NADS: Neural Architecture Distribution Search for Uncertainty Awareness

Randy Ardywibowo [1] Shahin Boluki [1] Xinyu Gong [2] Zhangyang Wang [2] Xiaoning Qian [1]

Abstract

Machine learning (ML) systems often encounter Out-of-Distribution (OoD) errors when dealing with testing data coming from a distribution different from training data. It becomes important for ML systems in critical applications to accurately quantify its predictive uncertainty and screen out these anomalous inputs. However, existing OoD detection approaches are prone to errors and even sometimes assign higher likelihoods to OoD samples. Unlike standard learning tasks, there is currently no well established guiding principle for designing OoD detection architectures that can accurately quantify uncertainty. To address these problems, we first seek to identify guiding principles for designing uncertainty-aware architectures, by proposing *Neural Architecture Distribution Search* (NADS). NADS searches for a distribution of architectures that perform well on a given task, allowing us to identify common building blocks among all uncertainty-aware architectures. With this formulation, we are able to optimize a stochastic OoD detection objective and construct an ensemble of models to perform OoD detection. We perform multiple OoD detection experiments and observe that our NADS performs favorably, with up to 57% improvement in accuracy compared to state-of-the-art methods among 15 different testing configurations.

1. Introduction

Detecting anomalous data is crucial for safely applying machine learning in autonomous systems for critical applications and for AI safety (Amodei et al., 2016). Such anomalous data can come in settings such as in autonomous driving (Kendall & Gal, 2017; NHTSA, 2017), disease monitoring (Hendrycks & Gimpel, 2016; Ardywibowo et al., 2019; 2018), and fault detection (Hendrycks et al., 2019b). In these situations, it is important for these systems to reliably detect abnormal inputs so that their occurrence can be overseen by a human, or the system can proceed using a more conservative policy.

The widespread use of deep learning models within these autonomous systems have aggravated this issue. Despite having high performance in many predictive tasks, deep networks tend to give high confidence predictions on Out-of-Distribution (OoD) data (Goodfellow et al., 2015; Nguyen et al., 2015). Moreover, commonly used OoD detection approaches are prone to errors and even assign higher likelihoods to samples from other datasets (Lee et al., 2018; Hendrycks & Gimpel, 2016).

Unlike common machine learning tasks such as image classification, segmentation, and speech recognition, there are currently no well established guidelines for designing architectures that can accurately screen out OoD data and quantify its predictive uncertainty. Such a gap makes Neural Architecture Search (NAS) a promising option to explore the better design of uncertainty-aware models (Elsken et al., 2018). NAS algorithms attempt to find an optimal neural network architecture for a specific task. Existing efforts have primarily focused on searching for architectures that perform well on image classification or segmentation. However, it is unclear whether architecture components that are beneficial for image classification and segmentation models would also lead to better uncertainty quantification (UQ) and thereafter be effective for OoD detection.

Because of this, it is necessary to tailor the search objective in order to find the architectures that can accurately detect OoD data. However, designing an optimization objective that leads to uncertainty-aware models is also not straightforward. With no access to labels for OoD data, unsupervised/self-supervised generative models maximizing the likelihood of in-distribution data become the primary tools for UQ (Hendrycks et al., 2019a). However, these models counter-intuitively assign high likelihoods to OoD data (Nalisnick et al., 2019a; Choi & Jang, 2018; Hendrycks et al., 2019a; Shafaei et al.). Because of this,

[1] Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA [2] Department of Computer Science and Engineering, Texas A&M University, College Station, Texas, USA. Correspondence to: Randy Ardywibowo <randyardywibowo@tamu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

maximizing the log-likelihood is inadequate for OoD detection. On the other hand, Choi & Jang (2018) proposed using the Widely Applicable Information Criterion (WAIC) (Watanabe, 2013), a penalized likelihood score, as the OoD detection criterion, showing that it was robust for OoD detection. However, the score was approximated using an ensemble of models that was trained on maximizing the likelihood and did not directly optimize the WAIC score. In line with this, previous work on deep uncertainty quantification show that ensembles can help calibrate OoD classifier based methods, as well as improve OoD detection performance of likelihood estimation models (Lakshminarayanan et al., 2017). Based on these findings, one might consider finding a distribution of well-performing architectures for uncertainty awareness, instead of searching for a single best performing architecture, as is typically done in existing NAS methods.

To this end, we propose *Neural Architecture Distribution Search* (**NADS**) to identify common building blocks that naturally incorporate model uncertainty quantification and compose good OoD detection models. NADS searches for a **distribution** of well-performing architectures, instead of a single best architecture, by formulating the architecture search problem as a stochastic optimization problem. We optimize the WAIC score of the architecture distribution, a score that was shown to be robust towards estimating model uncertainty. By taking advantage of weight sharing between different architectures, as well as through a particular parameterization of the architecture distribution, the discrete search problem for NADS can be efficiently solved by a continuous relaxation (Xie et al., 2018; Chang et al., 2019). Using the learned posterior architecture distribution, we construct a Bayesian ensemble of deep models to perform OoD detection, demonstrating state-of-the-art performance in multiple OoD detection experiments. Specifically, our main contributions with NADS include:

- NADS learns a posterior distribution on the architecture search space to enable UQ for better OoD detection, instead of providing a maximum-likelihood point estimate to the best model.
- We design a novel generative search space that is inspired by Glow (Kingma & Dhariwal, 2018), which is different from previous NAS methods.
- We use the WAIC score as the reward to guide the architecture search and provide a method to estimate this score for architecture search.
- NADS yields state-of-the-art performance in multiple OoD detection experiments, making likelihood estimation based OoD detection competitive against multiclass classifier based approaches. Notably, our method yields consistent improvements in accuracy among 15 different in-distribution – out-of-distribution testing pairs, with an improvement of up to 57% accuracy against existing state-of-the-art methods.

2. Background

2.1. Neural Architecture Search

Neural Architecture Search (NAS) algorithms aim to automatically discover an optimal neural network architecture instead of using a hand-crafted one for a specific task. Previous work on NAS has achieved successes in image classification (Pham et al., 2018), image segmentation (Liu et al., 2019), object detection (Ghiasi et al., 2019), structured prediction (Chen et al., 2018), and generative adversarial networks (Gong et al., 2019). However, there has been no NAS algorithm developed for uncertainty quantificaton and OoD detection.

NAS consists of three components: the proxy task, the search space, and the optimization algorithm. Prior work in specifying the search space either searches for an entire architecture directly, or searches for small cells and arrange them in a pre-defined way. Optimization algorithms that have been used for NAS include reinforcement learning (Baker et al., 2017; Zoph et al., 2018; Zhong et al., 2018; Zoph & Le, 2016), Bayesian optimization (Jin et al., 2018), random search (Chen et al., 2018), Monte Carlo tree search (Negrinho & Gordon, 2017), and gradient-based optimization methods (Liu et al., 2018b; Ahmed & Torresani, 2018; Xie et al., 2018; Chang et al., 2019). To efficiently evaluate the performance of discovered architectures and guide the search, the design of the proxy task is critical. Existing proxy tasks include leveraging shared parameters (Pham et al., 2018), predicting performance using a surrogate model (Liu et al., 2018a), and early stopping (Zoph et al., 2018; Chen et al., 2018).

To the best of our knowledge, all existing NAS algorithms seek a single best performing architecture. In comparison, searching for a distribution of architectures allows us to analyze the common building blocks that all of the candidate architectures have. Moreover, this technique can also complement ensemble methods by creating a more diverse set of models tailored to optimize the ensemble objective, an important ingredient for deep uncertainty quantification (Lakshminarayanan et al., 2017; Choi & Jang, 2018).

2.2. Uncertainty Quantification and OoD Detection

Prior work on uncertainty quantification and OoD detection for deep models can be divided into model-dependent (Lakshminarayanan et al., 2017; Gal & Ghahramani, 2016; Boluki et al., 2020; Liang et al., 2017), and model-independent techniques (Dinh et al., 2016; Germain et al., 2015; Oord et al., 2016). Model-dependent techniques aim to yield confidence measures $p(y|\boldsymbol{x})$ for a model's prediction y when given input data $\boldsymbol{x}$. However, a limitation of model-dependent OoD detection is that they may discard information regarding the data distribution $p(\boldsymbol{x})$ when learning the task specific model $p(y|\boldsymbol{x})$. This could happen when certain features of the data are irrelevant for the predictive task, causing information loss regarding the data distribution $p(\boldsymbol{x})$. Moreover, existing methods to calibrate model uncertainty estimates assume access to OoD data during training (Lee et al., 2018; Hendrycks et al., 2019b). Although the OoD data may not come from the testing distribution, this approach assumes that the structure of OoD data is known ahead of time, which can be incorrect in settings such as active/online learning where new training distributions are regularly encountered.

On the other hand, model-independent techniques seek to estimate the likelihood of the data distribution $p(\boldsymbol{x})$. These techniques include Variational Autoencoders (VAEs) (Kingma & Welling, 2013), Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), autoregressive models (Germain et al., 2015; Oord et al., 2016), and invertible flow-based models (Dinh et al., 2016; Kingma & Dhariwal, 2018). Among these techniques, invertible models offer exact computation of the data likelihood, making them attractive for likelihood estimation. Moreover, they do not require OoD samples during training, making them applicable to any OoD detection scenario. Thus in this paper, we focus on searching for invertible flow-based architectures, though the presented techniques are also applicable to other likelihood estimation models.

Along this direction, recent work has discovered that likelihood-based models can assign higher likelihoods to OoD data compared to in-distribution data (Nalisnick et al., 2019a; Choi & Jang, 2018) (see Figure 13 of the supplementary material for an example). One hypothesis for such a phenomenon is that most data points lie within the typical set of a distribution, instead of the region of high likelihood (Nalisnick et al., 2019b). Thus, Nalisnick et al. (2019b) recommend to estimate the entropy using multiple data samples to screen out OoD data instead of using the likelihood. Other uncertainty quantification formulations can also be related to entropy estimation (Choi & Jang, 2018; Lakshminarayanan et al., 2017). However, it is not always realistic to test multiple data points in practical data streams, as testing data often come one sample at a time and are never well-organized into in-distribution or out-of-distribution groups.

With this in mind, model ensembling becomes a natural consideration to formulate entropy estimation. Instead of computing the entropy by averaging over multiple data points, model ensembles produce multiple estimates of the data likelihood, thus "augmenting" one data point into as many data points as needed to reliably estimate the entropy. However, care must be taken to ensure that the model ensemble produces likelihood estimates that agree with one another on in-distribution data, while also being diverse enough to discriminate OoD data likelihoods.

In what follows, we propose NADS as a method that can identify distributions of architectures for uncertainty quantification. Using a loss function that accounts for the diversity of architectures within the distribution, NADS allows us to construct an ensemble of models that can reliably detect OoD data.

3. Neural Architecture Distribution Search

Putting Neural Architecture Distribution Search (NADS) under a common NAS framework (Elsken et al., 2018), we break down our search formulation into three main components: the proxy task, the search space, and the optimization method. Specifying these components for NADS with the ultimate goal of uncertainty quantification for OoD detection is not immediately obvious. For example, naively using data likelihood maximization as a proxy task would run into the issues pointed out by Nalisnick et al. (2019a), with models assigning higher likelihoods to OoD data. On the other hand, the search space needs to be large enough to include a diverse range of architectures, yet still allowing a search algorithm to traverse it efficiently. In the following sections, we motivate our decision on these three choices and describe these components for NADS in detail.

3.1. Proxy Task

The first component of NADS is the training objective that guides the neural architecture search. Different from existing NAS methods, our aim is to derive an ensemble of deep models to improve model uncertainty quantification and OoD detection. To this end, instead of searching for architectures that maximize the likelihood of in-distribution data, which tends to cause models to incorrectly assign high likelihoods to OoD data, we instead seek architectures that can perform entropy estimation by maximizing the Widely Applicable Information Criteria (WAIC) of the training data. The WAIC score is a Bayesian adjusted metric to calculate the marginal likelihood (Watanabe, 2013). This metric has been shown by Choi & Jang (2018) to be robust towards the pitfall causing likelihood estimation models to assign high likelihoods to OoD data. The score is defined as follows:

$$\begin{aligned}\text{WAIC}(X) = \mathbb{E}_{\alpha\sim p(\alpha)}\Big[\mathbb{E}_{p(\boldsymbol{x})}[\log p(\boldsymbol{x}|\alpha)]\Big] \\ - \mathbb{V}_{\alpha\sim p(\alpha)}\Big[\mathbb{E}_{p(\boldsymbol{x})}[\log p(\boldsymbol{x}|\alpha)]\Big]\end{aligned} \quad (1)$$

Here, $\mathbb{E}[\cdot]$ and $\mathbb{V}[\cdot]$ denote expectation and variance respec-

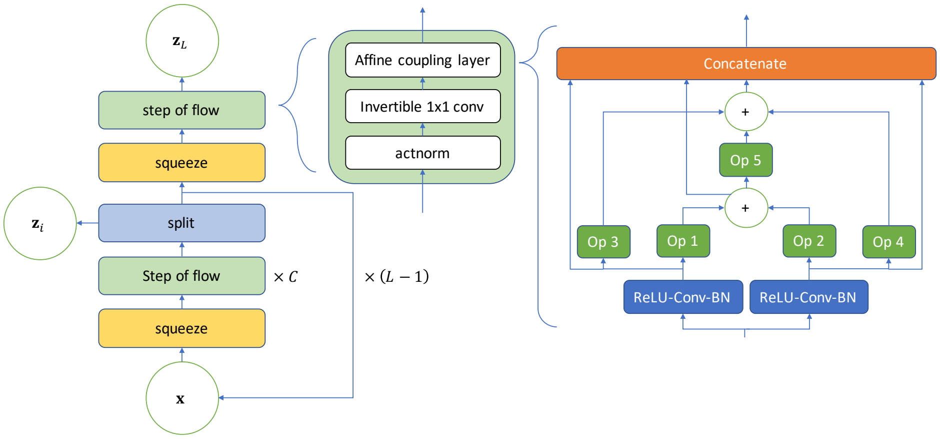


Figure 1. Search space of a single block in the architecture

tively, which are taken over all architectures α sampled from the posterior architecture distribution $p(\alpha)$. Such a strategy captures model uncertainty in a Bayesian fashion, improving OoD detection while also converging to the true data likelihood as the number of data points increases (Gelman et al., 2014). Intuitively, minimizing the variance of training data likelihoods allows its likelihood distribution to remain tight which, by proxy, minimizes the overlap of in-distribution and out-of-distribution likelihoods, thus making them separable.

Under this objective function, we search for an optimal distribution of network architectures $p(\alpha)$ by deriving the corresponding parameters that characterize $p(\alpha)$. Because the score requires aggregating the results from multiple architectures α, optimizing such a score using existing search methods can be intractable, as they typically only consider a single architecture at a time. Later, we will show how to circumvent this problem in our optimization formulation.

3.2. Search Space

NADS constructs a layer-wise search space with a pre-defined macro-architecture, where each layer can have a different architecture component. Such a search space has been studied by Zoph & Le (2016); Liu et al. (2018b); Real et al. (2019), where it shows to be both expressive and scalable/efficient.

The macro-architecture closely follows the Glow architecture presented in Kingma & Dhariwal (2018). Here, each layer consists of an actnorm, an invertible 1×1 convolution, and an affine coupling layer. Instead of pre-defining the affine coupling layer, we allow it to be optimized by our architecture search. The search space can be viewed in Figure 1. Here, each operational block of the affine coupling layer is selected from a list of candidate operations that include 3×3 average pooling, 3×3 max pooling, skip-connections, 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, identity, and zero. We choose this search space to answer the following questions towards better architectures for OoD detection:

- What topology of connections between layers is best for uncertainty quantification? Traditional likelihood estimation architectures focus only on feedforward connections without adding any skip-connection structures. However, adding skip-connections may improve optimization speed and stability.
- Are more features/filters better for OoD detection? More feature outputs of each layer should lead to a more expressive model. However, if many of those features are redundant, it may slow down learning, overfitting nuisances and resulting in sub-optimal models.
- Which operations are best for OoD detection? Intuitively, operations such as max/average pooling should not be preferred, as they discard information of the original data point "too aggressively". However, this intuition remains to be confirmed.

3.3. Optimization

Having specified our proxy task and search space, we now describe our optimization method for NADS. Specifically, let $\mathcal{A}$ denote our discrete architecture search space and $\alpha \in \mathcal{A}$ be an architecture in this space. Let $l_{\theta^*}(\alpha)$ be the loss function of architecture α with its parameters set to θ^* such that it satisfies $\theta^* = \arg\min_\theta l(\theta|\alpha)$ for some loss function $l(\cdot)$. We are interested in finding a distribution $p_\phi(\alpha)$ parameterized by ϕ that minimizes the expected loss of an architecture α sampled from it. We denote this loss function as $L(\phi) = \mathbb{E}_{\alpha \sim p_\phi(\alpha)}[l_{\theta^*}(\alpha)]$. For our NADS, this loss function is the negative WAIC score of in-distribution data $L(\phi) = -\text{WAIC}(X)$.

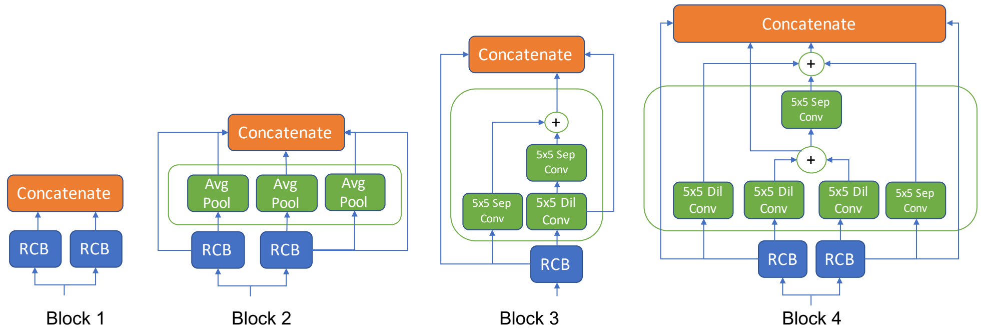


Figure 2. Summary of our architecture search findings: the most likely architecture structure for each block K found by NADS.

Several difficulties arise when naively attempting to optimize this setup. Firstly, the objective function involves computing an expectation and variance over all possible discrete architectures. We alleviate this problem by approximating the WAIC objective through Monte Carlo sampling. Specifically, we can sample M architectures from $p_\phi(\alpha)$ and approximate the WAIC score expectation and variance terms as

$$\mathrm{WAIC}(X) \approx \sum_{i=1}^{N} \left[\sum_{j=1}^{M} \log p(\boldsymbol{x}_i|\alpha_j) - \left(\sum_{j=1}^{M} (\log p(\boldsymbol{x}_i|\alpha_j))^2 - \left(\sum_{j=1}^{M} \log p(\boldsymbol{x}_i|\alpha_j) \right)^2 \right) \right] \tag{2}$$

Despite this approximation, optimizing (2) with respect to $p_\phi(\alpha)$, a distribution over high-dimensional discrete random variables α, is still intractable, as we would still need to search for the optimal network parameters for each newly sampled architecture. To circumvent this, we utilize a continuous relaxation for the discrete search space, allowing us to approximately optimize the discrete architectures through backpropagation and weight sharing between common architecture blocks, as similarly implemented by Xie et al. (2018) and Chang et al. (2019). Other potential possibilities for directly optimizing the discrete variables (Yin et al., 2019; Dadaneh et al., 2020b;a) are prohibitively computationally expensive for our setup.

For clarity of exposition, we first focus on sampling an architecture with a single hidden layer. In this setting, we intend to find a probability vector $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_K]$ with which we randomly pick a single operation from a list of K different operations $[o_1, \ldots, o_K]$. Let $\boldsymbol{b} = [b_1, \ldots, b_K]$ denote the random categorical indicator vector sampled from $\boldsymbol{\phi}$, where b_i is 1 if the i^{th} operation is chosen, and zero otherwise. Note that $\boldsymbol{b}$ is equivalent to the discrete architecture variable α in this setting. With this, we can write the random output $\boldsymbol{y}$ of the hidden layer given input $\boldsymbol{x}$ as

$$\boldsymbol{y} = \sum_{i=1}^{K} b_i \cdot o_i(\boldsymbol{x}).$$

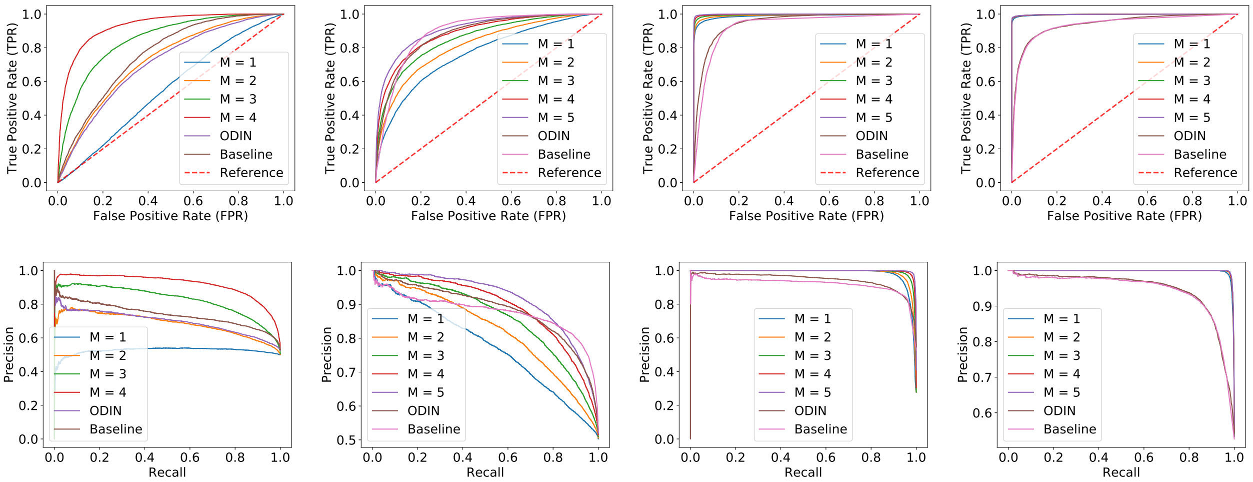
To make optimization tractable, we relax the discrete mask $\boldsymbol{b}$ to be a continuous random variable $\tilde{\boldsymbol{b}}$ using the Gumbel-Softmax reparameterization (Gumbel, 1954; Maddison et al., 2014) as follows:

$$\tilde{b}_i = \frac{\exp((\log(\phi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\phi_i) + g_i)/\tau)} \quad \text{for} \quad i = 1, \ldots, K.$$

Here, $g_1 \ldots g_k \sim -\log(-\log(u))$ where $u \sim \mathrm{Unif}(0,1)$, and τ is a temperature parameter. For low values of τ, $\tilde{\boldsymbol{b}}$ approaches a sample of a categorical random variable, recovering the original discrete problem. While for high values, $\tilde{\boldsymbol{b}}$ will equally weigh the K operations (Jang et al., 2016). Using this, we can compute backpropagation by approximating the gradient of the discrete architecture α with the gradient of the continuously relaxed categorical random variable $\tilde{\boldsymbol{b}}$, as $\nabla_{\theta,\phi}\alpha = \nabla_{\theta,\phi}\boldsymbol{b} \approx \nabla_{\theta,\phi}\tilde{\boldsymbol{b}}$. With this backpropagation gradient defined, generalizing the above setting to architectures with multiple layers simply involves recursively applying the above gradient relaxation to each layer. We can gradually remove the continuous relaxation and sample discrete architectures by annealing the temperature parameter τ, allowing us to perform architecture search without using a validation set.

3.4. Search Results

We applied our architecture search on five datasets: CelebA (Liu et al.), CIFAR-10, CIFAR-100, (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and MNIST (LeCun). In all experiments, we used the Adam optimizer with a fixed learning rate of 1×10^{-5} with a batch size of 4 for 10000 iterations. We approximate the WAIC score using $M = 4$ architecture samples, and set the temperature parameter $\tau = 1.5$. The number of layers and latent dimensions is the same as in the original Glow architecture (Kingma & Dhariwal, 2018), with 4 blocks and 32



(a) CIFAR100-CIFAR10 (b) CIFAR10-CIFAR100 (c) SVHN-CIFAR10 (d) MNIST-KMNIST

Figure 3. ROC and PR curve comparison of the most challenging evaluation setups for our NADS ensemble. Here, 'Baseline' denotes the method proposed by (Hendrycks & Gimpel, 2016). Subcaptions denote training-testing set pairs. Additional figures are provided in Section G of the supplementary material.

flows per block. Images were resized to 64×64 as inputs to the model. With this setup, we found that we are able to identify neural architectures in less than 1 GPU day on an Nvidia RTX 2080 Ti graphics card.

Our findings are summarized in Figure 2, while more samples from our architecture search can be seen in Section C of the supplementary material. Observing the most likely architecture components found on all of the datasets, a number of notable observations can be made:

- The first few layers have a simple feedforward structure, with either only a few convolutional operations or average pooling operations. On the other hand, more complicated structures with skip connections are preferred in the deeper layers of the network. We hypothesize that in the first few layers, simple feature extractors are sufficient to represent the data well. Indeed, recent work on analyzing neural networks for image data have shown that the first few layers have filters that are very similar to SIFT features or wavelet bases (Zeiler & Fergus, 2014; Lowe, 1999).
- The max pooling operation is almost never selected by the architecture search. This confirms our hypothesis that operations that discard information about the data is unsuitable for OoD detection. However, to our surprise, average pooling is preferred in the first layers of the network. We hypothesize that average pooling has a less severe effect in discarding information, as it can be thought of as a convolutional filter with uniform weights.
- The deeper layers prefer a more complicated structure, with some components recovering the skip connection structure of ResNets (He et al., 2016). We hypothesize that deeper layers may require more skip connections in order to feed a strong signal for the first few layers. This increases the speed and stability of training. Moreover, a larger number of features can be extracted using the more complicated architecture.

Interestingly enough, we found that the architectures that we sample from our NADS perform well in image generation without further retraining, as shown in Section D of the supplementary material.

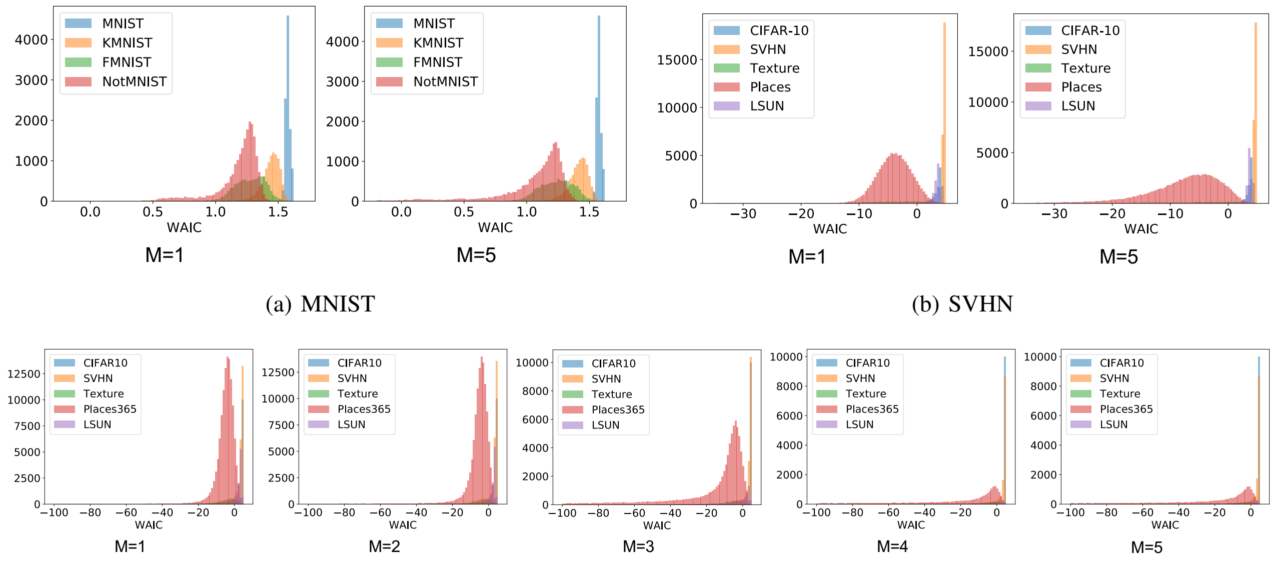
4. Bayesian Model Ensemble of Neural Architectures

4.1. Model Ensemble Formulation

Using the architectures sampled from our search, we create a Bayesian ensemble of models to estimate the WAIC score. Each model of our ensemble is weighted according to its probability as in Hoeting et al. (1999). The log-likelihood estimate as well as the variance of this model ensemble is given as follows:

$$\begin{aligned}\mathbb{E}_{\alpha\sim p_\phi(\alpha)}[\log p(\boldsymbol{x})] &= \sum_{\alpha\in\mathcal{A}} p_\phi(\alpha)\log p(\boldsymbol{x}|\alpha)\\ &\approx \sum_{i=1}^{M}\frac{p_\phi(\alpha_i)}{\sum_{j=1}^{M}p_\phi(\alpha_j)}\log p(\boldsymbol{x}|\alpha_i)\end{aligned}$$

$$\begin{aligned}\mathbb{V}_{\alpha\sim p_\phi(\alpha)}[\log p(\boldsymbol{x})] \approx \sum_{i=1}^{M}\frac{p_\phi(\alpha_i)}{\sum_{j=1}^{M}p_\phi(\alpha_j)}\Big(&\mathbb{V}[\log p(\boldsymbol{x}|\alpha_i)]\\ &+(\log p(\boldsymbol{x}|\alpha_i))^2\Big) - \mathbb{E}_{\alpha\sim p_\phi(\alpha)}[\log p(\boldsymbol{x})]^2\end{aligned}$$



(a) MNIST

(b) SVHN

(c) CIFAR-10

Figure 4. Effect of ensemble size to the distribution of WAIC scores estimated by model ensembles trained on different datasets. Larger ensemble sizes causes the WAIC score likelihood estimate of OoD data to be lower. Additional histograms for different ensemble sizes in Section F of the supplementary material are with higher resolution.

Intuitively, we are weighing each member of the ensemble by their posterior architecture distribution $p_\phi(\alpha)$, a measure of how likely each architecture is in optimizing the WAIC score. We note that for our setup, $\mathbb{V}[\log p_{\alpha_i}(x)]$ is zero for each model in our ensemble; however, for models which do have variance estimates, such as models that incorporate variational dropout (Gal et al., 2017; Boluki et al., 2020; Kingma et al., 2015; Gal & Ghahramani, 2016), this term may be nonzero. Using these estimates, we are able to approximate the WAIC score in Equation (1).

4.2. Ensemble Results

We trained our proposed method on 4 datasets D_{in}: CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), and MNIST (LeCun). In all experiments, we randomly sampled an ensemble of $M = 5$ models from the posterior architecture distribution $p_{\phi^*}(\alpha)$ found by NADS. We then retrained each architecture for 150000 iterations using Adam with a learning rate of 1×10^{-5}.

We first show the effects of increasing the ensemble size in Figure 4 and Section F of the supplementary material. Here, we can see that increasing the ensemble size causes the OoD WAIC scores to decrease as their corresponding histograms shift away from the training data WAIC scores, thus improving OoD detection performance. Next, we compare our ensemble search method against a traditional ensemble method that uses a single Glow (Kingma & Dhariwal, 2018) architecture trained with multiple random initializations. We find that our method is superior for OoD detection compared to the traditional ensemble method, as shown in Table 2 of the supplementary material.

We evaluate our NADS ensemble OoD detection method for screening out samples from datasets that the original model was not trained on (D_{out}). For SVHN, we used the Texture, Places, LSUN, and CIFAR-10 as the OoD dataset. For CIFAR-10 and CIFAR-100, we used the SVHN, Texture, Places, LSUN, CIFAR-100 (CIFAR-10 for CIFAR-100) datasets, as well as the Gaussian and Rademacher distributions as the OoD dataset. Finally, for MNIST, we used the not-MNIST, F-MNIST, and K-MNIST datasets. We compared our method against a baseline method that uses maximum softmax probability (MSP) (Hendrycks & Gimpel, 2016), as well as two popular OoD detection methods: ODIN (Liang et al., 2017) and Outlier Exposure (OE) (Hendrycks et al., 2019b).

ODIN attempts to calibrate the uncertainty estimates of an existing model by reweighing its output softmax score using a temperature parameter and through random perturbations of the input data. For this, we use DenseNet as the base model as described in Liang et al. (2017). On the other hand, OE models are trained to minimize a loss regularized by an outlier exposure loss term, a loss term that requires access to OoD samples, although they are not required to be from the tested OoD distribution.

We also show the improvements made by our design of the search space and the optimization objective by comparing

Table 1. OoD detection results on various evaluation setups. We compared our method with MSP (Baseline) (Hendrycks & Gimpel, 2016), NAS following the DARTS search design (DARTS) (Liu et al., 2018b), and Outlier Exposure (OE) (Hendrycks et al., 2019b).

D_{in}	D_{out}	FPR% at TPR 95%				AUROC%				AUPR%			
		Base	DARTS	OE	Ours	Base	DARTS	OE	Ours	Base	DARTS	OE	Ours
MNIST	not-MNIST	10.3	23.07	0.25	**0.00**	97.2	94.62	99.86	**100**	97.4	96.81	99.86	**100**
	F-MNIST	61.1	8.29	0.99	**0.00**	88.8	97.59	99.83	**100**	90.8	97.06	99.83	**100**
	K-MNIST	29.6	9.37	**0.03**	0.76	93.6	97.39	97.60	**99.80**	94.3	96.90	97.05	**99.84**
SVHN	Texture	33.9	23.43	1.04	**0.07**	89.3	94.25	**99.75**	99.26	86.8	80.98	**99.09**	97.75
	Places365	22.2	16.17	0.02	**0.00**	92.8	95.74	**99.99**	**99.99**	99.7	99.57	**99.99**	**99.99**
	LSUN	26.8	16.16	0.05	**0.02**	88.2	95.44	99.98	**99.99**	90.4	87.36	99.95	**99.99**
	CIFAR10	23.2	16.82	3.11	**0.37**	91.1	95.36	99.26	**99.92**	91.9	87.45	97.88	**99.83**
CIFAR10	SVHN	30.5	19.47	**8.41**	17.05	89.5	93.58	**98.20**	97.65	94.9	96.25	97.97	**99.07**
	Texture	39.8	24.25	14.9	**0.25**	87.7	92.18	96.7	**99.81**	79.8	83.51	94.39	**99.86**
	Places365	36.0	41.64	19.07	**0.00**	88.1	87.65	95.41	**100**	99.5	99.42	95.32	**100**
	LSUN	14.6	30.02	15.20	**0.44**	95.4	90.11	96.43	**99.83**	96.1	86.88	96.01	**99.89**
	CIFAR100	33.1	35.72	**26.59**	36.36	88.7	88.43	**92.93**	91.23	87.7	72.95	**92.13**	91.60
	Gaussian	6.3	11.67	0.7	**0.00**	97.7	95.55	99.6	**100**	93.6	87.46	94.3	**100**
	Rademacher	6.9	10.73	0.5	**0.00**	96.9	95.26	99.8	**100**	89.7	84.10	97.4	**100**
CIFAR100	SVHN	46.2	53.81	**42.9**	45.92	82.7	79.30	86.9	**94.35**	91.3	88.52	80.21	**96.01**
	Texture	74.3	62.49	55.97	**0.42**	72.6	75.00	84.23	**99.76**	60.1	57.77	75.76	**99.81**
	Places365	63.2	64.91	57.77	**0.012**	76.2	75.72	82.65	**99.99**	98.9	98.78	81.47	**99.99**
	LSUN	69.4	56.01	57.5	**38.85**	83.7	77.57	83.4	**90.65**	70.1	72.94	77.85	**90.61**
	CIFAR10	62.5	61.62	59.96	**45.62**	75.8	76.15	77.53	**83.27**	74.0	71.41	72.82	**81.48**
	Gaussian	29.3	26.70	12.1	**0.00**	86.5	87.82	95.7	**100**	66.1	69.05	71.1	**100**
	Rademacher	59.4	16.19	17.1	**0.00**	51.7	92.05	93.0	**100**	32.7	73.02	56.9	**100**

our method to applying architecture search without taking these factors into consideration. To do this, we applied neural architecture search with the goal of maximizing classification accuracy on in-distribution data. Here, our search formulation closely follows the Differentiable Architecture Search (DARTS) method (Liu et al., 2018b). After identifying the optimal architecture, we screen out OoD data using the maximum softmax probability (MSP) (Hendrycks & Gimpel, 2016), a score that gives classification architectures the ability to screen out OoD data.

As shown in Tables 1 and 3 in the supplementary material, our method outperforms the baseline MSP and ODIN significantly while performing better or comparably with OE, which requires OoD data during training, albeit not from the testing distribution. Notably, our method was able to achieve an improvement of 57% FPR on the CIFAR100 – Places365 setup compared to OE. Comparing the original architecture used by MSP and the identified architecture by DARTS, we can see that there is an improvement in OoD detection performance, however, because the architectures are not tailored to perform OoD detection, our NADS was also able to outperform it in our experiments.

We plot Receiver Operating Characteristic (ROC) and Precision-Recall (PR) curves in Figure 3 and Section G of the supplementary material for more comprehensive comparison. In particular, our method consistently achieves high area under PR curve (AUPR%), showing that we are especially capable of screening out OoD data in settings where their occurrence is rare. Such a feature is important in situations where anomalies are sparse, yet have disastrous consequences. Notably, ODIN underperforms in screening out many OoD datasets, despite being able to reach the original reported performance when testing on LSUN using a CIFAR10 trained model. This suggests that ODIN may not be stable for use on different anomalous distributions.

5. Conclusion

Unlike NAS for common learning tasks, specifying a model and an objective to optimize for uncertainty estimation and outlier detection is not straightforward. Moreover, using a single model may not be sufficient to accurately quantify uncertainty and successfully screen out OoD data. We developed a novel neural architecture distribution search (NADS) formulation to identify a random ensemble of architectures that perform well on a given task. Instead of seeking to maximize the likelihood of in-distribution data which may cause OoD samples to be mistakenly given a higher likelihood, we developed a search algorithm to optimize the WAIC score, a Bayesian adjusted estimation of the data entropy. Using this formulation, we have identified several key features that make up good uncertainty quantification architectures, namely a simple structure in the shallower layers, use of information preserving operations, and a larger, more expressive structure with skip connections for deeper layers to ensure optimization stability. Using the architecture distribution learned by NADS, we then constructed an ensemble of models to estimate the data entropy using the WAIC score. We demonstrated the superiority of our method to existing OoD detection methods and showed that our method has highly competitive performance without requiring access to OoD samples. Overall, NADS as a new uncertainty-aware architecture search strategy enables model uncertainty quantification that is critical for more robust and generalizable deep learning, a crucial step in safely applying deep learning to healthcare, autonomous driving, and disaster response.

Acknowledgement

The presented materials are based upon the work supported by the National Science Foundation under Grants CCF-1553281, IIS-1812641, and CCF-1934904; and the Defense Advanced Research Projects Agency under grand FA8750-18-2-0027. We also thank Texas A&M High Performance Research Computing and Texas Advanced Computing Center for providing computational resources to perform experiments in this work.

References

Ahmed, K. and Torresani, L. Maskconnect: Connectivity learning by gradient descent. In *European Conference on Computer Vision*, pp. 362–378. Springer, 2018.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., and Mané, D. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*, 2016.

Ardywibowo, R., Huang, S., Gui, S., Xiao, C., Cheng, Y., Liu, J., and Qian, X. Switching-state dynamical modeling of daily behavioral data. *Journal of Healthcare Informatics Research*, 2(3):228–247, 2018.

Ardywibowo, R., Zhao, G., Wang, Z., Mortazavi, B., Huang, S., and Qian, X. Adaptive activity monitoring with uncertainty quantification in switching Gaussian process models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 266–275, 2019.

Baker, B., Gupta, O., Naik, N., and Raskar, R. Designing neural network architectures using reinforcement learning. In *International Conference on Learning Representations*, 2017.

Boluki, S., Ardywibowo, R., Dadaneh, S. Z., Zhou, M., and Qian, X. Learnable Bernoulli dropout for Bayesian deep learning. *arXiv preprint arXiv:2002.05155*, 2020.

Chang, J., Zhang, X., Guo, Y., Meng, G., Xiang, S., and Pan, C. Differentiable architecture search with ensemble Gumbel-Softmax. *arXiv preprint arXiv:1905.01786*, 2019.

Chen, L.-C., Collins, M., Zhu, Y., Papandreou, G., Zoph, B., Schroff, F., Adam, H., and Shlens, J. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in Neural Information Processing Systems*, pp. 8699–8710, 2018.

Choi, H. and Jang, E. Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

Dadaneh, S. Z., Boluki, S., Yin, M., Zhou, M., and Qian, X. Pairwise supervised hashing with Bernoulli variational auto-encoder and self-control gradient estimator. *arXiv preprint arXiv:2005.10477*, 2020a.

Dadaneh, S. Z., Boluki, S., Zhou, M., and Qian, X. Arsm gradient estimator for supervised learning to rank. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3157–3161, 2020b.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059, 2016.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in Neural Information Processing Systems*, pp. 3581–3590, 2017.

Gelman, A., Hwang, J., and Vehtari, A. Understanding predictive information criteria for bayesian models. *Statistics and computing*, 24(6):997–1016, 2014.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pp. 881–889, 2015.

Ghiasi, G., Lin, T.-Y., and Le, Q. V. NAS-FPN: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7036–7045, 2019.

Gong, X., Chang, S., Jiang, Y., and Wang, Z. AutoGAN: Neural architecture search for generative adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio,

Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

Gumbel, E. J. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33, 1954.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2016.

Hendrycks, D., Lee, K., and Mazeika, M. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019a.

Hendrycks, D., Mazeika, M., and Dietterich, T. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019b.

Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. Bayesian model averaging: A tutorial. *Statistical science*, pp. 382–401, 1999.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-Softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jin, H., Song, Q., and Hu, X. Auto-keras: Efficient neural architecture search with network morphism. *arXiv preprint arXiv:1806.10282*, 2018.

Kendall, A. and Gal, Y. What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pp. 5574–5584, 2017.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pp. 2575–2583, 2015.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pp. 6402–6413, 2017.

LeCun, Y. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lee, K., Lee, H., Lee, K., and Shin, J. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *International Conference on Learning Representations*, 2018.

Liang, S., Li, Y., and Srikant, R. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *European Conference on Computer Vision*, pp. 19–35. Springer, 2018a.

Liu, C., Chen, L.-C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., and Fei-Fei, L. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

Liu, H., Simonyan, K., and Yang, Y. DARTS: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Large-scale celebfaces attributes (celeba) dataset.

Lowe, D. G. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pp. 1150–1157, 1999.

Maddison, C. J., Tarlow, D., and Minka, T. A* sampling. In *Advances in Neural Information Processing Systems*, pp. 3086–3094, 2014.

Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? In *International Conference on Learning Representations*, 2019a.

Nalisnick, E., Matsukawa, A., Teh, Y. W., and Lakshminarayanan, B. Detecting out-of-distribution inputs to deep generative models using a test for typicality. *arXiv preprint arXiv:1906.02994*, 2019b.

Negrinho, R. and Gordon, G. Deeparchitect: Automatically designing and training deep architectures. *arXiv preprint arXiv:1704.08792*, 2017.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Nguyen, A., Yosinski, J., and Clune, J. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 427–436, 2015.

NHTSA. Tesla crash preliminary evaluation report. Technical report, U.S. Department of Transportation, National Highway Traffic Safety Administration, Jan 2017.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, pp. 4092–4101, 2018.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4780–4789, 2019.

Shafaei, A., Schmidt, M., and Little, J. J. A less biased evaluation of out-of-distribution sample detectors.

Watanabe, S. A widely applicable Bayesian information criterion. *Journal of Machine Learning Research*, 14 (Mar):867–897, 2013.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: Stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*, 2018.

Yin, M., Yue, Y., and Zhou, M. Arsm: Augment-reinforce-swap-merge estimator for gradient backpropagation through categorical variables. In *International Conference on Machine Learning*, pp. 7095–7104, 2019.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhong, Z., Yan, J., Wu, W., Shao, J., and Liu, C.-L. Practical block-wise neural network architecture generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2423–2432. IEEE, 2018.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710. IEEE, 2018.

A. Fixed Model Ablation Study

Table 2. OoD detection results on various training and testing experiments comparing our method with a baseline ensembling method that uses a fixed architecture trained multiple times with different random initializations.

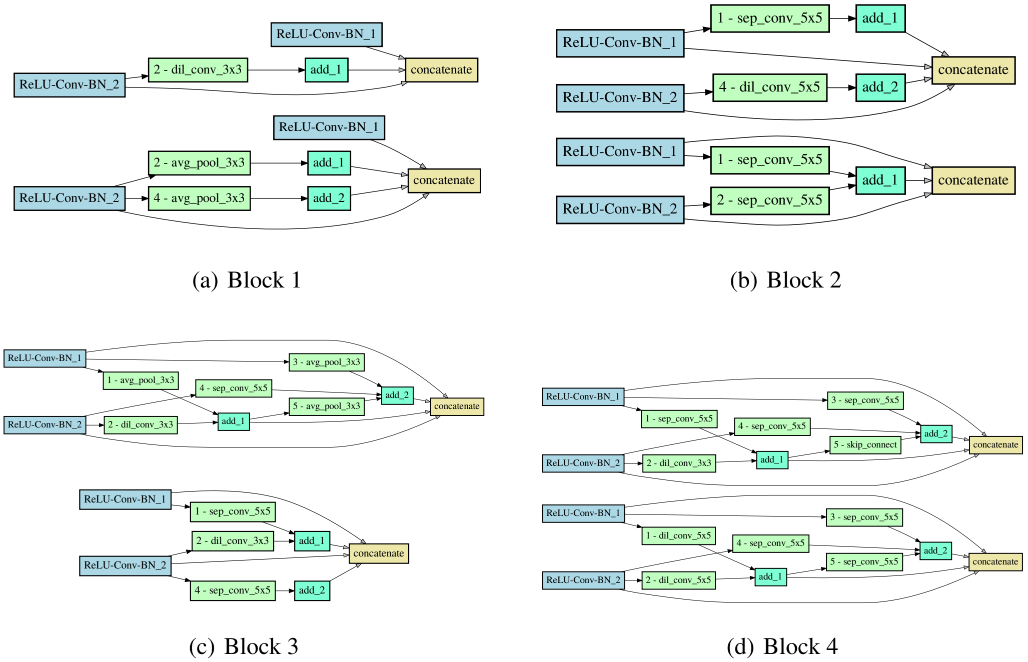
D_{in}	D_{out}	FPR% at TPR 95%		AUROC%		AUPR%	
		Base Ensemble	Ours	Base Ensemble	Ours	Base Ensemble	Ours
CIFAR10	SVHN	50.07	**17.05**	93.48	**97.65**	95.98	**99.07**
	Texture	6.22	**0.25**	97.68	**99.81**	97.44	**99.86**
	Places365	1.03	**0.00**	99.59	**100**	99.97	**100**
	LSUN	34.35	**0.44**	91.55	**99.83**	92.15	**99.89**
	CIFAR100	65.13	**36.36**	78.44	**91.23**	79.44	**91.60**
	Gaussian	**0.00**	**0.00**	**100**	**100**	**100**	**100**
	Rademacher	**0.00**	**0.00**	**100**	**100**	**100**	**100**

B. OoD Detection Performance Comparison with ODIN

Table 3. OoD detection results on various training and testing experiments comparing our method with ODIN (Liang et al., 2017).

D_{in}	D_{out}	FPR% at TPR 95%		AUROC%		AUPR%	
		ODIN	Ours	ODIN	Ours	ODIN	Ours
MNIST	not-MNIST	8.7	**0.00**	98.2	**100**	98.0	**100**
	F-MNIST	65	**0.00**	88.6	**100**	90.5	**100**
	K-MNIST	36.5	**0.76**	94.0	**99.80**	94.6	**99.84**
SVHN	Texture	33.9	**0.07**	92.4	**99.26**	88.2	**97.75**
	Places365	22.2	**0.00**	94.9	**99.99**	99.8	**99.99**
	LSUN	26.8	**0.02**	93.5	**99.99**	93.1	**99.99**
	CIFAR10	21.6	**0.37**	94.8	**99.92**	94.4	**99.83**
CIFAR10	SVHN	36.5	**17.05**	89.7	**97.65**	95.6	**99.07**
	Texture	76.2	**0.25**	81.4	**99.81**	76.7	**99.86**
	Places365	44.0	**0.00**	89.0	**100**	99.6	**100**
	LSUN	3.9	**0.44**	99.2	**99.83**	99.2	**99.89**
	CIFAR100	45.4	**36.36**	88.3	**91.23**	88.5	**91.60**
	Gaussian	0.1	**0.00**	**100**	**100**	99.9	**100**
	Rademacher	0.3	**0.00**	99.9	**100**	99.8	**100**
CIFAR100	SVHN	**32.8**	45.92	90.3	**94.35**	95.3	**96.01**
	Texture	78.9	**0.42**	75.7	**99.76**	64.5	**99.81**
	Places365	63.3	**0.012**	79.0	**99.99**	99.1	**99.99**
	LSUN	**17.6**	38.85	**96.8**	90.65	**96.5**	90.61
	CIFAR10	78.2	**45.62**	70.6	**83.27**	69.7	**81.48**
	Gaussian	1.3	**0.00**	99.5	**100**	97.8	**100**
	Rademacher	13.8	**0.00**	92.7	**100**	75.0	**100**

C. Additional Sample Architectures



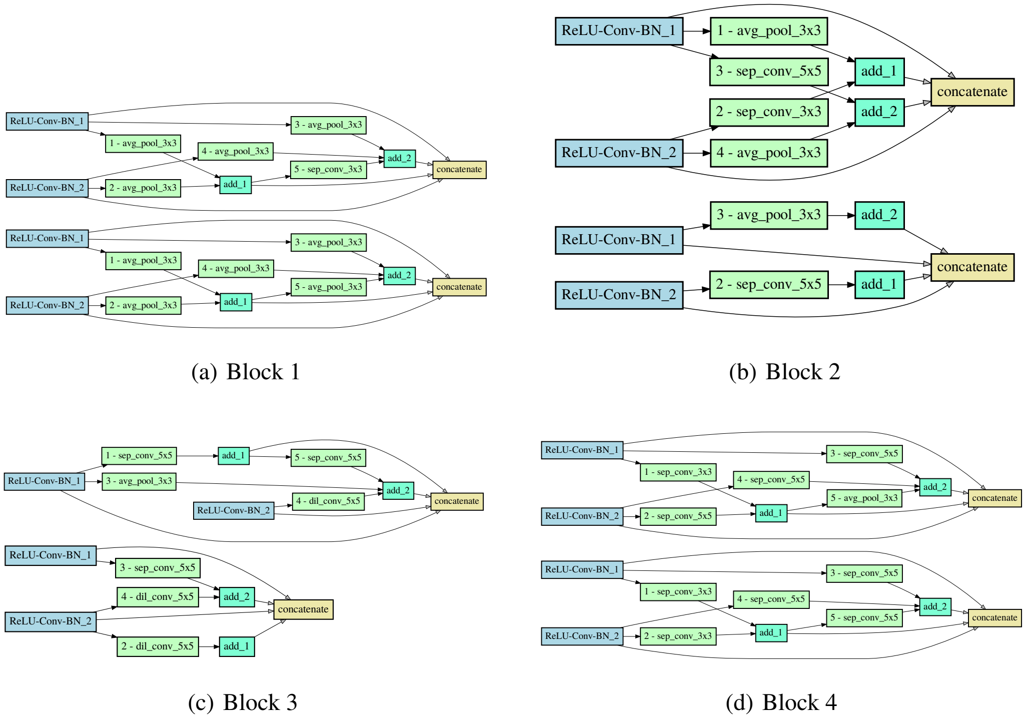
(a) Block 1

(b) Block 2

(c) Block 3

(d) Block 4

Figure 5. Maximum likelihood architectures inferred by our search algorithm on CelebA. Shown are two samples taken from each block.



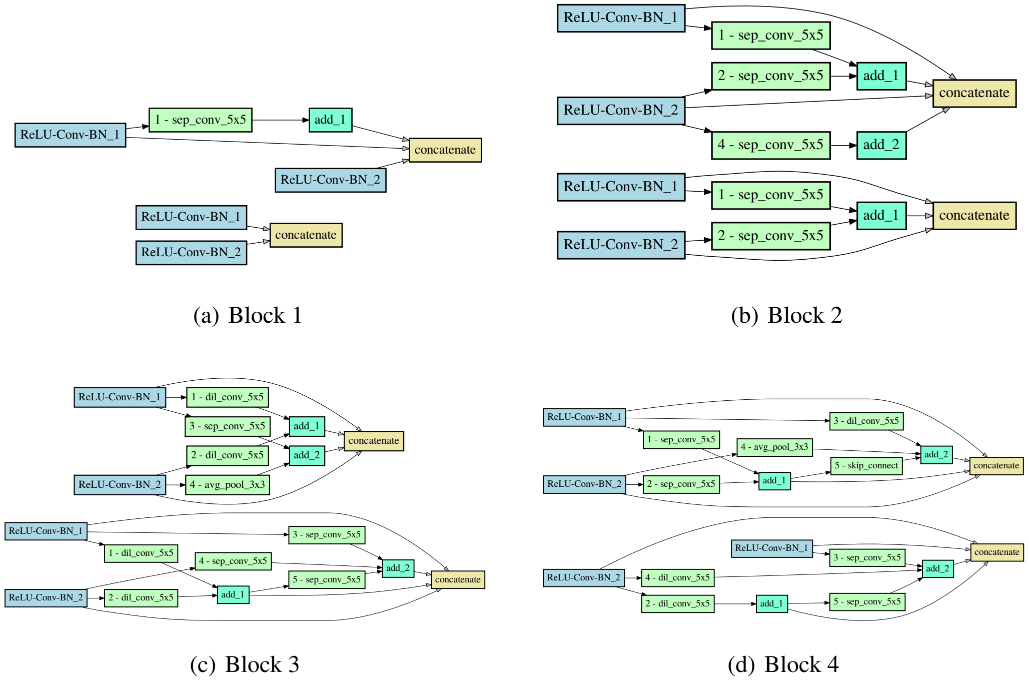
(a) Block 1

(b) Block 2

(c) Block 3

(d) Block 4

Figure 6. Maximum likelihood architectures inferred by our search algorithm on MNIST. Shown are two samples taken from each block.



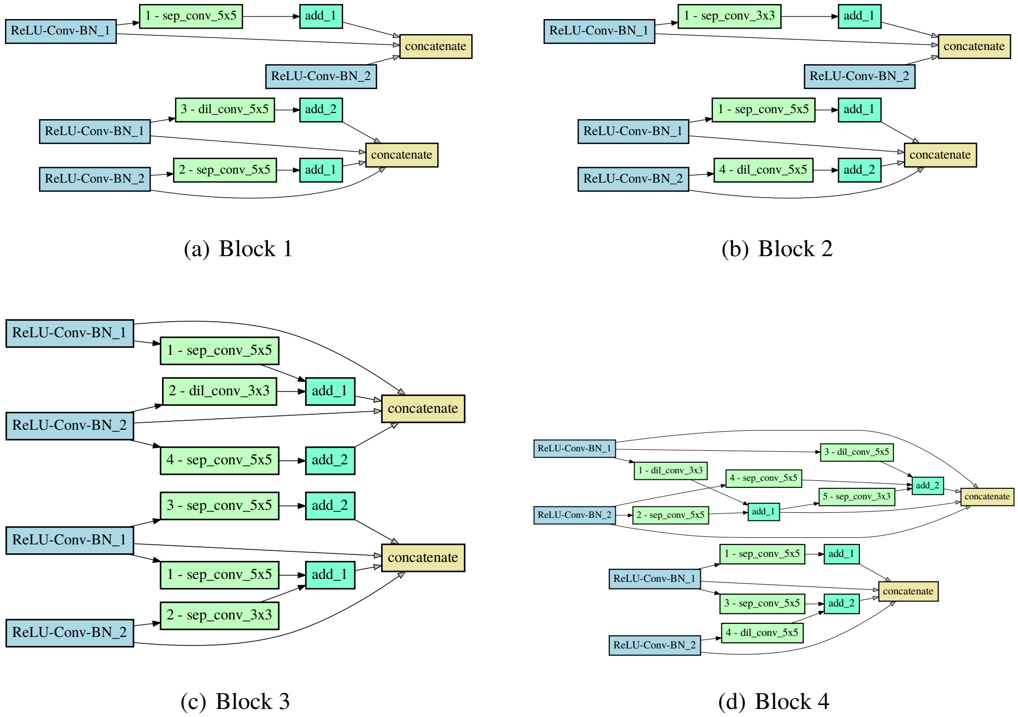
(a) Block 1

(b) Block 2

(c) Block 3

(d) Block 4

Figure 7. Maximum likelihood architectures inferred by our search algorithm on SVHN. Shown are two samples taken from each block.



(a) Block 1

(b) Block 2

(c) Block 3

(d) Block 4

Figure 8. Maximum likelihood architectures inferred by our search algorithm on CIFAR-10. Shown are two samples taken from each block.

D. Image generation samples

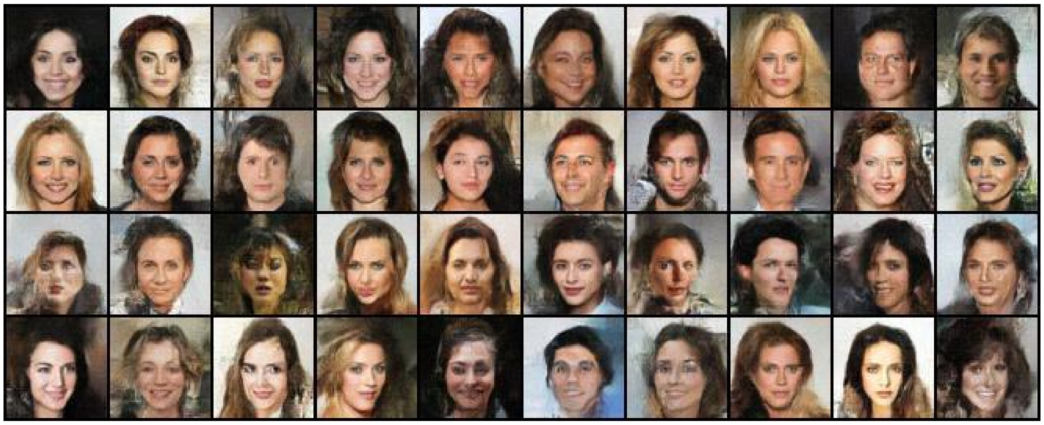


Figure 9. Samples taken from randomly sampled NADS architectures searched on CelebA. Images were not cherry-picked and the architectures were sampled without further retraining.

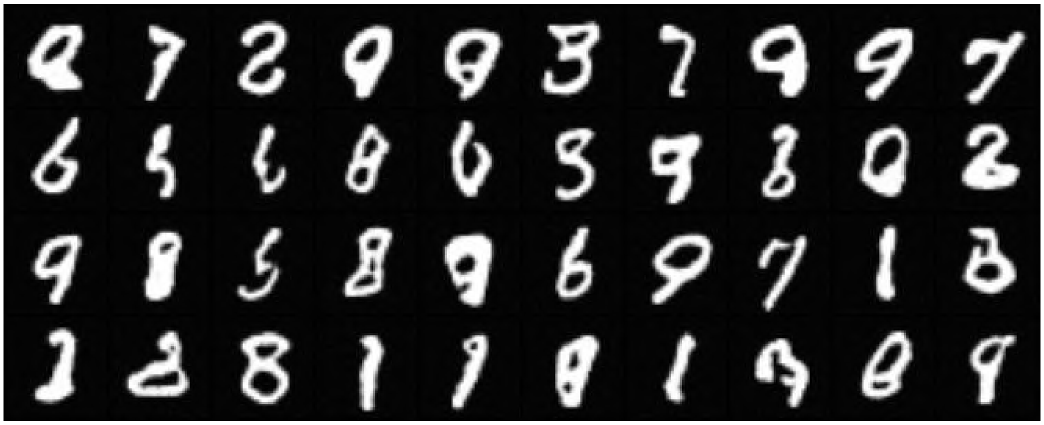


Figure 10. Samples taken from randomly sampled NADS architectures searched on MNIST. Images were not cherry-picked and the architectures were sampled without further retraining.

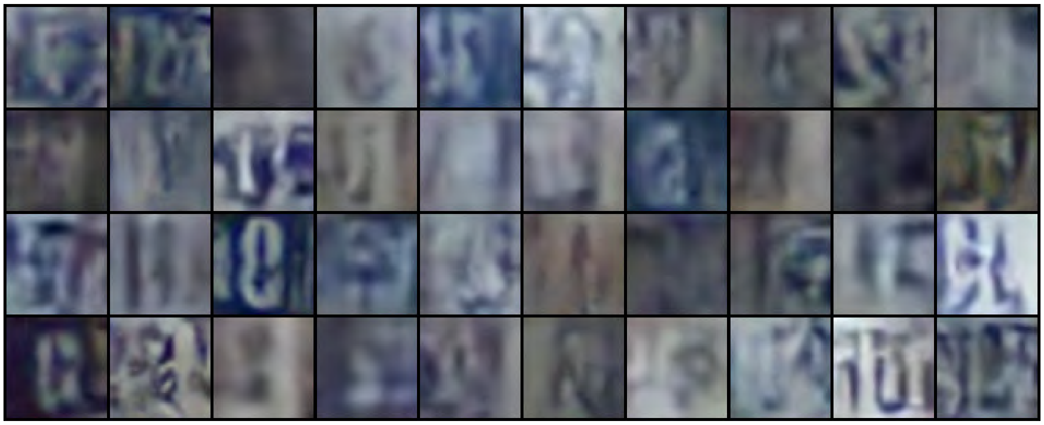


Figure 11. Samples taken from randomly sampled NADS architectures searched on SVHN. Images were not cherry-picked and the architectures were sampled without further retraining.

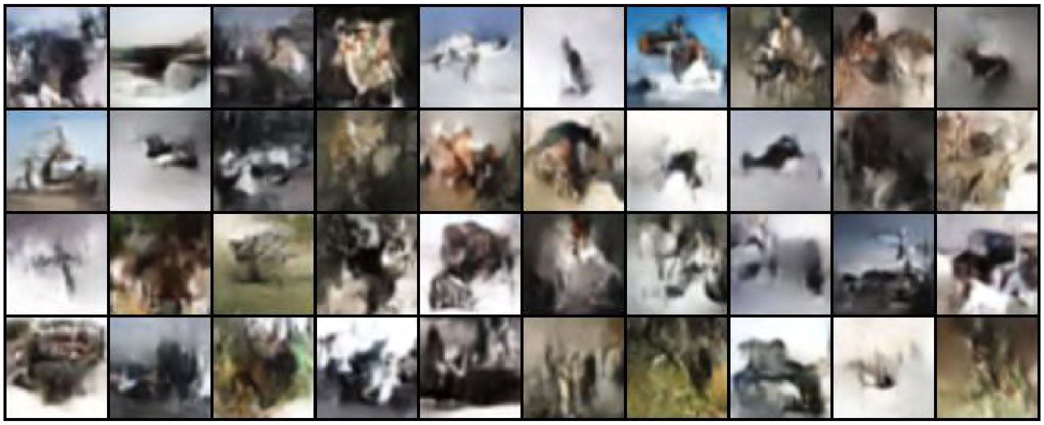


Figure 12. Samples taken from randomly sampled NADS architectures searched on CIFAR-10. Images were not cherry-picked and the architectures were sampled without further retraining.

E. Likelihood Estimation Models Assign Higher Likelihood to OoD Data

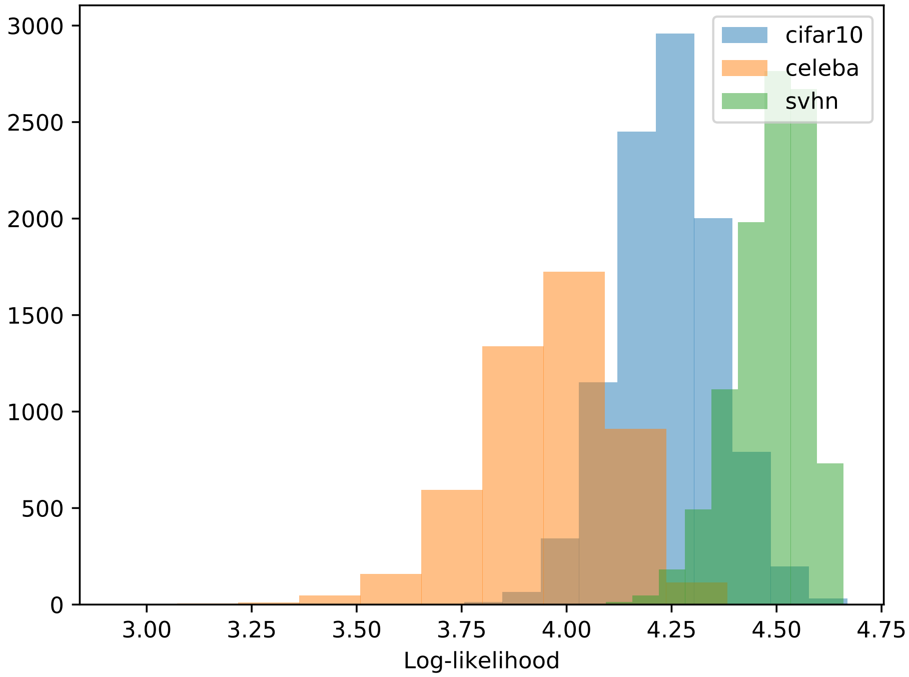
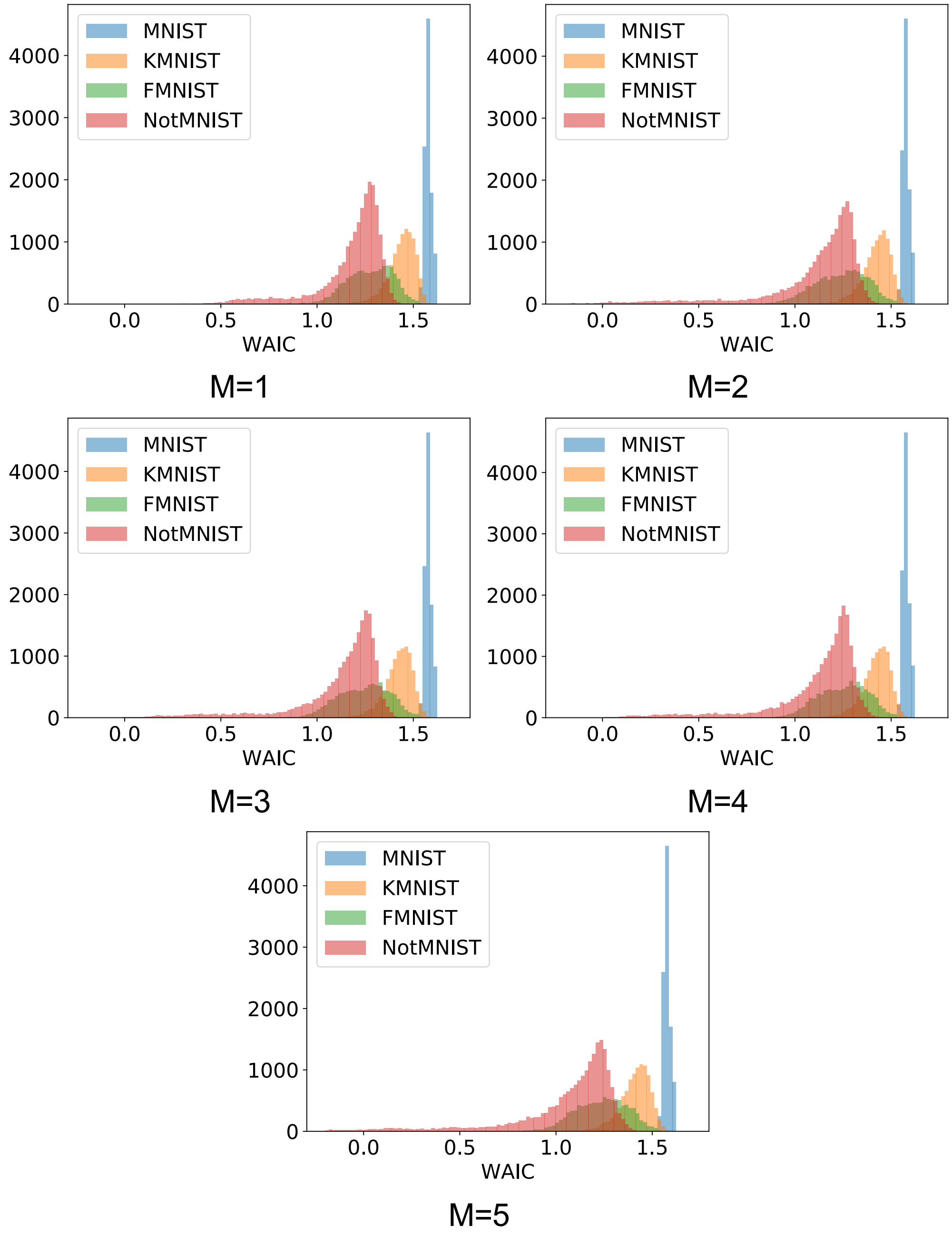


Figure 13. Likelihood distributions of different datasets evaluated on a Glow model trained on CelebA. The model assigns higher likelihood to OoD samples from CIFAR-10 and SVHN.

F. Effect of Ensemble Size



M=1

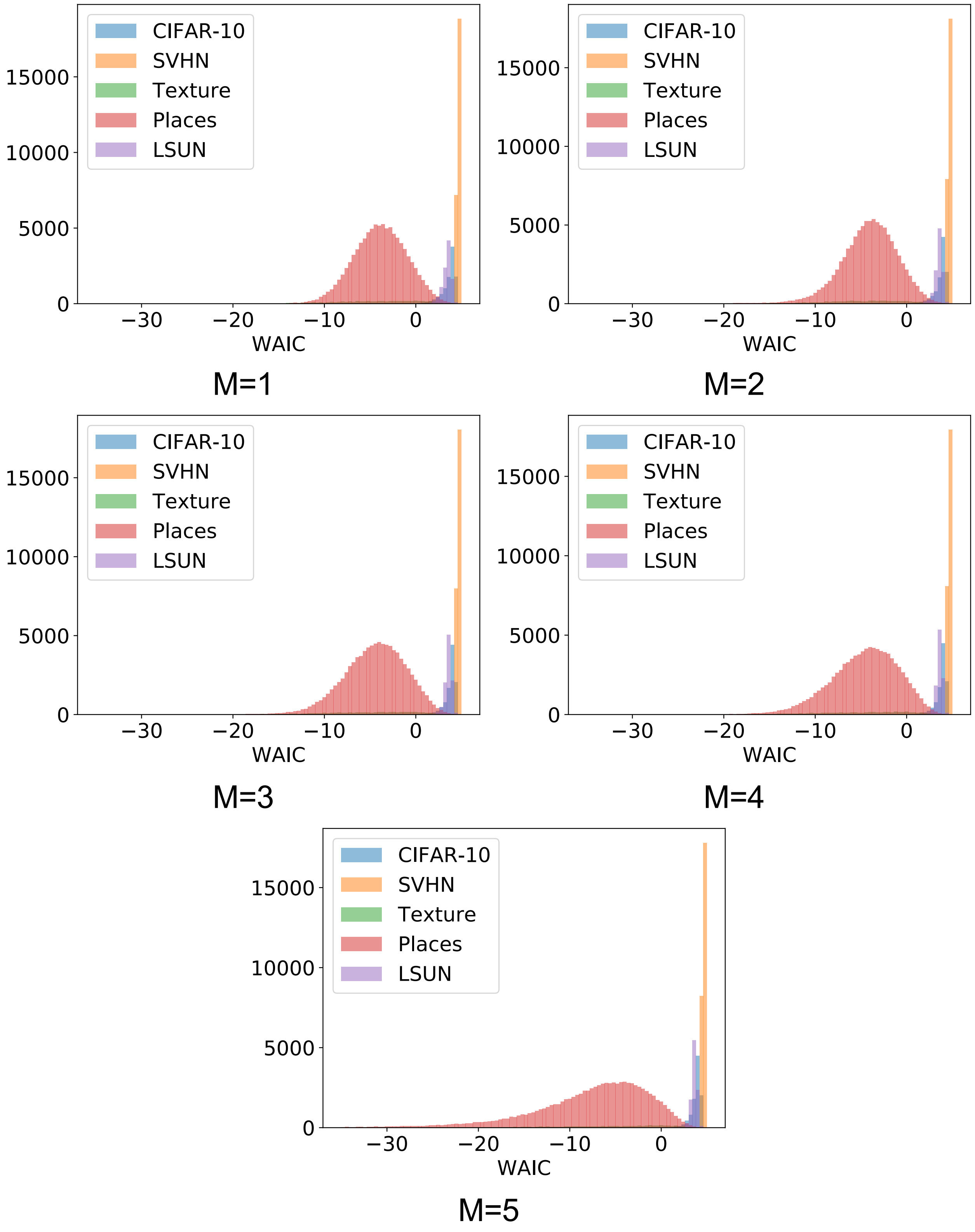
M=2

M=3

M=4

M=5

Figure 14. Effect of ensemble size to the distribution of WAIC scores estimated by model ensembles trained on MNIST.



M=1

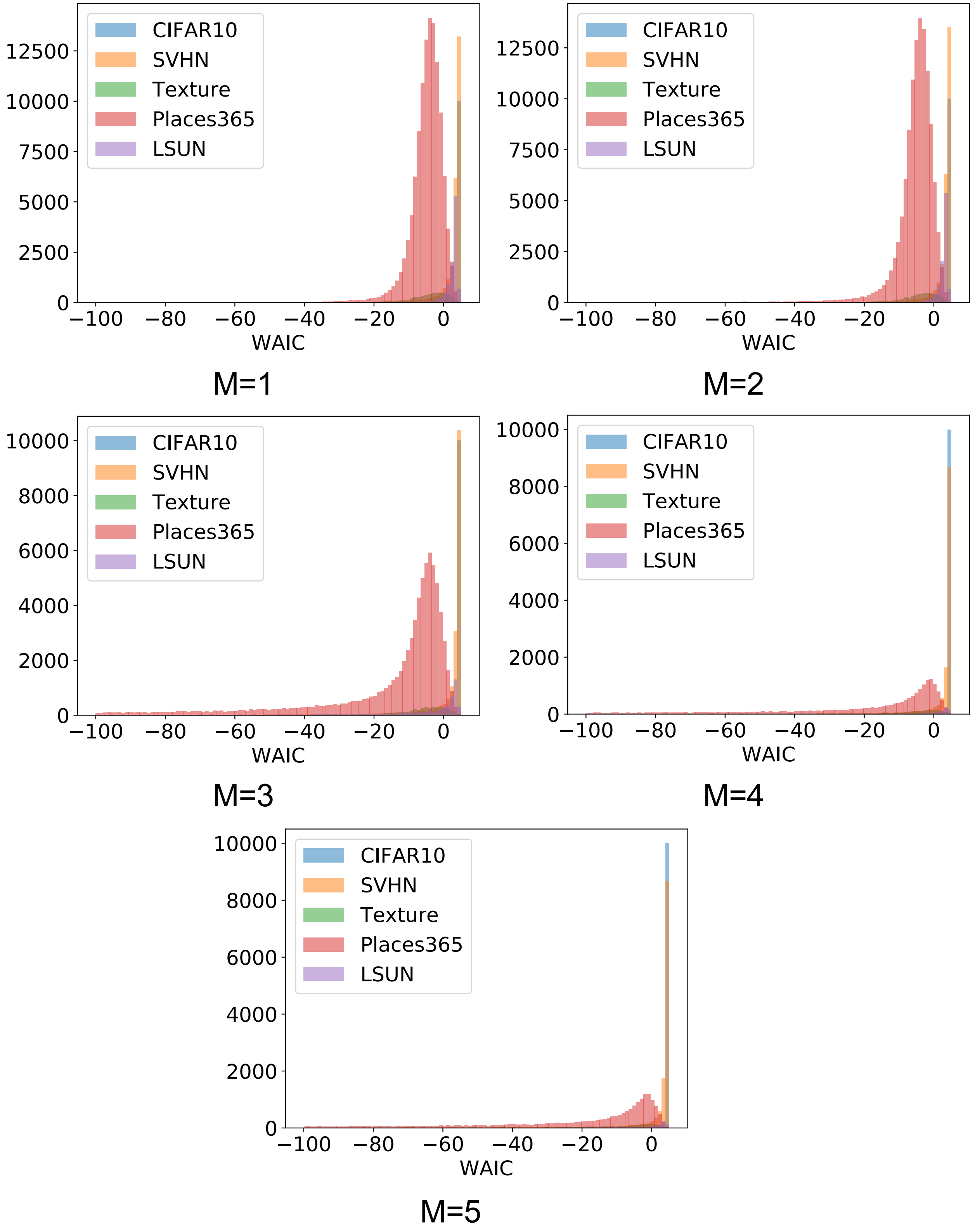
M=2

M=3

M=4

M=5

Figure 15. Effect of ensemble size to the distribution of WAIC scores estimated by model ensembles trained on SVHN.



M=1

M=2

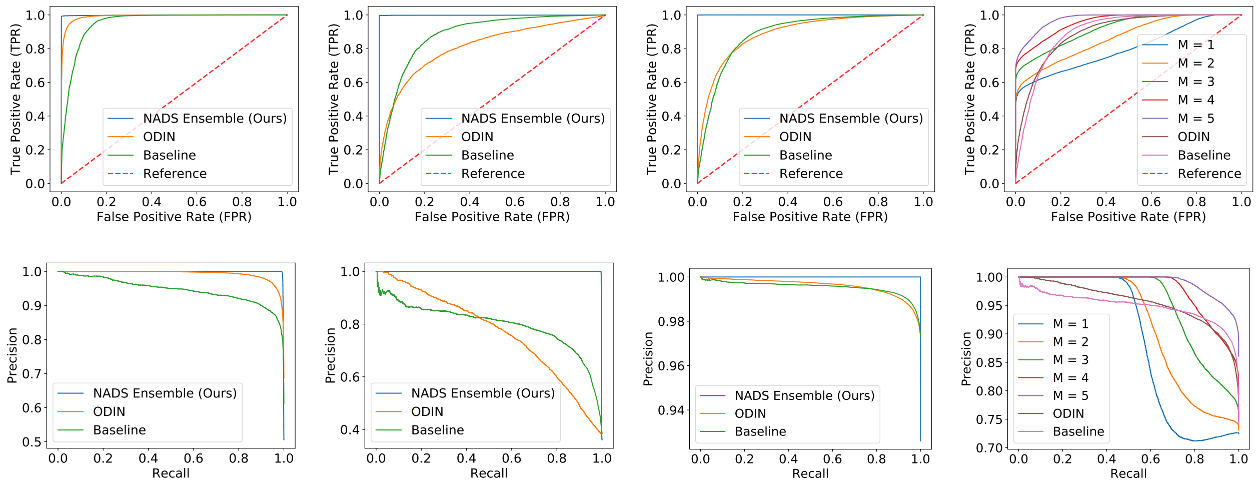
M=3

M=4

M=5

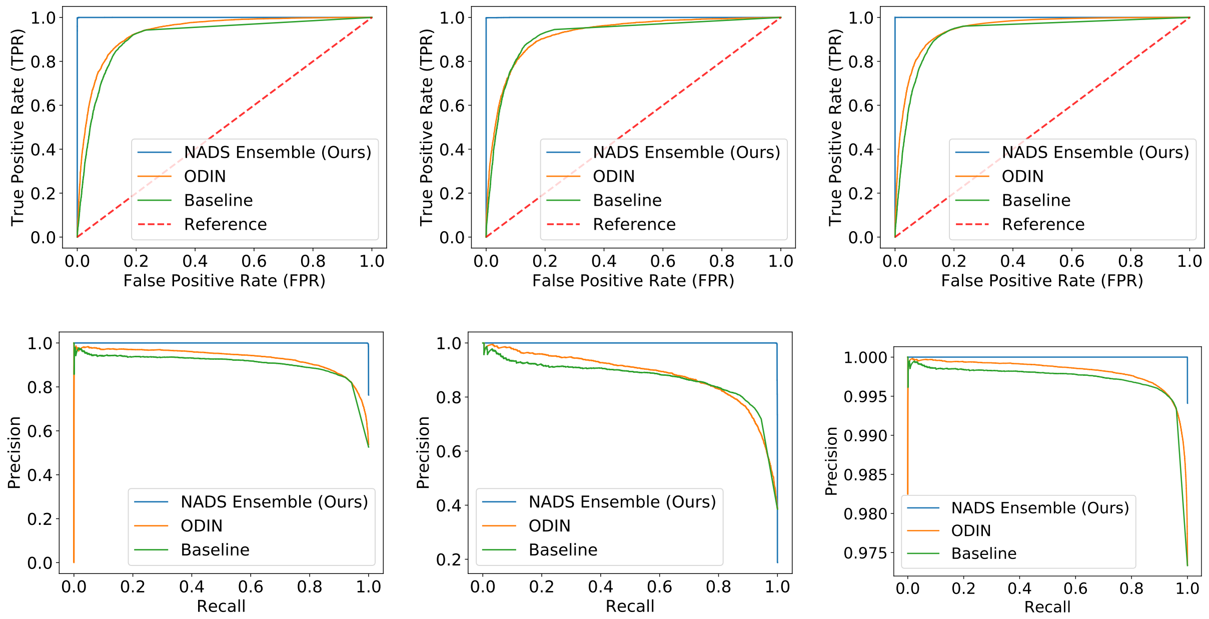
Figure 16. Effect of ensemble size to the distribution of WAIC scores estimated by model ensembles trained on CIFAR-10.

G. Additional ROC and Precision-Recall Curves



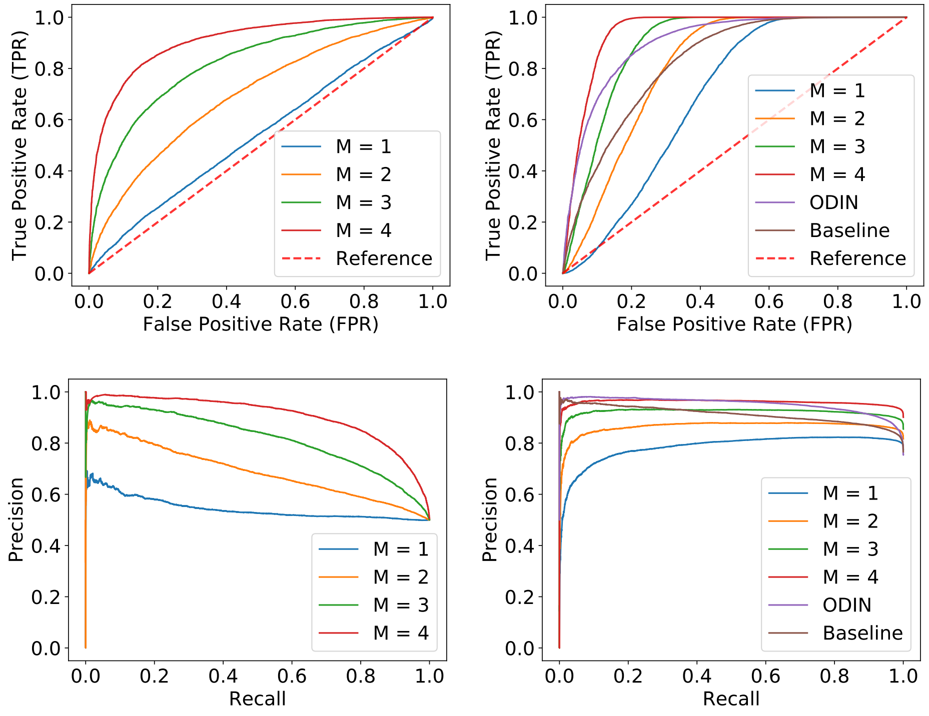
(a) LSUN (b) Texture (c) Places (d) SVHN

Figure 17. ROC and PR curve comparison of methods trained on CIFAR-10



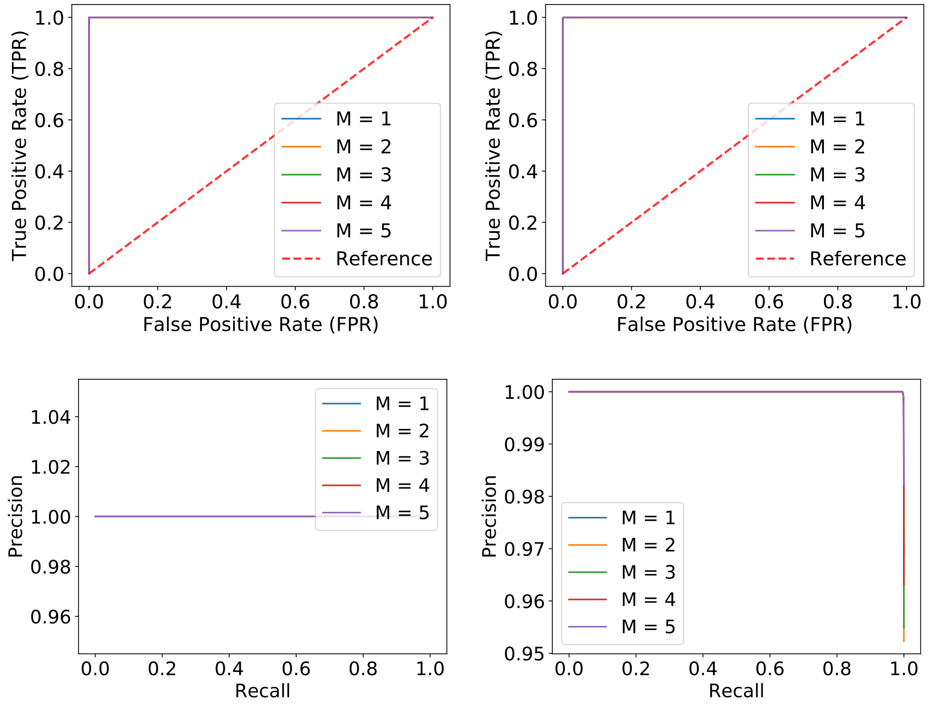
(a) LSUN (b) Texture (c) Places

Figure 18. ROC and PR curve comparison of methods trained on SVHN



(a) LSUN (b) SVHN

Figure 19. ROC and PR curve comparison of methods trained on CIFAR-100



(a) NotMNIST (b) FMNIST

Figure 20. ROC and PR curve comparison of methods trained on MNIST