On the Quest for Resource-Efficient AI

Lessons Learnt from Machine Learning and Neuro-Symbolic Integration

Andrea Agiollo - 14/04/2025

TU Delft

Who am I?

1. BSc in Information Engineering at University of Padova, Italy



Who am I?

1. BSc in Information Engineering at University of Padova, Italy
    A. Erasmus Exchange in Aberdeen, Scotland (UK)





Who am I?

1. BSc in Information Engineering at University of Padova, Italy
    A. Erasmus Exchange in Aberdeen, Scotland (UK)
2. MSc in Telecommunications Engineering at University of Padova, Italy



Who am I?

1. BSc in Information Engineering at University of Padova, Italy
    A. Erasmus Exchange in Aberdeen, Scotland (UK)
2. MSc in Telecommunications Engineering at University of Padova, Italy
    A. Double Degree Program in Taipei, Taiwan



Who am I?

1. BSc in Information Engineering at University of Padova, Italy
    A. Erasmus Exchange in Aberdeen, Scotland (UK)
2. MSc in Telecommunications Engineering at University of Padova, Italy
    A. Double Degree Program in Taipei, Taiwan
3. PhD at University of Bologna, Italy
    - Resource-constrained AI solutions
    - Neuro-Symbolic approaches for resource minimization
    - AI for cybersecurity

    A. Visiting PhD @ TU Delft
    B. Visiting PhD @ Purdue University



Who am I?

1. BSc in Information Engineering at University of Padova, Italy
   A. Erasmus Exchange in Aberdeen, Scotland (UK)
2. MSc in Telecommunications Engineering at University of Padova, Italy
   A. Double Degree Program in Taipei, Taiwan
3. PhD at University of Bologna, Italy
   - Resource-constrained AI solutions
   - Neuro-Symbolic approaches for resource minimization
   - AI for cybersecurity

   A. Visiting PhD @ TU Delft
   B. Visiting PhD @ Purdue University
4. PostDoc at TU Delft, The Netherlands
   - Neuro-Symbolic approaches for cybersecurity
   - Security of Neuro-Symbolic integration
   - Security of Machine Learning



Talk Outline

0. Introduction to resource-efficient AI

1. Efficientization as optimization of NN architecture

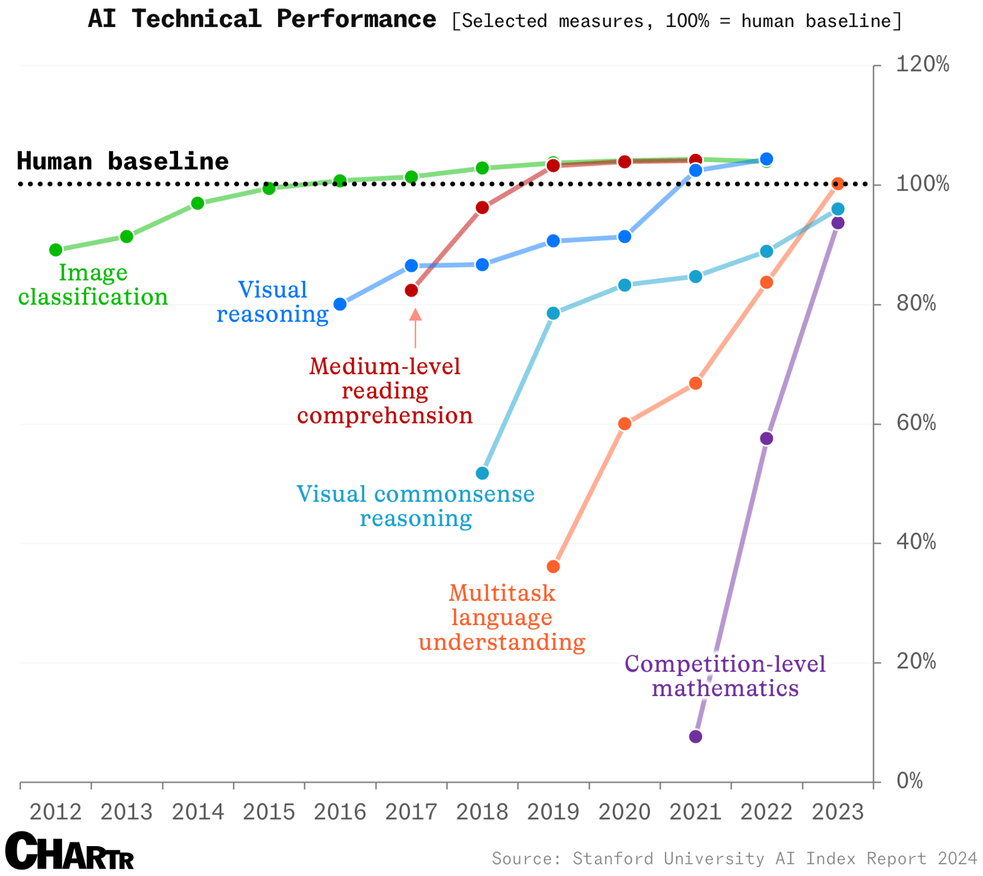
2. Efficientization as data optimization

3. Exploring the use of Neuro-Symbolic integration for efficientization

4. Future directions

0. Why Resource-efficient AI?

- Ever-increasing abundance of data
- Open data
- Steady increase of available computational resources
- Affordability of computational resources
- Machine Learning: Automatic optimization process from available data
- Broad field of applications
- Super-human performance



0. Why Resource-efficient AI?

The *world* with LLMs



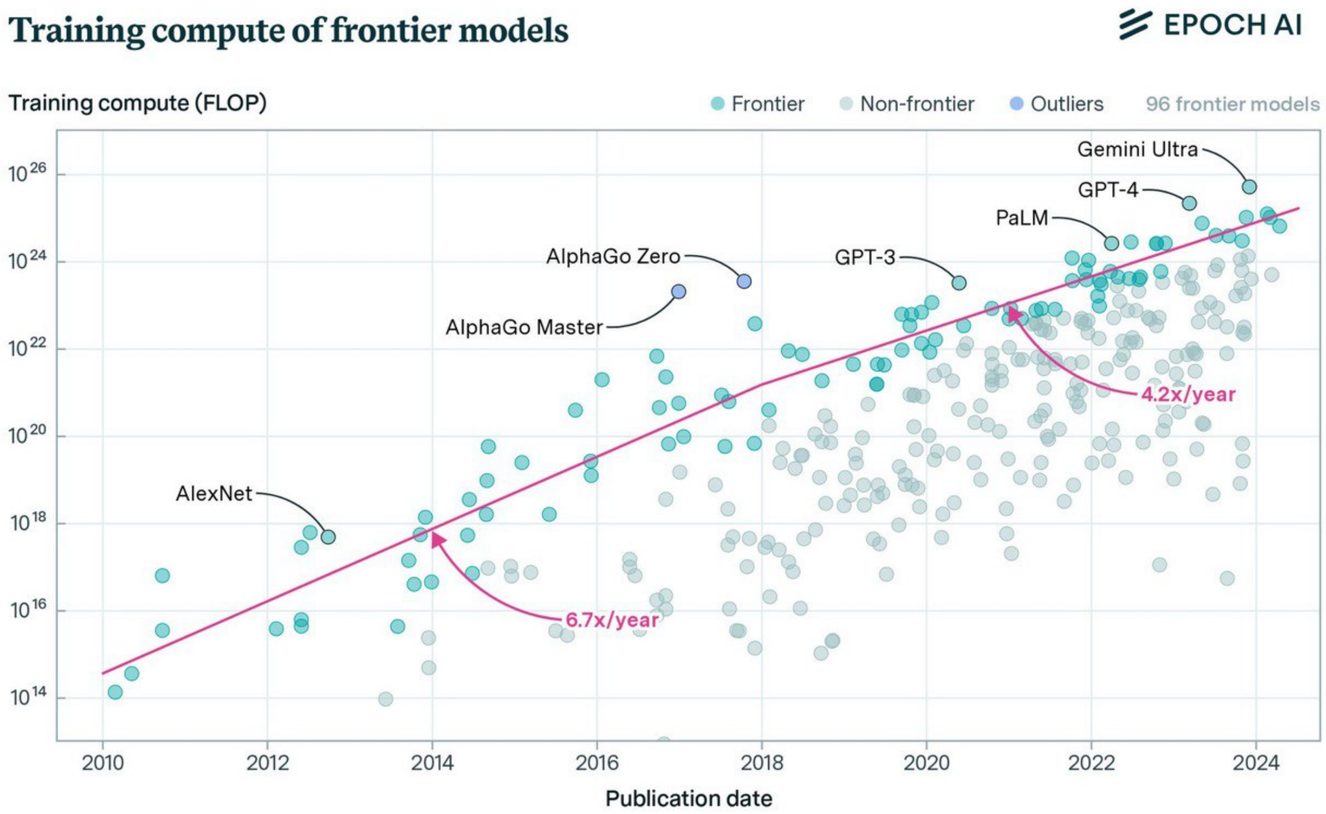
The *world* without LLMs



0. Why Resource-efficient AI?

SOTA NNs are *big*:

- GPT-3 $\simeq$ 175 billion parameters
- GPT-4 $\simeq$ 1.76 trillion parameters
- Performance improvement over parameters follow smoothly broken power law scaling

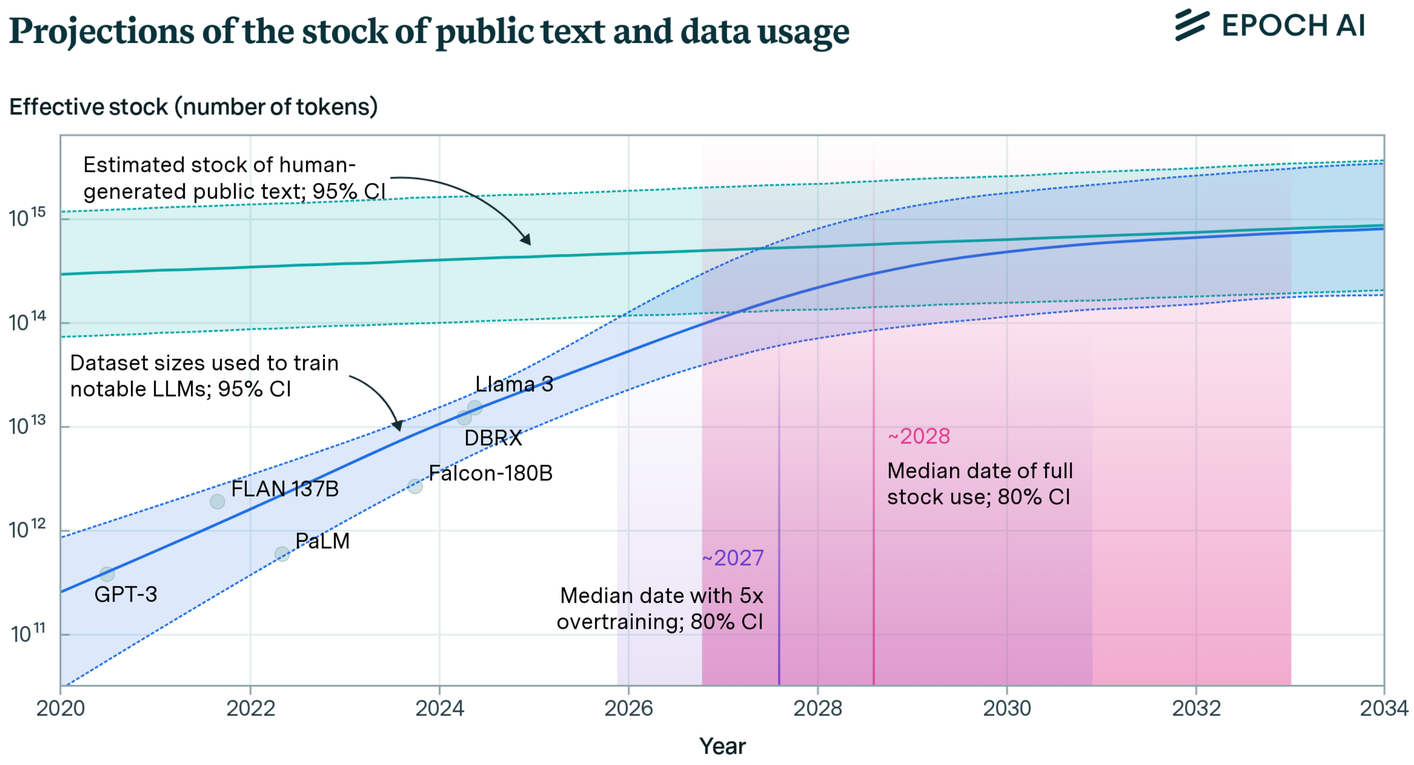


- Sardana, Nikhil, et al. "Beyond chinchilla-optimal: Accounting for inference in language model scaling laws." arXiv preprint arXiv:2401.00448 (2023).
- Hoffmann, Jordan; Borgeaud, Sebastian; Mensch, Arthur; Buchatskaya, Elena; Cai, Trevor; Rutherford, Eliza; Casas, Diego de Las; Hendricks, Lisa Anne; Welbl, Johannes; Clark, Aidan; Hennigan, Tom; Noland, Eric; Millican, Katie; Driessche, George van den; Damoc, Bogdan (2022-03-29). "Training Compute-Optimal Large Language Models". arXiv:2203.15556.

0. Why Resource-efficient AI?

SOTA NNs are *data-hunger*:

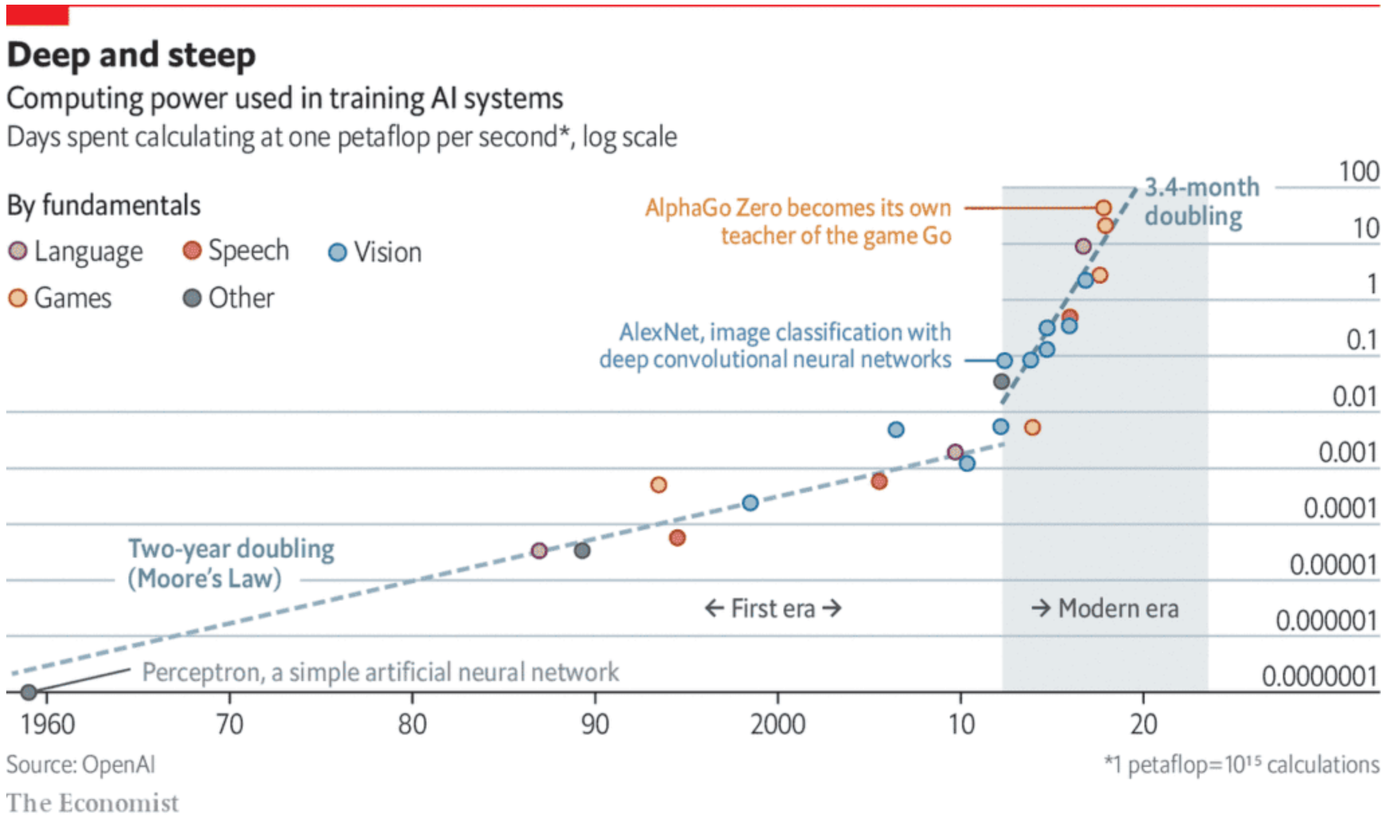
- Llama 3 $\simeq$ 15 trillion tokens
- GPT-4 $\simeq$ 6.5 trillion tokens
- Human at age 20 $\simeq$ 200 million tokens
- High-quality *public* text upper limit $\simeq$ 40 to 90 trillion tokens
- Google with all gmails could get up to 400 trillion tokens



0. Why Resource-efficient AI?

SOTA NNs are *energy-hunger*:

- 1000 inferences of efficient text generation ≃ 9% of smartphone charge
- 1000 inferences of image generation ≃ 522 smartphone charges
- 1 inference of image generation ≃ 50% of smartphone charge



- Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2024. Power Hungry Processing: Watts Driving the Cost of AI Deployment? In Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency (FAccT '24). Association for Computing Machinery, New York, NY, USA, 85–99. https://doi.org/10.1145/3630106.3658542

0. Why Resource-efficient AI?

SOTA NNs are *expensive*:

- ChatGPT-3 cost $2 to $4 million (*2020*)
- DALL-E cost $335,000 (*2021*)
- ChatGPT-4 cost $41 to $78 million (*2023*)
- Gemini up to $191 million (*2023*)

Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$\text{Transformer}_{base}$	P100x8	1415.78	12	27	26	$41–$140
Transformer_{big}	P100x8	1515.43	84	201	192	$289–$981
ELMo	P100x3	517.66	336	275	262	$433–$1472
BERT_{base}	V100x64	12,041.51	79	1507	1438	$3751–$12,571
BERT_{base}	TPUv2x16	—	96	—	—	$2074–$6912
NAS	P100x8	1515.43	274,120	656,347	626,155	$942,973–$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055–$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902–$43,008

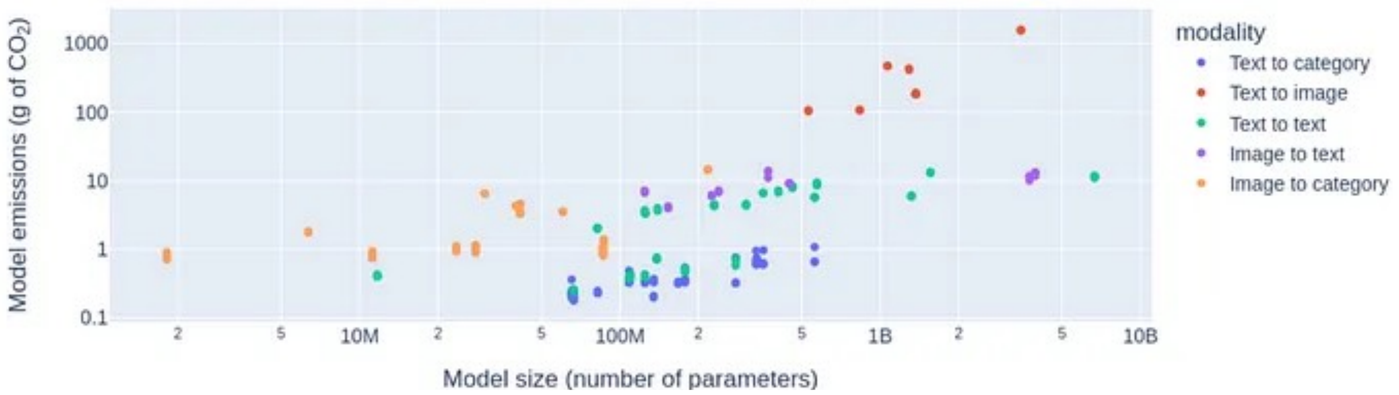
Table 3: Estimated cost of training a model in terms of CO_2 emissions (lbs) and cloud compute cost (USD).[7] Power and carbon footprint are omitted for TPUs due to lack of public information on power draw for this hardware.

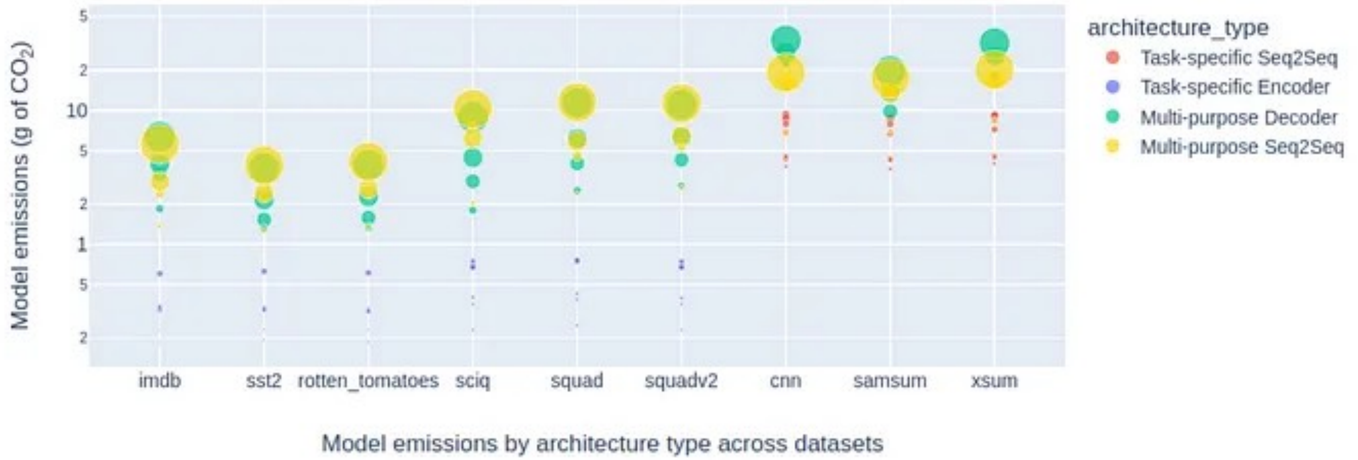
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP." arXiv preprint arXiv:1906.02243 (2019).
- Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. "Estimating the carbon footprint of bloom, a 176b parameter language model." Journal of Machine Learning Research 24.253 (2023): 1-15.

0. Why Resource-efficient AI?

SOTA NNs are *polluting*:

- A100 NVIDIA GPU, average carbon intensity of 297.6 grams CO2 eq per kWh
- Training GPT-3 (175B) ≃ estimated 502 tonnes of CO2 eq
- GPT-4 (1.76T) up to 10 times larger environmental impact
- End-to-end model optimization emit up to 626,000 pounds of CO2 eq ≃ 5x lifetime emissions american car





- Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP." arXiv preprint arXiv:1906.02243 (2019).
- Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. "Estimating the carbon footprint of bloom, a 176b parameter language model." Journal of Machine Learning Research 24.253 (2023): 1-15.

0. Why Resource-efficient AI?

The *planet* with LLMs



The *planet* without LLMs



0. Why Resource-efficient AI?

- **Solution**: *Resource-efficient AI*
- **Challenge**: *Resource-efficient AI*
- Pillars of Resource-efficient AI:
    1. Less computations
    2. Less data
    3. Less energy

0. Efficientization state-of-the-art

1. Less computations:

   - *Neural network pruning*: systematically removing part non-relevant parameters from existing NN model
   - *Knowledge distillation*: transferring knowledge learnt from large teacher model to small student model
   - *Neural network quantization*: storing weights tensors in lower bit precision than usual 16 or 32-bit precision
   - *Neural architecture search*: automating construction of effective NN structures

- Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A aurvey. Neurocomputing, 461:370–403, 2021.
- Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. Knowledge distillation: A survey. International Journal of Computer Vision, 129(6):1789–1819, 2021.
- Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. J. Mach. Learn. Res., 20:55:1–55:21, 2019.

0. Efficientization state-of-the-art

2. Less data:

- *Dataset summarization*: identifying most informative subset of data, so that model trained on selected subset achieve similar generalization to original
- *Dataset distillation*: starting from large dataset generate much smaller one containing similar information so that trained model reach similar performance
- *Active learning*: iteratively select most informative examples from unlabeled dataset to be labeled by oracle (e.g., human annotator) during model training
- *Transfer learning*: leveraging knowledge learned from one task or domain to apply to another task or domain reducing number of data required for optimization

- Dan Feldman. "Core-sets: Updated survey." Sampling techniques for supervised or unsupervised tasks (2020): 23-44.
- Shiye Lei, and Dacheng Tao. "A comprehensive survey of dataset distillation." IEEE Transactions on Pattern Analysis and Machine Intelligence 46.1 (2023): 17-32.
- Pengzhen Ren, et al. "A survey of deep active learning." ACM computing surveys (CSUR) 54.9 (2021): 1-40.
- Fuzhen Zhuang, et al. "A comprehensive survey on transfer learning." Proceedings of the IEEE 109.1 (2020): 43-76.

0. Efficientization state-of-the-art

3. Less energy:

   - *Less computations*: reducing model complexity positive feedback on energy consumption (with grain of salt)
   - *Less data*: reducing data required to optimize model positive feedback on energy consumption
   - *Efficient hardware support*: designing hardware components tailored for low-precision highly efficient numerical processing over tensors (e.g., TPUs)
   - *Distributed learning*: splitting learning task and distribute across several devices

- Shahriya Zaman, et al. "Custom hardware architectures for deep learning on portable devices: A review." IEEE Transactions on Neural Networks and Learning Systems 33.11 (2021): 6068-6088.
- Joost Verbraeken, et al. "A survey on distributed machine learning." Acm computing surveys (csur) 53.2 (2020): 1-33.

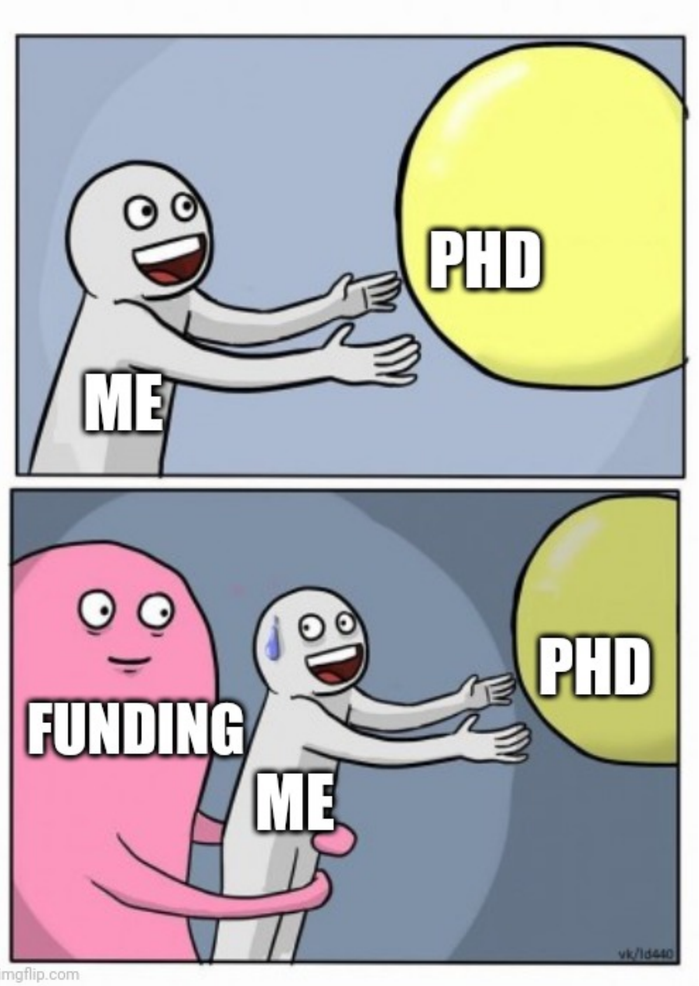
0. Focus

Topics:

1. ***Neural architecture search*** (less computations)
2. ***Dataset summarization and active learning*** (less data)
3. ***Neuro-Symbolic integration for efficiency*** (less data)

Reasons:

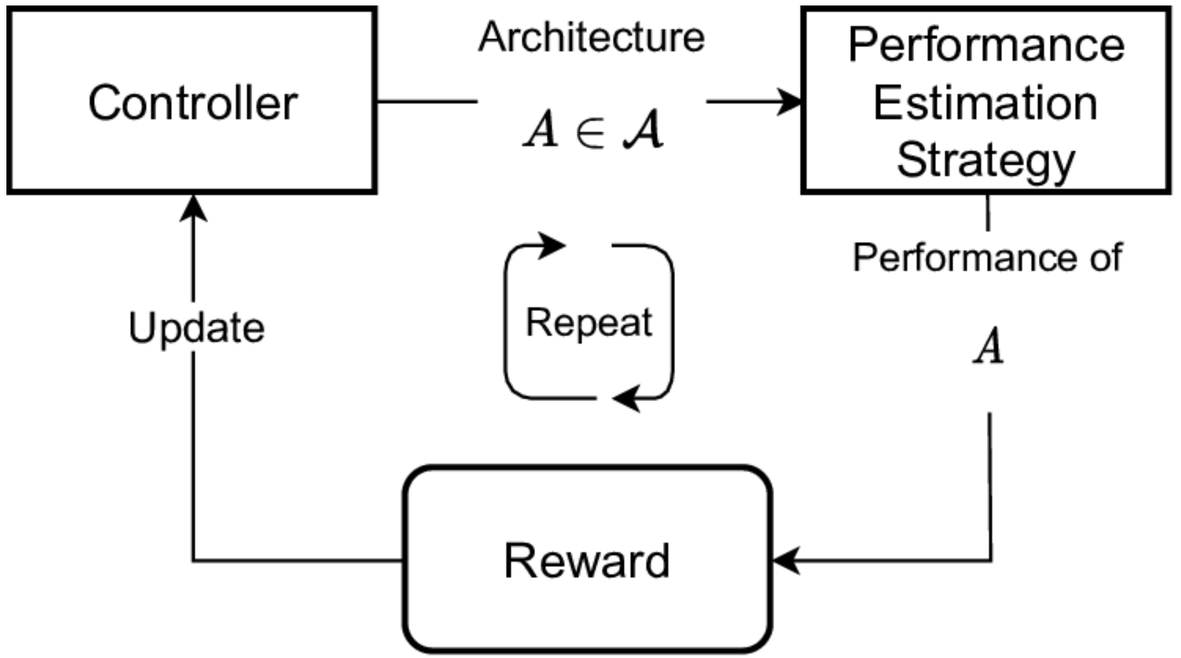
- Architecture optimization proven superior against pruning, quantization and distillation for reducing computations
- Large data collection is often overlooked challenge for companies
- Data efficiency effectively reduce cost gap avoiding large data collection
- A-priori knowledge remove part of learning burden from NNs
- The company funding the PhD position paid for it



- Davis Blalock, et al. "What is the state of neural network pruning?." Proceedings of machine learning and systems 2 (2020): 129-146.

1. Neural Architecture Search

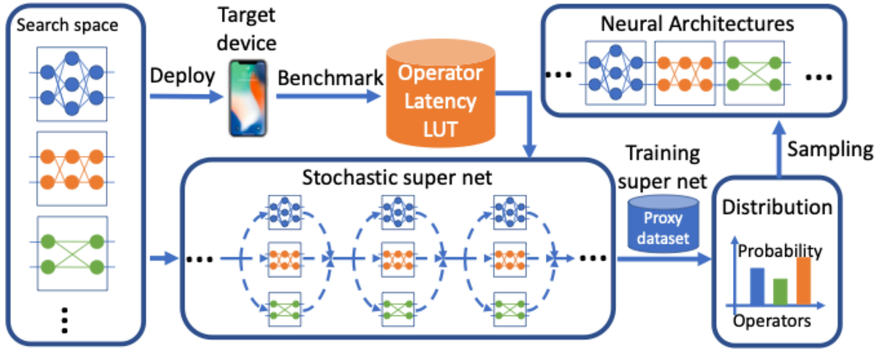
- How to come up with effective sequences of operations for NN architectures?
- *Search space*: set of operations and possible interconnections
- *Controller*: exploration policy for traversing the search space
- *Performance estimation*: test if sampled model perform well
- Update controller policy depending on outcome of performance estimation

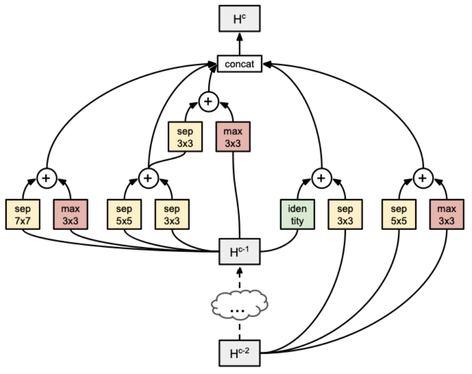


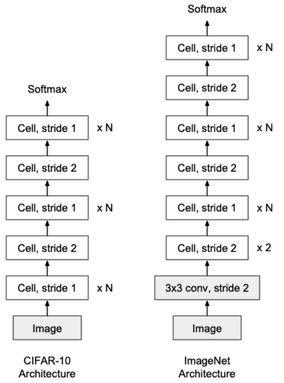
1. Neural Architecture Search

Advantages:

- Performance estimation and controller update may be efficiency-dependent:
    1. Computational requirements (FLOPs or MACs)
    2. Latency
    3. Hardware-aware search
- Tunable controller policy:
    4. Evolutionary-based approaches
    5. Reinforcement learning
    6. Bayesian optimization
    7. Gradient-based approaches
- (Sub-)Optimal architecture is achievable







- Yuhui Xu, et al. "Latency-aware differentiable neural architecture search." arXiv preprint arXiv:2001.06392 (2020).
- Bichen Wu, et al. "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019.
- Chenxi Liu, et al. "Progressive neural architecture search." Proceedings of the European conference on computer vision (ECCV). 2018.

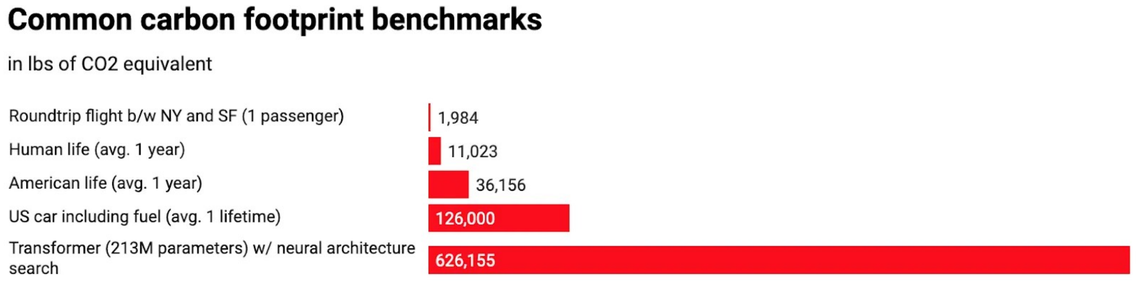
1. Neural Architecture Search

Disadvantages:

- Searching requires training of thousands/millions of models to reach (sub)-optimal architecture
- Search space complexity scale with model size and complexity of operations
- Search space heavily restricted in practice by forcing specific cell operations or cell sequences
- Controller policy learns best way to *search* (*traverse*) a search space rather than learning good model *construction* policies
- Limited flexibility in generalization across tasks

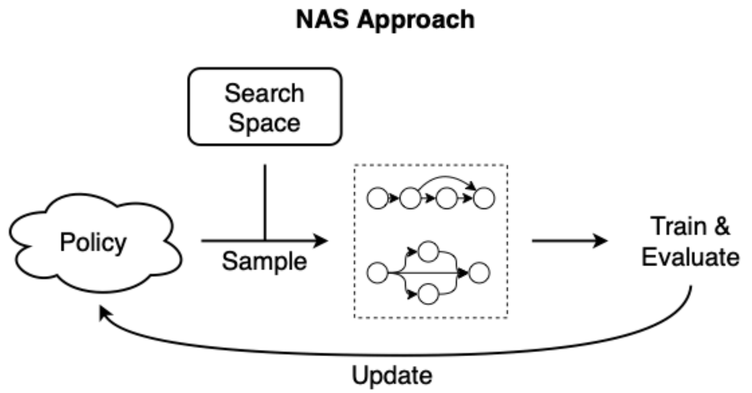
Model	Hardware	Power (W)	Hours	kWh·PUE	CO_2e	Cloud compute cost
$T2T_{base}$	P100x8	1415.78	12	27	13	$41–$140
$T2T_{big}$	P100x8	1515.43	84	201	96	$289–$981
ELMo	P100x3	517.66	336	275	131	$433–$1472
$BERT_{base}$	V100x64	12,041.51	79	1507	719	$3751–$12,571
$BERT_{base}$	TPUv2x16	—	96	—	—	$2074–$6912
NAS	P100x8	1515.43	274,120	656,347	313,078	$942,973–$3,201,722
NAS	TPUv2x1	—	32,623	—	—	$44,055–$146,848
GPT-2	TPUv3x32	—	168	—	—	$12,902–$43,008

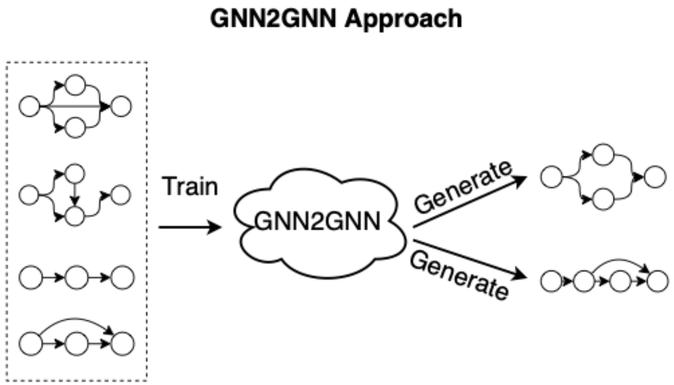
Table 2: Estimated cost of training a model in terms of CO_2 emissions (lbs) and cloud compute cost (USD) (Strubell, Ganesh, and McCallum 2019). Power and carbon footprint are omitted for TPUs due to lack of public information on power draw.



1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

- Is it possible to learn to *construct* NN architectures rather than *searching* for them?
- NN architectures are simple DAGs processable using GNNs
- One-shot generation process, avoiding training of multiple NN architectures
- Generation should take into account both validity and performance of the DAG
- Are design principles valid in general and translate well across tasks?

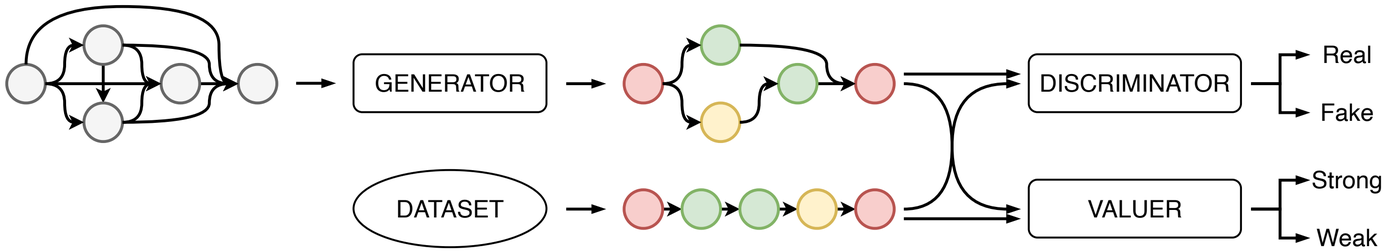




- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

- Take inspiration from Generative Adversarial Network, train a generator GNN to automatically construct effective NN architectures (DAGs)

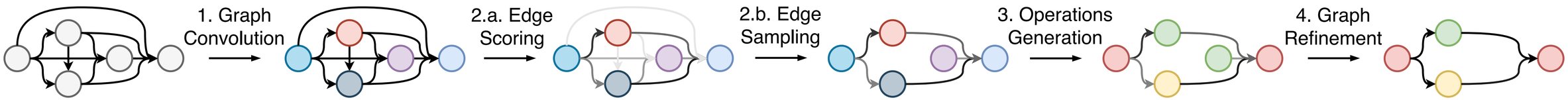


- Three-way adversarial training process:
  1. Generator aim at construct DAG
  2. Discriminator aim at distinguish architecture validity
  3. Valuer aim at predict architecture performance

- Generator loss function: $\mathscr{L}_G = \lambda \cdot \underbrace{\log(1 - D(G(z)))}_{\text{standard GAN loss}} + (1 - \lambda) \cdot \underbrace{\mathscr{L}_R(V(G(z)))}_{\text{reward loss}}$

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks



- Generator process:
    1. Input randomly initialized fully connected DAGs
    2. Compute new graph embedding via graph convolution
    3. Compute edge scores, depending on adjacent node embeddings
    4. Sample “strong” edges over threshold using gumbel softmax (to avoid non-differentiability of thresholding)
    5. Compute graph operations for each node, depending on node embedding and edge scores
    6. Remove unsampled edges and isolated nodes

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

Experimental setup:

- Testing on NAS datasets (NAS101 and NATS) contain a set of NN architectures and recorded performance
- NAS101 $\simeq$ 423k architectures over CIFAR-10, NATS $\simeq$ 7k architectures over three datasets
- Similar search spaces with repetition of optimized cells
- Set of operations: 3 $\times$ 3 convolution, 5 $\times$ 5 convolution, max-pooling, average-pooling
- NAS101 cells with at most 7 nodes and 9 edges
- NATS cells with at most 8 nodes, no limit on edges
- Remove best 10% of architectures from datasets to test generation quality
- Randomly remove portion of remaining architectures to ensure training on small number of architectures
- Thorough ablation study to optimize GNN2GNN hyperparameters (loss balancing factor, number of layers, etc.)

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

GNN2GNN vs GANs:

- MOLGAN: generate simultaneously nodes and edges (effective in chemistry domain)
- RNN: generates starting from input node appending new vertices with edges until stopping criterion
- GNN2GNN more accurate predictions for strong NN architectures
- GNN2GNN more than 80% of predicted architectures with accuracy > 90%, MOLGAN less than 60%
- 20% improvement on Top-5% architectures
- 3% higher average accuracy over state-of-the-art

Dataset	Model	Novelty	Top-5	Top-10	Top-50	Acc_{90}	|Acc|
NAS101	GNN2GNN	82.60%	**45.91%**	**52.21%**	**57.37%**	**81.79%**	**92.04%**
	MOLGAN	65.41%	22.63%	27.29%	45.20%	59.71%	89.39%
	RNN	**96.34%**	1.69%	2.32%	4.81%	53.04%	89.32%
	Random	51.81%	11.17%	13.74%	28.43%	43.18%	88.54%

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

Resistance to quality degradation:

- Flexibility against poorly-constructed datasets
- High number of strong models removed from dataset
- Remove best N% of models from NAS101
- Minimal performance loss even when best 90% removed
- 42% among Top-5% of models even when best 90% removed

Dataset	N	Novelty	Top-5	Top-10	Top-50	Acc_{90}	$\lvert\text{Acc}\rvert$
NAS101	10%	82.60%	45.91%	52.21%	57.37%	81.79%	92.04%
	20%	83.01%	45.54%	52.32%	58.48%	82.05%	91.98%
	30%	83.67%	46.10%	51.07%	57.04%	80.14%	91.68%
	40%	84.89%	45.80%	51.03%	58.71%	81.03%	91.55%
	50%	85.00%	44.01%	48.81%	56.30%	79.24%	91.33%
	60%	84.58%	44.40%	49.12%	57.72%	80.30%	91.30%
	70%	84.20%	44.66%	49.38%	56.71%	79.52%	91.37%
	80%	84.33%	43.90%	48.27%	55.51%	78.16%	91.27%
	90%	85.71%	42.77%	46.70%	55.72%	78.24%	91.22%

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

Generalization between datasets:

- GNN2GNN over NATS poor performance due to small size of NATS
- Apply generator model trained over NAS101 to NATS
- Performance strongly increase when transferred from NAS101 to NATS
- Only 576 NATS architectures are common for NAS101

Dataset		Novelty	Top-5	Top-10	Top-50	|Acc|
Train	Test					
NATS_{C10}	NATS_{C10}	**76.73%**	1.65%	3.39%	15.61%	68.91%
NAS101	NATS_{C10}	73.67%	**2.64%**	**5.08%**	**16.55%**	**70.25%**
NATS_{C100}	NATS_{C100}	71.71%	1.30%	3.28%	13.10%	33.03%
NAS101	NATS_{C100}	**72.03%**	**2.64%**	**4.82%**	**16.90%**	**35.40%**
NATS_{I}	NATS_{I}	81.03%	0.91%	2.17%	8.42%	16.75%
NAS101	NATS_{I}	**82.30%**	**1.93%**	**3.64%**	**11.49%**	**18.84%**

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

GNN2GNN vs NAS:

- GNN2GNN no search space exploration
- Compare GNN2GNN against state-of-the-art NAS over generated architecture on NAS101
- Average accuracy comparable with NAS
- Improved best accuracy
- Lower rank value (generated architectures closer to the optimal one)
- Close to best achievable accuracy in NAS101 (95.06%)
- Removing 90% of best performing architectures still GNN2GNN > NAS

Model	|Acc|	Best Acc	Best Rank
DARTS Liu et al. [2019]	92.21%	93.02%	57079
NAO Luo et al. [2018]	**92.59%**	93.33%	19552
ENAS Pham et al. [2018]	91.83%	92.54%	96939
GNN2GNN_{10}	92.04%	**94.32%**	**5372**
GNN2GNN_{30}	91.68%	94.18%	7843
GNN2GNN_{60}	91.30%	94.01%	9371
GNN2GNN_{90}	91.22%	93.69%	12570

- Andrea Agiollo, and Andrea Omicini. "GNN2GNN: Graph neural networks to generate neural networks." Uncertainty in Artificial Intelligence. PMLR, 2022.

1. GNN2GNN: Graph Neural Networks to Generate Neural Networks

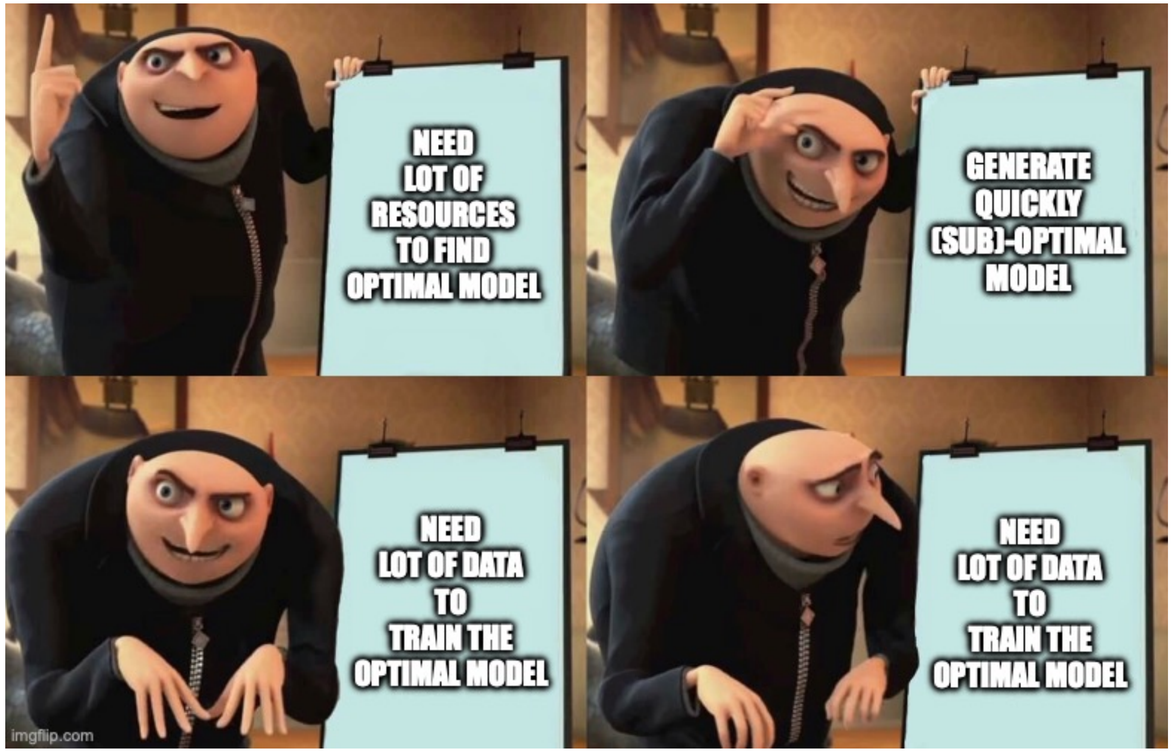
Summary:

- Possible to generate effective NN architecture in one-shot vs exploring search space
- *Resource saving*: GNN2GNN outperform NAS while requiring information on few architectures performance
- *Adaptability*: GNN2GNN performs well even when exploring ineffective architectures
- *Generalizability*: generator model output effective architecture on different domains
- *Flexibility*: GNN2GNN generator coupled with exploration approaches
- Several subsequent NAS approaches rely on similar ideas: one-shot generation or embedding dataset information into exploration

- Zeyang Zhang, et al. "Disentangled continual graph neural architecture search with invariant modular supernet." Forty-first International Conference on Machine Learning. 2024.
- Lianbo Ma, et al. "Single-domain generalized predictor for neural architecture search system." IEEE Transactions on Computers (2024).
- Zhihua Chen, et al. "Mngnas: distilling adaptive combination of multiple searched networks for one-shot neural architecture search." IEEE Transactions on pattern analysis and machine intelligence 45.11 (2023): 13489-13508.

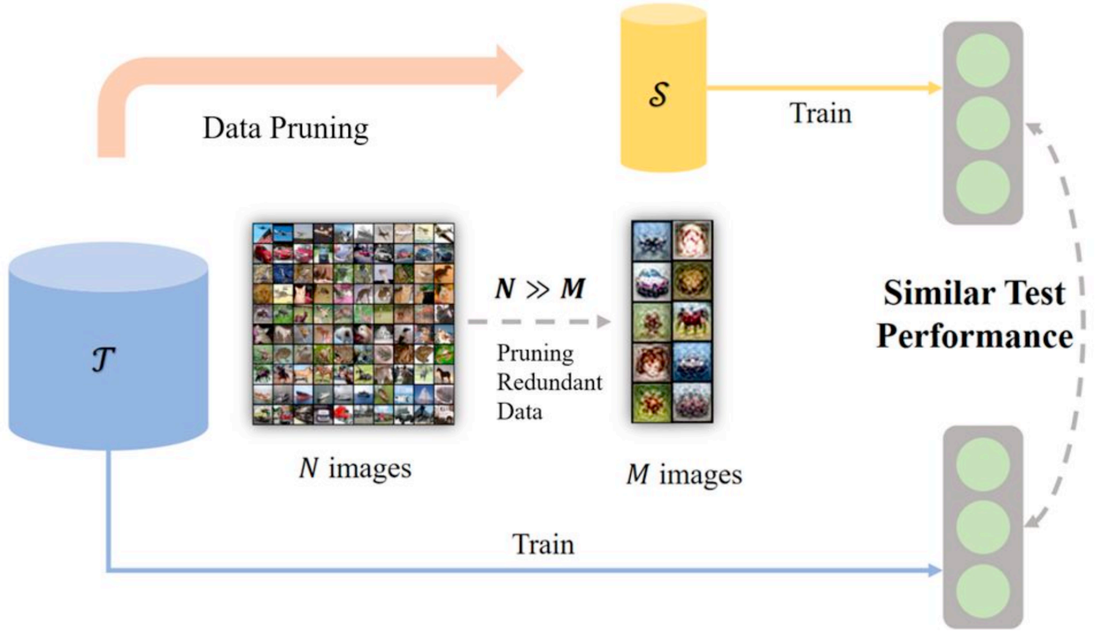
1. Lessons Learnt from NAS

- Optimizing architecture reduce computational requirements
- Optimal architecture is task-dependent
- Optimal architecture is non trivial
- Search space complexity proportional to task complexity and operation set
- Searching optimal architecture is expensive, time-consuming and resource inefficient
- Embed resource-efficiency requirements in NAS
- Generate automatically sub-optimal architecture with GANs and meta-learning
- (Sub)-optimal model still requires lot of data to train



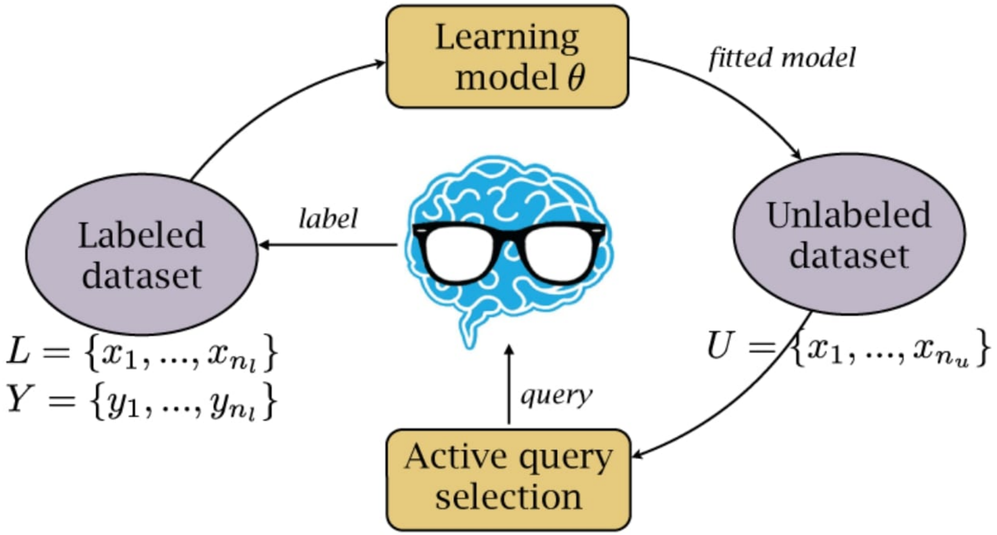
2. Dataset Summarization

- Compress data used for training with minimal loss in predictive performance
- Retain data interpretability (avoid data transformation)
- One-shot summarization
- Useful whenever several downstream tasks use same data or whenever training multiple models (e.g., NAS)
- *Key:* find informative example data



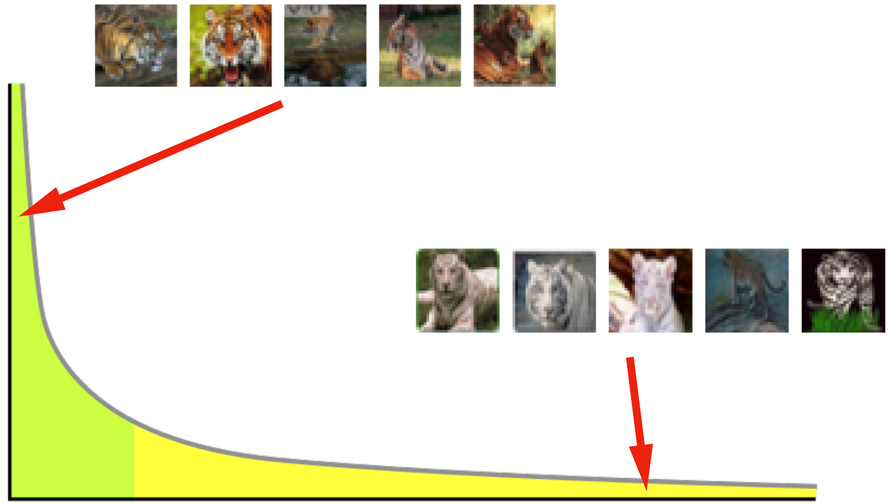
2. Active Learning

- Iterative sample selection process
- Most informative examples from unlabeled dataset labeled by an oracle
- Labeled samples used to train model
- Iterate cycle until convergence
- From small labeled dataset, optimal queries equal high data reduction
- Useful when lots of unlabeled data are available but labeling is costly
- *Key:* find informative example data points



2. How to Find Informative Samples?

- Different samples bring different information
- Data contain noise
- Models may differ in how they process information
- Data made of typical subpatterns and atypical subpatterns
- Model must identify atypical subpatterns as well
- Proxies for importance of a data point:
  - Ease of learning: how soon is a data point learnt?
  - Forgetfulness: once it is learnt, is it ever “forgotten”?
  - Signal strength: how strong is the signal contained in a sample?



- Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. "Deep learning on a data diet: Finding important examples early in training." Advances in neural information processing systems 34 (2021): 20596-20607.
- Mariya Toneva, et al. "An empirical study of example forgetting during deep neural network learning." arXiv preprint arXiv:1812.05159 (2018).

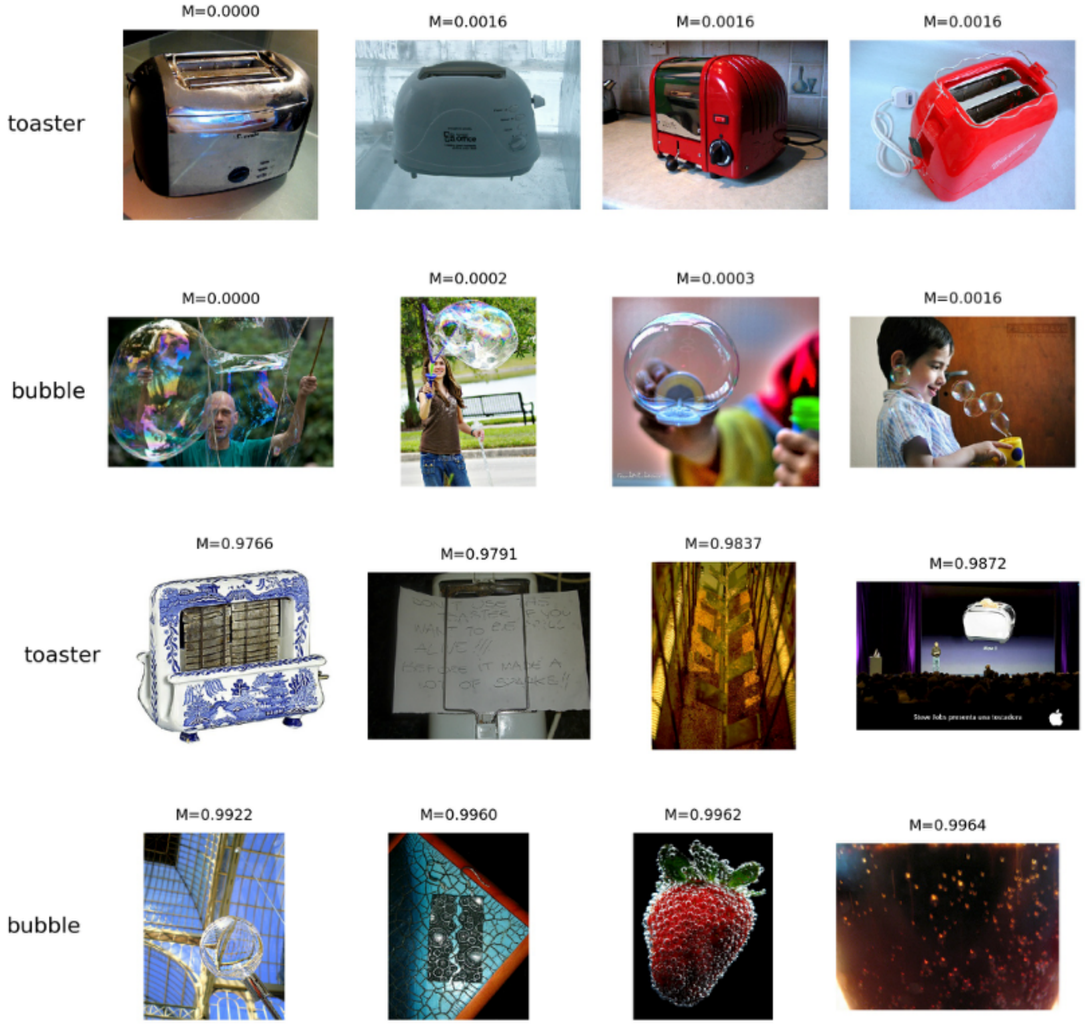
2. Memorization Effect of NNs

- NNs should extract general-purpose patterns from seen sample that allow generalizing to unseen samples applying the same patterns
- Modern NNs are highly overparametrized
- Overparametrized models tend to “memorize” samples with low signal as distinctive patterns to reach perfect training score
- Identification of the subpatterns lies between perfect learning and perfect memorization
- Memorization score identify which samples are memorized and which are learnt
- $$m(\mathbf{x}_i) = P\left[h^{\mathscr{A}}_{\mathscr{T}}(\mathbf{x}_i) = y_i\right] - P\left[h^{\mathscr{A}}_{\mathscr{T}\setminus i}(\mathbf{x}_i) = y_i\right]$$

- Vitaly Feldman, and Chiyuan Zhang. "What neural networks memorize and why: Discovering the long tail via influence estimation." Advances in Neural Information Processing Systems 33 (2020): 2881-2891.

2. Memorization Effect of NNs

- $m(\mathbf{x}_i) = P\left[h^{\mathscr{A}}_{\mathscr{T}}(\mathbf{x}_i) = y_i\right] - P\left[h^{\mathscr{A}}_{\mathscr{T}\setminus i}(\mathbf{x}_i) = y_i\right]$
- Low $m(\mathbf{x}_i)$: removing from training set has minimal effect as model learned subpatterns from other similar data
- Low $m(\mathbf{x}_i)$ → Highly typical samples
- High $m(\mathbf{x}_i)$: removing from training set has significant effect as model memorized the subpatterns solely from itself or few others
- High $m(\mathbf{x}_i)$ → Highly atypical samples



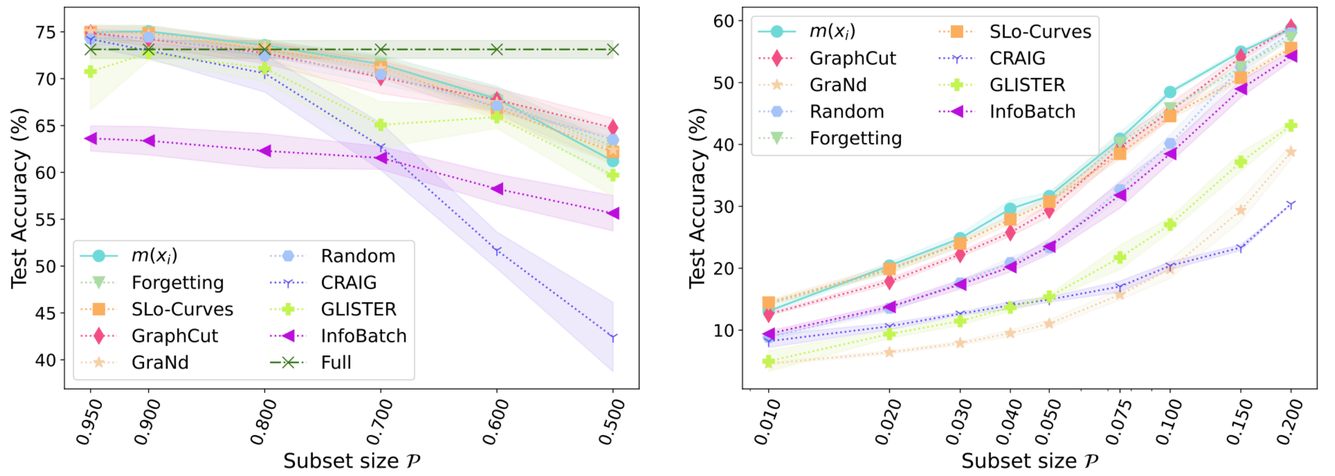
- Vitaly Feldman, and Chiyuan Zhang. "What neural networks memorize and why: Discovering the long tail via influence estimation." Advances in Neural Information Processing Systems 33 (2020): 2881-2891.

2. Memorization as Informativeness Measure

- Is it possible to use memorization as metric for defining informativeness level of sample?
- Want to retain typical samples that generalize well → low $m(\mathbf{x}_i)$
- Don’t want too many typical samples as they contain redundant information → high $m(\mathbf{x}_i)$
- Want to include fair amount of atypical samples to not miss out on rare subclasses → high $m(\mathbf{x}_i)$
- Few catches here and there
- Dataset large enough → Prioritize removing redundant information coming from low $m(\mathbf{x}_i)$
- Dataset not large enough → Prioritize keeping only simple patterns coming from low $m(\mathbf{x}_i)$

2. Memorization as Informativeness Measure

- Sample high $m(\mathbf{x}_i)$ data for data pruning (large subset)
- Sample low $m(\mathbf{x}_i)$ data for few-shot data summarization (small subset)



(a) Data pruning **(b) Few-shot data summarization**

- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "Approximating Memorization Using Loss Surface Geometry for Dataset Pruning and Summarization." Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024.

2. Issues with Memorization Scores

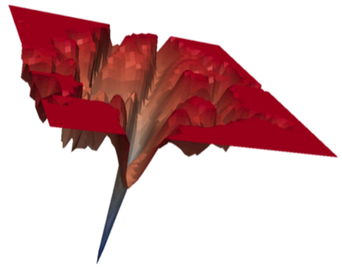
- Exact memorization score computation is computationally infeasible
- With sampling strategy requires training 4000 NNs to compute memorization scores over CIFAR100
- Approximate measures (e.g., influence) found to be fragile
- *Challenge*: tractably approximate memorization scores



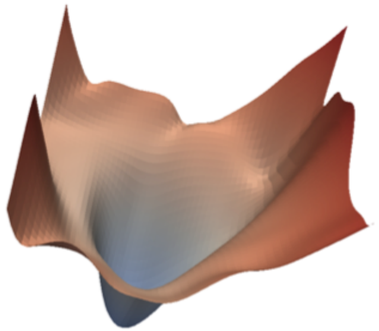
2. Sharpness Aware Minimization (SAM)

- Alternative to standard SGD optimization algorithm
- Optimization algorithm designed to find a flatter minima

$$\min_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) + \lambda ||\boldsymbol{w}||_2^2 \quad \text{where} \quad L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon})$$



- Two steps:
    1. Gradient ascent to maximum loss in the vicinity of the current minima
    2. Gradient descent



- Empirically shown performance gains compared to SGD
- SAM gains come from generalization on atypical samples

- Pierre Foret, et al. "Sharpness-aware minimization for efficiently improving generalization." arXiv preprint arXiv:2010.01412 (2020).
- Young In Kim, et al. "On memorization and privacy risks of sharpness aware minimization." arXiv preprint arXiv:2310.00488 (2023).

2. SAMIS to Approximate Memorization

- Theorem on SGD vs SAM generalization capability on typical and atypical data

  A. $|L^{0-1}_{\mathsf{D}_{\text{typ}}}(\mathbf{W}_{\text{SAM}}) - L^{0-1}_{\mathsf{D}_{\text{typ}}}(\mathbf{W}_{\text{SGD}})| \leq \epsilon$

  B. $|L^{0-1}_{\mathsf{D}_{\text{typ}}}(\mathbf{W}_{\text{SAM}}) - L^{0-1}_{\mathsf{D}_{\text{typ}}}(\mathbf{W}_{\text{SGD}})| \geq 0.1 + \epsilon$

- *Idea*: reverse-engineer SAM vs SGD discrepancy to approximate data typicality and memorization

- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "Approximating Memorization Using Loss Surface Geometry for Dataset Pruning and Summarization." Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024.

2. SAMIS to Approximate Memorization

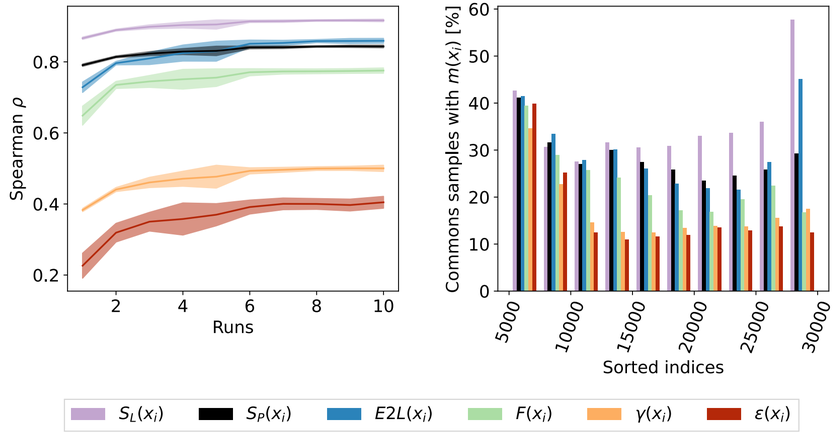
- SAMIS-L: absolute difference in loss (supervised):

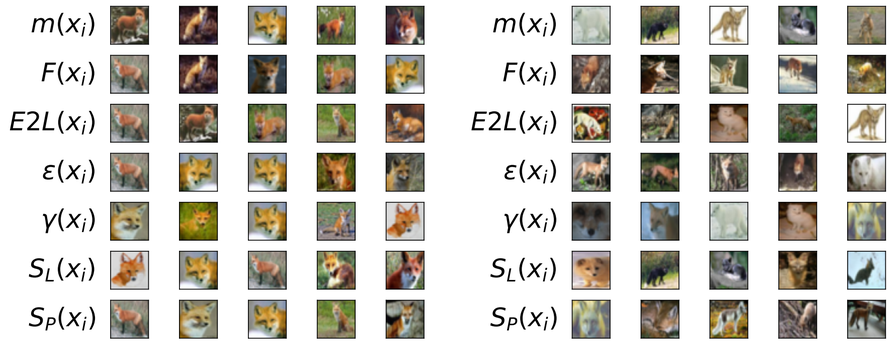
$$S_L(\mathbf{x}_i) = |\mathscr{L}\left(h^{SAM}_{\mathcal{S}\setminus i,..,z}(\mathbf{x}_i), y_i\right) - \mathscr{L}\left(h^{SGD}_{\mathcal{S}\setminus i,..,z}(\mathbf{x}_i), y_i\right)|$$

- SAMIS-P: absolute difference in prediction probability (unsupervised):

$$S_P(\mathbf{x}_i) = \frac{\sum_{c=1}^{C} |P\left(h^{SAM}_{\mathcal{S}\setminus i,..,z}(\mathbf{x}_i) = c\right) - P\left(h^{SGD}_{\mathcal{S}\setminus i,..,z}(\mathbf{x}_i) = c\right)|}{C}$$

- SAMIS correlate well on low and high memorization extremes
- Samples matching between memorization scores and SAMIS
- Similar results with other non flatness-aware optimizers (Adam, etc.)
- Outperforms other memorization proxies

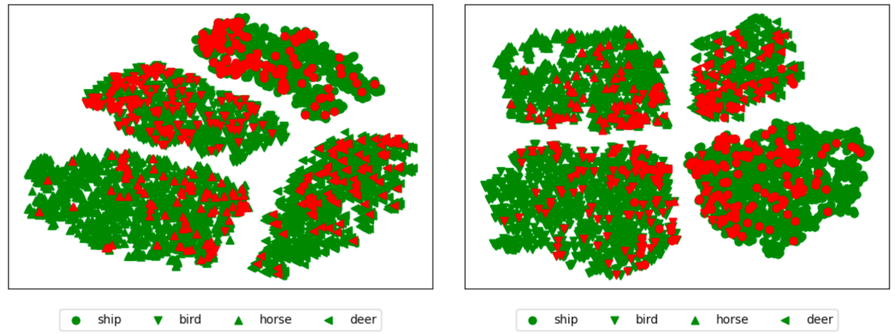


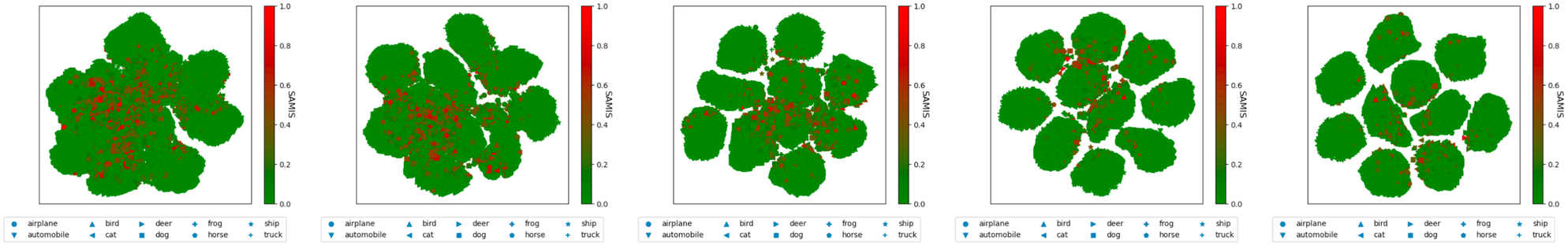


- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "Approximating Memorization Using Loss Surface Geometry for Dataset Pruning and Summarization." Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024.

2. SAMIS as Informativeness Measure

- Samples with high SAMIS ≃ critical positions of embedding space
- Early epochs → SAMIS ≃ clusters are still joined together
- Central epochs → SAMIS ≃ decision boundaries
- Later epochs → SAMIS ≃ singletons and outliers
- Sampling with SAMIS select data critical for decision boundary refinement



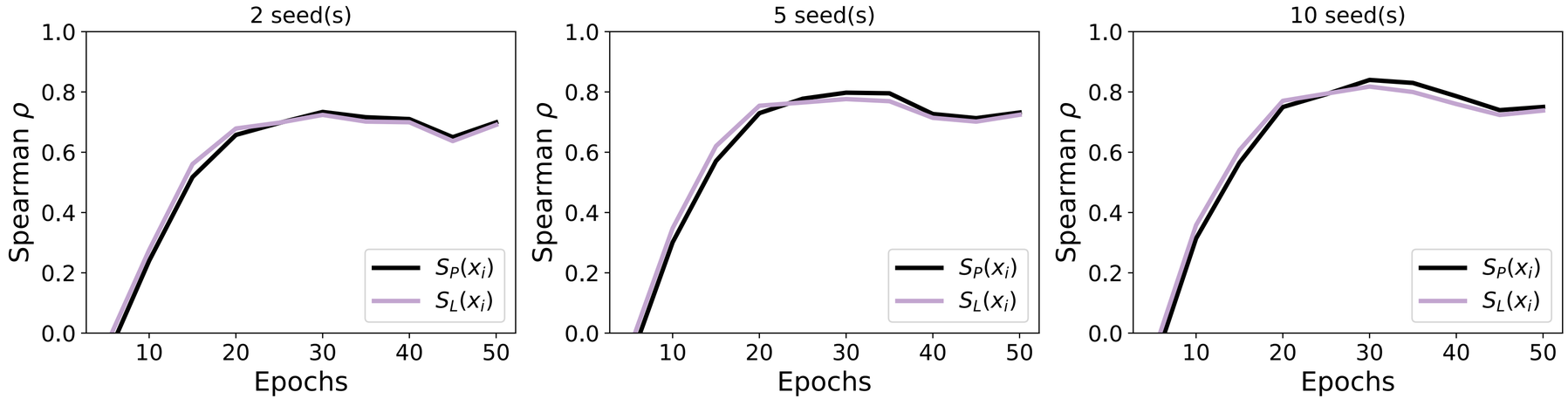


(a) Epoch 30 (b) Epoch 50 (c) Epoch 70 (d) Epoch 90 (e) Epoch 160

- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "SAMOSA: Sharpness Aware Minimization for Open Set Active learning." Under Submission. 2025.

2. SAMIS Efficiency

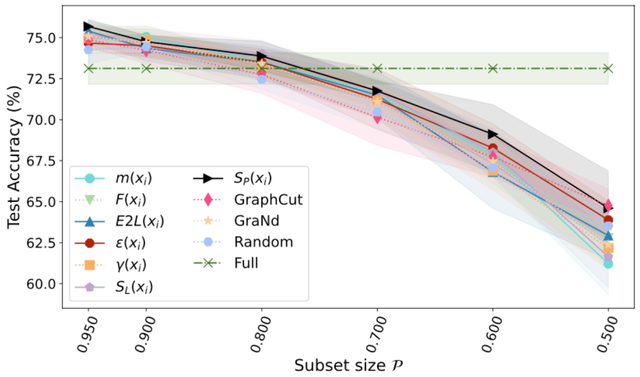
- SAMIS requires training $2nk$ models: n number of seeds and k number of dataset splits
- n=5 and k=10 highest correlation and 40× speed up compared to 4000 NNs for memorization
- SAMIS alternative version using premature training: 0.8 correlation and 2000× speed-up
- Training only on 10 seeds for 30 epochs ≃ 600 epochs total ≃ training a couple of models



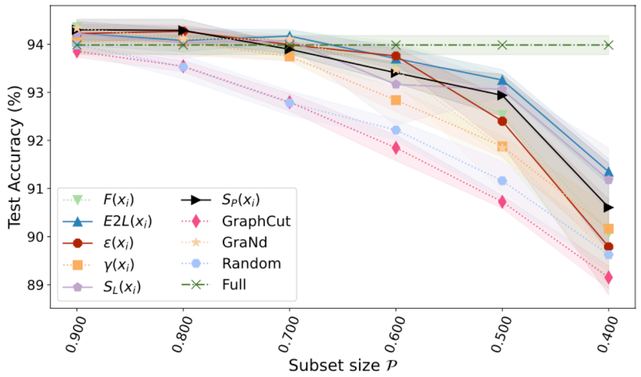
- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "Approximating Memorization Using Loss Surface Geometry for Dataset Pruning and Summarization." Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024.

2. SAMIS for Dataset Summarization

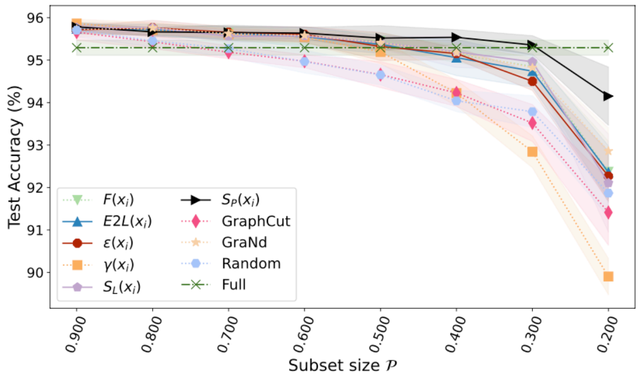
- SAMIS best-performing approaches over majority of compression values
- Other memorization proxies well-performing but slower to compute
- SVHN: remove up to 70% of data obtain test accuracy identical to full training
- CIFAR10: prune 50% of data achieve 1% accuracy degradation
- High performance drop when stability of typical samples removed



(a) CIFAR100



(b) CIFAR10

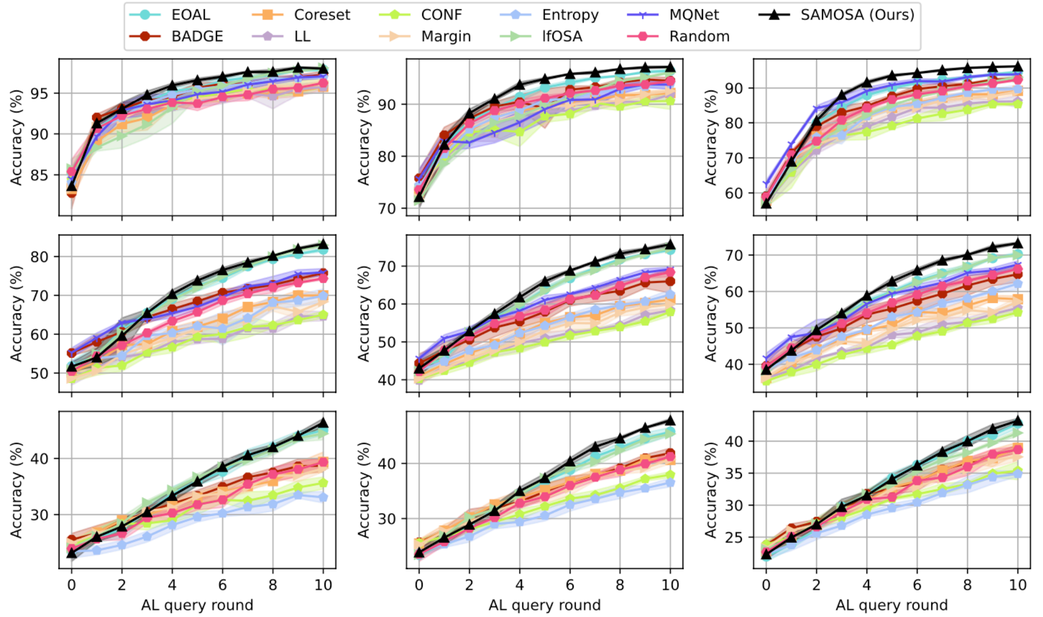


(c) SVHN

- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "Approximating Memorization Using Loss Surface Geometry for Dataset Pruning and Summarization." Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2024.

2. SAMIS for Active Learning

- Comparing against state-of-the-art on 3 datasets
- AL procedure over 11 query rounds
- 1500 samples queried from unlabeled pool at each round
- Up to 3% accuracy improvements
- Best average rank across datasets
- Improvements proportional to percentage of mismatch
- Small initial dataset causes struggle
- High sampling precision



Strategy	CIFAR10			CIFAR10			TinyImageNet			Average Rank (↓)
	20%	30%	40%	20%	30%	40%	20%	30%	40%	
Random	96.25 ± 0.15	94.50 ± 0.15	92.33 ± 0.21	74.36 ± 0.38	68.40 ± 1.15	66.18 ± 0.06	39.34 ± 0.85	41.23 ± 0.72	38.68 ± 0.48	5.67
BADGE	97.26 ± 0.39	94.58 ± 0.62	92.76 ± 0.40	75.74 ± 0.46	66.00 ± 3.09	64.73 ± 2.15	38.73 ± 1.13	41.87 ± 0.89	38.80 ± 0.61	5.00
LL	95.76 ± 0.62	91.92 ± 0.86	86.16 ± 1.70	64.64 ± 1.18	58.37 ± 0.51	55.46 ± 1.38	-	-	-	10.33
Coreset	95.78 ± 0.30	92.22 ± 0.54	89.21 ± 0.70	70.00 ± 0.42	61.18 ± 1.01	57.88 ± 0.89	39.39 ± 1.73	40.62 ± 0.64	38.95 ± 0.89	7.22
Entropy	96.91 ± 0.20	94.03 ± 1.04	89.76 ± 0.97	69.89 ± 1.76	62.52 ± 1.25	62.13 ± 1.61	33.08 ± 0.99	36.53 ± 0.77	34.90 ± 0.40	7.56
CONF	96.24 ± 0.54	90.64 ± 1.61	85.35 ± 0.30	65.03 ± 0.86	57.93 ± 1.12	54.25 ± 0.51	35.64 ± 1.05	37.94 ± 0.53	35.38 ± 1.40	9.67
Margin	96.49 ± 0.67	92.62 ± 0.44	89.09 ± 0.87	68.36 ± 0.67	59.93 ± 1.11	57.20 ± 1.33	38.25 ± 0.61	40.70 ± 0.72	37.55 ± 0.65	7.89
MQNet	97.05 ± 0.22	93.82 ± 0.87	93.89 ± 0.28	75.94 ± 0.93	69.19 ± 0.52	67.38 ± 1.02	-	-	-	4.50
lfOSA	98.09 ± 0.22	96.17 ± 0.19	93.07 ± 0.65	82.94 ± 0.50	74.77 ± 0.42	70.00 ± 0.37	44.41 ± 1.10	45.35 ± 0.15	41.29 ± 0.43	2.67
EOAL	97.89 ± 0.27	96.49 ± 0.33	94.61 ± 0.25	81.78 ± 0.52	74.29 ± 0.37	70.25 ± 0.55	45.03 ± 0.84	45.76 ± 0.75	42.76 ± 0.71	2.33
SAMOSA (Ours)	98.00 ± 0.15	97.23 ± 0.18	96.19 ± 0.08	83.24 ± 0.47	75.75 ± 0.58	73.24 ± 0.30	46.38 ± 0.72	47.79 ± 0.34	43.22 ± 0.32	1.11

- Andrea Agiollo, Young In Kim, and Rajiv Khanna. "SAMOSA: Sharpness Aware Minimization for Open Set Active learning." Under Submission. 2025.

2. Lessons Learnt from Data Efficientization

- NNs overparametrization lead to memorization issues
- Memorization equivalent to lack of generalization
- Memorization target long tail distribution of natural data
- Memorization exploited to identify informative data
- Approximate memorization relying on generalization properties different optimization algorithms
- Memorization proxies effective for data efficiency
- Reduction of data required to train NNs



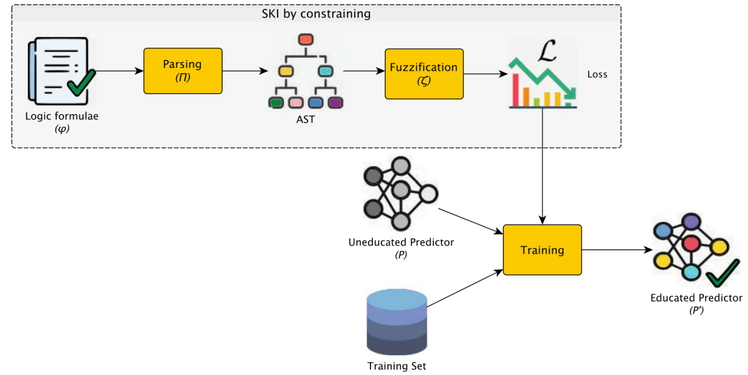
3. Neuro-Symbolic Integration for Efficientization

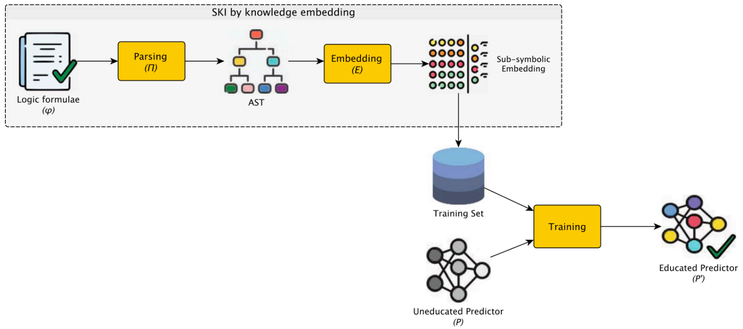
- Integration of *symbolic* knowledge and *sub-symbolic* predictors recent popular trend
- Symbolic knowledge is more reliable than data-driven approaches
- Integrating logical knowledge into NNs remove part of the learning burden
- *Idea*: hybridisation of sub-symbolic mechanisms with symbolic a-priori or expert knowledge may produce more efficient NNs.
- *Issue*: NeSy approaches focus on foundational results aiming at performance improvements (e.g., accuracy)
- Measure efficientization gains from integration of symbolic and sub-symbolic models

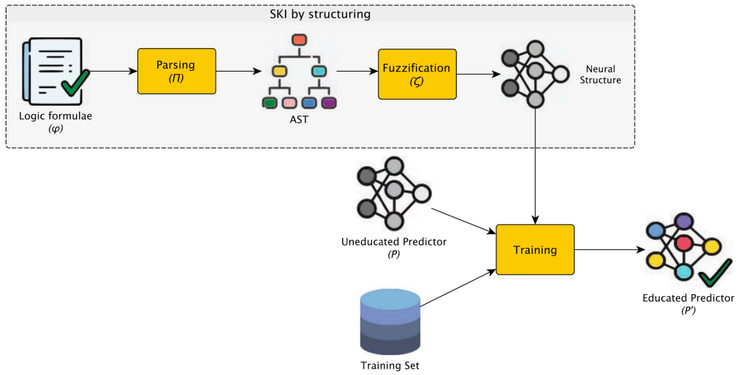
- Andrea Agiollo, Andrea Rafanelli, and Andrea Omicini. "Towards quality-of-service metrics for symbolic knowledge injection." CEUR Workshop Proceedings. Vol. 3261. Sun SITE Central Europe, RWTH Aachen University, 2022.

3. Symbolic Knowledge Injection (SKI)

- **Definition**: 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.
- 117 methods categorized into:
    1. **Structuring**: predictor created to mirror symbolic knowledge via its internal structure
    2. **Constraining**: knowledge used to steer learning of ML predictors penalizing inconsistent behaviors or incentivizing consistent ones
    3. **Embedding**: symbolic knowledge converted into numeric-array form used as ordinary input for ML predictor







- Giovanni Ciatto, Federico Sabbatini, Andrea Agiollo, Matteo Magnini, and Andrea Omicini. "Symbolic knowledge extraction and injection with sub-symbolic predictors: A systematic literature review." ACM Computing Surveys 56.6 (2024): 1-35.

3. Measuring SKI for Efficientization

- Uneducated predictor N, knowledge base K and injection process $\mathcal{I}(K, N)$
- Quality of Service metrics for SKI efficientization
- Memory footprint:
  - $\mu(\mathcal{I}) = \Psi(N) - \Psi(\mathcal{I}(K, N))$
- Energy consumption:
  - $\varepsilon^{\mathsf{i}}(\mathcal{I}) = \Upsilon^{\mathsf{i}}_{v}(N, T) - \Upsilon^{\mathsf{i}}_{v}(\mathcal{I}(K, N), T)$
  - $\varepsilon^{\mathsf{t}}(\mathcal{I}) = \Upsilon^{\mathsf{t}}_{v,\gamma}(e, N, T) - \Upsilon^{\mathsf{t}}_{v,\gamma}(e, \mathcal{I}(K, N), T)$
- Latency:
  - $\lambda(\mathcal{I}) = \dfrac{1}{|T|} \sum_{t \in T} \left(\Theta(N, t) - \Theta(\hat{N}, t) \right)$
- Data efficiency:
  - $\delta(\mathcal{I}) = \Delta_{\pi}(e, N, D, T) - \Delta_{\pi}(e, \mathcal{I}(K, N), D', T)$

Dataset	Model	Set	Data efficiency (KB)	Energy (mWh)	Memory (FLOPs)	Latency (ms)	Accuracy (%)
BCW	uneducated		–	–	–	–	94.53
	KBANN	train test	35.89	-1.47 -0.10	3933	-1.70	95.45
	KILL	train test	4.09	-0.99 0	0	0.35	94.63
	KINS	train test	-9.97	-1.22 -0.09	-559	-1.41	94.29
PSJGS	uneducated		–	–	–	–	93.91
	KBANN	train test	-4946.81	-4.67 -0.22	-66944	-2.56	92.84
	KILL	train test	553.89	-3 0	0	0.04	94.02
	KINS	train test	-954.80	-6.53 -0.51	-161779	-4.75	93.70
CI	uneducated		–	–	–	–	84.63
	KBANN	train test	1653.79	-1.41 -0.02	-2468	-0.43	84.78
	KILL	train test	4016.90	-0.70 0	4200	0	84.81
	KINS	train test	4263.50	-1.41 -0.02	-6220	-0.44	84.77

- Andrea Agiollo, Andrea Rafanelli, Matteo Magnini, Giovanni Ciatto, and Andrea Omicini . "Symbolic knowledge injection meets intelligent agents: QoS metrics and experiments." Autonomous Agents and Multi-Agent Systems 37.2 (2023): 27.

3. Measuring SKI for Efficientization

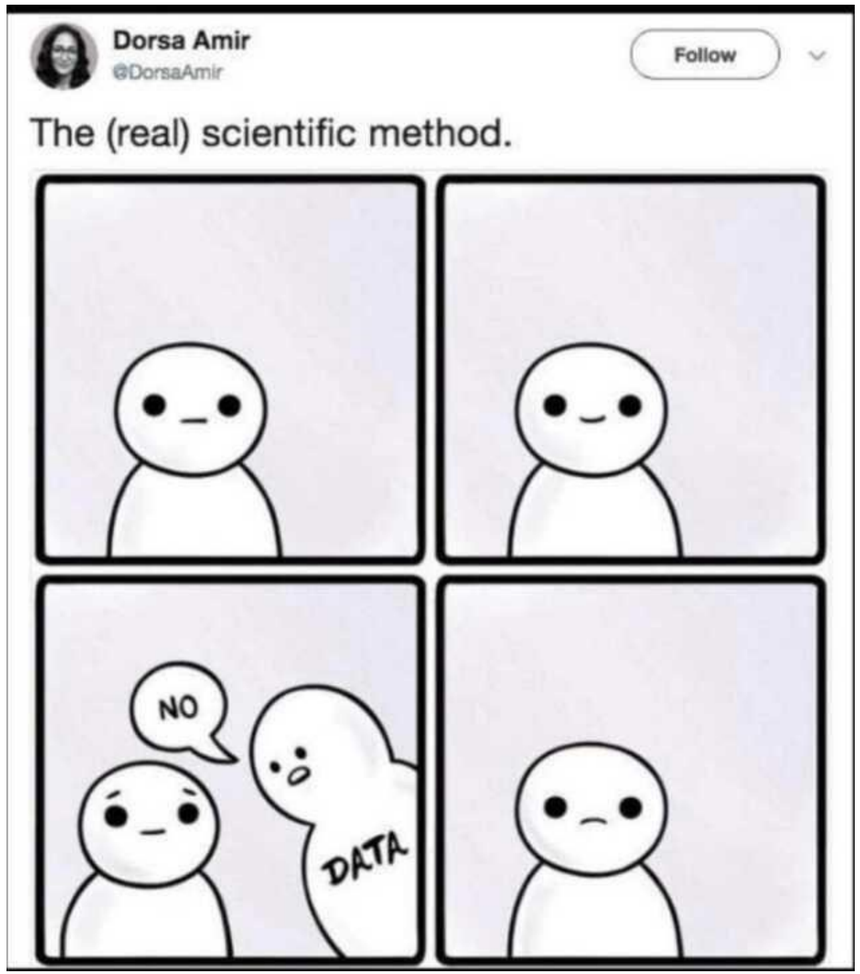
- Test QoS metrics on 3 SKI approaches and 3 datasets
- Memory footprint:
  - Depending on $\mathcal{J}(K, N)$ either positive or negative efficiency
- Energy consumption:
  - $\mathcal{J}(K, N)$ introduce complex conversion operations not optimized in available hardware
- Latency:
  - $\mathcal{J}(K, N)$ introduce complex conversion operations, but constraining apply to training only, thus achieve positive results for latency
- Data efficiency:
  - K provides information difficult to extract from data, $\mathcal{J}(K, N)$ reduce learning burden for N

Dataset	Model	Set	Data efficiency (KB)	Energy (mWh)	Memory (FLOPs)	Latency (ms)	Accuracy (%)
BCW	uneducated		–	–	–	–	94.53
	KBANN	train test	35.89	-1.47 -0.10	3933	-1.70	95.45
	KILL	train test	4.09	-0.99 0	0	0.35	94.63
	KINS	train test	-9.97	-1.22 -0.09	-559	-1.41	94.29
PSJGS	uneducated		–	–	–	–	93.91
	KBANN	train test	-4946.81	-4.67 -0.22	-66944	-2.56	92.84
	KILL	train test	553.89	-3 0	0	0.04	94.02
	KINS	train test	-954.80	-6.53 -0.51	-161779	-4.75	93.70
CI	uneducated		–	–	–	–	84.63
	KBANN	train test	1653.79	-1.41 -0.02	-2468	-0.43	84.78
	KILL	train test	4016.90	-0.70 0	4200	0	84.81
	KINS	train test	4263.50	-1.41 -0.02	-6220	-0.44	84.77

- Andrea Agiollo, Andrea Rafanelli, Matteo Magnini, Giovanni Ciatto, and Andrea Omicini . "Symbolic knowledge injection meets intelligent agents: QoS metrics and experiments." Autonomous Agents and Multi-Agent Systems 37.2 (2023): 27.

3. Lessons Learnt from SKI Efficientization

- SKI techniques inject a-priori knowledge into NNs via constraining, structuring or embedding
- NeSy research focus on improve accuracy or trust
- No available metrics for measuring efficientization
- SKI *not* improve energy efficiency
- SKI *not* improve latency
- SKI *not* improve size
- SKI improve data efficiency



4. Current Work & Future Directions

SKI and NeSy research focus on foundational research with less interest in real-world applications:

- One of those applications is cybersecurity → actively working on NeSy and SKI for cybersecurity (intrusion detection, phishing, etc.)
- Missing platform to share results on NeSy real-world application → ANSyA workshop @ ECAI 2025

4. Current Work & Future Directions

Not enough focus on security and trustworthiness of NeSy and SKI:

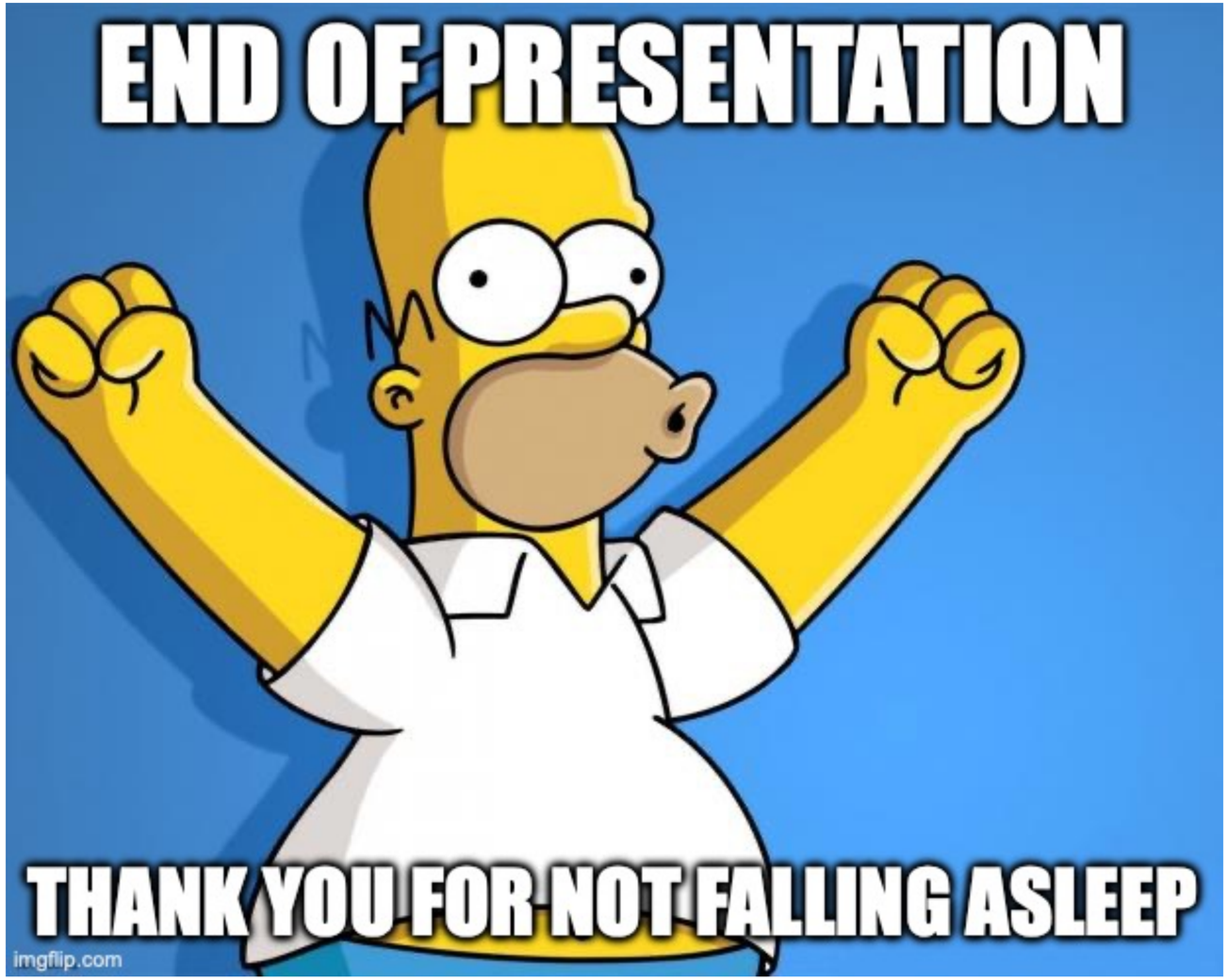
- Is knowledge actually learnt?
- Reasoning shortcuts
- Need to study extensively impact of ML attacks in NeSy → actively working on backdoor and adversarial attacks on NeSy and SKI

4. Current Work & Future Directions

Other projects:

- Impact of learning optimization on model security (with collaborators @ Purdue University)
- Integration of vulnerability knowledge in intrusion detection (with collaborators @ Sapienza University of Rome)
- Trustable/Verifiable usage of LLMs for cybsecurity applications such as IDSs (with collaborators @ Sapienza University of Rome)

Questions?



LinkedIn

GitHub