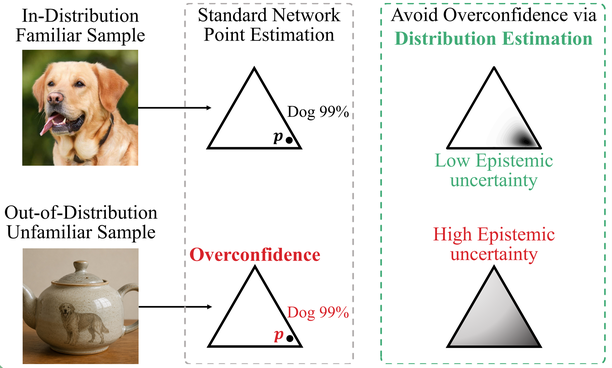
Possibilistic Predictive Uncertainty for Deep Learning

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]Data61♥CSIRO
{yao.ni; jeremie.houssineau; asysong}@ntu.edu.sg, piotr.koniusz@{unsw.edu.au; data61.csiro.au}

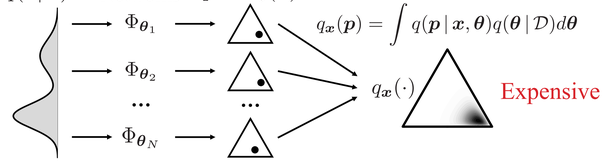
Background: Overconfidence Issue

Standard networks use point estimation and become overconfident. We overcome this via distribution estimation.

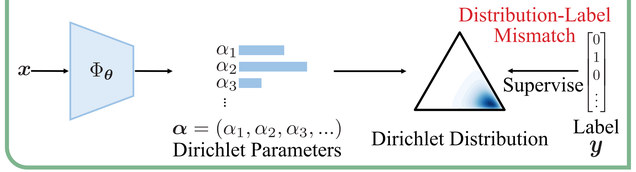


Prior Predictive Distribution Methods

Principled Bayesian Learning for Predictive Distribution



Efficient Second-Order Predictors



Preliminary: Possibility Theory

Possibility function: $f : \Theta \to [0, 1]$, with $\sup_{\theta\in\Theta} f(\theta) = 1$.

$$\Pi(B) \doteq \sup_{\theta\in B} f(\theta), \text{ where } B \subseteq \Theta$$

For an event $A \subseteq \Theta$, difference between probability/possibility theory to represent "A cannot be excluded" w/o evidence:

Probability theory: $P(A) + P(A^C) = 1$

$$P(A) = 1 \quad \to \quad P(A^C) = 0.$$

False Precision

Possibility theory: $\max(\Pi(A), \Pi(A^C)) = 1$

$$\Pi(A) = 1 \quad \text{and} \quad \Pi(A^C) = 1$$

Ignorance Preservation

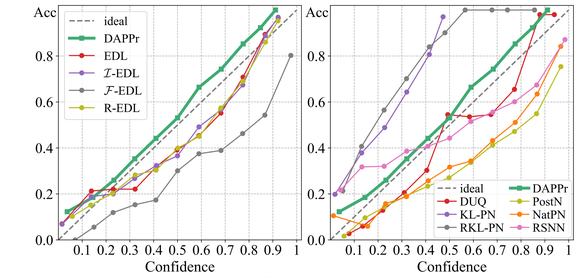
Experiments

Vision Results: Avg. over CIFAR-10/100, CUB, 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**

LLM Results: Mistral-7B on OBQA; OOD AUROC↑ on ARC-C/E and CSQA.

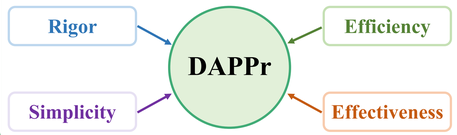
Method	Test Acc. ↑	ECE ↓	NLL ↓	ARC-C	ARC-E	CSQA
CE	88.06	11.29	0.85	60.40	53.30	63.70
MC Drop	88.07	11.22	0.84	60.39	53.30	63.70
Ensemble	88.51	8.87	0.67	60.67	54.05	63.80
EDL	87.23	6.23	0.47	77.34	74.18	78.28
I-EDL	88.06	9.63	0.45	82.28	79.42	82.07
R-EDL	88.33	5.43	0.41	72.85	67.56	71.93
IB-EDL	88.73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



Reliability Diagram on StanfordDogs: **DAPPr matches the diagonal well.**

Key Takeaway:

- **First** possibility-theoretic second-order predictor.
- Closed-form objective with **10-line** implementation.
- **State-of-the-art uncertainty estimation** for vision and language.



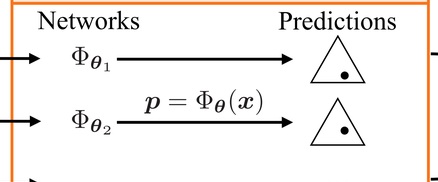
Method

Parameter Plausibility

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

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

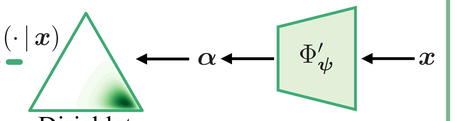
Project to Prediction Space



Supremum

$$g^*_{\boldsymbol{x}}(\boldsymbol{p}) = \sup_{\Phi_{\boldsymbol{\theta}}(\boldsymbol{x})=\boldsymbol{p}} \pi(\boldsymbol{\theta}\,|\,\mathcal{D})$$

Align with the reference function using efficient second-order predictor



Implementation: Pytorch-Style pseudo code for DAPPr (∼10 lines)

```
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()
```

Usage by simply replacing cross-entropy loss in training

```
for x, labels in train_loader:

  logits = model(x)

- loss = F.cross_entropy(logits, labels)

+ loss = DAPPr_loss(logits, labels, lamb)

  ...
```

Key Results: Under cross-entropy loss and high-capacity nets.

$$g^*_{\boldsymbol{x}}(\boldsymbol{p}) \propto \exp(-\ell(\boldsymbol{p},\boldsymbol{y})), \quad \tilde{\boldsymbol{p}}^* = (\boldsymbol{\alpha}-\boldsymbol{y})/(\|\boldsymbol{\alpha}\|_1 - 1)$$

$\tilde{\boldsymbol{p}}^*$ identifies the worst-case discrepancy between $g_\psi(\cdot|\boldsymbol{x})$ and $g^*_{\boldsymbol{x}}(\cdot)$. The DAPPr loss is:

$$\ell_\psi(\boldsymbol{x}) = \log g_\psi(\tilde{\boldsymbol{p}}^*|\boldsymbol{x}) + \lambda \cdot \|(\mathbf{1}-\boldsymbol{y}) \odot \boldsymbol{\alpha}\|_2^2$$

↓ Align $g_\psi(\cdot|\boldsymbol{x})$ with $g^*_{\boldsymbol{x}}(\cdot)$ ↓ Suppress unsupported evidence