Uses and Abuses of the Cross-Entropy Loss: Case Studies in Modern Deep Learning

Elliott Gordon-Rodriguez
Department of Statistics
Columbia University
eg2912@columbia.edu

Gabriel Loaiza-Ganem
Layer6 AI
gabriel@layer6.ai

Geoff Pleiss
Zuckerman Institute
Columbia University
gmp2162@columbia.edu

John P. Cunningham
Department of Statistics
Columbia University
jpc2181@columbia.edu

Abstract

Modern deep learning is primarily an experimental science, in which empirical advances occasionally come at the expense of probabilistic rigor. Here we focus on one such example; namely the use of the categorical cross-entropy loss to model data that is not strictly categorical, but rather takes values on the simplex. This practice is standard in neural network architectures with label smoothing and actor-mimic reinforcement learning, amongst others. Drawing on the recently discovered continuous-categorical distribution, we propose probabilistically-inspired alternatives to these models, providing an approach that is more principled and theoretically appealing. Through careful experimentation, including an ablation study, we identify the potential for outperformance in these models, thereby highlighting the importance of a proper probabilistic treatment, as well as illustrating some of the failure modes thereof.[1]

1 Introduction

The cross-entropy loss is one of the most commonly used loss functions for training deep neural network models, most notably in (multi-class) classification problems. When applied to categorical data, this loss function corresponds to a probabilistic log-likelihood, therefore resulting in favorable estimation properties. On the other hand, several prominent methods in modern machine learning are concerned with fitting data that is not quite categorical, but simplex-valued; key examples being the "soft targets" in label smoothing (LS) [27], and "expert policies" in actor-mimic reinforcement learning (AMN) [23], amongst others [16, 28]. In these methods, the deep learning community has defaulted to borrowing the same cross-entropy loss from the categorical case, despite the fact that it no longer defines a bona fide probability model. As well as highlighting this practice and putting it into question, our work proposes adjusting the LS and AMN objective functions by replacing the cross-entropy loss with the log-likelihood of the recently discovered *continuous-categorical* (CC) distribution [12]. Doing so amounts to incorporating a normalizing constant to our model, or in other words, adding the factor that scales the cross-entropy loss to a valid probability density function over the simplex. As of yet, such an approach has only been considered in the context of knowledge distillation [12], although the one-dimensional special case (corresponding to the binary cross-entropy with $[0, 1]$-valued data) has been studied more extensively [20]. Our inspiration draws from both of these works, although our focus is primarily on LS and AMN architectures instead.

[1] Our code is available at https://github.com/cunningham-lab/cb_and_cc.

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

Our exposition is organized as follows (note that our two main sections, 3 and 4, are based on the same idea, but are broadly independent of one another and can be read separately):

- In section 2 we detail the relevant background on the continuous-categorical distribution, highlighting its close connection to the cross-entropy loss.
- Section 3 focuses on label smoothing. We propose a novel CC-LS model and perform an ablation study to isolate its potential as a regularizer for classification networks, as well as a qualitative assessment of its learned representations.
- Section 4 focuses on actor-mimic reinforcement learning. We recast the AMN model as the solution to a regression problem of simplex-valued data, we propose a novel CC-AMN model, and we provide an experimental evaluation thereof.
- Section 5 concludes, combining insights from CC-LS and CC-AMN, and discussing potential directions for future research.

2 Background

We preface the introduction of the continuous-categorical distribution with a brief notational overview of the categorical cross-entropy loss, which will highlight the close connection between the two and will provide an orthogonal viewpoint to its original presentation in [12].

Categorical data refers to observations y that take values in a discrete sample space Ω formed by K distinct elements, which are typically expressed using the K one-hot vectors that form the standard basis of $\mathbb{R}^K$, namely $\Omega = \{e_1, \dots e_K\}$, where $(e_k)_j = \mathbb{1}(k = j)$. In this notation, the cross-entropy loss is equivalent to the negative log-likelihood of $y \in \Omega$ under a categorical distribution with parameter π:

$$l(\pi; y) = -\sum_{k=1}^{K} y_k \log \pi_k \iff p(y; \pi) = \prod_{k=1}^{K} \pi_k^{y_k}. \tag{1}$$

In other words, the cross-entropy loss defines a coherent probabilistic model for discrete data over K classes. This elementary fact should not be overlooked; it provides the benefits of the theory of maximum likelihood estimation, including frequentist consistency and asymptotic efficiency, as well as enabling efficient Bayesian inference by specifying a conjugate prior.

2.1 From the Cross-Entropy to the Continuous-Categorical

So far so good. However, what happens when the observation is not quite categorical, but instead takes values on the simplex, $\Delta^K = \{y \in \mathbb{R}_+^K : \sum_{k=1}^K y_k = 1\}$? Such data is called *compositional*, and is common in the sciences [1]. In the deep learning literature, while not explicitly referred to as such, compositional data plays a key role in label smoothing [27], actor-mimic reinforcement learning [23], knowledge distillation [16], and domain adaptation [28]. In all of these methods, neural networks are trained to target a simplex-valued outcome, $y \in \Delta^K$, using the cross-entropy loss $l(\lambda; y) = -\sum_{k=1}^K y_k \log \lambda_k$, where λ represents the output of a neural network. Crucially though, the change in sample space from Ω to Δ^K breaks the equivalence in (1) because the right-hand expression no longer defines a proper probability distribution; its integral over Δ^K does not normalize to 1.

Given the attractive properties of maximum likelihood estimation, there are still good reasons why a legitimate probability model is desirable (see [20] for a more detailed discussion). The classical statistics literature offers some possibilities, notably the use of *logratios* [1, 2, 3, 9], or *Dirichlet regression* [6, 15]. However, we argue that the most natural probabilistic solution is to apply the recently discovered continuous-categorical distribution [12], since this corresponds to normalizing the cross-entropy loss directly so that it becomes a genuine log-likelihood model, namely:

$$l(\lambda; y) = -\log C(\lambda) - \sum_{k=1}^{K} y_k \log \lambda_k \iff p(y; \lambda) = C(\lambda) \cdot \prod_{k=1}^{K} \lambda_k^{y_k}, \tag{2}$$

where $C(\lambda)$ is the normalizing constant:

$$C(\lambda) = \left(\int_{\Delta^K} \prod_{k=1}^{K} \lambda_k^{y_k} dy_k \right)^{-1}. \tag{3}$$

This distribution was found to possess a number of attractive theoretical and empirical properties [12]; we highlight the closed form expression of its normalizing constant:

$$C(\lambda) = \left((-1)^{K+1} \sum_{k=1}^{K} \frac{\lambda_k}{\prod_{i \neq k} \log \frac{\lambda_i}{\lambda_k}} \right)^{-1}, \tag{4}$$

which enables the use of automatic differentiation for optimizing models with the continuous-categorical log-likelihood (2). We also highlight that the continuous-categorical outperformed the Dirichlet distribution in regression models of compositional data, including neural network models [12].

3 Continuous-Categorical Label Smoothing

Label smoothing [27] has enjoyed rapid growth and widespread use as a means to reduce overfitting and improve the out-of-sample accuracy of neural network classifiers across a range of tasks including computer vision [31, 24], speech recognition [8], and machine translation [30]. The mechanism is simple: given a neural network classifier $f_\theta : x \to y$, we replace our one-hot labels $y \in \Omega$ with "soft" targets:

$$y^{\text{LS}} = (1 - \varepsilon)y + \varepsilon u, \tag{5}$$

where $u = (1/K, \ldots, 1/K)^\top$ is a uniform vector and $\varepsilon > 0$ is a constant. The network weights θ are then trained to minimize the cross-entropy loss between the network output $f_\theta(x)$ and the modified data y^{LS}.

Equation 5 maps $y \in \Omega$ to $y^{\text{LS}} \in \Delta^K$, so that our targets are no longer categorical, but simplex-valued. Thus, even though they are not continuously distributed, it is natural to consider label-smoothed classification through the lens of compositional regression. Our proposal is therefore to use a continuous-categorical log-likelihood in lieu of the cross-entropy loss, and we refer to this model as CC-LS. Namely, we are interested in comparing the usual label smoothing loss:

$$\min_\theta \mathcal{L}^{\text{LS}}(\theta) = -\sum_{(x,y)} \sum_k y_k^{\text{LS}} \cdot \log[f_\theta(x)]_k, \tag{6}$$

against its continuous-categorical counterpart:

$$\min_\theta \mathcal{L}^{\text{CC-LS}}(\theta) = -\sum_{(x,y)} \left\{ \log C(f_\theta(x)) + \sum_k y_k^{\text{LS}} \cdot \log[f_\theta(x)]_k \right\}. \tag{7}$$

We remark that, strictly speaking, in order to make our targets continuous over the simplex, we would also have to add continuous noise to the labels, for example by drawing u uniformly at random on the simplex. Such an approach produced little difference over using the fixed value $u = (1/K, \ldots, 1/K)^\top$, neither in LS nor CC-LS, and we will omit the results for clarity. However, given the wealth of existing methods that achieve improved generalization error by adding noise at different stages in the training procedure [5, 26, 25], the idea of smoothing the labels with random noise may still hold potential, and we leave its further analysis for future work.

3.1 Experiments

Following the experimental setup of Muller et al [22], we train a CNN classifier on CIFAR-10, with and without label smoothing as well as our novel CC-LS model (see appendix A.1 for the full details of our architecture). This is an example in which label smoothing provided no significant gain over the un-smoothed baseline, likely because the CNN is already regularized using dropout [26], weight decay [18], and batch normalization [17], which altogether are sufficient to provide a good model of

Table 1: Ablation study for label smoothing on CIFAR-10. We show out-of-sample accuracy for our baseline classifier (w/o LS), as well as vanilla LS and CC-LS, both with $\varepsilon = 0.1$. Errors indicate the standard deviation over 10 random initializations of the network. We consider the effect of LS and CC-LS over the baseline under each combination of dropout, weight decay and batch normalization, and find that CC-LS provides significant outperformance in the absence of BatchNorm.

Dropout	Weight decay	BatchNorm	w/o LS	with LS	CC-LS
Yes	Yes	Yes	**89.5** ($\pm$0.1)	89.1 ($\pm$0.2)	89.0 ($\pm$0.2)
No	Yes	Yes	**89.6** ($\pm$0.1)	89.2 ($\pm$0.1)	89.2 ($\pm$0.2)
Yes	No	Yes	**89.4** ($\pm$0.2)	89.3 ($\pm$0.2)	89.0 ($\pm$0.2)
No	No	Yes	**89.5** ($\pm$0.2)	89.4 ($\pm$0.1)	89.1 ($\pm$0.2)
Yes	Yes	No	88.6 ($\pm$1.2)	88.6 ($\pm$1.0)	88.7 ($\pm$0.6)
No	Yes	No	88.8 ($\pm$1.2)	88.7 ($\pm$1.0)	88.6 ($\pm$0.6)
Yes	No	No	87.0 ($\pm$0.2)	87.0 ($\pm$0.1)	**87.6** ($\pm$0.2)
No	No	No	86.8 ($\pm$0.1)	87.0 ($\pm$0.2)	**87.6** ($\pm$0.2)

the data, given the level of complexity of CIFAR-10. Likewise, we find that the CC-LS model also performs no better than the baseline in this setting (top row of Table 1).

Driven by these observations, we perform an ablation study over the different regularizers used in our network, and the results paint a more interesting picture (Table 1). Notably, we find that for the unregularized CNN (bottom row), CC-LS significantly outperforms both LS and the baseline. In the case where our network is partially regularized with dropout only, the baseline becomes equally good as LS, but the gap with CC-LS remains wide (penultimate row), and the gain from CC-LS persists after adding weight decay. On the other hand, batch normalization (top half) was sufficient to capture all the gain in test accuracy, with neither LS nor CC-LS outperforming the baseline in these cases. Under weight decay without batch normalization (rows 5 and 6), training became less stable (as evidenced by the large standard deviations), but CC-LS was able to reduce the variability in model accuracy. Overall, Table 1 indicates that the CC-LS loss function provides a different (and sometimes, significantly better) regularization effect than that of vanilla LS, suggesting its potential for novel applications, particularly in the settings where batch normalization may be undesirable [10]. We note further that numerous existing works have been devoted to analyzing the interplay between dropout, weight decay, and batch normalization [29, 11, 7, 19, 14]; our focus is specifically on their relation to label smoothing and CC-LS.

We end this section with a qualitative analysis of the learned representations from our trained classifiers. Again following [22], we define the "template" vector of the kth class, w_k, as the weight vector from the last CNN layer that is associated to the kth class, so that in other words:

$$[f_\theta(x)]_k = \frac{e^{w_k^\top z}}{\sum_{k'} e^{w_{k'}^\top z}}, \qquad (8)$$

where z is a vector containing the activations from the penultimate layer. We then fix three classes, and construct an orthonormal basis (consisting of two vectors) for the plane containing their three template vectors. For each of the classes, we pick a random sample of input data belonging to that class and project their penultimate layer activations onto this plane. The results are shown in Figure 1 for the (arbitrarily chosen) classes "airplane", "automobile", and "bird". As was noted by [22], while label smoothing can help the classifier achieve better accuracy on the test set, it comes at the cost of a less informative learned representation, as can be seen from the more concentrated centroids in the second column relative to the first. On the other hand, CC-LS achieves a somewhat richer representation than vanilla LS, as can be observed from the greater within-cluster variances in the third column, which we quantify in Table 2. This suggests that the CC may offer additional potential for combining LS with teacher models in the context of knowledge distillation, a setting in which the concentrated clusters enforced by LS proved detrimental to the training of a student model [22].

Table 2: Ratio of within-cluster sum of squares over between-cluster sum of squares, for the learned representations of Figure 1. Each cell shows the mean ratio over 10 random initializations, with standard errors.

Samples	w/o LS	with LS	CC-LS
Training	18% ($\pm$1)	9% ($\pm$1)	12% ($\pm$1)
Test	25% ($\pm$1)	20% ($\pm$1)	23% ($\pm$1)

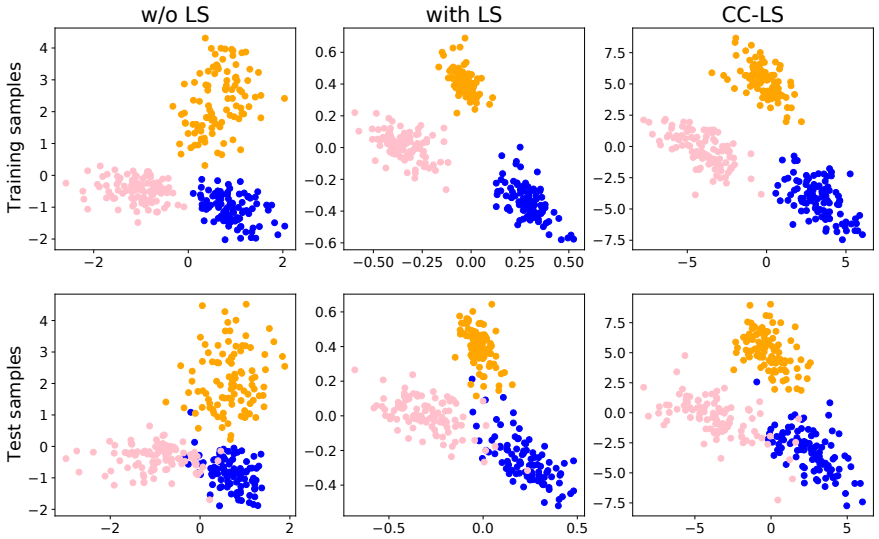

Figure 1: Learned representations for the classes "airplane" (blue), "automobile" (orange), and "bird" (pink), projected to an informative 2-dimensional affine subspace spanning the template-vectors of the 3 classes. We show the same plot for samples from the training (above) and test set (below), for the unsmoothed baseline (left), LS (middle), and CC-LS (right), trained with regularization (following the top row of Table 1). Note that CC-LS does not concentrate clusters as tightly as LS, suggesting the potential for richer learned representations.

4 Probabilistic Actor-Mimic Reinforcement Learning

In this section we summarize the Actor-Mimic Reinforcement Learning framework [23] and recast it as a compositional regression problem, highlighting the potential for a probabilistic model with the continuous-categorical distribution.

Actor-Mimic Networks (AMN) provide a method for multitask and transfer reinforcement learning. The goal of the AMN is to train a single agent to perform on several different "source games", $\{G_1, \ldots, G_L\}$, each of which corresponds to a Markov Decision Process defined on a common state space and action set (shared across source tasks), but driven by a different set of transition probabilities and reward functions (specific to each task). Formally, $G_i = (\mathcal{S}, \mathcal{A}, \mathcal{T}_i, \mathcal{R}_i)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{T}_i(s'|s, a)$ is the probability of transitioning from state s to state s' when executing action a in game G_i, and $\mathcal{R}_i$ is the reward function mapping states and actions to real-valued rewards representing the score of the ith game. In practice, each G_i corresponds to a different videogame from the Atari Learning Environment [4]; each game follows a different set of rules ($\mathcal{T}_i$ and $\mathcal{R}_i$) while taking place on the same console display ($\mathcal{S}$) and controller ($\mathcal{A}$).

In order to train an AMN, we first require access to a set of "experts" $\{E_1, \ldots, E_L\}$, each of which corresponds to an agent specialized in one of the source games. Expert E_i represents a policy π_{E_i} mapping states to distributions over actions, so that we can write $\pi_{E_i}(a_k|s_t)$ for the probability that E_i chooses action $a_k \in \mathcal{A}$ when in state $s_t \in \mathcal{S}$. Note that k indexes the action space, so that $\mathcal{A} = \{a_1, \ldots, a_K\}$, whereas t indexes time (i.e., frame number), so that $\mathcal{S} \supseteq \{s_1, s_2, \ldots\}$. In our implementation, E_i corresponds to a Deep Q-Network (DQN) [21] trained on game G_i, though the fact that E_i is a DQN is not necessary – any policy that performs well on G_i will suffice.

Given the set of expert policies, the AMN is trained to "mimic" the experts in their respective source games. We can reformulate the method as a two-stage process. First, we form an auxiliary dataset of "guidance vectors", $\mathcal{D}_{\text{aux}} = \left\{ y_t^{(i)} \right\}$. These vectors are obtained by generating, for each game G_i, a sequence of states $\left\{ s_t^{(i)} \right\}_{t=1}^{n}$, and then feeding these states through the corresponding expert policies,

Table 3: Mean evaluation score (and standard deviation) over the last 20 evaluation epochs (higher is better). With the exception of Pong, the performance of AMN and CC-AMN is similar.

Model	Breakout	Atlantis	Pong	SpaceInvaders
DQN	331 ($\pm$44)	32 833 ($\pm$14 430)	20.9 ($\pm$0.2)	442 ($\pm$119)
AMN	337 ($\pm$74)	31 558 ($\pm$9 084)	20.9 ($\pm$0.1)	415 ($\pm$126)
CC-AMN	320 ($\pm$66)	26 196 ($\pm$10 396)	8.8 ($\pm$11.9)	415 ($\pm$132)

in other words:

$$y_t^{(i)} = \left(\pi_{E_i}\left(a_1|s_t^{(i)}\right), \ldots, \pi_{E_i}\left(a_K|s_t^{(i)}\right)\right). \tag{9}$$

Second, the parameters of our Actor-Mimic Network, π_θ^{AM}, are learned by minimizing the categorical cross-entropy loss with respect to the auxiliary data:

$$\min_\theta \mathcal{L}^{\text{AMN}}(\theta) = -\sum_{t,i}\sum_k \pi_{E_i}\left(a_k|s_t^{(i)}\right) \cdot \log \pi_\theta^{\text{AM}}\left(a_k|s_t^{(i)}\right). \tag{10}$$

In practice, we minimize this loss using minibatch stochastic gradient descent, running the gameplay-generation in parallel with the gradient steps. The effectiveness of the AMN approach is fundamentally computational; the expert policies can be trained independently in parallel, and the AMN is much faster to optimize via the cross-entropy loss (10) than using policy gradients, as it is able to leverage the rich information from the expert policies directly, (with an entire guidance vector of probabilities containing information for all classes at each time step, rather than learning from noisy and biased n-step bootstrap estimates).

Since $y_t^{(i)}$ is a vector of probabilities over actions, our auxiliary data is simplex-valued rather than categorical. It is therefore clear from Equation 10 that the AMN model solves a compositional regression problem, whence we propose replacing the cross-entropy loss with its probabilistic counterpart, the continuous-categorical log-likelihood:

$$\min_\theta \mathcal{L}^{\text{CC-AMN}}(\theta) = -\sum_{t,i}\left\{\log C\left(\lambda_\theta^{\text{AM}}\left(s_t^{(i)}\right)\right) + \sum_k \pi_{E_i}\left(a_k|s_t^{(i)}\right) \cdot \log \lambda_\theta^{\text{AM}}\left(a_k|s_t^{(i)}\right)\right\}. \tag{11}$$

We call this model CC-AMN, and we compare its performance against the AMN model, as well as the DQN baseline.

4.1 Experiments

We follow the experimental setup of Parisotto et al [23], choosing a subset of games from the Atari Learning Environment in which the DQN model performed at super-human level. For each game, we pre-train a DQN with the same network architecture; these are then used as the expert policies. Our network architecture, described in appendix A.2, is taken directly from [21], and is also used for the AMN and CC-AMN models.

First, we reproduce the results of [23] and compare with our novel CC-AMN model, as shown in Table 3. The evaluation scores of the CC-AMN are similar to those of the AMN, except for the game of Pong, where using the CC likelihood leads to unstable training, resulting in worse performance and higher variability in the evaluation score. Note that both AMN and CC-AMN are generally able to achieve similar performance to the expert DQN.

Second, we focus specifically on the effect of the probabilistic objective (11) on network training by reducing the multi-task objective to a single-task objective, i.e., we no longer sum over i in Equation 10. This corresponds to running AMN and CC-AMN against the expert DQN, E_i, of a single game, and we do this separately for each game, as shown in Figure 2. While both CC-AMN and AMN are able to train much faster than the DQN, converging in just a few epochs, CC-AMN fails to outperform AMN, and can be slower to converge (Breakout) or worse overall (Pong).

The case of Pong highlights an important failure mode of CC-AMN, which also offers some insight as to why our model underperforms in the other games. The issue originates in the normalizing constant

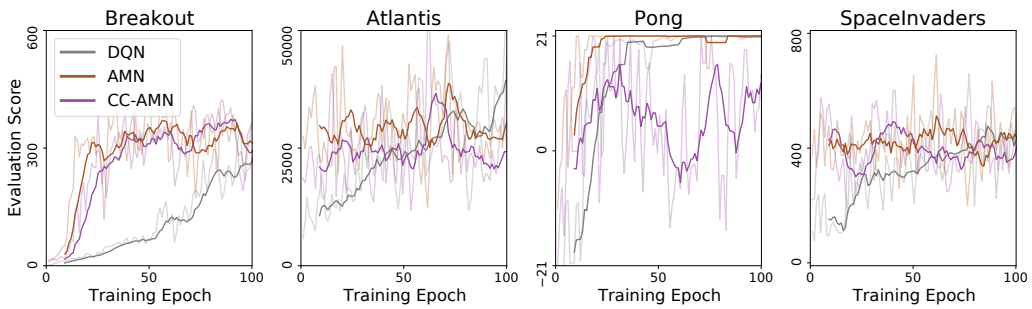

Figure 2: Training curves for the CC-AMN and AMN models, run on the simplified single-game objective. Solid lines reflect a moving average of the raw evaluation scores (faded lines). Each training epoch lasts $100\,000$ frames, with the evaluation scores being calculated from another $100\,000$ frames. While the actor-mimic models learn much faster than the DQN, the CC-AMN shows no improvement over the AMN model.

(4), which is numerically unstable when the parameter λ is close to uniform, due to the product of log-ratios vanishing in the in the denominator, as was noted in [12]. In the case of CC-AMN, our optimization hovers around this unstable region, since the guidance vectors tend to concentrate around the centroid of the simplex (this is because our expert policies correspond to the softmax of Q-value functions, which don't typically exhibit large variability across actions since, over small time steps, most actions are not individually critical to the outcome of the game). In practice this is not necessarily a problem, as we zero out the unstable gradients during optimization, as in [12]. Doing so provides a reasonable approximation, since $\nabla G(\lambda) = 0$ for $\lambda = (1/K, \ldots, 1/K)$, by symmetry ($\sum_k \lambda_k = 1$ is constrained by definition of the continuous-categorical). Nevertheless, this behavior likely results in a worse optimization landscape overall (unlike in CC-LS where y, and hence λ, are far away from the centroid), which in the game of Pong, derails our gradient search altogether. Further investigation may involve re-running our experiments with arbitrary-precision floating point, though we currently find this to be computationally prohibitive.

5 Discussion and Future Work

Comparing our experiments on CC-LS and CC-AMN, it may come as a surprise that the former yields the more promising empirical results, in spite of its less rigorous theoretical underpinning (with targets that are simplex-valued, but not genuinely continuous). This observation suggests that the continuous-categorical may provide useful modeling advances outside the realm of compositional data analysis and probabilistic modeling, for example in classification problems.

It is also worth noting that, throughout our experiments, the cross-entropy loss may have benefitted disproportionately from favorable network initialization, which has been developed for and become increasingly specialized toward networks with particular loss functions [13]. A similar argument can be made about network architecture, noting that a good architecture for the cross-entropy loss may not be equivalent to a good architecture for its continuous-categorical counterpart. In fact, we have observed experimentally that additional architecture or hyperparameter search can lead to improved performance for the CC-based approaches. However we deliberately chose not to focus on such experimentation, as doing so could further entangle the effect of the loss function on our models; we leave such analyses to future work.

At a more theoretical level, as identified in [12], it follows from the properties of exponential families that optimizing the continuous-categorical log-likelihood results in an unbiased estimator for its mean parameter, which corresponds to a (local) average of the observed data. This is, in fact, akin to the cross-entropy loss, which is also maximized at local average that approximates the conditional expectation of the outputs given the inputs. The optimization landscapes defined by the two loss functions could therefore be of similar nature, though this remains an open question.

Last, we highlight a computational limitation of our approach: the numerical instabilities noted in section 4.1 are exacerbated in high dimensions. In fact, evaluating Equation 4 for much more than 10

classes is problematic for all $\lambda \in \Delta^K$ (not only for λ around the centroid), since the K summands typically cancel out beyond numerical precision.[2] As a result, we constrained our experimentation to examples where $K \leq 10$, however, similar applications with $K \sim 100$ or greater are also of interest. Further advances in theory or numerical analysis will be needed to enable successful applications of the continuous-categorical at this scale.

We conclude by noting that, taken together with the theoretical and empirical results in [20] and [12], our work suggests that future methodological advances may be possible through a combination of careful probabilistic consideration of the cross-entropy loss, and the use of the continuous-categorical distribution.

Acknowledgments and Disclosure of Funding

We thank Andres Potapczynski and the anonymous reviewers for helpful conversations, and the Simons Foundation, Sloan Foundation, McKnight Endowment Fund, NSF 1707398, and the Gatsby Charitable Foundation for support.

References

[1] John Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):139–160, 1982.

[2] John Aitchison. Principles of compositional data analysis. *Lecture Notes-Monograph Series*, 24:73–81, 1994. ISSN 07492170.

[3] John Aitchison. Logratios and natural laws in compositional data analysis. *Mathematical Geology*, 31(5):563–580, 1999.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[5] Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[6] G Campbell and J Mosimann. Multivariate methods for proportional shape. In *ASA Proceedings of the Section on Statistical Graphics*, volume 1, pages 10–17. Washington, 1987.

[7] Guangyong Chen, Pengfei Chen, Yujun Shi, Chang-Yu Hsieh, Benben Liao, and Shengyu Zhang. Rethinking the usage of batch normalization and dropout in the training of deep neural networks. *arXiv preprint arXiv:1905.05928*, 2019.

[8] Jan Chorowski and Navdeep Jaitly. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016.

[9] Juan José Egozcue, Vera Pawlowsky-Glahn, Glòria Mateu-Figueras, and Carles Barcelo-Vidal. Isometric logratio transformations for compositional data analysis. *Mathematical Geology*, 35 (3):279–300, 2003.

[10] Angus Galloway, Anna Golubeva, Thomas Tanay, Medhat Moussa, and Graham W Taylor. Batch normalization is a cause of adversarial vulnerability. *arXiv preprint arXiv:1905.02161*, 2019.

[11] Christian Garbin, Xingquan Zhu, and Oge Marques. Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, pages 1–39, 2020.

[2] For an intuitive explanation, note that the Lebesgue measure of the K-dimensional simplex is $1/K!$. We therefore expect $C(\lambda)^{-1} \sim 1/K!$. However, as K increases, the "typical" summand in (4) decays slower than $1/K!$ (if at all). Thus, for large K, the individual summands in (4) will become much larger (in magnitude) than $C(\lambda)^{-1}$. Their summation will then result in (near) total cancellation and therefore total loss of numerical precision.

[12] Elliott Gordon-Rodriguez, Gabriel Loaiza-Ganem, and John P Cunningham. The continuous categorical: a novel simplex-valued exponential family. In *International Conference on Machine Learning*, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[14] Alex Hernández-García and Peter König. Do deep nets really need weight decay and dropout? *arXiv preprint arXiv:1802.07042*, 2018.

[15] Rafiq H Hijazi and Robert W Jernigan. Modelling compositional data using dirichlet regression models. *Journal of Applied Probability & Statistics*, 4(1):77–91, 2009.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[18] Anders Krogh and John A Hertz. A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957, 1992.

[19] Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2682–2690, 2019.

[20] Gabriel Loaiza-Ganem and John P Cunningham. The continuous bernoulli: fixing a pervasive error in variational autoencoders. In *Advances in Neural Information Processing Systems*, pages 13266–13276, 2019.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[22] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pages 4694–4703, 2019.

[23] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[24] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 4780–4789, 2019.

[25] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[27] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[28] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076, 2015.

[29] Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[31] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.

A Experimental details

A.1 Label Smoothing

We denote a convolutional layer by W×W×N-S, where W is the width of the convolution, N the number of filter maps, and S the stride. Our architecture is: 3×3×32-1 → BatchNorm → 3×3×32-1 → BatchNorm → MaxPooling (2×2) → Dropout (0.2) → 3×3×64-1 → BatchNorm → 3×3×64-1 → BatchNorm → MaxPooling (2×2) → Dropout (0.3) → 3×3×128-1 → BatchNorm → 3×3×128-1 → BatchNorm → MaxPooling (2×2) → Dropout (0.4) → 10 fully-connected units. We use weight decay of 0.0001 in the final fully-connected layer, which, together with dropout and batch normalization, was switched on and off in the different runs of our ablation study in Table 1. Our models were trained for 500 epochs using a minibatch size of 128 and the Adam optimizer with a learning rate of 10^{-3}. The label smoothing hyperparameter ε was set to 0.1 as per [22].

Note however, that we were unable to replicate the results of [22] exactly, as they did not share their code, nor did they describe their architecture in full.

A.2 Actor-Mimic Network

Our architecture is 8×8×32-4 → 4×4×64-2 → 3×3×64-1 → 7×7×1024-1 → 512 fully-connected units → 6 fully connected units (corresponding to 6 possible actions). We used the Adam optimizer with a learning rate of 10^{-5}, and a minibatch size of 32.