Benefits of Using Linear and Log RGB Data as Inputs to Deep Networks for Computer Vision Tasks

Kexin (Heather) Hao
DREAM research program
8/17/2023

Abstract

In recent years, the field of computer vision has witnessed remarkable advancements, largely attributed to the utilization of deep neural networks. As researchers continue to explore avenues to enhance the performance of these networks, the role of input data representation has gained prominence. This paper investigates the benefits of employing linear and logarithmic RGB data transformed from readily accessible sRGB color space images in the JPEG file format as inputs to deep networks for computer vision tasks. Drawing inspiration from previous work that highlights the potential advantages of logarithmic RGB data, we present a comprehensive study encompassing experimental methodologies, encountered challenges, and insights gained during the research process. Our findings shed light on the intricate interplay between data representation, neural network architecture, and preprocessing techniques, offering valuable insights for researchers and practitioners in the field.

1. Introduction

Deep neural networks have emerged as cornerstones of modern computer vision applications, achieving remarkable feats in image classification, object detection, and more. A critical aspect influencing the performance of these networks is the choice of input data representation. An extensively studied and important aspect is the retention of physics in the input data. The principles governing the phenomenon of reflection and the process of capturing images through sensors adhere to well-established laws of physics that have undergone thorough examination in the domains of computer vision and related disciplines. Foundational research has focused on the representation of surface appearances and the analysis of reflective attributes.[1] Subsequent investigations have delved into leveraging the constraints and frameworks derived from physics to partition images, classify objects, detect highlights, compute lighting conditions, and discern shadows.[2] All these works underscore the significance of physics presented in images for completing various computer vision tasks. As a result, linear and logarithmic RGB data, which preserve the physics of the world, offer significant potential benefits in improving the network's ability to learn meaningful features and generalize to diverse scenarios. For example, a 2020 study by Chen demonstrated that using linear RGB data as input to their network can lead to improved performance in recovering details from low-light scenes.[3]

Capturing images in RAW format is a relatively straightforward process, and both Android and Apple phones possess the capability to capture and process RAW images.[4] However, a challenge arises when using RAW images in deep learning for computer vision. Many prevalent datasets utilized for deep learning in computer vision consist solely of preprocessed images, lacking access to the original linear data. This situation is common across various widely used datasets, including ImageNet[5], COCO[6], Pascal VOC[7], Faces in the Wild[8], and Intrinsic Images in the Wild[9]. Even datasets designed for specific tasks, such as highlight detection, which were historically tackled using physics-based approaches, resort to assembling datasets from web-acquired images.[10] Given these circumstances, there exists a potential advantage in transforming processed images into the linear RGB or log RGB color spaces, which could lead to performance enhancements in computer vision tasks where raw images are not readily available. Inspired by previous work that underscores the potential benefits of log RGB data,[17] we embark on a research journey to comprehensively investigate these advantages.

In this study, I delve into the exploration of using linear and logarithmic RGB data transformed from JPEG sRGB images as inputs to deep neural networks for computer vision tasks. Through iterative experimentation, architectural refinements, and considerations in data preprocessing, I aimed to achieve the best results in the CatsOrDogs task across all three data formats. The rationale behind selecting the CatsOrDogs task and dataset stems from the fact that the images within the dataset are captured under various lighting conditions. This selection offers a unique opportunity to gain valuable insights into the practical implications of employing different data representations, which either preserve or discard physics-based features present in the images, such as lighting and shadows.

2. Related Work

Previous research has demonstrated that utilizing linear RGB or log RGB transformation from raw data enhances the performance of a deep network in detection tasks when employing the same network architecture on images of differing formats. The same research team also suggested that log RGB data retains consistent structural information that might not necessarily be present in data converted to compressed formats, such as JPEG sRGB.[17] Moreover, log RGB data empowers standard convolution layers to compute pixel ratios, which have proven useful as illumination invariant features of objects.[11] While these findings provide the foundation for our investigation, our study aims to extend and validate these concepts even when the original dataset is in JPEG sRGB format, as opposed to raw data.

3. Methods, Experiment, and Results

In pursuit of our research objectives, we centered our focus on a specific image classification task known as "CatsOrDog."[12] The CatsOrDog task entails distinguishing whether an animal depicted in an image is a cat or a dog. It comprises an original dataset in sRGB color space, presented in jpeg format. This setup offers a pertinent context for evaluating the performance enhancements derived from employing linear and log RGB data transformations on the original sRGB color space images stored in jpeg format.

3.1. Data Preparation

Our experimentation involves the utilization of a CNN model[12] as our initial framework. We meticulously processed the dataset, first transforming the sRGB color space into a linear color space in PNG format using the Magick tool. Subsequently, the linear data was further converted into log data in .exr format. This transformation was achieved by utilizing OpenCV to load the linear data, adjusting the color space from BGR to RGB, performing geometric resizing to 224 x 224, and subsequently saving the loaded image data (224x224x3)[13] in log color space within the .exr file format.

Geometric resizing varies for the log data compared to the other two types of data. Most convolutional neural networks are accustomed to normalized or scaled images (x), often undergoing a transformation of the form:

$$x' = (x - \text{mean}) / \text{stdev}$$

This operation centers the values around zero, with the majority falling within the range of -2 to 2. Another common scenario involves data normalized to the range of 0 to 1:

$$x' = x / x\_max$$

For the linear and JPEG data, the built-in resize method of the transforms class from torchvision is used during the experiment. However, applying similar transformations to log images disrupts the inherent relationships and yields unpredictable outcomes. Therefore, geometric resizing for log data should be conducted in linear space before transforming to log space, as resizing frequently involves pixel value interpolation – an operation best suited for linear data. Subsequently, before saving the image, the loaded image data also needs to be cast to float32 format because OpenCV reads images in float64 format which is not supported by PyTorch. After the transformations, the image data is saved in .exr format. Consequently, we have three datasets (Figure 1) — linear RGB, log RGB, and original JPEG data — to facilitate a comprehensive performance comparison in the CatsOrDog task.

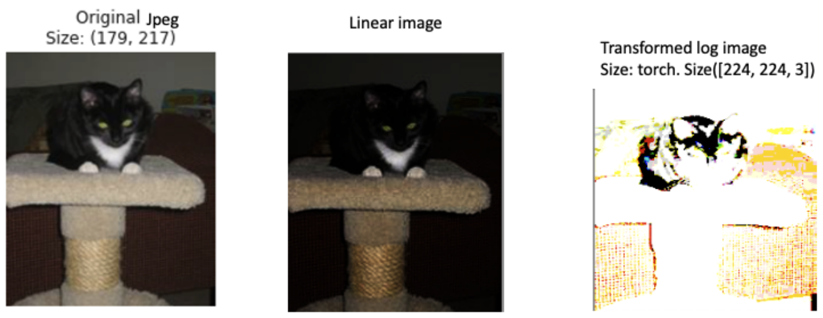

Figure 1. An example of three preprocessed image data inputs.

Following the data transformation, the sRGB and linear data are loaded using the ImageFolder function in PyTorch, as .jpg and .png image formats are natively supported. However, loading the log data demands additional steps due to the .exr file format not being natively supported by PyTorch. Hence, using the ImageFolder to load all image files in the folder is not feasible, as it processes images through PIL, which reduces them to 8-bits. Consequently, a custom dataset must be created, implementing the PyTorch Dataset class and employing the OpenCV library. This approach allows the log image data to be loaded into a float32 torch tensor, enabling the compilation of all .exr images into a PyTorch-compatible dataset for training and testing purposes.

3.2. Object Detection Experiment

The CNN structure used for the CatsOrDogs task consists of three convolutional layers to extract features, along with two linear layers that include a ReLU activation in between for classification purposes. The initial two convolutional layers employ 3x3 filters, while the last convolutional layer adopts 5x5 filters, designed to extract complex features comprehensively. The convolutional network consists of 3, 64, and 512 channels, respectively. Each convolutional layer is succeeded by ReLU activation and a 2x2 max pooling layer. Following the third max pooling layer, a dropout layer is introduced, employing a dropout rate of 0.3 to mitigate overfitting. The final pooling layer possesses 512 channels and is fully connected to a 512-node linear layer after flattening. The network is trained using CrossEntropy[14] as the loss function and Adam as the optimizer[15]. The learning rate for the log-based network was set at 1e-3, maintaining consistency with the training of the other two data types. For each training iteration, 4000 cat images and 4000 dog images are utilized, while 1000 images of each species are reserved for validation purposes. The selection of appropriate test data formats emerged as a significant consideration. Consistency in data format between the training and testing data proved vital in achieving accurate results during the experiment. To uphold this data consistency, a uniform data format is employed throughout both the training and testing phases of the network training process. For instance, if log data is used for training the network, log data is also employed for testing the network's accuracy.

3.3 Result

Table 1 displays the resulting training and testing accuracy for the model training with JPEG (2a), linear RGB (2b), and Log RGB (2c) data. Figures 2a, 2b, and 2c illustrate the evolution of training and testing accuracies as well as losses throughout the entire 25 epochs of the training process. In all experiments, we conducted network training at least twice and presented results from the iteration with the best validation accuracy.

Set	Number of Epochs	JPEG	linear RGB	Log RGB
Train	25	93.9%/0.1446	94.16%/0.1388	98.36%/0.0457
Test	25	89.68%/0.27	93.8%/0.1612	91.67%/0.3046

Table 1. Accuracy/Loss for the JPEG (2a), linear RGB (2b),
and Log RGB (2c) networks for both train and test sets

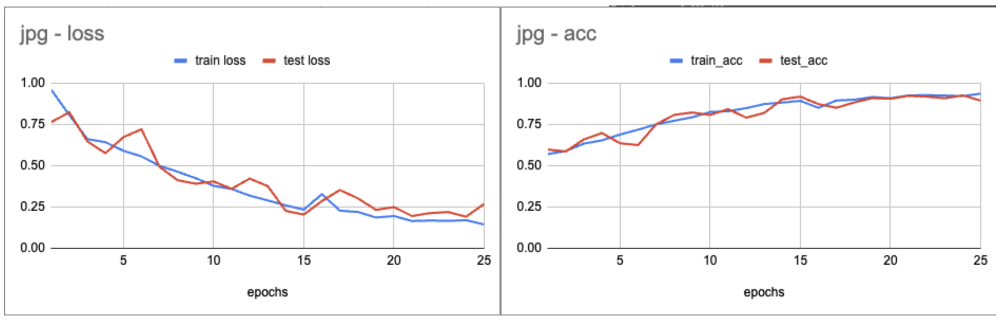

Figure 2a. Training curves for the JPEG network
showing loss and accuracy for the train and test sets.

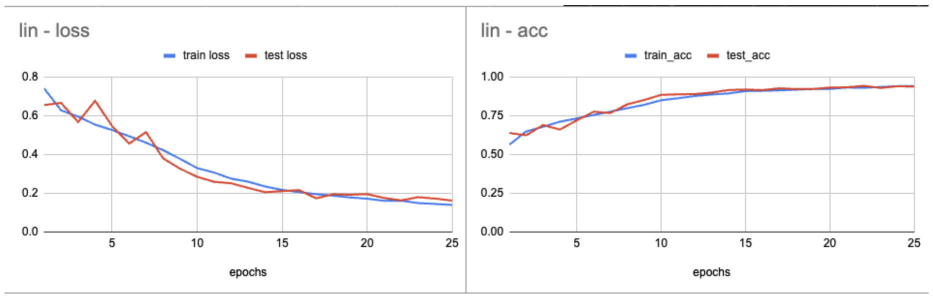

Figure 2b. Training curves for the linear network
showing loss and accuracy for the train and test sets.

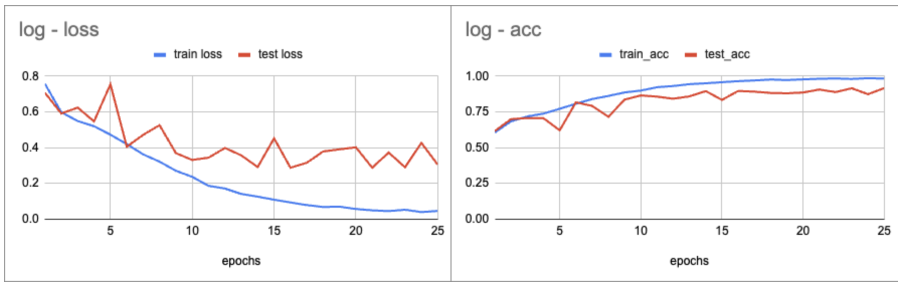

Figure 2c. Training curves for the log network
showing loss and accuracy for the train and test sets.

As presented in Table 1, utilizing log data during both training and testing yielded the highest training accuracy at 98.36%, albeit the test accuracy is approximately 2% lower than that achieved by the linear network, which exhibited an impressive 93.8% accuracy. Notably, both networks trained with linear and log data demonstrated higher training and testing accuracy than the network trained with sRGB data. Another intriguing observation is that employing log data during both training and testing led to a higher degree of overfitting, with the testing accuracy being approximately 7% lower than the training accuracy. In comparison, the network trained with sRGB data exhibited a difference of 4%, while the linear data-trained network displayed a difference of 0.4%.

These outcomes align with the hypothesis that using linear and log data transformed from sRGB data provides advantages to neural network performance, particularly in terms of testing and training accuracy, even when the original data is already in a compressed form. However, the anticipated result that log data, which retains the most physics features, would yield the highest performance was not achieved.

4. Discussion and Conclusion

Our results and analysis support the hypothesis that using linear or log data enhances performance in an object recognition task even when the data is transformed from compressed JPEG format rather than RAW data.

CNNs are designed to process data with a grid-like topology, such as images composed of pixels arranged in a 2D grid. The architecture of a CNN includes several layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers detect features in the input image by applying learnable filters. These filters convolve with the input image to generate feature maps, which are then processed through pooling layers to reduce spatial dimensions and retain salient features. Fully connected layers process pooled feature maps to produce final outputs, such as classification labels. As the original model proved overly complex in feature detection and extraction, I reduced the total number of convolutional layers and introduced an additional linear layer, resulting in a significant 6% increase in training performance.

The log network displays a degree of overfitting as it excels in learning the training dataset but performs less effectively on the testing dataset. PyTorch's DataLoader is employed to enable data shuffling, aiding in reducing overfitting. Furthermore, overfitting is mitigated by employing logarithmic test data (.exr files) for models trained on logarithmic training data, while linear data (png files) and sRGB jpg files are better suited for other training scenarios. Additionally, dropout regularization is utilized to adjust the network, aiming to decrease variance between validation and training performance. However, employing a dropout layer with a dropout rate exceeding 0.3 results in diminished training and testing accuracies, indicating excessive neuron deactivation.[16] Moreover, data augmentation is employed not only for equitable experimentation but also to augment the training dataset, preventing overfitting and promoting generalization.

In future work, overfitting could potentially be further diminished by experimenting with different optimizers (e.g., AdamW) and tuning hyperparameters. Previous research suggests that a smaller learning rate can yield more consistent log network training,[15] and adjusting default hyperparameters (e.g., epsilon) for ADAM can aid convergence for log networks.[17] While not attained in this research, I believe that resolving overfitting would likely enable log data to achieve the highest training and testing accuracy.

Ideally, working with raw image data, devoid of compression artifacts and retaining original color information, would offer a more accurate representation of underlying data characteristics. Unfortunately, obtaining datasets with raw images proved challenging. Raw image datasets suited to our research objectives were not readily accessible, limiting our ability to fully exploit the potential benefits of our proposed data representation. Exploring alternative techniques for data compression or preprocessing to minimize information loss could further enhance the accuracy of our findings. Despite this limitation, our study yields meaningful advancements and insights within the confines of the available dataset.

In conclusion, our research highlights the advantages of utilizing linear and log RGB data transformed from compressed images as inputs for deep neural networks in computer vision tasks. Using linear or log RGB images that preserve the principles of the physics of reflection can simplify certain visual tasks and increase robustness to specific types of visual variation. While restricted by model overfitting and dataset limitations, our study provides valuable insights into the interplay between data representation, network architecture, and preprocessing techniques. As the field of computer vision continues to evolve, our findings serve as a foundational step for future inquiries, inspiring researchers to explore novel strategies for leveraging data representations to achieve enhanced outcomes in deep learning.

8. References

1. B. V. Funt and M. S. Drew. Color space analysis of mutual illumination. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* 15(12):1319–1326, 1993.
2. G. D. Funka-Lea and R. Bajcsy. Combining color and geometry for the active, visual recognition of shadows. *In Proc. Fifth Int'l Conf. on Computer Vision*, Cambridge, 1995.
3. Chen, Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. *In Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3291-3300. 2018.
4. Apple. About apple proraw, 2022. URL https://support.apple.com/en-us/HT211965.
5. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *In IEEE conference on computer vision and pattern recognition.* IEEE, 2009.
6. Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. *Microsoft COCO: Common Objects in Context,* February 2015. URL http://arxiv.org/abs/1405.0312. arXiv:1405.0312 [cs].
7. Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision,* 88(2):303–338, June 2010. ISSN 1573- 1405. doi: 10.1007/s11263-009-0275-4. URL https://doi.org/10.1007/s11263-009-0275-4.

8. Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. *Technical Report 07-49*, University of Massachusetts, Amherst, October 2007.
9. Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. Graphics (SIGGRAPH)*, 33(4), 2014.
10. Gang Fu, Qing Zhang, Qifeng Lin, Lei Zhu, and Chunxia Xiao. Learning to detect specular highlights from real-world images. I*n Proceedings of the 28th ACM International Conference on Multimedia,* MM ’20, pages 1873–1881. Association for Computing Machinery, 2020.
11. X. Wang, G. Doretto, T. Sebastian, J. Rittscher, and P. Tu. Shape and appearance context modeling. *In International Conference on Computer Vision*, 2007.
12. https://www.kaggle.com/code/tirendazacademy/cats-dogs-classification-with-pytorch, accessed by 8/16/2023.
13. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." *Communications of the ACM* 60.6 (2017): 84-90.
14. Zhang, Zhilu, and Mert Sabuncu. "Generalized cross entropy loss for training deep neural networks with noisy labels." *Advances in neural information processing systems* 31 (2018).
15. Dokkyun Yi, Jaehyun Ahn, and Sangmin Ji. An Effective Optimization Method for Machine Learning Based on ADAM. *Applied Sciences*, 10(3):1073, January 2020. ISSN 2076-3417. doi: 10.3390/app10031073. URL https://www.mdpi.com/2076-3417/10/3/1073. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.
16. Pauls, Alexander, and J. Yoder. "Determining optimum drop-out rate for neural networks." *Midwest Instructional Computing Symposium (MICS)*. 2018.
17. Dr. Maxwell’s BMVC 2023 paper