Unsupervised Temperature Scaling

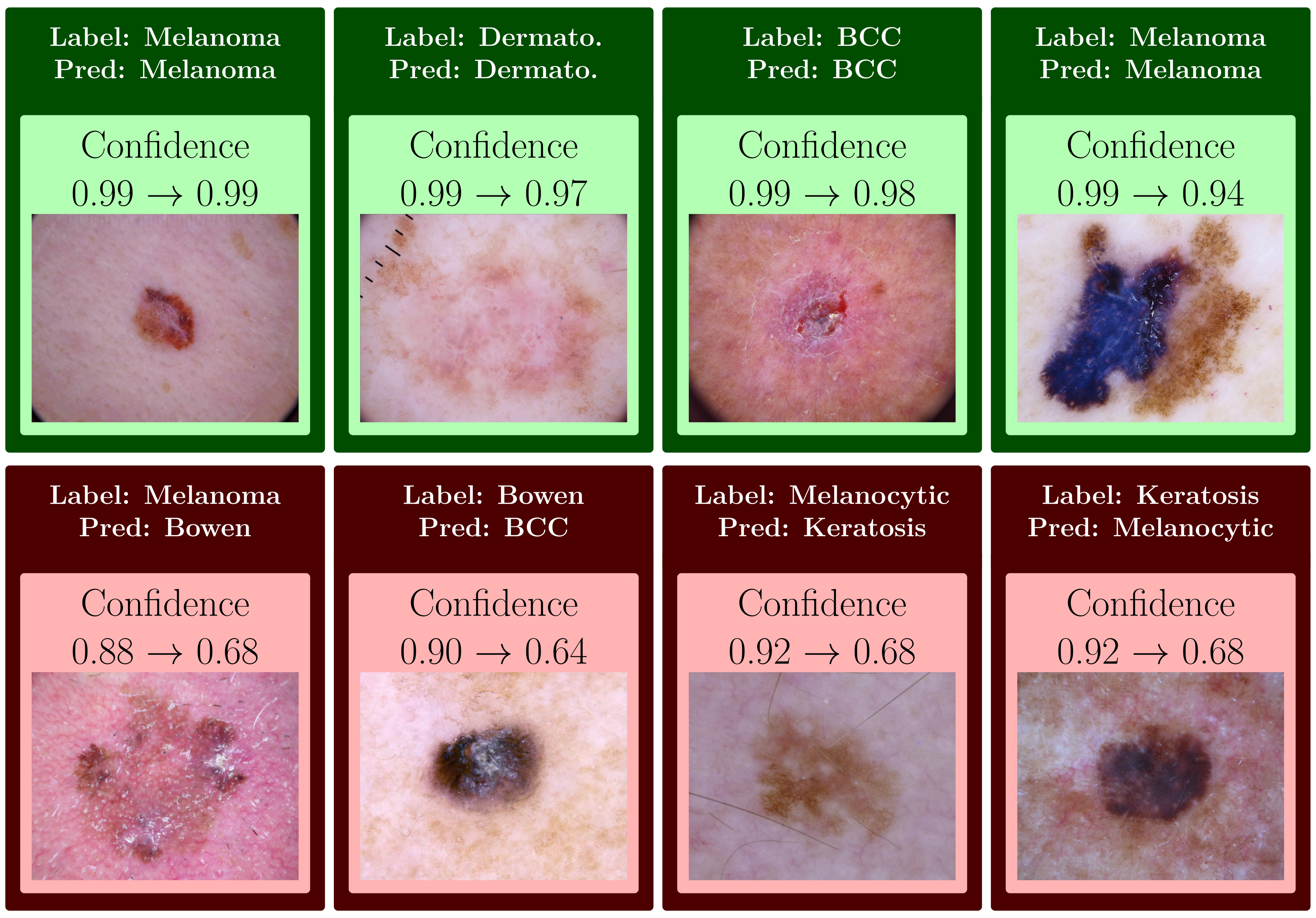
An Unsupervised Post-Processing Calibration Method of Deep Networks

Azadeh Sadat Mozafari, Hugo Siqueira Gomes, Wilson Leão, Christian Gagné

azadeh-sadat.mozafari.1@ulaval.ca - hugo.siqueira-gomes.1@ulaval.ca - wilson.leao@petrobras.com.br - christian.gagne@gel.ulaval.ca

Motivation

- **Deep neural networks achieve impressive results but suffer from being overconfident**
- **Confidence is important to bring trust and reliability to decision-making applications**



ISIC Dataset (Tschandl et al., 2018; Codella et al., 2017) ResNet152 Accuracy: 88.02%

How to measure calibration?

Negative Log-Likelihood (NLL)

similarity between distributions

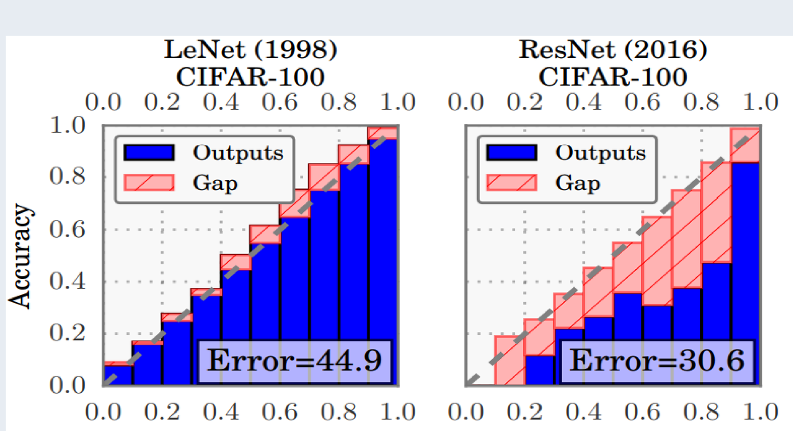
softmax output $S_y(\mathbf{x})$ and true conditional distribution $Q(y|\mathbf{x})$

$$\text{NLL} = -\sum_{(\mathbf{x}_i, y_i)} \log\Big(S_{y=y_i}(\mathbf{x}_i)\Big), \quad (\mathbf{x}_i, y_i) \sim Q(\mathbf{x}, y)$$

Expected Calibration Error (ECE)

x% of confidence ⇒ correct x% of the time

$$\text{ECE} = \sum_{l=1}^{L} \frac{|B_l|}{N} |\text{acc}(B_l) - \text{conf}(B_l)|$$



(Guo et al., 2017)

Temperature Scaling (Guo et al., 2017)

- **The goal is to rescale the logit layer with one single parameter T in order to minimize the calibration error of a pre-trained model.**

$$S(h_i) = \frac{\exp^{h_i}}{\Sigma_j^K \exp^{h_j}} \Rightarrow S(h_i, T) = \frac{\exp^{\frac{h_i}{T}}}{\Sigma_j^K \exp^{\frac{h_j}{T}}}$$

$$T^* = \arg\min_T \Bigg(\sum_{i=1}^{N} \overbrace{-\log\Big(S_{y=y_i}(x_i, T)\Big)}^{\text{NLL}} \Bigg)$$

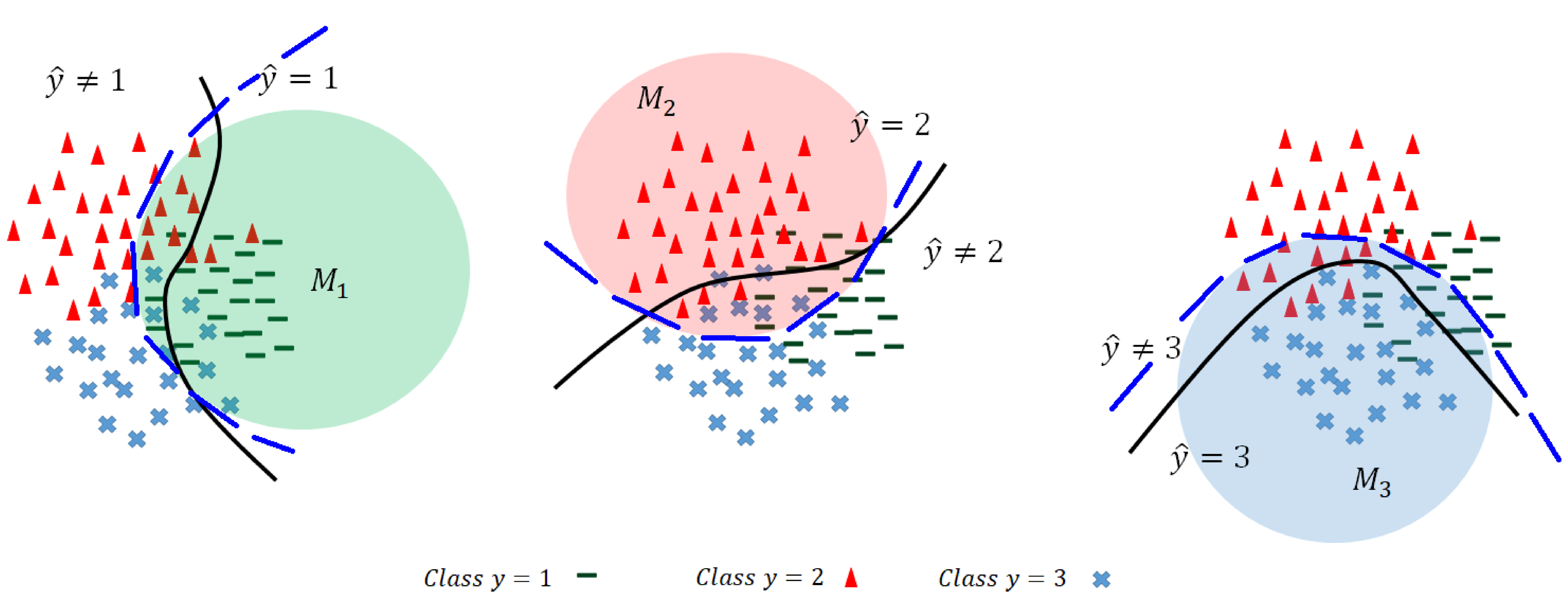
- **Post-processing method**
- **Preserves accuracy**
- **Low time and memory complexity**
- **Highly sensitive to the size and labels of validation set**
- **Gathering enough correctly labelled validation data is costly**

Unsupervised Temperature Scaling

- **How to make Temperature Scaling independent of labelled data?**

Key Insights:

- Estimate $Q(y=k|\mathbf{x})$ for each class k instead of $Q(y|\mathbf{x})$
- Use pre-trained model confidence to select the samples for each class distribution



Approach:

- First, use confidence of pre-trained model $S_{y=k}(\mathbf{x}_i)$ to select the samples of each class k distribution:

$$M_k = \{\mathbf{x}_i \mid S_{y=k}(\mathbf{x}_i) \geq \theta_k, \quad \mathbf{x}_i \in \mathcal{V}\}$$

- How to select threshold θ_k?

$$\theta_k = \bar{S}_{y=k} + \sqrt{\frac{1}{|\mathcal{U}|}\sum_{\mathbf{x}_i \in \mathcal{U}} (\bar{S}_{y=k} - S_{y=k}(\mathbf{x}_i))^2}\,, \text{where } \mathcal{U} = \{\mathbf{x}_i | \hat{y}_i \neq k\}$$

- Finally, find T^* that minimize NLL for each class distribution:

$$\mathcal{L}_{UTS} = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in M_k} \overbrace{-\log\Big(S_{y=k}(\mathbf{x}_i, T)\Big)}^{\text{NLL}}$$

$$T^* = \arg\min_T (\mathcal{L}_{UTS})$$

Analysis of samples' distribution selected in M_k

- Samples inside M_k are categorized in two groups:
  - The samples $\hat{y} = k$ which are from $Q(y = k|\mathbf{x})$
  - The samples $\hat{y} \neq k$ which are located near to the decision boundary

From Bayes rule, we have:

$$Q(\mathbf{x}, y = k) = \frac{Q(y = k|\mathbf{x})}{Q(y \neq k|\mathbf{x})} Q(\mathbf{x}, y \neq k).$$

- Notice that $Q(y = k|\mathbf{x}) \simeq Q(y \neq k|\mathbf{x})$ for samples located near to the decision boundary.
- Then, we can use samples generated from distributions $Q(\mathbf{x}, y = k)$ or $Q(\mathbf{x}, y \neq k)$ interchangeably.

Experiments

Benchmark Data

Model	Dataset	Accuracy	Uncalibrated NLL	Uncalibrated ECE	TS NLL	TS ECE	TS T	UTS NLL	UTS ECE	UTS T
DenseNet40	CIFAR10	92.61%	0.286	4.089	0.234	3.241	2.505	**0.221**	**0.773**	1.899
DenseNet40	CIFAR100	71.73%	1.088	8.456	**1.000**	**1.148**	1.450	1.001	1.945	1.493
DenseNet100	CIFAR10	95.06%	0.199	2.618	**0.156**	**0.594**	1.801	0.171	3.180	2.489
DenseNet100	CIFAR100	76.21%	1.119	11.969	0.886	4.742	2.178	**0.878**	**2.766**	1.694
DenseNet100	SVHN	95.72%	0.181	1.630	0.164	**0.615**	1.407	**0.162**	1.074	1.552
ResNet110	CIFAR10	93.71%	0.312	4.343	0.228	4.298	2.960	**0.207**	**1.465**	2.009
ResNet110	CIFAR100	70.31%	1.248	12.752	**1.051**	**1.804**	1.801	1.055	2.796	1.562
ResNet110	SVHN	96.06%	0.209	2.697	0.158	1.552	2.090	**0.152**	**0.550**	1.758
WideResNet32	CIFAR100	75.41%	1.166	13.406	0.909	**4.096**	2.243	**0.905**	4.872	1.651

This work was supported through funding from: NSERC-Canada, Mitacs, Prompt-Quebec, and E Machine Learning.