Set Based Stochastic Subsampling

Bruno Andreis [1] **Seanie Lee** [1] **A. Tuan Nguyen** [2] **Juho Lee** [1 3] **Eunho Yang** [1 3] **Sung Ju Hwang** [1 3]

Abstract

Deep models are designed to operate on huge volumes of high dimensional data such as images. In order to reduce the volume of data these models must process, we propose a set-based two-stage end-to-end neural subsampling model that is jointly optimized with an *arbitrary* downstream task network (e.g. classifier). In the first stage, we efficiently subsample *candidate elements* using conditionally independent Bernoulli random variables by capturing coarse grained global information using set encoding functions, followed by conditionally dependent autoregressive subsampling of the candidate elements using Categorical random variables by modeling pair-wise interactions using set attention networks in the second stage. We apply our method to feature and instance selection and show that it outperforms the relevant baselines under low subsampling rates on a variety of tasks including image classification, image reconstruction, function reconstruction and few-shot classification. Additionally, for nonparametric models such as Neural Processes that require to leverage the whole training data at inference time, we show that our method enhances the scalability of these models.

1. Introduction

Deep models operate on large volumes of high-dimensional dense inputs such as the pixels of an image (Deng et al., 2009; Krizhevsky et al., 2009; Liu et al., 2015). Training or evaluating models with such data is computationally expensive and several works (Balın et al., 2019; Huijben et al., 2019; Yoon et al., 2019) have proposed subsampling techniques to subsample such dense inputs. Subsampling methods have the potential to drastically reduce the data acquisition effort and reduce the inference time of algorithms that operate on dense inputs. Additionally, subsampling techniques have found applications in medical research for the purpose of interpretation (Ribeiro et al., 2016).

However, these methods have a major drawback in that they require a fixed input structure. For instance, some of these methods are only applicable when each feature (e.g. a pixel) of the input is 1-dimensional. This restriction is imposed by the way the subsampling methods are designed: the input (e.g. an image) is flattened to a single vector and a model predicts a binary mask for each feature (e.g. pixels). This becomes problematic when we consider a 3-channel image. Flattening out the image results in ambiguities as to which pixels to select since channels are treated independently. For instance, in order to perform subsampling on CIFAR10 images, INVASE (Yoon et al., 2019) and DPS (Huijben et al., 2019) convert the images into single channel images (e.g. grey-scaled images) before subsampling pixels. In more extreme cases such as subsampling of training instances (different from instance-wise feature selection), each feature is itself an image (each possibly multi-channel) and hence those subsampling techniques are inapplicable. Finally, we show in our experiments that most of these methods fail under extremely low subsampling rates and their performance is similar to random sampling in this setting.

In order to tackle these limitation, we propose to consider each feature or instance as an element of a set. We formulate the subsampling problem as selecting a subset of features or instances that minimizes the performance degradation of an arbitrary model on an arbitrary task such as image classification, regression or instance subsampling for target tasks such as few-shot classification. As a result, there are several advantages compared to the previous works. First, we can handle arbitrary input structure using set functions (Zaheer et al., 2017; Lee et al., 2019) parameterized with expressive neural networks. Second, a subsampling model with set functions can process arbitrary number of elements. As a result, the model is robust to a wide range of subsampling rates at test time even when trained with a fixed sampling rate. Lastly, the set-based formulation unifies the feature and instance subsampling tasks under a single framework.

However, it is prohibitively expensive to process all the

[1]Graduate School of AI, Korea Advanced Institute of Science and Technology (KAIST), Seoul, South Korea [2]University of Oxford, Oxford, United Kingdom [3]AITRICS, Seoul, South Korea. Correspondence to: Bruno Andreis <andries@kaist.ac.kr>, Sung Ju Hwang <sjhwang82@kaist.ac.kr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

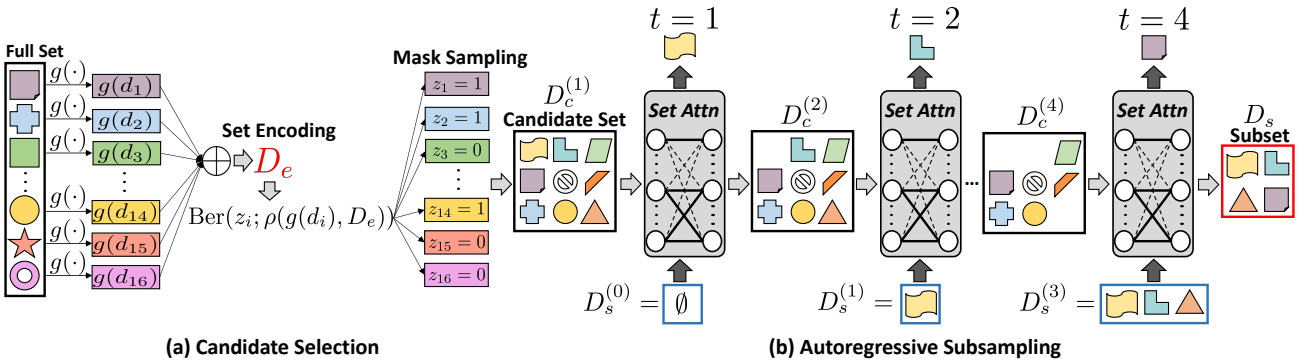

Figure 1. **Concept**: Two-stage set-based stochastic subsampling process. **(a)** In the first stage, we screen out less important samples to construct the candidate subset. **(b)** In the second stage, we autoregressively subsample from the candidate subset.

set elements (e.g. all the pixels in an image) with expressive set function such as Set Transformer (Lee et al., 2019) due to the self-attention among elements. Hence we propose an efficient two-stage subsampling method. In the first stage, as shown in Fig. 1-(a), we learn the sampling rate for individual samples and efficiently screen out less important ones resulting in a subset which we call the *candidate set*. The second stage is more fine-grained and designed to select a smaller subset from the candidate set by considering the relative importance of the samples in the candidate set using a conditionally dependent Categorical distribution through an autoregressive procedure as shown in Fig. 1-(b). Once optimized, the resulting subsampling model can perform stochastic subsampling of a given input with linear time complexity. We call the resulting model *Set based Stochastic Subsampling* (SSS) which is a general subsampling framework that is applicable to both *feature* and *instance* selection.

We validate SSS on multiple datasets and tasks such as 1D function regression, 2D image reconstruction and classification for both feature and instance selection. The experimental results show that SSS is able to subsample with minimal degradation on the target task performance under extremely low subsampling rates, largely outperforming the relevant baselines. We summarize our contribution as follows.

- We reformulate the feature and instance subsampling problem by treating all the features or instances as members of a *set*. This allows us to apply set based functions to the subsampling problem and extend its range of applicability.
- We propose a set based two-stage stochastic subsampling method that learns to efficiently subsample a set with minimal performance degradation on a target task.
- We verify the efficacy and generality of our method on various datasets for feature selection in the input space (e.g. pixels) and instance selection from a dataset, and show that it significantly outperforms the relevant baselines.

2. Related Work

Set Functions Recently, extensive research efforts have been made in the area of set representation learning with the goal of obtaining order-invariant (or equivariant) and size-invariant representations. Many propose simple methods to obtain set representations by applying non-linear transformations to each element before a pooling layer (Ravanbakhsh et al., 2016; Qi et al., 2017b; Zaheer et al., 2017; Sannai et al., 2019). However, these models have limited expressive power. Other approaches such as Set Transformer (Lee et al., 2019) consider the pairwise interactions among set elements and hence can capture more complex statistics of set distributions. For large sets, the computational cost is expensive due to the self-attention operation.

Deep Learning Based Subsampling Interest in deep learning based subsampling methods has produced many works mostly applied to feature selection. In Balın et al. (2019), continuous approximation of the Concrete Distribution (Maddison et al., 2017) is used for *global* feature selection where a fixed set of features are sampled across an entire dataset. In Chen et al. (2018), instance-wise feature selection is used for interpretation of deep learning models applied to medical data. Dovrat et al. (2019) propose learning to subsample by generating virtual points, then matching them back to the original input. Several works (Qi et al., 2017a;c; Li et al., 2018b; Eldar et al., 1997; Moenning & Dodgson, 2003) also propose *farthest point sampling*, which selects k points from an input by ensuring that the selected samples are far from each other on a metric space. However our work is most similar to the recent works of Yoon et al. (2019) and Huijben et al. (2019) which learn a subsampling model conditioned on an given task. However these models have limitations both in terms of their range of applicability and poor performance under extremely low subsampling rates. Our method on the other hand is flexible, performs well under extremely low subsampling rates, and applicable to a wide range of subsampling problems.

Image Compression Due to the huge demand for data trans-

fer over the internet, some works attempt to compress images with minimal distortion. These models (Toderici et al., 2017; Rippel & Bourdev, 2017; Mentzer et al., 2018; Li et al., 2018a) typically consist of an encoder and decoder, where the encoder transforms the image with a compact matrix and the decoder reconstructs the image. These methods, while highly successful for the image compression problem, are less flexible than ours. Our model can be applied to arbitrary set structured data while the aforementioned models mainly work for images represented in tensor form.

Active Learning Active learning aims to select data points for labeling given a small labeled set. This domain is different from ours since active learning does not consider the label information for the selected data points but our method does utilize label information. Also, our motivation is quite different. We focus on optimal subsampling conditioned on an arbitrary task and this greatly differs from the goal of active learning. Methods such as (Sener & Savarese, 2018; Coleman et al., 2020; Wei et al., 2015) all tackle the data selection problem in the active learning setting.

Core-set Selection Core-set (Feldman, 2020) methods aim at selecting a small *weighted* subset of a given dataset that approximates the full *dataset* with theoretical guarantees. They are mostly targeted at instance selection and not, in generally, applicable to feature selection. Although we can utilize our subsampling method, SSS, for some instance selection tasks, our subsamplimg method, as well as those of Huijben et al. (2019) and Yoon et al. (2019), is *not* a core-set selection method. Ours is based on a data driven approach, where we leverage expressive neural networks to learn to subsample the most representative subset for various downstream tasks.

3. Approach

3.1. Preliminaries

We consider a set $D = \{d_i\}_{i=1}^n$ as an input where each d_i either represents an instance-label pair (x_i, y_i) or a *feature* such as the pixel value of an image. We cast the subsampling problem as the selection of a subset $D_s = \{s_j\}_{j=1}^k \subset D$ with $k \ll n$ such that $\ell(\cdot, D) \approx \ell(\cdot, D_s)$ for an arbitrary loss function $\ell(\cdot, D)$ over the full set D. In order to apply set functions to the subsampling problem, we need to properly design the neural network components to have some symmetrical properties such as permutation invariance (Definition 3.2), equivariance (Definition 3.3), and exchangeability (Definition 3.4).

Definition 3.1 (Permutation)**.** We say a function π is a permutation iff $\pi \in \mathfrak{S}_n = \{f : [n] \to [n] \mid f \text{ is bijective}\}$.

Definition 3.2 (Permutation Invariance)**.** We say a function $f : X^n \to Y$ is permutation invariant iff $f(\pi(\mathbf{x})) = f(\mathbf{x})$ for all $\pi \in \mathfrak{S}_n$ and for all $\mathbf{x} \in X^n$.

Definition 3.3 (Permutation Equivariance)**.** We say $f : X^n \to Y^n$ is permutation equivariant iff $\pi(f(\mathbf{x})) = f(\pi(\mathbf{x}))$ for all $\pi \in \mathfrak{S}_n$ and for all $\mathbf{x} \in X^n$.

Definition 3.4 (Exchangeability)**.** A distribution for a set of random variables $X = \{\mathbf{x}_i\}_{i=1}^n$ is exchangeable iif $p(X) = p(\pi(X))$ for all $\pi \in \mathfrak{S}_n$

In the following sections, we propose a two-stage Set based Stochastic Subsampling (SSS) method that leverages permutation invariant and equivariant set functions parameterized by θ to learn the conditional distribution $p_\theta(D_s|D)$. The first stage, *candidate selection*, and the second stage, *autoregressive subset selection*, are illustrated in Fig. 1. In general, we estimate the parameters of the subsamping model θ by minimizing the following loss: $\mathbb{E}_{p(D)}[\mathbb{E}_{p_\theta(D_s|D)}[\ell(\cdot, D_s)]]$, where $p(\cdot)$ denotes some unknown data distribution.

3.2. Set based Stochastic Subsampling

To select D_s, we propose to model the pairwise interactions among the elements of D and then choose a few representative elements in D based on the relative sample importance computed from the interaction scores. However, when the cardinality of D is large, modeling pairwise interactions becomes computationally infeasible since we need to *compare each element in D with all the other elements*. This computational bottleneck motivates the first stage of SSS of which the goal is to construct a smaller subset D_c, which we refer to as the *candidate set*, at a coarse level without considering pairwise interaction. We call the first stage *candidate selection* and the second stage, which is more fine-grained, *autoregressive subset selection* and selects D_s from D_c.

3.3. Candidate Selection

We formulate the candidate selection problem as a random Bernoulli process where the parameters of the Beronulli distribution are conditioned on the *set representation* of D and the individual elements $d_i \in D$. Specifically, we first encode the set D to a single representation D_e with a set encoding function (see Fig. 1-(a)) as follows:

$$D_e = \frac{1}{n}\sum_{i=1}^{n} g(d_i), \quad n = |D| \tag{1}$$

where g is a neural network which projects each element in D independently to a lower dimension. D_e captures coarse-grained global information in D with computational efficiency. This encoding scheme is similar to DeepSets (Zaheer et al., 2017) except that we do not perform message-passing, which is computationally expensive, between the set elements.

Proposition 3.5. *Given the set D and the affine transformation with non-linearity g, the set encoding D_e in Eq. 1 is permutation invariant.*

We then concatenate every $g(d_i)$ with D_e, denoted as $\overline{d_i}$. That is, $\overline{d_i} = [d_i, D_e]$, where $[\,]$ is the concatenation operation. This ensures that each element of D has a *global* view of all the other elements in the set at a coarse level. For each $d_i \in D$, we sample a mask $z_i \sim p_\theta(z_i|d_i, D)$ with

$$p_\theta(z_i|d_i, D) = \text{Ber}(z_i; \rho(\overline{d_i})), \quad \rho(\overline{d_i}) = \sigma(h(\overline{d_i})) \quad (2)$$

where h is a neural network that outputs the logits for the probability that d_i is in the candidate set D_c and $\sigma(\cdot)$ is the sigmoid function, and Ber denotes the Bernoulli distribution. z_i is a binary random variable where $z_i = 1$ indicates that d_i is an element in D_c. We concatenate all z_i's to obtain a single vector $Z = [z_1, \ldots, z_n]$. Since sampling from the Bernoulli distribution is not differentiable, during training, we use the continuous relaxations of the Bernoulli distribution (Maddison et al., 2017; Jang et al., 2017; Gal et al., 2017) to sample z_i for each d_i. This is illustrated as Mask Sampling in Fig. 1-(a). Although pairwise interactions are not considered in this stage, the ablation studies (Appendix D) show that learning $p_\theta(z_i|d_i, D)$ leads to selecting highly informative samples compared to random selection of the candidate set D_c.

Constraining the size of D_c For computational efficiency, we want to restrict the size of D_c to save computational cost when constructing D_s. Hence we introduce a sparse Bernoulli prior $p(Z) = \prod_{i=1}^n \text{Ber}(z_i; r)$ with small $r > 0$ and minimize the KL divergence along with a target downstream task loss $\ell(\cdot, D_s)$ w.r.t θ as follows:

$$\mathbb{E}_{p(D)}\left[\mathbb{E}_{p_\theta(D_s|D)}[\ell(\cdot, D_s)] + \beta \text{KL}[p_\theta(Z|D)||p(Z)]\right] \quad (3)$$

where $p_\theta(Z|D) = \prod_{i=1}^n p_\theta(z_i|d_i, D)$ and $\beta > 0$ is a hyperparmeter used to control the sparsity level in Z.

Proposition 3.6. *The candidate selection function which outputs the probability for each element D is permutation equivariant and the probability $p_\theta(Z|D)$ is exchangeable.*

3.4. Autoregressive Subset Selection

At this stage in the pipeline, we have a set D_c with $m = |D_c| \ll |D|$, which is small enough to perform fine-grained subset selection through pairwise modeling. To select a subset with k elements from D_c, we require k iterative steps. As shown in Fig. 1-(b), at time step t, we have the subset $D_s^{(t-1)}$ constructed from the previous iteration with $D_s^{(0)} = \emptyset$ and $D_c^{(t)} = \{s_1^{(t)}, \ldots, s_{m_t}^{(t)}\} = D_c \setminus D_s^{(t-1)}$. Assuming we have a function $\varphi \circ f$ (Set Attention in Fig 1-(b)) for modeling pairwise interactions between the elements of an input set, we autoregressively compute the interaction scores at time step t as follows:

$$\tilde{\pi}^{(t)} = (\tilde{\pi}_1^{(t)}, \ldots, \tilde{\pi}_{m_t}^{(t)}) = \sigma(\varphi \circ f(D_c^{(t)}, D_s^{(t-1)})) \quad (4)$$

where $\sigma(\cdot)$ denotes the sigmoid function and $\varphi \circ f$ is a composition of two neural networks: f computes interaction scores between elements in $D_c^{(t)}$ and φ outputs element-wise logits using the interaction scores. Further, $\tilde{\pi}^{(t)}$ is the vector of interaction scores for all elements in $D_c^{(t)}$ at the current time step t. Given $\tilde{\pi}_i^{(t)} > 0$ for all $i = 1, \ldots, m_t$, we can compute the probability of an element $s_i^{(t)}$ being selected from $D_c^{(t)}$ as:

$$p_\theta(s_i^{(t)}|D_c^{(t)}, D_s^{(t-1)}) = \pi_i^{(t)}, \quad \pi_i^{(t)} = \frac{\tilde{\pi}_i^{(t)}}{\sum_{j=1}^{m_t} \tilde{\pi}_j^{(t)}}, \quad (5)$$

where $m_t = |D_c^{(t)}|$. That is, we normalize $\tilde{\pi}^{(t)}$ over all the elements in $D_c^{(t)}$ at time step t to obtain a valid probability distribution. The key to avoiding redundant elements in D_s lies in the fact that for each element added to D_s, its selection is conditioned on both the candidate set $D_c^{(t)}$ and all the elements in the subset $D_s^{(t-1)}$ as described in Eq. 4 & 5. For the choice of the function f, we use a MultiHead Attention Block (MAB) (Lee et al., 2019) which we describe in detail in **Appendix** I. Additionally, we can stack multiple MABs for the function f to model higher level interactions.

Proposition 3.7. *The functions φ and f, and the pairwise interaction score $\tilde{\pi}_t$ are permutation equivariant for all time steps t in the autoregressive subset selection stage.*

With Eq. 5, we can sample an element $s^{(t)} \sim \text{Cat}(\pi_1^{(t)}, \ldots, \pi_{m_t}^{(t)})$ from the candidate subset $D_c^{(t)}$ and construct $D_s^{(t)} = D_s^{(t-1)} \cup \{s^{(t)}\}$, where Cat is the Categorical distribution. During training, it can be expensive to sample k times from the Categorical distribution since it involves computing Eq. 4 k times. We remedy this by selecting l elements from $D_c^{(t)}$ at once, which reduces the number of iterations to k/l for selecting k elements. We may also sample l elements from the multinomial distribution with probability $\pi^{(t)}$ without replacement. However, this sampling procedure is non-differentiable, and hence it cannot be trained with backpropagation. Instead, we independently sample l elements from the continuous relaxation of Categorical distributions (Maddison et al., 2017; Jang et al., 2017) using the same probabilities in Eq. 5 to approximate sampling from the multinomial distribution as shown in Fig. 1-(b). Since we want to simulate sampling without replacement, we discard all elements sampled more than once. This sampling procedure guarantees that we get at most l elements at each iteration. A similar sampling procedure is adopted in previous works (Balın et al., 2019; Chen et al., 2018). We detail this training algorithm in **Appendix** B.

Proofs of Propositions 3.5, 3.6 & 3.7 are in Appendix E.

Time Complexity The time complexity of SSS depends heavily on the choice of the function f. Using MAB as f, the time complexity of SSS is $O(n) + O(k^2 m/l)$ where n, m, k correspond to $|D|$, $|D_c|$ and $|D_s|$ respectively.

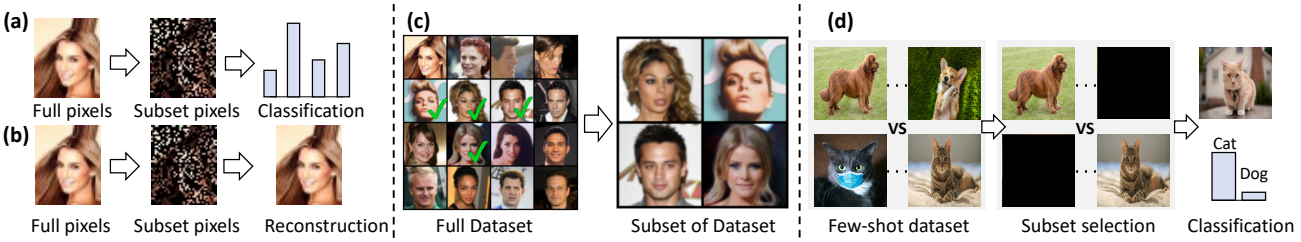

Figure 2. **Target Tasks: (a)** Feature selection for reconstruction. **(b)** Feature selection for prediction. **(c)** Selection of representative instances. **(d)** Instance selection for few-shot classification.

3.5. Tasks

Set Classification & Prediction As shown in Fig. 2-(a), we train a neural network parmeterized with ϕ to predict a single target value y_D for the subset D_s of the given full set D, where D is a collection of the features from a single instance such as the pixels of an image. For instance, the target y_D is either the class of an image for classification or the attributes of a face in an image. Here, our goal is learning to select the most representative subset $D_s \subset D$ such that we can maximize the log likelihood $\log p_\phi(y_D|D_s)$ with computational efficiency. In order to achieve this goal, we jointly train the SSS model and the neural network which predicts the target value y_D for D_s to minimize the negative log-likelihood, the loss function $\ell(\cdot, D_s)$ described in Eq. 3 and KL divergence to enforce sparsity in Z (the selection masks for the candidate set) as follows:

$$\mathbb{E}_{p(D)}[\mathbb{E}_{p_\theta(D_s|D)}[-\log p_\phi(y_D|D_s)] + \beta \mathrm{KL}[p_\theta(Z|D)||p(Z)]] \tag{6}$$

where $p(Z) = \prod_{i=1}^n \mathrm{Ber}(z_i; r)$ with small $r > 0$. We provide experimental results in Section 4.2 and a corresponding graphical model in **Appendix** F.

Set Reconstruction Given a full set $D = \{X, Y\}$ consisting of 2d coordinates $X = \{x_i \in \mathbb{R}^2\}_{i=1}^n$ and corresponding pixel values $Y = \{y_i \in \mathbb{R}^3\}_{i=1}^n$, we want to select the most representative subset $D_s = \{X_s, Y_s \mid X_s \subset X, Y_s \subset Y\}$ to reconstruct all pixel values $y_i \in Y$ for each $x_i \in X$, as shown in Fig. 2-(b). We jointly train the SSS model and a neural network with parameters ϕ predicting pixel values to minimize the loss function w.r.t θ and ϕ as follows:

$$\mathbb{E}_{p(D)}[\mathbb{E}_{p_\theta(D_s|D)}[-\log p_\phi(Y|X, D_s)] + \beta \mathrm{KL}[p_\theta(Z|D)||p(Z)]] \tag{7}$$

We enforce sparsity on the subset D_s by minimizing the KL-divergence between the mask probability $p_\theta(Z|D)$ and sparse prior $p(Z) = \prod_{i=1}^n \mathrm{Ber}(z_i; r)$ with small $r > 0$. Moreover, minimizing the negative log likelihood, which corresponds to $\ell(\cdot, D_s)$ in Eq. 3, ensures that the constructed D_s is the most representative for the downstream tasks. We implement $p_\theta(Y|X, D_s)$ as an Attentive Neural Process (ANP) (Kim et al., 2019). The ANP takes D_s as input and predicts a distribution of the elements in the original set D. It mimics the behaviour of a Gaussian Process but with reduced inference complexity. We present experimental results for this task in Section 4.3 and a corresponding graphical model depiction in **Appendix** F.

Dataset Distillation: Instance Selection In this task, we are given a collections of datasets $\mathcal{D} = \{D^{(1)}, \ldots, D^{(m)}\}$ with $D^{(i)} \cap D^{(j)} = \emptyset$ for $i \neq j$ and $D^{(i)} \overset{iid}{\sim} p(D)$. The goal is to select the most representative subset $D_s^{(i)}$ with $|D_s^{(i)}| \ll |D^{(i)}|$ for each dataset $D^{(i)} = \{d_1^{(i)}, \ldots, d_n^{(i)}\} \in \mathcal{D}$, where $d_i^{(i)}$ is a data point uniformly sampled from the entire datasets $\mathcal{D}$. Using CelebA dataset as an illustrative example, shown in Fig. 2-(c), $D^{(i)}$ consists of n randomly sampled faces from the entire dataset and the task is to construct a subset, $D_s^{(i)}$, most representative of $D^{(i)}$.

In order to learn to select the subset D_s from each $D \in \mathcal{D}$ with *unsupervised learning*, we jointly train the SSS model and a generative model such that the SSS model chooses the most representative subset so that the generative model can reconstruct all the images $d_i \in D$ from the subset. Naïvely, we can minimize the sum of negative log-likelihood $\sum_{d_i \in D} -\log p_\phi(d_i|D_s)$ for the loss function $\ell(\cdot, D_s)$ and KL divergence in Eq. 3. However, we find that the generative model outputs mean images for all d_i. To capture variations of different images, we introduce three latent variables α_i, c_i, and w_i which both depend on d_i. We provide graphical model illustration of this task in the **Appendix** F. Since it is intractable to compute the log likelihood $\log p_\phi(d_i|D_s)$ by marginalizing over all the latent variables, we derive the upper bound of the marginal negative log likelihood using variational inference and plug the upper bound into the loss function $\ell(\cdot, D_s)$ in Eq. 3 as follows:

$$\mathbb{E}_{p(D)}\Bigg[\mathbb{E}_{p_\theta(D_s|D)}\Big[\sum_{d_i \in D}[\mathbb{E}_{q_\psi(w_i, c_i|d_i, D_s)}[-\log p_\phi(d_i|w_i, c_i)] + \mathrm{KL}[q_\psi(w_i|d_i)||p_\xi(w_i)] + \mathrm{KL}[q_\psi(\alpha_i|d_i)||p_\xi(\alpha_i)] + \mathrm{KL}[q_\psi(c_i|D_s, \alpha_i)||p_\xi(c_i)]]\Big] + \beta \mathrm{KL}[p_\theta(Z|D)||p(Z)]\Bigg] \tag{8}$$

where $p_\xi(\cdot)$ are priors on their respective latent variables, $p(Z) = \prod_{i=1}^n \mathrm{Ber}(z_i; r)$ is sparse prior with small $r > 0$

over the mask for candidate set selection in SSS, $p_\phi(\cdot)$ is the decoder to reconstruct d_i, and all variational posteriors $q_\psi(\cdot)$ are parameterized with neural networks. All priors are the standard normal distribution.

In summary, we jointly train both the SSS and generative model to minimize the objective in Eq. 8 w.r.t θ, ϕ, and ψ for all $D \in \mathcal{D}$ and leverage the optimized SSS to select a few representative instances of the dataset, which results in distilled dataset. Experimental results are in Section 4.4.

Dataset Distillation: Classification Finally for the dataset distillation task, we consider the problem of selecting prototypes for few-shot classification as shown in Fig. 2-(d). We adopt Prototypical Networks (Snell et al., 2017) and deploy the SSS model for selecting representative prototypes from the support set for each class. We minimize the objective in Eq. 3, where we use the distance loss induced by the metric space from Prototypical Networks for the target task loss $\ell(\cdot, D_s)$, to jointly train the Prototypical Networks and SSS. Note that we use D_s, the subset of the support set, for computing loss and prediction. By learning to select the prototypes, we can remove outliers that would otherwise change the decision boundaries in the classification task where we need to predict the label y_* for an unseen instance x_*. Experimental results for this task are in Section 4.4 and its graphical model description is in **Appendix** F.

4. Experiments

An extensive **Ablation** on SSS can be found in **Appendix D**.

4.1. Baselines

Feature Selection for Classification We compare SSS with the following models on MNIST. 1) **Random Selection**: it randomly subsamples features. 2) **DPS** (Huijben et al., 2019): this model jointly optimizes the sampling parameters along with the parameters of a task model. 3) **INVASE** (Yoon et al., 2019): this model uses actor-critic (Peters & Schaal, 2008) to optimize the parameters of a sampling network and target task model (Section 4.2).

Additionally, we perform attribute classification on the CelebA dataset using the selected features from a given image. However, we *cannot* apply DPS and INVASE to this experiment. Since each feature is a tuple of 3-d RGB pixels, flattening these features results in ambiguities as to which features to select with INVASE or DPS. For instance, applying these methods to a 3-channel image can result in some channels being selected by the models, while others are zeroed out. It might lead to the entire pixel being preserved and hence violate the subsampling objective.

Feature Selection for Reconstruction In Section 4.3 we compare SSS against the followings. 1) **Random Selection**. 2) **LTS** (Dovrat et al., 2019): a model that learns to generate k virtual elements which can be matched to elements in D and optimized for the downstream task. We use LTS for both the function reconstruction and the image reconstruction tasks. DPS and INVASE are not applicable for these tasks since each feature is multi-dimensional. Note that LTS is not applicable for the classification tasks since the virtual points generated by LTS cannot be converted back into image form to serve as input to an image classifier.

Instance Selection In Section 4.4, we compare SSS with 1) **k-Center-Greedy**: this algorithm iteratively selects elements in D closest to a set of centroids and 2) **FPS**: this algorithm iteratively selects the most distant elements to a randomly initialized D_s and **3) Random Selection** on the instance selection tasks. Here also, DPS, INVASE, and LTS are all inapplicable.

Multiple Subsampling Rates For all the neural network based baselines (DPS, INVASE, LTS), we train a separate model for *every* subsampling rate. For instance, to select 15, 20, 25, 30, 50 and 100 pixels from MNIST images in Section 4.2, we need to train 6 different models for DPS, INVASE and LTS each with the corresponding target subsampling rate. However for SSS, we train a *single* model and vary the sampling rate on each iteration. During evaluation, we use this single model for *all* the different sampling rates. We find that applying a similar training technique to the baselines result in drastic performance degradation. Thus the set formulation of SSS makes it generalize to varying subsampling rates at test time with train time efficiency.

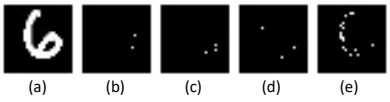

Figure 5. **(a)** Full-image, **(b)** Random, **(c)** DPS, **(d)** INVASE, **(e)** SSS. All models select 15 pixels of the original image. Note that under these low subsampling rates, the baseline models end up selecting background pixels as shown in **(c)** and **(d)**.

4.2. Feature Selection for Classification

In this subsection, we validate our model on the image classification task with feature selection as illustrated in Fig. 2-(a). The goal is to select a subset of pixels of an image and predict the label using the chosen subset.

MNIST Given an MNIST image with 784 pixels, the task is to subsamlpe 15, 20, 25, 30, 50 and 100 pixels to be used as input to train and evaluate two classification models, a MLP and a ConvNet. We detail the exact architectures in Appendix I.2. Since images in the MNIST dataset have single channel, each feature is of dimension 1 and thus we can train a classifier with DPS or INVASE. We keep the pixels values for the selected pixels and set all the other pixels to zero. Note that we set the subsampling rates to be much lower than the experimental setup in Huijben et al.

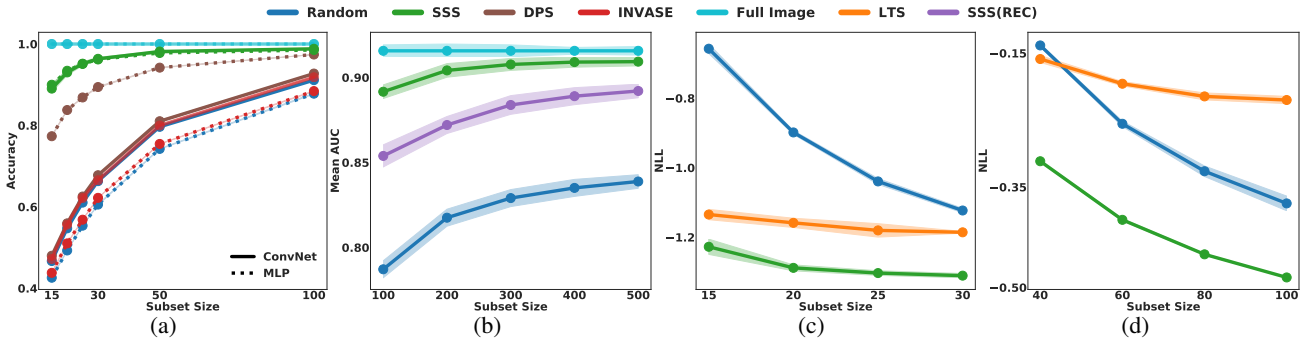

Figure 3. **(a)** MNIST Classification. **(b)** CelebA classification. **(c)** 1D Function Reconstruction. **(d)** Image Reconstruction on CelebA.

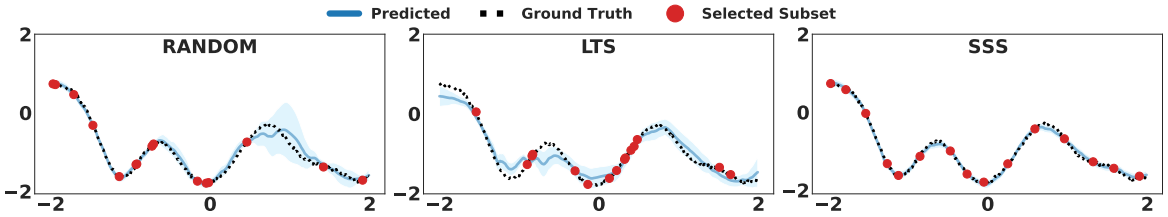

Figure 4. Visualization of 1D function reconstruction with three different subset selection models.

(2019) and Yoon et al. (2019).

As shown in Fig. 3a, SSS significantly outperforms all the baselines with large margin on both the MLP and ConvNet architectures. The MLP is the same architecture used in Huijben et al. (2019). However, we test on the full MNIST test set instead of reserving half the test set for validation as done in Huijben et al. (2019). SSS reaches 89% accuracy using only 15 pixels and shows better performance than the baselines with 100 pixels. Moreover under these low subsampling rates, the performance on the baselines are on par with random selection as shown by the ConvNet results in Fig. 3a. Crucially, the performance of SSS is consistent across both the MLP and ConvNet architectures. DPS, which performs relatively well using the MLP shows poor performance on the same dataset using the ConvNet architecture and we observe similar drop in performance as well for INVASE.

Lastly, we provide qualitative results in Fig. 5 where we visualize the 15 selected pixels by all the baselines and SSS. Again we find that SSS selects representative pixels (Fig. 5-(e)) so that the classifier can predict the correct label of the input image. However, all the baselines tend to select background pixels under these extremely low subsampling rates, which are uninformative for the classification task.

CelebA The CelebA dataset consists of high-quality images of size 218×178. Like the previous experiment, the task is to subsample 100, 200, 300, 400, and 500 pixels from the full 38804 pixels and perform binary classification for 40 attributes of a face. We use the ConvNet architecture described in Appendix I.3 as the classification network.

We report the mean AUC score on all 40 attributes for varying sizes of D_s. Fig. 3b shows that using only 500 pixels ($\sim$1.3% of total pixels in an image), SSS achieves a mean AUC of 0.9093 (99.3% of the accuracy obtained with the full image). SSS achieves a significantly higher AUC score than Random Selection, showing the effectiveness of our subset selection method. We also include another baseline, namely **SSS-rec**. This is the SSS model trained for image reconstruction in the Section 4.3, but then later used for classification without any finetuning. Our model also outperforms this variant, showing the effectiveness of training with the target task. Note that we cannot apply LTS, INVASE, or DPS to this experiment. During training, the virtual points generated by LTS cannot be converted back to an image in matrix form due to the virtual coordinate, thus we cannot train the LTS model with CNN-based classification for this task. For DPS and INVASE, they require the dimension of each feature to be 1, thus it is not applicable to multi-channel images.

4.3. Feature Selection for Regression

Function Reconstruction Suppose that we have a function $f : [a, b] \to \mathbb{R}$. We first construct a set of data points with $D = \{(x_1, y_1 = f(x_1)), \ldots, (x_n, y_n = f(x_n))\}$, where $(x_1, \ldots, x_n)$ are uniformly sampled from the interval $[a, b]$ and f is a Gaussian process. We sample $(y_1^{(i)}, \ldots, y_n^{(i)}) \overset{iid}{\sim} \mathcal{N}(\mathbf{0}, K_{XX} + \sigma_y^2 I_n)$ for $i = 1, \ldots, N$ where K_{XX} is a squared-exponential kernel with the set of inputs $X = \{x_1, \ldots, x_n\}$ and σ_y^2 is variance for small likelihood noise. This leads to a collection of sets $(D^{(1)}, \ldots, D^{(N)})$. We train our model which consists of the subset selection model $p_\theta(D_s|D)$ and a task network $p_\phi(Y|X, D_s)$, which is an Attentive Neural Process (ANP) (Kim et al., 2019) on this dataset and report the negative log-likelihood (NLL).

Table 1. FID Score (the lower is the better) with varying the number of instances

#Instances	2	5	10	15	20	30
K-Greedy	8.8800 ± 5.5857	4.4306 ± 1.3313	4.2199 ± 1.4214	3.7160 ± 1.1314	3.2431 ± 1.3881	2.7554 ± 0.8554
FPS	6.5014 ± 4.3502	4.5098 ± 2.3809	3.0746 ± 1.0979	2.7458 ± 0.6201	2.7118 ± 1.0410	2.2943 ± 0.8010
Random	3.7309 ± 1.1690	1.1575 ± 0.6532	0.8970 ± 0.4867	0.3843 ± 0.2171	0.3877 ± 0.1906	0.1980 ± 0.1080
SSS	**2.5307 ± 1.3583**	**1.0186 ± 0.1982**	**0.5922 ± 0.3181**	**0.3331 ± 0.1169**	**0.2381 ± 0.1153**	**0.1679 ± 0.0807**

Table 2. Accuracy on *mini*ImageNet

#Instances	1	2	5
FPS	0.432±0.005	0.501±0.002	0.598±0.000
Random	0.444±0.003	0.525±0.005	0.618±0.003
K-Greedy	0.290±0.006	0.413±0.005	0.570±0.002
SSS	**0.475±0.006**	**0.545±0.011**	**0.625±0.006**

Fig. 3c shows the performance (NLL) of SSS compared to the baselines, Random Selection and LTS. As shown in Fig. 3c, SSS outperforms the baselines, verifying that the subset selection model $p_\theta(D_s|D)$ learns a meaningful distribution over subsets. We visualize a reconstructed function and the selected points by each models in Fig. 4. As shown in the rightmost figure (Fig. 4), SSS tends to pick out more elements (red dots) in the drifting parts of the curve, which is reasonable since those are harder to reconstruct than the others. However, the other baselines sometimes fail to do that, which leads to inaccurate reconstructions.

CelebA Image Reconstruction Given an image, we learn to select a representative subset of pixels that best reconstructs the original image. Here, x_i is the 2d pixel coordinates and $y_i \in \mathbb{R}^3$ is the RGB pixel value. We use an ANP to reconstruct the remaining pixels from the subset $D_s = \{x_{i_l}, y_{i_l}\}_{l=1}^k$ constructed by each subsampling model. We conduct the experiment on the CelebA dataset (Liu et al., 2018). Fig. 3d shows that our model significantly outperforms Random Selection and LTS in terms of NLL. We provide qualitative examples in **Appendix** G.2.

Note that LTS performs worse than random selection. We find that the generated coordinates values by LTS are imprecise. This makes matching the virtual points with the original pixel values extremely difficult. Such inaccurate coordinate values result in poor performance as the subsampling rate increases even compared to random subsampling as depicted in Figure 3d. We observe similar pattern in the CIFAR10 reconstruction task presented in Figure 15a in **Appendix** J. On the other hand, for the function reconstruction task in Section 4.3, the point matching stage in LTS is fairly easy and hence the LTS model shows better performance than random subsampling.

Efficiency in Nonparametric models In all the experiments where we used an ANP, we greatly improve the inference time complexity. By design, these models need to leverage the full training data at inference time. However by subsampling few highly informative instances, we can efficiently perform inference with little degradation in accuracy. Similar gains can be obtained for models in the Neural Process family of models (Garnelo et al., 2018a;b).

4.4. Dataset Distillation

Instance Selection The goal is to select only a few representative images from a given dataset as described in Section 3.5 and Fig. 2. We split the CelebA dataset into m disjoint sets $\mathcal{D} = \{D^{(1)}, \dots, D^{(m)}\}$ and jointly train SSS and the generative model to minimize the objective in Eq. 8 with $\mathcal{D}$. After training, we discard the generative model and leverage the subsampling model to choose a few representative images from the full CelebA dataset.

We evaluate the selected subset with the Fréchet Inception Distance (FID) (Heusel et al., 2017), which measures similarity and diversity between two datasets and compare SSS to k-Center-Greedy, FPS and Random Selection. We report the experimental results in Table 1 where SSS achieves the lowest FID score for all selection sizes. Specifically, SSS outperforms all the baselines for selecting very few instances since SSS is able to model the interactions within the dataset and hence selects the most representative subset. Additionally, given that the dataset is highly imbalanced, k-Center-Greedy and FPS perform worst since by selecting extreme or similar elements in the given set and cannot capture the true representation of the full dataset. We provide selected images by SSS from the full dataset in **Appendix** H.

Classification In this task, we perform few-shot classification with the *mini*ImageNet dataset (Vinyals et al., 2016) where the models select 1, 2, or 5 instances from the support set with size 20. As shown in Table 2, we compare SSS against Random Selection, FPS, and k-Center-Greedy. SSS learns to select more representative prototypes than the others especially for small D_s where the choice of prototypes matters more. Notably, the K-Greedy method performs poorly for small subset sizes given that the model overfits to a few samples and does not generalize to unseen examples. We show samples of selected prototypes in **Appendix** H.1.

5. Conclusion

In this paper, we reformulated the subsampling problem as the selection of a subset from a set (e.g features and instances). Based on this reformulation, we proposed a Set based Stochastic Subsampling method that can han-

dle arbitrary input structure as well as variable input set sizes. Additionally, to reduce the cost of modeling pairwise-interactions for large sets, we devised a two-stage subsampling algorithm where we utilize set encoding functions to obtain coarse grained global information in the candidate selection stage followed by a more expressive set interaction network in the autoregressive subset selection stage. We validated the efficacy and generality of our model on various tasks such as feature selection for classification and set reconstruction, instance selection for few shot classification and dataset distillation. We demonstrated that SSS works well and outperforms the relevant baselines.

Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program(KAIST)), the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No. 2021-0-02068, Artificial Intelligence Innovation Hub), the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2021R1F1A1061655), and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2022-0-00713).

References

Balın, M. F., Abid, A., and Zou, J. Concrete autoencoders: Differentiable feature selection and reconstruction. In *International Conference on Machine Learning*, pp. 444–453. PMLR, 2019.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In *International Conference on Machine Learning*, pp. 883–892. PMLR, 2018.

Coleman, C., Yeh, C., Mussmann, S., Mirzasoleiman, B., Bailis, P., Liang, P., Leskovec, J., and Zaharia, M. Selection via proxy: Efficient data selection for deep learning. In *International Conference on Learning Representations*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dovrat, O., Lang, I., and Avidan, S. Learning to sample. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2760–2769, 2019.

Eldar, Y., Lindenbaum, M., Porat, M., and Zeevi, Y. Y. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 1997.

Feldman, D. Introduction to core-sets: an updated survey. *arXiv preprint arXiv:2011.09384*, 2020.

Gal, Y., Hron, J., and Kendall, A. Concrete dropout. In *Advances in neural information processing systems*, pp. 3581–3590, 2017.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y. W., Rezende, D., and Eslami, S. A. Conditional neural processes. In *International Conference on Machine Learning*, pp. 1704–1713. PMLR, 2018a.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018b.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pp. 6626–6637, 2017.

Huijben, I. A., Veeling, B. S., and van Sloun, R. J. Deep probabilistic subsampling for task-adaptive compressed sensing. In *International Conference on Learning Representations*, 2019.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations*, 2017.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. W. Attentive neural processes. In *International Conference on Learning Representations*, 2019.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 2009.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Li, M., Zuo, W., Gu, S., Zhao, D., and Zhang, D. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3214–3223, 2018a.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pp. 820–830, 2018b.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Liu, Z., Luo, P., Wang, X., and Tang, X. Large-scale celebfaces attributes (celeba) dataset. *Retrieved August*, 2018.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations*, 2017.

Mentzer, F., Agustsson, E., Tschannen, M., Timofte, R., and Van Gool, L. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4394–4402, 2018.

Moenning, C. and Dodgson, N. A. Fast marching farthest point sampling. Technical report, University of Cambridge, Computer Laboratory, 2003.

Peters, J. and Schaal, S. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017b.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pp. 5099–5108, 2017c.

Ravanbakhsh, S., Schneider, J., and Poczos, B. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.

Rippel, O. and Bourdev, L. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2922–2930. JMLR. org, 2017.

Sannai, A., Takai, Y., and Cordonnier, M. Universal approximations of permutation invariant/equivariant functions by deep neural networks. *arXiv preprint arXiv:1903.01939*, 2019.

Sener, O. and Savarese, S. Active learning for convolutional neural networks: A core-set approach. In *International Conference on Learning Representations*, 2018.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pp. 4077–4087, 2017.

Toderici, G., Vincent, D., Johnston, N., Jin Hwang, S., Minnen, D., Shor, J., and Covell, M. Full resolution image compression with recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5306–5314, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *NIPS*, 2017.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wei, K., Iyer, R., and Bilmes, J. Submodularity in data subset selection and active learning. In *International Conference on Machine Learning*, pp. 1954–1963, 2015.

Yoon, J., Jordon, J., and van der Schaar, M. INVASE: Instance-wise variable selection using neural networks. In *International Conference on Learning Representations*, 2019.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

A. Organization

Organization The Appendix is organized as follows: first, we describe the pseudo-code for the Training Algorithm, then provide ablation for each components of our method and proofs for propositions described in Section 3. Finally, we illustrate the generative process of each task using graphical models and and additional experimental results with detailed elaboration on experimental setups.

Algorithm 1 Greedy Training Algorithm

Input k (max subset size)
m (mini-batch size)
$p(D)$ (distribution of sets)
α (learning rate)
$\ell(\cdot, D_s)$ (loss function)
Output trained models with θ and ϕ

Randomly initialize parameter of SSS θ and downstream task model ϕ.
while not converged **do**
  Sample m sets $D^{(1)}, \dots, D^{(m)}$ from $p(D)$
  Sample $Z^{(j)} = \{z_1^{(j)}, \dots, z_n^{(j)}\} \sim p_\theta(Z|D^{(j)})$ for $j = 1, \dots, m$
  Construct $D_c^{(j)} = \{d_i^{(j)} \in D^{(j)} : z_i^{(j)} = 1\}$ for $j = 1, \dots, m$
  Sample integer $l \sim \text{Unif}[1, k]$
  $D_s^{(j)} \leftarrow$ select l-elements from $D_c^{(j)}$ (with the autoregressive model)
  $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{m} \sum_{j=1}^m \ell(\cdot, D_s^{(j)})$
  $\phi \leftarrow \phi - \alpha \nabla_\phi \frac{1}{m} \sum_{j=1}^m \ell(\cdot, D_s^{(j)})$
end while

B. Training Algorithm

In order to reduce the computational cost at training time, we use a greedy training algorithm with stochastic gradient descent as described in Algorithm 1. First, we uniformly sample an integer l from $\{1, \dots, k\}$, which is the subset size for a given mini-batch. Then we select l elements for D_S from the candidate set *at once* using the autoregressive selection model. Finally, we perform gradient descent with respect to the parameters of SSS model θ and target task network ϕ to minimize the target task loss on the selected subset D_s. As a result, we do not have to run the autoregressive model k time during training, which significantly reduces the computational cost during training.

C. Fixed-size Subset Selection

At test time, we run the fixed size subset selection algorithm to choose the most task relevant elements from the set D as described in Algorithm 2. We do not use the greedy training algorithm. Instead, we autoregressively select k elements from the candidate set D_c as described in line 13 from Algorithm 2 to construct the representative subset D_s.

Algorithm 2 Fixed Size Subsampling. k is the subset size. l is the number of elements to select at each iteration. D is the full set and $D_s \subset D$ is the final subset after running SSS.

1: **Input:** $k, l, D = \{d_1, \dots, d_n\}$
2: **Output:** $D_s = \{s_1, \dots, s_k\}$
3: **function** SSS(k, l, D)
4: $D_e \leftarrow \frac{1}{n} \sum_{i=1}^n g(d_i)$
5: $\overline{d_i} \leftarrow \text{Concat}(g(d_i), D_e)$
6: $z_i \sim \text{Ber}(z_i; \rho(\overline{d_i})$ for $i = 1, \dots, n$
7: $D_c \leftarrow \{d_i \in D \mid z_i = 1, 1 \le i \le n\}$
8: $D_s \leftarrow \emptyset$
9: **for** $t = 1$ **to** k/l **do**
10: $D_s \leftarrow D_s \cup \text{AUTOSELECT}(l, D_s, D_c)$
11: **end for**
12: **end function**
13: **function** AUTOSELECT$(l, D_s^{(t-1)}, D_c)$
14: $D_c^{(t)} = \{s_1^{(t)}, \dots, s_{m_t}^{(t)}\} \leftarrow D_c \setminus D_s^{(t)}$
15: $\tilde{\pi}_i^{(t)} \leftarrow \sigma(\varphi \circ f(s_i, D_s^{(t-1)}))$
16: $(\pi_1^{(t)}, \dots, \pi_{m_t}^{(t)}) \leftarrow \frac{1}{\sum_{j=1}^{m_t} \tilde{\pi}_j^{(t)}} (\tilde{\pi}_1^{(t)}, \dots, \tilde{\pi}_{m_t}^{(t)})$
17: $Q \leftarrow$ Sample l elements from $D_c^{(t)}$ with the probability $\pi^{(t)}$
18: return Q
19: **end function**

D. Ablation

We perform extensive ablation studies on the two-stage Set based Stochastic Susbsampling method using the function reconstruction tasks presented in Section 4.3. First, we explore the contribution of the candidate selection and autoregressive subset selection stages. Then, we verify the importance of stochasticity in SSS.

Random Selection with Autoregressive Subset Selection To show the importance of the candidate selection stage of SSS, we replace it with random selection (labelled **Random + Stage 2** in Fig. 2). As shown in Fig. 6, we find that while this model performs better than the model with only candidate selection, it performs worse than SSS (red line in Fig. 6) and the autoregressive subset selection stage used alone (green line in Fig. 6). We provide visualizations of the reconstructed functions in the second column of Fig. 8. Random selection in place of the candidlate selection can ignore elements from certain parts of the function and hence the autoregressive selection model cannot select elements from those regions for reconstruction. In short, filtering elements with candidate selection helps the autoregressive selection model to choose more informative instances from

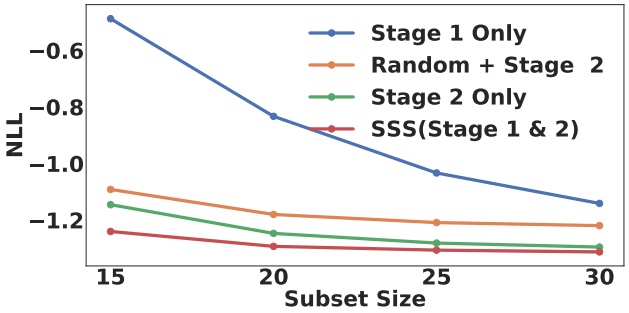

Figure 6. Ablation on SSS.

Table 3. CelebA Attributes Classification.

Model	# Pixels	Storage	mAUC
Full Image	All 38804	114KB	0.9157
RS	500	5KB	0.8471
SSS(rec)	500	5KB	0.8921
SSS(MC)	500	5*5KB	0.9132
SSS(ours)	500	5KB	0.9093

the input set than random selection.

AutoRegressive Subset Selection Only As shown in Fig. 6 we observe that the autoreressive subset selection model (labelled **Stage 2 Only** in Fig. 6) performs significantly better than the SSS model with candidate selection and autoregressive selection stage. Qualitative results are provided in third column of Fig. 8. While this model performs well, it is not very practical due to the high computational cost when the size of the set becomes large.

Candidate Selection Only In order to validate the importance of the autoregressive selection stage, we construct a subset using only the candidate selection stage. As shown in Fig. 6 (labelled **Stage 1 Only**), removing the autoregressive selection stage significantly degrades the performance of the SSS model. In the first column of Fig. 8, the model without autoregressive selection significantly underperforms compared to SSS since it heavily focuses on the drifting parts of the function and ignores the other parts of curve. In sum, it is not always desirable to select only highly activating samples in the set without considering any dependencies among the others since it may choose redundant elements.

Generally, we find that SSS performs better than the variants considered here and provide a better tradeoff between model performance and computational requirements.

Stochasticity of SSS Since our method is stochastic with the following predictive distribution: $\mathbb{E}_{p_\theta(D_s|D)}[p_\phi(y_D|D_s)]$, we approximate it with Monte

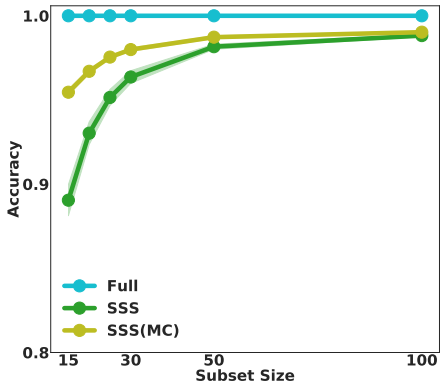

Figure 7. Accuracy with varying subset size and the number of particles for MCMC.

Carlo sampling as follows:

$$\mathbb{E}_{p_\theta(D_s|D)}[p_\phi(y_D|D_s)] \approx \frac{1}{n}\sum_{i=1}^{n} p_\phi(y_D|D_s^{(i)}), \quad \text{where } D_s^{(i)} \overset{i.i.d.}{\sim} p_\theta(D_s|D), \tag{9}$$

However in *all* experiments, we only report the result with one sampled subset, since it gives the best trade-off between computational cost and accuracy. We compare SSS against another variant, **SSS-MC**, which use 5 sampled subsets for MC sampling. As shown in Fig. 3b, it obtains a mean AUC of 91.32%, which is slightly better than SSS which achieves 90.93%. Note SSM-MC increases the computational cost (inference) and memory requirement up to 5 times as shown in Table 3. The result justifies the inference procedure of SSS which achieves good performance for the target tasks with memory and computational efficiency.

Additionally, in Figure 7 we show how the uncertainty of SSS decreases as we increases the subset size. For each subset size, we draw 5 different subsets from the subsampling model trained on the MNIST classification task described in Section 4.2 and average the predictions of the sampled subsets. This requires 5 forward pass of the model. At a subsampling rate of 50, the model with a single subset shows similar performance with the one with multiple draws of subsets.

Cost of Running SSS We report the cost of running SSS for pixel subsampling, which has the largest set size (38804 pixels). In this experiment, we select 500 pixels in total with $l = 20$, i.e., we select 20 pixels at once in the second stage described in Section 3.4. We measure the FLOPS and memory requirements for the forward pass of SSS. We find that the computational cost of running SSS is 8.38 GMac (40% of the FLOPS for the full model which is 20 GMac) with 217.09k memory requirement (compared to 958.85k for the full model) which shows that SSS is computationally cheap to run.

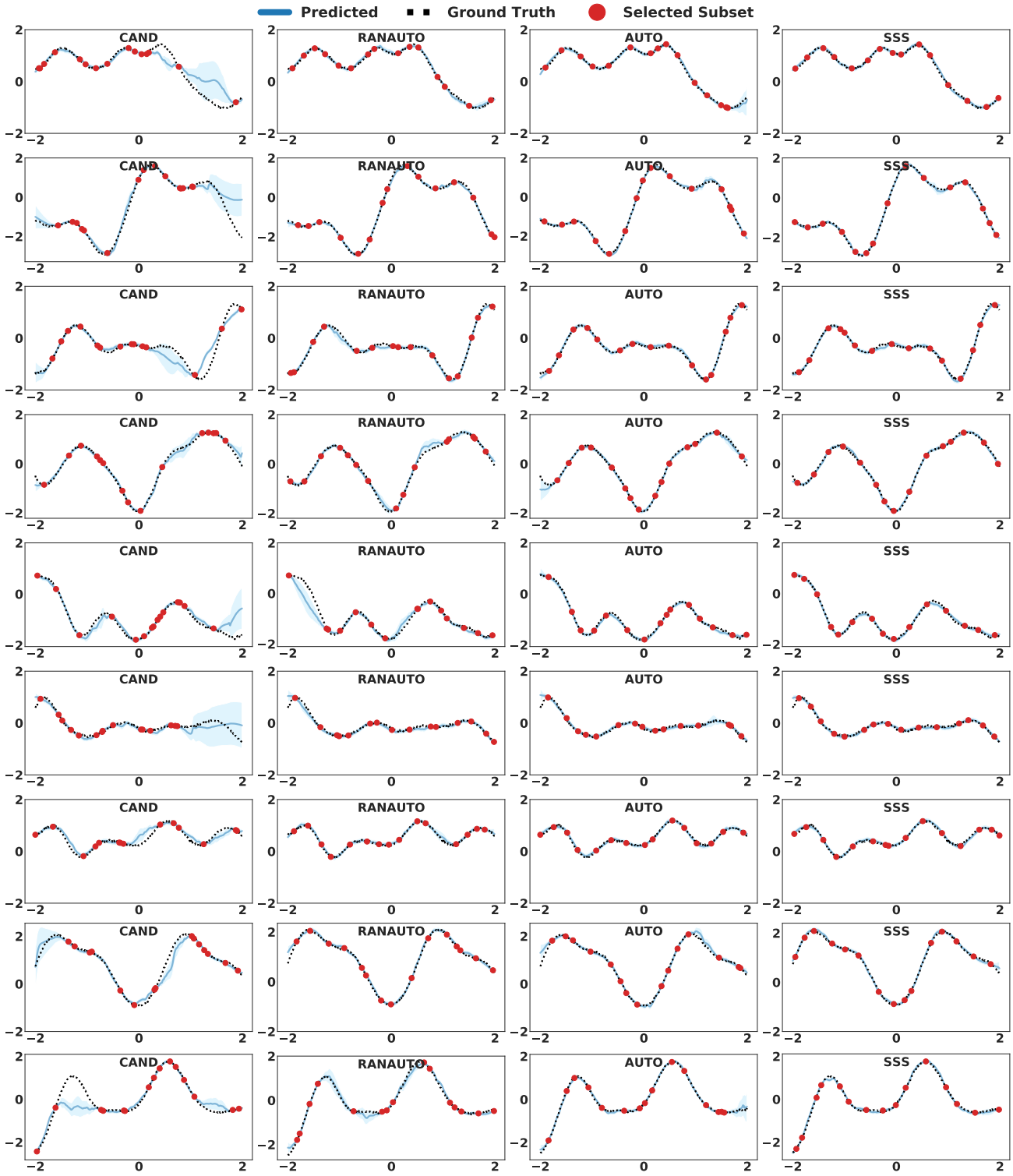

Figure 8. **Ablation:** Visualization of 1D function reconstruction. In the ablation studies, we compare the first stage (**CAND**) and the second stage (**AUTO**) with **SSS**. Additionally, we replace the Candidate Selection stage (Stage 1) in SSS with random selection (**RANAUTO**) and compare the performance of these models. As can be seen from the visualized reconstructed outputs, the combination of the Candidate Seletion stage with the Autoregressive stage results in the best subset selection for the reconstruction task.

E. Proofs

Definition E.1. Let $\mathfrak{S}_n \coloneqq \{g : [n] \to [n] \mid g \text{ is bijective}\}$ be a set of all permutation on n, where $[n] \coloneqq \{1, \dots, n\}$. We say $f : X^n \to Y$ is permutation invariant if and only if for any permutation $\pi \in \mathfrak{S}_n$, $f(\pi(\mathbf{x})) = f(\mathbf{x})$ for all $\mathbf{x} \in X^n$.

Definition E.2. Let $\mathfrak{S}_n \coloneqq \{g : [n] \to [n] \mid g \text{ is bijective}\}$ be a set of all permutation on n, where $[n] \coloneqq \{1, \dots, n\}$. We say $f : X^n \to Y^n$ is permutation equivariant if and only if for any permutation $\pi \in \mathfrak{S}_n$, $\pi(f(\mathbf{x})) = f(\pi(\mathbf{x}))$ for all $\mathbf{x} \in X^n$.

Proposition E.3. *Let $D = \{d_i \in \mathbb{R}^d \mid i = 1, \dots, n\}$ be a finite set and let g be an affine transformation with non-linearity. Define a function $k(d_1, \dots, d_n) \coloneqq \frac{1}{n}\sum_{i=1}^n g(d_i)$. Then the set encoding $D_e = k(d_1, \dots, d_n)$ is permutation invariant.*

Proof. Let a permutation $\pi \in \mathfrak{S}_n$ be given.

$$\begin{aligned} k(d_1, \dots, d_n) &= \frac{1}{n}\sum_{i=1}^n g(d_i) \\ &= \frac{1}{n}\sum_{i=1}^n g(d_{\pi(i)}) \\ &= k(d_{\pi(1)}, \dots, d_{\pi(n)}) \end{aligned} \tag{10}$$

□

Proposition E.4. *Let $f : X^n \to Y^n$ and $g : Y^n \to Z^n$ be permutation equivarant functions. Then the composition of two function $g \circ f$ is permutation equivariant function.*

Proof. Suppose that f and g are permutation equivariant. Let $\pi \in \mathfrak{S}_n$ be permutation. We want to show $\pi((g \circ f)(\mathbf{x})) = (g \circ f)(\pi(\mathbf{x}))$ for all $\mathbf{x} \in X^n$. Let $\mathbf{x} \in X^n$ be given.

$$\begin{aligned} (g \circ f)(\pi(\mathbf{x})) &= g(f(\pi(\mathbf{x}))) \\ &= g(\pi(f(\mathbf{x}))) \\ &= \pi(g(f(\mathbf{x}))) \\ &= \pi((g \circ f)(\mathbf{x})) \end{aligned} \tag{11}$$

The second and third equality holds since f and g are permutation equivariant. □

Proposition E.5. *Let $\zeta : X^n \to Y^n$ be a function, mapping $(d_1, \dots, d_n) \mapsto (\rho(\overline{d}_1), \dots, \rho(\overline{d}_n))$ for candidate selection. Then ζ is permutation equivariant.*

Proof. Let a permutation $\pi \in \mathfrak{S}_n$ be given and let $\mathbf{x} = (d_1, \dots, d_n)$ be given.

$$\begin{aligned} \zeta(\pi(\mathbf{x})) &= \left(\rho(\overline{d}_{\pi(1)}), \dots, \rho(\overline{d}_{\pi(n)})\right) \\ &= \left(\sigma(h(\overline{d}_{\pi(1)})), \dots, \sigma(h(\overline{d}_{\pi(n)}))\right) \\ &= \left(\sigma(h(d_{\pi(1)}; D_e)), \dots, \sigma(h(d_{\pi(n)}; D_e))\right) \\ &= \pi\left(\sigma(h(d_1; D_e)), \dots, \sigma(h(d_n; D_e))\right) \\ &= \pi\left(\zeta(\mathbf{x})\right) \end{aligned} \tag{12}$$

where ; denotes concatenation of two vectors and $D_e = \frac{1}{n}\sum_{i=1}^n g(d_i)$. $g(\cdot)$ and $h(\cdot)$ affine transformation followed by non-linear activation. Since element-wise operations $h(\cdot), \sigma(\cdot)$ are permutation equivariant and $(d_1, \dots, d_n) \mapsto (d_1; D_e, \dots, d_n; D_e)$ is permutation equivariant, composition of those functions is also permutation equivariant by Proposition E.4. Therefore, fourth equality holds, i.e., ζ is permutation equivariant. □

Proposition E.6. *The probability $p_\theta(Z|D)$ induced by candidate selection function is exchangeable.*

Proof. Let a permutation $\pi \in \mathfrak{S}_n$ be given.

$$p_\theta(Z|D) = \prod_{i=1}^n p_\theta(z_i|d_i, D) \tag{13}$$

$$= \prod_{i=1}^n \rho(d_i; D_e) \tag{14}$$

$$= \prod_{i=1}^n \rho(d_{\pi(i)}; D_e) \tag{15}$$

$$= \prod_{i=1}^n p_\theta(z_{\pi(i)}|d_{\pi(i)}, D) \tag{16}$$

$$= p_\theta(\pi(Z)|D) \tag{17}$$

where $D_e = \frac{1}{n}\sum_{i=1}^n g(d_i)$ and ; denotes concatenation of two vectors. Equality in 15 holds since D_e is permutation invariant and ρ is element-wise operation. □

Proposition E.7. *Let f be a stack of multi-head attention blocks from Set Transformer (Lee et al., 2019) and let φ be affine transformation. For all time step t in autoregressive selection, the functions f, φ, and $\sigma \circ \varphi \circ f$ are permutation equivariant, where σ is sigmoid function.*

Proof. Since each multi-head attention block in f is permutation equivariant, a stack of the blocks is also permutation equivariant by Proposition E.4. Since we apply φ independently to each element in a set, φ is permutation equivariant. Similarly, σ is permutation equivariant since it is an element-wise operation. As a result, $\sigma \circ \varphi \circ f$ is permutation equivariant again by Proposition E.4. □

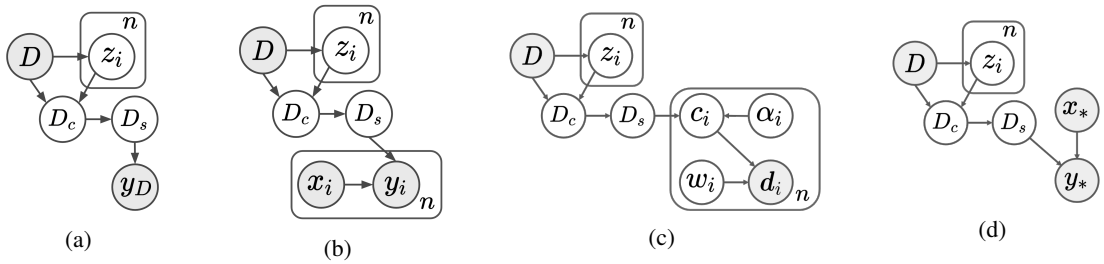

Figure 9. **Graphical Models: (a)** Feature selection for reconstruction. **(b)** Feature Selection for prediction task. **(c)** Instance selection for representative data points. **(d)** Instance selection for few-shot classification.

F. Graphical Model

In Figure 9, we illustrate the generative process using graphical models for each tasks — (a) feature selection for set reconstruction, (b) feature selection for prediction (c) Instance selection for representative data points and (d) instance selection for few-shot classification. The shaded circles denote observed variables and the others latent variables.

G. Instance Selection Samples

In this section, we show more qualitative examples for the 1D and CelebA experiments on how the models subsamples elements of the given set for the target task.

G.1. 1D Function - Reconstruction

Figure 10 shows the reconstruction samples of our model on the 1D function dataset, where SSS clearly outperforms Learning to Sample (LTS) and Random Subset (RS). Since RS selects the points randomly, it can leave out important part of the 1D curve leading to wrong reconstructions. Similarly, LTS also miss some parts of the curves, resulting in suboptimal reconstructions.

G.2. CelebA

Figure 11 shows samples of reconstructed images while varying the number of selected pixels. Additionally in Figure 12, we show the selected pixels of our model for both the classification and reconstruction task. For the attribute classification task, the model tends to select pixels mainly from the face, since the task is to classify characteristics of the person. For reconstruction, the selected pixels are more evenly distributed, since the background also contributes significantly to the reconstruction loss.

H. Dataset Distillation: Instance Selection

For the instance selection experiments, we construct a set by randomly sampling 200 face images from the full dataset. To evaluate the model, we create multiple such datasets and run the baselines (Random Sampling, K-Center Greedy and FPS) and SSS on the same datasets. We compute the FID metric (Heusel et al., 2017) on the instances and averaged on all the randomly constructed datasets. For FPS, we use the open-source implementation in `https://github.com/rusty1s/pytorch_cluster`. Further, we provide qualitative results on a single dataset in Figure 13 where our model picks 5 instances from the full set of 200 face images.

H.1. Dataset Distillation: Classification

In Figure 14 we provide visualizations for the instance selection problem as applied to the few-shot classification task. Here, we go from a 20-shot to a 1-shot classification problem where the prototype is selected from the support using SSS. The selected subset is then used in place of the support set and used to classify new query instances.

I. Model Specifications

In this section, we describe the main components of our Set based Stochastic Sumbsampling models — $g(d), \rho(\bar{d})$ and $\varphi \circ f(D_c^{(t)}, D_s^{(t-1)})$.

For all the experiments, we design g as feedforward neural network with ReLU to project each instance d to lower dimension and average them to obtain the set representation D_e, following DeepSets (Zaheer et al., 2017).

We parameterize $\rho(\cdot)$ with a 3 layered feedforward neural network h followed by sigmoid function as described in Equation 2 from Section 3.3. For f, we use the function g in Eq. 1 to extract feature map for each instances in $D_c^{(t)}, D_s^{(t-1)}$ and feed it to set transformer (Lee et al., 2019) for set classification as follows:

$$
\begin{aligned}
f(D_c^{(t)}, D_s^{(t-1)}) &= \text{MAB}(D_c^{(t)}, D_s^{(t-1)}) \\
\text{MAB}(D_c^{(t)}, D_s^{(t-1)}) &= \text{LayerNorm}(H + \text{rFF}(H)) \\
H &= \text{LayerNorm}(D_c^{(t)} + \text{Multihead}(D_c^{(t)}, D_s^{(t-1)}, D_s^{(t-1)}))
\end{aligned}
\tag{18}
$$

where rFF is a row-wise feedfoward layer which processes each instance independently and Multihead denotes Mul-

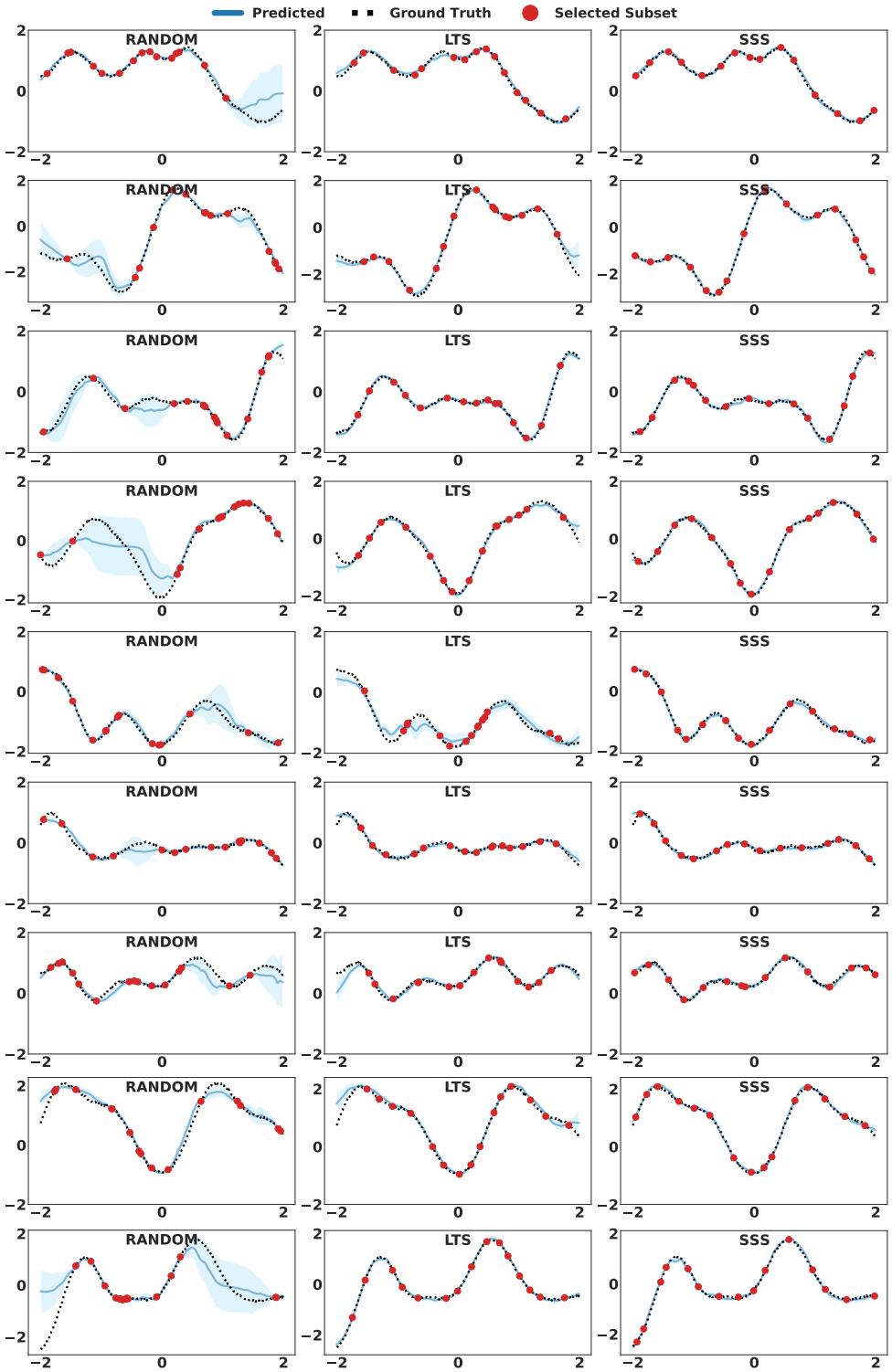

Figure 10. Visualization of 1D function reconstruction with three different subset selection models. Each method selects 15 out of 400 elements. As can be seen, SSS selects elements that result in better reconstructed functions.

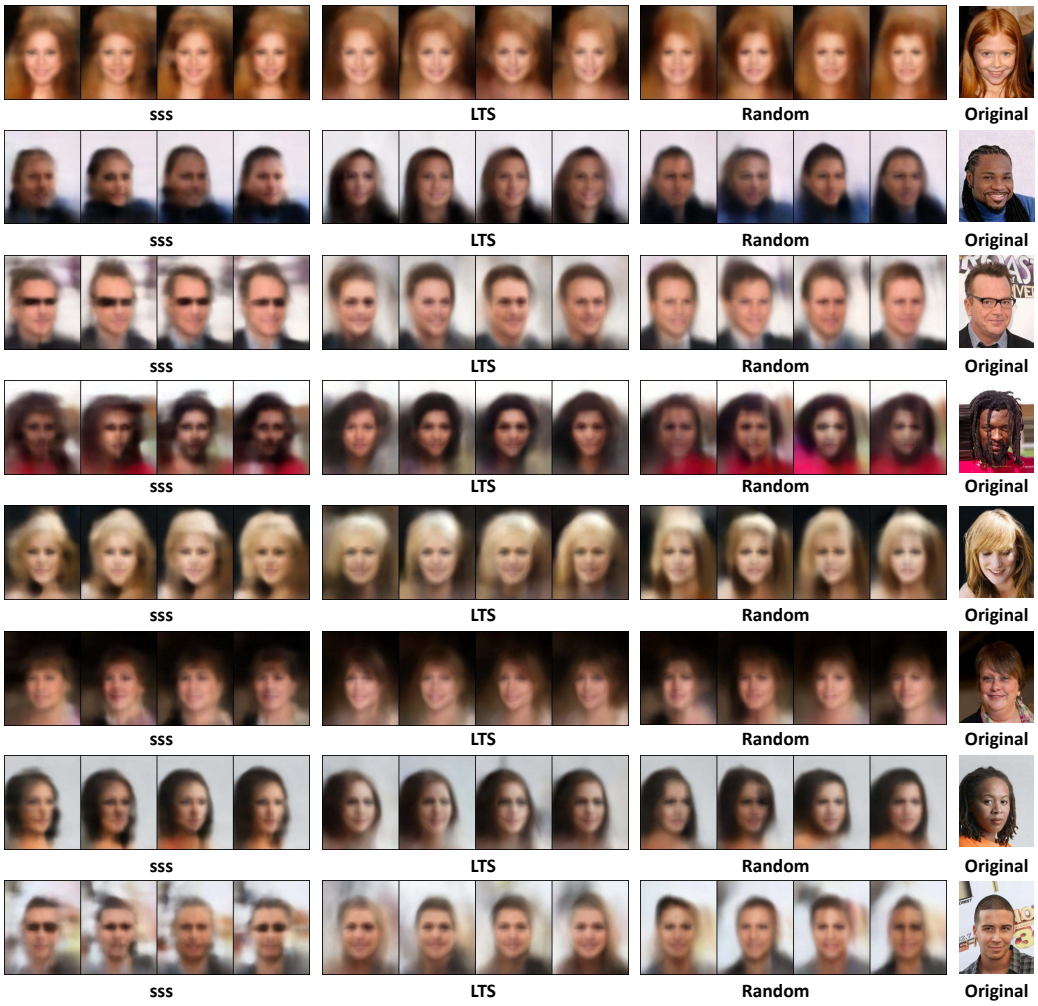

Figure 11. Visualization of reconstructed images for the CelebA dataset. Each model selects 40, 60, 80, and 100 pixels from a 218×178 image and reconstruct the full image using only the selected pixels.

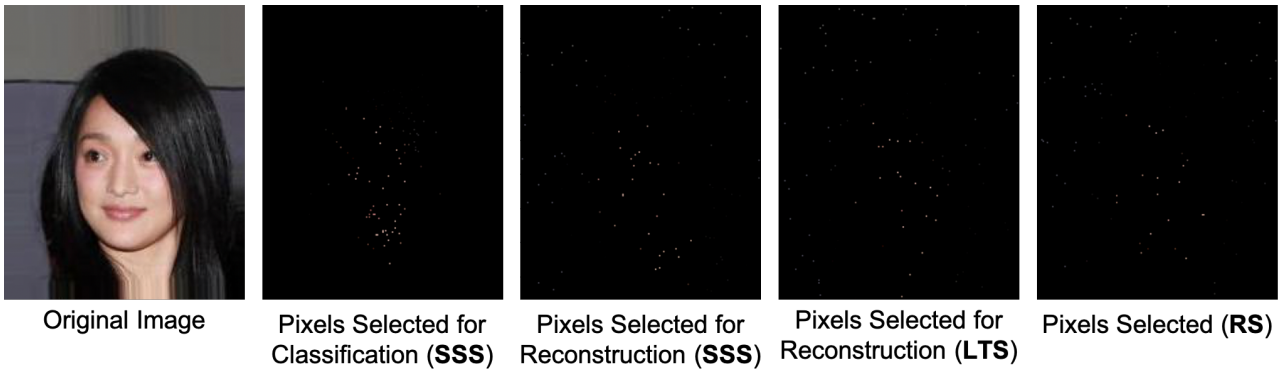

Figure 12. **Zoom-In for best view.** Selected pixels for different tasks on CelebA. As can be seen from the selected pixels, SSS adaptively selects different pixels for both reconstruction and classification. Pixels for reconstruction are more spread out to include the background since this contributes to the reconstruction loss. For classification, almost all the pixels are focused on the face since most of the attributes can be found there.

Figure 13. Visualization of a set with 200 images for instance selection. The two stage selection method in SSS is visualized as Candidate Set and Subset. A subset of size 5 is visualized.

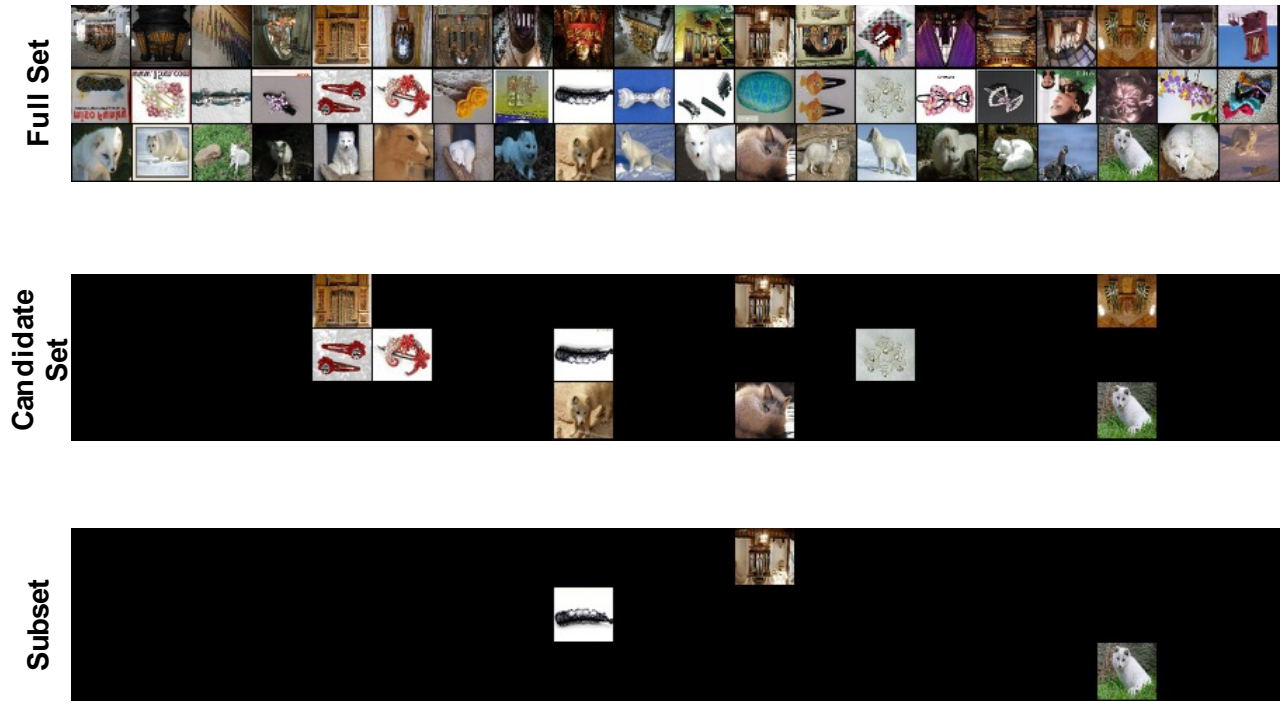

Figure 14. Sample visualization of prototype selection for the *mini*Imagenet dataset on the few-shot classification task. Each row represents a *set* that corresponds to the *support* from which a prototype is selected for the few-shot classification task.

tihead Attention (Vaswani et al., 2017) with each slot of Multihead$(\cdot,\cdot,\cdot)$ representing query, key, and value, respectively. We use linear layer for φ to output logits for each element in $D_c^{(t-1)}$.

I.1. Attention

We further elaborate the details of Attention and Multihead Attention for completeness. For a more thorough exposition, we refer the reader to Vaswani et al. (2017) and Lee et al. (2019).

An attention module computes the following interactions using the dot product:

$$\text{Att}(Q,K,V;\omega) = \omega(QK^\top)V \tag{19}$$

where $Q \in \mathbb{R}^{n_q \times d_q}$ are the n_q query vectors each with of dimension d_q. $K \in \mathbb{R}^{n_v \times d_q}$ and $V \in \mathbb{R}^{n_v \times d_q}$ are the keys and values respectively. Interactions are modelled through $QK^\top$ and ω is an activation function such as softmax or sigmoid.

Multihead attention projects Q, K, V to h different vectors each with d_q^M, d_q^M, d_v^M dimensions and computes h different attention modules according to the following:

$$\text{Multihead}(Q,K,V;\lambda,\omega) = \text{concat}(O_1,\ldots,O_h)W^O \tag{20}$$

where

$$O_j = \text{Att}(QW_j^Q, KW_j^K, VW_j^V;\omega_j) \tag{21}$$

The leanarble parameters for Multihead Attention are $\lambda = \{W_j^Q, W_j^K, W_j^V\}_{j=1}^h$, where $W_j^Q, W_j^K \in \mathbb{R}^{d_q \times d_q^M}$, $W_j^V \in \mathbb{R}^{d_v \times d_v^M}$ and $W^O \in \mathbb{R}^{hd_v^M \times d}$. In all our experiments, we use sigmoid as the activation function.

I.2. Architectures for MNIST Classification

MLP We use an architecture with 5 layers with outputs 784, 256, 128, 128 and 10 respectively. With the exception of the last layer, all layers are followed by a LeakyReLU activation function. The 3rd linear layer is also followed by a dropout layer with $p = 0.2$. This is the same architecture used in Huijben et al. (2019). However, we test on the full MNIST test set instead of using half for validation and the remaining for testing as done in Huijben et al. (2019).

ConvNet We use two convolutions with output channels 32 and 64 respectively. All convolutions have kernel size of 3 with stride 1 and are followed by ReLU activation. The final convolution layer is followed by a Maxpooling layer with kernel size 2. This is followed by two linear layers with output sizes of 128 and 10 and the first linear layer is followed by the ReLU activation function.

I.3. Architecture for CelebA Attribute Classification

ConvNet We use 4 convolutions each with outputs 64, 128, 256 and 512 respectively. Each convolution is followed by a batch normalization layer, ReLU activation and a Maxpooling layer with kernel size 2. This is followed by 3 linear layers with outputs 1024, 256 and 40 respectively. The first two linear layers are followed by the ReLU activation function and the last by the Sigmoid function.

I.4. Architecture for Regression Problems

For all the regression problems, we use the ANP model of Kim et al. (2019).

J. Experiments on CIFAR10 Dataset

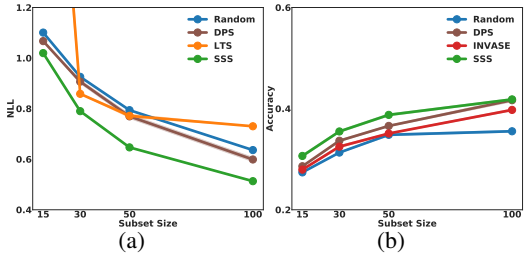

Figure 15. **(a)** CIFAR10 Reconstruction. **(b)** CIFAR10 Classification.

We provide further experimental results on the CIFAR10 dataset in which we subsample pixels for both image reconstruction and image classification. In order to compare with both Huijben et al. (2019) and Yoon et al. (2019), we convert all the images to grayscale, following Huijben et al. (2019) (see Section 4.3).

The experimental result of the CIFAR10 reconstruction task is presented in Figure 15a, where for the same subsampling rates, SSS outperforms the competing baselines (DPS, INVASE, LTS and Random Sampling) in terms of the negative log-likelihood. For this task, we use ANP model as we have done in CelebA and function reconstruction tasks in Section 4.3. We do not compare with INVASE since INVASE requires two copies of the reconstruction model and requires more GPU memory.

In Figure 15b, we present the results for the CIFAR10 classification task. Again we observe that for the same subsampling rate, SSS performs better than DPS, INVASE and Random Sampling and the performance of the same classification model trained on the full input image is 0.70 ± 0.02. Note that we cannot compare with LTS for the same reasons given in Section 4.1.

K. Experiments on OCT Dataset

We also provide additional results on the Optical Coherence Tomography dataset where we need to select the most informative features from tomographic images for prediction and interpretation of diseases in the retina. As shown in Fig. 16, SSS selects *bumpy* regions of the retina crucial for diagnosis of Diabetic Macular Edema. The quantitative result from Table 4 further confirms the effectiveness of our method.

Table 4. Acc. with **200** out of 1024 pixels on OCT.

Random	INVASE	DPS	**SSS**	Full
38.80	41.80	76.70	**85.90**	95.50

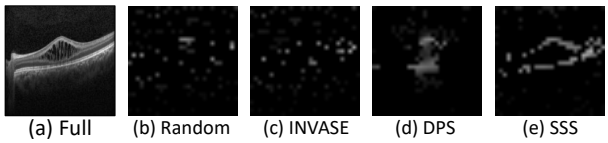

Figure 16. Visualization of selected 100 pixels.