Possibilistic Predictive Uncertainty for Deep Learning

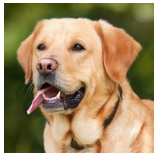
Yao Ni[1] Jeremie Houssineau[1] Yew-Soon Ong[1,2] Piotr Koniusz[3, 4]

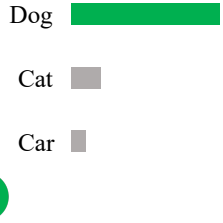
[1]Nanyang Technological University [2]A*STAR [3]University of New South Wales [4]CSIRO

ICML 2026

Background: Overconfidence issue in Deep Learning

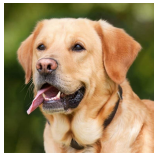
In distribution

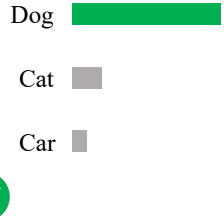




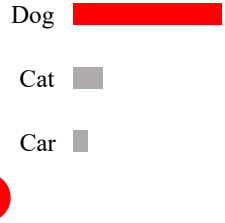
Background: Overconfidence issue in Deep Learning

In distribution



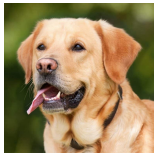


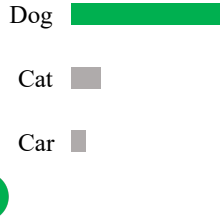
Out of distribution



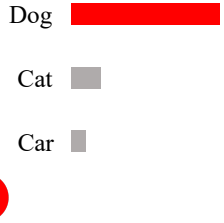
Background: Overconfidence issue in Deep Learning

In distribution





Out of distribution

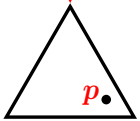


Poor decisions. Risky in applications: medicine, finance.

Background: Reliable Confidence via Uncertainty Modeling

Overconfidence

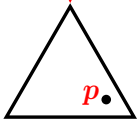
Unseen Sample



Background: Reliable Confidence via Uncertainty Modeling

Overconfidence

Unseen Sample

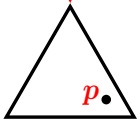


Point estimate

Background: Reliable Confidence via Uncertainty Modeling

Overconfidence

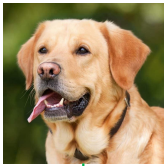
Unseen Sample

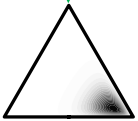


Point estimate

Avoid Overconfidence

Familiar Sample





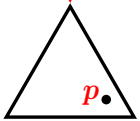
Concentrated

Epistemic uncertainty ↓

Background: Reliable Confidence via Uncertainty Modeling

Overconfidence

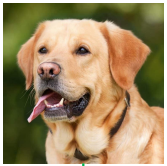
Unseen Sample



Point estimate

Avoid Overconfidence

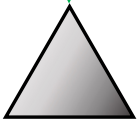
Familiar Sample



Concentrated

Epistemic uncertainty ↓

Unfamiliar Sample



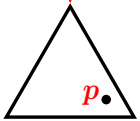
Dispersed

Epistemic uncertainty ↑

Background: Reliable Confidence via Uncertainty Modeling

Overconfidence

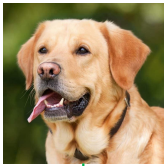
Unseen Sample

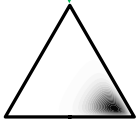


Point estimate

Avoid Overconfidence

Familiar Sample

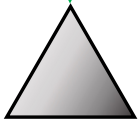




Concentrated

Epistemic uncertainty ↓

Unfamiliar Sample



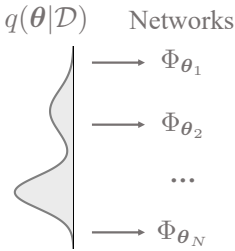
Dispersed

Epistemic uncertainty ↑

Distribution Estimate

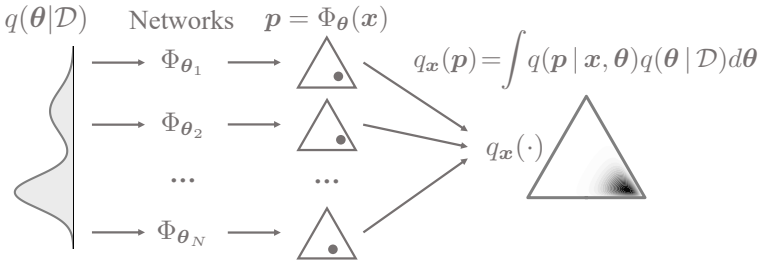
Background: Existing Methods

Principled Bayesian Learning for Predictive Distribution



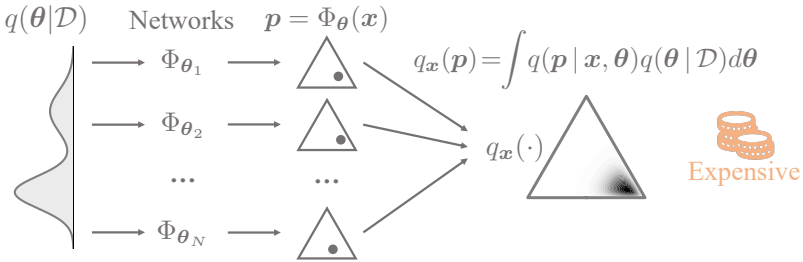
Background: Existing Methods

Principled Bayesian Learning for Predictive Distribution



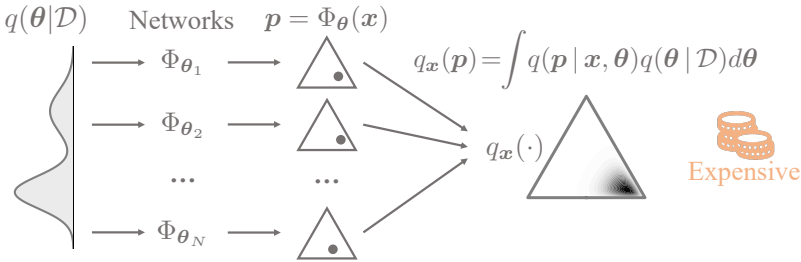
Background: Existing Methods

Principled Bayesian Learning for Predictive Distribution

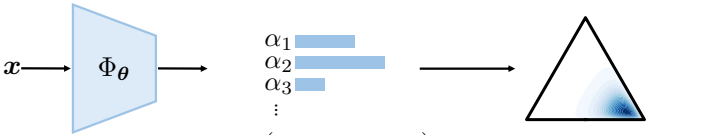


Background: Existing Methods

Principled Bayesian Learning for Predictive Distribution

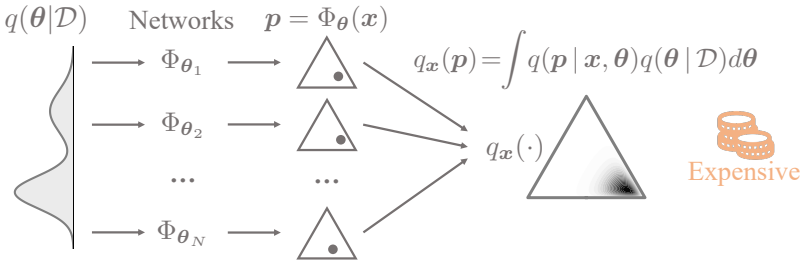


Efficient Second-Order Predictors

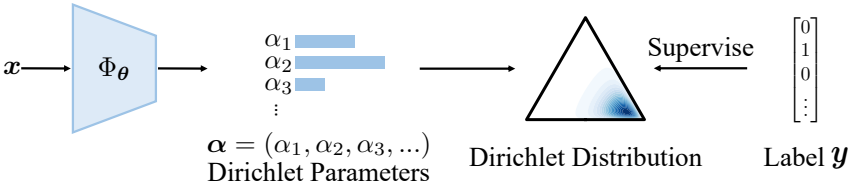


Background: Existing Methods

Principled Bayesian Learning for Predictive Distribution

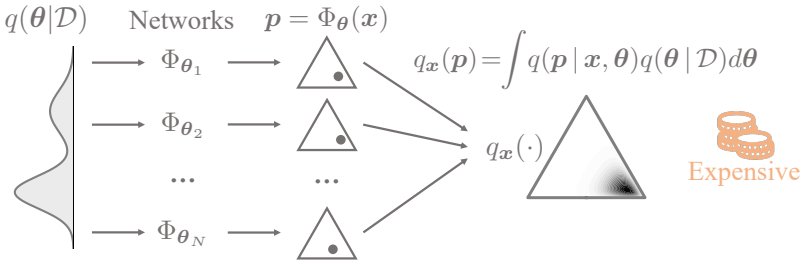


Efficient Second-Order Predictors



Background: Existing Methods

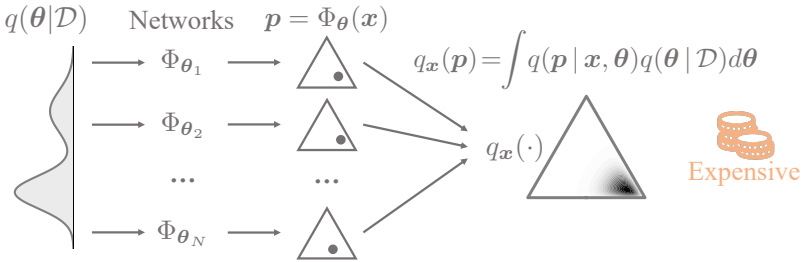
Principled Bayesian Learning for Predictive Distribution



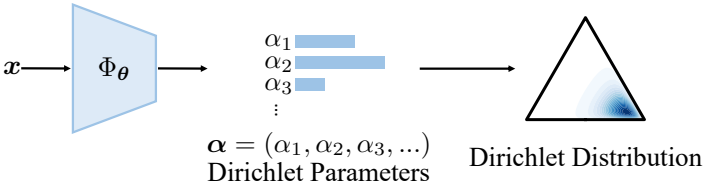
Efficient Second-Order Predictors

Methods: Motivation

Principled Bayesian Learning for Predictive Distribution

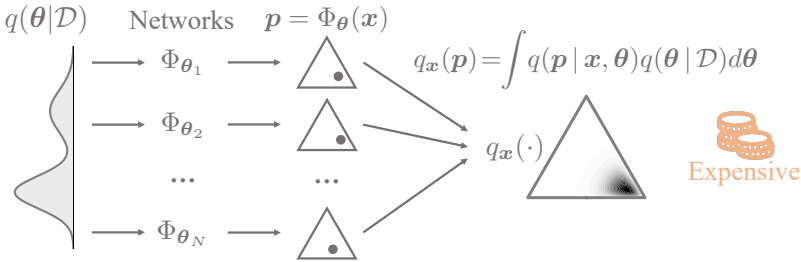


Efficient Second-Order Predictors

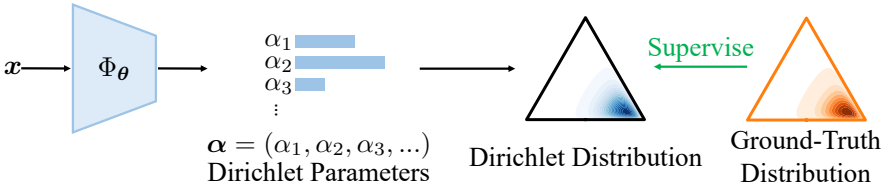


Methods: Motivation

Principled Bayesian Learning for Predictive Distribution

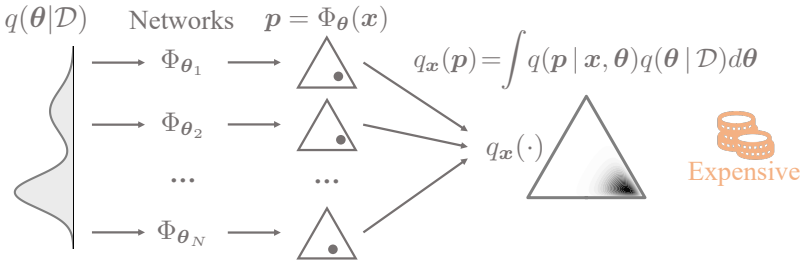


Efficient Second-Order Predictors

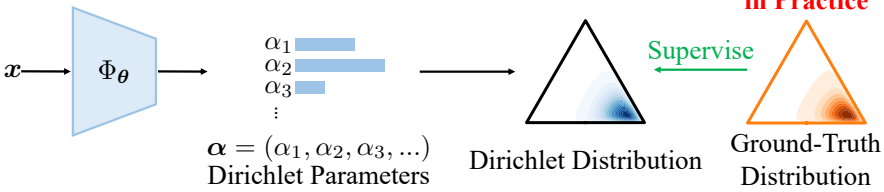


Methods: Motivation

Principled Bayesian Learning for Predictive Distribution

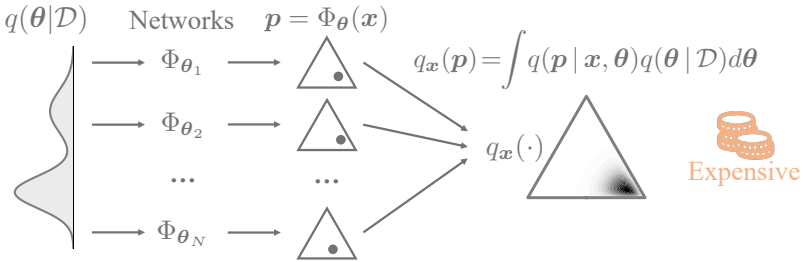


Efficient Second-Order Predictors

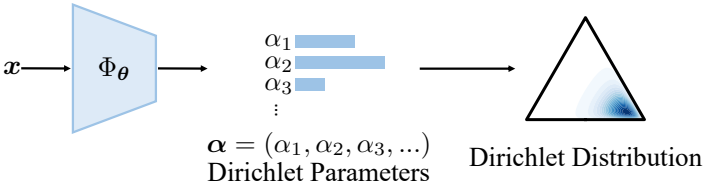


Methods: Motivation

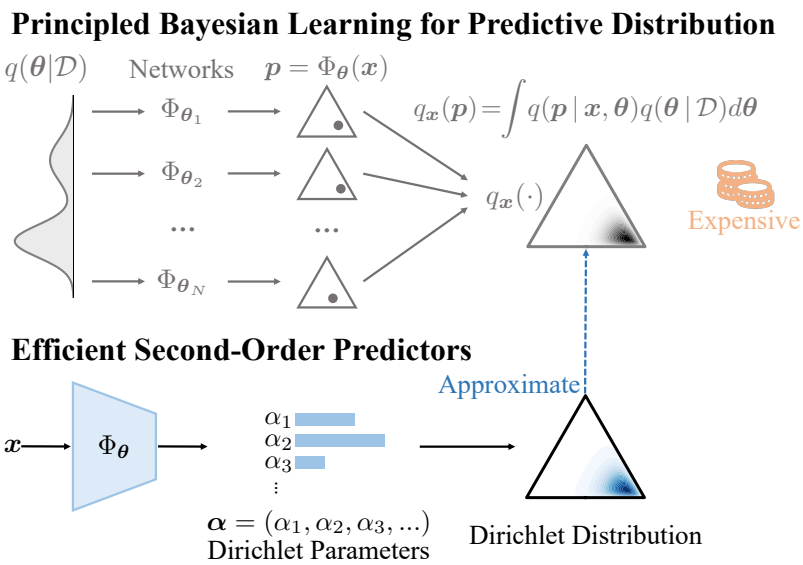
Principled Bayesian Learning for Predictive Distribution



Efficient Second-Order Predictors

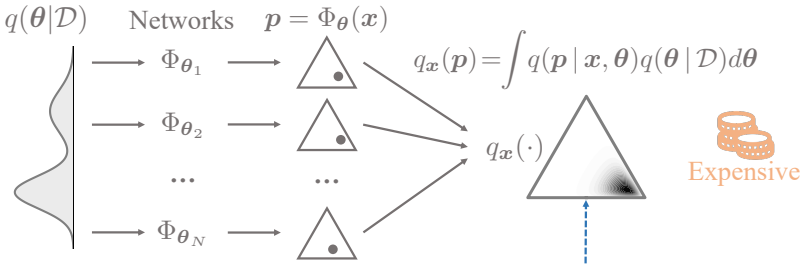


Methods: Motivation

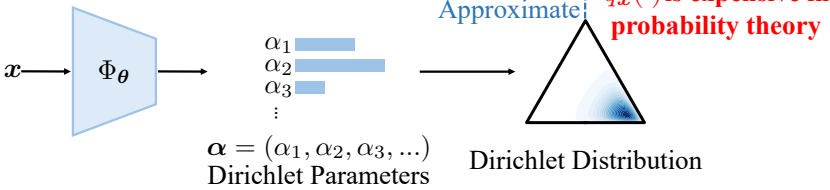


Methods: Motivation

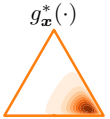
Principled Bayesian Learning for Predictive Distribution



Efficient Second-Order Predictors

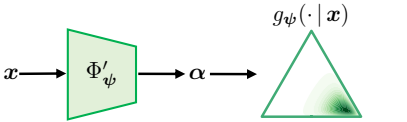


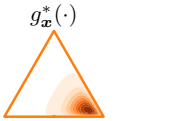
Methods: DAPPr



Well-Defined Reference Possibility Function

Methods: DAPPr





Methods: DAPPr

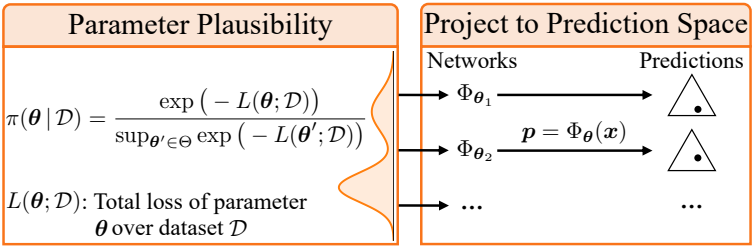
Methods: DAPPr

Parameter Plausibility

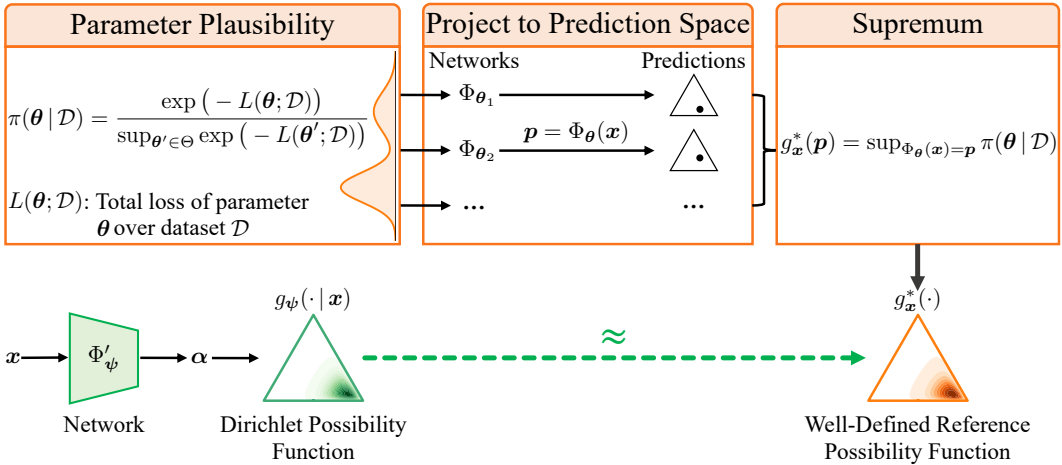
$$\pi(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{\exp\big(-L(\boldsymbol{\theta}; \mathcal{D})\big)}{\sup_{\boldsymbol{\theta}' \in \Theta} \exp\big(-L(\boldsymbol{\theta}'; \mathcal{D})\big)}$$

$L(\boldsymbol{\theta}; \mathcal{D})$: Total loss of parameter $\boldsymbol{\theta}$ over dataset $\mathcal{D}$

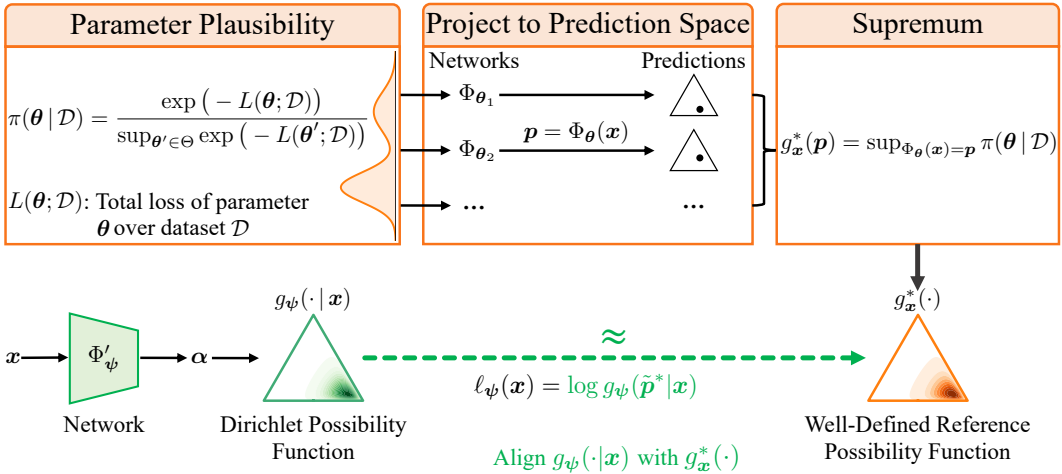
Methods: DAPPr



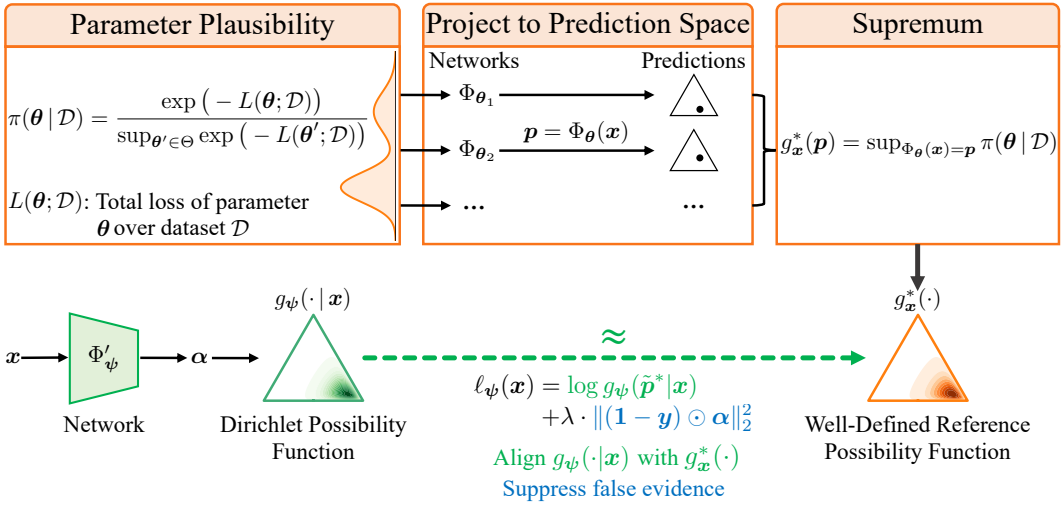
Methods: DAPPr



Methods: DAPPr



Methods: DAPPr



Method: Pytorch-style Code

DAPPr.py (10 lines) (https://github.com/MaxwellYaoNi/DAPPr/blob/main/DAPPr.py)

```python
import torch.nn.functional as F
def DAPPr_loss(logits, labels, lamb, eps=1e-8):
  alpha = F.softplus(logits) + 1
  y = F.one_hot(labels, logits.shape[-1]).float()
  alpha_star = alpha - y + eps # add eps to avoid log instability
  p_star = (alpha_star / alpha_star.sum(dim=1, keepdim=True)).detach()
  alpha_0 = alpha.sum(1)
  loss_alpha = alpha_0 * alpha_0.log() + (alpha * (p_star/alpha).log()).sum(dim=1)
  loss_reg = (alpha * (1 - y)).square().sum(dim=1)
  return loss_alpha.mean() + lamb * loss_reg.mean()
```

Method: DAPPr usage

DAPPr usage (https://github.com/MaxwellYaoNi/DAPPr#usage-guide)

```
import torch.nn.functional as F
+from DAPPr import DAPPr_loss, DAPPr_uncertainty

# Training
for x, labels in train_loader:
    logits = model(x)
-   loss = F.cross_entropy(logits, labels)
+   loss = DAPPr_loss(logits, labels, lamb=2e-4)
```

Method: DAPPr usage

DAPPr usage (https://github.com/MaxwellYaoNi/DAPPr#usage-guide)

```
import torch.nn.functional as F
+from DAPPr import DAPPr_loss, DAPPr_uncertainty

# Training
for x, labels in train_loader:
    logits = model(x)
-   loss = F.cross_entropy(logits, labels)
+   loss = DAPPr_loss(logits, labels, lamb=2e-4)

# Testing:  uncertainty estimation
for x, labels in test_loader:
    logits = model(x)
+   uncertainty = DAPPr_uncertainty(logits)
+   AU = uncertainty["AU"] # aleatoric uncertainty
+   EU = uncertainty["EU"] # epistemic uncertainty
```

Experiments Overview: Image Classification

Table: Average performance over (CIFAR-10/100, CUB-200-2011, StanfordDogs, TinyImageNet)

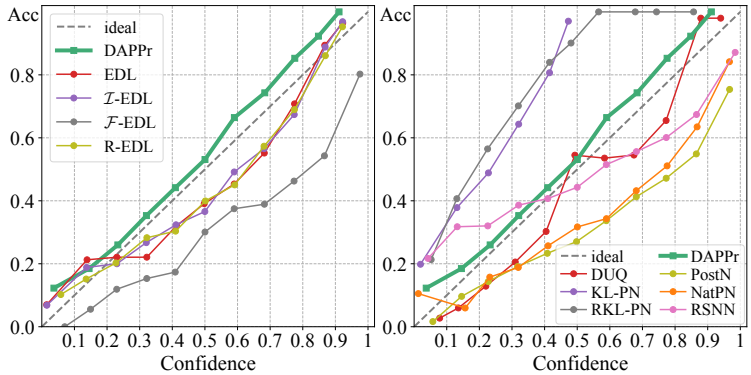
Method	Acc ↑	Conf. (AUPR ↑)	OOD (AUPR ↑)
RKL-PN	60.7	86.6	70.1
RSNN	64.0	88.7	75.6
R-EDL	61.7	89.5	76.6
FEDL	61.2	85.0	76.0
DAPPr	**67.7**	**92.2**	**79.5**
Ensemble	69.3	91.8	78.9
+DAPPr	**72.1**	**93.9**	**82.8**

Experiments: Uncertainty Quantification in LLMs

Table: As in IB-EDL (ICLR 2025), we finetune Mistral-7B on OBQA and report OOD AUROC ↑ on (ARC-C, ARC-E, CSQA)

Method	Acc. ↑	ECE ↓	NLL ↓	ARC-C	ARC-E	CSQA
CE	88.06	11.29	0.85	60.40	53.30	63.70
MC Drop	88.07	11.22	0.84	60.39	53.30	63.70
Ensemble	88.51	8.87	0.67	60.67	54.05	63.80
EDL	87.23	6.23	0.47	77.34	74.18	78.28
I-EDL	88.06	9.63	0.45	82.28	79.42	82.07
R-EDL	88.33	5.43	0.41	72.85	67.56	71.93
IB-EDL	88.73	**2.27**	0.41	88.58	94.29	83.85
DAPPr	**89.47**	3.11	**0.36**	**90.42**	**95.12**	**90.19**

Experiments: Reliability Diagram on StanfordDogs



DAPPr matches the diagonal well and exhibits conservative underconfidence.

Conclusions

- We introduce DAPPr, a principled, efficient framework grounded in possibility theory with a simple closed-form training objective.
- DAPPr achieves superior performance over SOTA second-order predictors across diverse benchmarks.