Video Frame Interpolation using Optical Flow

Kushaan Gowda
Columbia University
kg3081@columbia.edu

Sujeeth Bhavanam
Columbia University
sb4839@columbia.edu

Aastha Valecha
Columbia University
av3180@columbia.edu

***Abstract*—In this work, we propose a sophisticated method for video frame interpolation that aims to synthetically generate intermediate frames between consecutive frames in cartoon videos. Utilizing motion information encapsulated in optical flows, our approach employs a specially designed optical flow model with enhanced skip connections to identify areas of significant motion, which are indicative of key regions of interest. This model facilitates the estimation of motion dynamics between consecutive frames. We then deploy a U-Net architecture featuring a ResNet-based encoder and decoder with attention modules. This network architecture leverages the initial frame and its computed optical flow to interpolate intermediate frames accurately. The incorporation of attention mechanisms specifically targets regions with pronounced motion, thus refining the interpolation process. Various loss functions were explored, including Mean Squared Error (MSE), L1 loss, Sobel filter loss, and Multi-layer Perceptual loss. The final model employs a composite loss function that integrates Multi-layer Perceptual loss and L1 loss, strategically emphasizing motion-rich areas. This technique not only ensures the visual continuity of interpolated frames but also captures the unique motion and style characteristics inherent to cartoon animations, making it highly suitable for applications requiring detailed and precise motion representation in animated media. Our implementation is available on https://github.com/ecbme6040/e6691-2024spring-project-skar-sb4839-kg3081-av3180.**

I. Introduction

Creating cartoon videos is a detailed process that blends art with technical precision, largely dependent on the skilled hand drawings of animators. Each frame is drawn with great care, making the process both time-intensive and expensive. To cut costs, animation producers often repeat the same drawing across multiple frames, resulting in videos with lower frame rates. While this saves money, it also reduces the smoothness and liveliness of the animation. Consequently, there is a strong need for computational methods that can automatically fill in intermediate frames, enhancing video smoothness without the high costs of drawing each frame individually.

Recent advancements in video frame interpolation have shown promising results in enhancing the frame rate of natural videos by generating intermediate frames that transition smoothly between original frames. However, these techniques often fall short when applied to animation, where unique challenges arise from the stylistic and structural characteristics of cartoon imagery. For instance, the piece-wise smooth nature of animations, characterized by explicit lines and uniform color segments, presents significant difficulties in motion estimation and texture matching between frames. Additionally, the exaggerated and nonlinear movements common in cartoons pose further challenges that standard interpolation methods are not designed to handle.

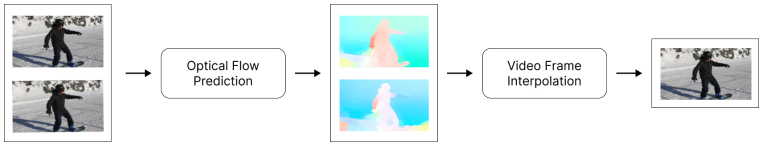


Fig. 1: Proposed method

In response to these challenges, as seen in Fig. 1, our project introduces a novel interpolation approach tailored specifically for the context of cartoon videos. Utilizing a dual-module architecture, our method focuses on capturing the intricate motion dynamics and artistic nuances typical of animated content. Initially, a custom-trained optical flow model is used to estimate the flow dynamics between consecutive frames. This model prioritizes capturing regions of high motion to better capture the lively movements crucial in animations.

Subsequently, our U-Net-based framework [1], which integrates attention mechanisms [3], utilizes the estimated optical flows to generate visually coherent intermediate frames by using attention on the regions of high motion. The interpolation process is refined through the application of a composite loss function, which combines Multi-layer Perceptual loss [4] and L1 loss, with an emphasis on accurately rendering areas of significant motion. This focused approach not only addresses the common interpolation challenges posed by the lack of textural diversity and the presence of large, nonlinear motions in animations but also sets a new standard for the automation of cartoon frame generation.

The contributions of this study are outlined as follows:

1) We introduce a sophisticated technique that involves extracting optical flows from consecutive frames to generate intermediate frames.
2) We develop and train two distinct models from the ground up, specifically for predicting optical flows and for frame generation.
3) We explore a variety of loss functions and architectural paradigms, providing an intuitive explanation of their outcomes through qualitative analysis.
4) We perform extensive experiments using the ATD-12K dataset to demonstrate the efficacy of our proposed method via both qualitative and quantitative analyses.

II. Related Work

Among recent advancements in video frame interpolation, AnimeInterp [5] introduces a significant innovation specifically tailored for animation videos, which traditionally pose unique challenges compared to natural video content due to a lack of textural details and exhibit non-linear, exaggerated motions. This framework uses Segment-Guided Matching and Recurrent Flow Refinement to specifically, tackle the smooth and piece-wise nature of animation art styles.

Continuing advancements in VFI technology have leveraged deep learning to effectively tackle complex motion dynamics and textural inconsistencies in videos. ACVI [6] introduces adaptive convolutions and machine learning to dynamically adjust the interpolation process based on detected motion patterns employing a series of convolutional neural networks (CNNs) that adapt their filters based on the velocity and direction of motion within the scene. By utilizing a context-aware kernel adaptation strategy, it can generate more accurate predictions of intermediate frames where motion is irregular or abrupt allowing for finer detail preservation and reducing artifacts commonly associated with fast-moving objects. FAOFE[7] enhances the capability of neural networks to focus on relevant features in the motion estimation process, particularly beneficial in densely textured or rapidly moving scenes by selectively amplifying features that are crucial for accurate motion prediction while suppressing irrelevant information. These techniques highlight the shift towards more adaptive and context-aware systems in video processing, aiming to achieve higher fidelity in motion portrayal and overall video realism.

III. Dataset

To support the development and evaluation of our video frame interpolation method for cartoon animations, we make use of a large-scale dataset from [5], named ATD-12K. This dataset is specifically designed for the animation industry and consists of a total of 12,000 frame triplets. The dataset is divided into two main subsets: a training set comprising 10,000 triplets and a testing set comprising 2,000 triplets. These triplets are sourced from a diverse array of cartoon movies to ensure a broad representation of animation styles and motion dynamics. The testing set is carefully curated from different sources than the training set to maintain objectivity and ensure the robustness of the evaluated interpolation methods.

The dataset consists of 3 frames, namely the input frame, the intermediate frame (frame to be predicted), and the next frame. The dataset also had optical flow values corresponding to going from the input frame to the next frame and vice versa.

A. Dataset creation

The dataset creation process involved several steps to ensure the data was suitable for training our sophisticated deep-learning model for frame interpolation. The original frame dimensions of 540x960 were resized to 256x256. This resizing was crucial not only to make the computational requirements more manageable but also to standardize the input size for our neural network models.

For the optical flow data, which includes the flow from the input frame to the next frame and vice versa, resizing was handled with particular care to preserve motion information accurately. Optical flows were adjusted by multiplying them with an appropriate scale factor. This normalization adjusted the magnitude of the flow vectors to correspond accurately to the new, lower resolution of 256x256. This step is critical as it ensures that the motion vectors remain representative of the actual motion between frames even after the spatial reduction. By scaling the optical flows appropriately, we maintain the integrity and usefulness of the motion data, which is essential for the effective training and performance of our interpolation models.

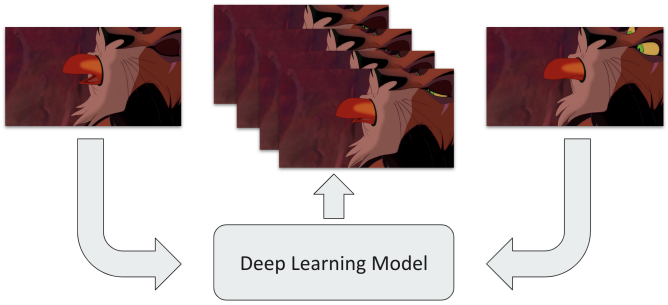


Fig. 2: Animation VFI

IV. Methodology

Our project focuses on video frame interpolation by synthesizing intermediate frames between two consecutive frames, using optical flows to guide the interpolation process as seen in Fig. 2. The methodology involves a two-stage training approach with specialized neural networks. First, we develop a custom optical flow model followed by a video frame generator using the UNet model.

In the detailed sections that follow, we will discuss:

- **Optical Flow Model Training**: The architecture and training specifics for precise motion prediction.
- **U-Net Configuration**: Details on the encoder-decoder structure and the role of attention modules
- **Loss Function Selection**: Examination of various loss functions, leading to our choice of a hybrid loss function that optimizes interpolation in motion-intensive areas.

A. Optical flow model

Our approach to optical flow estimation employs a dual-stream convolutional neural network, which we refer to as OptNet (as depicted in Fig. 3). This architecture draws inspiration from the FlowNet architecture, as discussed in [8]. In our model, each input image is processed separately within its respective stream before the streams merge to integrate the information. OptNet is specifically designed to independently encode the motion characteristics of each frame, subsequently combining these features to predict the optical flow between frames.

1) Architecture Overview:

- The network utilizes two parallel encoder streams, each dedicated to processing one of the input frames. These encoders are composed of multiple ResNet-style blocks, incorporating convolutional layers, batch normalization, and ReLU activations. This configuration is followed by max pooling to decrease the spatial dimensions while preserving essential features. This architecture allows the network to effectively and independently capture the unique characteristics of each frame, enhancing its ability to robustly analyze motion.
- After independently processing the frames, the features from both encoders are concatenated. This combined feature map is subsequently channeled through a bottleneck structure, which is crafted to integrate and refine the features, thereby priming them for precise flow estimation. The bottleneck comprises convolutional layers that first compress and then expand the feature dimensions, thereby facilitating an effective synthesis of features. This design enhances the network's ability to generate a comprehensive and accurate representation of the optical flow.
- The integrated features undergo a decoding process through several stages that incorporate upsampling to restore the resolution diminished during the encoding phase. At each decoder stage, features from the bottleneck are fused with prior features from each encoder stream through skip connections. This approach enhances the detail and accuracy of the reconstructed features. In the final stages, the decoder utilizes convolutional layers to further refine these features into an accurate two-channel optical flow map. This map effectively represents the motion vectors between the two frames, encapsulating the dynamics of the observed motion with high precision.

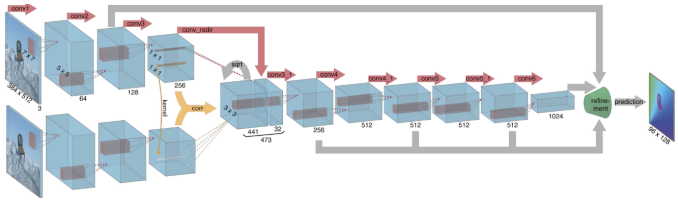


Fig. 3: OptNet model

B. Video Frame generator

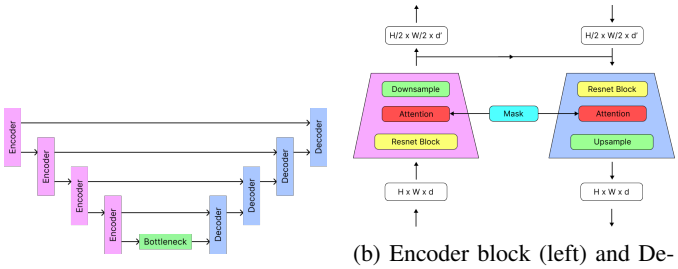
Our video frame interpolation approach employs a modified version of the UNet architecture, which we have named ResUNet, as illustrated in Fig. 4. This model is specifically tailored for generating intermediate frames in cartoon videos, utilizing both input frames and optical flow data. ResUNet integrates elements from U-Net and ResNet architectures, as discussed in [9], and includes attention mechanisms inspired by the [3] paper. These attention mechanisms are strategically incorporated to enhance the model's focus on areas exhibiting significant motion, thereby improving the accuracy and visual quality of the interpolated frames.

1) Architecture Overview: The ResUNet architecture is comprehensively structured into several main components: an encoder, a bottleneck, a decoder, and an output layer. Each of these components plays a crucial role in processing the inputs — the initial frame and an optical flow map — to predict the subsequent frame.

To predict an intermediate frame, we first generate two optical flow maps namely $flow_0$ and $flow_1$ which correspond to the flow map of going from frame 0 to 1 and vice versa. We then predict the optical flow of going from frame 0 to any intermediate frame as a convex combination of $flow_0$ and $flow_1$.

$$flow_t = (1-t) * flow_0 + (t) * flow_1 \text{ for } t \in [0,1] \quad (1)$$

Note that if $t = 0.5$, we predict the middle intermediate frame.



(a) UNet architecture

(b) Encoder block (left) and Decoder block (right)

Fig. 4: ResUNet model

- **Encoder**: The encoder processes the concatenated input of the initial frame and the optical flow map through multiple ResEncoderBlocks. Each block consists of a ResNet block for robust feature extraction, an attention module that applies a mask to focus specifically on areas with significant motion as indicated by the optical flow, and a downsampling layer to reduce spatial dimensions while retaining important features.
- **Bottleneck**: At the heart of the network, the bottleneck layer refines the features extracted and integrated by the encoder. It uses a sequence of convolutional layers and activations to prepare the features for accurate reconstruction by the decoder.
- **Decoder**: The decoder mirrors the encoder's architecture but in reverse, employing several ResDecoderBlocks. Each block includes an upsampling layer to progressively restore the feature maps' spatial resolution, combined with features forwarded from the encoder via skip connections. This helps in preserving and enhancing high-resolution details. Attention modules within the decoder again focus on the regions of significant motion to refine the frame reconstruction.
- **Output Layer**: The final output layer of the model transforms the processed features into the next frame. This layer ensures that the predicted frame accurately represents the motion and appearance transitions dictated by the optical flow inputs.

2) Implementation Details: In our ResUNet model, the attention mechanism's mask is generated by calculating the magnitude of the optical flow values using the formula $\sqrt{flow_x^2 + flow_y^2}$. This calculation is followed by a thresholding process and subsequent normalization. This method is crucial as it ensures that the network concentrates its processing on regions indicated by the optical flow. These regions are likely to encapsulate critical motion information, which is pivotal for the accurate prediction of intermediate frames. Such focused processing preserves the fluidity and dynamic expressions characteristic of animated sequences, thereby enhancing the model's effectiveness in video frame interpolation tasks.

V. Experimental Results

In this section, we begin by detailing the experimental setup used for our studies. We then proceed to discuss the performance and outcomes of the OptNet model. Following this, we describe the various experiments conducted during the training of the ResUNet model and elucidate the key findings from these experiments.

A. Setup

The training of our ResUNet and OptNet models was conducted on an N1-8 Standard Google Cloud Platform (GCP) instance equipped with **2 NVIDIA Tesla V100 GPUs**. This setup was chosen for its high performance in deep learning tasks, providing the necessary computational power to process large image datasets and complex neural network architectures.

Due to the high memory demands of our models, particularly when employing larger batch sizes, we utilized **PyTorch's Distributed Data Parallel (DDP)** module to facilitate distributed training across the two GPUs. Initially, we observed that a batch size of 32 yielded optimal results; however, a single V100 GPU encountered CUDA out-of-memory errors when handling more than 16 samples per batch. To circumvent this limitation, we distributed the batch across the two GPUs, each processing 16 samples, effectively mirroring the conditions of a single GPU handling a batch size of 32. This was achieved through synchronized gradient updates, ensuring that each training step was consistent across both GPUs, thereby maintaining the computational equivalence to a single GPU with a larger batch size.

This distributed training setup not only enhanced the training speed but also ensured that model optimization was not compromised by hardware limitations. By effectively doubling the available memory and computational resources, we were able to train our models more efficiently and achieve superior interpolation performance.

B. OptNet results

The OptNet model was trained using pairs of image frames, with the objective of predicting optical flow values from frame 1 to frame 2. A simple mean squared error (MSE) loss was utilized to assess the accuracy of the predictions against the ground truth, which was derived from the ATD-12K dataset. As depicted in Fig. 5, the training loss exhibited a decreasing trend over the epochs. Although the model was trained for a total of 100 epochs, it converged to an optimal solution by approximately epoch 30. Following the successful training and convergence of the OptNet model, its weights were subsequently frozen. This pre-trained model was then integrated into the ResUNet framework to facilitate the prediction of intermediate frames, ensuring that the ResUNet benefited from the robust optical flow estimation provided by OptNet.

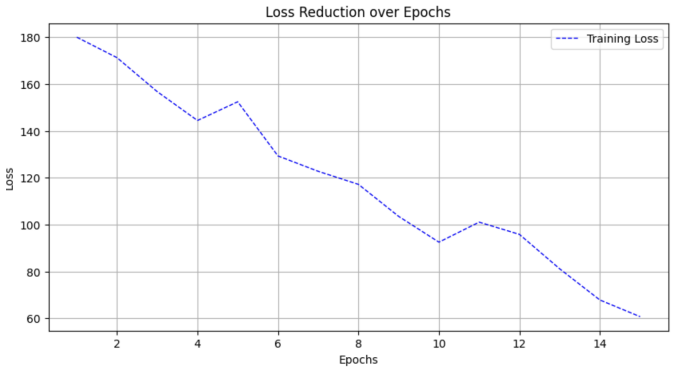


Fig. 5: OptNet loss

C. ResUNet results

In this section, we detail the series of experiments conducted to incrementally improve our ResUNet model, which is designed to predict intermediate frames in a video sequence. This methodical approach not only aids in understanding the impact of various modifications but also ensures that our model becomes progressively more effective at frame interpolation.

Initially, our experiments began with a baseline ResUNet model devoid of attention modules in both the encoder and decoder blocks. The culmination of these experiments involved integrating attention modules into the encoder and decoder blocks of the ResUNet model. This final configuration, incorporating attention mechanisms, represents our most advanced iteration and was determined to significantly enhance the model's ability to focus on and interpolate critical areas of motion within the video frames, leading to more accurate and visually coherent intermediate frames.

The metrics used to evaluate the model were:

1 **PSNR**: Peak signal to noise ratio is a statistical metric used in image and video analysis to measure the quality of a reconstructed image or video by quantifying the ratio between the maximum possible power of a signal (typically the maximum pixel intensity) and the power of corrupting noise that affects its fidelity.

2 **SSIM**: Structural Similarity Index Measure is an advanced metric used to assess the perceived quality of digital images and videos by comparing local patterns of pixel intensities that have been normalized for luminance and contrast

1) Experiment 1: ***MSE loss:*** In our initial experiment, we utilized the Mean Squared Error (MSE) loss function to train our model for image reconstruction. While MSE is a straightforward and widely used metric, it computes the average of the squares of the differences between the predicted and actual pixel values. This approach inherently penalizes high-frequency details, which could represent legitimate features of the image, leading to an excessive smoothing effect. Consequently, the resultant images as seen in Fig. 6 were notably blurry. This blurring arises because MSE does not account for the perceptual aspects of human vision, focusing strictly on pixel-level accuracy. Thus, the outcomes, although numerically accurate in terms of error reduction, lacked perceptual sharpness and detail, resulting in smoother but less visually accurate predictions. Table. I provides quantitative results of the model for this experiment.

(a) Input Frame

(b) Expected Output

(c) Predicted Output

Fig. 6: Predictions using MSE loss

TABLE I: Performance Metrics of MSE loss

	PSNR	SSIM
Train	7.87	0.3013
Test	7.5	0.292

2) Experiment 2: ***L1 loss:*** In this experiment, we implemented the L1 loss function in an effort to address the blurriness observed with the MSE loss. The L1 loss, which minimizes the average absolute difference between the predicted and actual pixel values, did reduce some of the blurriness compared to MSE. However, the improvement was not sufficient as seen in Fig. 7. Similar to MSE, L1 loss primarily focuses on pixel-level accuracy, which means it still fails to capture high-frequency details essential for achieving sharpness and detail in images. As a result, while there was a noticeable decrease in blurriness, the images produced under L1 loss still lacked the desired clarity and detail, highlighting the limitation of L1 in handling the nuances of image textures and edges effectively. Table. II provides quantitative results of the model for this experiment.

(a) Expected Output

(b) Predicted Output

Fig. 7: Predictions using L1 loss

TABLE II: Performance Metrics of L1 loss

	PSNR	SSIM
Train	8.8580	0.3533
Test	8.9905	0.354

3) Experiment 3: ***Sobel loss:*** In this experiment, we used a Sobel filter loss to improve the sharpness of predicted images by emphasizing edge detection. This approach involved processing both the predicted images and the ground truth through a Sobel filter, which emphasizes gradients and edges, followed by applying MSE to measure the discrepancies specifically at these edge regions. The intent was to ensure that the model accurately captures and reconstructs the edges, which are critical for visual sharpness and reducing blurriness.

However, while this method did enhance edge definition, it introduced new challenges. The primary issue was a loss imbalance; the model overly prioritized edge information at the expense of accurately reconstructing non-edge areas such as color and texture. This imbalance led to images that, while sharp at the edges, were often lacking in overall visual quality and fidelity in non-edge regions. Additionally, the Sobel filter's sensitivity to rapid intensity changes also meant that it could amplify noise. This sensitivity caused the model to sometimes interpret noise as edges, resulting in grainy and visually noisy images as seen in Fig. 8. Table. III provides quantitative results of the model for this experiment.

TABLE III: Performance Metrics of sobel filter loss

	PSNR	SSIM
Train	4.3	0.153
Test	4.132	0.1397

4) Experiment 4: ***Multi-layer Perceptual loss:*** In this experiment, we implemented a Multi-layer perceptual loss using the VGG16 architecture to address the limitations of previous loss functions. In this approach, images were processed

(a) Expected Output

(b) Predicted Output

Fig. 8: Predictions using sobel filter loss

through different layers of a pre-trained VGG16 network to extract hierarchical features from layers 9, 12, and 16. These features were then used to compute the Mean Squared Error (MSE) at multiple scales and abstraction levels, which allowed the model to evaluate similarity beyond mere pixel-level matching. This method emphasizes context and texture, offering robustness to small misalignments and typically leading to visually pleasing results as seen in Fig. 9.

Despite these improvements, a significant issue arose with the model's tendency to "learn shortcuts." Since in the majority of the dataset, the region of motion (ROM) was concentrated in a small area of the image, the model exploited this by essentially predicting the input frame as the output to minimize the perceptual loss. This behavior indicates that while the model effectively reduced the perceptual loss across the feature maps from VGG16, it did so by ignoring the optical flow information that was crucial for accurate motion prediction between frames. As a result, the model often returned the input frame as its prediction, undermining the purpose of generating a meaningful intermediate frame based on the provided motion dynamics. Table. IV provides quantitative results of the model for this experiment.

(a) Expected Output

(b) Predicted Output

Fig. 9: Predictions using Multi-layer perceptual loss

TABLE IV: Performance Metrics of Multi-layer perceptual loss

	PSNR	SSIM
Train	18.1500	0.6490
Test	17.9953	0.6363

5) Experiment 5: ***Multi-layer Perceptual loss and ROM loss****:* In this experiment, we combined Multi-layer perceptual loss with a Region of Motion (ROM) loss to better capture image features and focus on areas with significant movement. We used a mask based on optical flow magnitude values to highlight these active regions (active regions will have a high magnitude and the rest of the regions' values will be close to 0), expecting this would guide the model to accurately predict these areas.

However, the results were mixed as seen in Fig. 10. While the overall image quality improved, the predictions within the ROM were unexpectedly blurry. This suggested that although the model was now recognizing and focusing on the ROM, it was not effectively capturing the fine details of movement within these areas. This led us to our next and final experiment which was to introduce attention modules into the encoder and decoder layers. Table. V provides quantitative results of the model for this experiment.

(a) Expected Output

(b) Predicted Output

Fig. 10: Predictions using Multi-layer perceptual loss with ROM loss

TABLE V: Performance Metrics of Multi-layer perceptual loss with ROM loss

	PSNR	SSIM
Train	15.0245	0.4869
Test	14.3143	0.4736

6) Experiment 6: ***Multi-layer Perceptual loss with ROM loss and attention modules****:* To tackle the issue of blurriness in the Region of Motion (ROM), we refined our ResUNet model by incorporating attention modules into both the encoder and decoder blocks. These modules were designed to focus specifically on the ROM, utilizing a mask derived from the magnitude of the optical flow. This targeted attention helped enhance the model's precision in these crucial areas, effectively reducing blurriness and improving overall prediction quality as seen in Fig. 11.

Additionally, we conducted a sanity check to confirm that the model was indeed utilizing the optical flow information correctly. By inputting an optical flow map composed entirely of zeros, we observed that the model returned the input frame as the output. This outcome verified that the model's predictions were reliant on the optical flow data provided,

affirming the functionality of our attention-based approach in addressing previous limitations and achieving more accurate predictions in motion-intensive regions. Table. VI provides quantitative results of the model for this experiment and Fig. 12 shows the plots of the metrics during training and testing of the attention-based ResUNet model.

(a) Expected Output

(b) Predicted Output

(c) Expected Output

(d) Predicted Output

Fig. 11: Multi-layer Perceptual loss with ROM loss and attention modules

TABLE VI: Performance Metrics of Multi-layer perceptual loss with modified ROM loss

	PSNR	SSIM
Train	22.3106	0.7006
Test	22.2474	0.6889

D. Comparison with the original paper

TABLE VII: Results comparison with AnimeInterp[5]

Model	PSNR	SSIM
ResUNet	22.2474	0.6889
AnimeInterp	29.68	0.958

In the comparative analysis presented in table VII, our ResUNet model yielded a PSNR of 22.2474 and an SSIM of 0.6889, which are notably lower than those achieved by the AnimeInterp model, as cited from the original paper [5], which reported a PSNR of 29.68 and an SSIM of 0.958. Several factors contribute to this disparity in performance: (1) The AnimeInterp model utilizes a more sophisticated segmentation-based optical flow module, which likely enhances its ability to capture and utilize motion information more accurately. (2) AnimeInterp benefits from the use of a

(a) Train loss

(b) Test loss

(c) Train PSNR

(d) Test PSNR

(e) Train SSIM

(f) Test SSIM

Fig. 12: Plots of the metrics

pretrained model for generating images, which brings in pre-optimized weights and biases likely better suited for the task at hand, whereas our approach involved a simpler CNN-based model with a customized loss function designed from scratch. Additionally, (3) they pretrained their model first on a real-world dataset [2], and later fine-tuned it on their dataset [5], while our model was trained solely on the dataset used in their study. These factors collectively explain the performance difference between the two models.

VI. Challenges

During the course of our research, we faced several significant challenges:

- **Model Architecture Exploration:** Determining the optimal model architecture was a substantial challenge. Without using any pre-trained models, we had to experiment with various architectures to discover which ones would be most effective for our specific application.
- **Training from Scratch:** Starting without pre-trained models meant that every aspect of our model training was more complex, particularly in choosing and tuning hyperparameters, which required extensive testing and validation to optimize.
- **Compatibility Issues:** Working on a Google Cloud Platform (GCP) virtual machine introduced multiple compatibility challenges, particularly with NVIDIA drivers and PyTorch. These issues often disrupted our workflow and required additional troubleshooting.
- **Memory Limitations:** Given the large size of our dataset, we frequently encountered CUDA out-of-memory errors. Managing these errors involved optimizing our model's memory usage and adjusting batch sizes, which was a delicate balance to maintain performance without sacrificing accuracy.

VII. Conclusion

This study has made significant strides in improving video frame interpolation for cartoon animations. Initially, we explored traditional loss functions such as Mean Squared Error (MSE) and L1 loss, which tended to blur detailed image features. To overcome this, we introduced Sobel filter loss to better capture edges, though it sometimes overemphasized them at the expense of overall image quality.

Further refinements led us to adopt a Multi-layer perceptual loss, which improved texture representation by utilizing features from multiple layers of the VGG16 network. However, this approach sometimes resulted in the model predicting the input frame due to its lack of focus on regions of motion.

To address this, we combined Multi-layer perceptual loss with a Region of Motion (ROM) loss, enhanced by attention modules in our ResUNet model. This combination sharpened details in motion-rich areas, significantly enhancing the visual quality of interpolated frames. Our experiments confirm that these modifications provide a more balanced and effective solution for cartoon video frame interpolation, offering the potential for broader applications in animated content production.

VIII. Contributions

Sujeeth Bhavanam (sb4839), Kushaan Gowda (kg3081), Aastha Valecha (av3180)

TABLE VIII: Contributions of Each Team Member

Task	sb4839	kg3081	av3180
Data Preprocessing (%)	33	33	33
Model Coding (OptNet) (%)	0	0	100
Training (OptNet) (%)	0	0	100
Model Coding (ResUNet) (%)	50	50	0
Training & Parameter Tuning (ResUNet) (%)	50	50	0
Inference & Evaluation (Loss functions) (%)	50	50	0
Report Writing (%)	40	35	25

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015. 1

[2] X. Xu, L. Siyao, W. Sun, Q. Yin, and M. H. Yang, "Quadratic video interpolation," in *Advances in Neural Information Processing Systems*, vol. 32, 2019. 7

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. 1, 3

[4] M. S. Rad, B. Bozorgtabar, U.-V. Marti, M. Basler, H. K. Ekenel, and J.-P. Thiran, "SROBB: Targeted Perceptual Loss for Single Image Super-Resolution," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 1

[5] L. Siyao, Z. Shiyu, Y. Weijiang, S. Wenxiu, M. Dimitris, L. Chen Change, and L. Ziwei, "Deep animation video interpolation in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6587–6595, 2021. 2, 7

[6] L. Chen, et al., "Adaptive Convolutional Neural Networks for Video Frame Interpolation," in *Journal of Computer Vision and Image Understanding*, vol. 205, pp. 103022, 2022. 2

[7] W. Zhao, et al., "Deep Feature-level Attention Network for Optical Flow Estimation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[8] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2758–2766, 2015. 2

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. 3