Video Upscaling

Guide:
Prof. Satish Mulleti

- *Ashwin Prajapati (24m1064)*
- *Joel Anto Paul (210070037)*
- *Mahendra Pratap Singh (24m1226)*
- *Nirmal S (20d070057)*
- *Saad Bin Ayaz (24m1067)*

Problem Statement

- Real-time video streaming demands high bandwidth for smooth, high-resolution playback.
- Users in bandwidth-limited regions face a trade-off:
  - Lower video quality
  - Frequent interruptions and buffering
- This leads to a poor viewing experience.
- **Proposed solution**:
  - Stream video at lower quality in real-time
  - Perform dynamic upscaling on the user's device
  - Achieve high-definition playback with reduced bandwidth usage

Original image Upscaled

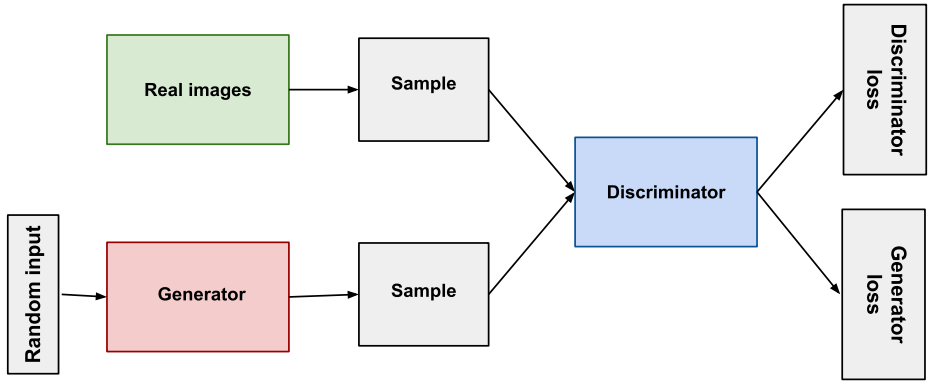
Image Reference:
https://www.google.com/url?sa=i&url=https%3A%2F%2Ffilmora.wondershare.com%2Fvideo-editing%2Fai-upscaling-video.html&psig=AOvVaw0Rk7Iv40JxCPxD9HGc0JFa&ust=1744816105973000&source=images&cd=vfe&opi=89978449&ved=0CBQQjRxqFwoTCMDujaCp2owDFQAAAAAdAAAAABAJ

Our Approach:

- To address the problem, we explored deep learning-based video upscaling techniques.
- Two primary models were implemented and evaluated:
- **GAN-based Model:**
  - Utilizes a generator-discriminator setup
  - Focuses on enhancing perceptual quality and generating realistic high-resolution frames
- **Transformer-based Model:**
  - Leverages self-attention mechanisms for capturing temporal dependencies
  - Excels in maintaining consistency across video frames and fine details

GAN-based Approach

- **What is a GAN?**
- A **Generative Adversarial Network (GAN)** consists of two neural networks:
  - **Generator**: Tries to produce realistic high-resolution images
  - **Discriminator**: Tries to distinguish between real high-res images and generated ones
  - They are trained in a competitive setup, improving each other over time
- **Why Use GANs for Upscaling?**
  - Capable of generating **fine-grained textures and details**
  - Outperforms traditional interpolation methods in perceptual quality
  - Particularly effective for **super-resolution tasks**



SRGAN

- **Why SRGAN?**
  - Designed specifically for **photo-realistic single image super-resolution**.
  - Capable of **4× upscaling** with realistic texture generation.
- **Architecture Highlights:**
  - **Generator**: Deep ResNet with 16 residual blocks, skip connections, and sub-pixel convolutions for upsampling.
  - **Discriminator**: CNN with increasing filter depth, classifies images as real or generated
- **Loss Function (Perceptual Loss):**
  - Combines:
    - **Content Loss** (based on VGG16 feature maps, not pixel-wise MSE)
    - **Adversarial Loss** (to encourage realism)
  - Helps align outputs with the **natural image manifold** rather than just minimizing pixel differences

Implementation Details

- **Dataset Preparation:**
  - Collected **1080p 60fps** videos from YouTube
  - Extracted individual frames to create a diverse set of high-resolution images
- **Training Pipeline:**
  - Downsampled HR frames to simulate low-resolution inputs
  - Trained the model using paired LR–HR images
- **Testing Phase:**
  - Applied the trained model to **new unseen low-resolution videos**
  - Reconstructed upscaled outputs frame-by-frame
  - Combined upscaled frames to regenerate the final HD video

Why Use Diverse Scenarios for Video Enhancement Training?

Reason	Explanation
Generalization	So the model learns to enhance details across a wide range of textures, motions, and lighting conditions.
Avoiding Bias	Prevents the model from biasing towards specific content features (e.g., smooth gradients of skies) while ignoring others (like sharp text or thin wires).
Handling Different Degradations	Video compression artifacts and resolution losses differ by scene type — diverse training helps the model learn to restore details in multiple scenarios.
Robustness to Motion & Lighting Changes	Different scenarios have varying motion patterns and lighting, which affects temporal consistency and detail recovery in video enhancement.

Why Use Videos with Different Frame Rates (FPS) When Generating Datasets for Video Enhancement Models?

Reason	Explanation
Handle Variable Motion	Low-FPS videos have large motion between frames; high-FPS videos have smoother, smaller motion steps. The model should learn to enhance both cases.
Improve Temporal Consistency	Different FPS values produce different temporal artifacts. Training with varied FPS helps the model maintain consistent detail across time in both slow and fast-moving scenes.
Deal with Motion Blur & Compression	Fast scenes in low-FPS videos often have motion blur; high-FPS videos might have cleaner frames but finer details to enhance.
Make Model Deployment-Ready	n real-world applications (YouTube, CCTV, live sports, phone videos), FPS varies a lot — models trained on diverse FPS are more robust and generalize better.
Improve Frame Interpolation Capability (if used)	Some video SR models interpolate or align frames temporally. Having videos of different FPS helps them learn better motion estimation and compensation techniques.

What Happens When You Feed Results of Different Scenario Images as Feedback to Improve the Model?

What Happens?	Explanation
Error Correction on Edge Cases	The model adjusts its learned parameters to better handle difficult patterns it initially struggled with.
Better Generalization	By iteratively exposing it to tough cases, the model learns more varied data distributions, making it perform better on unseen content.
Bias Reduction	If the model was biased towards scenarios it saw more of in the initial training set, this feedback loop helps correct that.
Adaptive Detail Restoration	GAN-based models (like SRGAN) adjust their generator-discriminator balance based on where the discriminator detected fakeness — focused feedback from scenario-specific images strengthens weak spots in the generator.

Training Details

- **Upscaling Targets:**
  - Models were trained for **4× and 8× upscaling**.
- **Training Input:**
  - Only **1080p high-resolution (HR)** video frames were used as training data.
  - Images were **cropped into 96×96 patches** before processing.
- **Low-Resolution Simulation:**
  - HR patches were **blurred using a Gaussian filter** ($\sigma = 2.5$).
  - **LANCZOS/Bicubic** interpolation (higher-quality) used to downsample to low resolution.
  - This approach **mimics YouTube-style video compression** and downscaling artifacts.
- **Loss Functions:**
  - Total Generator Loss =
    - **Image Loss** (pixel-wise MSE)
    - **Perceptual Loss** using **VGG16** for feature similarity
    - **Adversarial Loss** to align output with the natural image manifold
    - **Total Variation (TV) Loss** to enforce spatial smoothness
- **Training Framework:**
  - Generator: ResNet with residual blocks and sub-pixel convolutions
  - Discriminator: CNN trained to distinguish real HR vs. generated images
  - Trained using **PyTorch**, with **Adam optimizer**, on the **Custom dataset**

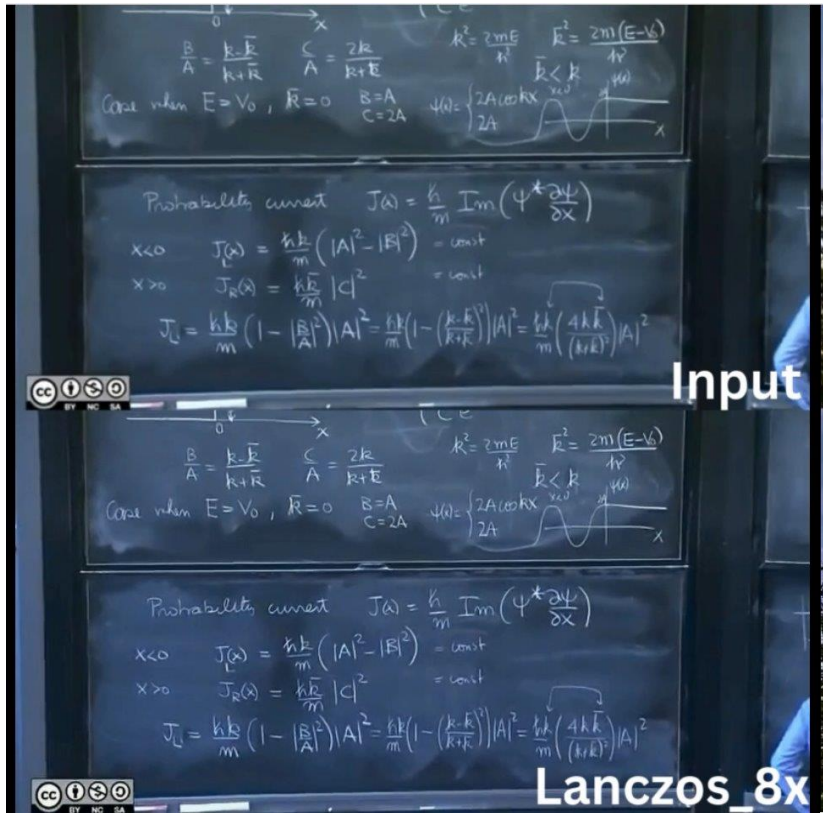
Testing and Inference Pipeline

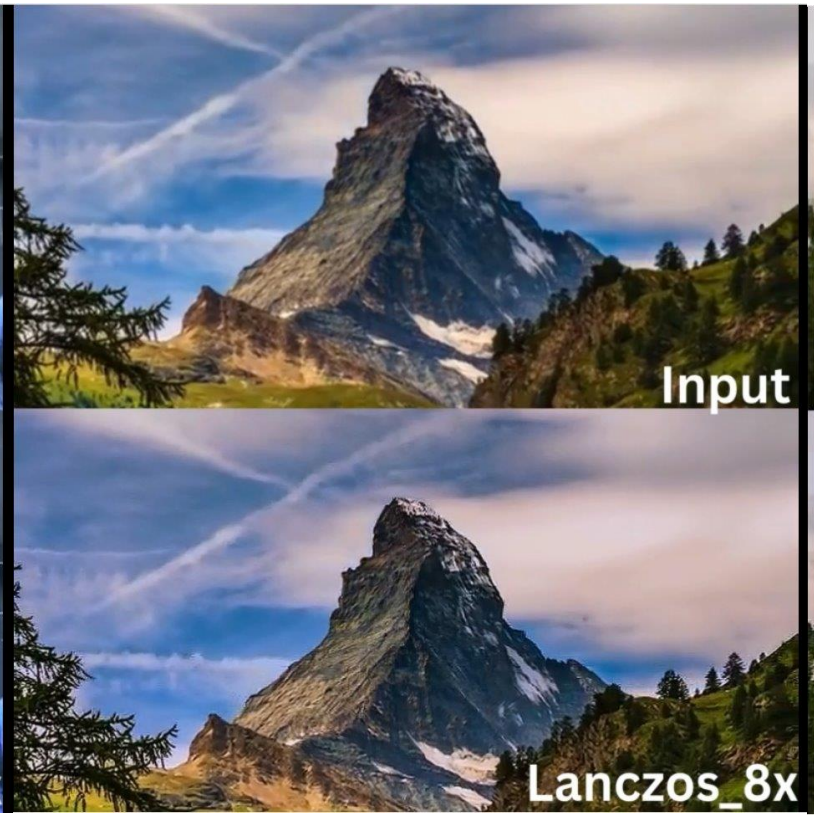
- **Load the Trained Model:**
  - The **SRGAN Generator** is loaded with pre-trained weights
  - Model is set to **evaluation mode** and moved to GPU if available.
- **Video Preprocessing:**
  - A **low-resolution video** is read using OpenCV (`cv2.VideoCapture`).
  - Each frame is extracted one-by-one and converted into a tensor format
- **Frame-wise Upscaling:**
  - Every frame is passed through the **SRGAN generator** for super-resolution.
  - Output tensors are converted back to **NumPy arrays** and formatted as high-resolution RGB frames.
- **Stitching into Video:**
  - All **upscaled frames** are written back into a video using `cv2.VideoWriter`.
  - Output is saved in **AVI format** at the original frame rate.
- **Output:**
  - A **super-resolved video** with 4× or 8× enhancement is generated.

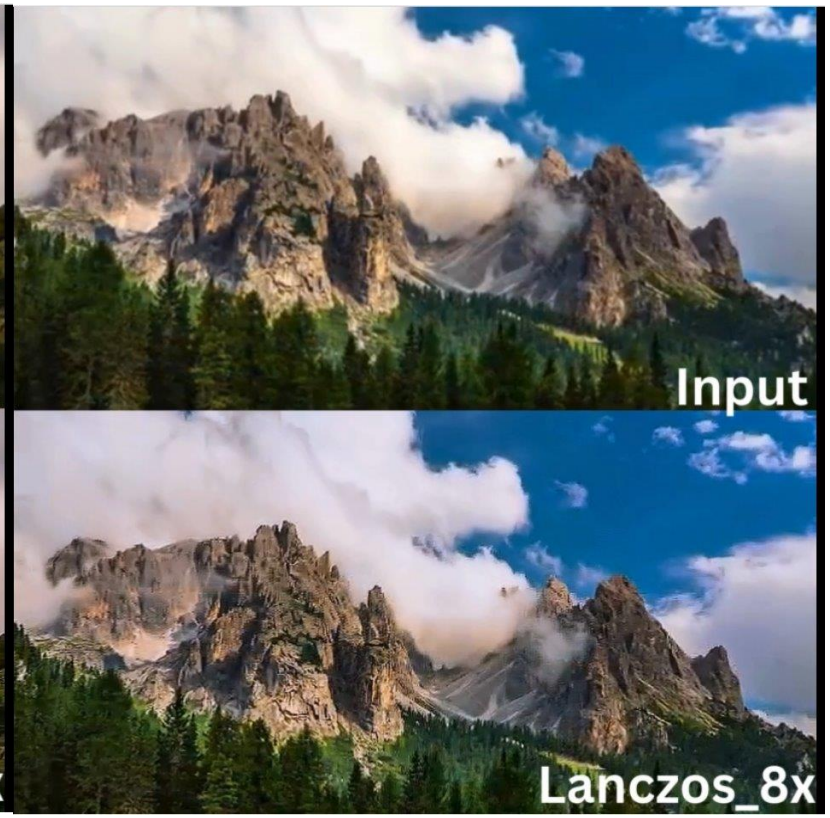
Results

We have attached the final upscaled video here: [Results]

Results

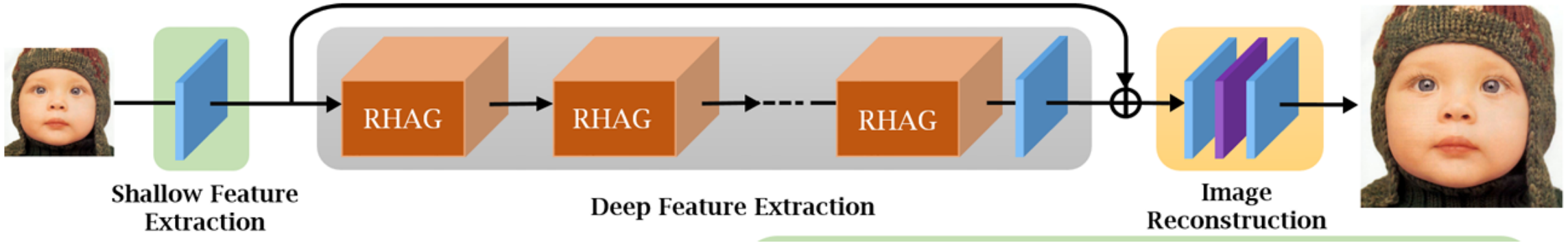






Transformer Based Approach

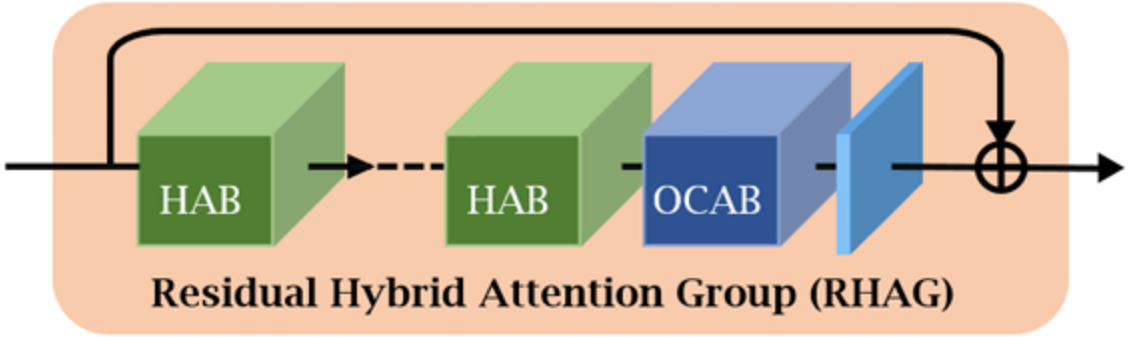
Hybrid Attention Transformer



Combines channel and self-attention for richer feature representation.

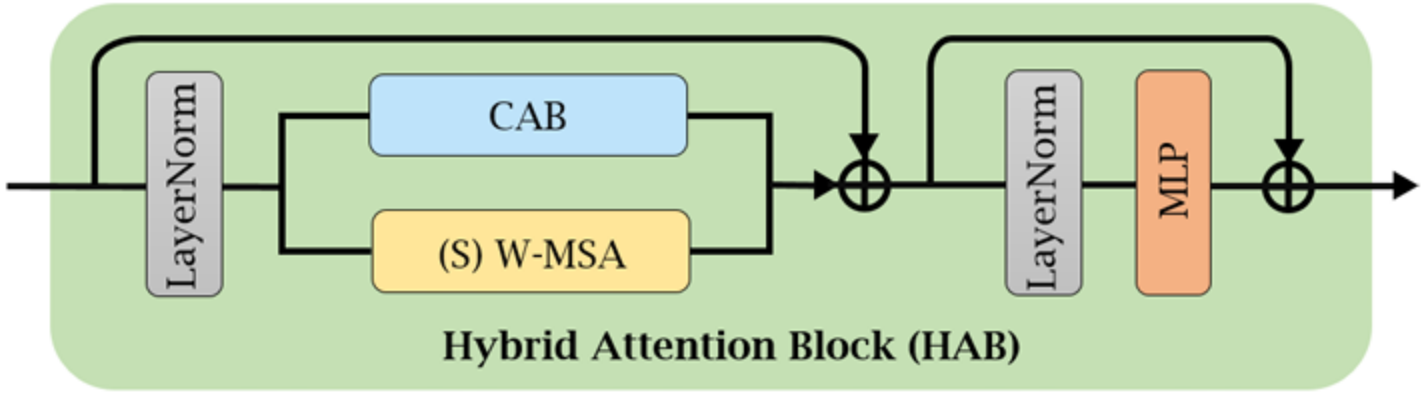
Uses overlapping cross-attention to improve window interaction.

Residual Hybrid Attention Group (RHAG)



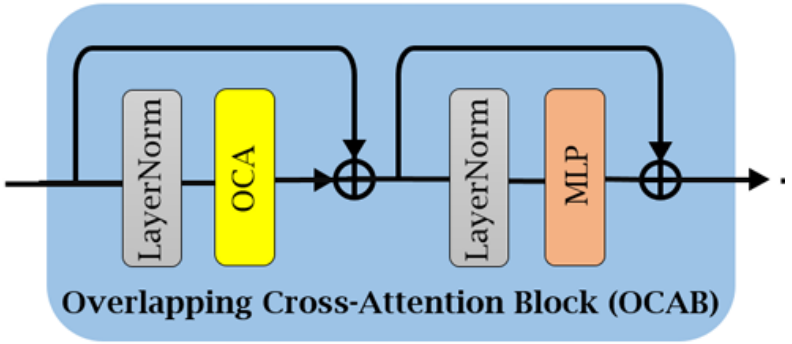
- Each RHAG serves as a modular unit in the deep feature extraction stage of HAT, containing multiple **Hybrid Attention Blocks (HABs)** and one **Overlapping Cross-Attention Block (OCAB)**, followed by a convolution layer with a residual connection to enhance information flow and representation capacity.
- By integrating HABs (which fuse window-based self-attention with channel attention) and OCAB (which strengthens cross-window communication), RHAGs enable the network to capture both **fine-grained details** and **global consistency**—crucial for super-resolution tasks.
- Each RHAG includes a residual connection from its input to output, which stabilizes gradient flow and facilitates **deeper model stacking** without degradation, improving both convergence and final performance.

Hybrid Attention Block (HAB)



- The HAB integrates a Channel Attention Block (CAB) with the standard Swin Transformer architecture. The CAB is placed after the first LayerNorm layer and works in parallel with the window-based multi-head self-attention (W-MSA) module. This addition helps the network focus on more important channels, thereby enhancing its representation power.
- The HAB employs a self-attention mechanism inside local windows, allowing the model to focus on small regions of the image. It uses a shifted window approach to enhance the connections between neighboring windows, which increases the range of pixels considered for attention. This strategy significantly improves the model's ability to capture global context and fine details in the image.

Overlapping Cross Attention Block (OCAB)



- OCAB consists of an Overlapping Cross-Attention (OCA) layer and a Multi-Layer Perceptron (MLP) layer. The OCA layer is designed to establish cross-window connections and enhance the window self-attention mechanism. The MLP layer follows the OCA layer and contributes to the transformation and integration of features.
- OCA partitions the input feature X into non-overlapping windows for queries (XQ) and overlapping windows for keys (XK) and values (XV). The partitioning is controlled by specific window sizes and a constant α to determine the size of overlapping windows relative to the query windows. OCA computes key/value pairs from a larger field compared to the query, facilitating cross-attention operations within each window feature.

Implementation Details

- **Dataset Preparation:**
  - Collected **1080p** and **360p lecture videos** from YouTube
  - Extracted individual frames to create a diverse set of high-resolution images
- **Training Pipeline:**
  - Re-sized **360p** video frames to **64x64** size images
  - Re-sized **1080p** video frames to **256x256** size images
  - Trained the model using **Adam optimizer**
  - Used **Huber loss** as loss function
  - Implemented **4x** super-resolution
- **Testing Phase:**
  - Applied the trained model to **new unseen low-resolution videos**
  - Reconstructed upscaled outputs frame-by-frame
  - Combined upscaled frames to regenerate the final video

Results

- It is present in this link
  Results

Challenges Faced and Future Prospects

Challenges Faced:

- **Understanding the Pretrained Architecture:**
  - Interpreting the SRGAN generator/discriminator and the VGG-based perceptual loss was non-trivial.
- **GPU Utilization & Debugging:**
  - High memory usage during training and inference.
  - Handling CUDA-specific issues and performance bottlenecks

Future Work:

- **Model Optimization:**
  - Reduce network depth to **speed up inference** without sacrificing too much quality.
- **Temporal Consistency:**
  - Use **frame-to-frame correlations** to ensure smoother video upscaling and reduce redundant computation.
- **Class-Based Specialization:**
  - Dynamically **select trained weights** based on the video type (e.g., wildlife, animation, sports).
- **App Development:**
  - Build a **lightweight app or tool** to bring super-resolution video enhancement to end-users

Contributions

- Ashwin: Worked on SRGAN Testing the model
- Joel: Worked on 2 sub classes of video training
- Mahendra: Worked on SRGAN Implementation and Training for general class video
- Nirmal: Worked on Transformer model
- Saad: Worked on Dataset generation and Fps management

Thank You