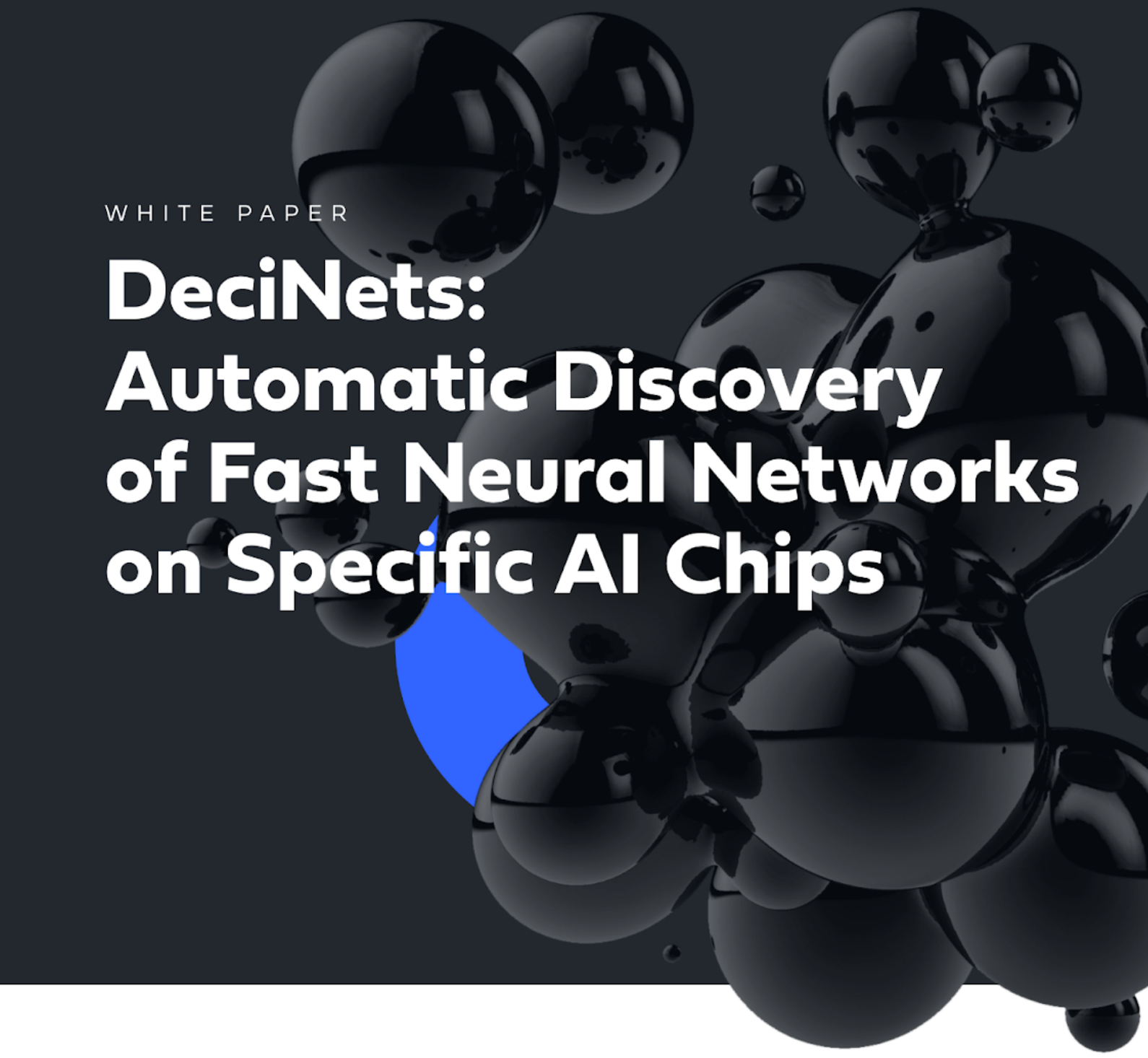
deci

Technical White Paper



WHITE PAPER

DeciNets: Automatic Discovery of Fast Neural Networks on Specific AI Chips

How Deci's AutoNAC Automatically Extended the Accuracy-Latency Efficient Frontier for NVIDIA Cloud and Edge GPUs

by Prof. Ran El-Yaniv
Chief Scientist and Co-founder at Deci

Technical White Paper

DeciNets: Automatic Discovery of Fast Neural Networks on Specific AI Chips

How Deci's AutoNAC Automatically Extended the Accuracy-Latency Efficient Frontier for NVIDIA Cloud and Edge GPUs

Executive Summary

Fast and accurate deep neural networks (DNNs) are key for successfully solving and deploying commercial AI applications. A wide and growing range of new exciting applications can be built upon deep learning models, as they become increasingly large and more accurate. However, the computational costs of operating DNNs can also be very high, placing a ceiling on the cost-effectiveness of DNN inference. Another related but distinct obstacle is the need to deploy strong DNNs on edge devices that have limited computing power. If DNNs are to achieve affordable inference costs or be deployed on edge devices, they must be made computationally efficient while retaining their accuracy and robustness. To achieve lightweight-but-accurate DNNs, DNN architectures will need to be designed for specific AI chips, while taking into consideration all available inference acceleration techniques, including compilation and quantization. Developing such neural designs requires a rare skill set, which very few commercial parties possess. Neural architecture search (NAS) is a potentially viable approach to creating such models but existing NAS algorithms fall short of this objective. Automated Neural Architecture Construction (AutoNAC), a new technology developed by Deci AI, is capable of optimizing any given neural architecture. It can also discover fast and accurate neural architectures for any task on any specific AI chip, while taking both compilation and quantization into consideration. This white paper introduces the AutoNAC technology and describes how it can be used to create proprietary DeciNet architectures that extend the efficient frontiers of latency/accuracy tradeoffs on NVIDIA cloud and edge GPUs.

1. Deep Neural Networks - the Blessing and the Curse

The modern incarnation of AI technologies is almost entirely manifested by deep neural networks (DNNs). The success of such models depends on training them to solve some task. Several types of learning frameworks exist; the main ones are supervised, unsupervised, and reinforcement learning. At present, supervised learning is the workhorse of most commercial AI applications. It works by fitting the parameters of a neural network to learn the functional relationship between input examples and their corresponding labels. The training process is defined in terms of a differentiable loss function that measures the discrepancy between the given example labels and the corresponding network predictions. The training progresses in greedy steps to correct the loss discrepancies using some variant of the stochastic gradient descent (SGD) algorithm.

DNNs have several distinct advantages relative to classical machine learning (ML) algorithms. The beauty of DNN models is their ability to learn on their own, producing effective representations and predictive features from raw data--without human intervention. Although some classical ML algorithms such as gradient boosting trees (Friedman, 2001) are also capable of learning basic predictive features, DNNs are far superior in this respect. SGD optimization algorithms train DNNs in a piecemeal fashion, which allows them to be trained with virtually unlimited amounts of data. This blessing is unmatched by classical machine learning algorithms such as support vector machines (Cortes and Vapnik, 1995) and gradient boosting trees (e.g., XGBoost), which have limitations on how they utilize data in training. For example, XGBoost suffers from the computational disadvantage that it needs to store (almost) the entire training data in memory during training. In Figure 1 we depict this major difference between DL and classical ML. While classical ML algorithms can often achieve better performance in small-data regimes, the generalization performance they can achieve as a function of the training set size reaches a plateau, whereas DL models keep improving with increasing dataset size.

> ***The beauty of DNN models is their ability to learn on their own, producing effective representations and predictive features from raw data--without human intervention***

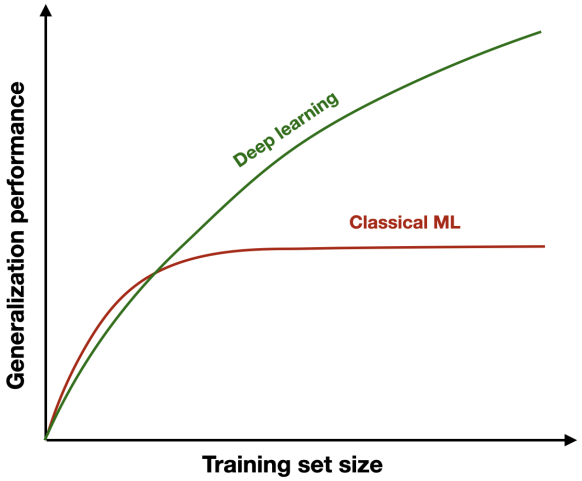


Figure 1. Comparison of DL and classical ML in terms of performance and training dataset size. As data size increases, DL models improve, while ML algorithms cannot make full use of the data.

We recently observed a number of unprecedented examples of massive training sets that are used to train huge neural models. One striking example is Open AI's GPT-3 language model, which was trained with SGD on 45 TB of text from the Internet (Brown et al., 2020). A few months after the release of GPT-3, Google announced the Switch Transformer (Fedus et al. 2021), a model that is six times larger with a trillion trainable parameters. Clearly, handling these models is a costly endeavor reserved for the very few who have the resources. In the case of GPT-3, it is estimated that the cost of training reached around $12M.

Mammoth-sized models such as GPT-3 may be extremely rare and unique, but deep net architectures are constantly growing larger and more complex, even when handling standard tasks like visual classification and detection. Deep learning engineers are more likely to achieve better accuracy by training neural networks that are larger, which is why they tend to increase model size. Using SGD training, a large model simplifies the automatic creation of better features and representations. The nature of deep learning contrasts with classical machine learning, where larger models tend to be less accurate beyond a certain complexity-generalization tradeoff. Figure 2 illustrates one manifestation of this surprising trend (Belkin et al. 2018, Naktkrian et al. 2019) referred to as 'double descent'. Here, we observe that increasing the complexity of the ResNet-18 network first harms the accuracy of the test over a distorted version of the Cifar-10 image classification dataset, but then improves it as the network width increases.

Deep learning engineers are more likely to achieve better accuracy by training neural networks that are larger, which is why they tend to increase model size

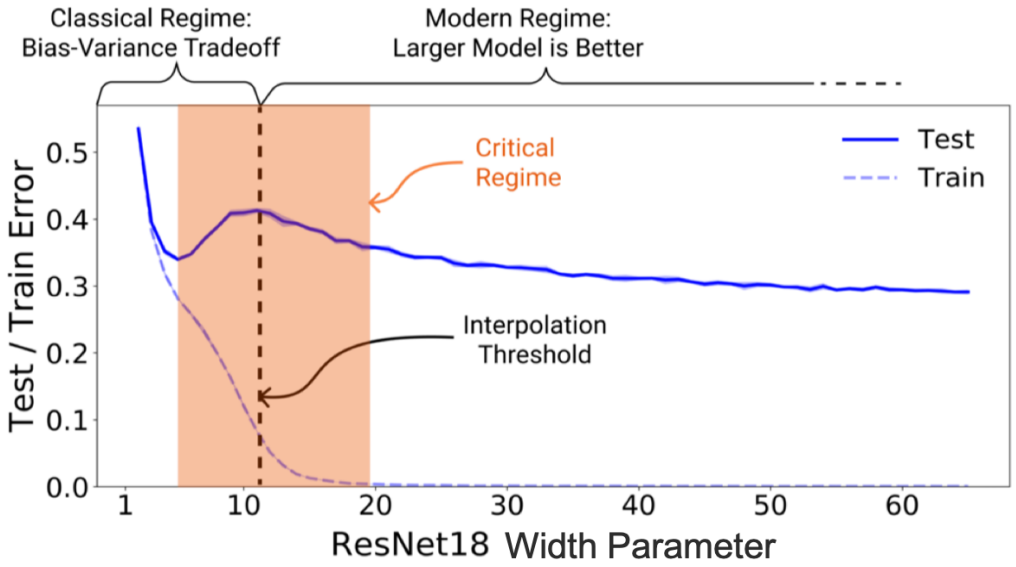


Figure 2. Double descent phenomenon. Training and testing error as a function of model size for ResNet-18s of varying width on CIFAR-10 with 15% label noise. Source: graph taken from Nakkiran et al. 2019.

We know that excessively large DNNs are easier to train for better accuracy than smaller ones, and that superior accuracy is also dependent on more data. What about the training and inference costs of excessively large deep neural networks? Can these computations be effectively supported by current and future AI chips? Two factors affect the answers to these questions: the availability of very fast and affordable AI chips as well as algorithmic acceleration techniques such as compilation, quantization, and better neural architecture designs.

> ***What about the training and inference costs of excessively large deep neural networks? Can these computations be effectively supported by current and future AI chips?***

To better understand the computational limitations of hardware, we consider forecasts regarding the developments of computing devices. As far back as 1965, Gordon Moore, who eventually co-founded Intel in 1970, wrote a paper in which he observed that the number of transistors that could fit inside an integrated circuit was doubling every year. The trend in Figure 3 illustrates this phenomenon, which is referred to as Moore's law. In the '70s, the rate of doubling was corrected to occur approximately once every two years. The curve clearly indicates that the law remained in force until at least a year ago. Moore's law is essentially about transistor density. A smaller transistor is better in principle because it can run on a lower current while wasting less energy. Thus, as long as transistors can be reduced in size, we will keep getting more powerful computers at roughly the same price, without significantly increasing their operational costs.

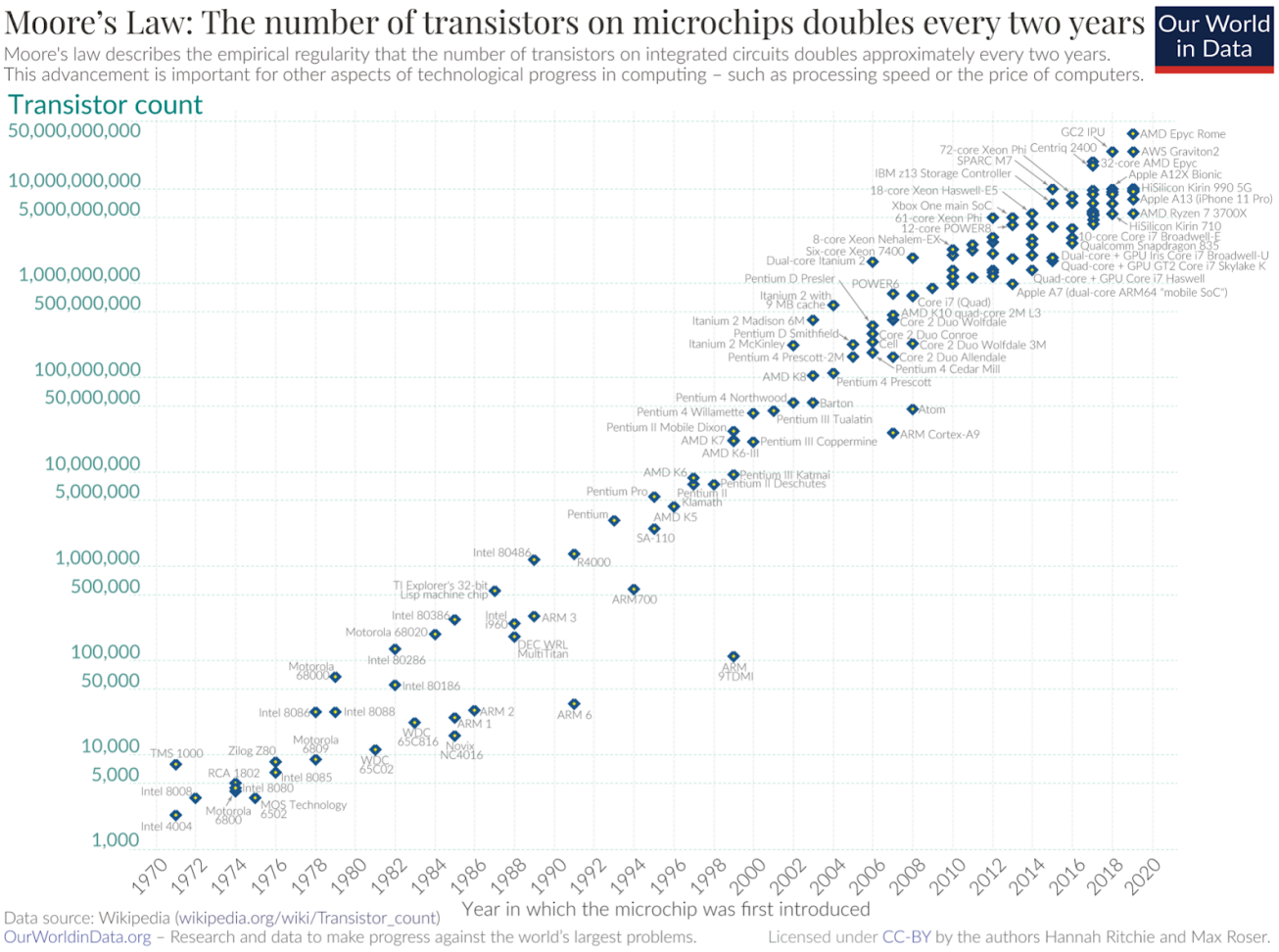


Figure 3. Moore's law, 1970 - 2020. The number of transistors in microchips doubles every two years. Source: Wikipedia [from OurWorldInData.org]

A continuation of Moore's law is good news for computationally-intensive deep learning models, allowing them to grow with supporting hardware while keeping the costs relatively constant for state-of-the-art performance. Moreover, packing more computation into smaller AI chips paves the way for the emerging trend of AI at the edge. Indeed, we are beginning to see how the smallest neural networks will operate on tiny edge devices and phones.

> ***We are beginning to see how the smallest neural networks will operate on tiny edge devices and phones***

However, several forecasters, including Gordon Moore himself, predict that Moore's law will end around 2025 (Cross, 2016). Such projections generally take into account transistor density using the current CMOS technology. In fact, transistors are becoming increasingly difficult to manufacture at ultra-small scale. The transistor's source and drain are only a few nanometers apart, which causes excessive current leakage and heating. Consequently, it's possible that the current trend of Moore's law based on CMOS

technology will break down. Often, technology leaps occur just in time to generate disruptive advances that ignite the next generation. Some of the new candidates for alternative technologies are transistors built using other materials like graphene, neuromorphic computing (Mead, 1990), biological computing (Amir et al. 2014), and quantum computing (Shor, 1998). But these technologies aren't ready yet.

> ***Often, technology leaps occur just in time to generate disruptive advances that ignite the next generation***

Even if the 2025 projection materializes, we can still witness doubling two more times, clearing the way to apply interesting AI solutions at the edge. A declining Moore's law and the absence of a replacement technology means that operating very large DNNs could be prohibitively expensive. Even today, the costs of operating DNNs at scale on the cloud can be huge. For this reason, there is a growing need for methods that can reduce large neural networks, thus shrinking their size, latency, or energy parameters while preserving high-accuracy performance.

> ***There is a growing need for methods that can reduce large neural networks, thus shrinking their size, latency, or energy parameters while preserving high-accuracy performance***

2. The Importance of Hardware Dependence and the Deep Inference Stack

The efficiency metrics we are interested in depend not only on the neural architecture, but also on the hardware and software layers on top of the target device. In terms of neural network production utility, we are primarily concerned with latency, throughput, power consumption, and memory footprint. These factors can directly influence monetary considerations and user experience or perhaps even enable operation on smaller AI chips at the edge. FLOPs, short for floating-point operations per second, are often used to evaluate the efficiency of architectures in studies that do not consider hardware. However, an architecture with a higher FLOP count may not necessarily have a lower latency or higher throughput. For instance, NasNet-A (Zoph et al. 2018) has a similar FLOP count to MobileNetV1 (Howard et al., 2017), but its complex cell-level structure makes it less amenable to certain hardware devices (e.g., Google Pixel 1 phone CPU, TF-Lite) and its latency can be higher (Sandler et al. 2017).

> ***In terms of neural network production utility, we are primarily concerned with latency, throughput, power consumption, and memory footprint***

There are huge variations in the relative efficiency of neural architectures when their performance is measured on different hardware platforms, depending on the exact objective of interest, such as latency or energy consumption. Li et al., 2021 presented a systematic study on the latency and energy consumption of many architectures over multiple AI chips. Figure 4 illustrates the correlation between architecture performance for three common visual classification benchmark datasets on various AI chips, including edge GPU, edge TPU, ASIC-Eyeriss, and others. The correlation between devices is often low (indicated by dark green and blue shading), particularly between edge-GPU or TPU and the rest of the devices.

> ***There are huge variations in the relative efficiency of neural architectures when their performance is measured on different hardware platforms***

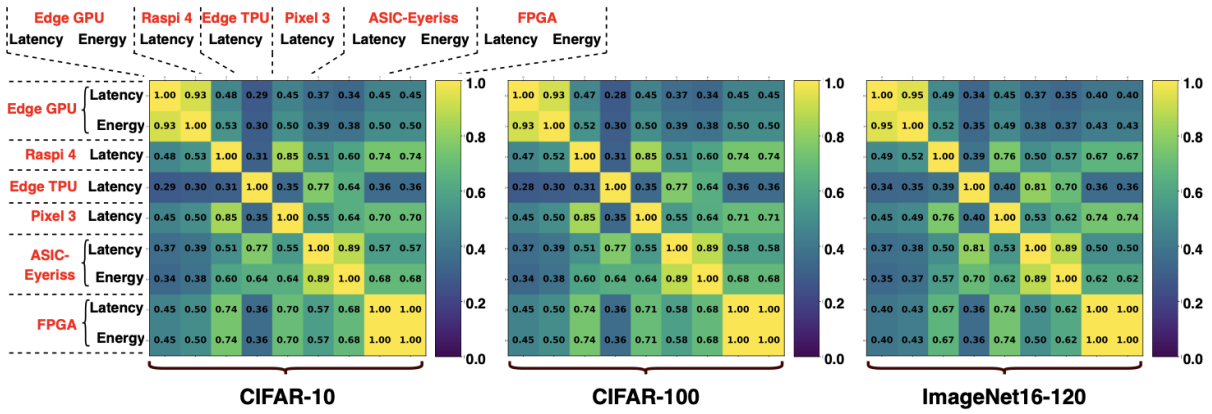


Figure 4. Kendall Rank Correlation Coefficient between real-measured/estimated hardware-cost (latency in ms, energy in mJ) in different devices calculated on the architecture space of the NAS-Bench-201 search space (Dong and Yang, 2020). Source: Li et al. (2021).

Although the different runtime properties of different architectures on different hardware families (e.g., CPUs vs. GPUs) may seem natural, we frequently encounter real-life cases where the runtime metrics of two architectures diverge between two devices in the same family. We demonstrate this phenomenon in Figure 5, illustrating the inference runtime (ms) of ResNet-50 and EfficientNet-B0 on two types of GPUs. Clearly, EfficientNet-B0 is superior to ResNet-50 when run on the K80 GPU; however, on the V100 GPU, ResNet is slightly faster.

Although the different runtime properties of different architectures on different hardware families (e.g., CPUs vs. GPUs) may seem natural, we frequently encounter real-life cases where the runtime metrics of two architectures diverge between two devices in the same family

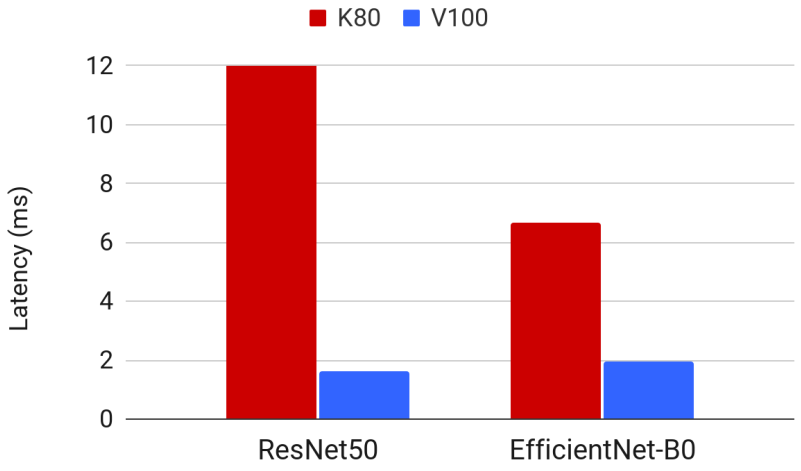


Figure 5. Latencies of ResNet-50 and EfficientNet-B0 on K80 and V100 GPUs. Source: Deci.AI

What are the factors that affect the runtime, or other performance metrics, of a given architecture on a specific type of hardware? One obvious factor is the architecture as a whole, with its inner structure and level of depth, as well as the hardware properties of the chip. Other primary factors are the degree of parallelism, often measured by the number of basic computing cores, and the effectiveness (size, speed) of the cache memory system. In this context, it is rational to assume that with only a few cores and a very effective cache system, a CPU would be more inclined to support narrow and taller nets, while a GPU would favor smaller and wider networks.

In addition to the architecture and the hardware itself, other critical factors have a direct impact on the run-time performance. These include the drivers and graph compilers, which implement the actual neural network computation on the hardware, as well as various algorithmic techniques, like weight pruning and quantization, which affect the memory footprint and runtime of the networks, respectively. An overview of the full stack of factors determining the effectiveness of inference is shown in Figure 5.

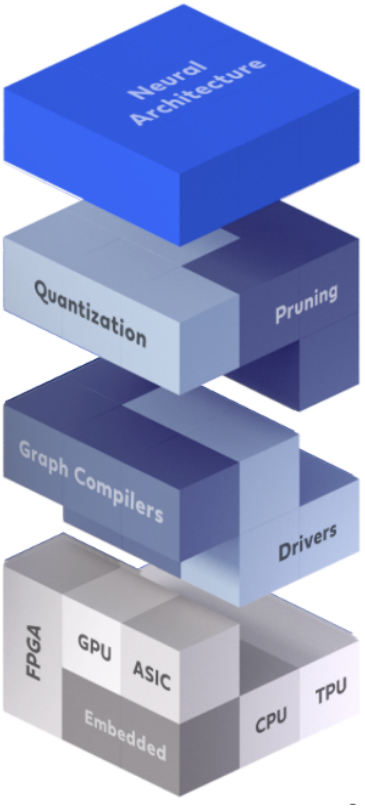


Figure 6. The deep inference stack, containing elements that affect the runtime efficiency of a given architecture on a specific hardware.

Academic papers tend to ignore graph compilers, but they are essential for achieving optimal performance on any processor. In Figure 7, we show the ResNet-50 and EfficientNet-B0 once more. The plain PyTorch runtime shows ResNet-50 to be the better

choice demonstrating almost twice the speed with 6.79 ms for ResNet vs. 11.18 ms for EfficientNet. A remarkable result was obtained after the TensorRT compilation of these nets for NVIDIA T4; EfficientNet became 5 times faster and went on to outperform ResNet-50 by a factor of 2.

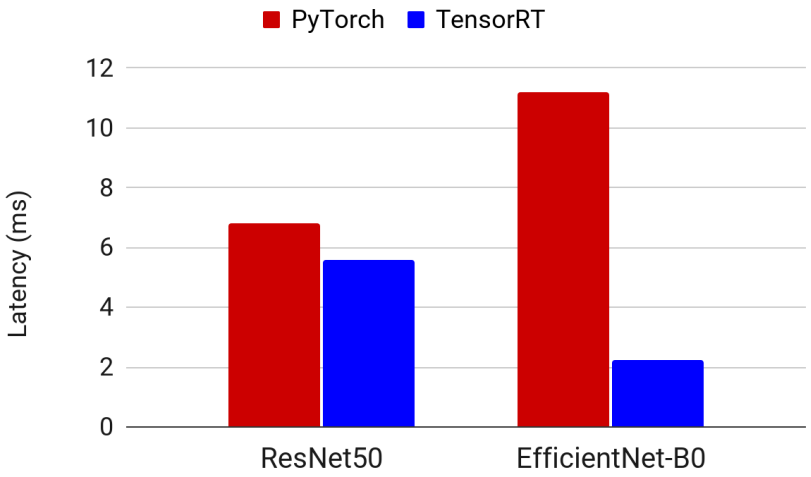


Figure 7. Latencies of ResNet-50 and EfficientNet-B0 on NVIDIA T4 (FP 32) with/without TensorRT compilation.

Compiling deep networks involves various tricks and transformations that maintain the neural network's functionality while accelerating its computation speed. Vertical and horizontal fusion are among the primary manipulations of a neural compiler. Figure 8 shows how the three consecutive layers of convolution, batch normalization, and Relu activation are combined into a single CBR block that can be computed much faster. Additionally, three blocks that perform the same computation and are connected to the input are vertically fused and then combined into a single CBR block to eliminate redundant computations.

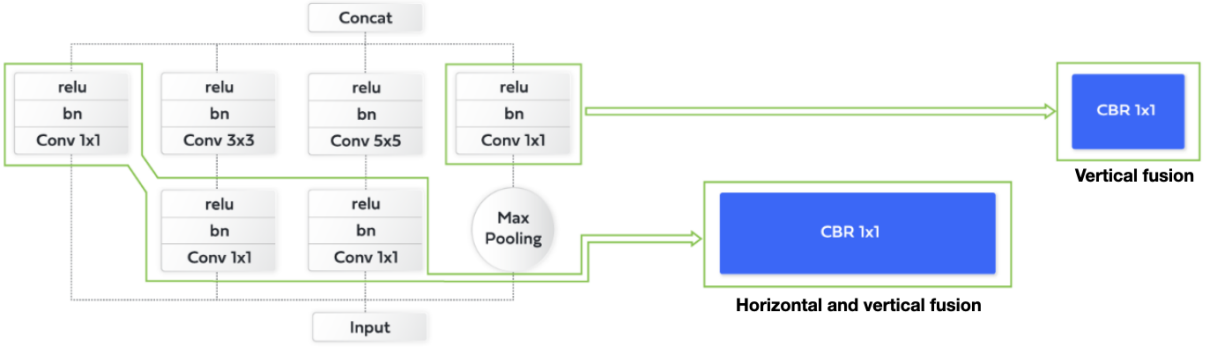


Figure 8. Example of (1) vertical fusion of three consecutive layers resulting in the rightmost CBR block (2) horizontal and vertical fusion combining the three identical blocks connected to the input, resulting in a single CBR block.

We now discuss the concept of quantization and its effects. An objective of network quantization is to represent floating point weights and/or activations using low precision compact representations (Hubara et al. 2018). Quantization allows for significantly smaller networks and more efficient computation, both in terms of tensor arithmetic operations and memory access. The extreme case of quantization is binarization, where both weights and activations are binary numbers. One challenge in constructing a quantized network is that the low bitwidth weights and activations lead to information loss, which distorts the network representation and may result in accuracy degradation.

> ***Quantization allows for significantly smaller networks and more efficient computation, both in terms of tensor arithmetic operations and memory access***

The number of bits used by conventional quantization is the same for all layers. Nevertheless, there is the potential to exploit the redundancy profiles of different layers (Wang et al., 2019). Several contemporary hardware inference accelerators already support mixed precision quantization, which allows different bit widths to be defined and applied individually to each layer. For example both the NVIDIA Turing and the more recent Ampere GPU architecture support 1-bit, 2-bit, 4-bit, 8-bit, and 16-bit operations, which can be defined per layer. Future acceleration hardware units will likely support mixed precision.

The architecture itself should be considered in a holistic manner that includes its quantization. There are some architectures that are more quantization friendly and can potentially boost performance more effectively than others. Consider the example in Figure 9, where we can observe that EfficientNet-B0 has a better latency than ResNet-50 when using 32 bits. ResNet-50, on the other hand, can be accelerated effectively by quantizing its bit width to 16 bits, whereas EfficientNet cannot be effectively accelerated through this method.

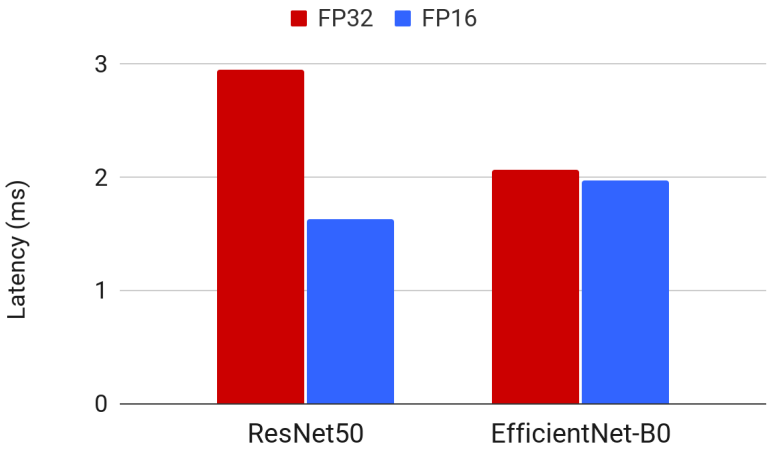


Figure 9. Effects of quantization. At bitwidth FP32, EfficientNet-B0 latency is superior to ResNet-50. ResNet-50 is faster after quantizing both networks to FP16 on NVIDIA V100 using Tensor-RT.

3. On Neural Architecture Design, Search, and AutoNAC

One of the central barriers to creating neural networks (NNs) that can successfully solve AI challenges, is the design of an appropriate “neural architecture”. This is the network's blueprint that specifies the number of neurons in each layer and how they are interconnected. Network architecture design methods have evolved over the years, and the first class of architectures were biologically "inspired". Since biological brains are still only vaguely understood, new methodologies are needed to develop effective architectures. In the second generation of neural networks, designers drew on mathematical reasoning and human insight to discover several key design principles that improve the efficiency and accuracy of their models. The most advanced architectures today were created using third-generation methods, which rely on both human intelligence and heavy-duty computation to search large architectural spaces for useful models. These neural architecture search (NAS) techniques have discovered some of the most accurate models, but require Google-scale computational resources. A fourth generation of automatic NN architecture design techniques is now emerging. Techniques in this new family consisting of sophisticated NAS algorithms are considerably more refined. They can optimize several objective metrics at once and search for hardware-dependent architectures. These new NAS techniques are expected to perform their search much more efficiently and allow for meaningful NAS applications without the need for Google-scale resources.

> ***One of the central barriers to creating neural networks (NNs) that can successfully solve AI challenges, is the design of an appropriate “neural architecture”***

> ***The most advanced architectures today were created using third-generation methods, which rely on both human intelligence and heavy-duty computation to search large architectural spaces for useful models***

> ***A fourth generation of automatic NN architecture design techniques, which consist of sophisticated NAS algorithms that can optimize several objective metrics at once and search for hardware-dependent architectures, are expected to perform much more efficiently and allow for meaningful NAS applications without the need for Google-scale resources***

Among the early architecture types is the infamous feed-forward fully-connected network structure. Here the network is organized in layers and each neuron in each layer is connected to all neurons in the subsequent layer. This

simple architecture, which was based on a crude approximation of biological brains, appears to be one of the main reasons for the futility of neural network research and applications in previous decades. The performance of these neural networks was far lower than alternative approaches such as support vector machines (Cortes and Vapnik, 1995) or gradient boosting trees (Freedman, 2001). Their performance was so inferior that grant funding was scarcely available to NN researchers in the '80s and '90s, an era that is often referred to as the (second) AI winter (see https://en.wikipedia.org/wiki/AI_winter).

The early 2000s marked the advent of a very different type of NN, called the convolutional neural network (CNN), which was very successful in applying NNs to machine vision tasks such as the recognition of handwritten digits. CNNs employ a completely different wiring of the neurons and use the concept of "weight sharing" to compute something similar to a convolution operator from signal processing. The CNN class of architectures was already envisioned earlier by Fukushima and Miyake (1982) and was also inspired by a more refined understanding of the visual cortex. At the time, it was still difficult to train large-sized CNNs that could tackle meaningful machine vision applications because of their computational complexity. One of the key engineering achievements that led to the development of large-scale applications of CNNs is the use of graphics processing units (GPUs). These GPUs are optimized to perform very fast parallel matrix multiplication, which is the core computation process for NNs and CNNs.

One of the key engineering achievements that led to the development of large-scale applications of CNNs is the use of graphics processing units (GPUs)

Human ingenuity, inspired by studies of mathematical optimization processes, and intuition drawn from statistical machine learning, resulted in numerous architectural innovations. These innovations tremendously improved the accuracy performance of CNNs, which reached human level precision in performing low-level visual recognition tasks such as classification (Shankar et al. 2020). Some of the well-known architectures for classification are VGG (Simonian and Zisserman, 2014), ResNet (He et al., 2016), Inception (Szegedy et al., 2015), and ResNeXt (Xie et al., 2017). In other application domains, such as time-series prediction and natural language processing (NLP), an innovative architecture called the Transformer was developed, based on a profound human insight stating that great representations can be constructed by hierarchical context-dependent interactions between features. Currently, these architectures are the model of choice for all NLP applications. One of the biggest models in this category is OpenAI's GPT-3, with 175 billion parameters. In almost all NLP tasks, similar giant Transformer models are currently achieving matchless performance and are also starting to be used in vision tasks.

Modern NAS systems, which emerged only a few years ago, depend on heavy computation and often on reinforcement learning controllers

Back in 1994, a neural architecture search (NAS) implementation was considered based on evolutionary optimizations. Modern NAS systems, which emerged only a few years ago, depend on heavy computation and often on reinforcement learning controllers. There are NAS algorithms that have been successfully applied to discover architectures that outperform manually designed architectures in a number of tasks such as image classification, detection, and segmentation. For instance, NAS technology is widely credited for developing several extremely successful architectures, such as MNasNet (Tan et al., 2019) and MobileNet V3 (Howard et al. 2019). A major disadvantage of these NAS algorithms is the computation time and resources they require, which can be in the order of hundreds of GPU/TPU days. A second disadvantage is that they use a heuristic utility objective function in which the hardware costs and accuracy scores are combined using an exponentially weighted product of the relevant metrics. This approach is prone to settling on suboptimal solutions.

A major disadvantage of these NAS algorithms is the computation time and resources they require, which can be in the order of hundreds of GPU/TPU days

To achieve optimal runtime, neural architectures need to be optimized for specific chips, be able to utilize graph compilers, and be capable of using quantization

The inference speed of neural networks depends critically on a number of factors along the deep inference stack (Figure 6). As we have shown above, a given neural architecture can have very different inference runtime profiles on different AI chips. To achieve optimal runtime, neural architectures need to be optimized for specific chips, be able to utilize graph compilers, and be capable of using quantization. These factors, however, severely complicate the design process. As AI chips become increasingly diverse, the implementation of hardware-dependent neural architectures becomes more complex and expensive.

A unique NAS technology developed by Deci AI enables the optimization of given NN models, as well as the exploration and invention of totally new architectures. This technology is referred to as automated neural architecture construction (AutoNAC). AutoNAC is fully aware of the entire deep learning stack, including graph compilers (e.g., Tensor RT for NVIDIA GPUs and Open Vino for Intel compilers) and quantization. In addition to its full awareness of all relevant factors, AutoNAC performs computations very quickly in comparison to known NAS methods. AutoNAC is able to perform constrained multi-objective optimization. For example, it can be used to find the fastest NN architecture compiled for a particular AI chip, achieving a prescribed

A unique NAS technology developed by Deci AI enables the optimization of given NN models, as well as the exploration and invention of totally new architectures

AutoNAC is able to perform constrained multi-objective optimization

accuracy threshold and ensuring a memory footprint that is smaller than a predefined threshold. AutoNAC can find such an architecture with a high probability if it exists in its search space.

The general objective of AutoNAC is to discover an optimal architecture a* that solves a constrained optimization problem such as the following:

$$a^* = \arg\min_{a \in A} Lat_H(a, D)$$
$$subject\ to\ Acc\,(a, D) = Acc\,(a_0, D)\,,$$

where a^* is an optimal architecture, a_0 is the original baseline model provided by the user, A is an architecture search space, D is the user dataset, $Lat_H(a,D)$ is the mean inference latency of model a over dataset D, computed over hardware platform H, and $Acc(a,D)$ is the accuracy of model a over dataset D. The optimization objective can be altered to suit other constraints. For example, one may want to maximize throughput (instead of minimizing latency), preserve accuracy, and minimize memory footprint.

AutoNAC leverages machine learning processes and principles to perform its extremely fast searches in predefined architecture spaces. To apply AutoNAC for a target AI inference chip, one needs to define a search space in accordance with the operations and neural layers supported by the target chip. The search is performed autonomously and relies on probing certain candidate architectures for their inference time (or other metrics) directly on that chip. As a result, AutoNAC optimizes the required metric explicitly rather than relying on remotely relevant proxies such as FLOPS. Despite AutoNAC's hardware awareness, the optimization strategy and process are nearly identical for any AI chip, enabling Deci to handle any use case.

AutoNAC leverages machine learning processes and principles to perform its extremely fast searches in predefined architecture spaces

AutoNAC optimizes the required metric explicitly rather than relying on remotely relevant proxies such as FLOPS

Figure 10 illustrates a typical application of AutoNAC. The inputs to the optimization process are: (1) the baseline model, a_0; (2) the data D used to train the baseline model; and (3) an access to the target AI chip H on which the optimized model is to operate. Obtaining a good starting point for its search space comes from extracting information from a_0. It also utilizes the same or similar training regime to that used to train a_0. The access to the

hardware H is used to obtain precise measurements of the optimization objectives (e.g., latency) for certain candidate architectures.

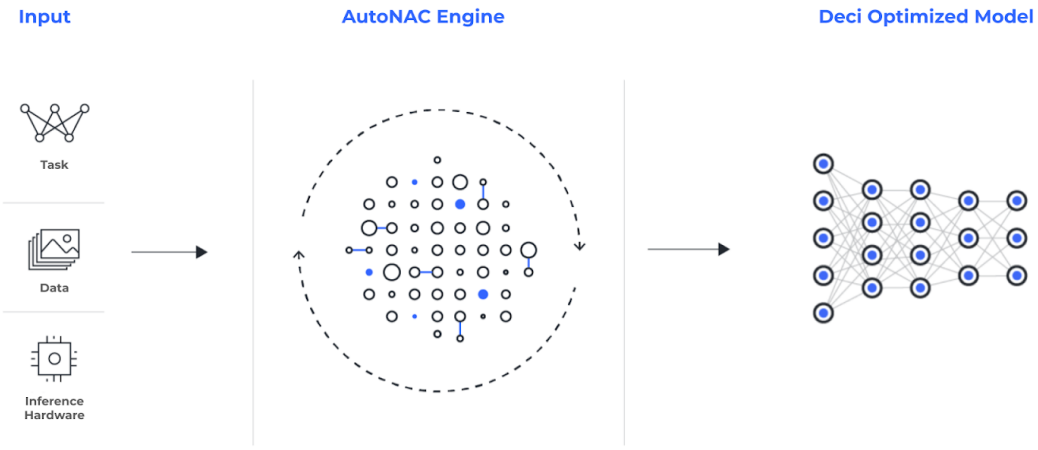


Figure 10. Schematic overview of the AutoNAC process

4. Results

To examine and showcase AutoNAC's optimization performance on specific target devices, Deci applied AutoNAC on the open source ImageNet classification task over the NVIDIA T4 GPU chip and the NVIDIA Jetson Xavier NX edge GPU. AutoNAC generated novel neural architectures called DeciNets. Strikingly, **the new DeciNet models outperform many well-known and powerful models including EfficientNets and MobileNets.** In fact, when considering the accuracy versus latency performance tradeoff for each chip, DeciNets significantly extend the efficient frontier for this task. The results for the NVIDIA T4 GPU are depicted in the scatter plot of Figure 11, showing the ImageNet top-1 accuracy performance versus inference latency (batch size 1) for many well-known models. Each presented model was compiled to the T4 GPU using Tensor RT, and quantized to both 8-bit (INT8) and 16 bit (FP16) precision. For each model, the better among these two quantized models is presented in the plot. The new DeciNets architectures clearly dominate the baseline architectures and advance the state-of-the-art for the T4 chip. For instance, EfficientNet B2 achieves 0.803 top-1 ImageNet accuracy with an inference time of 1.950 mili-seconds. DeciNet-4 has the same accuracy and is almost 30% faster, while DeciNet-5 is both more accurate and faster. Importantly, each of the DeciNets was discovered by AutoNAC using roughly four times the computation required to train a single network. It is interesting to note that EfficientNet was discovered using a third-generation NAS technology that required roughly two orders of magnitude more compute power for its optimization.

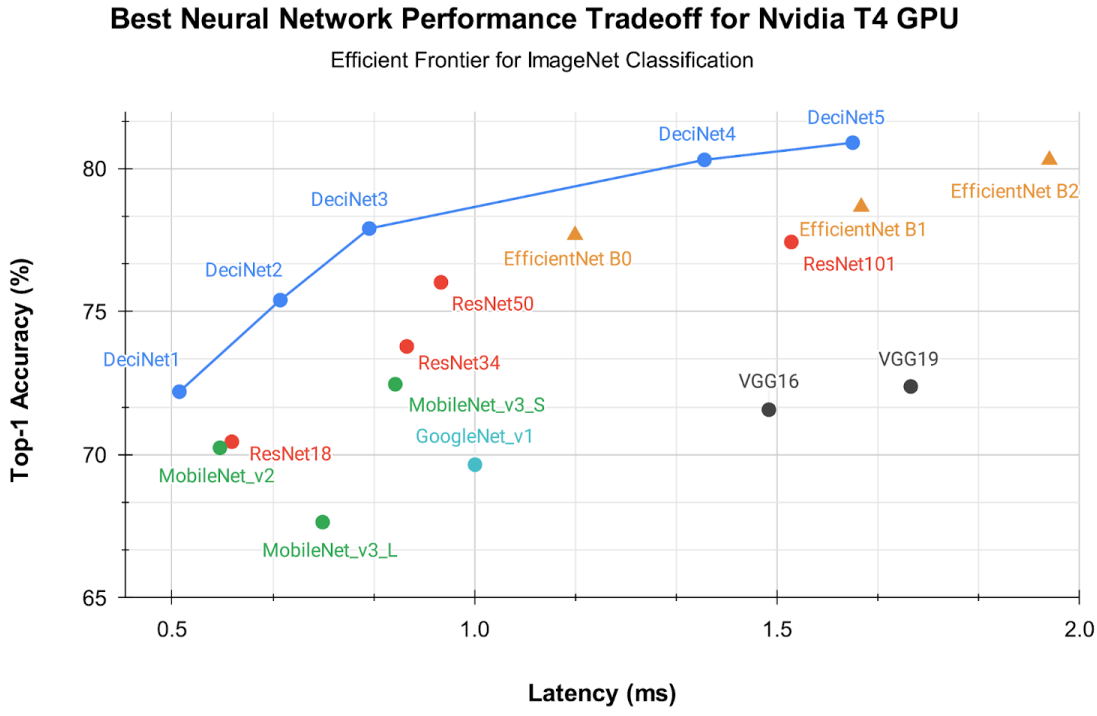


Figure 11. Best neural network performance tradeoff for Nvidia T4 GPU. Accuracy vs. latency (ms) for DeciNet instances (blue) and various well-known deep learning classification models.

Quantization levels were selected for each model to maximize accuracy-latency tradeoff. FP16 quantized models appear as triangles, while INT8 quantized models appear as dots. All models were quantized using TensorRT quantization following MLCommons rules. Source: Deci AI.

Precise performance metrics for the NVIDIA T4 GPU of all the baselines and DeciNets are presented in Table 1. The table is grouped based on latency measurements. Each section presents the dominating DeciNet as compared to its relevant baselines alongside the percent of DeciNet improvements in terms of latency and accuracy. With the exception of a single case where DeciNet5 lost 0.03% in accuracy relative to EfficientNet-B1 (and improved its latency by 10%), the DeciNets dominated all baselines in terms of both accuracy and latency.

Best Neural Network Performance Tradeoff for Nvidia T4 GPU

Model	Top-1 Accuracy (%)	Latency (ms)	DeciNet Latency Improvement (%)	DeciNet Accuracy Improvement (%)
DeciNet1	**72.2**	**0.512**		
MobileNet_v3_S	67.64	0.749	31.60	6.74
GoogleNet_v1	69.65	1.001	48.81	3.66
MobileNet_v2	70.24	0.579	11.58	2.79
ResNet18	70.45	0.599	14.46	2.48
DeciNet2	**75.4**	**0.679**		
VGG16	71.57	1.486	54.33	5.35
VGG19	72.38	1.721	60.55	4.17
MobileNet_v3_L	72.46	0.869	21.87	4.06
DenseNet121	72.57	3.125	78.27	3.90
ResNet34	73.78	0.888	23.54	2.20
DeciNet3	**77.9**	**0.826**		
ResNet50	76.02	0.944	12.55	2.47
DenseNet161	76.42	5.976	86.18	1.94
ResNet101	77.43	1.523	45.78	0.61
EfficientNet B0	77.67	1.166	29.16	0.30
DeciNet4	**80.3**	**1.380**		
EfficientNet B1	78.66	1.639	15.81	2.08
DeciNet5	**80.9**	**1.625**		

Table 1. NVIDIA T4 GPU, Top-1 ImageNet accuracy (&) and inference latency (ms) and accuracy/latency performance improvement (%) of DeciNets relative to baselines.

In a second study, we examined the application of AutoNAC where the target device is the NVIDIA Jetson NX GPU. This AI chip is designed for deploying and optimizing visual perception systems, autonomous robots and drones, handheld medical devices, smart appliances, surveillance cameras, and more. Here again the resulting architectures, called DeciNet Jets, which are now optimized to the Xavier GPU, significantly outperform the well-known baseline architectures. Figure 12 presents the resulting efficient frontier, comparing the accuracy-latency profiles of the various models. Once again AutoNAC was able to extend the efficient frontier and generate novel and powerful architectures for the Xavier edge GPU. In this Xavier edge GPU case, we observe even more remarkable speedups. For instance, the EfficientNet-B0 with its 0.777 top-1 accuracy at 0.525 ms latency, is improved by DeciNet-3 both in terms of accuracy and by a factor of over 2x in latency.

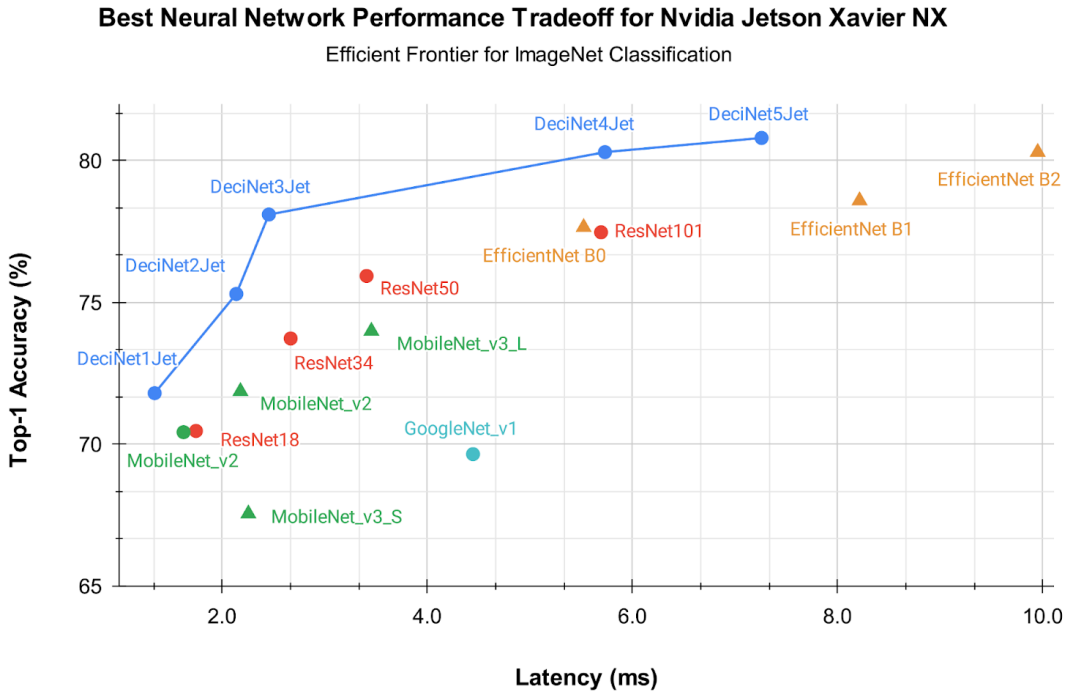


Figure 12. Best neural network performance tradeoff for Nvidia Jetson Xavier NX edge GPU. Accuracy vs. latency (ms) for DeciNet instances (blue) and various well-known deep learning classification models. Quantization levels were selected for each model to maximize accuracy-latency tradeoff. FP16 quantized models appear as triangles, while INT8 quantized models appear as dots. All models were quantized using TensorRT quantization following MLCommons rules. Source: Deci AI.

Precise performance metrics for the NVIDIA Jetson Xavier NX edge GPU of all the baselines and DeciNets are presented in Table 2. As in Table 1, this table is divided into latency sections. With the exception of a single case where DeciNet4 Jet lost 0.75% in

accuracy relative to EfficientNet-B1 (and improved its latency by 126%), the DeciNets dominated all baselines in terms of both accuracy and latency.

Best Neural Network Performance Tradeoff for Nvidia Jetson Xavier NX

Model	Top-1 Accuracy (%)	Latency (ms)	DeciNet Latency Improvement (%)	DeciNet Accuracy Improvement (%)
DeciNet1 Jet	**71.800**	**1.343**		
MobileNet_v3_S	67.550	2.257	40.5	6.29
GoogleNet_v1	69.650	4.446	69.8	3.09
MobileNet_v2	70.424	1.624	17.3	1.95
ResNet18	70.468	1.746	23.1	1.89
DeciNet2 Jet	**75.300**	**2.139**		
ResNet34	73.728	2.669	19.8	2.13
MobileNet_v3_L	73.992	3.456	38.1	1.77
ResNet50	75.934	3.409	37.3	-0.83
DeciNet3 Jet	**78.100**	**2.456**		
ResNet101	77.474	5.695	56.9	0.81
EfficientNet B0	77.652	5.525	55.6	0.58
DeciNet4 Jet	**80.300**	**5.733**		
EfficientNet B1	78.594	8.215	30.2	2.17

Table 2. NVIDIA Jetson Xavier NX GPU Top-1 ImageNet accuracy (%) and inference latency (ms) and accuracy/latency performance improvement (%) of DeciNets relative to baselines.

In this paper, we only report on AutoNAC's results for the publicly available ImageNet dataset. Nevertheless, AutoNAC was already applied dozens of times on a wide variety of proprietary datasets and tasks of Deci's customers, and on a wide variety of AI chips. In many of these cases, AutoNAC achieved better (accuracy-preserving) speedups. In general, we expect AutoNAC to achieve more striking speedups when optimized for less explored AI chips and less studied tasks and datasets. At the same time, an attractive property of AutoNAC is that while it is hardware-aware, its application routine is nearly identical on all devices.

5. Concluding Remarks and Outlook

We presented AutoNAC, a powerful neural architecture search and design algorithm that has full awareness of data, hardware, and the deep inference stack. Although hardware aware NAS algorithms have been proposed in recent academic papers (Tan et al., 2019), these algorithms require intensive computation to perform their search. AutoNAC operates approximately two orders of magnitude faster. We described two applications of AutoNAC to explore the Pareto frontier for two NVIDIA GPUs: the T4 and Jetson Xavier NE. While the T4 is a popular choice for inference on the cloud, the Jetson Xavier is one of the chips of choice for deep learning inference at the edge to accommodate exciting new applications such as autonomous driving.

> ***Although hardware aware NAS algorithms have been proposed in recent academic papers, these algorithms require intensive computation to perform their search--AutoNAC operates approximately two orders of magnitude faster***

Hardware-dependent neural architecture design is essential for obtaining optimal inference performance. While here we have only considered the publicly available ImageNet dataset, there is also considerable importance for data-dependent architecture design. For example, well-known open source architectures such as the EfficientNets were discovered to maximize ImageNet classification performance. Even though this classification task has 1000 classes, many commercial use cases often consider smaller and more refined tasks. As a result, architectural designs that are aware of the task at hand, as well as the target hardware device, have much to gain in terms of inference cost. One of the barriers in designing such specialized architectures is the complexity of the functional interaction between relevant factors such as the architecture structure, the actual task/dataset, and the structure of the hardware device. Overcoming this considerable complexity is extremely challenging even for the best human experts. AutoNAC makes these highly beneficial capabilities accessible to any party. To the best of our knowledge, AutoNAC is the only commercially available NAS technology offering affordable neural architecture design that can handle any task and hardware.

Deep learning modeling is progressing rapidly due to burgeoning research attention from universities and industry giants. New neural architecture designs and huge architectures are frequently announced. As a result of these new ideas and implementations, both accuracy and size records are frequently broken. Moreover, new inductive bias ideas are demonstrated to work well on old and new tasks. For instance, tabular datasets have been considered the last unconquered territory for deep

> ***AutoNAC is the only commercially available NAS technology offering affordable neural architecture design that can handle any task and hardware***

learning, with classical methods still outperforming DNNs (Kadra et al., 2021). New specialized architectures (Katzir et al. 2020) and regularization techniques (Kadra et al., 2021) have been demonstrating that DNNs are also capable in this domain.

Looking forward, it is anticipated that deep models will soon conquer almost all AI tasks and datasets. We will likely see larger models that can simultaneously handle multi-modal tasks involving many types of data that achieve superior results in very advanced AI applications. The importance of algorithmic acceleration techniques is expected to grow to accommodate the cost-effectiveness of these new models on the cloud, and enable their application on edge devices. Tools and techniques such as AutoNAC are instrumental in breaking those design barriers and enabling deep AI applications everywhere.

References

Amir, Yaniv, Eldad Ben-Ishay, Daniel Levner, Shmulik Ittah, Almogit Abu-Horowitz, and Ido Bachelet. "Universal computing by DNA origami robots in a living animal." Nature nanotechnology 9, no. 5 (2014): 353-357.

Belkin, Mikhail, Daniel Hsu, Siyuan Ma, and Soumik Mandal. "Reconciling modern machine learning and the bias-variance trade-off." arXiv preprint arXiv:1812.11118 (2018).

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. and Agarwal, S., 2020. Language models are few-shot learners. arXiv preprint arXiv:2005.14165.

Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

Cross, Tim. "After Moore's Law". The Economist Technology Quarterly. Retrieved 2016-03-13.

Dong, Xuanyi, and Yi Yang. "Nas-bench-201: Extending the scope of reproducible neural architecture search." ICLR 2020.

Fedus, W., Zoph, B. and Shazeer, N., 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. arXiv preprint arXiv:2101.03961.

Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." Annals of statistics (2001): 1189-1232.

Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." In Competition and cooperation in neural nets, pp. 267-285. Springer, Berlin, Heidelberg, 1982.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In European conference on computer vision, pp. 630-645. Springer, Cham, 2016.

Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V. and Le, Q.V., 2019. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1314-1324).

Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. The Journal of Machine Learning Research, 18(1), 6869-6898.

Kadra, Arlind, Marius Lindauer, Frank Hutter, and Josif Grabocka. "Regularization is all you Need: Simple Neural Nets can Excel on Tabular Data." arXiv preprint arXiv:2106.11189 (2021).

Katzir, Liran, Gal Elidan, and Ran El-Yaniv. "Net-DNF: Effective Deep Modeling of Tabular Data." In International Conference on Learning Representations. 2020.

Li, Chaojian, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, and Yingyan Lin. "Hw-nas-bench: Hardware-aware neural architecture search benchmark." ICLR, 2021.

Mead, Carver. "Neuromorphic electronic systems." Proceedings of the IEEE 78, no. 10 (1990): 1629-1636.

Nakkiran, Preetum, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep double descent: Where bigger models and more data hurt." arXiv preprint arXiv:1912.02292 (2019)

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

Shankar, Vaishaal, Rebecca Roelofs, Horia Mania, Alex Fang, Benjamin Recht, and Ludwig Schmidt. "Evaluating machine accuracy on imagenet." In International Conference on Machine Learning, pp. 8634-8644. PMLR, 2020.

Shor, Peter W. "Quantum computing." Documenta Mathematica 1, no. 1000 (1998): 467-486.

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

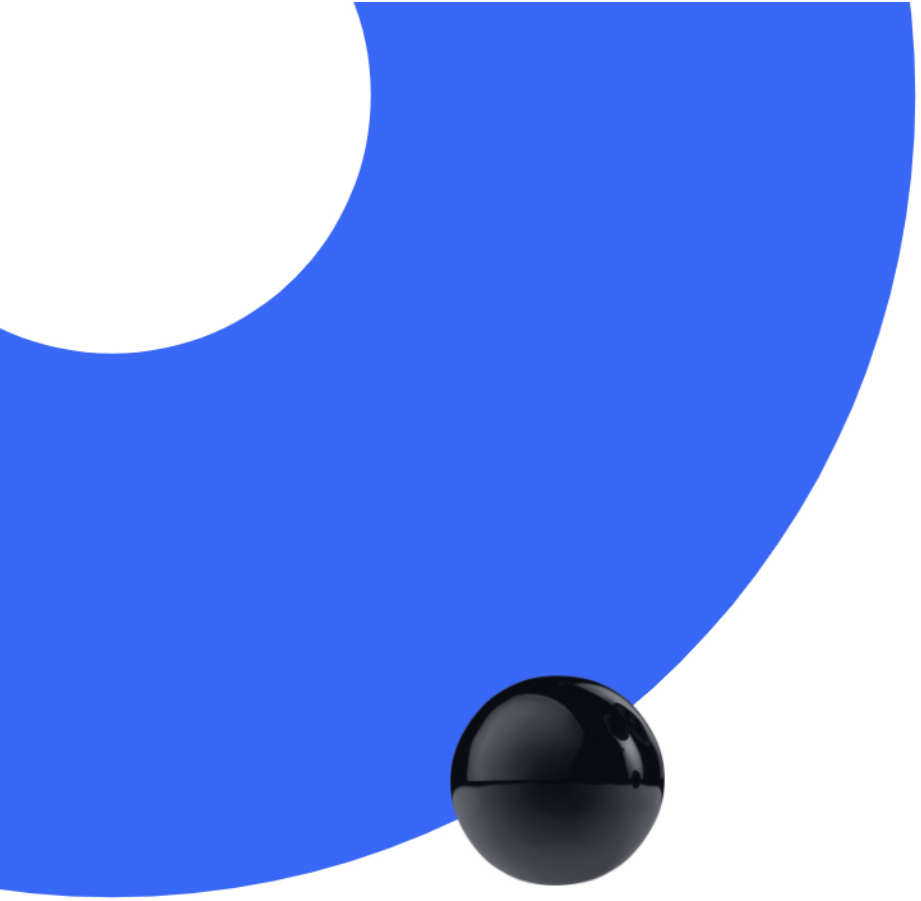
Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

Tan, Mingxing, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. "Mnasnet: Platform-aware neural architecture search for mobile." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820-2828. 2019.

Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8612-8620).

Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1492-1500. 2017.

Zoph, Barret, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. "Learning transferable architectures for scalable image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697-8710. 2018.



About the Author

Ran El-Yaniv is a co-founder and the chief scientist at Deci AI, and a full professor of computer science at the Technion - Israel Institute of Technology. Prior to founding Deci, Ran was a visiting Staff Research Scientist at Google. Ran's research activities span numerous topics in machine learning, deep learning, and AI. He received a PhD in computer science from the University of Toronto, and completed his post-doctoral studies at the MIT Computer Science Laboratory and the Center for Rationality at the Hebrew University. Ran has been serving as area chair in premier machine learning and AI conferences including NeurIPS, ICML, and IJCAI. He is a co-author of the book Online Computation and Competitive Analysis (Cambridge University Press), and served as associate editor at the Journal of Artificial Intelligence Research, and a member of the editorial board at the Journal of Machine Learning Research. Ran received the 2016 Yanai Prize for Excellence in Academic Education and the Best Research paper in ISMIR 2020.

About Deci

Deci enables deep learning to live up to its true potential by using AI to build better AI. With the company's end-to-end deep learning acceleration platform, AI developers can build, optimize, and deploy faster and more accurate models for any environment, including cloud, edge or mobile.

With Deci's platform, developers can increase deep learning model inference performance by 3x-15x, on any hardware, while still preserving accuracy. This translates directly into new use cases on limited hardware, substantially shorter development cycles, and reduced compute costs by up to 80%. The platform is powered by Deci's Automated Neural Architecture Construction (AutoNAC) technology, an algorithmic optimization engine that squeezes maximum utilization out of any hardware. The AutoNAC engine contains a Neural Architecture Search (NAS) component that redesigns a given trained model's architecture to optimally improve its inference performance (throughput, latency, memory, etc.) for specific target hardware while preserving its baseline accuracy.

Deci achieved a record-breaking 11.8x accelerated inference speedup on Intel CPUs at MLPerf Industry Benchmark and has been named to the CBInsights top 100 AI companies. Led by a team of world-class deep learning experts, Deci lets AI developers focus on what they do best - creating innovative AI-based solutions for our world's most complex problems. For more information visit us at www.deci.ai