EXEMPLAR-BASED VIDEO COLORIZATION USING VISION TRANSFORMER

FPT UNIVERSITY

Duong Thanh Tran
Nguyen Doan Hieu Nguyen
Trung Thanh Pham

Supervisors: M.S. Thuy-Duong Thi Vu
Dr. Duc Ngoc Minh Dang

Department of ITS
FPT University

A capstone project submitted in partial fulfillment of the requirement for the
Degree of Bachelor of Artificial Intelligence in Computer Science

FPT University December 2023

We dedicate this work to our families, teachers, and friends, whose unwavering support, guidance, and encouragement have been invaluable throughout our academic journey. Their belief in us, sacrifices, and constant presence have played a pivotal role in our achievements. We are grateful for the love, understanding, and inspiration they have provided, making this endeavor all the more meaningful.

Declaration

I hereby affirm that, except for explicitly referenced materials, this dissertation is the result of our original work and has not been previously submitted, either in its entirety or in part, to obtain any other degree or qualification, within this university or any other academic institution. This dissertation represents our independent efforts and does not incorporate any collaborative work with others, except where explicitly acknowledged in the text and Acknowledgements Section . Furthermore, this dissertation adheres to the prescribed word limit of fewer than 30000 words, inclusive of appendices, bibliography, footnotes, tables, and equations, and contains fewer than 30 figures.

Duong Thanh Tran
Nguyen Doan Hieu Nguyen
Trung Thanh Pham
December 2023

Acknowledgements

In the acknowledgment section, we express our gratitude to the individuals who have contributed their support, guidance, and encouragement to this project.

- Our esteemed supervisors, M.S. Thuy-Duong Thi Vu and Dr. Duc Ngoc Minh Dang, for their invaluable guidance, unwavering assistance, and meticulous review throughout the entirety of this research endeavor.
- Mr. Nam Phuong Tran for his generous provision of his hardware, which played a crucial role during the initial phase of our project.
- The organizers of the FPT Education Research Festival for their sponsorship greatly supported the project's successful completion.

This section is a tribute to the collaborative efforts of mentors, colleagues, friends, and family members who have played a vital role in our journey. We acknowledge their invaluable contributions and sincerely appreciate their support.

Abstract

Video colorization is a captivating and significant domain within the field of Computer Vision. Traditionally, Convolutional Neural Networks (CNNs) have been employed to extract features from individual video frames, while recurrent networks have been utilized to capture the inter-frame information. This conventional approach has achieved notable success in the colorization process. However, a major limitation of traditional CNNs is their restricted receptive field size, which restricts them to only capturing local information within a fixed-sized window. Consequently, these models struggle to directly capture long-range dependencies or pixel relationships across large image or video frame areas. To address this limitation, recent advancements in the field have embraced the use of Vision Transformers (ViT) and its variations to enhance performance. In this research, we propose ViTExCo and SwinTExCo, two end-to-end models specifically designed for the video colorization process. These models incorporate the Vision Transformer and Swin Transformer architectures as the backbones, respectively, leveraging their ability to capture global dependencies effectively. By employing Vision Transformer and Swin Transformer, ViTExCo and SwinTExCo surpass many other state-of-the-art methods in terms of quantitative and qualitative metrics. Through comprehensive experiments, we demonstrate the superiority of these proposed methods, highlighting their ability to achieve remarkable results in video colorization tasks. Overall, this research contributes to the advancement of video colorization techniques by integrating the powerful Vision Transformer and Swin Transformer architecture into the colorization process. The empirical evidence presented in this thesis underscores the effectiveness and superiority of the proposed ViTExCo and SwinTExCo models, establishing them as a valuable approach for video colorization in the field of Computer Vision.

Keywords: Video colorization, Exemplar-based, Vision Transformer, Swin Transformer, Computer Vision

Table of contents

List of figures

List of tables

Acronyms

BEiT BERT Pre-Training of Image Transformers.

CNN Convolutional Neural Network.

DEiT Data-Efficient Image Transformers.

GANs Generative Adversarial Networks.

RAFT Recurrent All-Pairs Field Transforms.

SwinT Swin Transformer.

TCVC Temporal Consistent Automatic Video Colorization via Semantic Correspondence.

VCGAN Video Colorization with Hybrid Generative Adversarial Network.

ViT Vision Transformer.

Chapter 1

INTRODUCTION

1.1 Overview

1.1.1 Context

In the field of Computer Vision, colorization is crucial for enhancing the appearance and historical value of old black-and-white images and videos. Accurately deducing the original colors and transforming them into colorized versions has traditionally posed challenges. However, recent advancements in deep learning have provided us with automated and realistic methods for videos with colors. By utilizing neural networks, we can revive vintage footage and generate compelling and engaging content that resonates with viewers while preserving the source material's integrity.

1.1.2 Image Colorization

Image colorization involves the process of adding colors to black-and-white images, and there are various methods to accomplish this. In the past, artists would manually apply colors to each image, requiring substantial time and skill.

But now, with the progress in computer vision and deep learning, we have various methods for colorizing images automatically. These techniques use large datasets to train deep neural networks to understand how grayscale and color images are related. One common approach is to use Convolutional Neural Networks (CNNs) [3] to identify features and Generative Adversarial Networks (GANs) [4] to predict colors for each pixel.

1.1.3 Video Colorization

Video colorization is a natural progression of image colorization techniques, incorporating temporal consistency across a sequence of frames. While image colorization focuses on adding colors to individual images, video colorization considers the sequential context of a video. As a result, applying image colorization methods directly to videos presents additional challenges. Some of the key problems encountered in video colorization include:

- **Temporal consistency**: Maintaining color consistency across the entire video is crucial to avoid flickering or abrupt color changes in the resulting video. Methods need to consider the temporal relationship between frames to ensure smooth and visually coherent colorization.
- **Time efficiency**: Video colorization requires efficient algorithms capable of processing frames at high speeds. The computational complexity of colorization methods must be carefully addressed to achieve smooth responsiveness.
- **Artifacts and noise**: Video colorization introduces artifacts and noise due to imperfect color predictions or inconsistencies across frames. Reducing these artifacts is essential to produce visually pleasing and realistic colorized videos.

1.2 Motivation

The motivation behind video colorization stems from the following factors:

- **Technical motivation**: In recent years, Vision Transformer (ViT) [5] and its variants have achieved impressive performance in the Computer Vision field. By applying these new architectures, we desire to create a novel model for video colorization that not only achieves competitive performance compared to the existing models but also minimizes computational complexity, enabling faster processing without compromising the quality of the results.
- **Practical motivation**: The historical value of colorizing old videos and films cannot be overstated. By adding color to the monochromatic videos, we enhance their historical significance and make them more accessible and relatable to contemporary audiences. It allows us to connect with history on a deeper level and gain a richer understanding of our shared heritage.

Chapter 2

RELATED WORK

2.1 Related Approaches

The domain of video colorization encompasses three principal methodologies: Interactive Colorization, Fully Automatic Colorization, and Exemplar-based Colorization. Interactive Colorization involves techniques that rely on user guidance and input for colorization. Fully Automatic Colorization strives to automate the colorization process without requiring any user-provided hints. Exemplar-based Colorization, on the other hand, employs reference images as a means to propagate colors onto the target video frames. Each approach exhibits distinct advantages, limitations, and contributions in the field of video colorization.

2.1.1 Interactive Colorization

One of the most basic and direct methods for colorization was using user-provided hints [6–8]. These hints could be in the form of color points, strokes, or scribbles, which guide the model to determine which colors should be applied to specific positions in the image. Methods based on the interactive approach often regress pixels' color values with the assumption that pixels belonging to the same object or texture pattern area would have a connection in colors. The hints could be local and global [9], which were fed into an overall network. Some models leveraged ViT [5] to colorize images such as iColoriT [10]. This method solved the suffering of partial colorization even in the same local area with the same intensity level by using ViT to capture long-range dependencies of color areas to overcome the previous methods's drawbacks.

Although the interactive approach was relatively simple, it had proven effective in producing colorized images. However, user-guided methods were unsuited for video colorizing tasks due to the significant human effort and aesthetic skills required to produce colorful videos.

The complexity of these techniques necessitated a considerable investment of time and resources, which may have needed to be more practical for video colorization projects.

2.1.2 Fully Automatic Colorization

Another approach to colorizing grayscale images or videos was training a deep neural network. Several approaches have been proposed, including using CNNs [3] to encode the input grayscale image to a latent space and then decode to the colorized image, as demonstrated in [11, 12]. Other methods [13, 14] employed GANs to generate the color images. Recently, [15] applied Swin Transformer (SwinT) [16] instead of CNNs for a better result in colorizing images because it outperformed traditional CNNs in computer vision tasks.

GANs were also applied to video colorization in [17, 18] and achieved remarkable results. One of the latest papers for fully automatic colorization was Video Colorization with Hybrid Generative Adversarial Network (VCGAN) [17]. This paper divided the model's training into 2 stages so that it could ensure the production of perceptually plausible colorization. However, this colorization approach faced several challenges. Firstly, it required a large dataset and a large network, making the training process difficult. If the training dataset was limited, the model's ability to generalize to new videos might have been compromised. Another limitation was the lack of easy customization in the resulting colorization. Unlike interactive or exemplar-based methods that allowed users to specify desired colors for specific objects or regions, the deep learning approach proved to be less flexible in this regard. Consequently, it might not have been suitable for applications that necessitated customized colorization.

2.1.3 Exemplar-based Colorization

Examplar-based colorization utilizes many architectures to learn the similarity map between grayscale images and color reference images. These neural networks are trained on a large dataset of colorized images, allowing them to understand the complex relationships between different color palettes and corresponding grayscale inputs. This approach could also be applied to video colorization by colorizing each frame individually. Deep exemplar-based Video colorization [19] used 2 subnetworks: the correspondence subnetwork and the colorization subnetwork to ensure consistency. The correspondence subnetwork identifies the matching areas between the video frame and the reference image in the feature space and adjusts the reference color accordingly. Using the intermediate outcome of the correspondence map and the previously colorized frame, the colorization subnet estimates the color for the current

frame. Another research using an exemplar-based method is Temporal Consistent Automatic Video Colorization via Semantic Correspondence (TCVC) [20], which maintains long-range consistency by combining semantic correspondence into automatic video colorization. TCVC proposed a network with two stages: The first stage in the colorization process is the reference colorization network, which is built to automatically colorize the first frame of every video to create a reference image for guiding the subsequent colorization stage. The second stage, which consists of a semantic correspondence network and an image colorization network, was then built to colorize the remaining frames using the reference.
In summary, this approach could lead to flickering and blending issues if not implemented carefully.

2.2 Backbones

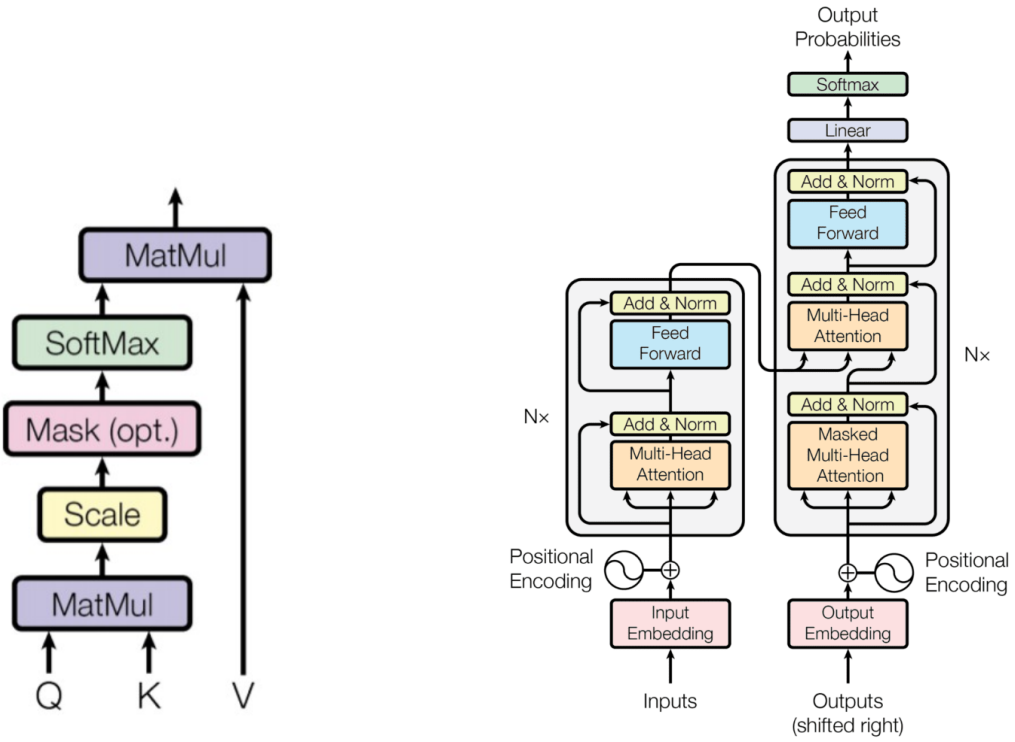
2.2.1 Overview

In the context of colorization models, the process of embedding input images into feature maps plays a crucial role in extracting meaningful representations from the visual content. These feature maps serve as the foundation for subsequent stages of the colorization model, facilitating the generation of accurate and plausible colorized outputs. Several backbone models have been widely adopted for this purpose, each offering distinct architectural characteristics and performance trade-offs. In this section, we provide an overview of some commonly used backbone models for image feature extraction in colorization.

2.2.2 Transformer architecture

2.2.2.1 Definition

Transformer architecture [21], which is displayed in Figure 2.1b, is a novel neural network design primarily used for natural language processing (NLP) tasks, first introduced in the paper "Attention is All You Need". It has become a leading approach in the field, delivering state-of-the-art results across various NLP tasks. In distinction to traditional recurrent neural networks (RNNs), Transformer does not use a recurrence mechanism for sequence processing. Instead, it leverages the concept of Attention, enabling the network to selectively focus on relevant parts of the input sequence. This mechanism allows Transformer to effectively capture long-range dependencies and contextual information.
Transformer architecture comprises an encoder and a decoder, consisting of multiple layers of self-attention, cross-attention, masked self-attention, and feedforward neural networks.



(a) Attention mechanism architecture

(b) Transformer architecture

Fig. 2.1 Attention mechanism architecture and Transformer architecture

The encoder processes the input sequence while the decoder generates the output sequence. Self-attention facilitates capturing dependencies and relationships between elements within the input sequence. Meanwhile, the feedforward networks enable the Transformer to model complex nonlinear associations between the input and output.
A vital advantage of the Transformer architecture is its parallelizability and efficiency due to the absence of recurrent computations. This advantage allows for faster training and inference procedures. The Transformer has significantly improved the performance of NLP models, particularly in tasks that necessitate comprehending long-range dependencies and intricate relationships between input and output elements. Its success stems from its ability to effectively process sequential data, such as text, and exploit Attention mechanism to focus on the most relevant information within the input sequence.

2.2.2.2 Attention Mechanism:

2.2.2.2.1 General Attention Mechanism The general Attention mechanism, as shown in Figure 2.1a, is a key component in many deep learning models, particularly in natural language processing and computer vision. It allows the model to focus on specific parts of the input data deemed relevant for the task.

One popular form of Attention is the scaled dot-product Attention, which can be represented by the following formula:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

where

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.2}$$

The Q matrix represents the **q**ueries or questions we have about each element in a sequence. The K matrix contains the **k**eys, representing the elements we want to compare against the queries. The V matrix holds the **v**alues associated with each element. By computing the relevance scores between the queries and keys through multiplication and applying softmax, we obtain attention weights.

This general attention mechanism has proven to be effective in various domains, enabling models to capture meaningful dependencies and improve performance on a wide range of tasks.

2.2.2.2.2 Self-attention mechanism The self-attention mechanism, also known as intra-attention or self-attention, is a specific type of attention mechanism that allows a model

to attend to different positions within its own input sequence. It has gained significant popularity, especially in natural language processing tasks such as machine translation and language understanding. One of the commonly used formulations for self-attention is the scaled dot-product attention, which can be written as:

$$\text{SelfAttention}(X) = \text{softmax}\left(\frac{XW_Q(XW_K)^T}{\sqrt{d_k}}\right)XW_V \tag{2.3}$$

In the above formula, X represents the input sequence, and W_Q, W_K, and W_V are learnable weight matrices for the query, key, and value projections, respectively.

The self-attention mechanism allows the model to capture dependencies between different positions in the input sequence, enabling it to attend to relevant information regardless of the position's distance.

2.2.2.3 Transformer Components

The Transformer consists of several key components that enable effective modeling of sequential data:

2.2.2.3.1 Positional Encoding In the Transformer, positional encoding is used to inject information about the relative or absolute positions of the tokens in the input sequence. Because the architecture lacks any inherent notion of order, positional encoding ensures that the model can distinguish between different positions and capture sequential dependencies. One commonly used method for positional encoding is to add sinusoidal functions of different frequencies and phases to the input embeddings. This allows the model to learn representations that encode both the content and the position of each token.

One common formulation for positional encoding is as follows:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{2.4}$$

$$\text{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right), \tag{2.5}$$

where pos represents the position of the token, i represents the dimension index, and d_{model} is the dimensionality of the model's input embeddings. The positional encoding is added to the input embeddings before being fed into the encoder. By introducing these sinusoidal functions, the model can encode positional information in a way that allows it to generalize to sequences of arbitrary lengths.

2.2.2.3.2 Encoder The encoder component of the Transformer is responsible for processing the input sequence and capturing its contextual information. It consists of multiple layers of self-attention and feed-forward neural networks. Each layer in the encoder independently attends to different positions in the input sequence, allowing the model to capture both local and global dependencies. The self-attention mechanism enables the model to focus on relevant parts of the input at each position, while the feed-forward networks provide non-linear transformations to enhance the learned representations. The encoder layers are connected in a sequential manner, allowing the model to refine its understanding of the input sequence iteratively.

2.2.2.3.3 Decoder The decoder component of the Transformer is designed to generate an output sequence based on the encoded representation of the input sequence. It also consists of multiple layers, each containing masked self-attention and cross-attention mechanisms. The masked self-attention mechanism in the decoder enables it to attend to various positions in the output sequence, ensuring that each position has access to the relevant context. During model training, a mask is applied to prevent the model from observing future tokens, allowing it to focus solely on previous and current tokens. The cross-attention mechanism enables the decoder to attend to the encoded representation of the input sequence, helping in the generation of output tokens based on both the input and the previously generated tokens. The decoder layers are stacked sequentially, allowing the model to progressively refine its output sequence.

The combination of positional encoding, encoder, and decoder components forms the core of the Transformer architecture, enabling it to effectively model complex sequential dependencies and achieve state-of-the-art performance on a wide range of natural language processing tasks.

2.2.3 Vision Transformer (ViT)

ViT [5] was a breakthrough invention that revolutionized the field of Computer Vision when it provided impressive results compared to CNNs [3] in the same domain. Several variants of ViT were invented after that, such as MobileViT [22], SwinT [16], Data-Efficient Image Transformers (DEiT) [23], BERT Pre-Training of Image Transformers (BEiT) [24], etc. They solved the problems that the original model faced and gained remarkable results. These models quickly became dominants in playing the role of the backbone in many Computer Vision research. The birth of many transformer-based models was thanks to the original Vision Transformer model.

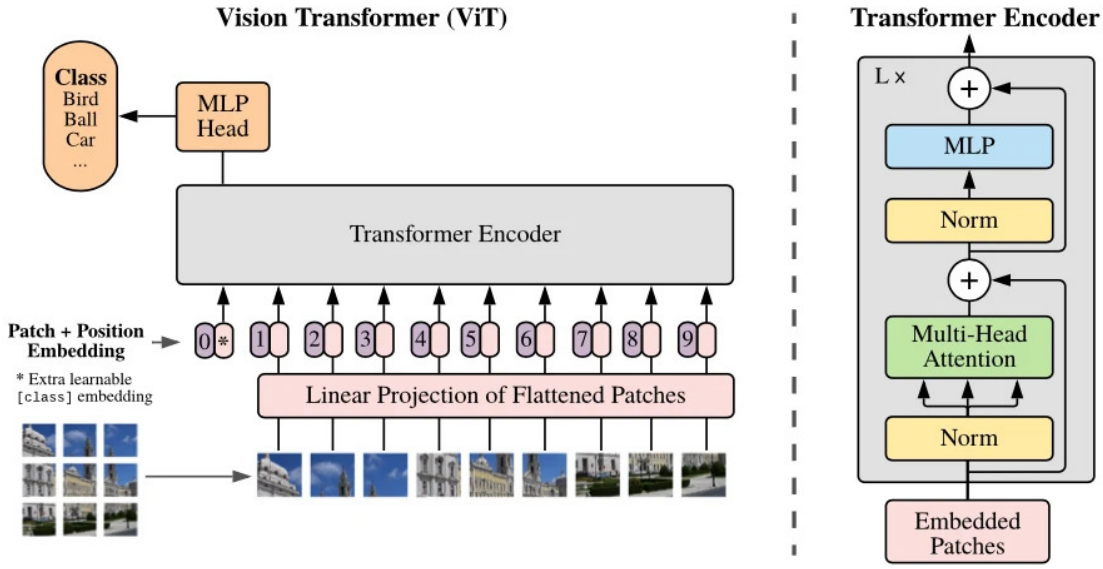


Fig. 2.2 Vision Transformer architecture overview

2.2.3.1 Overall Architecture

The model's architecture is presented in Figure 2.2, which is characterized by its departure from the traditional CNNs commonly used in Computer Vision tasks. ViT relies on a self-attention mechanism that allows it to capture global contextual information. Instead of using convolutional layers, ViT employs a series of self-attention layers and feed-forward neural networks. The self-attention layers enable the model to attend to different parts of the input image, establishing contextual relationships between them. The feed-forward networks, consisting of multiple layers, transform the learned representations. This combination of self-attention and feed-forward networks forms the core architecture of Vision Transformer, enabling them to effectively process and understand visual data.

2.2.3.2 Patch and Position Embedding

One key aspect of ViT is the concept of patch and position embedding. To process images, it divides the input image into smaller patches, treating each patch as an independent token. These patches are then linearly projected to create a set of embeddings that represent the visual content of the image. The position embedding, on the other hand, encodes the spatial information of the patches by assigning each patch a unique positional encoding. This positional encoding allows the ViT to understand the relative positions and spatial relationships between patches. By combining patch and position embedding, ViT can

effectively process the global contextual information of the image while preserving the spatial information necessary for image understanding.

2.2.3.3 Transformer Encoder

The transformer encoder is a crucial component of ViT. It consists of multiple stacked self-attention layers and feed-forward networks. Each self-attention layer allows the model to attend to different patches in the input image, capturing their dependencies. The self-attention mechanism computes attention weights for each patch based on its relationship with other patches, enabling the model to focus on relevant visual information. The feed-forward networks process the output of the self-attention layer, transforming the learned representations. The transformer encoder operates iteratively, with each layer refining and enriching the representations learned in the previous layer. This hierarchical structure enables Vision Transformer to capture complex visual patterns and dependencies, leading to their impressive performance in various computer vision tasks.

2.2.4 Swin Transformer (SwinT)

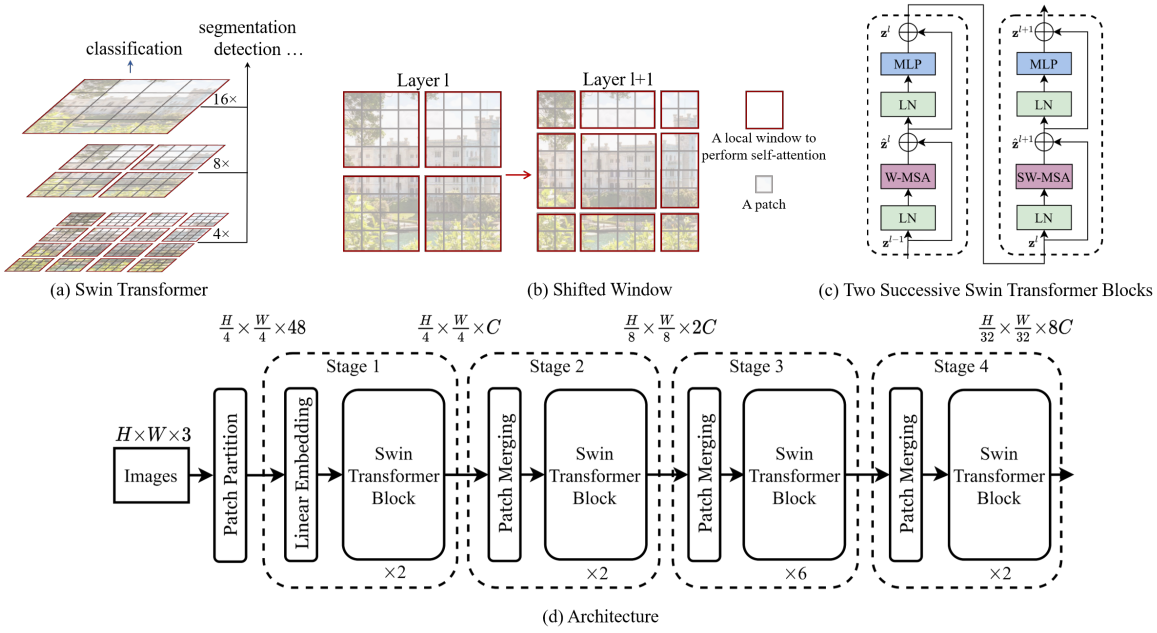


Fig. 2.3 Swin Transformer architecture overview

The SwinT [16] is a state-of-the-art architecture for computer vision tasks that extends the Transformer model to effectively capture visual information in images. It introduces a novel approach of using shifted windows and self-attention mechanisms to model both local

and global dependencies within an image. By leveraging self-attention, the SwinT achieves remarkable performance in various computer vision tasks, including image classification, object detection, and semantic segmentation.

2.2.4.1 Overall Architecture

The SwinT architecture is composed of multiple layers of shifted window-based self-attention blocks. These blocks are stacked hierarchically, allowing the model to capture information at different scales. Each block operates on a grid of non-overlapping patches or tokens, treating them as independent entities for self-attention computations. Figure 2.3 gives an overview of the SwinT architecture.

2.2.4.2 Shifted Window-based Self-Attention

2.2.4.2.1 Self-attention in non-overlapped windows To enable efficient computation and capture local dependencies, the SwinT performs self-attention within non-overlapping windows. This approach reduces the computational complexity compared to traditional self-attention, which operates on every pair of tokens. The self-attention mechanism computes attention weights between queries and keys within each window, which are then used to obtain a weighted sum of the corresponding values.

2.2.4.2.2 Shifted window partitioning in successive blocks In successive blocks, the SwinT introduces a shifted window partitioning strategy. This strategy shifts the windows by a certain stride, creating overlapping regions between adjacent blocks. The overlapping regions help capture contextual information that extends beyond the boundaries of a single window. This shift-based partitioning enhances the SwinT's ability to model long-range dependencies efficiently.

2.2.4.2.3 Hierarchical architecture The SwinT builds hierarchical feature maps by merging image patches in deeper layers. This process involves aggregating information from smaller patches to form larger receptive fields, enabling the model to capture context at different scales. The merging of image patches is depicted by shaded areas in the architecture, representing the combination of patch-level features into higher-level representations.

2.2.4.3 Relative position bias

To account for the spatial relationship between different tokens, the SwinT [16] employs a relative position bias. The relative position bias provides additional positional information

that helps the model attend to relevant locations within the image. It is incorporated into the attention calculation by adding a learned bias term to the dot product of queries and keys. The relative position bias term can be included in the attention formula as follows:

$$\text{Attention}(Q,K,V) = \text{SoftMax}(QK^T/\sqrt{d}+B)V \tag{2.6}$$

2.2.4.4 Improvement in Swin Transformer v2

SwinT v2 [25] introduces several adaptations, as displayed in Figure 2.4, to enhance its capacity scaling capabilities and performance:

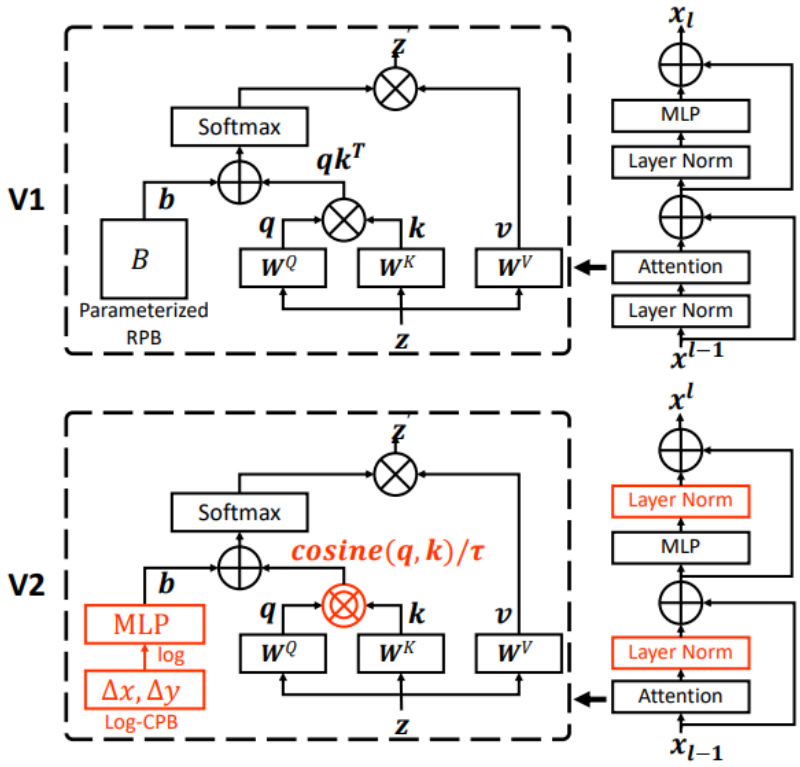


Fig. 2.4 Adaptions made on the original SwinT architecture

2.2.4.4.1 Res-Post-Norm Configuration for Improved Capacity Scaling To facilitate easier capacity scaling, SwinT v2 replaces the previous pre-norm configuration with a res-post-norm configuration. In the res-post-norm configuration, layer normalization and residual connections are applied in the opposite order. This change simplifies the architecture and improves training stability, making it easier to scale up the model's capacity without encountering optimization difficulties.

2.2.4.4.2 Scaled Cosine Attention for Enhanced Capacity Scaling In SwinT v2, the original dot product attention is replaced with scaled cosine attention to improve capacity scaling. The scaled cosine attention computes the cosine similarity between queries and keys and scales it by the square root of the dimension. This modification helps alleviate the vanishing gradient problem, which is critical for scaling up the model's capacity. By enhancing gradient flow, the SwinT v2 can more effectively handle larger models with increased capacity. The detailed formula of Scaled Cosine Attention is shown as follows:

$$\text{Sim}(q_i, k_j) = \cos(q_i, k_j)/\tau + B_{ij} \tag{2.7}$$

2.2.4.4.3 Log-Spaced Continuous Relative Position Bias SwinT v2 [25] introduces a log-spaced continuous relative position bias approach to replace the previous parameterized approach. While this adaptation is not directly related to capacity scaling, it provides more flexibility in modeling relative positions and allows the model to capture long-range dependencies more effectively. By using a log-spaced grid, the model can encode both local and global positional information in a more precise and adaptable manner, which can be advantageous when scaling up the model's capacity.

2.2.4.5 Compare Swin Transformer to Vanilla Vision Transformer

Compared to the vanilla ViT [5], the SwinT offers several advantages. The use of shifted windows and self-attention in the Swin Transformer allows it to efficiently capture both local and global dependencies, making it more effective in modeling spatial relationships within an image. Additionally, the shifted window partitioning strategy enables the Swin Transformer to capture contextual information across different scales, enhancing its ability to model long-range dependencies.

Furthermore, SwinT v2 introduces improvements such as the res-post-norm configuration, scaled cosine attention, and log-spaced continuous relative position bias. These enhancements enhance training stability, improve information propagation, alleviate the vanishing gradient problem, and provide better modeling of positional information.

In contrast, the vanilla ViT operates on fixed-size patches without considering local dependencies explicitly. It relies solely on the self-attention mechanism to capture relationships between patches. While effective, this approach may be computationally expensive and less efficient in modeling local information compared to the SwinT.

Overall, the SwinT demonstrates superior performance in various computer vision tasks and offers enhancements that address specific limitations of the vanilla ViT architecture.

2.3 Optical flow

2.3.1 Overview

Video colorization models aim to generate realistic and coherent colorized frames for a given grayscale video sequence. One challenge in video colorization is maintaining consistency in color predictions across frames to ensure natural and visually pleasing results. To address this challenge, optical flow has been widely utilized as a constraint mechanism in video colorization models. In this section, we provide an overview of optical flow and its application in maintaining color consistency throughout an entire video.

Optical flow refers to the pattern of apparent motion of objects in an image or video sequence. It represents the displacement vectors of points between consecutive frames, capturing the pixel-level correspondence between frames. Optical flow estimation allows us to understand the motion information in a video and can be used as a valuable tool for various computer vision tasks.

In the context of video colorization, optical flow is employed as a consistency constraint to ensure that the color predictions remain consistent across frames. The basic idea is to propagate the color information from one frame to the next based on the estimated optical flow, thereby maintaining the coherence of colors throughout the video sequence.

By aligning the colorized frames with the estimated optical flow, the color information is propagated smoothly from one frame to the next, resulting in temporally consistent colorization. This constraint mechanism helps to alleviate flickering or abrupt changes in color that might otherwise occur between consecutive frames when colorizing them independently. Figure 2.5 gives a visual demonstration for optical flow.

Fig. 2.5 An example of optical flow image on the right, corresponding to two consecutive images on the left.

2.3.2 Recurrent All-Pairs Field Transforms (RAFT)

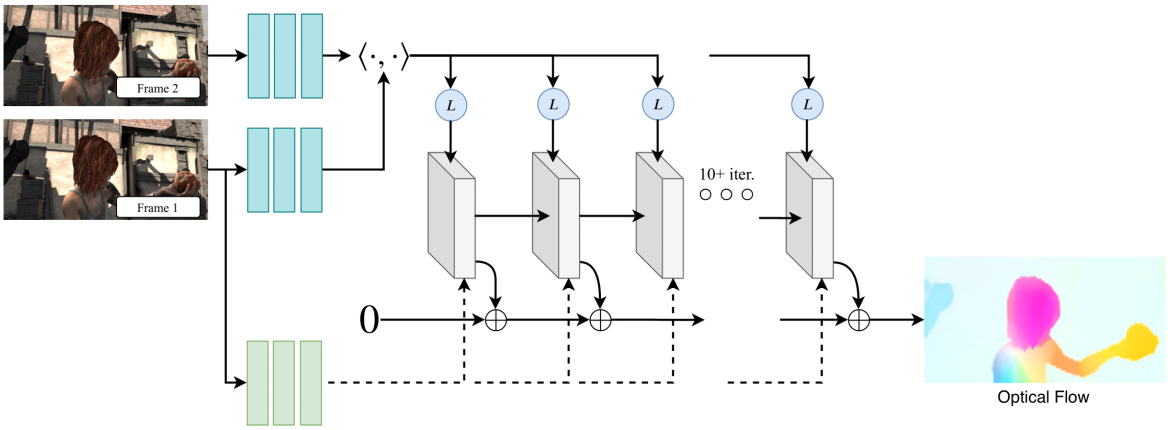


Fig. 2.6 RAFT architecture overview

Recurrent All-Pairs Field Transforms (RAFT) [26] is a novel optical flow estimation method. RAFT tackles the challenge of accurately estimating dense optical flow by leveraging a recurrent neural network architecture that explicitly models the interactions between all pairs of pixels in the image. This approach enables RAFT to achieve state-of-the-art performance on benchmark datasets, surpassing existing optical flow methods.

2.3.2.1 Architecture of RAFT

The RAFT's architecture consists of a recurrent neural network that iteratively refines the optical flow estimation. The network takes two consecutive frames as input and generates an initial flow estimate, which is then refined through multiple iterations. Each iteration of the network consists of a series of recurrent computations, where the network processes information from all pairs of pixels in the image to refine the flow estimation.
The recurrent computations in RAFT are based on the concept of all-pairs field transforms, which capture the interactions between pixel pairs. By considering the interactions of all pairs of pixels, RAFT can effectively model complex motion patterns, handle occlusions, and estimate accurate flow vectors even in the presence of large displacements.

Chapter 3

PROJECT MANAGEMENT PLAN

3.1 Overview

The Project Management Plan chapter provides an overview of the project management approach utilized in this thesis. The focus is on employing the Scrum model and completing the project within 14 weeks, divided into 14 sprints. The chapter covers project organization, sprint planning, execution, review, and retrospective processes. It also addresses communication, reporting, and risk management strategies. The plan aims to ensure efficient project execution, stakeholder engagement, and timely delivery of project objectives within the allocated time frame.

3.2 Planning

3.2.1 Sprints

Table 3.1 Sprints plan

Sprint	Tasks
Sprint 1	• Identify the research topic • Plan for the research topic • Investigate relevant papers • Experiment with related models

Sprint	Tasks
Sprint 2	• Conduct research on baseline models. • Evaluate the foundational models. • Implement training code using PyTorch. • Initialize the training environment.
Sprint 3	• Investigate the dataset, data preparation techniques, and data preprocessing methods • Proceed with the implementation and training of the model • Conduct model experimentation • Design a data flow diagram
Sprint 4	• Proceed with the training of the AI model • Examine and draw conclusions regarding the results of the new model • Propose improvements for the next training iteration • Initiate report writing
Sprint 5	• Proceed with the implementation, enhancement, and training of the model • Improve data preprocessing techniques • Evaluate the model on the test set • Write a report for the research topic • Address encountered errors • Consider recommendations and feedback from reviewers

Sprint	Tasks
Sprint 6	• Introduce a new approach with changes in the model architecture • Train the model using the new approach • Modify data processing modules (change optical flow module, change data preprocessing module) • Illustrate the new architecture diagram of the model • Refine and optimize the training source code • Continue writing the report
Sprint 7	• Test and evaluate the model in Approach 1 • Test and evaluate the model in Approach 2 • Propose the next steps to improve the approaches • Rent additional training servers • Write code for automated deployment of training on the training servers • Continue writing the report
Sprint 8	• Continue training the model in different approaches with additional data and experiment with varying model hyperparameters • Augment the training data • Test the newly concluded model versions • Enhance the quality of the training dataset by filtering out noisy images • Continue writing the report
Sprint 9	• Continue experimenting with training using different hyperparameters • Optimize the training time of the model by employing distributed data-parallel training • Conduct a survey on the model's results • Send out survey invitations via email • Continue writing the report

Sprint	Tasks
Sprint 10	• Train the AI model • Continue experimenting and improving the data transfer speed on the server • Prepare presentation slides • Continue writing the report
Sprint 11	• Deploy the model • Create a user interface for the model and deploy it on HuggingFace • Write the report • Summarize the survey results
Sprint 12	• Prepare a presentation • Conduct a trial presentation • Demonstrate the product • Edit the report
Sprint 13	• Edit the presentation • Conduct a trial presentation • Demo the product • Edit the report
Sprint 14	• Revise the presentation • Perform a trial presentation • Demonstrate the product • Edit the report

3.2.2 Detailed tasks

Table 3.2 Detailed tasks by members

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Task 1	System construction (main)	Do research (main)	Document writing (main)
Task 2	Do research (secondary)	Document writing (secondary)	Do research (secondary)
Task 3	Comprehensive inspection and document writing (secondary)	System construction (secondary)	System construction (secondary)
Sprint 1	• Formulate the research idea and conduct the study • Identify the difficulties and challenges • Develop a plan for the research topic • Assign tasks to team members • Finalize the applied methodology • Establish an environment for testing the baseline model	• Choose a topic for the project. • Read and summarize relevant research papers. • Read and condense the source code of the base model.	• Propose solutions for the project. • Build the development environment for the model. • Gather relevant research papers. • Evaluate the base model on common metrics.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 2	• Write PyTorch training code • Research methods to enhance the model • Write source code for additional metrics	• Run experiments and evaluate existing models. • Inspect and analyze the training data. • Set up the machine and deploy the model training. • Run experiments and evaluate existing models. • Inspect and analyze the training data. • Set up the machine and deploy the model training.	• Draw the pipeline flow for the base model. • Write code to preprocess the input data for the model. • Fine-tune the training parameters of the model.
Sprint 3	• Collect and search for relevant training data • Filter out inappropriate videos from the dataset • Preprocess the data • Train the model	• Create a diagram representing the model architecture. • Read additional research papers. • Set up WandB to log the metrics.	• Prepare the test dataset. • Evaluate the trained model. • Compare the results with other models.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 4	• Set up parallel training to accelerate the process • Apply additional data augmentation methods	• Modify the model architecture diagram. • Add dropout to the model. • Adjust the hyperparameters. • Change the sampling ratio.	• Check the colorization results of the model to make assumptions and propose improvement strategies. • Conduct A/B testing to decide on the padding and scaling methods used to adjust image sizes. • Create a storage repository on Google Cloud to enable faster data retrieval from the training server. • Evaluate the developed model in this sprint on the DAVIS dataset. • Compare this model with other models using the same metrics.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 5	• Augment the dataset • Resolve padding issues in optical flow • Compare this model with other models based on the same metrics • Continue refining the report • Integrate training graphs into WandB for monitoring • Reduce the complexity of the model • Select a platform or framework for deploying the model in future sprints	• Find ways to address the limitations of PyTorch data parallel when encountering out-of-memory (OOM) errors. • Evaluate the developed model in this sprint on the Davis dataset. • Continue refining the report. • Modify the parameters. • Find ways to provide a more detailed explanation of the project name during presentations.	• Check the colorization results of the model to make assumptions and propose improvement strategies. • Consider upgrading the Google Cloud data repository to a multi-region setup to ensure stability in data download and training deployment. • Modify the model architecture diagram. • Continue refining the report. • Provide a more practical application scenario for the project.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 6	• Propose a new approach with changes in the model architecture • Modify the optical flow module and regenerate relevant files associated with this module • Compile a report summarizing the improvements in applying the new optical flow module • Refine and optimize the new training code • Continue writing the report	• Write training code for the new approach. • Test and debug the training code for any issues. • Train the model using the new approach. • Modify the architecture diagram for the new model.	• Modify and optimize the training code for the previous approach. • Apply FP16 data type to the output files of the optical flow module. • Train the model using the previous approach. • Create a diagram representing the new model architecture. • Continue writing the report.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 7	• Test and evaluate the model results using the second approach • Acquire additional training servers • Develop automated deployment code for training on the new server • Upload the training data to a cloud platform to facilitate the training process on the new server	• Propose solutions based on the testing results of model 2. • Test the automatic training deployment code on the server. • Modify the training code based on the proposed solutions. • Set up a cloud-based platform to serve as the storage for training data.	• Test and evaluate the model in approach 1. • Propose solutions based on the testing results of model 1. • Modify the training code based on the proposed solutions. • Write and revise the report to reflect the latest updates.
Sprint 8	• Search and gather additional training data sources • Experiment with training the model using approach 1 with different sets of hyperparameters • Filter and enhance the quality of the supplemented dataset • Upload and append the new data to the cloud platform	• Set up the training source code with the new approach on the server. • Troubleshoot any errors during the training process of the new source code.	• Train the model using approach 2 with different sets of hyperparameters. • Filter out noisy images in the new dataset. • Evaluate the testing results of the model versions and propose the next optimal solution. • Write a supplementary report on the new dataset.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 9	• Continue experimenting with training using different sets of hyperparameters • Modify the source code to optimize training time through parallel data distribution methods	• Set up the training source code with the new approach on the server. • Troubleshoot any errors during the training process of the new source code.	• Continue writing and editing the report, addressing any errors or inconsistencies. • Evaluate the models on the test set, analyzing their performance and comparing the results.
Sprint 10	• Run the models to generate results for the group's survey submission • Send the survey to relevant participants • Prepare presentation slides	• Create a survey regarding the model's results. • Send the survey to appropriate recipients. • Prepare the presentation slides.	• Run the old models from other research papers to generate results for the survey. • Edit the report to include information about the survey. • Prepare the presentation slides.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 11	• Develop a user interface using Gradio • Compose an email requesting the use of Huggingface's GPU resources • Send an email inviting participants to complete the survey	• Send an email inviting participants to fill out the survey form. • Write a program to aggregate the survey results. • Include the survey results in the report.	• Send an email inviting participants to fill out the survey form. • Edit the report as needed.
Sprint 12	• Find a design template for the presentation. • Assign different sections to team members for the presentation. • Work on the presentation based on the assigned sections.	• Prepare the presentation. • Edit the report as needed. • Redesign the diagram.	• Prepare the presentation. • Edit the report as needed. • Redesign the diagram.

Task \ Member	**D.T.Tran** SE160185	**N.D.H.Nguyen** SE161261	**T.T.Pham** SE160030
Sprint 13	• Modify the Hugging Face interface • Conduct a test presentation • Edit the presentation based on feedback and improvements	• Adjust the layout of the report. • Add an appendix section to the report. • Conduct a trial presentation. • Edit the presentation based on the trial.	• Adjust the layout of the report. • Add an appendix section to the report. • Conduct a trial presentation. • Edit the presentation based on the trial.
Sprint 14	• Summarize the report • Conduct a trial presentation	• Summarize the report • Conduct a trial presentation	• Summarize the report • Conduct a trial presentation

Chapter 4

MATERIALS AND METHODS

4.1 Materials

4.1.1 Project Management Tool

The utilization of Google Sheets as a versatile project management tool for organizing and tracking project progress, particularly within the framework of the Scrum model mentioned in Chapter 3. This platform provides us with a collaborative and customizable system, enabling effective planning, monitoring, and resource allocation for each sprint. By leveraging the features of Google Sheets, such as spreadsheets and real-time collaboration, we significantly enhance our project management efficiency, ultimately ensuring the successful execution of our Scrum-based project.

4.1.2 Hardware

4.1.2.1 Google Colab

Google Colab is a cloud-based integrated development environment (IDE) that provides researchers and developers with a convenient platform for running and executing machine learning tasks. One of the notable features of Google Colab is the provision of free GPUs, specifically the NVIDIA Tesla T4 GPUs, which can significantly accelerate deep learning model training. These free GPUs are made available to users for a limited duration of time, allowing them to harness the computational power required for training complex models without the need for expensive hardware. In the initial research phase, we leveraged the cost-saving benefits and testing capabilities of Google Colab, utilizing a free T4 GPU. This allowed us to explore the training process and conduct preliminary experiments. However,

due to the limitations in GPU quota and computational power, we encountered challenges in performing complete training runs. As a result, we have to find other other alternatives.

4.1.2.2 Kaggle Notebook

Kaggle is another platform that provides researchers and data scientists with access to powerful computational resources for machine learning and data science tasks. These GPUs (1 NVIDIA Tesla P100 GPU and 2 NVIDIA Tesla T4 GPUs), available through Kaggle's kernels, enable accelerating experiments and tackling complex problems by leveraging the platform's robust infrastructure. However, GPU availability and usage time on Kaggle may be limited due to high demand (30 hours per week). Therefore, we use Kaggle for the inference stage, where we get the colorization results for the evaluation.

4.1.2.3 Personal Computer

During the later training phase, we encountered limitations with the computational power and usage quotas provided by platforms such as Colab and Kaggle. As our model's architecture grew, we needed a more powerful hardware setup to accelerate the training process. Consequently, we made the decision to leverage a personal computer equipped with an NVIDIA GeForce RTX 3090 GPU. This hardware choice allowed us to enhance the training speed and efficiency, enabling us to handle the increasing complexity of our models effectively. By utilizing this upgraded hardware, we were able to overcome the previous limitations and continue training our models with improved performance and scalability.

4.1.2.4 Vast.ai

Vast.ai is a leading platform for low-cost cloud GPU rental, making powerful GPU computing accessible to all. It saves 5-6 times on GPU costs with our intuitive platforms.

During training model, we realized the limitations of using a personal computer not optimized for AI training. To overcome this, we rented dedicated training servers from Vast.ai. We utilized instances with powerful GPUs such as NVIDIA GeForce RTX 4090, RTX A6000, and RTX 3090, among others. This transition to dedicated servers accelerated our training and experimentation, allowing us to handle the increased complexity and scale of our models more effectively.

4.1.3 Data Management Platforms

4.1.3.1 Kaggle

Kaggle is a platform offering a robust data storage feature that provides a reliable and convenient solution for storing and sharing large-scale datasets. With the integration of the Kaggle API, we leverage this feature to access a generous allocation of 100GB of free storage space for our datasets. This capability simplifies our data management process and enables us to securely store and share datasets with ease. By utilizing Kaggle's data storage feature, we ensure the availability, accessibility, and scalability of our data, enhancing the efficiency and effectiveness of our data-driven projects.

4.1.3.2 Google Cloud Platform

Google Cloud Platform (GCP) Storage is a secure and scalable cloud storage solution that provides various storage classes to meet different performance, availability, and cost requirements. We chose GCP Storage over Kaggle due to faster download speeds and the ability to overcome download quotas. With a dedicated storage server in Western Europe, we ensure efficient data transfer to Vast.ai's training servers, which reduces wasted time and saves money. GCP Storage offers scalability, reliability, and performance for storing and accessing datasets in our machine-learning projects.

4.1.4 Framework and Libraries

4.1.4.1 PyTorch

PyTorch is a popular open-source machine learning framework known for its flexibility and dynamic computational graph. It offers efficient model development, GPU acceleration, and a rich ecosystem of tools and libraries. It is widely used by researchers and practitioners for building and training neural networks.

In this research, PyTorch was selected as the preferred framework for several reasons. Firstly, PyTorch demonstrates superior flexibility in terms of customization capabilities, allowing for more extensive model adaptations. Secondly, the utilization of PyTorch aligns with the use of a baseline model implemented in the same framework, facilitating seamless integration and comparison. Lastly, the decision to opt for PyTorch was influenced by the limited ongoing development efforts observed in TensorFlow, thereby emphasizing the relevance and continued advancements within the PyTorch ecosystem.

4.1.4.2 Transformers

The **transformers** package in Python, developed by Hugging Face, is a versatile and user-friendly tool for natural language processing and computer vision tasks. This package offers a wide range of functionalities, including easy-to-use pretrained models and seamless integration into various applications. With the **transformers** package, researchers and practitioners can leverage state-of-the-art pretrained models such as BERT [27], GPT [28], and ViT [5] for their NLP and computer vision projects. Installing pretrained models from the **transformers** package is straightforward, requiring minimal effort and ensuring quick access to powerful pretrained models. This ease of installation allows users to rapidly incorporate and fine-tune pretrained models in their workflows, enabling efficient experimentation and advancing the state-of-the-art in NLP and computer vision domains.

4.1.5 Visualization and Tracking tools

Weights and Biases (WandB) is an advanced platform that simplifies the training and monitoring of machine learning models. It offers comprehensive logging, visualization, and collaboration features, seamlessly integrating with popular frameworks. Users gain insights through interactive dashboards and can ensure reproducibility with robust experiment tracking. WandB accelerates model development, fosters collaboration, and promotes informed decision-making in machine learning projects. In this research, WandB is used for tracking the performance of the currently trained model by plotting the graph and sending messages to the Slack group. See more in Appendix B

4.2 Method

4.2.1 Overview

Based on the end-to-end network proposed by Zhang *et al.* [19], this research proposes two novel methods for video colorization. Instead of utilizing VGG19 [29] as employed in the aforementioned model, alternative backbones such as ViT [5] and its variants are explored. The field of Computer Vision is currently witnessing the emergence and prevalence of ViT techniques, which have demonstrated remarkable performance. Consequently, employing ViT and its variants holds promise as a potential approach. This research introduces two novel end-to-end models, each employing a distinct backbone as the feature extractor for subsequent processing. The first model, referred to as ViTExCo, employs the pretrained ViT with 12 attention-based blocks [21]. The second model, known as SwinTExCo, employs the

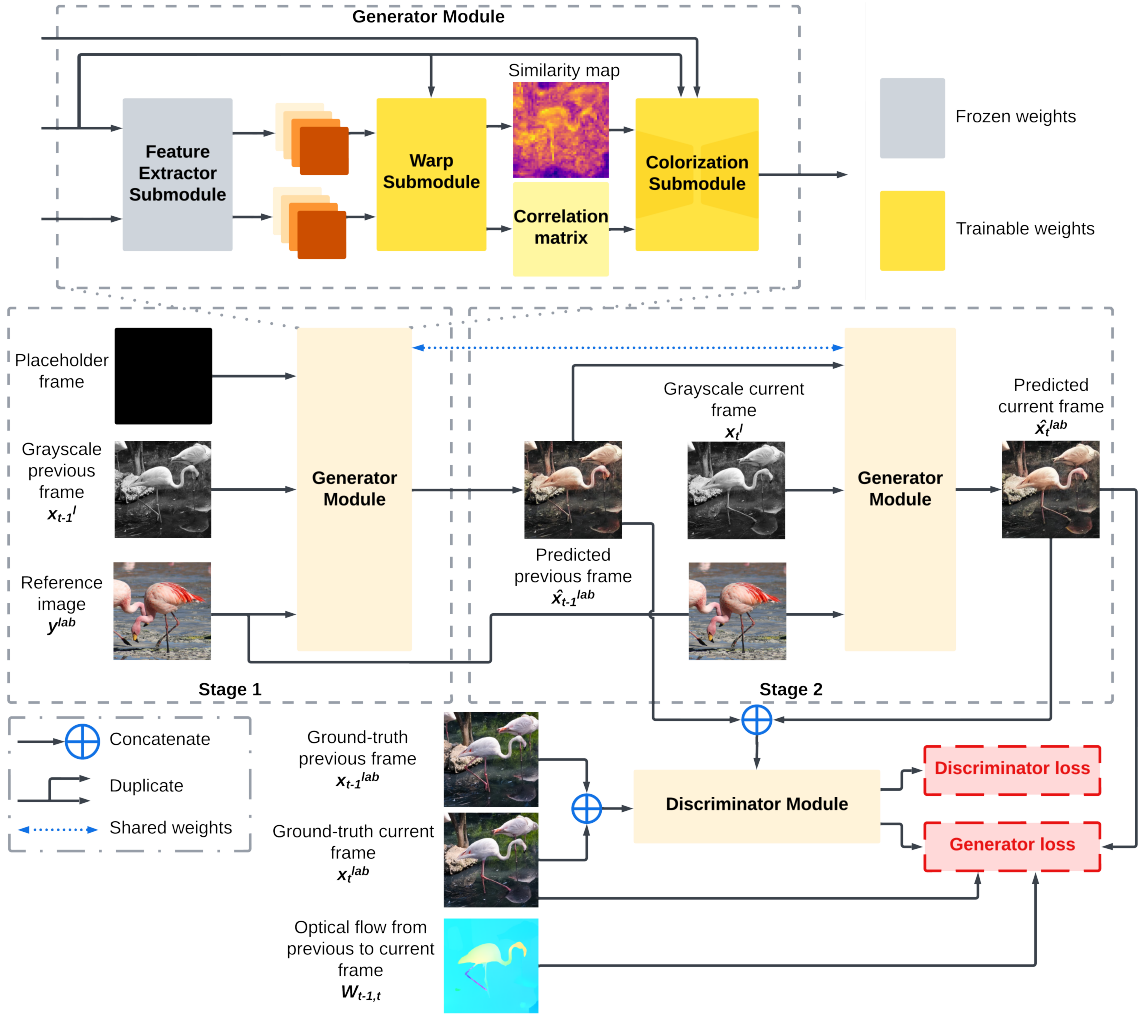


Fig. 4.1 The overall architecture of ViTExCo and SwinTExCo

pretrained SwinT [25], which incorporates improvements in processing speed and scalability by utilizing the Hierarchical and Shifted Windows techniques. This section provides an overview of the ViTExCo and SwinTExCo architectures, as depicted in Figure 4.1.

4.2.2 Proposed methods

4.2.2.1 Overall network

In this research, we proposed two end-to-end video colorization models, namely ViTExCo and SwinTExCo. They have the same overall architecture and both use the exemplar-based colorization approach. Each model comprises two main components: the **Generator module** and the **Discriminator module**. The Generator module consists of three distinct submodules, each serving a specific purpose. The first submodule, namely the **Feature Extractor submodule**, is responsible for extracting features from both the video frame and the reference image. Leveraging the ViT architecture in ViTExCo and the SwinT architecture in SwinTExCo, this submodule effectively captures relevant visual information. The second submodule, referred to as the **Warp submodule**, generates a Correlation Matrix $\mathscr{M}$ and a Similarity Map $\mathscr{S}$ based on the extracted features from the Feature Extractor submodule. This component is crucial for establishing meaningful relationships between the video frame and the reference image. Finally, the **Colorization submodule**, which constitutes the core of the colorization process, utilizes the Correlation Matrix $\mathscr{M}$ and Similarity Map $\mathscr{S}$ to effectively colorize the video frames. On the other hand, the **Discriminator module** plays a pivotal role in evaluating the quality of the colorization process performed by the Generator module. Through an adversarial training scheme, the Discriminator module provides feedback to guide and refine the colorization process, ensuring the generation of visually pleasing and realistic colorized videos.

Our approach employs the LAB color space instead of RGB or alternative color spaces. This choice is motivated by the fact that the colorization process only needs to adjust two channels, namely a and b. In LAB color space, the l channel represents perceptual lightness, exhibiting values within the $[-100, 100]$ range. The a channel corresponds to the red/green value, ranging from $[-128, 128]$. Similarly, the b channel represents the blue/yellow value, spanning the $[-128, 128]$ range. By utilizing the LAB color space, ViTExCo and SwinTExCo can simplify the colorization process while simultaneously preserving the contextual information of the video frames.

Figure 4.1 provides a comprehensive overview of the architectural design employed in this research. The input video, denoted as $X^l = \{x_t^l\}$, represents a grayscale video consisting of T frames, where $t = \overline{1,T}$, denotes the t^{th} video frame. Each frame, $x_t^l \in \mathbb{R}^{H \times W \times 1}$, is

represented by a three-dimensional tensor, where H and W correspond to the height and width of the frame, respectively. These dimensions reflect the spatial resolution of the video, providing information about the size and aspect ratio of the frames.

The main objective of this research is to generate a colorized video, denoted as $\hat{X}^{lab} = \{\hat{x}_t^{lab}\}$, where $t = \overline{1,T}$. Each frame, $\hat{x}_t^{lab} \in \mathbb{R}^{H\times W\times 3}$, represents the colorized version of the t^{th} video frame within the *LAB* color space.

The colorization process relies on two crucial components: a reference color image, denoted as $y^{lab} \in \mathbb{R}^{H\times W\times 3}$, and the previous colorized frame, x_{t-1}^{lab}. The reference color image provides guidance and serves as a consultation for the desired colorization outcome. By incorporating the previous colorized frame, the model can leverage temporal dependencies and ensure consistency in colorization throughout the video.

Through the utilization of the architectural design outlined in Figure 4.1, this research aims to achieve accurate and visually appealing colorization of grayscale videos. Integrating the reference image and the previous colorized frame as guiding references enhances the model's ability to capture temporal dependencies and produce coherent colorization results.

4.2.2.2 Generator module

4.2.2.2.1 Feature Extractor submodules (FES) This aforementioned submodule extracts features from both the video frame and the reference image, which are subsequently reserved for future utilization. Two distinct variants of Feature Extractor submodules are implemented, namely the ViT Extractor submodule and the SwinT Extractor submodule. These variants are specifically designed for integration within the ViTExCo and SwinTExCo frameworks, respectively. Each variant operates by accepting either a video frame or a reference image as input and conducts a process of feature extraction. As a result of this extraction process, four tensors are generated, effectively representing the extracted features corresponding to the specific video frame or image.

ViT Extractor submodule (VTES): The ViT Extractor submodule represents a pre-trained ViT model [5] that has been designed to extract features from video frames or reference images. Specifically, it focuses on extracting features from the ViT blocks positioned at indices 5, 7, 9, and 11, as illustrated in Figure 4.2. Consequently, it produces four distinct feature tensors, each capturing pertinent information at different hierarchical levels. These feature tensors serve as valuable representations of the input data, facilitating the effective transfer of colors from the reference image to the corresponding video frame by subsequent submodules.

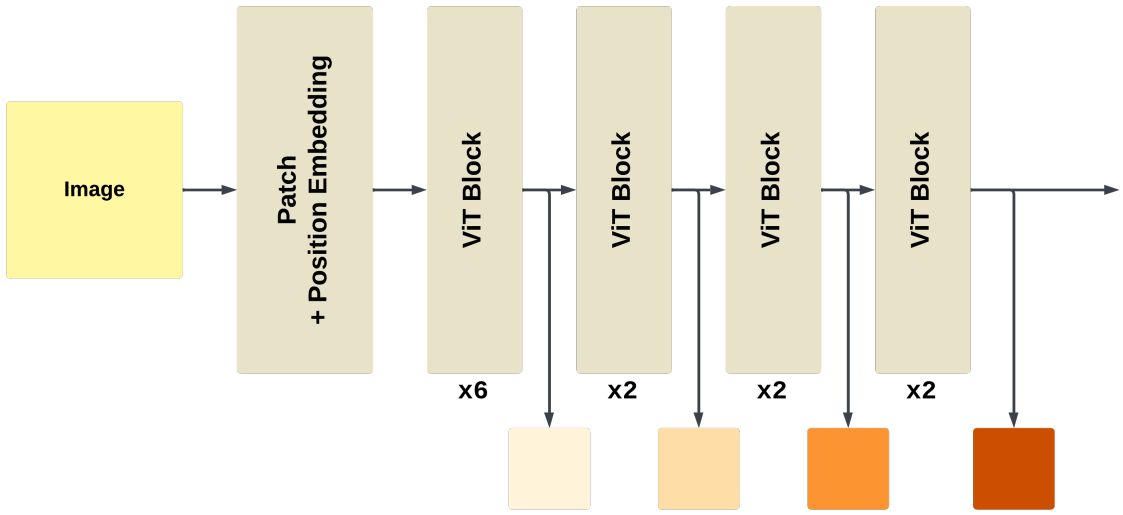


Fig. 4.2 The architecture of the ViT Extractor submodule (VTES)

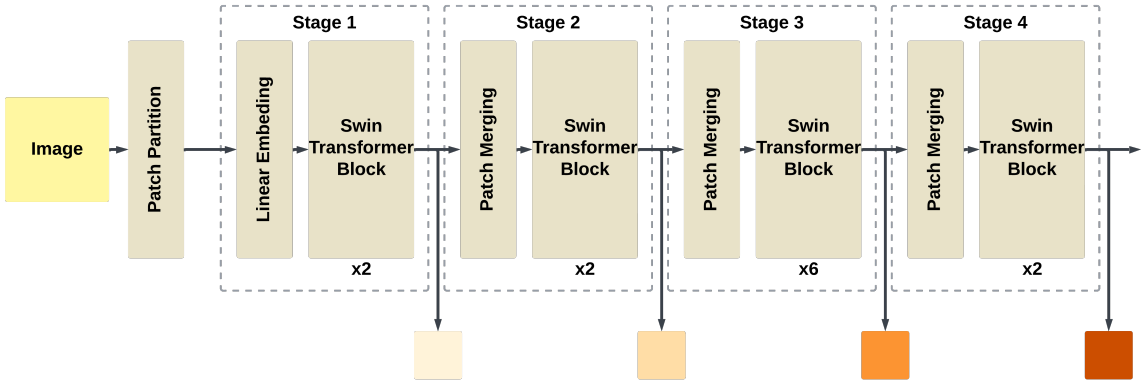


Fig. 4.3 The architecture of the SwinT Extractor submodule (STES)

SwinT Extractor submodule (STES): The SwinT Extractor submodule, presented in Figure 4.3, is of a pretrained SwinT model [25]. Similar to the functional capability of the ViT Extractor submodule, it is employed to extract features from video frames or reference images. The architecture of the SwinT comprises four stages, each with a distinct configuration. This arrangement enables the extraction of features at various levels, ranging from local to global, from the input image or video frame. Exploiting this characteristic, we obtain four outputs from the four stages and convert them into four feature tensors for utilization by subsequent submodules.

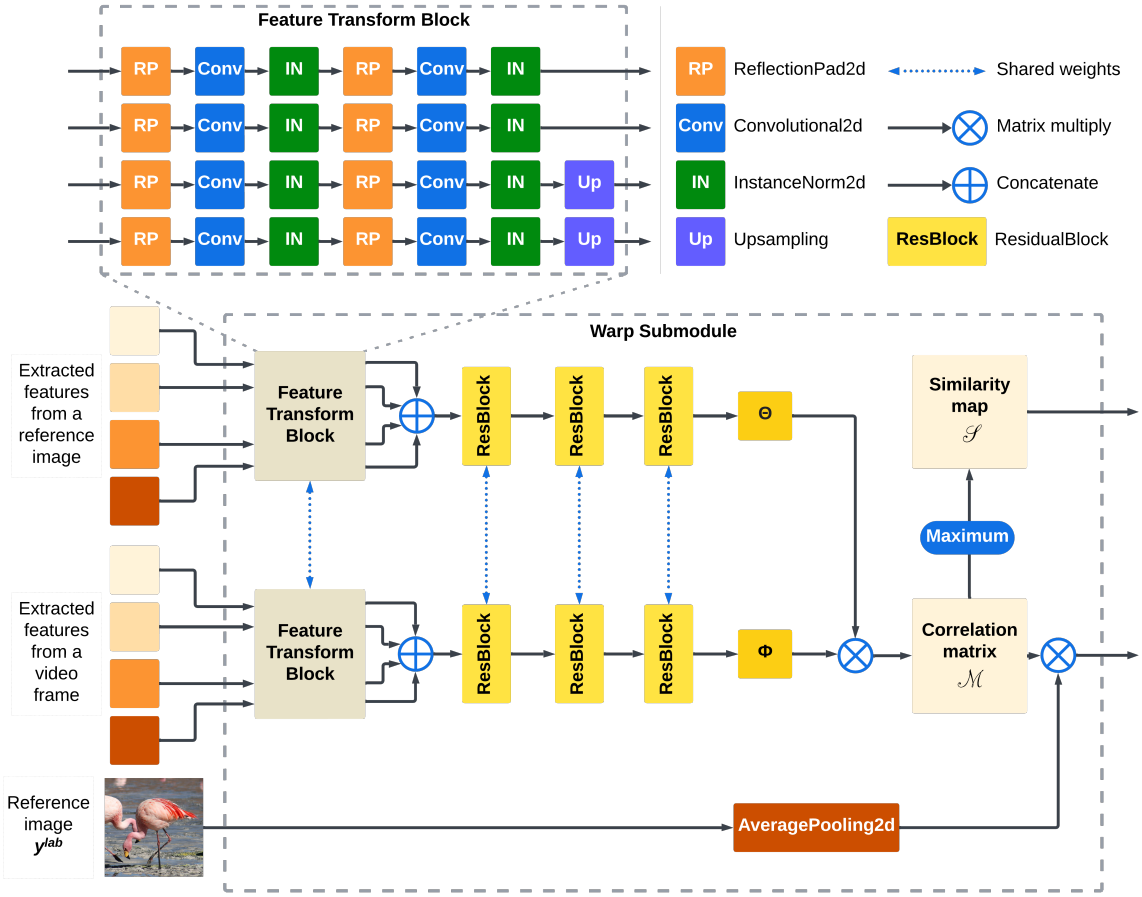


Fig. 4.4 The architecture of the Warp submodule (WS)

4.2.2.2.2 Warp submodule (WS) The Warp submodule, depicted in Figure 4.4, assumes a critical role in the colorization process. It is a trainable CNN model [3] responsible for determining the similarity between the reference image and the currently colorized video frame. This similarity information is utilized by subsequent modules to accurately transfer color from the reference image to the video frame.

To commence, the Warp submodule receives 2 inputs of 4 feature tensors extracted from the Feature Extractor submodule (ViT Extractor or SwinT Extractor). These tensors are obtained from the reference image and the presently colorized video frame, respectively. The feature tensors undergo a Feature Transform block to be reshaped into a suitable format for concatenation. The Feature Transform block is a CNN block of multiple layers, consisting of Reflection Padding, Convolutional, Instance Normalization, Upsampling, etc. Subsequently, the transformed features pass through several Residual blocks to undergo deeper transformations, resulting in the generation of two tensors: Θ for the reference image and Φ for the grayscale video frame. These tensors are then employed to compute the Correlation Matrix $\mathscr{M} \in \mathbb{R}^{HW \times HW}$ through a matrix multiplication operation. The Correlation Matrix captures the relationships between the features of the video frame and the reference image, providing insights into their similarity and dissimilarity. Furthermore, the Warp submodule computes the Similarity Map $\mathscr{S} \in \mathbb{R}^{H \times W}$ from the Correlation Matrix $\mathscr{M}$. This is achieved by applying the Maximum operator to the Correlation Matrix, resulting in a map where each element represents the maximum correlation between corresponding spatial locations in the video frame and the reference image. The Similarity Map serves as a representation of the spatial correspondence and similarity between the two input sources.

In summary, the Warp submodule utilizes the feature tensors extracted from the reference image and the video frame as input to generate the Correlation Matrix $\mathscr{M}$ and the Similarity Map $\mathscr{S}$. These outputs play a critical role in facilitating the accurate transfer of color between the two sources.

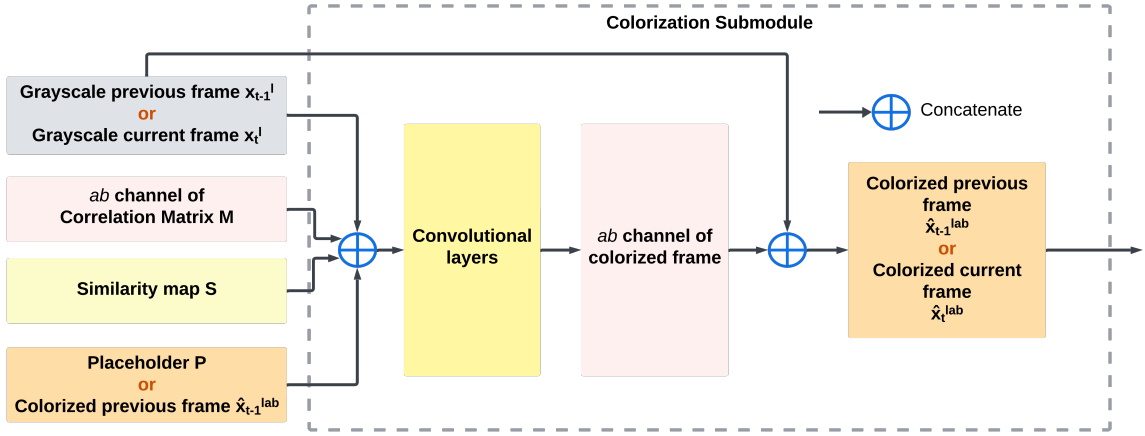


Fig. 4.5 The architecture of the Colorization submodule (CS)

4.2.2.2.3 Colorization submodule (CS) The Colorization submodule, depicted in Figure 4.5, serves as a pivotal component within the overall model, responsible for executing the

colorization process. It is a trainable CNN model [3], consisting of multiple layers, including Convolutional layers, Normalization layers, and activation functions. In this research, this submodule is employed to colorize the current video frame based on its colorized previous frame. This mechanism aims to adjust the colors in the predicted current frame, ensuring consistency in coloration across consecutive frames.

The colorization process within this submodule encounters two distinct scenarios as presented in Figure 4.5. These scenarios are distinguished between the colorization of the first frame in a pair of adjacent frames and the colorization of the latter. Both situations will be described in more detail in Section 4.2.2.4. However, in any scenario, the flow of a data sample in each iteration will be the same. The Colorization will get 4 inputs, consisting of the grayscale video frame (previous or current), *ab* channels of Correlation Matrix $\mathscr{M}$, Similarity Map $\mathscr{S}$, and the Placeholder P or the colorized previous frame $\hat{x}_{t-1}^{lab}$. These inputs will be transformed to a suitable shape for concatenation before being passed through several Convolutional layers to predict the *ab* channel of the presently colorized frame. This result will be concatenated with the grayscale input to produce the final result (colorized previous frame $\hat{x}_{t-1}^{lab}$ or colorized current frame $\hat{x}_{t}^{lab}$)

To summarize, the Colorization submodule receives the grayscale video frame x^{l}, the Correlation Matrix $\mathscr{M}$, the Similarity Map $\mathscr{S}$, and another input based on the situation. It then generates the colorized video frame $\hat{x}_{t-1}^{lab}$ or $\hat{x}_{t}^{lab}$ as its output.

4.2.2.3 Discriminator module

The input tensors provided to the Discriminator module are formed by concatenating a pair of consecutive video frames. Specifically, the predicted previous video frame $\hat{x}_{t-1}^{lab}$ and the predicted current video frame $\hat{x}_{t}^{lab}$ are concatenated, as well as the ground truth previous video frame x_{t-1}^{lab} and the ground truth current video frame x_{t}^{lab}. These concatenated tensors capture the temporal relationship between consecutive frames and serve as input to the Discriminator module.

Differentiating between colorized video frames and ground truth video frames is a critical step in assessing the effectiveness of the colorization process. To perform this assessment, the Discriminator module, depicted in Figure 4.1, is utilized. The Discriminator module is a model designed for distinguishing two consecutive video frames as input, whether they are colorized consecutive frames or ground truth consecutive frames. The output of the Discriminator module is used to calculate the Discriminator Loss and the Adversarial Loss, which are described in detail in Section 4.2.3. These loss functions quantify the similarity or dissimilarity between the predicted video frames and the ground truth video frames as

well as the reference image. By comparing the colorized frames to them, the Discriminator module assesses the realism and fidelity of the colorization process.

In essence, the Discriminator module plays a pivotal role in judging the quality of the colorized video frames. It serves as a discriminator between the predicted frames and the ground truth frames, distinguishing between realistic and unrealistic colorizations. This assessment is crucial in guiding the optimization of the colorization model and ensuring that the generated video frames closely resemble the ground truth data while transforming the information from reference images into the generated video frames.

4.2.2.4 Training process

4.2.2.4.1 Reference image selection The procedure for selecting the reference image is intricate. Firstly, the current grayscale video frame is compared with the images in the ImageNet-1K dataset [30] using the Cosine similarity measure, which leverages features extracted from a pretrained SwinT model [25]. The top 5 images with the highest similarity scores are selected as **real-reference** images. From this set, one image is randomly chosen with uniform probability. The selection of 5 images, rather than just 1, aims to diversify the reference and mitigate overfitting, aligning with the approach employed by Zhang *et al.* [19] for fair comparison purposes. Empirical evidence suggests that utilizing 5 **real-reference** images yields superior performance compared to alternative choices. Furthermore, 2 adjacent ground truth frames, consisting of the ground truth of the presently colorized frame and its ground truth previous frame will be selected as **ground-truth-reference** images. One image is randomly selected with uniform probability from these two images. The final reference image is determined by randomly selecting either the **real-reference** image or the **ground-truth-reference** image, with different probabilities assigned to each.

This entire process ensures consistency in the colorization process across frames, while the designated ratio prevents overfitting and discourages the model from simply learning the exact colors of the ground truth.

4.2.2.4.2 An iteration of training In every iteration of the training procedure, as illustrated in Figure 4.1, the colorization task is partitioned into two stages. Each stage is responsible for colorizing a frame within a pair of adjacent video frames (denoted as x^l_{t-1} and x^l_t), where the input of the model for colorizing the later video frame is the colorized result of the sooner frame. This approach is employed to ensure that these frames possess coherent and continuous colors, thereby mitigating potential flickering artifacts. To initiate the colorization process, a reference image is selected for each video frame. Considering that the training dataset encapsulates a single coherent scene throughout an entire video, the same reference

image is consistently chosen for all frames within a video during each training iteration. In an iteration, there are 2 situations as mentioned in Section 4.5, namely the colorization of the first frame in a pair of adjacent frames and the colorization of the latter. These 2 situations lead to the differences in 2 stages of an iteration because there is no previous frame for the first frame of the pair. In that situation, a black image called Placeholder $\mathscr{P}$ will be used as an alternative input. This selection serves two primary purposes. Firstly, it provides a simplified approach that requires minimal additional programming or processing steps, thereby enhancing efficiency, particularly when dealing with extensive datasets. Secondly, using a black image as the preceding frame ensures the model's equilibrium during the normalization of all video frames. In most cases, adopting a black image proves to be a sensible and pragmatic approach.

The training iteration begins with the first stage after the reference image selection process. The Placeholder $\mathscr{P}$, as defined previously, is combined with the grayscale previous frame x^l_{t-1} and the reference image y^{lab} as the inputs. These inputs are fed into the Generator module, resulting in the colorized previous frame $\hat{x}^{lab}_{t-1}$. In this stage of this situation, the Colorization submodule uses the grayscale previous frame x^l_{t-1} and the Placeholder $\mathscr{P}$ as inputs. Subsequently, in the second stage, the output from the first stage ($\hat{x}^l ab_{t-1}$) is supplied to the Generator module along with the grayscale current frame represented as x^l_t and the reference image y^{lab}. This configuration enables the generation of the colorized current frame $\hat{x}^{lab}_t$. The outcomes obtained from the two stages of the generation process are evaluated by the Discriminator module. The input to the Discriminator module is formed by concatenating the previous and current frames, as illustrated in Figure 4.1 and described in 4.2.2.3. Backward propagation then occurs, facilitating the advancement of the training process to the subsequent iteration.

4.2.3 Loss functions

The training process of our model incorporates two distinct losses, namely the Generator Loss and the Discriminator Loss, as illustrated in Figure 4.1. The Generator Loss is utilized to optimize the parameters of the Generator module (Section 4.2.2.2) through backpropagation. Conversely, the Discriminator Loss is employed during the backpropagation of the Discriminator (Section 4.2.2.3) to refine its ability to distinguish. It is worth noting that when backpropagating the Discriminator, the parameters of the Generator are frozen to ensure a focused learning process. In the following sections, we will delve deeper into the intricate workings of these two losses, providing a comprehensive understanding of their individual roles in enhancing the overall performance of our model.

4.2.3.1 Generator Loss

4.2.3.1.1 L1 Loss The L1 Loss function measures the color discrepancies between a colorized video frame $\hat{x}_t$ and its corresponding ground truth frame x_t. By optimizing the Generator module on this loss function, the video colorize model can generate more accurate predictions that closely approximate the appearance of the ground truth video frames. The detailed mathematical expression for L_{L1} is provided below:

$$L_{L1} = ||\hat{x}_t^{lab} - x_t^{lab}||_1 \tag{4.1}$$

4.2.3.1.2 Adversarial Loss In order to enhance the realism of the colorized videos, Adversarial Loss is integrated as a constraint for the training process. Diverging from the conventional use of an image discriminator, a video discriminator is employed, following the approach adopted in Deep Exemplar-based Video Colorization [19]. This strategic decision stems from the video discriminator's capability to identify potential flickering in the colorized videos, distinguishing them from ground truth footage. By leveraging the video discriminator, the model can acquire the ability to colorize videos while minimizing flickering concerns and ensuring the verisimilitude of the colorized video frames. This approach effectively elevates the quality and visual realism of the colorized videos. The mathematical expression for L_{adv} is provided below:

$$\begin{aligned} L_{adv} =& \mathbb{E}_{(\hat{x}_{t-1},\hat{x}_t)}\left[\left(D(\hat{x}_{t-1},\hat{x}_t) - \mathbb{E}_{(x_{t-1},x_t)}D(x_{t-1},x_t) - 1\right)^2\right] \\ &+ \mathbb{E}_{(x_{t-1},x_t)}\left[\left(D(x_{t-1},x_t) - \mathbb{E}_{(\hat{x}_{t-1},\hat{x}_t)}D(\hat{x}_{t-1},\hat{x}_t) + 1\right)^2\right] \end{aligned} \tag{4.2}$$

The expression $D(x_{t-1},x_t)$ refers to feeding two consecutive frame x_{t-1} and x_t into the Discriminator module 4.2.2.3. The output of $D(x_{t-1},x_t)$ is a floating-point value that falls within the range of [0,1]. This scalar signifies the probability that the input frames are ground truth frames.

4.2.3.1.3 Perceptual Loss The Perceptual Loss, originally introduced in [31], is leveraged to train a neural network in a semantic manner, enabling the transfer of image style from one to another. It quantifies the dissimilarity between the high-level pattern of two images, including edges, textures, and shapes. By incorporating this loss, we ensure the perceptual plausibility of the output. This loss penalizes the semantic discrepancies between the colorized frame $\hat{x}_t$ and the corresponding ground truth frame x_t. The mathematical expression for L_{perc} can be

expressed as follows:

$$L_{perc} = ||\phi_{\hat{x}}^s - \phi_x^s||_2^2 \tag{4.3}$$

where $\phi_{\hat{x}}^s$ represents the s^{th} output features map of the Feature Extrator submodule 4.2.2.2.1 when forwarding the input $\hat{x}$.

4.2.3.1.4 Contextual Loss The Contextual Loss expresses the disparity between the higher-level characteristics of two images, taking into account the overall image context, as well as edges, textures, and shapes. In the work presented in [32], the Contextual Loss was employed to train a neural network in style transfer while preserving the underlying structure. This enables the network to learn how to transform an input image into an output image with the desired features. Given its capability to compare contextual meaning between a video frame and a reference image, the Contextual Loss is well-suited for promoting similarity in colors between the predicted frame $\hat{x}_t$ and the corresponding reference image. Initially, we compute the cosine similarity $d^s(i,j)$ between each pair of feature points $\phi_{\hat{x}}^s(i)$ and $\phi_y^s(j)$, where $\hat{d}^s(i,j) = d^s(i,j)/(min_k d^s(i,k) + \varepsilon)$ and $\varepsilon = 10^{-5}$. The pairwise affinities $A^s(i,j)$ across features are then computed using the following formula:

$$A^s(i,j) = \underset{j}{\text{softmax}}\left(1 - \hat{d}^s(i,j)/h\right) \tag{4.4}$$

The bandwidth parameter $h = 0.1$ was mentioned in [19] determines the range of influence for calculating $A^s(i,j)$. The values of $A^s(i,j)$ range between $[0,1]$, indicating the similarity between $\hat{x}_t(i)$ and $y(j)$ in the features of the s^{th} stage output of FES module 4.2.2.2.1. Building upon the forward matching approach described in [19], we identify the most relevant feature $\phi_{y,j}^s$ in y for each feature $\phi_{\hat{x},i}^s$. In essence, the Contextual Loss aims to maximize the contextual similarity between the predicted frame and the reference frame, leveraging the values of $A^s(i,j)$:

$$L_{ctx} = \sum_s w_s \left[-\log\left(\frac{1}{N_s} \sum_i \max_j A^s(i,j) \right) \right] \tag{4.5}$$

We use 4 feature maps: s = 1 to 4. N_s indicates the feature number of the s^{th} stage from the FES module 4.2.2.2.1. w_s is the suitable weight for s^{th} stage.

4.2.3.1.5 Temporal Consistency Loss To ensure temporal coherence in video colorization, a Temporal Consistency Loss [33] is integrated into the colorization process. This loss function penalizes color changes that occur along the flow trajectory, explicitly considering the consistency of colors over time. By incorporating this constraint, the model strives to

maintain consistent color transitions in video frames, leading to improved visual quality and coherency of the colorized video. This approach effectively addresses issues such as flickering, promoting smoother and more natural-looking color changes. To reinforce temporal consistency in our model, we leverage the RAFT algorithm [26] to generate optical flow information for each frame pair. Additionally, we utilize a method [34] to generate an occlusion mask, which helps identify regions where the flow estimation might be less reliable. By combining these flow and mask components, we calculate the temporal consistency loss, which contributes to enhancing the overall visual quality of the colorized output. The formula for L_{temp} is expressed as follows:

$$L_{temp} = ||m_{t-1} \odot W_{t-1,t}(\hat{x}_{t-1}^{lab}) - m_{t-1} \odot \hat{x}_t^{lab}|| \tag{4.6}$$

where $W_{t-1,t}$ refer to the forward optical flow from the previous frame x_{t-1} to x_t and m_{t-1} is the binary mask and $\odot$ denotes Hadamard product operation.

4.2.3.1.6 Smoothness Loss Smoothness loss was also introduced in [19] to encourage spatial smoothness. In video colorization tasks, it is often assumed that neighboring pixels of the predicted frame $\hat{x}_t$ should have similar chrominance values if they have similar chrominance in the corresponding ground truth frame x_t. A smoothness loss is incorporated into the colorization process to enforce this constraint. This loss function calculates the difference between the current pixel's color and the weighted color of its 8-connected neighborhood. By minimizing this loss, the model can ensure that the color transitions in the video frames are smooth and consistent, thereby improving the overall visual quality of the colorized video. This approach is particularly effective in producing natural-looking color transitions and reducing the occurrence of color artifacts in the colorized video. The formula for L_{smooth} can be expressed as follow:

$$L_{smooth} = \frac{1}{N} \sum_{c \in \{a,b\}} \sum_{i} \left(\hat{x}_t^c(i) - \sum_{j \in \mathbb{N}(i)} w_{i,j} \hat{x}_t^c(j) \right) \tag{4.7}$$

where N is the number of samples in a training step, a and b are 2 channels in LAB color space. $\mathbb{N}(i)$ represents the adjacent pixels of a pixel i. $w_{i,j}$ is weighted least squares (**WLS**) weight indicating correlations between nearby pixels.

4.2.3.1.7 Overall Loss We add all the losses above to create the total loss that we want to optimize

$$L = \lambda_{L1}L_{L1} + \lambda_{adv}L_{adv} + \lambda_{perc}L_{perc} + \lambda_{ctx}L_{ctx} + \lambda_{temp}L_{temp} + \lambda_{smooth}L_{smooth} \quad (4.8)$$

where λ is the weight of each loss function contribution to the total loss formula.

4.2.3.2 Discriminator Loss

The Discriminator Loss is used for the backpropagation process of the Discriminator Module. Its primary purpose is to enhance the Discriminator's ability to differentiate between a ground truth frame and a colorized frame. The Discriminator Loss $L^{\mathbb{D}}$ can be calculated as follow:

$$\begin{aligned} L^{\mathbb{D}} = \Bigg[& \mathbb{E}_{(\hat{x}_{t-1},\hat{x}_t)} \Big(\big(D(\hat{x}_{t-1},\hat{x}_t) \\ & - \mathbb{E}_{(x_{t-1},x_t)} D(x_{t-1},x_t) + 1 \big)^2 \Big) \\ & + \mathbb{E}_{(x_{t-1},x_t)} \Big(\big(D(x_{t-1},x_t) \\ & - \mathbb{E}_{(\hat{x}_{t-1},\hat{x}_t)} D(\hat{x}_{t-1},\hat{x}_t) - 1 \big)^2 \Big) \Bigg] / 2 \end{aligned} \quad (4.9)$$

The Discriminator Loss exhibits a comparable characteristic to the Adversarial Loss (Section 4.2.3.1.2) implemented for the Generator (Section 4.2.2.2), but differs slightly in its operator. This discrepancy in the operator choice leads to the gradient tensor being in the opposite direction compared to the Generator's loss. By optimizing the Discriminator Loss, we aim to train the Discriminator (Section 4.2.2.3) to become more proficient in discerning between ground truth frames and colorized frames, ultimately improving the overall performance of the colorization process. See more about the log of training losses in Appendix B

4.2.4 Implementation detail

4.2.4.1 Dataset

4.2.4.1.1 Training dataset We curate an extensive collection of video datasets specifically designed for training purposes. Our dataset comprises a wide range of sources, including the Hollywood2 dataset [35], videos obtained from Pixabay [36], the DAVIS dataset [1], the REDS dataset [37], and the training dataset from the NTIRE 2023 Video Colorization

Previous frame Current frame Optical flow Reference image

Fig. 4.6 Example from training dataset

Previous frame Current frame Reference frame

Fig. 4.7 Example from augmented ImageNet-1K.

Challenge [38]. The combined dataset consists of 269 videos from the Hollywood2 dataset, 783 videos downloaded from Pixabay using the Pixabay API, and 90 videos from the DAVIS dataset, which provides densely annotated video object segmentation data. Moreover, we leverage the REDS dataset as a valuable resource for video restoration and enhancement tasks. Furthermore, we incorporate the NTIRE 2023 Video Colorization Dataset, which includes video sequences specifically created for the video colorization domain.

In the context of our methodology described in Section 4.2.2.4, for each pair, we carefully curate the five most similar images from the ImageNet-1K dataset [30]. These images serve as example guides for the model in the colorization process. For extracting optical flow information, this research uses the RAFT algorithm [26]. Figure 4.6 provides a sample of our training dataset.

To enhance the diversity of the dataset, we randomly select 70,000 images from the ImageNet-1K dataset [30], ensuring that each image had a corresponding reference image. We apply random geometric distortion and luminance noise techniques to introduce greater variety and realism into the dataset. This augmentation process can be observed in Figure 4.7, where the resulting video frames display enhanced diversity and increased realism. Augmented *Current frame* (middle) created by applying random geometric distortion and

luminance noise to the image. *Previous frame* (left) represents the original unaltered image. *Reference image* (right) also obtained from ImageNet-1K.

We ensure the quality of our dataset through a filtering process, eliminating low-quality and black-and-white videos from the aforementioned sources. Additionally, we conduct a rigorous manual verification process to ensure adherence to our quality standards. Consequently, we accumulate 172,032 samples for the training dataset, summarized in Table 4.1.

Table 4.1 Summary of training datasets

Dataset Name	**No. videos**	**No. samples**
Hollywood2 [35]	269	12,560
Pixabay [36]	783	38,416
DAVIS [1]	90	6,118
REDS [37]	300	23,760
NTIRE 2023 [38]	170	21,178
Augmented dataset from ImageNet-1K [30]	-	70,000
Total	1,612	172,032

4.2.4.1.2 Evaluation dataset We evaluate models on the DAVIS dataset [39] and Videvo [2]. DAVIS consists of 30 diverse videos while Videvo is a collection of 35 clips collected from the Videvo platform. Those evaluation datasets are widely recognized benchmarks in the field of video object segmentation and have played a crucial role in video colorization research by providing standardized evaluations.

4.2.4.2 Hyper-parameters

We utilized a modified version of SwinV2-T [25], the tiny pretrained version of the SwinT v2 model from Huggingface [40] and ViT-T [5], the tiny pretrained version of the ViT model from Hugginface [40] as the backbones of two approaches' architecture. Each approach has 6 loss weights (λ_{L1}, λ_{adv}, λ_{perc}, λ_{ctx}, λ_{temp} and λ_{smooth}) corresponding to 6 loss functions mentioned in 4.2.3. For optimization, we employed the AdamW optimizer [41] for both the Generator module and the Discriminator module with the hyper-parameters: learning rate α, momentum estimates β_1 and β_2. Specific values for the six loss weights, α, β_1, and β_2 are shown for each approach in Table 4.2. We also adopt the "poly" learning rate decay which is inspired by the work of DeepLab [42]. The formula is given below:

$$\gamma = \gamma_0(1 - \frac{N_{iter}}{N_{total}})^{0.9} \tag{4.10}$$

Table 4.2 Summary of hyper-parameters

Hyper-parameter	**ViTExCo**	**SwinTExCo**
α	1×10^{-5}	3×10^{-5}
β_1	0.500	0.500
β_2	0.999	0.999
λ_{L1}	2.000	2.000
λ_{adv}	0.200	0.500
λ_{perc}	0.005	0.100
λ_{ctx}	0.500	1.000
λ_{temp}	0.020	0.300
λ_{smooth}	5.000	5.000

where N_{iter} and N_{total} represent the current iteration number and the total iteration number. The model is trained for 20 epochs using a batch size of 4.

We implemented the distributed data-parallel technique to increase the training speed and trained the models on 2 NVIDIA RTX A6000 GPUs.

Chapter 5

RESULTS

5.1 Evaluation metrics

5.1.1 Quantitative

The quantitative metric presents a systematic approach for objectively evaluating and analyzing the performance of trained models. For comparison with our two models, We select other five models: FAVC [11], Color2Embed [43], VCGAN [17], DeepExemplar [19] and TCVC [20]. DeepExemplar and TCVC are state-of-the-art in exemplar-based colorization, while the others are commonly used models. By including these models in the evaluation, we aim to establish a comprehensive and meaningful benchmark for assessing the performance of our proposed approach.

- **Peak signal-to-noise ratio (PSNR)**: This metric quantifies the disparity between the original signal and a compressed or transmitted version of that signal, accounting for the amount of noise introduced during compression or transmission. In this research, PSNR was employed to assess the fidelity of the colorized video frames by measuring the signal (i.e., colors) present in them.

- **Structural Similarity Index Measure (SSIM)** [44]: SSIM gauges the similarity between two images. It is grounded on the principle that the perceived quality of an image by humans is closely related to its structural information. SSIM takes into consideration the luminance, contrast, and structural resemblance between a reference image and a distorted image. In this research, SSIM was utilized to compare each colorized video frame with its corresponding ground truth.

- **Learned Perceptual Image Patch Similarity (LPIPS)** [45]: Unlike conventional metrics like PSNR and SSIM, which rely on handcrafted features and assumptions

about the human visual system, LPIPS is predicated on a deep neural network trained on a large dataset of images to learn features that are pertinent to human perception. LPIPS was employed to evaluate the similarity between each colorized video frame and its ground truth.

- **Fréchet inception distance (FID)** [46]: This metric is commonly employed to assess the quality and diversity of images generated by Generative Adversarial Networks (GANs). It employs a pre-trained Inception network to extract feature vectors from the ground truth and colorized frames and then computes the mean and covariance of these vectors. The FID is calculated as the Fréchet distance between the distributions defined by these statistics.

- **Color Distribution Consistency (CDC)** [47]: CDC is frequently employed to evaluate the similarity of color distributions between two images or video frames. It takes into account the distribution of colors within an image or frame, rather than solely focusing on individual pixel values or overall image quality. CDC assesses the consistency of the color distribution in a video by comparing the frequencies of different color values across consecutive frames of that video.

The metrics described above are computed on a per-frame basis for the predicted video, and their average values are used for the overall evaluation. This process is repeated for the entire evaluation dataset, and the mean value is calculated to obtain the final evaluation score.

5.1.2 Qualitative

Although quantitative metrics have inherent limitations in fully capturing the breadth of human sensory perception, we have surveyed to evaluate human perception using the Mean Opinion Score (MOS) metric. To ensure a fair and meaningful comparison, we have selected two models, specifically DeepExemplar [19] and TCVC [20], because they have also utilized an exemplar. This selection ensures that our evaluation is conducted in a manner that is relevant and comparable to the existing literature in the field.

5.2 Comparison

5.2.1 Quantitative

Quantitative evaluations were conducted as outlined in Section 5.1.1, providing a comprehensive analysis of each model's performance. The results obtained from these evaluations, as

Table 5.1 Comparison with state-of-the-art video colorization models on DAVIS [1] and Videvo [2] datasets about quantitative metrics. The blue data represents the model with the best performance, while the red data represents the model with the second-best performance.

	DAVIS dataset [1]				
Models	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	CDC ↓
FAVC [11]	16.80765	0.49951	0.36293	181.54643	0.00339
Color2Embed [43]	27.09984	0.92238	0.12797	97.70031	0.00381
VCGAN [17]	23.74646	0.89443	0.17274	146.69747	0.00828
DeepExemplar [19]	29.63041	0.94850	0.08847	73.30805	0.00451
TCVC [20]	31.81122	0.96309	0.06695	58.48240	0.00393
ViTExCo (Ours)	31.95127	0.96329	0.07682	63.85472	0.00407
SwinTExCo (Ours)	32.24419	0.96586	0.06328	56.66980	0.00398
	Videvo dataset [2]				
Models	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	CDC ↓
FAVC [11]	17.47619	0.54666	0.36249	153.20350	0.00163
Color2Embed [43]	27.25240	0.93831	0.14053	97.58350	0.00177
VCGAN [17]	25.02977	0.92525	0.18520	119.60610	0.00300
DeepExemplar [19]	29.75574	0.96154	0.10054	77.69207	0.00192
TCVC [20]	31.19570	0.97091	0.07714	65.11399	0.00168
ViTExCo (Ours)	31.28165	0.93971	0.11872	68.94613	0.00188
SwinTExCo (Ours)	31.96999	0.94402	0.10871	43.36435	0.00172

presented in Table 5.1, offer valuable insights into the capabilities of each model. Table 5.1 demonstrates that SwinTExCo outperforms the other models across multiple evaluation metrics while ViTExCo also achieves remarkable results. In the DAVIS dataset [1], the SwinTExCo outperformed the other models in PSNR, SSIM, LPIPS and FID. In the Videvo dataset [2], SwinTExCo demonstrates slightly lower scores on SSIM and LPIPS. This could be attributed to the fact that DeepExemplar [19] and TCVC [20] were trained on videos that are similar to those in the Videvo dataset. The ViTExCo displays slightly lower performance compared to the SwinTExCo on both datasets due to the deep embedding of spatial information in the feature maps' values, which poses challenges during the decoding process. However, the predicted outputs of the ViTExCo demonstrate that it is a competitive approach. The CDC metric assesses the consistency across predicted frames in a video. Therefore, if the model produces pale-colorized videos, the CDC still recognizes them as consistent colorized videos. That's why both the ViTExCo and SwinTExCo only show acceptable results on the CDC metric. In summary, these results indicate that SwinTExCo and ViTExCo

produce colorized videos that exhibit greater structural similarity and perceptual image patch similarity while maintaining consistency in color distribution.

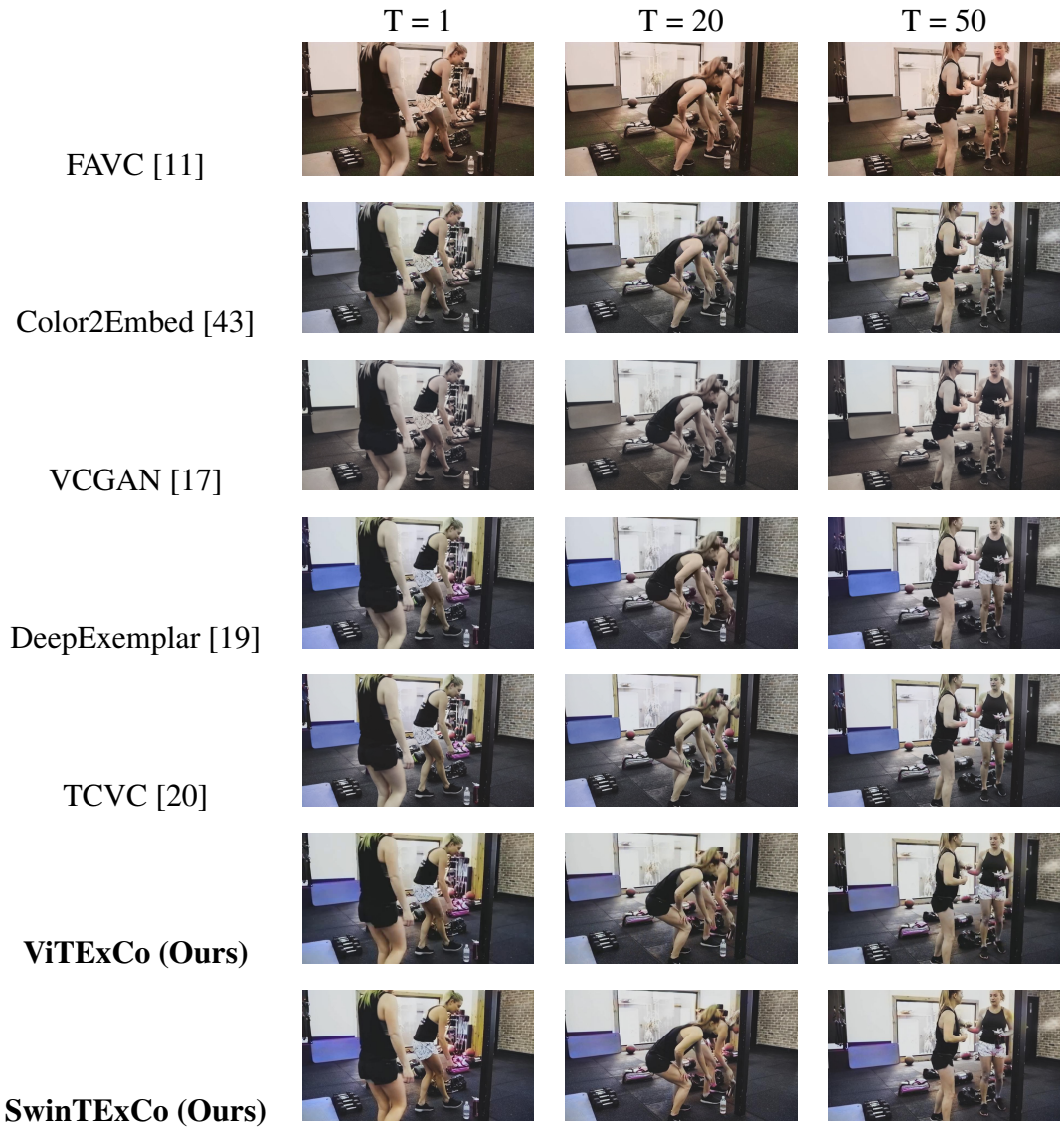


Fig. 5.1 Comparison of colorization consistency between other models and ours at the 1st, 20th, and 50th frames in the video "gym" of the DAVIS dataset.

Figures 5.1 and 5.2 showcase the output generated by the other models compared to SwinTExCo, specifically at the 1st, 20th, and 50th frame of two videos: *gym* and *horsejump-stick*, extracted from the DAVIS dataset. It is apparent from these figures that SwinTExCo excels in generating visually compelling and vibrant results while ensuring color consistency throughout the entire duration of the videos.

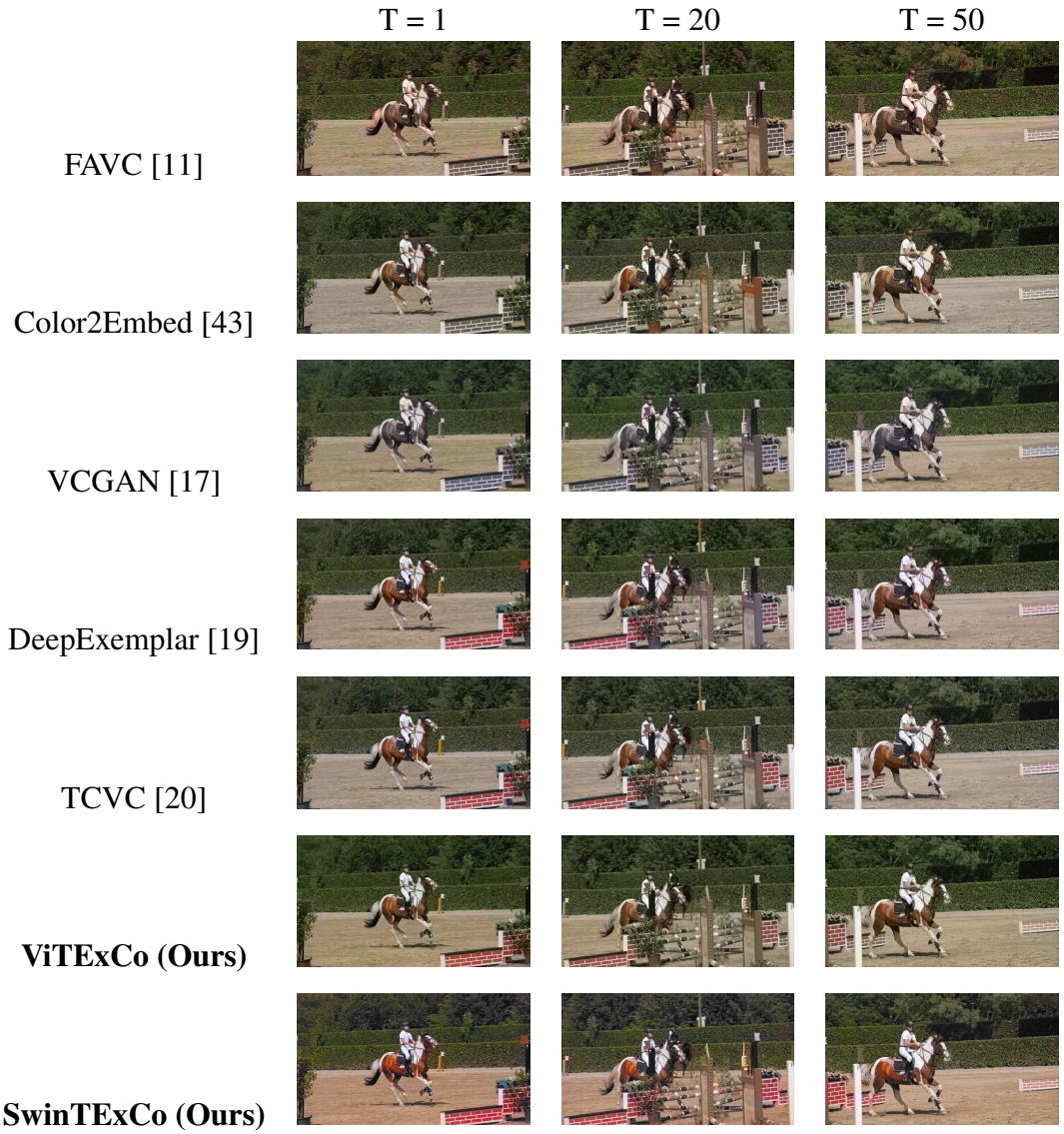


Fig. 5.2 Comparison of colorization consistency between other models and ours at the 1st, 20th, and 50th frames in the video "horsejump-stick" of the DAVIS dataset.

5.2.2 Qualitative

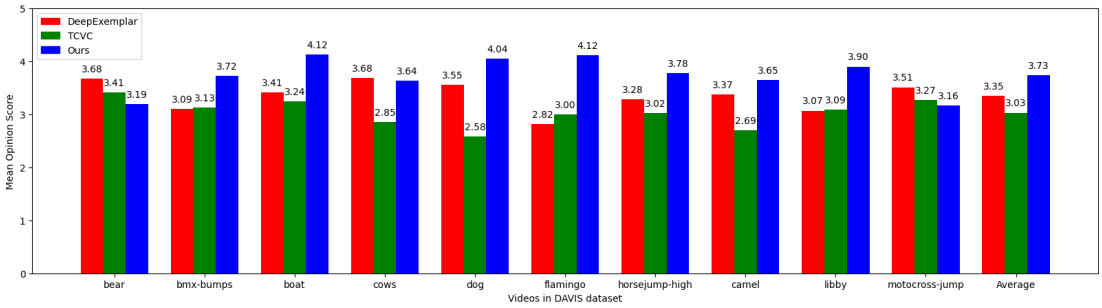


Fig. 5.3 Evaluation results obtained from users' opinions

To conduct a comprehensive evaluation of the colorized output, we randomly selected 10 videos from the DAVIS dataset and administered a survey to a total of 500 individuals. Each participant was tasked with rating the quality of the colorized output generated by each model on a visual appearance scale, which ranged from 1 (indicating poor colorized output) to 5 (indicating excellent colorized output).

Figure 5.3 illustrates the survey results, presenting users' opinions regarding the colorized videos produced by the three models under evaluation. The results clearly demonstrate that SwinTExCo surpasses both DeepExemplar [19] and TCVC [20] in terms of vibrant and realistic colorization, as perceived by the users. This finding highlights the superior performance of SwinTExCo in terms of meeting users' expectations and preferences for visually appealing and natural-looking colorized videos.

5.2.2.1 Exemplar-based colorization ability

To assess the exemplar-based colorization capability of our model, we conducted experiments where we generated images using multiple reference images. The results of these experiments are depicted in Figure 5.4 and Figure 5.5. The outcomes clearly demonstrate that SwinTExCo and ViTExCo both exhibit a remarkable capacity to adapt colors in a highly flexible manner. It effectively and accurately colors areas and objects that exist in both the input and reference images, showcasing its ability to leverage the information provided by the reference images to generate visually coherent and realistic colorized outputs. This finding further underscores the efficacy and versatility of SwinTExCo and ViTExCo in the context of exemplar-based colorization tasks.

Fig. 5.4 Different reference images for colorizing the same image with SwinTExCo

Fig. 5.5 Different reference images for colorizing the same image with ViTExCo

Table 5.2 Comparison of the computational complexity for single frame colorization

Model	**#params**	**FLOPs ↓**	**FPS ↑**
DeepExemplar [19]	**59.74M**	172.12G	2.4
TCVC [20]	124.32M	600.87G	2.3
SwinTExCo (ours)	78.66M	**84.99G**	**23.7**

5.2.2.2 Computational complexity

We conducted a thorough analysis to compare the computational complexity of SwinTExCo with two exemplar-based models, namely DeepExemplar [19] and TCVC [20]. The analysis involved inferring the models on the same set of videos and using the same reference image, with the NVIDIA Tesla T4 GPU employed in an identical computational environment.

Table 5.2 presents the results of our analysis, including the number of parameters, the number of floating-point operations (FLOPs) required for single-frame calculation, and the FPS of each model. The findings reveal that SwinTExCo exhibits the lowest FLOPs requirement, totaling 84.99 GFLOPs, while achieving an impressive FPS of 23.7 frames per second. In comparison, DeepExemplar necessitates 2.03 times more calculations than SwinTExCo, while TCVC requires 7.07 times more calculations. Additionally, SwinTExCo achieves a FPS that is 9.88 times higher than DeepExemplar and 10.30 times higher than TCVC.

These results unequivocally demonstrate that SwinTExCo possesses a significant computational efficiency advantage. It demands fewer computations than its counterparts while delivering comparable performance. This efficiency renders SwinTExCo an attractive choice, as it offers a more scalable and resource-efficient solution for exemplar-based colorization tasks.

Chapter 6

DISCUSSIONS

6.1 Problem solving

SwinTExCo and ViTExCo address the aforementioned challenges in Section 2. These models effectively overcome the limitations associated with the Interactive approach, which often demands significant time and expertise from users to provide colorization hints. By offering flexible customization capabilities, SwinTExCo and ViTExCo empower users with more control over the colorization process. Moreover, these models achieve remarkable outcomes by leveraging state-of-the-art architectures in Computer Vision, distinguishing themselves from other techniques within the Exemplar-based approach.

6.2 Application

As outlined in Section 1.2, this research project is driven by both technical motivations and the preservation of historical value. The objective is to enhance the visual appeal of monochromatic old films and documentaries by effectively colorizing them. The proposed models, ViTExCo and SwinTExCo, offer realistic colorization capabilities and customizable features, thereby demonstrating the potential to generate high-quality colorized videos. By captivating the attention of not only teenagers but also adults, these models contribute to raising awareness about the importance of preserving historical value.

The implications of this research are far-reaching, with numerous potential applications. The model we have developed can serve as a pre-trained model, possessing robust knowledge of various color patterns, for tasks such as color enhancement or color profile inference. Additionally, it can be employed as a colorization module for night vision Closed-Circuit Television (CCTV) systems or monochrome cameras used in Auto Exploration Rovers

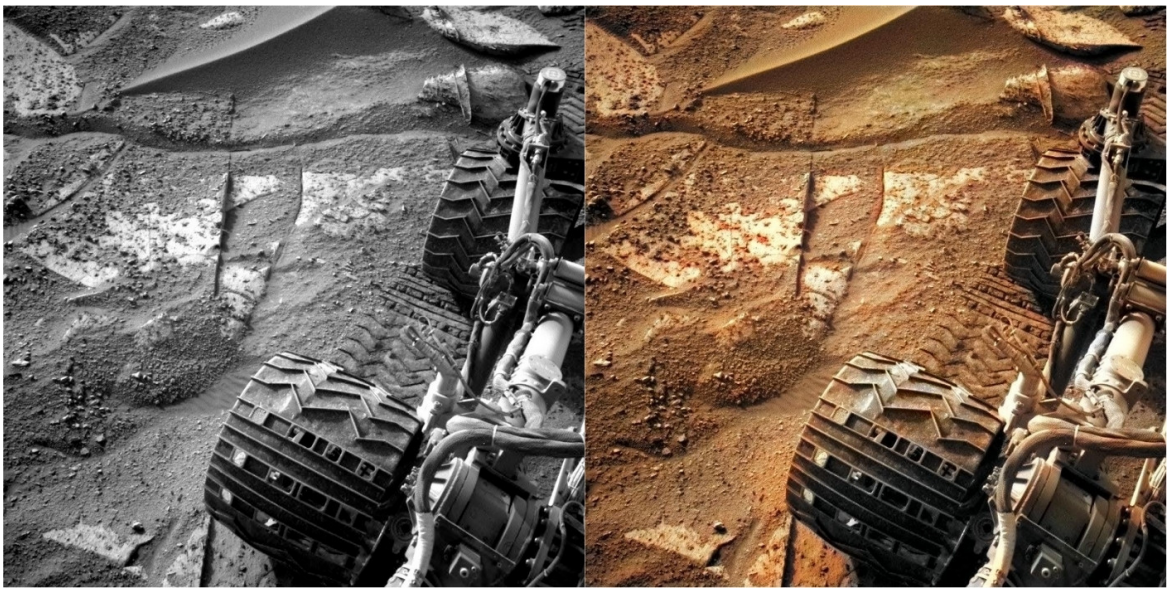


Fig. 6.1 The image on the left was captured on Sol 3912 by the Navigation Camera, which is installed on NASA's Mars rover Curiosity. The image on the right is the colorized version produced by our SwinTExCo model.
Source: The image on the left was adapted from [48]

(shown in Figure 6.1). Note that to conserve signal bandwidth and storage memory during the extended exploration missions of rovers, video data is commonly transmitted and stored in grayscale format.

6.3 Publication

The two proposed models, ViTExCo and SwinTExCo, have been independently submitted to two different publication organizations. ViTExCo has been submitted to The 14th International Conference on ICT Convergence (ICTC2023) and has undergone a rigorous review process. It gives us great pleasure to announce that ViTExCo was accepted and presented at the Conference on October 12th, 2023. Further details about the acceptance can be found in Appendix E. On the other hand, SwinTExCo has been submitted to the Expert Systems with Applications Journal and is currently undergoing the evaluation process.

Chapter 7

CONCLUSIONS

Through a comprehensive series of training and evaluation experiments, our research team has made significant advancements in the field of "Exemplar-based Video Colorization." These experiments encompassed various aspects such as model backbones, overall architecture, datasets, data processing techniques, model optimization, and hyperparameters. The results obtained from our research have been remarkable, surpassing both quantitative metrics and qualitative metrics (such as surveys) when compared to numerous state-of-the-art models available until 2023. This achievement has not only provided valuable insights into potential solutions and challenges in the field of Machine Learning/Artificial Intelligence but has also shed light on the feasibility of conducting research within limited time constraints and computing resources while maintaining the integrity and quality of the research outcomes.

Moving forward, our team is committed to further advancing this project. We intend to enhance the project by incorporating larger and more diverse training datasets, leveraging advanced model backbones, refining optimization algorithms, and designing a robust architecture. Moreover, we aspire to develop a specialized video colorization framework catering to researchers from various domains who seek to explore and utilize this technology.

References

[1] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool, "The 2017 davis challenge on video object segmentation," *arXiv:1704.00675*, 2017.

[2] L. Stival, "Videvo," 2 2023. [Online]. Available: https://figshare.com/articles/figure/Videvo/21766271

[3] Y. LeCun, Y. Bengio *et al.*, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, Jan. 1995.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27. Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[6] X. Chen, D. Zou, Q. Zhao, and P. Tan, "Manifold preserving edit propagation," *ACM Trans. Graph.*, vol. 31, no. 6, Nov. 2012.

[7] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 689–694.

[8] Q. Luan, F. Wen, D. Cohen-Or, L. Liang, Y.-Q. Xu, and H.-Y. Shum, "Natural image colorization," in *Proceedings of the 18th Eurographics conference on Rendering Techniques*, 2007, pp. 309–320.

[9] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," 2017.

[10] J. Yun, S. Lee, M. Park, and J. Choo, "icolorit: Towards propagating local hints to the right region in interactive colorization by leveraging vision transformer," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Jan. 2023, pp. 1787–1796.

[11] C. Lei and Q. Chen, "Fully automatic video colorization with self-regularization and diversity," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[12] M. Limmer and H. P. A. Lensch, "Infrared colorization using deep convolutional neural networks," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Dec. 2016, pp. 61–68.

[13] P. Vitoria, L. Raad, and C. Ballester, "Chromagan: Adversarial picture colorization with semantic class distribution," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, Mar. 2020.

[14] Y. Wu, X. Wang, Y. Li, H. Zhang, X. Zhao, and Y. Shan, "Towards vivid and diverse image colorization with generative color prior," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 14 377–14 386.

[15] Z. Wan, B. Zhang, D. Chen, and J. Liao, "Bringing old films back to life," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 17 694–17 703.

[16] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 10 012–10 022.

[17] Y. Zhao, L.-M. Po, W. Y. Yu, Y. A. U. Rehman, M. Liu, Y. Zhang, and W. Ou, "Vcgan: Video colorization with hybrid generative adversarial network," *IEEE Transactions on Multimedia*, 2022.

[18] P. Kouzouglidis, G. Sfikas, and C. Nikou, "Automatic video colorization using 3d conditional generative adversarial networks," in *Advances in Visual Computing: 14th International Symposium on Visual Computing, ISVC 2019, Lake Tahoe, NV, USA, October 7–9, 2019, Proceedings, Part I 14*. Springer, 2019, pp. 209–218.

[19] B. Zhang, M. He, J. Liao, P. V. Sander, L. Yuan, A. Bermak, and D. Chen, "Deep exemplar-based video colorization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.

[20] Y. Zhang, S. Chen, M. Wang, X. Zhang, C. Zhu, Y. Zhang, and X. Li, "Temporal consistent automatic video colorization via semantic correspondence," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2023, pp. 1836–1845.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," vol. 30, 2017.

[22] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," 2022.

[23] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, Jul. 2021, pp. 10 347–10 357.

[24] H. Bao, L. Dong, and F. Wei, "Beit: Bert pre-training of image transformers. arxiv 2021," *arXiv preprint arXiv:2106.08254*, 2021.

[25] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 12 009–12 019.

[26] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 402–419.

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[28] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." Computational and Biological Learning Society, 2015, pp. 1–14.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255.

[31] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," 2015.

[32] R. Mechrez, I. Talmi, and L. Zelnik-Manor, *The Contextual Loss for Image Transformation with Non-Aligned Data*, Sep. 2018.

[33] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua, "Coherent online video style transfer," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017.

[34] M. Ruder, A. Dosovitskiy, and T. Brox, "Artistic style transfer for videos," in *Pattern Recognition*, B. Rosenhahn and B. Andres, Eds. Cham: Springer International Publishing, 2016, pp. 26–36.

[35] M. Marszałek, I. Laptev, and C. Schmid, "Actions in context," in *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[36] "Pixabay," https://pixabay.com/.

[37] S. Nah, S. Baik, S. Hong, G. Moon, S. Son, R. Timofte, and K. M. Lee, "Ntire 2019 challenge on video deblurring and super-resolution: Dataset and study," in *CVPR Workshops*, June 2019.

[38] X. Kang, X. Lin, K. Zhang, Z. Hui, W. Xiang, J.-Y. He, X. Li, P. Ren, X. Xie, R. Timofte, Y. Yang, J. Pan, Z. Peng, Q. Zhang, J. Dong, J. Tang, J. Li, C. Lin, Q. Li, Q. Liang, R. Gang, X. Liu, S. Feng, S. Liu, H. Wang, C. Feng, F. Bai, Y. Zhang, G. Shao, X. Wang, L. Lei, S. Chen, Y. Zhang, H. Xu, Z. Liu, Z. Zhang, Y. Luo, and Z. Zuo, "Ntire 2023 video colorization challenge," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2023, pp. 1570–1581.

[39] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[40] R. Wightman, "Pytorch image models," https://github.com/huggingface/pytorch-image-models, 2019.

[41] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[42] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 06 2016.

[43] H. Zhao, W. Wu, Y. Liu, and D. He, "Color2embed: Fast exemplar-based image colorization using color embeddings," *arXiv preprint arXiv:2106.08017*, 2021.

[44] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[45] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[46] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6629–6640.

[47] Y. Liu, H. Zhao, K. C. K. Chan, X. Wang, C. C. Loy, Y. Qiao, and C. Dong, "Temporally consistent video colorization with deep feature propagation and self-regularization learning," 2021.

[48] "Sols 3914-3915: Blazing a path to the gediz vallis ridge," 2023. [Online]. Available: https://mars.nasa.gov/msl/mission-updates/9461/sols-3914-3915-blazing-a-path-to-the-gediz-vallis-ridge/

[49] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, Jul. 2021.

[50] "Video scene cut detection and analysis tool," https://www.scenedetect.com/, 2023.

Appendix A

Hugging Face

A.1 Overview

Hugging Face is a leading platform for hosting machine learning models. It provides a centralized hub for researchers and developers to store, access, and share pre-trained models across different domains and tasks. With a user-friendly interface, versatile API, and support for multiple programming languages, Hugging Face allows seamless integration into applications. The platform encourages collaboration and knowledge sharing through its community contributions feature, enabling users to upload and share their own models, datasets, and training pipelines. It also excels in model versioning and management, facilitating iterative development and experimentation. With its extensive model repository and efficient serving capabilities, Hugging Face is a valuable resource for the machine-learning community. The SwinTExCo inference model used in this research is publicly available on the Hugging Face model Space at: https://huggingface.co/Spaces/chronopt-research/SwinTExCo

A.2 Gradio

Gradio is a platform that facilitates the deployment of machine learning models within the Hugging Face ecosystem. It offers researchers and developers the ability to create interactive and customizable interfaces for their models, enabling users to input data and receive real-time predictions. Gradio supports a variety of input and output interfaces, such as text boxes, image uploaders, sliders, and dropdown menus, to cater to different model requirements. Integration with Hugging Face allows for seamless deployment of models using minimal code. The platform also supports multi-model deployments, enabling the side-by-side comparison

of different models. The deployment of SwinTExCo on Huggingface Space is implemented with the Gradio framework for both the front-end and back-end.

A.3 Technical specifications

Upon submitting a request for a complimentary GPU allocation to deploy the SwinTExCo model, we were granted access to an Nvidia T4 small GPU along with 4 vCPUs, 15 GB of RAM, and 16 GB of VRAM. The deployment of SwinTExCo is facilitated within a dedicated Space provided by the Huggingface platform, wherein the model is set to enter sleep mode after one hour of inactivity. It is worth noting that the primary objective of this research does not revolve around deploying the model on a web hosting platform for unrestricted public usage, thereby eliminating the need to upgrade to higher-tier plans.

Appendix B

WandB

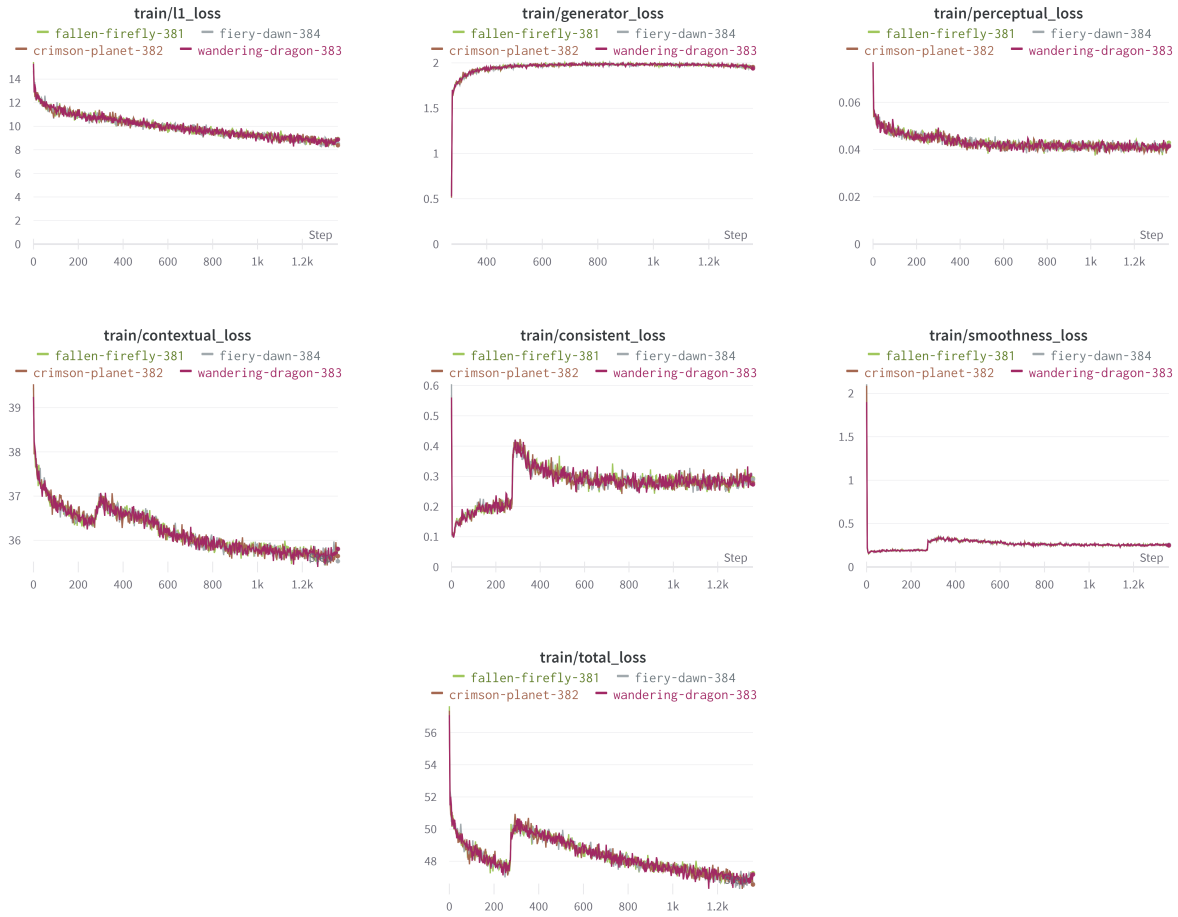


Fig. B.1 The training losses of SwinTExCo generated by WandB using 2 GPUs A6000 with batch size of 2

Appendix C

Automatic Video Colorization Deployment

C.1 FAISS

C.1.1 Overview

FAISS (Facebook AI Similarity Search) [49] is a powerful open-source library developed by Facebook AI Research for efficient similarity search and clustering of large-scale datasets. It is widely used in various applications, including recommendation systems, image and video retrieval, natural language processing, and more.

C.1.2 Indexing Techniques

FAISS offers several indexing techniques optimized for different types of data and search requirements. These include:

- **Flat Index:** The most straightforward index, suitable for small datasets.
- **IVF (Inverted File) Index:** A partitioning-based index that divides the dataset into clusters for faster search.
- **HNSW (Hierarchical Navigable Small World) Index:** A graph-based index that builds a navigable graph for efficient approximate nearest neighbor search.
- **PQ (Product Quantization) Index:** An index that compresses the data using quantization techniques for memory-efficient search.

- **IVFFlat Index:** By combining the Inverted File and Flat indexes, the IVFFlat index takes advantage of the strengths of both approaches. It benefits from the fast pruning capabilities of the Inverted File index, which reduces the number of vectors to consider during the search. At the same time, it leverages the simplicity and efficiency of the Flat index for the final refinement step within the selected cells.

C.1.3 Similarity search

FAISS provides highly optimized algorithms for similarity search, allowing you to find the nearest neighbors of a given query efficiently. It supports both exact and approximate search methods, with approximate search offering significant speed-ups for large-scale datasets.

C.2 PySceneDetect

C.2.1 Overview

PySceneDetect [50] is an open-source software library designed to identify shot changes within videos and automatically partition the video into distinct clips. It offers a range of detection methods, varying from straightforward threshold-based detection of fade in/out transitions to sophisticated content-aware fast-cut detection for each shot. PySceneDetect is freely available and provides researchers and users with a versatile tool for effectively analyzing video content.

C.2.2 Scenes spliting methods

C.2.2.1 Content detector

The input video undergoes a content detection algorithm for analysis. For each frame, a score ranging from 0 to 255.0 is computed, indicating the dissimilarity between the current frame and the preceding one (a higher score denotes greater dissimilarity). A cut is generated when a frame score surpasses the specified threshold.
The scores are derived from distinct components:

- **delta_hue:** Represents the disparity in pixel hue values between adjacent frames.
- **delta_sat:** Measures the variation in pixel saturation values between adjacent frames.
- **delta_lum:** Quantifies the distinction in pixel luma (brightness) values between adjacent frames.

- **delta_edges:** Reflects the dissimilarity in calculated edges between adjacent frames. This component typically yields larger values compared to others, necessitating a potential threshold adjustment for compensation.

C.2.2.2 Adaptive detector

Apply an adaptive detection algorithm to the input video using a two-pass approach. The algorithm calculates frame scores using detect-content in the first pass and applies a rolling average for result processing. This helps mitigate false detections, particularly in cases involving camera movement.

C.2.2.3 Threshold detector

Apply a threshold detection algorithm to analyze the input video. The algorithm identifies fade-in and fade-out events by examining the average pixel values. Cuts are then positioned between consecutive fade-out and fade-in events, indicating significant transitions in the video content.

C.3 Deployment

Initially, we aimed to train the exemplar-based video colorization **scene-by-scene**, which means all of the frames in each scene are consecutive and relevant. Due to this, it is necessary to detect scenes in the video and separate them to input them into the model. We describe the detailed pipeline of the Automatic Video Colorization Deployment in Figure C.1.

To handle a long video with multiple scenes, we first divide it into distinct scenes by using PySceneDetect [50]. Next, a reference image is identified for each scene to facilitate the colorization process. Subsequently, the colorization procedure is applied to all scenes using the corresponding reference images. To enhance the efficiency of the reference image search, we have incorporated a high-performance vector database. This database leverages the FAISS [49] library for expedited searching based on cosine similarity. By utilizing the advanced searching mechanism, we can retrieve the most relevant reference images for colorization rapidly. Lastly, we combine the output colorized scenes into a single, colorized video.

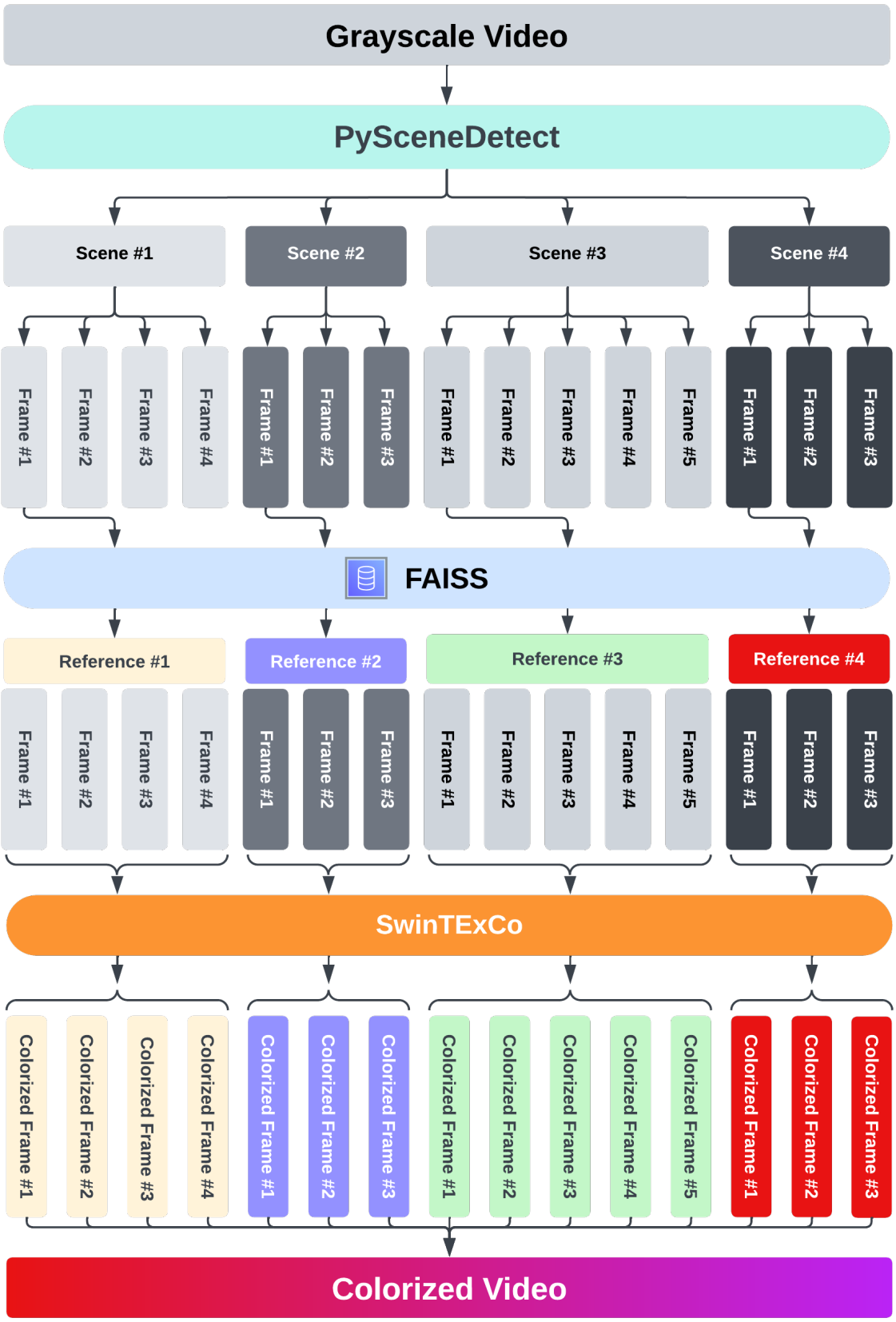


Fig. C.1 Visualization of Automatic Video Colorization Deployment pipeline

Appendix D

Half-precision FP16

D.1 Introduction to Half-precision FP16

Half-precision floating-point (FP16) format is a compact representation of floating-point numbers that utilizes 16 bits. It consists of 1 bit for the sign, 5 bits for the exponent, and 10 bits for the significand. FP16 has gained attention for its potential advantages in terms of reduced storage requirements and increased computational efficiency.

D.2 Advantages of Half-Precision FP16

Reduced Storage Requirements: By utilizing only 16 bits, FP16 consumes half the storage space compared to single-precision FP32 and one-fourth the space of double-precision FP64. This reduction in storage requirements is particularly advantageous in scenarios where memory is limited, such as in embedded systems or high-performance computing environments.

Increased Computational Efficiency: The compact representation of FP16 allows for higher data throughput and improved parallelism. In applications involving large-scale computations, such as deep learning or scientific simulations, the use of FP16 can lead to faster processing times and improved overall performance.

D.3 Challenges and Considerations

Precision Loss: The main drawback of FP16 is the reduced precision compared to higher precision formats. With a smaller range of representable numbers and a limited

number of significant bits, FP16 is susceptible to precision loss and rounding errors. This may impact the accuracy of computations, especially in scenarios that require high precision, such as numerical simulations or certain scientific calculations.

Numerical Stability: Due to the decreased precision, certain numerical operations, such as subtraction or division, can be more prone to producing inaccurate results or introducing significant errors. Careful consideration and appropriate techniques, such as scaling or error analysis, are necessary to mitigate these stability issues when working with FP16 data.

D.4 Application in this research

In this project, the implementation of FP16 was employed to reduce the storage memory of optical flow files in the training dataset. The training dataset comprised hundreds of thousands of optical flow files, with an average size of 1.153 MB (megabytes). By converting the optical flow files to FP16 format, the average size was reduced to 0.566 MB, resulting in a reduction of approximately 50.91%. After converting to FP16, the average precision loss is less than 1% compared to the original values, resulting in negligible changes in precision. On the other hand, This reduction translates to saving between 70 and 80 gigabytes in training dataset storage and loading.

Appendix E

The 14th International Conference on ICT Convergence (ICTC 2023)

E.1 Overview

ICTC is a Scopus indexed (link) and globally recognized conference focusing on the emerging industrial convergence surrounding information and communication technologies (ICTs). It is a platform for researchers, industry professionals, and academics to explore the challenges and advancements in machine learning, wireless communication, smart devices, mobile cloud computing, green communication, healthcare, and intelligent transportation. The conference features technical sessions, tutorials, and invited industrial sessions, providing knowledge dissemination and collaboration opportunities.

E.2 Related Information

The novel methodology for colorization - ViTExCo - was accepted at ICTC 2023 and will be published in late 2023. The online proceeding version of ViTExCo paper can be accessed at section number B1-4 on ICTC 2023 website (link).