“Double-DIP”:
Unsupervised Image Decomposition via Coupled Deep-Image-Priors

Yossi Gandelsman Assaf Shocher Michal Irani
Dept. of Computer Science and Applied Mathematics
The Weizmann Institute of Science, Israel
Project Website: www.wisdom.weizmann.ac.il/~vision/DoubleDIP/

Abstract

Many seemingly unrelated computer vision tasks can be viewed as a special case of image decomposition into separate layers. For example, image segmentation (separation into foreground and background layers); transparent layer separation (into reflection and transmission layers); Image dehazing (separation into a clear image and a haze map), and more. In this paper we propose a unified framework for unsupervised layer decomposition of a single image, based on coupled “Deep-image-Prior” (DIP) networks. It was shown [38] that the structure of a single DIP generator network is sufficient to capture the low-level statistics of a single image. We show that coupling multiple such DIPs provides a powerful tool for decomposing images into their basic components, for a wide variety of applications.

This capability stems from the fact that the internal statistics of a mixture of layers is more complex than the statistics of each of its individual components. We show the power of this approach for Image-Dehazing, Fg/Bg Segmentation, Watermark-Removal, Transparency Separation in images and video, and more. These capabilities are achieved in a totally unsupervised way, with no training examples other than the input image/video itself.¹

1. Introduction

Various computer vision tasks aim to decompose an image into its individual components. In image/video segmen-

¹Project funded by the European Research Council (ERC) under the Horizon 2020 research & innovation programme (grant No. 788535)
Within each separate layer is the distribution of small patches.

In this paper we combine the power of the internal patch recurrence (its strength in solving unsupervised tasks), with the power of Deep-Learning. We propose an unsupervised Deep framework for decomposing a single image into its layers, such that the distribution of “image elements” within each layer is “simple”. We build on top of the “Deep Image Prior” (DIP) work of Ulyanov et al. [38]. They showed that the structure of a single DIP generator network is sufficient to capture the low-level statistics of a single natural image. The input to the DIP network is random noise, and it trains to reconstruct a single image (which serves as its sole output training example). This network was shown to be quite powerful for solving inverse problems like denoising, super-resolution and inpainting, in an unsupervised way.

We observe that when employing a combination of multiple DIPs to reconstruct an image, those DIPs tend to “split” the image, such that the patch distribution of each DIP output is “simple”. Our approach for unsupervised multi-task layer decomposition is thus based on a combination of multiple (two or more) DIPs which we coin “Double-DIP”. We demonstrate the applicability of this approach to a wide range of computer vision tasks, including Image-Dehazing, Fg/Bg Segmentation of images and videos, Watermark Removal, and Transparency Separation in images and videos.

Double-DIP is general-purpose and caters many different applications. Special-purpose methods designed for one specific task may outperform Double-DIP on their own challenge. However, to the best of our knowledge, this is the first framework that is able to handle well such a large variety of image-decomposition tasks. Moreover, in some tasks (e.g., image dehazing), Double-DIP achieves comparable and even better results than leading methods.

2. Overview of the Approach

Observe the illustrative example in Fig. 3a. Two different textures, X and Y, are mixed to form a more complex image Z which exhibits layer transparency. The distribution of small patches and colors inside each pure texture is simpler than the distribution of patches and colors in the combined image. Moreover, the similarity of patches across the two textures is very weak. It is well known [12] that if X and Y are two independent random variables, the entropy of their sum Z = X + Y is larger than their individual entropies: \( \max\{H(X), H(Y)\} \leq H(Z) \). We leverage this fact to separate the image into its natural “simpler” components.

\[
\begin{align*}
\text{Loss} &= \text{Loss}_{\text{Reconst}} + \text{Loss}_{\text{Excl}} + \text{Loss}_{\text{Reg}}
\end{align*}
\]
2.1. Single DIP vs. Coupled DIPs

Let’s see what happens when a DIP network is used to learn pure images versus mixed images. The graph in Fig. 3.c shows the MSE Reconstruction Loss of a single DIP network, as a function of time (training iterations), for each of the 3 images in Fig. 3.a: (i) the orange plot is the loss of a DIP trained to reconstruct the texture image X, (ii) the blue plot – a DIP trained to reconstruct the texture Y, and (iii) the green plot – a DIP trained to reconstruct their superimposed mixture (image transparency). Note the larger loss and longer convergence time of the mixed image, compared to the loss of its individual components. In fact, the loss of the mixed image is larger than the sum of the two individual losses. We attribute this behavior to the fact that the distribution of patches in the mixed image is more complex and diverse (larger entropy; smaller internal self-similarity) than in any of its individual components.

While these are pure textures, the same behavior holds also for mixtures of natural images. The internal self-similarity of patches inside a single natural image tends to be much stronger than the patch similarity across different images [41]. We repeated the above experiment for a large collection of natural images: We randomly sampled 100 pairs of images from the BSD100 dataset [27], and mixed each pair. For each image pair we trained a DIP to learn the mixed image and each of the individual images. The same behavior exhibited in the graph of Fig. 3.c repeated also in this case (see project website).

We further observe that when multiple DIPs train to jointly reconstruct a single input image, they tend to “split” the image patches among themselves. Namely, similar small patches inside the image tend to all be generated by a single DIP network. In other words, each DIP captures different components of the internal statistics of the image. We explain this behavior by the fact that a single DIP network is fully convolutional, hence its filter weights are shared across the entire spatial extent of the image. This promotes self-similarity of patches in the output of each DIP.

The simplicity of the patch distribution in the output of a single DIP is further supported by the denoising experiments reported in [38]. When a DIP was trained to reconstruct a noisy image (high patch diversity/entropy), it was shown to generate along the way an intermediate clean version of the image, before overfitting the noise. The clean image has higher internal patch similarity (smaller patch diversity/entropy), hence is simpler for the DIP to reconstruct.

Building on these observations, we propose to decompose an image into its layers by combining multiple (two or more) DIPs, which we call “Double-DIP”. Figs. 3.a,b show that when training 2 DIP networks to jointly recover the mixed texture transparency image (as the sum of their outputs), each DIP outputs a coherent layer on its own.

2.2. Unified Multi-Task Decomposition Architecture

What is a good image decomposition? There are infinitely many possible decompositions of an image into layers. However, we suggest that a meaningful decomposition satisfies the following criteria: (i) the recovered layers, when recombined, should yield the input image. (ii) Each of the layers should be as “simple” as possible, namely, it should have a strong internal self-similarity of “image elements”. (iii) The recovered layers should be as independent
of each other (uncorrelated) as possible.

These criteria form the basis of our general-purpose Double-DIP architecture, illustrated in Fig. 2. The first criterion is enforced via a “Reconstruction Loss”, which measures the error between the constructed image and the input image (see Fig. 2). The second criterion is obtained by employing multiple DIPs (one per layer). The third criterion is enforced by an “Exclusion Loss” between the outputs of the different DIPs (minimizing their correlation).

Each DIP network (DIP_i) reconstructs a different layer y_i of the input image I. The input to each DIP_i is randomly sampled uniform noise, z_i. The DIP outputs, y_i = DIP_i(z_i), are mixed using a weight mask m, to form a reconstructed image \( \hat{I} = m \cdot y_1 + (1 - m) \cdot y_2 \), which should be as close as possible to the input image I.

In some tasks the weight mask m is simple and known, in other cases it needs to be learned (using an additional DIP). The learned mask m may be uniform or spatially varying, continuous or binary. These constraints on m are task-dependent, and are enforced using a task-specific “Regularization Loss”. The optimization loss is therefore:

\[
L_{\text{oss}} = L_{\text{Reconst}} + \alpha \cdot L_{\text{Excl}} + \beta \cdot L_{\text{Reg}}
\]  

where \( L_{\text{Reconst}} = \| I - \hat{I} \| \), and \( L_{\text{Excl}} \) (the Exclusion loss) minimizes the correlation between the gradients of y_1 and y_2 (as defined in [40]). \( L_{\text{Reg}} \) is a task-specific mask regularization (e.g., in the segmentation task the mask m has to be as close as possible to a binary image, while in the dehazing task the t-map is continuous and smooth). We further apply guided filtering [22] on the learned mask m to obtain a refined mask.

Inherent Layer Ambiguities: Separating a superposition of 2 pure uncorrelated textures is relatively simple (see Fig. 3.a). There are no real ambiguities other than a constant global color ambiguity \( c \cdot I = (y_1 + c) + (y_2 - c) \). Similarly, pure non-overlapping textures are relatively easy to segment. However, when a single layer contains multiple independent regions, as in Fig. 3.b, the separation becomes ambiguous (note the switched textures in the recovered output layers of Fig. 3.b). Unfortunately, such ambiguities exist in almost any natural indoor/outdoor image.

To overcome this problem, initial “hints” are often required to guide the Double-DIP. These hints are provided automatically in the form of very crude image saliency [20]. Namely, in the first few iterations, DIP_1 is encouraged to train more on the salient image regions, whereas DIP_2 is guided to train more on the non-salient image regions. This guidance is relaxed after a few iterations.

When more than one image is available, this ambiguity is often resolved on its own, without requiring any initial hints. For example, in video transparency, the superposition of 2 video layers changes from frame to frame, resulting in different mixtures. The statistics of each layer, however, remains the same throughout the video (despite its dynamics) [34, 33]. This means that a single DIP suffices to represent all the frames of a single video layer. Hence, Double-DIP can be used to separate video sequences into 2 dynamic layers, and can often do so with no initial hints.

Optimization: The architecture of the individual DIPs is similar to that used in [38]. As in the basic DIP, we found that adding extra non-constant noise perturbations to the input noise adds stability in the reconstruction. We gradually increase noise perturbations with iterations. We further enrich the training set by transforming the input image I and the corresponding random noise inputs of all the DIPs using 8 transformations (4 rotations by 90° combined with 2 mirror reflections - vertical and horizontal). Such an augmentation was also found to be useful in the unsupervised internal learning of [35]. The optimization process is done using ADAM optimizer [25], and takes a few minutes per image on Tesla V100 GPU. In the case of video, the runtime grows sub-linearly with the number of frames, since all frames are used to train the same DIP.

3. Segmentation

Fg/Bg segmentation can be viewed as decomposing an image I into a foreground layer y_1 and background layer y_2, combined by a binary mask m(x) at every pixel x:

\[
I(x) = m(x)y_1(x) + (1 - m(x))y_2(x)
\]  

This formulation naturally fits our framework, subject to y_1 and y_2 complying to natural image priors and each being

![Figure 4: Foreground/Background Image Segmentation. (Please see many more results in the project website)](image-url)
Figure 5: Video Decomposition using Double-DIP.

'nicer' to generate than \( I \). This requirement is verified by [5] that defines a 'good image segment' as one which can be easily composed using its own pieces, but is difficult to compose using pieces from other parts of the image.

The Zebra image in top row of Fig. 1 demonstrates the decomposition of Eq. 2. It is apparent that the layers \( y_1 \) and \( y_2 \), generated by \( DIP_1 \) and \( DIP_2 \), each complies with the definition of [5], thus allowing to obtain also a good segmentation mask \( m \). Note that \( DIP_1 \) and \( DIP_2 \) automatically filled-in the 'missing' image parts in each output layer.

In order to encourage the learned segmentation mask \( m(x) \) to be binary, we use the following regularization loss:

\[
\text{Loss}_{\text{Reg}}(m) = \left( \sum_x |m(x) - 0.5| \right)^{-1}
\]

(3)

While Double-DIP does not capture any semantics, it is able to obtain high quality segmentation based solely on unsupervised layer separation, as shown in Fig. 4. Please see many more results in the project website. Other approaches to segmentation, such as semantic-segmentation (e.g., [21]) may outperform Double-DIP, but these are supervised and trained on many labeled examples.

Video segmentation: The same approach can be used for Fg/Bg video segmentation, by exploiting the fact that sequential video frames share internal patch statistics [34, 33]. Video segmentation is cast as 2-layer separation as follows:

\[
I(i)(x) = m(i)(x)y_1(i)(x) + (1 - m(i)(x))y_2(i)(x) \quad \forall i
\]

(4)

where \( i \) is the frame number. Fig. 5 depicts how a single DIP is shared by all frames of a separated video layer.

Figure 6: Video Layer Separation via Double-DIP. Double-DIP exploits the fact that all frames of a single dynamic video layer share the same patches. This promotes: (a) video transparency separation, and (b) Fg/Bg video segmentation. (See full videos in the project website).

We implicitly enforce temporal consistency in the segmentation mask, by imposing temporal consistency on the random noise inputted to the mask DIP in successive frames:

\[
z_m(i+1)(x) = z_m(i)(x) + \Delta z_m(i+1)(x)
\]

(5)

where \( z_m(i)(x) \) is the noise input at frame \( i \) to the DIP that generates the mask. These noises change gradually from frame to frame by \( \Delta z_m(i) \) (which is a random uniform noise with variance significantly lower than that of \( z_m(i) \)).

4. Transparent Layers Separation

In the case of image reflection, each pixel value in image \( I(x) \) is a convex combination of a pixel from the transmission layer \( y_1(x) \) and the corresponding pixel in the reflection layer \( y_2(x) \). This again can be formulated as in Eq. 2, where \( m(x) \) is the reflective mask. In most practical cases, it is safe to assume that \( m(x) \equiv m \) is a uniform mask (with...
Ambiguity can be resolved using external training, as in [40]. However, since Double-DIP is unsupervised, we resolve this ambiguity when 2 different mixtures of the same layers are available. This gives rise to coupled equations:

\[
\begin{align*}
I^{(1)}(x) &= m^{(1)}(x) y_1(x) + (1 - m^{(1)}) y_2(x) \\
I^{(2)}(x) &= m^{(2)}(x) y_1(x) + (1 - m^{(2)}) y_2(x)
\end{align*}
\]  

(6)

Since the layers \(y_1, y_2\) are shared by both mixtures, one Double-DIP suffices to generate these layers using \(I^{(1)}, I^{(2)}\) simultaneously. The different coefficients \(m^{(1)}, m^{(2)}\) are generated by the same DIP using 2 random noises, \(z^{(1)}_m, z^{(2)}_m\). See such an example in Fig. 8 (real transparent images).

**Video Transparency Separation:** The case of a static reflection and dynamic transmission can be solved in a similar way. This case can be formulated as a set of equations:

\[
I^{(i)}(x) = m^{(i)}(x) y_1(x) + (1 - m^{(i)}) y_2(x)
\]  

(7)

where \(i\) is the frame number, and \(y_2\) is the static reflection (hence has no frame index \(i\), but could have varying intensity over time, captured by \(m^{(i)}\)). Applying Double-DIP to a separate transparent video layers is done similarly to video segmentation (Sec. 3). We employ one DIP for each video layer, \(y_1, y_2\) and one more DIP to generate \(m^{(i)}\) (but with a modified noise input per each frame, as in the video segmentation). Fig. 6.a shows examples of video separation. For full videos see the project website.

**Watermark removal:** Watermarks are widely-used for copyright protection of photos and videos. Dekel et al. [14] presented a watermark removal algorithm, based on recurrence of the same watermark in many different images. Double-DIP is able to remove watermarks shared by very few images, often only one.

We model watermarks as a special case of image reflection, where layers \(y_1, y_2\) are the clean image and the watermark, respectively. This time, however, the mask is not a constant \(m\). The inherent transparent layer ambiguity is resolved by one of two practical ways: (i) when only one watermarked image is available, the user provides a crude hint (bounding box) around the location of the watermark; (ii) given a few images which share the same watermark (2-3 typically suffice), the ambiguity is resolved on its own.

When a single image and a bounding box are provided, the learned mask \(m(x)\) is constrained to be zero outside the bounding box. This hint suffices for Double-DIP to perform reasonably well on this task. See examples in Fig. 7. In fact, the Tennis image in Fig. 7 was provided by [14] as an example image immune to their watermark-removal method.

When multiple images contain the same watermark are available, no bounding-box is needed. E.g., if 3 images share a watermark, we use 3 Double-DIPs, which share DIP\(_2\) to output the common watermark layer \(y_2\). Independent layers \(y^{(i)}_1, i=1, 2, 3\), provide the 3 clean images. The
opacity mask \( m \), also common to the 3 images, is generated by another shared DIP. Example is shown in Fig. 9.

5. Image Dehazing

Images of outdoor scenes are often degraded by a scattering medium (e.g., haze, fog, underwater scattering). The degradation in such images grows with scene depth. Typically, a hazy image \( I(x) \) is modeled [23]:

\[
I(x) = t(x) J(x) + (1 - t(x)) A(x)
\]  

where \( A(x) \) is the Airlight map (A-map), \( J(x) \) is the haze-free image, and \( t(x) \) is the transmission (t-map), which exponentially decays with scene depth. The goal of image dehazing is to recover from a hazy image \( I(x) \) its underlying haze-free image \( J(x) \) (i.e., the image that would have been captured on a clear day with good visibility conditions).

We treat the dehazing problem as a layer separation problem, where one layer is the haze-free image \( (y_1(x) = J(x)) \), the second layer is the A-map \( (y_2(x) = A(x)) \), and the mixing mask is the t-map \( (m(x) = t(x)) \).

Handling non-uniform airlight: Most single-image blind dehazing methods (e.g. [23, 28, 18, 6, 8]) assume a uniform airlight color \( A \) for the entire image (i.e., \( A(x) \equiv A \)). This is true also for deep network based dehazing methods [10, 30], which train on synthesized datasets of hazy/non-hazy image pairs. The uniform airlight assumption, however, is only an approximation. It tends to break, e.g. in outdoor images captured at dawn or dusk, when the sun is positioned close to the horizon. The airlight color is affected by the non-isotropic scattering of sun rays by haze particles, which causes the airlight color to vary across the image. When the uniform airlight assumption holds, estimating a single uniform airlight color \( A \) from the hazy \( I \) is relatively easy (e.g., using the dark channel prior of [23], or the patch-based prior of [6]), and the challenging part remains the t-map estimation. However, when the uniform airlight assumption breaks, this produces dehazing results with distorted colors. Estimating a varying airlight, on the other hand, is a very challenging and ill-posed problem. Our Double-DIP framework allows simultaneous estimation of a varying airlight-map and varying t-map, by treating the A-map as another layer, and the t-map as a mask. This results in higher-quality image dehazing. The effect of estimating a uniform vs. varying airlight is exemplified in Fig. 10.

In dehazing, \( \text{Loss}_{\text{Reg}} \) forces the mask \( t(x) \) to be smooth.
Table 1: Comparison of dehazing methods on O-Haze Dataset.

|-------|---------|-----------|-------------|----------|------------|------------|----------|----------|---------|

Figure 11: Comparing Double-DIP’s dehazing to specialized dehazing methods. (many more results in the project page)

(by minimizing the norm of its Laplacian). The internal self-similarity of patches in a hazy image $I(x)$ is weaker than in its underlying haze-free image $J(x)$ [6]. This drives the first DIP to converge to a haze-free image. The $A$-map, however, is not a typical natural image. While it satisfies the strong internal self-similarity requirement, it tends to be much smoother than a natural image, and should not deviate much from a global airlight color. Hence, we apply an extra regularization loss on the airlight layer: $\|A(x) - A\|_2$, where $A$ is a single initial airlight color estimated from the hazy image $I$ using one of the standard methods (we used the method of [6]). Although the deviations from the initial airlight $A$ are subtle, they are quite crucial to the quality of the recovered haze-free image (see Fig. 10).

We evaluated our framework on the O-HAZE dataset [4] and compared it to unsupervised and self-supervised dehazing algorithms. Results are presented in Table 1. Numerically, on this dataset, we ranked second of all dehazing methods. However, visually, on images outside this dataset, our results seem to surpass all dehazing methods (see the project website). We further wanted to compare to the winning methods of NTIRE’2018 Dehazing Challenge [2], but only one of them had code available [39]. Our experiments show that while these methods obtained state-of-the-art results on the tiny test-set of the challenge (5 test images only!), they seem to severely overfit the challenge training-set. In fact, they perform very poorly on any hazy image outside this dataset (see Fig. 11 and the project website). A visual comparison to many more methods and on many more images is found in the project website.

6. CONCLUSION

“Double-DIP” is a unified framework for unsupervised layer decomposition, applicable for a wide variety of tasks. It needs no training examples other than the input image/video. Although general-purpose, in some tasks (e.g., dehazing) it achieves results comparable or even better than leading methods in the field. We believe that augmenting Double-DIP with semantic/perceptual cues, may lead to advancements also in semantic segmentation and in other high-level tasks. This is part of our future work.
References


