Image colorization through CNN's & Segmentation models

Monaco Saverio
SaverioMonaco@github.com

Ziliotto Filippo
ZiliottoFilippoDev@github.com

Abstract

Image colorization is a challenging topic of ongoing reasearch in Computer Vision field and many models have been developed in the past years to tackle this problem. We take a greyscale image as input and attempt to produce a coloring scheme. The goal is to make the output image as realistic as possible, although not necessarily the same as the ground truth version. We use a feed-forward pass in a Res-Net18 [2] building on top of it many deconvolutional layers to upscale our features. For general scenarios, we added a pretrained segmentation model merging several trained weights for each segmented part of the scene.
We show how a simplistic model (trained on different scene categories) can produce credible outputs in many types of scenes, requiring reasonable computational hardware (w.r.t to the state-of-the art mondels). The work is inspired by the known papers from Iizuka et al.[4], Zhang et al.[8], and Larsson et al.[5] on image colorization.

1. Introduction

In image colorization, we aim to automatically generate a colored image from a grayscale input. Any color photo can be used as a training example, simply by taking the image's L channel as input and its ab channels as the supervisory signal [8].
Colorization poses a significant challenge to traditional computer vision techniques. This because a single black-and-white image may have multiple plausible colorizations (outputs). This ambiguity makes it difficult to formulate an objective loss function to work with, and difficulty to improve the work already done in literature. As such we state that there is no one correct solution. For instance, when comparing an image of a car that is colored blue but appears red in the ground truth image, results require subjective inspection. So deciding whether an image is plausibly colored via automated means is a problem nearly as difficult as colorization itself. [1]

Figure 1. Image colorization example in computer vision's field. On the left we have out input image while on the right the model's expected prediction.

1.1. Related work

Until recent years, automatic colorization was a problematic task to achieve. In 2005 a paper [3] described how human can actually suggest to the computer which colors to fill in, but this process still required heavy input by the user and this isn't part of the final (fully automatic) goal. Nevertheless recent work has produced vibrant, natural coloured images with deep convolutional neaural networks (CNNs).

[1] All the code, references, related work and additional material of our project can be found at https://github.com/SaverioMonaco/ImageColorization

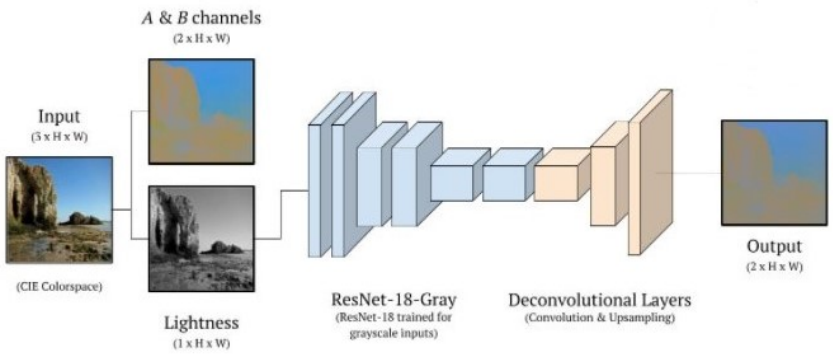

Figure 2. our model's architecture built from a Res-Net18 already implemented by Pytorch, and adding some deconvolutional layers on top of it. Every convolutional layer is followed by a batch normalization and *Relu* layers. No pooling layer is used.

In Zhang et al. [8] paper CNN's are used to convolute the image, here we use the same technique with a ResNet-18 model (modifying the first layer to accepts grayscale input) and extracting the mid-level features. We also add some deconvolution layers to upsample the image. In [5] paper they combine the mid-level feature extracted to the global ones by employing hypercolumns, vectors of concatenated features extracted from all levels of the network. Here we try to keep thing as simple as possible to show how a simpler model can still give plausible results.
As for many papers [8] [4], for the ease of use we also work in the LAB colorspace, which contains the same information as the *RGB* channels.

1.2. Dataset

We train our model separately on different kinds of datasets: the known Places365 [10], Caltech-UCSD Birds-200 [7], Animals-10 [2] and celeb-A [6].
The choice of these datasets was done in order to have a variety of different input images, showing how our model behaved in different situations; e.g. bird images have many dominant colors such as blue, red, yellow, green, while faces have more or less the same coloring scheme. In addition, training was done on a mixture of outdoor and indoor scenes, foreground and background subjects. The datasets were all used in their integrity due to the small size of memory usage they had, except for Places365 and Celeb-a which only a small part was taken for training.
Preprocessing was needed in order to have a training and a validation set. Of course large pretrained models as ResNet-18 uses 256×256 input images so a uniform cropping was applied, but we also added some data augmentation (e.g. random cropping and horizontal/vertical flipping) for generalization purposes. Due to the desaturated outputs produced we slightly enhanced the training images saturation.

1.3. Colorspace

Finally as a part of the preprocessing, we converted RGB layers of each image into Lab colorspace preserving all the overall colour information. In Lab space, we have an *"L"* layer which represents the grayscale image, an *"a"* layer that represents the red-green color spectrum, and a *"b"* layer that relates to the blue-yellow color spectrum. The L layer acts as the input to our model and the a,b layers become the target image we want to infer, recombining all three layers for the final result. From a technical perspective, the L layer ranges from 0-100, whereas the a,b layers are pixel values range from 0-255 as usual.

[2] A basic kaggle dataset, which labels ten types of animals in various scene configuration.

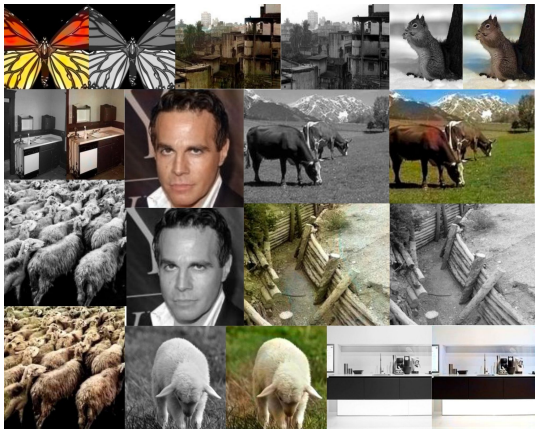

Figure 3. Example input grayscale photos and output colorizations from our model. These examples are cases where our model works especially well.

2. Method

We developed one colorization network with different training weights. This gave plausible outputs only for certain scenarios. To further improve the generalization of the model we added a pretrained segmentation network merging the weights of the various categories recognized in the scene.
The main idea leading the project was getting plausible results even with an overall simple model. Indeed, there are many other possible approaches to the problem producing more color accurate outputs, but they require both deeper architectures and larger datasets (e.g. Imagenet [1]) leading to exponentially more training time and perhaps more powerful hardware.

2.1. Loss function

Given the input lightness channel $\in \mathbb{R}^{H\times W\times 1}$ and the two associated color channels $\in \mathbb{R}^{H\times W\times 2}$ we try to infer the learnable mapping $Y = F(X)$, where (H, W) are image dimensions. We choose the simple mean squared error loss function. The benefit of using this loss function is in its simplicity. In *Lab* colorspace the model's perceptual distance it's a natural objective function, i.e. the Euclidean loss $L_2(\cdot,\cdot)$ between predicted and ground truth colors. So we can clearly measure the distance between the target pixel value and the predicted pixel output for each color layer.
To be fair we also tried other simple loss functions, e.g. L_1 loss, but the results were not as good.

$$L_2(\hat{Y}, Y) = \frac{1}{2}\sum_{h,w}\left\|\hat{Y}_{h,w} - Y_{h,w}\right\|_2^2$$

The main drawback is that this loss in most cases encourages conservative predictions[8]. For example, it heavily penalizes picking green instead of red, when either choices might be realistic (e.g. in bird scenes). In color prediction, this averaging effect favors grayish, desaturated results. This is the reason for which some outputs are dominated with a brown/green tint. As stated before, the core problem is the multimodality of the color choice. A further solution could be to integrate in the loss function a classification term to include the *semantic prior* [9] information we already have about a certain picture, e.g. we know that the grass in the majority of the cases is green.

2.2. Approach

Our approach is to first apply a number of convolutional layers (from the ResNet-18 architecrure) to extract image features, then applying some deconvolutional layers to upscale. We didn't freeze the learning of the first part due to the fact that it was trained for classification purposes (and of

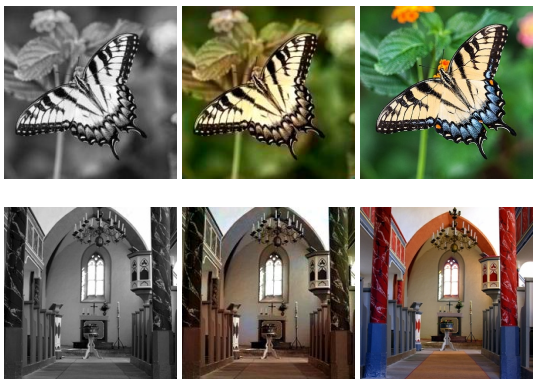

Figure 4. Comparison between our model output and the ground truth image. Here we see the multimodality problem of the colorization task.

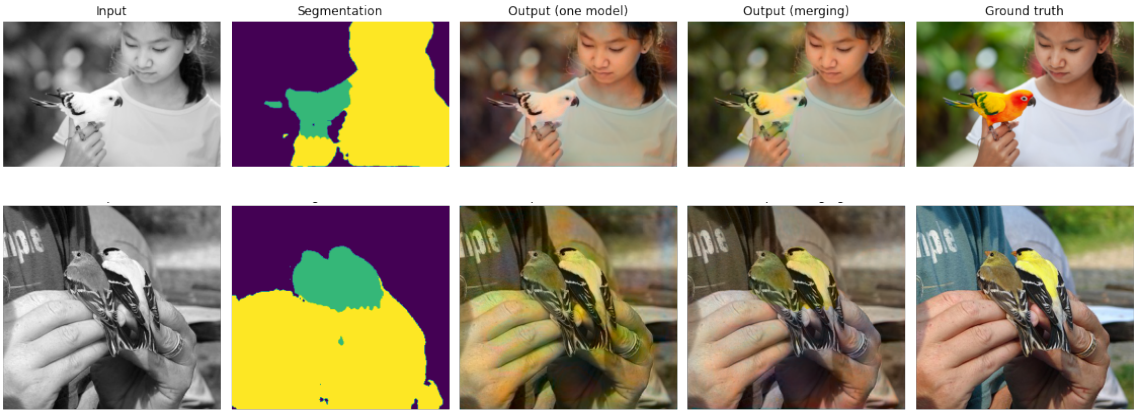

Figure 5. Improving the model generalization by adding a pretrained segmentation architecture to apply the different training weights to the segmented image (4^{th} image). In this case we used three model weights related to people, birds and backgrounds through the places dataset.

course it required *RGB* images), a very different problem with respect to ours.
Performing the training on different datasets resulted in having different models for each scene type. Trying various learning rates we find that the best was an *adam* optimizer with default values, expect for the weight decay set to *0.001*. We also chose to have an adaptive step decay learning rate, starting from $l_r = 10^{-2}$ to $l_r = 10^{-5}$ after about 50 epochs. Ultimately we settled on a batch size of *64*.

3. Experiments

The training part was done with a multicore 32-CPU gear via *CloudVeneto* resources, but also Google Colab GPU's were used. Results varied from the different scene types we trained on. The best results are related to the celeb-A dataset, were images are very similar and dominant colors (e.g. pink/brown skin tone) remain more or less the same. Here the model succeeded in predicting a credible output (Fig.9). Good results are achieved also for animal scenes, especially for foreground on focus subjects (Fig.4). Concerning the places dataset, results varied from having a good credible image to a total lack of generalization in more complex situations. On the other hand the model does not reproduce the ground truth image for bird scenes in the majority of the cases. These poor results can be explained by the lack of a sosphisticated enough model to represent the task and a small-sized dataset to train on.
We compared our model with *Eccv16*[8] and *Siggraph17*[9] in Fig.9. These are very complex models trained on Imagenet [1]. Still ours managed to compete in some cases (see comparison table at the end). Another noticeable difference we see is the difficulty in clearly predicting the non identical color near the edges of the object, where our model isn't robust. It also seems to overfit the training data sometimes, e.g. it produces greenish trees even though the ground truth has bare trees or produced green grass around the subject where there's arid land (Fig.9, elephant/place pictures)[3].

In Fig.9 we show were our model fails to generalize effectively tending to predict desaturated images w.r.t. the ground truth, resulting in a more brownish image. The saturation enhancing on the input images partially solved the problem, this does depend on whether the model correctly predicted the input image in the first place (i.e. bad

[3]Fig.9. Last page figure, a comparison between ours and the models proposed by . In the first five comparison our model succeeds to reproduce a credible output, while in the last two cases it fails to generalize w.r.t. the other methods.

predictions still remain bad even after this process).

3.1. Improving the model

As it is showed in the last Table, our models produces fairly credible outputs when the images contain just one semantic subject, although it generally fails when more than one are present. This lack of generalization is due to the fact that our models were trained with small and specific datasets, due to hardware and time constrains.
Since our model performed well when applied to specific images, our solution was to merge multiple outputs of the same input image in the areas were the model performed better.
Consider for example this image and his segmentation map:

Figure 6. Input and segmentation map

The Segmentation model found that the image contains *human*, *bird*, and *background*, the grayscale image is then passed as input in the models trained with those specific datasets, respectively *human*, *birds*, and *places*.

Figure 7. Outputs of the models

In Figure 7 we show the outputs of the grayscale image for the all the single models. As expected they do not perform well when recoloring subjects they were not trained with. With the assumption that each model performs best in the area containing the semantic subject they were trained for, the final image is processed merging the parts according to the segmentation map.

Figure 8. Merged output and Groundtruth

Since the image needs to be processed multiple times, this technique is computationally less efficient and performing than having only one model trained with a larger and heterogeneous dataset (e.g. ImageNet[1]). Given our constrains that option could not be choosed, however our method still manages to provide plausible results for many complex images and it highly improves the results for images generated without the merging (See Fig.5).

4. Conclusion

Overall, we gained a newfound appreciation from the challenge of producing realistic colorizations, comparing our results with state-of-the-art ones (Fig. 9). The Lab colorspace seems the best choice to work with grayscale images. Our series of convolutional neural networks is a good simple model to tackle the task but, adding a parallel classifier could be a further implementation in order to exploit the *deep priors* of the scenes[9]. A good practice would be training on a large general dataset, e.g. ImageNet [1], leading more generalized model. We also showed how adding a segmentation network, merging the related weights to each part of the scene, improves dramatically the output for simple models tested in complex scenarios.
To summarise, while image colorization is a boutique computer graphics task, it is also an instance of a difficult pixel prediction problem in computer vision. A deep CNN and a well-chosen objective function can come closer to producing results indistinguishable from real color photos[8]. Our implementation provides some what useful graphical outputs in specific scenes still remaining as simple as possible.

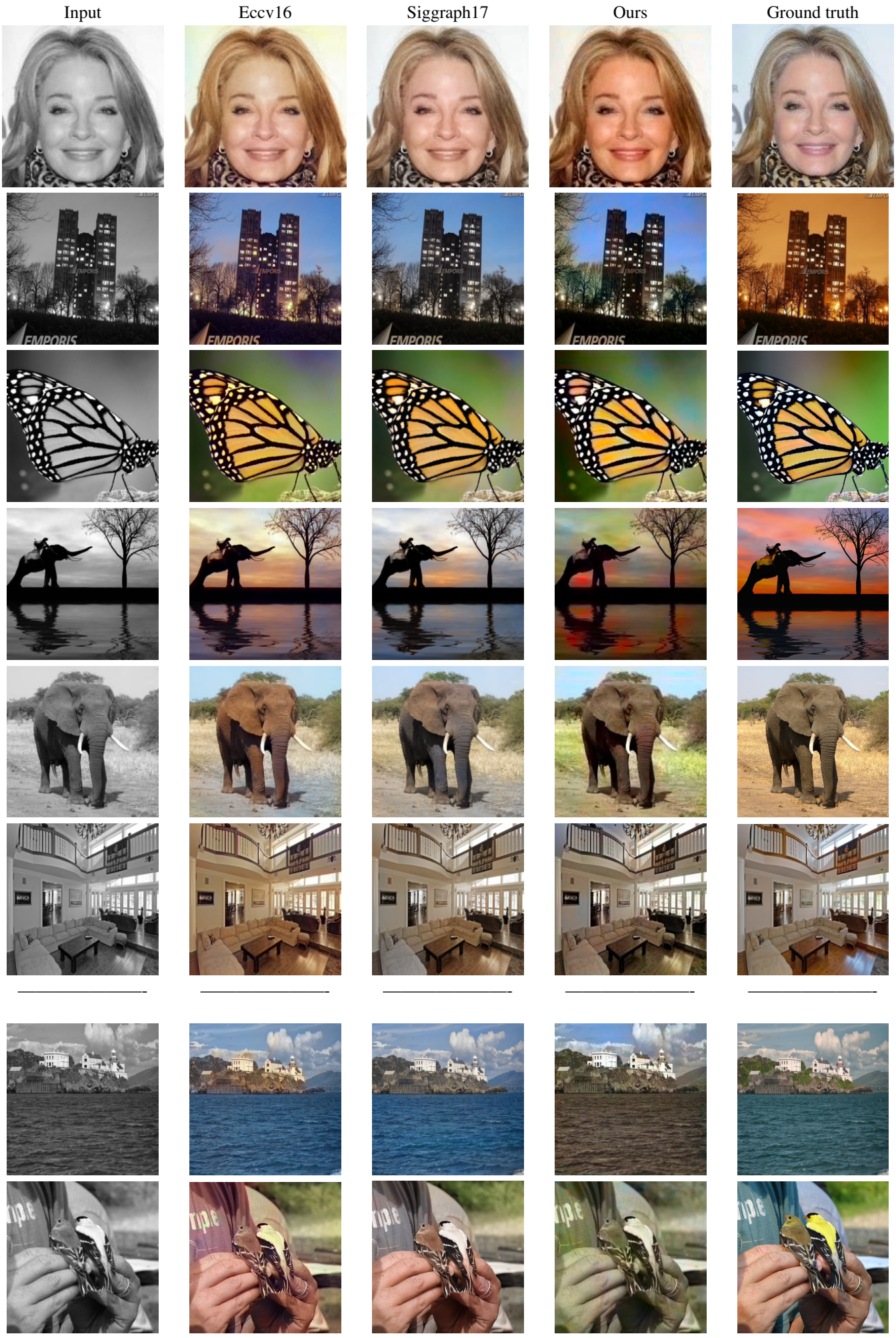
Input
Eccv16
Siggraph17
Ours
Ground truth
Success cases
Failure cases

Figure 9.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[3] Yi-Chin Huang, Yi-Shin Tung, Jun-Cheng Chen, Sung-Wen Wang, and Ja-Ling Wu. An adaptive edge detection based colorization algorithm and its applications. pages 351–354, 01 2005.

[4] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification.

[5] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization, 2017.

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[7] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[8] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.

[9] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

[10] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.