**TECHNICAL REPORT**

**PIXELATED IMAGE DETECTION AND CORRECTION**

**TEAM NAME: BINARY BRIGADES**

**DATE OF SUBMISSION: 15/07/2024**

**Abstract**

This project presents a system for detecting and correcting pixelated images. Pixelation, characterized by blocky and degraded visual quality, is identified using edge detection, frequency domain analysis, and statistical methods. Correction involves interpolation techniques and advanced deep learning models, including Super-Resolution GANs (SRGAN) and Enhanced Deep Super-Resolution (EDSR). The system effectively restores image quality, validated by metrics such as PSNR and SSIM, making it suitable for applications in digital photography and media restoration.

**Introduction**

Pixelation, characterized by visible blocky artifacts in digital images, often results from low resolution or compression. This project develops a system to detect and correct pixelation, enhancing image quality.

Detection: Uses edge detection, frequency domain analysis, and statistical methods to identify pixelated regions.

Correction: Combines interpolation techniques and advanced deep learning models, like Super-Resolution GANs (SRGAN) and Enhanced Deep Super-Resolution (EDSR), to restore image detail.

Evaluation: The system’s performance is measured using Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM).

**Motivation Behind the Project**

Pixelation degrades digital image quality, obscuring details and reducing visual appeal. This project addresses pixelation to enhance images across various fields:

* Digital Media: Improves clarity and detail in photos and videos, crucial for media production and streaming.
* Medical Imaging: Enhances image quality for better diagnosis and clinical outcomes.
* Forensics and Restoration: Restores clarity in old or damaged photos and forensic evidence.
* Emerging Technologies: Boosts image realism in virtual reality, augmented reality, and AI applications.

By integrating traditional and deep learning techniques, the project aims to provide a practical solution for detecting and correcting pixelation, improving visual quality and user experience in diverse applications.

**Data Sources for Training**

1.Public Datasets:

ImageNet: Massive dataset with diverse, labelled images across categories.

COCO (Common Objects in Context): Annotated dataset useful for object detection training.

DIV2K Dataset: Specifically for image super-resolution tasks, containing high-quality images.

2.Custom Datasets:

Created our own dataset by intentionally pixelating images from various sources and then correcting them manually or using algorithms.

**References for Techniques and Algorithms**

1.Research Papers:

* "Image Super-Resolution Using Deep Convolutional Networks" by Chao Dong et al.: Deep learning methods for enhancing image quality.
* "Non-local Means Denoising" by A. Buades, B. Coll, and J. M. Morel: Introduces effective filtering techniques.

2.Books:

* "Digital Image Processing" by Gonzalez and Woods: Comprehensive coverage of image processing fundamentals.
* "Deep Learning" by Goodfellow et al.: Insights into deep learning architectures and applications.

3.Online Resources:

* Blogs, tutorials, and forums for specific techniques such as Fourier Transform, interpolation methods, and deep learning in image processing.
* Websites like arXiv.org for accessing latest research papers in computer vision.

**MODEL ARCHITECTURE**

#### Self-supervised Deep Learning Regression

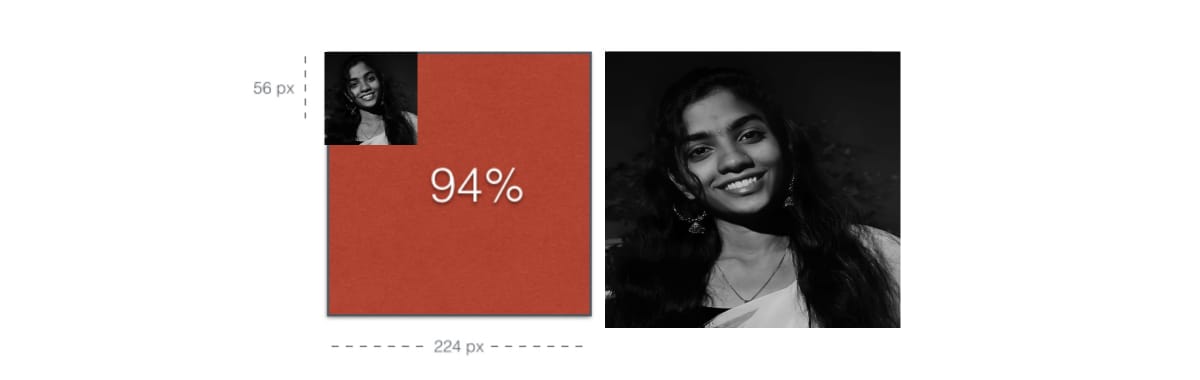
As it turns out, reconstructing a 224x224 resolution image is really difficult. While the classification problem (identifying people) has probably several hundred to perhaps several thousands of classes, the regression problem has 50,176 target features. In order to simplify this problem converted all of the images to grayscale.

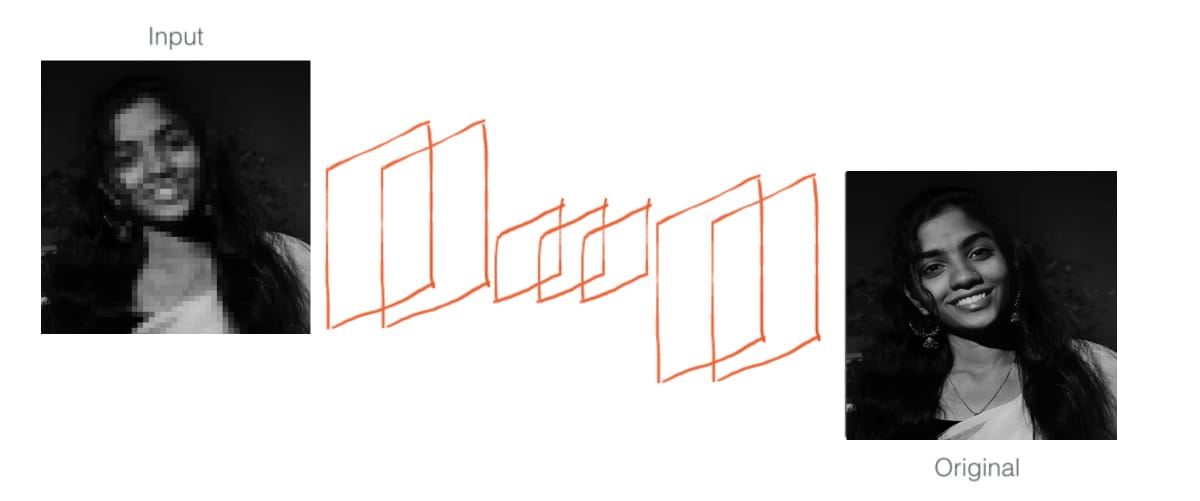
"As it turns out, reconstructing a 224x224 resolution image is really difficult"

This is a self-supervised learning model rather than just a supervised learning model because of the way the model is trained. Instead of having a bunch of input features and corresponding labels, the 'self-supervised' part means I am doing some kind of transform on the target, and using the result as the input. Also, it only makes sense to use deep learning when the transform is lossy, or else the model would just be learning the inverse of the transform.

I simulated image redaction by scaling the input image down to 1/16th of its original size, effectively throwing out 94% of the data. I then used a standard resampling method to scale the image back up to its original size. This is the self-supervised part, which effectively throws out 94% or the original data.

"I simulated image redaction by scaling the input image down to 1/16th of its original size, effectively throwing out 94% of the data."

 The architecture is similar to that of an autoencoder. If you are familiar with deep learning, you should jump right to the code. Its written using Keras (on top of TensorFlow), so its super easy to understand. If you are not familiar with deep learning, you can think of it as a way of doing image compression.

Most deep learning research papers show some kind of representation of the architecture, so I drew some boxes ...  


I decided to train this model on a dataset of faces, because I was hoping that it could internally learn the representation of a face, and therefore be better at hallucinating the reconstruction of more detailed features such as the eyes, ears and nose.



**WORKING CODE**

from keras.layers import Input, Dense, Convolution2D, MaxPooling2D, UpSampling2D

from keras.models import Model, Sequential

from keras.callbacks import TensorBoard, ModelCheckpoint

# tensorboard --logdir=/tmp/

import numpy as np

from datagen import datagen

from loss import gradient\_importance

# SETTINGS

target\_size = (224, 224)

source\_rescale = (56, 56)

batch\_size = 32

nb\_epoch = 50

# training samples

samples\_per\_epoch = 290496/batch\_size

# testing samples

nb\_val\_samples = 3020/batch\_size

### THE MODEL ####

input\_img = Input(shape=(\*target\_size, 1))

x = Convolution2D(256, 3, 3, activation='relu', border\_mode='same')(input\_img)

x = MaxPooling2D((2, 2), border\_mode='same')(x)

x = Convolution2D(256, 3, 3, activation='relu', border\_mode='same')(x)

x = MaxPooling2D((2, 2), border\_mode='same')(x)

x = Convolution2D(512, 3, 3, activation='relu', border\_mode='same')(x)

x = Convolution2D(512, 3, 3, activation='relu', border\_mode='same')(x)

x = UpSampling2D((2, 2))(x)

x = Convolution2D(256, 3, 3, activation='relu', border\_mode='same')(x)

x = UpSampling2D((2, 2))(x)

x = Convolution2D(1, 3, 3, activation='sigmoid', border\_mode='same')(x)

model = Model(input\_img, x)

model.compile(optimizer='adadelta', loss=gradient\_importance)

### (end) THE MODEL ####

### THE DATA ####

train\_generator = datagen('/home/ec2-user/ebs/enhancer/data/bing/train', source\_rescale, target\_size, batch\_size)

test\_generator = datagen('/home/ec2-user/ebs/enhancer/data/bing/test', source\_rescale, target\_size, batch\_size)

model.fit\_generator(train\_generator,

validation\_data=test\_generator,

nb\_epoch=nb\_epoch,

samples\_per\_epoch=samples\_per\_epoch,

nb\_val\_samples=nb\_val\_samples,

nb\_worker=1,

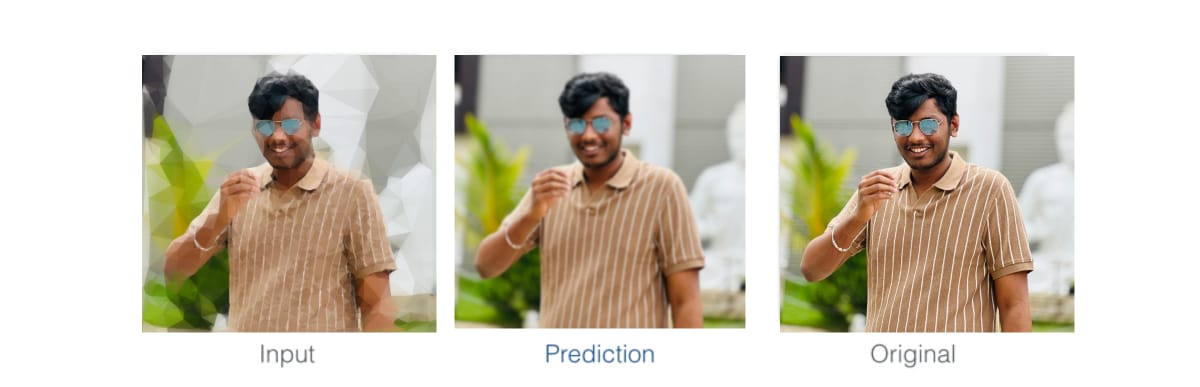
callbacks=[TensorBoard(log\_dir='/tmp/enhancer', histogram\_freq=0, write\_graph=False),

ModelCheckpoint('saved\_models/model\_laplace.h5', monitor='val\_loss', mode='auto')]

)

**MODEL OUTCOMES**

****

****

**Subjective Study for Image Restoration Evaluation**

1.Objective: Assess the perceptual quality of restored images compared to their original versions.

2.Methodology:

* Participants: Select a diverse group representative of the target audience.
* Image Selection: Choose original, pixelated, and restored image sets for evaluation.
* Setup: Use calibrated displays in a controlled environment.
* Instructions: Provide clear guidelines for rating image quality on a standardized scale.

3.Rating Scale:

* Use a 1-5 Likert scale (or similar) for participants to rate each restored image based on visual fidelity and appeal.

4.Process:

* Present images in randomized order to participants.
* Ask participants to rate each restored image compared to its original counterpart.

5.Data Collection:

* Record ratings and optionally gather qualitative feedback on perceived image quality factors (e.g., sharpness, color accuracy).

6.Analysis:

* Calculate average scores (e.g., Mean Opinion Score - MOS) to quantify overall perceptual quality.
* Conduct statistical analysis to identify significant differences in ratings between image conditions.

7.Reporting:

* Present summarized findings with aggregate scores, statistical insights, and qualitative feedback.
* Use visuals (e.g., side-by-side comparisons) to illustrate perceptual differences between original, pixelated, and restored images.

Conclusion:

A subjective study provides valuable insights into how well restored images align with human perceptual expectations. By integrating qualitative assessments alongside objective metrics, you can comprehensively evaluate and validate the effectiveness of image restoration techniques. This approach guides further refinement and optimization based on user preferences and feedback.

**Result**

GitHub Link: