Efficient Synthesis of Compact Deep Neural Networks

Wenhan Xia
Princeton University
wxia@princeton.edu

Hongxu Yin
Princeton University
hongxuy@princeton.edu

Niraj K. Jha
Princeton University
jha@princeton.edu

***Abstract*—Deep neural networks (DNNs) have been deployed in myriad machine learning applications. However, advances in their accuracy are often achieved with increasingly complex and deep network architectures. These large, deep models are often unsuitable for real-world applications, due to their massive computational cost, high memory bandwidth, and long latency. For example, autonomous driving requires fast inference based on Internet-of-Things (IoT) edge devices operating under run-time energy and memory storage constraints. In such cases, compact DNNs can facilitate deployment due to their reduced energy consumption, memory requirement, and inference latency. Long short-term memories (LSTMs) are a type of recurrent neural network that have also found widespread use in the context of sequential data modeling. They also face a model size vs. accuracy trade-off. In this paper, we review major approaches for automatically synthesizing compact, yet accurate, DNN/LSTM models suitable for real-world applications. We also outline some challenges and future areas of exploration.**

***Index Terms*—Convolutional neural network, deep learning, grow-and-prune synthesis paradigm, long short-term memory, machine learning, model compression.**

I. Introduction

Deep neural networks (DNNs) have revolutionized various artificial intelligence applications, such as computer vision, speech recognition, machine translation, and smart healthcare [1]–[4]. Unlike traditional machine learning methods that process hand-crafted features extracted from raw data, DNNs automatically learn to represent data features via multi-level abstraction. Today, DNNs even outperform humans in some tasks, such as image classification [5].

To achieve high accuracy, researchers tend to design wider and deeper DNN models. This requires massive computational resources and immense amounts of data for training [5]–[7]. The slowdown in Moore's Law, however, makes it difficult to keep pace with the high memory bandwidth and computational power requirements of these increasingly large models. Consequently, there is a widening gap between computational demands from DNNs and resources/capabilities offered by the underlying hardware, thus limiting DNN deployment in many real-world applications. For example, the substantial storage, memory bandwidth, and energy requirements may be too high for edge devices, such as mobile phones, smart watches, and Internet-of-Things (IoT) sensors [8], [9]. Bulky DNNs also tend to have a high inference latency, which is unacceptable in various delay-sensitive scenarios, such as autonomous driving [10]. In addition to computational power demands, the immense labeled datasets needed to train these models can be prohibitively costly and time-consuming to obtain. Dataset acquisition also raises privacy concerns, especially in domains like healthcare, where patient data must be protected [11].

LSTMs are a type of neural network that incorporate feedback. This enables them to perform sequential data modeling. They find use in applications like speech recognition [12], neural machine translation [3], health monitoring [13], and language modeling [14], [15]. LSTM accuracy is typically increased by increasing its depth, hence its size. Obtaining a compact, yet accurate, LSTM is also a challenging task.

In this paper, we provide an overview of effective techniques to address the above problems and facilitate DNN/LSTM deployment across the IoT hierarchy – from the cloud to edge to sensor.

The rest of this paper is organized as follows. In Section II, we review DNN development history and deployment constraints, and summarize existing approaches for designing compact models. Section III focuses on major techniques that yield accurate, yet compact, models for fast inference. Section IV describes compact DNN designs that take into account hardware specifications and traits. Section V discusses data constraints and availability, and techniques for overcoming these limitations. Finally, Section VI provides a summary, and outlines some research challenges for future exploration.

II. Overview

Deep learning is a branch of machine learning that uses DNNs, which learn high-level data representations through many processing layers. Due to its outstanding and robust performance, deep learning has been adopted by many academic disciplines, from engineering to biology to psychology.

A. A Brief History of DNNs

Although the popularity of deep learning only bloomed in the past decade, its foundations were laid in the 1940s, when McCulloch and Pitts showed that networks of artificial neurons, later referred to as neural networks, could execute first-order logical functions [16]. In 1958, Rosenblatt developed the perceptron that performed threshold-based classification on numerical inputs [17]. Following Rosenblatt's perceptron, Ivakhnenko and Lapa developed a multilayer perceptron: one of the first multi-layer neural network architectures [18]. In the l990s, LeCun et al. proposed LeNet-5 [19], a neural

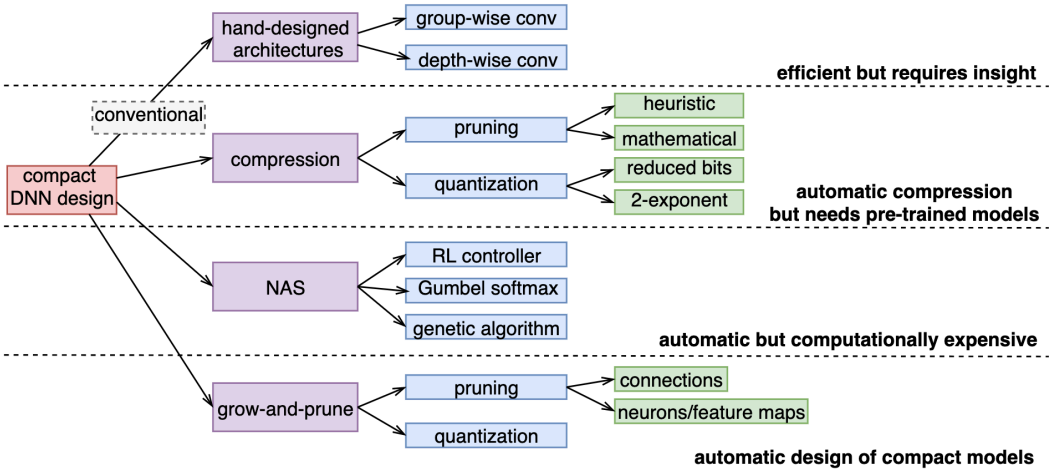

Fig. 1. Overview of approaches for compact DNN design; RL: reinforcement learning; NAS: neural architecture search.

network trained with the backpropagation algorithm. LeNet-5 is commonly regarded as the first practical application of neural networks and was widely used to recognize handwritten digits on checks.

The deep learning revolution occurred in the early 2010s. It is often attributed to three major factors: availability of large-scale datasets, significantly increased computational speed provided by GPUs, and novel DNN architectures, especially those involving convolutional layers. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [20] embodies the rapid evolution of deep learning during this decade. In 2012, AlexNet [6], a deep convolutional neural network (CNN), soundly defeated traditional machine learning methods. AlexNet inspired active research and development of deep CNNs, including VGG [7] and GoogLeNet [21]. In 2015, another deep learning milestone was established when the deep residual network, ResNet [5], achieved superhuman top-5 classification accuracy. In addition to breaking the human performance barrier, ResNet also solved the problem of stacking more layers with a residual framework. The rapid development of deep learning is mirrored by the substantial decrease in the top-5 classification error every year.

B. Complications

The yearly improving accuracy on the ImageNet dataset is associated with increasingly deeper DNN models. For example, AlexNet used eight layers, VGG 16 layers, GoogLeNet 22 layers, and ResNet 152 layers. A similar trend is observable in the field of speech recognition. For example, DeepSpeech2 [12], which has been widely used for speech recognition, is more than two times deeper and ten times larger than the initial DeepSpeech.

Most modern DNNs are deep and bulky, which may be an appropriate design choice when given enough training time, data, and computational resources. However, this ideal scenario does not necessarily hold for many real-world applications. We categorize constraints on developing DNNs into the following four groups:

- **Computational energy**: Inference platforms, such as wearable devices and autonomous drones, are highly energy-constrained, whereas DNN models are expected to continually make predictions. Thus, designing DNNs that can satisfy stringent energy budgets is now a major design objective.
- **Latency**: Applications like autonomous driving, real-time video analysis, and speech processing assistants like Siri and Cortana require fast inference. This limits the number of parameters and layers that a DNN can have.
- **Memory**: Deploying large, highly parameterized DNNs requires massive memory usage. This is expensive and infeasible for many applications.
- **Data availability**: Obtaining large-scale labeled training datasets can be costly and time-consuming. Most available datasets are of small or medium size. In addition, certain datasets, e.g., biomedical, may not be publicly available due to privacy concerns. Hence, designing DNNs that can achieve high performance in a limited data regime is a major new research thrust.

C. Overview of Compact DNN Design

Due to the limitations described above, considerable effort is being invested in efficient DNN design. Next, we summarize the major approaches for developing compact DNN models, as illustrated in Fig. 1.

Conventional approaches for building efficient DNNs involve the creation of efficient building blocks for removing redundancy. For example, MobileNetV2 [22] stacks inverted residual building blocks to shrink model size. Ma et al. achieve model compactness via channel shuffle operations and depth-wise convolution [23]. Wu et al. propose ShiftNet, a framework for using a shift-based module rather than spatial convolutional layers, to substantially reduce computational and storage cost [24]. Although these hand-crafted models can offer impressive compactness, they require substantial design insight and trial-and-error modifications to maximize performance.

Network compression is an alternative, automatic approach for compact DNN design that eliminates the need for *a priori* design principles and trial-and-error modification. It is easy to implement on pre-trained models and generally requires less development time. Pruning is a widely-used compression technique that removes individual weights or entire filters (neurons) such that model performance, e.g., accuracy, does not drop significantly. An ideal pruning solution would be to compute the ℓ_0 norm of weights. However, this is non-convex, NP-hard, and requires combinatorial search. These issues can be avoided with heuristics-based pruning criteria that are strongly correlated with the ground truth importance estimates of weights or neurons for final inference [25]–[28]. For example, Han et al. have shown the effectiveness of weight pruning based on magnitude and achieved substantial non-regular weight sparsity in modern CNNs [29]. Subsequent work has extended the methodology to ℓ_1 norm [30], Taylor expansion [25], [28], and batch norm [31], [32] based filter pruning for achieving structural sparsity. The model can be further compressed through low-bit quantization. For example, Zhu et al. show that CNNs with a low-bit representation of weights require significantly less memory for image recognition and object detection tasks, with only a slight loss in accuracy [33]. The Hardware-Aware Automated Quantization framework leverages reinforcement learning (RL) to automatically determine the quantization policy by incorporating hardware accelerator feedback in the design loop [34].

Recently, ideas from RL have been adapted to search for DNN architectures in an automated flow. This approach is known as neural architecture search (NAS) [35], [36]. NAS typically uses a controller, e.g., a recurrent neural network (RNN), to iteratively generate groups of candidate neural networks in the search process. Candidate performance is later used as a reward for enhancing the controller [37]. A recent work implements these ideas with an architecture called NASNet that achieves better performance than hand-crafted DNNs by using RL to search for architectural building blocks [36]. Furthermore, RL-based NAS can also be used to develop efficient DNNs for mobile devices and platforms. For example, MnasNet [38] running on a Pixel phone achieves 75.2% top-1 accuracy on the ImageNet classification dataset with only 78ms latency.

Despite their success in compact architecture search, RL-based methods remain computationally intensive. To enable an effective and direct gradient descent in the architecture space, Wu et al. propose FBNet that utilizes Gumbel softmax to jointly optimize connections and weights using the same objective function [39]. Genetic algorithms offer an alternative approach for this optimization. For example, a methodology called NEAT simultaneously learns neural network topologies and weights, resulting in optimized and increasingly complex models over generations [40]. However, it is beneficial to employ extremely efficient search algorithms based on various predictors in this context, as explained later in the discussion of the ChamNet approach [41].

NAS and related methods are appropriate when a large number of GPUs are available for training. However, this is not the case in many scenarios. Grow-and-prune offers an alternative approach for deriving efficient models without the need for many GPUs. In addition, unlike network compression, grow-and-prune does not require a pre-trained model. It simultaneously learns both network weights and architecture. A grow-and-prune synthesis paradigm typically reduces the number of parameters in multi-layer perceptron, CNN, and LSTM based models by another 2× relative to when only pruning is used, while increasing accuracy [42]–[44].

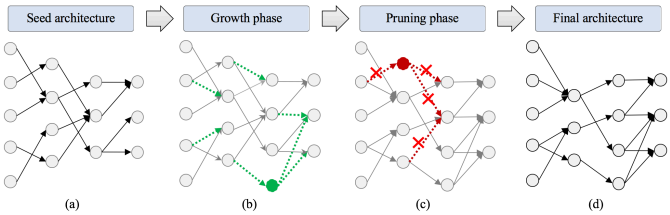

Fig. 2. An illustration of the grow-and-prune synthesis paradigm used in NeST [42].

III. Model Efficiency

In this section, we discuss synthesis tools that achieve both model efficiency and high accuracy.

NeST is a synthesis tool that automatically learns both DNN weights and compact architectures during training [42]. The synthesis process contains two phases: (1) a gradient-based growth phase where the gradient information is used to gradually grow new connections, neurons, and feature maps to boost model accuracy, and (2) a magnitude-based pruning phase where redundant connections, i.e., neurons with low magnitude, are removed. The grow-and-prune paradigm in NeST mimics the pattern of synaptic development in the human brain. Specifically, the number of biological synaptic connections increases over the period of a few months after birth and decreases steadily thereafter [45].

The NeST flow is illustrated in Fig. 2. Synthesis starts with a seed architecture that is initialized as a sparsely-connected DNN but with all neurons connected. Next, in the growth phase, NeST gradually grows new connections, neurons, and feature maps based on gradient information with the goal of increasing accuracy. Then, in the pruning phase, NeST prunes away redundant and insignificant neurons and connections based on their magnitudes. After several iterations of growth and pruning, NeST results in a lightweight DNN model with drastically reduced size and no accuracy degradation. In addition to its superior ability to reduce computational cost, NeST offers more freedom to DNN designers, since it can start with a wide range of seed architectures and scale to large datasets. For example, for LeNet-300-100 (LeNet-5) on the MNIST dataset, NeST reduces the number of network parameters and floating-point operations (FLOPs) by 70.2× (74.3×) and 79.4× (43.7×), respectively. For AlexNet and VGG-16 on the ImageNet dataset, NeST reduces the number

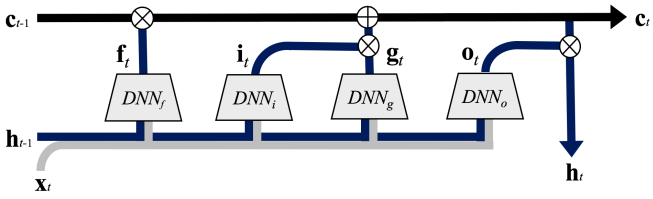

Fig. 3. Schematic diagram of the H-LSTM structure [43].

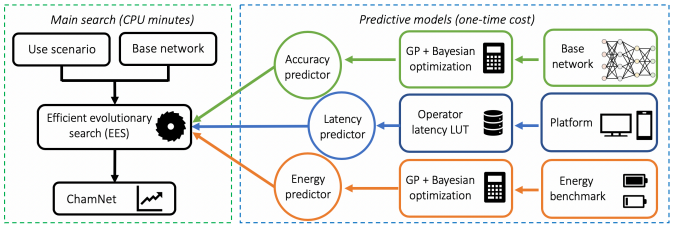

Fig. 4. The ChamNet model adaptation framework based on efficient evolutionary search [41].

of parameters (FLOPs) by 15.7× (4.6×) and 33.2× (8.9×), respectively.

Beyond typical feed-forward DNNs, a special type of RNN widely used for sequential data modeling, called LSTM, also faces a model size/accuracy trade-off. To address this problem, a hidden-layer LSTM (H-LSTM) has been proposed [43]. It uses the grow-and-prune paradigm proposed earlier in NeST to synthesize accurate yet compact LSTMs. The structure of an H-LSTM is illustrated in Fig. 3. As its name suggests, an H-LSTM improves learning by introducing hidden layers in conventional single-layer control gates. As a result, each LSTM cell has greater learning power. Hence, fewer stacking cells are needed to achieve the same or higher accuracy compared to a conventional LSTM. The grow-and-prune method is used to automatically learn the weights and architectures of the control gates. Extensive experiments have been performed to demonstrate the effectiveness of H-LSTMs. Compared to the baseline NeuralTalk architecture, H-LSTMs have 38.7× fewer parameters (45.5× fewer FLOPs), 4.5× lower run-time latency, and 2.8% higher CIDEr-D score on the MSCOCO dataset. Compared to the DeepSpeech2 architecture on the AN4 dataset, H-LSTMs have 19.4× fewer parameters (23.5× fewer FLOPs) and 37.4% lower run-time latency. Additionally, H-LSTMs reduce the word error rate from 12.9% to 8.7%. Compared to the Seq2Seq architecture, H-LSTMs have 10.8× fewer parameters, 14.2% lower run-time latency, and 3.2% higher BLEU score on the IWSLT 2014 German-English dataset.

Most automatic architecture synthesis strategies assume a fixed DNN depth during training [25], [29], [42], [43]. By eliminating this constraint, the size of the output model can be further reduced. SCANN [44] is a synthesis methodology that generates a general compact feed-forward architecture. In this method, rather than only receiving inputs from the immediate previous layer, hidden neurons can receive inputs from any preceding neuron. In conjunction with dataset dimensionality reduction, layer-wise neural network compression, and global neural network compression, SCANN demonstrates significant model compression power for small- and medium-size datasets: up to 5079× (geometric mean of 82×) relative to traditional neural networks, with little to no drop in accuracy. This can enable energy-efficient inference even on IoT sensors.

IV. Hardware-aware DNN design

In this section, we describe an integrated approach that considers hardware specifications and traits, in addition to DNN compactness, to achieve high efficiency and performance.

A large amount of effort in deriving compact and efficient DNNs is directed toward reducing network complexity. However, these approaches are mostly hardware-agnostic, and thus complexity reduction is not always tantamount to execution efficiency. To further improve efficiency by considering hardware characteristics, a hardware-guided symbiotic training methodology for execution-efficient and accurate inference models has been proposed [47]. By leveraging the hardware-impacted hysteresis effect (i.e., the non-monotonic latency surface when the model dimension shrinks) and using a multi-granular grow-and-prune synthesis approach, this method achieves model compactness and accuracy in addition to execution efficiency. For language modeling and speech recognition tasks, the method achieves 7-31× parameter reduction and 1.7-5.2× direct latency reduction relative to off-the-shelf NVIDIA GPUs and Intel CPUs, while improving accuracy.

Hardware-guided model adaptation is another new trend for compact DNN design that honors varying resource budgets upon model deployment. To enable fast yet automatic model adaptation to the underlying hardware, the Chameleon framework that leverages existing network building blocks and focuses on exploiting hardware traits to meet target latency and energy constraints has been proposed [41]. At the core of the approach lie three novel predictors for run-time accuracy, latency, and energy. Based on a one-time profiling effort, these three predictors significantly reduce the intermediate model evaluation time during efficient evolutionary search. In just minutes, the method produces a family of state-of-the-art compact DNNs, called ChamNets, for different targeted platforms with given constraints. For example, it achieves 73.8% (75.3%) top-1 accuracy on ImageNet with 20ms latency on a mobile CPU (DSP). Compared with MobileNetV2 and MnasNet, ChamNet models achieve up to 8.2% (4.8%) and 6.7% (9.3%) absolute top-1 accuracy improvements, respectively, on a mobile CPU (DSP). Compared to ResNet-101 and ResNet-152, ChamNet models achieve 2.7% (4.6%) and 5.6% (2.6%) accuracy gains, respectively, on an Nvidia GPU (Intel CPU).

V. Data Efficiency

Availability of large-scale training data, or a lack thereof, raises another major challenge for DNN deployment. In practice, most datasets are small- or medium-size due to the time and expense of collecting data. In addition, some datasets

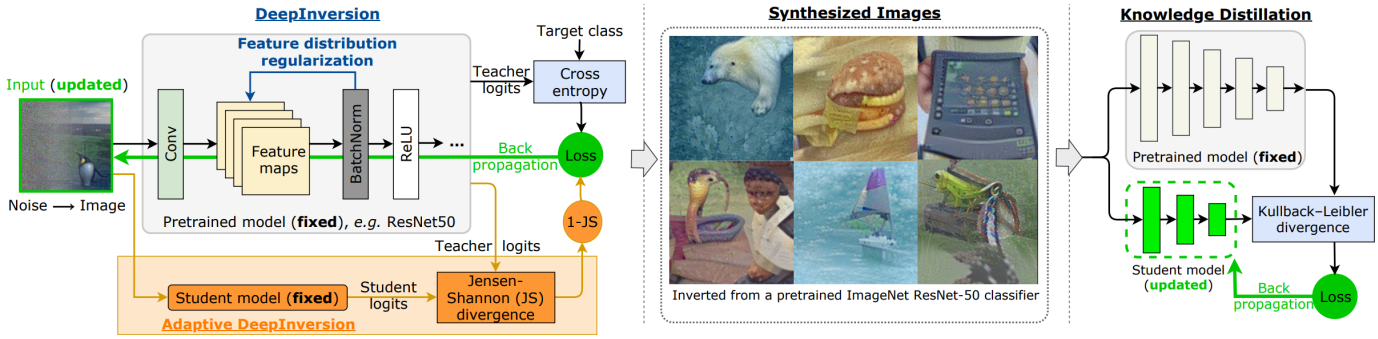

Fig. 5. The DeepInversion framework for converting random noise to high-fidelity class-conditional images, given just a pretrained CNN. Its adaptive version called Adaptive DeepInversion utilizes both the teacher and application-dependent student network to improve image diversity. The synthesized images can be used to perform various kinds of data-free knowledge distillation [46].

get updated as new data become available. For example, biomedical datasets may add new patient data over time or include new information as diseases develop. Furthermore, some datasets, such as those containing sensitive information about gender, health, or income, may not be available due to privacy concerns. To maximize efficient data handling, new approaches are being developed that actively exploit information already present in trained models.

Incorporating new training data effectively and efficiently poses a challenge for model deployment. In typical DNN development, when new data become available, the previously trained model is discarded and the DNN is retrained with all the data at hand. This is very inefficient with respect to training cost. Ideally, information learned by the neural network from the original data should be preserved, since these data instances still exist in the new dataset. A brain-inspired incremental learning framework based on the grow-and-prune synthesis paradigm can be used in this context [48]. Specifically, when new data arrive, the original model grows new connections in a gradient-directed manner. This increases the model learning capacity and ability to accommodate new data. Next, the framework prunes away insignificant connections with low-magnitude weights to derive a lightweight yet accurate model. With this incremental learning framework, the cost of training the model with new data is significantly reduced compared to traditional methods, such as retraining the model from scratch and network fine-tuning. In addition, the derived lightweight model is more accurate and requires less inference time. For ResNet-18 on the ImageNet dataset and DeepSpeech2 on the AN4 dataset, the training cost reductions compared to training from scratch (network fine-tuning) are 64% (60%) and 67% (62%), respectively. This method, however, assumes access to the original data when new data arrive.

There is growing concern for protecting data privacy. This concern further challenges efficient DNN development, since data typically used to derive compact models may not be available. In these cases, the previously mentioned methods for compact DNN design are not feasible. Recent work has made strides in making efficient DNN derivation data-free with only a trained model. One such method is DeepInversion [46], an image synthesis methodology that enables data-free knowledge transfer. One of its many applications is data-free pruning. The DeepInversion framework is shown in Fig. 5. Given only a trained CNN classifier, DeepInversion can "invert" out class-conditional images with high fidelity and diversity, without any prior information about the original dataset. It performs intermediate feature regularization using stored batch norm statistics. This enables generation of high-fidelity instances of all previously seen training classes. To further improve image diversity, an alternative framework, called Adaptive DeepInversion, uses competition regularization to encourage disagreement between a pretrained "teacher" network and an in-training "student" network. The method shows that (1) a trained deep CNN can encode a substantial amount of training distribution information, such that (2) the distribution can be sampled very effectively via DeepInversion. This work demonstrates state-of-the-art performance for data-free knowledge transfer, network pruning, and incremental learning on the CIFAR-10 and ImageNet datasets.

Fundamentally, DeepInversion transforms a discriminative network into a generative one. To synthesize images for data-free knowledge transfer, DeepInversion performs on par with BigGAN [49]. Furthermore, it does not rely on a model provider training additional generative adversarial networks on the original dataset, which can be computationally intensive, sensitive to the training setup, and challenging on large-scale datasets, e.g., ImageNet1K.

VI. Conclusions and Future Work

In this paper, we have reviewed several current trends in compact DNN design and described synthesis frameworks that automatically generate lightweight and accurate DNN architectures. So far, most existing work on efficient DNN synthesis focuses on reducing the computational and storage costs at inference time. However, training DNNs on large-scale datasets remains inefficient and expensive, easily consuming hundreds of GPU hours. More work on speeding up training will substantially benefit the entire deep learning community. In addition, more work can be done on novel model design

and optimization techniques such as hardware-software co-design. The DeepInversion methodology, for example, can be extended to explore different gradient sources for inversion, such as bounding boxes, and can be applied to different DNN architectures, e.g., LSTM and 3D CNN. Extending methods like DeepInversion to other DNN architectures can enable synthesis of video, speech, text, 3D objects, and even language models, while making all these tasks data-free.

Acknowledgments: This work was supported by NSF under Grant No. CNS-1907381.

REFERENCES

[1] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *Proc. NeurIPS*, 2011.

[2] A. Graves, A.-R. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NeurIPS*, 2014.

[4] H. Yin, B. Mukadam, X. Dai, and N. K. Jha, "DiabDeep: Pervasive diabetes diagnosis based on wearable medical sensors and efficient neural networks," *IEEE Trans. Emerging Topics in Computing*, 2019.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2014.

[8] H. Yin, Z. Wang, and N. K. Jha, "A hierarchical inference model for Internet-of-Things," *IEEE Trans. Multi-Scale Computing Systems*, vol. 4, pp. 260–271, 2018.

[9] A. O. Akmandor, H. Yin, and N. K. Jha, "Smart, secure, yet energy-efficient, Internet-of-Things sensors," *IEEE Trans. Multi-Scale Computing Systems*, vol. 4, no. 4, pp. 914–930, 2018.

[10] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LIDAR point cloud," in *Proc. ICRA*, 2018.

[11] H. Yin and N. K. Jha, "A health decision support system for disease diagnosis based on wearable medical sensors and machine learning ensembles," *IEEE Trans. Multi-Scale Computing Systems*, vol. 3, no. 4, pp. 228–241, 2017.

[12] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep Speech 2: End-to-end speech recognition in English and Mandarin," in *Proc. ICML*, 2016.

[13] B. Ballinger, J. Hsieh, A. Singh, N. Sohoni, J. Wang, G. H. Tison, G. M. Marcus, J. M. Sanchez, C. Maguire, J. E. Olgin, and M. J. Pletcher, "DeepHeart: Semi-supervised sequence learning for cardiovascular risk prediction," in *Proc. AAAI*, 2018.

[14] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," in *Proc. ICLR*, 2018.

[15] W. Wen, Y. He, S. Rajbhandari, W. Wang, F. Liu, B. Hu, Y. Chen, and H. Li, "Learning intrinsic sparse structures within long short-term memory," in *Proc. ICLR*, 2018.

[16] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[17] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[18] A. G. Ivakhnenko and V. G. Lapa, "Cybernetic predicting devices," Purdue University, Tech. Rep., 1966.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, 2015.

[22] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. CVPR*, 2018.

[23] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. ECCV*, 2018.

[24] B. Wu, A. Wan, X. Yue, P. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. Gonzalez, and K. Keutzer, "Shift: A zero FLOP, zero parameter alternative to spatial convolutions," in *Proc. CVPR*, 2018.

[25] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," in *Proc. ICLR*, 2017.

[26] J.-H. Luo, J. Wu, and W. Lin, "ThiNet: A filter level pruning method for deep neural network compression," in *Proc. ICCV*, 2017.

[27] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "NISP: Pruning networks using neuron importance score propagation," in *Proc. CVPR*, 2018.

[28] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," in *Proc. CVPR*, 2019.

[29] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. ICLR*, 2016.

[30] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient ConvNets," in *Proc. ICLR*, 2017.

[31] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proc. CVPR*, 2017.

[32] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," in *Proc. ICLR*, 2018.

[33] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. ICLR*, 2017.

[34] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. CVPR*, 2019.

[35] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. ICLR*, 2017.

[36] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. CVPR*, 2018.

[37] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, 2017.

[38] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. CVPR*, 2019.

[39] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search," in *Proc. CVPR*, 2019.

[40] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[41] X. Dai, P. Zhang, B. Wu, H. Yin, F. Sun, Y. Wang, M. Dukhan, Y. Hu, Y. Wu, Y. Jia, P. Vajda, M. Uyttendaele, and N. K. Jha, "ChamNet: Towards efficient network design through platform-aware model adaptation," in *Proc. CVPR*, 2019.

[42] X. Dai, H. Yin, and N. K. Jha, "NeST: A neural network synthesis tool based on a grow-and-prune paradigm," *IEEE Trans. Computers*, vol. 68, no. 10, pp. 1487–1497, 2019.

[43] ——, "Grow and prune compact, fast, and accurate LSTMs," *IEEE Trans. Computers*, vol. 69, pp. 441–452, 2020.

[44] S. Hassantabar, Z. Wang, and N. K. Jha, "SCANN: Synthesis of compact and accurate neural networks," *arXiv preprint arXiv:1904.09090*, 2019.

[45] J. Hawkins, "Special report: Can we copy the brain? What intelligent machines need to learn from the neocortex," *IEEE Spectrum*, vol. 54, no. 6, pp. 34–71, 2017.

[46] H. Yin, P. Molchanov, Z. Li, J. M. Alvarez, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via DeepInversion," in *Proc. CVPR*, 2020.

[47] H. Yin, G. Chen, Y. Li, S. Che, W. Zhang, and N. K. Jha, "Hardware-guided symbiotic training for compact, accurate, yet execution-efficient LSTM," *arXiv preprint arXiv:1901.10997*, 2019.

[48] X. Dai, H. Yin, and N. K. Jha, "Incremental learning using a grow-and-prune paradigm with efficient neural networks," *arXiv preprint arXiv:1905.10952*, 2019.

[49] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. ICLR*, 2019.