Out of Distribution Data Detection Using Dropout Bayesian Neural Networks

Andre T. Nguyen,[1,2,3] **Fred Lu,**[1,2,3] **Gary Lopez Munoz,**[1,2] **Edward Raff,**[1,2,3] **Charles Nicholas,**[3] **James Holt**[1]

[1]Laboratory for Physical Sciences
[2]Booz Allen Hamilton
[3]University of Maryland, Baltimore County
andre@lps.umd.edu, lu_fred@bah.com, dlmgary@lps.umd.edu, edraff@lps.umd.edu, nicholas@umbc.edu, holt@lps.umd.edu

Abstract

We explore the utility of information contained within a dropout based Bayesian neural network (BNN) for the task of detecting out of distribution (OOD) data. We first show how previous attempts to leverage the randomized embeddings induced by the intermediate layers of a dropout BNN can fail due to the distance metric used. We introduce an alternative approach to measuring embedding uncertainty, justify its use theoretically, and demonstrate how incorporating embedding uncertainty improves OOD data identification across three tasks: image classification, language classification, and malware detection.

1 Introduction

Detecting out of distribution (OOD) data at test time is critical in a variety of machine learning applications. For example, in the context of malware classification (Raff and Nicholas 2020), OOD data could correspond to the emergence of a new form of malicious attack. Gal and Ghahramani (2016b) developed an approach to variational inference in Bayesian neural networks (BNNs) that showed a neural network with dropout (Hinton et al. 2012; Srivastava et al. 2014), a technique commonly used to reduce overfitting in neural networks (NNs) by randomly dropping units during training, applied before every weight layer is equivalent to an approximation of a deep Gaussian process (Damianou and Lawrence 2013). Training with dropout effectively performs variational inference for the deep Gaussian process model, and the posterior distribution can be sampled from by leaving dropout on at test time. This approach to Bayesian deep learning has been popular in practice as it is easy to implement and scales well.

Measures of uncertainty usually are a function of the sampled softmax outputs of such a BNN, for example predictive entropy and mutual information. There is however useful information at every intermediate layer of a dropout BNN. The dropout based approach to Bayesian deep learning suffers, like most variational inference methods, from the tendency to fit an approximation to a local mode instead of to the full posterior because of a lack of representational capacity and because of the directionality of the KL divergence (Smith and Gal 2018; Wilson and Izmailov 2020). This behavior however allows us to expect the randomized intermediate representation samples in a dropout BNN to be meaningfully related as they are sampled from a local mode. In this paper, we explore how to leverage additional information generated at every layer of the network for the task of OOD data detection at test time. In particular, we interpret the intermediate representation of a data point at a particular layer as a randomized embedding. The embedding is randomized due to the use of dropout at test time.

The idea to use a randomized embedding induced by the intermediate layers of a dropout BNN has been attempted previously, but can fail due to the underlying Euclidean distance metric used in previous work. The use of Euclidean distance does not account for the confounding variability caused by changes in embedding magnitudes. We will theoretically justify and empirically show that by instead using a measure based on cosine distance, this problem can be rectified. We then leverage this improved uncertainty estimation to show better OOD data identification across three highly different tasks to demonstrate the robustness of our approach.

The objective of this paper is not to develop a state-of-the-art approach to OOD data detection, but rather in the context of dropout BNNs to: (1) show how to cheaply improve OOD data detection in systems where a dropout BNN is already deployed, by using intermediate computational results that are already being computed but not fully leveraged, and (2) provide theoretical and practical evidence to highlight why it is valuable to deconflate angular information about embedding dispersion from embedding norm information. Additionally, previous works have evaluated OOD detection by assuming access to a large OOD dataset of similar size to the in distribution dataset. This is an unrealistic assumption as in areas like cyber security where OOD examples are limited and expensive. So, we also examine the effect of small dataset sizes for OOD detection in our experiments.

2 Related Work

Two kinds of uncertainty can be distinguished (Kendall and Gal 2017). Aleatoric uncertainty is caused by inherent noise and stochasticity in the data. More training data will not help to reduce this kind of uncertainty. Epistemic uncertainty on the other hand is caused by a lack

of similar training data. In regions lacking training data, different model parameter settings that produce diverse or potentially conflicting predictions can be comparably likely under the posterior. OOD data is expected to have higher uncertainty, epistemic in particular. Mukhoti et al. (2021) prove that one cannot infer epistemic uncertainty from a deterministic model's softmax entropy, so additional information is needed to estimate epistemic uncertainty.

Uncertainty modeling using probabilistic embeddings has primarily been used for estimating aleatoric uncertainty (Oh et al. 2018; Shi and Jain 2019; Chun et al. 2021; Chang et al. 2020) in tasks such as determining the quality of a test input image. These methods do not easily translate to estimating epistemic uncertainty. For example, Oh et al. (2018) try to apply their method on an epistemic uncertainty estimation task and find that it did not work well for novel classes, and they leave the modeling of epistemic uncertainty as future work.

The only prior work we are aware of that looks at a randomized embedding approach similar to ours is by Terhörst et al. (2020), who use dropout at test time to generate a stochastic embedding. They estimate face image quality through the stability of the embedding as measured using Euclidean distance. As we will show, the use of Euclidean distance is problematic as it does not account for factors affecting embedding norms and more generally, the assumptions made by Terhörst et al. (2020) are not met in reality. We also note that they are actually estimating epistemic uncertainty (see (Oh et al. 2018) for an explanation) when test image quality is an inherently aleatoric uncertainty estimation problem. We will show both empirical evidence as well as mathematical grounding as to why our proposed approach, without the addition of any complexity, fixes these issues.

There is evidence that intermediate layers of a neural network contain information useful for epistemic uncertainty estimation and out of distribution detection. Postels et al. (2020) establish a connection between the density of hidden representations and the information-theoretic surprise of observing a specific sample in the setting of a deterministic neural network. In particular, they suggest that the first layers of a neural network should be used to estimate epistemic uncertainty due to feature collapse, a phenomena where out-of-distribution data is mapped to in-distribution feature representations in later layers of a network (van Amersfoort et al. 2020; Mukhoti et al. 2021), though they also suggest that OOD data detection can benefit from aggregating uncertainty information from several layers. Our work differs from their work as we are not fitting a density to representations of the training data, increasing the applicability of our approach to situations where fitting and storing a density is not an option for computational or regulatory reasons.

Other recent work has also looked at uncertainty estimation using a single forward pass of a neural network that has had its intermediate representations regularized to produce good uncertainty estimates (van Amersfoort et al. 2020; Liu et al. 2020). We note that many single forward pass based methods like (Mukhoti et al. 2021; Liu et al. 2020) require residual based networks in combination with spectral normalization to enforce a bi-Lipschitz inductive bias (Bartlett, Evans, and Long 2018). While the method of (van Amersfoort et al. 2020) is not residual network constrained, it requires significant changes to the model and training procedure. While our approach requires multiple forward passes (as is the case with all dropout BNNs), it does not require any modifications to existing dropout BNNs, by only using information that is already being computed within a dropout BNN.

(Mandelbaum and Weinshall 2017) propose a confidence score that uses a data embedding derived from the penultimate layer of a neural network. The embedding is achieved using either a distance-based loss or adversarial training. Similarly to other methods, this method requires density estimation, and our work differs as our method does not involve a comparison to nearest neighbors from the training set, which may be difficult to deploy in practice due to both storage and regulatory constraints.

Many works have investigated OOD data detection in probabilistic contexts. Ovadia et al. (2019) benchmarks Bayesian deep learning methods in the context of dataset shift and OOD data at test time. Xiao, Gomez, and Gal (2020) use epistemic uncertainty to detect OOD language data. Ren et al. (2019) detect OOD data using likelihood ratios in the context of deep generative models and evaluate on OOD genomic sequences. Our work makes a contribution to probabilistic OOD identification by being the first work to systematically investigate the appropriate use of the randomized embeddings induced by the intermediate layers of a dropout BNN.

3 Methods

In a supervised setting, suppose a neural network structure with N (non-linearity included) layers $f_i, i \in [1, N]$ where x_1 is the input and x_{N+1} is the prediction: $x_{i+1} = f_i(x_i)$. Gal and Ghahramani (2016b) showed that a neural network with dropout (Hinton et al. 2012; Srivastava et al. 2014) applied before every weight layer is equivalent to an approximation of a deep Gaussian process (Damianou and Lawrence 2013), and that training with dropout effectively performs variational inference for the deep Gaussian process model. At test time, the posterior distribution can be sampled from by leaving dropout on. This gives us the network structure:

$$x_{i+1} = f_i(\text{dropout}(x_i)) \tag{1}$$

3.1 Randomized Embeddings

Computing an Embedding In the context of a trained dropout Bayesian neural network, we can use the intermediate representations from the various layers (the x_{i+1} in Equation 1) as a randomized embedding of a data point. The embedding is randomized as multiple forward passes with dropout on will yield different embedding values. The variation in the embedding values could be used to measure epistemic uncertainty (Oh et al. 2018), allowing for the detection of OOD data and dataset shift.

Algorithm 1: Computing Randomized Embedding Based Features for OOD Data Detection

Input: A datum x, a N layer NN trained with dropout $\{f_1, ..., f_N\}$, and number of samples T.

Output: N randomized embedding based features $z_1, ..., z_N$, each corresponding to a layer in the network, for a OOD data detection task.

for $t \leftarrow$ *1 to T* **do**
  for $i \leftarrow$ *1 to N* **do**
    $x_{i+1,t} \leftarrow f_i(\text{dropout}(x_{i,t}))$
for $i \leftarrow$ *1 to N* **do**
  $z_i \leftarrow \max(\text{PairwiseCosineDistances}(x_{i,:}))$
return $z_1, ..., z_N$ `// Return features.`

Measuring Uncertainty A datum is embedded to a set of randomized embedding values at each layer. We can compute the maximum pairwise distance between the embeddings for a specific datum at a specific layer. This can be done at each layer in the BNN, giving us a feature for each layer that can then be used for tasks such as OOD identification. *All previous work has used Euclidean distance to compute the pairwise distances, without examining the appropriateness of Euclidean distance for the task*. Part of our contribution is an analysis in section 3.3 of why Euclidean distance is in fact not appropriate, and we introduce a preferable cosine distance based approach which we use in all of our experiments. A small value of 1e-6 was added to the embeddings to avoid numerical issues caused by corner-case zero normed embedding vectors.[1] In our experiments, embeddings from non-linear layers (such as convolutions) are flattened prior to computing this metric. A summary of our approach can be found in algorithm 1. The intuition behind this approach is that if measured appropriately, the "spread" or maximal variation in a datum's embedding contains uncertainty information. If all embedding samples are realized to a same point in the embedding space, then there is less uncertainty than if the embedding samples are realized to wildly different parts of the embedding space.

3.2 Baseline Features

We compare the addition of our randomized embedding based features to a set of common baseline features. For classification tasks, uncertainty estimates in dropout BNNs are usually a function of the sampled softmax outputs. In particular, overall uncertainty can be measured using predictive distribution entropy: $H[\mathbb{P}(y|x, D)] = -\sum_{y \in C} \mathbb{P}(y|x, D) \log \mathbb{P}(y|x, D)$. To isolate and measure epistemic uncertainty mutual information can be used: $I(\theta, y|D, x) = H[\mathbb{P}(y|x, D)] - \mathbb{E}_{\mathbb{P}(\theta|D)} H[\mathbb{P}(y|x, \theta)]$.

The terms of these equations can be approximated using Monte Carlo estimates obtained by sampling from the dropout BNN posterior (Smith and Gal 2018). In particular, $\mathbb{P}(y|x, D) \approx \frac{1}{T} \sum_{i=1}^{T} \mathbb{P}(y|x, \theta_i)$ and $\mathbb{E}_{\mathbb{P}(\theta|D)} H[\mathbb{P}(y|x, \theta)] \approx \frac{1}{T} \sum_{i=1}^{T} H[\mathbb{P}(y|x, \theta_i)]$ where the θ_i are samples from the posterior over models and T is the number of samples. In addition to predictive distribution entropy and mutual information, we also use maximum softmax probability (the value of the largest element of $\mathbb{P}(y|x, D)$) as a feature, shown by Hendrycks and Gimpel (2016) to be an effective baseline for the OOD data detection task.

3.3 How to Measure Embedding Dispersion

We will now explore why Euclidean distance as used by previous works is not appropriate to measure randomized embedding dispersion. We illustrate using a LeNet5 (Yann LeCun et al. 1998) model with added dropout before each layer trained on MNIST, with MNIST variants as OOD data. Further data, model, and experimental details correspond to those expanded upon in subsection 4.1.

The Problem With Euclidean Distance Terhörst et al. (2020) suggest the Euclidean distance to measure when a data point is suitable for a downstream task, where lower variability in the stochastic embedding induced by a dropout neural network suggests higher suitability for a data point. In particular, they use the sigmoid of the negative mean Euclidean distance between all stochastic embedding pairs for a data point as the measure of suitability. In other words, their hypothesis is that a form of uncertainty can be measured using the Euclidean distance between embedding samples.

We find that if Euclidean distance is used as the metric to measure distance between samples, their hypothesis holds only with excessive training and likely over-fitting. Figure 1a shows that with enough training to get to the accuracy plateau (10 epochs of training with a batch size of 64, with a test accuracy of 0.9885), we actually see the opposite effect. Embeddings for OOD data are actually less spread out than embeddings for in distribution data. Figure 1b shows that with excessive training (100 epochs of training, with a lower test accuracy of 0.9882), we see that the hypothesis holds better but note that there is still a good amount of overlap between the histograms, limiting the usefulness for OOD detection (and adding a difficult to select stopping criteria). We note that what we are observing is *not* feature collapse.

This points to two issues that we need to resolve. First, how can we get consistent behavior regardless of over/under-training? Second, how can we more usefully measure spread in a way that matches intuition?

Spectral Normalization Stabilizes Behavior Spectral normalization rescales the weights during training with the spectral norm of the weight matrix, enforcing a Lipschitz constraint that bounds the derivative of the learned function (Miyato et al. 2018). This helps to preserve distance as a data point makes its way through the network. Figure 1c shows that a spectral normalized version of the network results in consistent behavior even with longer training (100 epochs of training, with a test accuracy of 0.9927). So, there is a

[1] We also note that normalized Euclidean distance, where embedding vectors are normalized to unit length prior to computing Euclidean distance, could also be used in place of cosine distance as its square can be shown to be proportional to cosine distance.

solution to the first problem. However, we still see that the spread for OOD data is lower than for in distribution data.

Why Cosine Distance Is Needed To Properly Measure Embedding Dispersion Previous research around OOD detection has noted that a lower maximal softmax output value is correlated with a data point being OOD (Hendrycks and Gimpel 2016). One possible explanation could be logits (softmax inputs) of smaller norm. This would make intuitive sense as potentially, less neurons would activate for OOD data since OOD data would lack the in distribution features the network is looking for.

The squared Euclidean distance between vectors $\mathbf{u}$ and $\mathbf{v}$ can be written as, where θ is the angle between $\mathbf{u}$ and $\mathbf{v}$:

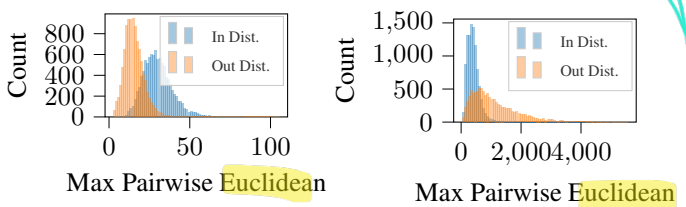
$$||\mathbf{u}-\mathbf{v}||^2 = ||\mathbf{u}||^2 + ||\mathbf{v}||^2 - 2\,||\mathbf{u}||\,||\mathbf{v}||\,\cos\theta \quad (2)$$

If embedding norms are inherently smaller for OOD data, then Euclidean distance which is norm dependent cannot be used to compare embedding spread across OOD and in distribution datasets, due to confounding. As shown in Equation 2, angular information is affected by norm in both an additive and multiplicative manner with Euclidean distance. So, assuming confounding caused by systematic norm differences, cosine distance should be used to isolate the angular information when measuring embedding dispersion. If Euclidean distance mostly captures information already captured by the norm, then the benefit of being Bayesian for this task is not fully leveraged as norm can be estimated with a single point estimate. *To take full advantage of a dropout BNN, angular information about embedding dispersion needs to be deconflated from embedding norm information.*

We explored this hypothesis and found it to be empirically true and formally justifiable. In Figure 2a, Euclidean distance is used to measure embedding dispersion, we see that dispersion is correlated with the logits norm and that the relationship is nearly identical for OOD and in distribution data. This means that measuring the spread of the embeddings using Euclidean distance conveys little extra information than just looking at the norm of the logits. In Appendix section A.4, we perform a simulation to further illustrate this problem in the case of a two layer ReLU activated network.

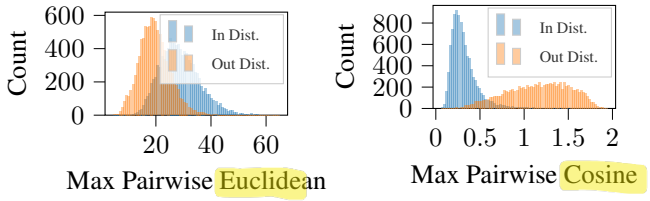
We want to measure spread in a way that is independent of the embedding norm. This can be done a couple of different ways. For example, a simple switch to cosine distance could be used, or the embeddings could be normalized prior to using Euclidean distance (which can be shown to be related to cosine distance). As illustrated in Figure 2b, using cosine distance results in OOD and in distribution data having behaviors that are no longer identical. Appendix subsection A.3 shows similar results in an unsupervised setting involving a stacked denoising autoencoder variant.

Figure 1d shows the same information as Figure 1a, except a cosine distance based measure of spread is used instead of a Euclidean based one. With cosine distance, we now see the expected behavior of OOD having more spread than in distribution, and we see a better separation as well which is good for OOD detection. We have shown results for the last layer of a network but note that a similar analysis can



(a) LeNet5 with 10 epochs of training, Euclidean based measure of embedding dispersion.

(b) LeNet5 with 100 epochs of training, Euclidean based measure of embedding dispersion.



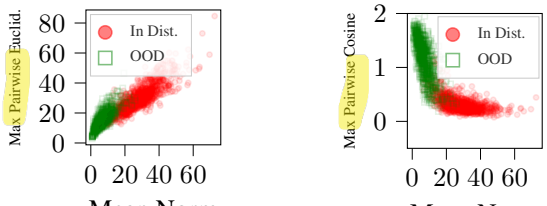
(c) Spectral Normalized LeNet5 with 100 epochs of training, Euclidean measure of dispersion.

(d) LeNet5 with 10 epochs of training, cosine based measure of embedding dispersion.

Figure 1: Comparison of last layer randomized embedding dispersion distributions for in distribution data (MNIST) and OOD data (Not-MNIST).

be done for each layer. Having shown empirical evidence for why angular information needs to be isolated from norm information when measuring embedding dispersion, we next provide a formal analysis for why cosine distance allows for an additional source of information.

Formal Analysis of Cosine Embedding Dispersion We aim to compute a metric that is invariant to the relative magnitudes among embedding samples, and also accurately represents the dispersion of the embedding samples. In the following, we argue that the mutual information score is not satisfactory for these two objectives. Our goal is not to replace the mutual information as an uncertainty measure, but rather to demonstrate that our pairwise cosine similarity yields an



(a) Euclidean distance.

(b) Cosine distance.

Figure 2: A comparison of the relationships between last layer randomized embedding mean norm and the maximum pairwise distance for Euclidean and cosine distances respectively, for in distribution data (MNIST) and OOD data (Not-MNIST). Both models using LeNet5 trained for 10 epochs.

additional source of information that is not captured otherwise.

Let $\{z_i\}_{i=1}^m$ denote m embedding vectors sampled through dropout. The mutual information score is defined as

$$I(w, y|D, x) = H[p(y|x, D)] - \mathbb{E}_{p(w|D)}H[p(y|x, w)]$$

and is approximated by

$$\hat{I}(w, y|D, x) = H\left[\frac{1}{m}\sum_{i=1}^{m}\text{softmax}(z_i)\right] - \frac{1}{m}\sum_{i=1}^{m}H\left[\text{softmax}(z_i)\right]$$

where $H(\cdot)$ is the entropy function $H(y) = -\sum_i y_i \log y_i$.

We first introduce a theorem from Amos (2019) that clarifies the geometric properties of the softmax function. The proof is readily shown using Lagrange multipliers.

Theorem 3.1. *The softmax function* $\text{softmax}(x)_j = \frac{\exp(x_j)}{\sum_i \exp(x_i)}$ *is a map from* $\mathbb{R}^d$ *to the* $(d-1)$*-simplex that satisfies*

$$\text{softmax}(x) = \underset{0<y<1}{\arg\min} -x^\top y - H(y) \;\text{ s.t. }\; 1^\top y = 1$$

From this we see that the softmax solution is a balance between two competing objectives: maximizing $x^\top y$ which aims to place all weight on the coordinate with the largest x_i value, and maximizing the entropy of y which steers toward the uniform vector with value $1/d$. In addition, the softmax temperature changes the relative weighting, which allows us to evaluate the effect of the magnitude of the embedding vector. We leverage this for a further Lemma and Theorem:

Lemma 3.2. *The softmax function with temperature* α*, defined by* $\text{softmax}(x/\alpha)$*, satisfies*

$$\text{softmax}(x/\alpha) = \underset{0<y<1}{\arg\max}\, x^\top y + \alpha H(y) \;\text{ s.t. }\; 1^\top y = 1$$

Proof. From the previous theorem we get $\text{softmax}(x/\alpha) = \arg\min_{0<y<1} -(x/\alpha)^\top y - H(y) \;\text{ s.t. }\; 1^\top y = 1$. Multiplying by scalar α and switching the optimization to maximizing the negative does not change the optimal solution, yielding the statement above. □

These facts help indicate that softmax-based metrics are not suited for assessing the angular dispersion among vectors. We note that the mapped vector is α-dependent and hence dependent on the L_2 magnitude of the input vector. Furthermore, arbitrary translations of the vector, which can completely change the direction of the vector, do not impact the softmax. These observations are formalized below.

Theorem 3.3. *The softmax function is invariant to translation of input vector* x*. It is not invariant to scaling* x *except in the special case when* $x_1 = x_2 = \ldots = x_d$*. Furthermore, as the magnitude of* x *increases (without changing direction), the softmax shifts weight to the vertex of the simplex corresponding to the largest coordinate in* x*.*

Proof. Invariance to translation follows from observing that $\text{softmax}(x + K) = \exp(x_j + K)/\sum_i \exp(x_j + K) = \exp(x_j)/\sum_i \exp(x_j) = \text{softmax}(x)$.

The dependence on scaling follows from Lemma 1.2. Consider two vectors x, x' such that $x' = x/\alpha$. The value of α adjusts the scale of the $H(y)$ term. Since the $\max x^\top y$ objective aims to shift weight in y to the largest x coordinate and the $\max H(y)$ objective aims to distribute weight evenly, their solutions do not coincide, giving $\text{softmax}(x)$ and $\text{softmax}(x')$ different solutions. In the special case that $x_1 = x_2 = \ldots = x_d$ then $x^\top y$ is constant, so the optimization of $H(y)$ gives the uniform distribution vector. Otherwise, increasing the magnitude of x' is equivalent to sending $\alpha \to 0$, which decreases the contribution of $H(y)$. This causes the solution vector to shift weight to the element with largest value in x. □

We confirm this analysis by simulation in Appendix section A.4, where we find that our new cosine-based feature adds an orthogonal measure of information that is not captured in previously used measures of uncertainty.

4 Experiments and Results

In this section, we evaluate the value of randomized embedding based features across three different OOD data detection tasks in the vision, language, and malware domains. All experiments were implemented in PyTorch (Paszke et al. 2019), and neural networks were optimized using Adam with the default recommended settings (Kingma and Ba 2014). A dropout probability of $p = 0.1$ was used, and when sampling from the base neural network models to compute features for OOD detection, 32 samples are used. Experiments were run on an 80 CPU core machine with 512GB of RAM using a single 16GB Tesla P100 GPU. Experiment specific details are described in their respective sections.

We explore the use of two model classes for the OOD detection algorithms. The first model is an L2-regularized logistic regression (LR) with the regularization strength chosen using 3-fold cross-validation. We min-max scaled the input features for the LR model to the range $[0, 1]$ based on the training data. The second model is a 500 tree random forest (RF) classifier. We choose these two models to assess linear vs. non-linear behavior in the OOD detection task. We also explore the effect of varied, small training set sizes for the OOD task in all of our experiments. In many production contexts such as cyber security, examples of OOD data are limited and usually expensive to obtain.

4.1 Image Classification

For our vision experiments, similarly to the evaluation protocol from (van Amersfoort et al. 2020; Ren et al. 2019; Postels et al. 2020; Mukhoti et al. 2021) we explore MNIST variants as OOD data. In particular, we train our base model, a LeNet5 (Yann LeCun et al. 1998) with added dropout before each layer, on MNIST and use Kuzushiji-MNIST (Clanuwat et al. 2018), notMNIST (Bulatov 2011), and Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) as OOD data. When training the downstream OOD data detection algorithms, we train the OOD detector on one of the OOD datasets and test on the other two. For example, we first train a digit classifier on MNIST. Then, we train an OOD data detector that uses randomized embedding based features from

the digit classifier to classify MNIST vs. notMNIST. Then we test the OOD data detector on MNIST vs. Kuzushiji-MNIST and Fashion-MNIST.

Due to its importance in practical use, we will test the sample efficiency of the OOD tasks (i.e., how few samples of OOD are needed to detect future OOD data). In particular, we evaluate performance, as measured by area under the receiver operating characteristic curve (captures desired data ordering performance) and accuracy (captures desired decision making value), using training datasets consisting of n=1000, 100, and just 10 data points from each class (in distribution and OOD). We note that this differs from most previous works which have evaluated by assuming access to a large OOD dataset of similar size to the in distribution dataset, an often unrealistic assumption. Each experiment was run 100 times with random training set samples, where all appropriate data not in the training set is included in the test set, and we report a mean and standard deviation for each. In all of our experiments, the standard deviations are much smaller than effect sizes, so we report only the means in this section, and standard deviations can be found in appendix subsection A.1.

Detecting OOD Data Table 1 compares performance with and without the cosine embedding spread features for various experimental configurations and OOD detection models for a dropout LeNet5 trained for 100 epochs. Features labeled as "Last" consist of common baseline features computed using softmax output samples from the network (predictive entropy, mutual information, and maximum softmax probability). Features labeled as "Last+Spread" consist of these baseline features plus our additional randomized embedding maximum cosine spread features for each layer.

The inclusion of the additional cosine spread features improves OOD detection performance consistently across datasets, training set sizes, and model types. In limited cases where the "Spread" features do not improve the LR model, the RF model with "Spread" features performs the best overall, suggesting that the relationship is not necessarily linear. Table 7 in the Appendix summarizes results from a similar experiment where the base model is a spectral normalized dropout LeNet5 trained for 100 epochs. A comparison of Table 1 and Appendix Table 7 suggests that, while spectral normalization is not required to see an improvement from the inclusion of cosine spread features, spectral normalization does improve OOD detection performance consistently.

In Appendix subsection A.5, we further examine the need for a small amount of OOD training data, evaluate Euclidean based spread features, and investigate the feature importances associated with our cosine spread features.

Table 1: Performance with and without the cosine randomized embedding spread features for various experimental configurations for a dropout LeNet5 trained on MNIST. Features labeled as "Last" consist of common baseline features computed using softmax output samples from the network (predictive entropy, mutual information, and maximum softmax probability). Features labeled as "Last+Spread" consist of these baseline features plus our additional randomized embedding maximum cosine spread features for each layer. Each experiment was repeated multiple times, and the mean is reported here while the standard deviation is reported in Appendix A. Best results are shown in **bold**.

OOD			Num/Class	n=1000		n=100		n=10	
			Metric	AUC	Acc	AUC	Acc	AUC	Acc
Train	Test	Model	Features						
Fashion	Kuzushiji	LR	Last	0.969	**0.914**	0.967	0.909	0.963	0.884
			Last+Spread	**0.979**	0.914	**0.973**	**0.911**	**0.967**	**0.901**
		RF	Last	0.960	0.917	0.952	0.905	0.942	0.884
			Last+Spread	**0.979**	**0.922**	**0.974**	**0.921**	**0.969**	**0.907**
	notMNIST	LR	Last	0.966	0.912	0.965	0.909	0.960	0.879
			Last+Spread	**0.983**	**0.932**	**0.979**	**0.925**	**0.967**	**0.892**
		RF	Last	0.959	0.920	0.950	0.903	0.938	0.880
			Last+Spread	**0.985**	**0.940**	**0.976**	**0.924**	**0.963**	**0.901**
Kuzushiji	Fashion	LR	Last	0.973	0.920	0.972	0.917	0.967	0.899
			Last+Spread	**0.989**	**0.948**	**0.983**	**0.937**	**0.978**	**0.922**
		RF	Last	0.964	0.920	0.956	0.907	0.946	0.896
			Last+Spread	**0.986**	**0.943**	**0.978**	**0.931**	**0.967**	**0.914**
	notMNIST	LR	Last	0.967	0.914	0.965	0.910	0.960	0.886
			Last+Spread	**0.984**	**0.931**	**0.975**	**0.914**	**0.966**	**0.888**
		RF	Last	0.960	0.922	0.950	0.904	0.938	0.888
			Last+Spread	**0.982**	**0.935**	**0.971**	**0.921**	**0.954**	**0.896**
notMNIST	Fashion	LR	Last	0.966	0.911	0.957	0.906	0.959	0.893
			Last+Spread	**0.978**	**0.937**	**0.969**	**0.928**	**0.977**	**0.925**
		RF	Last	0.960	0.910	0.955	0.904	0.946	0.887
			Last+Spread	**0.988**	**0.943**	**0.982**	**0.935**	**0.978**	**0.920**
	Kuzushiji	LR	Last	0.960	**0.900**	0.946	**0.893**	0.951	0.882
			Last+Spread	**0.966**	0.893	**0.949**	0.886	**0.967**	**0.902**
		RF	Last	0.956	**0.906**	0.950	0.901	0.941	0.883
			Last+Spread	**0.978**	0.906	**0.973**	**0.915**	**0.969**	**0.906**

4.2 Language Classification

Out of distribution data detection is also of interest in natural language processing, where systems are trained to work on specific languages, and inputs from other languages are considered OOD (Xiao, Gomez, and Gal 2020). For these experiments, we train a Char-CNN (Zhang, Zhao, and LeCun 2016) with dropout added before every layer to classify languages using the WiLI dataset (Thoma 2018). Training consisted of 50 epochs with a batch size of 128, where the 100 most common characters in the training set (after stripping accents) were used as the vocabulary and each datum was truncated/padded to a length of 200 characters. We train the language classification model to distinguish between French, Spanish, German, English, Italian, and Portuguese text. We use Basque, Polish, Luganda, Finnish, Tongan, and Xhosa as out of distribution languages. All of our in and out of distribution languages are chosen to use the Latin writing system. For the OOD task, training sets consisted of n=100, 50, 25, and 10 data points from each class (in distribution and OOD). Each experiment was run 100 times with random training data subsamples, where all languages not trained on are tested on. Table 2 shows that the inclusion of our randomized embedding based features consistently improves OOD detection across experimental settings, with average and maximal AUC improvements of 0.06 and 0.15.

We note that while OOD data detection is usually treated as a purely binary classification task by most previous work, OOD versus in distribution is a false binary. There are different levels and degrees of how OOD data can be. In the context of language, we can examine the nuances between different flavors of OOD data. While Basque is a language isolate that linguistically does not share any significant similarities to any other languages, Catalan is a Romance language with

Table 2: Performance with and without the cosine randomized embedding spread features for a Char-CNN with dropout added before every layer trained to classify languages using the WiLI dataset. Standard deviations are reported in Appendix A, and best results are shown in **bold**.

OOD			Num/Class	n=100		n=50		n=25		n=10	
Train	Test	Model	Metric / Features	AUC	Acc	AUC	Acc	AUC	Acc	AUC	Acc
Basque	rest	LR	Last	0.888	0.798	0.883	0.794	0.882	0.792	0.878	0.786
			Last+Spread	**0.926**	**0.843**	**0.919**	**0.836**	**0.921**	**0.835**	**0.926**	**0.828**
		RF	Last	0.862	0.797	0.857	0.793	0.851	0.792	0.842	0.789
			Last+Spread	**0.924**	**0.845**	**0.920**	**0.840**	**0.918**	**0.835**	**0.914**	**0.824**
Finnish	rest	LR	Last	0.888	0.795	0.885	0.792	0.881	0.790	0.883	0.786
			Last+Spread	**0.910**	**0.818**	**0.908**	**0.818**	**0.909**	**0.821**	**0.910**	**0.818**
		RF	Last	0.864	0.794	0.858	0.792	0.850	0.789	0.840	0.783
			Last+Spread	**0.913**	**0.821**	**0.907**	**0.821**	**0.905**	**0.818**	**0.904**	**0.814**
Luganda	rest	LR	Last	0.891	0.806	0.889	0.803	0.887	0.800	0.881	0.794
			Last+Spread	**0.943**	**0.864**	**0.939**	**0.854**	**0.935**	**0.847**	**0.931**	**0.837**
		RF	Last	0.866	0.800	0.859	0.797	0.854	0.796	0.843	0.785
			Last+Spread	**0.936**	**0.862**	**0.930**	**0.852**	**0.926**	**0.843**	**0.921**	**0.831**
Polish	rest	LR	Last	0.900	0.824	0.897	0.821	0.891	0.816	0.887	0.812
			Last+Spread	**0.939**	**0.866**	**0.938**	**0.864**	**0.935**	**0.860**	**0.934**	**0.852**
		RF	Last	0.870	0.793	0.860	0.787	0.854	0.783	0.850	0.780
			Last+Spread	**0.937**	**0.871**	**0.932**	**0.863**	**0.928**	**0.855**	**0.922**	**0.841**
Tongan	rest	LR	Last	0.857	**0.815**	0.841	**0.811**	0.815	0.800	0.791	0.771
			Last+Spread	**0.886**	0.811	**0.877**	0.810	**0.884**	**0.819**	**0.880**	**0.813**
		RF	Last	0.765	0.699	0.766	0.695	0.769	0.684	0.785	0.701
			Last+Spread	**0.915**	**0.847**	**0.913**	**0.845**	**0.906**	**0.836**	**0.903**	**0.823**
Xhosa	rest	LR	Last	0.894	0.807	0.891	0.804	0.886	0.800	0.879	0.794
			Last+Spread	**0.944**	**0.866**	**0.940**	**0.857**	**0.933**	**0.846**	**0.931**	**0.838**
		RF	Last	0.864	0.787	0.857	0.782	0.849	0.778	0.846	0.774
			Last+Spread	**0.939**	**0.868**	**0.934**	**0.860**	**0.928**	**0.852**	**0.921**	**0.835**

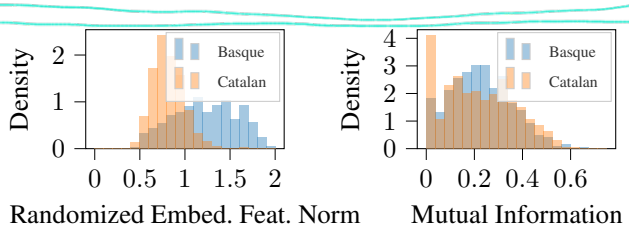


Figure 3: Basque and Catalan are linguistically similar but different languages. Our cosine based embeddings (left) show that they have high overlap but are more OOD than normal data. Prior work using MI (right) is unable to meaningfully distinguish any difference between the related languages.

many linguistic similarities to French and Italian (and Spanish to a lesser extent). While both Basque and Catalan are considered OOD in our setting, we expect good estimates of epistemic uncertainty to capture the property that Catalan is "less OOD" than Basque is. Figure 3 shows that this desired property is captured by the norm of our randomized embedding features, while the mutual information distributions for Basque and Catalan are nearly indistinguishable.

4.3 Malware Detection

Finally, we evaluate the usefulness of our randomized embedding based features in the context of malware detection. Uncovering new or significantly different malware is of particular interest in the quickly evolving cyber security space. We use a dropout variant of the MalConv model (Raff et al. 2017), a convolutional NN for malware detection that operates on raw byte sequences. We apply dropout before each fully connected layer of MalConv. Applying dropout to only the last layers of a NN corresponds to using maximum a posteriori (MAP) estimates for the initial layers and Bayesian estimates for the later layers (Gal and Ghahramani 2016a). We train the dropout MalConv model for 5 epochs with a batch size of 32 on the EMBER2018 dataset which consists of portable executable files (PE files) scanned by VirusTotal in or before 2018 (Anderson and Roth 2018).

Table 3: Performance with and without the cosine randomized embedding spread features for a MalConv model with dropout added before each fully connected layer trained to detect malware using EMBER2018. Standard devs. are reported in Appendix A, and best results are **bolded**.

OOD		Num/Class	n=100		n=50		n=25	
Experiment	Model	Metric / Features	AUC	Recall	AUC	Recall	AUC	Recall
EMBER2018	LR	Last	0.789	0.704	**0.786**	0.682	**0.778**	0.650
		Last+Spread	**0.793**	**0.718**	0.783	**0.689**	0.766	**0.658**
	RF	Last	0.757	0.735	0.752	0.727	0.748	0.714
		Last+Spread	**0.791**	**0.784**	**0.782**	**0.764**	**0.770**	**0.743**
Brazilian	LR	Last	0.685	**0.645**	0.680	0.607	0.668	0.584
		Last+Spread	**0.741**	0.620	**0.734**	**0.617**	**0.712**	**0.605**
	RF	Last	0.724	0.693	0.705	0.674	0.679	0.652
		Last+Spread	**0.839**	**0.797**	**0.813**	**0.772**	**0.776**	**0.736**

We run two experiments on the Bayesian MalConv model. First, of the 200000 files in the EMBER test set, 363 have as their top most likely malware family label (as labeled by AVClass (Sebastián et al. 2016)) a family that was not present in the train set. We evaluate OOD detection performance first on these unseen malware families. Second, we evaluate OOD detection performance on a different malware dataset containing malware samples obtained from a Brazilian financial entity (Ceschin et al. 2019). The malware from this dataset could be considered as OOD due to differing geographical specificity and intent, leading to the use of malware tactics, techniques, and procedures likely specific to a Brazilian banking target. There are also temporal differences as the Brazilian samples were all collected before the EMBER dataset, and we additionally only used malware first seen by VirusTotal before 2012. OOD task training sets consisted of n=100, 50, and just 25 data points from each class (in distribution and OOD). Each experiment was run 100 times with random train/test splits, where all of the data not in the training set is included in the test set. Results are summarized in Table 3, showing that the inclusion of our randomized embedding based features consistently improves OOD detection across experimental settings. Because of the high class imbalance in this use case, as access to good OOD data is more limited in the malware domain, we reported the ROC AUC and the recall for the OOD class in Table 3, noting that recall is often the primary metric of interest in practice for cyber security.

5 Conclusions

We have demonstrated why previous attempts at measuring randomized embedding dispersion using Euclidean distance are inherently flawed. Then we introduced and theoretically justified a cosine distance based, lightweight approach to test

time OOD data detection in the context of dropout Bayesian neural networks. Information that is already computed is used as randomized embeddings, training dataset information does not need to be stored, additional regularization methods are not needed (though do help), and auxiliary neural networks do not need to be trained to take advantage of this additional information. While we note that our approach is limited to dropout BNNs, the popularity of the dropout approximation to BNNs and the existence of previous works exploring the use of stochastic embeddings based on dropout BNNs suggests the applicability of our approach to practice. Our approach can be deployed anywhere a dropout BNN is already deployed with minimal additional overhead. Future work includes the investigation of more elaborate features based off of the randomized embeddings.

References

Amos, B. 2019. *Differentiable optimization-based modeling for machine learning*. Ph.D. thesis, Carnegie Mellon University.

Anderson, H. S.; and Roth, P. 2018. EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models.

Bartlett, P. L.; Evans, S. N.; and Long, P. M. 2018. Representing smooth functions as compositions of near-identity functions with implications for deep network optimization.

Bulatov, Y. 2011. notMNIST dataset.

Ceschin, F.; Pinage, F.; Castilho, M.; Menotti, D.; Oliveira, L. S.; and Gregio, A. 2019. The Need for Speed: An Analysis of Brazilian Malware Classifers. *IEEE Security and Privacy*, 16(6): 31–41.

Chang, J.; Lan, Z.; Cheng, C.; and Wei, Y. 2020. Data Uncertainty Learning in Face Recognition. Technical report.

Chun, S.; Oh, S. J.; de Rezende, R. S.; Kalantidis, Y.; and Larlus, D. 2021. Probabilistic Embeddings for Cross-Modal Retrieval.

Clanuwat, T.; Bober-Irizar, M.; Kitamoto, A.; Lamb, A.; Yamamoto, K.; and Ha, D. 2018. Deep Learning for Classical Japanese Literature.

Damianou, A. C.; and Lawrence, N. D. 2013. Deep Gaussian Processes. 31.

Gal, Y.; and Ghahramani, Z. 2016a. Dropout as a Bayesian Approximation: Appendix. *33rd International Conference on Machine Learning, ICML 2016*, 3: 1661–1680.

Gal, Y.; and Ghahramani, Z. 2016b. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *33rd International Conference on Machine Learning, ICML 2016*, 3: 1651–1660.

Hendrycks, D.; and Gimpel, K. 2016. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. 1–12.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. 1–18.

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 2017-Decem(Nips): 5575–5585.

Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. 1–15.

Liu, J. Z.; Lin, Z.; Padhy, S.; Tran, D.; Bedrax-Weiss, T.; and Lakshminarayanan, B. 2020. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv*, (NeurIPS).

Mandelbaum, A.; and Weinshall, D. 2017. Distance-based Confidence Score for Neural Network Classifiers. Technical report.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral Normalization for Generative Adversarial Networks.

Mukhoti, J.; Kirsch, A.; van Amersfoort, J.; Torr, P. H. S.; and Gal, Y. 2021. Deterministic Neural Networks with Appropriate Inductive Biases Capture Epistemic and Aleatoric Uncertainty.

Oh, S. J.; Murphy, K.; Pan, J.; Roth, J.; Schroff, F.; and Gallagher, A. 2018. Modeling Uncertainty with Hedged Instance Embedding.

Ovadia, Y.; Fertig, E.; Ren, J.; Nado, Z.; Sculley, D.; Nowozin, S.; Dillon, J. V.; Lakshminarayanan, B.; and Snoek, J. 2019. Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift. (NeurIPS).

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32(NeurIPS).

Postels, J.; Blum, H.; Strümpler, Y.; Cadena, C.; Siegwart, R.; Van Gool, L.; and Tombari, F. 2020. The Hidden Uncertainty in a Neural Networks Activations.

Raff, E.; Barker, J.; Sylvester, J.; Brandon, R.; Catanzaro, B.; and Nicholas, C. 2017. Malware Detection by Eating a Whole EXE.

Raff, E.; and Nicholas, C. 2020. A Survey of Machine Learning Methods and Challenges for Windows Malware Classification. 1–48.

Ren, J.; Liu, P. J.; Fertig, E.; Snoek, J.; Poplin, R.; DePristo, M. A.; Dillon, J. V.; and Lakshminarayanan, B. 2019. Likelihood Ratios for Out-of-Distribution Detection.

Sebastián, M.; Rivera, R.; Kotzias, P.; and Caballero, J. 2016. AVCLASS: A Tool for Massive Malware Labeling. Technical report.

Shi, Y.; and Jain, A. K. 2019. Probabilistic Face Embeddings. Technical report.

Smith, L.; and Gal, Y. 2018. Understanding measures of uncertainty for adversarial example detection. *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 2: 560–569.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15: 1929–1958.

Terhörst, P.; Niklas Kolf, J.; Damer, N.; Kirchbuchner, F.; and Kuijper, A. 2020. SER-FIQ: Unsupervised Estimation of Face Image Quality Based on Stochastic Embedding Robustness. In *CVPR*.

Thoma, M. 2018. The WiLI benchmark dataset for written language identification.

van Amersfoort, J.; Smith, L.; Teh, Y. W.; and Gal, Y. 2020. Uncertainty Estimation Using a Single Deep Deterministic Neural Network.

Vincent, P.; Larochelle, H.; Bengio, Y.; and Manzagol, P.-A. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *ICML*.

Vincent, P.; Larochelle, H.; Lajoie, I.; Bengio, Y.; and Manzagol, P.-A. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, 11: 3371–3408.

Wilson, A. G.; and Izmailov, P. 2020. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. (3).

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

Xiao, T. Z.; Gomez, A. N.; and Gal, Y. 2020. Wat zei je? Detecting Out-of-Distribution Translations with Variational Transformers. (NeurIPS): 4–7.

Yann LeCun; Léeon Bottou; Yoshua Bengio; and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *IEEE*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2016. Character-level Convolutional Networks for Text Classification.

A Appendix

A.1 Experimental Result Standard Deviations

We repeated each of our experiments multiple times and computed a mean and standard deviation for each experiment and evaluation metric. In all of our experiments, the standard deviations are much smaller than effect sizes, so we reported only the means in section 4. Here we report the complete results, which include standard deviations, for all of our experiments. Vision experiment results are summarized in Table 4. Language experiment results are summarized in Table 5. Malware experiment results are summarized in Table 6.

Table 4: Performance with and without the cosine randomized embedding spread features for various experimental configurations for a dropout LeNet5 trained on MNIST. Features labeled as "Last" consist of common baseline features computed using softmax output samples from the network (predictive entropy, mutual information, and maximum softmax probability). Features labeled as "Last+Spread" consist of these baseline features plus our additional randomized embedding maximum cosine spread features for each layer. Each experiment was repeated multiple times, and the mean and standard deviation are reported here. Best results are shown in **bold**.

			Num/Class	n=1000				n=100				n=10			
			Metric	AUC		Acc		AUC		Acc		AUC		Acc	
			Statistic	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
OOD Train	OOD Test	OOD Model	Features												
Fashion	Kuzushiji	LR	Last	0.969	0.000	**0.914**	**0.001**	0.967	0.004	0.909	0.009	0.963	0.024	0.884	0.023
			Last+Spread	**0.979**	**0.001**	0.914	0.003	**0.973**	**0.008**	**0.911**	**0.013**	**0.967**	**0.035**	**0.901**	**0.033**
		RF	Last	0.960	0.001	0.917	0.002	0.952	0.005	0.905	0.009	0.942	0.015	0.884	0.038
			Last+Spread	**0.979**	**0.001**	**0.922**	**0.004**	**0.974**	**0.003**	**0.921**	**0.005**	**0.969**	**0.008**	**0.907**	**0.018**
	notMNIST	LR	Last	0.966	0.001	0.912	0.002	0.965	0.003	0.909	0.011	0.960	0.004	0.879	0.024
			Last+Spread	**0.983**	**0.001**	**0.932**	**0.003**	**0.979**	**0.005**	**0.925**	**0.015**	**0.967**	**0.011**	**0.892**	**0.021**
		RF	Last	0.959	0.002	0.920	0.004	0.950	0.005	0.903	0.010	0.938	0.014	0.880	0.038
			Last+Spread	**0.985**	**0.001**	**0.940**	**0.004**	**0.976**	**0.004**	**0.924**	**0.006**	**0.963**	**0.009**	**0.901**	**0.020**
Kuzushiji	Fashion	LR	Last	0.973	0.000	0.920	0.001	0.972	0.001	0.917	0.006	0.967	0.011	0.899	0.016
			Last+Spread	**0.989**	**0.001**	**0.948**	**0.002**	**0.983**	**0.004**	**0.937**	**0.008**	**0.978**	**0.004**	**0.922**	**0.014**
		RF	Last	0.964	0.001	0.920	0.002	0.956	0.005	0.907	0.009	0.946	0.015	0.896	0.025
			Last+Spread	**0.986**	**0.001**	**0.943**	**0.002**	**0.978**	**0.004**	**0.931**	**0.006**	**0.967**	**0.009**	**0.914**	**0.016**
	notMNIST	LR	Last	0.967	0.000	0.914	0.001	0.965	0.002	0.910	0.010	0.960	0.005	0.886	0.020
			Last+Spread	**0.984**	**0.001**	**0.931**	**0.004**	**0.975**	**0.008**	**0.914**	**0.020**	**0.966**	**0.007**	**0.888**	**0.021**
		RF	Last	0.960	0.001	0.922	0.003	0.950	0.005	0.904	0.010	0.938	0.017	0.888	0.026
			Last+Spread	**0.982**	**0.001**	**0.935**	**0.003**	**0.971**	**0.006**	**0.921**	**0.006**	**0.954**	**0.010**	**0.896**	**0.019**
notMNIST	Fashion	LR	Last	0.966	0.003	0.911	0.003	0.957	0.018	0.906	0.012	0.959	0.037	0.893	0.023
			Last+Spread	**0.978**	**0.003**	**0.937**	**0.005**	**0.969**	**0.016**	**0.928**	**0.015**	**0.977**	**0.014**	**0.925**	**0.018**
		RF	Last	0.960	0.002	0.910	0.004	0.955	0.005	0.904	0.011	0.946	0.018	0.887	0.032
			Last+Spread	**0.988**	**0.001**	**0.943**	**0.005**	**0.982**	**0.004**	**0.935**	**0.007**	**0.978**	**0.006**	**0.920**	**0.017**
	Kuzushiji	LR	Last	0.960	0.006	**0.900**	**0.007**	0.946	0.030	**0.893**	**0.021**	0.951	0.057	0.882	0.035
			Last+Spread	**0.966**	**0.006**	0.893	0.010	**0.949**	**0.028**	0.886	0.031	**0.967**	**0.030**	**0.902**	**0.035**
		RF	Last	0.956	0.002	**0.906**	**0.005**	0.950	0.006	0.901	0.012	0.941	0.020	0.883	0.033
			Last+Spread	**0.978**	**0.001**	0.906	0.008	**0.973**	**0.004**	**0.915**	**0.012**	**0.969**	**0.008**	**0.906**	**0.023**

Table 5: Performance with and without the cosine randomized embedding spread features for a Char-CNN with dropout added before every layer trained to classify languages using the WiLI dataset. Best results are shown in **bold**.

OOD Train	OOD Test	OOD Model	Num/Class	n=100				n=50				n=25				n=10			
			Metric	AUC		Acc		AUC		Acc		AUC		Acc		AUC		Acc	
			Statistic	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
			Features																
Basque	rest	LR	Last	0.888	0.005	0.798	0.008	0.883	0.014	0.794	0.010	0.882	0.015	0.792	0.011	0.878	0.029	0.786	0.019
			Last+Spread	**0.926**	**0.019**	**0.843**	**0.019**	**0.919**	**0.033**	**0.836**	**0.023**	**0.921**	**0.034**	**0.835**	**0.025**	**0.926**	**0.026**	**0.828**	**0.023**
		RF	Last	0.862	0.008	0.797	0.007	0.857	0.013	0.793	0.011	0.851	0.015	0.792	0.013	0.842	0.015	0.789	0.021
			Last+Spread	**0.924**	**0.008**	**0.845**	**0.011**	**0.920**	**0.011**	**0.840**	**0.013**	**0.918**	**0.012**	**0.835**	**0.017**	**0.914**	**0.013**	**0.824**	**0.018**
Finnish	rest	LR	Last	0.888	0.003	0.795	0.006	0.885	0.006	0.792	0.008	0.881	0.013	0.790	0.011	0.883	0.008	0.786	0.015
			Last+Spread	**0.910**	**0.019**	**0.818**	**0.017**	**0.908**	**0.032**	**0.818**	**0.022**	**0.909**	**0.035**	**0.821**	**0.024**	**0.910**	**0.041**	**0.818**	**0.028**
		RF	Last	0.864	0.006	0.794	0.007	0.858	0.009	0.792	0.011	0.850	0.014	0.789	0.015	0.840	0.020	0.783	0.024
			Last+Spread	**0.913**	**0.011**	**0.821**	**0.012**	**0.907**	**0.013**	**0.821**	**0.012**	**0.905**	**0.014**	**0.818**	**0.014**	**0.904**	**0.016**	**0.814**	**0.015**
Luganda	rest	LR	Last	0.891	0.002	0.806	0.005	0.889	0.004	0.803	0.008	0.887	0.006	0.800	0.009	0.881	0.026	0.794	0.015
			Last+Spread	**0.943**	**0.006**	**0.864**	**0.016**	**0.939**	**0.008**	**0.854**	**0.017**	**0.935**	**0.009**	**0.847**	**0.014**	**0.931**	**0.009**	**0.837**	**0.017**
		RF	Last	0.866	0.006	0.800	0.007	0.859	0.010	0.797	0.011	0.854	0.014	0.796	0.013	0.843	0.027	0.785	0.038
			Last+Spread	**0.936**	**0.005**	**0.862**	**0.009**	**0.930**	**0.008**	**0.852**	**0.014**	**0.926**	**0.009**	**0.843**	**0.015**	**0.921**	**0.011**	**0.831**	**0.018**
Polish	rest	LR	Last	0.900	0.003	0.824	0.003	0.897	0.010	0.821	0.007	0.891	0.019	0.816	0.012	0.887	0.042	0.812	0.021
			Last+Spread	**0.939**	**0.010**	**0.866**	**0.014**	**0.938**	**0.014**	**0.864**	**0.017**	**0.935**	**0.021**	**0.860**	**0.023**	**0.934**	**0.020**	**0.852**	**0.022**
		RF	Last	0.870	0.010	0.793	0.011	0.860	0.017	0.787	0.015	0.854	0.021	0.783	0.023	0.850	0.029	0.780	0.034
			Last+Spread	**0.937**	**0.005**	**0.871**	**0.008**	**0.932**	**0.008**	**0.863**	**0.011**	**0.928**	**0.009**	**0.855**	**0.012**	**0.922**	**0.018**	**0.841**	**0.024**
Tongan	rest	LR	Last	0.857	0.115	**0.815**	**0.060**	0.841	0.159	**0.811**	**0.076**	0.815	0.198	0.800	0.088	0.791	0.244	0.771	0.155
			Last+Spread	**0.886**	**0.060**	0.811	0.056	**0.877**	**0.091**	0.810	0.069	**0.884**	**0.101**	**0.819**	**0.074**	**0.880**	**0.125**	**0.813**	**0.085**
		RF	Last	0.765	0.063	0.699	0.055	0.766	0.074	0.695	0.067	0.769	0.092	0.684	0.075	0.785	0.129	0.701	0.101
			Last+Spread	**0.915**	**0.016**	**0.847**	**0.029**	**0.913**	**0.019**	**0.845**	**0.028**	**0.906**	**0.030**	**0.836**	**0.035**	**0.903**	**0.051**	**0.823**	**0.054**
Xhosa	rest	LR	Last	0.894	0.004	0.807	0.008	0.891	0.007	0.804	0.008	0.886	0.019	0.800	0.013	0.879	0.046	0.794	0.022
			Last+Spread	**0.944**	**0.009**	**0.866**	**0.014**	**0.940**	**0.014**	**0.857**	**0.019**	**0.933**	**0.020**	**0.846**	**0.024**	**0.931**	**0.028**	**0.838**	**0.029**
		RF	Last	0.864	0.009	0.787	0.013	0.857	0.016	0.782	0.020	0.849	0.021	0.778	0.027	0.846	0.028	0.774	0.032
			Last+Spread	**0.939**	**0.006**	**0.868**	**0.011**	**0.934**	**0.009**	**0.860**	**0.013**	**0.928**	**0.012**	**0.852**	**0.017**	**0.921**	**0.024**	**0.835**	**0.021**

Table 6: Performance with and without the cosine randomized embedding spread features for a MalConv model with dropout added before each fully connected layer trained to detect malware using the EMBER2018 dataset. Best results are shown in **bold**.

Experiment	OOD Model	Num/Class	n=100				n=50				n=25			
		Metric	AUC		Recall		AUC		Recall		AUC		Recall	
		Statistic	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
		Features												
EMBER2018	LR	Last	0.789	0.007	0.704	0.043	**0.786**	**0.008**	0.682	0.054	**0.778**	**0.018**	0.650	0.066
		Last+Spread	**0.793**	**0.008**	**0.718**	**0.042**	0.783	0.013	**0.689**	**0.067**	0.766	0.027	**0.658**	**0.080**
	RF	Last	0.757	0.011	0.735	0.046	0.752	0.015	0.727	0.060	0.748	0.023	0.714	0.079
		Last+Spread	**0.791**	**0.011**	**0.784**	**0.045**	**0.782**	**0.014**	**0.764**	**0.057**	**0.770**	**0.018**	**0.743**	**0.084**
Brazilian	LR	Last	0.685	0.007	**0.645**	**0.054**	0.680	0.010	0.607	0.072	0.668	0.042	0.584	0.078
		Last+Spread	**0.741**	**0.023**	0.620	0.039	**0.734**	**0.023**	**0.617**	**0.049**	**0.712**	**0.034**	**0.605**	**0.063**
	RF	Last	0.724	0.016	0.693	0.038	0.705	0.023	0.674	0.055	0.679	0.035	0.652	0.081
		Last+Spread	**0.839**	**0.010**	**0.797**	**0.034**	**0.813**	**0.016**	**0.772**	**0.054**	**0.776**	**0.024**	**0.736**	**0.083**

A.2 Additional Spectral Normalization Results

We repeated the vision OOD data detection experiments from section 4.1 on a spectral normalized dropout LeNet5 trained for 100 epochs. While spectral normalization is not required to see an improvement from the inclusion of cosine spread features, spectral normalization improves OOD detection performance consistently, as shown in Table 7 when compared to Table 4.

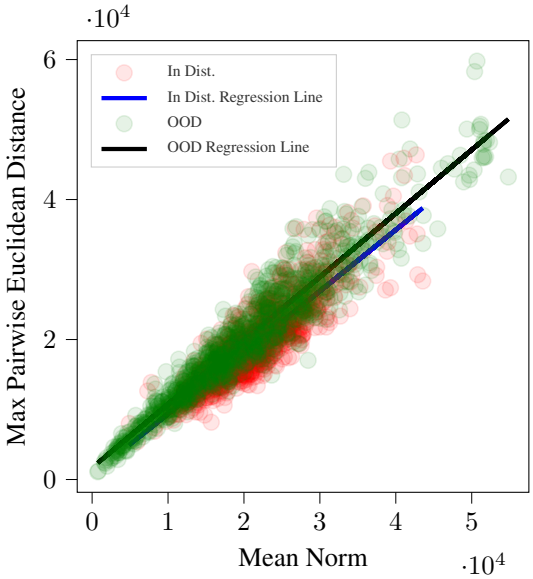
Table 7: Performance with and without the cosine randomized embedding spread features for a dropout, spectral normalized LeNet5 trained on MNIST. Best results are shown in **bold**.

			Num/Class	n=1000				n=100				n=10			
			Metric	AUC		Acc		AUC		Acc		AUC		Acc	
			Statistic	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
OOD Train	OOD Test	OOD Model	Features												
Fashion	Kuzushiji	LR	Last	0.982	0.003	0.921	0.005	0.978	0.016	0.921	0.016	0.980	0.040	0.908	0.032
			Last+Spread	**0.984**	**0.002**	**0.926**	**0.006**	**0.980**	**0.015**	**0.926**	**0.019**	**0.984**	**0.008**	**0.915**	**0.024**
		RF	Last	0.979	0.001	0.936	0.004	0.972	0.004	0.931	0.010	0.963	0.014	0.919	0.024
			Last+Spread	**0.985**	**0.001**	**0.939**	**0.004**	**0.982**	**0.002**	**0.940**	**0.007**	**0.981**	**0.008**	**0.932**	**0.017**
	notMNIST	LR	Last	**0.982**	**0.003**	0.916	0.008	0.975	0.025	0.915	0.024	0.979	0.049	0.905	0.036
			Last+Spread	0.982	0.002	**0.922**	**0.009**	**0.975**	**0.025**	**0.922**	**0.026**	**0.983**	**0.012**	**0.913**	**0.025**
		RF	Last	0.978	0.001	0.933	0.005	0.971	0.005	0.929	0.012	0.962	0.014	0.917	0.026
			Last+Spread	**0.984**	**0.001**	**0.942**	**0.004**	**0.981**	**0.003**	**0.940**	**0.008**	**0.980**	**0.007**	**0.932**	**0.018**
Kuzushiji	Fashion	LR	Last	**0.988**	**0.000**	0.948	0.002	**0.987**	**0.003**	0.944	0.008	0.979	0.045	0.918	0.033
			Last+Spread	0.987	0.001	**0.949**	**0.002**	0.985	0.003	**0.944**	**0.007**	**0.981**	**0.041**	**0.932**	**0.037**
		RF	Last	0.980	0.001	0.946	0.002	0.972	0.004	0.938	0.007	0.966	0.009	0.930	0.021
			Last+Spread	**0.985**	**0.001**	**0.950**	**0.002**	**0.982**	**0.004**	**0.947**	**0.003**	**0.983**	**0.002**	**0.943**	**0.007**
	notMNIST	LR	Last	**0.986**	**0.000**	0.937	0.001	**0.985**	**0.002**	0.934	0.007	0.979	0.029	0.910	0.025
			Last+Spread	0.986	0.001	**0.946**	**0.001**	0.984	0.003	**0.939**	**0.006**	**0.983**	**0.017**	**0.922**	**0.021**
		RF	Last	0.979	0.001	0.941	0.002	0.972	0.004	0.933	0.008	0.964	0.011	0.924	0.023
			Last+Spread	**0.985**	**0.001**	**0.947**	**0.001**	**0.982**	**0.002**	**0.944**	**0.003**	**0.982**	**0.002**	**0.937**	**0.010**
notMNIST	Fashion	LR	Last	0.988	0.001	0.946	0.002	**0.986**	**0.003**	0.941	0.009	0.983	0.018	0.916	0.027
			Last+Spread	**0.988**	**0.001**	**0.951**	**0.002**	0.985	0.006	**0.944**	**0.010**	**0.986**	**0.002**	**0.932**	**0.016**
		RF	Last	0.980	0.001	0.945	0.002	0.971	0.005	0.938	0.007	0.966	0.010	0.931	0.017
			Last+Spread	**0.987**	**0.001**	**0.952**	**0.001**	**0.984**	**0.002**	**0.947**	**0.004**	**0.983**	**0.004**	**0.941**	**0.010**
	Kuzushiji	LR	Last	0.986	0.000	0.936	0.002	0.984	0.003	0.934	0.007	0.984	0.002	0.911	0.023
			Last+Spread	**0.989**	**0.000**	**0.947**	**0.001**	**0.987**	**0.002**	**0.942**	**0.006**	**0.987**	**0.003**	**0.923**	**0.018**
		RF	Last	0.980	0.001	0.942	0.002	0.972	0.004	0.934	0.007	0.963	0.011	0.926	0.018
			Last+Spread	**0.987**	**0.000**	**0.947**	**0.001**	**0.984**	**0.002**	**0.944**	**0.004**	**0.983**	**0.005**	**0.936**	**0.012**

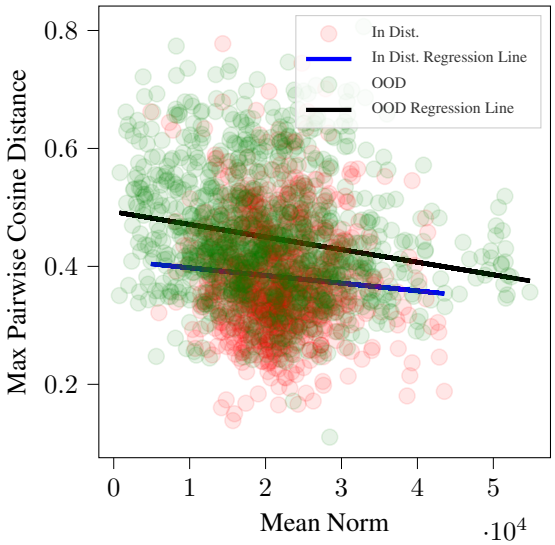
A.3 Cosine Distance vs. Euclidean Distance for Unsupervised Embeddings

We also investigated cosine distance versus Euclidean distance for measuring randomized embedding dispersion in the unsupervised setting. In particular, we investigated a stacked denoising autoencoder variant (Vincent et al. 2008, 2010) where all layers are trained at the same time instead of stage-wise, and dropout with a dropout probability $p = 0.1$ is used as the corrupting process at each layer of the encoder. At test time, the dropout corruption is left on to generate randomized embeddings. The denoising autoencoder was trained on MNIST for 20 epochs with a batch size of 64 using the Adam optimizer with a learning rate of 0.001, the default recommended settings, and a weight decay of 0.01. Image inputs were flattened, and the encoder architecture consisted of 6 ReLU activated linear layers of output dimensions: 784, 400, 400, 120, 120, and 84. The decoder architecture is similar to the encoder architecture but in reverse order.

Figure 4a and Figure 4b show consistent results. Embedding dispersion as measured by Euclidean distance is related to mean norm in an identical manner across in distribution and OOD data. While not as well separated as in the supervised setting, in distribution data has lower embedding dispersion as measured by cosine distance when compared to OOD data.



(a) Denoising autoencoder, Euclidean distance.



(b) Denoising autoencoder, cosine distance.

Figure 4: A comparison of the relationships between denoising autoencoder randomized embedding mean norm and the maximum pairwise distance for Euclidean distance and cosine distance respectively, for in distribution data (MNIST) and OOD data (Not-MNIST). Regression line fits are provided for each as well for easier comparison.

A.4 Simulations

Mean and Variance of the Embedding Norms We perform a simulation to further illustrate the problem with the use of Euclidean distance in the case of a two layer ReLU activated network. As the depth of the BNN increases, the mean and variance of the embedding norms dramatically increase across layers, in particular as a consequence of the ReLU activation. This is known and bounds for this can be derived mathematically using the identity $max(x, 0) = 0.5(x + |x|)$ in the normal random matrix situation. However, we identify that the variance of the norms experiences a further increase due to the effect of dropout on preceding layers causing a carryover of variance into subsequent layers. Because dropout samples are taken across all layers simultaneously, the signal representing the distance between two embedding samples in layer N is diluted with the inflated norm caused by preceding dropout in layers 1 to $N - 1$. This is confirmed by simulation on a two-layer neural network with dropout in Table 8, where the variance of the final embedding norms (4526.2) is much higher than it would be if dropout were only applied on that embedding layer (3124.0). This can explain why the Euclidean distance measure fails to perform for OOD detection.

Table 8: Mean and (variance) of the embedding norms in a simulated context.

	Dropout only layer 1	Dropout only layer 2	Dropout both layers
Layer 1 embedding norm	96.0 (58.6)	118.6 (0.0)	96.0 (58.6)
Layer 2 embedding norm	599.7 (3328.0)	606.1 (3124.0)	501.0 (4526.2)

Correlation Analysis Between Measures of Uncertainty To examine the relationships between the uncertainty features, we ran correlation analysis between all measures on the final embedding layer of a neural network, averaged over 1000 random matrix iterations. The embeddings form a $D \times B$ matrix, where D is the embedding dimension and B are the number of dropout samples, and we enforce a decaying correlation structure over the embedding dimensions. In Table 9, we summarize the correlations between all predictive features.

This result indicates that the previously used features have higher inter-correlation than the max cosine pairwise distance, suggesting that our new feature adds an orthogonal measure of information that is not previously captured. This helps explain our improvement in OOD detection.

Table 9: Correlation analysis between measures of uncertainty in a simulated setting.

	mutual info.	pred entr.	max softmax	max cos pdist	max euclid pdist	mean embed. norm
mutual info.	1.00	-0.31	0.23	0.08	0.32	0.51
pred entr.	-0.31	1.00	-0.64	0.01	-0.09	-0.26
max softmax	0.23	-0.64	1.00	0.01	0.06	0.13
max cos pdist	0.08	0.01	0.01	1.00	0.15	-0.14
max euclid pdist	0.32	-0.09	0.06	0.15	1.00	0.32
mean embed. norm	0.51	-0.26	0.13	-0.14	0.32	1.00

A.5 Additional Experiments on MNIST Variants

Is Some OOD Training Data Needed? To compare with methods that do not require any OOD training data at all, we attempted the following where a linear kernel one class SVM and an Isolation Forest are used as outlier detectors that would hopefully capture OOD data. Results are shown in Table 10. Generally, the best AUC is achieved using an Isolation Forest but the accuracy remains low. This is consistent with our conclusions that the relationship contains non-linear information and that some form of OOD data is needed to choose the appropriate threshold, and that as few as $n = 10$ OOD points can estimate that threshold with significantly greater accuracy and AUC.

Table 10: To compare with methods that do not require any OOD training data at all, we attempted the following where a linear kernel one class SVM and an Isolation Forest (IF) are used as outlier detectors.

OOD Test	OOD Model	Metric Features	AUC	Acc
Kuzushiji	SVM	Last	0.555574	0.539760
		Last+Spread	0.190757	0.253412
	IF	Last	0.876447	0.804457
		Last+Spread	0.858214	0.617729
notMNIST	SVM	Last	0.532724	0.523312
		Last+Spread	0.307177	0.335988
	IF	Last	0.842468	0.766183
		Last+Spread	0.869251	0.631671
Fashion	SVM	Last	0.526127	0.514582
		Last+Spread	0.212349	0.262806
	IF	Last	0.860657	0.791121
		Last+Spread	0.883436	0.649500

Results when using Euclidean Randomized Embedding Maximum Spread Features To compare Euclidean distance features with cosine distance features, we ran experiments and found that cosine does empirically does better, as expected. In Table 11 are the results for the MNIST experiments where the Spread features use Euclidean distance.

Classifier Feature Importances To further understand the contribution of our cosine distance measure, we compute the mean and standard deviation of feature Gini importances for the random forest classifiers fit across our MNIST variant experiments. Results are shown in Figure 5 and show that our spread based features are important with layer 3's spread having a Gini importance comparable to traditional features such as predictive entropy.

Table 11: MNIST variant results when using Euclidean randomized embedding maximum spread features.

OOD Train	OOD Test	OOD Model	Num/Class Metric Features	n=1000 AUC	n=1000 Acc	n=100 AUC	n=100 Acc	n=10 AUC	n=10 Acc
Fashion	Kuzushiji	LR	Last	0.970909	0.915305	0.968269	0.910754	0.967937	0.891701
			Last+Spread	0.959380	0.906095	0.953365	0.897221	0.941887	0.864347
		RF	Last	0.961252	0.916221	0.949761	0.899854	0.946880	0.880030
			Last+Spread	0.958593	0.896063	0.945695	0.892397	0.943430	0.879735
	notMNIST	LR	Last	0.966394	0.910832	0.965956	0.910824	0.961222	0.882841
			Last+Spread	0.966064	0.914921	0.955173	0.901126	0.922676	0.839990
		RF	Last	0.958357	0.916837	0.947242	0.898442	0.939856	0.876058
			Last+Spread	0.966978	0.927221	0.953618	0.910864	0.945218	0.882916
Kuzushiji	Fashion	LR	Last	0.972502	0.919816	0.971007	0.917156	0.968018	0.900125
			Last+Spread	0.968836	0.923116	0.962746	0.916281	0.961164	0.898184
		RF	Last	0.963407	0.920547	0.953377	0.903246	0.938179	0.885983
			Last+Spread	0.964453	0.919921	0.955725	0.908291	0.941819	0.894097
	notMNIST	LR	Last	0.967112	0.914289	0.965904	0.911111	0.960355	0.885173
			Last+Spread	0.975508	0.925026	0.966202	0.917970	0.938949	0.869555
		RF	Last	0.959856	0.920416	0.948595	0.903442	0.928044	0.876868
			Last+Spread	0.968210	0.925442	0.958487	0.912819	0.934295	0.887989
notMNIST	Fashion	LR	Last	0.964710	0.911000	0.947280	0.902452	0.968372	0.894182
			Last+Spread	0.963373	0.918600	0.949651	0.909618	0.962364	0.887704
		RF	Last	0.960297	0.909847	0.955611	0.903392	0.946035	0.888309
			Last+Spread	0.968320	0.920963	0.963052	0.906211	0.954943	0.906248
	Kuzushiji	LR	Last	0.960289	0.899979	0.938433	0.888839	0.967569	0.887464
			Last+Spread	0.956063	0.901247	0.938300	0.890312	0.950506	0.865883
		RF	Last	0.958770	0.905458	0.953364	0.902915	0.940991	0.884212
			Last+Spread	0.959689	0.892047	0.957301	0.895558	0.946487	0.895423

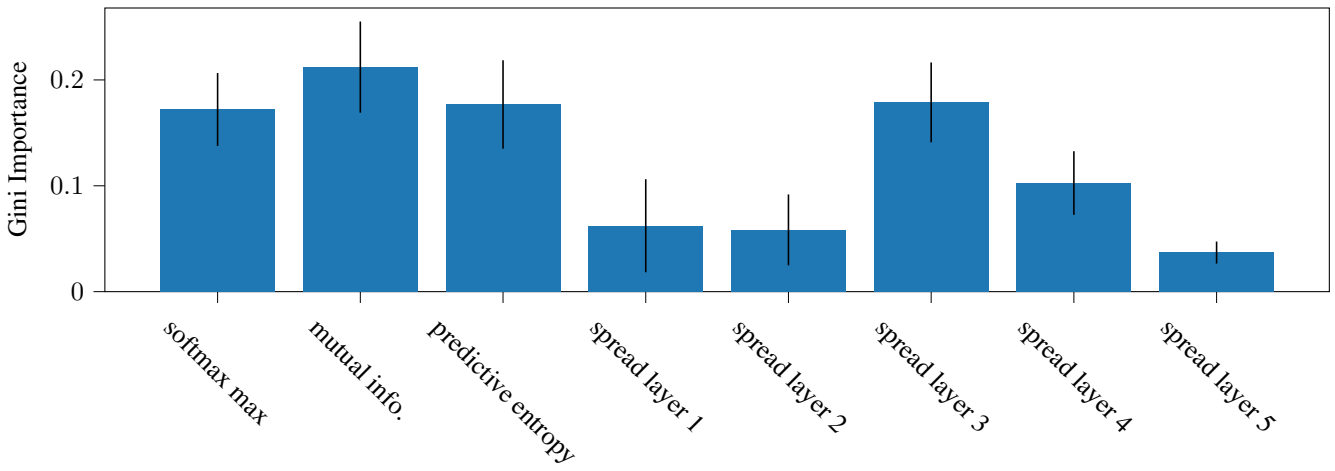


Figure 5: Random Forest Gini feature importances for MNIST variant experiments. Means and standard deviations are shown.

A.6 Embedding Component Variance

In the context of a linear layer with input $\mathbf{x}$ indexed by i, output $\mathbf{y}$ indexed by j, weight matrix W, bias $\mathbf{b}$, dropout with probability p of *not* being dropped, the layer can be written as

$$y_j = \left(\sum_i D_i \cdot W_{ij} \cdot x_i \right) + b_j$$

where $D_i \sim \text{Bern}(p)$ are i.i.d. Bernoulli random variables with probability parameter p. The variance of an embedding component can be written as follows:

$$\text{Var}(y_j) = \text{Var}\left[\left(\sum_i D_i \cdot W_{ij} \cdot x_i \right) + b_j \right]$$

Variance is invariant to changes in a location parameter, and the D_i are i.i.d. allowing us to write:

$$\text{Var}(y_j) = \sum_i (W_{ij} \cdot x_i)^2 \, \text{Var}(D_i) = \sum_i (W_{ij} \cdot x_i)^2 \, p \, (1 - p)$$

A.7 Dataset Links

Data used in the image classification experiments can be found here:

- http://yann.lecun.com/exdb/mnist/
- https://github.com/davidflanagan/notMNIST-to-MNIST
- https://github.com/rois-codh/kmnist
- https://github.com/zalandoresearch/fashion-mnist

Data used in the language classification experiments can be found here: https://zenodo.org/record/841984#.YK0r8S1h1pQ
Part of the data used in the malware detection experiments can be found here:

- https://github.com/elastic/ember
- https://github.com/fabriciojoc/brazilian-malware-dataset

The 1.1TB of raw PE files are not available as part of EMBER2018, but they can be downloaded via VirusTotal: https://www.virustotal.com/gui/