The Pennsylvania State University

The Graduate School

College of Engineering

SENSOR AWARE MACHINE LEARNING FOR EDGE DEVICES

A Thesis in

Computer Science and Engineering

by

Chris Dong Hyun Kim

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2019

The thesis of Chris Dong Hyun Kim was reviewed and approved* by the following:

Vijaykrishnan Narayanan
Distinguished Professor of Computer Science and Engineering
Thesis Advisor

John Sampson
Assistant Professor of Computer Science and Engineering

Chita Das
Distinguished Professor of Computer Science and Engineering
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School

ABSTRACT

Neural networks have produced breakthroughs in numerous domains, and many owe their success to the availability of large, labeled datasets. These datasets help us solve the problems for which they were designed, but are ineffective at yielding solutions to problems that differ drastically in context (e.g. daytime versus nighttime images). Even if the underlying task (e.g. species recognition) is very similar, the deployment conditions can vary so much that more labeled samples are needed. For instance, even if prior efforts expected that determining the species of an animal likely requires images of the animal at different times of the day, they may not have also considered indoor versus outdoor conditions in the training set that would impact zoo versus domestic versus in-nature classification rates. Despite an increasing number of public datasets, there are always more to be desired, namely more labeled datasets on which to learn for new tasks that are not yet popular enough to have justified the manual efforts.

In this thesis, we synthesize training datasets with awareness to environmental factors, specifically to lighting conditions and multiple viewpoints of the same object. We explore synthesizing the dataset as a mean to circumvent limitations of manually labeling and collecting samples under a larger range of potential environments. We consider how context aware datasets might produce models for achieving best classification accuracy under different deployment conditions, and how the correct model to use can be predicted by an endpoint device. We conclude that awareness of viewpoint matters more than aware of lighting condition, and that ultimately, training for the specific environment conditions produces the best model for that particular environment.

TABLE OF CONTENTS

LIST OF FIGURES v

ACKNOWLEDGEMENTS vi

Chapter 1 Introduction 1

Chapter 2 Background and Related Works 3

2.1 Overview of Neural Networks 3
2.2 Convolutional Neural Networks 4
2.3 Public Datasets 5
2.4 Challenges of Dataset Preparation 5
2.5 Synthetic Datasets 6
2.6 Finetuning MobileNet 6

Chapter 3 Implementation and Evaluation 8

3.1 Synthetic Data Generation on Unity 3D 8
3.1.1 Setting Up Unity 3D Environment 8
3.1.2 Defining Light Intensity Aware Datasets 9
3.1.2 Capturing Images of 3D Models 10
3.2 Results 12
3.2.1 Training Specifications 12
3.2.2 Training Results for Varying Lighting Conditions 13
3.2.3 Training Results for Multiple Viewpoint Setup 14

Chapter 4 Conclusion and Future Works 15

4.1 Comments on Training with Synthetic Data 15

BIBLIOGRAPHY 17

LIST OF FIGURES

Figure 1. Artificial Neural Network Showing Neuron Connectivity 3

Figure 2. MobileNet Layer Architecture 4

Figure 3. Example of MPI Sintel Flow Dataset 6

Figure 4. Classes in Synthetic Dataset 9

Figure 5. Unity 3D Scene Setup for Capturing Light and Dark Datasets 10

Figure 6. Images of Sun Chips in High Intensity and Low Intensity 11

Figure 7. Unity 3D Setup for Capturing Different Viewpoints 11

Figure 8. Images of Cheese Puffs Captured at Different Viewpoints 12

Figure 9. Testing Accuracy for Multiple Viewpoints 13

Figure 10. Testing Accuracy for Multiple Viewpoints 14

ACKNOWLEDGEMENTS

First, all honor and glory to our Lord and Savior Jesus Christ without whom nothing is possible. Thank you, Lord, for your abundant blessings, especially for all the opportunities and people around me.

I would like to extend my sincere gratitude to Professor Vijaykrishnan Narayanan and Professor Jack Sampson for their guidance on this thesis and their mentorship over the years. Also, I would like to thank my friends and colleagues at MDL.

A special shout-out to the goombas, tho.

Finally, I would like to thank my parents, Kyung Soon Kim and Sook Hee Kim, and my brother, Dong Chun Kim, for their love and support.

This work was supported in part by NSF Expeditions in Computing Program: Visual Cortex on Silicon CCF 1317560.

Chapter 1

Introduction

Computer vision encompasses a wide array of tasks, from object tracking to optical flow estimation (which is concerned with object motion), and convolutional neural networks (CNNs) are the driving force behind the advancements in those tasks. FlowNet trained a CNN to predict the optical flow provided two input images [2]. In [3], features learned from VGG, a type of CNN, improved robustness of an existing tracking algorithm. The research on object classification also progressed through neural networks. Notably since AlexNet in 2012, a variety of networks such as GoogLeNet and ResNet competed in the ImageNet challenge, each surpassing the previous year's state-of-the art results in object classification [4, 7, 8]. [1] showed that ResNet can achieve less error than a human on classifying ImageNet categories. Convolutional neural networks grew in complexity over the years – compare the hundreds of layers in ResNet (2015) to less than a dozen layer in AlexNet (2012). With the increasing complexity, CNNs achieved better results, pushing the boundaries of computer vision.

The success of CNNs gave rise to interests in (1) optimizing them and in (2) training them with synthetic datasets. Regarding the first interest, from exploration of network architectures to weight quantization, numerous optimizations exist. Optimizations are often desired on edge devices which have limited resources. Regarding the second interest, substituting real datasets for synthetic ones can save time spent on preparing a dataset. The challenge is creating a synthetic dataset that trains CNNs as well as a real one.

In this thesis, we touched on both interests. Consider an edge device, like the phone, that infers on a trained model. Could available sensor capabilities optimize how models are deployed on edge devices? For one, we explore the potential of improving classification accuracy rate by using light sensors. If we detect that we are in a well-lit environment, could we benefit from inferring on a model that is trained for well-lit environments? To train a CNN, many labeled samples are required. Furthermore, they must meaningfully capture the conditions on which the model will be deployed. Synthesizing our own dataset allowed us to address both needs. In addition, we explore how camera viewpoint affects classification accuracy rate. If we detect that multiple cameras are observing the same object at different viewpoints, could we benefit from selectively choosing which camera to use or not use? With these considerations, we trained a neural network on contrasting lighting conditions, and separately, on various viewpoints, using synthetic datasets.

Chapter 2

Background and Related Works

2.1 Overview of Neural Networks

A neural network is a system inspired by the human brain for making decisions provided an input. The decisions are made by the layers of neurons that compose this system. In the human brain, neurons are cells responsible for transmitting information. Similarly, neurons in a neural network pass information to its neighboring neurons. Each neuron contains properties that affect how it responds to incoming information. Figure 1 shows how a simple neural network might be structured. In the end, the purpose of a neural network is to determine a function, or a model, that maps input to useful output. The input-output mapping is determined through training, which involves observing a lot of data samples from the training set and the validation set. Then, the model can be evaluated on never-before-seen samples from the testing set.

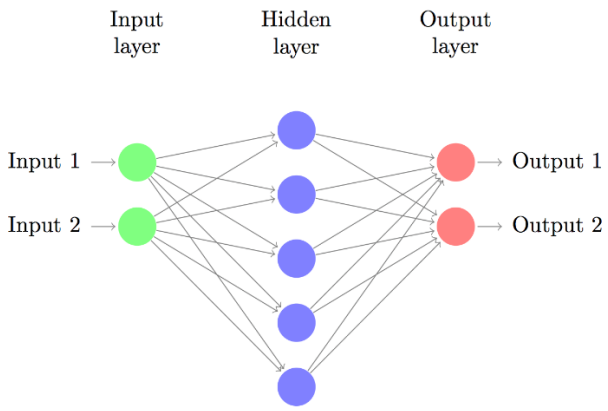

Figure 1. Artificial Neural Network Showing Neuron Connectivity (adapted from [19])

2.2 Convolutional Neural Networks

A convolutional neural network (CNN) is a class of deep (contains multiple layers) neural networks, especially effective in computer vision tasks. It is distinguished by the convolution operation used to learn features about input images. MobileNet [5] is a CNN designed for resource constraint environments like on the mobile and embedded platform. It produces smaller models than other state-of-the art CNNs despite having strong performance on ImageNet classification. We chose the MobileNet architecture in this thesis as we are interested in machine learning at the resource limited, edge devices. The details of the network are provided in Figure 2.

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Figure 2. MobileNet Layer Architecture (adapted from [21])

2.3 Public Datasets

The amount of quality training data is paramount to the success of deep neural networks. As a result, much time is spent on collecting large amount of useful data even before training takes place. It did not take long before research communities recognized the importance of having large datasets available to the public. Consequently, large datasets and competitions around those datasets were born. For example, ImageNet is a dataset containing more than 14 million images. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is a competition for object detection and image classification using the ImageNet dataset [16]. Similarly, Microsoft Common Objects in Context (MS COCO) is another dataset used for object recognition [15].

2.4 Challenges of Dataset Preparation

There are several challenges when collecting training samples the traditional way. First, a large amount of data is needed. Then all those data might need annotation, labels that indicate what the data represents. These tasks are both time-consuming and labor intensive. Furthermore, depending on what the training samples need to represent, labeling can be an impossible or an ambiguous task even for humans. For instance, given a video of human interaction, at what instance could we mark as the beginning and the end of one action? Synthesizing, rather than collecting, new training samples helps alleviate these limitations.

2.5 Synthetic Datasets

To overcome the challenges of manually collecting training data, synthetic datasets have been created. Shown in Figure 3, MPI-Sintel is a dataset derived from an animated short film Sintel, used for optical flow determination [17]. SceneNet is a tool for generating unlimited indoor dataset from open source CAD models [18]. Many of the synthetic data sets were made for different computer vision tasks such as optical flow estimation, camera pose estimation, and stereo disparity estimation. Using synthetic datasets for classification is a topic not explored in depth.

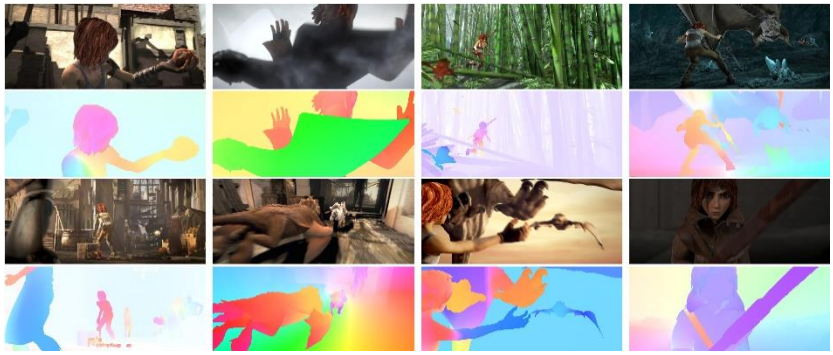

Figure 3. Example of MPI Sintel Flow Dataset (adapted from [17])

2.6 Finetuning MobileNet

MobileNet was trained on ImageNet [5]. To train MobileNet to recognize categories of objects not in ImageNet, the model must be finetuned. This process, known as transfer learning, takes a model trained for one task as the starting point to train for another task. The idea is to apply the knowledge learned in one domain to another domain. Finetuning becomes more effective if the second task is not drastically different from the first task. It saves training time as some features will not have to be learned again. Come implementation, finetuning entails freezing the weights

learned previously on most of the network layers and only learning new weights for the last several layers. The rationale for this is that earlier layers in a CNN learn general features while the later layers learn dataset specific ones.

Choosing how many layers to freeze or train, among other adjustable parameters, is often referred to as an art because how neurons collective learn is not fully understood. Experience goes a long way when deciding what parameters to tweak. To finetune MobileNet on our synthetic dataset, we trained the last 23 layers for 100 epochs. An epoch is one iteration through the training and validation data. Training few layers often resulted in overfitting where the model focused too much on the presented data that it failed to generalize on new input.

Chapter 3

Implementation and Evaluation

3.1 Synthetic Data Generation on Unity 3D

Unity 3D [20] is a game engine used to make industry standard 2D and 3D games. Among countless out-of-the box functionalities that make Unity 3D popular are realistic rendering and the ability to easily manipulate in-game camera through code. Both tools were necessary to synthesize the training datasets.

3.1.1 Setting Up Unity 3D Environment

The world we see through our eyes is made possible by contributions of direct and indirect light rays. Direct light rays refer to light rays that reach our eyes without bouncing off other objects beforehand while indirect light rays refer to those that do bounce off other objects before reaching our eyes. Modeling the interaction of direct and indirect light rays is computationally demanding, so shortcuts are often taken in games. Game engines like Unity 3D make use of various optimizations to achieve realistic lighting simulation at reduced computation cost. In generating the synthetic datasets, settings were chosen to favor realism over performance. For instance, the camera was set to store colors at greater precision, allowing for brighter range of luminance.

3.1.2 Defining Light Intensity Aware Datasets

We define the intensity of light sources in Unity 3D to be low lighting condition if ranging from 5% to 45% and high lighting condition if ranging from 55% to 95%. With this metric, we generated a dark-dataset, a light-dataset, and a combined-dataset. The dark-dataset includes images of objects in the low lighting condition. The light-dataset includes images of objects in the high lighting condition. Lastly, the combined-dataset includes images of objects in both low and high lighting conditions.

The 3D models we used are modeled after items in a grocery store. They were chosen to explore the possibility of applying the trained model on images of real grocery items in the future. Specifically, 30 items, each item representing a category/class, were chosen, they are listed in Figure 4.

Classes in Synthetic Dataset					
1	Barbaras Cheese Puffs	**11**	Lays Dill Pickle	**21**	Old Dutch Restaurant Rounds
2	Doritos Intense Pickle	**12**	Lays Ketchup	**22**	Old Dutch Restaurant Triangles
3	Doritos Zesty Cheese	**13**	Lays Smokey Bacon	**23**	Old Dutch RIPL Lightly Salted
4	Dutch Crunch Jalapeno	**14**	Lays Stax Original	**24**	Old Dutch RIPL Sour Cream
5	GHCretors Chicago Mix	**15**	Lays Stax Sour Cream	**25**	Old Dutch Sour Cream
6	Kettle Cheddar	**16**	Miss Vickie's Original	**26**	Pringles Large Original
7	Kettle Honey Dijon	**17**	Old Dutch Baked Ketchup	**27**	Pringles Small SeaSalt
8	Kettle SeaSalt	**18**	Old Dutch Box Original	**28**	Ruffles BBQ
9	Lays BBQ	**19**	Old Dutch Box RIPL	**29**	Ruffles Regula Nature
10	Lays Classic	**20**	Old Dutch Ketchup	**30**	Sun Chips Cheddar

Figure 4. Classes in Synthetic Dataset

3.1.2 Capturing Images of 3D Models

First, a 3D model is loaded into the scene. Then the camera assumes a new position (a random position inside a predefined boundary shown as a green wireframe cube in Figure 5) facing the object. Lastly, the light intensity of the scene is adjusted to different ranges, after which an image is captured and saved in respective datasets. For the light-dataset, the light intensity was set to a range between 55% to 95% and for the dark-dataset, the light intensity was set to a range between 5% and 45%. The combined-dataset was acquired by sampling from the light-dataset and the dark-dataset. Figure 5 illustrates the Unity 3D scene setup for capturing these images.

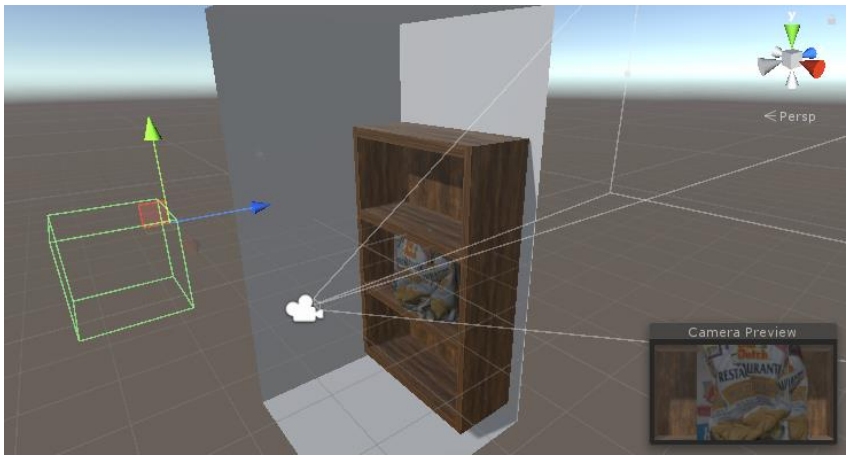

Figure 5. Unity 3D Scene Setup for Capturing Light and Dark Datasets

This process was iterated multiple times to acquire the number of images we needed. Per class, 220 images were taken resulting in 6600 images total. Of the 220 images per class, 30 were used for validation and 20 were used for testing (9 to 1 ratio of training to testing), leaving 170 for training. Figure 6 shows various images of one item in high intensity (left) and low intensity (right) lighting conditions.

Figure 6. Images of Sun Chips in High Intensity (left) and Low Intensity (right)

To generate the datasets to explore how viewpoints affect classification rate, we positioned 12 cameras equally spaced around the object of interest, facing the object. With this setup, each camera will capture a different side of the object. The images captured by each camera was collected as a dataset, so 12 datasets were generated, one from each camera. Figure 8 shows the images captured by each camera for the same product. 220 images per class were captured, where 30 were used for validation and 20 were used for testing. Like before, a combined dataset was acquired by sampling from the 12 generated datasets. The combined dataset serves to train the model under the condition that all the viewpoints of the object are considered. Figure 7 illustrates the Unity 3D setup for creating these datasets.

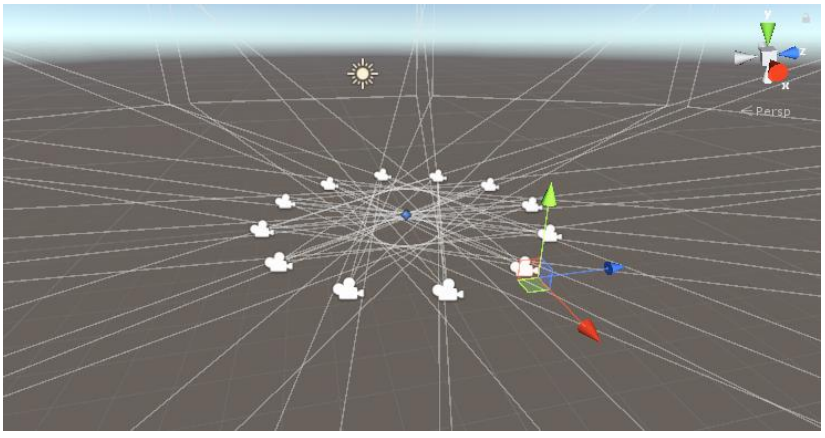

Figure 7. Unity 3D Setup for Capturing Different Viewpoints

Figure 8. Images of Cheese Puffs Captured at Different Viewpoints

3.2 Results

3.2.1 Training Specifications

With the datasets generated, training was done on Power9 with NVIDIA Tesla V100 GPUs, using Keras (version 2.2.2) and Tensorflow (1.10.0). Keras is a python library that offers high level API for using machine learning libraries like Tensorflow. It was chosen for its ease of use in experimenting with neural networks. For example, MobileNet architecture and its weights trained on ImageNet could be instantiated with just few lines of code. Since we only need to classify 30 objects, the last layer of MobileNet was substituted with a dense layer (nodes are fully connected) of 30 nodes configured with Softmax activation. For learning, Adam optimization with learning rate of 0.0001 was used, and the loss function was categorical cross-entropy. As previously mentioned, the last 23 layers were trained for 100 epochs on the synthetic datasets. To facilitate reproducibility with algorithms that rely on pseudo-randomness, the same random seed was used every time.

3.2.2 Training Results for Varying Lighting Conditions

Each model was trained on either the dark-dataset, the light-dataset, or the combined-dataset. Then the models were evaluated on all the test datasets.

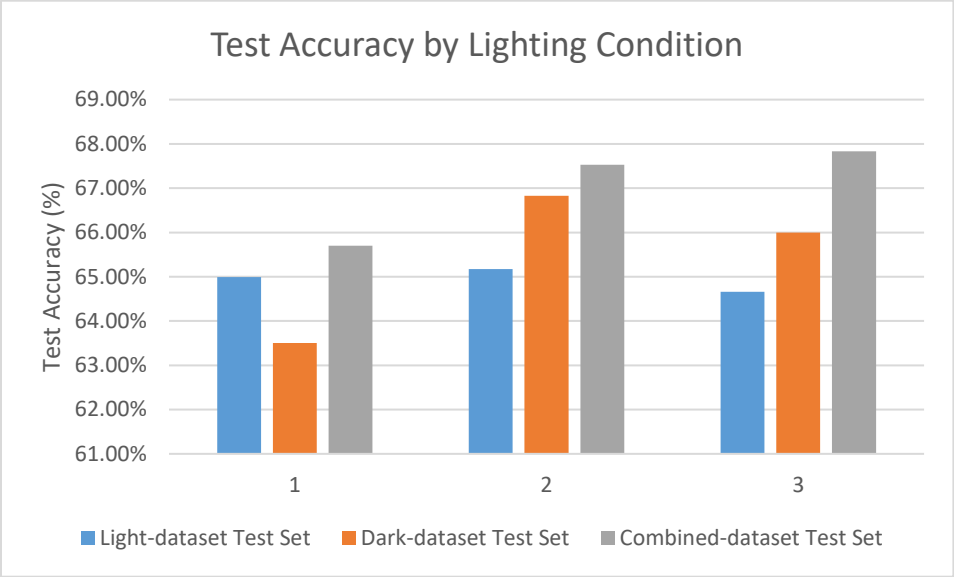

Figure 9. Testing Accuracy for Multiple Viewpoints

Index 1 represents the model trained with the light-dataset. Index 2 represents the model trained with the dark-dataset. Index 3 represents the model trained with the combined-dataset. Following closely to intuition, training on the combined-dataset produced a model that performs well under all lighting conditions (highest grey bar). Also, training on the dark-dataset produced a model the performs best in low light intensity environment (highest orange bar). Though the light-dataset did not perform the best among the others in high light intensity environment, it did compare similarly (only trailing .18%) to the best performing model, which was the dark-dataset model. These results suggest that training specifically for the environmental conditions produces the best model for that environment.

3.2.3 Training Results for Multiple Viewpoint Setup

All in all, 13 models were trained - one model for each camera (represented by indices 1-12) and one (represented by index 13) to represent all the cameras.

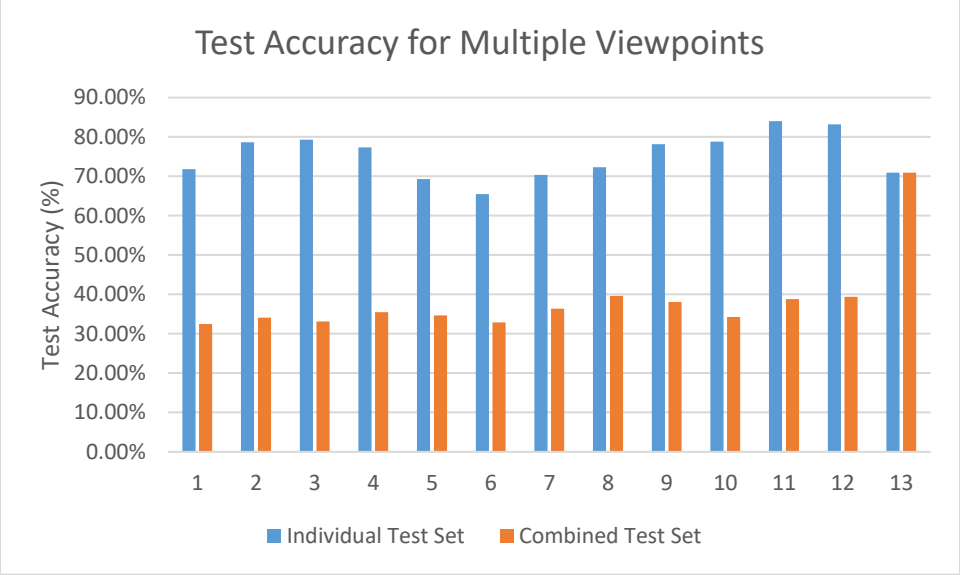

Figure 10. Testing Accuracy for Multiple Viewpoints

The model trained on all the viewpoints achieved an accuracy rate that was roughly 2 times better than any model trained on a single viewpoint. Each model trained for a specific viewpoint only performed well for that particular viewpoint. These results also suggest that training for the environment condition produces in general the best model for that environment. For instance, if an application only requires classifying images at one viewpoint, then the model should be trained using images from that viewpoint. This is supported by the fact that, most cameras (expect for camera 6) achieved similar to higher accuracy rates when compared to the camera considering all viewpoints.

Chapter 4

Conclusion and Future Works

In this thesis, we explored the classification accuracy rate of MobileNet trained on synthetic datasets designed for different lighting conditions and different viewpoints. The objective was to explore how knowledge of the environment lighting condition or the viewpoint of the object can be used to optimize neural network on edge devices. The conclusion was that training models for the environment condition produces the best model for that particular environment. In other words, certain environment conditions strongly influence how the trained model will perform. The results from varying the lighting condition reveal that training for a specific lighting condition does not produce models that perform too differently from one another. Training for specific viewpoints on the other hand, had significant impact on how well the model performed for all viewpoints. This shows viewpoints matter more than lighting conditions when considering these two environmental conditions.

Future works can study how other environmental factors affect the classification accuracy rate. For example, disparity maps and texture maps can be considered, as they can be synthesized through Unity 3D. Then, it would be interesting to explore how existing datasets can be augmented by combining it with a synthetic dataset that is more environment-aware.

4.1 Comments on Training with Synthetic Data

The appeal of synthetic data is that it costs very little to generate a lot of it. Combined with the idea that more training data is better, it is easy to get carried away with generating as much as possible. Though having more data may be better than having less, there are other aspects to

consider when putting together a dataset for training. For instance, does the dataset capture enough variance for the number of classes it embodies? Also, having a large dataset to begin with increases training time. This equates to many hours spent while frequently experimenting with network parameters. We originally generated 1500 images per class for 60 classes. Howeer, training MobileNet on this large dataset was unsuccessful. Rather than spending more time trying different parameters, we trained the model on a simpler dataset. The setup for the simpler dataset was used as a starting point for more complex datasets.

BIBLIOGRAPHY

[1] M. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, B. Esesn, A. Awwal, and V. Asari, “The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," arXiv preprint arXiv:1803.01164, 2018.

[2] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırba¸s, V. Golkov, P. Smagt, D. Cremers, and T. Brox, “Flownet: Learning Optical Flow with Convolutional Networks,” In IEEE Int. Conference on Computer Vision (ICCV), 2015.

[3] J. Czarnowski, S. Leutenegger, and A. Davison, “Semantic Texture for Robust Dense Tracking,” In IEEE International Conference on Computer Vision Workshops (ICCVW), pp. 851–859, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” In Advances in Neural Information Processing Systems 25 (NIPS), pp. 1097–1105, 2012.

[5] A. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, “Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” arXiv:1704.04861, 2017.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, “MobileNet: Inverted Residuals and Linear Bottlenecks,” arXiv:1801.04381, 2018.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” In CVPR, 2015.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” In Proceedings of CVPR, pages 770–778, 2016.

[9] K. Arulkumaran, A. Cully, and J. Togelius. “AlphaStar: An Evolutionary Computation Perspective,” arXiv:1902.01724, 2019.

[10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Driessche, T. Graepel, and D. Hassabis, “Mastering the game of go without human knowledge,” In Nature, 550:354– 359, 2017.

[11] A. O’Toole, P. Phillips, F. Jiang, J. Ayyad, N. P´enard, and H. Abdi, “Face Recognition Algorithms Surpass Humans Matching Faces Over Changes in Illumination,” In IEEE Trans. PAMI, in press 2007.

[12] J. Tremblay, A. Prakash, D. Acuna, M.Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, “Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization,” arXiv preprint arXiv:1804.06516, 2018.

[13] N. Mayer, E. Ilg, P. Fischer, C. Hazirbas, D. Cremers, A. Dosovitskiy, and T. Brox, “What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?” In International Journal of Computer Vision, 2018.

[14] J. Choi, Z. Hakimi, P. W. Shin, J. Sampson, and V. Narayanan, “Context-Aware Convolutional Neural Network over Distributed System in Collaborative Computing,” Accepted to 56th Design Automation Conference (DAC), 2019.

[15] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick, “Microsoft COCO: Common Objects in Context,” In European Conference on Computer Vision (ECCV), 2014.

[16] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, “ImageNet: A Large-Scale Hierarchical Image Database,” In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009.

[17] D. Butler, J. Wulff, G. Stanley, and M. Black, “A Naturalistic Open Source Movie for Optical Flow Evaluation,” In European Conference on Computer Vision (ECCV), pages 611–625, 2012.

[18] A. Handa, V. Patraucean, V. Badrinarayanan, S. Stent, and R. Cipolla, “SceneNet: Understanding Real World Indoor Scenes with Synthetic Data,” arXiv:1511.07041, 2015.

[19] Implementing a Neural Network from Scratch in Python – An Introduction. Retrieved April 08, 2018, from http://www.wildml.com/2015/09/implementing-a-neural-network-fromscratch/.

[20] Unity User Manual (2018.3). Retrieved April 08, 2018, from https://docs.unity3d.com/Manual/index.html.

[21] Transfer Learning using Mobilenet and Keras. Retrieved April 08, 2018, from https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299.