Benchmark analysis of popular ImageNet classification deep CNN architectures

Conference Paper · August 2017

DOI: 10.1109/SmartTechCon.2017.8358502

3 authors, including:

Mustafa Alghali
University of Khartoum
3 PUBLICATIONS **1** CITATION
SEE PROFILE

Tarig Khalid
Multimedia University
7 PUBLICATIONS **8** CITATIONS
SEE PROFILE

Benchmark Analysis of Popular ImageNet Classification Deep CNN Architectures

Mustafa Alghali Elsaid Muhammed, Ahmed Abdalazeem Ahmed, and Tarig Ahmed Khalid

Electrical and Electronics Engineering Department, University of Khartoum

{mustafa.alghali@outlook.com, aaaba007@gmail.com, t.a.khalid@ieee.org }

***Abstract*— Deep Convolutional Neural Networks (CNNs) have recently demonstrated the state-of-the-art classification performance on ImageNet Large Scale Visual Recognition Challenge (ILSVRC) since 2012, yet there is relatively no clear understanding of the reasons behind their outstanding performance, or how they might be improved, In this paper we present a novel benchmarking of multiple state-of-the-art deep CNN architectures by providing an analysis of important performance metrics: speed, memory consumption, and network parameters utilization. Key findings are: (1) fully connected layers are of a high cost on speed and memory consumption compared to the sparsely connected layers; (2) the depth and the performance of an architecture are in a nonlinear relationship and constrained by layers transformation types; (3) 1x1 convolutions are an efficient way to reduce dimensionality and pooling features; (4) addition units as in residual and densely connected networks accelerate the backpropagation time by distributing the gradients through the graph, we believe our set of benchmarks are a step towards better realization of the best architectural design choices of Deep CNNs.**

***Index Terms*— Computer Vision, Image Classification, ImageNet, Convolutional Neural Networks, Docker, Machine Learning, Caffe.**

I. Introduction

Since 90s [1] Deep Convolutional Neural Networks have demonstrated excellent performance in object recognition, detection, and localization and many other computer vision tasks, Deep Learning or specifically Convolutional Neural Networks has been characterized as a BUZZWORD only after the record beating performance achieved by AlexNet a large CNN architecture designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton at university of Toronto that was used to win the 2012 ILSVRC challenge scoring a top 5% error rate of 15.4% with The next best entry achieved an error of 26.2% [2], after then many modifications and deviated CNN architectures from AlexNet were proposed and they have become the household names in the competition, this re-evolution and rebranding of CNN architectures can be linked to the availability of large enough labeled Images datasets such as ImageNet which holds over 14 million hand annotated images that are available publicly for research purposes [3].

GPUs also helped accelerating the heavy computation processes carried out by these architectures and reduced the time needed to preform training phases, in the same time better overfitting reduction strategies were developed such as dropout by G. Hinton [2].

Despite all the progresses shown by the various proposed architectures still there were few insights and logical reasoning about how they had achieved the stat-of-the-art records, which makes further improvements subject to the trial and error strategies, for example one of our benchmarks findings is that ResNet152 architecture [4] which has a depth of 152 layers yet scored higher speed record than VGG-19 architecture [5] which has depth of 19 layers only, another finding example is the parameter space size of GoogLeNet [6] which has 22 layers that is far less than the size of the basic AlexNet architecture [2]. Another reason of the lack of the insights is that CNNs are relatively still new hot topic of research where the findings are updated more frequently.

II. Selected CNN architectures

ILSVRC has played a major role in advancing the state-of-the-art CNN architectures, where every year since 2012 new CNN architecture hits a lower classification error than its previous year records [7], for the purpose of our experiments we aren't only concerned with finding implementations of these CNN architectures but also having them trained on ImageNet dataset or subset of it, the major help in this part of the project was from Caffe Model Zoo a collection of trained models applied for problems ranging from simple regression, to large-scale visual classification [8] various sources for our base models implementations are referenced in the project Github repository [9]. Our selection wasn't only based on the ILSVRC results but also was dependent on which models were revealed and made available to the public, as there is a bunch of winner models which were designed by companies, institutes or even individuals who aren't welling to reveal their information publicly yet [7], and thus we have ended up selecting the following CNN architectures: a) AlexNet the winner of 2012 with a top 5 classification error of 0.153 [2], b) VGG ranked 2nd place in 2014 challenge with a top 5 classification error of 0.0732 [5], c) GoogLeNet winner of 2014 challenge with a top 5 classification error of 0.0665 [6], d) network in network (NIN) a model designed by National University of Singapore (NUS) in 2014 [10], e) residual networks (ResNets) winner of 2015 challenge with a top 5 classification error of 0.0356 [4], and f) densely connected convolutional networks (DenseNets) [11]. In section III we discuss the variations between these

architectures and their hyper-parameters selection in more details.

III. VARIATIONS BETWEEN THE SELECTED ARCHITECTURES

a) AlexNet which is considered the spark of deep CNN architectures re-evolution; with its outstanding improvement where it has achieved 15% top 5 error with next best entry of 26% [2]
Compared to recent architecture AlexNet is considered among the simplest, it was trained to classify 1.2 million high resolution images into 1000 different classes, the network consist of 5 convolutional layers, max-pooling layers, and 3 fully connected layers, they used ReLUs as nonlinearity function, implemented dropout layer to overcome overfitting, and optimized the weights using stochastic gradient descent with a batch size of 128 samples, the training took between five to six days on two GTX 580 GPUs as described in [2].
b) The VGG networks by Karen Simonyan and Andrew Zisserman from the University of Oxford were the first to invent the idea of using combinations of small (3x3) filters to emulate the effect of larger filters such as 5x5 and 7x7, which gave them the advantage of back to back stacking of multiple convolutional layers for greater depth and thus better performance, while reducing the number of parameters by using only 3x3 filters with stride and padding of 1, they also applied 2x2 maxpooling with stride of 2, and it was similar to AlexNet in using ReLUs layers, 3 fully connected layers, and batch gradient descent for optimization, the training was performed using 4 Nvidia Titan Black GPUs for two to three weeks as described in [5], two architectures VGG-16 and VGG-19 with 16 and 19 layers respectively were selected to perform our benchmarks experiments.
c) GoogleNet was the first CNN architecture that deviated from the conventional approach of stacking multiple convolutional and pooling layers on top of each other; by introducing the inception module which allows multi-level feature extraction from the same input by applying multiple convolutional filters as well as pooling and concatenate the results into one feature map with help of 1x1 convolution operations, the 1x1 convolutions result in reducing the depth of the volume, which is equivalent to performing cross-channel correlation known also as “feature pooling”, for example applying 20 filters of 1x1 convolution for an input volume of 120x120x50 would reduce it to a volume of 120x120x20, these inception layers are repeated many times, leading to a 22-layer deep model in the case of the our GoogLeNet selected architecture, one useful aspect of this design is that it aligns with the intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from different scales simultaneously [6], it's also worth mentioning that there is no use of fully connected layers, instead they used an average pooling, to go from a 7x7x1024 volume to a 1x1x1024 volume, and it took them a week to train the network using small number of GPUs as described in [6].
d) Network-in-Network (NIN) is an approach proposed by Lin et al by designing micro-networks as blocks of multilayer perceptron, these micro-networks are used to add non-linearity to the convolution operation instead of using linear filters, The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN, the output of this operation is then fed to the next layer, stacking these micro-networks in a repeating manner results in a deep NIN architecture, in 2014 NIN demonstrated state-of-the-art performance on CIFAR-10, CIFAR-100 and SVHN datasets as mentioned in [10].

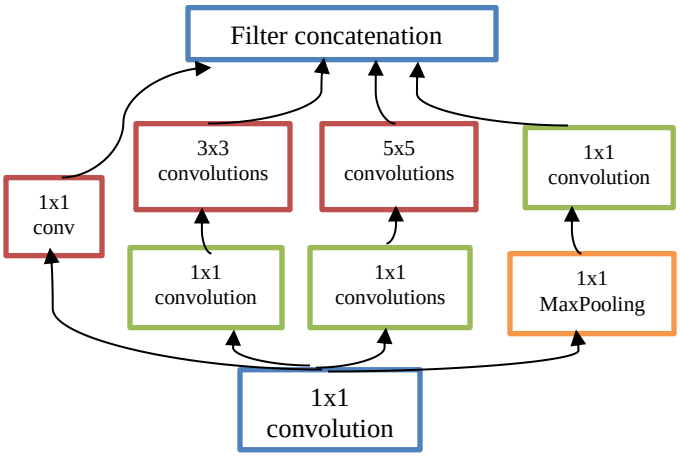


Fig. 1. Inception module with dimensionality reduction.

e) ResNets are characterized by two major deviations from the previous architectures, the first one is the introduction of Residual blocks in which the authors explicitly reformulated the layers as learning residual function with reference to the layer inputs as shown in figure 2.

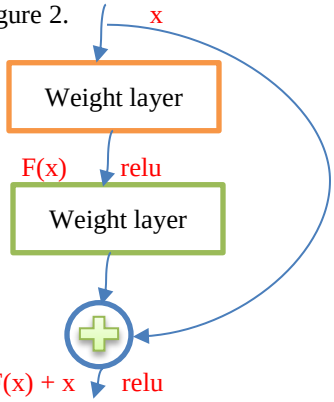


Fig. 2. Residual learning: Example residual building block.

The authors claim that building a reference for the function as we go forward through the network gives better representation when they are combined with raw data layers, instead of learning unreferenced functions. The other major deviation is their ultra-deep depth, 50, 101, and 152 are the numbers of layers for the models ResNet-50, ResNet-101, and ResNet-152 used in ILSVRC 2015 competition, The architecture is also missing fully connected layers at the end of the network, they have used 8 GPUs to train the network along two to three weeks as described in [4].
f) DenseNet published in 2016 it is also one of networks with more than 100 layers, but the major architectural feature of DenseNet is its dense connections, that each layer is connected directly to every other layer in a feed forward fashion, features of all pervious layers are fed to the subsequent layers in the network, the authors claim that the network alleviate the vanishing-gradient problem, strengthen feature propagation,

encourage feature reuse, and substantially reduce the number of parameters, DenseNet achieved state-of-the-art results across several highly competitive datasets such as CIFAR and ImageNet in 2016 as mentioned in [11], DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-161 are the select architectures in our experiments, The growth rate for the first 3 networks is k = 32, and k = 48 for DenseNet-161 [12].

IV. Implementation Tools

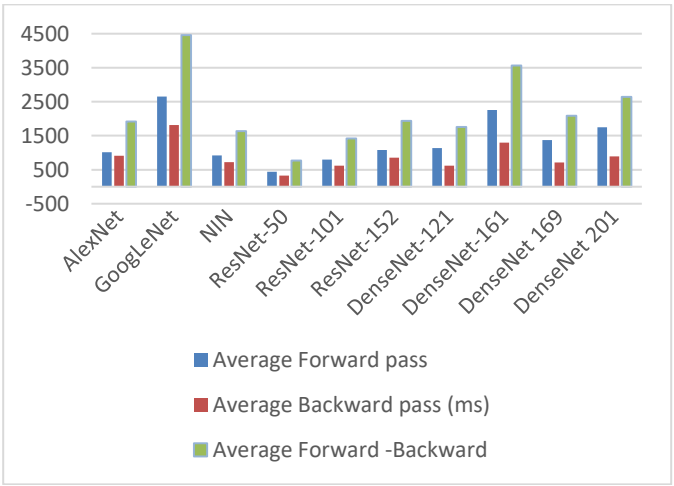
For the purpose of our experiments we have chosen Caffe a well-known Deep Learning framework developed by Berkeley AI Research with highly optimized C and C++ implementation which makes it speed enough to perform Deep CNN experiments on machines with limited computing resources as in the case of this project, we also have chosen Linux Docker container to resolve the dilemma of environment dependencies and the requirements to perform the test under the same configurations on multiple machine, we used the term "Dockerized" to reflect that our application is running inside a Docker container where the experiments were performed on the CPU version of "floydhub" Docker image that contains multiple deep learning frameworks installed along with group of Machine Learning python modules [13].

We can state that the availability of computing resources was one of the major bottlenecks in this project experiments, there was many failed attempts to got an access to high computing power resources, and thus we decided to proceed running our experiments using a local machine which has Intel Core i7 at 2.8 GHz x64 based processor and 8 GB RAM because most of benchmarking measures are relative between the different architectures and not obsolete numbers.

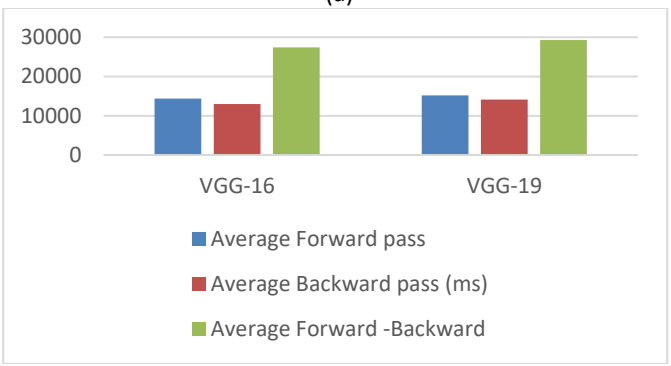
V. Speed and Memory Consumption Measures

In this section we report our results and comparisons, In speed performance measurement the test measures the average time taken by the selected architectures to perform a feed forward and backward for 50 Iterations using Caffe time benchmarks model execution layer-by-layer to check the system performance and measure relative execution times for models, the results are shown in figure 3, we have also recorded the average propagation time at each layer of the selected architectures and made them available in the project github repository [9].

In memory consumption test, we have recorded the percentage of RAM memory used by each architecture to perform a feedforward and backward operations, the results as shown in figure 4 consolidates the speed performance results with the fact that the depth and parameter space of the network add to its computation expense, for example it can be clearly seen that architectures which have the same building blocks such as VGGs, ResNets and DensNets do have proportional expense on speed and memory consumption as the number of layers increases, in contrast, the figures also reveals that architectures with different building blocks have vastly different depth-to-speed ratios, for example the 50 epochs time period of VGG-19 was 11 times bigger than of DenseNet-201's, subsequently VGG-19 was consuming memory in a double rate than of DensNet-201's. taking into consideration the types of layers it can be safe to link that depth-to-speed ratio variations with the network connections types, where sparsely connected architectures (who don't contain fully-connected layers) results in smaller parameter space, for example GoogleNet and NIN depend on global average pooling instead of fully connected layer in obtaining the classification probabilities that is why AlexNet and VGG networks generally have more expense on memory and speed than the others, another justification would be the use of 1x1 convolutions in addressing the depth of the network and pooling the feature maps, as in the case of GoogleNet which has 12x fewer parameters than the simple AlexNet architecture.



(a)



(b)

Fig. 3. Timing test results, the columns include the average time of the feed forward and backward propagation together and individually taken of 50 iterations (a) results of 10 selected architectures (b) results of VGG-16 and VGG-19.

On the other hand, VGGs although of their simple architectures have shown significant poor performance because of their huge parameter space which results from the usage of large amount of small filters (3x3) and small stride and paddings of 1, along with 3 fully connected layers.

In order to get an estimation of the parameter space utilization of each architecture we have plotted the size of Caffe binary file that holds the network values (weights + biases) for each architecture as shown in figure 5, similar to the memory consumptions results DenseNet and GoogleNet have shown to be the best architectures in terms of parameter space utilization, squeezing up to 4x less parameter space size with reference to AlexNet model, and 10x less with respect to VGG-19

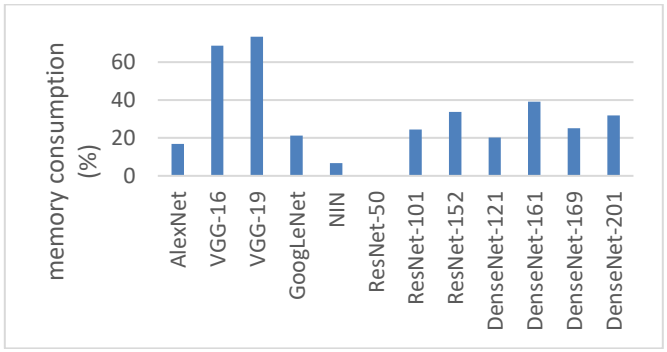


Fig. 4. Results of relative memory consumption benchmarks.

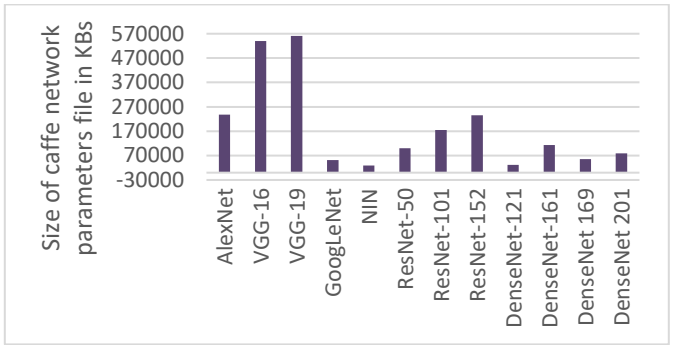


Fig. 5. Size of Caffe network parameters file in KBs

Figure 6 depicts backward-depth ratio for each architecture, with an interesting finding that ResNets and DenseNets have the lowest values, this can be justified by observing that during the backward pass of backpropagation, the gradient will flow easily through the graph because we have addition operations in both architectures, which distributes the gradient [4]. With reference to the propagation time measurements through each layer of the NIN architecture [9], it can be seen that micro-networks layers have recorded significant increase in the total forward-backward propagation time, which gives explanation to its poor performance when it's compared to its depth and the usage of 1x1 convolution for compression.

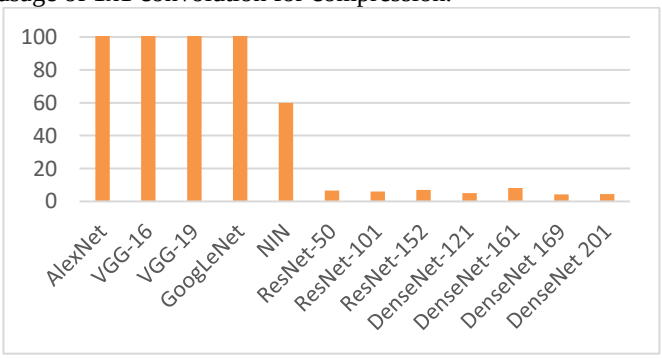


Fig. 6. Average backward-depth ratio for each architecture.

VI. NETWORK VISUALIZATION

Network parameters visualization gives better insights of the function of intermediate feature maps and the operations of the classifier; thus it can be used in a diagnostic role where it allows us to observe the evolution of features during training and diagnose potential problems with the model, for example learned filters in a well-trained network are expected to exhibit structure and being uncorrelated, also it helps answering how discriminative the features are in each layer in the pre-trained models [14], for example our selected layers for visualization of AlexNet architecture were: a) The first convolution layer filters and its output feature maps, for their intuitions about the lower feature representations, b) The last pooling layer for its intuitions about the higher feature representations, c) The first fully connected layer representing the histograms of the positive inner products, d) The output probability layer provides indications how determinate is the output class, the visualization test results are spanning across more than 55 pages and therefore we include here only the visualization of the selected AlexNet layers taken of a random sample feed image shown in figure 7, the rest of the visualization samples are available in the project github repository [9].

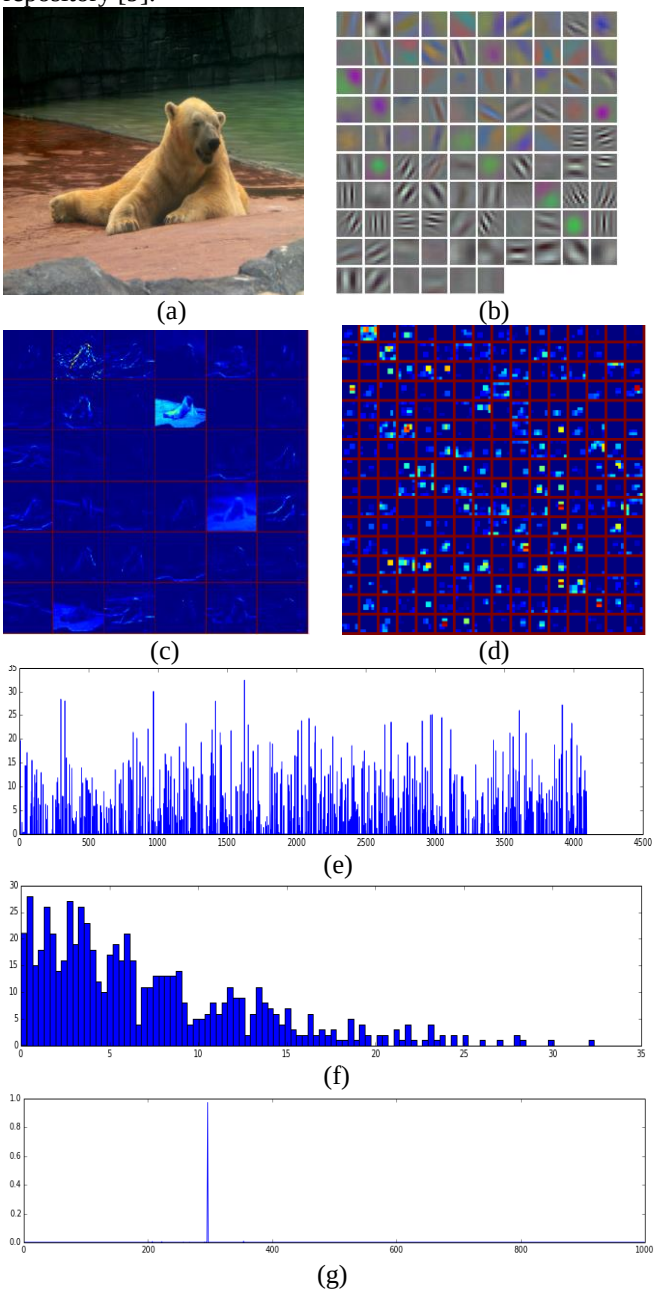
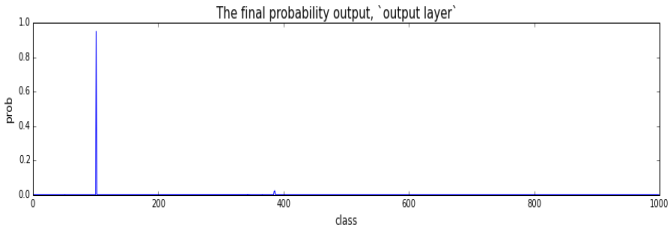


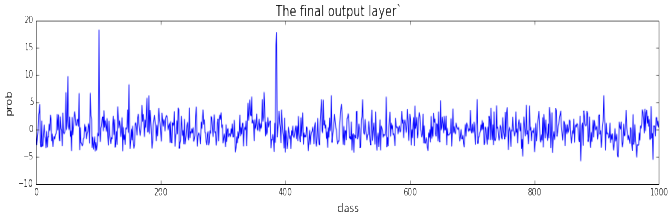
Fig. 7. (a) Input sample form ImageNet, (b) visualization of the first convolutional layer parameters, (c) visualization of the first convolutional layer output, (d) visualization of the fifth

pooling layer output, (e) visualization of the output of the first fully connected layer, (f) visualization of the PDF of the first fully connected layer, (g) visualization of the output (probability) layer.

The output layers' visualization shows that GoogLeNet, NIN and ResNets are showing floating non-determinative output when they are compared to VGG and AlexNet that can be referred to the usage of the Softmax probabilistic layer in the output instead of an inner product layer figure 8-(a) shows AlexNet response in the output layer for random input sample image while figure 8-(b) shows GoogLeNet response in the output layer for the same input sample



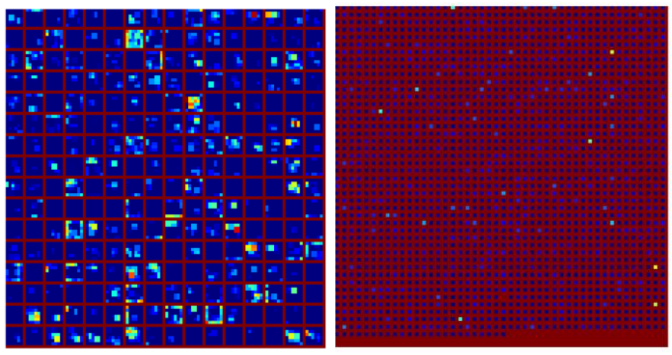
(a)



(b)

Fig. 8. (a) AlexNet response in the output layer for random input sample, (b) GoogLeNet response in the output layer for same input sample

It was also observable that the number of feature maps (depth) increase as we propagate throughout the different layers this can be clearly seen in figure 9 where ResNet-101 has shown to have much larger depth in the last pooling layer than of AlexNet



(a) (b)

Fig. 9. (a) The output of the fifth pooling layer in AlexNet, (b) The output of the fifth pooling layer in ResNet-101

VII. Conclusion

In this paper, we presented benchmarking of multiple state-of-the-art Deep CNN architectures in terms of speed, memory consumption and parameters, in order to push our goal towards better realization and logical reasoning between these models and the achieved state-of-the-art performances, and generate more insights that help the network designers evaluate the design choices that can lead to more efficient network architectures, we show that using sparsely connected layers gives better representation to the features and makes better utilization of the parameters than fully connected layers. We addressed the nonlinearity of the relationship between depth and the architecture performance where it's not necessarily to gain better accuracy or performance by stacking the same building blocks over each other, but combining them with more insightful transformation such as inception modules and residual blocks, we also spotted the light on the efficiency of 1x1 convolutions in dimensionality reduction and features pooling which yields in more performance boosting, we also show that addition units as in ResNets and DenseNets help accelerating the backpropagation speed, DenseNets have shown to be the best architecture in terms of speed-to-depth ratio scoring 26x lower than of AlexNet, finally we showed that DenseNets and GooglNets respectively are the best in terms of parameter space utilization, one of the proposed future work direction of this paper is to conduct studies on how to get benefits of the best design practices that led to performance, accuracy, or memory consumption improvements, and see how can they be imported into a hybrid model that utilize the best design tradeoffs.

VIII. References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Habbard and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," 1989.

[2] A. Krizhevsky, I. Sutskever and G. . E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012.

[3] ImageNet.org, "About ImageNet Summary and Statistics (updated on April 30, 2010)," [Online]. Available: http://image-net.org/about-stats. [Accessed 22 5 2017].

[4] K. He, X. Zhang , S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2015.

[5] A. Zisserman and K. Simonyan, "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-S CALE IMAGE RECOGNITION," 2015.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," 2014.

[7] J. Deng and O. Russakovsky, "ImageNet Large Scale Visual Recognition Challenge," in *IJCV*, 2015.

[8] b. v. labs, "Caffe Model Zoo," [Online]. Available: http://caffe.berkeleyvision.org/model_zoo.html. [Accessed 22 5 2017].

[9] M. Alghali, "Caffe-CNN-Benchmarks Github page," [Online]. Available: https://github.com/mustafaghali/Caffe-CNN-Benchmarks. [Accessed 22 5 2017].

[10] M. Lin, Q. Chen and S. Yan, "Network In Network," 2014.

[11] G. Huang, Z. Liu , L. van der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," 2016.

[12] liuzhuang, "Densely Connected Convolutional Network (DenseNet)," [Online]. Available: https://github.com/liuzhuang13/DenseNetCaffe. [Accessed 25 5 2017].

[13] S. Soundararaj. [Online]. Available: https://github.com/floydhub/dl-docker. [Accessed 24 5 2017].

[14] M. D.Zeiler, "Visualizing and Understanding convolutional networks," 2014.

[15] E. Shelhamer, "AlexNet Caffe model," [Online]. Available: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet. [Accessed 22 5 2017].