APPROVAL SHEET

Title of Thesis: PERSONALIZING APPAREL USING NEURAL STYLE TRANSFER

Name of Candidate: Prutha Dilip Date
M.S. in Computer Science,
2017

Thesis and Abstract Approved: ______________________
Dr. Tim Oates
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: ____________________________

ABSTRACT

Title of Thesis: PERSONALIZING APPAREL USING NEURAL STYLE TRANSFER

Prutha Dilip Date, M.S. Computer Science, May 2017

Thesis directed by: Dr. Tim Oates, Professor
Department of Computer Science and
Electrical Engineering

Convolutional Neural Networks have been highly successful in performing a set of computer vision tasks such as object recognition, object detection, image segmentation and texture synthesis. Gatys *et al.* (Gatys, Ecker, & Bethge 2015b) show how the artistic style of a painter can be extracted from an image of the painting and applied to another normal photograph, thus recreating the photo in the style of the painter. The method has been successfully applied to a wide range of images and has since spawned multiple applications and mobile apps. In this thesis, the neural style transfer method is applied to fashion to synthesize custom clothes. We create a personalization model that is able to generate new custom clothes based on a user's preference and by learning the user's fashion choices from a limited set of clothes from their closet. The approach is evaluated by analyzing the generated images of clothes and how they align with the user's fashion style.

Keywords: Convolutional Neural Networks, Personalization, Fashion, Neural Networks, Style Transfer, Texture Synthesis

Personalizing Apparel using Neural Style Transfer

by

Pratha Dilip Date

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
M.S. Computer Science
2017

Dedicated to my family, friends, and mentors

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Tim Oates for his invaluable guidance in my research. I am greatly indebted to his continuous support, patience, and motivation. I would like to express my sincere thanks to Ashwinkumar Ganesan for all his help in this thesis. His keen insight and observations were very crucial to the progress of my research. I would like to thank Dr. Hamed Pirsiavash for his valuable suggestions. I would also like to thank Dr. Kostas Kalpakis and Dr. Hamed Pirsiavash for agreeing to be on my committee.

I would like to express my deepest gratitude towards my parents, Dilip Date and Damayanti Date and my younger sister, Shreya Date for their unfailing support and encouragement throughout my life. Last but not the least, a big thanks to my friends who made my masters journey memorable.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

There has been significant interest and many advances in computer vision tasks like object recognition, detection and segmentation (Krizhevsky, Sutskever, & Hinton 2012; Ren *et al.* 2015; Chen *et al.* 2016). The revolution started with Krizhevsky *et al.* (Krizhevsky, Sutskever, & Hinton 2012) substantially improving object recognition in the ImageNet challenge using convolutional neural networks (CNNs). This was leveraged to conduct research and subsequent improvements in many tasks in the domain of fashion such as classification of clothes, predicting different kinds of attributes of a specific piece of clothing and improving the search and retrieval of clothes based on user's preference (Ziwei Liu & Tang 2016; Kalantidis, Kennedy, & Li 2013; Bossard *et al.* 2012; Xiao *et al.* 2015; Kiapour *et al.* 2014).

Giants of the e-commerce and online retail industry are expanding their investment in fashion. Recently, Amazon patented a system to manufacture clothes on-demand[1]. They also have started shipping their virtual assistant "Echo" with an integrated camera that clicks a picture of the user's outfit and rates its *style*[2]. They are working on providing styling assistance as well. A company called "StitchFix"[3] aims to simplify the user's experience

[1]https://www.recode.net/2017/4/18/15338984/amazon-on-demand-clothing-apparel-manufacturing-patent-warehouse-3d

[2]https://techcrunch.com/2017/04/26/amazons-new-echo-look-has-a-built-in-camera-for-style-selfies/

[3]https://www.stitchfix.com/welcome/schedule

for shopping online. It learns a user's style and provides a virtual fashion designer to put an outfit together for the user. As the online fashion industry looks to improve the kind of clothes that are recommended to users, understanding the customer's personal style preferences and recommending new custom designs becomes an important task.

Personalization and recommendation models are a well researched area with a number of methods from collaborative filtering (Linden, Smith, & York 2003) to content-based recommendation systems (e.g., probabilistic graph models, neural networks) as well as hybrid systems that combine both. Collaborative filtering (Linden, Smith, & York 2003) tries to analyze user's behaviour and preferences, and align users to predefined patterns and recommend a product. Content-based methods recommend a product based on the attributes or features that the user is searching for. Different kinds of features about the product or item are used to design the model. A hybrid system (knowledge-based system (Trewin 2000)) incorporates user preferences and product features to recommend an item. We attempt to build such a system that borrows styles from the user's own closet to create a template that will generate a new design. A neural network modelled to extract and transfer style superimposes the styles learned from a set of clothes on another piece of clothing to generate a new one.

Although Convolutional Neural Networks (CNN) provide state-of-the-art performance for multiple computer vision tasks, their complexity and opacity have been a substantial research challenge. Visualizing the features learned by the network, has been addressed in multiple efforts. Zeilar *et al.* (Zeiler & Fergus 2013) use a deconvolution network to reconstruct the features learned in each layer of the CNN. Simoyan *et al.*. (Simonyan & Zisserman 2014) backpropagate the gradients generated for a class with respect to the input image to create an artificial image (the initial image is just random noise) that represents what the network has learned about the class. The separation of style and content in an image as a part of the texture synthesis process (Gatys, Ecker, & Bethge 2015a) by Gatys *et al.*. shows the *variant* (content) and *invariant* (style) parts of the image.

An important part of our work is to generate new designs. Texture synthesis tries to learn the underlying texture of an image in order to generate new samples with the same texture. The research in this area is largely focused on parametric and non-parametric methods. Non-parametric methods try to resample specific pixels from the image or adopt patches from the original image to generate a new image (Efros & Leung 1999; Wei & Levoy 2000; Kwatra *et al.* 2003; Efros & Freeman 2001). Parametric methods define a statistical model that represents the texture (Julesz 1962; Heeger & Bergen 1995; Portilla & Simoncelli 2000). In 2015, Gatys *et. al.* (Gatys, Ecker, & Bethge 2015a) designed a new parametric model to synthesize texture from a picture using convolutional neural networks. They model the style of an image by extracting the feature maps generated when the image is fed through a pre-trained CNN, in this instance a 19 layer VGGNet (Simonyan & Zisserman 2014). They successfully separate the style and content of an arbitrary image and demonstrate how another image can be stylized using the textures of the prior. This model, however, does not retain the semantic content of an image. Building over the feature spaces used to extract texture, Gatys *et al.* extract style and content from two distinct images and recombine them to generate a new image that takes the shape from the original content image and design from the style image. Our system relies on the above algorithm to stylize and personalize clothes.

Our contribution in this thesis is a pipeline to learn the user's unique sense of fashion and generate new design patterns based on their preferences. We have built a personalization model that represents style as an average of individual style representations of every clothing item the user owns. The idea is to provide a baseline model for recommending personalized clothes to the user. Figure 1.1 shows a sample clothing item generated using *neural style transfer*. The first clothing item given by the user provides the shape for the new dress. The second is a piece of clothing initially provided by the user from his/her

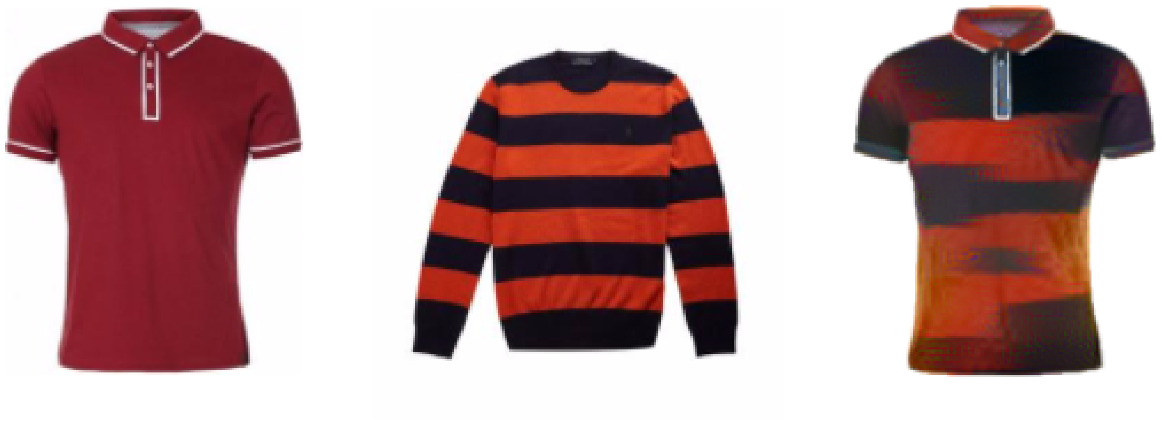


FIG. 1.1: (a) Clothing item providing the shape (b) Clothing item providing the style (c) Generated Clothing item

closet when the system attempts to learn their preference (one of the multiple style images used to generate the new image is shown here). The third is the final design generated for the user.

We discuss a few background concepts and related work in Chapter 2. Chapter 3 describes the dataset we use for experiments. Chapter 4 focuses on the detailed description of the architecture and methodology we use for carrying out the style transfer process and building our personalization model. Chapter 5 discusses the evaluation methods, experiments conducted, their results and observations. We conclude with Chapter 6 where we mention a couple of possibilities for future work.

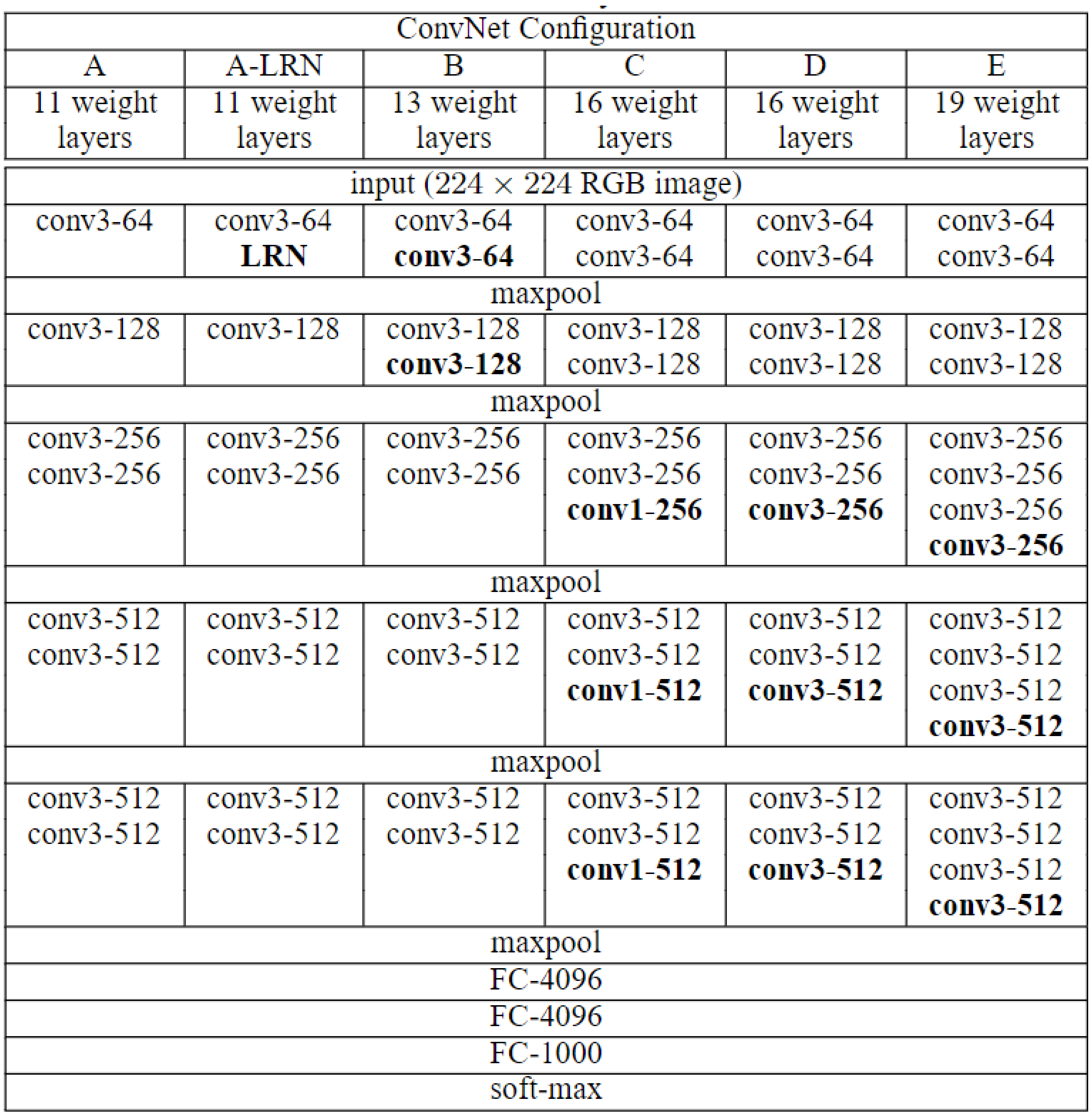
Chapter 2

RELATED WORK

2.1 VGGnet

In the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition", the authors K. Simonyan and A. Zisserman from the University of Oxford proposed a convolutional neural network model called the VGGnet (Simonyan & Zisserman 2014). They evaluated very deep convolutional networks for large-scale image classification to demonstrate how the depth of the convolutional neural network improves classification accuracy. With VGGnet, they showed that a conventional ConvNet architecture with substantially increased depth and constant filter size (with layerwise pre-training) can give state-of-the-art performance on the ImageNet challenge dataset (Krizhevsky, Sutskever, & Hinton 2012). The VGGNet models proposed are of varied depths of 11, 13, 16 and 19 weight layers. The 19 layer model (VGG-19) achieves 92.7% top-5 test accuracy on the ImageNet dataset of over 14 million images belonging to a 1000 classes. The 16 and 19 layer models are made publicly available and specifically these models generalize well to a wide range of datasets and tasks, matching or outperforming more complex object detection, recognition or classification pipelines.

The VGGnet configuration is as shown below in Figure 2.1. We use the VGG-19 model for style transfer as it is proven to perform better than the 16 layer version.



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

FIG. 2.1: VGGNet model configuration with 11, 13, 16 and 19 weight layers (Simonyan & Zisserman 2014)

2.2 Texture Synthesis

Gatys *et al.* use a convolutional neural network built for object recognition and classification to generate natural textures based on an example texture image (Gatys, Ecker, & Bethge 2015a). The goal is to infer a process to generate new samples of texture given a reference image. The new texture is generated using gradient descent from a white noise image to form a resultant image that has the same Gram-matrix representation of the original texture image. For this optimization, the loss (mean-squared distance) between the entries of the Gram matrix formed using the original image and the Gram matrix of the new texture image being generated is minimized. The authors attempted to combine the spatial information obtained from feature responses with the powerful CNN feature spaces of the model built for object recognition. Thus, they obtain a texture model that is parameterized by spatially invariant representations from the hierarchy of the network.

2.3 Neural Style Transfer

Gatys *et al.* describe the process of using image representations encoded by multiple layers of convolutional neural networks to separate, to some extent, the content and style of images and recombining them to form new visually appealing images (Gatys, Ecker, & Bethge 2015b). The idea of style transfer is based of the texture synthesis process. The texture model described in Section 2.2 is able to represent the spatial information in images but fails to give the image's semantic content. Using the feature spaces built to isolate texture, Gatys *et al.* extracted style across all layers from a pretrained convolutional neural network (Gatys, Ecker, & Bethge 2015b; 2016). The content is obtained by the feature representation of the image given by the higher layers in the model. A new artistic image is generated by minimizing style and content using the LBFGS optimizer and transforming a white noise image to possess activations similar to those of the style and content images.

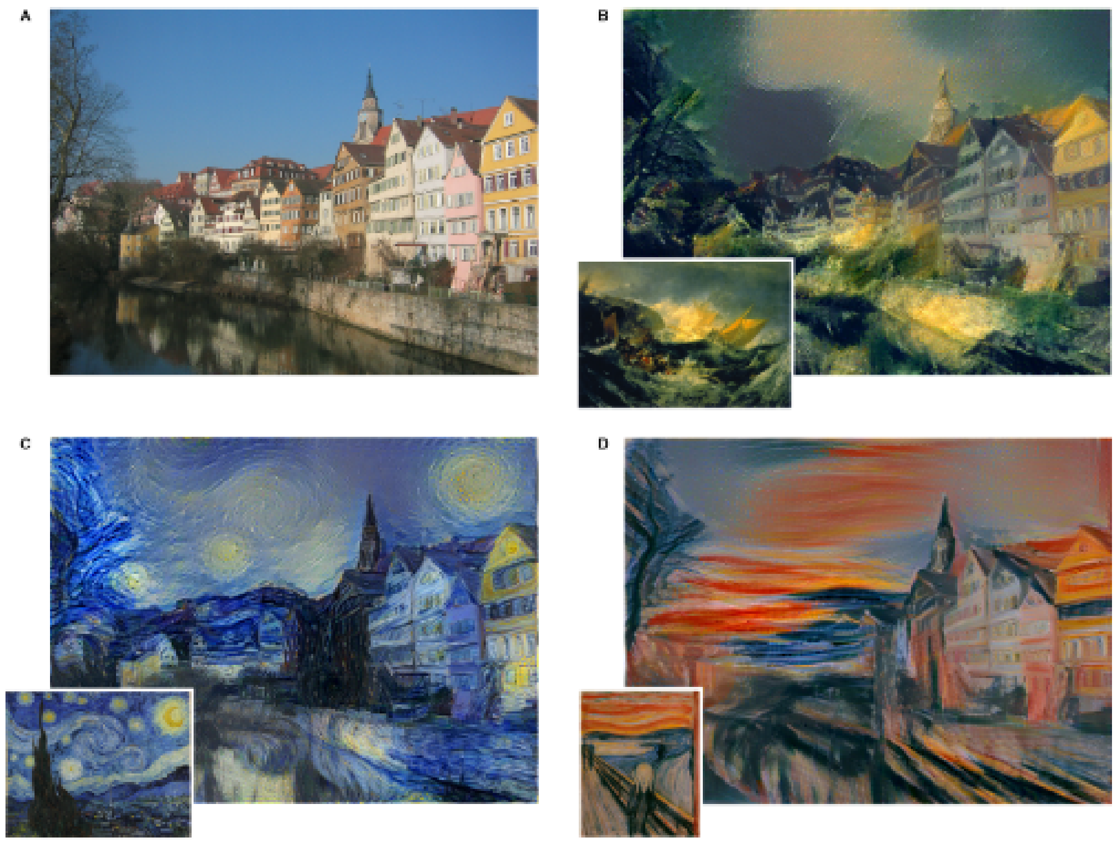


FIG. 2.2: Example for neural style transfer on artistic images.

The original photograph of the Neckarfront in Tübingen, Germany shown in (a) (Photo:Andreas Praefcke) is reinforced with styles of the paintings - (b) The Shipwreck of the Minotaurby J.M.W.Turner, 1805 (c) The Starry Night by Vincent van Gogh, 1889. (d) Der Schreiby Edvard Munch,1893. The painting that provides the style for the respective generated image is shown in the bottom left corner of each panel. (Gatys, Ecker, & Bethge 2016)

2.4 Real-time Style Transfer

Image transformation tasks use a feed-forward convolutional neural network trained in a supervised manner to compute a per-pixel loss function for measuring the difference between the result and original ground-truth images. Although efficient at test-time, the per-pixel loss functions fail to capture the perceptual information in the images. Another way to approach the image transformation problem is to use perceptual loss functions that are based on the differences between high level feature representations of the images obtained from pre-trained CNNs rather than differences between pixels. Images are generated by minimizing this loss. This, however, is a time consuming process since it involves solving an optimization problem. An example of such a transformation process is the neural style transfer as described in Section 2.3. Johnsos *et al.* discuss the results of the image transformation task by combining both the mentioned approaches (Johnson, Alahi, & Fei-Fei 2016). A feed-forward transformation network is trained to run real-time using perceptual loss functions that depend on high-level features from a pre-trained loss network rather than the per-pixel loss function based on low level pixel information. The authors test the experiment on the style transfer and single-image super-resolution tasks. The results, typically for style transfer, are shown to be qualitatively equivalent but faster by three orders of magnitude than the prior work.

Although efficient in terms of execution time, we abstain from using this approach as it does not allow us to separate and store the style loss functions that are unique to a user.

Chapter 3

DATASET

This chapter describes the dataset used in this thesis. Multiple datasets provide a cache of fashion pictures with appropriate labels. Prior work has been addressed segregating images of clothing pieces from existing datasets. The “Fashion 10000” dataset (Loni *et al.* 2014) is created by filtering and annotating images from the Flickr dataset that are associated with any term remotely related to fashion. “Street2Style” dataset (Hadi Kiapour *et al.* 2015) contains consumer images of street apparel. These datasets span a wide range of clothes. However, the quality and mix of images they offer are not very helpful to us. We use the *“DeepFashion”* dataset as it is found to be the most extensive and suitable to our requirement.

3.1 DeepFashion Dataset

The DeepFashion dataset is created by (Ziwei Liu & Tang 2016). It has over 800,000 images of different kinds of apparel that are a mix of well-posed pictures taken for retail and stores, street snapshots and unconstrained photos of consumers. The images are richly annotated for a variety of attributes, categories of apparel, bounding boxes and clothing landmarks.

We are particularly interested in the annotations for attributes. Using the large set of images, the authors built a deep learning model that predicts attributes for all these images. This is a subset of the DeepFashion dataset containing about 290,000 clothes, 50 clothing categories and 1000 clothing attributes. Each image is annotated with a bounding box and clothing type.

The 1000 clothing attributes are one of the 5 types: Texture, Fabric, Shape, Part or Style. Texture mostly consists of the prints and patterns on the clothing object, for example, polka dotted, butterfly print, abstract, etc. Fabric is the material of the apparel, e.g., Chiffon, Fur, Lace. The cut, length and adornments are collectively defined as the Shape of the clothing, e.g., Surplice, Halter, Slit. Part-related attributes describe apparel that has some distinguishing factor pertaining to a body-part. For example, a crisscross back dress or a V-neck blouse. The Style of a clothing object is defined by its type or graphic on it. A baseball T-shirt would be annotated as Athletic while a NYC graphic T-shirt would be annotated with the style Graphic or NYC. The style aspect here is purely based on the print and does not intend to represent any individual style. To give an example, jogger pants are not annotated as Athletic even though it would be very appropriate to consider this clothing item under the mentioned style.

The clothes in every image may belong to one of the 3 categories: Upper body clothing (Blouse, Coat, etc.), Lower body clothing (Jeans, shorts, etc.) and Full body clothing (Dress, Jumpsuit, etc.). An example of annotations available is shown in Table 3.1.

Table 3.1: An example of annotations for Figure 3.2

Image name	Crochet_Lace_Flounce_Dress/img_00000023.jpg
Image category	Dress
Attributes	Crochet, Lace
Attribute type	Fabric
Clothing category	Full-body
Bounding box	041 001 274 300

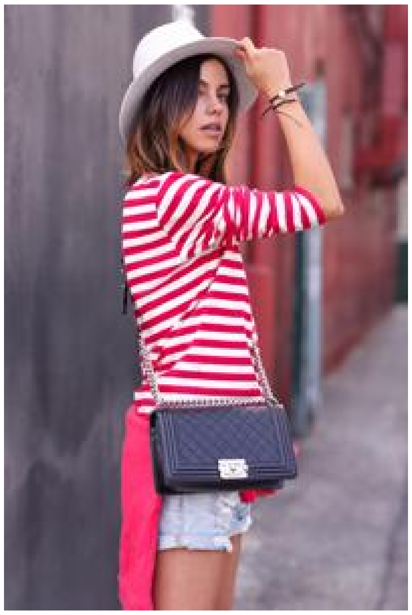


FIG. 3.1: Texture related attribute (type 1): Striped Top

While this is a large-scale fashion dataset, a subset of around 850 images are selected for our research. As mentioned earlier, the dataset contains a mix of pictures with people and objects other than clothes taken in a natural setting We carefully selected images with a white background since we wanted the model to learn features of the clothing object alone. Objects in the background would adversely impact style extraction. Images of only the upper body and full body clothing categories were picked. Attempting to transfer attributes of lower body clothes over objects belonging to any other category would create huge variations in shape and give unappealing results. Moreover, images were filtered over attribute types and specifically those that possess the texture and fabric related attributes were chosen. Considering the rest of the attribute types would mean digressing from the main goal of applying neural style transfer over apparel in terms of patterns and textures. Dealing with shape or style would be a whole new area of research and is not in the scope of this thesis. In this manner, 800 such images were segregated manually with random selection to form 4 closets with 200 images per closet - 100 for training and 100 for testing.

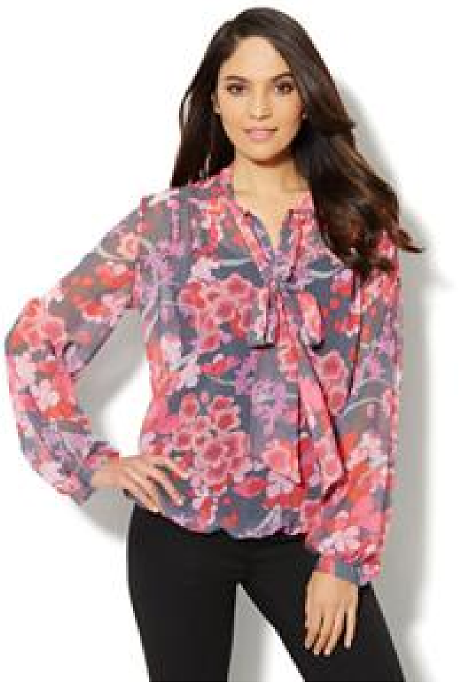


FIG. 3.2: Fabric related attribute (type 2): Chiffon Blouse

Another 150 images were picked in the same manner for providing content.

3.2 Assumptions

To summarize, with regards to the dataset, the entire process of personalization and style transfer explained in 4 is based on the following assumptions:

- Images should have white background.
- Images should contain only clothing objects with no humans or other artifacts.
- Images are of only upper-body or full-body apparel pieces.
- Style is learned only for attribute types texture and fabric. The clothing shape is not considered as a representation of the style of that object.

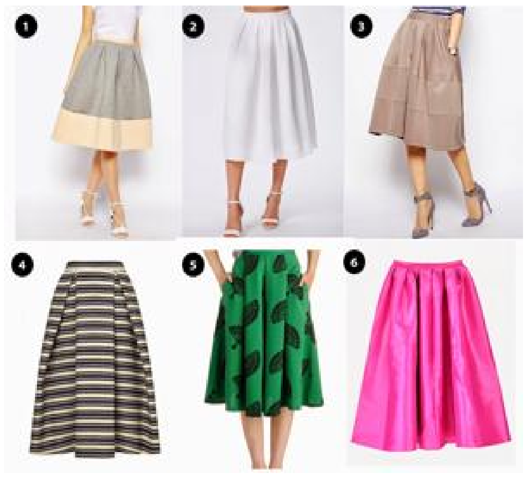


FIG. 3.3: Shape related attribute (type 3): Midi skirt

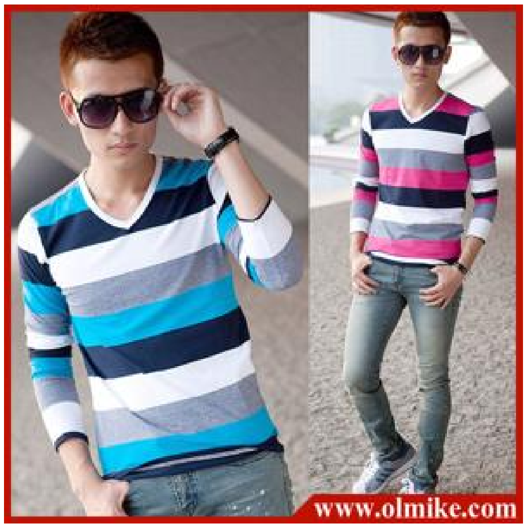


FIG. 3.4: Part related attribute (type 4): V-Neck tee

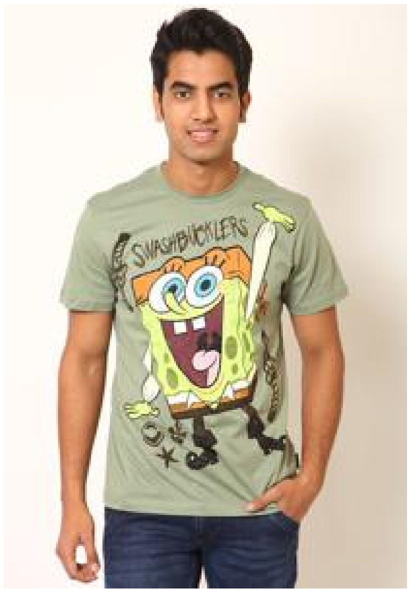


FIG. 3.5: Style related attribute (type 5): Spongebob Squarepants shirt

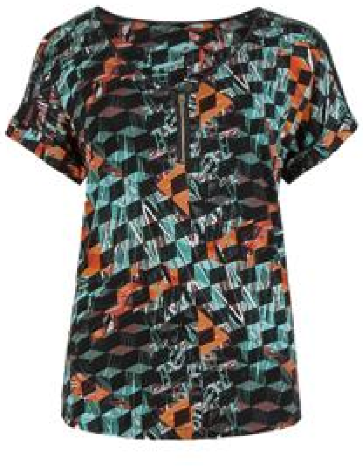


FIG. 3.6: White background image of clothing object with texture-related attribute and upper-body clothing category

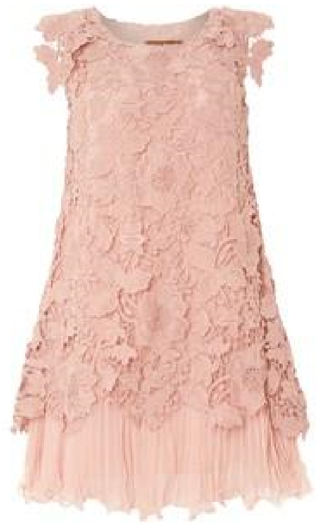


FIG. 3.7: White background image of clothing object with fabric-related attribute and full-body clothing category

Chapter 4

ARCHITECTURE AND METHODOLOGY

This chapter presents the architecture of the entire system. We describe the style and content extraction processes in Section 4.1. The process of selecting style functions and applying them over images of clothing objects is explained in Section 4.2. We need to process the images before and after passing them through the model. Section 4.3 talks about preparing images for the style transfer process. The final touches given to the output images are described in Section 4.4.

Figure 4.1 shows the overview of our system's architecture. Consider a simulated closet having images of clothing items representing the user's style. We input these images along with a content image to lend style and content, respectively, to the new clothing design. Initially, the images may or may not be of the same size, and that impacts the quality of style transfer significantly. We preprocess the images to bring them to a uniform size. We then learn the user's fashion choices from the set of images forming his/her closet. In the space of neural networks, the process of style transfer (Gatys, Ecker, & Bethge 2015b) represents style as Gram matrices and enables the separation of style and content from an image. We extract styles of the clothing pieces in the form of Gram matrices and store them for later use to generate a new style. For the process of personalization, the user inputs clothing attributes to state the style preference. We retrieve Gram matrices for these attributes and use an averaging model to compute the overall representation of style. We then reinforce this style on the content extracted from the content image provided earlier, and postprocess it to generate a new personalized clothing design. The details of the style transfer and personalization process are given in the following sections.

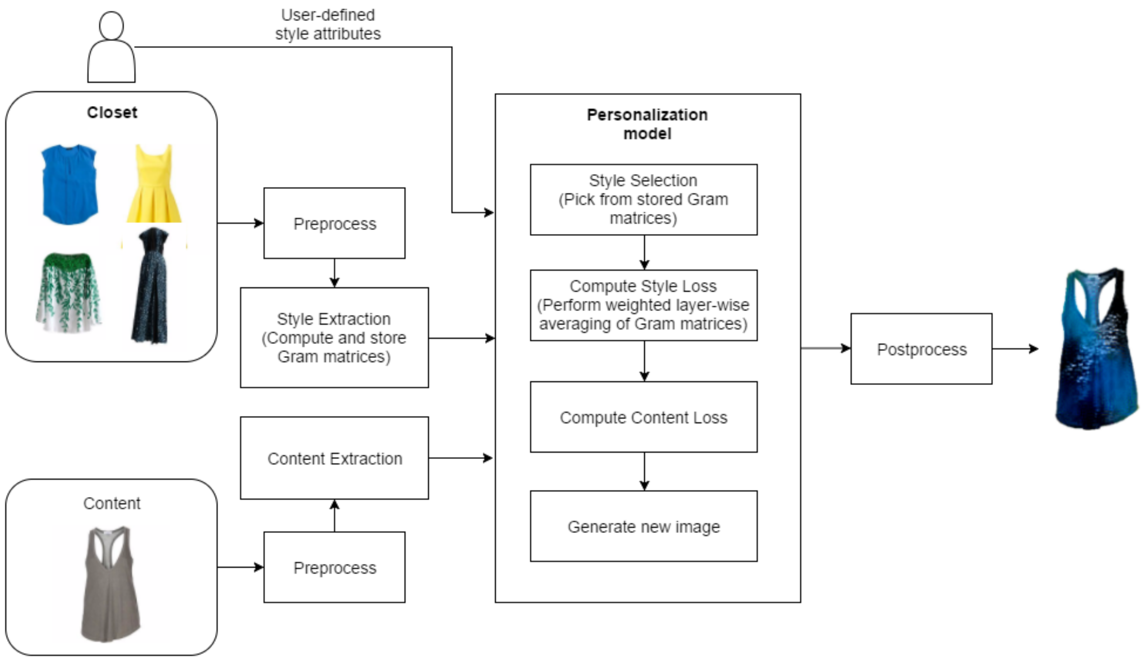

User-defined style attributes
Closet
Preprocess
Style Extraction (Compute and store Gram matrices)
Content
Preprocess
Content Extraction
Personalization model
Style Selection (Pick from stored Gram matrices)
Compute Style Loss (Perform weighted layer-wise averaging of Gram matrices)
Compute Content Loss
Generate new image
Postprocess

FIG. 4.1: Block diagram of the system architecture

We use the implementation given by (Smith 2016) for the model described in the paper "Image Style Transfer using Convolutional Neural Networks" (Gatys, Ecker, & Bethge 2016) as the reference implementation for style transfer. The model is a pre-trained 19 layer VGGnet model (VGG-19) that takes a content image and a set of style images as input. We work with feature responses of the images encoded through a combination of layers. The fully connected layers are not used.

4.1 Preliminaries

We use the notations and equations given by (Gatys, Ecker, & Bethge 2015b) to explain style and content extraction. For an input image $\overrightarrow{x}$, every layer l in the convolutional network has N_l distinct filters. Each filter produces a feature map of size M_l where the size is given by the image's height times it's width. Thus, the feature maps at every layer l can be given as $F^l_{ij} \in \mathcal{R}^{N_l \times M_l}$ where F^l_{ij} represents the activation of the i^{th} filter at position j. Using these features maps, we reconstruct the content and extract the style from the images passed. The process is explained in detail in subsequent sections. The model can be initialized with one of the content or style images, or a random white noise image. The white noise image produces an arbitrary number of distinct images. Using a fixed image assures the same output. We thus use the content image to initialize the model.

4.1.1 Style extraction

The style representation of an image is obtained by using the features spaces constructed to capture texture information as explained in Section 2.2. These features spaces are an indication of how likely it is for two activations for a given style image to coexist. Such a feature correlation is given by the Gram matrix, $G^l \in R^{N_l \times N_l}$ where G^l_{ij} is calculated by multiplying two flattened vectors of feature responses of i and j for layer l:

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

Any value in the Gram matrix closer to 0 indicates that the two features do not apply at the same time while the one further away from 0 indicates that the features would be applied simultaneously.

We use the 5 pooling layers for style extraction. A feature representation of the style image at every layer is stored as A^l. Gram matrices, G^l, are generated at every style layer of the model initialized with white noise. The style loss function is then computed for every layer as the mean squared error between G^l and A^l as,

$$E_l = \frac{1}{4M^2N^2}\sum_{i,j}(G^l_{ij} - A^l_{ij})^2$$

This makes up the first phase of the entire style transfer process where we store Gram matrices as representation of style. However, in the reference implementation we use, it is easier to store style loss functions directly rather than storing Gram matrices as mentioned earlier. We thus store the layer-wise style loss function for every image, to be used later for personalization. Figure 4.2 summarizes this process graphically.

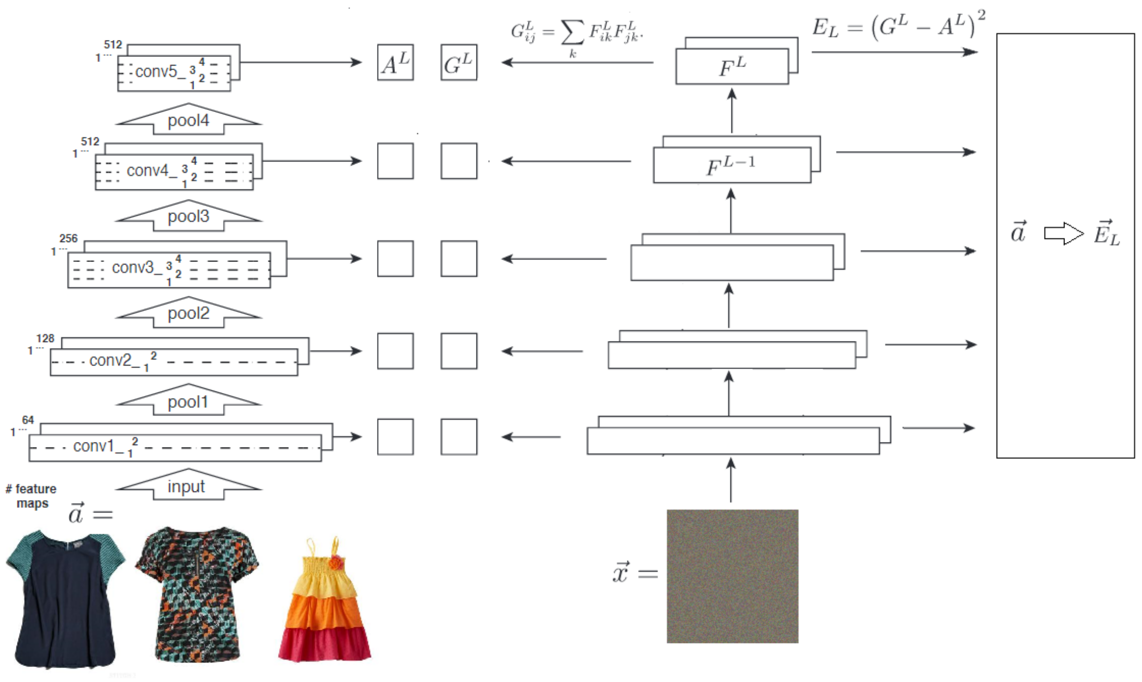


FIG. 4.2: Style extraction and storing loss functions

4.1.2 Content extraction

The feature responses of higher layers in the model give a representation of the image that is more biased towards the content rather than the texture. We use the feature representations of the *conv_4_2* layer to extract content. Given the feature representations in layer l of the original image $\overrightarrow{p}$ and the generated white noise image $\overrightarrow{x}$ as P^l and F^l respectively, we define the content loss as the mean squared difference between the two:

$$\mathcal{L}_{content}(\overrightarrow{x}, \overrightarrow{p}, l) = \frac{1}{2}\sum_{i,j}(F^l_{ij} - P^l_{ij})^2$$

The derivative of this loss with respect to the activations at layer l gives the gradient used to minimize the loss:

$$\frac{\partial \mathcal{L}_{content}}{\partial F^l_{ij}} = \begin{cases} (F^l - P^l)_{ij}, & if F^l_{ij} > 0 \\ 0, & if F^l_{ij} < 0 \end{cases}$$

Using gradient descent, we optimize the white noise image to have the same feature representations as that of the original image. This process is shown in Figure 4.3.

4.2 Personalization Process

After style extraction, we move to the actual personalization process. It involves picking the styles matching a user's preferences, computing the total style loss value across the selected styles and then carrying out the style transfer process.

4.2.1 Style selection

We feed the model certain texture and fabric related attributes to use for style transfer as input. From the stored style functions, we pick the ones that map to images containing

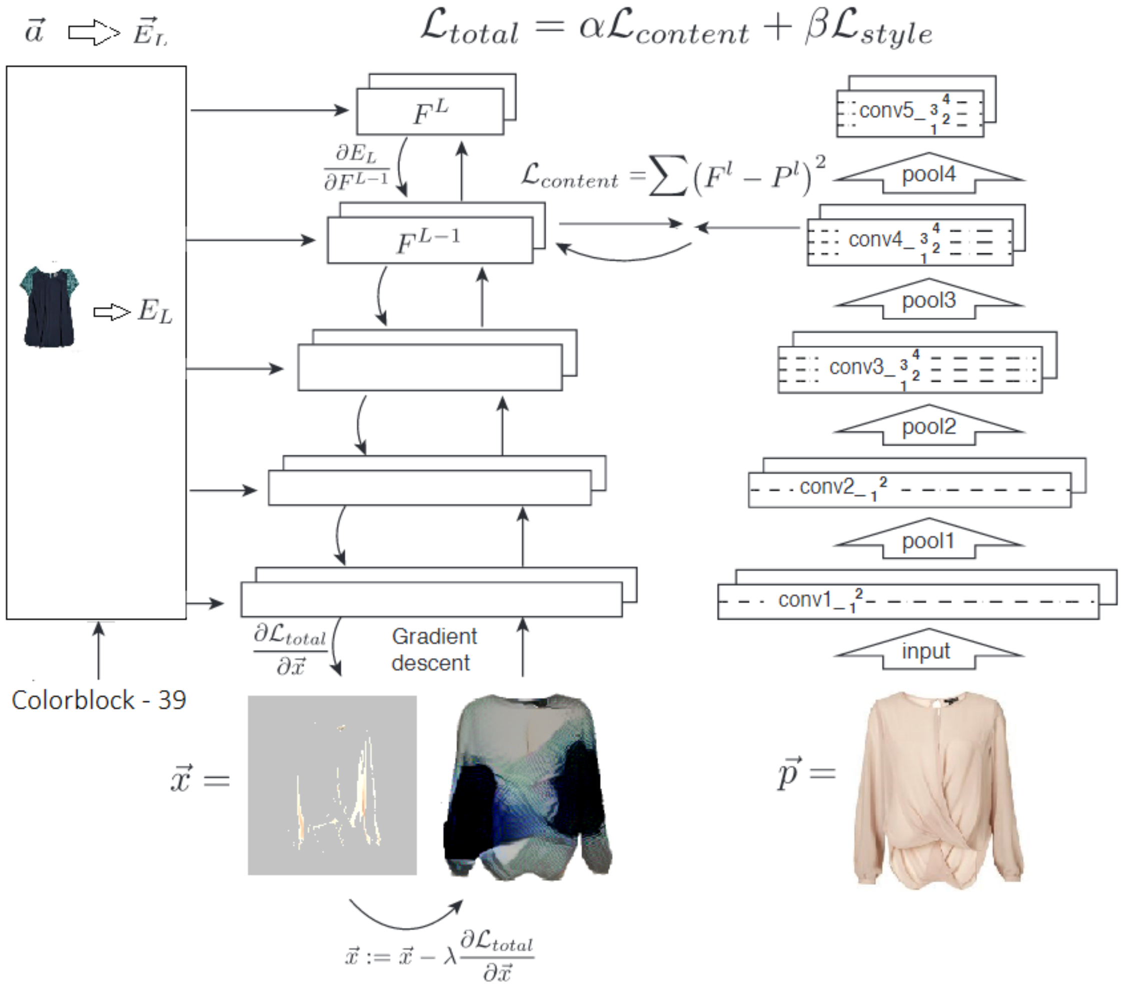

$\vec{a} \Rightarrow \vec{E}_L$
$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
F^L
$\frac{\partial E_L}{\partial F^{L-1}}$
F^{L-1}
$\mathcal{L}_{content} = \sum(F^l - P^l)^2$
$\Rightarrow E_L$
conv5_
pool4
conv4_
pool3
conv3_
pool2
conv2_
pool1
conv1_
input
Colorblock - 39
$\frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$
Gradient descent
$\vec{x} =$
$\vec{p} =$
$\vec{x} := \vec{x} - \lambda \frac{\partial \mathcal{L}_{total}}{\partial \vec{x}}$

FIG. 4.3: Content extraction and style transfer

the attributes specified. The number of images for every attribute picked depends on the distribution of the attribute across the entire list of images present. This makes sure that style transfer is biased towards the attributes specified, suppressing the extra ones every image brings with it. Thus, the style of images for only those attributes is reinforced in the resultant image.

4.2.2 Averaging Model & Style transfer

We combine all the styles selected for the user defined attributes to get a generic style representation of the user's fashion choices. This is done by computing an average over the Gram matrices for those styles to give a resultant Gram matrix representing the chosen styles collectively. However, as mentioned previously, our implementation makes it easier to work with style loss functions instead of the raw Gram matrices. We thus use the averaging function as the model for personalization with the assumption that the average of the style losses is equal to the loss computed over the average of style representations of images. We take a weighted, layer-wise average over the style losses for every style image selected. This gives us the final style loss value.

For a style image $\overrightarrow{a}$ and the initialized image $\overrightarrow{x}$, the style loss can be given as,

$$\mathcal{L}_s(\overrightarrow{a}, \overrightarrow{x}) = \sum_{l=0}^{L} w_l E_l$$

where $\mathcal{L}_s$ is the style loss for one image. We average the style losses computed for every image to get our final style loss value.

$$\mathcal{L}_{style} = \sum_{s=0}^{S} W_s \mathcal{L}_s$$

Here, $\mathcal{L}_{style}$ is the style loss computed over S images using the averaging model.

The derivative of the style loss E_l with respect to the activations at layer l can be given as,

$$\frac{\partial E_l}{\partial F^l_{ij}} = \begin{cases} \frac{1}{N_l^2 M_l^2}((F_l)^T)(G^l - A^l)_{ji}), & if F^l_{ij} > 0 \\ 0, & if F^l_{ij} < 0 \end{cases}$$

This can be used easily to calculate the gradient in a standard error back-propagation way.

We calculate the total loss as the addition of the style and content losses obtained. We can bias the output image towards the content or the style by defining weights for each of the loss values. The total loss can now be denoted as,

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content}(\overrightarrow{p}, \overrightarrow{x}) + \beta \mathcal{L}_{style}(\overrightarrow{a}, \overrightarrow{x})$$

Here, α and β are the weights assigned to the content and style losses respectively. We then perform gradient descent over the initialized image $\overrightarrow{x}$ to minimize the total loss value. This optimization is done using the LBFGS optimizer. Gatys *et al.*'s work proves to get better results with LBFGS than the Adam optimizer.

The output image is then postprocessed to get the final image. Figure 4.3 shows the process of selecting style functions to use for style transfer. The content and style losses are computed and minimized.

4.3 Preprocessing

Since the personalization process works in two phases, it is necessary for all the images to be on comparable scale. We transform all non-uniform size images to a standard 512 x 512 size. We first create a temporary white background image of the mentioned size. We then place the original image at the center of the temporary image created. This resizes the image to the expected size without deforming it.

We also extract and store the mask of the original content image using the 'grab cut' utility (Rother, Kolmogorov, & Blake 2004). This mask is used in the postprocessing step to get rid of patterns lying beyond the contours of the apparel.

4.4 Postprocessing

The output image contains patches of patterns transferred across the entire image. We resize the image to its original dimensions and apply the mask extracted by the grab cut tool in the preprocessing step to white out the background and get the transformed clothing object as the final result image.

Chapter 5

EXPERIMENTS AND OBSERVATIONS

In the previous chapter, we described our system architecture to generate new stylized images. Here, we discuss the evaluation metrics and their results. Evaluation of recommendation systems is a difficult challenge. One option to verify if the personalization of clothing is achieved is using a Mechanical Turk survey to measure how close the generated images are to the clothes from a sample user's closet. But opinions about clothes differ widely among users. Also, the annotations over our dataset are not 100% accurate and may lead to a discrepancy between the user's knowledge of attributes and what's actually there. Hence, we perform a different experiment where a user's wardrobe is simulated and we check how close the styles of the generated images are to the styles of these simulated wardrobes as a validation of the user's preferences being applied.

We present two approaches to the evaluate results of personalization using style transfer.

5.1 Quantitative Evaluation

We extract image features from a VGG-16 model and pass them through a linear Support Vector Machine (SVM). The attributes that represent the styles transferred on the content images are expected to be predicted by the classifier. We present the following approaches to train the classifier and verify if clothing images are personalized for the given styles.

5.1.1 Predicting Attribute labels

One way to evaluate if style is imparted in the given content image is to verify if the classifier is able to identify style attributes present in it. We train our SVM to learn attributes such as floral print, polka dots, etc. to predict one or a combination of these for the new generated images. If the predicted attributes are same as those picked for style transfer, we say that the expected personalization is achieved.

Predicting multiple attributes for an image makes it a multi-label classification problem and such problems, typically, are hard to evaluate. We use a 1-vs-all classifier in this scenario, where a binary classifier is built for every class, in this case, the attribute labels. Our data is formatted where every image is associated with a set of style attributes that represent its style. Originally, the distribution of attribute labels was heavily biased to one attribute, 'print'. We observed that 80% of the images had this attribute associated with them along with a few more labels. The classifier was mispredicting attributes for all images. Hence, we ignore that attribute during classification altogether. We also found the data to be skewed for some other attributes. Thus, we randomly sampled images to get a uniform distribution of attribute labels across our data. Figure 5.1 shows the evaluation model for predicting attribute labels on images separate for training and test data generation.

The training data is separate from the set of images used for style extraction. We compute the f1-score for the model for 600 training images and 50 test images using styles from another 250 images. Although it surpasses the baseline f1-score of 0.04 by a huge margin, the f1-score of 0.39 on the validation set and 0.13 on actual test images does not give a good prediction. Figure 5.2 represents these statistics graphically. However, if we

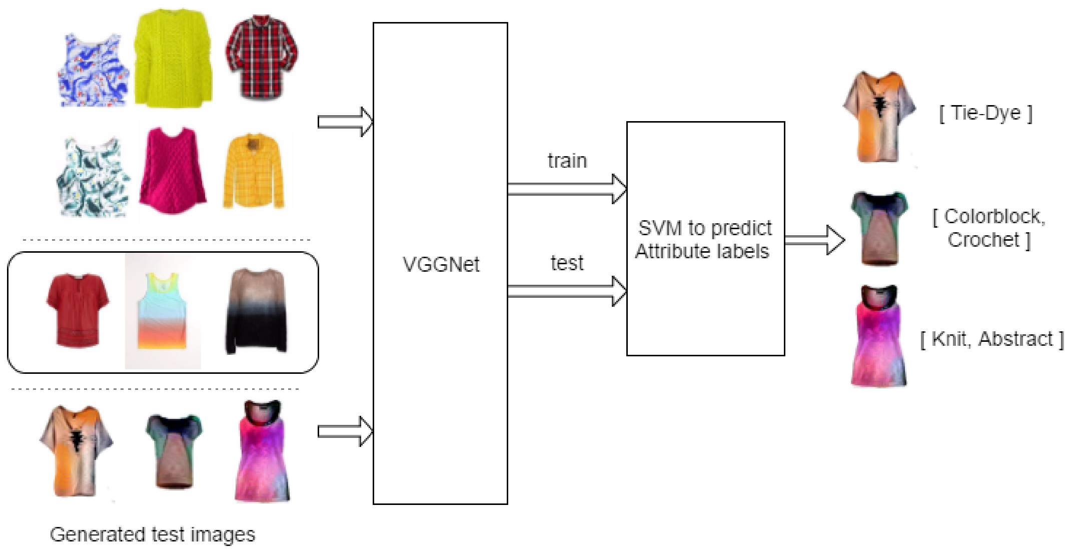


FIG. 5.1: Evaluation model for predicting attribute labels on separate training and test generation images

reduce the training data size to 400, and use 4 sets of 100 images for style extraction with 140 test images, the classifier does well with an f1-score of 0.54 as shown in Figure 5.4. We observe that using 250 images to extract style as opposed to just one hundred at a time affects the number of style images picked for style transfer. More style images impact the quality of the result, thus making it difficult to identify patterns.

In another approach, we build a classifier to predict attribute labels, but unlike the previous method there are no distinct training and style images. We train the classifier with the same set of style images used for personalization. The styles imparted in the new clothing image are essentially extracted from a predefined set of images. Thus, it is justified to train the classifier on the same set images used for personalization. We take two distinct sets of images, one to extract style from and the other to extract content from. The result images form our test data. We consider personalization to be successful if the style attribute labels predicted for the transformed images match the ones used earlier for style extraction.

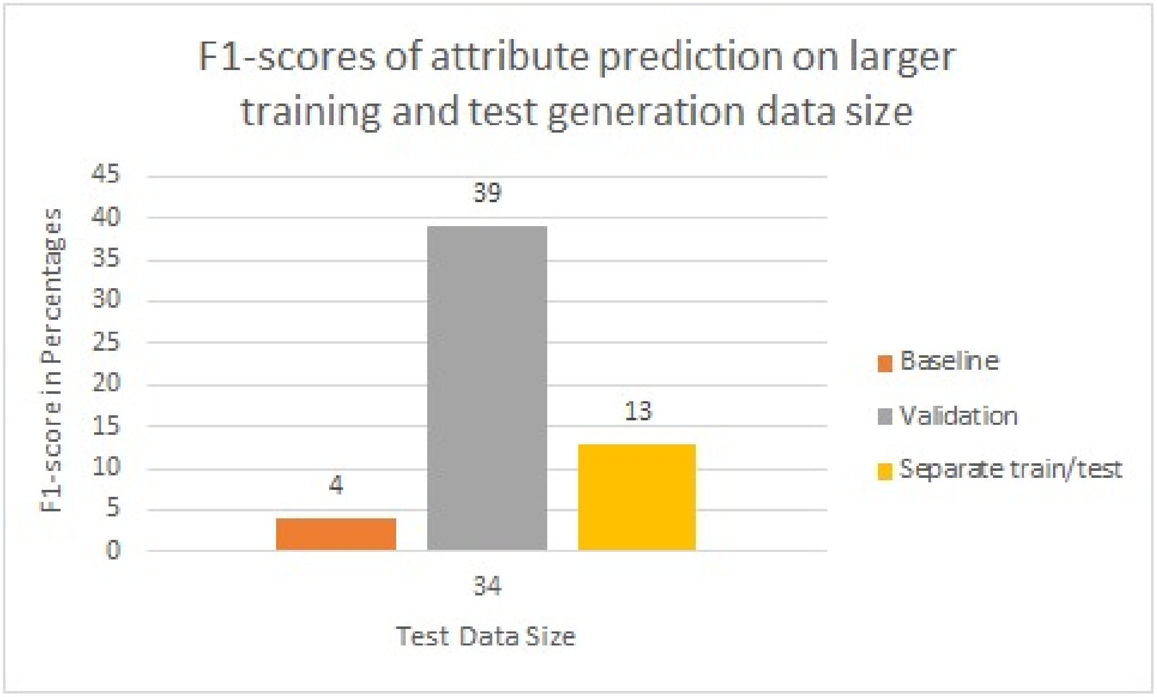


FIG. 5.2: Bar-chart showing F1-scores for the baseline, our model on validation data and actual test data

Our training set consists of 400 images, split into sets of 100 for style extraction and 140 test images. For a baseline, we consider predictions made using the simplest classification method that considers a uniform distribution of class labels across all training images. We get an f1-score of 0.58 for our model and 0.34 for the baseline. However, we see a good improvement in the accuracy if we increase the size of our test data. The number of true positives predicted are directly proportional to the test data size and that makes the accuracy go higher.

In the first approach, we train the classifier to learn patterns. The generated image is a blend of multiple styles which are difficult to trace individually. Whereas, when styles are extracted from the training images itself, the intrinsic properties of the content of these images are transferred as well. It becomes easy for the classifier to find similarity between

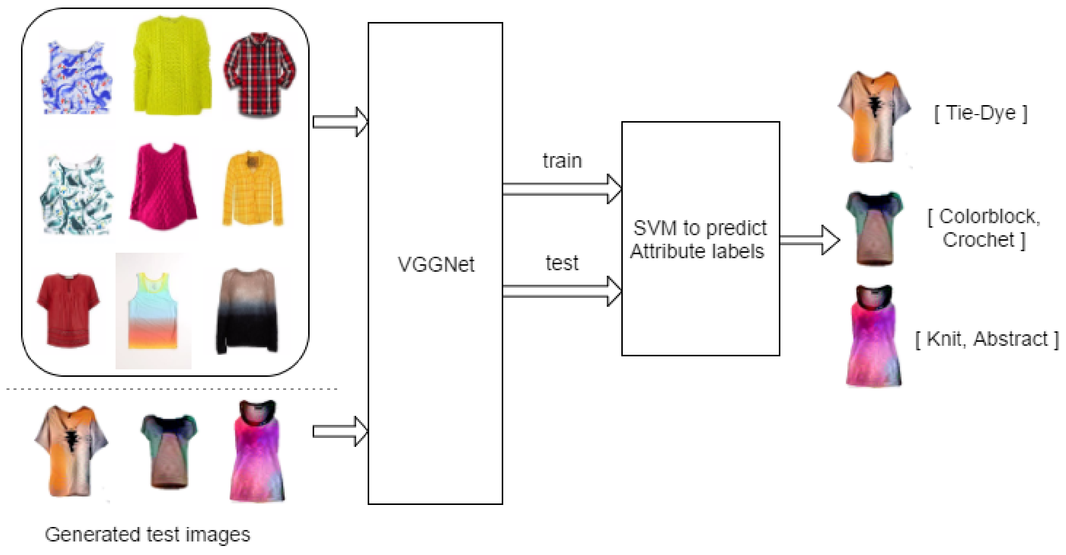


FIG. 5.3: Evaluation model for predicting attribute labels on same training and test generation images

these training and test images. Hence, we get better accuracy for the second approach.

5.1.2 Predicting Closet Ids

We experimented with a second approach to classification, but forfeited it due to low accuracy. With the assumption that a person can own around 100 pieces of clothing items on an average, we form 4 sets of one hundred images each. Each image has attribute labels and closet ids associated with it. The attribute labels are uniformly distributed within a set of images. Such distribution of attribute labels is maintained across all the 4 closets, but each having a set of different style images than the other. The idea is to show that variation in the users' preferences for the same style generates dissimilar results. Figure 5.5 presents an example of personalizing a dress using style images from four different closets.

We train the classifier over the total 400 images to predict the closet id. The same images are used to extract styles for the style transfer process. We generate 140 new images

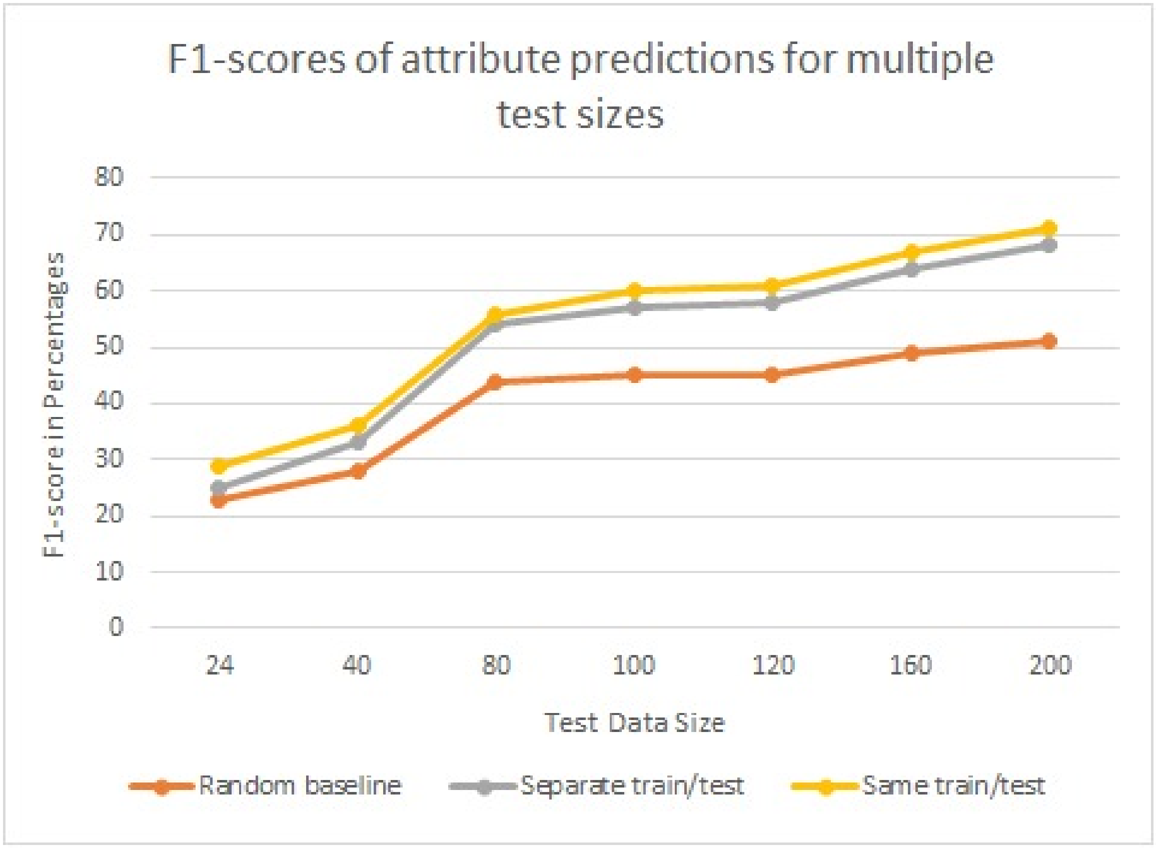


FIG. 5.4: Bar-chart showing F1-scores for the baseline and our model on actual test data using separate training and test generation images, and using same images for training and test data generation

- 35 for each closet, and give it to the classifier to get the closet id it would most closely belong to. The distribution of attribute labels across closets is uniform and the attributes are almost the same. Figure 5.6 graphically summarizes the process.

Since there are one hundred images for each closet, the accuracy by random selection would be 25%. The accuracy we get on test data is around 23% which is, in fact, worse than what we get using random selection. This may be because the distributions of labels in every closet is similar and that makes it difficult to show one closet different from the other. Our likely intuition is that now the classification problem narrows down to being

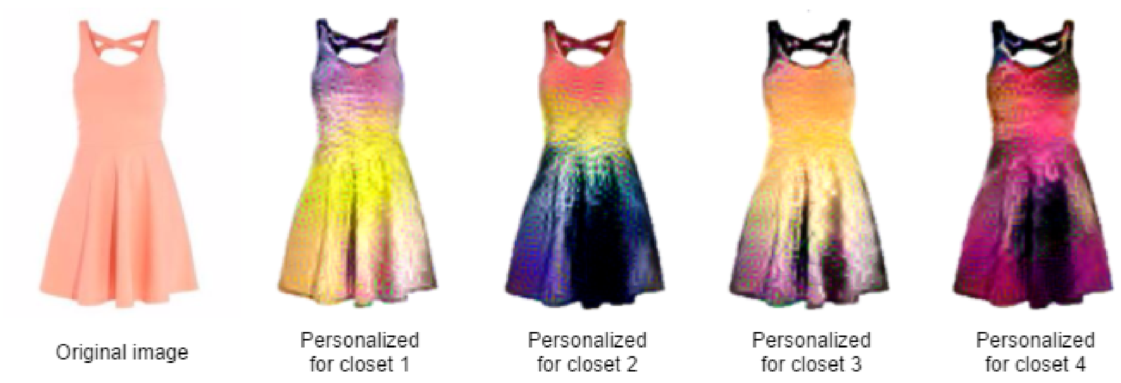

Original image
Personalized for closet 1
Personalized for closet 2
Personalized for closet 3
Personalized for closet 4

FIG. 5.5: Dress personalized for multiple closets

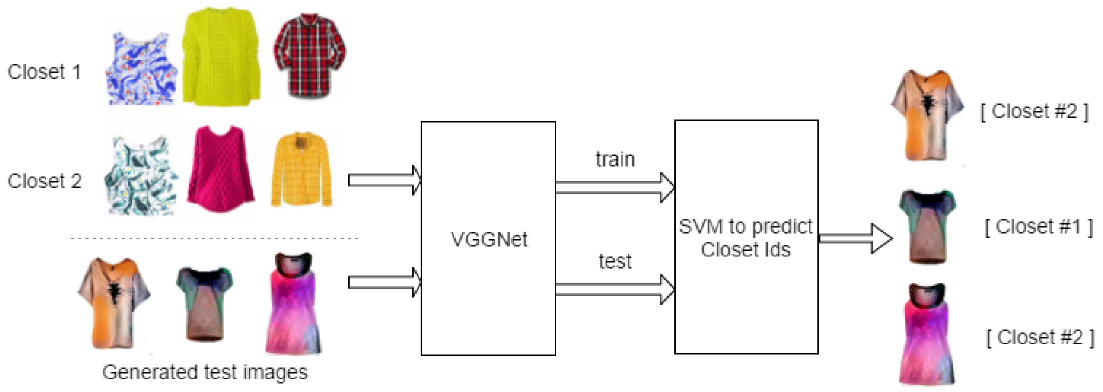


FIG. 5.6: Evaluation model for predicting closet ids

able to distinguish between similar patterns across different closets rather than among different styles. The number of style images also have a significant impact on the style of the generated clothing item as explained in Section 5.2.2. A convoluted design makes it harder to identify similarities in patterns of the images belonging to a particular closet and the new generated one. We are investigating the causes of closet misprediction further.

5.2 Qualitative Evaluation

We evaluate the personalization of clothing qualitatively by analyzing how similar the result image looks to the style images used in the style transfer process. The quality of the generated image is impacted by a number of factors. We present how each of them affect the style transfer process. In general, the idea is to pick styles from multiple images to transform a given image to suit the user's liking. The Figure 5.7 shows an image of a sheer draped blouse changed to adopt the styles extracted from a couple of images. The result is a nice blend of patterns borrowed from the style images given.

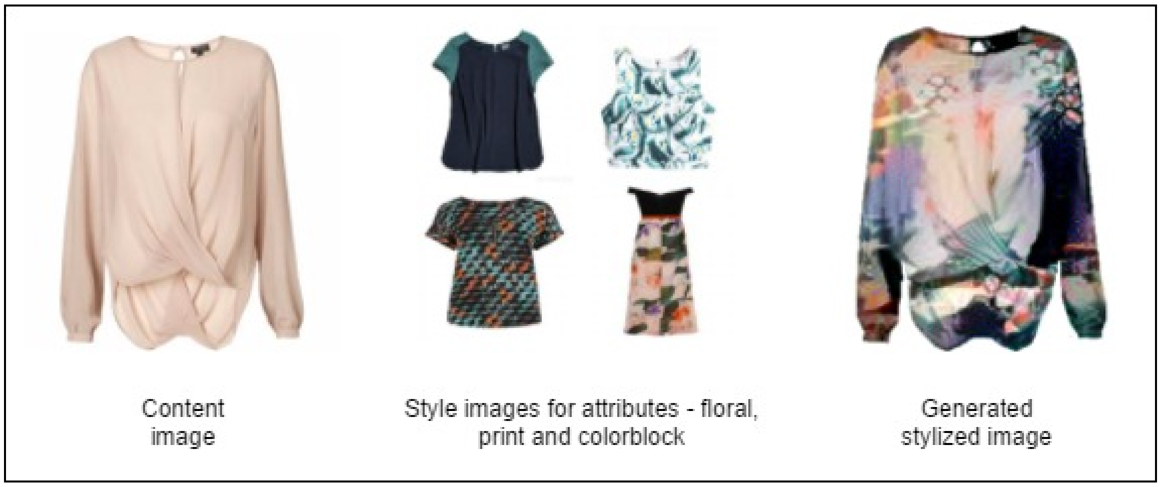


FIG. 5.7: Multiple styles reinforced in a content image

A single style superimposed on the same content image, but using multiple distinct style images produces interesting results. Figure 5.8 presents the style of four different "knit" garments over a tank top. The new image generated is beautifully in sync with the style images in terms of texture. Four different textures of the same fabric produce such distinct results. This justifies the fact that one user's choice of clothing is likely to be different from another user's liking for the same style.

5.2.1 Iterations

A default of 1000 iterations of the LBFGS optimizer gave good results. For every iteration, the style and content losses are minimized to reach the ideal image. Figure 5.9

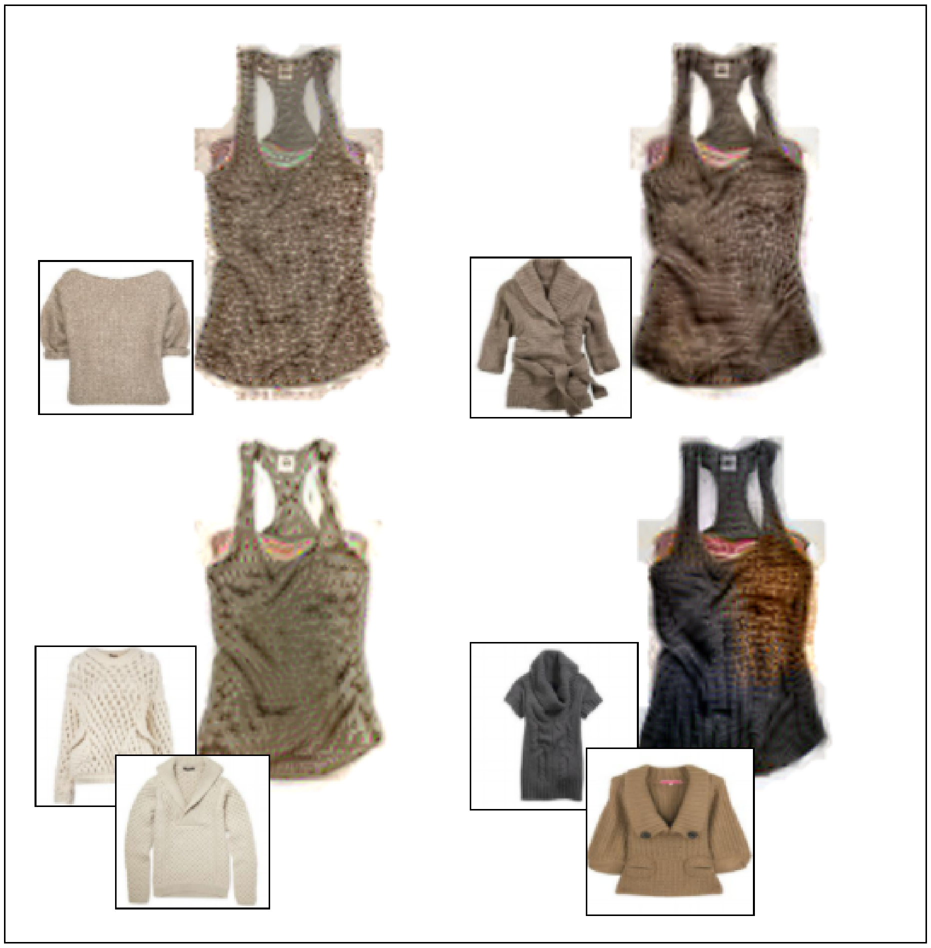


FIG. 5.8: Styles extracted from multiple images for the same attribute ”knit”

demonstrates the transformation the initial images go through at every iteration. The first column shows the content and style images followed by the results of intermediate iterations. The final output is the last image in the series. We can see that the patterns and textures transferred get sharper and clearer in the subsequent iterations. The new image maintains the shape of the content provided but is strikingly similar to the image providing it style.

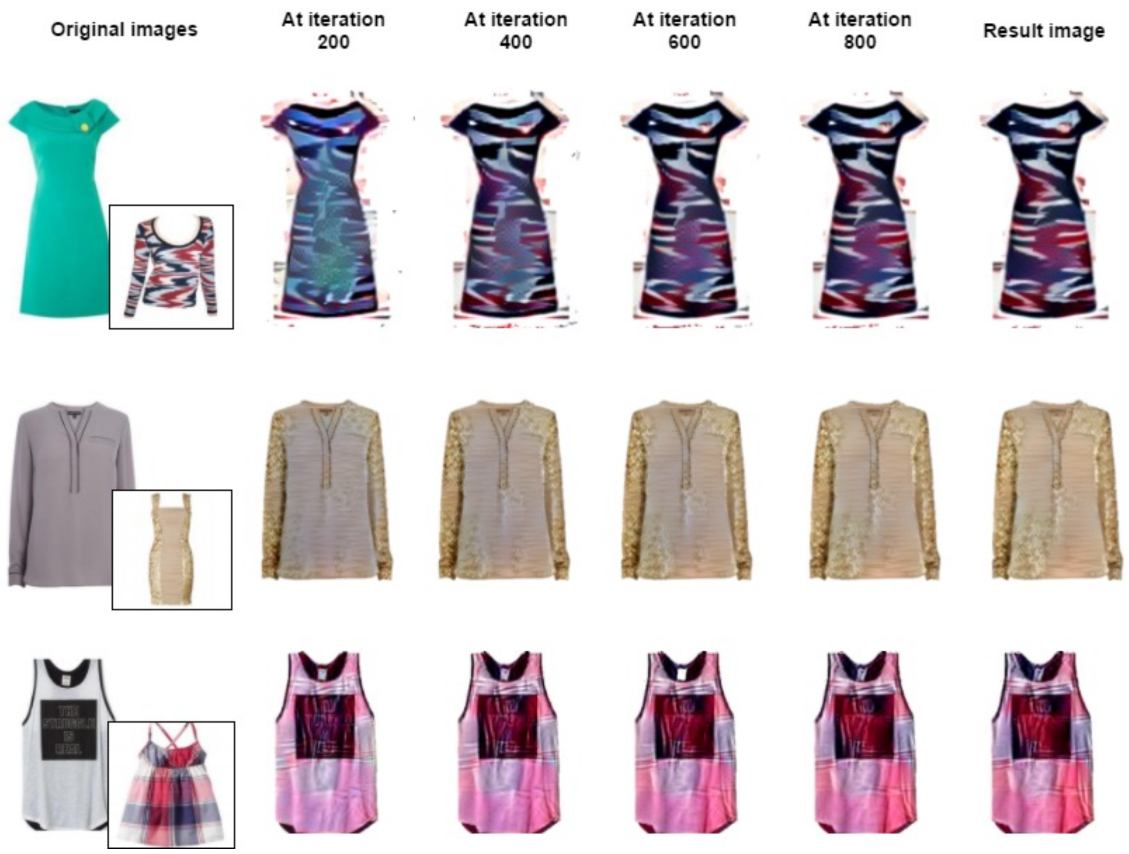


FIG. 5.9: Image output for intermediate iterations of the style transfer process

5.2.2 Number of style images

We normalize the weights across all images used in the style transfer process. Thus, the number of style images have a significant impact on the quality of the generated image. Fewer style images leads to better transfer of style. Too many images create a hodge-podge of styles and the detailing of patterns or textures is lost. The result is an image with patchy prints all over, or more so, a generic blend of colors. Figure 5.10 below shows a mix of multiple images of the Checkered pattern.

The results of transferring a floral pattern with two images and four images is shown in Figure 5.11.

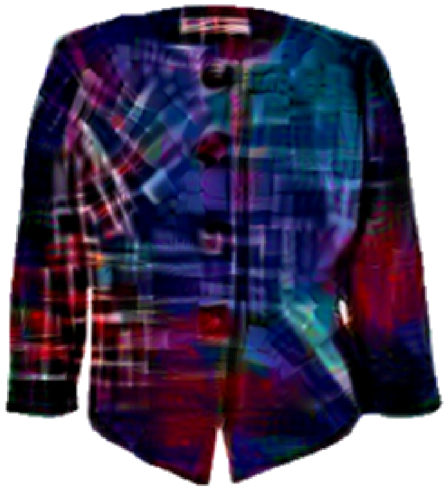


FIG. 5.10: A new generated image for multiple checkered patterns. The blend is not seamless and shows a lot of confusion of styles.

5.2.3 Combinations of multiple styles

As mentioned previously, combining multiple styles may or may not give decent style representation in the resultant image. Some styles combined may work well if the patterns inherently fuse well with each other. For example, as shown in Figure 5.12, tile print with colorblock results into a visually appealing mix of style as opposed to combining floral with the checkered pattern.

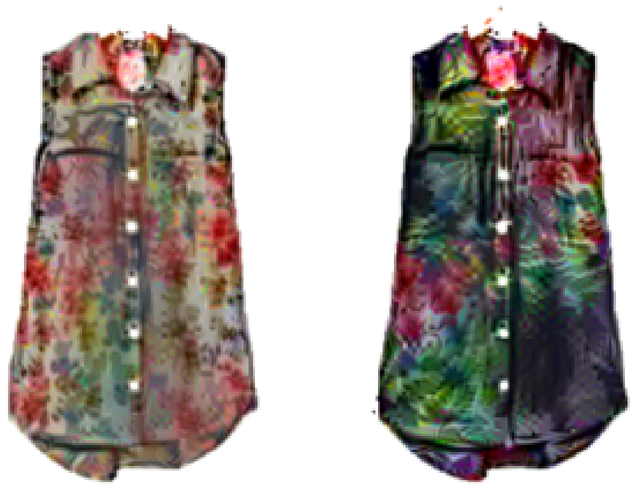


FIG. 5.11: Result for floral pattern imposed using 2 style images(Left) and 4 images(Right)

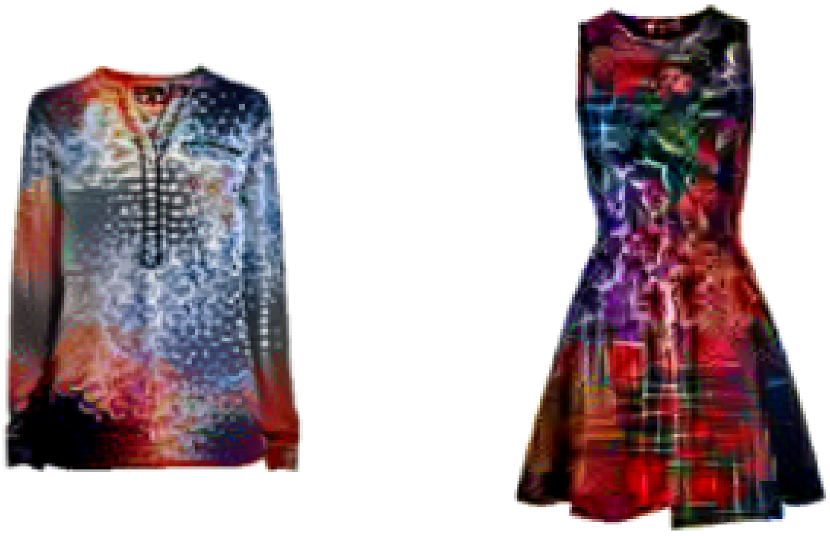


FIG. 5.12: (Left) Styles forming a valid combination. (Right) Styles that inherently don't go well with each other.

Chapter 6

CONCLUSION AND FUTURE WORK

In this thesis, we present an initial pipeline to generate new designs for clothes based on the style preference of the user. We demonstrate that the idea of using neural style transfer of personalizing clothing is legitimate and gives valid results. Although evaluation of recommendation systems is a difficult problem, we attempt to analyze (qualitatively and quantitatively) how close the generated designs are to the user's fashion choices. We show how the number of style images, the combination of styles, size of style and content images, etc. influence the quality of the result image.

There might be better approaches for evaluation that can be worked upon in the future. Taking a step back, it would in fact be very interesting to tweak the style transfer process to generate better results. Allowing the user to control the placement of styles on the new clothing image would be a valuable addition. Currently, we personalize in terms of patterns and textures alone. In the future, combining (Isola *et al.* 2016)'s work would enable transfer of shape as well.

We also see the need to incorporate real world images for style transfer. The set of style images could be photographs taken in a natural setting, from an online shopping website, with multiple objects in the background or with obscured segments of clothing. Extracting styles from such images would be a real challenge and can be an area of research.

REFERENCES

[1] Bossard, L.; Dantone, M.; Leistner, C.; Wengert, C.; Quack, T.; and Van Gool, L. 2012. Apparel classification with style. In *Asian conference on computer vision*, 321–335. Springer.

[2] Chen, L.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR* abs/1606.00915.

[3] Efros, A. A., and Freeman, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 341–346. ACM.

[4] Efros, A. A., and Leung, T. K. 1999. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, 1033–1038. IEEE.

[5] Gatys, L.; Ecker, A. S.; and Bethge, M. 2015a. Texture synthesis using convolutional neural networks. In *Advances in Neural Information Processing Systems*, 262–270.

[6] Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2015b. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*.

[7] Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2414–2423.

[8] Hadi Kiapour, M.; Han, X.; Lazebnik, S.; Berg, A. C.; and Berg, T. L. 2015. Where

to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE International Conference on Computer Vision*, 3343–3351.

[9] Heeger, D. J., and Bergen, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 229–238. ACM.

[10] Isola, P.; Zhu, J.-Y.; Zhou, T.; and Efros, A. A. 2016. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*.

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

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

[13] Kalantidis, Y.; Kennedy, L.; and Li, L.-J. 2013. Getting the look: clothing recognition and segmentation for automatic product suggestions in everyday photos. In *Proceedings of the 3rd ACM conference on International conference on multimedia retrieval*, 105–112. ACM.

[14] Kiapour, M. H.; Yamaguchi, K.; Berg, A. C.; and Berg, T. L. 2014. Hipster wars: Discovering elements of fashion styles. In *European conference on computer vision*, 472–488. Springer.

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

[16] Kwatra, V.; Schödl, A.; Essa, I.; Turk, G.; and Bobick, A. 2003. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, volume 22, 277–286. ACM.

[17] Linden, G.; Smith, B.; and York, J. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7(1):76–80.

[18] Loni, B.; Cheung, L. Y.; Riegler, M.; Bozzon, A.; Gottlieb, L.; and Larson, M. 2014. Fashion 10000: an enriched social image dataset for fashion and clothing. In *Proceedings of the 5th ACM Multimedia Systems Conference*, 41–46. ACM.

[19] Portilla, J., and Simoncelli, E. P. 2000. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision* 40(1):49–70.

[20] Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR* abs/1506.01497.

[21] Rother, C.; Kolmogorov, V.; and Blake, A. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, 309–314. ACM.

[22] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR* abs/1409.1556.

[23] Smith, C. 2016. neural-style-tf. `https://github.com/cysmith/neural-style-tf`.

[24] Trewin, S. 2000. Knowledge-based recommender systems. *Encyclopedia of library and information science* 69(Supplement 32):180.

[25] Wei, L.-Y., and Levoy, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 479–488. ACM Press/Addison-Wesley Publishing Co.

[26] Xiao, T.; Xia, T.; Yang, Y.; Huang, C.; and Wang, X. 2015. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2691–2699.

[27] Zeiler, M. D., and Fergus, R. 2013. Visualizing and understanding convolutional networks. *CoRR* abs/1311.2901.

[28] Ziwei Liu, Ping Luo, S. Q. X. W., and Tang, X. 2016. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.