                      | Access  | Creation date                           |
|--|---|---------------------------------|---|---|
|  | <a href="#">123577661076</a>                              |                                 |   | 05:00)                                  |
|  | <a href="#">sagemaker-studio-123577661076-pag93ptmjcr</a> | US East (N. Virginia) us-east-1 | <a href="#">Bucket and objects not public</a> | December 2, 2023, 11:59:07 (UTC-05:00)  |
|  | <a href="#">rrr-sourcedump</a>                            | US East (N. Virginia) us-east-1 | <a href="#">Bucket and objects not public</a> | November 29, 2023, 22:49:37 (UTC-05:00) |
|  | <a href="#">rrr-preprocessed</a>                          | US East (N. Virginia) us-east-1 | <a href="#">Bucket and objects not public</a> | December 2, 2023, 19:50:21 (UTC-05:00)  |
|  | <a href="#">rrr-deblurdump</a>                            | US East (N. Virginia) us-east-1 | <a href="#">Bucket and objects not public</a> | December 2, 2023, 19:52:39 (UTC-05:00)  |





Source dump

colab.research.google.com/drive/1J54TUT9UISix1Id0bsxviAUE08eeF12C?authuser=0

COUntitled10.ipynb

File Edit View Insert Runtime Tools Help All changes saved

40s

Stored in directory: /root/.cache/pip/wheels/27/5f/ba/e972a56dcbf5de9f2b7d2b2a710113970bd173c4dcd3d2c902

Successfully built ratelimit

Installing collected packages: ratelimit

Successfully installed ratelimit-2.2.1

Uploaded ubNqlG\_0daY.jpg to S3

Uploaded ruF9kGr5CPY.jpg to S3

Uploaded 4IjVzYsgg8I.jpg to S3

Uploaded tBqpSgKs85A.jpg to S3

Uploaded 5ZA3g\_lwGho.jpg to S3

Uploaded mFYm4SZjmCY.jpg to S3

Uploaded DHM1x00svg8.jpg to S3

Uploaded uFomxGheuGk.jpg to S3

Uploaded flsXgoPoIuY.jpg to S3

Uploaded WZBIiHa8B4w.jpg to S3

Uploaded FZYMMSeXMTI.jpg to S3

Uploaded bfMStQxPaqI.jpg to S3

Uploaded 8-dEiGzaicw.jpg to S3

Uploaded KYuANAFgTWA.jpg to S3

Uploaded cnOMHANKNX8.jpg to S3

Uploaded BqmUACmOytM.jpg to S3

Uploaded CwD51F20IAQ.jpg to S3

Uploaded fFA3mWH3A2M.jpg to S3

Uploaded UCd0wbsXgCE.jpg to S3

Uploaded ZXbeOqF1NFQ.jpg to S3

Uploaded WUBX3nUoJAY.jpg to S3

Uploaded pzTkp714StM.jpg to S3

Unloaded F68UvsivkEs.png to S3

RAM  
Disk

17°C  
Cloudy

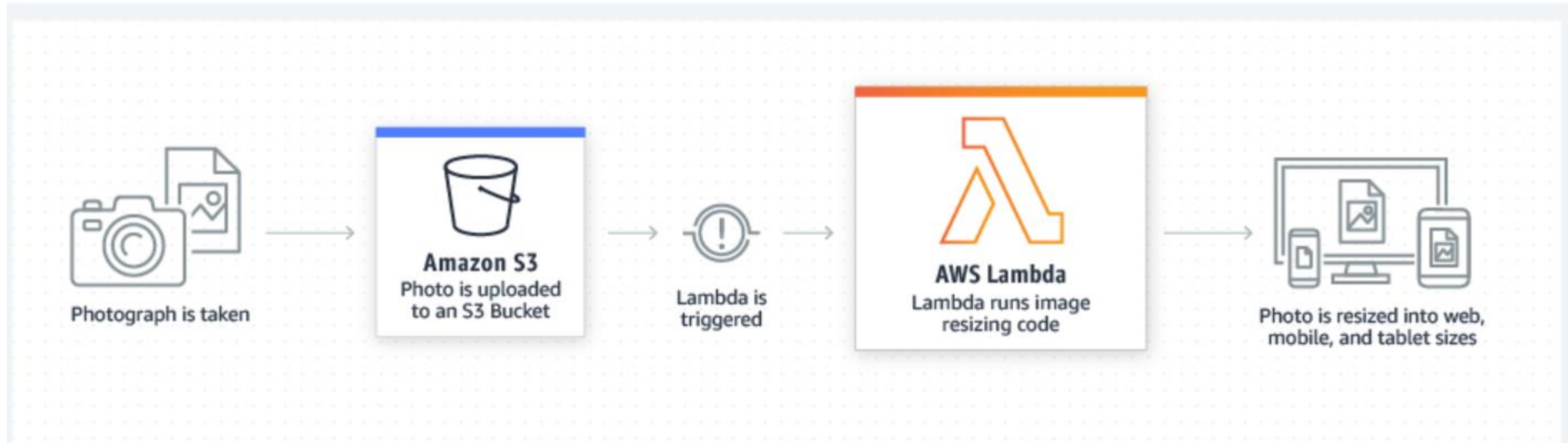
Search

ENG  
IN

11:13 PM  
12/2/2023



# AWS Lambda



# AWS Lambda Implementation



Event trigger action

EC2

☰

✔ The trigger rrr-sourcedump was successfully added to function rrr-preprocessing. The function is now receiving events from the trigger.

✕

▼ Function overview [Info](#)

Export to Application Composer

Download function

Diagram

Template

rrr-preprocessing

Layers (1)

S3

+ Add trigger

+ Add destination

Description

-

Last modified

1 minute ago

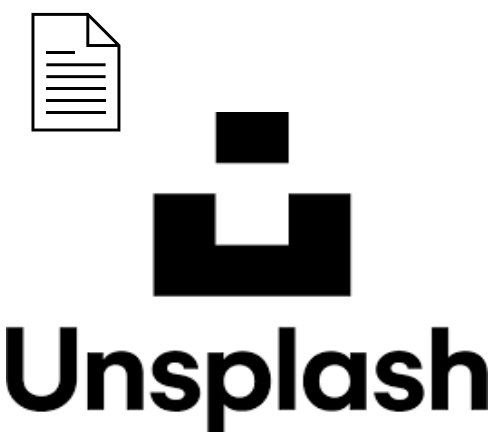
Function ARN

arn:aws:lambda:us-east-1:123577661076:func  
tion:rrr-preprocessing

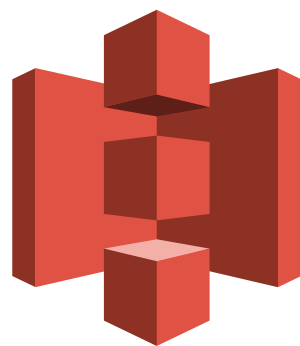
Function URL [Info](#)

-

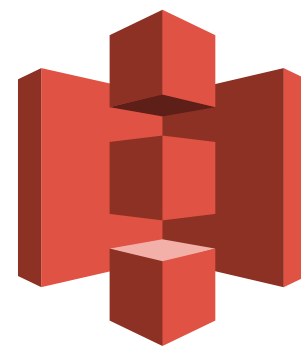




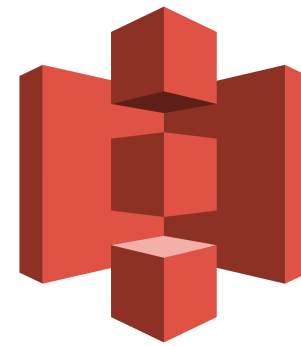
Source dump



Preprocessed dump



Deblurred Image dump



# AWS Lambda Implementation



Missing images lambda  
function

The screenshot shows the AWS Lambda console interface. At the top, there are tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is selected. Below the tabs, there's a 'Code source' section with an 'Info' link. A toolbar contains 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', a 'Test' button, and a 'Deploy' button. A search bar labeled 'Go to Anything (% P)' is present. On the left, an 'Environment' sidebar shows a folder 'CompareImageNames' and a file 'index.py'. The main area displays the Python code for 'index.py'. The code imports 'json' and 'boto3', defines a 'lambda\_handler' function, and uses 'boto3.client' to interact with an S3 bucket named 'my-bucket-547487'. It lists objects in 'Blur' and 'Sharp' folders, extracts image file names, and compares them to find missing images. The code is as follows:

```
1
2 import json
3 import boto3
4
5 def lambda_handler(event, context):
6     s3 = boto3.client('s3')
7     bucket_name = 'my-bucket-547487'
8     missing_images = []
9
10    folders = ['Test', 'Train']
11    for folder in folders:
12        try:
13            # List images in 'Blur' and 'Sharp' folders
14            blur_response = s3.list_objects_v2(Bucket=bucket_name, Prefix=f"{folder}/Blur/")
15            sharp_response = s3.list_objects_v2(Bucket=bucket_name, Prefix=f"{folder}/Sharp/")
16
17            # Extract image file names
18            blur_images = set(obj['Key'].split('/')[-1] for obj in blur_response.get('Contents', []))
19            sharp_images = set(obj['Key'].split('/')[-1] for obj in sharp_response.get('Contents', []))
20
21            # Compare image file names between 'Blur' and 'Sharp' folders
22            missing_in_blur = sharp_images - blur_images
23            missing_in_sharp = blur_images - sharp_images
24
25            if missing_in_blur:
26                missing_images.extend([f"{folder}/Blur/{img}" for img in missing_in_blur])
```



# AWS Lambda Implementation



## Image Resizing

The screenshot shows the AWS Lambda console interface for a function named 'ResolutionStandardization'. At the top, there's a header with the function name and a 'Layers' section showing '(0)' layers. Below this are buttons for '+ Add trigger' and '+ Add destination'. A sidebar on the right shows 'Last modified: 6 hours ago' and 'Function ARN: arn:aws:lambda:us-east-1:123456789012:function:ResolutionStandardization'. The main content area has tabs for 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code' tab is active, displaying a code editor with the following Python code:

```
1 import zipfile
2 import io
3
4 # Function code for image resolution standardization
5 lambda_code = '''
6 import json
7 import cv2
8 import numpy as np
9 import boto3
10
11 def process_image(path):
12     img = cv2.imread(path)
13     img = np.asarray(img, dtype="float32")
14     img = cv2.resize(img, (128, 128))
15     img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
16     img = img / 255.0
17     img = np.reshape(img, (128, 128, 3))
18     return img
19
20 def lambda_handler(event, context):
21     s3 = boto3.client('s3')
22     bucket_name = 'my-bucket-547487'
23
24     try:
25         # Retrieve the image file from S3
26         image_obj = s3.get_object(Bucket=bucket_name, Key=event['key'])
27         image_data = image_obj['Body'].read()
28         with io.BytesIO(image_data) as f:
29             with zipfile.ZipFile(f, 'w') as zipf:
30                 zipf.writestr('image.jpg', image_data)
31         zip_data = zipf.getvalue()
32         response = {'statusCode': 200, 'body': zip_data}
```

The code editor has a menu bar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', and 'Window'. There are also 'Test' and 'Deploy' buttons. The left sidebar shows the 'Environment' tab with a file tree containing 'ResolutionStandardization' and 'lambda\_function.py'. The bottom of the console has a 'CloudShell' button and a 'Feedback' link.

# Before Resizing



| General                |                                | Security | Details | Previous Versions |
|------------------------|--------------------------------|----------|---------|-------------------|
| Property               |                                | Value    |         |                   |
| <a href="#">Origin</a> |                                |          |         |                   |
| Date taken             |                                |          |         |                   |
| <a href="#">Image</a>  |                                |          |         |                   |
| Dimensions             | 1280 x 720                     |          |         |                   |
| Width                  | 1280 pixels                    |          |         |                   |
| Height                 | 720 pixels                     |          |         |                   |
| Bit depth              | 24                             |          |         |                   |
| <a href="#">File</a>   |                                |          |         |                   |
| Name                   | MicrosoftTeams-image (2).png   |          |         |                   |
| Item type              | PNG File                       |          |         |                   |
| File location          | C:\Users\Nikhil Anne\Downloads |          |         |                   |
| Date created           | 03-12-2023 00:44               |          |         |                   |
| Date modified          | 03-12-2023 00:44               |          |         |                   |
| Size                   | 1.15 MB                        |          |         |                   |

# After Resizing



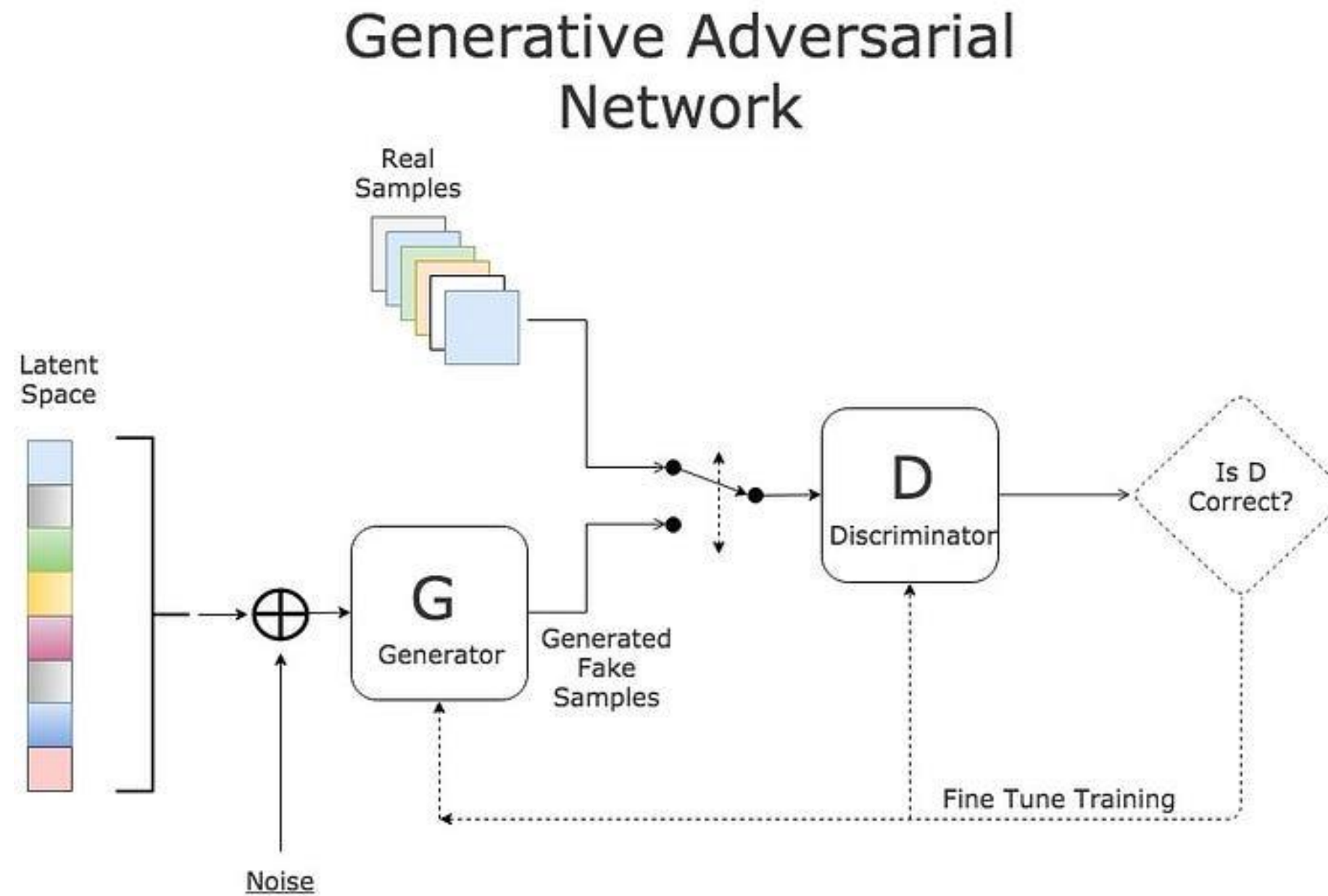
GeneralSecurityDetailsPrevious Versions

| Property               | Value                          |
|------------------------|--------------------------------|
| <a href="#">Origin</a> |                                |
| Date taken             |                                |
| <a href="#">Image</a>  |                                |
| Dimensions             | 128 x 128                      |
| Width                  | 128 pixels                     |
| Height                 | 128 pixels                     |
| Bit depth              | 24                             |
| <a href="#">File</a>   |                                |
| Name                   | MicrosoftTeams-image.png       |
| Item type              | PNG File                       |
| File location          | C:\Users\Nikhil Anne\Downloads |
| Date created           | 02-12-2023 21:21               |
| Date modified          | 02-12-2023 21:21               |
| Size                   | 40.8 KB                        |



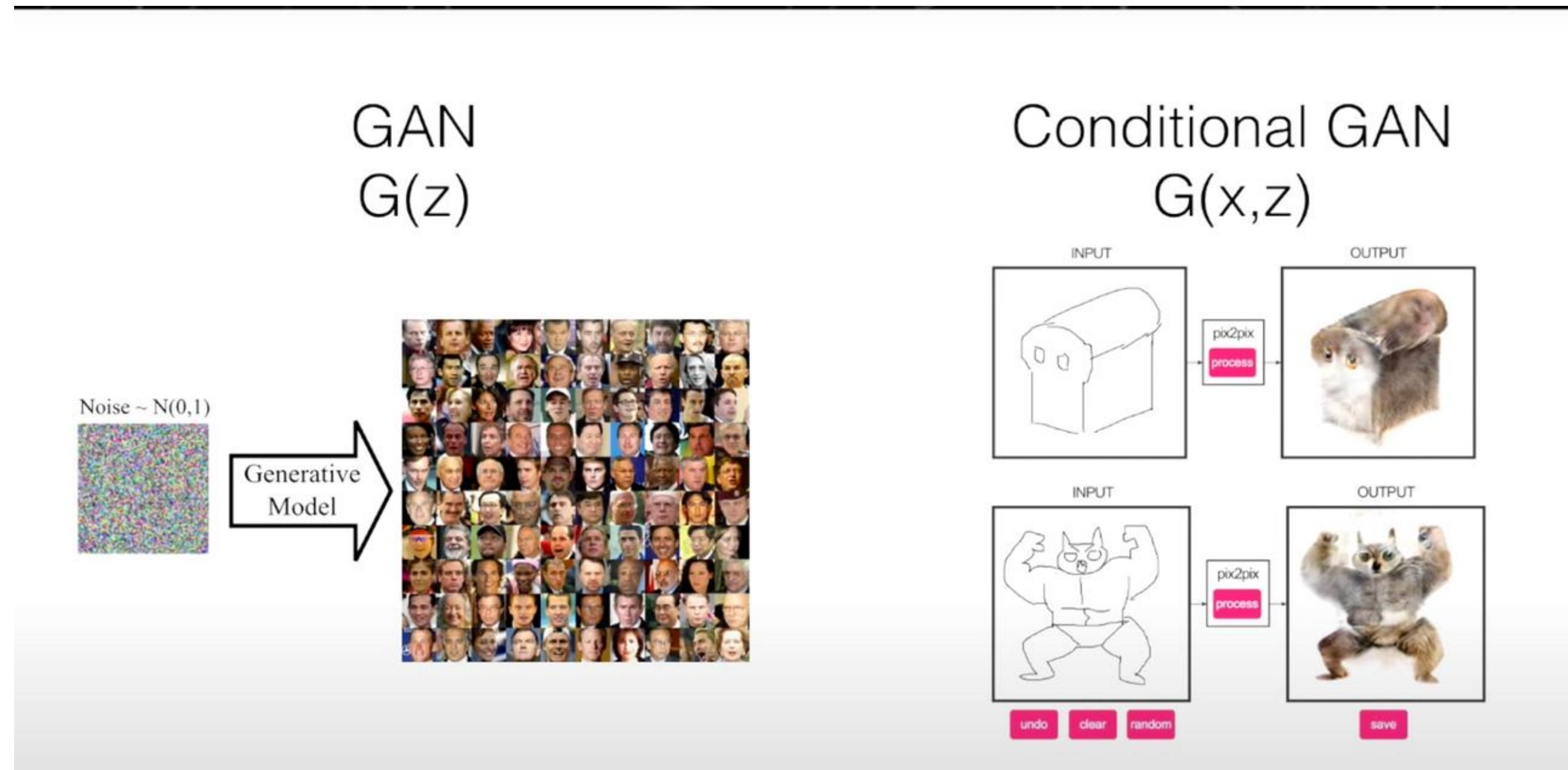
# GAN

- GANs work by pitting two neural networks against each other in a zero-sum game
- The first neural network, called the generator, is responsible for creating new data
- The second neural network, called the discriminator, is responsible for distinguishing between real and fake data



# Conditional GAN

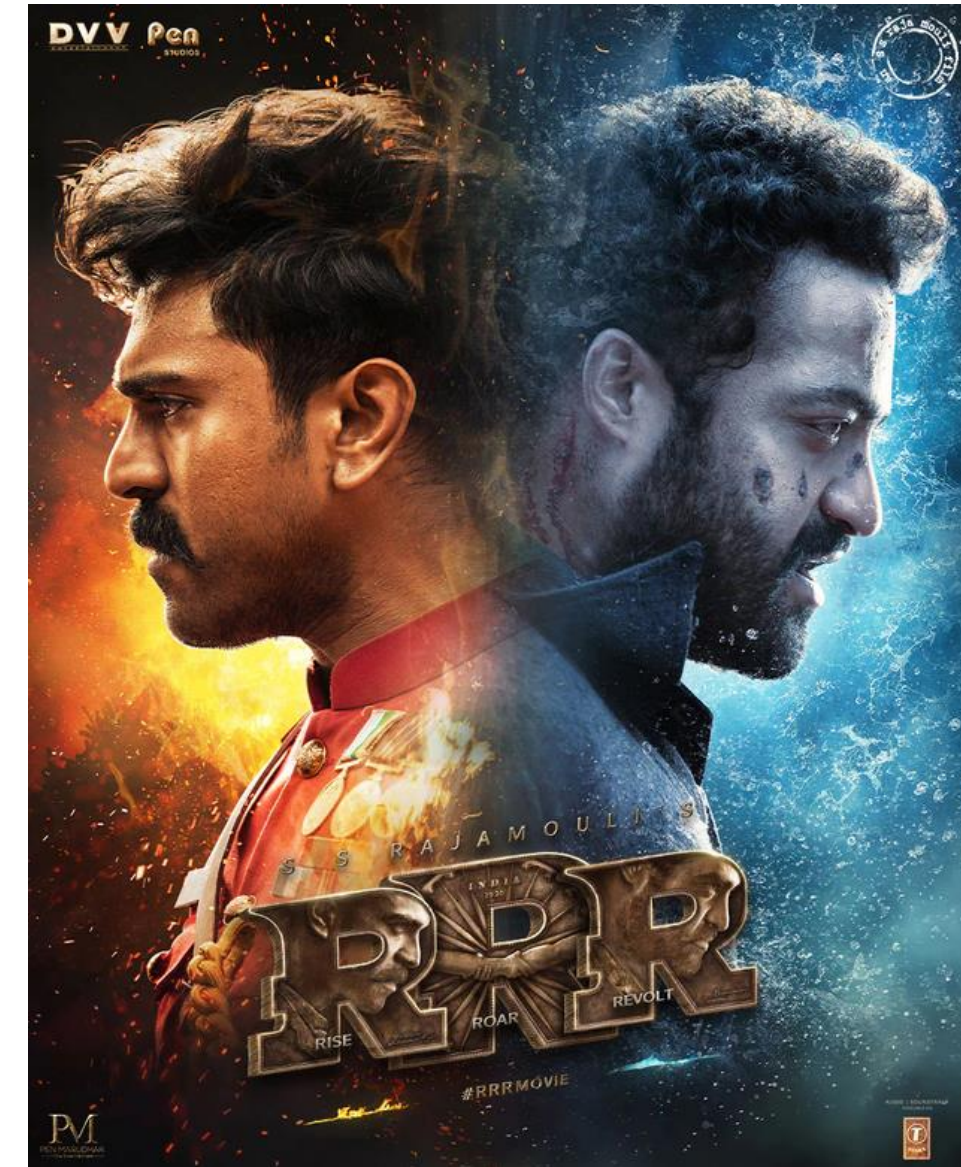
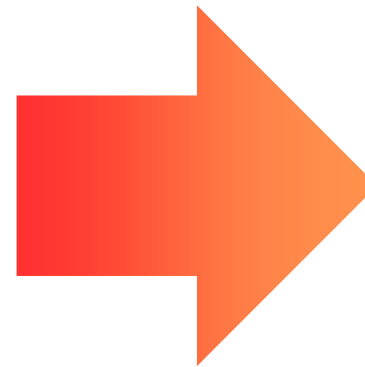
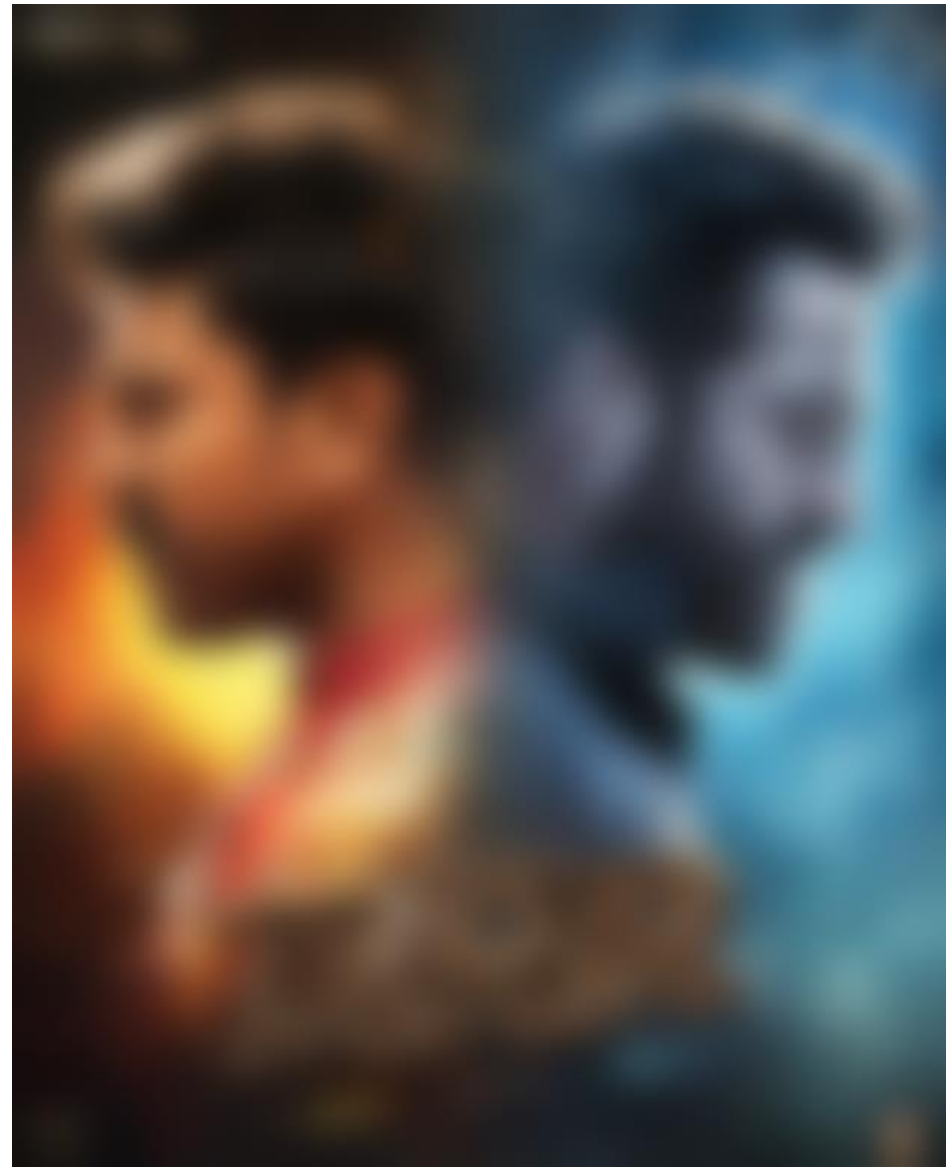
- Generate new data conditioned on some input
- This input can be anything from a text description to an image

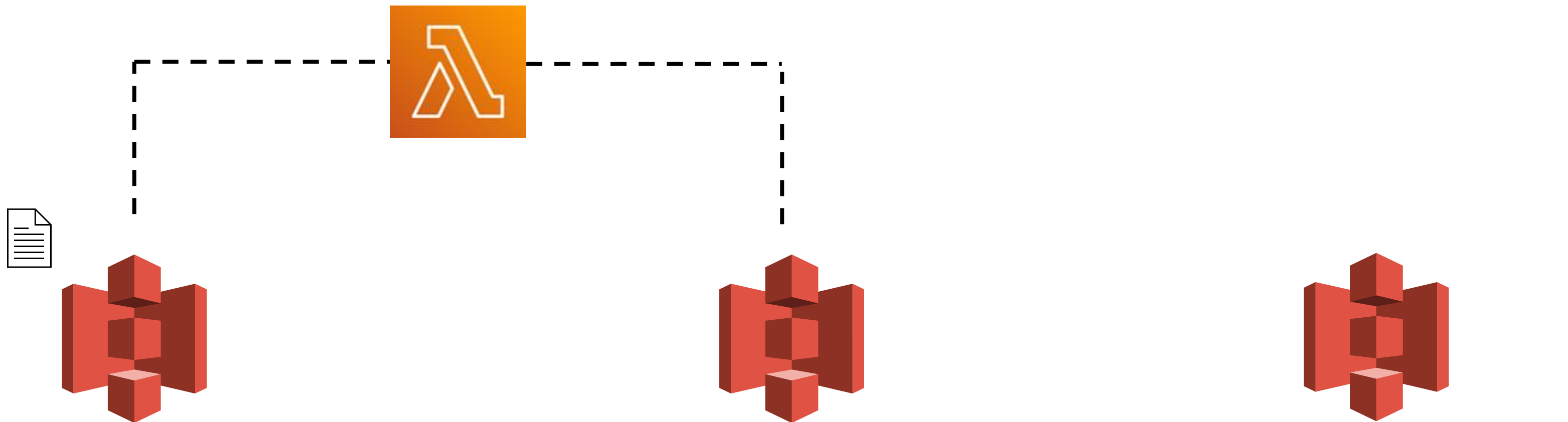




# DeblurGAN

- DeblurGAN is a cGAN-based method for image deblurring
- It takes a blurred image as input and generates a sharp image as output
- DeblurGAN is trained on a dataset of paired blurred and sharp images
- The generator learns to generate sharp images that are consistent with the blurred input images
- The discriminator learns to distinguish between real sharp images and generated sharp images
- DeblurGAN treats deblurring as an image-to-image translation problem

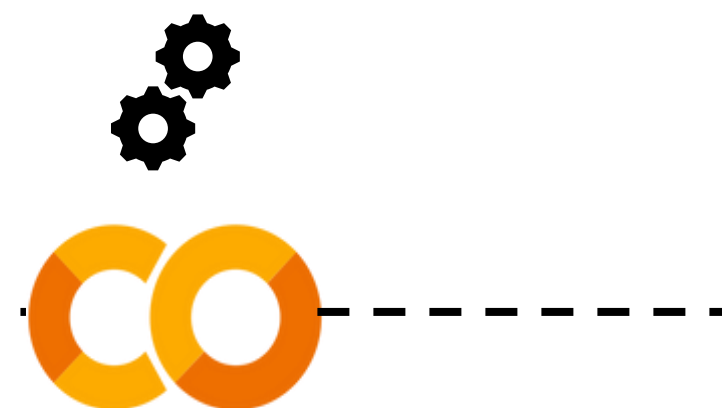
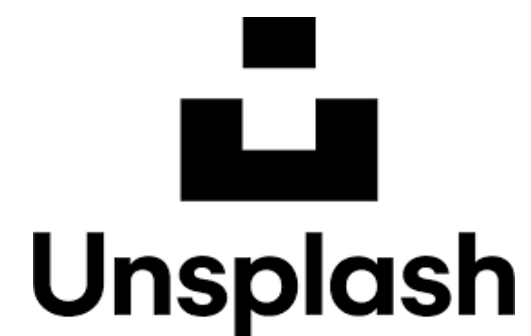




Source dump

Preprocessed  
dump

Deblurred Image dump





# Train Data Sample Images

```
In [*]: fit(train_dataset, steps=150000)
```

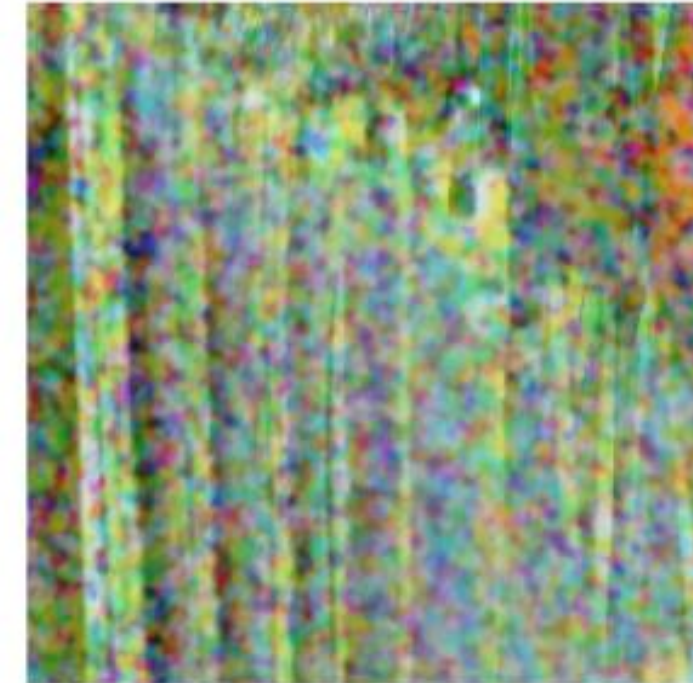
Input Image



Ground Truth



Predicted Image



Step: 0k

.....

**Testing**

.....



```
In [*]: fit(train_dataset, steps=150000)
```

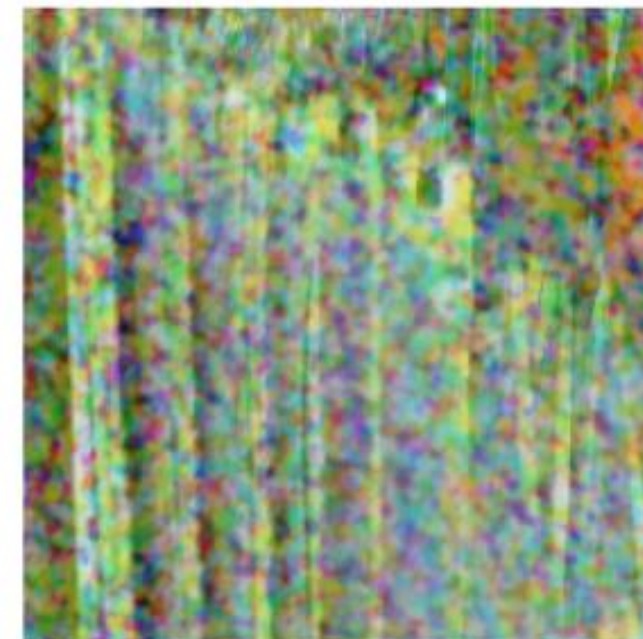
Input Image



Ground Truth



Predicted Image



Step: 0k  
.....

```
In [*]: fit(train_dataset, steps=150000)
```

Time taken for 1000 steps: 2880.86 sec

Input Image



Ground Truth



Predicted Image



Step: 15k  
.....

**Testing**





```
In [*]: fit(train_dataset, steps=150000)
```

Time taken for 1000 steps: 2880.86 sec



Step: 15k  
.....

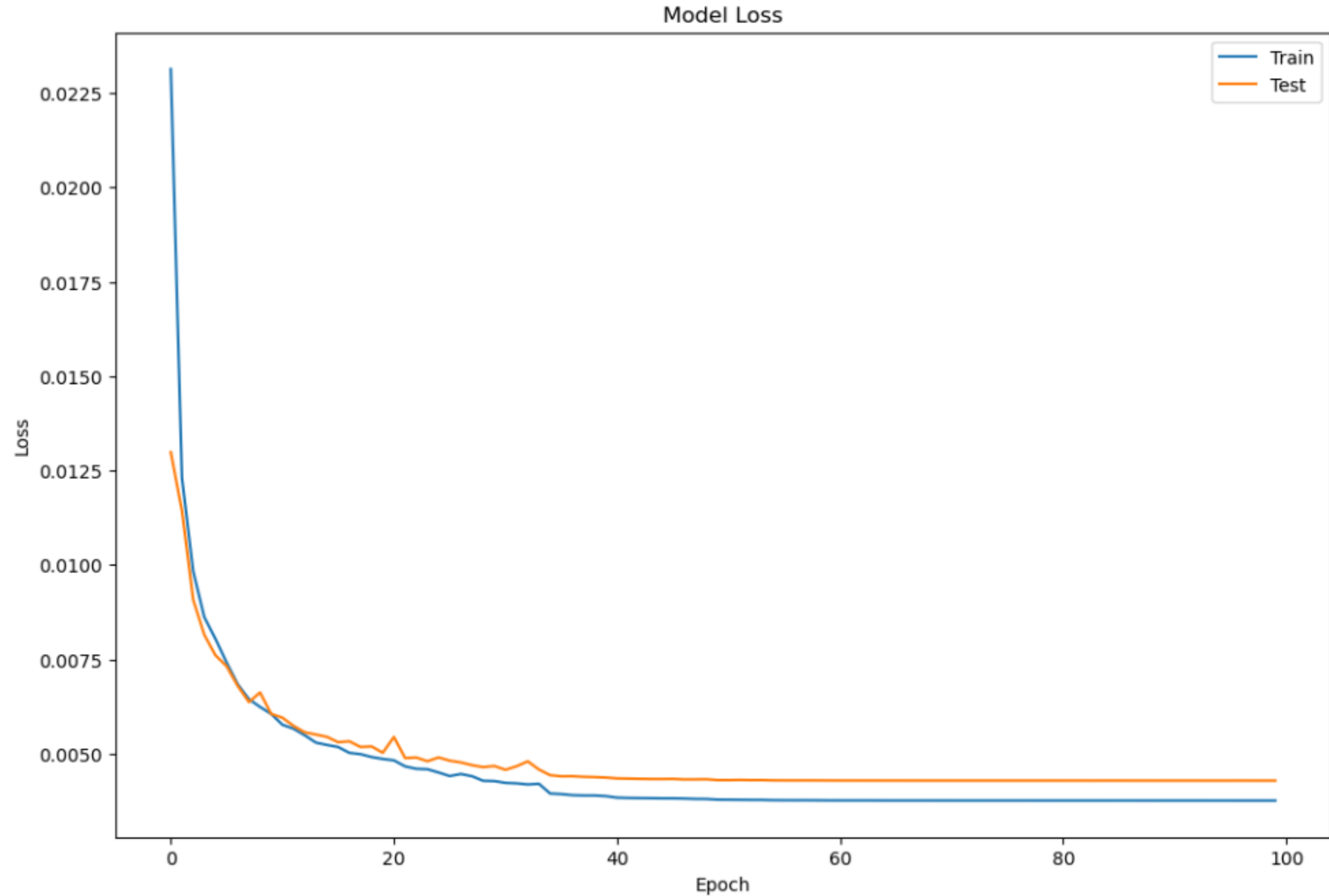
```
In [*]: fit(train_dataset, steps=150000)
```

Time taken for 1000 steps: 2862.04 sec



Step: 32k  
.....

# Model Loss





# Output Metrics

## PSNR (Peak Signal-to-Noise Ratio):

Metric commonly used to evaluate the quality of reconstructed or generated images

Higher Values: Typically, higher PSNR values indicate better image quality.

## Structural Similarity Index (SSIM):

Metric commonly used to evaluate the structural similarity between two images

Perfect Similarity: SSIM value of 1 indicates perfect similarity between images.

```
In [154]: ssims = []
          for inp, tar in train_dataset.take(30):
              ssim = tf.image.ssim(tar, generator(inp), max_val=1, filter_size=11,
                                   filter_sigma=1.5, k1=0.01, k2=0.03)
              ssims.append(ssim)
          tf.reduce_mean(ssims)
```

```
Out[154]: <tf.Tensor: shape=(), dtype=float32, numpy=0.60501456>
```

```
In [155]: psnrs = []
          for inp, tar in train_dataset.take(30):
              psnr = tf.image.psnr(tar, generator(inp), max_val=1)
              psnrs.append(psnr)
          tf.reduce_mean(psnrs)
```

```
Out[155]: <tf.Tensor: shape=(), dtype=float32, numpy=34.850668>
```



C:\Users\Nikhil  
e\Downloads\debl



```
plt.show()
```

```
python scripts/deblur_image.py --weight_path=/path/to/generator.h5 --input_dir=/path/to/image/dir --output_dir=/path/to/debl
```

Input Image



Deblur Image





Input Image



Debblur Image





# Future Scope

- Diversify the dataset by incorporating a range of blur types, scenes, lighting variations, and angles
- Explore sophisticated training methodologies such as WGAN-GP or similar variants to stabilize training dynamics
- Optimize the model for real-time or near real-time inference by exploring model compression techniques or deploying the model on edge devices
- Develop a user-friendly interface or an application showcasing the deblurring capabilities of the model, allowing users to upload and deblur their images



# Limitations

- Dataset Limitations Affecting Model Adaptability
- Training Duration Limitation
- Computational Intensity in GAN Training
- Evaluation Metrics Limitation
- Difficulty in assessing flipping images
- Real-Time Deployment Challenges

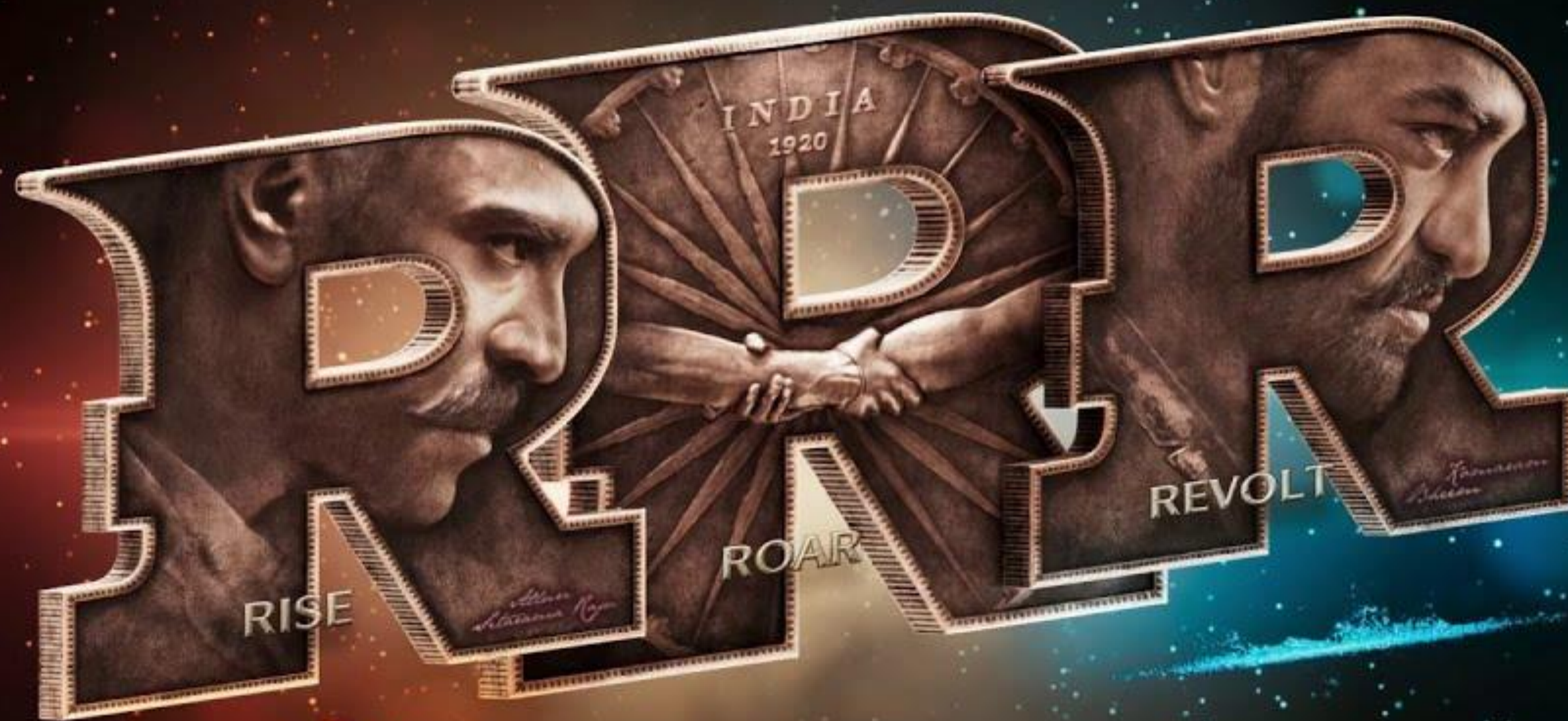




# Learnings

- Understanding the training process for both generator and discriminator networks in a GAN
- Importance of optimal data loading, preprocessing, and augmentation
- Utilization of hardware accelerators like GPUs and AWS to significantly speed up training
- Employment of AWS Lambda functions in preprocessing images
- SSIM and PSNR metrics used to evaluate the model on training and test sets effectively showcase the deblurring model's performance





THANK YOU