Signal, Image and Video Processing (2025) 19:766
https://doi.org/10.1007/s11760-025-04376-1

ORIGINAL PAPER

Small-data image classification via drop-in variational autoencoder

Babak Mahdian[1] · **Radim Nedbal**[2]

Received: 20 November 2024 / Revised: 8 May 2025 / Accepted: 9 June 2025

Abstract
It is unclear whether generative approaches can achieve state-of-the-art performance with supervised classification in high-dimensional feature spaces and extremely small datasets. In this paper, we propose a drop-in variational autoencoder (VAE) for the task of supervised learning using an extremely small train set (i.e., $n = 1, .., 5$ images per class). Drop-in classifiers form a usual alternative when traditional approaches to Few-Shot Learning cannot be used. The classification will be defined as a posterior probability density function and approximated by the variational principle. We perform experiments on a large variety of deep feature representations extracted from different layers of popular convolutional neural network (CNN) architectures. We also benchmark with modern classifiers, including Neural Tangent Kernel (NTK), Support Vector Machine (SVM) with NTK kernel and Neural Network Gaussian Process (NNGP). Results obtained indicate that the drop-in VAE classifier outperforms all the compared classifiers in the extremely small data regime.

Keywords Small data classification · Variational autoencoder · Neural Tangent Kernel · Supervised learning

1 Introduction

Large quantities of train data, thousands to millions, are often needed to train popular deep neural networks. This is an obstacle in their application on open-ended long-tail categories in the real-world, where data collection might sometimes be extremely hard, expensive, or even impossible. In such a case, *meta-learning* [1] can be used if a variety of suitable training tasks to pretrain a model is available.[1] In case this assumption is not met, meta-learning can not be used efficiently.

Learning with few labeled data is known also as *Few-Shot Learning* (FSL). It usually implements techniques of (i) meta-learning, sometimes together with *Graph Neural Networks*; and/or (ii) self-supervised learning (usually learning embeddings) or both. Note that recent FSL methods have explored alternatives to traditional Maximum Likelihood Estimation (MLE). For instance, Prototypical Networks [3] use metric-based learning, while methods like GEM apply Bayesian principles to adapt across tasks. Other approaches, such as MAML and Reptile [4], optimize for rapid adaptation without relying on likelihoods. However, these methods require meta-training on related tasks or access to unlabeled data to learn transferable representations. These assumptions do not hold in our setting.

In this paper, we address *supervised learning in the extremely low data regime*, in which none of the aforementioned approaches to FSL can be used. It is the least favorable scenario, which admits only a few labeled data, but no unlabeled data or suitable, related, training tasks.

In extremely low data regime, a usual approach is simply to apply a drop-in classifier such as Support Vector Machine (SVM) on feature representations (these can be hand crafted features or acquired using deep convolutional neural network (CNN) representations). However, supervised learning in the

[1] "Suitable tasks" must be related to the targeted classification – the task of interest: typically, a model is pre-trained on the source dataset consisting of base classes. The acquired prior knowledge is then transferred [2] into the target dataset consisting of novel classes with few examples, where base and novel classes are disjoint. If source (base classes) and target (novel classes) are from the same domain, the expectation is that representations learned will generalize to the target.

Both authors contributed equally to this work

✉ Babak Mahdian
mahdian@utia.cas.cz

Radim Nedbal
radim.nedbal@iit.it

[1] Institute of Information Theory and Automation, The Czech Academy of Sciences, Prague, Czech Republic

[2] Center for Translational Neurophysiology, Istituto Italiano di Tecnologia, Ferrara, Italy

Published online: 20 June 2025

extremely low data regime (and possibly high-dimensional spaces) is challenging. The lack of training data often leads to overfitting, which causes poor classification on unseen data. The right choice of the drop-in classifier is crucial for achieving a high accuracy and generalization. To this end, extensive experiments have been conducted [5] on UCI repository [6]. The UCI consists of a large range of classification tasks, including small data regime ones (mostly with low-dimensional feature vectors). The authors of [5] conclude that, in general, random forest (RF) is the winning classifier, followed by the SVM.

Despite heavy overparameterization, modern Neural Networks (NNs) have low generalization error [7]. This motivated recent research on NNs whose widths (i.e., number of nodes in layers, or number of channels in convolutional layers) goes to infinity. In the infinite-width limit, an inference on a large class of Bayesian NNs converges to Gaussian process regression with a kernel given by the architecture of the NN. This Gaussian process is called Neural Network Gaussian Process (NNGP). This correspondence was first established for shallow fully-connected networks by [8] and was extended to a multi-layer setting [9]. In the limit of infinite-width and infinitesimal learning rate, the trajectory of (non-Bayesian) NN training under ℓ_2 loss converges to the kernel ridge regression with a so-called Neural Tangent Kernel (NTK) [10]. Interestingly, it was shown [11] that NTK performs surprisingly very well in low data regime. It even beats the earlier "gold-standard," RF [5], on small datasets in a majority of tested classification tasks. In [11], it was also shown that SVM with NTK kernel (NSVM) beats linear SVM when trained on training sizes ranging from 20 to 160: the setting in which SVM arguably is the most widely used drop-in classifier.

Inspired by these results, demonstrating advantages of Bayesian learning in the low data regime, we choose to exploit the power of deep generative models formed by the fusion of probabilistic modeling and deep NNs and propose a multi-class classifier based on variational autoencoder (VAE) [12]. VAEs and their variants have seen significant advancements in recent years. Among the most promising developments are deep hierarchical VAEs [13, 14], which increase the expressiveness of the latent variable prior and posterior probability density functions (PDFs), and introduce new training techniques; VQ-VAE [15, 16], which quantize the latent space to provide better control over the latent variable distributions; β-VAE [17, 18], which aim to eliminate the issue of posterior collapse.

It is expected that VAE should perform well for the supervised learning in the extremely low data regime (same as NNGP) because its regularization mechanism is based on a probabilistic latent space. Given a diagonal prior of the latent space, VAE has an incentive to use only as many latent dimensions as needed to encode the available information. With only very few training examples per class, VAE identifies a minimal subspace – essentially, a manifold – that just "fits" these isolated points. In doing so, the VAE performs a kind of substantial dimensionality reduction where the latent space is "compressed" to reliably reconstruct those few training examples. We conjecture that if a high-dimensional feature space already contains well-structured clusters corresponding to different classes, the VAE's projection onto its lower-dimensional manifold preserves the relative separation among the classes because by distilling the data to its most relevant aspects, the VAE identifies and retains the directions in the data that are most influential for reconstructing and, by extension, separating the classes.

Contribution:

1. We propose a VAE for supervised learning using an extremely small dataset. The results of multi-class supervised learning with the number of training samples per class ranging from 1 to 5 indicate that the VAE classifier performs better than all the other tested classifiers making it a strong candidate for the drop-in classifier in the extremely small datasets.
2. The question, whether modern classifiers such as NTK, NNGP, or SVM with NTK kernel (NSVM) can perform well as a drop-in classifier on the extremely small datasets ($n = 1..5$) and high-dimensional spaces is still unanswered. We conduct a large-scale empirical study using a high variety of deep CNN feature representations to answer this question.

Our classifier is built on the standard VAE, serving as a baseline for VAE-based approaches. While we make no architectural changes, our contribution lies in demonstrating its effectiveness as a drop-in supervised classifier in an extremely low-data regime. This is a setting where most Few-Shot and generative models are inapplicable due to their reliance on meta-training or unlabeled data. We investigate how the generative nature of VAEs can enhance robustness compared to discriminative models. Furthermore, benchmarking against kernel-based methods such as NTK and NNGP helps reveal the strengths and weaknesses of VAE's stochastic nature.

The remainder of the paper is organized as follows. Section 2 reviews the theoretical background of VAE. Section 3 details the methodology and the development of the VAE classifier. Section 4 presents the experimental results and proposes future research directions. Section 5 concludes with a summary of the findings.

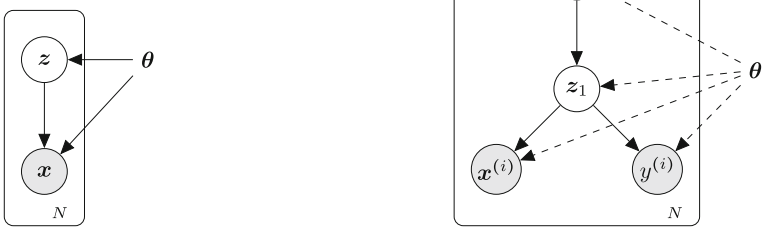

Fig. 1 Each of N i.i.d. observations (shaded) is generated by a (local) latent random variable $\boldsymbol{z}$. They all depend on the global parameter $\boldsymbol{\theta}$.

(a) The generative VAE model. (b) The generative model of the VAE classifier.

2 Review of the Theory

We review the VAE, a deep generative model defined by an explicit probability model, which lays the theoretical foundations for the VAE classifier.

We restrict ourselves to the common case with a dataset $\{\boldsymbol{x}^{(i)}\}$ of independent and identically distributed (i.i.d.) samples whose generation is modelled by a continuous latent variable $\boldsymbol{z}$. Accordingly, the likelihood function is a product of $p(\boldsymbol{x}^{(i)} \mid \boldsymbol{z}_i)$ where $\boldsymbol{z}_i$ are all i.i.d. Since this PDF is often unknown, we assume that it comes from a parametric family of PDFs $p(\boldsymbol{x} \mid \boldsymbol{z}) \triangleq p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$, where $\boldsymbol{\theta}$ is a parameter vector[2] (see Fig. 1a). Furthermore, we assume differentiability almost everywhere w.r.t. $\boldsymbol{\theta}$ as we wish to perform MLE or maximum a posteriori inference on $\boldsymbol{\theta}$ and variational inference on $\boldsymbol{z}$.

This setup for doing variational inference enables efficient inference and learning by using Auto-Encoding Variational Bayes (AEVB) algorithm introduced in [12]. The unobserved $\boldsymbol{z}$ is interpreted as a latent representation or code. Accordingly, a map $x \rightarrow z$ is called a probabilistic encoder since given a datapoint $\boldsymbol{x}^{(i)}$, it produces a PDF (e.g., a Gaussian) over the possible values of the code z from which the datapoint $\boldsymbol{x}^{(i)}$could have been decoded by $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid \boldsymbol{z})$, to which in turn we refer as a probabilistic decoder, denoted as $\mathrm{Dec}(\boldsymbol{z})$. In a VAE, encoders and decoders are implemented as NNs. We will define a decoder as a Gaussian multilayer perceptron (MLP): $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) \triangleq \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{I})$ where the mean vector $\boldsymbol{\mu}$ is calculated from $\boldsymbol{z}$ with an MLP whose parameters are $\boldsymbol{\theta}$. This allows $\boldsymbol{x}$ to depend on $\boldsymbol{z}$ in a complex, highly non-linear way.

Note that we can specify priors $p(\boldsymbol{z}_i)$ to inform and constrain our models. Also, there are no common simplifying assumptions about, e.g., $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$ tractability [19] to enable using the Expectation-Maximization (EM) algorithm [20, 21]. There is even no assumption about the tractability of required expectations for the mean-field variational Bayesian approach [20–22], which could approximate the intractable $p_{\boldsymbol{\theta}}(\boldsymbol{z} \mid \boldsymbol{x})$.

Taking the energy view perspective, we wish to minimize the overall free energy of our training samples, which is given by: $F(\boldsymbol{x}) = -\frac{1}{\beta} \ln \int_{\boldsymbol{z}} e^{-\beta E(\boldsymbol{x},\boldsymbol{z})} \mathrm{d}\boldsymbol{z}$, where $E(\boldsymbol{x}, \boldsymbol{z}) \triangleq C(\boldsymbol{x}, \mathrm{Dec}(\boldsymbol{z}))$ denotes the energy of the system, which is set to be equal to the cost, C, of $\boldsymbol{x}$ and the image of $\boldsymbol{z}$ under the function Dec. The integral is intractable because E is a complex function of $\boldsymbol{z}$, which leaves us with no analytic solution for this log-partition function, a well known problem from statistical physics. One of techniques to address this problem is variational approximation.

First, let

$$F(\boldsymbol{x}) = -\frac{1}{\beta} \ln \int_{\boldsymbol{z}} q(\boldsymbol{z} \mid \boldsymbol{x}) \left[\frac{e^{-\beta E(\boldsymbol{x},\boldsymbol{z})}}{q(\boldsymbol{z} \mid \boldsymbol{x})} \right] \mathrm{d}\boldsymbol{z} \ , \tag{1}$$

where q is a PDF. By Jensen's inequality,

$$F(\boldsymbol{x}) \leq \tilde{F}(\boldsymbol{x}) = \int_{\boldsymbol{z}} q(\boldsymbol{z} \mid \boldsymbol{x}) \left[-\frac{1}{\beta} \ln \frac{e^{-\beta E(\boldsymbol{x},\boldsymbol{z})}}{q(\boldsymbol{z} \mid \boldsymbol{x})} \right] \mathrm{d}\boldsymbol{z} \ , \tag{2}$$

and we can bound $F(\boldsymbol{x})$ by $\tilde{F}$:

$$\tilde{F}(\boldsymbol{x}) = \int_{\boldsymbol{z}} q(\boldsymbol{z} \mid \boldsymbol{x}) E(\boldsymbol{x}, \boldsymbol{z}) \mathrm{d}\boldsymbol{z} + \frac{1}{\beta} \int_{\boldsymbol{z}} q(\boldsymbol{z} \mid \boldsymbol{x}) \ln q(\boldsymbol{z} \mid \boldsymbol{x}) \mathrm{d}\boldsymbol{z} \ . \tag{3}$$

The first term is an expected free energy w.r.t. q, the second term is a negative entropy of q divided by β. This formula is known in thermodynamics as Helmholtz (free) energy: $\tilde{F} = \langle E \rangle - TS$, where $T \triangleq \frac{1}{\beta}$, $S \triangleq -\int_{\boldsymbol{z}} q(\boldsymbol{z} \mid \boldsymbol{y}) \ln q(\boldsymbol{z} \mid \boldsymbol{x}) \mathrm{d}\boldsymbol{z}$ and $\langle E \rangle$ denotes expected value of E. It is also known from thermodynamics that a system that tends towards the equilibrium will tend to minimize this free energy at the given temperature. We want to find parameters of E and q that minimize $\tilde{F}$, thus minimizing also the free energy $F(\boldsymbol{x})$.

[2] We will use interchangeably the notation: $p_{\boldsymbol{\theta}}(\cdot) \equiv p(\cdot; \boldsymbol{\theta})$ where the semicolon separates random variables on the left from parameters on the right.

3 Methodology

We introduce the general probability model for our scenario, define the corresponding VAE, and present the resulting VAE classifier, our main theoretical contribution.

The Scenario. Let $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$ be an observed i.i.d. dataset generated by a continuous latent variable z. A number of probabilistic graphical models [20] is possible with this choice. To increase the expressiveness of both prior and posterior PDF, we consider a simple hierarchical model depicted[3] in Fig. 1b, i.e., $p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}, y^{(i)}, z) = p_{\boldsymbol{\theta}}(z_2) \cdot p_{\boldsymbol{\theta}}(z_1 \mid z_2) \cdot p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)} \mid z_1) \cdot p_{\boldsymbol{\theta}}(y^{(i)} \mid z_1)$. A hierarchical model may reduce the risk of posterior collapse [23, 24].

Naturally, it depends in large part on the dataset in question when a particular factorization is appropriate. For the sake of concreteness, suppose that $\boldsymbol{x}^{(i)}$ is a text and $y^{(i)}$ is its label as "boring" or "interesting." In our model setup, the latent random variable z generates the observed document and also its observed label. Since $\boldsymbol{x}^{(i)}$ and $y^{(i)}$ are conditionally independent given z_1,[4]

$$\boldsymbol{x}^{(i)} \perp\!\!\!\perp y^{(i)} \mid z_1 \ , \tag{4}$$

let us say that the value of z_1 is, e.g., a detailed representation of the content of a text, such as a PDF of its topics [25].

The Architecture of the VAE. We let the prior $p_{\boldsymbol{\theta}}(z_2)$ be a centred isotropic multivariate Gaussian: $p_{\boldsymbol{\theta}}(z_2) \triangleq \mathcal{N}(z_2; \mathbf{0}, \boldsymbol{I})$, which puts pressure on increasing the entropy of q. Parameters of the other conditional PDFs are calculated from z with Gaussian and Bernoulli MLPs. We let $p_{\boldsymbol{\theta}}(z_1 \mid z_2)$, $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid z_1)$ be a multivariate Gaussian with a diagonal, unite covariance. The respective mean values are calculated from z_2 and z_1 with fully connected networks with one hidden layer: $\mathrm{Dec}_0(z_2) = z_1$, $\mathrm{Dec}_1(z_1) = \boldsymbol{\mu}$. We let $p_{\boldsymbol{\theta}}(y \mid z_1)$ be a Bernoulli: $p_{\boldsymbol{\theta}}(y \mid z_1) \triangleq \mathbf{Ber}(y; \mu)$. The parameter μ is calculated from z_1 with a fully connected network with no hidden layers: $\mathrm{Dec}_2(z_1) = \mu$.

The true posterior $p_{\boldsymbol{\theta}}(z \mid \boldsymbol{x}, y)$ is intractable in this case. We assume that it takes on an approximate Gaussian form with an approximately diagonal covariance.[5] Thus $q_{\boldsymbol{\phi}}(z_1 \mid \boldsymbol{x}, y)$ is a multivariate Gaussian with a diagonal covariance structure: $q_{\boldsymbol{\phi}}(z_1 \mid \boldsymbol{x}, y) \triangleq \mathcal{N}\left(z_1; \boldsymbol{\mu}, \boldsymbol{I}\boldsymbol{\sigma}^2\right)$. The parameters $\boldsymbol{\mu}, \boldsymbol{\sigma}$ are calculated from $\boldsymbol{x}$ with a fully connected NN with one hidden layer because, by design choice, we let the encoder ignore y: $\mathrm{Enc}(\boldsymbol{x}) = \boldsymbol{\mu}, \boldsymbol{\sigma}$.

VAE Classifier Our goal is to estimate the true conditional PDF $p(y \mid \boldsymbol{x})$, i.e., given a document, what is the probability of it being interesting? To this end, we derive the following classifier:

(I.) Design encoder that takes $\boldsymbol{x}$ on input and ignores y.

(II.) Solve $\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}} = \arg_{\boldsymbol{\theta},\boldsymbol{\phi}} \min \sum_i \tilde{F}(\boldsymbol{x}^{(i)}, y^{(i)}; \boldsymbol{\theta}, \boldsymbol{\phi})$, where

$$\tilde{F}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{x}, y) = \int_z q(z \mid \boldsymbol{x})[\alpha_1 E(\boldsymbol{x}, z) + \alpha_2 E(y, z)]\mathrm{d}z + \frac{1}{\beta}\int_z q(z \mid \boldsymbol{x}) \ln q(z \mid \boldsymbol{x})\mathrm{d}z \ ,$$

$\boldsymbol{\theta}, \boldsymbol{\phi}$ are parameters of E, q, respectively, and $\alpha_1, \alpha_2, \beta \in \mathbb{R}_{>0}$ modulate the learning constraints applied to the model.

(III.) Given a new $\boldsymbol{x}$, sample L values $z^{(l)}$ from the posterior $q(z_1 \mid \boldsymbol{x}; \hat{\boldsymbol{\phi}})$.

(IV.) Calculate the average of $p(y \mid z^{(l)}; \hat{\boldsymbol{\theta}})$:

$$p(y \mid \boldsymbol{x}) \approx \frac{1}{L}\sum_{l=1}^{L} p\left(y \mid z^{(l)}; \hat{\boldsymbol{\theta}}\right) \quad z^{(l)} \sim q\left(z_1 \mid \boldsymbol{x}; \hat{\boldsymbol{\phi}}\right) \ .$$

Since $p(y \mid \boldsymbol{x}) = \int p(y \mid \boldsymbol{x}, z_1) \cdot p(z_1 \mid \boldsymbol{x})\,\mathrm{d}z_1$, it suffices to show that

$$\begin{aligned} p(y \mid \boldsymbol{x}, z_1) &\approx p(y \mid z_1; \hat{\boldsymbol{\theta}}) \ , \\ p(z_1 \mid \boldsymbol{x}) &\approx q(z_1 \mid \boldsymbol{x}; \hat{\boldsymbol{\phi}}) \ . \end{aligned} \tag{5}$$

To this end, recall that our learning objective yields: (a) a generative model $p(\boldsymbol{x}, y, z; \hat{\boldsymbol{\theta}})$ of $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}$; (b) an approximation $q(z \mid \boldsymbol{x}, y; \hat{\boldsymbol{\phi}})$ of $p(z \mid \boldsymbol{x}, y; \hat{\boldsymbol{\theta}})$. By (a),

$$p(\boldsymbol{x}, y, z_1) \approx p(\boldsymbol{x}, y, z_1; \hat{\boldsymbol{\theta}}) \ , \tag{6}$$

and using (4), we get (5). By (6) and (b), $p(z_1 \mid \boldsymbol{x}, y) \approx p(z_1 \mid \boldsymbol{x}, y; \hat{\boldsymbol{\theta}}) \approx q(z_1 \mid \boldsymbol{x}, y; \hat{\boldsymbol{\phi}})$ where $\hat{\boldsymbol{\phi}}$ are the optimal parameter values of the encoder. By (I), $q(\cdot \mid \boldsymbol{x}, y; \hat{\boldsymbol{\phi}}) \equiv q(\cdot \mid \boldsymbol{x}; \hat{\boldsymbol{\phi}})$ and $p(z_1 \mid \boldsymbol{x}) = \sum_y p(z_1 \mid \boldsymbol{x}, y) \cdot p(y \mid \boldsymbol{x}) \approx q(z_1 \mid \boldsymbol{x}; \hat{\boldsymbol{\phi}})$.

4 Experiments

We compare the VAE classifier with (i) RF, which according to [5] can be considered a reference in low data regime to compare with new drop-in classifier proposals to asses

[3] The generative models in Fig. 1b is a special case of the one in Fig. 1a. Indeed, $\boldsymbol{x}$ corresponds to $(\boldsymbol{x}^{(i)}, y^{(i)})$, and z corresponds to (z_1, z_2). Also note that Fig. 1b is in the Markov equivalence class that includes also $\boldsymbol{x}^{(i)} \to z \to y^{(i)}$ and $\boldsymbol{x}^{(i)} \leftarrow z \leftarrow y^{(i)}$.

[4] This can be read directly from the directed graph in Fig. 1b, making use of the *d-separation* criterion [21].

[5] This assumption effectively encodes that each latent dimension is treated as independent, which is not only computationally efficient but also a step toward disentangled representations (which would require additional inductive biases, such as a weighted Kullback–Leibler (KL) divergence term) [26].

their performance; (ii) NNGP [27], NTK [27], and NSVM, which according to [11] are serious contenders of RF in low data regime; (iii) and Linear/RBF SVM, arguably the most frequently used classifiers for small data tasks.

4.1 Feature representations

To be able to test classifiers with a wide variety of feature representations (i.e., feature datasets), the following ImageNet pretrained convolutional layers followed by average pooling have been extracted: conv3 ($d = 256$), conv4 ($d = 512$), and conv5 ($d = 512$) from VGG 19; mixed 5d ($d = 288$), mixed 6e ($d = 768$), and mixed 7c ($d = 2048$) from Inception V3; block3 rep 0 ($d = 256$), block12 rep 0 ($d = 728$), and act4 ($d = 2048$) from Xception; and conv3 ($d = 512$), conv4 ($d = 1024$), and conv5 ($d = 2048$) of Resnet 50. CIFAR 10, MNIST, and Cat vs. Dog formed our test image datasets.

It is important to note that we do not claim the suitability of the aforementioned representations for the specific classification tasks in our experiments. Our objective is not to achieve the best possible performance on each dataset, but rather to compare classifiers under consistent conditions. To this end, we generate a wide variety of feature representations with varying dimensionality and quality. For the same reason, we do not apply additional pre-processing, data transformation, or feature selection. All classifiers are evaluated on these fixed representations with no fine-tuning performed on the pretrained models. This ensures a fair and consistent comparison focused solely on each classifier's performance under extremely limited labeled data.

4.2 Comparison Setup

The classifiers are trained on balanced training sets with n samples per class, where $n = 1, 2, 3, 4, 5$. We get 1,080 combinations of classifier – feature dataset– training size. The following classifier implementations have been used:

RF in `RandomForestClassifier` of `sklearn`. Number of trees selected from {16, 64, 128, 256, 512}.

NNGP and NTK in Neural Tangents [27][6]. Infinitely wide NNs implemented in several architectures: including with two hidden layers, each with 512 units followed by ReLU activation function or with four hidden layers, each with 2048 units followed by ReLU. The winning architecture was always selected.

SVM in `sklearn.svm.SVC`. We did grid search along kernel types (linear or RBF), regularization, $C \in \{1, 2, 5, 10, 100, 1000\}$, and kernel spread (for RBF), $\gamma \in \{0.1, 1, 10, 100\}$.

NSVM in [11][7]. Grid search performed along maximal depth in $\{0, 1, \ldots, 4\}$ and $C \in \{10^{-2}, 10^{-1}, \ldots, 10^{5}\}$.

VAE in Pyro. Latent variable space dimensions: 80 for z_2, $(80 + D)/2$ for z_1 where D is the dimension of $\boldsymbol{x}$. The numbers of hidden units in hidden layers are: $(3D+80)/4$ in $\text{Enc}(\boldsymbol{x})$; $(D+240)/4$ in the encoder of z_1; $(D+80)/2$ in the decoder of z_1; D in the decoder of $\boldsymbol{x}$, and 0 in the decoder of y. All parameters (variational and generative) are initialized by random sampling from $\mathcal{N}(0, 0.001)$. We set $\alpha_1 = 10$, $\alpha_2 = 100$, $\beta = 1$, and we apply the Bernoulli corruption process to $\boldsymbol{x}$ with the probability 0.1.

The training process of all classifiers is completely non-adaptive. Hyperparameter tuning is performed on half of the training data, and the second half was used for validation. Then the classifier is retrained using the entire selected training set of size n = 1, …, 5 samples per class, and tested on the standard test set samples of the datasets (the test sets are fixed for all evaluations). This process is repeated 50 times for each of the 1,080 combinations. Each split is defined as selecting a random train set (shuffled by a function parameterized by a random state that ranges from 0 to 49). Results from 50 splits are averaged.[8]

Following the non-adaptive methodology, we fixed all the VAE parameters for our experiments. This is in accordance with [28] that launches several criticisms about the usual practice in experimental comparison. One of them is whether the selected learners are properly configured for their best performance. It suggests that proposals of new classifiers usually design and tune them carefully, while the baseline classifiers are run using baseline configurations.

4.3 Results and Discussion

We report average values of Average Precision (AP) together with 25th and 75th percentiles (Q_{25}, Q_{75}) across the 50 splits (to assess also the stability of classification across different seed-runs[9]). The results obtained provide an interesting insight into the performance of the compared state-of-the-art classifiers on such small data tasks.

Table 1 demonstrates AP averaged over all feature datasets for each n. It indicates that the VAE outperforms all the other tested classifiers. It is followed by NTK, NNGP, and RF that

[6] https://github.com/google/neural-tangents

[7] https://github.com/LeoYu/neural-tangent-kernel-UCI

[8] Since results from the respective splits are i.i.d., we can apply the *strong law of large numbers*, by which the probability that the calculated average tends to the true average, as the sample size increases, is 1. In other words, the relatively high number of splits is more likely to eliminate undesirable bias caused by potential outliers.

[9] Since the AP may not be distributed symmetrically, percentiles are more adequate than standard deviation or variance, which are usually used to describe a symmetric PDF.

Table 1 Average Precision (Q_{25}- AP, Q_{75}- AP) averaged over all datasets and n, where n denotes the number of training samples per class

n	SVM	RF	NNGP	NTK	NSVM	VAE
1	19.1% (-1.2, 0.8)	37.7% (-2.9, 3.2)	43.7% (-3.1, 3.5)	43.9% (-3.1, 3.5)	18.8% (-1.4, 1.0)	**44.6% (-3.1, 3.6)**
2	29.5% (-7.0, 5.8)	48.5% (-3.2, 3.8)	50.2% (-2.6, 2.9)	50.2% (-2.4, 2.9)	20.5% (-2.9, 1.9)	**51.3% (-2.7, 3.0)**
3	41.7% (-5.5, 6.9)	50.5% (-3.3, 4.2)	53.9% (-2.3, 2.6)	53.9% (-2.2, 2.5)	41.6% (-5.3, 7.0)	**55.2% (-2.5, 2.7)**
4	42.2% (-9.2, 9.6)	55.9% (-2.4, 2.9)	56.3% (-2.2, 2.4)	56.3% (-2.1, 2.4)	42.6% (-7.4, 9.5)	**57.3% (-2.3, 2.5)**
5	53.3% (-3.1, 4.2)	57.7% (-2.2, 2.8)	58.3% (-1.9, 2.1)	58.2% (-1.9, 2.1)	53.1% (-3.2, 4.2)	**59.3% (-1.9, 2.1)**

have also been found to be strong options for the extremely small data regimes. It is interesting to note that widely used SVM ended up significantly behind. Table 2 demonstrates AP for each n and each feature dataset (i.e., tested CNN layer averaged over image datasets). As apparent in most cases, VAE outperforms all the other classifiers. Also note that as expected, the intervals given by quartile values tend to shrink with n getting larger, which corresponds to decreasing epistemic uncertainty, typical for training set getting bigger.

We observe that VAE does particularly well on datasets with lower Bayes error (feature datasets corresponding to higher feature representations). We can see this on feature datasets corresponding to lower-level CNN representations (e.g., Inception V3 mixed_5d, Resnet50 conv3, Xception block3.rep.0), which typically have a higher Bayes error rate. Note, that NTK, NNGP and sometimes also RF beat VAE mostly on these datasets.

We also performed a *Friedman test* to evaluate differences among classifiers (H_0: None of the classifiers performs significantly better or worse than the others), followed by a post hoc pairwise *Wilcoxon signed-rank test* to assess whether one classifier significantly outperforms another (*one-sided*, H_0: Classifier A performs at least as well as classifier B). Table 3 summarizes the Friedman ranking (FR) of AP, averaged over all datasets and tested representations for each n. The results indicate that VAE achieves the best FR, which is further supported by the Wilcoxon test (p-value ≤ 0.01). Additionally, the last column of Table 2 reports the top-ranked method for each $n = 1..5$ and feature representation averaged over all datasets, based on the FR, with the rank shown in parentheses. VAE ranks the highest on 78% of experiments followed by NTK.

Results obtained signify that VAE is superior for the extremely low-data tasks, beating all the other classifiers. We also note that NTK, NNGP, and RF are strong classifiers for such tasks. The proposed VAE classifier has an average inference time of 108 ms and training time of 14.59 seconds on our setup (2.3 GHz quad-core i7, 32 GB RAM). When operating in an extremely small data regime, all compared methods, including VAE, are computationally efficient.

While our experiments focus on image features, the proposed VAE-based classifier can in principle be applied directly on original images as features or extended to other data types such as text or time series by adapting the encoder and decoder architectures accordingly (e.g., using RNNs). These domains may present additional challenges, such as modeling temporal dependencies or handling sparse representations, but the core advantage of Bayesian learning in low-data regimes remains relevant. A limitation of this work is that the proposed VAE-based classifier is currently designed for datasets with a relatively small number of classes. Extending it to hundreds of classes may require adaptations such as *VAE with flows* [29] (effectively replacing the simple Gaussian latent prior with a flow-based transformation, increasing the flexibility and expressiveness of the learned latent representations), or *class-conditional VAEs* [30, 31]. Addressing these challenges represents a promising direction for future research.

5 Conclusion

We demonstrated that the proposed VAE classifier could generalize so well on small datasets that it outperforms all the other tested classifiers, including, in low data regime arguably the most frequently used, linear/RBF SVM and RF. Generative model classifiers perform well in a low data regime because they do Bayesian learning, which can control overfitting better than its frequentist counterparts like SVM, and RF, doing MLE. We base our conclusions on experiments for which we have chosen a variety of features acquired from different layers of popular NNs.

Table 2 AP (Q_{25}- AP, Q_{75}- AP) for each $n = 1..5$ averaged over all datasets (for each n reported separately). The last column (FR) indicates the top-ranked method based on the FR, with the rank shown in parentheses

Network Layer	SVM	RF	NNGP	NTK	NSVM	VAE	FR
n= 1							
VGG19 conv3	19.6% (-1.6, 1.5)	35.9% (-3.1, 3.2)	35.7% (-3.1, 2.8)	36.3% (-3.0, 3.0)	19.8% (-1.6, 1.4)	**36.5% (-3.6, 3.7)**	VAE (2)
VGG19 conv4	17.0% (-1.2, 0.6)	40.7% (-4.0, 3.1)	47.2% (-3.3, 4.5)	47.6% (-3.3, 4.3)	17.0% (-1.3, 1.0)	**49.9% (-3.4, 4.1)**	VAE (1.33)
VGG19 conv5	15.2% (-0.8, 0.0)	43.7% (-3.0, 4.0)	56.0% (-3.5, 3.6)	56.6% (-3.3, 3.9)	14.7% (-0.5, 0.1)	**58.8% (-3.9, 4.7)**	VAE (1)
IncV3 mixed_5d	20.3% (-1.2, 1.1)	34.8% (-2.4, 2.8)	38.9% (-3.4, 4.0)	**39.1% (-3.8, 3.8)**	19.9% (-1.8, 1.4)	36.7% (-3.3, 3.7)	NTK (2)
IncV3 mixed_6e	15.4% (-0.9, -0.1)	43.2% (-3.6, 4.3)	63.5% (-2.7, 4.9)	63.7% (-2.7, 4.7)	15.6% (-0.9, 0.2)	**66.0% (-2.7, 5.1)**	VAE (1.67)
IncV3 mixed_7c	15.6% (-0.8, 0.2)	42.3% (-3.0, 4.2)	59.2% (-3.0, 4.3)	58.8% (-2.9, 4.3)	14.8% (-0.6, -0.2)	**63.4% (-2.8, 4.5)**	VAE (1)
Resnet50 conv3	20.5% (-0.9, 1.0)	36.0% (-2.4, 2.6)	38.2% (-3.6, 4.0)	**38.8% (-3.8, 3.9)**	19.8% (-1.9, 1.4)	35.8% (-2.7, 2.8)	NTK (1)
Resnet50 conv4	17.3% (-1.0, 0.7)	44.5% (-2.6, 2.9)	55.3% (-3.4, 4.1)	56.0% (-3.3, 4.3)	16.4% (-1.1, 0.4)	**58.6% (-3.5, 4.9)**	VAE (1)
Resnet50 conv5	14.7% (-0.4, 0.3)	47.4% (-3.2, 3.7)	65.4% (-3.6, 3.8)	65.2% (-3.5, 4.0)	14.3% (-0.2, 0.0)	**67.5% (-3.2, 3.5)**	VAE (1)
Xcep b3.rep.0	20.9% (-2.1, 1.7)	32.3% (-3.0, 2.7)	32.8% (-3.8, 3.6)	**33.4% (-3.6, 3.6)**	21.0% (-2.0, 2.3)	32.8% (-3.4, 3.2)	NTK (1.33)
Xcep b12.rep.0	17.2% (-1.1, 0.4)	42.4% (-3.4, 3.5)	53.8% (-4.3, 4.9)	54.0% (-4.2, 4.8)	16.7% (-1.4, 0.9)	**58.1% (-4.8, 4.8)**	VAE (1)
Xcep act4	15.4% (-0.9, 0.1)	47.1% (-3.4, 4.3)	59.8% (-1.7, 3.8)	59.5% (-1.9, 3.8)	14.7% (-0.6, -0.2)	**61.2% (-1.3, 4.1)**	VAE (1.67)
n= 2							
VGG19 conv3	28.6% (-3.6, 1.8)	42.2% (-3.6, 4.0)	42.1% (-3.1, 3.2)	42.1% (-2.9, 3.2)	18.7% (-1.2, 0.7)	**43.0% (-3.3, 3.3)**	VAE (2)
VGG19 conv4	30.5% (-7.0, 2.2)	55.1% (-4.2, 5.6)	56.8% (-3.8, 4.2)	57.1% (-3.7, 4.0)	16.1% (-0.9, 0.4)	**60.6% (-3.4, 3.5)**	VAE (1)
VGG19 conv5	35.1% (-8.6, 13.0)	62.7% (-3.4, 4.9)	65.3% (-2.2, 2.7)	66.0% (-2.0, 2.7)	14.8% (-0.8, -0.4)	**69.2% (-2.6, 3.1)**	VAE (1)
IncV3 mixed_5d	23.0% (-3.2, 2.2)	42.2% (-2.6, 3.2)	**46.1% (-3.2, 3.5)**	45.6% (-2.9, 3.4)	20.3% (-0.8, 0.7)	43.7% (-2.7, 3.1)	VAE (1.67)
IncV3 mixed_6e	36.1% (-15.5, 13.4)	65.0% (-3.3, 4.5)	74.5% (-1.9, 2.8)	74.7% (-2.1, 2.6)	19.1% (-0.9, 0.4)	**77.5% (-2.1, 3.1)**	VAE (1.67)
IncV3 mixed_7c	29.9% (-11.8, 16.6)	66.3% (-2.5, 4.6)	70.0% (-2.2, 3.0)	69.6% (-2.2, 3.1)	28.2% (-13.3, 17.2)	**73.3% (-2.1, 3.0)**	VAE (1)
Resnet50 conv3	28.2% (-3.6, 2.4)	43.7% (-3.5, 3.6)	43.7% (-3.3, 3.1)	**44.2% (-2.9, 3.1)**	19.2% (-1.6, 1.0)	41.5% (-3.7, 3.6)	NTK (1.67)
Resnet50 conv4	32.4% (-6.9, 5.1)	63.1% (-5.7, 5.0)	65.7% (-2.5, 3.8)	66.3% (-2.2, 3.7)	19.8% (-3.9, -0.5)	**67.7% (-3.0, 3.7)**	VAE (1.67)
Resnet50 conv5	42.1% (-18.6, 11.2)	71.2% (-2.1, 3.2)	75.9% (-1.9, 2.1)	75.9% (-1.7, 2.1)	14.1% (-0.2, -0.1)	**77.0% (-2.1, 2.3)**	VAE (1)
Xcep b3.rep.0	28.5% (-2.2, 3.5)	35.1% (-3.1, 3.1)	**36.5% (-2.9, 3.1)**	36.4% (-2.8, 3.1)	20.9% (-2.4, 1.5)	36.1% (-2.8, 3.1)	NTK (1.67)
Xcep b12.rep.0	30.9% (-8.5, 3.2)	61.6% (-4.4, 5.2)	65.7% (-3.7, 4.5)	65.6% (-3.2, 4.7)	15.6% (-0.8, 0.3)	**69.1% (-3.4, 4.3)**	VAE (1)
Xcep act4	37.5% (-17.0, 11.1)	65.2% (-3.0, 4.3)	69.8% (-2.1, 2.5)	69.4% (-2.0, 2.5)	31.3% (-17.2, 11.9)	**71.2% (-2.3, 2.8)**	NNGP (1.67)
n= 3							
VGG19 conv3	32.6% (-5.8, 5.0)	45.0% (-3.8, 3.9)	46.2% (-2.9, 3.9)	46.2% (-3.0, 3.5)	31.7% (-5.4, 4.7)	**47.9% (-3.6, 4.5)**	VAE (1.33)
VGG19 conv4	44.3% (-11.7, 13.0)	57.8% (-4.3, 5.8)	63.6% (-2.0, 3.2)	63.9% (-2.1, 3.0)	45.1% (-10.5, 12.8)	**67.8% (-2.6, 3.4)**	VAE (1)
VGG19 conv5	57.1% (-3.8, 7.8)	65.0% (-4.6, 6.6)	70.9% (-1.7, 1.9)	71.5% (-1.6, 2.1)	57.2% (-3.5, 8.4)	**74.7% (-2.1, 2.3)**	VAE (1)
IncV3 mixed_5d	33.3% (-5.7, 6.5)	44.7% (-3.6, 3.9)	**50.8% (-3.0, 3.5)**	50.3% (-3.2, 3.7)	34.6% (-5.0, 4.5)	49.4% (-3.8, 4.0)	VAE (1.67)
IncV3 mixed_6e	61.1% (-1.5, 11.0)	68.8% (-3.3, 4.9)	79.7% (-1.6, 2.3)	79.9% (-1.6, 2.0)	61.3% (-3.8, 10.4)	**82.8% (-1.8, 2.1)**	VAE (1.67)
IncV3 mixed_7c	55.6% (-5.3, 10.9)	66.7% (-5.3, 7.1)	75.1% (-1.9, 2.1)	74.9% (-1.9, 2.0)	56.4% (-5.6, 10.5)	**78.1% (-1.8, 2.0)**	VAE (1)
Resnet50 conv3	34.4% (-5.9, 6.7)	46.2% (-4.0, 4.2)	47.3% (-3.6, 3.5)	**48.0% (-3.3, 3.7)**	33.6% (-5.8, 5.4)	45.9% (-3.4, 3.0)	NTK (2)
Resnet50 conv4	52.4% (-13.1, 12.2)	65.5% (-3.7, 6.2)	71.7% (-3.0, 3.4)	72.2% (-2.5, 3.3)	51.0% (-13.0, 14.0)	**73.7% (-3.0, 3.0)**	VAE (1)
Resnet50 conv5	69.2% (-4.5, 6.0)	72.6% (-4.0, 6.2)	81.0% (-1.4, 1.7)	81.1% (-1.5, 1.7)	66.8% (-4.8, 8.1)	**82.3% (-1.7, 2.0)**	VAE (1.33)
Xcep b3.rep.0	30.3% (-3.7, 3.8)	36.8% (-2.3, 2.7)	**38.5% (-2.8, 3.2)**	38.1% (-2.8, 3.0)	30.2% (-4.5, 3.7)	37.7% (-2.5, 3.0)	NTK (1.67)
Xcep b12.rep.0	50.5% (-15.3, 14.0)	64.6% (-2.7, 3.9)	72.2% (-2.8, 3.4)	72.2% (-2.8, 3.3)	48.5% (-13.3, 14.8)	**75.9% (-2.6, 2.8)**	VAE (1)
Xcep act4	58.6% (-4.4, 9.2)	68.1% (-3.6, 5.4)	75.0% (-1.7, 2.0)	74.7% (-1.8, 1.9)	58.3% (-4.9, 8.5)	**76.5% (-1.6, 2.2)**	VAE (1.33)
n= 4							
VGG19 conv3	32.7% (-6.6, 6.0)	49.7% (-3.5, 3.1)	49.8% (-3.1, 3.2)	49.5% (-2.9, 3.1)	32.1% (-5.8, 5.7)	**52.0% (-3.9, 3.9)**	VAE (1)
VGG19 conv4	45.6% (-9.1, 16.3)	66.6% (-2.6, 3.4)	67.4% (-2.0, 3.4)	68.0% (-1.9, 3.3)	45.8% (-8.2, 16.4)	**71.7% (-2.3, 3.2)**	VAE (1)
VGG19 conv5	49.1% (-25.5, 20.1)	74.3% (-2.0, 2.4)	74.1% (-1.9, 1.8)	75.0% (-1.7, 1.8)	56.8% (-21.0, 13.1)	**77.8% (-1.5, 1.9)**	VAE (1)
IncV3 mixed_5d	33.9% (-8.4, 6.5)	50.2% (-2.8, 3.4)	**54.3% (-3.4, 3.4)**	53.7% (-3.2, 3.6)	35.3% (-6.7, 6.9)	53.4% (-3.1, 3.6)	VAE (2)

Table 2 continued

Network Layer	SVM	RF	NNGP	NTK	NSVM	VAE	FR
IncV3 mixed_6e	55.7% (-22.1, 22.2)	76.8% (-1.9, 3.0)	82.9% (-1.5, 1.8)	83.2% (-1.4, 1.8)	51.6% (-19.2, 26.8)	**85.7% (-1.5, 1.5)**	VAE (1.67)
IncV3 mixed_7c	63.1% (-17.9, 13.1)	76.6% (-1.7, 2.5)	**78.6% (-1.6, 1.9)**	78.4% (-1.6, 1.8)	62.6% (-3.6, 9.8)	77.7% (-1.8, 2.4)	NTK (2)
Resnet50 conv3	36.1% (-5.8, 5.3)	**51.9% (-3.1, 3.1)**	50.4% (-2.8, 3.1)	51.0% (-2.7, 3.1)	35.6% (-6.1, 5.8)	49.6% (-3.8, 3.6)	NTK (2.33)
Resnet50 conv4	55.0% (-13.1, 17.7)	74.3% (-3.0, 3.8)	75.5% (-2.5, 3.0)	76.0% (-2.6, 3.0)	55.7% (-13.4, 14.1)	**76.6% (-2.5, 2.9)**	VAE (1.67)
Resnet50 conv5	62.5% (-12.9, 16.7)	81.3% (-1.8, 2.2)	84.2% (-1.2, 1.6)	84.3% (-1.4, 1.6)	56.4% (-16.0, 23.7)	**85.6% (-1.4, 1.6)**	VAE (1)
Xcep b3.rep.0	30.6% (-3.0, 4.8)	38.5% (-2.5, 3.0)	**40.5% (-2.7, 2.7)**	39.5% (-2.5, 2.8)	30.9% (-4.3, 4.2)	38.8% (-2.5, 2.8)	NTK (1.67)
Xcep b12.rep.0	54.1% (-19.0, 17.4)	73.6% (-2.1, 3.9)	75.6% (-2.8, 2.7)	75.8% (-2.7, 2.7)	54.3% (-12.5, 16.4)	**79.6% (-2.4, 2.7)**	VAE (1)
Xcep act4	65.0% (-3.2, 11.1)	75.4% (-2.1, 2.6)	78.2% (-1.4, 1.7)	78.0% (-1.3, 1.7)	64.8% (-3.1, 10.4)	**79.1% (-1.4, 1.7)**	VAE (1.67)
n= 5							
VGG19 conv3	42.2% (-6.5, 6.0)	51.4% (-2.6, 3.4)	52.4% (-2.4, 2.6)	52.1% (-2.9, 3.1)	40.5% (-7.1, 6.6)	**55.0% (-2.8, 3.2)**	VAE (1)
VGG19 conv4	62.6% (-1.6, 7.4)	68.7% (-3.3, 3.9)	70.4% (-2.1, 2.7)	71.1% (-2.0, 2.6)	62.6% (-2.0, 7.2)	**74.6% (-2.0, 2.2)**	VAE (1)
VGG19 conv5	73.8% (-2.3, 2.8)	77.0% (-1.4, 2.6)	76.5% (-1.5, 1.2)	77.4% (-1.3, 1.3)	72.9% (-2.4, 3.1)	**80.2% (-1.3, 1.3)**	VAE (1.67)
IncV3 mixed_5d	46.2% (-6.2, 6.4)	52.6% (-2.6, 2.9)	**57.1% (-2.6, 2.9)**	56.5% (-2.7, 2.7)	46.2% (-6.5, 6.6)	56.7% (-2.7, 2.8)	VAE (1.67)
IncV3 mixed_6e	81.4% (-2.4, 2.7)	79.7% (-1.5, 2.4)	85.0% (-1.0, 1.2)	85.4% (-1.0, 1.0)	81.1% (-1.6, 2.7)	**87.6% (-0.8, 1.0)**	VAE (2.33)
IncV3 mixed_7c	76.7% (-3.0, 4.5)	79.2% (-1.2, 2.4)	80.9% (-1.2, 1.4)	80.8% (-1.2, 1.4)	77.6% (-2.5, 3.5)	**82.2% (-1.4, 1.4)**	VAE (1.67)
Resnet50 conv3	44.5% (-5.2, 6.1)	**53.8% (-2.9, 3.5)**	52.9% (-2.4, 3.0)	53.3% (-2.7, 2.7)	42.8% (-5.6, 6.4)	52.0% (-2.8, 3.0)	NTK (2.33)
Resnet50 conv4	71.4% (-1.4, 5.6)	76.4% (-2.5, 3.7)	78.3% (-2.2, 2.3)	78.7% (-2.2, 2.1)	70.5% (-0.9, 6.8)	**79.9% (-1.8, 2.1)**	VAE (1)
Resnet50 conv5	83.4% (-0.9, 2.3)	83.0% (-1.2, 1.8)	86.3% (-0.9, 1.1)	86.5% (-1.0, 1.0)	83.0% (-1.9, 2.6)	**87.3% (-1.1, 1.2)**	VAE (1.67)
Xcep b3.rep.0	35.1% (-3.4, 4.4)	39.9% (-2.5, 3.3)	**42.4% (-3.0, 2.8)**	40.9% (-2.7, 3.0)	35.0% (-4.2, 4.3)	40.1% (-2.3, 2.6)	NTK (1.67)
Xcep b12.rep.0	70.5% (-1.2, 6.7)	76.3% (-2.3, 2.8)	78.3% (-1.6, 2.5)	78.5% (-1.5, 2.4)	70.9% (-0.6, 5.7)	**82.2% (-1.3, 2.0)**	VAE (1)
Xcep act4	77.0% (-2.9, 3.2)	77.2% (-1.5, 2.3)	80.4% (-1.2, 1.3)	80.3% (-1.2, 1.2)	76.3% (-2.8, 4.5)	**81.8% (-1.2, 1.3)**	VAE (2.33)

The bold values indicate the best-performing method

Table 3 Friedman ranking of AP results averaged over all datasets and tested representations for each *n*

n	SVM	RF	NNGP	NTK	NSVM	VAE
1	5.6	3.1	2.81	2.18	5.4	**1.92**
2	5.05	2.94	2.73	2.4	5.95	**1.94**
3	5.45	3.32	2.71	2.45	5.37	**1.69**
4	5.58	2.63	2.81	2.68	5.32	**1.98**
5	4.77	3.11	3.02	2.87	5.05	**2.18**

The bold values in the tables are important, as they highlight the best-performing method

Funding Open access publishing supported by the institutions participating in the CzechELib Transformative Agreement.

References

1. Finn, C., Abbeel, P. & Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks (2017)
2. Kornblith, S., Shlens, J. & Le, Q. V. Do better imagenet models transfer better? (2019)
3. Snell, J., Swersky, K. & Zemel, R. S. Prototypical networks for few-shot learning. *CoRR* arXiv:1703.05175 (2017)
4. Nichol, A., Achiam, J. & Schulman, J. On first-order meta-learning algorithms. *CoRR* **abs/1803.02999** (2018)
5. Delgado, M.F., Cernadas, E., Barro, S.: Do we need hundreds of classifiers to solve real world classification problems? J. M. L. R. **15**, 3133–3181 (2014)
6. Dua, D. & Graff, C. UCI machine learning repository (2017)
7. Olson, M., Wyner, A. & Berk, R. NeurIPS (ed.) *Modern neural networks generalize on small data sets*. (ed. NeurIPS) , 123–456 (2018)

8. Neal, R. M. *Priors for Infinite Networks*, 29–53 (Springer New York, 1996)
9. Lee, J. *et al.* ICLR (ed.) *Deep neural networks as Gaussian processes*. (ed. ICLR) *ICLR* (OpenReview.net, 2018)
10. Jacot, A. & Hongler, C. NeurIPS (ed.) *Neural tangent kernel: Convergence and generalization in neural networks*. (ed. NeurIPS) , 8580–8589 (2018)
11. Arora, S. *et al.* ICLR (ed.) *Harnessing the power of infinitely wide deep nets on small-data tasks*. (ed. ICLR) *ICLR 2020, Ethiopia, April 26-30, 2020* (2020)
12. Kingma, D. P. & Welling, M. ICLR (ed.) *Auto-encoding variational bayes*. (ed. ICLR) (2014)
13. Vahdat, A.: & Kautz, J. A deep hierarchical variational autoencoder, Nvae (2021)
14. Havtorn, J. D., Frellsen, J., Hauberg, S. & Maaløe, L. ICML (ed.) *Hierarchical vaes know what they don't know*. (ed. ICML) , Vol. 139, 4117–4128 (PMLR, 2021)
15. Razavi, A., van den Oord, A. & Vinyals, O. Generating diverse high-fidelity images with vq-vae-2 (2019)
16. Hoyos, A. & Rivera, M. Attentive vq-vae (2024)
17. Li, Z. & Liu, H. Beta-vae has 2 behaviors: Pca or ica? (2023)
18. Burgess, C. P. *et al.*: Understanding disentangling in β-vae (2018)
19. Blei, D. M., Kucukelbir, A. & McAuliffe, J. D.: Variational inference: A review for statisticians. *CoRR* arXiv:1601.00670 (2016)
20. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics) (Springer-Verlag, New York Inc, Secaucus, NJ, USA (2006)
21. Murphy, K. P. *Machine learning : a probabilistic perspective* (MIT Press, 2012)
22. Jordan, M.I., Ghahramani, Z., Jaakkola, T.S., Saul, L.K.: An introduction to variational methods for graphical models. Mach. Learn. **37**, 183–233 (1999)
23. Higgins, I. *et al.* ICLR (ed.) *beta-vae: Learning basic visual concepts with a constrained variational framework*. (ed. ICLR) *ICLR* (2017)
24. Lucas, J., Tucker, G. & Norouzi, M. CoRR (ed.) *Don't blame the ELBO! A linear VAE perspective on posterior collapse*. (ed. CoRR) , 9403–9413 (2019)
25. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
26. Fil, M., Mesinovic, M., Morris, M.: & Wildberger, J. Challenges and extensions, Beta-vae reproducibility (2021)
27. Novak, R. *et al.* Neural tangents: Easy infinite nns in python. *CoRR* (2019)
28. Macià, N., Bernadó-Mansilla, E., Orriols-Puig, A., Ho, T.: Learner excellence biased by data set selection. Pattern Recognition **46**, 1054–1066 (2013)
29. Su, J. & Wu, G. f-vaes: Improve vaes with conditional flows (2018)
30. Harvey, W., Naderiparizi, S. & Wood, F. Conditional image generation by conditioning variational auto-encoders (2022)
31. Ramchandran, S., Tikhonov, G., Lönnroth, O., Tiikkainen, P. & Lähdesmäki, H. Learning conditional variational autoencoders with missing covariates (2022)