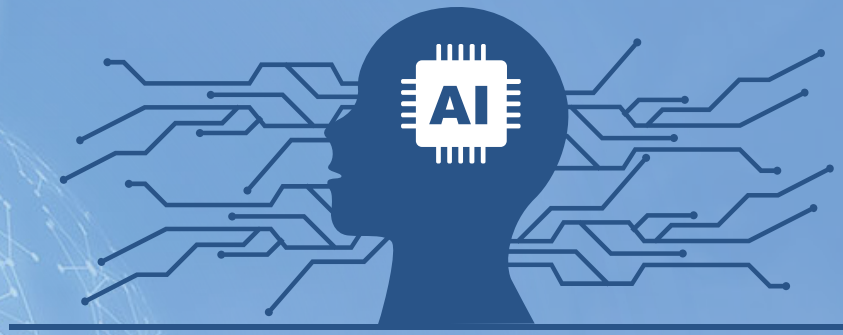
Artificial Intelligence



Machine Learning

Wen-Huang Cheng (鄭文皇)

National Taiwan University

wenhuang@csie.ntu.edu.tw

A Dataset and Model for Realistic License Plate Deblurring

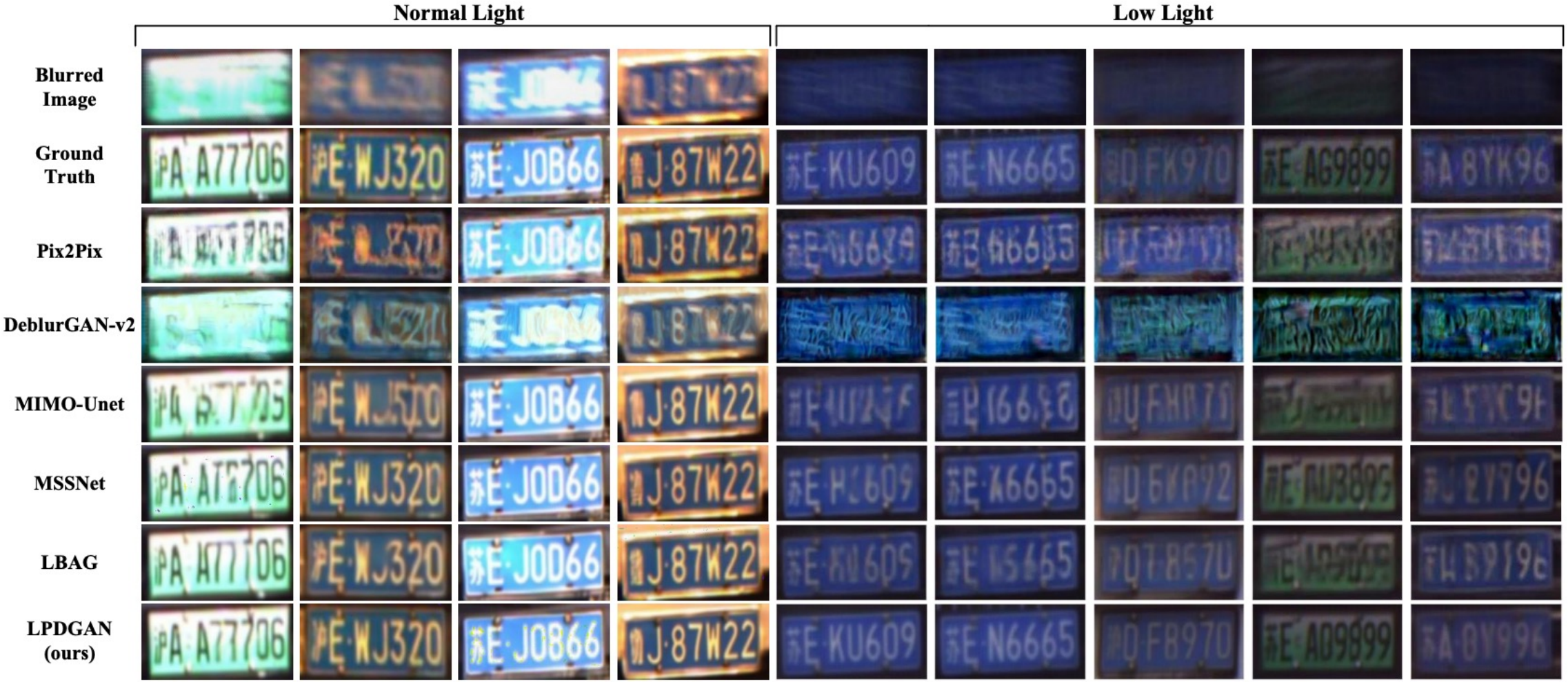
Haoyan Gong, **Yuzheng Feng**, **Zhenrong Zhang**, **Xianxu Hou**, **Jingxin Liu**, **Siqi Huang** and **Hongbin Liu** *

School of AI and Advanced Computing, Xi'an Jiaotong-Liverpool University

{haoyan.gong21, yuzheng.feng21, zhenrong.zhang21}@student.xjtlu.edu.cn, {xianxu.hou, jingxin.liu, siqi.huang, hongbin.liu}@xjtlu.edu.cn

The proposed License Plate Blur (LPBlur) dataset is partitioned into a training set with 9,288 image pairs, and a test set with 1,000 image pairs.

AI Weekly


Normal Light
Low Light
Blurred Image
Ground Truth
Pix2Pix
DeblurGAN-v2
MIMO-Unet
MSSNet
LBAG
LPDGAN (ours)

An entirely AI-generated Terminator 2 remake crafted by 50 artists has its LA premiere on March 6

Our T2 Remake reimagines James Cameron's action classic, starring Arnold Schwarzenegger, using AI tools and 50 artists.

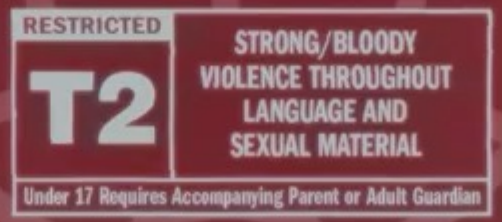
BY STEVE SEIGH FEBRUARY 23RD 2024, 1:55PM

1 12 20%

Last Updated on February 26, 2024

"Our T2 Remake" is a NON-PROFIT parody fan film created for private use only and is in no way intended for commercial enterprise of any kind. This parody fan film is not affiliated with - and is neither sponsored nor approved by - the current owners of the TERMINATOR franchise. It makes no claim to any footage, characters, storylines, prop design, names, logos or situations that are trademarked, copyrighted, or otherwise protected by federal, state, international or other intellectual property law. This work is produced solely for the personal, uncompensated enjoyment of this community of artists, for the purpose of scientific artificial intelligence research and experimentation, and for other fans of the TERMINATOR franchise.

www.t2remake.com



AI Weekly

ElevenLabs 
@elevenlabsio

Got a story to tell?

Together with @pika_labs, we're launching our very first 72-hour FilmFAST. Join us April 12-14 and create a short film using ElevenLabs & Pika.

Sign up to compete via the link below and stay tuned for an announcement of the theme tomorrow.

forms.gle/B1uqVR6HnCp25p...

翻譯貼文

下午10:26 · 2024年4月10日 · **5.6萬** 次查看

OpenStreetView-5M: The Many Roads to Global Visual Geolocation

Guillaume Astruc* [1,2,5] Nicolas Dufour* [1,6] Ioannis Siglidis* [1] Constantin Aronssohn [1]
Nacim Bouia [1] Stephanie Fu [1,4] Romain Loiseau [1,2] Van Nguyen Nguyen [1]
Charles Raude [1] Elliot Vincent [1,3] Lintao XU [1] Hongyu Zhou [1] Loic Landrieu[1]

[1] LIGM, Ecole des Ponts, CNRS, UGE [2] UGE, IGN, ENSG, LASTIG [3] Inria Paris [4] UC Berkeley

[5] CESBIO, Univ de Toulouse, CNES/CNRS/IRD/INRAE/UPS [6] LIX, CNRS, Ecole Polytechnique, IP Paris

drivephotograph, and_eng, gclem, bootprint, Mapillary, licensed under CC-BY-SA.

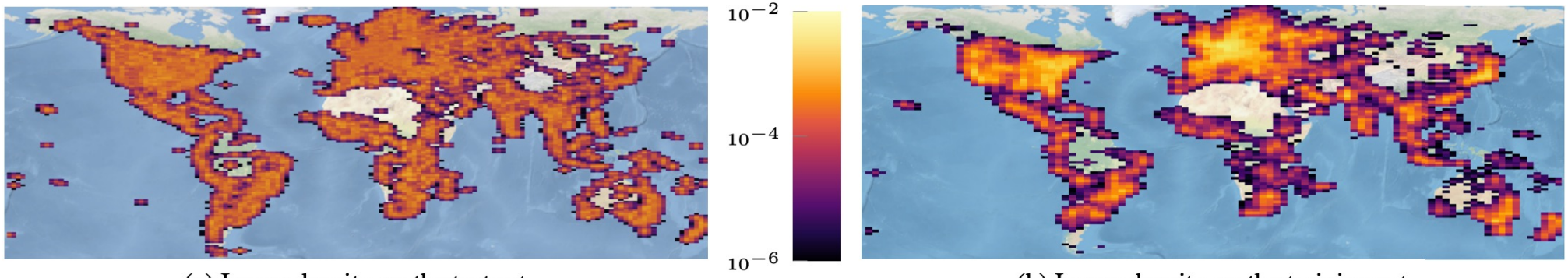
climate/vegetation traffic markers architecture culture/script

Figure 1. **Global Visual Geolocation.** Predicting the location of an image taken anywhere in the world from just pixels requires detecting a combination of clues of various abstraction levels [44]. Can you guess where these images were taken?[1]

bob3bob3, jthnz, jerome_pietri, nurzham, Mapillary, licensed under CC-BY-SA.

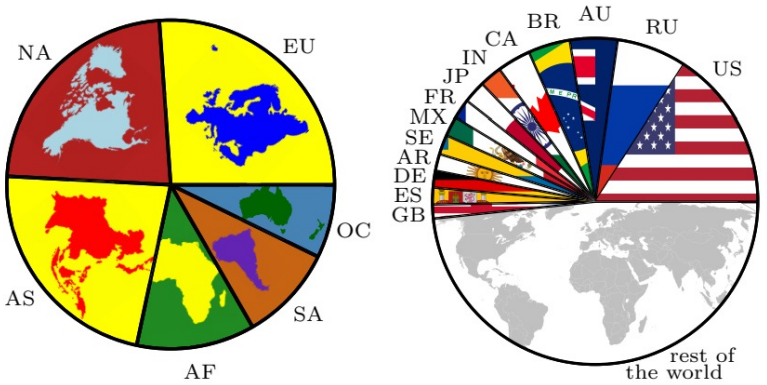
non-localizable ⟶ localizable ⟶ landmarks

Figure 2. **Localizable vs Non-Localizable.** Images from our dataset (green) occupy the space between weakly localizable images (red) like the ones from the test set of Im2GPS3k [70] and landmark images used to advertise CV conferences (blue).

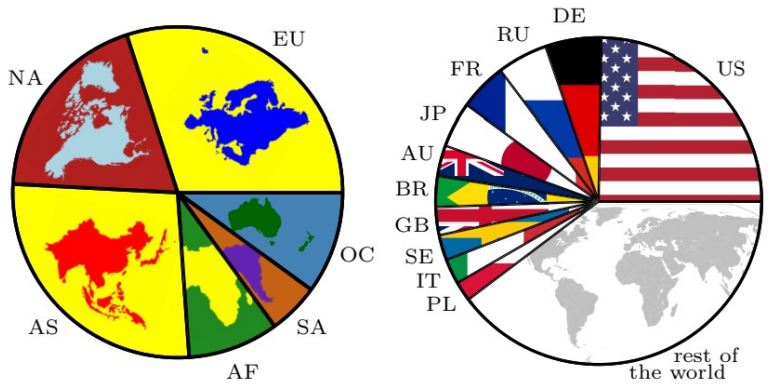


(a) Image density on the test set

(b) Image density on the training set



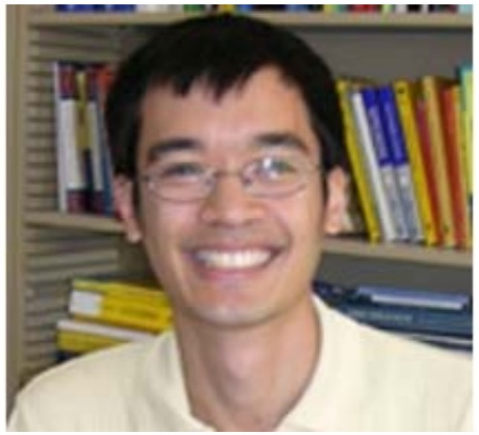
(c) Continent and country distributions of the test set



(d) Continent and country distributions of the training set

AI Weekly

陶哲軒是澳大利亞人。**7** 歲開始自學微積分，**8** 歲半升入中學，九歲去學院上數學課，**11** 歲讀微積分，**1986** 年，**11** 歲的他就在華沙獲得了奧數銅牌；**1987** 年，在哈瓦那，他獲得銀牌；**1988** 年，**12** 歲他獲得在坎培拉奧數金牌，**20** 歲獲得普林斯頓大學博士學位，陶哲軒那時每天都在玩計算器，但他絕頂聰明，是個天才。**24** 歲被洛杉磯加州大學聘為正教授。



陶哲軒

Terence Tao
@tao@mathstodon.xyz

Over the past six months, the #PCAST working group in #AI that I have been co-leading with Laura Greene has been working to fulfill a tasking from part of an executive order of the President (see whitehouse.gov/briefing-room/p...) to report on the potential impacts of AI technologies on research (particularly those aimed at societal and global challenges), and what barriers exist to fully achieving the benefits of AI assistance in this space. I'm pleased to announce that the report is now complete and can be found at whitehouse.gov/wp-content/uplo... (see also the press release at whitehouse.gov/pcast/briefing-...), as part of a longer list of outcomes from the executive order that can be found at whitehouse.gov/briefing-room/s... .

In our report (the main body of which is about 40 pages in length, with a five page executive summary), we give multiple vignettes from across the sciences in which AI tools are already transforming the field. We identified three key pillars that require attention to fully benefit from this transformation: empowerment of human scientists; responsible use of AI tools; and sharing of basic AI resources. We then have five recommendations to further these goals, including the full funding of the National AI Research Resource (#NAIRR) and requiring reponsible AI use plans for any federally funded research project that uses AI tools.

A presentation of the report to PCAST, together with a question and answer session, can be found at whitehouse.gov/pcast/meetings/...

The White House · 2023年10月31日
Executive Order on the Safe, Secure, and ...
By the authority vested in me as President by the Constitutio...

2024年4月30日 01:54 · 編輯於 4月30日 22:59 · Web · 25 · 27

★★★

REPORT TO THE PRESIDENT

Supercharging Research: Harnessing Artificial Intelligence to Meet Global Challenges

Executive Office of the President

President's Council of Advisors on Science and Technology

APRIL 2024

This Lecture – Agenda

- **Reliability of Deep Learning for Real-World Deployment**
  - Uncertainty Estimation
    - Data Uncertainty
    - Knowledge Uncertainty
    - Ensemble Approaches
  - Adaptation to Unseen Concepts on the Fly
    - Dynamic Adaptation: Test-time Adaptation
    - Static Adaptation: Vision-Language Learning

This Lecture – Agenda

- **Reliability of Deep Learning for Real-World Deployment**
  - Uncertainty Estimation
    - Data Uncertainty
    - Knowledge Uncertainty
    - Ensemble Approaches
  - Adaptation to Unseen Concepts on the Fly
    - Dynamic Adaptation: Test-time Adaptation
    - Static Adaptation: Vision-Language Learning

The Reliability of Deep Learning for Real-World Deployment

- Deep learning models may make errors in predictions even in controlled test environment.
- Many more under distribution shifts
  - An example: “Tesla Autopilot driving directly into the sun”

https://www.youtube.com/watch?v=s1eTBMYEs-Y

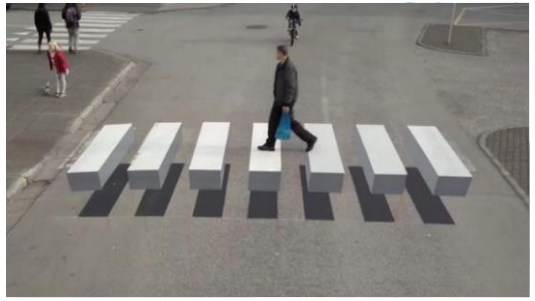
From intended to covered domain

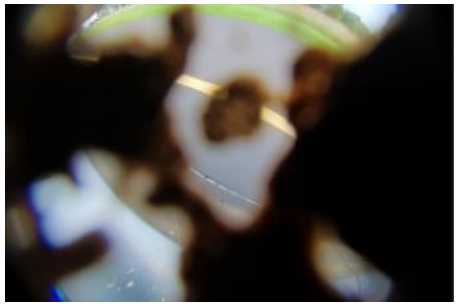
Dataset defines the actual domain, often with limited coverage of:

- Rare pose/appearance of known objects, rare objects
- Rare, e.g. dangerous, scene configurations
- All sorts of perturbation, e.g., adverse conditions, sensor blocking







Expectations for real-world AI systems

Useful and safe models should

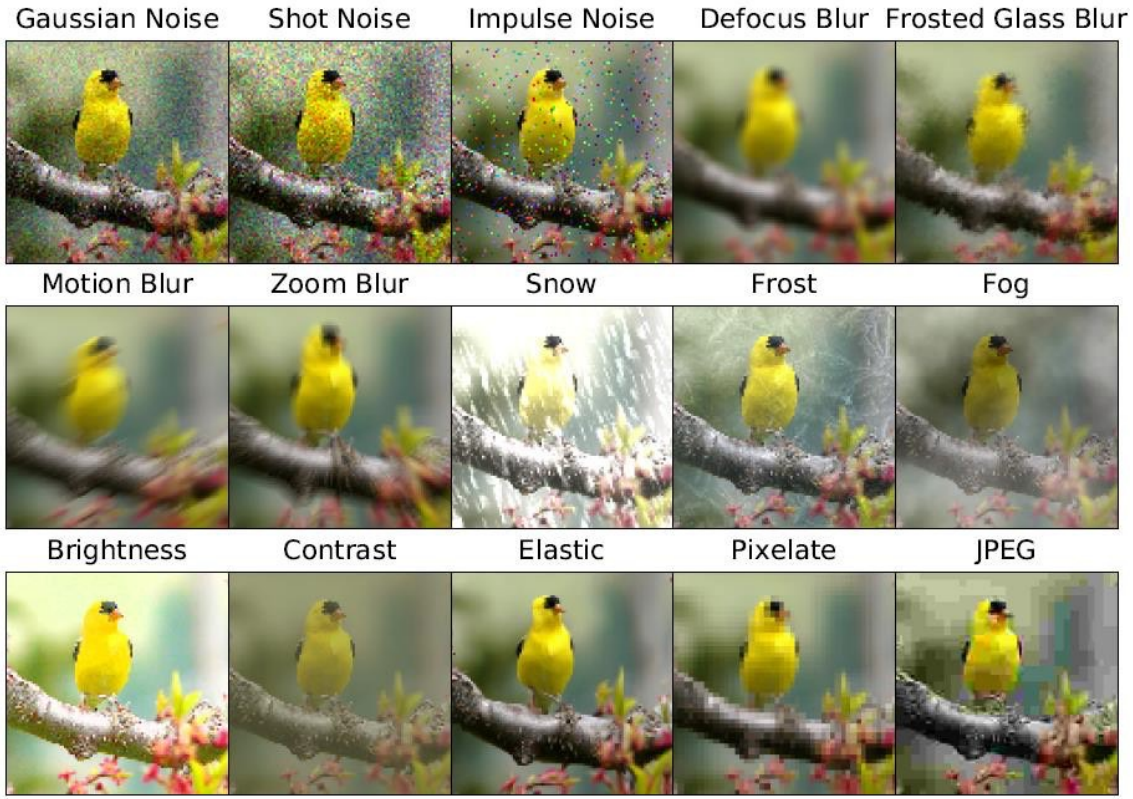
- Be **accurate** over *intended* domain, inc. corner cases
- Be **robust** to perturbations in-domain
- **Self-assess its confidence** for each prediction
- **Refuse to predict** if too uncertain, detect out-of-domain inputs
- **Adapt/generalize** to new domains or conditions

good test accuracy does not suffice

All faces of runtime reliability should be assed and improved

Assessing robustness to corruption

- Various types/degrees of synthetic corruption on val/test data
- Measure their influence on model performance

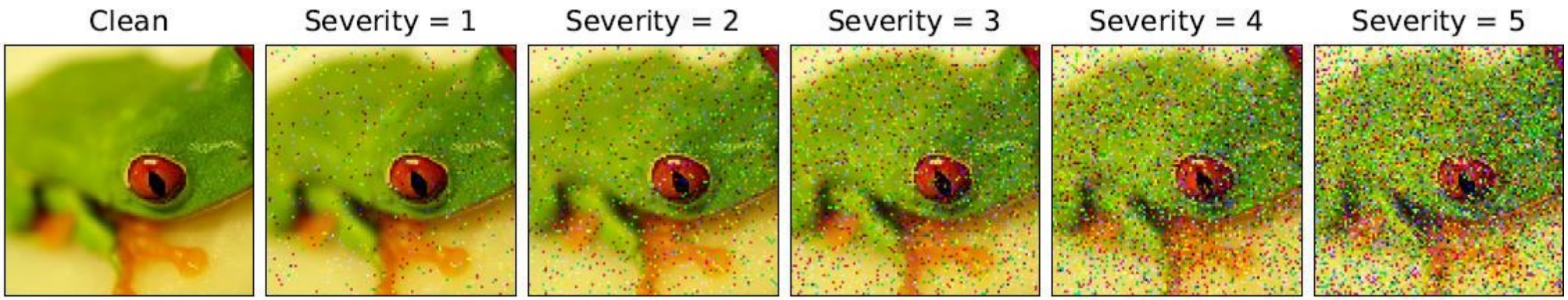


ImageNet-C

[Hendrycks ICLR 19]

Assessing robustness to corruption

- Various types/degrees of synthetic corruption on val/test data
- Measure their influence on model performance



ImageNet-C

[Hendrycks ICLR 19]

Uncertainty Estimation and Ensemble Approaches

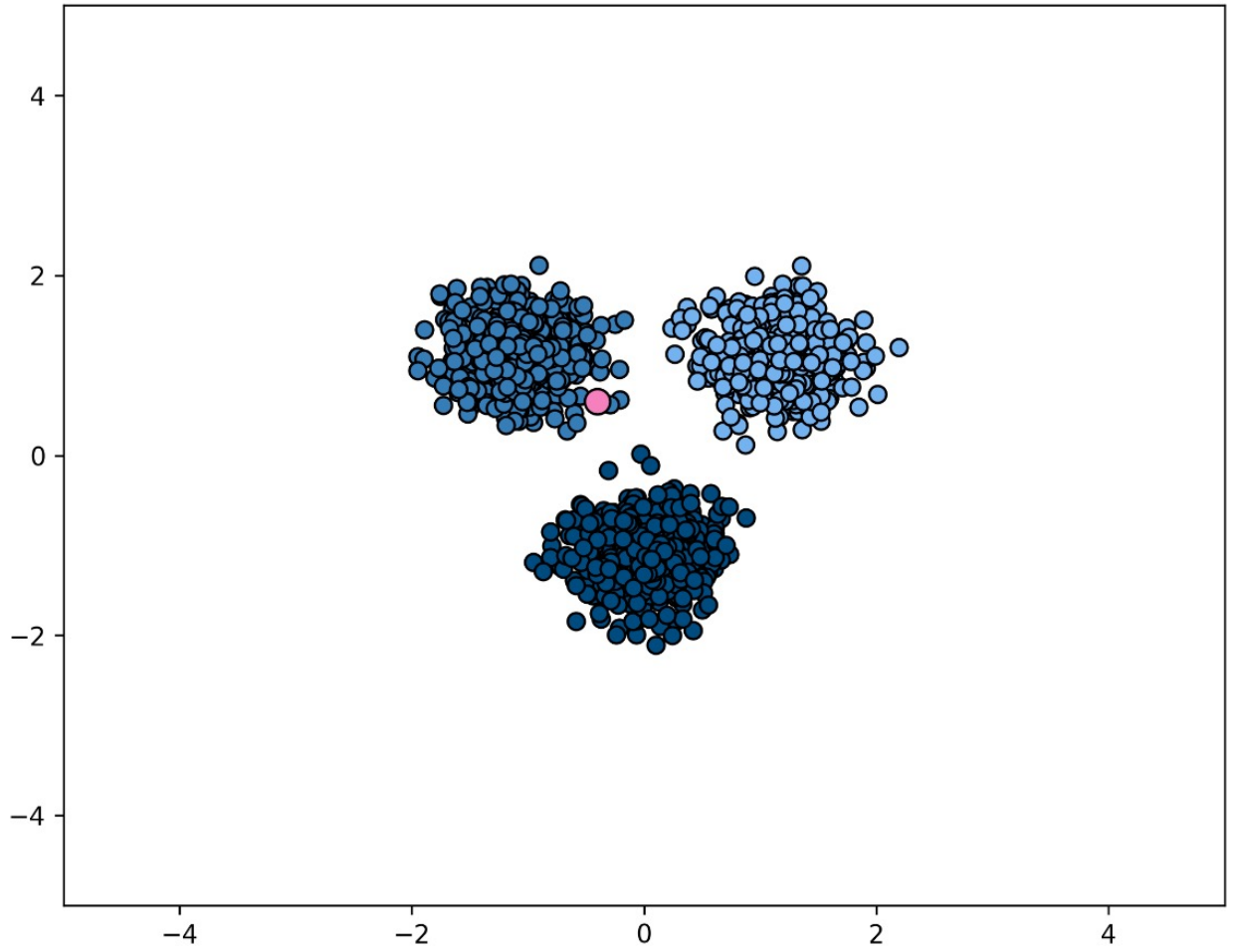
- Good uncertainty estimates quantify when we can trust the model's predictions → helps avoid mistakes.

- Uncertainty estimation is an essential function for improving reliability and safety of systems running on machine learning models.

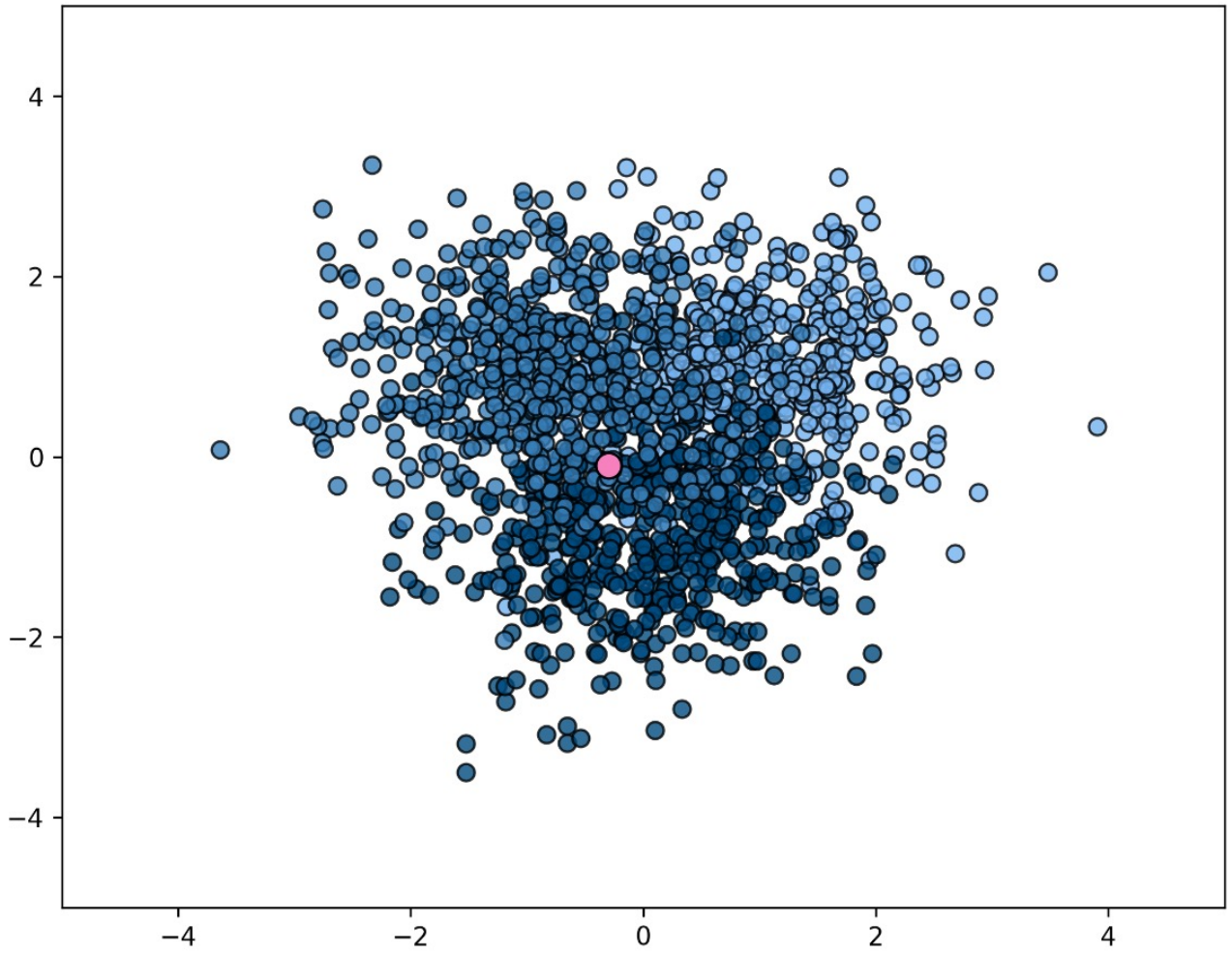
Sources of Uncertainty

Case 1

Sources of Uncertainty



Sources of Uncertainty



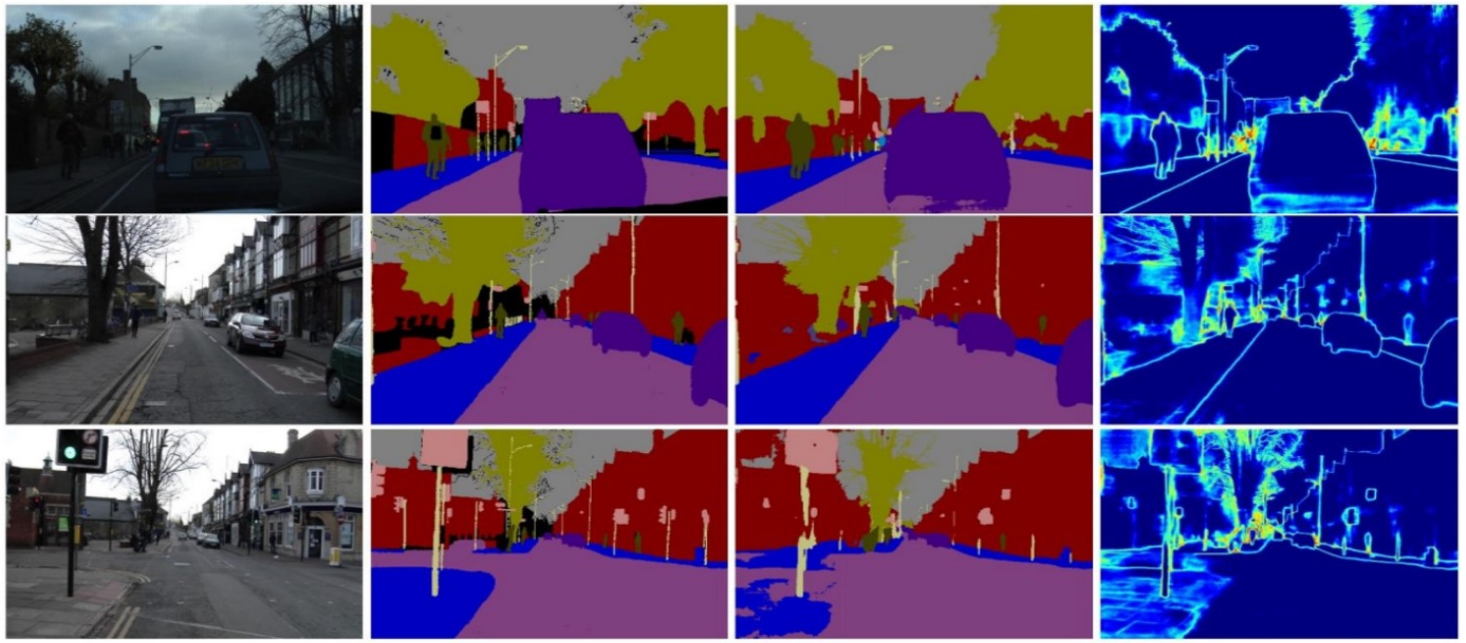
Aleatoric / Data uncertainty

Data/Aleatoric Uncertainty

Similarly looking objects also fall into this category.

Data/Aleatoric Uncertainty

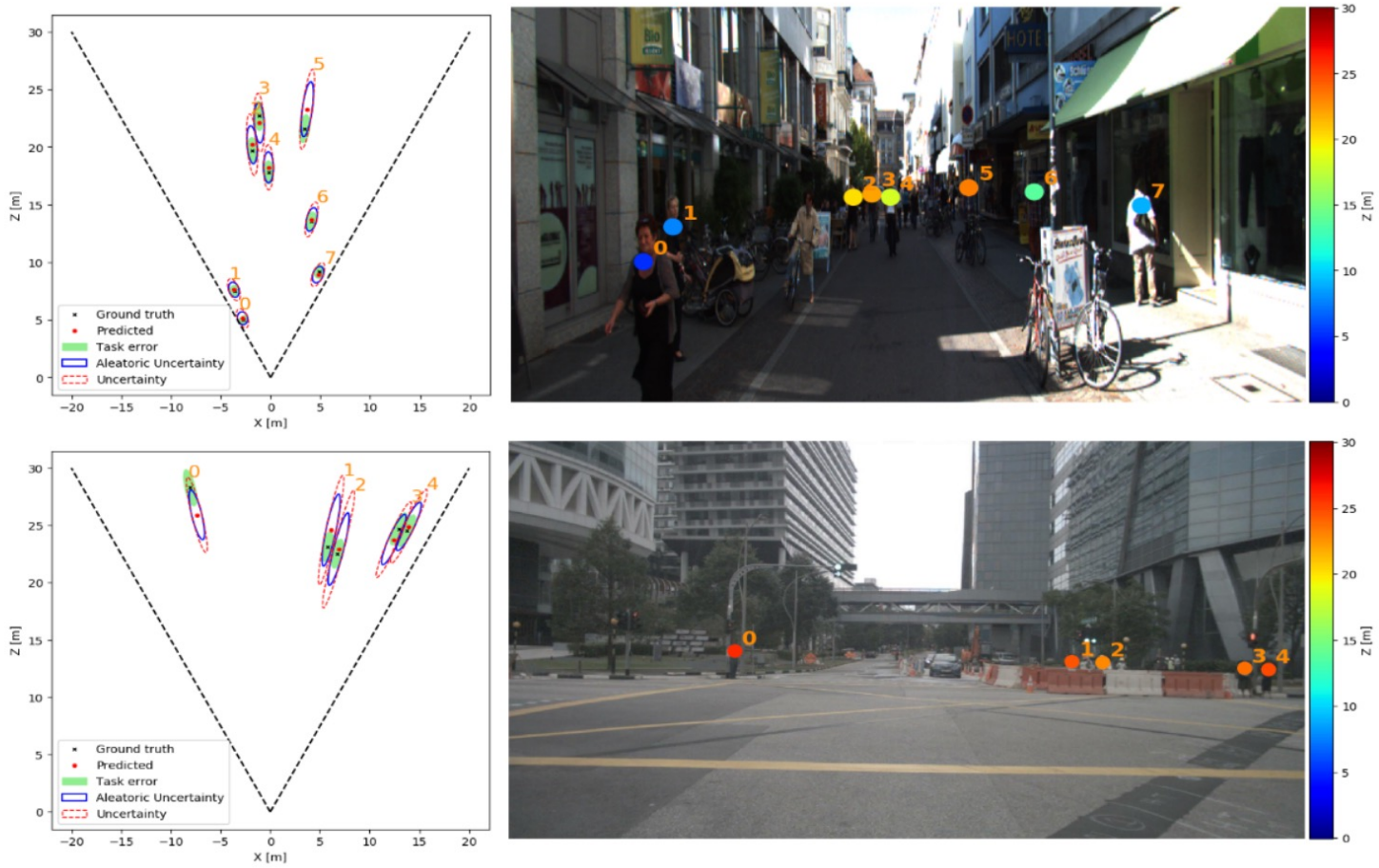
Data/Aleatoric Uncertainty



(a) Input Image (b) Ground Truth (c) Semantic Segmentation (d) Aleatoric Uncertainty

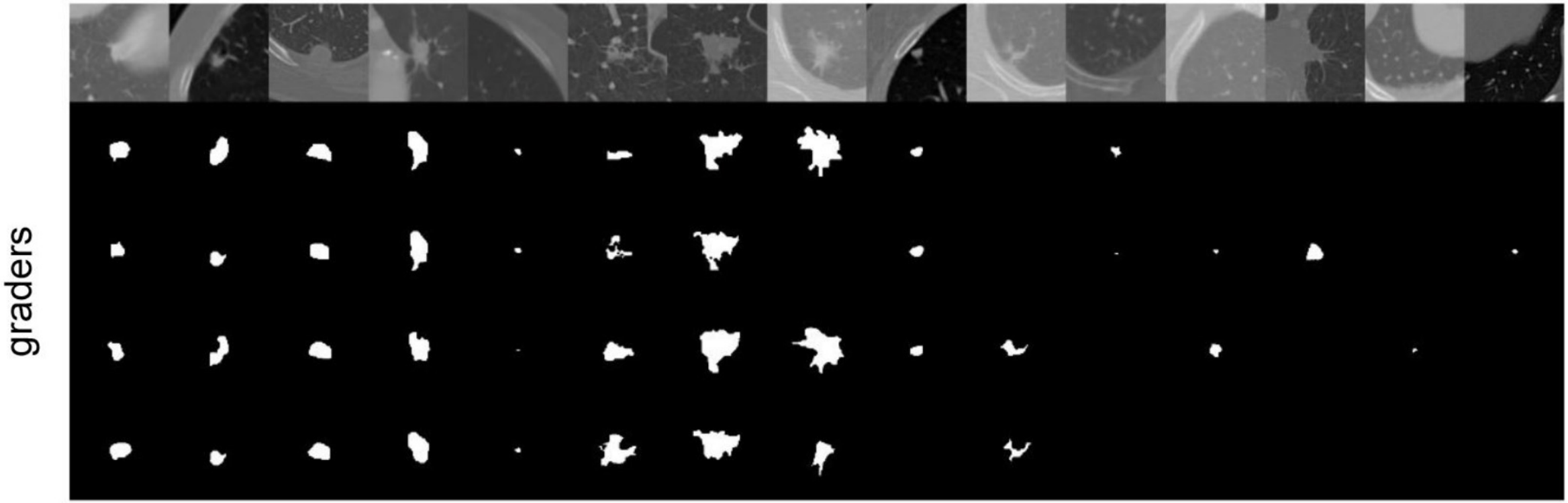
- In urban scenes this type of uncertainty is frequently caused by similarly-looking classes:
  - pedestrian - cyclist - person on scooter
  - road - sidewalk
  - also at object boundaries

Data/Aleatoric Uncertainty



Also caused by sensor limitations: localization and recognition of far-away objects is less precise.
Datasets with low resolution images, e.g., CIFAR, also expose this ambiguity.

Data/Aleatoric Uncertainty



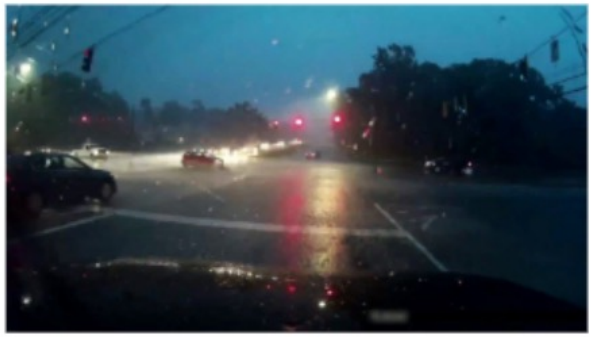
Samples and annotations from different graders on LIDC-IDRI dataset.

Difficult or ambiguous samples with annotation disagreement

Data/Aleatoric Uncertainty

Rain drops

Glare



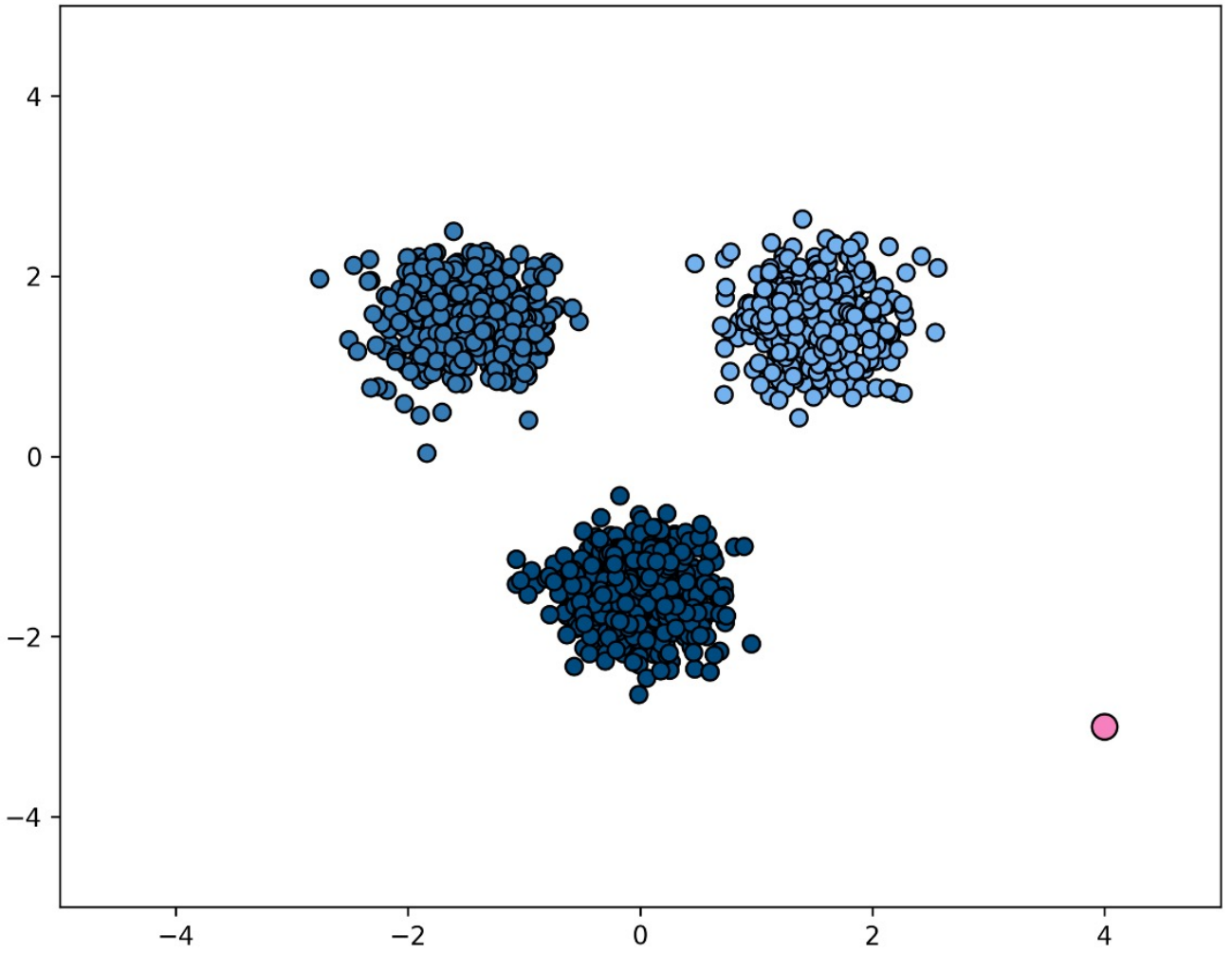
Low light

- Data uncertainty is often encountered in practice due to sensor quality, natural randomness, that cannot be explained by our data.
- It cannot be reduced (irreducible uncertainty), but can be learned. Could be reduced with better measurements.
- In layman words data uncertainty is called the: known unknown

Sources of Uncertainty

Case 2

Sources of Uncertainty



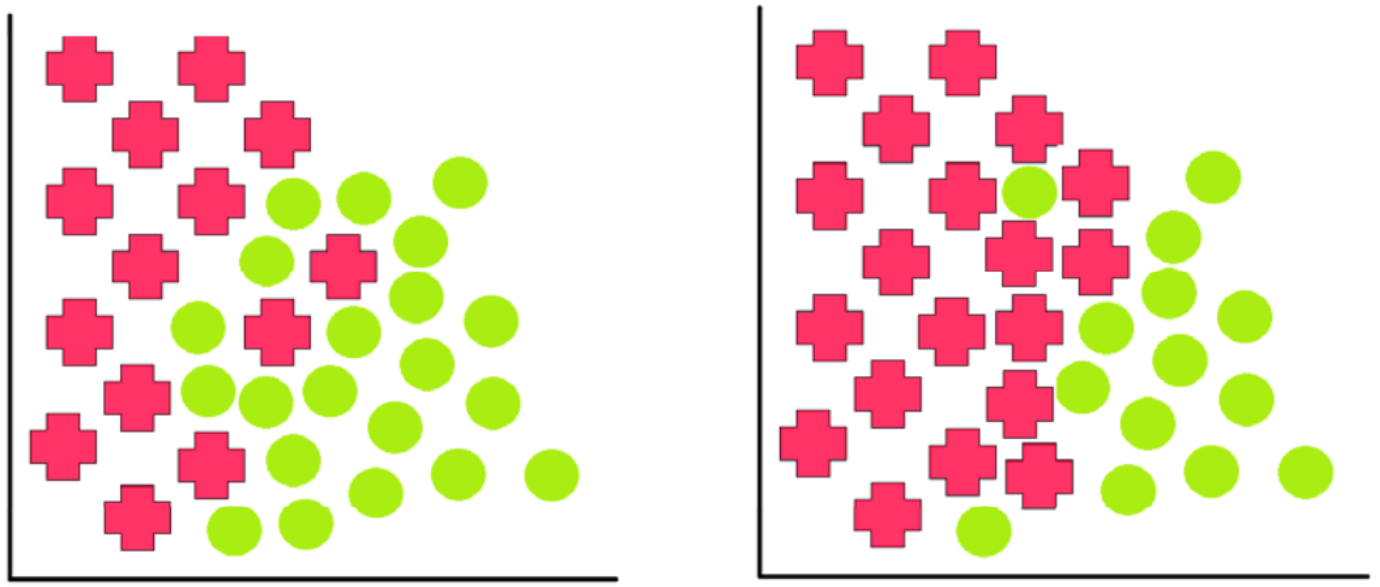
Epistemic / Knowledge uncertainty

Sources of Uncertainty

Independent and identically distributed (IID) assumption:
training and testing data are drawn from the same distribution

I.I.D.: $p_{\text{train}}(x, y) = p_{\text{test}}(x, y)$

O.O.D.: $p_{\text{train}}(x, y) \neq p_{\text{test}}(x, y)$



An example of differing training and test distributions.

Sources of Uncertainty

Independent and identically distributed (IID) assumption:
training and testing data are drawn from the same distribution

I.I.D.: $p_{\text{train}}(x, y) = p_{\text{test}}(x, y)$ O.O.D.: $p_{\text{train}}(x, y) \neq p_{\text{test}}(x, y)$

There are different forms of out-of-distribution / distribution shift:

Sources of Uncertainty

Independent and identically distributed (IID) assumption:
training and testing data are drawn from the same distribution

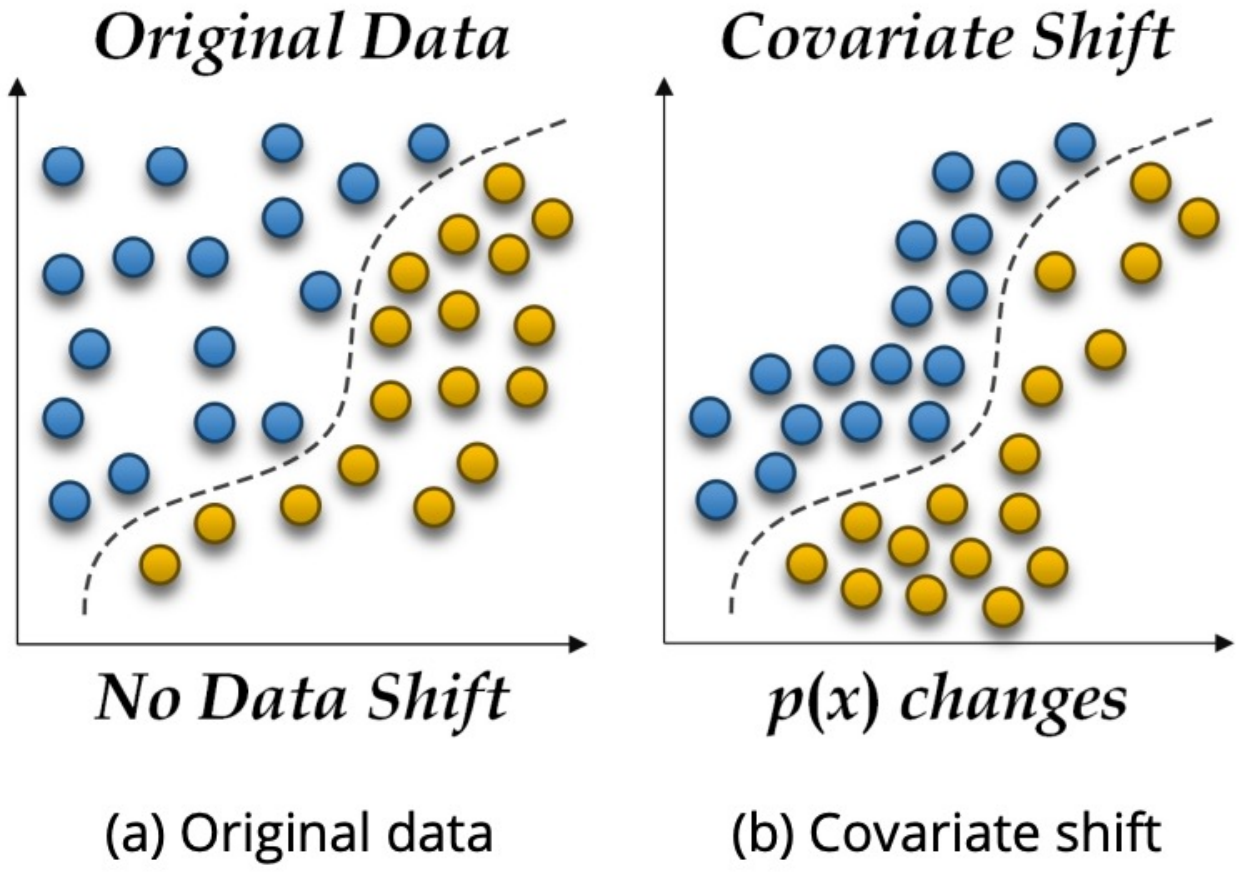
I.I.D.: $p_{\text{train}}(x, y) = p_{\text{test}}(x, y)$ O.O.D.: $p_{\text{train}}(x, y) \neq p_{\text{test}}(x, y)$

There are different forms of out-of-distribution / distribution shift:

- covariate shift: distribution of $p(x)$ changes while $p(y \mid x)$ remains constant

Covariate shift appears in data due to inadequate sampling, biased sampling, and non-stationary environment.

Covariate Shift



(a) Original data

(b) Covariate shift

covariate shift represents a shift in the input distribution

Sources of Uncertainty

Independent and identically distributed (IID) assumption:
training and testing data are drawn from the same distribution

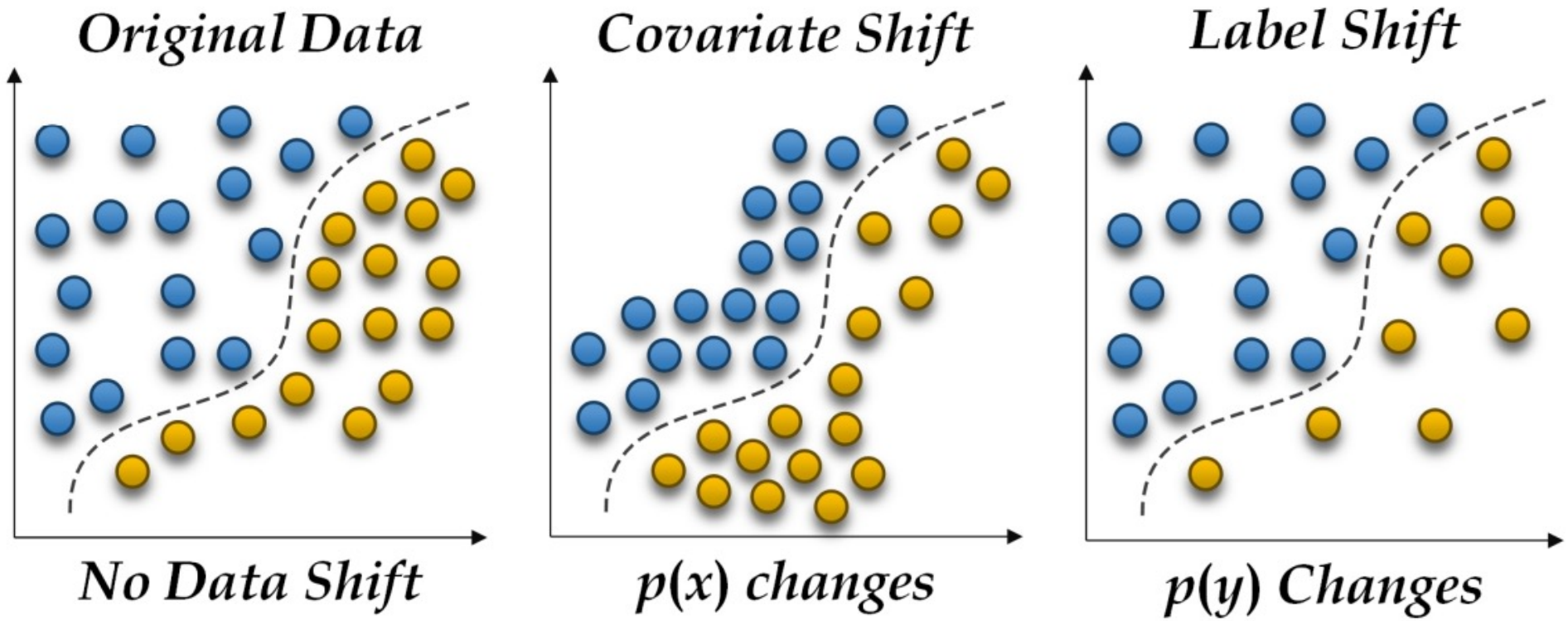
I.I.D.: $p_{\text{train}}(x, y) = p_{\text{test}}(x, y)$ O.O.D.: $p_{\text{train}}(x, y) \neq p_{\text{test}}(x, y)$

There are different forms of out-of-distribution / distribution shift:

- covariate shift: distribution of $p(x)$ changes while $p(y \mid x)$ remains constant
- label shift: distribution of labels $p(y)$ changes while $p(x \mid y)$ remains constant

Label shift happens when some concepts are undersampled or oversampled in the target domain compared to the source domain.

Label Shift



(a) Original data

(b) Covariate shift

covariate shift represents a shift in the input distribution

(c) Label shift

label shift is a shift in the output distribution

Sources of Uncertainty

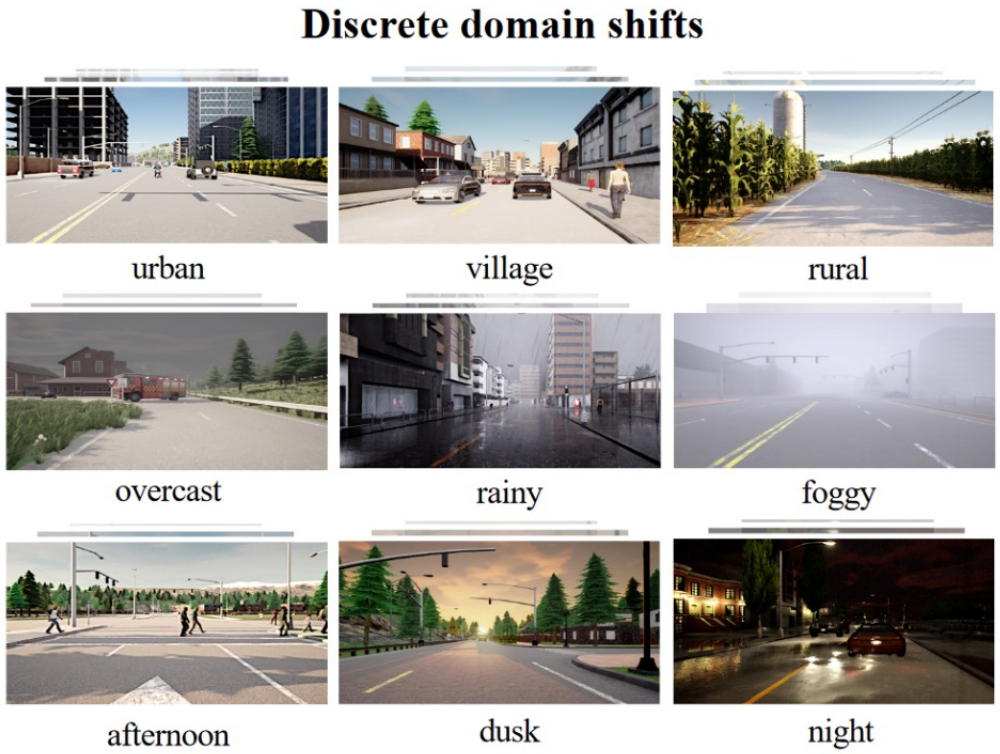
Independent and identically distributed (IID) assumption:
training and testing data are drawn from the same distribution

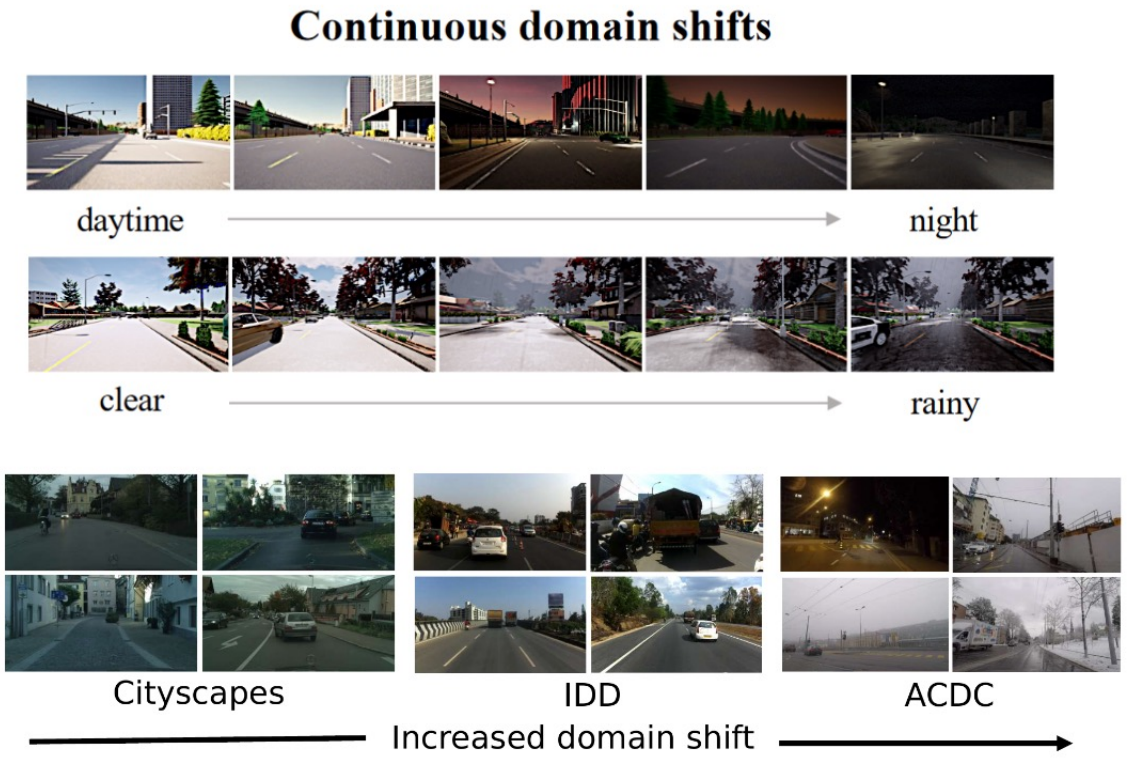
I.I.D.: $p_{\text{train}}(x, y) = p_{\text{test}}(x, y)$ O.O.D.: $p_{\text{train}}(x, y) \neq p_{\text{test}}(x, y)$

There are different forms of out-of-distribution / distribution shift:

- covariate shift: distribution of $p(x)$ changes while $p(y \mid x)$ remains constant
- label shift: distribution of labels $p(y)$ changes while $p(x \mid y)$ remains constant
- OOD or anomaly: new object classes appear at test time

Domain Shift





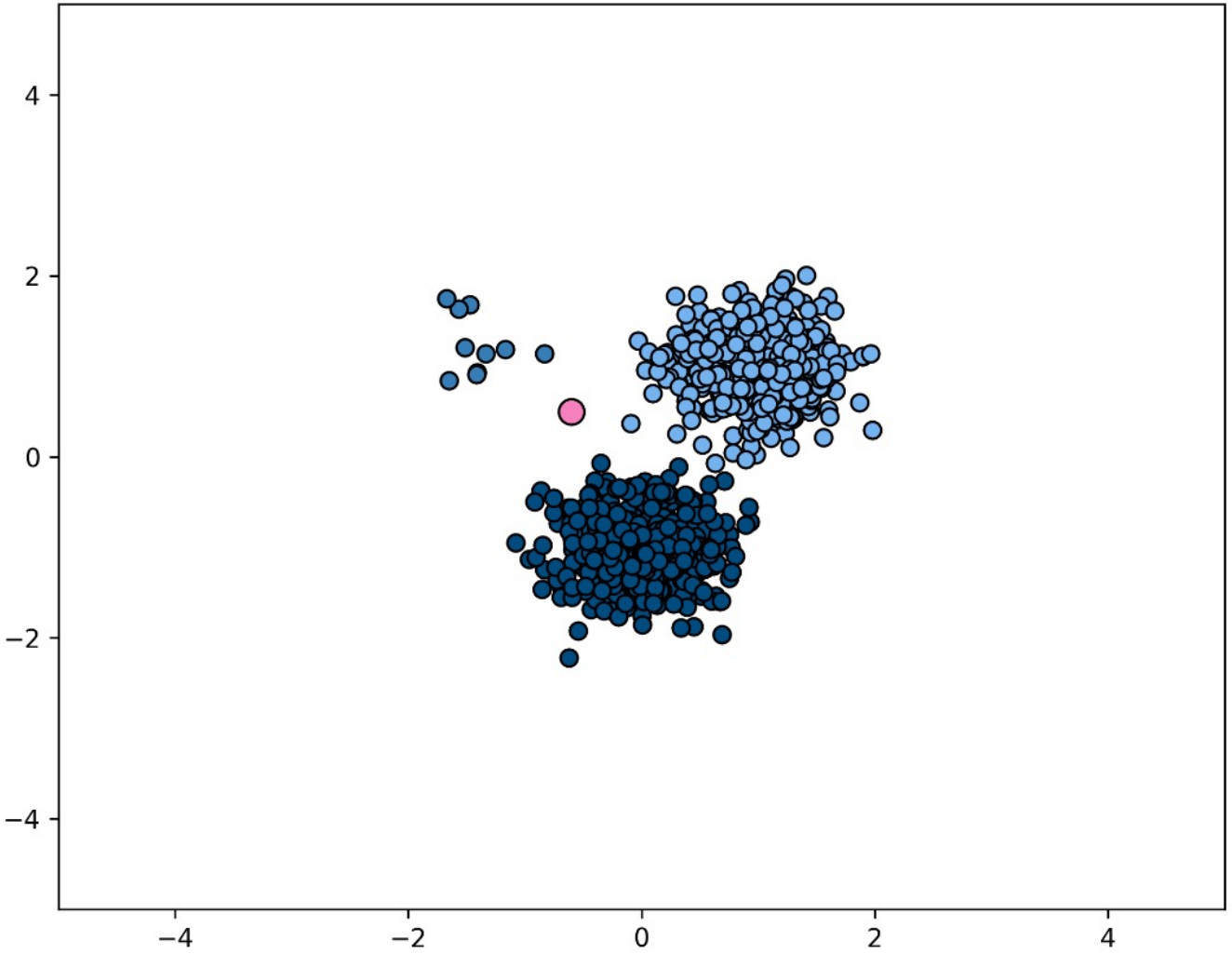
Distribution shift of varying degrees is often encountered in real world conditions

Object-level Shift

Row 1: Lost&Found; Row 2: SegmentMeIfYoucan; Row 3: BRAVO synthetic objects

Reference#1: P. Pinggera et al., Lost and Found: Detecting Small Road Hazards for Self-Driving Vehicles, IROS 2016
Referene#2: R. Chan et al., SegmentMeIfYouCan: A Benchmark for Anomaly Segmentation, NeurIPS Datasets and Benchmarks 2023
Reference#3: BRAVO Challenge 2023, https://valeoai.github.io/bravo/#challenge

Data scarcity



Also causing Epistemic / Knowledge uncertainty

Data scarcity

Train samples

Test samples: unseen variations of seen classes

Knowledge uncertainty




- Knowledge uncertainty is caused by the lack of knowledge about the process that generated the data.
- It can be reduced with additional and sufficient training data (reducible uncertainty)
- In layman words data uncertainty is called the: unknown unknown

Knowing source of uncertainty

Knowing which source of uncertainty predominates can be useful for:

- active learning, reinforcement learning (knowledge uncertainty)
- new data acquisition (knowledge uncertainty)
- decide to fall-back to a complementary sensor, e.g., Lidar, radar, turning on or increasing beam intensity of headlamp, etc. (data uncertainty)
- switch model to output multiple predictions for the same sample or zoom-in on tricky image areas (data uncertainty)
- failure detection (total uncertainty)

Knowing source of uncertainty

The described data and knowledge uncertainty sources are idealized:

- In practice, real data have both uncertainties intermingled and accumulating in total uncertainty.
- Similarly models do not always satisfy conditions for data uncertainty estimation, e.g., over-confidence.

Separating sources of uncertainty typically requires Bayesian approaches.

Preliminaries

- We consider a training dataset $\mathcal{D} = (x, y)$ with N samples and labels
- Most models find a single set or parameters to maximize the probability on conditioned data

$$\begin{aligned}\theta^* &= \arg\max_\theta p(\theta \mid x, y) \\ &= \arg\max_\theta \sum_{x,y \in \mathcal{D}} \log p(y \mid x, \theta) + \log p(\theta) \\ &= \arg\min_\theta - \sum_{x,y \in \mathcal{D}} \log p(y \mid x, \theta) - \log p(\theta)\end{aligned}$$

- The probabilistic approach: estimate a full distribution for $p(\theta \mid x, y)$
- The ensembling approach: find multiple good sets of of parameters θ^*

Ensembles preliminaries

- We view our network as a probabilistic model with $P(y = c \mid x_{test}, \theta)$
- The model posterior $p(\theta \mid \mathcal{D})$ captures the uncertainty in θ and we compute it during training:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})}$$

- From θ we can sample an ensemble of models:

$$\{P(y \mid x_{test}, \theta_m)\}_{m=1}^{M}, \theta_m \sim p(\theta \mid \mathcal{D})$$

- For prediction we use Bayesian inference

$$P(y \mid x_{test}, \mathcal{D}) = \mathbb{E}_{p(\theta|\mathcal{D})}[P(y \mid x_{test}, \theta)] \approx \frac{1}{M} \sum_{m=1}^{M} P(y \mid x_{test}, \theta_m), \theta_m \sim p(\theta \mid \mathcal{D})$$

Total uncertainty

- The total uncertainty is the combination of data uncertainty and knowledge uncertainty
- We can compute it from the entropy of predictive posterior:

$$\mathcal{H}[P(y \mid x_{test}, \mathcal{D})] = \mathcal{H}[\mathbb{E}_{p(\theta|\mathcal{D})}[P(y \mid x_{test}, \theta)]]$$

$$\approx \mathcal{H}[\frac{1}{M}\sum_{m=1}^{M} P(y \mid x_{test}, \theta_m)], \theta_m \sim p(\theta \mid \mathcal{D})$$

A. Mobiny et al., DropConnect Is Effective in Modeling Uncertainty of Bayesian Deep Networks, Nature Scientific Reports 2021
A. Malinin, Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment, PhD Thesis 2019
S. Depeweg et al., Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning, JMLR 2018

Data uncertainty

- Under certain conditions and assumptions (sufficient capacity, training iterations and training data), models with probabilistic outputs capture estimates of data uncertainty
- The expected data uncertainty is:

$$\mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y \mid x_{test}, \theta)]] \approx \frac{1}{M} \sum_{m=1}^{M} \mathcal{H}[P(y \mid x_{test}, \theta_m)], \theta_m \sim p(\theta \mid \mathcal{D})$$

- Each model $P(y \mid x_{test}, \theta_m)$ captures an estimate of the data uncertainty

A. Mobiny et al., DropConnect Is Effective in Modeling Uncertainty of Bayesian Deep Networks, Nature Scientific Reports 2021
A. Malinin, Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment, PhD Thesis 2019
S. Depeweg et al., Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning, JMLR 2018

Knowledge uncertainty

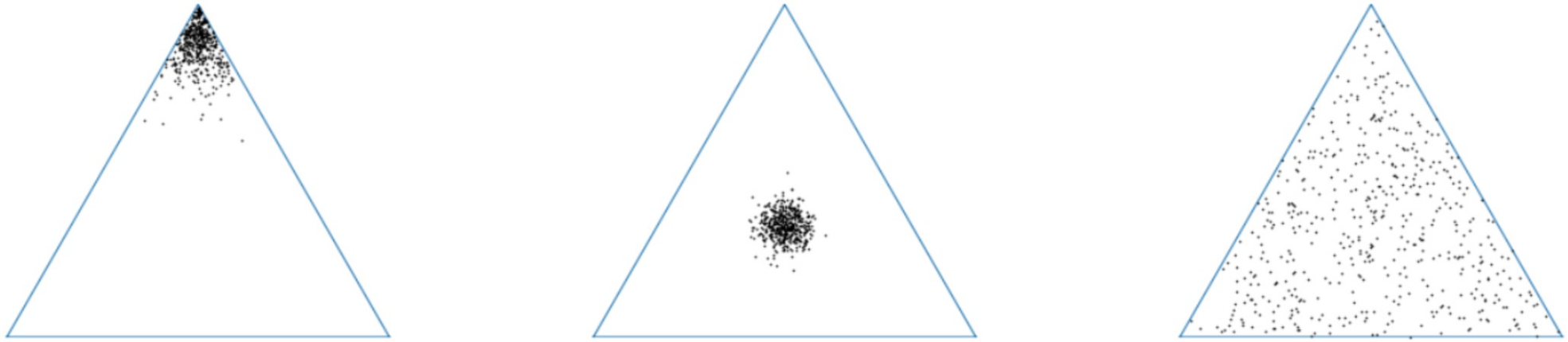
- We can obtain a measure of knowledge uncertainty from the difference of the total uncertainty and data uncertainty, i.e., the mutual information:

$$\underbrace{\mathcal{MI}[y, \theta \mid x_{test}, \mathcal{D}]}_{\text{knowledge uncertainty}} = \underbrace{\mathcal{H}[P(y \mid x_{test}, \mathcal{D})]}_{\text{total uncertainty}} - \underbrace{\mathbb{E}_{p(\theta|\mathcal{D})}[\mathcal{H}[P(y \mid x_{test}, \theta)]]}_{\text{data uncertainty}}$$

- Intuitively, knowledge uncertainty captures the amount of information about the model parameters θ that would be gained through knowledge of the true outcome y
- Mutual Information is essentially a measure of the diversity of the ensemble

A. Mobiny et al., DropConnect Is Effective in Modeling Uncertainty of Bayesian Deep Networks, Nature Scientific Reports 2021
A. Malinin, Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment, PhD Thesis 2019
S. Depeweg et al., Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning, JMLR 2018

Ensemble predictions on a simplex



a) Low uncertainty

b) Data uncertainty

c) Knowledge uncertainty

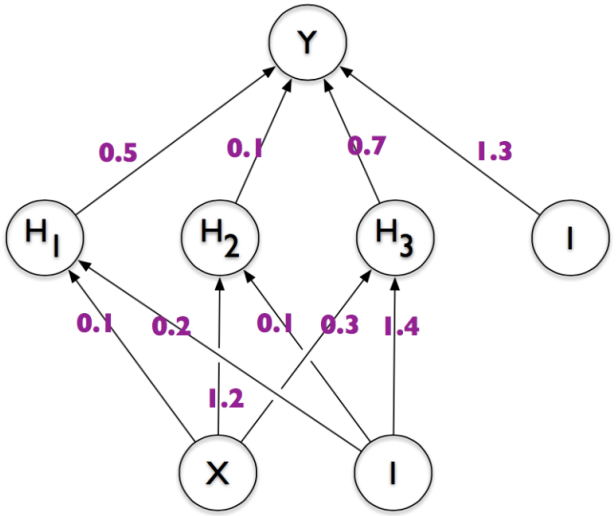
- 3-way classification task
- each point represents the picture of ensemble member

A. Malinin, Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment, PhD Thesis 2019

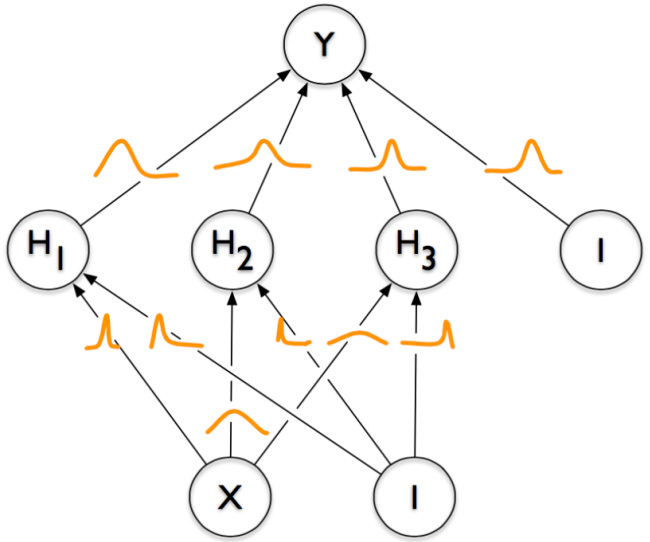
Ensemble Approaches

Bayesian Neural Networks (BNNs)

Standard Neural Network



Bayesian Neural Network



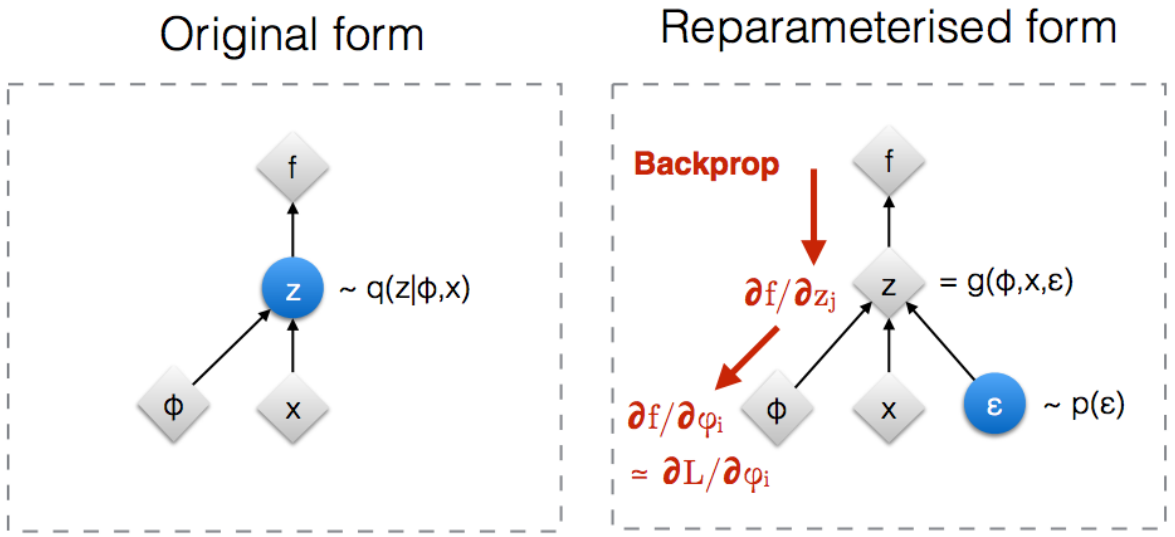
- Parameters represented by single, fixed values (point estimates)
- Conventional approaches to training NNs can be interpreted as approximations to the full Bayesian method (equivalent to MLE or MAP estimation)

- Parameters represented by distributions
- For Gaussian priors: each parameter consists of a pair (μ, σ) describing a distribution over it (2x more parameters)

C. Blundell et al., Weight Uncertainty in Neural Networks, ICML 2015

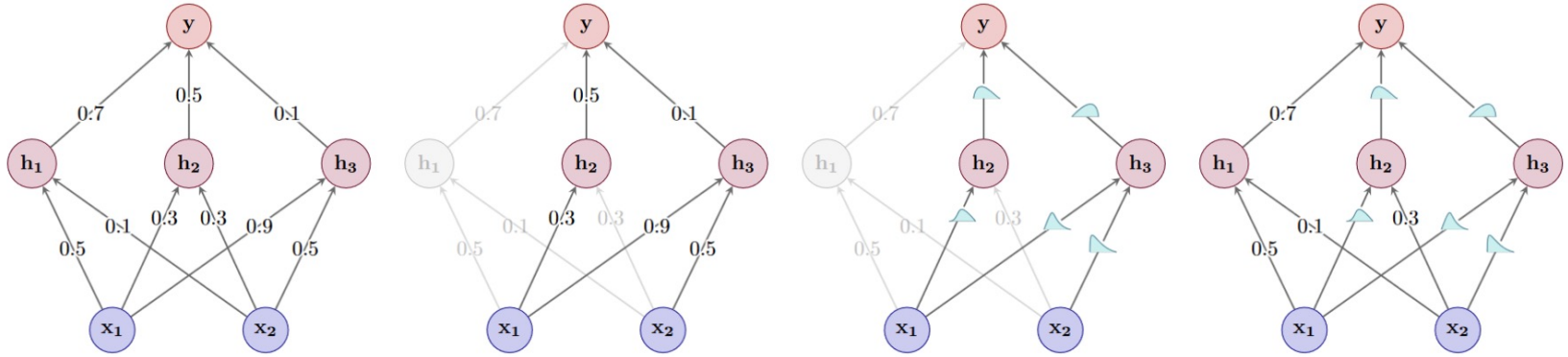
Bayesian Neural Networks (BNNs)

- Bayesian Neural Networks (BNNs) are easy to formulate, but notoriously difficult to perform inference in.
- Modern BNNs are trained with variational inference (reparameterization trick) assuming parameter independence.



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Bayesian Neural Networks (BNNs)



Subnetwork inference

Recent methods consider only the last layer as variational (Laplace Redux) or a part of parameters across layers as variational with some parameter inter-dependence (LP-BNN, Subnetwork Inference).

E. Daxberger et al., Laplace Redux -- Effortless Bayesian Deep Learning, NeurIPS 2021
G. Franchi et al., Encoding the latent posterior of Bayesian Neural Networks for uncertainty quantification, arXiv 2020
E. Daxberger et al., Bayesian Deep Learning via Subnetwork Inference, ICML 2021

Deep Ensembles

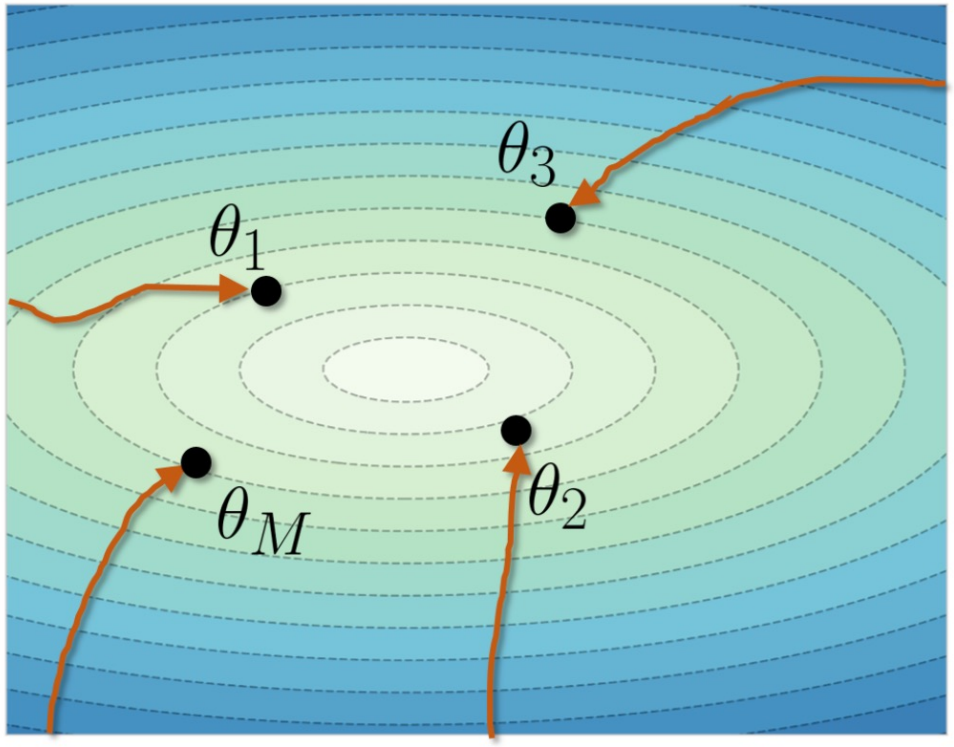
Idea: Run regular SGD training with different random seeds. At test time, average predictions.

Pros

- extremely simple
- top performance across benchmarks
- agnostic to the underlying architecture

Cons

- computational cost grows linearly both for training and testing



B. Lakhsminarayanan et al., Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, NeurIPS 2017

Deep Ensembles

Deep Ensembles have long been SOTA across tasks and benchmarks in terms of predictive performance and uncertainty estimation (diversity).

While they are used heavily for offline computation (e.g., pseudo-labeling, active learning, etc.), their computational cost is prohibitive for real-time decision systems.

They have inspired numerous subsequent works aiming to improve computational cost (memory, runtime) during training and/or testing, size and number of networks needed, etc., while preserving as many of the desirable properties of ensembles.

A quick tour of deep ensemble variants

Ensembles from 1 single training run

Snapshot Ensembles

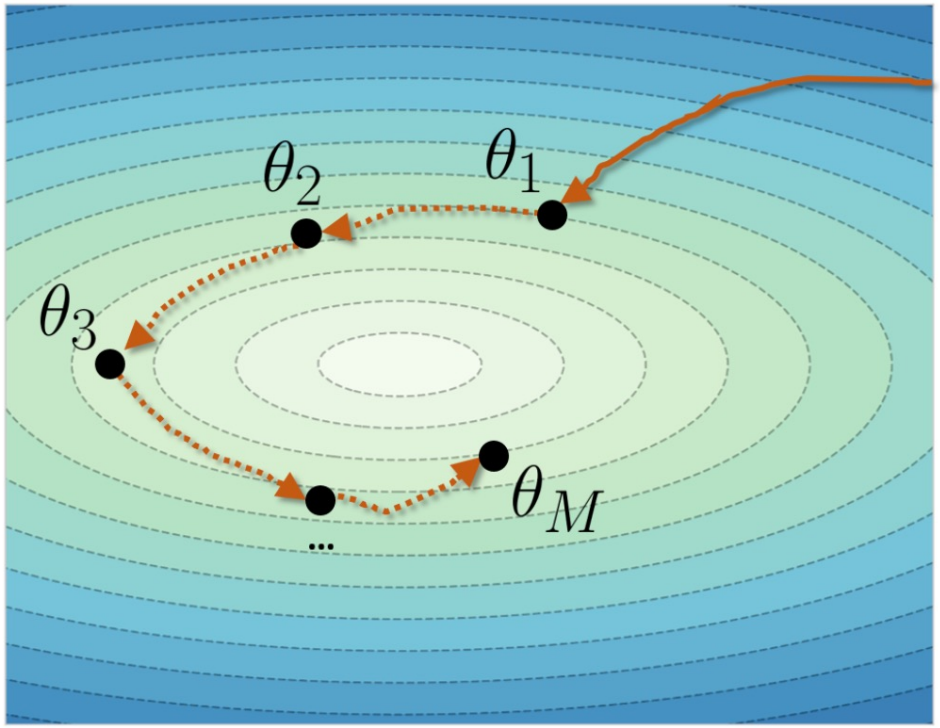
Idea: Collect checkpoints from cyclic learning rate cycles and use them as an ensemble

Pros

- relatively simple to setup
- good predictive performance
- low computational cost at training

Cons

- still an ensemble at test time: multiple networks, multiple forwards
- limited diversity in the predictions
- some instability if checkpoints are sampled from early training steps



G. Huang et al., Snapshot Ensembles: Train 1, get M for free, ICLR 2017

Snapshot Ensembles

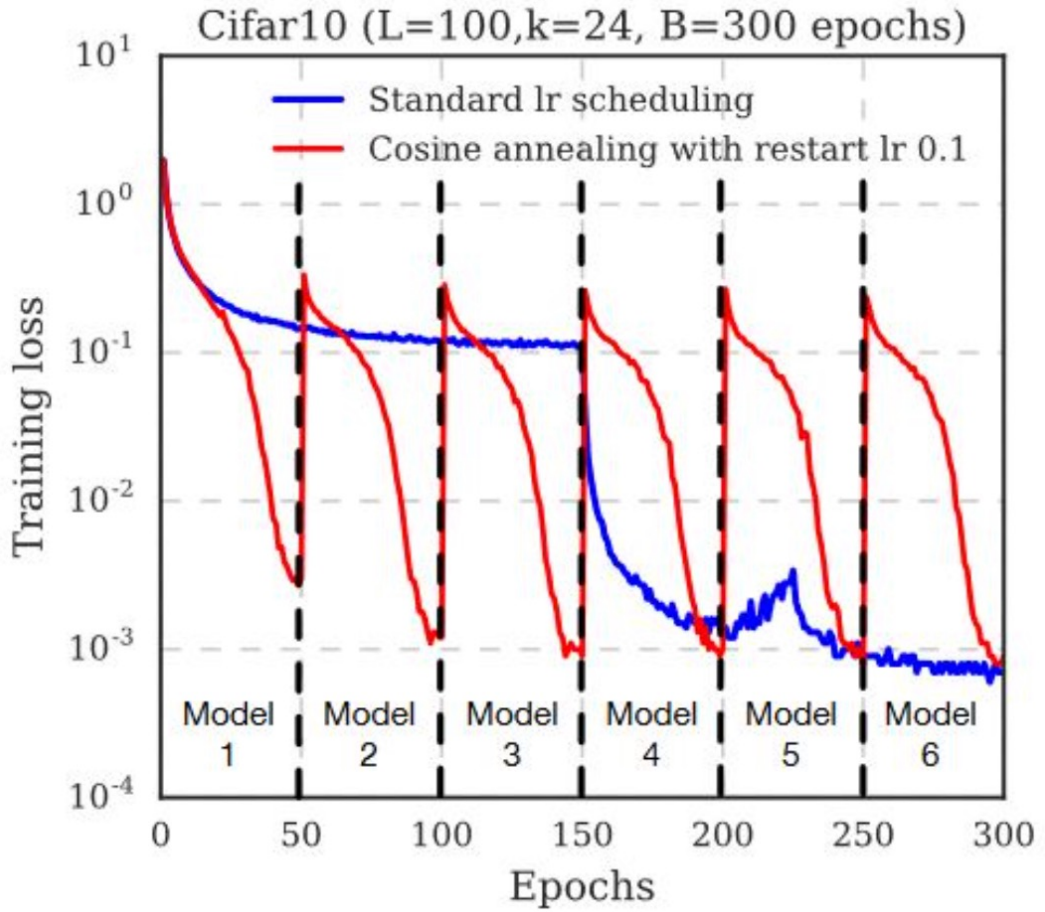
Idea: Collect checkpoints from cyclic learning rate cycles and use them as an ensemble

Pros

- relatively simple to setup
- good predictive performance
- low computational cost at training

Cons

- still an ensemble at test time: multiple networks, multiple forwards
- limited diversity in the predictions
- some instability if checkpoints are sampled from early training steps



G. Huang et al., Snapshot Ensembles: Train 1, get M for free, ICLR 2017

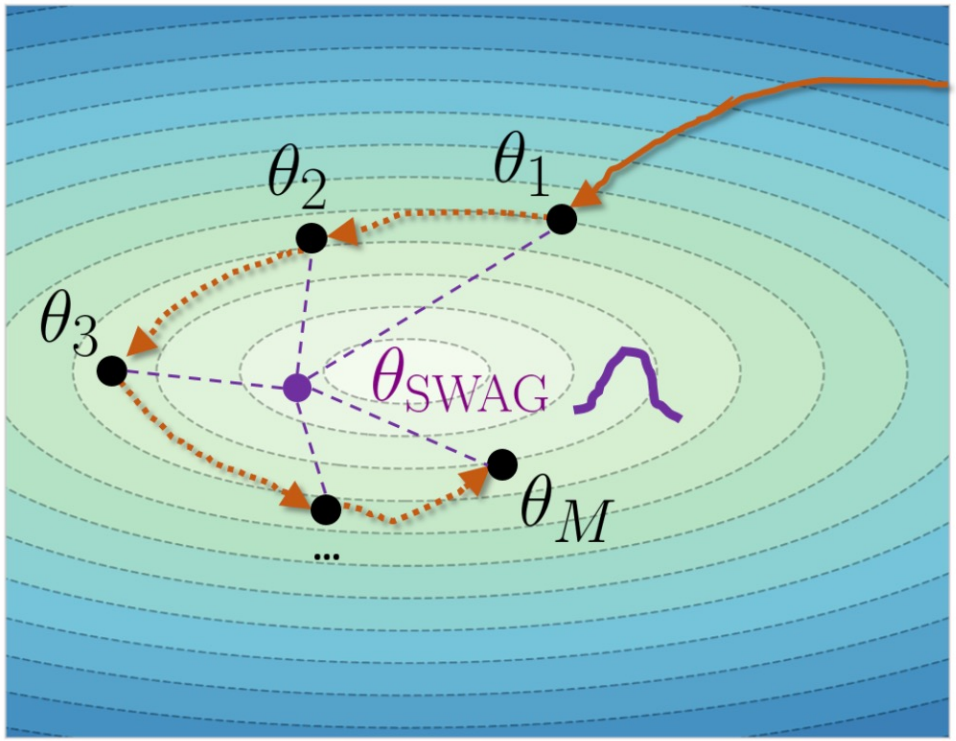
A quick tour of deep ensemble variants

BNNs from 1 single training run

SWAG

Idea: Use checkpoints to estimate a distribution over the network parameters.

- training: collect checkpoints at end of cyclic LR cycles.
- test time: sample ensemble from final distribution and run multiple forwards.



W. Maddox et al., A Simple Baseline for Bayesian Uncertainty in Deep Learning, NeurIPS 2019

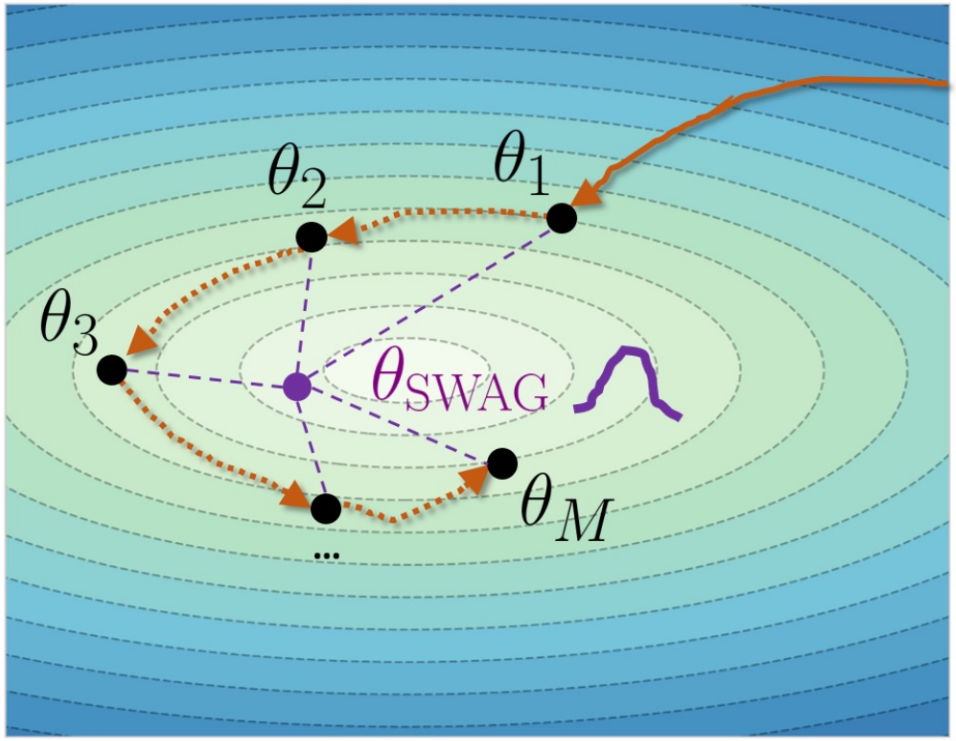
SWAG

Pros

- relatively simple to setup
- good predictive performance and faster training

Cons

- needs many epochs to explore checkpoints
- still an ensemble at test time: multiple networks, multiple forwards
- limited diversity in the predictions

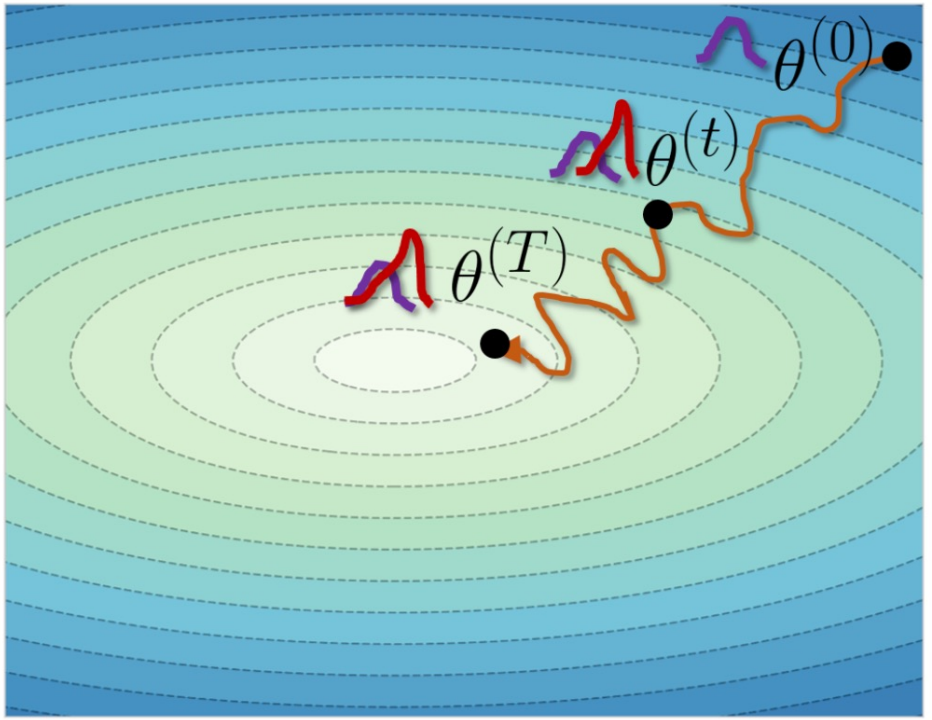


W. Maddox et al., A Simple Baseline for Bayesian Uncertainty in Deep Learning, NeurIPS 2019

TRADI

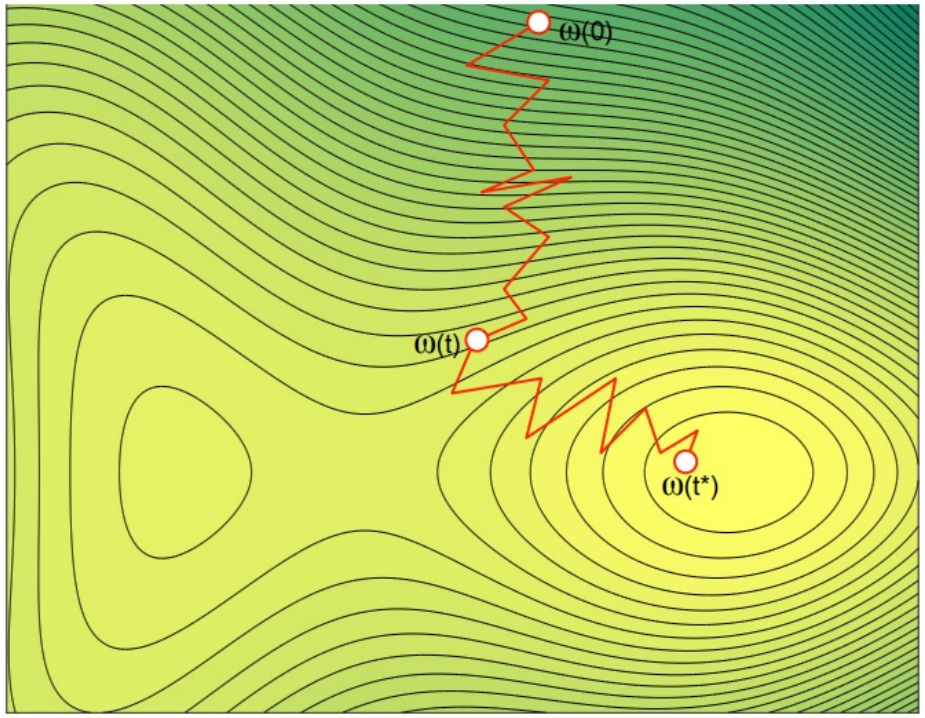
Idea: Track parameters over their optimization trajectory from initialization towards minimum.

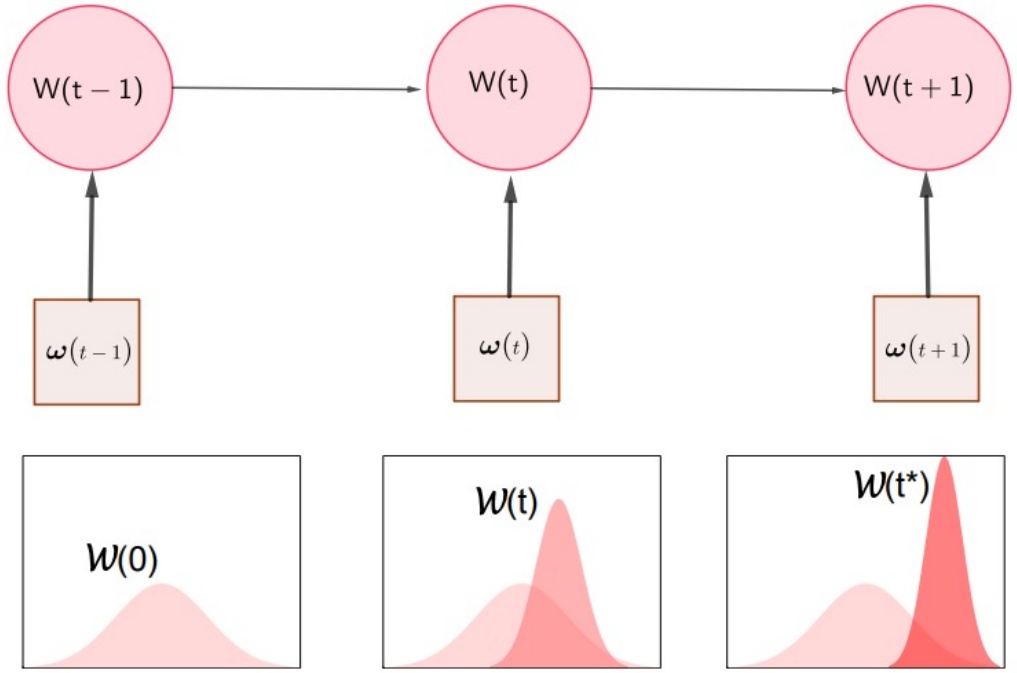
- training: use Kalman filter to compute per-parameter distribution.
- test time: sample ensemble from final distribution and run multiple forwards.



G. Franchi et al.,TRADI: Tracking deep neural network weight distributions for uncertainty estimation, ECCV 2020

TRADI





The idea is to track the distribution W of all DNN weights across training steps from a generic prior W(0) to the final estimate W(t*).

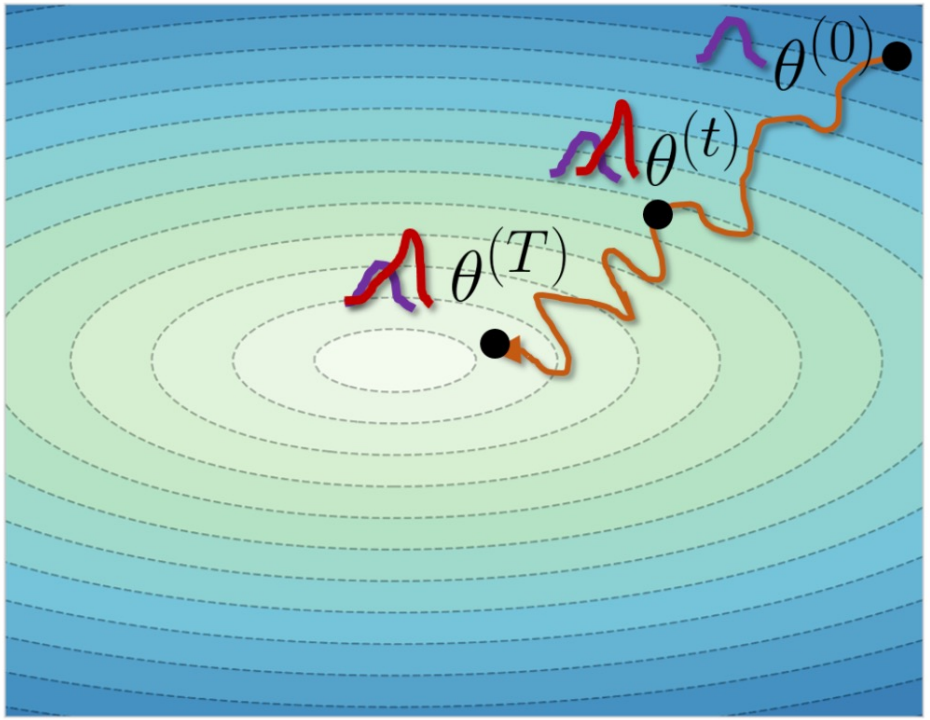
TRADI

Pros

- good predictive performance
- very low computational overhead and impact on training

Cons

- still an ensemble at test time: multiple networks, multiple forwards
- limited diversity in the predictions



G. Franchi et al.,TRADI: Tracking deep neural network weight distributions for uncertainty estimation, ECCV 2020

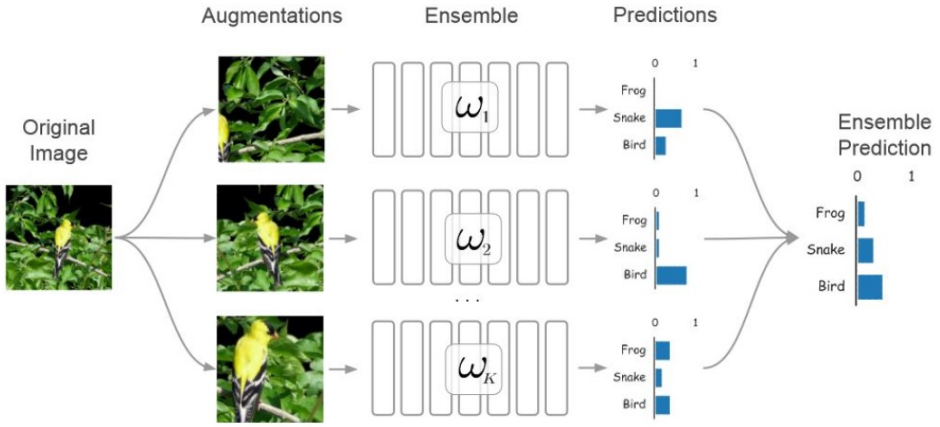
A quick tour of deep ensemble variants

Multiple forwards over 1 network

Test-time augmentation

Idea: Leverage data augmentations at test time to mimic an ensemble

- training: no change
- test: apply different data augmentations on the input and run multiple forwards



A. Ashukha et al., Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning, ICLR 2020

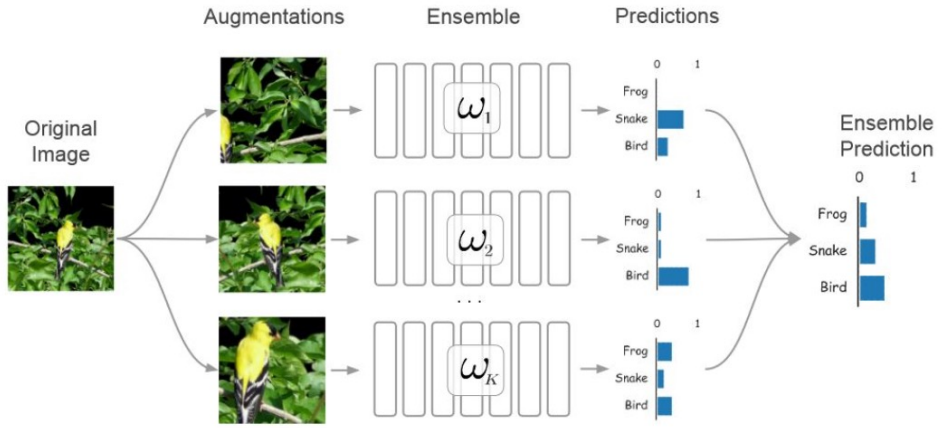
Test-time augmentation

Pros

- very simple
- boosts predictive performance
- 1 network at test time

Cons

- multiple forwards at train time
- limited diversity in the predictions

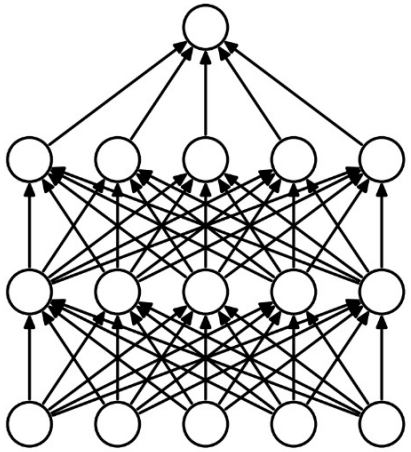


A. Ashukha et al., Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning, ICLR 2020

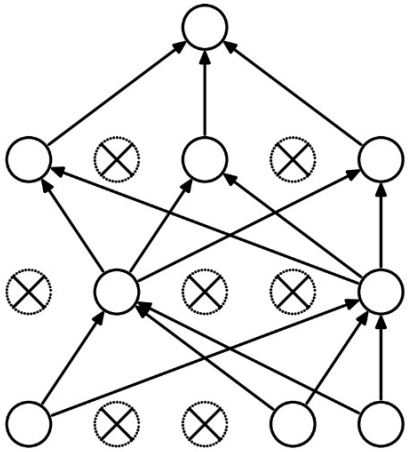
MC-Dropout

Idea: Use Dropout during training: multiple subnetwork configurations.

- training: Dropout layers active
- test: Keep Dropout active and forward with multiple Dropout masks



(a) Standard Neural Net



(b) After applying dropout.

Y. Gal & Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, ICML 2016

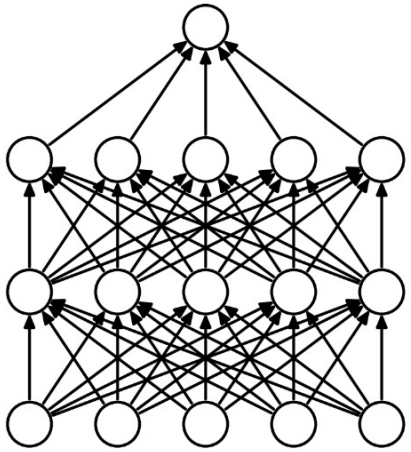
MC-Dropout

Pros

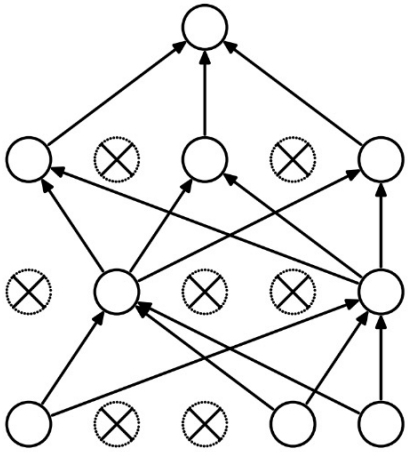
- simple to train
- good predictive performance
- 1 network at test time

Cons

- multiple forwards at train time
- limited diversity in the predictions



(a) Standard Neural Net



(b) After applying dropout.

Y. Gal & Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, ICML 2016

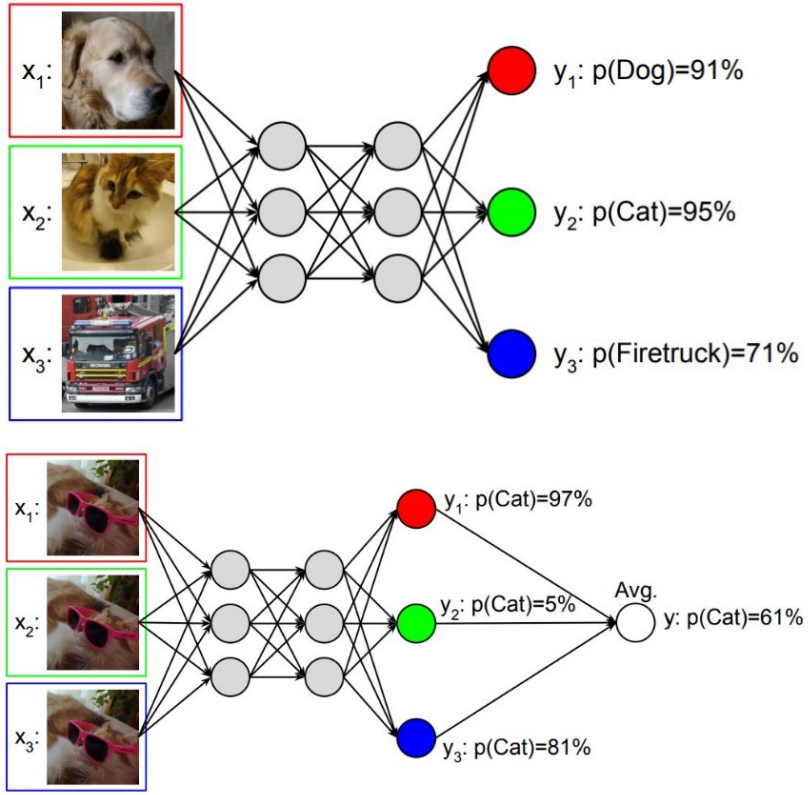
A quick tour of deep ensemble variants

Ensemble-like architectures with 1 forward

MIMO (Multi-Input Multi-Output)

Idea: Harness multiple subnetwork paths learned implicitly by large networks

- training: different images forwarded together
- test: forward different augmentations of the same image



M.Havasi et al., Training independent subnetworks for robust prediction, ICLR 2021
S. Lee et al., Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks, arXiv 2015

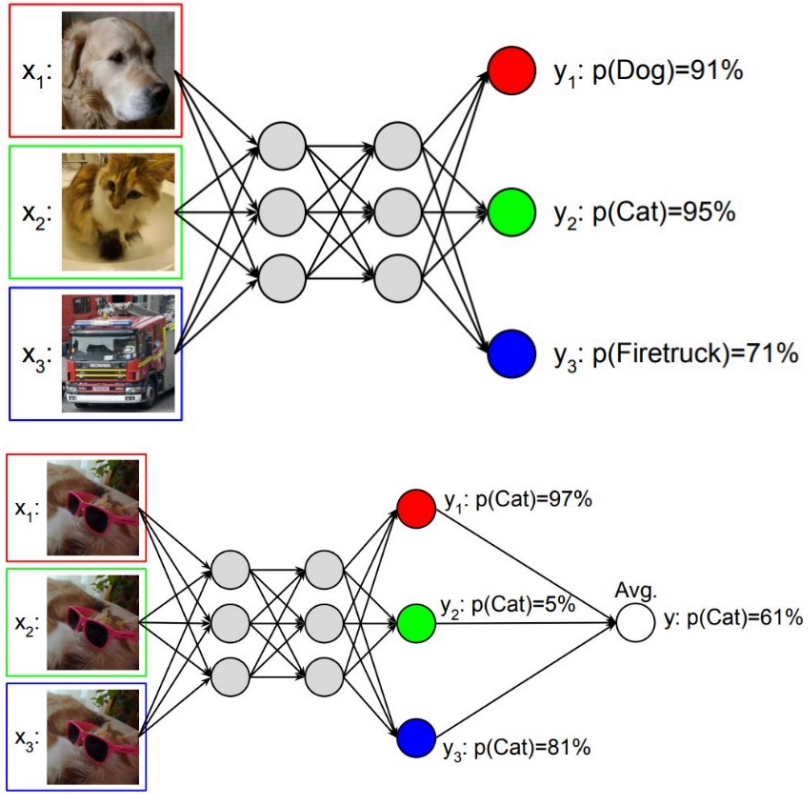
MIMO (Multi-Input Multi-Output)

Pros

- simple to train
- good predictive performance
- minor computational overhead

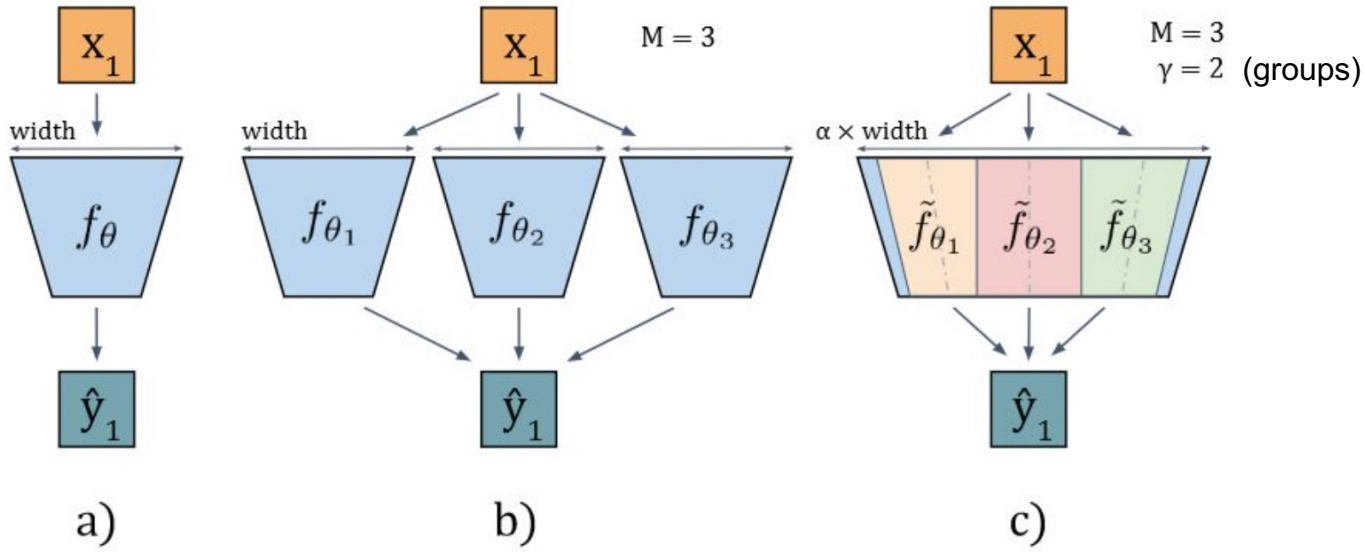
Cons

- limited diversity in the predictions
- subnetworks not independent
- tends to saturate with increasing number of subnetworks



M.Havasi et al., Training independent subnetworks for robust prediction, ICLR 2021
S. Lee et al., Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks, arXiv 2015

Packed-Ensembles

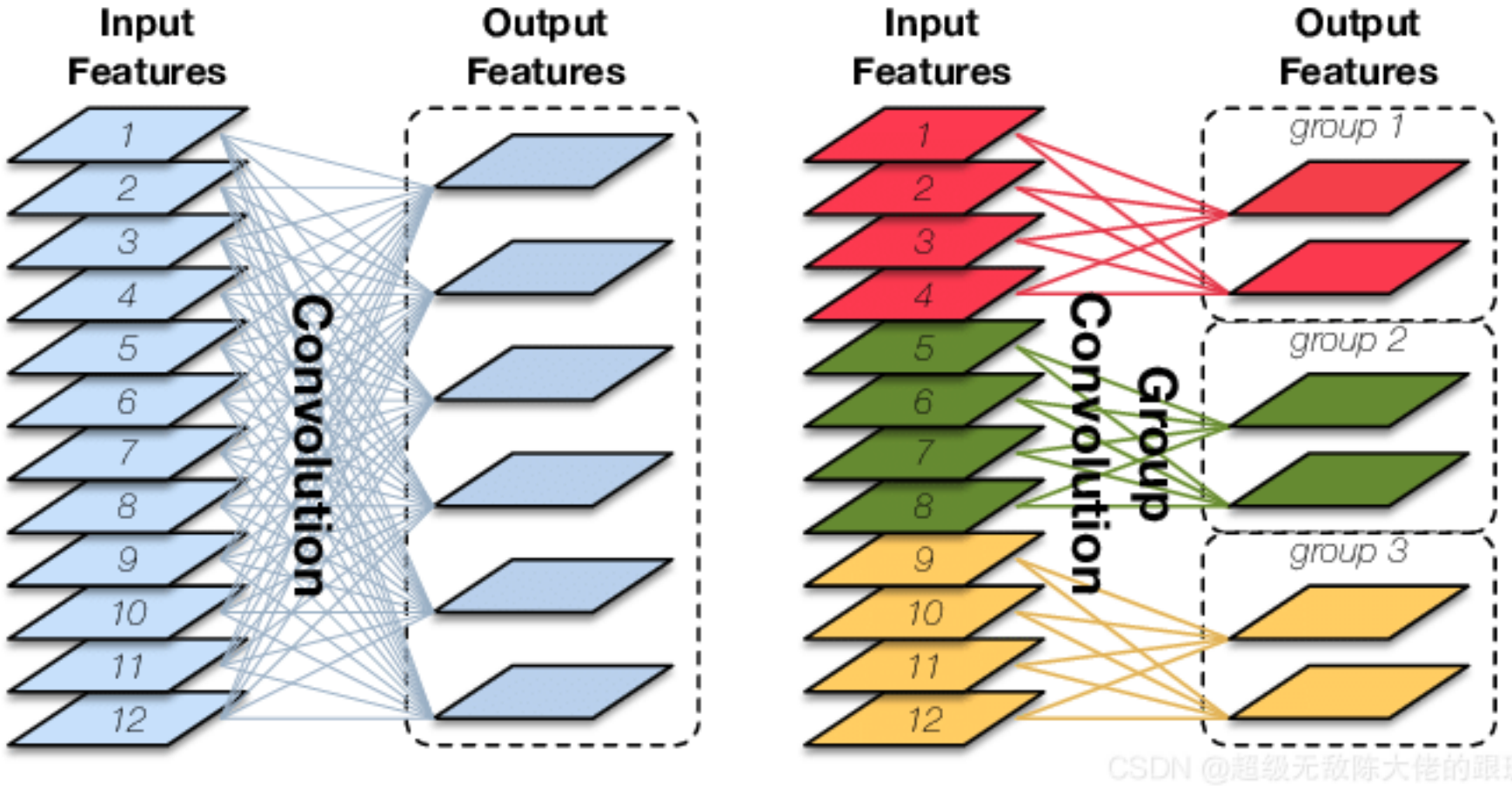


a) Single network; b) Deep Ensembles; c) Packed-Ensembles

Idea: Leverage grouped convolutions to separate independent subnetworks trained together in the envelope of a single larger model.

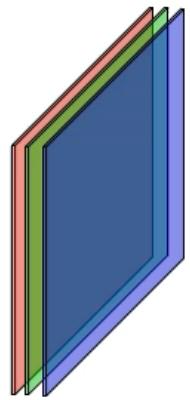
O. Laurent & A. Lafage et al., Packed-Ensembles for Efficient Uncertainty Estimation, ICLR 2023

Grouped Convolution



CSDN @超级无敌陈大佬的眼斑

Packed-Ensembles



Performance comparison

- Image classification

			Classification		Calibration	OOD Detection			Complexity	
Method	**Data**	**Net**	**Acc ↑**	**NLL ↓**	**ECE ↓**	**AUPR ↑**	**AUC ↑**	**FPR95 ↓**	**Params (M) ↓**	**Mult-Adds ↓**
Single Model	C10	R50	95.1	0.211	0.031	95.2	91.9	23.6	23.52	1.30
BatchEnsemble	C10	R50	93.9	0.255	0.033	94.7	91.3	20.1	23.63	5.19
MIMO ($\rho = 1$)	C10	R50	95.4	0.197	0.030	95.1	90.8	26.0	23.59	1.30
Masksembles	C10	R50	95.3	0.175	0.019	95.7	92.2	22.1	23.81	5.19
Packed-Ensembles	C10	R50	95.9	0.137	**0.008**	**97.3**	**95.2**	**14.4**	**14.55**	**1.00**
Deep Ensembles	C10	R50	**96.0**	**0.136**	**0.008**	97.0	94.7	15.5	94.08	5.19
Single Model	C100	R50	78.3	0.905	0.089	87.4	77.9	57.6	23.70	1.30
BatchEnsemble	C100	R50	66.6	1.788	0.182	85.2	74.6	60.6	23.81	5.19
MIMO ($\rho = 1$)	C100	R50	79.0	0.876	0.079	87.5	76.9	64.7	24.33	1.30
Masksembles	C100	R50	78.5	0.832	0.046	90.3	81.9	52.3	23.81	5.19
Packed-Ensembles	C100	R50	**81.2**	**0.703**	**0.020**	**90.0**	**81.7**	56.5	**15.55**	**1.00**
Deep Ensembles	C100	R50	80.9	0.713	0.026	89.2	80.8	**52.5**	94.82	5.19

Performance comparison (averaged over five runs) on CIFAR-10/100 using ResNet-50 (R50) architecture. All ensembles have M = 4 subnetworks, we highlight the best performances in bold.

Performance comparison

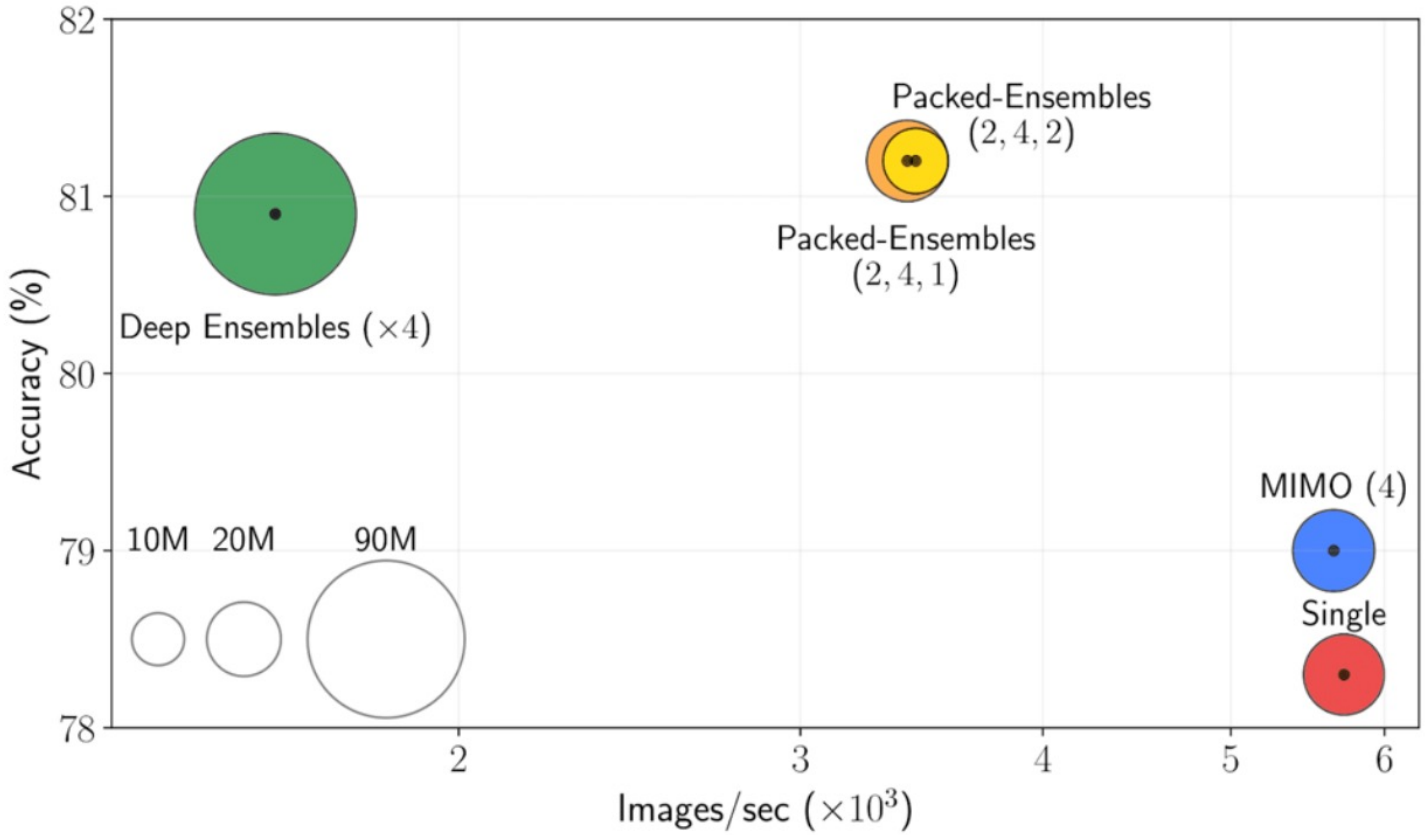
- Image classification

			Classification		Calibration	OOD Detection			Complexity	
Method	**Data**	**Net**	**Acc ↑**	**NLL ↓**	**ECE ↓**	**AUPR ↑**	**AUC ↑**	**FPR95 ↓**	**Params (M) ↓**	**Mult-Adds ↓**
Single Model	C10	R50	95.1	0.211	0.031	95.2	91.9	23.6	23.52	1.30
BatchEnsemble	C10	R50	93.9	0.255	0.033	94.7	91.3	20.1	23.63	5.19
MIMO ($\rho = 1$)	C10	R50	95.4	0.197	0.030	95.1	90.8	26.0	23.59	1.30
Masksembles	C10	R50	95.3	0.175	0.019	95.7	92.2	22.1	23.81	5.19
Packed-Ensembles	C10	R50	95.9	0.137	**0.008**	**97.3**	**95.2**	**14.4**	**14.55**	**1.00**
Deep Ensembles	C10	R50	**96.0**	**0.136**	**0.008**	97.0	94.7	15.5	94.08	5.19
Single Model	C100	R50	78.3	0.905	0.089	87.4	77.9	57.6	23.70	1.30
BatchEnsemble	C100	R50	66.6	1.788	0.182	85.2	74.6	60.6	23.81	5.19
MIMO ($\rho = 1$)	C100	R50	79.0	0.876	0.079	87.5	76.9	64.7	24.33	1.30
Masksembles	C100	R50	78.5	0.832	0.046	90.3	81.9	52.3	23.81	5.19
Packed-Ensembles	C100	R50	**81.2**	**0.703**	**0.020**	**90.0**	**81.7**	56.5	**15.55**	**1.00**
Deep Ensembles	C100	R50	80.9	0.713	0.026	89.2	80.8	**52.5**	94.82	5.19

Performance comparison (averaged over five runs) on CIFAR-10/100 using ResNet-50 (R50) architecture. All ensembles have M = 4 subnetworks, we highlight the best performances in bold.

- Independent ensembles achieve better performance
- Packed Ensembles does so with a lower computational budget

Performance comparison



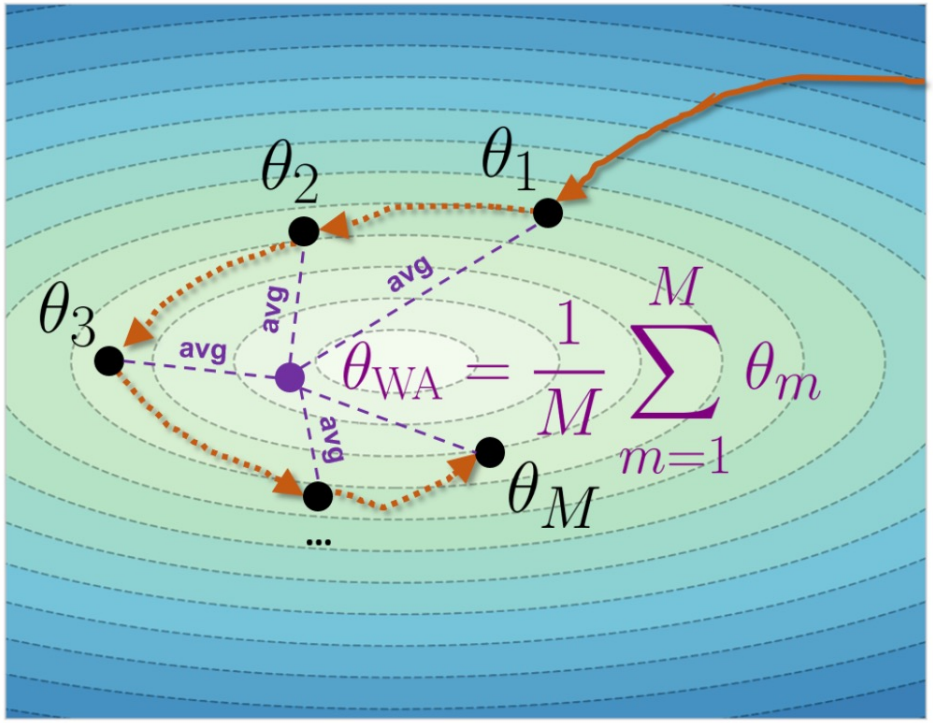
Evaluation of computation cost vs. performance trade-offs for multiple uncertainty quantification techniques on CIFAR-100.

Recent trends

Weight averaging

Idea: Average multiple checkpoints along the SGD trajectory with cyclic learning rate. This leads to better generalization.

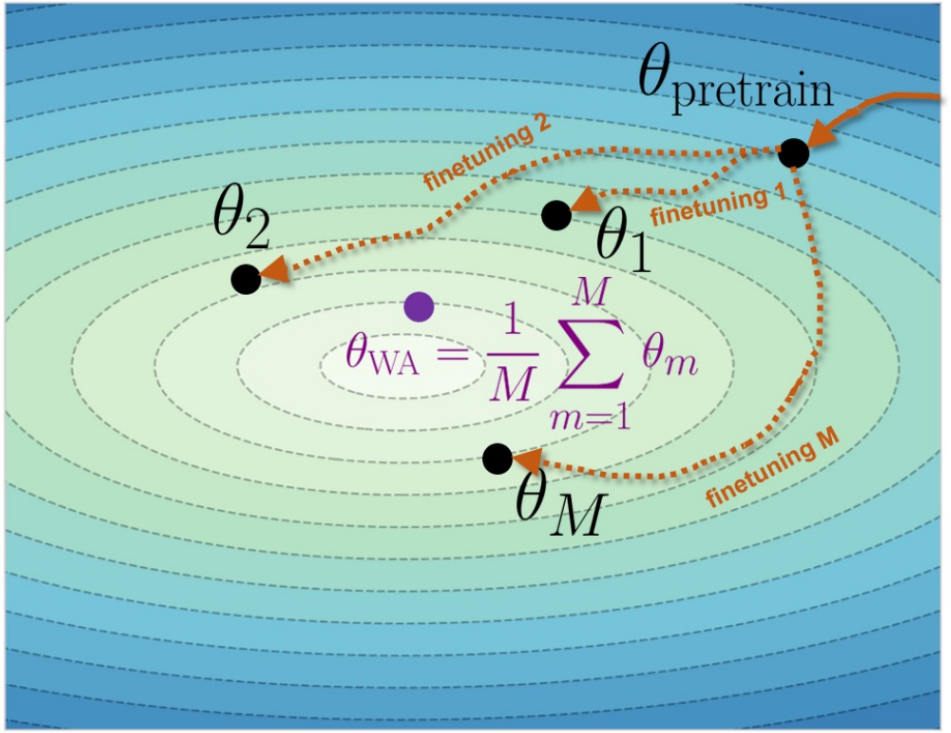
- training: collect checkpoints at end of cyclic LR cycles and then average them
- test time: single forward.



P. Izmailov et al., Averaging Weights Leads to Wider Optima and Better Generalization, UAI 2018

Model soups

- The idea has been revisited in the context of foundation models that offer a good starting point
- Checkpoints are computed from different finetunings: different hyperparams, data order, augmentation
- Requirement: the weights should remain linearly connected
- Boosts performance on distribution shift, but loses in calibration



M. Wortsman et al., Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, ICML 2022
J. Frankle et al., Linear mode connectivity and the lottery ticket hypothesis, ICML 2020
T. Garipov et al., Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs, NeurIPS 2018

Ensembles in vision-language models

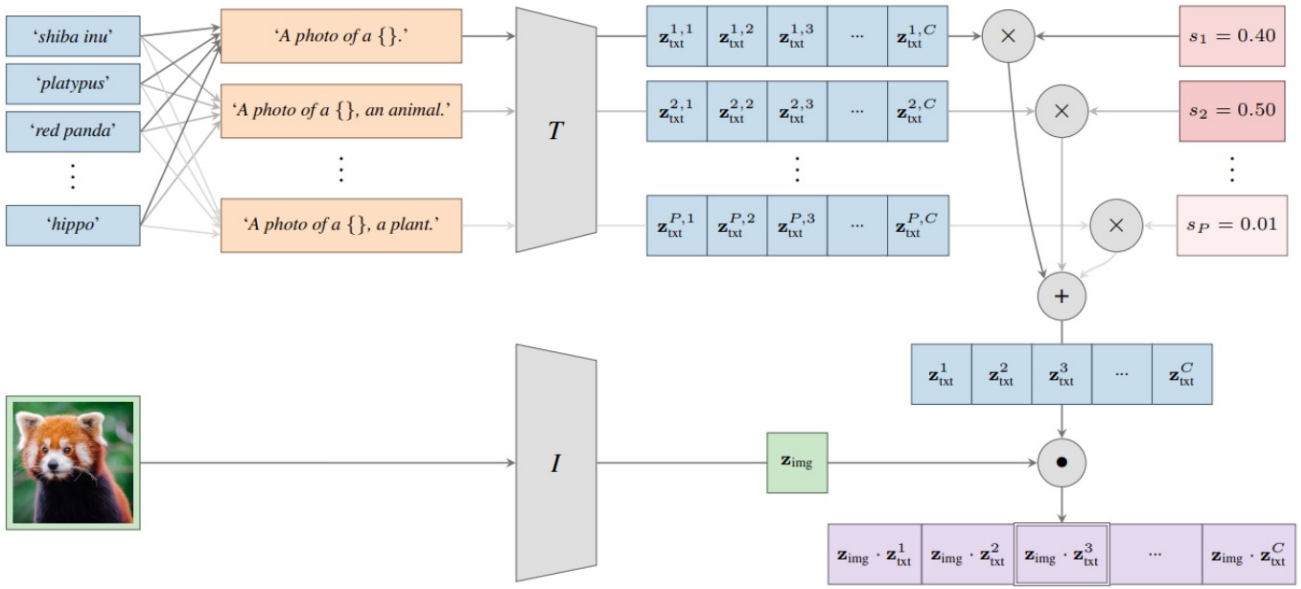


Figure 1. Construction of a zero-shot classifier with zero-shot prompt ensembling (ZPE) for text-image models. Logits (□) are calculated by combining text (□) and image (□) representations. The final text representation is a weighted ensemble of representations corresponding to different prompts (□). Crucially, the ZPE scores (□) for weighting each prompt are calculated without access to any labeled training data, as described in Section 3 and Algorithm 2.

- Emergence of different ensembling strategies on the text-encoder side

J. Urquhart et al., A Simple Zero-shot Prompt Weighting Technique to Improve Prompt Ensembling in Text-Image Models, ICML 2023

A Summary on Uncertainty Estimation and Ensemble Approaches

- Understanding different sources of uncertainty can be useful for different actions and applications
- Ensembles are better equipped to separate sources of uncertainty
- Diversity is a key property for ensemble models
- Many emerging computationally efficient alternatives for ensembles mimic ensemble properties. Contenders are often below or specific to a type of architecture
- Vision-language models: first text-based ensembles

This Lecture – Agenda

- **Reliability of Deep Learning for Real-World Deployment**
  - Uncertainty Estimation
    - Data Uncertainty
    - Knowledge Uncertainty
    - Ensemble Approaches
  - Adaptation to Unseen Concepts on the Fly
    - Dynamic Adaptation: Test-time Adaptation
    - Static Adaptation: Vision-Language Learning

Adaptation to Unseen Concepts: A Motivating Example

We have a robot that has been trained to identify various objects in a garden.

But what happens if:

- The *season changes*?
- *New items* appear in the garden?
- *New instances or compositions* of items appear? (e.g., yellow tomatoes instead of red tomatoes)
- …and more…

Why Adaptation?

- **Traditional Machine Learning Assumption:**
  - The training set contains all information needed to perform the target task
  - That is, the **training distribution** is the **comfort zone**

- **In practice, this assumption is often violated:**
  - It is difficult to create a training set capturing **all possible semantic concepts** and **their compositions**
  - The world is **continuously evolving and changing** ⇒ new concepts emerge

Why must we go beyond the Comfort Zone?

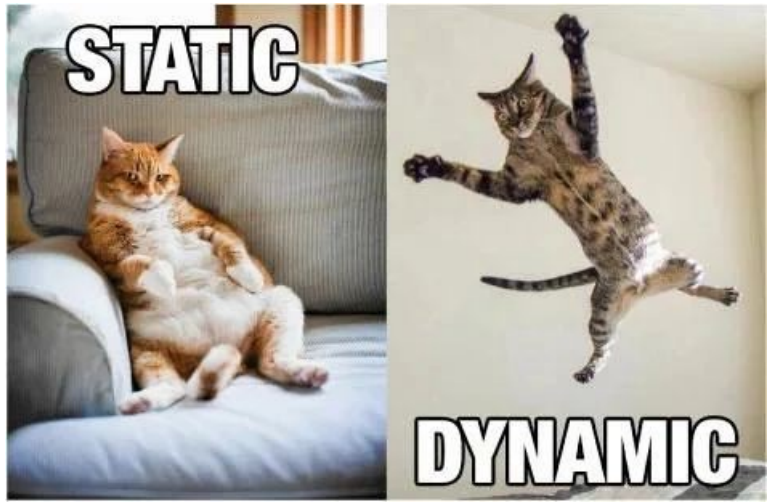
- **Data collection is expensive**
- **Some data sub-populations are underrepresented in training sets**
- **Adaptation provides an alternative to re-training from scratch**
  - Often infeasible due to loss of data (e.g., due to **privacy concerns**)
- **Safety-critical applications require models that generalize beyond their training distribution (e.g., self-driving cars, medical applications)**

Do Foundation Models Solve the Problem?

- No!
- Foundation models (Vision, LLMs, VL) *still suffer from distribution shift issues*
- Example: CLIP (a V+L zero-shot model) does not perform well in certain domains such as satellite images

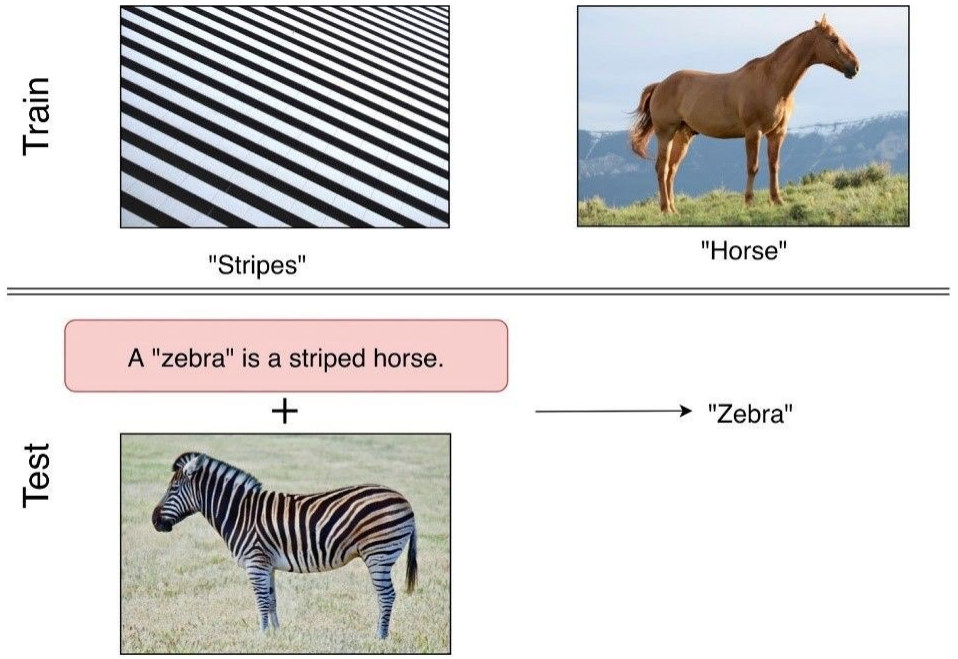
- Not all modes of the full distribution can be captured with in-the-wild data!

So what do we do?



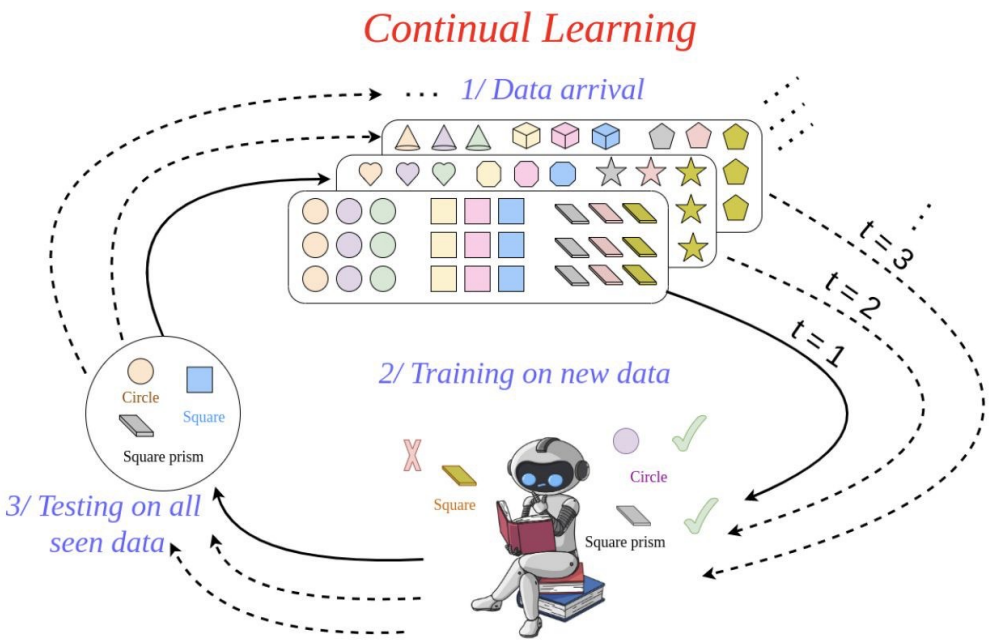
Static Adaptation

- **Prepare the model in advance** for unavailable semantic concepts
- **Examples:** zero-shot learning or transfer



Dynamic Adaptation

- **Adapt the model on the fly** – exploit stream of incoming data at deployment
- **Examples:** continual learning, open-world learning, or test-time training



The Space of Problems and Methods

- What has changed? **Prior, covariate, or semantic distribution shift**

$$P_{tr}(Y|X) = P_{tst}(Y|X) \quad \text{and } P_{tr}(X) \neq P_{tst}(X)$$

$$P_{tr}(Y|X) \neq P_{tst}(Y|X) \quad \text{and } P_{tr}(X) = P_{tst}(X)$$

- What data is available? **Supervised or unsupervised/transductive**
- What priors are assumed? **Compositionality, etc.**
- How are they used? **Model adaptation (dynamic) or composition/ inference-only (static)**

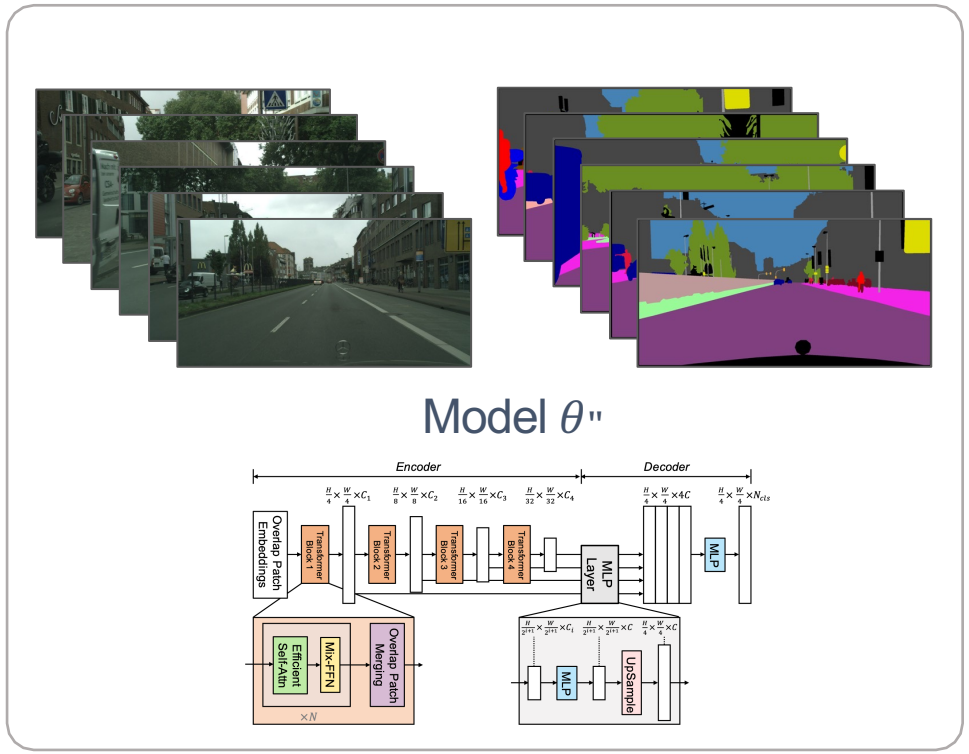
Some Adaptation Scenarios

- Test Time Adaptation - **Covariate shift, unsupervised, dynamic**
- Lifelong/Active Learning - **Semantic shift, supervised, dynamic**
- Composition Zero-Shot - **Semantic shift, unsupervised w/ composition, static**
- Vision-Language Learning - **Semantic shift, supervised (paired image-text data), static**

Learning i.i.d.

- **Domain shift**: the image distribution shift wrt train time ($P_X^{\text{train}} \neq P_X^{\text{test}}$)

Images Masks

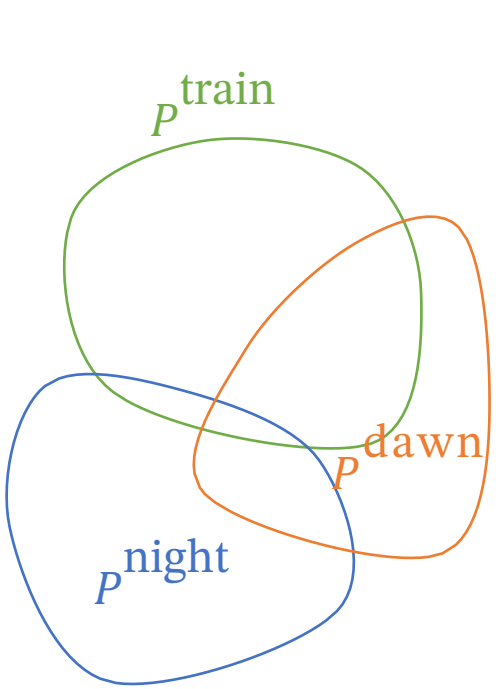


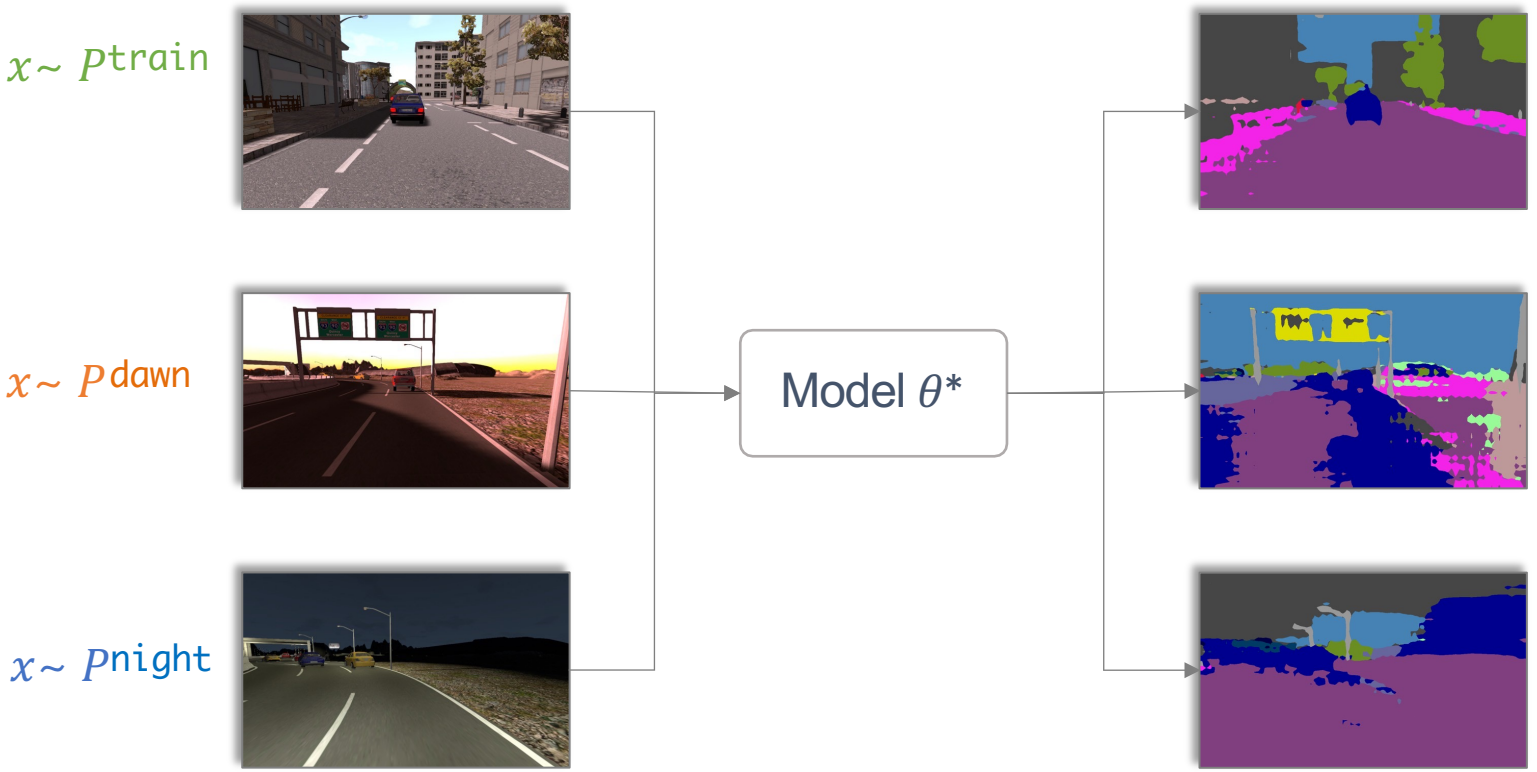
Learning algorithm

Trained model θ*

Learning i.i.d.

- **Domain shift**: the image distribution shift wrt train time ($P_X^{\text{train}} \neq P_X^{\text{test}}$)





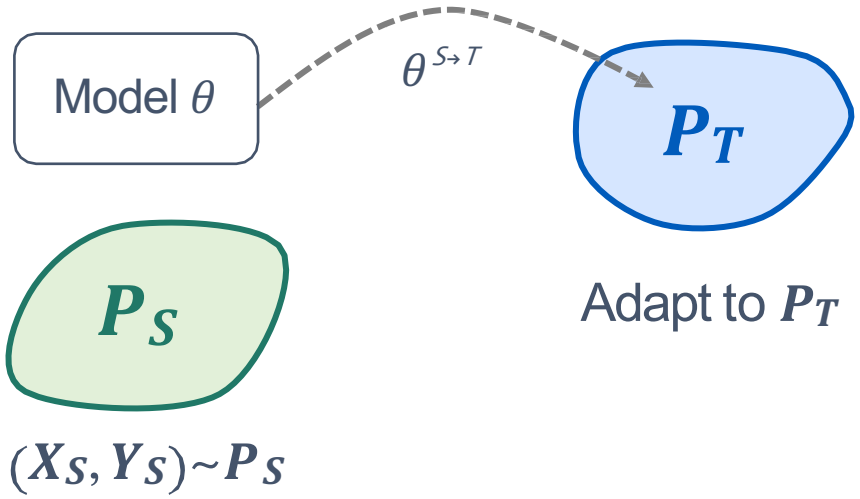
How to address domain shifts?

- A very large number of sub-fields
  - Supervised domain adaptation
  - Semi-supervised domain adaptation
  - Unsupervised domain adaptation
  - Domain generalization
  - …

- We focus here on **test-time adaptation**

Problem formulation

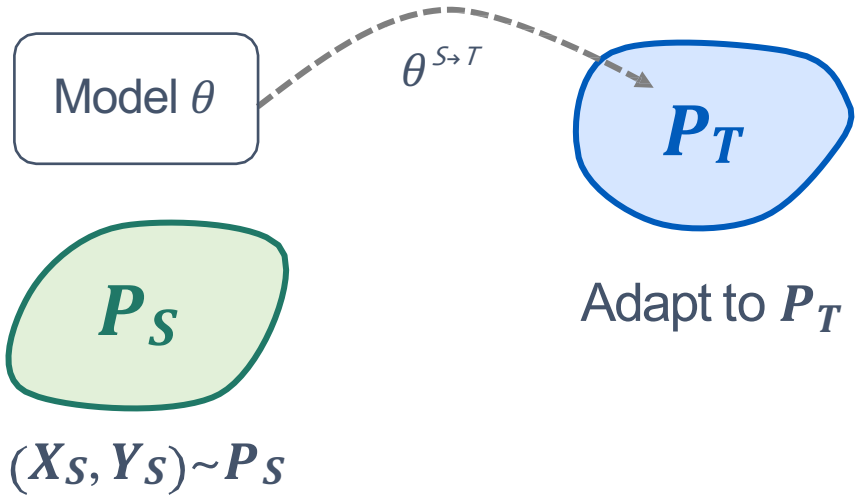
- **"Standard" UDA:** adapt from one or few **source** domains to one or few **target** domains



- Adaptation happens **offline**

Problem formulation

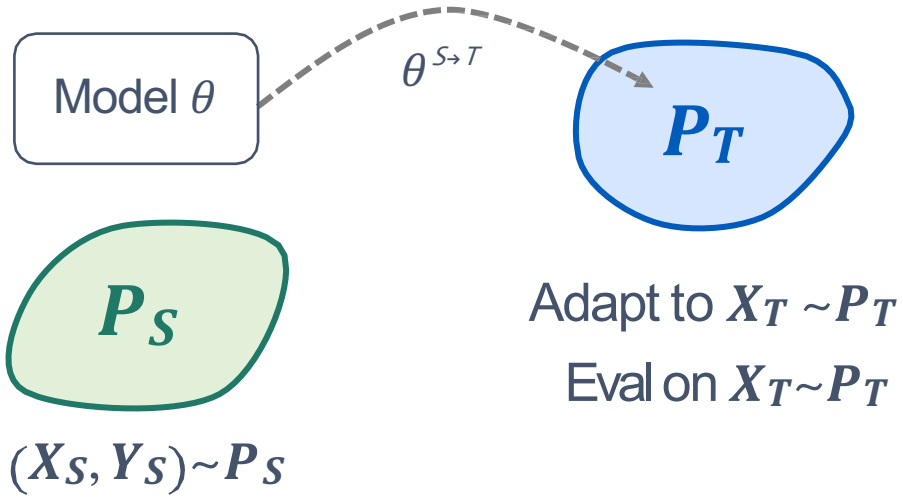
- **"Standard" UDA:** adapt from one or few **source** domains to one or few **target** domains



- Adaptation happens **offline**
- Can be
  - **Transductive** (adapt/test on same data)
  - **Inductive** (adapt/test on different data)

Problem formulation

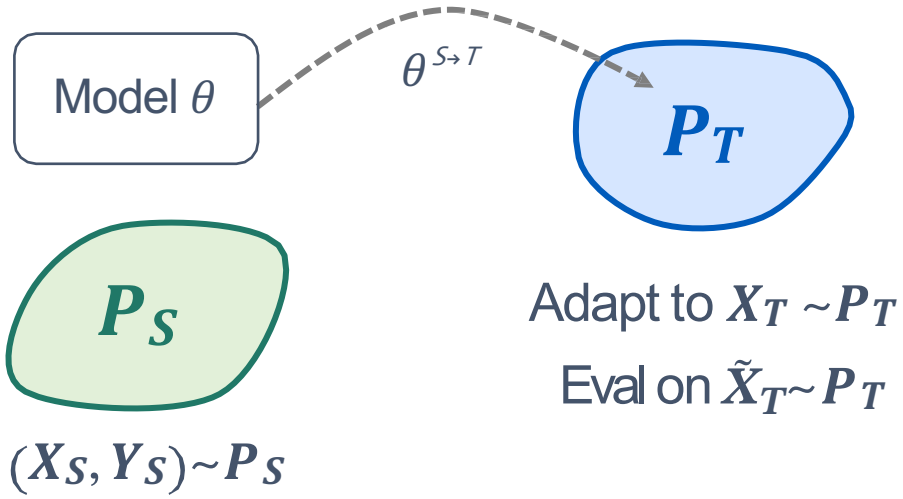
- **"Standard" UDA:** adapt from one or few **source** domains to one or few **target** domains



- Adaptation happens **offline**
- Can be
  - **Transductive** (adapt/test on same data)
  - **Inductive** (adapt/test on different data)

Problem formulation

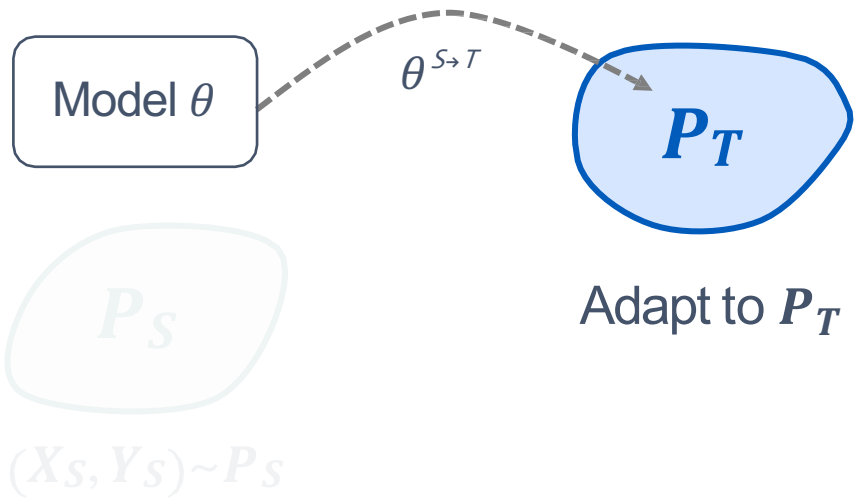
- **"Standard" UDA:** adapt from one or few **source** domains to one or few **target** domains



- Adaptation happens **offline**
- Can be
  - **Transductive** (adapt/test on same data)
  - **Inductive** (adapt/test on different data)

Problem formulation

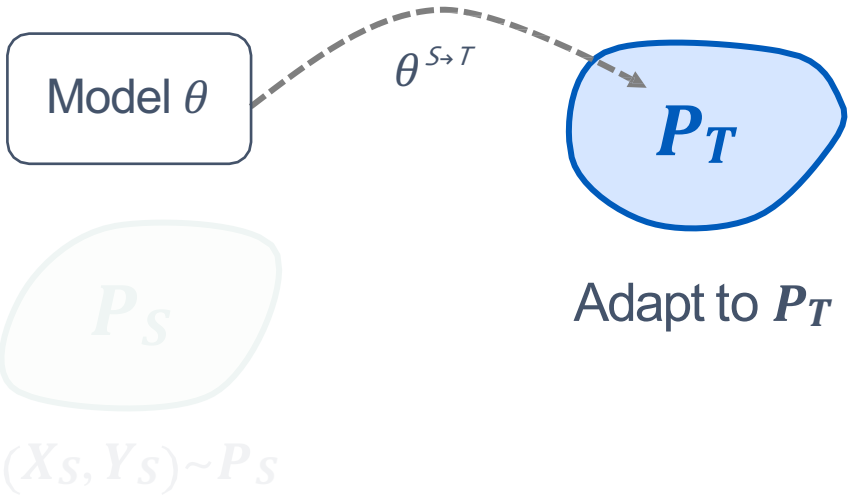
- **"Source-free" UDA:** adapt from one or few **source** domains to one or few **target** domains



- Adaptation happens **offline**
- Can be
  - **Transductive** (adapt/test on same data)
  - **Inductive** (adapt/test on different data)
- No access to the source dataset

Problem formulation

- **“Test-time Adaptation”**

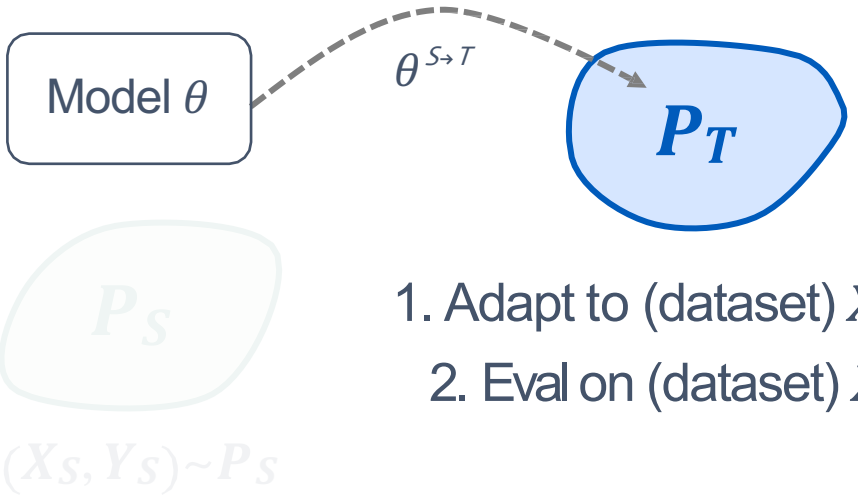


- Adaptation can happen
  - **Offline**
  - **Online**

- No access to the source dataset

Problem formulation

- **“Test-time Adaptation” = “Source-free Adaptation”**



1. Adapt to (dataset) $X_T \sim P_T$
2. Eval on (dataset) $X_T \sim P_T$

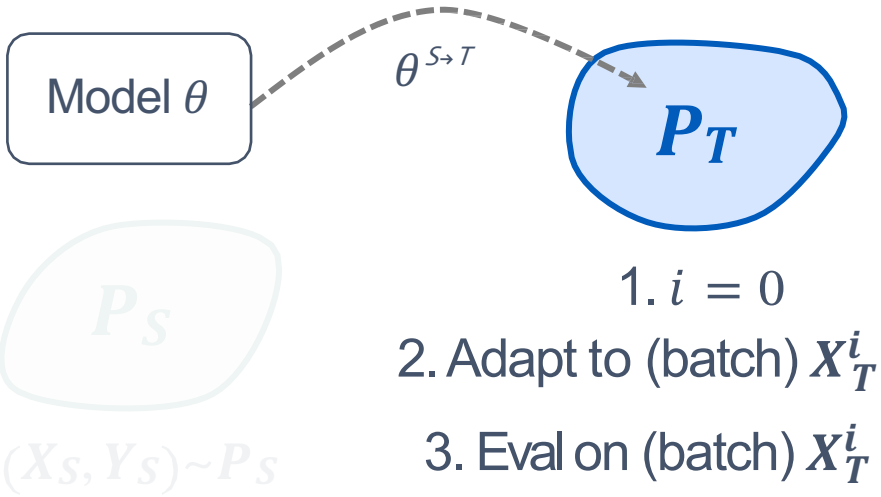
- Adaptation can happen
  - **Offline**
  - **Online**
- No access to the source dataset

Wang et al., “**Tent: Fully Test-time Adaptation by Entropy Minimization**” ICLR 2021

Chidlovskii et al., **“Domain Adaptaiton in the Absence of Source Domain Data”, SIGKDD 2016**

Problem formulation

- **“Test-time Adaptation”**



1. $i = 0$
2. Adapt to (batch) $X_T^i \sim P_T$
3. Eval on (batch) $X_T^i \sim P_T$
4. i++, back to 2.

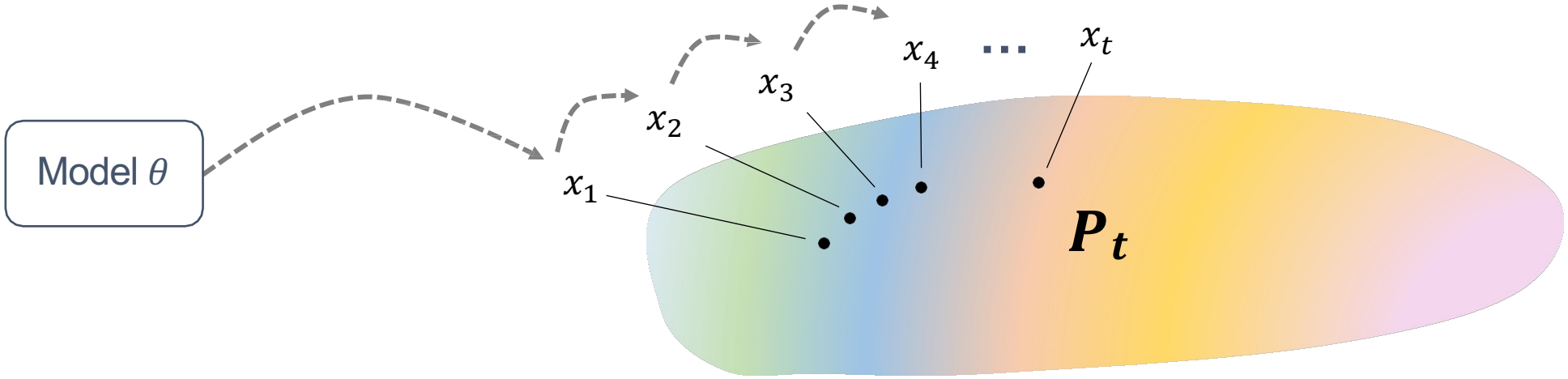
- Adaptation can happen
  - **Offline**
  - **Online**

We can also relax this assumption

- No access to the source dataset

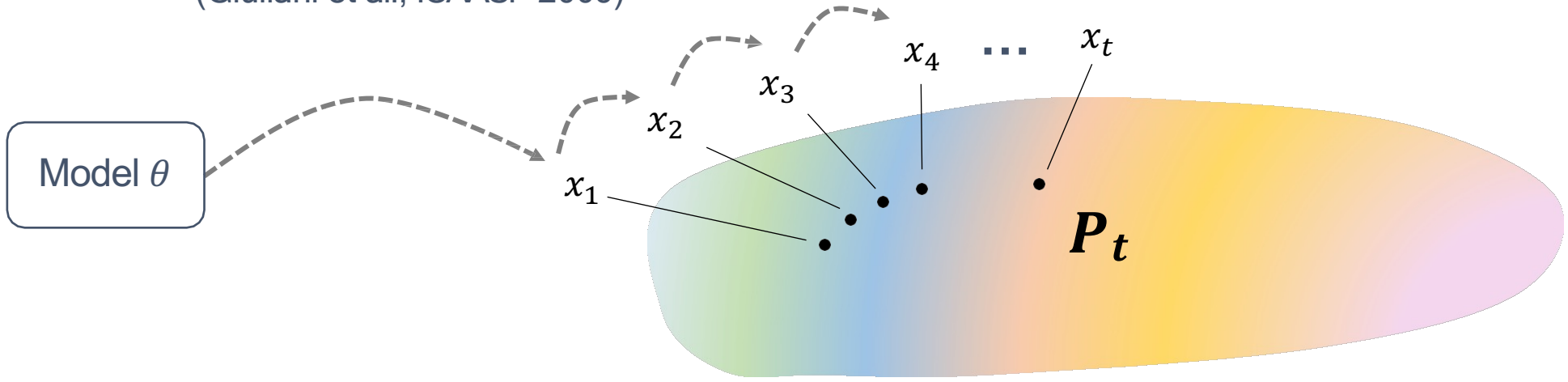
Problem formulation

- **“Continual TTA”: frame-by-frame** adaptation with **continuous shifts**
  - Samples are drawn from an **ever-changing distribution** $\longrightarrow$ $(x_t)_0^\infty \sim P_t$
  - Each sample/batch X_t represents an **adaptation problem in itself**



Problem formulation

- **“Continual TTA”: frame-by-frame** adaptation with **continuous shifts**
  - Seminal works in this setting are from the NLP literature
    - (Dredzer and Crammer, EMNLP 2009)
    - (Giuliani et al., ICAASP 2009)

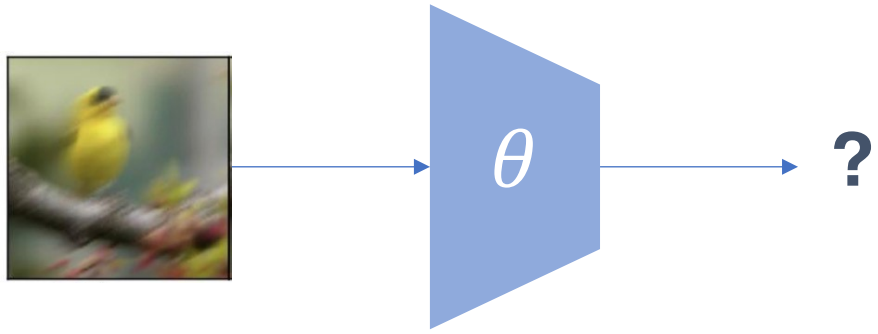


Dredzer and Crammer, “**Online Methods for Multi-domain Learning and Adaptation**”, EMNLP 2009

Giuliani et al., “**On-line speaker adaptation on telephony speech data with adaptively trained acoustic models**”, ICAASP 2009

Methods

- Overall goal: adapting a given model to new batches of data
  - Extreme case: single-sample adaptation

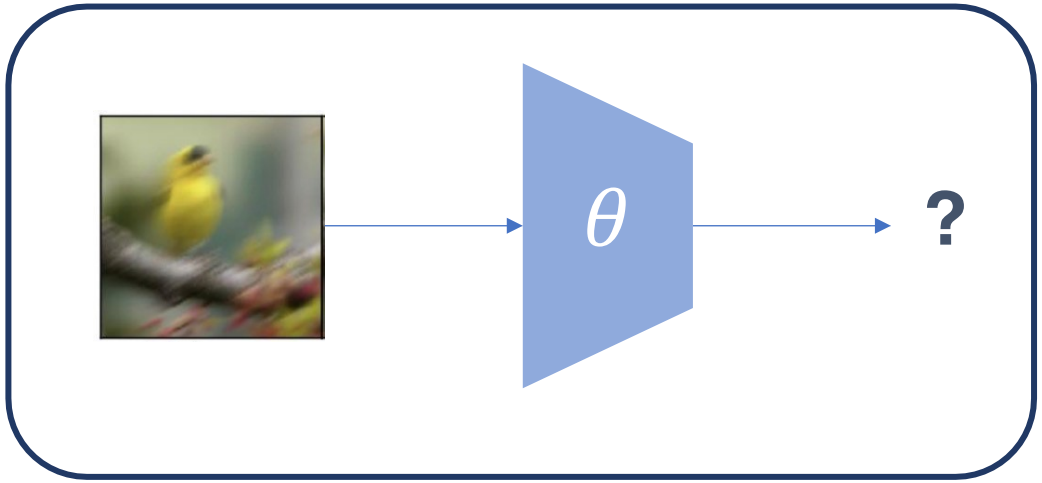


- Update model weights in order to maximize our chances of correct prediction
- We're not given any label

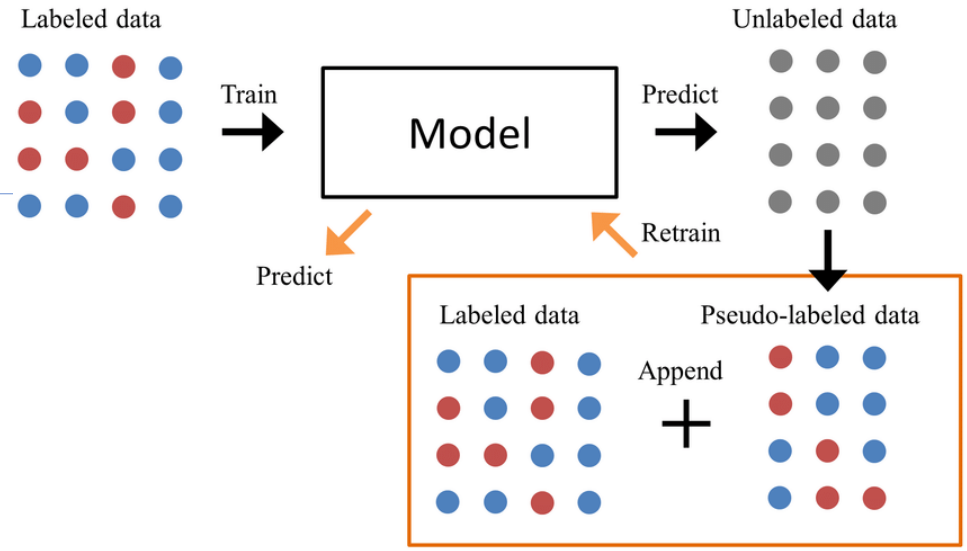
Image from Hendrycks and Dietterich **"Benchmarking Neural Network Robustness to Common Corruptions and Perturbations"**, ICLR 2019

Methods

- Self-training with pseudo-labels
- BatchNorm statistics adaptation
- BatchNorm parameters adaptation
- Self-supervised training
- Data augmentation



Methods



- **Self-training with pseudo-labels**
- **Standard recipe**
  - Trust (some of) your model's predictions
  - Use them as ground truth to update your model
  - Repeat
- Originally for **semi-supervised learning**
  - Large application in DA
  - Standard **baseline in TTA**

Lee et al., "**Pseudo-label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks**" ICMLW 2013

Methods

- **BatchNorm statistics adaptation**

- In BN layers we generally use the statistics from the training set
- We can update them with the target's
  - **Online** [Mancini et al. 2018]
  - **Offline** [Schneider et al. 2020]
- Often important not to completely replace the training ones (weighted)

$$\widehat{F^l(x_i^t)} = \gamma \cdot \frac{F^l(x_i^t) - \mu_l}{\sigma_l^2} + \beta$$

$$\mu_l := (1-\alpha) \cdot \mu_l + \alpha \cdot \mathbb{E}\{F^l(x_i^t)\}$$

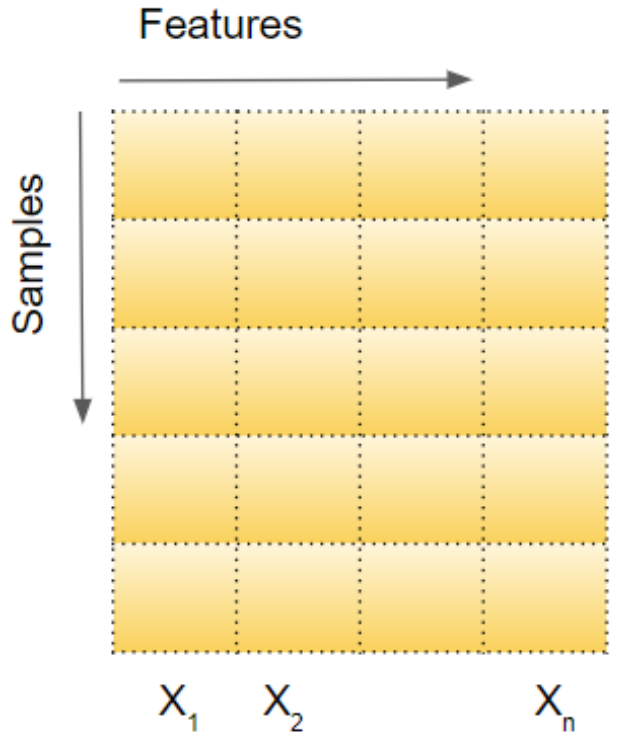
$$\sigma_l^2 := (1-\alpha) \cdot \sigma^2 + \alpha \cdot \mathbb{E}\{(F^l(x_i^t) - \mathbb{E}\{F^l(x_i^t)\})^2\}$$

Mancini et al., "**Kitting in the Wild through Online Domain Adaptation**" IROS 2018
Schneider et al., "**Improving robustness against common corruptions by covariate shift adaptation**", NeurIPS 2020

BatchNorm

- Batch Norm is a neural network layer that is now commonly used in many architectures. It often gets added as part of a Linear or Convolutional block and helps to stabilize the network during training.

How we normalize

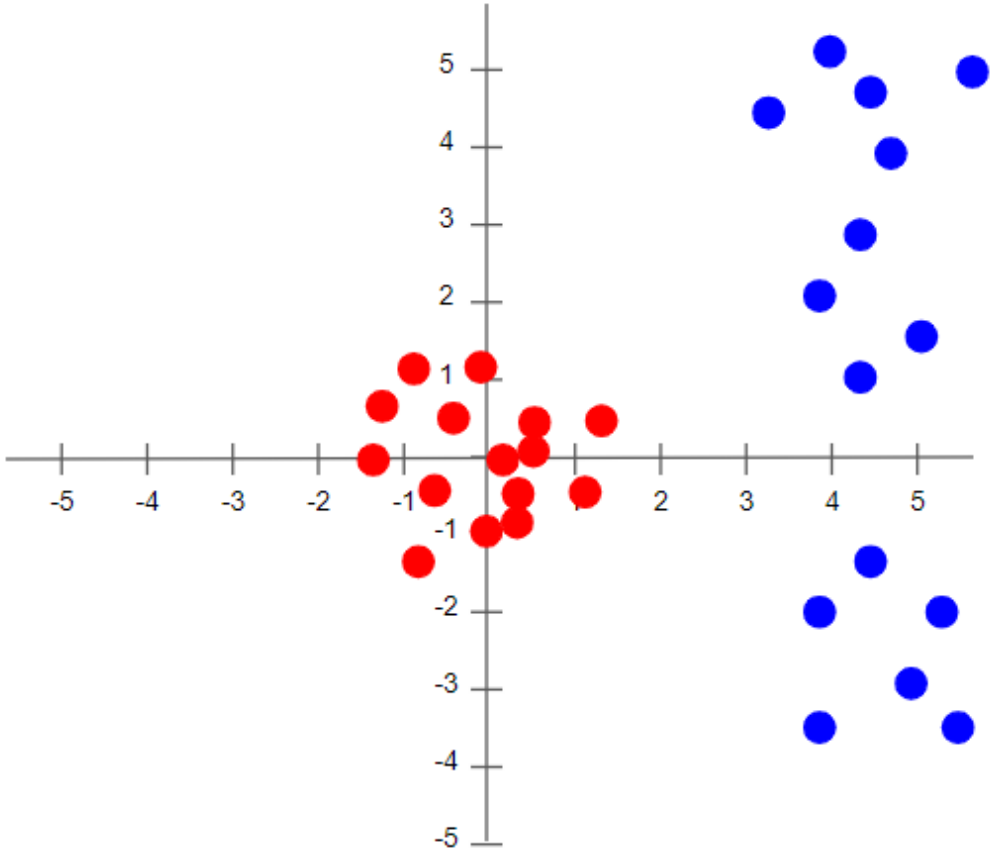


$$X_i = \frac{X_i - Mean_i}{StdDev_i}$$

https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

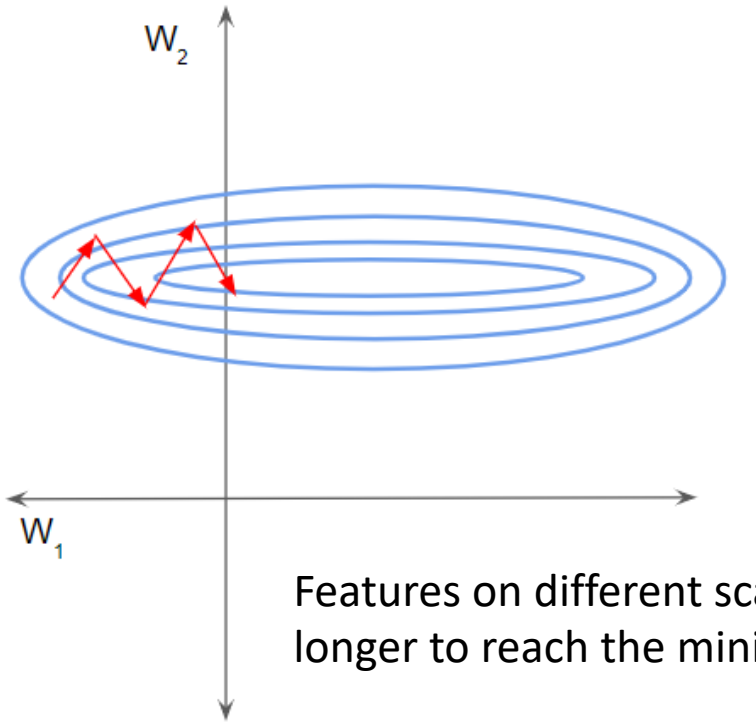
What normalized data looks like

- In the picture, we can see the effect of normalizing data. The original values (in blue) are now centered around zero (in red). This ensures that all the feature values are now on the same scale.

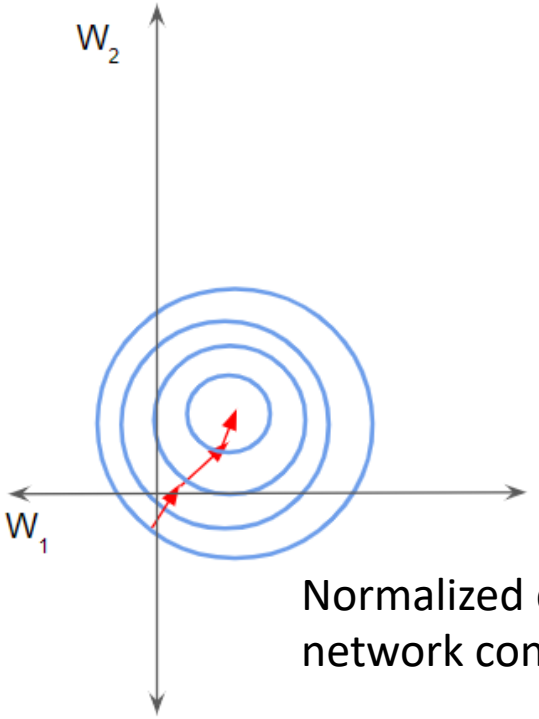


What normalized data looks like

- To understand what happens without normalization, let's look at an example with just two features that are on drastically different scales.



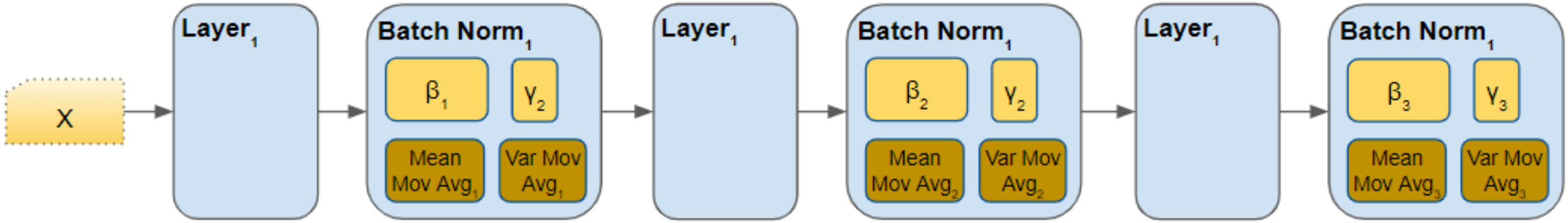
Features on different scales take longer to reach the minimum



Normalized data helps the network converge faster

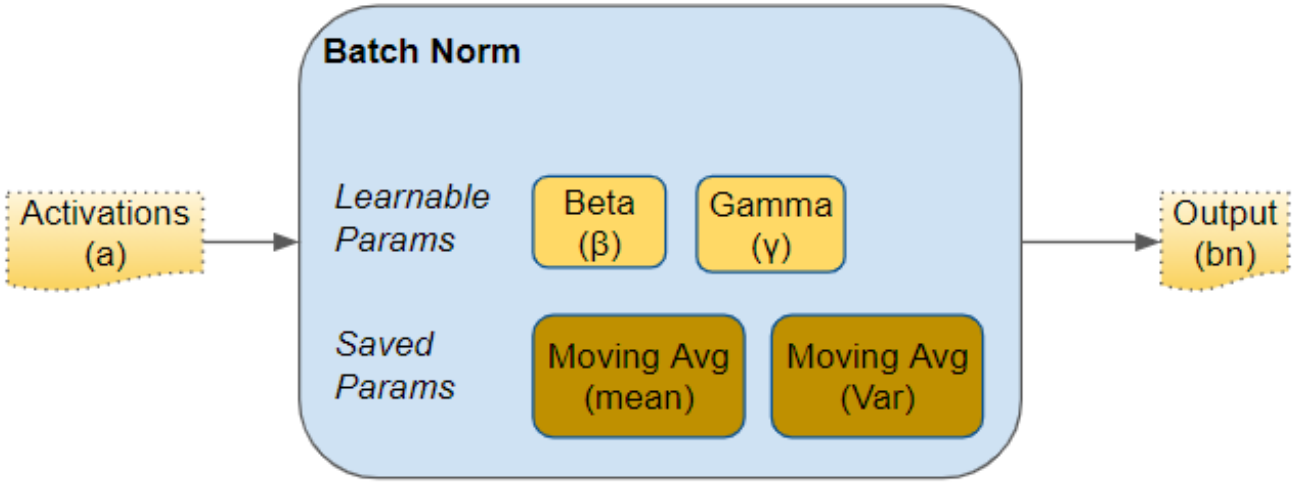
How Does BatchNorm work?

- Batch Norm is just another network layer that gets inserted between a hidden layer and the next hidden layer. Its job is to take the outputs from the first hidden layer and normalize them before passing them on as the input of the next hidden layer.



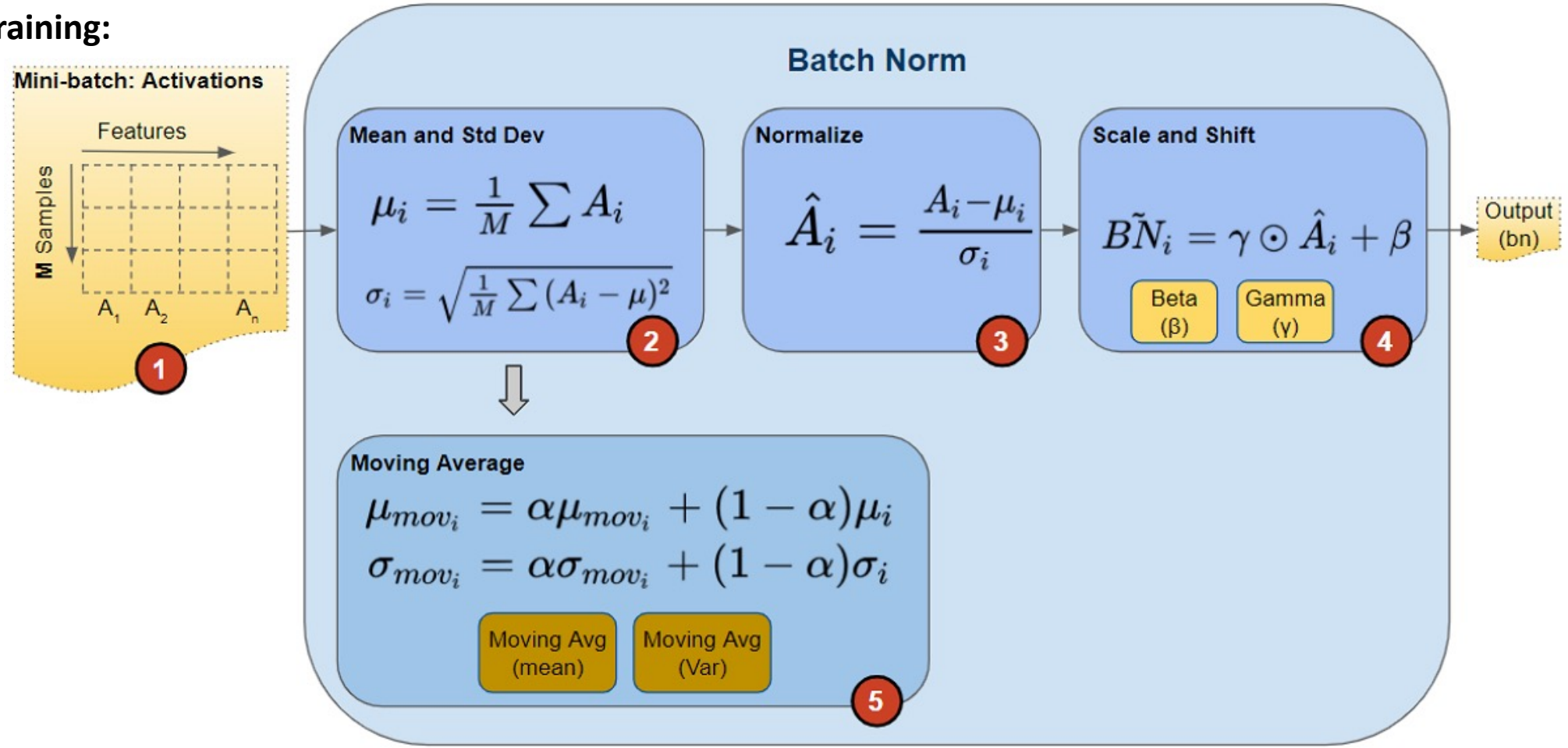
How Does BatchNorm work?

- Just like the parameters (eg. weights, bias) of any network layer, a Batch Norm layer also has parameters of its own:
  - Two learnable parameters called beta and gamma.
  - Two non-learnable parameters (Mean Moving Average and Variance Moving Average) are saved as part of the ‘state’ of the Batch Norm layer.



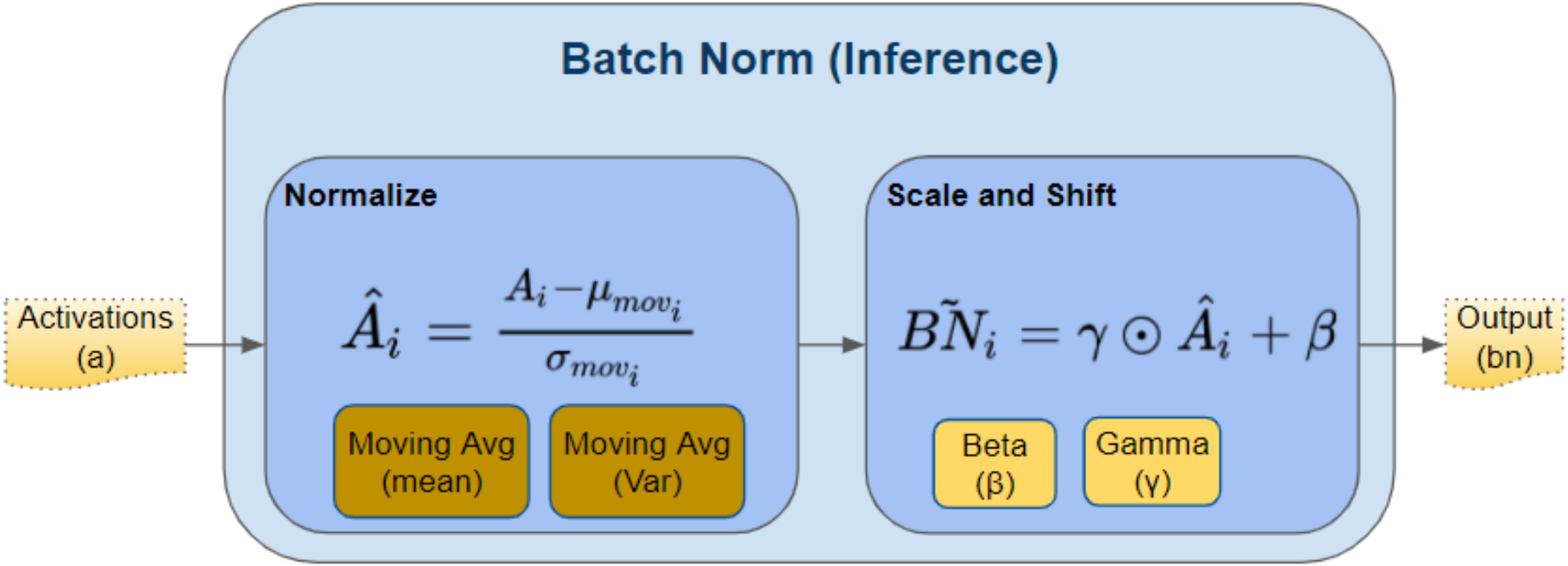
How Does BatchNorm work?

Training:



How Does BatchNorm work?

Inference:



Methods

- **(Batch)Norm parameters adaptation**

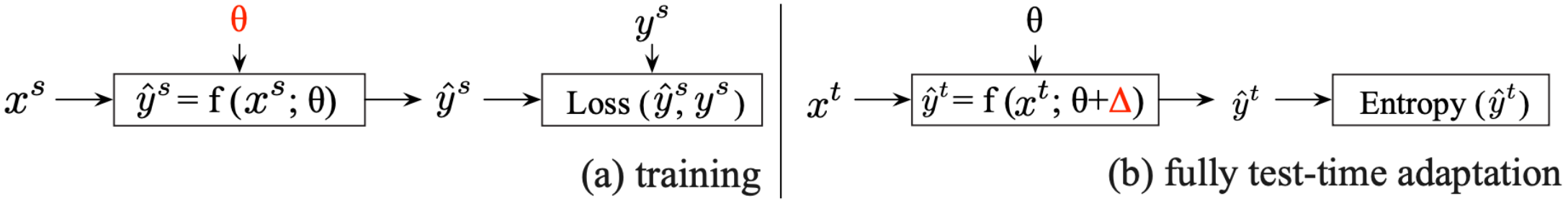
- **Entropy minimization** is another standard technique from semi-supervised learning
- But updating all network parameters cause **huge drifts from the original model**
- We can just **update the BatchNorm parameters** (or LayerNorm, etc.) via entropy minimization
- At the same time, we can update statistics

$$\underset{\beta,\gamma}{\operatorname{argmin}} \mathcal{L}_H := -\sum_{p \in x_i^t} \sum_{c}^{C} \hat{y}_{i,c}^{p} \log \hat{y}_{i,c}^{p}$$

Methods

- **(Batch)Norm parameters adaptation**

- **We do not alter training (a), but minimizes the entropy of predictions during testing (b) over a constrained modulation Δ, given the parameters θ and target data x^t.**



Methods

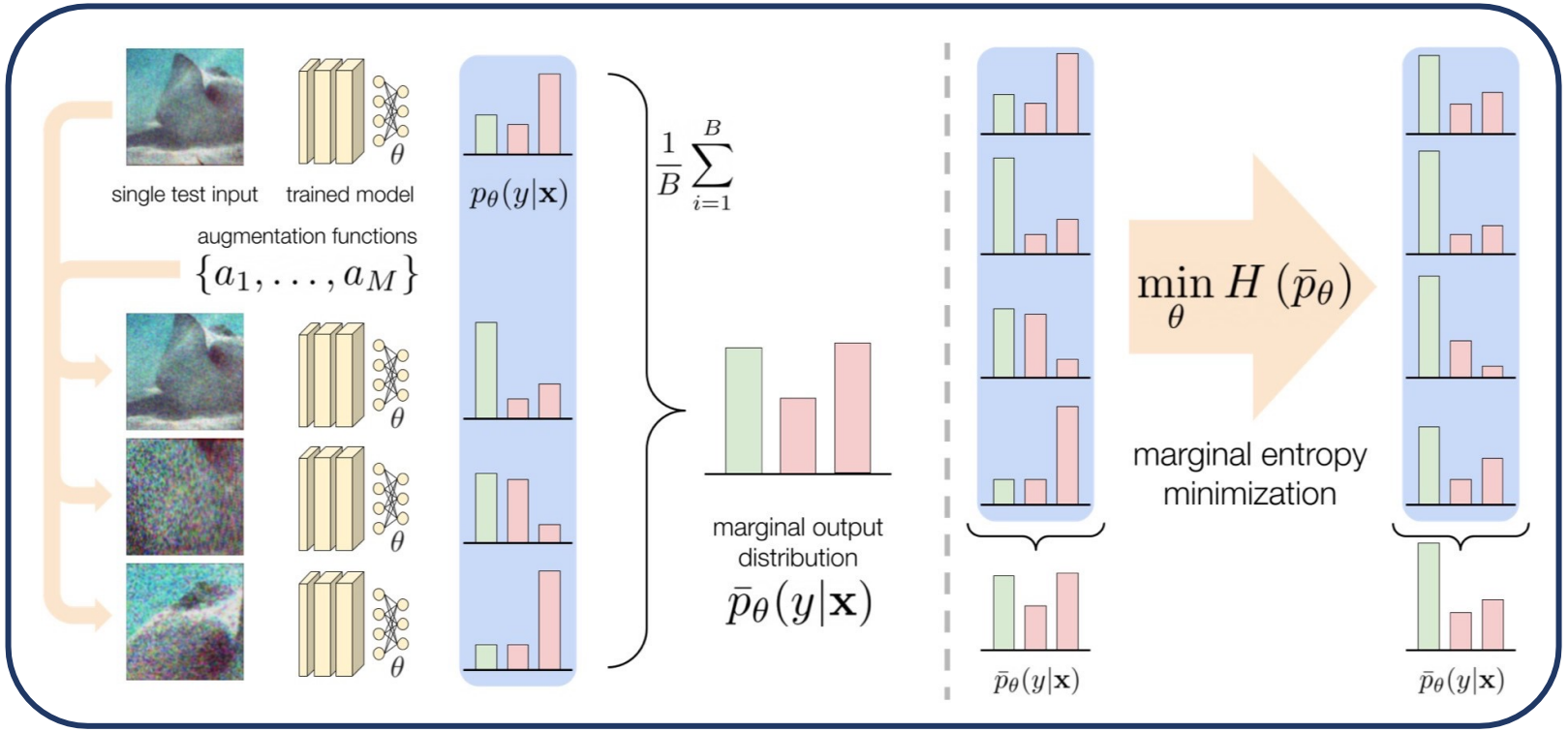
- **Self-supervised learning**

- We can solve a SSL objective using the test data
- Given a test-sample or a batch, we solve a SSL problem before making a prediction
- Note: SSL pre-training itself helps robustness
  - See Hendrycks et al., "**Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty**", NeurIPS 2019

Methods

- **Data augmentation**

- We can generate several copies of the current batch and use some of the previously mentioned objectives (e.g. entropy minimization)

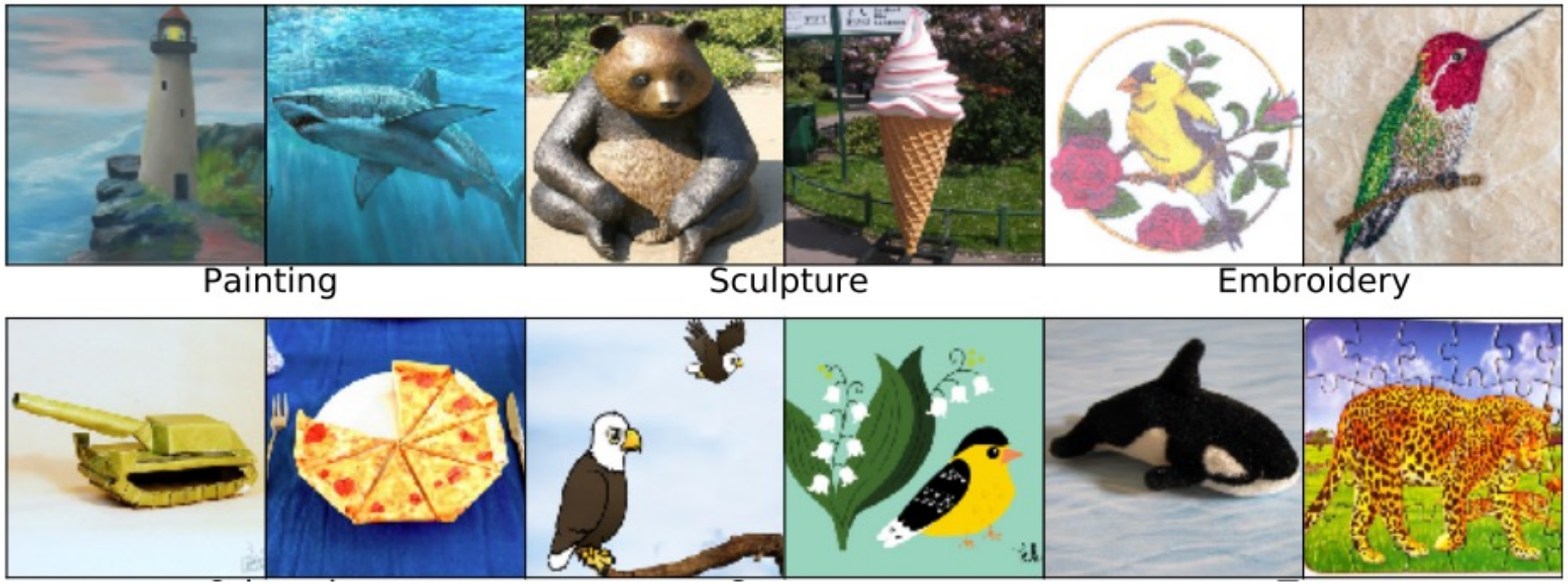


Benchmarks

- In general, train on one dataset and adapt to another one
- Researchers have mostly played with
  - ImageNet to ImageNet-C/A/R
  - CIFAR10 to CIFAR10-C
  - CIFAR100 to CIFAR100-C
- The only constraint, is that the set of classes need to be the same
  - TTA does not fit class-incremental purposes
  - We *could* have new classes, but we would be helpless

Benchmarks

- ImageNet-R (ImageNet-Rendition)



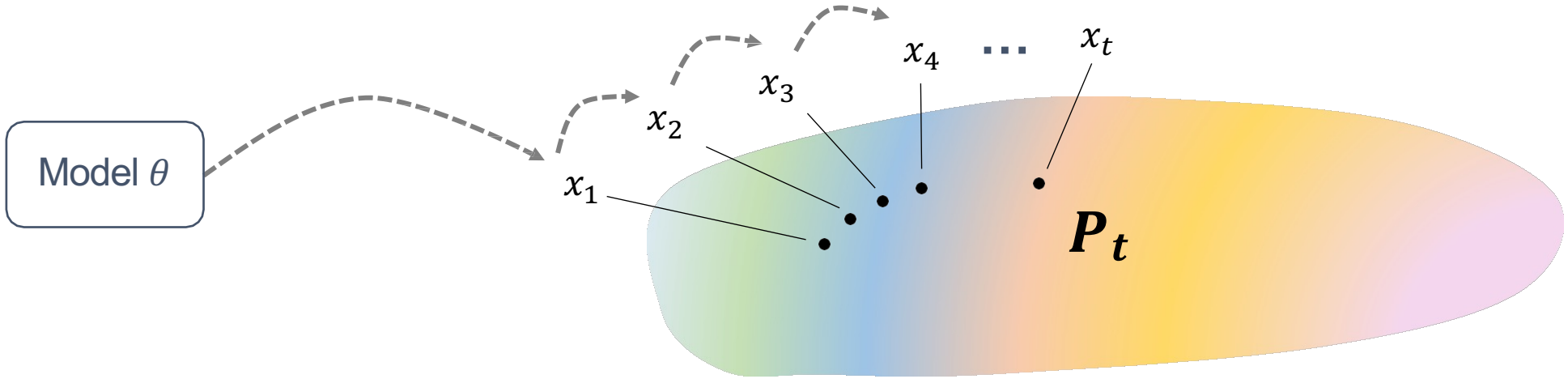
Benchmarks

	ImageNet-C mCE ↓	ImageNet-R Error (%)	ImageNet-A Error (%)
Baseline ResNet-50 [11]	76.7	63.9	100.0
+ TTA	77.9 (+1.2)	61.3 (−2.6)	98.4 (−1.6)
+ Single point BN	71.4 (−5.3)	61.1 (−2.8)	99.4 (−0.6)
+ MEMO (ours)	69.9 (−6.8)	58.8 (−5.1)	99.1 (−0.9)
+ BN ($N = 256, n = 256$)	61.6 (−15.1)	59.7 (−4.2)	99.8 (−0.2)
+ Tent (online) [46]	54.4 (−22.3)	57.7 (−6.2)	99.8 (−0.2)
+ Tent (episodic)	64.7 (−12.0)	61.0 (−2.9)	99.7 (−0.3)

From Zhang et al., "**NEMO: Test Time Robustness via Adaptation and Augmentation**" NeurIPS 2022

Continual TTA

- Addressing TTA in a continually evolving environment



- Additional challenge: **catastrophic forgetting**

Methods

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- Baselines:
  - **Self-training with pseudo-labels**
  - **BN statistics adaptation**
  - **BN parameters adaptation**
  - **Self-supervised training**

Methods

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- Baselines:
  - **Self-training with pseudo-labels**
  - **BN statistics adaptation**
  - **BN parameters adaptation**
  - **Self-supervised training**

1. Trust (some of) your model's predictions
2. Use them as ground truth to update your model
3. Repeat

Lee et al., "**Pseudo-label: The Simple and Efficient Semi-supervised Learning Method for Deep Neural Networks**" ICMLW 2013

Methods

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- Baselines:
  - **Self-training with pseudo-labels**
  - **BN statistics adaptation**
  - **BN parameters adaptation**
  - **Self-supervised training**

$$\widehat{F^l(x_i^t)} = \gamma \cdot \frac{F^l(x_i^t) - \mu_l}{\sigma_i^2} + \beta$$

$$\mu_l := (1-\alpha) \cdot \mu_l + \alpha \cdot \mathbb{E}\{F^l(x_i^t)\}$$

$$\sigma_l^2 := (1-\alpha) \cdot \sigma^2 + \alpha \cdot \mathbb{E}\{(F^l(x_i^t) - \mathbb{E}\{F^l(x_i^t)\})^2\}$$

Mancini et al., "**Kitting in the Wild through Online Domain Adaptation**" IROS 2018

Schneider et al., "**Improving robustness against common corruptions by covariate shift adaptation**", NeurIPS 2020

Methods

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- Baselines:
  - **Self-training with pseudo-labels**
  - **BN statistics adaptation**
  - **BN parameters adaptation**
  - **Self-supervised training**

BN statistics adaptation

+

$$\underset{\beta,\gamma}{\operatorname{argmin}} \mathcal{L}_H := -\sum_{p \in x_i^t} \sum_{c}^{C} \hat{y}_{i,c}^{p} \log \hat{y}_{i,c}^{p}$$

Methods

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- Baselines:
  - **Self-training with pseudo-labels**
  - **BN statistics adaptation**
  - **BN parameters adaptation**
  - **Self-supervised training**

Solve a side SSL objective on the target samples

Catastrophic forgetting

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- **Main problem**: like often in continual learning, **catastrophic forgetting**
- We're learning in an unsupervised way, so it's not trivial how to avoid the model to forget classes.
- Classes that are more rare will disappear, leaving their space to the more abundant ones

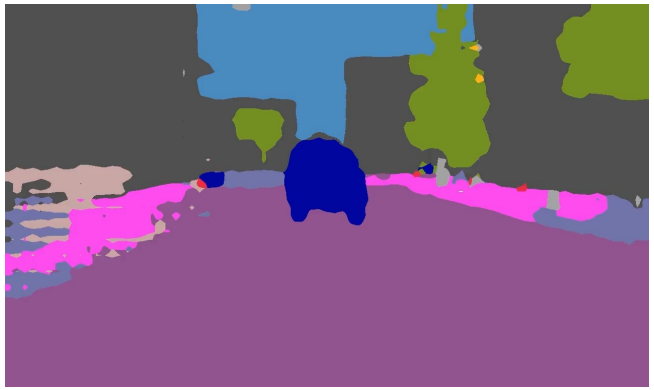
- **Example:** in urban street segmentation, it's easy to forget about ***things*** (countable objects), overtaken by the more abundant ***stuff*** (street, sky, buildings, etc.)

Catastrophic forgetting

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- **Main problem**: like often in continual learning, **catastrophic forgetting**





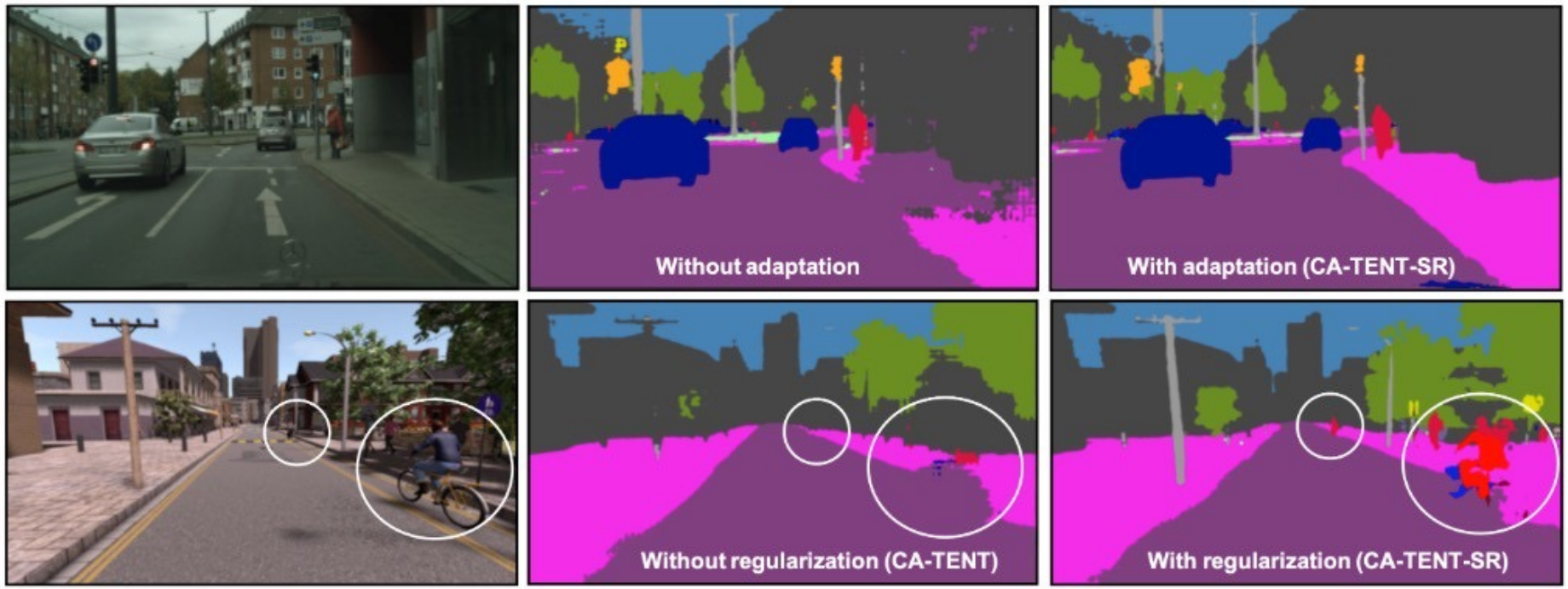

BEST BURGER IN TOWN

Catastrophic forgetting

- The goal is adapting **frame-by-frame** to streams of **temporally correlated, unlabeled samples**
- Each sample from the sequence $(x)_{t=1}^{\infty} \sim P_t$ represents an adaptation problem itself

- **Some solutions:**
  - **"Naive" learning:** instead of doing continual learning, at each frame re-start from the original model
  - **Memories:** keep rehearsing the original (labelled) training samples to the model
  - **Reset strategies:** use the original model as a checkpoint, and reset when some thershold is met

Results


Without adaptation
With adaptation (CA-TENT-SR)
Without regularization (CA-TENT)
With regularization (CA-TENT-SR)

Conclusions

- Test-time adaptation is a recent and active research area

- Yet, its roots are from well established fields
  - Domain adaptation
  - Online learning
  - Self-training

- Its continual counterpart introduces additional challenges
  - Catastrophic forgetting
  - Evaluating in a ever-changing environments

References

No representation learning

- (NLP) Dredze and Crammer, Online Methods for Multi-Domain Learning and Adaptation, EMNLP 2008
- (NLP) Giuliani et al., On-line speaker adaptation on telephony speech data with adaptively trained acoustic models, ICASSP 2009
- (supervised) Zao and Hoi, OTL: A Framework of Online Transfer Learning, ICML 2010
- Hoffman et al., Continuous Manifold Based Adaptation For Evolving Visual Domains, CVPR 2014
- (supervised) Xu et al., Incremental Domain Adaptation of Deformable Part-based Models, BMVC 2014
- Lampert, Predicting the Future Behavior of a Time-Varying Probability Distribution, CVPR 2015
- Soleymani et al., Incremental Evolving Domain Adaptation, IEEE Transactions on Knowledge and Data Engineering 2016
- Li et al., Domain Generalization and Adaptation Using Low Rank Exemplar SVMs, TPAMI 2018
- Moon et al., Multi-step Online Unsupervised Domain Adaptation, ICASSP 2020

Deep learning-based

- Mancini et al., Kitting in the Wild through Online Domain Adaptation, IROS 2018
- Zhang et al., Online Adaptation through Meta-Learning for Stereo Depth Estimation, arXiv 2019
- Ashukha et al., Pitfalls of in-Domain Uncertainty Estimation and Ensembling in Deep Learning, ICLR 2020
- Sun et al., Test-Time Training with Self-Supervision for Generalization under Distribution Shifts, ICML 2020
- Schneider et al., Improving robustness against common corruptions by covariate shift adaptation, NeurIPS 2020
- Wang et al., Tent: Fully Test-time Adaptation by Entropy Minimization, ICLR 2021
- Ikasawa and Matsuo, Test-Time Classifier Adjustment Module for Model-Agnostic Domain Generalization, NeurIPS 2021
- Liu et al., TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive?, NeurIPS 2021

References

Deep learning-based

- Nado et al., Evaluating Prediction-Time Batch Normalization for Robustness under Covariate Shift, ICML 2020 Woskshops
- Karani et al., A Field of Experts Prior for Adapting Neural Networks at Test Time, arXiv 2022
- Xiao et al., Learning to Generalize across Domains on Single Test Samples, ICLR 2022
- Volpi et al., On the Road to Online Adaptation for Semantic Image Segmentation, CVPR 2022
- Wange et al., Continual Test-Time Domain Adaptation, CVPR 2022
- Klingner et al., Continual BatchNorm Adaptation (CBNA) for Semantic Segmentation, IEEE T. on Intelligent Transportation Systems 2022
- Chen et al., Contrastive Test-Time Adaptation, CVPR 2022
- Valanarasu et al., On-the-Fly Test-time Adaptation for Medical Image Segmentation, MIDL 2023
- Yang et al., Test-time Batch Normalization, arXiv 2022
- Bateson et al., Test-Time Adaptation with Shape Moments for Image Segmentation, MICCAI 2022
- Jung et al., CAFA: Class-Aware Feature Alignment for Test-Time Adaptation, arXiv 2022
- Gao et al., Back to the Source: Diffusion-Driven Test-Time Adaptation, CVPR 2023
- Rusak et al., If your data distribution shifts, use self-learning, TMLR 2022
- Niu et al., Efficient Test-Time Model Adaptation without Forgetting, ICML 2022
- Choi et al., Improving Test-Time Adaptation via Shift-agnostic Weight Regularization and Nearest Source Prototypes, ECCV 2022
- Liu et al., Single-domain Generalization in Medical Image Segmentation via Test-time Adaptation from Shape Dictionary, AAAI 2022
- Kojima et al., Robustifying Vision Transformer without Retraining from Scratch by Test-Time Class-Conditional Feature Alignment, IJCAI 2022

References

Deep learning-based

- Thopalli et al., Domain Alignment Meets Fully Test-Time Adaptation, ACML 2022
- Ma et al., Test-time Adaptation with Calibration of Medical Image Classification Nets for Label Distribution Shift, MICCAI 2022
- Saltori et al., GIPSO: Geometrically Informed Propagation for Online Adaptation in 3D LiDAR Segmentation, ECCV 2022
- Cordier et al., Test-Time Adaptation with Principal Component Analysis, ECML/PKDD workshops 2022
- Frey et al., Continual Adaptation of Semantic Segmentation using Complementary 2D-3D Data Representations, RAL 2022
- Boudiaf et al., Parameter-free Online Test-time Adaptation, CVPR 2022
- Gandelsman et al., Test-Time Training with Masked Autoencoders, NeurIPS 2022
- Zhang et al., MEMO: Test Time Robustness via Adaptation and Augmentation, NeurIPS 2022
- Shu et al., Test-Time Prompt Tuning for Zero-Shot Generalization in Vision-Language Models, NeurIPS 2022
- Goyal et al., Test-time Adaptation via Conjugate Pseudo-labels, NeurIPS 2022
- Sinha et al., TeST: Test-time Self-Training under Distribution Shift, WACV 2023
- Khurana et al., SITA: Single Image Test-time Adaptation, arXiv 2021
- Lin et al., Video Test-Time Adaptation for Action Recognition, CVPR 2023
- Yu et al., Mitigating Forgetting in Online Continual Learning via Contrasting Semantically Distinct Augmentations, arXiv 2022
- Lim et al., TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation, ICLR 2023
- Gaillochet et al., TAAL: Test-time Augmentation for Active Learning in Medical Image Segmentation, MICCAI-DALI 2022
- Han et al., Rethinking Precision of Pseudo Label: Test-Time Adaptation via Complementary Learning, arXiv 2023

References

Deep learning-based

- Ma et al., Test-time Adaptation with Calibration of Medical Image Classification Nets for Label Distribution Shift, MICCAI 2022
- Qian and del Hougne, Noise-Adaptive Intelligent Programmable Meta-Imager, arXiv 2022
- Jung et al., CAFA: Class-Aware Feature Alignment for Test-Time Adaptation, arXiv 2023
- Das et al., TransAdapt: A Transformative Framework for Online Test Time Adaptive Semantic Segmentation, ICASSP 2023
- Yang et al., AUTO: Adaptive Outlier Optimization for Online Test-Time OOD Detection, arXiv 2023
- Liang et al., A Comprehensive Survey on Test-Time Adaptation under Distribution Shifts, arXiv 2023
- Yu et al., Benchmarking Test-Time Adaptation against Distribution Shifts in Image Classification, arXiv 2023
- Lim et al., TTN: A Domain-Shift Aware Batch Normalization in Test-Time Adaptation, ICLR 2023
- Li et al., On the Robustness of Open-World Test-Time Training: Self-Training with Dynamic Prototype Expansion, ICCV 2023
- Zhang et al., DomainAdaptor: A Novel Approach to Test-time Adaptation, arXiv 2023
- Hakim et al., ClusT3: Information Invariant Test-Time Training, ICCV 2023
- Bertrand et al., Test-time Training for Matching-based Video Object Segmentation, NeurIPS 2023

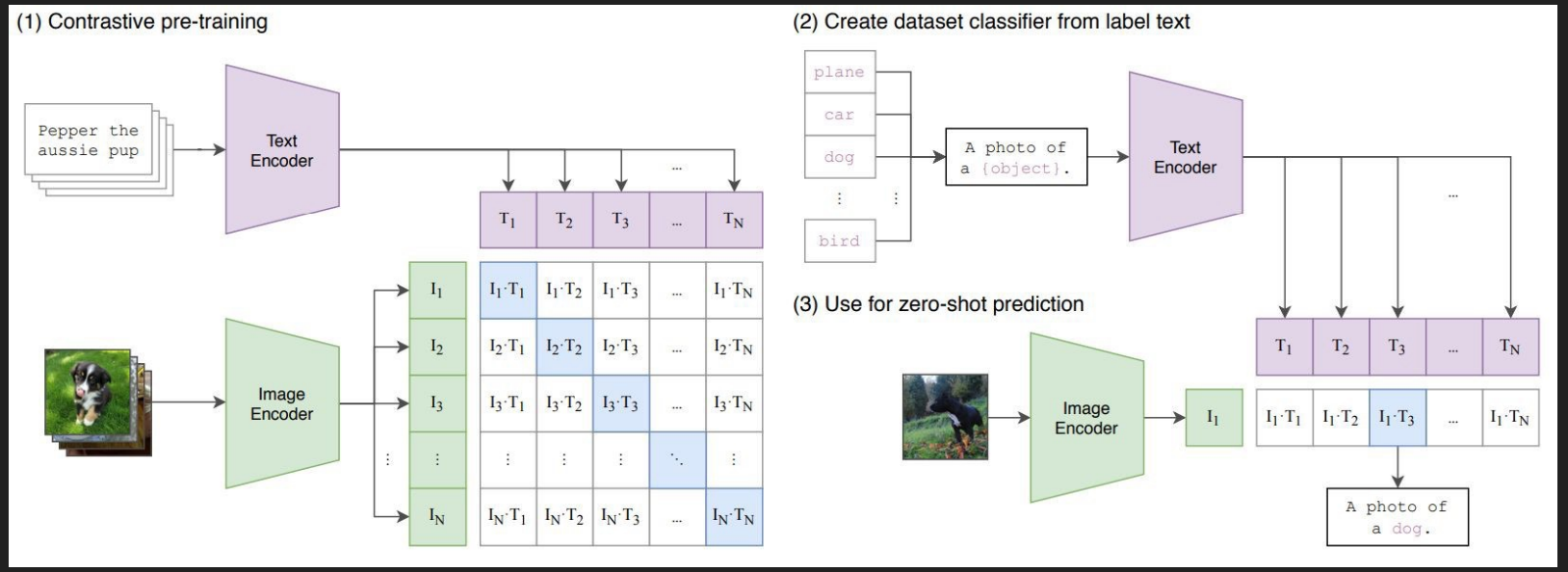
Many works surely missing, please also check

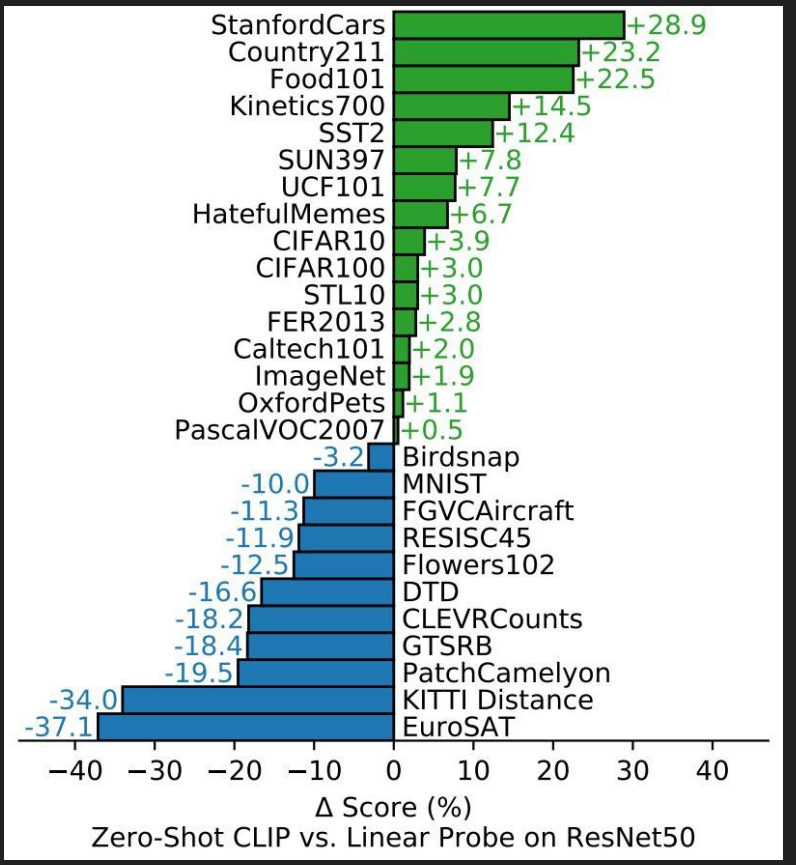
- **https://github.com/tim-learn/awesome-test-time-adaptation**
- **https://github.com/YuejiangLIU/awesome-source-free-test-time-adaptation**

This Lecture – Agenda

- **Reliability of Deep Learning for Real-World Deployment**
  - Uncertainty Estimation
    - Data Uncertainty
    - Knowledge Uncertainty
    - Ensemble Approaches
  - Adaptation to Unseen Concepts on the Fly
    - Dynamic Adaptation: Test-time Adaptation
    - Static Adaptation: Vision-Language Learning

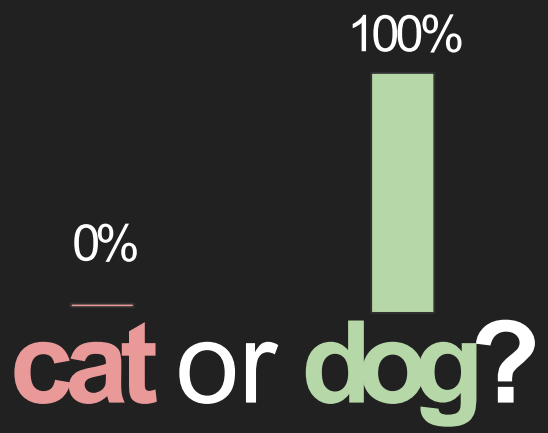
VLMs work!

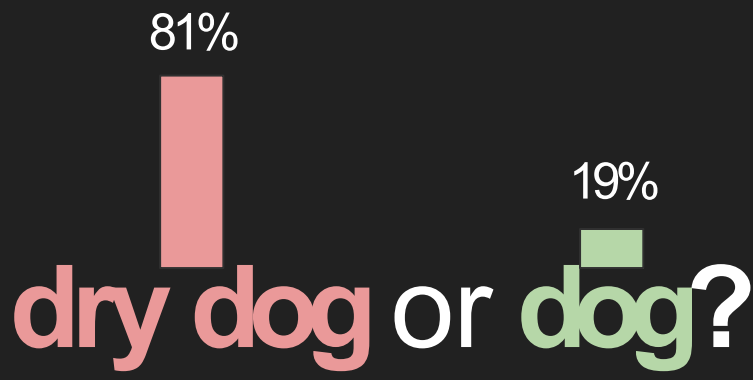




A. Radford et al., "*Learning Transferable Visual Models from Natural Language Supervision*", ICML 2021.

Let's try it out!





https://huggingface.co/spaces/ShivamShrirao/CLIP-Zero-Shot-Classifier