Neural Architecture Search and Robustness

Margret Keuper – Universität Mannheim

19.10.2023

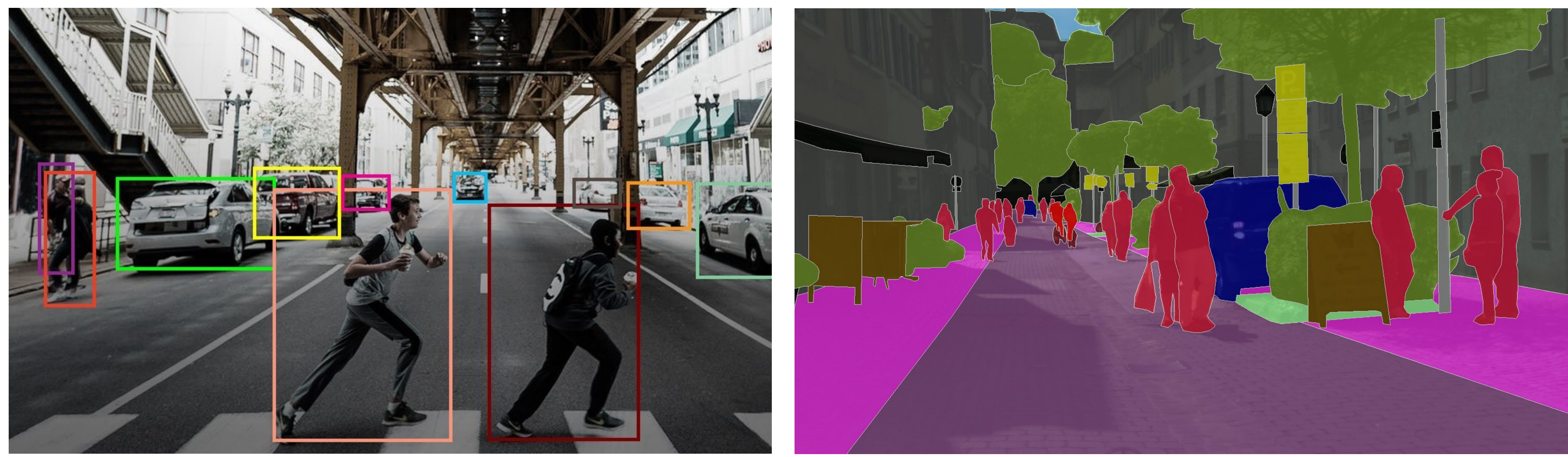
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.

High Accuracies for
Image Classification

ImageNet
(Russakovskyl et al., IJCV 2015)

Deployable Results on Object
Detection and Segmentation



Yolov4 on Cityscapes (Cordts et al., CVPR 2016)

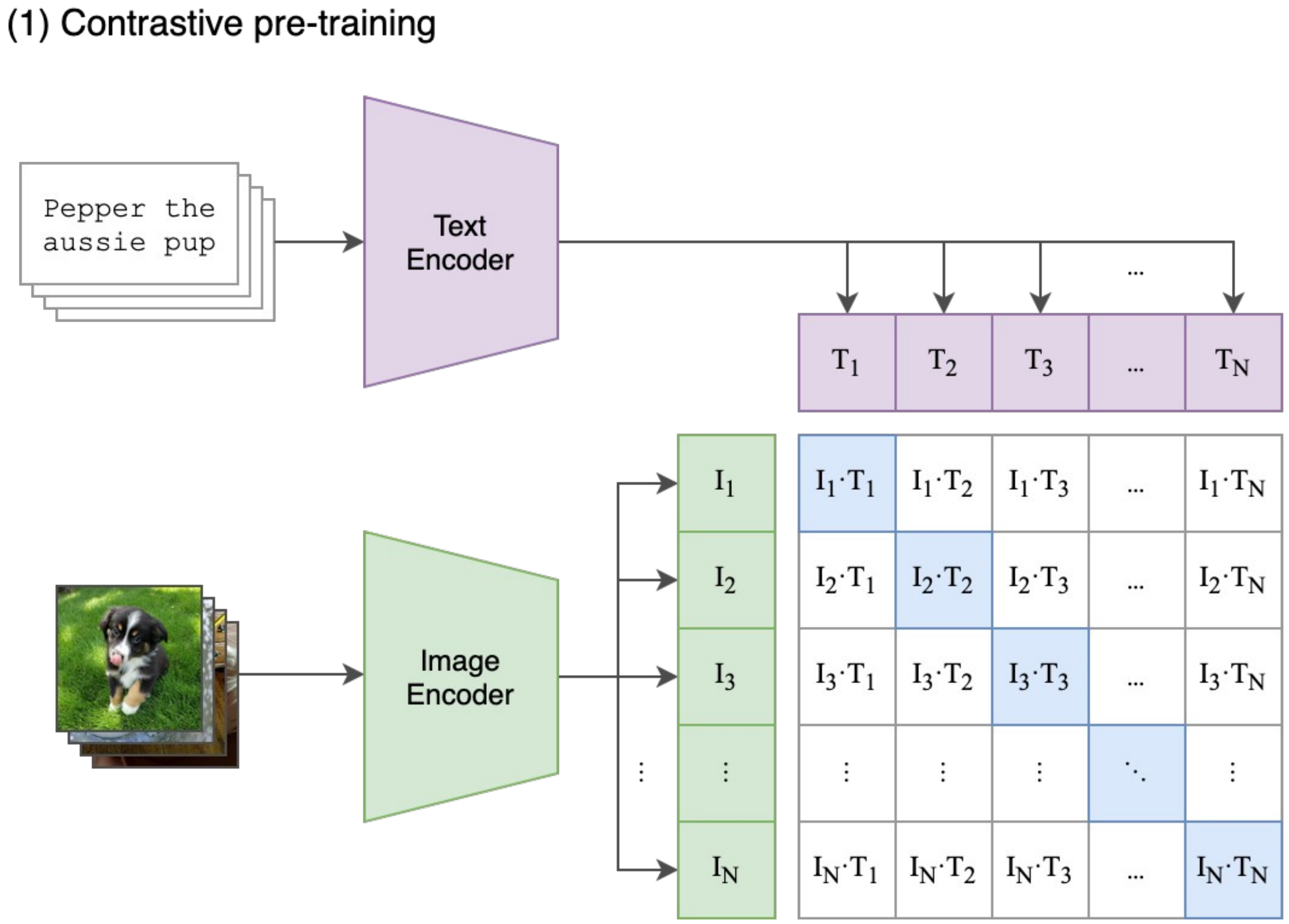
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero shot Image Classification with

OpenAI's

Contrastive Language-Image Pre-Training CLIP



ImageNet
(Russakovskyl et al., IJCV 2015)

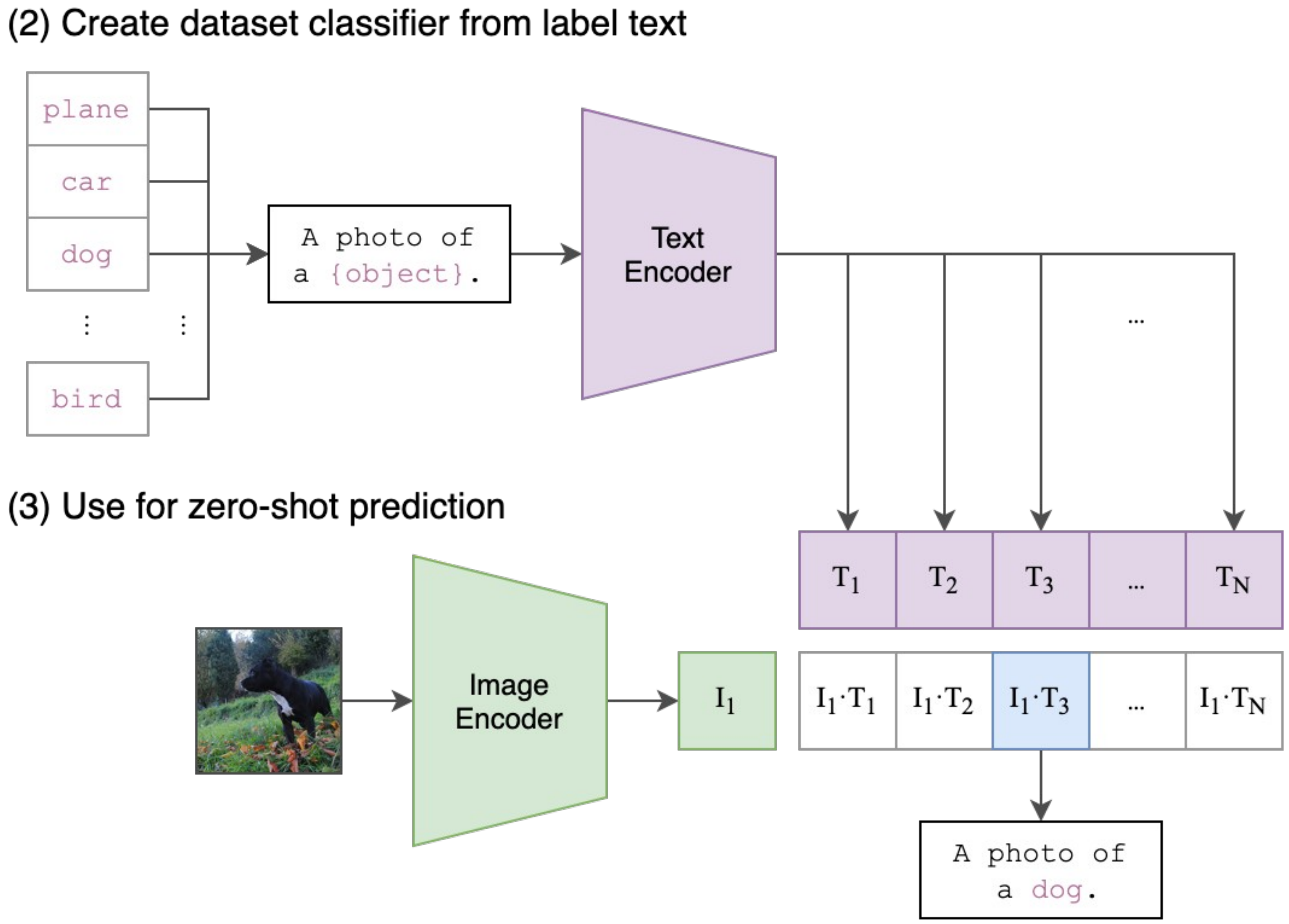
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero shot Image Classification with

OpenAI's

Contrastive Language-Image Pre-Training CLIP



Learning Transferable Visual Models From Natural Language Supervision
Alec Radford et al., ICLR 2022

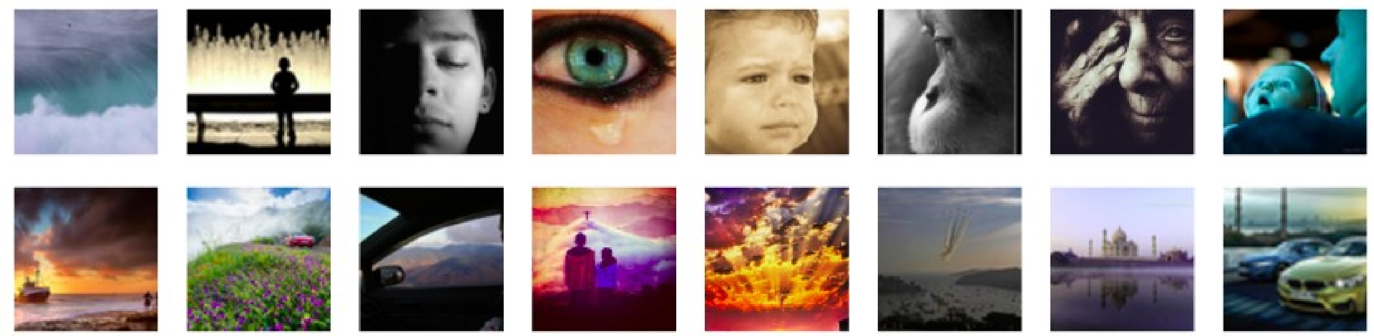
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero shot Image Classification with

OpenAI's

Contrastive Language-Image Pre-Training CLIP



(c) Awe

(d) Contentment

Learning Transferable Visual Models From Natural Language Supervision
Alec Radford et al., ICLR 2022

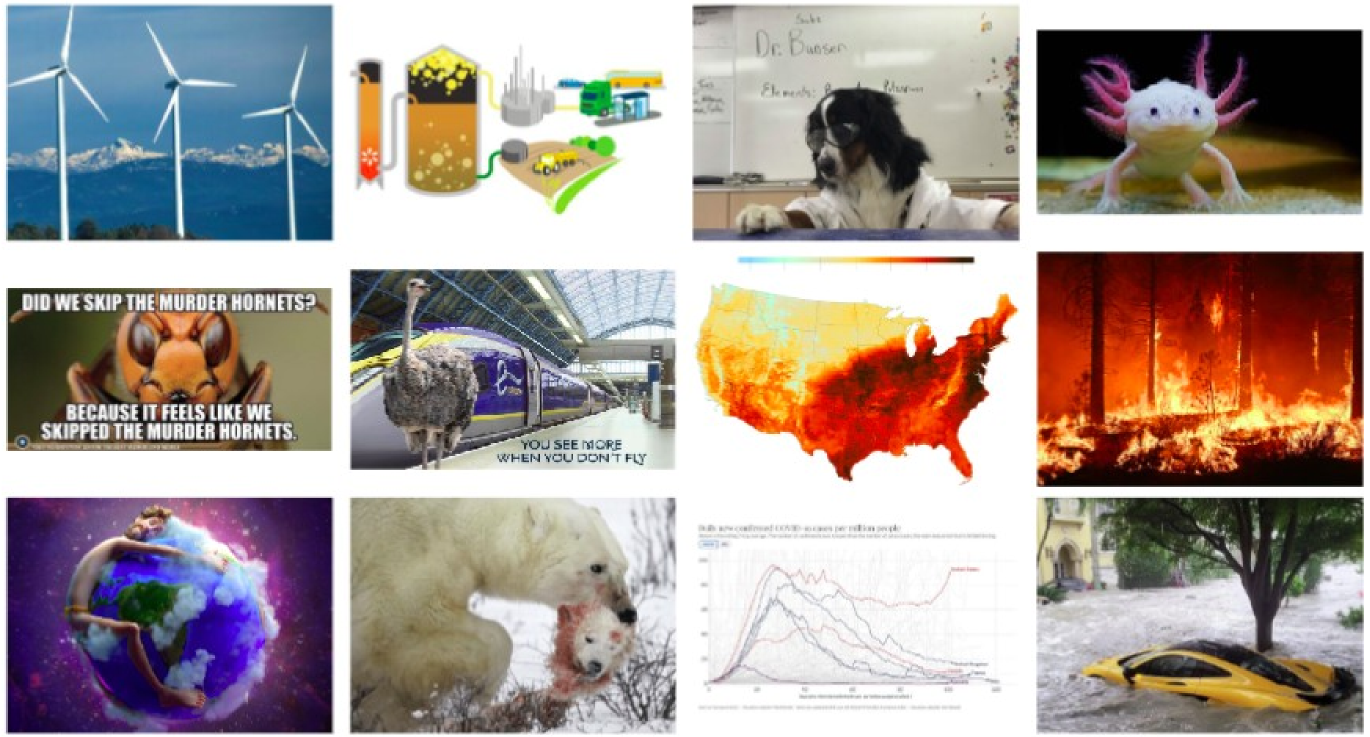
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero shot Image Classification with

OpenAI's

Contrastive Language-Image Pre-Training CLIP



Applicable to diverse classes?
Example: Understanding Climate Change Communications in Social Media.

Learning Transferable Visual Models From Natural Language Supervision
Alec Radford et al., ICLR 2022

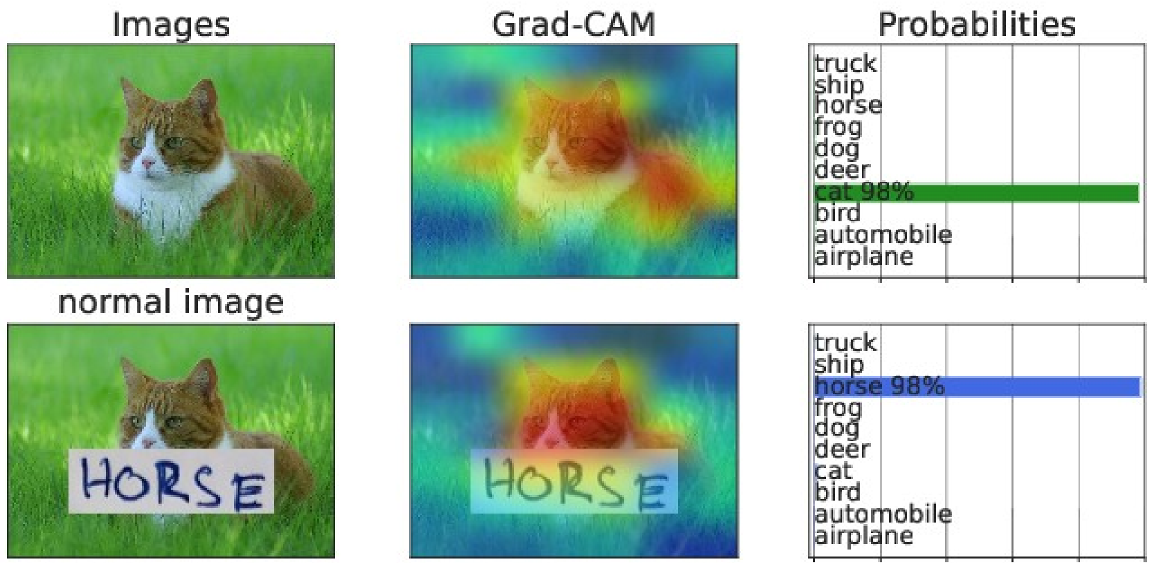
Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero shot Image Classification with

OpenAI's

Contrastive Language-Image Pre-Training CLIP



CLIP4STR: A Simple Baseline for Scene Text Recognition with Pre-trained Vision-Language Model

Learning Transferable Visual Models From Natural Language Supervision
Alec Radford et al., ICLR 2022

Machine Learning Success Story – Computer Vision

What is on the refrigerator?

magnet, paper

What is the color of the comforter?

blue, white

How many drawers are there?

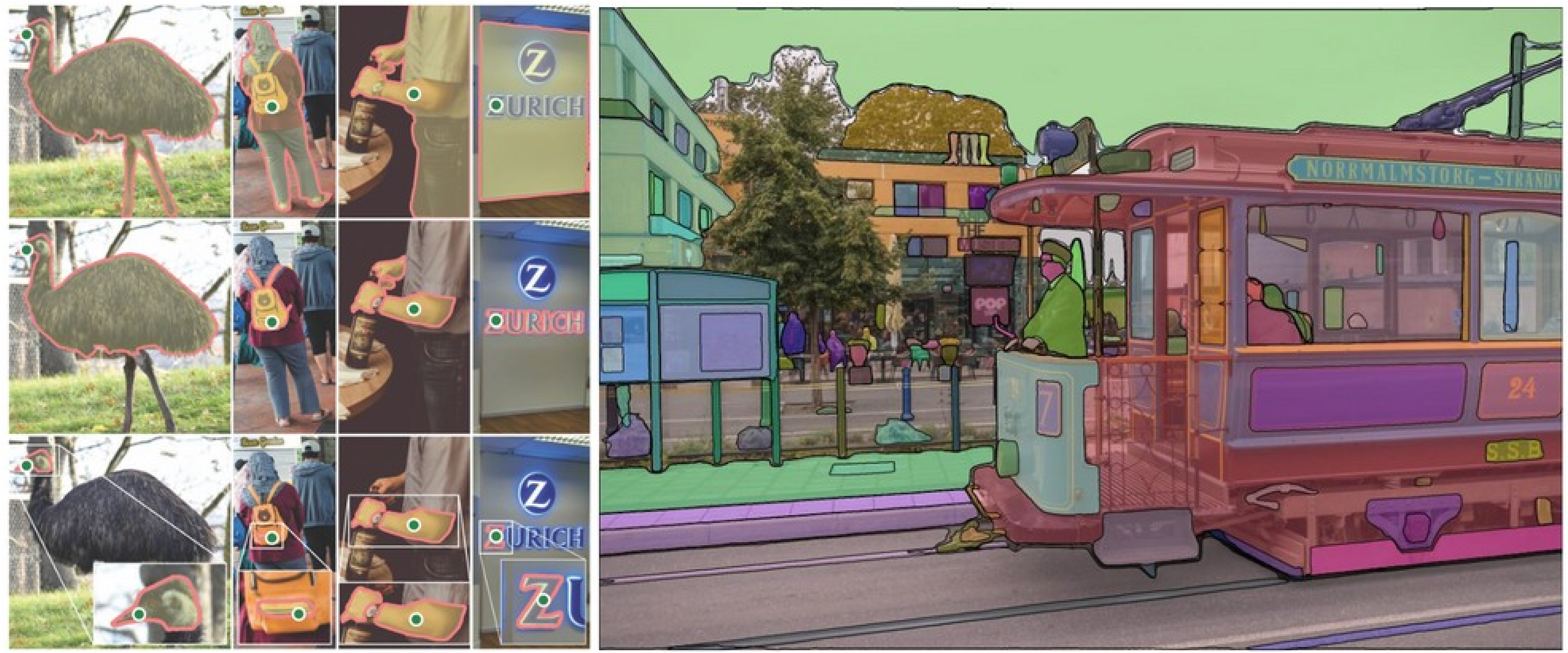
3

Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Zero-Shot Image Segmentation:

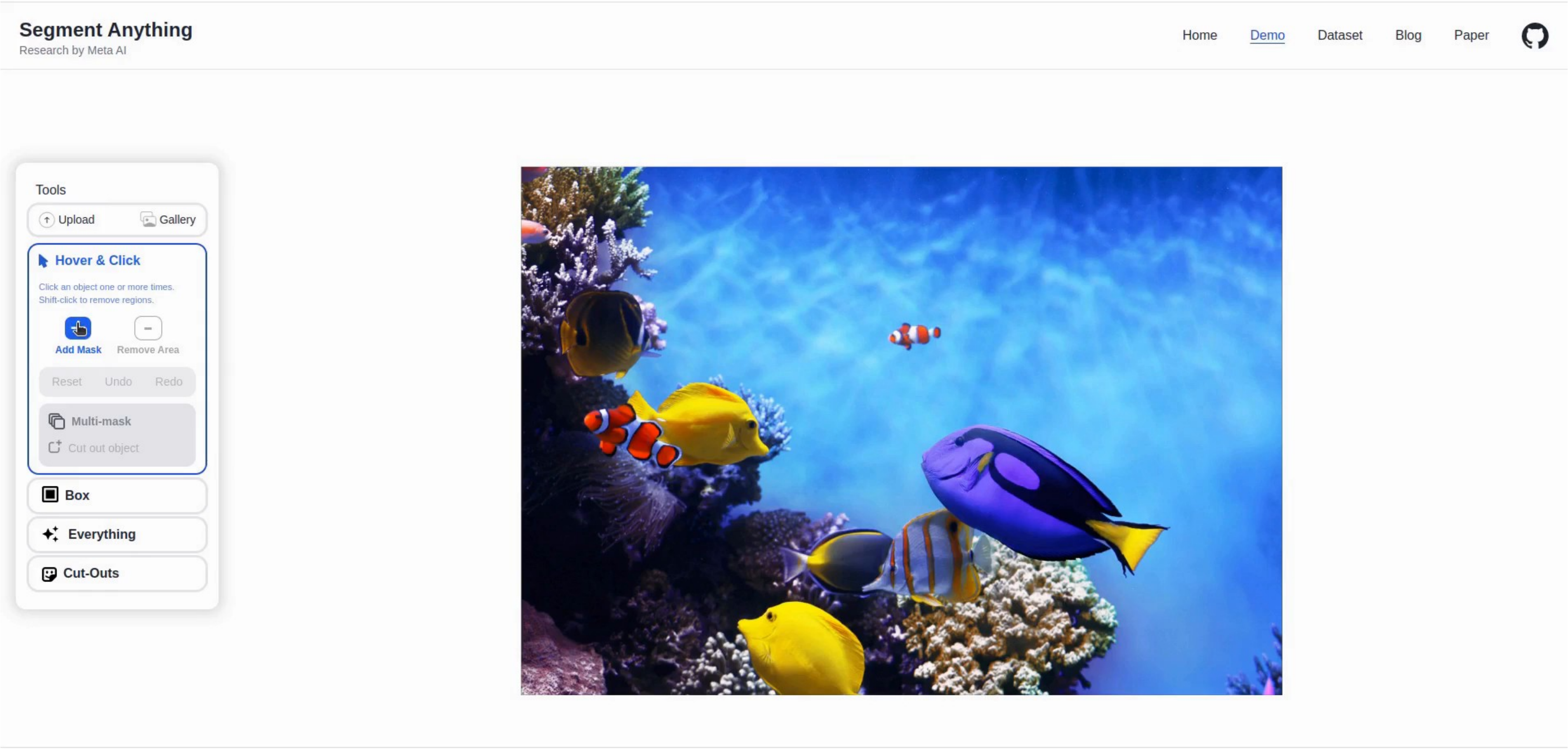
Meta's
Segment Anything



Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012.
Recently, large pre-trained models provide a further leap towards solving new tasks!

Meta’s
Segment Anything



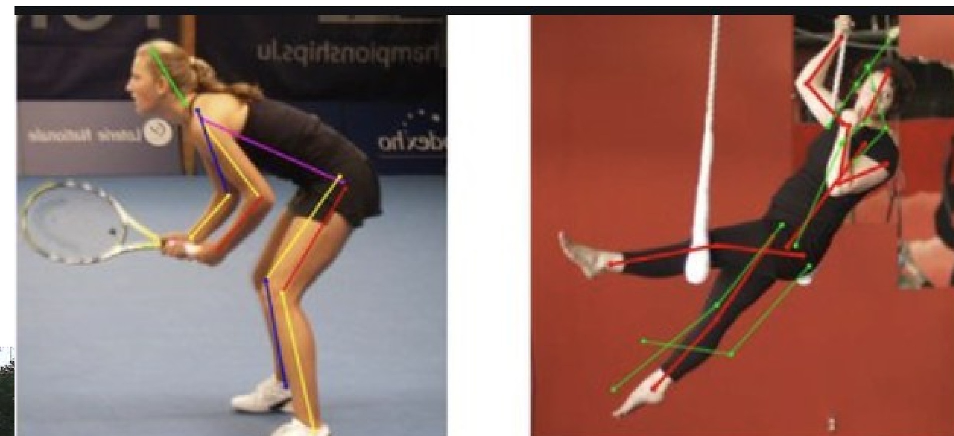
Demo video: https://learnopencv.com/segment-anything/

Machine Learning Success Story – Computer Vision

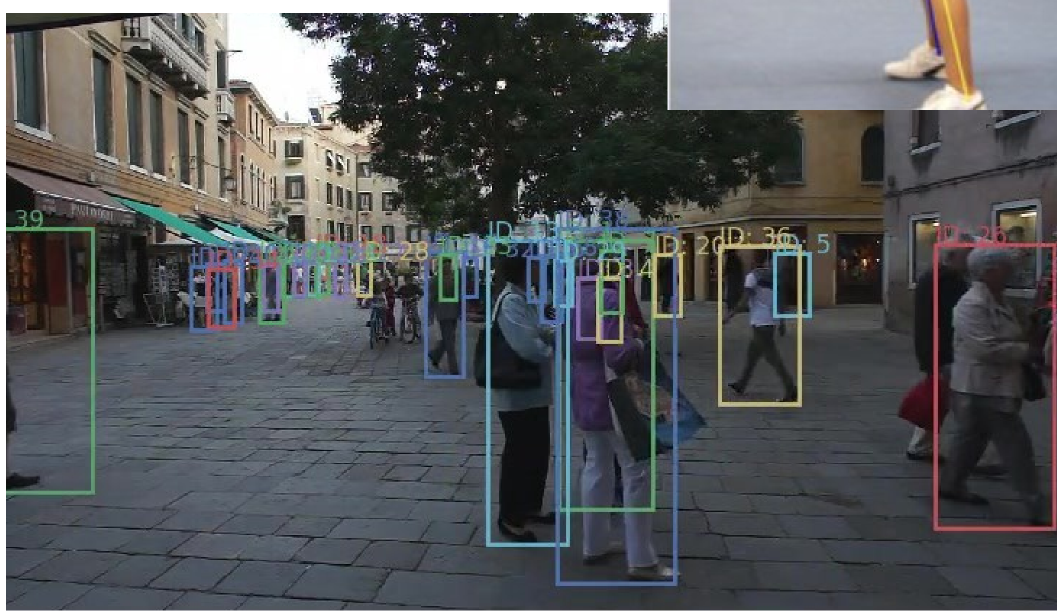
CV as been revolutionized by ML since 2012.

Reasonable Results for Object Tracking

Reasonable Results for Human Pose Estimation

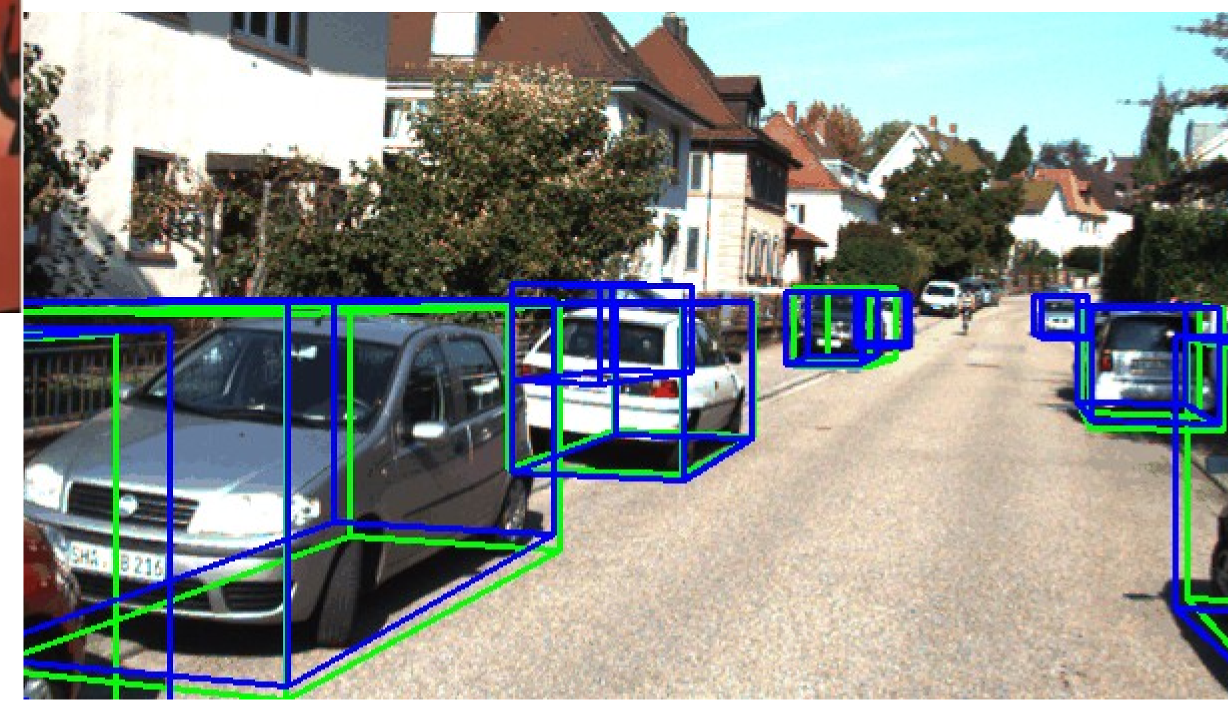


Leeds sports Pose, (Johnson et al., BMVC 2010)



MOT17 (Milan et al., 2016)

Reasonable Results for 3D Object Detection



KITTI (Geiger et al., CVPR 2012)

Machine Learning Success Story – CV Example

Optical Flow: Estimate Motion between neighboring frames.

Machine Learning Success Story – CV Example

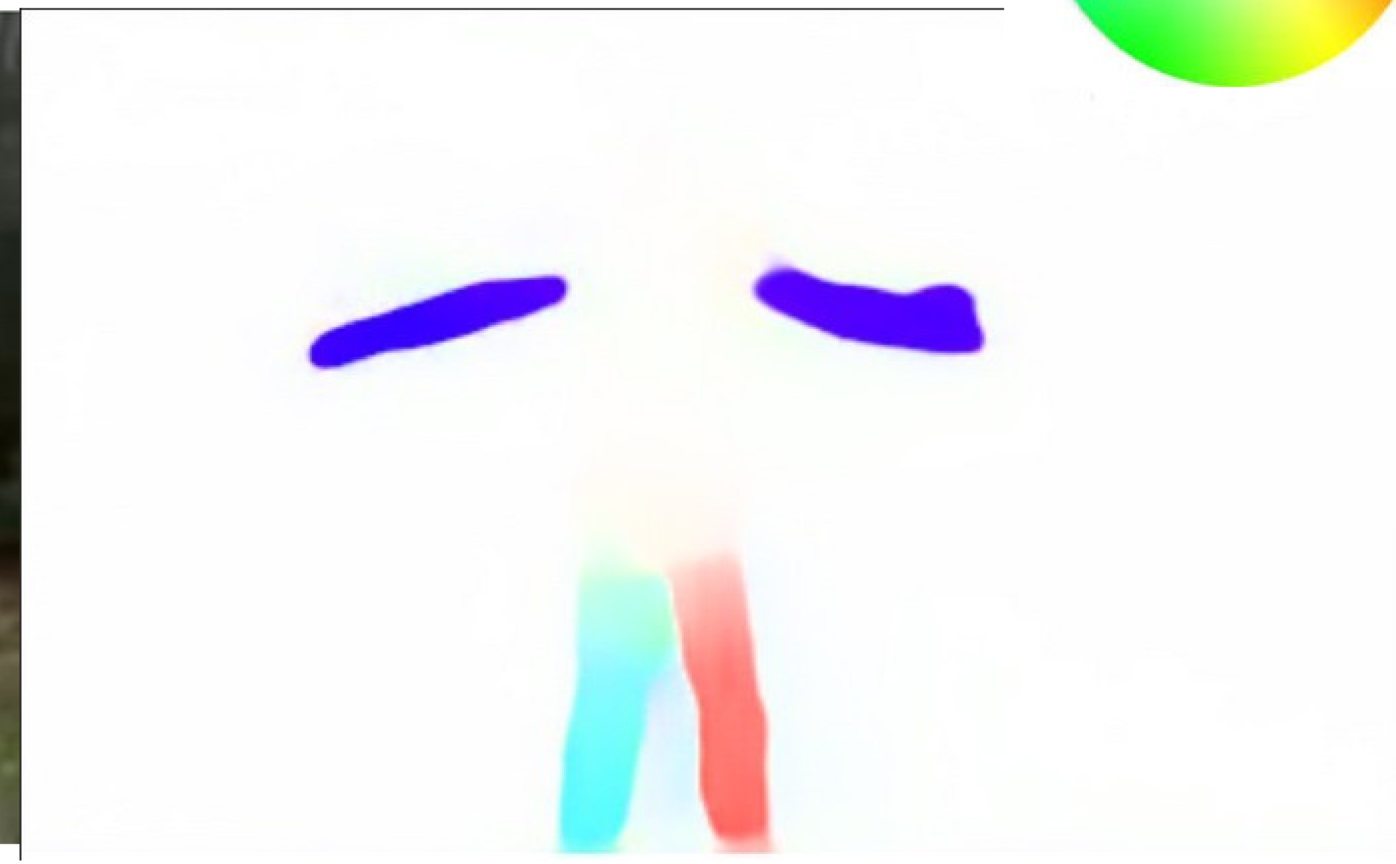
Optical Flow: Estimate Motion between neighboring frames.

Machine Learning Success Story

Optical Flow: Estimate Motion between neighboring frames.

Machine Learning Success Story

Optical Flow: Estimate Motion between neighboring frames.



E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks, CVPR'17

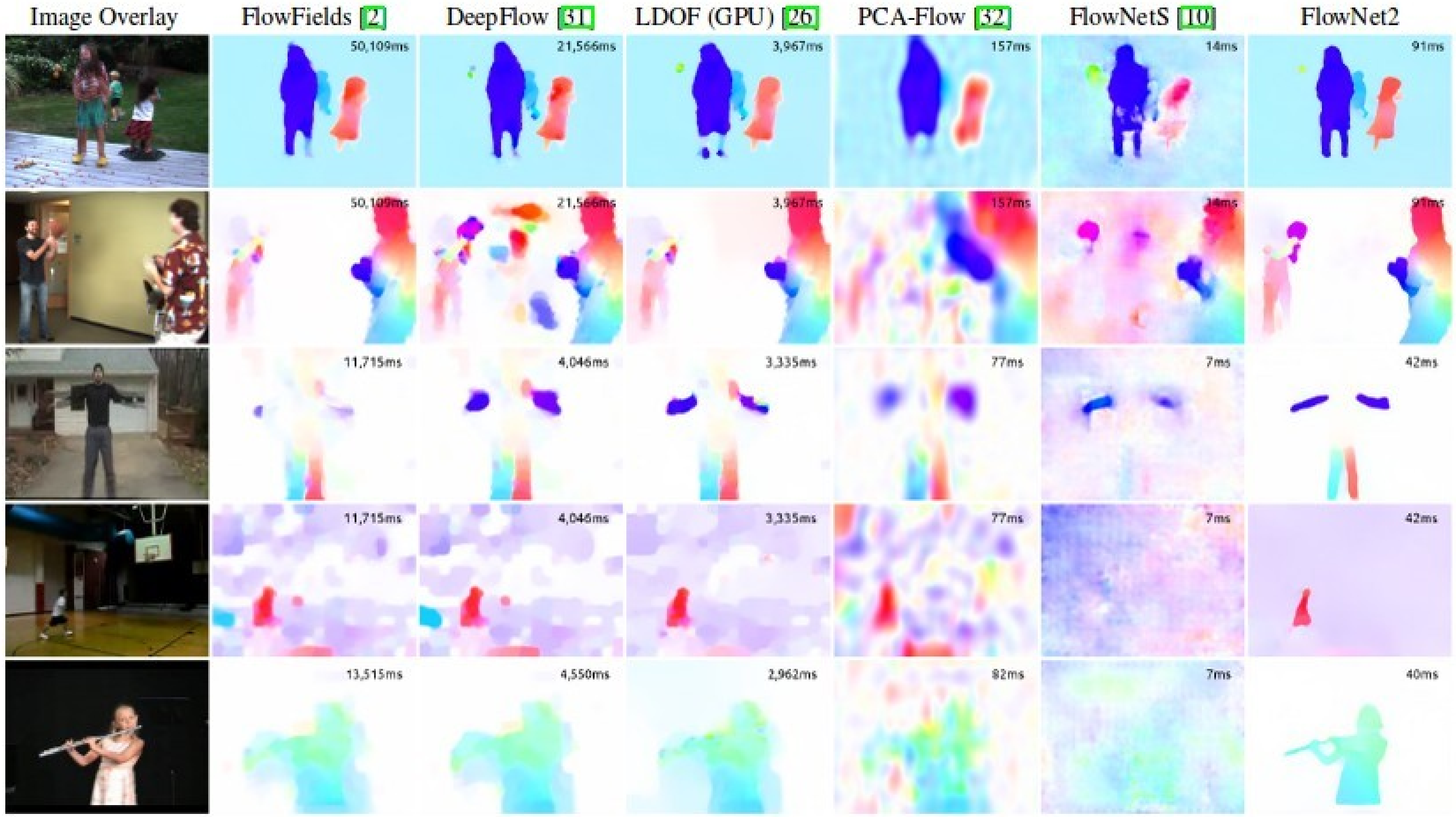
Machine Learning Success Story

Optical Flow: Estimate Motion between neighboring frames.

E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks, CVPR'17

Machine Learning Success Story

Optical Flow: Estimate Motion between neighboring frames.



Highly accurate in practice

Improved over the SotA in several benchmarks

Fast computation

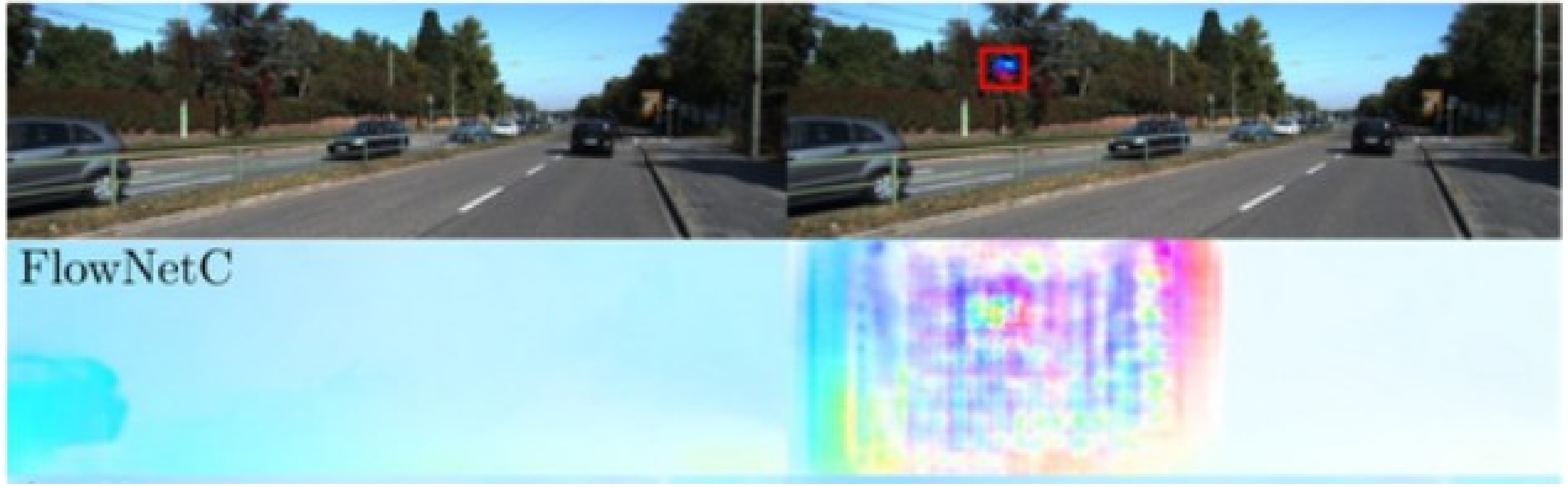
Several follow up papers (including our own)

Collected more than 3000 citations (google scholar)

E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, T. Brox, FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks, CVPR'17

Machine Learning Success Story

Optical Flow: Estimate Motion between neighboring frames.



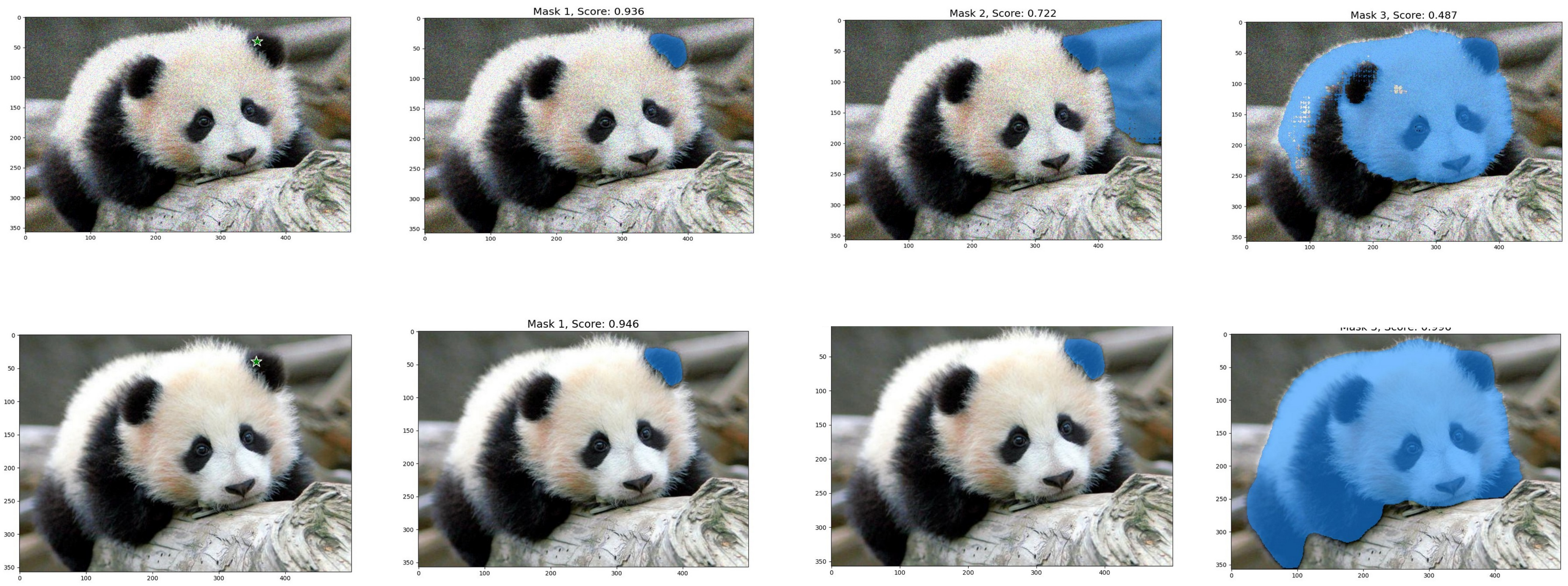
Ranjan et al., ICCV 2019

What if such models can be easily fooled by attackers?

What are implications for sustainable progress?

Machine Learning Success Story - Segment Anything?

- Machine learning (ML) is omnipresent in computer vision.
- To be successful, it needs to be reliable.



Shot noise

Clean data

Machine Learning Success Story

- Machine learning (ML) is omnipresent in computer vision.
- To be successful, it needs to be reliable.

Machine Learning Success Story – Computer Vision

CV as been revolutionized by ML since 2012

What about very specific application domains with little annotated data?

Prof. H. Kümper

(Universität Mannheim)

Benefit in many practical applications:

Example: Categorization of medieval seals with little or no supervision.

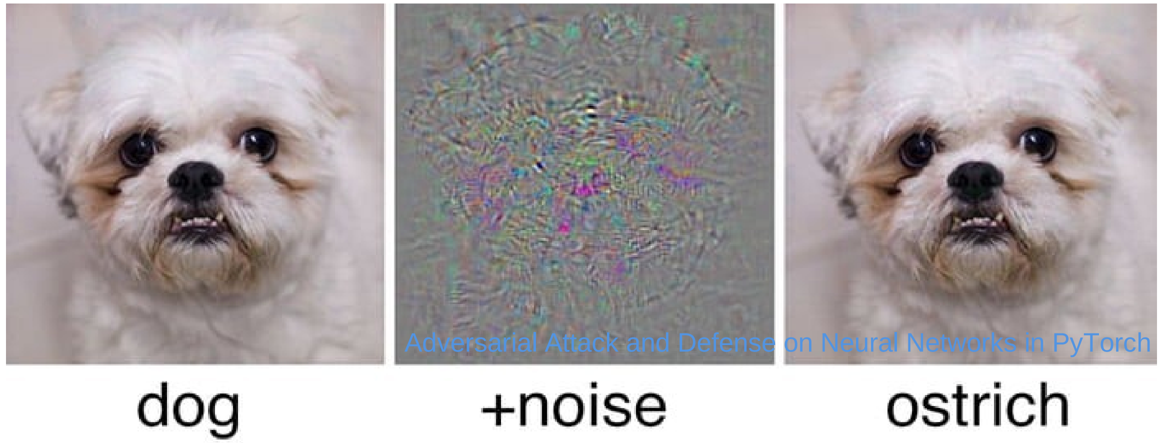
Collaboration with history department.

Machine Learning Success Story

- Machine learning (ML) is omnipresent in computer vision.
- To be successful, it needs to be reliable.
- It also needs to be easily adaptable to low data regimes

Machine Learning - Robustness

- Machine learning (ML) is omnipresent in computer vision.
- To be successful, it needs to be reliable.
  - **Robust Deep Learning for Computer Vision?**



Current models can be easily fooled by adversarial attacks.

Current models don’t generalize well to unseen domains (ACDC dataset)

How can we increase Robustness and Reliability in Neural Network Predictions?

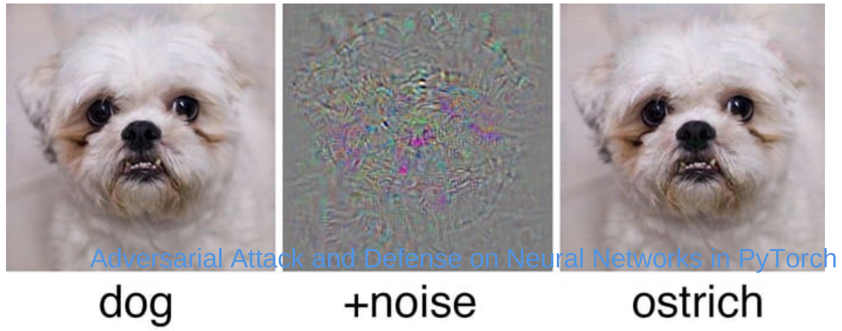
Towards Robust and Reliable Predictions in CNNs

What is Robustness?

Stable behavior when

- adding noise to the data.
- (slightly) corrupting the data.

(Low confidence when the label is wrong.)



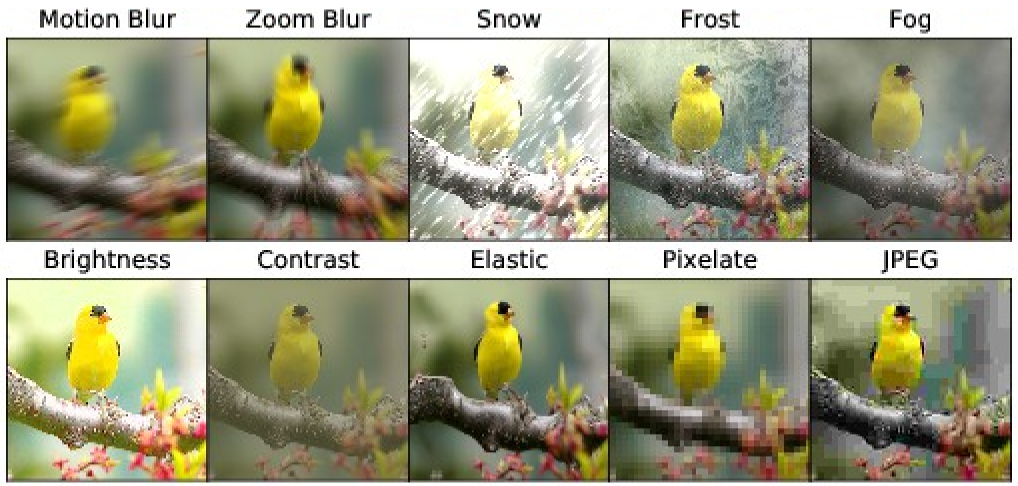
How can we measure robustness?

Evaluate on common corruptions.

Pro: simple and somewhat informative

Con: limited expressiveness

What happens in the worst case?



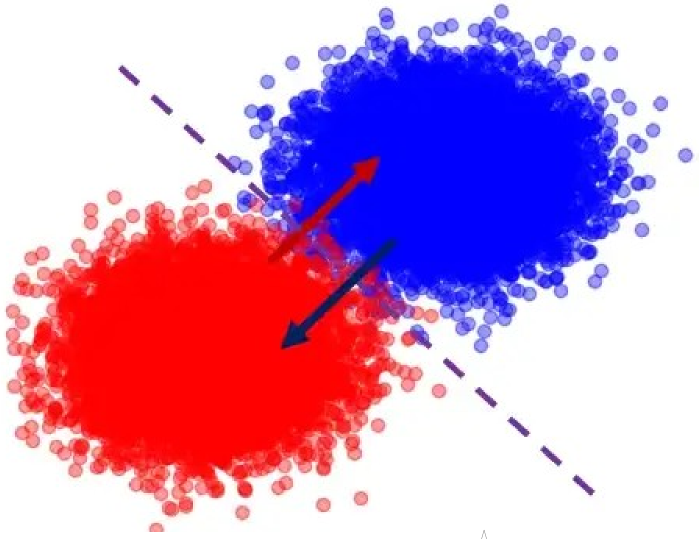
ImageNet-C, Hendrycks et al., ICLR'19

How can we measure robustness?

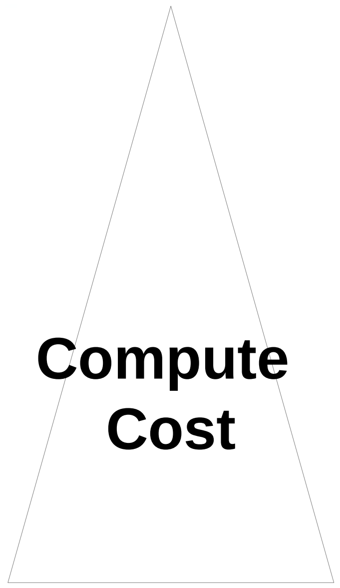
What happens in the worst case? Hardly perceivable input perturbation (within the epsilon-ball of the input), that flips the label.

Idea: Craft adversarial perturbations as a probe.

Optimize using full access to the model!



- **FGSM:** Single step white-box attack, not very strong
  (Goodfellow et al, ICLR‘15)
- **PGD:** Multi-step white-box attack, can be strong
  (Kurakin et al., ICLR workshop‘17)
- **AutoAttack:** Ensemble including adaptive PGD, even stronger
  (Croce et al., ICML‘20)



How can we increase robustness?

Hypothesis: The lack of robustness in NNs has **multiple causes**

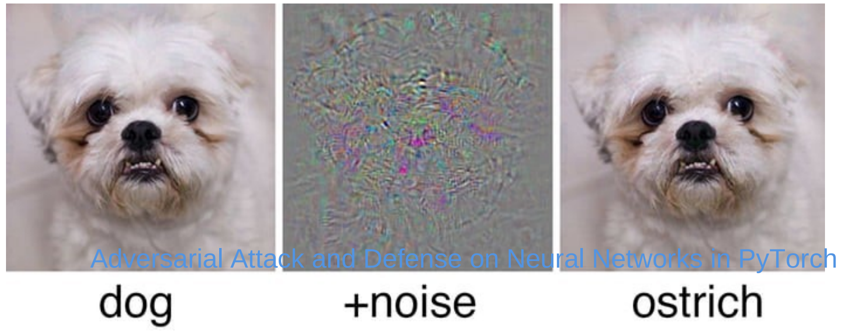
- Training Procedure

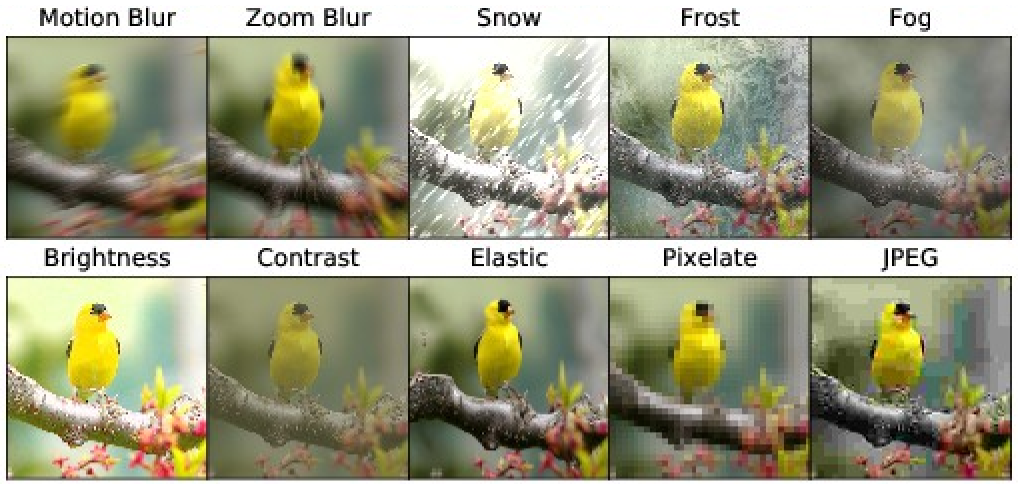
    Loss

    Data Augmentation

- Particular Architecture Design Components

- The overall NN Architecture





ImageNet-C, Hendrycks et al., ICLR'19

How can we increase robustness?

Hypothesis: The lack of robustness in NNs has **multiple causes**

- Training Procedure

    Loss

    Data Augmentation

- Particular Architecture Design Components

- The overall NN Architecture

Intra-Source Style Augmentation for Improved Domain Generalization, Li, Zhang, Keuper, Khoreva, WACV'23

How can we increase robustness?

Hypothesis: The lack of robustness in NNs has **multiple causes**

- Training Procedure

  Loss

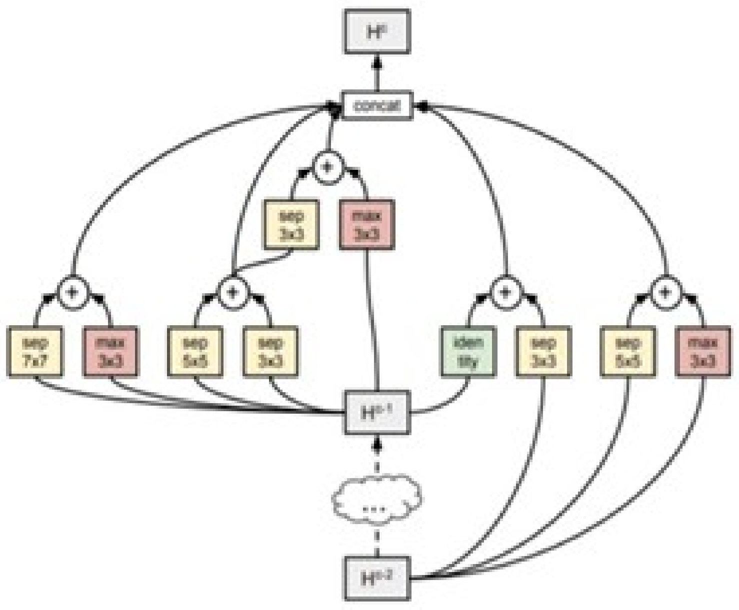
  Data Augmentation

- Particular Architecture Design Components

- The overall NN Architecture

  Today's perspective:

  Neural Architecture Search



Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, Fei-Fei L, Yuille A, Huang J, Murphy K, "Progressive Neural Architecture Search", ECCV 2018

Neural Architecture Search

Overall goal: Find high-scoring and efficient and robust networks

Limitations:

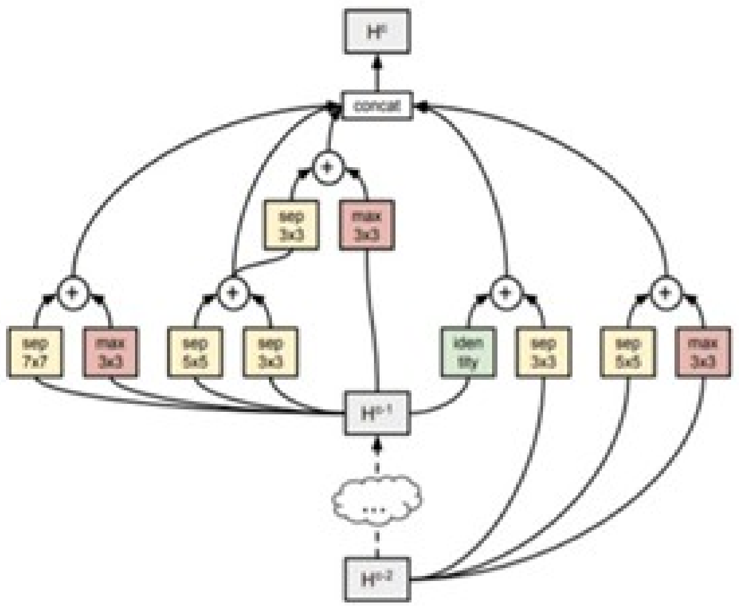
1. Time-consuming

2. Trial and error

3. Expert knowledge

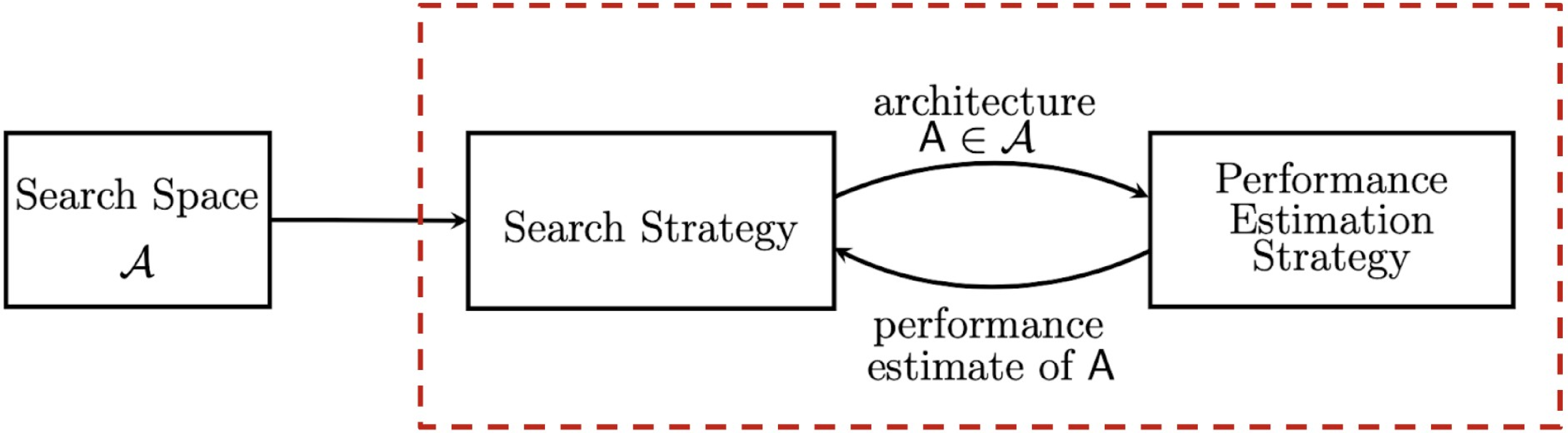
Consequence: **Automate the architecture design** process in terms of query efficiency

⇝ Neural Architecture Search



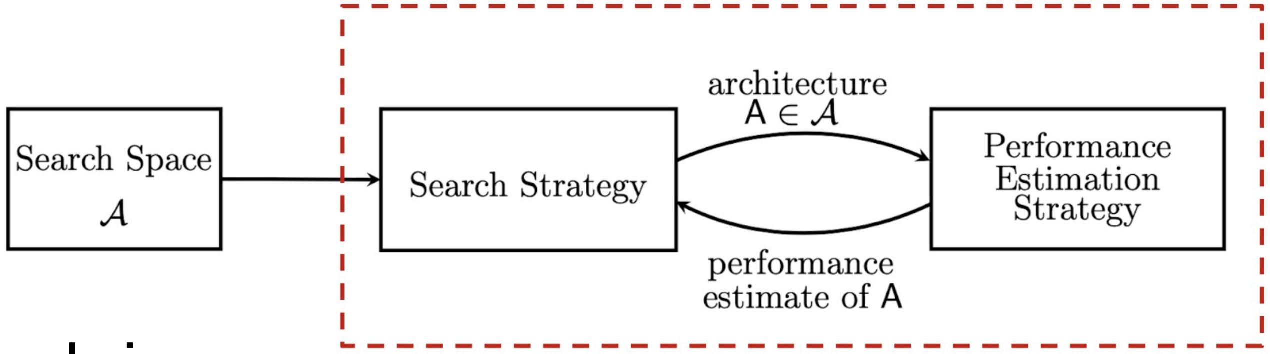
Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, Fei-Fei L, Yuille A, Huang J, Murphy K, “Progressive Neural Architecture Search”, ECCV 2018

Neural Architecture Search



Elsken et al, Neural Architecture Search: A Survey, 2018.

Neural Architecture Search



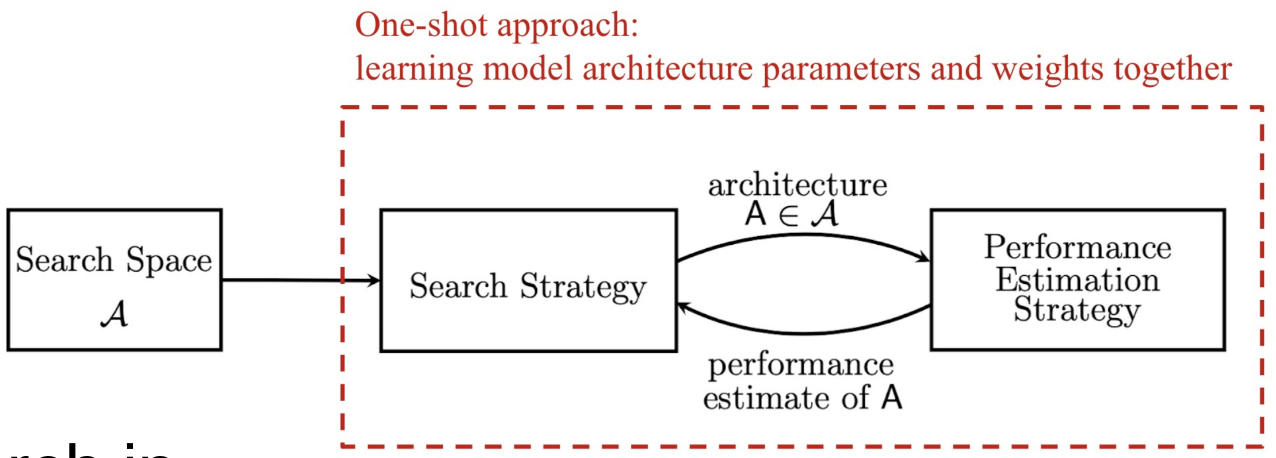
Elsken et al, Neural Architecture Search: A Survey, 2018.

Naive NAS methods are expensive

Speedup-techniques to improve the query-search in NAS

→ One query implies a full training of the architecture

Neural Architecture Search



Elsken et al, Neural Architecture Search: A Survey, 2018.

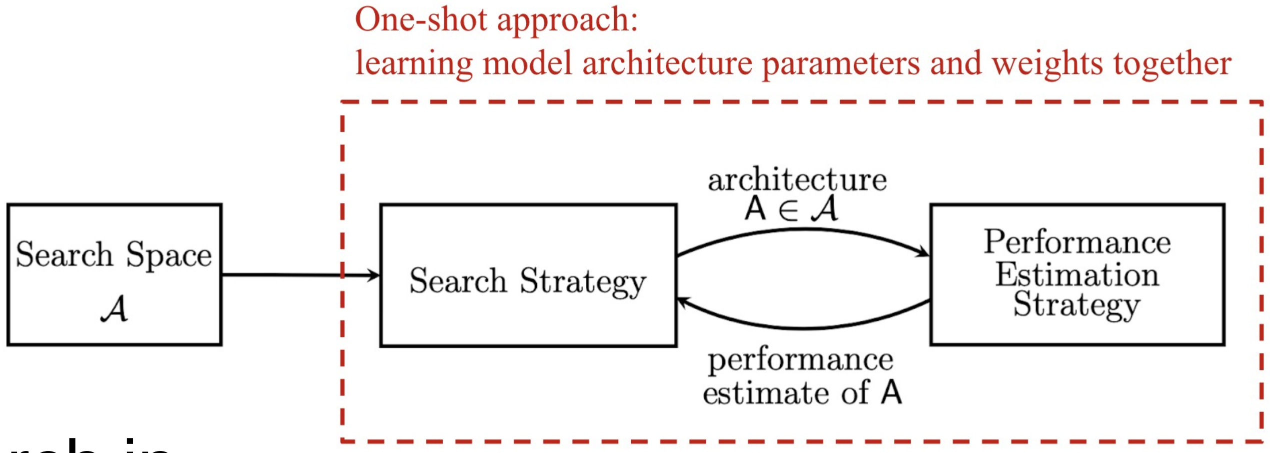
Naive NAS methods are expensive

Speedup-techniques to improve the query-search in NAS

Techniques to improve the query-search in NAS

- One-shot methods
- Predictor-based methods

Neural Architecture Search



Elsken et al, Neural Architecture Search: A Survey, 2018.

Naive NAS methods are expensive

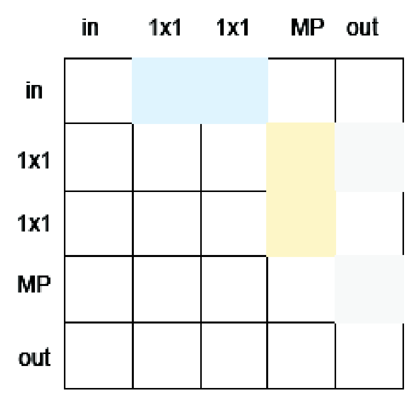
Speedup-techniques to improve the query-search in NAS

Techniques to improve the query-search in NAS

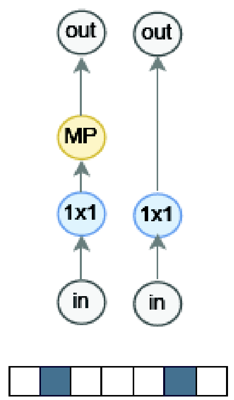
- One-shot methods
- Predictor-based methods

1. Predict performance of architectures before training them fully
2. Can be wrapped in different disguises

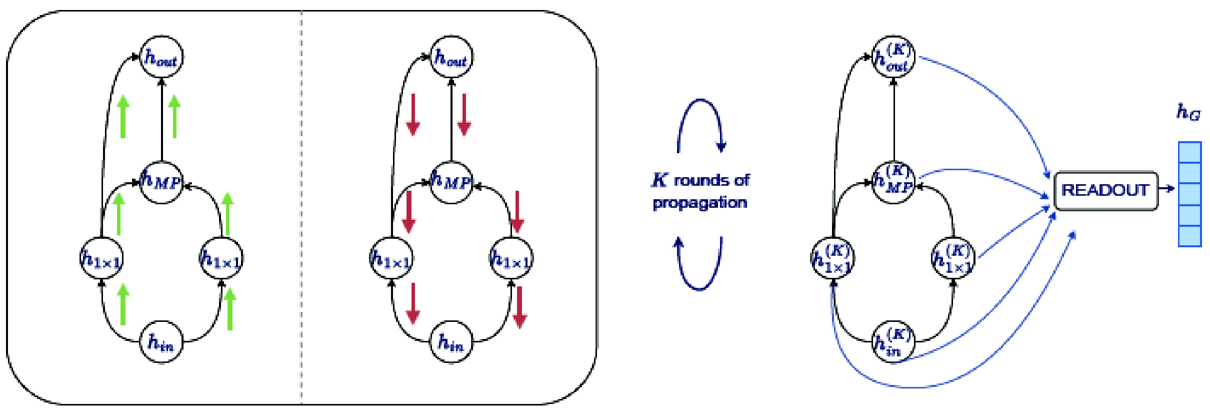
Neural Architecture Representations



(a) Adjacency matrix



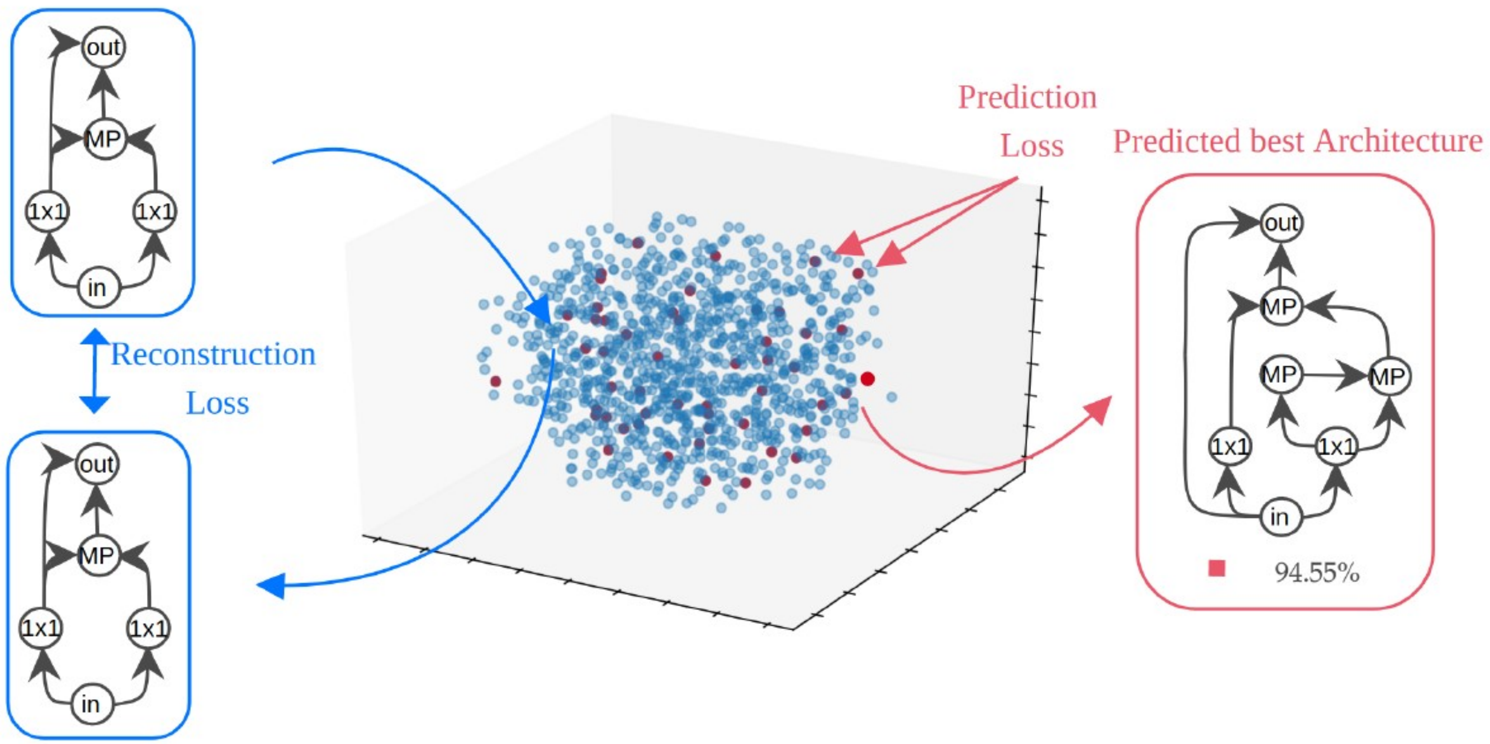
(b) Path-based encoding



(c) Learned graph encoding using a GNN encoder

- Scalability issues
- Task-dependent

Neural Architecture Representations



Create a latent space of neural architectures to facilitate more efficient NAS

Two-step approach:

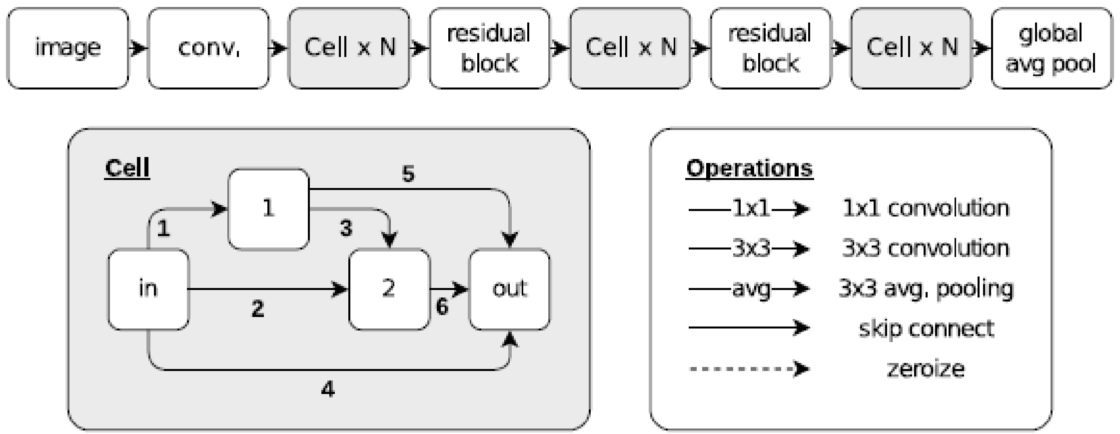
- **Unsupervised** learning of architecture latent space
- Allow for search approaches within latent space by means of generated architectures

Neural Architecture Search

Most common search space: Cell-based search space
Architectures are represented as directed acyclic graphs (DAGs)

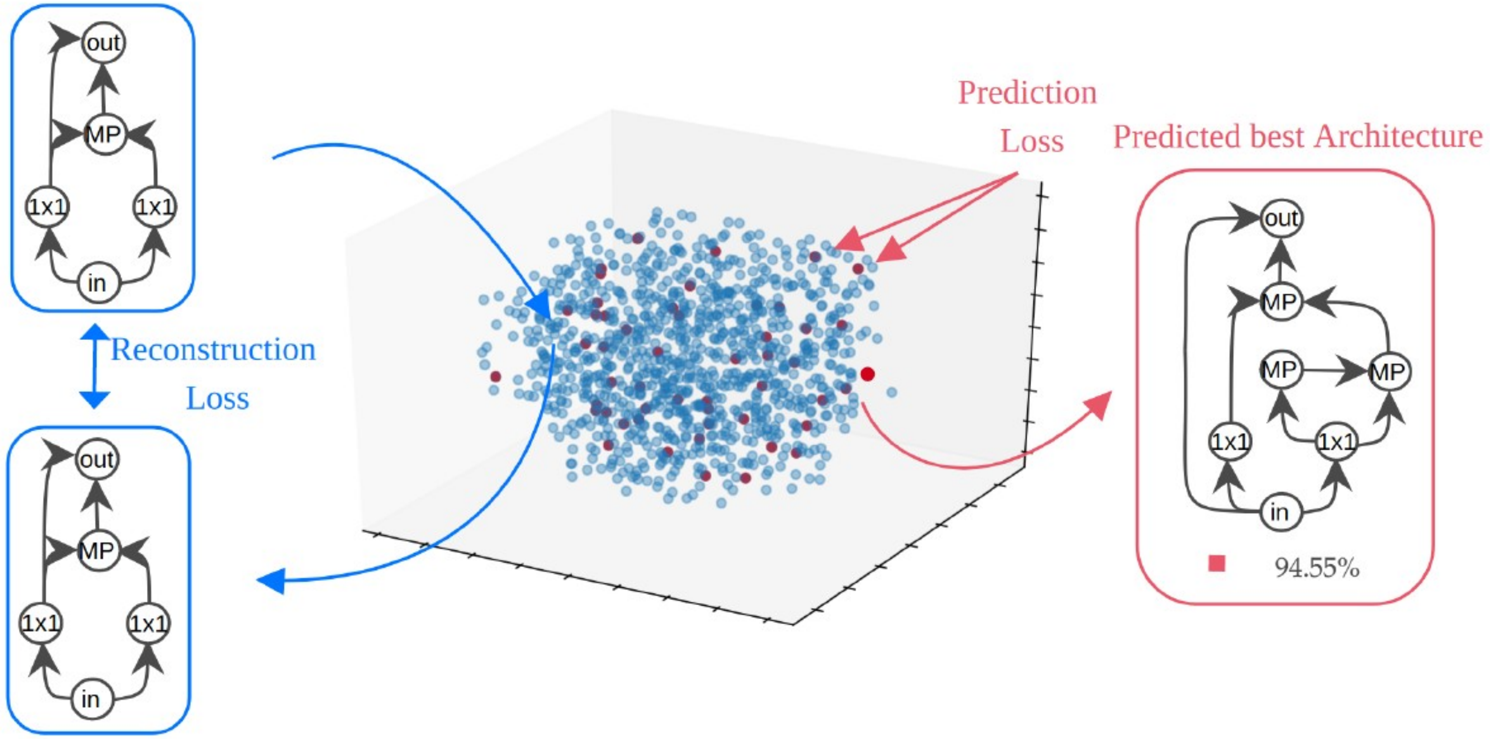
NAS-Bench-201:

- 4 nodes (features), 6 edges (operations),
- 5 predefined operations
- 15 625 (6 466 non-isomorphic) architectures
- Training Data:
  - CIFAR-10
  - CIFAR-100
  - Imagenet16-120
- Tabular benchmark



Dong et al. *NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search*. ICLR, 2020.

Neural Architecture Representations

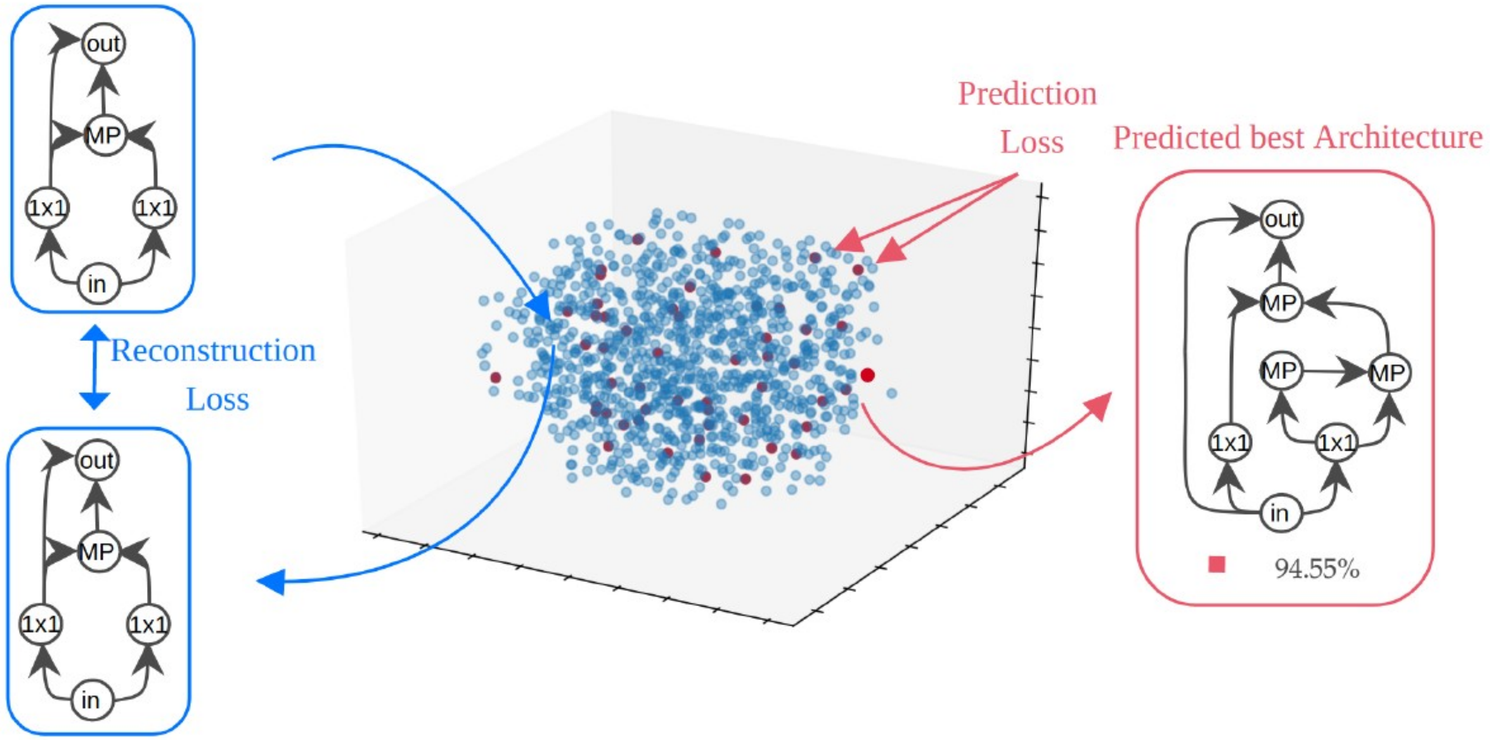


Create a latent space of neural architectures to facilitate more efficient NAS

Two-step approach:

- **Unsupervised** learning of architecture latent space
- Allow for search approaches within latent space by means of generated architectures
- Be able to generate **valid architectures** from the latent space

Generative NAS



Finetune latent space of generator to generate *well-performing* architectures.

Contributions:

- Efficient generator training: reconstruction loss without encoder
- First algorithm to optimize *directly* an NAS architecture latent space
- State of the art results on several NAS Benchmarks and on ImageNet
- Direct application to **multi-objective** optimization

Generative NAS

Graph Generator Network for NAS learns to generate architectures of a target space using backpropagation

Pretrained on search space

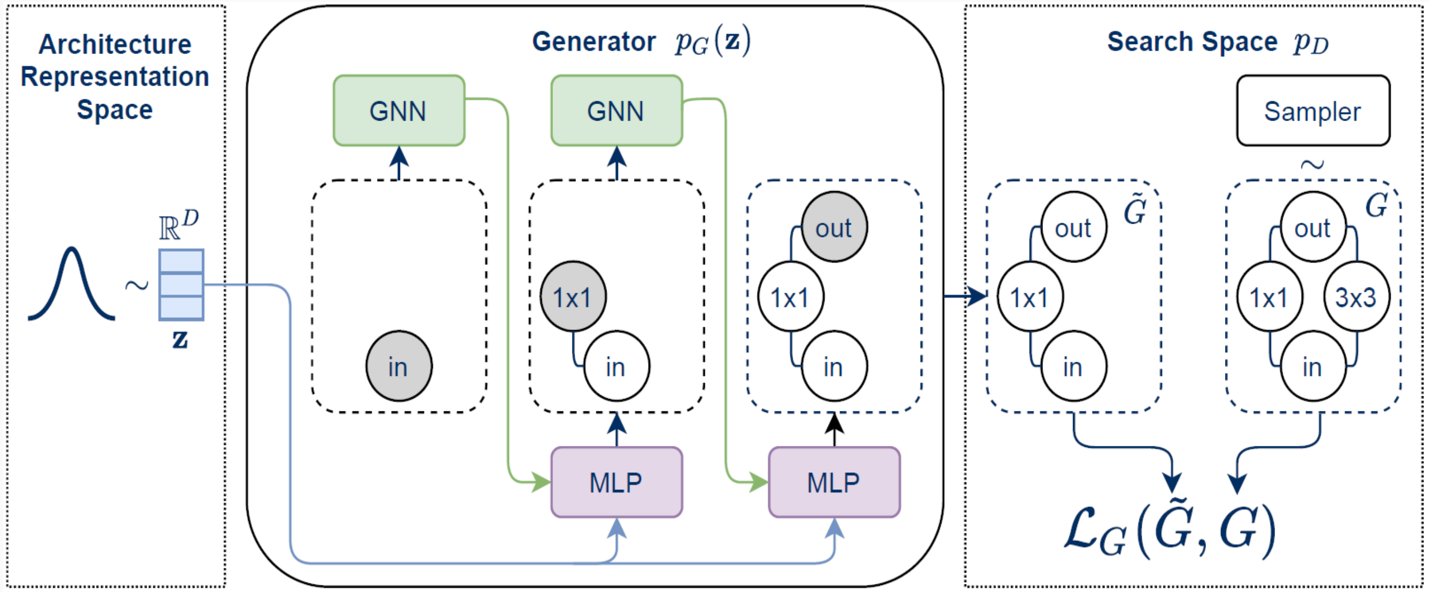


Fig. 2: Representation of the training procedure for our generator in AG-Net. The input is a randomly sampled latent vector $\mathbf{z} \in \mathbb{R}^d$. First, the input node is generated, initialized and input to a GNN to generate a partial graph representation. The learning process iteratively generates node scores and edge scores using $\mathbf{z}$ and the partial graph representation until the output node is generated. The target for this generated graph is a randomly sampled architecture.

Lukasik, Jung, Keuper: Learning where to look – Generative NAS is surprisingly efficient, ECCV 2022.

Generative NAS - Generator

Algorithm 3: Graph Generation

Input: $\mathbf{z} \sim \mathcal{N}(0, 1)$

Output: random sampled reconstructed graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$

initialize one-hot encoded InputNode v_0, with embedding
$\mathbf{h}_0 \leftarrow f_{\text{initNode}}(\mathbf{z}, f_{\text{Embedding}})[\text{InputType}])$

$V \leftarrow \{v_0\}$, $E \leftarrow \emptyset$, $\mathbf{h}_G \leftarrow \mathbf{z}$,

while $|V| \leq$ *Max Number of Nodes* **do**

- $v_{t+1} \leftarrow f_{\text{addNode}}(\mathbf{z}, \mathbf{h}_G)$
- $V \leftarrow V \cup \{v_{t+1}\}$
- $\mathbf{h}_{t+1} \leftarrow f_{\text{initNode}}(\mathbf{z}, \mathbf{h}_G, f_{\text{Embedding}}(v_{t+1})])$
- **for** $v_j \in V \setminus v_{t+1}$ **do**
  - $s_{\text{addEdges}}(j, t+1) \leftarrow f_{\text{addEdges}}(\mathbf{h}_{t+1}, \mathbf{h}_t, \mathbf{h}_G, \mathbf{z})$
  - $e_{(j,t+1)} \sim \text{Eval}(s_{\text{addEdges}}(j, t+1))$; ▷ evaluate whether to add edge
  - **if** $e_{(j,t+1)} = 1$ **then**
    - $E \leftarrow E \cup \{e_{(j,t+1)} = (v_j, v_{t+1})\}$
  - **end**
- **end**
- $\mathbf{h}_t \leftarrow \text{concat}(\mathbf{h}_t, \mathbf{h}_{t+1})$
- $G \leftarrow (V, E)$
- $\mathbf{h}_t \leftarrow (\mathbf{h}_t, G)$; ▷ update node embeddings
- $\mathbf{h}_G \leftarrow \text{aggregate}(\mathbf{h}_t)$; ▷ update graph embedding
- $t \leftarrow t + 1$

end

$V \sim \text{Categorical}(V)$; ▷ Sample node types

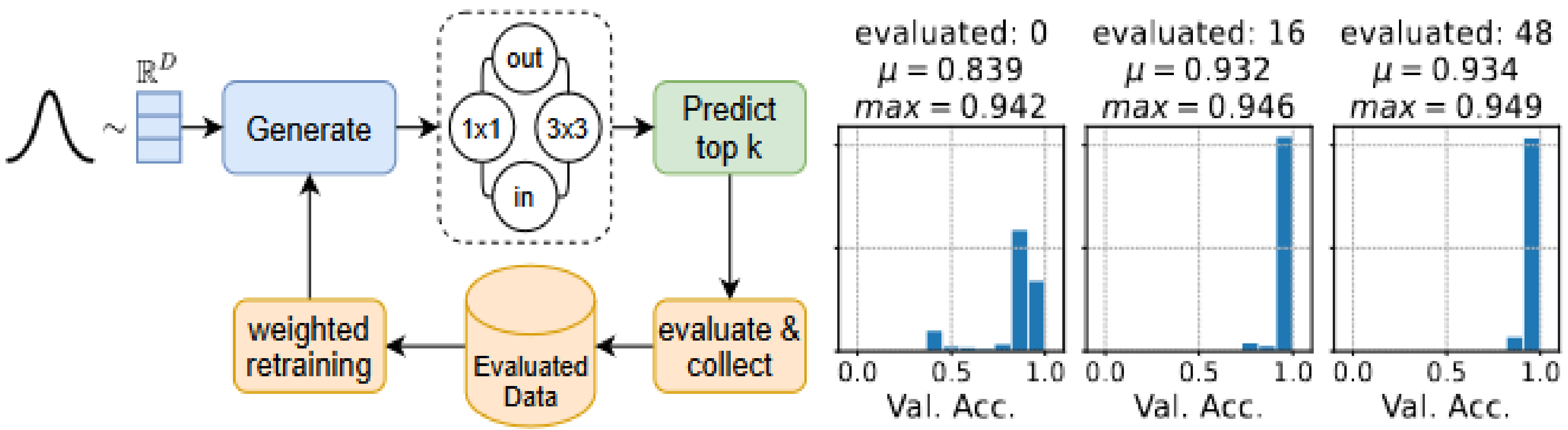
$E \sim \text{Ber}(E)$; ▷ Sample edges

$\widetilde{G} = (V, E)$

Generative NAS - Search

Use pretrained generative model and couple with the surrogate model for target predictions, e.g. accuracy, robustness, latency, etc.

This model is fully differentiable, which enables a stronger coupling with the target for the generation process



Algorithm 1: Unconstrained Search Algorithm

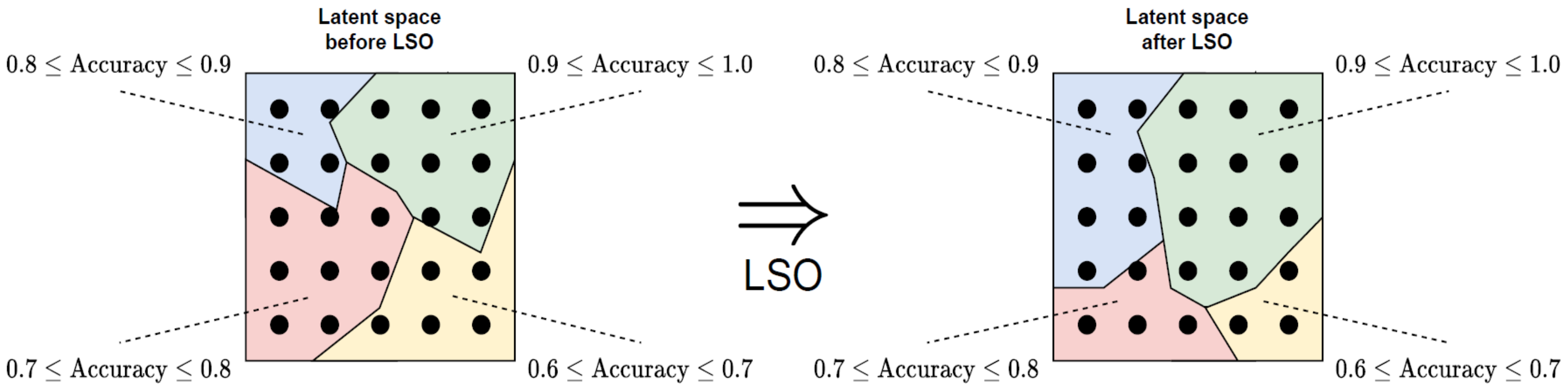
Input: (i) Search space p_D
Input: (ii) Pretrained generator G
Input: (iii) Untrained performance predictor P
Input: (iv) Query budget b
Input: (v) e epochs to train G and P

▷ `Initialize training data`
$\mathrm{D} \leftarrow \{\}$
while $|\mathrm{D}| < 16$ **do**
  $\mathrm{D} \leftarrow \mathrm{D} \cup \{d \sim p_D\}$
end
▷ `Evaluate architectures (get accurracies on target image dataset)`
$\mathrm{D} \leftarrow \mathrm{eval}(\mathrm{D})$
▷ `Randomly initialize predictor weights`
$P \leftarrow \mathrm{init}(P)$
▷ `Search loop`
while $|\mathrm{D}| < b$ **do**
  ▷ `Weight training data by performance`
  $\mathrm{D_w} \leftarrow \mathrm{weight}(\mathrm{D})$
  ▷ `Train generator and predictor`
  $\mathrm{train}(G, P, \mathrm{D_w}, e)$
  ▷ `Generate 100 candidates`
  $\mathrm{D_{cand}} \leftarrow \{\}$
  while $|\mathrm{D}_{cand}| < 100$ **do**
    $\mathrm{z} \sim \mathcal{U}(-3, 3)$
    $\mathrm{D_{cand}} \leftarrow \mathrm{D_{cand}} \cup G(\mathrm{z})$
  end
  ▷ `Select top 16 candidates with P`
  $\mathrm{D_{cand}} \leftarrow \mathrm{select}(\mathrm{D_{cand}}, P, 16)$
  ▷ `Evaluate and add to data`
  $\mathrm{D} \leftarrow \mathrm{D} \cup \mathrm{eval}(\mathrm{D_{cand}})$
end

Generative NAS – Improving the Latent Space

Optimize architecture representation space via **weighted retraining** (Tripp et al, 2020): weight training data and loss

$$w(G; p_D, k) \propto \frac{1}{kN + \mathrm{rank}_{f,p_D}(G)}$$



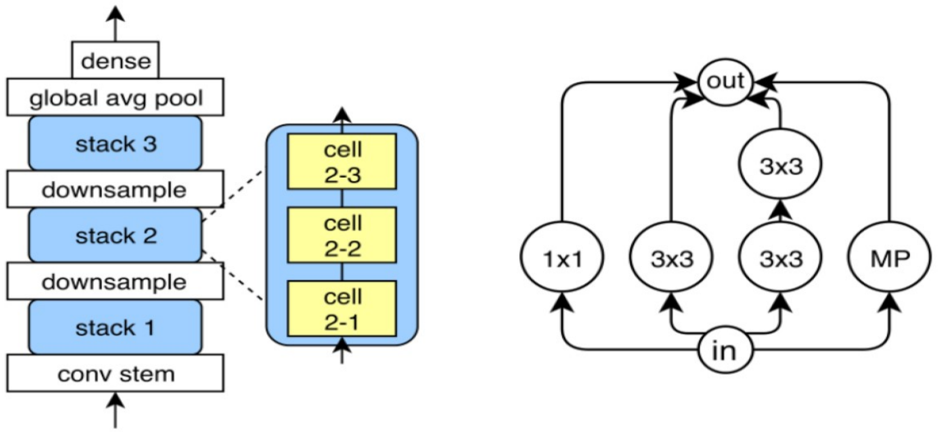
How to evaluate?

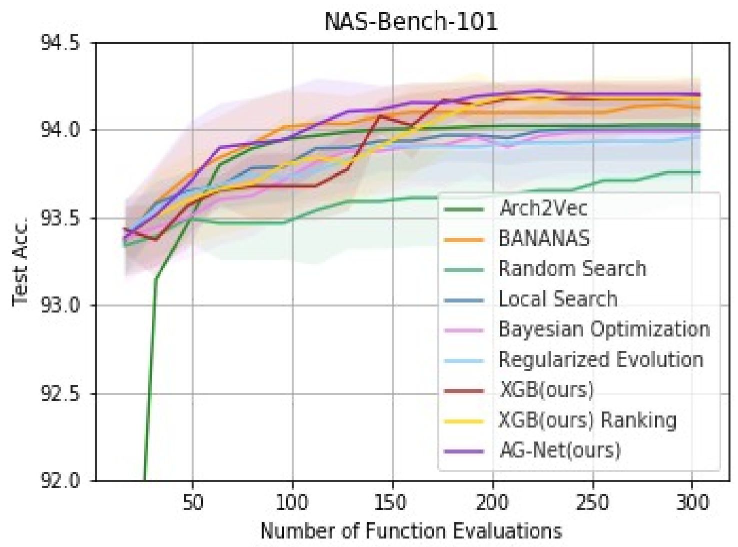
- **NAS-Bench-101**: 423,624 architectures evaluated on CIFAR-10
- **NAS-Bench-201**: 15,625 evaluated on CIFAR-10, CIFAR-100, and ImageNet-16
- **NAS-Bench-301**: ~ 60k sampled and evaluated architectures on CIFAR-10 in the DARTS search space()
- **NAS-Bench-NLP**: 14,322 sampled and evaluated architectures on Penn TreeBank ()
- **Hardware-Aware NAS-Bench**: NAS-Bench-201 search space, but we have latencies for different devices
  - Joint optimization (joint=1)
  - Constrained Optimization (joint=0)

Robustness?

Small to medium sized CNNs

NAS-Bench-101: 423,624 architectures evaluated on CIFAR-10



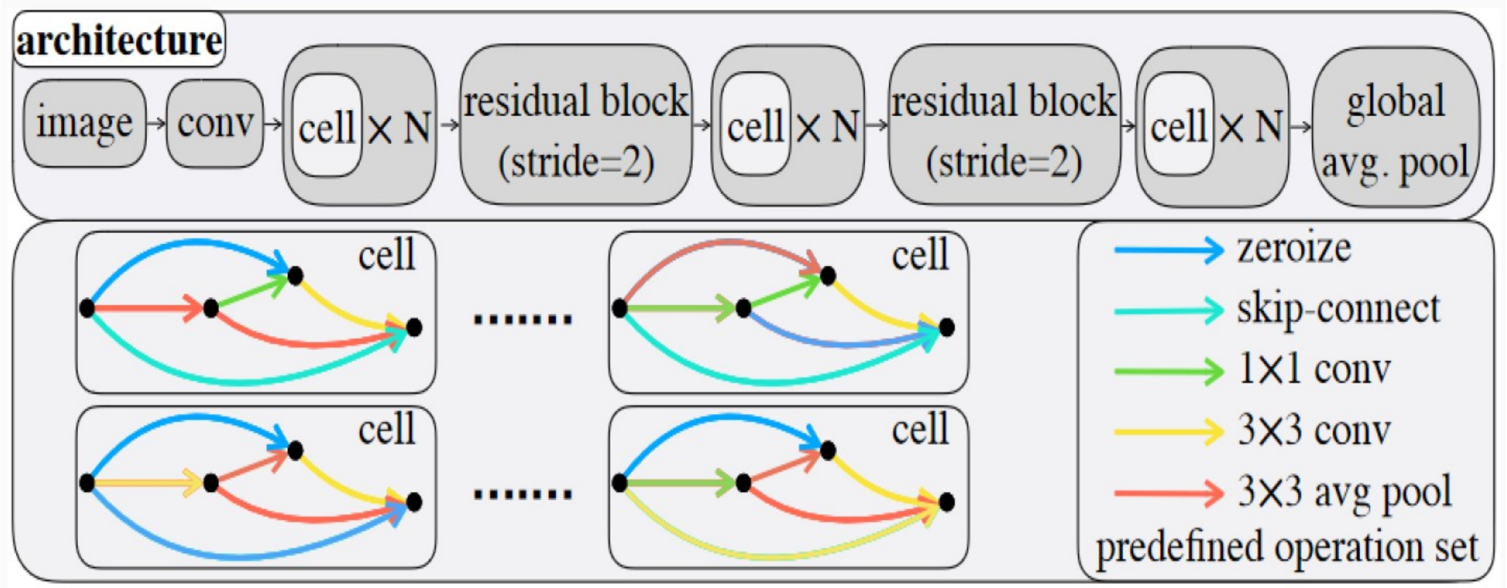


NAS Method	Val. Acc (%)	Test Acc (%)	Queries
Optimum*	95.06	94.32	
Arch2vec + RL [65]	-	94.10	400
Arch2vec + BO [65]	-	94.05	400
NAO ‡ [36]	94.66	93.49	192
BANANAS† [56]	94.73	94.09	192
Bayesian Optimization† [50]	94.57	93.96	192
Local Search† [57]	94.57	93.97	192
Random Search† [31]	94.31	93.61	192
Regularized Evolution† [42]	94.47	93.89	192
WeakNAS [60]	-	**94.18**	200
XGB (ours)	94.62	94.14	192
XGB + ranking (ours)	94.60	94.14	192
AG-Net (ours)	**94.90**	**94.18**	192

Table 1: Results on NAS-Bench-101 for the search of the best architecture in terms of validation accuracy on CIFAR-10 to state-of-the-art methods (mean over 10 trials).

Small CNNs on several datasets

NAS-Bench-201: 15,625 evaluated on CIFAR-10, CIFAR-100, and ImageNet-16



Small CNNs on several datasets

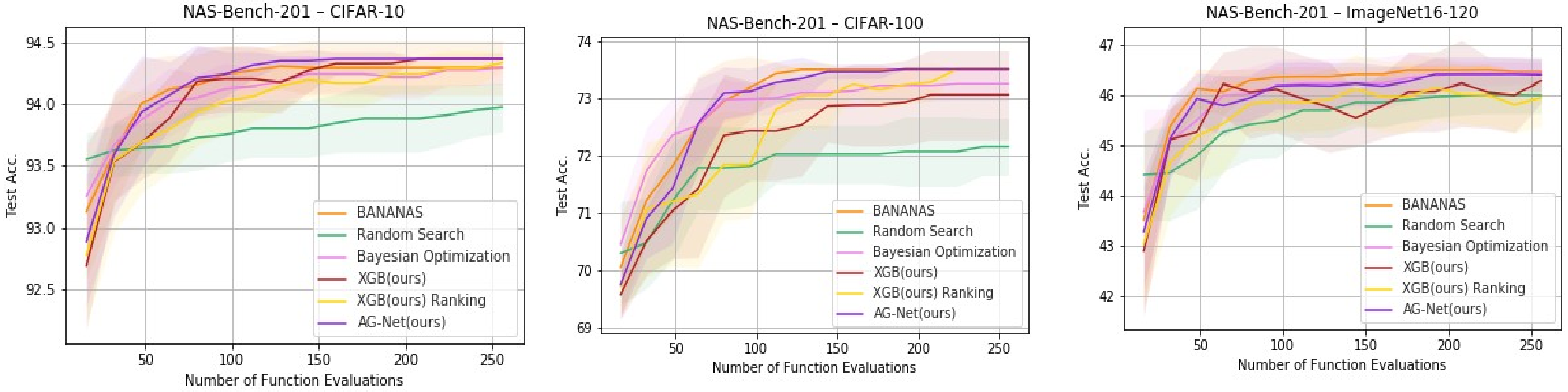
NAS-Bench-201: 15,625 evaluated on CIFAR-10, CIFAR-100, and ImageNet-16

NAS Method	CIFAR-10 Val. Acc	CIFAR-10 Test Acc	CIFAR-100 Val. Acc	CIFAR-100 Test Acc	ImageNet16-120 Val. Acc	ImageNet16-120 Test Acc	Queries	Search Method
Optimum*	91.61	94.37	73.49	73.51	46.77	47.31		
SGNAS [23]	90.18	93.53	70.28	70.31	44.65	44.98		Supernet
Arch2vec + BO [65]	91.41	94.18	**73.35**	**73.37**	46.34	46.27	100	Bayesian Optimization
AG-Net (ours)	**91.55**	**94.24**	73.2	73.12	46.31	46.2	96	Generative LSO
AG-Net (ours, topk=1)	91.41	94.16	73.14	73.15	**46.42**	**46.43**	100	Generative LSO
BANANAS† [56]	91.56	94.3	**73.49***	73.50	**46.65**	**46.51**	192	Bayesian Optimization
BO† [50]	91.54	94.22	73.26	73.22	46.43	46.40	192	Bayesian Optimization
RS † [31]	91.12	93.89	72.08	72.07	45.87	45.98	192	Random
XGB (ours)	91.54	94.34	73.10	72.93	46.48	46.08	192	Generative LSO
XGB + Ranking (ours)	91.48	94.25	73.20	73.24	46.40	46.16	192	Generative LSO
AG-Net (ours)	**91.60**	**94.37***	**73.49***	**73.51***	46.64	46.43	192	Generative LSO
GANAS [44]	-	94.34	-	73.28	-	**46.80**	444	Generative Reinforcement Learning
AG-Net (ours)	**91.61***	**94.37***	**73.49***	**73.51***	**46.73**	46.42	400	Generative LSO

Table 2: Architecture Search on NAS-Bench-201. We report the mean over 10 trials for the search of the architecture with the highest validation accuracy.

Small CNNs on several datasets

NAS-Bench-201: 15,625 evaluated on CIFAR-10, CIFAR-100, and ImageNet-16

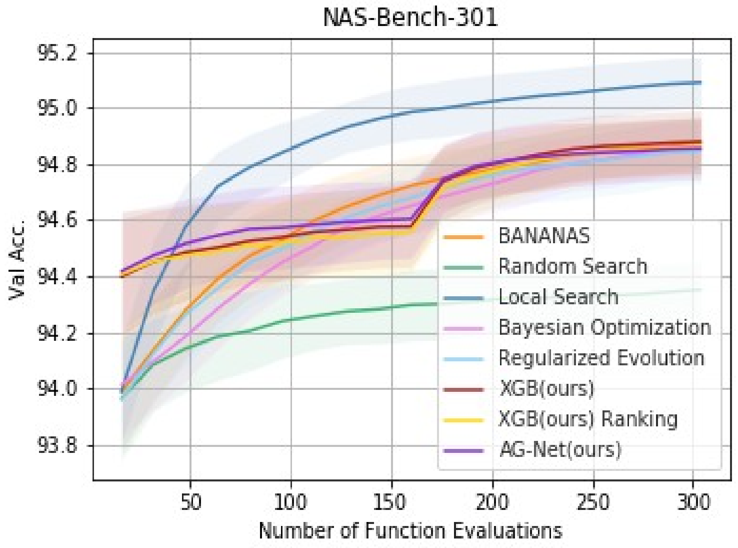


Larger CNNs – proxy benchmark

NAS-Bench-301: ~ 60 k sampled and evaluated on CIFAR-10 in the DARTS Search space

NAS Method	Val. Acc (%)	StD (%)	Queries
BANANAS† [56]	94.77	0.10	192
Bayesian Optimization† [50]	94.71	0.10	192
Local Search† [57]	**95.02**	0.10	192
Random Search† [31]	94.31	0.12	192
Regularized Evolution† [42]	94.75	0.11	192
XGB (ours)	94.79	0.13	192
XGR + Ranking (ours)	94.76	0.14	192
AG-Net (ours)	94.79	0.12	192

Table 10: Results on NAS-Bench-301 (mean and standard deviation over 50 trials) for the search of the best architecture in terms of validation accuracy compared to state-of-the-art methods.



Large CNNs

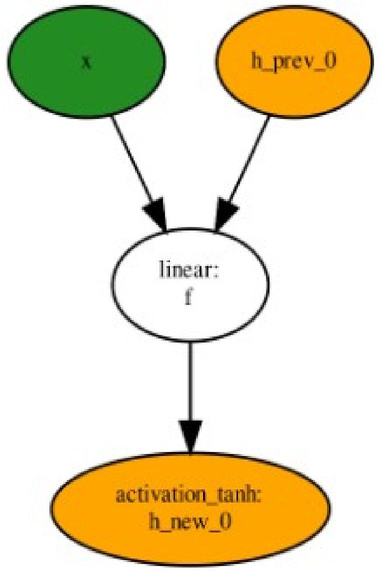
DARTS: ImageNet evaluations using NAS-Bench-301 and TENAS [2]

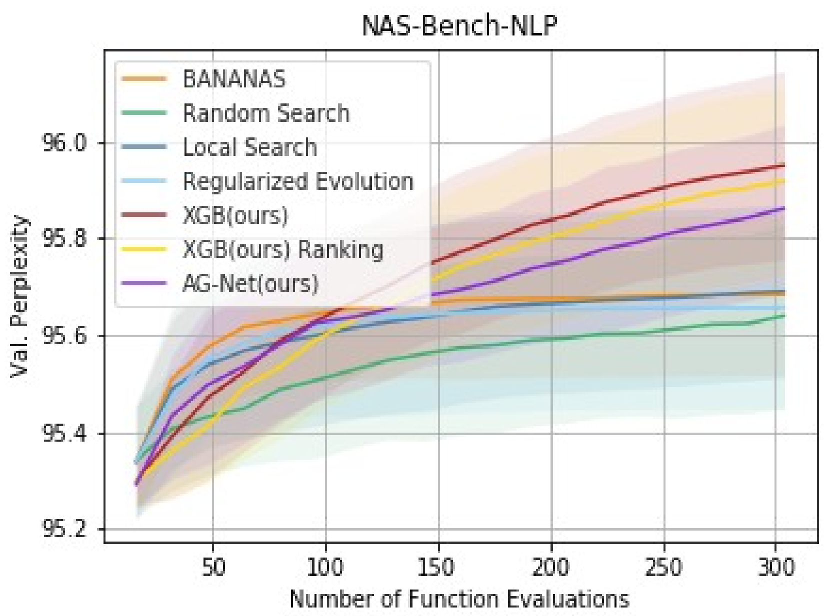
NAS Method	Top-1↓	Top-5↓	# Queries	Search GPU days
Mixed Methods				
NASNET-A (CIFAR-10) [70]	26.0	8.4	20000	2000
PNAS (CIFAR-10) [33]	25.8	8.1	1160	225
NAO (CIFAR-10) [36]	24.5	7.8	1000	200
Differentiable Methods				
DARTS (CIFAR-10) [34]	26.7	8.7	-	4.0
SNAS (CIFAR-10) [61]	27.3	9.2	-	1.5
PDARTS (CIFAR-10) [12]	**24.4**	**7.4**	-	0.3
PC-DARTS (CIFAR-10) [63]	25.1	7.8	-	**0.1**
PC-DARTS (ImageNet) [63]	**24.2**	**7.3**	-	3.8
Predictor Based Methods				
WeakNAS (ImageNet) [60]	**23.5**	**6.8**	800	2.5
XGB (NB-301)(CIFAR-10) (ours)	24.1	7.4	304	**0.02**
XGB + Ranking (NB-301)(CIFAR-10) (ours)	24.1	7.2	304	**0.02**
AG-Net (NB-301)(CIFAR-10) (ours)	24.3	7.3	304	0.21
Training-Free Methods				
TE-NAS (CIFAR-10) [11]	26.2	8.3	-	0.05
TE-NAS (ImageNet) [11]	24.5	7.5	-	0.17
AG-Net (CIFAR-10) (ours)	**23.5**	7.1	208	**0.02**
AG-Net (ImageNet) (ours)	**23.5**	6.9	208	**0.09**

Table 4: ImageNet **error** of neural architecture search on DARTS.

Large RNNs

NAS-Bench-NLP: 14,322 [3] sampled and evaluated on Penn TreeBank in the RNN-derived benchmark





NAS Method	Val. Perplexity (%)	StD (%)	Queries
BANANAS† [56]	95.68	0.16	304
Local Search† [57]	95.69	0.18	304
Random Search† [31]	95.64	0.19	304
Regularized Evolution† [42]	95.66	0.21	304
XGB (ours)	**95.95**	0.20	304
XGR + Ranking (ours)	95.92	0.19	304
AG-Net (ours)	95.86	0.18	304

Table 11: Results on NAS-Bench-NLP (mean and standard deviation over 100 trials) for the search of the best architecture in terms of validation perplexity compared to state-of-the-art methods.

Small CNNs on optimizing Hardware Properties

Hardware-Aware NAS-Bench: NAS-Bench-201 search space, but we have latencies for different devices

Account for latencies during LSO

Joint optimization (joint=1)

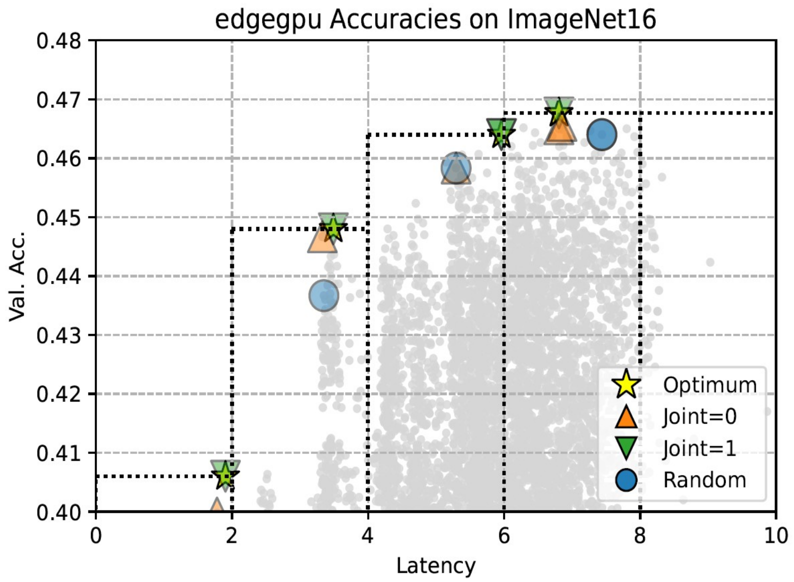
$$\max_{G \sim p_D} f(G) \wedge \min_{G \sim p_D,} g_h(G)$$
$$\text{s.t. } g_h(G) \leq L, \exists\, h \in H,$$

Constrained Optimization (joint=0)

$$\max_{G \sim p_D} f(G)$$
$$\text{s.t. } g_h(G) \leq L, \exists\, h \in H,$$

Small CNNs on optimizing Hardware Properties

Settings: Constraint Device	Lat.↓	Mean Joint=0 Acc.↑	Feas.↑	Joint=1 Acc.↑	Feas.↑	Random Acc.↑	Feas.↑	Optimum* Acc.↑	Lat.↓
Edge GPU	2	**0.397**	0.29	*0.391*	0.31	0.372	0.05	0.406	1.90
Edge GPU	4	*0.428*	0.29	**0.433**	0.43	0.417	0.22	0.448	3.49
Edge GPU	6	**0.453**	0.64	*0.450*	0.79	0.449	0.72	0.464	5.96
Edge GPU	8	**0.463**	0.98	*0.462*	0.99	0.457	1.00	0.468	6.81



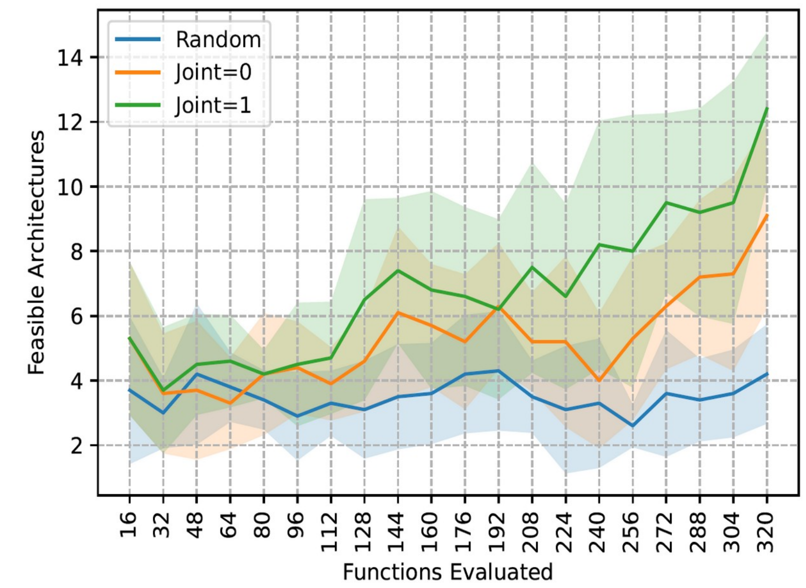


Fig. 4: (**left**) Exemplary searches on HW-NAS-Bench for image classification on ImageNet16 with 192 queries on Pixel 3 and latency conditions $L \in \{2, 4, 6, 8, 10\}$ (y-axis zoomed for visibility). (**right**) Amount of architectures generated and selected in each search iteration (at most 16) that satisfy the latency constraint. In this example we searched on Edge GPU with $L = 2$.

What about Robustness?

No dataset available?

Neural Architecture Design and Robustness

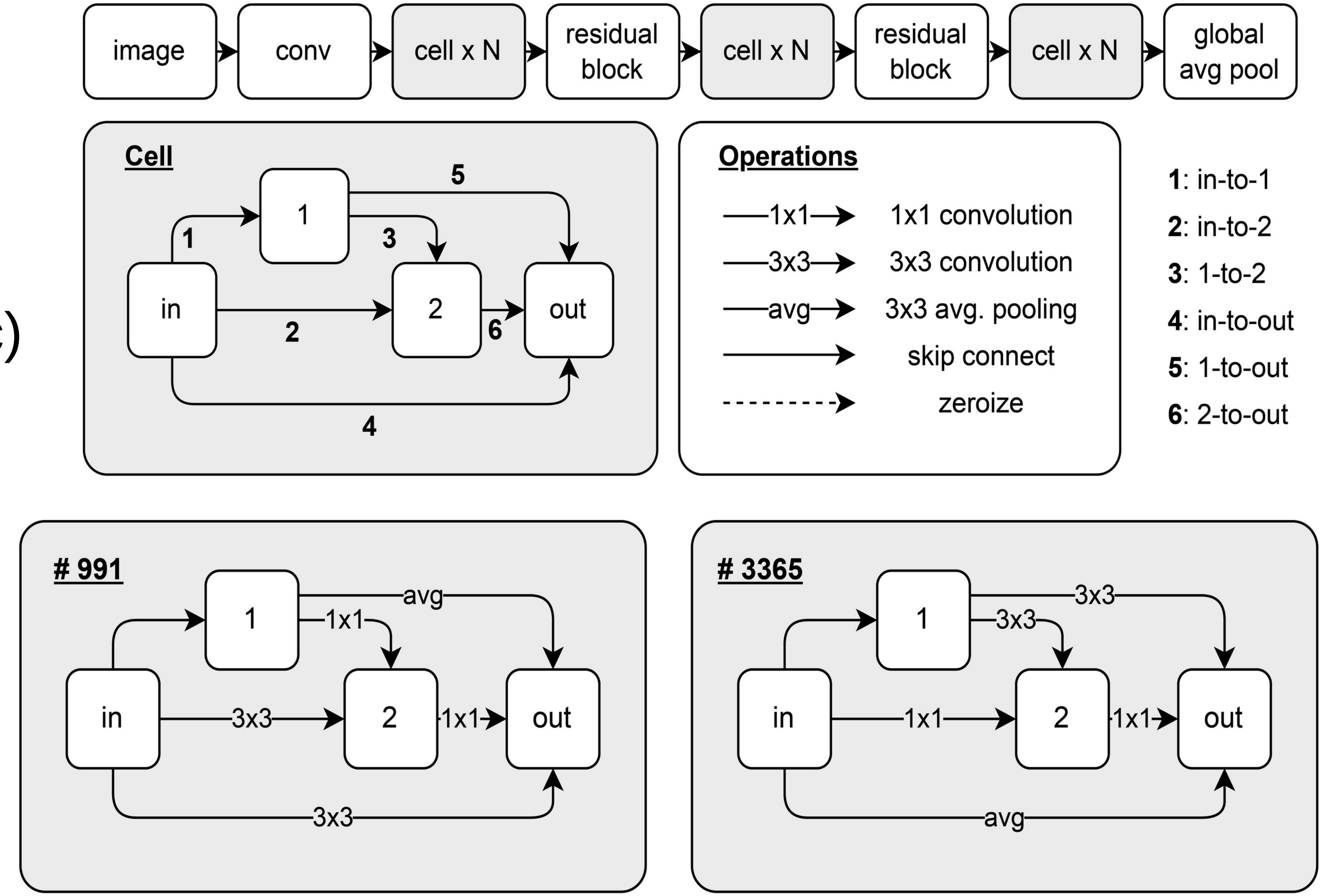
A Novel Dataset

- **Idea**: Collect robustness evaluations for a whole NAS search space
- **Motivation**: Evaluating a complete search space lets us investigate small architectural changes
- **Applications**:
  - NAS on robustness evaluation
  - Proxy measures: training-free robustness metrics
  - Architectural design analyses

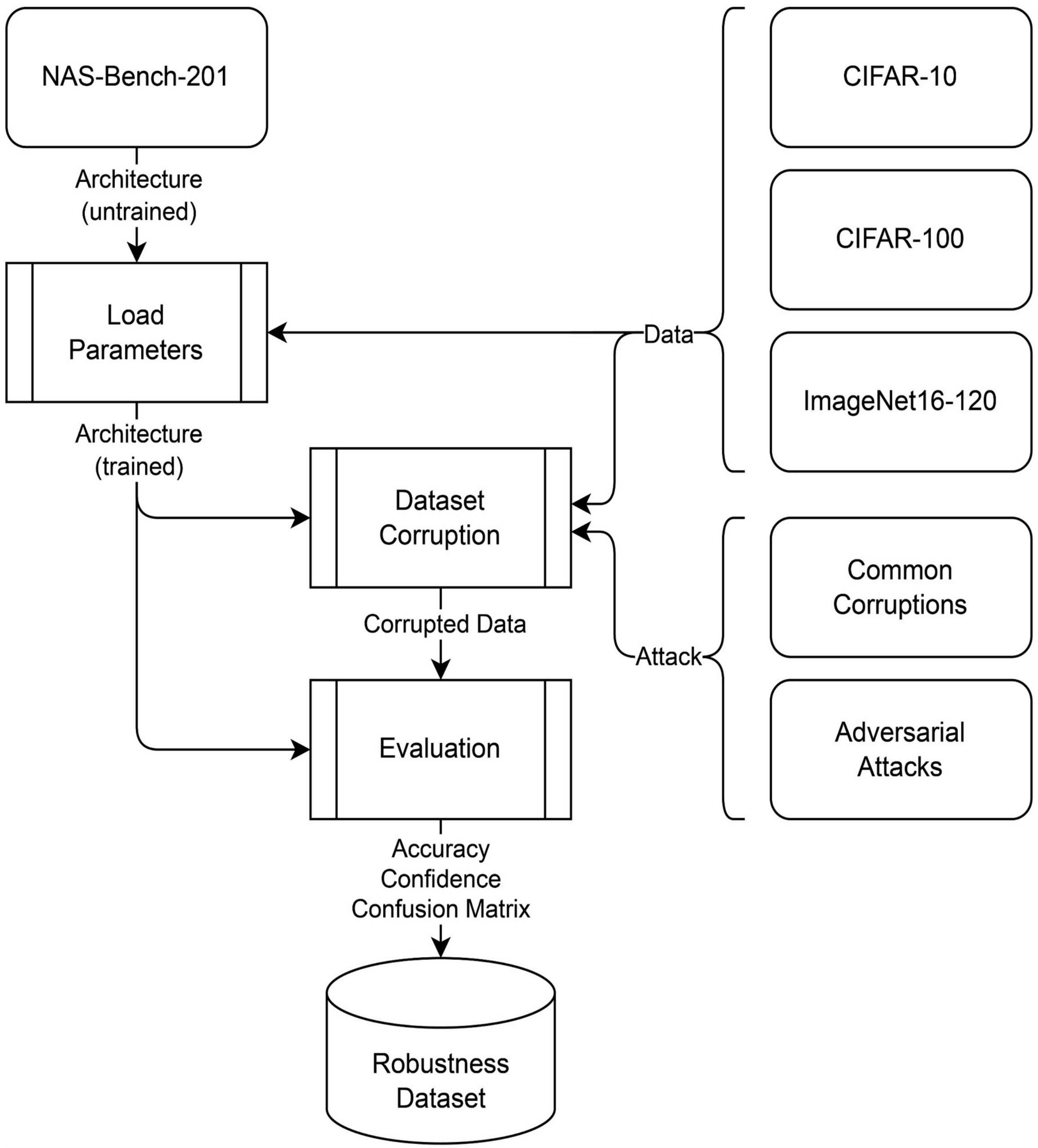
Jung, Lukasik, Keuper: Neural Architecture Design and Robustness – a Dataset, ICLR 2023.

NAS-Bench 201 Search Space

- Cell-based:
  - 4 nodes: features
  - 6 edges: operations
- 15 625 architectures (6 466 non-isomorphic)
- N = 5, C = 16
- Training data:
  - CIFAR-10
  - CIFAR-100
  - ImageNet16-120



Data Collection



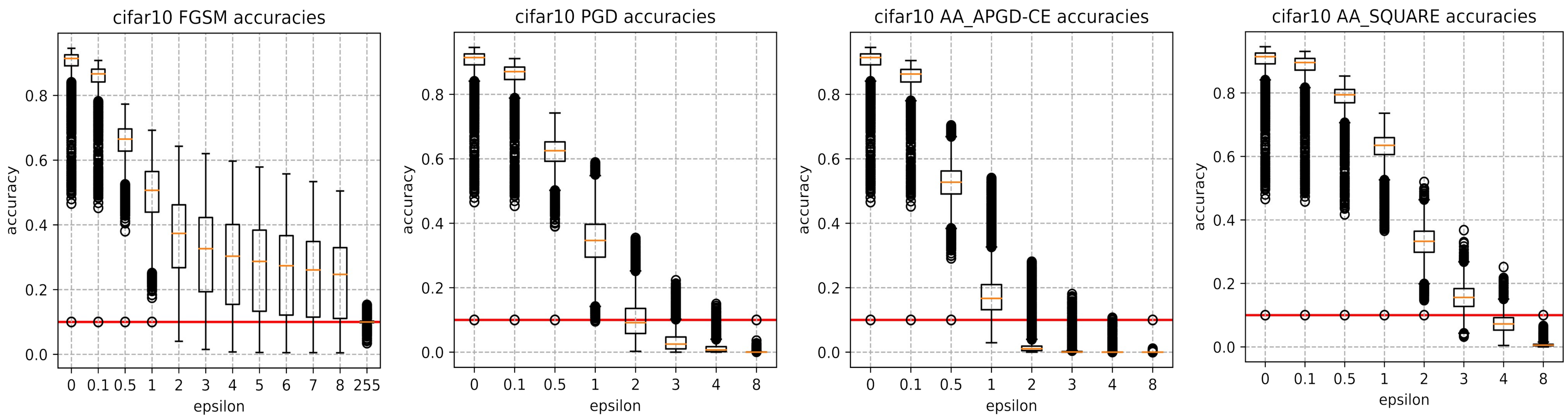
Adversarial Attacks

Table 2: Hyperparameter settings of adversarial attacks evaluated.

Attack	**Hyperparameters**
FGSM	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 5, 6, 7, 8, 255\}/255$
PGD	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ $\alpha = 0.01/0.3$ 40 attack iterations
APGD	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ 100 attack iterations
Square	$\epsilon \in \{.1, .5., 1, 2, 3, 4, 8, 255\}/255$ 5 000 search iterations

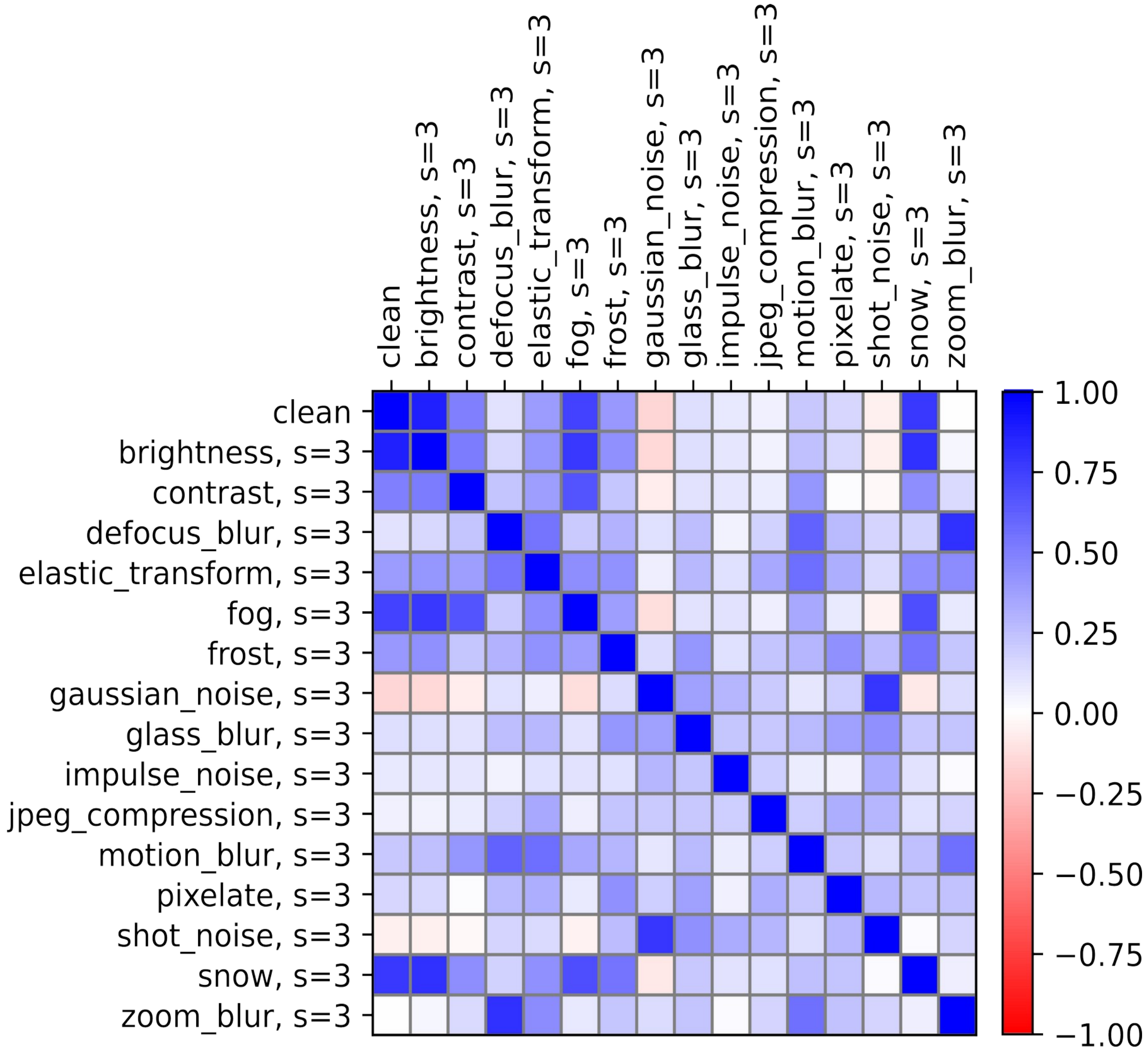
Adversarial Attacks

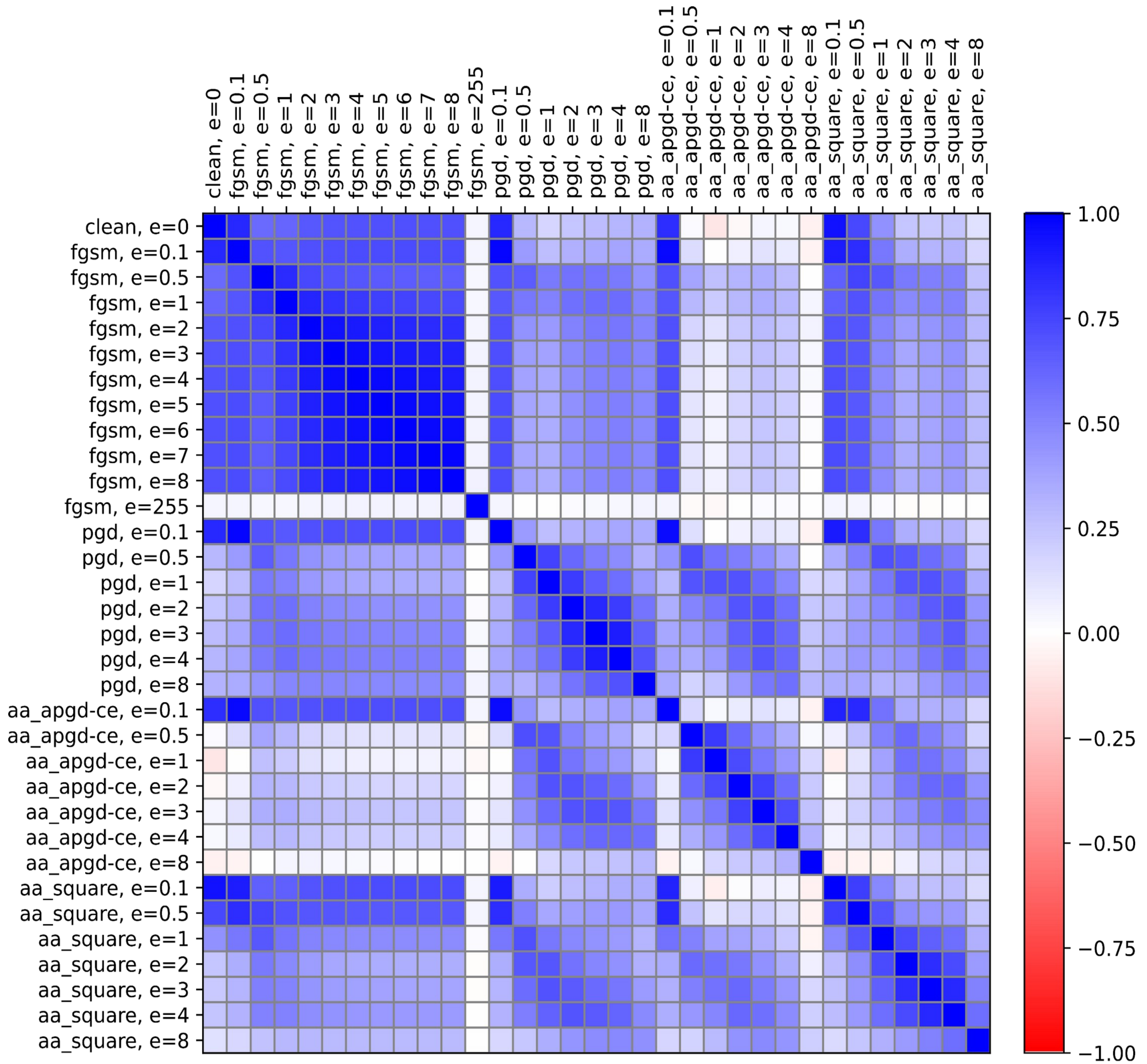
CIFAR-10 accuracy distributions



Correlation between Corruptions

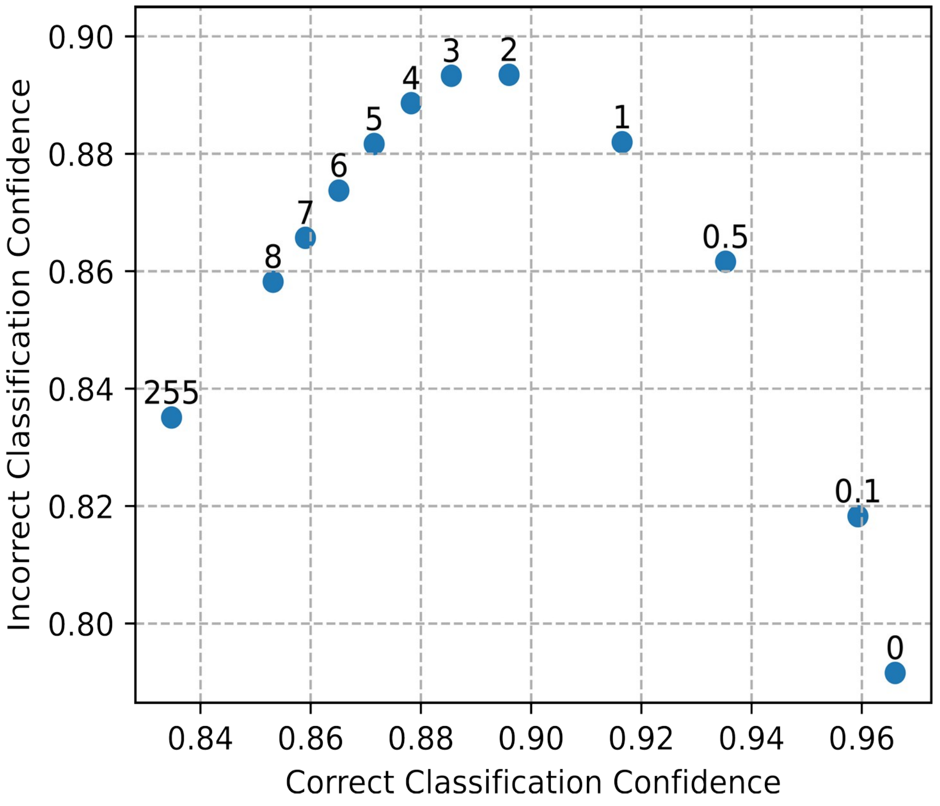
Kendall tau rank correlation

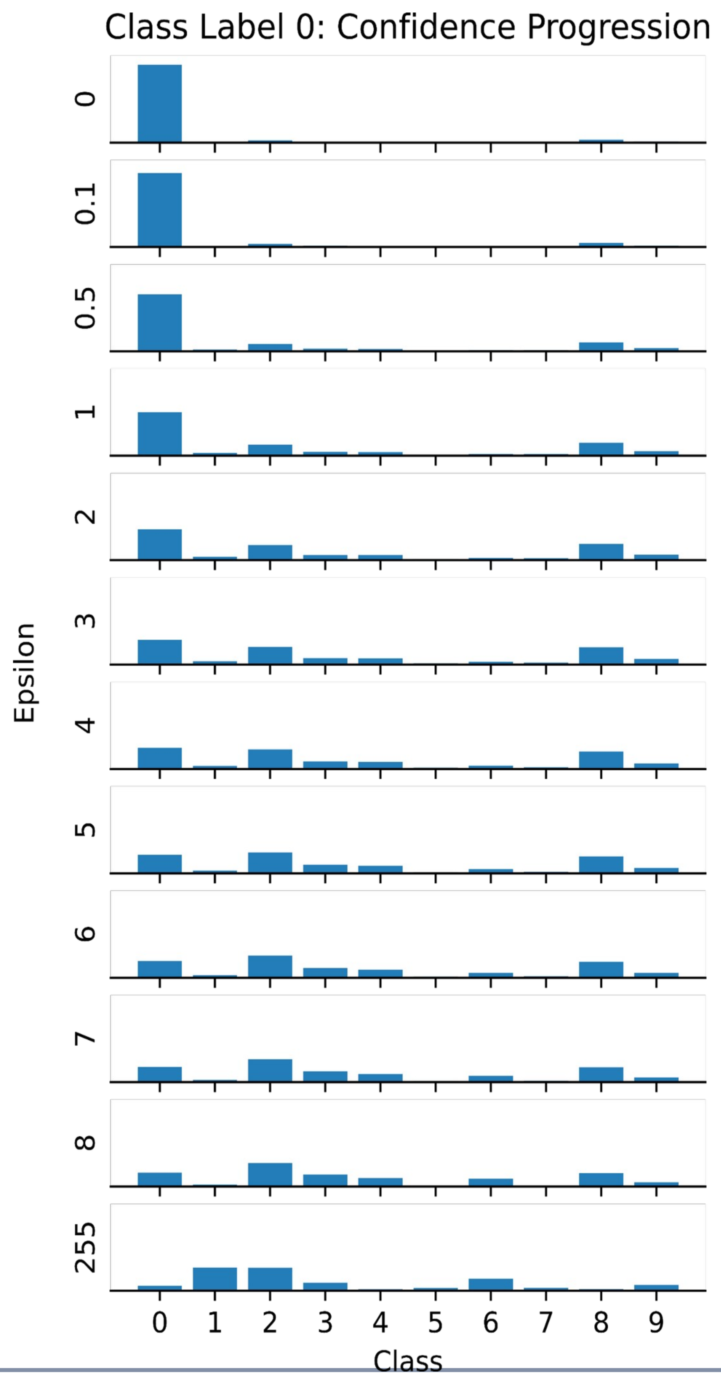




Confidences

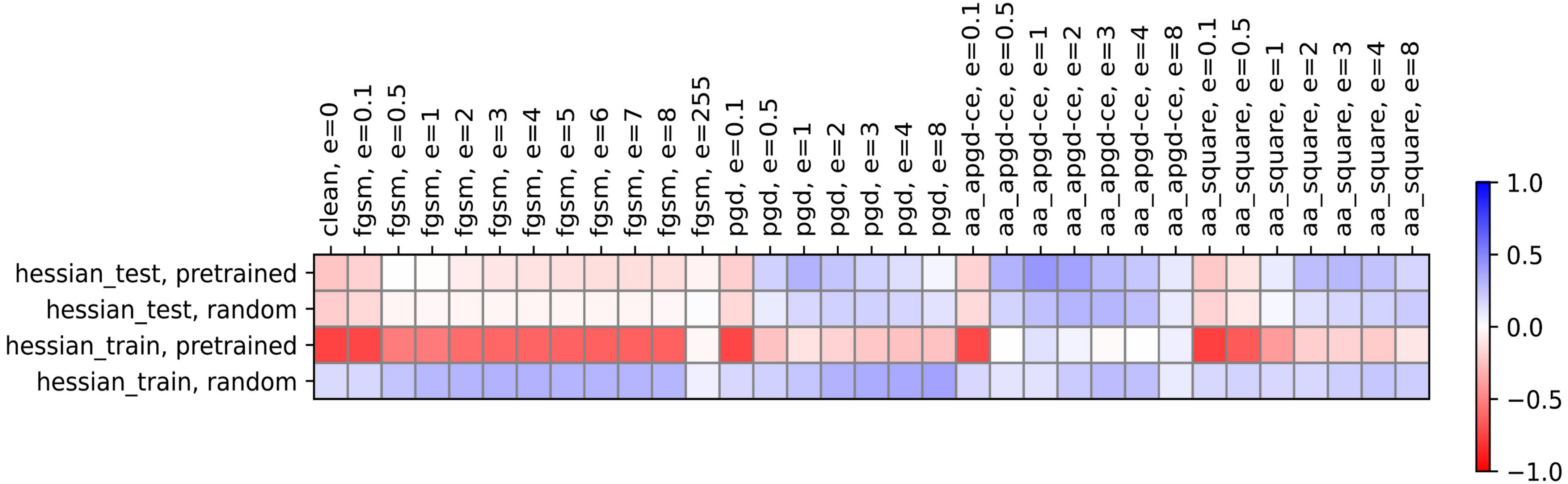
Networks become more confident in their wrong decisions… up to a certain point





Proxies for Robustness?

Kendall tau rank correlation



Analyzing Design Choices w.r.t. Robustness

Best architectures

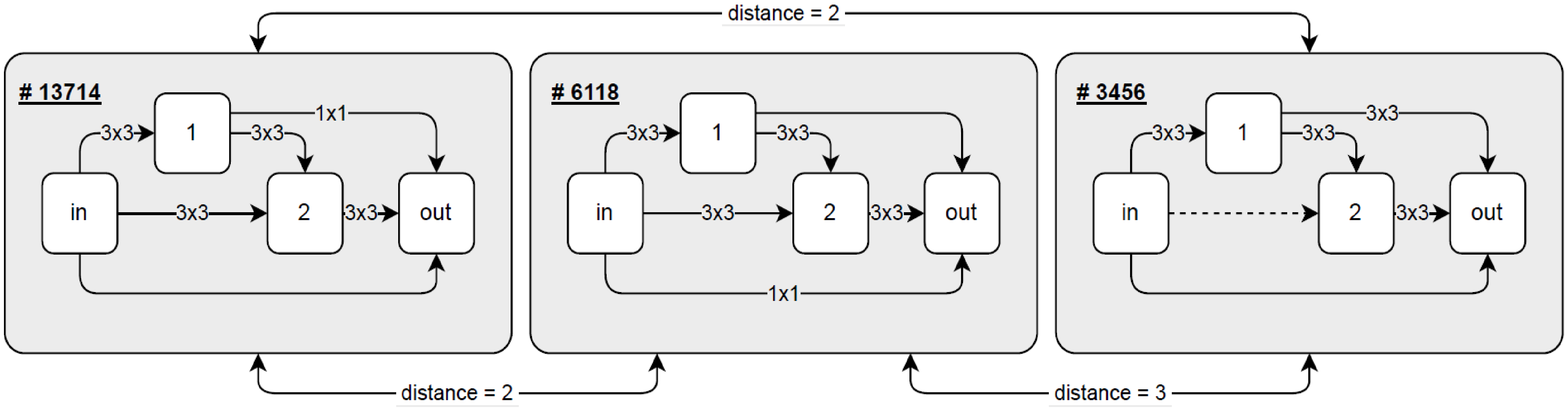
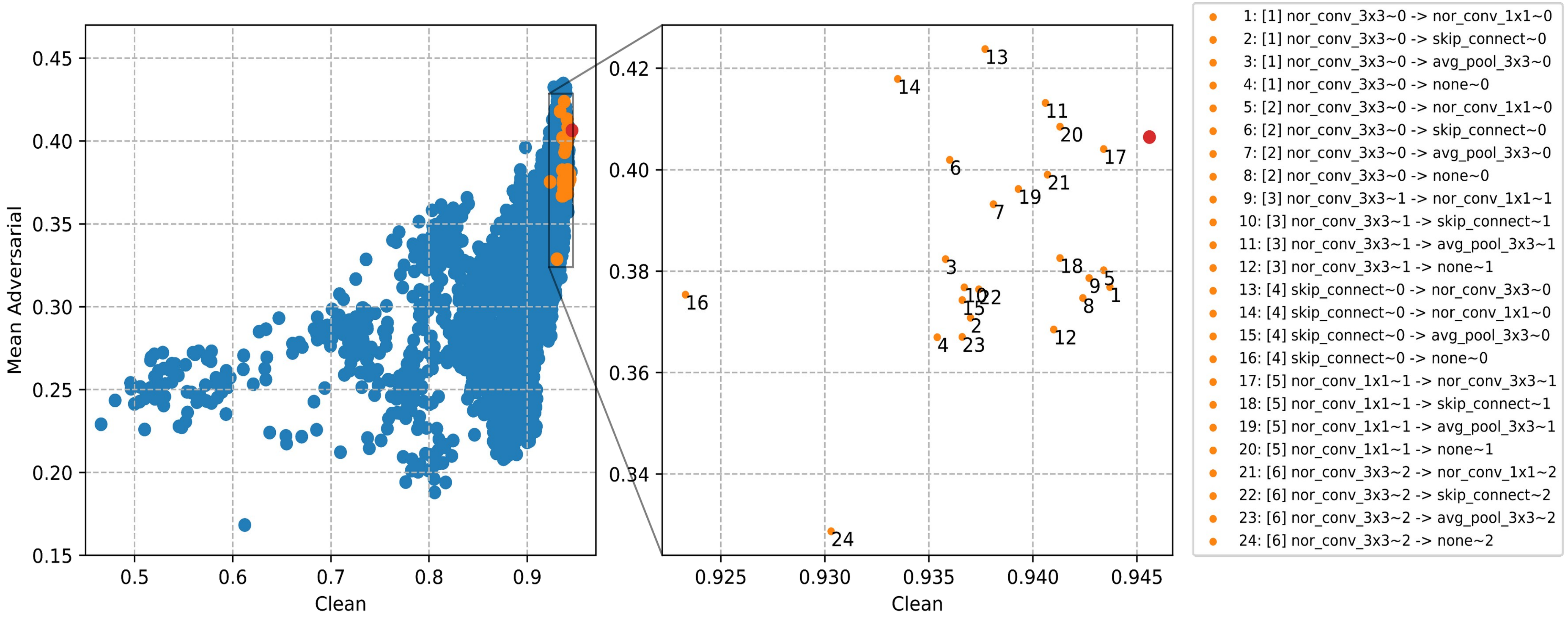


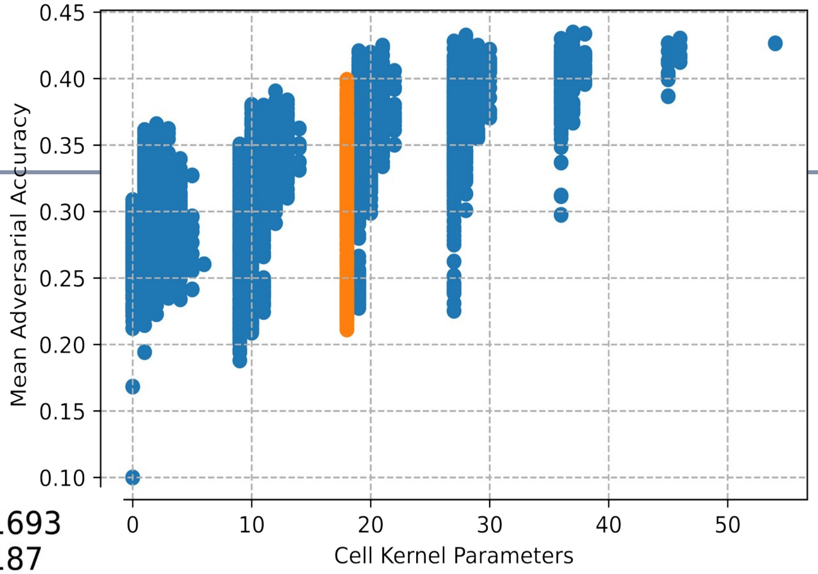
Figure 26: Best architectures in NAS-Bench-201 according to (**left**) clean accuracy, (**middle**) mean adversarial accuracy (over all attacks and ϵ values as described in subsection 3.2), and (**right**) mean common corruption accuracy (over all corruptions and severities) on CIFAR-10.

Analyzing Design Choices

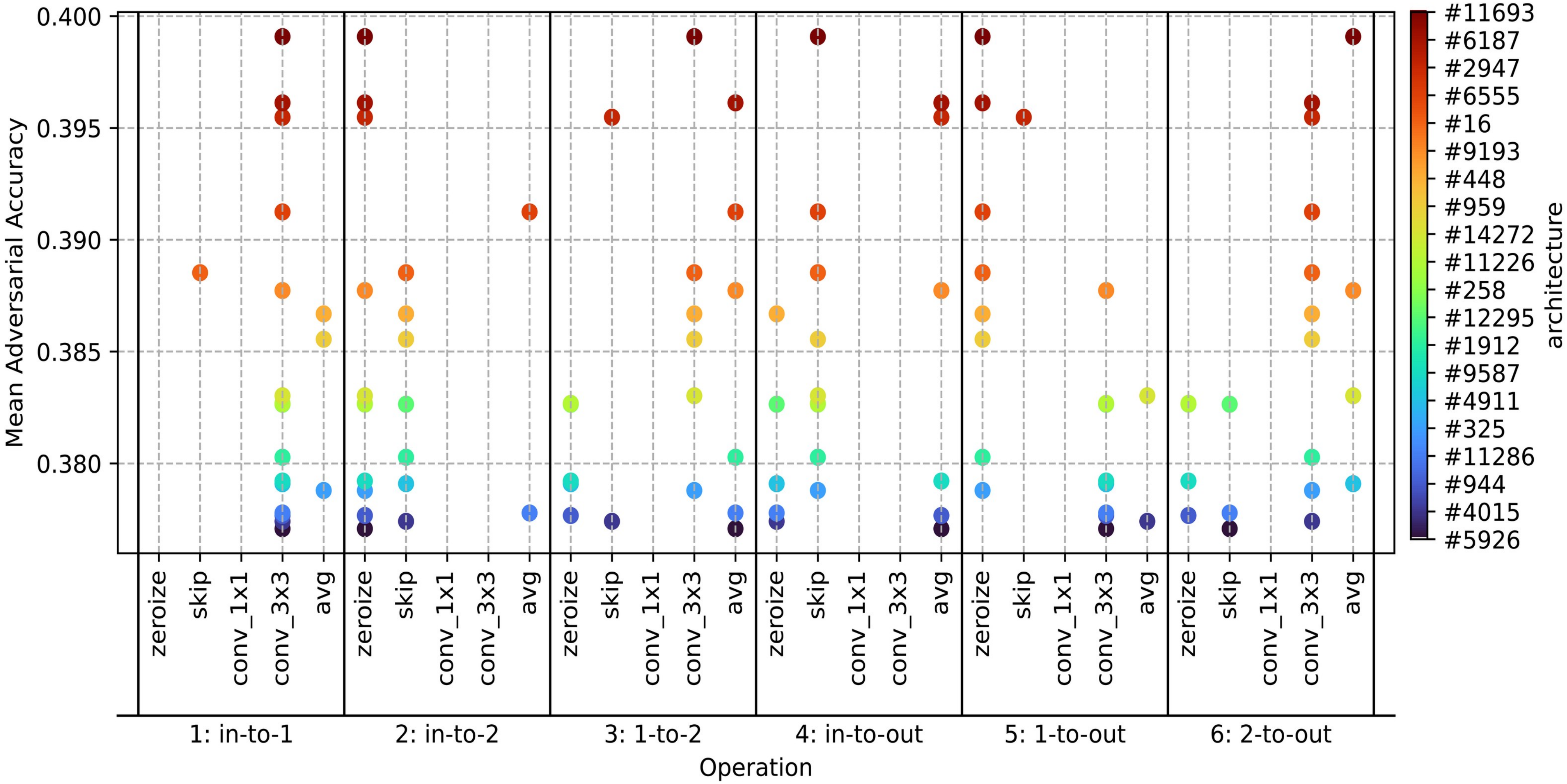
Neighbors of best network according to clean accuracy

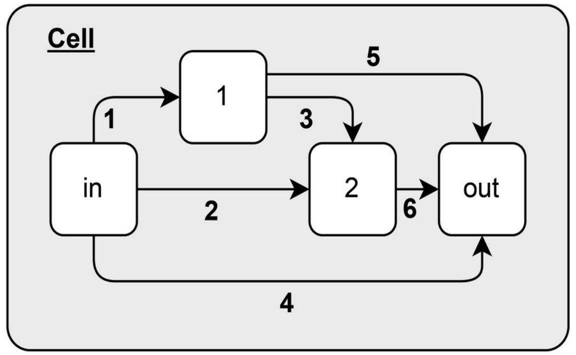


Analyzing Design Choices



Limited cell parameter count: Top 20 architectures



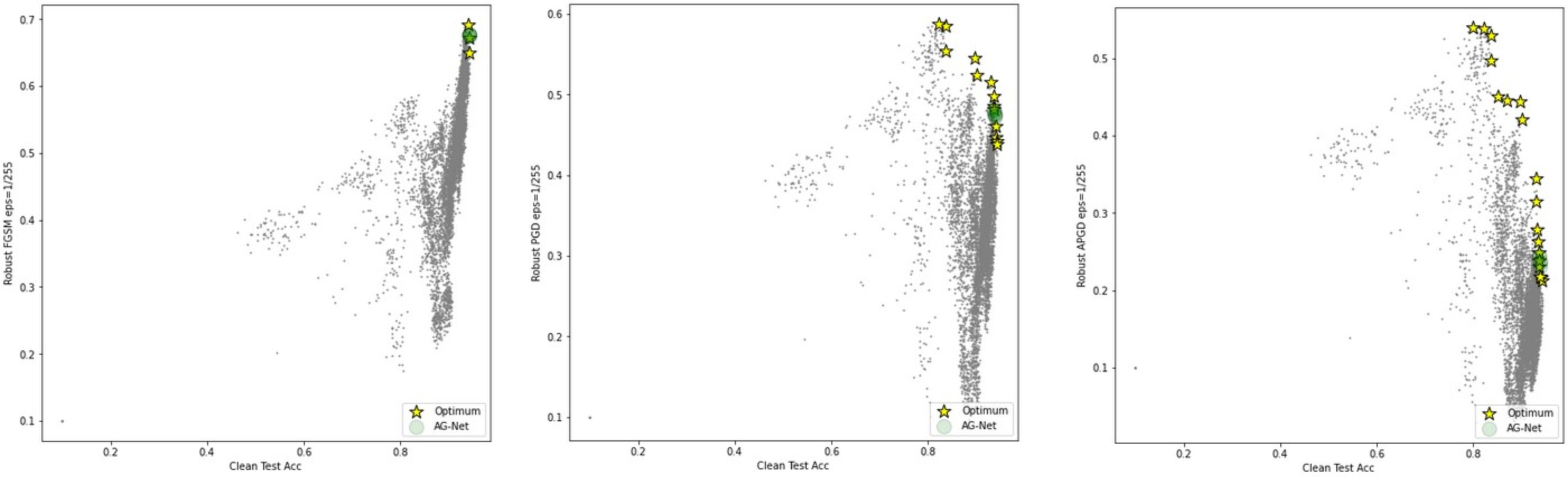


NAS on Robust Accuracy

Table 1: Neural Architecture Search on the clean test accuracy and the FGSM ($\epsilon = 1$) robust test accuracy for different state of the art methods on CIFAR-10 in the NAS-Bench-201 (Dong & Yang, 2020) search space (mean over 100 runs). Results are the mean accuracies of the best architectures found on different adversarial attacks and the mean accuracy over all corruptions and severity levels in CIFAR-10-C.

	Method	Test Accuracy ($\epsilon = 1.0$) Clean	FGSM	PDG	APGD	Squares	
		CIFAR-10					CF-10-C
	Optimum	94.68	69.24	58.85	54.02	73.61	58.55
Clean	BANANAS (White et al., 2021a)	94.21	64.25	41.10	18.62	68.69	55.52
Clean	Local Search (White et al., 2021b)	94.65	63.95	41.17	18.74	69.59	56.90
Clean	Random Search (Li & Talwalkar, 2019)	94.22	63.38	40.09	17.84	68.40	55.60
FGSM	BANANAS (White et al., 2021a)	93.52	66.35	45.59	20.72	68.01	54.88
FGSM	Local Search (White et al., 2021b)	93.86	69.10	48.27	23.18	69.47	56.57
FGSM	Random Search (Li & Talwalkar, 2019)	93.57	67.25	46.15	20.93	68.44	55.10

NAS on Robust Accuracy



applying
Lukasik, Jung, Keuper: Learning where to look – Generative NAS is surprisingly efficient, ECCV 2022.

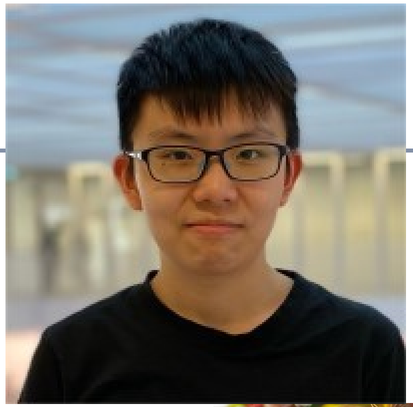
Conclusions

Even generally well-performing models can suffer from low robustness and generalization ability – when applying pre-trained models to specific tasks, they don't always behave the way we expect them to behave

Robustness can be improved through data augmentation or regularizations or combinations of both

Yet, particular neural architecture properties also relate to a model's robustness. Conducting NAS with respect to multiple objectives, including robustness, can provide insights on architecture design choices that have positive impact on a model's robustness.

Thank You!

Thank You!