BDA Experience Project

Image Deblurring Using GAN

Team RRR

Nikhil Anne

Srivatsav Busi

Sahithi Nallani

Lokesh Paineni

Hemanth Kumar Enuguri

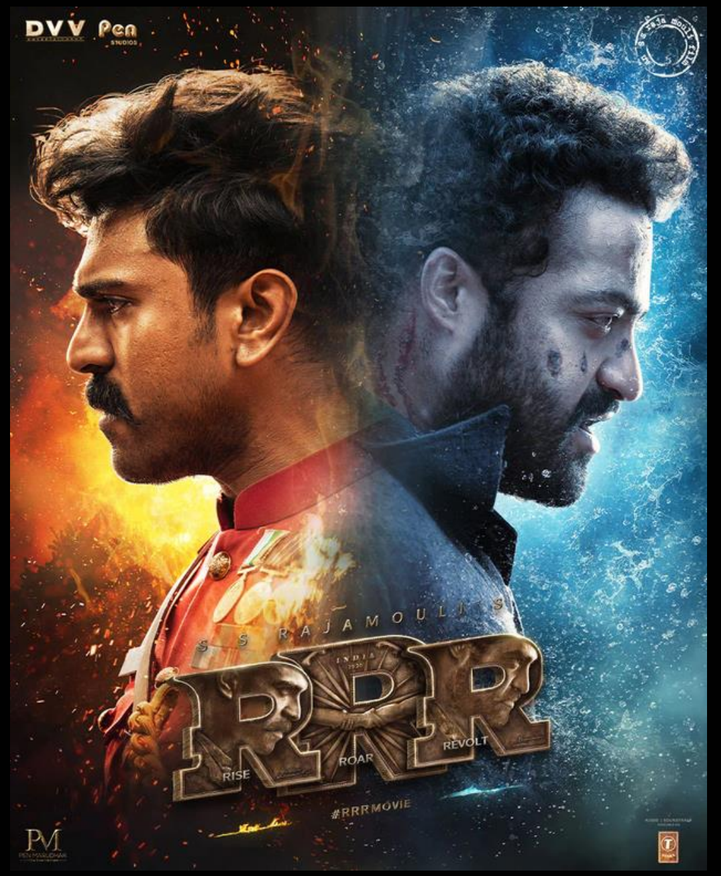


Table of Contents

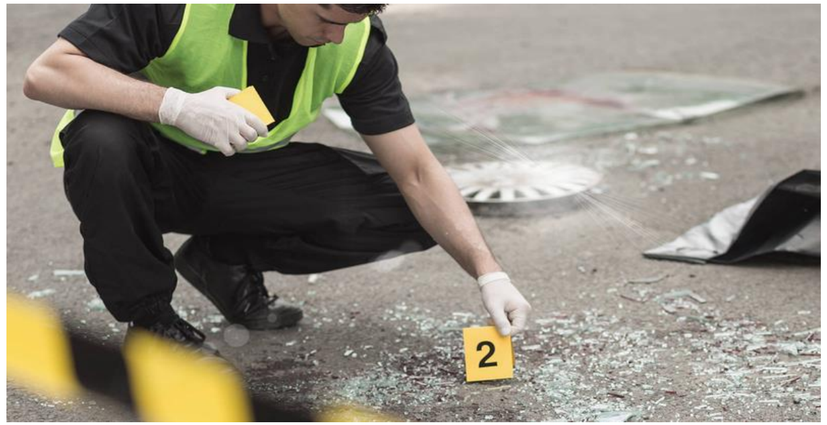
OBJECTIVE

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

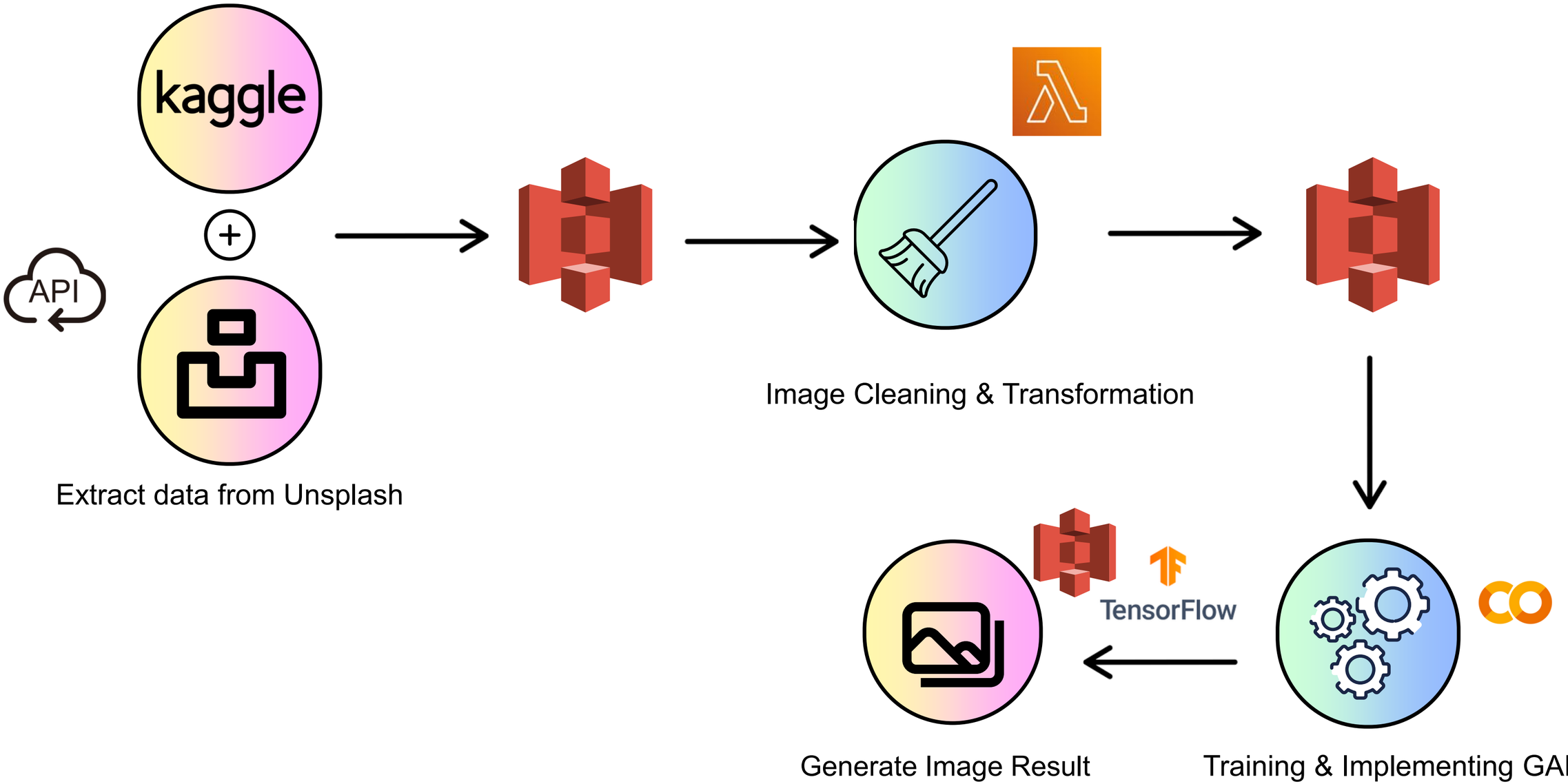
Property Appraisal & Real Estate:

- 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:

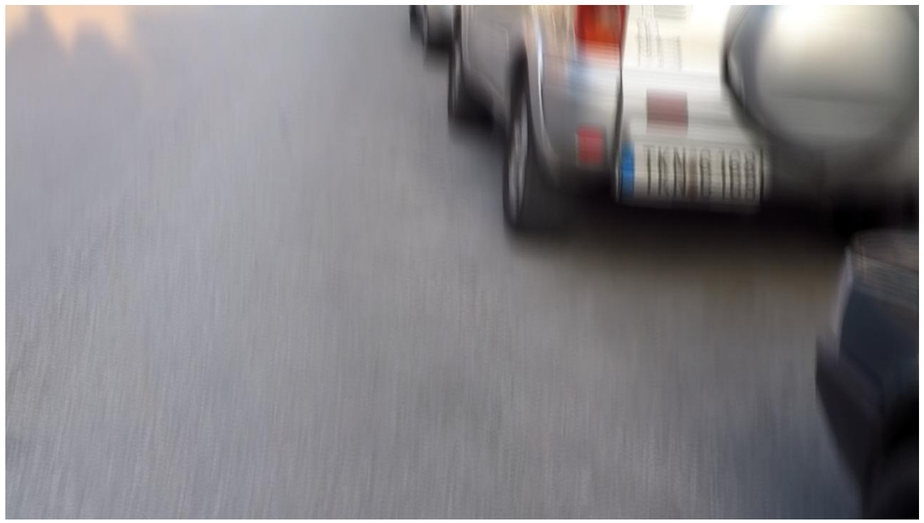
- Realistic immersive virtual tour experience
- Urban exploration games & training simulations
- Used in AR games to interact with real-world environments

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

Unsplash

Unsplash

Import required Libraries (Boto 3)

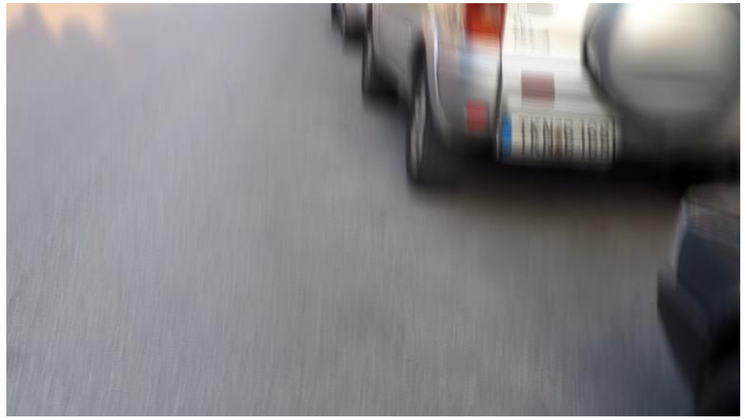
Set API keys and AWS credentials

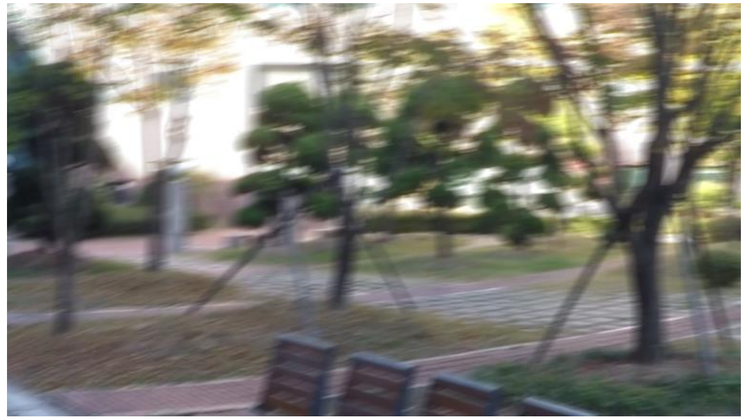
Define function to get sharp and blur image pair

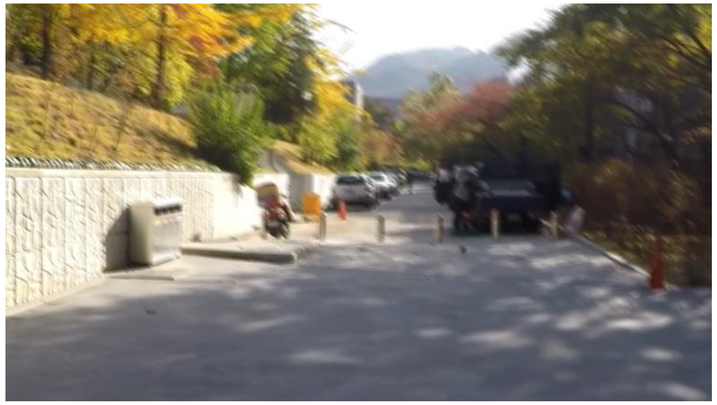
Calls the Unsplash API

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

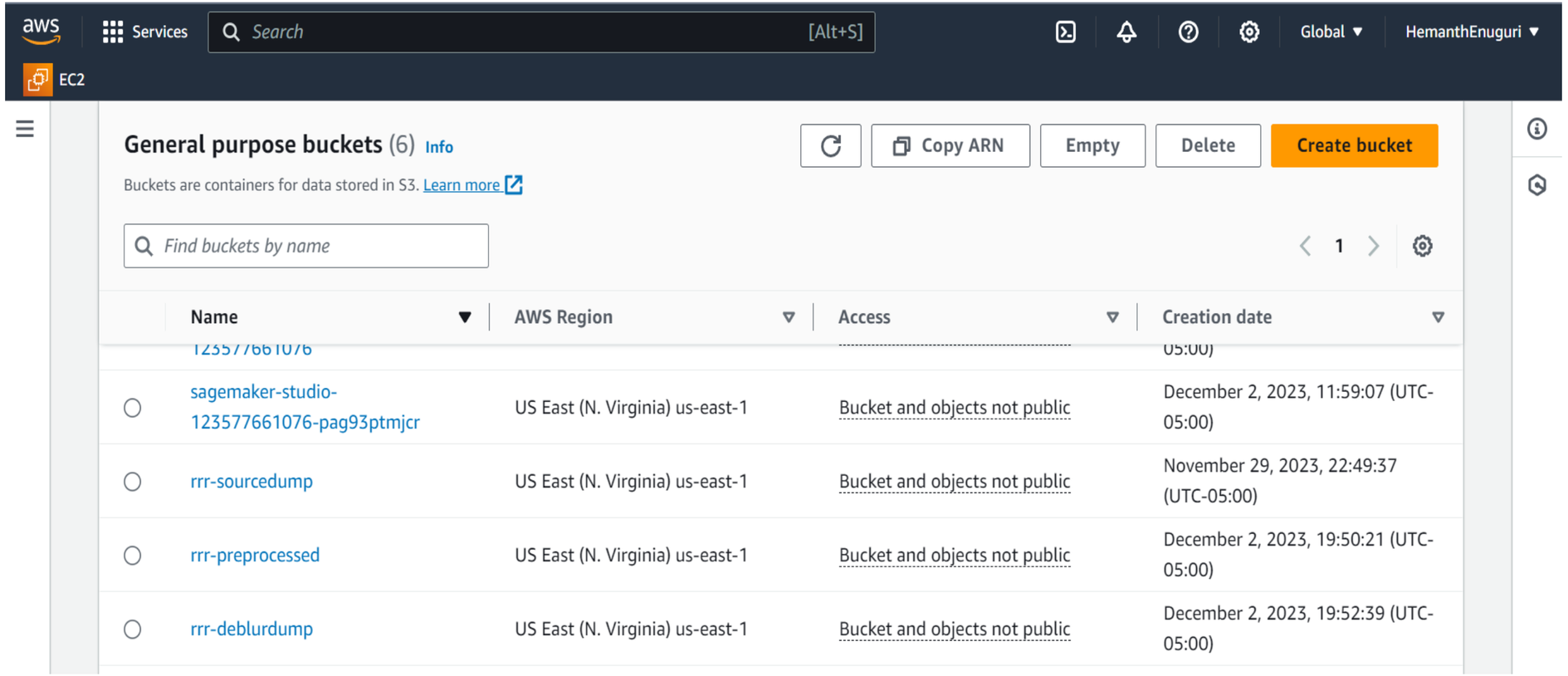
Preprocessed dump

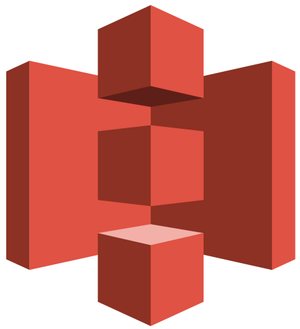
Deblurred Image dump

Source dump

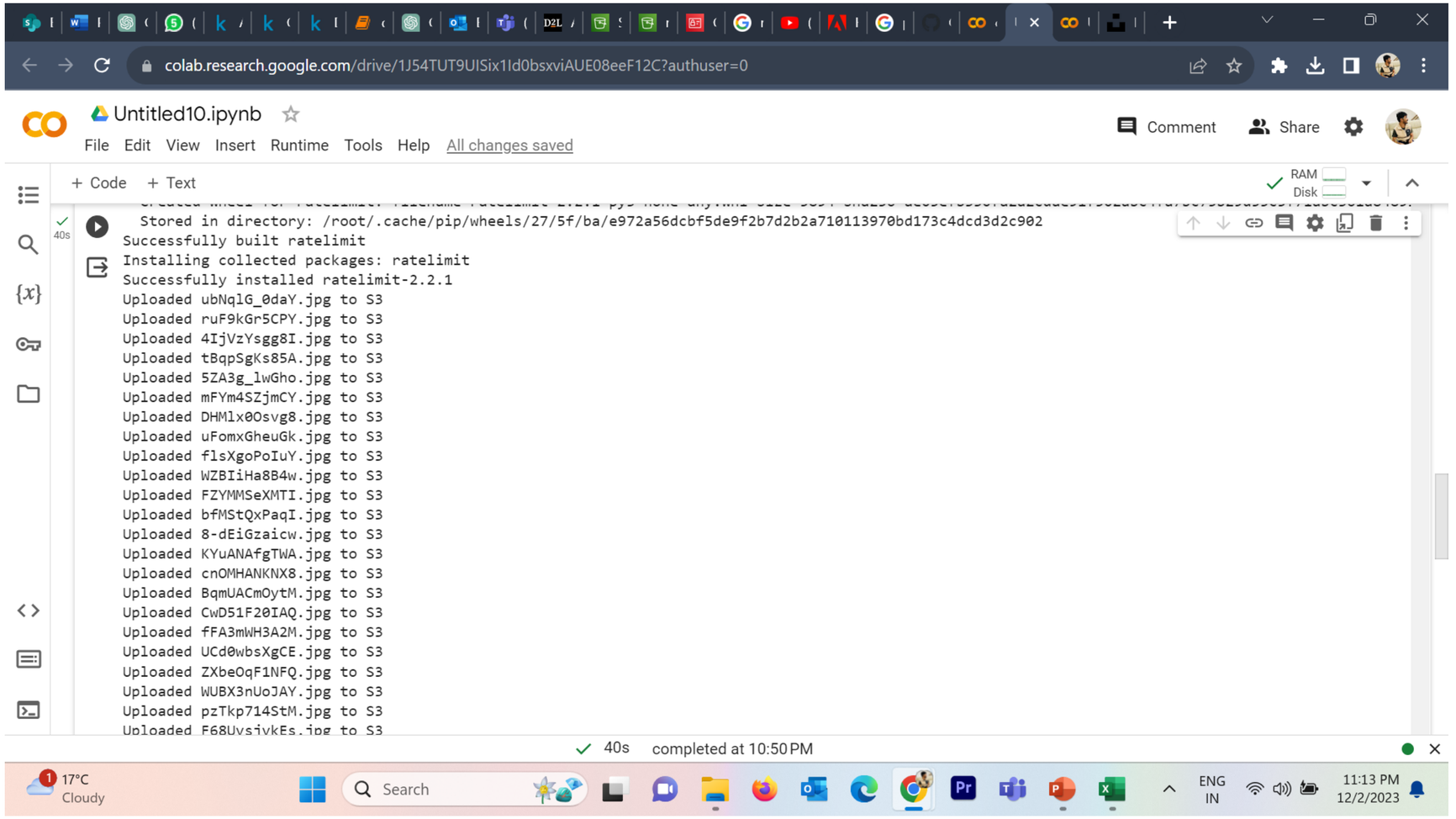
Preprocessed dump

Deblurred Image dump

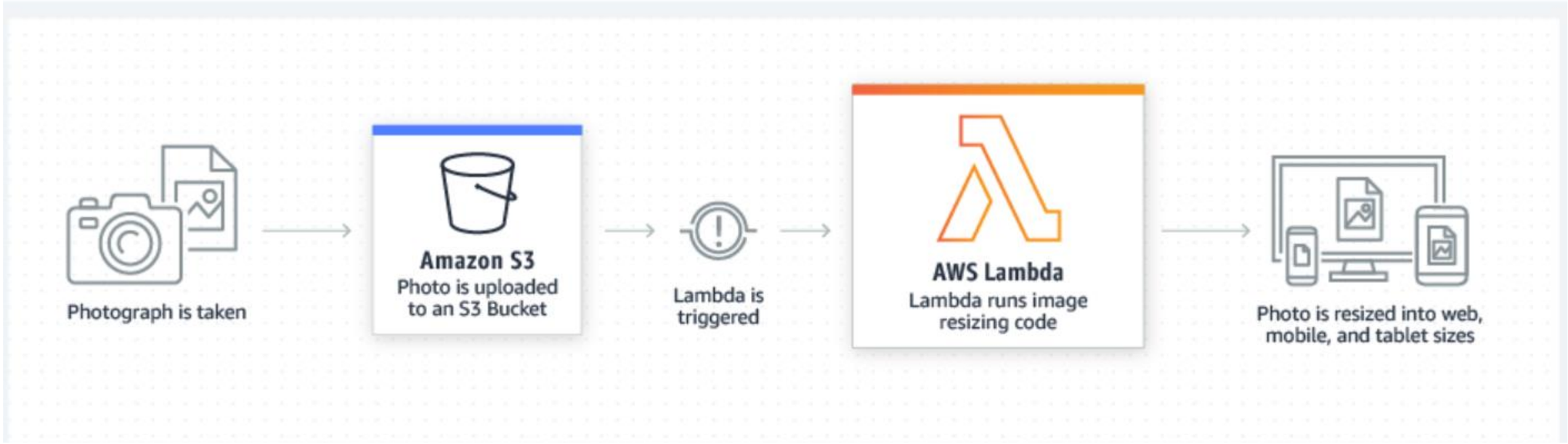




Source dump

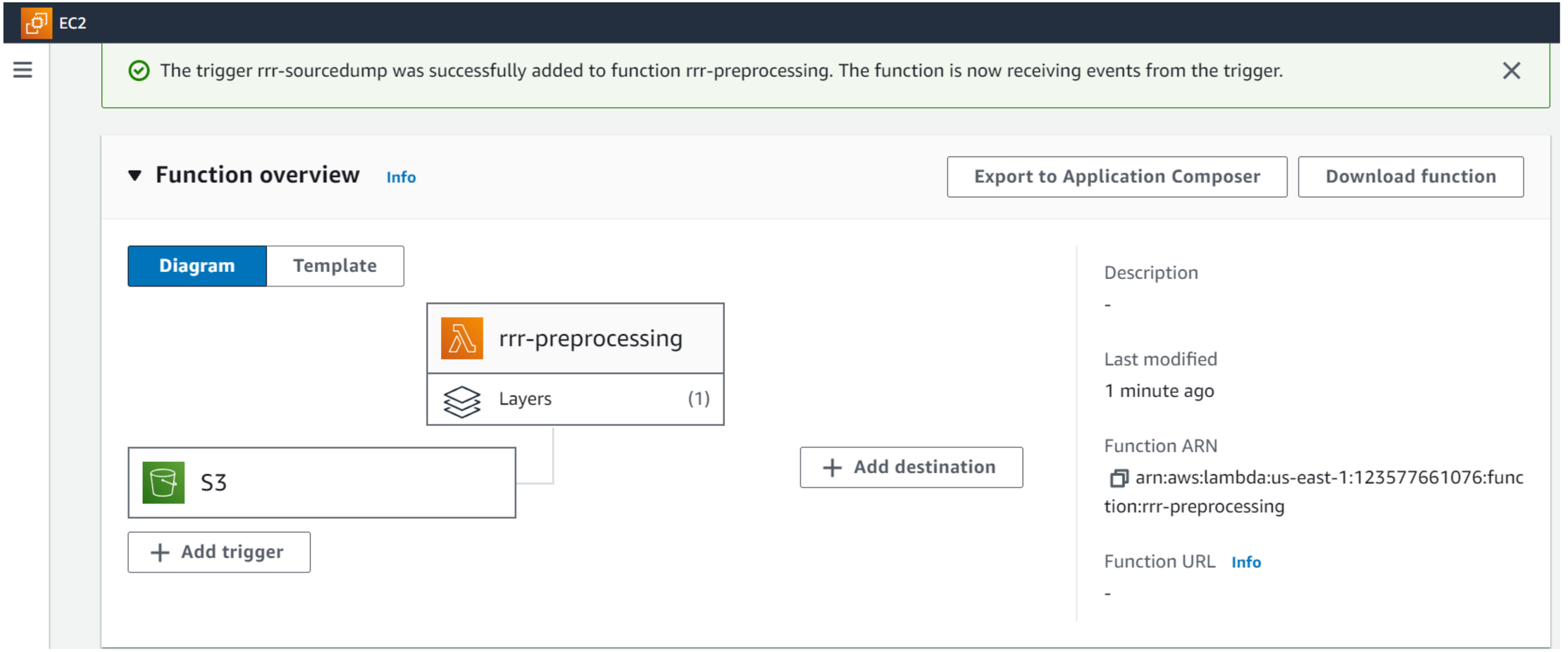

colab.research.google.com/drive/1J54TUT9UISix1Id0bsxviAUE08eeF12C?authuser=0
Untitled10.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
RAM
Disk
+ Code
+ Text
40s
Stored in directory: /root/.cache/pip/wheels/27/5f/ba/e972a56dcbf5de9f2b7d2b2a710113970bd173c4dcd3d2c902
Successfully 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 DHMlx0Osvg8.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
Uploaded F68UvsjvkEs.jpg to S3
40s completed at 10:50 PM
17°C Cloudy
Search
ENG IN
11:13 PM 12/2/2023

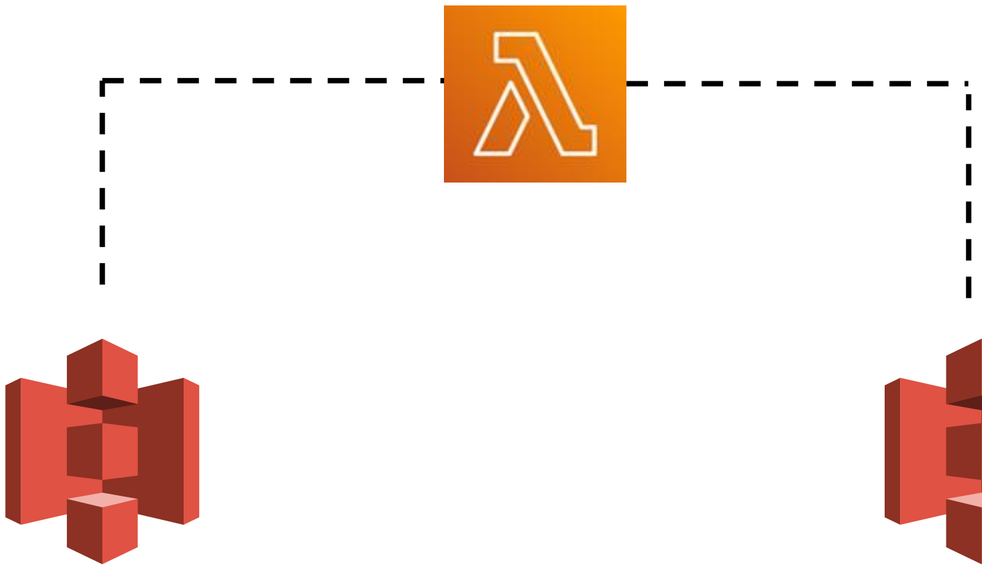
AWS Lambda


Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

AWS Lambda Implementation

Event trigger action





Source dump

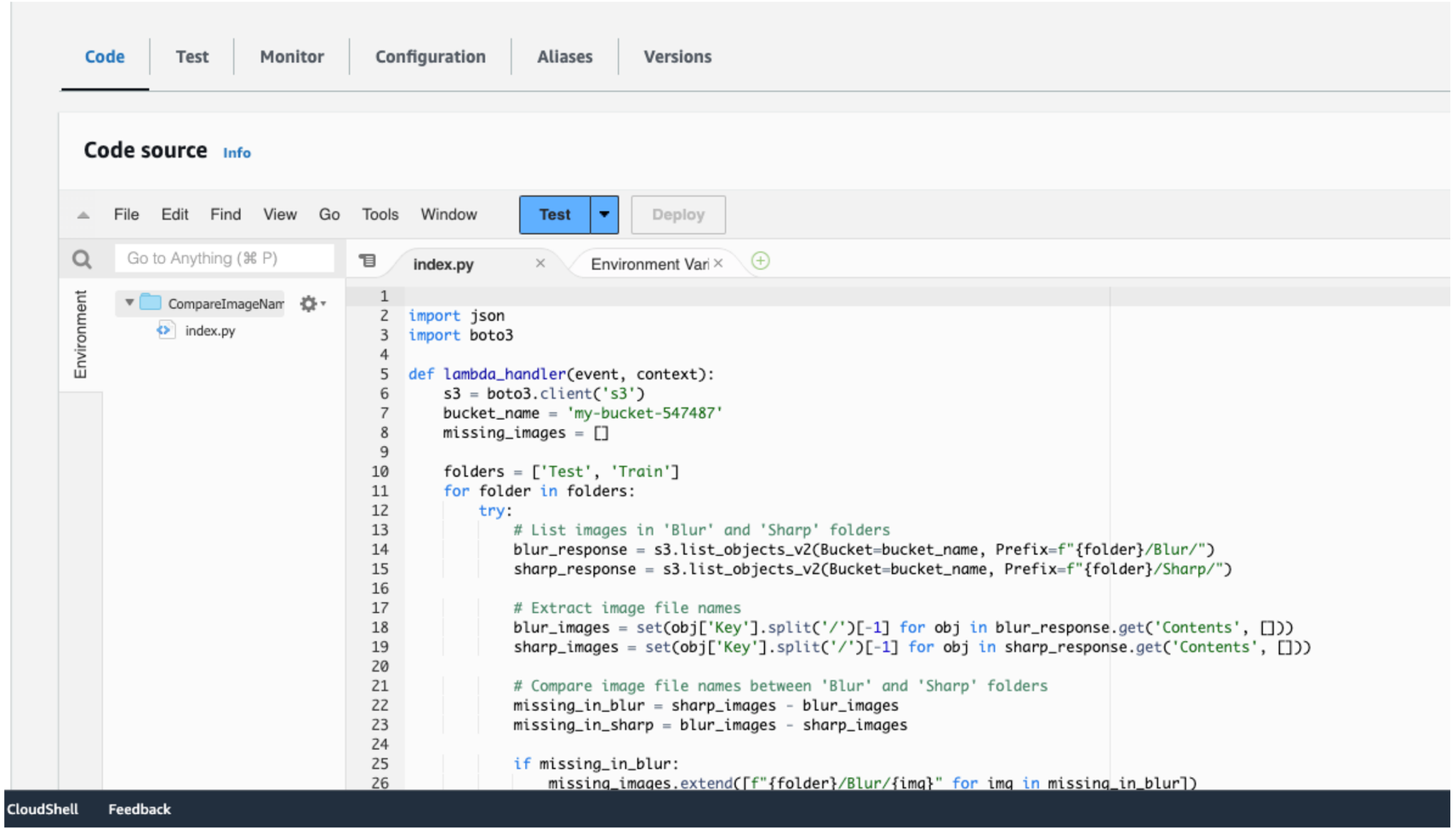
Preprocessed dump

Deblurred Image dump

Unsplash

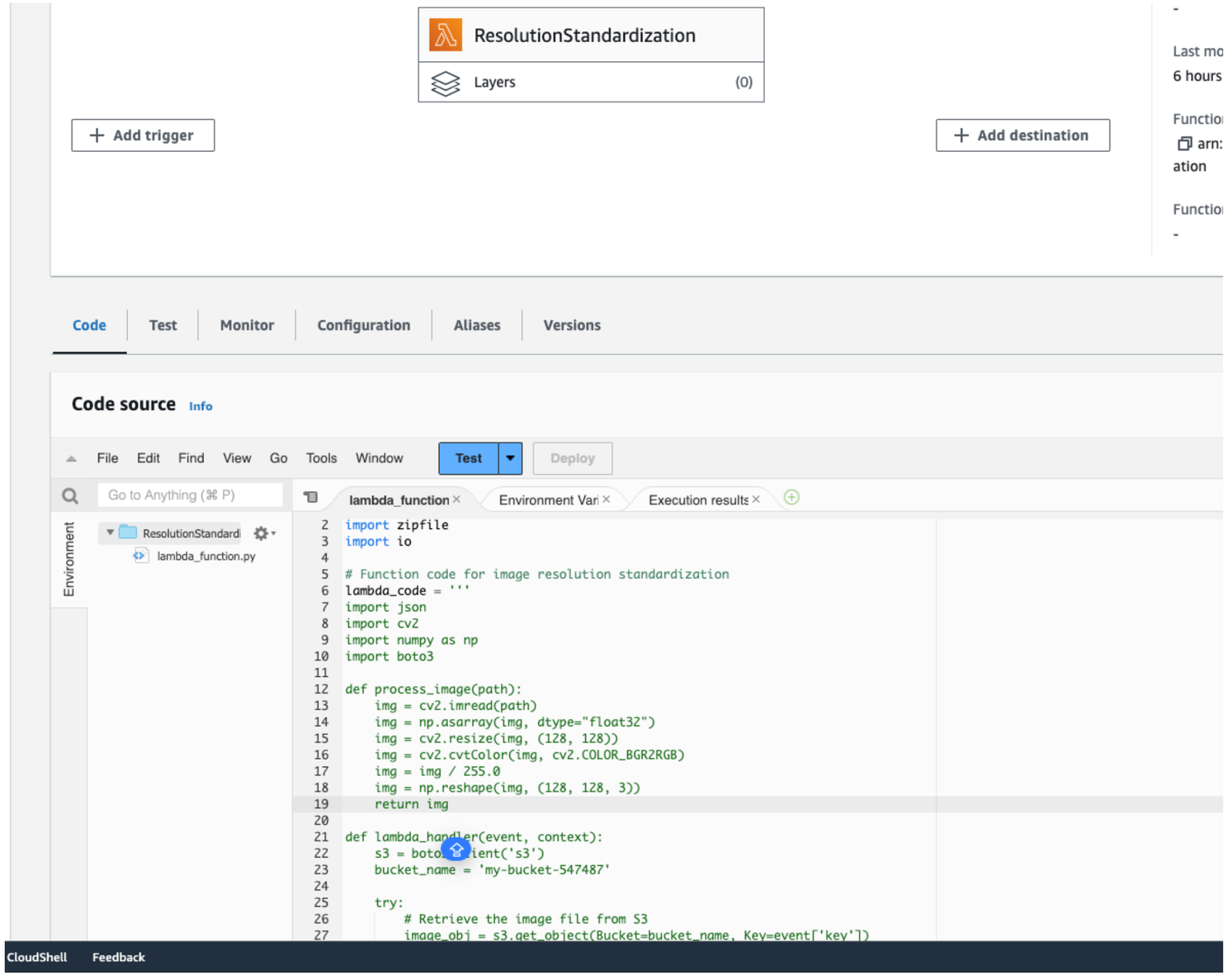
AWS Lambda Implementation

Missing images lambda function

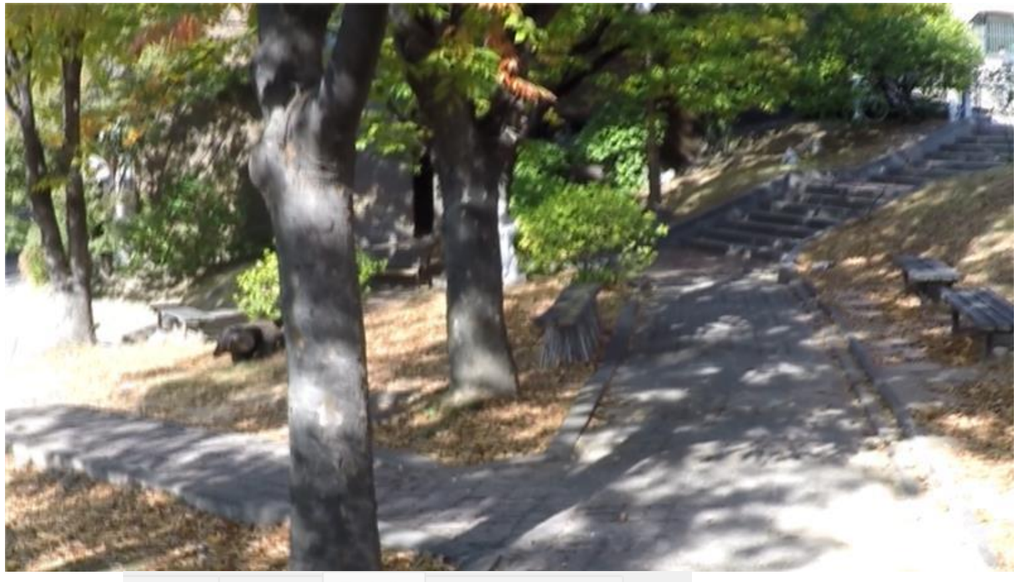


AWS Lambda Implementation

Image Resizing



Before Resizing



General | Security | Details | Previous Versions

Property	Value
Origin	
Date taken	
Image	
Dimensions	1280 x 720
Width	1280 pixels
Height	720 pixels
Bit depth	24
File	
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

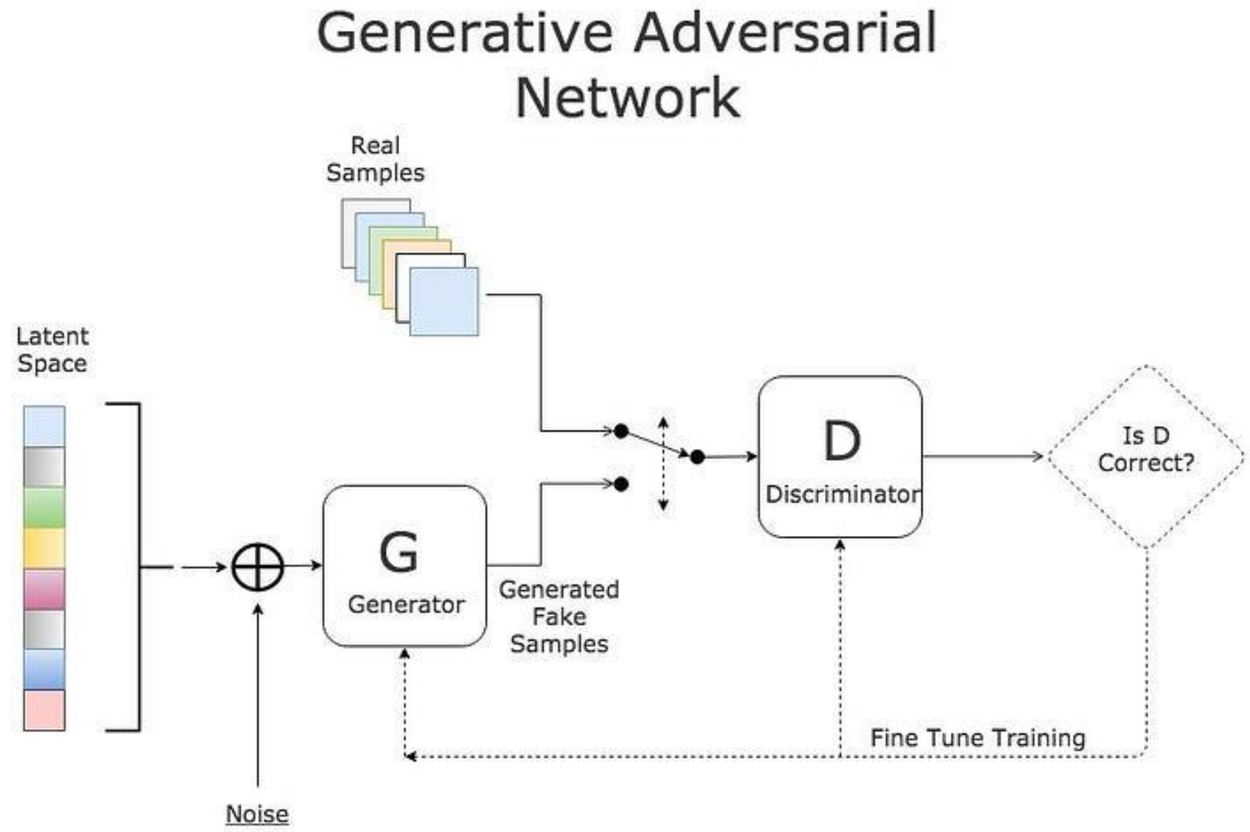
After Resizing

General | Security | Details | Previous Versions

Property	Value
Origin	
Date taken	
Image	
Dimensions	128 x 128
Width	128 pixels
Height	128 pixels
Bit depth	24
File	
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

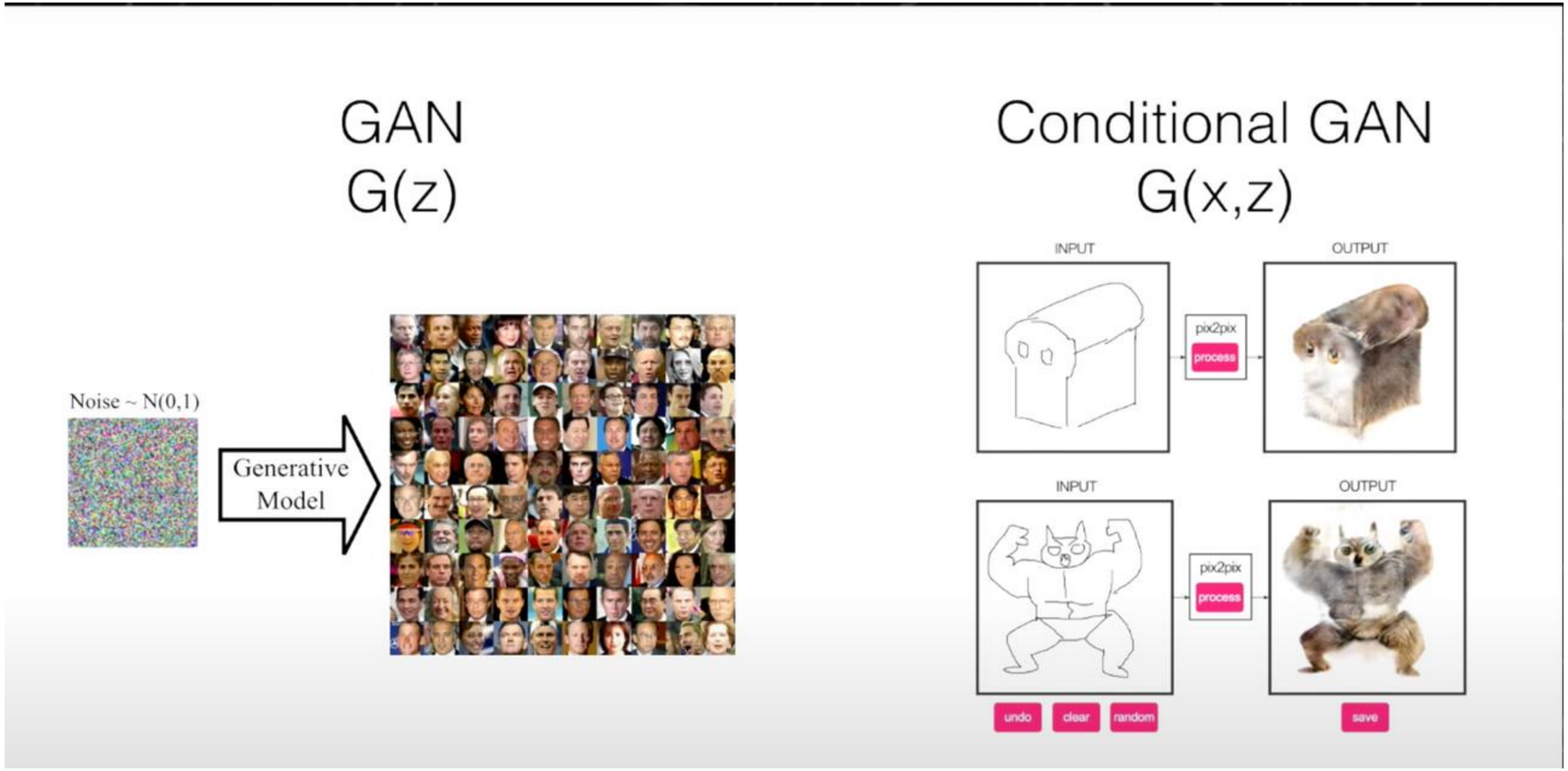
GAN

- 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



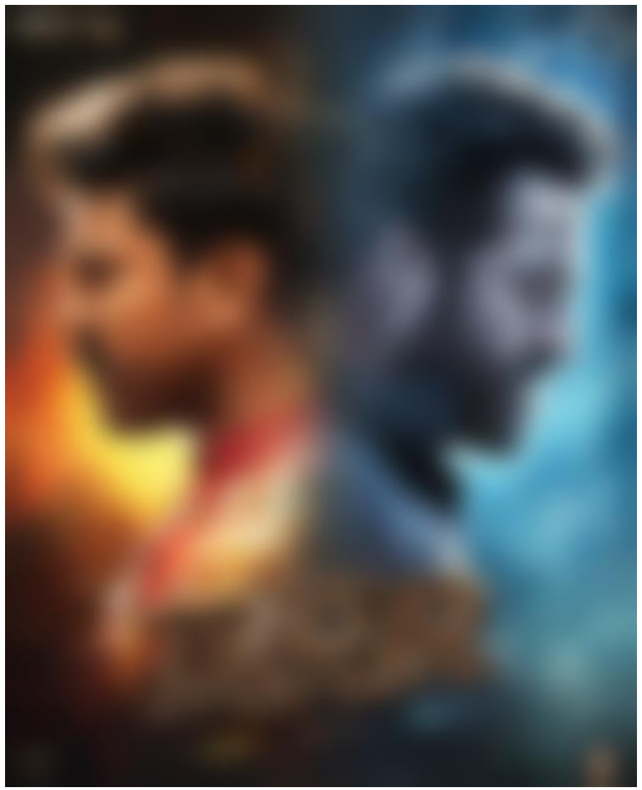
Conditional GAN

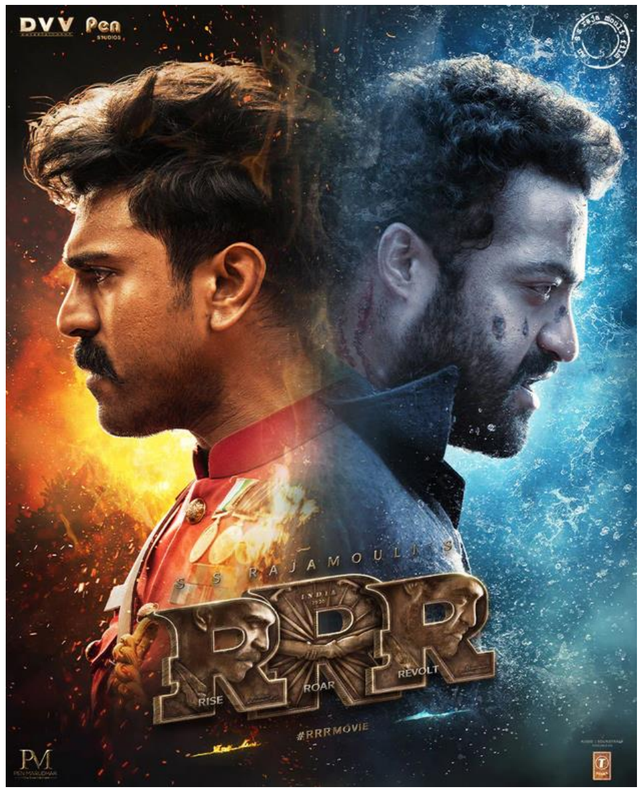
- Generate new data conditioned on some input
- This input can be anything from a text description to an image

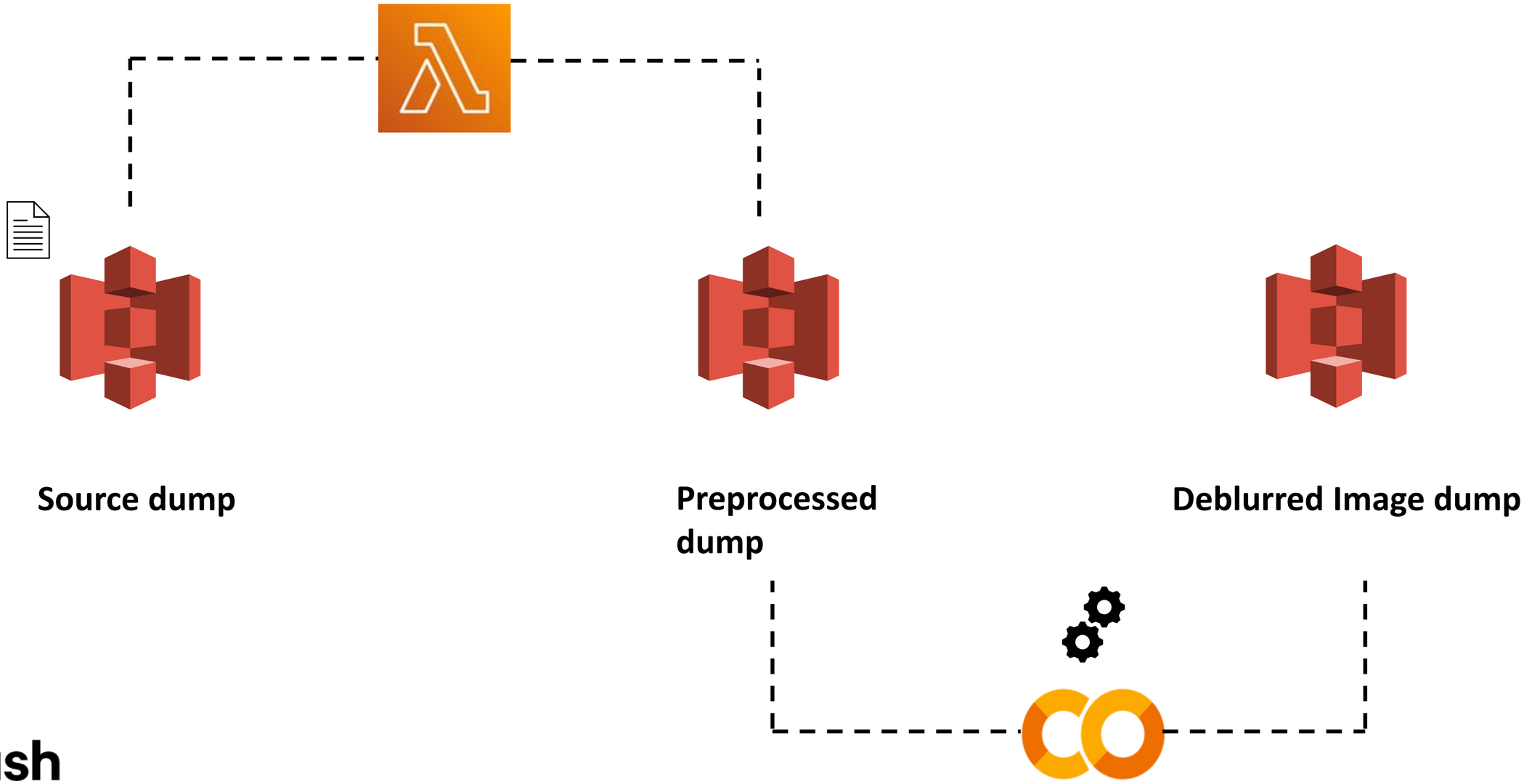


DeblurGAN

- 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
Unsplash

Train Data Sample Images

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


```
Step: 0k
.................................................
```

Testing

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


```
Step: 0k
```

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

```
Time taken for 1000 steps: 2880.86 sec
```


```
Step: 15k
.....................................................................................
```

Testing

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

```
Time taken for 1000 steps: 2880.86 sec
```

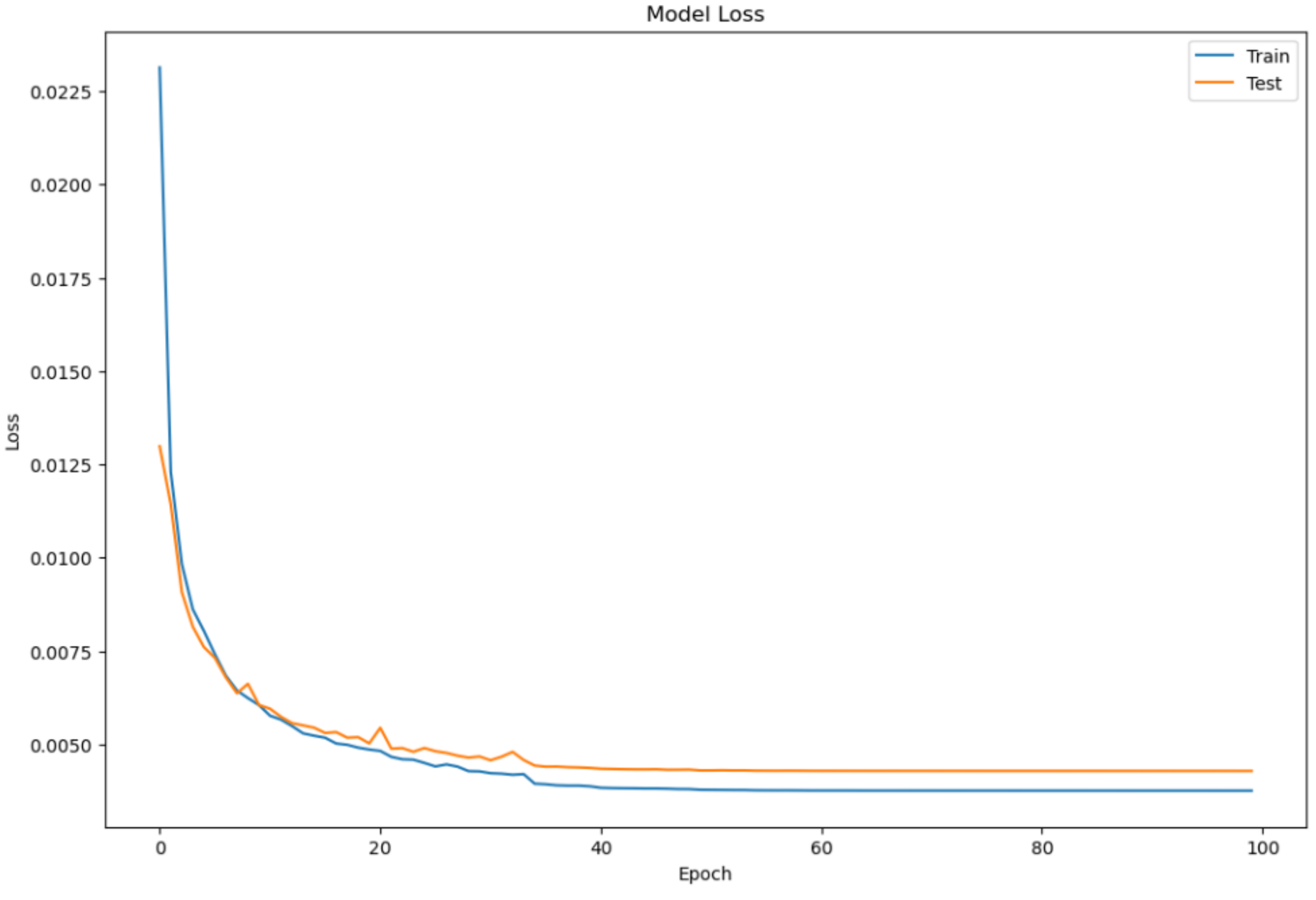

```
Step: 15k
.....................................................................................
```

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

```
Time taken for 1000 steps: 2862.04 sec
```


```
Step: 32k
.....................................................................................
```

Model Loss


Model Loss
Train
Test
Loss
0.0225
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0
20
40
60
80
100
Epoch

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

Debblur 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

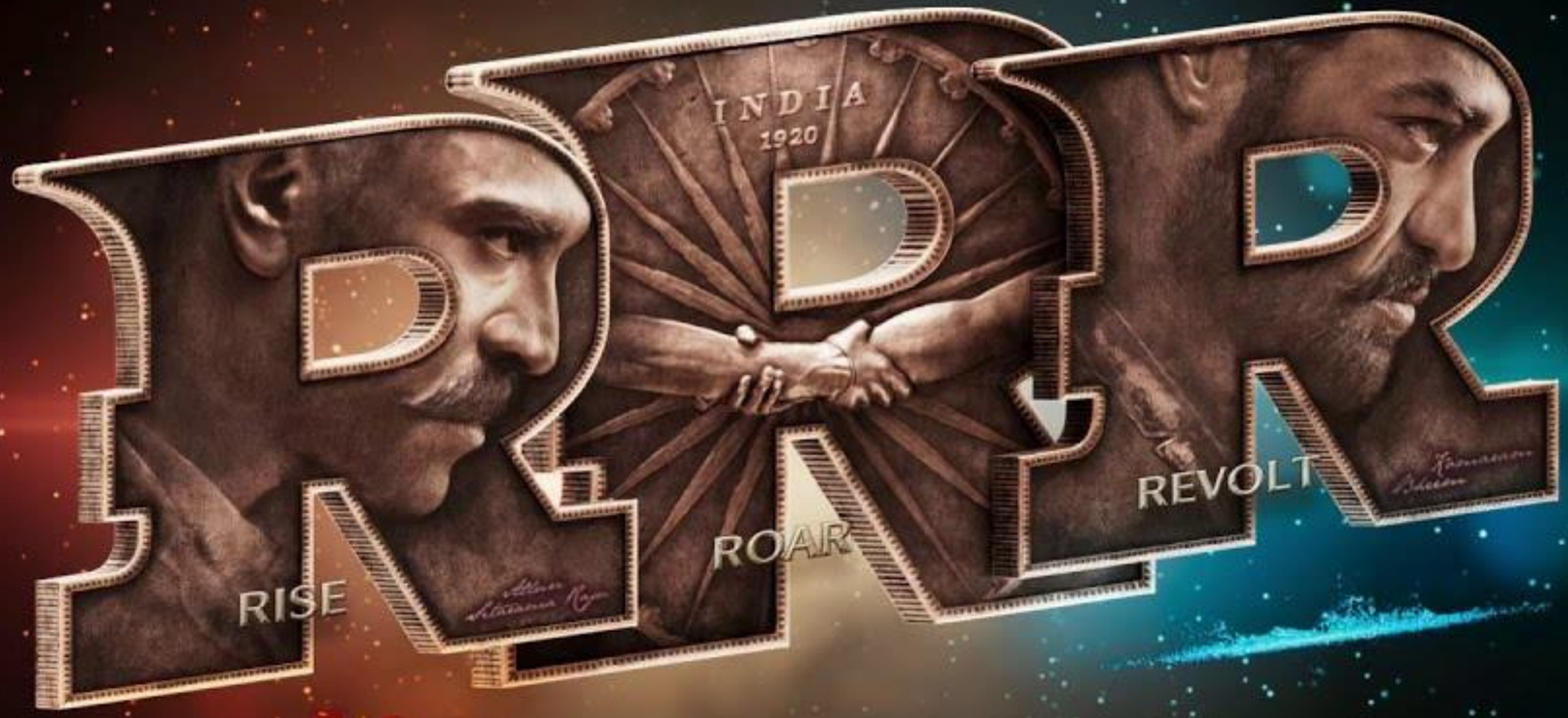
•Develop a user-friendly interface or an application showcasing the deblurring capabilities of the model, allowing users to upload and deblur their images

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