João Manuel Pedro Agria

deepSTAII: Style Transfer for Artificial Illustrations

Master's Dissertation in Electrical and Computer Engineering, supervised by
Prof. Dr. Paulo Jorge Carvalho Menezes (ISR) and presented to the
Faculty of Science and Technology of the University of Coimbra

February of 2020

· U C ·

Universidade de Coimbra

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

deepSTAIl: Style Transfer for Artificial Illustrations

João Manuel Pedro Agria

February of 2020

deepSTAIl: Style Transfer for Artificial Illustrations

Supervisor:

Prof. Dr. Paulo Jorge Carvalho Menezes (ISR)

Jury:

Prof. Dr. Jorge Manuel Miranda Dias

Prof. Dr. Jorge Manuel Moreira de Campos Pereira Batista

Prof. Dr. Paulo Jorge Carvalho Menezes

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

February of 2020

Agradecimentos

Em primeiro lugar, devo agradecer ao meu orientador, o Professor Paulo Jorge Carvalho Menezes, por me ter concedido esta oportunidade de aprender e trabalhar num tema tão interessante e complexo como o de transferência de estilo com redes neuronais; pela dedicação, ajuda e partilha de conhecimento que proporcionou ao longo do último ano.

À minha família, que sempre me apoiou nos tempos difíceis. Este trabalho não teria sido possível sem o vosso apoio incondicional.

Finalmente, a todos os meus amigos e colegas da Universidade de Coimbra e do curso de Engenharia Electrotécnica e de Computadores, que tornaram os últimos anos inesquecíveis. Devo ainda uma especial menção para o grupo do IS3Lab e do LRM, que fizeram com que os últimos meses passassem a voar.

Coimbra, Fevereiro de 2020

Resumo

Transferência de estilo com redes neuronais é a versão mais recente do ramo de interpretações artísticas baseadas em imagens. Historicamente, algoritmos de estilização para interpretações não-realistas foram desenvolvidos especificamente em torno de certas primitivas. Por exemplo, uma interpretação baseada em pinceladas colocava pinceladas virtuais numa imagem, mas era desenvolvida cuidadosamente para um estilo particular de pincelada e revelava-se incapaz de simular um estilo arbitrário. Esta limitação inerente de flexibilidade, estilo e diversidade que alguns algoritmos de interpretações artísticas baseadas em imagens tinham era equilibrada pela sua capacidade de representar fielmente os estilos artísticos para os quais eram criados. A procura por novos algoritmos que respondessem a estas limitações resulta no aparecimento da transferência de estilo com redes neuronais. A introdução de redes neuronais convolucionais causou uma mudança profunda nesta velha área de investigação, e atraiu a atenção de círculos académicos e industriais.

Esta dissertação tem como objectivo ultrapassar as limitações computacionais do algoritmo clássico de transferência de estilo treinando uma rede geradora para realizar a mesma tarefa centenas de vezes mais depressa. A continuação lógica de uma transferência de estilo mais rápida, que é a transferência de estilo em vídeo, é um tópico que será explorado nesta dissertação devido às suas variadas aplicações em cenários de realidade aumentada e de realidade virtual, e na indústria de animação. Para solucionar o problema de processamento de vídeo com redes neuronais, duas alternativas são consideradas: utilizar métodos do ramo de visão por computador para guiar o treino da rede, ou alterar a arquitectura da rede para aferir informação temporal e espacial ao mesmo tempo.

Palavras-Chave: Redes neuronais convolucionais, VGG, Transferência de estilo com redes neuronais, Rede Convolucional LSTM, Processamento de Vídeo

Abstract

Neural style transfer is the most recent facet of image-based artistic rendering. Historically, stylization algorithms for non-photorealistic rendering were designed specifically around certain primitives. For example, stroke based rendering placed virtual strokes on an image, but was carefully designed for only one particular style of stroke and not capable of simulating an arbitrary style. This inherent limitation on flexibility, style and diversity some IB-AR algorithms had was balanced by their capability of faithfully depicting those certain prescribed styles. The demand for novel algorithms to address these limitations gives birth to the field of NST. The introduction of convolutional neural networks caused a paradigm shift in this long standing area of research, and attracted the attention of both academic and industrial circles.

This dissertation has the goal of enabling classical neural style transfer to overcome its computational limitations by training a generative network to perform the same task hundreds of times faster. The logical continuation of faster neural style transfer, video style transfer, is a topic that will be explored due to its many possible applications in augmented reality and virtual reality scenarios, and in the animation industry. To solve the task of video processing with neural networks, two alternatives are considered: using computer vision methods to guide a network's training, or changing a networks architecture to take into account spatial and temporal information at the same time.

Keywords: Convolutional Neural Networks, VGG, Neural Style Transfer, Convolutional LSTM network, Video Processing

Contents

List of Acronyms

NPR Non-photorealistic rendering

IB-AR Image-based artistic rendering

NST Neural Style Transfer

VGG Visual Geometry Group

CNN Convolutional neural network

LSTM Long Short Term Memory

ITN Image Transformation Network

VTN Video Transformation Network

List of Figures

List of Tables

1 Introduction

Since immemorial times, artistic expression has been a pivotal aspect of mankind's cultural life. Paintings, in particular, are the expression of ideas and emotions by way of a two dimensional visual language with certain aesthetic qualities. Shapes, lines, colours, tones and textures are some of the constituent parts of this language, and each are applied to a canvas in a precise manner to produce an harmonious composition of volume, space, movement and light. Paintings have always had a symbiotic relationship with society's paradigm shifts, mirroring changes brought on by technological advances or by emerging schools of thought and new cultural sensibilities. Therefore, it can be logically concluded that artificial intelligent systems, recent targets of much attention and publicity both in academic and in industrial circles, are seen as somewhat useful tools for creative output tasks.

While the ultimate goal of artificial intelligent systems has been set as their capacity of reaching, or even exceeding, human performance in certain tasks, the question remains whether an ill defined problem such as creative production can be mimicked. Art and creativity are fundamental features inherent in human intelligence, and fundamental signatures of humankind. It is a way through which we express our sentience; the capacity to feel, perceive, or experience subjectively. Can we create something that is in itself creative? With recent advances in methodology and technology, we are now able to build systems that can paint original artwork in a given style.

This thesis is concerned with the use of Convolutional Neural Networks to generate images resembling the overall appearance of a given style image while preserving its original semantic content. The remainder of this chapter will delve into the relevancy of this subject, some of its key concepts, and summarise this thesis main objectives and what has been accomplished.

1.1 Generative Modelling

In deep learning, one of the objectives is to find models that represent probability distributions over data. Discriminative models, whose task is to map (highly) dimensional inputs to class labels, have been the focus of deep learning until recently. Discriminative modelling is easier to monitor, and performance can be measured against high-profile classification methods to determine the best methodology. Generative models received less attention, comparatively to discriminative models, partly due the inherent difficulty in evaluating its output.

Generative modelling is finding more practical uses in academia and industry, such as in the field of reinforcement learning. For artificial intelligence to be comparable to a human's, then it needs to transcend the usual discriminative models. According to David Foster [5], current neuroscientific theory suggests that our perception of reality is not a highly complex discriminative model operating on our sensory input to produce predictions of what we are experiencing, but is instead a generative model that is trained from birth to produce simulations of our surroundings that accurately match the future. Some theories even suggest that the output from this generative model is what we directly perceive as reality. A deep understanding of how we can build machines to acquire this ability will be central to our continued understanding of the workings of the brain and general artificial intelligence.

A generative model is defined as a model which describes how a dataset is generated in terms of a probabilistic model. To generate new data, one has to sample from this model. Generative modelling differs from its counterpart, discriminative modelling, by not requiring labelled data during training. When performing discriminative modelling, each observation in the training data has a label. Discriminative modeling and supervised learning are, therefore, synonymous. Generative modeling is (usually) performed with an unlabelled dataset, though in some cases a labelled dataset can be applied. In mathematical terms, discriminative modelling estimates $p(y|x)$: the probability of a label y given observation x; generative modelling estimates $p(x)$: the probability of observing observation x.

A generic framework for generative modelling begins with X, a dataset of observations. Assuming that the observations have been generated according to some unknown distribution p_{data}, then a generative model p_{model} tries to mimic p_{data}. Then, by sampling p_{model}, it is possible to generate observations that appear to have been drawn from p_{data}. These observations need to be different from the observations in X; the model shouldn't simply reproduce things it has already seen.

1.2 Image Based Artistic Rendering

Image-based artistic rendering is a field of non-photo realistic rendering (NPR) focusing on techniques for transforming 2D inputs (images and video) into artistically stylized renderings. While simulation of artistic media with high fidelity has been historically limited because it relies on predefined image pairs for training and because it informs only low-level image features for texture transfers, recent advancements in deep learning showed to alleviate these limitations by matching content and style statistics via activations of neural network layers, thus making a generalised style transfer practicable.

In more concise terms, deep learning introduced a major paradigm shift in artistic rendition of images because the layers of deep convolutional neural networks pre-trained to accurately classify high-level image contents across generalised data sets have implicitly learned to encode the perceptual and semantic information - style and content statistics - that can be used to perform a neural style transfer between arbitrary images. The separation of content from style is considered to be a key factor in neural style transfer, since it allows us to distinguish between the mechanisms used for capturing the essence of an image - its semantic information - and the design aspects that drive the aesthetic appeal to stimulate human senses - the style.

1.3 Texture Synthesis

To comprehend style transfer, style needs to be mathematically defined. Style can be understood as a texture, and, therefore, texture synthesis is an integral part of style transfer. This section will provide an historical overview of methods for texture synthesis. Before all else, it is necessary to introduce a precise definition of texture [35].

In the field of computer graphics, a texture is any common image. In other contexts, such as computer vision and image processing fields, textures are usually referred to as visual or tactile surfaces composed of repeating patterns, such as a fabric. This last definition of texture is more restricted than the notion of texture in graphics. However, since a majority of natural surfaces consist of repeating elements, this narrower definition of texture is still powerful enough to describe many surface properties.

A stationary image is one where, given a moving window of a certain size, the observable portions always appear similar to each other. Textures differ from general images in their stationarity, as seen in fig. 1.1. Examples of non-stationary textures include textures with

large-scale irregular structures, or ones that exhibit spatial variance in various attributes, such as color, local orientation, and local scale.

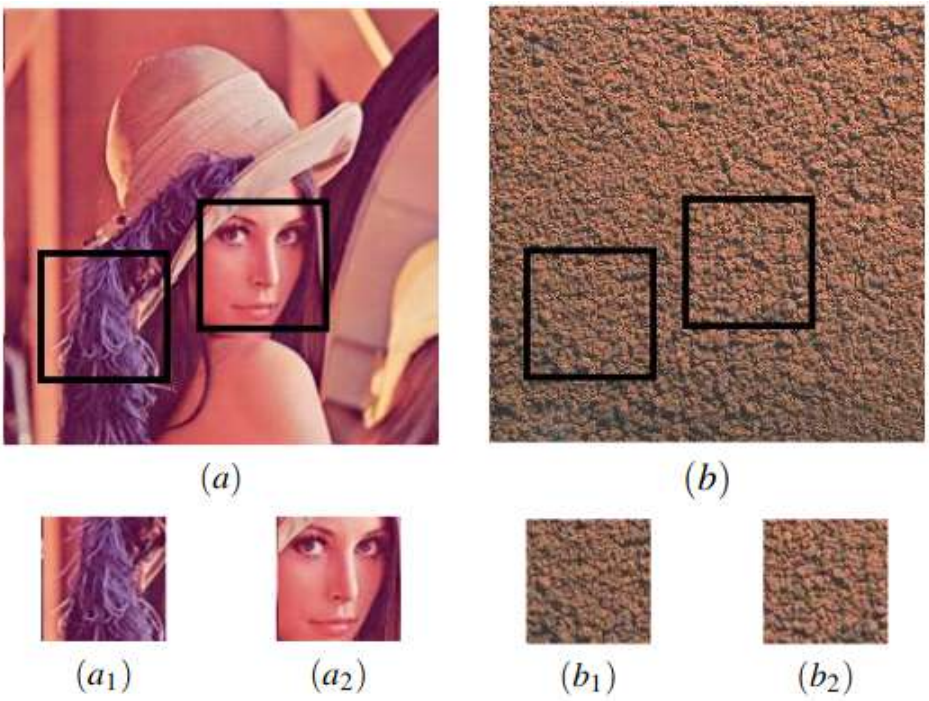

Figure 1.1: (a) is a general image while (b) is a texture. A movable window with two different positions are drawn as black squares in (a) and (b), with the corresponding contents shown below. Different regions of a texture are always perceived to be similar (b1,b2), which is not the case for a general image (a1,a2). This characteristic is called stationarity.

The quest for a mathematical formulation for texture description has been an important subject in the fields of computer vision and computer graphics in the past 50 years. In his seminal paper from 1962 [19], Julesz initiated researches on texture by asking the following fundamental question:

What features and statistics are characteristic of a texture pattern, so that texture pairs that share the same features and statistics cannot be told apart by pre-attentive human visual perception?

If texture images are, loosely speaking, spatially homogeneous and consist of repeated elements, then Julesz pioneered the statistical characterisation of textures by hypothesising that histograms of image pixels could be used to partition textures into classes that are pre-attentively indistinguishable to a human observer. The Julesz conjecture was proven to be wrong, yet the basic idea to describe a texture by a set of spatial summary statistics forms the basis of parametric texture modelling. The notion of images with equal pixel histograms is still applied fruitfully today.

Simoncelli and Freeman [30] synthesised textures by matching histograms of filter outputs. The algorithm modifies an uniform white noise input image to make it look like the input texture by making use of an invertible image representation known as an image pyramid. The steerable pyramid transform decomposes the image into several spatial frequency bands and further divides each frequency band into a set of orientation bands. Their work exploits filter responses to analyse textures, instead of direct pixel-based measurements.

Following that, Portilla and Simoncelli [25] introduce a texture model based on multi-scale oriented filter responses and use gradient descent to improve synthesised results. They calculate a statistical description of an image on the responses of a linear filter bank: the steerable pyramid. Neural Style Transfer is very similar to the model proposed by Portilla and Simoncelli: to generate a texture from a given source image, features of different sizes are extracted, spatial summary statistics are calculated on these feature responses in order to obtain a stationary description of the source image, and a new image with the same stationary description is generated by performing gradient descent on an uniform white noise image. In Neural Style Transfer, instead of a linear filter bank and a set of carefully chosen summary statistics, the feature space is provided by a pre-trained deep neural network and the chosen spatial summary statistics are the correlations between feature responses in each layer of the network.

All the approaches described up until now rely on an explicitly defined parametric texture model in order to synthesise a texture. The model usually consists of a set of statistical measurements that are taken over the spatial extent of the image, or over the spatial extent of filter responses to the image. In a parametric model, a texture is uniquely defined by the outcome of those measurements and every image that produces the same outcome should be perceived as the same texture. Therefore, new samples of a texture can be generated by finding an image that produces the same measurement outcomes as the original texture [7].

Another approach to a texture generating process is to generate a new texture by resampling either pixels or whole patches of the original texture. These non-parametric resampling techniques and their numerous extensions and improvements are capable of producing high quality natural textures very efficiently. However, they do not define an actual model for natural textures, but rather give a mechanistic procedure for how one can randomise a source texture without changing its perceptual properties.

1.4 State of the Art

Neural Style Transfer was defined for the first time in the seminal work of Gatys et. al [8]. Their algorithm for style transfer will be described in detail in chapter 3. This section presents an overview of current advances in Neural Style Transfer.

NST revolves around image reconstruction algorithms building from abstract representations produced by a CNN. Mahendran and Vedaldi [22] iteratively optimise an image with gradient descent, starting from random noise, until it reaches the desired CNN representation. Similarly, NST is a computationally slow process. Training a feed forward network in advance removes this computational burden; the image reconstruction process is sped up since it can now be done with a single network forward pass. NST can, therefore, be sectioned into two major categories: the optimisation based approach and the neural network training based approach.

The model training approach was first proposed by Ulyanov et al. [33] and by Johnson et al. [18]. Even though both methods share the same similar idea, they differ in the network architecture. Johnson et al.'s design roughly follows the network proposed by Radford et al. [26], but with residual blocks as well as fractionally strided convolutions. Ulyanov et al. use a multi-scale architecture as the generator network, and find that the network converges faster and that stylisation quality greatly improves when batch normalisation layers are replaced by instance normalisation layers [34]. In both cases, separate generative networks have to be trained for each particular style image.

To overcome this inflexibility, Dumoulin et al. [4] introduce a new type of normalisation layer: the conditional instance normalisation layer, based on the instance normalisation layer [34]. It is proposed that, when a set of style images share similar paint strokes and only differ in their colour palettes, maintaining the same convolutional parameters and tuning the parameters in the normalisation layers is all that's necessary to model those styles. Therefore, one model can be trained for a set of style images by tying a small number of parameters in each normalisation layer to each one.

Expanding from fast style transfer, the design of a video style transfer algorithm needs to consider the smooth transition between adjacent video frames. The first video style transfer algorithm is proposed by Ruder et al. [28]. It is an image optimisation method that makes use of optical flow to eliminate temporal artefacts and produce smooth stylised videos. Since it takes several minutes to process each frame, training a network to stylise a sequence of frames will improve efficiency, especially for long videos with several hundreds of frames.

There have been, up until now, two ways of solving the problem of video consistency: the first one doesn't use optical flow during test time to guide the stylisation along motion trajectories; the second one does.

Introducing the first known approach to fast video style transfer, Huang et al. [14] train a network by passing two consecutive frames from a scene dataset. The network will output both frames stylised and enforce pixel-wise consistency between them both. Another feed-forward method was proposed by Gupta et al. [12]: finetuning a pre-trained feed-forward network with a video dataset and temporal consistency constraint. Gupta et al. train a network to adapt a previously processed frame to conform to the overall appearance of its predecessor. What both these approaches have in common is their independence from optical flow during test time; they belong to the first category.

In the second category, and in the vein of finetuning pre-trained stylisation networks, Chen et al. [3] propose a flow subnetwork to produce feature flow and incorporate optical flow information in feature space. This feature flow will be used to wrap feature activations from the pre-trained network. Meanwhile, Ruder et al. [29] adopt the conventional temporal consistency constraint guided by optical flow in pixel space - instead of feature space. Their approach is closer to Huang et al.'s [14] than to Gupta et al.'s [12], since they train a network instead of finetuning a pre-trained one. However, instead of using a pair of consecutive frames, Ruder et al. feed the network with an unprocessed frame and the previously stylised frame wraped with optical flow.

All these four methods were built on a network architecture inspired by Johnson et al. [18], with small adaptations. For example, Huang et al. [14] propose a residual network with less filters in each convolutional layer than [18]; Ruder et al. [29] experiment with a network composed by dilated convolutional layers; and Gupta et al. [12] use a different type of training algorithm - backpropagation through time.

Their results vary in several key aspects. In the field of optical flow independent methods, Huang et al. [14] produce temporally coherent results by sacrificing stylisation appearance. The resulting video frames aren't a faithfull representation of the source style image, lacking most of the perceptual details that characterise it. On the other hand, Gupta et al. [12] achieve images perceptually similar to those produced by an image stylisation network, but can't maintain temporal consistency in image regions with little semantic structure (the floor, the sky or a background wall). Optical flow dependent methods achieve better result, both in temporal consistency and in faithfulness to the source style image. However, they incur the cost of optical flow estimation during test time.

1.5 Objectives and Contributions

The overall objective of this dissertation is to apply the concept of neural style transfer to frame sequences in a way that is independent of optical flow estimation during test time.To achieve this aim, an exploratory work on a new model architecture based on LSTM convolutional layers is conducted, and its results compared with two alternative methods. The first alternative method is an optical flow dependent implementation inspired on the work of Ruder et al. [29]. The second is an image transformation network inspired by the architecture and training procedure of Johnson et al. [18] and Ulyanov et al. [34].

The main contributions of this thesis are two generative network architectures to conduct fast style transfer on images and videos, and their proposed training methods.

1.6 Thesis Structure

This document is organised in seven chapters. The second chapter introduces several concepts related to deep neural networks, such as types of layers and types of optimiser. The third chapter describes in some detail the image optimisation algorithm for NST. In chapter 4, a training procedure and an architecture for a feed-forward network are presented. In chapter 5, video style transfer is introduced, along with its key concepts and training procedure. Two methods of video style transfer are described: one is optical flow dependent during test time, and the other is not. Chapter 6 will present and summarise the results obtained from the implementations described in chapters 3, 4 and 5. Chapter 7 addresses the main conclusions from this thesis and future work.

2 Deep Neural Networks

A model that maps a set of random variables to samples from a data distribution is a very complex function, and the state-of-the-art model for learning arbitrary relations is a deep neural network. Deep feedforward networks are the quintessential deep learning models. A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. Training a feedforward neural network is the search for suitable parameters θ such that $f(x; \theta)$ has desirable properties. The large number of parameters confers a versatility that allows networks to mimic highly complex functions. For the task of approximating a generative process, a neural network's output should be samples from the data distribution.

The models described in this thesis are built around neural networks, and neural networks are composed by non-linear and linear functions. Non-linear functions are normally applied element-wise and differentiable. Linear functions have adjustable parameters, or weights, that change the output of the neural network. These functions, modular units composing the network, are also called layers. Intermediate outputs of each layer are called features or activations. The differentiability of the entire network is one of its most important characteristics and a requirement for learning, since it requires computing the gradients of complicated functions. In deep learning, the execution of the function is called forward propagation and computing the gradient (first derivatives) with respect to its input and weights is called backpropagation, as gradients are propagated from the end of the network to the input layer.

2.1 Gradient-Based Optimisation

Most deep learning algorithms involve optimisation of some sort. Optimisation refers to the task of either minimising or maximising some function $f(x)$ by altering x. Optimisation problems are usually phrased in terms of minimisation, and the function we want to minimise

is called the objective function or loss function.

Gradient descent, a first-order optimisation technique, finds the local minima of continuous functions

$$\arg\min_{\theta} L(X, \theta) \tag{2.1}$$

where X represents the data, θ are the weights of the network, and L is a loss function. Gradient descent is an iterative method where, in every iteration, the weights θ are perturbed slightly to improve the loss function. The update equation is

$$\theta_{next} = \theta - \alpha \nabla_{\theta} L(x, \theta) \tag{2.2}$$

where $\nabla_{\theta} L(X, \theta)$ is the gradient, denoting the multivariate first-order derivatives of $L(X, \theta)$ with respect to θ, and α is the learning rate.

Since gradient descent is dependent on the amount of data, stochastic gradient descent is used in practice. The idea is to estimate the gradient for a small sample of randomly chosen training inputs. By averaging over this small sample, a good estimate of the true gradient is reached. For a batch of N inputs, the update equation is:

$$\theta_{next} = \theta - \frac{\alpha}{N} \sum_{i=1}^{N} \nabla_{\theta} L_i(X_i, \theta) \tag{2.3}$$

While the number of iterations required to find the best solution may be higher than solving 2.1, stochastic optimisation will typically find decent solutions quickly, as the weights can be updated before even observing the entire dataset. In practice, a small number of samples, called a mini-batch, is used per iteration.

2.2 Adam Optimisation

Adam, or adaptive moment estimation, is a method for efficient stochastic optimisation that only requires first-order gradients with little memory requirement. This method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. To estimate the moments, Adam utilises exponential moving

averages, computed on the gradient evaluated on a current mini-batch:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \tag{2.4}$$

Where m and v are moving averages, g is gradient on current mini-batch, and β_1 and β_2 are hyper-parameters of the algorithm, with default values of 0.9 and 0.999 respectively. The vectors of moving averages are initialised with zeros at the first iteration. Since the moving averages are estimates of the 1^{st} moment (the mean) and the 2^{nd} raw moment (the uncentered variance) of the gradient, then the following expression should hold true:

$$\begin{aligned} E[m_t] &= E[g_t] \\ E[v_t] &= E[g_t^2] \end{aligned} \tag{2.5}$$

However, as these moving averages are initialised as zeros, moment estimates are biased towards zero, especially during the initial timesteps, and especially when the decay rates are small (i.e. β_1 and β_2 are close to 1). This initialisation bias can be easily counteracted, resulting in bias-corrected estimates:

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{(1-(\beta_1)^t)} \\ \hat{v}_t &= \frac{v_t}{(1-(\beta_2)^t)} \end{aligned} \tag{2.6}$$

The moving averages are then used to scale the learning rate individually for each parameter:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.7}$$

The method is straightforward to implement, is computationally efficient,has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients.

2.3 Convolutional Neural Networks

Convolutional neural networks, or CNNs, are a specialised kind of neural network for processing data that has a known grid-like topology, such as images. The key breakthrough in the field of deep learning came in 2012 when a deep convolutional neural network (CNN) outperformed the state-of-the-art in object recognition in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [20]. Convolutional networks are perhaps the greatest success story of biologically inspired artificial intelligence. Though convolutional networks have been guided by many other fields, some of the key design principles of neural networks were drawn from neuroscience.

2.3.1 Convolutional Layer

The convolution operation is typically denoted with an asterisk:

$$s(t) = (x * w)(t) \tag{2.8}$$

where x, the first argument to the convolution, is often referred to as the input, and w, the second argument, referred to as the kernel.

However, a convolution, in the context of neural networks, is not the same as the standard discrete convolution operation as it is usually understood in the mathematical literature. Convolution is understood as a operation consisting in many applications of convolution in parallel [10]. This is because convolution with a single kernel can extract only one kind of feature, albeit at many spatial locations. Usually, each layer of a CNN will extract many kinds of features, at many locations. Additionally, the input is usually not just a grid of real values. Rather, it is a grid of vector-valued observations. For example, a colour image has a red, green and blue intensity at each pixel. In a multilayer convolutional network, the input to a given layer is the output of the preceding layer, which usually has the output of many different convolutions at each position. When working with images, the input and output of the convolution are 3-D tensors, with one index into the different channels and two indices into the spatial coordinates of each channel, as seen in fig. 2.1. The output is sometimes referred to as the feature map.

To downsample the output of the convolution operation, it is possible to sample only every s pixels in each direction in the output. The convolution stride, s, can be separate for each direction of motion. To allow for independent control of kernel width and output

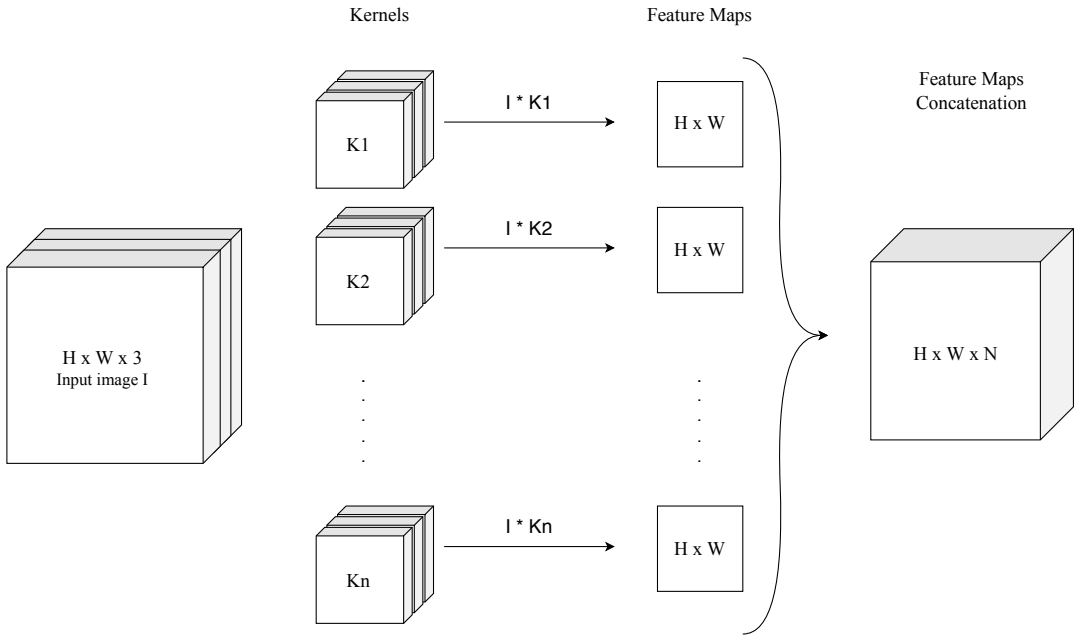

Figure 2.1: Image convolution, with N generic square kernels with three channels.

size, the input to the convolution operation can be padded to make it wider. Without this feature, the width of the representation shrinks by one pixel less than the kernel width at each layer. Padding is a tool to prevent the shrinking the spatial extent of the network and the use of small kernels, since this imposes a limit to the expressive power of the network.

2.3.2 Pooling Layer

In addition to discrete convolutions themselves, pooling operations make up another important building block in CNNs. Pooling operations reduce the size of feature maps by using some function to summarise subregions, such as taking the average or the maximum value.

Pooling works by sliding a window across the input and feeding the content of the window to a pooling function. In some sense, pooling works very much like a discrete convolution, but replaces the linear combination described by the kernel with some other function.

2.3.3 Transposed Convolutional Layer

The need for transposed convolutions generally arises from the desire to use a transformation going in the opposite direction of a normal convolution, i.e., from something that has the

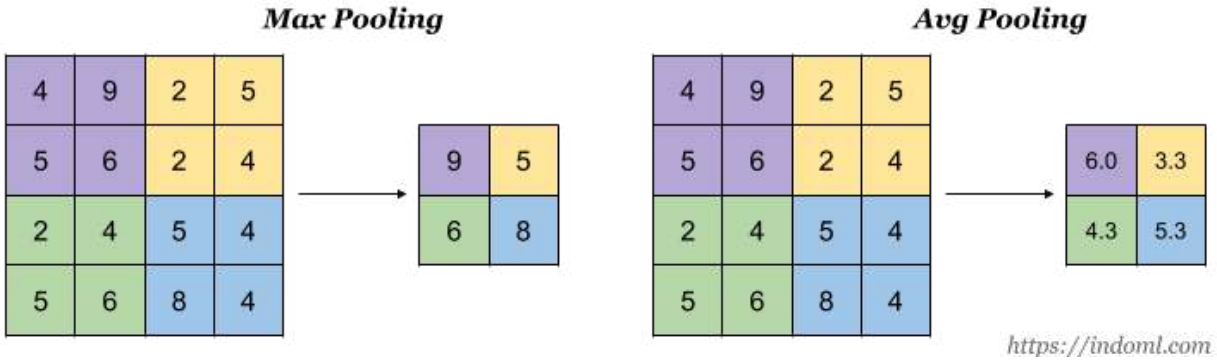

Figure 2.2: Example of a max pooling layer and a average pooling layer.

shape of the output of some convolution to something that has the shape of its input while maintaining a connectivity pattern that is compatible with said convolution. For instance, one might use such a transformation as the decoding layer of a convolutional autoencoder or to project feature maps to a higher-dimensional space.

Transposed convolutions – also called fractionally strided convolutions or deconvolutions – work by swapping the forward and backward passes of a convolution. Its has been shown [24] that deconvolution experiences uneven overlap when the kernel size is not divisible by the stride. The overlap pattern also forms in two dimensions. The uneven overlap creates a characteristic checkerboard-like pattern of varying magnitudes. While the network could, in principle, carefully learn weights to avoid this, ensuring the output is evenly balanced, in practice neural networks struggle to avoid it completely. To find a balance is difficult and significantly restricts the possible filters, sacrificing model capacity, especially when there are multiple channels interacting.

In order to avoid these checkerboard patterns, it is possible to separate out upsampling to a higher resolution from convolution to compute features. Using nearest-neighbour interpolation or bilinear interpolation, the image might be resized and the result passed to a convolutional layer. In image generative networks, nearest-neighbour interpolation has been shown to work better than bilinear interpolation.

2.3.4 Residual Conections

In feedforward networks with many stacked layers, the gradient diminishes dramatically as it is propagated backward through the network. A small gradient means that the weights of the initial layers will not be updated effectively with each training session. Since these initial layers are often crucial to recognising the core elements of the input data, it can lead to overall inaccuracy of the whole network. As such, this problem is referred to as the

"vanishing gradients" problem. Different approaches to training feedforward networks have been studied and applied in an effort to address vanishing gradients, such as pre-training, better random initial scaling, better optimisation methods, specific architectures, orthogonal initialisation, etc.

The simplest solution is to use other activation functions, such as ReLU. Residual networks [13] are another solution, as they provide residual connections straight to earlier layers. As seen in Fig. 2.3, the residual connection directly adds the value at the beginning of the block, x, to the end of the block ($F(x) + x$). This residual connection doesn't go through activation functions that "squashes" the derivatives, resulting in a higher overall derivative of the block.

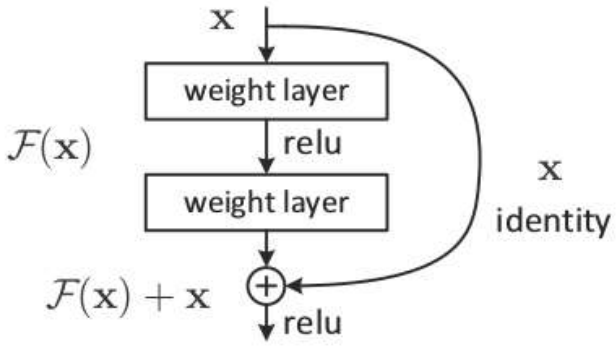

Figure 2.3: Generic residual block.

2.3.5 Normalisation Layer

When training deep neural networks, the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change, slows down the training by requiring lower learning rates and careful parameter initialisation. By making normalisation a part of the model architecture and performing the normalisation for each training mini-batch, or batch normalisation [16], it becomes possible to use much higher learning rates and be less careful about initialisation.

Both the input and output of a batch normalisation layer are four dimensional tensors, referred to as $I_{b,h,w,c}$ and $O_{b,h,w,c}$, respectively. The dimensions correspond to batch b, two spatial dimensions h and w, and channel c, and respectively. Batch normalisation applies the same normalisation for all activations in a given channel:

$$O_{b,h,w,c} = \gamma_c \frac{I_{b,h,w,c} - \mu_c}{\sqrt{\sigma^2 + \epsilon}} + \beta_c \tag{2.9}$$

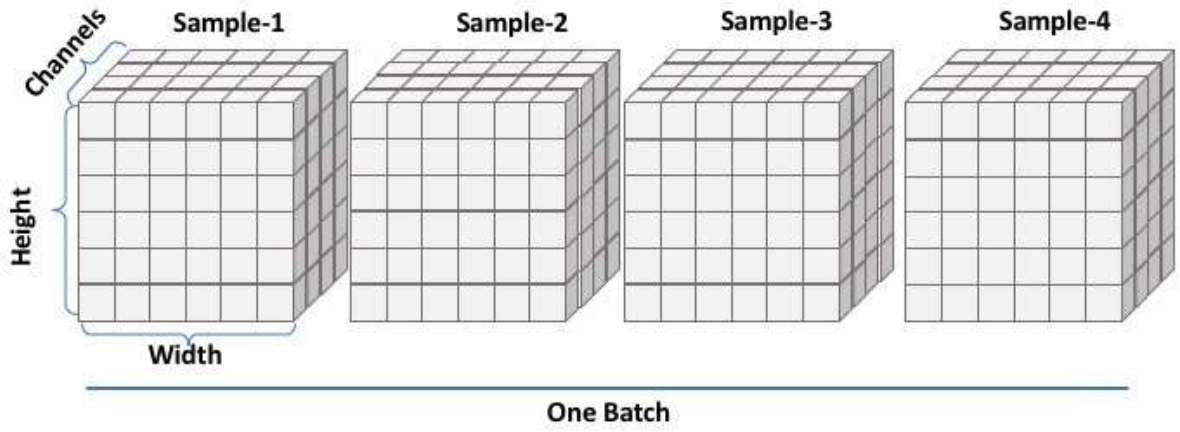

Figure 2.4: One batch of four feature maps.

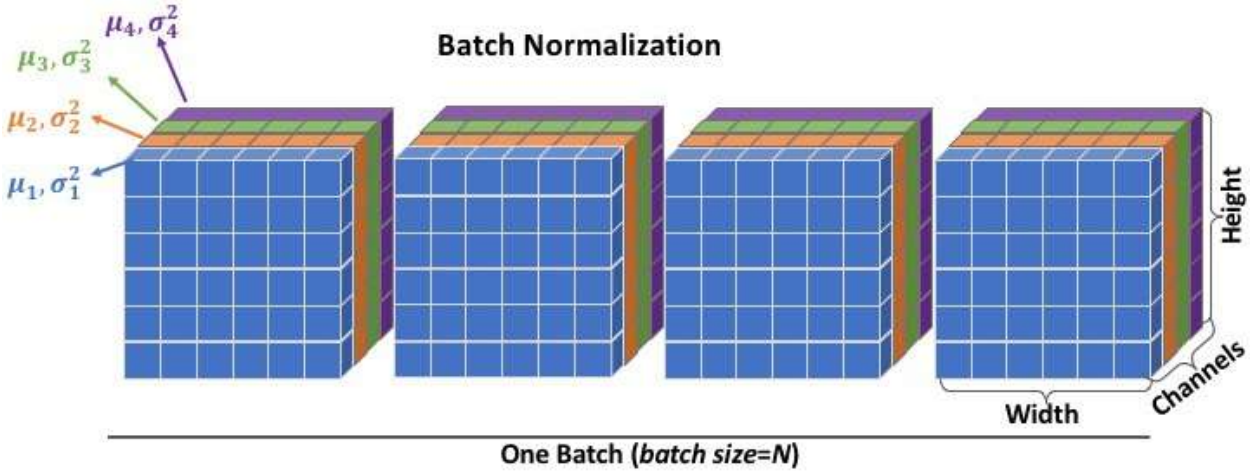

Figure 2.5: Batch Normalisation.

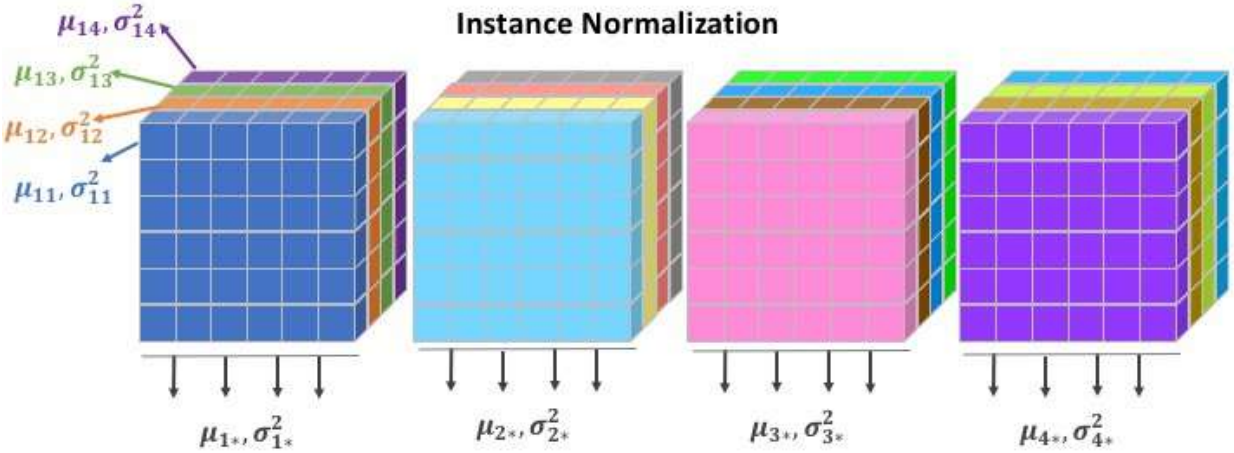

Figure 2.6: Instance Normalisation.

Where, μ_c is the mean activation from all input activations in channel c across all features b in the entire mini-batch and all spatial h, w locations; and σ_c is the standard deviation, calculated in an analogous manner (fig. 2.5). Normalisation is followed by a channel-wise affine transformation parameterized through γ_c, β_c, which are learned during training.

In order to combine the effects of instance-specific normalisation and batch normalisation, [34] proposes to replace the latter by the instance normalisation (also known as "contrast normalisation") layer (see fig. 2.6). Different from batch normalisation layers, in instance normalisation μ and σ are computed across spatial dimensions independently for each channel and each sample.

2.3.6 Convolutional LSTM Layer

When data is collected during a time interval, it is characterised as a Time Series. A Recurrent Neural Network architecture is usually deployed in such occasions. In this kind of architecture, the model passes the previous hidden state to the next step of the sequence, therefore holding information on previous data the network has seen before and using it to make decisions. In other words, the data order is extremely important.

The use of convolution LSTMs [36] to simultaneously learn spatial and temporal information in videos has gained traction recently. A deep network of convolutional LSTMs allows the model to access the entire range of temporal information at all spatial scales of the data, which makes it ideal for video processing. A convolutional LSTM layer contains an internal cell state c_t and calculates a hidden state h_t, used as the output for subsequent layers, as well as for state-to-state transitions. While processing a video sequence, c_t and h_t can be viewed as images of appropriate size maintained by the network with relevant information based on what it has seen in the past.

2.4 VGG

A VGG [31] is a very deep convolutional network (up to 19 weight layers) for large-scale image classification. In neural style transfer, the feature space is provided by the 16 convolutional and 5 pooling layers of the VGG-19 network, and the fully connected layers are discarded. The network's architecture is based on two fundamental computations:

- Linearly rectified convolution with filters of size $3 \times 3 \times k$ where k is the number of input feature maps. Stride and padding of the convolution is equal to one such that the output feature map has the same spatial dimensions as the input feature maps.
- Maximum pooling in non-overlapping 2×2 regions, which down-samples the feature maps by a factor of two.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.7: VGG schematic overview. The last two columns (D and E) correspond to the VGG-19 and the VGG-16 versions, typically used in neural style transfer.

These two computations are applied in an alternating manner (see Fig. 2.7). A number of convolutional layers are followed by a max-pooling layer. After each of the first three pooling layers, the number of feature maps is doubled. Together with the spatial down-sampling, this transformation results in a reduction of the total number of feature responses by a factor of two. Fig. 2.7 provides a schematic overview over the network architecture and the number of feature maps in each layer. Neural style transfer relies only on the convolutional layers, so the input images can be arbitrarily large. The first convolutional layer has the same size as the image and, for the following layers, the ratio between the feature map sizes remains fixed. Generally, each layer in the network defines a non-linear filter bank, whose complexity increases with the position of the layer in the network.

3 Classic Neural Style Transfer

Style transfer concerns the rendering of the semantic content of an image in different styles [8]. The most important prerequisite of style transfer is to to find image representations that independently model variations in the semantic image content and the style in which it is presented. In other words, to generate an arbitrary photograph with the appearance of a given artwork, it is necessary to separate image semantic information, or content, from style information, so that they can be recombined and new images be generated.

In neural style transfer, content and style are modelled on deep image representations. New images are generated by performing a *pre-image search* (a term sometimes used in the literature) to match feature representations of a target (or content) image and a source (or style) image to feature representations of a generated image. The VGG [31] with 19 convolutional layers and without fully connected layers is the network of choice for style transfer. The first section of this chapter describes how to generate a new image with similar semantic content as a given target image using feature maps taken from a pre-trained VGG. In the second section, a model of natural texture generation based on the feature spaces of a VGG is explained. The third section combines the knowledge of the previous two to introduce the fulcral algorithm of this thesis: neural style transfer. The final section conciliates the algorithm with parametric texture models from chapter 1.

3.1 Content Representations

Convolutional Neural Networks trained on object recognition develop a representation of the image that makes object information increasingly explicit along the processing hierarchy i.e., the input image is transformed into feature maps representations that codify the actual content of the image, and not its detailed pixel values.

In [22], the authors reconstruct an image based on features extracted of CNNs in an effort to understand what information is being retained by its layers. To do so, a representation

is modelled as a function $\phi(x)$ of the image x, so that an approximated inverse ϕ^{-1} can be computed and x reconstructed from $\phi(x)$.

In more formal terms, given a representation function $\phi : R^{H \times W \times C} \rightarrow R^d$ and a representation $\phi_0 = \phi(x_0)$ to be inverted, reconstruction is achieved by finding an image $x \in R^{H \times W \times C}$ that minimises the objective:

$$x^* = \arg \min_{x \in R^{H \times W \times C}} l(\phi(x), \phi_0) \tag{3.1}$$

where the loss l compares the image representation $\phi(x)$ to the target one ϕ_0. A typical loss function l is usually the Euclidean distance:

$$l(\phi(x), \phi_0) = \|\phi(x) - \phi_0\|^2 \tag{3.2}$$

Gatys et al [8] demonstrate what information is being contained in each layer of a VGG network by reconstructing the image only from the feature maps in that layer (see fig. 3.3). High-level content, meaning objects and their arrangement in the input image, is captured by higher layers in the network; while lower-level content, the exact pixel values of the original image, is captured by lower layers. The feature responses in higher layers of the CNN are refered to as the content representation in Neural Style Transfer literature.

Given an input image $\vec{x}$ and a pre-trained Convolutional Neural Network where each layer defines a non-linear filter bank, the filter responses to that image are encoded in the hidden layers of the CNN. A layer l with N_l distinct filters has N_l feature maps, each of size M_l, where M_l is the height (H_l) times the width (W_l) of the feature map. The filter responses in a layer l, a 3-D matrix of size $H_l \times W_l \times N_l$ can be reorganized as a 2-D matrix $F^l \in R^{N_l \times M_l}$, where F^l_{ij} is the activation of the i^{th} filter at position j in layer l, as seen in fig. 3.1.

By performing gradient descent on a white noise image, it is possible to find another image that matches the feature responses of the original image. Let $\vec{p}$ and $\vec{x}$ be the original content image and the image that is generated, and P^l and F^l their respective feature representation in layer l. The squared-error loss between two feature representations, based on 3.2, is

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \tag{3.3}$$

The gradient with respect to the image $\vec{x}$ can be computed using standard error back-propagation, so that the initial white noise image $\vec{x}$ can be be iteratively changed until it

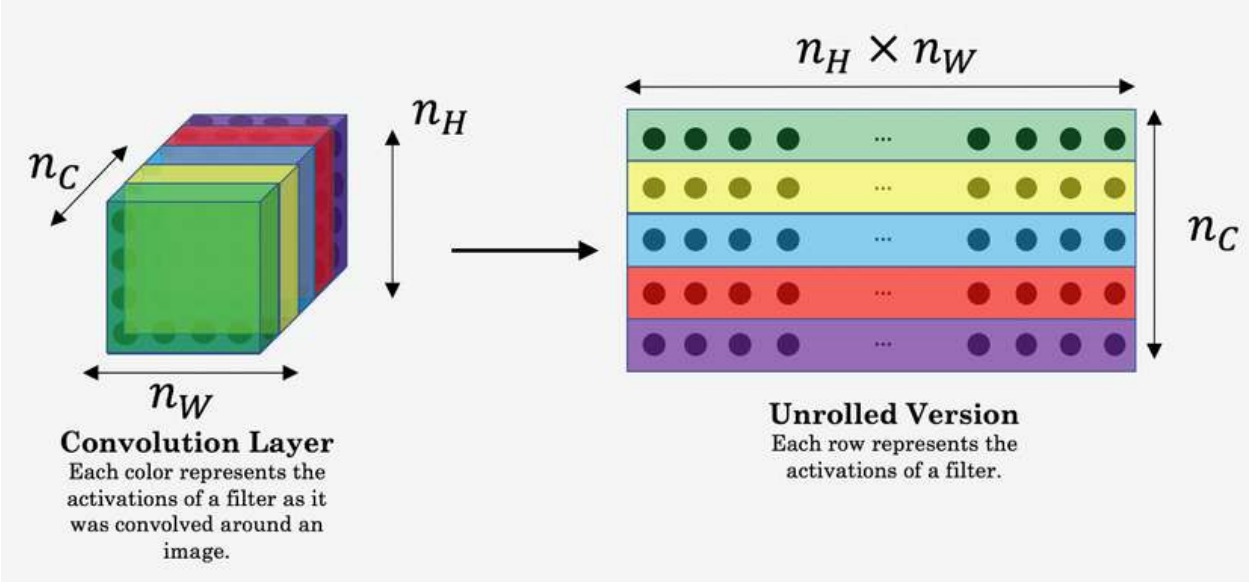

Figure 3.1: Reshaping feature maps by joining both spatial dimensions into one.

generates the same response in a certain layer of the VGG as the original content image $\vec{p}$.

3.2 Style Representations

Transferring the style from one image onto another can be interpreted as a problem of texture transfer, where the objective is to synthesise a texture from a source image while constraining the texture synthesis in order to preserve the semantic content of a target image. Chapter 1 provides an historical overview of the evolution of parametric texture synthesis models.

To characterise a given texture $\vec{s}$, it is necessary to pass it through the convolutional neural network first and extract the activations for each layer l in the network. Since textures are per definition stationary, a texture model needs to be agnostic to spatial information [7]. A summary statistic that discards the spatial information in the feature maps is given by the similarities between the responses of different features. These feature similarities are, up to a constant of proportionality, given by the Gram matrix $G^l \in R^{N_l \times N_l}$, where G^l_{ij} is the inner product between feature map i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{3.4}$$

A set of Gram matrices $\{G_1, G_2, ..., G_L\}$ computed with feature maps extracted from a set of layers $\{l_1, ..., l_L\}$ in the network in response to a given texture provides a stationary description of the texture, which fully specifies it.

To generate a new texture on the basis of a given image, we use gradient descent from

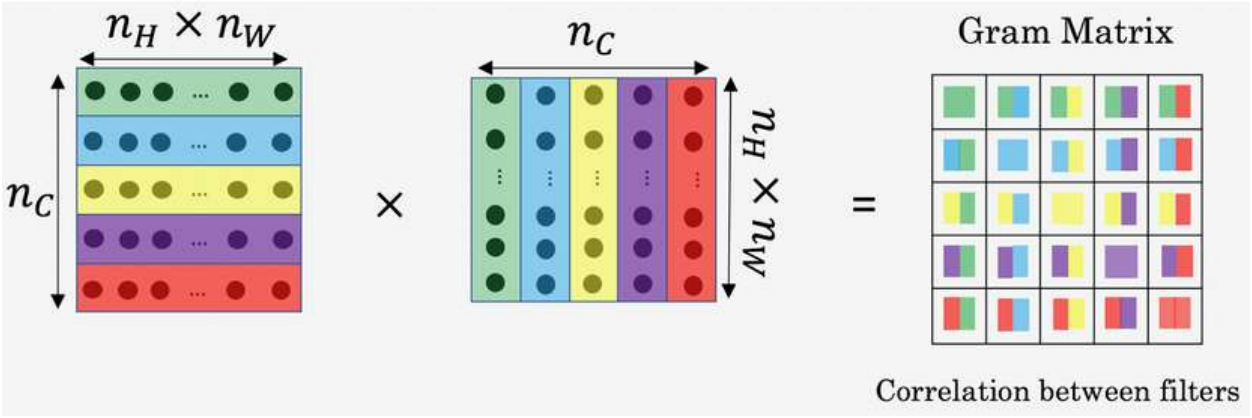

Figure 3.2: A Gram matrix as a result of matrix multiplication of feature maps.

a white noise image to find another image that matches the Gram-matrix representation of the original image. This optimisation is done by minimising the mean-squared distance between the entries of the Gram matrix of the original image and the Gram matrix of the image being generated.

If $\vec{x}$ and $\hat{\vec{x}}$ are the original image and the image that is generated, and G^l and $\hat{G}^l$ their respective Gram-matrix representations in layer l, then the style loss for this layer is:

$$E_l(\vec{x}, \hat{\vec{x}}) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - \hat{G}_{ij}^l)^2 \tag{3.5}$$

And the total style loss:

$$L(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l \tag{3.6}$$

where w_l are weighting factors of the contribution of each layer to the total loss. The gradients of $L(\vec{x}, \hat{\vec{x}})$ with respect to the pixels $\hat{\vec{x}}$ can be computed with back-propagation.

As the source image passes through the layers of the VGG network, the receptive field size associated to each layer increases, which translates to the size of the regions in which spatial information is preserved increasing as well, as seen in fig. 3.3.

3.3 Style Transfer

To transfer the style of an artwork $\vec{a}$ onto a photograph $\vec{p}$, a new image is generated to simultaneously match the content representation of $\vec{p}$ and the style representation of $\vec{a}$. Style transfer is then achieved by jointly minimising the distance of the feature representations of a white noise image from the content representation of the photograph in one layer and the style representation of the painting defined on a number of layers of the Convolutional

Neural Network. The total loss function is a linear combination of 3.3 and 3.5:

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x}) \tag{3.7}$$

where α and β are the weighting factors for content and style reconstruction, respectively. Tuning both these parameters can regulate the emphasis on style or content reconstruction in the final image. A strong emphasis on style will result in images that match the appearance of the artwork, effectively giving a texturised version of it, but show hardly any of the photograph's content. Placing strong emphasis on content, one can clearly identify the photograph, but the style of the painting is not as well-matched.

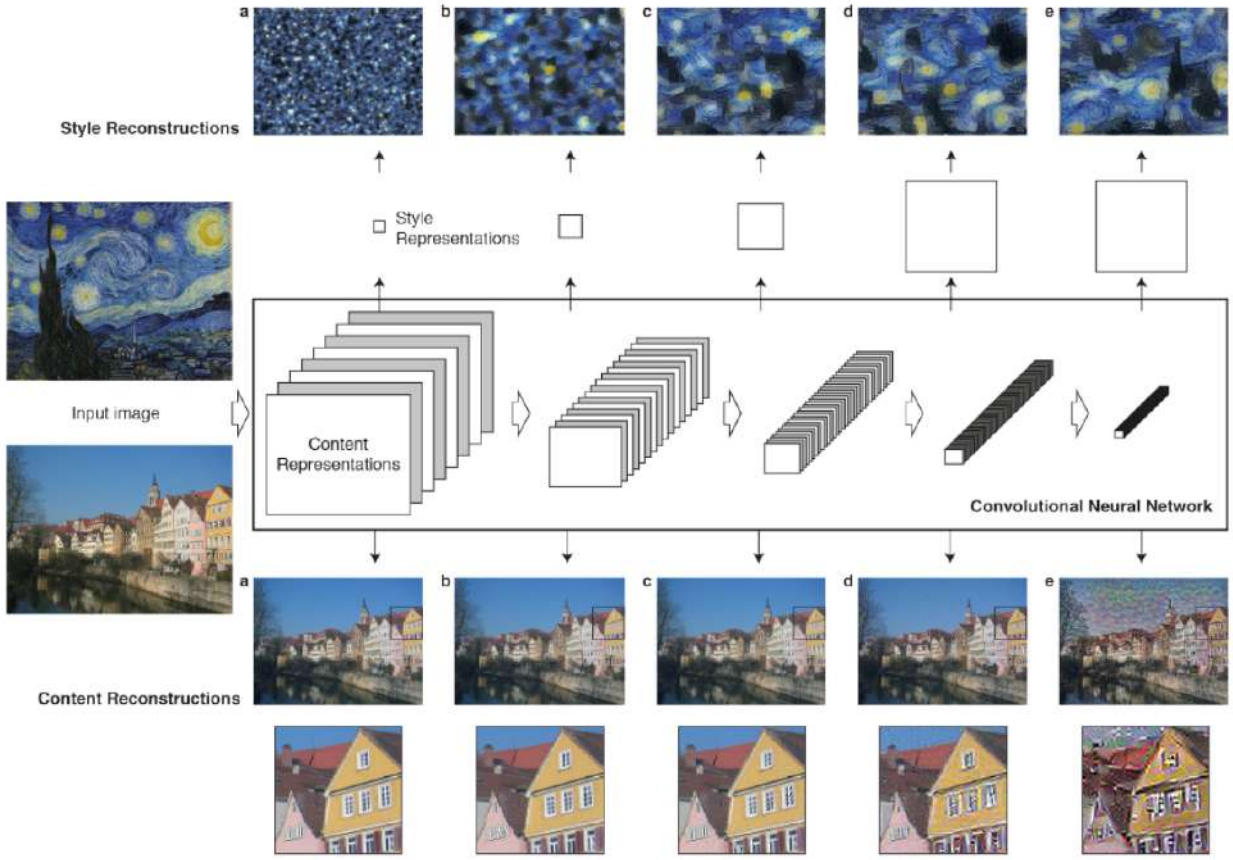

Figure 3.3: A content image and a style image pass through a VGG network. Five different layers with increasing number of filters are chosen to extract feature maps with increasing number of feature maps. Style representations, as Gram matrices, are used to compute style reconstructions. The first style image, a (top of the image), was reconstructed with the first of five layers; the second style image, b, with the first two layers; and so forth until reconstruction style image e, built with feature maps from all five layers. Content reconstructions computed with feature maps taken from individual layers. Image taken from [6].

3.4 Interpreting the Gram Matrix

Following the analysis on style loss function presented in [27], a Gram matrix is statistically related to neither the mean nor covariance matrices, but instead to the matrix of non-central second moments. Considering the case of a feature map F with m features, so that $F \in R^{n \times m}$ (as seen in fig 3.2), then the statistics of the features in F can be summarized with a m dimensional random variable X. If the Gram matrix $G(F)$ is normalized by the number of samples, n, then it becomes a sample estimator for the second non-central mixed moments $E[XX^T]$:

$$\frac{1}{n}G(F) = E[XX^T] \tag{3.8}$$

Let the mean feature $\mu = E[X]$; by a general property of covariance matrices, $\Sigma(X) = E[XX^T] - \mu\mu^T$, where Σ indicates a covariance matrix. Finally,

$$\frac{1}{n}G(F) = E[XX^T] = \Sigma(X) + \mu\mu^T \tag{3.9}$$

It is then established that the classic neural style algortihm uses a pre-trained network to extract image statistics, and image generation is reduced to a problem of sampling at random from the set of images that match certain statistics.

4 Feed Forward Style Transfer

The method introduced by Gatys et al. [8] is a slow, memory consuming optimisation process. Moving the computational burden to a learning stage can help overcome its limitations. The works introduced by [18] and [33] introduce the training of a feed-forward convolutional network to generate multiple samples of the same texture of arbitrary size and to transfer artistic style from a given image to any other image. The results obtained are of comparable perceptual quality to [8] and hundreds of time faster, since they require a single evaluation of the network and do not incur in the cost of backpropagation. However, it is necessary to train a separate generator network for each style and set of hyperparameters α and β defined in equation 3.7. The feed-forward network cannot generalise beyond its trained set of style images, losing the characteristic versatility of the image optimisation approach presented in chapter 3. On the other hand, once trained, it can synthesize an arbitrary number of images of arbitrary size in an efficient, feed-forward manner.

In this chapter, a fast generator system for artistic style will be described in detail. The first section will provide a high-level overview of the system and its associated training method. The second section will give a low-level overview of the generator network's architecture and its constituting layers.

4.1 Proposed Method

The training procedure for this network is most similar to the one proposed in the work of Johnson et al. [18], with some key differences. For instance, Johnson et al.s' train a network with a total variation loss to prevent checkerboard patterns and image noise from appearing. However, replacing fractionally strided convolutional layers by nearest-neighbour upsampling layers followed by convolutional layers prevents the use of total variation loss during training. To improve results, all batch normalisation layers in Johnson et al.'s original architecture are replaced by instance normalisation layers.

In addition to the pre-trained VGG network used to define style and loss functions, the system is also composed by an image transformation network f_θ. This second one is a deep residual convolutional neural network parameterized by weights θ that maps input images x into output images y: $y = f_\theta(x)$. For a chosen set of VGG layers, the total content loss and the total style loss are calculated as seen in chapter 3. Denoting the two sets of loss functions used to train this network by $l_i^c(y, p)$ and $l_j^s(y, a)$, where p is the target content image and a is the target style image, then training a image transformation network by stochastic gradient descent is akin to minimising a weighted combination of loss functions:

$$\theta^* = \arg\min_\theta E_{x,p,a}[\sum_{i=1} \lambda_i l_i^c(f_\theta(x), p) + \sum_{j=1} \lambda_j l_j^s(f_\theta(x), a)] \tag{4.1}$$

The image transformation network is trained with the system depicted in Algorithm 1 and Fig. 4.1, consisting of an image dataset and a loss network, the 16 layer VGG. The feature maps extracted from the VGG come from a previously selected set of style layers (**conv1_2**, **conv2_2**, **conv3_3** and **conv4_3**) and one content layer (**conv3_3**). Before beginning training the network, a source style image is passed through the VGG-16. The 4 feature maps produced in the chosen layers are used to compute 4 target Gram matrices that will be stored and later used in the style reconstruction function 3.5. Training the image transformation network can be resumed to following the next steps:

Compounding on this algorithm, Fig. 4.1 displays the training system, which can be summarised in the following steps:

1. Each training iteration begins with an image batch (referred to as *Input Content Image* in diagram 4.1) taken from the image dataset and fed through the VGG network and the image transformation network.

2. The produced batch of feature maps is extracted from the VGG content layer and will be the target objective of content reconstruction.

3. The batch of images produced by the last layer of the image transformation network (*Output Image* in diagram 4.1) are then fed through the VGG in order to extract a batch of feature maps.

4. Feature maps recovered from style layers will be used to compute Gram matrices; their distance to target Gram matrices from step 1 measured by the style loss function (eq. 3.6).

Algorithm 1 Training an Image Transformation Network

procedure TRAININGf_θ
  $b \leftarrow$ batch size
  $a \leftarrow$ style image
  for each VGG style layer l_s **do**
    $G_s^l \leftarrow$ batch of Gram matrices from layer l_s, image a
  end for
  for two epochs on chosen image dataset **do**
    $TotalLoss \leftarrow 0$
    $p \leftarrow$ batch of images of size b from dataset
    $f_\theta(p) \leftarrow$ batch of network outputs for batch p
    for each VGG content layer l_c **do**
      $F_c^l \leftarrow$ batch of feature maps from layer l_c, batch p
      $\hat{F}_c^l \leftarrow$ batch of feature maps from layer l_c, batch $f_\theta(p)$
      $TotalLoss \mathrel{+}= \frac{\lambda_{lc}}{b}\|\hat{F}_c^l - F_c^l\|_F^2$
    end for
    for each VGG style layer l_s **do**
      $\hat{G}_s^l \leftarrow$ batch of gram matrices from layer l_s, batch $f_\theta(p)$
      $TotalLoss \mathrel{+}= \frac{\lambda_{ls}}{4N_l^2M_l^2b}\|\hat{G}_s^l - G_s^l\|_F^2$
    end for
    $\theta \leftarrow$ ADAM(θ, $TotalLoss$)
  end for
end procedure

5. Feature maps recovered from the content layer will be measured against the feature maps from step 3 with content loss function (eq. 3.3).

6. The weight parameters θ of the image transformation network are updated so that the total loss (eq. 3.7), a linear combination of losses computed in step 5 and step 6, is minimised.

The image transformation network is trained on the Microsoft COCO dataset [21]; each one of the 80.000 training images is resized to 256x256; the batch size is 4; training takes 40.000 iterations, roughly two epochs over the training data; optimiser of choice is Adam with a learning rate of 1×10^{-3}.

4.2 Proposed Network Architecture

The image transformation network follows architectural guidelines set forth for generative networks by [24, 18, 34, 26]. Relying on the architecture proposed by Johnson et al. [18], results are improved by replacing transposed convolutional layers by nearest neighbour upsampling followed by a convolutional layer [24], and batch normalisation layers by instance

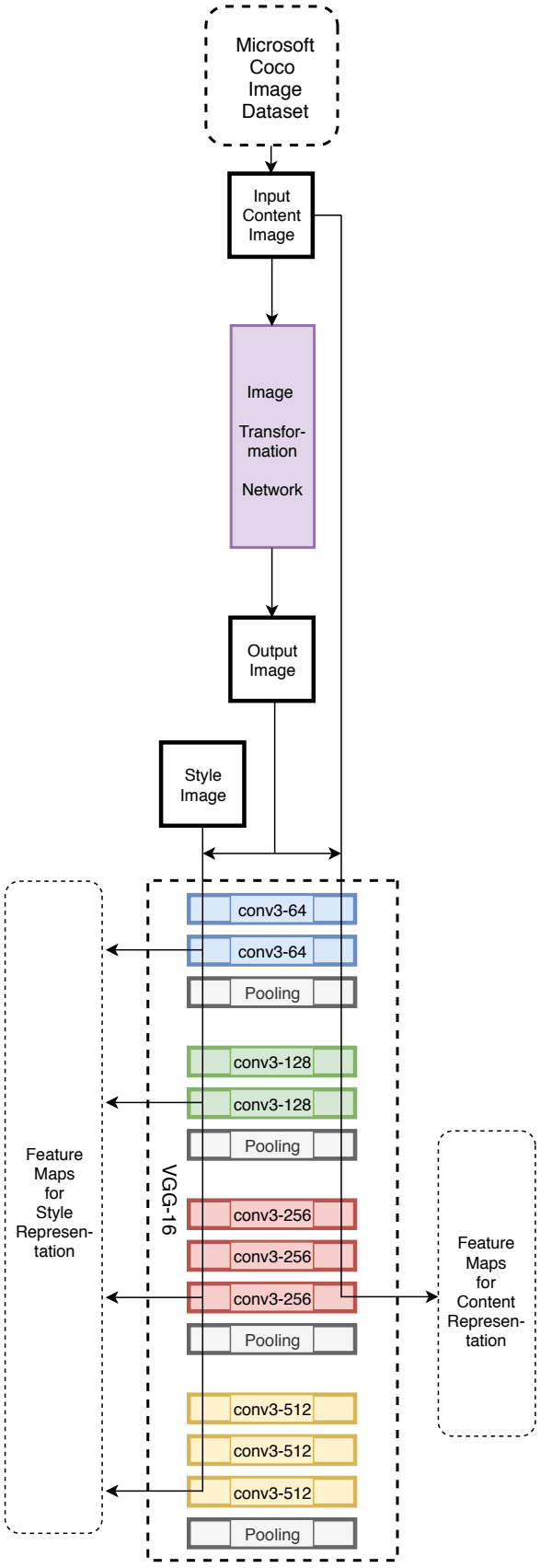

Figure 4.1: System overview for image transformation network training.

normalisation layers [34].

- The network is fully convolutional, and the usual down sampling pooling functions are replaced by strided convolutions, allowing the network to learn its own spatial downsampling;

- To combat checkerboard patterning, the deconvolution operation, typically used to generate a larger image from lower resolution descriptions, is replaced by nearest-neighbour interpolation followed by a convolutional layer. This was first proposed by Odena et al. [24];

- The ReLU activation is used in all layers except for the output, which uses Tanh as an activation function. A bounded activation allows the model to learn more quickly to saturate and cover the color space;

- Ulyanov et al. [34] introduced an instance normalisation layer. In this network's architecture, it follows all non-residual convolutional layers in order to stabilize learning by normalising each feature map to have zero mean and unit variance;

- The network body is comprised of five residual blocks [13], where style is effectively applied, just as it was in Johnson et al.'s architecture [18];

- Each convolutional layer is preceded by reflection padding, to avoid border patterns;

- Other than the first and last layers which use 9×9 kernels, all convolutional layers use 3×3 kernels;

A more detailed outlook on network architecture is presented in table 4.1.

There are several benefits to downsampling and upsampling in a generative network, even though the input and output are both color images of the same shape, $256 \times 256 \times 3$. In computational terms, downsampling allows for a deeper network with the same number of multiply-add operations as a shorter network where downsampling isn't applied. Another benefit associated with this practice is the augmentation of receptive field sizes. Without downsampling, each additional 3×3 convolutional layer in a stack model increases the effective receptive field size by 2. After downsampling by a factor of D, each 3×3 convolution increases the effective receptive field size by $2D$, giving larger effective receptive fields with the same number of layers. As each pixel in the output image increases its effective receptive

Layer	Details	Activation Size
Input		256 x 256 x 3
Convolutional Layer	kernel size 9; stride 1; 32 filters	256 x 256 x 32
Convolutional Layer	kernel size 3; stride 2; 64 filters	128 x 128 x 64
Convolutional Layer	kernel size 3; stride 2; 128 filters	64 x 64 x 128
Residual Block	kernel size 3; stride 1; 128 filters	64 x 64 x 128
Residual Block	kernel size 3; stride 1; 128 filters	64 x 64 x 128
Residual Block	kernel size 3; stride 1; 128 filters	64 x 64 x 128
Residual Block	kernel size 3; stride 1; 128 filters	64 x 64 x 128
Residual Block	kernel size 3; stride 1; 128 filters	64 x 64 x 128
Upsampling Layer	nearest-neighbour; factor of 2	128 x 128 x 128
Convolutional Layer	kernel size 3; stride 1; 64 filters	128 x 128 x 64
Upsampling Layer	nearest-neighbour; factor of 2	256 x 256 x 64
Convolutional Layer	kernel size 3; stride 1; 32 filters	256 x 256 x 32
Convolutional Layer	kernel size 9; stride 1; 3 filters	256 x 256 x 3

Table 4.1: Feed-forward network architecture

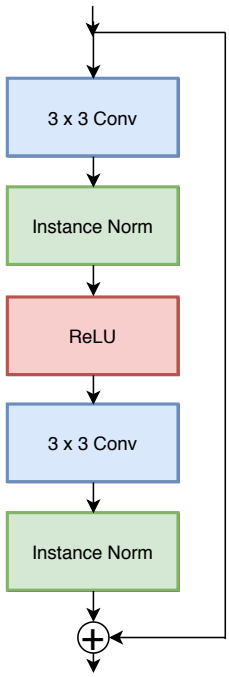

Figure 4.2: Inside the residual block.

field in the input, results will bear higher perceptual quality, since large parts of the output image will be stylized in a coherent way.

To train very deep networks for image classification, [13] proposes the use of residual connections. This is also an appealing property for image transformation networks, since in most cases the output image should share structure with the input image. The residual block used in the image transformation network is built according to guidelines from [11], and can be observed in fig. 4.2.

5 Style Transfer Applied to Video

Building upon the approaches from chapters 3 and 4, style transfer is extended to video sequences in this chapter. Given an artistic image, we transfer its particular style of painting to the entire video. Processing each frame of the video independently leads to flickering and false discontinuities, since the solution of the style transfer task is not stable.

The source of this instability is the Gram matrix style loss; the solution set of the Gram matrix matching objective is defined as a *n-sphere* - a generalisation of the three dimensional sphere to dimensions $n \geq 4$ - with radius determined by the trace of the style image's Gram matrix [12]. Due to being a non-convex loss function, minor changes in semantic content in the target image can pull the synthesized image to different solutions of this Gram matrix matching objective. For a small solution set, different solutions will result in similarly stylized images (low instability). In the opposite scenario, with a large solution set, different solutions may result in very different stylized images (high instability).

In an effort to regularise style transfer and to preserve smooth transitions between individual frames of the video, a temporal loss function [28] and a different network training method are used. In the first section of this chapter, concepts of computer vision, such as image warping and optical flow, are introduced. The second section is concerned with the new temporal loss function; and the third with the practical estimation of optical flow. In the fourth section, a training method for a video transformation network will be described. This method, which is optical flow dependent, follows a training procedure inspired from the work of Ruder et al. [29] and the architecture proposed in chapter 4. The final section in this chapter presents an alternative approach to the network architecture and training procedure.

5.1 Occlusion and Motion Boundary Detection

Algorithms for estimating motion in video sequences are among the most widely used in computer vision. The most general (and challenging) version of motion estimation is to compute an independent estimate of motion at each pixel, which is generally known as optical flow. Dense optical flow estimation will enable dense motion tracking and to extract information with fine granularity. This type of information is typically stored in a optical flow field that relates the positions of all pixels between pairs of consecutive frames. Forward and backward optical flow fields can be used to detect two types of regions necessary to maintain temporal consistency in a sequence of frames: occlusions and motion boundaries.

Occlusions occur whenever two or more images are taken of the same scene from slightly different viewpoints or at slightly different times, and there are points in one image that do not appear in the other image. More precisely, an occlusion is a set of points that appear in one image whose corresponding world points are not visible in another image because an opaque object is blocking the view of those points in the other image.

Optical flow can be simply described as a field that consists of large regions where it varies smoothly, divided by boundaries where the flow abruptly changes. Motion boundaries are then defined as the discontinuities of the optical flow between two frames and reveal the location of occlusion boundaries, which very often correspond to physical object boundaries. Most state of the art methods for estimating motion boundaries are based on optical flow gradient analysis [32].

Image warping is a transformation which maps all positions in one image plane to positions in a second plane. Given a frame I_t, it should be possible to find the correspondence between the original position $\mathbf{p}_i = (u_i, v_i)^T$ of a set of points $i = 1, ..., n$ in I_t and their new position $\mathbf{q}_i = (x_i, y_i)^T$ in the following frame I_{t+1}. For each point i both frames have in common, $I_{t+1}(\mathbf{q}_i) = I_t(\mathbf{p}_i)$. Information on how relate the positions of each individual pixels between frame I_t and frame I_{t+1} is typically stored in a displacement field. A displacement field is formally a function f used in a mapping function T with the general form $T(\mathbf{p}) = \mathbf{p} + f(\mathbf{p})$. This mapping process is also referred to as forward warping, since pixels are in a sense moved 'forward' from the coordinate frame of the old image I_t to the new image I_{t+1}.

The problem with this intuitive approach is that the transformation function is generally neither injective nor surjective. Due to the discrete nature of pixel images, non-integer values of the transformation function T have to be rounded. As a result, not every pixel in the

new image I_{t+1} will be necessarily assigned a value and some pixels can be assigned several times.

Backward warping, a mapping function where, for every pixel of the new image I_{t+1}, a coordinate in the original image I_t is computed, is used to eliminate this problem. This involves the inverse T^{-1} of the transformation function. In analogy to forward warping, it is possible that $T^{-1}(\mathbf{q})$ yields a non-integer value.

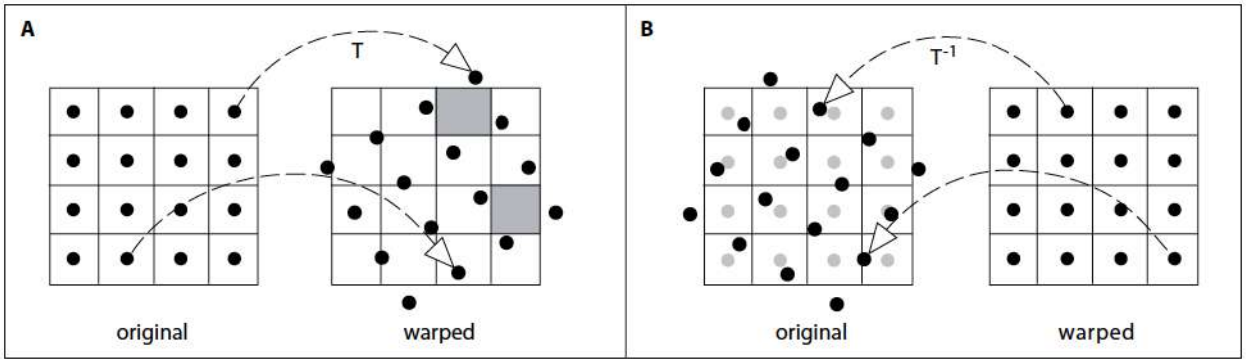

Figure 5.1: Forward and backward image warping. In the case of foward warping (A), holes can occur in the warped image, marked in gray. Backward warping (B) eliminates this problem since intensities at locations that do not coincide with pixel coordinates can be obtained from the original image using an interpolation scheme.

Extrapolating from some of the concepts presented in the previous two paragraphs, it can be concluded that each of the points in frame I_t can be tracked to the next frame I_{t+1} by using the forward optical flow field $\mathbf{o}_t(x, y) := (u_t(x, y), v_t(x, y))$. Analogously, points in frame I_{t+1} find their correspondence in frame I_t with the backward optical flow $\mathbf{o}_{t+1}$. In mathematical notation, the forward flow is:

$$(x_{t+1}, y_{t+1})^T = (x_t, y_t)^T + (u_t(x_t, y_t), v_t(x_t, y_t))^T. \tag{5.1}$$

As expected, the optical flow is subpixel accurate; x_{t+1} and y_{t+1} will usually end up between grid points, as seen in fig. 5.1 A.

In the specific case of video style transfer, given a pair of consecutive frames, there is a demand to find out common points between the two images in order to enforce a consistent appearance between the both of them. Taking frame I_{t+1} as reference, if a given point (x_{t+1}, y_{t+1}) doesn't have a correspondent point (x_t, y_t) in the preceding frame I_t, then either it belongs to a disoccluded region, or to a motion boundary region, and temporal consistency can't, and shouldn't, be enforced. Let

$$(\hat{x}_t, \hat{y}_t)^T = (x_{t+1}, y_{t+1})^T + \mathbf{o}_{t+1}(x_{t+1}, y_{t+1})^T \tag{5.2}$$

be a pair of coordinates, with sub-pixel accuracy, in frame I_t. To infer the forward optical flow $\hat{\mathbf{o}}_t = (\hat{u}_t, \hat{v}_t)$ at this point, bilinear interpolation is used. In a non-disocclusion case, the backward flow vector $\mathbf{o}_{t+1}$ points in the inverse direction as the forward flow vector $\hat{\mathbf{o}}_t$, so the following identities should be true:

$$\begin{aligned} u_{t+1}(x_{t+1}, y_{t+1}) &= -\hat{u}_t(x_{t+1} + u_{t+1}, y_{t+1} + v_{t+1}) \\ v_{t+1}(x_{t+1}, y_{t+1}) &= -\hat{v}_t(x_{t+1} + u_{t+1}, y_{t+1} + v_{t+1}) \end{aligned} \tag{5.3}$$

If this consistency requirement is not satisfied, the point is either getting disoccluded from frame I_t to frame I_{t+1}, or the flow was not correctly estimated. Since there are always some small estimation errors in the optical flow, Sundaram *et al.* [32] suggest a tolerance interval that allows estimation errors to increase linearly with the motion magnitude:

$$|\hat{\mathbf{o}}_t + \mathbf{o}_{t+1}|^2 \geq 0.01(|\hat{\mathbf{o}}_t|^2 + |\mathbf{o}_{t+1}|^2) + 0.5 \tag{5.4}$$

As for motion boundaries, their location, as estimated by the optical flow gradient, is given by the next identity:

$$|\nabla(u_{t+1})|^2 + |\nabla(v_{t+1})|^2 > 0.01|\mathbf{o}_{t+1}|^2 + 0.002 \tag{5.5}$$

Should both these identities be proven true, then the pixel in question either suffers from disocclusion in frame I_{t+1} or belongs to a motion boundary region.

5.2 Temporal Consistency Constraint

The total loss function eq. 3.7 introduced in the classic neural style algorithm can be augmented with a temporal consistency loss $L_{temporal}$ to encourage temporally stable results by penalising the network when its outputs at adjacent time steps significantly vary. The result is a hybrid loss to capitalise on the content information of an input frame, the style information of a given style image, and the temporal information of previously stylised frames.

When styling video sequences, the semantic content in stylised video frames should maintain its appearance in a manner consistent with the motion in the input video. A certain object has to look the same independently of the angle of observation. Optical flow will allow for object and camera movement in the original video to be faithfully processed, hence

its usage in $L_{temporal}$. The basis of a temporal loss function comes from equation 5.1. For all corresponding pixel coordinates in a pair of stylised frames I_t and I_{t+1}, and the forward optical flow field $\mathbf{o}_t(x, y) := (u_t(x, y), v_t(x, y))$, the following difference should be minimised:

$$I_{t+1}(x_{t+1}, y_{t+1}) - I_t(x_t + u_t(x_t, y_t), y_t + v_t(x_t, y_t)) \tag{5.6}$$

This per-pixel difference is accomplished by warping frame I_t and by using bilinear interpolation. Bilinear interpolation has been proven to be differentiable [17]. In fact, this sampling mechanism allows for backpropagation of the loss.

The objective is to warp the stylised result from the previous frame to the current one, and adaptively fuse both together. In other words, some traceable points/regions from the previous frame remain unchanged, while some untraceable points/regions need to be stylised with information from the current frame. This strategy makes sure stylised results along the motion paths will be as stable as possible.

Due to object and camera motion between frames, some pixels in frame I_t may become occluded in frame I_{t+1}; likewise some pixels which are occluded in frame I_t may become disoccluded in frame I_{t+1}. As a result, enforcing the temporal consistency loss between all pixels in both frames would result in artifacts at motion boundaries. A pre-computed occlusion mask is therefore necessary to avoid enforcing the temporal consistency loss for occluded and disoccluded regions and motion boundaries. Let $\mathbf{m}$ be a binary per-pixel mask with the shape of the input image, where it assumes the value 0 in disoccluded regions and motion boundaries, and 1 elsewhere, or in other words, in every pixel where the represented world points are seen in both frames; $\mathbf{x}_t$ be the output of the stylisation network in timestep t; $\mathbf{x}_{t-1}$ the output of the stylisation network in the preceding timestep $t-1$; T^t_{t-1} the warping transformation function, defined with the pre-computed optical flow field between frames I_{t-1} and I_t, applied to $\mathbf{x}_{t-1}$. The formal definition of a temporal loss function, featured extensively in video processing networks [1, 3, 12, 28, 29], is:

$$L_{temporal} = \frac{1}{D}\|\mathbf{m}_{t-1,t} \odot (\mathbf{x}_t - T_{t-1,t}(\mathbf{x}_{t-1}))\|^2_F \tag{5.7}$$

where $D = H \times W \times C$ is the dimensionality of the image.

According to [28], there are two kinds of temporal consistency in videos: long-term consistency and short-term consistency. Long-term consistency is more appealing since it produces stable results over larger periods of time, and can even enforce consistency of the synthesised frames before and after the occlusion. This constraint can be easily enforced

in optimisation-based methods. Unfortunately, it is quite difficult to incorporate it in feed-forward networks, due to limited batch size, computation time and cache memory. Therefore, short-term consistency seems to be more affordable by feed-forward networks in practice.

The implementation presented is a kind of compromise between consistency and efficiency. The feed-forward network is designed to mainly consider short-term relationships (only two frames), but the long-term consistency is partially achieved by propagating the short-term ones. Even if the resulting image obtained at a certain timestep will be a composite of the previous frame combined with features of the current frame, this training loss guarantees that, as long as a point can be traced along motion trajectories, its overall appearance remains unchanged. This approximation will suffer from inconsistency before and after the occlusions, but will prevent flickering artifacts.

Finally, it should be stressed that, unlike the calculations of the content loss and style loss, the temporal loss is calculated using the output of the stylising network and not the features maps extracted from of the loss network. It has been shown [14] that using the higher-level feature maps of the loss network to compute the temporal loss won't prevent flickering effects. The main reason is that higher-level feature maps only capture abstract information.

5.3 Optical Flow Estimation

FlowNet 2.0 [15] is used to obtain the optical flow between consecutive pairs of frames. In its first iteration, FlowNet was the first ever deep convolutional neural network designed to directly estimate the optical flow. FlowNet 2.0 shows several improvements to the FlowNet idea: accuracy that is fully on par with state-of-the-art methods while running orders of magnitude faster than conventional methods. It also reduces the estimation error by more than 50% compared to the initial FlowNet. The authors assert the reliability of their method on a large variety of scenes and applications, highlighting the possible range of applications, made possible by its very crisp motion boundaries results and its successful retrieval of fine structures. In fact, since its release, FlowNet 2.0 has become the working horse for all applications that require accurate and fast optical flow computation, which is why its usage should not be surprising.

5.4 Training a Feed-Forward Network

The final form of the hybrid training loss for video style transfer is:

$$L_{total}(\mathbf{I}_t, \mathbf{a}, \mathbf{x}_t) = \alpha L_{content}(\mathbf{I}_t, \mathbf{x}_t) + \beta L_{style}(\mathbf{a}, \mathbf{x}_t) + \gamma L_{temporal}(\mathbf{x}_t, T_{t-1,t}(\mathbf{x}_{t-1}), \mathbf{m}_{t-1,t}) \quad (5.8)$$

where $\mathbf{a}$ is the style image, $\mathbf{I}_t$ is the current frame to be stylised, $\mathbf{x}_t$ is the image transformation network output, $\mathbf{m}_{t-1,t}$ is the binary mask codifying consistency between both frames, $T_{t-1,t}(\mathbf{x}_{t-1})$ is the optical flow warping function applied to the previously stylised frame.

To train a style transfer video network is to find the network h parameters θ that minimise the total loss function:

$$\theta^* = \arg\min_{\theta} E_{I_t,a,x_t}[L_{total}(\mathbf{I}_t, \mathbf{a}, \mathbf{x}_t)] \quad (5.9)$$

where $\mathbf{x}_t = h_\theta(\mathbf{I}_t)$.

The network architecture is the same used for still images in chapter 4. Before detailing the training procedure, a brief description of the dataset pre-processing is given.

The training datasets used were the Hollywood2 scene dataset [23] and the Microsoft Coco image dataset [21]. The latter was already used to train a still image network in chapter 4; the former is a collection of short videos depicting scenes from films.

From each video, pairs of consecutive frames with sufficient pixel-wise difference between them are extracted. In total, there are about 40.000 frame sequences in the dataset. For every frame pair obtained, the forward and backward optical flow fields are obtained with the FlowNet 2.0, and from there the binary masks are obtained by following equations 5.5 and 5.4. Three parallel datasets result from the original Hollywood2 dataset: a dataset of pairs of frames, a dataset with the corresponding optical flow fields, and a dataset with the corresponding binary masks.

To train a video style transfer network h, a pre-trained image transformation network f is necessary. In each iteration:

1. The first frame of a pair is passed through the pre-trained image transformation network, producing a stylised frame;

2. The resulting stylised frame is warped with the corresponding pre-computed optical flow, taken from the optical flow dataset;

3. The warped frame is masked with the corresponding mask taken from the mask dataset;

4. The warped and masked first frame, the corresponding binary mask and the second frame in the sequence are concatenated and fed through the video transformation network;

5. The resulting output, the stylised second frame, has its content and style representations measured against the original second frame and the source style image, respectively;

6. Temporal loss is calculated between the output of the video transformation network, the binary mask and the warped and masked first frame fed as input;

7. The weight parameters θ of the video transformation network are updated so that the total loss 5.8, a linear combination of losses computed in step 5 and step 6, is minimised.

More precisely, the algorithm is as follows:

Algorithm 2 Training a Video Transformation Network

procedure TRAINING h_θ
  $b \leftarrow$ batch size
  $a \leftarrow$ style image
  $f \leftarrow$ a pre-trained image transformation network
  for each VGG style layer l_s **do**
    $G_s^l \leftarrow$ batch of Gram matrices from layer l_s, image a
  end for
  for 60.000 iterations **do**
    $TotalLoss \leftarrow 0$
    $I_t \leftarrow$ batch of first frames from video dataset
    $f(I_t) \leftarrow$ batch of network outputs for image batch I_t
    $I_{t+1} \leftarrow$ batch of second frames from video dataset
    $h_\theta(I_{t+1}) \leftarrow$ batch of network outputs for image batch I_{t+1}
    for each VGG content layer l_c **do**
      $F_c^l \leftarrow$ batch of feature maps from layer l_c, batch I_{t+1}
      $\hat{F}_c^l \leftarrow$ batch of feature maps from layer l_c, batch $h_\theta(I_{t+1})$
      $TotalLoss \mathrel{+}= \frac{\alpha}{b}\|\hat{F}_c^l - F_c^l\|_F^2$
    end for
    for each VGG style layer l_s **do**
      $\hat{G}_s^l \leftarrow$ batch of gram matrices from layer l_s, batch $h_\theta(I_{t+1})$
      $TotalLoss \mathrel{+}= \frac{\beta}{4N_l^2M_l^2b}\|\hat{G}_s^l - G_s^l\|_F^2$
    end for
    $TotalLoss \mathrel{+}= \frac{\gamma}{D}\|m_{t-1,t} \odot (h_\theta(I_{t+1}) - T_{t-1,t}(f(I_t)))\|_F^2$
    $\theta \leftarrow$ ADAM(θ, $TotalLoss$)
  end for
end procedure

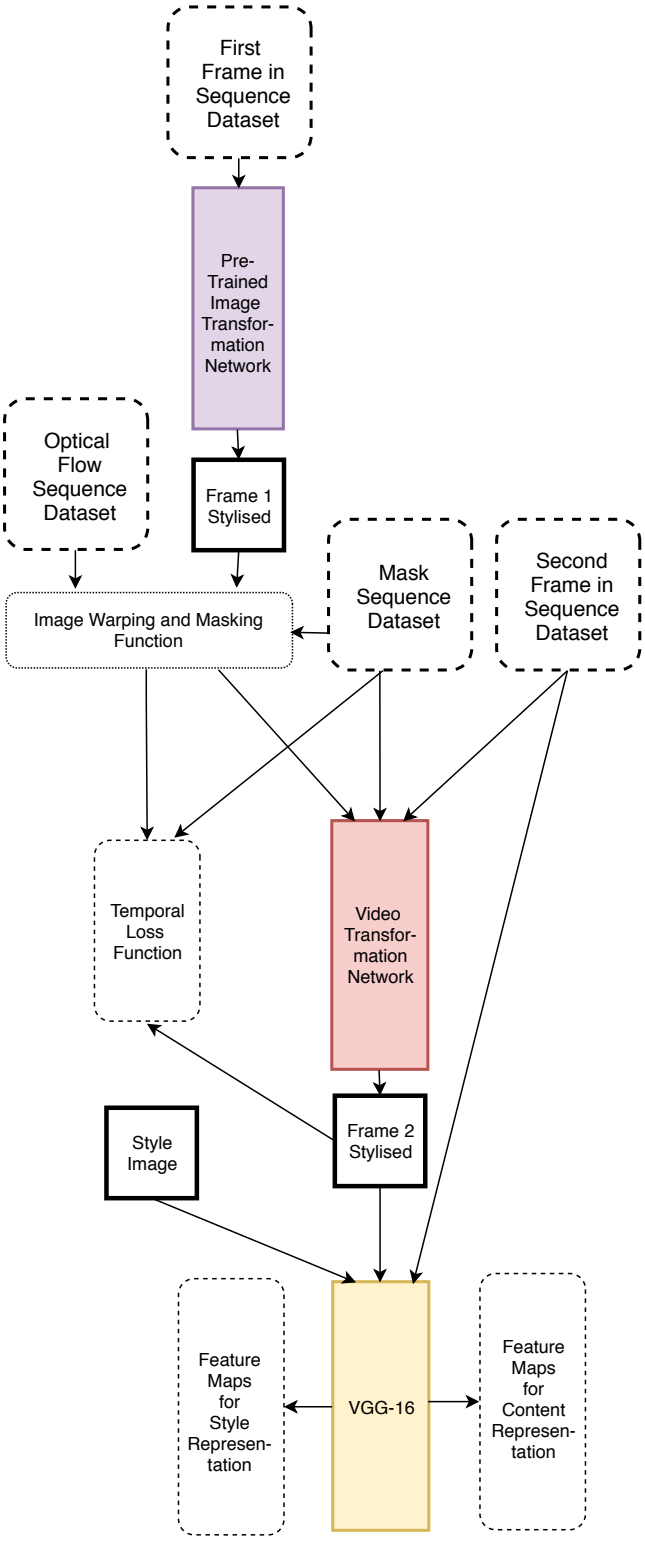

Figure 5.2: An overview of a training system for a video style transfer network.

To prevent the propagation of errors, the network is also trained to stylise images rather than just copying the previous frame. To this end, a mixed training modality [29] is used. If, during some of training iterations, the first stylised frame is an empty image, and the corresponding binary mask is set to 0 in every pixel, then the network will learn to stylise an image without prior information. In other words, the temporal consistency loss is disabled and the training loss assumes the form 3.7.

The network was trained for 60.000 iterations using ADAM optimizer with a learning rate of 0.001 and a batch size of 4. At the beginning of every training iteration, there's a 50% chance the network will receive an image extracted from the Microsoft Coco dataset without prior information, instead of a frame from the video dataset.

5.5 Convolutional LSTM Architecture

One of the main drawbacks of the method in the previous section is the dependency of optical flow computation during test time, since the network never learns to associate an input in a given timestep to inputs from preceding timesteps. To process a video, one needs to compute dense optical flow fields between frame pairs, creating a computational bottleneck that will negate some of the advantages introduced by the network training approach. While the usage of optical flow allows for a consistent appearance between adjacent frames by following pixels' motion trajectories, a network architecture based on convolutional LSTM layers will combat flickering effects where they are most prominent: in image regions with less semantic content. A convolutional LSTM layer will hold on to the pixel value present in the previous frame and prevent abrupt changes in its value from frame to frame. One of the main drawbacks from this approach will be its difficulty in correctly applying a style to frame sequences where abrupt content changes occur. A network consisting of a mixture of convolutional layers and a convolutional LSTM layer has been used to correct videos processed without temporal consistency constraints in [9].

Figure 5.3 shows the architecture of the proposed network. The only convolutional LSTM layer is at the tail end of the residual block line, since that is where the image stylisation is finalised. The training procedure for this network is the same as the one from the previous section: a hybrid loss function combining perceptual and temporal terms; and in each time step, the network learns to generate an output frame x_t that is temporally consistent with respect to x_{t-1}. The current output frame is then fed as the input at the next time step. Other specifications include: an ADAM optimiser with a learning rate of 0.001, a batch

size of 4, and 60000 training iterations over a pairs of consecutive frames from the Hollywood2 dataset [23]. The training algorithm for training an optical flow dependent video transformation network is the same for training a LSTM video transformation network.

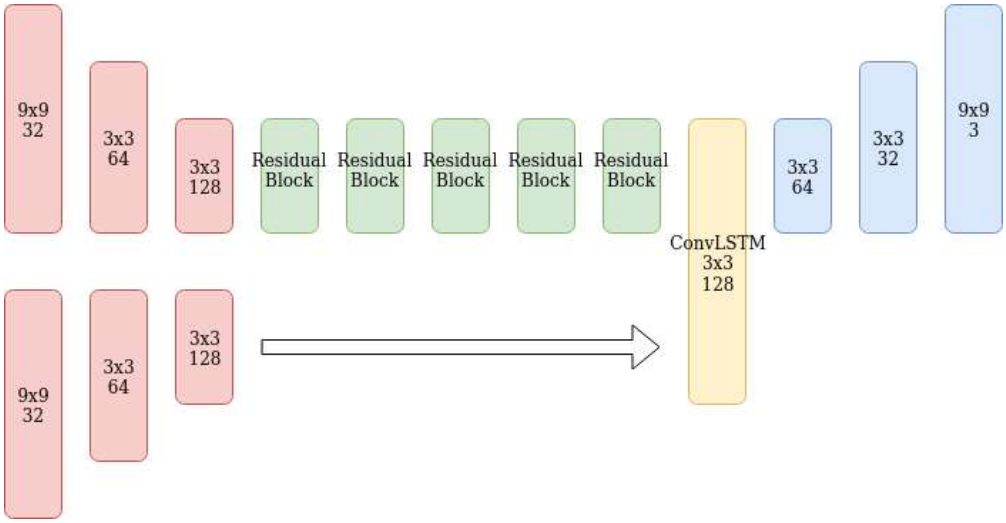

Figure 5.3: Proposed architecture based on with two streams; the upper one for the current unprocessed frame, and the bottom one for the previously stylized frame.

6 Results and Discussion

Quantitative and qualitative evaluations are conducted on the experimental results obtained from the four different implementations described on the previous three chapters.

6.1 Classical and Fast Style Transfer

In chapter 4, it was stated that a network based style transfer produces results with a similar perceptual quality to the classical optimisation algorithm. To assess the truth of this statement, four content and four style images are chosen and compared side-by-side. The optimisation algorithm uses the VGG-19 network; the fast style transfer algorithm the VGG-16. The set of style layers selected for the classic algorithm are **conv1_2**, **conv2_2**, **conv3_3**, **conv4_3** and **conv5_3**; while the content layer is the **conv4_3**. The optimisation algorithm took 100 iterations to produce the following results.

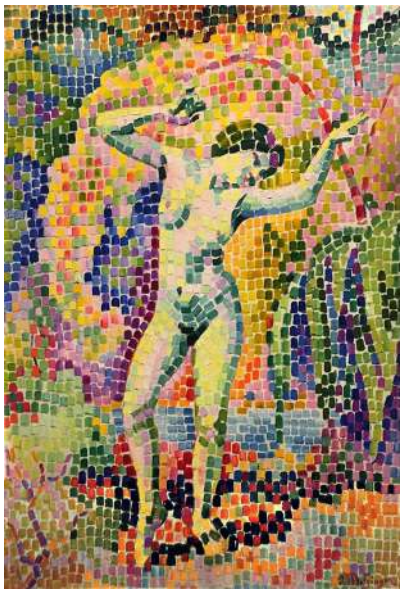

Figure 6.1: Style image.

Figure 6.2: Classic.

Figure 6.3: Fast style.

Figure 6.4: Content image.

Figure 6.5: Classic.

Figure 6.6: Fast style.

Figure 6.7: Classic.

Figure 6.8: Fast style.

Training the network on a GeForce RTX 2080 took approximately 2h20min. Passing an image through the network took approximately 200ms. Optimising a 1152×1624 image over 100 iterations took approximately 30 minutes, or 20 seconds per iteration. In conclusion, fast style transfer achieves a speed-up of 150. A quick observation of the selected images allows us to conclude that fast style transfer is actually better at preserving the original colour palettes of the source style image. However, a network with this architecture has drawbacks has well: with approximately 1.6 million parameters in total and a size of 6,3 MB.

Figure 6.9: Classic.

Figure 6.10: Fast style.

6.2 Instance Normalisation and Batch Normalisation

This section will present the main perceptual differences between an image stylised by a network trained with instance normalisation layers and the batch normalisation network implemented by Johnson et al. [18]. Both networks were trained with the same style image, and at the same resolution in order to better compare results. An open-source implementation of Johnson et al.'s article is available at `https://github.com/jcjohnson/fast-neural-style`. Results can be observed in fig.6.11. Instance normalisation layers are shown to be better in preserving image's semantic content by improving contrast between objects and maintaining a sense of depth perception in stylised images when compared to batch normalisation layers.

6.3 Video Style Transfer

A quantitative evaluation of video style transfer is conducted with the MPI Sintel dataset. The MPI Sintel dataset [2] includes 23 scenes with 20 to 50 frames of resolution 1024×436 pixels with ground truth optical flow and ground truth binary masks. In this section, results obtained with a video transformation network (**VTN**) and a LSTM video transformation network (**LSTM-VTN**) will be compared to results obtained from a image transformation network (**ITN**) using the MPI dataset. Henceforth, every image will be accompanied with

Figure 6.11: On the left, an image resulting from Johnson et al.'s architecture, with batch normalisation layers and transposed convolutional layers for upsampling. On the right, an image resulting from an adapted architecture with instance normalisation layers and nearest-neighbour upsampling layers.

the appropriate acronym identifying what network produced it.

Figure 6.12: Candy, Mosaic and Muse, style images used in this section.

6.4 Optical Flow Dependent Video Network

Firstly, an evaluation of the first solution to video style transfer introduced in chapter 5: the flow dependent video network.

As difficult as it is to see flickering effects and discontinuities by comparing frames side by side, they are impossible to miss when those same frames are seen in quick succession. Figures 6.17, 6.18 and 6.19 were the result of applying an image transformation network on a sequence of three consecutive frames. In the lower left corner of each frame, two blue squares signal areas where flickering is visible; the colors change drastically from one frame to the next. Figures 6.21, 6.22 and 6.23 were produced with a video transformation network and no longer present flickering in the lower left corner. This example demonstrates that, even when the relative position of the camera and the object remains constant during a frame

sequence, an **ITN** will produce flickering effects in the final result simply due to a varying level of light in the scene [12].

Figures 6.33, 6.34 and 6.35 show three non consecutive frames from a scene where the character lifts its arm, occluding the background scenario, and, after a while, lowers it again, disoccluding the background. A direct consequence of using a short term consistency loss to train the network is the lack of memory regarding areas temporarily occuded. From fig. 6.33 to fig. 6.35, the background wall completely changed its appearance. This can be contrasted with fig. 6.29, 6.30 and 6.31, where it can be observed that the wall didn't change its overall appearance after being disoccluded. However, it should be once again pointed out that the sequence of frames stylised with an **ITN** suffers from heavy flickering effects, even if it doesn't appear to in these images. Even with short term temporal consistency related effects, the videos stylised by a **VTN** present a smoother transition from frame to frame.

Finally, to conclude a qualitative evaluation of this **VTN**, we observe the effects of stylisation on a scene with no camera motion, where the background has little structure, using a more unstable style image: the mosaic from fig. 6.12. In fig. 6.41, 6.42 and 6.43, it should be immediately noticeable how changeable the background is from frame to frame, even though there is no discernible change in its appearance in images 6.37, 6.38 and 6.39. To facilitate visualisation, areas of interest where the instability is unmistakable were signalled with a blue square.

A quantitative comparison between the results obtained from a **VTN** versus from an **ITN** can be achieved by computing the difference between a given frame and the preceding, warped frame (see eq. 5.7). Table 6.1 presents the average values over a scene in the MPI Sintel dataset, for the three scenes referenced in this section, stylised with the candy and mosaic style images.

Scene	Candy **ITN**	Candy **VTN**	Mosaic **ITN**	Mosaic **VTN**
Alley 1	55.23	41.93	73.30	53.08
Alley 2	60.55	44.47	74.75	55.05
Ambush 7	62.58	29.49	72.34	40.26

Table 6.1: Average temporal consistency over entire scenes.

Training the network on a GeForce RTX 2080 took approximately 9 hours for a total of 60000 iterations.

Figure 6.13: Frame 7

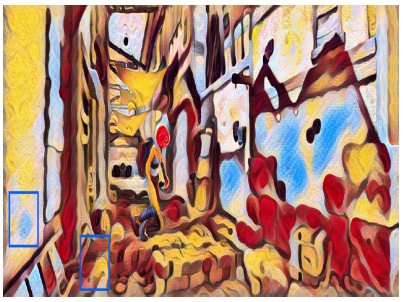

Figure 6.17: Frame 7

Figure 6.21: Frame 7

Figure 6.14: Frame 8

Figure 6.18: Frame 8

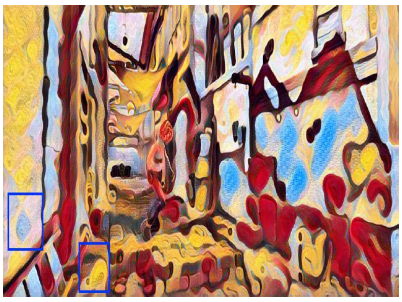

Figure 6.22: Frame 8

Figure 6.15: Frame 9

Figure 6.16: *Alley 2*

Figure 6.19: Frame 9

Figure 6.20: *Alley 2*, **ITN**

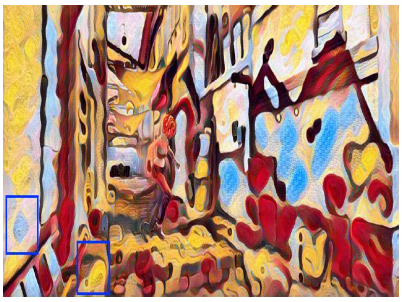

Figure 6.23: Frame 9

Figure 6.24: *Alley 2*, **VTN**

Figure 6.25: Frame 9

Figure 6.26: Frame 20

Figure 6.27: Frame 25

Figure 6.28: *Alley 1*

Figure 6.29: Frame 9

Figure 6.30: Frame 20

Figure 6.31: Frame 25

Figure 6.32: *Alley 1*, **ITN**

Figure 6.33: Frame 9

Figure 6.34: Frame 20

Figure 6.35: Frame 25

Figure 6.36: *Alley 1*, **VTN**

Figure 6.37: Frame 2

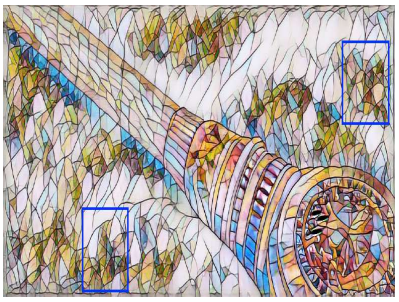

Figure 6.41: Frame 2

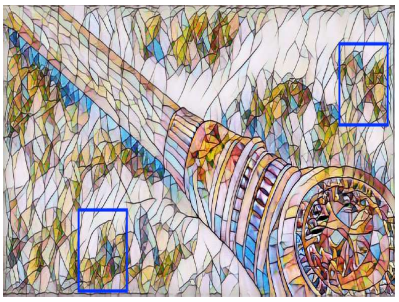

Figure 6.45: Frame 2

Figure 6.38: Frame 3

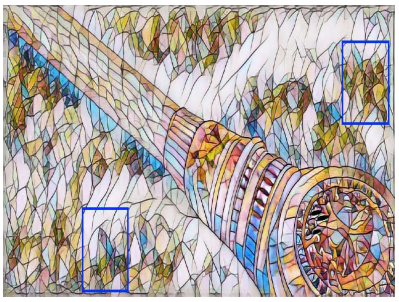

Figure 6.42: Frame 3

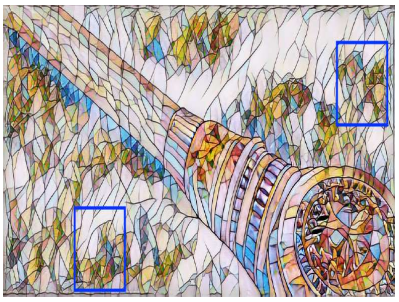

Figure 6.46: Frame 3

Figure 6.39: Frame 4

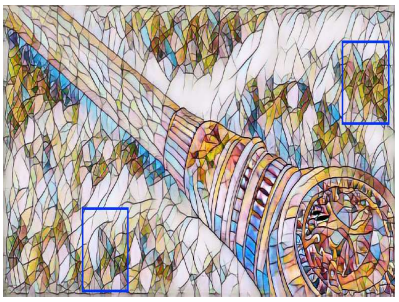

Figure 6.43: Frame 4

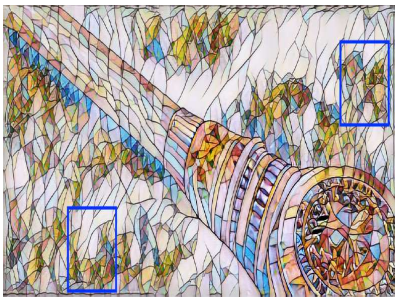

Figure 6.47: Frame 4

Figure 6.40: *Ambush 7*

Figure 6.44: *Ambush 7*, **ITN**

Figure 6.48: *Ambush 7*, **VTN**

6.5 LSTM Video Network

Figure 6.49 shows a side by side pair of adjacent frames taken from the *alley 1* sequence stylised by an **ITN** and by a **LSTM-VTN**. It is patently clear the LSTM approach achieves a higher degree of short term temporal consistency when compared to independent, frame to frame stylisation on an **ITN**; in other words, a LSTM network was trained to preserve as much as possible from the previously stylised frame and, in return, short term discrepancies caused by the appearance and disappearance of new stylisation artefacts have been averted. However, over long frame sequences, error propagation has a tendency to accumulate. This results in stylisation artefacts moving in the scenery while not being constrained by its general semantic content disposition. We can observe this in Figure 6.50.

To evaluate in a more impartial manner how difficult it is to balance semantic information from an unprocessed frame and its stylised antecedent when training a network with convolutional LSTM layers, we measured Structural Similarity Indexes [37] and the mean squared error for pairs of adjacent frames over four whole sequences: the original, unprocessed one; and three sequences processed by each of the three networks (**ITN**, **VTN**, **LSTM-VTN**). The results for the *alley 1* and the *ambush 7* sequences can be observed in Figures 6.52 and 6.51. A **LSTM-VTN** discards relevant semantic information, maintaining perceptual constancy over the entire sequence of frames.

Optical flow dependency at test time for video stylisation networks remains one of the hardest bottlenecks to overcome. Convolutional LSTM layers have been successfully used to codify temporal and spatial relations in image sequences, and even to ameliorate consistency in processed videos. This section presented an analysis of how it would impact a video stylisation network if we used these layers while training on video frame sequences. Overall, we conclude that in its present usage, convolutional LSTM layers don't provide a valid substitution for optical flow calculations during test time.

Training the network on a GeForce RTX 2080 took approximately 12 hours for a total of 60000 iterations.

Figure 6.49: A qualitative comparison of a pair of consecutive frames stylized by two different networks; the first pair by a **LSTM-VTN**, the last pair by a **ITN**. Red rectangle borders point to areas of interest.

Figure 6.50: Disconnection between overall semantic image content and stylization effects over a long frame sequence in *LSTM*. The top image was processed with the 2^{nd} frame in the *alley 1* sequence; the bottom image with the 47^{th} frame in the same sequence.

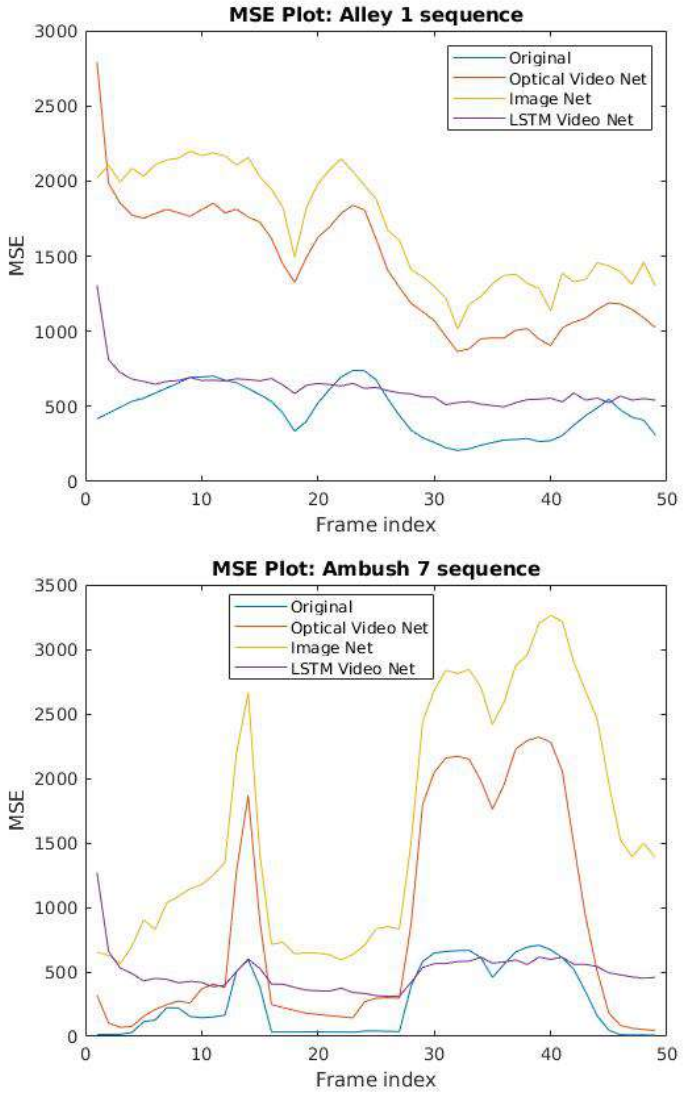

Figure 6.51: Mean squared error calculated in adjacent frame pairs over two whole sequences stylized by 3 different networks. Original refers to the original, unprocessed frame sequence; Optical Video Net to **VTN**; Image net to **ITN**; LSTM Video Net to **LSTM-VTN**.

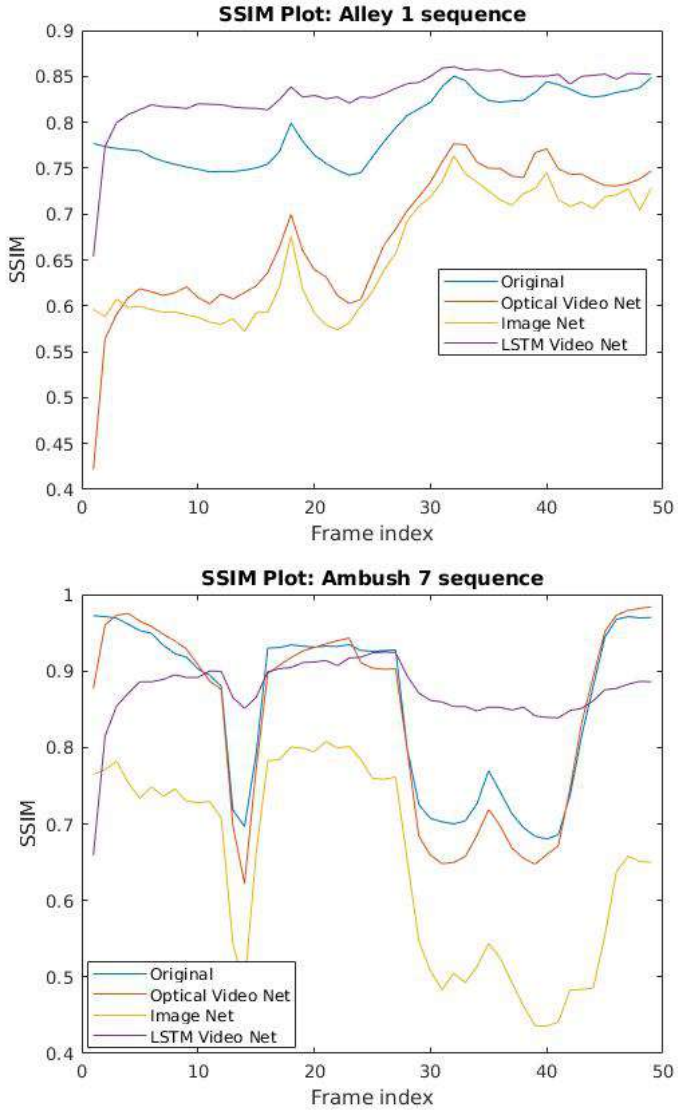

Figure 6.52: Structural Similarity Index calculated in adjacent frame pairs over two whole sequences stylized by 3 different networks. Original refers to the original, unprocessed frame sequence; Optical Video Net to **VTN**; Image net to **ITN**; LSTM Video Net to **LSTM-VTN**.

7 Conclusions

This dissertation describes the work of adapting a generative feed-forward network, trained to transform an image, into a network trained to consistently process a sequence of video frames into a temporally coherent output. This resulted in the development of two versions: the first one guides a network during training and test time using computer vision algorithms and is inspired by the training procedure described by Ruder et al. [29]; the second one relies on the network's architecture to achieve a spatial and temporal understanding of its inputs without dependence of optical flow during test time.

Considering several experiments performed with the video transformation network dependent on optical flow during test time, it became clear it learns to associate the warped previous frame and the binary mask with the temporal loss function, hence why it became dependent on it during test time. There are precedents for this behaviour of generative networks in literature: a mask taken as input serving as a spatial guide for the network. It is also interesting how a problem of temporal consistency in frame sequences is interpreted as a purely spatial problem by the network when it is given as input the mask and previously stylised frame. Even if the LSTM-video style network produced less than desirable results, modifying the network's architecture in order to learn temporal relations between inputs, and not just spatial relations, should remain a research topic.

The work of this dissertation was submitted to the Eurographics Conference, and is attached at the end of this document. Feedback is still pending.

7.1 Future Work

Some of the possible improvements to build on the baseline implementations (image transformation network, video transformation network and LSTM-video transformation network) developed for this thesis include:

- Investigate the classical style loss function, following the thread left by Julesz conjec-

ture. Whether or not higher order statistics from VGG feature maps improve style transfer by accurately representing images with arbitrary patterns should be a point of research. If style transfer can be interpreted as a domain adaptation problem, like it has been suggested in recent literature, and the style loss as a probabilistic distance metric - the Maximum Mean Discrepancy - then it should be possible to take the next logical step and train a generative network with the Maximum Mean Discrepancy with a characteristic kernel to lessen the distance between all statistical moments and not just the first two.

- Another area of interest would be to investigate a solution to the apparent incompatibility of training a network with a mixture of convolutional layers and LSTM convolutional layers using the hybrid, spatial and temporal loss function.

- Virtual reality applications usually require a light-weight application, so the need arises for a smaller stylisation network to fit the physical constraints of mobile devices.

- While it provides fast solution to the problem of estimating the optical flow between two images, FlowNet 2.0 will sometimes output blatantly wrong results. More accurate alternatives, like DeepFlow, run on the CPU and take much longer to produce a result, creating a bottleneck when the stylisation of a sequence of frames is dependent on optical flow. Even though overcoming this bottleneck is not directly related to the task of style transfer, it would certainly improve the implementation of video style transfer.

- Expand video style transfer to generate videos with dynamic texture patterns that exhibit stationarity in the temporal domain.

8 Bibliography

[1] A. G. Anderson, C. P. Berg, D. P. Mossing, and B. A. Olshausen. Deepmovie: Using optical flow and deep neural networks to stylize movies. *arXiv preprintarXiv:1605.08153*, 2016.

[2] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, October 2012.

[3] D. Chen, J. Liao, L. Yuan, N. Yu, and G. Hua. Coherent online video style transfer. *ICCV*, page 1114–1123, 2017.

[4] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *International Conference on Learning Representations*, 2017.

[5] David Foster. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play.* O'Reilly Media, 2019.

[6] L.A. Gatys, A.S. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv*, Aug 2015.

[7] L.A. Gatys, A.S. Ecker, and M. Bethge. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems 28*, May 2015.

[8] L.A. Gatys, A.S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2016.

[9] W. Gong, W. Wang, W. Li, and S. Tang. Temporal consistency based method for blind video deblurring. *22nd International Conference on Pattern Recognition*, 2014.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[11] S. Gross and M. Wilber. Training and investigating residual nets. 2016.

[12] A. Gupta, J. Johnson, A. Alahi, and L. Fei-Fei. Characterizing and improving stability in neural style transfer. In *ICCV*, page 4087–4096, 2017.

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

[14] H. Huang, H. Wang, W. Luo, L. Ma, W. Jiang, X. Zhu, Z. Li, and W. Liu. Real-time neural style transfer for videos. *CVPR*, page 7044–7052, 2017.

[15] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[16] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, pages 448–456, 2015.

[17] M. Jaderberg and and A. Zisserman K. Simonyan. Spatial transformer networks. In *NIPS*, 2015.

[18] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *European Conference on Computer Vision*, page 694–711, Jan. 2016.

[19] B. Julesz. Visual pattern discrimination. *IRE transactions on Information Theory*, 8(2):84–92, 1962.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[21] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C.L. Zitnick. Microsoft coco: Common objects in context. *Computer Vision–ECCV*, page 740–755, 2014.

[22] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *ArXiv:1409.1556*, April 2014.

[23] Marcin Marszałek, Ivan Laptev, and Cordelia Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

[24] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[25] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, pages 49–70, October 2000.

[26] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ArXiv e-prints*, Nov. 2015.

[27] E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neu-ral texture synthesis and style transfer using histogram losses. *ArXiv e-prints*, Jan. 2017.

[28] M. Ruder, A. Dosovitskiy, and T. Brox. Artistic style transfer for videos. *Pattern Recognition*, page 26–36, 2016.

[29] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, Apr 2018.

[30] E. P. Simoncelli and W. T. Freeman. The steerable pyramid: A flexible architecture for multi-scale derivative computation. *Proceedings, International Conference on Image Processing*, 1995.

[31] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ArXiv:1409.1556*, April 2014.

[32] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV)*, Lecture Notes in Computer Science. Springer, Sept. 2010.

[33] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *International Conference on Machine Learning*, page 1349–1357, 2016.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition, page 6924–6932, 2017.

[35] L.-Y. Wei, S. Lefebvre, V. Kwatra, and G. Turk. State of the art in example-based texture synthesis. *Eurographics State of the Art Report, EG-STAR*, pages 93–117, 2009.

[36] S. Xingjian, Z. Chen, H. Wang, D.Y. Yeung, W.K. Wong, and W.C. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS*, 2015.

[37] H. Sheikh Z. Wang, A. Bovik and E. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, page 600–612, 2004.

Appendix A

Eurographics Conference 2020

Breaking Optical Flow Dependency in Fast Video Style Transfer

SUBMISSION ID: short1059

Abstract

Style transfer has become a very fertile field of research ever since neural networks introduced a paradigm shift into what was once a computer vision problem. Initially employing an iterative optimisation based approach, neural style transfer suffered a metamorphosis with the addition of offline training generative networks. As faster stylization became a possibility, so did real-time video style transfer. However, fast style transfer is inherently unstable and has shown limitations in producing temporally consistent results. In this paper we propose a recursive architecture to stabilise consecutive pairs of stylized video frames using a convolutional LSTM layer positioned in the latent space's tail end on a encoder-decoder architecture in order to encode spatial and temporal dependencies. This way we train a generative network to learn how to maintain perceptual similarities between pairs of processed images without resorting to optical flow during test time.

CCS Concepts

• ***Computing methodologies*** → *Neural networks;*

1. Introduction

Rendering a content image in the style of a painting is a area of research that received a second wind in both academia and industry when convolutional neural networks pre-trained on purely discriminatory tasks where found to convey information relevant to both style [GEB15] and content [MV15] representations in its feature activations. Gatys *et al.* [GEB16] devised an iterative optimisation procedure to gradually change an image into a visually appealing result. Since this is a time consuming process, Johnson *et al.* [JAFF16] propose a new generative network architecture to be trained on the same loss function as [GEB16]. At test time, one network forward pass will transform the target image in a stylized output of comparable quality to [GEB16] but orders of magnitude faster.

Transferring style to every frame of a given video is one of the many natural extensions of image style transfer. Both online optimisation and offline training based approaches have produced visually appealing results by circumventing the main obstacle in video style transfer: maintaining temporal consistency between adjacent frames and preventing flickering effects from appearing. The work presented on this paper is focused on proposing and validating an alternative fast, recurrent style transfer algorithm for videos. While several solutions have been proposed to tackle the problem of fast video style transfer, optical flow dependence at test time remains the most difficult bottleneck to overcome. We propose an adaptation to an existing generative network and an accompanying training protocol, and evaluate our results in qualitative terms against a state of the art method dependent on optical flow at test time.

2. Related Work

2.1. Image Style Transfer

In this section we offer a succinct overlook of Gatys' *et al.* [GEB16] style transfer algorithm. An iterative image synthesis task, it outputs a new image I_n that matches VGG [SZ14] content and style feature representations to those from a pair of content and style images (I_c and I_s respectively). We obtain I_n by finding

$$\arg\min_{I_n} \beta L_c(I_n, I_c) + \alpha L_s(I_n, I_s)$$

where α and β are loss function hyperparameters keeping balance between style and content; L_c and L_s content reconstruction loss and style reconstruction loss.

Using the VGG-19 [SZ14] pretrained on ImageNet dataset, F^l being a feature map of size $H^l \times W^l \times C^l$ (height, width, channel) from layer l, the content reconstruction loss is defined as the squared-error loss between feature representations,

$$L_{content}(I_c, I_n) = \sum_{l \in C} \frac{1}{C_l H_l W_l} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \quad (1)$$

where F^l_{ij} is the activation of the i^{th} filter at position j in layer l belonging to a set of content layers C. Style reconstruction loss is defined as:

$$L_{style}(I_s, I_n) = \sum_{l \in S} \frac{1}{C_l H_l W_l} \sum_{i,j} (G^l_{ij} - G^l_{ij})^2 \quad (2)$$

where the Gram matrix for the l^{th} layer is:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \quad (3)$$

where G^l_{ij} is the inner product between the vectorised feature map i and j in layer l.

2.2. Occlusion and Motion Boundary Detection

In Sundaram *et al.* [SBK10], a pixel in frame f_i can be tracked to the next frame using the optical flow field $\mathbf{w} := (u,v)^T$ as

$$(x_{i+1}, y_{i+1})^T = (x_i, y_i)^T + (u_t(x_i, y_i), v_t(x_i, y_i))^T \quad (4)$$

An occlusion is detected by means of a forward and backward consistency check on optical flow. Unless caused by a flow estimation error, in a image point that doesn't suffer occlusion between two frames, the backward flow vector should have the inverse direction of the forward flow vector. However, to account for margins of error, Sundaram *et al.* [SBK10] established that disocclusions occur where the following inequality is true:

$$|\tilde{w} + \hat{w}|^2 \geq 0.01(|\tilde{w}|^2 + |\hat{w}|^2) + 0.5 \quad (5)$$

where $w = (u,v)$ is the optical flow in the forward direction (from frame f_i to frame f_{i+1}), $\hat{w} = (\hat{u}, \hat{v})$ is the optical flow in the backward direction (from frame f_{i+1} to frame f_i), and $\tilde{w}(x,y) = w((x,y) + \hat{w}(x,y))$ is the forward flow warped to the second image. On the other hand, motion boundaries are detected with the following inequation:

$$|\nabla(u)|^2 + |\nabla(v)|^2 > 0.01|\hat{w}|^2 + 0.002 \quad (6)$$

2.3. Online Optimisation Based Video Style Transfer

Ruder *et al.* [RDB16] improved Gatys *et al.* [GEB16] seminal neural style transfer algorithm by introducing a temporal constraint 7 in its loss function to prevent discrepancies between adjacent frames. Optical flow is used to guide the temporal constraint along point trajectories, allowing for regions in the new frame also present in the previous one to keep a consistent appearance; and for occluded areas and motion boundaries to be identified [SBK10] and be excluded from it. The formula for temporal consistency loss between frames f_i and f_{i+1} can be seen below,

$$L_{temporal}(I_n^{i+1}, \omega, c) = \frac{1}{D}\sum_{k=1}^{D} c(I_n^{i+1} - \omega)^2 \quad (7)$$

where c is a pixel-wise binary mask distinguishing between image pixels seen in both frames and those belonging to occluded regions, disoccluded regions and motion boundaries; $\omega_i^{i+1}(I_n^i)$ is a warping function using optical flow estimated between both frames; D is the size of the image.

2.4. Offline Network Based Video Style Transfer

While fast style transfer improves the algorithm's efficiency, it loses flexibility on more than one front: a new model has to be trained for each new style image and for each new pair of content and style hyperparameters α and β. Furthermore, fast style transfer is unstable and is bound to create wildly differing results between pairs of images with almost imperceptible changes in semantic content between them, resulting in flickering effects when applied independently to each frame. Gupta *et al.* [GJAFF17] have mathematically demonstrated why fast style transfer is an unstable process, and that its instability is proportional to the style Gram matrix trace. Since it is dependent on the input style, there is a demand for a style blind, universal solution. Gupta *et al.* [GJAFF17] proposed solution is to fine tune a pre-trained style transfer feedforward network on a video dataset consisting of images, ground truth optical flow and ground truth occlusion masks, in order to maintain temporal consistency between adjacent frames.

Their model follows the architecture set by [DSK17] and receives as input at each training step the previous frame, stylized, and the current frame, unprocessed. Finally, they perform a pixel-wise comparison between the stylized input frame and the stylized output frame, the former being backward warped using optical flow 7.

In another solution to fast video style transfer, a new generative network architecture and a new recurrent training procedure involving the same temporal loss function as before, Ruder *et al.* [RDB18] achieved results comparable to [GJAFF17]. Their method captures movement in a more precise manner than [GJAFF17], since it uses optical flow during test time, and retains a higher degree of temporal consistency, especially in image regions with little semantic structure.

2.5. Broad Spectrum Video Temporal Consistency

Using neural networks for video processing produces inconsistent results in areas other than style transfer. Tackling this issue, Lai *et al.* [WSLY18] train a network to minimise short term and long term temporal losses, and perceptual losses, reducing the flickering problem in processed videos while maintaining perceptual similarity between adjacent frames. Their algorithm does not need optical flow computation at test time due to a convolutional LSTM layer, positioned in the architecture's latent space in order to capture spatial and temporal correlations existing in frame pairs.

3. Architecture

Our major contribution is a recurrent generative network architecture based on [DSK17], with two streams - one for the previous stylized frame and the other for the current pre-processed frame

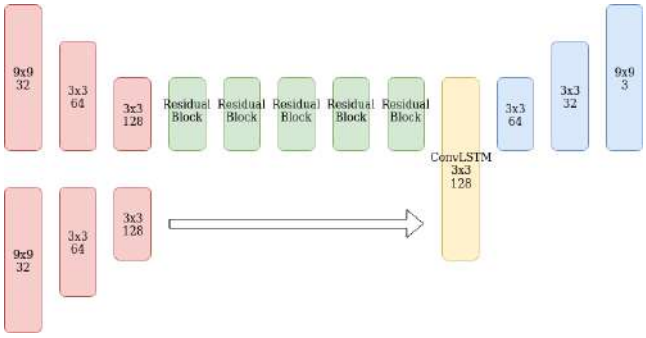

Figure 1: *Proposed architecture based on with two streams; the upper one for the current unprocessed frame, and the bottom one for the previously stylized frame.*

- consisting in two pairs of downsampling layers, one for each stream, followed by five residual blocks for the latter stream, a convolutional LSTM layer to merge both streams, and finally two nearest-neighbour upsampling layers, each followed by a convolutional layer. Instance normalization and ReLu non-linearities are present after every convolutional layer. After the last residual block, our convolutional LSTM layer serves to model spatial and temporal dependencies in pairs of input images. A schematic for our architecture can be seen in Figure 1.

3.1. Training and Experimental Results

The dataset used for video training was the Hollywood2 video dataset [MS09], every scene partitioned into five consecutive frames. For every pair of consecutive frames, we compute their forward and backward optical flow, and the corresponding binary consistency masks following 5 and 6. For a sequence of five frames from a given scene, we run every image through a pre-trained image stylization network and form image tuples consisting of a stylized frame i and an unprocessed, original frame $i+1$. Our network learns to stylize our unprocessed frame in a temporal consistent manner, to avoid undesirable effects from appearing and further deviations from the previously stylized frame.

Our video network was trained with a loss function resulting from the linear combination of 1, 3 and 7: $\lambda_{content}L_{content} + \lambda_{style}L_{style} + \lambda_{temporal}L_{temporal}$, where $\lambda_{content}$, λ_{style} and $\lambda_{temporal}$ are loss weights. The VGG-16 layers used for the content loss where *relu3_3*, and for the style loss *relu1_2*, *relu2_2*, *relu3_3* and *relu4_3*. Other specifications include: an ADAM optimizer with a learning rate of 0.001, a batch size of 4, and 18000 training iterations.

Henceforth, we shall refer to a stylization network trained on still images, such as [DSK17], by *still net*; to a stylization network trained on image sequences presenting optical flow dependency at test time, such as [RDB18], by *optical video net*; and to a stylization network trained on image sequences and presenting a LSTM layer in its architecture, by *LSTM video net*. All proceding evaluations and tests were conducted on the MPI Sintel dataset [BWSB12].

Figure 2 shows a side by side pair of adjacent frames taken from the *alley 1* sequence [BWSB12] stylized by a *still net* and by a *LSTM video net*. It is patently clear our approach achieves a higher degree of short term temporal consistency when compared to independent, frame to frame stylization on a *still net*; in other words, our network was trained to preserve as much as possible from the previously stylized frame and, in return, short term discrepancies caused by the appearance and disappearance of new stylization artefacts have been averted. However, over long frame sequences, error propagation has a tendency to accumulate. This results in stylization artefacts moving in the scenery while not being constrained by its general semantic content disposition. We can observe this in Figure 3.

To evaluate in a more impartial manner how difficult it is to balance semantic information from an unprocessed frame and its stylized antecedent when training a network with convolutional LSTM layers, we measured Structural Similarity Indexes [ZWS04] and the mean squared error for pairs of adjacent frames over four whole

Figure 2: *A qualitative comparison of a pair of consecutive frames stylized by two different networks; the first pair by a LSTM video net, the last pair by a still net. Red rectangle borders point to areas of interest.*

sequences: the original, unprocessed one; and three sequences processed by each of the three networks (*still net*, *optical video net*, *LSTM video net*). The results for the *alley 1* and the *ambush 7* sequences can be observed in Figures 5 and 4. A *LSTM video net* discards relevant semantic information, maintaining perceptual constancy over the entire sequence of frames.

3.2. Conclusions

Optical flow dependency at test time for video stylization networks remains one of the hardest bottlenecks to overcome. Convolutional LSTM layers have been sucessfully used to codify temporal and spatial relations in image sequences, and even to ameliorate consistency in processed videos. In this paper we analysed how it would impact a video stylization network if we used these layers while training on video frame sequences. Overall, we conclude that in its present usage, convolutional LSTM layers don't provide a valid substitution for optical flow calculations during test time.

Figure 3: *Disconnection between overall semantic image content and stylization effects over a long frame sequence in LSTM. The top image was processed with the 2^{nd} frame in the alley 1 sequence; the bottom image with the 47^{th} frame in the same sequence.*

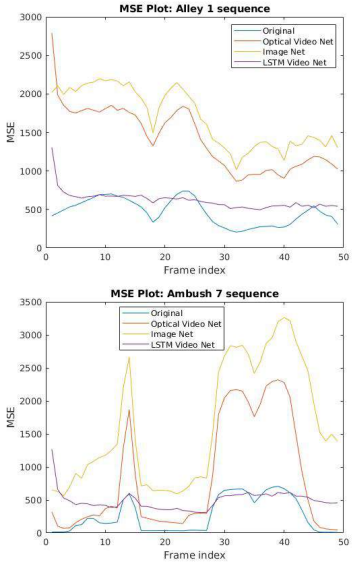

Figure 4: *Mean squared error calculated in adjacent frame pairs over two whole sequences stylized by 3 different networks.*

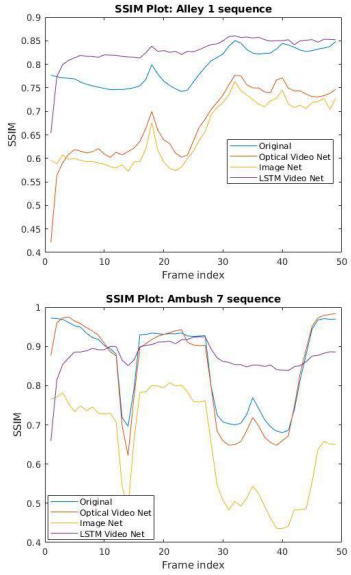

Figure 5: *Structural Similarity Index calculated in adjacent frame pairs over two whole sequences stylized by 3 different networks.*

References

[BWSB12] BUTLER D. J., WULFF J., STANLEY G. B., BLACK M. J.: A naturalistic open source movie for optical flow evaluation. *European Conf. on Computer Vision (ECCV)* (2012), 611–625. 3

[DSK17] DUMOULIN V., SHLENS J., KUDLUR M.: A learned representation for artistic style. 2, 3

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: Texture synthesis using convolutional neural networks. *Advances in Neural Information Processing Systems* (2015), 262–270. 1

[GEB16] GATYS L. A., ECKER A. S., BETHGE M.: Image style transfer using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 2414–2423. 1, 2

[GJAFF17] GUPTA A., JOHNSON J., ALAHI A., FEI-FEI L.: Characterising and improving stability in neural style transfer. *Proceedings of the IEEE International Conference on Computer Vision* (2017), 4067–4076. 2

[JAFF16] J.JOHNSON, ALAHI A., FEI-FEI L.: Perceptual losses for real-time style transfer and super-resolution. *European Conference on Computer Vision* (2016), 694–711. 1

[MS09] M MARSZAEK I. L., SCHMID C.: Actions in context. *IEEE Conference on Computer Vision & Pattern Recognition* (2009). 3

[MV15] MAHENDRAN A., VEDALDI A.: Understanding deep image representations by inverting them. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), 5188–5196. 1

[RDB16] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos. *German Conference on Pattern Recognition* (2016), 26–36. 2

[RDB18] RUDER M., DOSOVITSKIY A., BROX T.: Artistic style transfer for videos and spherical images. *International Journal of Computer Vision* (2018). 2, 3

[SBK10] SUNDARAM N., BROX T., KEUTZER K.: Dense point trajectories by gpu-accelerated large displacement optical flow. *ECCV* (2010), 438–451. 2

[SZ14] SIMONYAN K., ZISSERMAN A.: Very deep convolutional networks for large-scale image recognition. *ArXiv:1409.1556* (April 2014). 1

[WSLY18] W.-S. LAI J.-B. HUANG O. W. E. S. E. Y., YANG M.-H.: Learning blind video temporal consistency. *Computer Vision – ECCV 2018 Lecture Notes in Computer Science* (2018), 179–195. 2

[ZWS04] Z. WANG A. BOVIK H. S., SIMONCELLI E.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* (2004), 600–612. 3

Appendix B

Figures used in chapter 6

Figure B.1: Frame 7

Figure B.2: Frame 8

Figure B.3: Frame 9

Figure B.4: *Alley 2*

Figure B.5: Frame 7

Figure B.6: Frame 8

Figure B.7: Frame 9

Figure B.8: *Alley 2*, **ITN**

Figure B.9: Frame 7

Figure B.10: Frame 8

Figure B.11: Frame 9

Figure B.12: *Alley 2*, **VTN**

Figure B.13: Frame 9

Figure B.14: Frame 20

Figure B.15: Frame 25

Figure B.16: *Alley 1*

Figure B.17: Frame 9

Figure B.18: Frame 20

Figure B.19: Frame 25

Figure B.20: *Alley 1*, **ITN**

Figure B.21: Frame 9

Figure B.22: Frame 20

Figure B.23: Frame 25

Figure B.24: *Alley 1*, **VTN**

Figure B.25: Frame 2

Figure B.26: Frame 3

Figure B.27: Frame 4

Figure B.28: *Ambush 7*

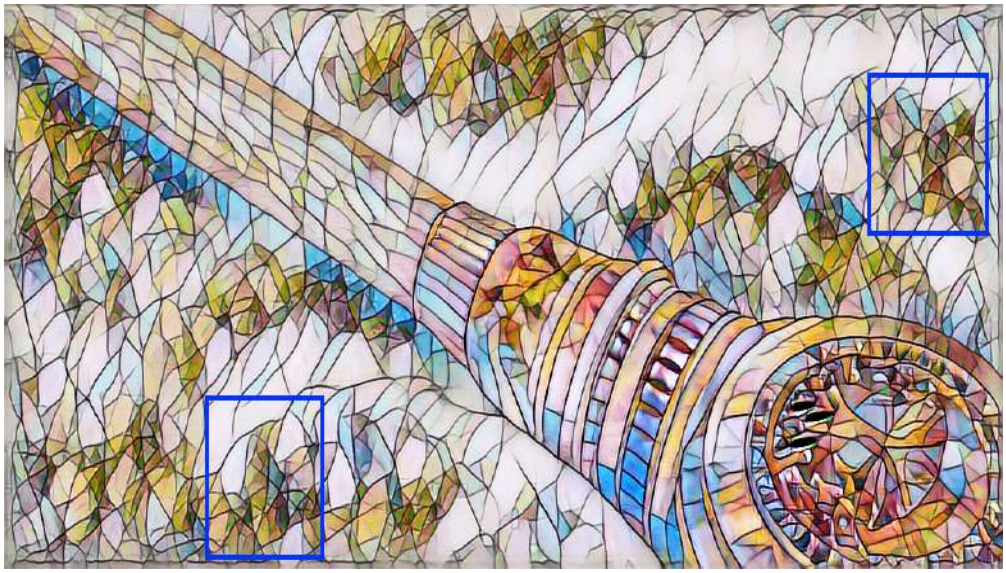

Figure B.29: Frame 2

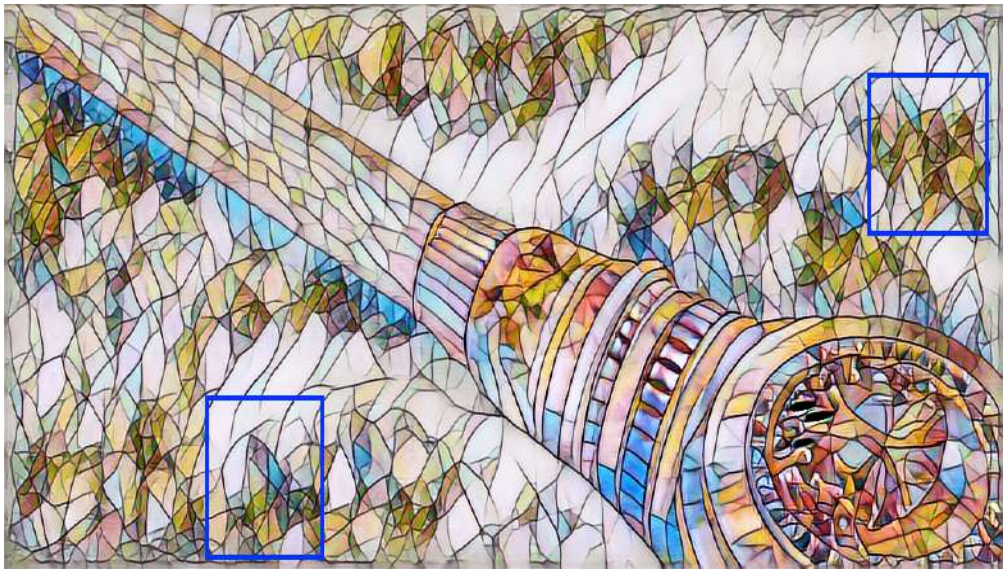

Figure B.30: Frame 3

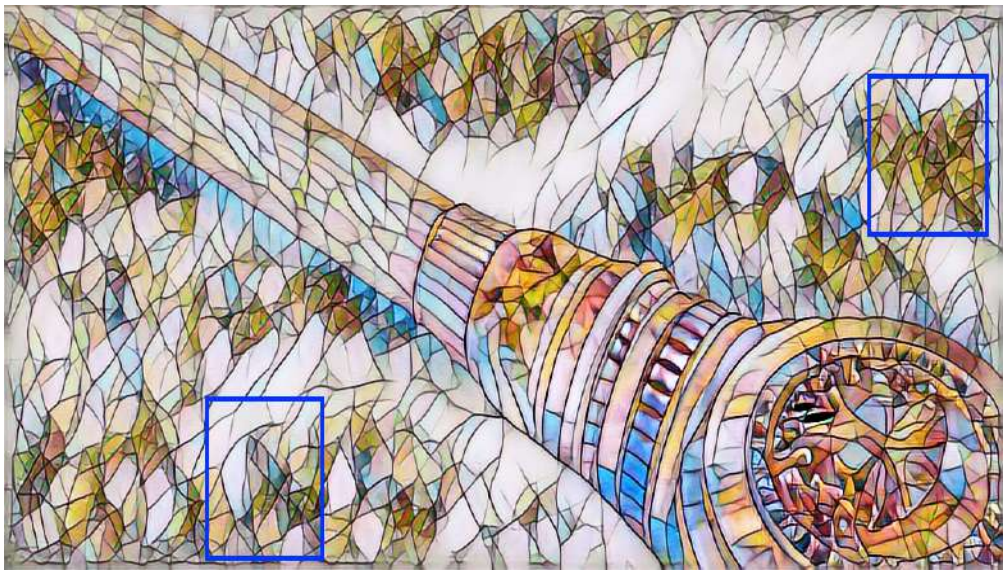

Figure B.31: Frame 4

Figure B.32: *Ambush 7*, **ITN**

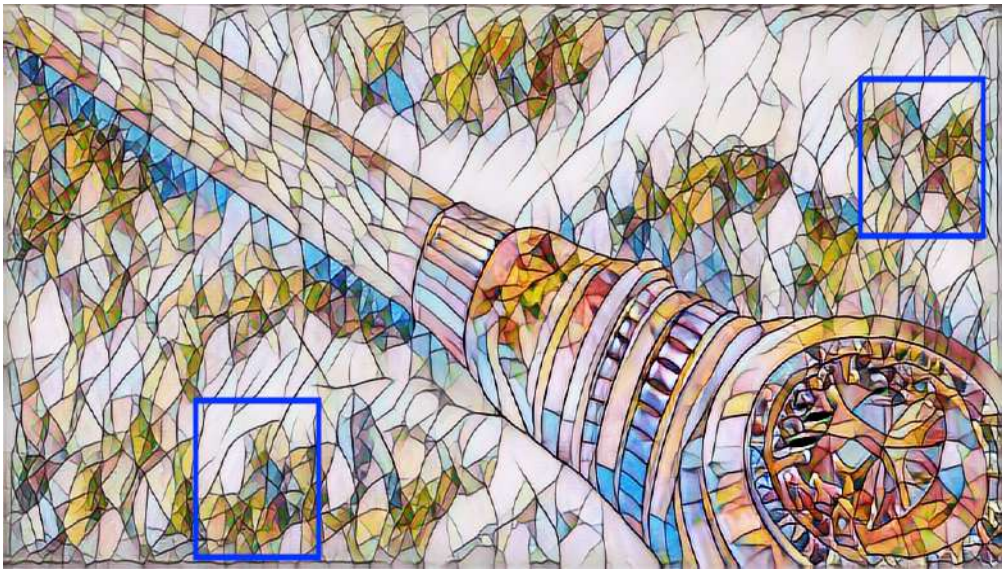

Figure B.33: Frame 2

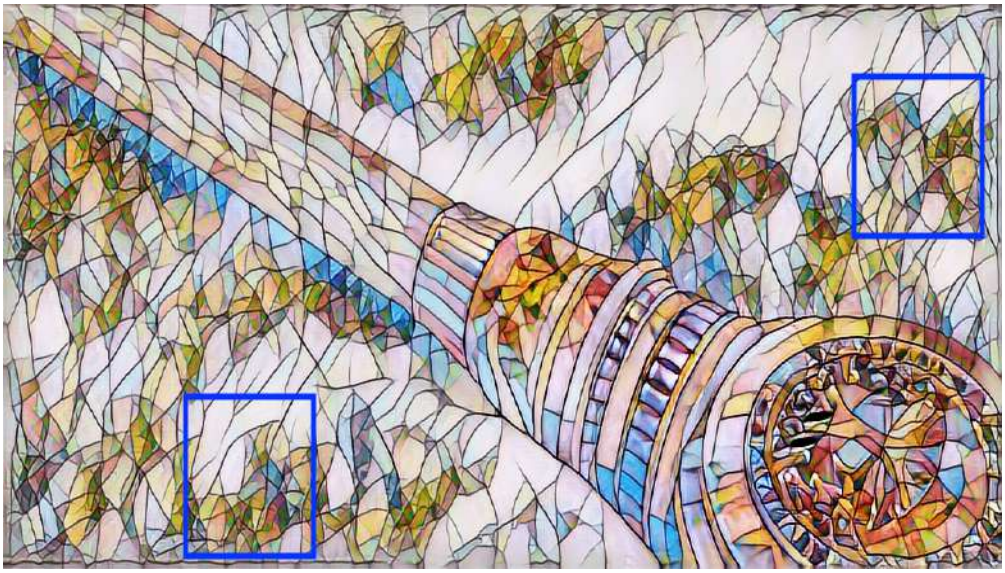

Figure B.34: Frame 3

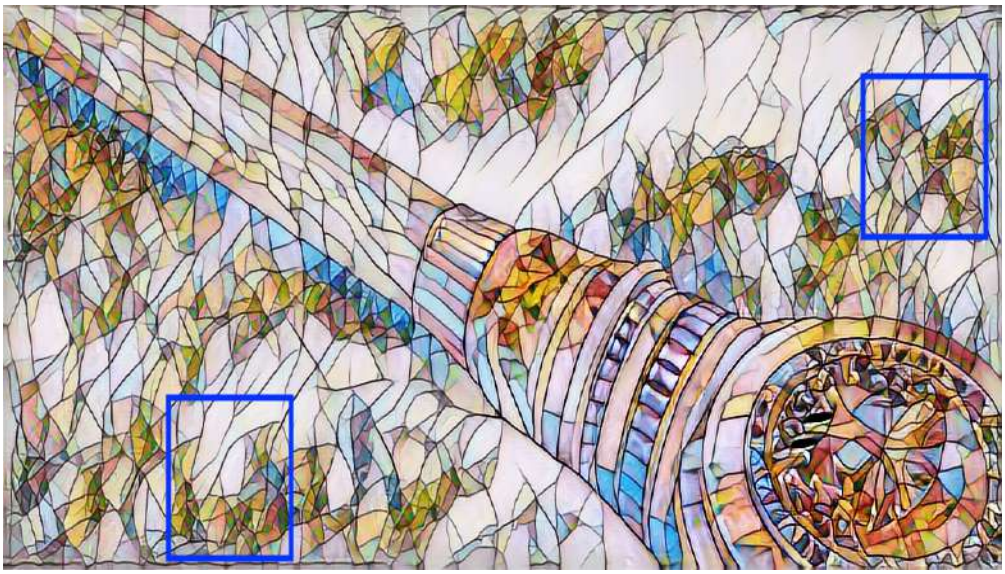

Figure B.35: Frame 4

Figure B.36: *Ambush 7*, **VTN**