Visual Object Recognition Using Deep Convolutional Neural Network

by

Sabbir Ahmed

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF SCIENCE INCOMPUTER SCIENCE

BRAC UNIVERSITY, DHAKA

SPRING, 2017

Declaration

This is to certify that the research work titled as "Visual Object Recognition Using Deep Convolutional Neural Network" is submitted by Sabbir Ahmed to the Department of Computer Science and Engineering, BRAC University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science. I hereby declare that this thesis is based upon results obtained from my own work. The materials of work found by other researchers and sources are properly acknowledged and mentioned by reference. This thesis, neither in whole nor in part, has been previously submitted to any other University or Institute for the award of any degree or diploma.

Supervisor

Moin Mostakim
Lecturer
Department of Computer Science and Engineering
BRAC University

Author

Sabbir Ahmed
17141013
trickster0179@gmail.com

"If we knew what it was we were doing, it would not be called research, would it?"
— Albert Einstein

Acknowledgements

I would like to express my deepest gratitude to my supervisor Mr. Moin Mostakim (Lecturer, Department of Computer Science and Engineering, BRAC University) for the useful comments, remarks and engagement through the learning process of this work. Furthermore, I would like to thank Dr. Muhammad Abul Hasan (Former Assistant Professor, Department of Computer Science and Engineering, BRAC University) for introducing me to the topic, as well for the guidance on the way. Also, I would like to thank the participants, who have willingly shared their precious time and supported me throughout the entire process, both by keeping me harmonious and helping me to put pieces together. I will be forever grateful to all of your contributions.

Abstract

Visual object recognition has been lying at the convergence point between machine learning, computer vision and AI since the very beginning. From robotics to information retrieval, many desired applications demand the ability to identify and localize objects into different categories. Despite a number of object recognition algorithms and systems being proposed for a long time in order to address this problem, there still lacks a general and comprehensive solution for the modern challenges. Most prominently, new approaches and computational models of vision to analyzing data, such as the convolutional neural networks (CNNs), have enabled a much more nuanced understanding of visual representation. In this paper, I have proposed a deep CNN model to solve the aforementioned problem of object recognition and reported a promising performance on a benchmark classification dataset called CIFAR10.

Keywords: Deep Learning, Convolutional Neural Network, Object Recognition, Data Augmentation.

Table of Contents

List of Tables

List of Figures

1

Introduction

Object recognition [1] is referred as a technology in the field of computer vision for finding and identifying objects in an image or video sequence. Humans recognize a multitude of objects in images with little effort, despite the fact that the image of the objects may vary somewhat from different viewpoints, due to various sizes and scales, or due to translation and rotation. We can even recognize objects when they are partially obstructed from view. But this task is still a challenge for computer vision systems, demanding efficient techniques to the ever growing problems.

1.1 CONVOLUTIONAL NEURAL NETWORKS

Not too long ago, deep neural networks were observed to be the most influential among all innovations in the field of computer vision, generating remarkable performance on image classification. CNNs [2] are particularly intriguing as a tool for studying biological vision since this class of artificial vision systems exhibits visual recognition capabilities that are comparable to those of human observers. As these models improve in their recognition performance, it appears that they also become more effective in predicting, hence accounting for neural responses in human ventral cortex. Recent benchmarks have shown that deep CNNs are excellent approaches for object recognition. Technological developments have even allowed the

use of high-end general purpose Graphic Processor Units (GPUs) [4] for accelerating numerical problem solving using this approach. They resort not only to lower computational time, but also allow considering much larger networks. Hence, computers are now able to drive deeper, wider and more powerful models. State of the art CNNs have achieved human-like performance in several recognition tasks, such as handwritten character recognition, face recognition, scene labelling, image search, image auto-annotation and much more. Thus applying this technique to empirical data bears the potential to demonstrate great promise for enabling future progress in fulfilling the demand of latest visual recognition challenges.

1.2 RESEARCH OBJECTIVE

Many approaches to object recognition task have been proposed as well as implemented over the past few decades. Yet, there still lacks a general and comprehensive solution to the modern recognition challenges, such as the security surveillance domain where number of CCTV cameras is growing exponentially, digital devices that require efficient detection techniques and so on. Google and Microsoft are among the top research companies working in the area. Google's driverless car and Microsoft's Kinect system [3] both use object recognition; yet they are striving for even better and effective techniques. Meanwhile, mobile devices [4] have become powerful enough to handle the computations required for deploying CNN models in near real-time. Keeping that in mind, this research is developed upon proposing as well as implementing a simple yet efficient and powerful approach through using a lightweight CNN scheme for domain specific objection recognition tasks, utilizing minimal hardware resources ideal for low-end devices.

1.3 THESIS ORGANIZATION

This thesis is organized as follows. Chapter 2 presents a description of the related object recognition works. Chapter 3 describes the proposed CNN design for the given problem. Details of the experimental setup are discussed in Chapter 4. The results and its analysis are reported in Chapter 5. Finally, Section 6 summarizes the conclusion of this thesis.

2

Literature Review

There is a broad agreement in the computer vision community about the valuable role that visual objection recognition (VOR) plays in any image understanding task. Numerous research based studies have highlighted unique approaches demonstrating favorable recognition performance under the context of the aforementioned problem. With the discovery of the Scale Invariant Feature Transform (SIFT) [5], multiple opportunities for vocabulary learning techniques have been successfully developed, including Bag of Features (BoF) [6] and Improved Fisher Vector Encoding (IFV) [8] for instances. These techniques are simple yet effective and can be summarized in well defined steps [9]: dense sampling of local descriptors, encoding into a high-dimensional representation and finally pooling to create a single descriptor per image. Despite their simplicity, such methods are hand-crafted and require a certain amount of engineering behind them. These techniques are known as shallow, where the learning is done only at mid-level by training classifiers [4] such as Support Vector Machines (SVM), Random Forest or Naive Bayes classifier.

Deep learning models, such as the CNNs, have become the state of the art for a variety of large-scale pattern recognition problems in the last few years. CNNs are regarded as deep architectures since they involve a hierarchy of layers, such that the outputs of a layer are connected to the next layer's inputs. The exploitation of a large number of layers, up to 22 in

case of GoogleNet model [10] for example, has led to very significant gain in VOR tasks compared to shallow strategies. GoogLeNet, an incarnation of the Inception architecture used for winning the ILSVRC 2014 with a top-5 error rate of 6.7%, was one of the first CNNs that strayed from the general approach of simply stacking convolutional and pooling layers on top of each other in a sequential structure [11]. The main hallmark of this architecture was improved utilization on memory and power usage. To optimize quality, the architectural decisions were based on Hebbian principle and intuition of multi-scale processing. The network comprised of 9 Inception modules in the whole architecture, with over 100 layers in total. There was zero presence of fully connected layers. To go from a 7x7x1024 volume to a 1x1x1024 volume, an average pool was used instead; this saved a huge number of parameters. The network was trained on a few high-end GPUs within a week. During testing, multiple crops of the same image were created, fed into the network and softmax probabilities were averaged to generate the final solution.

In [7], it was proven how a large, deep CNN model called AlexNet is capable of scoring record-breaking results through achieving a winning top-5 test error rate of 15.3% on a highly challenging dataset like ImageNet in the ILSVRC-2012 competition, using purely supervised learning. The network, which had 60 million parameters and 650,000 neurons, consisted of five convolutional layers, some of which were followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. ReLU was used for the nonlinearity functions whereas data augmentation techniques were followed via image translations, horizontal reflections and patch extractions. Through implementing dropout layers, in order to combat the problem of overfitting to the training data, and using batch stochastic gradient descent with specific values for momentum and weight decay, the model was trained on two GTX 580 GPUs

for five to six days. This designed network was used for classification up to 1000 possible categories.

The use of such models for domain-specific and small-scale VOR challenges is an active topic, as deep architectures typically require large-scale datasets and tremendous computational power for their learning.

3

Proposed Framework

In order to tackle an indispensable task as difficult as recognizing real-life objects, I have proposed a simple, yet powerful deep version of CNN which is capable of offering excellent performance while requiring minimal hardware resources and computational costs. The following sections would provide further details for each part of the proposed framework.

3.1 Preprocessing

The input dataset used in this research included common objects such as airplanes, automobiles, birds, cats and so on. The photos were in color with red, green and blue components, measuring 32x32 pixel squares. Regardless of being small, the original aspect ratio was maintained in order to avoid image distortion initially. The pixel values were in the range of 0 to 255 for each of the red, green and blue channels. Because the input values were well understood, I normalized to the range 0 to 1 by dividing each value by the maximum observation, which was 255. The initial data was loaded as integers, so I had to cast it into floating point values in order to perform the division. The output variables were defined as a vector of integers from 0 to 1 for each class. I used one hot encoding [12] to transform them into a binary matrix in order to best model the

classification problem. It was previously known that there were 10 classes for this problem, so I expected the binary matrix to have a width of 10.

To improve the regular performance of my model, image data augmentation was applied. Since the object in the images may vary in their respective positions, a boost in the model performance was likely to be achieved by using some data augmentation. Methods such as random height shifts, random width shifts, horizontal image flips, vertical image flips and small rotations turned out to be quite beneficial in the process.

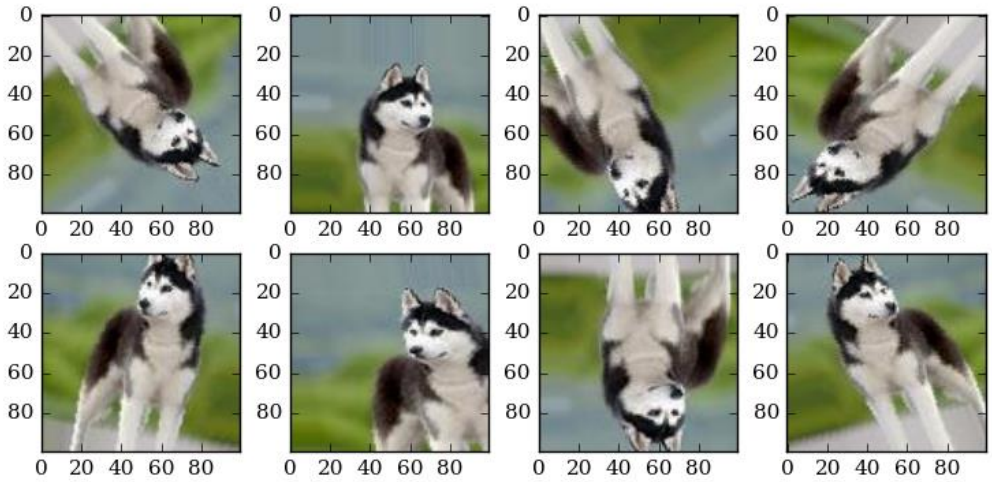

Figure 3.1: Example of distorted images by applying random rotation, shift and flip methods to the training data.

3.2 LEARNING

The worked model bears a deep, albeit light network topology. After seeing that a simple CNN performs poorly on the concerned problem, the consideration for scaling up the size and complexity of the model became a prime necessity. The following part would provide short insights on the core building blocks of my proposed CNN architecture.

Compared to original neural networks, deep CNNs involve a large number of layers. Let us briefly review the different layers involved in my CNN model. Each layer of the CNN is

composed of neurons connected to nodes of previous layers, such that the output at a given node for layer L is a function of outputs of nodes in layer L-1. There are five main types of layers involved in the architecture [4]:

I **Convolution layers:** Convolution layers are characterized by weights (filter values). There exist multiple convolutions per layer with a fixed size, and each kernel is applied over the entire image with a fixed step (stride). The first convolution layers learn the low-level features such as edges, lines and corners. Next layers learn more complex representations (e.g., parts and models). The deeper the network is, the higher-level the learnt features.

II **Pooling layers:** Pooling layers perform a nonlinear downsampling. In this category, there are also several layer options, with maxpooling being the most popular. This basically takes a filter (normally of size 2x2) and a stride of the same length; then applies it to the input volume and outputs the maximum number in every subregion that the filter convolves around. Thus amount of parameters or weights is reduced and overfitting is controlled.

III **Activation layers:** Activation functions mimic the behavior of the neuron's axon that fires a signal when a specific stimulus is presented. Some of the most common activations functions are the Hyperbolic Tangent, Sigmoid and the Rectified Linear Units (ReLU) among others. ReLU has emerged as a key feature of CNN. It is defined as $f(x)=max(0, x)$.

IV **Dropout layers:** Dropout layers [11] "drop out" a random set of activations by setting them to zero in the forward pass. This forces the network to be redundant; meaning, the network should be able to provide the right classification or output for a specific example even if some of the activations are dropped out, which makes sure that the network is not getting too fitted to the training data and thus helps alleviate the overfitting problem.

V **Fully-connected layers:** A fully-connected layer (FC) differs from the above mentioned layers by the fact that all outputs of the previous layer are connected to all inputs of the FC layer. These layers can be mathematically represented by inner products.

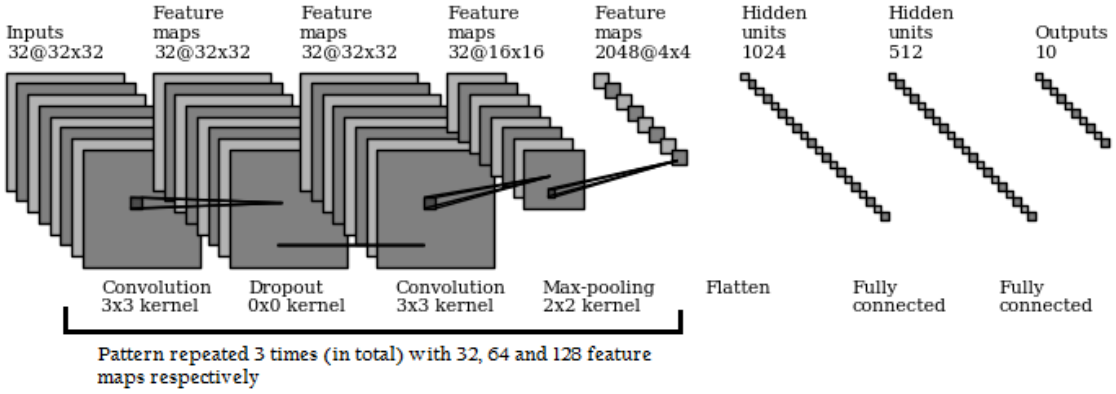

Figure 3.2: Proposed CNN architecture for the VOR task.

3.3 PREDICTION

In an attempt to better translate the large number of feature maps to class values, additional and larger dense layers had to be used at the output end in case of my proposed network architecture. Rectified linear unit (ReLU) activation functions were used for the neurons in such layers. A softmax activation function was used on the output layer to turn the outputs into probability-like values and allow one class of the 10 to be selected as the model's output prediction. Efficient Stochastic Gradient Descent (SGD) optimizer was used to learn the weights and logarithmic loss (cross-entropy error function) was used as the loss function, given by:

$$L(X,Y) = -(1/n)\Sigma(i=1,n)y\hat{}(i)\ln a(x\hat{}(i)) + (1-y\hat{}(i))\ln(1-a(x\hat{}(i)))$$

Here, $X=\{x(1),\ldots,x(n)\}$ is the set of input examples in the training dataset, and $Y=\{y(1),\ldots,y(n)\}$ is the corresponding set of labels for those input examples. The $a(x)$ represents the output of the neural network given input x.

4

Experiments

For experimenting with the proposed VOR framework, it was important to choose a well adjusted dataset, a feasible architectural design built upon suitable hyperparameters and required hardware resources to fit the training process. The detailed experimental procedure, which was followed based on these factors, is provided in the sections below.

4.1 DATASET

The Canadian Institute for Advanced Research's CIFAR-10 [13], an established computer-vision dataset used for object recognition, was used for experimenting in this research. It is a subset of the 80 million tiny images dataset and consists of 60,000 32x32 color images containing one of 10 object classes, with 6000 images per class. It was collected by Alex Krizhevsky, Vinod Nair and Geoffrey Hinton. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. The classes are completely mutually exclusive.

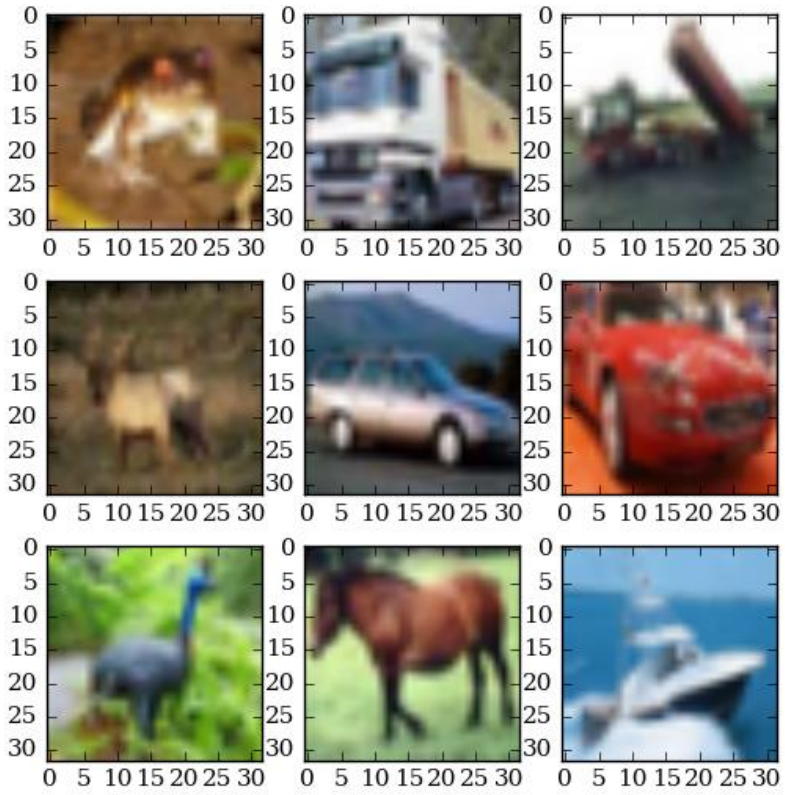

Figure 4.1: Sample images (scaled for improving viewers' experience) taken from the CIFAR-10 dataset.

4.2 ARCHITECTURE SELECTION

Choosing the correct architecture can be a challenging issue for a proposed framework built upon CNN, since the process requires a great deal of tinkering for fine-tuning the hyperparameters. It can be quite difficult to know how many layers to use, what should be filter sizes, correct values for stride, etc. These are not trivial questions and there is no fixed standard set by researchers. This is because, a network will largely depend on the type of data that may in turn vary by size of the image, complexity of given data, available hardware resources and much more. Through looking at the above-mentioned dataset and experimenting between several

hyperparameters, I have chosen the best performing model using right combination that created abstractions of image at a proper scale and exhibited promising outcomes.

The CNN architecture, as proposed for the VOR problem in this research, was comprised of a pattern, in which Convolutional, Dropout, Convolutional and Max Pooling layers were placed sequentially. This pattern was repeated three times with 32, 64, and 128 feature maps. The effect of that was having an increasing number of feature maps with smaller and smaller sizes, given the max pooling layers. Zero padding approach was applied in the Convolutional layers, since it tries to pad evenly left and right when needed; if the amount of columns to be added is odd, it will add extra column to the right, and the same logic applies vertically where there may be an extra row of zeros at the bottom. Finally, additional and large Dense layers were planted at the output end of the network in an attempt to better translate the

Layer type/number	Kernel/Neurons	Activation function	Channels	Output size	Parameters
Convolution 1	3x3	ReLU	32	32x32	896
Dropout 1 (20%)			32	32x32	0
Convolution 2	3x3	ReLU	32	32x32	9248
Max-pooling 1	2x2		32	16x16	0
Convolution 3	3x3	ReLU	64	16x16	18496
Dropout 2 (20%)			64	16x16	0
Convolution 4	3x3	ReLU	64	16x16	36928
Max-pooling 2	2x2		64	8x8	0
Convolution 5	3x3	ReLU	128	8x8	73856
Dropout 3 (20%)			128	8x8	0
Convolution 6	3x3	ReLU	128	8x8	147584
Max-pooling 3	2x2		128	4x4	0
Flatten	2048				0
Dropout 4 (20%)	2048				0
Fully-connected 1	1024	ReLU			2098176
Dropout 5 (20%)	1024				0
Fully-connected 2	512	ReLU			524800
Dropout 6 (20%)	512				0
Fully-connected 3	10	softmax			5130

Table 4.2: Summary of proposed CNN architecture of the model trained on 32x32 colored images.

large number of feature maps to class values. A summary of the established deep network architecture is shown in Table 4.2.

4.3 TRAINING

The discussed model was trained using a logarithmic loss function with stochastic gradient descent optimization algorithm configured with a large momentum and weight decay that started with a learning rate of 0.01. It was fitted with 50 epochs and a large batch size of 64, found through some minor experimentation. Normally, the number of epochs would be one or two orders of magnitude larger for this problem. Each epoch took an average computational time of 145 seconds, causing the entire training phase to finish in approximately 2 hours. These epochs acted like smaller training sessions that ran over all of the data given in training set. During that run, filter values (or weights) were adjusted through a process called backpropagation. It was the part where weights of the layers were correctly tuned with respect to the loss function while carrying out forward and backward passes. The ultimate goal was to achieve a set of parameters which have a certain ability to generalize toward new data. And that ability was reflected by the validation accuracy, which we shall find out in Chapter 5.

Running my proposed model delivered classification accuracy and loss function values on the training dataset through each epoch, where the best achieved classification accuracy without image data augmentation was 95.87% along with a loss function value of 0.1146. However, a light augmentation resulted into having classification accuracy up to 75.13% followed by a loss function value of 0.7016. This training process was performed on a personal computer featuring a dual-core Intel Core i3-5010U CPU at 2.1 GHz, 4GB system memory, powered by a NVIDIA GeForce 920M GPU at 954 MHz. The learning model was developed in

Keras, a Python library for deep learning, using Theano backend and trained over GPU, utilizing cuDNN library and keeping CNMeM enabled with initial size - 60% of memory.

5
Results

This section reports the best accuracy achieved by the proposed model for the VOR task and provides an analysis as well as highlights a list of top recognition rates achieved by professional researchers, followed by a discussion showing some scope for improvement.

5.1 TESTING

The trained model built in accordance to specified configuration, as described in previous section, was tested on the validation dataset to estimate its ability to generalize toward new data, because the validation set contains only data that the model has never seen before and therefore cannot just memorize. Although the model initially achieved a classification accuracy of 81.46% along with a loss function value of 0.7199, data augmentation pushed the classification to reach a baseline accuracy of 77.56% followed by a loss function value of 0.6297. That means, the estimate of classification accuracy for the latter model is slightly below 4 points worse than the previous model.

5.2 ANALYSIS

Taking a closer look at the difference between the classification accuracy and loss function values obtained from training and validation datasets before and after performing image data

augmentation, would tell us that there was bit of an overfitting situation at the former stage. Even though the training data accuracy before augmentation kept improving up to 95.87% while maintaining a loss function value as low as 0.1166, validation data accuracy had gotten lot worse and fell down to 81.46% while loss function reached a surprisingly high value of 0.7099, i.e. the model started to basically just memorize the data. But the second approach, one performed with augmentation of data, controlled overfitting. Training data accuracy after augmentation turned out to be 75.13% while maintaining a loss function value of 0.7016, but validation data accuracy reached 77.56% while loss function value went lower to 0.6297. Even though the accuracy obtained from original dataset came out to be slightly higher than that from augmented dataset, the latter possessed a greater potential of achieving even higher classification accuracy given a few more rounds of training, while keeping overfitting to a minimum at the same time.

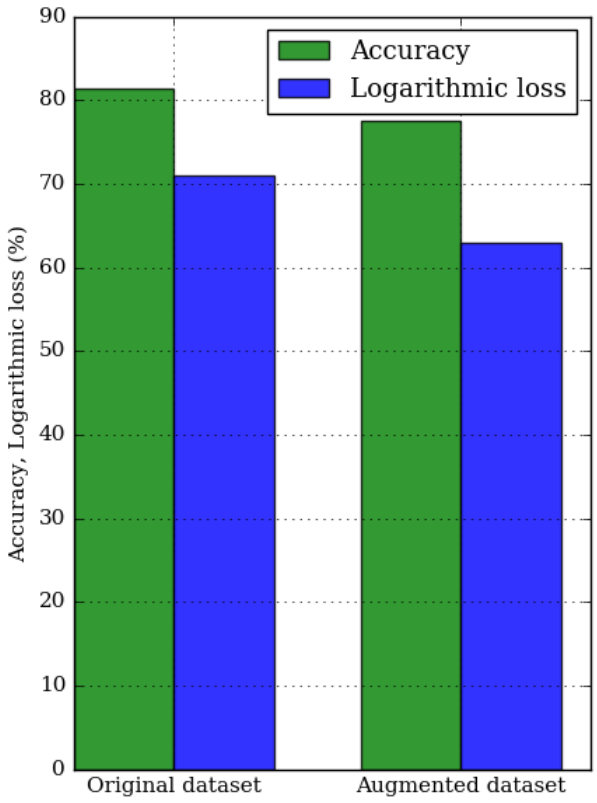

Figure 5.2: Classification accuracy obtained using original and augmented datasets.

Despite achieving good results on this very difficult problem, I am still a few miles away from beating the world's top-notch records. Details [14] of classification performance achieved by some state of the art methods on the CIFAR-10 dataset, is given in Table 5.2.

Test accuracy	Method	Venue	Used data augmentation?
96.53%	Fractional Max-Pooling	arXiv 2015	Yes
95.59%	Striving for Simplicity: The All Convolutional Net	ICLR 2015	Yes
94.16%	All you need is a good init	ICLR 2016	Yes
93.45%	Fast and Accurate Deep Network Learning by Exponential Linear Units	arXiv 2015	No
92.40%	Training Very Deep Networks	NIPS 2015	No
91.73%	BinaryConnect: Training Deep Neural Networks with binary weights during propagations	NIPS 2015	No
90.50%	Practical Bayesian Optimization of Machine Learning Algorithms	NIPS 2012	Yes
86.70%	An Analysis of Unsupervised Pre-training in Light of Recent Advances	ICLR 2015	Yes
82.18%	Convolutional Kernel Networks	arXiv 2014	No
78.67%	PCANet: A Simple Deep Learning Baseline for Image Classification?	arXiv 2014	No

Table 5.2: Details of classification performance on CIFAR-10 by state of the art methods.

There are a lot of decisions to make when designing and configuring a deep learning model such as the one which I have proposed. Most of these decisions must be resolved empirically through trial and error, and evaluating the model on problem data. As such, it is critically important to have a robust way to evaluate the performance of the deep neural networks. Below are a few ways [12] that could have been followed to elevate the proposed model's performance:

I **Train for more epochs:** Current model was trained for a very small number of epochs, 100 to be exact. It is common to train large convolutional neural networks for few hundreds or

thousands of epochs to obtain desired results. State of the art performance can be achieved by significantly raising the number of training epochs.

II **Further data augmentation:** Although augmentation strategies like random shifts, flips and small rotations in training image data were implemented, methods such as feature standardization, ZCA whitening and higher random rotations are yet to be tried out.

III **Deeper network topology:** The network presented in this research was large and deep, but even larger and deeper networks could have been designed for the problem. This may involve more feature maps closer to the input and perhaps less aggressive pooling. Additionally, standard convolutional network topologies that have been shown useful in world class implementations, might have been adopted and evaluated on the problem.

6
Conclusion

Recognition of visual objects is an important, yet challenging vision task. However, it is still an open problem due to the complexity of object classes as well as limitation of computational resources. Using the concept of deep learning, I have demonstrated how a highly accurate VOR pipeline, built upon a deep CNN module, can be expected to be used in near real-time for commercial deployment with minimal hardware requirements. The results yield a solid evidence that the proposed model bears strong potential to be superior in terms of minimal memory storage, computational complexity and recognition performance compared to the existing CNN models for low-end devices. Through applying fine-tuning and data augmentation strategies, a variant of the initially proposed model achieved an error rate that came out to be as low as 18.54%.

6.1 FUTURE WORK

Future work will further investigate optimization for some of the current implementation issues, which would require tweaking with the network hyperparameters with a goal to achieve an even higher accuracy for object classification while maintaining a decent level of computational complexity and recognition performance at the same time. Despite the current work using a GPU implementation and offering favorable outcomes using minimal resources on a personal computer, computational time and memory requirement are yet to be checked

through a mobile GPU implementation. Ongoing research also includes the extension of the proposed model for VOR on mobile devices from RGB-D images, as depth sensors will be embedded in the next generation of mobile devices.

Bibliography

[1] Foundation, W. (2016, November 12). *Outline of object recognition.* Retrieved March 31, 2017, from Wikipedia: https://en.wikipedia.org/wiki/Outline_of_object_recognition

[2] Gauthier, I., & Tarr, M. J. (2016). Visual Object Recognition: Do We (Finally) Know More Now Than We Did?. *Annual Review of Vision Science*, *2*, 377-396.

[3] Rouse, M. (2015, February 28). *What is object recognition*?. Retrieved March 31, 2017, from WhatIs.com: http://whatis.techtarget.com/definition/object-recognition

[4] Tobias, L., Ducournau, A., Rosseau, F., Fablet, R., & Mercier, G. (2016). Convolutional Neural Networks for Object Recognition on Mobile Devices: a Case Study. *International Conference on Pattern Recognition* (pp. 2-7). Cancun: ResearchGate.

[5] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

[6] Csurka, G., Dance, C., Fan, L., Willamowski, J., & Bray, C. (2004, May). Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV* (Vol. 1, No. 1-22, pp. 1-2).

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[8] Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. *Computer Vision–ECCV 2010*, 143-156.

[9] Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

[10] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-9).

[11] Deshpande, A. (2016, July 20). *A Beginner's Guide To Understanding Convolutional Neural Networks.* Retrieved March 31, 2017, from adeshpande3.github.io: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

[12] Brownlee, J. (2016, July 1). *Object Recognition with Convolutional Neural Networks in the Keras Deep Learning Library.* Retrieved March 31, 2017, from Machine Learning Mastery: http://machinelearningmastery.com/object-recognition-convolutional-neural-networks-keras-deep-learning-library

[13] Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.

[14] Benenson, R. (2016, February 22). *Classification datasets results.* Retrieved March 31, 2017, from rodrigob.github.io: http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html