university of groningen

faculty of science and engineering

Probability Calibration and Out of Distribution Detection

Dr. Marco Zullich & Dr. Matias Valdenegro
Department of Artificial Intelligence
University of Groningen

February 8, 2024

Today's Agenda

1. Calibration
   - Introduction
   - Calibration Methods
   - Issues and Challenges

2. Out of Distribution Detection
   - Introduction
   - Evaluation of OOD Performance
   - Out of Distribution Detection Methods

Outline

Outline

What does Calibrated Mean?

According to Merriam-Webster dictionary, calibrate means:

1. to ascertain the caliber of (something)
2. to determine, rectify, or mark the graduations of (something, such as a thermometer tube)
3. to standardize (something, such as a measuring instrument) by determining the deviation from a standard so as to ascertain the proper correction factors
4. to adjust precisely for a particular function
5. to measure precisely

Calibration of Forecasters

From [Song et al., 2021], this is an important quotation.

> *Determining the degree to which a forecaster is well-calibrated cannot be done on a per-forecast basis, but rather requires looking at a sufficiently large and diverse set of forecasts.*

This means calibration can only be measured on a dataset, not directly in each prediction. It is fundamentally a frequentist measure.

What is Probability Calibration?

The general concept of calibration is that

$$\underset{\substack{\downarrow \\ \text{Probability predicted by your Model}}}{\text{Confidence}} = \underset{\substack{\downarrow \\ \text{Observed frequency of correct predictions}}}{\text{Accuracy}} \tag{1}$$

This simple equality should hold for all confidence levels.

Additionally, the interpretation of a predicted confidence score should be that is represents the probability of a prediction being correct.

What is Probability Calibration? - Classification

We can now formally define what calibration of probabilities for classification means. The following should hold for all confidence levels $\alpha \in [0, 1]$:

$$\mathbb{P}(y_\alpha \mid x) = |\hat{y}_\alpha|^{-1} \sum_{i \in \hat{y}_\alpha} 1[\hat{y}_i = y_i] \tag{2}$$

Where $\hat{y}_\alpha$ is the set of all predictions $\hat{y}$ with confidence α. This is usually where binning is required to group confidences around α since it is a continuous value.

In simple words, this means that the empirical probabilities are equal to the predicted probabilities or confidences by the model.

What is Probability Calibration? - Regression

For regression, we use confidence intervals:

$$\mathbb{P}(l_\alpha \leq y \leq u_\alpha) = \alpha \tag{3}$$

Where $[l_\alpha, u_\alpha]$ are the bounds of the confidence interval, which is a function of the confidence level α, and predicted mean $\hat{\mu}_i$ and variances $\hat{\sigma}_i^2$.
In terms of observed frequencies, with $\hat{y}_\alpha$ is the set of all predictions $\hat{y}$ around confidence interval with confidence α:

$$|\hat{y}_\alpha|^{-1} \sum_{i \in \hat{y}_\alpha} 1[\hat{y}_{\alpha,i}^l \leq y \leq \hat{y}_{\alpha,i}^h] = \alpha \tag{4}$$

Again this should hold for all levels of α.

Why Calibration Matters?

As we previously discussed, calibration in general means using a known scale.

For probabilities, you might be producing numbers in $[0, 1]$, but these are not probabilities, only when you can give them a probabilistic interpretation, and that they represent the probability of a prediction being correct, you can use these as probabilities and confidences.

A model might make random confidence predictions, which are not calibrated. We have extensively discussed the safety needs for calibrated probabilities, specially when using predictions to make decisions over humans.

Predicting Bad Calibrated Probabilities

Calibration is not a unique objective, the model still needs to make good predictions.

For example, in regression, if we predict the dataset mean, and a very large (or infinite) variance, this will always be calibrated (have zero calibration error), since it will always cover the true values. But now the confidence interval bounds do not have any meaningful information.

Basically, there are ways to *cheat* calibration, so calibration by itself is not the only goal during model training.

Why models are Miscalibrated? - Classification

This happens for multiple reasons:

Loss The cross-entropy loss with one-hot encoded labels, usually makes the predicted probabilities to be maximized (towards 1.0), without considering that they can be overconfident due to incorrect predictions.

Multi-Class Depending on the multi-class formulation (for example, SVMs with one-vs-all or all-vs-all), which break the assumptions used to evaluate calibration, and generally are hacks to transform a binary model into supporting multiple classes.

Not Probabilities Some machine learning models like SVMs do not output confidences, but abstract measures of confidence, which are generally not calibrated and are not probabilities.

Why models are Miscalibrated? - Regression

Loss The Gaussian negative log-likelihood supervises the variance outputs, and tries to cover data points according to the estimated noise, estimating only aleatoric uncertainty. Generally this estimation is pretty good, but once epistemic uncertainty is added, then the combined variances are most likely miscalibrated.

Confidence Levels The Gaussian NLL loss models a Gaussian, but it is not aware that we use particular confidence intervals to evaluate calibration, depending on aleatoric uncertainty, the true label might be outside of the expected confidence interval.

Why models are Miscalibrated? - Regression

The overall message is that models are not trained to be well calibrated as an explicit learning goal.

In general incorporating calibration is difficult due to calibration error being non-differentiable (due to the use of bins for confidence).

But there are methods to correct confidences after training (post-hoc) so they are better calibrated.

Calibration and Sources of Uncertainty

Note that calibration could be evaluated separately for epistemic and aleatoric uncertainty.

In general epistemic uncertainty is better expected to be calibrated, since it relates to the model and how it makes a prediction, so it should gauge the confidence level of the prediction being correct.

Aleatoric uncertainty has an effect of modeling the label noise, which might produce incorrect predictions (due to noise), or increase the size of confidence intervals.

Over and Under Confidence

Overconfidence

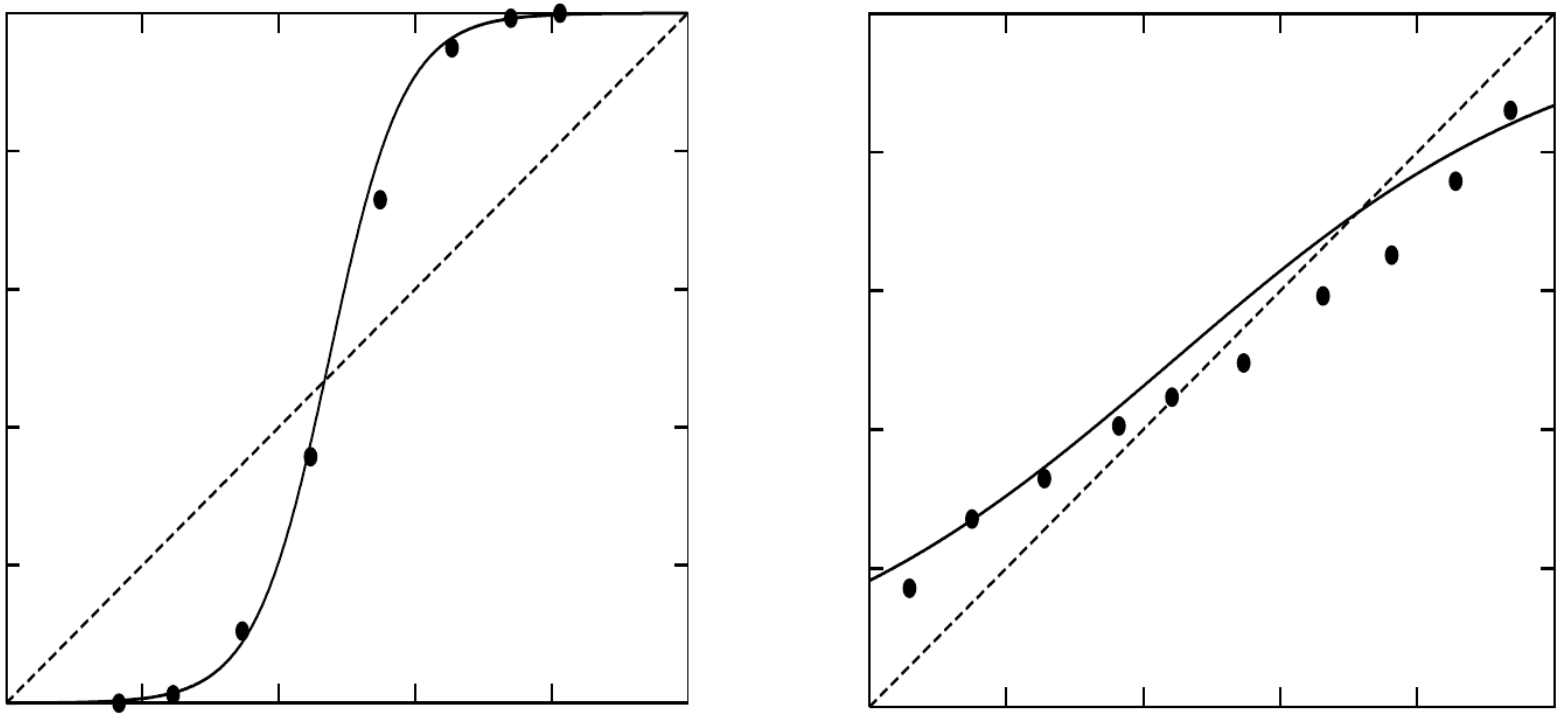
Regions where overall confidence is higher than accuracy indicate an overconfident model, so confidences should be lower than they are.
This is the worst case as a high confidence gives a false sense of security.

Underconfidence

Regions where overall confidence is lower than accuracy indicate that the model is underconfident, which means that the confidence should actually be higher than it is, in order to match accuracy.

Note that a model can be both under and over confident at the same time, but in different regions of the confidence space.

Over and Under Confidence



(a) Underconfidence

(b) Overconfidence

Fig. 4: Examples of under- and overconfident classifiers. The dots represent the reliability diagram, and the line shows the best-fit logistic curve. Note that the logistic sigmoid is a good fit for underconfident scores but not for overconfidence. Figures reproduced from Niculescu-Mizil and Caruana (2005).

Figure from [Song et al., 2021].

Types of Classification Calibration

Confidence Calibration

The class with largest probability is evaluated for calibration.

$$\sigma = \max_i p_i \tag{5}$$

Classwise Calibration

Each class needs to be individually calibrated, in a one-vs-all or one-vs-rest setting.

This means that calibration is evaluated as binary classification for each class, with one class versus the rest.

For this purpose a calibration error can be computed and evaluated for each class (independent of each other).

Types of Classification Calibration

Multi-class Calibration

All classes are jointly calibrated, and calibration error is evaluated for all classes at the same time.

For example, this would be combining a mean calibration error for all classes, and optimizing it jointly.

These definitions are ordered by strength of calibration, with Multi-class Calibration being the strongest form.

$$\text{Confidence} < \text{Classwise} < \text{Multiclass}$$

Regression Calibration

For regression, there is a different concept of calibration, using quantiles $\tau \in [0, 1]$:

$$\mathbb{P}(Y \leq g(x, \tau)) = \tau \tag{6}$$

Here g acts as a quantile function or inverse CDF, which the model can output, this is called quantile regression.

The quantiles can be scaled or changed using a calibration map, which can be learned to improve calibration.

Setting Number of Bins

One important detail for bin-based methods like computing calibration errors, is how to set the number of bins $N = |B|$?

There are multiple choices, as it is a hyper-parameter, so no unique way to do it.

One option is tune this parameter on the training set, select the value of N that minimizes the ECE.
Another option is to visually look at the calibration plot (including bins), and vary N, balancing smoothing and noise.

The effect of changing N is that a small N will oversmooth the plot, while a large N might have lots of bins with no or zero samples.

Decomposition of Proper Scoring Rules

All proper scoring rules can be decomposed into two interesting terms:

$$\underset{\substack{\downarrow \\ \text{Expected} \\ \text{Scoring Rule}}}{\mathbb{E}[d(S,Y)]} = \underset{\substack{\downarrow \\ \text{Expected} \\ \text{Calibration} \\ \text{Loss}}}{\mathbb{E}[d(S,C)]} + \underset{\substack{\downarrow \\ \text{Expected} \\ \text{Refinement} \\ \text{Loss}}}{\mathbb{E}[d(C,Y)]}$$

Where Y are the ground truth probabilities, S are the predicted probabilities of your model, and C are the calibrated probabilities (perfect empirical probabilities, which we usually do not know).

Here d is a divergence associated to the scoring rule. For the cross-entropy loss, d is the KL-divergence, and for the Brier score it is the mean squared error.

Decomposition of Proper Scoring Rules

In the previous decomposition, calibration loss is self explanatory. Refinement loss is also called *sharpness*, and is basically a measure of how wide the distribution is.
A sharp prediction is precise, and confidence indicates that the model is very confident, while a blunt prediction has a large variance and low confidence.

If we replace $C = Q$, where Q are the true posterior probabilities from a model, then:

$$\underset{\substack{\downarrow \\ \text{Expected} \\ \text{Scoring Rule}}}{\mathbb{E}[d(S,Y)]} = \underset{\substack{\downarrow \\ \text{Expected} \\ \text{Epistemic} \\ \text{Loss}}}{\mathbb{E}[d(S,Q)]} + \underset{\substack{\downarrow \\ \text{Expected} \\ \text{Aleatoric} \\ \text{Loss}}}{\mathbb{E}[d(Q,Y)]}$$

Decomposition of Proper Scoring Rules

Epistemic Uncertainty

Loss or error produced by the model, miscalibration is part of epistemic uncertainty.

Aleatoric Uncertainty

Loss or error produced by the data, including the difference between calibrated probabilities and ground truth probabilities.

$\uparrow$ aleatoric uncertainty will make it hard for the model to have a sharp (confident) prediction, while $\downarrow$ aleatoric uncertainty makes it easy to learn a confident/sharp predictions.

Outline

Concept

We have discussed method to evaluate calibration, but now the question is, how can we improve the calibration of a trained model? Or train a better calibrated model?

For this there are two major types of methods:

Post-Hoc Modify or recompute predicted probabilities to improve their calibration.

A priori Improve model structure or training to produce a better calibrated model.

Calibration Maps

A calibration map is a function $g : \mathbb{S}_{\mathbb{Y}} \to \mathbb{P}_{\mathbb{Y}}$, where the mapping is applied to the confidences produced by a model, and the mapping g is learned in a way that improves calibration of the predicted confidences.

- $\mathbb{S}$ is the uncalibrated confidence or probability space, this is produced by your model.
- $\mathbb{P}$ is the calibrated probability space on $\mathbb{Y}$, given by $\mathbb{P}_{\mathbb{Y}} = \{[s_0, s_1, ..., s_{C-1}] \,|\, s_i \geq 0, \sum_{i=0}^{C-1} s_i = 1]\}$.
- $\mathbb{Y}$ is the categorical space for C classes, given by $\mathbb{Y} = \{0, 1, ..., C-1\}$

Platt Scaling [Platt et al., 1999]

Platt Scaling was developed to produce calibrated probabilities for Support Vector Machines, but it can be used for any model. It uses the following calibration map:

$$g(x) = \frac{1}{1 + \exp(Ax + B)} \tag{7}$$

The parameters A and B are learned using gradient descent, using the binary cross-entropy loss. The labels y can be softened by:

$$y_+ = \frac{N_+ + 1}{N_+ + 2} \qquad y_- = \frac{1}{N_- + 2} \tag{8}$$

Where the $_+$ indicates positive class, and $_-$ the negative class, and N indicates the number of samples.

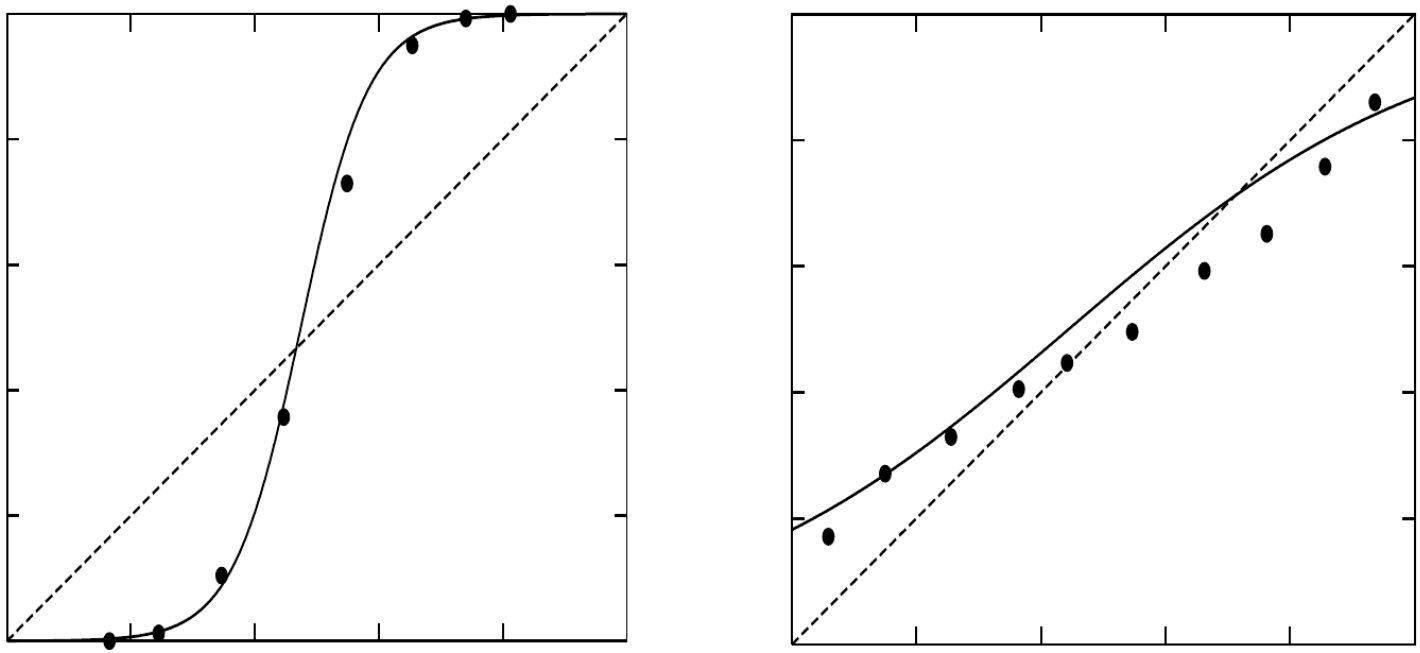
Training

For calibration to work properly and test for overfitting, a three way split of a dataset must be used. This applies to any calibration method.

To use test data, the calibration map should be trained on the training set, and evaluated on the validation set, including any hyper-parameter tuning. Then finally, a single evaluation should be made on test data. Missing any of these steps will cause leakage.

By working properly, I mean that any calibration map applied to your predictions, should generalize from one dataset to another.

Platt Scaling Example

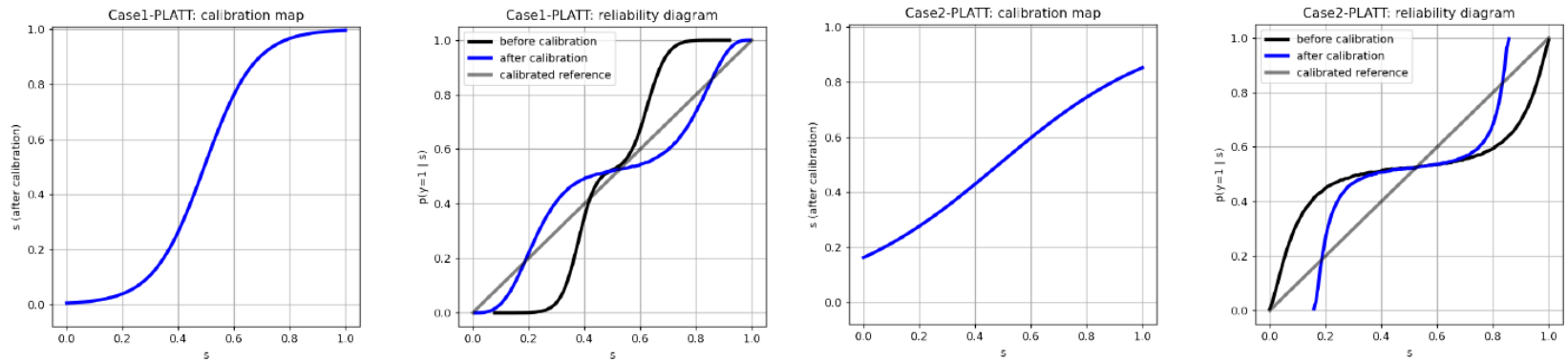


(a) Underconfidence (b) Overconfidence

Fig. 4: Examples of under- and overconfident classifiers. The dots represent the reliability diagram, and the line shows the best-fit logistic curve. Note that the logistic sigmoid is a good fit for underconfident scores but not for overconfidence. Figures reproduced from Niculescu-Mizil and Caruana (2005).

Figure from [Song et al., 2021]. This is an example of Platt Scaling applied to a reliability plot.

Platt Scaling Example



(a) Test case 1: under-confidence

(b) Test case 2: over-confidence

Figure from [Song et al., 2021]. This is a comparison for under and over confident classifiers, and it can be seen that platt scaling works differently on both cases. Work well on underconfidence, but fails in overconfidence.

Empirical Binning

When we estimate any calibration error metric, we also need to compute bins $\text{conf}(B_i)$ (confidences) and $\text{acc}(B_i)$ (empirical accuracies).

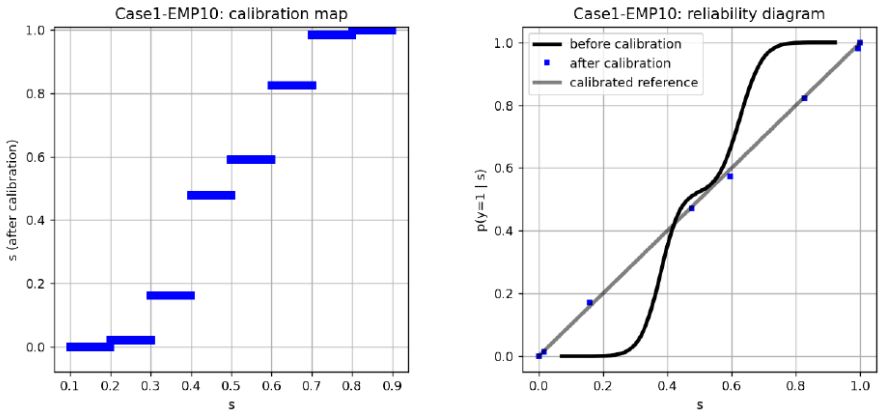
If we use the following calibration map:

$$g(x) = \text{acc}(B_i) \qquad \text{if } x \in B_i. \tag{9}$$

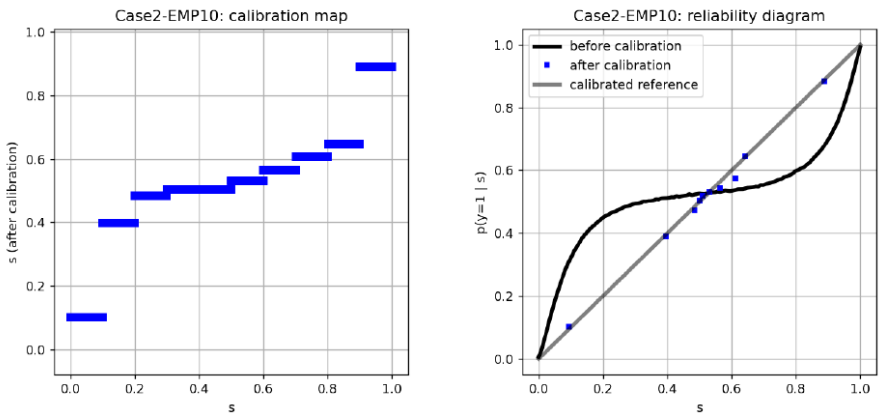
Basically we use a piecewise constant function that for confidences in each bin, it outputs the mean accuracy of that bin. This would definitely improve calibration of the model.

This a non-parametric calibration method.

Empirical Binning



(a) Test case 1: under-confidence



(b) Test case 2: over-confidence

Figure from [Song et al., 2021].

Isotonic Calibration

Isotonic regression is a method for regression of monotonic (increasing) functions, and generally we can assume that calibration maps are monotonic.

In Isotonic calibration, we use the same setup as empirical binning, but now the bin edges $B_i = [l_i, u_i]$ with $l_i < u_i$, are learned from the data instead of being fixed values that are usually set from a decided number of bins N. The bin sizes do not have to be equal.

This allows bins to adapt and better fit the true calibration map. This also a non-parametric calibration method.

Temperature Scaling

The softmax activation function also has a version that uses a temperature to scale the logits:

$$\text{softmax}(\mathbf{x}) = \left[\frac{\exp(\mathbf{x}_i / T)}{\sum_j \exp(\mathbf{x}_j / T)} \right]_i \quad \forall i \in [0, C-1] \qquad (10)$$

The logits x_i are divided by a temperature factor T. This has the effect of softening the output distribution for increasing T, and can improve calibration.

One interpretation of per-class temperatures is class imbalance, as each T_i will scale to counteract class imbalances and adjust confidences accordingly.

Temperature Softmax

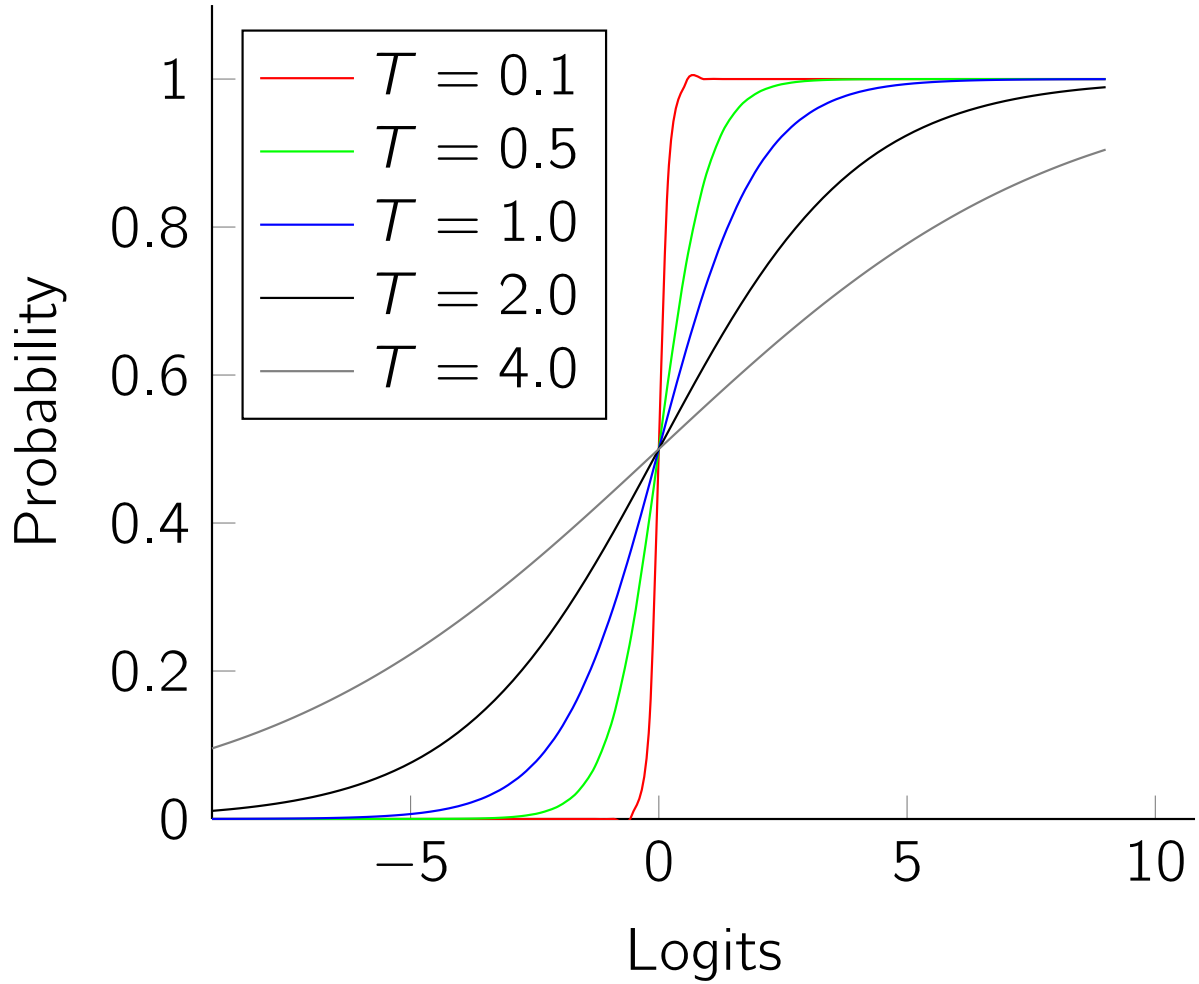


Figure: Plot of the Sigmoid Activation with Temperature, as an easy way to visualize how the Softmax with Temperature behaves (both have the same behavior).

Temperature Scaling

A model can be re-trained to optimize the T parameter, using another loss such as the expected calibration error (with fixed bins), in order to improve calibration. This would require to use non-gradient methods since the ECE is not differentiable. Empirical confidences can also be used as targets.

This can be done globally for all classes, with a common T, or to have a T_i specific for each class:

$$\text{softmax}(\mathbf{x}) = \left[\frac{\exp(\mathbf{x}_i / T_i)}{\sum_j \exp(\mathbf{x}_j / T_i)} \right]_i \quad \forall i \in [0, C-1] \qquad (11)$$

Where now T is a vector of length C that is optimized to improve the calibration of the model.

Mixup Training [Thulasidasan et al., 2019]

The concept of Mixup training is to train not only on training samples, but also in their neighborhoods. It is basically a data augmentation method:

$$\tilde{x} = \lambda x_i + (1 - \lambda) x_j \tag{12}$$

$$\tilde{y} = \lambda y_i + (1 - \lambda) y_j \tag{13}$$

Where x_i, x_j are pairs of training samples (features, images, etc), and y_i, y_j are their corresponding one-hot encoded labels. λ is the mixing factor which is sampled from $\lambda \sim \text{Beta}(\alpha, \alpha)$, where $\alpha \in [0, \infty]$ is a hyper-parameter, usually set to $[0.1, 0.4]$.

Mixup Training

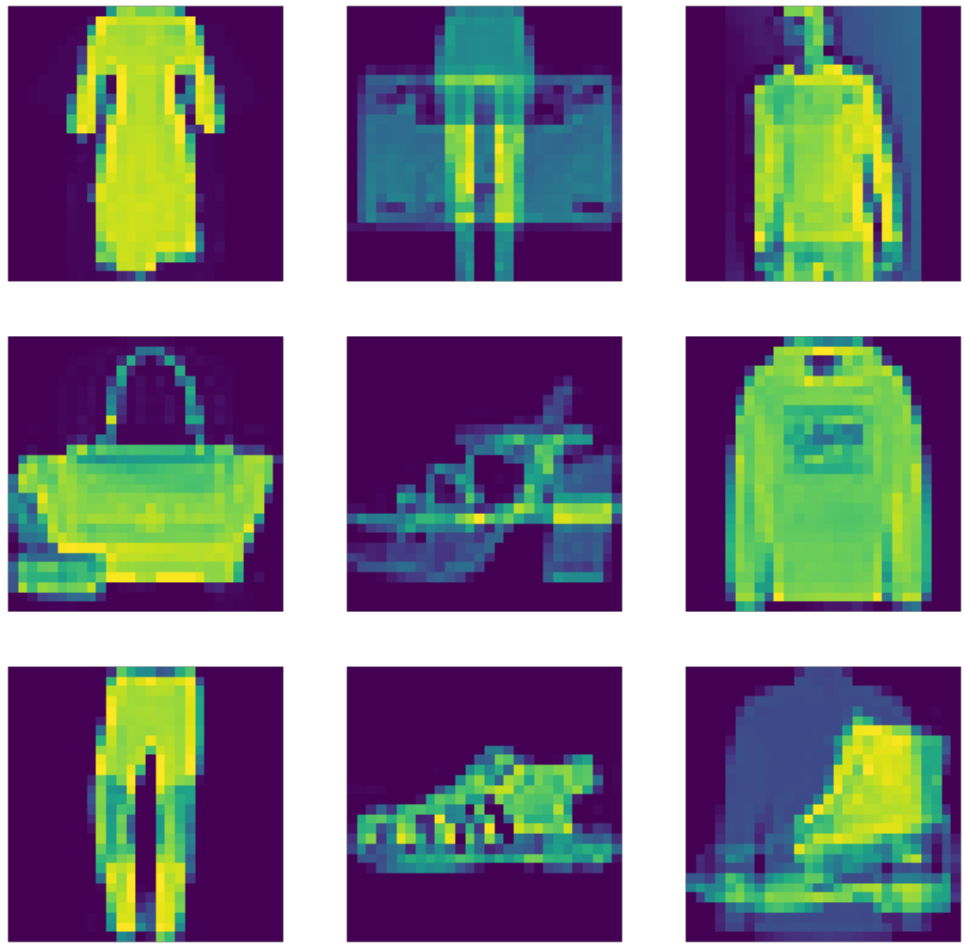


Figure from `https://keras.io/examples/vision/mixup/`

Mixup Training [Thulasidasan et al., 2019]

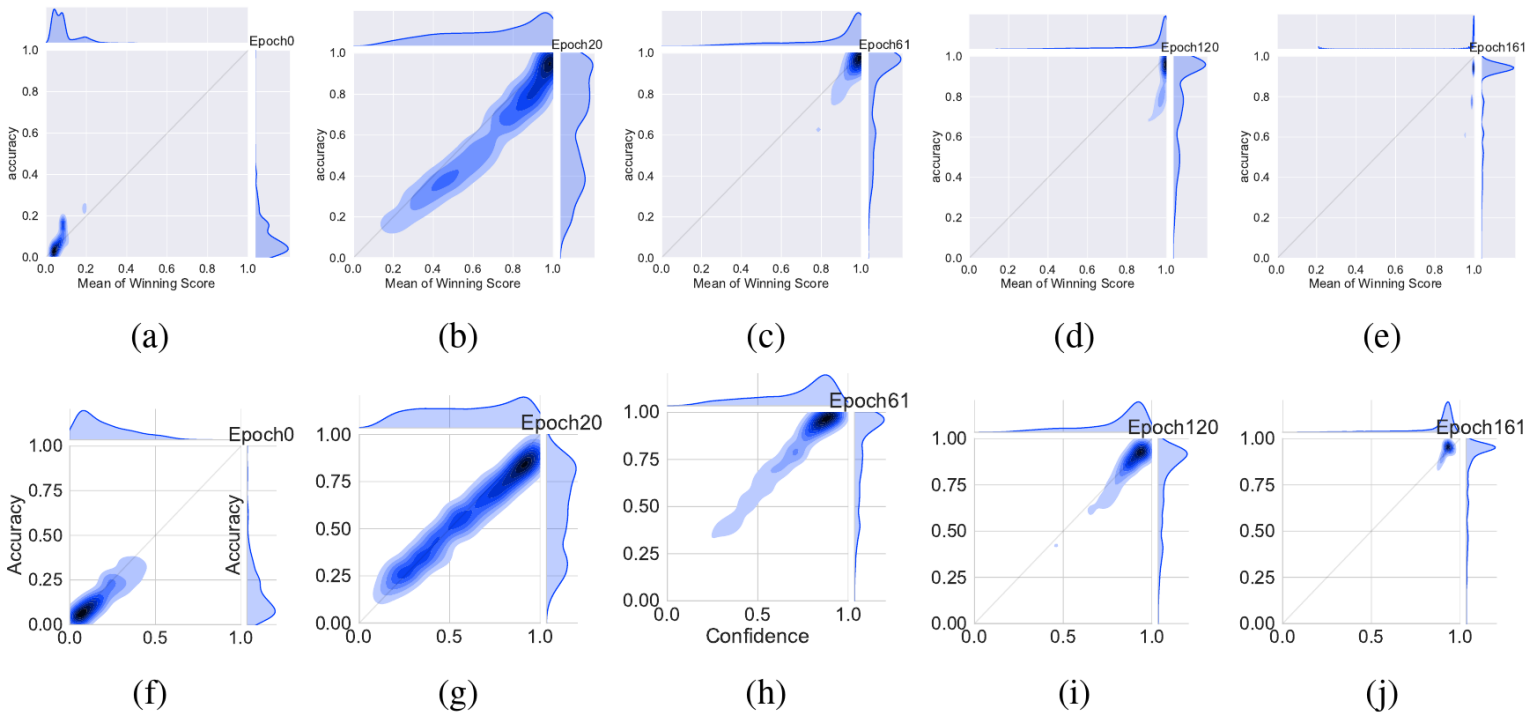
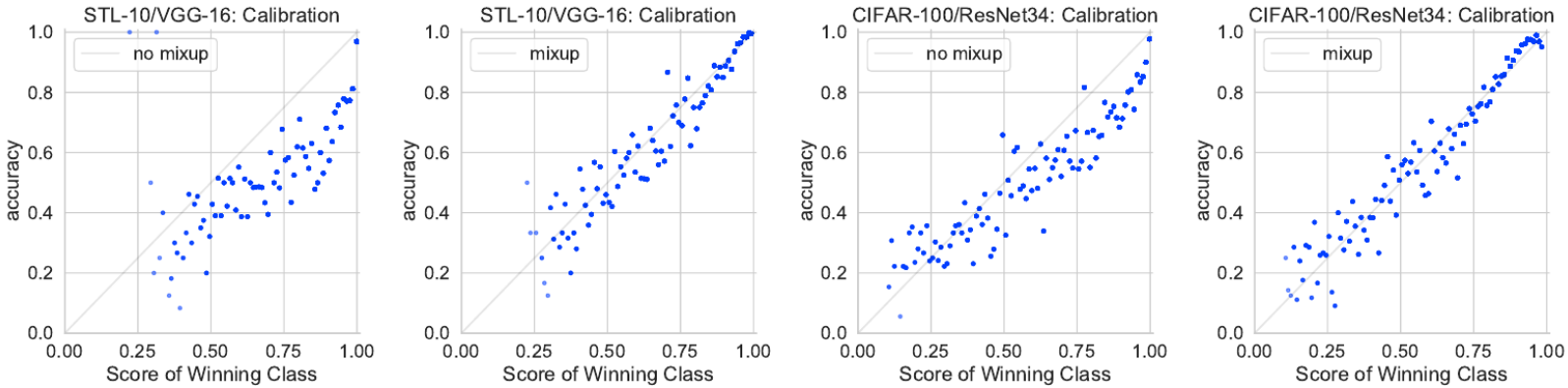
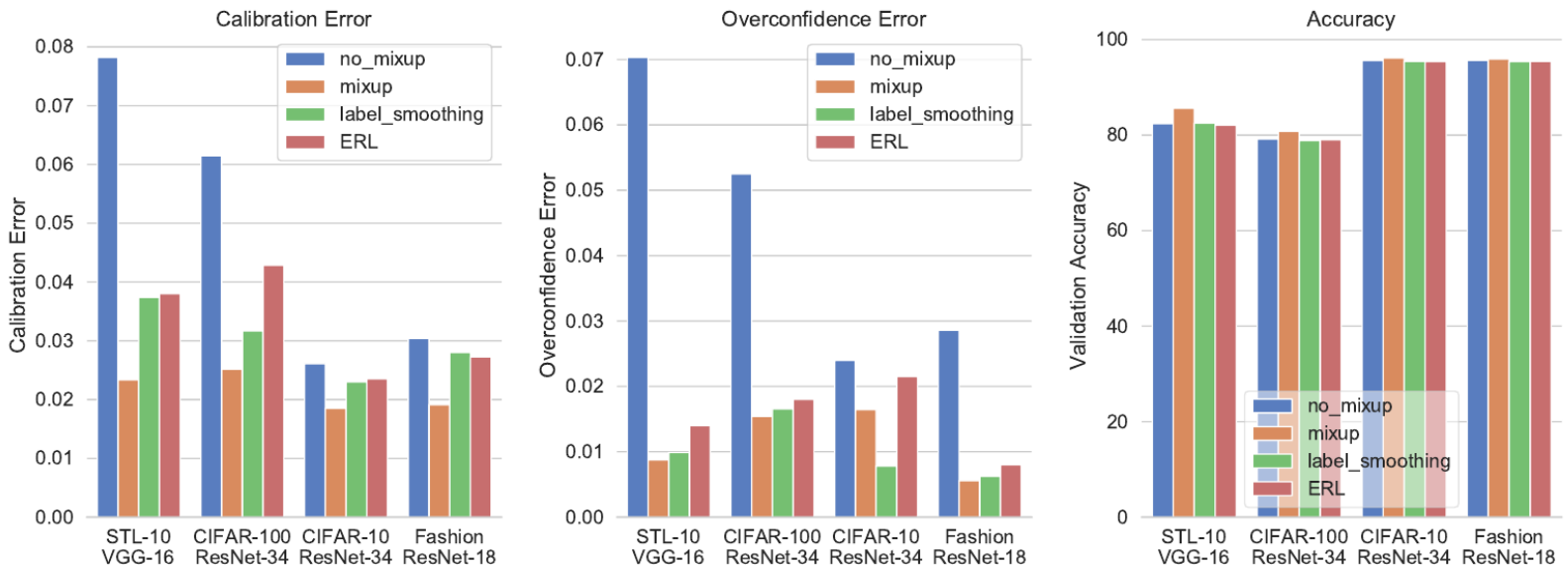


Figure 1: Joint density plot of accuracy vs confidence (captured by the winning softmax score) on the CIFAR-100 validation set at different training epochs for the VGG-16 deep neural network. **Top Row:** In regular training, the DNN moves from under-confidence, at the beginning of training, to overconfidence at the end. A well-calibrated classifier would have most of the density lying on the $x = y$ gray line. **Bottom Row:** Training with mixup on the same architecture and dataset. At corresponding epochs, the network is much better calibrated.

Mixup Training [Thulasidasan et al., 2019]



(a) (b) (c) (d)



(e) (f) (g)

Improving Regression Calibration

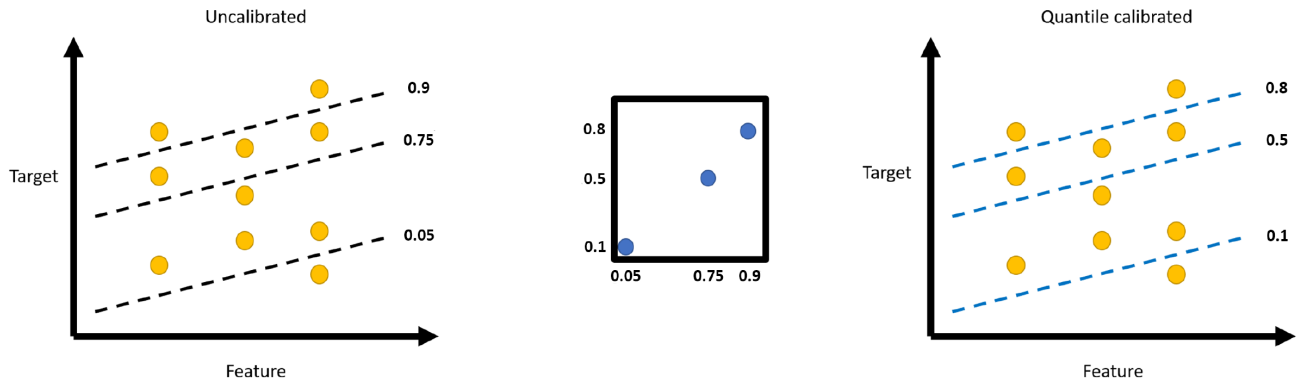


Fig. 27: An example of quantile calibration, calibrating the quantiles corresponding to $\tau = 0.05, 0.75, 0.9$ (left plot) using a calibration map (middle plot) into quantiles corresponding to $\tau = 0.1, 0.5, 0.8$ (right plot).

Figure from [Song et al., 2021]. This is a visual comparison of how quantile calibration works. Note how the quantiles change through a calibration map (the box in the middle).

Outline

Generalization

The biggest challenge for any machine learning method, including calibration, is that of generalization.

Generally speaking, that a model/method is well calibrated in one dataset, does not mean that it will be well calibrated in other datasets, particularly a test set, or sets very far from the training distribution.

There are no guarantees of adequate performance in the test set.

Can we Trust our Uncertainties? [Ovadia et al., 2019]

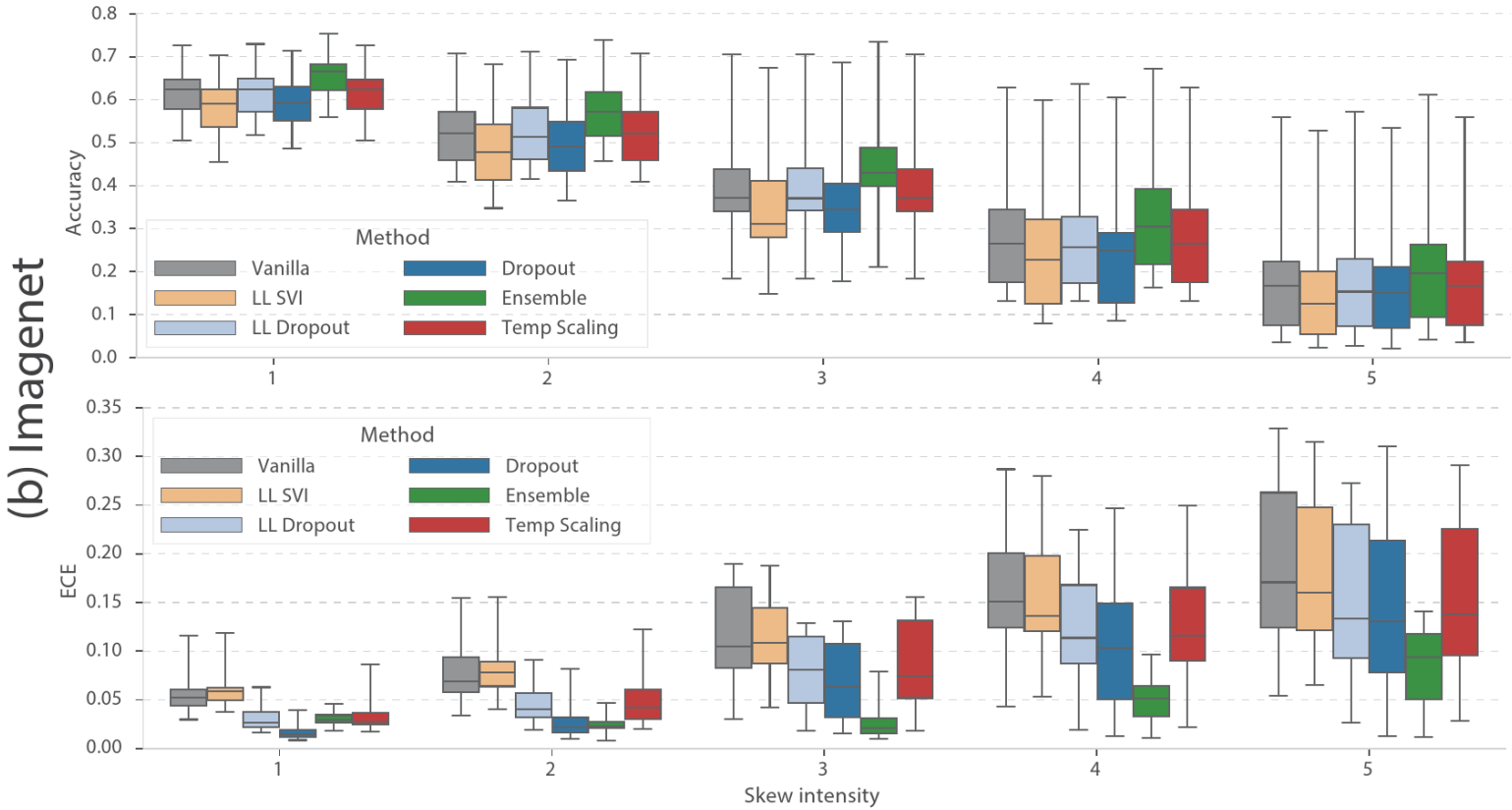


Figure 3: Calibration under distributional shift: boxplots showing a detailed comparison of Brier score and ECE under all types of corruptions on (a) CIFAR-10 and (b) ImageNet. Each box shows the quartiles summarizing the results across all types of skew while the error bars indicate the min and max across different skew types. Figures showing additional metrics are provided in Figures S5 (CIFAR-10) and S6 (ImageNet). Tables for numerical comparisons are provided in Appendix E.

Questions to Think About Calibration

- Since we use calibration maps, what kind of issues can happen by using parametric model?
- Using Empirical Binning, why is a model not perfectly calibrated?
- Compare the concepts for improving calibration (for example, Temperature Scaling vs Platt Scaling). How are these methods different or similar?
- Can overfitting happen when training a calibration map?

Outline

Outline

Concept

In this course we have extensively explored that confidence should be proportional to error or be a probability of correct classification.

We also know that when a model is given an input that was not trained on, it will answer incorrectly.

The question is then, how to detect when a model is given inputs that are very different from its training set?

This is the concept of Out of Distribution Detection.

Model Extrapolation

One important related concept is interpolation and extrapolation.

When a model is given an input that is similar to what it learned from the training set, then the model is **interpolating**, and generally this works very well using neural networks.

But when a model is given input that is dissimilar or outside of its training set, then it **extrapolates**, basically guessing, and this is the situation where failure is almost guaranteed, and needs to be detected at inference time.

Closed Set Assumption

There is a big implicit assumption on training most machine learning models.

The training, validation, and test set contains a pre-defined set of classes or a distribution of values, and no other distribution or classes exist.

This is called the closed set assumption, the opposite is the open set assumption or setting.

Model vs World Knowledge

Model/World	Known	Unknown
Known	Known Knowns	Known Unknowns
Unknown	Unknown Knowns	Unknown Unknowns

In the closed set assumption, there are only known knowns, and the world is assumed to contain only the known knowledge.

In the open set setting, there are all combinations, with Unknown unknowns being the most difficult to handle, basically, things the model is not aware that it does not know.

Model vs World Knowledge

Known Knowns

Basic and only knowledge in the closed set assumption.

Known Unknowns

The model is aware that there is knowledge that is unknown, for example, there might be more classes than the ones used in the training set. This is the basic open set recognition setting.

These can also be seen as risks that the model is aware of.

Model vs World Knowledge

Unknown Knowns

The model is unaware of some of the things it knows.

This relates more to the designer of the model than the model itself, for example, Transformers have properties that were there, but we were not aware of them.

Unknown Unknowns

This is the most difficult setting, as there is knowledge that is unknown, and the model is completely unaware of this lack of knowledge.

The open set recognition setting partially covers this case, as there are possibly infinite classes, and it is not possible to enumerate them all.

Related to the frame problem (enumerate all effects of an action in an environment).

Concept of Anomaly, Novelty, OSR, OOD [Yang et al., 2021]

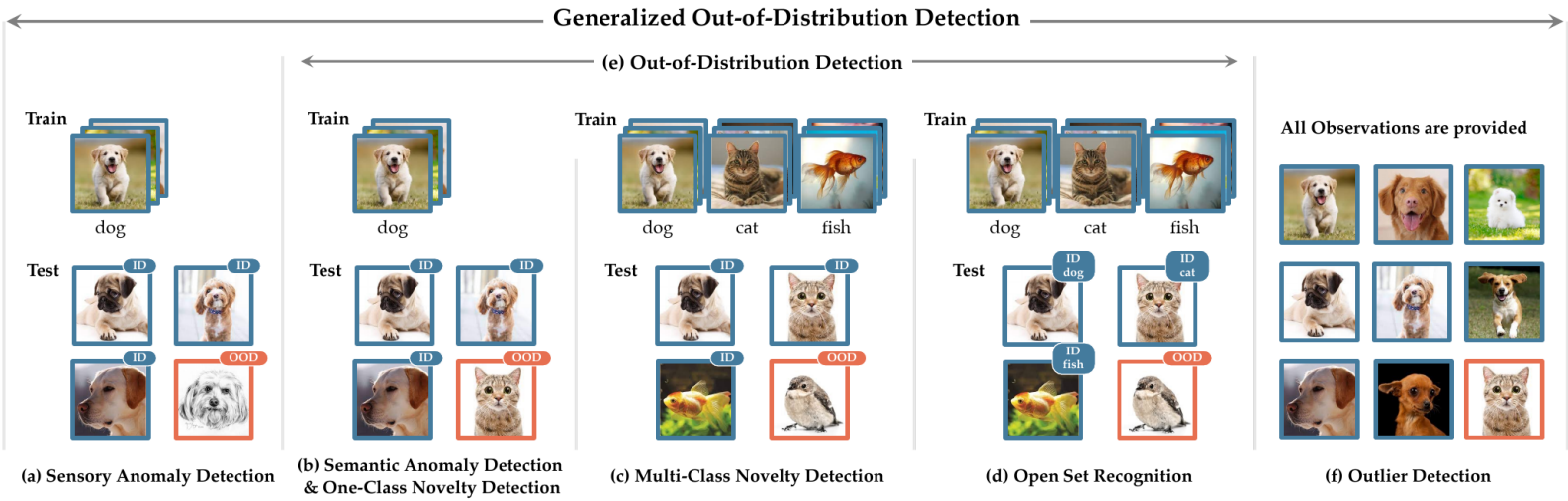
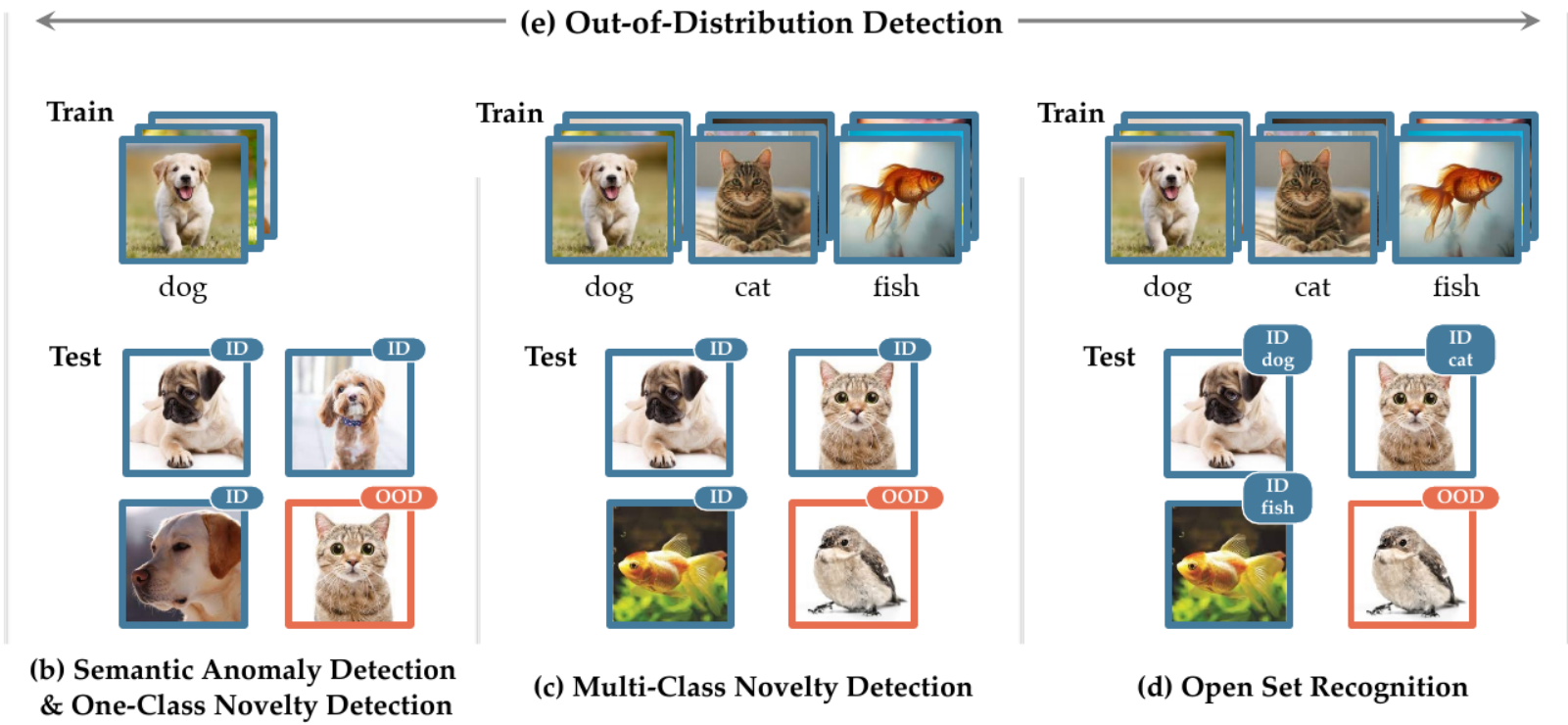


Fig. 2: Exemplar problem settings for tasks under generalized OOD detection framework. Tags on test images refer to model's expected predictions. **(a)** In sensory anomaly detection, test images with covariate shift will be considered as OOD. No semantic shift occurs in this setting. **(b)** In semantic anomaly detection and one-class novelty detection, normality/ID images belong to one class. Test images with semantic shift will be considered as OOD. No covariate shift occurs in this setting. **(c)** In multi-class novelty detection, ID images belong to multiple classes. Test images with semantic shift will be considered as OOD. No covariate shift occurs in this setting. **(d)** Open set recognition is identical to multi-class novelty detection in the task of detection, with the only difference that open set recognition further requires in-distribution classification. **(e)** Out-of-distribution detection is a super-category that covers semantic AD, one-class ND, multi-class ND, and open-set recognition, which canonically aims to detect test samples with semantic shift without losing the ID classification accuracy. **(f)** Outlier detection does not follow a train-test scheme. All observations are provided. It fits in the generalized OOD detection framework by defining the majority distribution as ID. Outliers can have any distribution shift from the majority samples.

Concept of OOD [Yang et al., 2021]



Types of Anomalies

Anomaly Detection

To detect anomalous samples during testing, by defining what is normal, and everything that is not normal, is defined as anomalous.

During training, only normal samples are available, and no anomalous samples are provided. During testing, normal and anomalous samples, and anomalous ones have to be detected. Generally normal is defined homogeneously.

Novelty Detection

Detect test input samples that belong to new classes, basically detect the presence of new/novel classes. Usually ND is multi-class, while AD is generally a single class.

Types of Anomalies

Open Set Recognition

In Open Set Recognition, the concept of *known known classes* and *unknown unknown classes* is introduced.

The idea of OSR is to accurately classify known known classes, while at the same time, detect samples that belong to unknown unknown classes.

Outlier Detection

The aim to detect outliers, that is, samples that differ significantly, but this is done over a whole set of data points, not over a train/test split, using all observed data points.
An outlier is defined by comparison to all observed data points, looking for significant differences.

Types of Anomalies

Out of Distribution Detection

This is the process of rejecting input samples with labels that do not overlap with the training set. This is usually called label shift.

But more generally, it is the process of rejecting any input that is dissimilar with the training set. The training set of your model defines what is out of distribution.

OOD detection requires accurate epistemic uncertainty.

Good OOD performance should not harm ID performance.

Out of Distribution - ID vs OOD Data

In Distribution Data

These are the samples and semantic meaning implied by the training set. For example, if our training set has dogs and cats, in distribution would be the specific semantic meaning of the dogs and cats in the training set, which could be generated from particular species of dogs and cats, with specific background, animal pose, and illumination conditions.

Out of Distribution Data

Anything that is not inside the In Distribution set.

Example given our cats/dogs dataset, OOD would be other non-trained semantic classes (foxes, ducks, humans, houses, etc), and

Out of Distribution - Covariate/Feature Shift

Covariate is another word for feature in statistics.

Here the input distribution changes, for example, new semantic classes might be present, but the model is not aware of these classes. Or currently trained classes are the same, but are visually very different from the samples in the training set for the same class.

Basically, the feature distribution changes, but the label distribution is the same (the model has not been retrained).

Out of Distribution - Covariate/Feature Shift

Figure from the Book Dive into Deep Learning, available here.

Out of Distribution - Label Shift

The feature distribution is the same, but the label distribution changes.

This can be a change in the domain that generates data and labels.

For example, in a medical setting, we know that disease generates symptoms, and the latter is the one that we measure to infer disease.
But these causal relationships are not the same around the world, or not constant due to advances in medical science.

Out of Distribution - Concept Shift

Another possible way for data/labels to shift is concept, that is, the meaning of labels might change over time or over geographical regions.

A very good example is the name for concepts, like soda, soft drink, pop, or coke, mean the same over different regions.

Brand names can also differ around the world, Aldi is the same in The Netherlands, but in Germany there is Aldi Nord and Aldi Süd. In Chile we have a supermarket chain called Jumbo, but is different from the one in The Netherlands.

In Summary - OOD Shifts

There are some well known failure cases/tragedies that showcase the need for OOD Detection.

Urban legend of some Army Research center training to detect different kinds of tanks, to end up overfitting to the color of the sky in the images.

More well known examples of medical image classification, where the model learned to identify X-ray machine features or hospital annotations.

Outline

Concept

We have seen that OOD detection is a concept that covers many previously defined tasks, but now the important question is, how is it evaluated?

What is the proper formulation that is required for evaluation?

Do we need new metrics, or are there some we can reuse?
What datasets can we use for evaluation?

What are some proper evaluation protocols?

Concept - Datasets

For evaluation of OOD, we need *at least* two datasets.

In Distribution Dataset Dataset (including train and test split) where your model is trained and tested.

Out of Distribution Datasets Datasets (usually test splits) where the model is evaluated for OOD detection performance. This dataset must be out of distribution, covering different semantic classes, have some kind of corruption, or lack some intersection with the ID dataset.

But they must be the same format (color images, greyscale images, same input size), overall it should be the same modality.

Concept - Datasets

Some examples of common pairs of ID-OOD datasets used for evaluation.

Fashion MNIST vs MNIST Both are 28×28 grayscale images, while semantic classes are very different (fashion items vs digits).

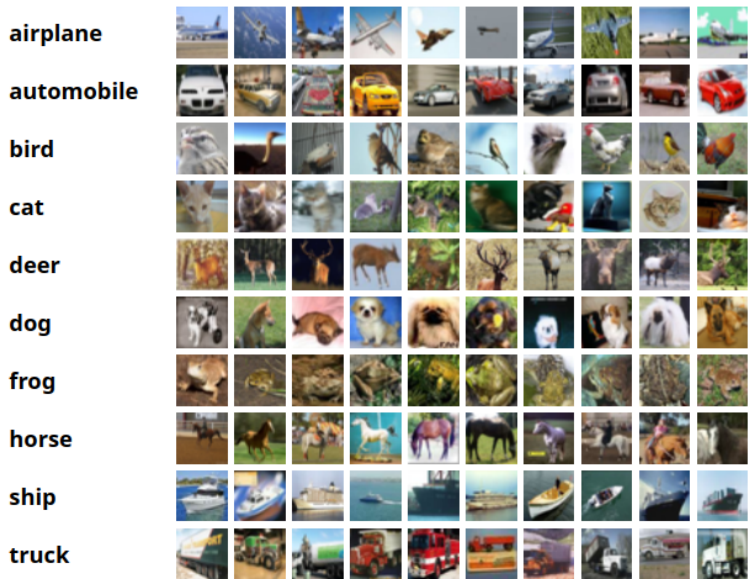
CIFAR10 vs SVHN Both are 32×32 color images (RGB), with different semantic classes (animals/vehicles vs digits in house numbers).

Split Across Classes Another common choice is to use any dataset, but train only in a subset of classes, and use the remaining classes as OOD data.

Fashion MNIST vs MNIST

CIFAR10 vs SVHN

Here are the classes in the dataset, as well as 10 random images from each:



Evaluation Labels

For evaluation, we usually need labels. Since OOD detection is formulated like a binary classification problem, we need binary labels:

Class 0 In-Distribution data.

Class 1 Out of Distribution data.

Note that these labels are *separate* from the ones in the main task (classification or regression).

Evaluation Protocol

1. Decide on your ID and OOD datasets.
2. Train your model on the train split of the ID dataset.
3. Make confidence/probability/uncertainty predictions on the ID and OOD dataset, both test splits. We will call these $\hat{y}_{id}$ and $\hat{y}_{ood}$.
4. Form virtual labels, consisting of 0's for the ID dataset, and 1's for the OOD dataset. This can be done with the following numpy code:
   `np.concatenate([np.zeros_like(`$\hat{y}_{id}$`),`
   `np.ones_like(`$\hat{y}_{ood}$`)], axis=0)`
5. Use any evaluation metrics to compute an evaluation score.

Confidences for OOD Detection

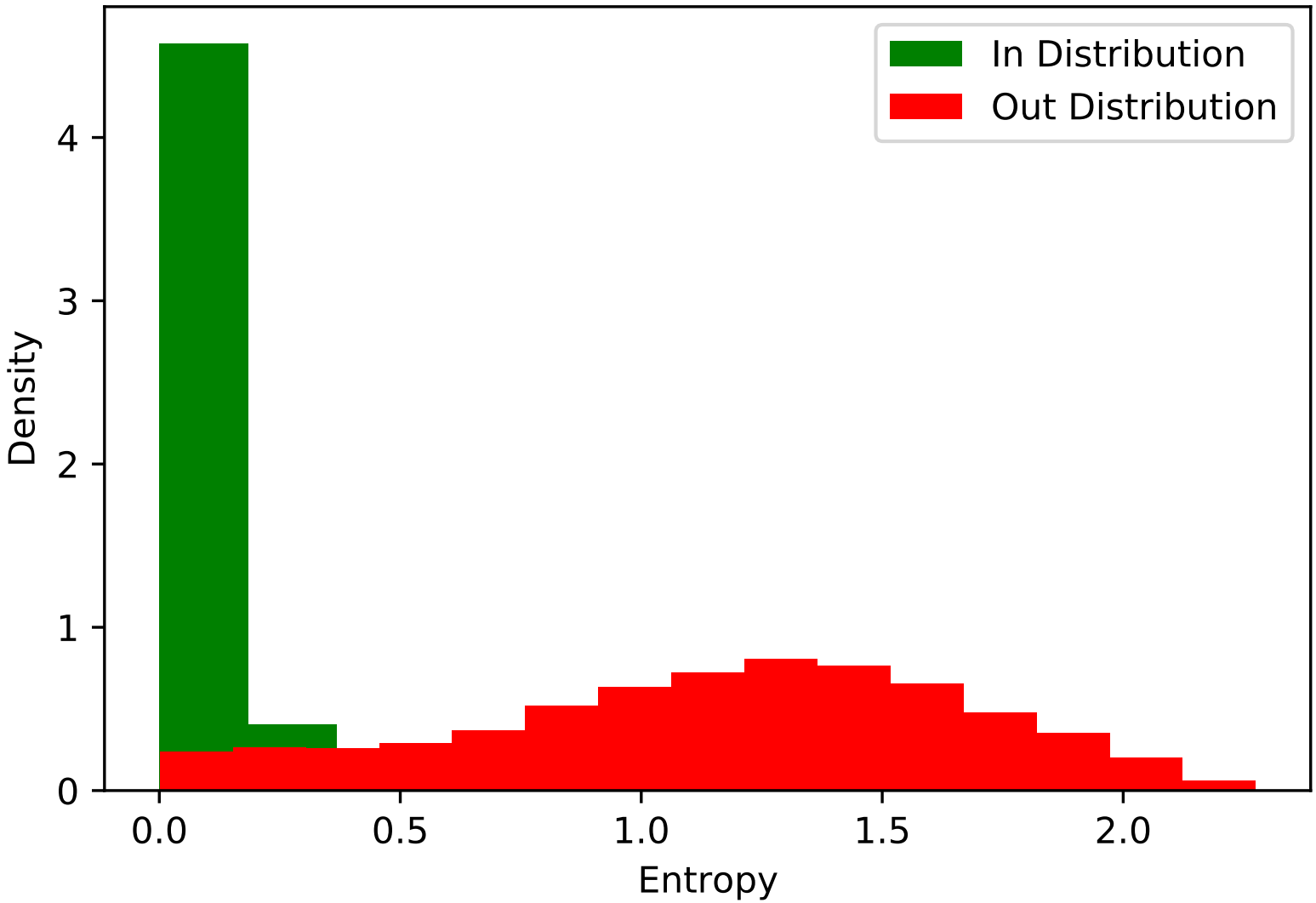
We have covered many confidence scores in this course, but for completeness.

Maximum Probability The inverse of the maximum softmax probability is used for OOD detection ($1 - \max$).

Entropy Shannon entropy for classification, or differential entropy for regression.

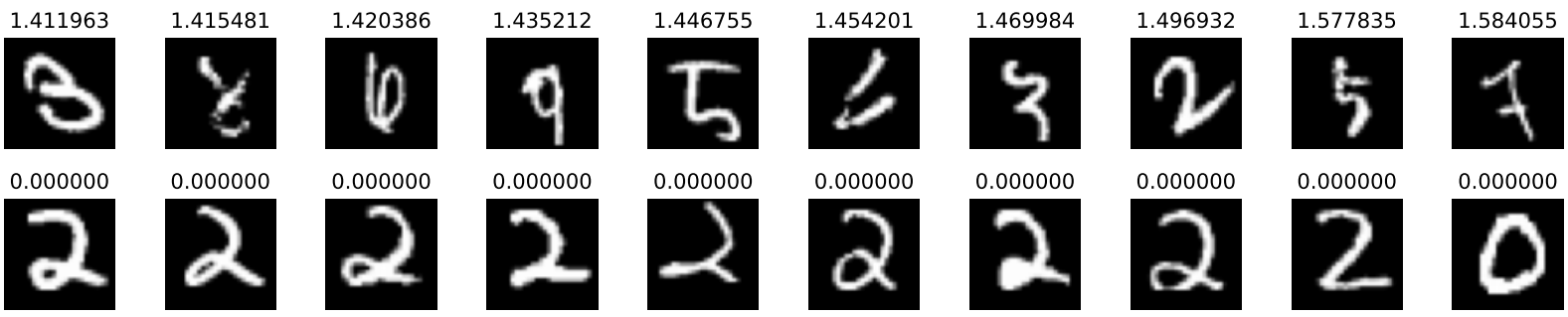
Output Standard Deviation For regression, the standard deviation output is a confidence that can be used for OOD.

OOD Detection - MNIST vs Fashion MNIST

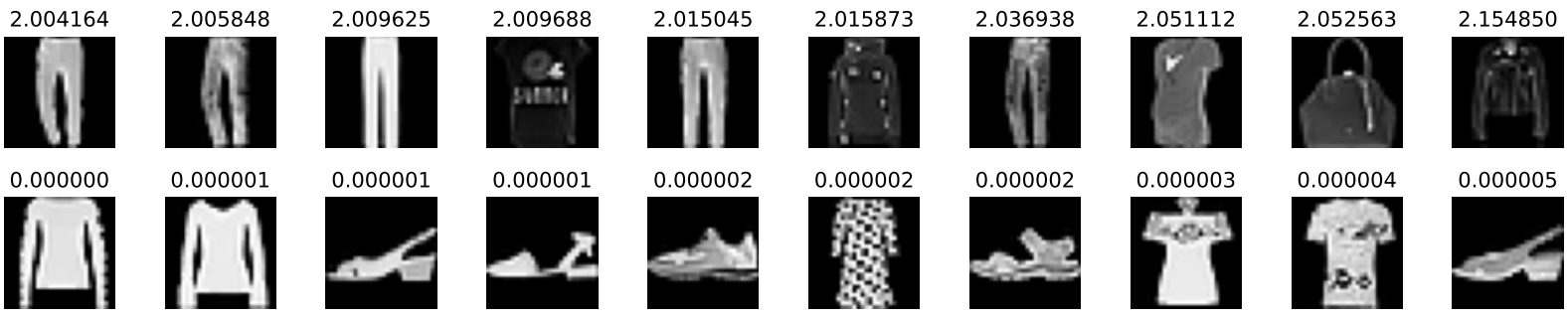


OOD - MNIST vs Fashion MNIST

MNIST - In-Distribution



Fashion MNIST - Out of Distribution



Evaluation Metrics

From previous results, the idea for OOD detection evaluation, is that confidence scores are low for ID samples, and high for OOD samples. Ideally there should be no overlap (but in practice there is).

Which metrics can we use to evaluate such scores?

1. Precision and Recall.
2. False Positive Ratio and True Positive Ratio.
3. ROC Curves and AUC.

Outline

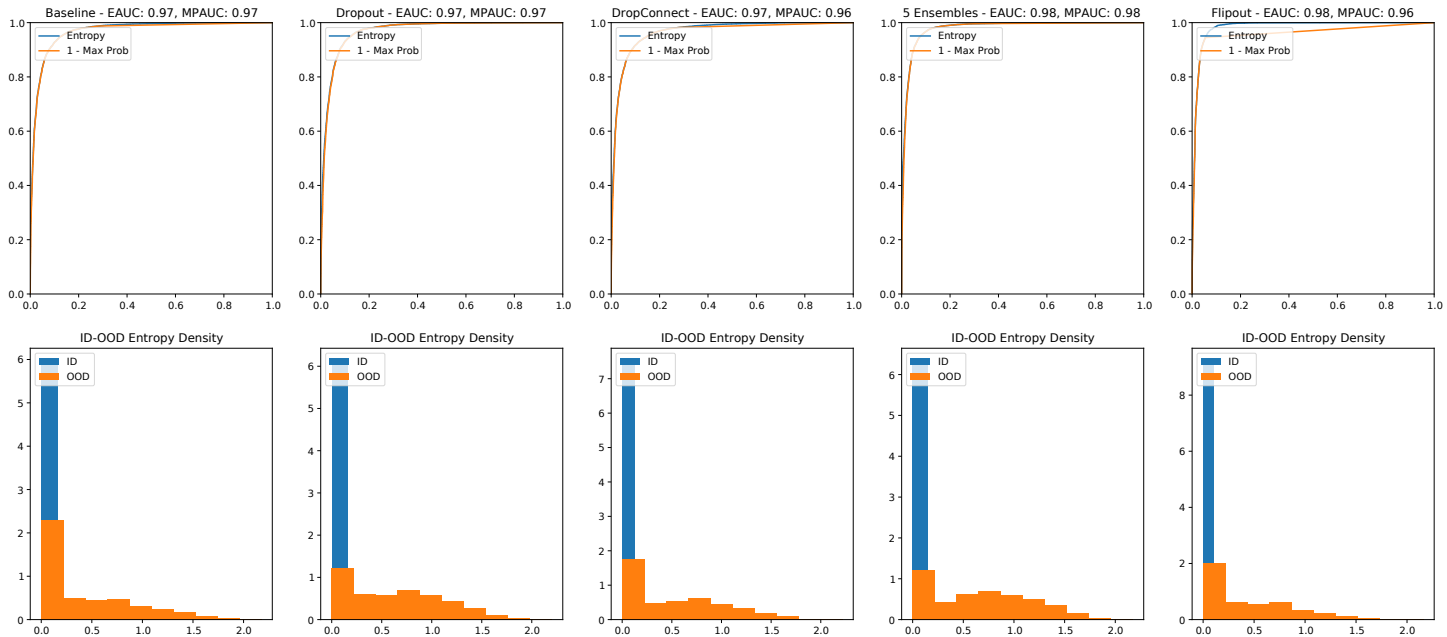
Epistemic Uncertainty

The premier method for out of distribution detection should be epistemic uncertainty.

- Epistemic uncertainty should be low inside samples of the training set.
- Epistemic uncertainty should/will be high outside of the training set, exactly in the out of distribution setting.

Entropy ($-\sum_c \mathbb{P}(y = c \,|\, x) \log$) or the maximum probability ($1 - \max_c \mathbb{P}(y = c \,|\, x)$) can be used for classification, while the standard deviation output is used for regression.

Comparison of Epistemic Uncertainty for OOD Detection



This example is a classifier trained on MNIST (ID), and evaluated on Fashion MNIST (OOD). Entropy is the score used for OOD detection. The bottom plot is the histogram of entropy for ID (blue) and OOD (orange), and on top is the corresponding ROC curve.

Training on OOD Data

One approach for OOD Detection is to use using an additional output head (contrastive loss, maximum discrepancy, etc), and tune the extra output head to produce a high score on OOD data, and a low score on ID data.

Additional Head

Train using binary cross-entropy, so the additional head learns to discriminate ID and OOD data. Does not work very well due to generalization gaps.

Maximum Discrepancy

Learn a model that maximizes discrepancy between features or softmax scores between ID and OOD data.

ODIN - OOD Detection in Image Networks [Liang et al., 2017]

ODIN makes use of temperature scaling, with the observation that a high temperature ($T > 100$) improves OOD detection performance when using the maximum probability. In the paper they use $T = 1000$.

This is because for high temperatures, the softmax output is flatter, closer to the input logit space, where it is easier to distinguish ID from OOD samples.

There are variations of ODIN like Generalized ODIN [Hsu et al., 2020] which trains an additional head to output a per-sample temperature T. In ODIN T is learned from OOD data.

Energy-based OOD Detection [Liu et al., 2020]

This uses a energy score based on the softmax outputs

$$E(x, f) = -T \log \sum_{i}^{C} \exp(f_i(x)/T) \tag{14}$$

This represents the *free energy* of the prediction, based on the softmax denominator. OOD inputs usually get a lower energy score than ID inputs.

Energy-based OOD Detection [Liu et al., 2020]

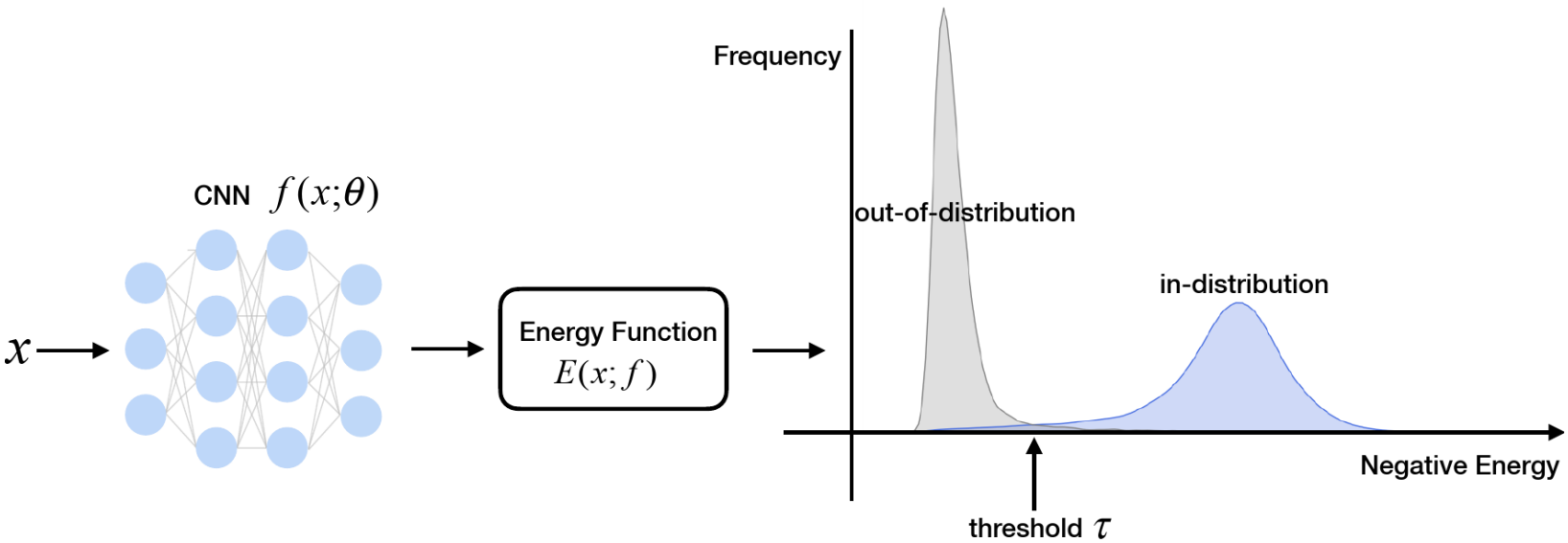


Fig. 29. The energy-based OOD detection framework. The energy function maps the logit outputs to a scalar through a convenient `logsumexp` operator. Test samples with lower energy are considered ID and vice versa.

Generative Models for OOD Detection

A generative model models the density $\mathbb{P}(x)$ instead of discriminative models that model density $\mathbb{P}(y \mid x)$. Examples of generative models are GANs and Variational Autoencoders.

In theory, a well trained generative model would assign lower probabilities (or zero) for OOD data than ID data, but this does always happen in practice.

One way to improve this for OOD detection is to use a likelihood ratio [Ren et al., 2019], which includes information from another generative model that models background information ($\mathbb{P}_{\theta_o}(x)$) for the same modality:

$$\text{LLR}(x) = \log \frac{\mathbb{P}_\theta(x)}{\mathbb{P}_{\theta_o}(x)} = \log \mathbb{P}_\theta(x) - \log \mathbb{P}_{\theta_o}(x) \qquad (15)$$

Generative Models for OOD Detection [Ren et al., 2019]

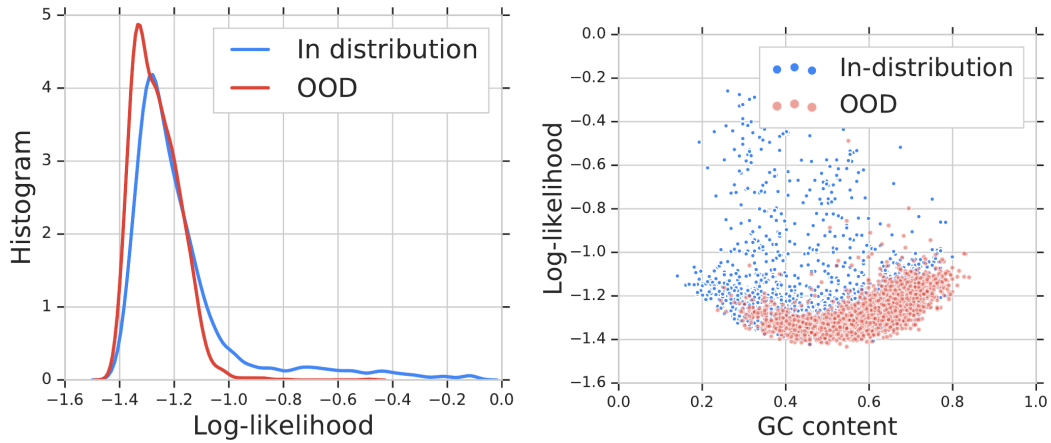


Figure 1: (a) Log-likelihood hardly separates in-distribution and OOD inputs. (b) The log-likelihood is heavily affected by the GC content of a sequence.

Generative Models for OOD Detection [Ren et al., 2019]

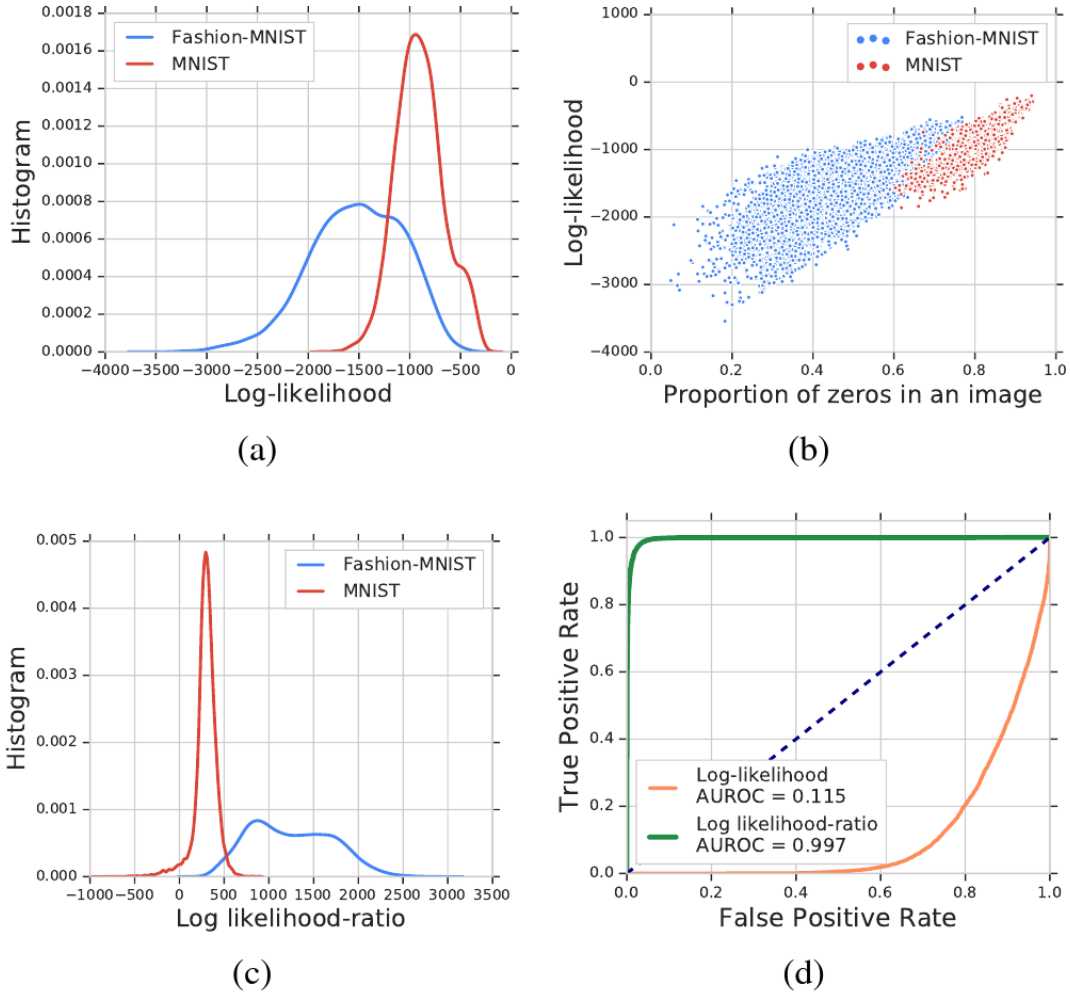


Figure 2: (a) Log-likelihood of MNIST images (OOD) is higher than that of Fashion-MNIST images (in-distribution). (b) Log-likelihood is highly correlated with the background (proportion of zeros in an image). (c) Log-likelihood ratio is higher for Fashion-MNIST (in-dist) than MNIST (OOD). (d) Likelihood ratio significantly improves the AUROC of OOD detection from 0.115 to 0.997.

OOD Detection in Regression

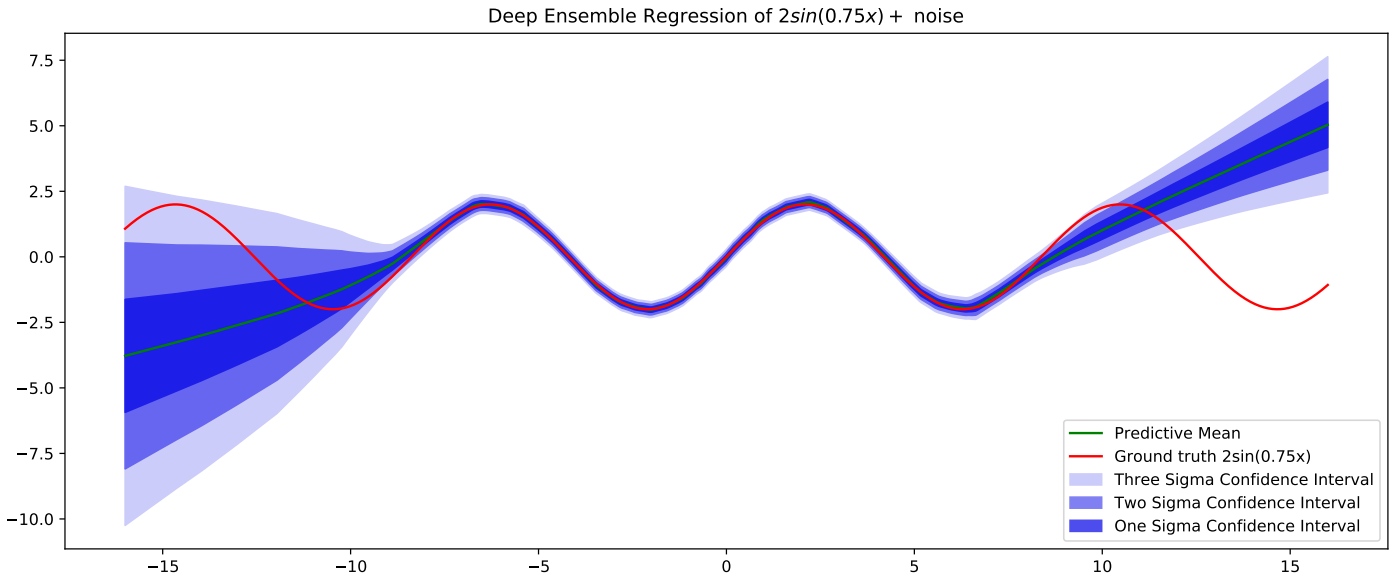
Unfortunately there are not many OOD detection methods particularly designed for regression tasks.

But still epistemic uncertainty works very well, as we have seen in many examples, there is another in the next slides.

For this purpose we use the variance/standard deviation head, the higher its output, the more likely the input is out of distribution.

But one important detail is that variance is unbounded, unlike entropy or probability for classification.

OOD Detection in Regression - Sinusoid with Ensembles



In this example, the training set is $x \in [-8, 8]$, it can be visually seen that outside this range the standard deviation of the output (uncertainty) increases considerably, and increases as with the distance to that range.

Out of Distribution Detection - Pitfalls

- It is not easy to completely separate ID and OOD examples, as some ID examples have still high uncertainty, and sometimes OOD examples have low uncertainty. This is due to variability in classes.
- Choosing a threshold is not easy, as lots of analysis has to be performed.
- Unfortunately there are no guarantees on OOD performance, and there are known cases of bad effects. (See Ovadia et al.)
- Uncertainty should be used as additional information from where further human analysis can be decided, instead of enabling fully automatic processing.

Questions?

Questions to Think About OOD Detection

- Let's say we evaluate ID-OOD detection using datasets A and B, and we get a certain AUROC. Would performance be the same if we swap the datasets? (Now B is ID and A is OOD).
- Why do methods for OOD detection work poorly when using OOD training data?
- Can label and covariate shift happen at the same time?

Bibliography I

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020.

Bibliography II

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. *Advances in Neural Information Processing Systems*, 32, 2019.

Bibliography III

Hao Song, Miquel Perello-Nieto, Raul Santos-Rodriguez, Meelis Kull, Peter Flach, et al. Classifier calibration: How to assess and improve predicted class probabilities: a survey. *arXiv preprint arXiv:2112.10327*, 2021.

Sunil Thulasidasan, Gopinath Chennupati, Jeff A Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.