Reflectance estimation using an EURNet deep network architecture

Sergio Sancho Asensio

Abstract

In this work we present an efficient end-to-end deep network architecture EURNet *(Efficient U-Shaped Reflectance Network)* for estimating intrinsic image reflectance. This network consists of a fifth scale level Encoder combined with hypercolumns, and followed by a quasi-symmetric Decoder. EURNet is based on the U-Net architecture that provides a pixel labeling estimation that efficiently deals with low level features such as color and texture. The combination with early hypercolumn activations is used to provide a counter effect to the typical Encoder-Decoder network artifacts such as edge blurring. Furthermore, EURNet is improved by exploiting inception modules, in this way, the herein proposed method obtains competitive results while requiring a reduced amount of parameters, which allows the implementation of the architecture using mid-end general-purpose computing on graphics processing units (GPGPUs). Due to the difficulties of obtaining the ground truth of reflectance from natural images, the large-scale synthetic ground truth MPI Sintel dataset is used to evaluate the proposal. Performed experiments highlight the wellness of the proposed architecture, showing an efficient trade-off between the model complexity and provided accuracy when using the standard evaluation metrics.

Index Terms

Intrinsic Image Decomposition, MPI Sintel dataset, Deep learning, Encoder-Decoder, Hypercolumns, Split inception module, ELU, Huber, Adam, Keras.

I. Introduction

Nowadays, computer vision is a successful interdisciplinary field with multiple methods that are used to tackle numerous and challenging problems. This research area is continuously growing due to the high capability of the new CPU/GPUs, fostering richer and more accurate models. Computer vision deals with how computers can be programed for gaining high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do [1].

The motivation behind this work is two-fold: (1) understanding the image decomposition problem and (2) learn *Keras*: the current top-notch Python deep learning library. These are detailed in the following paragraphs.

By understanding the intrinsic image decomposition (IID) problem, we are addressing a problem as an underlying tool to enable to develop more robust algorithms when tackling complex machine vision problems. For instance, extracting the reflectance map given an input image (as depicted in Fig. 1) is an useful pre-processing step for many computer vision solutions. As an illustrative example, if a practitioner wants to track cars in a highway, an algorithm to perform foreground segmentation to capture moving objects is needed. As shadows move together with vehicles, pixels corresponding to shadows would be misclassified as vehicles. A solution to this problem would be to remove the shading component as the initial step for the foreground extraction algorithm.

Learning to work with Keras is one of the main goals of this thesis and we expect to achieve by implementing the most adequate techniques that fits the requirement of the problem, it will bring us to learn everything related to a machine learning project setup: exploratory data analysis, cross validation, data augmentation, early stopping and neural networks design. Keras simplifies the design of complex learning models while providing a front-end for current state-of-the-art backend such as Tensorflow and Theano.

Regarding IID problem, it was first defined by *Barrow and Tenenbaum* [2], they propose that any image can be decomposed as a mixture of the intrinsic scene characteristics like range, orientation, reflectance and illumination.

Intrinsic images are an intermediate representation of a picture that represents an isolated property of the image such as reflectance or shading amongst others [3]. The reflectance component captures the color of surfaces in the scene, it is a lightning-invariant quantity also referred as albedo, and it is an intrinsic property of the surface material [4]. On the other hand, the shading component captures the shade effects caused by interactions between scene illumination, camera position and surface geometry. The intrinsic image model aims at splitting an image $I \in \mathbb{R}^3$ into per-pixel product of reflectance $R \in \mathbb{R}^3$ (i.e., base color) and shading $S \in \mathbb{R}^3$ such that:

$$I(x) = R(x) \cdot S(x) \tag{1}$$

where x is a pixel coordinate.

Notice that S can be computed with the element-wise division between I and R.

Although IID offers many benefits, it is an ill-posed problem due to the fact that for each pixel there exist twice as many unknown variables (reflectance and shading) as there are measurements [5].

Due to the problem complexity, deep learning techniques are proposed to learn to estimate the reflectance component.

Author: Sergio Sancho Asensio, sergiosanchoasensio@gmail.com

Advisor: Maria Vanrell Martorell, Computer Vision, Universitat Autònoma de Barcelona

Thesis dissertation submitted: July 2017

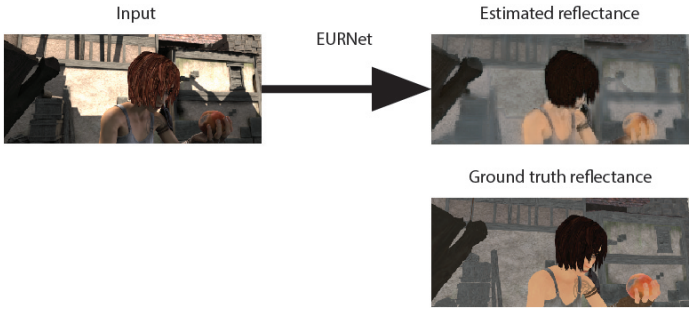


Fig. 1. EURNet estimates the reflectance component given an input image.

Therefore, by using deep learning this work presents the following hypothesis: estimating the R component from an image I can be dealt as a segmentation task due to the fact that image segments can be assumed as regions of homogeneous materials and consequently homogeneous reflectance. Segmentation is dealt in deep learning by Encoder-Decoder architectures that aims to learn a convolutional regression mapping from color image input to its corresponding reflectance output label. However, this segmentation approach can be too coarse for reflectance estimation, therefore we will need to add additional mechanisms to the architecture in order to preserve edges and textures at a finer scale level, we propose to do it by inserting hypercolumns in the architecture.

Following this method, given the input image, a new one is predicted where each pixel value corresponds to an output label. In this case, the output (the predicted pixel value) has to be a three-component vector with the estimated RGB coordinates for each object in scene as representing its reflectance surface value.

Recall that it is hard to extract the reflectance map from a real scene, thus it is difficult to obtain training data, as it is highly complex to extract the components of real-world images due to the complex lightning interactions and material properties. The MPI Sintel dataset is used [6], which provides rendered-images and the corresponding reflectance ground truth derived from the underlying 3D models and assets.

The remainder of this document, is presented as follows: first the related work is introduced, next the method is detailed, afterwards the learning procedure is discussed, following that experiments and results are described and finally, the conclusions and future work are presented.

II. Related work

In this section, the literature related to our work is reviewed. Firstly, we survey different solutions that have been designed for the IID problem, giving more attention to recent deep network architectures proposed to solve this problem. Secondly, since intrinsic estimation is a pixel-wise labeling task, standard architectures for semantic segmentation are also analysed. Finally, as the information of interest is distributed across all the levels of the CNN and should be exploited in this way, the hypercolumns technique is reviewed.

A. Intrinsic image estimation

As we already defined before, images can be decomposed into several intrinsic components, such as reflectance and shading apart from specularity, depth, shape, amongst others [7], [8]. The estimation of these intrinsic components is an ill-posed problem: for an image with M pixels, this implies to have $2M$ unknowns with only M observations. Reviewing the literature in the field, one finds (1) non-deep learning solutions, and recently, (2) deep learning solutions, which correspond to the current state-of-the-art.

The most relevant methods within the first group are listed below [9]:

- the Retinex algorithm [10], which thresholds gradients to obtain the intrinsic images
- *Barron et al.* [11], which use a shading rendering engine and learned priors on shapes and illuminations to estimate the most likely intrinsics,
- *Lee et al.* [12], which from RGB and depth video subject to additional shading and temporal constraints estimate intrinsics, and
- *Chen and Koltun* [13], which proposes a refined shading model by decomposing it into direct and indirect irradiance, and a color component.

In the second group, we have differentiated between two approaches, that is: (a) single intrinsic estimation methods, , and the most recent techniques (b) combined intrinsic estimation methods, which employ generative adversarial networks (GANs) [14]. These are detailed in what follows.

1) Single intrinsic estimation: Single intrinsic estimation methods predict the reflectance R and the shading S regardless the scale, and both are separately evaluated on scale-invariant metrics. However, there is no guarantee on energy preservation in Eqn. 1. In these models, physics-inspired priors are made, typically retinex assumptions: sparsity of R, which is generally unique per object in the scene and slow variation of S [13]. Several approaches have been explored, and the most accurate single intrinsic estimation method so far is *Direct intrinsics* [9] by *Narihira et al.*, which uses the Multi-scale CNN Regression model (MSCR) shown in Fig. 2 –motivated by the multi-scale architecture used by *Eigen and Fergus* [15].

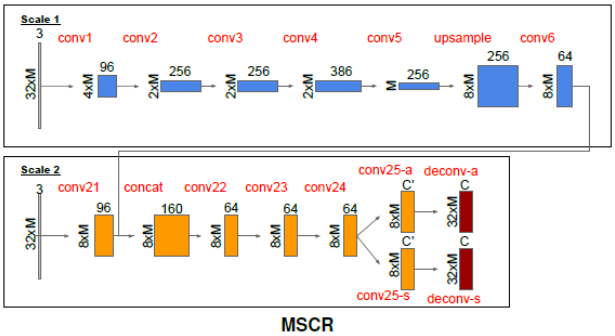


Fig. 2. MSCR implements the *Direct intrinsics* network –taken from [9].

More specifically, MSCR is a two-level feed-forward CNN architecture that analyze images at two different scales.

- Scale 1 predicts the global context. Layers from one through five takes the common *AlexNet* [16] design, then upsample the feature map to a quarter of the original image size and feeds it to a 1×1 convolutional layer with 64 feature channels.
- Scale 2 uses the output of the coarse network to predict the finer resolution result. It consists of 4 convolutional layers for feature extraction followed by reflectance and shading prediction [9].

Narihira et al. emphasize the importance of choosing an adequate loss function for solving the IID problem. They combine both the sum of the scale invariant L2 loss for R and S, expressed as

$$\mathcal{L}_{SIL2}(Y, \hat{Y}) = \frac{1}{N}\sum_{i,j,c} y_{i,j,c}^2 - \lambda \frac{1}{N^2}\left(\sum_{i,j,c} y_{i,j,c}\right)^2 \quad (2)$$

where Y is the ground truth image, $\hat{Y}$ the prediction map, $y = Y - \hat{Y}$, $\mathcal{L}$ refers to a loss function, i, j, c are the pixel coordinates, N is the number of pixels and λ is a coefficient for balancing the scale-invariant term, $\lambda = 0.5$ is used for the MPI Sintel dataset.

And the gradient L2 loss only defined on R, in order to favor recovery of piecewise constant (since S cannot be assume as piecewise constant), defined as

$$\mathcal{L}_{grad}(Y, \hat{Y}) = \frac{1}{N}\sum_{i,j,c}\left[\nabla_i y_{i,j,c}^2 + \nabla_j y_{i,j,c}^2\right] \quad (3)$$

where ∇ denotes the gradient operator.

2) Combined intrinsic estimation: Combined intrinsic estimation methods predict R and S jointly and preserve the energy following Eqn. 1, including scale.

The most accurate algorithm so far is *DARN (Deep Adversarial Residual Network)* [4]. *DARN* authors propose a fully data-driven (it does not require physical priors) deep supervised learning method for decomposing a single image into its intrinsic components. It is composed of a single end-to-end deep sequence of residual blocks and a perceptually-motivated metric formed by two discriminative networks, as shown in Fig. 3. This design consistently predicts R and S, because the dependence between them is included in the training process.

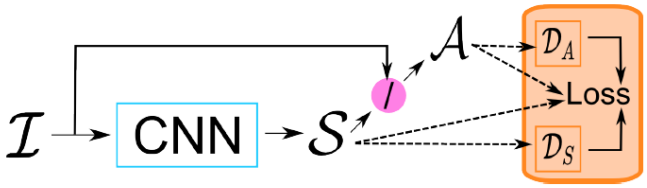
To fit this system, network parameters are trained to minimize a loss $\mathcal{L}$ composed of three terms, expressed as an addition in this way

$$\mathcal{L}(R, S) = \mathcal{L}_{data}(R, S) + \mathcal{L}_{\nabla}(R, S) + \lambda \mathcal{L}_{adv}(R, S) \quad (4)$$

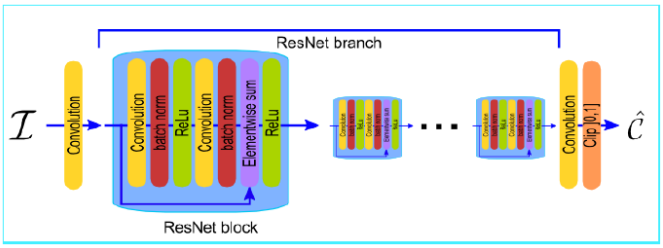
each one of these terms are described in what follows:

- *data loss*, is a measure sensitive to scale that imposes consistent global intensity, given by

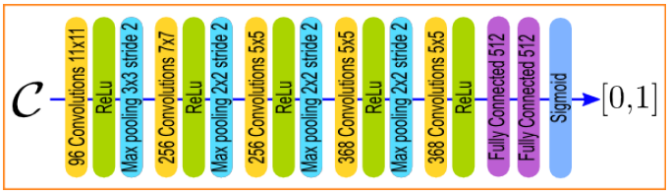
$$\mathcal{L}_{data}(R, S) = \|R_{gt} - R\|^2 + \|S_{gt} - S\|^2 \quad (5)$$



(a) *Global DARN architecture. Blue and orange boxes correspond to the generative and discriminative networks, respectively.*



(b) The generative CNN, blue box in (a).



(c) The discriminative network, orange boxes in (a).

Fig. 3. *DARN* architecture –reproduced from [4].

- *gradient loss*, favors estimations that exhibit variations consistent with both components and avoid over-smoothing, defined as

$$\mathcal{L}_{\nabla}(R, S) = \|\nabla R_{gt} - \nabla R\|^2 + \|\nabla S_{gt} - \nabla S\|^2 \quad (6)$$

- *adversarial loss,* removes visual artifacts produced by the generative CNN. An extra parameter λ is added to weight this loss, see Eqn. 4. This parameter is set to zero for initial iterations due to the poorness of the first estimations of the generator network. This loss is given by

$$\mathcal{L}_{adv}(R, S) = -\log(D_A(R) \cdot D_S(S)) \quad (7)$$

where discriminator networks D_A and D_S are trained using a binary classification loss, in this way

$$\mathcal{L}_{Discr}(C_{gt}, C) = -\log(D_C(C_{gt})) - \log(1 - D_C(C)) \quad (8)$$

where C represent either R or S.

This architecture shows very competitive results in MSE, LMSE, and DSSIM metrics[1].

[1]MSE, LMSE and DSSIM metrics are discussed in chapter V.

B. Segmentation architectures

The main works dealing with deep neural networks for the segmentation problem have been compiled in this section. Nowadays, SegNet is the state-of-the-art of pixel-wise semantic segmentation [17] and U-Net is the current state-of-the-art network for biomedical image processing [18], being the latter widely used in *Kaggle* competitions [19]. We review both architectures in what follows.

1) SegNet: Is a neural network architecture for semantic pixel-wise segmentation [17]. It consist of an Encoder-Decoder architecture followed by a soft-max classifier for pixel-wise classification as shown in Fig. 4. The Encoder is identical to the first 13 layers of VGG16 [20] and the Decoder maps the low resolution encoded feature maps to full input resolution feature maps.

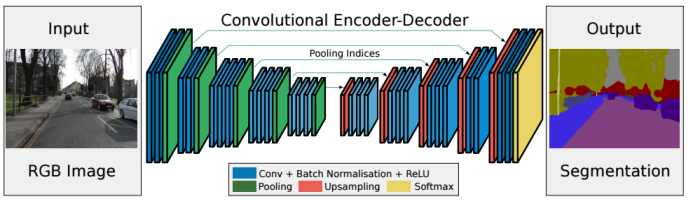


Fig. 4. SegNet architecture –extracted from [17].

- *Encoder,* consist of a successive application of a dense convolution, a rectified linear unit (ReLU) activation, a max-pooling with a 2×2 non-overlapping window to achieve translation invariance over small spatial shifts, and finally a downsampling operation.
- *Decoder,* upsamples its input feature maps using the memorized max-pooling indices from the corresponding Encoder feature maps. Afterwards, these feature maps are convolved with a Decoder filter bank to produce dense feature maps. The final layer is a soft-max classifier which outputs a K channel image, where K is the number of classes.

SegNet was motivated by scene understanding applications. It assigns a label to each pixel in a scene. Each label corresponds to an object class. It performs specially well in segmentation problems in which objects are well-defined –i.e. segmentation in road scenes.

2) U-Net: Is a neural network architecture composed of a contracting path which is an Encoder and a quasi-symmetric expanding path, which is a Decoder. The contracting path follows the typical architecture of a convolutional network capturing context, and expanding path combines feature maps from the contracting path with the upsampled feature maps of the previous scale level enabling precise localization –see Fig 5. Therefore, the main difference between U-Net and SegNet is that U-Net does not reuse pooling indices but instead transfers the entire feature map to the corresponding Decoders and concatenates them to the corresponding upsampled Decoder feature maps [17]. Hereby, U-Net design captures little texture variations such as roughness, surface granularity and gradual color variations due to scene geometry at a higher computational cost. For this reason, U-Net becomes an interesting option for the IID problem.

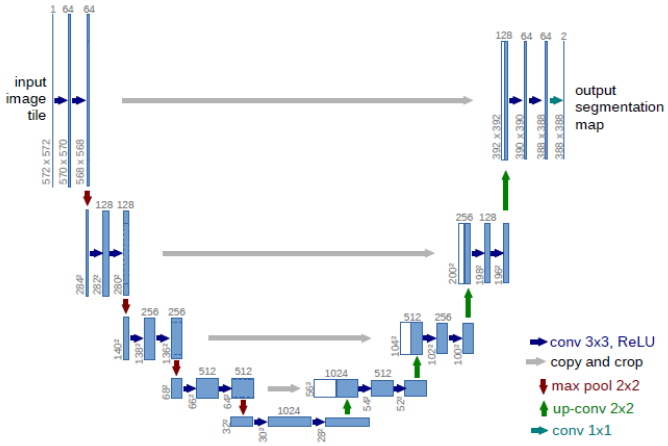


Fig. 5. U-net architecture –taken from [18]. Contracting-expanding path are depicted on the left and right side respectively. The x-y size and the number of channels are on the left and on the top of the convolutions respectively. Blue boxes are multi-channel feature maps and white boxes are copied feature maps.

The union of both contracting path and expanding path yields an U-shaped architecture that names the network.

- *Contracting path,* composed of five scale-levels. While early scale-levels capture universal features such as curves and edges, the highest captures context. Levels one to four consist of two 3×3 convolutions, each one followed by a ReLU. Then it follows a 2×2 max pooling operation with stride 2 for downsampling. At each downsampling step the number of feature channels are doubled. Level 5 is analogous to previous levels, consist of two 3×3 convolutions with 512 feature channels, each followed by a ReLU.
- *Expanding path,* composed of five scale-levels. Allow the network to propagate context information to higher resolution layers. Each level consist of a concatenation of the cropped current contracting path scale level with the previous scale level upsampled by 2×2 (i.e. to compute the fourth scale level of the expanding path, the feature maps from the fourth scale level of the contracting path are cropped and concatenated with the upsampled fifth scale level feature maps, cropping operation is required due to the loss of pixels in every convolution). Finally, two 3×3 convolutions, each one followed by a ReLU, are applied. The highest level is followed by an 1×1 convolution used to map each 64-component feature vector to the desired number of classes.

C. Boosting prediction accuracy with hypercolumns

Encoder-Decoder architectures can be too coarse in representing spatial information due to they can lose precision in localization. In fact, earlier convolutional neural networks

(CNN) layers may be precise in localization but do not capture semantics, on the other side higher layers may contain rich semantic information but be poor in spatial. The usage of hypercolumns (HC) allows practitioners to get the best of both worlds [21].

An HC at a pixel is the vector of activations of all units in the CNN for that pixel, as depicted in Fig. 6.

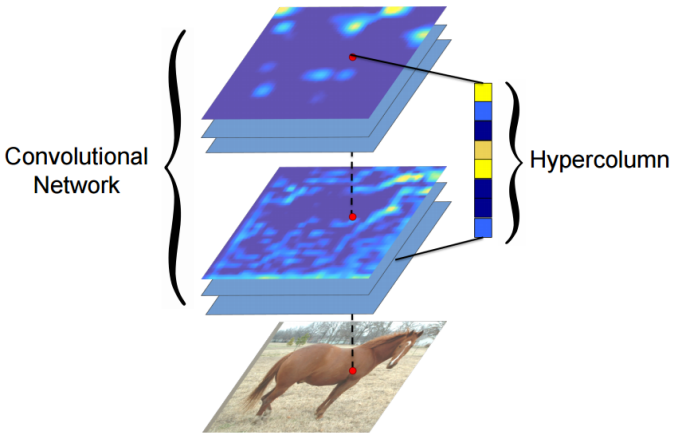


Fig. 6. The hypercolumn representation –extracted from [21].

Fig. 7 show different HC activations. While the first HC activations resemble edge detectors, the last HC activations are abstract and semantically interesting but the spatial information is fuzzy.

In the literature regarding the IID problem, *Direct intrinsics* is the only method that adopted the HC terminology. It added HC on the first scale of MSCR by directly connecting intermediate layers to the output –this increased the overall architecture accuracy. The proposed EURNet architecture that we will present in the next section uses early HC activations to boost localization.

III. Method

In the previous section we introduced the state-of-the-art in IID learning machines, semantic segmentation architectures and hypercolumns. The purpose of this section is to expose our hypothesis and discuss the EURNet architecture that will combine U-Net and hypercolumns for the reflectance estimation problem.

A. U-Net and hypercolumns for pixel labeling

The goal of the project is, given an input image I, extract the reflectance R component without requiring more data such as shading or depth. Recall that it is difficult to obtain a dataset with a precise reflectance ground truth and a precise shading ground truth.

In order to solve the reflectance estimation problem, our hypothesis is combining U-Net and HC allows to reach the following effects.

- The goal of the U-Net, is that the contracting-expanding path (Encoder-Decoder) join pixels with a similar reflectance to a connected-labeled region, joining shading effects or minor texture differences creating blur over edges and halos.
- The goal of earlier hypercolumn activations is helping to preserve edges that persist through multiple scales. This provokes a counter effect to the blurring of the U-Net in the IID problem, this is a sharpening and anti-halo effect.

The U-Net architecture shown in section II-B2 has been adapted to avoid the loss of resolution after each convolution. Consequently, cropping operations from original U-Net are discarded. Notice that this is implemented by using *border='same'* property in *Keras* [22]. We call this modified architecture as full-size U-Net. This full-size U-Net has been set in such a way that the output segmentation map size is the same as the input image size.

Fig. 8 illustrates the results of applying the (a) full-size U-Net to the MPI Sintel Dataset: results are contaminated with artifacts such as blur, black halos and therefore edges are not conserved, (b) combination of the contracting path with HC: resulting images highlight edges and contours and finally, (c) combining both full-size U-Net and HC with weight $\eta = 0.2$ [2]: artifacts are diminished significantly and consequently, edges are preserved and halos have been vanished.

Due hardware constraints, and as the full-size U-Net architecture is memory-intensive, an inception mechanism [23] is introduced. Inception blocks are used in both Encoder and Decoder to reduce the overall network complexity in terms of parameters. This way, the training stage is much faster, being less prone to overfitting and requiring much less memory.

B. Reducing the number of parameters

In this section, inception modules are introduced and an improvement of these, referred to as split inception modules.

1) Inception modules: The most straightforward way of improving the performance of deep neural networks is by increasing their size. This includes both increasing the depth - the number of network levels - as well as its width: the number of units at each level [23]. Bigger size typically means a larger number of parameters. This rationale has two fundamental drawbacks: the model is more complex and therefore is more prone to overfit (the model begins to "memorize" training data rather than "learning" to generalize), and the computational complexity is rocketed. A way to solve both problems, is to introduce sparsity and replace the fully connected layers by the sparser ones.

Inception architecture introduce sparsity [23]. This architecture is based on two main ideas: the approximation of a sparse structure with spatially repeated dense components and using dimension reduction to keep the computational complexity in bounds, but only when required. Thus, inception modules permit to increase the depth and width of the network while keeping the computational budget constant. They act as multiple convolution filter inputs that are processed on the same input. It also does pooling at the same time. All the

[2]The EURNet weighting mechanism is discussed in section IV-E2.

(a) Input image. (b) HC average for layers 1-3. (c) HC average for layers 1-5.

Fig. 7. Hypercolumns from the Encoder of the U-shaped architecture.

(a) Full-size U-Net.

(b) Encoder and early HC.

(c) Full-size U-Net and HC.

(d) Ground truth.

Fig. 8. Scene-split experiment with different configurations.

results are then concatenated, see Fig. 9. This allows the model to take advantage of multi-level feature extraction from each input. For instance, it extracts general 5×5 and local 1×1 features at the same time.

2) Split inception modules: Following the recommendations given in [24], a customization of this architecture has been implemented in order to obtain a better performance for the IID problem without increasing the amount of parameters –see Fig. 10. This customization is two-fold: (1) the 3×3 convolution has been split into a: 3×1 convolution, followed by a ReLU, batch normalization and a 1×3 convolution. Analogously, (2) the 5×5 convolution has been split into a: 5×1 convolution, followed by an ReLU, batch normalization

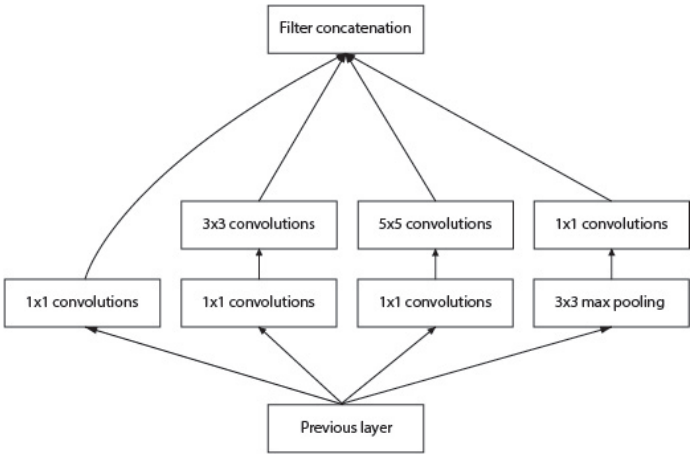


Fig. 9. Inception module with dimensionality reduction [23].

and a 1×5 convolution. This new architecture is named as split inception module. Batch normalization allows to use much higher learning rates and to be less careful about initialization, it also acts as a regularizer [25]. The presented split inception module is detailed in the following:

Given an input layer with d feature channels, it performs four branches of operations:

- Branch 1, 1×1 convolutions with $d/4$ feature channels.
- Branch 2, 1×1 convolutions to compute reductions with $d/8$ feature channels followed by 1×3 convolutions with $d/2$ feature channels, ReLU, batch normalization and 3×1 convolution with $d/2$ feature channels.
- Branch 3, analogous to before, 1×1 convolutions to compute reductions with $d/16$ feature channels followed by 1×5 convolutions with $d/8$ feature channels, ReLU, batch and 5×1 convolutions with $d/8$ feature channels are stacked.
- Branch 4, 3×3 max pooling layer with a pool size of 3×3 and a stride of 1×1 to diminish the resolution of the grid. Then 1×1 convolutions with $d/8$ feature channels are stacked.

The output is the concatenation of the previous four branches. Then a batch normalization operation and a ReLU are applied.

C. EURNet method for reflectance estimation

EURNet design (shown in Fig. 11), is based on the full-size U-Net architecture and its main characteristics are: (1)

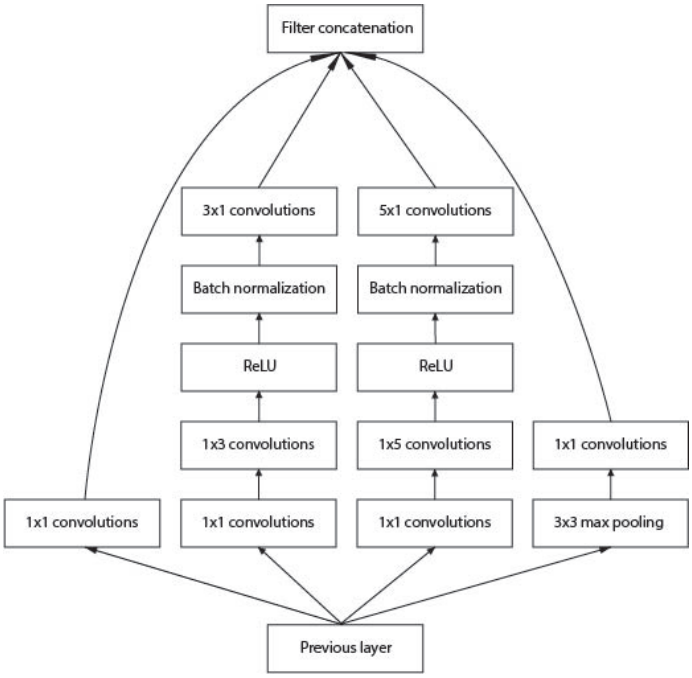


Fig. 10. Split inception module with dimensionality reduction.

it replaces each pair of convolutions with a split inception block. TABLE I shows a big improvement of using inception modules, they drastically reduce the number of parameters. Hereby, at the same computational cost practitioners can train the model with higher resolution images and higher batch size, (2) performs the downsampling operation using a 3×3 convolution with stride 2 –notice that U-Net and SegNet methods use a max-pooling operation instead. EURNet does not use max-pooling in downsampling operations to avoid propagating whitish regions through the Decoder network – recall that max-pooling operations always take the higher intensity pixels in regions–, and (3) adopts the HC terminology following the design detailed in [26]. More precisely, early HC activations corresponding to the first, second and third scale level of the Encoder. As these contain spatial information, they act as a sharpening effect. Entering more in detail:

- Encoder, consist of a successive application of an inception block, followed by a 3×3 convolution with a downsampling operation. At each downsampling step the number of feature channels are doubled, starting from 32 before applying the downsampling on the first scale level, to 256 before applying downsampling at the fourth scale level –the number of feature channels at each downsampling step are doubled because as the spatial dimensions are becoming smaller, given a constant computational budget, the number of channels in subsequent layers can be increased without increasing the computational load. Each downscaling operation is followed by a dropout of 50% –dropout prevents overfitting by randomly disconnecting inputs when training the model [27].
- Hypercolumns, obtained by concatenating the feature maps from the first scale level of the Encoder with the 2×2 upsampled feature maps of the second scale level and the 4×4 upsampled feature maps from the third scale level into a single volume. This volume has high depth while keeping the original input image width and height, so depth has to be reduced to three. To do so, an inception module with 128 feature channels, a dropout of 50%, an inception module with with 64 feature channels, another dropout of 50% and finally, a convolution that reduces the number of feature channels to three are applied.
- Decoder, the Encoder of the current scale level is concatenated with the Encoder at the previous scale level upsampled by 2×2, then an inception block and a dropout of 50% are stacked. The highest scale level is followed by an 1×1 convolution that reduces the number of feature channels to three.

Method	**#Params**
Full-size U-Net (without inception blocks)	7.8M
Full-size U-Net + HC (without inception blocks)	8.2M
Full-size U-Net with inception blocks	**1.8M**
Full-size U-Net + HC with inception blocks	1.9M

TABLE I
COMPARISON BETWEEN DIFFERENT FULL-SIZE U-NET CONFIGURATIONS IN NUMBER OF PARAMETERS, LOWER IS BETTER.

D. Activation functions

A review of the distinct activation functions has been done.

1) Hidden layers: Recent work has demonstrated that Exponential Linear Units (ELU) activation functions obtain higher classification accuracy than ReLUs [28]. ELU, which is shown in Fig. 12 and Eqn. 9, is monotonic if $\gamma \geq 0$. The derivative is shown in Eqn. 10 and it is monotonic if $0 \leq \gamma \leq 1$. It approximates the identity near the origin if $\gamma = 1$ and has range $(-\gamma, \infty)$. This activation function tries to set the mean activations closer to zero, speeding up learning and it also avoids the vanishing gradient issue via the identity for positive values. Therefore, ELU can obviate the use of batch normalization.

Section V-B1 shows that in the MPI Sintel Dataset, there is an improvement in terms of performance in the evaluated metrics by replacing ReLU (or ReLU+batch normalization) with ELU using $\gamma = 1$.

$$f(x) = \begin{cases} \gamma(e^x - 1) & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{9}$$

$$f'(x) = \begin{cases} f(x) + \gamma & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{10}$$

where f is the ELU function, f' its derivative and γ an hyperparameter that controls the value to which an ELU saturates for negative inputs.

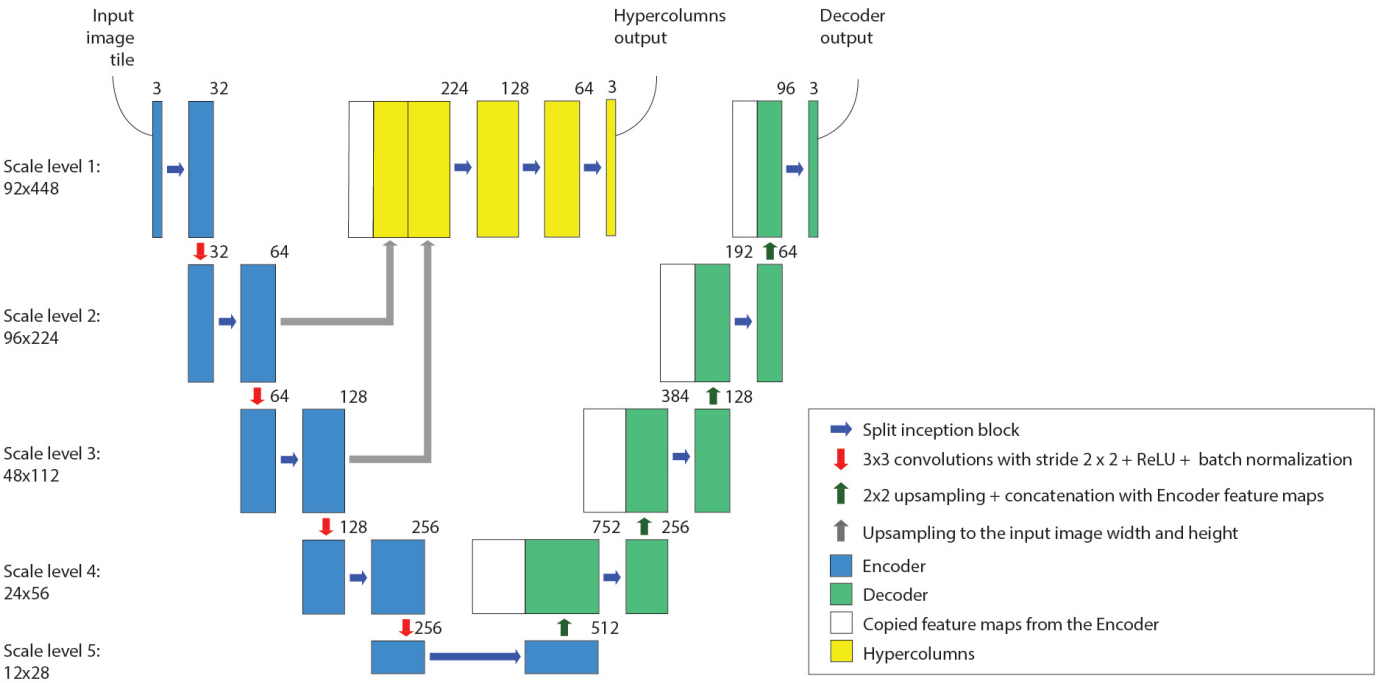


Fig. 11. Proposed EURNet architecture. The number of channels is depicted on the top of each box.

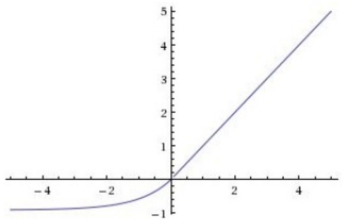


Fig. 12. ELU activation function.

2) Output layers: Output layers transform previous feature maps into an RGB image. Two distinct activation functions have been tested in both hypercolumns output and Decoder output.

- Sigmoid, this function bound results in the $(0, 1)$ range. Analyzing predictions, issues were found as depicted in Fig. 13. In fact, sigmoid suffers from the vanishing gradient problem. It squashes their input into a very small output range in a non-linear manner. As a result, there are large regions of the input space which are mapped to an extremely small range. In these regions of the input space, even a large change in the input will produce a small change in the output [29].
- Linear output, this activation function does not suffer from the inconveniences of the sigmoid in this problem. An extra operation that clip results in the $(0, 1)$ range is required.

IV. Learning stage

In this section, the learning stage is discussed. This involves the loss function for fitting the network, the optimizer, the dataset, the data augmentation procedure, and implementation details.

A. Loss function

In order to find the optimal output, a precise way of measuring the quality of a solution is needed. This is done via the objective function. This function, taking data and model parameters as arguments, can be evaluated and the performance of the model is, therefore, measured. The goal is to find values for the model parameters which minimize the loss. In the case of the IID problem, if the objective function is ill-suited to the problem, predicted output images will be of poor quality, that is: blurred and filled with artifacts.

Huber loss function is therefore proposed. Notice that it is also known as smooth L1 loss function. The Huber loss, defined as

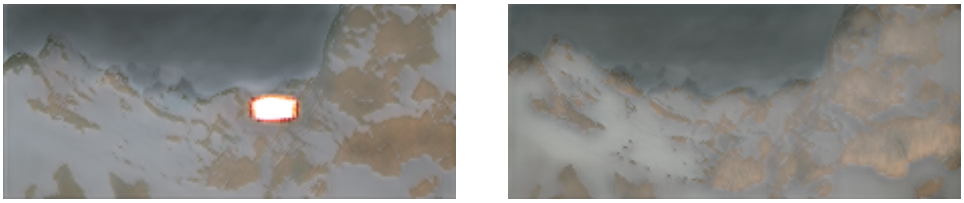
$$\mathcal{L}_\delta(Y, \hat{Y}) = \begin{cases} \frac{1}{2}(Y - \hat{Y})^2 & \text{for } |Y - \hat{Y}| \leq \delta \\ \delta|Y - \hat{Y}| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \quad (11)$$

where δ is set to 0.1 by default.

Is used in robust regression, and it is less sensitive to outliers in data than squared error loss.

B. Optimizer

Ideally, the objective function must be minimized until convergence –that is: until a global minimum is found in a convex problem. This is performed with a gradient descent technique: the direction of the slope of the surface created by the objective function downhill is followed until a valley is reached.



(a) (b)

Fig. 13. Scene-split experiment using (a) sigmoid and (b) linear output. The effect of saturated sigmoid units are clearly visible as a white artifact.

As a robust gradient descent technique is required, Adam fits the proposal [30]. Adam is a stochastic gradient descent technique widely used by the deep learning community. Its main advantages are that it is efficient, works well with sparse gradients and successfully deals with non-stationary objectives. The algorithm is based on estimation of first order moment (the gradient mean) and second order moment (element-wise squared gradient).

In what follows the Adam update rule is described:

- compute gradient g_t at current time t,
- update biased first moment estimate,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{12}$$

- update biased second raw moment estimate,

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{13}$$

- compute bias-corrected first moment estimate,

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{14}$$

- compute bias-corrected second raw moment estimate,

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{15}$$

- update parameters,

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + e} \tag{16}$$

where e is a small number used to prevent division by 0, β_1 and β_2 are the forgetting factors for gradients and second moments of gradients, respectively. A learning rate = $2 \cdot 10^{-4}$, a decay = 0.0 and the default parameters provided in [22]: $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $e = 10^{-8}$ are used.

C. Dataset

Normally, the only information one has when taking a picture from a real-world scene are pixel intensities. A pixel intensity of a picture is formed by a combination of the color of an object, complex lightning interactions, material properties and so on. Since practitioners only have pixel intensities, it is almost unfeasible to extract reflectance and shading from real-world pictures, thus it is difficult to obtain training data. Hence, the MPI Sintel dataset [6] becomes handy, fostering the research on the topic. Despite its drawbacks (i.e., it contains mostly brown-colored scenes, surrealistic effects like fluorescence and so on), it is the best alternative at hand due to its precise ground truth [4]. Sintel is an open-source short computer animation movie that contains indoor and outdoor scenes, it has been published in various formats, among which its reflectance layers, –the latter have been rendered without illumination.

It contains 18 sequences, 17 of which are composed of 50 frames each and one of 40 for a total of 890. However, a number of scenes from the dataset could not be used due to software issues that resulted in defects in the provided ground truth reflectance maps [13]. This reduces the number of sequences to 15. These are: are “alley 1”, “alley 2”, “bamboo 1”, “bamboo 2”, “bandage 1”, “bandage 2”, “market 2”, “market 5”, “market 6”, “mountain 1”, “shaman 2”, “sleeping 1”, “sleeping 2”, “temple 2”, and “temple 3”. The resolution of each image is 1024×436 pixels.

For training the model, we follow [13] and use the “clean pass” images instead of “final pass” images; the latter adds computer graphics effects such as fog, depth of field and motion blur which may distract the model from our application, and it does not respect Eqn. 1.

As MPI Sintel dataset does not contain a large amount of images, data augmentation is required to avoid overfitting. In the following, the data augmentation procedure is detailed.

D. Data augmentation

Due to the small dataset size, a real-time data augmentation has been implemented in order to generate more samples to improve the generalization power of the learning machine. This data augmentation is similar to the one proposed by [4]: consists of randomly cropping patches within the images after scaling by a random factor in $[0.8, 1.2]$, rotating by a random angle of maximum 15°, and using a random horizontal mirroring with a likelihood of 0.5.

E. Implementation details

In this section, the model fitting settings, the weighting mechanism utilized for combining both U-Net and HC and the hardware used for the experimentation are discussed.

1) Model fitting: Due to hardware constraints, the resolution of training images is reduced to 192×448 pixels and the batch size is set to 4 instances.

Training, validation and test data are used. Validation set is used for parameter selection and to avoid overfitting. The

training vector $< X_t, y_t >$ consist of the standard procedure of randomly getting the 80% of the training data $< X, y >$ vector and validation vector $< X_v, y_v >$ the remaining 20% [31].

To speed up the training stage, initial weights of layers are initialized with the recommended He normal methodology [32] –it draws samples from a truncated normal distribution centered on 0 with the standard deviation given by

$$std(u) = \sqrt{\frac{2}{u}} \tag{17}$$

where u is the number of input units in the weight tensor.

2) EURNet weighting mechanism: So far, to improve localization and to minimize artifacts, the EURNet network combines both Decoder losses E and hypercolumn losses H in a weighted manner, this weighting mechanism is provided by

$$\mathcal{L}(\theta, \mathcal{D}) = (1 - \eta)E(\theta, \mathcal{D}) + \eta H(\theta, \mathcal{D}) \tag{18}$$

where $\mathcal{D}$ is the dataset, and η the weighting factor.

The effects of the weighting are the following: recall that the herein proposed method uses earlier HC activations, more precisely the ones found in the first, second and third scale levels of the Encoder, higher η means that the algorithm is more focused on spatial information such as edges. On the other hand, lower η means the network is more focused on semantics, obtaining images contaminated with artifacts such as blur and halos. To preserve the scene structure while having color fidelity, the optimal weighting found experimentally taking into account the characteristics of the dataset is $\eta = 0.2$.

It is important to highlight that the best validation loss is kept: at each epoch, the loss is computed. If the current loss is better than any of the previously computed losses, weights are saved to disk. Moreover, the number of epochs is set to 10.000, but an early stopping technique is adopted [33]: if after 20 epochs the loss does not improve, the training process is stopped and the weights corresponding to the best loss at the moment are stored. Thus, the weights associated to the epoch with the best loss can be loaded at any time. Also, this strategy helps to reduce the overfit of the model.

3) Hardware used: In order to develop and deploy the proposed system, the following hardware has been used:

- CPU: AMD FX-8320 3.5Ghz
- GPU: GeForce GTX 1070
- RAM: 64GB DDR3 1600 MHz
- HDD: TL100 SSD 240 GB

Debian Linux 8.1 has been used as base operating system. With the aforesaid configuration, the training time per epoch is of 200s. Also, the model has been fit in a total of 4-5 days for training in both image split and scene split experiments. The following section details these.

V. Experiments and results

In the present section, experiments and metrics to evaluate the model performance the obtained results are detailed.

A. Empirical evaluation

Two different experiments are performed. These are: scene split and image split. Scene split was proposed in [9] and the idea is the following: half of the sequences (a sequence is a set of images that jointly construct a movie scene) are used for training and the remaining for testing. Image split is less stringent and was proposed in [13]: half of the images are used for training, and the remaining for testing.

For comparison to prior work in the IID problem, the same metrics are used. These are two data-related metrics: mean-squared error (MSE), local mean-squared error (LMSE), and a perceptually motivated one: structural dissimilarity index (DSSIM).

- Mean-squared error (MSE). Measures the average of the squares of the errors between the estimated reflectance and their ground truth, the expression is

$$MSE(Y, \hat{Y}) = \frac{||Y_i - \alpha\hat{Y}_i||^2}{N} \tag{19}$$

  where $\alpha = argmin_\alpha ||Y_i - \alpha\hat{Y}_i||^2$.
  The absolute brightness of the estimation is adjusted to minimize the error.
- Local mean-squared error (LMSE). Measures which is the average of the scale-invariant MSE computed on overlapping square windows of size 10% of the image along its larger dimension. The overlap between neighboring windows is 50%.
- Structural dissimilarity index (DSSIM). SSIM characterizes image similarity as perceived by human observers. This measure accounts for multiple independent structural and luminance differences, and it is defined by

$$SSIM(Y, \hat{Y}) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{20}$$

  where μ_x is the average of $\hat{Y}$, μ_y the average of Y, σ_x^2 the variance of $\hat{Y}$, σ_y^2 the variance of Y, σ_{xy} the covariance of $\hat{Y}$ and Y. Variables c_1 and c_2 are to stabilize the division with weak denominator, depend on the dynamic range of the pixel-values.
  DSSIM is a distance metric derived from structural similarity index (SSIM), denoted by

$$DSSIM(Y, \hat{Y}) = \frac{1 - SSIM(Y, \hat{Y})}{2} \tag{21}$$

B. Results

So far, the methodology of experimentation has been detailed. The current section exposes the qualitative and quantitative results of evaluating (1) the different proposals between them in the scene split experiment, and (2) EURNet with the literature and the current state-of-the-art in intrinsic

image methodologies [9] in both image split and scene split experiments.

1) Comparison amongst the distinct analyzed methodologies: Results of all distinct configurations are shown in TABLE II.

As a first approach, the full-size U-Net is applied to the scene split problem. This model is too coarse spatially and consequently, produces blurry estimations. Replacing ReLUs to ELUs, MSE to Huber and SGD to Adam, increases the score in the evaluation metrics. However, predictions are still contaminated with artifacts. These are minimized by adding early HC activations to the network architecture, as they force the model to learn edges that preserves through the different scales. However, due hardware constraints and that the architecture is costly in computational terms, HC activations cannot be added directly to the full-size U-Net model with the same learning settings.

Therefore, the full-size U-Net is implemented with inception blocks. This allows its implementation on GPGPUs with a limited amount of memory. Consequently, the number of parameters is drastically reduced while obtaining similar scores in the evaluation metrics. Further, split inception blocks improve perceptual results obtained with vanilla inception blocks in the MPI Sintel Dataset while having the same computational complexity by tackling the overfitting problem.

Thanks to the inception blocks, HC can be added to the architecture. EURNet, by combining the effects of Encoder-Decoder and HC, outperforms all other configurations in both data-driven scores and the perceptual score while maintaining a low number of parameters –recall that earlier HC activations act as a sharpening effect. In this regard, experiments in the MPI Sintel Dataset confirm the hypothesis –qualitative results of EURNet are shown in Fig. 14.

Method	**#Params**	**MSE**	**LMSE**	**DSSIM**
Full-size U-Net				
with ReLU	7.8M	2.16	1.73	23.98
with ELU	7.8M	2.02	1.51	23.07
Using ELU				
with MSE and SGD	7.8M	2.02	1.51	23.07
with Huber and Adam	7.8M	1.89	1.27	21.91
Using Huber and Adam				
with inception modules	**1.8M**	1.92	1.33	22.09
with split inception modules	**1.8M**	1.94	1.35	21.92
Using split inception modules				
with HC (EURNet)	1.9M	**1.86**	**1.23**	**20.83**

TABLE II
QUANTITATIVE RESULTS OF THE DISTINCT ANALYZED METHODOLOGIES IN THE SCENE SPLIT EXPERIMENT (×100), LOWER SCORE IS BETTER.

2) Comparison with literature approaches: EURNet is benchmarked with the literature methods for intrinsic image estimation –these were previously exposed in section II-A. From all of these only EURNet, *Direct intrinsics* and *DARN* are using a discriminative deep learning architecture and excluding "depth images". Notice that EURNet also excludes "shading images".

Method	**MSE**	**LMSE**	**DSSIM**
Baseline: reflectance constant	3.69	2.40	22.80
Retinex [10]	6.06	3.66	22.70
Lee et al. [12]	4.63	2.24	19.90
Barron et al. [11]	4.20	2.98	21.00
Chen and Koltun [13]	3.07	1.85	19.60
Direct intrinsics [9]	1.00	0.83	20.14
DARN [4]	1.30	**0.61**	**14.43**
EURNet	**0.98**	0.81	18.39

TABLE III
QUANTITATIVE RESULTS IN THE IMAGE SPLIT EXPERIMENT (×100), LOWER SCORE IS BETTER.

Method	**MSE**	**LMSE**	**DSSIM**
Direct intrinsics [9]	2.01	1.31	20.73
DARN [4]	**1.77**	**0.98**	**14.21**
EURNet	1.86	1.23	20.83

TABLE IV
QUANTITATIVE RESULTS IN THE SCENE SPLIT EXPERIMENT (×100), LOWER SCORE IS BETTER.

The quantitative results between the methods detailed on section II-A, (including the baseline, which is a trivial decomposition where the reflectance is assumed uniform grey) and EURNet are shown in TABLE III and IV.

In terms of score, the top three methods in the benchmark are: the *DARN*, followed by EURNet and *Direct intrinsics*.

By adopting the combined intrinsic estimation methodology, *DARN* obtains the better out-of-sample generalization performance yielding the best score overall in the MPI Sintel Dataset. Recall that single intrinsic estimation techniques are affected by the inherent ambiguity on scale, and influenced by unknown exposure at capture time [4]: there is no guarantee on energy preservation as they predict two quantities R and S regardless the scale. *DARN* proposes a powerful architecture that respects Eqn. 1; this leads to consistent results.

EURNet has the advantages of (1) being computationally less expensive than *DARN* and *Direct intrinsics* [34] while providing competitive results and (2) requiring only reflectance training data.

Fig. 15 shows the qualitative results. In these, EURNet obtains brownish estimations due the characteristics of the dataset, and focus its attention to connected regions close to the camera –i.e., in the characters. Therefore, it learned to eliminate the shading component from these objects.

VI. CONCLUSIONS AND FUTURE WORK

In this chapter, results provided along the thesis are summarized and concluded. Finally, the forthcoming research that will be made as a consequence of the insights and results obtained is discussed.

A. Summary

The human visual system has a good ability to factorize the jumbled mess of confounds that is our visual world into simpler underlying factors. Even from a single image, one can tease apart effects of the surface reflectance vs. scene illumination [35]. In the Computer Vision field, this problem

Fig. 14. Qualitative results of EURNet on the image split experiment. Two examples of the obtained results are shown horizontally, where the prediction comes first and the ground truth is preceded.

is usually referred to as the intrinsic image decomposition: decomposing an image into its intrinsic properties reflectance and shading. Recall that having reflectance, shading can be predicted by performing an element-wise division between the input image and reflectance. While this problem has been studied extensively, it is still challenging [36]. Thus, it requires a high performance visual machine vision technique.

Hereby, the reflectance estimation is tackled with a deep learning approach. More precisely, the hypothesis is to solve the intrinsic estimation by combining the effects of an Encoder-Decoder architecture predictions which contains minimal texture differences and artifacts such as blur and fine-tune it with hypercolumn early activations as these act as a counter-attack to the artifacts, similar to an edge detector and providing a sharpening effect to predictions.

The proposed method is based in U-Net which is the current state-of-the-art in biomedical imaging [18], this architecture fits with the IID problem as it is designed to capture little variations in context such as gradual texture variations due to object geometry. This architecture is formed by a Encoder that

Fig. 15. Qualitative results on the image split experiment. Comparison between *Lee et al.* [12], *Chen & Koltun* [13], *Direct intrinsics* [9], *DARN* [4] and EURNet.

captures context and a Decoder that enables precise localization. As the U-shaped architecture is memory-intensive, it is implemented using inception modules to reduce the overall network complexity. Moreover, it is improved by using split inception blocks which avoid overfitting while having the same computational cost.

The model is trained using the well-known MPI Sintel Dataset. As there are a low amount of images, a data augmentation technique is implemented.

Following the experimentation proposed by *Narihira et al.* [9], two experiments are executed: image split and scene split, and three metrics are used: MSE, LMSE and DSSIM.

Results reflect that the proposed architecture is really efficient: it achieves state-of-the-art results while requiring a low number of parameters and only reflectance training data. Therefore, the proposed hypothesis is verified in the MPI Sintel Dataset.

B. Conclusions

The main goal of the thesis has been to propose an architecture to estimate the reflectance of an image. This difficult problem has been successfully tackled by our proposed EURNet: a deep neural network that obtains state-of-the-art results while requiring a moderate number of parameters. The conclusions have been organized in two different sets: (1) personal conclusions, and (2) scientific conclusions.

As personal conclusions we have compiled the skills I think have been achieved while doing this project, they are described below:

- Understanding the image decomposition problem. Learning reflectance estimation is a tough problem even using the most advanced techniques (CNNs). However, this task has a huge potential in industry because its possible application in computer vision and computer graphics – i.e., in a texture image generator for videogames.
- Learning the neural networks API Keras. Thanks to this API, complex architectures can be easily implemented and deployed, hugely decreasing the amount of time required to implement the network. Therefore, we strongly believe that this API will become an industrial standard. The most complex part in terms of code was to implement the architecture with inception modules due to the fact that it resulted difficult for us to debug, as Keras error messages are cryptic.
  Finally, it has been feasible to update the Keras library to the requisites of the problem, for example updating the image data augmentation object. This way, any practitioner can easily customize the library to its needs.

Our main conclusions about the scientific problem of intrinsic image estimation have also been grouped in two sets. About the problem of Intrinsic image decomposition we can list the following conclusions:

- Decomposing an image into its intrinsic properties is a challenging problem and part of the difficulty lies in the fact that the problem is ill-posed: a single input image can be explained by a continuum of reflectance and illumination combinations [13].
- The difficulty becomes worse because training data is scarce; especially coming from pictures taken from the real world. Here, a synthetic dataset is used, more precisely the MPI Sintel Dataset. But, the use of this dataset is overfitted by the CNN architecture and more effort should be addressed to create new datasets.
- Results shown in section V-B, demonstrates that deep learning approaches outperform previous methods in the evaluated dataset. Deep learning techniques automatically learn problem-specific features while traditional architectures have been heavy reliant on hand-crafted features and priors [37].
- Regarding our proposal, it shows promising results (nearly as good as the current state-of-the-art techniques) taking into account the hardware limitations and that our proposal is trained only with reflectance ground truth, discarding shading and depth.

About deep neural networks, we have concluded that deep learning is not a substitute for understanding the problem in depth. Instead of investing time constructing a complex deep neural network architecture, practitioners should always invest time studying, analyzing and dissecting the data first, then choose a technique with stronger theoretical foundations. In regard to the experience gained by working with deep neural networks:

- It is important to be constantly checking the obtained predictions to detect problems. Some experiences are listed below:
  - In a first approach, sigmoid was used as the output activation function. This produced images with burnt areas such as white regions which drastically diminished the network performance in the evaluated metrics. Replacing sigmoids by linear activation functions fixed the issue, as shown in section III-D2.
  - The full-size U-Net produced predictions with artifacts, which resulted in blurry images and images which did not preserve edges. This was solved by adding early HC activations as they acted as an edge-detector with sharpening effects.
- Deep learning architectures require a huge amount of data to generalize, the proposed architecture did not converge until the real-time data augmentation technique was implemented.
- Using inception modules reduces drastically the number of parameters: it became feasible to train the model from scratch with a GPU with a limited amount of VRAM memory. With the full-size U-Net architecture it was impossible to train the model with HC and the same learning settings.
- Split and conquer improves the network. Instead of applying a NxN convolution, splitting a NxN convolution into a: $Nx1$ convolution, followed by an activation function, batch normalization and a $1xN$ convolution accelerated the training stage. This is how the split inception architecture is constructed, as detailed in section III-B2.

We want to highlight that although and extensive research has been done during this thesis, there is still plenty of work to do.

C. Future lines

Further research additions can be studied to improve the results of EURNet. These are listed below.

- Explore EURNet visualization mechanisms for readability enhancement. In many domains, it is important to provide a concise explanation of what the model is learning, as studied in [38]. Then, the next step will be to define a basic vocabulary of interesting characteristics in order to represent the intrinsic properties.
- Include shading to the approach. The proposed approach only uses reflectance data, thus shading can be estimated performing a pixel-wise division between the input image and the reflectance prediction. As the MPI Sintel Dataset has a precise reflectance ground truth, shading data ground truth can be extracted. This shading data can be included to the network training stage. Afterwards, EURNet can be modified adding a subnetwork –similar to the hypercolumn philosophy– that simulates Eqn. 1 and outputs the shading component.
- Add a regularizer network. To improve the semantic characteristics, an 1×1 convolution with three feature maps followed by a linear activation could be attached to the fourth Encoder scale level –in the fifth scale level, images are too small and the spatial information is fuzzy. The ground truth data required to train this additional network output would be the same ground truth data but having width and height dimensions divided by eight.
- Add an attention mechanism. Every dataset has its own characteristics, i.e. The MPI Sintel Dataset is mainly brownish. Therefore an attention mechanism can be added to focus more on this particularity.
- Prove the generalization capability. Try EURNet methodology with other datasets, –i.e., specially the incoming promising CVC dataset.

ACKNOWLEDGMENT

The present thesis is the result of a year of painstaking work, and it would not have been possible without the guidance of many individuals who contributed in the completion of the herein presented dissertation.

I would like to thank Maria Vanrell for her valuable support and guidance as supervisor. And also all the professors of the Master in Computer Vision for the well-made master classes and the concepts learned which played an important role to the consecution of the thesis.

Last, but not least, I would like to thank the unconditional support that all my family and friends have given me over these time. Especially, I want to thank my parents Andreu and Maria Isabel for their patience and my brother Andreu, who introduced me to the exciting world of machine learning.

REFERENCES

[1] M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis, and Machine Vision. Thomson-Engineering, 2007.

[2] H. Barrow, "Recovering intrinsic scene characteristics from images," Computer Vision Systems, pp. 3–26, 1978, cited By (since 1996) 143.

[3] N. Bonneel, B. Kovacs, S. Paris, and K. Bala, "Intrinsic decompositions for image editing," Computer Graphics Forum (Eurographics State of the Art Reports 2017), vol. 36, no. 2, 2017.

[4] L. Lettry, K. Vanhoey, and L. V. Gool, "DARN: a deep adversial residual network for intrinsic image decomposition," CoRR, vol. abs/1612.07899, 2016. [Online]. Available: http://arxiv.org/abs/1612.07899

[5] L. Shen, P. Tan, and S. Lin, "Intrinsic image decomposition with non-local texture cues," in 2008 IEEE Conference on Computer Vision and Pattern Recognition, 2008.

[6] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in European Conf. on Computer Vision (ECCV), ser. Part IV, LNCS 7577, A. Fitzgibbon et al. (Eds.), Ed. Springer-Verlag, Oct. 2012, pp. 611–625.

[7] M. Serra, O. Penacchio, R. Benavente, M. Vanrell, and D. Samaras, "The photometry of intrinsic images," in 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014, 2014, pp. 1494–1501. [Online]. Available: https://doi.org/10.1109/CVPR.2014.194

[8] J. T. Barron and J. Malik, "Intrinsic scene properties from a single RGB-D image," IEEE Trans. Pattern Anal. Mach. Intell., vol. 38, no. 4, pp. 690–703, 2016. [Online]. Available: https://doi.org/10.1109/TPAMI.2015.2439286

[9] T. Narihira, M. Maire, and S. X. Yu, "Direct intrinsics: Learning albedo-shading decomposition by convolutional regression," CoRR, vol. abs/1512.02311, 2015. [Online]. Available: http://arxiv.org/abs/1512.02311

[10] IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009. IEEE Computer Society, 2009. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5453389

[11] J. Barron and J. Malik, "Shape, illumination, and reflectance from shading," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-117, May 2013. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-117.html

[12] A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI, ser. Lecture Notes in Computer Science, vol. 7577. Springer, 2012. [Online]. Available: https://doi.org/10.1007/978-3-642-33783-3

[13] Q. Chen and V. Koltun, "A simple model for intrinsic image decomposition with depth cues," in IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013, 2013, pp. 241–248. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2013.37

[14] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," CoRR, vol. abs/1701.00160, 2017. [Online]. Available: http://arxiv.org/abs/1701.00160

[15] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. IEEE Computer Society, 2015, pp. 2650–2658. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.304

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States., 2012, pp. 1106–1114. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," CoRR, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[18] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," CoRR, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[19] A. Goldbloom, "Kaggle," 2010. [Online]. Available: http://www.kaggle.com

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," CoRR, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[21] B. Hariharan, P. A. Arbeláez, R. B. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," CoRR, vol. abs/1411.5752, 2014. [Online]. Available: http://arxiv.org/abs/1411.5752

[22] F. Chollet et al., "Keras," https://github.com/fchollet/keras, 2015.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," CoRR, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[24] S. P. Singh and S. Markovitch, Eds., Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA. AAAI Press, 2017. [Online]. Available: http://www.aaai.org/Library/AAAI/aaai17contents.php

[25] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," CoRR, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[26] G. Larsson, M. Maire, and G. Shakhnarovich, "Learning representations for automatic colorization," CoRR, vol. abs/1603.06668, 2016. [Online]. Available: http://arxiv.org/abs/1603.06668

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[28] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," CoRR, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

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

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," CoRR, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[31] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," Neural Comput., vol. 10, no. 7, pp. 1895–1923, Oct. 1998. [Online]. Available: http://dx.doi.org/10.1162/089976698300017197

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," CoRR, vol. abs/1502.01852, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852

[33] L. Wang, Y. Yang, M. R. Min, and S. T. Chakradhar, "Accelerating deep neural network training with inconsistent stochastic gradient descent," CoRR, vol. abs/1603.05544, 2016. [Online]. Available: http://arxiv.org/abs/1603.05544

[34] V. Chandrasekhar, J. Lin, Q. Liao, O. Morère, A. Veillard, L. Duan, and T. A. Poggio, "Compression of deep neural networks for image instance retrieval," in 2017 Data Compression Conference, DCC 2017, Snowbird, UT, USA, April 4-7, 2017, 2017, pp. 300–309. [Online]. Available: https://doi.org/10.1109/DCC.2017.93

[35] 2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015. IEEE Computer Society, 2015. [Online]. Available: http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7407725

[36] D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII, ser. Lecture Notes in Computer Science, vol. 8695. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-10584-0

[37] S. Srinivas, R. K. Sarvadevabhatla, K. R. Mopuri, N. Prabhu, S. S. S. Kruthiventi, and R. V. Babu, "A taxonomy of deep convolutional neural nets for computer vision," Front. Robotics and AI, vol. 2016, 2016. [Online]. Available: https://doi.org/10.3389/frobt.2015.00036

[38] I. Rafegas, M. Vanrell, and L. A. Alexandre, "Understanding trained cnns by indexing neuron selectivity," CoRR, vol. abs/1702.00382, 2017. [Online]. Available: http://arxiv.org/abs/1702.00382