Pulse-Net: Dynamic Compression of Convolutional Neural Networks

1st Browne
Insight-Centre
University College Cork
Cork, Ireland
david.browne@insight-centre.org

2nd Giering
UTRC
United Technologies Research Center
East Hartford Connecticut, U.S.A.
gierinmj@utrc.utc.com

3rd Prestwich
Insight-Centre
University College Cork
Cork, Ireland
steven.prestwich-centre.org

***Abstract*—Convolutional Neural Networks (CNNs) are used in a range of computer vision tasks, with state-of-the-art CNNs such as AlexNet and VGG16 constructed using a large number of parameter and multiply-add operations (MACs). These tasks require high computational power and high energy requirements to run the CNNs, making them unsuitable for deployment on Internet of Things devices. To overcome this issue and facilitate the use of CNNs on these resource-constrained devices, compression technology through pruning research has gained momentum and is an important tool for improving performance during inference. Our work focuses on pruning unwanted filters and nodes in all layers of a network. The network is pruned iteratively during training via a novel approach we call Pulse-Net, and a significant number of filters/nodes are removed while ensuring any loss in accuracy is within a predetermined range. The unpruned network can be extracted from the original structure for the inference stage. This novel method has an easy-to-set parameter to control the trade-off between accuracy and compression. Pulse-Net gives greater compression, while maintaining competitive accuracy loss, than other reported methods like, efficient convnets, ThiNet and Cross-Entropy Pruning. It also has better robustness against adversarial attacks than other compression and pruning techniques.**

***Index Terms*—Deep Learning, Classification, Compression, Image Recognition, IoT, Inference Efficiency**

I. Introduction

Recent years have seen the explosion of increasingly deep neural networks, which achieve start-of-the-art results in areas including computer vision. The rapid growth of deep convolutional neural networks (CNNs) is due to hardware developments in the form of powerful GPUs, software developments in the form of stochastic gradient descent and new network architectures, and the public availability of large labelled datasets such as the ImageNet classification tasks.

However, because deep CNNs rely heavily on powerful GPUs and consume a great deal of memory, their practical uses may be limited. AlexNet [14] has about 61 million parameters and needs over 200 MB of storage, while VGG16 [24] has 138 million parameters requiring 500 MB. The more parameters the model has, the more memory it consumes and the more energy is needed during inference. This is particularly important when these networks are deployed on mobile devices, and memory consumption is the key resource for usage on the cloud. Cost and power requirements can also be important determining factors when considering introducing CNNs for Internet of Things (IoT) applications. Inference time can be just as important as accuracy for online image recognition where thousands of images per second may require analysis. We show that by compressing/pruning the network we can greatly reduce the number of parameters, leading to a significant reduction in the number of FLOPs, which is directly associated to inference time.

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

As deep neural networks become deeper and wider, methods to prune filters/nodes and compress the network structure have gained interest. Most recent work in CNN compression is focused on reducing the number and size of weights or parameters, but does not take into account the value of an entire filter or node. According to Denil *et al.* [4] and Hinton *et al.* [11], deep neural networks are known to be overparameterized, which facilitates convergence to good local minima of the loss function during training. After training, these redundant parameters can be removed with little loss in accuracy. There are two general approaches to compressing a network: during training [2], [3], [13] or after training [1], [6]–[8], [30]. Our proposal, called Pulse-Net, falls in the first group.

We demonstrate the robustness, efficiency and strength of our proposed approach on a range of computer vision tasks of varying degrees of difficulty (CIFAR-10, CIFAR100, Tiny-ImageNet), using well-known CNN structures of different depths and widths. We compress and evaluate these structures, showing how Pulse-Net can significantly reduce the model size, runtime and energy consumption, while approximately maintaining accuracy. The advantages of these compressed models are that they are easier to run on embedded IoT devices, need less energy for computation and use less bandwidth for updating. Adversarial images are tested on both the original and the compressed networks, demonstrating the effects of adversarial attacks on pruned networks.

The main contributions of this research are as follows. Firstly, we introduce a novel approach called Pulse-Net for pruning filters and nodes during training. Secondly, we demonstrate that this achieves close to state-of-the-art classification accuracy on the well-known datasets CIFAR-10, CIFAR100 and Tiny-ImageNet while using only a fraction of the parameters. Thirdly, we show how the compressed network can be extracted and loaded onto a new narrower CNN for the inference phase, and that simulations of its usage on an IoT device shows substantial improvements in computational speed and energy efficiency. Finally, robustness under adversarial attack of the compressed network created by Pulse-Net will be demonstrated, showing the compressed network to be just as accurate as the original network, and in some cases to have better accuracy.

This rest of this paper is organised as follows. Section 2 discusses the related work on filter pruning and network compression. We explain our proposed method Pulse-Net in Section 3, with the descriptions of the datasets and networks used in Section 4, along with the experimental design in Section 5. The results are given in Section 5, as well as their evaluation and discussions. Finally, we conclude the paper in Section 6.

II. Related Work

Research on the compression of deep neural networks has gained pace in the last couple of years, to enable these state-of-the-art networks to run on Internet of Things devices, and to use less bandwidth for updating. As already stated, CNNs are generally over-parameterized, so pruning them helps to improve generalization, as well as achieving compression and decreasing energy consumption with little loss in accuracy. Molchanov *et al.* [19], following the same idea, uses the Dropout regularization method to help decide which weights can be pruned.

Han *et al.* [7] and Guo *et al.* [6] trained a network then pruned redundant weights, resulting in a sparse network. To regain accuracy this sparse network was retrained. A disadvantage of this method is that additional libraries are required to utilize sparse networks. Also, though this can reduce the size of a network, it does not neccessarily make the network more efficient or faster at inference time. This is due to the structure of the networks studied, AlexNet [14] and VGG16 [24], in which the number of parameters in the fully connected layers dominates the number of parameters in the convolutional layers. Most of the computation time during inference is spent in the convolutional layers, so removing these redundant weights result in little improvement in inference time. The recently developed Highway Networks [28] and ResNets [10] counter this by removing the fully connected layers, but this does not help with computation time which greatly increases as networks become deeper. Li *et al.* [16] proposed a filter pruning method which is more hardware-compatible: we follow this line, but we also prune nodes in the fully-connected layers to achieve greater compression.

Han *et al.* [8] extended [7] by using quantization and Huffman coding in conjunction with network pruning, an approach called Deep Compression. To show its application value they released a hardware accelerator called Efficient Inference Engine where the compressed model runs more energy-efficiently [9]. Polyak and Wolf [20] pruned their network based on low-level channels, while Sun *et al.* [27] learned a sparse network which reduced the number of parameters by 88%. The ideas of Han *et al.* [8], Courbariaux *et al.* [3] and Rastegari *et al.* [21] work could be applied to our compressed networks to further reduce their size.

Some recent state-of-the-art CNN architectures reduce the network parameters by reducing filter size. The VGG network [24] uses 3×3 filters, while googleNet [28] and Network-in-Network [17] both use 1×1 filters in some layers. Other recent research in CNN structure shows that, although adding layers increases accuracy, it is possible to skip layers in the network, resulting in further improvement in accuracy [10] [25].

Hinton *et al.* [12] used the outputs from a large pre-trained teacher network to train a smaller student network, with similar accuracy but faster computation. A drawback of this approach is that it requires a pre-trained network, and that further training operations are carried out on the large teacher network. Both Romero *et al.* [22] and Luo *et al.* [18] expanded on this idea, the former using not only the outputs but also representations learned by the teacher network to train a narrower and deeper student network. [18] used the weights learned by the teacher at the last hidden layer before the softmax layer, reasoning that these weights are highly correlated to the prediction classes.

Yang *et al.* [31] introduced energy consumption as the metric to decide which layers to prune. This work shows that the energy a model consumes has two components: (1) energy used to carry out the MAC operations, and (2) energy needed for memory accesses. We report how Pulse-Net can effectively reduce (1) by reducing the number of MAC calculations, and reduce (2) by shrinking the network structure.

Both Wang *et al.* [29] and Ye *et al.* [32] showed that compressed CNNs can be vulnerable to adversarial attack. This is shown by testing adversarial images created using the Fast Gradient Sign Method, on both the original network and the network compressed by 60%. The accuracy of both networks were greatly reduced and the difference in accuracy between them was over 13%. This showed that their compressed network was more vunerable to adversarial attack. We will show that Pulse-Net can reduce the network 95.63% while maintaining similar accuracy during adversial attacks, in some cases even beating the original network.

III. Pulse-Net

In this section we describe Pulse-Net, our network pruning technique, and show the pseudo code behind the idea.

Pulse-Net, so-called due to the pulsing nature of the number actively updated model parameters in time during training, is the main algorithm of our pruning method. It prunes the network iteratively, but also allows the network to expand when the loss of accuracy is too great. To check when the network converged, a moving average with window size 10 was used to smooth out the validation loss curve. The learning rate list was $[0.1, 0.01, 0.001, 0.0001, 0.00001]$ (lr-list for short in Algorithm 1). This compression and decompression of the network allows Pulse-Net to prune the network intelligently, and approach a compressed state more gently. This idea follows the use of cooling schedules in Simulated Annealing, a well-known optimization algorithm with good convergence properties. It allows Pulse-Net to explore ambitious pruning but focus on pure improvement when this fails. Another optimization algorithm that Pulse-Net takes properties from is NeuroEvolution of Augmenting Topologies (NEAT) [26] in which neurons can be added as necessary to regain accuracy. The full training process of Pulse-Nets varies depending on the compression rate achieved, but on average takes 2–5 times longer than training a model without compression. But once the model is compressed, the inference runtime is far more efficient than an uncompressed model.

Once the network is trained, all its layers are reduced by the same percentage. This ensures that the network is pruned more quickly, by pruning both the convolutional and fully-connected layers together. The network is fine-tuned until it stabilises, normally only requiring a few loops of the training data, then the procedure is repeated until set break-points are met or the loss in accuracy is too great.in each layer, the filters/nodes which have the lowest average value are removed first. During the training stage, a binary mask matrix is used to simulate the removal of the corresponding filters/nodes from these redundant feature maps. Once Pulse-Net has finished the reduction stage, the remaining relevant filters/nodes are extracted and reloaded onto a compressed network for the inference stage.

We now introduce some notation. Let α be the rate of reduction, and set it to 10%. This is the percentage all layers in the network that will be pruned by until an accuracy limit is breached. Let λ be the maximum loss in accuracy which allows the network to continue pruning at the current α rate. This is the parameter that controls how much a network is willing to sacrifice accuracy for greater compression. For all experiments in this work we set it to 2%, which gave high compression with very little accuracy loss. Let β be the stopping citerion: we set it to 2 nodes/filters of the widest current layer of the network. This means that if α prunes less than β in this layer, Pulse-Net stops trying to further compress the network. The pseudocode for Pulse-Net is shown in Algorithm 1.

Algorithm 1 Pulse-Net

```
Initialize lr-step $= 0$
Initialize lr-rate $=$ lr-list[lr-step]
Train Network until validation loss convergence
X = Calcul(10-ma)
Calculate validation acc and store as best acc
Repeat until # Filters/Nodes of max layer removed $< \beta$:
    While $|$validation acc $-$ Best acc$| < \lambda$:
        Remove $\alpha$ min[Filters] in all layers of Network
        Fine-Tune Network till validation loss converges
        $\mathrm{X} = \mathrm{Calcul}(10 - \mathrm{ma})$
        Calculate validation acc
        If validation acc $>$ best acc:
            best acc $\leftarrow$ validation acc
        Else:
            If lr-step $<$ length ( lr-list )
                $\alpha = 0.5(\alpha)$
                lr-step $=$ lr-step $+1$
            Else:
                Halt
```

IV. Datasets and Networks

We now describe the datasets and network architectures used in our experiments.

A. Datasets

We used 3 well-known datasets to demonstrate the effectiveness of Pulse-Net: CIFAR10 [15], CIFAR100 [15] and Tiny-ImageNet. Tiny-ImageNet is a subset of the ILSVRC2014 dataset containing 200 classes created by Lucas Hansen at Stanford University [23]. The training dataset has 500 images per class, while the testing dataset has 50 images per class. Each image is 64×64 pixels and is an RBG color image. Both the CIFAR10 and CIFAR100 datasets are 32×32 pixels and are also RBG color images. CIFAR10 has 10 classes with 5000 images in each class in the training data and 1000 images per class for the testing dataset. CIFAR100 is made up of 100 classes, each class in the training datset has 500 examples, while each class in the testing dataset has 100 examples.

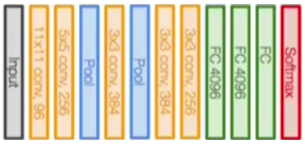

Fig. 1. The AlexNet architecture used for experiments

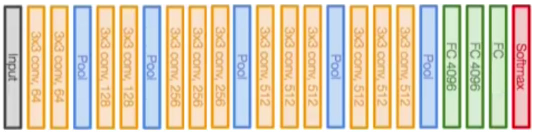

Fig. 2. The VGG16 architecture used for experiments

B. AlexNet

A slightly altered version of AlexNet was used for this work, as shown in figure 1. The network had 5 convolutional layers followed by 3 fully-connected layers with the last layer being the number of classes in the dataset. The activation function used in all layers except for the last layer, where softmax was used, was the recified linear unit (ReLu). Batch normalization (BN) was also used in all layers except the final layer.

C. VGG16

The VGG16 Network had 13 convolutional layers connected to 3 fully-connected layers with the last layer being the number of classes in the dataset, as shown in figure 2. Again the activation function used in all layers except for the last layer, where softmax was used, was ReLu; and BN was used in all layers expect the final layer.

D. TensorFlow 2 convolutional layer Network

The TensorFlow convolutional Network had 2 convolutional layers both with 64 filters, with a kernel size of 5 in both layers. Max pooling was used after both convolutional layers. Finally, the convolutional layers were connected to 3 fully-connected layers of sizes 384, 192 and the number of classes in the dataset. As above, the activation function used in all layers except for the last layer, where softmax was used, was ReLu; and BN was used in all layers expect the final layer.

V. Results

This section is broken down into 4 subsections: an outline of the experimental design, followed by a subsection for each dataset. In each subsection we will compare the accuracy, multiply-add operations (MACs), storage, computational speed and energy efficency of the compresseed and uncompressed networks. Note that an accuracy loss threshold of 2% was enforced during the pruning, meaning the network was allowed to decrease in accuracy by this amount during training to prune the network to a highly compressed state. The results in this section of the accuracy and performance of each test is carried out on a single image put through the network, using the experimental design described above.

A. Experiment Design

All training and testing was carried out on the NVIDIA GeForce GTX 1080 graphics card, which has 8GB of memory. The OS used was Ubuntu 16.04.3, Python version was 3.5.2 and TensorFlow version was 1.4.A mini batch size of 128 was used on all experiments, and to create the validation dataset, the last 10% was used. The Stochastic Gradient Descent (SGD) optimizer was used during the training phases with a repeated step learning rate method, explained above in section 3.2.

To check the robustness of the networks under adversarial attack, we also test them on adversarial images constructed using the Fast Gradient Sign Method [5]. We created adversarial images of CIFAR10, CIFAR100 and tiny-Imagenet using separately trained models of the TensorFlow network, AlexNet and VGG16, dedicated to creating only these adversarial test sets. The full validation sets of these datasets were used as the adversarial test sets. To avoid confusion we name these adversarial networks Adv-TensorFlow, Adv-AlexNet and Adv-VGG16. We compute statistics showing the noise required to create the adversarial images needing the most and least amount of noise for each model per dataset. These statistics include mean square error, entropy and structural similarity, which is used to compare images for likeness. Standard statistics such as interquartile range, mean and standard deviation show the variation of noise required for the adversarial effect, while the range and outliers show the max and min values of noise added, along with the extreme values.

B. CIFAR10

It can be seen from table I that Pulse-Net was able to reduce AlexNet by 95.63% and VGGG16 by 87.85%, sacrificing only 2% and 0.9% accuracy respectively. The TensorFlow model was pruned by 76.25% with a decrease in accuracy of 3.6%. It is worth remembering that this model was designed specifically to demonstrate image recognition tasks using the CIFAR datasets, and was already an optimally sized network. The relevant filters and nodes of these pruned networks were extracted and reloaded onto a network suited to their size. These reduced networks were then tested for efficiency, displaying substantial savings in storage and up to 50% reduction in inference time, with a saving of between 14% and 65% in energy required for inference.

These extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics from table II describe the noise added to the images that required both the maximum and minimum amount of variation needed to create the adversarial images to fool the network. The images where most noise was added were deer (2995), airplane (7835) and frog (1578), while the images requiring the least amount were airplane (2473), ship (3140) and car (6010) on the networks Adv-AlexNet, Adv-TensorFlow and Adv-VGG16, respectively. It can be seen that there was a great difference between the amount of noise required between creating the max and min adversarial images. This is highlighted in the results ofthe MSE statistic and the R-Squared values, along with the noise range values.

Table III shows accuracy results for the adversarial attacks using the CIFAR10 dataset. Taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unprune models, the table shows that out of the 9 attacks, 6 of Pulse-Net's pruned models were more robust than the original models. This illustrates that for the CIFAR10 dataset Pulse-Net created networks were not only more efficient but also over 66% more robust to adversarial attacks.

C. CIFAR100

It can be seen from table IV that Pulse-Net was able to reduce AlexNet by 85.95% and VGG16 by 87.6%, with only a loss in accuracy of 2.3% and 1.1% respectively. The TensorFlow model was pruned by 65.57% with a decrease in accuracy of 4.5%. Again, this network was already optimised. After extraction these reduced networks were tested for efficiency, and like the CIFAR10 results showed great improvement in storage, reducing the inference times by between 0.6% and 48.33%, and with a saving of energy between 9% and 67%.

Following the same testing method as for the CIFAR10 dataset, these extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics for the CIFAR100 dataset follow a very similar pattern to those for CIFAR10, and therfore due to page restrictions are omitted, but available on request.

Table V shows the accuracy results from the adversarial attacks using the CIFAR100 dataset. Again, taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unprune models, the table shows that out of the 9 attacks, 8 of Pulse-Net's pruned models were more robust than the original models. This

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Speed (ms)	Energy (mJ)
AlexNet	91.16	5.83 X 107	874	222.5	4.14	0.95
Compressed	2%	95.63%	95.31%	95.63%	50.72%	65.26%
TensorFlow	85.36	1.07 X 106	18.4	4.08	1.69	0.39
Compressed	3.6%	76.25%	72.83%	76.25%	2.95%	13.59%
VGG16	90.87	3.36 X 107	287.2	128.36	6.32	0.84
Compressed	0.9%	87.85%	87.89%	87.85%	47.94%	59.52%

TABLE I

PERFORMANCE AND ACCURACY OF TENSORFLOW MODEL, ALEXNET AND VGG16 USING CIFAR10 DATASET

	Adv-AlexNet		Adv-TensorFlow		Adv-VGG16	
Statistic	Maximum	Minimum	Maximum	Minimum	Maximum	Minimum
Noise range	[-72, 161]	[-12, 15]	[-82, 82]	[-12, 13]	[-158, 151]	[-14, 12]
IQR	11	0	2	0	14	2
Mean	0.23	0.009	-0.18	-0.004	0.05	0.04
STD	21.45	2.03	11.96	2.1	18.46	2.5
Outliers	655	1420	717	1361	296	250
MSE	1381.04	13.32	428.98	13.29	1021.94	18.71
Structural Similarity	0.27	0.98	0.72	0.98	0.8	0.98
Entropy	5.86	2.52	3.83	2.5	5.83	3.12

TABLE II

STATISTICS OF THE CIFAR10 ADVERSARIAL IMAGES THAT REQUIRED THE MAXIMUM AND MINIMUM AMOUNT OF NOISE ADDED

	Adv-AlexNet		Adv-TensorFlow		Adv-VGG16	
Network	Original (%)	Compressed (%)	Original (%)	Compressed (%)	Original (%)	Compressed (%)
AlexNet	20.08	20.74	31.61	20.51	42.69	35.11
TensorFlow	37.90	39.49	9.89	9.38	45.94	44.21
VGG16	40.92	43.68	37.54	36.06	29.26	29.34

TABLE III

ACCURACY RESULTS OF CIFAR10 ADVERSARIAL IMAGES CREATED BY TENSORFLOW MODEL, ALEXNET AND VGG16

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Speed (ms)	Energy (mJ)
AlexNet	69.77	5.87 X 107	874.5	223.9	4.16	1.04
Compressed	2.31%	85.95%	85.76%	85.95%	46.15%	67.46%
TensorFlow	58.05	1.09 X 106	18.5	4.14	1.68	0.40
Compressed	4.5%	65.57%	63.03%	65.57%	0.6%	8.79%
VGG16	64.2	3.4 X 107	287.6	129.77	6.29	0.83
Compressed	1.14%	87.6%	87.87%	87.6%	48.33%	57.52%

TABLE IV

PERFORMANCE AND ACCURACY OF TENSORFLOW MODEL, ALEXNET AND VGG16 USING CIFAR100 DATASET

illustrates that for the CIFAR100 dataset Pulse-Net created networks were not only more efficient but were also nearly 90% more robust under adversarial attacks.

D. Tiny-ImageNet

Finally, looking at table VI, Pulse-Net was able to reduce AlexNet by 80.79% and VGG16 by 83.15%, with only a loss in accuracy of 3.75% and 2% respectively. The TensorFlow model was pruned by 74.25% with a decrease in accuracy of 2.91%. The reduced extracted networks were tested for efficency, and like both CIFAR results showed great improvement in storage, reduced the inference times by between 15% and 57%, and a saving of energy between 13% and 65%.

Following the same testing method as the CIFAR datasets, these extracted networks pruned by Pulse-Net were analysed for robustness against adversarial attacks. The statistics for the Tiny-ImageNet dataset follow a very similar pattern to those of CIFAR10, and therfore due to page restrictions are omitted, but available on request.

Table VII shows the accuracy results from the adversarial attacks using the Tiny-ImageNet dataset. Again, taking into account the accuracy of the networks on the original images, and the relative difference between the pruned and unprune models, the table shows that out of the 9 attacks, 6 of Pulse-Net's pruned models were more robust than the original models. Again, displaying that for the Tiny-ImageNet dataset Pulse-Net created networks were not only more efficient but were also over 66% more robust to adversarial attacks.

VI. CONCLUSION

In this paper, we proposed a novel deep CNN pruning method called Pulse-Net, which compresses a network during training to create a more efficient model for inference. The proposed compression method shows significant improvements in storage, inference timings and energy efficiency, as well as greater robustness under adversarial attack, ideal for IoT devices.

In future work we would like to explore different metrics for pruning, as well as removing filters in other types of networks like ResNet. In addition, we believe research into pruning the depth of networks as well as the width, using Pulse-Net, would be an interesting expansion of the work.

	Adv-AlexNet		Adv-TensorFlow		Adv-VGG16	
Network	Original (%)	Compressed (%)	Original (%)	Compressed (%)	Original (%)	Compressed (%)
AlexNet	14.90	17.25	18.48	15.84	32.38	30.10
TensorFlow	26.47	26.03	4.14	4.28	26.92	25.22
VGG16	29.35	30.71	20.56	20.13	19.23	22.29

TABLE V

ACCURACY RESULTS OF CIFAR100 ADVERSARIAL IMAGES CREATED BY TENSORFLOW MODEL, ALEXNET AND VGG16

Network	Accuracy (%)	Parameters	MACs (M)	Storage (MB)	Speed (ms)	Energy (mJ)
AlexNet	54.8	7.27 X 107	1266.2	277.47	8.68	1.32
Compressed	3.75%	80.79%	79.20%	80.79%	56.57%	65.15%
TensorFlow	40.45	5.04 X 106	100	19.22	2.1	0.31
Compressed	2.91%	74.25%	71.26%	74.25%	15.24%	12.9%
VGG16	56.05	4.07 X 107	1010.8	155.33	6.67	0.86
Compressed	2%	83.15%	83.81%	83.15%	44.68%	61.63%

TABLE VI

PERFORMANCE AND ACCURACY OF TENSORFLOW MODEL, ALEXNET AND VGG16 USING TINY-IMAGENET DATASET

	Adv-AlexNet		Adv-TensorFlow		Adv-VGG16	
Network	Original	Compressed	Original	Compressed	Original	Compressed
AlexNet	40.93	36.99	39.15	33.77	47.29	42.49
TensorFlow	32.79	30.88	16.02	15.95	34.86	32.61
VGG16	46.89	45.34	43.16	41.77	30.49	35.54

TABLE VII

ACCURACY RESULTS OF TINY-IMAGENET ADVERSARIAL IMAGES CREATED BY TENSORFLOW MODEL, ALEXNET AND VGG16

ACKNOWLEDGMENT

This work was supported in part by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and also in part by United Technologies Research Center.

REFERENCES

[1] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, Y. Chen. ”Compressing neural networks with the hashing tr, ” *In ICML*, 2015.

[2] M. Courbariaux, Y. Bengio, J.P. David. ”Binaryconnect: training deep neural networks with binary weights during propagations,” *In NIPS*, 2015, pp. 3105-–3113.

[3] M. Courbariaux, I. Hubara, D.l Soudry, R. El-Yaniv, Y. Bengio.” Binarized neural networks: Training neural networks with weights and activations constrained to +1 or -1,” *arXiv:1602.02830*, 2016.

[4] M. Denil, B. Shakibi, L. Dinh,M. A. Ranzato, N. de Freitas, ”Predicting parameters in deep learning,” *In Advances in Neural Information Processing Systems*, 2014. pp. 1269-–1277.

[5] I. J, Goodfellow, J. Shlens, and C. Szegedy. ”Deep residual learning for image recognition,” *arXiv:1512.03385*, 2015.

[6] Y. Guo, A. Yao, and Y. Chen.” Dynamic network surgery for efficient dnns,” *In NIPS*, 2016, pp. 1379-–1387.

[7] S. Han, J. Pool, J. Tran, W. Dally.”Learning both weights and connections for efficient neural networks,”*In NIPS*, 2015.

[8] S. Han, H. Mao, W. Dally.”Deep compression: Compressing DNNs with pruning,trained quantization and huffman coding,”*arxiv:1510.00149*, 2015.

[9] S. Han, Xi. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, W. J. Dally. ”Eie: Efficient inference engine on compressed deep neural network,” *International Symposium on Computer Architecture (ISCA)*, 2016.

[10] K. He, X. Zhang, S. Ren, J. Sun. ”Deep residual learning for image recognition,” *arXiv:1512.03385*, 2015.

[11] G. E. Hinton, N.Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov. ”Improving neural networks by preventing co-adaptation of feature detectors,” *arXiv:1207.0580*, 2012.

[12] G. Hinton, O. Vinyals, J. Dean. ”Distilling the knowledge in a neural network,” *In NIPS Workshop*, 2014.

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, K. Keutzer. ”Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 1mb model size, ” *arXiv:1602.07360*, 2016.

[14] A. Krizhevsky, I. Sutskever, G. E. Hinton. ”Imagenet classification with deep convolutional neural network,” *In NIPS*, 2012, pp. 1097-–1105.

[15] A. Krizhevsky, G. E. Hinton.” Learning multiple layers of features from tiny images,” *Master's thesis*, Department of Computer Science, University of Toronto, 2009.

[16] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf. ”Pruning Filters for Efficient ConvNets,” *arXiv:1608.08710*, 2016.

[17] M. Lin, Q. Chen, S. Yan. ”Network in network,” *arXiv:1312.4400*, 2013.

[18] P. Luo, Z. Zhu, Z. Liu et al, ”Face model compression by distilling knowledge from neurons,” *AAAI*, 2016.

[19] D. Molchanov, A. Ashukha, D. Vetrov. ”Variational dropout sparsifies deep neural networks,” *arXiv:1701.05369*, 2017.

[20] A. Polyak, L. Wolf. ”Channel-level acceleration of deep face representations,” *IEEE*, 2015.

[21] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi. ”XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks,” *In ECCV*, 2016.

[22] A. Romero, N. Ballas, Y. Bengio et al. ”Fitnets: Hints for thin deep nets,” *In ICLR*, 2015.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M Bernstein, A.C. Berg. ”Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, 115(3) 2015, pp. 211–52.

[24] K. Simonyan, A. Zisserman.” Very deep convolutional networks for large-scale image recognition,” *In ICLR*, 2015, pp. 1-–14.

[25] R. K. Srivastava, K. Greff, J. Schmidhuber. ”Inception-v4, inception-resnet and the impact of residual connections on learning,” *arXiv:1602.07261*, 2016.

[26] K. O. Stanley, R. Miikkulainen. ”Evolving Neural Networks Through Augmenting Topologies,” *Evolutionary Computation*, 10(2) 2002, pp. 99–127.

[27] Y. Sun, X. Wang, X. Tang. ”Sparsifying neural network connections for face recognition,” *IEEE CVPR*, 2016.

[28] C. Szegedy, S. Ioffe, V. Vanhoucke. ”Highway networks,” *In ICML Deep Learning Workshop*, 2015.

[29] L. Wang, G. W. Ding, R. Huang, Y. Cao, C. Yanshuai, Y. Lui. ” Adversarial Robustness of Pruned Neural Networks,” *In ICLR*, 2018.

[30] W. Wen, C. Wu, Y. Wang, Y. Chen, H. Li. ”Learning structured sparsity in deep neural networks,” *In Advances In Neural Information Processing Systems*, 2016, pp. 2074-–2082.

[31] T. J. Yang, Y. H. Chen, V. Szei.”Designing energy-efficient convolutional neural networks using energy-aware pruning,”*arXiv:1611.05128*, 2016.

[32] Y. Shaokai, S. Wang, X. Wang, B. Yuan, W. Wen, X. Lin. ” Defending DNN Adversarial Attacks with Pruning and Logits Augmentation,” *In ICLR Workshop*, 2018.