Deep-Demosaicing Using CNNs

Stanford University, *Ike Osafo Nkansah*, *Guillermo de Leon Archila*

***Abstract*—The resurgence of AI methods, facilitated by methods such as backpropagation and the growth in computational power, as viable approaches to solving complex problems has amassed a lot of good traction in many different fields. In the field of image processing, AI methods have been applied to many different stages of the image processing pipeline: denoising, demosaicing, deconvolution etc. This paper aimed to explore the application of deep neural networks to the problem of demosaicing; in particular, this paper explored full end-to-end CNN architectures which takes in raw images from camera sensors and directly yield images in the linear or colored space. The core find of this paper is that designing an optimal CFA-agnostic demosaicing model require having very deep architectures as it is generally harder for the model to infer the color filtering pattern of a random raw image but designing a CFA-specific demosaicing model is more tractable and generally warrants shallower networks, yet sufficiently deep, to surpass traditional demosaicing algorithms. This work distinguishes itself from other works in the literature in that it seeks to build a full CNN that includes minimal prior knowledge and assumptions from the traditional demosaicing algorithms, such as bilinear interpolation etc. In this way, this approach allows the model the flexibilty to explore unconventional parameters that may help improve the effectiveness of the demosaicing task.**

***Index Terms*—Bayer pattern, CFA, Transfer learning, Encoders, SSIM, L2, L1, ReLU, MS-SSIM, CNN, DMCNN**

I. Introduction

Digital cameras are readily available all around the world. This was made possible by the development of efficient technologies that improve each step in the image processing pipeline. Many digital cameras use a color filter array (CFA) pattern when recording an image (most commonly the Bayer pattern). The raw image captured by the camera sensor, through the CFA, contains information of only one color per pixel. Here the necessity of an efficient interpolation algorithm for demosaicing arises as a fundamental part of the image processing pipeline.

Traditionally, the demosaicing task has been done by the implementation of linear and nonlinear filters that seek to achieve the best interpolation paradigm. In [1] we can find a comparative study in the performance of nine different demosaicing algorithms. In all nine cases, color artifacts can be observed in some of the processed images. These methods have been outperformed by data-driven approaches, such as the application of convolutional neural networks (CNNs) to solve the demosaicing problem [2].

An interesting conundrum with data-driven approaches is that any training dataset of raw images to colored images would have relied on some specific demosaicing scheme to generate the colored images. This means in principle that with data-driven approaches, we are constrained in performance by whatever approach was used to generate the training dataset. In the work we will be discussing in the subsequent sections, we seek to evaluate the performance of our own implementation of a convolutional neural network and compare it to traditional demosaicing methods and other data-driven techniques. In particular, our approach seeks to incorporate priors and regularizers that can mitigate the effect of the prior demosaicing algorithms used to generate the training dataset images, all the while seeking to outperform traditional demosaicing methods.

II. Related Work

Our work seeks to supplement other works within the literature of applying convolutional neural networks to the different problems in the image processing pipeline. The ideas we explore here builds primarily on the works of [2], [4], [8]. All of these papers use a patch-based method, where the input raw image is first expanded out into the mosaic pattern in the linear rgb color space depending on the type of CFA pattern that was used to capture the dataset. In doing so, these methods were not aiming to be agnostic to the type of CFA pattern that was used to acquire the raw image. Our work explores the possibility of universalizing an architecture that is agnostic to the type of CFA pattern. Secondly, the patch-based method assumes strong pixel locality. For example, in [2] patches of 33x33 are extracted from the raw image dataset and used to train their DMCNN and DMCNN-VD models. Though the patch sizes are considerably large, allowing flexibility for the model to learn dependencies between both near and far neighboring pixels, the field of view of the model is inherently limited, regardless of how deep the neural network is due to the limited spatial size of the input patches. What this means is that assuming there are dependent pixels that are farther than 33 pixels in euclidean distance away from a particular center pixel, these architectures will not be able to capture the holistic effect. In using relatively deep models with the full raw image as input, we enforce spatial locality in defining the pixel color values, through small fields of view of the overall model. Zhao Hang et. al. note in their research that for imaging tasks, the L2 loss function may not be a good function as it check pixel-wise differences. They recommend patch-based similarity index methods such as Structural Similarity Index Metric (SSIM) and Multiscale-SSIM (MS-SSIM). From the results they compiled, we noted that the L2 loss produced images that were less visually pleasing (with more color artifacts and blurry grid artifacts) and so inspired by this, we ended up defaulting to L1 loss function in our model training because it had very comparable visual appeal to the MS-SSIM and the SSIM results the paper explored. The rest of the cited works on demosaicing CNNs helped us with quickly iterate through some good hyperparameter tunings as we explored our own CNN versions for the demosaicing task.

III. Methods

The problem of demosaicing can be formulated as an inverse problem of the form given below:

$$y = Ax$$

with y as either the vectorized version of the raw image (compressed CFA pattern), as shown in 1 with the bayer pattern, or a vectorized version of the decompressed 3-channel raw image (shown in 1, also with the bayer pattern) and x as the vectorized reconstructed full image in the sRGB space.

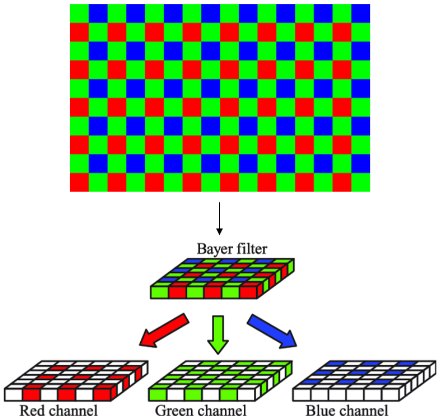


Fig. 1. Bayer Pattern splitting *Figure recovered from* [10]

This problem is better formulated algebraically as a generalized least norm problem of the form below:

$$\text{minimize } \|x - By\|$$
$$\text{subject to: } y_{\text{filtered}} = x_{\text{filtered}}$$

where B is the inverse of matrix A as defined in the inverse problem above. As such matrix B takes in the vectorized raw image and outputs a reconstructed rgb image. $y_{filtered}$ is defined as a vector of the non-zero values in the decompressed CFA pattern raw image and $x_{filtered}$ is similarly the non-zero values in x that matches in the same index positions as the vectorized y. This formulation inspires a different way of thinking about the problem. The traditional methods such as bilinear interpolation, bilinear interpolation with ycrcb filtering conforms to the algebraic formulation above where prior to the demosaicing step, the color channels in the output colored image are first prefilled with the non-zero values of the CFA-filtered raw image before the interpolation. This ensures that the constraint in the generalized least-norm formulation is enforced. The objective in this least norm problem is not to find x, the ground truth images in sRGB/linear space, but instead to find the matrix B (inverse of A) that transforms the input measured raw image into the ground truth images. In the traditional demosaicing approaches, we are interested in designing linear filters which can be transformed into the linear matrix B of interest. But in the context of using CNNs, the goal is to learn a function f, such that:

$$B \approx f(y)$$

with the variables defined as above. This formulation allows for more flexibility in using non-linear filters to attempt to solve this least-norm problem of demosaicing. As far as the goal for this project is concerned, we aimed to design a deep neural network (Deep CNN) that is able to faithfully demosaic raw images, even on images with high spatial frequencies.

Dataset: The very first step in the implementation of any data-driven learning algorithm is to get a reliable dataset. There are many readily available data-sets that have been used before for the demosaicing task, such as in [6] and [7]. To reduce the complexity of the problem, we decided to focus on only bayer pattern dataset. This led us to the Microsoft Demosaicing Dataset, which contains images that were shot with 2 different cameras: 1. Panasonic Lumix DMC-LX3 (500 images) and 2. Canon EOS 550D (57 images). We decided to use only the panasonic dataset to train and evaluate our model. Though 500 images is appreciably large, compared to most computer vision datasets, this dataset size is very small and hence there was the potential to ran into overfitting even with appreciably sized models. Our resolve was to train models that weren't too deep (keeping the maximum number of trainable hidden layers to no more than 10. We ended up using the consistent split below for training all our models:

$$(Training, Validation, Testing) \rightarrow (425, 60, 15)$$

Finally for all our models, we first preprocessed the input raw images by and then scaling the input 16-bit values to between 0 and 1 and then decompressing the scaled input raw image into its mosaiced channels before feeding it through our various networks. We did not apply any normalizations to the input raw images and we will discuss why this choice was made in the later sections.

CNN Models: We considered 3 primary models as illustrated in the 2, 3 and 4. The very first model was a simple excite-squeeze model which takes in as input the three mosaiced channels of the raw image and then outputs the 3-channel rgb ground truth image (in linear space). The name excite-squeeze comes from the fact that we first *excite* the channels out to a large dimension and then *squeeze* it back into the three channel output image of concern. For this model, we experimented with different hyperparameter fine tuning for this model, which we will discuss in detail later. This model is illustrated in fig. 2

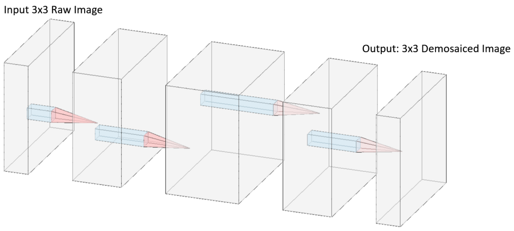


Fig. 2. Excite Squeeze Architecture

The next architecture, shown in fig. 3 was supposed to leverage an already trained model on a large dataset of natural images. The decision to use this was motivated by the fact that these models have seen a lot of images and hence would have ”learnt” how a natural image is supposed to look like. This is intended to be a form of regularization on the generated/predicted grounded truth images by our model. The architecture as seen below in fig. 3 ingests the mosaiced 3-channel raw image and outputs a hidden representation that passes through a *squeeze* network. The function of the squeeze network here is both to squeeze the channel dimension and to restore the image width and height spatial dimensions back to match that of the input raw image.

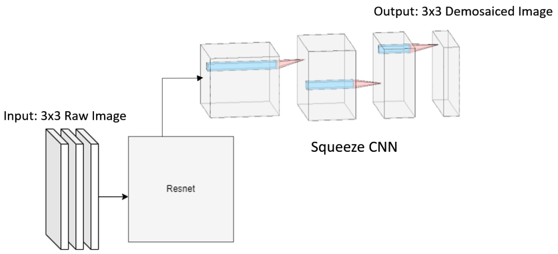


Fig. 3. Transfer Learning with ResNet152 + Squeeze CNN

Our third and final model was inspired by the idea of creating a different representation other than the mosaiced 3-channel input raw image that will be fed into the pretrained model and then finally out through the *squeeze* network to recover the appropraite dimensions. We trained an autoencoding stage to yield this desirable encoding. We hoped that this encoding (though in the space image space as the mosaiced 3-channel image) will allow for the later pretrained and *squeeze* stage to be able to better demosaic from raw to ground truth images.

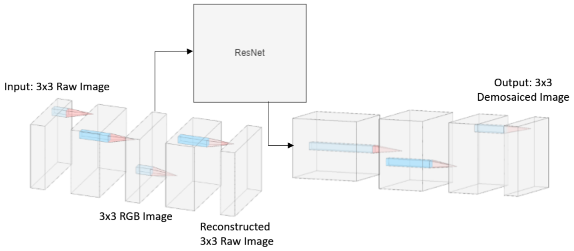


Fig. 4. AutoEncoder + Transfer Learning with ResNet152 + Squeeze CNN

Loss functions: As we described already in the related work section, we noted that the authors of [8] found that L2 loss doesn’t yield visually pleasing outputs so we ended up opting for the L1 loss function, which were visually comparable to the MS-SSIM and SSIM approaches the paper suggested. The L1 loss function is defined below:

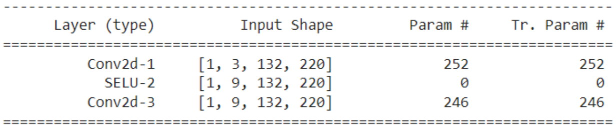
$$Loss(\hat{X}, X) = \frac{1}{n}\sum_{j=1}^{n} \|\hat{X} - X\| \tag{1}$$

where X and are the ground truth images and predicted ground truth images respectively.

IV. Analysis & Evaluation

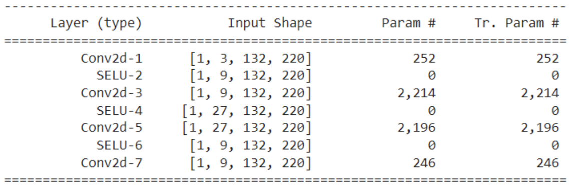
In this section, we will talk about each model and the hyperparameter finetuning we used for them.

Excite-Squeeze Model: To begin with, we will consider our excite-squeeze architecture as described in the methods section. For this architecture, we ended up training 3 flavors (varying only in the depth of the hidden layers). The model architectures we trained with hidden layer depths of 0 and 1 are shown below in fig. 5 and 6.



```
----------------------------------------------------------------------
      Layer (type)           Input Shape          Param #     Tr. Param #
======================================================================
          Conv2d-1      [1, 3, 132, 220]              252             252
            SELU-2      [1, 9, 132, 220]                0               0
          Conv2d-3      [1, 9, 132, 220]              246             246
======================================================================
```

Fig. 5. Parameters Excite-Squeeze with 0 Hidden Layers



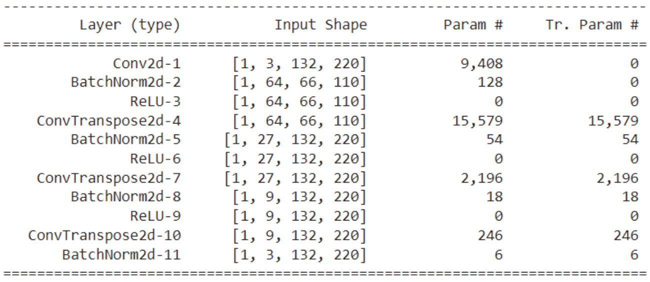
```
----------------------------------------------------------------------
      Layer (type)           Input Shape          Param #     Tr. Param #
======================================================================
          Conv2d-1      [1, 3, 132, 220]              252             252
            SELU-2      [1, 9, 132, 220]                0               0
          Conv2d-3      [1, 9, 132, 220]            2,214           2,214
            SELU-4     [1, 27, 132, 220]                0               0
          Conv2d-5     [1, 27, 132, 220]            2,196           2,196
            SELU-6      [1, 9, 132, 220]                0               0
          Conv2d-7      [1, 9, 132, 220]              246             246
======================================================================
```

Fig. 6. Parameters Excite-Squeeze 1 Hidden Layer

With this simple architecture, we wanted to test whether the depth of the hidden layer affects performance, albeit noting the constraint of a small dataset size. The small MDD (Microsoft Demosaicing Dataset) meant that we shouldn’t train a very deep neural network because of the issues of overfitting. Though we could have used a corresponding L2 regularizer term to reduce this, we reasoned that this will increase our iteration time for converging on a good model so we decided not to explore this route. On the output of these models, we had a sigmoid function that transforms the output of the last conv layer to values between 0 and 1. This was necessary in order to compare the predicted groundtruth images to the original ground truth labels. Additionally, after several iterations exploring different learning rates, we converged on a learning rate of 0.001 and trained both models (fig. 5 and 6) for an average of 150 epochs each. After training, we evaluated on our test set and also on the entire set of 500 images and recorded the final PSNR values in table I. It is worth mentioning that we used kiaming initialization for our convolutional layers, and used SELU activations as can be seen in the table. The reason for this choice was that [2] used this activation for their training and it yielded better results compared to other activations like ReLU or LeakyReLU. Also, we confirmed that this was the case for us with the MDD dataset by training for about 20 epochs to observe the rate of convergence. Both ReLU and SELU seemed to be at par so we just randomly decided to use SELU instead.

ResNet + Squeeze CNN model: In this section, we decided to explore 2 pretrained models, i.e. vgg and resnet. With each of these models, we experimented with different pretrained model sizes: for vgg, we looked at vgg11, vgg14, vgg19 etc.

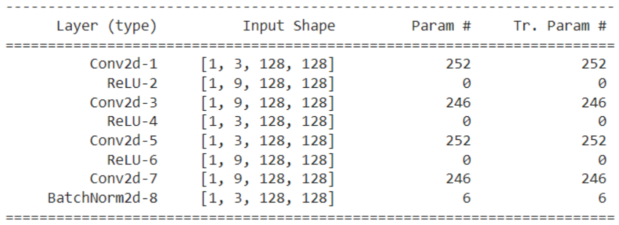
and for resnet, we explored resnet101, resnet18, resnet152. We generally noticed that resnet performed better and particularly resnet152 performed the best, as we can see in table I. This made sense because of the theory that any sufficiently large model can learn any arbitray complex function and so within the domain of computer vision, it was arguable that the larger models would be able to better learn the distribution of natural images compared to the shallower ones. With this insight, we converged on training our model with both vgg19 and resnet152 but we have only reported the results for resnet152 because it performed the best. For both pretrained models, we froze the layers we were interested in and we trained our architecture with an L1 loss, epoch number of rough 150 and just the panasonic image dataset (without any standard image transformations). We froze the layers because we realized that fine-tuning the early layers resulted in slower convergence, though we believed given enough time (epochs), there wouldn't be a difference between the models trained with finetuning and those without. The model architecture is described below:



```
------------------------------------------------------------------------
        Layer (type)             Input Shape       Param #     Tr. Param #
========================================================================
           Conv2d-1      [1, 3, 132, 220]           9,408               0
      BatchNorm2d-2     [1, 64, 66, 110]             128               0
             ReLU-3     [1, 64, 66, 110]               0               0
  ConvTranspose2d-4     [1, 64, 66, 110]          15,579          15,579
      BatchNorm2d-5    [1, 27, 132, 220]              54              54
             ReLU-6    [1, 27, 132, 220]               0               0
  ConvTranspose2d-7    [1, 27, 132, 220]           2,196           2,196
      BatchNorm2d-8     [1, 9, 132, 220]              18              18
             ReLU-9     [1, 9, 132, 220]               0               0
 ConvTranspose2d-10     [1, 9, 132, 220]             246             246
     BatchNorm2d-11     [1, 3, 132, 220]               6               6
========================================================================
```

Fig. 7. Parameters ResNet152 + Squeeze CNN

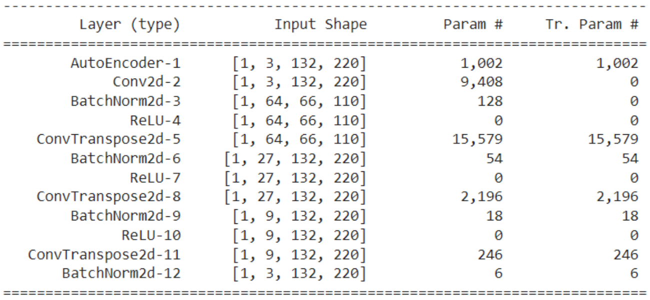
Autoencoder + ResNet152 + Squeeze CNN model: The models we explored here, as already highlighted in the methods section, just built on top of the ResNet152 + Squeeze CNN model by first learning an encoding in the 3-channel dimension space. The aim was to make the later squeeze layer, which performs the main demosaicing interpolations ”easier”. Similar to section above, we tried both resnet and vgg pretrained models, in particular we tried vgg19 and resnet152 and found that resnet152 performed better so we ended up reporting that as our final model. The encoder architecture we used can be found below:



```
----------------------------------------------------------------------
        Layer (type)             Input Shape      Param #     Tr. Param #
======================================================================
            Conv2d-1     [1, 3, 128, 128]            252             252
              ReLU-2     [1, 9, 128, 128]              0               0
            Conv2d-3     [1, 9, 128, 128]            246             246
              ReLU-4     [1, 3, 128, 128]              0               0
            Conv2d-5     [1, 3, 128, 128]            252             252
              ReLU-6     [1, 9, 128, 128]              0               0
            Conv2d-7     [1, 9, 128, 128]            246             246
       BatchNorm2d-8     [1, 3, 128, 128]              6               6
======================================================================
```

Fig. 8. Parameters Autoencoder

The encoder was trained with 4 trainable layers as shown above. The middle layer with the 3 channels was used as our encoding and we trained this model for about 100 epochs with the L1 loss function. The only thing to note here was that our loss function took as input the predicted raw image and the input raw image. We also explored a deeper encoding architecture but realized that the loss converged to similar values after the same number of epochs so we stuck to the shallow encoder that is reported in this paper. The rest of the hyperparameters are as reported for the ResNet152 + Squeeze CNN model. We report the full architecture, including the autoencoder below:



```
------------------------------------------------------------------------
        Layer (type)             Input Shape       Param #     Tr. Param #
========================================================================
       AutoEncoder-1      [1, 3, 132, 220]           1,002           1,002
            Conv2d-2      [1, 3, 132, 220]           9,408               0
       BatchNorm2d-3     [1, 64, 66, 110]             128               0
              ReLU-4     [1, 64, 66, 110]               0               0
   ConvTranspose2d-5     [1, 64, 66, 110]          15,579          15,579
       BatchNorm2d-6    [1, 27, 132, 220]              54              54
              ReLU-7    [1, 27, 132, 220]               0               0
   ConvTranspose2d-8    [1, 27, 132, 220]           2,196           2,196
       BatchNorm2d-9     [1, 9, 132, 220]              18              18
             ReLU-10     [1, 9, 132, 220]               0               0
  ConvTranspose2d-11     [1, 9, 132, 220]             246             246
      BatchNorm2d-12     [1, 3, 132, 220]               6               6
========================================================================
```

Fig. 9. Parameters Autoencoder + ResNet152 + Squeeze CNN

V. Results

Table I shows the mean PSNR of the demosaiced images after applying a series of demosaicing methods. The first four methods correspond to linear filters. These serve as a baseline to compare the performance of the demosaicing methods using CNNs. The best performing method in our data-set corresponds to ”High Quality Interpolation” [9] with low-pass filtering of the chrominance channels in the yrcbc space. The best performing CNN method corresponds to an excite-squeeze architecture, as shown in figure 2, with one hidden layer.

TABLE I
Mean PSNR Over complete Dat-aset for Demosaicing Methods

Model	**PSNR over the Data set**
Bilinear Interpolation	30.26
Bilinear Interpolation + yrcbc filtering	32.45
High Quality interpolation	33.39
High Quality interpolation + yrcbc filtering	34.93
Excite-Squeeze No hidden layers	30.20
Excite-Squeeze 1 Hidden Layer	32.19
Excite-Squeeze 2 Hidden Layers	31.87
Resnet152 + Squeeze CNN	31.01
Autoencoder + Resnet152+ Squeeze CNN	30.32

The following figures show a qualitatively comparison of a ground truth image example vs the demosaicing methods shown in table I. Is the opinion of the authors that among the demosaicing method using CNNs, the higher quality image is obtained with the excite-squeeze architecture with one hidden layer.

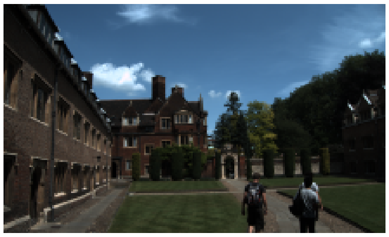


Fig. 10. Ground Truth

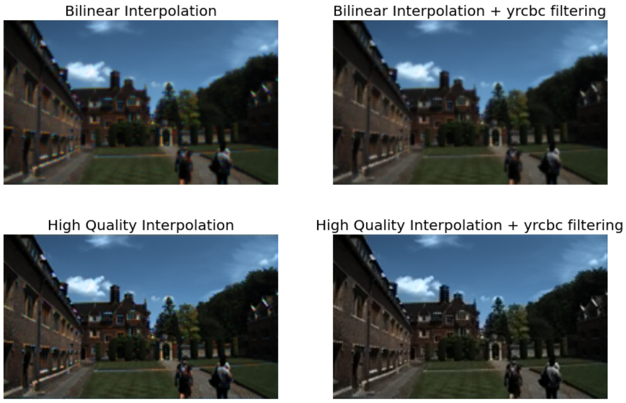

Bilinear Interpolation
Bilinear Interpolation + yrcbc filtering
High Quality Interpolation
High Quality Interpolation + yrcbc filtering

Fig. 11. Demosaiced Image using Linear Filters

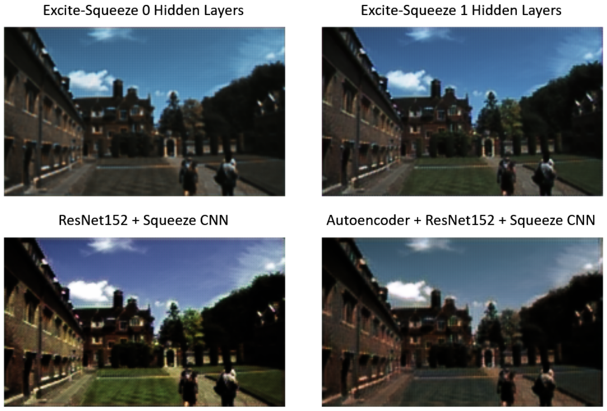

Excite-Squeeze 0 Hidden Layers
Excite-Squeeze 1 Hidden Layers
ResNet152 + Squeeze CNN
Autoencoder + ResNet152 + Squeeze CNN

Fig. 12. Demosaiced Image using Linear Filters

VI. Discussion

Quantitative and Qualitative Results: We evaluate the qualitative performance of our algorithms by the presence of color artifacts, tone of the colors (with respect to the ground truth) and the presence of artifacts in the textures of the demosaiced images. As shown in figure 12 and 11, all the methods tested in this work, both the baselines and the CNNs, generate demosaiced images which display color and texture artifacts in different degrees. Additionally, the tonality of the colors in the demosaiced image diverges from the ground truth image more noticeably in the networks that use transfer learning as part of the architecture. We suspect the last effect is due to the fact that we are feeding a linear image to a network that was trained using gamma corrected images. It is interesting to note that the demosaiced images generated with these methods outperform the bilinear interpolation in terms of the PSNR, even with the evident distortion in the color tones.

Regarding color artifacts and texture artifacts. The best performing CNN is the one corresponding to the excite-squeeze architecture with one hidden layer. The presence of color artifacts in with this method is comparable to the High quality interpolation with filtering of the chrominance channels. We suspect that a deeper network trained over a larger dataset could outperform all, both qualitatively and quantitatively, the baseline methods.

A. Future Work

As far our explorations went, we realized there are different directions we could have taken that would have helped with the demosaicing task. We set out to explore a CFA-agnostic method, which we tried but we generally realized that it was performing quite poorly. These results were what we reported in our presentation work. We however realized that we trained the networks for much fewer epochs and that may have resulted in the performance bottlenecks we saw. Moving forward we would train those models and the ones presented here for a much longer time. Additionally, we would also impose the least norm constraint to enforce that our predicted ground truth images have the same channel values as in the mosaiced 3-channel raw image; we didn't enforce this in any of our current models. Finally, building on top of the work of [2], we would have explored a residual learning architecture, which is known to help with training deeper models. As we observed with the excite squeeze architecture, the deeper network performed the best in the overall set of architectures we tried. We noted through an experimental process that the deeper networks would require more epochs to train to the same validation psnr values hence our suggestion to train our models longer. We would also explore data transformations such as flips and jittering etc. where necessary to increase our dataset size in hopes of allowing our models to better learn the required mapping functions (or distributions). As far as the autoencoding stage is concerned, we would have explored even higher dimensional embeddings in hopes of getting a better decomposition of the pattern before reconstructing into the 3-channel color space images.

[1]

References

[1] B. K. Gunturk, J. Glotzbach, Y. Altunbasak, R. W. Schafer, and R. M. Mersereau, "Demosaicking: Color filter array interpolation," IEEE Signal Processing Magazine, vol. 22, no. 1, pp. 44–54, Jan. 2005.

[2] N. S. Syu, Y. S. Cheng, Y. Y. Chuang, "Learning Deep Convolutional Networks for Demosaicing", Arxiv: Computer Vision and Pattern Recognition. Feb. 2018.

[1] The previous results can be replicated by executing the code in the notebooks: https://drive.google.com/drive/folders/14sGSkADs5vBx6mhNiX3TFB67Zi6oeO32?usp=sharing
You will need to create a direct access of the folder in the root folder of your google drive

[3] D. S. Tan, W. Chen and K. Hua, ”DeepDemosaicking: Adaptive Image Demosaicking via Multiple Deep Fully Convolutional Networks,” in IEEE Transactions on Image Processing, vol. 27, no. 5, pp. 2408-2419, May 2018, doi: 10.1109/TIP.2018.2803341.

[4] Y. Wang, ”A multilayer neural network for image demosaicking,” 2014 IEEE International Conference on Image Processing (ICIP), 2014, pp. 1852-1856, doi: 10.1109/ICIP.2014.7025371.

[5] K. Cui, Z. Jin and E. Steinbach, ”Color Image Demosaicking Using a 3-Stage Convolutional Neural Network Structure,” 2018 25th IEEE International Conference on Image Processing (ICIP), 2018, pp. 2177-2181, doi: 10.1109/ICIP.2018.8451020.

[6] D. Khashabi, S. Nowozin, J. Jancsary, and A. W. Fitzgibbon, “Joint demosaicing and denoising via learned nonparametric random fields,” IEEE Transactions on Image Processing, vol. 23, no. 12, pp. 4968–4981, 2014.

[7] M. Gharbi, G. Chaurasia, S. Paris, and F. Durand, “Deep joint demosaicking and denoising,” ACM Transactions on Graphics, vol. 35, no. 6, pp. 1–12, 2016.

[8] Zhao, Hang, et al. “Loss Functions for Neural Networks for Image Processing.” CoRR, abs/1511.08861, 2015. doi: http://arxiv.org/abs/1511.08861.

[9] H. Malvar, L. He, R. Cutler. ”High-quality linear interpolation for demosaicing of Bayer-patterned color images”, International Conference of Acoustic, Speech and Signal Processing, May 2004

[10] P. Lou, M. Zhang, Z. Ghassemlooy, D. Han, ”Experimental Demonstration of RGB LED-Based Optical Camera Communications”, IEEE Photonics Journa, October 2015