Evidential Deep Learning to Quantify Classification Uncertainty.

Machine Learning in Practice Reading Group

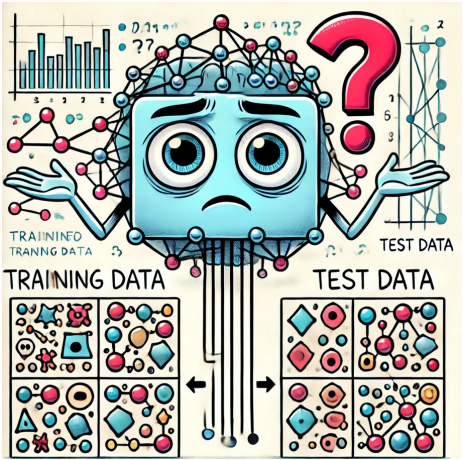
Duke B&B

February 23, 2024

Presented by Aditya Parekh

ChatGPT Art

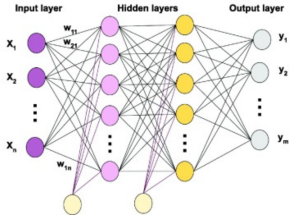
"Draw me a picture of a neural network being very confused because its test data does not match its training data"



Introduction: Classification using Deep Learning

- Deep learning performs really well for various classification tasks (image classification, text classification, etc.)
- A standard approach: for K-class classification, treat output $y = f(X) \in \mathbb{R}^k$ of neural network as probabilities after softmax transformation.

a)



b)

$$\hat{p}_k = \frac{e^{y_k}}{\sum_{j=1}^{K} e^{y_j}}$$

Figure: (Left) Neural Network classifier for K-class classification problem. (Right) Outputs of a neural network are transformed into probabilities using the softmax transformation.

Introduction: Classification using Deep Learning

Pros

- Works extremely well: high predictive accuracy.

Cons

- Poor calibration: models are overconfident on incorrect predictions.

Bayesian Neural Networks (BNNs) can address this issue: estimate uncertainty by examining posterior predictive distributions.

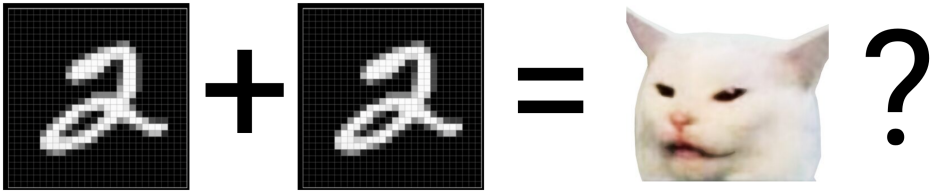
- Problem: posterior distribution is often intractable; must be estimated using MCMC methods which can be noisy.

Gaussian Processes are also useful as uncertainty-aware predictors [1].

Introduction: Classification using Deep Learning

Other Questions: can a model say "I don't know"?

Example: train a model on MNIST, but at test time feed it a picture of a cat.

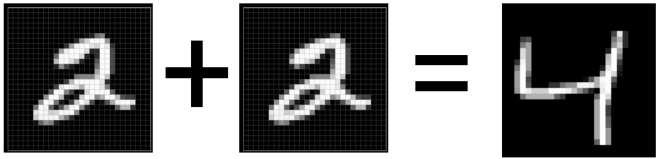


Introduction: Classification using Deep Learning

Other Questions: can a model say "I don't know"?

Example: train a model on MNIST, but at test time feed it a picture of a cat.

Homework:



Test:

Introduction: Classification using Deep Learning

In summary, we have two desires:

1. An accurate quantification of uncertainty for a given prediction
2. An overall quantification of model uncertainty when no confident prediction can be made at all

Introduction: Classification using Deep Learning

Current (2018) SOTA for uncertainty-aware prediction models: Bayesian Neural Networks

- In BNNs, parameters of neural network are treated as random variables
- Hence, the output of a BNN is a posterior distribution over the possible classes
- Non-linear activation functions often make this posterior distribution intractable
- Thus, inference can be quite expensive (or noisy)
- What if we interpret the output of a deterministic(?!) neural network as a distribution instead?

Introduction: Classification using Deep Learning

- What if we interpret the output of a deterministic(?!) neural network as a distribution instead?

Introduction: Classification using Deep Learning

Authors' idea:

- Keep the classical neural network architecture, but interpret the output of the neural network as parameters of a Dirichlet distribution.
- Thus, output is a distribution over the set of possible class probabilities, rather than point estimate of the class probabilities themselves.

Introduction: Classification using Deep Learning

1. An accurate quantification of uncertainty for a given prediction
2. An overall quantification of model uncertainty when no confident prediction can be made at all

Introduction: Classification using Deep Learning

Additionally, we want the neural network to be able to say "I don't know" for inputs that cannot be classified correctly.

This is addressed using the Theory of Subjective Logic (See Background).

Introduction: Classification using Deep Learning

1. An accurate quantification of uncertainty for a given prediction
2. An overall quantification of model uncertainty when no confident prediction can be made at all

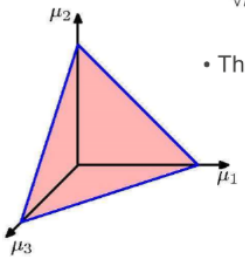
Background: Dirichlet Distribution

Slide courtesy of COMPSCI 675D:

Dirichlet Distribution

• Consider a distribution over the K-dimensional simplex, subject to constraints:

$$\forall k : \mu_k \geqslant 0 \quad \text{and} \quad \sum_{k=1}^{K} \mu_k = 1$$



• The Dirichlet distribution is defined as:

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1}$$

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k$$

where $\alpha_{11},\ldots,\ \alpha_k$ are the parameters of the distribution, and Γ(x) is the gamma function.

• The Dirichlet distribution is confined to a simplex as a consequence of the constraints.

Background: Dirichlet Distribution

In the notation of the paper:

$$D(\mathbf{p}|\alpha) = \begin{cases} \frac{\beta(p_1+\cdots+p_K)}{\beta(p_1)\cdots\beta(p_K)} \prod_{k=1}^{K} p_k^{\alpha_k - 1}, & \text{if } \mathbf{p} \in \mathcal{S}_K \\ 0 & \text{otherwise.} \end{cases}$$

where

$$\mathcal{S}_K = \left\{ \mathbf{p} | \sum_{k=1}^{K} p_k = 1, 0 \leq p_1, \ldots, p_K \leq 1 \right\}.$$

is the K-dimensional simplex.

Background: Dirichlet Distribution

Some reminders:

If $\mathbf{p} = (p_1, \ldots, p_K)$ has distribution $D(\mathbf{p}|\alpha)$:

- Each p_k has a beta distribution with parameters α_k, S
- $\mathbb{E}[p_k] = \frac{\alpha_k}{S}$
- $\text{Var}[p_k] = \frac{\alpha_k(S-\alpha_k)}{S^2(S+1)}$

where $S = \sum_j \alpha_j$.

Background: Theory of Subjective Logic

Subjective Logic motivation [2]:

> *A fundamental limitation of probabilistic logic (and of binary logic likewise) is the inability to take into account the analyst's levels of confidence in the probability arguments, and the inability to handle the situation when the analyst fails to produce probabilities for some of the input arguments.*

Background: Theory of Subjective Logic

The most seemingly-obvious way to handle situations of "I don't know" would be to assign uniform probability mass. However [2]:

> *An analyst might for example want to give the input argument "I don't know", which expresses total ignorance and uncertainty about some statement. However, an argument like that can not be expressed if the formalism only allows input arguments in the form of Booleans or probabilities. The probability $p(x) = 0.5$ would not be a satisfactory argument because it would mean that x and x are exactly equally likely, which in fact is quite informative, and very different from ignorance.*

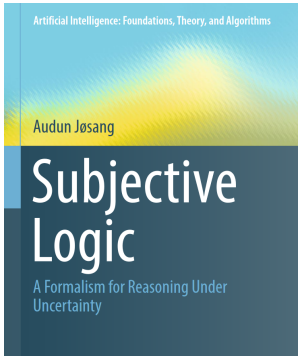
Background: Theory of Subjective Logic

The most seemingly-obvious way to handle situations of "I don't know" would be to assign uniform probability mass. However [2]:

> *An analyst might for example want to give the input argument "I don't know", which expresses total ignorance and uncertainty about some statement. However, an argument like that can not be expressed if the formalism only allows input arguments in the form of Booleans or probabilities. The probability $p(x) = 0.5$ would not be a satisfactory argument because it would mean that x and x are exactly equally likely, which in fact is quite informative, and very different from ignorance.*

Background: Theory of Subjective Logic

If you want all the gory details, see [2]:



However, for our purposes all we really need to understand Subjective Logic is to understand the Dirichlet Distribution.

Background: Theory of Subjective Logic

Consider K mutually-exclusive singletons $1, ..., K$. We represent beliefs for the k-th singleton as a "belief mass" $b_k \geq 0$.

Additionally, we provide an overall uncertainty mass value $\mu \geq 0$.

These $K + 1$ values satisfy

$$\mu + \sum_{k=1}^{K} b_k = 1$$

Background: Theory of Subjective Logic

If $e_k \geq 0$ is the "evidence" for singleton k, then

$$b_k = \frac{e_k}{S} \quad \text{and} \quad \mu = \frac{K}{S},$$

where $S = \sum_{k=1}^{K}(e_k + 1)$.

In the context of neural networks, e_k would be the "evidence" collected from data towards class assignment k.

A *belief mass assignment* corresponds to a Dirichlet distribution with parameters $\alpha_k = e_k + 1$.

Background: Theory of Subjective Logic

Central Idea of the paper: learn a mapping $f : X \to \{\alpha | \alpha_1, \ldots, \alpha_k \geq 0\}$ in the form of neural networks.

In other words, train a model to form "opinions" on class of a given sample in the form of belief mass assignments.

The belief mass assignment provides a probability distribution over the set $\{p_1, \ldots, p_K\}$ of classification probabilities. The expected probability for class k is then

$$\hat{p}_k = \frac{\alpha_k}{S}.$$

Methods

Consider K-class classification problem. Our training data is in the form $(\mathbf{x}_i, \mathbf{y}_i)$, where $\mathbf{y}_i$ is a one-hot encoding vector.

Let $f : X \to \mathbb{R}^K$ represent a neural network. In the standard classification scheme, the final layer of f is a softmax activation layer. Here, we replace the softmax transformation with a ReLU transformation to ascertain non-negative outputs. Let θ be the parameters of f.

The output $f(\mathbf{x}_i|\theta)$ is taken to be the evidence vector predicted by f. That is,

$$\alpha_i = (\alpha_{i1}, \ldots, \alpha_{iK}) = f(\mathbf{x}_i|\theta).$$

The class probabilities $\mathbf{p}_i = (p_{i1}, \ldots, p_{iK})$ are then modeled to follow the distribution $D(\mathbf{p}_i|\alpha_i)$.

Methods

We need a training loss for f. The authors consider 3 options:

- Type II Maximum Likelihood:

$$\mathcal{L}_i(\Theta) = -\log\left(\int \prod_{j=1}^{K} p_{ij}^{y_{ij}} \frac{1}{B(\alpha_i)} \prod_{j=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathbf{p}_i\right)$$

- Bayes risk w/ cross-entropy

$$\mathcal{L}_i(\Theta) = \int \left[-\sum_{j=1}^{K} y_{ij} \log p_{ij}\right] \frac{1}{B(\alpha_i)} \prod_{j=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathbf{p}_i$$

- Mean-Squared Error (next slide)

Long story short: chose mean-squared error based on empirical results (not shown).

Methods

Mean-squared error loss:

$$\begin{aligned}\mathcal{L}_i(\Theta) &= \int ||\mathbf{y}_i - \mathbf{p}_i||^2 \frac{1}{B(\alpha_i)} \prod_{j=1}^{K} p_{ij}^{\alpha_{ij}-1} d\mathbf{p}_i \\ &= \sum_{j=1}^{K} \mathbb{E}\left[y_{ij}^2 - 2y_{ij}p_{ij} + p_{ij}^2\right] \\ &= \sum_{j=1}^{K} \left(y_{ij}^2 - 2y_{ij}\mathbb{E}[p_{ij}] + \mathbb{E}[p_{ij}^2]\right)\end{aligned}$$

Methods

Using the fact $\mathbb{E}[p_{ij}^2] = \mathbb{E}[p_{ij}]^2 + \text{Var}(p_{ij})$, we obtain

$$\begin{aligned}
\mathcal{L}_i(\Theta) &= \sum_{j=1}^{K} (y_{ij} - \mathbb{E}[p_{ij}])^2 + \text{Var}(p_{ij}) \\
&= \sum_{j=1}^{K} (y_{ij} - \alpha_{ij}/S_i)^2 + \frac{\alpha_{ij}(S_i - \alpha_{ij})}{S_i^2(S_i + 1)} \\
&= \sum_{j=1}^{K} (y_{ij} - \hat{p}_{ij})^2 + \frac{\hat{p}_{ij}(1 - \hat{p}_{ij})}{(S_i + 1)} \\
&= \sum_{j=1}^{K} \mathcal{L}_{ij}^{err} + L_{ij}^{var}
\end{aligned}$$

Methods

The authors prove 3 claims with respect to the mean-squared error loss:

Proposition (Proposition 1)

For any $\alpha_{ij} \geq 1$, the inequality $L_{ij}^{var} \leq L_{ij}^{err}$ is satisfied.

Proposition (Proposition 2)

For a given sample i with the correct label j, L_i^{err} decreases when new evidence is added to α_{ij} and increases when evidence is removed from α_{ij}.

Proposition (Proposition 3)

For a given sample i with the correct label j, L_i^{err} decreases when some evidence is removed from the biggest Dirichlet parameter α_{il} such that $l \neq j$.

Together, these ensure that the training loss leads to good data fit, without arbitrarily increasing α_{il} for all labels l.

Methods

1. An accurate quantification of uncertainty for a given prediction
2. An overall quantification of model uncertainty when no confident prediction can be made at all

Methods

To accomplish aim (2), introduce regularization term to bias distribution towards uniform:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{N} \mathcal{L}_i(\Theta) + \lambda_t \sum_{i=1}^{N} KL(D(p_i|\tilde{\alpha}_i)||D(p_i|(1,\ldots,1))),$$

where $\lambda_t = \min(1, t/10)$ is an annealing coefficient for epoch t, and

$$\tilde{\alpha}_i = \mathbf{y}_i + (1 - \mathbf{y}_i)\alpha_i,$$

is the currently estimated evidence vector with non-misleading information removed.

Methods

1. An accurate quantification of uncertainty for a given prediction
2. An overall quantification of model uncertainty when no confident prediction can be made at all

Section 6: Experimental Results

Two Datasets:

- MNIST
- CIFAR10

Two settings:

- Original dataset images
- Dataset images subject to adverserial perturbation

Comparison:

- Prediction Accuracy
- Prediction uncertainty (quantified as % of maximum possible entropy in terms of class probability distribution output)

Compared against other top-performing image classification models, including uncertainty-aware model types such as BNNs.

Section 6: Comparison with softmax

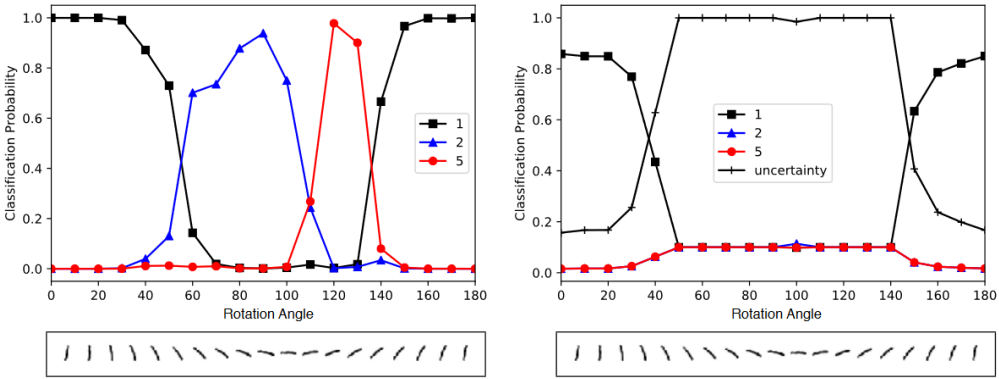


Figure 1: Classification of the rotated digit 1 (at bottom) at different angles between 0 and 180 degrees. **Left:** The classification probability is calculated using the *softmax* function. **Right:** The classification probability and uncertainty are calculated using the proposed method.

Section 6: Effect of including uncertainty threshold for predictions

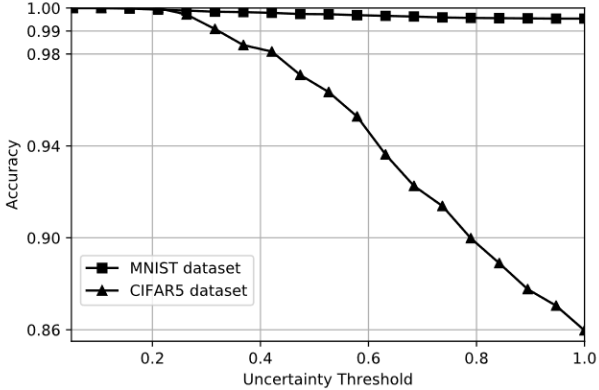


Figure 2: The change of accuracy with respect to the uncertainty threshold for *EDL*.

Section 6: Predictive accuracy results with existing baselines

Method	MNIST	CIFAR 5
L2	99.4	76
Dropout	99.5	84
Deep Ensemble	99.3	79
FFGU	99.1	78
FFLU	99.1	77
MNFG	99.3	84
EDL	99.3	83

Table 1: Test accuracies (%) for MNIST and CIFAR5 datasets.

Section 6: "This wasn't on the homework" experiment

For next experiment: trained on MNIST, tested on notMNIST.

So, we would expect to see high predictive distribution entropies (all classifications are incorrect, and the model should be fairly uncertain about all of its predictions).

Section 6: "This wasn't on the homework" experiment

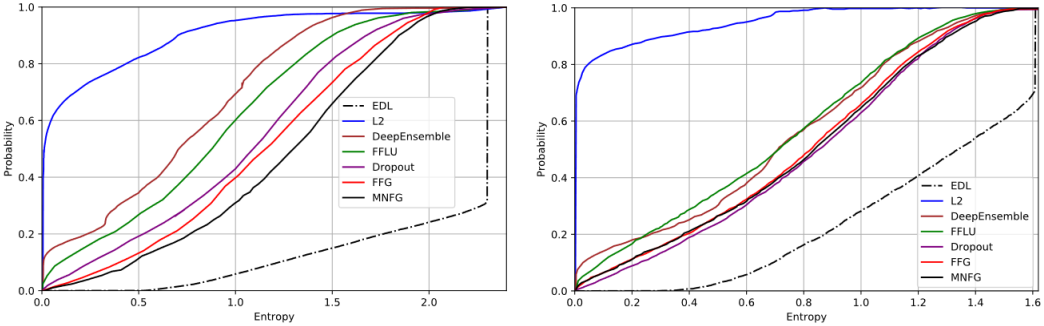


Figure 3: Empirical CDF for the entropy of the predictive distributions on the notMNIST dataset (left) and samples from the last five categories of CIFAR10 dataset (right).

Section 6: Adversarial Perturbation

Note that as ϵ increases, the perturbations become harder to detect.

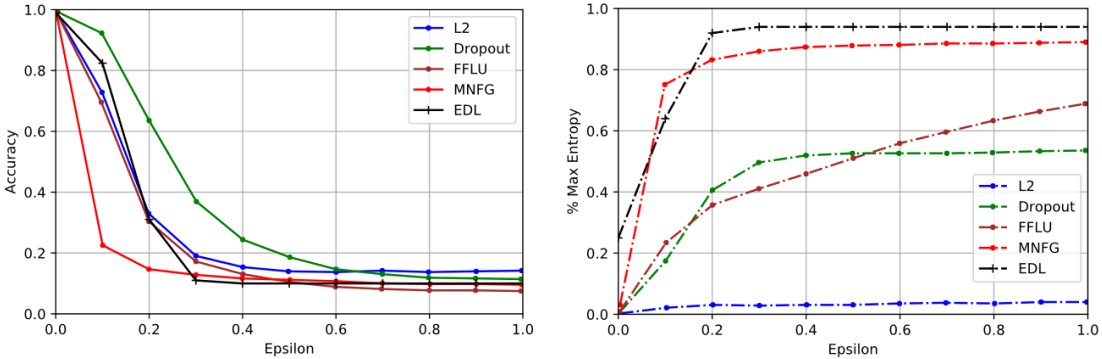


Figure 4: Accuracy and entropy as a function of the adversarial perturbation ϵ on the MNIST dataset.

Section 6: Adversarial Perturbation

Note that as ϵ increases, the perturbations become harder to detect.

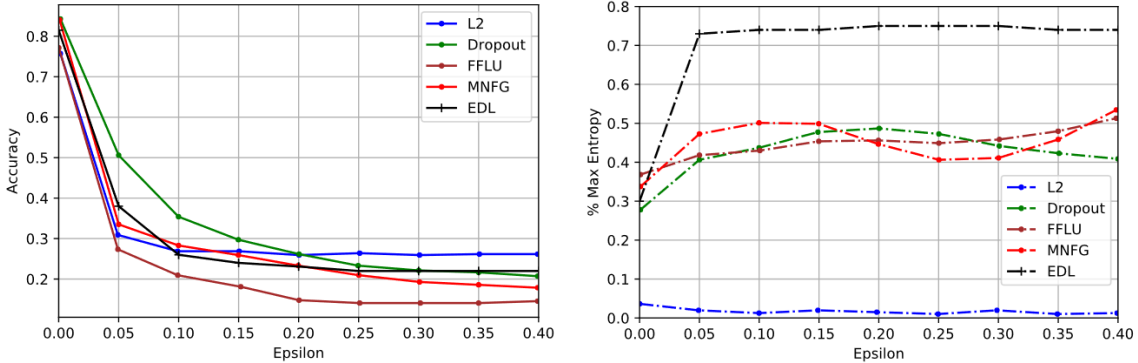


Figure 5: Accuracy and entropy as a function of the adversarial perturbation ϵ on CIFAR5 dataset.

Section 7: Recommendations

Conclusions

- Clever idea, that is very tractable since it fits within the standard neural network architecture.
- Quantifies overall uncertainty of model as well as uncertainty in specific predictions (so a model could flag an input as "I don't know, haven't seen this before" rather than giving a garbage prediction).
- Note that this model does not do everything a Bayesian model can do (e.g., model selection based on model evidence).

Recommendations

- Worth reading? Yes, paper is easy to read, not overly technical, and presents the authors' thinking quite nicely.
- Worth implementing? Yes. Could be implemented using standard pytorch code.

Has it been used?

- Cited 986 times (source: Google Scholar)
- Example application: classifying molecular structures [3], image-based brain tumor segmentation [4], pedestrian detection [5], etc.

References

[1] Murat Sensoy, Lance Kaplan, and Melih Kandemir. "Evidential deep learning to quantify classification uncertainty". In: *Advances in neural information processing systems* 31 (2018).

[2] Audun Jsang. *Subjective Logic: A formalism for reasoning under uncertainty*. Springer Publishing Company, Incorporated, 2018.

[3] Esther Heid et al. "Chemprop: a machine learning package for chemical property prediction". In: *Journal of Chemical Information and Modeling* 64.1 (2023), pp. 9–17.

[4] Ke Zou et al. "TBraTS: Trusted brain tumor segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2022, pp. 503–513.

[5] Qing Li et al. "Stabilizing Multispectral Pedestrian Detection With Evidential Hybrid Fusion". In: *IEEE Transactions on Circuits and Systems for Video Technology* 34.4 (2024), pp. 3017–3029. DOI: 10.1109/TCSVT.2023.3306870.