MACHINE LEARNING meetups

BRATISLAVA PRAHA KOŠICE BRNO

Today:

Slavomír Maťašovský

Tachyum™

Data Analytics Meetings

World's 1st Universal Processor for Servers / AI / HPC

Server / Supercomputer / AI Chip

- For hyperscale datacenters

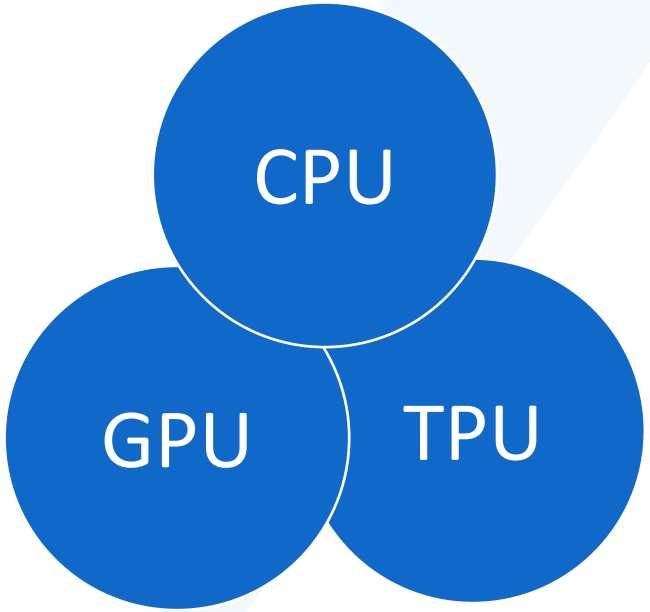
Humanity: 1st human brain sized AI

- Not only Focus on Deep Learning AI
- Also Explainable, Bio, Spiking and General AI

Prodigy is faster than Xeon/GPU/TPU

- Faster, 10x less power, 1/3 cost of Xeon
- Faster than NVIDIA A100 in HPC and AI

Tachyum Universal Processor is Best of



AI The Most Important Driver of GDP Growth

Bloomberg 2018

AI adds $15T to the economy by 2030

Forbes 2017 AI & GDP

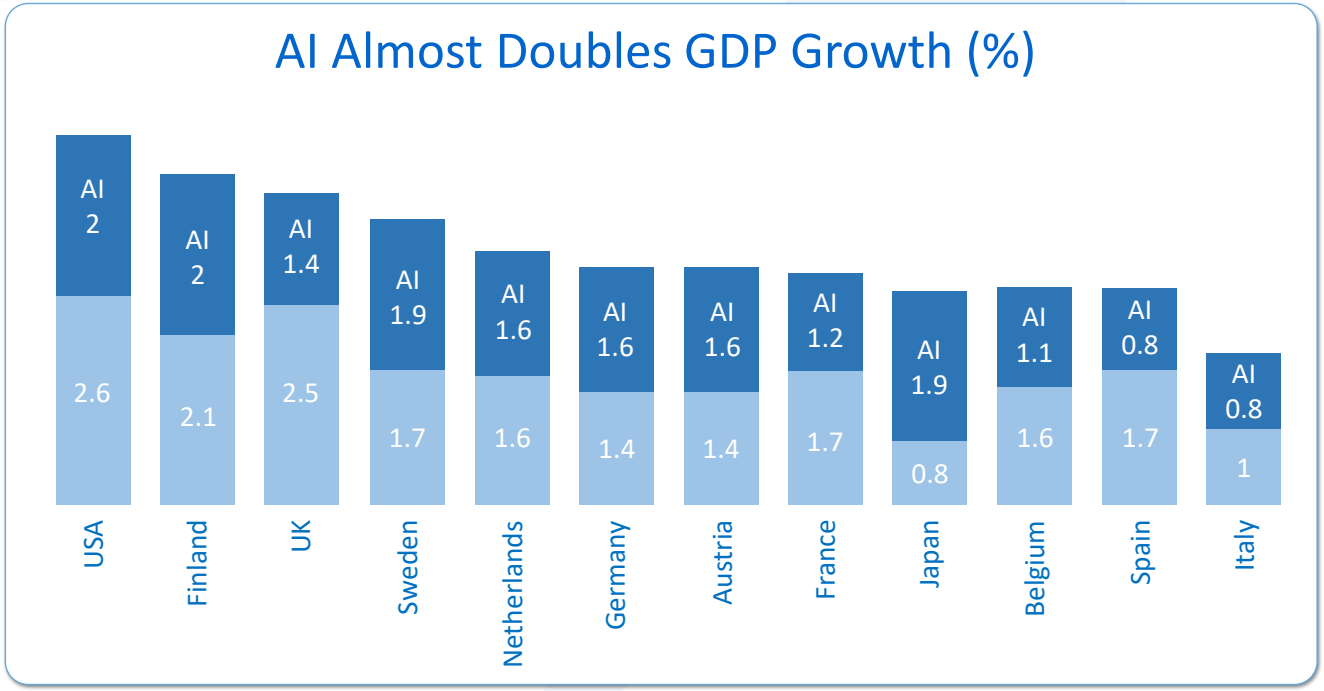
AI 40% productivity growth by 2035

PwC 2017

AI adds 14% to GDP by 2030

Putin 2017

“the leader in AI will rule the world”



Tachyum™

Tachyum is Critical for Datacenter Growth

3% of planet's electricity today

60% more than UK

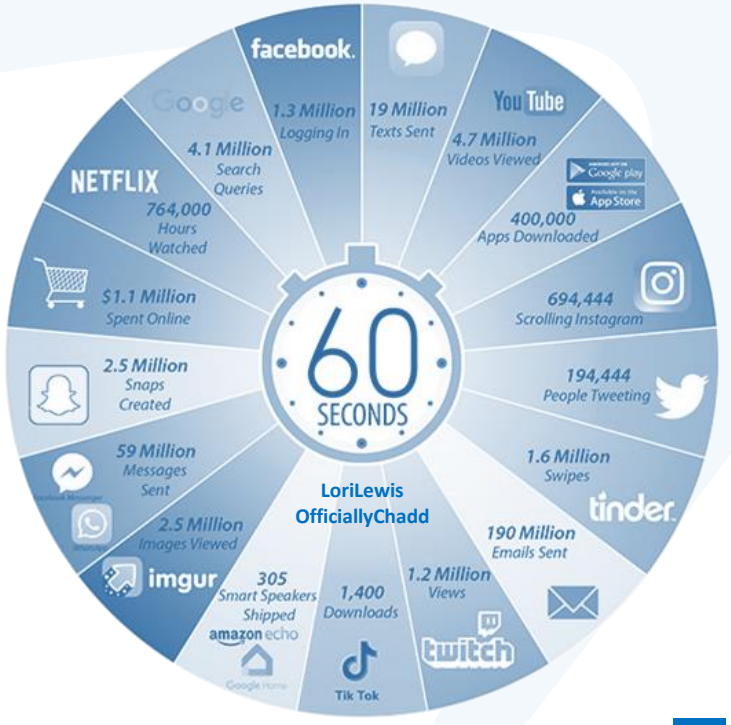
50% of the planet's energy by 2040

At 27% growth, it will be 33% by 2030

Largest CO_2 reduction impact

More than solar panels, windmills, …

10x lower power is needed to extend current datacenter growth



Tachyum™

Prodigy Universal Processor Reduces Carbon Footprint

Lowers Greenhouse Gasses

High Performance Low Power

- 3x higher performance
- 10x lower power

24/7 Server "On" Time

- Unified CPU, GPU & TPU
- Homogenous & composable

Prodigy's High Efficiency Helps to Keep Our Planet Green

AI Supercomputer: once-in-a-decades opportunity

EU AI is today in the hands of other countries, misaligned with EU interests

- Relying on other countries who are competitors and potential adversaries is not safe anymore
- Now, EU top priority is digital and technological sovereignty especially semiconductors

Slovakia needs to transition from cars and assembly to a knowledge-based economy

- EU consumes 30% of world compute resources, but has only 5% of world's resources
- Tachyum offers unique once-in-a-decade opportunity for Slovakia, and to fulfill EU critical needs
- Replace "brain drain" with "brain gain" by creating world class job opportunities in Slovakia

The world's fastest and most-powerful AI Supercomputer is built in Slovakia

- Unifying Europe by bridging language divide
- Fostering new high-tech industry
- Scientists from around the world will come to Slovakia to conduct ground-breaking research

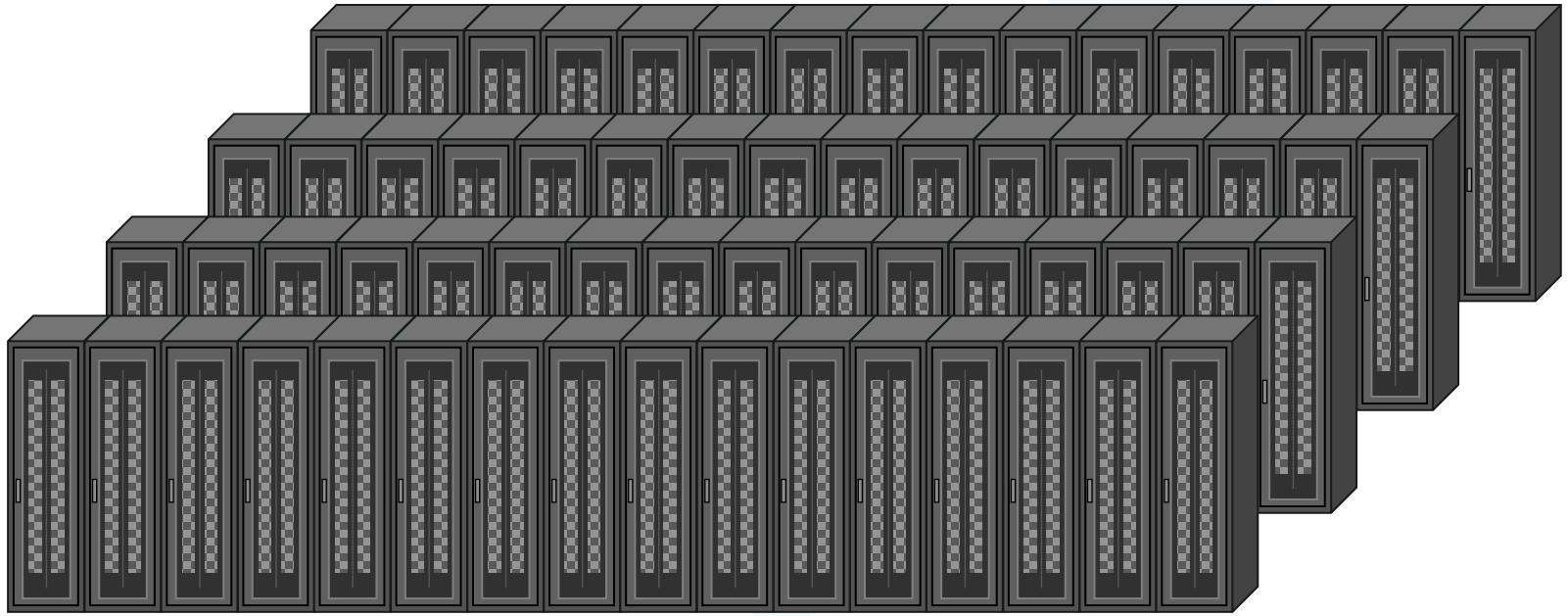
World's Fastest AI Supercomputer

64 Compute Racks

64 AI ExaFLOPs

Operational in 2022

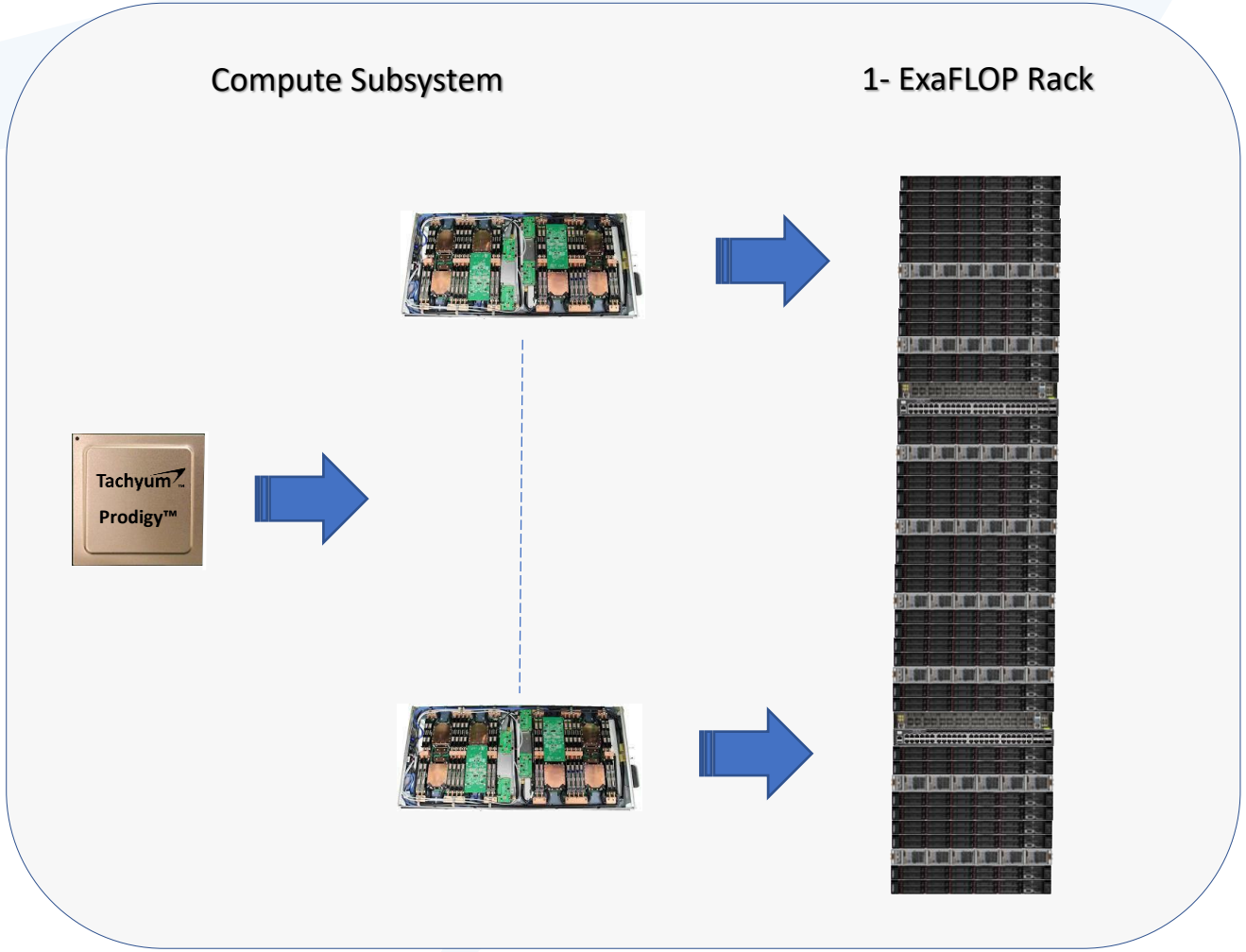
NSCC Slovakia Supercomputer



Prodigy-Powered

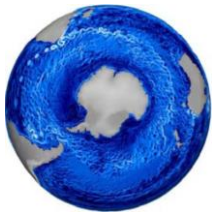
NSCC – SC Compute Rack

- High – Performance
  - 1 AI ExaFLOPs of Training and Inferencing per rack
- Prodigy T16128 Universal Processor
  - 128 64-bit cores
  - 2 vector units
  - Maximizes performance and efficiency
- Rack Configuration
  - 32 Prodigy 1U Compute Nodes
  - 8 sockets per compute node
  - 256 sockets per rack

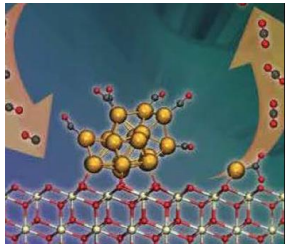


NSCC-SC and Prodigy Addressing the World's Problems

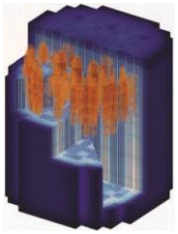
Climate change impact assessment



Biofuel catalyst design

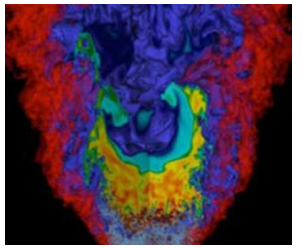


Next generation nuclear reactors

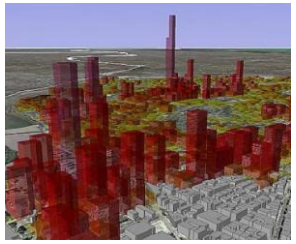


Improve efficiency and reduce cost

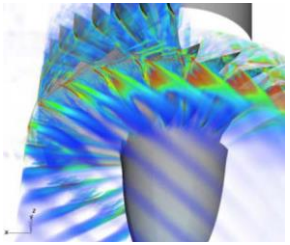
Design of low-emission engine



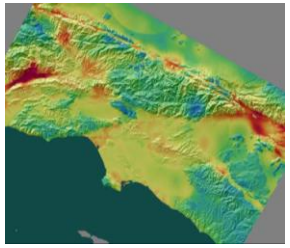
Energy and water nexus



Scaling carbon capture designs

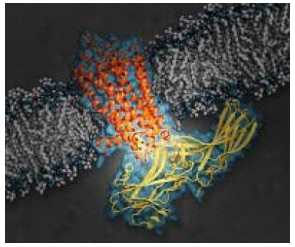


Modeling and risk assessment



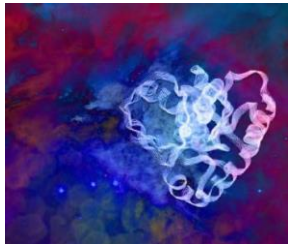
Renewable energy planning

Protein structure and dynamics



Process of additive manufacturing

Drugs and vaccines discovery



Tachyum™

6x Faster Drugs and Vaccine Discovery

Discovery 1.5 years
Screen million molecules

Optimization 3 years
Design & test 1000 molecules

Trials 1.5 years
On animals

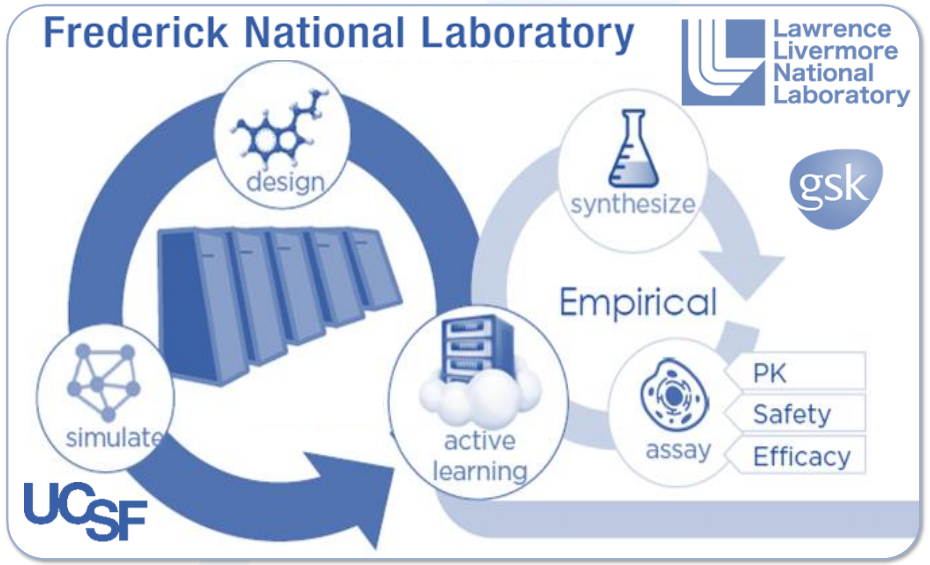
Clinical Trial

6 years → 1 year

Tachyum
Low Cost HPC
Available for ALL

Patient's data

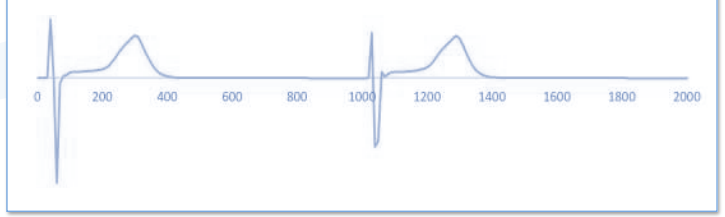
Personalized Medicine



Tachyum

25,000 Lives To Save Per Year

ECG simulation for healthy heart



Arrythmia simulation after drug Sotalol

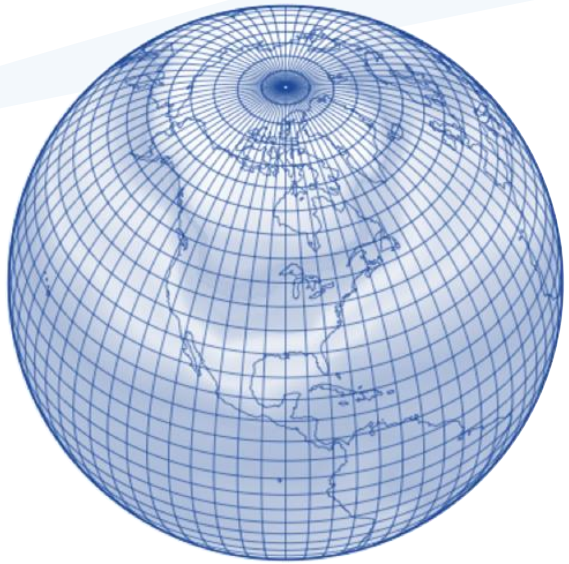
$100,000

Tachyum $10,000

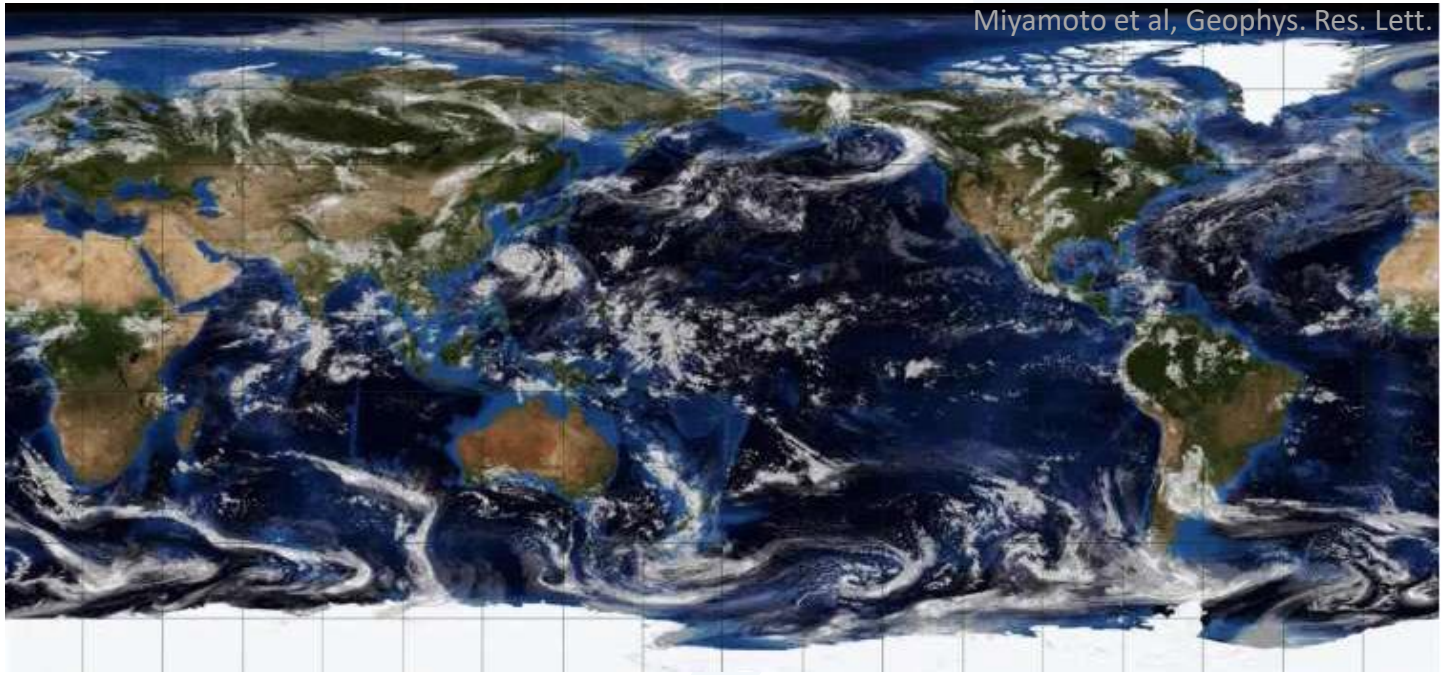
Tachyum Is Democratizing HPC

Key to Understand Climate Change

Existing models not accurate

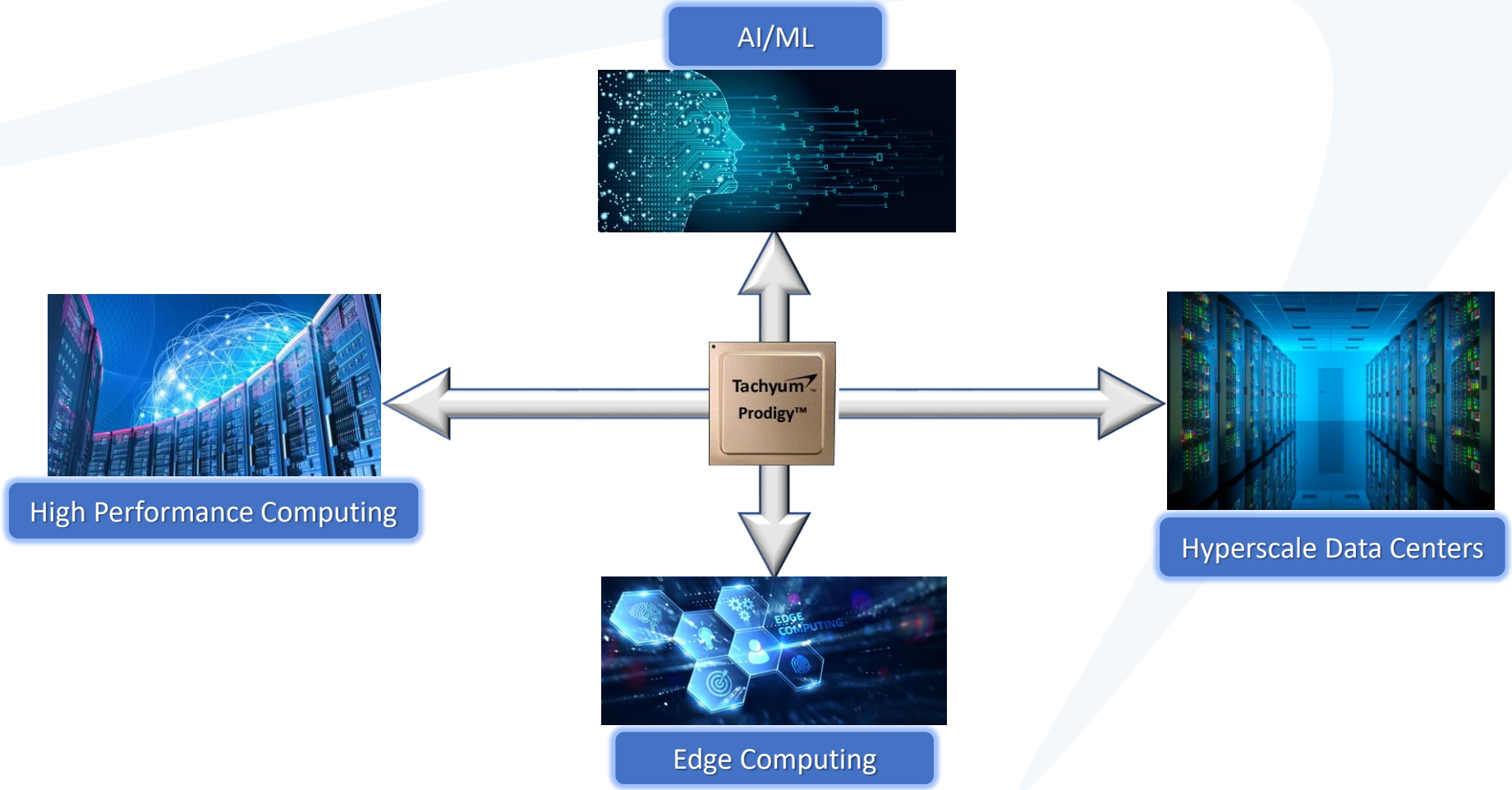


Tachyum enables <1 km resolution to accurately model clouds

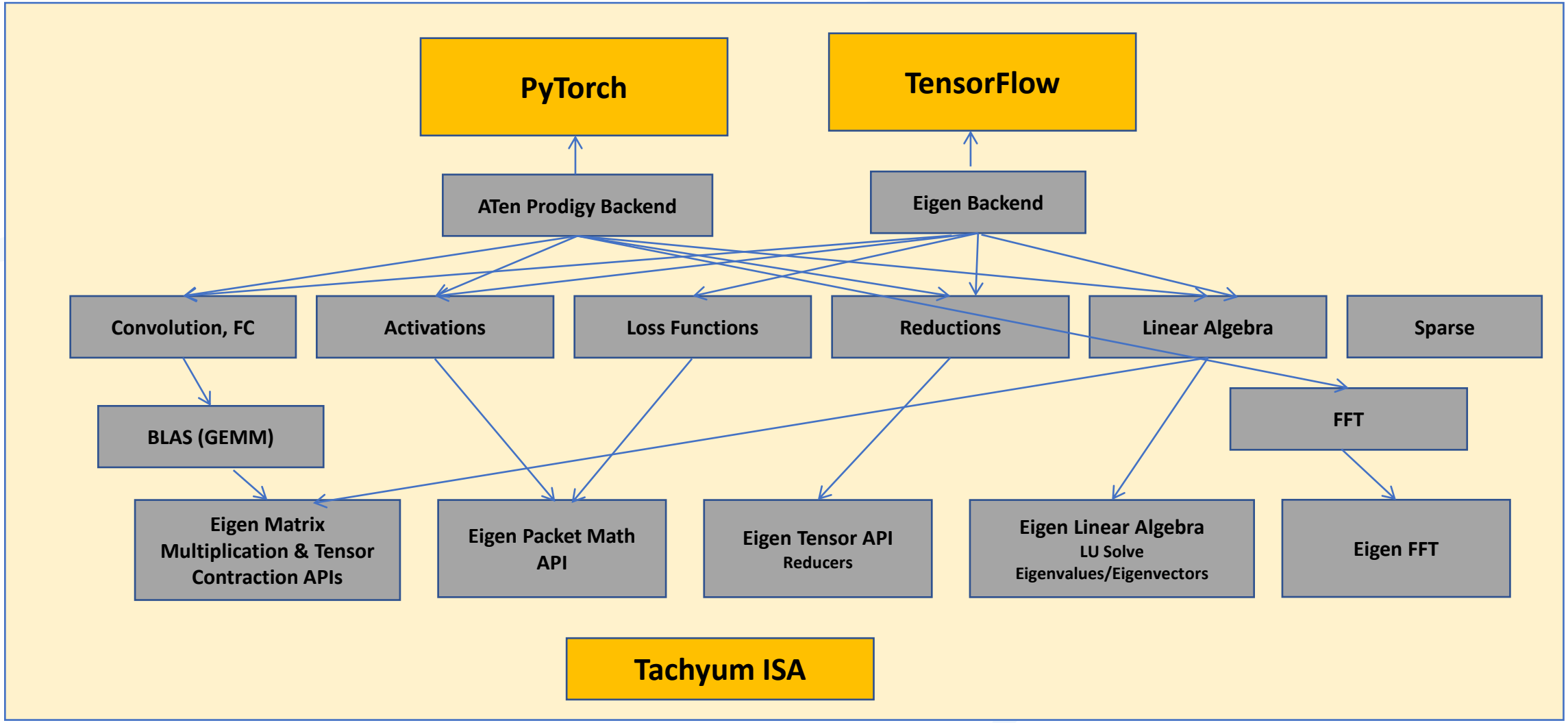


Prodigy Target Platforms

Prodigy has multiple SKUs that align with a wide range of markets, applications, and workloads

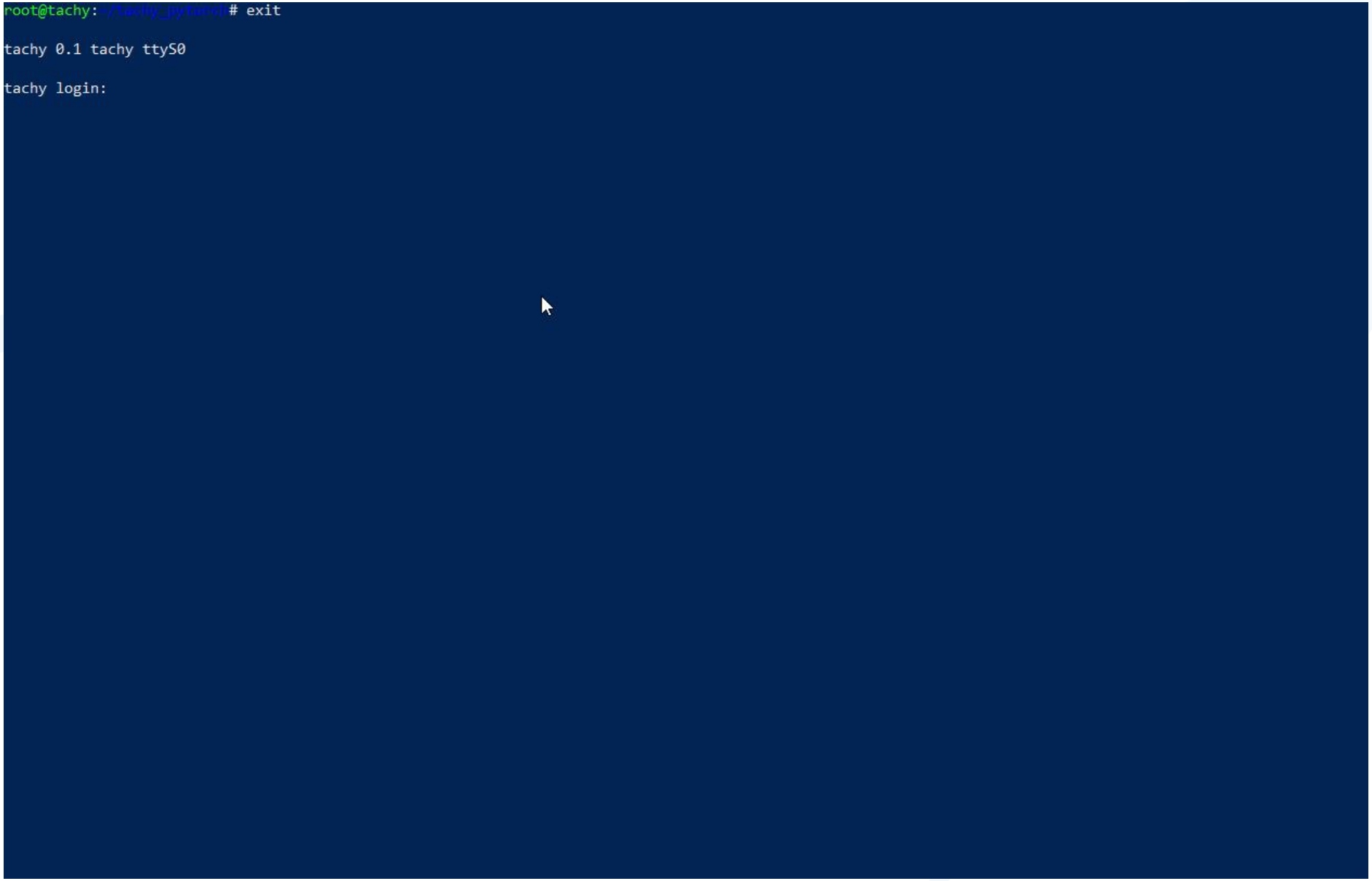


Native support for AI frameworks



- Activation & Loss Function – optimized utilizing Tachyum vector instructions in standard and low precision modes
- Dense GEMM library implemented utilizing Tachyum matrix instructions in standard and low precision modes, stochastic rounding, single and multithreaded
- Custom Sparse GEMM library implemented utilizing Tachyum vector and matrix instructions
- Convolutional and Dense operators implemented utilizing Tachyum matrix instructions in standard and low precision modes, including depthwise separable and pointwise convolutions
- Circulant and Butterfly Convolutional and Dense operators implemented utilizing custom FFT for matrix multiplication

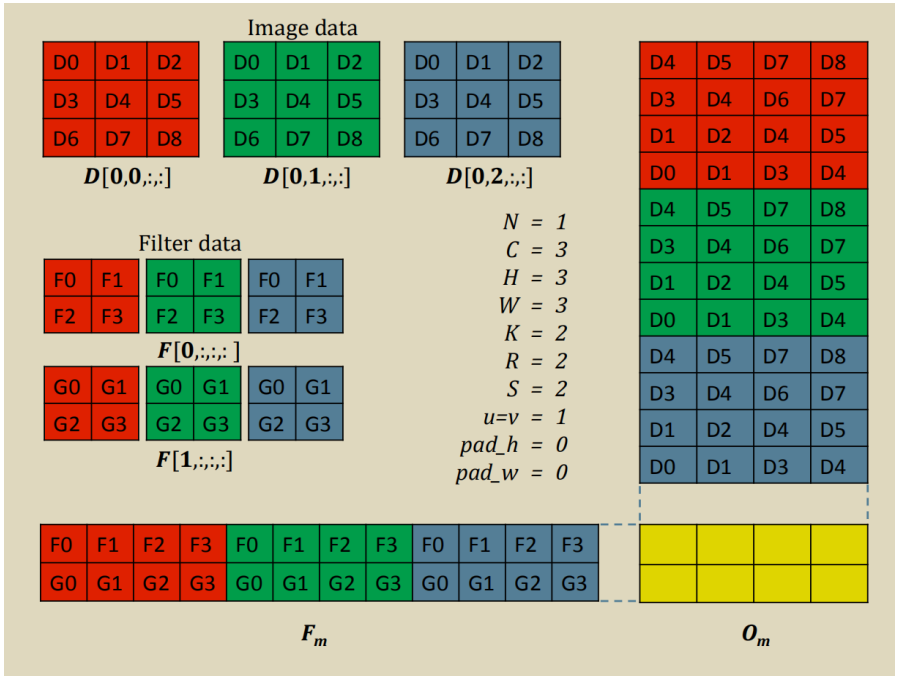
Native support for AI frameworks



Convolutions Algorithms

- **Direct Convolution**

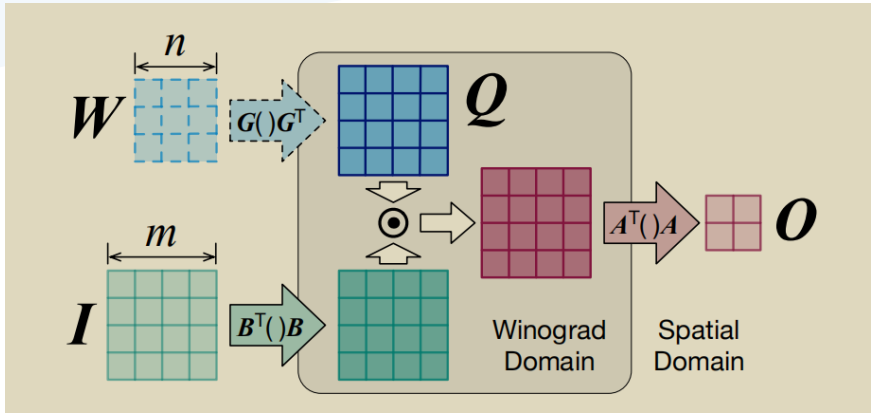
- **GEMM (BLAS) based convolution**
  - One approach is to lower the convolutions into a matrix multiplication. This can be done by reshaping the filter tensor F into a matrix Fm with dimensions K×CRS, and gathering a data matrix by duplicating the original input data into a matrix Dm with dimensions CRS×NPQ. The computation can then be performed with a single matrix multiply to form an output matrix Om with dimension K×NPQ.



Tachyum™

Convolutions Algorithms

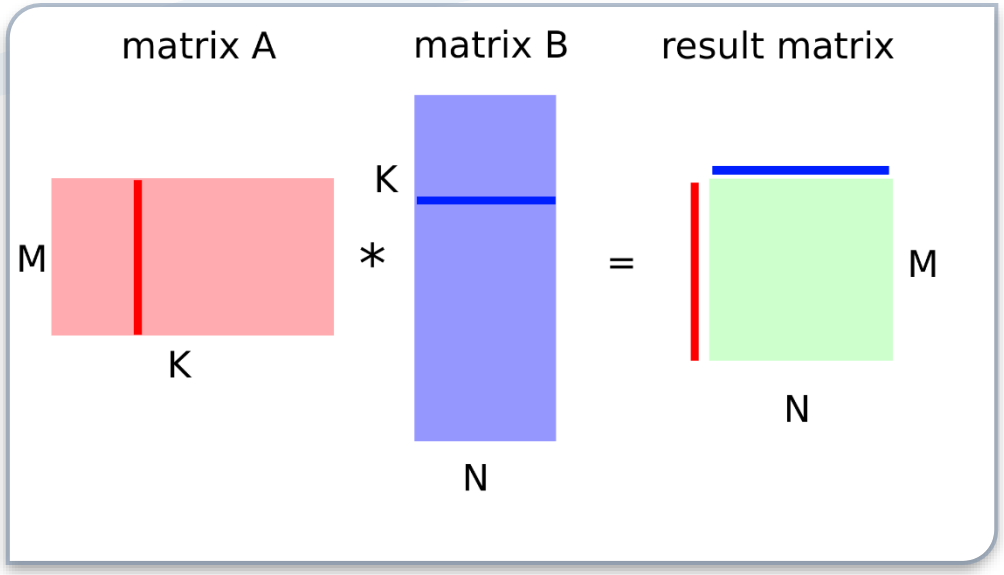
- FFT Convolution
- Winograd Convolution



Revolutionizing AI training with high performance Prodigy Matrix Instructions with reduced precision

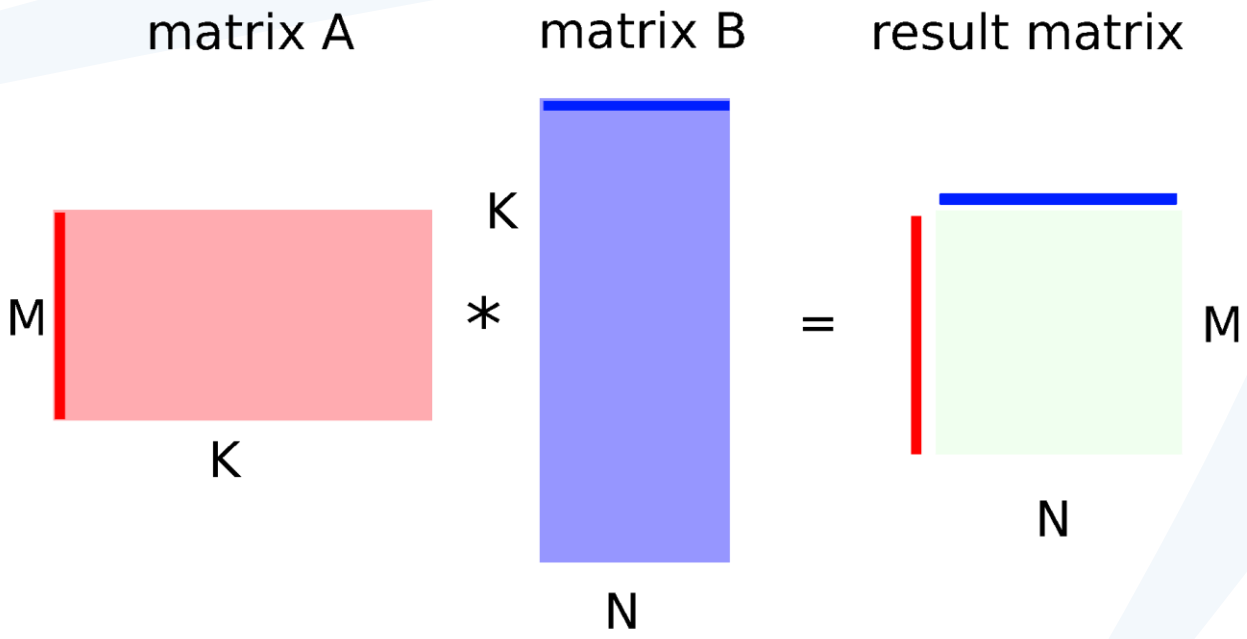
- Prodigy CPU addresses continuing trends in AI models, explosion in complexity as demanded by more complex NLP models and more accurate conversational AI.
- NLP transformer models are hundreds of times larger and more complex than image classification models like ResNet-50. Training these massive models in FP32 precision can take days or even weeks.
- Matrix multiplication in Prodigy CPU provide an order-of-magnitude higher performance with reduced precisions substantially reducing training-to-convergence times while maintaining accuracy.

Matrix multiplication using VECTOR instructions



1. tile matrix A to K vectors of $M \times 1$ elements
2. tile matrix B to K vectors of $1 \times N$ elements
3. K-times perform element-wise vector multiplications
4. **in one step N MACs operations are performed**

Matrix multiplication using vector instructions



Matrix multiplication using vector instructions

- **Total** $K \times M$ vector instructions calls

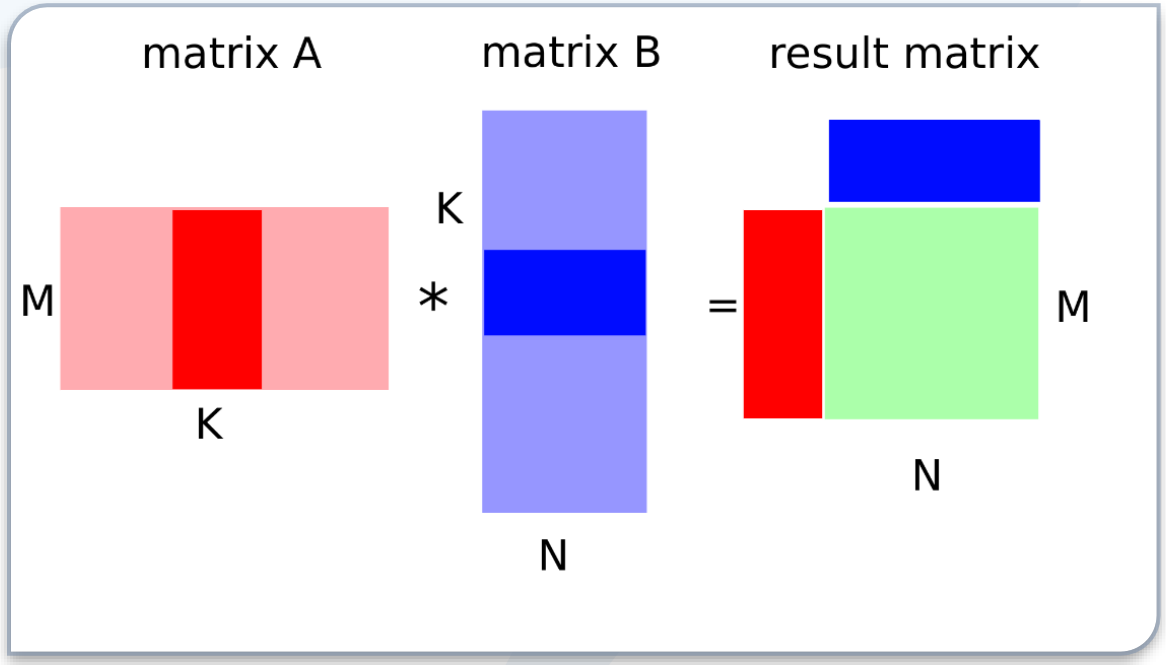
```
for (int k = 0; k < Kc; k++)
{
    for (int j = 0; j < Mr; j++)                //rows in matrix A
        for (int i = 0; i < Nr; i+= Nk)         //cols in Matrix B
            for (int ii = 0; ii < Nk; ii++)     //cols in Matrix B
                y[j*Mr + i + ii]+= a[j]*b[i + ii];

    a+= Mr; //next row
    b+= Nr; //next row
}
```

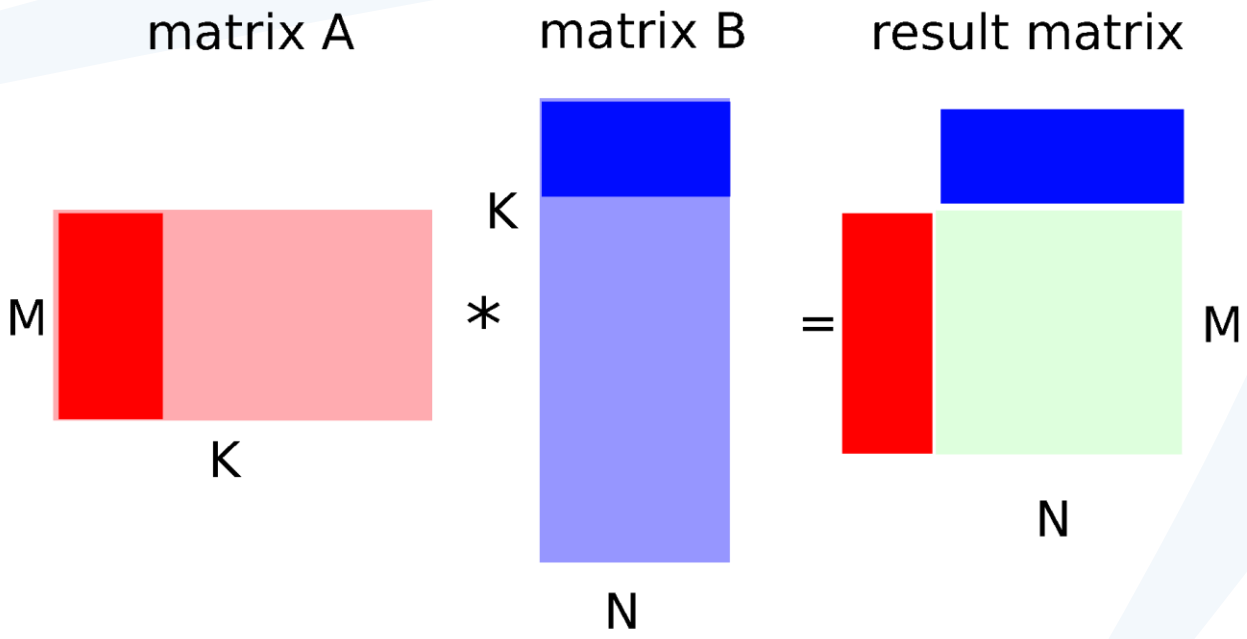
- note, loop **for ii** is performed in vector instruction

Matrix multiplication using MATRIX instructions

- tile matrix A to K/K_r matrices of $M \times K_r$ elements
- tile matrix B to K/K_r matrices of $K_r \times N$ elements
- K/K_r -times perform small matrix multiplications
- **in one step $M_r \times N_r \times K_r$ MACs operations are performed**



Matrix multiplication using matrix instructions



Matrix multiplication using matrix instructions

- no loops required in code - just call MMADD

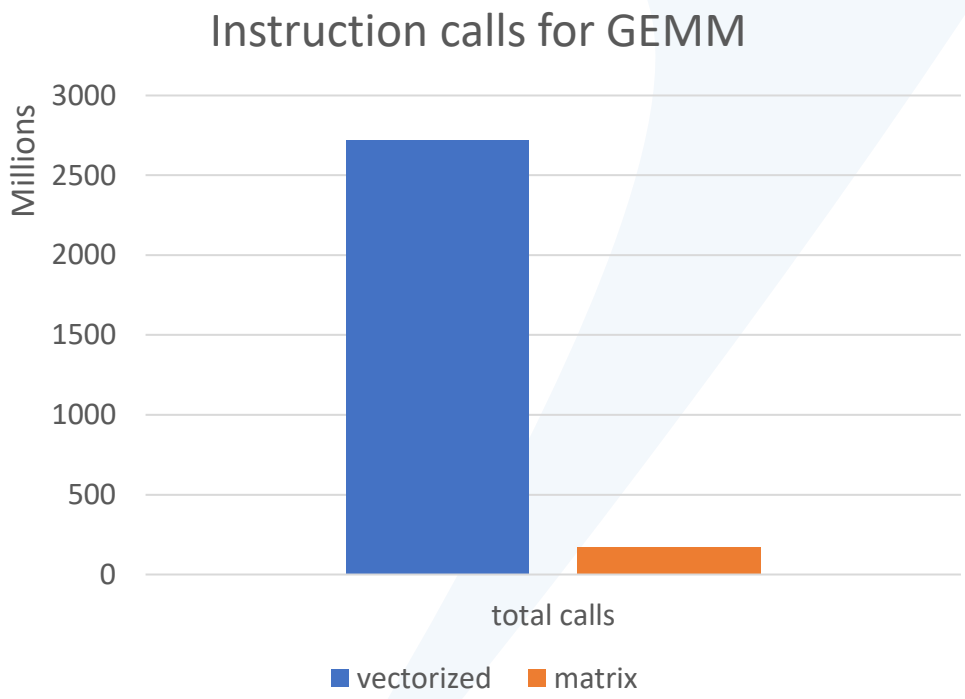
```
//call float MMADD
__asm__ volatile (
"vld w16, p0, [%0]\n\t"
"vld w8, p0, [%1]\n\t"
"vld w0, p0, [%2]\n\t"
"mmaddq f0, p0, f8, f16, f0\n\t"

"vst [%2], p0, w0\n\t"
:
: "r" (b),
  "r" (a),
  "r" (y)
: "v0", "v8", "v16"
);
```

GEMM COMPUTATION REDUCTION USING MATRIX INSTRUCTIONS

- **MACs instructions calls for vectorized version:**
- fadd4 1,439,257,600 calls
- fmul4 1,280,659,456 calls
- **total 2,719,917,056 calls**

- **MMADDs instructions calls for matrix version:**
- fadd4 164,189,184 calls
- mmaddqfff 9,961,472 calls
- **total 174,150,656 calls**



this is **15x less** instructions calls

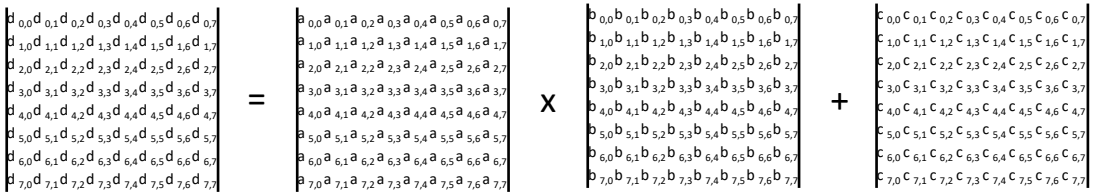
Vector and Matrix Execution

Floating-Point/ Integer Units	• IEEE Double, Single, and Half-Precision FPU • AI 8-bit Floating-Point Data Type • 2 x 1024-bit Multiply-Add Vector/Matrix Units • 2 x 1024-bit ALUs Supporting 8, 16, and 32-bit Integers with No/Signed/Unsigned Saturation
Vector and Matrix Operations	• Matrix Operations: 4x Less Power than competition • 8-bit Int/FP: 16 x 16 • 16-bit Int/FP: 8 x 8 • FP64, FP32: 4 x 4 • 8 x 8 Matrix Multiply-Add = 1024 Flops ○ Uses 6 Source and 2 Destination Registers • Ability to Increase Performance 2x in the Future
Maximum Issue Rate per Clock	• 2 x 1024-bit Multiply-Add • 2 x 1024-bit Integer Instructions • 1 Load, 1 Load/Store, 1 Store

P16128 Total FLOPS by Data Type

Data Type	FLOPS/ Core	Total FLOPS – P16128 (128 cores x 4 GHz x FLOPS/Core)
Double Precision	2 x 32 FLOPS = 64	32 TeraFLOPS
Single Precision	2 x 128 FLOPS = 128	128 TeraFLOPS
Half Precision	2 x 512 FLOPS = 1024	512 TeraFLOPS
FP8	2 x 2048 = 4096	4 PetaFLOPS

Prodigy Supports 16x16, 8x8, and 4x4 Matrix Operations



Tachyum™

Quantization

- Quantization is an effective method for reducing memory footprint and inference time of Neural Networks.

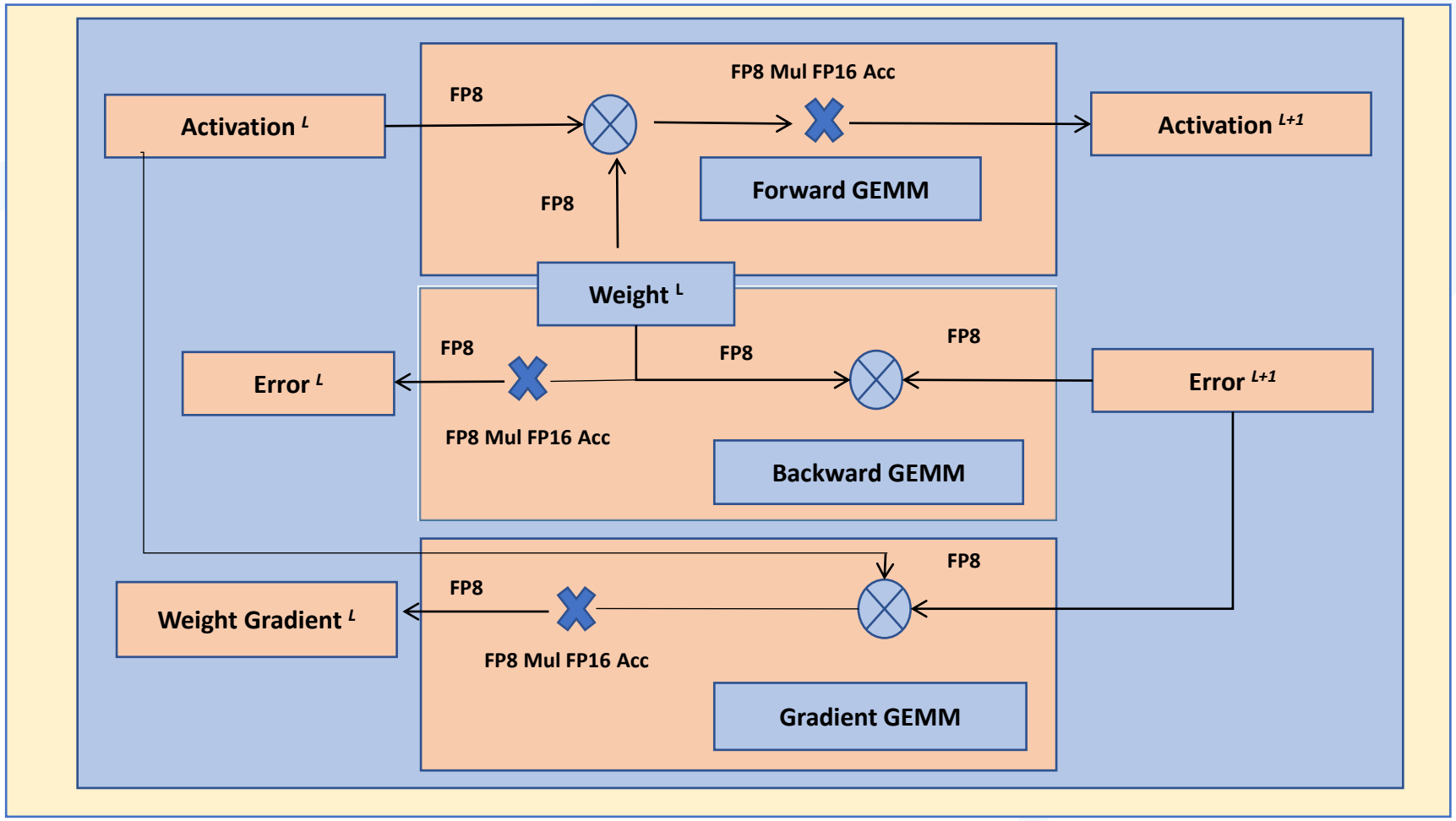
- **Quantization Aware Training**
  - **Mixed Precision Training**
    - master copy of weights and gradient momentum in BF16
    - Loss and per-layer gradient scaling
  - Supported Lows Precision Data Types: BF16, Float8, Float4

- **Post Training Quantization Inference**
  - Supported low precision data types: INT8, Float8, Float4

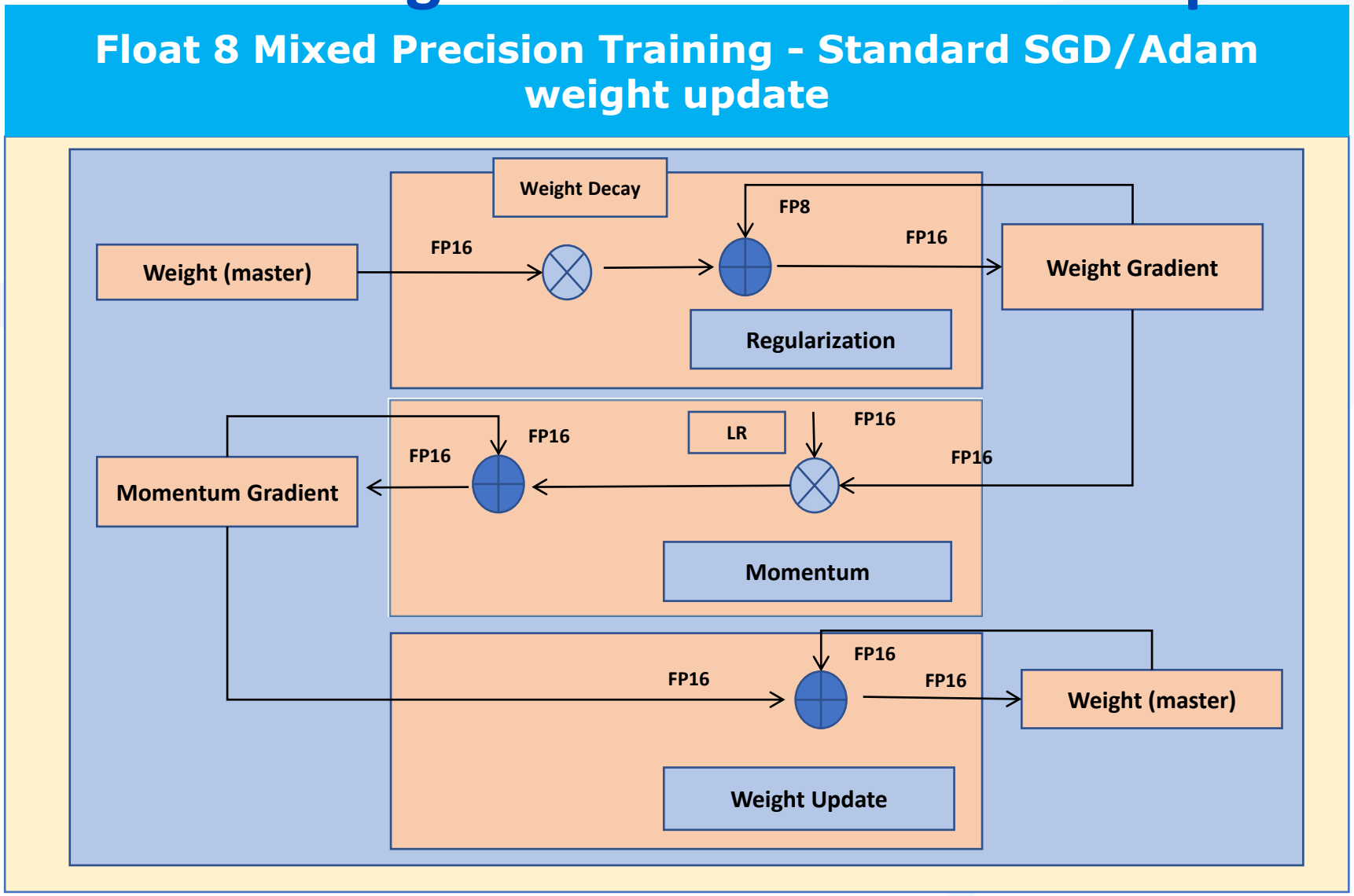
- Ultra-low precision quantization could lead to significant degradation in model accuracy. A promising method to address this is to perform mixed-precision quantization, where more sensitive layers are kept at higher precision. However, the search space for a mixed-precision quantization is exponential in the number of layers.

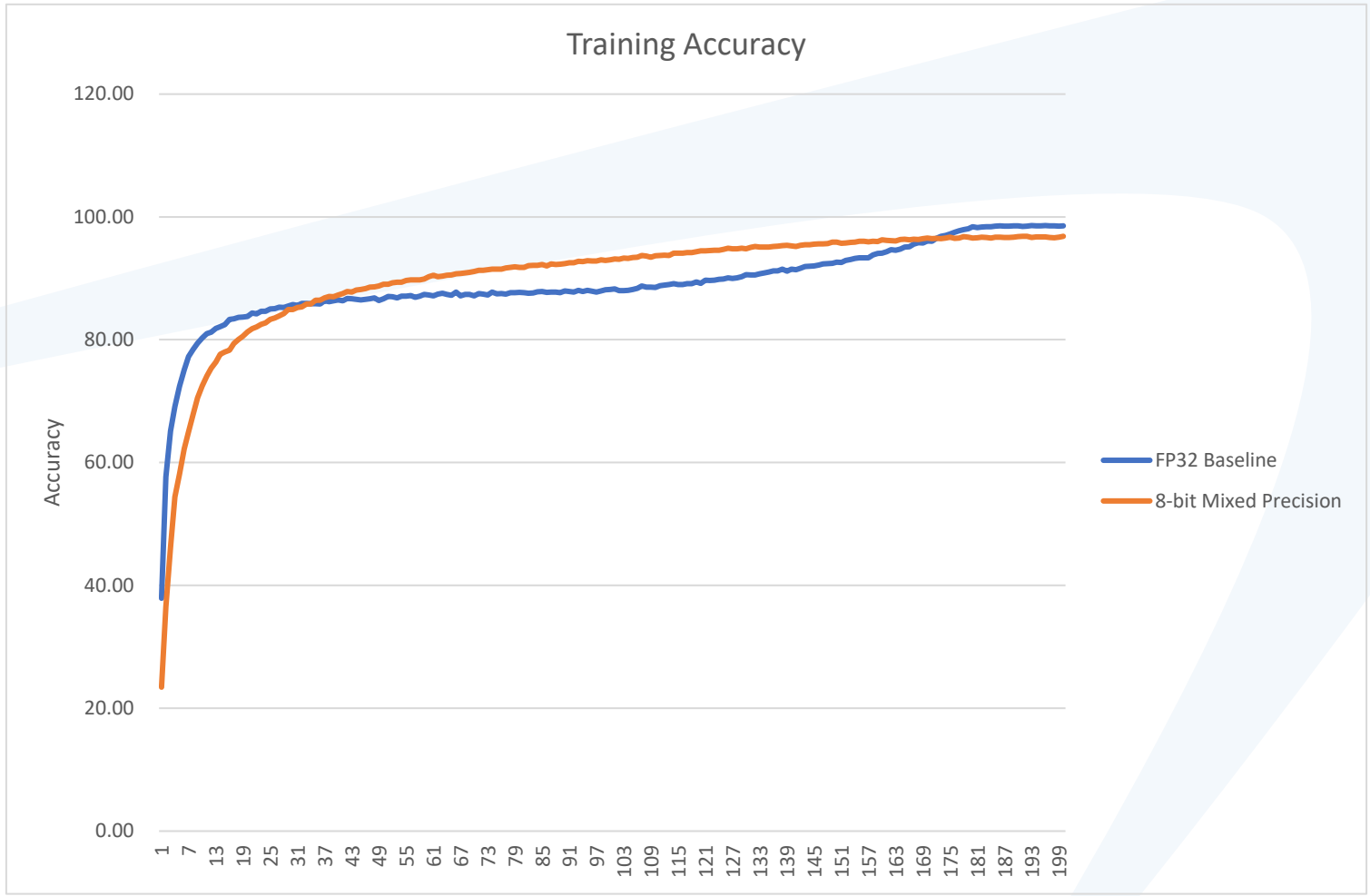
- Hessian based framework, with the aim of reducing this exponential search space by using second-order information. Hessian based framework provides a method for automatic bit precision selection of different layers without any manual intervention by analyzing sensitivity of loss surface ith respect to bit precision of different layers to bit precision

Float 8 Mixed Precision Training – GEMM operations during forward and backward passes



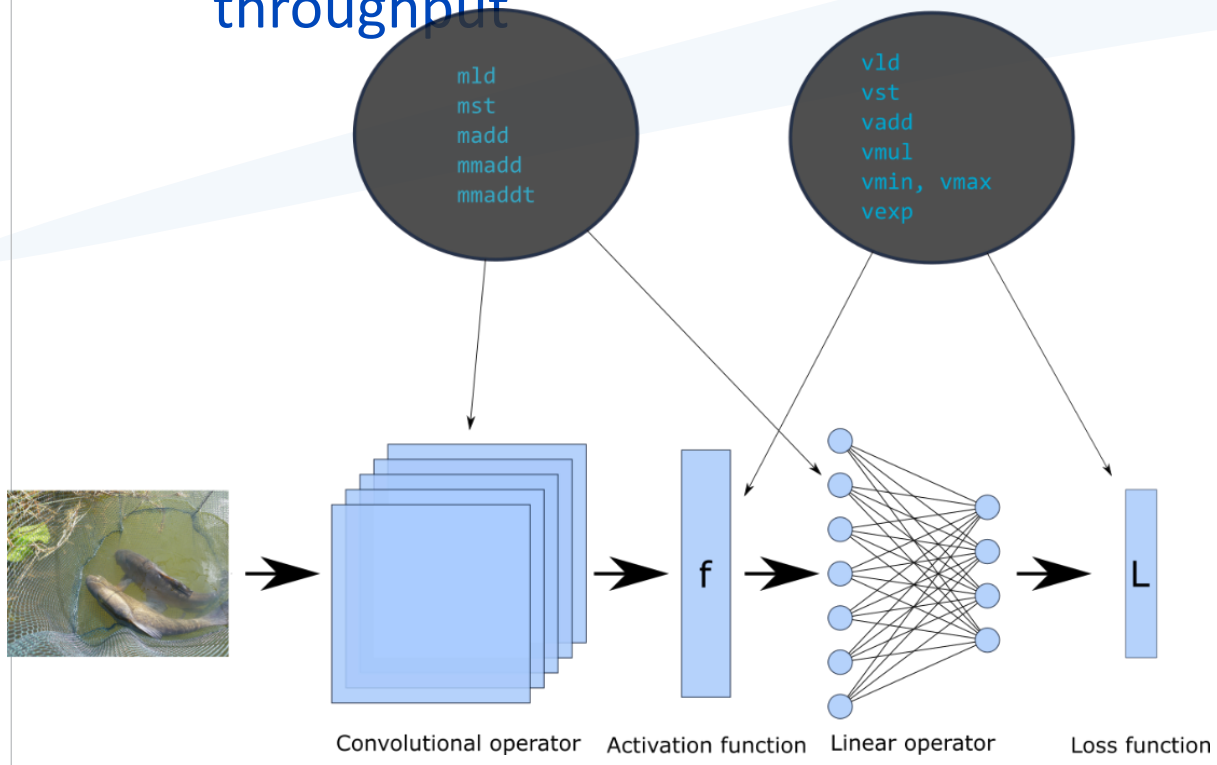
Float 8 Mixed Precision Training – GEMM operations during forward and backward passes




Training Accuracy
120.00
100.00
80.00
60.00
40.00
20.00
0.00
Accuracy
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145
151
157
163
169
175
181
187
193
199
FP32 Baseline
8-bit Mixed Precision

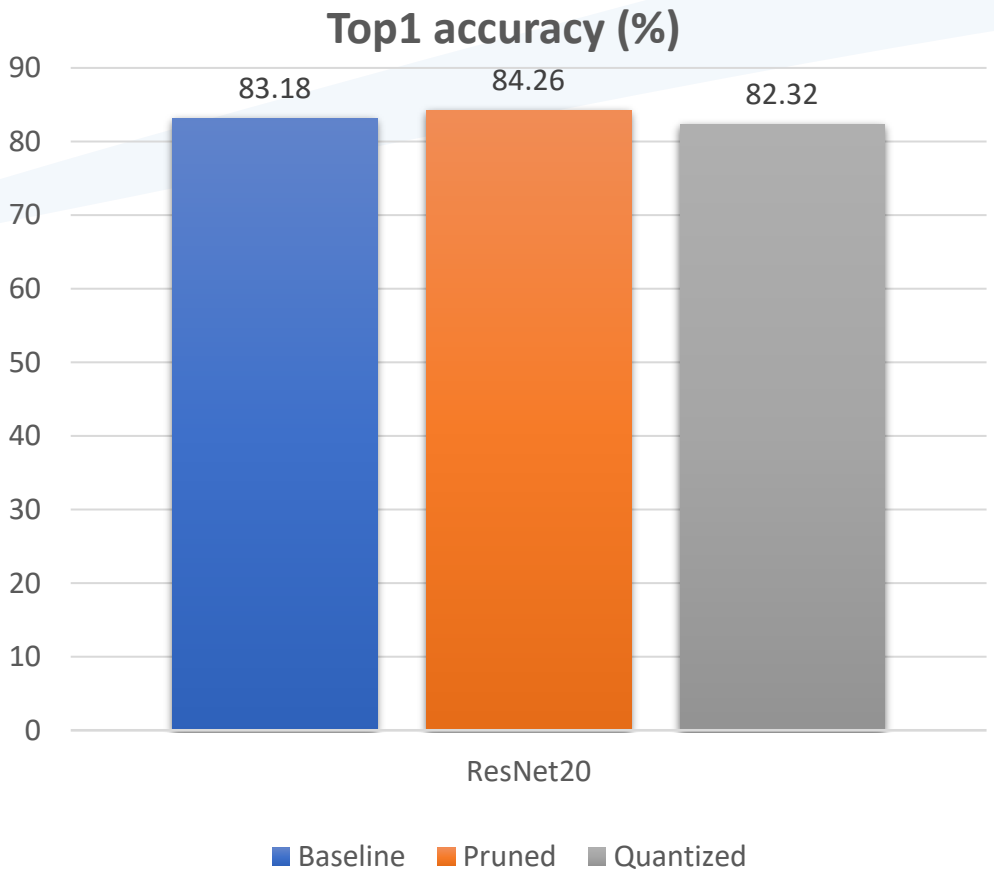
Efficient AI inference

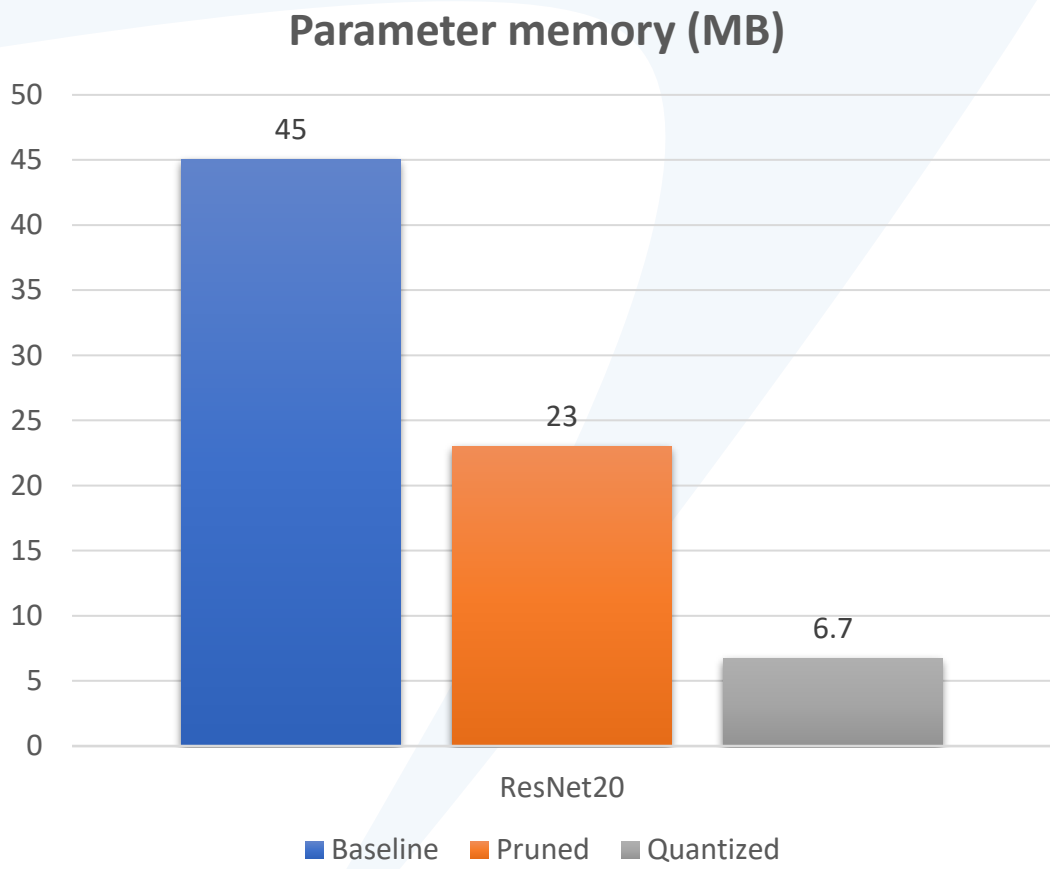
- Compressed and quantized models exploiting the Prodigy low precision data types for vector instructions and matrix multiplication and compressed matrix multipliers while still maintaining high accuracy, low latency and high throughput



Tachyum™

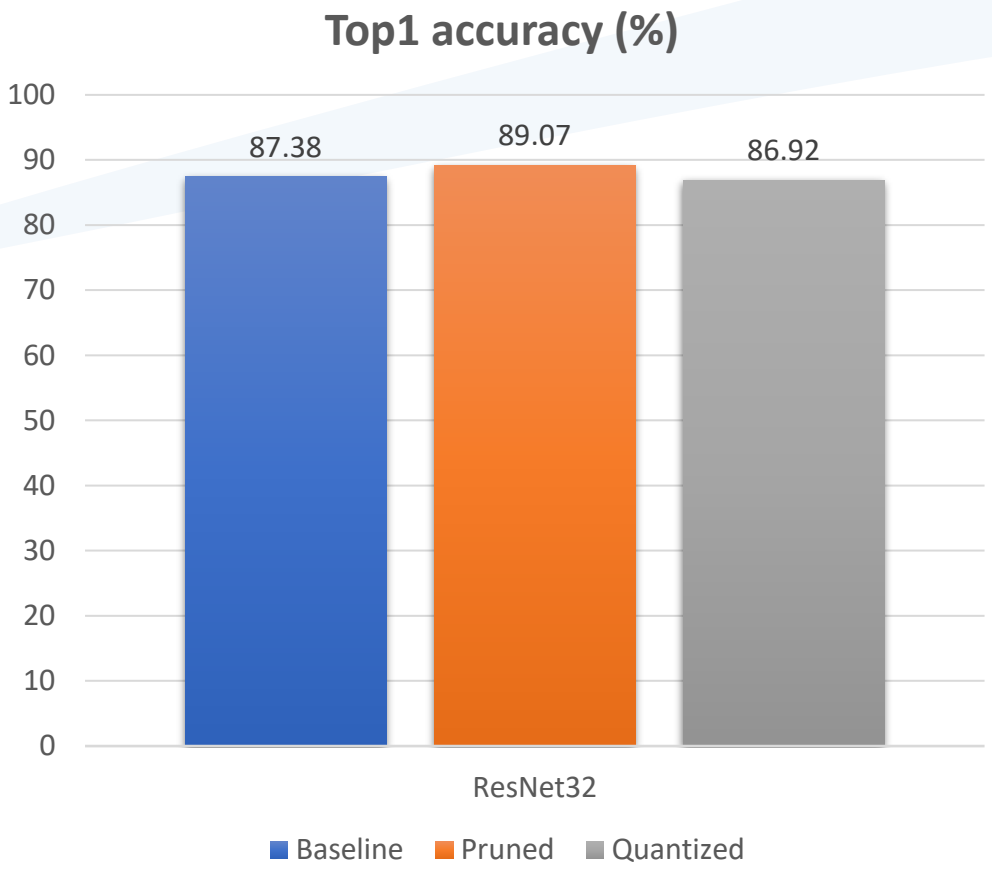
ResNet20 INT4W/INT8A quantization

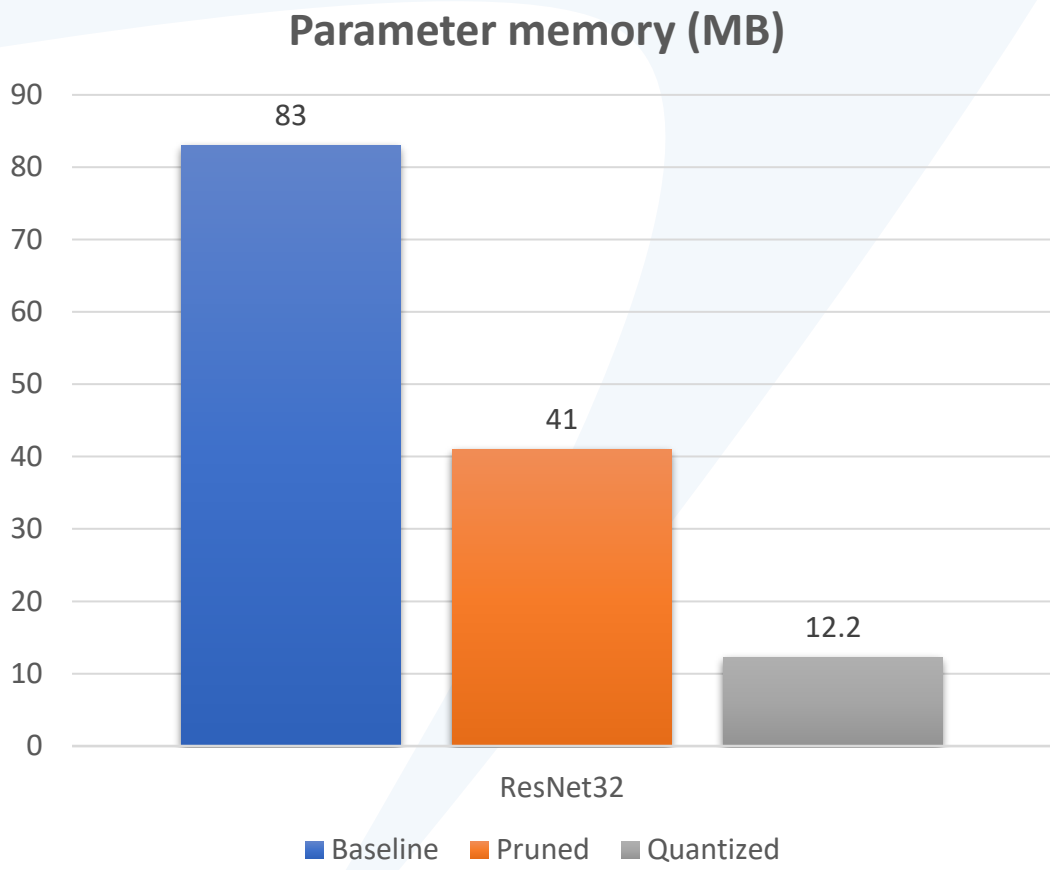


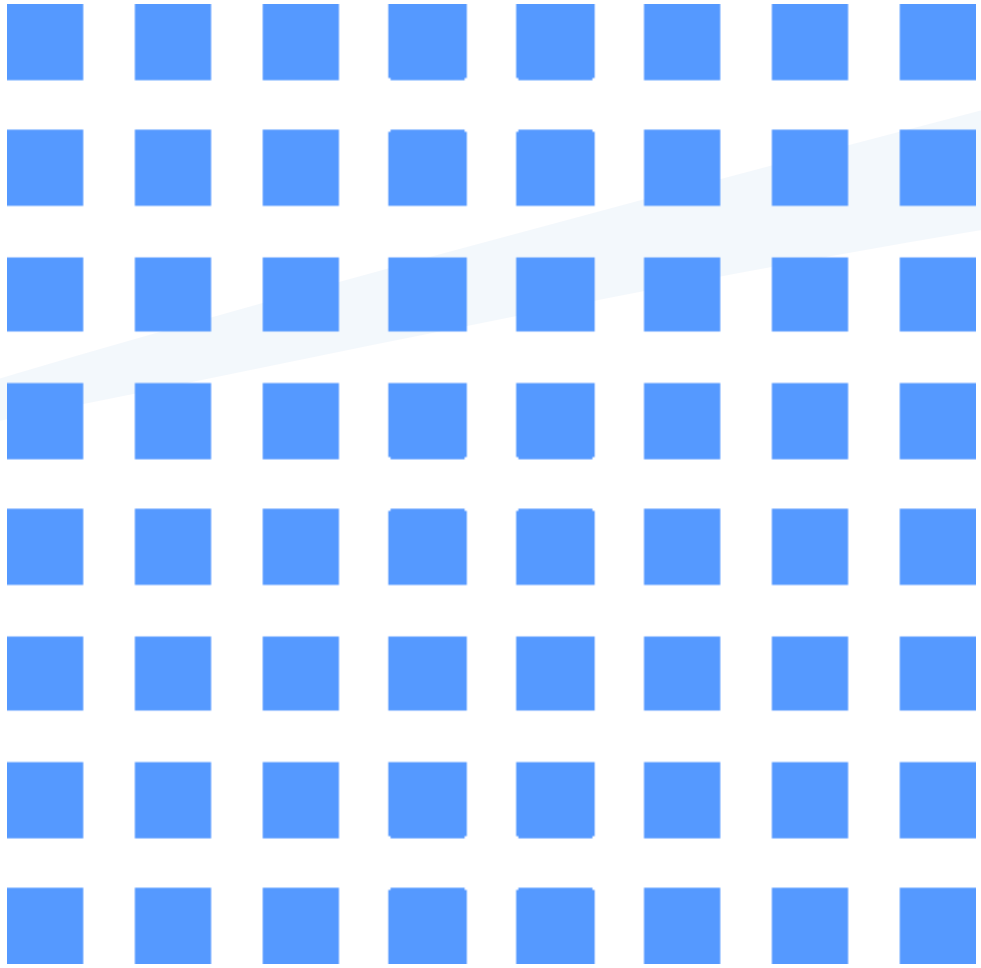


Tachyum™

ResNet32 INT4W/INT8W quantization



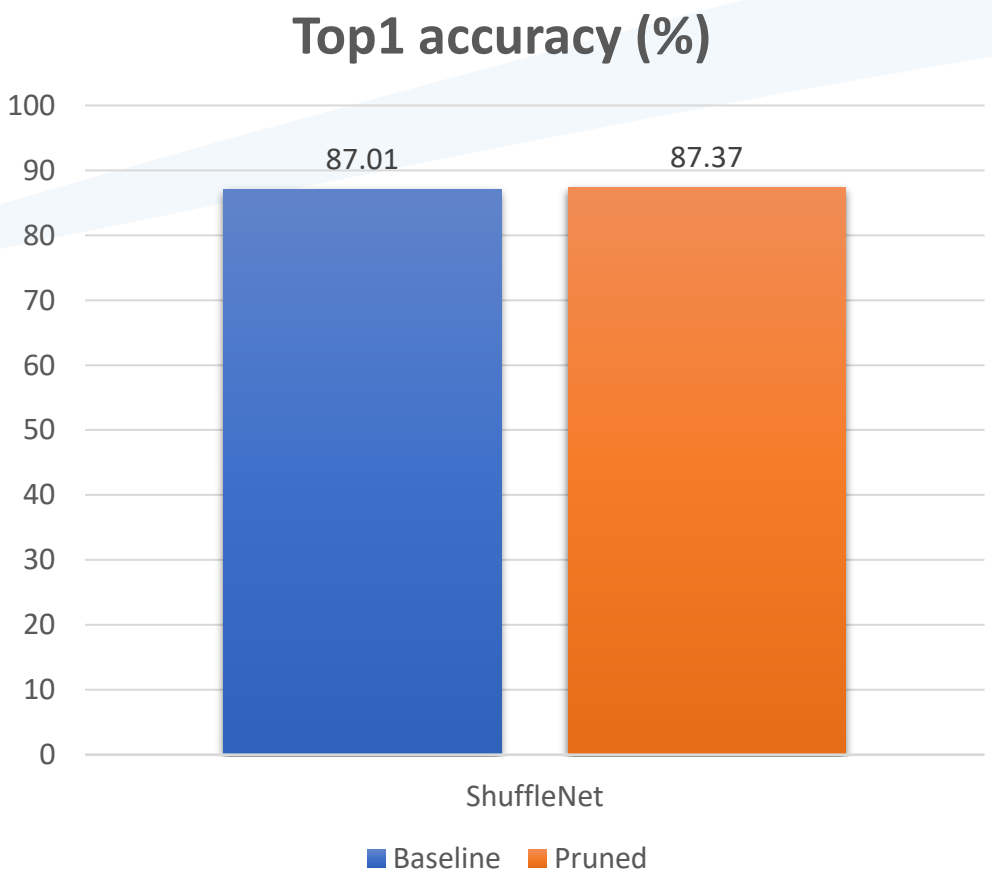


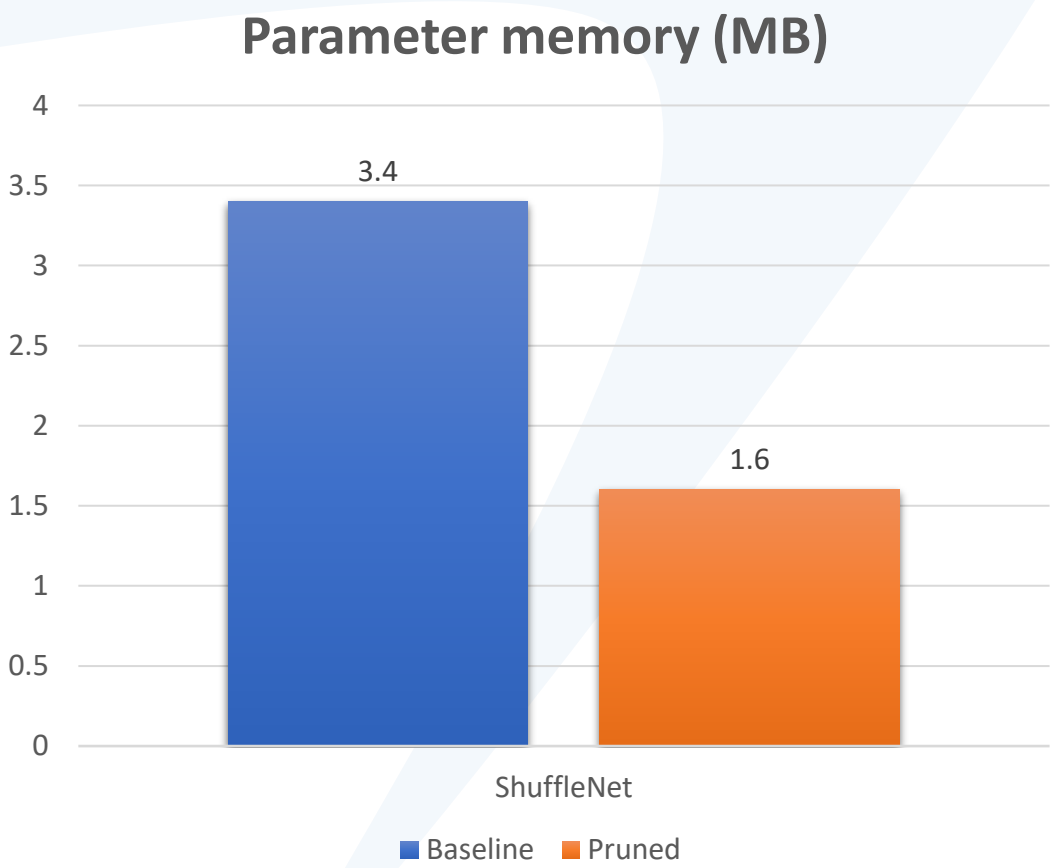


Compression, Pruning

- Magnitude based weight pruning – N:M block pruning
- Lottery Tickets – pruning weights and retrain
- Support for sparse matrix operations (block sparsity) optimized for compressed networks/models thus reducing memory and computation requirements
- Specific instructions for efficient storing and loading sparse matrices and for sparse structured matrix multiplication

ShuffleNet pruning test



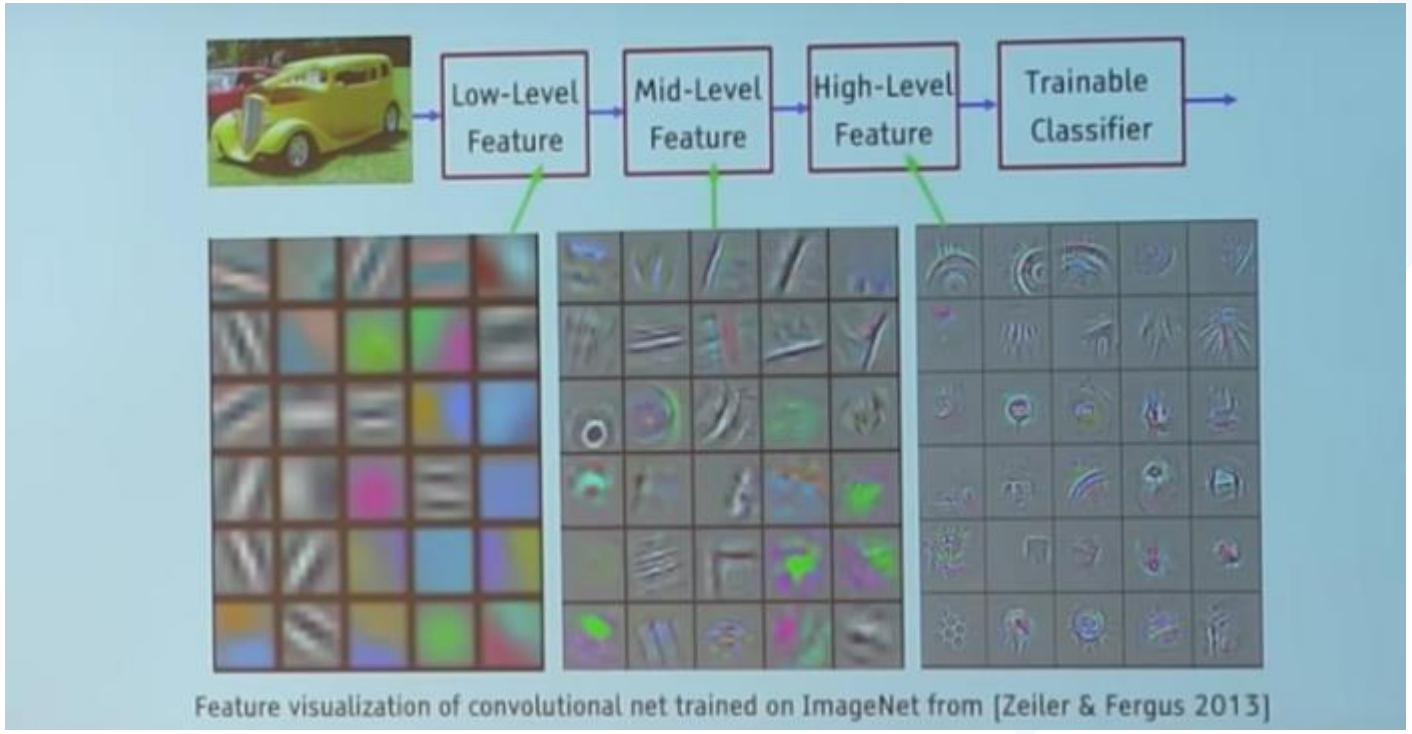


Computer Vision

- **Image Classification**
  - Resnet, MobileNet, ShuffleNet, DenseNet
  - Vision Transformer – ViT, DeIT
- **Object detection**
  - YOLO, SSD,
- **Semantic segmentation**
  - the goal of semantic image segmentation is to label each pixel of an image with a corresponding class of what is being represented. Because we're predicting for every pixel in the image, this task is commonly referred to as dense
  - MaskRCNN
- **Supervised pre-training**
- **Transfer learning**
- **Fine-tuning**

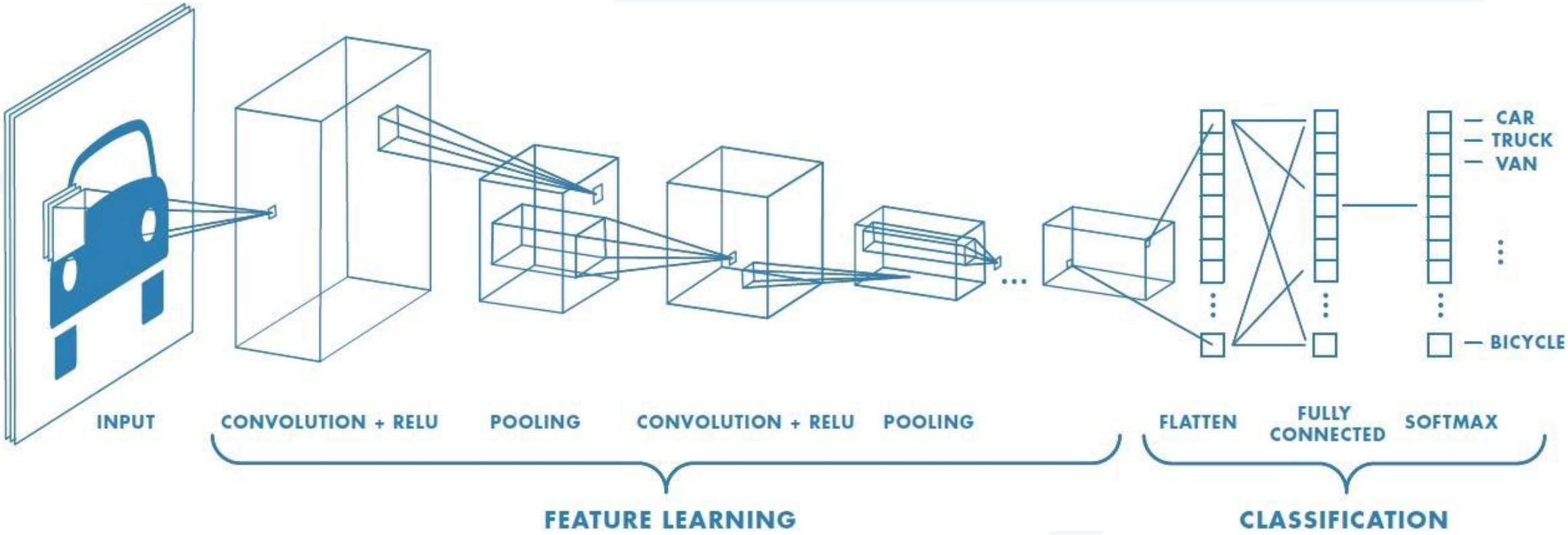
Computer Vision – Convolutional Neural Networks

- The role of the Convolutional Network is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction
- Conventionally, the first Convolutional Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc.
- With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images



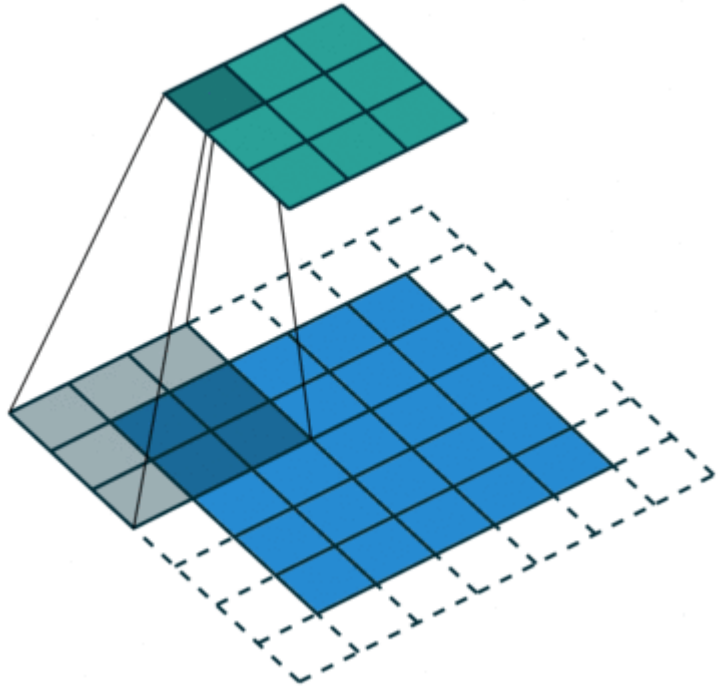
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

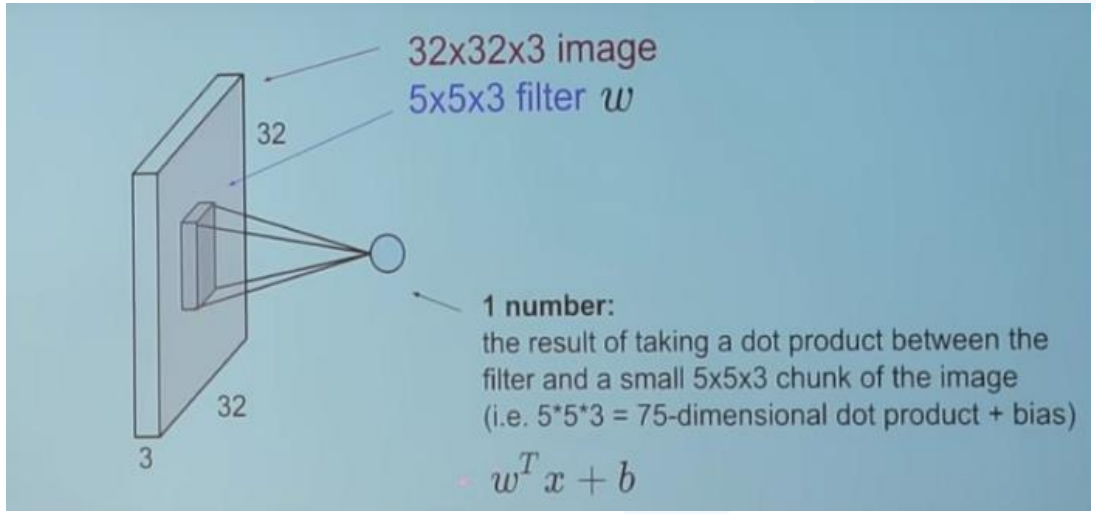
Computer Vision – Convolutional Neural Networks



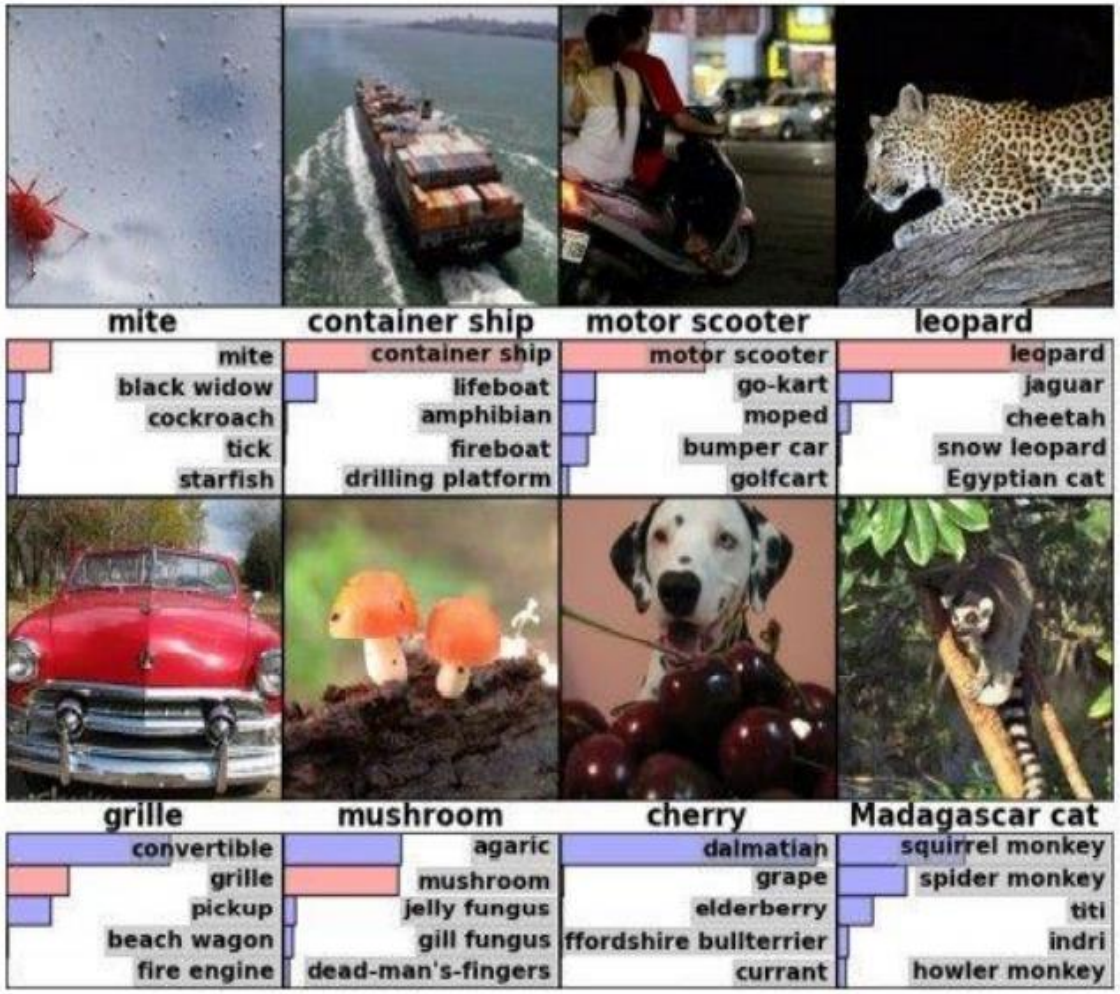
Computer Vision – Convolutional Neural Networks

- Convolutional Operation



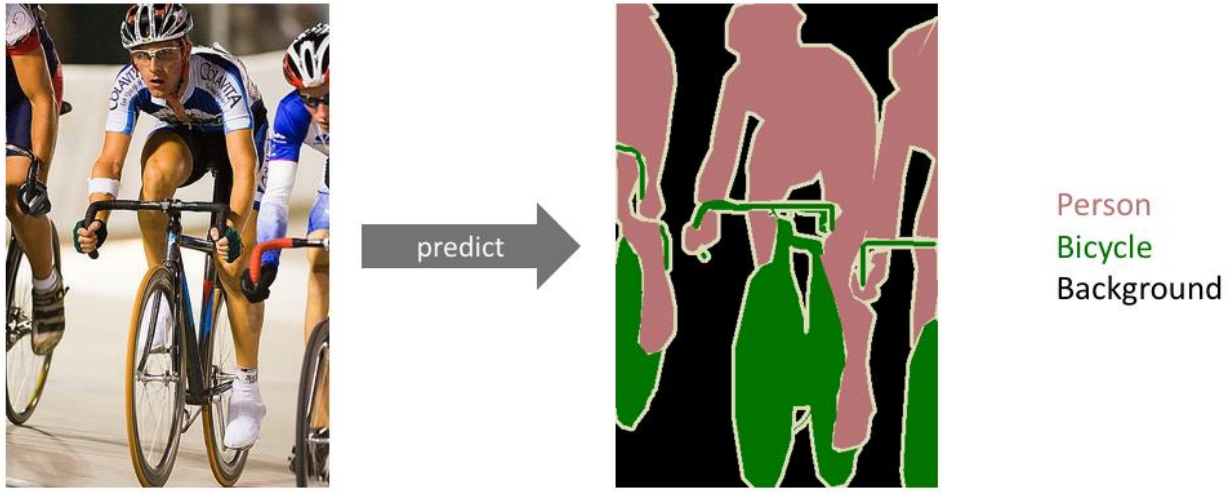


Computer Vision – Applications Image Classification



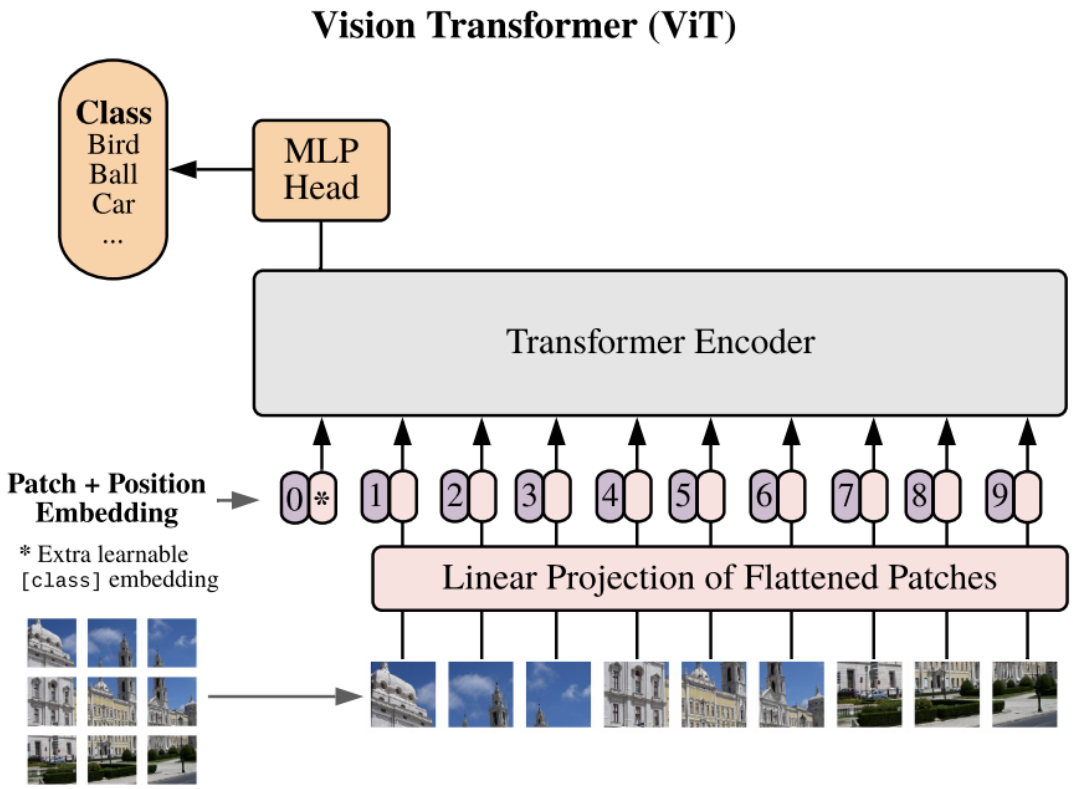
Computer Vision – Applications Object Detection

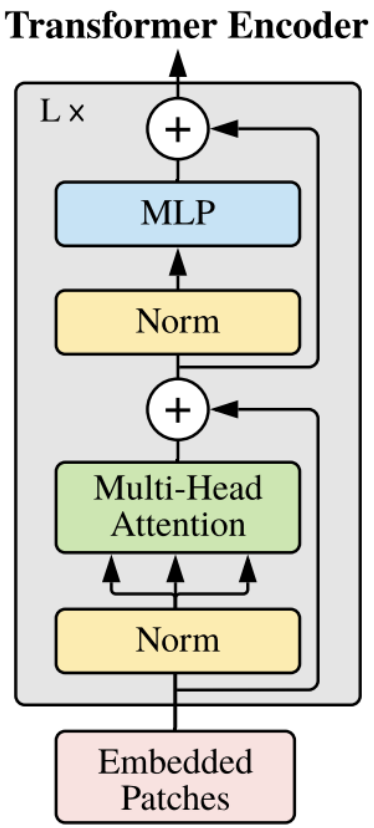
Computer Vision – Applications Semantic Segmentation



One popular approach for image segmentation models is to follow an encoder/decoder structure where we downsample the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and the upsample the feature representations into a full-resolution segmentation map.

Computer Vision – Vision Transformer





Challenges – training requires large datasets 14-300 million images

Augmentation
- RandAugment, Mixup

Regularization
- L1/L2, Stochastic depth

Computer Vision – Vision Transformer Self-supervised learning of visual features

- New Paradigm - Self-supervised pre-training, supervised fine-tuning
- Self-supervised learning allows to learn a task without a labels
- Methods: DINO, SwAV, Moco, SimCLR
- Contrastive loss to compute pairwise image similarities

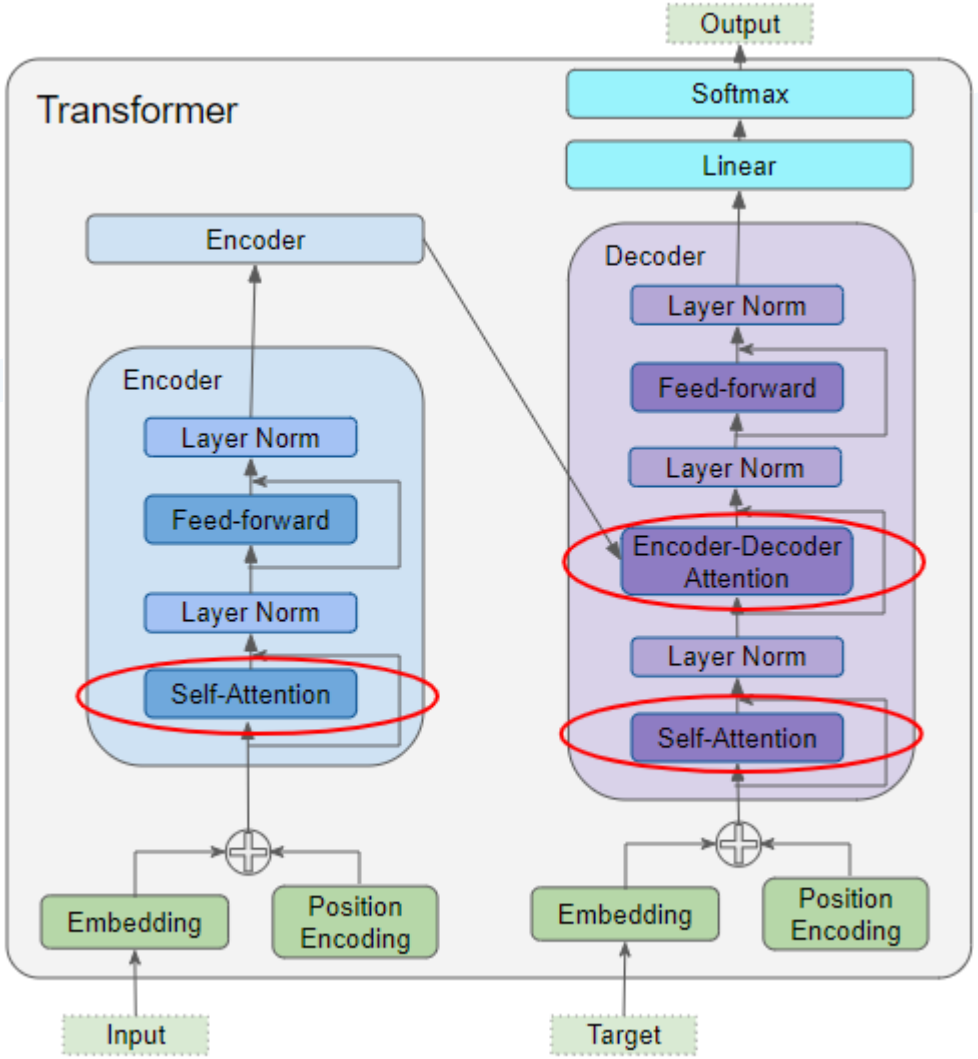
$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \; _{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)} ,$$

- Self-supervised vision transformer (ViT features contain explicit information about the semantic segmentation of an image, which does not emerge as clearly with supervised ViTs, nor with convnets

NLP – Natural Language Processing Applications

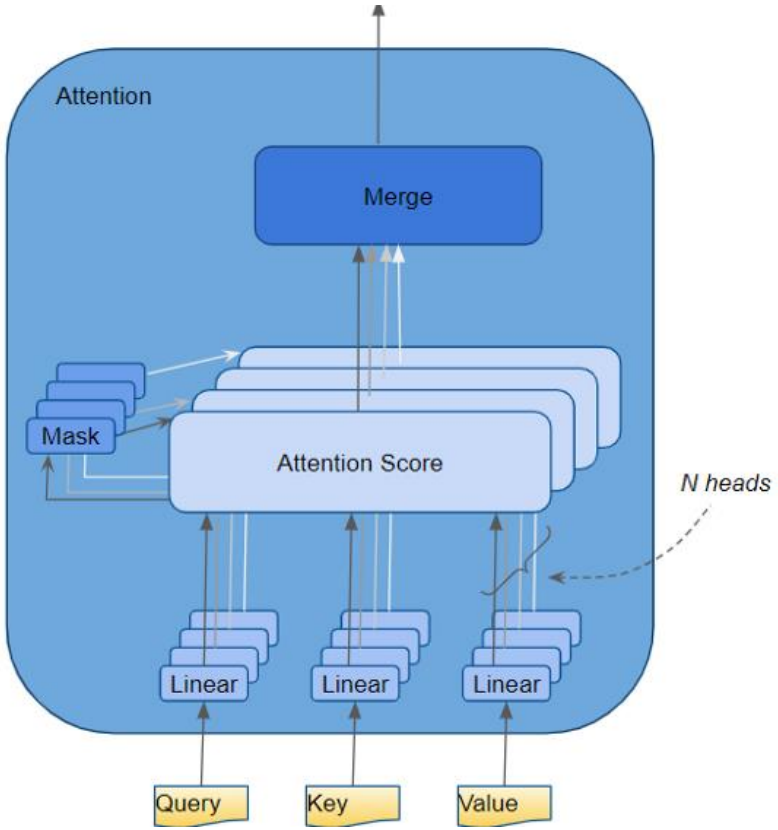
- **Search Engines**: NLP is what enables search engines to finish phrases and provide relevant results based on similar searches. However, NLP takes this even further. Search engines like Google don't just analyze the specific search words, but rather a wider picture that includes user intent.
- **Email Filters**: These were one of the first applications of NLP online, and while email filters used to be fairly basic, they are now far more complex. For example, Gmail utilizes NLP for email classification, labeling emails as either primary, social, or promotions based on the content. This creates an organized and manageable inbox for users with relevant emails standing out.
- **Voice Assistants**: One of the fastest-growing areas of NLP, voice assistants are already commonplace. Some of the most popular include Apple's Siri and Amazon's Alexa, which can infer meaning from human language, but there are countless others on the market as well. Voice assistants are quickly evolving and becoming even more complex, with many being able to pick up on contextual clues, respond with humor and other human emotions, and develop more personal connections with us.
- **Predictive Text**: Another one of the areas of NLP that we encounter every day is predictive text, which is present in things like autocorrect on our smartphones. It is hard for us to imagine what it was like before autocorrect, since it has become so ingrained in our communications. Predictive text can also customize itself to your language behaviors, and it becomes more accurate as time goes on.
- **Personality Analysis**: NLP can also be used for personality analysis from personal open-source data. Social media platforms contain massive amounts of data that provide insight into individuals' personalities, and algorithms can extract meaning from texts for a personality prediction system.
- **Sentiment Analysis**: NLP tools enable companies to analyze customer interactions, such as reviews, social media comments, and brand name mentions to gain insight into the emotional tone and opinions of customers. These can then be used to determine the efficiency of things like marketing campaigns, and it helps improve customer experience.

NLP – Natural Language Processing – Transformer Architecture



- **Self-attention in the Encoder** — the input sequence pays attention to itself
- **Self-attention in the Decoder** — the target sequence pays attention to itself
- **Encoder-Decoder-attention** in the Decoder — the target sequence pays attention to the input sequence
- **Models**: BERT, GPT-3, Sparse Transformer, LongFormer, BigBird, Performer, DistilBERT, Q8BERT

NLP – Natural Language Processing – Transformer Architecture



$$Z = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

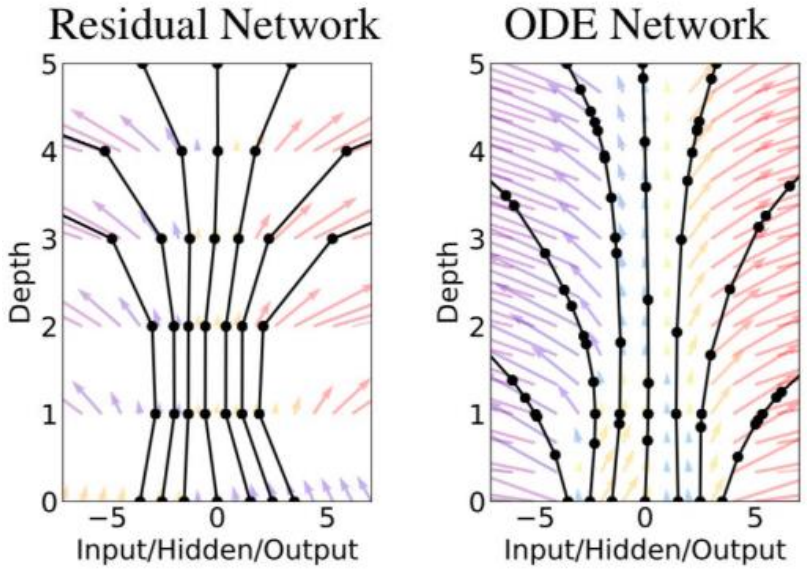
Attention Score is capturing some interaction between a particular word, and every other word in the sentence, by doing a dot product, and then adding them up.
Dot Product tells us the similarity between words

Scientific ML models, Physics Informed NN, Differentiable Programming

- **NODE-Neural ODE, NPDE-Neural PDE**

- ResNets as ODEs solutions - The y_{n+1} = y_n + f(t_n, y_n) is nothing but a residual connection in ResNet, where the output of some layer is a sum of the output of the layer f() itself and the input y_n to this layer. **This is basically the main idea of neural ODEs: a chain of residual blocks in a** neural network is basically a solution of the ODE with the Euler method!

- Adjoint sensitivity method – backpropagation through the solution of ODE



Geometric Deep Learning

- Graph Neural Network
  - BERT, Sparse Transformer, LongFormer, BigBird, Performer, DistilBERT, Q8BERT

Reinforcement Learning

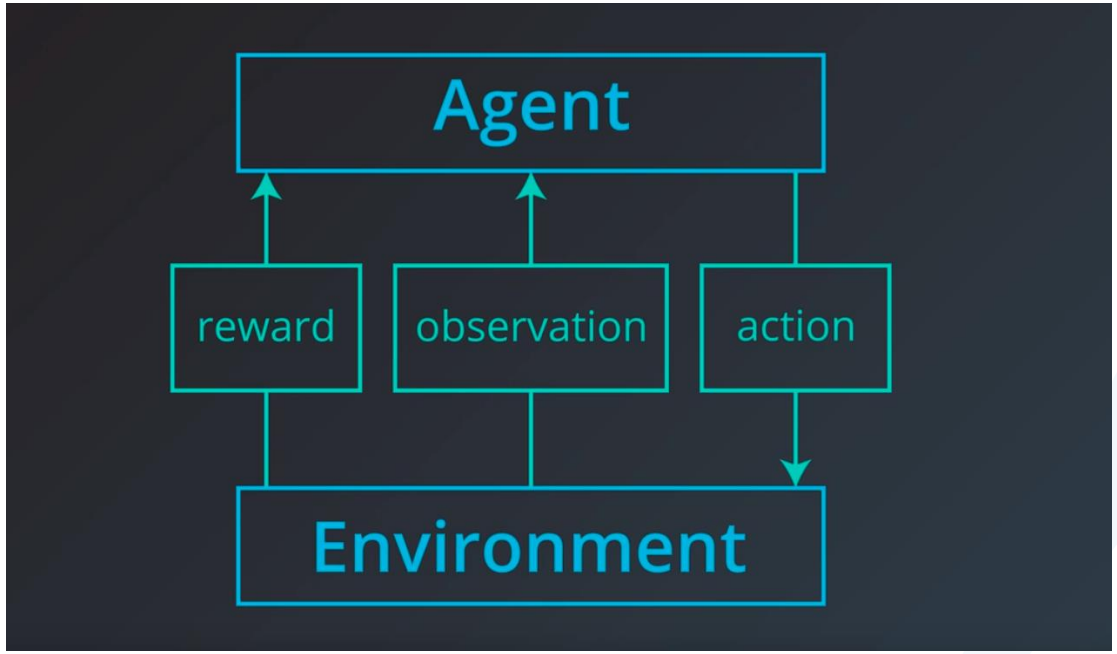
Reinforcement Learning is a subfield of machine learning which addresses the problem of automatic learning of optimal decisions over time.

RL is the task of learning through interaction. In this type of task, no human labels data and no human collects or designs the collection of data. These machine learning algorithms can be thought of as agents because of the need for interaction. The agents need to learn to perform a specific task, like in other machine learning paradigms. They also need to collect the most relevant data.

Very often, in RL, you must provide a reward signal. This signal is fundamentally different from the labels in supervised learning. In RL, agents receive reward signals for achieving a goal and not for specific agent behaviors. Additionally, this signal is related to an obviously desired state like winning a game, reaching an objective or location, and so on, which means that humans do not need to intervene by labeling millions of samples.

Reinforcement Learning

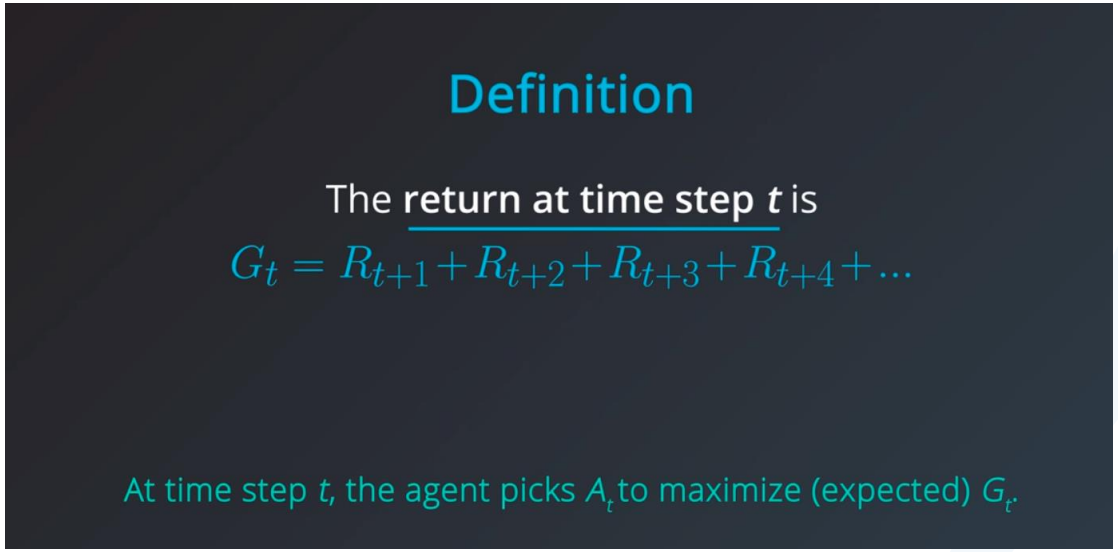
Major RL entities: **Agent and Environment**
Communication channels: **Actions, Reward, and Observations**



Reinforcement Learning

Objective of a Decision Making Agent

The agent's goal is to find a sequence of actions that will maximize the sum of rewards during the course of an episode or the entire life of the agent, depending on the task. The collection of rewards in a trajectory is called expected return.



Reinforcement Learning

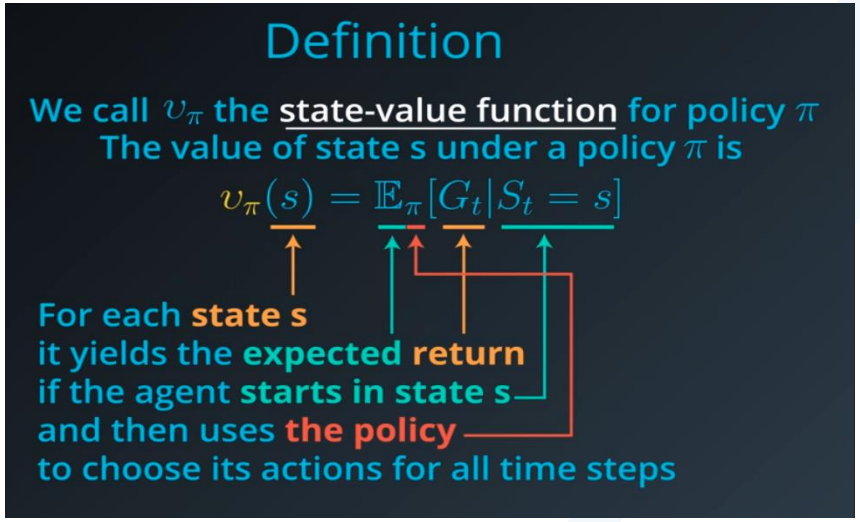
Objective of a Decision Making Agent

- The agent seeks to find the strategy for choosing actions with the cumulative reward is likely to be high.
- The agent can only accomplish this by interacting with the environment. This is because at every timestep, the environment decides how much reward the agent receives. In other words, the agent must play by the rules of the environment.
- Through interaction, the agent can learn those rules and choose appropriate actions to accomplish its goal.
- It's important to emphasize that all of this is just a mathematical model for a real world problem.

Reinforcement Learning

State-Value Function

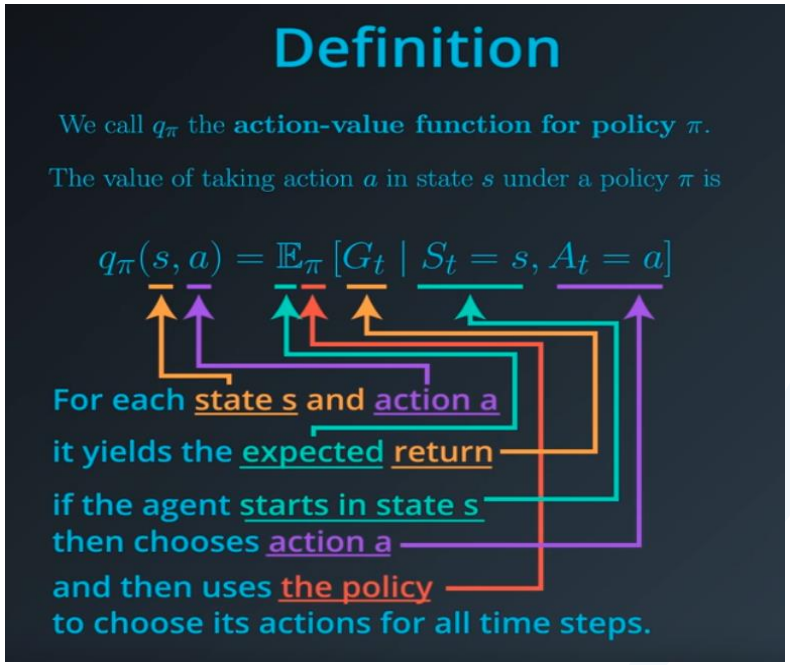
We have defined expected returns as a function of future rewards that the agent is trying to maximize. We now define the value of states following a policy: the value of a state s under policy π is the expectation of returns if the agent follows policy π starting from state s. This definition is called the state-value function and it simply represents the return an agent can expect to receive given an environment state and following a given policy.



Reinforcement Learning

Action-Value Function

The value of a state depends on the value of taking actions in that state. Therefore, we need a way to compare the values of taking any action in any state, so that we can identify the best action and thereby come up with the best policy. The action-value function is the expectation of returns if the agent follows policy π starting from state s and taking action a.



From Reinforcement Learning to Deep Reinforcement Learning

High-Dimensional State Space

- The main drawback of 'tabular' reinforcement learning is that the use of a table to represent value functions is no longer practical in complex problems. Environments can have high-dimensional state spaces, meaning that the number of variables that comprise a single state is very large.

Continuous State Space

- Environments can also have continuous variables, meaning that the number of values a single variable can be is infinite. For instance, the position and angles of a robot can be of infinitesimal precision; it could be 1.56 or 1.5683 or 1.5683256 and so on.

And of course, there are environments that have both high-dimensional and continuous variables

From Reinforcement Learning to Deep Reinforcement Learning

Function Approximators - Using neural networks to approximate Q-functions

- Neural networks are shown to be effective as universal function approximators. In fact, there is a universal approximation theorem that states that a single hidden layer feedforward neural network can approximate any continuous function that is closed and bounded in. It basically means that even simple (shallow) neural networks can approximate several functions.

- In environments with high-dimensional or continuous state spaces there is really no practical reason to create a table to store value functions. Sure, discretizing or binning the values could make tables possible. But, again, even if we could engineer a way to use tables and store our value functions there, by doing so, we'd be missing out on the advantages of generalization.

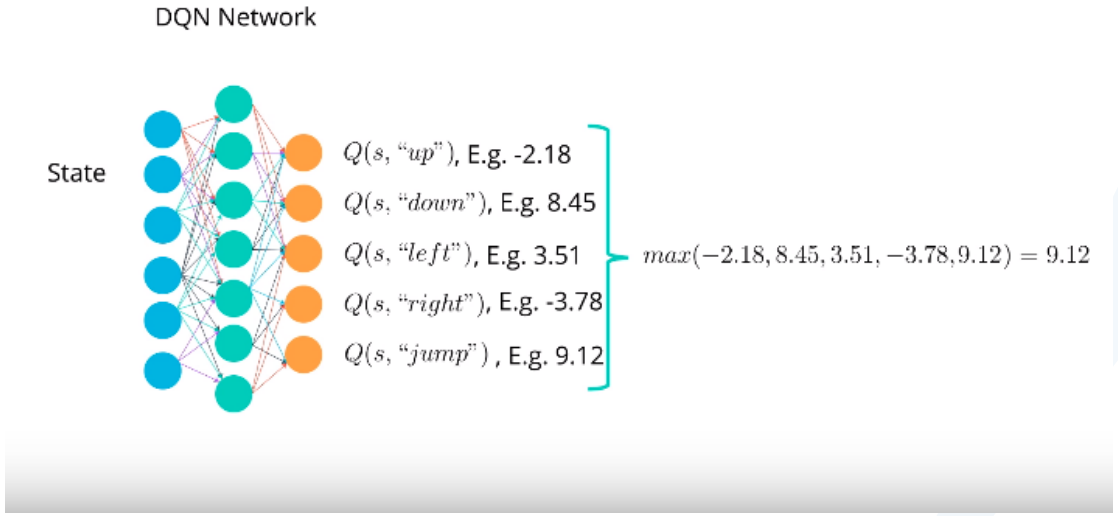
From Reinforcement Learning to Deep Reinforcement Learning

Generalization

- We would like our agents to generalize for example that 0.1 units away from the center is similar to 0.2, at least more so than 2.4.
- Action-value functions often have some underlying relationship that agents can exploit.
- We want to use generalization because it is a more efficient use of experiences. With function approximation, such as neural networks, agents learn and exploit patterns with less data (and perhaps faster).

Value Based Methods

Deep Q-Network - we will be approximating the action-value function Q(s,a), just like in Q-learning. We refer to the approximate action-value function as Q(s,a; w); that mean Q is parameterized by "w", the weights of a neural network, and indexed by a state "s" and an action "a".



Policy Gradient Methods

Policy Gradients - Introduction

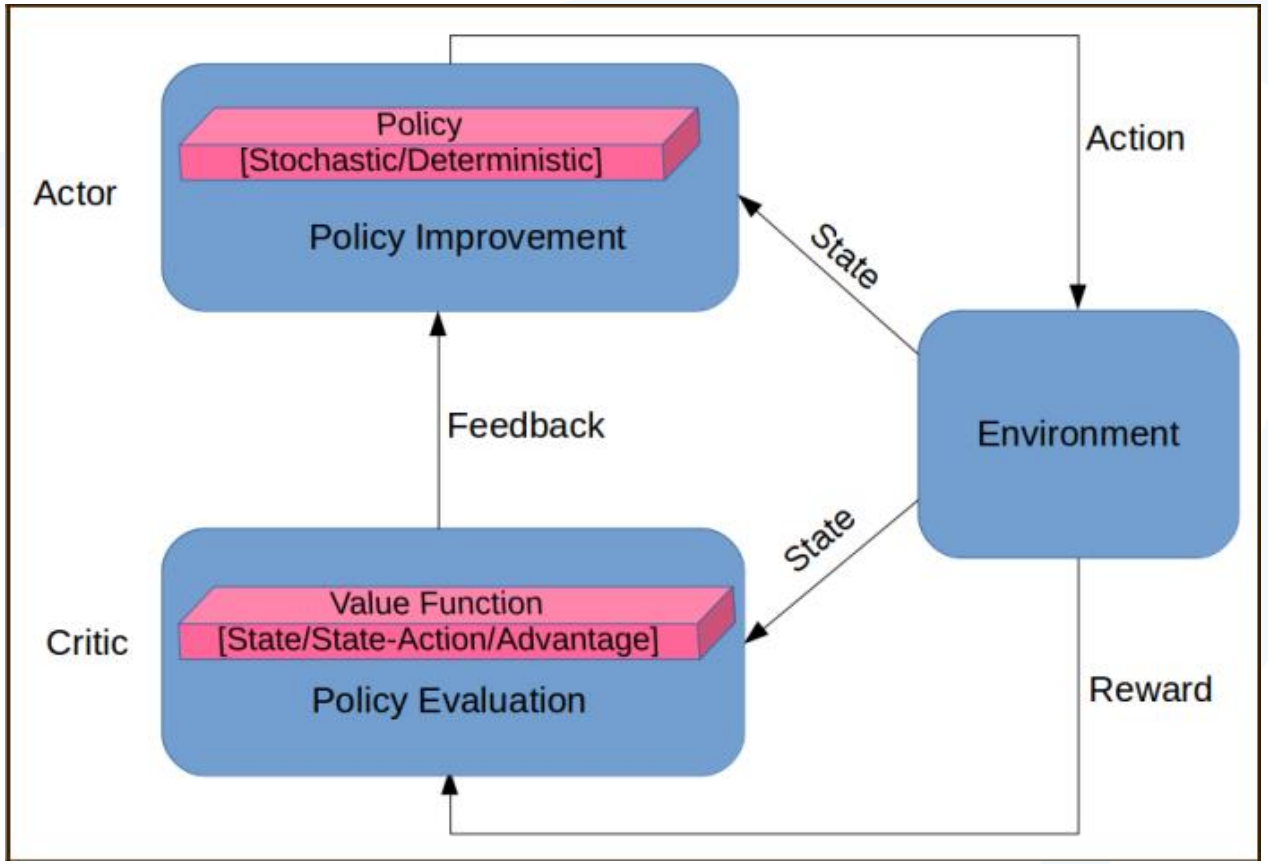
In Deep Q-Learning based intelligent agent implementation, we use a deep neural network as the function approximator to represent the action-value function. The agent then uses the action-value function to come up with a policy based on the value function. In particular, we use the epsilon-greedy algorithm.

Ultimately the agent has to know what actions are good to take given an observation/state. Instead of parametrizing or approximating a state/action action function and then deriving a policy based on that function, can we not parametrize the policy directly? Yes we can! That is the exact idea behind policy gradient methods.

Actor Critic Methods

- There are two components in the actor-critic algorithm.
- The actor is responsible for acting in the environment, which involves taking actions, given observations about the environment and based on the agent's policy. The actor can be thought of as the policy holder/maker.
- The critic takes care of estimating the state-value, or state-action-value, or advantage-value function (depending on the variant of the actor-critic algorithm used).

Actor Critic Methods



MACHINE LEARNING meetups

BRATISLAVA PRAHA KOŠICE BRNO

Special thanks for support to :

Data Analytics Meetings