An Analysis of Deep Neural Network Models for Practical Applications

Alfredo Canziani & Eugenio Culurciello
Weldon School of Biomedical Engineering
Purdue University
{canziani,euge}@purdue.edu

Adam Paszke
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw
Warsaw, Poland
a.paszke@students.mimuw.edu.pl

Abstract

Since the emergence of Deep Neural Networks (DNNs) as a prominent technique in the field of computer vision, the ImageNet classification challenge has played a major role in advancing the state-of-the-art. While accuracy figures have steadily increased, the resource utilisation of winning models has not been properly taken into account. In this work, we present a comprehensive analysis of important metrics in practical applications: accuracy, memory footprint, parameters, operations count, inference time and power consumption. Key findings are: (1) fully connected layers are largely inefficient for smaller batches of images; (2) accuracy and inference time are in a hyperbolic relationship; (3) energy constraint are an upper bound on the maximum achievable accuracy and model complexity; (4) the number of operations is a reliable estimate of the inference time. We believe our analysis provides a compelling set of information that helps design and engineer efficient DNNs.

1 Introduction

Since the breakthrough in 2012 ImageNet competition [9] achieved by AlexNet [5] — the first entry that used a Deep Neural Network (DNN) — several other DNNs with increasing complexity have been submitted to the challenge in order to achieve better performance.

In the ImageNet classification challenge, the ultimate goal is to obtain the highest accuracy in a multi-class classification problem framework, regardless of the actual inference time. We believe that this has given rise to several problems. Firstly, it is now normal practice to run several trained instances of a given model over multiple similar instances of each validation image. This practice, also know as model averaging or ensamble of DNNs, dramatically increases the amount of computation required at inference time to achieve the published accuracy. Secondly, model selection is hindered by the fact that different submissions are evaluating their (ensemble of) models a different number of times on the validation images, and therefore the reported accuracy is biased on the specific sampling technique (and ensemble size). Thirdly, there is currently no incentive in speeding up inference time, which is a key element in practical applications of these models, and affects resource utilisation, power-consumption, and latency.

This article aims to compare state-of-the-art DNN architectures, submitted for the ImageNet challenge over the last 4 years, in terms of computational requirements and accuracy. We compare these architectures on multiple metrics related to resource utilisation in actual deployments: accuracy, memory footprint, parameters, operations count, inference time and power consumption. The purpose of this paper is to stress the importance of these figures, which are essential hard constraints for the optimisation of these networks in practical deployments and applications.

2 Methods

In order to compare the quality of different models, we collected and analysed the accuracy values reported in the literature. We immediately found that different sampling techniques do not allow for a direct comparison of resource utilisation. For example, central-crop (top-5 validation) errors of a single run of VGG-16[1] [10] and GoogLeNet [11] are 8.70% and 10.07% respectively, revealing that VGG-16 performs better than GoogLeNet. When models are run with 10-crop sampling,[2] then the errors become 9.33% and 9.15% respectively, and therefore VGG-16 will perform worse than GoogLeNet, using a single central-crop. For this reason, we decided to base our analysis on re-evaluations of top-1 accuracies[3] for all networks with a single central-crop sampling technique [13].

For inference time and memory usage measurements we have used Torch7 [2] with cuDNNv4 [1] back-end. All experiments were conducted on an NVIDIA Jetson TX1 board [7]: an embedded visual computing system with a 64-bit ARM® A57 CPU, a 1 T-Flop/s 256-core NVIDIA Maxwell GPU and 4 GB LPDDR4 of shared RAM. We use this resource-limited device to better underline the differences between network architecture, but similar results can be obtained on most recent GPUs, such as the NVIDIA K40 or Titan X, to name a few. Operation counts were obtained using an open-source tool that we developed [8]. For measuring the power consumption, a Keysight 1146B Hall effect current probe has been used with a Keysight MSO-X 2024A $200\,\mathrm{MHz}$ digital oscilloscope with a sampling period of $2\,\mathrm{s}$ and $50\,\mathrm{kSa/s}$ sample rate. The system was powered by a Keysight E3645A GPIB controlled DC power supply.

3 Results

In this section we report our results and comparisons. We analysed the following DDNs: AlexNet [5], batch normalised AlexNet [13], batch normalised Network In Network (NIN) [6], GoogLeNet [11], VGG-16 and -19 [10], ResNet-18, -34, -50 and -101 [4] and Inception-v3 [12], since they obtained the highest performance, in these four years, on the ImageNet [9] challenge.

3.1 Accuracy

Figure 1 shows one-crop accuracies of the most relevant entries submitted to the ImageNet challenge, from the AlexNet [5], on the far left, to the best performing Inception-v3 [12]. The newest ResNet-101 [4] and Inception-v3 surpass all other architectures by a significant margin of at least 7% (excluding shallower ResNet versions).

Figure 2 provides a different, but more informative view of the accuracy values, because it also visualises computational cost and number of network's parameters. The first thing that is very apparent is that VGG, even though it is widely used in many applications, is by far the most expensive architecture — both in terms of computational requirements and number of parameters. Its 16- and 19-layer implementations are in fact isolated from all other networks. The other architectures form a steep straight line, that seems to start to flatten with the latest incarnations of Inception and ResNet. This might suggest that models are reaching an inflection point on this data set. At this inflection point, the costs — in terms of complexity — start to outweigh gains in accuracy. We will later show that this trend is hyperbolic.

[1] In the original paper this network is called VGG-D, which is the best performing network. Here we prefer to highlight the number of layer utilised, so we will call it VGG-16 in this publication.

[2] From a given image multiple patches are extracted: four corners plus central crop and their horizontal mirrored twins.

[3] Accuracy and error rate always sum to 100, therefore in this paper they are used interchangeably.

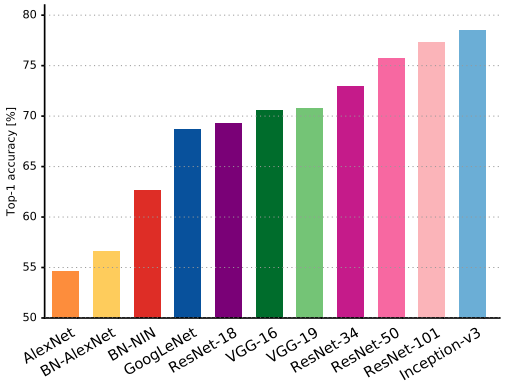

Figure 1: **Top1 *vs.* network.** Single-crop top-1 validation accuracies for top scoring single-model architectures. We introduce with this chart our choice of colour scheme, which will be used throughout this publication to distinguish effectively different architectures and their correspondent authors. Notice that network of the same group share colour, for example ResNet are all variations of pink.

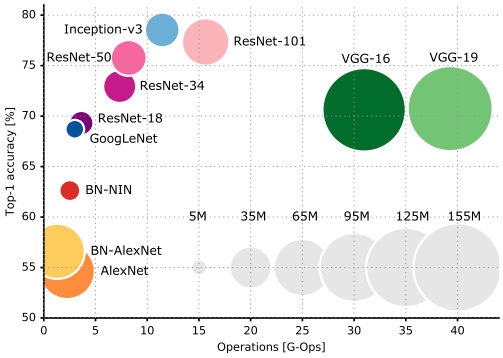

Figure 2: **Top1 *vs.* operations, size $\propto$ parameters.** Top-1 one-crop accuracy versus amount of operations required for a single forward pass. The size of the blobs is proportional to the number of network parameters; a legend is reported in the bottom right corner, spanning from 5×10^6 to 155×10^6 params.

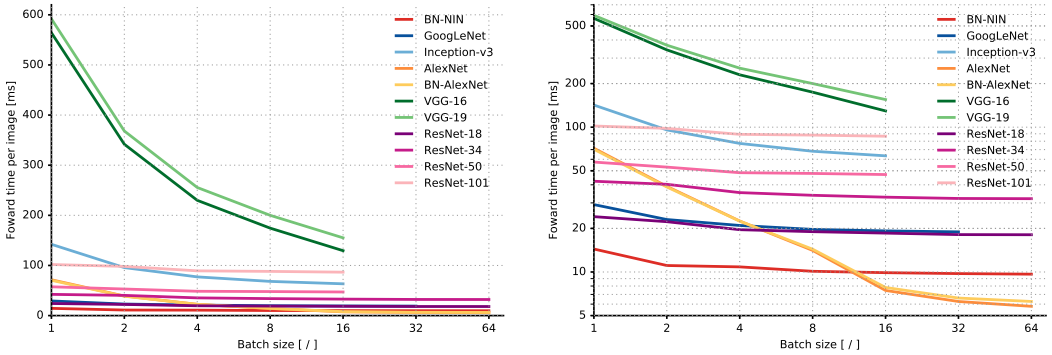

Figure 3: **Inference time *vs.* batch size.** These two charts show inference time across different batch sizes with a linear and logarithmic ordinate respectively and logarithmic abscissa. Missing data points are due to lack of enough system memory required to process bigger batches.

3.2 Inference Time

Figure 3 reports inference time per image on each architecture, as a function of image batch size (from 1 to 64). We notice that VGG processes one image in more than half second, making it a less likely contender in real-time applications on a NVIDIA TX1. AlexNet shows a speed up of roughly $15\times$ going from batch of 1 to 64 images, due to weak optimisation of its fully connected layers. It is a very surprising finding, that will be further discussed in the next subsection.

3.3 Power

Power measurements are complicated by the high frequency swings in current consumption, which required high sampling current read-out to avoid aliasing. In this work, we used a 200 MHz digital oscilloscope with a current probe, as reported in section 2. Other measuring instruments, such as an AC power strip with 2 Hz sampling rate, or a GPIB controlled DC power supply with 12 Hz sampling rate, did not provide enough bandwidth to properly conduct power measurements.

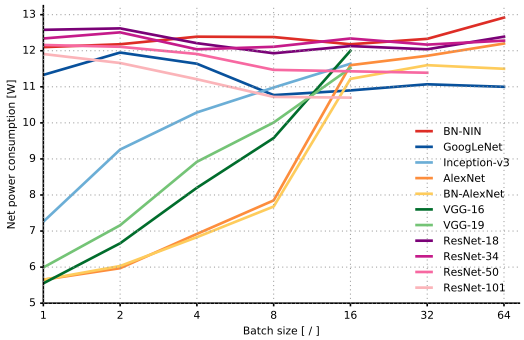

Figure 4: **Power *vs.* batch size.** Net power consumption (due only to the forward processing of several DNNs) for different batch sizes. The idle power of the TX1 board, with no HDMI screen connected, was 1.30 W on average. The max frequency component of power supply current was 1.4 kHz, corresponding to a Nyquist sampling frequency of 2.8 kHz.

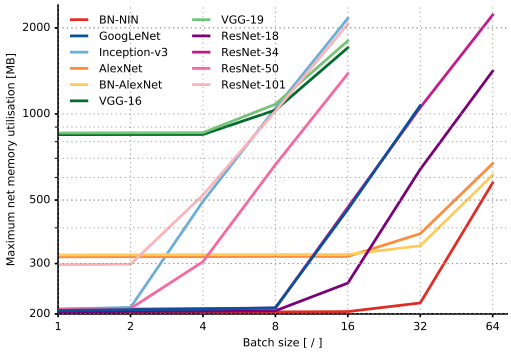

Figure 5: **Memory *vs.* batch size.** Maximum system memory utilisation for batches of different sizes. Memory usage shows a knee graph, due to the network model memory static allocation and the variable memory used by batch size.

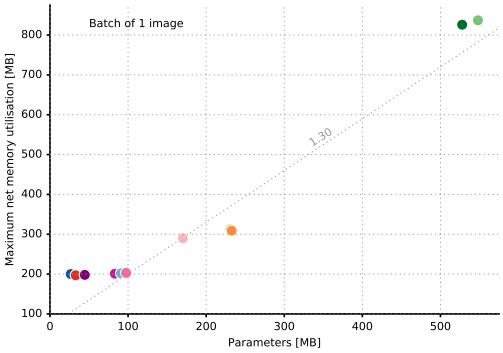

Figure 6: **Memory *vs.* parameters count.** Detailed view on static parameters allocation and corresponding memory utilisation. Minimum memory of 200 MB, linear afterwards with slope 1.30.

In figure 4 we see a dependency on batch-size for some models. This is due to the difference between the fully-connected layer used in each models. Note that[4] for AlexNet, its fully connected layers account for as much as 84% of its inference time, for batch size of 1, and for 33%, for batch size of 16. Analogously, VGG MLP takes 64% and 8% of its inference time for batches of 1 and 16 images respectively. Therefore, we note that the matrix-matrix multiplication (`gemm` operation) is optimised for larger batches, and thus only in this condition it is able to fully utilise all available hardware resources. On the other side, for architectures like ResNet and GoogLeNet, their fully connected layers are so small, accounting for only 1% of total operation count, such that they do not suffer from this issue.

3.4 Memory

We analysed system memory consumption of the TX1 device, which uses shared memory for both CPU and GPU. Figure 5 shows that the maximum system memory usage is initially constant and then raises with the batch size. This is due the initial memory allocation of the network model — which is the large static component — and the contribution of the memory required while processing the batch, proportionally increasing with the number of images. In figure 6 we can also notice that the initial allocation never drops below 200 MB, for network sized below 100 MB, and it is linear afterwards, with a slope of 1.30.

[4] Using the NVIDIA Visual Profiler it is possible to see detailed running times for the kernels used.

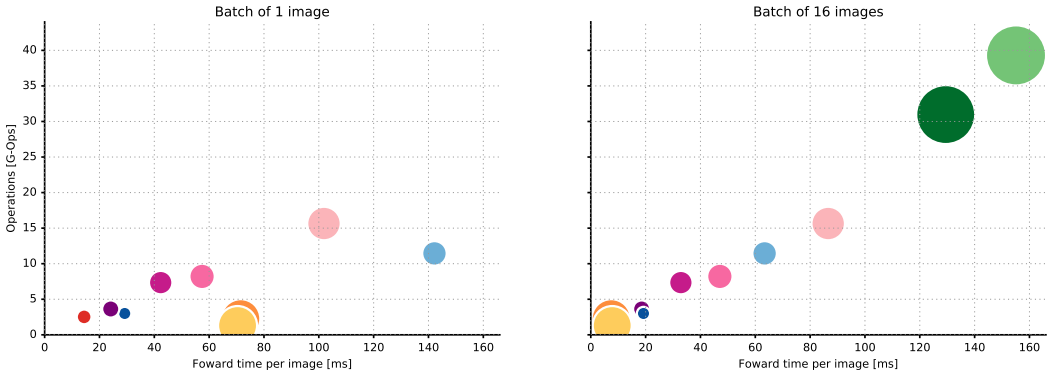

Figure 7: **Operations *vs.* inference time, size $\propto$ parameters.** Relationship between operations and inference time, for batches of size 1 and 16 (biggest size for which all architectures can still run). Not surprisingly, we notice a linear trend, for batches of 16 images, where resources are fully utilised, and therefore operations count represent a optimal estimation of inference time.

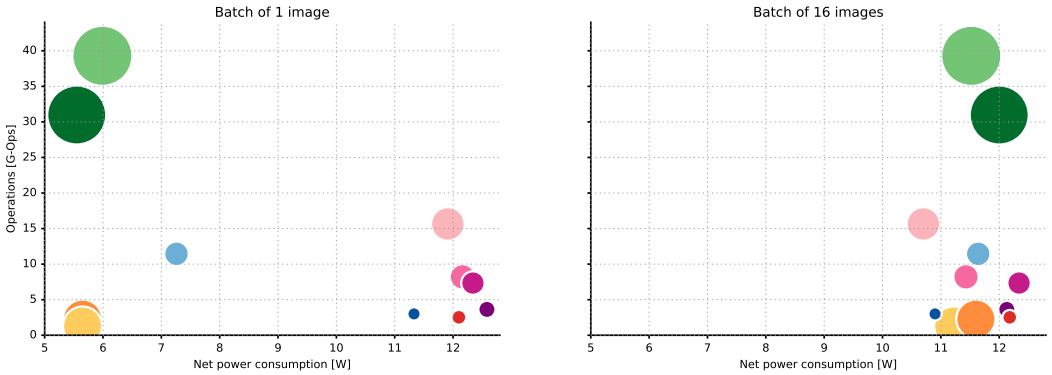

Figure 8: **Operations *vs.* power consumption, size $\propto$ parameters.** Independency of power and operations is shown by a lack of directionality of the distributions shown in these scatter charts.

3.5 Operations

Operations count is essential for establishing a rough estimate of inference time and hardware circuit size, in case of custom implementation of neural network accelerators. In figure 7, for a batch of 16 images, there is a linear relationship between operations count and inference time per image. Therefore, at design time, we can pose a constraint on the number of operation to keep processing speed in a usable range for real-time applications or resource-limited deployments.

3.6 Operations and Power

In this section we analysed the relationship between power consumption and number of operations required by a given model. Figure 8 reports that there is no specific power footprint for different architectures. When full resources utilisation is reached, generally with larger batch sizes, all networks consume roughly the same amount of power, with a standard deviation of 1 W. This corresponds to the maximum system power at full utilisation. Therefore, if energy consumption is one of our concerns, for example for battery-powered devices, one can simply choose the fastest architecture which satisfies the application minimum requirements.

3.7 Accuracy and Throughput

We note that there is a non-trivial linear upper bound between accuracy and number of inferences per unit time. Figure 9 illustrates that for a given frame rate, the maximum accuracy that can be achieved is linearly proportional to the frame rate itself. All networks analysed here come from

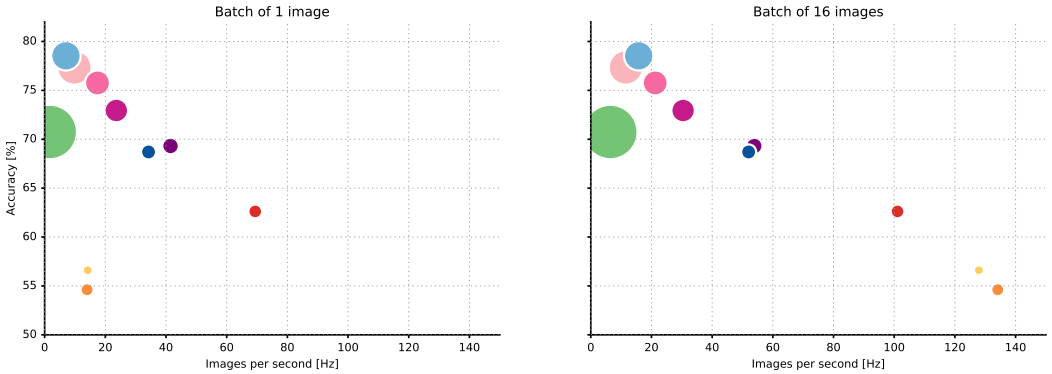

Figure 9: **Accuracy *vs.* inferences per second, size $\propto$ operations.** Non trivial linear upper bound is shown in these scatter plots, illustrating the relationship between prediction accuracy and throughput of all examined architectures. These are the first charts in which the area of the blobs is proportional to the amount of operations, instead of the parameters count. We can notice that largers blobs are concentrated on the left side of the charts, in correspondence of low throughput, *i.e.* longer inference times.

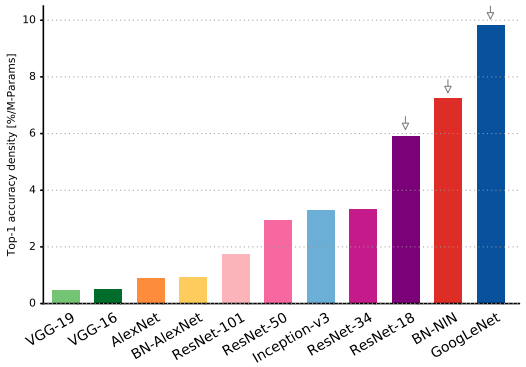

Figure 10: **Accuracy per parameter *vs.* network.** Information density (accuracy per parameters) is an efficiency metric that highlight that capacity of a specific architecture to better utilise its parametric space. Models like VGG and AlexNet are clearly oversized, and do not take fully advantage of their potential learning ability. On the far right, ResNet-18, BN-NIN [6] and GoogLeNet (marked by grey arrows) do a better job at squeezing all their neurons to learn the given task, and are the winners of this section.

several publications, and have been independently trained by other research groups. Here we show how their accuracy lies on a straight line, perhaps giving us the first architecture designing rule that can be applied for more systematic neural network engineering. In particular, chosen a specific inference time, one can now come up with the theoretical accuracy upper bound when resources are fully utilised, as seen in section 3.6. Since the power consumption is constant, we can even go one step further, and obtain an upper bound in accuracy even for an energetic constraint, which could possibly be an essential designing factor for a network that needs to run on an embedded system.

As the spoiler in section 3.1 gave already away, the linear nature of the accuracy *vs.* throughput relationship translates into a hyperbolical one when the forward inference time is considered instead. Then, given that the operations count is linear with the inference time, we get that the accuracy has an hyperbolical dependency on the amount of computations that a network requires.

3.8 Parameters Utilisation

DNNs are known to be highly inefficient in utilising their full learning power (number of parameters / degrees of freedom). Prominent work [3] exploits this flaw to reduce network file size up to $50\times$, using weights pruning, quantisation and variable-length symbol encoding. It is worth noticing that, using more efficient architectures to begin with may produce even more compact representations. In figure 10 we clearly see that, although VGG has a better accuracy than AlexNet (as shown by figure

1), its information density is worse. This means that the amount of degrees of freedom introduced in the VGG architecture bring a lesser improvement in terms of accuracy. Moreover, GoogLeNet achieves the highest score, showing that $20\times$ less parameters are sufficient to provide state-of-the-art results.

4 Conclusions

In this paper we analysed multiple state-of-the-art deep neural networks submitted to the ImageNet challenge, in terms of accuracy, memory footprint, parameters, operations count, inference time and power consumption. Our goal is to provide insights into the design choices that can lead to efficient neural networks for practical application, and optimisation of the often-limited resources in actual deployments. We show that there is much room for improvement for the fully connected layers, which show strong inefficiencies for smaller batches of images. We show that accuracy and inference time are in a hyperbolic relationship: a little increment in accuracy costs a lot of computational time. We show that number of operations in a network model can effectively estimate inference time. We show that an energy constraint will set a specific upper bound on the maximum achievable accuracy and model complexity, in terms of operations counts. Finally, we show that GoogLeNet is the best architecture in terms of parameters space utilisation, squeezing up to $10\times$ more information per parameter used respect to the reference model AlexNet, and $20\times$ respect VGG-16.

Acknowledgment

This paper would have not look so pretty without the *Python Software Foundation*, the `matplotlib` library and the communities of *stackoverflow* and TEX of *StackExchange* which I ought to thank. This work is partly supported by the Office of Naval Research (ONR) grants N00014-12-1-0167, N00014-15-1-2791 and MURI N00014-10-1-0278. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the TX1, Titan X, K40 GPUs used for this research.

References

[1] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient Primitives for Deep Learning. *arXiv.org arXiv:1410.0759*, 2014.

[2] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011.

[3] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[7] nVIDIA. Jetson tx1 module. `http://www.nvidia.com/object/jetson-tx1-module.html`.

[8] Adam Paszke. torch-opcounter. `https://github.com/apaszke/torch-opCounter`.

[9] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *arXiv preprint arXiv:1512.00567*, 2015.

[13] Sergey Zagoruyko. imagenet-validation.torch. `https://github.com/szagoruyko/imagenet-validation.torch`.