Greedy Policy Search: A simple baseline for learnable test-time augmentation

Dmitry Molchanov Alexander Lyzhov Yuliya Molchanova

Arsenii Ashukha Dmitry Vetrov

SAMSUNG AI Center - Moscow

Higher School of Economics

p(B|A)yesgroup.ru

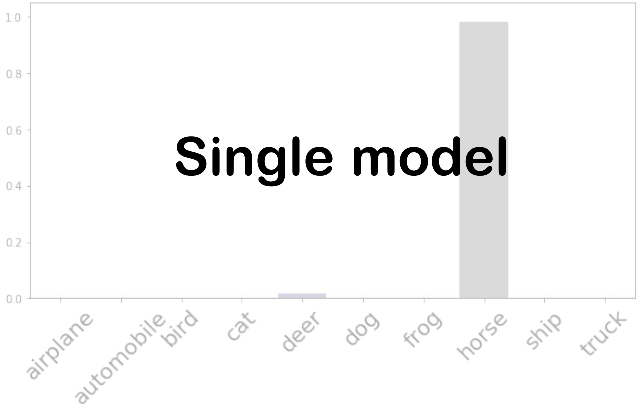
Skoltech Skolkovo Institute of Science and Technology

Real-world risk-sensitive scenarios require reliable machine learning models

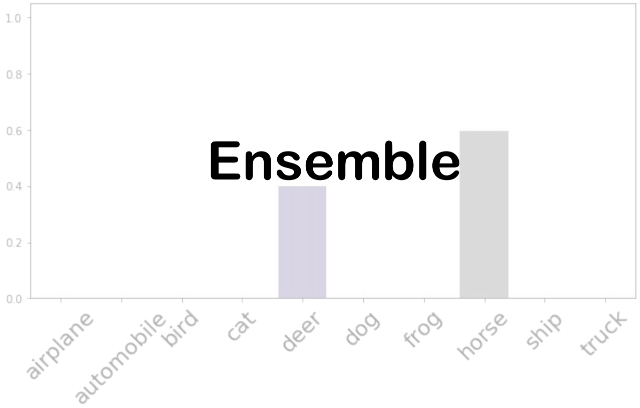
- robustness under dataset shift
- reporting a level of confidence in a prediction
- ...

Deer

Overconfident



More uncertain



Original Image

What the network sees during training

Data augmentation

Previously: hand-crafted data augmentation

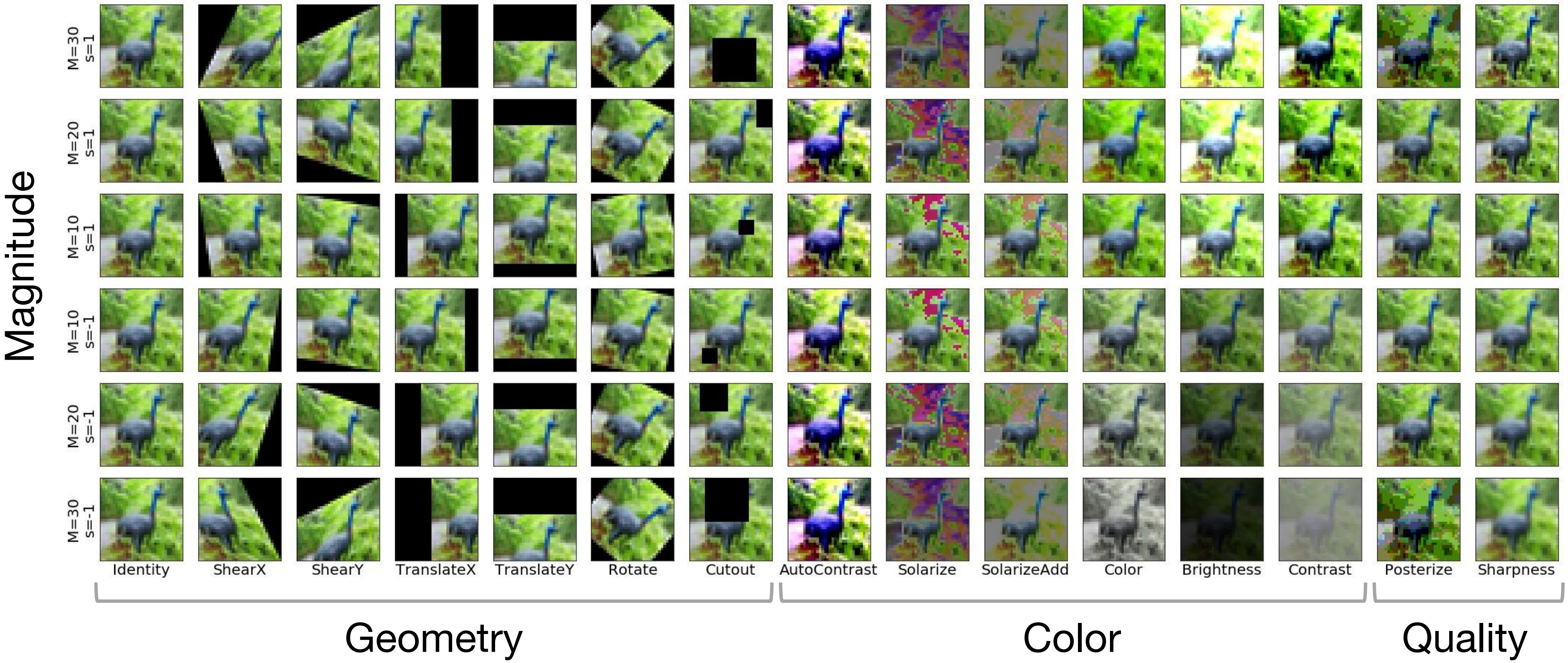
- Random resize + crop + flip
- Color jitter
- ...

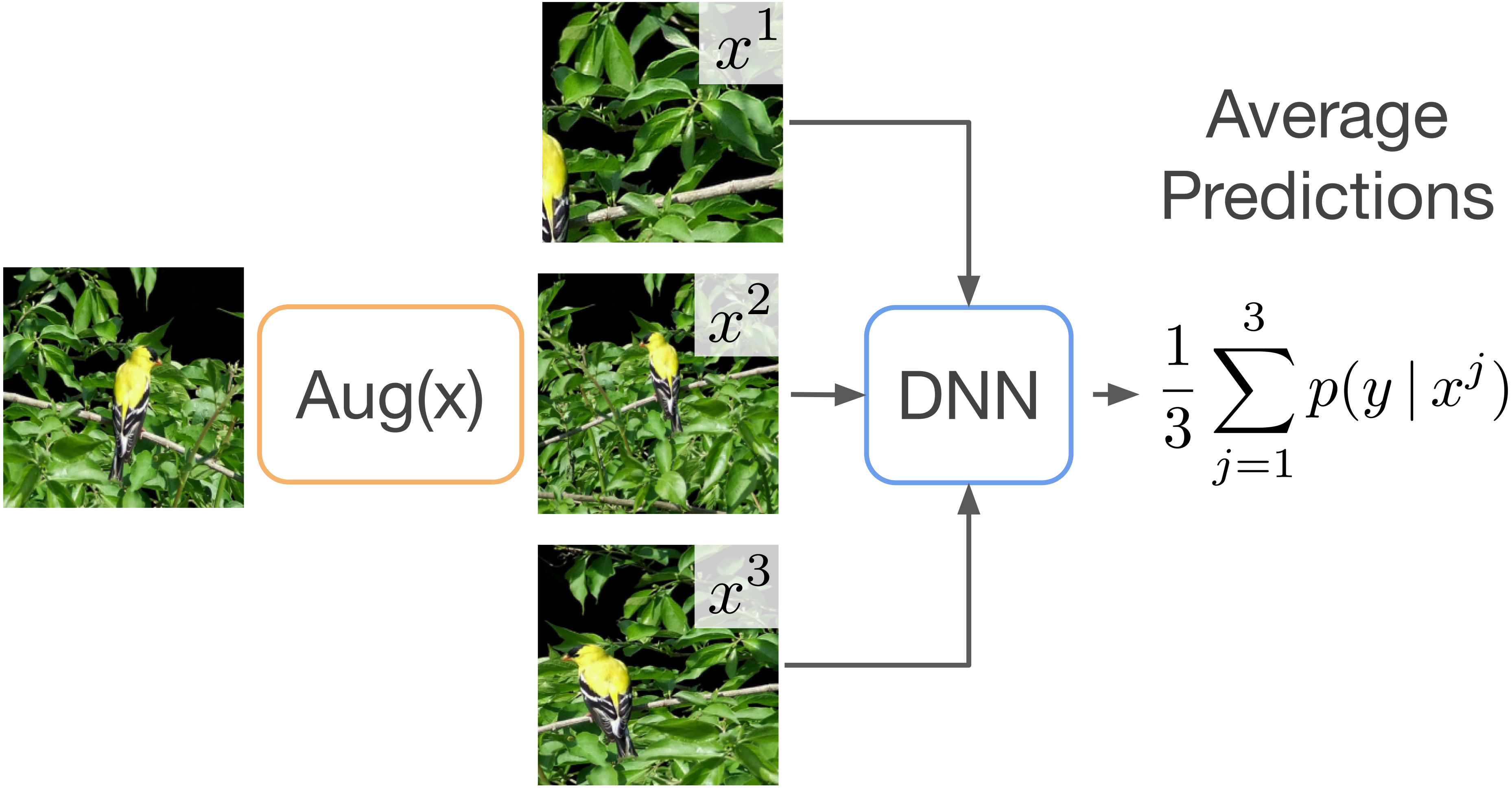
Now: learnable data augmentation policies

- AutoAugment (Cubuk, et al. 2018)
- RandAugment (Cubuk, et al. 2019)
- Adversarial AutoAugment (Zhang, et al. 2019)
- AugMix (Hendrycks, et al. 2019)
- ...

Transformation
Identity
ShearX
ShearY
TranslateX
TranslateY
Rotate
Autocontrast
Solarize
SolarizeAdd
Posterize
Contrast
Brightness
Color
Sharpness
Cutout

Transformations available for learnable augmentations



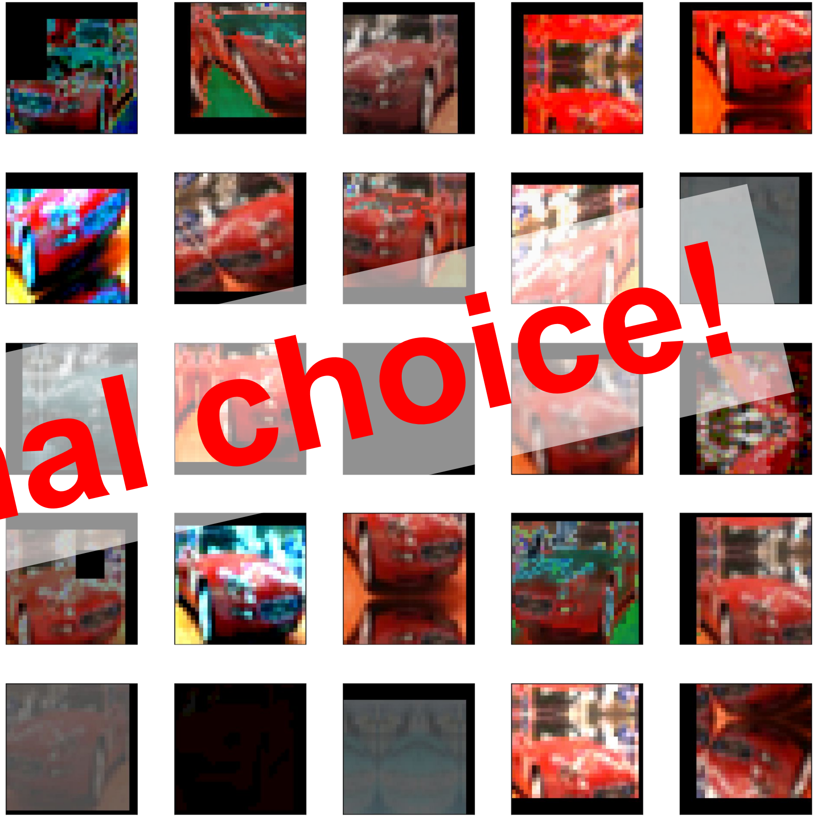


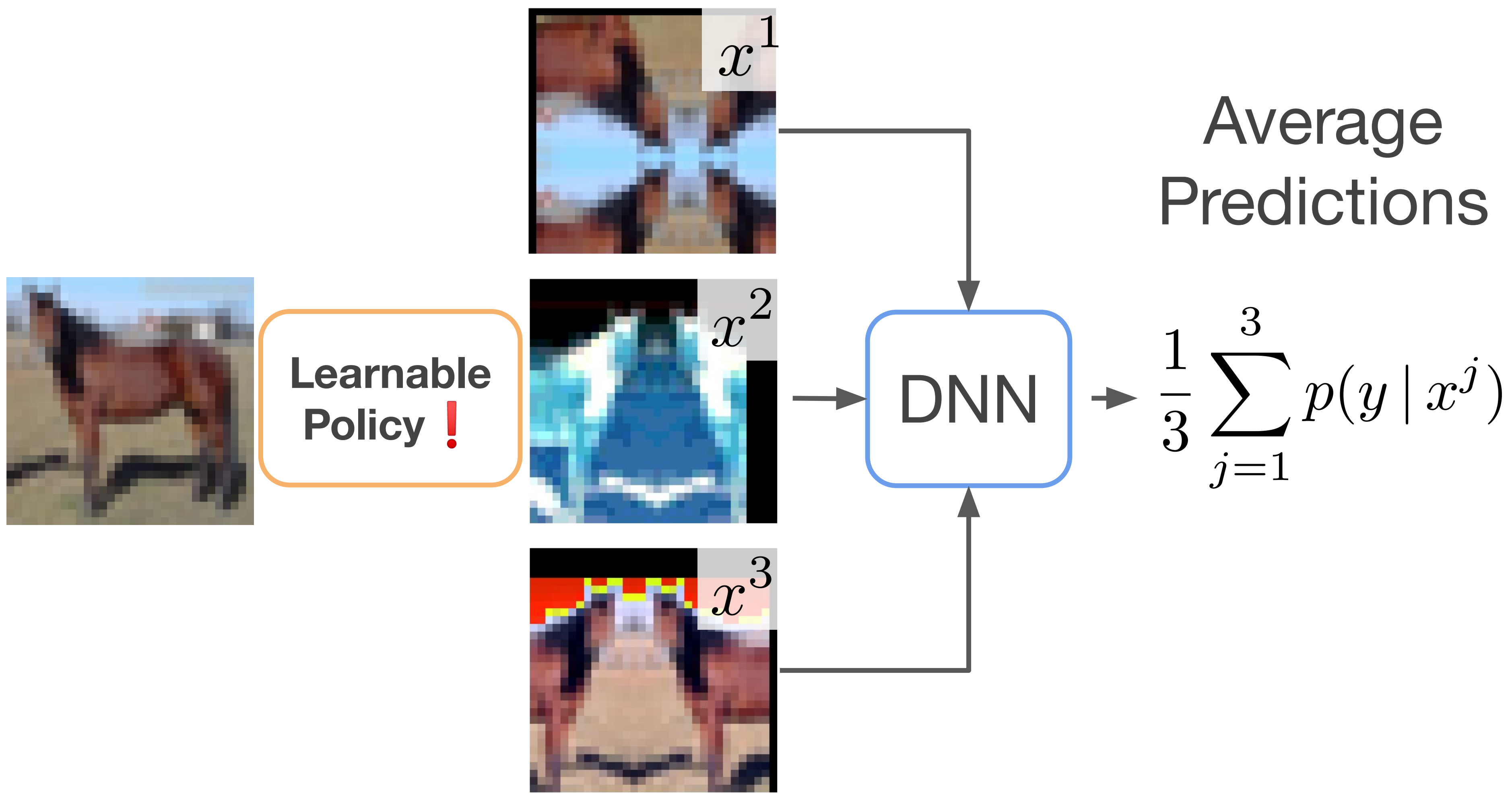
Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, Imagenet classification with deep convolutional neural networks, 2012

Conventional test-time augmentation

- 5-crop / 10-crop evaluation
- Same augmentation as during training

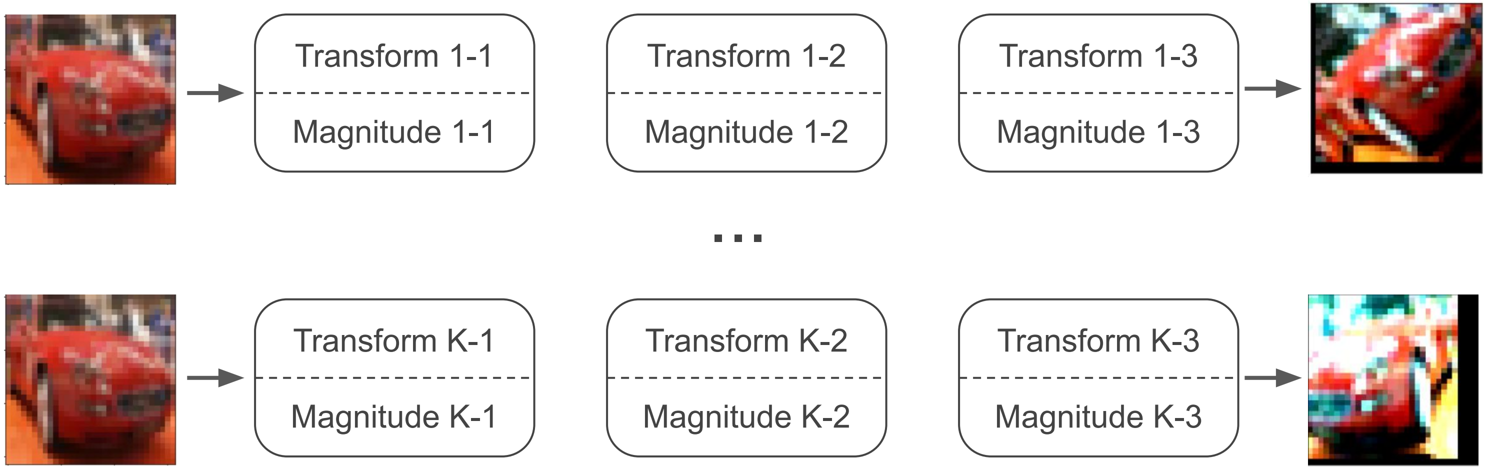
RandAugment(N=3, M=45)





Learnable Policy ❗
x^1
x^2
x^3
DNN
Average Predictions
$\frac{1}{3}\sum_{j=1}^{3} p(y \mid x^j)$

Learnable test-time augmentations

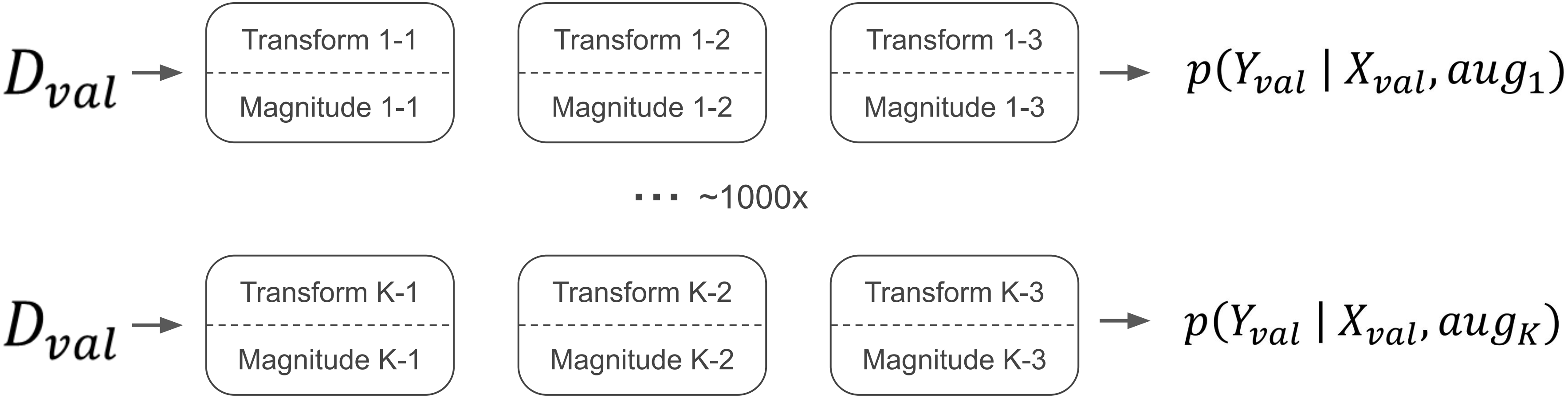


Given a fixed trained DNN…
…learn test-time augmentation policy by optimizing the performance on validation data.

How to learn? Greedy selection.

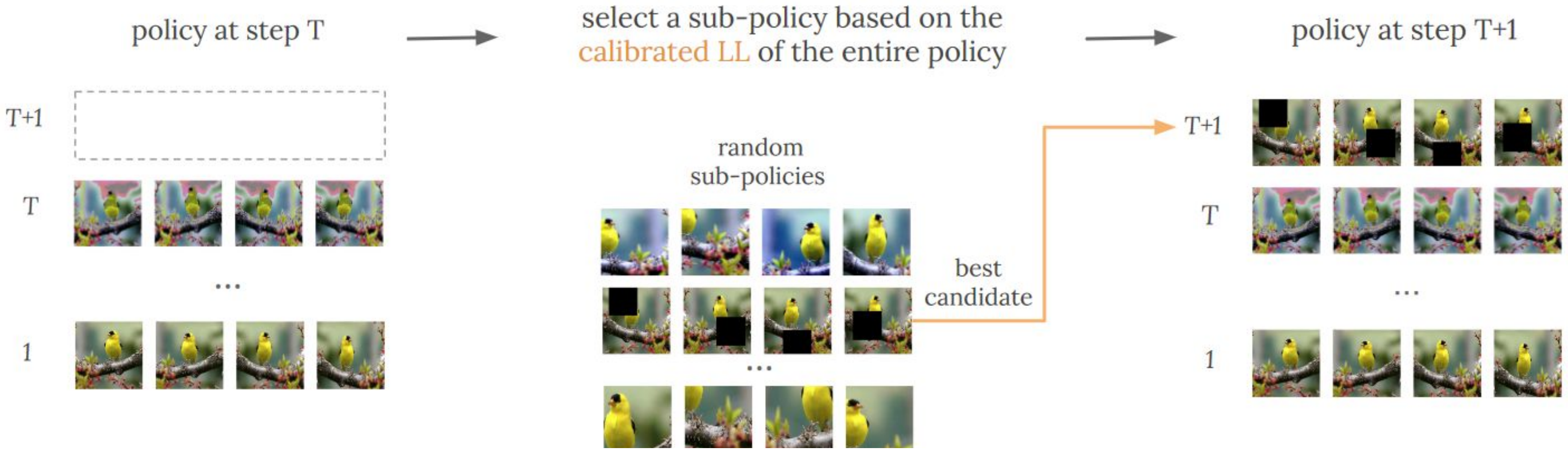
Greedy Policy Search

1. Create a pool and collect the predictions on validation set
   - Sample ~1000 random sub-policies with different magnitudes
   - Collect predictions using those sub-policies (1000 x size of validation set)



Greedy Policy Search

1. Create a pool and collect the predictions on validation set
   - Sample ~1000 random sub-policies with different magnitudes
   - Collect predictions using those sub-policies (1000 x size of validation set)
2. Find an augmentation that works best when added to the current policy
   - Try to add each augmentation to the current policy
   - Average the predictions over the assembled policy to compute the loss
   - Choose the best candidate and add it permanently



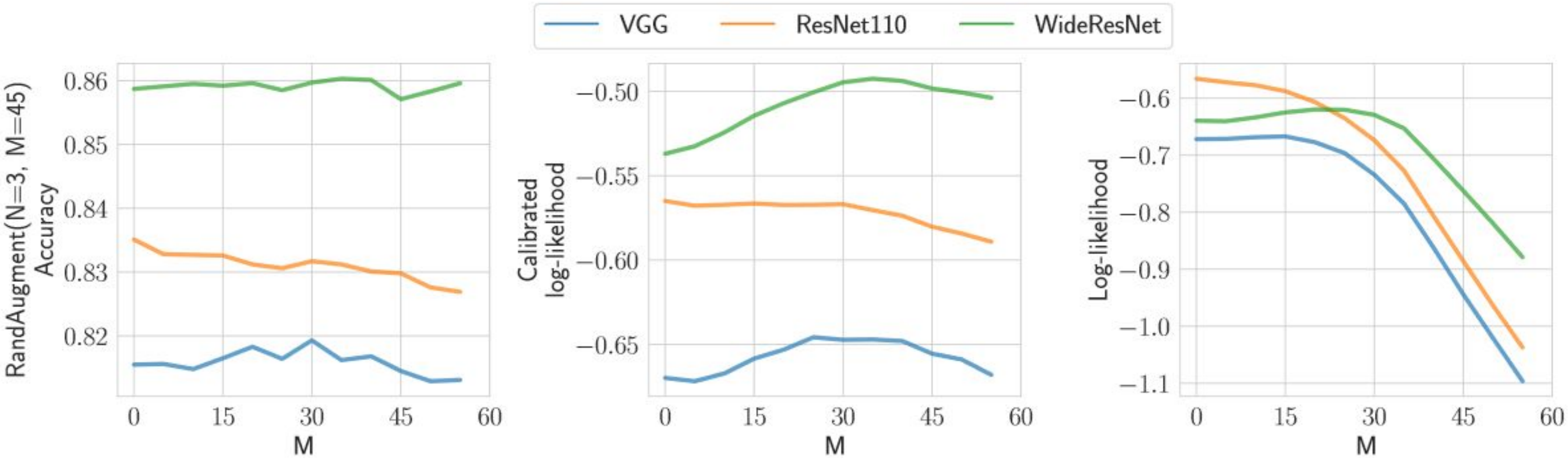
Greedy Policy Search: what objective to use?

	GPS criterion	VGG	ResNet110	WideResNet
Acc.(%)	Acc.	81.17 ± 0.15	83.01 ± 0.18	85.71 ± 0.10
	LL	81.89 ± 0.07	$\mathbf{83.55 \pm 0.09}$	86.22 ± 0.05
	cLL	$\mathbf{82.21 \pm 0.17}$	$\mathbf{83.54 \pm 0.06}$	$\mathbf{86.44 \pm 0.05}$
cLL	Acc.	-0.837 ± 0.003	-0.691 ± 0.001	-0.661 ± 0.003
	LL	-0.640 ± 0.001	-0.560 ± 0.001	-0.489 ± 0.001
	cLL	$\mathbf{-0.623 \pm 0.001}$	$\mathbf{-0.552 \pm 0.001}$	$\mathbf{-0.479 \pm 0.001}$

Calibrated log-likelihood is a much better target than log-likelihood or accuracy!

Greedy Policy Search: what objective to use and why?

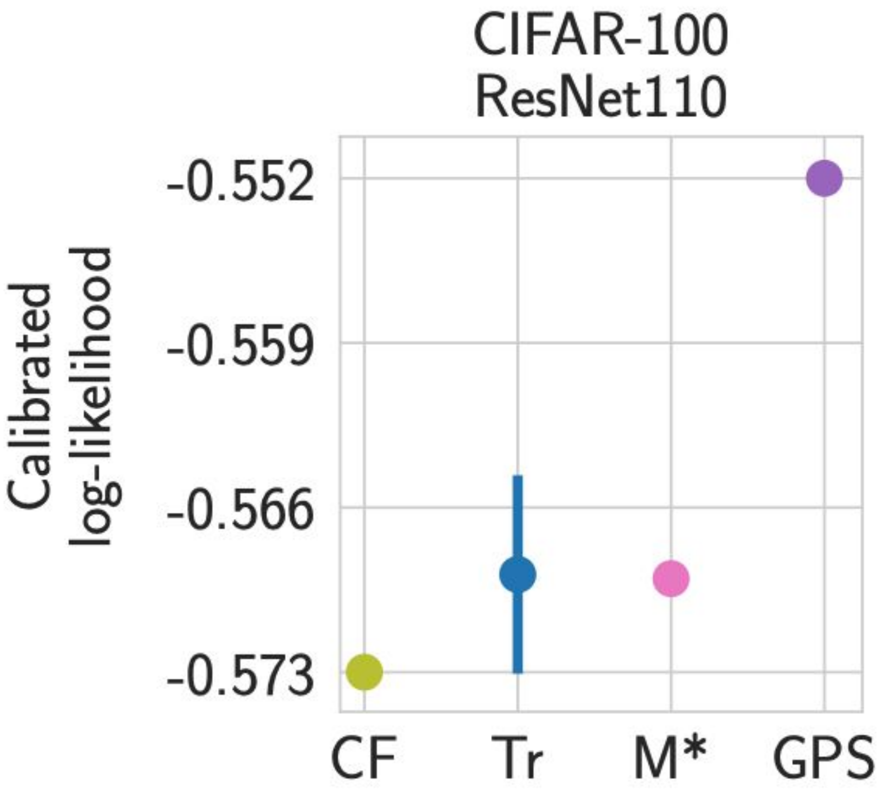
Find the optimal magnitude for TTA with RandAugment

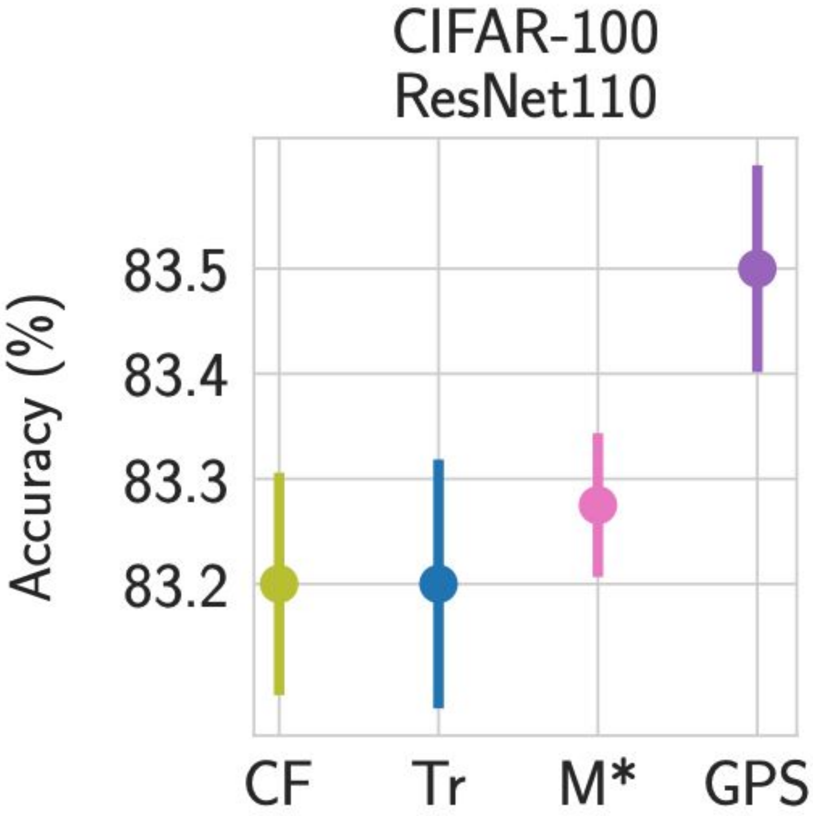


Accuracy is too noisy

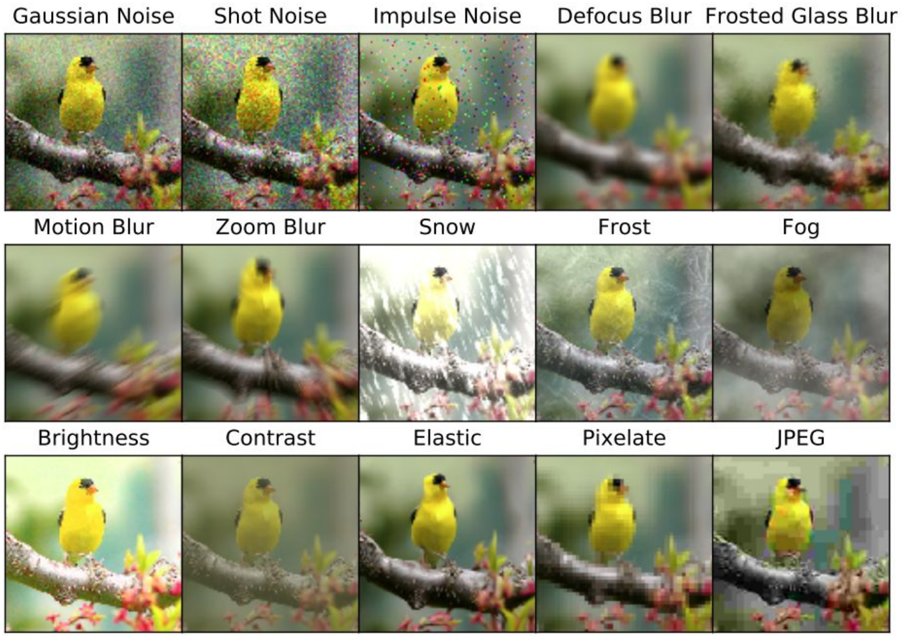
Strong augmentations decalibrate the model

Results of in-domain uncertainty estimation

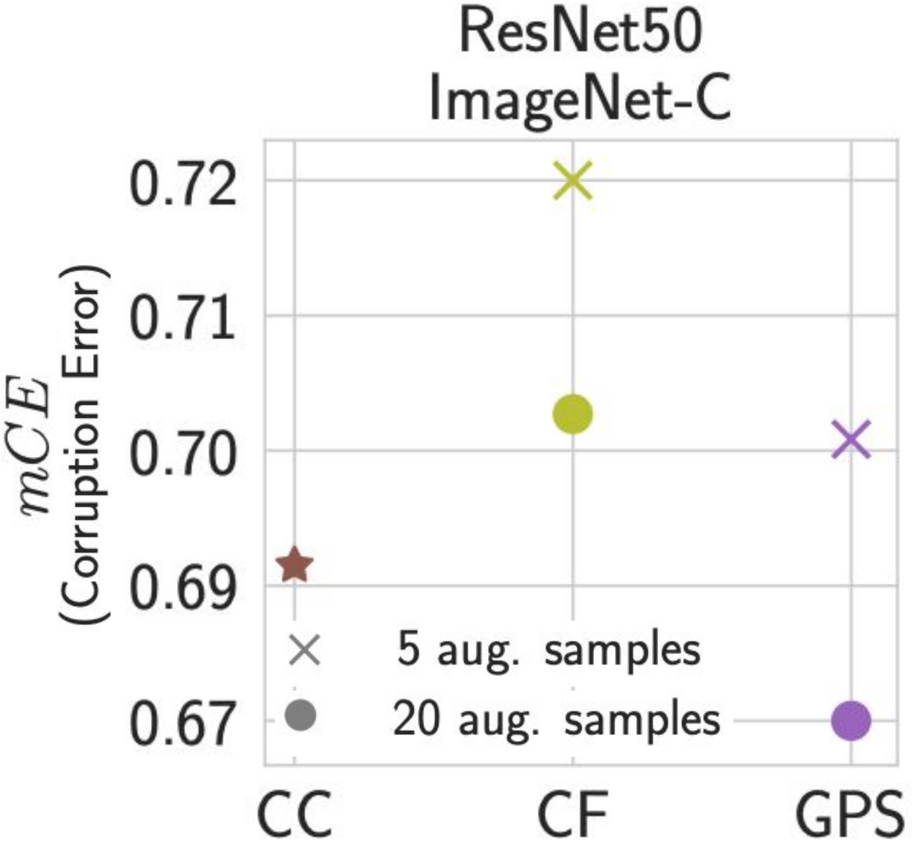




Results under domain shift



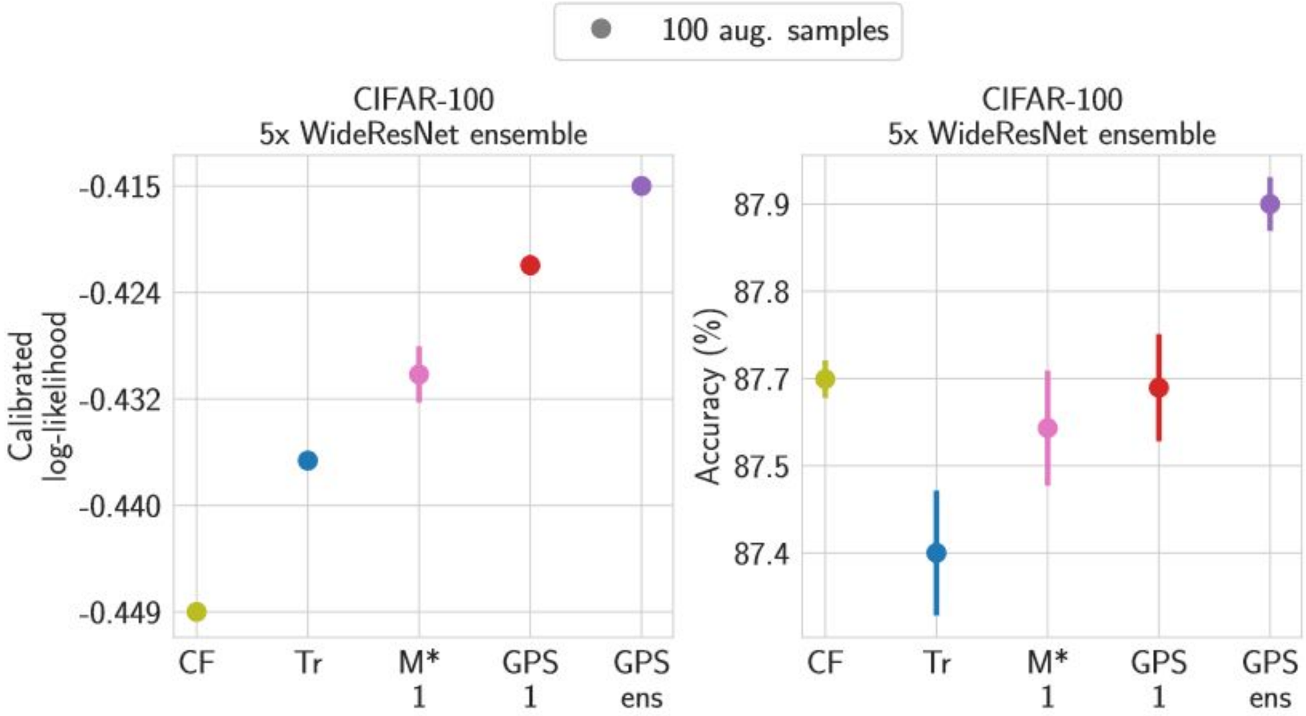
Hendrycks & Dietterich, Benchmarking Neural Network Robustness to Common Corruptions and Perturbations, 2019



Do policies transfer?

Evaluate policy on		Search policy on: CIFAR10			CIFAR100			
		VGG	ResNet	WRN	VGG	ResNet	WRN	Crop/flip policy
CIFAR10	VGG	0.000	-0.002	-0.002	-0.004	-0.003	-0.006	-0.080
	ResNet	0.000	0.000	-0.000	-0.002	-0.001	-0.004	-0.052
	WRN	0.001	-0.000	0.000	-0.001	-0.000	-0.002	-0.058
CIFAR100	VGG	-0.015	-0.020	-0.008	0.000	-0.010	-0.003	-0.276
	ResNet	-0.001	-0.004	-0.001	-0.001	0.000	-0.003	-0.219
	WRN	-0.018	-0.015	-0.009	0.001	-0.006	0.000	-0.266

Does it work for ensembles?



Greedy Policy Search: conclusions

- Test-time augmentation policies **can** and **should** be learned.
    - Better predictive performance and log-likelihood
    - Works for a variety of single models and ensembles
    - Transferable policies

- Using calibrated log-likelihood is a must!
  Likely important for other problems (e.g., NAS and meta-learning)

@bayesgroup

arXiv

arxiv.org/abs/2002.09103

GitHub

bayesgroup/gps-augment