HIGH-RES

(A website to enhance the resolution of low-quality images)

A MINI-PROJECT REPORT

Submitted by

SANTHOSH P (210701234)

SONIYA V (210701254)

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

RAJALAKSHMI ENGINEERING COLLEGE, THANDALAM

MAY - 2024

BONAFIDE CERTIFICATE

Certified that this project **“HIGH-RES”** is the bonafide work of **“SANTHOSH P 210701234”** and **“SONIYA V 210701254” who** carried out the project work under my supervision.

SIGNATURE

Dr. M. RAKESH KUMAR, M.E., Ph.D.,

Assistant Professor,

Computer Science & Engineering

Rajalakshmi Engineering College

(Autonomous)

Thandalam, Chennai-602105

This mini project report is submitted for the viva voce examination to be held on __________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honourable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M. THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honourable principal **Dr. S. N. MURUGESAN** for his able support and guidance

No words of gratitude will suffice for the unquestioning support extended to us by our Head of The Department **Dr. P. KUMAR M.E Ph.D.,** and our Academic Head **Dr. N. DURAIMURUGAN,** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Dr. M. RAKESH KUMAR, M.E Ph.D.,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

Santhosh P (210701234)

Soniya V (210701254)

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	**7**
1	**INTRODUCTION**	**8**
	1.1 INTRODUCTION	8
	1.2 SCOPE OF THE WORK	8
	1.3 PROBLEM STATEMENT	8
	1.4 AIM AND OBJECTIVES OF THE PROJECT	9
2	**SYSTEM SPECIFICATIONS**	**10**
	2.1 HARDWARE SPECIFICATIONS	10
	2.2 SOFTWARE SPECIFICATIONS	10
3	**MODULE DESCRIPTION**	**11**
4	**SYSTEM DESIGN**	**13**
	4.1 SYSTEM ARCHITECTURE DIAGRAM	13
	4.2 NETWORK ARCHITECTURE DIAGRAM	14
5	**TABLE**	**15**
	5.1 PHOTO	**15**
6	**SAMPLE CODING**	16
7	**SCREEN SHOTS**	19
8	**CONCLUSION AND FUTURE ENHANCEMENT**	21
	REFERENCES	22

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	SYSTEM ARCHITECHTURE DIAGRAM	13
4.2	NETWORK ARCHITECTURE DIAGRAM	14

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
5.1	PHOTOS	15

ABSTRACT

The goal of the project is to design a web application that utilizes Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) in order to increase image quality and transform low-resolution images. ESRGAN, which was authored by Wang et al., became a ground-breaking development that could render high-quality images with a similarity to real handmade ones when the input is low-quality. The application was created with a very user interface and interacting to React. Another crucial decision is the framework to use. The combination of JS and Django ensures easy navigation and full functionality. The PyTorch library is used by the image processing component for ESRGAN model implementation, Celery is employed for task management, and OpenCV is utilised for image processing.The developed Web application can be found at https://highres-esrgan.vercel.app/.

The system architecture comprises three primary components: the download part, the process part, and the upload component. Offered is the opportunity to upload images in different image formats, choose the required double-resolution scale factor and download the new images. Quantitative measures like Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index (SSIM), and Perceptual Index (PI) is being used to see how ESRGAN performed against an earlier super resolution model like SRCNN, EDSR, and SRGAN. Benchmark datasets consists of Set5, Set14, BSD100, and Urban100 are applied for performance evaluation.

The research trial has revealed that the ESRGAN is quite successful compared to the other models in terms of visual quality and numerical assessment. However, along with the issues as computational complexity, model size and overfitting, it is also explained how to resolve these problems. This work illustrates deep learning methods well known for image enhancement in a user-friendly way, a versatile tool to improve the quality of images.

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The main aim of the project will be to construct a web program that leverages Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) to improve low-resolution pictures. By utilizing the latest deep learning technology, our application enables users to engage in a user-friendly manner, in order to not only improve the quality but also the visual appeal of their pictures. Digital photography, medical imaging, and satellite imagery analysis are the primary domains that this project wants to advance by means of improving image enhancement processes.

1.2 SCOPE OF THE WORK

Project's scope consists in web application creation using React for a responsive user interface (UI). Meanwhile we use JavaScript for the frontend and Django for the backend. The application will integrate PyTorch, for running the ESRGAN model, which will, in turn, further support efficient image processing. Also, this project will fully rely on a robust SQLite database management system in order to store edited and uploaded images. In the end, this product will be a web-based episode and will allow users to upload, process, and download high-resolution images.

1.3 PROBLEM STATEMENT

Low resolution images are not preferred today due to lack of details and quality that make them unsuitable for this and that purposes. Conventional image enhancement methods, which include but are not limited to interpolation, often do not produce sharp images. This project aims at tackling these challenges by building a web application that applies ESRGAN to low-resolution images so it gives the users the high-quality realistic images which can be further employed in many applications.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim: This project's goal is to produce a web application that will serve as a user-friendly and seamless platform for the users to improve images of low resolution using ESRGAN.

Objectives:

1. To design a user-friendly web app featuring a navigable user interface.
2. The aim of using PyTorch library for implementing the ESRGAN model in image enhancement.
3. To offer a medium where users will be able to upload low resolution images and obtain high quality photos.
4. In order to keep data file well-organized and protected by using SQLite.
5. Implementing a secure regime for data handling and image processing procedures
6. In terms of user interfaces (UI) we will offer real-time feedback on the status of image processing and the estimated time of completion.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	React JS
Back - End	:	Django
Database	:	SQLite

CHAPTER 3

MODULE DESCRIPTION

The architecture and design of the ESRGAN-Web application are based on the idea of creating an easy to use and intuitive platform for the users that lets them enhance the low-resolution images using the latest deep learning techniques. React is a tool that can help in the utilization of the abilities of the React technology. Therefore, the application is made up of front-end applications in JS for the front-end and the back-end applications in Django for the back-end, the application ensures a responsive and interactive user experience while at the same time, it uses PyTorch, Celery and OpenCV for efficient image processing. The developed Web application can be found at https://highres-esrgan.vercel.app/.

3.1. Upload Module

The upload is the first interaction point of users, thus, making the low-resolution images uploading as easy as possible. The users can either upload images from their own devices or provide the URLs of their remote images. The supported image formats include the most popular formats like JPEG, PNG, BMP, and TIFF, the reason is that these formats are used by a large number of image sources and therefore, there is no compatibility issue with them. The system, in its effort to avoid efficiency loss and performance slump, sets a limit of 10 MB for each file to be uploaded. After the success of the upload, images are safely stored in SQLite database, thus, a secure data management system is created that is connected to the Django back-end. Besides, users get a real time preview of the uploaded image which helps them to check the selection and to be sure that they chose the right picture before they decide to do something.

3.2. Process Module

The core of the application is the process module, which allows the users to adjust their image enhancing settings to meet their particular needs. The user interface is an intuitive one, so that users can easily select their own super-resolution scale factor and choose from the options that are available, such as 2x, 4x, or 8x. The case study aims to help users in making reasonable choices through the real-time insights into the estimated processing times depending on the chosen scale factor. The ESRGAN model is built on the PyTorch framework that is the deep learning framework of the server and it is used to execute the process of image enhancement seamlessly on the server . This method is about putting the low-resolution input image to the ESRGAN model that, after, produces the corresponding image that is similar to the given scale factor. In the processing phase, users are given detailed status updates, which gives them a clear idea of the application's work and hence, gives them confidence in the application's functionality. After the augmentation is finished, the improved image is saved together with the original one in the SQLite database, which makes the data secure and available for the next reference.

3.3. Download Module

The download module is the final step of the image enhancement chain, providing the users with a convenient way to get and use the enhanced images. Thanks to a user-friendly interface, users can easily download the processed images to their local devices and thus, they can easily integrate with their own workflows and applications. Moreover, the download component gives the users a complete view of the improved image, along with the information about the resolution and quality of the picture. The component helps the users to do the comparisons and evaluation easily with the help of the interactive comparison feature which allows the images to be compared together at the same time. Through the use of a horizontal or vertical slider, the users can see the enhancements that have been made by ESRGAN and thus, they will get the knowledge about the effects and the usefulness of the image enhancement process.

CHAPTER 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

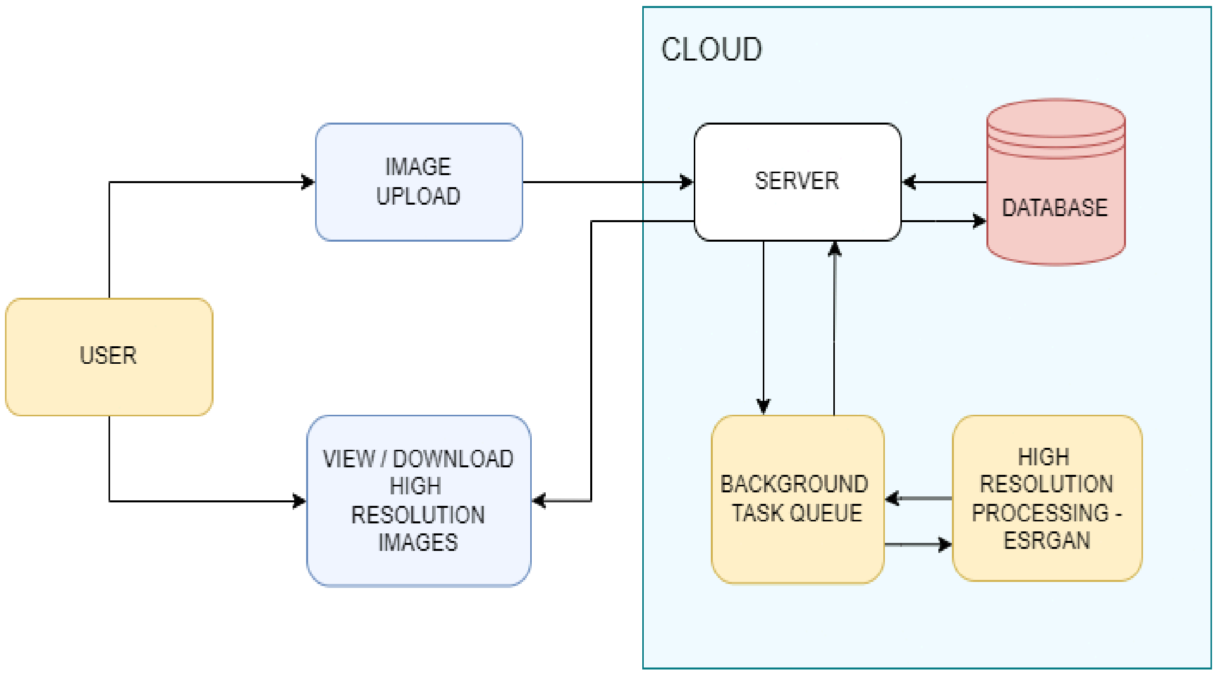

CLOUD
IMAGE UPLOAD
SERVER
DATABASE
USER
VIEW / DOWNLOAD HIGH RESOLUTION IMAGES
BACKGROUND TASK QUEUE
HIGH RESOLUTION PROCESSING - ESRGAN

FIG.4.1 SYSTEM ARCHITECTURE DIAGRAM

This ESRGAN-Web application architecture is based on user-friendliness and high efficiency for image enhancement of low resolution. The front-end application is developed using React at the front-end. js is used here which renders a responsive and interactive user interface. This interface enables users to upload their pictures that are then taken to the back-end server which is handled by the Django. The back-end part uses PyTorch to run the ESRGAN model for image enhancement and processes the pictures to produce high-resolution outcomes. The images augmented with the original ones are saved on SQLite database for the purpose of efficient data management and retrieval. The app also has celery for performing asynchronous tasks therefore ensuring efficient and timely execution of image optimization requests. Such architecture assures that users can simply upload, process, and download improved photos while their system is still impressive, robust and functional.

4.2 NETWORK ARCHITECTURE

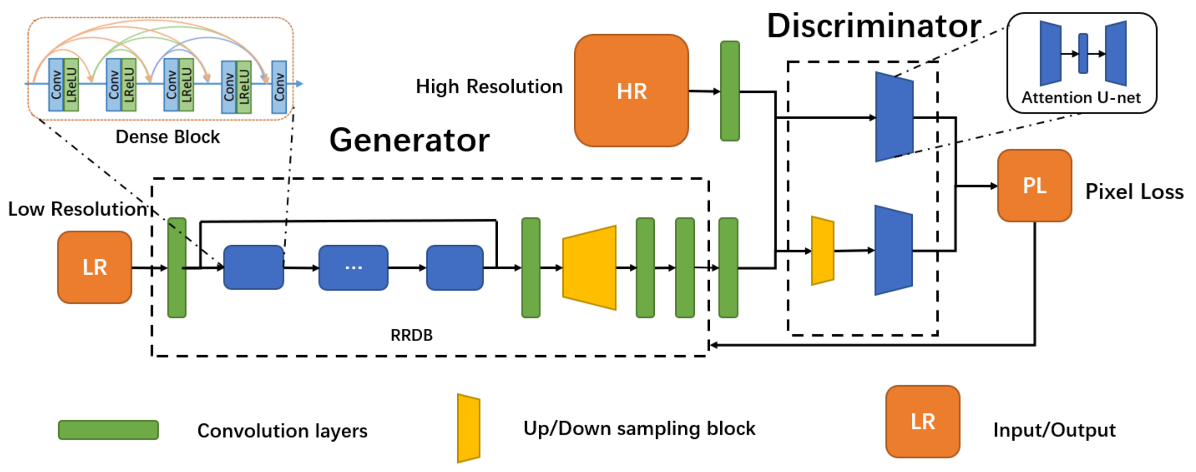


FIG.4.2 NETWORK ARCHITECTURE DIAGRAM

The Fig.4.2 depicts a convolutional neural network architecture for an Enhanced Super-Resolution Generative Adversarial Network (ESRGAN) used for image upscaling. ESRGAN comprises a generator network that takes low-resolution images and creates high-resolution ones through a series of convolutional layers. A discriminator network concurrently analyzes these generated images to determine their authenticity. By continuously evaluating these generated images against real ones, the ESRGAN refines its ability to produce realistic high-resolution images from low-resolution inputs.

CHAPTER 5

TABLES

Table 5.1: Photo table

SL. NO.	ATTRIBUTE	DATATYPE
1	LOW_RES	IMAGE
2	HIGH_RES	IMAGE
3	STATUS	BOOLEAN

CHAPTER 6
SAMPLE CODING

celery.py

```
from __future__ import absolute_import, unicode_literals
import os
from celery import Celery
from django.conf import settings

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'highres_backend.settings')
app = Celery('highres_backend')
app.conf.enable_utc=False
app.conf.update(timezone= 'Asia/Kolkata')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()


@app.task(bind=True)
def debug_task(self):
    print(f'Request: {self.request!r}')
```

models.py

```
from distutils.command import upload
from django.db import models

# Create your models here.
class Photo(models.Model):
    low_res=models.ImageField(upload_to="low_res/")
    high_res=models.ImageField(upload_to="high_res/", null=True, blank=True)
    status=models.BooleanField(default=False)

    def __str__(self):
        return self.low_res.name
```

serializers.py

```
from rest_framework import serializers
from .models import Photo

class PhotoSerializer(serializers.ModelSerializer):
    class Meta:
        model = Photo
        fields = '__all__'
```

views.py

```
from rest_framework.generics import ListCreateAPIView
from .serializers import PhotoSerializer
from .models import Photo
from .tasks import predict_image  # Import your Celery task

class PhotoAPIView(ListCreateAPIView):
    queryset = Photo.objects.all().order_by('-id')
    serializer_class = PhotoSerializer

    def perform_create(self, serializer):
        super().perform_create(serializer)
        data = serializer.data
        predict_image.delay(data)
```

urls.py

```
from django.urls import path
from .views import PhotoAPIView

urlpatterns = [
    path("photo/", PhotoAPIView.as_view())
]
```

tasks.py

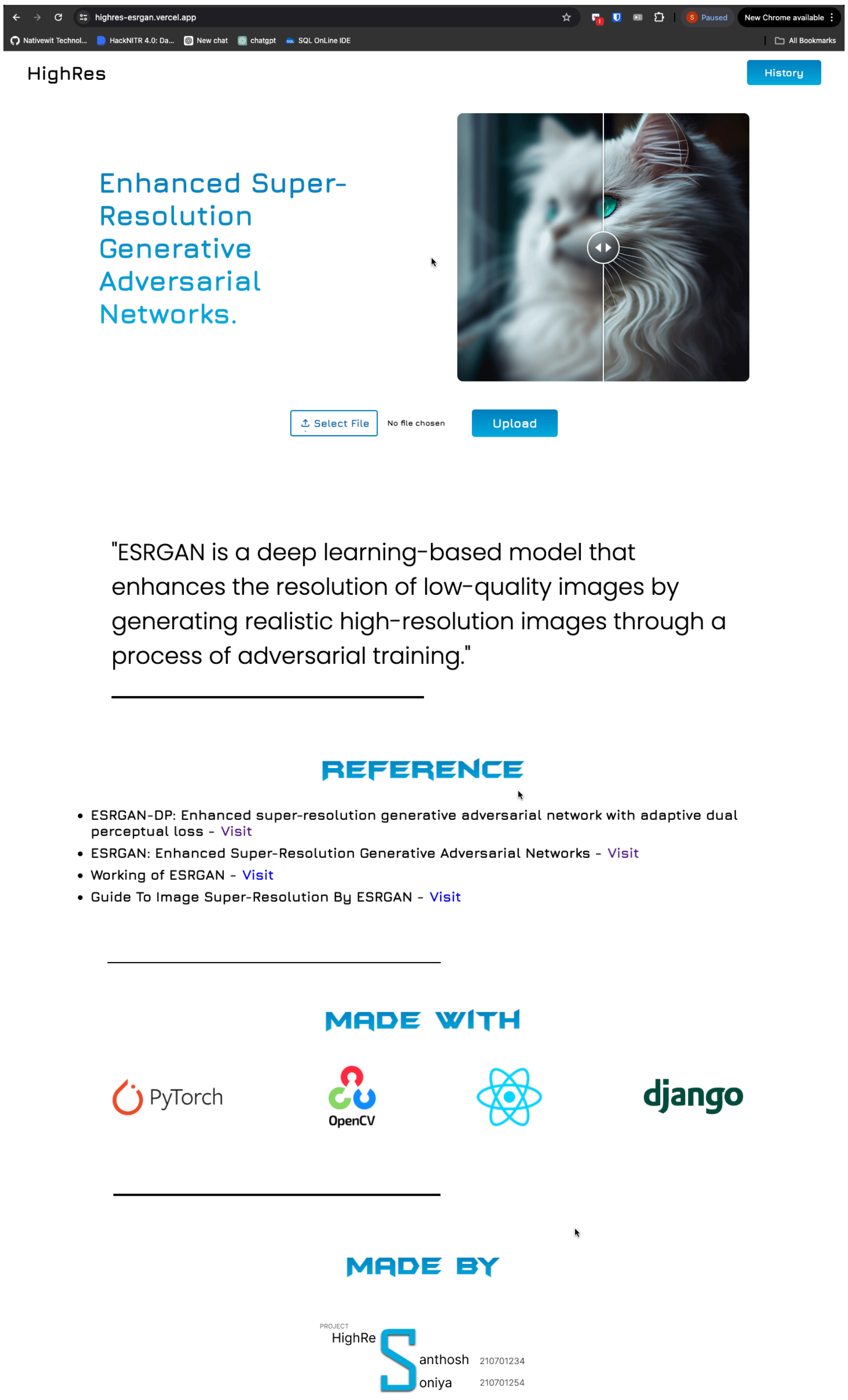
```
import os.path as osp
import cv2
import numpy as np
import torch
from .RRDBNet_arch import RRDBNet
from celery import shared_task
from django.core.files.uploadedfile import InMemoryUploadedFile
import os
import io
from .models import Photo

model_path = os.path.join(os.getcwd(), 'mainapp', 'esrgan_model.pth')
device = torch.device('cpu')
```

```
model = RRDBNet(3, 3, 64, 23, gc=32)
model.load_state_dict(torch.load(model_path), strict=True)
model.eval()
model = model.to(device)

@shared_task(bind=True)
def predict_image(self, data):
    path = os.path.join(os.getcwd(), 'media', 'low_res', data['low_res'].split("/")[-1])
    base = osp.splitext(osp.basename(path))[0]
    img = cv2.imread(path, cv2.IMREAD_COLOR)
    img = img * 1.0 / 255
    img = torch.from_numpy(np.transpose(img[:, :, [2, 1, 0]], (2, 0, 1))).float()
    img_LR = img.unsqueeze(0)
    img_LR = img_LR.to(device)

    with torch.no_grad():
        output = model(img_LR).data.squeeze().float().cpu().clamp_(0, 1).numpy()
    output = np.transpose(output[[2, 1, 0], :, :], (1, 2, 0))
    output = (output * 255.0).round()

    output_image = cv2.imencode('.png', output)[1].tostring()
    file_buffer = io.BytesIO(output_image)
    file = InMemoryUploadedFile(file=file_buffer, field_name=None,
name='{}.png'.format(base), content_type='image/png', size=len(output_image),
charset=None)
    instance = Photo.objects.get(pk=data['id'])
    instance.status=True
    instance.high_res.save('{}.png'.format(base), file, save=True)
    return "Done"
```

The developed Web application can be found at https://highres-esrgan.vercel.app/.

CHAPTER 7

SCREENSHOTS



Enhanced Super-Resolution Generative Adversarial Networks.

"ESRGAN is a deep learning-based model that enhances the resolution of low-quality images by generating realistic high-resolution images through a process of adversarial training."

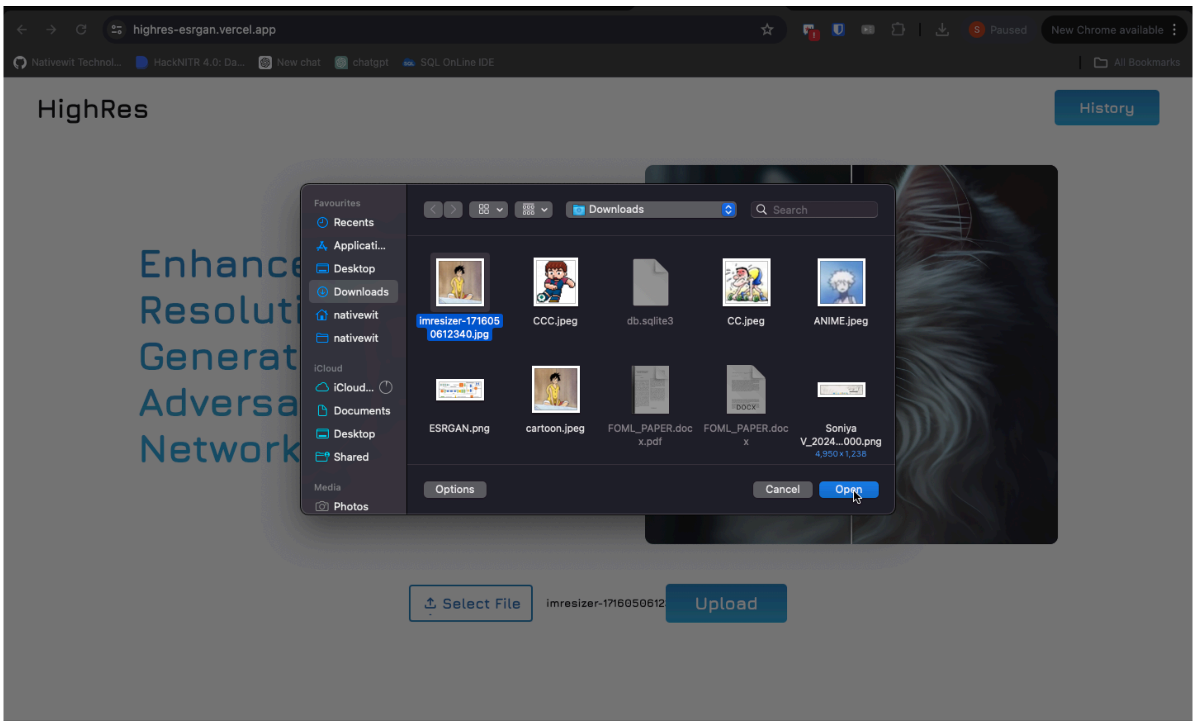
REFERENCE

- ESRGAN-DP: Enhanced super-resolution generative adversarial network with adaptive dual perceptual loss - Visit
- ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks - Visit
- Working of ESRGAN - Visit
- Guide To Image Super-Resolution By ESRGAN - Visit

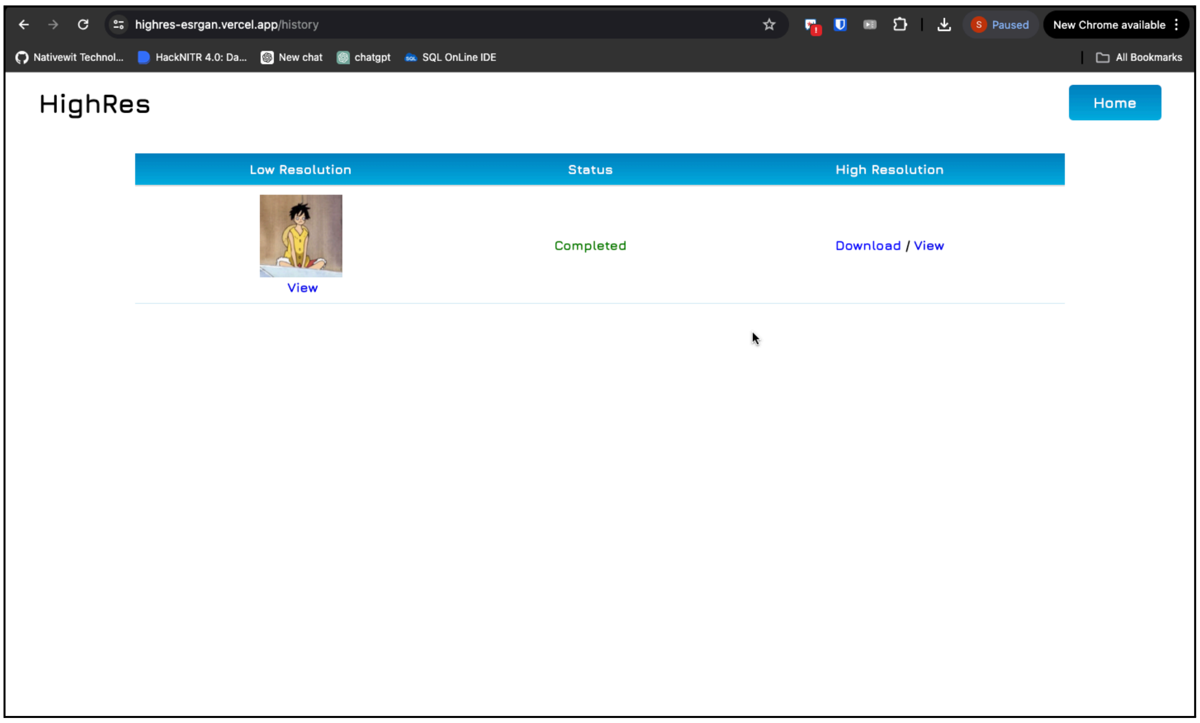
MADE WITH

MADE BY

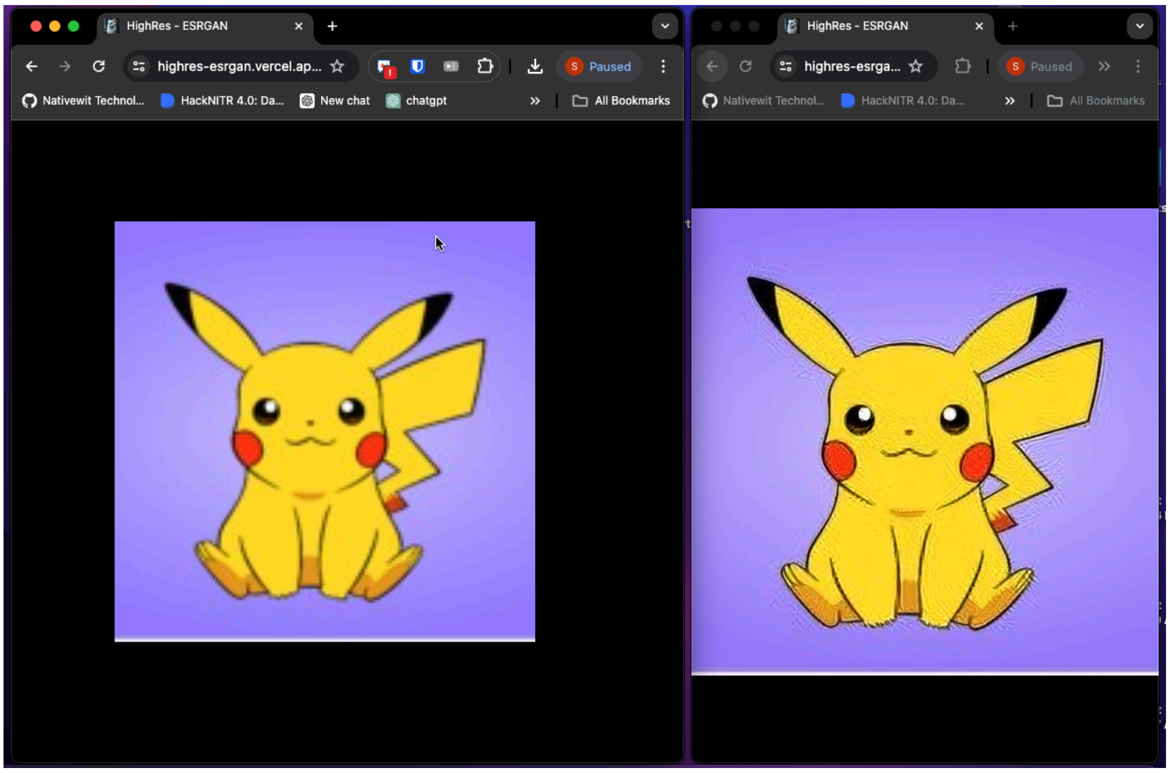
The Homepage of the application provides us with the information about the model used and the stack used to build the application.



On Clicking on the "Select File" option, a low-resolution image can be selected from the local system and then on clicking the "Upload" Button, the image will be uploaded into the model. Later, the resultant high-resolution image can be viewed and downloaded from the "History" option.



Low Resolution	Status	High Resolution
View	Completed	Download / View



In the provided comparison, the image on the left represents the low-resolution input, characterized by pixelation and lack of fine details, while the image on the right showcases the high-resolution output generated by the ESRGAN-Web application. The transformation is evident, with the high-resolution image exhibiting enhanced clarity, sharpness, and intricate details, demonstrating the effectiveness of the ESRGAN model in upscaling low-resolution images to produce visually stunning results. The developed Web application can be found at https://highres-esrgan.vercel.app/.

CHAPTER 8
CONCLUSION AND FUTURE ENHANCEMENT

To sum up, our web application uses the ESRGAN to make the low-resolution images better and thus provides the users with a simple and easy to use tool to improve the quality and clarity of the images. Through the use of cutting-edge deep learning technologies, our application gives users a smooth image enhancing experience in different content categories and resolutions. Nonetheless, it is necessary to recognize the shortcomings of our web application and to pinpoint the areas that need to be improved in the future. First, the ESRGAN may not perform well in some cases, for instance, text, logos, faces, or objects with complex shapes, and this may lead to artifacts or suboptimal enhancements. The problems of these challenges can be solved by the continuous research and development of the super-resolution models to make them more robust and versatile.

Besides, the scalability and resource constraints of our server infrastructure may be the limitation of the application in handling a large volume of concurrent requests or in processing high-resolution images efficiently. The scaling of the server infrastructure and the optimization of the resource utilization are vital steps towards the enhancement of the application's responsiveness and the performance under heavy workloads. Moreover, the issues related to data security and privacy should be considered. Our application keeps the uploaded and enhanced images in a SQLite database, but there is a danger of data leakage or misuse, especially for the sensitive or personal images. Installing strong security measures and privacy protections is the key to protect the user data and to make the user trust and trust the application.

REFERENCES

1. https://arxiv.org/pdf/1809.00219
2. https://arxiv.org/pdf/1609.04802
3. https://ieeexplore.ieee.org/document/8099582
4. https://ieeexplore.ieee.org/document/1163711
5. https://ieeexplore.ieee.org/document/1163711
6. https://www.sciencedirect.com/science/article/abs/pii/104996529190045L
7. https://ieeexplore.ieee.org/document/7115171
8. https://arxiv.org/pdf/1406.2661
9. https://docs.celeryproject.org/en/stable/
10. https://opencv.org/
11. https://pytorch.org/
12. https://www.sqlite.org/index.html