Clay-Wolkin Fellowship ISP: Mid-Year Report

Nick Draper, Christine Goins, Kaveh Pezeshki, Zunyan Wang, Nancy Wei

November 2017

Contents

List of Figures

List of Tables

Abstract

Chapter 1

Introduction

Chapter 2

Project Goals

2.1 Constraints

2.2 Hardware Specification

Chapter 3

Background Information

3.1 Image Signal Processing

3.1.1 Typical ISP Pipelines

3.2 Image Recognition via CNN

Chapter 4

Microarchitecture Specifications

Chapter 5

ISP Pipeline Testing

5.1 Testing Goals

In order to establish a pipeline specification for the ISP, it was first necessary to discover what pipeline steps were necessary to achieve adequate machine learning performance, and furthermore, what algorithms were readily available, and easy to implement in hardware, to accomplish these steps. Our required statistic was 90%+ recognition through all image sets.

5.2 Background

5.2.1 RAW Conversion

In industry, tools such as Adobe Photoshop and Lightroom are often used to edit and convert RAW image formats to standardized formats such as PNG and JPG[1]. However, these products are 'black boxes' and therefore inadequate for the Fellowship's approach to optimizing an ISP for machine learning.

The team therefore investigated the open-source tools UFraw[2] and DCraw [3]. While both tools are open-source and provide image conversion and editing capabilities, UFraw provides a powerful and user-friendly wrapper to the DCraw engine. These tools provided a simple way to fine-tune and convert RAW images for machine learning.

Research on the image quality - machine learning performance correlation already exists[4], and indicates that image deviations that can be compensated for in post-processing have substantial impact on recognition rate. The team therefore decided to proceed to test image optimization via UFraw and DCraw.

[1] https://digital-photography-school.com/raw-workflow-a-pros-approach/
[2] http://ufraw.sourceforge.net/
[3] https://www.cybercom.net/ dcoffin/dcraw/
[4] https://arxiv.org/pdf/1604.04004.pdf

5.2.2 Metric Calculation

In order to judge the efficacy of image optimization, it was first necessary to establish a metric. Focusing on the Inception-V3 image recognition model, trained on ImageNet[5] and implemented on Google's Tensorflow CNN framework[6], the team decided on a simple binary calculation scheme.

Each image set would focus on a specific keyword, with images of the keyword's subject as well as a distribution of images of related and unrelated entities. The metric would be equal to:

$$\text{metric} = \frac{\text{Number of correct top-5 positive identifications}}{\text{number of images}} + \frac{\text{Number of correct negative identifications}}{\text{number of images}}$$

This calculation, as well as an automated image loading and output parsing script, are present in a set of Python scripts. These are available at a Github repository[7] as well as in Appendix C.

5.3 Image Set Acquisition

5.3.1 Image Set 1: Cats

The initial keyword chosen was 'cat'. This was due to the large proportion of cats and related cat breeds in the ImageNet database (CITATION REQUIRED), as well as similar quadripedal animals. These characteristics of the ImageNet database would provide a large amount of granularity in the Inception-V3 output. This granularity would allow for a greater level of fine-tuning in ISP pipeline optimization.

There is an animal shelter near Harvey Mudd College, Priceless Pet Rescue[?]. The initial image set was composed of 179 RAW images of cats and dogs under CFL-lit conditions. There were (X) dogs and (X) cats in this image set. Images were taken in simultaneous RAW-JPG mode with a Canon Rebel t30i, as (ADD IMAGE DETAILS) pixel resolution. (ADD INFO ON CAMERA SETTINGS)

ADD EXAMPLE IMAGES

5.3.2 Image Set 2: Cats

Realizing that a set of images taken under optimal, well-lit, conditions are likely not representative of images in potential applications of the team's ISP, the team elected to take another set of images at the Priceless Pet Rescue. There were (X) number of cats and (X) number of dogs in this image set for a total of (X) images. Pictures were again taken in simultaneous RAW-JPG mode on a Canon Rebel t30i, except with unoptimal ISO, shutter speed, camera shake, and other parameters.

ADD EXAMPLE IMAGES

[5] http://www.image-net.org/

[6] https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwiLjZT008LXAhXoqlQKHXg

[7] https://github.com/Arcturus314/tensorflow_loader

5.3.3 Image Set 3: Squash

The third and final image set was taken with an unrelated subject –squash– to help ensure that identification rates were not directly correlated with the cat subject. This image set was taken in simultaneous RAW-JPG mode on a Canon Eos 80D, under outdoor nighttime conditions. Negative samples were of street signs, food products, and other similar objects.

ADD EXAMPLE IMAGES

5.4 Experimental Design

The team had two goals in pipeline testing: to ensure that it was possible for the open-source DCraw engine to provide acceptable recognition rates (90%+), and to minimize the number and complexity of image processing steps required to achieve these recognition rates.

5.4.1 Overall Parameter Testing

The first step in this investigation was to examine whether the DCraw engine can match the Inception-V3 binary recognition rate of the integrated camera ISP, and, if so, the parameter distribution required to do so.

For each image set, 46 (CHECK) converted sets were generated by UFraw, each with a different set of parameter distributions. The settings of each UFraw parameter are shown in Figure 5.4.1, with all parameters not described by the given settings left at their defaults, such that an image set exists for every parameter value. A parameter description can be seen on the UFraw man page[8].

A full parameter command list is available in Section 7.5.

After image set generation was completed, the Tensorflow test pipeline scripts in Section 7.4 were used to generate output metrics for each set of UFraw parameters.

5.4.2 Pipeline Step Testing

After image generation via UFraw, the team's next goal was to minimize the pipeline steps required to achieve adequate recognition rates. These tests were run without the UFraw frontend, as the raw DCraw engine provides much more control over applied ISP steps.

The team first eliminated all DCraw steps requiring external files, such as dark frame correction. As shown in the previous parameter testing, DCraw conversion pipeline is capable of producing images with adequate recognition rates without these image processing steps.

The condensed DCraw pipeline is shown in Figure 5.4.2. These tests were run on the third image set with keyword squash.

[8] https://www.freebsd.org/cgi/man.cgi?query=ufraw&manpath=FreeBSD+Ports+7.0-RELEASE

Figure 5.1: UFraw Parameter Test Settings

temperature	green	gamma	exposure	saturation	black-point
2700	0.2	0.1	-3	0	0
3000	0.6	0.3	-2	1	0.2
3300	1	0.5	-1	2	0.4
3600	1.4	0.7	0	3	0.6
3900	1.8	0.9	1	4	0.8
4200	2.2		2	5	1
4500			3	6	
4800				7	
5100				8	
5400					
5700					
6000					
6300					

Figure 5.2: DCraw Pipeline

Conversion Step	Algorithm Options	Command-Line Arguments
Decode raw data	camera-specific	n/a
Set up gamma curve	universal	n/a
White balancing	auto	-a
	camera statistics	-w
	default statistics	n/a
	none	-r 1 1 1 1
Colorspace conversion calculation	camera-specific	n/a
Wavelet noise removal	universal	-n
Color scaling	universal	n/a
Interpolation	bilinear	-q 0
	VNG	-q 1
	PPG	-q 2
	AHD	-q 3
Highlight reconstruction	Clip all	-H 0
	Leave unclipped	-H 1
	Blend highlights	-H 2
	Reconstruct highlights	-H 5

Figure 5.3: DCraw Pipeline Statistics

Conversion Step	Algorithm Options	Recognized Images	Percent Recognition
White balancing	auto	96	94.1%
	camera statistics	93	91.1%
	default statistics	94	92.2%
	none	92	90.1%
Interpolation	bilinear	93	91.1%
	VNG	93	91.1%
	PPG	93	91.1%
	AHD	93	91.1%
Highlight Reconstruction	Clip all	93	91.1%
	Leave unclipped	93	91.1%
	Blend highlights	93	91.1%
	Reconstruct highlights	93	91.1%
Wavelet Noise Removal	No denoising	93	91.1%
	92	91.1%	

5.5 Results and Data Analysis

5.5.1 Overall Parameter Testing

While raw data is available in 7.1, the relevant data can be seen in Figure (FIGURE NAME HERE). It is clear that while parameter-recognition rate relationships differ between image sets, it is possible to achieve 90%+ recognition rates through all image sets, and therefore fulfilled the required statistic.

5.5.2 Pipeline Step Testing

Raw data is available in Figure 5.5.2. All options provide similar percent recognition, all above the 90% threshold. Therefore it is possible to only implement the simplest algorithms in the hardware ISP: no white balancing, bilinear interpolation, no highlight reconstruction, and no denoising.

5.6 Conclusion

The data from both pipeline tests illustrated that it is feasible to base the hardware ISP off of the algorithms found in DCraw. Importantly, recognition rates past the 90% threshold are possible by implementing only the simplest DCraw pipeline, allowing a simpler and more power and area efficient hardware design.

Chapter 6

Spring Plans

Chapter 7

Appendix

7.1 DCRaw Test Data

7.2 Condensed DCraw Codebase

7.3 Verilog ISP Implementation

7.4 Tensorflow Testing Pipeline

7.5 UFraw Test Commands