Stochastic Subset Selection for Efficient Training and Inference of Neural Networks

Andreis Bruno\textsuperscript{1}, A. Tuan Nguyen\textsuperscript{3}, Seanie Lee\textsuperscript{1}, Juho Lee\textsuperscript{1}, Eunho Yang\textsuperscript{1,2}, Sung Ju Hwang\textsuperscript{1,2}

KAIST\textsuperscript{1}, South Korea
AITRICS\textsuperscript{2}, South Korea,
University of Oxford\textsuperscript{3}, UK

andries@kaist.ac.kr, tuan.nguyen@cs.ox.ac.uk,
{lsnfamily02, juholee, eunhoy, sjhwang82}@kaist.ac.kr

Abstract

Current machine learning algorithms are designed to work with huge volumes of high dimensional data such as images. However, these algorithms are being increasingly deployed to resource constrained systems such as mobile devices and embedded systems. Even in cases where large computing infrastructure is available, the size of each data instance, as well as datasets, can be a bottleneck in data transfer across communication channels. Also, there is a huge incentive both in energy and monetary terms in reducing both the computational and memory requirements of these algorithms. For nonparametric models that require to leverage the stored training data at inference time, the increased cost in memory and computation could be even more problematic. In this work, we aim to reduce the volume of data these algorithms must process through an end-to-end two-stage neural subset selection model. We first efficiently obtain a subset of candidate elements by sampling a mask from a conditionally independent Bernoulli distribution, and then autoregressively construct a subset consisting of the most task relevant elements via sampling the elements from a conditional Categorical distribution. We validate our method on set reconstruction and classification tasks with feature selection as well as the selection of representative samples from a given dataset, on which our method outperforms relevant baselines. We also show in our experiments that our method enhances scalability of nonparametric models such as Neural Processes.

1 Introduction

The recent success of deep learning algorithms partly owes to the availability of huge volumes of data such as ImageNet, CIFAR10/100, CelebA [10, 23, 28] to name a few, which enables training of very large deep neural networks. However, the high dimensionality of each data instance and the large size of datasets in general makes it difficult, especially for resource constrained/limited devices [5, 26, 2], to store and transfer the dataset, or even to perform on-device learning. This problem becomes more problematic for nonparametric models such as Neural Processes [15, 20] which require the training dataset to be stored for inference. It is therefore appealing to reduce the size of the dataset, both at the instance [11, 27, 27] and the dataset level, such that we select only a small number of samples from the dataset, called the subset, each of which contains only a few selected input features (e.g. pixels) or data points. The selected subset could then be used for tasks such as reconstruction of the original dataset or for prediction tasks such as classification of individual instances.

A simple and naïve way to obtain such a subset is random selection, but it is highly sub-optimal given that it treats all the elements in the set equally. However, the pixels from a given image and examples from a dataset will have varying degree of importance [19] to a target task, whether it is
reconstruction or prediction, and hence random selection generally results in low performance on the target task. There exist some works on subset selection [17, 3, 4] which propose to select a small subset consisting of the most important samples — core sets for Bayesian posterior inference. These methods are designed specifically for Bayesian inference and are not straightforwardly applicable to deep neural networks on arbitrary tasks and inference schemes. How can we then select elements from a given set to construct a subset with minimal performance degradation on an arbitrary target task? To tackle this problem, we propose to learn a subset selection model that learns to select the most important samples for a given task by training it jointly with the target task.

Specifically, we learn the sampling rate for individual samples via a two-stage selection procedure. In the first stage, as shown in Fig. 1-(a), we learn the Bernoulli 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 conditional Categorical distribution through an iterative procedure as shown in Fig. 1-(b). Once learned, the resulting subset selection model can perform stochastic subset selection of a given set with linear time complexity. We call the resulting model Stochastic Subset Selection (SSS) which is a general framework for subset selection and can be applied to both feature and instance selection. SSS can reduce the memory and computational cost required for processing large volumes of data while retaining performance on arbitrary downstream tasks.

Our model can benefit from a wide range of practical applications. For instance, when sending an image to an edge device with low computing power, instead of sending the entire image, we can instead transfer a subset of pixels with their coordinates which will reduce both the inference and communication cost. Similarly, edge devices may need to perform inference on a huge amount of data which are naturally represented as sets (e.g. video, point clouds) in real-time and SSS could be used to speed up the process. Moreover, SSS has the potential to be applied to on-device learning on personal data (e.g. photos), as it can select out samples to train models at a reduced cost. Finally, SSS improves scalability of nonparametric models which require storage of training examples, such as Neural Processes [14], to scale up to large-scale problems.

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 results show that SSS is able to select samples with minimal degradation on the target task performance, largely outperforming random selection and other selection methods. Our contribution in this work is threefold:

- We propose a novel two-stage stochastic subset selection method that learns to select a subset from a larger set with linear time complexity and with minimal degradation in performance on the downstream task.

- We propose a framework that trains the subset selection model over a distribution of sets so that it can generalize to unseen tasks such as in few-shot classification.

- We validate the efficacy and generality of our model on various datasets for feature selection from an instance and instance selection from a dataset and show that it significantly outperforms the relevant baselines.
2 Related Work

Set encoding 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 [36, 34, 45, 38]. However, these models have limited expressive power. Yet, approaches such as Stochastic Deep Network [9] and Set Transformer [24] consider the pairwise interactions among set elements and hence can capture more complex statistics of set distributions. They often show better performance but require at least \(O(n^2)\) time complexity.

Subset Sampling There exist some works to handle large sets. Dovrat et al. [11] propose learning to sample a subset by generating virtual points, then matching them back to a subset of the original set. Such generation and matching process is highly inefficient, however, our method learns to select from the original elements and does not suffer from such overhead. Wang et al. [43] propose to distill the knowledge of a large dataset to the small number of artificial data points. However, they are only for faster training and do not capture the statistics of the original set. Moreover, they are artificial and may differ from the original set, making it less applicable to other tasks. Several works [33, 35, 27, 12, 32] propose farthest point sampling, which selects \(k\) points from a set by ensuring that the selected ones are far from each other on a metric space.

Image Compression Due to the huge demand for data transfer over the internet, some works attempt to compress images with minimal distortion. These models [41, 37, 31, 25] typically consist of encoder and decoder, where the encoder transfers the image with a compact matrix and the decoder reconstructs the image back. These methods, while highly successive in the image compression problem, are less flexible than ours. Our model can be applicable to any type of sets while the aforementioned models mainly work for images represented in tensor form. Furthermore, our method can be applied both at the instance and dataset level.

Representation learning Our instance-sampling model is also related to the Variational Auto Encoder (VAE) [22]. However, while VAE learns a compact representation of a data point, our model learns a compact representation of a set. Balin et al. [1] propose a global feature selection model for reconstruction of the input data from selected features via unsupervised learning. Chen et al. [7] learns instance-wise feature selection with the goal of model interpretation by extracting subset of features most informative for a given sample.

Active Learning Active learning methods aim at selection of data points for labeling given a small labeled set. This domain is different from ours since active learning does not consider the label information but our method does utilize label information. Also, our motivation is quite different. We focus on efficiency in inference and training of nonparametric models by reducing the sizes of the inputs — pixels or instances, and this greatly differs from the goal of active learning. Methods such as [39, 8, 44] all tackle the data selection problem in the active learning setting.

3 Approach

3.1 Preliminaries

We consider a set \(D = \{d_1, \ldots, d_n\}\) as an input with \(D \sim p(\cdot)\) for some unknown data distribution, where individual \(d_i\)’s either represent a pair of input \(x_i\), and target \(y_i\) or a feature such as the pixel value of an image. We assume there exists a subset \(D_s = \{s_1, \ldots, s_k\} \subset D\) such that \(\ell(\cdot, D) \approx \ell(\cdot, D_s)\) for an arbitrarily defined loss function \(\ell(\cdot, D)\) that we are interested in optimizing over the full set \(D\) with \(k \ll n\). In what follows, we present a method that learns the conditional distribution \(p_k(D_s|D)\) of the subset \(D_s\) via a two stage subset selection method consisting of candidate selection and autoregressive subset selection as illustrated in Fig. 2. In general, we minimize the loss function \(\mathbb{E}_{p(D)}[\mathbb{E}_{p_k(D_s|D)}[\ell(\cdot, D_s)]]\) with respect to \(\xi\) the parameters of the subset selection model.

3.2 Stochastic Subset Selection

To select \(D_s\), we need 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 using the interaction scores. However, when the cardinality of \(D\) is large or its elements, \(d_i\)’s, are high dimensional, modeling pairwise interactions becomes computationally infeasible since we need to
We formulate the candidate selection problem as a random Bernoulli process where the parameters of the Bernoulli distribution are conditioned on the set representation of $D$ and the individual elements $d_i \in D$. Specifically, we first encode the dataset $D$ to a single representation $D_e$ as follows:

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

where $g$ is a neural network (Set Encoder in Fig. 2-(a)), which projects each element in $D$ independently to a lower dimension and $n = |D|$. This encoding scheme is similar to the one proposed in DeepSets [45] except that we do not perform message-passing between the set elements. Then we concatenate every $d_i$ with $D_e$, denoted as $\overline{d}_i$. It ensures each element of $D$ has a global view of all the other elements in the set at a coarse level. For every $d_i \in D$, we can sample a mask $z_i$ as follows:

$$p_\xi(z_i|d_i, D) = \text{Ber}(z_i|\rho(\overline{d}_i)), \quad \rho(\overline{d}_i) = \frac{\sigma(h(\overline{d}_i))}{\sum_{j=1}^{n} \sigma(h(\overline{d}_j))} \quad (2)$$

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 Sigmoid function, and Ber denotes the Bernoulli distribution. $z_i$ is a binary random mask where $z_i = 1$ implies that $d_i$ is an element in $D_c$. We concatenate all $z_i$’s to obtain a single vector $Z = \{z_i\}_{i=1}^{n}$. Note that we feed the output of $h$ into the Sigmoid function and normalize its output over all samples in $D$ to obtain a valid probability distribution and induce sparsity in the selected candidate set $D_c$. Since sampling from the Bernoulli distribution is not differentiable, during training, we use the continuous relaxations of the Bernoulli distribution [30, 18, 13] to sample the mask $z_i \sim \text{Ber}(\rho(\overline{d}_i))$ for $d_i$ as illustrated in the black boxes in Fig. 2-(a). Although pairwise interactions are not considered in this stage, the ablation studies show that learning $p_\xi(z_i|d_i, D)$ leads to selecting highly representative samples compared to random selection.

**Constraining the size of $D_c$**. For computational efficiency, we want to restrict the size of $D_c$ to save computational cost when constructing $D_c$. Hence we introduce a sparse Bernoulli prior $r(Z)$ and minimize the KL divergence along with the target downstream task loss $\ell(\cdot, D_s)$ w.r.t $\xi$ as follows:

$$E_{p(D)} \left[ E_{p_\xi(D_c|D)}[\ell(\cdot, D_s)] \right] + \beta \text{KL}[p_\xi(Z|D)||r(Z)]], \quad \text{where } p_\xi(Z|D) = \prod_{i=1}^{n} p_\xi(z_i|d_i, D) \quad (3)$$

where $\beta > 0$ is a hyperparameter used to control the sparsity level in $Z$.

### 3.4 Autoregressive Subset Selection

At this stage 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 requires

```
```
We enforce sparsity on the subset we have a function $\tilde{\pi}(t)$. That is, we normalize the vector of interaction scores $\tilde{\pi}(t)$ to approximate sampling from the multinomial distribution as shown in Fig. 2-(b). Since we want the continuous relaxation of Categorical distributions [30, 18] using the same probabilities in Eq. 5 hence it cannot be trained with backpropagation. Instead, we independently sample $\pi_k / l$ elements from the Categorical distribution since it involves computing Eq. 4 $k$ times. We remedy this by selecting $l$ elements from $D_c^{(t)}$, which reduces the number of iterations to $k / l$ for selecting $k$ elements. We may sample $l$ elements from multinomial distribution with the probability $\pi(t)$ without replacement. However, this sampling procedure is not differentiable, and hence it cannot be trained with backpropagation. Instead, we independently sample $l$ elements from the continuous relaxation of Categorical distributions [30, 18] using the same probabilities in Eq. 5 to approximate sampling from the multinomial distribution as shown in Fig. 2-(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 [1, 6]. We present this training algorithm in the Appendix.

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

### 3.5 Tasks

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

$$E_{p(D)} \left[ E_{p_\xi(Z|D)} \left[ -\log p_\theta(Y|X, D_s) \right] + \beta \text{KL}[p_\xi(Z|D)||r(Z)] \right]$$

(6)

We enforce sparsity on the subset $D_s$ by minimizing the KL-divergence between the mask probability $p_\xi(Z|D)$ and sparse prior $r(Z)$. 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) [21]. 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 4.1 and a corresponding graphical model depiction in Appendix D.

**Set Classification/Prediction** As shown in Fig. 3-(b), we train a neural network to predict a single target value $y_D$ for the subset $D_s$ of the given dataset $D$, where the set $D$ is a collection of the features
Dataset Distillation: Feature Selection

In this task, we are given a collection of datasets \( D = \{D^{(1)}, \ldots, D^{(m)}\} \) with \( D^{(i)} \cap D^{(j)} = \emptyset \) for \( i \neq j \) and \( D^{(i)} \overset{\text{iid}}{\sim} p(D) \), and the goal is to select the most representative subset \( D_s \) with \( k = |D_s| \ll |D| \) for each dataset \( D = \{d_1, \ldots, d_n\} \in D \), where \( d_i \) is a data point uniformly sampled from the entire dataset \( D \). Using CelebA dataset as an illustrative example, shown in Fig. 3, \( D \) consists of \( n \) randomly sampled faces from the entire dataset and the task is to construct a subset, \( D_s \), most representative of \( D \).

In order to learn to select the subset \( D_s \) from each \( D \in D \) 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 \( \sum_{d_i \in D} -\log p_D(d_i|D_s) \) for the loss function \( \ell(\cdot, D_s) \) and KL divergence in Eq. 3. However, we find that the generative model outputs a mean images for all \( d_i \). To capture variations of different images, we introduce three latent variables \( \alpha_i, \epsilon_i, \) and \( w_i \) which both depend on \( d_i \). We provide graphical model illustration of this task in the Appendix D. Since it is intractable to compute the log likelihood \( \log p_D(d_i|D_s) \) by marginalizing over all the latent variables, we derive the upper bound of the marginal likelihood using variational inference and plug the upper bound into the loss function \( \ell(\cdot, D_s) \) in Eq. 3 as follows:

\[
\mathbb{E}_{p(D)} \left[ \mathbb{E}_{p(\alpha, \epsilon, w|D_s, D)} \left[ -\log p_D(d_i|w, \epsilon) \right] + \text{KL}[q_\phi(w|d_i)||p_\psi(w_i)] \right]
\]

\[
+ \text{KL}[q_\phi(\alpha|d_i)||p_\psi(\alpha_i)] + \text{KL}[q_\psi(\epsilon|d_s, \alpha_i)||p_\psi(\epsilon_i)] + \beta \text{KL}[p_\psi(Z|D)||r(Z)] \right]
\]

where \( p_\psi(\cdot) \) are priors on their respective latent variables, \( r(Z) \) is sparse prior over the mask for candidate set selection in SSS, \( p_\theta(\cdot) \) is the decoder to reconstruct \( d_i \), and all variational posteriors \( q_\psi(\cdot) \) are parameterized with neural networks. All priors are chosen as 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 \( \theta, \phi, \) and \( \xi \) for all \( D \in D \) and leverage the optimized SSS to select a few representative instances of the dataset, which results in distilled dataset. We report all the experimental results in Section 4.3.

Dataset Distillation: Classification

Finally for the dataset distillation task, we consider the problem of selecting prototypes for few-shot classification as shown in Fig. 3-(d). We adopt Prototypical Networks [40] 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 $\ell(i, D_s)$, 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 class prediction boundaries in the classification task where we need to predict the label $y_s$ for an unseen instance $x_s$. Experimental results for this task are in Section 4.3 and an accompanying graphical model description is provided in Section D of the Appendix.

4 Experiments

Detailed model architectures and hyperparams are given in Section of the Appendix.

Baseline for feature selection 1) Random Selection: this model randomly selects $k$ elements from the set $D$ for the subset $D_s$. 2) Learning to Sample (LTS) [11]: this model learns to generate $k$ virtual elements, which can be matched to elements in $D$, that optimized for the downstream task.

Baseline for instance selection 1) Random Selection and two instance selection algorithms 2) k-Center-Greedy: this algorithm iteratively selects elements in $D$ closest to a set of centroids. 3) FPS: this algorithm iteratively selects the most distant elements to a randomly initialized $D_s$.

4.1 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 Gaussian process. We sample $(y_1, \ldots, y_n)$ \iid $\mathcal{N}(0, K_{XX} + \sigma_y^2 I_n)$ 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 $\sigma_y^2$ is variance for small likelihood noise. This will lead to a collection of the sets $(D^{(1)}, \ldots, D^{(N)})$. We train our model which consists of the subset selection model $p_k(D_s|D)$ and a task network $p_{y}(Y|X, D_s)$, which is Attentive Neural Process (ANP) [21], on this dataset and report the negative log-likelihood (NLL).

Fig. 4a shows the performance (NLL) of SSS compared to the baselines — Random Selection and LTS. As shown in Fig. 4a, SSS outperforms the baselines, verifying that the subset selection model $p_k(D_s|D)$ learns a meaningful distribution over subsets. We visualize a sample reconstructed function and the selected points by each models in Fig. 5. As shown in the rightmost figure (Fig. 5), SSS tends to pick out more elements (presented as 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 fails to do that, which leads to inaccurate reconstructions.

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
We evaluate the selected subset with the Fréchet Inception Distance (FID) [16], which measures the 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 our selection method achieves the lowest FID score for all selection sizes. Specifically our model outperforms all the baselines for selecting very few instances since our method 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 cannot capture the true representation of the whole dataset. We provide selected images by SSS from the full CelebA dataset in the Appendix.

### 4.2 Feature Selection for Classification

In this subsection, we validate our model on the image classification task illustrated in Fig. 3-(b). The goal is to select a subset of pixels of an image and predict the label of the chosen subset. We jointly train subset selection models and classifiers with CelebA dataset, where each classifier performs binary classification for 40 attributes of a face. Note that only the selected pixels of an image by the subset selection models are used for prediction and the other pixel values are set to zeros.

We report the mean AUC score on all 40 attributes for varying sizes of $D_s$. Fig. 4c shows that using only 500 pixels (~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 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, 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 to this experiment because during training, the generated virtual points with 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.

### 4.3 Dataset Distillation

**Instance Selection** The goal is to select only a few representative images from the given dataset as described in Section 3.5 and Fig. 3. We split the CelebA dataset into $m$ disjoint sets $D = \{D^{(1)}, \ldots, D^{(m)}\}$ and jointly train SSS and the generative model to minimize the objective in Eq. 8 with $D$. After training, we discard the generative model and leverage the subset selection model to choose a few representative images from the full CelebA dataset.

We evaluate the selected subset with the Fréchet Inception Distance (FID) [16], 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 our selection method achieves the lowest FID score for all selection sizes. Specifically our model outperforms all the baselines for selecting very few instances since our method 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 cannot capture the true representation of the whole dataset. We provide selected images by SSS from the full CelebA dataset in the Appendix.

| #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 | 0.1990 ± 0.1080 |
| SSS        | 2.5307 ± 1.3583 | 0.1086 ± 0.1982 | 0.5922 ± 0.3181 | 0.3333 ± 0.1169 | 0.2381 ± 0.1153 | 0.1679 ± 0.0807 |

| #Instances | 1    | 2    | 5    | 10   | 15   | 20   |
|------------|------|------|------|------|------|------|
| FPS        | 0.432 ± 0.0005 | 0.501 ± 0.005 | 0.598 ± 0.0000 | 0.598 ± 0.0000 | 0.598 ± 0.0000 | 0.598 ± 0.0000 |
| Random     | 0.444 ± 0.0000 | 0.525 ± 0.0005 | 0.618 ± 0.0003 | 0.618 ± 0.0003 | 0.618 ± 0.0003 | 0.618 ± 0.0003 |
| K-Greedy   | 0.290 ± 0.0006 | 0.413 ± 0.0005 | 0.570 ± 0.0002 | 0.570 ± 0.0002 | 0.570 ± 0.0002 | 0.570 ± 0.0002 |
| SSS        | 0.475 ± 0.0006 | 0.545 ± 0.0111 | 0.625 ± 0.0026 |

**Table 2: Accuracy on miniImageNet**

- **Table 1:** FID Score with varying the number of instances
- **Table 2:** Accuracy on miniImageNet

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 the Appendix.
4.4 Ablation

We perform extensive ablation studies on the two stages of SSS using the function reconstruction task presented in Section 4.1. First we explore the contribution of the candidate selection and autoregressive subset selection stages. We then replace the candidate selection stage with random selection in the SSS model and compare it with the full SSS model.

**Candidate Selection Only** In Fig. 6, the model with only the candidate selection stage shows poor performance. This is because it is not always desirable to select only highly activating samples in the set without considering any dependencies among the others, which may leads to constructing a candidate set with redundant elements.

**Random Selection with Autoregressive Subset Selection** To show how much the candidate selection stage contributes to the performance of SSS, we replace it with random selection. 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 and the autoregressive subset selection stage used alone. This shows that filtering elements with candidate selection helps the model to select more informative instances from the input set in the autoregressive subset selection stage than random selection.

**AutoRegressive Subset Selection Only** As shown in Fig. 6 we observe that the model with only autoreressive subset selection performs significantly better than the model with the candidate selection and the model with random selection followed by autoregressive subset selection. While this model performs well, it is not very practical due to the high computational cost when the size of the set becomes large. Additionally, processing lots of elements with the interaction model can lead to flattened attention scores over the elements, which makes it difficult to distinguish important elements from the others. The candidate selection stage plays an important role of remedying this problem.

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

**Stochasticity of SSS** Since our method is stochastic, the predictive distribution is $E_{p_L(D_s|D)}[p_Y(y|D_s)]$ for classification and we approximate it with Monte Carlo sampling. However in all experiments, we only report the result with one sampled subset, since it gives the best tradeoff in memory and computation. We compare it against another variant: SSS-MC which use 5 samples of subset for MC sampling and obtains a mean AUC of 91.32%. Note SSM-MC increases the computational cost (inference) and memory requirement up to 5 times. Our model SSS with a single subset sample achieves 90.93% (Fig. 4c) accuracy, which is comparable to SSS-MC. The result justifies that our model SSS achieves good performance for target tasks with memory and computational efficiency. Storage comparisons for the selected pixels with these variants of SSS are in the Appendix.

**Cost of Running SSS** We report the cost of running SSS in selection of subset of pixels, 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 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 (compared to 958.85k for the full model) which shows that SSS is computationally cheap to run.

5 Conclusion

In this paper, we have proposed a stochastic subset selection method to reduce the size of an arbitrary set while preserving performance on a target task. Our selection method utilizes a Bernoulli mask to perform candidate selection, and a stack of Categorical distributions to iteratively select a representative subset from the candidate set. As a result, the selection process does take the dependencies of the set’s members into account. Hence, it can select a compact set that avoids samples with redundant information. By using the compact subset in place of the original set for a target task, we can save memory, communication and computational cost. We hope that our work can facilitate the use of machine learning algorithm in resource-limited systems such as mobile and embedded devices.
References

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

[2] A. Bhatia, P. Varakantham, and A. Kumar. Resource constrained deep reinforcement learning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, pages 610–620, 2019.

[3] T. Campbell and T. Broderick. Bayesian coreset construction via greedy iterative geodesic ascent. *arXiv preprint arXiv:1802.01737*, 2018.

[4] T. Campbell and T. Broderick. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

[5] M. Chan, D. Scarafoni, R. Duarte, J. Thornton, and L. Skelly. Learning network architectures of deep cnns under resource constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1703–1710, 2018.

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

[7] J. Chen, L. Song, M. J. Wainwright, and M. I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv preprint arXiv:1802.07814*, 2018.

[8] C. Coleman, C. Yeh, S. Mussmann, B. Mirzasoleiman, P. Bailis, P. Liang, J. Leskovec, and M. Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[9] G. De Bie, G. Peyré, and M. Cuturi. Stochastic deep networks. *arXiv preprint arXiv:1811.07429*, 2018.

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

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

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

[13] H. Hensel. Neural processes in thermoregulation. *Physiological Reviews*, 1973.

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

[15] J. Huggins, T. Campbell, and T. Broderick. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088, 2016.

[16] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[17] A. Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. *arXiv preprint arXiv:1803.00942*, 2018.

[18] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

[19] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh. Attentive neural processes. *arXiv preprint arXiv:1901.05761*, 2019.

[20] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[23] A. Krizhevsky, V. Nair, and G. Hinton. Cifar-10 and cifar-100 datasets. URL: https://www.cs.toronto.edu/kriz/cifar.html, 2009.

[24] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh. Set transformer. arXiv preprint arXiv:1810.00825, 2018.

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

[26] M. Li, E. Yumer, and D. Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. arXiv preprint arXiv:1905.04753, 2019.

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

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

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

[30] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712, 2016.

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

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

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

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

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

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

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

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

[39] O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489, 2017.

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

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

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

[43] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros. Dataset distillation. arXiv preprint arXiv:1811.10959, 2018.

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

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

Organization This supplementary file is organized as follows. We provide the full pseudo-code for the Greedy Training Algorithm and Fixed-size Subset Selection Algorithm and illustrate the generative process of each task using graphical models. We then provide qualitative results for the 1D function reconstruction task, qualitative results on the ablation studies and sample visualizations for the CelebA reconstruction task, Instance Selection and on Instance Selection as applied to few-shot classification on the minImageNet dataset. Finally we provide more details on specifications of models used in the experiments.

Algorithm 1 Greedy Training Algorithm

| Input | $k$ (max subset size) | $m$ (mini-batch size) | $l$ (# elements selected at each iteration) | $p(D)$ (distribution of sets) | $\alpha$ (learning rate) | $\ell(\cdot, D_{x})$ (loss function) |
|-------|----------------------|----------------------|---------------------------------|-----------------|-----------------|---------------------------------|
| Output | trained model with converged $\theta$ and $\phi$ |

1: Randomly initialize $\theta, \phi, \xi$.
2: while not converged do
3: Sample $n$ sets $D^{(1)}, \ldots, D^{(m)}$ from $p(D)$
4: Sample $Z^{(j)} = \{z^{(j)}_{1}, \ldots, z^{(j)}_{n}\} \sim p_{\xi}(Z|D^{(j)})$ for $j = 1 \ldots m$
5: Construct $D_{c}^{(j)} = \{d^{(j)} \in D^{(j)} : z^{(j)} = 1\}$ for $j = 1 \ldots m$
6: Sample integer $i \sim \text{Unif}[0, k - l]$
7: $I^{(j)} \leftarrow$ random $i$-element subset of $D_{c}^{(j)}$ for $j = 1 \ldots m$
8: $Q^{(j)} \leftarrow$ select a $l$-element subset from $D_{c}^{(j)} \setminus I^{(j)}$ (with the auto-regressive model)
9: $\theta \leftarrow \theta - \alpha \nabla_{\theta} \frac{1}{m} \sum_{j=1}^{m} \ell(\cdot, I^{(j)} \cup Q^{(j)})$
10: $\phi \leftarrow \phi - \alpha \nabla_{\phi} \frac{1}{m} \sum_{j=1}^{m} \ell(\cdot, I^{(j)} \cup Q^{(j)})$
11: $\xi \leftarrow \xi - \alpha \nabla_{\xi} \frac{1}{m} \sum_{j=1}^{m} \ell(\cdot, I^{(j)} \cup Q^{(j)})$
12: end while

B Greedy 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. It selects only the top $l$ elements from the candidate set to train the auto-regressive model by minimizing the target loss on the selected samples and randomly selects the remaining $k - l$ elements from the candidate set. As a result, we do not have to run the auto-regressive model $k/l$ time during training, which significantly reduces the computational cost and shows reasonable performance.

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 12 from Algorithm 2 to construct the representative subset $D_{x}$.

D Graphical Model

In Figure 7, 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. Only the shaded circles denote observable variable and the others are latent variables.
Algorithm 2 Fixed Size Subset Selection. $k$ is the required subset size. $l$ is the number of elements to select at each iteration. $D$ is the full input set and $D_s$ is the selected subset after running SSS.

```
1: Input: $k$, $l$, $D = \{d_1, \ldots, d_n\}$
2: Output: $D_s = \{s_1, \ldots, s_k\}$
3: procedure SSS($k$, $l$, $D$)
4: $D_e \leftarrow \frac{1}{n} \sum_{i=1}^{n} g(d_i)$, $D_i \leftarrow \text{Concat}(d_i, D_e)$
5: $z_i \sim \text{Ber}(z_i | p(d_i))$ for $i = 1, \ldots, n$
6: $D_c \leftarrow \{d_i \in D | z_i = 1\}$ for $i = 1, \ldots, n$
7: $D_s \leftarrow \emptyset$
8: for $t = 1$ to $k/l$ do
9: $D_s \leftarrow D_s \cup \text{AUTOSELECT}(l, D_s, D_c)$
10: end for
11: end procedure
12: procedure AUTOSELECT($l$, $D_s$, $D_c$)
13: $D_{(t)} = \{w_1, \ldots, w_{m_t}\} \leftarrow D_c \setminus D_{(t)}$
14: $\tilde{\pi}_{(t)} \leftarrow \sigma(\varphi \circ f(w_i, D_{(t-1)}))$
15: $(\pi_{1}, \ldots, \pi_{m_t}) \leftarrow (\tilde{\pi}_{(t)}^{(1)}, \ldots, \tilde{\pi}_{(t)}^{(m_t)}) / \sum_{j=1}^{m_t} \tilde{\pi}_{(t)}^{(j)}$
16: $Q \leftarrow \text{Select} l$ elements $\in D_{(t)}$ with $\pi_{(t)}$
17: return $Q$
18: end procedure
```

Figure 7: 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.

**E Instance Selection Samples**

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

**E.1 1D Function - Reconstruction**

Figure 8 shows the reconstruction samples of our model on the 1D function dataset, which is objectively better than that of Learning to Sample (LTS) or Random Subset (RS). Since RS selects the set elements randomly, it can leave out important parts of the 1D curve leading to wrong reconstructions. LTS also selects insufficient amount of set elements in some parts of the curves, resulting in suboptimal reconstructions.

**E.2 CelebA**

Figure 4b shows samples of reconstructed images for varying selection number of pixels. Additionally in Figure 10, 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.

**F 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. The FID metric is then
Figure 8: 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 9: Visualization of reconstructed images for the CelebA dataset. Each model selects 40, 60, 80, and 100 pixels from a $218 \times 178$ image and reconstruct the full image using only the selected pixels.

Figure 10: 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.
computed 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 11 where we show how our model picks 5
instances from the full set of 200 images face images.

F.1 Dataset Distillation: Classification

In Figure 12 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.

G Model Specifications

In this section, we describe the main components of our Stochastic Subset Selection models —
g(d), ρ(̄d) and ϕ (Dc(t), Ds(t−1)). For all the experiments, we use g feedforward neural network with ReLU to project each instance d
to lower dimension and average it to encode set representation Dc, following DeepSets [45].

We parameterize ρ(·) with a 3 layered feedforward neural network h followed by sigmoid function as
follows as described in Equation (2) from Section 3.3. For ϕ, we use conv-net to extract feature
map for each instances in Dc(t), Ds(t−1) and feed it to set transformer [24] for set classification as
follows:

\[ f(D_{c(t)}, D_{s(t-1)}) = \text{MAB}(D_{c(t)}, D_{s(t-1)}) \]
\[ \text{MAB}(D_{c(t)}, D_{s(t-1)}) = \text{LayerNorm}(H + r\text{FF}(H)) \]
\[ H = \text{LayerNorm}(D_{c(t)} + \text{Multihead}(D_{c(t)}, D_{s(t-1)}, D_{s(t-1)})) \]

where rFF is a row-wise feedforward layer which processes each instance independently and Multihead
represents Multihead Attention [42] with each slot of Multihead denoting Multihead Attention [42] with each slot of Multihead
queries, key and value, respectively. For the other experiments, we use Deepsets for f. We use linear layer for \( \varphi \) to
output logits for each element in \( D_{s(t-1)} \).

G.1 Attention

We details on Attention and Multihead Attention here for completeness. For a more thorough
exposition, we refer the reader to Vaswani et al. [42] and Lee et al. [24].

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

\[ \text{Att}(Q, K, V; \omega) = \omega(QK^T)V \quad (10) \]

where \( Q \in \mathbb{R}^{n \times d_q} \) are the n query vectors each with of dimension \( d_q \), \( K \in \mathbb{R}^{n \times d_k} \) and \( V \in \mathbb{R}^{n \times d_v} \)
are the keys and values respectively. Interactions are modelled through \( QK^T \) and \( \omega \) is an activation
function such as Softmax or Sigmoid.

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

\[ \text{Multihead}(Q, K, V; \lambda, \omega) = \text{concat}(O_1, \ldots, O_h)W^O \quad (11) \]

where

\[ O_j = \text{Att}(QW^{Q,j}, KW^{K,j}, VW^{V,j}; \omega_j) \quad (12) \]

The Multihead Attention module has learnable parameters \( \lambda \) where \( \lambda = \{W^{Q,j}, W^{K,j}, W^{V,j}\}_{j=1}^h \) and
\( W^{Q,j}, W^{K,j} \in \mathbb{R}^{d_q \times d^M_{q,j}}, W^{V,j} \in \mathbb{R}^{d_k \times d^M_{v,j}} \) and \( W^O \in \mathbb{R}^{hd^M_{v,j} \times d} \). In all our experiments, we use Sigmoid
as the activation function.
Figure 11: Visualization of a set with 200 images for instance selection. The two stage selection method in SSS is visualized as Candidate Set and coreset. A subset of size 5 is visualized.
Figure 12: Sample visualization of prototype selection for the miniImagenet 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.

H Function Reconstruction: Ablation

In the ablation studies in Section 4.4 of the main paper, we investigated the contribution of the Candidate Selection stage (Stage 1) and the AutoRegressive subset selection stage (Stage 2). Additionally, we replaced Stage 1 with random selection in SSS and compare the performance of these three variants with SSS. In Figure 13, we provide qualitative results on the function reconstruction tasks for these 3 variants as well as SSS. Figure 13 is meant to supplement the results presented in Section 4.4 of the paper. As can be seen from the figure, using the Candidate Selection stage only to select elements generally focuses on the difficult sections of the curves ignoring the other parts which are relatively easier to reconstruct and this results in poorer overall reconstructions. The Autoregressive Subset Selection stage on the other hand selects elements that reconstruct the functions well. The downside to this model is that it requires a large amount of computation since we have to execute the interactions model for all elements in the input set and this renders this model impractical. On the other hand, the model that replaces the Candidate Selection stage in SSS with Random Selection performs better than using only the Candidate Selection stage only but results in function reconstruction that is sub-par compared to the Autoregressive Stage only and SSS. This is because using random selection in place of the first stage can ignore elements from certain parts of the function when we randomly sample the candidate set and hence these parts cannot be reconstructed since the second stage cannot select elements from those regions. This behaviour can be seen in some of the functions in the second column of Figure 13. In conclusion, SSS offers a good trade-off between computation and performance and results in better reconstructions since both the Candidate Selection stage and AutoRegressive Subset Selection stages work together to select the most relevant elements for the task.
Figure 13: **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 Selection stage with the Autoregressive stage results in the best subset selection for the reconstruction task.