Progress Report

Rahul Rahaman

Department of Statistics and Applied Probability
National University of Singapore

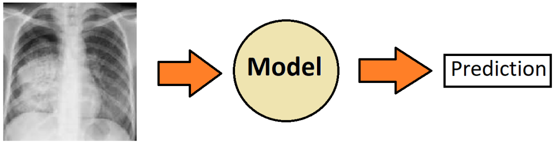
July 22, 2020

Table of Contents

Table of Contents

Introduction

Let us think of a model that predicts whether a patient is suffering from Tuberculosis by looking at a chest X-ray image.



After seeing an image, the model outputs a probability 0.7. What can a user (e.g. a Doctor) do with this number? Should one be 70% sure that the patient has Tuberculosis or is it true that in 7 out of 10 cases the patient will have Tuberculosis?

Introduction: Confidence

- Consider a classification task with $C \geq 2$ possible classes $\mathcal{Y} \equiv \{1, \dots, C\}$.
- For a sample $x \in \mathcal{X}$, the quantity $\mathbf{p}(x) \in \Delta_C = \{\mathbf{p} \in \mathbb{R}_+^C : p_1 + \ldots + p_C = 1\}$ represents a probabilistic prediction.
- It is often obtained as $\mathbf{p}(x) = \sigma_{\mathrm{SM}}[\mathbf{f_w}(x)]$ for a neural network $\mathbf{f_w} : \mathcal{X} \to \mathbb{R}^C$ with weight $\mathbf{w} \in \mathbb{R}^D$ and softmax function $\sigma_{\mathrm{SM}} : \mathbb{R}^C \to \Delta_C$.
- We set $\widehat{y}(x) \equiv \arg\max \mathbf{p}(x)$ and $\widehat{p}(x) = \max \mathbf{p}(x)$.

The term $\widehat{p}(x)$ is commonly referred to as *Confidence*.
Example: In our first example, $p = 0.7$ was the confidence.

Introduction: Accuracy

Accuracy: Given a confidence, the correctness of the model is termed as *Accuracy*. Formally, if p is model prediction, then

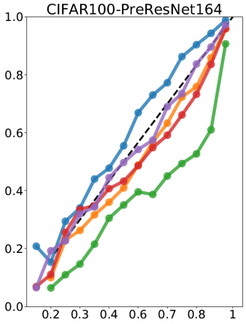
$$\mathrm{Acc}(p) = \mathbb{P}_{\mathcal{X}}[\text{model is correct}|p]$$

is called the accuracy, where the probability is computed with respect to the true data generating distribution. Practically, we do not know the true distribution, instead we estimate an empirical version

$$\widehat{\mathrm{Acc}}(p) = \frac{\#\text{of correctly predicted samples with conf} = p}{\#\text{of total samples with conf} = p}$$

Example: In our binary example, if 6 out of 10 patients with model prediction 0.7 actually had Tuberculosis, then $\widehat{\mathrm{Acc}}(0.7) = \frac{6}{10}$

Over-confidence and Under-confidence



According to the definition, Accuracy can be considered as a function $\mathrm{Acc} : [0,1] \to [0,1]$ from unit interval to the unit interval.
If plotted on the unit interval, the graph $\{(x, \mathrm{Acc}(x)) : x \in [0,1]\}$ is commonly referred to as the *Calibration Curve*.
A typical example of a calibration curve is shown here. Ideally one would like the curve to be as close as possible to the line $y = x$.
Example: In our original example, an user will expect that 7 out of 10 times the prediction 0.7 will be correct.

Over-confidence and Under-confidence

- If the curve is under (resp. over) the line $y = x$, then Accuracy is mostly less (resp. greater) than Confidence, and the model is said to be *over-confident* (resp. *under-confident*).
- Formally, a model is said to be over-confident (resp. under-confident) if $\mathrm{Acc}(p) \leq p$ (resp. $\geq p$) a.s.

Example: In our original example, $\widehat{\mathrm{Acc}}(0.7) = 0.6$, hence for $p = 0.7$ the model is over-confident.

Table of Contents

Expected Calibration Error (ECE)

- It is natural to use the distance between confidence and accuracy as a measure of miscalibration.
- One can work with the L_1 or L_2 distance.
- *Expected Calibration Error* (ECE) is the expected L_1 distance between confidence and accuracy

$$\mathrm{ECE} = \mathbb{E}_X[|\mathrm{Conf}(\widehat{p}_X) - \widehat{p}_X|]$$

where $\widehat{p}_X$ is the confidence for the random variable X.

Expected Calibration Error (ECE)

In practice, the ECE is calculated quite differently than its original definition, due to

1. unavailability of the true data distribution
2. availability of the Accuracy value only for a specific set of observed dataset.

For a partition $0 = c_0 < \ldots < c_M = 1$ of the unit interval and a labelled set $\{x_i, y_i\}_{i=1}^N$, set $B_m = \{i : c_{m-1} < \widehat{p}(x_i) \leq c_m\}$ and $\mathrm{acc}_m = \frac{1}{|B_m|} \sum_{i \in B_m} \mathbf{1}(\widehat{y}(x_i) = y_i)$ and $\mathrm{conf}_m = \frac{1}{|B_m|} \sum_{i \in B_m} \widehat{p}(x_i)$. The quantities ECE is defined as

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left| \mathrm{conf}_m - \mathrm{acc}_m \right|. \tag{1}$$

Brier Score

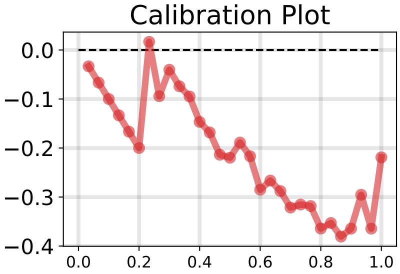
Another widely used metric for calculating calibration is Brier Score [Bri50], which calculates the L_2 distance between the predictions $\mathbf{p}(x)$ and its corresponding one-hot encoded target $\bar{y}$,

$$\text{Brier} := \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{p}(x_i) - \overline{y}_i\|_2^2$$

Usually these metrics are computed on unobserved test dataset. In conjunction with these metric, the usual *Accuracy* % and *Negative log-likelihood* are also used to check models generalization ability.

Neural Networks are Over-confident

Neural Networks are known to output over-confident predictions. The distribution of predicted probabilities is usually heavily shifted towards the right side of the unit interval.



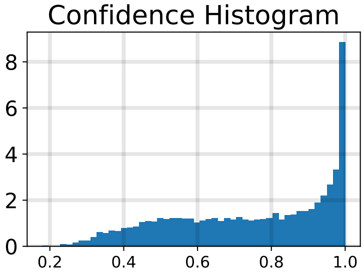


Table of Contents

Methods: Augmentation

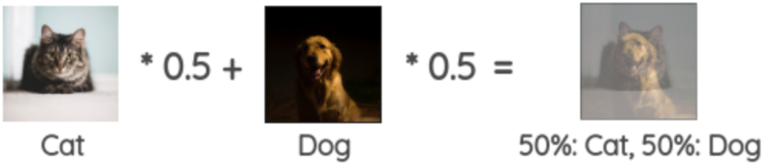
- Consider a training dataset $\mathcal{D} \equiv \{x_i, y_i\}_{i=1}^{N}$ and denote by $\overline{y} \in \Delta_C$ the one-hot encoded version of the label $y \in \mathcal{Y}$. A stochastic augmentation process $\text{Aug} : \mathcal{X} \times \Delta_C \to \mathcal{X} \times \Delta_C$ maps a pair $(x, \overline{y}) \in \mathcal{X} \times \Delta_C$ to another augmented pair $(x_\star, \overline{y}_\star)$.
- In computer vision, standard augmentation strategies include rotations, translations, brightness and contrast manipulations.

Methods: Augmentation

Mixup augmentation strategy [ZCDL17] augments a pair $(x, \overline{y}) \in \mathcal{X} \times \Delta_C$ to a different version $(x_\star, \overline{y}_\star)$ which is defined as

$$x_\star = \gamma\, x + (1-\gamma)\, x_J \qquad \text{and} \qquad \overline{y}_\star = \gamma\, \overline{y} + (1-\gamma)\, \overline{y}_J \tag{2}$$

for a random coefficient $\gamma \in (0,1)$ drawn from a fixed mixing distribution often chosen as Beta(α, α), and a random index J drawn uniformly within $\{1, \dots, N\}$.



Methods: Model Averaging

- Ensembling methods leverage a set of models by combining them into a aggregated model. In the context of deep learning, Bayesian averaging consists in weighting the predictions according to the Bayesian posterior $\pi(d\mathbf{w} \mid \mathcal{D}_{\text{train}})$ on the neural weights.
- Instead of finding an optimal set of weights by minimizing a loss function, predictions are averaged. Denoting by $\mathbf{p}_{\mathbf{w}}(x) \in \Delta_C$ the probabilistic prediction associated to sample $x \in \mathcal{X}$ and neural weight $\mathbf{w}$, the Bayesian approach advocates to consider

$$\text{(prediction)} \equiv \int \mathbf{p}_{\mathbf{w}}(x)\, \pi(d\mathbf{w} \mid \mathcal{D}_{\text{train}}) \in \Delta_C. \tag{3}$$

Methods: Model Averaging

- The posterior distribution $\pi(d\mathbf{w}|\mathcal{D}_{\text{train}})$ is multi-modal, high-dimensional, concentrated along low-dimensional structures, and any local exploration algorithm (eg. MCMC, Langevin dynamics and their variations) is bound to only explore a tiny fraction of the state space.
- Usually the integral is intractable in (3) and is approximated by a simple non-weighted average over several neural weights $\mathbf{w}_1, \ldots, \mathbf{w}_K$ simulated from the posterior distribution or found by minimizing the negative log-posterior, or some approximations of it, with standard optimization techniques:

$$\text{(prediction)} \;\equiv\; \frac{1}{K}\Big\{\mathbf{p}_{\mathbf{w}_1}(x) + \ldots + \mathbf{p}_{\mathbf{w}_K}(x)\Big\} \in \Delta_C. \tag{4}$$

Methods: Deep Ensembles

- Introduced by [LPB17], different modes of the loss function is reached through randomization in the optimization and initial conditions.
- Neural weights are randomly initialized, minibatch formation and orderings are randomized.
- These randomizations are usually enough for the stochastic optimization process to converge to different solutions.
- Currently it is the state-of-the art [NMC05] among all the model averaging methods, including different Bayesian methods.

Methods: Post-processing Calibration

- The article [GPSW17] proposes a class of post-processing calibration methods that extend the more standard *Platt Scaling* approach [Pla99]. *Temperature Scaling*, the simplest of these methods, transforms the probabilistic outputs $\mathbf{p}(x) \in \Delta_C$ into a tempered version $\text{Scale}[\mathbf{p}(x), \tau] \in \Delta_C$ defined through the scaling function

$$\text{Scale}(\mathbf{p}, \tau) \equiv \sigma_{\text{SM}}(\log \mathbf{p}/\tau), \tag{5}$$

  for a temperature parameter $\tau > 0$.
- The optimal parameter $\tau_\star > 0$ is usually found by minimizing a proper-scoring rules [GR07], often chosen as the negative log-likelihood, on a validation dataset.
- Crucially, during this post-processing step, the parameters of the probabilistic model are kept fixed: the only parameter being optimized is the temperature $\tau > 0$.

Table of Contents

Setups and Datasets

- We believe that current studies only focus on setups with high data availability. Hence these setups provide very limited challenge in terms of generalization as well as calibration.
- We instead work with low data setting where the models are extremely overconfident, hence it provides completely different viewpoint.
- In our study we use 5 popular image classification datasets with limited training observations. We use CIFAR10 (1000 samples), CIFAR100 (5000 samples), ImageNette [How], Imagewoof (1000 samples) and Diabetic Retinopathy [CB09] (5000 samples).

Setups and Datasets

Figure: *Left: CIFAR10 images, Middle: Imagenette dataset, Right: Diabetic Retinopathy*

Table of Contents

Mixup: Hyperparameter

Mixup augmentation has been shown to improve generalization as well as calibration properties. Recall that in mixup the samples are mixed according to the policy

$$x_\star = \gamma\, x + (1-\gamma)\, x_J \qquad \text{and} \qquad \overline{y}_\star = \gamma\, \overline{y} + (1-\gamma)\, \overline{y}_J \tag{6}$$

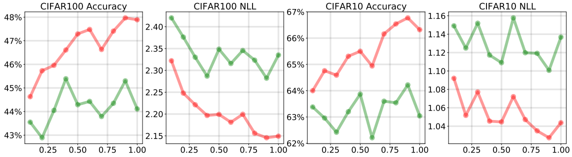
The co-efficient γ is usually simulated from Beta(α, α). The Beta distribution is peaked at $\{0, 1\}$ for $\alpha \in (0, 1)$ and becomes flat is *alpha* approaches 1, where it becomes an uniform distribution on the unit interval. Thus

- when α is small, samples are generated in close proximity of the original samples
- with bigger α, samples lie somewhere in the middle of the line joining the two samples

meaning, α controls how much exploration is done in the data manifold along lines joining samples.

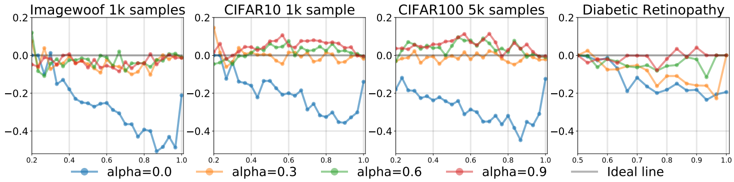
Mixup: Smoothness

- The gain that comes from the mixup augmentation policy is that it ensures a smoothness in the probabilistic output of the Neural Networks.
- In the vanilla training the learned function is typically sharp around seen examples. But mixup ensures smooth decrement of entropy. Hence the models become less over-confident.
- But this exploration comes with a cost. The distribution of the generated samples start to be quite different from the underlying true data distribution.
  As a simple example, if $\mathbf{Var}[x] = \mathbf{Var}[y]$ and $\gamma \sim \text{Beta}(\alpha, \alpha)$ then $\mathbf{Var}[\gamma x + (1-\gamma)y] = \frac{\alpha+1}{2\alpha+1}\mathbf{Var}[x]$



Mixup: Calibration

- The general belief is that low entropy mixup (i.e. mixup with $\alpha \in (0, 0.4]$) works best to reduce over-confidence. But the general setup that most of these experiments work with only provides models that are mildly over-confident.
- Our low data setting setup shows that higher α is warranted to tackle extreme over-confidence.



Mixup: Calibration

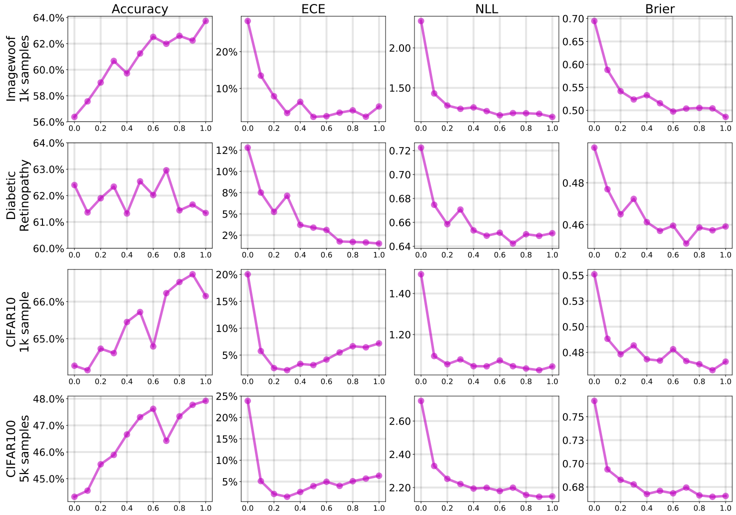


Figure: *Performance of mixup with different α.*

Distance To Training Data

- We examine several metrics (i.e. signed ECE (sECE), Negative Log-likelihood (NLL), entropy) as a function of the distance to the (small) training set $\mathcal{D}_{\mathrm{train}}$.
- we first use an unsupervised method (i.e. labels were not used) for learning a low-dimensional and semantically meaningful representation of dimension $d = 128$. For these experiments, we obtained a mapping $\Phi : \mathbb{R}^{32,32} \to \mathsf{S}^{128}$, where $\mathsf{S}^{128} \subset \mathbb{R}^{128}$ denotes the unit sphere in $\mathbb{R}^{128}$, with the *simCLR* method of [CKNH20].
- We used the distance $d(x, y) = \|\Phi(x) - \Phi(y)\|_2$, which in this case is equivalent to the cosine distance between the 128-dimensional representations of the CIFAR10 images x and y. The distance of a test image x to the training dataset is defined as $\min\{d(x, y_i) \ : \ y_i \in \mathcal{D}_{\mathrm{train}}\}$.

Distance To Training Data

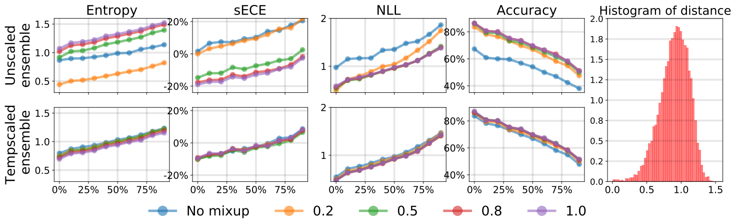


Figure: Deep Ensembles trained on $N = 1000$ CIFAR10 samples with different amount of mixup regularization. The x-axis represents a quantile of the distance to the CIFAR10 training set. In the second row, before averaging the predictions of the members of the ensemble, each individual network is first temperature scaled on a validation set of size $N_{val} = 50$

Distance To Training Data

- Not surprisingly, we note that the average Entropy, Negative Log-likelihood and Error Rate all increase as test samples are chosen further away from the training set.
- the predictions associated to samples chosen further away from the training set have a higher sECE. This indicates that the *over*-confidence of the predictions increases with the distance to the training set. In other words, even if the entropy increases as the distance increases (as it should), calibration issues do not vanish as the distance to the training set increases.
- increasing the amount of mixup augmentation consistently leads to an increase in entropy, decrease in over-confidence (i.e. sECE), as well as a more accurate predictions (lower NLL and higher accuracy).
- the second row indicates that a post-processing temperature scaling for the individual models almost washes-out all the differences due to the mixup-augmentation scheme.

Table of Contents

Model Averaging and Calibration

- It has been observed in several studies that averaging the probabilistic predictions of a set of independently trained neural networks, i.e. deep-ensembles, often leads to more accurate and better-calibrated
  forecasts [LPB17, BC17, LPC$^+$15, SLJ$^+$15, FHL19].
- The general belief about model averaging leading to calibrated model is so strong that almost every work about Bayesian Model Averaging uses calibration metrics as a way to gauge the efficacy of different posterior approximation methods.
- [NMC05] shows that among all the Deep Model averaging methods, Deep Ensemble performs significantly better than the other methods like MCD, Variational Approximation, other Gaussian Approximations. Hence we choose Deep Ensemble as our choice of method for analysis

Model Averaging and Calibration

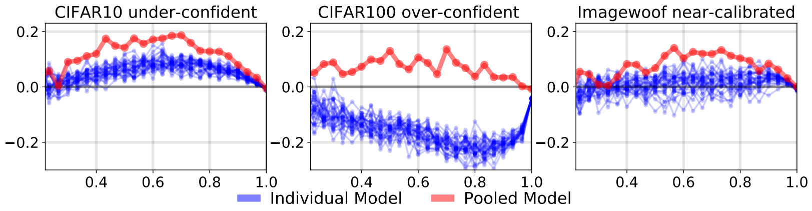


Figure: Reliability Curves with confidence conf_m *on the x-axis and difference* $(\text{acc}_m - \text{conf}_m)$ *on the y-axis. The plots display the reliability curves of* $K = 30$ *individual networks, as well as the pooled estimates obtained by averaging the K individual predictions. This linear averaging leads to consistently less confident predictions (i.e. higer values of* $(\text{acc}_m - \text{conf}_m)$*). It is only beneficial to calibration when each network is over-confident. It is typically detrimental to calibration when the individual networks are already calibrated, or under-confident.*

Reasons: Concavity of the Entropy Functional

In order to gain some insights into this phenomenon, recall the definition of the entropy functional $\mathcal{H} : \Delta_C \to \mathbb{R}$,

$$\mathcal{H}(\mathbf{p}) \;=\; -\sum_{k=1}^{C} p_k \, \log p_k . \tag{7}$$

The entropy functional is concave on the probability simplex Δ_C, i.e. $\mathcal{H}(\lambda \mathbf{p} + (1-\lambda)\,\mathbf{q}) \geq \lambda\,\mathcal{H}(\mathbf{p}) + (1-\lambda)\,\mathcal{H}(\mathbf{q})$ for any $\mathbf{p}, \mathbf{q} \in \Delta_C$.
Furthermore, tempering a probability distribution $\mathbf{p}$ leads to increase in entropy if $\tau > 1$, as can be proved by examining the derivative of the function $\tau \mapsto \mathcal{H}[\mathbf{p}^{1/\tau}]$.
The entropy functional is consequently a natural surrogate measure of (lack of) confidence. The concavity property of the entropy functional shows that ensembling a set of K individual networks leads to predictions whose entropies are higher than the average of the entropies of the individual predictions.

Reasons: Deviation from Calibration (DC)

To obtain a more quantitative understanding of this phenomenon, consider a binary classification framework. For a pair of random variables (X, Y), with $X \in \mathcal{X}$ and $Y \in \{-1, 1\}$, and a classification rule $p : \mathcal{X} \to [0, 1]$ that approximates the conditional probability $p_x \approx \mathbb{P}(Y = 1|X = x)$, define the *Deviation from Calibration* score as

$$\mathrm{DC}(p) \equiv \mathbb{E}\Big[\big(\mathbf{1}_{\{Y=1\}} - p_X\big)^2 - p_X(1 - p_X)\Big]. \tag{8}$$

The term $\mathbb{E}\Big[\big(\mathbf{1}_{\{Y=1\}} - p_X\big)^2\Big]$ is equivalent to the Brier score of the classification rule p and the quantity $\mathbb{E}[p_X(1 - p_X)]$ is an entropic term (i.e. large for predictions close to uniform). Note that DC can take both positive and negative values and $\mathrm{DC}(p) = 0$ for a well-calibrated classification rule, i.e. $p_x = \mathbb{P}(Y = 1|X = x)$ for all $x \in \mathcal{X}$.

Reasons: DC identity

Algebraic manipulations readily shows that, for a set of $K \geq 2$ classification rules $p^{(1)}, \ldots, p^{(K)}$ and non-negative weights $\omega_1 + \ldots + \omega_K = 1$, the linearly averaged classification rule $\sum_{i=1}^{K} \omega_i \, p^{(i)}$ satisfies

$$\mathrm{DC}\left(\sum_{i=1}^{K} \omega_i \, p^{(i)}\right) = \sum_{i=1}^{K} \omega_i \, \mathrm{DC}\left(p^{(i)}\right) - \underbrace{\sum_{i,j=1}^{K} \omega_i \omega_j \, \mathbb{E}\left[\left(p_X^{(i)} - p_X^{(j)}\right)^2\right]}_{\geq 0}.$$

This shows that averaging classifications rules decreases the DC score (i.e. the aggregated estimates are less confident). Furthermore, the more dissimilar the individual classification rules, the larger the decrease. Even if each individual model is well-calibrated, i.e. $\mathrm{DC}(p^{(i)}) = 0$ for $1 \leq i \leq K$, the averaged model is not well-calibrated as soon as at least two of them are not identical.

Pooling Methods

The standard average and median pooling of a set $\mathbf{p}^{1:K}$ of $K \geq 2$ probabilistic predictions $\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(K)} \in \Delta_C \subset \mathbb{R}^C$ are defined as

$$\mathbf{Agg}_{\text{avg}}(\mathbf{p}^{1:K}) = \frac{\mathbf{p}^1 + \ldots + \mathbf{p}^K}{K} \quad \text{and}$$
$$\mathbf{Agg}_{\text{med}}(\mathbf{p}^{1:K}) = \frac{\mathbf{median}(\mathbf{p}^1, \ldots, \mathbf{p}^K)}{Z},$$

for a normalization constant $Z > 0$, the median operation being executed component-wise over the $C \geq 2$ components.

Pooling Methods

Finally, $\textbf{trim}(z^{1:K})$, the trimmed mean [JW08] of K real numbers $z_1, \dots, z_K \in \mathbb{R}$, is obtained by first discarding the $1 \leq \kappa \leq K/2$ largest and smallest values before averaging the remaining elements. This means that $\textbf{trim}(z^{1:K}) = [z_{\sigma(\kappa+1)} + \dots z_{\sigma(K-\kappa-1)}]/(K-2\kappa)$ where $\sigma(\cdot)$ is a permutation such that $z_{\sigma(1)} \leq \dots \leq z_{\sigma(K)}$. The trimmed mean pooling method is consequently defined as

$$\textbf{Agg}_{\text{trim}}(\mathbf{p}^{1:K}) = \frac{\textbf{trim}(\mathbf{p}^1, \dots, \mathbf{p}^K)}{\mathrm{Z}}, \tag{9}$$

for a normalization constant $\mathrm{Z} > 0$, with the trimmed-averaging being executed component-wise.

Table of Contents

Possible Options

(A) Do nothing and hope that the averaging process intrinsically leads to better calibration

(B) Calibrate each individual network before aggregating all the results

(C) Simultaneously aggregate and calibrate the probabilistic forecasts of each individual model.

(D) Aggregate first the estimates of each individual model before eventually calibrating the pooled estimate.

Pool-Then-Calibrate

Any of the above-mentioned aggregation procedure can be used as a pooling strategy before fitting a temperature $\tau_\star$ by a minimizing proper scoring rules on a validation set. In all our experiment, we minimized the negative log-likelihood (i.e. cross-entropy). In other words, given a set $\mathbf{p}^{1:K}$ of $K \geq 2$ probabilistic forecasts, the final prediction is defined as

$$\mathbf{p}_\star \equiv \mathrm{Scale}\left[\mathbf{Agg}(\mathbf{p}^{1:K}), \tau_\star\right] \quad \text{where} \quad \mathrm{Scale}(\mathbf{p}, \tau) \equiv \sigma_{\mathrm{SM}}(\log \mathbf{p}/\tau). \tag{10}$$

Note that the aggregation procedure can be carried out entirely independently from the fitting of the optimal temperature $\tau_\star$.

Joint Pool-and-Calibrate

We also suggest learning the optimal temperature $\tau_\star$ concurrently with the aggregation procedure. The optimal temperature $\tau_\star$ is found by minimizing a proper scoring rule Score$(\cdot)$ on a validation set $\mathcal{D}_{\text{valid}} \equiv \{x_i, y_i\}_{i=1}^{N_{\text{val}}}$,

$$\tau_\star = \arg\min \Big\{ \tau \; \mapsto \; \frac{1}{\mathcal{D}_{\text{valid}}} \sum_{i \in \mathcal{D}_{\text{valid}}} \text{Score}(\mathbf{p}_i^\tau, y_i) \Big\}, \qquad (11)$$

where $\mathbf{p}_i^\tau \;=\; \mathbf{Agg}\big[\text{Scale}(\mathbf{p}^{1:K}(x_i), \tau)\big] \in \Delta_C$ denotes the aggregated probabilistic prediction for sample x_i. In all our experiments, we have found it computationally more efficient and robust to use a simple grid search for finding the optimal temperature; we used $n = 100$ temperatures equally spaced on a logarithmic scale in between $\tau_{\min} = 10^{-2}$ and $\tau_{\max} = 10$.

Importance of the Pooling and Calibration order

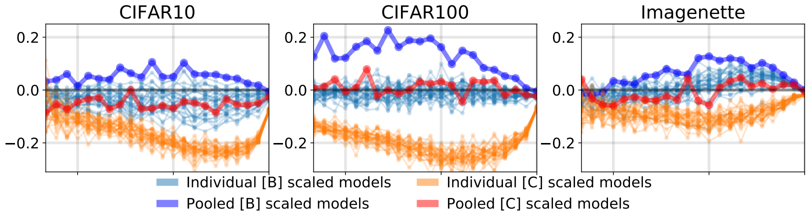


Figure: Calibration curve (x-axis confidence, y-axis difference between accuracy and confidence) of: **(light blue)** *each model calibrated with one temperature per model (i.e. individually temperature scaled),* **(dark blue)** *average of individually temperature scaled models (i.e. method* **[B]***),* **(orange)** *each model scaled with a global temperature obtained with method* **[C]**, **(red)** *result of method* **[C]** *that consists in simultaneously aggregating and calibrating the probabilistic forecasts of each individual model.* **Datasets:** *a train:validation split of size* 950 : 50 *was used for the CIFAR10 and IMAGENETTE datasets, and of size* 4700 : 300 *for the CIFAR100 dataset.*

Performance of Methods

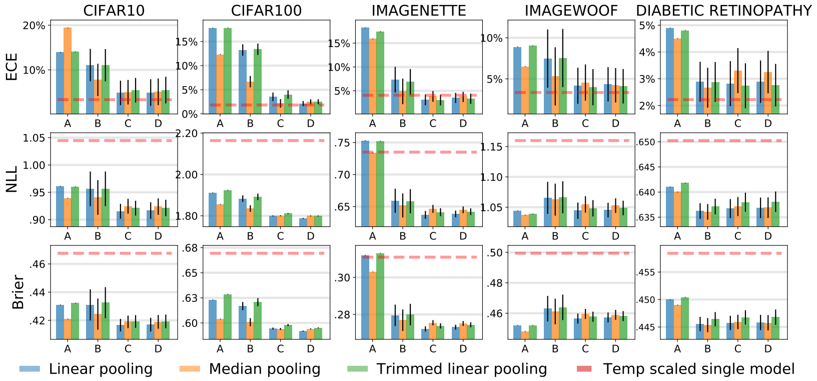


Figure: *Performance of different pooling strategies (***A-D***) with $K = 30$ models trained with mixup-augmentation ($\alpha = 1$) across multiple datasets. Experiments were executed 50 times on the same training data but different validation sets. The dashed red line represents a baseline performance when a single model was training with mixup augmentation ($\alpha = 1$) and post-processed with temperature scaling.*

Performance of Methods

CIFAR10 1000 samples

Metric	Group [A] Linear Pool	Group [B] Linear Pool	Group [C] Linear Pool	Group [D] Linear Pool
test acc	70.67	69.94	69.93	69.95
test ECE	13.9	11.1 ± 3.6	4.8 ± 2.7	4.9 ± 2.9
test NLL	0.961	$0.956 \pm .031$	$0.915 \pm .013$	$0.916 \pm .015$
test BRIER	0.431	$0.431 \pm .011$	$0.416 \pm .004$	$0.417 \pm .005$

CIFAR100 5000 samples

Metric	Group [A] Linear Pool	Group [B] Linear Pool	Group [C] Linear Pool	Group [D] Linear Pool
test acc	55.32	54.03	53.99	54.05
test ECE	17.8	13.1 ± 1.2	3.5 ± 0.9	$2.1 \pm .5$
test NLL	1.911	$1.883 \pm .016$	$1.799 \pm .002$	$1.787 \pm .002$
test BRIER	0.623	0.616 ± 0.004	$0.594 \pm .001$	$0.592 \pm .0$

*Table: Numerical table for the performance of linear pooling under different groups (**[A]**-**[D]**) and different datasets. The number of samples used for different setup are the same as mentioned in the main text. The mean and standard deviation is reported out of 50 different validation sets.*

Importance of Validation Dataset

It would be practically useful to be able to fit the temperature without relying on a validation set. We report that using the training set instead (obviously) does not lead to better calibrated models (i.e. the optimal temperature is close to $\tau_\star \approx 1$). We have tried to use a different amount of mixup-augmentation (and other types of augmentation) on the training set for fitting the temperature parameter, but have not been able to obtain satisfying results.

Size of the Ensemble

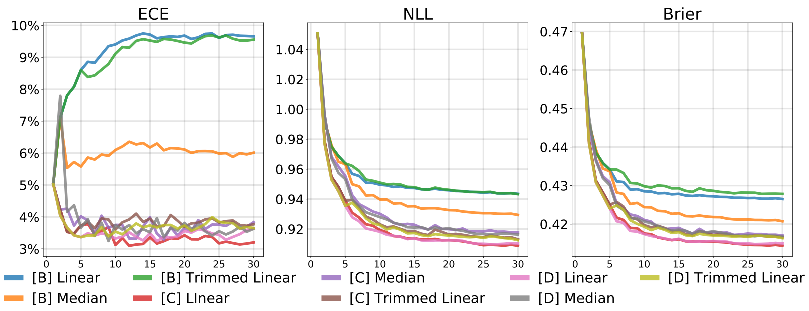


Figure: Comparison of methods **B-C-D** *on the CIFAR10 dataset with* $N = 1000$ *samples (950:50 split). The x-axis denotes the number of models. To avoid clutter and due to significantly worse performance, method* **[A]** *is omitted.*

Methods in group **[C]** and **[D]** performs similarly. For the CIFAR10 dataset, we observe that the performance under most metrics saturates for ensemble of sizes ≈ 15.

Ablation Study

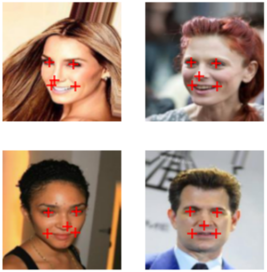
Metric	(Ours) 30 models temp scaled Augment + mixup	30 models mixup Augment	single model mixup Augment	single model no mixup Augment	single model no mixup no Augment
test acc	69.92 $\pm$ .04	**70.67**	66.45 $\pm$ .61	63.73 $\pm$ .51	49.85 $\pm$ .66
test ECE	**3.3** $\pm$ 1.9	13.9	7.03 $\pm$ .7	20.7 $\pm$ .4	23.4 $\pm$ 1.0
test NLL	**0.910** $\pm$ .012	0.961	1.03 $\pm$ .13	1.509 $\pm$ .017	1.770 $\pm$ .045
test BRIER	**0.414** $\pm$ .002	0.431	0.463 $\pm$ .005	0.556 $\pm$ .006	0.718 $\pm$ .009

Table: Ablation study performed on CIFAR10 1000 samples. For ensemble temp scaling we use 950 training samples and 50 validation set. For setups with variation we report metric mean and standard deviation.

Table of Contents

Motivation

- One of the many core and long-studied objectives of computer vision is to learn object structures from images.
- Earlier studies in computer vision relied on hand-crafted features that aimed to describe local information around a pixel.
- With the recent advancements and outstanding performance of deep learning based models, feature creation has become data-driven rather than based on domain knowledge.


(a) Identifying facial keypoints

(b) change in expression

(c) Matching different images

Motivation

- To use data driven features for the task of landmark detection, a substantial amount of data is required. Owing to technological advancements and online databases, the availability of images and videos is no longer a challenge.
- But the presence of raw, unlabelled datasets does not help models that require supervision for training. Labeling images for landmarks or tagging skeletons of human pose in videos at the pixel level is quite challenging and costly.
- Under such settings, it is very natural to use unsupervised learning. In this text, we review some of the landmark detection methods that come under this category and describe a few of our experiments.

Table of Contents

Methods

The recent works on landmark detection can mainly be divided into a few categories which we describe in this section.

- Regression based models
- Consistency based methods
- Methods based on local features
- Landmark detection through image reconstruction

Regression Based Methods

Introduced in [NHMP18], this method provided a unique way to get numerical coordinates of landmarks from fully convolutional unnormalized and unbounded landmark maps.

- Let $L(\mathbf{x})_{H' \times W'}$ be the output of a fully convolution neural network of an image $\mathbf{x}$ of dimension $H \times W$.
- The map $L(\mathbf{x})$ is then transformed into a two-dimensional probability distribution ψ by passing it through a spatial softmax layer.
- Finally, the coordinate $(\bar{u}, \bar{v})$ of the landmark is determined by

$$\bar{u} = \sum_u u \sum_v \psi(u, v; \mathbf{x}) \tag{12}$$

$$\bar{v} = \sum_v v \sum_u \psi(u, v; \mathbf{x}). \tag{13}$$

Regression Based Methods

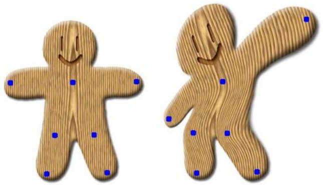
- These coordinates can then be used either in a form of consistency loss defined below, or can be used in a regression when performing supervised learning with known coordinates of the landmarks.
- This formulation of the landmark location provides meaningful and smooth gradients making it a suitable choice for most of the modern deep learning based landmark detection models.

Consistency/Invariance based methods

- Now let us take a function $g : \mathbb{R}^2 \to \mathbb{R}^2$, that maps the image $\mathbf{x}$ to a new image $\mathbf{x}' = g \circ \mathbf{x}'$ where g comes from several classes of image deformations (rotation, translation, affine, non-rigid transformations).
- The task of learning landmark location can be thought of learning a function $f_p(\mathbf{x})$ that takes an image as an input and returns the location of the landmark as output.

Then one would ideally like the invariance property i.e. for each keypoint p we would like to have

$$g \circ f_p(x) = f_p(g \circ x). \tag{14}$$



Consistency Loss

In order to use this consistency in an optimization environment, one must use it as a differentiable loss with *smooth* gradients. As suggested by [TBV17], a possible way is to obtain a landmark distribution ψ_p over a 2D space Λ i.e. for each $v \in \Lambda, \psi_p(v; \mathbf{x})$ gives the probability that $v = \varphi(p, \mathbf{x})$. Then a plausible way to implement a loss is to calculate

$$\mathcal{L}_{\text{consist}} = \mathbb{E}_{u \sim \psi_p(\cdot;\mathbf{x}'), v \sim \psi_p(\cdot;\mathbf{x})}\left[\left\|u - g(v)\right\|^2\right] \tag{15}$$

$$= \sum_u \sum_v \left\|u - g(v)\right\|^2 \cdot \psi_p(u; \mathbf{x}')\psi_p(v; \mathbf{x}). \tag{16}$$

Consistency Loss: Diversity constraint

- A pitfall of this loss is that the model can learn all landmarks to be at the center of the image.
- Thus a diversity constraint need to be used in conjunction with the mentioned consistency loss.
- One way to do that is to use

$$\mathcal{L}_{\text{diversity}} = \sum_{v} \left(\sum_{k=1}^{K} \psi_k(v; \mathbf{x}) - \max_k \psi_k(v; \mathbf{x}) \right). \tag{17}$$

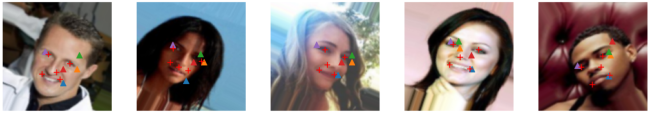


Figure: Baseline result from optimizing a linear combination of the consistency and diversity loss on CELEBA dataset. The triangles represent the predicted landmark location, and the crosses represent the actual landmark locations of five landmarks (two eyes, nose, two mouth endpoints).

Local Description based Methods

- In early computer vision literature, in order to match different images taken from different perspectives or deformation, hand-crafted features with local information were constructed [Can86, Lin12, Low99].
- Formally if $\mathbf{x}$ is an image of dimension $H \times W$, then feature $\mathbf{f}$ of dimension $H \times W \times d$ are created, i.e. each pixel $v \in \Lambda$ in $\mathbf{x}$ is represented by $\mathbf{f}(v) \in \mathbb{R}^d$, a d-dimensional feature that encodes the local information around the pixel u.
- These features are created with desired properties of being invariant to certain image transformations such as rotation, translation e.t.c.

Local Description based Methods

- To match two different images $\mathbf{x}, \mathbf{x}'$ the features of every two pair of pixel locations $v \in \mathbf{x}, u \in \mathbf{x}'$ are matched and a matching score m (of shape $H \times W \times H \times W$) is generated, where $m(u, v) \in \mathbb{R}$ denotes the strength of match between u, v.
- Finally, several filters are applied to choose keypoints that do not have multiple matches in the other image and remove noisy matches. Example of such filter criteria are *non-maximum suppression*, *mutual nearest neighbour*, *local random sample concensus* (RANSAC).
- This matching matrix can then be used either as a way to get back the deformation that was used to create $\mathbf{x}'$ from $\mathbf{x}$ or to directly use the matched landmarks as coordinate regression.

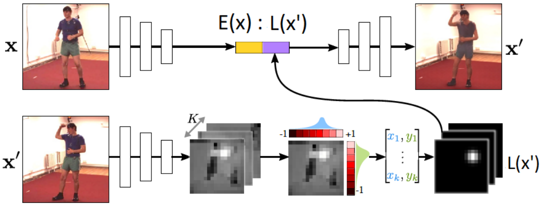
Landmark detection through Image generation

- The key idea behind this class of works is to generate landmarks in an unsupervised method.
- The landmarks should have the property that when provided with basic (possible from another warped/deformed image) structure of the object and the landmark locations (from the original image) then a full reconstruction of the original image should be possible.
- This method can also be thought of as a way to disentangle the appearance and the structure from images.

Landmark detection through Image generation

- The images $\mathbf{x}, \mathbf{x}'$ are first passed through image encoder and landmark encoder to obtain encoded image $E(\mathbf{x}) \in \mathbb{R}^{H' \times W' \times d}$ and encoded landmark $L(\mathbf{x}) \in \mathbb{R}^{H' \times W' \times K}$.
- Then $L(\mathbf{x}')$, are transformed into gaussian heatmaps H by first computing the coordinates of the landmark $l_k \in \Lambda \subset \mathbb{R}^2, 1 \leq k \leq K$ as in the method coordinate regression and then creating a 2D gaussian density image around it with intensity

$$H_k(u, v) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{d((u,v),l_k)}{2\sigma^2}} \tag{18}$$



Landmark detection through Image generation

- These appearance features and gaussian heatmaps are then concatenated together along the feature dimension (final dimension $H' \times W' \times (d + K)$) and passed through an image decoder D which then tries to render the image..
- This ensures that the landmarks contain sufficient information about the deformation that transformed $\mathbf{x} \to \mathbf{x}'$.
- The loss finally is

$$\mathcal{L} = \mathcal{L}_{\text{percep}}[\mathbf{x}', D(E(\mathbf{x}) : H(\mathbf{x}'))] \tag{19}$$

where $\mathcal{L}_{\text{percep}}$ is the perceptual loss between the two images.

Drawbacks / Inefficiencies

- Most of these methods based on image generation or relies on perceptual loss for generating realistic images. This leads to challenges when the images are not natural images (e.g. medical image, 3D scans). Also, the fact that the landmarks or hidden representation does not have direct relationship with the actual pixel locations, makes the networks learn landmarks that carries structural information in a non-trivial fashion.
- Due to the formation and manipulation of matrices of shape $R \times R$ where R is the resolution of the original image, matching methods typically suffer from heavy computational cost for high-resolution images and become near-infeasible.

Thank you!

Hamed Bonab and Fazli Can.
Less is more: a comprehensive framework for the number of components of ensemble classifiers.
arXiv preprint arXiv:1709.02925, 2017.

Glenn W Brier.
Verification of forecasts expressed in terms of probability.
Monthly weather review, 78(1):1–3, 1950.

J. Canny.
A computational approach to edge detection.
IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6):679–698, 1986.

Jorge Cuadros and George Bresnick.
EyePACS: An Adaptable Telemedicine System for Diabetic Retinopathy Screening.
Journal of Diabetes Science and Technology, 3(3):509–516, May 2009.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton.
A simple framework for contrastive learning of visual representations.
arXiv preprint arXiv:2002.05709, 2020.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan.
Deep ensembles: A loss landscape perspective.
arXiv preprint arXiv:1912.02757, 2019.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger.
On calibration of modern neural networks.
Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017, 2017.

Tilmann Gneiting and Adrian E Raftery.
Strictly proper scoring rules, prediction, and estimation.
Journal of the American statistical Association, 102(477):359–378, 2007.

Jeremy Howard.
Imagenette and imagewoof.

Victor Richmond R Jose and Robert L Winkler.
Simple robust averages of forecasts: Some empirical results.
International journal of forecasting, 24(1):163–169, 2008.

T. Lindeberg.
Scale Invariant Feature Transform.
Scholarpedia, 7(5):10491, 2012.
revision #153939.

D. G. Lowe.
Object recognition from local scale-invariant features.
In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

B. Lakshminarayanan, A. Pritzel, and C. Blundell.
Simple and scalable predictive uncertainty estimation using deep ensembles.
31st Conference on Neural Information Processing Systems, Long Beach, CA, USA, 2017.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra.
Why m heads are better than one: Training a diverse ensemble of deep networks.
arXiv preprint arXiv:1511.06314, 2015.

Aiden Nibali, Zhen He, Stuart Morgan, and Luke Prendergast.
Numerical Coordinate Regression with Convolutional Neural Networks.
2018.

A. Niculescu-Mizil and R. Caruana.
Predicting good probabilities with supervised learning.
Proceedings of the 22Nd International Conference on Machine Learning, ICML '05, pages 625–632, 2005.

J. Platt.
Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods.
Advances in Large Margin Classifiers, 10(3), 1999.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich.
Going deeper with convolutions.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

James Thewlis, Hakan Bilen, and Andrea Vedaldi.
Unsupervised Learning of Object Landmarks by Factorized Spatial Embeddings.
Proceedings of the IEEE International Conference on Computer Vision, 2017-October:3229–3238, 2017.

Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz.
mixup: Beyond empirical risk minimization.
CoRR, abs/1710.09412, 2017.