When in Doubt: Improving Classification Performance with Alternating Normalization

Menglin Jia1,2 Austin Reiter2 Ser-Nam Lim2 Yoav Artzi1 Claire Cardie1
1Cornell University 2Facebook AI
{mj493,yya5,claire}@cornell.edu,{areiter,sernamlim}@fb.com

Abstract

We introduce Classification with Alternating Normalization (CAN), a non-parametric post-processing step for classification. CAN improves classification accuracy for challenging examples by re-adjusting their predicted class probability distribution using the predicted class distributions of high-confidence validation examples. CAN is easily applicable to any probabilistic classifier, with minimal computation overhead. We analyze the properties of CAN using simulated experiments, and empirically demonstrate its effectiveness across a diverse set of classification tasks.

1 Introduction

Classification is core to NLP, and many language problems can be effectively addressed as supervised classification tasks. However, even the most effective classifier can suffer when given examples to classify that are close to its decision boundary. The reasons for such failures vary, and include lack of training data coverage, limited representation expressivity, or over-fitting the training data. Despite significant progress, including using pre-trained models (Devlin et al., 2019) to address these issues, every classifier has its weak spots, and some examples will be hard to classify correctly.

In this paper, we study a simple non-parameterized post-processing step to improve classifier accuracy on difficult examples. At the core of our approach is using Alternating Normalization (AN; Sinkhorn and Knopp, 1967) to re-adjust the prediction of low-confidence examples using the predicted class distributions of a reference set of high-confidence validation examples.

Our process, Classification with Alternating Normalization (CAN), is applicable to any classifier that generates a distribution over target classes. We first identify challenging examples and a disambiguating reference set using the ambiguity level of the predicted class probability distributions. Then we perform a series of normalizations, alternating between normalizing across examples for each class and for each example across classes. For example, in Figure 1, we classify an input example (“smile”) to one of two labels (and ). The example sits at the decision boundary between the two target labels, and is completely ambiguous. The reference set includes a single example (“hat”), which the classifier can resolve with high confidence. Here, we use a single alternating normalization step, including normalizing across rows (examples) and columns (target labels), which disambiguates this simple example to classify it correctly.

We study CAN on randomly generated matrices (Section 3) and evaluate it on several text classification tasks (Section 4). In general, we find that CAN is most effective when the original predictions are

1We use a classical pragmatic reasoning example for our illustration, highlighting our inspiration in the Rational Speech Act (RSA; Frank and Goodman, 2012) model, which we discuss in Section 6.
of higher ambiguity. Our experiments also suggest CAN is more beneficial for tasks with many labels.

2 Classification with Alternating Normalization

We modify the output of a probabilistic classifier for an ambiguous input example at test time using alternating normalization with respect to an additional set of examples to re-adjust the input example’s predicted probabilities. Let $\mathcal{X}$ and $\mathcal{Y}$ be random variables for input and labels, respectively. Given a challenging example $x \in \mathcal{X}$, and a probability distribution $\mathbb{P}(y|x)$ over a set of $m$ classes produced by a probabilistic classifier, we compute $\mathbb{P}'(y|x)$ as an adjusted distribution in three steps, the second of which is an iterative process.

Our method requires a reference set of examples. We use the confident portion of the validation data commonly used in standard evaluation settings. Given $n$ confident examples, we create a row stochastic matrix $A_0 \in \mathbb{R}^{n \times m}$ by concatenating the predicted probability distributions. We describe the proposed method below.

Step 1: Identify Hard Examples Let $b_0 \in \mathbb{R}^m$ be the source prediction, i.e., the predicted class probability distribution for a challenging example $x$. We identify examples that can benefit from CAN by computing the ambiguity level of its class probability distribution $b_0$. With a selected example, we construct a matrix $L_0 \in \mathbb{R}^{(n+1) \times m}$ by concatenating $A_0$ and $b_0$ along the rows:

$$ L_0 = \begin{bmatrix} A_0 \\ b_0^T \end{bmatrix}. $$

A common way to identify challenging examples is by measuring the entropy of their predicted class distribution $H(b_0) = -\sum_i b_{0,i} \log_2(b_{0,i})$. The higher the value of $H(b_0)$ is, the more uniform the distribution $v$ is, which indicates higher ambiguity level. We observe $H(b_0)$ may not be ideal to capture ambiguity well for our classification purpose, which is concerned mainly with high probability events. For example, consider two distributions $b_0^1 = [0.5 \ 0.25 \ 0.25]$ and $b_0^2 = [0.5 \ 0.5 \ 0]$, for which $H(b_0^1) > H(b_0^2)$. However, $b_0^2$ expresses a more uncertain classification result.

Instead, we select challenging examples based on the entropy near the peak of the distribution. We define top-$k$-entropy to focus on the top of the distribution. Let $T : \mathbb{R}^m \rightarrow \mathbb{R}^k$ be the top-$k$ operator. The top-$k$-entropy is:

$$ H_{\text{top-k}}(b_0) = H(T(b_0, k)). $$

We use a base of $m$ so that $0 < H_{\text{top-k}}(b_0) \leq 1$.

Given a scalar threshold $0 \leq \tau \leq 1$ and the number of classes $k_{\text{max}}$, the ambiguity level of a probability distribution $b_0$ is larger than $\tau$, if for any $k \in [2, k_{\text{max}}]$, the top-$k$-entropy of $b_0$ is greater than $\tau$.

Step 2: One Iteration of AN Each iteration $d$ of AN normalizes $L_0$, first across its rows (row norm) and then along its columns (column norm). Let $\mathcal{D} : \mathbb{R}^n \rightarrow \mathbb{R}^{n \times n}$ turn a vector $v$ into a diagonal matrix, and let $e$ be a vector of ones.

Step 2.1: Row Norm The row normalization of $L_{d-1}$ is:

$$ \Lambda_S = \mathcal{D}((L_{d-1}^{-1})^T e) $$

$$ S_d = L_{d-1}^{-1} \Lambda_S^{-1} $$

where $\alpha > 0$, $L_{d-1}^\alpha$ is the matrix exponentiation of $L_{d-1}$, and $\Lambda_S^{-1}$ is the inverse of $\Lambda_S$. The diagonal entries of $\Lambda_S \in \mathbb{R}^{m \times m}$ represent the column sums of $L_{d-1}^\alpha$, so that $S_d$ is column stochastic. The parameter $\alpha$ controls the rate of convergence of $b_0$ to a high confidence state.

Step 2.2: Column Norm The column normalization step is:

$$ \Lambda_L = \mathcal{D}(S_d \Lambda_q e) $$

$$ L_d = \Lambda_L^{-1} S_d \Lambda_q $$

where $\Lambda_q \in \mathbb{R}^{m \times m}$ is a diagonal matrix that represents the class priors, which we approximate using the training class distribution. The diagonal entries of $\Lambda_L \in \mathbb{R}^{(n+1) \times (n+1)}$ are the row sums of $S_d \Lambda_q$ so that $L_d$ is row stochastic.

Step 3: Re-adjusted Output Extraction Let $L_d$ be the resulting matrix after $d$ steps of Step 2:

$$ L_d = \begin{bmatrix} A_d \\ b_d^T \end{bmatrix}. $$

We keep $b_d$ as the re-adjusted class probability distribution $\mathbb{P}'(y|x)$, and discard $A_d$.

3 Simulations on Random Matrices

We study the effect of the ambiguity level of the source prediction and the reference set using Monte

---

1. $b_0^T$ denotes the transpose of $b_0$.

2. Appendix B.1 provide a step-by-step visualization of the alternating normalization process, and Appendix B.2 discusses the its convergence.

3. Each normalization step (Steps 2.1–2.2) takes $O(mn)$ because the matrices $\Lambda_S, \Lambda_L, \Lambda_q$ are diagonal.
We define two metrics (\( \text{Baseline} \)) matrices for simplicity. In practice, we explore how \( \text{arg max} \) measures the accuracy gain (i.e., of the performance gain of a set of input examples) \( \text{change}: (a) \delta(b_0, B_1) \) to evaluate the expected classifier performance \( \text{b}\)-compute each \( A \) \( \text{Carlo simulations. We randomly generate expected accuracy gain.} \)

**Setup** The ambiguity levels are grouped into 4 intervals: \( \{[0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1]\} \). Given the number of classes \( m \) and an ambiguity interval, we randomly generate \( A_0 \) in \( \mathbb{R}^{(m-1)\times m} \times \lambda_q \), and \( B_0 \in \mathbb{R}^{n\times m} \) independently, where each row of \( B_0 \) represents a randomly generated \( b_0 \). For each interval and each \( m \in \{2, \ldots, 10, 20, \ldots, 100\} \), we randomly generate \( A_0, B_0 \), and \( \Lambda_q \) 200 times with \( n = 100 \). We compute each \( b_1 \) separately using \( \text{CAN} \) with \( \alpha \in \{0.1, \ldots, 0.9, 1, \ldots, 9\} \), and construct \( B_1 \).

**Evaluation Metrics** We define two metrics to evaluate the expected classifier performance change: (a) \( \delta(b_0, b_1) \) measures the expected performance gain of \( b_1 \) w.r.t. \( b_0 \); and (b) \( \Delta(B_0, B_1) \) measures the accuracy gain (i.e., of the \text{arg max}) of a set of input examples \( B_1 \) w.r.t. \( B_0 \). We explore how \( \delta(b_0, b_1) \) and \( \Delta(B_0, B_1) \) change as a function of the ambiguity level of \( B_0 \) and \( A_0 \).

**Effect of Ambiguity Level** Figure 2 shows the averaged \( \delta(b_0, b_1) \) and \( \Delta(B_0, B_1) \) across all matrix sizes, simulations, and values of \( \alpha \). We observe that (a) the expected accuracy gains are positive across all ambiguity levels of \( A_0 \) and \( b_0 \); (b) \( \text{CAN} \) tends to improve the performance of \( b_0 \) with high ambiguity level, especially using a reference set with low top-k entropy; and (c) the performance is robust to the ambiguity level of \( A_0 \).

### 4 Empirical Experiments

We evaluate \( \text{CAN} \) using three classification tasks: ultrafine entity typing (\( \text{Choi et al., 2018} \)), and dialogue-based relation extraction (\( \text{DialogRE; Yu et al., 2020} \)). We compare off-the-shelf classifiers (\( \text{Baseline} \)) and our method.\(^8\) We select \( \text{CAN} \)'s hyperparameters \( (\alpha, d, \tau) \) using the official validation sets \( \text{val} \) and evaluate on the official test sets \( \text{test} \). The challenging subsets of the \( \text{val} \) and \( \text{test} \) are identified as the source predictions using top-k entropy. The rest of \( \text{val} \), which has low ambiguity, is used as the reference set \( A_0 \).\(^9\)

**Results** Table 1 summarizes the experimental results. \( \text{CAN} \) offers consistent performance gains for different classifiers and datasets by re-adjusting the uncertain examples only, especially when the task has many classes. Table 1 suggests more effective classifiers benefit less from \( \text{CAN} \), but still see improvements. For example, in the ultrafine entity typing experiment, we observe a larger improvement for the multitask version (\( \text{Choi et al., 2018} \)) compared with the denoised one (\( \text{Onoe and Durrett, 2019} \)), which uses a cleaned up version of the training data.

**Tuning \( \text{CAN} \)** Figure 3 shows the how the values of the hyperparameters \( (\alpha, d, \tau) \) affect \( \text{CAN} \) on the

| Datasets          | # Classes | Method                               | Marco F1 | Mirco F1 |
|-------------------|-----------|--------------------------------------|----------|----------|
|                   |           |                                      | val      | test     |
|                   |           |                                      | val      | test     |
| Ultrafine         | 10331     | \text{Baseline} (Multitask; Choi et al., 2018) | 31.32    | 31.98    | 27.92    | 28.80 |
|                   |           | \text{CAN}                            | (12.15)  | (12.71)  | (12.51)  | (12.89) |
|                   |           | \text{Baseline} (Denoised; Onoe and Durrett, 2019) | 40.07    | 40.22    | 37.88    | 37.87 |
|                   |           | \text{CAN}                            | (10.34)  | (10.53)  | (10.75)  | (10.59) |
| DialogRE          | 36        | \text{Baseline} (BERT; Yu et al., 2020) | 35.89    | 35.76    | 59.44    | 57.93 |
|                   |           | \text{CAN}                            | (10.91)  | (10.70)  | (10.45)  | (10.16) |
|                   |           | \text{Baseline} (BERTs; Yu et al., 2020) | 40.58    | 39.45    | 62.18    | 59.52 |
|                   |           | \text{CAN}                            | (10.83)  | (10.68)  | (10.43)  | (10.29) |

\(^8\)Similar to \( \text{Yuan et al., 2018} \), we set \( L / S \) as square matrices for simplicity. In practice, \( L / S \) do not need to be square, as shown in Section 4.

\(^7\)Appendix A.2 provides formal definitions.
Figure 3: Effect of $\alpha$, recursion depth $d$, and $\tau$ on the macro F1 scores of ultrafine entity typing val set with the Baseline multitask classifier.

val set using ultrafine entity typing multitask Baseline. We observe that the effect of $d$ and $\tau$ diminish gradually as $\alpha$ grows, because $\alpha$ controls how quickly CAN transforms the source predictions to high-confidence ones. For a fixed recursion depth $d$, the performance does not always improve using a larger $\alpha$. This suggests that larger $\alpha$ can deteriorate the performance through over-calibration. We also see that a small value of recursion depth $d$ yields the best results in general.

5 Analysis

We hypothesize that classifiers with better performance require less re-adjustment. We test this hypothesis by controlling the number of “hard” examples, therefore controlling the performance of the Baseline classifier. We use ImageNet (Deng et al., 2009) because of the existence of established ways for image perturbation without modifying the image semantics. This dataset contains 1.3 million training and 50k validation natural scene images with 1000 classes. We use a ResNet-50 (He et al., 2016) model from the default pretrained models in torchvision (Paszke et al., 2017) as Baseline.

We systematically make the task harder by convolving the images with a Gaussian function with zero-mean and various values of standard deviation $\sigma$. Higher values of $\sigma$ result in more blurred image, which emulate examples with higher uncertainty and lower Baseline classifier performance. Figure 4 shows example images with different values of $\sigma \in \{2, 4, 8, 16, 32\}$.

We assess the effect of CAN by comparing two variants of CAN with Baseline using this setup: (a) CAN-NAIVE: $\alpha = 1.0$, recursion depth is 1; and (b) CAN-BEST: an upperbound-version of CAN with $\alpha$ and depth $d$ optimized on the test data.

Figure 5 presents the absolute top-1 accuracy (%) gain on the hard subset of val. Optimizing $\alpha$ and $d$ (CAN-BEST) indeed offers larger performance gains across six different image settings comparing with CAN-NAIVE. We observe a general positive correlation between the number of hard examples and the relative gain using CAN, confirming that our method benefits classifiers that receive harder data. However, this advantage diminishes when a classifier significantly under-performs, as the accuracy gain dropped for Blur ($\sigma = 8$) onward and its top-1 accuracy is only 3.25% on hard subset.

6 Related Work

Rational Speech Act (RSA) RSA is a framework for pragmatic reasoning, where a speaker and a listener generate and understand utterances by reasoning about the understanding and intentions of their interaction partner (Frank and Goodman, 2012; Goodman and Frank, 2016). Both agents are probabilistic, and RSA uses alternating normalization in a recursive process. Our technique is motivated by the type of reasoning provided by RSA, whereas interpretation and generation of messages are considered within the context of other interpretations and messages to resolve ambiguities.

We adapt the RSA technique to post-process the output of probabilistic classifiers. The classifier takes a similar role to the RSA listener in the alternating normalization process. The matrices $S_d$ and $L_d$ from Equations 4 and 6 are aligned with the
stochastic matrices for the derived speaker and listener in RSA. The row norm (Equation 4) mirrors the special case of a speaker that considers the cost of generating a message as zero. This assumption is reasonable for classification. It is also common practice when working with a finite set of intents in RSA (Monroe et al., 2017; Zarrieß and Schlangen, 2019). The column norm (Equation 6) describes a mathematical formulation aligned with how a pragmatic listener infers speaker expectations.

Previous work has applied RSA to systems that generate and understand language (Andreas and Klein, 2016; Mao et al., 2016; Vedantam et al., 2017; Cohn-Gordon et al., 2018; Zarrieß and Schlangen, 2019) in both referential games (Frank and Goodman, 2012; Goodman and Frank, 2016; Monroe et al., 2017) and sequential decision-making systems (Fried et al., 2018a,b). Our method departs from these applications by focusing on the ambiguity avoidance property of the listener agent as applied to generic classification tasks.

Confidence Calibration Similar to confidence calibration techniques (Platt et al., 1999; Zadrozny and Elkan, 2002; Guo et al., 2017; Kumar et al., 2019), our method rescales the posterior distribution produced by the classifier at test time. However, the aim of calibration is to make the output probabilities more representative of the correctness likelihood, whereas our’s is to resolve ambiguity.

7 Conclusions and Future Work

We propose Classification with Alternating Normalization (CAN) as a simple and light-weight post-processing step. Our method adjusts the predicted class distribution of “edge cases” for a generic classifier during test time. Via experiments on both simulated and real-world NLP tasks, we show that CAN helps improve performance using a fixed reference set with low ambiguity and increases the performance of standard classifiers. Future work may further investigate the properties of CAN. For example, Appendix 5 describes an initial study on vision examples showing the benefit of CAN increases as examples become noisier, even with the same classifier. Advancing this study and generalizing it to language is an important direction for future work. Finally, one could study improving CAN, for example by selecting $A_0$ strategically, or applying it during training.

Acknowledgements

We would like to thank the members of the Cornell NLP group, and anonymous reviewers for their helpful feedback. Photo of Figure 4 by Joe Caione on Unsplash.

References

Jacob Andreas and Dan Klein. 2016. Reasoning about pragmatics with neural listeners and speakers. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182, Austin, Texas. Association for Computational Linguistics.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Reuben Cohn-Gordon, Noah Goodman, and Christopher Potts. 2018. Pragmatically informative image captioning with character-level inference. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 439–443, New Orleans, Louisiana. Association for Computational Linguistics.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Michael C Frank and Noah D Goodman. 2012. Predicting pragmatic reasoning in language games. Science, 336(6084):998–998.

Daniel Fried, Jacob Andreas, and Dan Klein. 2018a. Unified pragmatic models for generating and following instructions. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 1951–1963, New Orleans, Louisiana. Association for Computational Linguistics.
A Implementation Details

A.1 Extension to Multilabel Problems

The problem we describe in Section 2 is supervised multi-class classification. For multi-label classification task, given \( n \) input examples and \( m \) classes, a classifier usually produces an array of size \( n \times m \) where the value in each position represents the predicted probability of one input-class pair. We transform this array into shape \( nm \times 2 \), where each row is the binary probability distribution for each input-class pair. This is similar to the common binary cross-entropy loss for this task.

A.2 Simulation Metrics Definitions

Under the framework described in Section 2, we define two metrics to evaluate the expected classifier performance. Expected performance gain measures the performance of \( b_1 \) w.r.t. \( b_0 \). Let \( \mathcal{Y}, \hat{\mathcal{Y}} \) be two random variables. \( \mathcal{Y} \) represents the true label of a given instance, and \( \hat{\mathcal{Y}} \) is the predicted class label out of a set of classes \( C \). The expected accuracy of \( b_0 \) would be:

\[
E_0[Acc] = \frac{1}{|C|} \sum_{c \in C} P(\mathcal{Y} = c | \hat{\mathcal{Y}} = c)
\]

(8)

where each entry at position \( c \) of \( q \in \mathbb{R}^{|C|} \) is the probability mass function (PMF) of \( \mathcal{Y} \) when \( \hat{\mathcal{Y}} = c \). \( q \) is the main diagonals of the randomly generated \( \Lambda_q \). Each entry at position \( c \) of \( b_0 \) is the PMF of \( \mathcal{Y} \) when \( \hat{\mathcal{Y}} = c \). The relative expected performance gain is then defined as:

**Definition 1 (Expected Performance Gain).** The relative expected performance gain of \( b_1 \) w.r.t. \( b_0 \) is:

\[
\delta(b_0, b_1) = \frac{E_1[Acc] - E_0[Acc]}{E_0[Acc]}
\]

(9)

\( \delta \) quantifies the performance on individual example. Next we introduce a second metric, accuracy gain, to measure a set of input examples.

**Definition 2 (Accuracy Gain).** Given \( B_1 \in \mathbb{R}^{n \times m} \), where the \( i \)\(^{th} \) row represent a re-adjusted predicted distribution, denoted as \( b_1^i \), the overall performance of \( B_1 \) w.r.t. \( B_0 \) is evaluated as:

\[
\Delta(B_0, B_1) = \frac{1}{n} \sum_{i} \text{max} \left\{ \delta_i > 0, \arg \max_{b_0} \neq \arg \max_{b_1} \right\}
\]

(10)

where \( \delta_i = \delta(b_0^i, b_1^i) \). We consider CAN is successful when the predicted classes change from \( b_1 \) to \( b_0 \) and \( \delta(b_0, b_1) > 0 \). \( \Delta \) is used for the primary metric in our simulation study.

B Technical Analysis

B.1 Geometric Interpretation

To intuitively understand the alternating normalization, we visualize the effect of the Row Norm and Column Norm in Section 2 to the source prediction in Figure 6. It portrays a step-by-step transformations of the source prediction \( b_0 \rightarrow b_1 \), where \( b_0, b_1 \in \mathbb{R}^2 \) using the example in Figure 1.

Consider two diagonal matrices \( \Lambda_1, \Lambda_2 \), where \( \Lambda_2 = \Lambda_L^{-1} \in \mathbb{R}^{(n+1) \times (n+1)}, \Lambda_1 = \Lambda_S^{-1} \Lambda_q \in \mathbb{R}^{m \times m} \). \( \Lambda_L, \Lambda_S, \Lambda_q \) are the scaling matrices in Eq. 3-4. The transformation of \( L_{d-1} \rightarrow L_d \) can be written as:

\[
L_d = \Lambda_2 L_{d-1}^\top \Lambda_1
\]

(12)

Since we care about the transformation of \( b_d \), combining Eq. 7 and 12 yields the main operations of our method:

\[
b_d = \lambda_2 \Lambda_1 b_{d-1}^\top
\]

(13)

\( \lambda_2 \) is a scalar at the \((n + 1, n + 1)\) entry of \( \Lambda_2 \).
Visualizing the effect of $\Lambda_1, \lambda_2$ Figure 6(a)-6(c) show that starting from $b_0$, which has unit 1-norm, $\Lambda_1$ rotates and scales $b_0$ (Figure 6(b)), $\lambda_2$ scales the resulting vector so $b_1$ has unit 1-norm (Figure 6(c)). $\Lambda_1$ determines which direction $b_0$ rotates to. Intuitively speaking, it is the context-aware nature of the RSA that governs the transformation of $b_0$.

Effect of $\alpha$ Figure 6(d) shows the trajectory of $\Lambda_1 v^\alpha$ for $v$ that has unit 1-norm in a toy example where $L \in \mathbb{R}^{2 \times 2}, A_0 = \begin{bmatrix} 0.1 & 0.9 \end{bmatrix}$. The larger the value of $\alpha$, the closer $\Lambda_1 v^\alpha$ is to the base vector in the row space of $L$. This observation reconfirms the effect of $\alpha$ in Section 3: it controls how much to take into account of the input vector $v$ and how quickly for $v$ to achieve a high confident state.

B.2 Convergence of $L_d$

Let $\alpha = 1$ and $\Lambda_q$ be an identity matrix. Given a square $L_d$, Sinkhorn and Knopp (1967) have proved that in this setting, $\lim_{d \rightarrow \infty} L_d$ will converge if $L_0$ is full rank. Consider a $2 \times 2$ case, where $A_0 = \begin{bmatrix} p_A & 1 - p_A \\ 1 - p & p \end{bmatrix}$, $b_0^T = \begin{bmatrix} p \\ 1 - p \end{bmatrix}$. According to the Sinkhorn-Knopp algorithm, if $p_A(1 - p) - p(1 - p_A) \neq 0$,

$$p_A \begin{bmatrix} 1 - p_A \\ p \end{bmatrix} \xrightarrow{d \rightarrow \infty} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} or \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$ (14)

Therefore, $\lim_{d \rightarrow \infty} b_d$ will always converge to a unit vector which has the lowest entropy possible if $p_A \neq p$.

This illustrate that $b_0$ and $A_0$ are key to the convergence and the operations in Eq. 13. This observation lays a foundation for the simulation study in Section 3, where we study the effect of the ambiguity levels of $A_0$ and $b_0$ in generalized settings.

C Reproducibility Details

The experiments in this work do not require training or GPU. We either download the publicly available pre-trained model checkpoints, or obtain the model output on val and test sets from the researchers who propose the Baselines.

C.1 Datasets and Baselines

The statistics of the evaluated tasks and the associated datasets are shown in Table 2. All evaluation and optimization protocols are based on the practices of the corresponding Baselines for each task.

Ultrafine Entity Typing This task is to predict a set of semantic types of a given entity mention within a sentence. The dataset (Choi et al., 2018) contains 10,331 entity types, including coarse, fine and ultra-fine grained classes. Since each entity can have more than one types, this task is a multi-label classification problem. Following previous practices (Ling and Weld, 2012; Choi et al., 2018), we adopt the loose Macro and loose Micro F1 score as metrics. We optimize the loose Macro F1 score on the val set to select hyperparameters. Two existing classifiers are evaluated: (1) Multitask (Choi et al., 2018): a model proposed along with the dataset and utilizes a LSTM-based AttentiveNER model (Shimaoka et al., 2016); (2) Denoised (Onoe and Durrett, 2019): uses denoised distant training data.

DialogRE The task of Dialogue-based relation extraction (DialogRE) (Yu et al., 2020) dataset is to predict one or more types of relations between two entities mentioned in dialogues. We applied CAN to the two baselines proposed in the paper: BERT, and BERTs. Each experiment has five runs following the practices of Yu et al. (2020), so we report the average Macro F1 and Micro F1. The Micro F1 is used for the parameter optimization following Yu et al. (2020).

C.2 Hyperparameters

A hold-out validation set is required by our method. In practice, this can be the same set for fine tuning other hyperparameters during training neural classifiers. Our assumption is that the different splits of a dataset (train, val, test) are sampled from the same distribution. The training class distribution is utilized as prior $\Lambda_q$. We use the original val set of the datasets to optimize for $\alpha \in \{0.1, \ldots, 0.9, 1, \ldots, 35\}$, iteration number $d \in \{1, \ldots, 5\}$, and the ambiguous threshold $\tau \in \{0.25, 0.5, 0.75\}$. Table 3 summarizes all the hyperparameters used in this paper. Table 4 presents resulting number of ambiguous distributions to re-adjust. We find that the optimized values of $d$ are mostly $\leq 2$ (77.3 % of all experiments), which is in line with the discussions previously in the main text.
### Table 2: Evaluation tasks and off-the-shelf BASELINE methods used. Multiple BASELINE architectures types are evaluated in our experiments.

| Task                        | Datasets                  | BASELINE                        | Cls. Type | # Classes | train/val/test |
|-----------------------------|---------------------------|---------------------------------|-----------|-----------|----------------|
| Entity Typing               | Ultrafine Entity Typing   | Multitask (Choi et al., 2018)  | Multi-label | 10331     | 1998 / 1998 / 1998 |
|                             |                           | Denoised (Onoe and Durrett, 2019) |           |           |                 |
| Relation Extraction         | DialogRE                  | BERT (Yu et al., 2020)         | Multi-label | 36        | 5997 / 1914 / 1862 |
|                             |                           | BERTs (Yu et al., 2020)        |           |           |                 |
| Image Classification        | ImageNet                  | ResNet-50 (He et al., 2016)    | Single-label | 1000   | 1,281,167 / 50,000 / - |
|                             |                           |                                 |           |           |                 |

### Table 3: Hyper-parameters used for all experiment. We report details of all the output from five runs of DialogRE dataset.

| Task                        | Datasets                  | BASELINE | α     | d     | τ     |
|-----------------------------|---------------------------|----------|-------|-------|-------|
| Entity Typing               | Ultrafine Entity Typing   | Multitask| 16    | 1     | 0.25  |
|                             |                           | Denoised | 22    | 2     | 0.75  |
| Relation Extraction         | DialogRE                  | BERT     | 5, 0.7, 8, 5, 0.6 | 2, 3, 1, 3, 2 | 0.75, 0.25, 0.25, 0.75, 0.5 |
|                             |                           | BERTs    | 0.8, 7, 4, 7, 4  | 5, 1, 1, 1, 1 | 0.25, 0.5, 0.5, 0.75, 0.25 |
| Image Classification        | ImageNet-Original         | 0.6      | 1     |       |       |
|                             |                           | 0.7      | 2     |       |       |
|                             |                           | 0.9      | 5     |       |       |
|                             |                           | 0.9      | 3     |       | 0.75  |
|                             |                           | 0.5      | 1     |       |       |
|                             |                           | 0.6      | 1     |       |       |

### Table 4: Number of ambiguous examples in val and test sets (absolute | relative (%)).

| Task                        | Datasets                  | BASELINE | val     | test    |
|-----------------------------|---------------------------|----------|---------|---------|
| Entity Typing               | Ultrafine Entity Typing   | Multitask| 14269 | 0.07% | 19007 | 0.09% |
|                             |                           | Denoised | 4454 | 0.02% | 4601 | 0.02% |
|                             |                           | BERT     | 1478 | 2.15% | 1430 | 2.13% |
|                             |                           |          | 4598 | 6.67% | 4512 | 6.73% |
|                             |                           |          | 4126 | 5.99% | 3993 | 5.96% |
|                             |                           |          | 1210 | 1.76% | 1148 | 1.71% |
|                             |                           |          | 2610 | 3.79% | 1148 | 1.71% |
|                            |                           | BERT     | 3725 | 5.41% | 3841 | 5.73% |
|                            |                           |          | 1539 | 2.23% | 1513 | 2.26% |
|                            |                           |          | 1579 | 2.29% | 1547 | 2.31% |
|                            |                           |          | 885  | 1.28% | 859  | 1.28% |
|                            |                           |          | 3131 | 4.54% | 3153 | 4.70% |
| Relation Extraction        | DialogRE                  |          | 13484| 26.97% |       |
|                             |                           |          | 24788| 49.58% |       |
|                             |                           |          | 34421| 68.84% |       |
|                             |                           |          | 41230| 82.46% |       |
|                             |                           |          | 45721| 91.44% |       |
|                             |                           |          | 48889| 97.78% |       |
| Image Classification        | ImageNet-Original         |          | 33864| 68.84% |       |
|                             |                           |          | 41230| 82.46% |       |
|                             |                           |          | 45721| 91.44% |       |
|                             |                           |          | 48889| 97.78% |       |
|                             |                           |          | 3131 | 4.54% | 3153 | 4.70% |