NCS based ultra low power optimized machine learning techniques for image classification

Naman Tiwari
Department of Electrical Engineering
IIT(ISM) Dhanbad
naman.15je001149@ee.ism.ac.in

Koushik Mondal
Computer Centre, IIT(ISM) Dhanbad
koushik@iitism.ac.in

Abstract—In recent years there has been an extensive development in the field of convolutional neural network-based image classification because of the human-like inference results obtained, but these massive networks are resource intensive and have high memory and computational requirements. Intel's Neural Compute Stick brings real time inference, prototyping and deployment of these DNNs to the network edge. In this paper we will discuss the development of a model for classification of book cover images into genres, and subsequently compiling the trained model for use with the Neural Compute Stick, so as to receive the optimized results in constrained environments thus ultimately leading to a system to judge a book by its cover which can be used even within a low power environment like a mobile device or Raspberry Pi, as the stick runs on power values as low as 1.2W.

Keywords—Convolutional Neural Networks, Neural Compute Stick, OpenVINO Toolkit, Deep Neural Network

I. INTRODUCTION

"Do not judge a book by its cover" is something we are told to do, but the cover is the first thing that someone notices about a book and trying to extrapolate ideas about the content cannot be helped. Our primary goal is to classify book cover images into genres without any prior information about the title, author and their previous works etc.

Due to the high accuracy that can be achieved, using convolutional neural networks to classify images has been worked on rigorously in recent years. This surge in popularity of CNNs helped us in obtaining highly efficient pre-trained neural networks for use. We used the concept of transfer learning [1] to train the last layers of these networks over our dataset to obtain the best possible classification results while also reducing the training time from a few weeks to a few hours. Recent implementations of neural networks consist of millions of weights, which in turn, require a training dataset with millions of examples. Transfer learning paradigm helped us here as well, since we are only training the last layers, a few thousand examples should be enough, to achieve decent levels of accuracy.

Motivation behind this project is two-fold:

- First, since we are only using the cover images for inferring the genre it is similar to the situation of a relatively lesser known author releasing a new book and an unbiased reader's perception of the cover has to be known. Thus, the results obtained from the network can be utilized to predict what an uninitiated human would conclude from the cover and use it to design better covers. An interesting future project could be to find if there is a relation between a book's sales and the ease with which its genre could be perceived from the cover.
- Second, the Neural Compute Stick (NCS) [2] is developed by Intel Movidius for computer vision applications at the network edge. The Myriad2 VPU, which powers the stick, consists of 12 Very Large Instruction Word (VLIW) shave cores, 2 CPUs and dedicated vision accelerators all connected by an "intelligent memory fabric" [3] which helps in achieving high performance at low power. This in turn makes it suitable for use with various devices like mobile phones, drones, cameras etc. enabling them to run complex neural network enabled inferential tasks. The NCS would thus also help in making the various components in the Internet of Things and Big Data [4] paradigm be able to run computationally intensive inference tasks. NCS is not dependent on a connection to cloud and can be used for tuning, profiling and deploying deep neural networks without the need for any high-power input.

NCS mainly used as a machine learning accelerator for inference on the edge without any external connected computational networks. It is helpful in creating offline artificial intelligence prototyping having supported by compile, tune, validate and accelerate models . It supports popular Neural Network models like Caffe and TensorFlow, apart from self designed models. The NCS supports OpenVINO toolkit distribution of Intel which helped us to build fast track development and deployment of high performance computer vision and deep learning inference applications on Intel platforms. We used on-chip neural network accelerator for developing the model with the help of OpenVINO toolkit. We used a part of code provided by TensorFlow Slim [5], a lightweight API of tensorflow, with some modifications for re-training purposes. This library also provides implementations of various well-known networks from which we used Inception-V3 [6], Inception-V4 [7] and MobileNet-V2 [8]. Recent developments have made training neural networks more efficient by using Graphical

978-1-7281-0297-9/19/$31.00 ©2019 IEEE

Processing Units (GPUs), leaving the prediction part of the process to be worked upon. NCS is a giant leap forward in the area and opens up a plethora of interesting application possibilities which can be developed for user-centric mobile devices.

II. Related Work

In a recently published work Xiaofan Xu et. al [9] used a CNN for 3D object recognition. They used 3D point cloud occluded volumes depicting real world scenes for training the network and then compared the time taken and power used for running an inference task across multiple devices like the Nvidia Titan X, an older version of the NCS (Fathom NCS [10]) and Intel i7-5930K. As evident from the Inference/Time/Power (inference/millisecond/Watt) values obtained for these devices, they concluded that the Fathom NCS performed significantly better and provided a low-cost & low-power solution for inference purposes. The crucial result obtained from their work is that the Fathom NCS used the least power and completed the inference task at hand reasonably fast without any loss of accuracy that was obtained through other conventional methods.

III. Creating the Dataset

We considered several online book databases like Goodreads [11], OpenLibrary [12] and Google Books [13]. While the image quality for the covers were best in the Google Book's case, but its functionalities are implemented for fetching a list of books created by a particular user. The Goodreads API has no feature to fetch books of a particular genre.

A. Open Library API

The Open Library API can be used to fetch a list of books of a given genre. We used it to fetch a list of Open Library ID (OLID, a unique ID which identifies each book in the Open Library Database) spread over 9 genres of literature. These OLIDs were then used to procure detailed information about each of the books like its cover image, title and author's name. Different genres and their corresponding number of examples are mentioned in Table I:

TABLE I. NUMBER OF EXAMPLES IN THE DATASET FOR EACH GENRE

Genre	Example Count
Biography	2001
Business	2001
Drama	1452
Fantasy	2001
Health	1300
Mystery	1787
Romance	2001
Science Fiction	2001
Travel	2001
Total: 16542	

We collected a total of 16542 cover images for the 9 genres selected and divided them into an 80:20 train and test split. The genres chosen were because these would have a lesser probability of sharing a book and availability large number of books with distinct covers.

B. Preprocessing of Images

We resized every cover image to a size of 400px x 400px for consistency over the dataset. Since, most of the images were significantly smaller than this size we used interpolation, particularly OpenCV's INTER_CUBIC [14], to resize the images. We stored images in two forms RGB and B/W to find out if the neural networks perform better when trained with one over the other. In order to the convert the RGB images to threshold black and white images we used Adaptive Thresholding [15] to obtain the best output. Since, most of the cover images in the Open Library database are contributed by users they have varied lighting conditions and adaptive thresholding helped in obtaining the best output.

C. Filtering out irrelevant images

Since the primary goal of the project is to classify book cover images, images with a low number of features on the cover for instance covers with just text on them had to be discarded. We used the ratio of threshold pixels to the number of pixels as a parameter to decide if the image should be included or not. We performed multiple iterations of testing with different images to decide this threshold pixel count and found that the value 15% yielded the most appropriate balance between discarding the images with less features and not discarding too many images. Thus, images which had at least 15% of pixels below the threshold of 200 were included in the dataset.

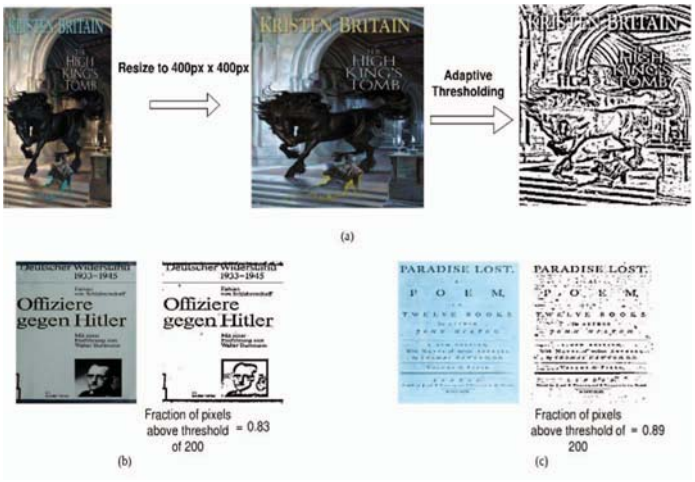

Fig. 1. (a) Steps in preprocessing of the cover images, (b) Cover image with fraction of pixels above threshold of 200 as 0.83 and included in dataset, (c) Fraction of pixels above threshold 0.89 and not included in dataset.

ISBN: 978-1-7281-0297-9

PART: CFP19TEO-ART

D. Converting images to TFRecords

TFRecord [16] is a binary file format which can be used to create import pipelines for a neural network. Large datasets can be broken down into several TFRecord file which makes the training process efficient by reducing the memory required for loading training data and binary format helps in speeding up the reading step. We split our images into the training & testing sets with 13236 and 3306 images respectively. Then we split these further into 40 TFRecord shards each i.e. all images and corresponding labels were randomly distributed amongst 40 binary files.

IV. Training the Neural Networks

We fetched pre-trained versions of the networks from the Tensor Flow Slim repository on Github and trained their last layer on our dataset. These pre-trained networks are available as checkpoint files, which consists of all nodes in the graph and corresponding weights obtained after training the network on the original dataset. The retraining script loads weights for TensorFlow graphs from these checkpoints, for each layer except the last. This last layer is trained on the new dataset and a bottleneck layer is added to convert the final logits from classifying to 1000 categories (ImageNet [17] dataset) to the 9 genres required. We trained each of the networks on the colored and black & white images followed evaluation metric in both cases. We trained the last layer of each neural network for 1000 steps over both color and threshold image datasets. We used an initial learning rate of 0.001 and employed a polynomial decay for a final rate of 0.0009. Any higher value of the initial learning rate lead to an underfit on the data and thus a reduction in the accuracy obtained. For gradient descent, the Adagrad Optimiser with an initial accumulator of 0.1 was used due to its performance benefits when training large neural networks.

V. Deploying on NCS

A. Exporting Graph for deployment

Finally, the trained networks were "frozen" for inference. This is an important step when working with neural networks. A graph saved at intermediate steps of training contains many unnecessary features of the graph like the gradients which are only required when resuming training from this checkpoint. While exporting for inference all these are skimmed out and the vars in the graph are converted to constants to make it light-weight. The Intel Movidius NCSDK [18] provides a utility, *mvNCCompile*, to compile this frozen graph for the NCS. When trying to compile the graph for the stick we ran into multiple errors of the form "*[Error 5] Toolkit Error: Stage Details not supported*" which might be due to two possibilities:

1. The graph has not been exported for inference and still consists of variable weights which are required while training.
2. Some features in the graph are not yet supported by NCSDK which are further discussed in the section VII.

VI. Results

We quantified results for each network by calculating the accuracy obtained over the test set. We trained each network for 1000 steps over the color and threshold image dataset separately and then calculated corresponding accuracy over the test-set of 3306 images. Networks tend to perform better when trained on the colored dataset, thus it can be inferred that the more features in the cover images aid in the improvement of accuracy of the classifier as mentioned in Table II.

TABLE II. Accuracy of networks trained on Color and Threshold images

Network	Accuracy	
	Color Images	***Threshold Images***
Inception v3	15.1%	12.3%
MobileNet v2	16.4%	12.9%
Inception v4	14.7%	12.06%

We then, deployed the color trained version of each of the networks on the NCS and noted the time required for each inference. We compared the results with a Core-i5200. Though, the neural compute stick is a little slower when comparing inference times, it performed significantly better in terms of the inference performed per unit time per unit of power consumed without any loss in accuracy as mentioned in Table III.

TABLE III. Different metrics for the deployment on a Core i5 and the NCS

Metric	**Corei5-5200**			**NCS**		
	Inception v3	***Mobilenet v2***	***Inception v4***	***Inception v3***	***Mobilenet v2***	***Inception v4***
Inference Time(s)	0.037	0.027	0.039	0.287	0.238	0.290
Power Consumed(W)	69*	69*	69*	1.2	1.2	1.2
Inference/time/Watt	0.391	0.536	0.371	2.90	3.501	2.873

[a.] *Maximum TDP of this mobile CPU at 100% usage

VII. Conclusion and Future Work

We presented 3 CNN models trained on the book dataset and also ran inference using these networks on the neural compute stick. We introduced a viable method of constructing a dataset of books with requirements like good quality cover images. While the results obtained from the networks are satisfactory the following should be considered as future steps for obtaining improved results:

ISBN: 978-1-7281-0297-9

PART: CFP19TEO-ART

- We trained the neural networks for 1000 steps only, given more time and resources it would be interesting to find out the accuracy that these CNNs could achieve on the book dataset and thus help in concluding if the cover images and its genre when clearly related lead to the better sales of a book.
- Multiple networks that we worked with, like ResNet v2 [19] and Visual Geometry Group (VGG-16) [20] could not be compiled for the neural compute stick. The NCSDK is still under development and thus does not support many Tensorflow graph features currently. It would be especially riveting to be able to run these high accuracy networks on the neural compute stick for various computer vision applications.
- Even though we formulated an algorithm to include only those books which have sufficient features on their cover, it should be noted that cover images of the books can be vague and do not represent their genres well. For better results such cases should be removed from the dataset. This would require manual effort as deciding whether an image is good representation of its class would be best done by a human oracle.
- Ensemble learning [21] method can be used to improve the classification results. Using the title of the book as an input to the network will aid in the better simulation of the situation of a human's perception of new book. The title of the book can be given to a text-based classifier network and then the outputs of this network and CNN can be combined to improve the accuracy of classification of the system.
- Taking a disjoint set of genres for the classification of images would also lead to an increment in accuracy. We considered 9 genres for classification in order to get well-rounded results but reducing these to genres that do not have any common books like Biography and Fantasy etc. would help in a significant improvement of accuracy.

Being able to run complex neural network at the edge with low power requirements and no loss in accuracy will immensely improve the intelligent classification capabilities of mobile devices like camera, drones, mobile phones etc.

VIII. Acknowledgement

We would like to acknowledge the help offered by the Intel Movidius development team for providing the device for our experiment and also helping in the initial setup process.

References

[1] Lisa Torrey and Jude Shavlik, "Transfer Learning", Appears in the "Handbook of Research on Machine Learning Applications", published by IGI Global, edited by E. Soria, J. Martin, R. Magdalena, M. Martinez and A. Serrano, 2009

[2] Intel Movidius Neural Compute Stick, Intel Corporation, https://software.intel.com/en-us/movidius-ncs

[3] Intel AI, https://ai.intel.com/intel-movidius-myriad-vpus/

[4] K. Mondal, "Big Data Parallelism: Issues in different X-Information Paradigms", 2015, Procedia Computer Science Volume 50, pp. 395-400

[5] N. Silberman and S. Guadarrama, "TensorFlow-Slim image classification model library", 2016. https://github.com/tensorflow/models/tree/master/research/slim

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision", 2015, arxiv preprint arXiv:1512.00567v3 [cs.CV]

[7] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning", 2016, arxiv preprint arXiv:1602.07261v2 [cs.CV]

[8] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", 2018, arxiv preprint arXiv:1801.04381v3 [cs.CV]

[9] Xiaofan Xu, João Amaro, Sam Caulfield, Gabriel Falcao and David Moloney, "Classify 3D voxel based point-cloud using convolutional neural network on a neural compute stick", in 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, IEEE, 2017, pp. 37-43

[10] "Movidius announces deep learning accelerator and Fathom software framework." https://www.movidius.com/solutions/machine-vision-algorithms/machine-learning.

[11] Goodreads API, https://www.goodreads.com/api

[12] Open Library API, https://openlibrary.org/developers/api

[13] Google Books APIs, https://developers.google.com/books/

[14] Bradski, G, "The OpenCV Library", (2000), Dr. Dobb's Journal of Software Tools

[15] Yu-Ting Pai, Yi-Fan Chang, Shanq-Jang Ruan, "Adaptive thresholding algorithm: Efficient computation technique based on intelligent block detection for degraded document images", 2010, Pattern Recogintion Volume 43 Issue 9, pp. 3177-3187

[16] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo,Zhifeng Chen, Craig Citro et. al,"TensorFlow: Large-scale machine learning on heterogeneous systems",2015. Software available from tensorflow.org.

[17] Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L., "ImageNet: A Large-Scale Hierarchical Image Database", 2009, CVPR

[18] Intel Corporation, Intel NCSDK v2, 2018, https://github.com/movidius/ncsdk/

[19] Thomas Dietterich, "Ensemble methods In Machine Learning", 2000, MCS '00 Proceedings of the First International Workshop on Multiple Classifier Systems, pp. 1-15

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Identity Mappings in Deep Residual Networks.", 2016, arxiv preprint arXiv: 1603.05027

[21] Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014, arxiv:1409.1556