Deep Neural Network Capacity

Aosen Wang  
Midea Emerging Technology Center  
San Jose, CA, 95134  
aosen.wang@midea.com

Hua Zhou  
Midea Emerging Technology Center  
San Jose, CA, 95134  
hua.zhou@midea.com

Wenyao Xu  
Computer Science and Engineering Department  
State University of New York at Buffalo  
NY, USA  
wenyaoxu@buffalo.edu

Xin Chen*  
Midea Emerging Technology Center  
San Jose, CA, 95134  
chen1.xin@midea.com

Abstract

In recent years, deep neural network exhibits its powerful superiority on information discrimination in many computer vision applications. However, the capacity of deep neural network architecture is still a mystery to the researchers. Intuitively, larger capacity of neural network can always deposit more information to improve the discrimination ability of the model. But, the learnable parameter scale is not feasible to estimate the capacity of deep neural network. Due to the overfitting, directly increasing hidden nodes number and hidden layer number are already demonstrated not necessary to effectively increase the network discrimination ability.

In this paper, we propose a novel measurement, named “total valid bits”, to evaluate the capacity of deep neural networks for exploring how to quantitatively understand the deep learning and the insights behind its super performance. Specifically, our scheme to retrieve the total valid bits incorporates the skilled techniques in both training phase and inference phase. In the network training, we design decimal weight regularization and 8-bit forward quantization to obtain the integer-oriented network representations. Moreover, we develop adaptive-bitwidth and non-uniform quantization strategy in the inference phase to find the neural network capacity, total valid bits. By allowing zero bitwidth, our adaptive-bitwidth quantization can execute the model reduction and valid bits finding simultaneously. In our extensive experiments, we first demonstrate that our total valid bits is a good indicator of neural network capacity. We also analyze the impact on network capacity from the network architecture and advanced training skills, such as dropout and batch normalization.

1. Introduction

Deep Learning [11] is emerging in recent years in both academia and industrial communities, especially in computer vision domain. The deep neural network, whose representative is convolutional neural network [10], has demonstrated great success in many applications, such as image classification, object tracking and scene understanding. However, network capacity is still a big challenge to the researchers. The capacity of network is extremely significant because it is the indicator of discrimination ability of deep models. On one hand, researchers can design more efficient and effective network architecture with the guidance of capacity changes in the training phase. This will result in significant accuracy improvement. On the other hand, network capacity can help us remove the redundancy in the neural network, especially for deep networks. It can achieve less computation and small memory footprint. This will benefit the facilitation of large and complex deep models onto embedded devices to enable intelligence, like smartphone.

Up to now, there is no in-depth work exploring the network capacity for deep neural networks. The current hot research topics are concentrating on two directions: one is skilled network architecture design and the other one is advanced techniques for the training. By introducing imagenet large scale visual recognition challenge (ILSVRC), several state-of-the-art network architectures are proposed to improve the accuracy of image classification. Some representatives include AlexNet [9], VGG Net [14], GoogLenet [16] and ResNet [4]. The other direction is to design techniques...
for the training to suppress overfitting in order to improve
the accuracy. The typical examples comprises dropout [15]
and batch normalization [5]. However, all above methods
are only trying to enlarge network capacity heuristically.

In this paper, we propose a novel network capacity as-
scoring criterion, “total valid bits”, to quantitatively analyze
the network architecture and training skills. We provide an
example of network capacity comparison among the four
state-of-the-art networks in Figure 1. We can observe that
four different markers indicate the total valid bits of the four
networks, AlexNet, VGG Net, GoogLenet and ResNet. The
larger total valid bits means larger network capacity, which
also correspondingly improve the discrimination ability of
the network. The radius of the outer circle shows the aver-
age valid bits, which is critical to explore the reasons why
specific architecture and training skills achieve better re-
results. We will also illustrate this in details in the following
sections.

Specifically, our strategy to identify the network capacity
comprises two-phase processing, including training phase
and inference phase. The goal of our training phase opti-
mization is to obtain integer-oriented network representa-
tions to reduce information loss of quantization in inference
phase. We propose decimal weight regularization to penal-
ize the decimals of the weights and 8-bit forward quanti-
zation to constrain the weight range under control. In the
inference phase, we collect the statistics of each layer re-
response using a subset of validation set. We apply adaptive-
bitwidth and non-uniform quantization on the weights and
intermediate layer response. We allow zero bitwidth in the
quantization in order to remove the trivial parts of all learn-
able weights. The information loss is applied to determine
the final quantization bits for each learnable weight. The
total valid bits are then summed up by weights in all net-
work layers. The extensive experimental results show that
our total valid bits can properly characterize the deep neural
network capacity. We also explore the network architecture
design and training skill from the network capacity perspec-
tive.

2. Related Work

In the literature, we didn’t find the prior work quanti-
tatively analyzing the network capacity in deep neural net-
work domain. But we found two topics, CNN quantization
and model reduction, which were potential to help explore
network capacity. We illustrated them in this section.

2.1. CNN Quantization

The CNN quantization is a hot topic in the current deep
learning domain. This technique can enable deep learn-
ing on embedded system. The optimization goal of CNN
quantization is to reduce the computation and memory foot-
print by properly reduce the bitwidth of model weights.
Work in [13] proposed XNOR net, which considers to quan-
tize the weights and inputs only with 1 bit. By carefully
compensating the training phase with a full-precision range
scalar, the XNOR net could achieve a good recognition ac-
curacy. Another representative work [17] improve the draw-
back of XNOR net. It proposed new strategy to take the
model accuracy into the optimization goal. Experiments
also demonstrated its good performance. However, all these
techniques are all tending to optimize the model discrimina-
heuristically.

2.2. Model Reduction

The typical work in model reduction of deep neural net-
work is “Deep Compression” [3]. The author applied net-
work pruning, weight clustering and weight encoding to ag-
grassively reduce the redundancy of the deep models. Its
results were astonishing by compressing the original huge
model into a very small footprint. However, this method
sacrifice the accuracy of deep learning models.

3. Network Capacity Identification

In this section, we elaborate our design of identifying
the deep neural network capacity. Specifically, we add INT
weight regularization and 8-bit forwarding quantization into
the training phase. We execute the total valid bits calcula-
tion in the inference phase with the adaptive-bitwidth and
non-uniform quantization strategy.

3.1. Preparation in Network Training

Our basic idea is to retrieve the most slender represent-
tation of deep neural network models. Thus, before we
identify the valid bits number, we always expect a compa-
ct and range-bounded weight representation of a trained
deep learning model. For the simplicity of presentation,
we choose the convolutional neural network to illustrate
our processing in the training phase. Formally, for the

Figure 1. The network capacity comparison on the four state-of-
the-art networks architectures based on our proposed total valid
bits.

| Network       | Year | Total Valid Bits | Top5: | 
|---------------|------|------------------|-------|
| AlexNet       | 2012 | ∞                | 83.6% |
| VGG Net       | 2014 |                  | 92.7% |
| GoogLenet     | 2014 |                  | 93.3% |
| ResNet        | 2015 |                  | 96.43%|
convolution layer, we consider the input feature map as \( S \in R^{h \times w \times c} \) and layer response at channel \( c_i \) as \( T(c_i) \):
\[
T(c_i) = \sum_{(d_k, d_s)} \langle W_{c_i, d_k}, S_{d_s} \rangle,
\]
(1)

where the \( W_{c_i} \) indicates the convolutional kernel at channel \( c_i \) with \( d_k \) as its receptive field. The \( d_s \) is the 2D spatial data field of the input feature map convoluting with the kernel \( W_{c_i} \). Another significant layer of convolutional neural network is the fully-connected layer, whose layer response can be represented as the followings:
\[
T(c_i) = \langle W_{c_i}, S \rangle.
\]
(2)

### 3.1.1 Decimal Weight Regularization

The compact and range-bounded representation of neural network is a critical requirement for the quantization quality in the inference phase. To achieve this goal, the first task is to push the full-precision parameters approximating to the nearest integers. We propose decimal weight regularization to address this issue. This idea is motivated from \( L_2 \)-norm regularization \([12]\) to suppress the overfitting in the traditional shallow neural network:
\[
T'(c_i) = T(c_i) + \lambda_1 W_{c_i}^T W_{c_i},
\]
(3)

where \( \lambda_1 \) is a parameter to control the weights of regularization item. This strategy can effectively reduce the possibility of network weight \( W \) becoming too large. Similarly, our method tends to penalize the decimals of network learnable weights as the following:
\[
T''(c_i) = T'(c_i) + \lambda_2 \| W - [W'], [W] - W \|_2,
\]
(4)

where \( \lambda_2 \) is the parameter to control the portion of decimal weight regulation. \([\cdot]\) indicates the floor function and \( [\cdot] \) is the ceil operation. Intuitively, this technique will guide the weights to proceed to its nearest integer. In the backpropagation of neural network, we also need to calculate the derivative of our decimal weight regularization:
\[
\frac{\partial T''(c_i) - T'(c_i)}{\partial W_j} = \begin{cases} 
\lambda_2 & 0 < W_j - [W_j] < 0.5, \\
-\lambda_2 & 0.5 < W_j - [W_j] < 1, \\
0 & \text{else}.
\end{cases}
\]
(5)

We can observe that our regularization has three singleton points, where the scalar value \( W_j - [W_j] \) is 0, 0.5 and 1. We manually define the derivatives of these spots as zero.

### 3.1.2 8-Bit Forwarding Quantization

With decimal weight regularization, we can approximate the learnable weights of deep neural networks by integers without accuracy compromise. The next step is to constrain the integer ranges of the weights to avoid large information loss in the quantization. Before the uniform quantization, we can find that the bias term in the convolution layer and fully-connected layer are harmful to the compactness of the networks.

**Lemma 1** Bias item in the convolutional neural network will not increase the entire network capacity.

**Proof:** According to Eq. (2), the weight parameter corresponding bias item is 1. This constant value can be easily integrated into any other items with learnable weight parameters. Therefore, the accuracy will not be affected by adding or removing the bias item.

Based on the above Lemma [1], we can directly remove the bias item from the original deep neural networks to make the trained model representation more compact.

Furthermore, to constrain the range of integers, we develop 8-bit forwarding quantization strategy. The 8 bits is a loose bound to preserve the salient information in the forwarding pass of network of the training phase [2]. We choose the uniform quantization strategy to model the quantization function \( Q \) as the following:
\[
Q(x) = \left\lfloor \frac{x}{F_p \times 2^7} \right\rfloor \times F_p / 2^7,
\]
(6)

where \( F_p \) is the compensation parameter to memorize the maximal value based on the whole layer response. The format of weight parameter is still as full-precision, but the total number of such response levels are bounded by \( 2^8 - 1 \), with half in positive and half in negative. In each iteration of training phase, we apply this quantization function into the forwarding pass:
\[
T(c_i) = \sum_{(d_k, d_s)} \langle Q(W_{c_i, d_k}), S_{d_s} \rangle.
\]
(7)

\[
T'(c_i) = \langle Q(W_{c_i}), S \rangle.
\]
(8)

The Eq. (7) and Eq. (8) are indicating the convolution and fully-connected layer response under our 8-bit forwarding quantization strategy, respectively. Note that we only do the quantization in the forwarding pass in every training information and the back-prorogation still uses the full-precision number for learnable parameter updating. Therefore, with the compensation parameter \( F_p \), we can effectively limit the range of weight integers. The whole training phase algorithm can be summarized as Algorithm [1].

### 3.2 Valid Bits Identification in Inference Phase

In this section, we propose how to extract all the valid bits from the trained deep neural network models. The valid bits characterizes the information capacity of skilled-designed neural network model. An intuitive solution is to
directly brute force the quantization bit width for all the learnable weights after training. However, as the weight scale increases, this method becomes intractable to be calculated in a reasonable time budget.

One interesting observation is that the input data format is another significant factor influencing the deep neural network capacity. The input format of neural network is an easily-ignored factor when we consider exploring the capacity of neural network. Taking computation complexity of exploring the design space into account, we propose our inference-phase valid bits identification strategy, which includes auxiliary stimulating set, weight and layer response clustering, adaptive bit quantization and information loss pruning. Based on our proposed procedures, the final slender network architecture can be found and the network capacity, total valid bits, is also extracted.

3.2.1 Auxiliary Stimulating Set

The auxiliary stimulating set is the source to execute the trained model to collect the accurate statistics of each layer response. These layer responses are also taken as the input of the next layer, which can significantly affect the network capacity. We denote this auxiliary stimulating set as $A_s$:

$$A_s = \{ I \mid \exists l_i, T^w_{l_i}(I) \in Ext_{l_i} \},$$

where the $Ext_{l_i}$ is the set of maximal and minimal values of the $l_i$-th layer response. This definition reveals that the idea auxiliary simulating set should consist of the inputs whose can achieve extreme response in any layer of the neural networks. However, this condition is hard to be satisfied in practice, especially for the big data volume in the training set. Thus, we propose to apply random sampling from the validation set to construct this auxiliary stimulating set. According to our experiments, the statistics can be accurately acquired on condition that the size of this set is larger than 1000. Therefore, we will collect the statistics of each layer when applying auxiliary stimulating set on the trained deep learning model.

3.2.2 Weight and Layer Response Clustering

Weight clustering is an effective approach to reduce the computation complexity and even improve the result accuracy, when cooperating with the optimizations in the training phase.

Lemma 2 The similar weight in different layers of deep learning model should share the similar quantization bitwidth.

Proof: As discussed above, the intermediate layer response affect the network capacity. The similar value of the trained model should have the similar information entropy. Therefore, we consider the weights and response values as equal weight. We summarize this problem as the standard clustering problem, whose loss function is as the following:

$$\min_{c_1, \ldots, c_n, \mu_1, \ldots, \mu_k} J(c_1, \ldots, c_n, \mu_1, \ldots, \mu_k) \quad s.t.$$

$$J(c_1, \ldots, c_n, \mu_1, \ldots, \mu_k) = \sum_i \frac{1}{2} \| x_i - \mu_{c_i} \|_2,$$

where $c_i$ is the index number and $\mu_k$ indicates the $k$-th center position under Euclidian distance. This problem can be solved by K-means algorithm [6].

3.2.3 Adaptive Bitwidth Quantization

In this part, we elaborate the core part od valid bit identification, adaptive bitwidth quantization, which can find the most slender network architecture in digital domain. The research problem we tend to solve is as follows:

$$B^* = \min_b \quad s.t. \quad Acc(Q_b(W)) = \epsilon_0,$$

where $\epsilon_0$ is the accuracy of the original full-precision deep neural network. $Acc$ indicates accuracy statistics under given condition which is in the bracket and $Q_b(.)$ is the uniform quantization with $b$ as the given bitwidth. Therefore, Eq. (11) reveals that the optimal bitwidth $\hat{B}$ is the minimal bitwidth, which can keep the accuracy of the original deep neural network. We execute such quantization on each cluster after K-means algorithm.

Quantization on Network Weight: One characterization of our adaptive bitwidth is only quantizing the network learnable weights, not the intermediate layer response. This
is because our final total valid bits only accounts the bit consumption of model weights. The layer-wise response of the neural network is blended to attribute to properly model the effect on weights clustering result from layer response.

Zero Bitwidth Allowed in Quantization: Another characterization is our quantization allows zero bit, which means to remove the current weight parameter from the network architecture. This is reasonable for the deep learning, which is assumed to achieve good results by sparse valid connection in the vast network. This also demonstrates that the parameter scale of the network is not feasible to characterize the network capacity. This strategy also enables model reduction and quantization simultaneously.

3.2.4 Information Loss Pruning

In the adaptive bitwidth quantization, accuracy statistics needs to be collected on each bitwidth configuration setup. We further propose the information loss scheme to do the pruning. After each bitwidth configuration is set up, we calculate its information loss with the original data distribution by Jensen-Shannon Divergence [1]:

\[
JSD(U||V) = \frac{1}{2}D(U||M) + \frac{1}{2}D(V||M),
\]

where \(M = \frac{1}{2}(U + V)\), \(U\) and \(V\) are two independent data distributions. We also want to claim that \(D(U||V)\) is the Kullback-Leibler divergence [8]. After we obtain the JS divergence, we compare it with the pre-defined threshold \(\eta\). If it is larger than parameter \(\eta\), accuracy collection operation is ready to start. Otherwise, the current configuration is removed from the candidate list.

3.2.5 Valid Bits Identification Algorithm

Our valid bits identification algorithm includes all above four procedures. When we execute them step by step, the final quantization bitwidth can be summed up as the network capacity, total valid bits. We can also calculate the average valid bits, which indicates the average bit budget for one network weight. The average valid bit can also help us understand the impact from network architecture and training skills on the network discrimination ability. The entire algorithm is summarized in Algorithm 2.

4. Experiments

In this section, we conduct the extensive experiments to demonstrate the performance of our total valid bits for characterizing the deep neural network capacity. We first illustrate the experimental setup. We demonstrate the positive-correlated relationship between network capacity and its discrimination ability. Moreover, we quantitatively analyze the architecture effect and training skills effect on the network capacity. Finally, we examine the benefits from the optimization techniques in the training phase.

4.1. Experimental Setup

We choose the most prevalent deep neural network architectures as the benchmark in our experiments, including AlexNet [9], VGG Net [13], GoogLenet [10] and ResNet [4]. Specifically, the AlexNet and GoogLenet use the same configuration with their original work. We choose to use VGG-16 structure and construct ResNet-50. All our network are trained from scratch. Our platform is GPU server with 4 Nvidia Tesla P100 computing processors, each with 16 GMb memory. For the deep learning library, we apply Caffe [7] in all of our model training and inference pass. For our valid bits identification, we program a Python-based processing framework.

Algorithm 2 Valid Bits Identification Algorithm

Input: A trained \(L\)-layer deep learning model \(D_L\), auxiliary stimulating set \(A_s\) and original full-precision network accuracy \(e_0\)

Output: Total valid bits \(TVB\) and Average valid bits \(AVB\)

1. \(E \leftarrow \emptyset\) //Set to store statistics
2. \(E \leftarrow E \cup (M \mid D_L)\)
3. for \(a_i\) in \(A_s\) do
4. for \(i = 1\) to \(L\) do
5. \(a_i \leftarrow T^r(a_{i-1}) + \lambda_2 \min(W_i, −[W_i], [W_i] − W_i)\parallel 2\);
6. \(E \leftarrow E \cup \text{Statistics}(a_i)\);
7. end for
8. end for
9. \(Clusters \leftarrow \text{K-means} (E)\)
10. \(BWVec \leftarrow \text{BitWidthGenerator}(\text{size}(Clusters))\)
11. //Adaptive Bitwidth Quantization
12. for each \(i\) in \(BWVec\) do
13. for \(j = 1\) to \(\text{size}(Clusters)\) do
14. \(W_q(i) \leftarrow W_q(i) \cup Q_{BWVec(i,j)}(Clusters(j))\)
15. end for
16. if InformationLoss\((W_q(i)) < \eta\) then
17. continue
18. end if
19. \(acc \leftarrow \text{InferenceOnTest}(D_L(W_q(i)))\);
20. if \(acc = e_0\) and \(TVB < \sum W_q(i)\parallel 0\) then
21. \(TVB, AVB \leftarrow \text{Update}(W_q(i))\);
22. end if
23. end for
24. Return \(TVB\) and \(AVB\);

4.2. Network Capacity and Discrimination Ability

Our first experiment tends to examine the relationship between network capacity and model discrimination ability. In this part, we use our total valid bits to indicate the neural network model’s capacity and image classification rate, Top 5 Accuracy, as the model discrimination ability. We choose the best current deep neural network, ResNet-50, as
the testing bed. We collect all the statistics in the training phase, also including the value of total valid bits. The results are shown in the Figure 2.

![Figure 2](image.png)

Figure 2. The capacity and accuracy (Top 5) for ResNet-50 as the iteration increases.

We can observe that the network capacity and network discrimination ability increases correspondingly as the iteration goes. We use square marker to indicate the accuracy on the validation set and red triangle to show the change of network capacity. Basically, the training accuracy and network capacity are both increasing gradually. They also share the similar fluctuation trend. Especially, around the iteration 40K, the accuracy has a large-step jump. In the meantime, the total valid bits also exhibits a large-step jump. We can find that the change of model accuracy has positive-correlated relationship with network capacity, total valid bits. Therefore, the network capacity is a good indicator to characterize the network discrimination ability.

4.3. Network Architecture Impact on Network Capacity

In the literature, the network architecture design is highly depend on the experience of the skillful designer. But its impact on network capacity is still not clear. In this experiment, we apply total valid bits to help understand the insights of different effective architectures. Thus, we choose all four networks in our experiments, AlexNet, VGG Net, GoogLenet and ResNet. We collect the information of their total valid bits, average valid bits and final accuracy (Top 5) on the validation set. We illustrate all these information in Figure 3.

![Figure 3](image.png)

Figure 3. The capacity, AVB and accuracy (Top 5) for different state-of-the-art network architectures.

The most important observation is that the network capacity increases according to the order of AlexNet, VGG Net, GoogLenet and ResNet. There are also two large improvement between AlexNet and VGG Net, and also between GoogLenet and ResNet. This two turning points are also the accuracy achieving large improvements.

**Network Architecture Impact:** If we have a close look at the average valid bits (AVB), some insights can even be duged out. First, although the network capacity of VGG Net is larger than AlexNet, their AVBs are very close. This indicates that the VGG Net increases the activating neuron numbers, which results in larger network capacity. This conclusion can also be supported by the AVB of GoogLenet. It is the smallest in all the four network architectures, but its accuracy is only smaller than ResNet. This confirms that the design goal of GoogLenet is to approximate the optimal sparse structure. For the case of ResNet, its contribution includes both AVB and network capacity increasing. The shortcut connect transports the salient information advancing to the deeper layers, which not only activate more neurons to work, but also promote the activation strength of the working neurons.

4.4. Advanced Training Skills Impact on Network Capacity

4.5. Training Phase Optimization on Network Capacity Estimation

4.6. Network Capacity Repeatability

5. Conclusions and Future Work

In this work, we investigated the network capacity for deep models, which was a good indicator of network discrimination ability. Specifically, we proposed “total valid bits” to characterize the network capacity. To accurately retrieve the total valid bits, we designed optimization strategies in both training phase and inference phase. Our extensive experimental results showed that network capacity can properly represent the model discrimination ability. The network architecture and advanced training skills were also quantitatively explained from the network capacity perspective.

**References**

[1] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *Information Theory, 2004. ISIT*
2004. Proceedings. International Symposium on, page 31. IEEE, 2004.

[2] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115, 2014.

[3] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pages 448–456, 2015.

[6] A. K. Jain. Data clustering: 50 years beyond k-means. Pattern recognition letters, 31(8):651–666, 2010.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM international conference on Multimedia, pages 675–678. ACM, 2014.

[8] J. M. Joyce. Kullback-leibler divergence. In International Encyclopedia of Statistical Science, pages 720–722. Springer, 2011.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[10] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

[12] G. B. Orr and K.-R. Müller. Neural networks: tricks of the trade. Springer, 2003.

[13] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In European Conference on Computer Vision, pages 525–542. Springer, 2016.

[14] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[17] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4820–4828, 2016.