Geometric Style Transfer: Re-enactment of Artistic Poses

Author:
Shu Zhao

Supervisor:
Almila Akdag

Second supervisor:
Ronald Poppe

August 16, 2021

Utrecht University

Contents

1 Introduction **2**

1.1 Motivation 2

1.2 Challenges 3

1.3 Research question 4

2 Related work **5**

2.1 CNN 5

2.2 Autoencoder 14

2.3 RNN 17

2.4 GAN 20

2.5 HPE 25

3 Datasets **31**

3.1 Natural poses 31

3.2 Artistic poses 32

3.3 Annotations 35

3.4 Statistical analysis 41

3.5 Datasets overview 45

4 Methodologies **47**

4.1 Contours 47

4.2 CycleGAN 54

4.3 Warping 63

5 Results **64**

5.1 CycleGAN 64

5.2 Warping 70

5.3 Results overview 72

6 Conclusion **75**

7 Discussion **77**

A PoC of vanilla CNN **78**

B Dendrogram **80**

Chapter 1: Introduction

1.1 Motivation

In recent years, artistic works such as photographs, paintings, and music have been created and represented dominantly in digital format. The digitized art is easy-to-spread via Internet, long-standing against time, and provides a means to analyze them computationally. For the masterpieces created before the digital age, numerous efforts have been invested by libraries and museums in digitization of their large art historical collections. Replica project [66] employed new techniques to scan existing archive of images to standard digital format, which can be indexed and used for information retrieval. Though digitized paintings lack the odour of pigment, or the touch of bumps of coloured blobs, it allows remote access to a wide range of audiences, e.g., tourists, museum-goers, students and art historians, around the world in cultural heritage. Beyond easy access, by tapping digitization, the past art can be re-enacted and brought back to our everyday life, for us to interact with. These implementations include AR (Augmented Reality) applications, such as AR tour guide [75] for tourists, AR Van Gogh [42] for museum visitors, or AR map at school to enable children to create narratives for their surroundings [17]. Moreover, computers can assist art historians in discovery of implicit links between digital artworks, which are otherwise buried in a multitude of art databases. Inspired by these technical advances, it leads us to wonder whether it would be possible to re-enact a past era computationally by means of Artificial Intelligence (AI). Sitting on a treasure trove of digital assets, the re-enactment can be a new linkage to be found [37], a new interpretation to be established [34], or a new way of interaction to be created. Thus, we can relive the past.

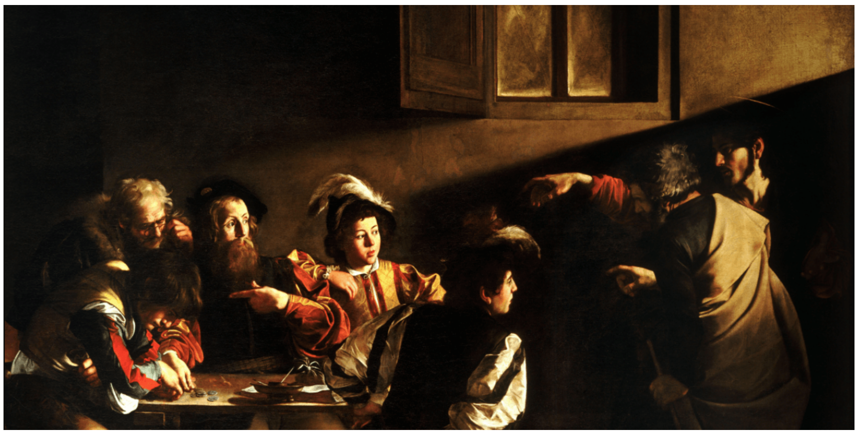


Figure 1.1: The Calling of Saint Matthew (1599–1600), Caravaggio.

Take the paintings of the Baroque period as an example, it encompasses a wide variety of styles, which are characterized by great drama, rich color, and intense light and dark shadows. One of its most influential contributors is Caravaggio. Caravaggio's paintings mainly consist of multiple characters staged against a backdrop of mellow lighting as shown in Figure 1.1, and they are full of narratives, which are expressed and perceived in the composition of poses.

The poses are informative to reflect a wide range of emotions, e.g., happiness, sadness, anger, fear, etc., and to embed societal or gender differences [54]. Moreover, the poses can convey intense emotions, which cannot be discriminated by facial expressions only [5]. Naturally, in artworks, the poses are used to express emotions as well; the portrayal of emotions via poses of a human body was first documented by one of the first cultural and art historians, Aby Warburg with his concept of Pathosformel [34]. The term pathosformel comes from the combination of Pathos (emotion) and formel (a formula). Warburg traces the pathosformeln back to ancient Greek vase paintings [49], where a narrative is illustrated in the interactions and compositional relationships between characters. The same visual elements of postures and gestures are used in recurrent narratives. For art historians, the composition transfer can be identified through the similarities between the postures of individual characters [37], as a painter often incorporated a stylistic element by copying human poses of another artist. To analyze the evolution of poses, it is possible to track how a theme transforms and spreads throughout time and space.

Thus, to explore the narratives conveyed by the poses in the classical and modern paintings piques our curiosity. In this thesis, we want firstly to analyze, what defines the geometric shapes of an artistic pose in a computational manner. Secondly, we want to build a geometry-aware style transfer, which can transform a natural pose in a photograph to an artistic pose. By style transfer, the goal is to realize the re-enactment of a daily activity in the style of past artworks.

1.2 Challenges

There are basically two tasks to be addressed in this thesis: (1) to extract the geometric shapes of artistic poses in the paintings (2) based on the extracted shapes, to build geometry-aware style transfer which is able to transform a natural pose to an artistic pose. The basis for both tasks are the well-annotated datasets in which two sets of images are needed: (1) one annotated dataset for the natural poses (2) one annotated dataset for the artistic poses. what we mean with the annotation of natural and artistic poses is the following: the joints and segments of the human bodies are either manually marked and masked, or correctly inferred by the trained pose estimation models.

There is already an abundance of natural-pose datasets. COCO dataset is one of those, which includes a wide variety of common activities of human activities, such as bow, climb, crouch, kneel, drink, eat, etc. The state-of-the-art pose estimation models such as OpenPose [9] and DensePose [62] are trained with the COCO dataset, in which the natural poses have already been manually annotated with respect to the joints and body segments. However, that the COCO dataset lacks the training data of the unusual poses could be a potential drawback.

There are no artistic-pose datasets that are manually annotated. Here, the challenge is the trade-off between accuracy and effort. The highest accuracy results from the manual annotation of the poses from selected paintings, but this needs the most effort. The second option is to rely on the inferred outcome of OpenPose and DensePose, but the question is whether the inferred joints and body segments are accurate enough to base pose analysis upon. In [49], OpenPose is applied on the Greek vase paintings, and it demonstrates that OpenPose does not generalize well in artistic domain, as it is trained with natural poses. Thus, it is critical to test the inference accuracy of OpenPose and DensePose when they are applied to paintings in order to further decide whether manual annotation is needed or we can rely on the inferred results to implement generic methods for pose analysis. Once the the annotated datasets for both natural and artistic poses are available, the first task to analyze these poses can be facilitated.

the second task is to build geometry-aware style transfer. Style transfer typically requires paired datasets. One of our challenges is the lack of paired datasets of natural and artistic poses. It is almost impossible to have the same pose present in both a photograph and in a painting in the training datasets, thus the architecture of a neural network model is critical, so that unpaired datasets can be used during training. CycleGAN [81] can address this challenge, but it brings with it another challenge: since the natural and artistic poses are not paired during training, we need to find a way to accurately pair the matched locations, as the goal of style transfer is to transfer from source body segments to target body segments, and from source background to target background on a coarse-grained scale. Furthermore, on a fine-grained scale, the body segments can be paired with each other between natural and artistic poses, namely, from head to head, from torso to torso, from arms to arms, and from legs to legs. In summary, the challenge is to match the paired locations in the unpaired datasets.

Finally, as our goal is to build a geometry-aware style transfer, we need to find a way to transmit the shape-morphing signal. The shapes can be changed manually by warping. We would like to explore whether the shapes can be changed automatically during the training of a CycleGAN model. If the previous challenge focuses on “where”, then this challenge focuses on “what”. CycleGAN has been demonstrated to transfer color and texture well, but to fail in transferring shape. The test case is to transfer from cat to dog, and from apple to orange. The result is a cat-shaped dog and a orange-colored apple. Thus, we need to explore how to embed the shape signal into CycleGAN.

To summarize, the challenges come into the following aspects:

1. In the domain of artworks, the manually annotated datasets of poses are almost none, and the state-of-the-art pose estimation models such as OpenPose and DensePose do not generalize well across domains;

2. In the training of a neural-network based style transfer, the paired datasets of natural and artistic poses do not exist;

3. In terms of geometry-aware style transfer, the previous works show that the shapes cannot be automatically changed by a single model.

1.3 Research question

The research question is to build a geometry-aware style transfer, which is able to impose the geometric characteristics of an artistic pose to a natural pose. It breaks down into the following three tasks:

1. Infer the joints and body segments from natural and artistic poses;

2. Extract the computational features for natural and artistic poses;

3. Build geometry-aware style transfer, which can stylise a natural pose given an aimed artistic pose.

The first task will address challenge 1 to get the annotation of the joints and body segments for artistic poses. Combining with the second task, they form the basis to build geometry-aware style transfer. The third task is the main goal of this thesis, and it can address challenge 2 and 3 to impose the texture and geometry from an artistic pose to a natural pose in the right locations.

Chapter 2: Related work

“Big Data” has facilitated AI in many ways. First, all kinds of information, i.e., image, sound and document, can be easily digitized, efficiently transmitted and thus intermingled. Second, Internet enables quick access to such a multitude of data at any place and time. Third, digital data is much easier for computer to process than for human, as it is computer-readable rather than human-readable. The core of AI is deep learning, and “deep” comes from the structure that a typical neural network has many layers [27]. Deep learning has been therefore extensively explored and adopted in the creative process. In paintings, Magenta’s SketchRNN is a RNN-based (Recurrent Neural Network) model, which can draw abstract sketches [30]. OpenAI’s DALL·E can create imaginative illustrations based on the instructions expressed in natural language. In music, DeepBach [31] is a trained RNN to generate chorales in the style of Bach. Magenta’s MusicVAE is an autoencoder-based model that can generate MIDI-formatted (Musical Instrument Digital Interface) music with long-term structure [63]. And Magenta’s NSynth can learn from the raw audio waveform to generate new types of sounds with WaveNet autoencoder [19]. In language, OpenAI’s GPT-3 (Generative Pre-trained Transformer 3) model generates poetry and prose in a novel way. In computer vision, deep learning’s building block is based on CNN (Convolutional Neural Network). CNN has boosted the performance of a wide variety of computer vision tasks tremendously, as it can be trained automatically to learn visual patterns, e.g., low-level patterns like edges and colors, or high-level patterns like objects and structure.

In Section 2.1, we will introduce what is a typical CNN architecture and why CNN outperforms the traditional algorithms in computer vision tasks. Then we will explain how CNN can be applied in neural style transfer, as transferring style between two domains is close to our research question. Next, we will introduce how CNN can contribute to arts in the domain of artistic paintings. In Section 2.2, we will introduce the advanced form of CNN, based on the encoder-decoder design. In Section 2.3, we will introduce RNN, which is another advanced form of CNN that can capture the pixel-level spatial dependencies of an image. In Section 2.4, GAN will be explained, as GAN is mostly used to bridge two domains and generate high-quality images. In Section 2.5, the human pose estimation will be introduced, as our research question focuses on the poses extracted from the paintings.

2.1 CNN

The basic topology of deep learning is based on feedforward neural networks, also called multilayer perceptron (MLP) [27]. Each layer is also regarded as a neuron, which can be represented by a function. One layer’s output is subsequently fed into another layer as input, thus multiple layers can be chained together in a network. For example, given an input x and three layers denoted by function $f^{(1)}$, $f^{(2)}$ and $f^{(3)}$, the neural network can be represented as $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. $f^{(1)}$ is considered the input layer, $f^{(3)}$ is the output layer, and any layers in-between, such as $f^{(2)}$, are called the hidden layers. For a function f, generically it can be written as $f(x, \theta)$, where x is the input, and θ denotes all the parameters of f. For a neural network, it can be represented by two models respectively: a linear model and a non-linear model. Mostly, a non-linear model is used, as it can tackle much more complicated tasks, e.g., when the decision boundary is not simply linear. The linear model is often written as Equation 2.1, and the non-linear model is written as Equation 2.2, with θ breaking down into w and b, where w means weight, b stands for bias. What is important in a non-linear neural network is that it consists of two parts: (1) $wx + b$, which is a linear transformation, and (2) a non-linear function, e.g. *tahn*, *sigmoid* or ReLU (Rectified Linear

Unit), etc., which is also known as activation.

$$f(x, \theta) = wx + b \tag{2.1}$$

$$f(x, \theta) = tahn(wx + b) \tag{2.2}$$

CNNs are a special kind of neural network for processing data that has a grid-like topology [27]. A typical CNN architecture is shown in Figure 2.1. For CNNs, there are two basic operations involved: (1) convolution, and (2) subsampling (downsampling or pooling). To explain, convolution maps a patch of multiple values to a single value, from one layer to the next layer. This mapping is calculated by dot product of an image with a filter window. The filters are also known as kernels or masks. The result of convolution is a feature map. By applying different filters, it will result in different feature maps. The feature maps can be further filtered by activation as an option. A convolution operation without activation can be represented by Equation 2.1, whereas one with activation can be denoted as Equation 2.2. After convolution, pooling further filters the noises out by choosing a single value out of a pooled window. The selection mechanism can be either to choose a maximum value, or to choose an average value. The outcome of pooling is a sparse feature map, in which the spatial coordinates of the original image are lost, but the hierarchies of the image are preserved. Thus, through the combination of convolution and pooling, the image representations are abstracted and captured in the hierarchical and low-dimensional feature maps. What is important in CNN is that the filters and hierarchies are not manually engineered, but learned during the training phase, which are stored as the network's parameters θ. In the shallow and lower layers, the low-level features are captured, such as edges or contours. In the deep and higher layers, the high-level features are preserved, such as the shapes of objects or the overall structure. These high-level features are also known as the semantic features, as they are semantically more abstract.

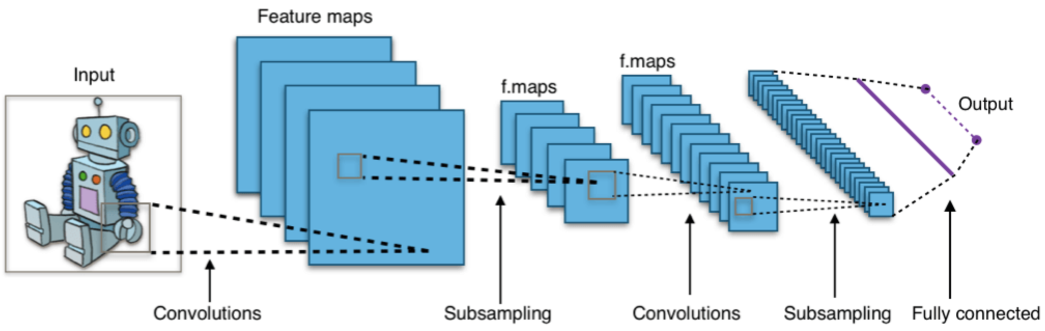


Figure 2.1: Typical CNN architecture.

In order to have a better comparison as to why CNN has reshaped the design of algorithms for the traditional computer vision tasks, in the following section, Canny Edge Detector and HOG will be introduced.

2.1.1 Filter and hierarchy

Canny Edge Detector has been applied in various computer vision systems to detect edges by any input color photographs, which was developed by John Canny in 1986. It is a multi-stage algorithm based on manually-engineered filters. In the first stage, a 5×5 Gaussian filter is used in the convolution to blur the image in order to remove the noise, as edge detector is inclined to render noise as edges as well. In the second stage, the smoothed image is further convolved with one 3×3 Sobel horizontal kernel and one 3×3 Sobel vertical kernel to get the horizontal and vertical derivatives of all the pixels in the image respectively, which are represented as G_x and G_y. The Gaussian filter and Sobel kernels are written and visualized in Figure 2.2.

Next, the edge gradient is calculated by Equation 2.3, which captures the magnitude of the contrasting intensity values for every pixel in the image, and the direction of the edge gradient is

$$\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

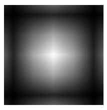
(a) Gaussian filter

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

(b) Sobel horizontal kernel

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

(c) Sobel vertical kernel



(d) Visualization of Gaussian filter

(e) Visualization of Sobel horizontal kernel

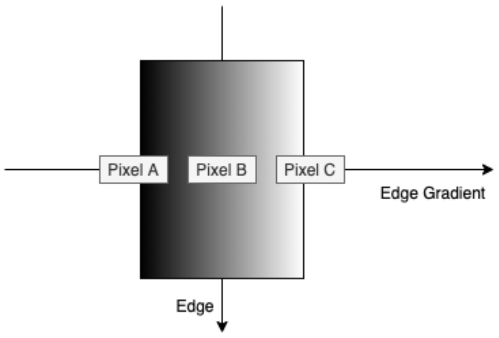
(f) Visualization of Sobel vertical kernel

Figure 2.2: Gaussian and Sobel filters to detect the edges.

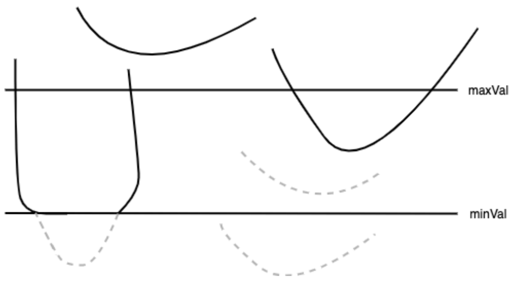
calculated by Equation 2.4. The gradient direction is always normal to edges. Till this stage, the rough edges can be already estimated by the calculated gradient.

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.3}$$

$$\theta = tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{2.4}$$



(a) Non-Maximum Suppression: Pixel A has the local maximum magnitude, whereas Pixel B and C don't. Therefore, Pixel A will be kept, and Pixel B and C will be removed.



(b) Hysteresis Thresholding: Black solid lines are kept, whereas dotted gray lines are pruned away. Because black solid lines are either above $maxVal$ or connected to the pixels above $maxVal$, whereas dotted gray lines are either below $minVal$ or not connected to the pixels above $maxVal$.

Figure 2.3: The third and fourth stages of Canny Edge Detector.

To further trim the edges, non-maximum suppression and hysteresis thresholding can be applied separately in the third and fourth stage. In non-maximum suppression, as shown in Figure 2.3a, every pixel is checked with its previous and next pixel along the gradient direction whether it has the maximum magnitude or not. If it is a local maximum in its neighbourhood, it will be kept, otherwise this pixel will be removed. This process will result in thinner edges. In hysteresis thresholding, as shown in Figure 2.3b, a maximum and a minimum threshold of magnitude are specified as $maxVal$ and $minVal$. If the magnitude of a pixel is above $maxVal$, it counts as a "sure-edge", which will be kept, and if it is below $minVal$, it is categorized as a "non-endge", which will be removed. If a pixel lands in between $maxVal$ and $minVal$, then if it is connected to a "sure-edge", it will be kept, otherwise it will be pruned away. After these four stages, the final edges can be displayed as outcome. By applying the Canny Edge Detector to Caravaggio's "The Calling of Saint Matthew" in Figure 1.1, the edge image is shown in Figure 2.4.

Based on edge detection, object recognition can be further implemented, as every object can be bounded by its edges. By detecting the objects' edges, the objects' contours can be depicted, thus the taxonomy of these objects can be categorized, such that a person and a dog have distinguished

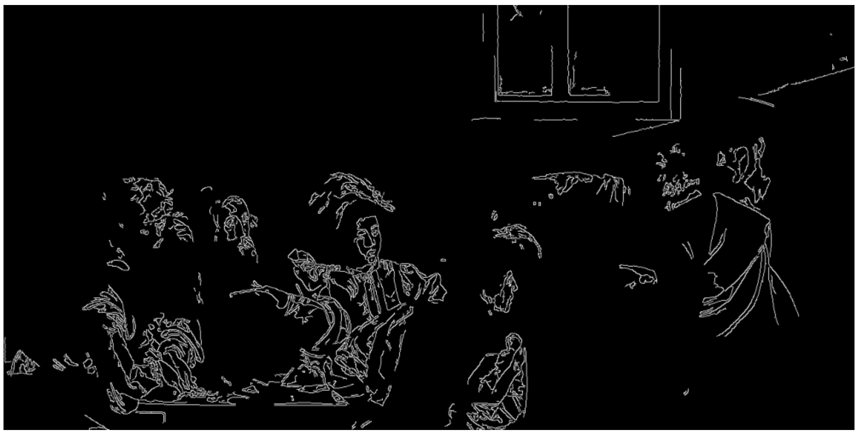


Figure 2.4: The edges of "The Calling of Saint Matthew" calculated by the Canny Edge Detector.

outlines. One of the use cases is for pedestrian detection. But one of the disadvantages to detect an object based on edges is that edge is a low-level image descriptor, which is not invariant to scale, rotation or change of viewpoint.

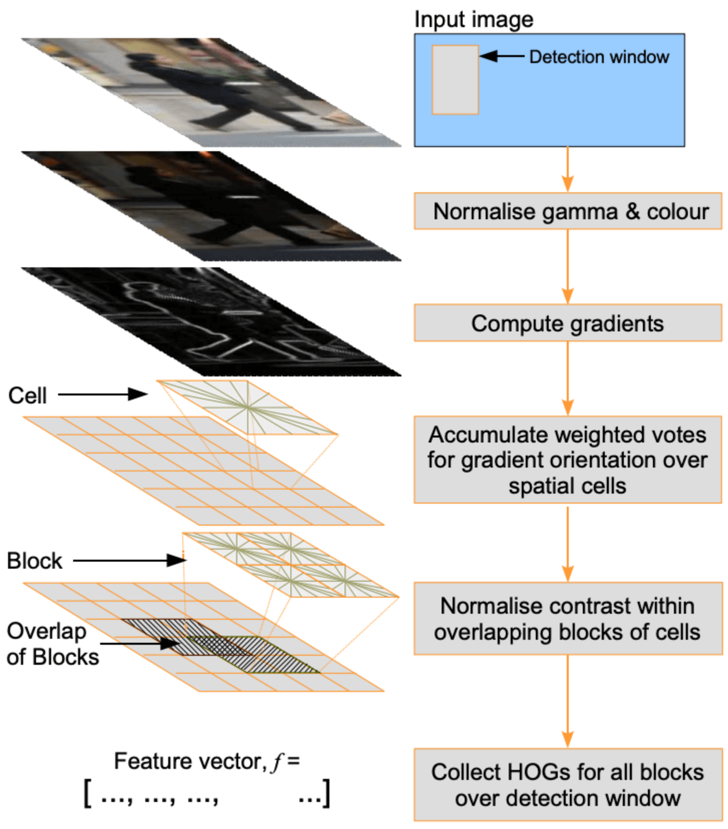

Input image
Detection window
Normalise gamma & colour
Compute gradients
Cell
Accumulate weighted votes for gradient orientation over spatial cells
Block
Overlap of Blocks
Normalise contrast within overlapping blocks of cells
Feature vector, f =
[..., ..., ..., ...]
Collect HOGs for all blocks over detection window

Figure 2.5: Process of calculating the HOG descriptor [14].

A high-level image descriptor needs to be formulated in order to capture the semantic understanding of an object, so that a person can be recognized regardless of perspectives or poses, such as near or far, standing or squatting. HOG (Histograms of Oriented Gradients) was thus implemented for human detection [14], which is invariant to scale. The HOG descriptor of an unknown object is first calculated and stored as vector A. One way to use A is to compare it with another HOG descriptor of a human image, which is pre-calculated as standard reference and stored as vector B, to get its similarity score. The similarity of these two vectors is measured by cosine similarity in Equation 2.5, where n is the dimension of HOG vectors A and B. If the score is above a threshold, then this object is categorized as human. Another way to use A is to feed it into a linear SVM (Support Vector Machine) model to further classify it as person or non-person. The process of calculating the HOG descriptor is shown in Figure 2.5.

$$similarity(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.5}$$

Similar to Canny Edge Detector, HOG uses Gaussian filter and Sobel mask in the first and second step to get the edge gradient. The difference is that HOG will divide the input image into a fixed number of cells, based on which blocks will be further constructed to contain a fixed number of cells. For each cell, a histogram of directions of the edge gradient will be calculated, with bins representing different gradient orientations weighted by their corresponding magnitudes. For each block, the histograms of all its cells will be concatenated to form a vector representative of this block. For the input image, every block's representative vector will be further concatenated to constitute a final feature vector, namely a HOG descriptor of the image. As the number of cells and blocks are fixed, no matter how large or small the input image is scaled, it will result in the same HOG descriptor. For example, if an input image contains n blocks, and each block has m cells, and each cell has a histogram of k bins, then the dimension of the feature vector is $n \times m \times k$. The architecture of HOG bares visual similarity to CNN, as it builds upon low-level features to further induce high-level features hierarchically, from cells to blocks, in order to have a whole view of the image. Compared to the low-level features which are susceptible to nuisance factors, such as scale, the high-level features can capture the semantic meaning of the object better, as it can grasp the underlying statistical distribution of the input image, which is invariant to a specific individual case. Different from edge detector, HOG is applied based on the bounding box of a detected object, and then it can further determine what kind of object it is. In Figure 2.6, HOG is applied to the full image in order to have a comparative view as to how it describes figures differently from edge detector in Figure 2.4.

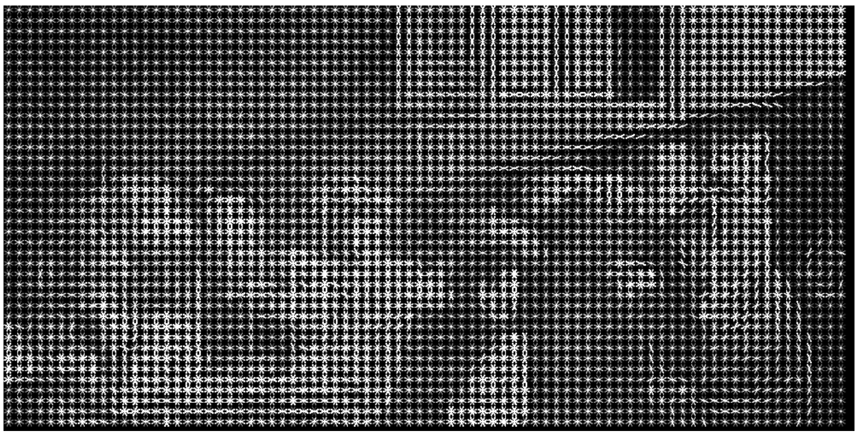


Figure 2.6: Visualization of the HOG descriptor of "The Calling of Saint Matthew". Each cell is represented by a histogram binned by 8 directions in total, and the brighter the direction, the more weight it possesses due to the magnitude of pixels contributing to this direction.

Compared to Canny Edge Detector and HOG, CNN has the advantages that the filters, e.g., Gaussian filter or Sobel kernel, are not manually hand-crafted but learned by the objective set by a given task. If the task is designed to classify human and non-human, the filters will learn to capture the common features of human and also the distinguishing features of human from non-human, and edge might not be the only factor to distinguish human from non-human anymore. There might be various factors, such as texture, color, etc. which are inherent in the image, but not obvious to algorithm designers. Moreover, the hierarchies, e.g., cells and blocks, are not manually specified as well. By applying the convolution and pooling operations iteratively, the layers will get deeper, while the feature maps will get smaller. But the spatial relationship between pixels are well preserved, thus the hierarchies can be implicitly learned during this process. In Appendix A, we will train a vanilla CNN model to illustrate that CNN can automatically learn the filters and hierarchies, given a specific task. Moreover, CNN generally has better performance than the models trained based on the hand-crafted features, such as HOG. Thus, CNN can be said to have tremendously boosted the performance of the conventional tasks in computer vision.

2.1.2 Neural style transfer

Style transfer is an application which simply tap the hierarchical features learnt by CNN to achieve the goal to transfer an arbitrary style from a painting to an image, mostly a photograph. Neural style transfer [21] is the basic version of this approach and has inspired many variants [38] [33]. The major difference is that [21] is optimization-based, and each step of optimization needs a forward and backward pass through the network, which is computationally slow, whereas [38] and [33] are both model-based and run in real-time for an input image to be stylized as output in the feed-forward transformation network. The transformation network is basically an encoder-and-decoder based CNN architecture, so these model-based methods also belong to the image-to-image translation problem, as introduced in Section 2.4.1.

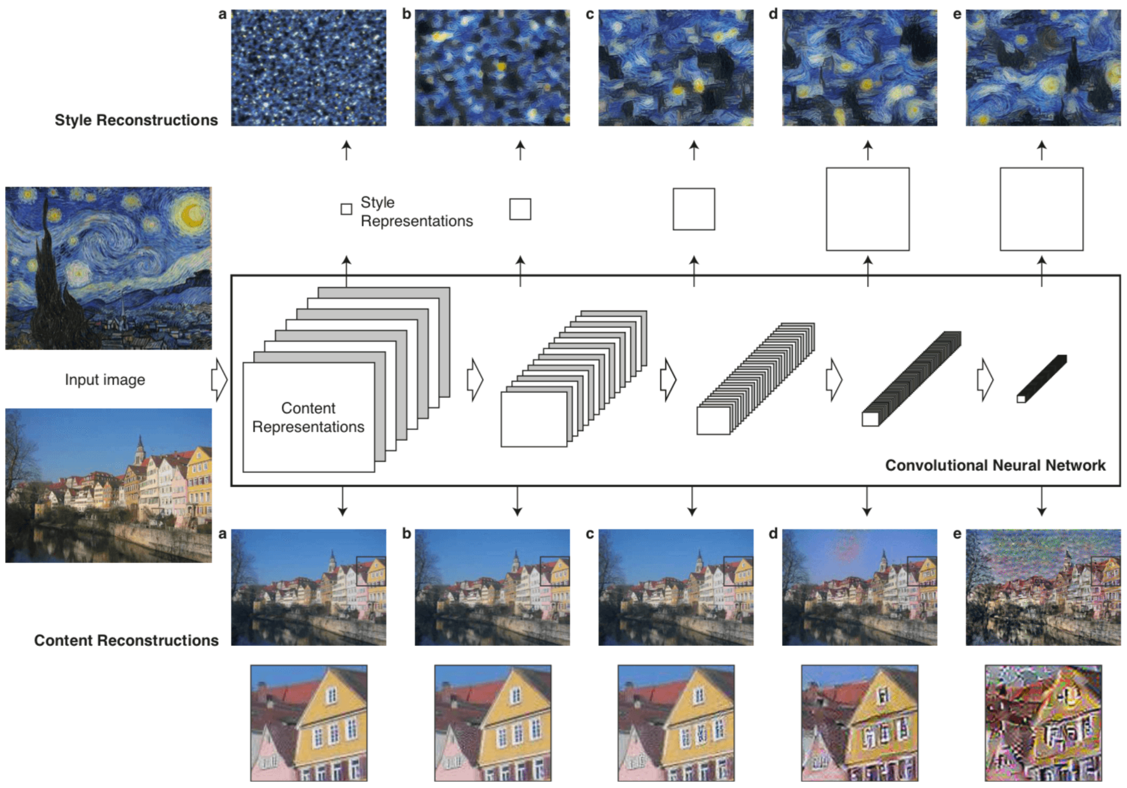


Figure 2.7: The architecture of neural style transfer [21]. A style loss is computed within the first multiple layers, whereas a content loss is calculated just using the last layer. It can be seen that in the last layer, the pixel-level details are lost, but the form of the house is preserved.

The neural style transfer requires three images as input: one is the image of the desired content, the second is the image of the reference style, the third is a random white noise. The loss function is thus designed to minimize the content loss between the white noise and the content image, and in the meanwhile, to minimize the style loss between the white noise and the reference style. The design of the neural style transfer is based on the observations that CNN can learn the hierarchical features of an image, where in the lower layers, the lower-level styles, such as brush strokes, color or texture, can be captured, whereas in the higher layers, the semantic contents, such as objects or spatial structure, can be learnt. This architecture is shown in Figure 2.7. Neural style transfer can be considered as a baseline of our goal: transfer the Caravaggio style to a video-captured frame. As it only utilizes the hierarchical features learnt by CNN, but it cannot produce realistic paintings with style transferred to the right patches of the image.

Inspired by [21], [38] defines a perceptual loss function based on the features extracted from various layers in a pre-trained VGG-16 network, as various layers capture various semantic information. Similar to neural style transfer, as shown in Figure 2.8, the perceptual loss consists of two part: (1) A feature reconstruction loss $\boldsymbol{\ell}_{feat}$, which is the content loss between the original image x: $x = y_c$ and the output from the image transform net $\hat{y}$. (2) A style reconstruction loss $\boldsymbol{\ell}_{style}$, which is the style loss between the style target y_s and $\hat{y}$. The style loss is denoted as the difference of the Gram matrices between the target style image and the inferred image. Since the Gram matrix calculates the correlation among all activations of a single layer, it is capable of capturing the features that tend to be activated together. Thus Gram matrix can be used to describe the learnt style at each layer. Both neural style transfer [21] and perceptual-loss based model [38] use Gram matrix to represent style features.

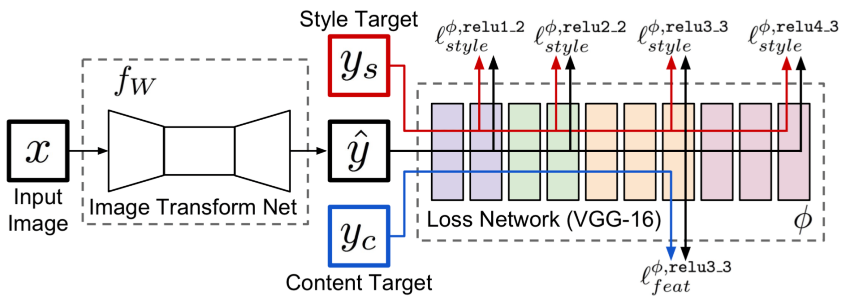

Style Target
$\ell^{\phi,\text{relu1\_2}}_{style}$
$\ell^{\phi,\text{relu2\_2}}_{style}$
$\ell^{\phi,\text{relu3\_3}}_{style}$
$\ell^{\phi,\text{relu4\_3}}_{style}$
y_s
f_W
x
$\hat{y}$
Input Image
Image Transform Net
y_c
Loss Network (VGG-16)
ϕ
Content Target
$\ell^{\phi,\text{relu3\_3}}_{feat}$

Figure 2.8: The architecture of the perceptual-loss based style transfer [38].

Furthermore, AdaIN (Adaptive Instance Normalisation) [33] was proposed. AdaIN also bases its content and style loss functions on the features extracted from a fixed VGG-19 network. Aditionally, it introduces an AdaIN layer, which is used to perform style transfer in the feature space between an encoder and a decoder. Instead of using batch normalization (BN), AdaIn uses instance normalization (IN), and it argues that IN performs a kind of style normalization, which can adjust the mean and variance of the content features to match those of the style features per sample. AdaIn is defined in Equation 2.6, where x is the content input, y is the style input, $\sigma(x)$ and $\sigma(y)$ are the variance of x and y, and $\mu(x)$ and $\mu(y)$ are the mean of x and y.

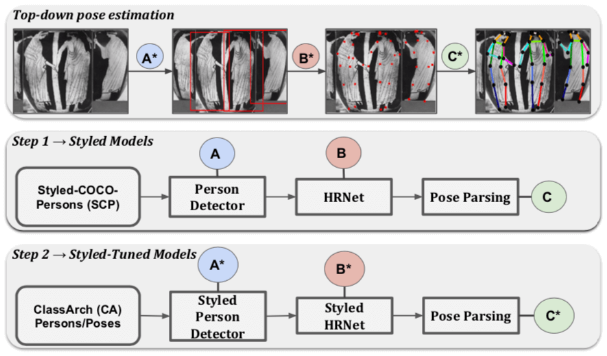
$$AdaIn(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \tag{2.6}$$

Since $AdaIn(x, y)$ is computed very fast given any style input y, AdaIN can hence adapt to arbitrary styles. In contrast, [38] is only fixed to a single style. What AdaIn and perceptual-loss model share in common is that: (1) They both use the same generator architecture, i.e. image transform net, as shown in Figure 2.8. (2) They both use a pre-trained VGG network to measure the content loss and style loss. (3) They are both trained for style transfer based on a pair of images, i.e. a single content image and a single style image. In contrast, generative models can perform style transfer given two domains, i.e. two sets of images, which will be introduced in Section 2.4.1. The only difference between AdaIn and perceptual loss is that AdaIn represents style features as style normalization, whereas perceptual loss and neural style transfer use Gram matrix.

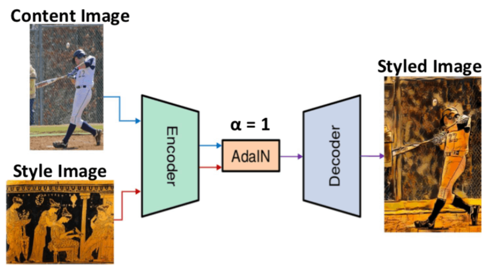
2.1.3 CNN and artworks

Machine learning has been applied to the computational analysis of artworks, such as Aby Warburg's Bilderatlas [34], Greek vase paintings [49], and artistic portraits [76]. Both [49] and [76] are based on the trained CNN models dedicated to the artistic paintings. [49] is to estimate the keypoints of poses, whereas [76] is to estimate the landmark points of faces.

Let's first delve into the pose estimation as in the vase paintings. The ancient Greek vase paintings are full of visual narratives, in which the protagonists are depicted by their actions and interactions conveyed through their poses composed against a certain scene. Similar scenes often encompass similar postures. Thus, to analyze and categorize the various poses can shed light on the characteristics of the protagonists. But the state-of-the-art (SOTA) pose estimation methods, e.g., OpenPose [8], do not generalize well in the domain of artistic paintings, as they are only trained based on the annotated dataset of photographs and videos. In order to improve the performance of SOTA pose estimators in the field of paintings, a custom pose estimator needs to be trained. As shown in Figure 2.9a, it includes two steps: (1) Train a styled model based on a styled dataset, as shown in the second row. (2) Fine-tune the styled model based on the dedicated dataset, as shown in the third row. The styled dataset is called SCP (styled COCO Persons), as it is comprised of stylized images from the original COCO Persons dataset. AdaIn [33] is applied to convert the images in style from the COCO Persons to SCP, as shown in Figure 2.9b. Generally, the COCO (Common Objects in Context) dataset contains a wide variety of labelled objects, which is mainly used to train a object detection model, and the COCO Persons dataset is a subset of COCO, which consists of annotated poses of people in various activities, such as baseball. Based on SCP, a styled model will be trained, and it is comprised of (A) person detector based on Faster R-CNN, and (B) pose estimator based on HRNet. The detailed architecture of Faster R-CNN and HRNet will be introduced in Section 2.5.2. Further, this styled model will be fined-tuned based on the dedicated CA (Classical Archaeology) dataset in order to zoom in and optimize for the main task: estimate the poses in the vase paintings. The CA dataset consists of 2629 person annotations and 1728 pose annotations from over 1000 Greek vase paintings, and it includes five different narratives, such as "Pursuits", "Abductions" and "Wrestling" in Agonal and Mythological contexts. After the second fine-tuning step, it will result in a style-tuned model. It is worth noting that the styled model, which was not trained on the CA dataset, increases the mean accuracy precision (mAP) of pose estimation by 7.62%, when tested on the CA dataset. The style-tuned model can further enhance the performance, when applied to any unlabelled vase paintings.



(a) The architecture of the top-down pose estimation for Greek vase paintings. The second row shows the first step, in which a styled model is trained based on the SCP dataset. The third row shows the second step, in which a style-tuned model is further trained based on the CA (Classical Archaeology) dataset. The style-tuned model is a fine-tuned model based on the styled model resulted from the first step.

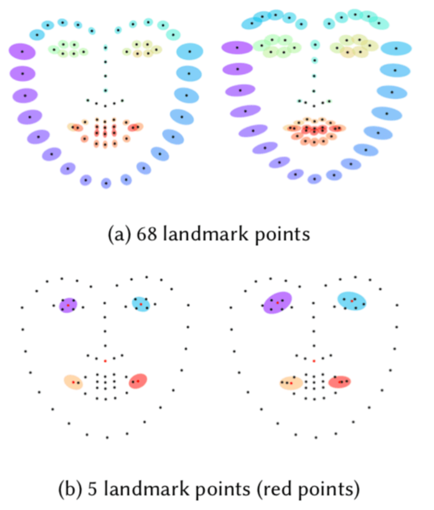


(b) AdaIn [33] was used as style transfer to prepare the SCP dataset in the first step.

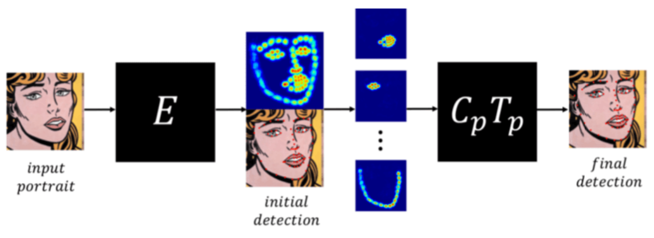
Figure 2.9: The overview of the CNN-based analysis of Greek vase paintings [49].

Second, let's zoom in on the facial landmark estimation for the portraits. First, the training data is mainly collected from the Painter by Numbers dataset, which consists of 103,250 paintings. Further, 16 artists are selected with a wide variety of styles, such as Israhel van Meckenem and Utagawa Kunisada. For each artist, 10 paintings are randomly chosen. In total, there are 160 images chosen for the custom artistic portrait dataset. To pre-process the images in the dataset, first, a multi-task cascaded CNN [78] is used to automatically detect the faces in the paintings. The bounding boxes containing the faces are then cropped and resized to images of 256×256 pixels. Then, initial landmark detection is applied to the resized images to extract 68 facial landmarks using the Dlib-ml toolkit [40]. Finally, after the detected facial landmarks are manually corrected, the artistic-faces dataset is ready. Additionally, a natural-faces dataset [64] is used as well, which contains the natural faces in-the-wild with 68-landmark annotations.

The geometric difference between a natural face in the photos and an artistic face in the paintings is shown in Figure 2.10a. The distributions are calculated based on the aforementioned artistic-faces dataset and the natural-faces dataset. It can be seen that the artistic faces have a larger geometric variation, which is common in artworks where artistic exaggeration and deformation often exist. Aside from the geometric variation, the artworks also possess a larger texture variation, with richer color palettes and wider options of medium, such as oil, acrylic and charcoal. Gram matrix in neural style transfer [21] can capture the texture variation in artworks.



(a) The landmark distributions of natural faces (left) and artistic faces (right). The black points (above) and red points (below) show the mean positions for 68 and 5 landmarks respectively, and the colored ellipses show the standard deviation in terms of each landmark. The artistic faces have larger variability with respect to the landmarks of eyebrows, eyes, corners of lips and chins.



(b) The framework of landmark detection in artistic portraits. For the input portrait, the estimation step (E) with the STN sub-network is performed to extract the initial 68 landmarks captured in the heatmaps. For each facial landmark, there is one corresponding heatmap. The coarse landmarks are further corrected and tuned in the part-based correction and tuning steps, C_p and T_p, in order to get the final detection. The part stands for each of the 68 facial landmarks, such as eyebrows, eyes or lips.

Figure 2.10: The overview of the geometry-aware style transfer for portraits [76].

The landmark detector is based on the ECT approach [77], and there are basically three steps: Estimation, Correction and Tuning, as shown in Figure 2.10b. First, in the estimation step, the network will be trained to predict the initial facial landmarks, given an artistic portrait. The network consists of a STN (Spatial Transformer Network) sub-network [36], which can warp the feature maps in the training phase to make the model more robust against geometric warping. The training data is based on the natural-faces dataset, upon which neural style transfer [21] and random geometric distortion are further applied in the data augmentation phase. For each facial landmark i, there will be a corresponding ground-truth heatmap M^i, which is a single-channel image of the same size as that of the input portrait, with each pixel (x, y) of the heatmap $M^i_{x,y}$ indicating the probability that the true landmark lies in the position (x, y). To train the estimation network, an input portrait (augmented natural face) and its paired ground-truth heatmaps of facial landmarks are used. The final estimation model can hence be applied to infer the heatmap for each facial landmark, given any artistic portrait. Second, in the correction step, each facial feature (based on the heatmap of each facial landmark) is further corrected using a pre-trained PDM (Point Distribution Model) [13] to remove the outliers and impose a global constraint on the shape of each facial feature, denoted as part-based correction C_p. Finally, in the tuning step, the predicted facial landmarks are adjusted between the heatmaps inferred from the estimation model and the global constraint calculated from the PDM, denoted as part-based tuning T_p. Moreover, the tuning step is only applied to the outline of the face, i.e. landmarks of the jaw and the union of the jaw and eye brows.

The result of applying different landmark detector methods is illustrated in Figure 2.11. *A* stands for whether data augmentation is applied to the natural-faces dataset. *ECT* is the baseline method based on the ECT approach [77], and EC_pT_p is the enhanced method, which can adjust each facial feature's shape without dependence between facial features, as illustrated in

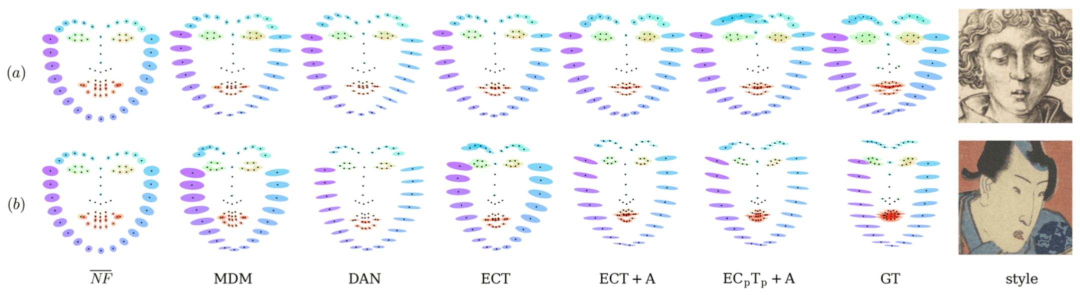


Figure 2.11: (a) contains artwork distributions of Israhel van Meckenem. (b) contains artwork distributions of Utagawa Kunisada. For each style, the ground-truth distribution (GT) is calculated from the artistic-faces dataset. ECT is the baseline method, and EC_pT_p is the enhanced method with part-based correction and tuning for each facial feature. A indicates whether data augmentation is used in the training phase to pre-process the natural-faces dataset. It shows that $EC_pT_p + A$ can better capture the geometric style compared to GT

Figure 2.10b. It shows that $EC_pT_p + A$ can better capture the geometric style of a specific artist compare to the ground truth GT.

In summary, in the domain of artistic paintings, the annotated dataset is nearly none in terms of pose keypoints and facial landmarks. Thus, it is necessary to apply style transfer to pre-process an already-annotated dataset of photographs. Based on the augmented dataset, a trained model can hence gain better performance as to pose and landmark estimation.

2.2 Autoencoder

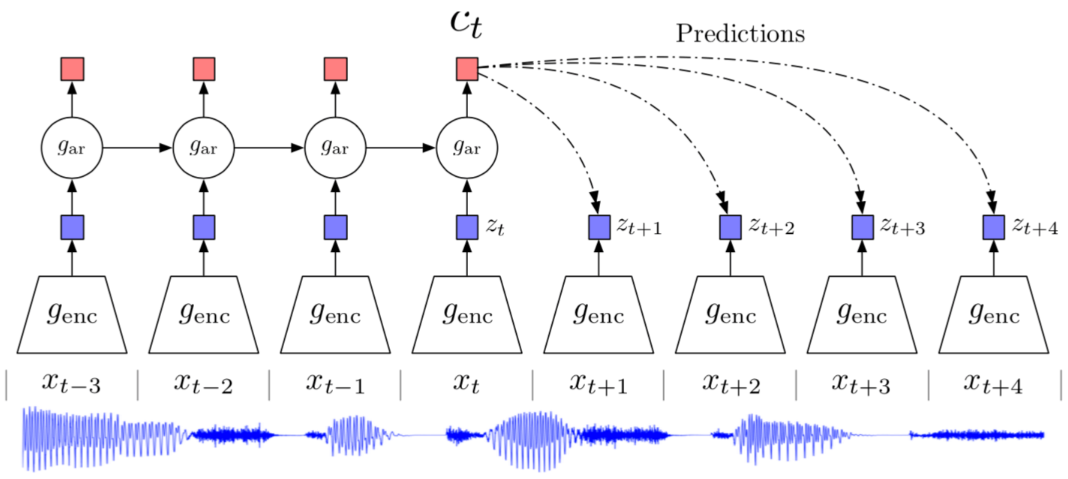


Figure 2.12: The architecture of Contrastive Predictive Coding (CPC) [73]: g_{enc} is a non-linear encoder, and g_{ar} is an autoregressive model.

CNN architecture is regarded as encoder, as it can bring high-dimensional data further to low-dimensional space. This deeply embedded space is known as latent space. It poses the question of how to extract meaningful representations from this latent space. **Discriminative autoencoder** tackles this problem by designing an objective function, which can help shape the latent space until it can embed discriminative information. One of such objective functions is to maximize the Mutual Information (MI), and there are basically two MI objectives: (1) The global MI objective is to maximize MI between the input and output representations of an encoder. (2) The local MI objective is to maximize MI between the local patches of the input. Deep InfoMax (DIM) [32], Augmented Multiscale Deep InfoMax (AMDIM) [6], and Contrastive Predictive Coding (CPC) [73] are its variants. Take CPC as an example, the overview of its architecture is shown in Figure 2.12.

First, an encoder g_{enc} maps the input sequence x_t to a sequence in the latent space z_t by Equation 2.7. Second, an autoregressive model g_{ar} computes the conditional distribution by integrating all $z_{\leq t}$ in the latent space to generate a context latent representation c_t by Equation 2.8. c_t can be used to predict multiple steps k in the future, where $k \geq 1$. Either z_t or c_t could be used as representation for downstream tasks, and c_t provides extra context from the past.

$$z_t = g_{enc}(x_t) \tag{2.7}$$

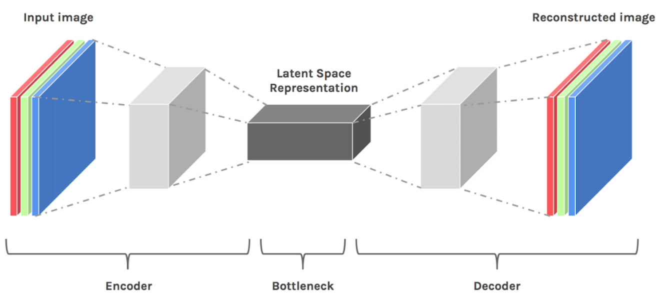
$$c_t = g_{ar}(z_{\leq t}) \tag{2.8}$$

The mutual information between input x and context c is defined as $I(x; c)$ in Equation 2.9. For a specific context c_t, in order to predict a future observation x_{t+k}, their mutual information $I(x_{t+k}; c_t)$ is calculated, instead of directly modeling the conditional distribution $p_k(x_{t+k}|c_t)$. $I(x_{t+k}; c_t)$ can be estimated by a lower bound, as in Equation 2.10, where N is the number of samples, and $\boldsymbol{\ell}_N$ is a NCE (Noise-Contrastive Estimation) loss function. The NCE loss is basically the categorical cross-entropy of classifying the positive sample correctly. Based on NCE loss, the whole model can be trained end-to-end, as minimizing the NCE loss $\boldsymbol{\ell}_N$ can maximize the mutual information $I(x_{t+k}; c_t)$.

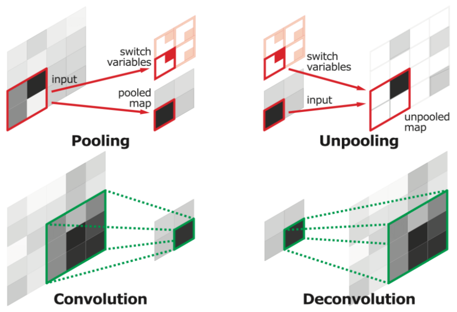
$$I(x; c) = \sum_{x,c} p(x, c) \log \frac{p(x|c)}{p(x)} \tag{2.9}$$

$$I(x_{t+k}; c_t) \geq \log(N) - \boldsymbol{\ell}_N \tag{2.10}$$

Compared to supervised CNN, CPC can learn representative features to achieve better performance in audio, image and text classification. In general, discriminative autoencoder is self-supervised, in a manner that it learns more generic representations from prior knowledge about useful structure in the data, rather than from explicit labels.



(a) The architecture of auto-encoder. The input image is encoded in the latent space, and then decoded to reconstruct the original input image.



(b) The comparison of the operations [53]: convolution vs. deconvolution, downsampling (pooling) vs. upsampling (unpooling).

Figure 2.13: The architecture of auto-encoder.

A **generative autoencoder** introduces a generative sub-network, also known as decoder, to the encoder, which is shown in Figure 2.13a. The encoder is often a standard CNN sub-network, and the decoder is just the reverse of encoder, which consists of two operations: (1) deconvolution, as opposed to convolution, and (2) upsampling (unpooling), as apposed to subsampling (pooling), which is shown in Figure 2.13b.

In detail, deconvolution maps a single value to a patch of multiple values, from one layer to the next layer. Similar to convolutional layers, deconvolutional layers can also capture different levels of details in its hierarchical structure. After deconvolution, unpooling will enlarge a feature map and restore each single value to its original location by referencing its switch variables, which record the source and destination locations of each value exactly at the time of pooling. The encoder reduces the size of feature maps through the combination of convolution and pooling, whereas the decoder enlarges the size of feature maps through the combination of deconvolution and unpooling. The reasoning behind the encoder-and-decoder design will be further explained in Section 2.4.1.

One of the generative autoencoder algorithms is variational autoencoder (VAE) [41]. Given a dataset $X = x_1, \dots x_N$ of N random samples, it is assumed that the data x are generated by a random process, involving an unobserved continuous random variable z in two steps: (1) A value z_i is generated from a prior distribution $p_\theta(z)$. (2) A value x_i is generated from a conditional distribution $p_\theta(x|z)$. Thus, the marginal likelihood of x can be denoted as $p_\theta(x)$ in Equation 2.11, and the posterior distribution of z can be denoted as $p_\theta(z|x)$ in Equation 2.12.

$$p_\theta(x) = \int p_\theta(x|z)p_\theta(z)\,dz \tag{2.11}$$

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)} \tag{2.12}$$

The likelihood of all N samples occurring together is denoted as joint distribution $p_\theta(x)$ in Equation 2.13, and the goal of a neural network is to learn the parameters θ that can maximize the joint likelihood $p_\theta(x)$. Maximize the likelihood $p_\theta(x)$ is computationally the same as maximizing the log-likelihood of it, as shown in Equation 2.14.

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i) \tag{2.13}$$

$$\log\big(p_\theta(x)\big) = \log\left(\prod_{i=1}^{n} p_\theta(x_i)\right) = \sum_{i=1}^{n} \log\big(p_\theta(x_i)\big) \tag{2.14}$$

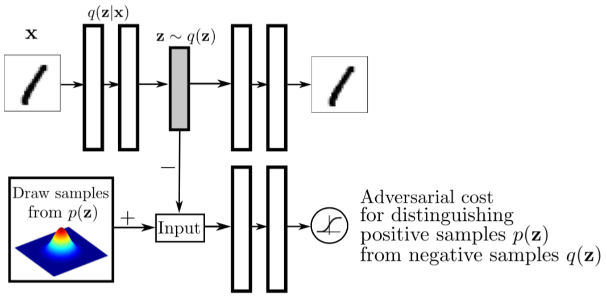
But the problem is that both $p_\theta(x)$ and $p_\theta(z|x)$ are computationally intractable, and $p_\theta(z|x)$ can be only approximated by a recognition model $q_\phi(z|x)$. In VAE, $q_\phi(z|x)$ is known as a probabilistic encoder with ϕ capturing the variational parameters, and $p_\theta(x|z)$ is a probabilistic decoder with θ capturing the generative parameters. ϕ and θ can be learned jointly in VAE, in which by minimizing the KL (Kullback-Leibler) divergence between $q_\phi(z|x_i)$ and $p_\theta(z)$, $\mathcal{L}(\phi, \theta; x_i)$ can be maximized. As shown in Equation 2.15, $\mathcal{L}(\phi, \theta; x_i)$ serves as a lower bound for the log-likelihood $\log\big(p_\theta(x_i)\big)$, maximizing $\mathcal{L}(\phi, \theta; x_i)$ can ultimately maximize the joint likelihood of $p_\theta(x)$.

$$\log\big(p_\theta(x_i)\big) \geq \mathcal{L}(\phi, \theta; x_i) \tag{2.15}$$

An **adversarial autoencoder (AAE)** [50] introduces an adversarial network, as shown in Figure 2.14a. Let x be the input and z be the hidden code in the latent space. Let $p_d(x)$ be the input data distribution, then $q(z|x)$ denotes an encoding distribution, $p(x|z)$ stands for the decoding distribution. Let $p(z)$ be the the prior distribution aimed to be imposed on z, $q(z)$ is then an aggregated posterior distribution of z over x, defined in Equation 2.16. The adversarial network is introduced to guide the aggregated posterior $q(z)$ to match an arbitrary prior $p(z)$.

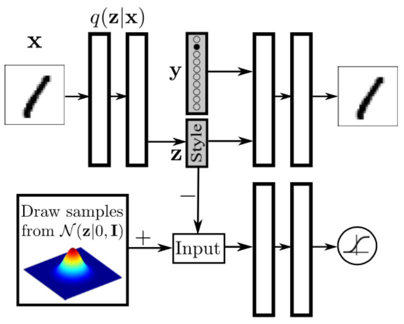
$$q(z) = \int_x q(z|x)p_d(x)\,dx \tag{2.16}$$

The difference between VAE and AAE is that: (1) For VAE, $q(z)$ is shaped by both data distribution $p_d(x)$ and the Gaussian distribution of $q(z|x)$: $z_i \sim \mathcal{N}(\mu_i(x), \sigma_i(x))$. For AAE, $q(z)$ is shaped by $p_d(x)$ and a random noise η as the input for the encoder, as defined in Equation 2.17. Thus, $q(z|x)$ is not constraint to be Gaussian any more, and the encoder can learn any posterior distribution for an input x. If η is a Gaussian noise, then VAE is a special case of AAE. (2) For VAE, a KL divergence penalty is used to impose the Gaussian distribution in z. For AAE, an adversarial network is used to impose an arbitrary distribution in z.



x
$q(\mathbf{z}|\mathbf{x})$
$\mathbf{z} \sim q(\mathbf{z})$
−
Draw samples from $p(\mathbf{z})$
+
Input
Adversarial cost for distinguishing positive samples $p(\mathbf{z})$ from negative samples $q(\mathbf{z})$

(a) The top row is a standard autoencoder, which can reconstruct an image x from a latent code z. The bottom row is an adversarial network, which is trained jointly with the above autoencoder to predict whether a sample comes from the latent space or from a prior distribution specified by the user.



x
$q(\mathbf{z}|\mathbf{x})$
y
z
Style
−
Draw samples from $\mathcal{N}(\mathbf{z}|0,\mathbf{I})$
+
Input

(b) The top row is an autoencoder, in which an image x can be reconstructed by the decoder, given a latent style vector z and a one-hot vector of its label y.

Figure 2.14: The architecture of an adversarial autoencoder (AAE) [50].

$$q(z) = \int_x \int_\eta q(z|x,\eta) p_d(x) p_\eta(\eta)\, d\eta\, dx \tag{2.17}$$

Compared to VAE, AAE has a significantly better performance in MNIST classification, which might indicate that AAE can learn better representations than VAE, as its latent space is more meaningfully shaped. Moreover, AAE can be used as style transfer, as illustrated in Figure 2.14b. To disentangle label information y from latent code z while training a decoder, content and style can be separated with y capturing the content, and z only encoding the style. For example, given a fixed label 1 in the MNIST dataset, by using different latent code z, different handwriting styles of 1 can be synthesized, e.g., italic, bold, etc. Similar to AAE's latent style code z, CIN (Conditional Instance Normalization) [16] also uses a random latent code to represent a specific style during style transfer. Thus, by choosing different codes, CIN can generate different styles given the input image x. The code of each style s is denoted as a set of parameters γ_s and β_s learnt during the training by CIN layer $CIN(x, s)$, as defined in Equation 2.18, where $\mu(x)$ is the mean of x, and $\sigma(x)$ is the variance of x.

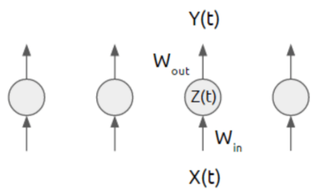
$$CIN(x,s) = \gamma_s\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \beta_s \tag{2.18}$$

Note that CIN is very similar to AdaIn [33], as shown in Equation 2.6, and the only difference is that various style codes in CIN are learnt during the training, whereas AdaIn computes a single target style in real-time based on an arbitrary reference style.

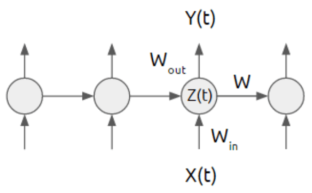
2.3 RNN

RNN is an advanced form of CNN, in which the previous states of the network will be used, as shown in Figure 2.15. Thus, RNN can learn the hidden dependence inherent in an array of data in order to predict the next character or the next note, given the previous sequence of characters or notes. So it is mostly used in the machine translation or music generation. By applying RNN to images, the relationships of the neighbouring pixels of an image can be learnt.

PixelRNN [74] is one of the RNN-based networks, which can be trained to automatically complete an occluded image, as shown in Figure 2.16a. The goal of PixelRNN is to estimate the distribution of natural images by modeling the pixels as discrete values using a multinomial distribution. The image is processed row by row, from left to right, from top to bottom. The probability of each pixel is a conditional distribution, given all the previously processed pixels as context. The distribution of an image is hence the product of the conditional distributions for all the pixels.



(a) The typical module of CNN.

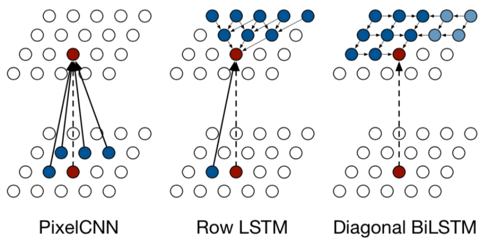


(b) The typical module of RNN.

Figure 2.15: The architecture of CNN and RNN [67]. $X(t)$ is the input of the state $Z(t)$, and $Y(t)$ is the output of the state $Z(t)$. W_{in} and W_{out} are the input and output parameters learnt during the training of CNN and RNN, while the input $X(t)$ is mapped to the state $Z(t)$. W are the parameters learnt to capture the inter-dependencies between states, while the previous state is mapped to the next state.

There are three proposed architectures, as illustrated in Figure 2.16b. LSTM (Long Short-Term Memory) is an advanced form of RNN modules, which can better conquer the vanishing gradient problem in the very deep network so as to better learn the long-distance dependencies. It argues that both PixelCNN and PixelRNN are able to capture the pixel inter-dependencies without introducing independence assumptions, as for the models based on autoencoders.

(a) The result of image completions from occluded images based on PixelRNN. Occluded images are on the left, and original images are on the right. The middle images are completed images inferred by PixelRNN.



(b) Visualization of the input-to-state (red-dot-to-red-dot) and state-to-state (blue-dots-to-red-dot) mappings for the three proposed architectures. PixelCNN is a fully convolutional network with masked convolutions applied. PixelRNN has two variants: (1) Row LSTM applies the convolution along each row, thus it only has a triangular receptive field which cannot capture the entire context. (2) Diagonal BiLSTM applies the convolution along the diagonals of the image, thus it can capture the entire context. Context denotes all the previous pixels.

Figure 2.16: The overview of PixelRNN [74].

Furthermore, SketchRNN [30] adopts the RNN modules in its encoder and decoder sub-networks. As shown in Figure 2.17, it is a sequence-to-sequence VAE, and its encoder is a bi-directional RNN. The input sequence S is first encoded as latent vector z, then the output sequence S' is generated conditioned on z. Compared to PixelRNN that builds the model based on pixel images, SketchRNN trains the model using the vector images, which are formed by a sequence of strokes. Let (x, y) be the absolute coordinates of the origin where the drawing starts, each point of a vector is denoted as $(\Delta_x, \Delta_y, p_1, p_2, p_3)$. Δ_x and Δ_y are the offset of moving the pen to the current position from the origin, and (p_1, p_2, p_3) is the one-hot encoded vector that captures the current states of the pen, where p_1 indicates whether the pen is touching the paper, p_2 indicates whether the pen is lifting, and p_3 denotes whether the drawing ends. SketchRNN is trained based on a dataset of human-drawn sketches from QuickDraw, and QuickDraw consists of 345 common objects, such as apples, boats, spiders, faces and yoga poses. For each object, there are $70k$ training samples, $2.5k$ validation samples and $2.5k$ test samples.

What PixelRNN and VAE share in common is that they both aim to learn the parameters θ

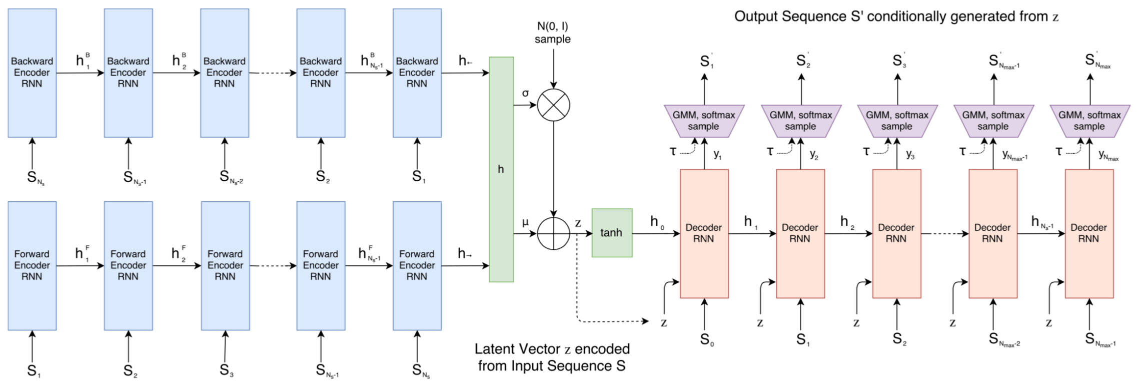

Output Sequence S' conditionally generated from z
N(0, I) sample
Backward Encoder RNN
Forward Encoder RNN
h
tanh
Decoder RNN
GMM, softmax sample
Latent Vector z encoded from Input Sequence S

Figure 2.17: The architecture of SketchRNN [30] is a sequence-to-sequence VAE. The encoder is a bi-directional RNN, and the input sequence S is encoded as latent vector z, the output sequence S' is generated conditioned on z. A sketch is represented as S', i.e. a sequence of strokes, and the maximum number of strokes is denoted as N_{max}. When the size of the generated sequence reaches N_{max} or $p_3 = 1$, the drawing ends. The sampled output is a random sequence, and its randomness can be controlled by a temperature parameter τ. When $\tau \longrightarrow 0$, the output will just consist of the most likely points from the probability density function.

in a neural network that can maximize the likelihood $p_\theta(x)$ for a dataset X of N random samples, i.e. $X = x_1, \ldots x_N$. VAE adopts an approximation approach by minimizing the KL divergence between posterior $q_\phi(z|x)$ and prior $p_\theta(z)$ for the latent code z, so as to indirectly maximize the lower bound of $p_\theta(x)$. In contrast, PixelRNN directly learns the parameters θ to estimate the real data distribution $p_\theta(x)$ by means of fully visible Bayesian networks (FVBN) [7] [20]. FVBN decomposes the observation's probability distribution of a n-dimensional vector v into a product of one-dimensional conditional distribution, as shown in Equation 2.19. Each conditional probability of v_i is dependent on its parents $v_{parents(i)} = v_1, \ldots, v_{i-1}$, and this time-series dependent property is also called autoregressive property.

$$p(v) = \prod_{i=1}^{n} p(v_i|v_{parents(i)}) \tag{2.19}$$

Since all $p(v_i|v_{parents(i)})$ are tractable, $p(v)$ is finally tractable. One of FVBN's variants is fully visible sigmoid belief network (FVSBN), and each conditional distribution $p(v_i|v_{parents(i)})$ can be calculated as a logistic regressor, as shown in Equation 2.20.

$$p(v_i|v_{parents(i)}) = sigmoid\left(\sum_{j<i} w_{ij}v_i + b_i\right) \tag{2.20}$$

Such logistic regressors can be further estimated by neural networks, as it uses the same formula as Equation 2.2, which can be learned during the training. Neural autoregressive distribution estimator (NADE) [43] is one of its implementations, and NADE's architecture is shown in Figure 2.18.

In summary, PixelRNN and autoregressive models, such as NADE, can model the data distribution explicitly by approaches of FVBN, and hence the high-dimensional data distribution estimation becomes tractable. Whereas, VAE can only approximate the estimation in a way that instead of modeling the data distribution directly, VAE models the data sampling process, which can generate the best data reconstruction from the latent space. Measured by performance, since PixelRNN and autoregressive models both generate the output y_i depending on the previous generated data, i.e. $y_{parents(i)}$, the generation process cannot be parallelized. The computation is slow, compared to VAE.

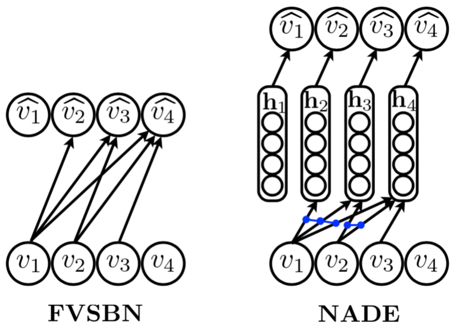

FVSBN
NADE
$\hat{v}_1$ $\hat{v}_2$ $\hat{v}_3$ $\hat{v}_4$
$\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$
v_1 v_2 v_3 v_4

Figure 2.18: Left: Illustration of fully visible sigmoid belief network (FVSBN). Right: Illustration of neural autoregressive distribution estimator (NADE) [43].

2.4 GAN

The two basic components of **GAN (Generative Adversarial Network)** are generator and discriminator. The generator is a decoder network, and the discriminator is an adversarial network. GAN and AAE share similar underlying architecture, however the difference is that GAN is used to impose the data distribution $p_d(x)$ at the pixel level on the output of a decoder, rather than on the output of an encoder. Thus, the generator of GAN is the decoder, whereas the generator of AAE is the encoder. GAN is often posed as a minimax (zero-sum) game between two players: generator and discriminator. Generator is defined by a function G with parameters $\theta^{(G)}$ that reads latent code z. Discriminator is defined by a function D with parameters $\theta^{(D)}$ that reads image data x. Both players have their own cost functions J defined in both players' parameters. The generator aims to minimize $J^{(G)}(\theta^{(D)}, \theta^{(G)})$ by only controlling $\theta^{(G)}$. The discriminator aims to minimize $J^{(D)}(\theta^{(D)}, \theta^{(G)})$ by only controlling $\theta^{(D)}$. Each player's cost function is partially dependent on the other player's parameters, but they can only control their own parameters. Thus, a Nash equilibrium is defined as a tuple $(\theta^{(G)}, \theta^{(D)})$, where $\theta^{(G)}$ is a local minimum of $J^{(G)}$ with respect to $\theta^{(G)}$, and $\theta^{(D)}$ is a local minium of $J^{(D)}$ with repsect to $\theta^{(D)}$ [26].

The cost function of a discriminator $J^{(D)}$ is defined in Equation 2.21, also called adversarial loss. Equation 2.21 is just a cross-entropy loss when training a binary classifier with a sigmoid activation. To train a discriminator, it needs two mini-batches of data: one comes from the real images of the training data x with all labels 1, and the other comes from the generated images of the generator $G(z)$ with all labels 0.

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbf{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbf{E}_z \log \big(1 - D(G(z))\big) \tag{2.21}$$

The cost function of a generator $J^{(G)}$ is defined in Equation 2.22, as in a zero-sum game, the total cost of both players should be zero. If a discriminator tries to minimize the cross-entropy as defined in Equation 2.21, a generator then tries to maximize the same cross-entropy by bringing $\big(1 - D(G(z))\big)$ near to 0, which is equivalent to bring $D(G(z))$ near to 1. Equation 2.23 is a non-saturating version of a generator's cost function, which can be interpreted as the generator tries to deceive the discriminator into classifying the fake image as true, thus $D(G(z))$ is equal to 1 and $J^{(G)}$ is equal to 0, which is hence minimized. It's worth noting that during the training, the generator is not directly exposed to the true image data x, and its performance is solely dependent on how good a discriminator can be to distinguish fake images from the real ones. Besides, mostly in GANs, the discriminator uses the same cost function as in Equation 2.21, but the generator will use a variety of cost functions, and non-saturating loss is just the original version of them.

$$J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -J^{(D)}(\theta^{(D)}, \theta^{(G)}) \tag{2.22}$$

$$J^{(G)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbf{E}_z \log D(G(z)) \tag{2.23}$$

The design of GAN was first introduced in [28], and the roles of discriminator and generator is depicted in Figure 2.19.

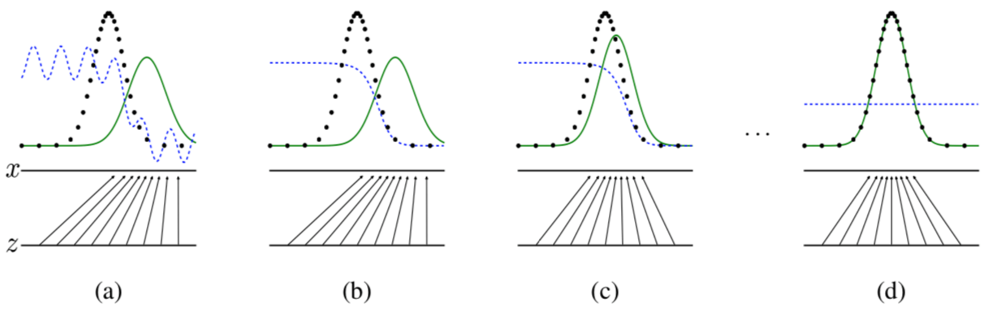


Figure 2.19: The architecture of GAN [28]. (a), (b), (c) and (d) shows the progress of the training. The **discriminator** (blue dashed line) learns the true distribution of samples, so it can distinguish between the images generated from the distribution of the latent space (green solid line) and the real training images (black dotted line), whereas the **generator** will learn to generate the fake images which pretend to come from the distribution of the training images. Finally, the discriminator cannot tell the fake images from the real images, so the probability to predict a fake or real image is equal to 0.5.

A typical realization of the GAN design for CNN is DCGAN (Deep Convolutional GAN) [59], which is an all convolutional architecture that contains neither pooling nor unpooling layers, as referenced from the all convolutional net [68]. Most GANs are to some extent based on DCGAN architecture, and its generator architecture is shown in Figure 2.20.

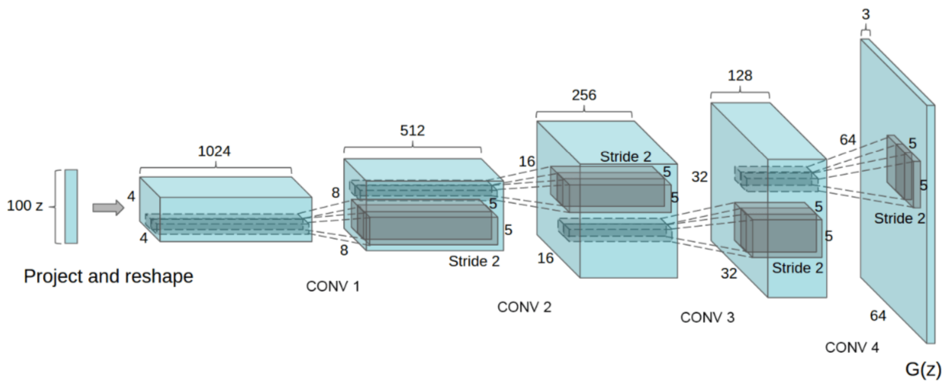
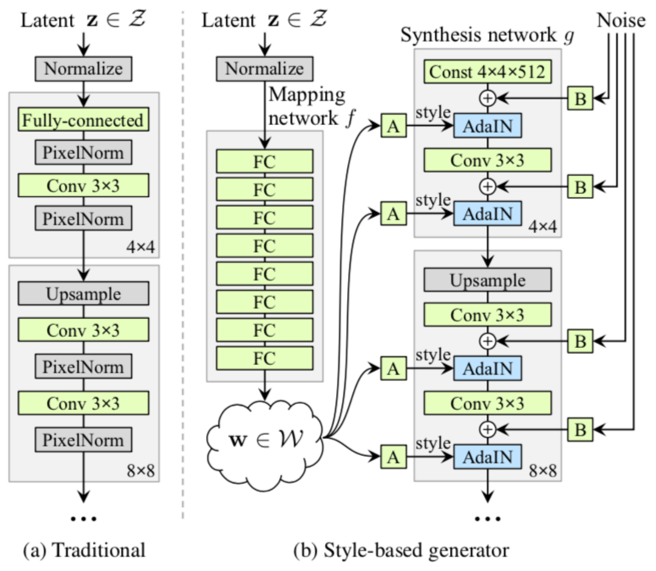
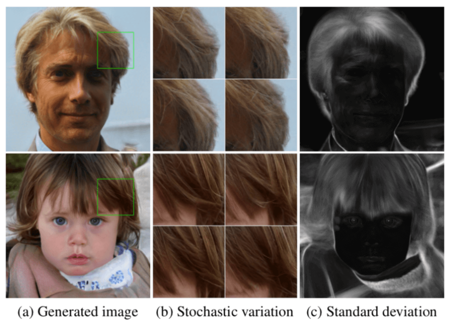


Figure 2.20: The architecture of DCGAN [59].

As illustrated in Figure 2.20, from the 100-dimensional latent code z, a 64×64 image can be reconstructed. Compared to VAE, the advantage of GAN is that it can generate high-resolution images, other than blurry ones, thanks to the contribution of discriminator. A blurry image will definitely be determined as a fake image by discriminator. Hence, it poses critical challenges as to how to train a discriminator. DCGAN adopts an architectural strategy by using the fully-convolutional layers only, eliminating the pooling, unpooling and fully-connected layers and employing batch normalization in order to make the model more stable and easier to train. But in general, GAN is subject to being unstable and hard to train, and a careful balance is needed for the training of the generator and the discriminator. Thus, Wasserstein GAN (WGAN) [4] was proposed, which does not require a special network design like DCGAN to stabilize the training. WGAN uses the Wasserstein distance instead of the original GAN's non-saturating loss, which is continuous and differentiable almost everywhere. Thus, WGAN can train a discriminator till optimality, as it has better gradients rather than vanishing gradients. However, in order to constrain the discriminator's trainable weights in a compact space $[-c, c]$, WGAN clips its weights by a fixed clipping parameter c after each gradient update. The drawback that arises for WGAN is that it is hard to tune the clippling parameter. If the clipping is small, it can lead to vanishing gradients, and if the clipping is large, it becomes harder to train the discriminator. Hence, an improved version of WGAN was proposed as WGAN-GP (WGAN Gradient Penalty) [29]. It was observed in [29] that (1) Weight clipping will either explode or vanish the gradients, which pushes the weights towards the lower and upper limits of the clipping range. (2) The discriminator trained via weight clipping is biased towards simpler functions, which cannot capture the real data distribution. Instead of directly interacting with the weights, WGAN-GP hence constrains the gradient norm of the discriminator by almost 1 everywhere by adding an extra gradient penalty to WGAN's loss function. With this update, WGAN-GP is more flexible with various GAN architectures and hyper-parameters, and is more stable to train with faster convergence.



(a) The architecture of StyleGAN: (a) Traditional generator. (b) StyleGAN generator: (1) Mapping network f, which is based on an 8-layer fully-connected (FC) network that maps a latent style code $z \in Z$ to $w \in W$. (2) Synthesis network g, in which a style A will be imposed on a constant input x, combined with noise B at multiple scales.



(b) Generated images of StyleGAN: (a) Generated image. (b)Stochastic variation introduced by injected noise inputs B. (c) Standard deviation of each pixel over 100 generations by different noise inputs. It shows that noise mostly affects hairs, and does not affect face and pose.

Figure 2.21: The architecture and generated images of StyleGAN [39].

Further, StyleGAN [39] was proposed based on WGAN-GP, which can generate stylized images based on a latent style code z, and architecture for the generator of StyleGAN is shown in Figure 2.21a. Compared to traditional generators, StyleGAN adopts a mapping network f which can map from latent space Z to W, and it argues that W is more linear and more disentangled, thus the factors of style variation can be properly separated. After computing style $y = (y_s, y_b)$ from factor w, y can be further fed into AdaIn layers at various scales of the synthesis network g, which is formulated as Equation 2.24, where x_i represents one feature map i, and $\mu(x_i)$ is the mean of x_i, $\sigma(x_i)$ is the variance of x_i. And $y_{s,i}$, $y_{b,i}$ are style parameter and bias for the feature map i.

$$AdaIn(x_i, y_i) = y_{s,i}\left(\frac{x_i - \mu(x_i)}{\sigma(x_i)}\right) + y_{b,i} \tag{2.24}$$

As introduced in Section 2.1.2, AdaIn transfers style by adjusting the feature statistics. Additionally, noise inputs are injected at each scale before AdaIn. The noise inputs are just single-channel random images consisting of Gaussian noise, which will introduce stochastic variations in the synthesize image, such as random curls of hairs, or random skin pores, which vary from image to image. As a result shown in Figure 2.21b, StyleGAN can generate photo-realistic images with finer details, and styles ranging from coarse, middle to fine features can be easier to control at various scales.

Aside from above, the general limitation of GAN is that it can only learn to generate images coming from the distribution of one domain. In other words, GAN cannot map pixel-by-pixel between two domains.

2.4.1 cGAN

To map images between two domains is known as the image-to-image translation problem. The related applications in computer vision include:

1. super-resolution [15] [44], in which a low-resolution image can be transformed into a high-resolution image in a CNN-based feed-forward network;
2. colorization [11] [79], in which a grayscale photograph is fed into a cross-channel encoder, and this network will learn to colorize the photograph which can pass "colorization Turing test";
3. depth and surface normal estimation [18], in which a photograph is paired with its depth and normal images in the multi-task training phase, and CNN will learn to detect the depth and normal of a given scene;
4. semantic segmentation [53] [48], in which an arbitrary image is paired with its corresponding segmented counterpart image as input, and CNN will learn to segment various objects (semantic contents) out of the image;
5. inpainting [56], in which an context encoder is trained to generate the missing region of an arbitrary image, given its surroundings;
6. style transfer [38] [33] [39], in which an input image can be stylised as output, given the reference style.

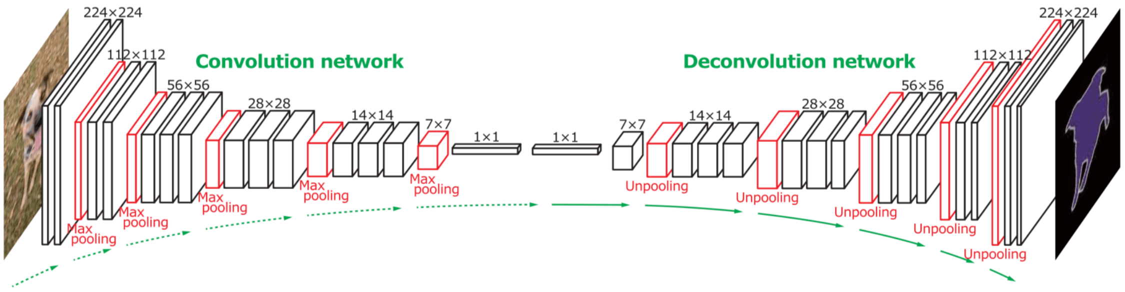


Figure 2.22: Semantic segmentation [53].

The semantic segmentation [53] example is shown in Figure 2.22, as most of them share the similar encoder-decoder architecture. The encoder-decoder design is an advanced CNN architecture, as (1) Encoder is comprised of convolutional layers, which transform from high-resolution to low-resolution, from high-dimension to low-dimension. (2) Decoder is comprised of deconvolutional layers, which transform from low-resolution to high-resolution, from low-dimension to high-dimension. The reasoning behind the mixture of encoder and decoder design is two-fold: (1) It is computationally intractable to estimate the probability distribution of high-dimensional data, e.g., data distribution $p_d(x)$, whereas it is tractable to estimate the probability distribution of low-dimensional data, e.g., prior distribution $p(z)$ and posterior distribution $q(z)$ in the latent space. (2) Downsampling combined with upsampling enables each pixel of the output image to have a larger receptive field from the input image. (3) Downsampling can reduce computational cost, as the size of feature maps gets smaller. In comparison, an encoder-only CNN design preserves the global information as to "what" the image is, thus it is aimed for classification tasks. Whereas a mixture of encoder-and-decoder design can preserve the spatial coordinates of the local information, thus it can link "what" simultaneously to "where" and solve the dense problems, such as super-resolution, colorization, semantic segmentation and inpainting as listed above. These image translation problems share the same encoder-decoder design, but with different loss functions to achieve the translation goal. How to define the style loss between an image in the target domain and an inferred image from the source domain for the transformation network is a non-trivial task. Mostly, a per-pixel loss function is used to minimize the per-pixel delta between a target image and a synthesized image. But the drawback of using per-pixel losses is that if two identical images are just offset by one pixel, though they are perceptually almost the same, they will be treated as two totally different images based on pixel-wise difference. So for style transfer [38], a perceptual loss function is used, as described in Section 2.1.2. Compared to it, cGAN introduces the discriminator to guide the training, thus adversarial losses are used, e.g., texture synthesis by markovian GAN [46]. Other loss functions were also proposed, such as MRF (Markov Random Fields) loss [45], histogram loss [61], CORAL (correlation alignment) loss [57] and MMD (Maximum Mean Discrepancy) loss [47].

Based on the GAN architecture, the image-to-image translation problem can be further generalized as a problem of **conditional GAN (cGAN)** [51] [22], in which a GAN network is trained to generate an image from the latent space conditioned on an input image. For example, a high-resolution image is generated conditioned on a low-resolution image, a colored photograph is generated conditioned on a grayscale image, and a semantically segmented image is generated conditioned on the original image. A generic cGAN implementation has been thus realized, which is called "pix-to-pix" [35], which can translate images between two different domains, whether they belong to grayscale-to-color, dog-to-cat, horse-to-zebra, edge-to-photo, etc. Moreover, this implementation requires minimal hyper-parameter tuning, and is used by various artists for a spectrum of interesting applications, e.g., color palette completion, sketch-to-cat, etc. To have a slight dip into the architecture of cGAN, Figure 2.23 can be referenced. A generator will learn to generate an image conditioned on variable y injected as input at the start, whereas a discriminator will decide whether the image is fake or genuine based on the same variable y in the end.

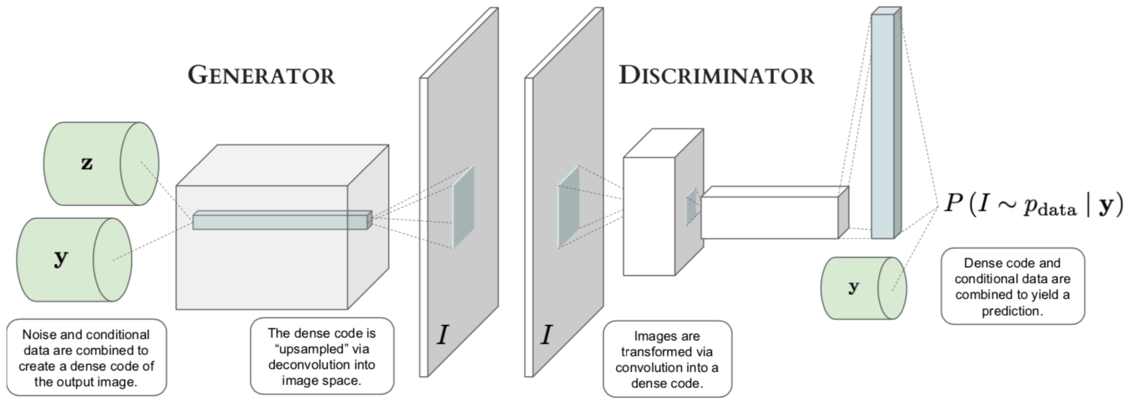

GENERATOR
DISCRIMINATOR
z
y
I
I
y
$P(I \sim p_{\text{data}} \mid \mathbf{y})$
Noise and conditional data are combined to create a dense code of the output image.
The dense code is "upsampled" via deconvolution into image space.
Images are transformed via convolution into a dense code.
Dense code and conditional data are combined to yield a prediction.

Figure 2.23: conditional GAN [22]. The **generator** will learn to generate the image I from the latent code z conditioned on variable y. The **discriminator** will learn to predict whether the image I comes from the true data distribution p_{data} conditioned on the same variable y.

The limitation of cGAN is that it needs a dataset of exactly paired images for the training phase, which is non-trivial for the dataset preparation. Thus, CycleGAN is introduced.

2.4.2 CycleGAN

CycleGAN was implemented in [81]. By using CycleGAN, the datasets no longer need to be paired, and the unpaired datasets can be used in the training phase, which significantly extend cGAN's capabilities into more applications, since an abundance of data already exists, ready to be tapped.

With respect to architecture, GAN and cGAN only have one generator and one discriminator, whereas CycleGAN contains two generators, G and F, and two discriminators, D_X and D_Y. Given two domains of input images X and Y, there are two mapping functions, in which $G : X \longrightarrow Y$, and inversely, $F : Y \longrightarrow X$. The adversarial loss of the discriminator D_Y for the generator G is defined in Equation 2.25. Reversely, the adversarial loss of the discriminator D_X for the generator F is defined in Equation 2.26. Compared to GAN's adversarial loss Equation 2.21, instead of mapping between input data x and latent code z, CycleGAN aims to map between input data x from domain X and input data y from domain Y.

$$\mathcal{L}(G, D_Y, X, Y) = \mathbf{E}_{y \sim p_{data}(y)} \log D_Y(y) + \mathbf{E}_{x \sim p_{data}(x)} \log\left(1 - D_Y(G(x))\right) \tag{2.25}$$

$$\mathcal{L}(F, D_X, X, Y) = \mathbf{E}_{x\sim p_{data}(x)} \log D_X(x) + \mathbf{E}_{y\sim p_{data}(y)} \log\left(1 - D_X(F(y))\right) \tag{2.26}$$

Additionally, cycle consistency loss is introduced, and it will encourage that $F(G(x)) \approx x$, and $G(F(y)) \approx y$. $F(G(x)) \approx x$ is defined as forward cycle consistency, and $G(F(y)) \approx y$ is called backward cycle consistency. The whole cycle consistency loss is defined in Equation 2.27, where $L1$ norm is used.

$$\mathcal{L}(G, F) = \mathbf{E}_{x\sim p_{data}(x)} \left\|F(G(x)) - x\right\| + \mathbf{E}_{y\sim p_{data}(y)} \left\|G(F(y)) - y\right\| \tag{2.27}$$

Combined with the adversarial losses on domain X and Y, the full loss function is thus defined in Equation 2.28, with λ controlling the relative importance of back and forward consistency.

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}(G, D_Y, X, Y) + \mathcal{L}(F, D_X, X, Y) + \lambda\mathcal{L}(G, F) \tag{2.28}$$

The illustration of cycle consistency loss is illustrated in Figure 2.24. Aside from CycleGAN's advantage of using unpaired data, when it comes to performance, CycleGAN performs well in style transfer. One test scenario is to transfer style of Monet, Van Gogh, Cezanne or Ukiyo-e to photos. The other test scenario is to convert horses to zebras. Specific to style transfer, it shows strength in: (1) Texture and color synthesis, and (2) By using PatchNCE loss, it can more accurately transfer the stripes of a zebra to the body of the horse, other than on other areas. The disadvantages are also obvious that it cannot morph between geometric changes, i.e. two domains having very different shapes. The failure cases are: (1) Transfer from dog to cat, and (2) Transfer from apple to orange. It can only convert a red apple to an orange apple, or convert a dog to a slightly different-colored dog.

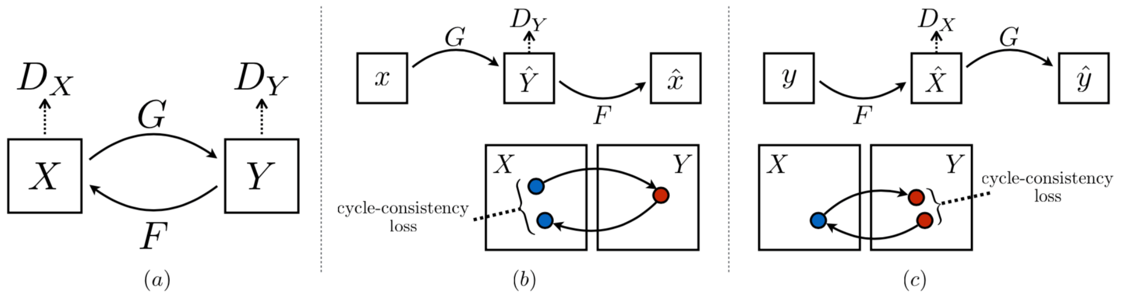


Figure 2.24: CycleGAN [81]. (a) G and F are two mapping functions, and D_X and D_Y are tow discriminators in domain X and Y. (b) (c) The cycle consistency loss in domain X and Y.

In summary, the challenges of style transfer based on the aforementioned varied architectures and loss functions are two-fold: (1) for color and texture, can they be transferred from the source patches to the right target patches? The intuitive example is that given the same shape "horse", how can the stripes of a zebra be transferred to only the body of a horse? (2) for geometric shapes, how can the shapes be morphed from the source domain to the target domain? The intuitive example is to morph from apple to orange.

2.5 HPE

Human Pose Estimation (HPE) basically comes into 2D and 3D HPE methods, which are typically based on three types of human body modeling as shown in Figure 2.25: (1) Kinematic model (used for 2D / 3D HPE), (2) Planar model (used for 2D HPE), and (3) Volumetric model (used for 3D HPE).

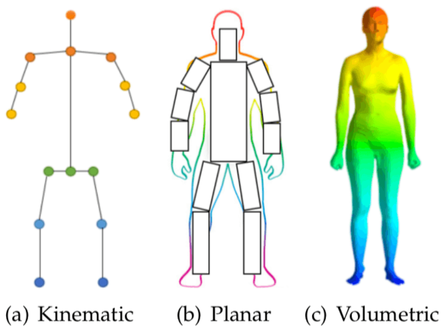


Figure 2.25: Three types of human body modeling [80].

2.5.1 2D single-person pose estimation

For the kinematic model, the points stand for the joint positions (keypoints), and the lines represent the limbs. In order to estimate the keypoints of a pose, the early works such as **k-poselet** [25] further breaks a pose down into k poselets, with each poselet representing a certain body part, i.e., a head or a torso. A poselet can be described by a learnt HOG filter as introduced in Section 2.1.1, and a deformation score that marks the relative shift between itself and an anchor poselet. Thus, a mixture of k HOG filters and k deformation scores can describe the whole pose with respect to the interaction between different body parts. In the same year, a CNN-based approach **DeepPose** [71] was raised as well.

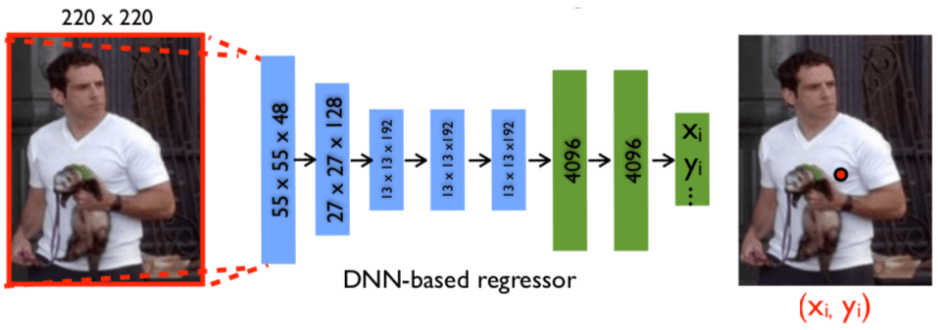
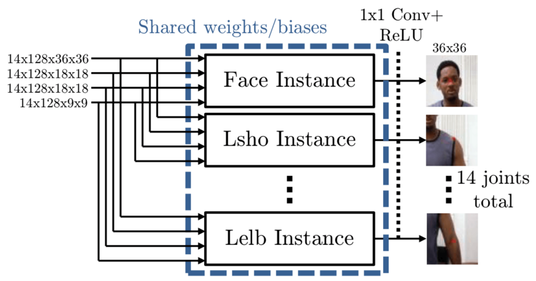


Figure 2.26: The 7-layered CNN architecture of DeepPose [71], which maps from the input 220×220 image, through 5 convolutional layers in blue and 2 dense layers in green, to the output pose vector $\boldsymbol{y} = (\ldots \boldsymbol{y_i} \ldots)$, where $\boldsymbol{y_i} = (x_i, y_i)$, which is the joint i's absolute coordinates in the input image.

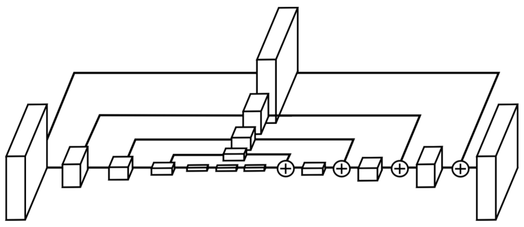
What makes DeepPose outstanding is that (1) The filters are not handcrafted for each body part. (2) The articulations between the body parts are not handcrafted in the deformation scores. (3) The average accuracy to predict the keypoints of arm and leg is much higher. As for DeepPose, the average accuracy is 60% on the Leeds Sports Dataset, whereas for k-poselet, it is 12% on the PASCAL VOC (Visual Object Classes) 2009 validation dataset. DeepPose formulates the pose estimation problem as a joint regression problem from the input image. And for a pose comprised of k joints, the pose vector is denoted as $\boldsymbol{y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \ldots \boldsymbol{y_k})$, where $\boldsymbol{y_i} = (x_i, y_i)$, $i \in (1, 2, \ldots k)$, which is the joint i's absolute coordinates in the input image. A 7-layered CNN architecture is used to map from the input image to the output pose vector, as shown in Figure 2.26.

An alternative to represent a pose of k joints is a set of **heatmaps**: $\{H_1, H2, \ldots H_k\}$. For the joint i's heapmap $H_i(x, y)$, where $i \in (1, 2, \ldots k)$, (x, y) stands for a pixel's coordinates in the input image, and $H_i(x, y)$ indicates the probability that the joint i's keypoints lie in the position (x, y). The CNN network is thus designed to minimize the Mean Squared Error (MSE) between the target heatmaps and the predicted heatmaps. Compared to the pose vectors, the heatmaps can capture richer information about the poses. An implementation of it is [70], and its architecture is shown in Figure 2.27a. The PCK (Percentage of Correct Keypoints) for full body on average is evaluated at 82% based on the MPII dataset. Generally, heatmap-based approaches perform the best for body part detection, according to the latest survey [80], when various approaches are tested based on the same MPII dataset using the same mAP measure for all body parts.

The GAN architectures were also explored, and the basic building block of GAN is the encoder-decoder (conv-deconv) module, as shown in Figure 2.22. The encoder-decoder module can be either symmetric or asymmetric, with an heavy encoder and a light decoder. In contrast to it, a hourglass module was proposed [52]. The hourglass module, as shown in Figure 2.27b, is a symmetric encoder-decoder module, with residual connections at every scale. Thus, a hourglass-based GAN can learn to be invariant to scale in a single pipeline, without using multiple resolution banks to process the image in separate pipelines as shown in Figure 2.27a. The hourglass module can be further stacked to allow for repeated inferences. By stacking the hourglass modules as a whole for pose estimation, the intermediate prediction accuracy increases gradually module by module. The PCK is evaluated at 90.9% on average based on the MPII dataset, which is much higher than 82% from the multiple-resolution-bank approach [70].



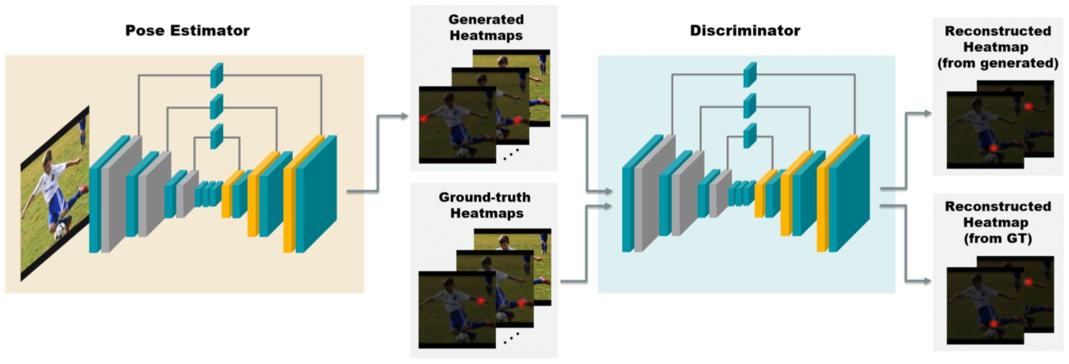
(a) For a total of 14 joints, the Siamese network learns 14 heatmaps correspondingly [70], where the input are of multiple resolutions banks, i.e., 36×36, 18×18 and 9×9, which will go through separate instances of the same 4-layered convolutional networks, i.e., Face Instance, Lsho (Left shoulder) Instance, and Lelb (Left elbow) Instance, till it will finally output 14 heatmaps for 14 joints respectively.



(b) The architecture of a single hourglass module [52]. It is a symmetric encoder-decoder architecture with residual connections linking corresponding scales.

Figure 2.27: The architecture of two heatmap-based pose estimators. To be scale invariant, the one on the left uses multiple resolution banks of the input image during the training phase in multiple pipelines, whereas the one on the right uses a symmetric hourglass module with corresponding scales being associated by residual connections in a single pipeline.

The adversarial loss can also be introduced to guide the pose estimation, and one example [12] is comprised of two stacked hourglass networks with exactly the same architecture, as shown in Figure 2.28.



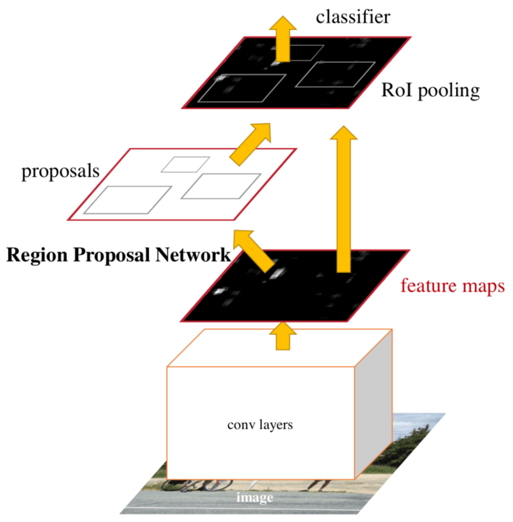
Pose Estimator
Generated Heatmaps
Ground-truth Heatmaps
Discriminator
Reconstructed Heatmap (from generated)
Reconstructed Heatmap (from GT)

Figure 2.28: The architecture of the heatmap-based two-stacked-hourglass GAN network to estimate the keypoints of a pose [12]: Pose estimator is the generator network, which produces the predicted heatmaps. The discriminator is jointly trained to distinguish between generated heatmaps and ground-truth heatmaps.

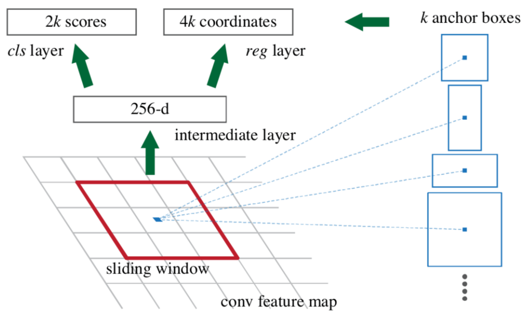
One of the hourglass networks is a generator, which is used as the original pose estimator that predicts the heatmaps for each joint from the input image. The other hourglass network is a discriminator, which is trained jointly to distinguish the target heatmaps from the generated heatmaps. By capitalizing on the adversarial training strategy, this GAN pose estimator's PCK is evaluated at 91.8%, which is slightly higher than 90.9% from the stacked-hourglass approach [52], when tested on the same MPII dataset.

2.5.2 2D multi-person pose estimation

Aside from the above methods to estimate the pose of a single person, there also are multi-person HPE methods, which can be classified into top-down and bottom-up approaches. In the top-down pipeline, there are two sub-tasks: (1) A human body detector to obtain each person's bounding box. (2) A single-person pose estimator to predict the locations of keypoints within the bounding boxes. In the bottom-up pipeline, there are also two sub-tasks: (1) Body joint detector to extract human body joint candidates. (2) Cluster the body joints into individual bodies. In general, the bottom-up methods outperform the top-down methods [80]. The reasons are: (1) The top-down methods suffer from early commitment. If the person detector fails at the first stage, the pose estimator will certainly fail subsequently, especially when multiple people are close to each other. (2) The runtime of the top-down approaches is proportional to the number of people in the image. In contrast, the bottom-up approaches are more robust, and run more efficiently regardless of how many people in the image.



(a) The overview architecture of Faster R-CNN. Region Proposal Network (RPN) is a type of fully-convolutional network (FCN), which can be regarded as a kind of decoder in the cGAN architecture. RPN shares the convolutional layers with the classifier network, and is used to generate the region proposals, which serve as "attention" mechanism to guide the classifier as to where to look.



(b) The architecture of RPN. Based on the feature maps of the conv layers, an anchor is centered at each sliding window. And for one sliding window, there are correspondingly k anchor boxes, each of which is associated with a scale and aspect ratio. By default, there are 3 scales and 3 aspect ratios, so $k = 9$ for each sliding window. For a conv feature map of size $W \times H$, there are in total $W \times H \times k$ anchor boxes. Each anchor box thus forms a proposal, and for one location, there are k proposals. The cls layer is a classification layer, which outputs $2k$ scores: (1) k scores for the probability of each proposal being an object, and (2) k scores for the probability of each proposal being a non-object. The reg layer is a regression layer, which outputs: (1) x coordinate of the center of the anchor box, (2) y coordinate of the center of the anchor box, (3) the width of the anchor box, and (4) the height of the anchor box for each proposal, so there are in total $4k$ coordinates.

Figure 2.29: The overview architecture of Faster R-CNN and the detail of RPN [60].

For the top-down methods, Faster R-CNN [60] is one implementation of person detectors, and HRNet (High-Resolution Net) [69] is one of pose estimators. Furthermore, the custom Faster R-CNN combined with the custom HRNet models have been successfully used to estimate the poses in the ancient Greek vase paintings [49]. Faster R-CNN [60] belongs to the object detection applications, which is not only limited to person detection. It is an improved version of Fast R-CNN [23] with respect to computational speed, and both versions are based on R-CNN [24] (Region-based CNN), in which multiple regions of interest (RoI) from the input image are extracted to form the proposals to classify the potentially detected objects. The overview of Faster R-CNN is shown in Figure 2.29. RPN (Region Proposal Network) can output pyramids of proposals with various scales and aspect ratios, thus Faster R-CNN is translation invariant. HRNet is just another variant of the heatmap-based pose estimators. Compared to the hourglass-based architectures, HRNet maintains high-resolution representations throughout the whole training phase, only adding high-to-low resolution subnetworks gradually in later stages to form multi-scale pipelines. Tested on the same MPII dataset, HRNet's PCK is 92.3%, which is slightly higher than 91.8% from the hourglass-based adversarial network [12], and also slightly higher than 90.9% from the stacked-hourglass network [52]. What is worth mentioning is that DensePose [62] also belongs to the top-down HPE methods, which is built basically based on Faster R-CNN and Feature Pyramid Network (FPN). DensePose can estimate the keypoints and segmentations of body parts for multiple people in real-time. Segmentation means the various areas of head, torso, arms, hands, legs and feet, and "Dense" is named after it, as these body parts can be densely predicted as areas, as opposed to points. The architecture of DensePose is shown in Figure 2.30. Based on the region proposal generation and RoI pooling, a fully-convolutional network follows to densely estimate the body parts and UV textures. Moreover, the predicted UV textures can be further used to carry out the texture transfer based on 3D SMPL (Skinned Multi-Person Linear) model. Based on the COCO (Common Objects in Context) mini validation dataset, the reported accuracy precision (AP) of segmentation and keypoints is 85.6% by multi-task learning, and 87.5% by multi-task learning with cascading, when geodesic point similarity (GPS) is set around 0.5. GPS threshold ranges from 0.5 to 0.95. GPS is 0.5 if the geodesic distance between the predicted keypoint and the ground truth equals the average half-size of a body segment, which is normally around 30 cm. If GPS goes higher, AP tolerates less deviation in the geodesic distance, and the precision of localization must go higher accordingly.

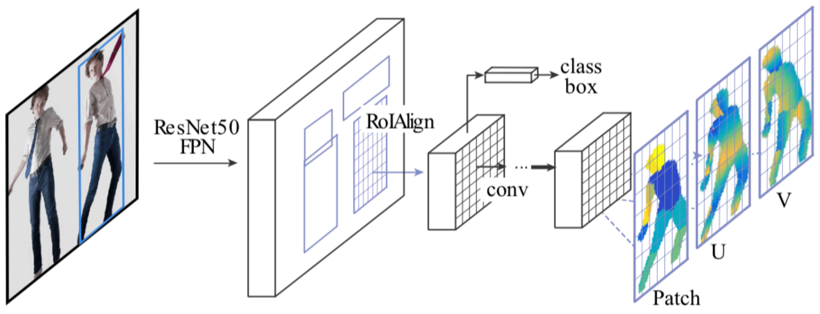


Figure 2.30: DensePose-RCNN architecture [62]: Based on the region proposal generation and feature pooling, a fully-convolutional network is used to densely predict discrete labels of body parts and UV textures.

For the bottom-up methods, OpenPose [8] is one of them. OpenPose is a real-time approach to estimate the poses for multiple people in one image, and its overall pipeline is shown in Figure 2.31.

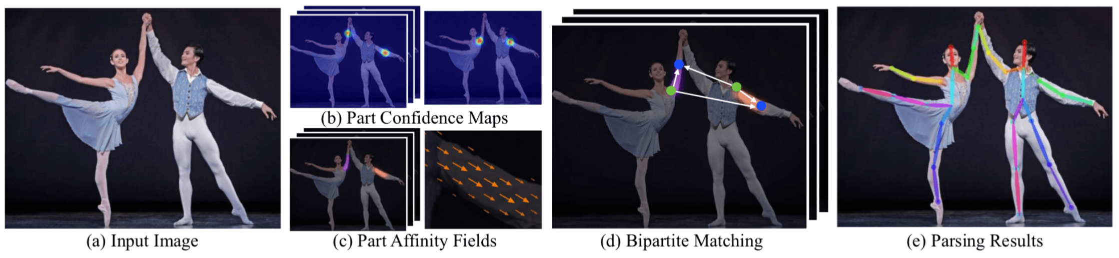


Figure 2.31: The pipeline of OpenPose [8].

For the input image, it will be fed into the first 10 layers of VGG-19 to extract a set of feature maps $\boldsymbol{F}$. $\boldsymbol{F}$ will be subsequently fed into the first branch of a CNN network to produce a set of part affinity fields (PAFs) $\boldsymbol{L}$. PAF is a 2D vector field representing one limb, which associates one joint to another, and hence, $\boldsymbol{L}$ is a set of PAFs that represent all the limbs. The degree of association is encoded in PAF as well, as PAF preserves (1) location, and (2) orientation information across the region of the limb, as shown in Figure 2.31 (c). After the iteratively multi-stage training of the first branch is complete, the latest $\boldsymbol{L}$ is then fed into the second branch of another CNN network, and this network will go through the same multi-stage training, and finally produce a set of 2D confidence maps $\boldsymbol{S}$, as shown in Figure 2.31 (b). A confidence map is a heatmap of one joint, and hence, $\boldsymbol{S}$ represents all the joints. Finally, the predicted confidence maps $\boldsymbol{S}$ and PAFs $\boldsymbol{L}$ are parsed by greedy inference, as shown in Figure 2.31 (d), to cluster the keypoints into each individual person. The final result is shown in Figure 2.31 (e). The tested mAP of OpenPose on the MPII dataset is 75.6%. Though it is much lower than a single-person pose estimator, it is still a promising result, given it runs in real-time for multiple-people pose estimation, and it can generalize to vehicle keypoint estimation as well.

Chapter 3: Datasets

For style transfer, there are two sources of images needed: (1) content images, and (2) style images. In Section 3.1, we will illustrate the chosen dataset for natural poses, which are used as content images. In Section 3.2, we will illustrate the chosen dataset for artistic poses, which are used as style images. In Section 3.3, we will explain how the annotations of joints and body segments are carried out for natural and artistic poses. In Section 3.4, we will explore the datasets of natural and artistic poses in a statistical manner to get an in-depth overview of the chosen datasets, which can further explain the accuracy of annotations to some extent. Finally, an overview of all the datasets used in the methodologies in Chapter 4 will be illustrated in Section 3.5, as for pragmatic reasons, different sets of images will be used for different purposes based on the trade-off between the effort of manual annotation and annotation accuracy.

3.1 Natural poses

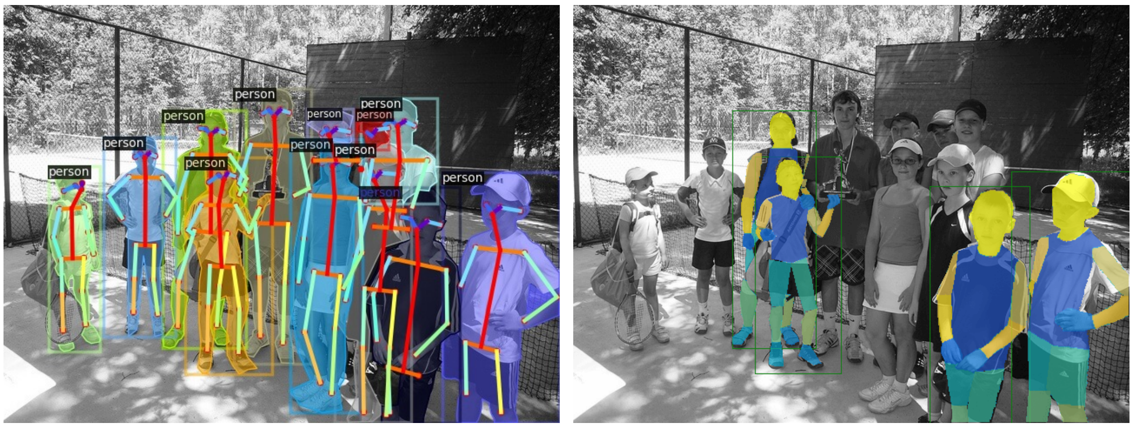
For natural poses, the datasets of the following sources have been considered:

- COCO (Common Objects in Context) dataset: It contains 80 categories of objects that belong to super categories including indoor, sports, outdoor, food, animal, vehicle, electronic, person, etc. For the annotations of category "person", it comes into bounding box, semantic segmentation, caption and pose in the format of Keypoints and DensePose, where Keypoints represent joints, and DensePose represents body segments, such as head, torso, arms and legs.

- **Academic institutions**: e.g., **People Snapshot Dataset** [2] which includes the monocular videos in which a person is moving, **VGG Human Pose Estimation Datasets** [58] which is a set of large video datasets based on Youtube and BBC videos annotated with human upper-body pose, **MPII Human Pose Dataset** [3] which includes around 25k images containing over 40k people with annotated body joints involved in 410 common activities, **LSP (Leeds Sports Pose)** dataset which contains 2000 annotated images of poses with sports people collected from Flickr, **UCF101** dataset which contains 101 action categories of realistic action videos from Youtube, and **FLIC (Frames Labeled In Cinema)** [65] dataset which contains 5003 images from Hollywood movies with actors in mostly front-facing standing-up poses.

- **Choreography, Dance, and other poses**: e.g., Living Archive by Wayne McGregor which contains a series of poses used in the choreography. AIST Dance Video Database which contains 515 Gigabytes of street dance videos [72].

Finally, the COCO dataset is selected, as COCO contains the human poses involved in daily activities that are shot and collected in Flickr. More specifically, the COCO train2014 dataset that contains annotated DensePose is chosen, where there are **26437** training images with 39210 people, **5984** validation images with 7297 people, and **1508** mini validation images with 2243 people. The training and validation images are used for training on a large and small dataset respectively, and the mini validation images are solely used for validation. The visualization of the COCO dataset for Keypoints and DensePose is illustrated in Figure 3.1. As shown, keypoints focus on joints, which come into **18** separate joints: left and right ears, eyes, shoulders, elbows, wrists, hips, knees, ankles, and one nose and neck. Whereas dense poses focus on body segments, which come into **14** coarse segments: left and right upper arms, lower arms, hands, upper legs, lower legs, feet, and one torso and head. Furthermore, these 14 coarse segments can be divided into 24 fine segments. In comparison, keypoints can show various kinds of poses, such as standing, sitting, or waving arms. Whereas dense poses can show the areas of muscles as to whether this person is thin or fat.



(a) The COCO Keypoints 2014 (b) The COCO DensePose 2014

Figure 3.1: The COCO 2014 dataset which contains the annotated bounding boxes, semantic segmentation and keypoints (Left), and annotated bounding boxes and semantic segmentation of body parts (Right) for each person.

3.2 Artistic poses

For artistic poses, the datasets of the following sources have been considered:

- Kaggle dataset: It contains several datasets of oil paintings with various styles, such as: 85 GB of Painter by Numbers, 35 GB of Wiki-Art : Visual Art Encyclopedia, and 600 MB of Art Images: Drawing/Painting/Sculptures/Engravings. “Painter by Numbers” is built based on “Wiki-Art” and Wikipedia, thus it contains an almost all-inclusive collection of all the paintings in the history with metadata describing their artists, style and fine-grained genre;

- **Museums**: To list a few, there are National Portrait Gallery, The Metropolitan Museum of Art and Rijksmuseum. Compared to the Kaggle dataset, a custom crawler needs to be built to download the raw images from these museums, as most of them only provide a search-engine based API to scan through their collections.

- Google Arts and Culture and WikiArt: These public-facing platforms provide a wide collection of artworks. It also needs a custom crawler, i.e., a tile fetcher that can download the different-level-zoomed images from Google Arts and Culture. One potential issue is that “HTTP Error 429” will be encountered when downloading above 200 images, as it is technically forbidden to send “Too Many Requests” to download their data.

Finally, “Painter by Numbers” is chosen, as it contains an almost full collection of paintings that range from the early 11th century to the 2010s. Every painting is uniquely indexed by numbers, and there are in total **103250** images, in which (1) For training, it contains 79433 images. (2) For test, it contains 23817 images. There are **2319** unique artists, (1) with styles ranging from Impressionism, Romanticism, Expressionism, Rococo, Baroque to Ink and wash painting and Gongbi, and (2) with genres ranging from religious painting, portrait, landscape, still life, nude painting (nu) to shan shui and calligraphy. Aside from artist, style and genre, the metadata of “Painter by Numbers” also consists of date, numbered filename and whether it belongs to only training, only test or both training and test.

The paintings that are of interest to us are a trade-off between varied styles, vast exposure of limbs and a balance of gender. To explain, the reasoning is that: (1) With styles as varied as possible, we can further explore whether there exist outstanding characteristics pertaining to a specific style. Thus, the styles across a wide time span are first sifted through into two major categories - “classical” and “modern”, which are demarcated by before and including Impressionism. (2) To estimate the poses more accurately, it is the best that the people depicted are nude or with close-fitting clothes. Thus, we tend to select images from the genre - nude paintings. Moreover, it is also desired that in the paintings, people are not occluded with each other in a crowd, and their limbs are fully exposed. (3) Men and women are physically different. Thus, we tend to choose half men and half women for pose analysis. But after roughly leafing through the dataset, the classical paintings appear to have disproportionately far more nude men, whereas the modern paintings seem to focus merely on nude women. If the aforementioned three preconditions are met, the paintings will be shortlisted. Last but not least, pornography will be discarded. Now, we will explore the dataset to determine the painters and their paintings to be chosen.

Figure 3.2 shows an overview of the styles contained in the dataset ordered by the timeline, given the precondition that the number of paintings of each style must be greater than **1000**. As illustrated, there are in total 21 styles with sufficient paintings to choose from for a start. Specifically, there are 4400 paintings of Baroque style, and prior to Baroque, there are 6386 paintings in Renaissance. Moreover, Impressionism (10643), Realism (10523), Romanticism (9285), Expressionism (7013) and Post-Impressionism (5778) have the top-5 number of paintings, all of which is above 5k.

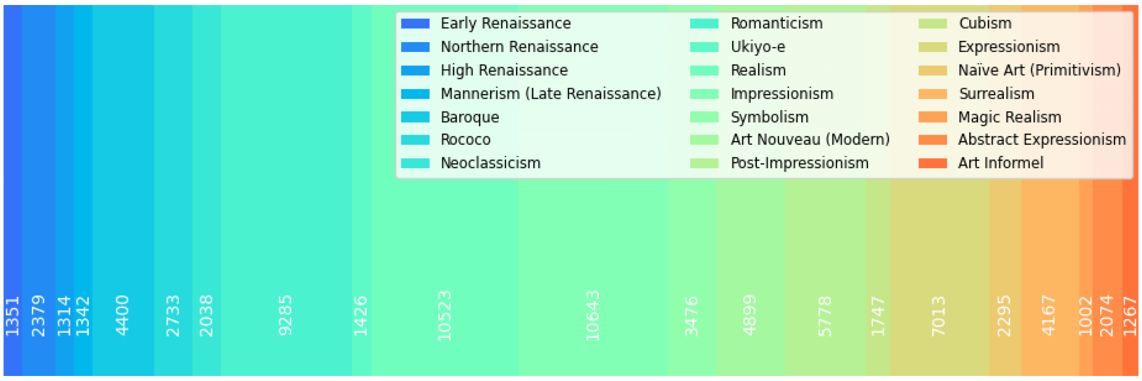


Figure 3.2: Top-21 styles from the “Painter by Numbers” dataset temporally ordered from left to right. The number of paintings of each style is shown by the numbers, and illustrated by its relative width in the timeline.

Next, we want to explore which artists have the most nude paintings. Figure 3.3 shows all the artists who drew more than **15** nude paintings.

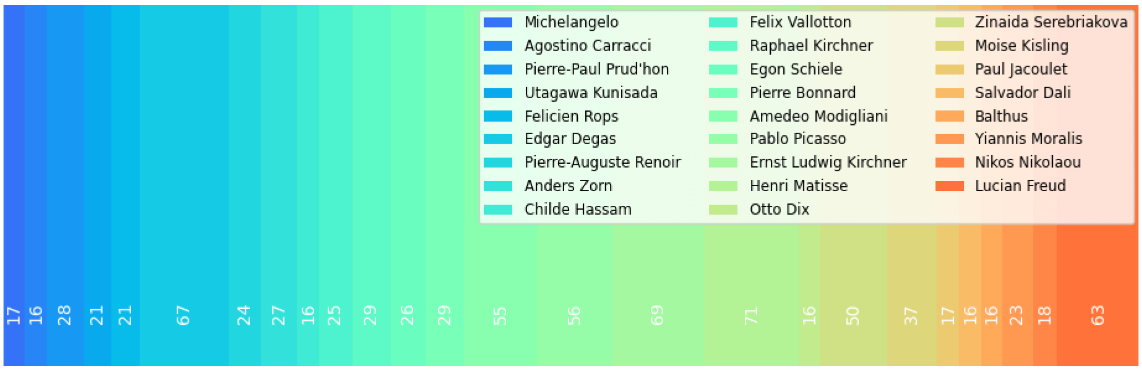


Figure 3.3: Top-26 artists who has painted more than 15 nude paintings, which is temporally ordered from left to right.

Figure 3.3 illustrates that Henri Matisse (71), Ernst Ludwig Kirchner (69), Edgar Degas (67), Lucian Freud (63), Pablo Picasso (56), Amedeo Modigliani (55) and Zinaida Serebriakova (50) have at least 50 nude paintings, and they all belong to modern artists. Zooming in, the nude paintings of Henri Matisse belong to styles ranging from Post-Impressionism, Abstract Expressionism, Expressionism, Fauvism and Orientalism. Ernst Ludwig Kirchner falls into Expressionism and Japonism. Edgar Degas belongs to Impressionism and Japonism. Lucian Freud is categorised as Contemporary Realism and Expressionism. Amedeo Modigliani is from Expressionism. Pablo Picasso straddles from Neoclassicism, Realism, Post-Impressionism, Analytical Cubism, Cubism, Expressionism, Naive Art and Surrealism. Zinaida Serebriakova ranges from Art Nouveau, Expressionism, Art Deco and Symbolism. In contrast, Michelangelo (High Renaissance), Pierre-Paul Prud'hon (Neoclassicism) and Pierre-Auguste Renoir (Impressionism) are classical artists. We can see that modern artists tend to switch between multiple styles, whereas classical artists more revolve around a single style.

Figure 3.4 further shows the top-16 styles for the genre - nude paintings. Expressionism and Impressionism dominate the major parts, most of which depict nude women. In comparison, Baroque and Renaissance only contribute the least, and most of which draw nude men.

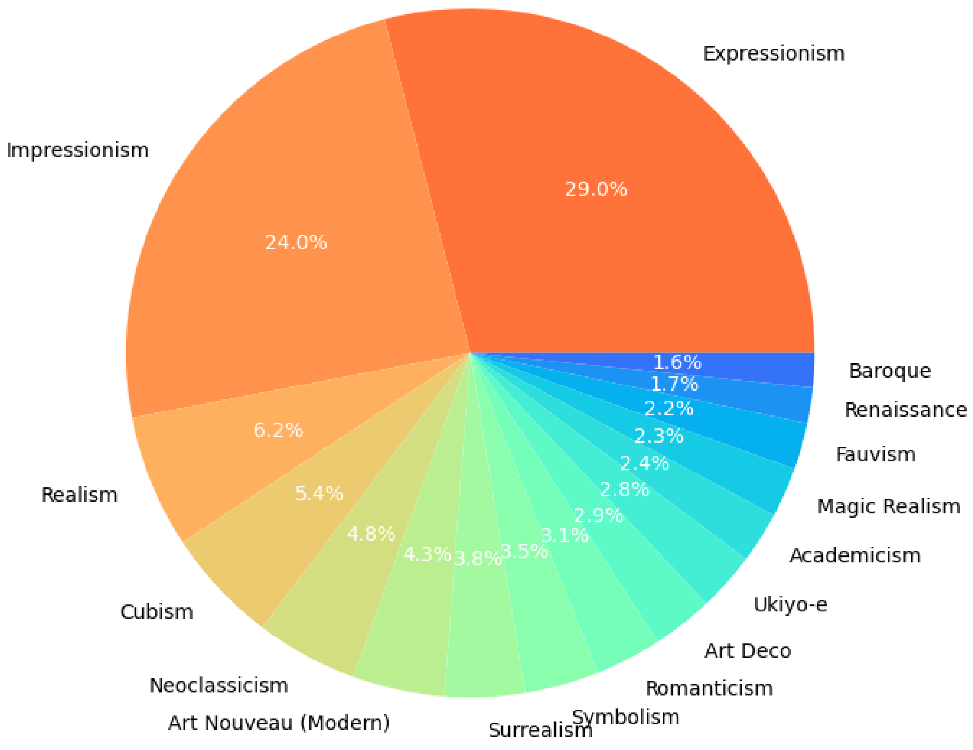


Figure 3.4: Top-16 styles of nude paintings.

To have a well-balanced set of classical and modern paintings, our goal is to choose 5 classical artists and 5 modern artists, each with 15 paintings of consistent styles. For classical artists, Michelangelo, Pierre-Paul Prud'hon, and Pierre-Auguste Renoir are first chosen, as they have sufficient nude paintings. Furthermore, to balance the gender difference and enhance the variance of styles, Artemisia Gentileschi (Baroque) and El Greco (Mannerism) are chosen with a selected collection of poses with close-fitting clothes. For modern artists, Felix Vallotton (Magic Realism) and Amedeo Modigliani (Expressionism) are first chosen. Henri Matisse, Ernst Ludwig Kirchner, Edgar Degas, Lucian Freud, and Zinaida Serebriakova are not chosen, as their paintings mostly fall into Impressionism, Post-Impressionism and Expressionism, which overlap in styles that have been already represented by Pierre-Auguste Renoir and Amedeo Modigliani. Pablo Picasso is not chosen, as the poses in his paintings are mostly abstract that cannot be confined by one style. Furthermore, Paul Gauguin (Cloisonnism), Tamara de Lempicka (Art Deco), and Paul Delvaux (Surrealism) are chosen, as their paintings uniquely contribute to the reservoir of styles.

Finally, the following artists are chosen, with the styles ranging from High Renaissance, Mannerism, Baroque, Neoclassicism, and Impressionism for classical artists, to Cloisonnism, Magic Realism, Expressionism, Art Deco, and Surrealism for modern artists respectively.

- Classical artists:
  1. Michelangelo
  2. El Greco
  3. Artemisia Gentileschi
  4. Pierre-Paul Prud'hon
  5. Pierre-Auguste Renoir

- Modern artists:
  1. Paul Gauguin
  2. Felix Vallotton
  3. Amedeo Modigliani
  4. Tamara de Lempicka
  5. Paul Delvaux

Figure 3.5 and Figure 3.6 give a glimpse of one typical pose of the chosen painter's paintings respectively.

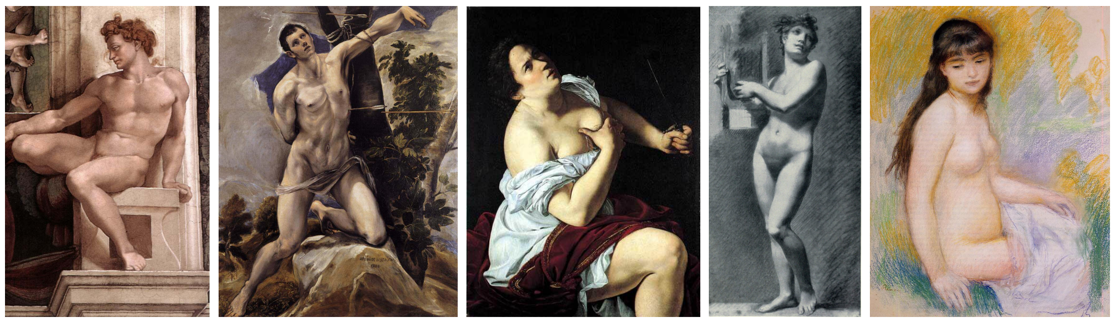


Figure 3.5: Classical artists (from left to right): Michelangelo, El Greco, Artemisia Gentileschi, Pierre-Paul Prud'hon, Pierre-Auguste Renoir.

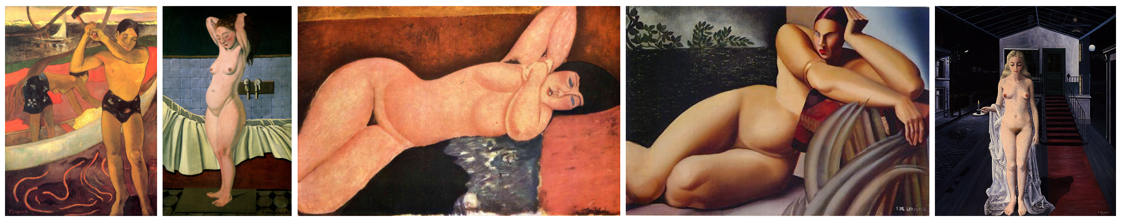


Figure 3.6: Modern artists (from left to right): Paul Gauguin, Felix Vallotton, Amedeo Modigliani, Tamara de Lempicka, Paul Delvaux.

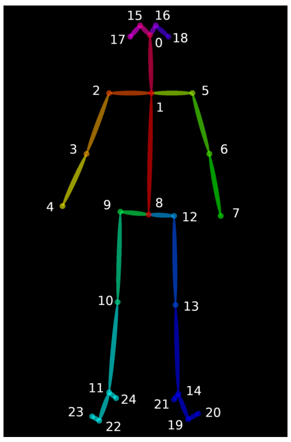
3.3 Annotations

For natural poses, since the COCO people dataset has already been manually annotated, we will use its annotations of joints and body segments directly as in Figure 3.1. For artistic poses, we will use OpenPose to infer their joints, and use DensePose to infer their body segments respectively. The inferred joints and body segments will be used as their annotations. The reason why we prefer inference to manual annotation of artistic poses is a trade-off between annotation accuracy and effort. The manual annotation can lead to the highest accuracy at the cost of intensive effort, whereas for the inference, we can adopt the state-of-the-art pose estimation tools such as OpenPose and DensePose in the domain of paintings, and focus on implementing a generic method for geometry-aware style transfer between natural and artistic domains with acceptable accuracy resulting from inference. Thus, in Section 3.3.1, we will analyze the inference accuracy of joints, and in Section 3.3.2, we will analyze the inference accuracy of body segments in order to determine whether we can proceed with our methodologies based on inference of OpenPose and DensePose, and which datasets are suitable for which purposes, which will be further listed and illutrated in Section 3.5.

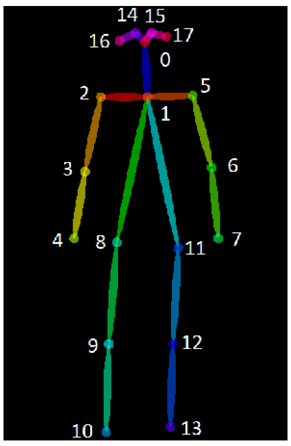
3.3.1 Joints

To analyze keypoints for each pose, we first carry out an experiment to compare the performance of OpenPose and DensePose respectively, regarding their inference time and accuracy. For inference time per image, when executed on MacOS without GPU, OpenPose takes 10 seconds on average, and DensePose uses 9 seconds. For inference accuracy, the types of error are defined as, for one person, (1) Incorrect keypoints, and (2) Missed-out keypoints. For the whole image, (3) missed-out people, e.g., if there are two people in the painting, only the keypoints of one person are fully inferred.

For OpenPose, the BODY 25 model is used that consists of 25 annotated joints, as shown in Figure 3.7a. For DensePose, the COCO model is adopted that otherwise contains only 18 joints, as illustrated in Figure 3.7b. The COCO model does not annotate the middle hip and the extra 6 feet joints, i.e., left and right big toes, small toes and heels, which are numbered from 19 to 24 in Figure 3.7a.



(a) The BODY 25 model with 25 keypoints.



(b) The COCO model with 18 keypoints.

Figure 3.7: Two models of keypoints.

Additionally, to calculate accuracy, two scopes of validity are defined: one is for the valid keypoints. The other is for the valid people. For the complete set of the valid keypoints, they include: (1) Upper limbs, which are left and right shoulders, elbows, wrists (Indexed from 2 to 7 in both models), (2) Neck (indexed as 1 in both models), (3) Lower limbs, which are left and right hips, knees, ankles (Indexed from 9 to 14 in the BODY 25 model. From 8 to 13 in the COCO model), (4) Middle hip (Indexed as 8 in the BODY 25 model) only by OpenPose, and it is marked as the average point between left and right hips by DensePose, and (5) Nose (Indexed as 0 in both models). The valid keypoints are those included in the above complete list (15 joints by OpenPose and 14 by DensePose), plus one extra condition that they exist within the frames, thus, they can be implicitly inferred. For the valid people, they are the people in the foreground. The people in the background, appearing in mirrors or as sculptures and shadows, etc., don't count. The reason to adopt these two scopes is that (1) For pose analysis, only torso, upper and lower limbs are required, whereas eyes, ears and feet are not necessary. (2) For people, we only attach importance to the protagonists, which are also highlighted in a scene to convey the theme of a painting.

To measure accuracy, PCK (Percentage of Correct Keypoints) is used. The keypoint is considered correct if the distance between the inferred and the true keypoint is within a certain threshold. Normally, PCKh@0.5 or PCK@0.2 is used. PCKh@0.5 denotes that the threshold is within 0.5 of the head size, and head is represented by its bounding rectangle. Thus, head size is defined by the diagonal of the rectangle. PCK@0.2 means that the threshold is within 0.2 of the torso diameter, which is the diagonal of the torso's bounding box. In MPII, PCKh@0.5 is used. In FLIC, PCK@0.2 is used. Here, PCK is used as a raw measure of accuracy without the threshold computationally defined, as it is quite obvious to observe whether a keypoint is correctly or incorrectly inferred, and the manual checking effort is not large, as it is only limited to 150 paintings of 10 artists.

The accuracy for each artist is shown in Table 3.1. The total number of the valid people is shown in the second column, and that of the valid keypoints is shown in the third column. PCK by OpenPose and DensePose is shown in the fourth and fifth column respectively. For all artists, the keypoints are solely inferred by OpenPose and DensePose, and no manual annotation has been done yet. The reasoning to compare OpenPose with DensePose even when they adopt different body models is that we only use 15 keypoints for pose analysis, which are nose, left and right shoulders, elbows, and wrists, left and right hips, knees, and ankles, neck, and midhip. Out of these 15 keypoints, OpenPose and DensePose have 14 keypoints in common, and the midhip can be also conveniently calculated by the average of the left and right hips.

	Total people	Total men	Total women	Total keypoints	Accuracy by OpenPose	Accuracy by DensePose
Michelangelo	15	14	1	214	186 (87%)	135 (63%)
El Greco	34	32	2	375	274 (73%)	207 (55%)
Artemisia Gentileschi	21	6	15	214	184 (86%)	129 (60%)
Pierre-Paul Prud'hon	15	7	8	216	201 (93%)	177 (82%)
Pierre-Auguste Renoir	19	0	19	237	190 (80%)	150 (63%)
Paul Gauguin	31	4	27	414	339 (82%)	305 (74%)
Felix Vallotton	20	1	19	243	210 (86%)	164 (67%)
Amedeo Modigliani	15	0	15	180	72 (40%)	54 (30%)
Tamara de Lempicka	18	1	17	227	157 (69%)	118 (52%)
Paul Delvaux	35	4	31	455	407 (89%)	404 (89%)

Table 3.1: The comparison of PCK by OpenPose and DensePose.

On average for all artists, the inference accuracy of OpenPose is around 80%, and it is around 66% for DensePose. As a whole, OpenPose performs better than DensePose. The factors that matter in inference of keypoints in paintings are: (1) The number of people in the painting, (2) The occlusions of clothes and interactive people, (3) The niche poses, i.e., cuddling oneself or lying on one's arms, whereas the easy poses are sitting or standing, (4) The niche perspectives, i.e., viewing from bottom up, (5) The color contrast, i.e., the background merges with the contour of a person, (6) The shapes of body segments, especially when the body parts are exaggerated, and (7) The body proportions, e.g., the elongated torso and limbs.

The reason why OpenPose performs better might be that it is a bottom-up method, whereas DensePose is a top-down one. Generally, the former performs better than the latter, as the top-down method suffers from early commitment by object detection. The outcome of inference based on bounding box is double-edged: (1) It can introduce the wrongly-connected limbs, which finally lead to the wrongly-inferred pose. (2) The limbs can be deduced given its bounding box, which can otherwise lead to the correctly-inferred pose.

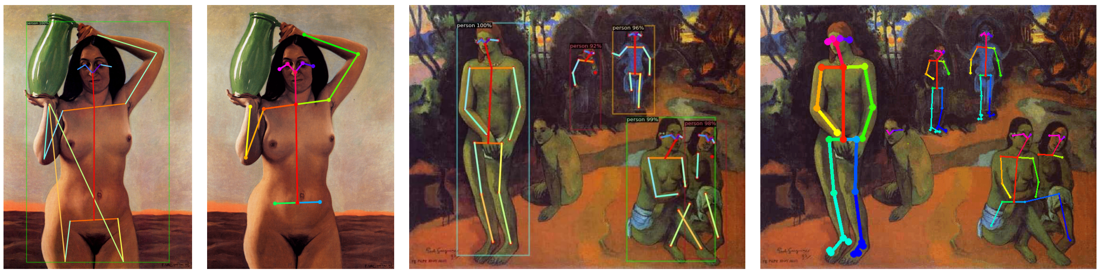


Figure 3.8: The outcome of DensePose (left) vs. OpenPose (right) given bounding box.

As shown in Figure 3.8, there are two pairs of paintings, where the leftmost ones result from DensePose, and the rightmost ones are of OpenPose. The failure case is the standing woman holding a vase on her shoulder. Constrained by its bounding box, the legs are inferred pointing skyward. The success case is the painting where there are two women sitting in the right corner cross-legged. DensePose deduces the joints based on two bounding boxes of person, so the limbs are correctly inferred, and they are disconnected, whereas OpenPose tends to connect the limbs of two people into one, thus the limbs are wrongly inferred.

Generally, OpenPose performs better under scenarios such as low contrast, multiple people and niche poses. The case of low contrast is shown in Figure 3.8. The rest examples are shown in Figure 3.9 and Figure 3.10.

The universally difficult scenarios are such that (1) Two people hug or interact closely with each other. (2) The poses with twisted limbs. (3) The niche perspectives. (4) The exaggerated

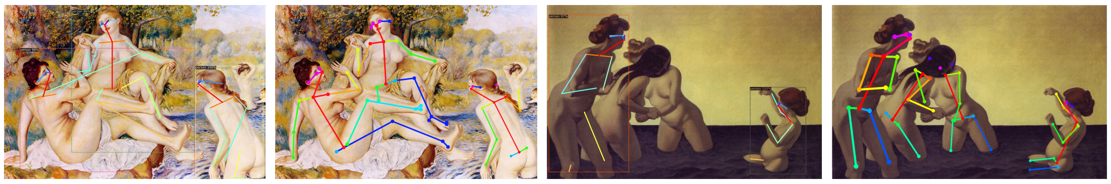


Figure 3.9: The outcome of DensePose (left) vs. OpenPose (right) under scenario - multiple people.

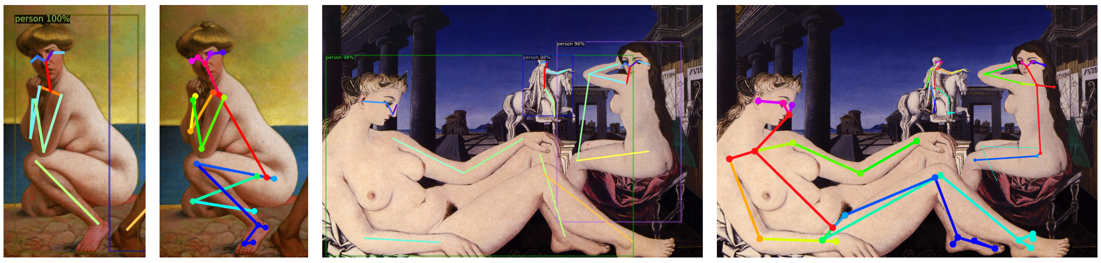


Figure 3.10: The outcome of DensePose (left) vs. OpenPose (right) under scenario - niche poses.

shapes. (5) the exaggerated body proportions. They are illustrated in Figure 3.11 in order. As demonstrated in Table 3.1, the inference of joints by the paintings of Tamara de Lempicka and Amedeo Modigliani both has the least accuracy due to their exaggeration of shapes and proportions. In contrast, Paul Delvaux's paintings have the highest accuracy by both OpenPose and DensePose, because most of the poses are resting standing ones with naturally hanging arms, which constitute easy cases for prediction.

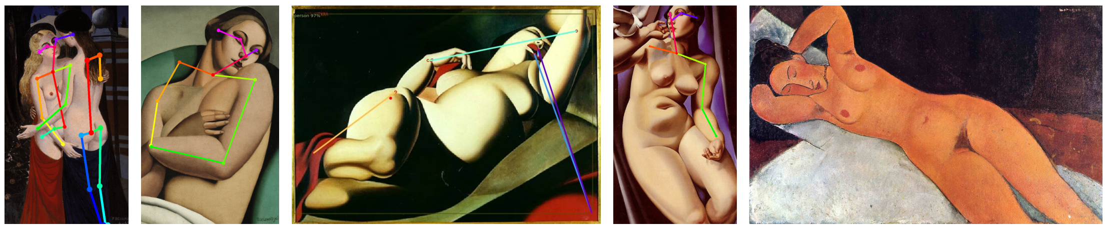


Figure 3.11: The difficult scenarios for both OpenPose and DensePose.

3.3.2 Body segments

The segments of DensePose come into two types: coarse segments and fine segments. The former are comprised of 15 annotations: (1) background, (2/3) Right and left hands, (4/5) Right and left feet, (6) Torso, (7/8) Right and left upper legs, (9/10) Right and left lower legs, (11/12) Right and left upper arms, (13/14) Right and left lower arms, and (15) Head. The latter further break down each segment from torso to head into two sub-segments, which consists of front and back ones respectively, which leads to a total of 25 annotations. Figure 3.12 illustrates the annotations of coarse and fine segments in colored masks and points.

Next, we will carry out the same accuracy test of DensePose for the selected 150 paintings for the 10 artists. The accuracy is calculated by the percentage of the correctly inferred coarse segments over the total visible segments. The visible segments are those which are drawn in the paintings and not occluded by other objects, whose total number is bounded by 14 (without background). The inference accuracy for each artist is shown in Table 3.2. The factors that impact inference are: (1) The number of people in the painting, as DensePose suffers from early commitment. (2) The occlusions of clothes, for which inference of upper and lower limbs has higher accuracy than that of torso. (3) The interactive people. (4) The niche poses. (5) The niche perspectives. (6) The

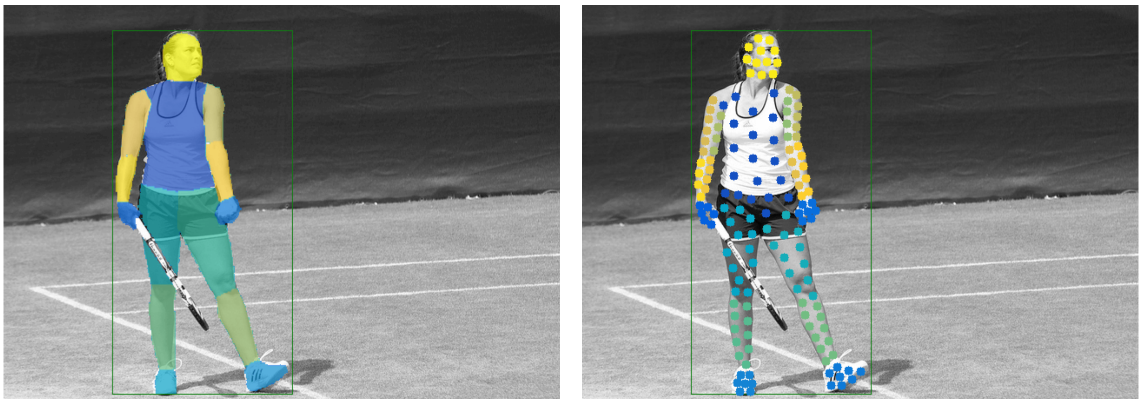

(a) Annotated segmentation masks colored according to the coarse segments.
(b) Annotated points colored according to the fine segments.

Figure 3.12: The annotations of DensePose

artistic effects, i.e., the colors and brushes of body segments. (7) The shapes of body segments.

	Total people	Total men	Total women	Total segments	Accuracy
Michelangelo	15	14	1	163	129 (79%)
El Greco	34	32	2	259	110 (42%)
Artemisia Gentileschi	21	6	15	182	108 (59%)
Pierre-Paul Prud'hon	15	7	8	196	127 (65%)
Pierre-Auguste Renoir	19	0	19	200	97 (49%)
Paul Gauguin	31	4	27	335	216 (64%)
Felix Vallotton	20	1	19	206	135 (66%)
Amedeo Modigliani	15	0	15	147	31 (21%)
Tamara de Lempicka	18	1	17	188	101 (54%)
Paul Delvaux	35	4	31	371	303 (82%)

Table 3.2: The accuracy of coarse segments by DensePose.

As shown in Table 3.2, the paintings of Michelangelo and Paul Delvaux have the highest accuracy, and that's because (1) Most of the poses are nude, hence without interference of clothes. (2) For Michelangelo, it is one pose per painting, and for Paul Delvaux, the people are distanced from each other. (3) All the poses from Michelangelo are sitting ones, and most poses from Paul Delvaux are standing ones, which form the most common two poses. (4) All of their poses are of a frontal view. (5) All of their poses are painted by nude color. (6) All of their poses follow the natural body proportions.

The examples for these two artists are shown in Figure 3.13. The first one is the 100% percent correctly inferred case for Michelangelo. The second one is also fully correct, but due to existence of the hanging robe, the legs cannot be inferred. The third one is a failure, and the potential reason might be that the torso is twisted and occluded by the right arm when viewed from aside, thus the torso is wrongly deduced, which further leads to the incorrect inference of lower limbs. The fourth one is the success case for Paul Delvaux, as most people painted by him are distanced between one another in order to create the lonesome ambience, the occlusion problem is largely avoided. The fifth one is the wrong case, as the lying pose constitutes one of the difficult poses. Even by rotating the lying pose 90 degrees to a standing pose, DensePose still fails to infer correctly. The potential reason might be that the relative spatial relationships between body segments are different from those of a natural standing pose. In Section 3.4.2, it will further illustrate that the training dataset of DensePose has only very few lying natural poses as well. Thus, the lying poses are under-represented during training, which might lead to incorrect inference during test. Moreover, the sitting person in the background is missing in inference, as DensePose doesn't recognize this as a person, which reflects the problem of early commitment of top-down methods.

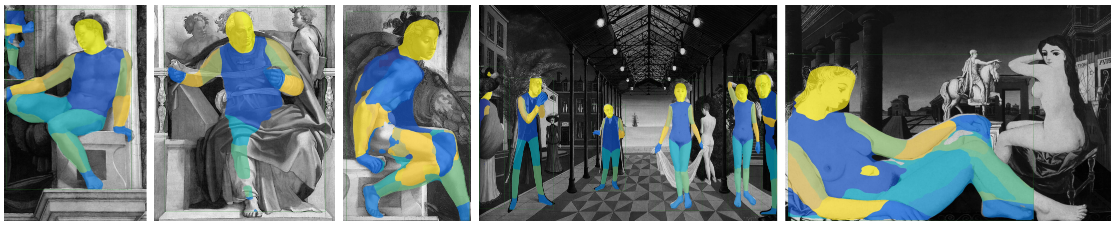


Figure 3.13: The inference of segments for Michelangelo (first three) and Paul Delvaux (last two).

Further shown in Table 3.2, the paintings of Amedeo Modigliani and Pierre-Auguste Renoir have the least accuracy. For Modigliani, the potential reasons might be that (1) Most poses are lying poses. (2) The body proportions are overly slender and elongated. For Renoir, the causes might be that (1) The bodies are painted by blobs of various colors. (2) Some are lying poses. The examples are shown in Figure 3.14. The first two are both failed cases for Modigliani: one is totally blank without any inference, the other is with wrong inference for torso, arm and right thigh. The third and fourth ones are failed cases for Renoir, and they might be due to his colored brushes, plus the twisted sitting pose and lying pose. The fifth one is a partially correct case, the sitting person is totally ignored, but for the standing pose, it is much easier to infer.

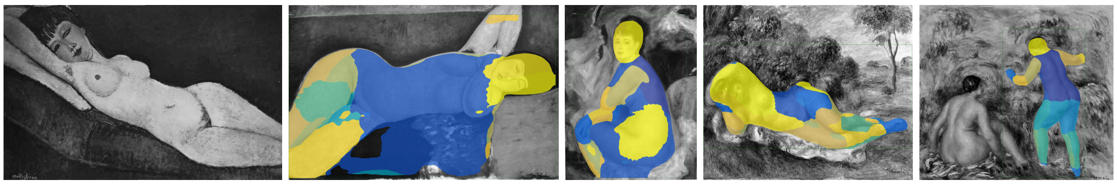


Figure 3.14: The inference of segments for Modigliani (first two) and Renoir (last three).

For the other artists, the examples are shown in Figure 3.15. For El Greco, the elongated body cannot be inferred for its segments. For Prud'hon, the standing pose viewed from aside is hard to infer, and especially, the torso is wrongly masked. For Gauguin, color contrast might impact inference. The man sitting closely behind and the woman hunching her back are totally missed. For Felix Vallotton, the intertwined arms are incorrectly inferred. For Lempicka, the niche perspective poses a challenge for inference. Moreover, the exaggerated body segments are themselves hard to infer, in which upper limbs have comparatively higher accuracy than that of torso and lower limbs.

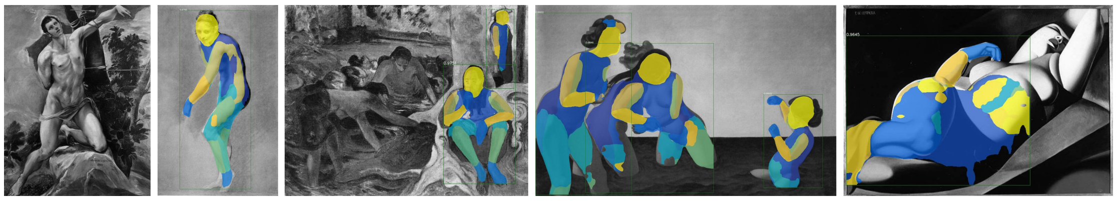


Figure 3.15: The inference of segments for El Greco, Prud'hon, Gauguin, Felix Vallotton and Lempicka (from left to right).

In summary, the root causes of these failed cases might be that (1) The training COCO dataset has lower density of natural poses, such as lying, sitting with back turned to observers, crowd of overlapped people, etc., as most of them are of one person doing sports or with only the upper torso captured. (2) The training data collects mostly the common people with common body proportions and height, which are neither too slender nor too plump.

3.4 Statistical analysis

For the selected datasets of natural Section 3.1 and artistic Section 3.2 poses, we will carry out statistical analysis for them both in order to illustrate the insight as to the difference between natural and artistic poses, and the common and niche poses, which might provide further explanation to the inference accuracy in Section 3.3, and pave the way for how we can select various datasets for specific purposes in later methodologies section. Section 3.4.1 will illustrate the joint distribution of natural and artistic poses, and Section 3.4.2 will illustrate the common and niche poses. With niche poses, it might explain why OpenPose and DensePose perform worse on the poses such as lying or sitting with back turned to observers.

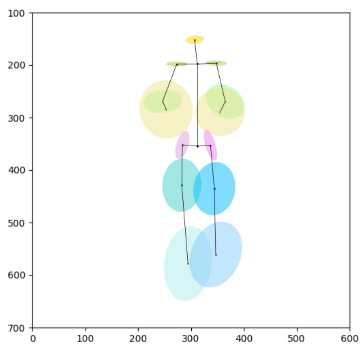
3.4.1 Elliptical distribution

For artistic poses, the paintings are imbalanced across gender. As shown in Table 3.1, classical artists mostly drew nude men, whereas modern artists mostly drew nude womenn. The imbalance tilts from men towards women after Impressionism, i.e., Pierre-Auguste Renoir. Specifically for classical artists, Michelangelo and El Greco mostly drew men. The exceptional case is Artemisia Gentileschi, who drew many women instead. As gender is roughly demarcated between classical and modern poses, we will hence illustrate the joint distribution for them respectively. Out of this mixed-gender reason, 60 pixels is used as a scaling reference instead, as it is the average value of 62 and 58. Moreover, all the inferred joints are used without differentiating their correctness. The reasoning is that the inference accuracy by OpenPose is roughly around 80%, which is satisfactory.

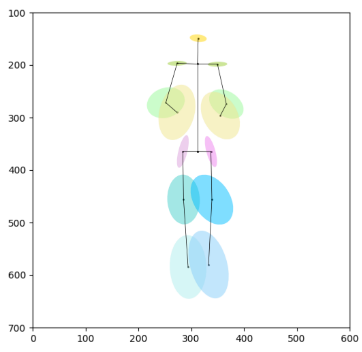
First, we will calculate elliptical distribution of joints, since it can give an overview of limb length and joint articulation. For elliptical distribution, keypoints are normalized in three steps:

1. Validate whether all the mandatory keypoints are detected. For artistic poses, nose, neck, and midhip must all exist. For natural poses, a full set of 15 keypoints must be present;
2. Rotate the whole pose to vertical position, so the spine is vertical;
3. For natural poses, use 62 pixels for men and 58 pixels for women as reference to scale head and limbs accordingly. For artistic poses, use 60 pixels for both.

The reason why the natural poses have stricter validation condition is that there are sufficient natural poses ready for analysis. To validate whether all joints exist can lead to more accurate results. Whereas, for artistic poses, it is not possible to collect enough poses with full joints. Thus, the validation is relaxed in order to have sufficient poses for analysis. After the first step of validity check, for classical poses, there are 90 valid poses left; for modern ones, 137 are valid. Figure 3.16 shows the final joint distribution.



(a) Elliptical distribution of the classical poses.

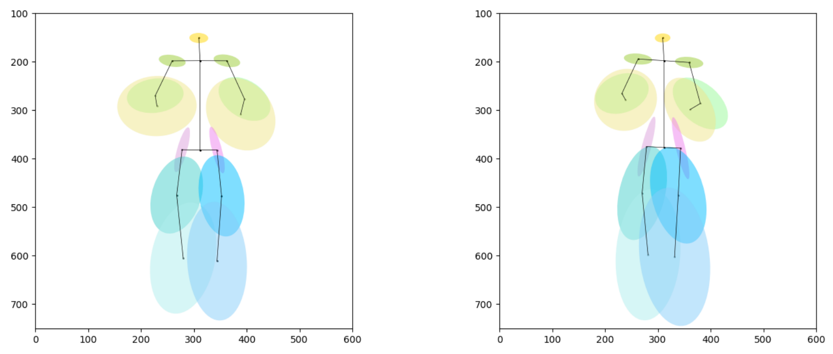


(b) Elliptical distribution of the modern poses.

Figure 3.16: The mean and 0.5 standard elliptical deviation for artistic poses.

As shown in Figure 3.16, different from the elliptical distribution of facial landmarks, the keypoints are prone to high variance of articulation. For the upper limbs, the wrists occupy a larger circle than the elbows, so do the elbows than the shoulders. Similarly, for the lower limbs, the areas of ellipses are in descending order: ankle > knee > hip. The reason might be that the outer joints swing over a larger circle than the inner joints. For the skeleton poses represented by black dots and their connecting lines, the black dots are the mean locations of joints during articulation. Figure 3.16 shows intuitively that the skeletons are similar for both classical and modern poses, as they are both with arms and legs hanging alongside the torso.

For natural poses, by filtering through the annotated caption of the COCO people dataset, there are 2294 images with only men, and 735 images with only women. Besides, for one image, there may exist multiple people. After the first step of validity check, for 2294 images of men, there are finally **653** men chosen. For 735 images of women, **222** women are selected for the mean and standard deviation analysis. The result is shown in Figure 3.17.



(a) Elliptical distribution of the COCO men.

(b) Elliptical distribution of the COCO women.

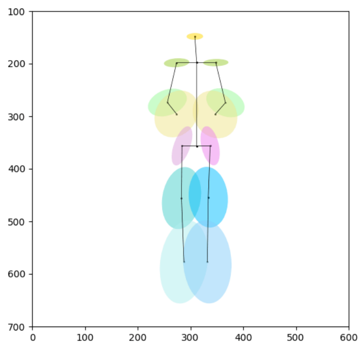
Figure 3.17: The mean and 0.5 standard elliptical deviation for natural poses in the COCO people dataset.

As shown in Figure 3.17, natural poses illustrate the same pattern as observed in artistic poses that there is more variation in the position of the outer joints. Compared with artistic poses, natural poses tend to be more varied with wider movement of joints. The reason might be that (1) For pose analysis, the natural poses are comprised of more people, whereas the artistic poses are hand-picked and limited to 75 images for each category. (2) The natural poses are formed by a wide variety of daily activities and sports, whereas the artistic poses are staged and hence restrained to certain poses.

Thus, we want to further explore artistic poses as to whether they are more varied, given (1) all the paintings of the selected 10 artists, and (2) all the paintings of the genre "nude painting (nu)". For all the paintings of the 10 artists, there are 96 for Michelangelo, 151 for El Greco, 17 for Artemisia Gentileschi, 101 for Pierre-Paul Prud'hon, 318 for Pierre-Auguste Renoir, 267 for Paul Gauguin, 134 for Felix Vallotton, 226 for Amedeo Modigliani, 64 for Tamara de Lempicka, and 121 for Paul Delvaux. The total is 1495 paintings with 1912 recognized people. For all the nude paintings, there are in total 1375 paintings with recognized 1180 people. As shown in Figure 3.18, provided with the full set of artistic poses, the joint articulation is still very constrained compared with natural poses, which further supports the claim that the poses in the paintings are staged that lead to limited movement of joints.

3.4.2 Hierarchical clustering

Second, we will carry out hierarchical clustering for both natural and artistic poses, as it can further illustrate the common and niche poses in detail, which can to some extent explain why OpenPose and DensePose perform better or worse for some poses. For hierarchical clustering, keypoints are normalized in three steps:


(a) Elliptical distribution of all the paintings of the selected 10 artists.

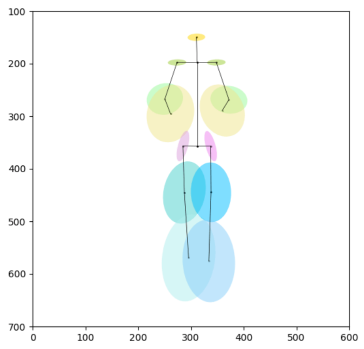

(b) Elliptical distribution of all the nude paintings.

Figure 3.18: The mean and 0.5 standard elliptical deviation for artistic poses of an extended full set of paintings.

1. Validate whether all the 6 torso keypoints are detected, i.e., neck, right and left shoulders, midhip, right and left hips;
2. Rotate the whole pose to a standing-up position, with the spine being vertical;
3. Calculate the inner angles for all the triplets of joints.

In total, there are 13 such joint triplets, i.e., 1 triplet of (nose, neck, midhip), 6 triplets of the right body: (shoulder, neck, midhip), (elbow, shoulder, neck), (wrist, elbow, shoulder), (hip, midhip, neck), (knee, hip, midhip), (ankle, knee, hip), and 6 symmetric triplets of the left body. The inner angles of each pose on the 2-dimensional plane are treated as a 13-dimensional vector representative of each pose. These pose vectors are used for agglomerative hierarchical clustering, in which each pose starts in its own cluster, which will be merged with other clusters if their pairwise Euclidian distance is the smallest. Finally, the clustering process will result in one cluster containing all the poses, which can be depicted in a dentrogram. In dendrogram, similar poses will be connected with each other by shorter distances, whereas different poses will be connected with each by longer distances. In previous works, the agglomerative hierarchical clustering is also used to analyse the poses in Aby Warburg's bilderatlas [34]. As only angles of joints are concerned, so the fixed length is adopted for all the bones, where the length of spine is 70 pixels, and the length of limbs and neck is 30 pixels. With only angles present, the information as to whether the person stands or lies is lost or can be deduced from angles implicitly. For the missing joint pairs, the smallest positive number 5e-324 will be used, otherwise errors will occur in hierarchical clustering.

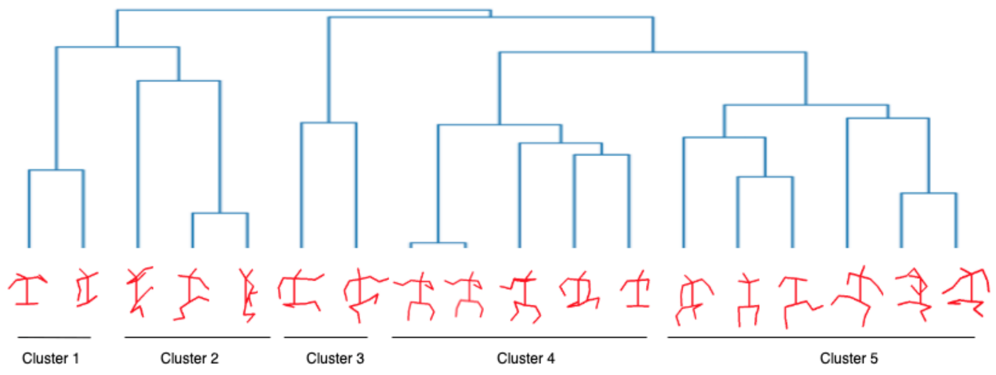


Figure 3.19: All 18 poses drawn by Michelangelo from 15 paintings.

Take Michelangelo as example, there are 18 poses extracted from 15 paintings of him. After hierarchical clustering with pre-determined 5 clusters, Figure 3.19 shows the result in the form of dendrogram. The distance between a pair of poses is denoted by the route traced in the tree in order to connect them. As shown, 3 major clusters are differentiated from each other by the orientation of poses. Cluster 2 and 3 are composed of poses facing right, cluster 4 consists of poses facing straight, and cluster 5 contains poses facing left. The stretch of limbs also plays an important role, as cluster 3 and 4 contain more stretched poses, whereas cluster 2 has all the curled-up poses. Moreover, the completeness of the detected joints influences the recognized difference of poses as well. In cluster 1, it contains only partially detected poses with missing legs. In summary, the hierarchical clustering of the pose vectors can tell the difference of poses in: (1) the orientation, (2) whether the limbs are stretched or compressed, and (3) the completeness of the joints.

Compared to Michelangelo, Gauguin tends to draw more standing people. As shown in Figure 3.20, cluster 3 consists of all the front-facing standing poses whose legs are stretched and parallel to each other, and cluster 2, 4 and 5 mostly contain the sitting poses facing from right to left. Cluster 1 stands out, as it is the only pose whose arms are thrown upward into the air.

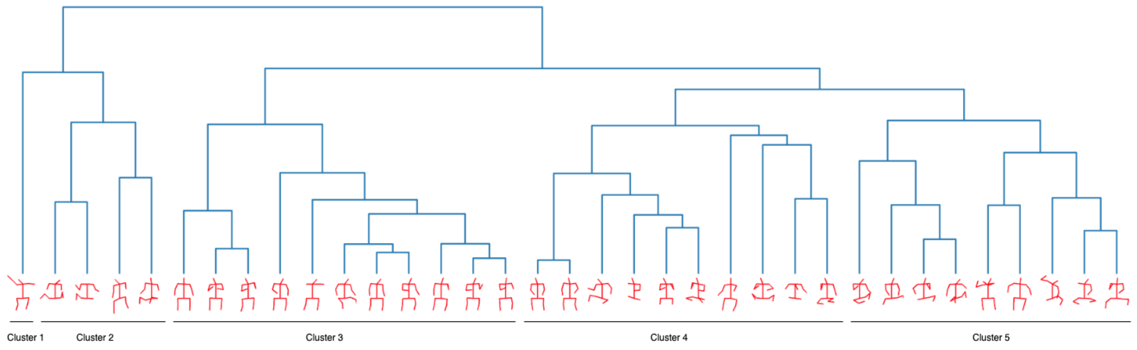


Figure 3.20: All 35 poses drawn by Paul Gauguin from 15 paintings.

When the number of clusters is set to 10 in order to generate the hierarchical clustering for all the artists, the outcome shows that there are no clear boundaries between classical and modern artists, and furthermore, there are no specific characteristics typical of each individual artist, that is one cluster for one artist. Out of 10 clusters, there are 2 outstanding clusters with only a small fraction of poses. These poses are shown in Figure 3.21, where the left image shows the original pose, and the right image shows the normalized poses with all limbs equal to 30 pixels. The normalized hip width is equal to the length of two limbs, which seems a bit wide, but it can better illustrate the poses with crossed legs. The commonality is that they are all standing poses with the arms thrown upward. This pose is a rarity both for the classical and modern ones: in classical times, the pose of opening arms appears only in the religious paintings as in those of El Greco and Artemisia Gentileschi. In modern times, this pose mostly comes with the lying nude women as in those of Amedeo Modigliani.

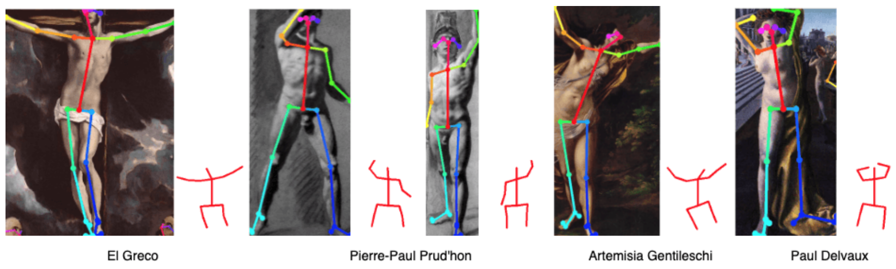


Figure 3.21: The standing poses with the arms thrown upward into the air.

In the same vein, we want to know whether there exist outlying niche poses in the COCO dataset for both men and women. We carried out hierarchical clustering for them separately, and Figure 3.22 shows the rare poses for men, Figure 3.23 illustrates the rare poses for women. As shown in Figure 3.22, the male poses with more stretched arms, i.e., swinging back and forth, or the ones with more twisted legs are quite rare. These niche poses are also common in their shooting perspectives, in which they are shot either from behind, aside, top or bottom. Moreover, the lying poses barely appear for men.

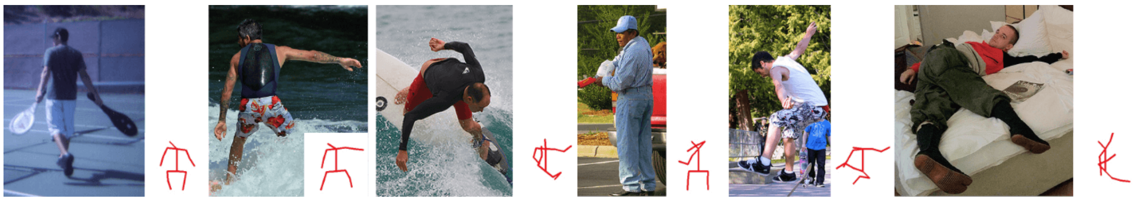


Figure 3.22: The niche natural poses in the COCO dataset for only men.

For women, the niche poses are those with twisted or overly stretched legs, as shown in the first row of Figure 3.23, or those with arms thrown upward, as shown in the second row of Figure 3.23.

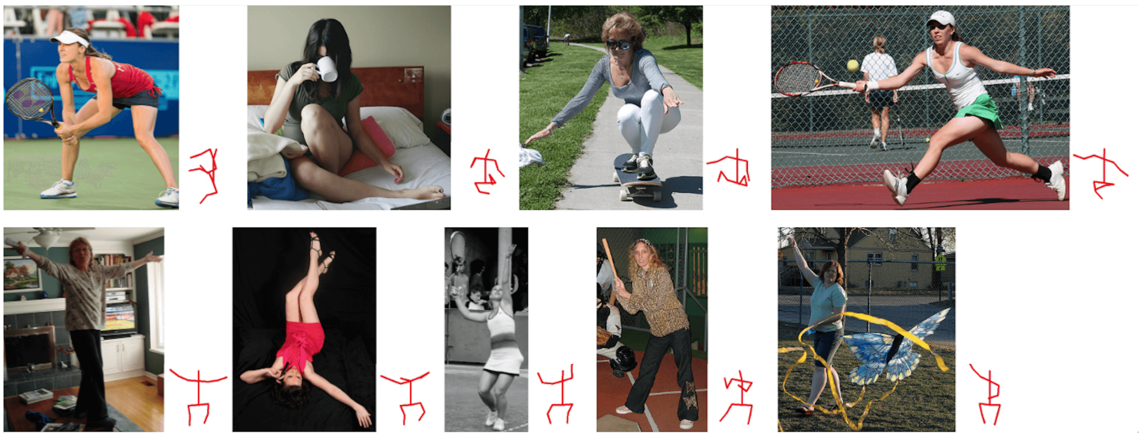


Figure 3.23: The niche natural poses in the COCO dataset for only women.

In summary, the poses with arms stretched upward are quite rare for both artistic and natural poses. In classical paintings, they appear as a religious posture. In modern paintings, they often come with nude women stretching their arms. In natural poses nowadays, they appear in women's hugging or lying poses, or in sports. It might be assumed that if trained with more niche poses and niche perspectives from natural poses, OpenPose and DensePose might also perform better at inference of artistic poses.

3.5 Datasets overview

As a whole, the COCO people dataset is used for natural poses, and the "Painter by Numbers" dataset is used for artistic poses. They form the basis of the training data for the implementation of the geometry-aware style transfer, except for some pragmatic adjustments made in the datasets to train different CycleGAN models. The reasoning is that (1) For the training of CycleGAN, it needs more than 15 nude paintings with a uniform artistic style. Thus, all the nude paintings from Impressionism or Expressionism can be considered as the best candidate training data of artistic poses, as they form the majority of nude paintings, which are shown in Figure 3.4. (2) For the training of CycleGAN, the goal of style transfer is to transfer from natural poses to artistic poses, rather than from with clothes to without clothes, and the colors of clothes might be distractive during style transfer. Thus, it is better to choose the nude or near-nude natural poses. After skimming through the COCO people dataset, the surfing men and women are finally selected as the training data of natural poses. (3) In order to illustrate the expected outcome of a geometry-aware style transfer, the paintings of Amedeo Modigliani are used during training, as the poses drawn by him have the extremely elongated head, torso and limbs, so that the shape-changing outcome is easy to observe.

Moreover, the chosen datasets are further split in a gender-specific manner for natural and artistic poses. The reasoning is that (1) There exists an imbalance of male and female nude paintings drawn by classical and modern artists. Classical artists mostly drew nude men, whereas modern artists mostly drew nude women. This imbalance tilts from men towards women after

Impressionism, i.e., Pierre-Auguste Renoir. (2) From the statistical analysis of the datasets, it is found that men and women tend to have different poses, and physically they are also differently built. In [54], it is found that poses can embed gender differences. Thus, the computational features are calculated for men and women separately. As to how to compute the computational features gender-wise, Section 4.1 will explain it in detail.

Based on the aforementioned reasons, we have generated different datasets for different usages. The overview of these gender-specific datasets and their corresponding usages are shown in Table 3.3.

Dataset	Gender	Images	Usage
COCO men	Men	2294	Statistical analysis in Section 3.4
COCO women	Women	735	Statistical analysis in Section 3.4
5 Classical artists	Mostly men	75	Statistical analysis in Section 3.4
5 Modern artists	Mostly women	75	Statistical analysis in Section 3.4
COCO surfing women	Women	67	CycleGAN-based style transfer in section 4.2
Nude paintings of Impressionism	Women	244	CycleGAN-based style transfer in section 4.2
Paintings of Amedeo Modigliani	Women	335	Warping-based style transfer in Section 4.3

Table 3.3: The overview of the gender-specific datasets and their corresponding usage scenarios

Generally, for natural and artistic poses, they are split into corresponding gender datasets. It is worth mentioning that during the training of each CycleGAN model, the gender-specific datasets are paired with each other, in which the COCO surfing men are paired with the nude paintings of Renaissance, whereas the COCO surfing women are paired with the nude paintings of Impressionism. The nude paintings of Renaissance are chosen, because Renaissance strides across multiple stages from Early Renaissance, Northern Renaissance, High Renaissance to Mannerism (Late Renaissance), which contains the most nude male paintings in a uniform artistic style. For the detail of each dataset, it will be explained in the corresponding section as well.

Chapter 4: Methodologies

In order to realize geometry-aware style transfer, two methods will be proposed. The first method is based on CycleGAN, as in Section 4.2, which is aimed to transfer shape, color and texture automatically in one step without warping. The second method is by warping, as in Section 4.3, which is carried out in two steps: (1) For shape, the body segments are morphed from a natural pose to an artistic pose by manually warping their keypoints to match the desired artistic pose. (2) For color and texture, it is imposed from an artistic pose to a natural pose by style transfer. For both methods, the shapes of a pose are represented by a combination of all its body segments arranged in a T-pose. Each body segment is represented as a rectangular or square contour. The reasons why contours are used are that (1) The abstracted contours can to some extent mitigate the occlusion and perspective issues that are present in 2D DensePose inference to get more accurate representations of body segments. (2) The neural network can only deal with regular-size images such as rectangles or squares as training data, whereas the inferred body segments by DensePose are irregular. Thus, in Section 4.1, we will first illustrate how the contours of a pose are calculated.

4.1 Contours

In this section, the contours will be extracted based on the body segments inferred by DensePose. The contours will be calculated based on the assumption that the body segments are convex and symmetrical, and be scaled based on the assumption that men and women have different head sizes. The assumptions based on which to calculate the contours are explained in Section 4.1.1. In Section 4.1.2, normalization will be carried out to scale different-size contours to a uniform size, and to translate any pose constituted by contours to a uniform T-pose. By normalization, different poses with different sizes can be compared with each other on the same scale, where poses are either sitting or standing, either near in the foreground or far in the background. Finally, in Section 4.1.3, the average contours will be further calculated to illustrate the contours of natural poses and artistic poses respectively.

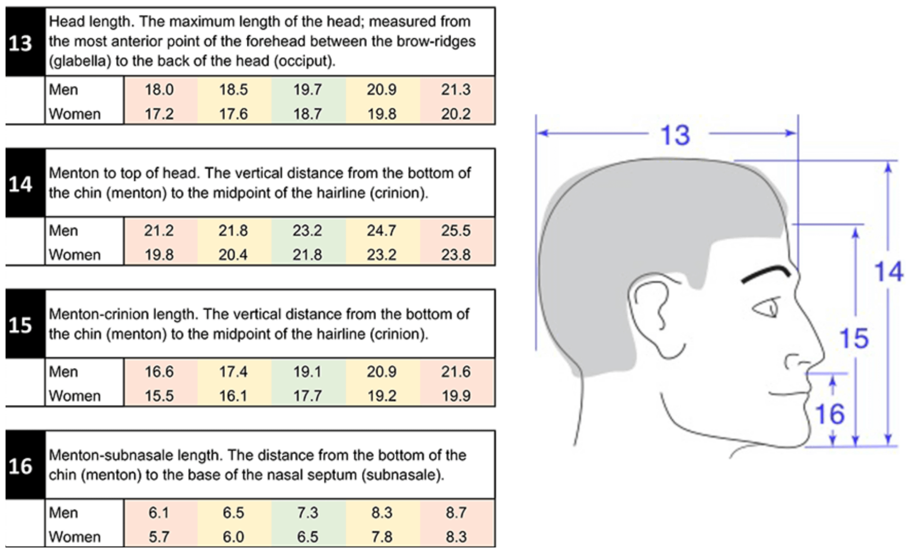
4.1.1 Assumptions

In general, there are four assumptions based on which we normalize the body segments:

1. The head size is different for men and women. The height of head is equal to the vertical distance between the nose and above the chest.
2. The body proportion is the same for men and women;
3. The body segments are convex, which can be abstracted as rectangles;
4. The body segments are symmetrical between left and right. For a specific segment, it is symmetrical around its centroid.

Firstly, head size is used as a normalized reference, based on which length of limbs and area of segments can be scaled accordingly.

In the 14th row of Figure 4.1, it shows statistically the height of human head for men and women respectively. The 3rd column in green is the median value. The 2nd and 4th columns in yellow denote the 5% and 95% values. The 1st and 5th columns in red stand for the 1% and 99%



13	Head length. The maximum length of the head; measured from the most anterior point of the forehead between the brow-ridges (glabella) to the back of the head (occiput).				
Men	18.0	18.5	19.7	20.9	21.3
Women	17.2	17.6	18.7	19.8	20.2

14	Menton to top of head. The vertical distance from the bottom of the chin (menton) to the midpoint of the hairline (crinion).				
Men	21.2	21.8	23.2	24.7	25.5
Women	19.8	20.4	21.8	23.2	23.8

15	Menton-crinion length. The vertical distance from the bottom of the chin (menton) to the midpoint of the hairline (crinion).				
Men	16.6	17.4	19.1	20.9	21.6
Women	15.5	16.1	17.7	19.2	19.9

16	Menton-subnasale length. The distance from the bottom of the chin (menton) to the base of the nasal septum (subnasale).				
Men	6.1	6.5	7.3	8.3	8.7
Women	5.7	6.0	6.5	7.8	8.3

Figure 4.1: The height of human head by centimeters, which is illustrated in the 14th row.

values. We will use the median height as a reference, that is 23.2 centimeters for men and 21.8 centimeters for women. Thus, measured by pixels, **62** pixels are used as the standard head height for men, and **58** pixels are used as the standard head height for women during normalization in order to retain the same proportion, as shown in Equation 4.1. 62 pixels are chosen out of convenience, as the generated image of the normalized pose in Section 3.4.1 and Section 3.4.2 is of 600×600 dimension. 58 pixels are calculated from Equation 4.1. With the standard head sizes set as 62 and 58 pixels, the full body can be captured by the normalized pose in the generated image.

$$\frac{Standard\ head\ height\ of\ women}{Standard\ head\ height\ of\ men} = \frac{21.8}{23.2} \tag{4.1}$$

For an input pose from any image, the scale factor is calculated by Equation 4.2. Other limbs are scaled accordingly by the same scale factor. Finally, all poses whose measuring unit is pixels can be compared relatively.

$$Scale\ factor = \frac{Standard\ head\ height\ in\ pixels}{Actual\ head\ height\ in\ pixels} \tag{4.2}$$

Since it is only explicitly defined that the height of head is the vertical distance from the bottom of the chin to the midpoint of the hairline, we extend as one minor assumption that the height of head is also equal to the vertical distance between the nose and above the chest. It is assumed out of convenience because only this distance is known in the inferred keypoints both by OpenPose and DensePose, and this distance needs not to be realistic, as the height of head is only a reference according to which the whole body can be scaled. For the remaining three assumptions, they are used as precondition to carry out normalization in Section 4.1.2 to extract the contours of a pose.

4.1.2 Normalization

Normalization is carried out based on the aforementioned four assumptions to extract the contours of a pose in a standard T-pose. Thus, various poses can be compared with each other with regard to shapes. For the standard T-pose, the Vitruvian Man drawn by Leonardo da Vinci is adopted as a reference for both natural and artistic poses, because (1) It has been used as the drawing canon for artistic poses. (2) It was summarized based on observations of natural poses. The rules are based on [1], which are referenced as follows:

1. The length of the outspread arms is equal to the height of a man;
2. From below the chin to the top of the head is one-eighth of the height of a man;

3. From above the chest to the top of the head is one-sixth of the height of a man;
4. The maximum width of the shoulders is a quarter of the height of a man;
5. The distance from the elbow to the tip of the hand is a quarter of the height of a man;
6. The length of the hand is one-tenth of the height of a man;
7. The distance from the elbow to the armpit is one-eighth of the height of a man;
8. The root of the penis is at half the height of a man;
9. From below the foot to below the knee is a quarter of the height of a man;
10. From below the knee to the root of the penis is a quarter of the height of a man;

Figure 4.2 visualizes the aforementioned rules and maps out each keypoint on the Vitruvian Man. Pinpointed to the corresponding keypoints, "above the chest" is equal to the keypoint of neck. The "elbow" is the keypoint of elbow, and the "armpit" is the keypoint of shoulder. the keypoint of wrist is calculated by subtracting "the length of the hand" from "the distance from the elbow to the tip of the hand". "The root of the penis" is the keypoint of midhip. The "knee" is the keypoint of knee, and the keypoint of ankle can be inferred by subtracting an absolute margin from the distance "from below the foot to below the knee", as the position of ankle was not explicitly stipulated in the canon.

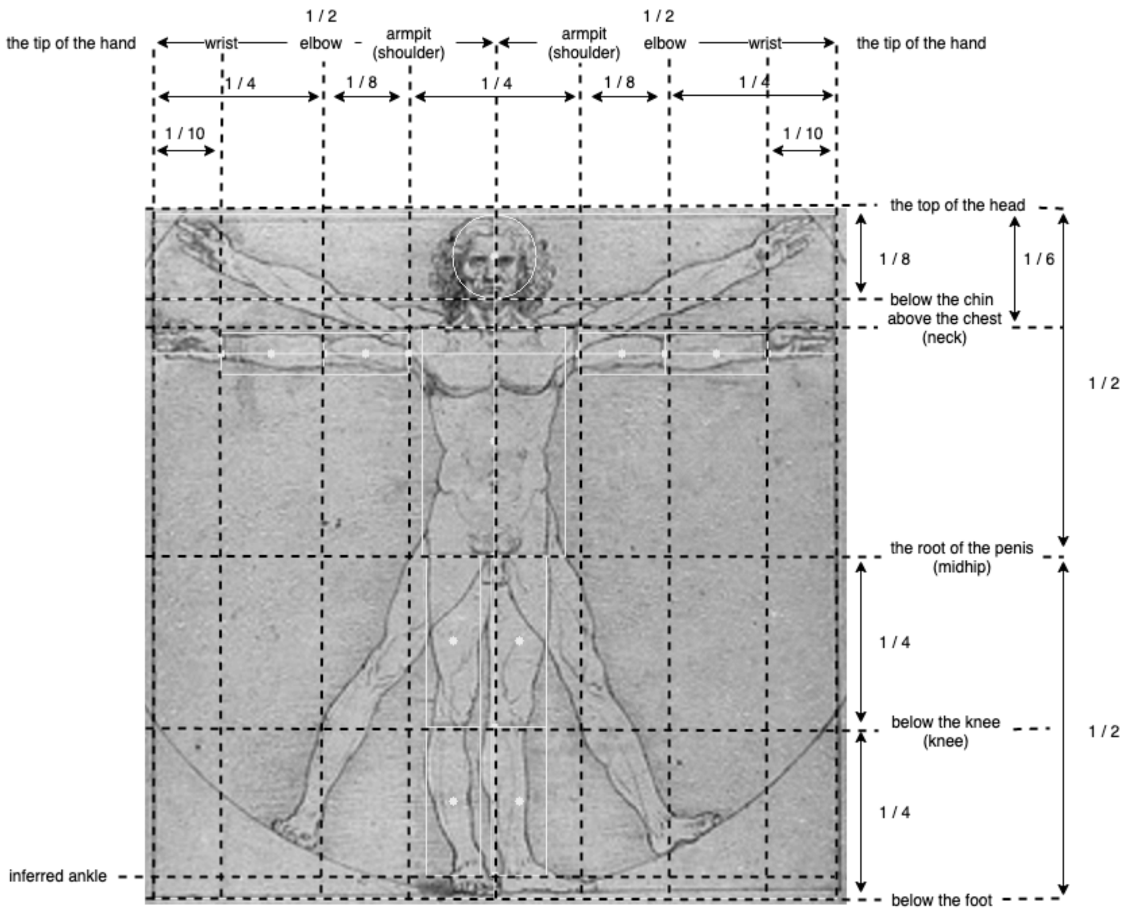


Figure 4.2: The canon of the Vitruvian Man.

Compared with scaling the limbs that are represented by the distance between keypoints, to scale the segments, i.e., areas rather than lines, we use centroids of segments as the pivot around which to scale surrounding pixels within the boundary of corresponding segments. The centroid of head is the average point of an area occupied by the head segment, which is later mapped on the Vitruvian Man as the midpoint between "below the chin" and "the top of the head". The centroid of torso is the midpoint between the keypoints neck and midhip, which is mapped on the Vitruvian Man also as the midpoint between the neck and the midhip. The centroids of arms and legs are the midpoints between the keypoints wrist and elbow, elbow and armpit, hip and knee, knee and ankle, which are mapped on the Vitruvian Man as the midpoints of the corresponding joints. All the mapped centroids are shown in Figure 4.2 as white dots. To differentiate, the centroid can be either an average point of an area, or a midpoint between two keypoints, around which each segment can be scaled area-wise during normalization. On the other hand, the midpoint on the Vitruvian Man is a fixed point on the standard image, to which each normalized segment can be translated, so that the centroid of each segment can be superimposed on the midpoint of the Vitruvian Man.

Furthermore, the baseline Vitruvian Man is referenced by a fixed dimension, i.e., (624×624) in pixels, with the male head height being fixed by **62** pixels, and the female one by **58** pixels. The scale factors can be calculated in the same way by Equation 4.1 and Equation 4.2 for men and women respectively. With centroids and scale factors determined, each segment can be scaled to the normalized size.

Next, we will dip into the desired result of normalization and the corresponding process. The output of normalization is a T-pose figure superimposed on the Vitruvian Man with 10 segments, i.e., head, torso, upper and lower arms, upper and lower legs. The normalization process for one person is based on the input from: (1) The keypoints by OpenPose. (2) The segments and bounding box by DensePose. The first step is to match each person's OpenPose data with DensePose data, as they are extracted by different processes, and stored in separate files. For one painting, one OpenPose file and one DensePose file will be generated, and mapping will be further carried out to determine in this painting: (1) For only person, whether its OpenPose data is matched with its DensePose data. (2) for multiple people, which index of the OpenPose data is matched with that of the DensePose data. We use the centroid of keypoints to determine its matched bounding box. If the centroid falls within the bounding box, then the keypoints are matched with this bounding box, describing the same person. In this way, the OpenPose data is matched with its corresponding DensePose data. This matching method works under the precondition that any two people's bounding boxes are not overlapping, otherwise it will lead to mismatched data. For example, if two centroids fall into the intersection of two bounding boxes, it will lead to many-to-many mapping that one centroid belongs to two bounding boxes, and one bounding box contains two centroids. If many-to-many mapping occurs, we will discard the data during matching. After matching, we will validate the data by checking whether the keypoints of torso are all detected. The keypoints of torso are neck, left and right shoulders, midhip, and left and right hips. If the torso is only partially detected, we will discard the data of this person, and skip to the next one. Third, for each person, we rotate the keypoints and subsequently the segments, so the spine is vertical, which is exactly the same as in the previous keypoints normalization. Fourth, the centroid of head is calculated in order to rotate head to vertical position. Similarly, the midpoints of arms and legs are calculated based on keypoints, so upper and lower arms and legs can be rotated around its corresponding midpoint to horizontal and vertical position respectively. At this stage, the orientation of each segment has been completed, and they are ready to be composited further into T-pose. At last, each segment is translated to a standard T-pose image with the centroid of head and the midpoints of torso and limbs at the fixed coordinates. The head is scaled to the predetermined height, and the scale factor is calculated in order to scale other segments. After the this final scaling phase, the normalized T-pose is complete. To summarize, the normalization steps are as follows:

1. Match the OpenPose data with its corresponding DensePose data for each person in the painting;
2. Validate whether the keypoints of the torso are all detected;
3. Rotate the whole pose to vertical position, so the spine is vertical;
4. Rotate each segment separately to vertical or horizontal position, as required by a T-pose;
5. Translate each segment to its fixed coordinate by translating its centroid to the midpoint in the standard Vitruvian Man;
6. Scale all the segments to their normalized shapes by the calculated scale factor;

7. Dilate all the segments to their tightest-fitting rectangles;
8. Symmetrize all the segments, so that they are symmetrical between left and right. For a specific segment, it is symmetrical around its centroid.

Take Michelangelo and Paul Delvaux as example, Figure 4.3 and Figure 4.4 illustrate the input keypoints and segments data in the left and middle columns, and the output normalized segments in the right columns. From the two normalized poses, it shows that the body drawn by Michelangelo are more inflated and full to the brim of the underneath contour. Compared with Paul Delvaux's standing pose, Michelangelo's sitting pose exposes three major issues: (1) The right thigh is occluded by the right arm, which leads to the partial segment with a gap in-between in the normalized pose. (2) The thighs are not fully stretched out due to the sitting pose and the viewing perspective. Thus, the corresponding normalized thighs are both shorter. (3) The lower left arm is retracted a little behind the torso, which is further away from observers. Thus, the corresponding normalized lower left arm is shorter and thinner, compared with the lower right arm.

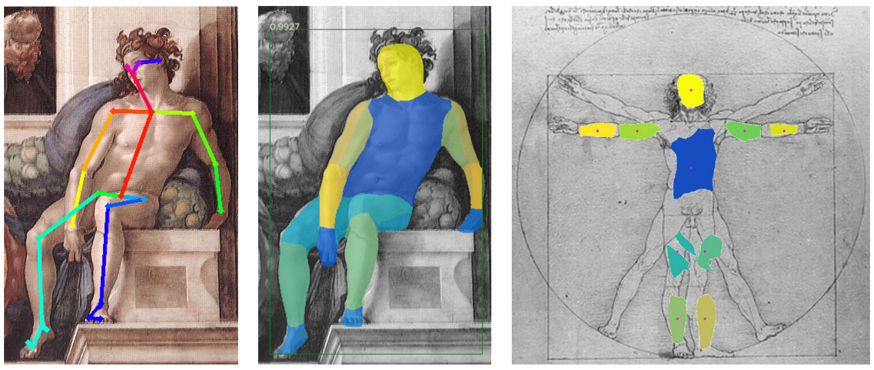


Figure 4.3: Man drawn Michelangelo. Left: Keypoints by OpenPose. Middle: Segments and bounding box by DensePose. Right: Normalized DensePose.

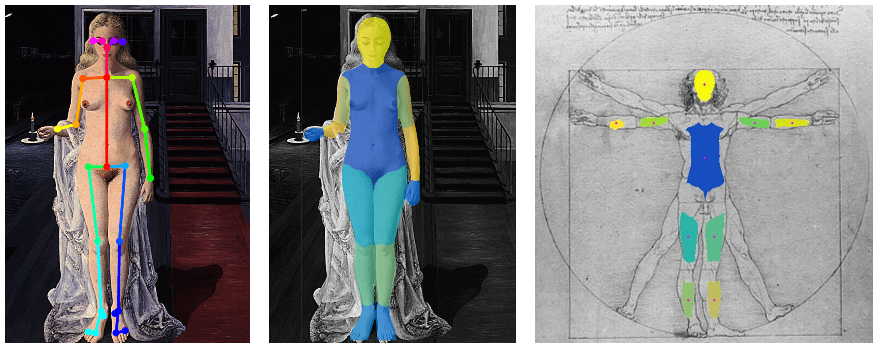


Figure 4.4: Woman drawn by Paul Delvaux. Left: Keypoints by OpenPose. Middle: Segments and bounding box by DensePose. Right: Normalized DensePose.

To resolve the above issues, first, we will tend to use the frontal-view standing poses instead of others, which can reduce the interference of the viewing perspectives during normalization. Second, all segments will be dilated to their tightest-fitting rectangles in order to fill the inner gaps and holes that result from occlusion. Third, symmetry will be enforced, so the right thigh is of the same shape as the left thigh, so are the calves, upper and lower arms. For each segment, it will be further rendered symmetrically around its centroid, with the longest radius mirrored on both sides. In order to further reduce noise caused by the wrong inference of DensePose, the outlier pixels of each segment will be removed before dilation.

The final normalized poses are shown in Figure 4.5 after dilation and the enforced symmetry. We select as best as possible only the standing poses, except for Michelangelo, as there are no standing poses drawn by him in the dataset. The corresponding paintings are shown in Figure 4.6, in which two male figures are chosen from the classical category, and two female figures are chosen from the modern category. The normalized poses show intuitively that the male figures have longer arms and legs than those of women. But the size of the arm and leg rectangles may not be accurate due to the out-of-the-plane rotations during normalization. Moreover, Figure 4.6 is formed by putting the dilated segments back to their original coordinates. With Figure 4.5 and Figure 4.6 combined together, it demonstrates that the extra steps of dilation and symmetrization during normalization can mitigate the issues of occlusion and perspectives in a rudimentary manner. What is worth noting are the dilated thighs from Michelangelo's painting, which are dilated and symmetrized disproportionately due to the fact that this sitting pose distorts the thighs while being viewed from front. We might get the right size when viewing from above, but as they can be only viewed in a two-dimensional space, their true size cannot be deduced simply by dilation and symmetrization. It also indicates that in order to get the near-correct normalized segments of bodies based on the paintings, it is better to use standing pose by a frontal view.

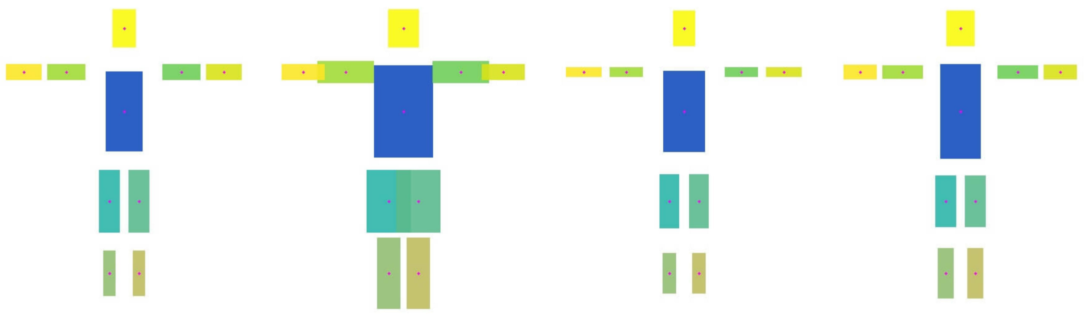


Figure 4.5: The normalized DensePose after dilation and the enforced symmetry.

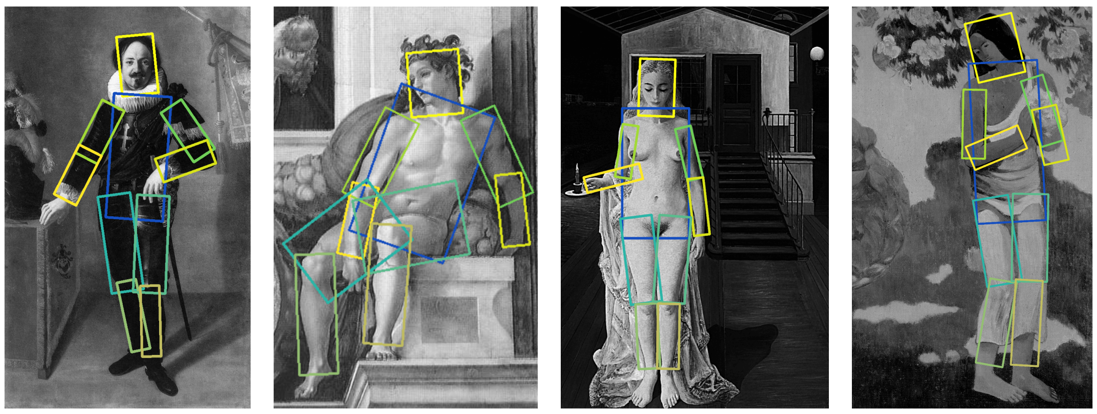


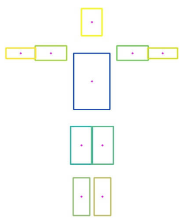
Figure 4.6: Put the dilated segments back to their original coordinates. The paintings from left to right are from Artemisia Gentileschi, Michelangelo, Paul Delvaux and Paul Gauguin.

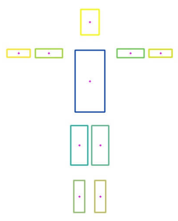
4.1.3 Average Contours

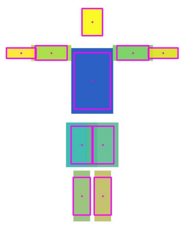
Based on normalization in Section 4.1.2, we can calculate the contours for one pose segment-wise, based on which the average contour can be further calculated. For the average contour, it is the average width and height for each segment respectively, i.e., head, torso, upper limbs, and lower limbs. The advantages of average contour are three-fold: (1) If some segments are occluded and hence only partially inferred in one pose, it can be compensated from other poses. (2) The segment contour of each artist can give an intuitive impression as to how each artist tends to draw human bodies, slender or inflated. (3) The segment contour can be compared with each other to further explore whether there exists significant difference between the drawing styles for each artist. (4) The artistic segment contour can be compared with that of the COCO men and women contour to analyze whether the artistic contour conforms to or deviates from the natural contour.

The precondition of calculating the average contours for each artist or style is based on the assumption that for the same artist, they tend to draw the poses of men and women consistently, and the contours are only different gender-wise. For example, for Modigliani, he tends to draw women in elongated faces and limbs, which do not pertain to a specific woman. For Michelangelo, he tends to draw men with muscular arms and legs, which are characteristic of his drawing style that do not stick to a particular man.

First, we will calculate the average contour for Michelangelo and Paul Delvaux respectively, as the former stands for the classical men, and the latter represents the modern women. For Michelangelo, 5 poses from 5 paintings are chosen, as only men were drawn, and most of their segments are correctly inferred. For Paul Delvaux, 15 poses from 12 paintings are selected, as only women were drawn, and all of them stand in the foreground. Figure 4.7 shows the average contours for these two artists respectively. As illustrated, for Michelangelo, the torso is shorter and wider, and the upper and lower limbs are more elongated and inflated. Comparatively, for Paul Delvaux, the figure is more slender.


(a) The average contour of Michelangelo.


(b) The average contour of Paul Delvaux.


(c) The 2nd pose of Figure 4.5 imposed on the contour of Michelangelo.

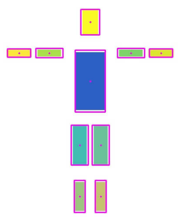

(d) The 3rd pose of Figure 4.5 imposed on the contour of Paul Delvaux.

Figure 4.7: The DensePose contour of artistic poses.

Next, we want to see whether each specific pose sticks to or deviates from the contour. After superimposing one normalized pose from Michelangelo and Paul Delvaux on their contours, which are now highlighted in bright magenta thick lines in Figure 4.7, it is observed that (1) Paul Delvaux conforms to his contour more closely. (2) For Michelangelo, the torso, thighs and calves overflow their contour boundaries. The reason might be that the normalized pose has inference error with the thighs over-inflated due to sitting pose and perspective.

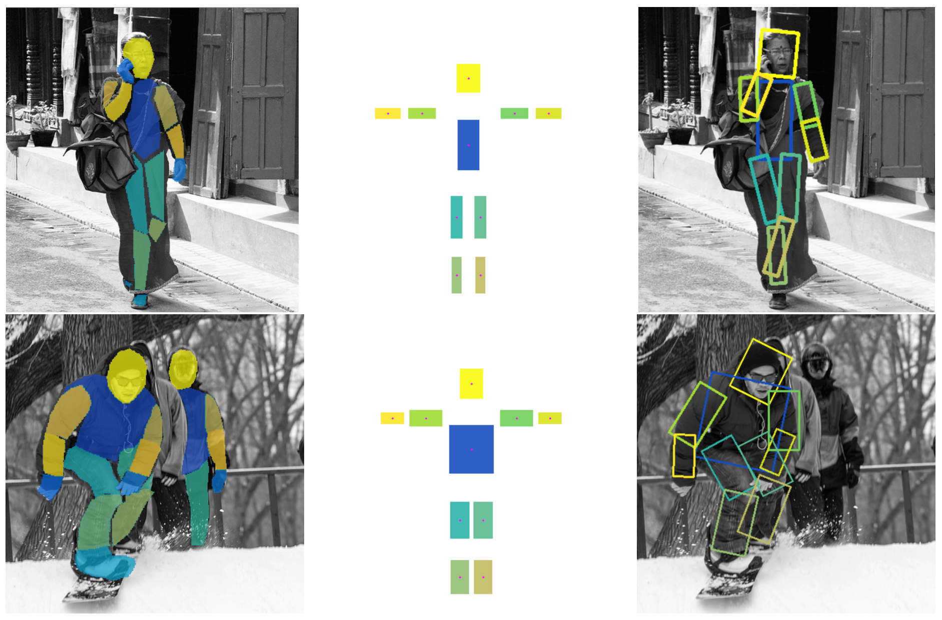
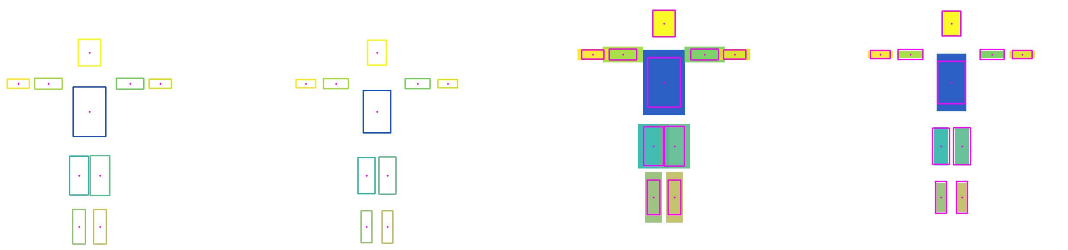


Figure 4.8: One of the annotated DensePose for the COCO men and women (left). The normalized pose (middle). Put the dilated segments back to their original coordinates (right).

Finally, we want to know the contour of the COCO men and women respectively. The same COCO men and women data is used as in the analysis of joint distribution from Section 3.4.1. The validity check is kept the same, that is to check whether all 15 keypoints have positive scores and are valid. Besides, an additional precondition is that for each pose, the annotated DensePose must exist and be valid. In this way, 144 men and 150 women are finally chosen to generate the average contour for men and women in natural poses. Before calculating the contour, the same normalization process is executed with dilation and symmetry. Figure 4.8 shows the annotated DensePose, the normalized pose and the original pose superimposed with the normalized segments from left to right, for men and women respectively.

Figure 4.9 shows the average contours of the COCO men and women respectively. The contour of natural poses are smaller segment-wise compared with that of artistic poses in Figure 4.7. For men, it is shorter and less muscular in natural poses. For women, it seems to be shorter and less slender with respect to torso and limbs. After superimposing the same pose from Michelangelo and Paul Delvaux on the natural men and women contours, the result is shown in Figure 4.9. It can be seen that Michelangelo tends to exaggerate the muscles of men, as every segment is inflated outside the brim. On the contrary, Paul Delvaux tends to draw the women more slender than their counterparts in the natural poses, as the torso and the limbs shrink a bit width-wise within the borders.



(a) The average contour of the COCO men.

(b) The average contour of the COCO women.

(c) The 2nd pose of Figure 4.5 imposed on the contour of the COCO men.

(d) The 3rd pose of Figure 4.5 imposed on the contour of the COCO women.

Figure 4.9: The DensePose contour of natural poses.

In summary, the mean contours can give us an intuitive view of artistic and natural poses, focusing on height and width of body segments. Moreover, the potential usage of the contours are twofold: (1) Use it as the bounding box for each segment in order to further enhance the inference accuracy of DensePose. (2) Use it to guide the geometry-aware style transfer, e.g., in warping by providing the length of limbs, or in CycleGAN-based style transfer by providing the body segment patches as shape signal. They will be explained in detail Section 4.2 and in Section 4.3 respectively.

4.2 CycleGAN

As in [81] and [55], CycleGAN is used for style transfer. The advantage of CycleGAN-based style transfer is that in order to transfer the stripes from zebra to horse, unpaired training datasets of horses and zebras can be used. The usage of unpaired datasets can solve the challenge that paired datasets of natural and artistic poses are difficult to find. However, the disadvantage of CycleGAN comes with the unpaired datasets. It poses challenges for CycleGAN as to where to look at when the training data are unpaired. On a coarse-grained scale, the challenge is to match the source pose with the target pose, and to match the source background with the target background. On a fine-grained scale, the challenge is to match each source body segment with each target body segment, e.g., from head to head, from torso to torso, from arms to arms, and from legs to legs. Moreover, it poses challenges as to what to transfer. Asides from color and texture, we also want CycleGAN to transfer shapes from an artistic pose to a natural pose. As in [81], CycleGAN failed to transfer from cat to dog, from apple to orange with respect to shapes. Thus, in this section, we would like to explore how we can convey the signal of shapes into the network of CycleGAN, as the shapes of body segments can be represented by standard contours.

Three variants of CycleGAN models will be implemented, which are the vanilla CycleGAN model as a baseline in Section 4.2.1, the CycleGAN model with patch-wise loss in Section 4.2.2, and the CycleGAN model with coutour-wise loss in Section 4.2.3. The reason why we start from the vanilla CycleGAN model is that we want to focus on the implementation as to how the shape signal can be conveyed in CycleGAN, as the contours of the natural and artistic poses can be matched with each other in the training data and used as the shape signal. If the shape signal can be successfully conveyed, then we will experiment with more sophisticated CycleGAN architectures in order to achieve better results in style transfer of color and texture.

4.2.1 Baseline

Two domains of training images are used. For artistic poses, 244 images of the Impressionism nude paintings are used. For natural poses, 67 images of the COCO surfing women are used. The reasoning is that (1) For the training of CycleGAN, it needs more than 15 nude paintings with a uniform artistic style. Thus, all the nude paintings from Impressionism are chosen. (2) For the training of CycleGAN, the goal of style transfer is to transfer from natural poses to artistic poses, rather than from with clothes to without clothes, and the colors of clothes might be distractive during style transfer. Thus, the COCO surfing women are selected, as they are with the least clothes. Other activities have also been considered such as swimming, surfing is chosen because it already contains sufficient training images.

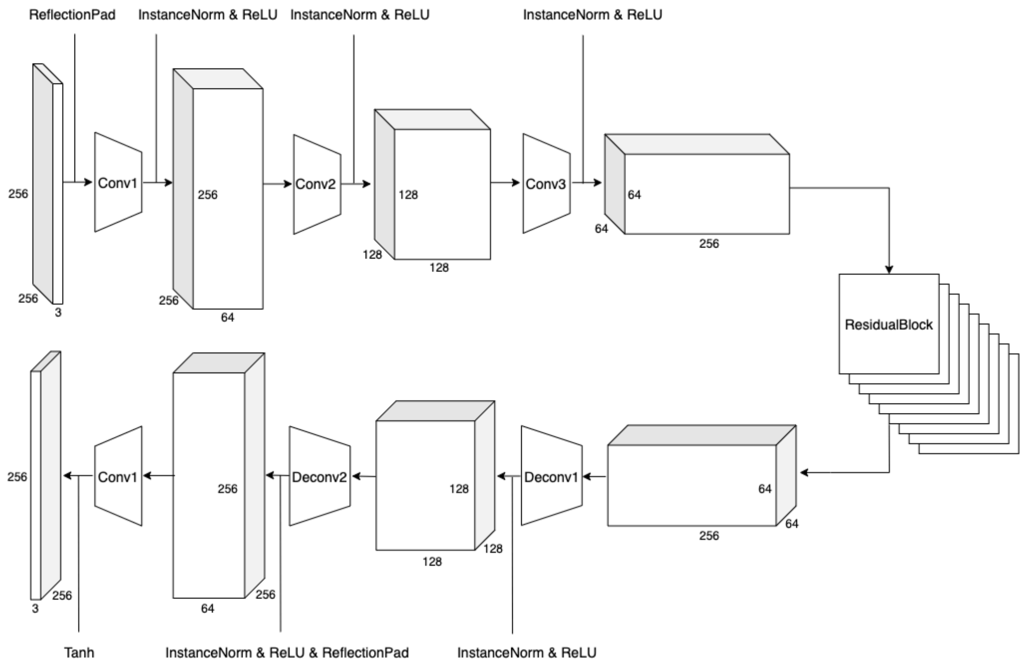


Figure 4.10: The architecture of our vanilla CycleGAN's generator, $G_{X\to Y} = G_{Y\to X}$.

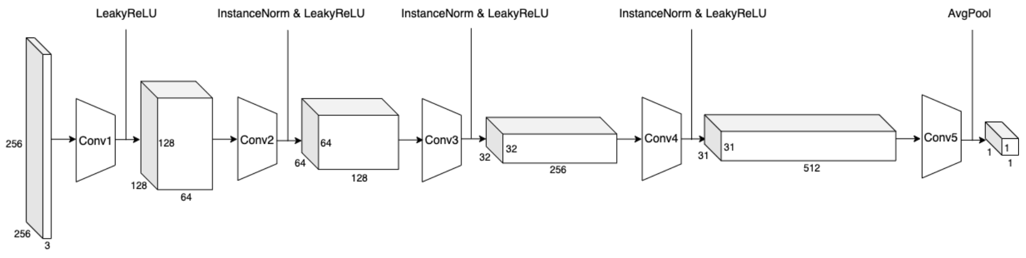


Figure 4.11: The architecture of our vanilla CycleGAN's discriminator, $D_X = D_Y$.

Next, we will implement a vanilla CycleGAN to perform style transfer. By "vanilla", it is with regard to the architecture of the neural network and its loss function. The reasoning is that (1) We can foremost have a baseline of CycleGAN to check out its results and limitations. (2) We can then explore how to transfer shapes in CycleGAN in its simplest form, as it is assumed that the current architecture and the loss function of neural networks only affect the texture synthesis. Now, we will explain the architecture and the loss function one by one. For architecture, CycleGAN consists of two components: generator and discriminator. For generator, CycleGAN contains two generators: one is from domain X to Y, denoted by $G_{X\to Y}$, and the other is from domain Y to X, denoted by $G_{Y\to X}$. Likewise, for discriminator, CycleGAN has two discriminators: one is for domain X, denoted by D_X, and the other is for domain Y, denoted by D_Y. The two generators and discriminators are domain-wise symmetric, so the two generators share the same neural network architecture, and so do the two discriminators. It is not necessary that the same architecture is used for both generators and discriminators, but it is common. The architecture of our vanilla CycleGAN's generator is shown in Figure 4.10, and the architecture of discriminator is illustrated in Figure 4.11.

For generator, the input and output are both colored images of dimension 256×256. The network is formed by 3 convolutional layers and 3 deconvolutional layers that are symmetric around 9 residual blocks. Each residual block is comprised of 2 convolutional layers with the setting that the reflection pad is $(1, 1)$, the padding is $(1, 1)$, and the stride is $(1, 1)$ to ensure that the input and output tensors of this residual block will not change their dimensions. Instance normalization is used instead of batch normalization, so the normalization operation is only carried out per image rather than across a batch of images in order to reduce the noise and generate high-quality fake images as in the design of the original CycleGAN [81].

For discriminator, the network is formed by 5 convolutional layers with average pooling at the end to produce a floating point number between 0 and 1, in which 0 is the label of fake images, and 1 is the label of real images. LeakyRelu is used instead of ReLU, because LeakyRelu has a slight slope for negative values, whereas ReLU treats non-positive values all as zero. Thus, LeakyReLU can avoid saturation during the training of a GAN network to some extent.

Generator			
Encoder	Conv2d_1	kernel size	7
		stride	1
	Conv2d_2 Conv2d_3	kernel size	3
		stride	2
		padding	1
	ReflectionPad2d	padding	3
	InstanceNorm2d	momentum	0.1
Residual Block	Conv2d_1 Conv2d_2	kernel size	3
		stride	1
	ReflectionPad2d	padding	1
	InstanceNorm2d	momentum	0.1
Decoder	ConvTranspose2d_1 ConvTranspose2d_2	kernel size	3
		stride	2
		padding	1
		output padding	1
	Conv2d_1	kernel size	7
		stride	1
	ReflectionPad2d	padding	3
	InstanceNorm2d	momentum	0.1
Discriminator			
Network	Conv2d_1 Conv2d_2 Conv2d_3 Conv2d_4	kernel size	4
		stride	2
		padding	1
	Conv2d_5	kernel size	4
		stride	1
		padding	1
	InstanceNorm2d	momentum	0.1
	LeakyReLU	negative slope	0.2
Training			
Epoch	Total epochs	200	
	Decay epoch	150	
Optimizer	Adam	learning rate	0.0002
		betas	(0.5, 0.999)
	Scheduler	LambdaLR	Linearly decay from 1 to 0
Data augmentation	CenterCrop		

Table 4.1: The parameters of the vanilla CycleGAN

Currently, we will not change the architecture of our vanilla CycleGAN in order to focus on transferring patches and shapes of body segments, rather than enhance the visual results of color and texture. Table 4.1 gives an overview of the parameters of our vanilla CycleGAN, which will be kept the same throughout our experiment. Next, we will delve into the loss function designed for our vanilla CycleGAN. The total loss function comes with two parts: the generator loss and the discriminator loss. For the generator loss, it is comprised of three parts: the identity loss, the GAN loss and the cycle loss, as denoted in Equation 4.3. For the discriminator loss, it is comprised of two parts: the real loss and the fake loss, as shown in Equation 4.4. All of the losses are domain-wise symmetric. So, in the following equations, only the losses for one domain X are shown. For notations, the real samples from domain X and domain Y are denoted as $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$ respectively.

$$\mathcal{L}_{generator} = \mathcal{L}_{identity} + \mathcal{L}_{GAN} + \mathcal{L}_{cycle} \tag{4.3}$$

$$\mathcal{L}_{discriminator} = \mathcal{L}_{real} + \mathcal{L}_{fake} \tag{4.4}$$

For the identity loss, it is formulated in Equation 4.5. The identity loss means that if a real sample y is fed as input to generator $G_{X\to Y}(y)$, what is generated as output should be equal to the input of this very y.

$$\mathcal{L}_{identity} = \frac{1}{n}\sum_{i=1}^{n} \left|G_{X\to Y}(y_i) - y_i\right| \tag{4.5}$$

For the GAN loss, it is written in Equation 4.6. The GAN loss means that given a real sample x, the generated fake y cannot be distinguished by the discriminator D_Y from real samples of domain Y.

$$\mathcal{L}_{GAN} = \frac{1}{n}\sum_{i=1}^{n} \Big(D_Y\left(G_{X\to Y}(x_i)\right) - 1\Big)^2 \tag{4.6}$$

Equation 4.7 shows the cycle loss. The cycle loss is to ensure that to transfer a real sample x from domain X to domain Y, and then back to domain X should be consistent with itself.

$$\mathcal{L}_{cycle} = \frac{1}{n}\sum_{i=1}^{n} \left|G_{Y\to X}\left(G_{X\to Y}(x_i)\right) - x_i\right| \tag{4.7}$$

For the real loss, it is calculated by Equation 4.8. The real loss means that the discriminator is capable of recognizing the real images in each domain.

$$\mathcal{L}_{real} = \frac{1}{n}\sum_{i=1}^{n} \left(D_X(x_i) - 1\right)^2 \tag{4.8}$$

For the fake loss, it is illustrated in Equation 4.9. The fake loss means that the discriminator is able to detect the fake images correctly.

$$\mathcal{L}_{fake} = \frac{1}{n}\sum_{i=1}^{n} \Big(D_Y\left(G_{X\to Y}(x_i)\right)\Big)^2 \tag{4.9}$$

As a whole for domain X and Y, the loss can be expressed as a sum of the total generator loss and the total discriminator loss. The following equations of losses are simplified notations of the above-defined loss functions. For example, $Loss\_G\_identity\_X$ denotes Equation 4.5. These notations are used to record the losses throughout the loss charts shown in the results in Chapter 5.

$Loss\_G$ is the total generator loss, which can further break down into the identity loss $Loss\_G\_identity$, the GAN loss $Loss\_G\_GAN$, and the cycle loss $Loss\_G\_cycle$ as follows:

$$Loss\_G = Loss\_G\_identity + Loss\_G\_GAN + Loss\_G\_cycle$$

For each of the above three losses, it is a sum of two symmetric components:

$$\begin{aligned} Loss\_G\_identity &= Loss\_G\_identity\_X + Loss\_G\_identity\_Y \\ Loss\_G\_GAN &= Loss\_G\_GAN\_X2Y + Loss\_G\_GAN\_Y2X \\ Loss\_G\_cycle &= Loss\_G\_cycle\_XYX + Loss\_G\_GAN\_YXY \end{aligned}$$

$Loss\_D$ is the total discriminator loss, $Loss\_D\_X$ and $Loss\_D\_Y$ are the discriminator losses for domain X and Y respectively, which can be written down as follows:

$$Loss\_D = Loss\_D\_X + Loss\_D\_Y$$

In the next step, we would like to experiment with the design of CycleGAN regarding the following aspects:

1. Train with a patch-wise loss for the discriminators and generators in Section 4.2.2, so the source patch and the target patch can be paired;

2. Train with a contour-wise loss for the generators in Section 4.2.3, which is based on the normalized DensePose body segments.

4.2.2 Patch-wise loss

In this section, we will only experiment to improve the performance of the vanilla CycleGAN for the style transfer from the COCO surfing women to the Impressionism nude paintings, as these two domains have sufficient training images. The experiment will be carried out in two steps involving a patch-wise loss for the discriminators and the generators respectively, which are listed as below:

1. For the discriminators, PatchGAN is used;

2. For the generators, PatchNCE loss is used.

We will implement PatchGAN for the discriminators. The reasoning is that, as the learning capabilities of the discriminators of the vanilla CycleGAN have not been tapped to the full, PatchGAN might help to adjust how the loss is calculated for the discriminators. As described in [81], PatchGAN can introduce a patch-wise loss per image, rather than the loss for the image as a whole, thus the spatial relationships can be preserved during the training. The architecture of the discriminators remains the same, as in Figure 4.11. What PatchGAN does is to split the input image into multiple patches along the x and y axis and feed these patches into the discriminator. For example, if the input image's shape $(Batch, Channel, Height, Width)$ is equal to (1, 3, 256, 256), after being split into 8×8 patches along each axis, the shape of the patches fed into the discriminator becomes (64, 3, 32, 32), where 64 is the number of total patches, and 32 is the height and width of each patch which is the outcome of 256 divided by 8, 3 is the RGB channel.

Next, we will try to implement a PatchNCE loss for the generators to guide their training as well. The purpose of PatchGAN and PatchNCE loss is to enforce the spatial relationships between the patches of a image. Thus, the spatial statistics of a image can be kept as content, while changing the color and texture of it as style. As described in [55], PatchNCE is a patch-wise contrastive loss. The name originates from InfoNCE loss [73], in which the encoder and the autoregressive networks are trained jointly to maximize mutual information between the input and output patches, and NCE stands for Noise-Contrastive Estimation. The goal of PatchNCE is to associate the patches with each other that come from the same location of the images in two different domains, and to dissociate the patches that are spatially different. As a result, take the horse-to-zebra conversion as an example, a horse's head in domain X can be matched with a zebra's head in domain Y, and it is however contrasted with other areas, such as those of body and legs. The expectation is that it can help to solve one of the challenges of style transfer in unsupervised learning: how can the source patch be transferred stylistically only to the target patch, as raised in Section 2.4.2. For the architecture of the autoregressive network, a two-layer MLP (Multi-Layer Perceptron) network is used, as in SimCLR [10]. This MLP network learns to project all the patches to a shared embedding space, where the matched patches are targeted to be equal to each other, and the contrasted patches are aimed to be unequal.

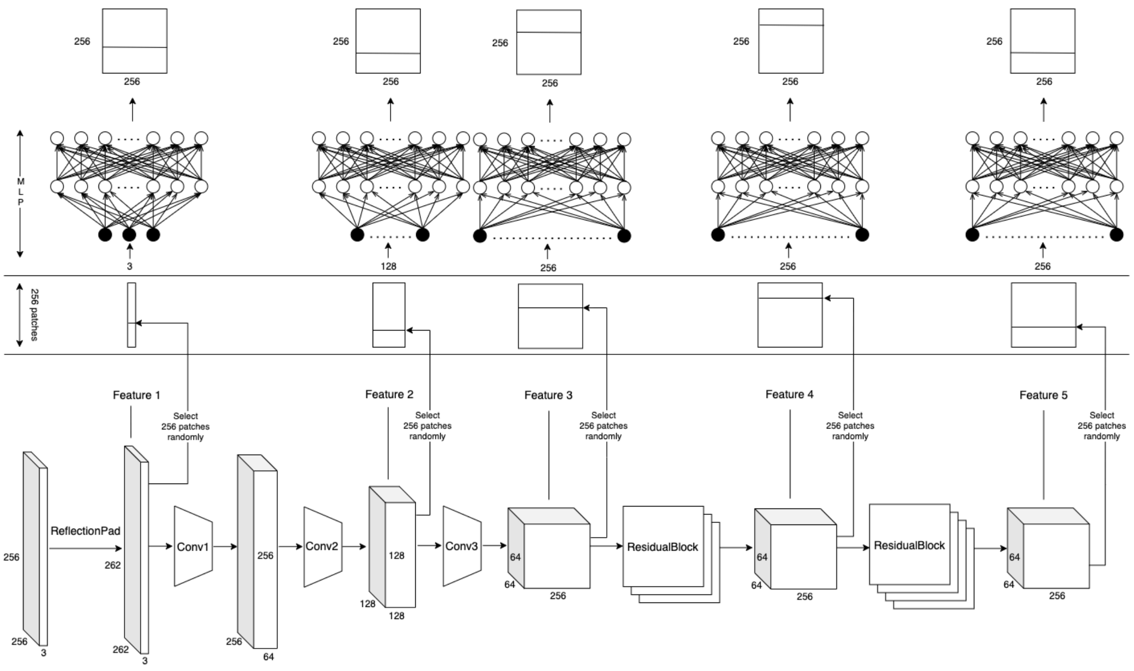


Figure 4.12: The two-layer MLP networks to project randomly selected 256 patches to a shared embedding space for each feature.

The MLP networks are trained jointly with the generator network, and more specifically, the encoder part of the generator. The architecture is shown in Figure 4.12. To explain, first, 5 features are selected from the outputs of 5 layers, each deeper in the encoder network. As the layers deepen, the features go from low-level to high-level. For example, for one location, i.e., $(x,\ y)$, of feature 1, it purely represents the information of one pixel in RGB. Whereas for one location of feature 2 till 5, it represents the condensed information involving larger and larger patches, when this location is projected back onto the original input image. For each feature, 256 locations are randomly sampled, in which one location stands for one patch. Afterwards, these 256 patches are fed into a MLP network whose output will be a (256, 256) tensor. This tensor can be regarded as a representation of all the 256 patches out of a shared embedding space. To calculate the PatchNCE loss for the generator $G_{X\to Y}$ from domain X to Y, it is in two steps. First, a real image x of domain X and its counterpart, a fake image $G_{X\to Y}(x)$ of domain Y, are fed into the encoder part of the generator $G_{X\to Y}$ one by one. For each feature, for the real image, the patches are randomly selected in the first pass. For the fake image however, the same patches are chosen as those of the real images in the second pass, so the same location of two different domains can be bridged with each other. These patches are then used by the MLP networks to retrieve the representative tensors from the embedding space. The PatchNCE loss is hence calculated as a cross-entropy loss of the tensors of a real image and a fake image combined, which is denoted as $Loss\_G\_NCE\_X$. Next as the second step, the same process is carried out for the inputs which are a real image y of domain Y and a fake image $G_{X\to Y}(y)$ of domain Y, which is similar to identity loss in a patch-wise manner. The result is denoted as $Loss\_G\_NCE\_Y$.

In summary, the PatchNCE loss of the generator $G_{X\to Y}$ is comprised of two components:

$$Loss\_G\_NCE = Loss\_G\_NCE\_X + Loss\_G\_NCE\_Y$$

For $Loss\_G\_NCE\_X$, it means the NCE loss between a real sample x of domain X and a fake sample $G_{X\to Y}(x)$ of domain Y, where x is the input. For $Loss\_G\_NCE\_Y$, it denotes the NCE loss between a real sample y of domain Y and a fake sample $G_{X\to Y}(y)$ of domain Y, where y is the input. Finally, the total generator loss can be rewritten as below:

$$Loss\_G = Loss\_G\_identity + Loss\_G\_GAN + Loss\_G\_cycle + Loss\_G\_NCE$$

It is worth noting that we only use one-way of the PatchNCE loss, which is for the generator from domain X to Y, as we are now interested in converting a photo to an Impressionism painting. The PatchNCE loss for the generator from domain Y to X is symmetrical as well. However, we will not use it for now in the total generator loss.

4.2.3 Contour-wise loss

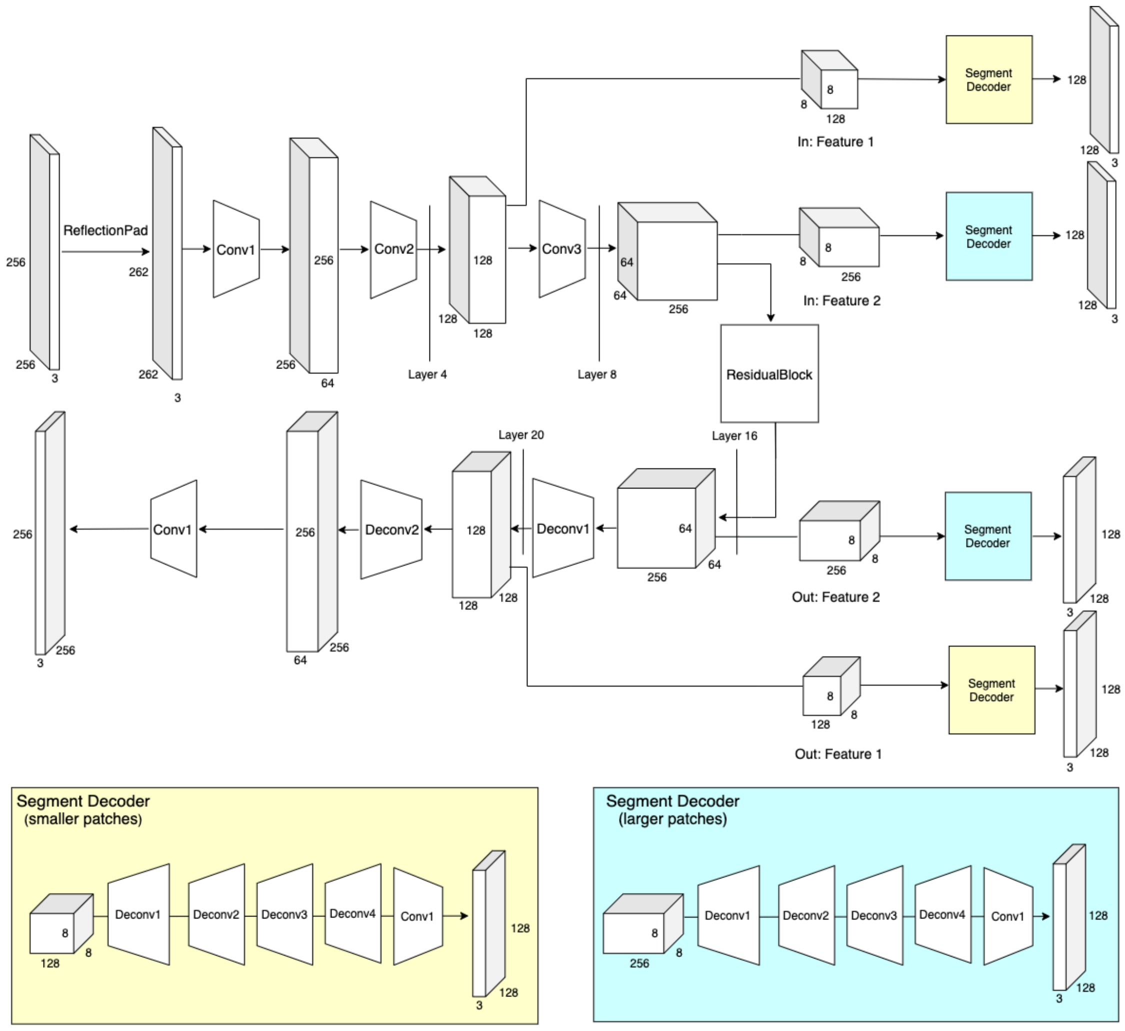


Figure 4.13: The network of segment decoder.

Finally, based on the CycleGAN with PatchGAN for discriminator and a PatchNCE loss for generator, we will explore how we can convey a signal of shape into the network. Each body segment can be represented by a square patch centered around its centroid with its edges along the x- and y-axis of the training image. The centroid is the midpoint of each segment. Moreover, there are two kinds of patches: small patches (16×16) and large patches (32×32). The purpose is to bridge the small patches with the corresponding large patches in a loss function in order to enlarge each body segment twice the original size. If this loss function works, the next step is to convey the right scale factor by the paired small and large patches of different sizes, which are proportional to the aimed scale factor.

The architecture of our shape-aware CycleGAN is built upon the generator of the vanilla CycleGAN with four extra segment decoder networks. The segment decoder network comes into two types: one is for the smaller patches, and the other is for the larger patches. Both of them are based on the features extracted from the encoder and decoder of the generator respectively. The architecture is shown in Figure 4.13.

The segment decoder for a smaller patch is built upon the features extracted from a shallower layer, whereas the segment decoder for a larger patch is based on the features extracted from a deeper layer. The reasoning is that through a shallow layer, the image is downsized from (256×256) to (128×128). Though a deep layer, the image is further downsized from (128×128) to (64×64). As a result, a (32×32) patch in the original image can be mapped to a (16×16) patch in a shallow layer, and to a (8×8) patch in a deep layer. Reversely, for a (8×8) patch in a deep layer, it can be mapped to a (32×32) patch in the original image. Whereas for the same (8×8) patch in a shallow layer, it can only be mapped to a (16×16) patch in the original image. Thus, as in Figure 4.13, a (8×8) patch is used in both shallow and deep layers. The 4th and 20th layer, i.e., shallow layers, are used to extract the features used as input to a small-patch segment decoder, whereas the 8th and 16th layer, i.e., deep layers, are used to extract the features for a large-patch segment decoder. What a segment decoder does is to decode the patches of different sizes into a uniform (128×128) colored image, which will be used later to compute the loss between the small and large patches.

The patches are selected based on midpoints. For an image of natural pose, its keypoints have been already annotated. For an image of artistic pose, its keypoints can be inferred by OpenPose. With the keypoints known, the midpoints can be calculated in a way that the nose is the midpoint of the head, the midpoint between the neck and the midhip is the midpoint of the torso, the midpoints between the shoulders and the elbows, between the elbows and the wrists are the midpoints of the arms, and the midpoints between the hips and the knees, between the knees and the ankles are the midpoints of the legs. As a result, the midpoints of each segment are known, which are also the midpoints of the uniform (32×32) patches in the original image. Next, based on the features extracted inside the generator, for a shallow (128×128) feature, the coordinates of the midpoints are halved, for a deep (64×64) feature, the coordinates of the midpoints are further halved. As a result, for each segment, a (32×32) patch in the original image can be mapped to a (8×8) patch in the shallow and deep layers with the centers of the patches known as the previously calculated midpoints. Furthermore, with a (8×8) patch in a shallow layer, it can be mapped to a (16×16) patch in the original image. Whereas with a (8×8) patch in a deep layer, it can be mapped to a (32×32) patch in the original image, which is comparatively larger. The mapping of the patches from the shallow and deep layers to the original image is illustrated in Figure 4.14.

The loss function is designed to enlarge a smaller patch to the size of a larger patch in both domains. The loss function used is the L1 loss for the output of the segment decoders between the small patches and the large patches. In detail, there are two pairs of inputs. A real image x of domain X is paired with a fake image y' of domain Y. A fake image x' of domain X is paired with a real image y of domain Y. The pairs can be denoted as $x' = G_{Y \to X}(y)$ and $y' = G_{X \to Y}(x)$. The loss function for domain X is defined as the sum of L1 losses between these two pairs, which is written in Equation 4.10. The function $features(generator, layer)$ returns the features from the designated layer of the designated generator. The function $segments(features)$ returns the uniform (128×128) image as the output of the segment decoder. Likewise, the loss function for domain Y is defined in Equation 4.13.

$$\mathcal{L}_X = \mathcal{L}_x + \mathcal{L}_{x'} \tag{4.10}$$

$$\mathcal{L}_x = L1\Bigg(segments\Big(features\big(G_{X \to Y}(x), 4\big)\Big), segments\Big(features\big(G_{Y \to X}(y'), 16\big)\Big)\Bigg) \tag{4.11}$$

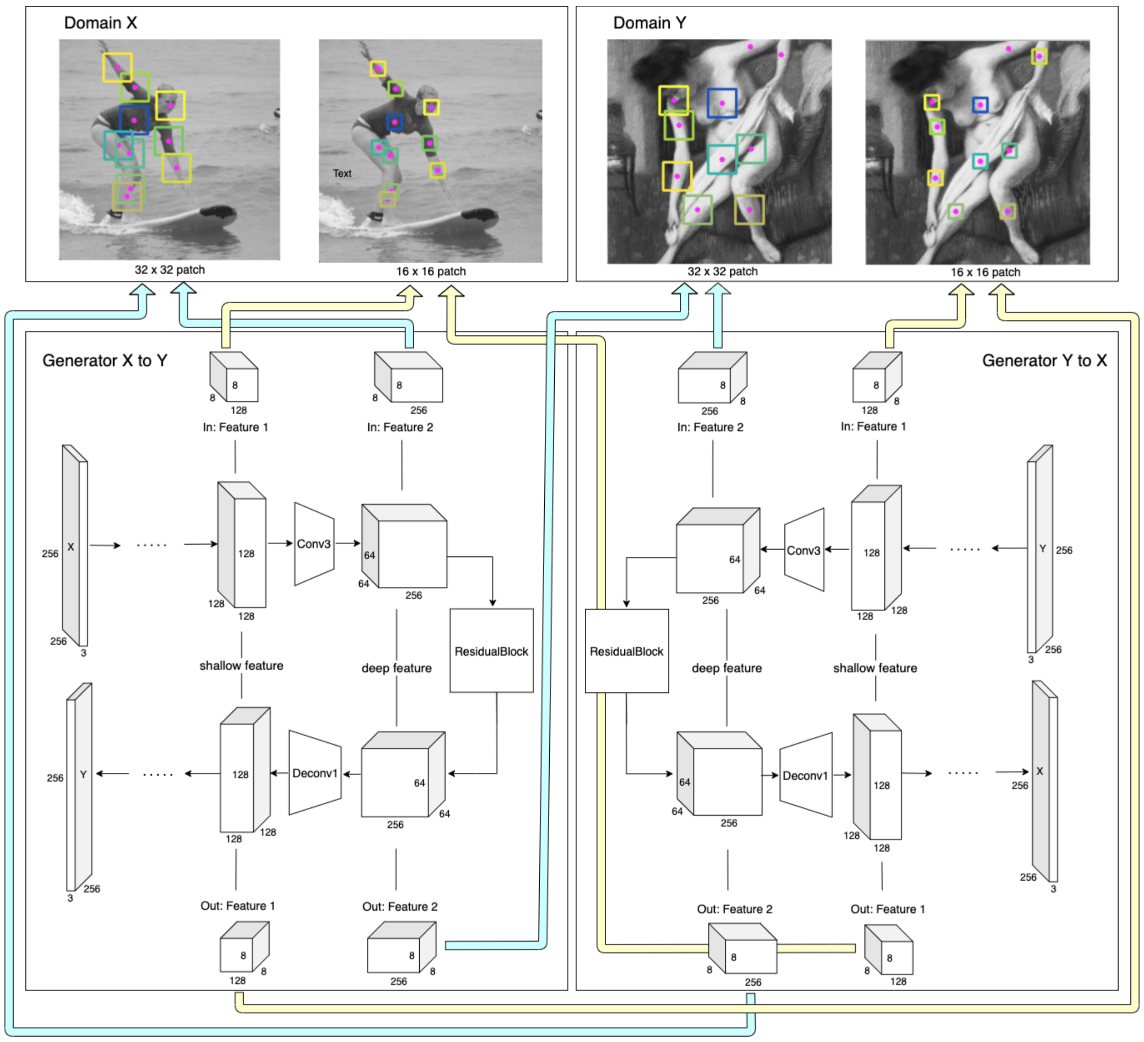


Figure 4.14: The mapped patches from the shallow and deep layers of the two symmetric generators to the original image.

$$\mathcal{L}_{x'} = L1\left(segments\left(features\left(G_{X\to Y}(x'),4\right)\right), segments\left(features\left(G_{Y\to X}(y),16\right)\right)\right) \tag{4.12}$$

$$\mathcal{L}_Y = \mathcal{L}_y + \mathcal{L}_{y'} \tag{4.13}$$

$$\mathcal{L}_{y} = L1\left(segments\left(features\left(G_{Y\to X}(y),4\right)\right), segments\left(features\left(G_{X\to Y}(x'),16\right)\right)\right) \tag{4.14}$$

$$\mathcal{L}_{y'} = L1\left(segments\left(features\left(G_{Y\to X}(y'),4\right)\right), segments\left(features\left(G_{X\to Y}(x),16\right)\right)\right) \tag{4.15}$$

The total segment loss function is defined in Equation 4.16 as the sum of the losses in domain X and in domain Y.

$$\mathcal{L}_{segment} = \mathcal{L}_X + \mathcal{L}_Y \tag{4.16}$$

4.3 Warping

As in [76], Warping was used in the geometric style transfer for portraits. In order to change the contour of a face, thin plate spline (TPS) is applied to the triangular meshes formed by the facial landmarks during interpolation. The more the landmarks, the smoother the warped artistic face. By feeding the artistically augmented faces, either textually by style transfer or geometrically by warping, further into the neural network, the trained model can more accurately predict the landmarks of an unknown artistic face. Now, in a similar way, we aim to use warping to push the contours of a natural pose to the artistic extremes, thus we can observe the synthesized result more easily. The style of Modigliani is chosen, as it focuses mostly on extremely elongated figures. Since OpenPose and DensePose don't perform well on the paintings of Modigliani, we will first superimpose the average contour of Paul Delvaux, as in Figure 4.7b, on one of Modigliani's paintings to see whether it fits or needs further adjustment. The painting chosen belongs to the training dataset of the selected 15 paintings of Modigliani, whose keypoints are all accurately inferred. The result is shown in the leftmost image of Figure 4.15. We can see that the arms and thighs fit well, whereas the calves are missing. Thus, the contours of the arms and thighs are referenced from the average contour of Paul Delvaux, whereas the contour of the calves are assumed. The average height of the upper arm contour for Paul Delvaux is 23 pixels, whereas the height of it for the tested COCO woman is 18 pixels, which means that the length of the upper arms needs to be elongated by 1.28. The average height of the thigh contour for Paul Delvaux is 80 pixels, whereas the height of it for the COCO woman is 90 pixels, which means that the length of the thighs needs to be shortened by 0.89. The calves of the COCO woman is also elongated by 1.28, as it is assumed that it has the same scaler as that of the arms. The middle image of Figure 4.15 shows the direction of warping, in order to transform the COCO woman to the assumed average contour of Modigliani. The warping is carried out point-wise, given the original keypoints and their corresponding shift vectors along the x and y axis. The warped output is shown in the rightmost image of Figure 4.15.

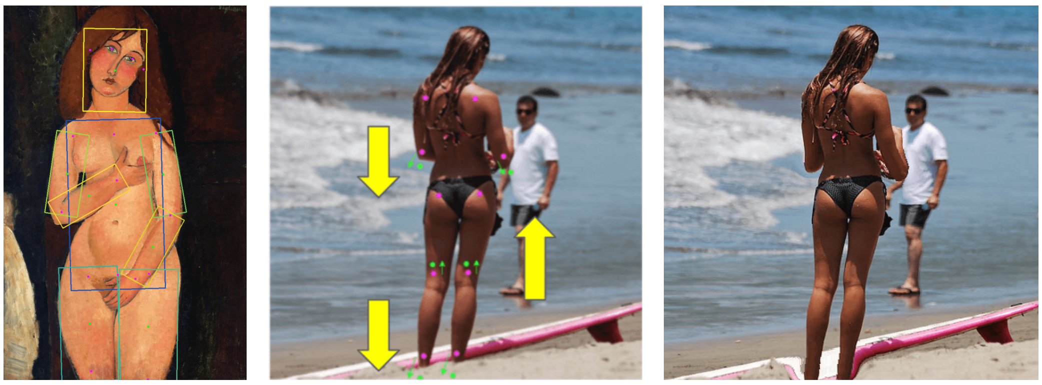


Figure 4.15: The poses of Modigliani, to which a natural pose is aimed to warp.

The final results of the CycleGAN-based style transfer will be shown in Section 5.1, and the final results of the warping-based style transfer will be shown in Section 5.2.

Chapter 5: Results

In this section, the results of the CycleGAN-based and the warping-based style transfer will be shown in Section 5.1 and Section 5.2 respectively. As the goal of the CycleGAN-based style transfer is to solve the challenges as to (1) how to match the source and target locations, and (2) how to change the shapes of body segments, only one test image will be used during style transfer in order to focus on the outcome as to whether the shapes are successfully changed. The test image is a standing women with all the limbs shown and her back turning to observers, because the full pose is needed to test the shape-changing effect, whereas the face can be ignored, as to change the face artistically is currently not taken into consideration. This is the same for the warping-based style transfer, so the same test image will be used as well in order to compare the results based on the same image.

5.1 CycleGAN

The results of CycleGAN will be shown for three variants respectively, which are the vanilla CycleGAN model in Section 5.1.1, the CycleGAN model with patch-wise loss in Section 5.1.2, and the CycleGAN model with contour-wise loss in Section 5.1.3. For all the three variants, the architecture of generators and discriminators will be kept the same except for different loss functions to enforce the mapping between the source and target locations. The number of training epochs is set to 200. During each epoch, the weights of generator $G_{X \to Y}$ and generator $G_{Y \to X}$, discriminator D_X and discriminator D_Y will be updated in order. For one epoch, all the natural poses will be iterated through to pair with a randomly-chosen artistic pose, based on which the total loss will be calculated and recorded, and the whole networks will be updated by backpropagation. When the training has been completed, all the losses will be depicted in the loss charts, and the final style transfer outcome will be shown at last.

5.1.1 Baseline

The loss chart of the vanilla CycleGAN model is shown in Figure 5.1. It can be observed that the total discriminator loss decreases slightly from 0.43 to 0.16, in which the discriminator X declines from 0.22 to 0.0044, and the discriminator Y drops from 0.21 to 0.15. It means that for the discriminator to recognize true surfing photos, it learns quickly, whereas for the discriminator to appreciate the Impressionism paintings, it learns relatively slower. Moreover, the total generator loss decreases tremendously from 10.24 to 3.46, in which the identity loss drops from 2.97 to 0.60, the GAN loss increases from 0.97 to 1.30, and the cycle loss declines from 6.30 to 1.56. We can see that the cycle loss decreases the most, whereas the GAN loss is unstable and fluctuates up and down.

The generated images of these two CycleGANs. Figure 5.2 shows the generated images of the Impressionsim CycleGAN. The first column shows a pair of the original training images from domain X (upper row) and domain Y (lower row) respectively. The second column shows the generated images of the corresponding domains that are trained by identity loss: a generated image by the generator $G_{Y \to X}$ (upper row), and one by the generator $G_{X \to Y}$ (lower row). The third column shows the generated images to the counterpart domain, which are trained by GAN loss. The fourth column illustrates the generated images trained by cycle loss: the original image of domain X, which is translated to Y and back to X (upper row), one of domain Y, which is

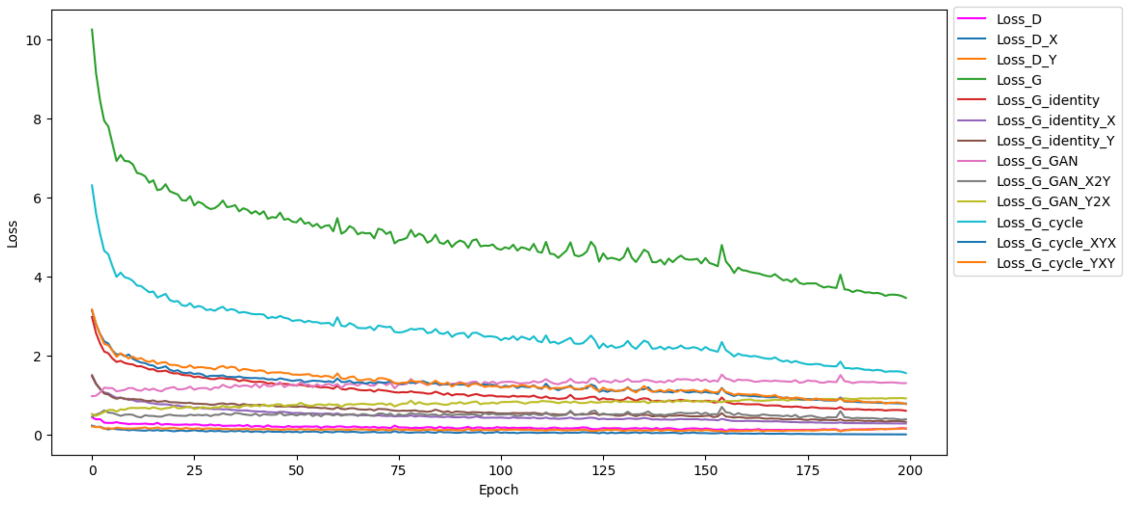


Figure 5.1: The losses of the vanilla CycleGAN, translate from the COCO surfing women to the Impressionism nude paintings.

translated to X and back to Y (lower row).

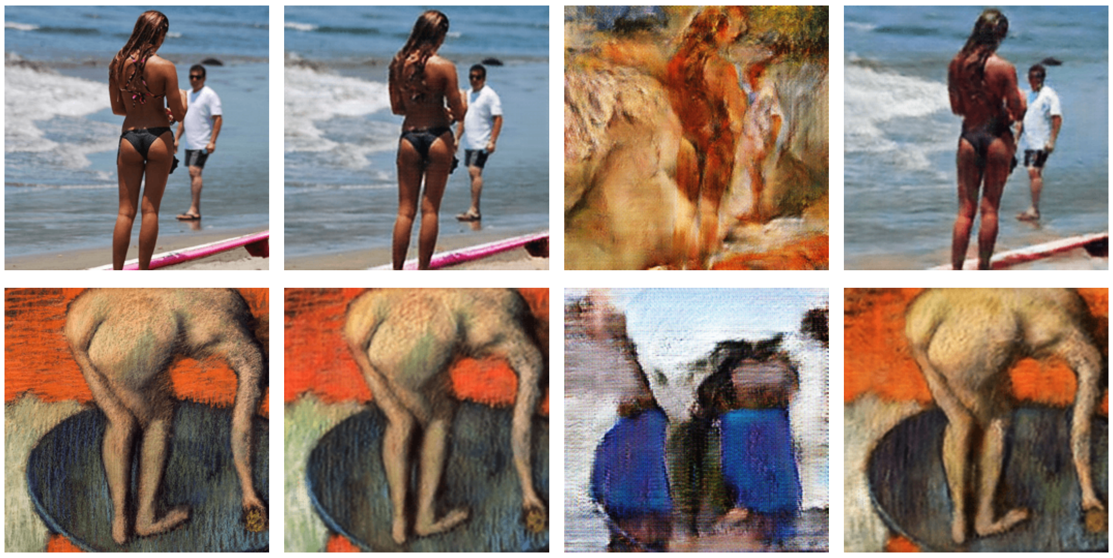


Figure 5.2: The generated images of the vanilla CycleGAN to translate from the COCO surfing women (domain X) to the Impressionism nude paintings (domain Y). 1st column: the original images. 2nd column: the images generated by identity loss. 3rd column: the images generated by GAN loss. 4th column: the images generated by cycle loss.

It can be observed that the images generated by identity loss (2nd column) and cycle loss (4th column) are quite similar to the original images (1st column), which means that our CycleGAN learns well given these two losses. What is most important as a goal of our CycleGAN is however to generate images from domain X to domain Y and vice versa, which are illustrated in the 3rd column. It can be seen that the fake Impressionism nude woman (upper row) picks up the color of domain Y, but the body segments are mingled with the reddish background and very blurry, which thus don't stand out as a standing person. On the other hand, the fake photographic COCO woman (lower row) is totally unrecognizable, as the body segments are colored as tides of the sea. Compared with translation from photo to painting, translation from painting to photo is relatively more complicated, and the potential reason might be that photos contain more information than that of paintings.

In summary, the vanilla CycleGAN is able to learn color and texture from each domain, but it fails to transfer them from the right source locations to the right target locations.

5.1.2 Patch-wise loss

The CycleGAN model with patch-wise loss comes into two versions: (1) PatchGAN for discriminator, and (2) PatchNCE loss for generator. First, the results of PatchGAN version will be shown. PatchGAN uses patched images during training, and one example of the patched input images is shown in Figure 5.3a. After training for the same 200 epochs, its result is illustrated in Figure 5.3b. As comparison, the result of the vanilla CycleGAN is shown in Figure 5.3c.

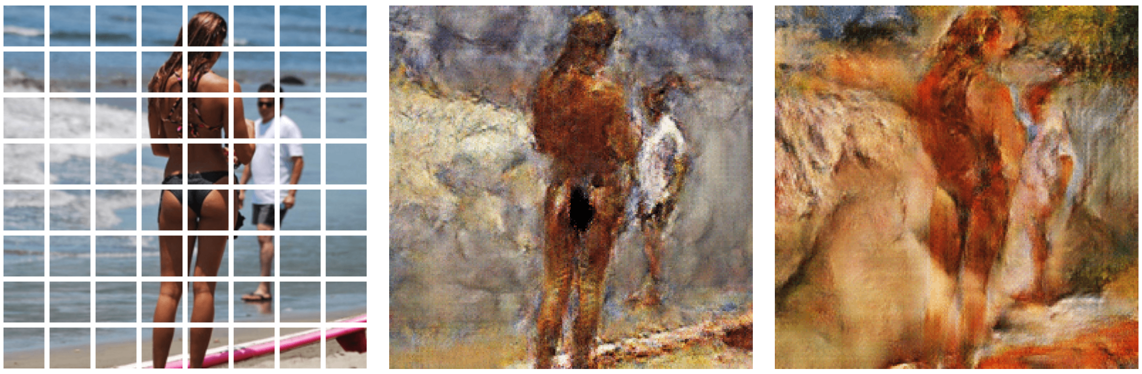

(a) The patched image as input (b) CycleGAN with PatchGAN (c) The vanilla CycleGAN

Figure 5.3: The comparison of the results of the CycleGAN variants.

Compared with the vanilla CycleGAN, PatchGAN improves the result with respect to the spatial restraint of the colors, which leads to more vivid contour of the figure. To explain, in the result of PatchGAN, the colors of the blue sea, the white tides, and the brownish skin of the surfing woman are more well-preserved. Whereas in the result of the vanilla CycleGAN, the mixed color of reddish-nude overflows the generated painting as a mean color of the whole image, and the colors of the background are lost.

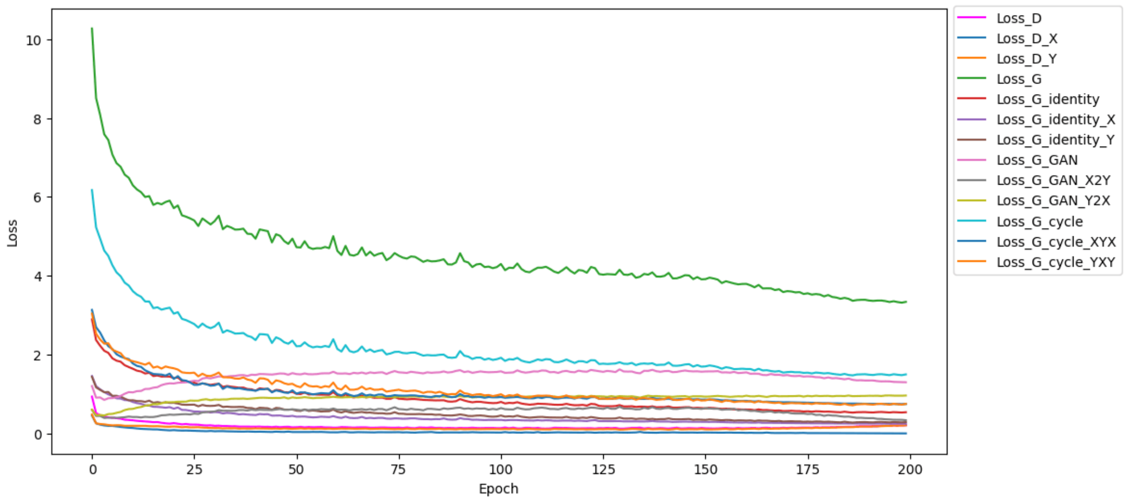


Figure 5.4: The losses of the vanilla CycleGAN with PatchGAN as discriminator.

The loss graph of this version of CycleGAN is plotted in Figure 5.4. As shown, the total discriminator loss decreases from 0.94 to 0.21 from start to finish, with 0.13 as its lowest point around epochs 80 to 160. It halves itself drastically at the beginning to 0.40 at epoch 5, then it declines slowly afterwards. For the discriminator X, it drops from 0.45 to 0.003, and for the discriminator Y, it decreases from 0.48 to 0.21. The total generator loss decreases from 10.27 to 3.34, which is almost the same as that of the vanilla CycleGAN. We can see that compared to the vanilla CycleGAN, the absolute loss of the PatchGAN discriminator spans a wider range, which might suggest that it learns better and thus acquires the knowledge of spatial relationships.

For an input image of the size (256×256), the patch size (32×32), as shown in the result of

Figure 5.3a, is the smallest possible one, given the kernel size is set to be 4 in the discriminator. To further explore how the patch size can influence the result, larger patch sizes are tried as well, which are (64×64) and (128×128). The comparison of the generated images by various parameters of the patch sizes is shown in Figure 5.5. It can be observed that as the patch size gets larger, the tiling artifacts get smoothed out. The tiling artifacts stand out in the photographs as the traces left by the processed patches, but in the paintings, they might be perceived as broad brush strokes that are characteristic of Impressionism. Moreover, the smaller the patch size, the more the spatial statistics are kept, so the background is more recognizable. Finally, the smallest patch size (32×32) is chosen, as it keeps the spatial information the best.

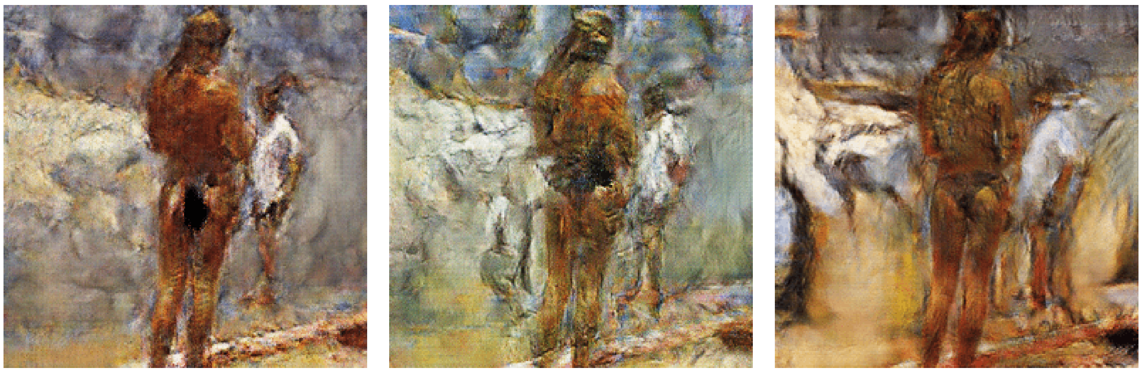

(a) Patch size (32×32) (b) Patch size (64×64) (c) Patch size (128×128)

Figure 5.5: The comparison of the results by the setting of various patch sizes in PatchGAN.

In summary, the patch size of PatchGAN can have effect on the generated images in two ways: (1) The smaller the patch size, the more the spatial statistics. By spatial statistics, it means that the content of the image is kept with respect to the spatial relationships of its contained objects. For example, the standing pose is clear and not smeared with fuzzy outlines. (2) The larger the patch size, the less the tiling artifacts. By tiling artifacts, it means that the image seems to be formed by layers of small tils on top of each other.

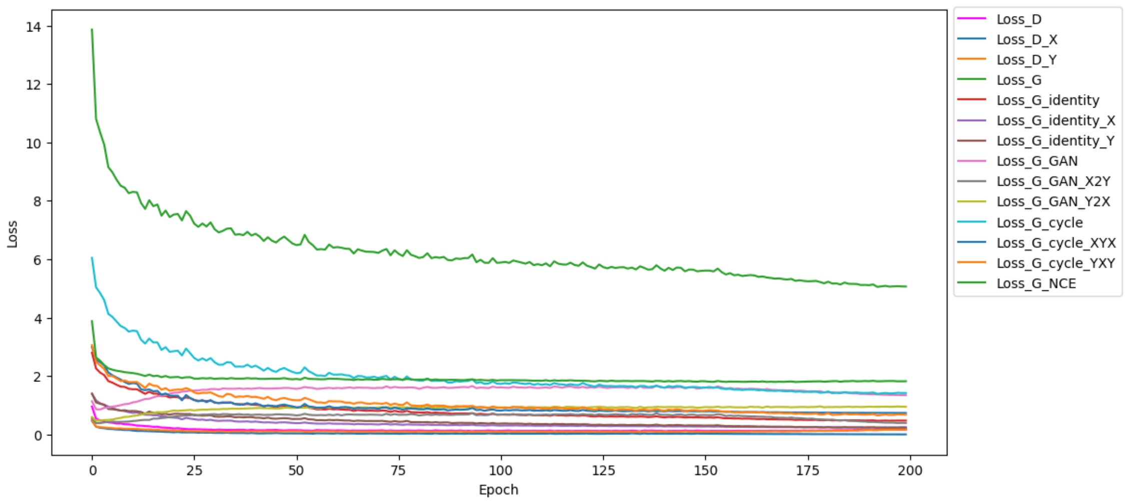


Figure 5.6: The losses of the vanilla CycleGAN with PatchGAN and PatchNCE loss introduced.

Second, the results of PatchNCE-loss version will be shown. After training for the same 200 epochs, its loss graph is shown in Figure 5.6. From the loss graph, we can see that the PatchNCE loss decreases from 3.88 to 1.82. The decline stagnates at around epoch 112 at 1.83, and fluctuates afterwards. The identity loss drops from 2.80 to 0.48, the cycle loss decreases from 6.05 to 1.42, which are very similar to that of the vanilla CycleGAN. The total discriminator loss goes from 0.95 to 0.17. For the discriminator X, the loss decreases from 0.47 to 0.0036, and for the discriminator Y, it drops from 0.48 to 0.17. For the discriminator loss as a whole, it is also very similar to that of the PatchGAN version. By the losses, we know that the PatchNCE loss does learn during the training. Next, we will see what the PatchNCE loss contributes to the final output of the generated images.

Figure 5.7a shows the generated image with the PatchNCE loss introduced. It can be observed that in comparison, the contour of the figures becomes sharper, with the woman more outstanding out of the background, and the man in the background more recognizable. Besides, it has greater color diversity, with the color of the sea ranging from sky blue, purplish blue to emerald green, whereas in the PatchGAN-only version, the sea is smeared in purple.

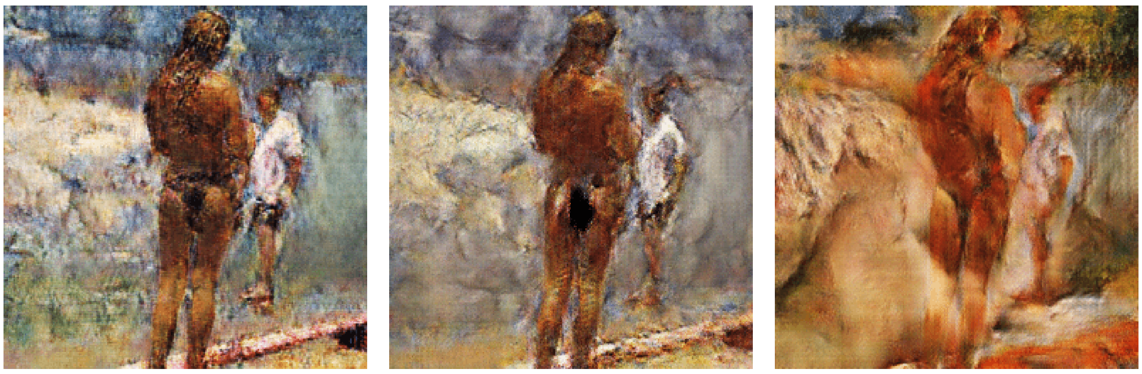

(a) PatchGAN with PatchNCE (b) PatchGAN only (c) The vanilla CycleGAN

Figure 5.7: The comparison of the results of the CycleGAN variants.

Finally, we would like to explore how the number of patches can impact the visual outcome, thus 128 as less patches, and 512 as more patches are chosen and tried respectively. The results are shown in Figure 5.8a and Figure 5.8c. As observed, no matter how many patches there are set for the PatchNCE loss, the color of the generated images stays diversified. Besides, the visual difference of all the three images from Figure 5.8 is very subtle. When the patch number is 128, the man's shirt is less white, and he is more smudged in the background. When the patch number is 512, the man is also very blurry regarding the area around his armpit. The potential reason might be that there exists a trade-off between spatial statistics and color statistics. If the patch number is smaller, the spatial statistics are not sufficiently learnt, thus the image is blurry. If the patch number is larger, the color statistics are averaged more times over more locations, which could result in blurry images.

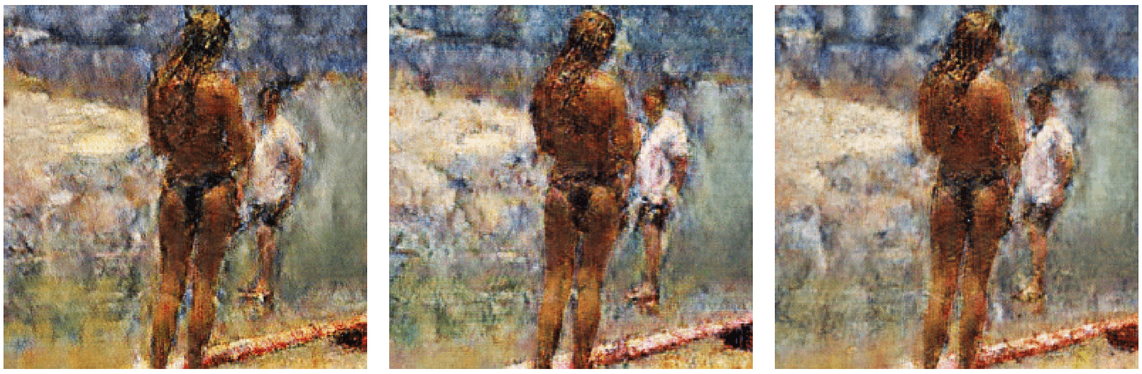

(a) 128 patches (b) 256 patches (c) 512 patches

Figure 5.8: The comparison of the results of various number of patches of a PatchNCE loss.

In order to verify whether the above assumption holds, less patches, i.e., 32 and 64 patches, and more patches, i.e., 1024 and 2048 patches, are tested as well, and the result is shown in Figure 5.9. It is expected that less and more patches will both lead to perceptibly fuzzier images, but the difference is very subtle. Thus, our final experiment with the contour-wise loss will be based on the CycleGAN variant with 256 patches for a PatchNCE loss and a (32×32) PatchGAN.

In summary, with the PatchNCE loss introduced, the performance of the vanilla CycleGAN is

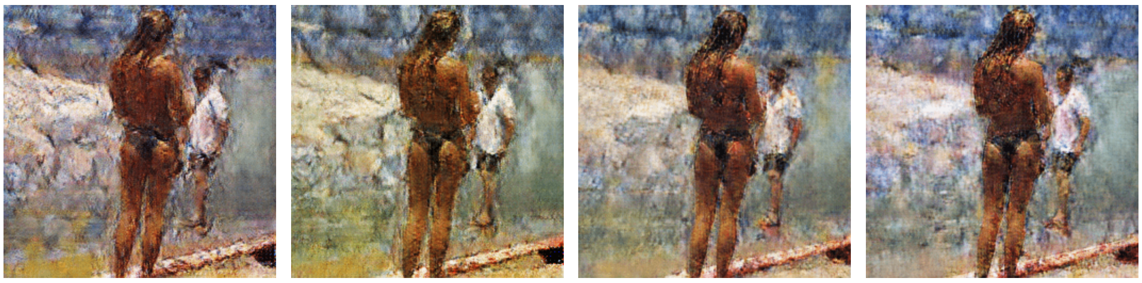

(a) 32 patches (b) 64 patches (c) 1024 patches (d) 2048 patches

Figure 5.9: The comparison of the results of less and more patches.

improved with respect to spatial statistics and color diversity.

5.1.3 Contour-wise loss

Finally, the results of the CycleGAN model with contour-wise loss will be shown. After training for 200 epochs, the total loss is shown in Figure 5.10.

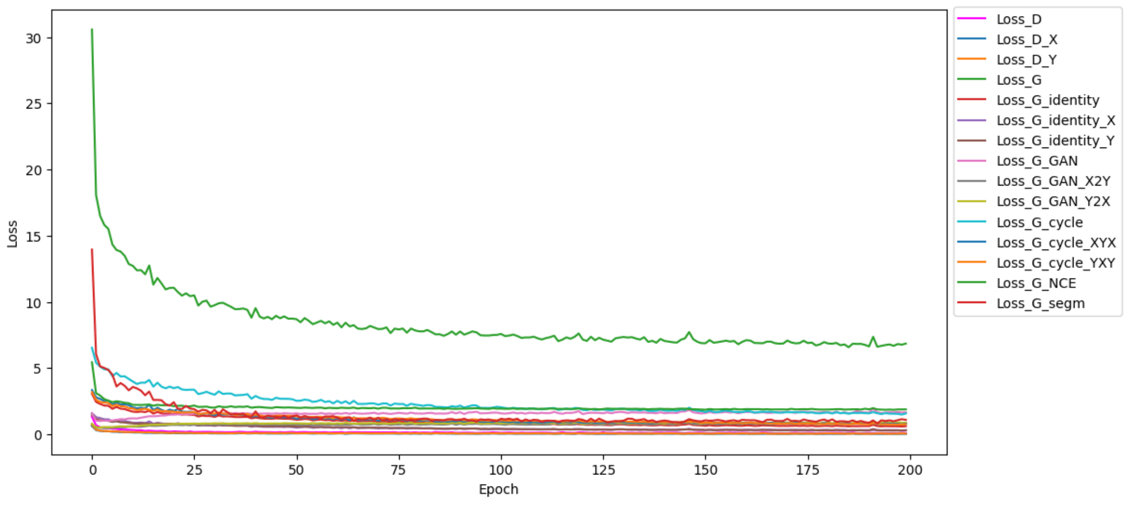


Figure 5.10: The losses of the CycleGAN with contour-wise loss introduced.

As illustrated in Figure 5.10, the contour loss decreases from 13.96 to 1.09 from epoch 1 to epoch 200. In the meantime, the total generator loss, including the contour loss, drops from 30.57 to 6.85, in which the identity loss decreases from 3.07 to 0.62, the GAN loss fluctuates between 1.55 and 1.77, the cycle loss drops from 6.54 to 1.59, and the patchNCE loss decreases from 5.44 to 1.89. The total discriminator loss decreases from 1.33 to 0.10, in which the discriminator X loss drops from 0.63 to 0.04, and the discriminator Y loss goes from 0.70 to 0.06. We can see that except for the contour loss, the other losses remain almost the same as those with PatchNCE loss introduced.

As the learning is reflected in the contour loss, now we will show the outcome to check out whether the shape of a pose can be changed in the generated image with the contour loss introduced. Figure 5.11a shows the result of the CycleGAN with the contour loss introduced, in comparison with other two variants. Since the goal of the contour loss is to enlarge a smaller (16×16) body segment to be matched with a larger (32×32) body segment, it is expected to see that the standing figure is enlarged twice. But the result does not meet this expectation, and the generated image gets more blurry, with more patch artifacts, one of those is like salt and pepper noise sprinkled above the whole image, which is far from ideal.

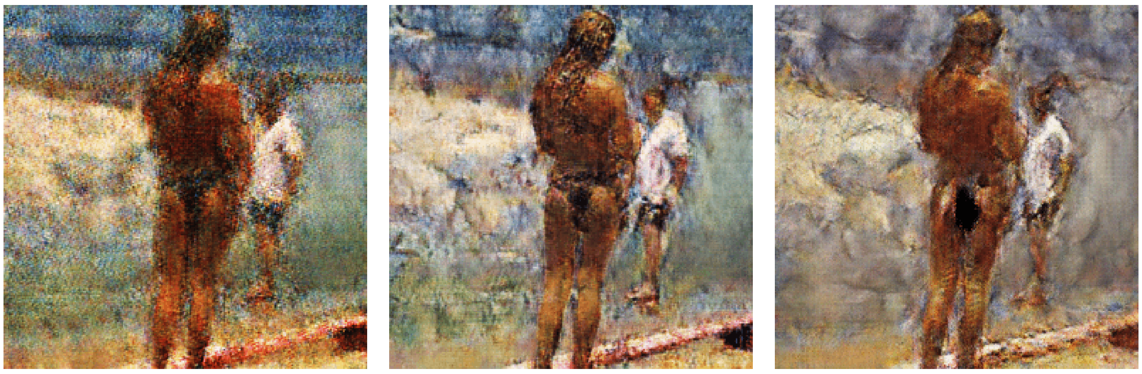

(a) CycleGAN with contour loss (b) PatchGAN with PatchNCE (c) PatchGAN only

Figure 5.11: The comparison of the results of the CycleGAN variants.

5.2 Warping

By the method of warping, style transfer will be carried out in two steps. The first step is to warp the keypoints of a pose to match the desired artistic contour by shape. The second step is to transfer color and texture by a neural-network-based style transfer model. For style transfer, there already exist a variety of neural network models. For CycleGAN-based models, there are contrastive unpaired translation (CUT) [55] and single image contrastive unpaired translation (SinCut) [55]. For other neural network models, adaptive instance normalization (AdaIN) style transfer [33] will be selected as the state-of-the-art model. In order to compare the results of various style transfer models, in total, four models have been chosen: (1) Our best-performing CycleGAN model, i.e., PatchGAN with PatchNCE loss, (2) CUT, (3) SinCUT, and (4) AdaIN. The reasoning is that (1) Our model is based on the same architecture with a ResNet-based generator, a PatchGAN-based discriminator, and the PatchNCE loss as in [55]. (2) CUT is trained with more fine-tuned parameters and architecture, thus it is expected that CUT can generate better result. (3) SinCUT is designed to be trained with unpaired single images, so that only one image of domain X and Y are needed for the training, during which the image from each domain will be decomposed into matched patches. Compared to CUT, SinCUT has the advantage that it needs only one pair of images, so style transfer can be better manipulated with respect to the desired color and texture, other than based on the color statistics of a set of artistic training images. (4) AdaIN also uses one pair of images during style transfer, which are one content image and one style image. It is intriguing to compare AdaIN with SinCUT, as SinCUT uses contrastive learning to preserve the spatial statistics in the content image, whereas AdaIN uses perceptual loss with adaptive instance normalization to preserve the spatial statistics in the content image.

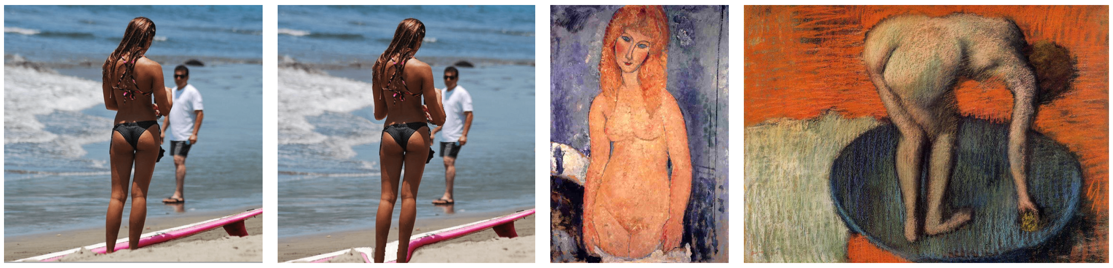
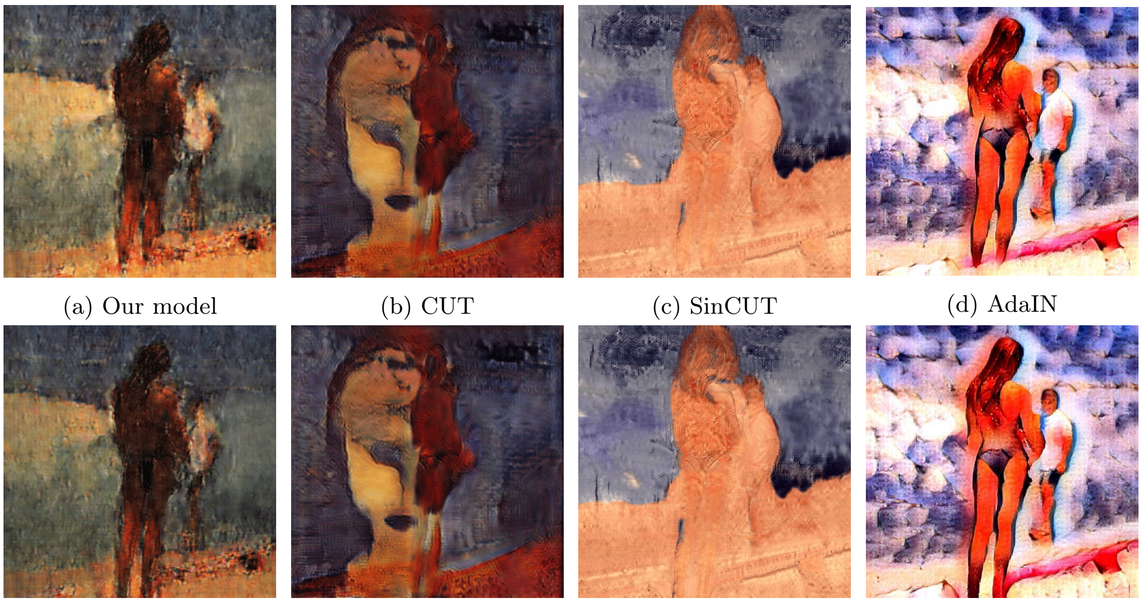


Figure 5.12: Leftmost two images: Content images with original and warped poses. Rightmost two images: Style images from the paintings of Modigliani and Impressionism

As the goal of the warping-based method is to check out the results of geometry-aware style transfer in its artistic extremes, the poses drawn by Modigliani with extremely elongated limbs are used. In order to match the expected style, the training data of artistic poses are changed from the paintings of Impressionism to all of the 335 paintings of Modigliani from “Painter by Numbers”, which are used by our model and CUT. For SinCUT and AdaIN, one painting of Modigliani is used as the input style image, which is shown in the third image of Figure 5.12. Finally, in order to check out the results of these four style transfer models given different desired styles, Impressionism will be used in comparison. The same training data of Impressionism will be used to train our model and CUT. For SinCUT and AdaIN, one painting of Impressionism is used as the input style image, which is shown in the fourth image of Figure 5.12.

The output of the generated stylized poses are shown in Figure 5.13, in which the first row illustrates the style transfer results of the original image, and the second row shows the parallel results of the warped image. It can be observed that either CUT or SinCUT does not respect the outline of the figure, so the surfing COCO woman is smeared to unrecognizable. For SinCUT, the color of the painting is well-preserved with fading shades and smooth brush strokes. For CUT, the generated image might reflect the color statistics of all the training paintings. For our model, the figure is clear with the skin color changed to reddish. Compared to the generated pose of Impressionism in Figure 5.14a, the brush strokes are more smooth towards the style of Modigliani, other than the short and jumpy brush strokes dominant in the style of Impressionism. Compared to our model, AdaIN can also preserve the outline of each figure with bold lines, but the color and brush strokes are more fractured, which do not well preserve the color and texture statistics of the painting of Modigliani, whereas SinCUT performs better in this sense.


(a) Our model (b) CUT (c) SinCUT (d) AdaIN

(e) Our model (f) CUT (g) SinCUT (h) AdaIN

Figure 5.13: Comparison of the style transfer results for Modigliani.

The same four models have been trained for the style of Impressionism as well in order to compare the texture and color transfer results across styles. Figure 5.14 shows the results.

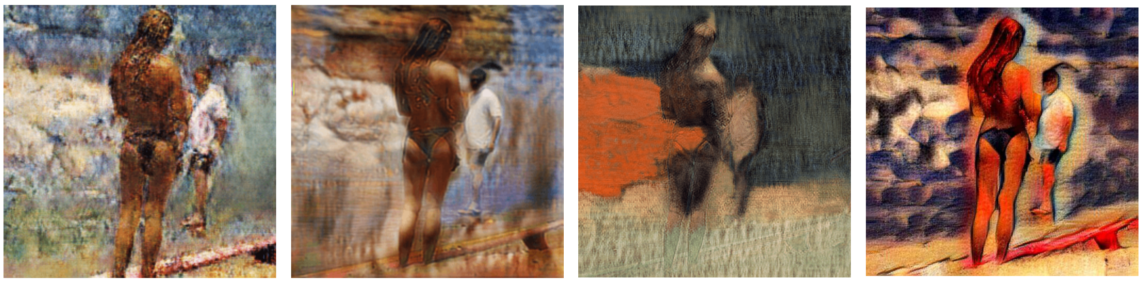

(a) Our model (b) CUT (c) SinCUT (d) AdaIN

Figure 5.14: Comparison of the style transfer results for Impressionism.

In summary, our model is able to (1) translate the color and texture statistics from artistic poses to natural poses, as in style, and (2) preserve the spatial statistics of natural poses, as in content. However, it still needs to be explored as to how to change the shapes of the body segments of a pose to match the desired artistic styles in one step without manual warping.

5.3 Results overview

In this section, we will extend our experiment from only one pose to other poses of the COCO surfing women. The purpose is to compare the style transfer results of the aforementioned models by various styles, given a variety of poses. The poses chosen are shown in Figure 5.15, which range from lying, crouching and standing. Two styles are aimed. One is the style of Modigliani, as shown in Figure 5.16, and the other is the style of Impressionism, as shown in Figure 5.17. These two styles are chosen, as they are distinctive in color and texture. For Modigliani, the color used is mostly dark red and blue, and the brush strokes are smooth. For Impressionism, it is colorful, and the brush strokes are short and jumpy. The purpose is to test by style transfer, whether the color and texture can be distinctively transferred from style to style.

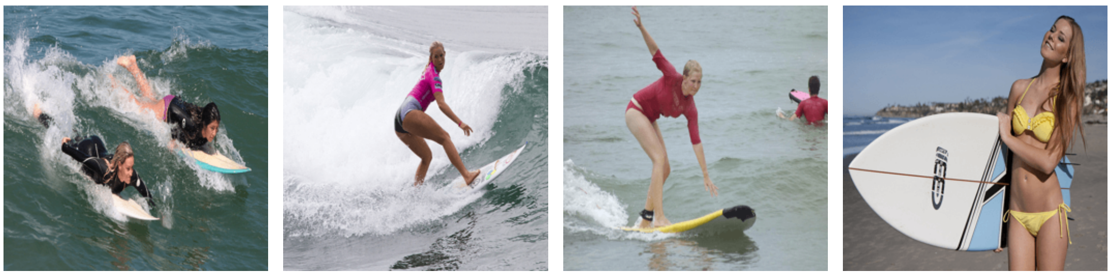


Figure 5.15: The poses of the COCO surfing women

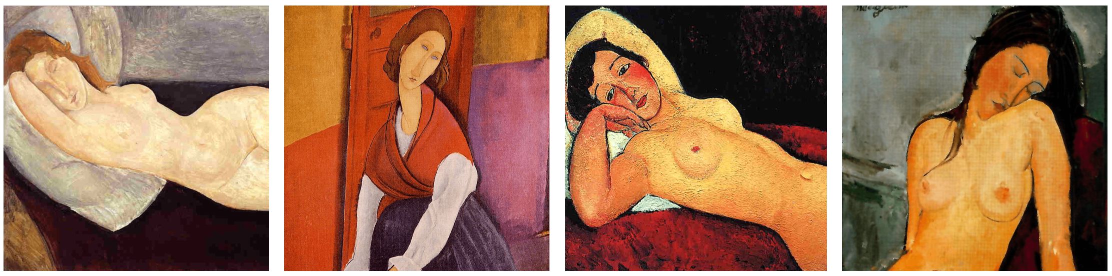


Figure 5.16: The paintings of Modigliani

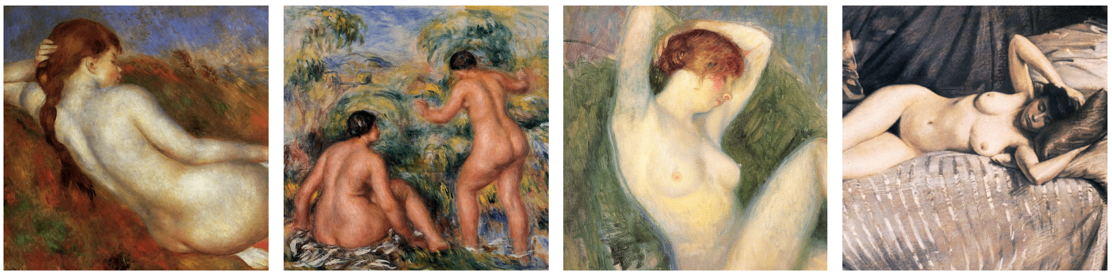


Figure 5.17: The paintings of Impressionism

The results of style transfer for the style of Modigliani are shown in Figure 5.18, and the results for the style of Impressionism are illustrated in Figure 5.19. For both Figure 5.18 and Figure 5.19, the first row shows the results of our model, the second row shows the results of CUT, and the third row shows the results of AdaIN.

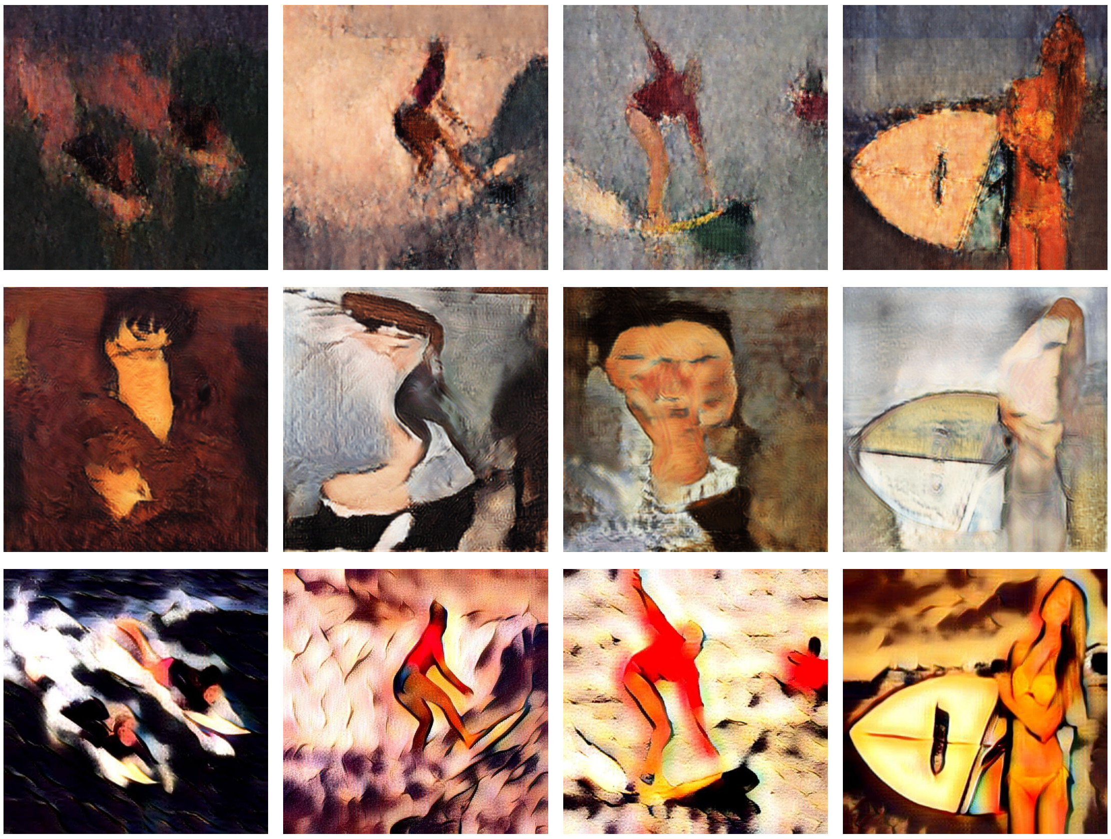


Figure 5.18: Results of style transfer for the style of Modigliani

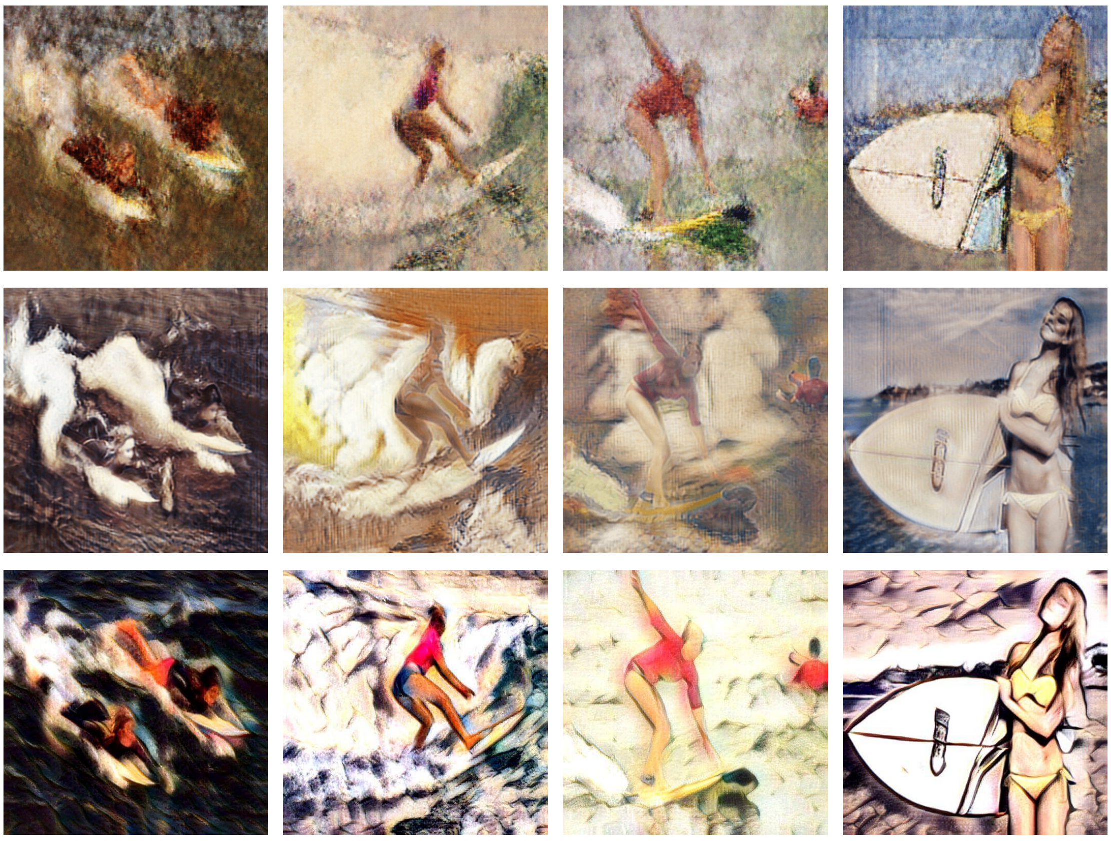


Figure 5.19: Results of style transfer for the style of Impressionism

It can be observed that by CUT, the outlines of the poses are generally not respected. For the results of Impressionism, though the surfing figures are still recognizable, the colors of the original COCO photographs are almost overwritten by the colors of the randomly paired training paintings from Impressionism. The colors of the results do not reflect the overall color statistics of the datasets of Impressionism, but are influenced by the randomly paired one painting during training.

For AdaIN, the difference between the styles of Modigliani and Impressionism is not distinguishing anymore in the final results. In the final results, the outlines are emboldened, but the brush strokes are lost in the bold lines.

For our model, though the color statistics can reflect the difference between these two styles, it results from the random pairing during training as well. If the input natural poses are randomly paired with the colorful paintings, the style transfer results will be colorful as well. For brush strokes, they are short and with much noise for both styles. The patches of color are not as smooth as those of CUT, but are sprinkled with black and white dots.

Chapter 6: Conclusion

The goal of this thesis is to build a geometry-aware style transfer which can stylise a natural pose, given an aimed artistic pose. The natural poses come from the COCO people dataset, as it contains a variety of human poses in common activities and sports such as walking and surfing, and the joints and body segments of each pose have already been manually annotated. The artistic poses come from the "Painter by Numbers" dataset, as it consists of an almost full collection of paintings that range from the early 11th century to the 2010s. What we have contributed in this thesis is a small set of annotated artistic poses. By annotation, the state-of-the-art pose estimation tools are used, which are OpenPose for joints, and DensePose for body segments. The annotation process is carried out automatically by these two tools, and the results are from their inference.

In order to further analyze the annotation results, a small set of paintings are chosen with various styles. The purpose is to test given different styles, how well OpenPose and DensePose can perform by inference in the artistic domain, as they are trained only with natural poses. In detail, 10 artists are selected with 5 classical artists ranging from Renaissance to Impressionism, and 5 modern artists post Impressionism. It is found that the accuracy of inference is prone to the following factors: (1) The number of people in the painting, (2) The occlusions of clothes and interactive people, (3) The niche poses, i.e., cuddling oneself or lying on one's arms, whereas the easy poses are sitting or standing, (4) The niche perspectives, i.e., viewing from bottom up, (5) The artistic effects, i.e., the color contrast and brush strokes, (6) The shapes of body segments, especially when the body parts are exaggerated, and (7) The body proportions, e.g., the elongated torso and limbs. As a result, the poses drawn by Paul Delvaux have the highest accuracy by both OpenPose and DensePose, as most of the depicted poses are standing nude women. Whereas the poses drawn by Modigliani and Lempicka have the lowest accuracy due to the elongated body proportions, the inflated shapes, the niche lying poses, and the niche viewing perspectives.

In order to further explore the dataset, the statistic analysis is carried out for both natural and artistic poses. There are two kinds of analyses. First, he elliptical distribution is carried out to illustrate the articulation of the joints. The result shows that the artistic poses have a comparatively limited range of articulation, whereas the natural poses are more varied. The potential reason might be that the artistic poses are staged and constrained, whereas the natural poses are collected from all the daily activities and sports. Second, the elliptical distribution is carried out to illustrate the common and niche poses in clusters. The clusters can be shown in a dendrogram, which can further tell the differences of poses in: (1) The left and right orientation, (2) Whether the limbs are stretched or compressed, i.e., angles between two neighbouring limbs, and (3) The completeness of the joints. The cluster with many similar poses forms the common poses, whereas the cluster with only few poses constitutes the niche poses. For artistic poses, the niche pose is to spread both arms skyward. For natural poses, the niche poses come mostly from the twisted legs, stretched arms, and lying. By the niche natural poses from the COCO people dataset, it might further explain why OpenPose and DensePose perform worse in the inference of paintings, as these niche poses are also under-represented in the training dataset.

With natural and artistic poses, we finally start to build a geometry-aware style transfer. There are two methods. The first method is based on CycleGAN which is aimed to transfer shape, color and texture automatically in one step. The second method is by warping which is carried out in two steps: (1) For shape, the body segments are morphed from a natural pose to an artistic pose by manually warping their keypoints to match the desired artistic pose. (2) For color and texture, it is imposed from an artistic pose to a natural pose by style transfer. The challenge of the CycleGAN-based style transfer is how the shape signal can be conveyed during training. We have implemented a segment-loss function in order to enlarge each body segment by bridging the small and large patches, but it failed. The challenge of the warping-based style transfer is that based on keypoints, there are not enough points to form the triangular meshes of a pose for interpolation. The result of warping is better than that of the only CycleGAN method, as the shape can be directly manipulated by warping.

At last, we have compared various state-of-the-art style transfer methods with respect to color and texture, which are CUT, SinCUT, and AdaIN. CUT and SinCUT are CycleGAN-based models, whereas AdaIN is an improved version of the original style transfer method based on perceptual loss. The challenge of the CycleGAN-based methods is that for the unpaired training data, how the model can learn to transfer from the source locations to the right target locations. We have found that the CycleGAN model with PatchGAN for discriminator and PatchNCE loss for generator can generate the best results, as it can preserve the spatial relationships between patches as content, and transfer the color statistics from the paintings as style. In comparison, CUT and SinCUT cannot preserve the contours of the poses. AdaIN has limited capabilities to learn the color and texture from various paintings styles.

Chapter 7: Discussion

One of the directions that might be interesting to explore in the future is to enhance the inference accuracy of DensePose for artistic poses. There are two possible ways. One is to use OpenPose in conjunction with DensePose, in which the contour can be first generated based on the inferred keypoints. Subsequently, the contour can be used as the bounding box for each segment to trim away the incorrectly inferred pixels. The other is to manually warp the natural poses and then perform style transfer on them according to a specific style. The resulting images can be treated as the augmented training data to train a custom OpenPose and DensePose model. For warping, the current issue might be that the keypoints are sparse, therefore it cannot form dense triangular meshes based on which the interpolation can be performed smoothly. By transforming the inferred body segments further into patches of equidistant dots, the triangular meshes can be established, based on which deformation can be better performed.

One question posed is that whether style transfer is only able to be used as data augmentation in order to prepare the training data to train other neural network models for them to be used across domains, or it can be used to generate artistic paintings. As shown in the results, even in terms of copying a specific style with respect to color and texture, it still needs to be explored whether better methods exist.

Appendix A: PoC of vanilla CNN

As introduced in Section 2.1.1, HOG is built based on Canny Edge Detector, where the features in terms of edges are extracted by the hand-crafted filters, e.g., Sobel kernel. Based on each object's edge descriptor, HOG can be trained to distinguish a person from other objects, so HOG is mainly used as the pedestrian detector.

In order to illustrate that (1) CNN can automatically learn the filters and hierarchies; (2) the features extracted from these filters are not only limited to edges, and can be miscellaneous depending on the given task, we will train a vanilla CNN model and visualize the filters and feature maps in order to illustrate the basic mechanism of CNN. The task we select is to distinguish a portrait from a landscape painting, and the reasoning behind this task is that (1) we want to test how well CNN can perform in the domain of artistic paintings other than photographs; (2) this is similar to the task of classifying person and non-person in HOG.

In HOG [14], a training dataset INRIA contains 2478 images of humans (including their left-right reflections), and 12180 patches sampled randomly from 1218 person-free cityscape photos. A SVM model was trained based on the HOG descriptors. As a result, it reaches average precision of 0.755 to successfully detect a person at a recall rate of 0.7. For the vanilla CNN model, it contains only 6 convolution layers, with max-pooling layer after every 2 convolution layers to downsample the input images. And it uses the categorical cross-entropy loss, which is common in classification tasks. Adam optimizer is used with L2 kernel regularizer set at 0.001 to reduce overfitting. The training dataset comes from Kaggle's Wiki-Art, and it is balanced with 13000 landscape and portrait paintings respectively with a validation split 0.2. The test dataset is reserved to verify this CNN model's classification accuracy, and it contains 2000 landscape and portrait paintings respectively. All the images have been converted to grayscale. The confusion matrix will be calculated based on the result of test dataset in order to compare the precision and recall rate on an equal foot with HOG. Furthermore, the learned filters and the images calculated by the corresponding filters (activations) will be visualized to verify what has been learnt in CNN. We want to explore whether any edge filters contribute to the classification process, and whether there are other filters which also contribute to the classification task of person and non-person.

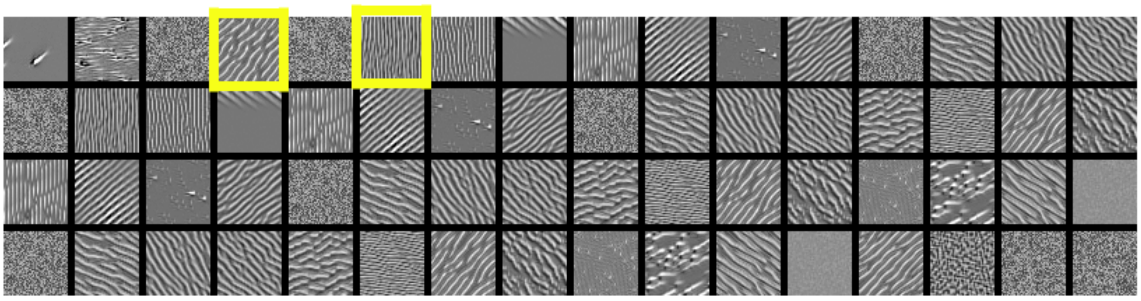
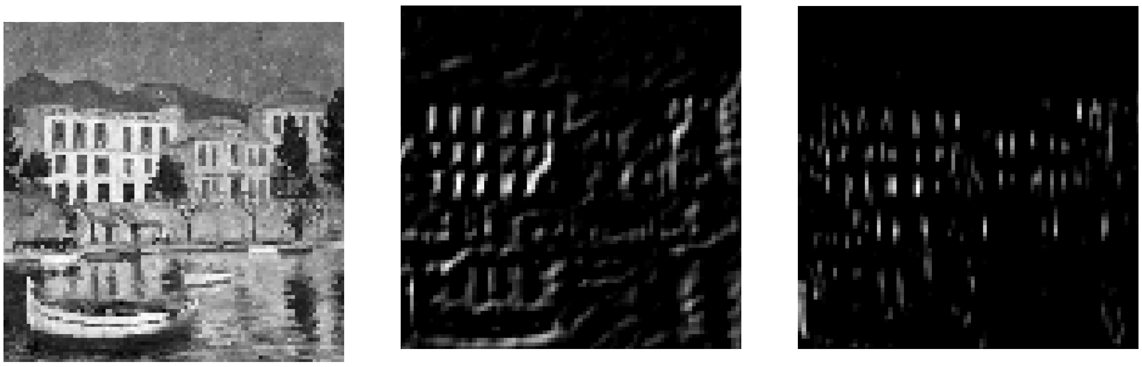


Figure A.1: Visualization of the filters learnt in the second convolutional layer of CNN. Various orientations of edges have been learned.

After being trained for 50 epochs, the training accuracy has reached 0.97, and its validation accuracy is 0.95. The test accuracy is evaluated as 0.94. The precision rate of recognizing landscape is 0.95, and the recall rate is 0.93. The precision rate of recognizing portrait is 0.93, and the recall rate is 0.95. Compared to HOG, CNN has a much higher precision and recall rate, and the reason might be that the features learnt include more than edge descriptors, which contribute significantly to the classification accuracy. Now, let's dig deeper into what CNN has learned during training.

The filters learned in the second convolutional layer are illustrated in Figure A.1. It can be seen that various orientations of edges have been learned, either horizontally, vertically or diagonally.

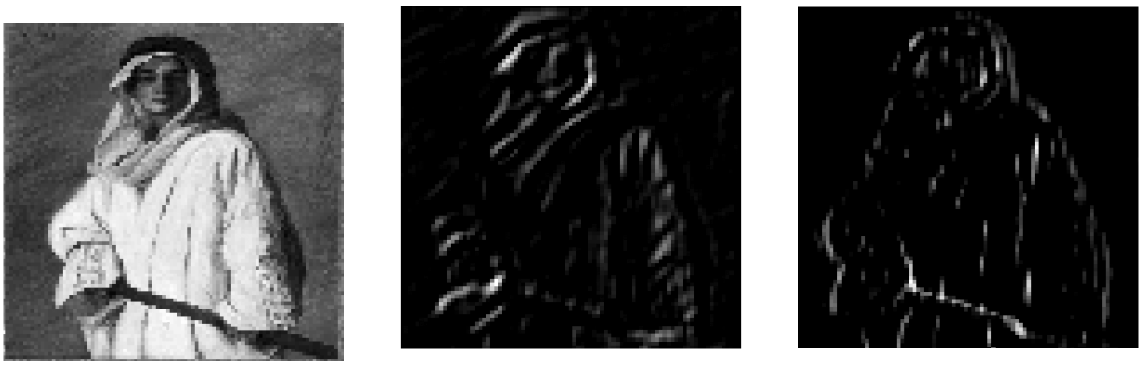
Further, to illustrate the activation of the these filters, the fourth and sixth filter highlighted in the first row of Figure A.1 have been selected, as they capture the diagonal and vertical directions. Visualization of activation can give us an intuitive view as to what these filters actually deal with in the original input image. The activation of a random landscape painting chosen from the test dataset is shown in Figure A.2, and the activation of a random portrait painting is shown in Figure A.3. As illustrated, the contours of the house, boat and human figure are captured by these edge filters.


(a) The input landscape image.

(b) The activation of fourth filter for diagonal edges.

(c) The activation of sixth filter for vertical edges.

Figure A.2: The activation of edge filters for a random painting of landscape.


(a) The input portrait image.

(b) The activation of fourth filter for diagonal edges.

(c) The activation of sixth filter for vertical edges.

Figure A.3: The activation of edge filters for a random painting of portrait.

Last, a CNN model has been trained with the same architecture and hyper-parameters, except that the images are not converted to grayscale, but remain in the color mode of RGB. The test accuracy is slightly improved to 0.95, which might indicate that the contrast of intensity contributes mostly to the classification of landscapes and portraits, and the hue of color is not a significant factor.

Appendix B: Dendrogram

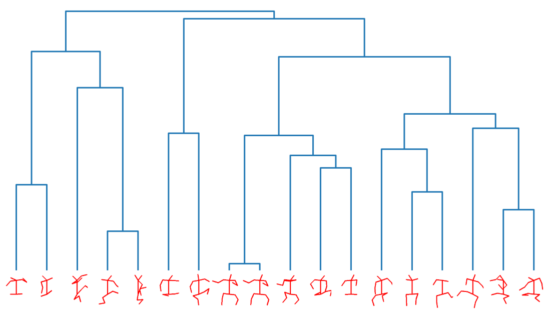


Figure B.1: All 18 poses drawn by Michelangelo from 15 paintings.

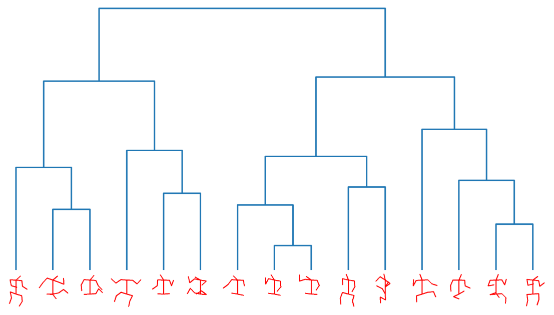


Figure B.2: All 15 poses drawn by Pierre-Auguste Renoir from 15 paintings.

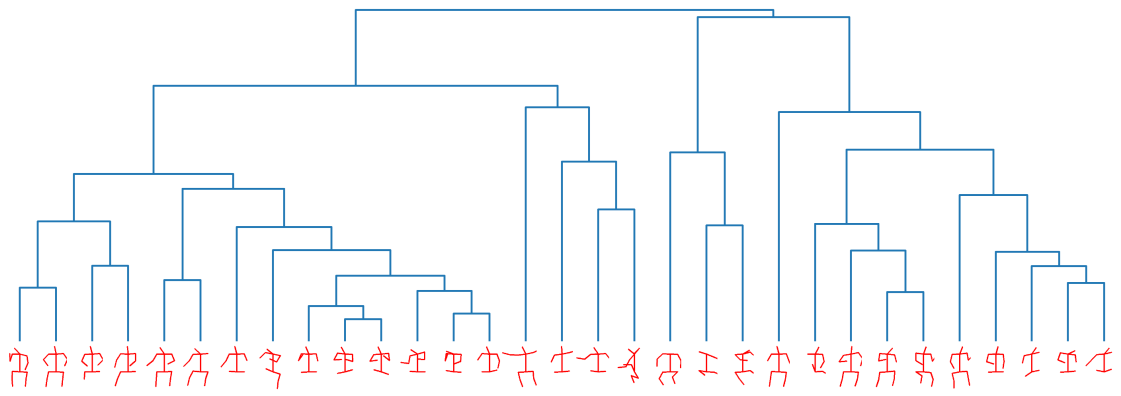


Figure B.3: All 31 poses drawn by El Greco from 15 paintings.

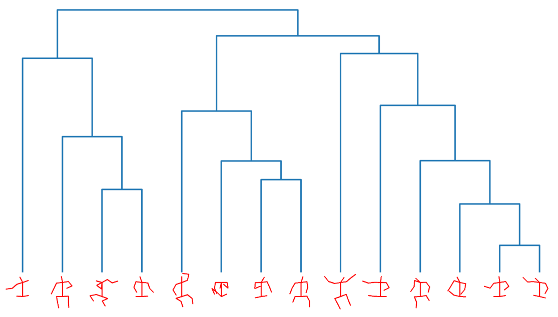


Figure B.4: All 14 poses drawn by Artemisia Gentileschi from 15 paintings.

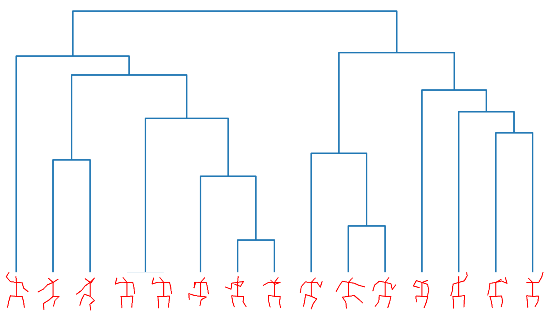


Figure B.5: All 15 poses drawn by Pierre-Paul Prud'hon from 15 paintings.

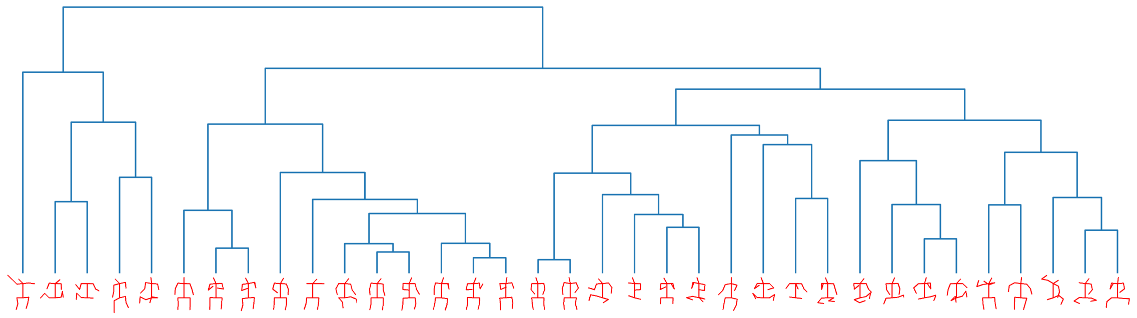


Figure B.6: All 35 poses drawn by Paul Gauguin from 15 paintings.

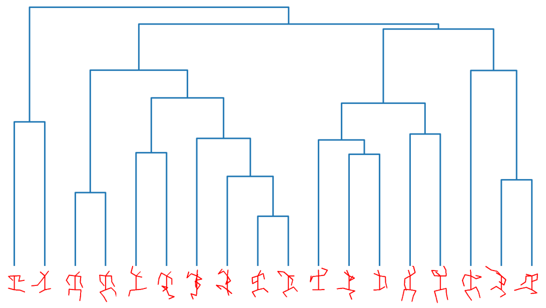


Figure B.7: All 18 poses drawn by Felix Vallotton from 15 paintings.

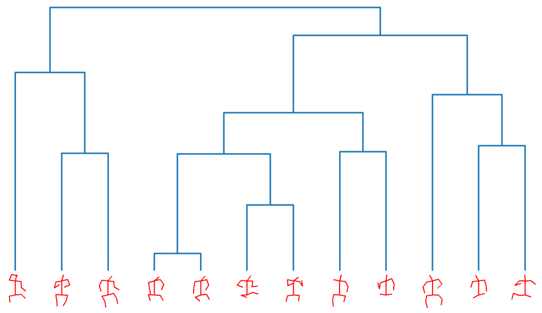


Figure B.8: All 12 poses drawn by Tamara de Lempicka from 15 paintings.

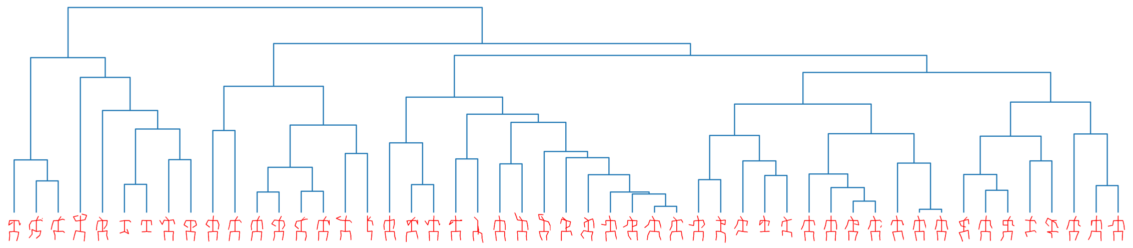


Figure B.9: All 51 poses drawn by Paul Delvaux from 15 paintings.

Bibliography

[1] Vitruvian man. https://en.wikipedia.org/wiki/Vitruvian_Man, 2021.

[2] T. Alldieck, M. Magnor, W. Xu, C. Theobalt, and G. Pons-Moll. Video based reconstruction of 3d people models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8387–8397, Jun 2018. CVPR Spotlight Paper.

[3] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele.

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

[5] H. Aviezer, Y. Trope, and A. Todorov. Body cues, not facial expressions, discriminate between intense positive and negative emotions. *Science*, 338(6111):1225–1229, 2012.

[6] P. Bachman, R. D. Hjelm, and W. Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pages 15535–15545, 2019.

[7] Y. Bengio and S. Bengio. Modeling high-dimensional discrete data with multi-layer neural networks. *Advances in Neural Information Processing Systems*, 12:400–406, 2000.

[8] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[9] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[10] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations, 2020.

[11] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 415–423, 2015.

[12] C.-J. Chou, J.-T. Chien, and H.-T. Chen. Self adversarial training for human pose estimation. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 17–30. IEEE, 2018.

[13] T. F. Cootes and C. J. Taylor. Active shape models—'smart snakes'. In *BMVC92*, pages 266–275. Springer, 1992.

[14] N. Dalal. *Finding people in images and videos.* PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006.

[15] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[16] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016.

[17] K. R. Echavarria, L. Dibble, A. Bracco, E. Silverton, and S. Dixon. Augmented reality (ar) maps for experiencing creative narratives of cultural heritage. In *EUROGRAPHICS Workshop on Graphics and Cultural Heritage (2019)*, 2019.

[18] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pages 2650–2658, 2015.

[19] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR, 2017.

[20] B. J. Frey, J. F. Brendan, and B. J. Frey. *Graphical models for machine learning and digital communication*. MIT press, 1998.

[21] 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*, pages 2414–2423, 2016.

[22] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. *Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester*, 2014(5):2, 2014.

[23] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[25] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3582–3589, 2014.

[26] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

[27] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[28] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680, 2014.

[29] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[30] D. Ha and D. Eck. A neural representation of sketch drawings, 2017.

[31] G. Hadjeres, F. Pachet, and F. Nielsen. Deepbach: a steerable model for bach chorales generation. In *International Conference on Machine Learning*, pages 1362–1371. PMLR, 2017.

[32] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[33] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[34] L. Impett and S. Süsstrunk. Pose and pathosformel in aby warburg's bilderatlas. In *European Conference on Computer Vision*, pages 888–902. Springer, 2016.

[35] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[36] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015.

[37] T. Jenicek and O. Chum. Linking art through human poses. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1338–1345. IEEE, 2019.

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

[39] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[40] D. E. King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[41] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[42] Y. Kolstee and W. Van Eck. The augmented van gogh's: Augmented reality experiences for museum visitors. In *2011 IEEE International Symposium on Mixed and Augmented Reality-Arts, Media, and Humanities*, pages 49–52. IEEE, 2011.

[43] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 29–37. JMLR Workshop and Conference Proceedings, 2011.

[44] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

[45] C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2479–2486, 2016.

[46] C. Li and M. Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *European conference on computer vision*, pages 702–716. Springer, 2016.

[47] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *arXiv preprint arXiv:1701.01036*, 2017.

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

[49] P. Madhu, A. Villar-Corrales, R. Kosti, T. Bendschus, C. Reinhardt, P. Bell, A. Maier, and V. Christlein. Enhancing human pose estimation in ancient vase paintings via perceptually-grounded style transfer learning. *arXiv preprint arXiv:2012.05616*, 2020.

[50] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[51] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[52] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.

[53] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[54] F. Noroozi, D. Kaminska, C. Corneanu, T. Sapinski, S. Escalera, and G. Anbarjafari. Survey on emotional body gesture recognition. *IEEE transactions on affective computing*, 2018.

[55] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu. Contrastive learning for conditional image synthesis. In *ECCV*, 2020.

[56] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[57] X. Peng and K. Saenko. Synthetic to real adaptation with generative correlation alignment networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1982–1991. IEEE, 2018.

[58] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *IEEE International Conference on Computer Vision*, 2015.

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

[60] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[61] E. Risser, P. Wilmot, and C. Barnes. Stable and controllable neural texture synthesis and style transfer using histogram losses. *arXiv preprint arXiv:1701.08893*, 2017.

[62] I. K. R{iza Alp Güler, Natalia Neverova. Densepose: Dense human pose estimation in the wild. 2018.

[63] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music. In *International Conference on Machine Learning*, pages 4364–4373. PMLR, 2018.

[64] C. Sagonas, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 397–403, 2013.

[65] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3681, 2013.

[66] B. Seguin, L. Costiner, I. di Lenardo, and F. Kaplan. New techniques for the digitization of art historical photographic archives-the case of the cini foundation in venice. In *Archiving conference*, volume 2018, pages 1–5. Society for Imaging Science and Technology, 2018.

[67] N. Shukla and K. Fricklas. *Machine learning with TensorFlow*. Manning Greenwich, 2018.

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

[69] K. Sun, B. Xiao, D. Liu, and J. Wang. Deep high-resolution representation learning for human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5693–5703, 2019.

[70] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 648–656, 2015.

[71] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.

[72] S. Tsuchida, S. Fukayama, M. Hamasaki, and M. Goto. Aist dance video database: Multi-genre, multi-dancer, and multi-camera database for dance information processing. In *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pages 501–510, Delft, Netherlands, Nov. 2019.

[73] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding, 2019.

[74] A. Van Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[75] V. Vlahakis, J. Karigiannis, M. Tsotros, M. Gounaris, L. Almeida, D. Stricker, T. Gleue, I. T. Christou, R. Carlucci, N. Ioannidis, et al. Archeoguide: first results of an augmented reality, mobile computing system in cultural heritage sites. *Virtual Reality, Archeology, and Cultural Heritage*, 9(10.1145):584993–585015, 2001.

[76] J. Yaniv, Y. Newman, and A. Shamir. The face of art: landmark detection and geometric style in portraits. *ACM Transactions on graphics (TOG)*, 38(4):1–15, 2019.

[77] H. Zhang, Q. Li, Z. Sun, and Y. Liu. Combining data-driven and model-driven methods for robust facial landmark detection. *IEEE Transactions on Information Forensics and Security*, 13(10):2409–2422, 2018.

[78] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[79] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[80] C. Zheng, W. Wu, T. Yang, S. Zhu, C. Chen, R. Liu, J. Shen, N. Kehtarnavaz, and M. Shah. Deep learning-based human pose estimation: A survey. *arXiv preprint arXiv:2012.13392*, 2020.

[81] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.