Dictionary Learning for Image Style Transfer

Dictionary Learning for Image Style Transfer

Hyeon-Jae Seo

Supervised by Prof. Demba Ba

Presented to the Department of Applied Mathematics
in partial fulfillment of the requirements
for a Bachelor of Arts degree with Honors

Harvard University
Cambridge, Massachusetts

April 3, 2020

Abstract

Deep image representations learned by CNNs have been shown to successfully separate image content from style, but the generic feature representations learned by CNNs are far from interpretable. Drawing upon recent developments in neural network inspired extensions of Dictionary Learning (DL) to multi-layer models, I present a framework using dictionary learning and sparse code representations to separate image content and style in natural images. This framework provides an interpretable decomposition of images into high and low frequency representations that can be used to generate new images by combining the style and content of any two arbitrary images. These decompositions and visualizations of convolutional filters provide novel insights into the image representations learned by convolutional neural networks and demonstrate the potential of DL and sparse code representations for image analysis and transformation.

Contents

1 Introduction

Advancements in deep learning and convolutional neural networks (CNNs) present a significant paradigm shift in the landscape of automated image style transfer, in which the semantic content of an image is rendered in a different style. Deep CNNs have demonstrated the capability to perform neural style transfer (NST) between arbitrary images [7], which entails training a deep neural network to generate artistic images by combining the content and style of arbitrary images. Thus, deep learning provides the potential for broad-ranging and flexible style transfers, but CNN-based approaches are computationally expensive due to the heavy optimization involved in obtaining synthesized images close to the content image. Furthermore, the challenge of interpreting the style representations learned by CNNs persists.

In this work, I present a novel framework to address both of these issues that draws upon the theory of wavelet analysis and dictionary learning. Representation analysis with wavelets is a classical, interpretable, and well understood theory that allows us to decompose images/signals into a linear combination of basis functions at different scales to represent an image and recover it perfectly via convolution operations [13, 12], while sparse coding (SC) [1] finds sparse representations of images, or signals more broadly. Thus, wavelet analysis and sparse coding provide a way to decompose images into distinct style and content representations, while dictionary learning provides a framework for learning the shared structure of images. I call this framework a hierarchical generative model for image analysis and show that this model learns visually interpretable convolutional filters and highly efficient representations of images that can very faithfully reconstruct even complex colored images.

The challenges regarding the computational complexity and limited interpretability of CNNs are of great importance due to the widespread use of CNNs. Reducing the dependence on exceptionally large computational resources would diminish the financial and environmental impact of training and developing CNN based models [17], while obtaining more efficient and interpretable representations from CNNs would help ease the adoption of such models in new application areas. While in this work we focus on a specific use case in the artistic domain, a more detailed and nuanced understandings of the role of convolutions in image style transfer would have great significance in the fields of computer graphics, vision, and image processing more broadly.

1.1 Related Work

The diverse literature on image style transfer has its roots in the simpler task of texture synthesis, in which the goal is to synthesize a new texture that appears to have been generated via the same process as a given texture sample. Several techniques, including physical simulation of their generation process [3, 20, 21], Markov random fields with Gibbs sampling [5, 14, 15], and feature matching [2, 8, 16], have all been successfully applied towards this task.

These developments were followed by demonstrations of texture transfer via techniques like image quilting [10], image analogies [9], and extensions of texture synthesis algorithms to account for content similarities [6]. On the machine learning front, several attempts have been made to develop a feedforward style transfer network that is trained to go from a content image to a stylized image in one pass [19, 11]. Yet a significant drawback of these style transfer networks is that they are tied to the single style on which they are trained, which means separate networks need to be trained for each style modeled. Dumoulin et al. (2017) tackled this issue by introducing conditional instance normalization, in which a single conditional style transfer network is trained on multiple styles, and while each specific style uses specialized parameters, the convolutional weights are share across all styles. This extension allows style transfer networks to learn multiple styles while still maintaining both the qualitative and convergence properties of the single-style transfer networks.

1.2 Problem Specification

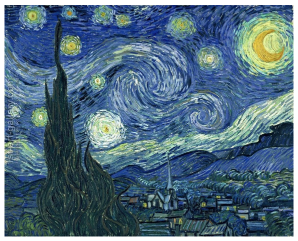
Given two datasets $\mathcal{D}_s$ and $\mathcal{D}_t$ of images in styles s and t, respectively, our question is twofold. First, can we separately capture style or texture and content information in images with representation analysis? I show that the latent representations learned by the hierarchical generative model presented here does indeed capture image specific texture and content details, while the convolutional filters capture common structural information that is shared across images in a given dataset. Here, the term "image texture" refers to the spatial arrangement of colors or intensities in a given image. This is the first primary contribution of this project – while previous work in this area has looked at the role of shared filters in convolutional neural networks for style transfer, to my knowledge this is the first attempt to obtain interpretable decompositions of an image into style and content representations.

Second, can we synthesize new images by arbitrarily combining the content of one image with the style of another? Although measures of style are not quantitatively defined, we can assess the extent to which the style and content aspects of the source images are transferred to the target image through visual inspection. The second major contribution of this work is a demonstration of arbitrary style transfer using the set of filters learned by the hierarchical generative model with no additional training.

1.2.1 Motivating Example

To illustrate the task at hand, consider the following setting. I have a photographic portrait of a woman, as in Fig 1(a), and a copy of Vincent Van Gogh's *Starry Night*, as in Fig 1(b). I would like to synthesize a new image featuring the woman in the portrait but in the style of Van Gogh's painting. One possible rendition of such a transfer can be seen in Fig 1, which was generated by [4] using a deep neural network. My goal is to attempt a similar task but with the proposed hierarchical generative model instead of a neural network, while maintaining generality so that we can conduct such a style transfer between any two arbitrary images of the selected styles.

(a) Photographic portrait of a woman (from Dumoulin et al. 2017)


(b) Vincent Van Gogh's *Starry Night*

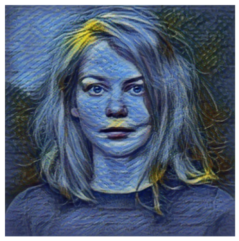

(c) Newly generated image combining content from (a) with the style from (b) (also from Dumoulin et al. 2017)

Figure 1: Motivating example of style transfer

1.3 Overview of Thesis

The remainder of this thesis is organized as follows. Section 2 provides the background on the technical concepts that are drawn upon in this project and section 3 specifies the model used. Section 4 details the simulations used to develop intuition for the application of the hierarchical generative model for image analysis, while section 5 presents the results of experiments conducted on natural images. Section 5 concludes.

2 Background

2.1 A Generative Model of Images

Given a dense signal $\mathbf{x}_L$ and the sparse signals $\mathbf{u} = [\mathbf{u}_1, \dots, \mathbf{u}_L]$, I can define the following recursive generative model proposed by [22]

$$\mathbf{x}_{\ell-1} = \mathbf{A}_\ell * \mathbf{x}_\ell + \mathbf{B}_\ell * \mathbf{u}_\ell + \epsilon_\ell, \quad \ell \in \{1, \dots, L\} \tag{1}$$

Here ℓ indicates the layer the signals correspond to, where $\ell = 0$ represents the input image and $\ell > 0$ denotes deeper layers, with a total depth of L. Throughout this manuscript, I will interchangeably refer to $\mathbf{x}_\ell$ and $\mathbf{u}_\ell$ as *signals*, *encodings*, and *latent representations*. $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ represent *texture* and *content* filters, respectively. Filters are tensors of dimension $C_\ell \times D_\ell \times H_\ell \times W_\ell$, denoting the number of output channels, number of input channels, filter height, and filter width, respectively. The $*$ operation denotes a full convolution between a filter and a signal, which is more explicitly defined by the following

$$(\mathbf{A}_\ell * \mathbf{x}_\ell)(c) = \sum_{d=1}^{D_\ell} \mathbf{A}_\ell(c, d) * \mathbf{x}_\ell(d), \forall c \in 1, \dots, C_\ell \tag{2}$$

where c indexes the output channels and d indexes the input channels. Throughout this manuscript, the convolution between $\mathbf{A}$ and $\mathbf{x}$, here denoted as $\mathbf{A} * \mathbf{x}$, may be shortened to just $\mathbf{Ax}$ when it is clear that this operation denotes a convolution.

Furthermore, we assume the following latent prior distributions:

$$\mathbf{u}_\ell \sim \mathit{Laplace}(0, \lambda_\ell) \tag{3}$$
$$\mathbf{x}_\ell \sim \mathcal{N}(0, \sigma^2_{x_\ell}) \tag{4}$$

The Laplacian prior on $\mathbf{u}_\ell$ supports its sparsity and facilitates the capture of high frequency information in $\mathbf{u}_\ell$, while the Gaussian prior on $\mathbf{x}_\ell$ supports a non-sparse form and facilitates the encoding of low frequency information in $\mathbf{x}_\ell$. As textural information primarily takes the form of color and patterns, it presents a source of low frequency information, whereas content information primarily entails edge detection, and therefore presents high frequency information. Thus, $\mathbf{A}_\ell$ and $\mathbf{x}_\ell$ are intended to capture texture information, while $\mathbf{B}_\ell$ and $\mathbf{u}_\ell$ are intended to capture content information.

Figure 2 depicts a graphical representation of the generative model defined by Eq. 1 for three layers. As the sparse signals $\mathbf{u}_\ell$ are independent across layers, the sparsity requirement does not impose restrictions on the depth of the model and any depth is attainable.

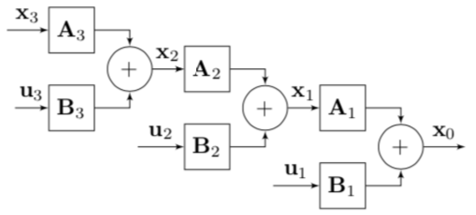


Figure 2: Hierarchical model representation for $L = 3$

In the following sections, I further specify the assumptions of this model and methods for finding the encodings $\mathbf{x}_\ell$ and $\mathbf{u}_\ell$.

2.2 Convolutional Sparse Coding

While I can synthesize images given the signals $\mathbf{x}_L$ and $\mathbf{u} = [\mathbf{u}_1, \ldots, \mathbf{u}_L]$ following the generative model of Eq 1, recovering these encodings from an input image $\mathbf{x}_0$ requires solving the inverse problem. This problem has its roots in the theory of sparse coding, which refers to a class of algorithms for factoring input data into a linear combination of over-complete basis vectors, $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_K]$. A commonly used cost function for this task is

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{u}} \ \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{y}_n - \mathbf{B}\mathbf{u}_n\|_2^2 + \lambda\|\mathbf{u}_n\|_1 \\ &\text{subject to } \|\mathbf{b}_k\|_2^2 \leq 1 \text{ for } k = 1, \ldots K \end{aligned} \tag{5}$$

where λ controls the L_1 regularizer and the constraints on $\mathbf{b}_k$ ensure that the dictionary $\mathbf{B}$ doesn't absorb all magnitude-related information. This formulation is also commonly known as LASSO [18] and its usefulness has been demonstrated in a variety of machine learning contexts.

One advantage of over-complete bases over complete bases is that the former are more conducive to capturing structures and patterns in the data. However, traditional sparse coding assumes that the input vectors $\{\mathbf{y}_n\}_{n=1}^N$ are independent, which leads to basis vectors that are translations of each other for natural images. Convolutional sparse coding addresses this problem by incorporating shift invariance into the objective function:

$$\begin{aligned} &\min_{\mathbf{B},\mathbf{u}} \ \frac{1}{2}\|\mathbf{y} - \sum_{k=1}^{K} \mathbf{b}_k * \mathbf{u}_k\|_2^2 + \lambda\sum_{k=1}^{K} \|\mathbf{u}_k\|_1 \\ &\text{subject to } \|\mathbf{b}_k\|_2^2 \leq 1 \text{ for } k = 1, \ldots K \end{aligned} \tag{6}$$

Now, when $\mathbf{b}_k * \mathbf{u}_k$ is summed over all k, the result should approximate the full input signal $\mathbf{y}$, rather than independent sections, as would have resulted from the objective in Eq. 5.

2.3 Hierarchical CSC for Image Encodings

To recover the dense signals $\mathbf{x}_\ell$ across layers in addition to the sparse codes, I need to solve a derivation of the sparse coding coding problem developed by [22] and termed hierarchical convolutional sparse coding (H-CSC). In this setting, an input image $\mathbf{x}_0$ is decomposed into both dense *scale* and sparse *detail* signal components by solving the following problem

$$\hat{\mathbf{x}}_\ell, \hat{\mathbf{u}}_\ell = \arg\min_{\mathbf{x}_\ell, \mathbf{u}_\ell} f(\hat{\mathbf{x}}_{\ell-1}, \mathbf{x}_\ell, \mathbf{u}_\ell) + \lambda_\ell\|\mathbf{u}_\ell\|_1 + \gamma_\ell\|\mathbf{x}_\ell\|_2^2 \tag{7}$$

where

$$f(\hat{\mathbf{x}}_{\ell-1}, \mathbf{x}_\ell, \mathbf{u}_\ell) = \frac{1}{2}\|\hat{\mathbf{x}}_{\ell-1} - \mathbf{A}_\ell * \mathbf{x}_\ell - \mathbf{B}_\ell * \mathbf{u}_\ell\|_2^2 \tag{8}$$

for every $\ell \in \{1, \ldots, L\}$. Here, $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ is the given input image and subsequent estimates $\hat{\mathbf{x}}_\ell$ are obtained by solving Eq. 7. The ℓ_1-norm enforces sparsity on $\mathbf{u}_\ell$ with the parameter λ controlling the level of sparsity.

Note that by setting $\mathbf{A}_\ell = 0$, we return to the traditional sparse coding setting of the previous section.

2.4 Proximal Algorithms

Because Eq. 7 is non-smooth and convex, it can be solved with a proximal gradient method, a common framework for solving nonsmooth, constrained optimization problems of the form

$$\min_x f(x) + g(x) \tag{9}$$

where both f and g are proper closed convex and f is differentiable. The non-smooth nature of g is handled via a proximal operator, defined as

$$prox_g(y) = \arg\min_x \left(g(x) + \frac{1}{2}\|x - y\|_2^2 \right) \tag{10}$$

where $\| \cdot \|_2$ is the Euclidean norm. This function is strongly convex and not everywhere infinite, so it has a unique solution $y \in \mathbb{R}^N$. For a scaled function λg, where $\lambda > 0$, the corresponding proximal operator is

$$prox_{\lambda g}(y) = \arg\min_x \left(g(x) + \frac{1}{2\lambda}\|x - y\|_2^2 \right) \tag{11}$$

The proximal operator can be considered to be like a gradient step for g, as we have that

$$prox_{\lambda g}(y) \approx y - \lambda \Delta g(y) \tag{12}$$

Furthermore, $prox_{\lambda g}(x^*) = x^*$ if and only if x^* minimizes g. I can therefore minimize g by finding a fixed point of $prox_g$. In particular, due to the firm nonexpansiveness property of $prox_g$, the iterative proximal method given by

$$x^k := prox_{\lambda g}(x^{k-1}) \tag{13}$$

converges to the set of minimizers of g, provided that g has a minimum. Here k denotes an iteration counter and x^k is the kth iteration of the algorithm. Returning to the motivating minimization problem, the basic proximal gradient method to solve Eq. 9 is

$$x^k := prox_{\lambda g}\left(x^{k-1} - \lambda \Delta f(x^{k-1})\right) \tag{14}$$

By replacing the λ with λ^k, I can also handle parameter values that change in each iteration. Convergence is still guaranteed as long as $\lambda^k > 0$ and $\sum_{k=1}^{\infty} \lambda^k = \infty$.

2.4.1 FISTA

Adopting the idea behind the basic proximal gradient method presented in Eq. 14 to a nonsmooth regularized problem of the form

$$\min_x f(x) + \lambda \|x\|_1 \tag{15}$$

the gradient iteration can be written as

$$x^k = \arg\min_x \left\{ f(x^{k-1}) + \langle x - x^{k-1}, \Delta f(x^{k-1}) \rangle + \frac{1}{2t^k}\|x - x^{k-1}\|^2 + \lambda \|x\|_1 \right\} \tag{16}$$

As I can drop constant terms and separate the l_1 norm, this minimization reduces to solving

$$x^k = \mathcal{T}_{\lambda t^k}(x^{k-1} - t^k \Delta f(x^{k-1})) \tag{17}$$

where $\mathcal{T}_\alpha : \mathbb{R}^n \to \mathbb{R}^n$ is the shrinkage operator given by

$$\mathcal{T}_\alpha(x) = (|x| - \alpha)_+ sgn(x) \tag{18}$$

The iterative scheme of Eq. 17 forms the basis of a popular proximal gradient method known as ISTA, or *Iterative Shrinkage-Thresholding Algorithm.* This method has been extended to an accelerated version known as FISTA, or the *Fast Iterative Shrinkage-Thresholding Algorithm.* This procedure is the one we use and is outlined in Algorithm 1.

Algorithm 1 FISTA for H-CSC

```
1: Input: $\mathbf{x}_\ell, \alpha, \lambda_\ell, \gamma_\ell$
2: Initialize: $x^1_{\ell+1}, \mathbf{u}^1_{\ell+1}, t^1 = 1$
3: for $k \in \{1, \dots, K\}$ do
4:     $t^{k+1} \leftarrow 1 + \sqrt{1 + 4(t^k)^2}/2$
5:     $\bar{\mathbf{u}} \leftarrow \mathbf{u}^k_\ell + (t^k - 1)(\mathbf{u}^k_\ell - \mathbf{u}^{k-1}_\ell)/t^{k+1}$
6:     $\bar{\mathbf{x}} \leftarrow \mathbf{x}^k_\ell + (t^k - 1)(\mathbf{x}^k_\ell - \mathbf{x}^{k-1}_\ell)/t^{k+1}$
7:     $\mathbf{r} \leftarrow \mathbf{A}_\ell * \bar{\mathbf{x}} + \mathbf{B}_\ell * \bar{\mathbf{u}} - \mathbf{x}_\ell$
8:     $\mathbf{x}^{k+1}_\ell \leftarrow \bar{\mathbf{x}} - \alpha(\mathbf{r} \star \mathbf{A}_\ell + \gamma_\ell \bar{\mathbf{x}})$
9:     $\mathbf{u}^{k+1}_\ell \leftarrow \mathcal{S}_{\lambda_\ell \alpha}(\bar{\mathbf{u}} - \alpha(\mathbf{r} \star \mathbf{B}_\ell))$
10: end for
11: return $\mathbf{x}^K_\ell, \mathbf{u}^K_\ell$
```

2.5 Convolutional Dictionary Learning

I have now described how to recover the variables $\mathbf{x}_\ell$ and $\mathbf{u}_\ell$ across layers assuming the filters $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ are fixed, so it remains to expound on the method for learning the filters $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$. To update the filters, we fix $\mathbf{x}_\ell$ and $\mathbf{u}_\ell$ and minimize the following loss function:

$$\min_{\mathbf{A}_\ell, \mathbf{B}_\ell} \sum_{i=1}^{N} \|\hat{\mathbf{x}}_{\ell-1,i} - \mathbf{A}_\ell * \hat{\mathbf{x}}_{\ell,i} - \mathbf{B}_\ell * \hat{\mathbf{u}}_{\ell,i}\|_2^2, \quad \text{s.t. } \mathbf{A}_\ell \star \mathbf{A}_\ell = 1, \mathbf{B}_\ell \star \mathbf{B}_\ell = 1 \tag{19}$$

where $\hat{\mathbf{x}}_{\ell,i}, \hat{\mathbf{u}}_{\ell,i}$ represents the estimates of the encodings at depth ℓ for the ith image in a dataset indexed by $\{1, \dots, N\}$.

I can then update the filters by directly computing the first order gradient of Eq. 19. Since the filters are four dimensional tensors ($C_\ell \times D_\ell \times H_\ell \times W_\ell$) whereas the images and intermediate signals are three dimensional ($D_\ell \times H_\ell \times W_\ell$), we extend the dimension of the images by one and obtain the following gradients

$$\begin{aligned} \frac{\partial}{\partial \mathbf{A}_\ell} \|\hat{\mathbf{x}}_{\ell-1,i} - \mathbf{A}_\ell * \hat{\mathbf{x}}_{\ell,i} - \mathbf{B}_\ell * \hat{\mathbf{u}}_{\ell,i}\|_2^2 &= (\mathbf{A}_\ell * \hat{\mathbf{x}}_{\ell,i} + \mathbf{B}_\ell * \mathbf{u}_{\ell,i} - \hat{\mathbf{x}}_{\ell-1,i}) * \hat{\mathbf{x}}_{\ell,i} \\ \frac{\partial}{\partial \mathbf{B}_\ell} \|\hat{\mathbf{x}}_{\ell-1,i} - \mathbf{A}_\ell * \hat{\mathbf{x}}_{\ell,i} - \mathbf{B}_\ell * \hat{\mathbf{u}}_{\ell,i}\|_2^2 &= (\mathbf{A}_\ell * \hat{\mathbf{x}}_{\ell,i} + \mathbf{B}_\ell * \hat{\mathbf{u}}_{\ell,i} - \hat{\mathbf{x}}_{\ell-1,i}) * \mathbf{u}_{\ell,i} \end{aligned} \tag{20}$$

which are then used to update the filters $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ appropriately.

Given these methods for recovering both $\hat{\mathbf{x}}_\ell$ and $\hat{\mathbf{u}}_\ell$ assuming $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ are fixed, as well as for identifying $\mathbf{A}_\ell$ and $\mathbf{B}_\ell$ assuming $\hat{\mathbf{x}}_\ell$ and $\hat{\mathbf{u}}_\ell$ are fixed, I can solve Eq. 1 by utilizing an optimization procedure that alternates between these two methods. The following section specifies the framework used to combine these methods.

3 Model

3.1 Notation

Notations and conventions used throughout this manuscript are summarized in Table 1.

Symbol	Description
$\mathbf{A}$	Matrix (upper-case bold letter)
$\hat{\mathbf{A}}$	Matrix learned by a model
$\mathbf{x}$	Vector (lower-case bold letter)
$\hat{\mathbf{x}}$	Estimation of vector $\mathbf{x}$
$\mathbf{x}_{\ell,i}$	Vector $\mathbf{x}$ corresponding to the ℓ^{th} layer of the i^{th} data sample
$\mathbf{x}_i$	Vector $\mathbf{x}$ corresponding to the i^{th} data sample when the # of layers $L = 1$
$\mathbf{x}^k$	Vector $\mathbf{x}$ at the k^{th} iteration of an iterative algorithm
$\lVert\mathbf{x}\rVert_p$	The ℓ_p-norm of vector $\mathbf{x}$
$*$	Linear convolution
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2

Table 1: Notation and conventions

3.2 Implementation

To incorporate the methods described in the previous section into a cohesive method, I utilize an autoencoder framework as follows:

1. The filters $\hat{\mathbf{A}}_\ell$ and $\hat{\mathbf{B}}_\ell$ are initialized with values drawn from $\mathcal{N}(0, 1)$.
2. A batch of images, $\mathcal{I}$, is drawn from the chosen dataset, $\mathcal{D}$, and is fed into the network.
3. In the first half of the forward pass, the model uses FISTA to estimate the underlying signals $\hat{\mathbf{x}}_{\ell,i}$ and $\hat{\mathbf{u}}_{\ell,i}$ for each image $\mathbf{y}_i \in \mathcal{I}$ and each layer $\ell \in \{1, \ldots, L\}$. This is commonly known as the *encoding* step.
4. In the second half of the forward pass, the model reconstructs estimates of the input images, $\hat{\mathbf{y}}_i$, from the encodings $\hat{\mathbf{x}}_{\ell,i}$ and $\hat{\mathbf{u}}_{\ell,i}$ identified in the previous step following the recursive formula in Eq. 1. This is commonly known as the *decoding* step.
5. In the backward pass, the loss function in Eq. 19 is used to measure the discrepancy between the original images, $\mathbf{y}_i$, and the reconstructed images, $\hat{\mathbf{y}}_i$, which is then used to update $\hat{\mathbf{A}}_\ell$ and $\hat{\mathbf{B}}_\ell$ accordingly.
6. Steps 3 through 5 are repeated for the specified number of epochs, drawing a different batch from the image dataset each time.

Note that the filters $\hat{\mathbf{A}}_\ell$ and $\hat{\mathbf{B}}_\ell$ are shared across all images in the dataset, whereas the encodings $\hat{\mathbf{x}}_{\ell,i}$ and $\hat{\mathbf{u}}_{\ell,i}$ are image specific.

The entire framework is created using PyTorch and relies heavily on the torchvision package for manipulating images.

4 Simulations

To visually illustrate the hierarchical generative model framework, I generate some sample images and provide intuition for the theory of style transfer in this context.

4.1 Image Data Generation

I focus on the 1 layer case with grayscale images (i.e. one input channel and one output channel), and thus to simplify notation, I drop the subscripts on the filters and encodings that denote the layer of each component. This means that the generation process for an image $\mathbf{y}_i$ is defined as follows:

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i \tag{21}$$

where $\mathbf{A}, \mathbf{B}$ are the shared texture and content filters, respectively, and $\mathbf{x}_i, \mathbf{u}_i$ are the image-specific texture and content encodings, respectively. $\mathbf{y}_i$ denotes the final image generated from a linear combination of these components. Note that in this generation process, I am initializing the latent representations to synthesize an image, whereas later I will be solving the reverse problem, i.e. decomposing a given image into its latent representations.

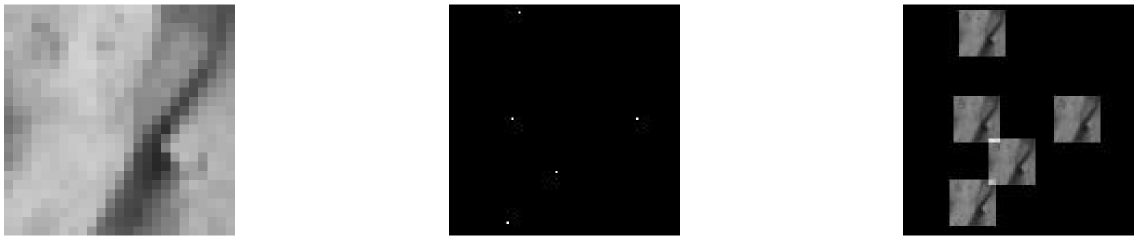
To obtain $\mathbf{A}$, I start with an arbitrarily selected colored texture image T (see Figure 3). I crop T to the desired dimensions, which is 25×25 pixels in this case. I then convert the cropped image to grayscale, recast it as a tensor, and normalize the values in the tensor.

(a) Marble (b) Sand

Figure 3: Sample close-up images of textures taken from an opensource database. (a) Marble (b) Sand

For $\mathbf{B}$, I use an edge filter from the Leung-Malik (LM) filter bank, which consists of 48 derivatives of Gaussians, providing a mix of edge, bar, and spot filters at multiple scales and orientations. For this simulation I focus on a simple horizontal edge filter for visual clarity.

Next, I initialize $\mathbf{u}_i$ and $\mathbf{x}_i$ to two randomly generated sparse vectors. Figure 4 illustrates how $\mathbf{A}$ convolved with the randomly generated $\mathbf{x}$ generates an image where $\mathbf{A}$ is precisely replicated at each non-zero location in $\mathbf{x}_i$. Figure 5 illustrates the analogous process for the convolution of $\mathbf{B}$ and $\mathbf{u}$.



(a) $\mathbf{A}$ (b) $\mathbf{x}_i$ (c) $\mathbf{A} * \mathbf{x}_i$

Figure 4: Generative process for $\mathbf{A}\mathbf{x}_i$. (a) Texture filter, $\mathbf{A}$ (b) Texture encoding, $\mathbf{x}_i$, where the white pixels represent the 5 non-zero values (c) The final product of a linear convolution, $\mathbf{A} * \mathbf{x}_i$

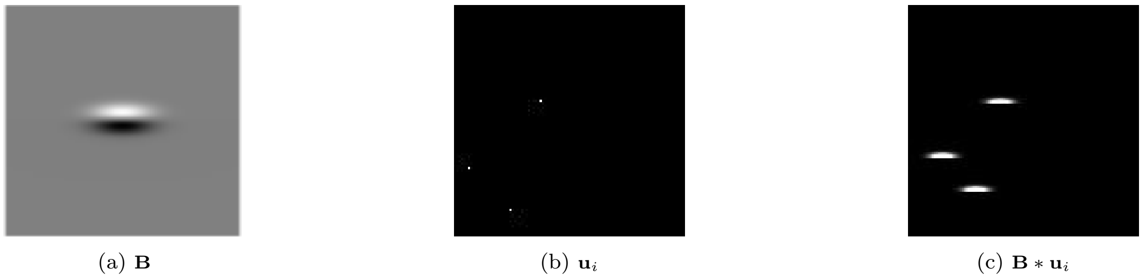


Figure 5: Generative process for $\mathbf{Bu}$. (a) Edge filter, $\mathbf{B}$ (b) Content signal, $\mathbf{u}$, where the white pixels represent the three non-zero values (c) The final product of a linear convolution, $\mathbf{B} * \mathbf{u}_i$

While $\mathbf{x}_i$ is intended to be a relatively dense signal, for illustration purposes I keep $\mathbf{x}_i$ sparse in this simulation so its visual effect is more readily observed. As is seen in Figure 6, increasing the number of non-zero values in $\mathbf{x}_i$ leads to a visually incomprehensible overlapping of the texture filter. Here, n-sparse refers to a tensor with n non-zero values. It appears that the effect of $\mathbf{x}_i$ appears to be increasingly dominant in the convolution as $\mathbf{x}_i$ becomes less sparse. Thus initializing $\mathbf{x}_i$ to a dense signal would lead to an image in which the values of $\mathbf{x}_i$ completely dominate the generated image.

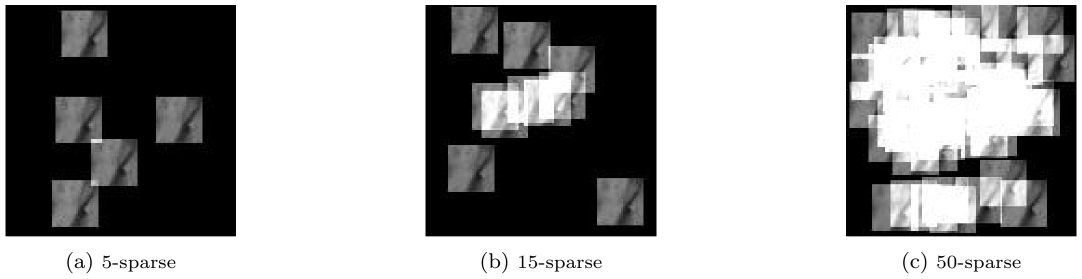


Figure 6: Visualization of $\mathbf{Ax}_i$ for different sparsity levels of $\mathbf{x}_i$, using the same filter $\mathbf{A}$. (a) 5-sparse (b) 15-sparse (c) 50-sparse

Once I have generated $\mathbf{Ax}_i$ and $\mathbf{Bu}_i$, I perform a linear convolution to generate $\mathbf{y}_i$, presented in Figure 7. As expected, $\mathbf{y}_i$ looks like a superposition of the visualizations of $\mathbf{Ax}_i$ and $\mathbf{Bu}_i$ from Figures 4 and 5.

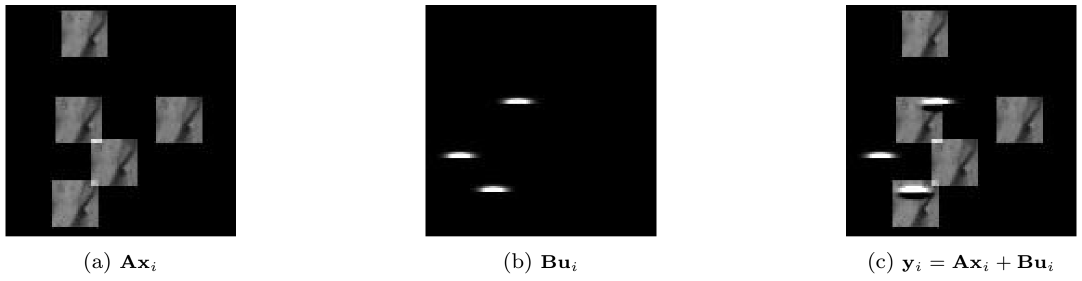
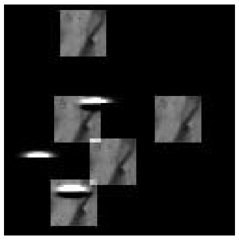


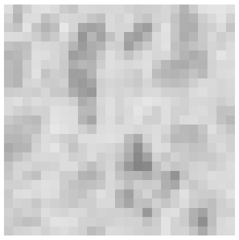
Figure 7: Generation of the image $\mathbf{y}$ through a linear combination of $\mathbf{Ax}$ and $\mathbf{Bu}$.

Finally, I demonstrate a very rudimentary example of style transfer using this hierarchical generative framework by taking advantage of the modularity of the model. Figure 8 shows that I can easily exchange $\mathbf{A}$ for a different filter, $\mathbf{A}^*$, to change the appearance of the texture filter replicated at the locations determined by $\mathbf{x}_i$, while the locations and appearance of the edge filter, $\mathbf{B}$, remain exactly the same.

4.2 Dictionary Learning

Having illustrated the data generation process, I now demonstrate the applicability of FISTA and dictionary learning on the simulated images. Again I stick to the one layer case and start by looking at the texture and


(a) $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i$


(b) $\mathbf{A}^*$

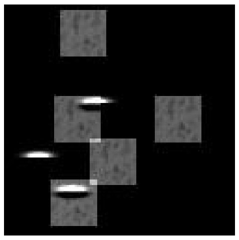
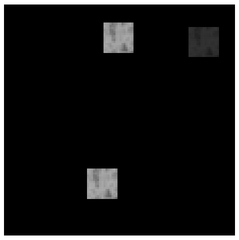

(c) $\mathbf{y}_i^* = \mathbf{A}^*\mathbf{x}_i + \mathbf{B}\mathbf{u}_i$

Figure 8: Demonstration of style transfer, where only the texture filter, $\mathbf{A}$, is changed to $\mathbf{A}^*$ and $\mathbf{B}, \mathbf{u}_i$, and $\mathbf{x}_i$ are kept the same

content components separately. First, I generate images using the $\mathbf{y}_i = \mathbf{A}\mathbf{x}_i$ framework with a 1 channel texture filter of size 5×5 for $\mathbf{A}$ and a randomly generated vector for $\mathbf{x}_i$, resulting in an image of size 100×100. I generate a total of 1000 such sample images and pass these images through the autoencoder framework in batches of 64 to learn both the filter and the signal. Here, I denote $\hat{\mathbf{A}}$ as the filter estimated through dictionary learning and $\hat{\mathbf{x}}$ as the signal estimated through FISTA. Further details regarding the hyperparameters used for FISTA and dictionary learning are specified in Table 2. I verify that the model is capable of recovering both of these components with reasonable accuracy, as can be seen in Figure 9.


(a) Original

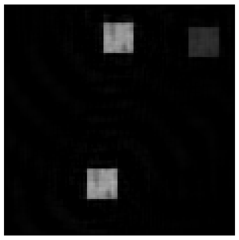
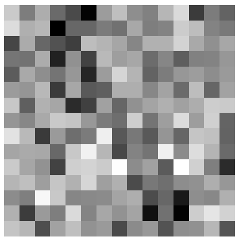

(b) Reconstruction

Figure 9: Comparison of the original $\mathbf{A}\mathbf{x}_i$ to its reconstruction, $\hat{\mathbf{A}}\hat{\mathbf{x}}_i$

(a) Original $\mathbf{A}$


(b) Random initialization of $\hat{\mathbf{A}}$

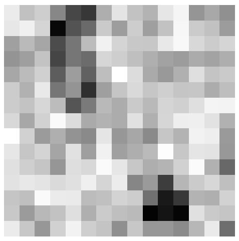
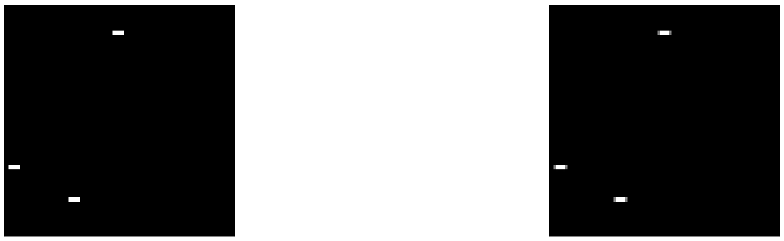

(c) $\hat{\mathbf{A}}$ after training

Figure 10: Comparison of the original texture filter, $\mathbf{A}$, to the filter learned by the model, $\hat{\mathbf{A}}$. (a) Original $\mathbf{A}$ (b) Random Gaussian initialization of $\hat{\mathbf{A}}$ (c) $\hat{\mathbf{A}}$ after training for 150 epochs with a learning rate of 0.02

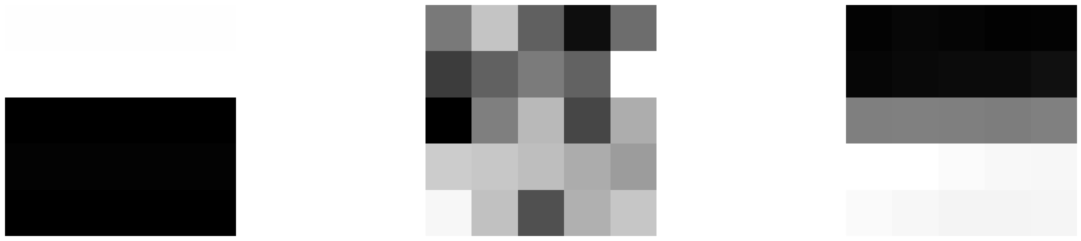
I repeat this process with the framework $\mathbf{y}_i = \mathbf{B}\mathbf{u}_i$. I begin by generating images using a 5×5 edge filter from the LM filter bank for $\mathbf{B}$ and a randomly generated vector for $\mathbf{u}_i$ and pass the resulting images through the model. As before, I generate a total of 1000 images of size 100×100, split into batches of size 64, and use the same hyperparameters listed in Table 2. Again, I verify that the model recovers both the filter $\mathbf{B}$ and the signal $\mathbf{u}_i$ with high accuracy, as is seen in Figure 11.

Of particular significance is the extent to which the learned convolutional filters visually resemble the original filters. While Figures 12(a) and 12(c) appear to be mirror images of one another, this merely demonstrates the fact that the model is only concerned with reproducing the edges present in the input image, which may lead to some variation in the learned filters.



(a) Original (b) Reconstruction

Figure 11: Comparison of the original $\mathbf{B}\mathbf{u}_i$ to its reconstruction, $\hat{\mathbf{B}}\hat{\mathbf{u}}_i$



(a) Original (b) Randomly initialization of $\hat{\mathbf{B}}$ (c) $\hat{\mathbf{B}}$ after training

Figure 12: Comparison of the original edge filter, $\mathbf{B}$, to the filter learned by the model, $\hat{\mathbf{B}}$. (a) Original $\mathbf{B}$ (b) Random Gaussian initialization of $\hat{\mathbf{B}}$ (c) $\hat{\mathbf{B}}$ after training for 40 epochs with a learning rate of 0.02

Sparseness of code ($\|\mathbf{x}_i\|_0$ and $\|\mathbf{u}_i\|_0$)	3
# layers L	1
# filter input channels C_{in}	1
# filter output channels C_{out}	1
Filter kernel size K	5
Dimensions of each sample D	100×100
# samples N	1000
Batch size B	64
# epochs	40
FISTA iterations T	200
FISTA learning rate α	2×10^{-3}
FISTA regularizer λ	0.03
FISTA regularizer γ	0
Dictionary learning rate	0.02

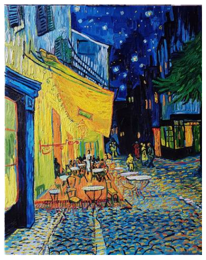
Table 2: Details of datasets and training parameters for experiments on simulated data.

Thus, I have demonstrated that both the filters learned by the model and the encodings obtained through FISTA are visually meaningful and interpretable, and also that the hiearchical image generation process used in this section is highly intuitive.

5 Experimental Results

Having demonstrated the viability of the hierarchical generative framework and developed intuition for style transfer using simulated images, I move to experimenting on natural images. Returning to our original motivating application in the artistic domain, I train the hierarchical model on datasets of images generated using paintings by historically prominent artists. To generate the first dataset, $\mathcal{D}_{VG}$, each of the three Van Gogh paintings shown in Figure 13 are randomly cropped to 100×100 sized images 100 times, to generate a dataset with a total of 300 image croppings (100 per painting). The filters learned from this dataset will be referred to as $\hat{\mathbf{A}}_{VG}$ and $\hat{\mathbf{B}}_{VG}$. To generate the second dataset, $\mathcal{D}_{ML}$, the Mona Lisa painting shown in Figure 14 is similarly randomly cropped to 100×100 pixel sized images 100 times, to generate a dataset with a total of 100 image croppings. The corresponding filters learned from this dataset will be referred to as $\hat{\mathbf{A}}_{ML}$ and $\hat{\mathbf{B}}_{ML}$.

(a) *Starry Night*


(b) *Cafe Terrace At Night*

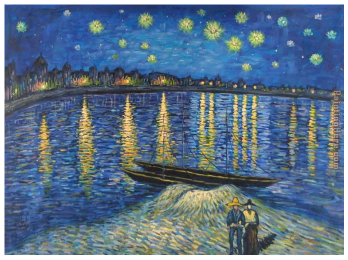

(c) *Starry Night Over The Rhone*

Figure 13: Set of Vincent Van Gogh paintings used to generate samples in $\mathcal{D}_{VG}$.

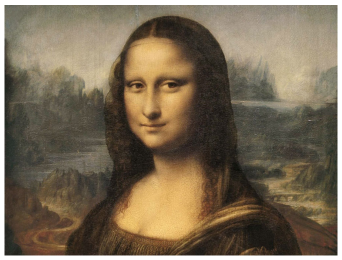
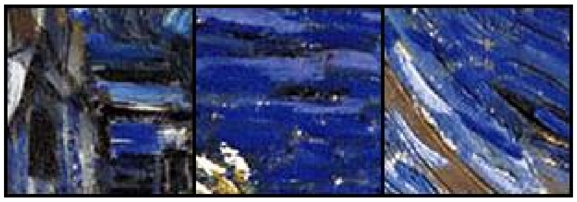


Figure 14: Leonardo da Vinci's *Mona Lisa*

As before, to solve the inverse problem of obtaining the image encodings outlined in Eq. 7, I use the FISTA algorithm, and to learn the filters, I use the stochastic gradient descent method from Section 2.5. The number of layers in the model is kept to $L = 1$, as I found that increasing the number of layers did not significantly change the results but significantly increased training time. The texture filters $\hat{\mathbf{A}}_{VG}$ and $\hat{\mathbf{A}}_{ML}$ have 32 channels, while the content filters $\hat{\mathbf{B}}_{VG}$ and $\hat{\mathbf{B}}_{ML}$ have just 1 channel. All filters are of size 5×5. The filter training procedure is unsupervised and minimizes the reconstruction error of the input images, detailed in Eq. 19. Further details of the dataset and hyperparameters used in training are summarized in Table 3.

Figures 15 and 17 confirm that we can successfully reconstruct images using the filters and encodings learned from both the $\mathcal{D}_{VG}$ and $\mathcal{D}_{ML}$ datasets. More importantly, Figures 16 and 18 illustrate the type of visual information learned by the filters $\hat{\mathbf{A}}_{VG}$ and $\hat{\mathbf{A}}_{ML}$. Figure 16(b) clearly captures the distinctive shades of blue, yellow, brown, and black present in the Van Gogh paintings while Figure 18(b) represents the more subdued shades of brown and green present in the *Mona Lisa* painting. Combined with the intuition for linear convolutions and combinations developed in the previous section with simulated images, these two figures also provide some indication as to how a series of such operations can generate an image as varied and complex as Van Gogh's *Starry Night* or Da Vinci's *Mona Lisa*.


(a) Original Samples

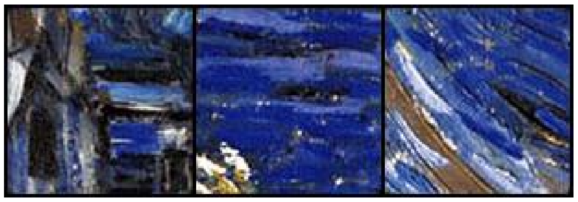
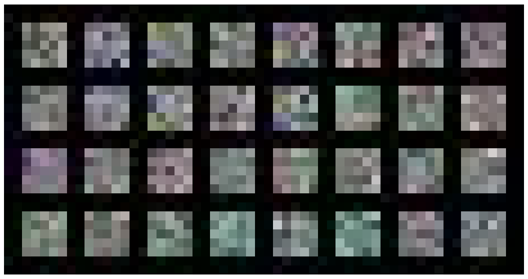

(b) Reconstructed Samples

Figure 15: Reconstruction of sample images in $\mathcal{D}_{VG}$


(a) Filters $\hat{\mathbf{A}}_{VG}$ before training

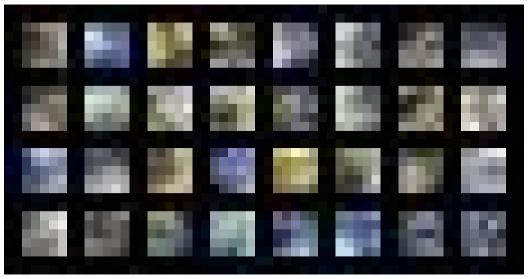
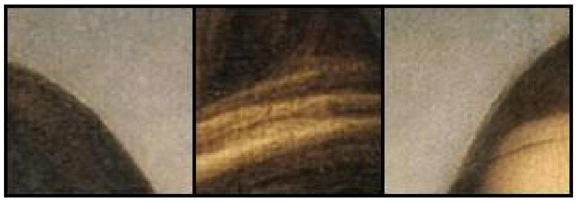

(b) Filters $\hat{\mathbf{A}}_{VG}$ after training

Figure 16: Texture filters $\hat{\mathbf{A}}_{VG}$ learned from $\mathcal{D}_{VG}$; 32 channels. (a) Random initialization, before training (b) After training


(a) Original samples

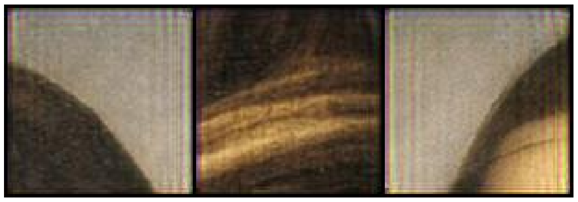
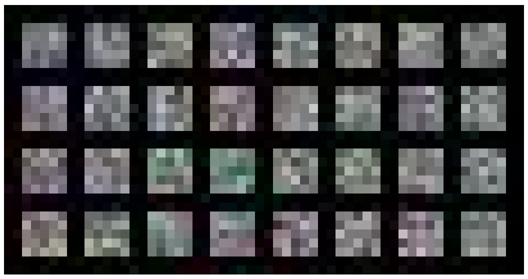

(b) Reconstructed samples

Figure 17: Reconstruction of sample images in $\mathcal{D}_{ML}$


(a) Filters $\hat{\mathbf{A}}_{ML}$ before training

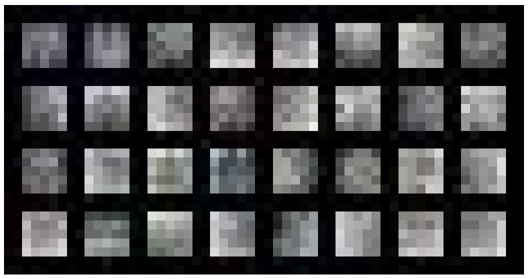

(b) Filters $\hat{\mathbf{A}}_{ML}$ after training

Figure 18: Texture filters $\hat{\mathbf{A}}_{ML}$ learned from $\mathcal{D}_{ML}$; 32 channels. (a) Random initialization, before training (b) After training

5.1 Style Transfer

Once we have learned the filters $\hat{\mathbf{A}}_{VG}, \hat{\mathbf{B}}_{VG}, \hat{\mathbf{A}}_{ML}$, and $\hat{\mathbf{B}}_{ML}$, I can conduct a style transfer operation much like in Section 4. Let $\mathbf{y}_{ML}$ represent the original *Mona Lisa* painting and let $\mathbf{y}_t$ represent the output of the style transfer operation. I start by obtaining the encodings $\hat{\mathbf{x}}_{ML}$ and $\hat{\mathbf{u}}_{ML}$ such that the error between the reconstructed image $\hat{\mathbf{y}}_{ML} = \hat{\mathbf{A}}_{ML} * \hat{\mathbf{x}}_{ML} + \hat{\mathbf{B}}_{ML} * \hat{\mathbf{u}}_{ML}$ and the original image $\mathbf{y}_{ML}$ is minimized. I do this via a single pass through the FISTA algorithm using the filters $\hat{\mathbf{A}}_{ML}$ and $\hat{\mathbf{B}}_{ML}$. Then, I exchange $\hat{\mathbf{A}}_{ML}$ for $\hat{\mathbf{A}}_{VG}$ and generate $\mathbf{y}_t$ as follows: $\mathbf{y}_t = \hat{\mathbf{A}}_{VG} * \hat{\mathbf{x}}_{ML} + \hat{\mathbf{B}}_{ML} * \hat{\mathbf{u}}$.

	Van Gogh	Mona Lisa
# layers L	1	1
# filter input channels for $\mathbf{A}$ C_{in}	3	3
# filter input channels for $\mathbf{B}$ C_{in}	3	3
# filter output channels for $\mathbf{A}$ C_{out}	32	32
# filter output channels for $\mathbf{B}$ C_{out}	1	1
Filter kernel size K	5×5	5×5
Dimensions of each sample D	100×100	100×100
# samples N	300	100
Batch size B	64	64
# epochs	75	75
Optimizer	Adam	Adam
FISTA iterations T	1000	1000
FISTA learning rate α	2×10^{-4}	2×10^{-4}
FISTA regularizer λ	1×10^{-3}	1×10^{-3}
FISTA regularizer γ	0	0
Dictionary learning rate	0.05	0.05

Table 3: Details of datasets and training parameters for experiments on natural images.

The results of this operation are shown in Figure 19. As expected, Figure 19(b) retains most of the content in the original painting but displays a color palette more in line with the Van Gogh paintings in Figure 13.

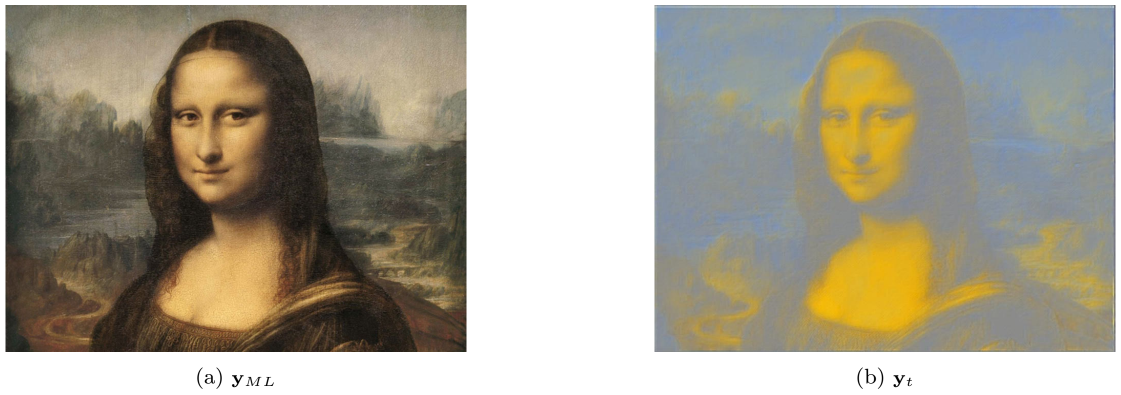

(a) $\mathbf{y}_{ML}$ (b) $\mathbf{y}_t$

Figure 19: Visualization of the *Mona Lisa* before and after conducting a style transfer. (a) Original *Mona Lisa* painting (b) *Mona Lisa* rendered using the filters $\hat{\mathbf{A}}_{VG}$

6 Conclusion

The primary contribution of this manuscript is a visual illustration and interpretation of the convolutional filters in the context of dictionary learning. These visualizations were obtained by training an autoencoder framework based on convolutional sparse coding and dictionary learning first on simplified, grays-scale, simulated images, and then on more complex, colored, natural images. In both contexts, I also demonstrated the application of a hierarchical generative model to decompose images into shared filters and image-specific encodings that can be used both to faithfully reconstruct images and to conduct arbitrary style transfer.

Future work will investigate the extent to which the learned filters can capture diversity in images, i.e. determining the relationship between the variance in the images of a dataset and the number of filters required to fully capture relevant information in that dataset. However, style transfer is only one of several potential applications of the convolutional sparse coding and dictionary learning based autoencoder framework used in this project. In particular, image segmentation tasks would be a high potential continuation of this work, as I have already illustrated that the convolutional filters in the model learn recognizable segments of the input image. Filters with larger sizes may visually align more closely with specific components of an input image. Image classification is another high potential area, one that [22] has already examined in the context of the MNIST database but could be extended to more varied and complex datasets, particular ones with colored images.

7 Acknowledgements

I am incredibly grateful towards a number of people without whom this project would not have been possible. First and foremost, I would like to thank Bahareh Tolooshams and Javier Zazo for their invaluable guidance, numerous discussions, and mentorship throughout this project. I would also like to thank my advisor, Demba Ba, for providing further direction and suggestions, as well as taking the time to read and evaluate this work along with my second reader, Hanspeter Pfister. Additionally, I must thank the associate directors of undergraduate studies in the Applied Mathematics Department, Margo Levine and Sarah Iams, for their assistance in identifying a thesis topic I was truly excited and navigating a very late course switch.

Last but not least, I would like to thank my friends and family, who have been with me from conception to completion of this work. In particular, thank you to my mother for always believing in me, even when I was plagued with self-doubt, and Adriano Borsa for his unwavering support and companionship.

References

[1] Hilton Bristow, Anders Eriksson, and Simon Lucey. "Fast Convolutional Sparse Coding". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Portland, OR, USA: IEEE, June 2013, pp. 391–398. ISBN: 978-0-7695-4989-7. DOI: 10.1109/CVPR.2013.57. URL: http://ieeexplore.ieee.org/document/6618901/ (visited on 01/25/2020).

[2] Jeremy S. De Bonet. "Multiresolution Sampling Procedure for Analysis and Synthesis of Texture Images". In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. USA: ACM Press/Addison-Wesley Publishing Co., Aug. 3, 1997, pp. 361–368. ISBN: 978-0-89791-896-1. DOI: 10.1145/258734.258882. URL: https://doi.org/10.1145/258734.258882 (visited on 03/02/2020).

[3] Julie Dorsey et al. "Modeling and Rendering of Weathered Stone". In: *ACM SIGGRAPH 2005 Courses*. SIGGRAPH '05. Los Angeles, California: Association for Computing Machinery, July 31, 2005, 4–es. ISBN: 978-1-4503-7833-8. DOI: 10.1145/1198555.1198697. URL: http://doi.org/10.1145/1198555.1198697 (visited on 03/02/2020).

[4] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. "A Learned Representation For Artistic Style". In: (Feb. 9, 2017). arXiv: 1610.07629 [cs]. URL: http://arxiv.org/abs/1610.07629 (visited on 11/30/2019).

[5] A.A. Efros and T.K. Leung. "Texture Synthesis by Non-Parametric Sampling". In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Proceedings of the Seventh IEEE International Conference on Computer Vision. Vol. 2. Sept. 1999, 1033–1038 vol.2. DOI: 10.1109/ICCV.1999.790383.

[6] Michael Elad and Peyman Milanfar. "Style-Transfer via Texture-Synthesis". In: *IEEE Transactions on Image Processing* 26.5 (May 2017), pp. 2338–2351. ISSN: 1057-7149, 1941-0042. DOI: 10.1109/TIP.2017.2678168. arXiv: 1609.03057. URL: http://arxiv.org/abs/1609.03057 (visited on 03/02/2020).

[7] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: (Sept. 2, 2015). arXiv: 1508.06576 [cs, q-bio]. URL: http://arxiv.org/abs/1508.06576 (visited on 11/01/2019).

[8] David J. Heeger and James R. Bergen. "Pyramid-Based Texture Analysis/Synthesis". In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '95. New York, NY, USA: Association for Computing Machinery, Sept. 15, 1995, pp. 229–238. ISBN: 978-0-89791-701-8. DOI: 10.1145/218380.218446. URL: https://doi.org/10.1145/218380.218446 (visited on 03/02/2020).

[9] *Image Analogies — Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. Library Catalog: dl-acm-org.ezp-prod1.hul.harvard.edu. URL: http://dl.acm.org/doi/abs/10.1145/383259.383295 (visited on 03/02/2020).

[10] *Image Quilting for Texture Synthesis and Transfer — Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. Library Catalog: dl-acm-org.ezp-prod1.hul.harvard.edu. URL: http://dl.acm.org/doi/abs/10.1145/383259.383296 (visited on 03/02/2020).

[11] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 694–711. ISBN: 978-3-319-46475-6. DOI: 10.1007/978-3-319-46475-6_43.

[12] Stéphane Mallat. *A wavelet tour of signal processing*. Elsevier, 1999.

[13] Stephane G Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE transactions on pattern analysis and machine intelligence* 11.7 (1989), pp. 674–693.

[14] R. Paget and I.D. Longstaff. "Texture Synthesis via a Noncausal Nonparametric Multiscale Markov Random Field". In: *IEEE Transactions on Image Processing* 7.6 (June 1998), pp. 925–931. ISSN: 1941-0042. DOI: 10.1109/83.679446.

[15] Kris Popat and Rosalind W. Picard. "Novel Cluster-Based Probability Model for Texture Synthesis, Classification, and Compression". In: *Visual Communications and Image Processing '93*. Visual Communications and Image Processing '93. Vol. 2094. International Society for Optics and Photonics, Oct. 22, 1993, pp. 756–768. DOI: 10.1117/12.157992. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/2094/0000/Novel-cluster-based-probability-model-for-texture-synthesis-classification-and/10.1117/12.157992.short (visited on 03/02/2020).

[16] E.P. Simoncelli and J. Portilla. "Texture Characterization via Joint Statistics of Wavelet Coefficient Magnitudes". In: *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*. Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269). Vol. 1. Oct. 1998, 62–66 vol.1. DOI: 10.1109/ICIP.1998.723417.

[17] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP". In: *arXiv preprint arXiv:1906.02243* (2019).

[18] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. ISSN: 2517-6161. DOI: 10.1111/j.2517-6161.1996.tb02080.x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x (visited on 01/25/2020).

[19] Dmitry Ulyanov et al. "Texture Networks: Feed-Forward Synthesis of Textures and Stylized Images". In: (), p. 9.

[20] Andrew Witkin and Michael Kass. "Reaction-Diffusion Textures". In: *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '91. New York, NY, USA: Association for Computing Machinery, July 1, 1991, pp. 299–308. ISBN: 978-0-89791-436-9. DOI: 10.1145/122718.122750. URL: http://doi.org/10.1145/122718.122750 (visited on 03/02/2020).

[21] Steven Worley. "A Cellular Texture Basis Function". In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, Aug. 1, 1996, pp. 291–294. ISBN: 978-0-89791-746-9. DOI: 10.1145/237170.237267. URL: https://doi.org/10.1145/237170.237267 (visited on 03/02/2020).

[22] Javier Zazo, Bahareh Tolooshams, and Demba Ba. "Convolutional Dictionary Learning in Hierarchical Networks". In: (July 23, 2019). arXiv: 1907.09881 [cs, stat]. URL: http://arxiv.org/abs/1907.09881 (visited on 10/07/2019).