A PROJECT REPORT ON

“DenoisePro”- Video Noise Removal

Title: DenoisePro	**Report Date**: 17 October 2024
Author: Manish Meghana D S	**USN** NNM23MC073 NNM23MC074
Name of the Department Department of MCA	
Type of Report and Period Covered AIML Project report, October 2024	

Abstract

This project implements a Flask-based web application designed for video processing, specifically focusing on noise analysis and denoising. Users can upload video files, which the application analyzes for two types of noise: Gaussian noise and salt-and-pepper noise. Utilizing OpenCV, the application calculates noise levels and provides feedback on the quality of the uploaded video.

The denoising functionality leverages advanced image processing techniques to reduce noise while preserving the original audio track, which is extracted using FFmpeg. The processed video is then combined with the retained audio and saved for user download. The application features an intuitive web interface for easy file uploads and outputs results in real time.

This project not only serves as a practical tool for enhancing video quality but also demonstrates the integration of computer vision, audio processing, and web development technologies. It is suitable for various applications, including content creation, video editing, and noise reduction in multimedia presentations.

Keywords

Gaussian Noise, Salt-and-Pepper Noise, Audio Extraction, Denoising, Flask, OpenCV, FFmpeg

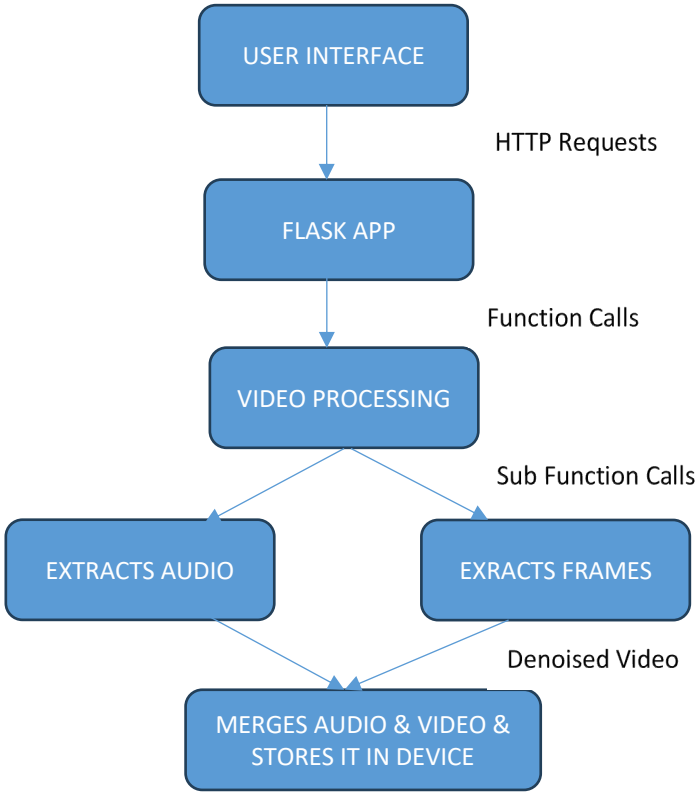
Introduction

The **Video Noise Analysis and Denoising Application** leverages the power of Flask, OpenCV, and FFmpeg to provide a user-friendly platform for analyzing and denoising

videos. This project addresses the common issue of noise in video recordings, which can detract from the visual quality and overall viewing experience. By employing advanced techniques for noise analysis and denoising, this application aims to enhance video quality while preserving the original audio.

Noise in videos can manifest in various forms, including **Gaussian noise** and **salt-and-pepper noise**, which can be particularly problematic in low-light conditions or when the video is compressed. The application allows users to upload video files, analyze them for noise levels, and subsequently apply denoising techniques to improve the overall quality.

Architecture diagram



Methodology

1. System Setup:

Configure the development environment by installing necessary libraries, including Flask for the web framework, OpenCV for video processing, and FFmpeg for audio handling. This step ensures that all components are ready for integration.

2. File Upload and Validation:

Create a user-friendly web interface that allows users to upload video files. Implement server-side checks to validate the file formats (e.g., MP4, AVI) to ensure only compatible videos are processed, enhancing application reliability.

3. Noise Analysis:

Utilize OpenCV to analyze the uploaded video for noise levels. This involves calculating Gaussian noise through variance analysis and identifying salt-and-pepper noise by counting extreme pixel values. The results provide users with insight into the video quality.

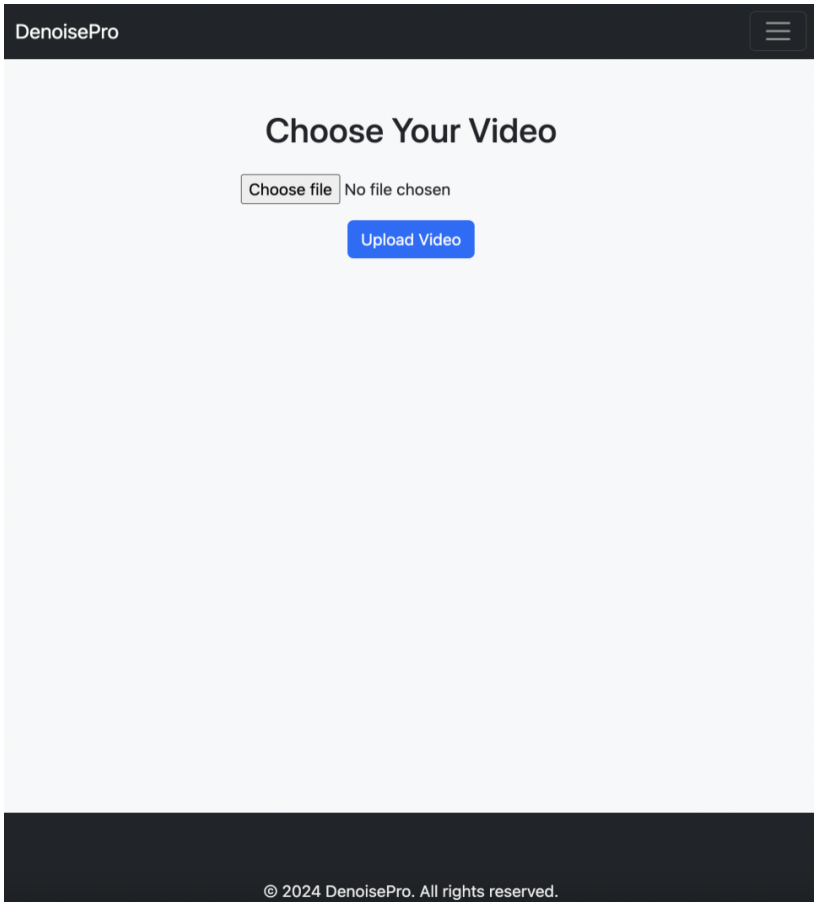
4. Denoising and Audio Extraction:

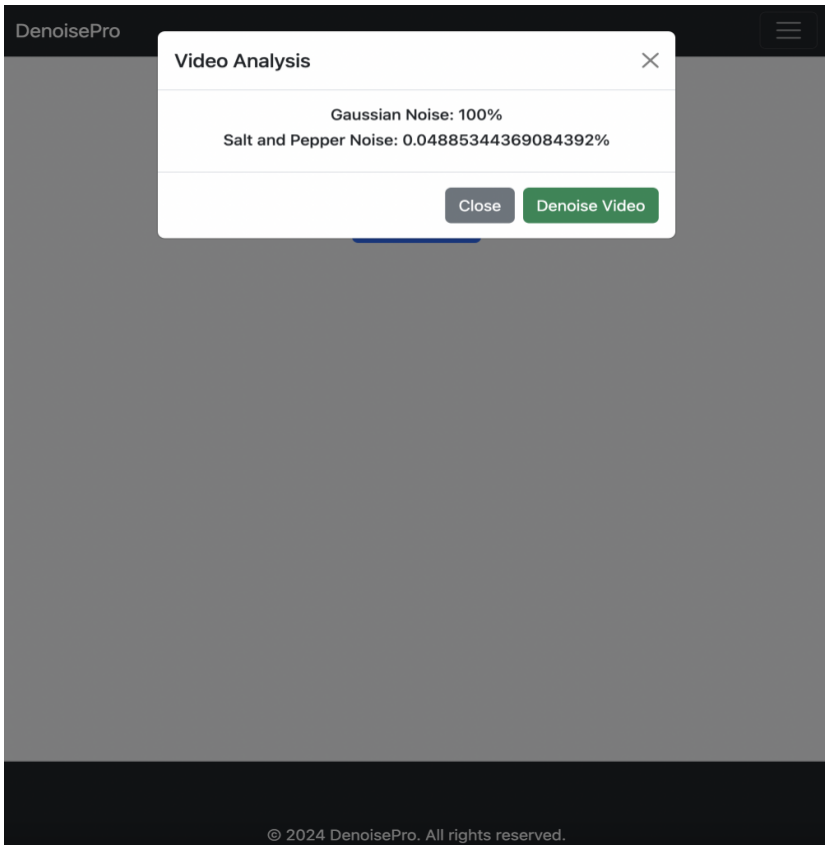
Use FFmpeg to extract the audio track from the uploaded video, ensuring high-quality sound preservation. Simultaneously, apply a denoising algorithm to the video frames to reduce noise without compromising visual quality, saving the processed frames temporarily.

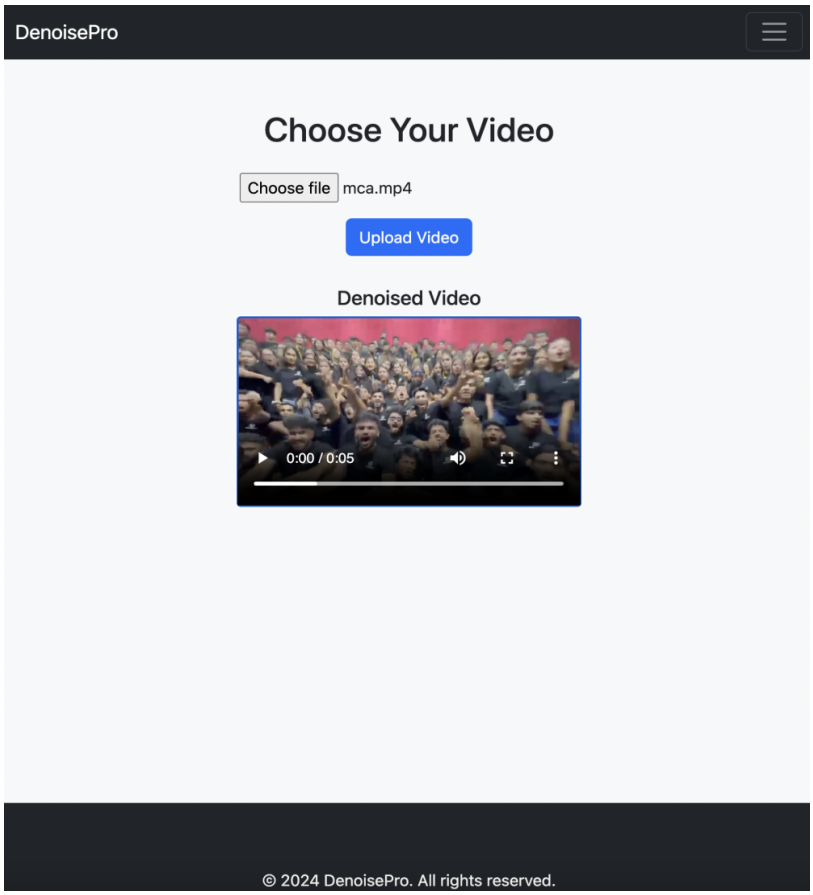
5. Output Generation and Deployment:

Combine the denoised video with the extracted audio using FFmpeg to create the final output video. Once all functionalities are tested and validated, deploy the application on a web server, making it accessible to users for processing their videos online.

Results


DenoisePro
Choose Your Video
Choose file
No file chosen
Upload Video
© 2024 DenoisePro. All rights reserved.


DenoisePro
Video Analysis
Gaussian Noise: 100%
Salt and Pepper Noise: 0.04885344369084392%
Close
Denoise Video
© 2024 DenoisePro. All rights reserved.



Future Scope

Enhanced Denoising Algorithms: Explore and implement more advanced denoising techniques, such as deep learning-based methods.

User Interface Improvements: Develop a more interactive and visually appealing user interface.

Performance Optimization: Optimize processing speed and resource usage, especially for larger video files.

Additional Features: Consider adding features like video format conversion and batch processing capabilities.

Mobile Application Development: Developing a mobile version of the application could broaden its user base. Users could analyze and denoise videos directly from their smartphones

Conclusion

In conclusion, this project successfully demonstrates the integration of video processing and noise analysis using Flask, OpenCV, and FFmpeg. The application allows users to upload videos, analyze them for noise levels, and apply denoising techniques while preserving the original audio. This dual functionality not only enhances the quality of video content but also provides valuable insights into the noise characteristics of the uploaded videos.

The analysis of noise levels, including Gaussian and salt-and-pepper noise, offers users a quantitative measure of video quality, which can be crucial for various applications, such as content creation, video editing, and broadcasting. The denoising process, utilizing advanced algorithms, ensures that the output video maintains its visual integrity while reducing unwanted noise, making it suitable for professional use.

Code

app.py :

```
from flask import Flask, request, jsonify, render_template
import os
os.environ["TF_ENABLE_ONEDNN_OPTS"] = "0"
import cv2
import numpy as np
import subprocess

app = Flask(__name__)

# Set upload and denoised folder paths
UPLOAD_FOLDER = 'static/uploads'
DENNOISED_FOLDER = 'static/denoised'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
os.makedirs(DENNOISED_FOLDER, exist_ok=True)

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def extract_audio(video_path, audio_output_path):
    """Extract audio from video using ffmpeg"""
    command = f"ffmpeg -i \"{video_path}\" -q:a 0 -map a \"{audio_output_path}\""
    subprocess.run(command, shell=True, check=True
```

```
def analyze_video(video_path, frame_sample_rate=10):
    """Analyze video for noise levels"""
    cap = cv2.VideoCapture(video_path)
    gaussian_noise_total = 0
    salt_pepper_noise_total = 0
    total_frames = 0

    if not cap.isOpened():
        raise Exception("Could not open video for analysis.")

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Only analyze every nth frame
        if total_frames % frame_sample_rate == 0:
            total_frames += 1

            # Gaussian noise analysis (calculating variance)
            gaussian_noise_total += np.var(frame)

            # Salt-and-pepper noise analysis (counting noisy pixels)
            noisy_pixels = np.sum((frame == 0) | (frame == 255))  # Count white (255) and black (0) pixels
            salt_pepper_noise_total += noisy_pixels / (frame.size) * 100  # Percentage of noisy pixels

        total_frames += 1  # Increment total frames for sampled frames

    cap.release()

    # Calculate average noise levels
    if total_frames > 0:
        average_gaussian_noise = gaussian_noise_total / total_frames
        average_salt_pepper_noise = salt_pepper_noise_total / total_frames
    else:
        average_gaussian_noise = 0
        average_salt_pepper_noise = 0

    # Convert Gaussian noise from variance to a percentage (0-100 scale)
    gaussian_noise_percentage = min(average_gaussian_noise / 255 * 100, 100)  # Assuming pixel values are from 0-255

    return {
        'gaussian_noise': gaussian_noise_percentage,
        'salt_pepper_noise': average_salt_pepper_noise,
    }
```

```
def denoise_video(video_path):
    """Denoise video while preserving audio"""
    original_file_name = os.path.splitext(os.path.basename(video_path))[0]
    output_path = os.path.join(DENNOISED_FOLDER, f"{original_file_name}_denoised_temp.mp4")
    audio_path = os.path.join(DENNOISED_FOLDER, 'audio.mp3')  # Temp audio file path

    try:
        # Extract audio from the original video
        extract_audio(video_path, audio_path)

        # Open the video using OpenCV
        cap = cv2.VideoCapture(video_path)
        if not cap.isOpened():
            raise Exception("Could not open video.")

        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out = cv2.VideoWriter(output_path, fourcc, 30.0, (int(cap.get(3)), int(cap.get(4))))

        frame_count = 0  # Counter to sample every nth frame

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            # Apply denoising only on every nth frame
            if frame_count % 5 == 0:  # Change this number for more or less frequent processing
                denoised_frame = cv2.fastNlMeansDenoisingColored(frame, None, 10, 10, 7, 21)
                out.write(denoised_frame)

            frame_count += 1

        cap.release()
        out.release()

        # Combine the denoised video with the original audio
        final_output_path = os.path.join(DENNOISED_FOLDER, f"{original_file_name}_denoised.mp4")
        command = f"ffmpeg -i \"{output_path}\" -i \"{audio_path}\" -c:v copy -c:a aac \"{final_output_path}\""
        subprocess.run(command, shell=True, check=True)

        # Remove the temporary audio file and temp video file
        os.remove(audio_path)
        os.remove(output_path)
```

```
            return final_output_path
        except Exception as e:
            print(f"Error: {e}")
            return None

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'video' not in request.files:
        return jsonify({'error': 'No file part'}), 400
    file = request.files['video']
    if file.filename == '':
        return jsonify({'error': 'No selected file'}), 400
    if file:
        video_path = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(video_path)

        # Analyze video
        noise_data = analyze_video(video_path)
        return jsonify(noise_data), 200

@app.route('/denoise', methods=['POST'])
def denoise():
    video_name = request.form['video_name']
    video_path = os.path.join(app.config['UPLOAD_FOLDER'], video_name)

    denoised_video_path = denoise_video(video_path)
    if denoised_video_path is None:
        return jsonify({'error': 'Failed to denoise video.'}), 500

    return jsonify(original=video_path, denoised=denoised_video_path), 200

if __name__ == '__main__':
    app.run(debug=True)
```

index.html

```
!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>DenoisePro v2</title>
   <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
   <style>
      html,body {
         background-color: #f8f9fa;
         height:100%;
      }
      .container {
         margin-top: 50px;
      }
      .video-container {
         display: none; /* Initially hidden */
         justify-content: space-around;
         margin-top: 30px;
      }
      video {
         width: 100%;
         max-width: 480px;
      }
      .progress {
         height: 20px;
      }
   </style>
</head>
<body>
   <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container-fluid">
         <a class="navbar-brand" href="#">DenoisePro</a>
         <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-
bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-
label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
         </button>
      </div>
   </nav>

   <div class="container d-flex justify-content-center align-items-center vh-50">
      <div class="text-center">
         <h1 class="mb-4">Choose Your Video</h1>
         <form id="upload-form" enctype="multipart/form-data">
            <div class="mb-3 d-flex justify-content-center">
```

```
            <input type="file" name="video" id="video" accept="video/*" required class="text-center">
          </div>
          <button type="submit" class="btn btn-primary">Upload Video</button>
        </form>
      </div>
    </div>

    <!-- Analysis Modal -->
    <div class="modal fade" id="analysisModal" tabindex="-1" aria-labelledby="analysisModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="analysisModalLabel">Video Analysis</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
          <div class="modal-body text-center">
            <div id="loadingMessage">Loading... Please wait.</div>
            <div id="noiseResults" style="display: none;">
              <h6>Gaussian Noise: <span id="gaussianNoise"></span>%</h6>
              <h6>Salt and Pepper Noise: <span id="saltPepperNoise"></span>%</h6>
            </div>
          </div>
          <div class="modal-footer">
            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
            <button id="denoise-button" type="button" class="btn btn-success" style="display: none;">Denoise Video</button>
          </div>
        </div>
      </div>
    </div>

    <!-- Loading Modal -->
    <div class="modal fade" id="loadingModal" tabindex="-1" aria-labelledby="loadingModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-body text-center">
            <h5>Processing...</h5>
            <div class="spinner-border" role="status">
              <span class="visually-hidden">Loading...</span>
            </div>
            <div class="progress mt-3">
              <div id="progress-bar" class="progress-bar" role="progressbar" style="width: 0%;" aria-valuenow="0" aria-valuemin="0" aria-valuemax="100">0%</div>
```

```
            </div>
          </div>
        </div>
      </div>
    </div>

    <div class="container video-container text-center" id="video-container">
      <h5>Denoised Video</h5>
      <div class="row justify-content-center">
        <div class="col-12 col-md-6 mb-3">
          <video id="originalVideo" controls class="w-100"></video>
        </div>
      </div>
    </div>


    <footer class="footer bg-dark text-white text-center" style="position: fixed; bottom:
0; width: 100%; margin-bottom: 0; padding: 1rem;">
      <div class="container">
        <span>&copy; 2024 DenoisePro. All rights reserved.</span>
      </div>
    </footer>

    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></
script>
    <script>
      $(document).ready(function() {
        let lastUploadedVideo = '';

        $('#upload-form').on('submit', function(e) {
          e.preventDefault();
          let formData = new FormData(this);
          $.ajax({
            type: 'POST',
            url: '/upload',
            data: formData,
            contentType: false,
            processData: false,
            success: function(response) {
              $('#loadingMessage').hide();
              $('#gaussianNoise').text(response.gaussian_noise);
              $('#saltPepperNoise').text(response.salt_pepper_noise);
              $('#noiseResults').show();
              $('#denoise-button').show();

              // Save the last uploaded video name
              lastUploadedVideo = $('#video').val().split('\\').pop(); // Extracts the
file name
```

```
                $('#analysisModal').modal('show');
            },
            error: function() {
                alert('Error in uploading video.');
            }
        });
    });

    $('#denoise-button').on('click', function() {
        $('#analysisModal').modal('hide');
        $('#loadingModal').modal('show');

        // Simulate processing video and update progress
        let progress = 0;
        const interval = setInterval(() => {
            if (progress < 100) {
                progress++;
                $('#progress-bar').css('width', progress + '%').attr('aria-valuenow',
progress);
                $('#progress-bar').text(progress + '%');
            } else {
                clearInterval(interval);
                $.ajax({
                    type: 'POST',
                    url: '/denoise',
                    data: { video_name: lastUploadedVideo }, // Send the last uploaded
video name
                    success: function(data) {
                        $('#loadingModal').modal('hide');
                        $('#originalVideo').attr('src', data.original);
                        $('#denoisedVideo').attr('src', data.denoised);
                        $('#video-container').show(); // Show the video container after
processing
                    },
                    error: function() {
                        alert('Error in denoising video.');
                    }
                });
            }
        }, 3200); // Update every 1.2 seconds (2 minutes total to reach 100%)
    });
});
</script>
</body>
</html>
```