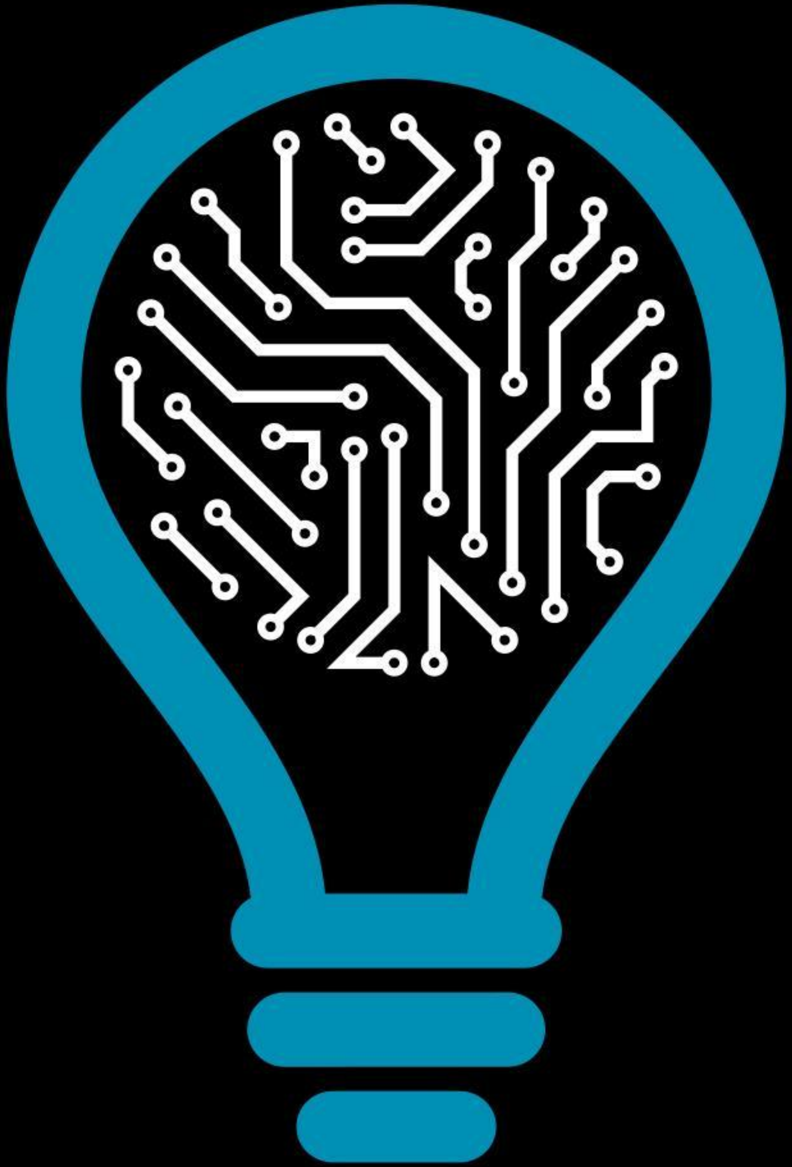
Edge AI - Lecture 1

TAIA - Advanced Topics on Artificial Intelligence

Tiago Filipe Sousa Gonçalves

tiago.f.goncalves@inesctec.pt | tiagofs@fe.up.pt

INESCTEC INSTITUTE FOR SYSTEMS AND COMPUTER ENGINEERING, TECHNOLOGY AND SCIENCE



Outline

1. Why are we investing in embedded AI?

What is embedded AI?

- **An embedded system is defined as any electronic system that includes some onboard computing capabilities, whether multi-purpose or specific[1]**, such as field programmable gate array (FPGA), system-on-a-chip (SoC), computer-on-module (COM), microcontroller unit (MCU), session border controllers (SBCs), graphics processing units (GPUs) or tensor processing units (TPUs)

- **Embedded AI is the application of machine and deep learning in software at the device level[2]**, i.e., the integration of algorithms at an hardware level

Figure - Virtex UltraScale+ HBM VCU128 FPGA Evaluation Kit (Xilinx)

Sources: [1] https://www.nwengineeringllc.com/article/computer-vision-in-embedded-systems-and-ai-platforms.php, [2] https://www.fierceelectronics.com/electronics/what-embedded-ai

INESCTEC

Why is AI moving to the device level?

- **There are several advantages of running AI models in embedded systems[1, 2], rather than running in cloud-based systems:**
    - **Less dependency on the cloud:** there is no need to transfer large amounts data to the cloud, which can incur considerable network latency and economise on bandwidth and network resources. This also reduces the risks of data breaches and privacy leaks
    - **Environmentally-friendly:** many embedded systems are power-efficient and can operate for a long time without being charged, thus leading to a lower carbon footprint, which yields much better sustainability
    - **Growing open-source ecosystem:** the open-source community and major software vendors have created a number of embedded OS options and accompanying applications. This also increases the level of modularity of embedded systems, allowing designers to build modular AI devices easily

Sources: [1] https://www.nwengineeringllc.com/article/the-future-of-embedded-ai-and-custom-socs.php, [2] https://www.edgeimpulse.com/blog/the-embedded-machine-learning-revolution-the-basics-you-need-to-know

Can we use embedded AI in real applications?

- **The market for embedded systems is currently growing[1],** thus creating innovation opportunities for enterprises that wish to make the best possible use of their data
- **Currently, the most trending fields of application for embedded AI are[2, 3, 4]:**
  - **Manufacturing & Industry 4.0:** several areas (e.g., agriculture, aviation, supply chain, shipping, retail, healthcare, finance) will benefit of embedded AI systems for defect inspection, production asset inspection, and asset tracking inside and outside a connected factory
  - **Autonomous Vehicles & Robots:** these systems will eventually deploy their computer vision models into embedded systems
  - **Biometrics & Security:** embedded AI systems may enhance several tasks such as object detection, facial/fingerprint recognition or speech/audio/video processing, for the sake of efficiency
  - **Customer Experience:** can we depend less on the cloud to achieve state-of-the-art recommendation systems?

Sources: [1] https://www.nwengineeringllc.com/article/iot-market-trends-in-2019-pcb-design-and-manufacturing.php, [2] https://www.analyticsinsight.net/how-to-integrate-artificial-learning-with-embedded-systems/, [3] https://www.nwengineeringllc.com/article/computer-vision-in-embedded-systems-and-ai-platforms.php, [4] https://www.fierceelectronics.com/electronics/what-embedded-ai

What if we want to develop some hands-on projects?

- **Some technological companies have developed several development kits for enthusiasts that want to learn about AI or use embedded AI to power their personal/academic projects[1, 2]**

Figure - Jetson Nano 2GB Developer Kit (NVIDIA)

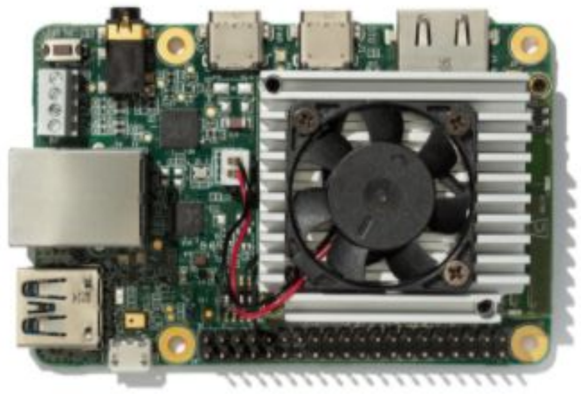


Figure - Dev Board (Coral, Google)

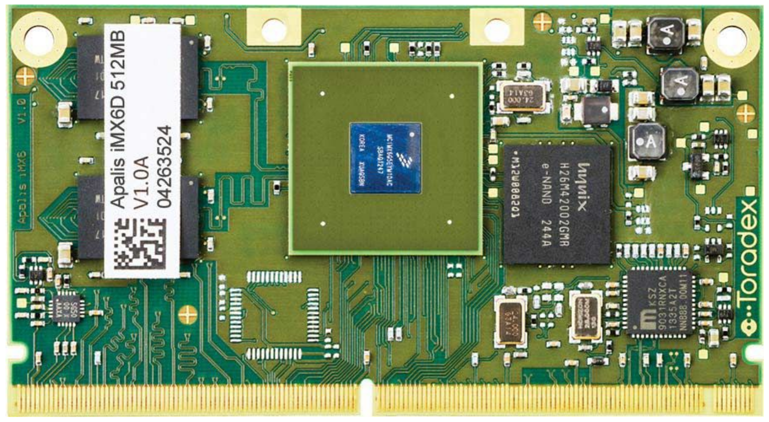


Figure - Apalis iMX6 (Toradex)

Sources: [1] https://www.nwengineeringllc.com/article/computer-vision-in-embedded-systems-and-ai-platforms.php, [2] https://analyticsindiamag.com/a-beginners-guide-to-machine-learning-for-embedded-systems/

Embedded AI is great, but it has a few challenges...

- **The talent gap with machine learning:** currently, there is a fierce competition for talent in fields related to data science, machine learning and embedded systems[1]
  - Embedded AI requires developers and engineers with knowledge, skills, and talent from both worlds, as well as familiarity with a variety of different systems and tools[2]

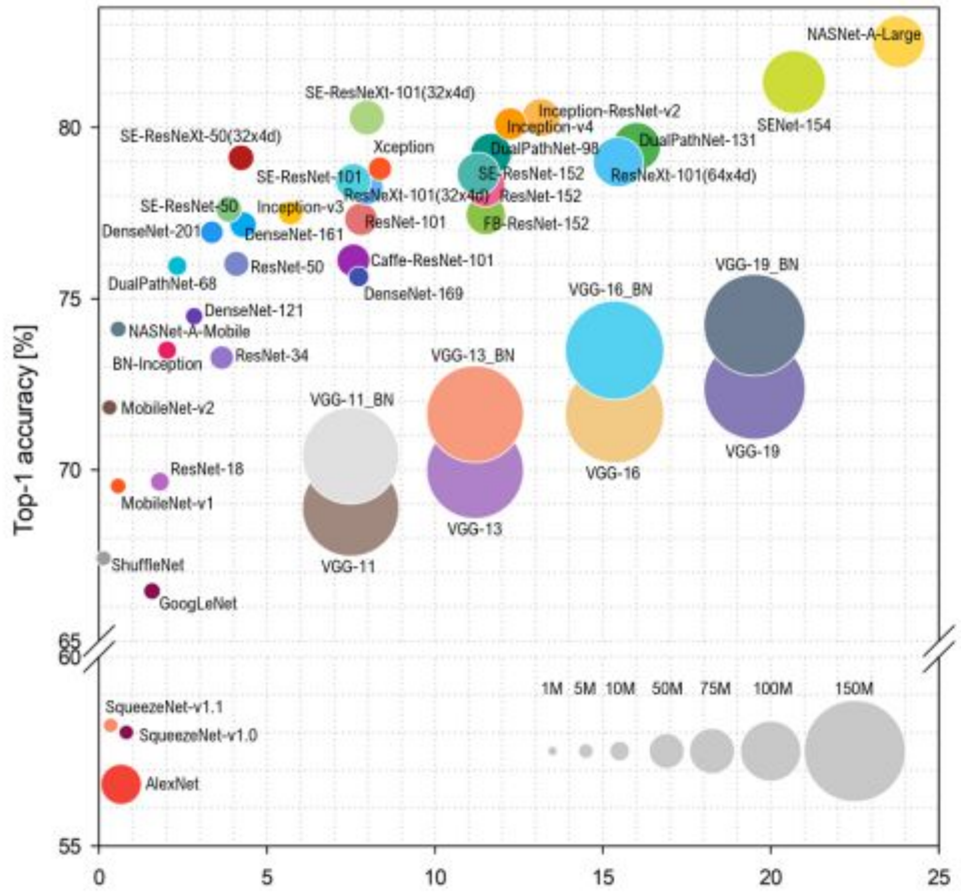
- **Data gathering:** the process of collecting and pre-processing data from embedded devices is typically more cumbersome than collecting and using data from other IT systems and databases[3]

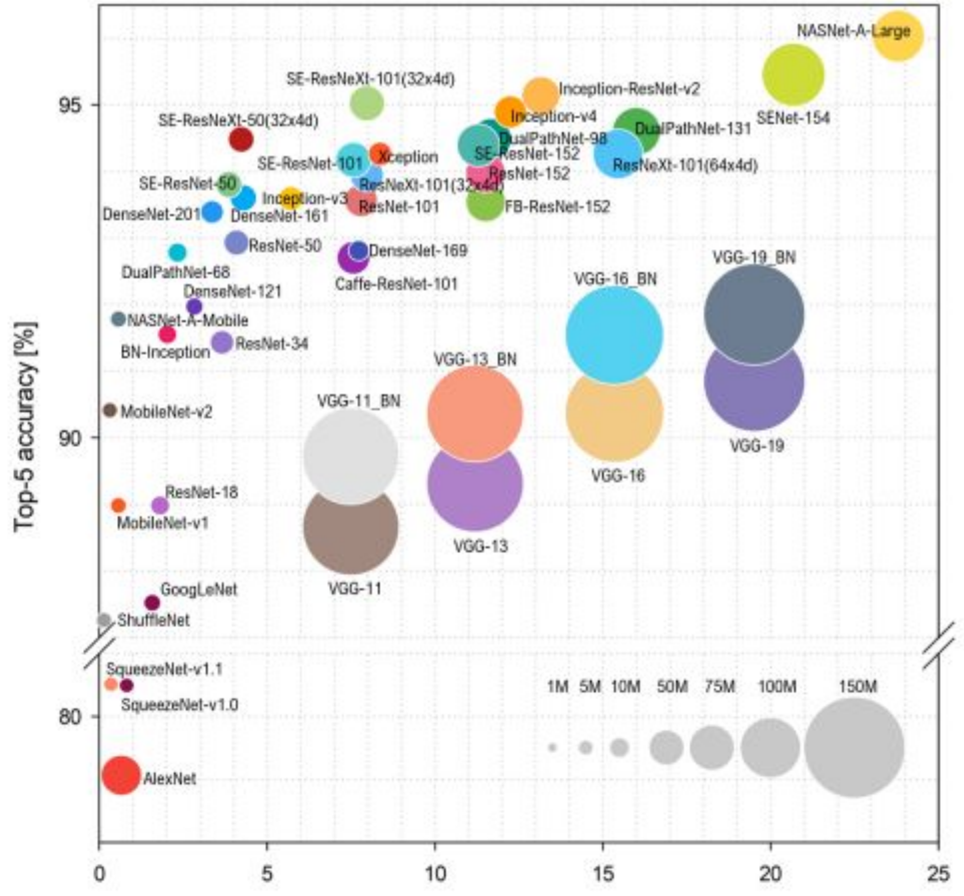
- **Sustainability:** there is a need to cope with energy efficiency and resource usage issues, which stem from the use of resource constrained devices[4]

Sources: [1] https://www.forbes.com/sites/louiscolumbus/2018/04/17/how-to-close-the-talent-gap-with-machine-learning/?sh=655703cc4e96, [2] https://www.edgeimpulse.com/blog/the-embedded-machine-learning-revolution-the-basics-you-need-to-know, [3] Lee et al. "Integrating machine learning in embedded sensor systems for Internet-of-Things applications,", [4] Andrade et al. "Overview of the state of the art in embedded machine learning,"

2. We need small models: an introduction to model compression

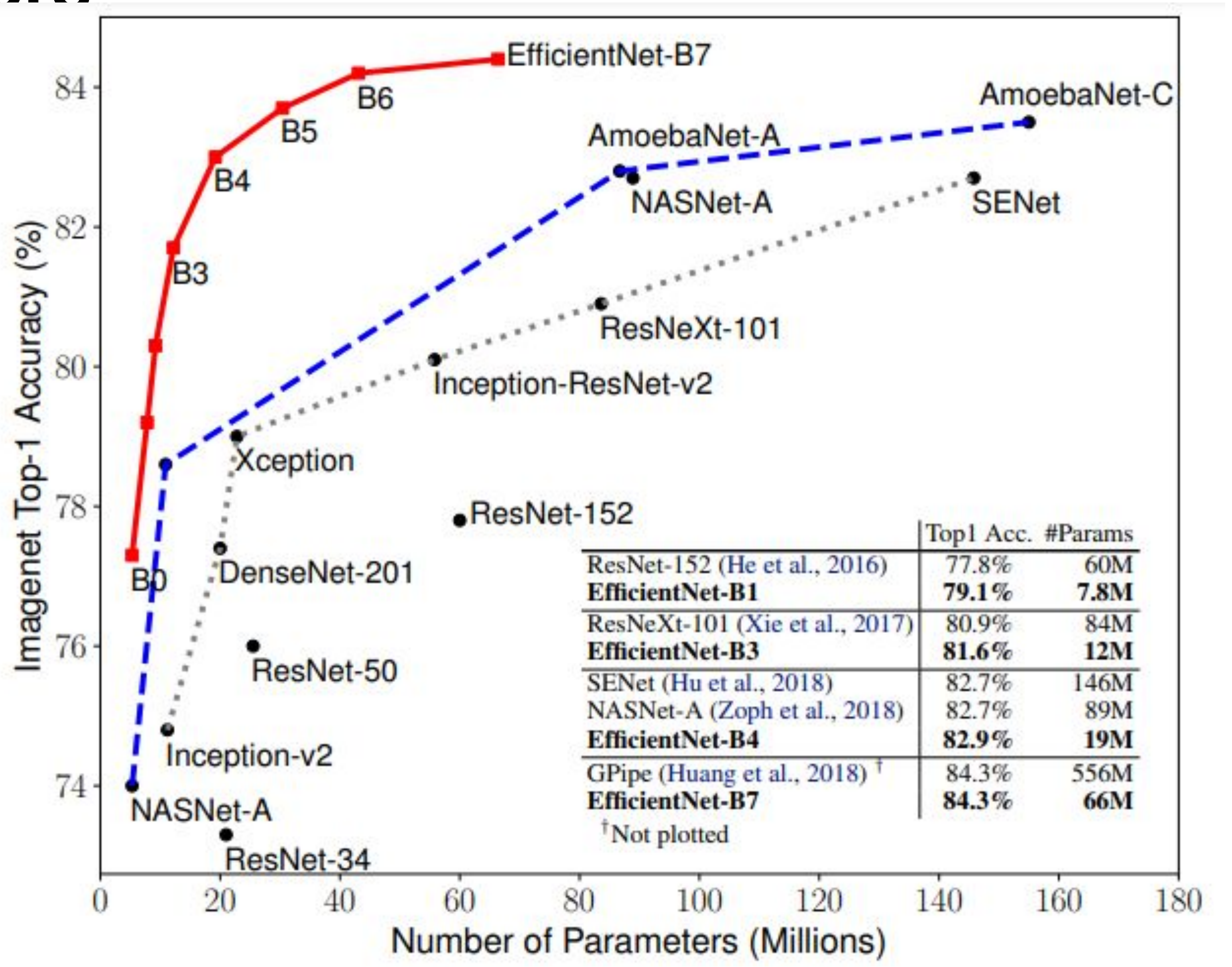
High-performing models have millions of parameters[1]





Sources: [1] Bianco et al. "Benchmark analysis of representative deep neural network architectures."

High-performing models have millions of parameters[1]



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Sources: [1] Mingxing and Le "Efficientnet: Rethinking model scaling for convolutional neural networks."

But… Do we really need millions of parameters?

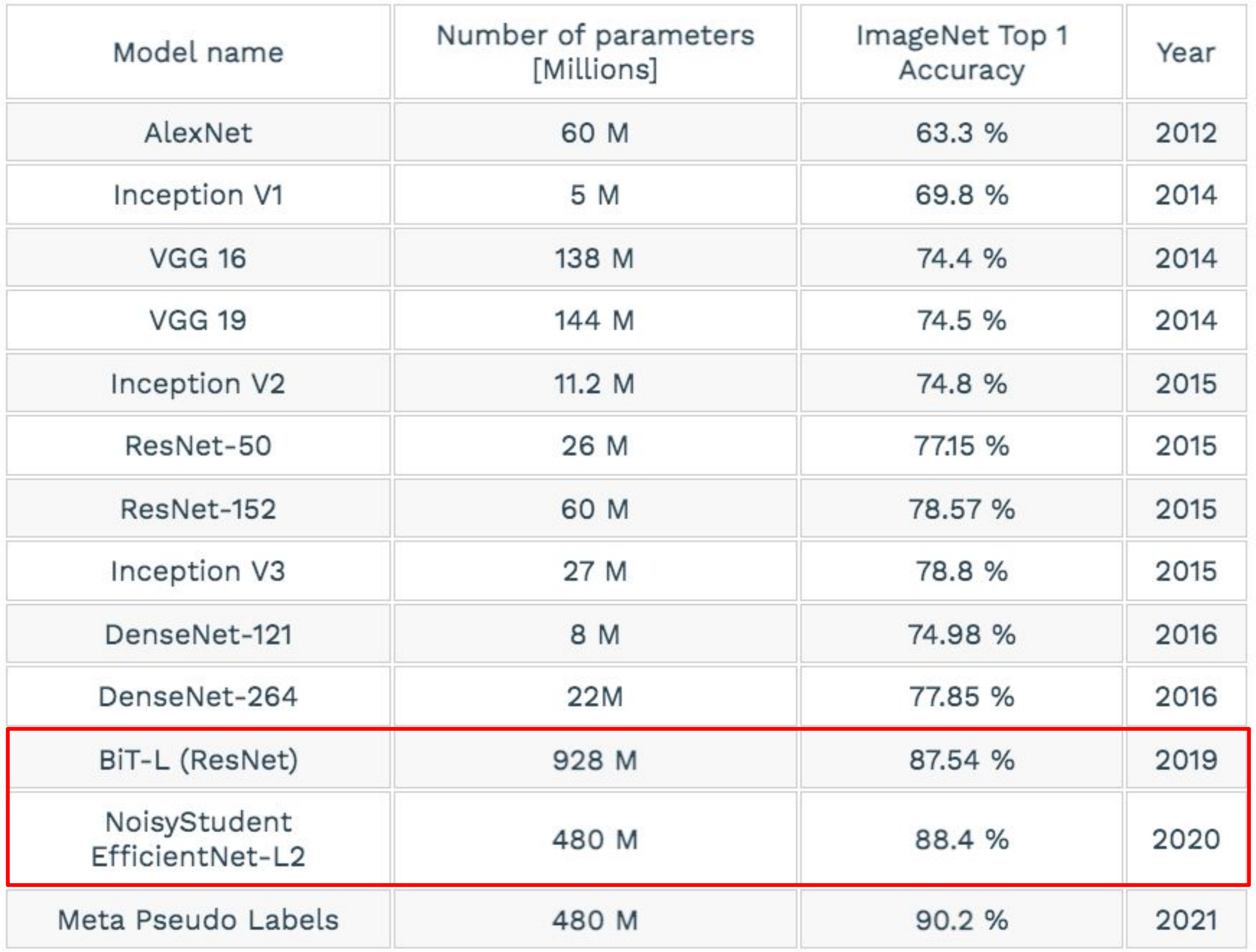
- **There is an inherent tension between complexity and predictive performance,** however, there is no directly dependence between model complexity and good performance[1]

- **Big models require lots of resources**, which means that we will have to address challenges related to difficulty in training and environmental costs (e.g., number of GPUs, RAM, processing power)

- **Once again, do we really want to use these models in production? Well, it depends on the use case[1]**
  - If you want to deploy your models for autonomous driving, perhaps, you want it to be integrated with the car, with no intermediate modules
  - On the other hand, if you want to use this model in a recommendation system that runs on the cloud, you may be comfortable with using an intermediate layer of processing
  - **Buzz words you should bear in mind:** lower latencies, data privacy, human-computer interaction

Sources: [1] https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/

But… Do we really need millions of parameters?[1]



Model name	Number of parameters [Millions]	ImageNet Top 1 Accuracy	Year
AlexNet	60 M	63.3 %	2012
Inception V1	5 M	69.8 %	2014
VGG 16	138 M	74.4 %	2014
VGG 19	144 M	74.5 %	2014
Inception V2	11.2 M	74.8 %	2015
ResNet-50	26 M	77.15 %	2015
ResNet-152	60 M	78.57 %	2015
Inception V3	27 M	78.8 %	2015
DenseNet-121	8 M	74.98 %	2016
DenseNet-264	22M	77.85 %	2016
BiT-L (ResNet)	928 M	87.54 %	2019
NoisyStudent EfficientNet-L2	480 M	88.4 %	2020
Meta Pseudo Labels	480 M	90.2 %	2021

Sources: [1] https://theaisummer.com/cnn-architectures/

What if we *compress* models instead?

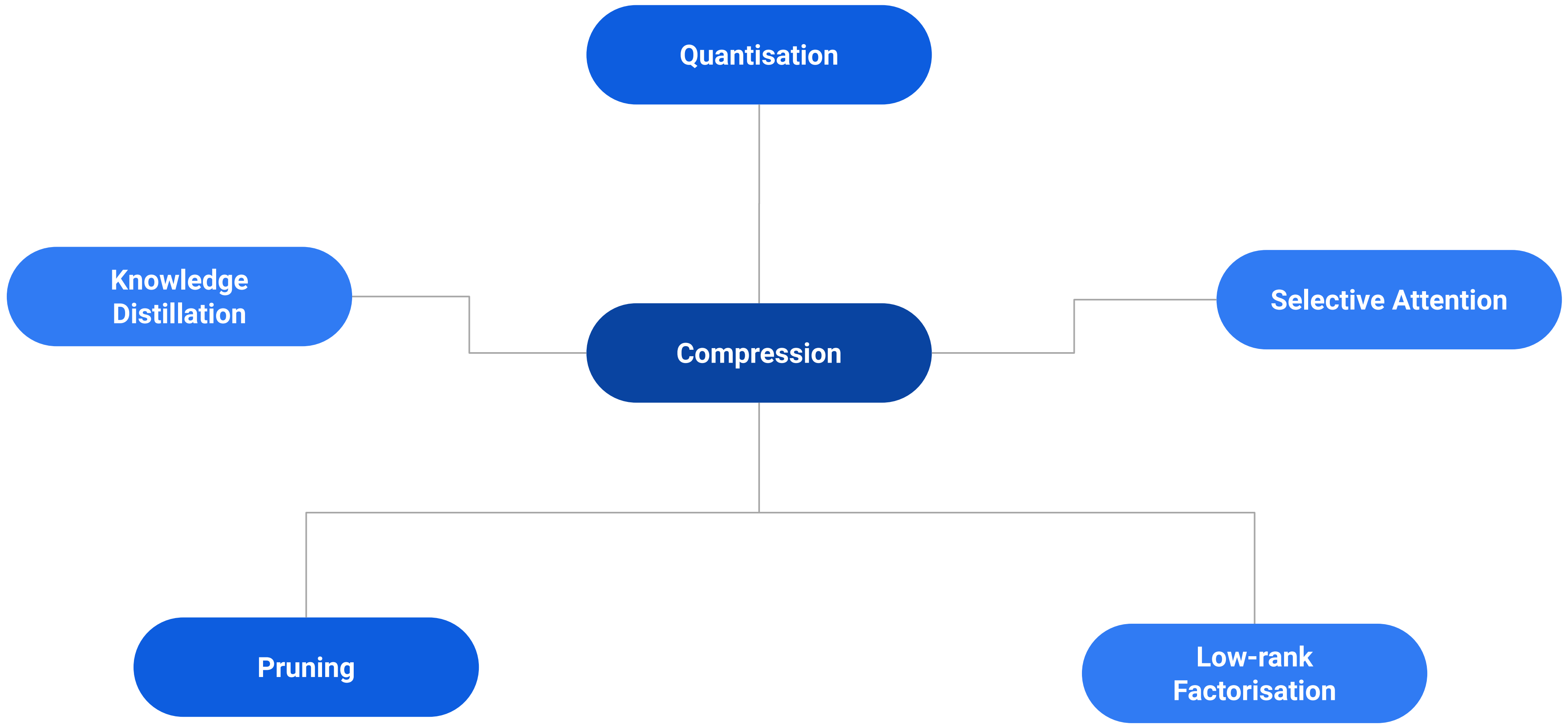
- **We do not need all the parameters of the model[1]**
  - Individual network weights can be redundant, and may not carry significant information
  - Although large models often have the ability to completely memorize datasets, they instead appear to learn generic task solutions
  - A standing hypothesis for why overcomplete representations are necessary is to discover robust solutions, which do not rely on precise weight values

- **Hypothesis: if large models are only needed for robustness during training, then significant compression of these models should be achievable, without impacting accuracy**

- **This leads us to *model compression*,** which is "an actively pursued area of research over the last few years with the goal of deploying state-of-the-art deep networks in low-power and resource limited devices without significant drop in accuracy"[2]

Sources: [1] Antonio et al. "Model compression via distillation and quantization", [2] https://paperswithcode.com/task/model-compression

There are several model compression techniques[1]



Sources: [1] Antonio et al. "Model compression via distillation and quantization"

Pruning: the fine art of removing parameters

- **Model pruning consists of removing the parameters that, perhaps, will have low impact on performance[1, 2]**

- **There are essentially two strategies:**
  - **Unstructured pruning**, which consists of removing individual parameters
  - **Structured pruning**, which consists of removing complete layers

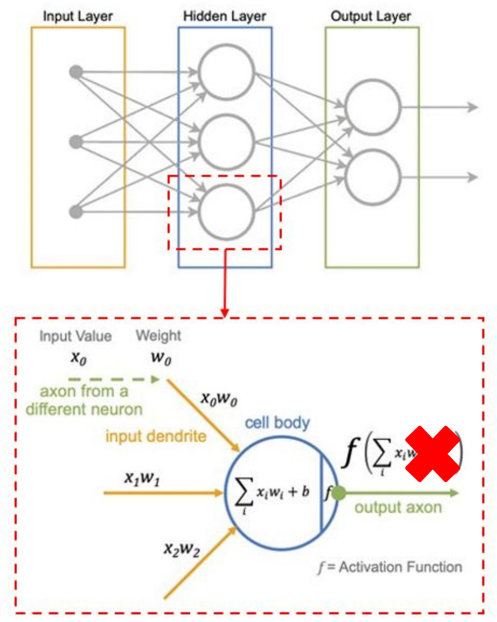


Figure - Unstructured pruning
(Image from [2])

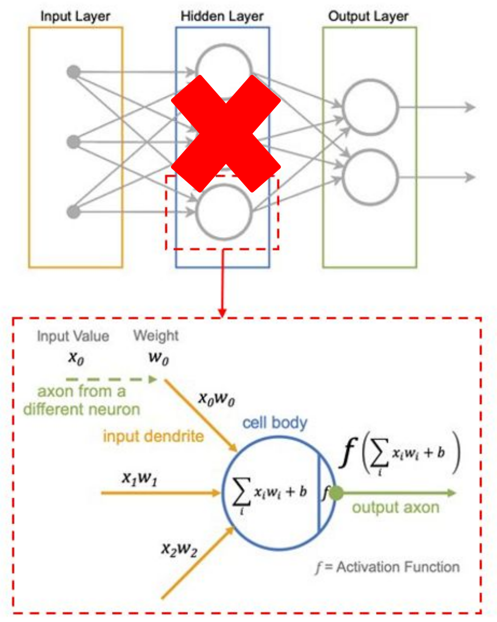


Figure - Structured pruning
(Image from [2])

Sources: [1] https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/, [2] https://siddharthasaxena.medium.com/model-compression-the-what-why-and-how-of-it-bc1416f5b18f

What is the typical pipeline for model pruning?

1. **We start by training a large model until it converges** (i.e., achieves good predictive performances)
2. **We then select and remove the layers or the parameters** that contribute less to the final outputs of the model
3. **We retrain the pruned model** until it converges
4. We repeat this process until it no longer converges

Note: Current studies suggest that training a pruned model from scratch performs worse than retraining a pruned model[1]. Moreover, during retraining, it is better to retain the weights from the initial training phase for the connections that survived pruning than it is to re-initialize the pruned layers, since gradient descent seems to be able to find a good solution when the network is initially trained, but not after re-initializing some layers and retraining them[2]

Sources: [1] Li et al. “Pruning filters for efficient convnets”, **[2]** Han et al. “Learning both weights and connections for efficient neural network”

Case Study: The Lottery Ticket Hypothesis[1]

- **Research Question:** "If a network can be reduced in size, why do we not train this smaller architecture instead in the interest of making training more efficient as well?"

- **The Lottery Ticket Hypothesis:** "A randomly-initialised, dense neural network contains a subnetwork that is initialized such that, when trained in isolation, it can match the test accuracy of the original network after training for at most the same number of iterations"
  - These trainable subnetworks are ***winning tickets***, since they have won the initialisation lottery with a combination of weights and connections capable of learning

- **Identifying *winning tickets*:**
  - Train a network and prune its smallest-magnitude weights
  - The unpruned connections constitute the architecture of the winning ticket
  - Reset each unpruned connection's value is then reset to its initialization from original network before it was trained

Sources: [1] Frankle and Carbin "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks"

Case Study: The Lottery Ticket Hypothesis[1]

- **Why is this study important?**
  - **Improved training performance:** can design training schemes that search for winning tickets and prune as early as possible?
  - **Design of better networks:** can we take inspiration from winning tickets to design new architectures and initialization schemes with the same properties that are conducive to learning? Or will we be able to transfer winning tickets discovered for one task to many others?
  - **Extend the theory of neural networks:** can we study why randomly-initialised feed-forward networks seem to contain winning tickets and understand their potential implications for theoretical study of optimization[2] and generalization[3, 4]?

- **Future challenges:**
  - Explore more efficient methods for finding winning tickets that will make it possible to study the lottery ticket hypothesis in more resource-intensive settings
  - Study the properties of the initialisations that, in concert with the inductive biases of the pruned network architectures, make these networks particularly adept at learning

Sources: [1] Frankle and Carbin "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks", [2] Du et al. "Gradient descent provably optimizes over-parameterized neural networks", [3] Zhou et al. "Compressibility and Generalization in Large-Scale Deep Learning", [4] Arora et al. "Stronger generalization bounds for deep nets via a compression approach"

Quantisation: decreasing the size of weights

- **Quantisation is a process of mapping values from a large set to values in a smaller set**, i.e., the output contains a smaller range of values compared to the input without losing much information in the process[1]
  - For instance, we may want to use **less bits** (e.g., *float16* instead of *float32*) to represent the model's weights (i.e., the model can be **stored in less space**) and still achieve **similar predictive performances** with **faster inference times**[2]

- **According to Intel**, researchers have demonstrated that deep learning inference can be performed with lower numerical precision, using 8-bit multipliers for inference with minimal to no loss in accuracy![3]

- **Benefits of lower numerical precision[3]:**
  - Reducing precision would allow for better usage of cache and reduction of bandwidth bottlenecks, since many operations are memory bandwidth bound
  - The hardware may enable higher operations per second at lower numerical precision, as these multipliers require less silicon area and power

Sources: [1] https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/, **[2]** https://siddharthasaxena.medium.com/model-compression-the-what-why-and-how-of-it-bc1416f5b18f, **[3]** https://www.intel.com/content/www/us/en/developer/articles/technical/accelerate-lower-numerical-precision-inference-with-intel-deep-learning-boost.html

Model quantisation can be applied in several ways

- **Similarly to other regularisation strategies, there are several approaches to model quantisation[1]:**
  - We start by defining a scaling function which normalizes vectors whose values come from an arbitrary range, to vectors whose values are in [0, 1]

- **We may want to apply several modifications to the scaling function[2]**
  - **Bucketing:** we will apply the scaling function separately to buckets of consecutive values of a certain fixed size[3]
  - **Uniform Quantisation:** we fix a parameter *s* describing the number of quantization levels employed; uniform quantization considers *s + 1* equally spaced points between 0 and 1 (including these endpoints)
  - **Non-Uniform Quantisation:** non-uniform quantisation takes as input a set of quantisation points and quantises each element to the closest of these points

Sources: [1] Polino et al. “Model Compression via Distillation and Quantization”, [2] He et al. “Effective quantization methods for recurrent neural networks”, 3] Alistarh et al. “Qsgd: Randomized quantization for communication-optimal stochastic gradient descent”

Case Study: One Model to Rule Them All[1]

- **Context:** naively quantising a floating point model to 4 bits, or lower, usually incurs a significant accuracy degradation. Studies have tried to mitigate this by offering different quantisation methods:
  - Methods that require training (quantisation aware training) simulate the quantisation arithmetic on the fly
  - Methods that avoid training (post-training quantisation) quantise the model after the training while minimising the quantisation noise

- **Research Problem:** these methods create models sensitive to the precise way quantisation is done (e.g., target bit-width)
  - For instance, to avoid accuracy degradation at inference time, it is essential to ensure that all quantisation-related artifacts are faithfully modeled at training time

- **The Generic Model Hypothesis:** “To allow rapid and easy deployment of DNNs on embedded low-precision accelerators, a single pre-trained generic model that can be deployed on a wide range of deep learning accelerators would be very appealing. Such a robust and generic model would allow DNN practitioners to provide a single off-the-shelf robust model suitable for every accelerator, regardless of the supported mix of data types, precise quantisation process, and without the need to re-train the model on customer side”

Sources: [1] Shkolnik et al. “Robust Quantization: One Model to Rule Them All”

Case Study: One Model to Rule Them All[1]

- **Proposal and Results:** they suggest a generic method to produce robust quantisation models, KURE (KUrtosis REgularization term), which is added to the model loss function
    - Two important use cases for improving quantisation robustness: robustness to quantisation across different bit-widths and robustness across different quantisation policies
    - Uniformly distributed tensors are much less sensitive to variations compared to normally distributed tensors, which are the typical distributions of weights and activations

- **Why is this important?**
    - Deep neural networks take up tremendous amounts of energy
    - Quantization can improve energy efficiency of neural networks on both commodity GPUs and specialised accelerators
    - Robust quantisation takes another step and create one model that can be deployed across many different inference chips avoiding the need to re-train it before deployment

Sources: [1] Shkolnik et al. "Robust Quantization: One Model to Rule Them All"

Knowledge distillation: the Teacher and the Student

- **The main principle behind knowledge distillation is to use an existing larger model (i.e., the Teacher) to train a smaller model (i.e., the Student)[1]**
  - We want the student model to have the same distribution of the teacher model (e.g., using the **Kullback-Leibler Divergence loss**)

- Interestingly, **this can be interpreted similarly to pruning**, but the final network's **characteristics are decided already**, unlike in pruning, where they are inferred[2]

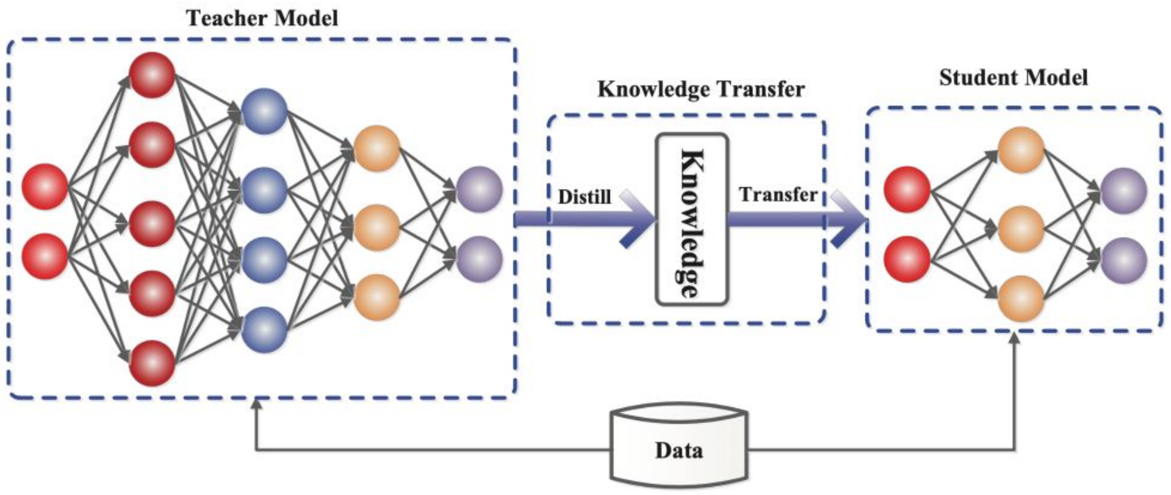


Figure - The generic framework for knowledge distillation
(Image from [3])

Sources: [1] https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/, [2] https://siddharthasaxena.medium.com/model-compression-the-what-why-and-how-of-it-bc1416f5b18f, [3] Gou et al. "Knowledge Distillation: A Survey"

The Student can choose different learning paths[1]

- In this context, **knowledge is defined by the learned weights and biases**
- There are three types of knowledge:
  - **Feature-based knowledge**
  - **Relation-based knowledge**
  - **Response-based knowledge**

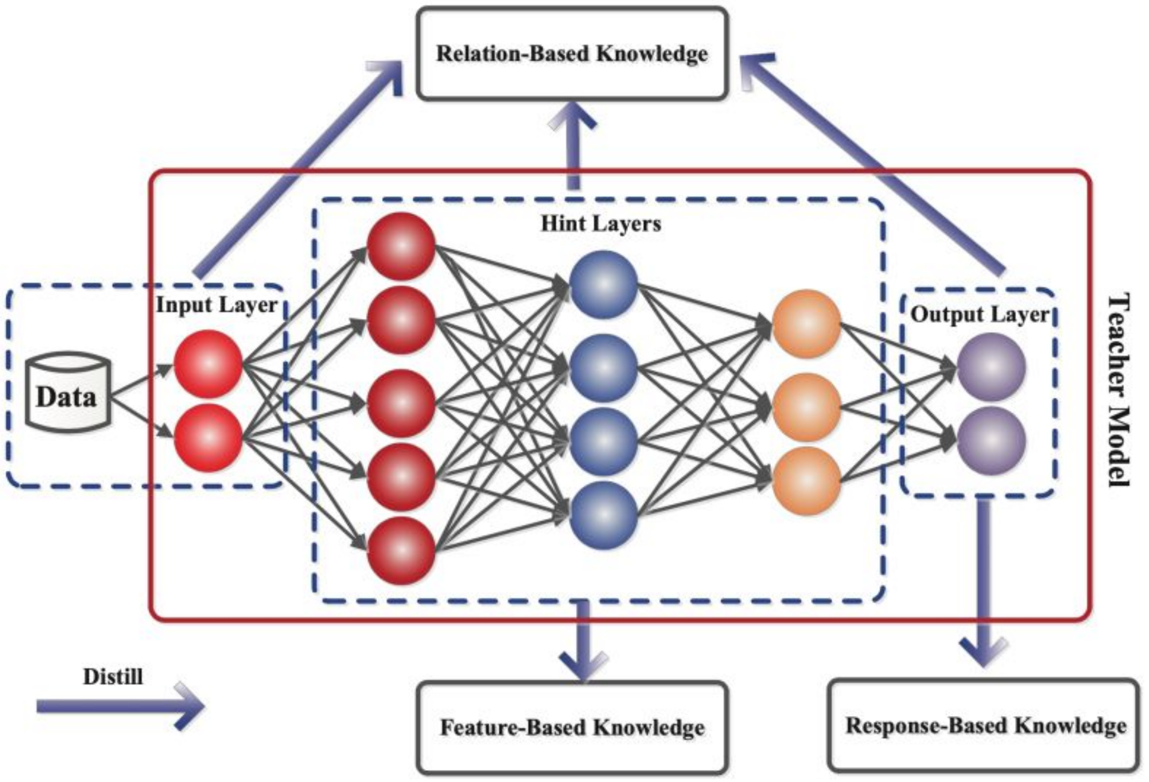


Figure - The different learning paths (Image from [2])

Sources: [1] https://neptune.ai/blog/knowledge-distillation, [2] Gou et al. “Knowledge Distillation: A Survey”

Feature-based knowledge distillation[1]

- **The main idea here is to gather knowledge from the intermediate layers of the Teacher model**
    - Intermediate layers are important because they learn to discriminate specific features

- Therefore, we want to **minimise the difference between the feature activations of the teacher and the student models** (distillation loss function)

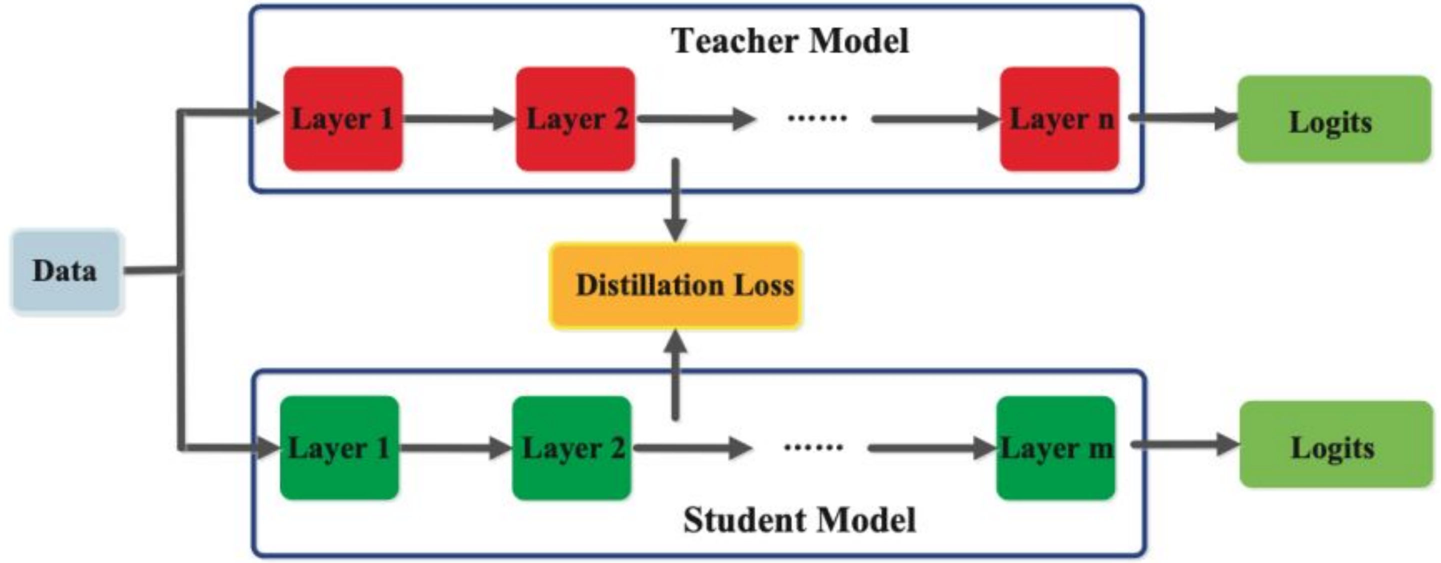


Figure - Feature-based knowledge distillation (Image from [2])

Sources: [1] https://neptune.ai/blog/knowledge-distillation, [2] Gou et al. "Knowledge Distillation: A Survey"

Relation-based knowledge distillation[1]

- **The main idea here is to gather knowledge from the relationship between feature maps of the Teacher model**
  - This relationship can be modeled as a correlation between feature maps, graphs, similarity matrix, feature embeddings, or probabilistic distributions based on feature representations[1]

- Therefore, we want to **minimise the difference between the instance relations**

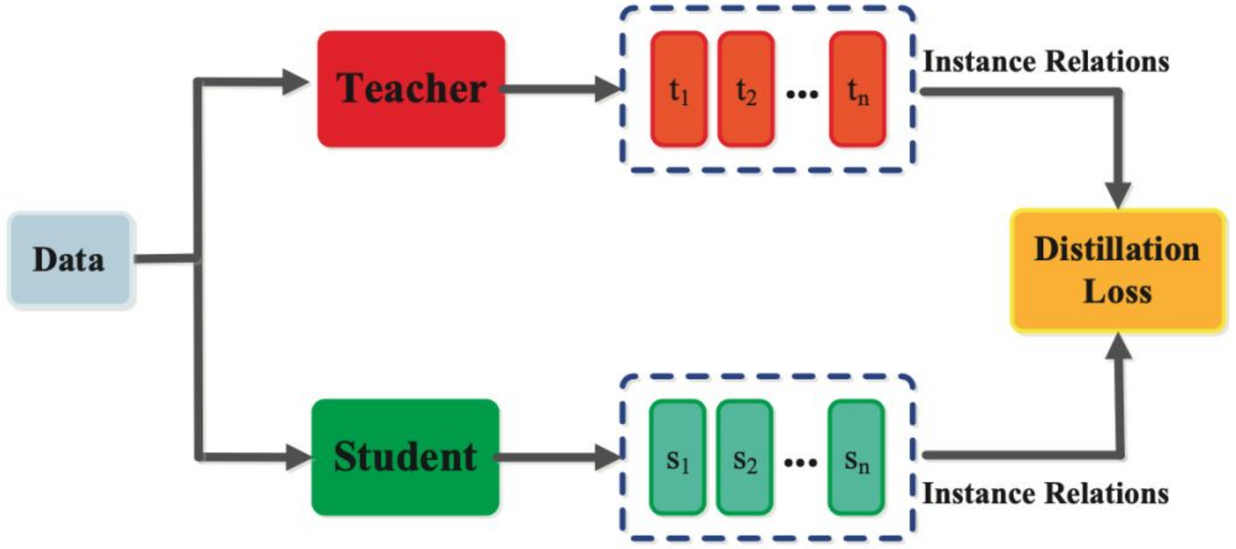


Figure - Relation-based knowledge distillation (Image from [2])

Sources: [1] https://neptune.ai/blog/knowledge-distillation, [2] Gou et al. “Knowledge Distillation: A Survey”

Response-based knowledge distillation[1]

- **The main idea here is to mimic the predictions of the Teacher model**

- Therefore, we want to **minimise the difference between the predictions**
  - As the distillation loss is minimized over training, the Student will become better at making the same predictions as the Teacher[1]

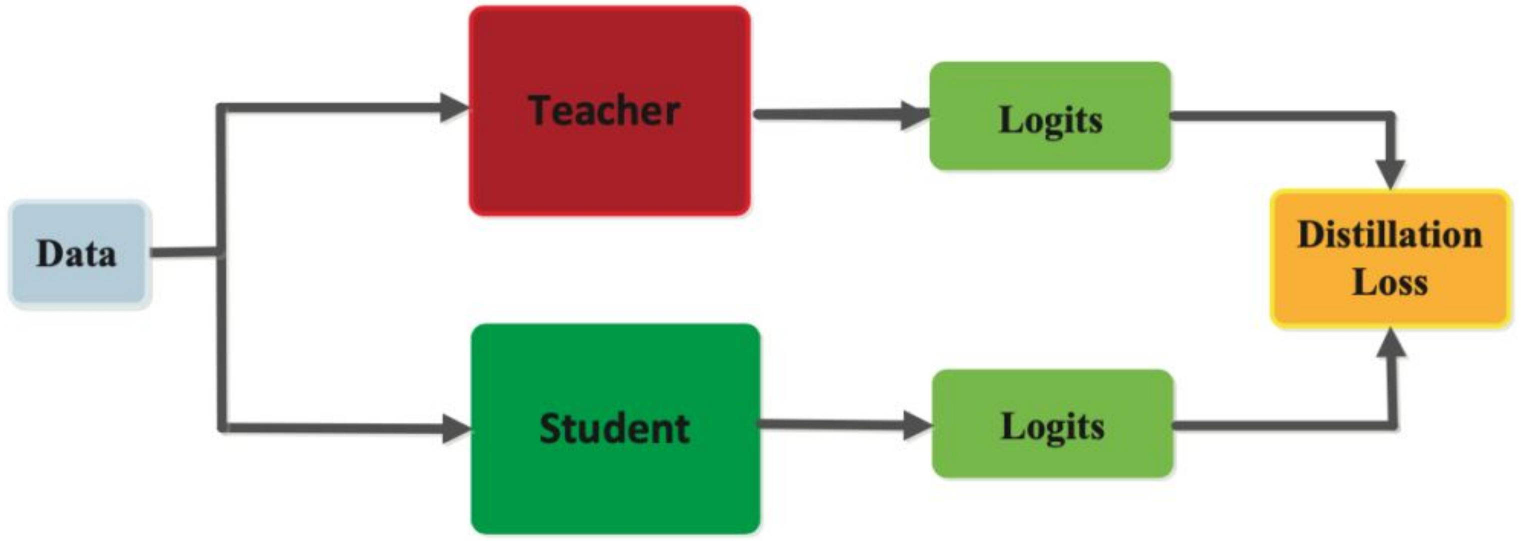


Figure - Response-based knowledge distillation (Image from [2])

Sources: [1] https://neptune.ai/blog/knowledge-distillation, [2] Gou et al. "Knowledge Distillation: A Survey"

Case Study: DistilBERT[1]

- **Context:** large-scale pre-trained language models became a basic tool in many natural language processing tasks
- **Problem:** there is an environmental cost of exponentially scaling these models' computational requirements and the growing computational and memory requirements of these models may hamper wide adoption
- **Proposal:** the student, DistilBERT, has the same general architecture of the Teacher, BERT[2], but with a reduced number of layers
  - Finding **the right initialization for the sub-network to converge** is and important element in their training procedure
  - Taking advantage of the common dimensionality between teacher and student networks, they initialize the student from the teacher by taking one layer out of two
- **Main Outcomes:** DistilBERT, a general-purpose pre-trained version of BERT, 40% smaller, 60% faster than BERT, that retains 97% of the language understanding capabilities
  - DistilBERT is a compelling option for edge applications

Sources: [1] Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter", **[2]** Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

INESCTEC

Selective attention: focusing on what is important

- **Attention is considered an important ability to flexibly control limited computational resources[1]**
  - In the machine learning context, this means that only the objects or elements of interest are focused while the background and other elements are discarded
  - This technique requires the addition of a selective attention network upstream of the existing AI system[2]

- **The main intuition comes from human perception:**
  - Humans do not tend to process a whole scene in its entirety at once
  - Instead, **humans focus attention selectively on parts of the visual space to acquire information when and where it is needed**, and combine information from different fixations over time to build up an internal representation of the scene[3], guiding future eye movements and decision making

- There are **lots** of strategies related to the implementation of attention mechanisms **(e.g., spatial attention, channel attention, multi-head attention, Transformers)**

Sources: [1] Grace Lindsay "Attention in Psychology, Neuroscience, and Machine Learning", **[2]** https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/, **[3]** Ronald A. Rensink. "The dynamic representation of scenes"

Case Study: Recurrent Models of Visual Attention[1]

- **Context:** deep neural networks have been advancing the state-of-the-art on challenging image classification and object detection datasets

- **Problem:** this excellent recognition accuracy comes at a high computational cost both at training and testing time
  - The large convolutional neural networks **typically used currently take days to train on multiple GPUs** even though the input images are downsampled to reduce computation

- **Hypothesis:** can we design a model that focuses the computational resources on parts of a scene?
  - This would save “bandwidth” as fewer “pixels” need to be processed

Sources: [1] Mnih et al. “Recurrent Models of Visual Attention”

Case Study: Recurrent Models of Visual Attention[1]

- **Proposal:** a novel framework for attention-based task-driven visual processing with neural networks
  - This model processes inputs sequentially, attending to different locations within the images one at a time
  - The information is combined iteratively from these fixations to build up a dynamic internal representation of the scene or environment
  - Instead of processing an entire image or even bounding box at once, at each step, the model selects the next location to attend to based on past information and the demands of the task
- **Important Outcomes:** both the number of parameters in our model and the amount of computation it performs can be controlled independently of the size of the input image

Sources: [1] Mnih et al. "Recurrent Models of Visual Attention"

INESCTEC

Low-rank factorisation: decomposing tensors

- **Let's recall some linear algebra concepts:**
  - **Rank:** the maximum number of linearly independent rows in a matrix *A* is called the **row rank** of *A*, and the maximum number of linearly independent columns in *A* is called the **column rank** of *A* (**the row rank of *A* = the column rank of *A***)[1]
  - **Matrix decomposition:** a way of reducing a matrix into its constituent parts and it is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself[2]
- **Low-rank factorisation:** this process uses matrix or tensor decomposition to estimate useful parameters[3]
  - The same can be done for layers of a neural network: a weight matrix with greater dimension and rank can be replaced with smaller dimension matrices through factorisation[4]
  - The key idea is that there exists latent structures in the data, by uncovering which we could obtain a compressed representation of the data[5]

Sources: [1] https://www.cliffsnotes.com/study-guides/algebra/linear-algebra/real-euclidean-vector-spaces/the-rank-of-a-matrix, [2] https://machinelearningmastery.com/introduction-to-matrix-decompositions-for-machine-learning/, [3] https://analyticsindiamag.com/model-compression-is-the-big-ml-flavour-of-2021/, [4] https://siddharthasaxena.medium.com/model-compression-the-what-why-and-how-of-it-bc1416f5b18f, [5] Lu and Yang "Notes on Low-rank Matrix Factorization"

Case Study: Low Rank and Sparse Decomposition[1]

- **Context:** deep models are usually over-parameterised, with redundancy often resulting in huge storage and computational resource demands

- **Opportunity:** this redundancy also provides a opportunity to compress a deep model without compromising accuracy provided that the deep network redundancy is properly removed

- **Hypothesis:** the weight matrix components usually reside in low-rank subspaces but some important entries are sparsely scattered in the weight matrices and mark the uniqueness of different filters
  - What if we are able to achieve a unified framework for deep compression by the low-rank and sparse decomposition?

Sources: [1] Yu et al. "On Compressing Deep Models by Low Rank and Sparse Decomposition"

Case Study: Low Rank and Sparse Decomposition[1]

- **Proposal:** to solve the low-rank and sparse decomposition with a feature map reconstruction term, the “greedy bilateral” scheme[2] is explored and exploited
  - This greedy scheme uses only QR[3] decompositions, random projections, and matrix multiplications, **which reduces computational complexity and is very efficient**

- **Discussion: low-rank and sparse decompositions usually result in smaller reconstruction errors of output features** compared with single value decomposition and pruning
  - This also motivates the idea that **weight matrices in deep models can be better represented by its low-rank and sparse components**

- **Main Outcomes:**
  - **Better initialisation** for the subsequent retraining, which helps the proposed model to achieve **high compression rates without loss of accuracy** for many popular models
  - **Save storage space**, which beats many recent methods using a single strategy

Sources: [1] Yu et al. “On Compressing Deep Models by Low Rank and Sparse Decomposition”, [2] Zhou and Tao “Greedy bilateral sketch, completion & smoothing”, [3] https://www.statlect.com/matrix-algebra/QR-decomposition

3. From theory to practice: can we code compressed models?

INESCTEC

Tiny Machine Learning: power in adverse conditions

- **Tiny Machine Learning (tinyML)** is broadly defined as a fast growing field of machine learning technologies and applications including hardware, algorithms and software capable of performing on-device sensor data analytics at extremely low power, typically in the mW range and below, and hence enabling a variety of always-on use-cases and targeting battery operated devices[1]

- With the **boost of key stakeholders** (e.g., Arduino, Google, Microsoft), this field is increasing its popularity[2]

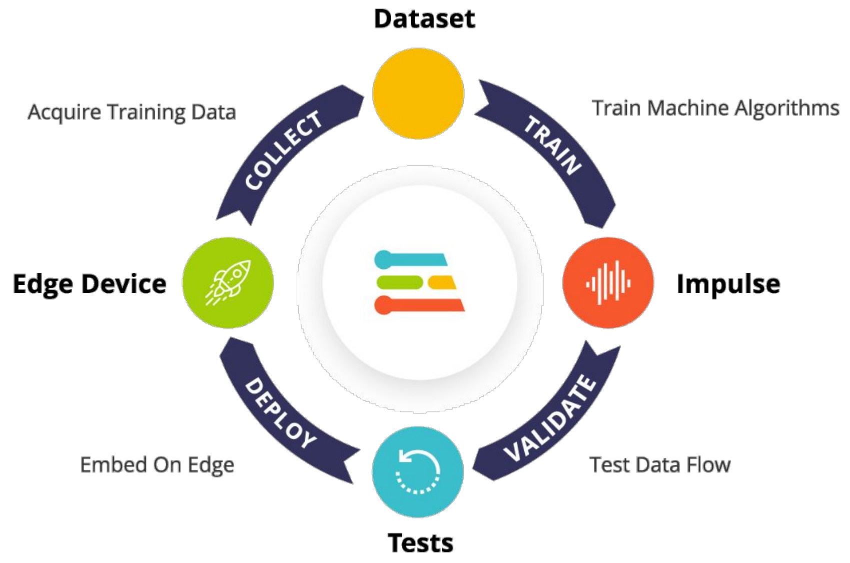
- For those who want to delve deeper into this field, there is a sustainable ecosystem and open-source materials for you to explore:
  - **TinyML Book** by Pete Warden and Daniel Situnayake[3]
  - **Tiny Machine Learning Open Education Initiative (TinyMLedu)**, an international group of academics and industry professionals working to improve global access to educational materials for the cutting-edge field of TinyML[4]

Sources: [1] https://www.tinyml.org, [2] https://towardsdatascience.com/tinyml-the-future-of-embedded-machine-learning-is-to-change-lives-for-the-better-87230668a08c, [3] https://tinymlbook.com, [4] https://tinyml.seas.harvard.edu

Edge Impulse: for beginner and advanced developers

- **Edge Impulse is a development platform** for machine learning on edge devices, with free plans for individual developers and industry plans for companies[1]

- This platform highlights the following use-cases:
    - **Industrial** (e.g., perform predictive maintenance)
    - **Infrastructure** (e.g., optimise facility usage, electric and water infrastructure)
    - **Conservation** (e.g., data-driven environmental health and animal conservation)
    - **Wearables** (e.g., monitor human health, improve worker safety)



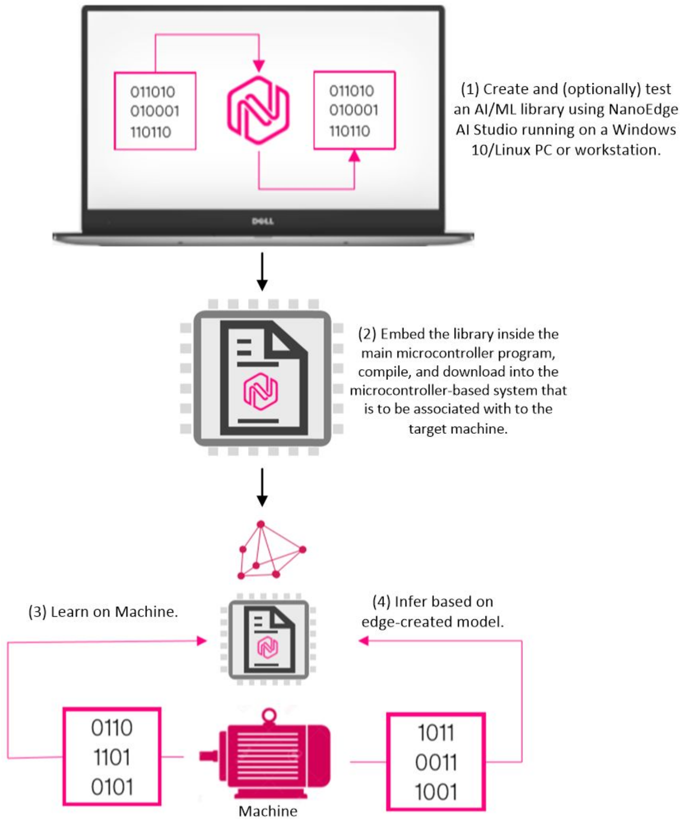
Sources: [1] https://www.edgeimpulse.com

INESCTEC

NanoEdge AI Studio: AI for embedded developers

- **NanoEdge AI Studio**[1] is a search engine desktop software for AI libraries, designed for embedded developers
  - This platform allows developers to find **a suitable AI library** for their embedded project[2]

- **It features two types of libraries[3]:**
  - **Anomaly detection library**
  - **Classification library**

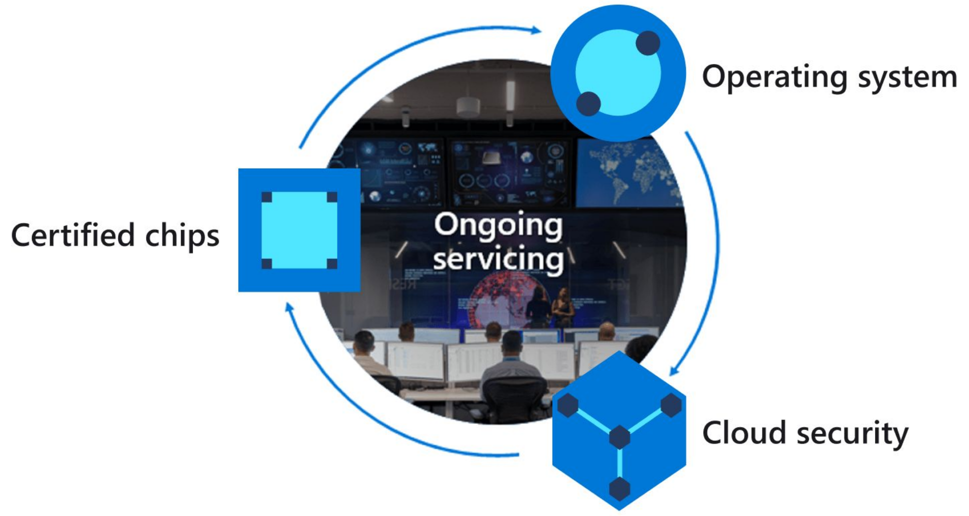


011010 010001 110110
011010 010001 110110
(1) Create and (optionally) test an AI/ML library using NanoEdge AI Studio running on a Windows 10/Linux PC or workstation.
(2) Embed the library inside the main microcontroller program, compile, and download into the microcontroller-based system that is to be associated with to the target machine.
(3) Learn on Machine.
(4) Infer based on edge-created model.
0110 1101 0101
Machine
1011 0011 1001

Sources: [1] https://cartesiam.ai, [2] https://medium.com/supplyframe-hardware/any-embedded-developer-can-create-ai-ml-systems-21a0ca19fdf2, [3] https://cartesiam-neai-docs.readthedocs-hosted.com

Azure Sphere: Edge AI from Microsoft

- **Azure Sphere[1] is an ecosystem that aims to securely connect microcontroller-powered devices from the silicon (hardware) to the cloud (software)**
  - **Interesting features: defense in depth** (multiple layers of protection to help guard devices against and respond to threats), **deployment flexibility** (that may help to secure existing equipment and build protection into new IoT investments), **over-the-air updates** (make it easy to add new features and improve performance throughout device lifecycles), **error reporting and automatic security updates** (help you stay ahead of new and evolving threats)



Sources: [1] https://azure.microsoft.com/en-us/services/azure-sphere/#overview

Edge AI is enhanced by cooperation: Part 1

- **Arduino released the "The Tiny Machine Learning Kit"[1], that results from a partnership with the "Tiny Machine Learning (TinyML)" specialization from EdX[2]**
  - The kit consists of a powerful board equipped with a microcontroller and a wide variety of sensors (movement, acceleration, rotation, temperature, humidity, barometric pressure, sounds, gestures, proximity, color, and light intensity)

- **Developers are able to explore algorithms and deep neural networks powered by TensorFlow Lite Microcontrollers[3]**

Sources: [1] https://store.arduino.cc/products/arduino-tiny-machine-learning-kit?selectedStore=eu, [2] https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning, [3] https://www.tensorflow.org/lite/microcontrollers

Edge AI is enhanced by cooperation: Part 2

- **Fraunhofer IMS has developed AIfES[1], a platform-independent and constantly growing machine learning library developed using the C programming language**
  - The program source code is reduced to a minimum, thus even the integration on a microcontroller including learning algorithms is possible

- **AIfES focus on two important use cases:**
  - **Privacy protection:** processing can take place offline on the device, hence, no sensitive data needs to be transferred
  - **Decentralized AI:** for instance, small intelligent embedded systems can process the data and provide calculated results to the next higher entity to avoid raw data overloading of the whole system

- **Fraunhofer IMS with AIfES and Arduino recently joined efforts to leverage the possibilities of Edge AI[2]**

Sources: [1] https://www.ims.fraunhofer.de/en/Business-Unit/Industry/Industrial-AI/Artificial-Intelligence-for-Embedded-Systems-AIfES.html, [2] https://github.com/Fraunhofer-IMS/AIfES_for_Arduino

4. Take-home messages and further readings

A (tentative) fair and accurate summary of this lecture

- **The transition of AI into embedded systems is currently growing and has attracted the eyes of both the Academia and the Industry[1], however, it has some challenges**
  - The talent gap with machine learning
  - Data gathering
  - Sustainability

- **The high performance of state-of-the-art machine learning learning is still related to the high number of parameters of these algorithms**
  - Some of the applications require that we pay attention to the computational requirements of the models we intend to deploy
  - Therefore, **model compression** is one of the possible strategies to achieve this goal

Sources: [1] https://www.g2.com/articles/embedded-ai

A (tentative) fair and accurate summary of this lecture

- **There are a few model compression techniques that deserve our attention:**
  - **Pruning:** consists of removing the parameters that, perhaps, will have low impact on performance
  - **Quantisation:** a process of mapping values from a large set to values in a smaller set, i.e., the output contains a smaller range of values compared to the input without losing much information in the process
  - **Knowledge distillation:** use an existing larger model (i.e., the Teacher) to train a smaller model (i.e., the Student)
  - **Selective attention:** focus in what is important
  - **Low-rank factorisation:** this process uses matrix or tensor decomposition to estimate useful parameters

- **Hardware and software stakeholders are currently leveraging several partnerships to achieve development environments that allow developers and/or researchers to test their own Edge AI models**

Further readings...

- Blier and Olivier "Do Deep Learning Models Have Too Many Parameters? An Information Theory Viewpoint"
- Lee et al. "Integrating machine learning in embedded sensor systems for Internet-of-Things applications"
- Andrade et al. "Overview of the state of the art in embedded machine learning"
- Deng et al. "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey"
- Choudhary et al. "A comprehensive survey on model compression and acceleration"
- Brandalero et al. "AITIA: Embedded AI Techniques for Embedded Industrial Applications"

Further readings...

- Wu et al. "Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation"
- Yang et al. "Quantization Networks"
- Szegedy et al. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning"
- Bucila et al. "Model Compression"
- Yuan et al. "Using a Flexible IoT Architecture and Sequential AI Model to Recognize and Predict the Production Activities in the Labor-Intensive Manufacturing Site"
- József Sütő "Embedded System-Based Sticky Paper Trap with Deep Learning-Based Insect-Counting Algorithm"
- Chin et al. "Smart-Object-Based Reasoning System for Indoor Acoustic Profiling of Elderly Inhabitants"

Edge AI - Lecture 1

TAIA - Advanced Topics on Artificial Intelligence

Tiago Filipe Sousa Gonçalves

tiago.f.goncalves@inesctec.pt | tiagofs@fe.up.pt

INESCTEC INSTITUTE FOR SYSTEMS AND COMPUTER ENGINEERING, TECHNOLOGY AND SCIENCE