PRACTICAL INTEGRATION VIA SEPARABLE BIJECTIVE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural networks have enabled learning over examples that contain thousands of dimensions. However, most of these models are limited to training and evaluating on a finite collection of *points* and do not consider the hypervolume in which the data resides. Any analysis of the model's local or global behavior is therefore limited to very expensive or imprecise estimators. We propose to formulate neural networks as a composition of a bijective (flow) network followed by a learnable, separable network. This construction allows for learning (or assessing) over full hypervolumes with precise estimators at tractable computational cost via integration over the *input space*. We develop the necessary machinery, propose several practical integrals to use during training, and demonstrate their utility.

1 INTRODUCTION

Most supervised learning problems operate by training a model on a finite collection, $\mathcal{T}$, of N (typically paired) examples, (x, y). The model is updated by comparing its predicted output to the expected output and performing some flavor of stochastic gradient descent based on the comparison and various regularizers. The process is repeated by reexamining the elements of $\mathcal{T}$ in a random order, possibly with augmentation, until the model parameters converge or an iteration budget is exceeded. This relatively simple procedure has proven to be remarkably effective in a variety of domains and these models have begun to permeate every aspect of modern science and everyday life (Brown et al., 2020; He et al., 2016; Silver et al., 2017).

The deep learning revolution has also resulted in highly effective generative models such as VAEs (Kingma & Welling, 2014), GANs (Goodfellow et al., 2014), and tractable likelihood models (Dinh et al., 2017; Grathwohl et al., 2019). These models are largely used to create novel samples of impressive quality. In addition to sampling, tractable likelihood models also provide an estimate of the probability density function of the data which can be used for additional, downstream processes.

We *augment the training process* by constructing neural networks that allow for tractable integration over the *input domain*. This differs from implicit layers which utilize integration over a *parameterized* variable (Chen et al., 2018; Grathwohl et al., 2019). Access to fast and differentiable integrals allows us to regularize a model's *average* behavior using metrics that may not be available otherwise.

Integration over the input space also allows us to supervise how the model behaves in *continuous regions that are not directly observed* in $\mathcal{T}$ and may even be out-of-distribution (OOD). Alternative methods attempt to supervise examples outside of $\mathcal{T}$ by performing random perturbations (Gutmann & Hyvärinen, 2010), along the line between known examples (Zhang et al., 2018), or via a generative process (Zenati et al., 2018; Akcay et al., 2018). However, these methods are only capable of observing a small quantity of the total space. By integrating over the entire regions, it is possible to observe a large portion of the space based on statistical relevance.

Main Contributions We propose a general architecture that enables tractable integration over the input space, enabling supervision and custom regularization over continuous regions. We demonstrate how to construct this network and how it allows for a reduction in the computation cost required for dense numeric integration from exponential in the number of dimensions to linear. We derive several useful integrated formulations over continuous regions. Finally, we explore the impact of this architecture and regularizers on the standard accuracy and robustness to OOD examples on several standard classification datasets.

Notation Throughout this work, we consider the M dimensional input features, $\mathbf{x} = [x_1, ..., x_M] \in \mathbb{R}^M$; the latent features, $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^M$; and the K-wise classification probability, y. The input features $\mathbf{x}$ the training set, $\mathcal{T}_x$, are drawn from the in-distribution data, $\mathcal{D} \subseteq \mathbb{R}^M$. Subscripts represent a particular dimension, e.g., $\mathcal{D}_m$ corresponds to the mth dimension of the space. Paranthetical superscripts represent the subspace corresponding to a particular class, e.g., $\mathcal{D}^{(c)}$ is the subset of $\mathcal{D}$ where the data belongs to class c. The bijective network is given as h such that $h : \mathcal{D} \to \mathcal{Z}$. Probability distributions over $\mathcal{D}$ and $\mathcal{Z}$ are given by p with the corresponding subscript. Classification networks are given as f and g. Variables with a "hat," $\hat{y}$, are predictions of the true quantity, y.

2 MOTIVATION

Neural networks are highly effective function approximators between two (typically) continuous spaces: $f : \mathcal{X} \to \mathcal{Y}$. However, networks are typically trained and evaluated using a finite collection of *points* without any explicit assessment of the complete hypervolume spanned by the data. This omission is understandable from an implementation perspective as the number of samples required to obtain a reasonable estimate over a volume scales exponentially with data dimensionality. However, human beings often have an understanding of how a process should behave *on average*. Ideally, we would like to embed this intuition into the model but currently cannot assess the average performance of a trained model outside of the held-out test set. Specifically, we would like to regularize the model by estimating the expected behavior of some metric, Ω, produced by the model over the training data

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}\left[\Omega(\hat{y}(\mathbf{x}))\right] = \int_{\mathcal{X}} \Omega(\hat{y}(\mathbf{x}))p(\mathbf{x})d\mathbf{x}. \tag{1}$$

There are many useful choices of Ω over a variety of applications. If it is known what the model output should be on average ($\bar{y}$), we can construct Ω to encourage that behavior, e.g., $\Omega(\hat{y}) = (\bar{y} - \hat{y})^2$. Minimizing consistency metrics (Xie et al., 2020) are a common method to improve learning in label-starved problems. These encourage the model to produce similar outputs over neighborhoods around (labeled) examples from $\mathcal{T}_x$ where neighborhoods are created by random or domain-specific augmentations. This process can be viewed as an approximation to an integral over the neighborhood, e.g.,

$$\mathbb{E}_{\epsilon\sim p(\epsilon)}\left[\mathcal{L}(y, \hat{y}(\mathbf{x}+\epsilon))\right] = \int \mathcal{L}(y, \hat{y}(\mathbf{x}+\epsilon))p(\epsilon)d\epsilon \tag{2}$$

where $\mathcal{L}$ is a distance-like metric, and ϵ is the neighborhood. Equation 2 can be generalized to other neighborhoods. We can recast the standard classification problem as a discrete approximation to an integral. Typically, we minimize the cross-entropy between $\hat{y}(\mathbf{x};\theta)$ and y over the model parameters, θ, for all $(\mathbf{x}, y) \in \mathcal{T}$ which becomes an integral over class-conditioned distributions, $p(\mathbf{x}|c)$,

$$\min_{\theta} - \sum_{x,y\in\mathcal{T}} \sum_{k} y_k \log\left(\hat{y}_k(x;\theta)\right) \Rightarrow \min_{\theta} - \sum_{k} \int_{\mathcal{D}^{(k)}} y_k \log\left(\hat{y}_k(x;\theta)\right) p(\mathbf{x}|k)d\mathbf{x}. \tag{3}$$

Unfortunately, integration in high dimension is difficult. Naive gridded solutions require an exponential number of points with error decreasing as $O(G^{-M})$, for G, M-dimensional points. Monte Carlo (MC) methods theoretically have better performance with error that decreases as $O(G^{-1/2})$. However, the rate of convergence for MC methods depends on the variance of samples (Veach, 1998), which may make for poor approximations in practice. Importance sampling (Bishop, 2006) can improve the performance of Monte Carlo methods to adapt to the regions with the greatest contribution.

We choose to model the data using a separable function. Separable functions have the key benefit of decomposing M-dimensional integrals into a combination of M one-dimensional integrals. Each of these one-dimensional integrals can then be solved using any number of highly-accurate solvers (e.g., Runge-Kutta (Shampine, 2005), Dormand-Prince (Dormand & Prince, 1980), Tsit (Tsitouras, 2011), etc.) that have error rates better than $O(G^{-4})$ but are unavailable in high dimensions. The use of separable functions is a component of the VEGAS algorithm (Peter Lepage, 1978) and is utilized in conjunction with adaptive Monte Carlo sampling to approximate high-dimensional integrals.

The use of a separable function over the input space may make estimation of integrals over the model more accessible; however, they impose a strong, inappropriate, inductive-bias. The obvious solution of using a standard neural network feature extractor and integrating over the learned features means

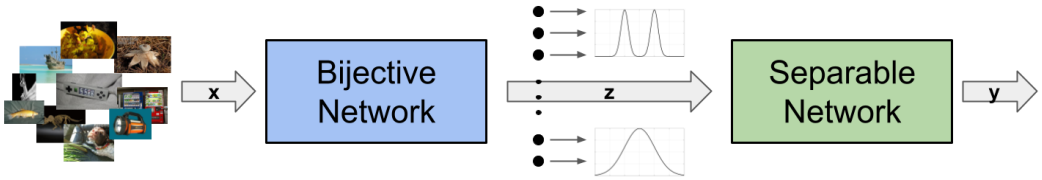


Figure 1: Depiction of the overall network with intervening distributions over the latent space, $\mathcal{Z}$

that we would no longer have a valid integrator over the input space. Fortunately, we can solve this problem by including a bijective transform prior to the separable network as a feature extractor. The bijective transform allows us to decouple the data into a latent space where the data can be modeled using a separable network and guarantees equality between integrals in the latent space and integrals in the input space. The bijective model has the added benefit of allowing us to create a tractable likelihood model (Dinh et al., 2017) over the data which gives us access to $p(\mathbf{x})$ (or possibly $p(\mathbf{x}|y)$) and makes it easier to draw new samples. We discuss bijective and separable functions in Sec. 3, how they come together in Sec. 4, and the specific choices we use in Sec. 5.

3 Background

We perform practical integration over the input space by breaking neural network models down into two key components: (1) a *bijective* feature extractor, (2) a *separable* task network, see Fig. 1. For the sake of simplicity, we only consider classification tasks in this work. This makes our total network analogous with the common network breakdown where a classifier, often a linear layer or an MLP, is constructed on a feature extractor, such as a CNN. However, unlike the typical process, we must place constraints on both networks so that we can integrate over the input domain. This network breakdown is similar to hybrid networks (Chen et al., 2019; Nalisnick et al., 2019b) except for the separability requirement on the classifier.

3.1 Bijective Networks

Bijective networks are the key component in flow-based likelihood models. A bijective network, $h : \mathcal{D} \rightarrow \mathcal{Z}$, has a known forward and inverse operation so that data can exactly be reconstructed after the transformation. This allows for exact likelihood estimation via the change of variables formula:

$$\mathbf{z} = h(\mathbf{x};\ \theta), \quad \mathbf{x} = h^{-1}(\mathbf{z};\ \theta), \quad p_X(\mathbf{x}) = p_Z(h(\mathbf{x};\ \theta)) \left| \frac{\partial h}{\partial \mathbf{x}} \right| \tag{4}$$

where p_Z is a predefined distribution over the latent space, often a standard Gaussian.

These models are trained via Eq. 4 to maximize the likelihood of the examples in $\mathcal{T}$. Once trained, flow-based likelihood models are commonly used as a generative process where samples are drawn from p_Z and are then inverted through h to arrive at an example in $\mathcal{D}$. Our goal is not generative, however it is possible for us to use this process once training is complete. Instead, we will take advantage of the fact that we can choose p_Z and then utilize it in downstream tasks.

Flow models are well-studied and we take advantage of several recent architectures to test our methods. In particular, we consider three different bijective models. The simplest models are constructed via stacked affine-coupling layers (Dinh et al., 2017) and invertible, fully-connected layers. Image data is processed using Glow's (Kingma & Dhariwal, 2018) invertible 1×1 convolutional operations.

3.2 Separable Functions

Separable functions have long been used in mathematics and physics to solve simple partial differential equations such as the homogeneous wave and diffusion equations (Strauss, 2007). We consider two types of separable functions, additive and multiplicative. All proofs can be found in Appendix A.

3.2.1 ADDITIVELY SEPARABLE FUNCTIONS

Definition 3.1. A function, $f : \mathbb{C}^M \to \mathbb{C}^K$, is additively separable if it is composed as a summation of element-wise functions operating independently on each dimension of the input:

$$f(\mathbf{v}) = \sum_{m=1}^{M} f_m(v_m; \phi_m) \tag{5}$$

Theorem 3.1 (Additive Independent Integration)**.** *Given an additively separable function, $f(\mathbf{v})$, an independent likelihood, $p(\mathbf{v}) = \prod_{m=1}^{M} p_m(v_m)$, and domain, $\mathcal{D}_v = \mathcal{D}_{v_1} \times ... \times \mathcal{D}_{v_M}$:*

$$\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}[f(\mathbf{v})] = \sum_{m=1}^{M} \int_{\mathcal{D}_{v_m}} f_m(v_m)\, p_m(v_m)\, dv_m \tag{6}$$

3.2.2 MULTIPLICATIVELY SEPARABLE

Definition 3.2. A function, $g : \mathbb{C}^M \to \mathbb{C}^K$, is multiplicatively separable if it is composed as a product of element-wise functions operating independently on each dimension of the input:

$$g(\mathbf{v}) = \prod_{m=1}^{M} g_m(v_m; \psi_m) \tag{7}$$

Theorem 3.2 (Multiplicative Independent Integration)**.** *Given a multiplicitively separable function, $g(\mathbf{v})$, an independent likelihood, $p(\mathbf{v}) = \prod_{m=1}^{M} p_m(v_m)$, and domain, $\mathcal{D}_v = \mathcal{D}_{v_1} \times ... \times \mathcal{D}_{v_M}$:*

$$\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}[g(\mathbf{v})] = \prod_{m=1}^{M} \int_{\mathcal{D}_{v_m}} g_m(v_m)\, p_m(v_m)\, dv_m \tag{8}$$

4 METHOD

Both forms of separable functions allow us to convert a single M-dimensional integral into M 1-dimensional integrals. If we desired to perform a dense estimation of the integral without taking advantage of a separable function using G points per dimension, we would require $\mathcal{O}(G^M)$ network evaluations. This is completely impractical for modern datasets where M is at least on the order of hundreds and G should be as large as possible. Exploiting separable functions allow us to reduce the number of network evaluations needed to estimate the integral to $\mathcal{O}(G)$.

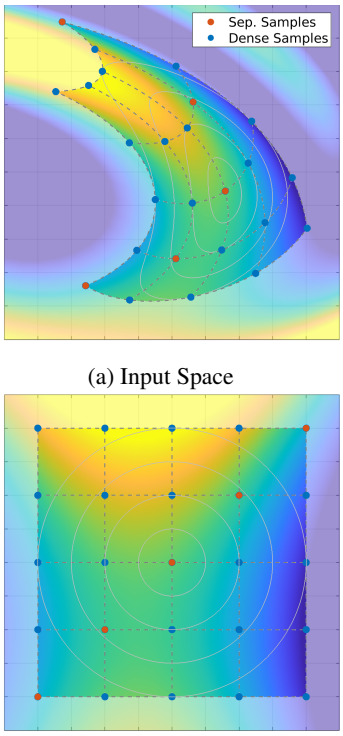
However, it is unlikely that we could construct a separable function directly on the input space and achieve reasonable performance. Doing so would essentially require that each input dimension contribute to the final output independently of the others. Instead, we can combine Eq. 4 and Eq. 6 (alternatively, Eq. 8) to perform practical integration over the input domain while still allowing for inter-dependent contributions from each input dimension.

To do so, we first choose to let the latent distribution, p_Z, be independently (but not necessarily identically) distributed: $p_Z(z) = \prod_m p_m(z_m)$. This allows us to write the integral over the input space in terms of the latent space and then simplify via Eq. 6.

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[f(h(\mathbf{x}))] = \int_{\mathcal{D}} f(h(\mathbf{x}))p_Z(h(\mathbf{x}))\left|\frac{\partial h}{\partial \mathbf{x}}\right| d\mathbf{x} = \sum_{m=1}^{M} \int_{\mathcal{Z}_m} f_m(z_m)\, p_m(z_m)\, dz_m \tag{9}$$

Each 1-dimensional integral is easily approximated using a variety of integration approaches. In addition to the requirements placed on the feature extractor and task network, this formulation also requires that we define the integration domain in the latent space as a Cartesian product of domains over each latent dimension. This may seem like a strong requirement since we apparently lose some level of interpretability; however, there are several advantages to defining the input domain in this learned space. Most notably, we know the data distribution in this space exactly and can tune the domain based on the goal of a particular integral. For example, if we are interested in performing average adversarial training, we might choose to integrate in directions that appear less likely as this takes us further from the data domain.

Figure 2 contains a cartoon example that demonstrates how these methods combine to allow for integration over the complex data space. Both plots show the value of f; Fig. 2a shows the non-separable function in the input space and Fig. 2b shows the same data in the (separable) latent space, after the bijective transformation. The colored points and dotted lines are in correspondence between the two spaces and illustrate how the space is warped by the bijector to create a separable, independent latent space. The light gray lines represent the contours of the underlying probability distribution. We define both the integration domain and perform the integration in the latent space. For simplicity, we illustrate the integration using five, equally-spaced points (in both dimensions). The naive procedure would require sampling f at each point in the grid (the blue points). The use of the separable functions allow us to perform the integral using only the red points (or any combination of the blue points, so long as the value from each row/column is included) and get the same estimate of the integral.



(a) Input Space

(b) Latent Space

Figure 2: $f(h(\mathbf{x}))$ and $f(\mathbf{z})$. The separable function allows for $O(G)$ samples (red) in the latent space that would require $O(G^M)$ samples (blue) in the input space. Emphasis added to the integration region.

5 Practical Applications

In this section we discuss the choices we make in practice when constructing the latent distribution, separable networks, and the various integrals used. For classification tasks, it is not possible to parameterize the normalized network output of class probabilities, $\hat{y}$, as a separable network since each prediction must sum to one. However, it is possible to parameterize each unnormalized logits as a separable network, e.g., $\hat{y}(\mathbf{x}) = \sigma\left(\mathbf{f}(h(\mathbf{x}))\right)$, where σ is the softmax operator and the output of $\mathbf{f}$ are the unnormalized logits. This limits what quantities can be integrated over exactly. Fortunately, it is often possible to integrate over approximations/bounds of a particular metric without resorting to brute-force methods.

5.1 Out-of-Distribution Detection

We construct a global integration regularizaton to provide resilience to out-of-distribution (OOD) examples. We enable OOD detection by introducing a "reject" option (Hendrickx et al., 2021) into the classification vector, increasing the number of classes by one, where the additional class indicates that the instance is OOD. An example is considered OOD if $\hat{y}_{K+1}$ exceeds a predefined threshold. We supervise the model over out-of-distribution examples by integrating the cross-entropy over the contrastive probability distribution, $q(\mathbf{z})$ (see Sec. 5.4).

5.2 Latent Distributions

A critical component of this method is that the likelihoods over the latent space must be independent: $p_Z(\mathbf{z}) = \prod_m p_Z(z_m)$. As this is extremely unlikely to occur in the input space, we learn a bijective (flow) transformation from the input space to a latent space where the likelihood can be decomposed into independent components of our choice. The typical choice for the latent distribution is a standard Gaussian. However, since we would like to use the latent features to discriminate between classes using a separable function, we choose to utilize a (bimodal) Gaussian mixture model. This allows the model to put different classes into one of two "buckets" per feature and enables classification through multiple features. This results in an exponential number of components (with respect to dimension) in the full latent space, e.g., 2^M components. However, we can easily offset this by choosing some dimensions to use a unimodal latent distribution while the remaining dimensions are still bimodal

$$p_{Z_m}(z_m) = \begin{cases} 0.5\,\mathcal{N}(-\mu, \sigma_1^2) + 0.5\,\mathcal{N}(\mu, \sigma_1^2), & \text{if } m \leq L \\ \mathcal{N}(0, \sigma_2^2), & \text{else} \end{cases} \tag{10}$$

In addition to the typical latent data distribution, $p_Z(\mathbf{z})$, we also explicitly include a contrastive distribution, $q(\mathbf{z})$. This distribution is critical if we desire to regularize our model outside the data distribution, i.e., setting some background/default behavior. When using a bimodal data distribution, we additionally desire that latent vectors are more likely under this distribution once the vector is sufficiently far away from any single in-distribution mode. Therefore, we select this distribution so that it fully encapsulates all the components of the data distribution, e.g., $q_m(z_m) > p_{Z_m}(z_m)$ for $|\,|z_m| - \mu| > \gamma$. When the data distribution is unimodal we choose $p_m = q_m$ so that the feature is uninformative.

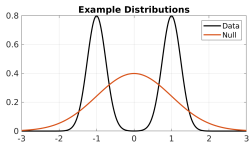


Figure 3: Data and contrastive distributions

We utilize a standard Gaussian for the contrastive distribution and set μ to 1 and experiment with $\sigma_1 \leq 0.5$. This choice allows us to formulate OOD examples that exist between the in-distribution data as near zero, and outside the in-distribution data, near the tails. Figure 3 illustrates the data and contrastive distributions for a latent feature for convenience.

5.3 Separable Networks

We explore three different formulations of separable networks. The first formulation learns a distinct quadratic function, the second learns a symmetric hinge, and the third learns a multi-layer perceptron. These functions have distinct parameterizations for each latent feature and each in-distribution class. The out-of-distribution logit is held fixed at zero: $l_{K+1}(\mathbf{x}) = 0$.

$$f_{k,m}^{(\text{quad})} = \alpha_{k,m} - \frac{(x_m - u_{k,m})^2}{2e^{2\nu_{k,m}}}, \qquad f_{k,m}^{(\text{hinge})} = \alpha_{k,m} - |x_m - u_{k,m}|\, e^{\nu_{k,m}} \tag{11}$$

where k is the logit class index and m is the feature dimension so that the total unnormalized logit, $l_k(\mathbf{x})$, is $l_k(\mathbf{x}) = \sum_m f_{k,m}(x_m)$. The separable MLP is constructed by concatenating a learned vector per feature and class to each 1-dimensional feature and constructing a standard MLP that operates on that augmented state. This formulation provides the greatest flexibility as it leverages all the benefits of the universal approximator theorem (Hornik, 1991). Unfortunately, we find the MLP version difficult to train in practice, often resulting in degenerate solutions.

Fixing the OOD logit at zero provides a fixed reference point for the other logits. We can interpret this formulation as each latent feature getting a vote on how in-distribution the example is on a per-class basis. Since each function, f_k, is exponentiated after the sum over features (due to the softmax normalization) we can also interpret the process as learning a Gaussian or Laplacian (for the quadratic or hinge parameterization, respectively) over the latent data. This is not exactly true due to the normalization, but it is a convenient intuition.

5.4 Integrals

The cross-entropy, $\text{CE}(\hat{y}(\mathbf{x}), y)$, between the normalized model output, $\hat{y}(\mathbf{x})$, and the true labels, y, is commonly used to train classification models. In this section, we explore global and local integrals of the cross-entropy loss used to train most classifiers (see Eq. 3). In practice, we can express this as $\min_\theta \sum_{\mathbf{x},y \in \mathcal{T}} \text{CE}(\hat{y}(\mathbf{x}), y)$ owing to the finite number of elements in $\mathcal{T}$. Ideally, we would minimize the model cross-entropy over all possible examples in $\mathcal{D}$. We can accomplish this by defining the subspace, $\mathcal{D}^{(c)} \subset \mathcal{D}$, that has a given label and corresponding conditional distribution, $p(\mathbf{x}|c)$

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|c)}\left[\text{CE}\left(\hat{y}(\mathbf{x}), \mathbf{c}\right)\right] = -\,\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|c)}\left[\log(\hat{y}_c(\mathbf{x}))\right] = -\int_{\mathcal{D}^{(c)}} \log(\hat{y}_c(\mathbf{x}))\; p(\mathbf{x}|c) d\mathbf{x}. \tag{12}$$

It is not possible to represent $\hat{y}(\mathbf{x})$ as separable network due to the softmax operation but it is possible to parameterize the unnormalized logits as a separable network. Substituting this parameterization into Eq. 12 and transforming to the latent space yields a bound for the expected cross-entropy

$$\begin{aligned}\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|c)}\left[\text{CE}\left(\hat{y}(\mathbf{x}), y\right)\right] \leq &-\sum_{m=1}^{M} \int_{\mathcal{Z}^{(c)}} f_{c,m}(z_m) p_m(z_m|c) dz_m \\ &+ \log\left(\sum_{j=1}^{K+1} \prod_{n=1}^{M} \int_{\mathcal{Z}^{(c)}} \exp\left(f_{j,n}(z_n)\right) p_m(z_m|c) dz_n\right).\end{aligned} \tag{13}$$

See Appendix E for a full derivation. We will utilize this formulation in conjunction with a contrastive prior to supervise OOD examples, Sec. 5.4.1, and with a local prior to enforce consistency, Sec. 5.4.2.

5.4.1 OUT-OF-DISTRIBUTION SUPERVISION

We can utilize the cross-entropy integral in different ways by choosing the label we would like to apply over a domain. For example, we can integrate the cross-entropy over the OOD latent space, $\mathcal{U}$, with the true label fixed to the reject class (c=K+1), and the latent distribution over codes is the contrastive distribution (Sec. 5.2), $p(z|c{=}K{+}1) = q(z)$:

$$\mathcal{L}_{\text{GLBL}} = \log \left(\sum_{j=1}^{K} \prod_{n=1}^{M} \int_{\mathcal{U}} \exp\left(f_{j,n}(z_n)\right) q_n(z_n) dz_n \right), \tag{14}$$

with $f_{K+1,n}{=}0$. Since the data and contrastive distributions overlap, the model could degenerate and always make OOD decisions; however, this would be in conflict with the standard cross-entropy loss applied to each example in the training set. So long as $\mathcal{L}_{\text{GLBL}}$ is not weighted too strongly, the model achieves equilibrium by making OOD predictions over regions where the contrastive distribution is more likely. In effect, we supervise OOD training without requiring any OOD data.

5.4.2 LOCAL CONSISTENCY

Alternatively, we can perform the integration over neighborhoods of data points in $\mathcal{T}_x$ and utilize that point's label over the entire region. This integral serves to improve consistency around observations. Adversarial attacks (Goodfellow et al., 2015; Madry et al., 2018) are often constructed as perturbations on real data points and adversarial training finds one such point and includes it in the training set. We can interpret local consistency integrations as a form of *average* adversarial training. We reformulate the integral to be centered around a particular datum, $\mathbf{x}_0$ and its class label, y

$$\mathcal{L}_{\text{LCL}} = -\sum_{k} y_k \int_{\mathcal{V}} \log(\sigma\left(f_k(h(\mathbf{x}_0) + \mathbf{v})\right)) \, p_V(\mathbf{v}) d\mathbf{v} \tag{15}$$

A complication resulting from local integration is how to select the local distribution, $p_V(\mathbf{v})$, and the neighborhood, $\mathcal{V}$. There are many reasonable choices depending on the goal of the local integration. For simplicity, we choose each $\mathcal{V}_m \in [-\varepsilon, \varepsilon]$ and $p_m(v_m)$ to be uniformly distributed.

5.5 LOSS COMPONENTS

The final training loss used to evaluate the model is composed of different combinations of the standard cross-entropy loss over class predictions, $\mathcal{L}_{\text{CE}} = -\sum_k y_k \log(\hat{y})$, the negative log-likelihood loss over in-distribution data points, $\mathcal{L}_{\text{NLL}} = -\log(p_{\mathcal{Z}}(h(\mathbf{x}))) - \log\left|\frac{\partial h}{\partial \mathbf{x}}\right|$, and the integration losses, $\mathcal{L}_{\text{GLBL}}$ and $\mathcal{L}_{\text{LCL}}$. We always include $\mathcal{L}_{\text{CE}}$ and $\mathcal{L}_{\text{NLL}}$. Based on results from previous hybrid networks (Chen et al., 2019; Nalisnick et al., 2019b), we weight the negative log-likelihood by $1/M$, which is analogous to using bits per dimension and introduce an additional weight over the cross-entropy, λ, which controls the relative importance of the generative and classification tasks. When included, each integration penalty shares the same weight as $\mathcal{L}_{\text{CE}}$ with additional binary weight, π

$$\mathcal{L}_{\text{total}} = \frac{1}{M}\mathcal{L}_{\text{NLL}} + \lambda\left(\mathcal{L}_{\text{CE}} + \pi_{\text{GLBL}}\mathcal{L}_{\text{GLBL}} + \pi_{\text{LCL}}\mathcal{L}_{\text{LCL}}\right). \tag{16}$$

6 RELATED WORK

Noise Contrastive Distributions Noise contrastive methods introduce a distribution that is distinct from the data distribution, preferably with minimal overlap. The constrastive distribution can be learned and provides a mechanism to supervise the model to discriminate between true and contrastive samples (Gutmann & Hyvärinen, 2010). This forces the model to learn discriminative statistical properties of the data. We utilize this idea to inform the null-space over which we will integrate. We fix the data and contrastive distributions and learn the bijective map between input and latent spaces.

Hybrid Models Normalizing flows are a family of flexible, tractable likelihood estimators based on the application of learnable, bijective functions (Dinh et al., 2017; Grathwohl et al., 2019). These methods have shown strong performance in a variety of domains, both as likelihood estimators and as generative processes. Recent work has coupled these advances in bijective networks to construct

invertible feature extractors that are capped by a more conventional classifier. This combination bijective/classifier structure are called *hybrid models* and show reasonable performance as likelihood and discriminative models (Chen et al., 2019; Nalisnick et al., 2019b). A known shortcoming of these models is that their descriminative performance is lower than those of more traditional models. We utilize a hybrid model structure but with additional constraints over the classification network to enable tractable integration. To our knowledge, other hybrid models construct their classifiers in a way that is less restricted but prohibits practical integration.

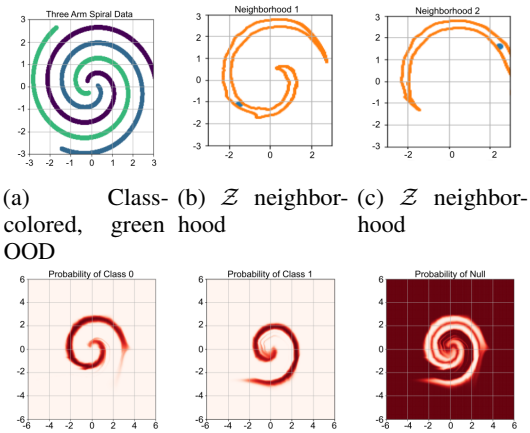
Out-of-Distribution Detection Out-of-distribution (OOD) detection is a challenge for many modern machine learning algorithms. An obvious method to assess if an example is OOD would be to train a likelihood model over the data space and threshold since OOD samples should have lower likelihoods than the in-distribution data. Surprisingly, recent work has demonstrated that data that should be OOD actually achieves higher likelihood than the data used in training (Hendrycks et al., 2019; Nalisnick et al., 2019a). Several recent methods have been developed to detect OOD examples including Maximum Softmax Probability (MSP) (Hendrycks & Gimpel, 2017), Outlier Exposure (OE) (Hendrycks et al., 2019), Multiscale Score Matching (MSMA) (Mahmood et al., 2021), ODIN (Liang et al., 2018), and Certified Certain Uncertainty (CCU) (Meinke & Hein, 2020). Lee et al. (2018) utilize a GAN trained with a classifier to produce examples near but out side the training distribution. These methods are constructed specifically for OOD detection whereas our method could be applied to a variety of problems based on the continuous regularizers chosen.

7 Experiments

All models were constructed using PyTorch (Paszke et al., 2019), trained using PyTorch-Lightning (Falcon, 2019), utilized bijectors and distributions from Pyro (Bingham et al., 2018), and were trained using Adam (Kingma & Ba, 2015). Here, we assess the integrable model's performance using the standard accuracy against a held-out set, the bits-per-dimension (BPD), and the impact on OOD detection. Additional experiments can be found in Appendix B.

7.1 Spirals

We construct a synthetic, 2D dataset composed of three intertwined spirals, see Fig. 4a. Suppose that due to an unknown sampling bias, we only observe two of the three arms (missing the green arm) and constructed our model based on these observations as a two-class problem with a third, reject class. See Appendix C for architectures, distributions, and training parameters.



(a) Class-colored, green OOD (b) $\mathcal{Z}$ neighborhood (c) $\mathcal{Z}$ neighborhood

(d) $p(y = 0)$ (e) $p(y = 1)$ (f) $p(\mathbf{x} \in \text{OOD})$

Figure 4: Three-arm spirals results

We sample points in a dense 2-dimensional grid at twice the range of the data in both dimensions and evaluate the probability that each point belongs to the two known classes and the OOD class. Figures 4d and 4e illustrate the probability of belonging to the two known classes and Fig. 4f contains the probability that each point is OOD. Red and white values indicate high and low probabilities, respectively. The model successfully identifies the two in-distribution regions. Unsurprisingly, the model also identifies data outside of the training data range as being OOD and, impressively, identifies the unobserved spiral and the region between spirals as being OOD.

Finally, we sample a square box (orange) around a data point (blue) in the latent space and invert the box to assess the appropriateness of the neighborhood in the input space and plot the result, Figs. 4b and 4c. As expected, the bijector converts the latent Euclidean neighborhood into a semantically meaningful neighborhood. Had we included the local cross-entropy integral in this training process, all points within the orange boundary would have received the same label as the data point. However, the fact that the neighborhoods are fairly narrow in directions perpendicular to the manifold indicates

that a naive local distribution would have washed out most of the adversarial directions and may reduce the effectiveness of average adversarial training.

7.2 Standard Performance

We test the impact of integrable models on OOD performance against several standard image classification datasets: MNIST (Deng, 2012), Fashion MNIST (FMNIST) (Xiao et al., 2017), SVHN (Netzer et al., 2011), and CIFAR10 (Krizhevsky et al., 2009). See Appendix B for architures, distributions, and training parameters. Table 1 contains the validation standard accuracy and bits per dimension for all datasets with all integration regularizers. The upper portion of the table is averaged over 10 runs and contains a reject option in the separable model. The lower portion is a single run without a reject option or any integrated regularizers. We find that the model performs reasonably well for MNIST and FMNIST but the integrated losses cause a large degredation in performance for SVHN and CIFAR10. The removal of these losses produces similar accuracies but much-improved BPD, consistent with the hybrid network results reported by ResidualFlows (Chen et al., 2019) when Glow is used.

Table 1: Standard accuracy and bits-per-dimension for different datasets

	Acc.	BPD
MNIST	98.3 ± 0.2	2.54 ± 0.13
FMNIST	88.1 ± 2.3	4.76 ± 0.64
CIFAR10	73.4 ± 1.8	5.62 ± 0.76
SVHN	89.9 ± 0.7	4.14 ± 0.08
CIFAR10*	76.5	3.78
SVHN*	90.0	2.24

7.3 Out of Distribution Detection

We test how our models respond to OOD examples when trained with the global integral over the noise-contrastive prior and the consistency integral and compare to methods designed specifically for OOD detection. See Sec. 6 for a discussion of the baselines. Table 2 contains the AUPR for the various methods against similar but clearly different datasets (the AUROC can be found in Appendix B). We juxtapose SVHN vs CIFAR10 and MNIST vs FMNIST and also include Extended MNIST (EMNIST) (Cohen et al., 2017). We include a separable, hybrid model without integral regularizations (hybrid) and the same model with regularizations (Int. Reg.) to illustrate the utility of learning over hypervolumes. When either MNIST or FMNIST are the in-distribution set, the regularized, integrable network performs on par with the best baseline methods. SVHN and CIFAR10 show reasonable OOD performance but are not as strong as the baselines. This is not surprising since the integrals rely on a reasonable estimate of $p(\mathbf{x})$, which both datasets have failed to achieve, Table 1.

Table 2: Area under the PR curve (percentage).

		GAN	ODIN	MSMA	OE	CCU	Hybrid	Int. Reg.
MNIST	FMNIST	99.4	98.8	-	99.9	99.9	93.7 ± 6.8	99.7 ± 0.17
	EMNIST	84.5	78.4	-	91.4	84.3	97.2 ± 1.9	99.8 ± 0.03
FMNIST	MNIST	99.9	99.2	80.8	97.0	98.3	63.9 ± 8.3	95.4 ± 0.04
	EMNIST	100.	99.3	-	98.6	99.1	93.8 ± 3.5	98.8 ± 0.47
SVHN	CIFAR10	98.6	97.3	92.5	100.	100.	71.8 ± 2.0	76.8 ± 1.4
CIFAR10	SVHN	80.5	92.7	99.0	98.5	97.5	84.6 ± 0.8	87.0 ± 2.1

8 Conclusions

In this work, we develop architectures that enable tractable integration over the data space. We demonstrate that the ability to supervise regions and not isolated points encourages the model to learn better representations and be less likely to degenerate. We consider several formulations that allow us to regularize the model's behavior based on consistency and contrast without relying on augmentations of and sparse comparisons between a finite collection of data points. We experiment with the various integrals to obtain promising out-of-distribution detection. Through now tractable integrals, this work enables future methods and applications for learning over continuous regions.

REFERENCES

Faruk Ahmed and Aaron Courville. Detecting semantic anomalies. In *Proceedings of 34th AAAI Conference on Artificial Intelligence*, 2020.

Samet Akcay, Amir Atapour-Abarghouei, and Toby P Breckon. Ganomaly: Semi-supervised anomaly detection via adversarial training. In *Asian Conference on Computer Vision*, pp. 622–637. Springer, 2018.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 274–283. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/athalye18a.html.

Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 2018.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Ricky T. Q. Chen, Jens Behrmann, David K Duvenaud, and Joern-Henrik Jacobsen. Residual flows for invertible generative modeling. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/5d0d5594d24f0f955548f0fc0ff83d10-Paper.pdf.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

J.R. Dormand and P.J. Prince. A family of embedded runge-kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980. ISSN 0377-0427. doi: https://doi.org/10.1016/0771-050X(80)90013-3. URL https://www.sciencedirect.com/science/article/pii/0771050X80900133.

et. al. Falcon, WA. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/gutmann10a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kilian Hendrickx, Lorenzo Perini, Dries Van der Plas, Wannes Meert, and Jesse Davis. Machine learning with a reject option: A survey, 2021.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxCxhRcY7.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(91)90009-T. URL https://www.sciencedirect.com/science/article/pii/089360809190009T.

Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations, ICLR 2014*, 12 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10236–10245, 2018.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, CIFAR-10 (Canadian Institute for Advanced Research), 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Ahsan Mahmood, Junier Oliva, and Martin Andreas Styner. Multiscale score matching for out-of-distribution detection. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xoHdgbQJohv.

Alexander Meinke and Matthias Hein. Towards neural networks that provably know when they don't know. 2020.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, and Balaji Lakshminarayanan. Detecting out-of-distribution inputs to deep generative models using typicality. *arXiv preprint arXiv:1906.02994*, 2019a.

Eric T. Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Görür, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. *Proceedings of the 36th International Conference on Machine Learning*, 2019b. URL http://arxiv.org/abs/1902.02767.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

G Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192–203, 1978. ISSN 0021-9991. doi: https://doi.org/10.1016/0021-9991(78)90004-9. URL https://www.sciencedirect.com/science/article/pii/0021999178900049.

L.F. Shampine. Solving odes and ddes with residual control. *Applied Numerical Mathematics*, 52(1): 113–127, 2005. ISSN 0168-9274. doi: https://doi.org/10.1016/j.apnum.2004.07.003. URL https://www.sciencedirect.com/science/article/pii/S0168927404001187.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

W.A. Strauss. *Partial Differential Equations: An Introduction*. Wiley, 2007. ISBN 9780470054567. URL https://books.google.com/books?id=PihAPwAACAAJ.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SyxAb30cY7.

Ch. Tsitouras. Runge–kutta pairs of order 5(4) satisfying only the first column simplifying assumption. *Computers Mathematics with Applications*, 62(2):770–775, 2011. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2011.06.002. URL https://www.sciencedirect.com/science/article/pii/S0898122111004706.

Eric Veach. *Robust Monte Carlo methods for light transport simulation*. Stanford University, 1998.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training, 2020. URL https://openreview.net/forum?id=ByeL1R4FvS.

Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient GAN-based anomaly detection. In *6th International Conference on Learning Representations, ICLR 2018, Workshop Track*, 2018. URL http://arxiv.org/abs/1802.06222.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=r1Ddp1-Rb.

A PROOFS

Theorem A.1 (Additive Independent Integration)**.** *Given an additively separable function, $f(\mathbf{v})$, an independent likelihood, $p(\mathbf{v}) = \prod_{m=1}^{M} p_m(v_m)$, and domain, $\mathcal{D}_v = \mathcal{D}_{v_1} \times ... \times \mathcal{D}_{v_M}$:*

$$\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}\left[f\left(\mathbf{v}\right)\right] = \sum_{m=1}^{M} \int_{\mathcal{D}_{v_m}} f_m\left(v_m\right) p_m\left(v_m\right) dv_m \tag{A1}$$

A.1 ADDITIVELY SEPARABLE INTEGRALS

Proof.

$$\int f_n\left(v_n\right) p_m\left(v_m\right) dv_m = \begin{cases} f_n\left(v_n\right), & \text{if } n \neq m \\ \int f_n\left(v_n\right) p_n\left(v_n\right) dv_n, & \text{else} \end{cases} \tag{A2}$$

$$\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}\left[f\left(\mathbf{v}\right)\right] = \int_{\mathcal{D}} \left(\sum_{n=1}^{M} f_n\left(v_n\right)\right) \left(\prod_{m=1}^{M} p_m(v_m)\right) d\mathbf{v} \tag{A3}$$

$$= \sum_{n} \int_{\mathcal{D}} f_n\left(v_n\right) \prod_{m=1}^{M} p_m(v_m)\, d\mathbf{v} \tag{A4}$$

Substituting Eq. A1 into Eq. A3 reduces the M-dimensional integral over independent likelihoods into a 1-dimensional integral and completes the proof. □

A.2 MULTIPLICITVELY SEPARABLE INTEGRALS

Theorem A.2 (Multiplicative Independent Integration)**.** *Given a multiplicitively separable function, $g(\mathbf{v})$, an independent likelihood, $p(\mathbf{v}) = \prod_{m=1}^{M} p_m(v_m)$, and domain, $\mathcal{D}_v = \mathcal{D}_{v_1} \times ... \times \mathcal{D}_{v_M}$:*

$$\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}\left[g\left(\mathbf{v}\right)\right] = \prod_{m=1}^{M} \int_{\mathcal{D}_{v_m}} g_m\left(v_m\right) p_m\left(v_m\right) dv_m \tag{A5}$$

Proof. Since each component of g_n is matched to a component in p_m, the two products can be recombined to isolate like-dimensions. Each dimension can then be integrated independently.

$$\begin{aligned}
\mathbb{E}_{\mathbf{v}\sim p(\mathbf{v})}\left[g\left(\mathbf{v}\right)\right] &= \int d\mathbf{v} \left(\prod_{n=1}^{M} g_n(v_n)\right) \left(\prod_{m=1}^{M} p_m(v_m)\right) \\
&= \int \cdots \int \left(\prod_{n=1}^{M} g_n(v_n) p_n(v_n)\right) dv_1 \cdots dv_M \\
&= \int g_1(v_1) p_1(v_1) dv_1 \cdots \int g_M(v_M) p_M(v_M) dv_M
\end{aligned}$$

□

B ADDITIONAL EXPERIMENTS

All image datasets were trained using Glow as the bijector with the number of levels adjusted for the size of the image. MNIST and Fashion MNIST were trained with Adam for 50 epochs with an exponentially decaying learning rate of $\gamma = 0.75$. SVHN and CIFAR10 utilized a similar training procedure but with 75 epochs and a slower exponential decay of $\gamma = 0.99$.

B.1 ADVERSARIAL ROBUSTNESS

We test the efficacy of the local cross-entropy integral by utilizing a synthetic dataset from (Tsipras et al., 2019). The data is constructed via

$$y \sim \{-1,\ +1\}, \qquad x_1 = \begin{cases} +y, & \text{w.p. } p \\ -y, & \text{w.p. } 1-p \end{cases}, \qquad x_2, ..., x_{M+1} \sim \mathcal{N}(\eta y, 1) \tag{B1}$$

and the adversarial examples are constructed analytically by, essentially, flipping the sign of η. We utilize this synthetic dataset to test adversarial robustness because it provides an upper bound on the adversarial accuracy relative to the standard accuracy which allows us to quantify our performance with respect to an absolute. Additionally, because adversarial examples can be constructed exactly without relying on an optimization procedures, there can be no concerns that our defense relies on obfuscated gradients (Athalye et al., 2018). We refer the interested reader to the original paper for a thorough discussion.

We choose $M = 10$, $p = 0.95$, and $\eta = 2/\sqrt{M}$. For these choices, a strong classifier will obtain near perfect standard accuracy with zero adversarial accuracy while a robust classifier will obtain 95% standard and adversarial accuracy. We use a similar architecture to that found in Sec. C.1 except we utilize ten blocks instead of five. We train one model with only the standard cross-entropy and negative log-likelihood losses and a second model with the additional global and local cross-entropy integration loss. We find that the model with only the standard losses achieves reasonably good standard performance and fairly poor adversarial accuracy. The model with the additional integration losses contains considerably better adversarial performance, nearing the upper and, necessarily lower, standard accuracy bound. Table 3 summarizes our results and the ideal performances.

Table 3: Std. & Adv. Accuracy

	Std.	Adv.
Accurate	100.0	0.0
Robust	95.0	95.0
Baseline	97.9	27.5
Integrated	94.5	90.8

B.2 TOY SEMI-SUPERVISED REGRESSION

To illustrate the utility of global regularizations, we construct a one-dimensional, semi-supervised problem with $x \sim \mathcal{N}(0, 4)$ and $y = \tanh(x)$. We keep all values of x but remove y values corresponding to negative x during training. Unlike the standard semi-supervised problem, the missing labels are not random but are the result of limitations or bias in the data collection process. We train two models that are identical except that the integrated model includes a penalty over $\mathbb{E}[\hat{y}(\mathbf{x})]$. Table 4 shows how the models perform over the supervised and unsupervised regions. The inclusion of the integration regularizer allows the model to reason about regions that it has not directly observed and has decreased the error in that region by two orders of magnitude.

Table 4: Semi-supervised integration regularization

$\mathbb{E}[y] = 0$	Standard	Integrated
Sup. MSE	1.195e-5 $\pm$ 1.882e-5	2.487e-5 $\pm$ 1.769e-5
Unsup. MSE	1.399 $\pm$ 1.198	0.06187 $\pm$ 0.04538
$\mathbb{E}[\hat{y}]$	0.1041 $\pm$ 0.08582	8.570e-4 $\pm$ 4.150e-4

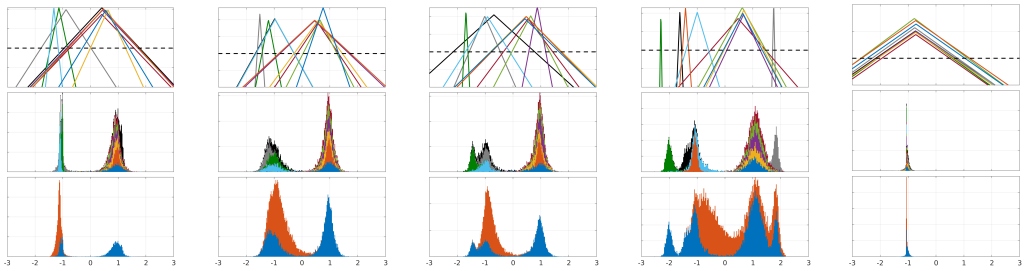
B.3 OUT OF DISTRIBUTION DETECTION AUROC COMPARISONS

B.4 INTERPRETABILITY

The separably model and independent latent dimensions allows us to reason about how the model is making decisions in the latent space. Unfortunately, for most bijectors, it is not possible to carry this interpretability back to the input space. Figure 5 demonstrates the final state of the model trained on Fashion MNIST for several features with respect to the latent distribution, per in-distribution class and juxtaposed with the out of distribution data. Specifically, the top row contains the logit components, $f_{k,m}$, learned by the separable network, color-coded by class; the middle row contains the distribution of each class (colors matched to logits); and the bottom row contains the distribution

Table 5: Area under the ROC curve (percentage)

		GAN	ODIN	MSMA	OE	CCU	Hybrid	Int. Reg.
MNIST	FMNIST	99.4	98.7	-	99.9	99.9	93.6 ± 6.5	99.8 ± 0.12
	EMNIST	92.8	88.9	-	95.8	92.0	77.7 ± 12.6	98.2 ± 0.28
FMNIST	MNIST	99.9	99.0	82.6	96.3	97.8	69.8 ± 9.2	95.4 ± 3.1
	EMNIST	99.9	99.3	-	99.3	99.5	63.4 ± 18.2	87.8 ± 4.4
SVHN	CIFAR10	96.8	95.9	97.6	100.	100.	87.0 ± 0.7	91.3 ± 0.6
CIFAR10	SVHN	83.9	96.7	99.1	98.8	98.2	73.5 ± 1.8	78.85 ± 2.7



(a) Feature 1 (b) Feature 2 (c) Feature 3 (d) Feature 4 (e) Feature 5

Figure 5: Each subplot corresponds to a single latent feature. The top and middle rows contain the unnormalized logit components and distributions per class, with matching colors. The bottom row compares the distribution of Fashion MNIST (blue) and MNIST (red). x-axis shared across all rows.

of the in-distribution data (blue) and OOD data (red). The top row illustrates how the classification network makes its decisions per feature over the distribution presented in the middle row. We see that the logit components map well to the distribution of the data in each feature and provides some intuition for how certain the classifier is over a feature. This demonstrates how this architecture allows for reasonable interpretibility from the latent space. Any value above zero (the dotted black line) is considered in-distribution and the most likely class is the line with the greatest value at that point. The features were chosen to demonstrate the diversity of the learned solution over features. Generally, the data maps reasonably well to the bimodal distribution, though we do occassionally see mode collapse as in Fig. 5e. Fortunately, in these cases the logits tend to be fairly uninformative and only introduce a small bias. Figures 5a through 5c show a common trend where the OOD data has heavy overlap with one of the two clusters but not the other. While we do see some diversity amongst the in-distribution classes that overlap with the OOD data the "bags" (gray) and "sandals" (cyan) class overlap most often. Finally, Fig. 5d demonstrates a latent feature where the OOD data falls in the region between the two data components.

B.5 MNIST Leave-one-out

Following (Ahmed & Courville, 2020), we test the robustness of our method against semantic OOD examples by training models on all but one class from MNIST (Deng, 2012) and assessing OOD performance on the held out class. We repeat the process for each class and report the AUROC and AUPR. In all cases, we train using ten different random seeds and report the average performance and standard deviation across seeds that do not degenerate against the in-distribution validation set. Table 7 and Table 6 contain the results of our method using both integrated losses compared to several relevant baselines. Results from Efficient GAN-Based Anomaly Detection (EGBAD) (Zenati et al., 2018) and GANomaly (Akcay et al., 2018) are taken by estimating the results from figures in both works. The Semantic Anomalies (Ahmed & Courville, 2020) results are obtained by executing the official version of the code using the rotation auxiliary task with two different batch sizes (128 and 256). The Semantic Anomalies results in both tables are the best across both batch sizes and detection methods (ODIN (Liang et al., 2018) and MSP (Hendrycks & Gimpel, 2017)) based on the AUPR.

We see that the Semantic Anomalies generally achieves the best AUROC across all digits. The integration often achieves the best AUPR but struggles with certain held out digits. In particular, the integration performs significantly worse on the "4" and "9" digits. This is a reasonable confuser due to the similarities and variability of the two classes. These results indicate that the integration penalties are helpful for inducing the model to detect semantic anomalies.

Table 6: MNIST leave one out: AUPR (%)

Method	0	1	2	3	4
EGBAD	78	29	67	52	46
SA: Rotation	89.24	83.57	78.20	66.33	90.45
Global & Local	95.26 ± 1.98	89.67 ± 4.33	86.02 ± 3.19	93.41 ± 1.88	69.89 ± 7.02
	5	6	7	8	9
EGBAD	43	57	35	54	35
SA: Rotation	83.38	75.57	95.19	68.84	84.88
Global & Local	87.76 ± 2.1	91.7 ± 2.84	83.43 ± 3.72	87.5 ± 2.91	72.9 ± 7.23

Table 7: MNIST leave one out: AUROC (%)

Method	0	1	2	3	4
EGBAD	78	29	67	52	45
GANomaloy	88	65	95	79	80
SA: Rotation	98.85	98.17	94.60	94.22	98.23
Global & Local	95.93 ± 1.91	89.32 ± 4.76	84.83 ± 3.94	94.25 ± 1.25	74.49 ± 6.98
	5	6	7	8	9
EGBAD	43	57	39	55	36
GANomaloy	85	85	68	85	55
SA: Rotation	97.21	92.52	99.16	92.69	97.02
Global & Local	89.47 ± 2.32	93.11 ± 2.98	83.45 ± 4.01	88.42 ± 2.97	75.66 ± 5.65

C Training and Architectures

C.1 Spirals

The bijective layer is composed of five blocks where each block contains an affine-coupling layer (Dinh et al., 2017) and an invertible fully connected layer. The data distribution is bimodal over both latent dimensions with means at ± 1 and standard deviations of 0.4. The separable network is composed of quadratic functions. We train the model as discussed, using the standard cross-entropy loss for each observed point, the negative log-likelihood over the input space, and the global cross-entropy integration with respect to the contrastive prior. The learning rate is set to 0.001 with standard weight decay of 1e-4 over 20 epochs using Adam (Kingma & Ba, 2015).

C.2 Fashion MNIST

The bijective layers are composed of the Glow (Kingma & Dhariwal, 2018) architecture with two levels composed of 32 steps and 512 channels. We utilize an open-source Glow implementation[1] wrapped in Pyro's (Bingham et al., 2018) bijective operators. The data distribution is bimodal over all latent dimensions with means at ± 1 and standard deviations of 0.5. The noise constrastive distribution is a standard Gaussian. The separable network is composed of hinge functions (see Sec. 4.3). We utilize a batch size of 256, a learning rate of 1.5e-4 over the bijective layers, and a learning rate of 1.0e-3 over the separable layers using Adam (Kingma & Ba, 2015). Both learning rates are

[1] https://github.com/chrischute/glow

exponentially decayed at a rate of 0.75 per epoch. Standard weight decay is applied with a weight of 1e-4. The network is trained for 50 epochs. The adversarial attacks are performed against the model at the epoch with the lowest validation cross-entropy loss.

D Approximate Expected Cross-Entropy over a Domain

We desire the expected cross-entropy, $\text{CE}(\hat{y}(\mathbf{x}), y_c)$, over a domain $\mathcal{D}_c$ where $\mathbf{y}_c$ is the desired probability vector over classes, $\hat{y}(\mathbf{x})$ is the model's prediction of $y \in \mathbb{R}^K$ from $\mathbf{x} \in \mathbb{R}^M$ and is composed of a bijective function, $\mathbf{h}$, a separable function, $f_k(\mathbf{x}) = \sum_m f_{k,m}(x_m)$, and the soft-max function, σ, such that $\hat{\mathbf{y}}(\mathbf{x}) = \sigma(\mathbf{f}(\mathbf{h}(\mathbf{x})))$. If we are attempting to regularize the model to produce a single class label over $\mathcal{D}_c$, then $\mathbf{y}$ will correspond to a one-hot vector. In general, however, $\mathbf{y}_c$ may be a dense vector with elements y_k. The expected cross-entropy is expressed as

$$\mathbb{E}_{\mathcal{D}_c}\left[\text{CE}(\hat{y}(\mathbf{x}), y_c)\right] = -\sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{D}_c}\left[\log(\hat{y}_k(\mathbf{x}))\right] = -\sum_{k=1}^{K} y_k \int_{\mathcal{D}_c} \log(\hat{y}_k(\mathbf{x})) \, p_{\mathcal{D}_c}(\mathbf{x}|\mathbf{y}) d\mathbf{x} \quad \text{(D1)}$$

where $p_{\mathcal{D}_c}(\mathbf{x}|\mathbf{y})$ is the probability density function over $\mathcal{D}_c$ conditioned on $\mathbf{y}$. We can take advantage of the bijective transform to convert this expression into an expectation over the latent space, $\mathbf{z}$, and latent domain, $\mathcal{Z}_c$, which allows us to write

$$\sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{D}_c}\left[\log(\hat{y}_k(\mathbf{x}))\right] = \sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{Z}_c}\left[\log(\sigma(\mathbf{f}(\mathbf{z}))\right] = \sum_{k=1}^{K} y_k \int_{\mathcal{Z}_c} \log(\sigma_k(\mathbf{f}(\mathbf{z}))) \, p_{\mathcal{Z}_c}(\mathbf{z}|\mathbf{y}) d\mathbf{z} \quad \text{(D2)}$$

where σ_k is the kth element after the soft-max normalization and $p_{\mathcal{Z}_c}(\mathbf{z}|\mathbf{y})$ is the independent probability density function in the latent space, e.g., $p_{\mathcal{Z}_c}(\mathbf{z}|\mathbf{y}) = \prod_m p_m(z_m)$. We can then expand the soft-max operator within the expectation to get

$$\sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{Z}_c}\left[\log(\sigma(\mathbf{f}(\mathbf{z}))\right] = -\,\mathbb{E}_{\mathcal{Z}_c}\left[\log\left(\sum_{j=1}^{K} \exp(f_j(\mathbf{z}))\right)\right] + \sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{Z}_c}\left[f_k(\mathbf{z})\right] \quad \text{(D3)}$$

The combination of the separable function and independent densities allows for an exact simplification of the second term into

$$\begin{aligned} \sum_{k=1}^{K} y_k \, \mathbb{E}_{\mathcal{Z}_c}\left[f_k(\mathbf{z})\right] &= \sum_{k=1}^{K} y_k \sum_{m=1}^{M} \int_{\mathcal{Z}_m} f_{k,m}(z_m) p_m(z_m) dz_m \\ &\approx \frac{1}{G} \sum_{k=1}^{K} y_k \sum_{m=1}^{M} \sum_{g=1}^{G} f_{k,m}(\tilde{z}_{m,g}) \end{aligned} \quad \text{(D4)}$$

where we have overloaded $\mathcal{Z}_m$ to correspond to the domain of the mth latent dimension and have approximated the integral via Monte Carlo integration with $\tilde{z}_{m,g}\tilde{p}_m(z_m)$ and G draws. The first term on the right-hand side of Eq. D3 does not enjoy the same exact simplification. However, it is possible to bound the term via Jensen's Inequality by moving the logarithm into the sum over j or outside the expectation operator. We consider the latter case where

$$\mathbb{E}_{\mathcal{Z}_c}\left[\log\left(\sum_{j=1}^{K} \exp(f_j(\mathbf{z}))\right)\right] \leq \log\left(\mathbb{E}_{\mathcal{Z}_c}\left[\sum_{j=1}^{K} \exp(f_j(\mathbf{z}))\right]\right). \quad \text{(D5)}$$

Then, we expand the separable function, exchange the order of the sum over dimensions and the exponent, take advantage of the multiplicative integral simplification (Eq. A5), approximate via

Monte Carlo integration, and utilize the $\log \sum \exp$ operator for stability:

$$\begin{aligned}
\log\left(\mathbb{E}_{\mathcal{Z}_c}\left[\sum_{j=1}^{K}\exp(f_j(\mathbf{z}))\right]\right) &= \log\left(\mathbb{E}_{\mathcal{Z}_c}\left[\sum_{j=1}^{K}\exp(\sum_{m=1}^{M}f_{j,m}(z_m))\right]\right) \\
&= \log\left(\sum_{j=1}^{K}\int_{\mathcal{Z}_c}d\mathbf{z}\prod_{m=1}^{M}\exp(f_{j,m}(z_m))\prod_{n=1}^{M}p_n(z_n)\right) \\
&= \log\left(\sum_{j=1}^{K}\prod_{m=1}^{M}\int_{\mathcal{Z}_m}\exp(f_{j,m}(z_m))p_m(z_m)dz_m\right) \\
&\approx \log\left(\sum_{j=1}^{K}\prod_{m=1}^{M}\sum_{g=1}^{G}\exp(f_{j,m}(\tilde{z}_{m,g}))\right) - M\log(G) \\
&= \log\sum_{j=1}^{K}\exp\sum_{m=1}^{M}\log\sum_{g=1}^{G}\exp(f_{j,m}(\tilde{z}_{m,g})) - M\log(G).
\end{aligned} \tag{D6}$$

Finally, we can substitute Eq. D4 and Eq. D6 into Eq. D1 to get

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_c}\left[\text{CE}(\hat{y}(\mathbf{x}), y_c)\right] \leq &-\sum_{k=1}^{K}y_k\sum_{m=1}^{M}\int_{\mathcal{Z}_m}f_{k,m}(z_m)p_m(z_m)dz_m \\
&+\log\left(\sum_{j=1}^{K}\prod_{m=1}^{M}\int_{\mathcal{Z}_m}\exp(f_{j,m}(z_m))p_m(z_m)dz_m\right) \\
\approx &-\frac{1}{G}\sum_{k=1}^{K}y_k\sum_{m=1}^{M}\sum_{g=1}^{G}f_{k,m}(\tilde{z}_{m,g}) \\
&+\log\sum_{j=1}^{K}\exp\sum_{m=1}^{M}\log\sum_{g=1}^{G}\exp(f_{j,m}(\tilde{z}_{m,g})) - M\log(G)
\end{aligned} \tag{D7}$$

In the event that y_c is a one-hot vector (at the cth element) this simplifies to

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_c}\left[\text{CE}(\hat{y}(\mathbf{x}), y_c)\right] \leq &-\sum_{m=1}^{M}\int_{\mathcal{Z}_m}f_{c,m}(z_m)p_m(z_m|c)dz_m \\
&+\log\left(\sum_{j=1}^{K}\prod_{m=1}^{M}\int_{\mathcal{Z}_m}\exp(f_{j,m}(z_m))p_m(z_m|c)dz_m\right)
\end{aligned} \tag{D8}$$

where we reintroduced the condition over the class within the probability densities $p_m(z_m|c)$ for emphasis.