CTU

CZECH TECHNICAL UNIVERSITY IN PRAGUE

How many channels should the 7th layer have?

Ondřej Týbl, Lukáš Neumann

50th Pattern Recognition and Computer Vision Colloquium
9.10.2025, CTU, Prague

How many channels should the 7th layer have?

- The question clearly needs additional information

- What task is the network going to be applied to?
  - image classification, depth estimation, generative AI, …
- What is the objective ?
  - maximise accuracy, minimise inference time, inference time vs accuracy trade-off, …

- You likely also need to consider preceding and successive layers

How many channels should the 7th layer have?

- Given a task and training data, what **network architecture** should we use to maximise given **objective**, subject to **design constraints**?
  - Network architecture = Network topology + hyper-parameters (number channels, non-linearity type, ...)
  - Objective = typically some measure of prediction accuracy
  - Design constraints = maximum number of parameters, inference time

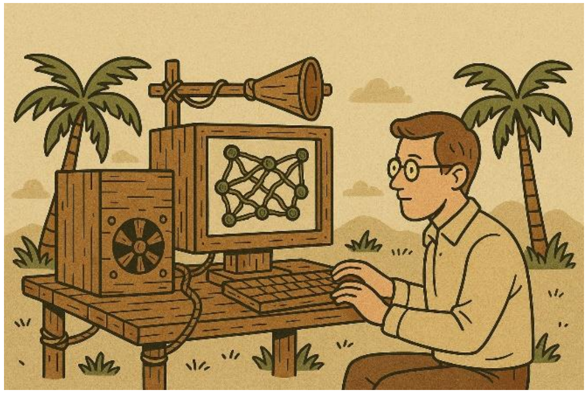
- Can we now answer how many channels should the 7th layer have?

How many channels should the 7th layer have?

- Given a and training data, what **network architecture** should we use to maximise given **objective**, subject to **design constraints**?

- Typical answers in the AI/ML community:
  1. Do the same thing the guys before us did
  2. Assume the specific choices do not matter too much and pick something "reasonable"

- This leads to "**cargo cult machine learning**"
  - The term "cargo cult science" coined by R. Feynmann
  - In short, "mindlessly copying elements of existing solutions without truly understanding the underlying principles"



Why do we resort to copying?

- The underlying machine learning theory is insufficient
- Therefore, we have to evaluate each design choice empirically
  - **To measure the objective (e.g. classification accuracy), we need to completely train the network, which is is extremely costly (time & money)**

- *If were in the business of designing car engines, that'd be like figuring out the number of cylinders, the compression ratio or the cooling system layout by building an engine in each configuration and then measuring its performance*

- We need an (approximate) method to measure the objective without going through the whole training process

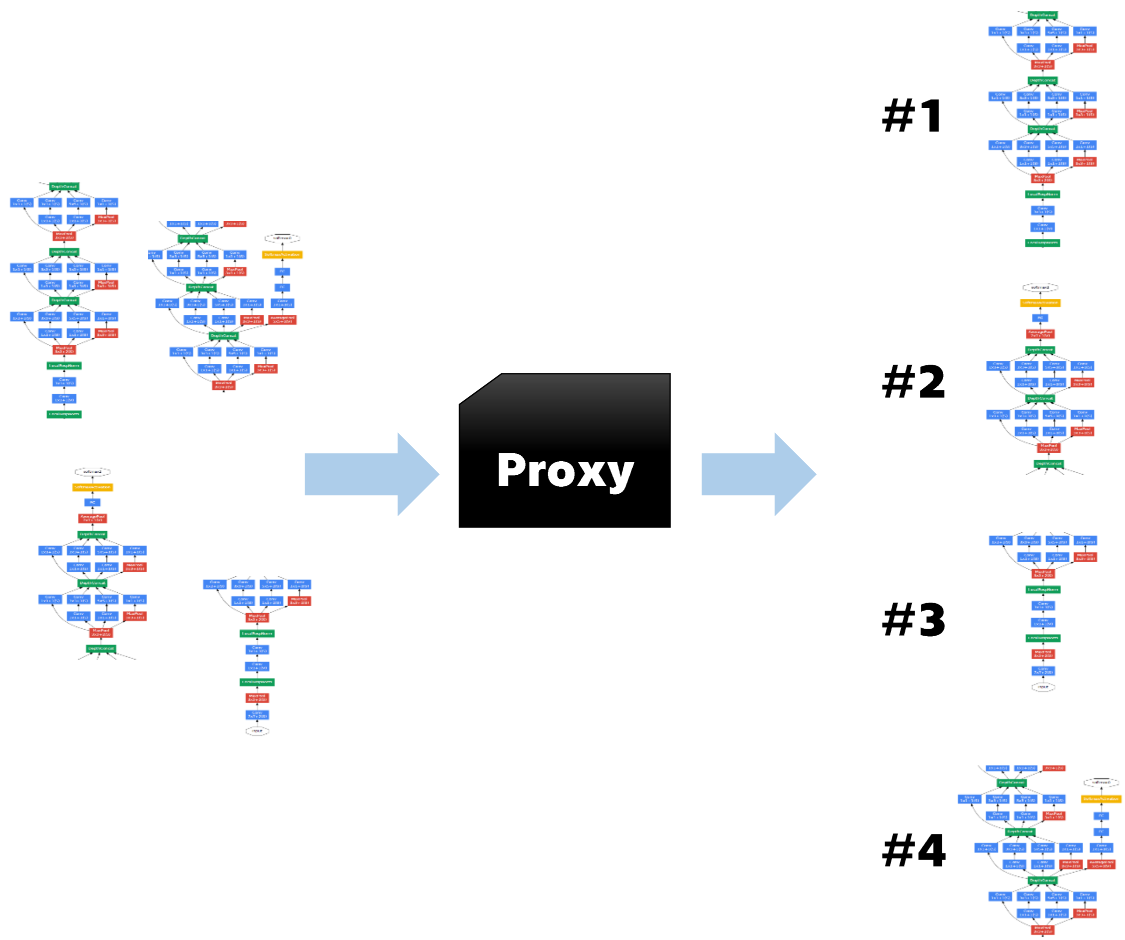
Training-free proxies

- Methods to estimate the objective, such as classification accuracy, without fully training the network
- Ideally, we'd want the method to output estimated accuracy on given dataset, but that's extremely hard (impossible?)

- Instead, we only ask the proxy to produce a score

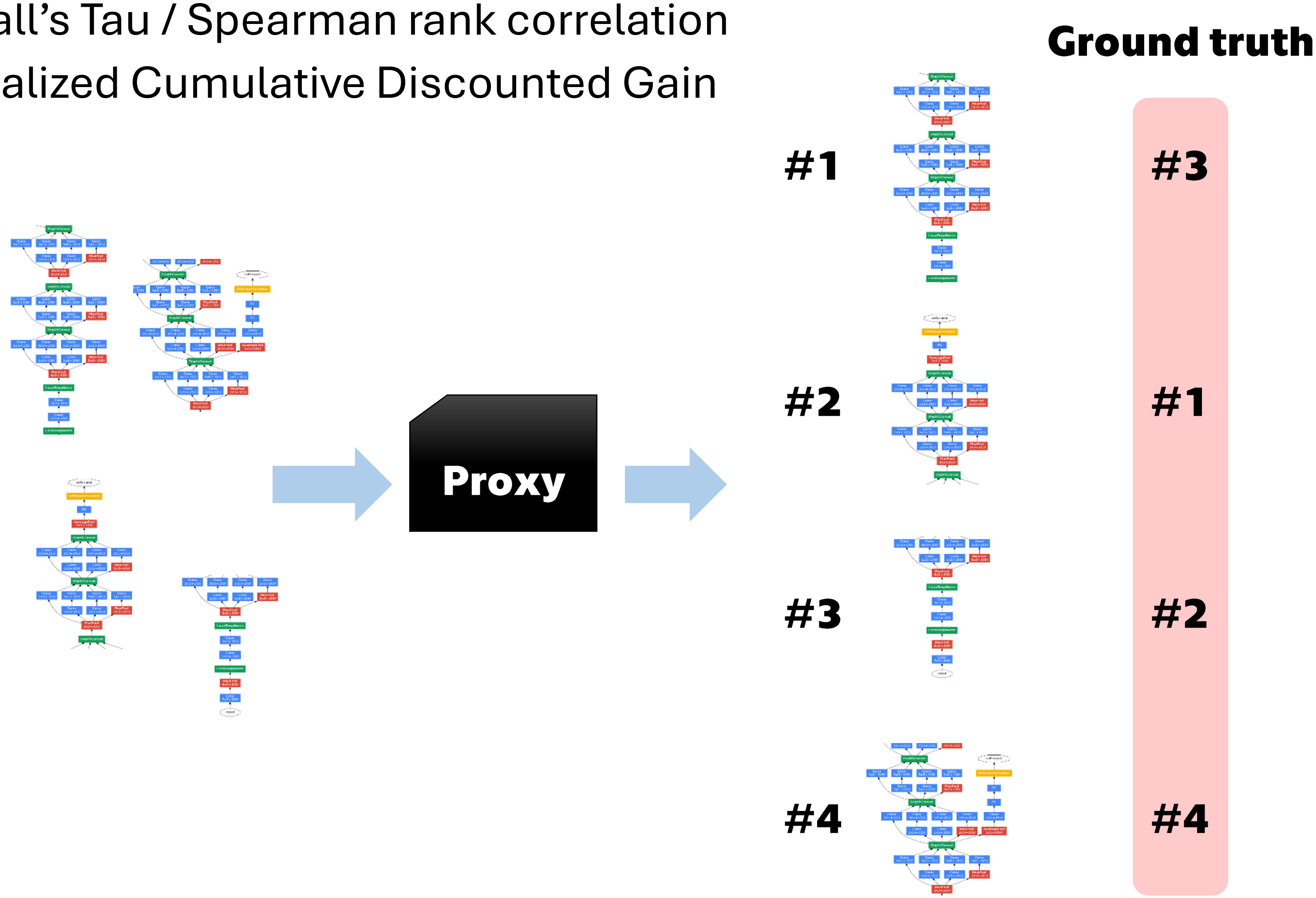
Training-free proxies

- Given an (infinite) set of networks, rank the networks from the best to the worst



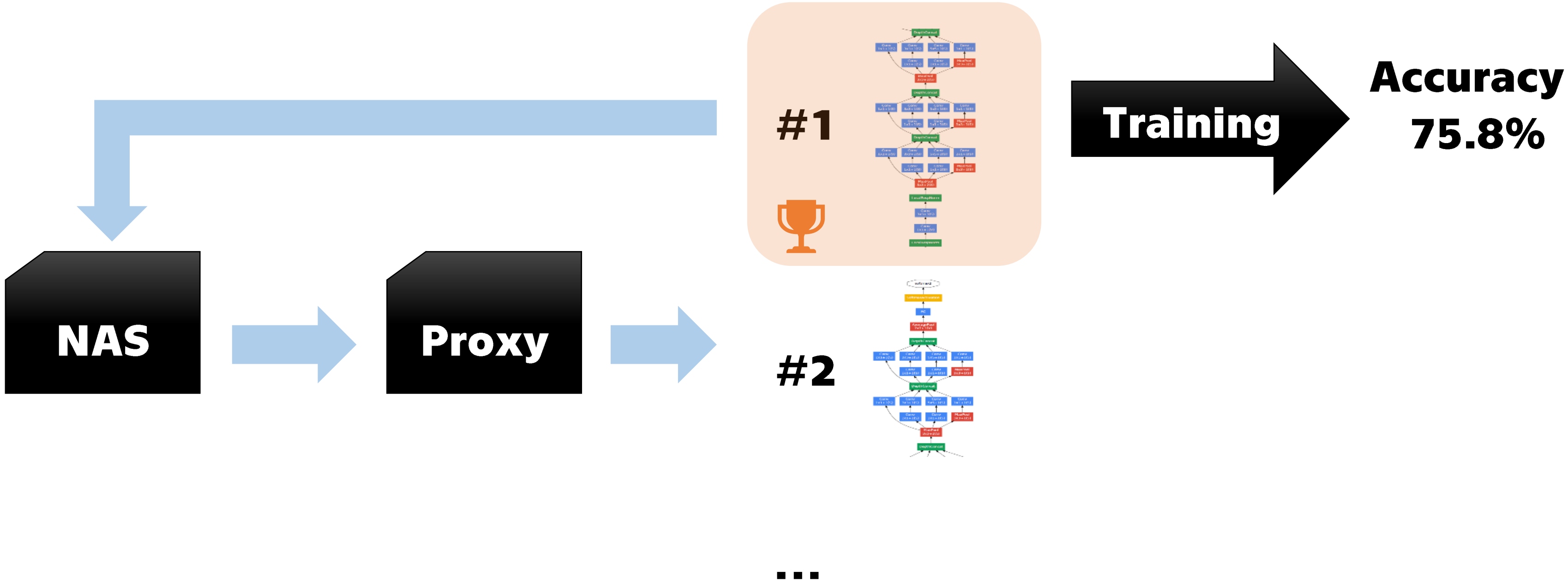
Evaluating training-free proxies

1. How well the proxy ranks a predefined set of architectures, where the accuracies are known but not available to the proxy (*NAS-Bench-101, …*)
   - Compare the ranking produced by the proxy to the ground truth
   - Kendall's Tau / Spearman rank correlation
   - Normalized Cumulative Discounted Gain



Evaluating training-free proxies

2. Use the proxy to find the best architecture in an infinite set of architectures
   - Run a Neural Architecture Search (NAS) algorithm (e.g. evolutionary algorithm) for a given number of iterations
   - When the search completes, train the top candidate to obtain true accuracy



NASWOT

- Scores networks at initialization by counting activations of ReLUs in a single mini-batch of training data
- The more similar activations for different inputs, the harder it is for the network to distinguish them

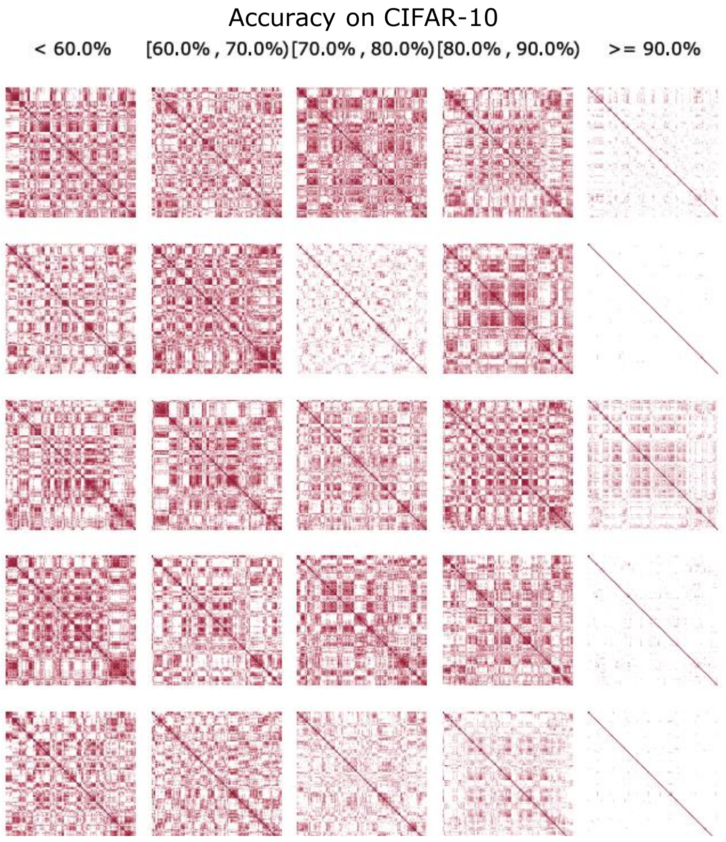
- The activations create a binary code $\mathbf{c}_\mathrm{i}$ for each training sample $\mathbf{x}_\mathrm{i}$
- The final score is the determinant of the matrix $\mathbf{K}_H$, where N_A is the number of ReLUs and d_H is the Hamming distance

$$\mathbf{K}_H = \begin{pmatrix} N_A - d_H(\mathbf{c}_1, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_1, \mathbf{c}_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(\mathbf{c}_N, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_N, \mathbf{c}_N) \end{pmatrix}$$

Mellor J. et al., "Neural architecture search without training.", ICML 2021

NASWOT

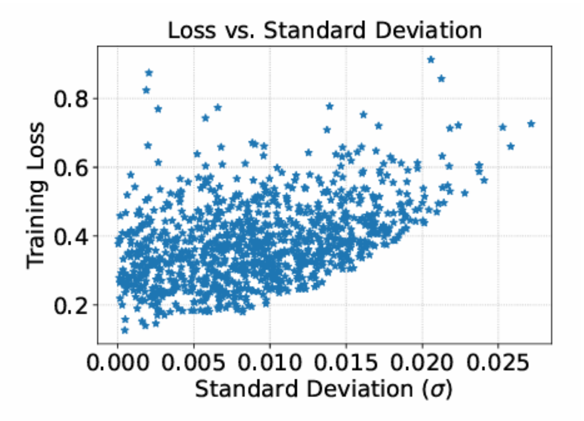
- Scores networks at initialization by counting activations of ReLUs in a <u>single</u> mini-batch of training data
- The more similar activations for different inputs, the harder it is for the network to distinguish them

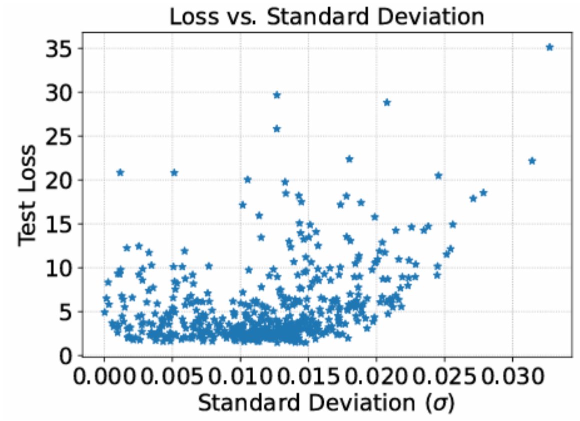


Mellor J. et al., "Neural architecture search without training.", ICML 2021

ZiCo

- For faster convergence, a network should have high loss gradient absolute mean and low loss gradient variance across different training samples
- Single forward & backward pass, average gradient mean and variance for each layer (signal to noise ratio)



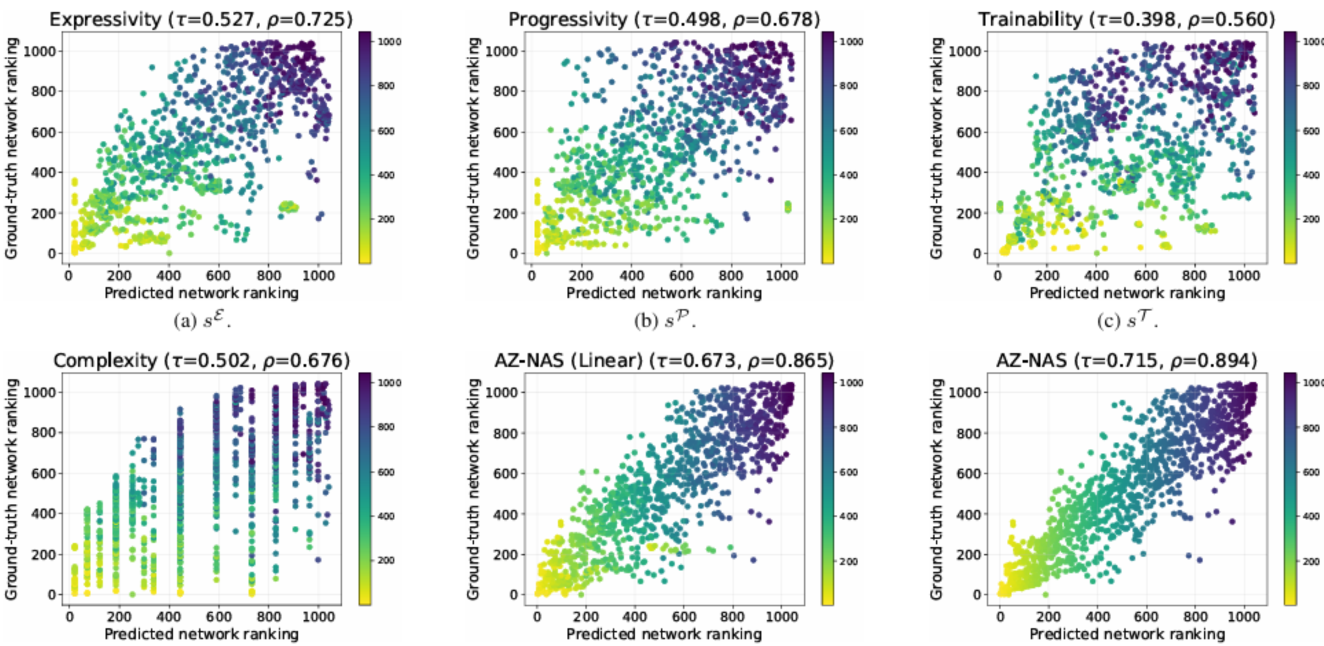


- The first training-free proxy to consistently outperform **#params** proxy

Li G. et al., "ZiCo: Zero-shot NAS via inverse Coefficient of Variation on Gradients.", ICLR 2023

AZ-NAS

- Single forward & backward pass
- Non-linear aggregation (multiplication) of 4 different rankings: Expressivity, Progressivity, Trainability and Complexity
- The network needs to score high in all 4 rankings to be ranked on top



(a) $s^{\mathcal{E}}$. (b) $s^{\mathcal{P}}$. (c) $s^{\mathcal{T}}$.

(d) $s^{\mathcal{C}}$. (e) s^{AZ} (Linear ranking aggregation). (f) s^{AZ} (Non-linear ranking aggregation).

Lee J. and Bumsub H., "AZ-NAS: Assembling zero-cost proxies for network architecture search.", CVPR 2024

VKDNW

- Variance of Knowledge of Deep Networks Weights (VKDNW)

- *"Vaclav Klaus Did Nothing Wrong"*

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

VKDNW

- *We posit that network architectures should be characterised by how easy it is to estimate their optimal network weights*

- We use **Fisher Information** framework to formally describe the expected behaviour of finding optimal weights (aka training the network)

GPT-5

Fisher information measures **how much information** a random variable carries about an unknown parameter.

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Fisher Information

- **Fisher Information Matrix (FIM)** of a network with parameters $\boldsymbol{\theta}$ is defined as

$$[F(\boldsymbol{\theta})]_{ij} = \mathbb{E}\left[\left(\frac{\partial}{\partial\theta_i}\sigma_c(X;\theta)\right)\left(\frac{\partial}{\partial\theta_j}\sigma_c(X;\theta)\right)\right], \quad \boldsymbol{\theta} = \begin{bmatrix}\theta_1\\ \theta_2\\ \vdots\\ \theta_p\end{bmatrix}$$

  where in the case of classification the $\sigma_c(X;\ \boldsymbol{\theta})$ is a posteriori probability of the true class c for the input X (= output of the softmax layer)

- The FIM captures how much information the data carries about each parameter θ_i as well as statistical coupling between parameters θ_i and θ_j

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Fisher Information

- The eigenvalues λ_i of the **Fisher Information Matrix** then describe the overall curvature of the (log-)likelihood space

- When all eigenvalues are of <u>similar magnitude</u>, the log-likelihood has similar curvature in all directions
  - All parameters are equally sensitive to data changes
  - There is no "weak" or "strong" direction in the parameter space → small perturbations in any direction change the likelihood by the same amount
  - <u>The parameter space is well-conditioned for optimisation</u>, there are no directions where gradients are tiny (flat) or huge (steep)

- In contrast, when eigenvalues vary, some directions are poorly informed → some parameters are ill-conditioned or redundant

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Fisher Information

- The eigenvalues λ_i of the **Fisher Information Matrix** then describe the overall curvature of the (log-)likelihood space

- When all eigenvalues are of similar magnitude, the log-likelihood has similar curvature in all directions

- We define **Variance of Knowledge for Deep Network Weights** as the entropy of Fisher Information Matrix eigenvalues

$$\mathrm{VKDNW}(f) := -\sum_{i=1}^{N} \tilde{\lambda}_i \log \tilde{\lambda}_i, \quad \tilde{\lambda}_i = \frac{\lambda_i}{\sum_{j=1}^{N} \lambda_j}$$

- The entropy attains its maximum when all eigenvalues are the same
- **For any reasonably-sized deep network, this is intractable** ☹

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

VKDNW approximation

1. Take only a subset of parameters (down to 1 parameter per trainable layer)

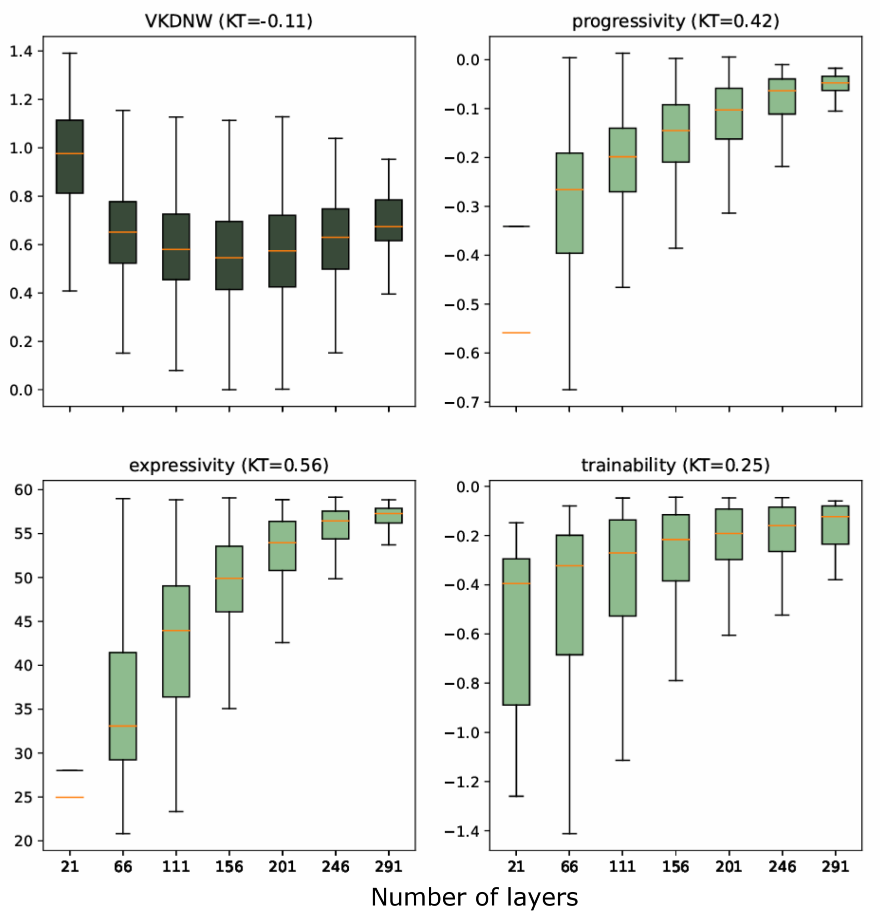
2. A novel algorithm for estimation of Fisher Information Matrix (FIM) eigenvalues that prevents the usual numerical instability

3. We use only 9 deciles of the FIM eigenvalues (smallest λ_0 is usually zero)
   $\rightarrow$ We get the same number of eigenvalues, irrespective of the number of network parameters

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Results

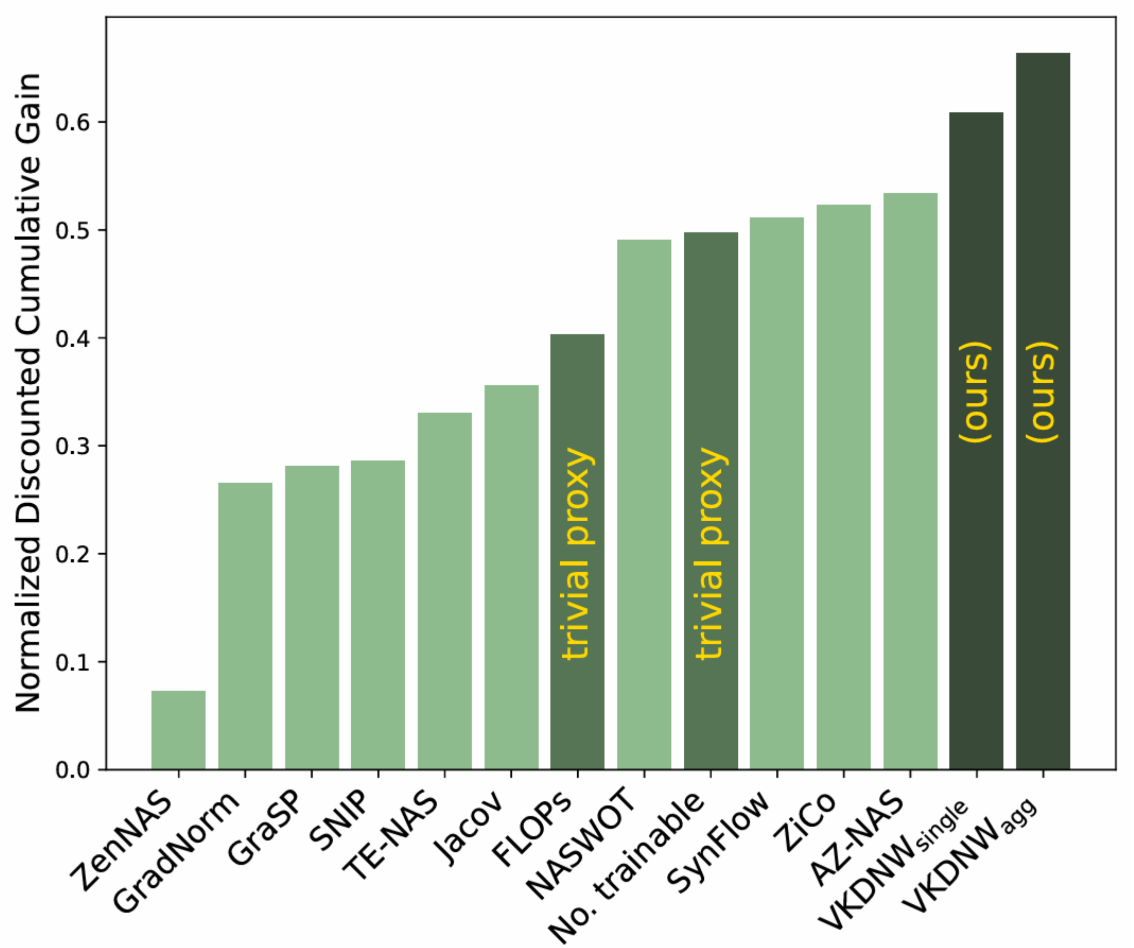
- VKDNW has the lowest correlation with the model size



Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Results

- Ranking architectures (NAS-Bench-201)



Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

Results

- Training-free NAS in MobileNetV2 search space
- We used Evolutionary Algorithm of AZ-NAS with VKDNW as the fitness function size of the model is limited to ~450M FLOPs
- The best model is trained on ImageNet-1k for 300 epochs

Method	FLOPs	Top-1 acc.	Type	Search cost (GPU days)
NASNet-B [50]	488M	72.8	MS	1800
CARS-D [45]	496M	73.3	MS	0.4
BN-NAS [5]	470M	75.7	MS	0.8
OFA [4]	406M	77.7	OS	50
RLNAS [48]	473M	75.6	OS	-
DONNA [32]	501M	78.0	OS	405
# Params	451M	63.5	ZS	0.02
ZiCo [25]	448M	78.1	ZS	0.4
AZ-NAS [21]	462M	78.6	ZS	0.4
$\text{VKDNW}_{\text{agg}}$ (ours)	480M	**78.8**	ZS	0.4

Tybl O. and Neumann L., "Training-free Neural Architecture Search through Variance of Knowledge of Deep Network Weights.", CVPR 2025

UniNAS space

- NAS methods so far have failed to produce an architecture that would meaningfully outperform hand-crafted architectures such as EfficientNet or ViT

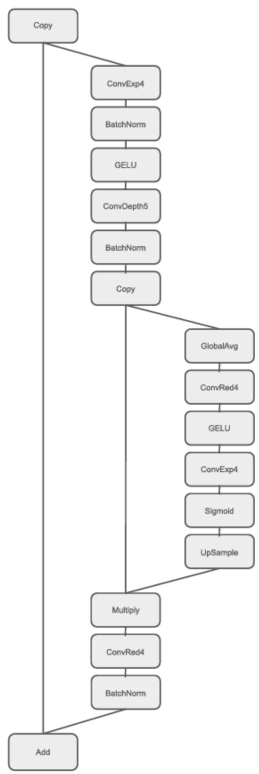
- NAS research is focused on tabular benchmarks (NAS-Bench-X) which in principle cannot produce novel architectures

- Or it looks for best hyper-parameters of existing blocks (MobileNetV2) → the search space is very restricted

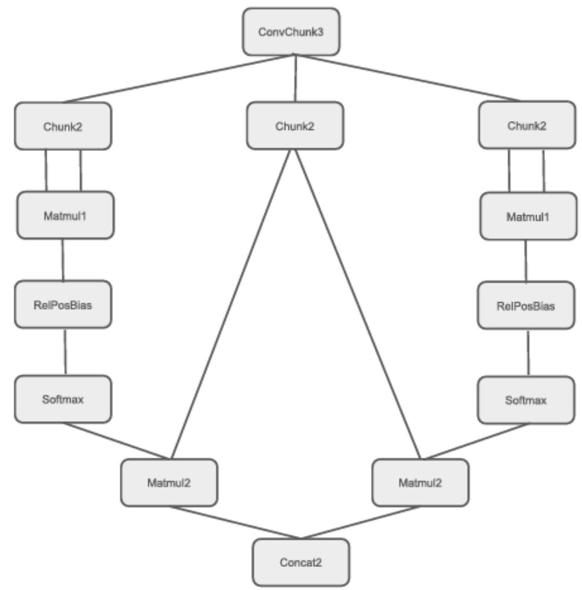
Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

UniNAS space

- We propose a new search space which contains all modern architectures



MBConv block (EfficientNet)

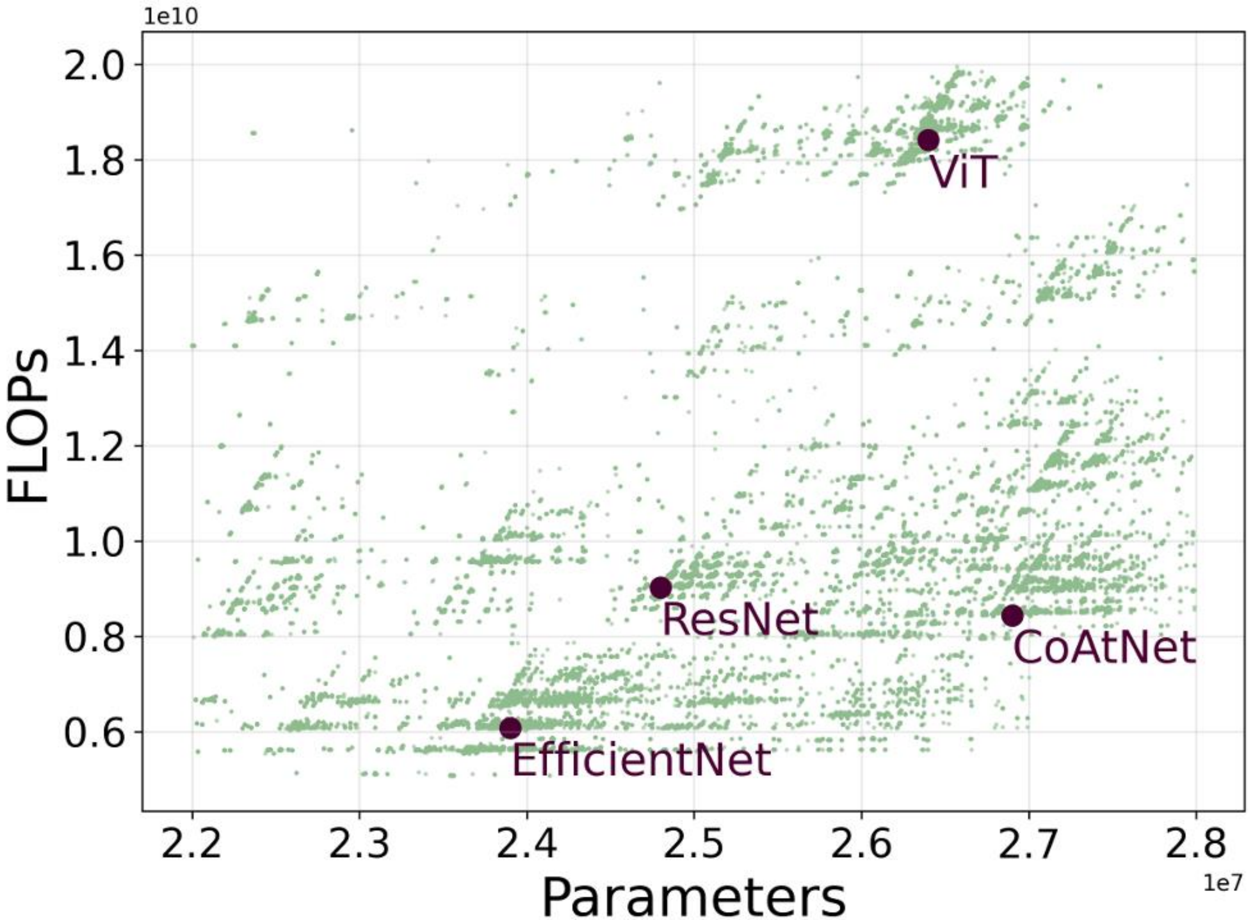


Self-attention with 2 heads (ViT)

Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

UniNAS space

- We propose a new search space which contains all modern architectures



Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

UniNAS space

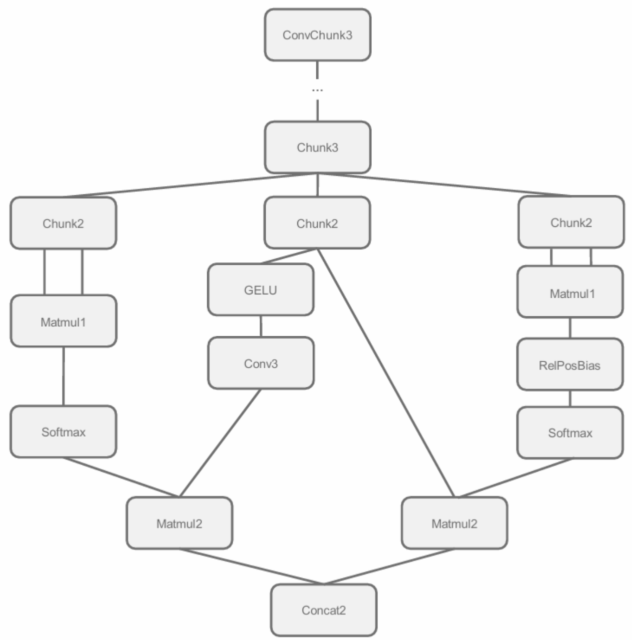
- We propose a new search space which contains all modern architectures
- We propose an algorithm to traverse the search space
- We include well-defined protocol for the final evaluation

	Image Classification	Object Detection	Image Segmentation
training data	ImageNet-1k [8]	COCO [24]	ADE20K [50]
network head	FC	Mask R-CNN [14]	UperNet [44]
GPU count	N	N	N
epochs	150	12	125
warmup epochs	5	5	5
batch size	48	4	4
optimizer	AdamW [29]	AdamW [29]	AdamW [29]
weight decay	0.05	0.05	0.05
LR schedule	cosine	multi-step	linear
warmup LR	$N \times 10^{-7}$	$N \times 10^{-7}$	$N \times 10^{-7}$
minimal LR	$N \times 10^{-6}$	$N \times 2.5 \times 10^{-6}$	0
learning rate	$N \times 10^{-4}$	$N \times 2.5 \times 10^{-5}$	$N \times 1.5 \times 10^{-5}$
data aug.	rand-m15-n2-mstd0.5	RandFlip0.5	PhotoMetricDist. RandFlip0.5
gradient clip	1.0	1.0	1.0
drop path	0.2	0.1	0.3
input resolution	224×224 px	1280×800 px	512×512 px

Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

Results

- Training-free NAS in UniNAS search space
- Model size limited to 30M params
- Run search for 1000 iterations (12 GPU-hours), using **VKDNW** as the proxy



UniNAS-A architecture

Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

Results

- Image classification (ImageNet-1k)

Model	Params	FLOPs	Accuracy (%)
ResNet [13]	24.8M	9.0G	75.19
EfficientNet [34]	24.0M	6.0G	80.52
ViT [7, 10]	26.4M	18.4G	80.64
CoAtNet [7]	26.9M	8.44G	80.58
UniNAS-A (ours)	26.8M	9.0G	**81.15**

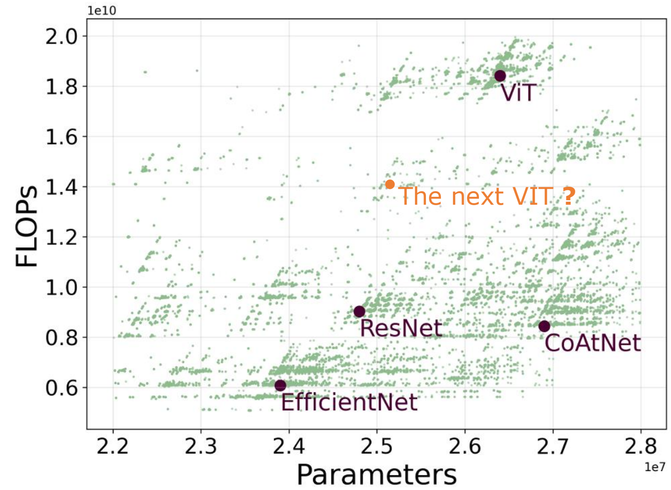
- Object Detection and Semantic Segmentation (MS-COCO and ADE20k)

		Object Detection and Segmentation				Semantic Segmentation		
Model	Params	AP^b	AP^m	FPS (images/s)	FLOPs	mIoU	FPS (images/s)	FLOPs
ResNet [13]	24.8M	37.7	34.7	102.8	184G	39.3	481.2	47G
EfficientNet [34]	24.0M	39.0	35.8	43.1	124G	37.0	152.6	32G
CoAtNet [7]	26.9M	41.3	38.4	14.4	296G	42.4	61.7	51G
UniNAS-A (ours)	26.8M	**42.4**	**39.0**	14.2	297G	**45.6**	88.2	51G

Tybl O. and Neumann L., "Universal Neural Architecture Space: Covering ConvNets, Transformers and Everything in Between", arXiv: 2510.06035

Conclusion

- Training-free proxies have democratised Neural Architecture Search
- You can use training-free proxies such as **VKDNW** to find optimal hyper-parameter values of your networks for a very small computation cost
- Still lot of room for improvement!



GitHub

Thank you for your attention!

VKDNW