Resource-Friendly Artificial Intelligence

Embedding Complex Intelligence into Constrained Devices

Andrea Agiollo

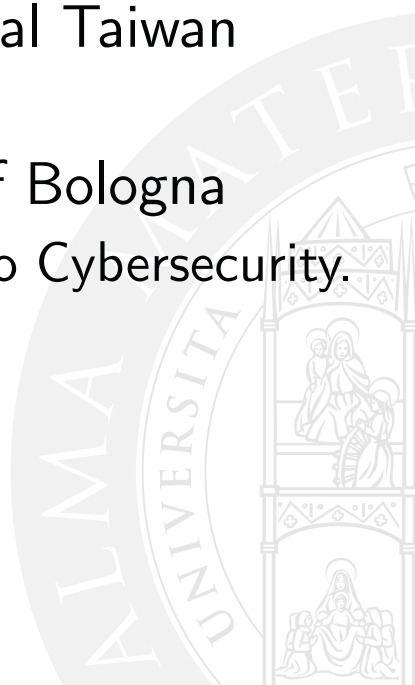
Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna
andrea.agiollo@unibo.it

January 17^{th}, 2023

Get to Know Me Better

- Bachelor Degree in Information Engineering at University of Padova
- Master Degree in ICT for Internet and Multimedia at University of Padova
- Master Degree in Communication Engineering at National Taiwan University
- PhD candidate in Computer Engineering at University of Bologna

  Research area span from resource-friendly AI solutions to Cybersecurity.



Outline

1. Introduction
2. Compressing Intelligence
   - Pruning
   - Neural Architecture Pruning
   - Quantization
   - Compression Shortcomings
3. Building Intelligence
   - Hand-Crafted Solutions
   - Neural Architecture Search
4. Injecting Intelligence
   - Structuring
   - Guided Learning
   - Embedding
   - SKI for Efficiency
5. Cybersecurity Entanglements

History of Neural Networks

Living in the NN era, everything solved through NNs (CV[17], NLP[5], Cybersecurity[13])

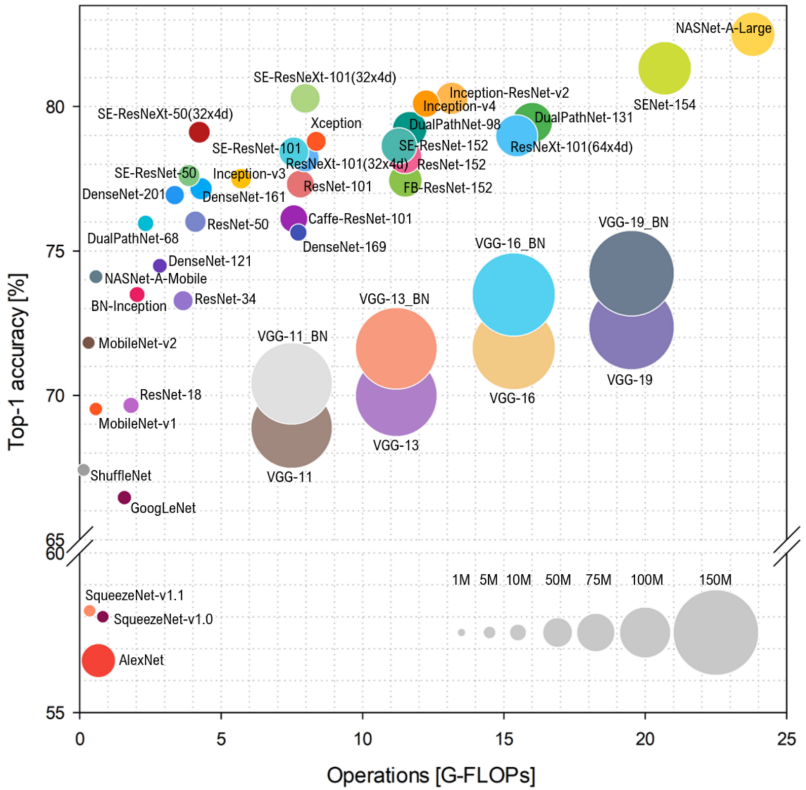
Advantages

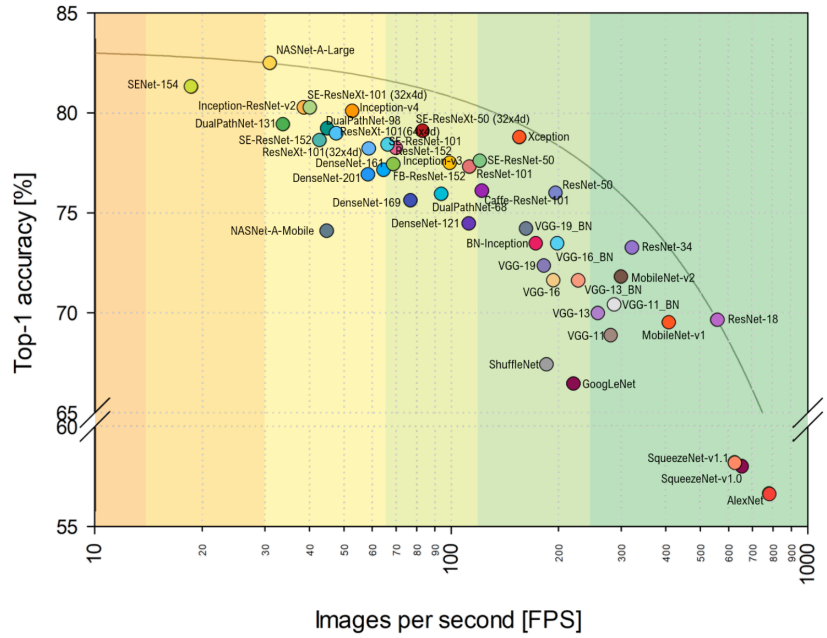
- Superior performance.
- Avoid cumbersome hand-crafted solutions.

Drawbacks

- Require high amount of processing power → Resource waste.
- Data requirement for every problem to tackle → Data science.
- Opaqueness issues → Explainable Artificial Intelligence.
- Security issues → How to fix NN bugs?

History of Neural Networks





Images courtesy of [6].

Proposed Solutions

Few effort to propose simpler NNs:

- MobileNet[9], EfficientNet[15], SqueezeNet[11] target mobile devices
- Up to 10 times more computationally efficient

Problem solved?

- Smartphones more and more powerful → Less need for resource-friendly NNs ✔
- Green solutions required ✖
- Still open issues for really constrained devices ✖

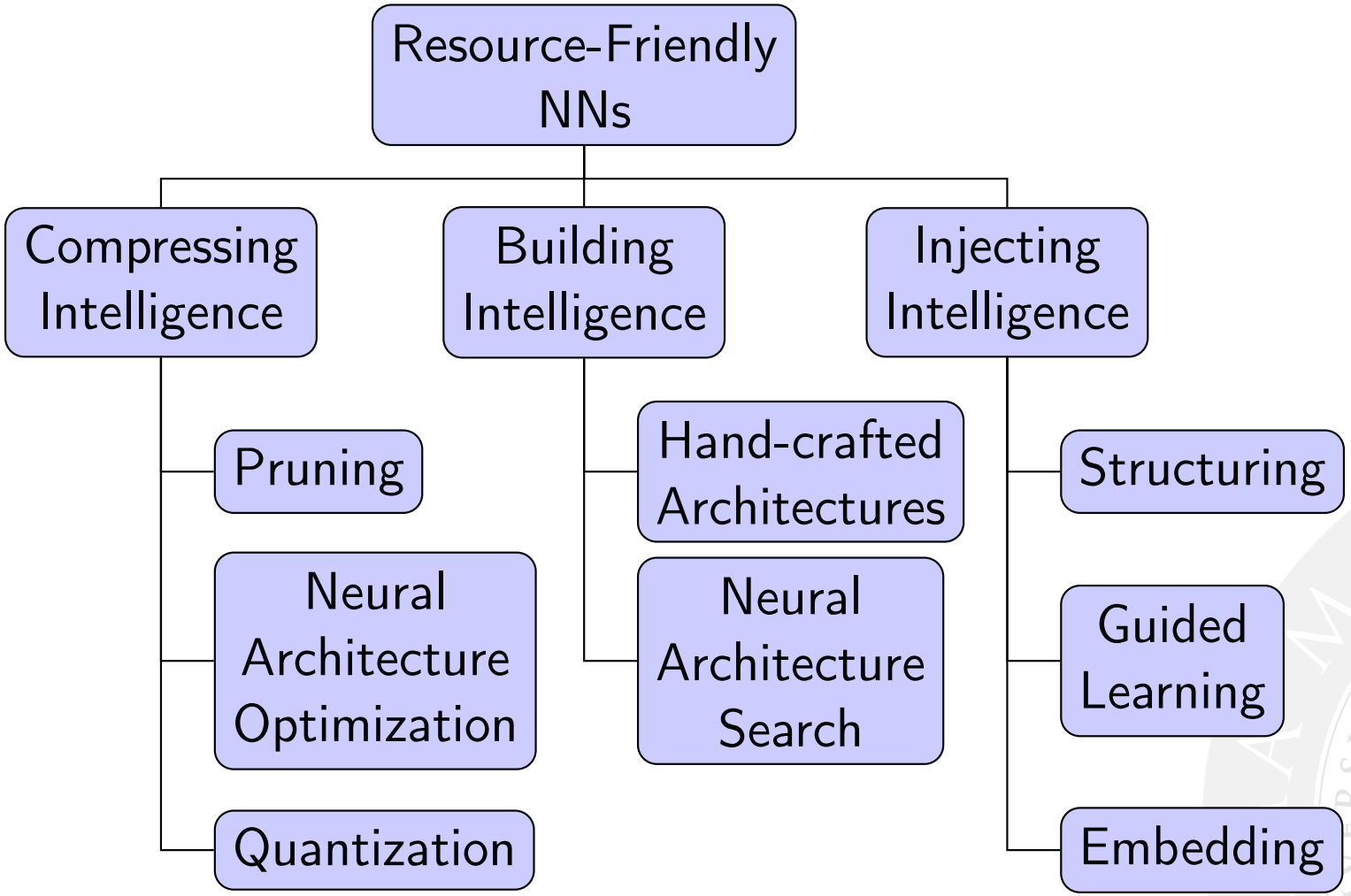
What About Truly Constrained Devices?

- Not all devices are smartphones
- $\sim 0.48 \times 10^9$ FLOPS (*ARM Cortex M7*) vs. $\sim 5 \times 10^{12}$ FLOPS (*Apple A13*)
- Industrial specific applications require truly constrained devices
- Price-bounded scenarios require resource constrained devices
- Interesting applications of NNs in hyper-constrained industrial scenarios

Problem

NNs deployment in constrained devices still open issue

Proposed Solutions: A Roadmap



Outline

1. Introduction
2. Compressing Intelligence
   - Pruning
   - Neural Architecture Pruning
   - Quantization
   - Compression Shortcomings
3. Building Intelligence
   - Hand-Crafted Solutions
   - Neural Architecture Search
4. Injecting Intelligence
   - Structuring
   - Guided Learning
   - Embedding
   - SKI for Efficiency
5. Cybersecurity Entanglements

Compressing NNs

- Cumbersome construction of efficient NN architectures from scratch
- Heavy NNs work fine for most tasks
- Heavy NNs are inefficient in most cases
- Not all tasks require fine-detailed knowledge of heavy NNs
- Can we reduce footprint and inefficiency of heavy NNs?

$$\Downarrow$$

Idea

Compress NN architectures via removal of unnecessary weights, connections, operations, etc.

NN Pruning

Intuition

Remove parameters – i.e., neurons – from a NN architecture

- Goal to maintain accuracy while increasing efficiency
- Remove neurons by their relevance in the NN
- Evaluate neurons importance

Open issues

- Is it easy/feasible to identify relevant neurons?
- How many neurons can be removed?
- What is the acceptable efficiency gain vs. performance degradation trade-off?

Few Catches on NN Pruning

Few things to highlight on NN pruning:

- Unstructured pruning – i.e., removal of single neurons – define sparse NN
- Sparse NN may have smaller footprint but same efficiency
- Common to re-train pruned NN $\rightarrow$ pruning quantity definition highly inefficient
- Cumbersome to identify stopping criteria for pruning $\rightarrow$ trial and error

$$\Downarrow$$

Is it worth it?

Most of the times it is, but with a catch...

Neural Architecture Pruning

Intuition

Is it possible/helpful to prune the NN architecture, rather than its single neurons

- Avoid sparse NN $\rightarrow$ decrease footprint *and* increase efficiency
- More relevant NN compression
- More complex to identify layers to remove
- What is layers role in NN?
- XAI implications

Caveat

This represents an open research question

Neural Network Quantization

- NNs represented via tensors
- Tensors are stored as floating point precisions
- Precision may be decreased in some applications to increase efficiency

$$\Downarrow$$

Definition

Quantization represent a technique for converting NN tensors to lower bit-width than floating point precision. [8]

Why Quantization?

Benefits:

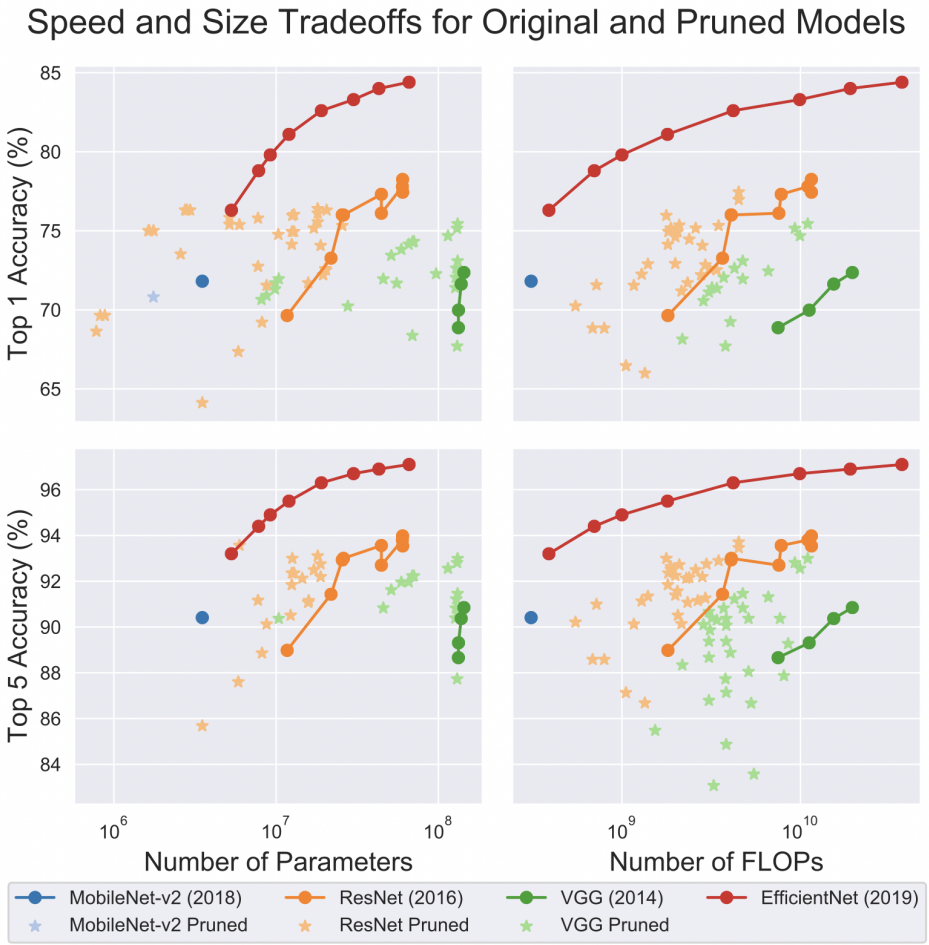
- Customisable compression level depending on the application
- Does not sacrifice too much performance
- Enables different hardware platforms even hyper-constrained

Caveat

Although simple idea, application not straightforward

- Selection of clipping range – i.e., $[0, 1] \rightarrow [a, b]$ [18]
- Symmetric vs. Asymmetric quantization $\rightarrow$ better $a = b$ or $a \neq b$
- Quantization granularity [14] $\rightarrow$ layer-wise or group-wise quantization
- Quantization-aware training [10] or not [12]

Compression vs Architecture Changes



Better to change architecture

Modifying the overall architecture to make it more efficient is usually a better solution

Image courtesy of [7].

Outline

1. Introduction
2. Compressing Intelligence
   - Pruning
   - Neural Architecture Pruning
   - Quantization
   - Compression Shortcomings
3. Building Intelligence
   - Hand-Crafted Solutions
   - Neural Architecture Search
4. Injecting Intelligence
   - Structuring
   - Guided Learning
   - Embedding
   - SKI for Efficiency
5. Cybersecurity Entanglements

On the Efficiency of NNs

- Are deep/wide NNs so necessary?
- Quest for efficient combinations of NN operations
- Identify performance/efficiency trade-off in NNs



Idea

Design simple NN architectures through smart combination of NN operations

Handcrafted solutions

Separable convolution[9], Depth-wise convolution[9], Squeeze and expand dimensionality tricks[11], Highway networks, etc.

Automating Search for Efficient NNs

- Hand-crafting NN architectures is costly
- Human experience is limited and may be misleading



Neural Architecture Search (NAS)

Meta-learning techniques to automatically design NN architectures.
NAS methods vary depending on:

search space: define NNs that can be designed

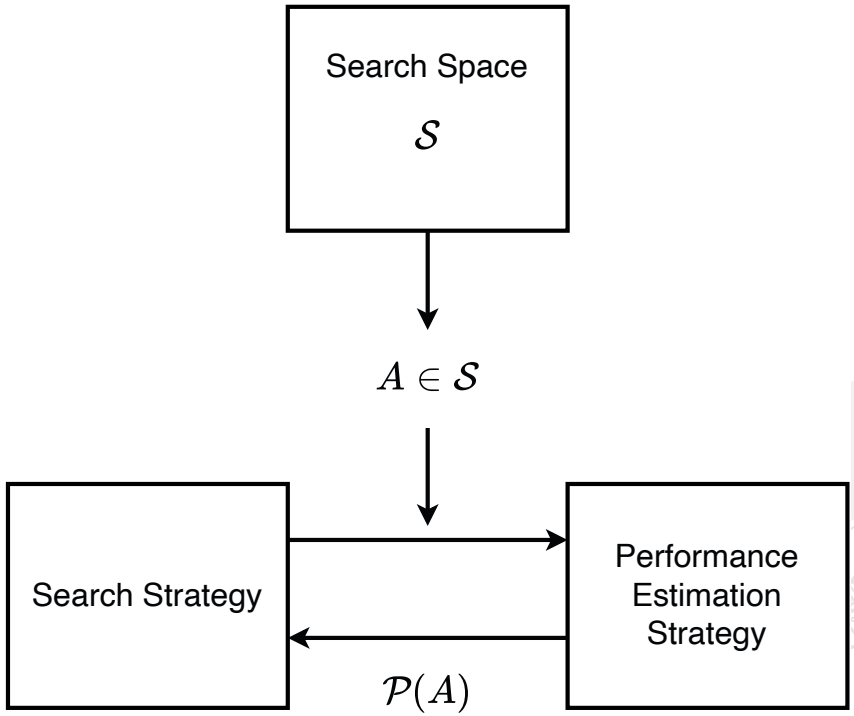
search strategy: define exploration approach used over search space

performance estimation strategy: define how to evaluate performance of designed NNs

NAS Workflow

Steps:

1. Sample strategy samples NNs from search space
2. Sampled NNs are evaluated using performance evaluation strategy
3. Sampling strategy is updated depending on obtained performance
4. Repeat 1-3 up until satisfaction criteria are met



NAS for Efficiency

Intuition

Search space and performance estimation strategy helpful to identify efficient NNs[1]*.

Shallow2Deep

- Limit NN depth $\rightarrow$ latency improvements.
- Promote local variability to reduce complexity.
- Simple evolutionary strategy to select fittest architectures.

Beyond NAS

NAS drawbacks:

- Very inefficient $\rightarrow$ trial and error approach
- Task dependent $\rightarrow$ one NAS for each data-driven task
- Learning to search vs. learning to build

$$\Downarrow$$

Intuition

Meta-learning approach to learn design criteria rather than search criteria

Graph Neural Network to Generate Neural Networks[3]*

Use Graph Neural Networks to learn to generate NN architectures

- GNN can model complex relations in graph structured data
- Starting from well-known NN architecture samples train generator model
- Generator aims at producing powerful NNs
- Three-way generative adversarial approach
- Results show we can learn to generate architectures
- No exploration phase
- Generator intrinsically has knowledge of design criteria

Outline

1. Introduction
2. Compressing Intelligence
   - Pruning
   - Neural Architecture Pruning
   - Quantization
   - Compression Shortcomings
3. Building Intelligence
   - Hand-Crafted Solutions
   - Neural Architecture Search
4. Injecting Intelligence
   - Structuring
   - Guided Learning
   - Embedding
   - SKI for Efficiency
5. Cybersecurity Entanglements

Definition

- Not all knowledge is learnable from data
- Data may not represent desirable knowledge
- Knowledge learnt from data differ from desired knowledge
- Is it possible to combine data-knowledge and a-priori knowledge?



Definition: Symbolic Knowledge Injection

We define SKI as any *algorithmic* procedure affecting how sub-symbolic predictors draw their inferences in such a way that predictions are either *computed* as a function of, or made *consistent* with, some *given* symbolic knowledge.

Meaning of Symbolic

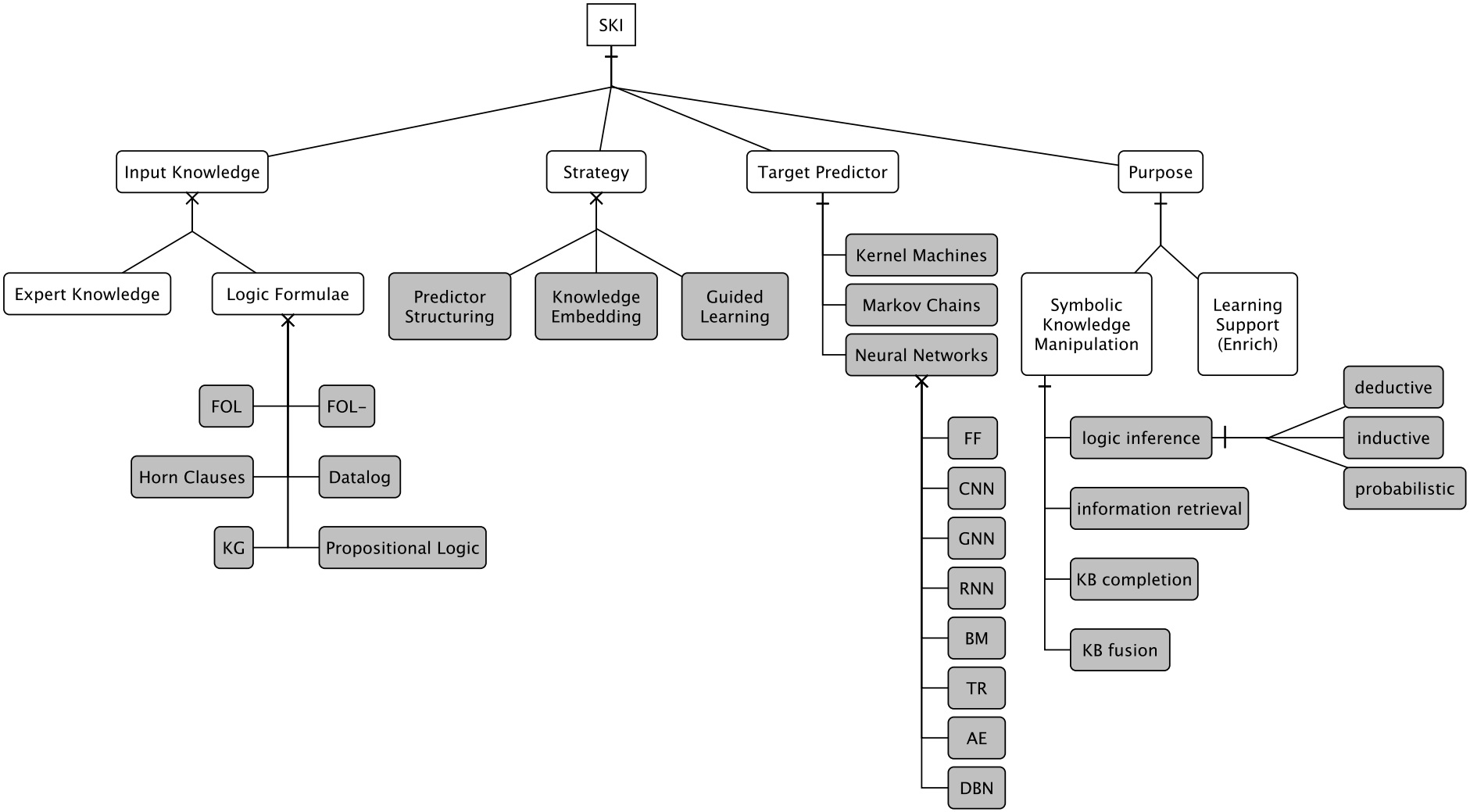
According to [16], symbolic representations of knowledge

- Set of meaningful symbols
- Symbols can be combined in (possibly) infinitely many ways
- Combinations follow precise syntactical rules
- Combinations can be assigned with meaning
  - ie symbols mapped into entities and combinations are relations between them

Notable Example

- Formal Logic, Horn Clauses, Knowledge Graphs, etc.

Taxonomy of SKI



SKI via Structuring

Definition

A sub-symbolic model (or part of it) is built to mirror symbolic knowledge via its own internal structure.

- NN architecture mimics behaviour of a-priori knowledge to be injected
- A-priori knowledge can be extended or constrained during data-driven training

Example

NN neurons represent logic predicates, while synapses map logic connectives among them

SKI via Guided Learning

Definition

A-priori knowledge used to steer NN learning process.

- Penalisation of inconsistent behaviours w.r.t. the desired knowledge
- Incentivisation of consistent behaviours w.r.t. the desired knowledge
- Alteration of training loss functions to increase (decrease) loss for inconsistent (consistent) behaviour

Example

Constraining term in loss function representing unfeasible relationships between entities

SKI via Embedding

Definition

Symbolic Knowledge is converted into numeric-array form in some meaningful way.
Numeric-form knowledge is provided as "ordinary" input for the NN.

- Numeric data is generated out of symbolic knowledge
- Numeric-form knowledge representation called *embedding*

Example

Knowledge graph embedded into training samples

SKI for Efficiency

Intuition

SKI focus of XAI related works, but helpful from efficiency perspective[4]*:

- Injected knowledge lifts part of the learning burden
- Reduce notions to be learnt data-drivenly ⇒ Data efficiency
- Reduce computations for training and running ⇒ Energy efficiency
- Remove unnecessary computations ⇒ Latency
- Possibility of shrinking sub-symbolic model ⇒ Memory efficiency

SKI Quality of Service definition

$$M_i = \Psi\left(\mathcal{N}^{ski}\left(\mathcal{K}, \mathcal{I}, \tau\right)\right) - \Psi(\mathcal{N}(\tau)) \quad \text{s.t.} \quad \mathcal{P}(\mathcal{N}^{ski}, \tau) \geq \mathcal{P}(\mathcal{N}, \tau) \tag{1}$$

Outline

1. Introduction
2. Compressing Intelligence
   - Pruning
   - Neural Architecture Pruning
   - Quantization
   - Compression Shortcomings
3. Building Intelligence
   - Hand-Crafted Solutions
   - Neural Architecture Search
4. Injecting Intelligence
   - Structuring
   - Guided Learning
   - Embedding
   - SKI for Efficiency
5. Cybersecurity Entanglements

Why Efficiency Matters?

- Security of constrained connected devices – e.g., Internet of Things – require memory efficiency
- Real-time heavy load analysis – e.g., border router traffic – require latency efficiency
- Security of battery powered devices require energy efficiency
- Few-shot attack detection require data efficiency

Intuition

Most of such scenarios rely on statistical analysis to allow defense deployment[2]*. However, efficient NNs would enable smarter, more effective solutions.

SKI for Cybersecurity

Cases where:

- Few rules are known to be true, but do not cover the totality of cases $\rightarrow$ cybersecurity mechanism to build on top of such rules and learn more.
- Few rules are known to be false, thus should be discarded by default $\rightarrow$ cybersecurity mechanism to avoid such rules and learn relevant ones.

Examples

- Smart firewalls $\rightarrow$ inject *if IP=115.17.89.62 then ALLOW*, and learn more.
- Avoid IP or port-based rules in smart malware detection $\rightarrow$ penalise unreliable rules – e.g., *if IP=115.17.89.62 then MALWARE* – and learn from data.

SKI for NN Security

NN are intrinsically black-boxes, and suffer:

- Bugs, depending on scenario and data used to train
- Bias, depending data used to train
- How to secure NN against bugs, bias, etc.?

Intuition

SKI coupled with its counterpart – i.e., Symbolic Knowledge Extraction – allows for NN debugging, treating them as common software.

Resource-Friendly Artificial Intelligence

Embedding Complex Intelligence into Constrained Devices

Andrea Agiollo

Dipartimento di Informatica – Scienza e Ingegneria (DISI)
Alma Mater Studiorum – Università di Bologna
andrea.agiollo@unibo.it

January 17^{th}, 2023

References I

[1] Andrea Agiollo, Giovanni Ciatto, and Andrea Omicini.
Shallow2deep: Restraining neural networks opacity through neural architecture search.
In *Explainable and Transparent AI and Multi-Agent Systems - Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3-7, 2021, Revised Selected Papers*, volume 12688 of *Lecture Notes in Computer Science*, pages 63–82. Springer, 2021.

[2] Andrea Agiollo, Mauro Conti, Pallavi Kaliyar, Tsung-Nan Lin, and Luca Pajola.
DETONAR: detection of routing attacks in rpl-based iot.
IEEE Transaction on Network and Service Management, 18(2):1178–1190, 2021.

[3] Andrea Agiollo and Andrea Omicini.
GNN2GNN: graph neural networks to generate neural networks.
In *Uncertainty in Artificial Intelligence, Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, UAI 2022, 1-5 August 2022, Eindhoven, The Netherlands*, volume 180 of *Proceedings of Machine Learning Research*, pages 32–42. PMLR, 2022.

[4] Andrea Agiollo, Andrea Rafanelli, and Andrea Omicini.
Towards quality-of-service metrics for symbolic knowledge injection.
In *WOA 2022 – 23rd Workshop "From Objects to Agents"*, volume 3261 of *CEUR Workshop Proceedings*, pages 30–47. Sun SITE Central Europe, RWTH Aachen University, 2022.

References II

[5] Basemah Alshemali and Jugal Kalita.
Improving the reliability of deep neural networks in NLP: A review.
Knowledge Based Systems, 191:105210, 2020.

[6] Simone Bianco, Rémi Cadène, Luigi Celona, and Paolo Napoletano.
Benchmark analysis of representative deep neural network architectures.
IEEE Access, 6:64270–64277, 2018.

[7] Davis W. Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John V. Guttag.
What is the state of neural network pruning?
In *Proceedings of Machine Learning and Systems 2020, MLSys 2020, Austin, TX, USA, March 2-4, 2020*. mlsys.org, 2020.

[8] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer.
A Survey of Quantization Methods for Efficient Neural Network Inference.
CoRR, abs/2103.13630, 2021.
arXiv: 2103.13630.

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam.
Mobilenets: Efficient convolutional neural networks for mobile vision applications.
CoRR, abs/1704.04861, 2017.

References III

[10] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio.
Binarized Neural Networks.
In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4107–4115, 2016.

[11] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer.
Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size.
CoRR, abs/1602.07360, 2016.

[12] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu.
BRECQ: Pushing the Limit of Post-Training Quantization by Block Reconstruction.
In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[13] Junyang Qiu, Jun Zhang, Wei Luo, Lei Pan, Surya Nepal, and Yang Xiang.
A survey of android malware detection with deep neural models.
ACM Computing Surveys, 53(6):126:1–126:36, 2021.

References IV

[14] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer.
Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT.
In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8815–8821. AAAI Press, 2020.

[15] Mingxing Tan and Quoc V. Le.
Efficientnet: Rethinking model scaling for convolutional neural networks.
In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[16] Tim van Gelder.
Why distributed representation is inherently non-symbolic.
In Georg Dorffner, editor, *Konnektionismus in Artificial Intelligence und Kognitionsforschung. Proceedings 6. Österreichische Artificial Intelligence-Tagung (KONNAI), Salzburg, Österreich, 18. bis 21. September 1990*, volume 252 of *Informatik-Fachberichte*, pages 58–66. Springer, 1990.

References V

[17] Zhengwei Wang, Qi She, and Tomás E. Ward.
Generative adversarial networks in computer vision: A survey and taxonomy.
ACM Computing Surveys, 54(2):37:1–37:38, 2022.

[18] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius.
Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation.
CoRR, abs/2004.09602, 2020.
arXiv: 2004.09602.

