Deep Demosaicing

Ramchalam Kinattinkara Ramakrishnan*, Shangling Jui†, and Vahid Partovi Nia*

Abstract

Most digital cameras use sensors coated with a Color Filter Array (CFA) to capture channel components at every pixel location, resulting in a mosaic image that does not contain pixel values in all channels. Current research on reconstructing these missing channels, also known as demosaicing, introduces many artifacts, such as zipper effect and false color. Many deep learning demosaicing techniques outperform other classical techniques in reducing the impact of artifacts. However, most of these models tend to be over-parametrized to be suitable for edge devices such as cell phones. Consequently, edge implementation of the state-of-the-art deep learning-based demosaicing algorithms on low-end edge devices is a major challenge. We show that an exhaustive search over a space of deep neural networks using some prior domain knowledge beats the state-of-the-art deep demosaicing architecture.

1 Introduction

The success of deep learning models has been proven in many high-level vision tasks such as image classification and object detection. However, deep learning solutions are less explored for low-level vision problems where edge implementation is an obligation. Previous work shows the effectiveness on other low-level vision problems such as image denoising and image demosaicing (Xie et al., 2012). Here we explore various architectures for demosaicing, with the aim of finding suitable models for deployment on the edge. It is imperative to make use of models that can be stored on edge devices.

Image demosaicing involves interpolating full-resolution images from incomplete color samples produced by an image sensor. Limitations in camera sensor resolution and sensitivity leads to the problem of mosaicing. For a digital camera to capture or produce a full color image, it uses sensors to detect all 3 colors (RGB) at every pixel location. One way this is done involves using a beam-splitter to project the image onto three separate sensors, one for each color (RGB). Color filters are placed in front of each sensor to filter specific wavelengths such that three full-channel color images are obtained. This is a costly process that is generally used in scientific-grade microscopes. Most digital cameras, on the other hand, make use of just one sensor coated with a Color Filter Array (CFA). In this system, one color component is captured at every pixel location and the missing channels are reconstructed. The resultant image is often called a mosaic image, derived from the CFA’s mosaic pattern. This mosaic image is then subjected to software-based interpolation, resulting in a full-resolution image; a process termed as demosaic-ing. Various existing CFA patterns are currently used, with the Bayer CFA being the most commonly used approach for mosaicing.

Let \( I^{CFA} : \mathbb{Z}^2 \to \mathbb{Z}^3 \) denote an \( M \times N \) Bayer CFA image. If we consider the default sampling pattern then the mosaic image is as follows:

\[
I(i,j) = \begin{cases} 
R_{i,j} & \text{for } i \text{ odd and } j \text{ even}, \\
B_{i,j} & \text{for } i \text{ even and } j \text{ odd}, \\
G_{i,j} & \text{otherwise},
\end{cases}
\]

where \( R_{i,j}, G_{i,j}, B_{i,j} \) includes values between 0 and 255.

To estimate the missing two missing channels for each pixel location using demosaicing,

\[
\hat{I}(i,j) = \begin{cases} 
(R_{i,j}, \hat{B}_{i,j}, \hat{G}_{i,j}) & \text{for } i \text{ odd and } j \text{ even}, \\
(\hat{R}_{i,j}, B_{i,j}, \hat{G}_{i,j}) & \text{for } i \text{ even and } j \text{ odd}, \\
(\hat{R}_{i,j}, \hat{B}_{i,j}, G_{i,j}) & \text{otherwise},
\end{cases}
\]

where \( \hat{R}_{i,j}, \hat{G}_{i,j}, \hat{B}_{i,j} \) are the estimates for the channels at each pixel location.

A common demosaicing approach is to capture the ground truth images using a professional 3-sensor camera, convert them into a mosaic format using Bayer CFA, interpolate using a demosaicing algorithm, and then compare the results with the ground truth image. Widely used performance criteria are the color mean squared error (CMSE) or the equivalent color Peak Signal to Noise Ratio (CPSNR). The Kodak / McMaster (Yu et al., 2018) dataset is generally used as a baseline because of the realistic scenes and varied complexities. The Color Mean Squared Error (Menon and Calvagno, 2011) averages the MSE over the color channel as well,

\[
CMSE(I, \hat{I}) = \frac{1}{3MN} \sum_{k \in \{R,G,B\}} \sum_{i=1}^{M} \sum_{j=1}^{N} \{\hat{I}(i,j) - I_k(i,j)\}^2,
\]

The color peak signal to noise ratio is defined as,

\[
\text{CPSNR}(k) = 10 \log_{10} \left( \frac{R^2}{CMSE} \right),
\]

where R is 1.0 (for float datatype), R is 255 (for integer datatype).

2 Architecture Search

For the exhaustive search of architectures, we made use of various parameters chosen based on state-of-the-art mod-

---

*Huawei Noah’s Ark Lab, † Huawei Kirin Solution
We first describe the dataset that was used for our experiments and elaborate on the results obtained. Flickr500 data is used by Syu et al. (2018) to generate around 3.5 million patches. We had to make some assumptions while generating our data due to some missing details in the original paper.

1. We generated random patches of $32 \times 32 \times 3$ to obtain approximately 1.5 million image patches from the 500 images. Since the method is not specified, we used random points in the image and extracted patches from the random points. 1382400 patches were used for the training set and 1536000 patches for the validation set. Note: In the paper, they use patches of size $33 \times 33 \times 3$ but we used $32 \times 32 \times 3$ because the training is much faster on GPU due the dimensions being a power of 2 for tiling.

2. The Bayer CFA generation is also based on the python package. We used the default setting of RGGB and filled the missing channels with zeros to maintain the channel dimensions.

3. All the results in the section were calculated against the McMaster dataset (MCM) containing 18 photos (58000 patches created), and the Kodak images containing 24 photos (78408 patches created).

The Pareto front of our exhaustive search is shown in Figure 1. The $y$-axis represents the validation loss $\mathcal{L}(.)$ which indicates the negative CPSNR and the $x$-axis represents the model complexity $C(.)$, which is the total number of parameters in the model. The standard error for each experiment is calculated and as they are comparable, they produce a similar prediction interval.

Most of the architectures in the search space outperform the current state-of-the-art (SOTA) as in Figure 1 in a similar setting. The total number of parameters in the current SOTA is approximately $7 \times 10^5$. Architectures on the Pareto front maintain the same accuracy while significantly reducing the number of parameters. In our search the effect of cosine annealing was not obvious and would need further tuning on cycle rate. The ResNet skip connections length of 2 blocks worked better than 1.

Once the initial search was completed, the models from the Pareto front architectures were fed through a Bayesian optimizer using the ORION package (Tsirigotis et al., 2018), to optimize the learning rate and regularization constant for $L_2$ regularization. Figure 1 indicates further improvement over the initial CPSNR obtained with a fixed learning rate $10^{-4}$ and regularization constant $10^{-8}$, so Bayesian optimization is recommended as an extra step after the architecture search as including it in the first step would further increase the complexity of the search. In our experiments, the PSNR was calculated separately for each image in the test dataset; the final CPSNR value was the mean of all the PSNRs of all images in the dataset.

3 Conclusion

Our designed space with a simple exhaustive search outperforms the state of the art and brings a range of loss versus complexity for edge implementation with varying resource constraints. Although in most of architecture search, the number of evaluations and complex search algorithm implementation is the bottleneck, here we showed using prior vision domain knowledge, we can overcome these drawbacks and a simple exhaustive search becomes an effective search tool. The theoretical analysis for using exhaustive search is further detailed in Ramakrishnan.
et al. (2019). We only focused on demosaicing as an example of a low-vision problem where edge implementation is an obligation.

References

Loshchilov, I. and F. Hutter
- Sgdr: Stochastic gradient descent with restarts. CoRR.

Ma, N., X. Zhang, H. Zheng, and J. Sun

Menon, D. and G. Calvagno

Ramakrishnan, R. K., J. Shangling, and V. P. Nia

Sandler, M., A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen

Sifre, L. and S. Mallat

Syu, N., Y. Chen, and Y. Chuang


Xie, J., L. Xu, and E. Chen

Yu, K., C. Wang, S. Yang, and L. et al.