

Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis

Chuan Li
Mainz University
Germany

chul i@uni -mai nz. de

Michael Wand
Mainz University
Germany

wandm@uni -mai nz. de

Abstract

This paper studies a combination of generative Markov random field (MRF) models and discriminatively trained deep convolutional neural networks (dCNNs) for synthesizing 2D images. The generative MRF acts on higher-levels of a dCNN feature pyramid, controlling the image layout at an abstract level. We apply the method to both photographic and non-photo-realistic (artwork) synthesis tasks. The MRF regularizer prevents over-excitation artifacts and reduces implausible feature mixtures common to previous dCNN inversion approaches, permitting synthesizing photographic content with increased visual plausibility. Unlike standard MRF-based texture synthesis, the combined system can both match and adapt local features with considerable variability, yielding results far out of reach of classic generative MRF methods.

1. Introduction

The problem of synthesizing content by example is a classic problem in computer vision and graphics. It is of fundamental importance to many applications, including creative tools such as high-level interactive photo editing [1, 2, 12], as well as scientific applications, such as generating stimuli in psycho-physical experiments [8].

In this paper, we specifically consider the problem of data-driven image synthesis: Given an example image, we want to fully automatically create a variant of the example image that looks similar but differs in structure. The intended deviation is controlled by additional constraints provided by the user, ranging from just changing image dimensions to detailed layout specifications. Concretely, we implement this by splitting up the input into a “style” image and a “content” image [8, 12]. The first describes the building blocks the image should be made of, the second constrains their layout. Figure 1 shows an example of style transferred images, where the input images are shown on

Figure 1: By combining deep convolutional neural network with MRF prior, our method can transfer both photorealistic and non-photorealistic styles to new images. Images credited to flickr users mricon (A) and Vidar Schiefloe (B).

the left. Our results are shown on the right. Notice our method produces plausible results for both art to photo and photo to art transfers. In particular, see how meso-structures in the style images, such as the mouth and eyes, are intentionally reused in the synthesized images.

The classic data-driven approach to generative image modeling is based on Markov random fields (MRFs): We assume that the most relevant statistical dependencies in an image are present at a local level and learn a distribution over the likelihood of local image patches by considering all local $k \times k$ pixel patches in the example image(s). Usually, this is done using a simple nearest-neighbor scheme [6], and inference is performed by approximate MRF inference [5, 14, 15] or greedy approximations [2, 6, 12, 26].

A critical limitation of MRFs texture synthesis is the difficulty of learning the distribution of plausible image patches from example data. Even the space of local $k \times k$ image patches (typically: $k = 5 \dots 31$) is already way too high-dimensional to be covered with simple sampling and nearest-neighbor estimation. The results are mismatched local pieces, which are subsequently stitched [15] or blended [14] together in order to minimize the perceptual impact of the lack of ability to generalize. The missing ingredient is a strong scheme for interpolating and adapting images from very sparse example sets of sample patches.

In terms of invariance and ability to generalize, discrim-

Figure 2: The input image is encoded by the VGG network (pixel colors show a 3D PCA embedding of the high-dimensional feature space). Related image content is mapped to semi-distributed, approximately spatially coherent feature constellations of increasing invariance [29]. Input image credited to flickr user Emery Way.

inatively trained deep convolutional neural networks have recently made dramatic impact [13, 24]. They are able to recognize complex classes of image features, modeling non-linear deformations and appearance variations far beyond the abilities of simple nearest neighbors search.

However, the discriminative design poses a problem: The corresponding dCNNs compress image information progressively over multiple pooling layers to a very coarse representation, which encodes semantically relevant feature descriptors in a semi-distributed (not spatially localized) fashion (Figure 2). While an inverse process can be defined [3, 4, 7, 8, 9, 20, 28, 29], it remains difficult to control: For example, simply maximizing the class-excitation of the network leads to hallucinatory patterns [21]. Rather than that, we need to reproduce the correct statistics for neural encoding in the synthesis images.

Addressing this problem, Gatys et al. [7, 8] have recently demonstrated remarkable results for transferring styles to guiding “content” images: Their method uses the filter pyramid of the VGG network [24] as a higher-level representation of images, benefitting from the vast knowledge acquired through training the dCNN on millions of photographs. Then, feature layout is simply controlled by penalizing the difference of the high-level neural encoding of the newly synthesized image to that of the content image. Further, the process is regularized by matching feature statistics of the “style” image and the newly synthesized picture by matching the correlations across the various filter channels, captured by a Gram matrix. The method yields very impressive results for applying artistic styles of paintings to photographs [7]. However, strict local plausibility remains difficult. In particular, using photographs as styles does not yield plausible results because only per-pixel feature correlations are captured at different layers and the spatial layout is constrained too weakly.

Our paper augments their framework by replacing the bag-of-feature-like statistics of Gram-matrix-matching by an MRF regularizer that maintains local patterns of the “style” exemplar: MRFs and dCNNs are a canonical combination — both models crucially rely on the assumption of locally correlated information and translational invariance. This equips the encoding of features in a dCNN

with approximate Markovian consistency properties: Local patches have characteristic arrangements of feature activations to describe objects, and higher-up encoding becomes more invariant under in-class variation (Figure 2). We therefore use the generative MRF model on such higher levels of the network (relu3.1 and relu4.1 of the 19-layer VGG network¹). This prescribes a plausible local layout of objects and, importantly, tries to ensure a consistent encoding of these higher-level features. The task of generalizing within object categories and plausible blending is then performed by the dCNN’s lower levels via inversion [20].

Technically, we implement the additional MRF prior by an additional energy term that models Markovian consistency of the upper layers of the dCNN feature pyramid. We then utilize the EM algorithm of Kwatra et al. [14] for MRF optimization: It easily integrates in the variational framework. Further, we will show that higher-level neural encodings are more perceptually linear, which matches well with the linear blending approach of the M-step.

We apply our method to a number of photo-realistic and non-photo-realistic image synthesis tasks, and show it is able to generalize among image patches far beyond the abilities of classic MRFs. In style-transfer scenarios, the combined method additionally benefits from the abilities of the dCNN to match semantically related image portions automatically, without user annotations. In comparison to previous methods that invert dCNNs, the MRF prior improves local plausibility of the feature layouts, avoiding hallucinatory artifacts and usually providing a more plausible meso-structure than the statistical approach of Gatys et al. [7]. In particular, we can strongly improve the plausibility of synthesizing photographs, which was not possible with the previous methods.

2. Related Work

Image synthesis with neural networks: The success of dCNNs in discriminative tasks [23] has also raised in-

¹Notice that in Figure 2 layer relu5.1 shows the most discriminative encoding for single pixels. However, in practice we found using 3×3 patches at layer relu4.1 produce the best synthesis results. Intuitively, using patches from a slightly lower layer has similar matching performance, but permits overlapped MRFs and increased details in synthesis.

terest in generative variants. Zeiler et al. [29] introduce a deconvolutional network to back-project neuron activations to pixels. Similarly, Mahendran and Vedaldi [20] reconstruct images from the neural encoding in intermediate layers. The work of Gatys et al [7], detailed above, can also be employed in unguided settings [8], outperforming traditional parametric texture synthesis which only uses a linear feature bank and no statistical priors [22].

A further approach is the framework of generative adversarial networks [10]. Here, two networks, one as the discriminator and other as the generator iteratively improve each other by playing a minmax game. In the end the generator is able to produce more natural images than naive image synthesis. However, in many cases the output quality is still rather limited. Gauthier et al. [9] extend this model by a Laplacian pyramid. This leads to clear improvement on output quality. Nonetheless, training for large images remains expensive and the results often still lack structure. Denton et al. [3] extend the model to a conditional setting, limited to generating faces. It is also possible to re-train networks for specific generative tasks, such as image deblur [28], super-resolution [4], and class visualization [19].

MRF-based image synthesis: MRFs are the classic framework for non-parametric image synthesis [6]. As explained in the introduction, a key issue is adapting local patches beyond simple stitching [15] or blending [14], and our paper focuses on this issue. Aside from this, MRF models suffer from a second, significant limitation: Local image statistics is usually not sufficient for capturing complex image layouts at a global scale. While local details appear plausible, the global arrangement often merely resembles an unstructured “texture soup”. Multi-resolution synthesis [12, 14, 26] provides some improvement here (and we adapt this in our method, too), but a principled solution requires additional high-level constraints. These can be either explicitly provided by the user [1, 2, 5, 7, 12, 16, 27], or learned from non-local image statistics [11, 18, 30]. Long range correlations have also been modeled by spatial LSTM neural networks; results so far are still limited to semi-regular textures [25]. Our paper opts for the first, simpler solution of explicit layout constraints through a “content” image [7, 12] — in principle, learning of global layout constraints is mostly orthogonal to our approach.

3. Model

We now discuss our combined MRFs and dCNNs model for image synthesis. We assume that we are given a style image, denoted by $x_s \in \mathbb{R}^{w_s \times h_s}$, and a content image $x_c \in \mathbb{R}^{w_c \times h_c}$ for guidance. The (yet unknown) synthesized image is denoted by $x \in \mathbb{R}^{w_c \times h_c}$. We transfer the style of x_s into the layout of x_c by making the high-level neural encoding of x similar to x_c , but using local patches similar to those of x_s . The latter is the MRF prior that maintains the

encoding of the style. Formally, x minimizes the following energy function:

$$x = \arg \min_x E_s(x, x_s) + \lambda_1 E_c(x, x_c) + \lambda_2 R(x) \quad (1)$$

E_s denotes the style loss function (MRFs constraint), where $\phi(x)$ is x ’s feature map from some layer in the network. E_c is the content loss function. It computes the squared distance between the feature map of the synthesis image and that of the content guidance image x_c . As shown in [7, 20], minimizing E_c generates an image that is contextually related to x_c . The additional regularizer $R(x)$ is a smoothness prior on the reconstruction. Next, we explain how to define these terms in details.

MRFs loss function: Let $\mathcal{P}(x)$ denote the list of all local patches extracted from x — a specified set of feature maps of x . Each “neural patch” is indexed as $\phi_i(x)$ and of the size $k \times k \times C$, where k is the width and height of the patch, and C is the number of channels for the layer where the patch is extracted from. We set the energy function to

$$E_s(x, x_s) = \sum_{i=1}^m \|\phi_i(x) - \text{NN}(i)(x_s)\|^2 \quad (2)$$

Here m is the cardinality of $\mathcal{P}(x)$. For each patch $\phi_i(x)$ we find its best matching patch $\text{NN}(i)(x_s)$ using normalized cross-correlation over all m_s example patches in $\mathcal{P}(x_s)$:

$$\text{NN}(i) := \arg \min_{j=1, \dots, m_s} \frac{\phi_i(x) \cdot \phi_j(x_s)}{|\phi_i(x)| \cdot |\phi_j(x_s)|} \quad (3)$$

We use normalized cross-correlation to achieve stronger invariance. The matching process can be efficiently executed by an additional convolutional layer (explained in the implementation details). Notice although we use normalized cross-correlation to find the best match, their Euclidean distance is minimized in Equation 2 for producing an image that is visually close to the reference style.

Content loss function: E_c guides the content of the synthesized image by minimizing the squared Euclidean distance between $\phi(x)$ and $\phi(x_c)$:

$$E_c(x, x_c) = \|\phi(x) - \phi(x_c)\|^2 \quad (4)$$

Regularizer: There is significant amount of low-level image information discarded during the discriminative training of the network. In consequence, reconstructing an image from its neural encoding can be noisy and unnatural. For this reason, we penalize the squared gradient norm [20] to encourage smoothness in the synthesized image:

$$R(x) = \sum_{i,j} (x_{i,j+1} - x_{i,j})^2 + (x_{i+1,j} - x_{i,j})^2 \quad (5)$$

Figure 3: Comparison of patch matching at different layers of a VGG network.

Minimization: We minimize Equation 1 using back-propagation with Limited-memory BFGS. In particular, the gradient of E_s with respect to the feature maps is the element-wise difference between (\times) and their MRFs based reconstruction using patches from (\times_s) . Such a reconstruction is essentially a texture optimization process [14] that uses neural patches instead of pixel patches. It is crucially important to optimize this MRF energy at the neural level, as the traditional pixel based texture optimization will not be able produce results of comparable quality.

Weight The α_1 and α_2 are weights for the content constraint and the natural image prior, respectively. We set $\alpha_1 = 0$ for non-guided synthesis. By default we set $\alpha_1 = 1$ for style transfer, while user can fine tune this value to interpolate between the content and the style. α_2 is fixed to 0.001 for all cases.

4. Analysis

Our key insight is that combining MRF priors with dCNN can significantly improve synthesis quality. This section provides a detailed analysis of our method from three perspectives: we first show compared to pixel values, neural activation leads to better patch matching and blending. We then show how MRFs can further improve the results.

4.1. Neural Matching

A key component of non-parametric image synthesis is to match the synthesized data with the example. (Figure 3 is a toy example that shows neural activation gives better matching than pixels. The task is to match two different car images. The first column contains the query patches from one car; every other column shows the best matching in the other car, found at different feature maps (including the pixel layer).

Figure 4: Linear blending behave differently in pixel space and in neural space. Input photos credited to: flickr user Jsome1 (cat A), flickr user MendocinoAnimalCare (cat B).

It is clear that patch matching using pixels or neural activation at the lower layers (such as relu2_1) is sensitive to appearance variation. The neural activations at layers relu3_1 and relu4_1 give better results. The top layers (relu5_1) seem to decrease the match quality due to the increasing invariance against appearance variation. This is not surprising because the features at middle layers are usually trained for recognizing object parts, as discussed in [29]. For these reason, we use neural patches at layers relu3_1 and relu4_1 as MRFs to constrain the synthesis process.

4.2. Neural Blending

The least-squared optimization for minimizing the texture term (E_s in Equation 2) leads to a linear blending operation for overlapping patches. Here we show that blending neural patches often works better than directly blending pixel patches. Specifically, we compare two groups of blending results: The first method is to directly blend the pixels of two input patches. The second method passes these patches through the network and blend their neural activations at different layers. For each layer, we then reconstruct the blending result back into the pixel space using the method described in [20].

Figure 4 compares the results of these two methods. The first two columns are the input patches A and B for blending. They are intentionally chosen to be semantically related and structurally similar, but are significantly different in pixel values. The third column shows the average of these two patches. Each of the remaining column shows the reconstruction of the blending at a different layer. It is clear that pixel based and neural based blending give very different results: averaging the pixels often gives strong ghost artifacts. This can be reduced as we dive deeper into the network: through experiments we observed that the middle level layers such as relu3_1 and relu4_1 often give more meaningful blendings. Lower layers such as relu2_1 behave similarly to pixels; reconstruction from layers beyond relu4_1 tends to be too fuzzy to be used for synthesis. This also matches our previous observation of middle layers' privilege for patch matching – because a representation that gives better discriminative performance is more robust to noise and enables better interpolation.

Figure 5: Effect of MRFs prior in neural based synthesis. We do not show the example patches here because they are visually identical to the synthesized patches cropped from our results (top). In contrast, patches cropped from [7]’s synthesis results have severe distortion and smear (bottom).

4.3. Effect of the MRF Prior

Despite the advantages in patch matching and blending, it is still possible for a dCNN to generate implausible results. For example, the matched patches from different layers might not always fire at the same place in the image space, indicated by the offsets between the patches found at layer relu3_1 and relu4_1 (Figure 3). The blending may also produce artifacts, for example, the destructed face of the dog (relu4_1, figure 4) and the ghosting eyes of the cat (relu3_1, figure 4). Extreme cases can be found at [21] which produces hallucinogenic images by simply stimulating neural activations without any constraint of the natural image statistics.

More natural results can be obtained through additional regularizer such as smoothness (total variation) or the Gram matrix of pixel-wise filter responses [20, 7]. Our approach adds an MRF regularizer to the middle / upper levels of the network. To show the benefits of the MRFs prior, we compare the synthesis results with and without the constraint. We compare against the “style constraint” based on matching Gram matrices from [7]. Figure 5 validates the intended improvements in local consistency. The first row shows image patches cropped from our results. They are visually consistent to the patches in the original style images. In contrast, [7] produces artifacts such as distortions and smears. The new MRF prior reduces flexibility in local adaptation in favor of reproducing meso-scale features more faithfully.

5. Implementation Details

This section describes implementation details of our algorithm². We use the pre-trained 19-layer VGG-Network from [24]. The synthesis \times is initialized as random noise, and iteratively updated by minimizing Equation 1 using back-propagation. We use layer relu3_1 and relu4_1 for MRFs prior, and layer relu4_2 for content constraint.

For both layer relu3_1 and relu4_1 we use 3×3 patches. To achieve the best synthesis quality we set the stride to

²We release code at: <https://github.com/chuanli11/CNNMRF>

Figure 6: Comparison with Gatys et al. [7] for artistic synthesis. Content Images credited to flickr users Christopher Michel and their.

one so patches are very densely sampled. The patch matching (Equation 3) is implemented as an additional convolutional layer for fast computation. In this case patches sampled from the style image are treated as the filters. The best matching of a query patch is the filter that gives the maximum response. We can pre-computed the magnitude of the filters ($\|i(\times_t)\|$ in Equation 3) as they will not change during the synthesis. Unfortunately the magnitude of patches in the synthesized image ($\|i(\times)\|$) needs to be computed on the fly.

In practice we use a multi-resolution process: We built a image pyramid using the scaling factor of two, and stop when the longest dimension of the synthesized image is less than 64 pixels. The reference texture and reference content images are scaled accordingly. We perform 200 iterations for each resolution, and the output of the previous resolution is bi-linearly up-sampled as the initialization for the next resolution. We implement our algorithm under the Torch framework. Our algorithm take about three minutes to synthesis an image of size 384×384 with a Titan X GPU.

To partially overcome the perspective and scale difference between the style and the content images, we sample patches from a number of copies of the style image with different rotations and scales: we use seven scales $\{0.85, 0.9, 0.95, 1, 1.05, 1.1, 1.15\}$, and for each scale we create five rotational copies $\{-\frac{1}{12}, -\frac{1}{24}, 0, \frac{1}{24}, \frac{1}{12}\}$. Since increasing the number of patches is computational expensive, in practice we only use the rotational copies for objects that can deform – for example faces.

6. Results

This section discusses our results. Here we focus on style transfer and refer readers to our supplementary ma-

terial for the results of un-guided synthesis. As discussed later in this section, our method has strong restriction with the input data, so a formal user study is less meaningful. For this reason, our evaluation is performed by demonstrating our method with a number of typical successful and failure cases, with comparisons and discussions related to the state of the art neural based image stylization method [7]. We refer readers to our supplementary report for more results.

Figure 6 shows two examples of stylizing photos by artwork. The input photos (left) are stylized using Pablo Picasso’s “Self-portrait 1907” and Wassily Kandinsky’s “Composition VIII”, respectively. In both cases, Gatys et al.’s [7] method (middle) achieves interesting results, preserving the content of the photos and the overall look and feel of the artworks. However, there are weaknesses on closer inspection: The first result (top row) contains many unnecessary details, and the eyes look unnatural. Their second result lost the characteristic shapes in the original painting and partially blends with the content exemplar. In contrast, our first result synthesized more plausible facial features. In the second result, our method also resembles the style better: notice the important facial features such as eyes and the mouth are synthesized as simple shapes. Figure 7 shows two examples of photorealistic synthesis. We transfer the style of a vintage car to two different modern vehicles. Notice the lack of photo-realistic details and strong smears in [7]’s results. With the MRFs constraint (right) our results are closer to photorealistic.

From an informal user study, we observed that [7] usually keeps the content better, and our method produces more accurate styles. Figure 8 gives detailed analysis between these two different characters. Here we show three patches from the content image (red boxes), and their closest matches from the style image and the synthesis images (using neural level matching). Notice our method produces more plausible results when a good match can be found between the content and the style images (the first patch), and performs less well when mis-matching happens (the second patch, where our synthesis deviates from the content image). For the third patch, there is no matching can be found for the car. In this case, our method replaces the car with texture synthesis, while [7] keeps the car and renders it with artifacts. In general, our method creates more stylish images, but may contain artifacts when the MRFs do not fit the content. In contrast, the parametric method [7] is more adaptable to the content, but at the cost of deviating from the style.

Here we summarize the two main differences between [7] and our method based on our experiments: First, by matching local $k \times k$ patches our method imposes the spatial coherence constraint that is missing from [7]. Let us briefly motivate this design: while dCNNs represent images at a more semantically meaningful level, they also create distributed encoding that are difficult to invert. Our obser-

vation is that Markovian consistency of spatial neighborhoods ($k \times k$ patches) reduces this problem: The encodings of a dCNN are (by construction) locally correlated; retaining these correlations improves the synthesis of invertible neural representations. Second, [7] match the global feature distributions using the Gram matrix. Their synthesized features can maneuver within the subspace spanned by the examples. Such behavior can lead to artifacts in the pixel space due to the non-linear inversion of the networks. In contrast, our synthesized features are copies of the examples. Notice [7] is not equivalent to our method with 1×1 patches, in which case the difference between statistical matching and data copying still holds.

Last but not the least, we discuss the difference between our method and a previous work [17] that used abstract guidance features for improving synthesis. First, from the perspective of guidance, the VGG network has learned very strong invariance, far beyond hand-crafted shallow features. It can match semantically related features that have strong appearance variation (e.g.: Figure 3). Second, it also improves synthesis itself: We do not directly synthesize pixels but neural activations on a higher network layer. This representation still has strong generalization ability and can plausibly blend image patches together even if their image patches are quite different (e.g.: Figure 4).

6.1. Limitations

Our method is an interesting extension for image based style transfer, especially for photorealistic styles. Nonetheless, it has many limitations. First and for most, it is more restricted to the input data: it only works if the content image can be re-assembled by the MRFs in the style image. For example, images of strong perspective or structure difference are not suitable for our method.

Figure 9 shows a typical example case where [7] works better. In this case the artistic style can be more easily transferred by the parametric method: Notice how the textures adapt to the content more naturally in [7]’s method. In contrast, our method tries to “reshuffle” building blocks in the style image and lost important features in the content image. In general, our method works better for subjects that allows structural deformation, such as faces and cars. For subjects that have strict symmetry properties such as architectures, it is often that our method will generate structural artifacts. In this case, structural statistics such as [11] may be used for regularizing the synthesized image.

Although our method achieved improvement for photorealistic synthesis, it is still not as sharp as the original photos. This is due to the loss of non-discriminative image details during the training of the network. This opens an interesting future work that how the dCNN can be retrained, or incorporated with stitching based texture synthesis such as [15] for achieving pixel-level photorealism.

Figure 7: Comparison with Gatys et al. [7] for photo-realistic synthesis. Input images credited to flickr users Brett Levin, Axion23 and Tim Dobbelaere.

Figure 8: Detailed analysis of the difference between our results and Gatys et al.'s [7]'s results. Input images credited to flickr users Eden, Janine and Jim and Garry Knight.

Figure 9: A typical case where Gatys et al [7] works better.

7. Conclusions

The key insight of this paper is that we can combine the discriminative power of a deep neural network with classical MRFs based texture synthesis. We developed a simple method that is able to produce encouraging new results for style transfer between images. We analyzed our results with

a number of typical successful and failure cases, and discussed its pros and cons compared to the state of the art neural based method for transferring image styles. Importantly, our results often preserve better mesostructures in the synthesized image. For this first time, this permits transferring photo-realistic styles with some plausibility. The stricter control of the mesostructure is also the biggest limitation at this point: The MRF prior only offers advantages when style and content images consists of similarly shaped elements without strong changes in perspective, size, or shape, as covered by the invariance of the high-level neural encoding. Otherwise, artifacts might occur. For pure artistic styles, the increased rigidity can then be a disadvantage.

Our work is only one step in the direction of leveraging deep convolutional neural networks for improving image synthesis. It opens many interesting questions and future work such as how to resolve the incompatibility between the structure guidance and the MRFs [11]; how to more efficiently study and transfer the middle level style across a

big dataset; and how to generate pixel-level photorealistic images by incorporating with stitching based texture synthesis [15], or with generative training of the network.

Acknowledgments

This work has been partially supported by the Intel Visual Computing Institute and by the International Max Planck Research School for Computer Science. We thank Hao Su and Daniel Franzen for inspiring discussions.

References

- [1] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen. Interactive digital photomontage. *ACM Trans. Graphics (Proc. Siggraph)*, 23(3):294–302, 2004. **1, 3**
- [2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. (Proc. Siggraph)*, 28(3):24:1–24:11, 2009. **1, 3**
- [3] E. L. Denton, R. Fergus, A. Szlam, and S. Chintala. Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)* 28, 2015. **2, 3**
- [4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks, 2015. ArXiv preprint: <http://arxiv.org/abs/1501.00092>. **2, 3**
- [5] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proc. ACM Siggraph*, pages 341–346, 2001. **1, 3**
- [6] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Washington, DC, USA, 1999. **1, 3**
- [7] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style, 2015. ArXiv preprint: <http://arxiv.org/abs/1508.06576>. **2, 3, 5, 6, 7**
- [8] L. A. Gatys, A. S. Ecker, and M. Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)* 28, May 2015. **1, 2, 3**
- [9] J. Gauthier. Conditional generative adversarial nets for convolutional face generation. <http://www.foldl.me/2015/conditional-gans-face-generation/>, 2015. **2, 3**
- [10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)* 27, pages 2672–2680. 2014. **3**
- [11] K. He and J. Sun. Statistics of patch offsets for image completion. In *Proc. European Conference on Computer Vision (ECCV)*, pages 16–29, 2012. **3, 6, 7**
- [12] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. ACM Siggraph* 2001, pages 327–340, 2001. **1, 3**
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)* 25, pages 1097–1105. 2012. **2**
- [14] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra. Texture optimization for example-based synthesis. *ACM Trans. Graph. (Proc. Siggraph)*, 24(3):795–802, 2005. **1, 2, 3, 4**
- [15] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick. Graphcut textures: Image and video synthesis using graph cuts. *ACM Trans. Graph. (Proc. Siggraph)*, 22(3):277–286, 2003. **1, 3, 6, 8**
- [16] H. Lee, S. Seo, S. Ryoo, and K. Yoon. Directional texture transfer. In *Proc. Int. Symp. on Non-Photorealistic Animation and Rendering (NPAR)*, pages 43–48. ACM, 2010. **3**
- [17] S. Lefebvre and H. Hoppe. Appearance-space texture synthesis. In *SIGGRAPH*, pages 541–548, 2006. **6**
- [18] C. Li and M. Wand. Approximate translational building blocks for image decomposition and synthesis. In *ACM Transactions on Graphics*, 2015. to appear. **3**
- [19] Y. Lu, S. Zhu, and Y. N. Wu. Learning FRAME models using CNN filters for knowledge visualization, 2015. ArXiv preprint: <http://arxiv.org/abs/1509.08379>. **3**
- [20] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, 2015. **2, 3, 4, 5**
- [21] A. Mordvintsev, C. Olah, and M. Tyka. Inceptionism: Going deeper into neural networks. <http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html>, 2015. **2, 5**
- [22] J. Portilla and E. P. Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. Comput. Vision*, 40(1):49–70, Oct. 2000. **3**
- [23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Computer Vision (IJCV)*, pages 1–42, April 2015. **2**
- [24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2014. Preprint: <http://arxiv.org/abs/1409.1556>. **2, 5**
- [25] L. Theis and M. Bethge. Generative image modeling using spatial lstms. In *Advances in Neural Information Processing Systems (NIPS)* 28, Jun 2015. **3**
- [26] L.-Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proc. ACM Siggraph* 2000, pages 479–488, 2000. **1, 3**
- [27] X. Xie, F. Tian, and H. S. Seah. Feature guided texture synthesis (fgts) for artistic style transfer. In *Proc. Int. Conf. Digital Interactive Media in Entertainment and Arts (DIMEA)*, pages 44–49. ACM, 2007. **3**
- [28] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Advances in Neural Information Processing Systems (NIPS)* 27, pages 1790–1798. 2014. **2, 3**
- [29] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *Proc. Europ. Conf. Comp. Vision (ECCV)*, pages 818–833, 2014. **2, 3, 4**
- [30] Y. Zhang, J. Xiao, J. Hays, and P. Tan. Framebreak: Dramatic image extrapolation by guided shift-maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1171–1178, 2013. **3**