CIS 8395 - Big Data Analytics Experience (Fall 2023)

Project Final Paper

Image Deblurring Using GAN

Team - RRR

Nikhil Anne
Srivatsav Busi
Sahithi Nallani
Lokesh Paineni
Hemanth Kumar Enuguri

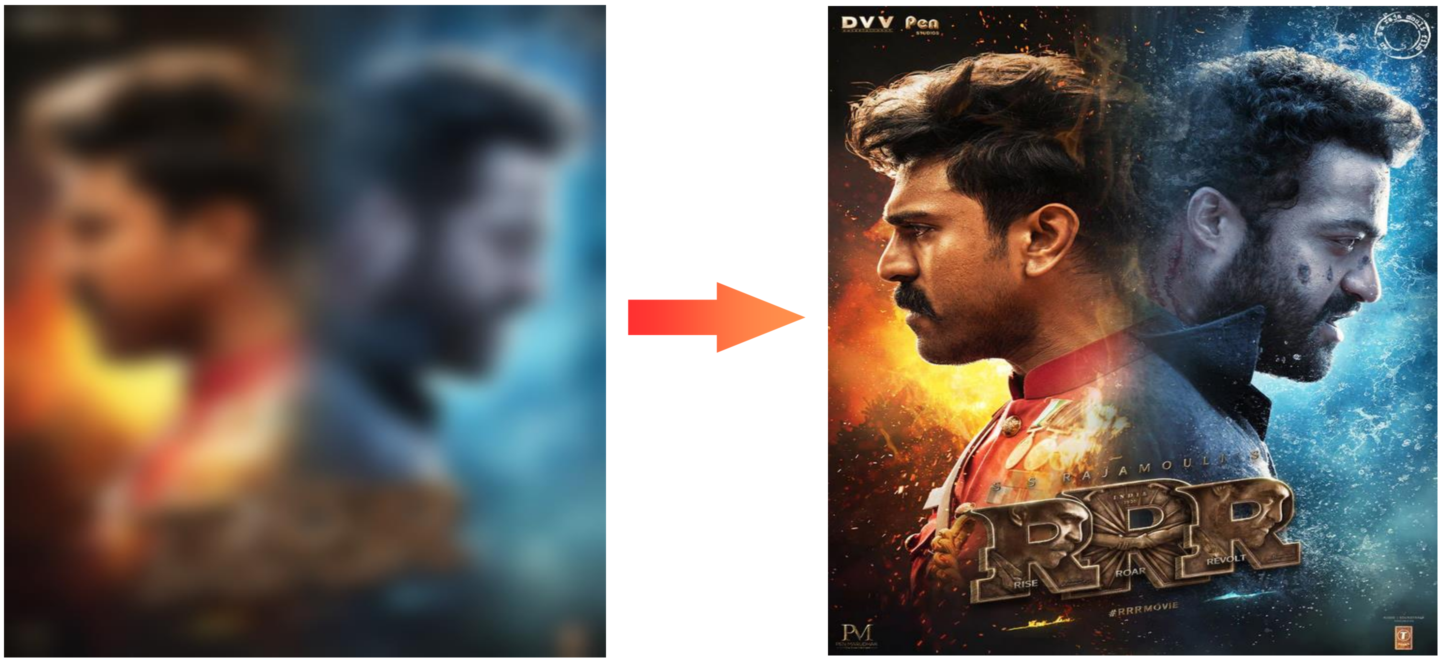


Table of Contents

1. Problem Statement

Blurred images from various street views provide a substantial difficulty in urban image processing, reducing their quality and utility. Current picture deblurring models frequently fall short of ideal accuracy and realism, limiting their usefulness in practical applications. Some of the practical problems which may arise due to blur images from street views include loss of critical details, reduced analytical accuracy in object recognition and understanding of scenes, impaired decision making in applications used in autonomous vehicles, surveillance equipment, forensic analysis etc., Some of the other problems posed by these blur street view images could be limiting the Virtual and Augmented reality experience, challenges in public safety and ineffective image retrieval from large databases.

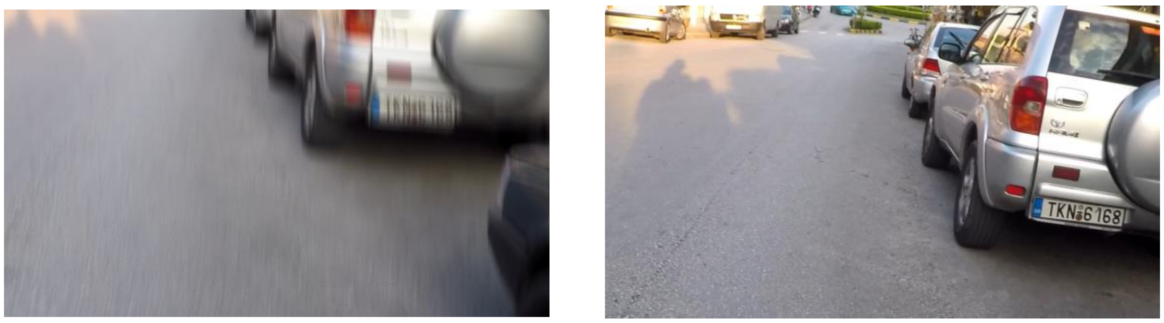


Fig. 1.1 Sharp and Blur image pair taken from GoPro dataset.

2. Proposed Solution

The goal of our project is to increase the quality and usability of blurred photographs taken from different street views. The objective is to create an innovative picture deblurring model that outperforms current approaches by leveraging a Generative Adversarial Network (GAN) for increased accuracy and realism. The images from our data sources are stored in AWS S3 storage and fed into the GAN model after several ETL transformations assuring data standardization.

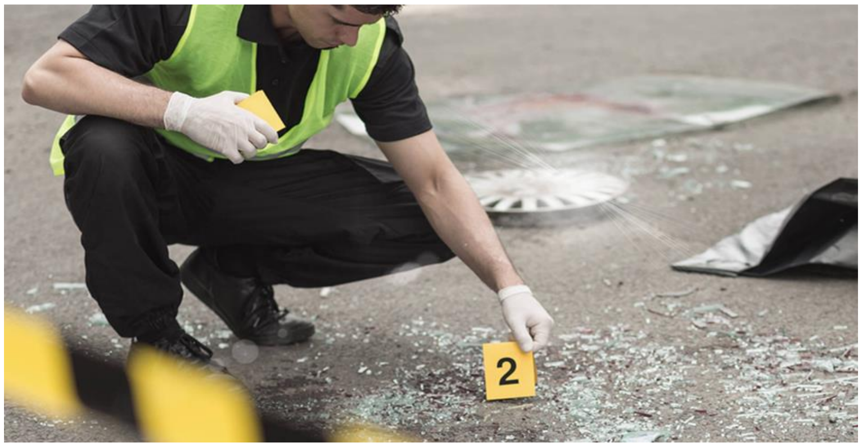
Our model aims to obtain improved outcomes in recovering crisp features from blurred photos which can be further used in several real-world applications. This project entails substantial research and development to increase the performance and robustness of the GAN-based deblurring model, eventually contributing to the improvement of image processing technology for better visual results in the context of street views.

3. Real World Applications:

Artificially blurred photos from numerous street views may be deblurred using Generative Adversarial Networks (GANs), which has several real-world applications, notably in computer vision and image processing. Here are some examples of how this technology may be used:

- **Forensic Analysis:**

  Law enforcement and forensic specialists use deblurring techniques to improve surveillance camera images that are essential for criminal investigations. These techniques are highly valuable assets for law enforcement agencies, including federal agencies and local police departments since they aid in extracting crucial information for crime investigation. Forensic experts, analysts, digital examiners, and crime scene investigators, play a crucial role in applying deblurring techniques to extract relevant information from images, contributing to the diligent processing and analysis of visual evidence. In the legal world, prosecutors and some of the defense attorneys employ deblurred images as impactful evidence in the court. They are used by prosecutors to support their arguments, while defense lawyers may refute the claims made by the prosecution. Judges and juries evaluate the admissibility and significance of deblurred photos inside the legal system, ultimately influencing crucial choices that impact the results of legal cases.



Improved Object Recognition:

Deblurred images are essential for the advancement of object recognition in many different fields. When it comes to autonomous driving, these improved photos are essential for accurate road feature recognition. By recognizing cars, pedestrians, signs, and obstructions, they help to guarantee the security and dependability of self-driving systems. Deblurred photographs provide more precise analysis of public amenities, roads, and infrastructure, which helps urban planners make well-informed judgments on city growth. Navigation systems also utilize deblurring to improve the detection of landmarks and navigational cues, giving users more precise directions. The deblurred images are useful for augmented reality applications that overlay digital information on the actual world, improving user experiences in travel, gaming, and education. They also help in identifying stores and signs, which improves the effectiveness of retail analytics, location-based advertising, and market research. The enhanced object recognition capabilities enabled by deblurred images also have significant benefits for public safety, environmental monitoring, robotics, industrial inspection, healthcare, and geospatial mapping. These benefits affect a wide range of fields, from emergency response to medical diagnostics and geographic information systems.

- **<u>Property Appraisal & Real Estate:</u>**

  Enhancing image quality can help with virtual property tours, real estate marketing, and property appraisal by giving prospective tenants or buyers a more accurate picture of the properties. Deblurred photographs provide a sharper and more detailed perspective of both interior and exterior property features, which has a huge impact on the real estate sector. When performing remote or virtual assessments, real estate appraisers find this to be quite helpful as it enables them to make more accurate assessments of the condition and value of properties. Improved picture quality is essential for real estate marketing since listings with eye-catching photos draw in more prospective tenants or purchasers, which may result in faster turnover and higher sales prices. With the use of deblurred images, virtual property tours are becoming more and more common as they allow for remote property investigation and give a thorough grasp of layouts and features. Higher image quality boosts user engagement and conversion rates on online property listings and platforms. Furthermore, crisp photos aid in the documentation and communication of a property's state during property inspections, promoting open dealings and higher client satisfaction. Real estate agents that regularly deliver high-quality photos have an advantage over their competitors since they draw in more business and may even see an increase in sales price, which strengthens their standing in the market.

- **<u>VR, AR & Gaming:</u>**

  Deblurred street view images are essential for improving virtual reality (VR) and gaming experiences because they create more captivating and immersive virtual worlds. Applications for the enhanced image quality can be found in many different contexts, including virtual travel experiences that let people explore cities and landmarks in a realistic manner from the comfort of their homes. Deblurred images aid in the creation of realistic and detailed city scenes in urban exploration games, allowing players to partake in activities such as parkour and narrative-driven excursions. The technology is useful for emergency response, military, and urban planning training simulations because it provides realistic urban surroundings for hands-on practice in a controlled environment. Deblurred images in virtual reality (VR) let architects and urban planners visualize and convey concepts in a realistic urban setting, which helps with project development and public involvement. Deblurred images are even used in educational apps to recreate historical cityscapes, which makes historical discovery and learning easier. They are also used by video game producers to create more realistic and immersive gaming worlds, which increases player engagement and believability.

4. Architecture

Below fig.1.2 is the high-level architectural diagram for our project from Data Sourcing, Cleaning, Model Training, and Inferencing. This diagram helps us to understand the architectural components involved which will be further explained below.

Combining **Unsplash** and **Kaggle** datasets requires a systematic strategy to combining disparate sources of picture data into a unified and complete dataset. To ensure consistency and uniformity, the datasets from Unsplash and Kaggle must first be filtered and preprocessed. This includes dealing with differences in picture formats, information structures, and any other discrepancies between the two databases.

Architecture

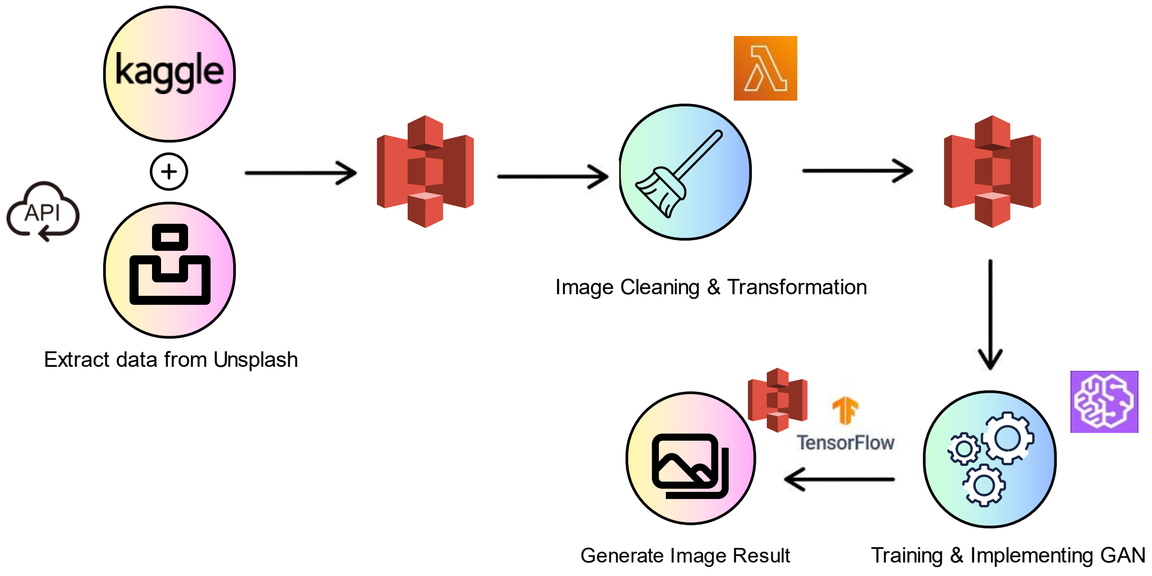

kaggle
API
Extract data from Unsplash
Image Cleaning & Transformation
TensorFlow
Generate Image Result
Training & Implementing GAN

Fig. 1.2 High Level Architecture

1. **ETL:** We primarily use the GoPro dataset provided from Kaggle in our ETL (Extract, Transform, Load) process and augment it with additional data gathered from Unsplash via their API. All acquired images are stored in a centralized repository for our diverse image datasets, AWS S3. This data flow is orchestrated by AWS Lambda functions, which are triggered by the availability of fresh data in the S3 bucket. When activated, these Lambda functions methodically scan through the newly supplied photos, applying pre-defined modifications designed to assure consistency and improve data quality. The changed data is then effortlessly placed into a new S3 bucket, enabling an automated and scalable workflow that effectively combines disparate datasets, conducts necessary transformations, and assures ongoing enrichment.
2. **ML Model:** Prepared images will be accessed by GAN model builder running in AWS Sage maker. GAN approaches are used to broaden the dataset by producing synthetic images, hence

increasing the model's robustness and generalizability. This method employs Generative Adversarial Networks (GANs), a form of machine learning model comprised of a generator and a discriminator. The generator generates synthetic pictures, which are then evaluated by the discriminator, resulting in a feedback loop that improves the generator's capacity to generate realistic data. This augmentation helps the model encounter a broader range of scenarios, improving its capacity to handle diverse inputs during training. The synthetic images produced by the GAN contribute to a more robust model by exposing it to variations and complexities that may not be adequately represented in the original dataset. By introducing GAN-generated images, the model learns to generalize more effectively, which means it becomes adept at dealing with previously unseen or slightly different instances of sharp and blurred images.

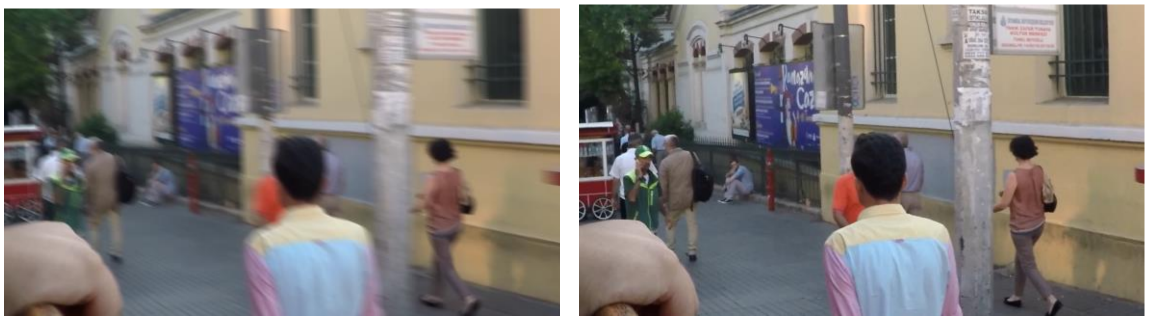
3. **Storage (AWS S3 Bucket):** AWS S3 stands as a fundamental storage service, offering a remarkably robust, scalable, and secure solution for object storage. At its core, S3 revolves around two primary elements: buckets and objects. Buckets serve as top-level containers, uniquely named and residing within specific AWS regions, providing a structured means to organize and store various digital objects. Objects, on the other hand, represent the individual files stored within these buckets, encompassing a broad array of file types, including images, videos, documents, and more.

   Following the GAN model's successful production of synthetic data, the next critical phase requires systematic storage for smooth integration into the workflow. This synthesis dataset is effectively stored in specified AWS S3 buckets and is augmented with various photos that contribute to the model's resilience. AWS S3 acts as a reliable and scalable repository, enabling for easy retrieval and use in subsequent phases of an image deblurring workflow. Structured storage in S3 not only simplifies data management but also allows for easy access and integration with other AWS services or external applications as needed.

5. Data Source:

Primary Source:

One of the datasets which we are using for this project is GoPro dataset, which is taken from Kaggle, containing 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. The images cover a wide range of scenes, including outdoor environments, with varying levels of blur.

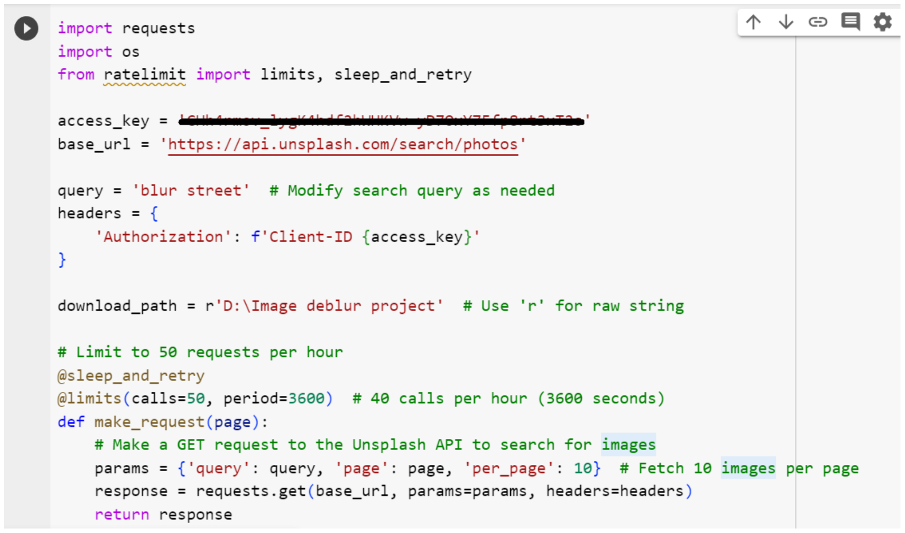


The images in the dataset are blurred using a variety of different methods, such as camera shake, fast motion, out-of-focus objects, etc. This helps the GAN to learn to deblur images that have been blurred by different types of blurs. The images in the GOPRO dataset are also decomposed into subfolders by scenes. This helps the GAN to learn to deblur images of different types of scenes, such as city streets etc.,

We used the GOPRO dataset to train a generative adversarial network (GAN) for image deblurring. GANs are a type of machine learning model that can be used to generate realistic images. In this case, the GAN was trained to generate sharp images from the blurred images in the GOPRO dataset.

Secondary Source:

Unsplash is a website dedicated to proprietary stock photography. The photos on Unsplash are free to use and can be used for most commercial, personal projects, and for editorial use.The website claims over 330,000 contributing photographers and generates more than 13 billion photo impressions per month on their growing library of over 5 million photos. By using the API, we access the image repository and retrieve photos based on specific criteria or search terms, in our case it was blur and sharp images. The code iniates a GET request to the Unsplash API endpoint (https://api.unsplash.com/search/photos) using the requests.get() function. The Unsplash API access key is included in the request headers using the Authorization header to authenticate the request.



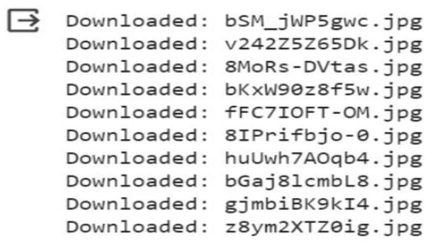
```python
import requests
import os
from ratelimit import limits, sleep_and_retry

access_key = '[redacted]'
base_url = 'https://api.unsplash.com/search/photos'

query = 'blur street'  # Modify search query as needed
headers = {
    'Authorization': f'Client-ID {access_key}'
}

download_path = r'D:\Image deblur project'  # Use 'r' for raw string

# Limit to 50 requests per hour
@sleep_and_retry
@limits(calls=50, period=3600)  # 40 calls per hour (3600 seconds)
def make_request(page):
    # Make a GET request to the Unsplash API to search for images
    params = {'query': query, 'page': page, 'per_page': 10}  # Fetch 10 images per page
    response = requests.get(base_url, params=params, headers=headers)
    return response
```



```
Downloaded: bSM_jWP5gwc.jpg
Downloaded: v242Z5Z65Dk.jpg
Downloaded: 8MoRs-DVtas.jpg
Downloaded: bKxW90z8f5w.jpg
Downloaded: fFC7IOFT-OM.jpg
Downloaded: 8IPrifbjo-0.jpg
Downloaded: huUwh7AOqb4.jpg
Downloaded: bGaj8lcmbL8.jpg
Downloaded: gjmbiBK9kI4.jpg
Downloaded: z8ym2XTZ0ig.jpg
```

The rate limit we mentioned, 50 requests per hour, typically means we can make up to 50 HTTP requests to the API endpoints in one hour. Each request returns a single page of data (e.g., 10 items per page), and the total number of items we can retrieve depends on the API's pagination limit. If the API's maximum pagination limit is 30 items per page, we could retrieve up to 30 items per request. So, with 50 requests per hour, and each request retrieving 30 items (the API's maximum pagination limit), we could retrieve up to 50 requests * 30 items = 1500 items per hour. These items could be images, or any other type of data specified by the API.

The search results are paginated, and the script retrieves the picture URL and unique identifier for each photo in the results. The image is then downloaded and saved as a JPG file in a directory named after the search query. We have organized the images into distinct folders, namely one for sharp images and another for blurred images.

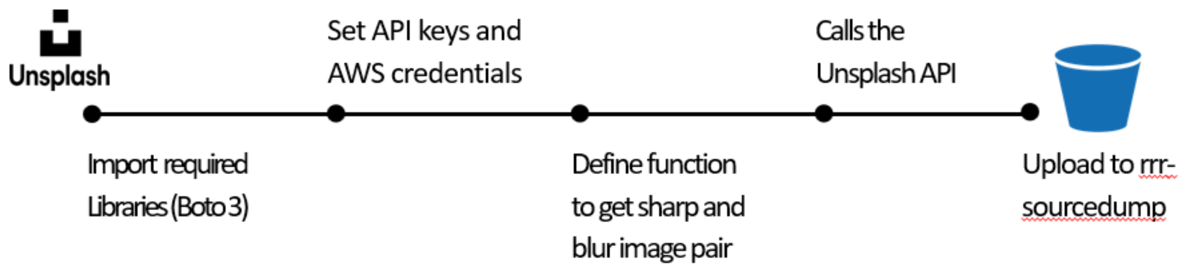

Unsplash
Set API keys and AWS credentials
Calls the Unsplash API
Import required Libraries (Boto3)
Define function to get sharp and blur image pair
Upload to rrr-sourcedump

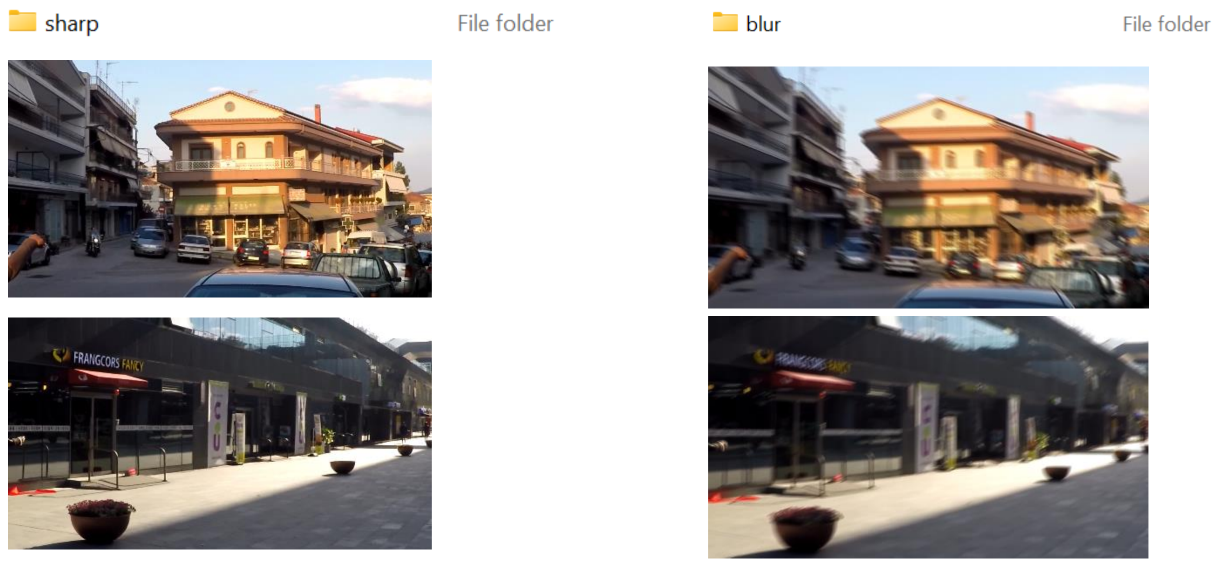
Fig. 5.1 Extracting images from Unsplash API

The diagram shows that the first step is to import the required libraries, which is Boto3. Boto3 is a Python library that provides access to AWS services. and set the API keys and AWS credentials. This is necessary to authenticate with Unsplash and AWS.

The next step is to define a function to get a sharp and blurred image pair. This function will take an Unsplash image URL as input and return a sharp and blurred version of the image. The final step is to upload the sharp and blurred images to a repository called "rrr-sourcedump".

The purpose of this data source is to provide sharp and blurred image pairs for training machine learning models. The sharp images can be used to train models to recognize objects and features in images. The blurred images can be used to train models to be robust to noise and occlusions.

6. Sample Images



Here it shows the segregation of both the sharp and blur image pairs in respective folders.

7. Preprocessing Data

Data preprocessing involves a series of actions and methods employed to ready and refine raw data prior to its utilization in analysis or machine learning. These steps are essential to ensure that the data is in a suitable format, is accurate, and contains the information needed for the intended purpose. Data preprocessing typically includes tasks like cleaning, transforming, and organizing data to make it more manageable and reliable.

7.1 Data Cleaning:

Data cleaning involves preparing and processing image data to ensure that it is accurate, complete, and consistent. Here are some common techniques we have used in data cleaning of our input data:

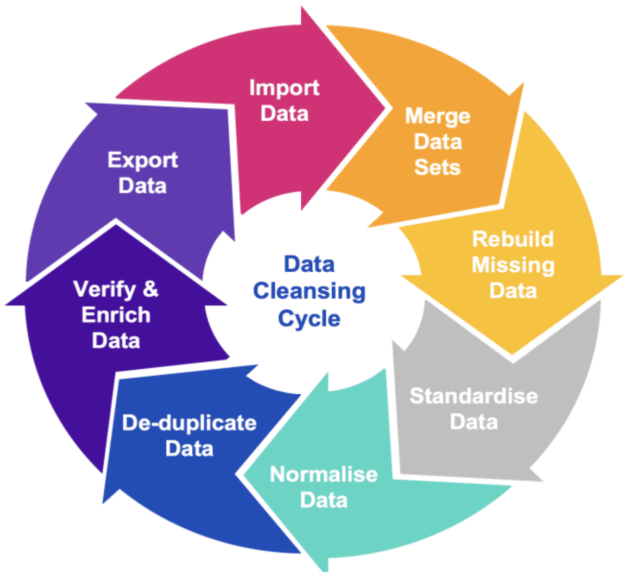


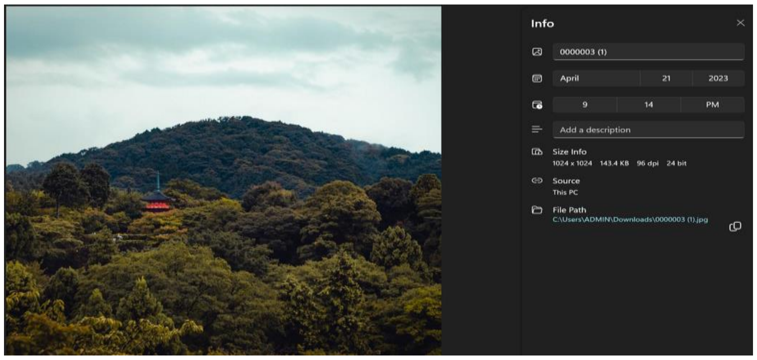
Fig 7-1: The Standard Data Cleaning Cycle

- **Data Collection and Labelling:** Ensuring that images are labeled and categorized correctly is a critical aspect of managing and organizing image data. In supervised machine learning tasks, labeled images serve as the training data. Correct labels are crucial for teaching the model to recognize patterns and make accurate predictions or classifications. For this, we maintain a consistent structure (taxonomy) for the labels.
- **Handling Missing Images**: Missing or corrupted images can lead to data inconsistencies, errors in analysis, and unreliable machine learning models. This is important to ensure Data Completeness, Data Quality. We can do this by File Existence

Check or Checksum Verification. Calculate a checksum (e.g., MD5 or SHA-256) for each image file in your dataset when you create or collect the dataset. Store these checksums in a separate file or database.

- **Resolution standardization:** This is the process of making sure that all images in a collection or dataset have the same resolution by resizing them to a consistent size. Inconsistent resolutions can lead to issues such as misalignment, poor display quality, or difficulties in processing the images, so standardization can help ensure uniformity and improve the overall quality of the visual content. This is done using Python's Pillow library.

Before Resizing:



Info

0000003 (1)

April 21 2023

9 14 PM

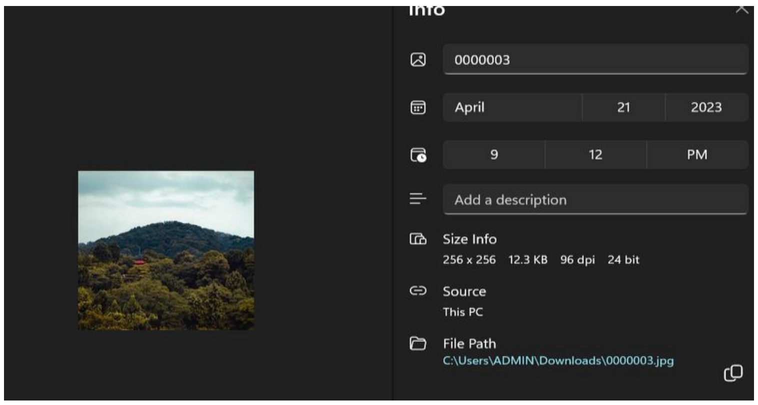
Add a description

Size Info
1024 x 1024 143.4 KB 96 dpi 24 bit

Source
This PC

File Path
C:\Users\ADMIN\Downloads\0000003 (1).jpg

After Resizing:



Info

0000003

April 21 2023

9 12 PM

Add a description

Size Info
256 x 256 12.3 KB 96 dpi 24 bit

Source
This PC

File Path
C:\Users\ADMIN\Downloads\0000003.jpg

- **Increasing the bit detail of image**: Increasing the bit-depth of an image can improve its quality by providing more levels of color or grayscale information per pixel. Bit-depth refers to the number of bits used to represent the color or grayscale value of each pixel in a digital image. A higher bit-depth allows for more nuances in color or intensity, which can result in smoother gradients and reduced quantization artifacts, thereby enhancing the image quality. We do this by using Dithering and Image Editing Software: like Adobe Photoshop or GIMP.

- **Normalization of data:** This process involves adjusting the pixel values of an image so that they fall within a specific range or scale. We will be doing this to make the training of machine learning models more stable and to improve convergence during training. This can be achieved by using Python and the NumPy library.
- **Removing noise:** Image datasets contain various forms of noise, including blur, compression artifacts, and pixelation. Techniques such as median filtering, Gaussian filtering, and edge detection are used to remove noise from the images.

By following the above steps in data preprocessing, we achieve the following:

1. Data Conformity: During the preprocessing phase we made sure that all images in a dataset have the same resolution or format ensuring interoperability and usability of the dataset.

2. Data Quality: The primary goal was to standardize the orientation, aspect ratio, and color space of all photographs. Image processing techniques were used to detect and correct orientation issues, modify aspect ratios using scaling algorithms to preserve proportional relationships, and convert pictures to a uniform color space. The systematic application of these preprocessing steps not only facilitated a consistent and high-quality visual representation of the dataset, but also laid the groundwork for reliable and unbiased analyses or model training, ensuring that subsequent tasks would be performed on a consistent and high-quality image dataset.

3. Data Integrity: Maintaining data integrity is critical for image datasets, and strict data integrity checks were applied during preprocessing to achieve this. Checks for damaged files and the detection of duplicate photos were essential components of this procedure. Detecting and fixing damaged files avoided possible difficulties caused by untrustworthy data, while identifying and removing duplicate photos reduced data redundancy, which might harm model performance. The dataset's integrity was protected by consistently executing these checks, supporting the supply of correct, trustworthy, and streamlined data for later analysis or model training, thereby improving the dataset's overall resilience and efficacy.

7.1.1. Increasing Detail or Depth of the image

The Python Imaging Library (PIL) provides several methods for increasing the detail or depth of an image. One approach is to increase the bit-depth of the image, which involves increasing the number of bits used to represent each pixel value. This can help improve the quality of an image, especially in applications such as medical imaging and satellite imaging, where high levels of detail are important. PIL provides a method called ImageOps.posterize() which can be used to reduce the number of bits used to represent each pixel value. By reducing the number of bits, we can increase the number of levels of detail in the image. Another method is to use the ImageOps.equalize() method, which can be used to equalize the histogram of the image, thereby improving contrast, and increasing the detail in the image. These methods can be used individually or in combination to improve the detail and quality of an image using PIL.

Fig 7-3: Increasing the Depth of the image using PIL

7.1.2. Changing resolution of the image

Changing the resolution of an image is an important task in image processing, as it can help to optimize an image for different applications or devices. In Python, the "Pillow" library provides a simple way to change the resolution of an image. To change the resolution of an image, we can use the Image class and resize() method. We can also specify a resampling filter to use, such as Image.LANCZOS or Image.BICUBIC, which can help to improve the quality of the resized image. After resizing the image, we can save the new image to a file using the save() method. It's important to note that when we change the resolution of an image, we are essentially resampling the image, which can result in a loss of information and image quality. Therefore, it's important to carefully choose the new resolution and resampling filter to ensure that the resized image is suitable for its intended use.

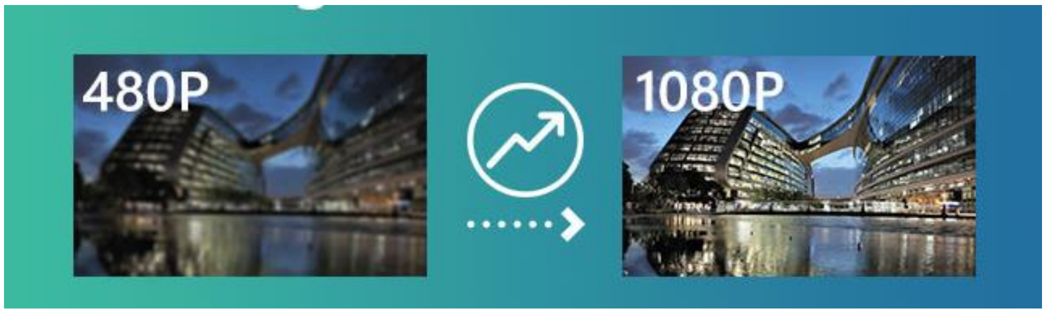


Fig 7-4: Increasing the resolution of the image using PIL

8. ETL (Extraction, Transformation and Load)

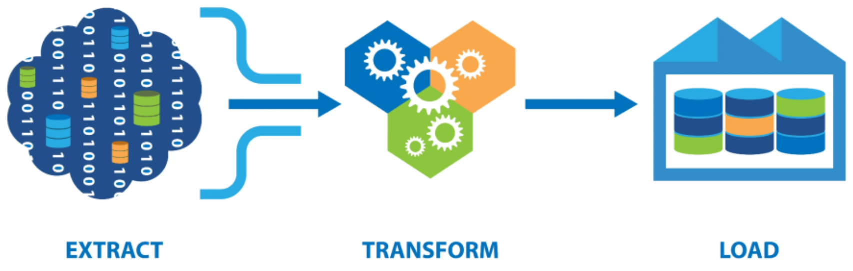


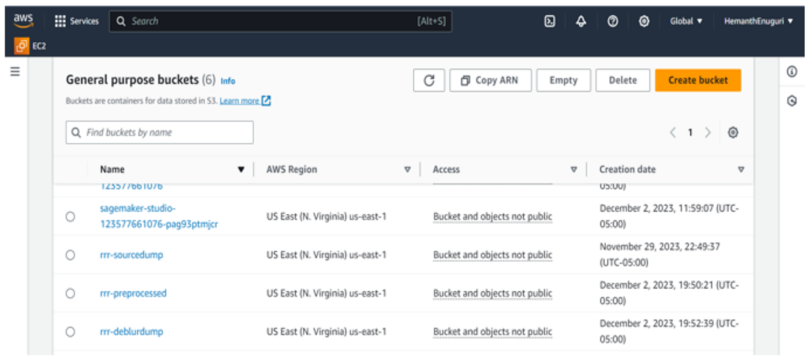
Fig 8-1: ETL-Data Pipeline

8.1. Extraction:

The first step of the ETL process is to extract data from the source systems. Extracting images from Unsplash API is a convenient and efficient way to collect large volumes of image data for our project. Unsplash provide API that allow users to retrieve images programmatically using their access key .

The Unsplash API endpoint we used is https://api.unsplash.com/search/photos. This endpoint is specifically designed for searching photos, and it allows users to retrieve a collection of photos based on a specified search query. The make_request function is responsible for making a GET request to the Unsplash API to search for images.The rate limit is enforced by the @limits decorator, preventing the function from being called more than 40 times within the specified period. If the rate limit is reached, the @sleep_and_retry decorator ensures that the function sleeps for the required duration before retrying. This helps prevent the script from making requests too quickly and exceeding the rate limits imposed by the Unsplash API.

Source dump **Preprocessed dump** **Deblurred Image dump**



General purpose buckets (6) Info

Buckets are containers for data stored in S3. Learn more

Find buckets by name

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

Source dumps are typically raw data images that have not been processed in any way. Preprocessed dumps are source dumps that have been processed to some extent, such as by resizing the images and converting it to a standard format. Deblurred image dumps are dumps of images that have been deblurred, which means that the images have been processed to remove blurriness.

This suggests that the data source is a real-time data source, which means that new data is being added to the source dump bucket on a regular basis. The preprocessed dumps and deblurred image dumps are then generated from the source dump bucket on a regular basis.

This type of data source is often used for machine learning applications. Machine learning models need to be trained on large amounts of data, and real-time data sources can provide a continuous stream of new data to train on.

8.2. Transformation:

The second step in the ETL process is to transform the data extracted from the source systems into a standardized format that is compatible with the target system. Transformations the images must undergo are as follows:

- Check for missing images
- Image Resizing
- Increasing details or depth
- Changing the resolution

Check for missing images:

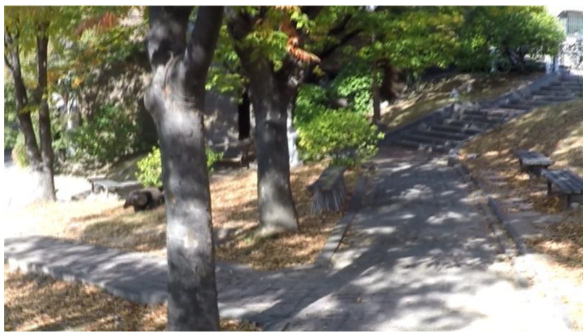
To automate the detection of missing images in a GoPro dataset a serverless computing service like AWS Lambda can be used. The dataset is divided into folders that correspond to sharp and blur photos, each with its own naming system (e.g., sharp_1, blur_1). We develop a systematic procedure using Lambda functions to cross-reference the anticipated file names in each folder and locate missing photos that depart from the predefined pattern. The Lambda function utilizes Python code to extract the list of files from the sharp and blur image folders, subsequently identifying discrepancies between the two sets. Any missing files are then logged or discarded based on user-defined preferences, providing a robust and automated solution for maintaining dataset integrity.

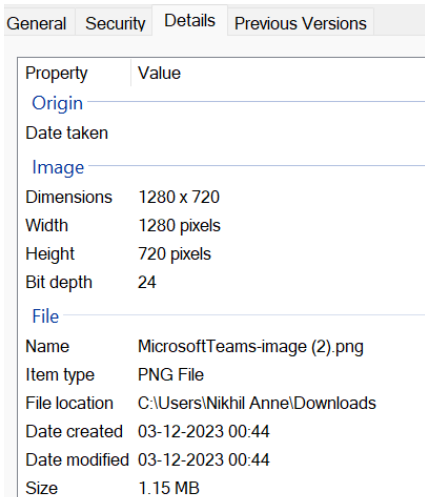
Image Resizing:

Image scaling and pixel normalization emerge as key pretreatment processes in the field of image processing for deblurring applications, enhancing the efficiency and efficacy of future deblurring techniques. The scaling procedure entails changing the dimensions of the input pictures, frequently to adhere to a defined size or to improve computing efficiency.

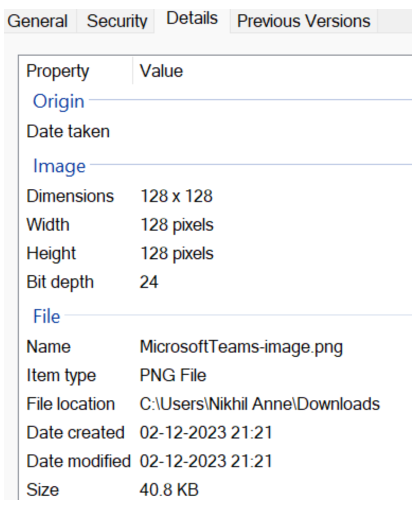
The decision to scale photos to a consistent dimension, in this case 128x128 pixels, was chosen to establish uniformity among datasets of varied resolutions. This common size not only simplifies computing operations, but also allows for the smooth integration and comparison of disparate information. It is especially important when dealing with datasets extracted from sources such as the Unsplash API, where images can have a wide range of dimensions.

At the same time, pixel normalization appears as a critical component in this preprocessing paradigm. The purpose of normalization is to equalize pixel values across images, reducing the influence of pixel intensity distribution variances. In the context of this deblurring attempt, the normalization procedure is critical for ensuring that discrepancies in pixel sizes do not significantly impact the deblurring algorithm.





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



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

Bit depth is a fundamental concept in digital imaging and represents the number of bits used to represent the color of each pixel in an image. The higher the bit depth, the greater the range of colors that can be represented. In our case we have chosen a bit depth of 24.

Changing the resolution of the image:

Changing the resolution of an image is an important task in image processing, as it can help to optimize an image for different applications or devices. In Python, the "Pillow" package provides a simple way to change the resolution of an image.

AWS Lambda:

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. Over 200 AWS services and software as a service (SaaS) apps can trigger Lambda, and you only pay for what you use. AWS Lambda offers a scalable and effective approach to carry out different image processing operations, which helps with the pretreatment of pictures for GAN (Generative Adversarial Network) models. Lambda functions can be integrated with other AWS services to provide a more efficient picture preprocessing pipeline for GAN training, such as S3 for storage or API Gateway for function triggering.

Process Flow of AWS Lambda:

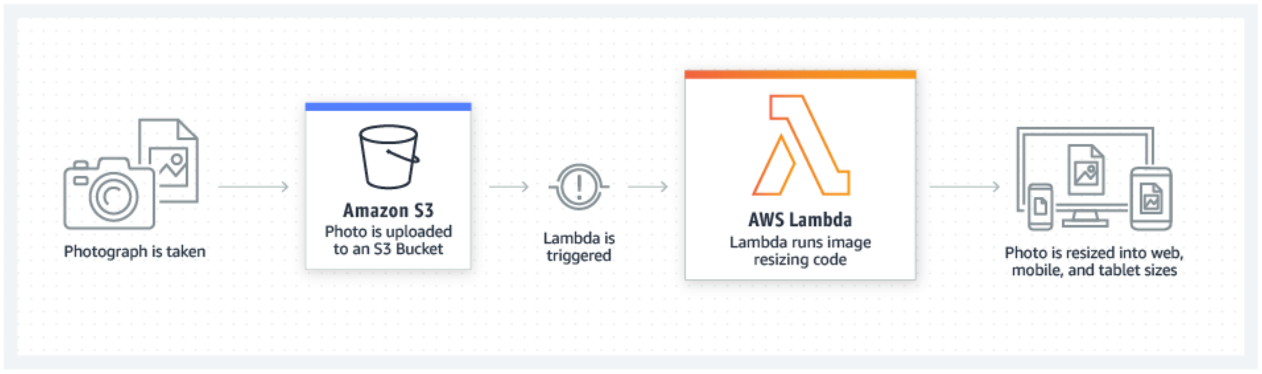
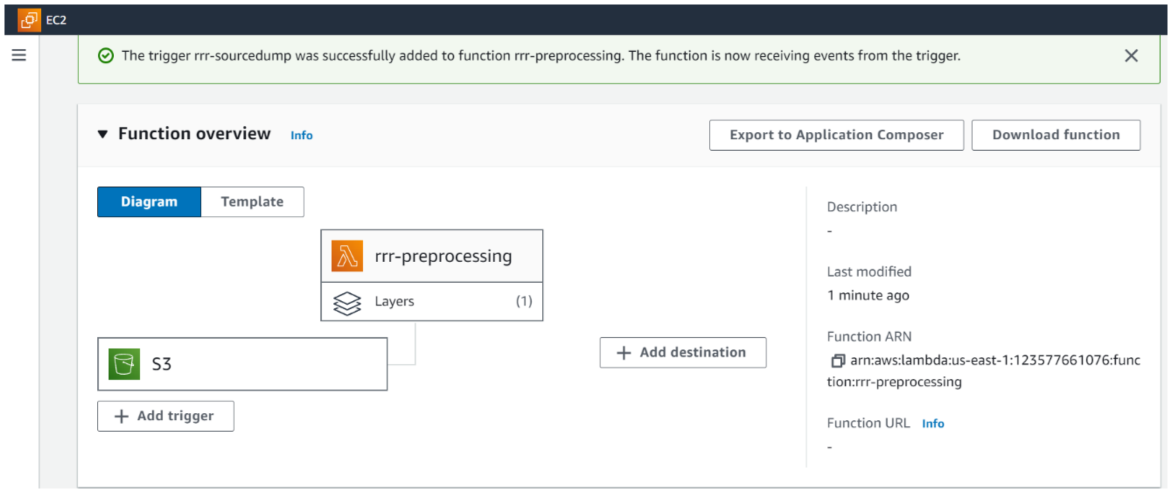

Photograph is taken
Amazon S3
Photo is uploaded to an S3 Bucket
Lambda is triggered
AWS Lambda
Lambda runs image resizing code
Photo is resized into web, mobile, and tablet sizes

Fig 8-2: AWS Lambda Workflow

In the first phase of this AWS workflow, the image dataset is programmatically uploaded to an S3 bucket using the AWS SDK, providing a scalable and reliable storage solution. Subsequently, an AWS Lambda function is crafted utilizing Python programming to automate image preprocessing tasks. This serverless computing approach allows for efficient, event-driven processing without the need for manual intervention. To invoke this Lambda function, a trigger is configured through S3 event notifications in the AWS Management Console. This trigger ensures that the Lambda function is automatically invoked whenever new objects are added to the specified S3 bucket, creating a seamless integration between data arrival and processing.

The Python code within the Lambda function orchestrates the retrieval of photos from the S3 bucket, utilizing the capabilities of third-party libraries such as Pillow or OpenCV to do various image preprocessing tasks such as resizing, cropping, or adding filters. Following the preparation processes, the Lambda function saves the processed photos in the same or a separate S3 bucket, depending on the workflow needs. This adaptable architecture enables customizable storage solutions depending on the application's individual requirements and practicality. To validate the Lambda function's end-to-end operation, extensive testing is performed by uploading a fresh picture to the S3 bucket and checking that the processed image is appropriately saved in the designated place.


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

Description
-
Last modified
1 minute ago
Function ARN
arn:aws:lambda:us-east-1:123577661076:function:rrr-preprocessing
Function URL Info
-

8.3. Load:

The ETL (Extract, Transform, Load) process concludes with the meticulously cleaned and processed picture data being loaded into the destination system, particularly an AWS S3 bucket. The raw data is extracted and placed in the **Source dump** bucket to begin the ETL workflow. Following that, the processed photos are assigned to a newly generated bucket entitled **Resized dump** after undergoing extensive cleaning and transformational operations. This establishes a simplified and well-organized framework for the preprocessed photos. Finally, the GAN (Generative Adversarial Network) model outputs representing deblurred photos arrive at their destination in yet another dedicated bucket entitled **Deblurred Image dump**.

This separation of storage buckets not only allows for systematic data organization at each level, but it also assures the availability of distinct datasets designed for various purposes, enhancing the overall efficiency and accessibility of the data storage and retrieval system.

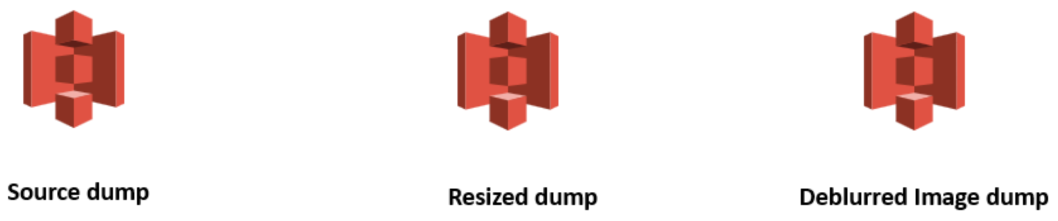
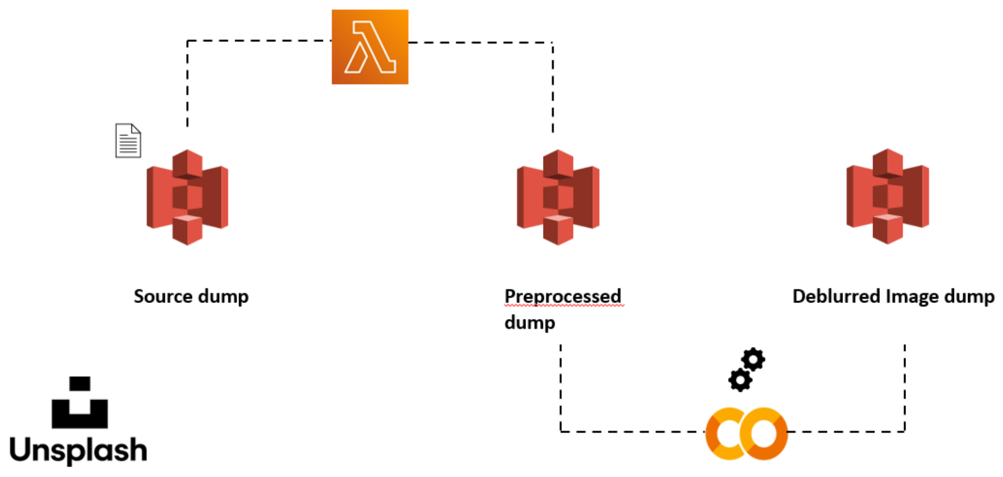


Fig 8-3: AWS S3 Buckets

The below image shows the image deblurring path flow for storing the images in s3 buckets before after preprocessing and model training.



9 Proposed Machine Learning Approach:

Generative models concentrate on capturing the underlying data distribution to produce new samples that mimic the training data, discriminative models aim to discriminate between several classes or categories by learning the decision boundaries in the input space. To put it simply, discriminative models are good at classification tasks—identifying distinctions between classes—while generative models are good at data synthesis—providing a more comprehensive view of the structure of the entire dataset.

Image-to-image translation is the process of creating a new image by carefully altering an existing one. Training such a model often requires datasets containing paired examples. For the neural model to properly train, the problem is that it is costly and difficult to gather these pairs across a variety of contexts.

9.1 Generative Adversarial Networks (GANs):

GANs are a class of Deep Learning based generative models. The first neural network, referred to as the generator, oversees producing fresh data. The discriminator, the second neural network, oversees telling authentic data from fake. The generator oversees producing fresh data. The generator is trained to produce data that is so lifelike the discriminator is unable to distinguish it from actual data. The discriminator is in charge of differentiating between authentic and fake data. It is trained to become better at distinguishing between real and fake data.

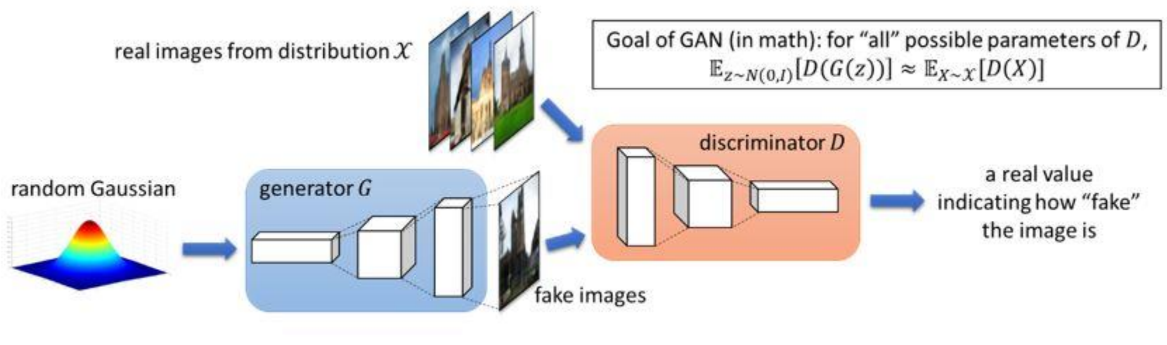

real images from distribution $\mathcal{X}$
Goal of GAN (in math): for "all" possible parameters of D,
$\mathbb{E}_{z\sim N(0,I)}[D(G(z))] \approx \mathbb{E}_{X\sim \mathcal{X}}[D(X)]$
discriminator D
random Gaussian
generator G
a real value indicating how "fake" the image is
fake images

Fig 9-1: The GAN Model (generator and discriminator)

How do GANs work:

Random noise is fed into the generator for GANs to function. The generator generates a fresh data sample by utilizing the noise input. The real data sample from the training dataset and the fresh data sample are supplied to the discriminator. The chance that the data sample is real is produced by the discriminator. To provide data samples that the discriminator is more likely to recognize as real, the generator is upgraded. To improve its ability to discern between authentic and fraudulent data samples, the discriminator is modified. Until the generator can produce data samples that the discriminator is unable to distinguish from actual data samples, this process is repeated.

Conditional GAN:

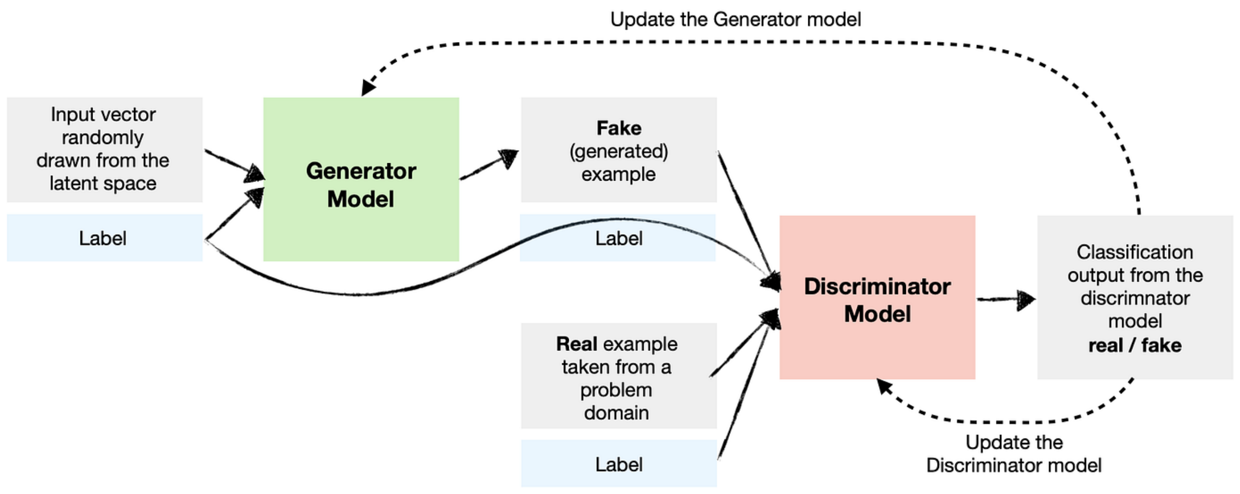

Update the Generator model
Input vector randomly drawn from the latent space
Label
Generator Model
Fake (generated) example
Label
Discriminator Model
Classification output from the discrimnator model real / fake
Real example taken from a problem domain
Label
Update the Discriminator model

Fig 9-2: The Conditional GAN work Cycle

Generative Adversarial Network (GAN) that produce new data conditioned on an input is called a conditional GAN, or cGAN. Anything from a text description to a picture can be used as this input. Tasks like text-to-image generation, super-resolution, and picture translation are frequently performed with cGANs. Because of this conditioning, generation can be more focused and regulated, allowing the model to generate samples according to features or qualities. cGANs are used for a variety of tasks, including style transfer, image-to-image translation, and the production of samples with specific features through conditioning on relevant information throughout the training and generation stages. By using a conditional approach, GANs can produce intended and controllable outputs with greater versatility.

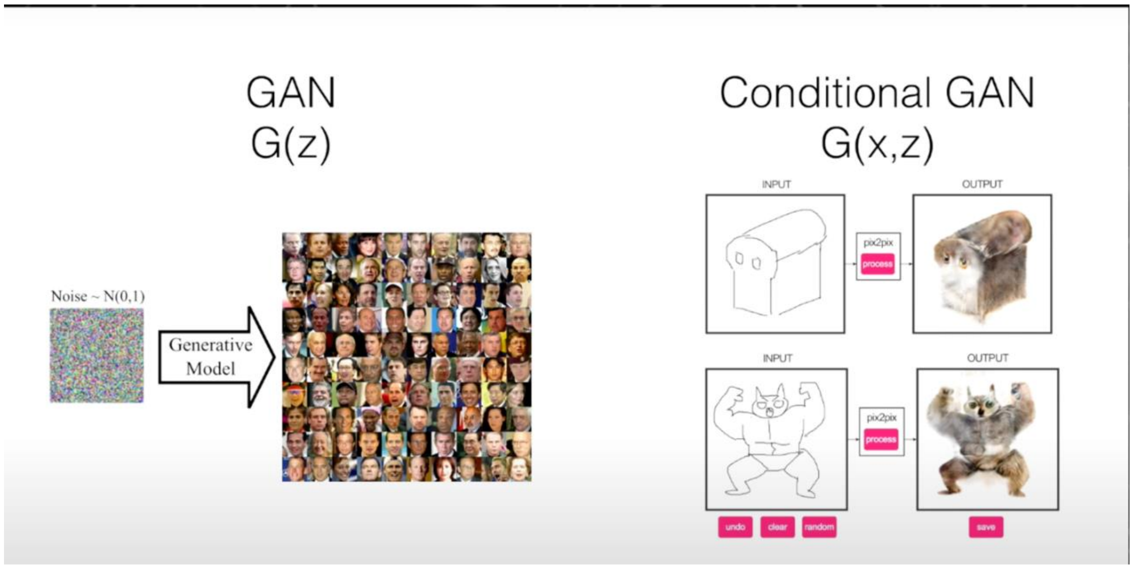


Fig 9-3: The Conditional GAN application

DeblurGAN:

A cGAN-based technique for motion deblurring is called DeblurGAN. It produces a sharp image as the output after receiving a blurry image as input. A dataset containing paired sharp and blurry images is used to train DeblurGAN. With practice, the generator produces sharp pictures that match the blurry input images. The discriminator gains the ability to discriminate between produced and real sharp pictures. Deblurring is handled as an image-to-image translation problem by DeblurGAN. It makes use of a multi-component loss function in conjunction with a conditional GAN architecture. Specifically, they add perceptual loss and use a type of GAN called Wasserstein GAN with a gradient penalty. In comparison to more conventional loss functions like Mean Squared Error (MSE) or Mean

Absolute Error (MAE), this option promotes solutions that closely match real sharp images and can recover finer texture information.

Loss Function:

The performance of the DeblurGAN is expressed mathematically by the loss function. The goal of the DeblurGAN is to produce clear images that are consistent with the blurry input photos by minimizing the loss function. The adversarial loss and the content loss make up the two components of the loss function. The adversarial loss measures the degree to which the DeblurGAN can deceive the discriminator. A neural network called the discriminator makes an attempt to discern between produced and actual sharp images. The generated sharp images' similarity to the original sharp images is measured by the content loss. The adversarial loss and the content loss are added together to form the overall loss function. The hyperparameters of the DeblurGAN define the weights. Since the loss function directs the DeblurGAN during training, it is significant. The goal of the DeblurGAN is to produce clear images that are consistent with the blurry input photos by minimizing the loss.

Loss Function

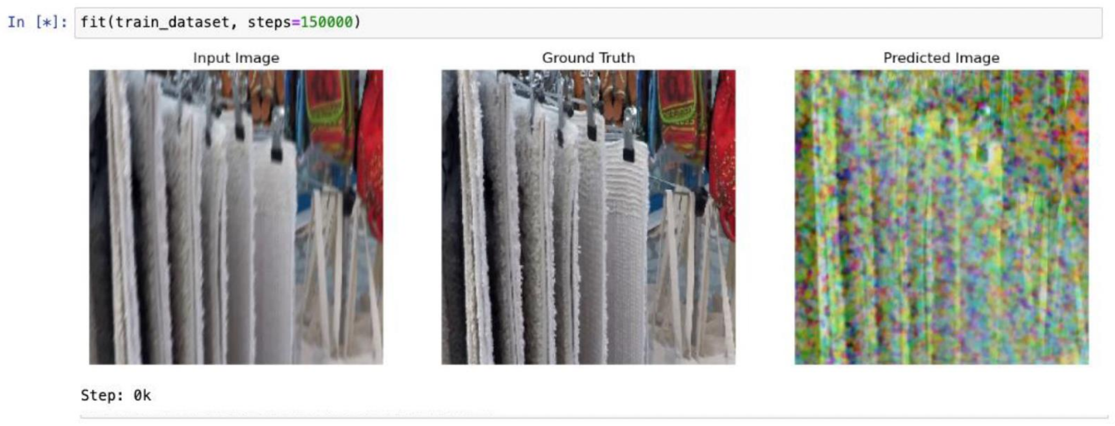
$$\mathcal{L} = \underbrace{\underbrace{\mathcal{L}_{GAN}}_{adv\ loss} \quad + \quad \underbrace{\lambda \cdot \mathcal{L}_X}_{content\ loss}}_{total\ loss}$$

Model Implementation:

Training an image deblurring model using a Generative Adversarial Network (GAN) involves a two-step process: generator training and discriminator training. During the generator training phase, a neural network known as the generator is taught to accept fuzzy pictures as input and output sharp, clear images. In a supervised learning approach, the generator seeks to reduce the disparity between its created images and the ground truth crisp images. Simultaneously, another neural network, the discriminator, is taught to discriminate between created crisp pictures and real sharp images during the discriminator training phase. The discriminator's goal is to accurately identify the source of the pictures, whether produced or genuine, whereas the generator's goal is to trick the discriminator into recognizing its generated images as real.

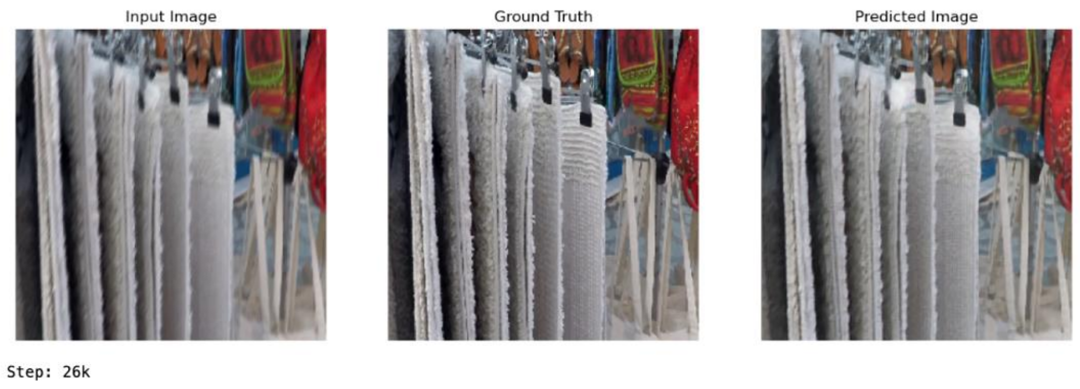
The objective is to strike a balance in which the generator creates high-quality deblurred images that are indistinguishable from genuine crisp photos, while the discriminator becomes progressively difficult to discern between the two. This adversarial training strategy enables the GAN to learn complicated mappings from fuzzy to clear pictures, allowing for successful image deblurring.

The generating network receives blurry picture input image as input. The image that the model is attempting to reconstruct is the ground truth crisp image. It is the input hazy image in sharper form. The image produced by the generator network is known as the predicted image. The expected image is usually noisy and hazy at 0k steps. The projected image will steadily improve in accuracy and sharpness as the model learns.


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

```
Step: 0k
```

Fig 9-4: GAN Predicted image at 0k steps



```
Step: 26k
```

Fig 9-5: GAN Predicted image at 25k steps

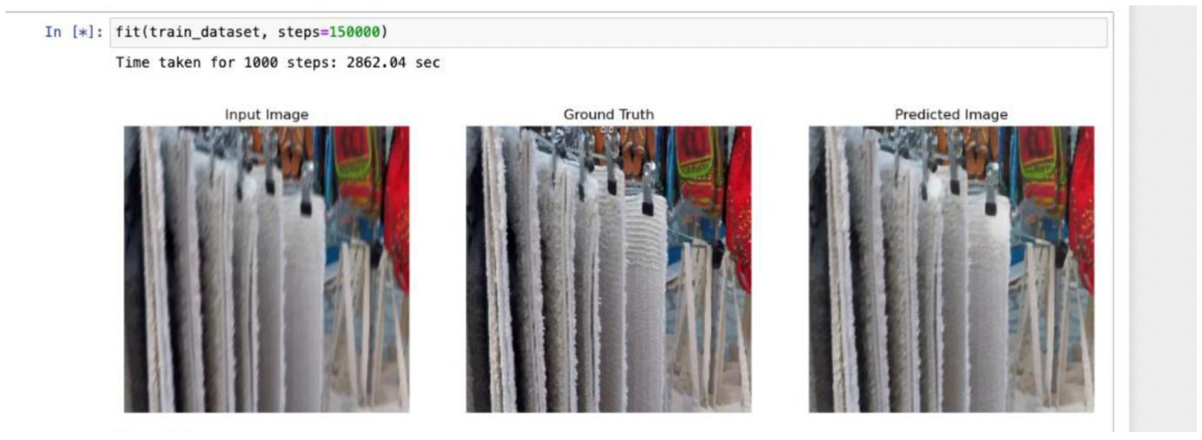
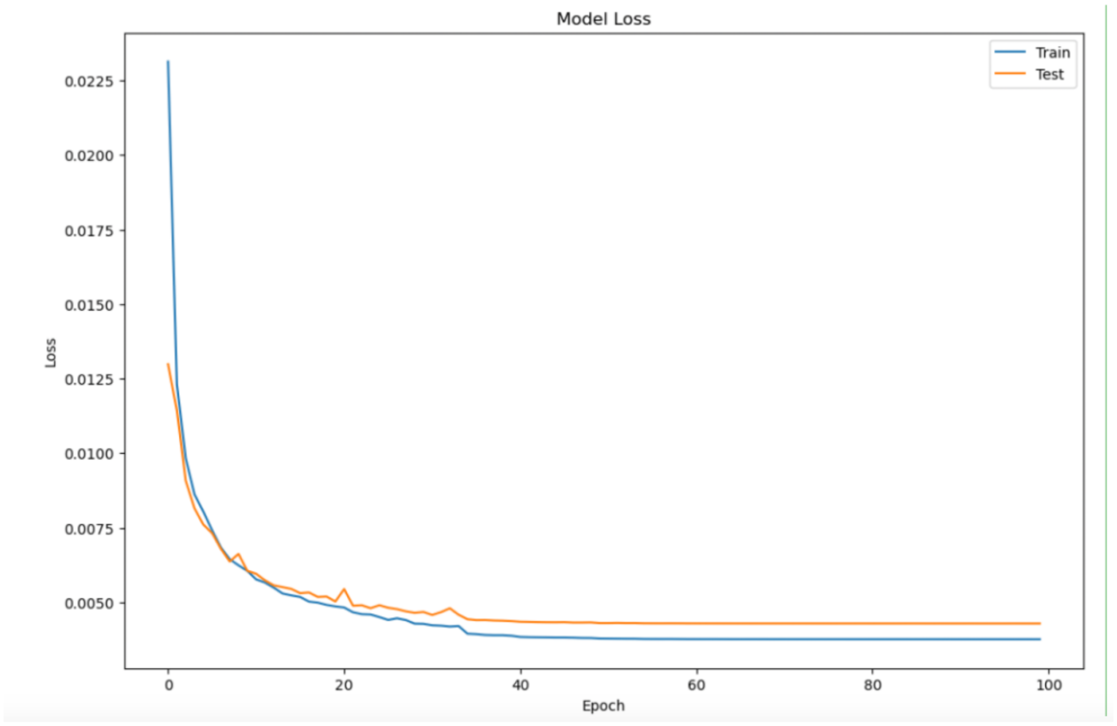

```
In [*]: fit(train_dataset, steps=150000)
Time taken for 1000 steps: 2862.04 sec
```

Fig 9-6: GAN Predicted image at 50k steps

The projected image is far clearer and sharper than the original input image after 50,000 steps. The details are considerably more obvious now that the model has trained to eliminate the image's noise and blur. The anticipated image has undergone the following enhancements after 50,000 steps:

The towel is easily recognized and visible having far more intricate folds and creases, Even the background is more consistent and quieter. After 15,000 steps, the model has significantly improved its ability to deblur the image overall.

Model Loss



The graph displays a DeblurGAN model's test and training loss with time. The model's loss on the training set is shown by the blue line, while its loss on the test set is shown by the red line. The model's performance is indicated by the loss. A reduced loss signifies an improved ability of the model to deblur the photos.

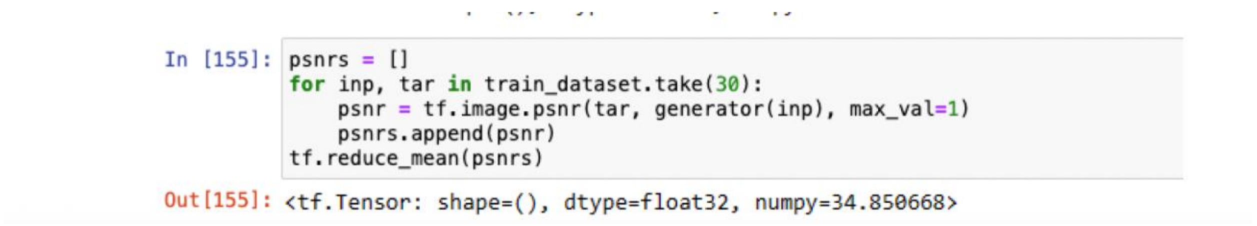
The graph shows that the training loss drops off faster than the test loss. This is since the model is more likely to overfit the training data because it was trained on the training set. When a model learns the training set too well and is unable to generalize to new data, this is known as overfitting. But over time, the test loss is likewise getting smaller. This indicates that without overfitting the training set, the model is improving its ability to deblur the photos.

Additionally, the graph demonstrates that after a predetermined number of training steps, the test and training losses begin to converge. This indicates that the model has hit its peak performance and is no longer picking up fresh data. The graph indicates that the DeblurGAN model is training effectively overall. The model can generalize to new data, and both the test and training losses are declining.

Output metrics:

PSNR:

The Peak Signal-to-Noise Ratio (PSNR) is an important metric in assessing image quality, with higher values generally indicating superior image quality. PSNR values typically fall between 20 and 50 dB, and in some cases, they can exceed this limit. PSNR values are simple to interpret,higher values indicate less distortion and greater fidelity between the original and generated images. A higher PSNR, in essence, represents nearby representation of the original image in the generated output, emphasizing the effectiveness of the image processing or generation technique used.



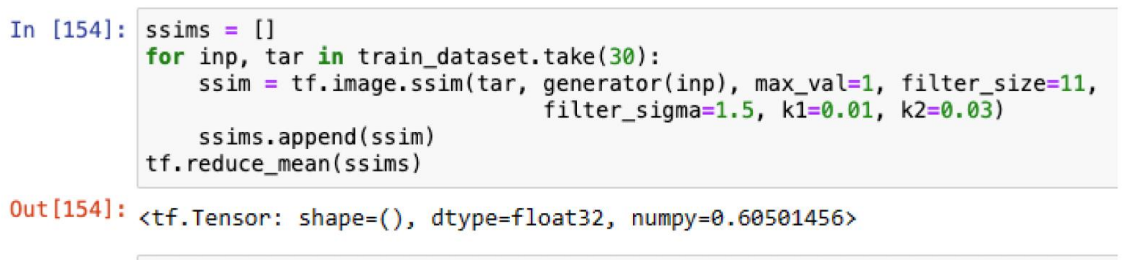
```
In [155]: psnrs = []
          for inp, tar in train_dataset.take(30):
              psnr = tf.image.psnr(tar, generator(inp), max_val=1)
              psnrs.append(psnr)
          tf.reduce_mean(psnrs)

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

The image has been deblurred to a good extent, but there is still some residual noise and blur, as indicated by the PSNR value of 34.85. Real-world images frequently have intricate textures and patterns, making it challenging to obtain PSNR values above 40 db.

SSIM:

The Structural Similarity Index (SSIM) is a useful tool for determining the structural similarity between two images, with a value ranging from -1 to 1. An SSIM value of 1 indicates perfect similarity, indicating an exact match between the images being compared. SSIM values typically range between -1 and 1. In terms of interpretation, values close to 1 indicate that the images have a high degree of similarity or a minimal perceptual difference. Values closer to zero, on the other hand, indicate diminishing similarity, emphasizing perceptual differences. Negative SSIM values indicate significant dissimilarity, indicating significant structural differences between the images under consideration. Overall, SSIM provides a nuanced assessment of structural similarity, providing insights into perceptual quality and image differences.

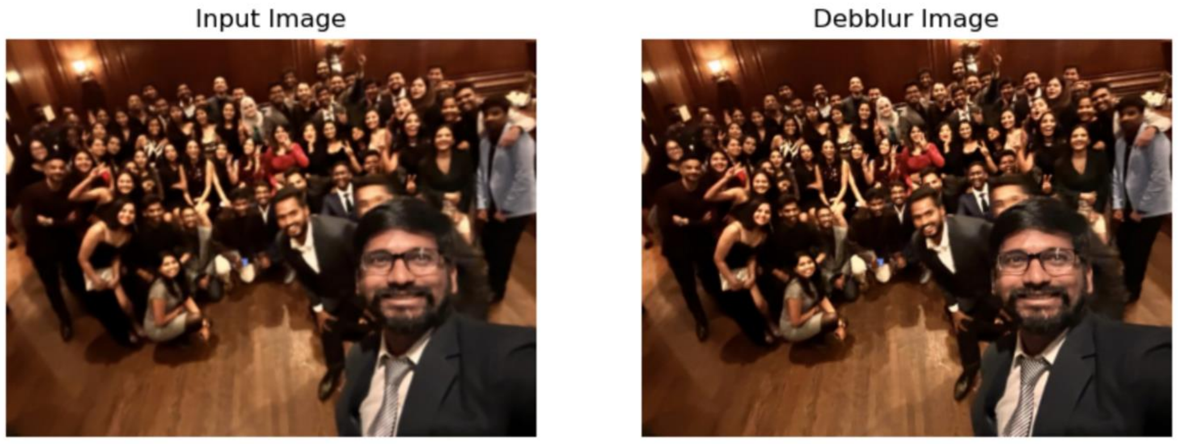


```
In [154]: ssims = []
          for inp, tar in train_dataset.take(30):
              ssim = tf.image.ssim(tar, generator(inp), max_val=1, filter_size=11,
                                   filter_sigma=1.5, k1=0.01, k2=0.03)
              ssims.append(ssim)
          tf.reduce_mean(ssims)

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

The predicted image and the original image have a moderate level of structural similarity, as indicated by the 0.6 SSIM value. This indicates that there is similarity between the original image and the predicted image.

Demonstration of our model:



The showcased image represents the output of our Deblur GAN model. By inputting a blurred image into the trained model, which was trained using approximately 10GB of data, we generate a deblurred version. Notably, in this image, distinct clarity is evident in the facial features of the individuals on the right side, showcasing the effectiveness of the model in restoring details.

Challenges/ Mitigation Strategies:

We have identified the following key limitations while working on our project involving image deblurring using a GAN model:

- **Dataset Limitations Affecting Model Adaptability**: The quality and diversity of the dataset used for training can have a major impact on the GAN model's performance and adaptability. The model's capacity to generalize to other types of blurs may be hampered by limited or biased data.
- **Training Time Restrictions**: GANs, particularly when working with high-resolution images, may necessitate substantial training time. When faced with limited computational resources or severe project timeframes, this might be a substantial barrier.
- **Computational Intensity in GAN Training:** GAN training is computationally intensive, necessitating the use of robust hardware resources like GPUs or TPUs. This can be a limitation for individuals or teams who do not have access to such resources.
- **Limitation of Evaluation measures**: It can be difficult to define useful evaluation measures for analyzing the effectiveness of deblurring models. Traditional measures may not correctly capture the visual quality of deblurred photos.
- **Difficulty in Evaluating Image Flipping**: GANs may learn image translations, rotations, and flipping, which might result in unexpected or unwanted results. These shifts can be difficult to manage and assess.
- **Real-Time Deployment Difficulties**: Moving from a trained model to real-time deployment might be difficult. It is crucial to optimize the model for inference speed without sacrificing deblurring quality.

To mitigate these limitations, some strategies could be:

- **Data Augmentation and Diverse Datasets**: Improve model adaptability by improving dataset quality and variety through augmentation strategies.

- **Transfer Learning**: Reduce training duration and computing effort by leveraging pre-trained models via transfer learning.
- **Alternative Evaluation Metrics**: To better measure deblurring quality, investigate perceptual-based metrics or human evaluation approaches.
- **Regularization Techniques**: Use regularization techniques to control transformations in the model, such as flipping.
- **Model Deployment Optimization:** Improve the trained model's inference speed and resource efficiency for real-time deployment.

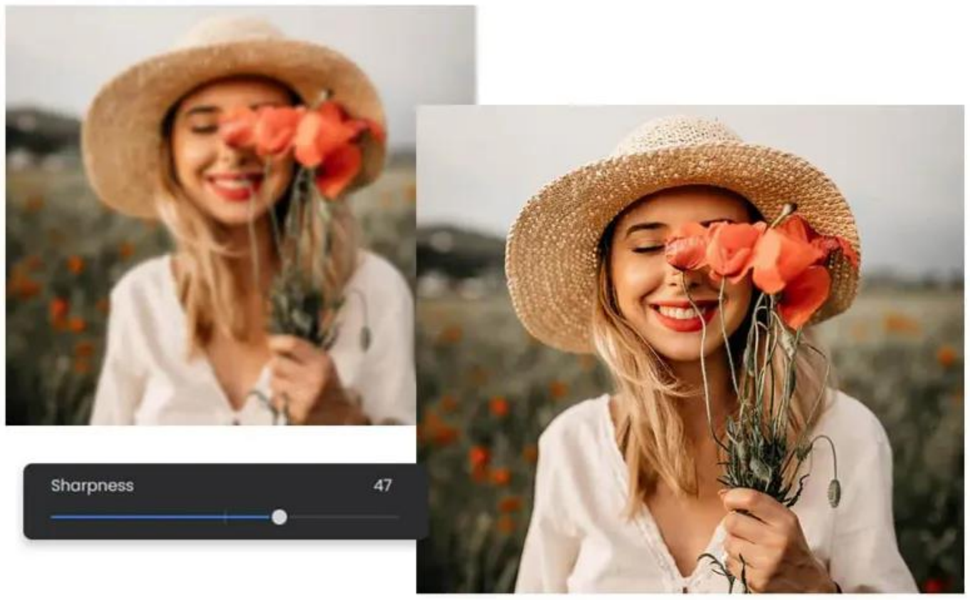
Future Scope

The prospects for the image deblurring project using GAN models present exciting opportunities for advancement and application. Some of them are:

- **Dataset Expansion for Diversity**: Increasing the dataset's versatility by include different types of blurs, different settings, different lighting conditions, and different viewpoints. This expansion guarantees that the model learns to handle a greater range of real-world circumstances, boosting its deblurring effectiveness.
- **Advanced Training Methodologies**: Investigating more advanced training methodologies such as WGAN-GP or other versions can assist stabilize the GAN model's training dynamics. These strategies frequently improve convergence, produce more realistic outputs, and boost the model's overall performance in deblurring tasks.
- **Optimization for Real-Time Inference**: To deploy the deblurring model in real-time or near real-time scenarios, focusing on model optimization becomes crucial. Techniques such as model compression or deploying the model on edge devices

can significantly reduce inference time while maintaining deblurring quality. This enables applications where immediate deblurring is required, like in photography or video processing.

- **User-Friendly Interface or Application Development**: Creating a user-friendly interface or a dedicated application showcasing the deblurring capabilities of the model offers immense practical value. Allowing users to upload their images for deblurring not only demonstrates the model's effectiveness but also serves as a practical tool for individuals seeking to improve image quality.



For instance, the application could have an intuitive interface where users upload blurred images and receive the deblurred versions promptly. Additionally, features like adjusting blur intensity or selecting specific deblurring algorithms (if available) could enhance user engagement and utility. The application's backend would leverage the trained GAN model to perform the deblurring task and ensure seamless user experience. It could be deployed on various platforms, including web, mobile, or even integrated into specific devices like cameras or image editing software.

Overall, these future endeavors aim to enhance the model's capabilities, make it more accessible to users, and extend its practical applications in real-world scenarios.

10 References:

- https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html
- https://seungjunnah.github.io/Datasets/gopro.html
- https://towardsdatascience.com/gan-by-example-using-keras-on-tensorflow-backend-1a6d515a60d0
- https://phillipi.github.io/pix2pix/
- https://www.tensorflow.org/tutorials/load_data/images
- https://chat.openai.com/
- https://medium.com/sicara/keras-generative-adversarial-networks-image-deblurring-45e3ab6977b5
- https://medium.com/@ajithkumarv/deblurv2-gan-with-custom-dataset-6fd138b290bf
- https://www.kaggle.com/code/prabhatpanda3/gandeblur/notebook#Generator

11. My Project Summary (Nikhil Anne)

Our team overcame several challenges to successfully complete a complex yet relevant project on generative AI, even if at times it made us doubt our ability to finish and meet our obligations. I have learned a great lot from my teammates, from independent research, and from hands-on experience thanks to this method. I would like to summarize my experience as follows:

Project selection: We chose this project to gain insights into the Gen AI technology which is now seeing tremendous industry growth. Our goal is to create a novel image deblurring model that outperforms current techniques by utilizing a Generative Adversarial Network (GAN) to achieve higher accuracy and more realistic results. To provide prospective applications in real-world scenarios including forensic analysis, increased object recognition, real estate assessment, VR/AR and gaming, our model aims to obtain better outcomes in restoring fine details from blurry photographs.

Data related: The Gopro dataset and real-world landscape images are the domains we have selected for our CycleGAN implementation. The Unsplash API is utilized to scrape real-world landscape images from the web, while the Gopro dataset is obtained from Kaggle. To extract the data, we used libraries like Boto3 and AWS credentials along with API keys. To train our GAN model, we retrieved the photos in pairs—both sharp and blurry. We completed data collection and labeling to clean the data. In supervised machine learning tasks, precise image labeling and classification are crucial. It is also essential to fix missing photographs to guarantee the accuracy, completeness of the data. For the development of our project, standardization was necessary while gathering photos in a variety of formats and resolutions. We used event-triggered LAMBDA functions to build an automated data pipeline to accomplish this, which required a novel learning process for our group. When these functions are called, pictures are retrieved from our S3 buckets and appropriately resized to 128x128 dimensions. Following this conversion, the photos are kept in a specified S3 bucket using a standard file format.

Using S3 buckets as our main image storage solution was convenient since it allowed us to access our images directly from our GCP Colab notebooks while we were creating the model. Our workflow was eased by this integration, which made it easier to handle and manipulate data while we worked on creating the deblurring model.

GAN model related: The most challenging yet exciting part of our implementation is the DeblurGAN model's training and inference phase. The free-tier machine's RAM turned out to be insufficient, even when processing 1000 128x128 photos. Then we tried the free tier of Google Colab, which included a GPU machine. There was a learning curve in using the GCP machine to access AWS S3 buckets, using picture data to train the DeblurGAN model, and then moving the trained models back to S3 buckets. Apart from the infrastructure barriers, we also invested a lot of time and energy into understanding the discriminator and generator networks described in the study paper and putting them into practice using the Keras package. Understanding the wider range of neural network variations—such as convolution networks and residual networks—that are frequently used in generative AI solutions was made possible by this learning.

We got a 4vCPU, 32GB RAM, and 15GB GPU Apple system for our training needs, and spent more than 12 hours of training our 128x128 model on it. For the data used and the number of training iterations finished, the model's learning and performance fulfilled our expectations. The resulting photos' quality is compromised, and pixels are lost since some features are lost at low resolutions. To do this, we ran 15,000 steps over the course of 12 hours. The number of steps was increased to 50k to obtain a higher-quality image. It was very difficult to observe this model during its 56-hour training period. At last, we succeeded in creating a model that produced photographs of higher quality than the prior model. To assess our model, we have produced the Structural Similarity Index (SSIM) and Peak Signal-to-Noise Ratio (PSNR). An important metric for assessing the quality and effectiveness of our model is the 34 PSNR (Peak Signal-to-Noise Ratio) value that we were able to get. The evaluation of our model yielded a Structural Similarity Index (SSIM) of 0.6, which sheds light on how close the generated deblurred images are to their corresponding ground truth or original images.

Limitations:

- Limitations of the dataset model adaptability factors, some of our data is from Kaggle and the rest is from Unsplash, so it was difficult to identify the right set of data which matches with our project vision.
- Limitation on training duration as our model ran for more than 60 hours just for the training purpose and we experienced several crashes in between as well.
- Limitations of computational intensity in GAN as some of the metrics in GAN are highly complex and we had difficulty in understanding the metrics from time to time.
- Assessing flipped pictures has been an issue for us as our data extracted from Unsplash had street view images which we were not so sure whether to flip them or not.

My Learnings/Takeaways:

- My key lessons learnt from this amazing journey include having enough confidence to convince myself that I can work on one of the most complex projects in generative AI.
- In a GAN architecture, we discovered the intricate training dynamics of the discriminator and generator networks. Understanding their interaction is crucial to achieving excellent deblurring results.
- The significance of utilizing suitable techniques for data loading, preprocessing, and augmentation became evident. These processes ensure the quality and diversity of the data, which is crucial for enhancing model performance.
- Utilizing hardware accelerators such as GPUs and AWS services expedited the training process. These accelerators expedited and improved the efficiency of our model training.
- By using Amazon Lambda services, we were able to automate and streamline this important stage of our operation. PSNR and SSIM metrics were effectively used to assess the deblurring model's performance on training and test sets. These measures provided useful details regarding the sharpness and integrity of the deblurred images.

Overall, I had a wonderful and rich learning experience in working on this project and would like to sincerely thank Dr. Vijay Gandapodi for streamlining an important project for me and my classmates. The advancement of Generative AI in the present world is unreal and I am so glad that I am involved in one of its projects.