Highly Scalable Deep Convolutional Neural Networks with Sparsely Aggregated Internal Connections

Siddharth Nijhawan
Department of Electrical Engineering
Columbia University
UNI: sn2951
sn2951@columbia.edu

Sushant Tiwari
Department of Electrical Engineering
Columbia University
UNI: st3425
st3425@columbia.edu

Aman Anand
Department of Electrical Engineering
Columbia University
UNI: aa4821
aa4821@columbia.edu

***Abstract*—The concept of sparsity in neural network parameters and architectures has been revolutionizing the world of deep convolution neural networks. Building a neural network architecture through sparse aggregation via internal skip connections can help not only in the significant reduction in training time but also in the computational complexity and convergence. Present state-of-the-art network architectures like ResNet and DenseNet focus on skip connections but not in an efficient manner. This leaves lot of scope in implementing sparsity with a focus on the selected layer-wise inputs for sparse aggregation in these architectures. Our work builds on this sparse aggregation concept at the architecture level by choosing only selected inputs at a given depth for propagating the most important information to the successive layers. Through our computational experiments and in-depth analysis, we propose an architecture design which combines the residual skip connections with the selected parameter sparse aggregation to train highly deep convolution neural networks. Through our results on image classification datasets CIFAR-10 and CIFAR-100, we argue that our design implementation gives better performance in terms of training time, training parameters and overall classification accuracy for a given layer depth and the feature growth rate. Also, we analyse the performance of our implementation of the architecture design called SparseNet with DenseNets on CIFAR-10 and CIFAR-100 datasets through experimentation and with ResNet and FractalNet analytically. We compare these architectures in terms of computational complexity by performing in-depth sparse aggregation analysis. We build a highly robust and scalable model by only aggregating a selected/sparse penultimate outputs sets for a current depth and thereby achieving reduction in number of incoming input links. This decrease in the link size from linear to logarithmic helps in a significant improvement in the above mentioned performance metrics**

***Index Terms*—SparseNets, DenseNets, ResNets, FractalNets, Training time, Computational Complexity, Sparse Aggregation, Concatenation, Feature Growth Rate**

I. Introduction

In a typical deep Convolutional Neural Network, the outputs of layers which are present at the beginning of the network are used by the deeper layers for forward as well as backward propagation during training and inference. A key aspect of this methodology is forming a pattern related to the use of skipping internal connections for output aggregation between penultimate layers and next layers.

We plan to modify this architectural dependency to improvise upon the scalability of a deep neural network. Since the step of aggregation is really important to train a deep neural network from in an end-to-end fashion, we plan to investigate various aggregation methodologies like concatenation, addition, etc. and focus our attention on how to decide whether to aggregate a particular connection or not. Therefore, by aggregating only a sparse set of penultimate outputs at any given depth, we plan to achieve an overall reduction in number of incoming links from linear to logarithmic, thus drastically reducing the total amount of trainable parameters with the increase in depth of the network.

Through our in-depth analysis, we compare the architectural dependencies and fine nuances with respect to the parameter sparsity in ResNets and DenseNets with SparseNets. Through our design, we observe following performance improvements: 1) For a given accuracy on the image classification datasets, SparseNet model achieves a much lower computational complexity and a higher efficiency in terms of number of operations as well as the number of trainable parameters. 2) For a given feature growth rate and CNN layer depth, with SparseNet, we achieve a model with higher accuracy and/or fewer trainable parameters as compared to DenseNet architecture. 3) We propose a highly scalable and robust architectural design for our sparse aggregation implementation thereby a model, which can be applied to different ResNet models as well as on complex datasets. We propose that the linear to logarithmic implementation of the incoming links in a neural network, will infact even better the performance of the baseline ResNet model. 4) We, thereby analyse the factors helping in analyzing the sparse implementation in different model architectures and argue that the sparse aggregation concept is independent of the architectural property difference across different architectures and hence can be independently applied to improve the computational performance.

A. Related Work

Implementation of sparsity in convolutional neural networks has been till now, limited to the use of different types of aggregations in the skip link connections[9]. This aggregation basically combines the previous layer inputs at every depth for further information propagation to the successive network depths.Prior research has focused on implementation of several operations like regularization, batch normalization, activation functions and optimization functions enabling the model to achieve better generalization and faster convergence [2][8]. Studies which highlight employing features like dimensionality reduction using Principle Component Analysis (PCA), L1 regularization or changing the kernel size have been effective in reducing the number of trainable parameters but these in some way or the other, are network and function dependent [1].

Hu et al. proposed the similar concept of achieving sparsity by sparse aggregation but did not achieve much improvement as compared to the baseline DenseNet accuracy on the image classification datasets. Our SparseNet image classification results are substantially better than those reported in Hu et al [10]. Our results represent an actual and significant improvement over the DenseNet baseline. The work of Xiao et al [3] focused on developing and analyzing the robustness of reverse networks using differential equations which will allow us to train deep convolutional neural networks.

II. Analysis and Comparison of Neural Network Aggregation Architectures

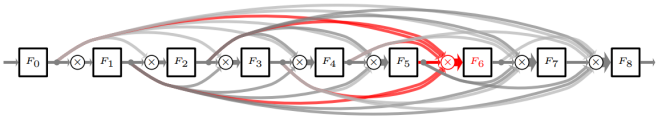

Fig. 1. DenseNet/ResNet topology: Dense Aggregation [6]

From the figure of DenseNet topolgy, we can deduce that the for a given neuron layer function involving convolution + Non-Linear Activation + Batch Normalization, N number of links terminate on a given node i.e. for a layer, there are N incoming connections hence a quadratic complexity is obtained. The aggregation operation in ResNets is addition whereas in DenseNets and FractalNets, it is concatenation and averaging respectively. From the FractalNet topology, it can be seen that for a given layer, N incoming links add on to the N/2, N/4, N/8...links resulting in the network of of 2N - 1 layers where N is network's depth [6]. Besides, there is an extra aggregation joining these small networks, which make the total complexity of the FractalNet as 2N.

There are some drawbacks in the ResNet and FractalNet topology. In ResNet topology, once we employ the aggregation operation, we cannot separate out the individual features [6].

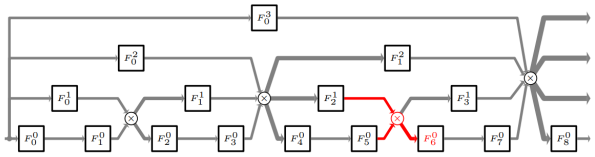

Fig. 2. FractalNet topology: Fractal Aggregation [6]

Hence, this makes it difficult to recover or transform the features once it has been aggregated or summed, resulting in the information loss in deeper neural networks having more than 1200 layers. Because of this, even ResNet architecture will under some information loss on training highly deep neural networks. Besides, design of the features which are combined, have no dynamism or flexibility and this static design affects the performance of the ResNet architectures[4].

DenseNet, on the other hand, suffers from high quadratic computational complexity of $O(N^2)$ as all the incoming links are included in aggregation for a specific layer. This means that for successive layers there is very little addition of useful information with successive layers. Also, most of bypass networks have kernel weights very close to null and hence in DenseNets some non-essential states are also mapped.

There are several variations of the DenseNet/ResNet topology but these just differ in the use of aggregation function and thus these architectures will also suffer from the problem of the dense aggregation described above [5][7]. Our design implementation of the sparse aggregation will result in improvement of the complexity from $O(N^2)$ to Nlog(N) which will give a better performance for a given network depth.

III. Proposed Methodology - Technical Approach

This section analyzes our proposed sparse aggregation implementation in terms of the computational complexity, trainable parameters and the image classification datasets used for performing the experiments.

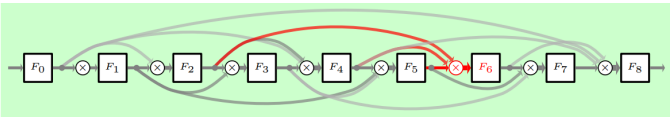

Fig. 3. SparseNet topology: Sparse Aggregation.[6]

We propose a sparse aggregation architecture where for a layer l, the sparse aggregation output depends on (l - m^k) inputs where $k > 0$ such that (l - m^k) is non-negative. This means that the output at the layer l depends on selected sparse inputs which are decided by the power of m such that (l - m^k) is greater than 0. This strategy of sparse aggregation

will, in worst case, have log(N) connections for a present layer, where N is the network's depth. Hence, under this assumption, for N layers, the complexity of the network will become O(Nlog(N)).This is an improvement over $O(N^2)$ time complexity obtained in the dense nets where all the previous connections are used in the aggregation output calculation. This sparse aggregation design is having higher sparsity as compared to the DenseNet design but has a less sparse architecture as compared to the FractalNet architecture.

CNN Architecture	Parameters	Gradient Path (Complexity)
SparseNet (Summation)	O(N)	O(log(N))
SparseNet (Concatenation)	O(Nlog(N))	O(log(N))
DenseNet	$O(N^2)$	O(1)
ResNet	O(N)	O(1)

Fig. 4. Complexity for different architectures: ResNets, DenseNets and SparseNets

SparseNets have shorter paths for the gradients (as calculated during backward propagation) because the links for a current layer are only connected in the exponential manner to previous layers [6]. The number of estimated training parameters in the SparseNet architecture depends on the aggregation methodology i.e. if previous inputs are concatenated in the exponential manner, then the parameters are of the order of O(Nlog(N) whereas if the summation sparse aggregation is used, then the parameters are of the order of O(N). Also, ResNet and DenseNets have constant O(1) complexity for the gradient path (shortest). For the neural networks which are plain and do not have any aggregation functions or skip connections, both the parameters and gradient paths have O(N) complexity.

Our approach involves building the SparseNet architecture using baseline hyperparameters and then using CIFAR-10 and CIFAR-100 data sets to analyse the architectural dependencies and fine nuances with respect to the parameter sparsity in ResNets, FractalNets, DenseNets and compare it with the proposed SparseNets topology. Through our computations, we test and analyze the effect of sparse aggregation topology (from linear to logarithmic) on SparseNet architectures. Rest of the hyperparameters are kept the same at the baseline level for both the image classification datasets- CIFAR-10 and CIFAR-100 and for both the architectures. This sparse aggregation architecture is scalable, robust and will improve the performance of the state of the art ResNets in training the layers which are many thousand layers deep.

We used following datasets on SparseNet and DenseNet architectures:

A. Data

CIFAR-10 and CIFAR-100 datasets are used in implementation of proposed sparse aggregation design. These datasets are widely popular as the baseline datasets for testing the performance of the neural networks across different architectures.

1) CIFAR-10: There are 60,000 total color images in CIFAR-10 out of which 10,000 are the test images and remaining 50,000 are the training images. There are total 10 classes and each class is having 6000 images. Each color image is of the dimension 32x32.There is no overlap between any class.

2) CIFAR-100: There are 60,000 total color images in CIFAR-100 out of which 10,000 are the test images and remaining 50,000 are the training images. There are total 100 classes and each class is having 600 images. Each class consists of 500 training images and 100 testing images. Each color image is of the dimension 32x32. There is no overlap between any class.

IV. Experiments

We performed experiments on different combinations of model type, network layer depth (N) and feature growth rate (k) where k is defined such that for $m > 0$, (1 - m^k) is non-negative. Performance of the SparseNet with the sparse aggregation topology is checked on CIFAR-10 and CIFAR-100 image classification datasets. Bottleneck Compression (BC) flag is set to True for some cases and False for other. We test our model architecture on 2 depths (N=40 and N=100). For each depth, the value of feature growth rate (k) is varied. This controls the amount of sparsity in the aggregation of SparseNet architecture. Higher the value of k, more the number of incoming connections to a specific layer and higher will be the number of parameters. The total number of parameters and the test accuracy are noted for various combinations of above parameters. The average training time per epoch is plotted and analyzed for both the model architectures . The entire pipeline described above are repeated for the DenseNet architecture. We tabulate the performance metrics of Testing Accuracy and Total Training Parameters for the different combinations of above parameters.

Here, our main objective is to analyze and compare the performance of sparse aggregation topology implemented in SparseNet with the Dense aggregation architectures like DenseNet. Because of the objective, we keep other hyperparameters like learning rate, batch size, optimizer used and momentum constant. Otherwise, if these model specific parameters are varied then we would not be able to compare the response of these architectures on the base set of conditions for image classification datasets.

The models are implemented in the PyTorch framework. Different models were trained using NVIDIA T4 Tensor Core GPU on the Google Cloud Platform (GCP). Model hyperparameters are initialized with the following values: validation train split ratio of 0.9, epochs are varied depending

on the stability/convergence for different configurations, learning rate of 0.1, batch size of 32, Optimizer type = Stochastic Gradient Descent (SGD) with momentum and momentum value of 0.9. SGD with momentum optimizer is chosen as the base model as it gives better generalization as compared to the adaptive gradient optimizers with a decent convergence rate for given model hyperparameter settings.

V. Observations and Results

A. CIFAR-10: SparseNet vs DenseNet model configurations

Model Type	BC	Depth (N)	Feature Growth Rate (k)	Total Params	Accuracy
SparseNet	Yes	40	12	0.1242M	83.0
	Yes	100	24	1.4378M	87.0
	Yes	100	36	3.2199M	86.0
	Yes	100	64	10.1343M	87.0
	Yes	100	[16, 32, 64]	4.3337M	87.0
	No	100	24	2.4773M	87.0
	No	100	36	5.5550M	86.0
DenseNet	Yes	40	12	0.1761M	84.0
	Yes	100	24	3.0205M	88.0
	Yes	100	36	6.7540M	88.0
	Yes	100	64	21.2297M	84.0
	Yes	100	[16, 32, 64]	7.7418M	84.0
	No	100	24	28.0717M	87.0
	No	100	36	62.9645M	83.0

Fig. 5. Performance Metrics: Parameters and Accuracy for CIFAR-10

To get the trend, we first check the performance of the SparseNet model on a shallow network i.e. N = 40. The value of feature growth rate k is set to 12 with Bottle Compression Flag = True. The model achieved a decent accuracy of 83% on test sets with our sparse aggregation model. In comparison, with the same configuration, DenseNet model achieved a minor 1% increase in test accuracy. SparseNet had only 0.1242M trainable parameters whereas DenseNet had 0.1761M parameters for the same model configuration. This suggests that we can significantly reduce the model complexity with almost similar accuracy with our sparse representation. Also, as can be seen from the curve of average time per epoch for CIFAR-10, it takes an average of 35.60msec time to train the SparseNet model whereas it takes around an average of 39.8msec for training DenseNet. While this is not a significant difference for shallow networks, it becomes considerable as the depth of the network and feature growth rate value (k) increase.

Next, we train the SparseNet and DenseNet models on N = 100 depth with increasing values of k = [24, 36, 64] and

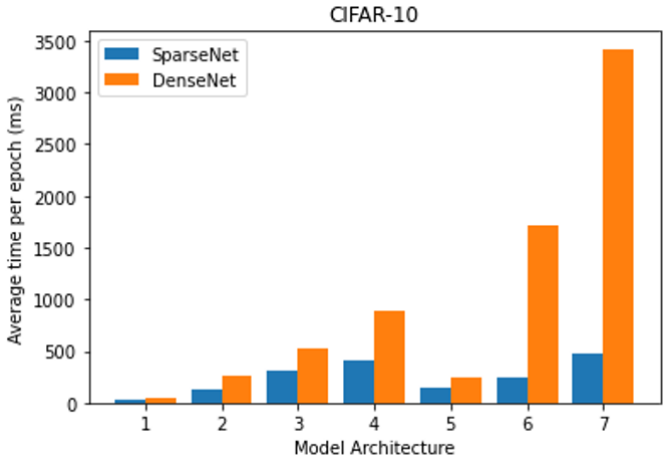

Fig. 6. CIFAR-10: Average time per epoch for DenseNet and SparseNet

also separately on k = [16, 32, 64] with BC = True. For k = 24 and 36, the test accuracy achieved by SparseNet is little lower (1% and 2% respective difference) as compared to that achieved by DenseNet but the total trainable parameters for SparseNet are almost half as compared to the DenseNet. Average training times per epoch for SparseNet in the above configurations are 134.88msec and 306.70msec whereas this value increases to 256.86msec and 529.28msec in case of DenseNet. For k = 64 and N = 100, the test accuracy achieved by SparseNet (87%) exceeds the test accuracy obtained by DenseNet (84%) with 10.13M trainable parameters for SparseNet and 21.23M for DenseNet. Hence, if we control the amount of sparsity with the value of feature growth rate k, we can achieve even higher test accuracy and shorter training times with SparseNet model as compared to the DenseNet model.

For the 5th configuration, we divide the training into 3 different stages and sequentially increase the k value as [16, 32, 64]. SparseNet achieves final test accuracy of 87% with 4.33M trainable parameters as compared to the 84% accuracy achieved by DenseNet with 28.07M trainable parameters. For the last model configuration we set the Bottleneck Compression (BC) flag to False and here again, the test accuracy achieved by SparseNet far exceeds that achieved by DenseNet with a lot fewer parameters. The average time per epoch for the last 2 configurations are only 241.66msec and 470.84msec for SparseNet whereas it is 1709.40msec and 3420.24msec for DenseNet. This shows that with sufficiently high value of feature growth rate k, SparseNet model achieves higher test accuracy with fewer parameters and training times as compared to the DenseNet architecture.

B. CIFAR-100: SparseNet vs DenseNet model configurations

Again, as was done in the case of CIFAR-10, we first check the performance of the SparseNet model on a shallow network i.e. N = 40. The value of feature growth rate k

Model Type	BC	Depth (N)	Feature Growth Rate (k)	Total Params	Accuracy
SparseNet	Yes	40	12	0.1276M	47.0
	Yes	100	24	1.4487M	57.0
	Yes	100	36	3.2362M	59.0
	Yes	100	64	10.1632M	63.0
	Yes	100	[16, 32, 64]	4.3626M	59.0
	No	100	24	2.4904M	60.0
	No	100	36	5.5745M	61
DenseNet	Yes	40	12	0.1881M	54.0
	Yes	100	24	3.0822M	65.0
	Yes	100	36	6.8465M	64.0
	Yes	100	64	21.3940M	63.0
	Yes	100	[16, 32, 64]	7.8639M	62.0
	No	100	24	28.2835M	62.0
	No	100	36	63.2821M	52.0

Fig. 7. Performance Metrics: Parameters and Accuracy for CIFAR-100

is set to 12 with Bottle Compression Flag = True. The model achieves test accuracy of 47% on test sets with our sparse aggregation model. In comparison, with the same configuration, DenseNet model achieved 54% test accuracy. SparseNet had only 0.1276M trainable parameters whereas DenseNet had 0.1881M parameters for the same model configuration. This suggests that for shallow networks with complex datasets like CIFAR-100, SparseNet model won't be as effective in terms of accuracy. Trend of the average training time per epoch is similar to that observed with CIFAR-10.

We train the SparseNet and DenseNet models on N = 100 depth with increasing values of k = [24, 36, 64] and also separately on k = [16, 32, 64] with BC = True. For k = 24 and 36, the test accuracy achieved by SparseNet is lower by 8% and 5% respectively as compared to that achieved by DenseNet but the total trainable parameters for SparseNet are almost half as compared to the DenseNet. Average training times per epoch for SparseNet in the above configurations are 134.88msec and 306.42msec whereas this value increases to 256.21msec and 529.87msec in case of DenseNet. For k = 64 and N = 100, the test accuracy achieved by SparseNet and DenseNet are equal, both being 63% with 10.163M trainable parameters for SparseNet and 21.394M for DenseNet. Hence, if we control the amount of sparsity with the value of feature growth rate k, we can achieve similar test accuracy and shorter training times with SparseNet model as compared to the DenseNet model.

Just like CIFAR-10, in the 5th configuration, we divide the

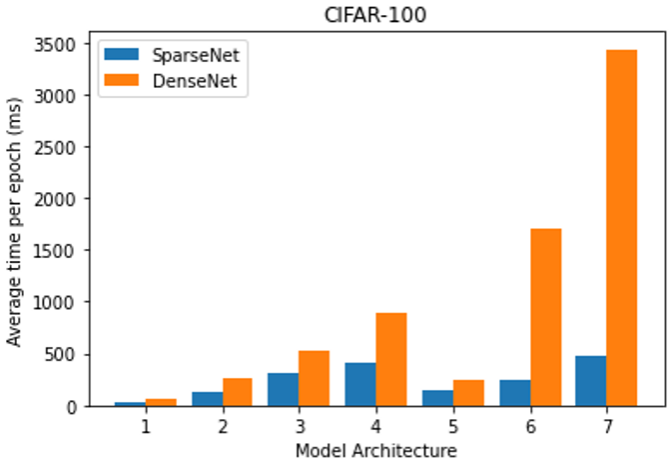

Fig. 8. CIFAR-100: Average time per epoch for DenseNet and SparseNet

training into 3 different stages and sequentially increase the k value as [16, 32, 64]. SparseNet achieves final test accuracy of 59% with 4.36M trainable parameters as compared to the 62% accuracy achieved by DenseNet with 7.86M trainable parameters. For the last model configuration we set the Bottleneck Compression (BC) flag to False and in this case, the test accuracy achieved by SparseNet (61%) far exceeds that achieved by DenseNet (52%) with a lot fewer parameters. The average time per epoch for the last 2 configurations are only 242.33msec and 472.36msec for SparseNet whereas it is 1705.37msec and 3433.46msec for DenseNet. This again reiterates that with sufficiently high value of feature growth rate k, SparseNet model achieves higher test accuracy with fewer parameters and training times.

VI. Discussion: Interpretation of Results and Conclusion

As can be seen from above experiments, for higher value of feature growth rates, the SparseNet model having sparse logarithmic aggregation topology performs better than the DenseNet architecture having dense linear aggregation topology. This may be attributed to the fact that as we increase the value of k, model learns more and more with just a little increase in number of parameters in the SparseNet. However, for DenseNet, as the percentage increase in parameters is less, there is not much of an increase in accuracy because of model saturation.

There is significant difference in the number of training parameters and the average training times per epoch with increase in the values of feature growth rates (k) which suggests that for deeper networks with higher k, not only we can achieve fewer training times and better computational complexity but also a better testing accuracy with the sparse logarithmic aggregation. Another point worth noting is that irrespective of the dataset complexity, we observed

similar trends in accuracy, training parameters and average time per epoch for both the image classification datasets CIFAR-10 and CIFAR-100. This reiterates that the concept of sparse aggregation is highly robust, scalable, invariant and independent of other factors like dataset complexity.

As the depth of the neural network increases, the dependent inputs of the previous layers in SparseNet, can be controlled by a small integer value because the logarithmic function grows slowly as compared to the linear function. This has several advantages over DenseNet architecture as it allows us to train much deeper networks and gives us the leeway of optimizing the hyperparameters more efficiently. For a particular model configuration, SparseNet achieves better or comparable accuracy with fewer trainable parameters whereas for a given number of trainable parameters / model size, DenseNet performs worse as compared to SparseNet. We only achieved a significant difference in test accuracy (with the DenseNet's accuracy being higher) with low feature growth rate values for CIFAR-100 dataset. For other cases, proposed implementation of SparseNet model performed much better than DenseNet.

To conclude, We propose a highly scalable and robust architectural design for our sparse aggregation implementation thereby a model, which can be applied to diverse models as well as on complex datasets. We propose that the linear to logarithmic implementation of the incoming links in a neural network, will infact even better the performance of the baseline ResNet/DenseNet model.We, thereby analyse the factors helping in analyzing the sparse implementation in different model architectures and argue that the sparse aggregation concept is independent of the architectural property difference across different architectures like FractalNet, ResNet, DenseNet etc. and hence can be independently applied to improve the computational performance.

VII. Future Scope

To take this work further, there is a scope in optimizing the sparse aggregation model for the optimum feature growth rate value for different datasets. The concept of sparse logarithmic aggregation can be combined with the selected linear aggregation via skip connections in ResNets so that the sparsity in the architecture can be employed to the selected part of the neural network. Also, there is a scope to explore different types of sparse aggregation strategies like addition and concatenation of incoming connections at a layer on different model architectures.

References

[1] Chang, B., Meng, L., Haber, E., Ruthotto, L., Begert, D., Holtham, E.: Reversible architectures for arbitrarily deep residual neural networks. AAAI (2018)

[2] Greff, K., Srivastava, R.K., Schmidhuber, J.: Highway and residual networks learn unrolled iterative estimation. ICLR (2017)

[3] Xiao, L., Bahri, Y., Sohl-Dickstein, J., Schoenholz, S.S., Pennington, J.: Dynam- ical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. ICML (2018)

[4] Zhiyang Zhang, Shihua Zhang: Towards Understanding Residual and Dilated Dense Neural Networks via Convolutional Sparse Coding

[5] Chen, W., Wilson, J.T., Tyree, S., Weinberger, K.Q., Chen, Y.: Compressing neural networks with the hashing trick. ICML (2015)

[6] Ligeng Zhu, Ruizhi Deng, Michael Maire, Zhiwei Deng, Greg Mori, Ping Tan: Sparsely Aggregated Convolutional Networks

[7] Chen, Y., Li, J., Xiao, H., Jin, X., Yan, S., Feng, J.: Dual path networks. NIPS (2017)

[8] Ilango Rajagopal: Intuition behind Residual Networks (22nd July2022) https://towardsdatascience.com/intuition-behind-residual-neural-networks-fa5d2996b2c7

[9] Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural net- works with pruning, trained quantization and huffman coding. ICLR (2016)

[10] Hu, H., Dey, D., Giorno, A.D., Hebert, M., Bagnell, J.A.: Log-DenseNet: How to sparsify a DenseNet. arXiv:1711.00002 (2017)