Smarter Prototyping for Neural Learning

Prabhu Pradhan*
Indian Institute of Science (IISc)
Bengaluru, India - 560012
prabhupradhan@pm.me

Meenakshi Sarkar
Indian Institute of Science (IISc)
Bengaluru, India - 560012
meenakshisar@iisc.ac.in

Debasish Ghose
Indian Institute of Science (IISc)
Bengaluru, India - 560012
dghose@iisc.ac.in

Abstract

Deep Neural Networks have become commercially viable in fields like machine vision, speech/language processing, data acquisition among other applications. Convolutional Neural Networks (CNNs), Recurrent Neural Network (RNNs) and their variants in some conditions have achieved performance better than human experts. However, existing deep network models are incompatible with low power devices or mission-critical applications due to either high computational & latency cost or memory storage, which makes them unfit to scale. Moreover, less effort has been put in making the architectural improvements modular or model-agnostic. In the developing regions of the world, efficient and frugal learning frameworks will have a huge socio-economic impact. AI can be a game-changer by enabling unique strategies to facilitate social good through domain-experts if ML-prototyping is intuitive. Thus, this paper serves a dual purpose, first is to present easily implementable structural modifications, and second is to provide a comparative overview of prevalent compression techniques. Finally, we conclude this paper discussing and proposing possible challenges in these areas.

1 Introduction

The deep learning networks which are being deployed commercially are GPU-hungry. State-of-the-art models in-lieu of delivering superb performance often contain billions of parameters. Such requirements are a critical hindrance in low-resource applications. While model quality has been shown to scale with model and data-set size (Hestness et al., 2017), the heavy resources required to train them can be prohibitive, especially in regions with low research budgets. More importantly, the current motivation for network design stems from beating the 'state-of-the-art'. These metrics are un-intuitive, and don't provide actionable feedback towards improvement (Anderson-Cook et al., 2019). As Goodhart's Law states- "When a measure becomes a target, it ceases to be a good measure", such metrics instead lead to models that are incomprehensible for further development (Lipton and Steinhardt, 2019). Various methodologies have been used to reduce the architectural complexity of such models (Simard, Steinkraus, and Platt, 2003). Residual Networks (ResNets) (He et al., 2016) and SqueezeNet (Iandola et al., 2017) achieve better classification results despite very small parameter count. Although, as shown in (Dubey, Chatterjee, and Ahuja, 2018), these compact-nets also have remnant redundancies.

*Undergraduate at FGIET (AKTU). This work was completed as part of AI Research Fellowship

The Neural Information Processing Systems (NeurIPS 2019) Workshop on Retrospectives, Vancouver, CA.

In this work we shed light on some methodologies for giving a jump-start to development pipelines aimed at leveraging the power of deep networks. This compilation should provide an intuitive platform for designing an existing model while also providing directions to conduct ablative studies.

2 Deep Learning for Good

Most non-profit organizations do not have a specialized workforce of data scientists, and engineers. With all the modern technology available to humankind, humble farmers across the world are still at the mercy of the environment for their livelihood. Remote Health Diagnostics in parts of the developing world still hasn't hit a critical point. While no silver bullet, machine learning can be an invaluable tool in a wide array of applications. It is quite evident that these problems already have baseline solutions using AI, although from a realistic perspective these are still out of reach from the populace. Neural Networks which *will not be a luxury* (Ferreira et al., 2019) to deploy can empower local officials and regional scientists. AI can be truly transformative if it receives contributions from all over the globe. Its significance as a tool depends on technology-penetration and adoption scale.

Domain-experts can use AI tools to harness innovations in Agriculture (Rußwurm et al., 2019), Climate Science (Kim et al., 2019), Healthcare (Koushik, Amores, and Maes, 2018) (Rawat, Li, and Yu, 2019) etc. However, these solutions will not come to fruition unless machine learning models are proficient both in terms of cost-effectiveness and ease of development. We address this two-pronged requirement by listing ideas to facilitate prototyping for real-world applications and also giving a gentle introduction to compact/compressed networks.

3 Prototyping-based improvements for Applied ML

In the ML community, it is well-known that data augmentation can be very beneficial for model performance (Goodfellow, Bengio, and Courville, 2016). Similarly, adding Gaussian Noise during training enhances validation and reduces overfitting (An, 1996). Also, cyclic learning rates have been shown to enhance training of the neural-nets to achieve faster convergence (Smith, 2015). Such tricks-of-the-trade are immensely useful during early stages of model design, especially for AI practitioners from eclectic backgrounds.

While there is a huge demand for AI-based solutions in other fields such as medical, disaster relief and management, such cross-disciplinary efforts will only succeed when deep learning models are frugal and easily customizable. The current trend of shipping black-box or non-interpretable neural networks is a serious bottleneck for custom-model development since they're counter-intuitive and unreliable. Thus, ease of prototyping is a crucial step forward.

Due to a meteoric rise in the general area of machine learning research, it is difficult to keep track of all the remarkable contributions while working on well-established theories. Hence, here we will focus on other such emerging techniques. In Table 1 (3.5), we summarize various techniques (Swish, OctConv, SwapOut, ZoneOut, Population-based Augmentation), and give an overview in Figure 1.

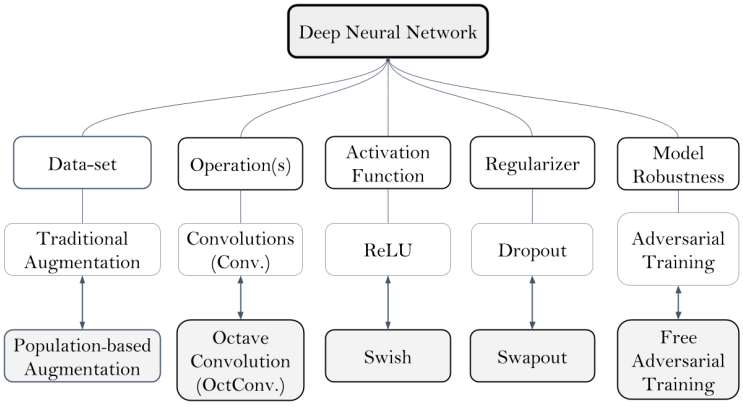

Figure 1: Modular Prototyping Techniques

3.1 Swish

Swish (Ramachandran, Zoph, and Le, 2018) is an activation function (as shown in Fig.2):

$$f(x) = x\,\sigma(\beta x)$$

where $\sigma(z) = \frac{1}{1+exp(-z)}$. Also, β can be defined as a constant (generally, 1) or a trainable parameter (especially helpful when encountering many dead ReLUs). Swish is non-monotonic but, like ReLU, bounded from below and unbounded from above. Its advantages are observable in deeper networks since it better handles vanishing gradients. It has a tendency to speed-up learning (despite using higher dropout), but may suffer from over-fitting in simpler tasks. It is highly recommended for dealing with complex data sets.

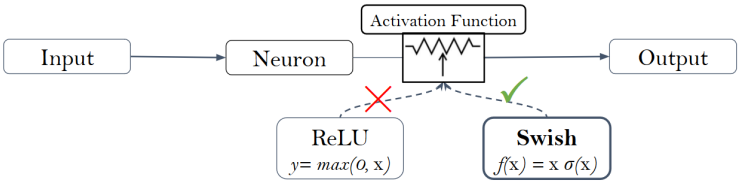

Figure 2: Swish instead of ReLU in a DNN

Can we replace ReLU with Swish everywhere? This is a tricky question to answer since ReLU has been extensively studied and is currently the most deployed activation function. Swish, at the cost of an increased training time, performs better on standard benchmarks than most of the other variants of ReLU and SELU (Klambauer et al., 2017). But there are no comparative results on adversarial training and generative models. A recent work on flow-based generative models (Chen et al., 2019b) coined a new swish function called Lipswish to address the vanishing gradient function in generative models. Although its promising, the present literature does not offer enough case studies to conclude that swish would outperform other linear activation functions in every other scenario.

Google's MobileNetV3 (Howard et al., 2019) achieves state-of-the-art results for mobile vision tasks and uses a Swish variant (hard-swish) which is faster and quantization-friendly.

3.2 Octave Convolution (OctConv)

There are a lot of spatial redundancies in CNN frameworks, thus OctConv (Chen et al., 2019c) enhances efficiency by leveraging low/high-frequency features independently. This characteristic also boosts classification performance due to better global context knowledge from a widened receptive field. It is a plug-and-play, orthogonal unit to substitute regular convolutions (2D and 3D) without any modifications to the network structure. To the best of our knowledge, OctConv has been used to stabilize GAN training (Durall, Pfreundt, and Keuper, 2019) and also reacts well with model compression (Zhou et al., 2019). It can be combined with techniques to decrease channel-wise redundancy (ex. depth-wise convolutions) and even for topological-improvements.

3.3 Regularizers

3.3.1 Swapout

Swapout (Singh, Hoiem, and Forsyth, 2016) is a stochastic training method that shows stable improvements using efficient parameter utilization. It can be seen as a clever merge of the two regularization techniques that is, dropout (Srivastava et al., 2014) and stochastic-depth (Huang et al., 2016), outperforming both in stand-alone comparisons. Also, linear decay of parameters (less dropping on early layers, more on later ones) significantly improves its results. Relatively shallow Swapout networks give similar performance to extremely deep ResNets.

JumpOut (Wang, Zhou, and Bilmes, 2019) is an orthogonal method which improves regularization and generalization by actively normalizing dropout rate based on active neurons. The compute overhead is relatively negligible, and it can be used with other versions of dropout.

Excitation Dropout (Zunino et al., 2018) takes cue from plasticity in brains where there may be alternate routes concerning a specific function. It disables the neurons most-contributing to network

prediction for enhancing model resilience. This inherently improves generalization, achieves evenly-distributed activations (higher neuronal utilization) and maintains post-compression robustness.

3.3.2 Zoneout

In the hidden units of RNNs, instead of *drop-out*, one can use *zoneout* (Krueger et al., 2016). In this method, unit activations are stochastically replaced with the preceeding time-step activations instead of removing them. This modification achieves higher performance than alternative regularizers, especially in speech processing applications. It seems to be inherently robust to changes in hidden state while streamlining information-flow through the network.

3.4 Population-based Augmentation (PBA)

Data augmentation has proved to be a valuable resource during training of deep learning models (Perez and Wang, 2017). Although, still there are no well-defined rules to select augmentation policies for a DNN. Population-based Augmentation or PBA (Ho et al., 2019) searches from a set of several operations and seeks out an effective combination. A carefully chosen policy enhances generalization and robustness but is a time-consuming process. AutoAugment (Cubuk et al., 2019) is the state-of-the-art method but is extremely compute-intensive, however, faster alternatives such as Fast AutoAugment (Lim et al., 2019) using density-matching based policy search are promising.

3.5 Adversarial Training for Free

Adversarial attacks can cause malfunctioning in other intelligent models through malicious inputs (Papernot et al., 2016), similar to optical illusions in human-vision. A natural defence is Adversarial training of a deep network which is very compute-intensive (Xie et al., 2018). This "free" technique (Shafahi et al., 2019) proposes an effective solution where instead of using separate gradient computations for each update during one simultaneous backward pass, simultaneous updates are applied on the model parameters and image-perturbations. This method also differs from previous approaches since it is orthogonal to other defenses, faster and applicable on big networks including high-resolution data sets ex. Imagenet (Krizhevsky, Sutskever, and Hinton, 2012).

Table 1: Summary of prototyping-based improvements for DNNs

Technique	Group	Brief	Pros	Cons
Swish	Activation Function	Non-monotonic, small bump for -ve input	Trains Deeper-Nets, faster convergence. Works with GANs	Over-fitting, slower epochs and inference (~15%)
OctConv	Convolution Operation	It bifurcates feature map tensors into low & high frequencies	Saves computation and memory. Boosts accuracy. Stabilizes GAN training	More hypertuning, Compatibility with other compression methods untested
SwapOut	Regularizer	Dropout + random skipping connections to generalize, Bonds parameters across layers	Prevents co-adaptation in neurons(units) & across network layers	Testing for best training schedules,
ZoneOut	Regularizer (RNNs)	Preserves hidden units (unlike dropout), regularize transition dynamics	Robust to Hidden-state changes, boosts gradient propagation	Not suitable for ResNets, More tests required for variants ex. GRU*
PBA	Augmentation	Hyperparameter search using evolutionary algorithms	Low compute-cost and memory. Effective optimization.	Not the most optimal result, Lacks extensive testing
Adversarial Training for Free	Model Robustness	Reuse the gradient info for concurrent updates to parameter & perturbation	Low compute and memory required. Multiple adversarial updates possible	Untested with other adversarial defense methods

(**GRUs= Gated Recurrent Units*)

4 Compact Networks

In a cloud-based environment with abundant computational capabilities, enabled by multiple graphical processing units (GPUs), such massive memory requirements may not be considered a restriction (Chollet, 2016). However, in case of mobile devices (ex. Robots, Internet of Things) with limited computational capabilities, such resource intensive deep neural networks cannot be readily applied (Sarkar, Pradhan, and Ghose, 2019). The exploitation of deep learning on sensory-devices, including smartphones (Lane, Georgiev, and Qendro, 2015) (Haffari et al., 2018), has pointed out this as a major hurdle to wide spread use.

Nowadays, sparsity is also used to refer to the proportion of a neural networks weights that are zero valued. Moreover, it has been shown empirically that DNNs can tolerate high levels of sparsity (Narang et al., 2017), and this property has been leveraged to significantly reduce the cost associated with implementing deep-nets, and to enable the deployment of state-of-the-art models in severely resource constrained environments (van den Oord et al., 2016). Higher sparsity corresponds to fewer weights, and smaller computational and storage requirements (Gale, Elsen, and Hooker, 2019).

Thus, the design of deep neural networks that require less storage and computation power has established itself as a new research direction. Particularly, the modification of large cumbersome models that reduces the memory requirements while retaining as much of its performance as possible is referred to as compression of neural networks (Han et al., 2015). Another direction is the design of more memory efficient network architectures from scratch (Hinton, Vinyals, and Dean, 2015).

In the following segment, we trace the foundational ideas of the prevalent methods for model-compression (Cheng et al., 2017), briefly comment on their key characteristics and tabulate their Pros & Cons. In Table 2 (3.5), we summarize four types of compression methods (Pruning, Quantization, Knowledge Distillation and Tensor Vectorization/Low-Rank Factorization).

4.1 Pruning

Despite the utilization of powerful regularization techniques like dropout or weight decay, some weights of a network will always contribute more to the prediction than others (Han et al., 2015). The process of removing the less contributing weights to compress (and/or further regularize) the network is called pruning (Han, Mao, and Dally, 2016). After some weights have been pruned, the network typically has to be fine tuned again to have it adapt to the change (Wen et al., 2016).

4.2 Quantization

Quantization is decreasing the (dispensable) numerical precision of a model. Its a popular approach for data compression and several methods have emerged for neural network acceleration as well (Krishnamoorthi, 2018). The most common methods are: scalar/vector quantization, and fixed-point quantization. However, experimentally (Google, 2019) it has been revealed that methods like k-means quantization do not improve speed or storage requirements in practice. Since the weight matrix has to be reconstructed to 32-bit floats from 8 or 16 bit floats during inference, the process is often referred as pseudo quantization in the community. It is different from real quantization where each weight is permanently encoded using fewer bits.

4.3 Knowledge Distillation

The thought in knowledge distillation is to "mimic" knowledge of a (large) teacher model into a smaller and efficient (student) model by learning the softmax-based class distributions. The cross entropy difference in the predictions from both the models is calculated while doing a forward pass during training, and is added to the student-model's loss value. In this method, the student also learns "dark knowledge" (Hinton, Vinyals, and Dean, 2014) of the closely associated categories from the teacher-model in addition to learning from ground-truth labels.

Although knowledge-distillation based techniques are very promising, and achieving impressive results in reducing compute-cost and complexity, they are currently severely restrictive in a sense that only models having softmax-based classification tasks can be distilled.

4.4 Vectorization/Tensor Decomposition (Low-Rank Factorization)

The core thought behind Matrix factorization (MF) is that by exploiting latent structures inherent to the data set we can obtain compressed feature representations (Denton et al., 2014). Tensor Decompositions are higher order matrix decomposition, leveraging redundancy in 4D tensors (Conv Kernels). Thus, any reductions in the convolutional layers would improve inference speed since Conv. operations are heavy. This method is controversial for requiring robust re-training since the model's learning is altered, but many consider the gains worthwhile. In its various forms, it is a popular tool for dimensionality-reduction, unsupervised clustering, and can be used in combination to other compression methods.

Table 2: Summary of common model-compression techniques
(**Conv= Convolutional Layer, FC=Fully-Connected*)

Technique	**Pruning**	**Quantization**	**Distillation**	**Vectorization**
Description	Sharing or removing redundant parameters	Reducing the model's numerical precision	Learning smaller models from big ones (mimic)	Approximate a weight matrix by sum minimization
Application*	Conv/FC-layer. (on synapses and/or neurons)	Post-Training (or quantization-aware training)	Conv & FC-layer (across network)	Conv/FC-layer
Pros	Reduces size, complexity and over-fitting	Faster Inference (Even better when Activations are quantized too)	Significant reduction in computational-cost and size	High compression and speed-up
Cons	Longer Training, more hyper-tuning	Hardware dependant benefits	Overfitting, works only for (Softmax) classifications	Rigorous Retraining, Small ranks may hurt model

5 Conclusion

The first-half of the paper is oriented towards discussing various techniques in the recent literature that would expedite early-stage prototyping and favour AI adoption in diverse domains and applications. It also gives pragmatic insights about improving network performance (training time, accuracy etc.) and thus could be of interest to the ML community as well. These techniques would bring maximum benefit when built with efficiency at their core. So, in the second-half we gave an overview of the main concepts and approaches to model compression. This paper presents the precursory tool-kit necessary for building and training an efficient network from scratch. In the following section we discuss on the prevalent issues related to implementing these various tools and how that may impact the future direction of research.

6 Discussion and Future Directions

We discussed the feasibility of compressing models for disk size and memory usage at negligible accuracy losses. Compressing for speed, on the other hand, is tricky in practice (Chen et al., 2019a). It can depend on faster integer multiplication or sparse matrix multiplication, here machine learning research hits the reality of computing (hardware) architecture. As evident in recent literature, there is a booming interest in Compact Networks for compatibility with small-scale embedded systems.

While we tried to provide a brief and yet comprehensive overview on the recent advancements made in network architecture design to the readers, we also discovered huge gaps in the existing literature on detailed comparative studies on various prototyping techniques in this rapidly evolving field. For example, we have presented a detailed discussion on the merits of using swish activation function compared to other linear activation functions for both generative and classification models. But while we explored the literature for similar analysis on other techniques such as swapout or zoneout, we could not find enough empirical analysis to comment about the generalizability of these approach across domain and model specifications. Exploring detailed analysis on these techniques would be beneficial to the deep learning community and might lead to exciting new findings. We believe such

studies will also reveal the limitation of these various prototyping techniques and help us establish more elaborate standards when it comes to building deep networks from the scratch.

Another interesting area we did not explore in this paper is Interpretable and Explainable Network Architectures (Chen et al., 2016), (Zhou et al., 2017). This field has a lot of potential and will be essential for building trust in AI Systems as well as setting-up benchmarks for future research. There is also a dire need of developing mathematical framework that would help us analyse and understand behaviour of neural networks under the influence of various activation or regularization functions. The existing literature relies heavily on empirical case studies for performance analysis and are often designed specifically for a particular problem. Often these kind of practices do not transfer well across different domains and lead to a lot of trial-and-error runs before one can narrow down the specifications that works well for their problem.

Acknowledgements

Authors acknowledge EPSRC-GCRF for partial funding (EP/P02839X/1), DeepMind for TFRC TPU credit award (MicroNet Challenge), and NeurIPS 2019 Board for PP's complimentary registration. We would like to thank Narendra Ahuja (UIUC) for valuable insights, and Gopikishan (IIT-R) for sharing results from Google Cloud experiments. PP extends thanks to Ruchit (NSIT), the members of GCDSL (AERO@IISc) and PAVIS/VGM (IIT Genoa) for their feedback on the final draft. PP also expresses special gratitude to R.P. Sharma (FGIET) for guidance and support.

References

An, G. 1996. The effects of adding noise during backpropagation training on a generalization performance. *Neural Computation* 8:643–674.

Anderson-Cook, C. M.; Myers, K. L.; Lu, L.; Fugate, M. L.; Quinlan, K. R.; and Pawley, N. 2019. How to host an effective data competition: Statistical advice for competition design and analysis. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 12(4):271–289.

Chen, X.; Duan, Y.; Houthooft, R.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chen, C.; Chen, Q.; Do, M. N.; and Koltun, V. 2019a. Seeing motion in the dark. In *The IEEE International Conference on Computer Vision (ICCV)*.

Chen, R. T. Q.; Behrmann, J.; Duvenaud, D.; and Jacobsen, J. 2019b. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*.

Chen, Y.; Fan, H.; Xu, B.; Yan, Z.; Kalantidis, Y.; Rohrbach, M.; Yan, S.; and Feng, J. 2019c. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In *Proc.. of International Conference on Computer Vision*, volume abs/1904.05049.

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A survey of model compression and acceleration for deep neural networks. *IEEE Signal Processing Magazine* ArXiv abs/1710.09282.

Chollet, F. 2016. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 1800–1807.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. Autoaugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Dubey, A.; Chatterjee, M.; and Ahuja, N. 2018. Coreset-based neural network compression. In *The European Conference on Computer Vision (ECCV)*, volume abs/1807.09810.

Durall, R.; Pfreundt, F.-J.; and Keuper, J. 2019. Stabilizing gans with octave convolutions. *ArXiv* abs/1905.12534.

Ferreira, J.; de Almeida Callou, G. R.; Josua, A.; Tutsch, D.; and Maciel, P. 2019. An artificial neural network approach to forecast the environmental impact of data centers. *Information* 10:113.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The state of sparsity in deep neural networks. *ArXiv* abs/1902.09574.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Google. 2019. Tensorflow (lite) guide. `https://www.tensorflow.org/lite/performance/`.

Haffari, R.; Cherry, C.; Foster, G.; Khadivi, S.; and Salehi, B. 2018. Workshop on deep learning approaches for low-resource nlp. Melbourne, Australia: Association for Computational Linguistics (ACL).

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Han, S.; Mao, H.; and Dally, W. J. 2016. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *International Conference on Learning Representations, (ICLR)*, volume abs/1510.00149.

He, K.; Zhang, X.; Ren, S.; and Sun., J. 2016. Deep residual learning for image recognition. In *Proc.. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Hestness, J.; Narang, S.; Ardalani, N.; Diamos, G. F.; Jun, H.; Kianinejad, H.; Patwary, M. M. A.; Yang, Y.; and Zhou, Y. 2017. Deep learning scaling is predictable, empirically. *CoRR* abs/1712.00409.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2014. Dark knowledge in deep learning. `https://www.ttic.edu/dl/dark14.pdf`.

Hinton, G. E.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume abs/1503.02531.

Ho, D.; Liang, E.; Chen, X.; Stoica, I.; and Abbeel, P. 2019. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, volume abs/1905.05393.

Howard, A.; Sandler, M.; Chu, G.; Chen, L.-C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; Le, Q. V.; and Adam, H. 2019. Searching for mobilenetv3. In *The IEEE International Conference on Computer Vision (ICCV)*.

Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; and Weinberger, K. Q. 2016. Deep networks with stochastic depth. In *Computer Vision – ECCV 2016*, 646–661. Cham: Springer International Publishing.

Iandola, F. N.; Moskewicz, M. W.; Ashraf, K.; Han, S.; Dally, W. J.; and Keutzer, K. 2017. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *ArXiv* abs/1602.07360.

Kim, S. K.; Park, S. H.; Chung, S.; joonseok Lee; Lee, Y.; Prabhat, M.; Kim, H.; and Choo, J. 2019. Focus and track : pixel-wise spatio-temporal hurricane tracking. In *Climate Change Workshop at ICML 2019*.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-normalizing neural networks. In *NIPS*.

Koushik, A.; Amores, J.; and Maes, P. 2018. Real-time sleep staging using deep learning on a smartphone for a wearable eeg. In *Machine Learning for Health (ML4H) Workshop at NeurIPS 2018*.

Krishnamoorthi, R. 2018. Quantizing deep convolutional networks for efficient inference: A whitepaper. *ArXiv* abs/1806.08342.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Proc.. of the Twenty-sixth International Conference on Neural Information Processing Systems*, NIPS'12, 1097–1105.

Krueger, D.; Maharaj, T.; Kramár, J.; Pezeshki, M.; Ballas, N.; Ke, N. R.; Goyal, A.; Bengio, Y.; Courville, A. C.; and Pal, C. 2016. Zoneout: Regularizing rnns by randomly preserving hidden activations. In *International Conference on Learning Representations, (ICLR)*, volume abs/1606.01305.

Lane, N. D.; Georgiev, P.; and Qendro, L. 2015. Deepear: robust smartphone audio sensing in unconstrained acoustic environments using deep learning. In *UbiComp*.

Lim, S.; Kim, I.; Kim, T.; Kim, C.; and Kim, S. 2019. Fast autoaugment. In *Advances in Neural Information Processing Systems 32, (NeurIPS)*. Curran Associates, Inc. 6662–6672.

Lipton, Z. C., and Steinhardt, J. 2019. Troubling trends in machine learning scholarship. *Queue* 17(1):80:45–80:77.

Narang, S.; Diamos, G.; Sengupta, S.; and Elsen, E. 2017. Exploring sparsity in recurrent neural networks. In *International Conference on Learning Representations, (ICLR)*, volume abs/1704.05119.

Papernot, N.; McDaniel, P. D.; Goodfellow, I. J.; Jha, S.; Celik, Z. B.; and Swami, A. 2016. Practical black-box attacks against machine learning. In *AsiaCCS*.

Perez, L., and Wang, J. 2017. The effectiveness of data augmentation in image classification using deep learning. *ArXiv* abs/1712.04621.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2018. Searching for activation functions. In *(Workshop Track) International Conference on Learning Representations, (ICLR)*, volume abs/1710.05941.

Rawat, B. P. S.; Li, F.; and Yu, H. 2019. Naranjo question answering using end-to-end multi-task learning model. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA*.

Rußwurm, M.; Tavenard, R.; Lefèvre, S.; and Körner, M. 2019. Early classification for agricultural monitoring from satellite time series. In *AI for Social Good (AISG) Workshop at ICML 2019*.

Sarkar, M.; Pradhan, P.; and Ghose, D. 2019. Planning robot motion using deep visual prediction. In *7th Workshop on Planning & Robotics (PlanRob), ICAPS 2019*. Berkeley, USA: CoRR, abs/1906.10182.

Shafahi, A.; Najibi, M.; Ghiasi, A.; Xu, Z.; Dickerson, J. P.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! In *Advances in Neural Information Processing Systems (NeurIPS)*.

Simard, P. Y.; Steinkraus, D.; and Platt, J. C. 2003. Best practices for convolutional neural networks applied to visual document analysis. *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.* 958–963.

Singh, S.; Hoiem, D.; and Forsyth, D. 2016. Swapout: Learning an ensemble of deep architectures. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*, 28–36. Curran Associates, Inc.

Smith, L. N. 2015. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)* 464–472.

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

van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A. W.; and Kavukcuoglu, K. 2016. Wavenet: A generative model for raw audio. In *SSW*.

Wang, S.; Zhou, T.; and Bilmes, J. A. 2019. Jumpout : Improved dropout for deep neural networks with relus. In *International Conference on Machine Learning (ICML)*.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Xie, C.; Wu, Y.; van der Maaten, L.; Yuille, A. L.; and He, K. 2018. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume abs/1812.03411.

Zhou, B.; Bau, D.; Oliva, A.; and Torralba, A. 2017. Interpreting deep visual representations via network dissection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41:2131–2145.

Zhou, D.; Jin, X.; Wang, K.; Yang, J.; and Feng, J. 2019. Deep model compression via filter auto-sampling. *ArXiv* abs/1907.05642.

Zunino, A.; Bargal, S. A.; Morerio, P.; Zhang, J.; Sclaroff, S.; and Murino, V. 2018. Excitation dropout: Encouraging plasticity in deep neural networks. *ArXiv* abs/1805.09092v2.