DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Learning Temporal Consistency in Video Generation

Felix Altenberger

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Learning Temporal Consistency in Video Generation

Erlernen der Zeitlichen Konsistenz bei der Videogenerierung

Author:	Felix Altenberger
Supervisor:	Prof. Dr. Matthias Nießner
Advisor:	Prof. Dr. Matthias Nießner
Submission Date:	15.06.2020

I confirm that this master’s thesis in data engineering and analytics is my own work and I have documented all sources and material used.

Munich, 15.06.2020 Felix Altenberger

Acknowledgments

I would like to thank the TUM Visual Computing Group for providing me with the resources required to conduct my research. In particular, I want to thank Prof. Matthias Nießner for both his supervision and his valuable insights, which had a strong influence on the outcome of this work. Furthermore, I would like to thank the other students in the lab that helped shape my research through valuable discussions and suggestions, specifically Felix Meissen and Moritz Wolf.

Abstract

We propose a novel general-purpose loss for temporal consistency in generative adversarial video generation. In our *TimeCycle Loss*, a GAN is trained jointly with a motion model, which is a neural network that shifts frames according to the natural movement in the specific domain. The motion model learns temporally-consistent shifting via a temporal cycle-consistency constraint that requires it to reconstruct its input through a backward-forward generation cycle. By applying the motion model to the generations of the GAN, we constrain the GAN to produce temporally-consistent results. Unlike existing methods, our loss has no task-specific model assumptions and can be easily applied to arbitrary GANs in any domain. Based on our novel loss, we propose the *TimeCycleGAN* meta-architecture for temporally-consistent video generation, where we additionally employ a recurrent generator design and an unconditional sequence discriminator. We apply the meta-architecture to two basic methods for paired and unpaired image-to-image translation and demonstrate significant improvements in temporal consistency, by up to 63 percent. Our models can generate photorealistic, temporally-consistent street-scene videos in both paired and unpaired settings, and achieve quantitative video quality and temporal consistency scores comparable to the respective task-specific state-of-the-art.

Kurzfassung

In dieser Arbeit präsentieren wir einen neuen Ansatz zum Erlernen der zeitlichen Konsistenz bei der Videogenerierung mit Generative Adversarial Networks. Hierfür verwenden wir eine neue Lossfunktion, den *TimeCycle* Loss, bei dem wir ein zeitlich-konsistentes Bewegungsmodell zusammen mit dem GAN trainieren. Das Bewegungsmodell ist ein neuronales Netz, welches lernt, Bilder einer Sequenz gemäß der natürlichen Bewegung zu verschieben. Die temporale Konsistenz des Bewegungsmodells wird durch eine zusätzliche Einschränkung gewährleistet, welche fordert, dass das Bewegungsmodell das ursprüngliche Bild nach einem zeitlichen Rückwärts-Vorwärts-Zyklus rekonstruieren kann. Dadurch, dass das GAN darauf trainiert wird, nachfolgende Bilder einer Sequenz ähnlich wie das Bewegungsmodell zu generieren, wird die temporale Konsistenz auch auf das GAN übertragen. Im Unterschied zu existierenden Ansätzen stellt unsere Methode keinerlei Anforderungen an das zugrundeliegende Modell und kann einfach in beliebige GANs eingefügt werden. Basierend auf dem TimeCycle Loss, präsentieren wir die *TimeCycleGAN* Metaarchitektur für temporal-konsistente Videogenerierung mit GANs, bei welcher zusätzlich eine Modifikation im Generator des GANs durchgeführt, und ein zusätzlicher sequentieller Diskriminator hinzugefügt wird. Wir wenden unsere Metaarchitektur auf zwei einfache Modelle für gepaarte und ungepaarte Bild-zu-Bild Übersetzung an und zeigen, dass sich dadurch temporal-konsistente Videos generieren lassen. In quantitativen Vergleichen erzielen unsere Modelle ähnliche Ergebnisse wie die besten Methoden im jeweiligen Bereich.

Contents

1. Introduction

In machine learning, success often depends on the amount of training data available. However, collecting sufficiently large datasets is often cumbersome, expensive, or even impossible. This is, for instance, the case in traffic accident simulation, or applications dealing with rare medical conditions. Video data, in particular, is often sparse, which inhibits research and development progress in corresponding areas. By providing means to obtain sufficiently large video datasets where only limited real data exists, artificial video generation methods could enable and empower many applications.

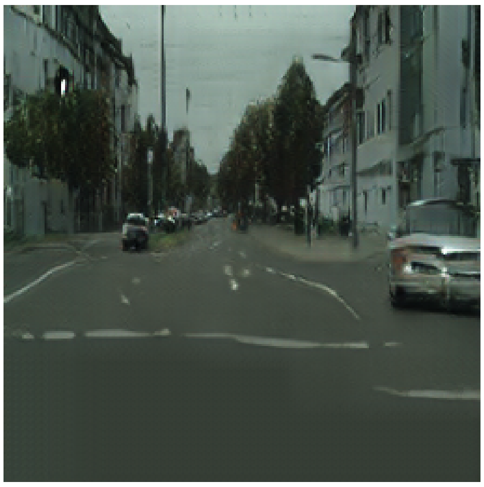
In recent years, deep learning approaches have significantly advanced the state-of-the-art in both image and video generation. The central innovation underlying this progress is the generative adversarial network (GAN), where not only the generation process is learned, but also an image quality measurement, which, unlike simple heuristics, guides the generation process to create detailed, photorealistic results.

While image generation with GANs is a popular research topic, GAN-based video generation is a much less explored task, which is fundamentally difficult because it requires the joint optimization of spatial consistency, temporal consistency, and image quality. Current video generation approaches are highly task- and method-specific, which makes it challenging to apply them to other GAN models in different settings.

We propose a novel approach for general-purpose GAN-based video generation with no assumptions on the underlying GAN architecture. Based on the idea of temporal cycle-consistency, we design a new loss function that enforces temporal consistency by jointly training a small motion model with the GAN. We also propose a simple sequential GAN design, which, when combined with the novel loss, forms a meta-architecture for temporal consistency that can be easily applied to *any* GAN.

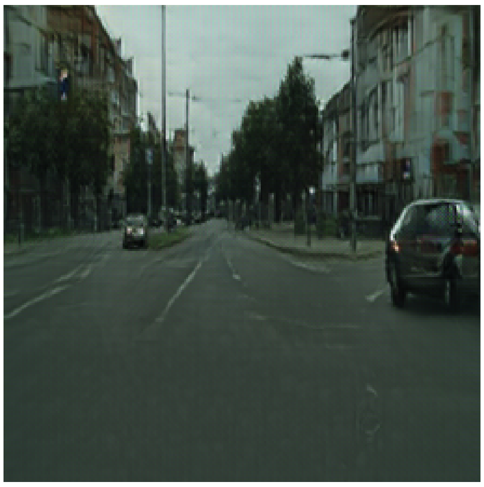
We demonstrate the meta-architecture by applying it to two basic per-frame image generation approaches to generate photorealistic, temporally-consistent street-scene videos in paired and unpaired video-to-video translation settings, as shown in Figure 1.1. We further compare the resulting video generation models to the task-specific state-of-the-art methods of the respective domains and demonstrate comparable results. In summary, we combine temporal cycle-consistency learning with a simple sequential GAN design to form the three key contributions of our work:

- a novel general-purpose loss for temporally consistent video generation,
- a meta-architecture for GAN-based video generation that can be applied to *any* GAN,
- two simple, yet powerful models for paired and unpaired video-to-video translation.

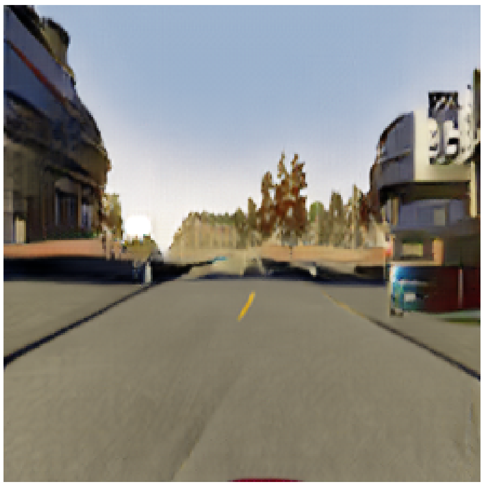


(a) pix2pix

$\Rightarrow$

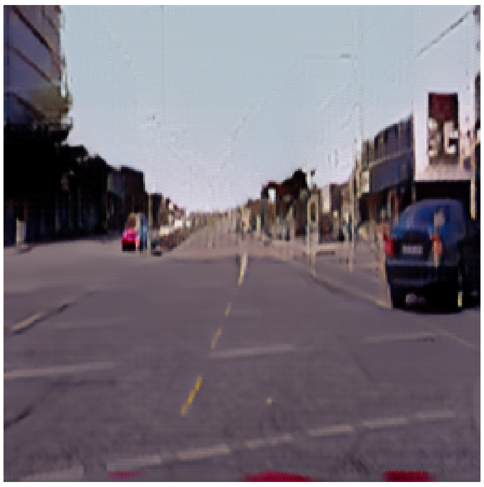


(b) TimeCycleGAN-P++



(c) CycleGAN

$\Rightarrow$



(d) TimeCycleGAN-UP

Figure 1.1.: Applying the TimeCycleGAN meta-architecture to two per-frame GAN methods to make them temporally consistent. Top: paired video-to-video translation with pix2pix [1]. Bottom: unpaired video-to-video translation with CycleGAN [2]. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*

2. Related Work

2.1. Generative Adversarial Image Generation

2.1.1. Generative Adversarial Networks

A *Generative Adversarial Network* (GAN) [3] consists of two main components: A *Generator G*, which generates images $\tilde{x}$ from input noise z, and a *Discriminator D*, which should distinguish the generated fake images $\tilde{x}$ from real images x. These two models are trained jointly in an adversarial manner, where the generator's task is to deceive the discriminator into judging the generated images as real ($D(G(z)) = 1$), and the discriminator's task is to correctly identify real images as real ($D(x) = 1$) and generated images as fake ($D(G(z)) = 0$). Thereby, the generator should learn to generate fake images that are indistinguishable from real images.

Objective Function The optimization scheme described above can be expressed as

$$G^* = \underset{G}{\operatorname{argmin}}\{\max_D\{\mathcal{L}_{GAN}(G,D)\}\}, \tag{2.1}$$

with the GAN loss

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}_{x\sim p_{data}(x)}[\log(D(x))] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))], \tag{2.2}$$

where $p_{data}(x)$ is the distribution of real images approximated by our training dataset, and $p_z(z)$ is the input noise distribution, which is typically chosen as the standard-normal distribution $\mathcal{N}(0,1)$. In the following, we will abbreviate $\mathbb{E}_{x\sim p_{data}(x)}$ and $\mathbb{E}_{z\sim p_z(z)}$ as $\mathbb{E}_x$ and $\mathbb{E}_z$ respectively.

Conditional GANs

It is also possible for GANs to learn conditional generations, simply by providing additional input data y to both generator and discriminator [3], which changes the GAN objective in Equation 2.1 to:

$$G^* = \underset{G}{\operatorname{argmin}}\{\max_D\{\mathcal{L}_{cGAN}(G,D)\}\} \tag{2.3}$$

with the conditional GAN loss

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log(D(x|y))] + \mathbb{E}_{z,y}[\log(1 - D(G(z|y)|y))], \tag{2.4}$$

where $\mathbb{E}_{x,y} = \mathbb{E}_x\mathbb{E}_y$ and $\mathbb{E}_{z,y} = \mathbb{E}_z\mathbb{E}_y$ with $\mathbb{E}_y = \mathbb{E}_{y\sim p_{data}(y)}$ for the distribution of conditioning inputs $p_{data}(y)$. To simplify the notation, we will abbreviate the conditional generation $G(z|y)$ as $G(y)$ in the remainder and omit $\mathbb{E}_z$ accordingly.

2.1.2. DCGAN

In [4], Radford et al. propose the *Deep Convolutional GAN* (DCGAN), which is used to learn unsupervised feature representations through generative adversarial training.

DCGAN Architecture Guidelines The main contribution of DCGAN are its proposed network architecture guidelines for stable GAN training, which are: replacing pooling layers with strided convolutions as proposed in [5], removing all fully-connected layers, using batch normalization [6] in both generator and discriminator, and using ReLU [7] and LeakyReLU [8, 9] activation functions in the generator and discriminator respectively. All of these guidelines are still applied in most state-of-the-art GAN methods today [1, 2, 10, 11, 12, 13].

DCGAN Model Based on the proposed architecture guidelines, Radford et al. design a novel fully-convolutional GAN model, which is an unconditional GAN, as introduced in Section 2.1.1, that is used to generate images of size 64×64.

2.1.3. Pix2Pix

One of the areas where GANs have been applied most successfully is *Image-to-Image Translation*, which is a special case of conditional generation, where the conditioning input is another image. In paired image-to-image translation, *pix2pix* [1] is a popular approach, which proposes a simple, general architecture that can be applied to many domains.

U-Net Generator The generator of pix2pix is an encoder-decoder network [14], which uses additional skip connections [15] between layers of equivalent size, leading to a design that is closely related to the U-Net architecture [16].

PatchGAN Discriminator Pix2pix also introduces a new discriminator architecture called *PatchGAN*, which is based on the semantic segmentation network in [17]. PatchGAN can be interpreted as a sliding window approach that generates one output per image patch, where all per-patch outputs are averaged in the end to obtain the final discriminator prediction.

L1 Loss Since the PatchGAN discriminator is only discriminating images patch-wise, it does not use global image information. To make generations also globally consistent, pix2pix adds an additional L1 loss term

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[||x - G(y)||_1]. \tag{2.5}$$

Objective Function By combining the L_1 loss with adversarial training, the objective function of pix2pix becomes

$$G^* = \underset{G}{\text{argmin}}\{\underset{D}{\max}\{\mathcal{L}_{cGAN}(G, D)\} + \lambda_{L1}\mathcal{L}_{L1}(G)\}, \tag{2.6}$$

where λ_{L1} is a hyperparameter that defines the relative weight of the L1 loss.

2.1.4. Pix2PixHD

pix2pixHD [10] extends the pix2pix implementation in order to enable the generation of more realistic high-resolution images. To do so, multiple architecture adjustments were made.

Coarse-to-Fine Generator The generator in pix2pixHD consists of two parts: the *global generator*, which generates lower-resolution images, and a *local enhancer*, which is built around the global generator to increase the resolution. Both generators are based on the architecture by Johnson et al. [18], which we also use in our proposed models, as we will discuss in Section 4.1.1.

Multi-Scale Discrimination The discriminator in pix2pixHD is based on the PatchGAN discriminator of pix2pix. However, pix2pixHD applies $N_{DS} - 1$ additional discriminators to downsampled versions of the image. In the following, we use D_i to denote the discriminator applied to images downsampled by a factor of 2^{i-1} for $i \in \{1, ..., N_{DS}\}$ with N_{DS} scales.

Feature Matching and Perceptual Losses In addition to the GAN losses, pix2pixHD uses *feature matching losses* [19]

$$\mathcal{L}_{FM}(G, D) = \mathbb{E}_{x,y}[\sum_{i=1}^{L} \frac{1}{N_i} ||D^{(i)}(x|y) - D^{(i)}(\tilde{x}|y)||_1], \tag{2.7}$$

as well as a *perceptual loss* [20] on five layers of a VGG19 [21] network

$$\mathcal{L}_P(G) = \mathbb{E}_{x,y}[\sum_{i \in \{2,7,12,21,30\}} \frac{1}{N_i} ||VGG^{(i)}(x) - VGG^{(i)}(G(y))||_1]. \tag{2.8}$$

We also apply both losses in one of our paired TimeCycleGAN models, and we will discuss them in more detail in Section 3.3.5.

Objective Function The total training objective of pix2pixHD is:

$$\begin{aligned} G^* = \underset{G}{\text{argmin}}\{\max_D\{\sum_{k=1}^{N_{DS}} \mathcal{L}_{cGAN}(G, D_k)\} \\ + \lambda_{FM} \sum_{k=1}^{N_{DS}} \mathcal{L}_{FM}(G, D_k) \\ + \lambda_P \mathcal{L}_P(G)\}, \end{aligned} \tag{2.9}$$

where $D = (D_1, ..., D_{N_DS})$ is a multi-scale discriminator, $L_{FM}(G, D)$ and $L_P(G)$ are feature matching and perceptual losses respectively, and λ_{FM} and λ_P are corresponding loss weights.

2.1.5. CycleGAN

In [2], Zhu et al. propose *CycleGAN*, a technique for unpaired image-to-image translation, where, in contrast to the paired setting, no information is available on how a given source image should look like in the target domain. In Section 3.3, we will discuss the differences between unconditional, unpaired, and paired generation in more detail. In order to learn the mapping from the source domain Y to the target domain X in an unpaired setting, two GANs (G_X, D_X) and (G_Y, D_Y) are trained jointly in CycleGAN, whereby each GAN should learn a mapping from one domain to the other such that $\tilde{y} = G_Y(x)$ and $\tilde{x} = G_X(y)$ are realistic images of domains Y and X respectively.

Cycle-Consistency Loss To learn meaningful mappings between the domains, an additional *cycle-consistency loss* is introduced:

$$\begin{aligned} \mathcal{L}_C(G_X, G_Y) = \mathbb{E}_x[||G_X(G_Y(x)) - x||_1] \\ + \mathbb{E}_y[||G_Y(G_X(y)) - y||_1] \end{aligned} \tag{2.10}$$

Note that this cycle-consistency loss will play an important role later, because it is closely related to our proposed TimeCycle loss in Section 3.2.

Identity Mapping Losses Additionally, two *identity mapping losses*

$$\begin{aligned} \mathcal{L}_{idt}(G_X) = \mathbb{E}_x[||G_X(x) - x||_1] \\ \mathcal{L}_{idt}(G_Y) = \mathbb{E}_y[||G_Y(y) - y||_1] \end{aligned} \tag{2.11}$$

are used in some CycleGAN applications to further constrain the generators to prevent them from performing unwanted color flipping.

Objective Function Using the cycle-consistency loss $\mathcal{L}_C(G_X, G_Y)$ and identity mapping losses $\mathcal{L}_{idt}(G_X)$ and $\mathcal{L}_{idt}(G_Y)$ with corresponding loss weights λ_C and λ_{idt}, the two GANs are then trained jointly in CycleGAN with the following objective:

$$\begin{aligned} G_X^*, G_Y^* = \underset{G_X, G_Y}{\operatorname{argmin}} \{ & \max_{D_X} \{\mathcal{L}_{GAN}(G_X, D_X)\} + \max_{D_Y} \{\mathcal{L}_{GAN}(G_Y, D_Y)\} \\ & + \lambda_C \mathcal{L}_C(G_X, G_Y) \\ & + \lambda_{idt}(\mathcal{L}_{idt}(G_X) + \mathcal{L}_{idt}(G_Y))\} \end{aligned} \tag{2.12}$$

2.2. Generative Adversarial Video Generation

2.2.1. Vid2Vid

Video-to-Video Translation refers to the task of generating videos from other videos and can be seen as an extension of image-to-image translation to video data. *vid2vid* [11] is a paired video-to-video translation technique that is based on pix2pixHD [10], and extends it in various ways to generate temporally-consistent high-resolution videos.

Recurrent Generator The first difference to pix2pixHD is that the generator G is trained to generate sequences of length T and is additionally conditioned on the N_G previous generations and source images. Thus, for source images y and generations $\tilde{x}$, the generator is adjusted from $\tilde{x}_{t+1} = G(y_{t+1})$ to $\tilde{x}_{t+1} = G(c_{t+1})$ with

$$c_{t+1} = (y_{(t-n_G+1):(t+1)}, \tilde{x}_{(t-n_G+1):t}). \tag{2.13}$$

Note that we will discuss such recurrent generators in more detail in Section 3.1.1.

Flow Prediction Network The core contribution of vid2vid is an optical flow prediction network W, and a corresponding occlusion mask prediction network M, which are both trained jointly with the generator G. These networks are used to optical-flow-warp a generation $\tilde{x}_t$, and to combine it with the subsequent generation $\tilde{x}_{t+1}$ for improved temporal consistency. A new fake image $\tilde{x}_{t+1}$, given the conditioning c_{t+1}, is then generated by

$$\tilde{x}_{t+1} = M(c_{t+1}) * G(c_{t+1}) + (1 - M(c_{t+1})) * \text{warp}(\tilde{x}_t, W(c_{t+1})), \tag{2.14}$$

where $\text{warp}(x, w)$ is an optical-flow-warping operation that warps image x according to optical flow w. In the following, we will use $\hat{G}$ to denote the three networks (G, W, M) that are used to generate images.

Flow Loss The flow prediction network W is additionally trained in a supervised fashion to mimic a ground truth flow w and to warp images according to the real movement of the target images, using the following *flow loss*:

$$\mathcal{L}_F(W) = \mathbb{E}_{x_{1:T}, y_{1:T}}[\frac{1}{T-1} \sum_{t=1}^{T-1} ||w_t - W(c_{t+1})||_1 + ||\text{warp}(x_t, W(c_{t+1})) - x_{t+1}||_1], \tag{2.15}$$

where $x_{1:T}$ and $y_{1:T}$ are target and source sequences of length T, w_t is the ground truth flow between x_t and x_{t+1}, and $\text{warp}(x, w)$ is the optical-flow-warping operation. To obtain the ground truth flow in domains where no ground truth flow is available, a FlowNet 2.0 [22] model is inferenced on the real target sequences.

Warp Loss The combined generator $\hat{G} = (G, W, M)$ is also trained with a *warp loss*

$$\mathcal{L}_W(\hat{G}) = \mathbb{E}_{x_{1:T}, y_{1:T}}[\frac{1}{T-1} \sum_{t=1}^{T-1} ||\text{warp}(\tilde{x}_t, w_t) - \tilde{x}_{t+1}||_1], \tag{2.16}$$

where $\tilde{x}_t$ and $\tilde{x}_{t+1}$ are fake images generated by $\hat{G}$ according to Equation 2.14. This loss asserts that the optical flow between subsequent generations is similar to the optical flow between the corresponding real images.

Sequence Discriminator For improve temporal consistency further, another discriminator D_S is added. Similar to the per-frame discriminator, D_S is also a multi-scale discriminator, as defined in Section 2.1.4. However, instead of discriminating a single frame x_t, D_S is run on a sequence $x_{t:(t+N_D-1)}$ of N_D subsequent frames. Furthermore, D_S is also conditioned on the $N_D - 1$ ground truth optical flows $w_{t:(t+N_D-2)}$ between subsequent frames. For a detailed discussion on sequence discriminators refer to Section 3.1.2.

Temporal Multi-Scale Discrimination To learn long-term temporal consistency, vid2vid also applies *temporal multi-scale discrimination*. The idea behind this approach is that, instead of applying the sequence discriminator D_S to a sequence of N_D subsequent frames $x_{t:(t+N_D-1)} = (x_t, x_{t+1}, x_{t+2}, ..., x_{t+N_D-1})$, it is applied to general sequences of equally-spaced frames $(x_t, x_{t+i}, x_{t+2i}, ..., x_{t+(N_D-1)*i})$. This can be interpreted as applying the sequence discriminator to a modified version of the original sequence, which had its frame rate reduced by a factor of i. By applying the sequence discriminator to N_{DT} different temporal scales, both short-term and long-term temporal consistency can be learned. Note that a separate sequence discriminator $D_{S,i}$ is used for each temporal scale i.

Objective Function By combining all above contributions, we can express the total objective function of vid2vid as

$$\begin{aligned} G^*, W^*, M^* = \underset{G,W,M}{\operatorname{argmin}}\{\max_{D,D_S}\{ &\sum_{k=1}^{N_{DS}} \mathcal{L}_{cGAN}(\hat{G}, D_k) + \sum_{i=1}^{N_{DT}} \mathcal{L}_{cGAN}(\hat{G}, D_{S;k,i})\} \\ &+ \lambda_{FM}(\sum_{k=1}^{N_{DS}} \mathcal{L}_{FM}(\hat{G}, D_k) + \sum_{i=1}^{N_{DT}} \mathcal{L}_{FM}(\hat{G}, D_{S;k,i})) \\ &+ \lambda_P \mathcal{L}_P(\hat{G}) \\ &+ \lambda_W \mathcal{L}_W(\hat{G}) \\ &+ \lambda_F \mathcal{L}_F(W)\}, \end{aligned} \tag{2.17}$$

where $\hat{G} = (G, W, M)$ is the combined generator, $D = (D_1, ..., D_{N_{DS}})$ is a spatial multi-scale discriminator as defined in Section 2.1.4, $D_S = (D_{S;1,1}, ..., D_{S;N_{DS},N_{DT}})$ is a spatio-temporal multi-scale discriminator with N_{DS} spatial scales and N_{DT} temporal scales, $\mathcal{L}_{FM}$ and $\mathcal{L}_P$ are feature matching and perceptual losses as introduced in Section 2.1.4, and λ_{FM}, λ_P, λ_F, λ_W are loss weights of the corresponding losses.

Other Contributions In addition to the contributions mentioned above, vid2vid also introduces foreground-background priors for improved video quality, weight sharing between networks for faster training, as well as a spatio-temporally progressive training approach. We also use temporally progressive training in our models and will discuss it in more detail in Section 4.1.3.

2.2.2. RecycleGAN

RecycleGAN [12] is an approach for unpaired video-to-video translation. It is based on CycleGAN [2], but, instead of the cycle-consistency loss from Equation 2.10, RecycleGAN uses two novel losses, which improve temporal consistency and alleviate the perceptual mode collapse problem.

Recycle Loss Instead of the spatial cycle-consistency loss of CycleGAN [2], a spatio-temporal cycle-consistency loss is introduced, which is defined as

$$\begin{aligned} \mathcal{L}_{RC}(G_X, G_Y, P_X, P_Y) = \mathbb{E}_x[\sum_{t=1}^{T-2} ||x_{t+1} - G_X(P_Y(G_Y(x_{t-1}, x_t)))||_1] \\ + \mathbb{E}_y[\sum_{t=1}^{T-2} ||y_{t+1} - G_Y(P_X(G_X(y_{t-1}, y_t)))||_1], \end{aligned} \tag{2.18}$$

where P_X and P_Y are auxiliary networks that should predict the next frame of a sequence of the corresponding domain, given the two previous frames.

Recurrent Loss During training, P_X and P_Y are optimized jointly with G_X and G_Y and are additionally trained in a supervised fashion using the loss

$$\begin{aligned} \mathcal{L}_{RCpred}(P_X, P_Y) = \mathbb{E}_x[\sum_{t=1}^{T-2} ||x_{t+2} - P_X(x_t, x_{t+1})||_1] \\ + \mathbb{E}_y[\sum_{t=1}^{T-2} ||y_{t+2} - P_Y(y_t, y_{t+1})||_1]. \end{aligned} \tag{2.19}$$

Objective Function The objective function to be optimized in RecycleGAN is then:

$$\begin{aligned} G_X^*, G_Y^*, P_X^*, P_Y^* = \underset{G_X, G_Y, P_X, P_Y}{\operatorname{argmin}} \{\max_{D_X}\{\mathcal{L}_{GAN}(G_X, D_X)\} + \max_{D_Y}\{\mathcal{L}_{GAN}(G_Y, D_Y)\} \\ + \lambda_{RC}\mathcal{L}_{RC}(G_X, G_Y, P_X, P_Y) \\ + \lambda_{RCpred}\mathcal{L}_{RCpred}(P_X, P_Y)\} \end{aligned} \tag{2.20}$$

2.2.3. TecoGAN

Temporally Coherent GAN (TecoGAN) [13] is a concurrent work on temporal consistency learning in GAN-based video generation, which proposes three major contributions: a recurrent optical-flow generator design, a spatio-temporal triplet discriminator design, and a novel loss for long-term temporal consistency.

Recurrent Optical-Flow Generator TecoGAN proposes a novel recurrent optical-flow-based generator, which, given previous generations $\tilde{x}_{1:t}$, generates the subsequent frame as

$$\tilde{x}_{t+1} = G(s_{t+1}, \text{warp}(\tilde{x}_t, F(s_t, s_{t+1}))), \tag{2.21}$$

where $\text{warp}(x, w)$ is the optical flow warping operation and F is a pretrained optical flow estimation network. Thus, the generator generates novel frames based on the corresponding source, as well as the previous generation, which was optical-flow-warped based on the motion extracted from the corresponding source images.

Spatio-Temporal Triplet Discriminator The main contribution of TecoGAN is a novel spatio-temporal discriminator design. This discriminator D always takes triplets (x_1, x_2, x_3) as input. For an intermediate frame x_t in a real or generated sequence, this triplet is then randomly chosen as either

$$(x_t, x_t, x_t) \tag{2.22}$$

or as

$$(x_{t-1}, x_t, x_{t+1}) \tag{2.23}$$

or as

$$(\text{warp}(x_{t-1}, F(s_{t-1}, s_t)), x_t, \text{warp}(x_{t+1}, F(s_{t+1}, s_t))), \tag{2.24}$$

i.e. the input is either the same frame three times, or three subsequent frames, or three subsequent ones where the first and last are optical-flow-warped to the middle. The probabilities according to which triplets are selected are dynamically adjusted throughout the training from (1, 0, 0) to (0.5, 0.25, 0.25) respectively.

Ping-Pong Loss In addition to the spatio-temporal discriminator, a *Ping-Pong* loss is introduced for further long-term temporal consistency. Given a source sequence $s_{1:T}$, the generator is tasked to first generate the corresponding fake sequence $\tilde{x}_{1:T}$, and then, starting with the last generation $\tilde{x}_T$, generate it backwards again to obtain a second generation sequence $\tilde{x}'_{1:T-1}$. The Ping-Pong loss is then defined as

$$\mathcal{L}_{PP} = \mathbb{E}_{\tilde{x}_{1:T}, \tilde{x}'_{1:T}}[\sum_{t=1}^{T-1} ||\tilde{x}_t - \tilde{x}'_t||_2], \tag{2.25}$$

which is the sum of differences between the two generations for each frame, where $\mathbb{E}_{\tilde{x}_{1:T}, \tilde{x}'_{1:T}}$ denotes the expected value over the generated sequences. Note that this loss is closely related to our proposed TimeCycle loss, and can even be seen as a special case of it, as we discuss in Section A.2.1.

Other Losses In addition to GAN and Ping-Pong losses, TecoGAN also uses an L_2 loss

$$\mathcal{L}_{L2}(G) = \mathbb{E}_{x,y}[||x - G(y)||_2] \tag{2.26}$$

for paired generation, as well as perceptual losses $\mathcal{L}_P(G)$, as introduced in Section 2.1.4, and the vid2vid optical flow warp loss $\mathcal{L}_W(G)$ from Section 2.2.1.

Objective Functions Combining all losses, the objective of the paired TecoGAN model can be expressed as

$$\begin{aligned} G^* = \underset{G}{\operatorname{argmin}}\{\max_{D}\{\mathcal{L}_{cGAN}(G,D)\} \\ + \lambda_{PP}\mathcal{L}_{PP}(G) \\ + \lambda_{L2}\mathcal{L}_{L2}(G) \\ + \lambda_{P}\mathcal{L}_{P}(G) \\ + \lambda_{W}\mathcal{L}_{W}(G)\}, \end{aligned} \tag{2.27}$$

and the objective of the unpaired TecoGAN model as

$$\begin{aligned} G_X^*, G_Y^* = \underset{G_X,G_Y}{\operatorname{argmin}}\{\max_{D_X}\{\mathcal{L}_{GAN}(G_X,D_X)\} + \max_{D_Y}\{\mathcal{L}_{GAN}(G_Y,D_Y)\} \\ + \lambda_{PP}(\mathcal{L}_{PP}(G_X) + \mathcal{L}_{PP}(G_Y)) \\ + \lambda_{C}\mathcal{L}_{C}(G_X,G_Y) \\ + \lambda_{P}(\mathcal{L}_{P}(G_X) + \mathcal{L}_{P}(G_Y))\}, \end{aligned} \tag{2.28}$$

where the perceptual loss in unpaired generation is defined as the average difference of gram matrices between real and fake images, as proposed for the style transfer loss in [23].

2.3. Temporal Cycle-Consistency Learning

2.3.1. Temporal Cycle-Consistency in Feature Representation Learning

In [24], a novel approach for self-supervised tracking, semantic segmentation, and optical flow prediction is proposed, which is based on feature representations that were learned with a novel temporal cycle-consistency loss.

Temporal Cycle-Consistency Given a sequence of T images $I_{1:T}$, a feature extraction network ϕ is trained to extract corresponding feature representations $x_t^I = \phi(I_t)$ for each image I_t. For the temporal cycle-consistency learning, an auxiliary tracker model $\mathcal{T} : x_s^I \times x_t^p \mapsto x_s^p$ is then defined, which tracks a patch p_t, represented by the corresponding feature representation $x_t^p = \phi(p_t)$, first backwards in time with

$$\hat{x}_{t-1}^p = \mathcal{T}(x_{t-1}^I, x_t^p), \tag{2.29}$$

and then forwards in time with

$$\hat{x}_t^p = \mathcal{T}(x_t^I, \hat{x}_{t-1}^p). \tag{2.30}$$

If the reconstruction $\hat{x}_t^p$ is close to the original representation x_t^p, we call the reconstruction *Temporally Cycle-Consistent*.

Long Cycles For cycle-consistency over more than one frame, two types of temporal cycles are proposed in [24]: The first temporal cycle type is a long, iterative cycle over consecutive frames, where we first track backwards i-times in succession

$$\hat{x}^p_{t-i,long} = \mathcal{T}(x^I_{t-i}, ...\mathcal{T}(x^I_{t-2}, \mathcal{T}(x^I_{t-1}, x^p_t))) \tag{2.31}$$

and then forwards i-times in succession

$$\hat{x}^p_{t-i+i,long} = \mathcal{T}(x^I_t, \mathcal{T}(x^I_{t-1}, ...\mathcal{T}(x^I_{t-i+1}, \hat{x}^p_{t-i,long}))). \tag{2.32}$$

Skip Cycles The second temporal cycle type is a direct cycle, where the tracker skips through time by directly predicting the target frames, first backwards with

$$\hat{x}^p_{t-i,skip} = \mathcal{T}(x^I_{t-i}, x^p_t), \tag{2.33}$$

and then forwards with

$$\hat{x}^p_{t-i+i,skip} = \mathcal{T}(x^I_t, \hat{x}^p_{t-i,skip}). \tag{2.34}$$

Temporal Cycle-Consistency Learning The feature extraction network ϕ and the tracker $\mathcal{T}$ are trained jointly on the given image sequence $I_{1:T}$ by minimizing the distances of the temporal cycle reconstructions of both long-range temporal cycle types. To do so, both types are applied with various cycle lengths on a patch x^p_T in the last frame I_T of the sequence. This can be expressed as the following objective function:

$$\phi^*, \mathcal{T}^* = \underset{\phi,\mathcal{T}}{\operatorname{argmin}}\{\mathbb{E}_{x^p_{1:T}}[\sum_{i=1}^{N_{TC}} ||x^p_T - \hat{x}^p_{T-i+i,long}|| + ||x^p_T - \hat{x}^p_{T-i+i,skip}||]\}, \tag{2.35}$$

where N_{TC} is a hyperparameter that specifies the longest temporal distance to track backwards. This loss formulation is strongly related to our proposed TimeCycle loss, as we will discuss in Section 3.2. Note that in [24] an additional feature similarity loss is applied, which was left out for simplicity here, as it is not required for the concept of temporal cycle-consistency.

2.3.2. Temporal Cycle-Consistency for Video Alignment

In [25], a different approach for temporal cycle-consistency is proposed, which is used for temporal video alignment. In this approach, two sequences $S = (s_1, ..., s_N)$ and $T = (t_1, ..., t_M)$ are again encoded with a feature encoding network ϕ first, where the encoded sequences U and V are obtained by $u_t = \phi(s_t)$ and $v_t = \phi(t_t)$. Then, for a given u_t, a temporal cycle is performed by first finding the nearest neighbor $\tilde{v}$ of u_t in V, then finding the neareast neighbor of $\tilde{v}$ back in U, which is optimized to be close to u_t.

2.3.3. Temporal Cycle-Consistency in GANs

To our knowledge, there is no current method that directly applies temporal cycle-consistency learning to GAN-based video generation. However, several state-of-the-art video generation methods share similarities with it.

TecoGAN The most closely related method is the Ping-Pong loss of TecoGAN, described in Section 2.2.3. As we will discuss in Section A.2.1, the Ping-Pong loss can even be seen as a special case of our proposed TimeCycle loss.

RecycleGAN The spatio-temporal cycle-consistency loss of RecycleGAN, as introduced in Equation 2.19, is also somewhat related to temporal cycle-consistency, where the next-frame-prediction with the auxiliary predictor models could be interpreted as half of a length-1 temporal cycle.

3. TimeCycleGAN

3.1. Sequential Generative Adversarial Networks

Let us assume we have an arbitrary GAN model consisting of a per-frame generator G and a per-frame discriminator D that are trained in an adversarial manner to optimize the standard GAN objective defined in Equation 2.1. We visualize such a GAN in Figure 3.1, where we use convolutional architectures to illustrate both G and D.

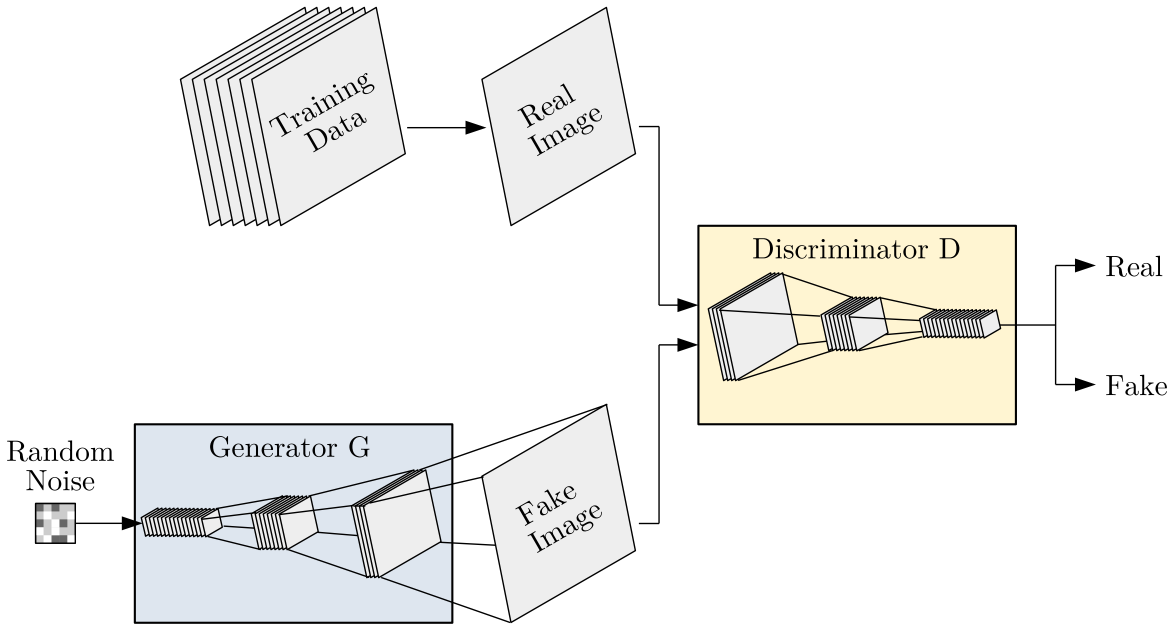


Figure 3.1.: A standard GAN model consisting of a per-frame generator G and a per-frame discriminator D. Both networks are depicted as convolutional networks for illustration purposes.

3.1.1. Sequential Generators

We argue that, in order to learn temporal consistency, there needs to be a correlation between generated frames in the generation process. Similar to vid2vid [11] and TecoGAN [13], we propose to establish this correlation by means of a recurrent generator design that generates subsequent frames $\tilde{x}_{t+1}$ based on the N_G previous generations $\tilde{x}_{(t-N_G+1):t}$ with

$$\tilde{x}_{t+1} = G(c_{t+1}, \tilde{x}_{(t-N_G+1):t}), \tag{3.1}$$

where c_{t+1} is an optional conditioning input, such as the corresponding source image in paired image-to-image translation. When generating the first generations $x_{1:N_G+1}$ of each sequence, we feed random noise into the generator in place of the missing previous generations. Thus, x_1 and x_{t+1} with $t \in \{1, ..., N_G - 1\}$ are generated from a sequence of noise images $z_{1:N_G}$ by

$$\tilde{x}_1 = G(c_{t+1}, z_{1:N_G}), \tag{3.2}$$

and

$$\tilde{x}_{t+1} = G(c_{t+1}, z_{(t+1):N_G}, \tilde{x}_{1:t}), \tag{3.3}$$

where c_{t+1} is again an optional conditioning input. The random noise images $z_{1:N_G}$ are sampled independently from a standard-normal distribution $\mathcal{N}(0,1)$. In Figure 3.2 we visualize this recurrent generator design for $N_G = 1$ and compare it to the per-frame generator of the GAN model depicted in Figure 3.1. For a detailed discussion on the specific generator design choices refer to Section A.1.1.

3.1.2. Sequence Discriminators

To learn temporal consistency in an adversarial setting, we use an additional sequence discriminator D_S that discriminates sequences $x_{t:(t+N_D-1)}$ of N_D frames at a time. For two given sequences $x_{1:T}$ and $\tilde{x}_{1:T}$ of real and fake frames respectively, the sequence discriminator is then applied to all sub-sequences of length N_D to produce $T - N_D + 1$ discriminator outputs

$$d^{real}_{1:(T-N_D+1)} = (D_S(x_1, ..., x_{N_D}), ..., D_S(x_{T-N_D+1}..., x_T)) \tag{3.4}$$

and

$$d^{fake}_{1:(T-N_D+1)} = (D_S(\tilde{x}_1, ..., \tilde{x}_{N_D}), ..., D_S(\tilde{x}_{T-N_D+1}..., \tilde{x}_T)) \tag{3.5}$$

for real and fake sequences respectively. For the case $N_D = T$, the sequence discriminator D_S is then only applied once per sequence, as opposed to the per-frame discriminator D that is applied T times. We compare and visualize both D and D_S for sequential generation in Figure 3.3.

3.1.3. Sequential Generative Adversarial Training

By training the sequential generator G from Section 3.1.1 with an unconditional sequence discriminator D_S from Section 3.1.2, we obtain the sequential generative adversarial objective

$$G^* = \underset{G}{\operatorname{argmin}}\{\underset{D_S}{\max}\{\mathcal{L}_{sGAN}(G, D_S)\}\} \tag{3.6}$$

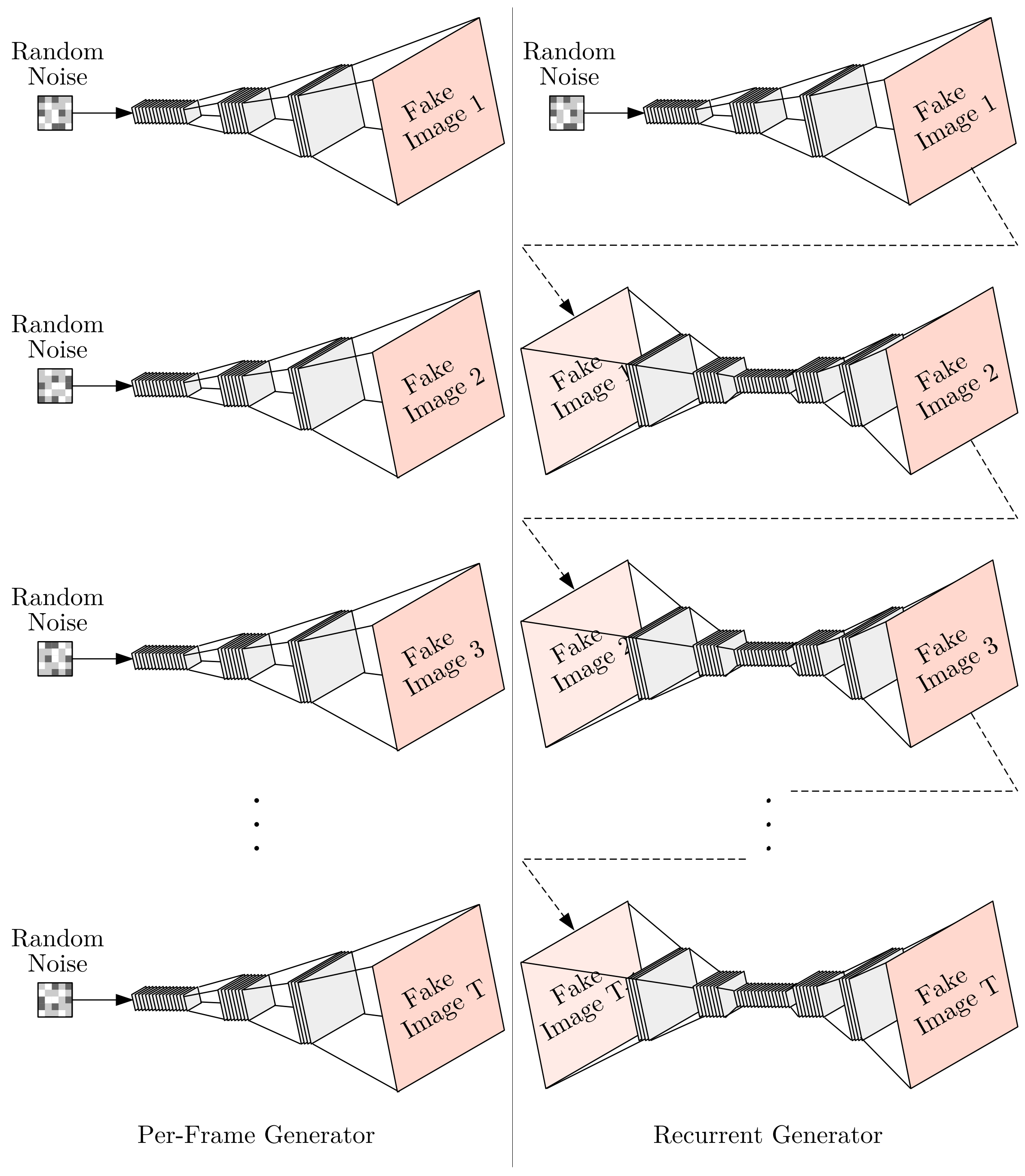


Figure 3.2.: Comparison of our proposed recurrent generator design to a corresponding per-frame generator. The per-frame generator generates each frame independently. The recurrent generator reuses previous generations.

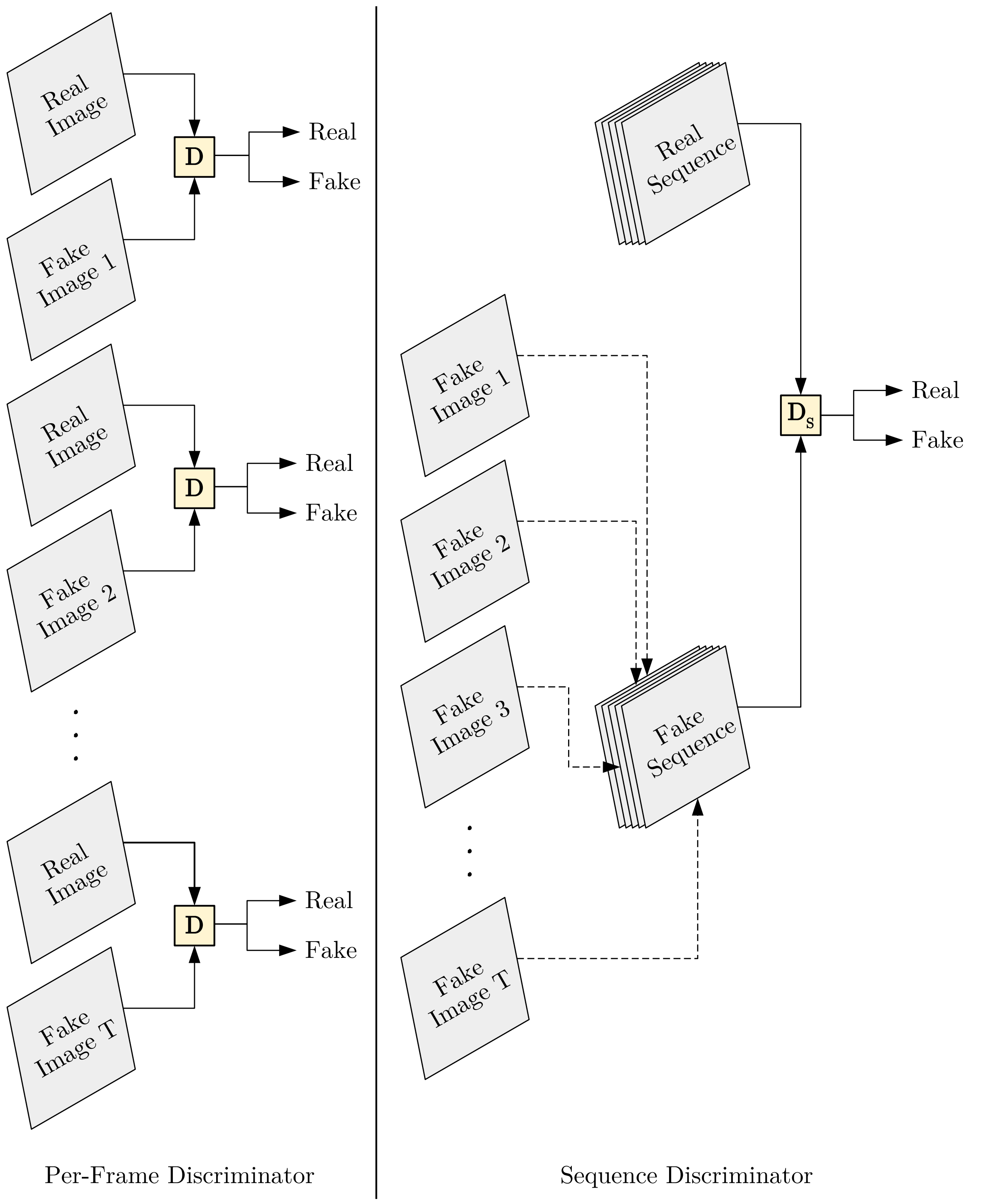


Figure 3.3.: Comparison of our proposed sequence discriminator with $N_D = T$ to a corresponding per-frame discriminator in a sequential GAN. The per-frame discriminator discriminates each frame independently. The sequence discriminator discriminates entire sequences at once.

where $\mathcal{L}_{sGAN}$ is the sequential GAN loss

$$\begin{aligned}\mathcal{L}_{sGAN}(G, D_S) = \mathbb{E}_{x_{1:T}}[\frac{1}{T - N_D + 1} \sum_{t=1}^{T-N_D+1} \log(D_S(x_{t:(t+N_D-1)}))] \\ + \mathbb{E}_{z_{1:N_G}}[\frac{1}{T - N_D + 1} \sum_{t=1}^{T-N_D+1} \log(1 - D_S(G(z_{1:N_G})_{t:(t+N_D-1)}))],\end{aligned} \tag{3.7}$$

where $x_{1:T}$ is a sequence of T real images, $z_{1:N_G}$ is a sequence of N_G standard-normal distributed random noise images, and $G(z_{1:N_G})$ is a sequence of T fake images generated from $z_{1:N_G}$ according to Equations 3.1, 3.2, 3.3. We visualize this sequential training in Figure 3.4.

Combined Objective We can further combine the sequential GAN objective from Equation 3.7 with the initial per-frame objective given in Equation 2.1, leading to the following combined objective:

$$G^* = \underset{G}{\operatorname{argmin}}\{\max_{D}\{\mathcal{L}_{GAN}(G, D)\} + \max_{D_S}\{\mathcal{L}_{sGAN}(G, D_S)\}\} \tag{3.8}$$

This sequential GAN design with recurrent generator, per-frame discriminator, and sequence discriminator is the basis of our TimeCycleGAN meta-architecture. We visualize the sequential GAN in Figure 3.5.

3.2. TimeCycle Loss

To further improve the temporal consistency of consecutive generations, we propose the TimeCycle loss. The idea behind this loss is to train a GAN jointly with a motion model, enforce temporal consistency of the motion model via a temporal cycle-consistency loss, and then pass the temporal consistency to the GAN.

3.2.1. TimeCycle Motion Model

The motion model M is a separate neural network that is applied to shift frames of the sequence generated by the GAN. Given a generated frame $\tilde{x}_t$ of a generated sequence $\tilde{x}_{1:T}$, we can apply the motion model M to shift the frame forwards or backwards in time with

$$\tilde{x}^f_{t+1} = M(\tilde{x}_t | c^f_t) \tag{3.9}$$

and with

$$\tilde{x}^b_{t-1} = M(\tilde{x}_t | c^b_t) \tag{3.10}$$

respectively, where c^f_t and c^b_t are additional conditioning inputs that describe the motion according to which the frame $\tilde{x}_t$ should be shifted forwards or backwards. Following [24], we designed the motion model to be much smaller than the generator and just complex enough to perform image warping. This is discussed in more detail in Sections 4.1.1 and A.1.2. While we can find a general-purpose definition of the conditioning inputs, we can also make use of task-specific information to depict the motion more accurately.

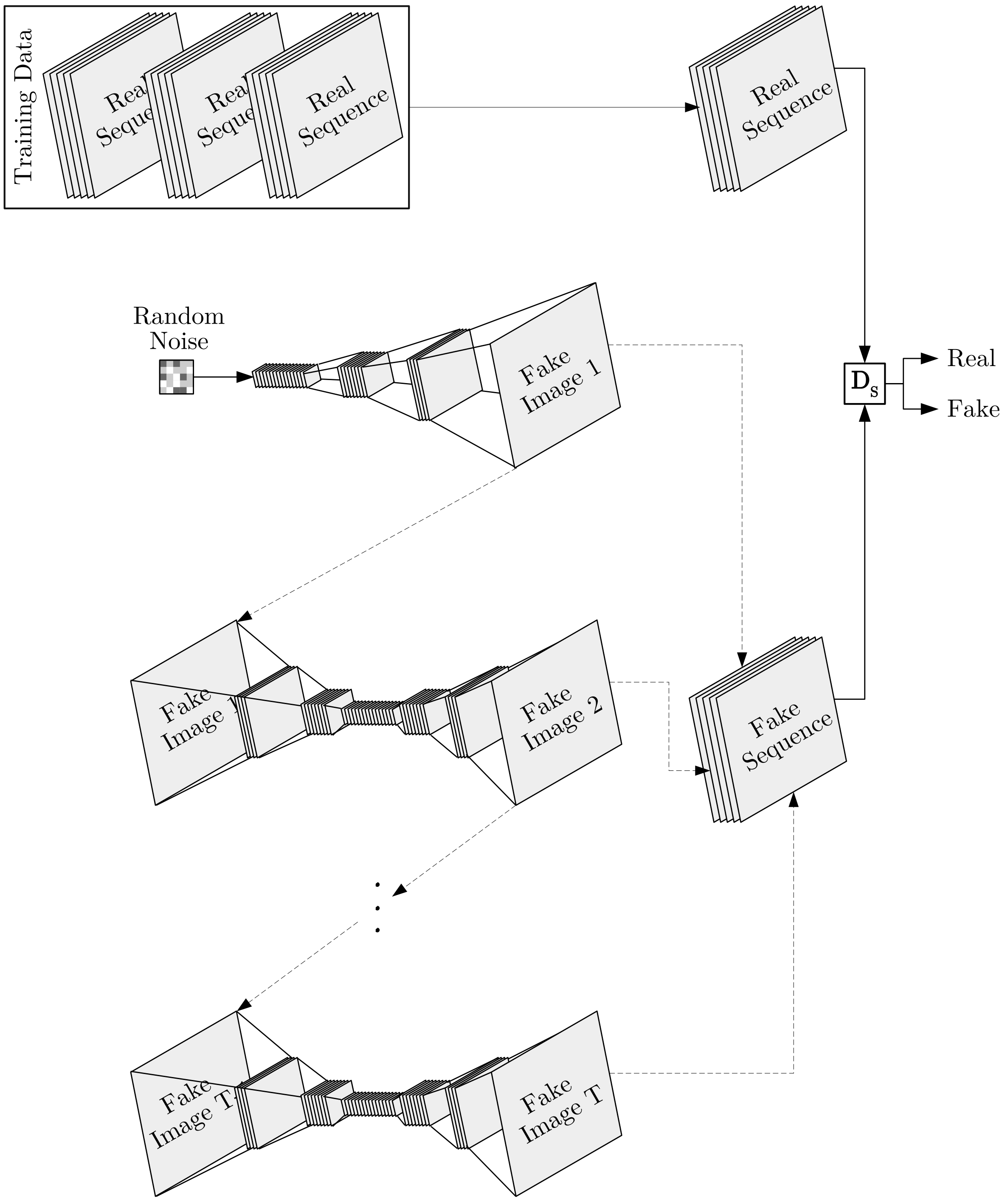


Figure 3.4.: Visualization of the sequential GAN training with a recurrent generator *G* and a sequence discriminator D_S that are trained jointly on a set of real training sequences.

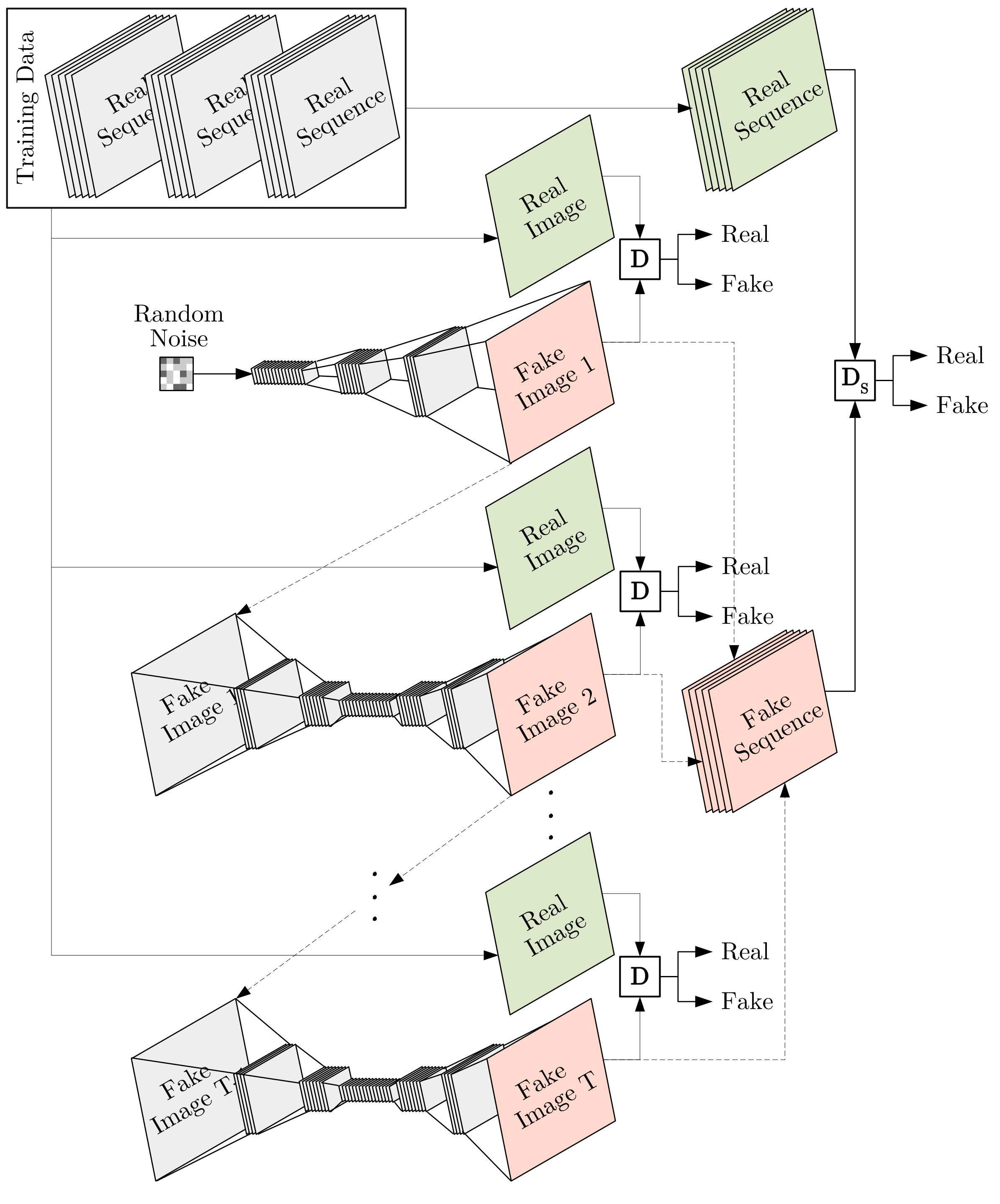


Figure 3.5.: Visualization of the proposed sequential GAN design consisting of a recurrent generator G, a per-frame discriminator D, and a sequence discriminator D_S.

General Motion Model For a general motion model definition, which can be applied to arbitrary tasks, we can define the conditioning inputs as the N_M previous generations in the respective motion direction:

$$c_t^f = \tilde{x}_{(t-N_M):(t-1)} \tag{3.11}$$

and

$$c_t^b = \tilde{x}_{(t+1):(t+N_M)}. \tag{3.12}$$

Thus, we use the sequence of $N_M + 1$ previous generations $\tilde{x}_{(t-N_M):t}$ to generate the next frame $\tilde{x}_{t+1}^f$ according to the motion present in the sequence. Similarly, we generate $\tilde{x}_{t-1}^b$ from $\tilde{x}_{(t:(t+N_M)}$.

Conditional Motion Model For certain tasks, we can adjust the conditioning input based on additional given information. In a conditional generation setting, such as paired or unpaired video-to-video translation, we can obtain the desired target location of the shift from the conditioning input of the GAN. Given a sequence of generations $\tilde{x}_{1:T}$ with corresponding conditioning $y_{1:T}$, we can then simply define the motion model conditioning inputs as

$$c_t^f = y_{t+1} \tag{3.13}$$

and

$$c_t^b = y_{t-1} \tag{3.14}$$

Thus, we shift $\tilde{x}_t$ to the location y_{t+1} for $\tilde{x}_{t+1}^f$ and to the location y_{t-1} for $\tilde{x}_{t-1}^b$.

Long-Range Shifts Instead of shifting a generation only for one time step at once, we can also define long-range shifts over k temporal steps as

$$\tilde{x}_{t+k}^{kf} = M(\tilde{x}_t | c_t^f, c_{t+1}^f, ..., c_{t+k-1}^f) \tag{3.15}$$

and

$$\tilde{x}_{t-k}^{kb} = M(\tilde{x}_t | c_t^b, c_{t-1}^b, ..., c_{t-k+1}^b). \tag{3.16}$$

3.2.2. Temporal Cycle-Consistency Loss

In order to make the generations of the motion model temporally consistent, we define a temporal cycle-consistency loss that requires the motion model to reconstruct its input through a backward-forward cycle. Given the motion model M and a generation $\tilde{x}_t$, we define *Temporal Cycle-Consistency* as the similarity of the reconstruction $\tilde{x}_t^{bf}$ to the input $\tilde{x}_t$, where $\tilde{x}_t^{bf}$ is the result of the backward-forward cycle

$$\begin{aligned} \tilde{x}_{t-1}^b &= M(\tilde{x}_t | c_t^b) \\ \tilde{x}_t^{bf} &= M(\tilde{x}_{t-1}^b | c_{t-1}^f)). \end{aligned} \tag{3.17}$$

Note that this idea is heavily inspired by the temporal tracking cycle of [24] as defined in Equations 2.29 and 2.30. Similar to [24], we also apply the same two types of long-range temporal cycles, as explained in the following.

Long Cycles As a reminder, in the long tracking cycle in [24] the tracker tracks an image patch repeatedly backwards for i-times and then forwards again for i time steps to obtain a reconstruction which is compared to the original image patch. Similarly, we define a long temporal cycle on our motion model by first successively shifting a generation $\tilde{x}_t$ backwards i-times with

$$\tilde{x}_{t-i}^{long} = M(...M(M(\tilde{x}_t|c_t^b)|c_{t-1}^b)...|c_{t-i+1}^b), \tag{3.18}$$

and then forwards i-times with

$$\tilde{x}_{t-i+i}^{long} = M(...M(M(\tilde{x}_{t-i}^{long}|c_{t-i}^f)|c_{t-i+1}^f)...|c_{t-1}^f). \tag{3.19}$$

Skip Cycles In the second type of tracking cycle in [24], the image patch is tracked to the target locations directly by performing a single large tracking step backwards and another single large step forwards to obtain the reconstruction. We can do this with our motion models by shifting a generation $\tilde{x}_t$ backwards in time for i steps at once, as defined in Equation 3.15, with

$$\tilde{x}_{t-i}^{skip} = M(\tilde{x}_t|c_t^b, c_{t-1}^b, ..., c_{t-i+1}^b), \tag{3.20}$$

and then forwards in time i steps at once, as defined in Equation 3.16, with

$$\tilde{x}_{t-i+i}^{skip} = M(\tilde{x}_{t-i}^{skip}|c_t^f, c_{t+1}^f, ..., c_{t+i-1}^f). \tag{3.21}$$

Loss Function Similar to [24], we calculate the temporal cycle-consistency loss $\mathcal{L}_{TC_{rec}}$ by applying the two types of long-range cycles to the last frame $\tilde{x}_T$ of each generated sequence $\tilde{x}_{1:T}$. Using N_{TC} to denote the longest temporal distance, we can then define the temporal cycle-consistency loss

$$\mathcal{L}_{TC_{rec}}(G, M) = \mathbb{E}_{\tilde{x}_{1:T}}[\frac{1}{N_{TC}} \sum_{i=1}^{N_{TC}} ||\tilde{x}_T - \tilde{x}_{T-i+i}^{long}||_1 + ||\tilde{x}_T - \tilde{x}_{T-i+i}^{skip}||_1], \tag{3.22}$$

where $\mathbb{E}_{\tilde{x}_{1:T}}$ denotes the expected value over generated sequences. Note again the similarity to the objective of [24] defined in Equation 2.35.

3.2.3. Prediction Similarity Loss

Through the temporal cycle-consistency loss, the motion model learns to generate temporally consistent outputs. In order to pass the temporal consistency to the GAN generator, we add another loss that compares the intermediate predictions of the motion model to the corresponding generator generations. Let us again use $\tilde{x}_{1:T}$ to denote a sequence of T generator generations on which we calculated the temporal cycle-consistency loss as defined in Equation 3.22, during which we computed intermediate predictions $\tilde{x}_{T-i}^{long}$ and $\tilde{x}_{T-i}^{skip}$ according to Equations 3.18 and 3.20 for all $i \in \{1, N_{TC}\}$. Because the motion model is temporally consistent, all $\tilde{x}_{T-i}^{long}$ and $\tilde{x}_{T-i}^{skip}$ are temporally consistent with $\tilde{x}_T$, so we can use the loss

$$\mathcal{L}_{TC_{pred}}(G, M) = \mathbb{E}_{\tilde{x}_{1:T}}[\frac{1}{N_{TC}} \sum_{i=1}^{N_{TC}} ||\tilde{x}_{T-i} - \tilde{x}_{T-i}^{long}||_1 + ||\tilde{x}_{T-i} - \tilde{x}_{T-i}^{skip}||_1] \tag{3.23}$$

to further constrain the generations $\tilde{x}_{(T-N_{TC}):(T-1)}$ to be temporally consistent with $\tilde{x}_T$.

3.2.4. Adversarial Motion Model Loss

Note that we use the L_1 distance for both the temporal cycle-consistency loss in Equation 3.22 and the prediction similarity loss in Equation 3.23. As a result, the outputs of the motion model might become blurry, which might then also compromise the image quality of the generations. To prevent this, we also apply the per-frame discriminator D of the GAN to all intermediate predictions $\tilde{x}_{T-i}^{long}$ and $\tilde{x}_{T-i}^{skip}$ and reconstructions $\tilde{x}_{T-i+i}^{long}$ and $\tilde{x}_{T-i+i}^{skip}$ to ensure they are realistic. Thereby, we obtain an additional adversarial loss term

$$\begin{aligned}\mathcal{L}_{GAN_{TC}}(G,M,D) = &\frac{1}{4}\mathbb{E}_x[\frac{1}{N_{TC}}\sum_{i=1}^{N_{TC}}\log(D(\tilde{x}_{T-i}^{long}))] + \frac{1}{4}\mathbb{E}_x[\frac{1}{N_{TC}}\sum_{i=1}^{N_{TC}}\log(D(\tilde{x}_{T-i}^{skip}))] \\ &+ \frac{1}{4}\mathbb{E}_x[\frac{1}{N_{TC}}\sum_{i=1}^{N_{TC}}\log(D(\tilde{x}_{T-i+i}^{long}))] + \frac{1}{4}\mathbb{E}_x[\frac{1}{N_{TC}}\sum_{i=1}^{N_{TC}}\log(D(\tilde{x}_{T-i+i}^{skip}))] \\ &+ \mathbb{E}_z[\log(1-D(G(z)))].\end{aligned} \tag{3.24}$$

3.2.5. TimeCycle Loss and TimeCycleGAN Objective

Let us combine both the temporal cycle-consistency loss $\mathcal{L}_{TC_{rec}}$ from Equation 3.22 and the prediction similarity loss $\mathcal{L}_{TC_{pred}}$ from Equation 3.23 to the *TimeCycle Loss*

$$\mathcal{L}_{TC}(G,M) = \mathcal{L}_{TC_{rec}}(G,M) + \mathcal{L}_{TC_{pred}}(G,M), \tag{3.25}$$

By adding both the TimeCycle loss, weighted by a factor of λ_{TC}, and the adversarial motion model loss from Equation 3.24 to the proposed sequential GAN objective defined in Equation 3.8, we then obtain the *TimeCycleGAN* objective

$$\begin{aligned}G^*, M^* = \underset{G,M}{\operatorname{argmin}}\{&\max_{D}\{\mathcal{L}_{GAN}(G,D) + \mathcal{L}_{GAN_{TC}}(G,M,D)\} \\ &+ \max_{D_S}\{\mathcal{L}_{sGAN}(G,D_S)\} \\ &+ \lambda_{TC}\mathcal{L}_{TC}(G,M)\},\end{aligned} \tag{3.26}$$

which will be the basis of all TimeCycleGAN models proposed in the following section. In Figure 3.6 we show a simple flow diagram of the TimeCycleGAN meta-architecture with the TimeCycle loss, and a detailed visualization can be seen in Figure 3.7.

3.3. TimeCycleGAN Models

Using the TimeCycleGAN approach, we design novel models for three different tasks: paired video-to-video translation, unpaired video-to-video translation, and unconditional video generation.

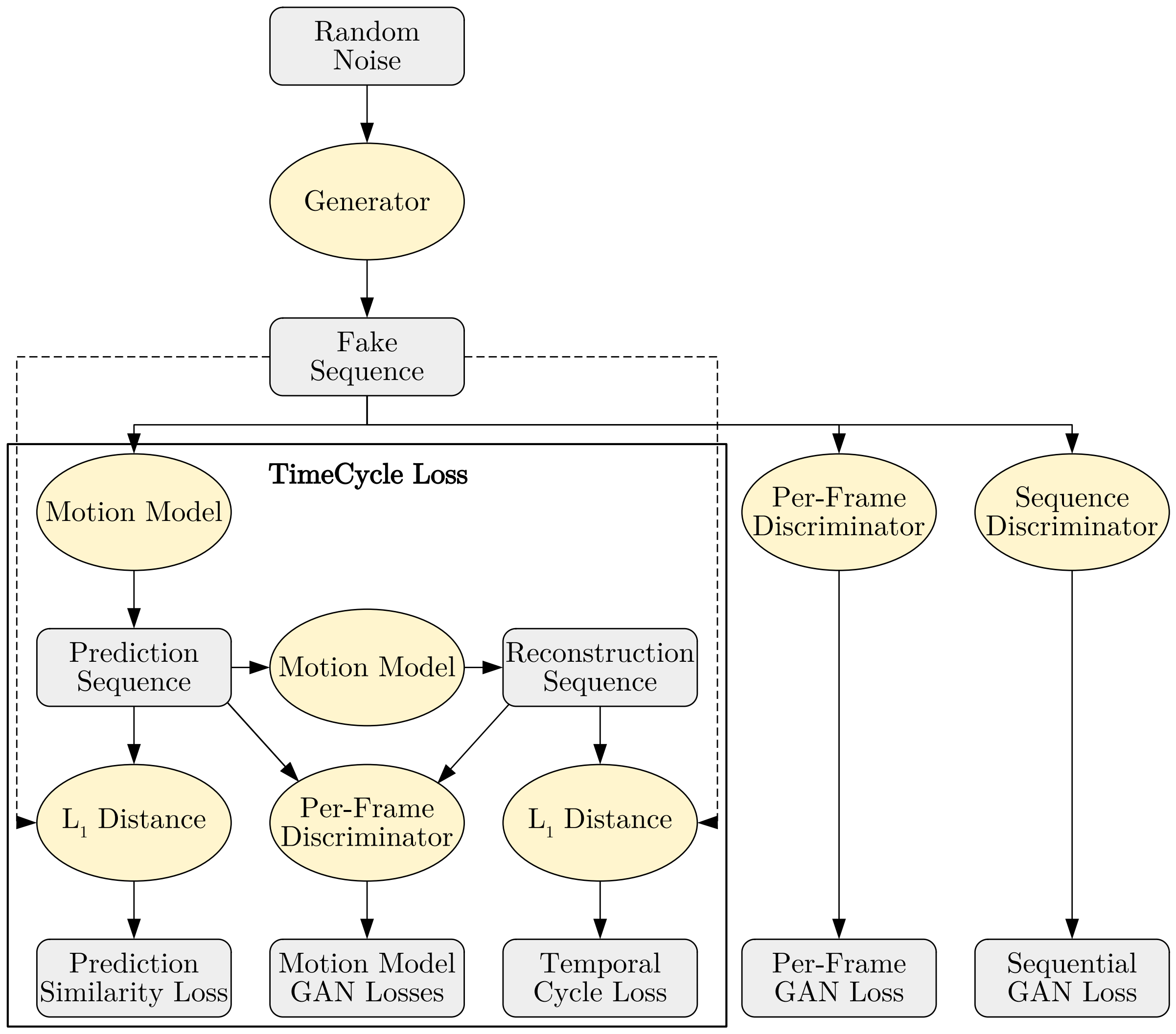


Figure 3.6.: A flow diagram of the TimeCycleGAN meta-architecture. The generator generates a fake sequence on which the TimeCycle loss and per-frame and sequential GAN losses are computed. Discriminator updates on real images are omitted for simplicity.

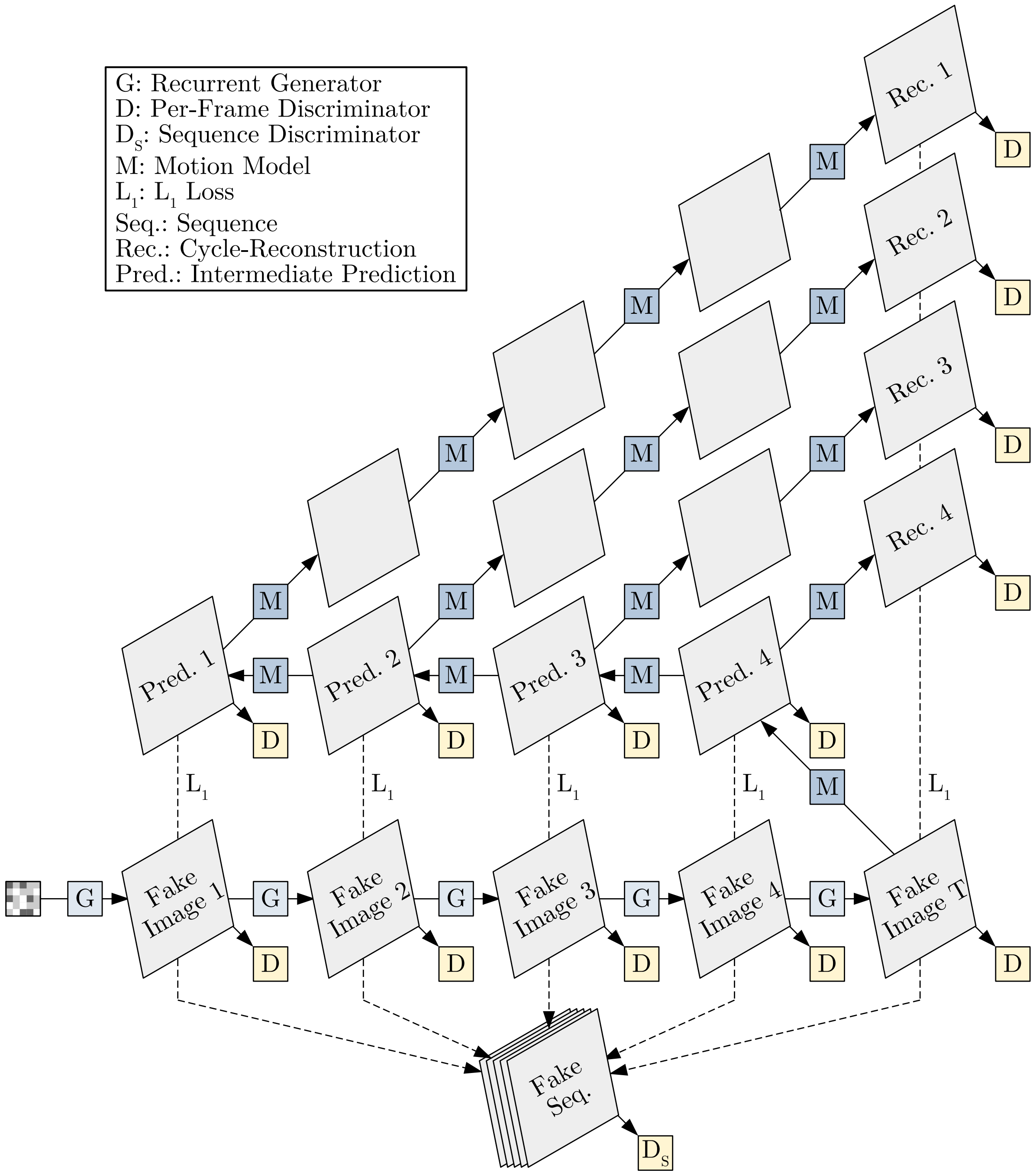


Figure 3.7.: A visualization of the TimeCycleGAN meta-architecture with long temporal cycle. The generator generates a fake sequence on which the TimeCycle loss and per-frame and sequential GAN losses are computed. Discriminator updates on real images and TimeCycle skip cycles are omitted for simplicity.

3.3.1. Overview of Video Generation Tasks

Unconditional Video Generation

In unconditional video generation, we are given several real target sequences, and the task is to learn the generation of similar sequences, without any further restrictions.

Paired Video-to-Video Translation

In paired video-to-video translation, we are additionally given a source image, such as a semantic segmentation map, for each target image. The generator is then conditioned on a sequence of source images, and, in addition to generating realistic videos, each frame of the generated video should also match the corresponding source image.

Unpaired Video-to-Video Translation

In unpaired video-to-video translation, we want to generate videos according to a given input source sequence, similar to paired video-to-video translation. However, instead of having sequences of source-target pairs $(y_t, x_t)_{t \in 1:T}$ during training, we only have two separate, unrelated sequences $x_{1:T}$ and $y_{1:T}$ of target and source images respectively, and we have no explicit information how a given target or source image should look like in the respective other domain. To perform this task, we typically train two generation methods for both translation directions jointly.

Comparison

In Figure 3.8 we show a comparison of the training data and learning objectives of the three video generation tasks. Note that, in terms of difficulty, having fewer additional inputs for a task makes it more difficult to generate realistic videos because there is less information available that can be used to estimate the correct motion of a generated sequence.

3.3.2. Unconditional TimeCycleGAN-U

For unconditional video generation, we apply the TimeCycleGAN meta-architecture to DCGAN, which we covered in Section 2.1.2. The resulting *TimeCycleGAN-U* (TCGAN-U) model is then trained with the default TimeCycleGAN objective

$$\begin{aligned} G^*, M^* = \operatorname*{argmin}_{G,M} \{ & \max_{D} \{ \mathcal{L}_{GAN}(G, D) + \mathcal{L}_{GAN_{TC}}(G, M, D) \} \\ & + \max_{D_S} \{ \mathcal{L}_{sGAN}(G, S_V) \} \\ & + \lambda_{TC} \mathcal{L}_{TC}(G, M) \}. \end{aligned} \tag{3.26 revisited}$$

In order to adjust the DCGAN generator to be recurrent according to Section 3.1.1, we add an image encoder network E before the generator, which encodes a given sequence of N_G previous generations $\tilde{x}_{(t-N_G):(t-1)}$ into a latent representation $\tilde{z}_t$ that has similar size as the

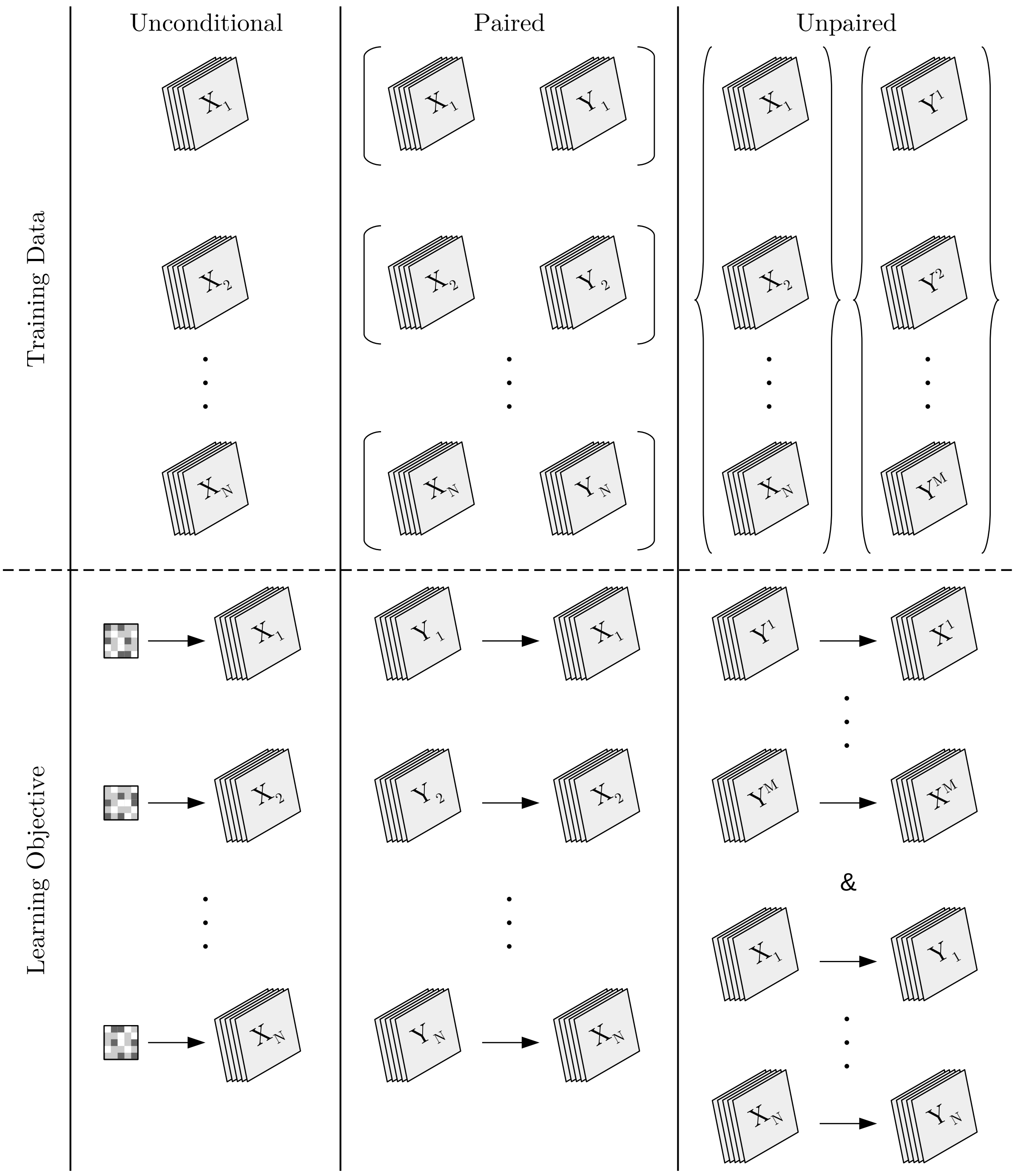


Figure 3.8.: Comparison of unconditional video generation (left), paired video-to-video translation (middle) and unpaired video-to-video translation (right). The top row shows the training data for each task: A set of sequences, a set of sequence pairs, and two sets of sequences from different domains. The bottom row shows the learning objective: Generating sequences from random noise and generating sequences from other sequences in one and two directions, respectively.

random noise input z_t that the standard image-wise DCGAN generator would usually be conditioned on. To obtain this image encoder, we simply transpose the generator architecture.

3.3.3. Unpaired TimeCycleGAN-UP

We use CycleGAN [2] from Section 2.1.5 as the basis of our unpaired *TimeCycleGAN-UP* (TCGAN-UP) model. As a reminder, CycleGAN consists of two GANs, (G_X, D_X) and (G_Y, D_Y), that are trained jointly with

$$\begin{aligned} G_X^*, G_Y^* = \underset{G_X, G_Y}{\operatorname{argmin}} \{ & \max_{D_X}\{\mathcal{L}_{GAN}(G_X, D_X)\} + \max_{D_Y}\{\mathcal{L}_{GAN}(G_Y, D_Y)\} \\ & + \lambda_C \mathcal{L}_C(G_X, G_Y) \\ & + \lambda_{idt}(\mathcal{L}_{idt}(G_X) + \mathcal{L}_{idt}(G_Y))\} \end{aligned} \tag{2.12 revisited}$$

where $\mathcal{L}_C(G_X, G_Y)$ is the cycle-consistency loss

$$\begin{aligned} \mathcal{L}_C(G_X, G_Y) = & \mathbb{E}_x[||G_X(G_Y(x)) - x||_1] \\ & + \mathbb{E}_y[||G_Y(G_X(y)) - y||_1]. \end{aligned} \tag{2.10 revisited}$$

and $\mathcal{L}_{idt}(G_X)$ and $\mathcal{L}_{idt}(G_Y)$ are the identity mapping losses

$$\begin{aligned} \mathcal{L}_{idt}(G_X) &= \mathbb{E}_x[||G_X(x) - x||_1] \\ \mathcal{L}_{idt}(G_Y) &= \mathbb{E}_y[||G_Y(y) - y||_1]. \end{aligned} \tag{2.11 revisited}$$

Objective Function After we apply the TimeCycleGAN meta-architecture from Equation 3.26 to both GANs (G_X, D_X) and (G_Y, D_Y), we obtain TimeCycleGAN-UP with objective

$$\begin{aligned} G_X^*, G_Y^*, M_X^*, M_Y^* = \underset{G_X, G_Y}{\operatorname{argmin}} \{ & \max_{D_X}\{\mathcal{L}_{GAN}(G_X, D_X) + \mathcal{L}_{GAN_{TC}}(G_X, M_X, D_X)\} \\ & + \max_{D_Y}\{\mathcal{L}_{GAN}(G_Y, D_Y) + \mathcal{L}_{GAN_{TC}}(G_Y, M_Y, D_Y)\} \\ & + \max_{D_{S,X}}\{\mathcal{L}_{sGAN}(G_X, D_{S,X})\} \\ & + \max_{D_{S,Y}}\{\mathcal{L}_{sGAN}(G_Y, D_{S,Y})\} \\ & + \lambda_{TC}(\mathcal{L}_{TC}(G_X, M_X) + \mathcal{L}_{TC}(G_Y, M_Y)) \\ & + \lambda_C \mathcal{L}_C(G_X, G_Y) \\ & + \lambda_{idt}(\mathcal{L}_{idt}(G_X) + \mathcal{L}_{idt}(G_Y))\} \end{aligned} \tag{3.27}$$

Note that we use separate motion models M_X and M_Y for the two domains, which are both conditional motion models, as defined in Section 3.2.1, conditioned on the sequence of the respective other domain. To make the generator recurrent, we simply increase its input dimension.

3.3.4. Paired TimeCycleGAN-P

For the task of paired video-to-video translation, we build upon the paired image-to-image translation method pix2pix, which we covered in Section 2.1.3. As a reminder, pix2pix is a conditional GAN with the objective

$$G^* = \underset{G}{\operatorname{argmin}}\{\max_{D}\{\mathcal{L}_{cGAN}(G,D)\} + \lambda_{L1}\mathcal{L}_{L1}(G)\}, \tag{2.6 revisited}$$

where $\mathcal{L}_{L1}$ is the L_1 loss

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[||x - G(y)||_1] \tag{2.5 revisited}$$

and $\mathcal{L}_{cGAN}(G, D$ is the conditional GAN loss

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log(D(x|y))] + \mathbb{E}_{z,y}[\log(1 - D(G(z|y)|y))]. \tag{2.4 revisited}$$

Objective Function We then apply the TimeCycleGAN approach in Equation 3.26 to pix2pix, resulting in the video-to-video translation model *TimeCycleGAN-P* (TCGAN-P) with the objective

$$\begin{aligned} G^*, M^* = \underset{G,M}{\operatorname{argmin}}\{&\max_{D}\{\mathcal{L}_{cGAN}(G,D) + \mathcal{L}_{cGAN_{TC}}(G,M,D)\} \\ &+ \max_{D_S}\{\mathcal{L}_{sGAN}(G,D_S)\}\} \\ &+ \lambda_{TC}\mathcal{L}_{TC}(G,M) \\ &+ \lambda_{L1}\mathcal{L}_{L1}(G)\}. \end{aligned} \tag{3.28}$$

Note that we also use the conditional GAN loss for the adversarial motion model loss here. Furthermore, we use a conditional motion model, as defined in Section 3.2.1, with source images as conditioning. Again, we simply increase the input dimension of the generator to make it recurrent.

3.3.5. Paired TimeCycleGAN-P++

Based on TimeCycleGAN-P, we also build another paired video-to-video translation model, *TimeCycleGAN-P*++ (TCGAN-P++), where we additionally apply *feature matching* [19] and *perceptual losses* [20] to make the training more stable and to further improve the image quality.

Feature Matching Loss

In the feature matching technique [19], the adversarial loss of the generator is modified such that the generated outputs should additionally match the statistics of the real data within the discriminator. This makes the training more stable because the generated images should now also lead to similar internal representations in the discriminator instead of only requiring a particular discriminator output.

Feature Matching Loss Function By computing the norm on the differences between corresponding feature activations in each layer of the discriminator, a feature matching loss on a conditional per-frame discriminator D can then be computed as

$$\mathcal{L}_{FM}(G,D) = \mathbb{E}_{x,y}[\sum_{i=1}^{L} \frac{1}{N_i} ||D^{(i)}(x|y) - D^{(i)}(\tilde{x}|y)||_1], \tag{2.7 revisited}$$

where y denotes the conditioning image, x the real image, $\tilde{x}$ the generated image, L the number of layers of D, $D^{(i)}$ the feature activations of the i-th layer of D, and N_i the number of neurons in the i-th layer. Note that, similar to pix2pixHD [10], we use the L1-norm because it is more robust to outliers than the L2-norm used in [19].

Feature Matching in Sequence Discriminators We calculate a feature matching loss for both the per-frame discriminator, as well as the sequence discriminator. Using similar notation, the feature matching loss of the unconditional sequence discriminator can be defined as

$$\mathcal{L}_{FM}(G,D_S) = \mathbb{E}_{x}[\sum_{i=1}^{L} \frac{1}{N_i} ||D_S^{(i)}(x_{t:(t+N_D-1)}) - D_S^{(i)}(\tilde{x}_{t:(t+N_D-1)})||_1], \tag{3.29}$$

where $x_{t:(t+N_D-1)}$ and $\tilde{x}_{t:(t+N_D-1)}$ are real and fake sequences of length N_D.

Perceptual Loss

For the perceptual loss [20], a convolutional neural network for visual perception is applied to both the ground truth and the corresponding generated image in paired image-to-image translation, and the loss is calculated as the difference of feature activations in some of the convolutional layers of the perception network. Similar to pix2pixHD [10], we use the VGG19 network [21] and compute the difference of feature activations in five of its layers, leading to the same perceptual loss

$$\mathcal{L}_{P}(G) = \mathbb{E}_{x,y}[\sum_{i \in \{2,7,12,21,30\}} \frac{1}{N_i} ||VGG^{(i)}(x) - VGG^{(i)}(G(y))||_1], \tag{2.8 revisited}$$

where $VGG^{(i)}$ are the feature activations of the i-th layer of the VGG19 network, and N_i is the number of neurons in the corresponding layer.

Perceptual Loss and Feature Matching Note that the perceptual loss is closely related to feature matching. In both losses, we compare the feature activations within a neural network between a generated image and the corresponding real image. The difference is that the feature matching loss uses the discriminator, and the perceptual loss an external pretrained perceptual network.

Perceptual Loss and L1 Loss While the loss formulation of the perceptual loss in Equation 2.8 seems very different from the pix2pix L_1 loss defined in Equation 2.5, both losses are, in fact, also closely related. In practice, both losses are helpful to stabilize the early adversarial training by providing an external image quality measurement that guides the optimization in the right direction, and both losses can also be used to improve global spatial image consistency. The perceptual loss that we defined in Equation 2.8 can even be seen as another form of the L1 loss, where the loss is calculated on the feature activations of some perceptual network instead of on the image itself, as shown in Figure 3.9, where we compare L_1, perceptual, and feature matching losses.

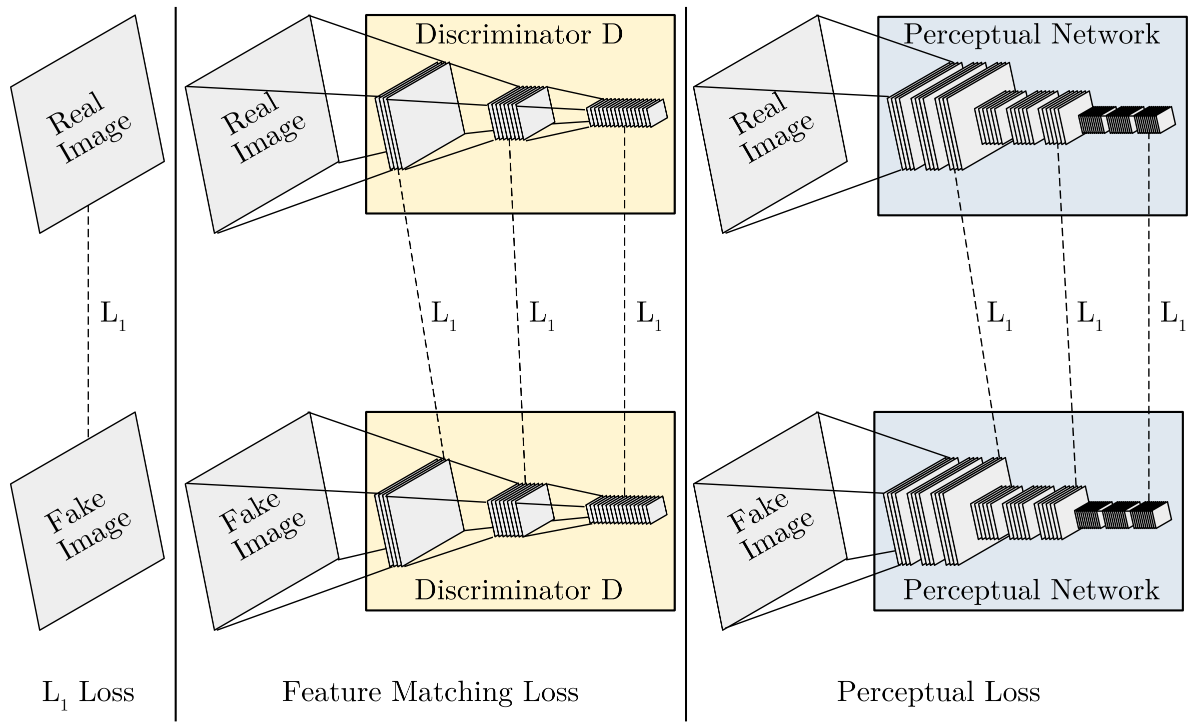


Figure 3.9.: Comparison of L_1 loss (left), feature matching loss (middle) and perceptual loss (right). The L_1 loss compares real to fake images directly. Feature matching and perceptual losses compare the feature activations of layers in the discriminator or a pretrained perceptual network, respectively.

Objective Function

By adding feature matching losses $\mathcal{L}_{FM}(G, D)$ and $\mathcal{L}_{FM}(G, D_S)$ of Equations 2.7 and 3.29, and a perceptual loss $\mathcal{L}_P(G)$ on VGG19 as defined in Equation 2.8, the TimeCycleGAN-P objective of Equation 3.28 is adjusted to the TimeCycleGAN-P++ objective

$$\begin{aligned}
G^*, M^* = \underset{G,M}{\operatorname{argmin}}\{ &\max_{D}\{\mathcal{L}_{cGAN}(G,D) + \mathcal{L}_{cGAN_{TC}}(G,M,D)\} \\
&+ \max_{D_S}\{\mathcal{L}_{sGAN}(G,D_S)\}\} \\
&+ \lambda_{FM}(\mathcal{L}_{FM}(G,D) + \mathcal{L}_{FM}(G,D_S)) \\
&+ \lambda_{TC}\mathcal{L}_{TC}(G,M) \\
&+ \lambda_{L1}\mathcal{L}_{L1}(G) \\
&+ \lambda_{P}\mathcal{L}_{P}(G)\}.
\end{aligned} \tag{3.30}$$

4. Implementation

4.1. Implementation Details

4.1.1. Network Architectures

Generators and Per-Frame Discriminators

Similar to CycleGAN [2] and RecycleGAN [12], we use a residual generator based on the architecture proposed by Johnson et al. [18] for all generators, and the pix2pix [1] patchGAN discriminators for all paired and unpaired video-to-video translation models. For 256×256 generation, we use six residual blocks in the generator and three layers in the discriminator. For 64×64 generation, we use three residual blocks in the generator and two layers in the discriminator and additionally divide all filter sizes by a factor of two. As described in Section 3.1.1, we condition the generators on previous generations, which we implement by increasing the input channel dimension of the networks accordingly. For the unconditional models, we use the standard DCGAN [4] networks, and we transpose the generator to obtain an image encoder for previous generations.

Sequence Discriminators

We also use the discriminator architecture described above for all sequence discriminators. However, our unconditional sequence discriminators are applied to sequences of N_D frames, so the input channel dimension is adjusted accordingly from $C_G + C_C$ to $N_D * C_G$, where C_G is the number of channels of generated images and C_C is the number of channels of conditioning images.

TimeCycle Motion Models

For the TimeCycle motion models, we use the same architecture as for all other generators, but we significantly reduce the number of layers of the network. In particular, we choose the number of residual blocks such that the receptive field of one generated pixel in the output image is roughly equal to one-fifth of the image size in each spatial dimension, e.g., for resolution 256×256 we use two residual blocks for a 45×45 receptive field, and for resolution 64×64 we use no residual blocks for a receptive field size of 13×13. The reduction of layers also leads to a reduction in network parameters to around ten percent. Note that the receptive field reduction factor of $\frac{1}{5}$ was specifically selected for the Cityscapes domain because the pixel-wise motions between frames do not exceed one-fifth of the image size. For domains with more extreme motions, a larger receptive field might be necessary because the motion model will otherwise not be able to reuse the relevant area of the previous generation.

4.1.2. Hyperparameter Choices

To train our models, we use the least-squares GAN objective [26], and two separate Adam optimizers [27] with coefficients $\beta_1 = 0.5$, $\beta_2 = 0.999$ for both generator and discriminator. All other hyperparameters for training on 64×64 images are listed in Table 4.1. For training on 256×256 images, we use batch size $BS = 1$ and, where necessary, lower T, N_D and N_{TS} to fit on a single GPU.

Hparam	Description	TCGAN-P	TCGAN-P++	TCGAN-UP	TCGAN-U
T	Sequence Length	5	5	5	5
N_G	Number of Inputs of G	1	1	1	1
N_D	Number of Inputs of D_S	5	5	5	5
N_{TS}	TimeCycle Length	4	4	4	4
λ_{L_1}	L1 Loss Weight	10	10	-	-
λ_{FM}	Feature Matching Loss Weight	0	5	-	-
λ_P	Perceptual Loss Weight	0	10	-	-
λ_C	Cycle-Consistency Loss Weight	-	-	10	-
λ_{TC}	TimeCycle Loss Weight	10	10	10	10
LR	Learning Rate	2e-4	2e-4	2e-4	2e-4
NE	Number of Epochs	40	40	40	40
BS	Batch Size	32	32	32	32

Table 4.1.: List of hyperparameters of all TimeCycleGAN models trained on 64×64 images.

4.1.3. Temporally-Progressive Training

Similar to vid2vid [11], we train our models in a temporally-progressive manner, where we divide the training into multiple stages while continuously increasing the lengths of generated sequences. In particular, we start with a sequence length of one. Therefore, we first train a model that can generate realistic individual frames, which we further fine-tune for temporal consistency. We found that such temporal growing stabilizes the training and leads to significantly faster convergence.

4.1.4. Sequence-Wise Data Augmentation

During training, we apply two techniques for data augmentation: random cropping, and random horizontal flipping. Instead of applying these augmentations on a per-frame level, we perform them sequence-wise, so that the randomized augmentation parameters are constant throughout the sequence.

Per-Sequence Parameter Sampling Given a sequence of T images of size $H \times W$, we first upscale all images to size $(H * 1.1) \times (W * 1.1)$. Then, we random sample the top left corner of the crop position (x_{crop}, y_{crop}) with $x_{crop} \in [0, W * 0.1]$ and $y_{crop} \in [0, H * 0.1]$, as well as a random binary value $\mathbb{1}_{flip} \in \{0, 1\}$ that indicates the horizontal flipping.

Sequence-Wise Random Cropping and Horizontal Flipping Using the sampled parameters, we then perform a random crop of size $H \times W$ on each upscaled frame of the sequence at location (x_{crop}, y_{crop}), and flip each crop horizontally if $\mathbb{1}_{flip} = 1$. Therefore, the augmentation is the same for all frames within the sequence, which is essential to have temporally consistent input sequences.

4.2. Applying TimeCycle Losses to Arbitrary GANs

As emphasized previously, our proposed TimeCycle loss is a general-purpose approach that can be applied to arbitrary GAN models. Furthermore, adding the TimeCycle loss to an existing PyTorch GAN model is very easy and does not require any modifications to the model itself if the model implementation follows a few simple assumptions. In the following, we will show how the TimeCycle loss can be added to custom GAN models.

4.2.1. TimeCycle Implementation

To separate the logic of the TimeCycle loss from the underlying model, we have implemented the TimeCycle using a Python mixin class, which is an abstract class that cannot be instantiated and provides additional functionality to subclasses that inherit from it. Thus, all it takes to apply the TimeCycle to an arbitrary GAN class is to inherit from the TimeCycle mixin and call its methods accordingly.

Required Methods In order to add a TimeCycle loss to a given GAN, we only need to call three methods:

- `define_timecycle()`: Needs to be called once in the beginning to define the motion models that shall be used for the TimeCycle.
- `timecycle_forward()`: Has to be called in every training iteration to define which sequence the TimeCycle should be run on.
- `timecycle_backward()`: Has to be called once every iteration to calculate the TimeCycle loss.

Note that all methods contain an argument `timecycle_name`. By using different values for this argument, arbitrarily many TimeCycles can be defined. Also note that `define_timecycle()` contains various arguments to customize the TimeCycle, such as `timecycle_type`, which can be used to specify special types of TimeCycle losses, such as the conditional TimeCycle loss used for our paired and unpaired video-to-video translation models, or the TecoGAN [13] Ping-Pong loss.

Optional Methods Additionally, the TimeCycle mixin contains several optional methods that can be used to train a given discriminator together with the TimeCycle, as we suggested in section 3.2.4:

- `add_timecycle_discriminator()`: Can be used to define discriminators that will be trained in an adversarial fashion with the TimeCycle motion models. If any discriminators are defined in this way, `timecycle_backward()` will automatically calculate all corresponding generator GAN losses.
- `set_timecycle_discriminator_input()`: Needs to be called in each iteration to define what real images and conditioning images a discriminator will use. The fake images do not need to be supplied, as these are already defined by `timecycle_forward()`.
- `timecycle_backward_D`: Computes the GAN loss on TimeCycle images for all discriminators that were defined with `add_timecycle_discriminator()`.

Note that all methods contain an argument `discriminator_name`, which can be used to define multiple discriminators for the same TimeCycle. Additionally, the TimeCycle mixin has a method `define_logging()`, which will return dictionaries of all TimeCycle images and losses to be used for Tensorboard logging.

4.2.2. Adding the TimeCycle Loss to an Existing GAN

Step-By-Step Guideline

The TimeCycle loss can be applied to any PyTorch GAN implementation in five easy steps:

1. If the current implementation is not object-oriented, wrap the GAN model in a class. We recommend the following structure:

```
class MyGAN:
    def forward():
    """Generate a sequence"""
        self.sequence = ...

    def backward():
    """Compute your losses for the sequence"""
        self.loss_D = ...
        self.loss_G = ...
    def optimize_parameters():
    """Update your losses"""
        # optimizer.zero_grad(), freeze models, ...
        self.loss_D.backward()
        self.loss_G.backward()
        # optimizer.step
```

Note that it is not required to have your code split in exactly these methods, you just need to have your per-iteraion code in separate methods, which could also be achieved by e.g. having a single method `train_iter()`. However, having the optimization split from the loss calculation is highly recommended, because it allows adding the TimeCycle losses in a separate subclass, as we do in the following steps, instead of having to modify the GAN model class itself.

2. Copy `timecycle_mixin.py` to your project, import it in the GAN implementation, and build a new subclass that inherits from both the GAN model and the TimeCycle mixin:

```
from .timecycle_mixin import TimecycleMixin
class MyGANWithTimecycle(TimecycleMixin, MyGAN):
    ...
```

3. Overwrite the `__init__()` method to define the TimeCycle, as well as optional discriminators that should be trained with it:

```
def __init__(*args, **kwargs):
    super().__init__(*args, **kwargs) # init your GAN and TimecycleMixin
    self.define_timecycle(...)
    self.add_timecycle_discriminator(...) # optional
```

4. Overwrite whichever method generates your sequence, add the TimeCycle forward pass, and, optionally, define inputs of the discriminators:

```
def forward(*args, **kwargs):
    super().forward(*args, **kwargs) # perform forward() of your GAN
    self.timecycle_forward(...)
    self.set_timecycle_discriminator_input(...) # optional
```

5. Overwrite whichever method calculates the losses and add the TimeCycle losses to the respective GAN losses:

```
def backward(*args, **kwargs):
    super().backward(*args, **kwargs) # perform backward() of your GAN
    self.loss_G += self.timecycle_backward(...)
    self.loss_D += self.timecycle_backward_D(...) # optional
```

Full Example

After performing all of the above steps, your new class should look like this:

```
class MyGANWithTimecycle(TimecycleMixin, MyGAN):

    def __init__(*args, **kwargs):
        super().__init__(*args, **kwargs) # init your GAN and TimecycleMixin
        self.define_timecycle(...)
        self.add_timecycle_discriminator(...) # optional

    def forward(*args, **kwargs):
        super().forward(*args, **kwargs) # perform forward() of your GAN
        self.timecycle_forward(...)
        self.set_timecycle_discriminator_input(...) # optional

    def backward(*args, **kwargs):
        super().backward(*args, **kwargs) # perform backward() of your GAN
        self.loss_G += self.timecycle_backward(...)
        self.loss_D += self.timecycle_backward_D(...) # optional
```

As we see, adding the TimeCycle loss to the custom GAN model only required 13 lines of code, three of which are optional. More complicated GAN architectures with multiple TimeCycles and multiple discriminators can be defined similarly easily, simply by repeating all TimeCycle method calls with different values for the `timecycle_name` and `discriminator_name` arguments.

5. Results and Evaluation

5.1. Datasets

5.1.1. Cityscapes

We use the Cityscapes dataset [28] as real target videos in all generation settings. Cityscapes is a dataset for semantic segmentation, containing a total of 5000 high-resolution 2048 $\times$ 1024 street-scene images recorded in various German cities, together with corresponding semantic segmentation maps.

Cityscapes Sequences In addition to the official dataset, Cityscapes provides various auxiliary material, including short 30-frame sequences surrounding each image of the official dataset. These sequences are what we will use to train all of our models.

Data Split We use the official training split of Cityscapes such that the training set consists of 2975 sequences, the validation set of 500 sequences, and the test set of 1525 sequences. With 30 frames per sequence, we, therefore, have 150000 total frames consisting of 89250 training frames, 15000 validation frames, and 45750 test frames.

Additional Demo Sequences In addition to the 5000 short 30-frame sequences, Cityscapes also provides three longer demo videos containing 600, 1100, and 1200 frames each, which allow us to observe long-term temporal consistency of a model. We use the first sequence for qualitative validation and the second and third sequences for qualitative testing.

Semantic Segmentation Estimation In contrast to the official 5000 Cityscapes images, all sequences and videos are unlabeled and do not contain any semantic segmentation information. Therefore, we use a DeepLabV3 [29] semantic segmentation network to estimate semantic segmentation maps for all sequences and videos. The DeepLabV3 model we used was pretrained on ImageNet [30] and finetuned on the 2975 labeled training frames of the official Cityscapes dataset.

Dataset Choice We selected the Cityscapes dataset specifically because it is a difficult dataset for temporal consistency learning. Unlike most other video generation datasets, the background of Cityscapes is dynamically changing, both the camera and the foreground objects are moving simultaneously, and both foreground and background objects vary considerably in size, shape, and type. Furthermore, due to the semantic segmentation estimation, the source images used for paired video-to-video generation are imperfect and not temporally consistent.

5.1.2. Carla

For unpaired video-to-video translation, we use synthetic street-scene videos as source images. For this, we created a novel dataset using the open-source CARLA simulator [31], version 0.9.6. Our generated dataset contains 200 simulated sequences with 200 frames each, captured in two different simulated cities: Town01 and Town02. Sequences are independent, and for each sequence, we randomly spawn 80 vehicles and 10 pedestrians that will populate the streets.

Data Split We split the generated sequences into train, validation, and test as follows: The train set contains 160 sequences, with 80 sequences captured in Town01 and the other 80 in Town02. Validation and test sets contain 20 sequences each, with 10 sequences from Town01 and 10 from Town02. Thus, our dataset contains 32000 training frames and 4000 frames for each validation and test.

5.2. Quantitative Evaluation Metrics

Throughout this chapter, we will use four quantitative metrics to evaluate the various GAN models. These metrics are:

- **Fréchet Inception Distance (FID)** [32] compares the distribution differences of InceptionNet [33] activations and is used to evaluate the overall image quality and image distribution similarity.
- **Learned Perceptual Image Patch Similarity (LPIPS)** [34] computes a weighted perceptual score between two images, based on AlexNet [35]. By computing the average LPIPS score between real and fake images in paired generation settings, we aim to evaluate the per-frame image quality and spatial consistency.
- **Temporal Distance on Learned Perception (tLP)** [13] measures the temporal consistency of videos by comparing the LPIPS differences of subsequent images. Similar to LPIPS, we compute the average tLP score over the entire test set.
- **Temporal Distance on Optical Flow (tOF)** [13] is the second temporal metric, which computes the difference of optical flows extracted from real and fake images. Again, we compute the average over the test set.

For all metrics, lower means better. In the following, we explain each metric in more detail:

5.2.1. Fréchet Inception Distance

Fréchet Inception Distance (FID) [32] has long been a standard score used to evaluate the performance of GANs. The Fréchet Inception Distance works as follows: First, an inception network [33] is inferenced on real images from the test set and on generated test images. Then, distributions of both outputs are generated, assuming Gaussian distributions for both. Finally, the Fréchet distance is calculated between the two distributions:

$$\mathrm{FID}(y, r) = \|\mu_r - \mu_y\|_2^2 + \mathrm{Tr}(\Sigma_r + \Sigma_y - 2(\Sigma_r \Sigma_y)^{\frac{1}{2}}), \tag{5.1}$$

where $\mathcal{N}(\mu_r, \Sigma_r)$ and $\mathcal{N}(\mu_y, \Sigma_y)$ are the Gaussian distributions of real and generated images, respectively.

FID and Mode Collapse By comparing the distributions of images, the FID score ensures that generated images are not only realistic but do also have a realistic image variance. Thus, the FID score punishes mode collapse.

FID and Adversarial Attacks However, the FID score also has a weakness: it is based on an inception network, which is itself imperfect. Thus, a higher score might not always correlate with better image quality. In particular, one could build adversarial samples for the inception network, which could look like random noise but achieve high FID scores.

5.2.2. Learned Perceptual Image Patch Similarity

The *Learned Perceptual Image Patch Similarity* (LPIPS) [34] is a metric that is closely related to the perceptual loss that we discussed in Section 3.3.5. Based on the finding that such perceptual losses coincide surprisingly well with differences in human perception, the LPIPS metric was introduced. Similar to the perceptual loss, LPIPS calculates the distance of deep feature activations in a pretrained perceptual network. However, instead of VGG19, the AlexNet [35] network architecture is used. Also, instead of the L1 distance, the L2 distance is used, and an additional weight layer is applied to the computed difference. This weight layer was learned using the *Berkeley Adobe Perceptual Patch Similarity* (BAPPS) dataset [34] and further improves the correlation between the deep feature activations and human perception.

Formula Given a real image x and a fake image $\tilde{x}$, the LPIPS score is computed as

$$LPIPS(x, \tilde{x}) = \sum_{i \in \{2,5,8,10,12\}} \frac{1}{N_i} ||w_i(AlexNet^{(i)}(x) - AlexNet^{(i)}(\tilde{x}))||_2, \tag{5.2}$$

where $AlexNet^{(i)}$ are the feature activations after the i-th layer in AlexNet, N_i is the number of neurons in the i-th layer, and w_i is the weight learned for the i-th layer.

Version There are several versions of the LPIPS metric. For all experiments in this paper, we use LPIPS version 0.1, with the default network AlexNet. This also applies to the tLP metric below.

5.2.3. Temporal Distance on Learned Perception

To assess the temporal consistency in paired and unpaired generation, two metrics were proposed in [13]: *tLP* and *tOF*. The first metric, tLP, measures the difference in LPIPS distances between subsequent frames. Given two subsequent real images x_t and x_{t+1} and corresponding fake images $\tilde{x}_t$ and $\tilde{x}_{t+1}$, tLP is computed as

$$tLP(x_t, x_{t+1}, \tilde{x}_t, \tilde{x}_{t+1}) = ||LPIPS(x_t, x_{t+1}) - LPIPS(\tilde{x}_t, \tilde{x}_{t+1})||_1, \tag{5.3}$$

where $LPIPS(x, \tilde{x})$ is the LPIPS score defined in Equation 5.2. For our final score, we calculate the tLP metric for all pairs of subsequent frames in the test set and average the results. We use the same version of LPIPS as described in Section 5.2.2.

5.2.4. Temporal Distance on Optical Flow

The second temporal metric proposed in [13] is tOF, where the difference between the estimated optical flows between two sequences is computed. Given two subsequent real images x_t and x_{t+1} and corresponding fake images $\tilde{x}_t$ and $\tilde{x}_{t+1}$, this can be expressed as

$$tOF(x_t, x_{t+1}, \tilde{x}_t, \tilde{x}_{t+1}) = ||OF(x_t, x_{t+1}) - OF(\tilde{x}_t, \tilde{x}_{t+1})||_1, \tag{5.4}$$

where $OF(x_t, x_{t+1})$ is the estimated optical flow between the two images x_t and x_{t+1}. Similar to tLP, we calculate tOF for all pairs of subsequent images and average the scores. Note that, instead of the optical flow estimation method by LucasKanade [36], we use FlowNet2 [22] to compute the optical flow between two images.

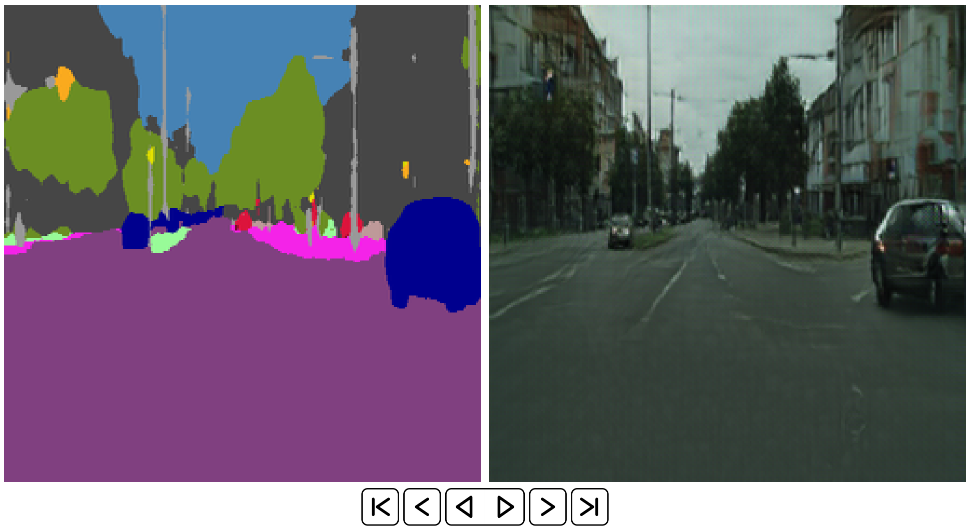
5.3. Main Results

5.3.1. Paired Video-to-Video Translation

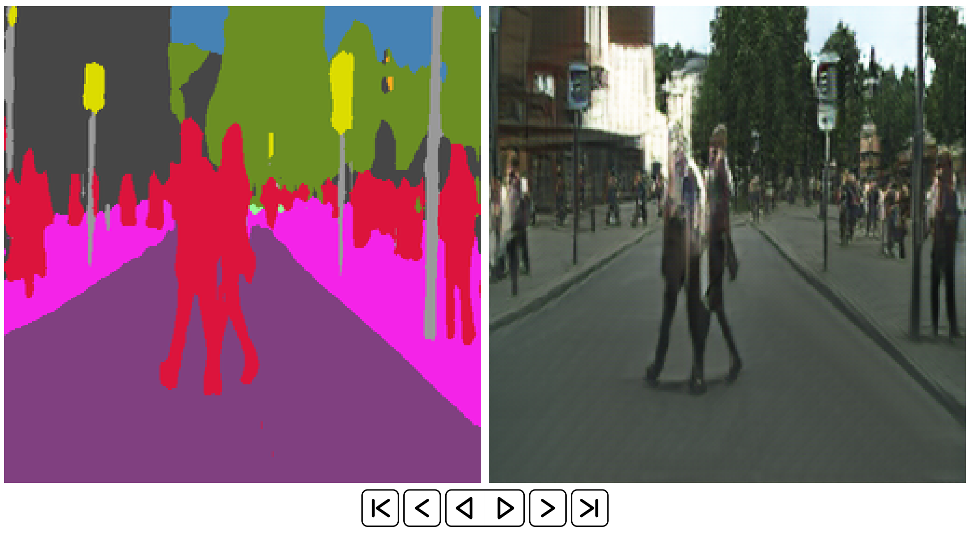
In our paired video-to-video translation setting we want to generate realistic street-scenes videos from given corresponding semantic segmentation maps. In Figure 5.1 we show videos generated by our strongest paired model, TimeCycleGAN-P++, on 256 × 256 images of the Cityscapes test demo sequences stuttgart_01 and stuttgart_02.

Qualitative Comparisons

We compare our models to both the baseline method pix2pix [1], as discussed in Section 2.1.3, as well as to the current state-of-the-art method vid2vid [11], which we covered in Section 2.2.1. We compare all methods on 64 × 64 images of the Cityscapes demo sequences stuttgart_01 and stuttgart_02, as shown in Figure 5.2. Compared to pix2pix, all other methods are significantly more temporally consistent. TimeCycleGAN-P++ seems similarly consistent as TimeCycleGAN-P, but the video quality is much better. The videos generated by vid2vid are sharper and the motion is more realistic than those of all other methods.

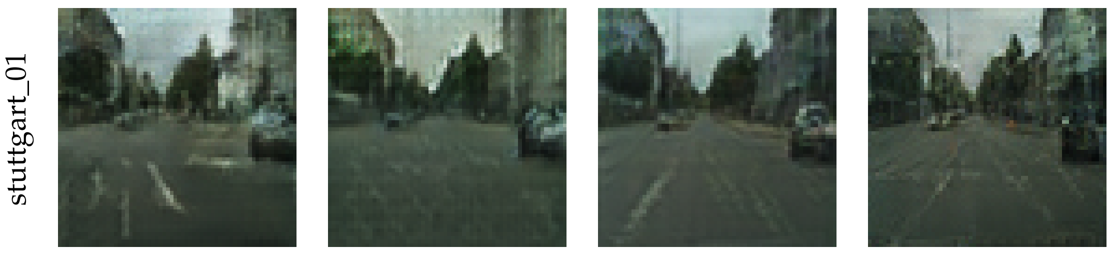


(a) Cityscapes demo sequence stuttgart_01

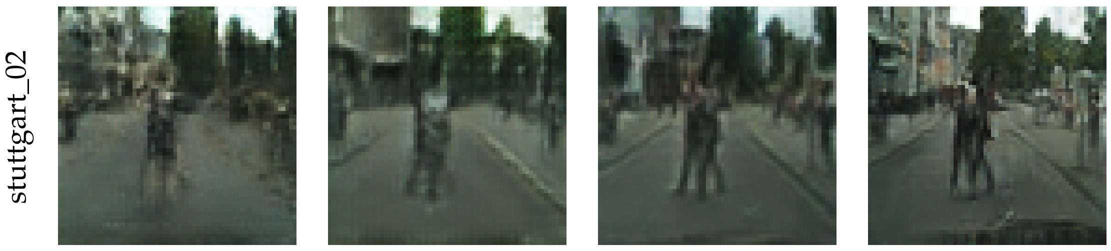


(b) Cityscapes demo sequence stuttgart_02

Figure 5.1.: Results of our best paired video-to-video translation model TimeCycleGAN-P++ on 256 × 256 Cityscapes test demo sequences. Top: demo sequence stuttgart_01. Bottom: demo sequence stuttgart_02. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*



(a) pix2pix **(b)** TCGAN-P **(c)** TCGAN-P++ **(d)** vid2vid



(a) pix2pix **(b)** TCGAN-P **(c)** TCGAN-P++ **(d)** vid2vid

Figure 5.2.: Qualitative comparison of paired video-to-video translation methods on 64 × 64 Cityscapes test sequences. Top: demo sequence stuttgart_01. Bottom: demo sequence stuttgart_02. Methods from left to right: pix2pix (baseline), TimeCycleGAN-P (ours), TimeCycleGAN-P++ (ours), vid2vid (state-of-the-art). All temporal methods are signficantly more temporally consistent than pix2pix. TimeCycleGAN-P++ and vid2vid additionally have higher video quality. Vid2vid produces the sharpest and most realistic results overall. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*

Quantitative Evaluation

To obtain quantitative results for all paired video-to-video translation methods, we inference all models on the Cityscapes test set and compute the four metrics FID, LPIPS, tLP, and tOF, as described in Section 5.2. The result of this comparison is shown in Table 5.1. As

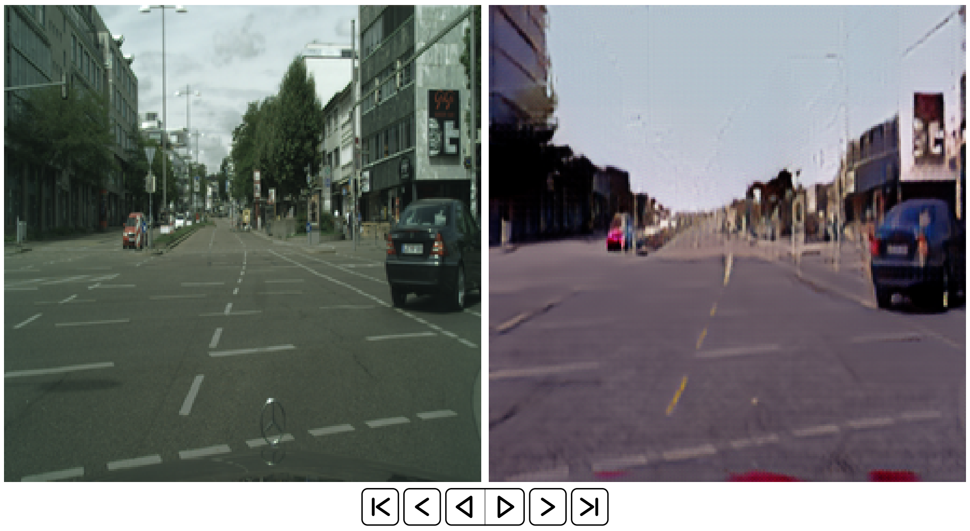
	Video Quality		Temporal Consistency	
Methods	FID	LPIPS	tLP	tOF
pix2pix	11.678e1	2.218e-1	20.938e-3	3.199e3
TCGAN-P	10.957e1	2.444 e-1	7.653e-3	2.902e3
TCGAN-P++	4.695e1	**1.967e-1**	**6.756e-3**	2.938e3
vid2vid	**3.259e1**	2.135e-1	7.382e-3	**1.720e3**

Table 5.1.: Quantitative comparison of paired video-to-video translation methods on the Cityscapes test set. FID and LPIPS metrics measure video quality, tLP and tOF measure temporal consistency. For all metrics, lower means better. Methods from top to bottom: pix2pix (baseline), TimeCycleGAN-P (ours), TimeCycleGAN-P++ (ours), vid2vid (state-of-the-art). TimeCycleGAN-P significantly improves temporal consistency of pix2pix while maintaining a similar video quality. TimeCycleGAN-P++ performs best in LPIPS and tLP, vid2vid performs best in FID and tOF. Overall, vid2vid is the best model, but TimeCycleGAN-P++ is close in both video quality and temporal consistency.

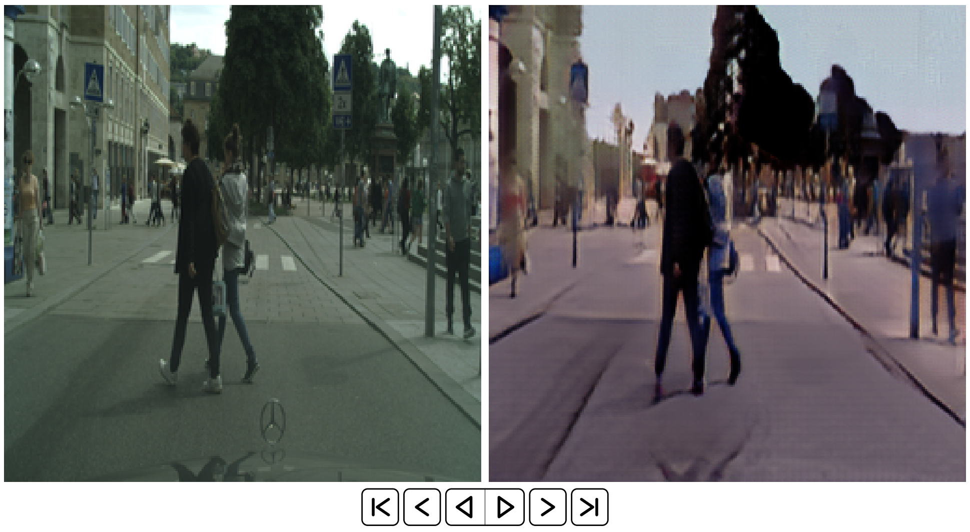
expected, all temporal models achieve significantly better scores than pix2pix in both temporal metrics, with TimeCycleGAN-P++ achieving the lowest tLP score, and vid2vid achieving the lowest tOF score. TimeCycleGAN-P is comparable to pix2pix in video quality, while TimeCycleGAN-P++ and vid2vid are significantly better, likely due to the added perceptual loss in both methods. Overall, TCGAN-P++ and vid2vid seem to be the best models, with both achieving the lowest score in two metrics each. However, while the LPIPS and tLP scores of TimeCycleGAN-P++ are only slightly below vid2vid, vid2vid performs significantly better in both FID and tOF. Thus, vid2vid is the best-performing model overall in both video quality and temporal consistency, confirming our previous qualitative analysis.

5.3.2. Unpaired Video-to-Video Translation

In unpaired video-to-video translation we jointly learn the generation of target sequences from source sequences with the generation of source sequences from target sequences. Using synthetic CARLA data as source, and Cityscapes as target, our unpaired models are then able to perform both CARLA-to-Cityscapes and Cityscapes-to-CARLA translation. In Figures 5.3 and 5.4 we show 256 $\times$ 256 test results of our TimeCycleGAN-UP model in both translation directions.



(a) Cityscapes demo sequence stuttgart_01



(b) Cityscapes demo sequence stuttgart_02

Figure 5.3.: Results of our unpaired TimeCycleGAN-UP model in Cityscapes-to-CARLA translation on two 256 × 256 Cityscapes test demo sequences. Top: demo sequence stuttgart_01. Bottom: demo sequence stuttgart_02. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*

Qualitative Comparisons

For qualitative comparisons in CARLA-to-Cityscapes translation, we inference all methods on the CARLA test set. For Cityscapes-to-CARLA translation, we inference all methods on the Cityscapes demo sequences stuttgart_01 and stuttgart_02. In Figures 5.5 and 5.6 we show comparisons of the sequences generated by each method in the respective translation direction. RecycleGAN and TimeCycleGAN-UP are more temporally consistent than CycleGAN in both translation directions. However, CycleGAN seems to translate the individual input frames more accurately, as background objects are sometimes ignored in the other two methods. In CARLA-to-Cityscapes generation, the results of both RecycleGAN and TimeCycleGAN-UP seem comparatively colorless. RecycleGAN seems to generate no color whatsoever, while TimeCycleGAN-UP seems only to preserve the color of foreground objects. For RecycleGAN, temporal style drifts can be observed in Cityscapes-to-CARLA generation, while TimeCycleGAN-UP maintains the same style throughout each generated sequence.

Quantitative Evaluation

To obtain quantitative results in unpaired video-to-video translation, we evaluate all models in both source-to-target and target-to-source generation. These two tasks are in the following abbreviated as *real* and *syn.*, which we define as:

- **real**: Generation of real images from synthetic images (CARLA $\rightarrow$ Cityscapes)
- **syn.**: Generation of synthetic images from real images (Cityscapes $\rightarrow$ CARLA)

Metrics Out of the four evaluation metrics of Section 5.2, only the FID score can be computed in settings other than paired video-to-video translation. However, as proposed by [13], we can approximate the tOF and tLP metrics in unpaired video-to-video translation by computing the respective temporal difference with regard to the conditioning sequence instead, i.e. by

$$tLP(y_t, y_{t+1}, \tilde{x}_t, \tilde{x}_{t+1}) = ||LPIPS(y_t, y_{t+1}) - LPIPS(\tilde{x}_t, \tilde{x}_{t+1})||_1, \tag{5.5}$$

and

$$tOF(y_t, y_{t+1}, \tilde{x}_t, \tilde{x}_{t+1}) = ||OF(y_t, y_{t+1}) - OF(\tilde{x}_t, \tilde{x}_{t+1})||_1, \tag{5.6}$$

for conditioning inputs y_t and y_{t+1} and corresponding generations $\tilde{x}_t$ and $\tilde{x}_{t+1}$.

Evaluation Results We compute these three metrics for both tasks on the test set of the respective translation direction. The corresponding evaluation results are shown in Table 5.2. Both RecycleGAN and TimeCycleGAN-UP improve upon the temporal consistency of CycleGAN considerably, with great improvements in tLP scores. However, the video quality of RecycleGAN is worse than that of CycleGAN, and, surprisingly, CycleGAN also has lower tOF scores. Overall, TimeCycleGAN-UP is comparable to CycleGAN in video quality and comparable to RecycleGAN in temporal consistency. Thus, TimeCycleGAN-UP is the best performing model overall.

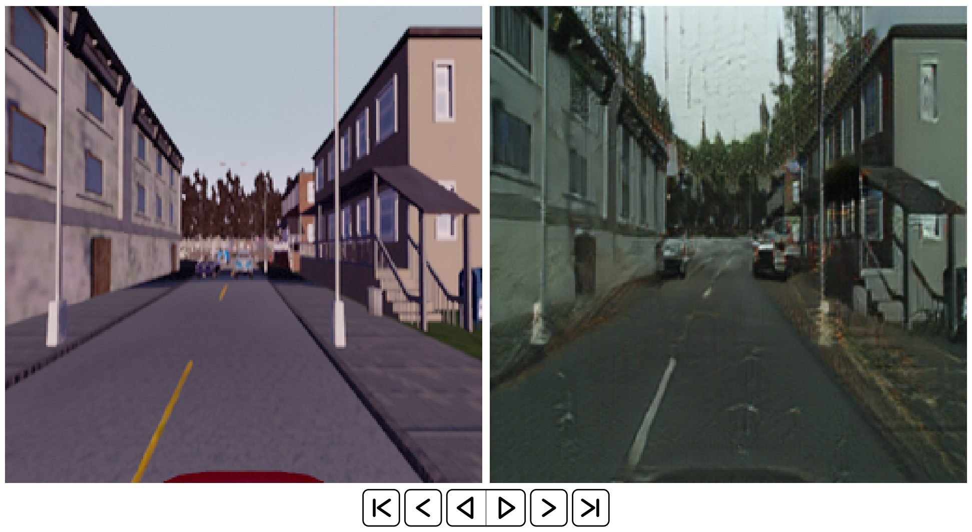


Figure 5.4.: Results of our unpaired TimeCycleGAN-UP model in CARLA-to-Cityscapes translation on three 256 × 256 CARLA test set sequences. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*

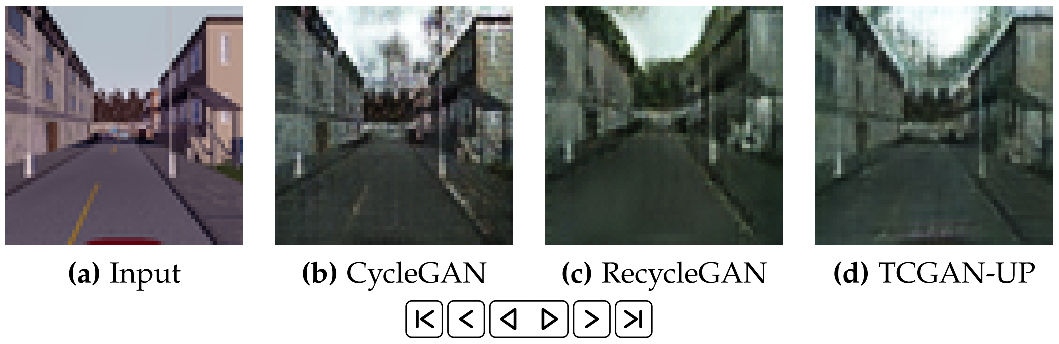
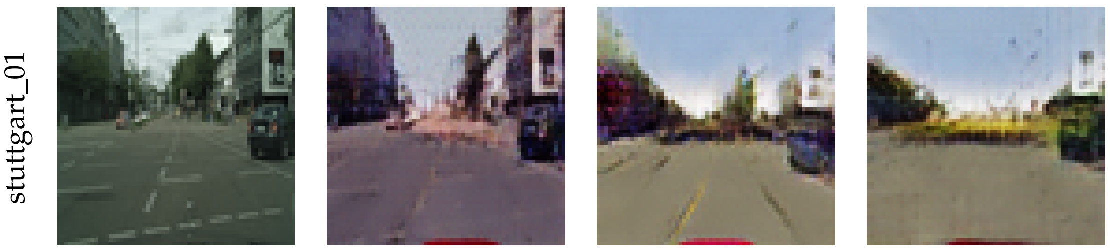
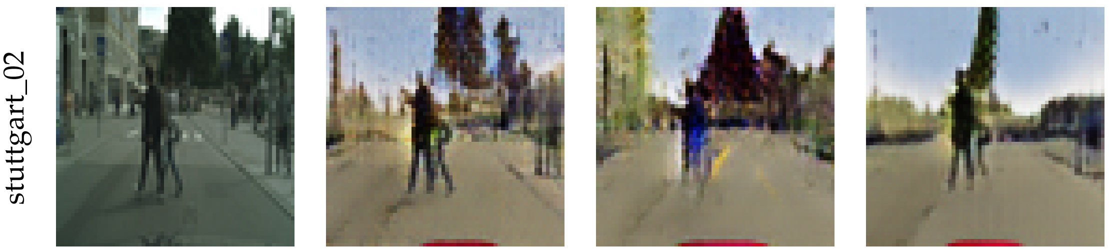


Figure 5.5.: Qualitative comparison of unpaired video-to-video translation methods in CARLA-to-Cityscapes translation on three 64 × 64 CARLA test set sequences. From left to right: CycleGAN (baseline), RecycleGAN (state-of-the-art), TimeCycleGAN-UP (ours). RecycleGAN and TimeCycleGAN-UP are more temporally consistent than CycleGAN, but result in a lack of color. The generations of RecycleGAN seem entirely colorless. In TimeCycleGAN-UP the color information of foreground objects is preserved. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*



(a) Input **(b)** CycleGAN **(c)** RecycleGAN **(d)** TCGAN-UP



(a) Input **(b)** CycleGAN **(c)** RecycleGAN **(d)** TCGAN-UP

Figure 5.6.: Qualitative comparison of unpaired video-to-video translation methods in Cityscapes-to-CARLA translation on Cityscapes test sequences. Top: demo sequence stuttgart_01. Bottom: demo sequence stuttgart_02. Methods from left to right: CycleGAN (baseline), RecycleGAN (state-of-the-art), TimeCycleGAN-UP (ours). RecycleGAN and TimeCycleGAN-UP are more temporally consistent than CycleGAN, but translate the individual frames less accurately. For RecycleGAN, temporal style drifts can be observed, while TimeCycleGAN-UP maintains the same style throughout the sequence. *The figure is best viewed with Acrobat Reader. Click on an image to play the video clip.*

	Video Quality		Temporal Consistency			
	FID		tLP		tOF	
Methods	real	syn.	real	syn.	real	syn.
CycleGAN	8.992e1	**19.686e1**	15.126e-3	11.142e-3	**3.888e3**	**2.284e3**
RecycleGAN	15.915e1	21.682e1	6.992e-3	**5.968e-3**	4.288e3	2.791e3
TCGAN-UP	**8.232e1**	21.997e1	**5.696e-3**	8.403e-3	3.976e3	2.468e3

Table 5.2.: Quantitative comparison of unpaired video-to-video translation methods in CARLA-to-Cityscapes generation (real) and Cityscapes-to-CARLA generation (syn.). FID measures video quality. tLP and tOF measure temporal consistency and are computed with regard to the conditioning sequence. For all metrics, lower means better. Methods from top to bottom: CycleGAN (baseline), RecycleGAN (state-of-the-art), TimeCycleGAN-UP (ours). RecycleGAN performs significantly better than CycleGAN in temporal consistency, but worse in video quality. TimeCycleGAN-UP is comparable to RecycleGAN in temporal consistency and close to CycleGAN in video quality. Thus, TimeCycleGAN-UP is the best model overall.

5.3.3. Unconditional Video Generation

In unconditional video generation, we simply inference the models several times in succession to obtain a generation sequence. In Figure 5.7 we show generated sequences of both DCGAN and TimeCycleGAN-U. As can be seen, the generations of DCGAN are not temporally consistent. While TimeCycleGAN-U correctly learned temporal consistency, it is not required to generate any motion in this setting, and, thus, learns to generate the easiest form of temporal consistency - absolute mode collapse.

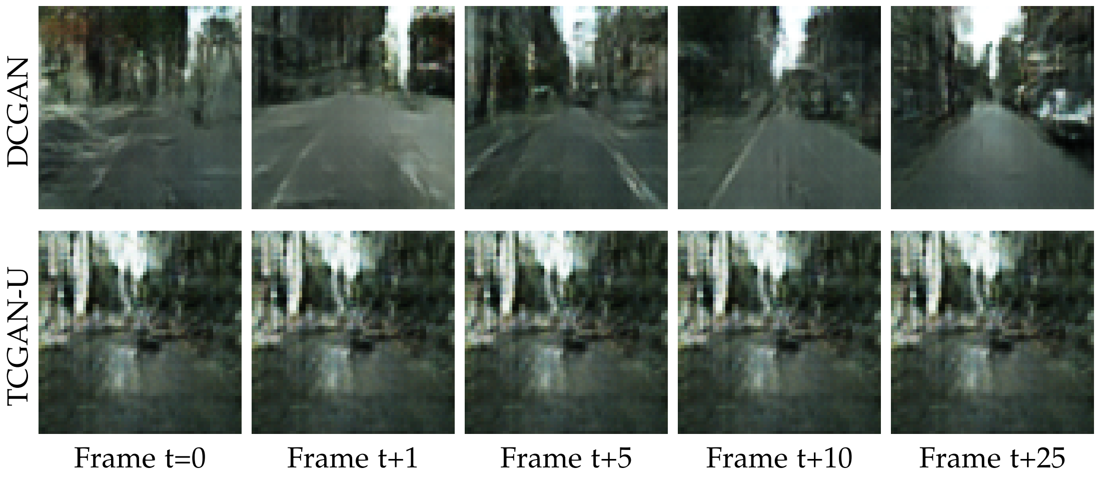


Figure 5.7.: Qualitative comparison of unconditional video generation methods on five generated frames (t=0, t+1, t+5, t+10, t+25). Top: DCGAN (baseline), bottom: TimeCycleGAN-U (ours). DCGAN is not temporally consistent, TCGAN-U learned total mode collapse.

5.4. Ablation Studies

5.4.1. TimeCycleGAN Components

To assess the effect of each component, we iteratively add them to a baseline model and measure the corresponding metric changes. Starting with the baseline, we first adjust the generator to be recurrent (G_R), then we add the unconditional sequence discriminator (G_R+D_S), followed by the TimeCycle loss (G_R+D_S+TC). We also build a model without sequence discriminator (G_R+TC) to check whether the TimeCycle loss requires a sequence discriminator or whether it can also improve temporal consistency on its own. Thus, the five models we compare are:

- Baseline: baseline model (pix2pix in paired setting, CycleGAN in unpaired setting)
- G_R: baseline model with recurrent generator
- G_R + D_S: baseline model with recurrent generator and sequence discriminator
- G_R + TC: baseline model with recurrent generator and TimeCycle loss
- G_R + D_S + TC: full TimeCycleGAN model (baseline model with recurrent generator, sequence discriminator and TimeCycle loss)

Paired Video-to-Video Translation

For the ablation study in paired video-to-video translation, we choose pix2pix [1] as baseline method such that the full model (G_R+D_S+TC) is exactly our proposed TimeCycleGAN-P, as covered in Section 3.3.4. We use the same setup as used in Section 5.3.1 and the hyperparameters listed in Table 4.1 for all models. Since the baseline model is trained on individual frames instead of sequences of length T, we increase the batch size of the baseline method by a factor of T such that the total number of parameter update steps is the same for all models. The result of the ablation study is shown in Table 5.3. As we can see, the recurrent generator design significantly improves the tLP metric, while all other metrics remain mostly unchanged. Adding the sequence discriminator further improves the tLP metric significantly, and also slightly lowers the tOF metric, however at the cost of significantly worse video quality as measured by both FID and $LPIPS$. Adding the additional TimeCycle loss mitigates the FID increase and further improves the tOF metric. Adding a TimeCycle loss without sequence discriminator also improves both temporal metrics and does not lead to an increase in FID. Note that each individual component leads to better scores in both temporal metrics. The best model overall is the full model with all three components, which achieves a 63.5 percent reduction in tLP and a 9.2 percent reduction in tOF compared to the baseline method.

Unpaired Video-to-Video Translation

In the unpaired video-to-video translation setting we choose CycleGAN [2] as baseline method such that the full model (G_R+D_S+TC) is the TimeCycleGAN-UP model described in

Section 3.3.3. We use the same setup as Section 5.3.2. Again, we use the hyperparameters listed in Table 4.1 for all models and increase the batch size of the baseline method by a factor of T such that the total number of parameter update steps is the same for all models. Ablation study results are shown in Table 5.4. Similar to the paired setting, we can observe that the recurrent generator significantly improves the tLP metric for generations in both directions. Additionally, it also improves *tOF* scores slightly. Adding the additional sequence discriminator again yields further improvements in tLP at the cost of worsened FID scores. However, unlike in the paired setting, the sequence discriminator seems to worsen the *tOF* metric considerably, and adding the additional TimeCycle loss leads to further deterioration in both FID and *tOF* instead of mitigating the issues. The TimeCycle loss without sequence discriminator seems to perform better in this setting, as it leads to a comparable reduction in tLP without affecting the video quality. Overall, the two best methods in this setting are the model with recurrent generator only and the model with recurrent generator and TimeCycle loss. Both result in comparable video quality, while the former performs better in *tOF* and the latter in tLP, reducing tLP by 46.3 percent compared to the baseline.

5.4.2. TimeCycle Loss Types

In Section 3.2 we proposed two versions of the TimeCycle loss: A general-purpose unconditional version where the motion model predicts subsequent frames from the two previous generations, and a task-specific version for conditional generation where the motion model shifts the previous generation to the location given by the conditioning input. The former was used in our unconditional TimeCycleGAN-U model, and the latter in our conditional TimeCycleGAN-UP, TimeCycleGAN-P and TimeCycleGAN-P++ models. In the following, we will compare the two types in the paired video-to-video translation setting with similar setup as Section 5.3.1 and same hyperparameters as shown in Table 4.1. Additionally, we compare them to the TecoGAN [13] Ping-Pong loss, which can be seen as another type of specialized TimeCycle loss, as we argue in Section A.2.1. Since the original Ping-Pong loss uses the L_2 distance, we perform all comparisons with both L_1 and L_2 loss formulations. Thus, the seven models we compare are:

- Baseline: baseline model (TimeCycleGAN-P without TimeCycle loss)
- *TC-C* L_1: baseline model with conditional TimeCycle loss with L_1 distance
- *TC-C* L_2: baseline model with conditional TimeCycle loss with L_2 distance
- *TC-U* L_1: baseline model with unconditional TimeCycle loss with L_1 distance
- *TC-U* L_2: baseline model with unconditional TimeCycle loss with L_2 distance
- *PP* L_1: baseline model with Ping-Pong loss with L_1 distance
- *PP* L_2: baseline model with Ping-Pong loss with L_2 distance

	Video Quality		Temporal Consistency	
Model	FID	LPIPS	tLP	tOF
Baseline	1.168e2	**2.218e-1**	20.938e-3	3.199e3
G_R	1.132e2	2.229e-1	14.482e-3	3.185e3
$G_R + D_S$	1.781e2	2.441e-1	**7.652e-3**	3.006e3
$G_R + TC$	1.138e2	2.518e-1	12.212e-3	3.161e3
$G_R + D_S + TC$	**1.096e2**	2.444 e-1	7.653e-3	**2.902e3**

Table 5.3.: Ablation study of TimeCycleGAN components in the paired video-to-video translation setting. Methods from top to bottom: baseline, baseline with recurrent generator, baseline with recurrent generator and sequence discriminator, baseline with recurrent generator and TimeCycle loss, baseline with all three components. Each individual component reduces both *tLP* and *tOF* metrics. The model with all three components performs best overall.

	Video Quality		Temporal Consistency			
	FID		tLP		tOF	
Model	real	syn.	real	syn.	real	syn.
Baseline	8.992e1	19.686e1	15.126e-3	11.142e-3	3.888e3	2.284e3
G_R	10.272e1	**18.643e1**	9.840e-3	9.041e-3	**3.870e3**	**2.181e3**
$G_R + D_S$	13.751e1	20.999e1	11.776e-3	**5.938e-3**	4.018e3	2.483e3
$G_R + TC$	**8.232e1**	21.997e1	**5.696e-3**	8.403e-3	3.976e3	2.468e3
$G_R + D_S + TC$	16.261e1	23.490e1	8.669e-3	8.122e-3	4.124e3	2.666e3

Table 5.4.: Ablation study of TimeCycleGAN components in the unpaired video-to-video translation setting. Methods from top to bottom: baseline, baseline with recurrent generator, baseline with recurrent generator and sequence discriminator, baseline with recurrent generator and TimeCycle loss, baseline with all three components. The recurrent generator design leads to improvements in both temporal metrics. Sequence discriminator and TimeCycle loss improve *tLP* but worsen *tOF*. The sequence discriminator also worsens *FID* significantly. Combining sequence discriminator and TimeCycle loss performs worse than both components individually. The best models are the model with recurrent generator only, and the model with recurrent generator and TimeCycle loss.

The results of the study can be seen in Table 5.5. As shown, the L_1 version of each loss type performs significantly better than the L_2 version with regard to both FID and $LPIPS$. For both the conditional TimeCycle loss and the Ping-Pong loss, the L_1 version also achieves noticeable improvements in tLP, while the L_2 version even leads to worse results than the baseline in all metrics except tOF. Thus, the L_1 loss formulations are strictly preferable for all loss types. Among all the L_1 loss type versions, the conditional TimeCycle loss achieves the lowest FID and tOF score, the unconditional TimeCycle loss the lowest tLP score, and the Ping-Pong loss the lowest $LPIPS$ score. Overall, the unconditional TimeCycle loss performs best with regard to temporal consistency, but the other two loss types perform slightly better with regard to video quality.

	Video Quality		Temporal Consistency	
Model	FID	LPIPS	tLP	tOF
Baseline	1.781e2	2.441e-1	7.652e-3	3.006e3
TC-C L_1	**1.096e2**	2.444e-1	7.653e-3	**2.902e3**
TC-C L_2	2.041e2	2.586e-1	8.022e-3	2.951e3
TC-U L_1	1.340e2	2.339e-1	6.507e-3	3.015e3
TC-U L_2	1.725e2	2.428e-1	**6.453e-3**	3.072e3
PP L_1	1.413e2	**2.190e-1**	7.423e-3	2.976e3
PP L_2	1.926e2	2.663e-1	9.870e-3	2.965e3

Table 5.5.: Comparison of TimeCycle loss types. From top to bottom: baseline model, baseline model with conditional L_1 TimeCycle loss, baseline model with conditional L_2 TimeCycle loss, baseline model with unconditional L_1 TimeCycle loss, baseline model with unconditional L_2 TimeCycle loss, baseline model with L_1 Ping-Pong loss, baseline model with L_2 Ping-Pong loss. L_1 versions are strictly preferable to L_2 versions for all loss types. Unconditional L_1 TimeCycle loss performs best with regard to temporal consistency. Conditional L_1 TimeCycle loss and L_1 Ping-Pong loss perform best with regard to video quality.

5.4.3. Sequence Discriminator Conditioning

In paired video-to-video translation, a conditional sequence discriminator D_S could be used, which also receives the source images $s_{t:(t+N_d-1)}$ as input for a given sequence $x_{t:(t+N_D-1)}$. In the following, we compare such conditional sequence discriminators to the unconditional sequence discriminator used in our TimeCycle, as introduced in section 3.1.2. We use the setup of Section 5.3.1 with the hyperparameters shown in Table 4.1. For both discriminator types, we measure the standalone performance and the performance when combined with the TimeCycle loss. As the baseline method, we choose pix2pix [1] with a recurrent generator such that the model with both unconditional sequence discriminator and TimeCycle loss corresponds to the TimeCycleGAN-P model of Section 3.3.4. In summary, the five models we compare are:

- Baseline: baseline model (pix2pix with recurrent generator)
- D_S-C: baseline model with conditional sequence discriminator
- D_S-C + TC: baseline model with conditional sequence discriminator and TimeCycle loss
- D_S-U: baseline model with unconditional sequence discriminator
- D_S-U + TC: baseline model with unconditional sequence discriminator and TimeCycle loss

In Table 5.6 the results of the study are shown. Compared to the baseline method, all models have considerably better tLP and tOF scores. However, the reduction of tLP is strongest for the unconditional sequence discriminator, so it can be concluded that the unconditional sequence discriminator performs strictly better with regard to temporal consistency. Concerning video quality, all models are worse than the baseline, with the standalone unconditional sequence discriminator performing worst. However, when combined with the TimeCycle loss, the unconditional sequence discriminator performs best, and even achieves a lower FID score than the baseline. Among the four sequence discriminator versions, the model with unconditional sequence discriminator and TimeCycle loss performs best with regard to both video quality and temporal consistency. Therefore, we use unconditional sequence discriminators in all our TimeCycleGAN models.

	Video Quality		Temporal Consistency	
Model	FID	LPIPS	tLP	tOF
Baseline	1.132e2	**2.229e-1**	14.482e-3	3.185e3
D_S-C	1.438e2	2.266e-1	9.396e-3	2.928e3
D_S-C + TC	1.334e2	2.542e-1	9.585e-3	3.014e3
D_S-U	1.781e2	2.441e-1	**7.652e-3**	3.006e3
D_S-U + TC	**1.096e2**	2.444 e-1	7.653e-3	**2.902e3**

Table 5.6.: Comparison of unconditional and conditional sequence discriminators in paired video-to-video translation. Models from top to bottom: baseline (pix2pix with recurrent generator), baseline with conditional sequence discriminator, baseline with conditional sequence discriminator and TimeCycle loss, baseline with unconditional sequence discriminator, baseline with unconditional sequence discriminator and TimeCycle loss. The unconditional sequence discriminator leads to better temporal consistency than the conditional sequence discriminator. The best model overall is the model with unconditional sequence discriminator and TimeCycle loss.

5.4.4. Sequence-Wise Data Augmentation

We also examine the effect of our sequence-wise data augmentation techniques proposed in Section 4.1.4 by comparing the performance of our TimeCycleGAN-P model with and without the data augmentation techniques. We use the setup in Section 5.3.1 and the hyperparameters in Table 4.1 again. The four models we compare are:

- Baseline: baseline model (TimeCycleGAN-P without data augmentation)
- Crop: baseline model with sequence-wise random cropping
- Flip: baseline model with sequence-wise horizontal random flips
- Crop + Flip: baseline model with both augmentation techniques

As shown in Table 5.7, both data augmentation techniques improve all four metrics individually and combining both techniques achieves the best performance overall. When combined, the two data augmentation techniques reduce FID by 29.2 percent and tLP by 36.4 percent.

	Video Quality		Temporal Consistency	
Model	FID	LPIPS	tLP	tOF
Baseline	1.584e2	2.589e-1	12.040e-3	2.972e3
Crop	1.474e2	2.461e-1	**7.152e-3**	2.966e3
Flip	1.333e2	**2.230e-1**	8.845e-3	2.915e3
Crop + Flip	**1.096e2**	2.444e-1	7.653e-3	**2.902e3**

Table 5.7.: Ablation study for sequence-wise data augmentations. Models from top to bottom: baseline, baseline with sequence-wise random crops, baseline with sequence-wise horizontal random flips, baseline with both sequence-wise data augmentation techniques. The data augmentation techniques improve all metrics, with significant reductions in FID and tLP.

6. Discussion

6.1. Summary

In this thesis, we propose *TimeCycleGAN*, a novel approach for temporal consistency learning in generative adversarial networks.

Contributions:

- We introduce a new loss for temporal consistency, which has fewer restrictions than existing approaches, and which can be applied to arbitrary GAN models with only a few lines of code.
- Based on the novel loss, we formulate a simple, yet powerful, meta-architecture for temporally-consistent GANs.
- Using the meta-architecture, we design novel models for unconditional video generation, paired video-to-video translation, and unpaired video-to-video translation.
- The paired video-to-video translation models improve temporal consistency scores of the baseline by up to 63.5 percent. In quantitative comparisons, our strongest model is close to the task-specific state-of-the-art.
- The unpaired video-to-video translation model improves temporal consistency scores by up to 46.3 percent, and even outperforms the state-of-the-art in its domain.

Key Insights:

- We argue that recurrent generator designs are currently best suited for video generation.
- We found that unconditional sequence discriminators perform better than conditional sequence discriminators for temporal consistency learning in paired video-to-video translation, especially when combined with our TimeCycle loss.
- We show that the L_1 distance is the preferable choice for both our TimeCycle loss and the TecoGAN Ping-Pong loss.
- We empirically confirm that perceptual losses correlate highly with human perception and that they can considerably improve the video quality of paired video-to-video translation methods.
- We show that sequence-wise data augmentation techniques can improve the generalization performance of video generation models.

6.2. Limitations

Background Synthesis The unpaired results in Section 5.3.2 show that most methods, including ours, do not accurately translate the background. This, however, is caused by a mismatch in datasets and could be solved by more careful task-specific dataset collection.

Motion Learning As shown in Section 5.3.3, our meta-architecture can enforce temporal consistency between subsequent frames, but it cannot enforce any motion. Thus, we cannot generate realistic videos in settings where the motion itself is not provided.

Training Instability A common issue of all our models is unstable training. In particular, sequential discriminators often perform too well, converging to zero loss, or the training, in general, converges to a poor local minimum, such as extremely dark, or generally unrealistic images. However, these issues can mostly be avoided by careful hyperparameter tuning.

Motion Model Mode Collapse An especially common training failure case is a sort of perceptual mode collapse that results in motion models that merely reconstruct its input, similar to an autoencoder, without learning any motion whatsoever. To prevent that, we tried to further constrain the motion models with additional losses, as outlined in Section A.3, however, without much success. The TimeCycle loss is unfortunately still highly susceptible to this issue, but, surprisingly, we found that realistic motion can be learned even in cases where this training failure occurs. This might suggest that even a simple L_1 loss between subsequent generations could already improve temporal consistency considerably.

Visual Artifacts Visual artifacts can arise from the adversarial loss on motion models if the motion model architecture is too simple. Also, for some of our models, temporal blurring or swimming can be observed, which is another type of visual artifact, caused by sequence discriminator training instability. Additional optical flow losses might alleviate this issue.

Memory Consumption The TimeCycle loss is highly memory intensive. Therefore, it cannot be applied to arbitrarily long sequences. However, realistic videos of several hundred frames can be generated even by models that were only trained on three-frame sequences at a time. Thus, it might not be necessary to perform TimeCycles over longer sequences in practice.

6.3. Future Work

TimeCycleGAN with Optical Flow As shown in Section 5.3.1, vid2vid [11] still outperforms our best TimeCycleGAN model, and particularly the motion between frames is more realistic. Thus, incorporating optical flow losses into our TimeCycleGAN approach could lead to great improvements.

Architectural Improvements We believe that significantly better results could be achieved by improving the network architectures of our models. In particular, the recurrent generators and motion models in our TimeCycleGAN models are all simple feed-forward networks, so replacing those by actual recurrent neural networks might make the training significantly faster and improve temporal consistency. Furthermore, our discriminator architectures are very simple and not custom-designed for video generation, so incorporating more sophisticated discriminator designs, such as the TecoGAN [13] triplet-discriminator, might lead to significant improvements. Lastly, it could be worthwhile to incorporate some other promising new GAN techniques into our TimeCycleGAN, such as spectral normalization [37] or self-attention [38, 39, 40], as used in some recent GAN methods [41, 42].

Motion Generation While our meta-architecture can currently not generate realistic videos in settings where the motion is not explicitly given, we believe that such applications could be enabled through additional losses or restrictions that punish inactivity. We also believe that the field of unconditional video generation still has ample opportunity for substantial improvements.

Supervised Video Generation Since deep features of perceptual networks seem to correlate highly with human image perception [34], we argue that it should be possible to train good video generation models in a purely supervised fashion, without any adversarial training, as the TimeCycle loss is not GAN-specific either and can, in theory, be used in any setting. We believe such research could be highly useful because it might provide valuable insights and effective stabilization losses for GAN training, which could then further advance the GAN-based state-of-the-art.

A. Appendix

A.1. Network Design

A.1.1. Recurrent Generator Design

Backpropagation Through Time

In our recurrent generators, we generate subsequent frames based on the previous ones and perform backpropagation through time such that subsequent frames also influence the gradient that the generator receives from previous frames. Experimentally we found that the backpropagation through time improves temporal consistency significantly and that there is no significant difference in training time, as long as all generator losses are backpropagated jointly (by summing up all losses and calling `.backward()` only once in PyTorch).

Number of Input Frames

In vid2vid [11], the generator is conditioned on L previous frames and corresponding semantic segmentation maps. Specifically, $L = 2$ is used because $L = 1$ was reported to lead to training instability, whereas larger values for L lead to slower training and larger memory consumption. In our models, we did not find any performance difference between L-values, and we did not encounter any additional training instability with $L = 1$. Furthermore, we found that the conditioning on previous semantic segmentation does not lead to performance improvements either. Thus, we only condition our generator on the current conditioning input and the last generation.

Input Random Noise

As denoted in equations 3.2 and 3.3, the recurrent generator generates each sequence from a sequence of N_G random noise images $z_{1:N_G}$. We experimented with three types of noise:

- no noise with $z_t = 0$,
- random noise images drawn from a standard-normal distribution $z_t \sim \mathcal{N}(0, 1)$,
- random noise images drawn from a uniform distribution $z_t \sim \mathcal{U}(-1, 1)$.

Experimentally we found that standard-normal random noise works best, and since it is also the preferred choice from a theoretical point of view [43], we use the standard-normal random noise for all models.

A.1.2. Motion Model Design

Motion Model Architecture

As mentioned in section 4.1.1, we use the same network architecture for the motion models that we use for the GAN generator, just with less residual blocks such that the receptive field size for each generated pixel is roughly one-fifth of the image. Experimentally we found that using a larger motion model does not improve the quality of generated videos; it only increases memory usage and slows down the training significantly. Furthermore, it makes the optimization more difficult because the motion model might generate realistic images without relying on the generator, which can result in a scenario where only the discriminator and the motion model learn together in adversarial fashion while the generator gets left behind.

Separate Motion Models

In our implementation, we only use a single motion model to perform both forward-warping and backward-warping. We tried splitting this model up into two separate models, one for forward-warping and one for backward-warping, but we found no significant differences between the generation results.

A.2. Comparison to Alternative Designs

A.2.1. TimeCycle Loss and Ping-Pong Loss

As discussed in Section 2.2.3, TecoGAN [13] proposes the Ping-Pong loss to learn long-term temporal consistency by generating a sequence first forwards and then backwards and by then constraining the corresponding generations of each frame to be close to each other. This can be seen as a special case of the TimeCycle loss, where the motion model is the generator itself, and where the cycle reconstructions are omitted since the backward predictions are now already reconstructions themselves. Because the TimeCycle and Ping-Pong losses are closely related, the two losses perform almost equally well, as we showed in Section 5.4.2. However, contrary to [13], we found that it is beneficial to use the L_1 distance in all losses, including the Ping-Pong loss, as the L_2 distance can result in visual artifacts.

A.2.2. Other Types of Sequential Generators

3D Generators

Another way of establishing correlation between frames in a sequence is to generate the entire sequence at once, as several methods have proposed by using 3D-convolutional generators [44, 45, 46, 47, 48]. While these methods show promising results, they are highly memory-intensive and scale poorly to higher resolutions and longer sequences. Therefore, we argue that such sequence-wise generators are currently not yet well suited for truly realistic video generation.

Temporal Consistency Through Mode Collapse

Some methods claim to learn temporal consistency with a simple per-frame generator where each frame is generated independently. However, we argue that the perceived temporal consistency of such methods is caused by an extreme mode collapse. While impressive results can be achieved in narrow domains even with mode collapse, such methods will not be able to handle dynamic style changes and will likely fail at generalizing to unseen data.

A.3. TimeCycle Extensions

A.3.1. TimeCycle Warp Losses

Inspired by the optical flow losses in vid2vid [11], we experimented with two additional warp losses for our TimeCycle:

Real Warp Loss The first loss,

$$\mathcal{L}_{TC_{W,real}}(M) = \mathbb{E}_{x_{1:T}}[\frac{1}{T-1}\sum_{t=1}^{T-1} ||M(x_t, c_t^f) - x_{t+1}||_1 + ||M(x_{t+1}, c_{t+1}^b) - x_t||_1], \tag{A.1}$$

is a loss for the motion model only, which trains it to correctly warp real frames. This should further constrain the motion model to ensure it learns proper temporal warping.

Fake Warp Loss The second loss,

$$\mathcal{L}_{TC_{W,fake}}(G|M) = \mathbb{E}_{x_{1:T}}[\frac{1}{T-1}\sum_{t=1}^{T-1} ||M(\tilde{x}_t, c_t^f) - \tilde{x}_{t+1}||_1 + ||M(\tilde{x}_{t+1}, c_{t+1}^b) - \tilde{x}_t||_1], \tag{A.2}$$

is similar to the first, except that we calculate it on the generated frames, and detach the result of the motion model. Thus, the motion model does not learn from this loss, but only the generator, which is thereby explicitly constrained to generate images such that the difference between subsequent frames is minimal, which should further improve short-term temporal consistency.

Similarity to vid2vid Losses If we assume that our motion model learns a motion function that is equivalent to the optical flow prediction combined with optical flow warping, then these two losses are similar to the flow loss and warp loss in vid2vid, as introduced in equations 2.15 and 2.16 respectively. The only difference is that we omit the comparison to the reference flow in equation A.1, since we never predict any optical flow explicitly.

Results In practice, we found that the first loss on real images in equation A.1 leads to lower video quality with similar temporal consistency. The second loss term in equation A.2 seemed to improve temporal consistency, however, so it might be worthwhile to conduct further experiments with these losses. For now, we did not include them in our TimeCycleGAN models, as the results were ambiguous.

A.3.2. TimeCycle + Sequence Discriminator

In addition to using the sequence discriminator to discriminate the sequence generated by the generator, we also tried to apply it to the sequence that is formed by the intermediate TimeCycle predictions used for the prediction similarity loss in section 3.2.3. We expected this to improve temporal consistency further but observed no significant improvements.

A.3.3. TimeCycle on Real Images

To further constrain the motion models, we experimented with an additional loss, where the motion models perform a TimeCycle on real sequences, and a TimeCycle loss is computed, similar to equation 3.25. We expected that this would stabilize the training by enforcing that the motion model learns valid motions. However, we observed no significant improvements, but a slight video quality deterioration instead.

List of Figures

List of Tables

Glossary

CycleGAN Method for unpaired image-to-image translation. 6

Discriminator Discriminator, part of a generative adversarial network. 3

Deep Convolutional GAN Method for unconditional image generation. 4

Fréchet Inception Distance Metric for evaluating the image quality of GAN generations. 42

Generator Generator, part of a generative adversarial network. 3

Generative Adversarial Network Deep learning approach for video generation: two neural networks are trained in adversarial fashion. 3

Image-to-Image Translation The task of generating images from other images. 4

Learned Perceptual Image Patch Similarity Metric for evaluating the image quality of GAN generations in paired settings. 42

TimeCycle Loss Our proposed general-purpose loss for temporal consistency. 24

PatchGAN Discriminator of pix2pix processing images patch-wise. 4

pix2pix Method for paired image-to-image translation. 4

pix2pixHD Method for paired high-resolution image-to-image translation. 5

RecycleGAN Method for unpaired video-to-video translation. 9

Temporal Cycle-Consistency Ability of a model to reconstruct its input through a temporal backward-forward generation cycle. 22

Temporally Cycle-Consistent Adjective refering to high temporal cycle-consistency. 12

TimeCycleGAN Our proposed meta-architecture for GAN-based video generation. 24

TimeCycleGAN-P Our TimeCycleGAN model for paired video-to-video translation. 30

TimeCycleGAN-P++ Our improved TimeCycleGAN model for paired video-to-video translation with additional feature matching and perceptual losses. 30

TimeCycleGAN-U Our TimeCycleGAN model for unconditional video generation. 27

TimeCycleGAN-UP Our TimeCycleGAN model for unpaired video-to-video translation. 27

Temporally Coherent GAN Method for paired and unpaired video-to-video translation. 9

vid2vid Method for paired high-resolution video-to-video translation. 7

Video-to-Video Translation The task of generating videos from other videos. 7

Acronyms

D Discriminator. 3

DCGAN Deep Convolutional GAN. 4

D_S Sequence Discriminator. 8, 15

FID Fréchet Inception Distance. 42

G Generator. 3

GAN Generative Adversarial Network. 3

$\hat{G}$ Combined generator in vid2vid. 7

λ Loss weight hyperparameter. 5

$\mathcal{L}_C$ Cycle-consistency loss for unpaired generation settings. 6

$\mathcal{L}_{cGAN}$ Conditional adversarial loss in conditional generative adversarial networks. 4

$\mathcal{L}_F$ Flow loss in vid2vid. 7

$\mathcal{L}_{FM}$ Auxiliary feature matching loss used in paired generation settings. 5, 30, 31

$\mathcal{L}_{GAN}$ Adversarial loss in generative adversarial networks. 3

$\mathcal{L}_{GAN_{TC}}$ Adversarial motion model loss, part of the TimeCycle Loss. 23

$\mathcal{L}_{idt}$ Identity mapping loss used in some unpaired generation applications. 6

$\mathcal{L}_{L1}$ Auxiliary L1 loss used in paired generation settings. 4

$\mathcal{L}_{L2}$ Auxiliary L2 loss used in TecoGAN. 11

$\mathcal{L}_P$ Auxiliary perceptual loss used in paired generation settings. 5, 31

LPIPS Learned Perceptual Image Patch Similarity. 42

$\mathcal{L}_{PP}$ Ping-Pong loss of TecoGAN. 10

$\mathcal{L}_{RC}$ Recycle loss - spatio-temporal cycle-consistency loss used in RecycleGAN. 9

$\mathcal{L}_{RCpred}$ Recurrent loss used for predictor training in RecycleGAN. 9

$\mathcal{L}_{sGAN}$ Sequential adversarial loss in our sequential generative adversarial networks. 18

$\mathcal{L}_{TC}$ TimeCycle Loss. 24

$\mathcal{L}_{TC_{pred}}$ Prediction similarity loss, part of the TimeCycle loss. 23

$\mathcal{L}_{TC_{rec}}$ Temporal cycle-consistency loss, part of the TimeCycle Loss. 23

$\mathcal{L}_W$ Optical flow warp loss in vid2vid and TecoGAN. 8

M Motion model, part of TimeCycleGAN. Also, mask prediction network in vid2vid. 7, 21

N_D Number of input frames of sequence discriminators. 8

N_{DS} Number of spatial scales in spatial multi-scale discrimination. 5

N_{DT} Number of temporal scales in temporal multi-scale discrimination. 8

N_G Number of previous generations a recurrent generator is conditioned on. 7

N_{TC} Number of previous frames over which temporal cycle-consistency learning is performed. 12

ϕ Feature extraction network in temporal cycle-consistency learning. 11

T Sequence length in sequential GAN training. 7

$\mathcal{T}$ Tracking network in temporal cycle-consistency learning. 11

TCGAN-P++ TimeCycleGAN-P++. 30

TCGAN-P TimeCycleGAN-P. 30

TCGAN-U TimeCycleGAN-U. 27

TCGAN-UP TimeCycleGAN-UP. 27

TecoGAN Temporally Coherent GAN. 9

tLP Metric for evaluating temporal consistency of GANs in paired settings, based on LPIPS. 43

tOF Metric for evaluating temporal consistency of GANs in paired settings, based on optical flow. 43

W Optical flow prediction network in vid2vid. 7

warp Optical flow warping operation. 7

x Real target images. 3

$\tilde{x}$ Fake (generated) target images. 3

y Real source/conditioning images. 3

$\tilde{y}$ Fake (generated) source/conditioning images. 6

z Random noise used as generator input. 3

Bibliography

[1] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134.

[2] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired image-to-image translation using cycle-consistent adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2223–2232.

[3] 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*. 2014, pp. 2672–2680.

[4] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[5] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[6] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[7] V. Nair and G. E. Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[8] A. L. Maas, A. Y. Hannun, and A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *Proc. icml*. Vol. 30. 1. 2013, p. 3.

[9] B. Xu, N. Wang, T. Chen, and M. Li. "Empirical evaluation of rectified activations in convolutional network". In: *arXiv preprint arXiv:1505.00853* (2015).

[10] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. "High-resolution image synthesis and semantic manipulation with conditional gans". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8798–8807.

[11] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. "Video-to-Video Synthesis". In: *Advances in Neural Information Processing Systems (NIPS)*. 2018.

[12] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. "Recycle-gan: Unsupervised video retargeting". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 119–135.

[13] M. Chu, Y. Xie, L. Leal-Taixé, and N. Thuerey. "Temporally coherent gans for video super-resolution (tecogan)". In: *arXiv preprint arXiv:1811.09393* (2018).

[14] G. E. Hinton and R. R. Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[15] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[16] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[17] J. Long, E. Shelhamer, and T. Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[18] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[19] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. "Improved techniques for training gans". In: *Advances in neural information processing systems*. 2016, pp. 2234–2242.

[20] A. Dosovitskiy and T. Brox. "Generating images with perceptual similarity metrics based on deep networks". In: *Advances in neural information processing systems*. 2016, pp. 658–666.

[21] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[22] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. "Flownet 2.0: Evolution of optical flow estimation with deep networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2462–2470.

[23] J. Johnson, A. Alahi, and L. Fei-Fei. "Perceptual losses for real-time style transfer and super-resolution". In: *European conference on computer vision*. Springer. 2016, pp. 694–711.

[24] X. Wang, A. Jabri, and A. A. Efros. "Learning correspondence from the cycle-consistency of time". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2566–2576.

[25] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. "Temporal cycle-consistency learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 1801–1810.

[26] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley. "Least squares generative adversarial networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2794–2802.

[27] D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[28] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. "The cityscapes dataset for semantic urban scene understanding". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 3213–3223.

[29] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. "Rethinking atrous convolution for semantic image segmentation". In: *arXiv preprint arXiv:1706.05587* (2017).

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* 115.3 (2015), pp. 211–252.

[31] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, pp. 1–16.

[32] 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: *Advances in neural information processing systems*. 2017, pp. 6626–6637.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[34] 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*. 2018, pp. 586–595.

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[36] B. D. Lucas, T. Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: (1981).

[37] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. "Spectral normalization for generative adversarial networks". In: *arXiv preprint arXiv:1802.05957* (2018).

[38] J. Cheng, L. Dong, and M. Lapata. "Long short-term memory-networks for machine reading". In: *arXiv preprint arXiv:1601.06733* (2016).

[39] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. "A decomposable attention model for natural language inference". In: *arXiv preprint arXiv:1606.01933* (2016).

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[41] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. "Self-attention generative adversarial networks". In: *arXiv preprint arXiv:1805.08318* (2018).

[42] A. Brock, J. Donahue, and K. Simonyan. "Large scale gan training for high fidelity natural image synthesis". In: *arXiv preprint arXiv:1809.11096* (2018).

[43] T. White. “Sampling generative networks”. In: *arXiv preprint arXiv:1609.04468* (2016).

[44] C. Vondrick, H. Pirsiavash, and A. Torralba. “Generating videos with scene dynamics”. In: *Advances in neural information processing systems*. 2016, pp. 613–621.

[45] D. Bashkirova, B. Usman, and K. Saenko. “Unsupervised video-to-video translation”. In: *arXiv preprint arXiv:1806.03698* (2018).

[46] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. Metaxas. “Learning to forecast and refine residual motion for image-to-video generation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 387–403.

[47] W. Wang, Q. Huang, S. You, C. Yang, and U. Neumann. “Shape inpainting using 3d generative adversarial network and recurrent convolutional networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2298–2306.

[48] W. Xiong, W. Luo, L. Ma, W. Liu, and J. Luo. “Learning to generate time-lapse videos using multi-stage dynamic generative adversarial networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 2364–2373.