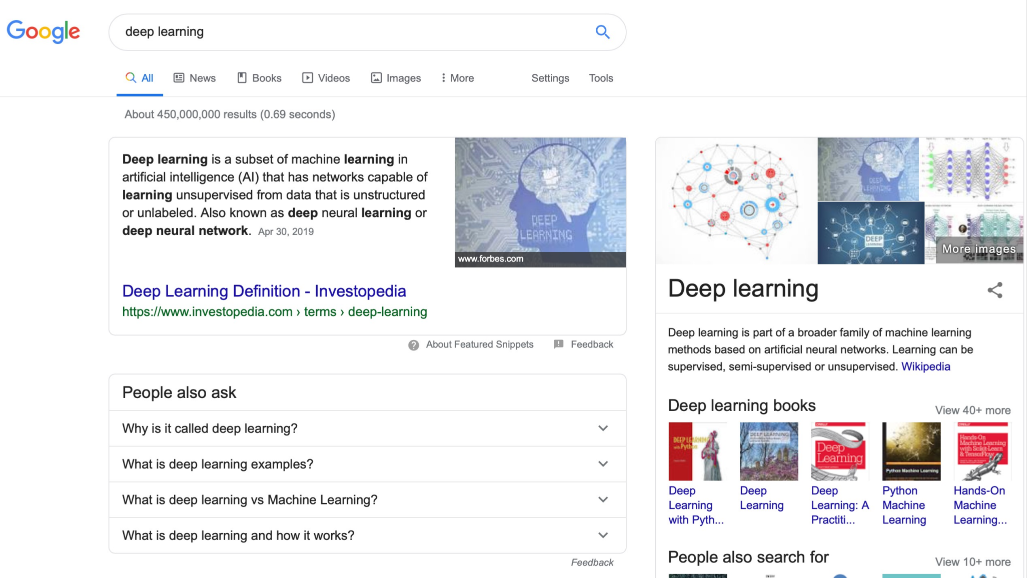
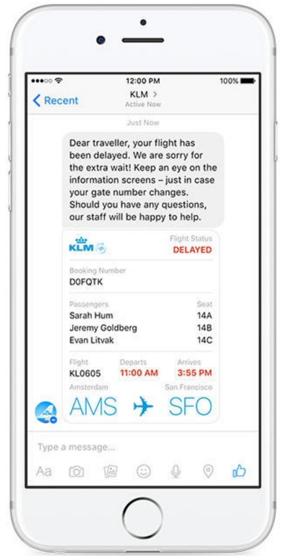
Bayesian Deep Learning:
Insights, Methods, and Applications

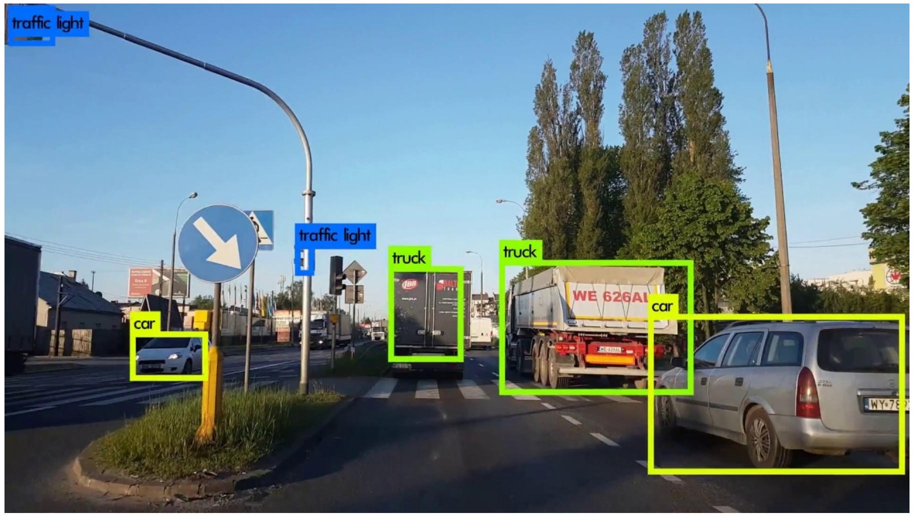
Zhijie Deng

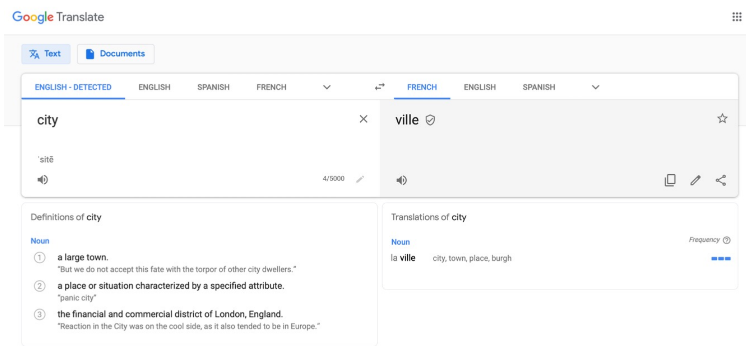
Tsinghua University

Deep learning success









Problems remain

- Lack uncertainty

- Data driven (label-eager, poor robustness, etc.)

The promise of probabilistic (Bayesian) modeling

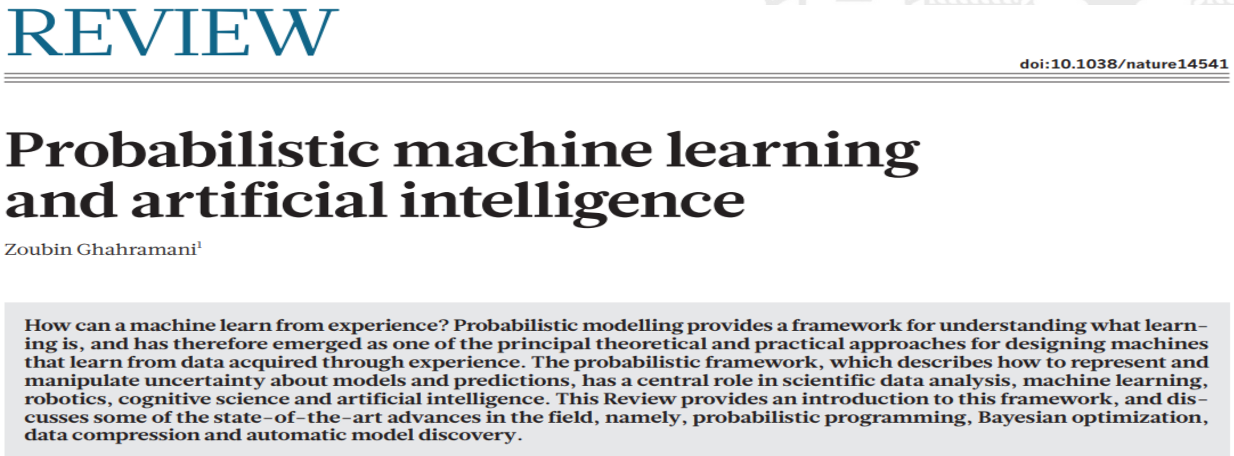
Uncertainty

Prior belief

$$p(\theta|D) = \frac{p(D|\theta)\pi(\theta)}{p(D)}$$

Thomas Bayes (1702 – 1761)

U. Cambridge
Fellow of the Royal Society (FRS)



REVIEW

doi:10.1038/nature14541

Probabilistic machine learning and artificial intelligence

Zoubin Ghahramani[1]

How can a machine learn from experience? Probabilistic modelling provides a framework for understanding what learning is, and has therefore emerged as one of the principal theoretical and practical approaches for designing machines that learn from data acquired through experience. The probabilistic framework, which describes how to represent and manipulate uncertainty about models and predictions, has a central role in scientific data analysis, machine learning, robotics, cognitive science and artificial intelligence. This Review provides an introduction to this framework, and discusses some of the state-of-the-art advances in the field, namely, probabilistic programming, Bayesian optimization, data compression and automatic model discovery.

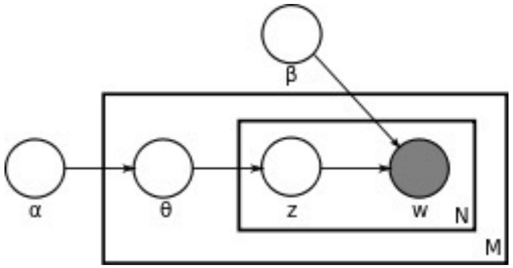
Uncertainty: let models knowing their limits

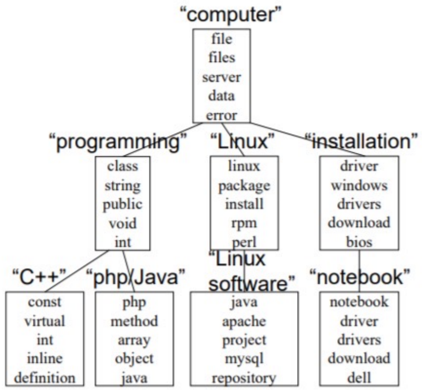
All models are wrong, but ~~some~~ models that know when they are wrong, are useful.

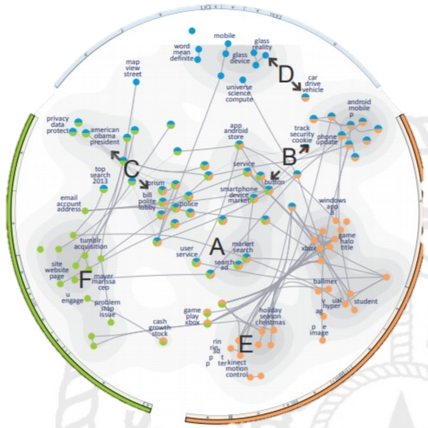
costly mistakes

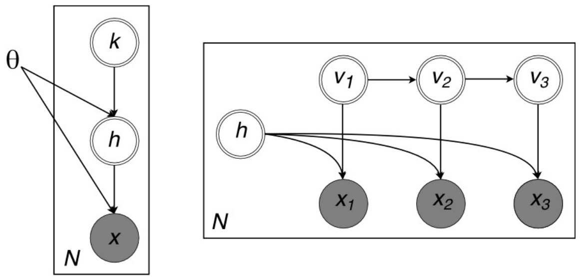
Prior knowledge: deconstruct the black-box of the model

Tell the machine what we know and let them focus on what we do not know.









The success of probabilistic (Bayesian) modeling

Uncertainty

Prior belief

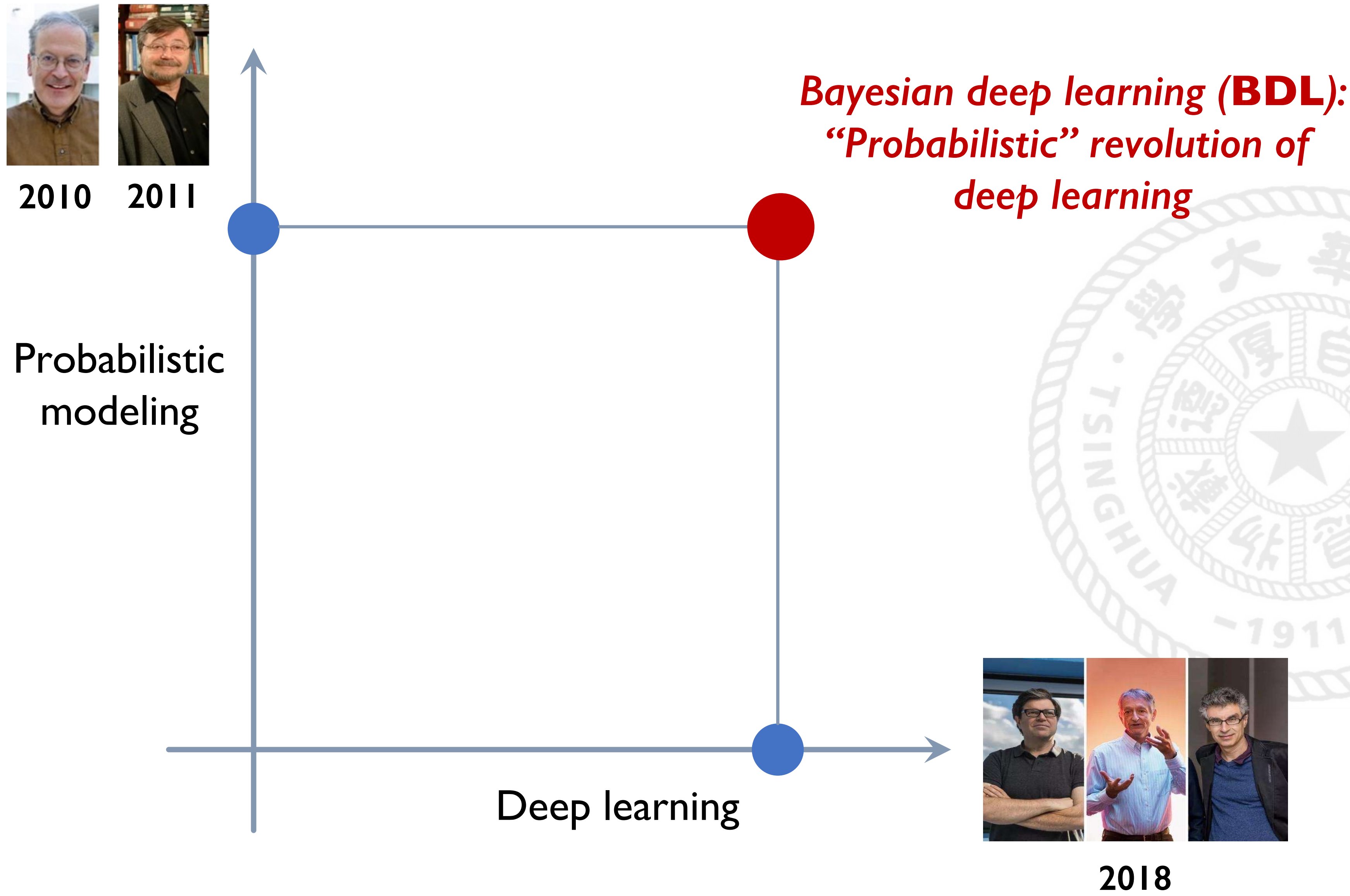
$$p(\theta|D) = \frac{p(D|\theta)\pi(\theta)}{p(D)}$$

Thomas Bayes (1702 – 1761)

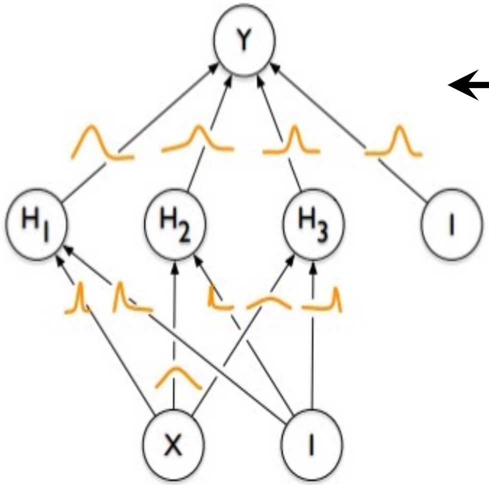
- Prediction: $p(\mathrm{x}|D,H) = \int p(\mathrm{x}|\theta,D,H)p(\theta|D)d\theta$
- Model selection $p(D|H_1) \geq ?\,(or \leq ?)\; p(D|H_2) \quad p(D|H) = \int p(D|\theta)p(\theta|H)d\theta$
- Regularization $\max_{\mathbf{w}} \; \log p(\mathbf{y}|X,\mathbf{w}) + \log p(\mathbf{w})$
- Modeling with latent var. $p(\mathrm{z}|\mathrm{x}) = \frac{p(\mathrm{x},\mathrm{z})}{p(\mathrm{x})} \propto p(\mathrm{z})p(\mathrm{x}|\mathrm{z};\theta)$

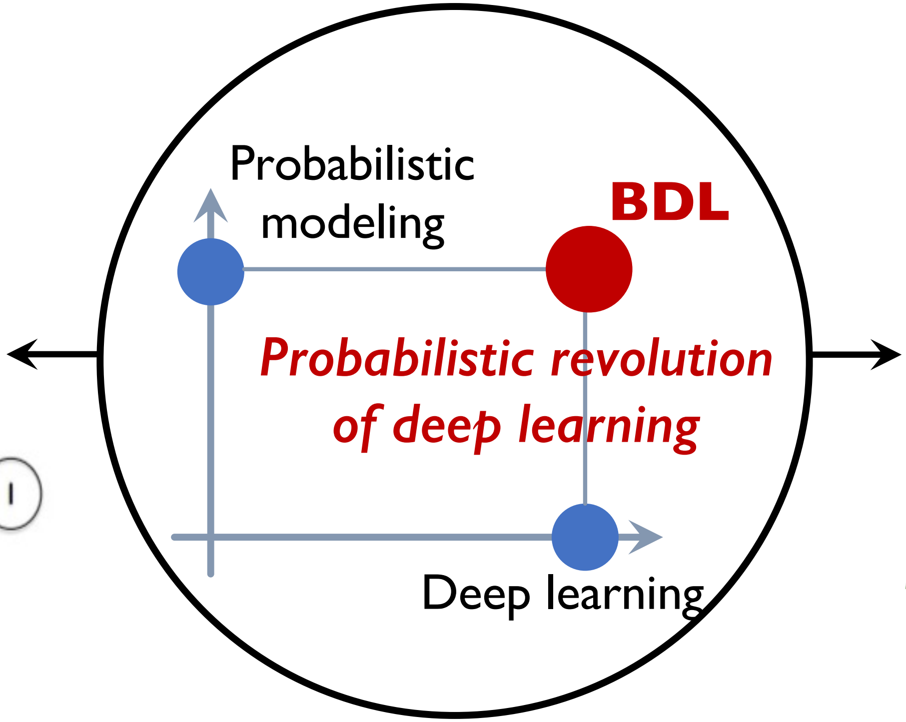
Research focus: Bayesian deep learning (BDL)

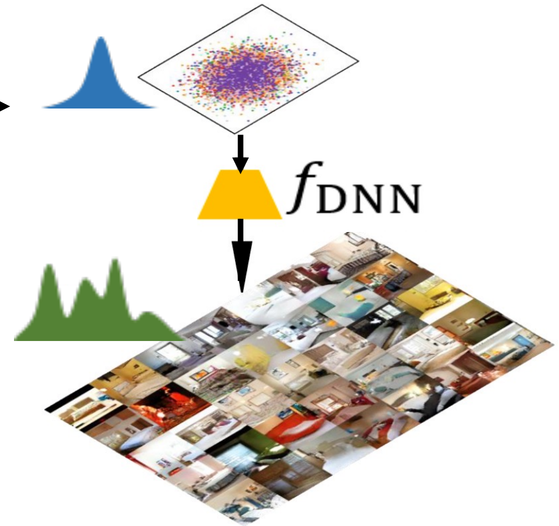
Probabilistic modeling meets deep learning



Research focus: Bayesian deep learning (BDL)



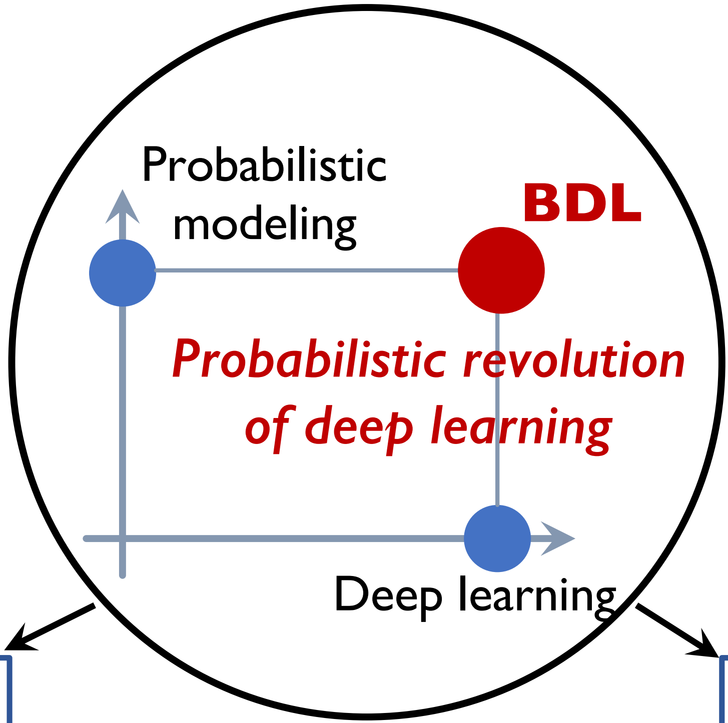




Probabilistic modeling of DNNs :
Bayesian neural networks (BNNs)
Benefits: uncertainty estimation

DNNs enrich probabilistic models: deep generative models (DGMs)
Benefits: incorporating prior knowledge

Research focus: Bayesian deep learning (BDL)



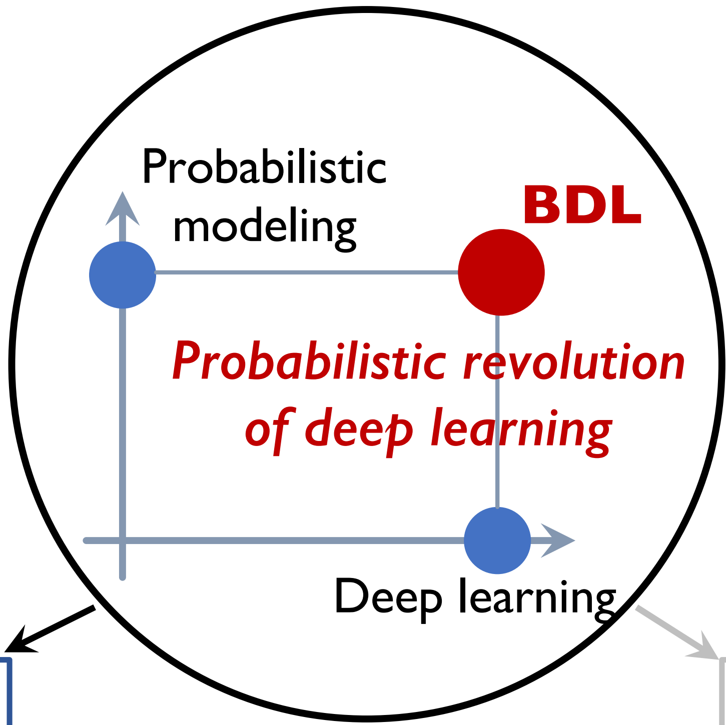
Theme I: **Uncertainty quantification via BNNs**

(NeurIPS 2020; CVPR 2021; ICLR 2021 workshop; NeurIPS sub.)

Theme II: **Knowledge incorporation via DGMs**

(NeurIPS 2017; ICCV 2019, NeurIPS 2020; ICML 2019 workshop)

Research focus: Bayesian deep learning (BDL)



Theme I: **Uncertainty quantification via BNNs**

(NeurIPS 2020; CVPR 2021; ICLR 2021 workshop; NeurIPS sub.)

Theme II: Knowledge incorporation via DGMs

(NeurIPS 2017; ICCV 2019; NeurIPS 2020; ICML 2019 workshop)

Uncertainty is ubiquitous

Road conditions

Traffics

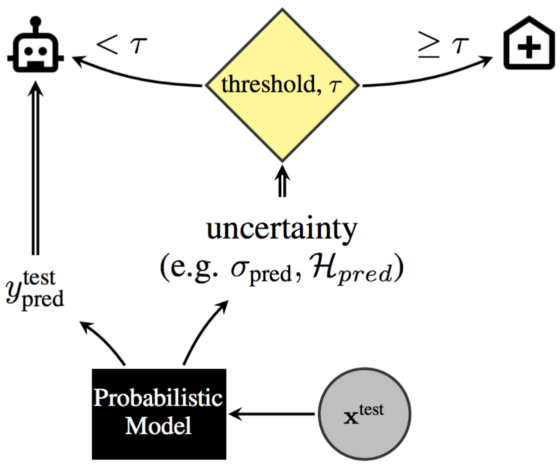
Pedestrian behaviors

Even **malicious**

AlexNet: lionfish, confidence 81.3%
VGG-16: lionfish, confidence 93.3%
ResNet-18: lionfish, confidence 95.6%

Uncertainty is the key to bringing DL to the masses

Automated diagnosis

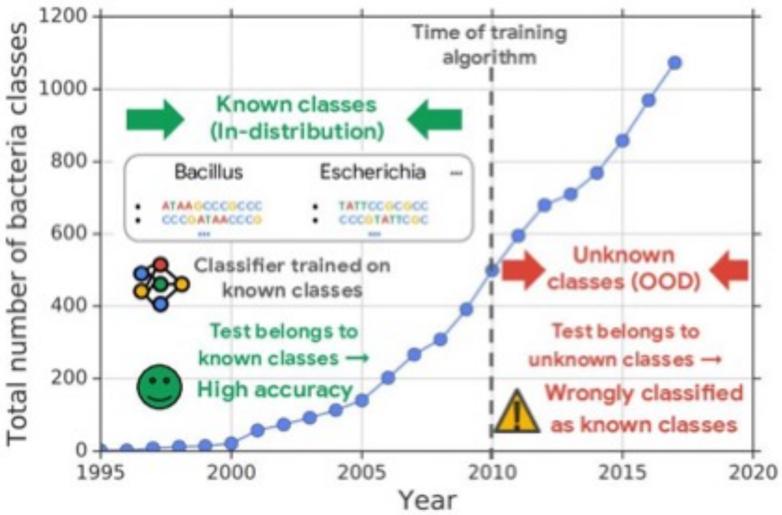


Self-driving cars

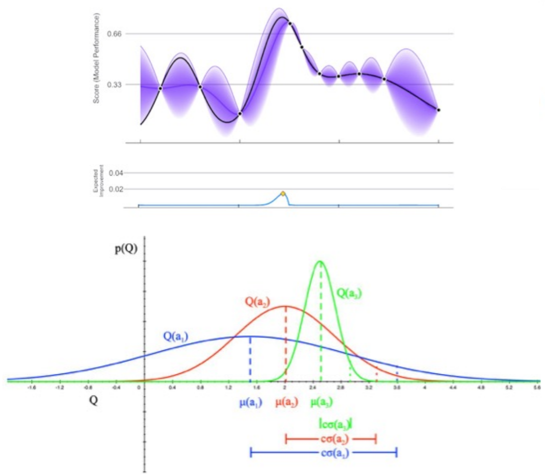
Conversational dialog systems

Figure 1: Example exchanges between a user (blue, right side) and a task-driven dialog system for personal finance (grey, left side). The system correctly identifies the user's query in 1, but in 2 the user's query is mis-identified as in-scope, and the system gives an unrelated response. In 3 the user's query is correctly identified as out-of-scope and the system gives a fallback response.

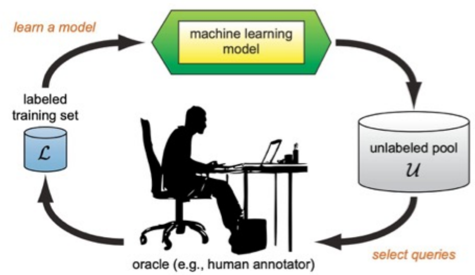
Open set recognition



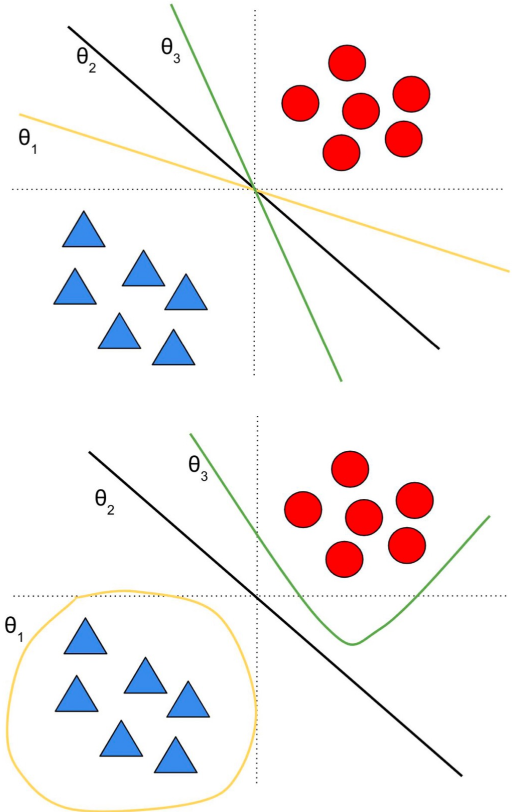
Bayesian optimization and reinforcement learning

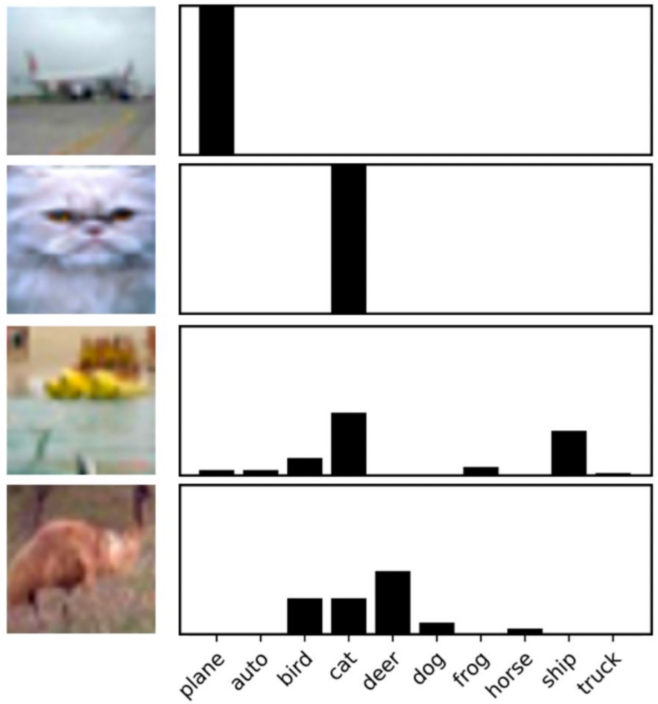


Active Learning



Types of Bayesian uncertainty



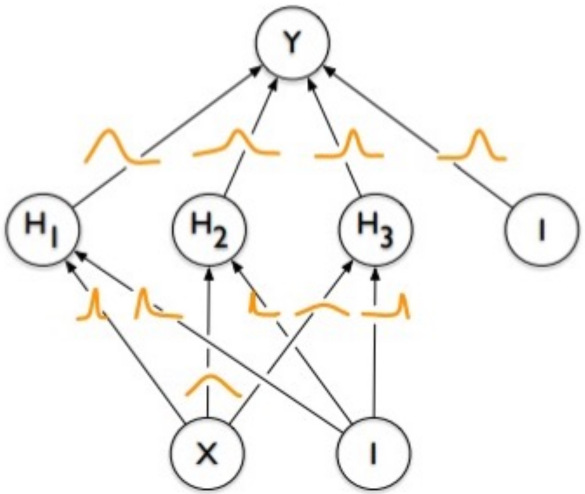


- Model (epistemic) uncertainty
- Various interpretations for the data
- Reducible
- Models can be from same hypotheses class or not

- Data (aleatoric) uncertainty
- Stem from labeling noise, measurement noise, or missing data
- Irreducible*

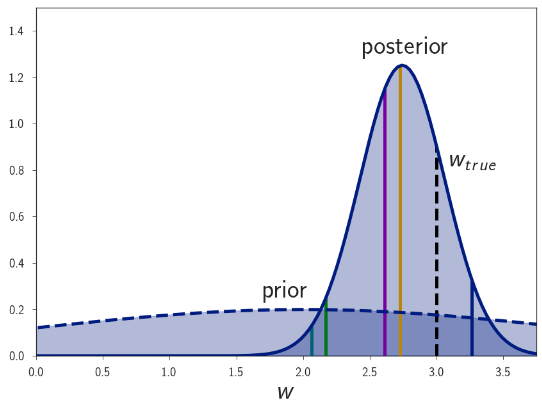
Bayesian uncertainty in DL
Bayesian neural networks (BNNs)

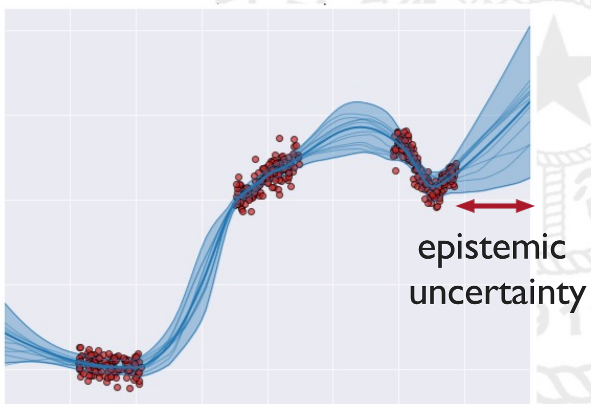
Bayesian treatment of DNNs (weights) captures uncertainty



$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

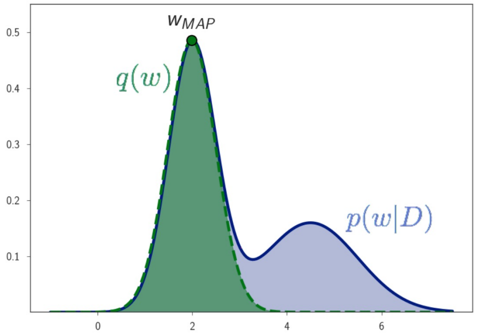
$$p(y^*|x^*, D) = \int_w p(y^*|x^*, w)p(w|D)dw$$



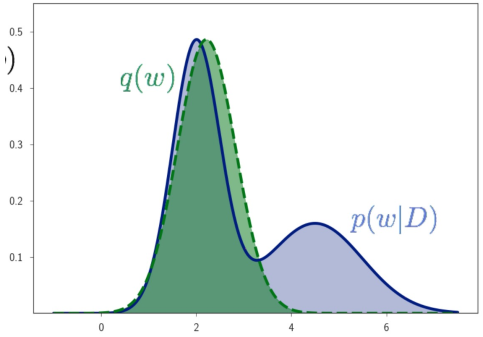


Core of BNNs: posterior inference

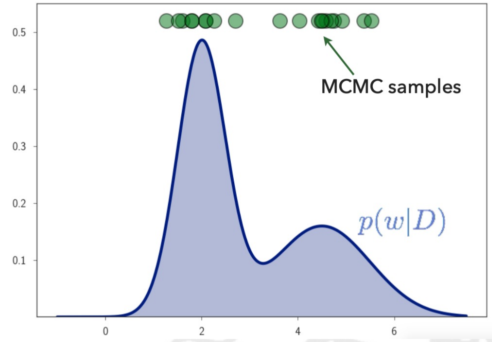
Methods and challenges



Laplace approx.



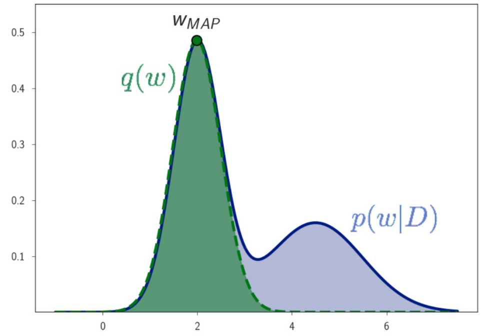
Variational inference



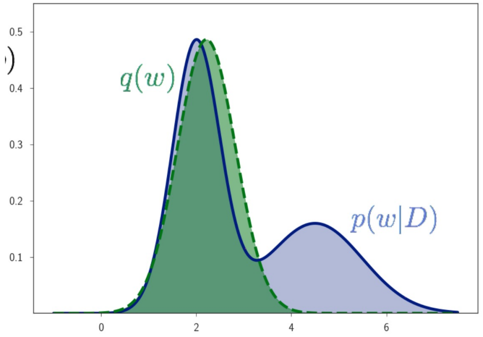
MCMC

Core of BNNs: posterior inference

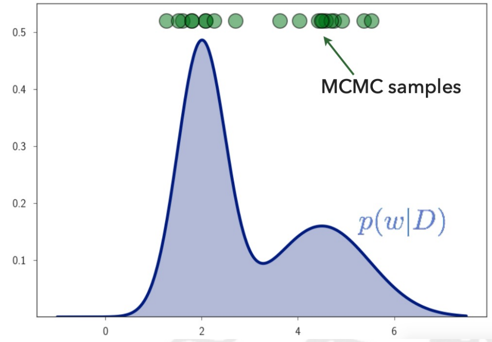
Methods and challenges



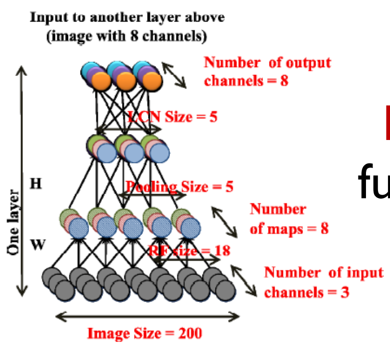
Laplace approx.



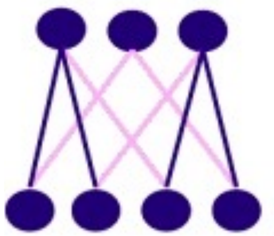
Variational inference



MCMC



High-dim. weight space poses fundamental obstacles for from-scratch inference



Over-parameterization nature of DNNs leads to collapsed weight uncertainty

Existing BNNs

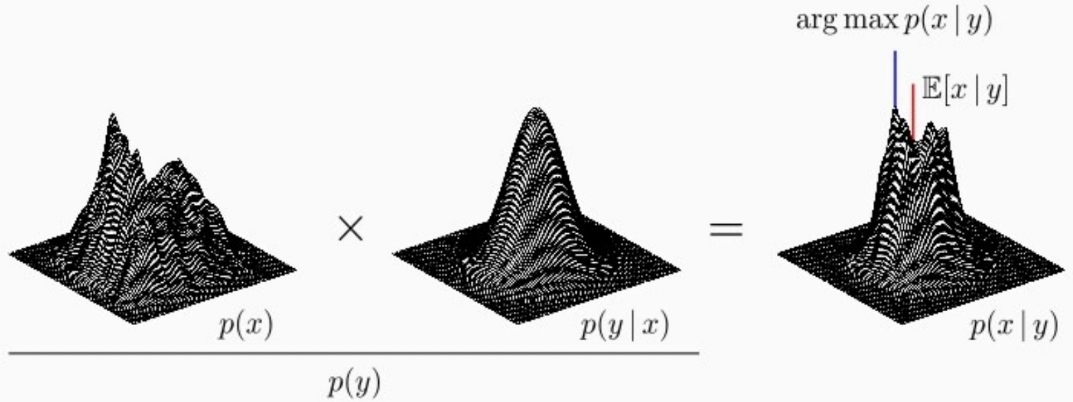
- *Poor performance*
- *Less calibrated uncertainty estimates*
- *Poor scalability*

The Bayesian viewpoint of deterministic training

Maximum a Posteriori (MAP)

- Take a mode of the posterior, or **Maximum a Posteriori**

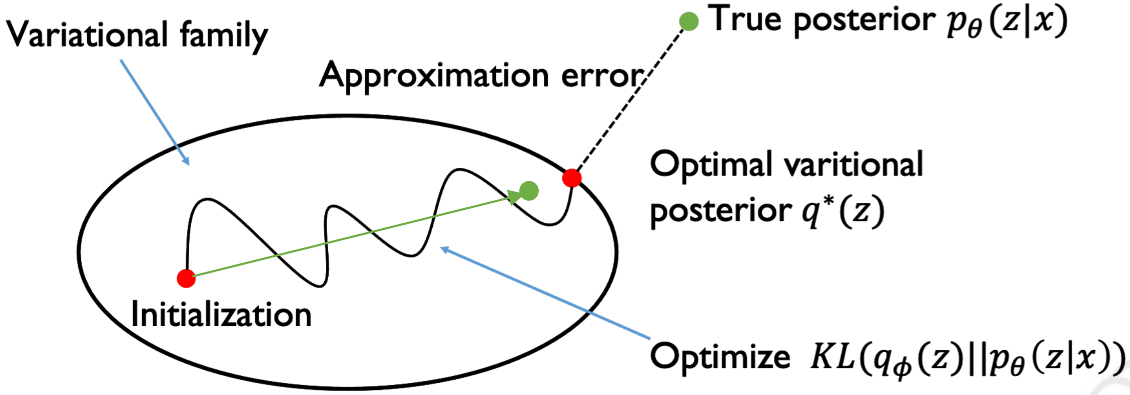
$$x^\star \in \underset{x}{\operatorname{argmax}}\, p(x \mid y)$$

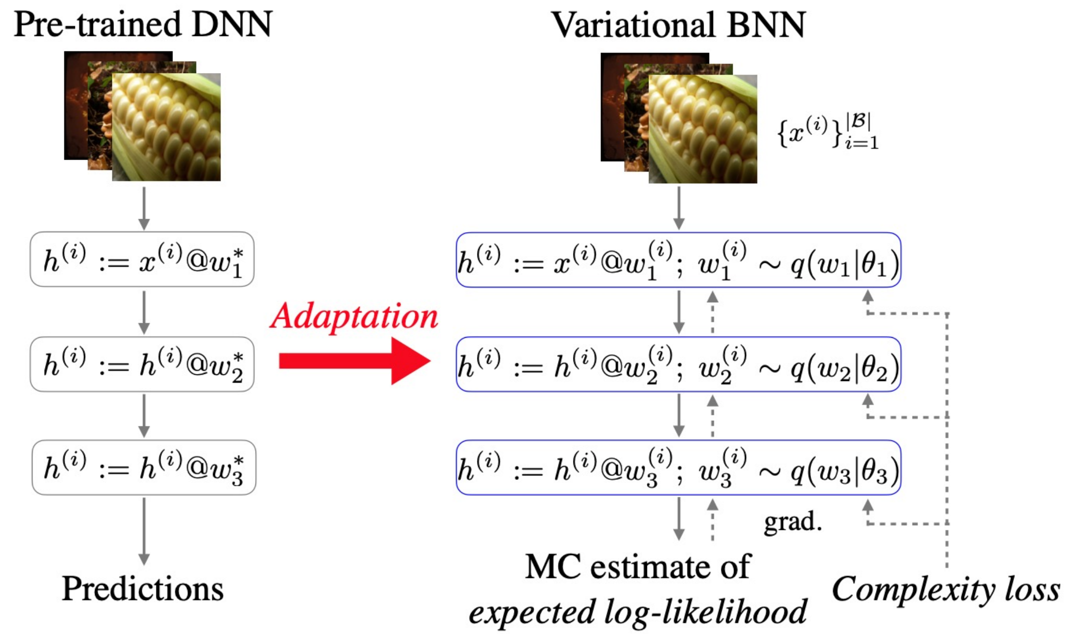


MAP: $$\max_{\boldsymbol{w}} \frac{1}{n} \sum_i [\log p(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{w})] + \frac{1}{n} \log p(\boldsymbol{w})$$

Variational inference (VI): $$\max_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{q(\boldsymbol{w}|\boldsymbol{\theta})} [\frac{1}{n} \sum_i \log p(y^{(i)} | \boldsymbol{x}^{(i)}; \boldsymbol{w})]}_{\mathcal{L}_{ell}} \underbrace{- \frac{1}{n} D_{\mathrm{KL}} \left(q(\boldsymbol{w}|\boldsymbol{\theta}) \| p(\boldsymbol{w})\right)}_{\mathcal{L}_c}.$$

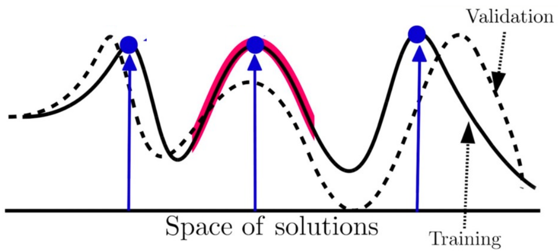
BayesAdapter: variational inference by *Bayesian fine-tuning*
[Deng et al., NeurIPS sub.; Deng et al., CVRP 2021]

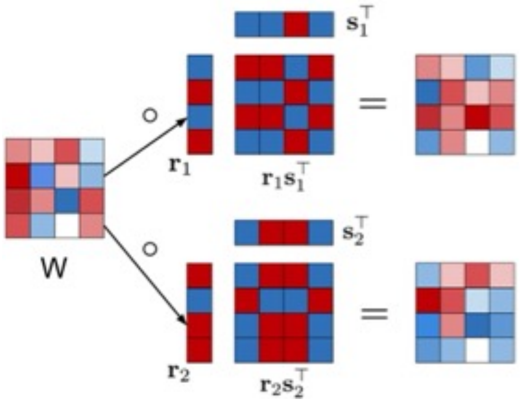




BayesAdapter: variational inference by *Bayesian fine-tuning*

[Deng et al., NeurIPS sub.; Deng et al., CVRP 2021]






```
def BayesAdapter_conv(x, theta, stride, padding, groups):
    b = x.shape[0]
    # sample a batch of parameters w: [b, o, i, k, k]
    w = mc_sample(theta, num_mc_samples=b)
    # reshape w to have shape [b*o, i, k, k]
    w = w.flatten(start_dim=0, end_dim=1)
    # reshape x to have shape [1, b*i, h, w]
    x = x.flatten(start_dim=0, end_dim=1).unsqueeze(0)
    # perform b convs in parallel
    y = conv2d(x, w, stride, padding, groups*b)
    # reshape the result to standard format
    return y.view(b, -1, y.shape[2], y.shape[3])
```

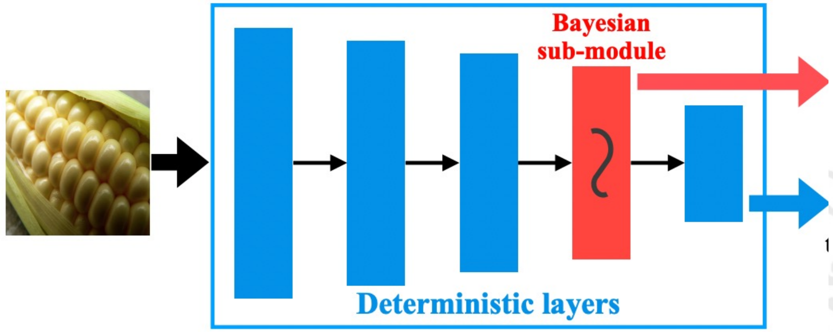
- To capture the multi-mode DNN posterior
  - mixture of delta (Gaussian) variational
- To maintain parameter efficiency
  - parameter sharing
- To reduce the variance of stochastic gradients
  - Exemplar reparameterization

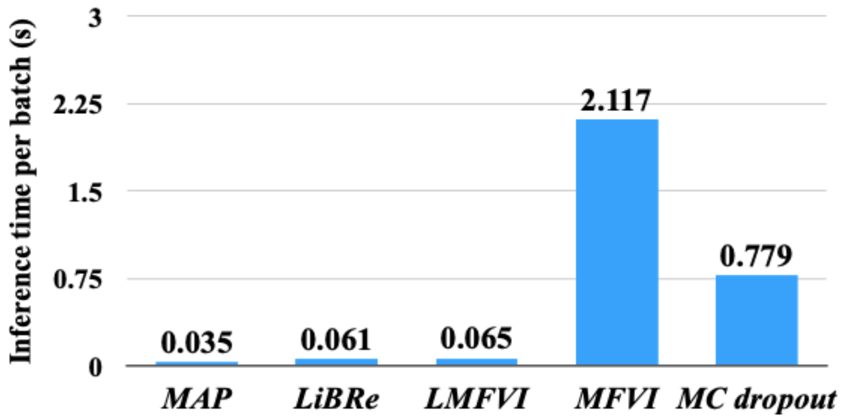
BayesAdapter: variational inference by *Bayesian fine-tuning*

[Deng et al., NeurIPS sub.; Deng et al., CVRP 2021]

Make Bayesian modeling *lightweight*

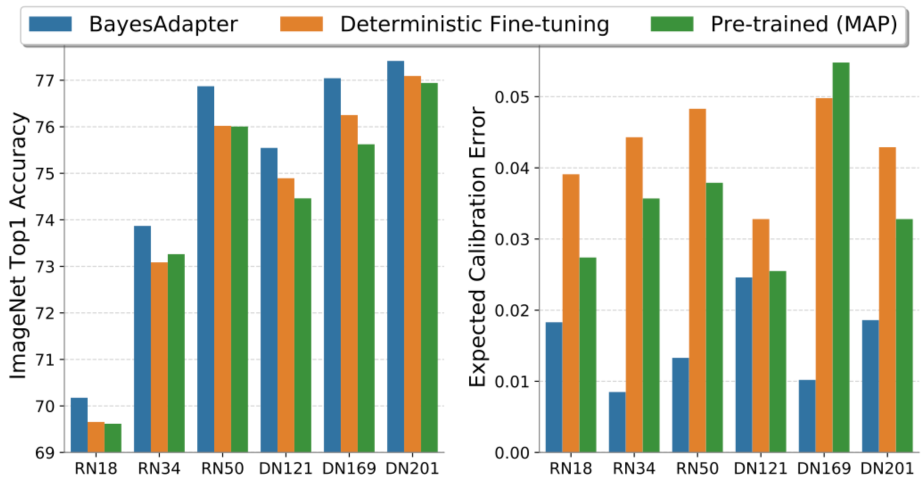
- A practical and theoretical sound BDL approach
- Need minimal added training cost
- Promising Bayesian model average speed



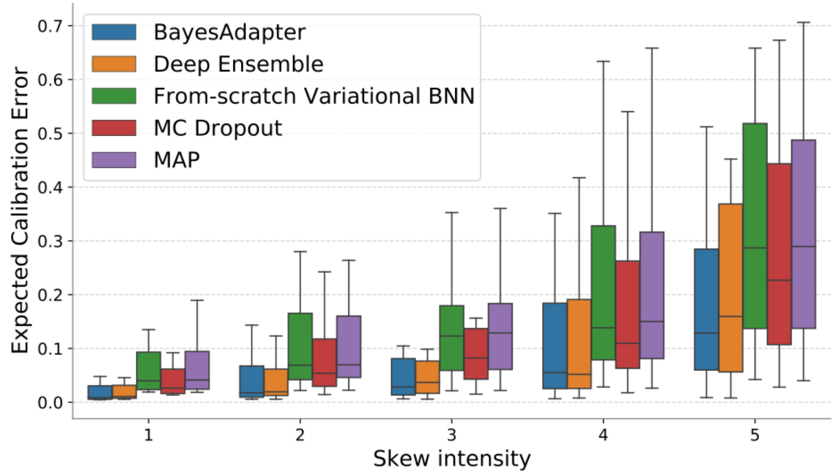


(a) Inference speed comparison

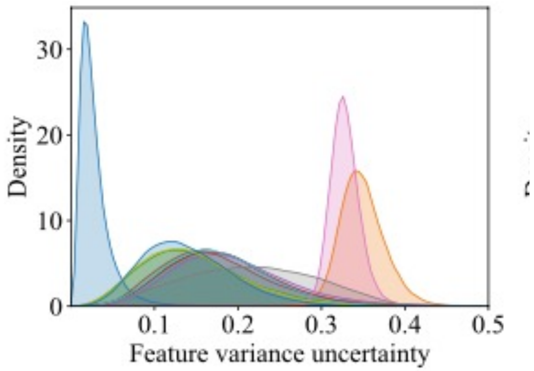
BayesAdapter: results



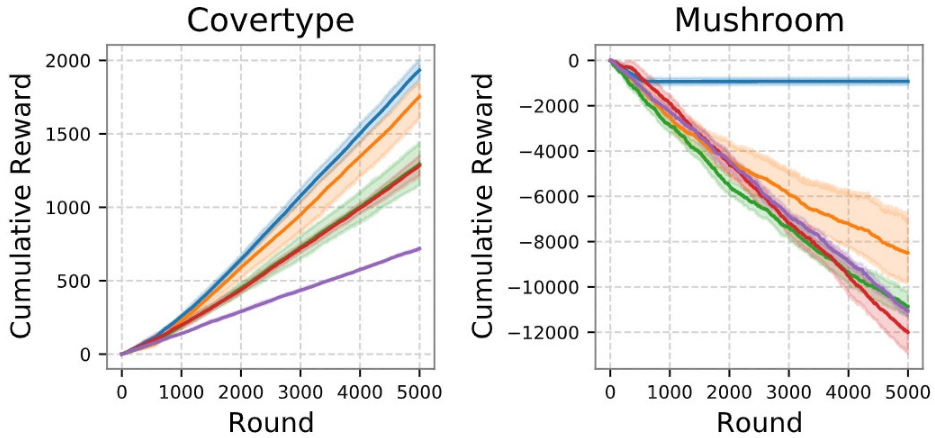
Bayesian model ensemble: one of the **first** variational BNNs that beat DNNs on *ImageNet*



OOD robustness (resistance to over-confidence)

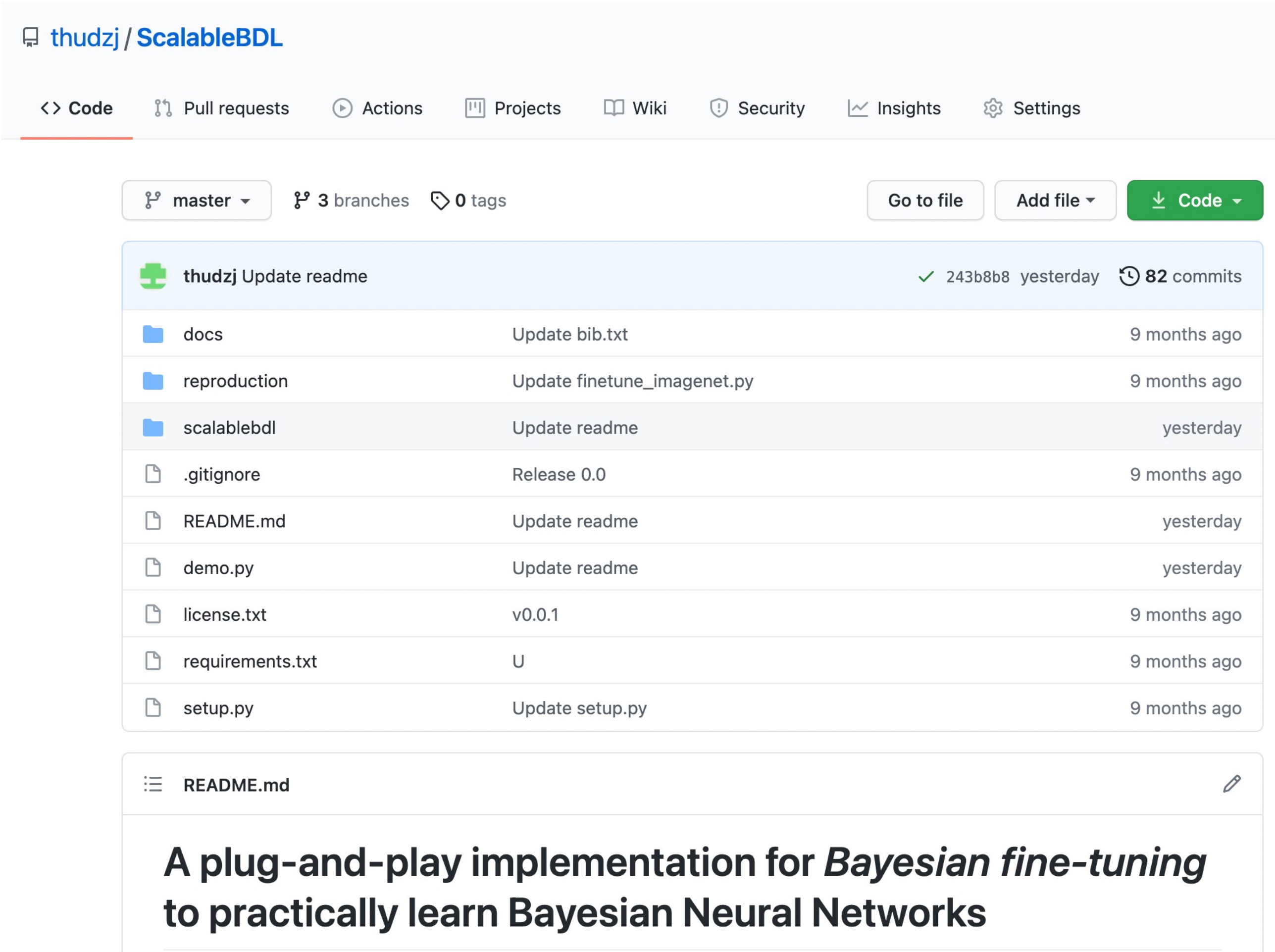


Uncertainty based detection of adversarial examples



Exploration in reinforcement learning (contextual bandit)

BayesAdapter: a Python library



Uncertainty over the DNN structure?

Is there a more scalable alternative to the weight uncertainty?

On weights

- Hard to specifying sensible priors
- Using flexible variational posterior for high-dim weights is expensive
- Over parameterization nature of DNNs may lead to degenerated weight posterior

VS

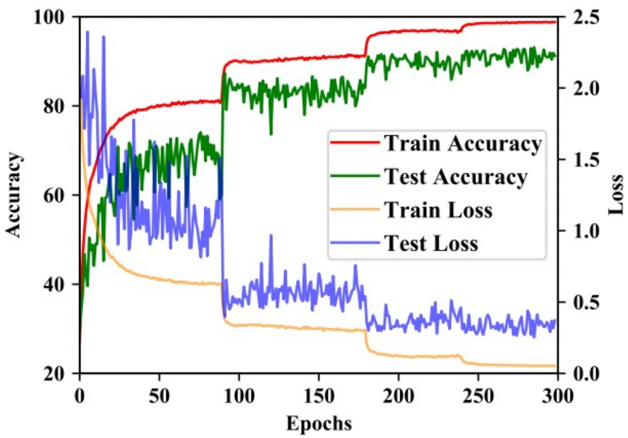
On structure

- Impose prior beliefs more explicitly
- As shown by NAS, the network structure can be defined in a compact manner
- Learning network structure can boost performance

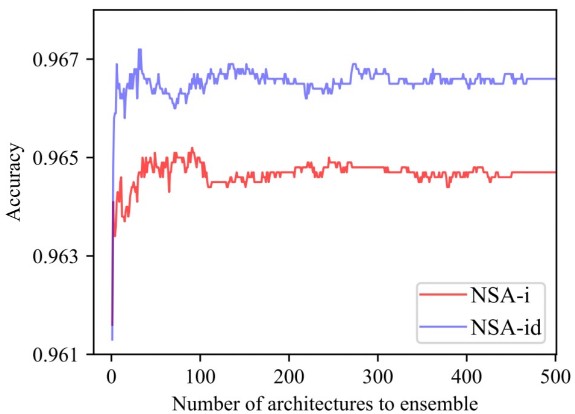
Structure uncertainty: a new BNN paradigm

Deng et al., NeurIPS 2020; Deng et al., ICLR 2021 NAS workshop

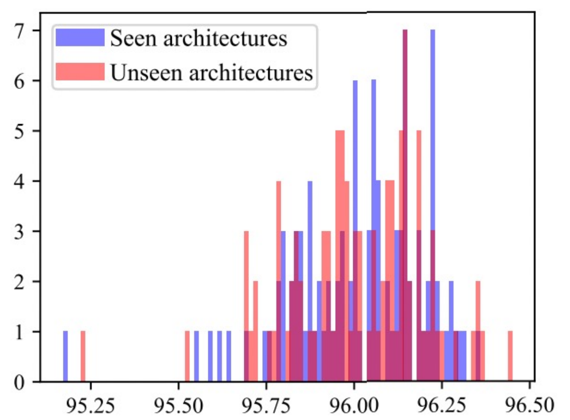
- We pre-specify the structural uncertainty and perform a **first** investigation/understanding on DNNs with such structure uncertainty



training/test disparity



function mode collapse

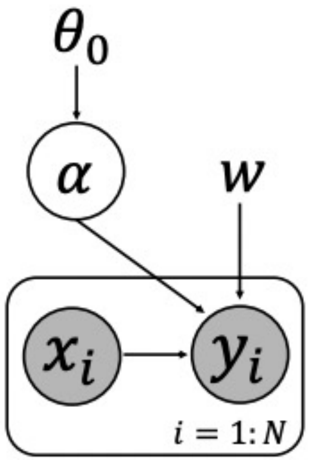


generalization

Structure uncertainty: a new BNN paradigm

Deng et al., NeurIPS 2020; Deng et al., ICLR 2021 NAS workshop

Structure uncertainty meets the advance in NAS



- Assume priors and define variationals:

$$p(\boldsymbol{\alpha}, \boldsymbol{w}) = p(\boldsymbol{\alpha})p(\boldsymbol{w}) \qquad q(\boldsymbol{\alpha}, \boldsymbol{w}) = q(\boldsymbol{\alpha}|\boldsymbol{\theta})\delta(\boldsymbol{w} - \boldsymbol{w}_0)$$

$$p(\boldsymbol{\alpha}) = \prod_{i<j} p(\boldsymbol{\alpha}^{(i,j)}) \qquad q(\boldsymbol{\alpha}|\boldsymbol{\theta}) = \prod_{i<j} q(\boldsymbol{\alpha}^{(i,j)}|\boldsymbol{\theta}^{(i,j)})$$

- A unified training objective (**ELBO**):

$$\min_{q\in\mathcal{Q}} D_{\mathrm{KL}}(q(\boldsymbol{\alpha}, \boldsymbol{w})\|p(\boldsymbol{\alpha}, \boldsymbol{w}|\mathcal{D})) = -\mathbb{E}_{q(\boldsymbol{\alpha},\boldsymbol{w})}[\log p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{w})] + D_{\mathrm{KL}}(q(\boldsymbol{\alpha}, \boldsymbol{w})\|p(\boldsymbol{\alpha}, \boldsymbol{w})) + constant.$$

- Continuous relaxation and reparameterization

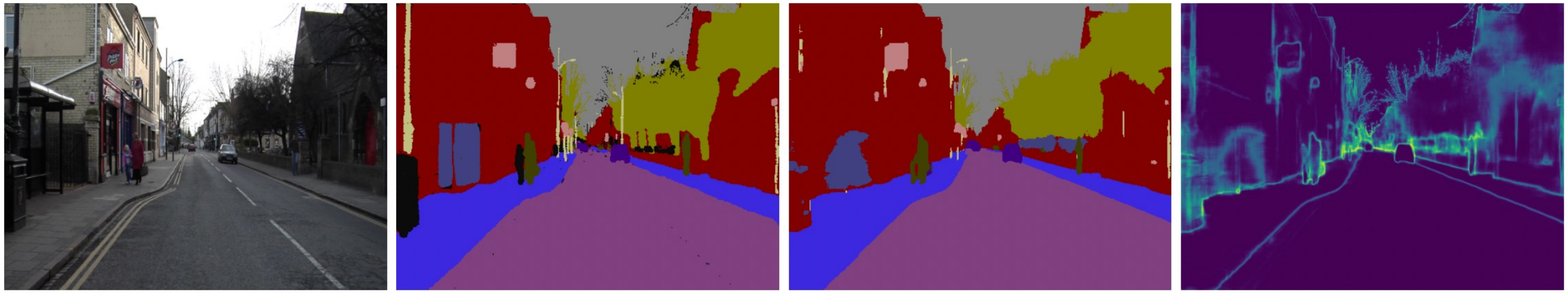
$$\boldsymbol{\alpha}^{(i,j)} = g(\boldsymbol{\theta}^{(i,j)}, \boldsymbol{\beta}^{(i,j)}, \boldsymbol{\epsilon}^{(i,j)}) = \mathrm{softmax}((\boldsymbol{\theta}^{(i,j)} + \boldsymbol{\beta}^{(i,j)}\boldsymbol{\epsilon}^{(i,j)})/\tau).$$

- "Cold posterior": sharpened concrete distribution

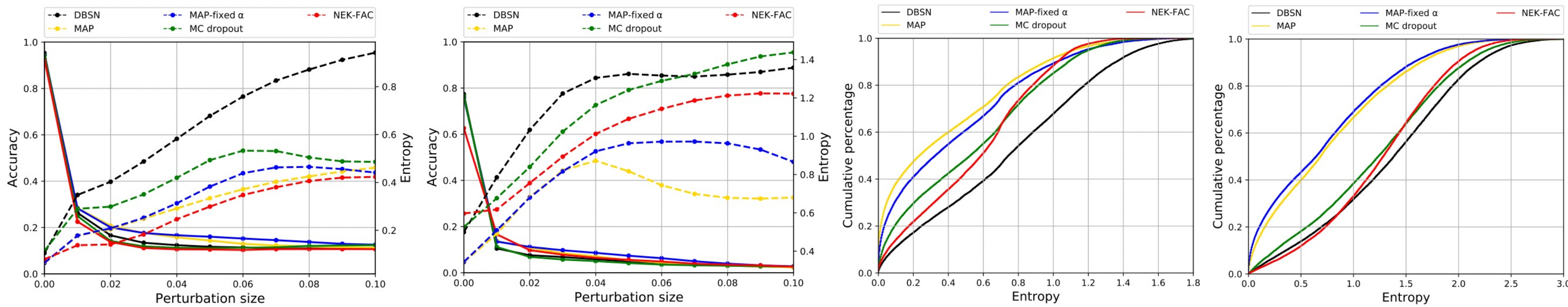
Structure uncertainty: results

Method	DBSN	MAP	MAP-fixed α	MC dropout	BBB	FBN	NEK-FAC
CIFAR-10	**0.0109**	0.0339	0.0327	0.0150	0.0745	0.0966	0.0434
CIFAR-100	**0.0599**	0.1240	0.1259	0.0617	0.0700	0.1091	0.1665

Less over-confidence than BNNs with weight uncertainty



Meaningful uncertainty estimates in semantic segmentation problems

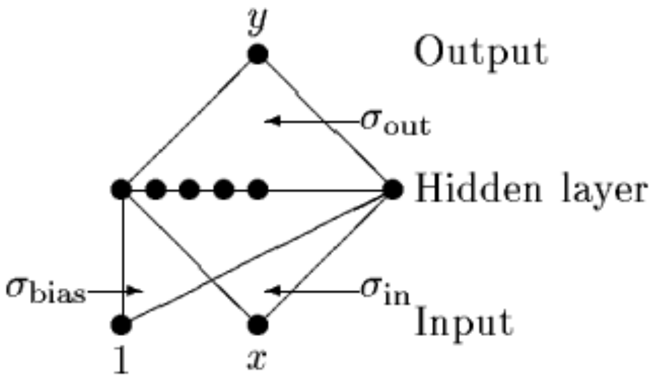


Better OOD robustness

BNNs are Gaussian processes (GPs) in the width limit

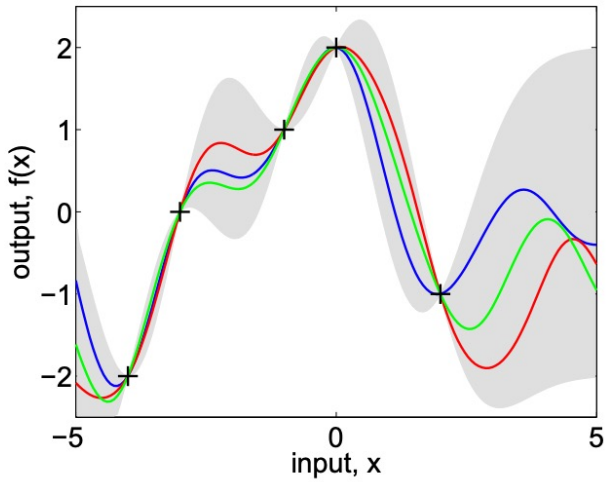
An exciting perspective

Neal, 1995; Lee et al., 2017



$width \rightarrow \infty$

$\Longrightarrow$



$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim \mathcal{N}\big(\bar{\mathbf{f}}_*,\ \mathrm{cov}(\mathbf{f}_*)\big), \text{ where}$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y},$$

$$\mathrm{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1} K(X, X_*)$$

Allows us to understand neural networks (e.g. generalization properties) without practically training them

Can deep ensemble be understood in this spirit?

Deep ensemble:

- One of the most performant prediction & uncertainty modeling approaches
- Lack a proper Bayesian justification

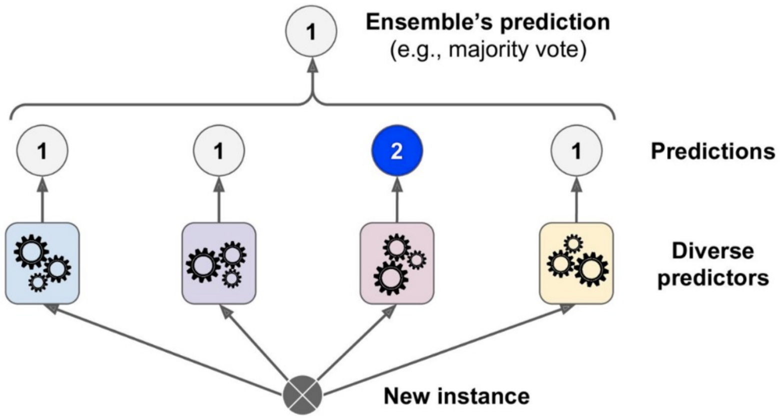


Figure 7-2. Hard voting classifier predictions

$\Longrightarrow$?

Deep ensemble defines a GP posterior
Deng et al., NeurIPS sub.

The form: $q(f|\boldsymbol{w}_1, ..., \boldsymbol{w}_M) = \mathcal{GP}(f|m_q(\boldsymbol{x}), k_q(\boldsymbol{x}, \boldsymbol{x}')),$

$$m_q(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} g(\boldsymbol{x}, \boldsymbol{w}_i),$$

$$k_q(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{M}\sum_{i=1}^{M} \left(g(\boldsymbol{x}, \boldsymbol{w}_i) - m_q(\boldsymbol{x})\right)\left(g(\boldsymbol{x}', \boldsymbol{w}_i) - m_q(\boldsymbol{x}')\right)^T + \lambda \mathbf{I}_C.$$

Deep ensemble defines a GP posterior
Deng et al., NeurIPS sub.

The form: $q(f|\boldsymbol{w}_1, ..., \boldsymbol{w}_M) = \mathcal{GP}(f|m_q(\boldsymbol{x}), k_q(\boldsymbol{x}, \boldsymbol{x}')),$

$$m_q(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} g(\boldsymbol{x}, \boldsymbol{w}_i),$$

$$k_q(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{M}\sum_{i=1}^{M} \left(g(\boldsymbol{x}, \boldsymbol{w}_i) - m_q(\boldsymbol{x})\right)\left(g(\boldsymbol{x}', \boldsymbol{w}_i) - m_q(\boldsymbol{x}')\right)^T + \lambda \mathbf{I}_C.$$

Bayesian inference in function space: theorem on the functional ELBO:

$$\begin{aligned}\mathcal{L}' &= \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{D}} \mathbb{E}_{q(f)}[\log p(\boldsymbol{y}_i|f(\boldsymbol{x}_i))] - D_{\mathrm{KL}}[q(\mathbf{f}^{\tilde{\mathbf{X}}})\|p(\mathbf{f}^{\tilde{\mathbf{X}}})] \\ &= \log p(\mathcal{D}) - D_{\mathrm{KL}}[q(\mathbf{f}^{\tilde{\mathbf{X}}})\|p(\mathbf{f}^{\tilde{\mathbf{X}}}|\mathcal{D})] \le \log p(\mathcal{D}),\end{aligned}$$

Deep ensemble defines a GP posterior

Deng et al., NeurIPS sub.

The form: $q(f|\boldsymbol{w}_1, ..., \boldsymbol{w}_M) = \mathcal{GP}(f|m_q(\boldsymbol{x}), k_q(\boldsymbol{x}, \boldsymbol{x}')),$

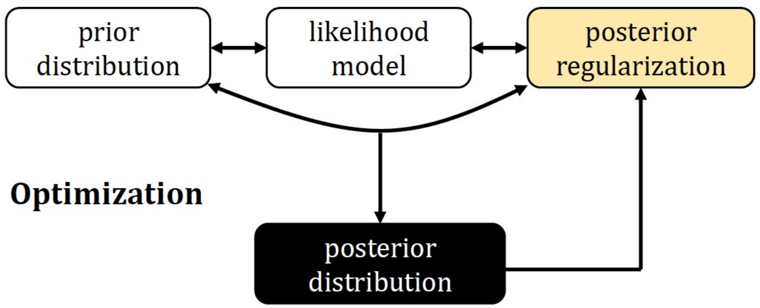
$$m_q(\boldsymbol{x}) = \frac{1}{M}\sum_{i=1}^{M} g(\boldsymbol{x}, \boldsymbol{w}_i),$$

$$k_q(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{M}\sum_{i=1}^{M} \left(g(\boldsymbol{x}, \boldsymbol{w}_i) - m_q(\boldsymbol{x})\right)\left(g(\boldsymbol{x}', \boldsymbol{w}_i) - m_q(\boldsymbol{x}')\right)^T + \lambda \mathbf{I}_C.$$

Bayesian inference in function space: theorem on the functional ELBO:

$$\begin{aligned}\mathcal{L}' &= \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i)\in\mathcal{D}} \mathbb{E}_{q(f)}[\log p(\boldsymbol{y}_i|f(\boldsymbol{x}_i))] - D_{\mathrm{KL}}[q(\mathbf{f}^{\tilde{\mathbf{X}}})\|p(\mathbf{f}^{\tilde{\mathbf{X}}})] \\ &= \log p(\mathcal{D}) - D_{\mathrm{KL}}[q(\mathbf{f}^{\tilde{\mathbf{X}}})\|p(\mathbf{f}^{\tilde{\mathbf{X}}}|\mathcal{D})] \leq \log p(\mathcal{D}),\end{aligned}$$

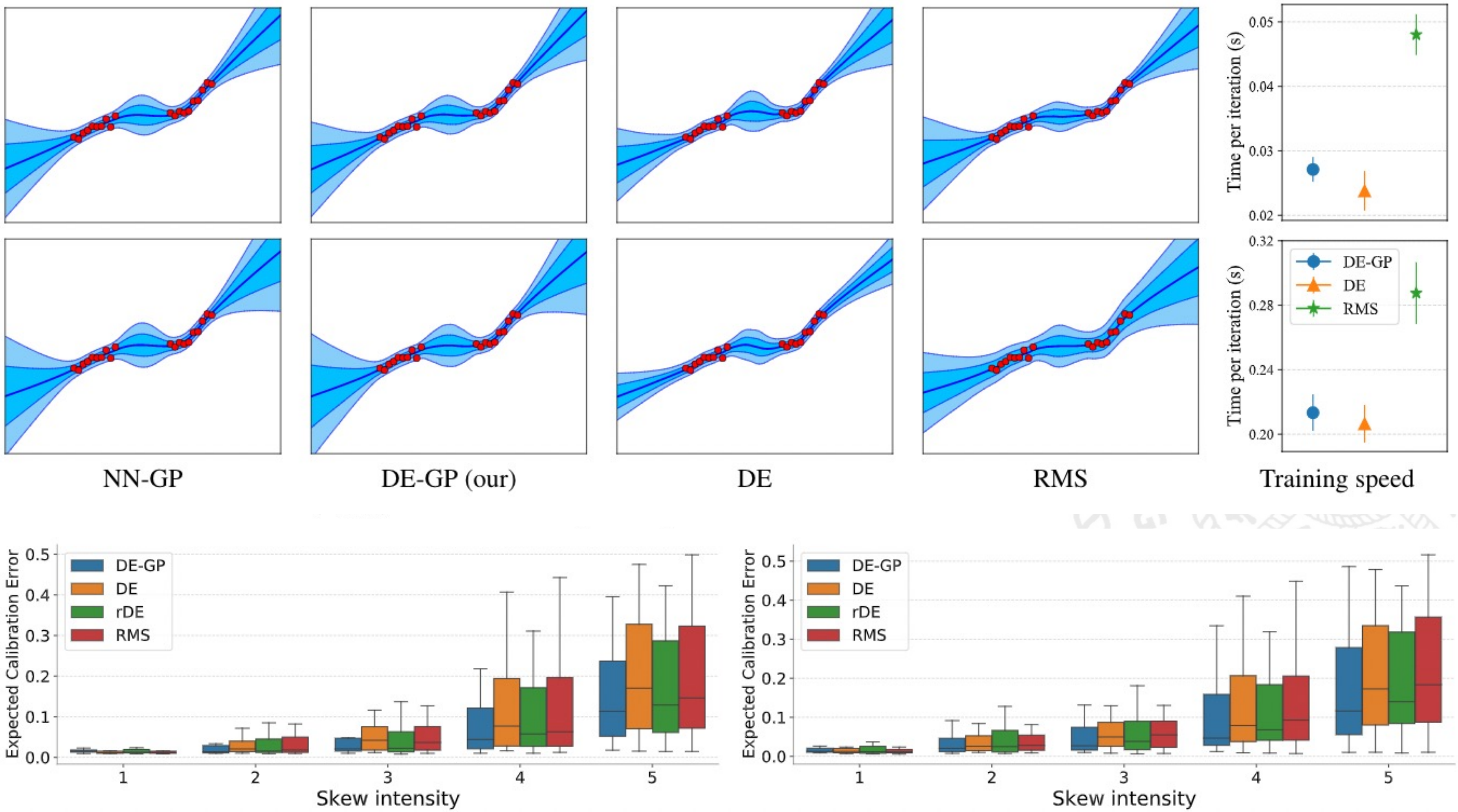
RegBayes:



One can encode *any differentiable constraints* on the functional posterior

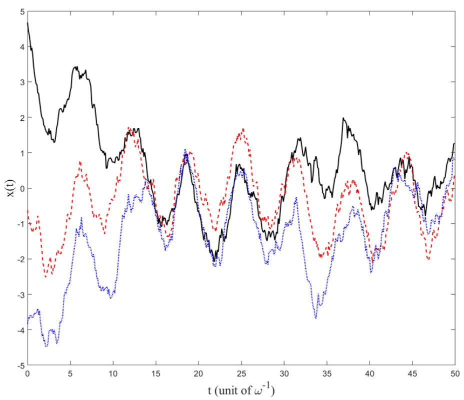
Deep ensemble defines a GP posterior: results

Deng et al., NeurIPS sub.

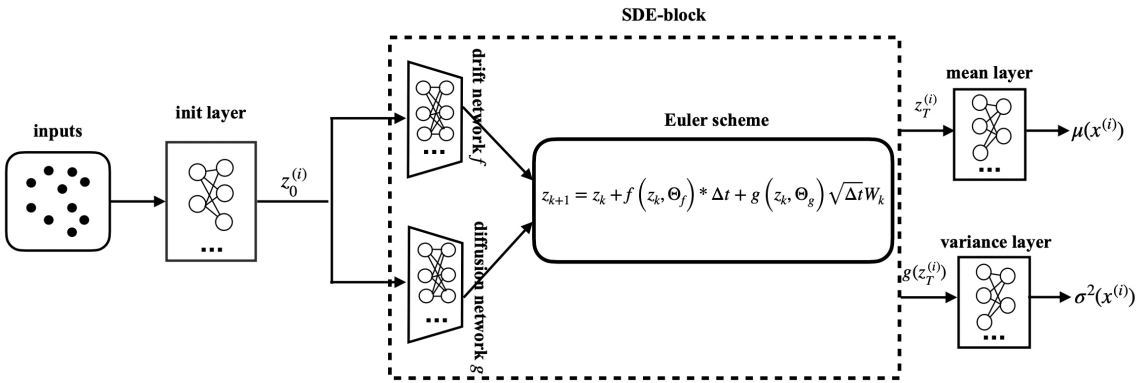


More calibrated/reliable uncertainty estimates then standard deep ensemble

Uncertainty quantification methods beyond BNNs
NeurIPS sub.



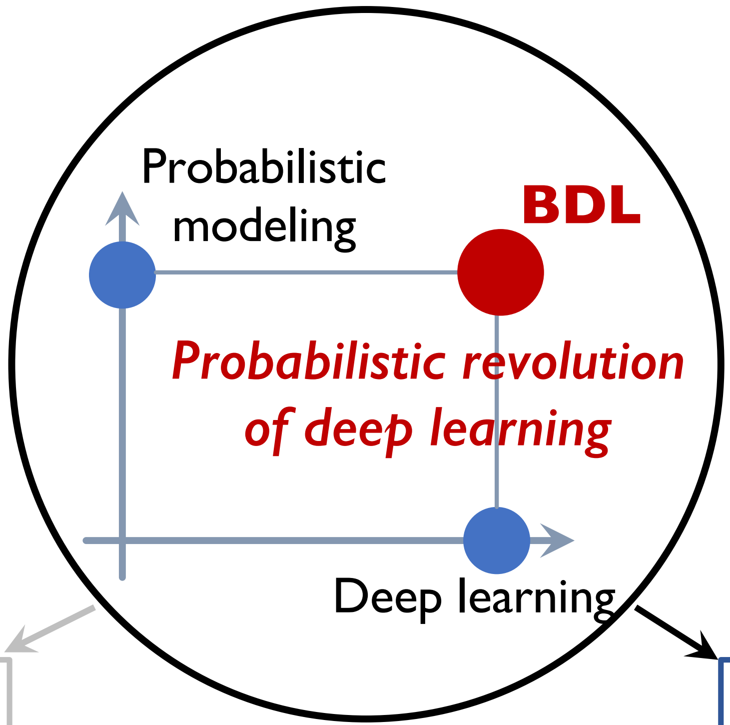
Stochastic differential equations (SDEs)



SDE based heteroscedastic neural networks

Dataset	Metric	MCD	DGP	BNN	Deep-ens	HNN	Proposed
Metro-traffic	RMSE	697.021	651.341	786.694	533.426	559.354	**483.639±2.657**
	$R^2 \uparrow$	0.877	0.892	0.843	0.928	0.920	**0.939±0.011**
	CWCE	52.152	10.552	21.486	9.078	9.305	**2.894±0.085**
	EPIW	~~*167.859*~~	1168.044	610.662	814.143	883.475	**539.254±19.334**
	R-CWCE	6.428	1.136	3.373	0.655	0.747	**0.177±0.014**
Pickups	RMSE	625.812	523.041	720.013	428.032	421.752	**340.331±5.072**
	$R^2 \uparrow$	0.878	0.914	0.838	0.943	0.945	**0.964±0.012**
	CWCE	34.441	22.799	42.570	4.878	6.043	**2.925±0.758**
	EPIW	~~*313.432*~~	1872.481	*247.229*	684.381	688.989	**438.324±19.222**
	R-CWCE	4.205	1.951	6.904	0.280	0.335	**0.173±0.012**

Research focus: Bayesian deep learning (BDL)



Theme I: Uncertainty quantification via BNNs

(NeurIPS 2020; CVPR 2021; ICLR 2021 workshop; NeurIPS sub.)

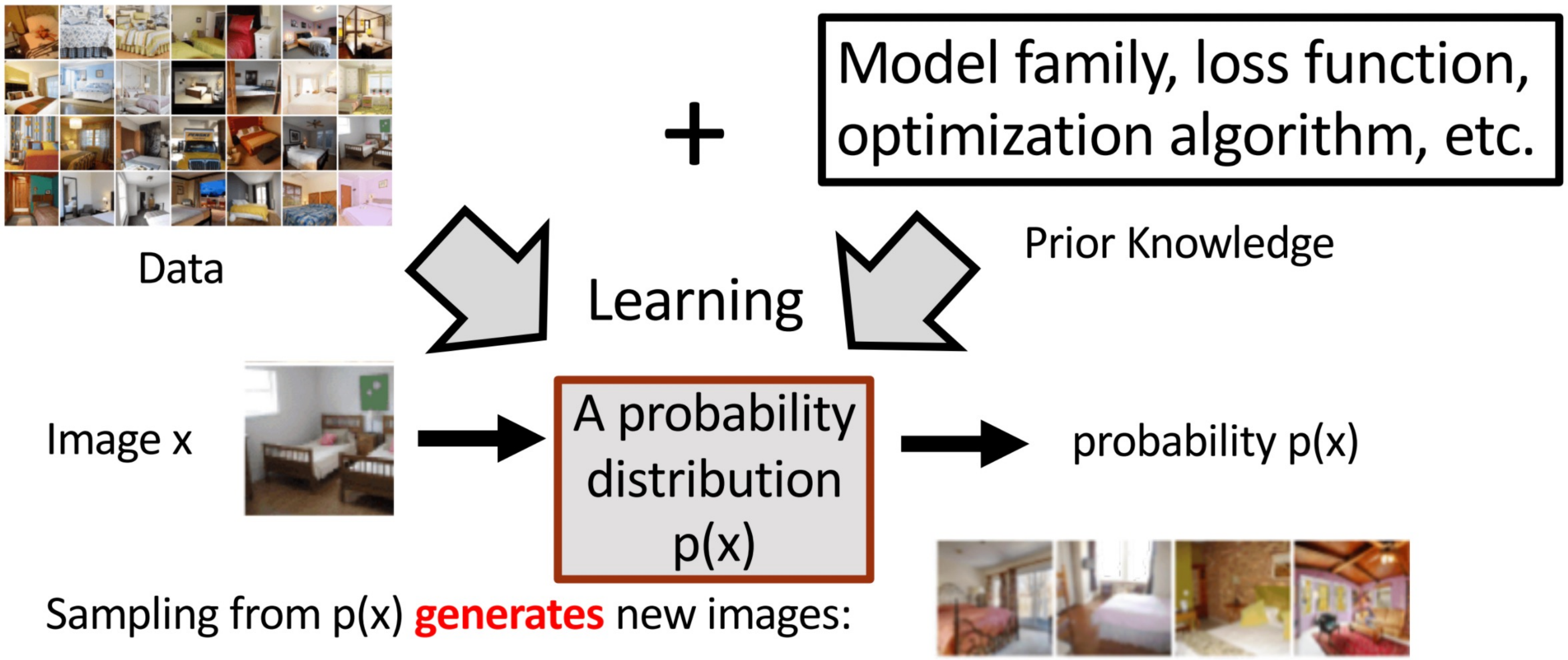
Theme II: Knowledge incorporation via DGMs

(NeurIPS 2017; ICCV 2019, NeurIPS 2020; ICML 2019 workshop)

Deep generative models (DGMs)

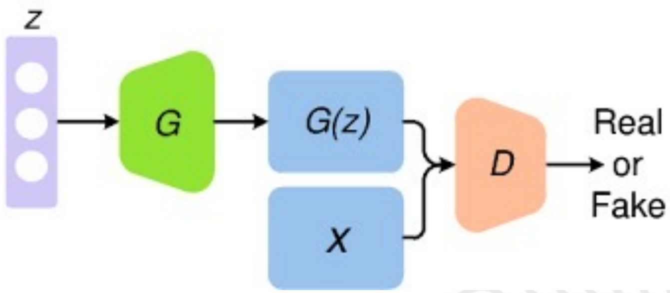
DNNs enrich probabilistic modeling

Richard Feynman: "What I cannot create, I do not understand"



Conditional Generative Adversarial Nets (GANs)

Generative models with implicit density



- GANs -- a two-player minimax game:

$$\min_G \max_D \mathcal{L}(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[1 - \log(D(G(\boldsymbol{z})))],$$

(Minimizing Jensen-Shannon divergence)

- cGANs – label-aware GANs: z - noise, y - label

$$\min_G \max_{D_{\boldsymbol{xy}}} \mathcal{L}_{\boldsymbol{xy}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim p(\boldsymbol{x},\boldsymbol{y})}[\log(D_{\boldsymbol{xy}}(\boldsymbol{x}, \boldsymbol{y}))] + \mathbb{E}_{\boldsymbol{y} \sim p(\boldsymbol{y}), \boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D_{\boldsymbol{xy}}(G(\boldsymbol{y}, \boldsymbol{z}), \boldsymbol{y}))]$$

Modeling the **joint** between data and label

Conditional generative modeling with few labels is non-trivial

SSL meets cGANs

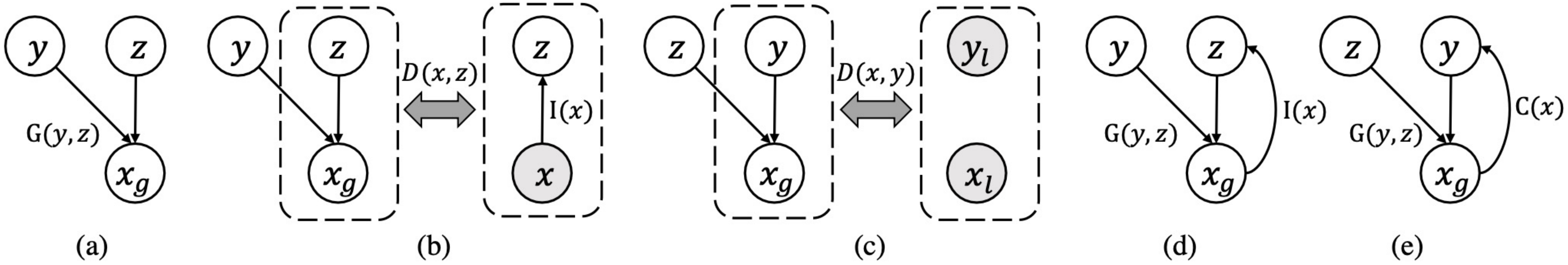
- The conditional generators in existing works exhibit inadequate ***controllability*** – the generator's ability to conditionally generate samples that have structures strictly agreeing with the condition

- Reason: noise z encodes some semantic info., confounding *G*

- Solution: **disentangle** the semantics of our interest and other variations

Structured GANs: cGANs with a structured hidden space

Deng et al., NeurIPS 2017



A simple prior knowledge

$$\min_{C,G} \mathbb{E}_{y\sim p(y)} \| p_c(y|G(y,z_1)), p_c(y|G(y,z_2))\|, \ \forall z_1, z_2 \sim p(z)$$

$$\min_{I,G} \mathbb{E}_{z\sim p(z)} \| p_i(z|G(y_1,z)), p_i(z|G(y_2,z))\|, \ \forall y_1, y_2 \sim p(y)$$

Implemented by optimizing re-construction error in hidden space

Adversarial games for aligning joint distribution

$$\min_{G} \max_{D_{xy}} \mathcal{L}_{xy} = \mathbb{E}_{(x,y)\sim p(x,y)}[\log(D_{xy}(x,y))] + \mathbb{E}_{y\sim p(y), z\sim p(z)}[\log(1 - D_{xy}(G(y,z),y))]$$

$$\min_{I,G} \max_{D_{xz}} \mathcal{L}_{xz} = \mathbb{E}_{x\sim p(x)}[\log(D_{xz}(x, I(x)))] + \mathbb{E}_{z\sim p(z), y\sim p(y)}[\log(1 - D_{xz}(G(y,z),z))]$$

Adversarial training

Main theorem: unbiased equilibrium

Theorem 3.3 *Minimizing $\mathcal{R}_z$ w.r.t. I will keep the equilibrium of the adversarial game $\mathcal{L}_{xz}$. Similarly, minimizing $\mathcal{R}_y$ w.r.t. C will keep the equilibrium of the adversarial game $\mathcal{L}_{xy}$ unchanged.*

Structured GANs: results

Deng et al., NeurIPS 2017

Method	MNIST $n=20$	MNIST $n=50$	MNIST $n=100$	SVHN $n=1000$	CIFAR-10 $n=4000$
Ladder [22]	-	-	**0.89($\pm$0.50)**	-	20.40($\pm$0.47)
VAE [12]	-	-	3.33($\pm$0.14)	36.02($\pm$0.10)	-
CatGAN [28]	-	-	1.39($\pm$0.28)	-	19.58($\pm$0.58)
ALI [5]	-	-	-	7.3	18.3
ImprovedGAN [27]	16.77($\pm$4.52)	2.21($\pm$1.36)	0.93 ($\pm$0.07)	8.11($\pm$1.3)	18.63($\pm$2.32)
TripleGAN [15]	5.40($\pm$6.53)	1.59($\pm$0.69)	0.92($\pm$0.58)	5.83($\pm$0.20)	18.82($\pm$0.32)
SGAN	**4.0($\pm$4.14)**	**1.29($\pm$0.47)**	**0.89($\pm$0.11)**	**5.73($\pm$0.12)**	**17.26($\pm$0.69)**

Table 2: Comparisons of semi-supervised classification errors (%) on MNIST, SVHN and CIFAR-10 test sets.

Impressive SSL classification accuracy

Fixed label in each row

Fixed style in each column

(a) MNIST

(b) SVHN

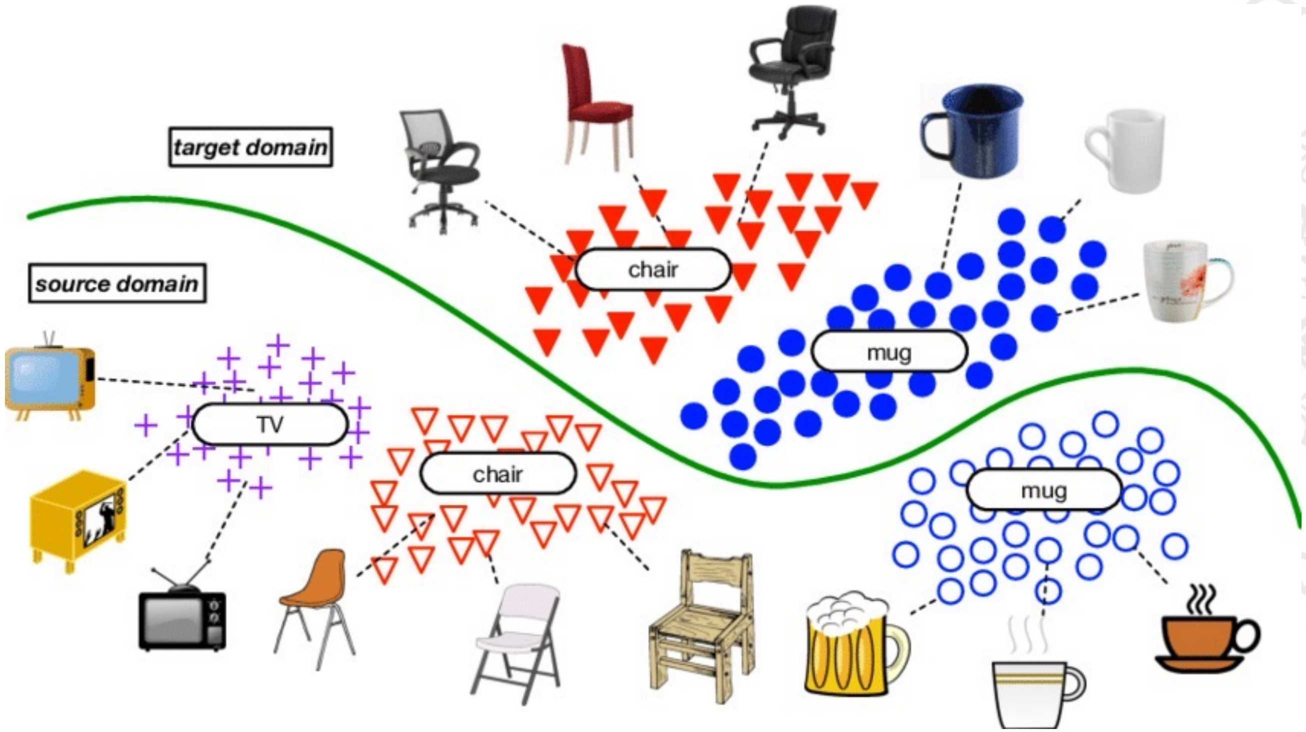
(c) CIFAR-10

Style transfer

A more extreme discriminative learning scenario: UDA

Unsupervised domain adaptation (UDA):

- the concerned domain (target domain) is unlabeled. We have only access to labeled data from a related domain (source domain)



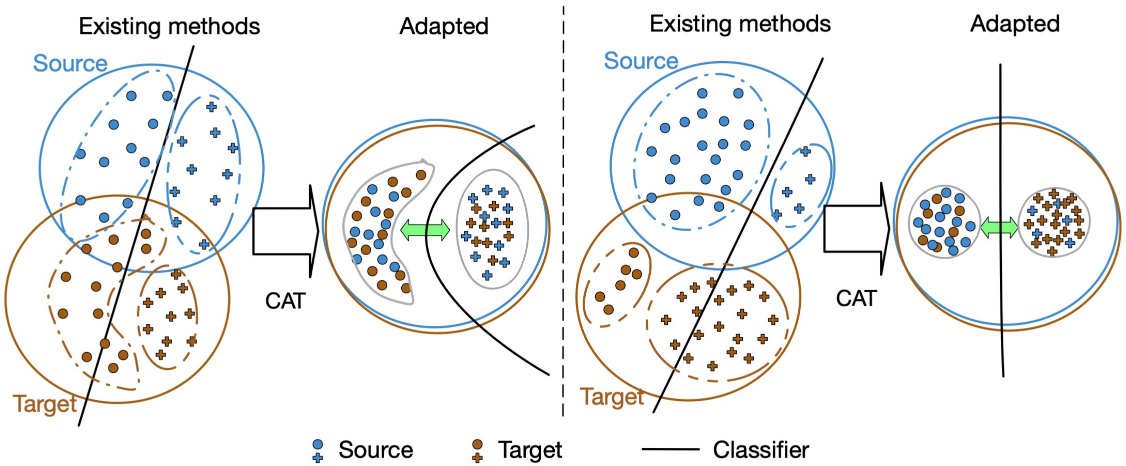
Marginal distribution alignment is not inadequate for UDA

The generalization bound for UDA

Mismatch between marginal dist.

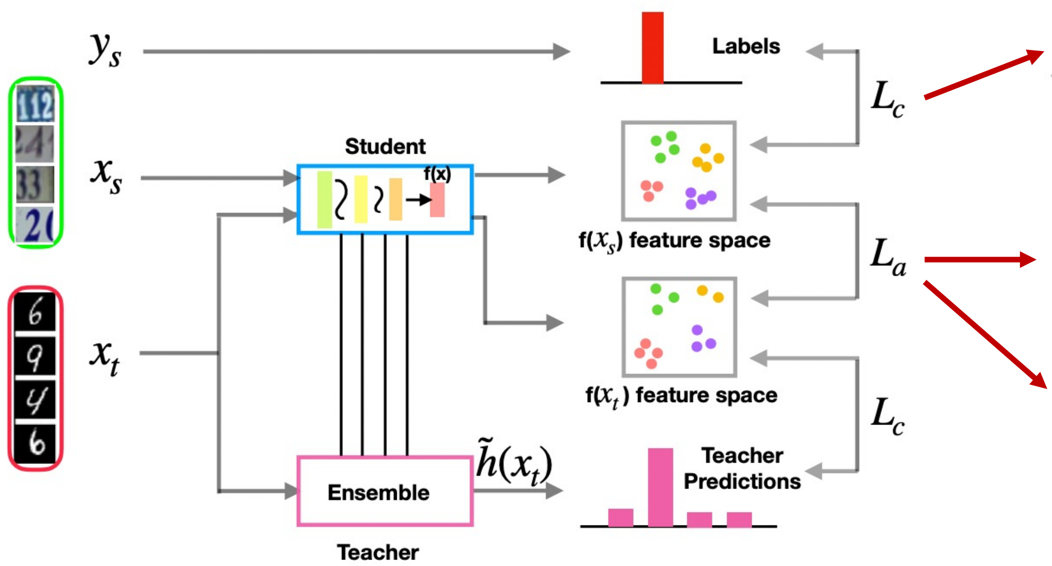
$$
\begin{aligned}
\epsilon_t(h) &\le \epsilon_s(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(s,t) \\
&\quad + \min_{\hat{h}\in\mathcal{H}} (\epsilon_s(\hat{h}, l_s) + \epsilon_t(\hat{h}, l_t)) \\
&\le \epsilon_s(h) + \boxed{\frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(s,t)} + \boxed{\epsilon_t(l_s, l_t)} \\
&\quad + \min_{\hat{h}\in\mathcal{H}} (\epsilon_s(\hat{h}, l_s) + \epsilon_t(\hat{h}, l_s)).
\end{aligned}
$$

Mismatch between labeling functions (class-conditional dist.)



Cluster alignment with a teacher for UDA

Deng et al., ICCV 2019



$$\min_{\theta} \mathcal{L}_c(\mathcal{X}_s, \mathcal{X}_t) = \mathcal{L}_c(\mathcal{X}_s) + \mathcal{L}_c(\mathcal{X}_t),$$

$$\mathcal{L}_c(\mathcal{X}) = \frac{1}{|\mathcal{X}|^2} \sum_{i=1}^{|\mathcal{X}|} \sum_{j=1}^{|\mathcal{X}|} \left[\delta_{ij} d\left(h(x^i), h(x^j)\right) + (1-\delta_{ij}) \max\left(0, m - d\left(h(x^i), h(x^j)\right)\right)\right],$$

$$\min_{\theta} \mathcal{L}_a(\mathcal{X}_s, \mathcal{Y}_s, \mathcal{X}_t) = \frac{1}{K} \sum_{k=1}^{K} [d(\lambda_{s,k}, \lambda_{t,k})]$$

$$\min_{\theta} \max_{\phi} \mathcal{L}_{cd}(\mathcal{X}_s, \mathcal{X}_t) = \frac{1}{N} \sum_{i=1}^{N} \left[\log c\left(h(x_s^i; \theta); \phi\right)\right] + \frac{1}{\tilde{M}} \sum_{i=1}^{\tilde{M}} \left[\log\left(1 - c\left(h(x_t^i; \theta); \phi\right)\right) \gamma_i\right]$$

Distribution alignment with class-conditional structure awareness:

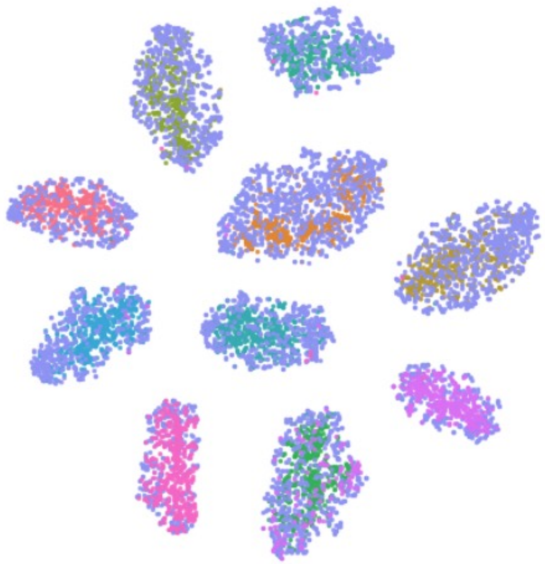
- implement the *cluster assumption* of discriminative learning

Cluster alignment with a teacher for UDA: results

Deng et al., ICCV 2019

Method	*SVHN to MNIST*	*MNIST to USPS*	*USPS to MNIST*
RevGrad [7]	27.4 ± 6.3	26.7 ± 2.0	17.9 ± 1.4
MSTN [49]	25.8 ± 3.6	30.3 ± 1.0	29.4 ± 0.5
CAT	$\mathbf{100.0} \pm 0.05$	$\mathbf{100.0} \pm 0.0$	$\mathbf{99.9} \pm 0.2$

Especially effective for class imbalanced tasks



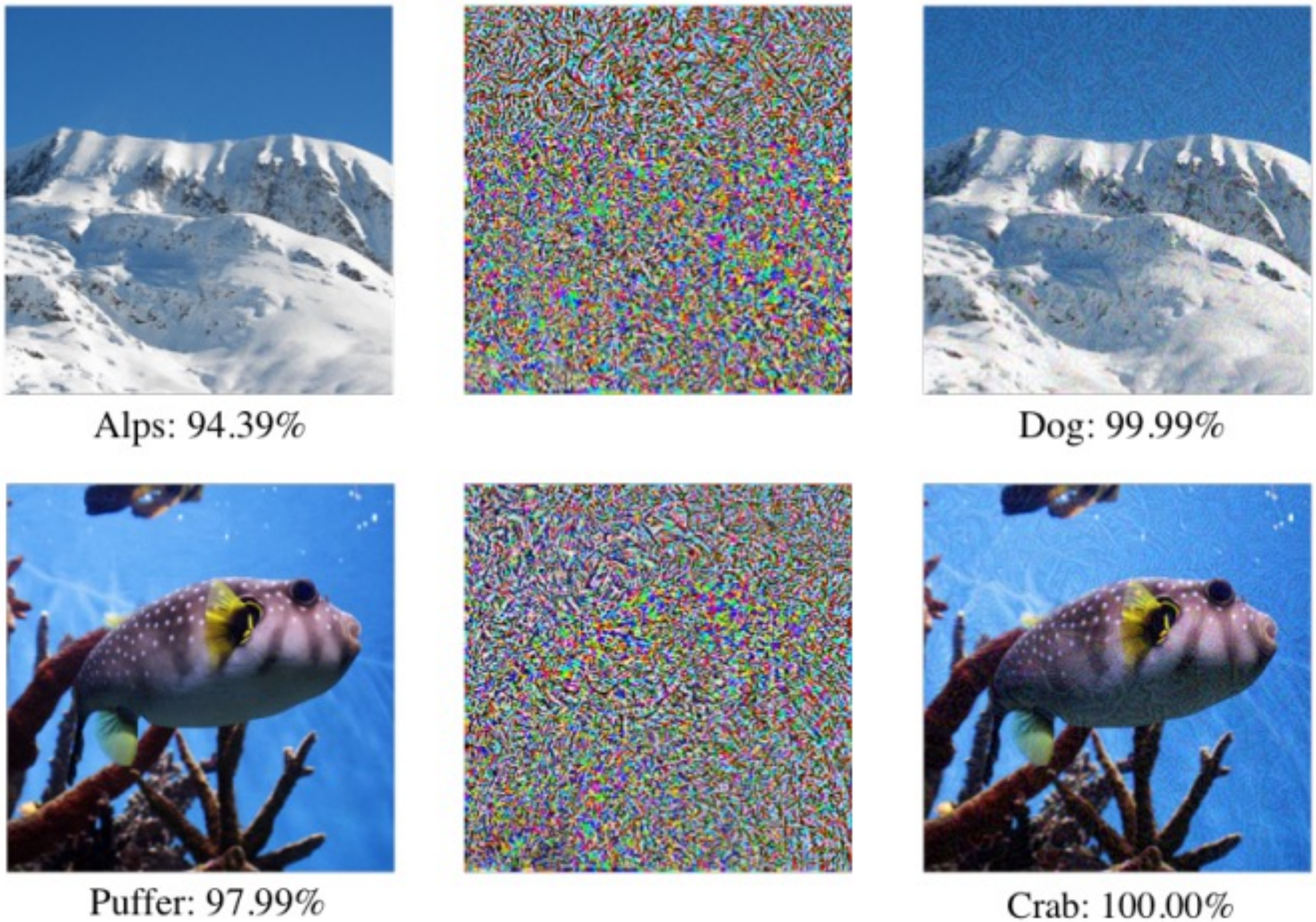
Separated clusters in the feature space

Method	*SVHN to MNIST*	*MNIST to USPS*	*USPS to MNIST*
Source Only	60.1 ± 1.1	75.2 ± 1.6	57.1 ± 1.7
DDC [45]	68.1 ± 0.3	79.1 ± 0.5	66.5 ± 3.3
CoGAN [20]	-	91.2 ± 0.8	89.1 ± 0.8
DRCN [8]	82.0 ± 0.1	91.8 ± 0.09	73.7 ± 0.04
ADDA [44]	76.0 ± 1.8	89.4 ± 0.2	90.1 ± 0.8
LEL [26]	81.0 ± 0.3	-	-
AssocDA [11]	97.6	-	-
MSTN [49]	91.7 ± 1.5	92.9 ± 1.1	-
CAT	98.1 ± 1.3	90.6 ± 2.3	80.9 ± 3.1
RevGrad [7]	73.9	77.1 ± 1.8	73.0 ± 2.0
RevGrad+CAT	98.0 ± 0.8	93.7 ± 1.1	95.7 ± 1.3
rRevGrad+CAT	$\mathbf{98.8} \pm 0.02$	94.0 ± 0.7	$\mathbf{96.0} \pm 0.9$
MCD [37]	96.2 ± 0.4	94.2 ± 0.7	94.1 ± 0.3
MCD+CAT	97.1 ± 0.2	$\mathbf{96.3} \pm 0.5$	95.2 ± 0.4
VADA [41]	94.5	-	-
VADA+CAT	95.2	-	-

SOTA UDA performance

DNNs are vulnerable to adversarial examples

Clean images | Adversarial noise | Adversarial examples



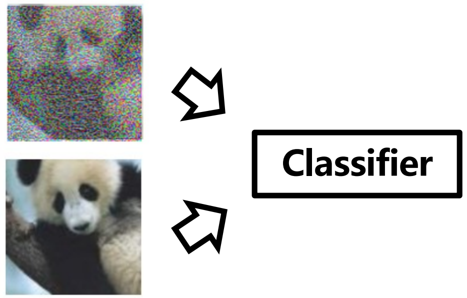
Dong et al. 2018

What is the underlying distribution of adversarial examples?

Modeling adversarial distribution may be helpful

In the sense of improving adversarially robustness

Adversarial training (AT):



Outer minimization: train a robust classifier

$$\overbrace{\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \underbrace{\max_{\delta_i \in S} L(f_\theta(x_i + \delta_i), y_i)}_{\text{Inner maximization: generate an adversarial example}}}^{\text{Outer minimization: train a robust classifier}}$$

Inner maximization: generate an adversarial example

Generalization issue of AT under point-estimate attacker

Model	$\mathcal{A}_{\text{nat}}$	FGSM	PGD-20	PGD-100	MIM	C&W	FeaAttack	$\mathcal{A}_{\text{rob}}$
Standard	**94.81%**	12.05%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
AT_{FGSM}	93.80%	**79.86%**	0.12%	0.04%	0.06%	0.13%	0.01%	0.01%
$\text{AT}_{\text{PGD}}{}^{\dagger}$	87.25%	56.04%	45.88%	45.33%	47.15%	46.67%	46.01%	44.89%
AT_{PGD}	86.91%	58.30%	50.03%	49.40%	51.40%	50.23%	50.46%	48.26%
ALP	86.81%	56.83%	48.97%	48.60%	50.13%	49.10%	48.51%	47.90%
FeaScatter	89.98%	**77.40%**	**70.85%**	**68.81%**	**72.74%**	**58.46%**	37.45%	37.40%

Adversarial distributional training
Deng et al., NeurIPS 2020

Outer minimization: train a robust classifier

A **probabilistic modeling** of **heterogeneous** adversarial examples

$$\underbrace{\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \max_{p(\delta_i)\in P} \mathbb{E}_{p(\delta_i)}[L(f_\theta(x_i+\delta_i), y_i)] + \lambda H\big(p(\delta_i)\big)}$$

Inner maximization: learn an adversarial distribution

- Under mild assumption, we theoretically prove iterative optimization method can still be used for solving the minimax problem

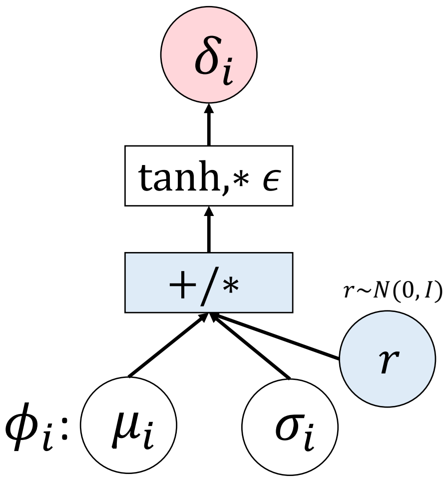
Theorem 1. *Suppose Assumptions 1 and 2 hold. We define* $\rho(\boldsymbol{\theta}) = \max_{p(\boldsymbol{\delta}_i)\in\mathcal{P}} \mathcal{J}\big(p(\boldsymbol{\delta}_i), \boldsymbol{\theta}\big)$, *and* $\mathcal{P}^*(\boldsymbol{\theta}) = \{p(\boldsymbol{\delta}_i) \in \mathcal{P} : \mathcal{J}\big(p(\boldsymbol{\delta}_i), \boldsymbol{\theta}\big) = \rho(\boldsymbol{\theta})\}$. *Then* $\rho(\boldsymbol{\theta})$ *is directionally differentiable, and its directional derivative along the direction* $\mathbf{v}$ *satisfies*

$$\rho'(\boldsymbol{\theta}; \mathbf{v}) = \sup_{p(\boldsymbol{\delta}_i)\in\mathcal{P}^*(\boldsymbol{\theta})} \mathbf{v}^\top \nabla_{\boldsymbol{\theta}} \mathcal{J}\big(p(\boldsymbol{\delta}_i), \boldsymbol{\theta}\big). \tag{6}$$

Particularly, when $\mathcal{P}^*(\boldsymbol{\theta}) = \{p^*(\boldsymbol{\delta}_i)\}$ *only contains one maximizer,* $\rho(\boldsymbol{\theta})$ *is differentiable at* $\boldsymbol{\theta}$ *and*

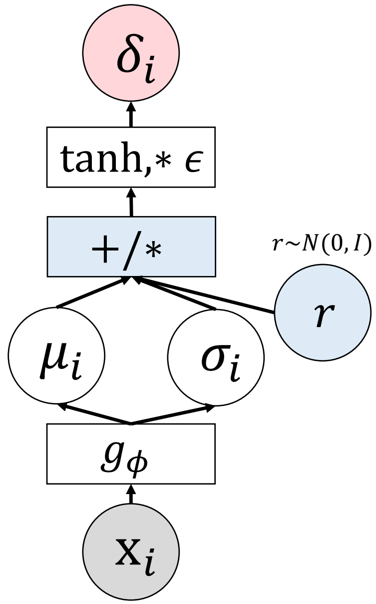
$$\nabla_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \mathcal{J}\big(p^*(\boldsymbol{\delta}_i), \boldsymbol{\theta}\big). \tag{7}$$

Use **DGMs** to instantiate the adversarial distributions



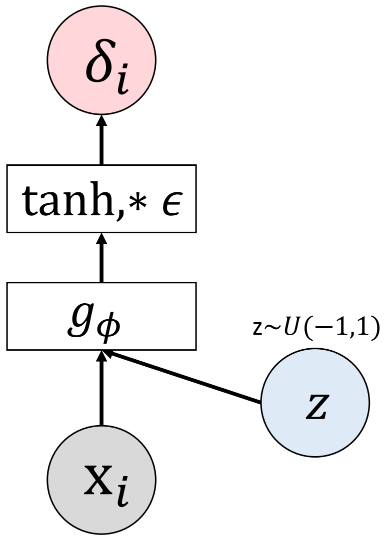
(a)

DGM with
explicit density



(b)

DGM with
explicit density
(amortized version)

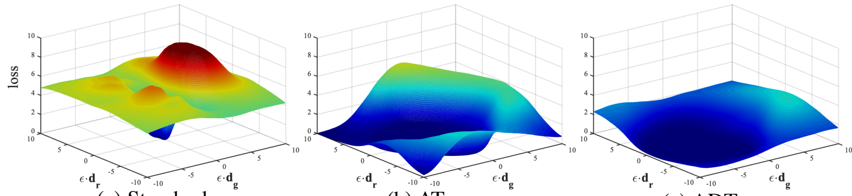


(c)

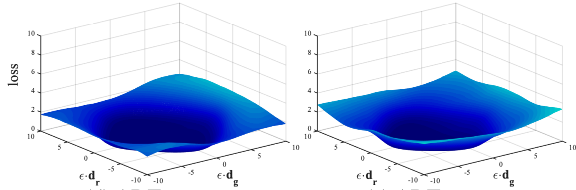
DGM with
implicit density
(amortized version)

Adversarial distributional training: results

The distribution captures *more diverse modes* of adv. examples



(a) Standard (b) AT_{PGD} (c) ADT_{EXP}



(d) $ADT_{EXP\text{-}AM}$ (e) $ADT_{IMP\text{-}AM}$

Model	Dominant eigenvalue
Standard	1.8301±6.3663
AT_{PGD}	0.0242±0.0478
ADT_{EXP}	**0.0180**±0.0311
$ADT_{EXP\text{-}AM}$	0.0181±**0.0270**
$ADT_{IMP\text{-}AM}$	0.0211±0.0353

(f) Comparison on the Hessian

ADT leads to flatter loss surfaces

Model	$\mathcal{A}_{nat}$	FGSM	PGD-20	PGD-100	MIM	C&W	FeaAttack	$\mathcal{A}_{rob}$
Standard	**94.81%**	12.05%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
AT_{FGSM}	93.80%	**79.86%**	0.12%	0.04%	0.06%	0.13%	0.01%	0.01%
AT_{PGD} †	87.25%	56.04%	45.88%	45.33%	47.15%	46.67%	46.01%	44.89%
AT_{PGD}	86.91%	58.30%	50.03%	49.40%	51.40%	50.23%	50.46%	48.26%
ALP	86.81%	56.83%	48.97%	48.60%	50.13%	49.10%	48.51%	47.90%
FeaScatter	89.98%	**77.40%**	**70.85%**	**68.81%**	**72.74%**	**58.46%**	37.45%	37.40%
ADT_{EXP}	86.89%	60.41%	52.18%	**51.69%**	**53.27%**	52.49%	**52.38%**	**50.56%**
$ADT_{EXP\text{-}AM}$	87.82%	62.42%	51.95%	51.26%	52.99%	51.75%	52.04%	**50.04%**
$ADT_{IMP\text{-}AM}$	88.00%	**64.89%**	**52.28%**	51.23%	52.64%	**52.65%**	51.89%	**49.81%**

Superior adversarial robustness over baselines with clear margins

Thanks!