DEEPCUBE

CubeIQ

CubeIQ PRODUCT BRIEF

MAKE YOUR MODELS 90% LIGHTER WITHOUT LOSING ACCURACY

Train your neural models to execute 10X faster on any target hardware, with full automation

MAIN FEATURES

- 85-90% reduction in parameters (sparsification), with less than 0.2% accuracy drop
- Full automation with ready-made recipes
- Multiple sparsification patterns support:
  - Block, tile, random, mix
  - CPU sparse kernel patterns
  - AI ASIC patterns
  - Custom instruction set patterns
  - GPU patterns
- Support PyTorch 1.7 and up
- Installed as PyTorch library (.whl)
- Compatible with all PyTorch optimizers (Adam, SGD, Lars, etc.)
- Compatible with in-training and post-training quantization
- Generate PyTorch, ONNX and DCM model formats
- SDK with examples available
- Custom Model Zoo available

Get Started Today

Self-service CubeIQ portal to experiment, benchmark and evaluate

www.deepcube.com/portal

ML research is on collision path with the physical world

The ML/AI industry has arrived at a massive, at-scale obstacle.

While breakthrough ML research has produced super-performing neural network models, these models are born and tested in the cloud or in a research setting, where compute resources and other physical constraints (memory, power, etc.) do not exist, or at least are very relaxed.

When those models are deployed in the real physical world (edge, cloud, enterprise and carriers), with costs and other metrics in action, hiccups, pain points and incompatibilities start emerging – throughput is insufficient, latency is out of reach, and costs, power consumption and size are out of control.

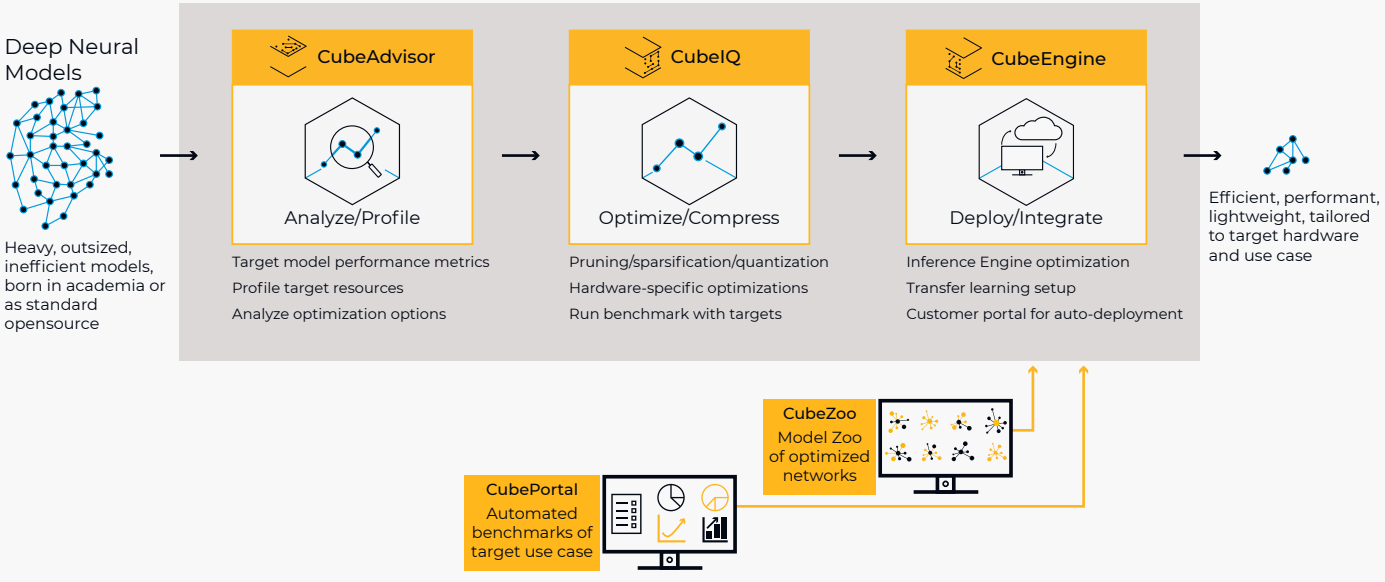
Many solutions have been proposed to take a model, and hack-prune-quantize-compress it post-training to fit onto the constrained environment. However, these attempts have fallen short as they inflict severe brain damage to the model, resulting in a poor performance, a decreased in accuracy, or both. A new plan is needed.

Why CubeIQ

CubeIQ is the first fully automated training framework that can take a model and surgically eliminate unnecessary parameters based on a multi-objective optimization. Our patented approach ensures that physical, real-world constraints and target hardware profiles are accounted for during training and enables the training algorithm to accomplish extreme model compression. Furthermore, the fingerprint of the target hardware (on which the model will be executed) is factored into the pruning and sparsification processes. While over the years, several sparsification research papers where offered, they are crippled by a few key challenges that CubeIQ has fixed from ground up:

- Implementing a research concept to a real hardened, automated and proven training library is a multi-year project. We've done the work.
- Applying one compression (sparsification) algorithm will work on a subset of models, but definitely not for all. CubeIQ implemented a wide array of patterns and algorithms to cover all model families – NLP, CNN, RNN, transformers, recommendation, generic MLP and non-standard models – with no accuracy drop.
- Automation is critical. Otherwise, the process will remain a one-time research exercise and will not scale. With CubeIQ, two python lines of code are inserted to the training scripts, and that's it: automation, unlocked.
- Sparsifying a model without prior knowledge of where it will run is like training a blind-folded person to read. CubeIQ is designed to have a set of arguments that define the target hardware (executing the inference), so the surgical pruning will yield 10X performance gains (speed-up and latency).
- Your CubeIQ license can be executed on-prem, you do not need to upload your sensitive dataset to a SaaS platform or cloud infrastructure.

DeepCube Workflow



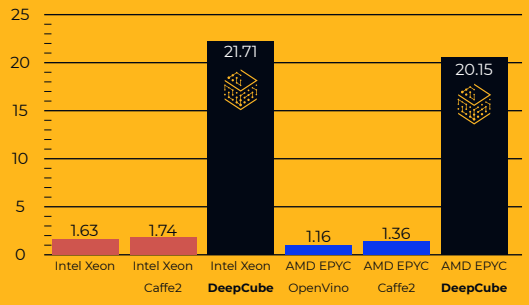
CubeIQ generates the industry's thinnest models in each category that offer the smallest compute footprint (and/or highest performance) to accomplish accurate predictions, anywhere and with any hardware.

CubeIQ can optionally work with CubeEngine, our accelerated inference engine for x86 architecture, and CubeAdvisor, our optional consulting and customer implementation service.

CubeIQ compressed models are purpose-built for extreme speed-up and low latency. The following benchmarks represent a sample of a larger set.

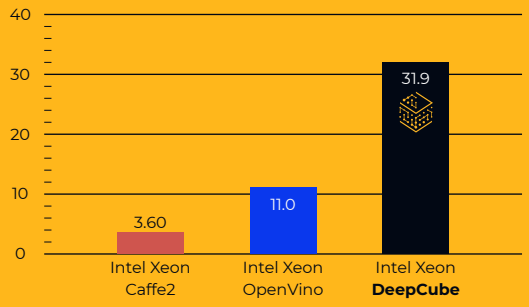
NLP

BERT-Large SQuAD samples per second (FPS)



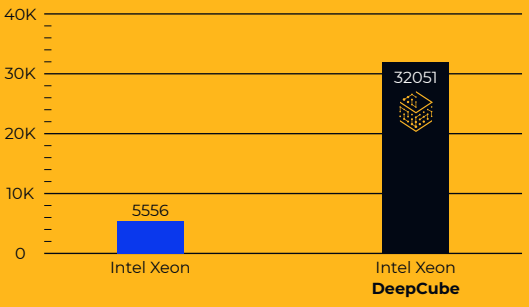
Computer Vision

ResNet-50 frames per second (FPS) single thread



DLRM

Kaggle Inferences Per Second



Start Now

Run these benchmarks and more on your own:

www.deepcube.com/portal

CubeIQ economics

CubeIQ will drive revenue and new customer business in a number of ways. First, it will minimize the compute foorptint needed for inference load, translating to 75% cost savings (and power, space, lifetime costs, maintenance, etc.). But this is just the beginning. Running cloud-class models at the edge, or on highly constrained devices or edge servers, will usher in a new set of use cases where ML can create new business models.

Imagine cameras that can deliver 60 FPS of full object detection and segmentation under 5Watt. Drones that can decode complex topography and imagery in real-time without active downlink; or power-optimized server supporting more than one computer vision model across 64 HD video feeds.

All market verticals will benefit from ML software acceleration: healthcare, with transcription of all day docs to an actionable database; manufacturing with real-time yield analysis across a vast array of images and video frames; gaming, with real-time in-game profiling of gamers/participants, and the list goes on. Whether it is revenue growth, Capex savings or Opex savings, CubeIQ is here to change the game for you.

CubeIQ Automation – optimizing data scientists' productivity

Pruning and compressing, without sacrificing accuracy, is hard. We know, because we researched this space for several years to arrive at automation breakthroughs that work across model families and are easy to run. The deep tech science is executed under the hood, with merely two python lines of code and simple recipes offered to data scientists. This alleviates the need for huge teams of expert researchers and top-notch data scientists to compress models or drive long-term projects. With CubeIQ, the training process is simple, user-friendly, and will not extend the training time – all in one training run.

```
from Deepcube import DC_S_optimizer
Optimizer = DC_S_optimizer)model,  optimizer,  target_density)
```

Training with DeepCube

A simple add-on of two python lines of code to your training script.

```
class BertAdam(Optimizer):
        """Implements BERT version of Adam algorithm with weight decay fix.
        Params:
            lr: learning rate
            warmup: portion of t_total for the warmup, -1  means no warmup. Default: -1
            t_total: total number of training steps for the learning
                rate schedule, -1  means constant learning rate. Default: -1
            schedule: schedule to use for the warmup (see above). Default: 'warmup_line
            b1: Adams b1. Default: 0.9
            b2: Adams b2. Default: 0.999
            e: Adams epsilon. Default: 1e-6
            weight_decay_rate: Weight decay. Default: 0.01
            max_grad_norm: Maximum norm for the gradients (-1 means no clipping). Defau
        """
        def __init__(self, params, lr, warmup=-1, t_total=-1, schedule='warmup_linear',
                     b1=0.9, b2=0.999, e=1e-6, weight_decay_rate=0.01,
                     max_grad_norm=1.0):
            if not lr >= 0.0:
                raise ValueError("Invalid learning rate: {} - should be >= 0.0".format(
            if schedule not in SCHEDULES:
                raise ValueError("Invalid schedule parameter: {}".format(schedule))
            if not 0.0 <= warmup < 1.0 and not warmup == -1:
                raise ValueError("Invalid warmup: {} - should be in [0.0, 1.0[ or -1".f
            if not 0.0 <= b1 < 1.0:
                raise ValueError("Invalid b1 parameter: {} - should be in [0.0, 1.0[".f
            if not 0.0 <= b2 < 1.0:
                raise ValueError("Invalid b2 parameter: {} - should be in [0.0, 1.0[".f
            if not e >= 0.0:
                raise ValueError("Invalid epsilon value: {} - should be >= 0.0".format(
            defaults = dict(lr=lr, schedule=schedule, warmup=warmup, t_total=t_total,
                            b1=b1, b2=b2, e=e, weight_decay_rate=weight_decay_rate,
                            max_grad_norm=max_grad_norm)
            super(BertAdam, self).__init__(params, defaults)

        def get_lr(self):
            lr = []
            for group in self.paran_groups:
                for p in group['params']:
                    state = self.state[p]
                    if len(state) == 0:
                        return [0]
                    if group['t_total'] != -1:
                        schedule_fct = SCHEDULES[group['schedule']]
                        lr_scheduled = group['lr'] * schedule_fct(state['step']/group['
                    else:
                        lr_scheduled = group['lr']
                    lr.append(lr_scheduled)
            return lr
```

Training without DeepCube

Competing solutions designed for researchers, with no automation. These require hand-tuning and multiple training passes; and typically, only work for one model family (e.g. computer vision exclusively).

Start with CubeIQ today

See CubeIQ's performance and usability for yourself. Contact us at contact@deepcube.com for access to our portal service (hosted in the cloud – no installation required!)

Need expert help with special ML projects and a constrained environment to run models? We are here to provide assistance through our CubeAdvisor platform.

Want to completely offload your training project? We'd be happy to run your training and generate a private model zoo.

We are ready to help you deploy ML at scale. Anywhere.

About DeepCube

DeepCube is an award-winning deep learning pioneer that provides the industry's first software-based deep learning acceleration platform that drastically improves performance on existing hardware. Modeled after the way the human brain develops during childhood, DeepCube's patented technology is the first to be purpose-built for deployment of deep learning models in data centers and on intelligent edge devices. Its proprietary framework can be deployed on top of any existing hardware, resulting in drastic speed improvement and memory reduction.

Led by a team of experienced deep learning researchers and developers, DeepCube has patented numerous innovations, including methods for faster and more accurate training of deep learning models, and drastically improved inference performance.

For more information, please contact us at: www.deepcube.com | in | contact@deepcube.com