Revisiting Score Function Estimators for k-Subset Sampling

Klas Wijk [1] **Ricardo Vinuesa** [1] **Hossein Azizpour** [1]

Abstract

Are score function estimators an underestimated approach to learning with k-subset sampling? Sampling k-subsets is a fundamental operation in many machine learning tasks that is not amenable to differentiable parametrization, impeding gradient-based optimization. Prior work has focused on relaxed sampling or pathwise gradient estimators. Inspired by the success of score function estimators in variational inference and reinforcement learning, we revisit them within the context of k-subset sampling. Specifically, we demonstrate how to efficiently compute the k-subset distribution's score function using a discrete Fourier transform, and reduce the estimator's variance with control variates. The resulting estimator provides *both* exact samples and unbiased gradient estimates while also applying to non-differentiable downstream models, unlike existing methods. Experiments in feature selection show results competitive with current methods, despite weaker assumptions.

1. Introduction

Subsets are essential in tasks such as feature selection (Balın et al. 2019; Huijben et al. 2019; Yamada et al. 2020), optimal sensor placement (Manohar et al. 2018), learning to explain (Chen et al. 2018), stochastic k-nearest neighbors (Grover et al. 2019), system identification (Brunton et al. 2016), and more. Understanding and effectively manipulating subsets is an important step in improving machine methods that model discrete phenomena.

A cornerstone of modern machine learning is efficient optimization, typically achieved through differentiable models optimized via stochastic gradient descent. However, not all operations are differentiable, necessitating approximate differentiation to leverage gradient-based optimization. This includes discrete sampling, and thus k-subset sampling which, unlike sampling from Gaussian distributions is not amenable to the reparametrization trick (Kingma and Welling 2014).

Differentiable optimization of Bernoulli and categorical distributions have been extensively studied (Bengio et al. 2013; Jang et al. 2017; Maddison et al. 2017; Dimitriev and Zhou 2021; De Smet et al. 2023; Liu et al. 2023). These distributions are less structured and do not share the combinatorially large support of subset distributions. Still, the methods employed in their optimization serve as a blueprint for more structured distributions. Existing approaches for differentiable subset sampling (Xie and Ermon 2019; Ahmed et al. 2023; Pervez et al. 2023) use either relaxed sampling methods or approximate pathwise gradient estimators. While these methods are effective, they produce relaxed samples (which cannot be used in all settings) and biased gradient estimates, respectively. This paper seeks to address these limitations by revisiting score function estimators (Glynn 1990; R. J. Williams 1992; Kleijnen and Rubinstein 1996), a technique well-established in variational inference and reinforcement learning, but overlooked for subset sampling.

We propose score function estimators for k-subset sampling (SFESS) as a complement to existing methods. This approach is fundamentally different to prior works on k-subset sampling, offering both exact samples and unbiased gradient estimates. Furthermore, it does not assume differentiable downstream models, broadening the possible applications of k-subset sampling to cases when the downstream model's gradient is unavailable or computationally expensive.

To realize our proposal, we develop an efficient method for computing the score function based on the discrete Fourier transform (DFT) for computing the Poisson binomial distributions' probability density function (Fernandez and S. Williams 2010). Furthermore, we use control variates to significantly reduce the high variance of the vanilla score function estimator.

2. Method

In this section, we give an overview of our method and provide details on how to compute the score function, reduce the variance with control variates, and anneal the size of the subset to the desired value during training.

[1]KTH Royal Institute of Technology, Stockholm, Sweden. Correspondence to: Klas Wijk <kwijk@kth.se>.

Published at the 2nd Differentiable Almost Everything Workshop at the 41st International Conference on Machine Learning, Vienna, Austria. July 2024. Copyright 2024 by the author(s).

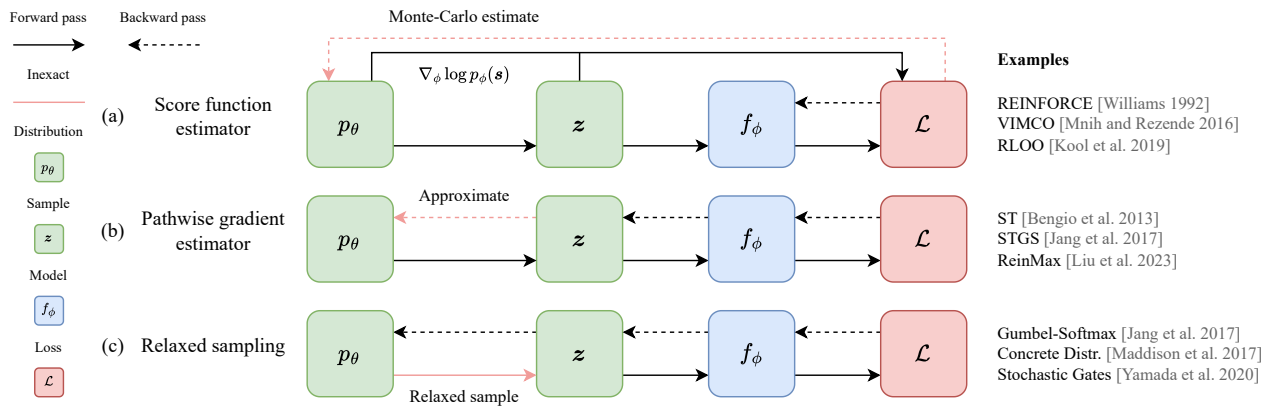

Figure 1: **Learning by sampling**. Three prominent approaches to learning by sampling: (a) score function estimator, (b) pathwise gradient estimator, and (c) relaxed sampling. We propose a score function estimator for k-subset distributions and compare it against existing methods based on approximate pathwise derivatives and relaxed sampling. Because it does not use the pathwise gradient, it is applicable in cases when f is non-differentiable.

We are interested in sampling subsets $\boldsymbol{z}$ of size k given a set of n variables. We consider the following conditional distribution:

$$p_{\boldsymbol{\theta},k}(\boldsymbol{z}) = p_{\boldsymbol{\theta}}\left(\boldsymbol{b} \,\middle|\, \textstyle\sum_{i=1}^n b_i = k\right) = \frac{\prod_{i=1}^n p_{\boldsymbol{\theta}}(b_i)}{p_{\boldsymbol{\theta}}\left(\sum_{i=1}^n b_i = k\right)} \mathbb{1}\left[\textstyle\sum_{i=1}^n b_i = k\right],$$

where $\boldsymbol{b} \in \{0,1\}^n$ is independently Bernoulli distributed with parameters $\boldsymbol{\theta} \in [0,1]^n$ and $\mathbb{1}[\cdot]$ denotes the indicator function. This equation induces a particular distribution over the $\binom{n}{k}$ possible subsets using only n parameters. Previous work has explored approximate derivatives of this distribution's samples (Xie and Ermon 2019; Ahmed et al. 2023). In this work, we instead consider score function estimators that are *exact* in expectation. Hence, we want to compute the score function defined on the region where $\sum_{i=1}^n b_i = k$,

$$\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta},k}(\boldsymbol{z}) = \sum_{i=1}^n \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(b_i) - \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}\left(\textstyle\sum_{i=1}^n b_i = k\right). \tag{1}$$

Computing the first term is easy, since each $p_{\boldsymbol{\theta}}(b_i)$ is Bernoulli distributed. The second term appears more challenging. Luckily, it follows a Poisson binomial distribution, a generalized binomial distribution where the samples are not necessarily identically distributed. Efficient methods exist for computing the Poisson binomial's density function, including approximate and recursive methods (Le Cam 1960; Wadycki et al. 1973; Ahmed et al. 2023). We follow Fernandez and S. Williams (2010) and compute it using a DFT (Cooley and Tukey 1965) – leveraging its $\mathcal{O}(n \log n)$ time-complexity and efficient implementation on modern hardware[1]. The gradient of the log probability is computed using automatic differentiation.

Now, being able to compute the score function in Equation (1), we can write the following score function estimator:

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{p_{\boldsymbol{\theta},k}(\boldsymbol{z})}[f_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{x})] \\
&= \mathbb{E}_{p(\boldsymbol{x})} \mathbb{E}_{p_{\boldsymbol{\theta},k}(\boldsymbol{z})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta},k}(\boldsymbol{z}) f_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{x})] \\
&\approx \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta},k}(\boldsymbol{z}^{(j)}) f_{\boldsymbol{\phi}}(\boldsymbol{z}^{(j)}, \boldsymbol{x}^{(i)}),
\end{aligned}$$

where N samples $\boldsymbol{x}^{(i)} \sim p(\boldsymbol{x})$ make up the training dataset and M samples $\boldsymbol{z}^{(j)} \sim p_{\boldsymbol{\theta},k}(\boldsymbol{z})$ Monte-Carlo estimate the inner expectation. We derive the standard estimator in Appendix D.

Efficiently computing the score function The second term of Equation (1) follows a Poisson binomial distribution. At first glance, the exact computation of this score function is prohibitive due to its combinatorial nature, naïvely expressed as:

$$p_{\boldsymbol{\theta}}\left(\textstyle\sum_{i=1}^n b_i = k\right) = \sum_{\boldsymbol{b} \in \{0,1\}^n} \mathbb{1}\left[\textstyle\sum_i b_i = k\right] p_{\boldsymbol{\theta}}(\boldsymbol{b}).$$

Fernandez and S. Williams (2010) derive this closed-form expression using the discrete Fourier transform:

[1]We use the Nvidia cuFFT implementation in PyTorch. See Appendix E for pseudocode.

$$p_{\boldsymbol{\theta}}\left(\sum_{i=1}^{n} b_i = k\right) \tag{2}$$
$$= \frac{1}{n+1}\,\mathrm{DFT}\left(\prod_{i=1}^{n} p_{\boldsymbol{\theta}}(b_i)e^{C} + (1 - p_{\boldsymbol{\theta}}(b_i))\right),$$

where $C = 2\sqrt{-1}\pi/(n+1)$. Note that this expression is solely a function of $\boldsymbol{\theta}$ and k which means we can cache any repeated calls when computing Equation (1) with different subsets $\boldsymbol{z}$ with the same size k. This is a common occurrence in e.g. instance-wise feature selection (Chen et al. 2018), where a new $\boldsymbol{z}$ is evaluated for each example $\boldsymbol{x}$.

Reducing variance with control variates The vanilla score function estimator generally suffers from high variance. While many variance reduction techniques have been proposed (Mnih and Gregor 2014; Gu et al. 2016; De Smet et al. 2023), we choose to employ control variates using multiple samples (Mnih and Rezende 2016; Kool et al. 2019) in this work due to its simplicity, unbiasedness, and lack of additional assumptions. The estimator with reduced variance is shown below:

$$\nabla_{\boldsymbol{\theta}}\,\mathbb{E}_{p(\boldsymbol{x})}\,\mathbb{E}_{p_{\boldsymbol{\theta},k}(\boldsymbol{z})}[f_{\boldsymbol{\phi}}(\boldsymbol{z},\boldsymbol{x})]$$
$$\approx \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta},k}(\boldsymbol{z}^{(j)})$$
$$\cdot\left(f_{\boldsymbol{\phi}}(\boldsymbol{z}^{(j)},\boldsymbol{x}^{(i)}) - \frac{1}{M-1}\sum_{k\neq j} f_{\boldsymbol{\phi}}(\boldsymbol{z}^{(k)},\boldsymbol{x}^{(i)})\right).$$

3. Related Work

In this section, we provide an overview of current methods for k-subset sampling.

Relaxed Subset Sampling (Xie and Ermon 2019) extends the Gumbel-Softmax distribution to distributions over subsets. Despite its elegance, relaxed subset sampling inherits the biased gradient estimation of the Gumbel-Softmax estimator. Furthermore, the top-k sampling procedure sequentially applies the softmax function k times, which limits scalability with respect to k and potentially degrades performance (Pervez et al. 2023).

SIMPLE (Ahmed et al. 2023) approximates the pathwise gradient of the sample using its exact marginals, achieving both lower bias and variance than relaxed subset sampling.

Neural Conditional Poisson Subset Sampling (NCPSS) (Pervez et al. 2023) relaxes k-subset sampling in a manner different from relaxed subset sampling (Xie and Ermon 2019), allowing subset sizes slightly smaller and larger subsets than k. Then, pathwise gradient estimates are used for differentiable optimization. The authors show that NCPSS is more scalable than relaxed subset sampling and that the subset size k can be optimized alongside the distribution's parameters.

Implicit Maximum Likelihood Estimation (I-MLE) (Niepert et al. 2021) uses a perturb-and-MAP approach that applies to general optimization problems, with subset sampling as a special case.

Other methods In some settings, a subset distribution can be modeled as either the concatenation of n Bernoulli variables or the sum of k categorical variables. This way, a host of gradient estimates for Bernoulli and categorical variables can be used (Yamada et al. 2020; Dimitriev and Zhou 2021; De Smet et al. 2023). However, neither option directly models k-subset sampling. Bernoulli variables require some constraint (e.g. a loss term) limiting the subset size (Ahmed et al. 2023), and categoricals run the risk of duplicate inclusions (Nilsson et al. 2024).

4. Experiments

We consider both feature selection for reconstruction and classification (Balın et al. 2019; Huijben et al. 2019; Yamada et al. 2020). Results on the test set are listed in Table 1 and the runtimes in Table 2. Additional plots and details about the experiments are presented in Appendices A to C.

Datasets We evaluate on three datasets: MNIST (LeCun et al. 1998), Fashion-MNIST (Xiao et al. 2017), and KMNIST (Clanuwat et al. 2018) split into training, validation, and test sets of sizes 40 000, 10 000, and 10 000.

Baselines We focus our comparison on two baselines from prior work: relaxed sampling using Gumbel-Softmax top-k sampling (GS) and pathwise gradient estimation using the straight-through version of it (STGS) (Xie and Ermon 2019). We compare these baselines against two score function estimator methods: the score function estimator for k-subset sampling (SFESS) and the version of the algorithm with control variates (SFESS-V). We use 5 samples for all methods with SFESS-V using them to to construct control variates, and the other methods reducing variance through averaging.

Discussion The experimental results show that SFESS-V is a competitive option for k-subset sampling for feature selection. Although GS and STGS converge faster in the early stages of training, SFESS-V gives the best final results, perhaps thanks to its unbiasedness. The weak results of SFESS without control variates demonstrate the detrimental effects of a high-variance gradient estimate and the need for variance reduction. The selections shown in Figure 3 in Appendix B further demonstrate this qualitatively.

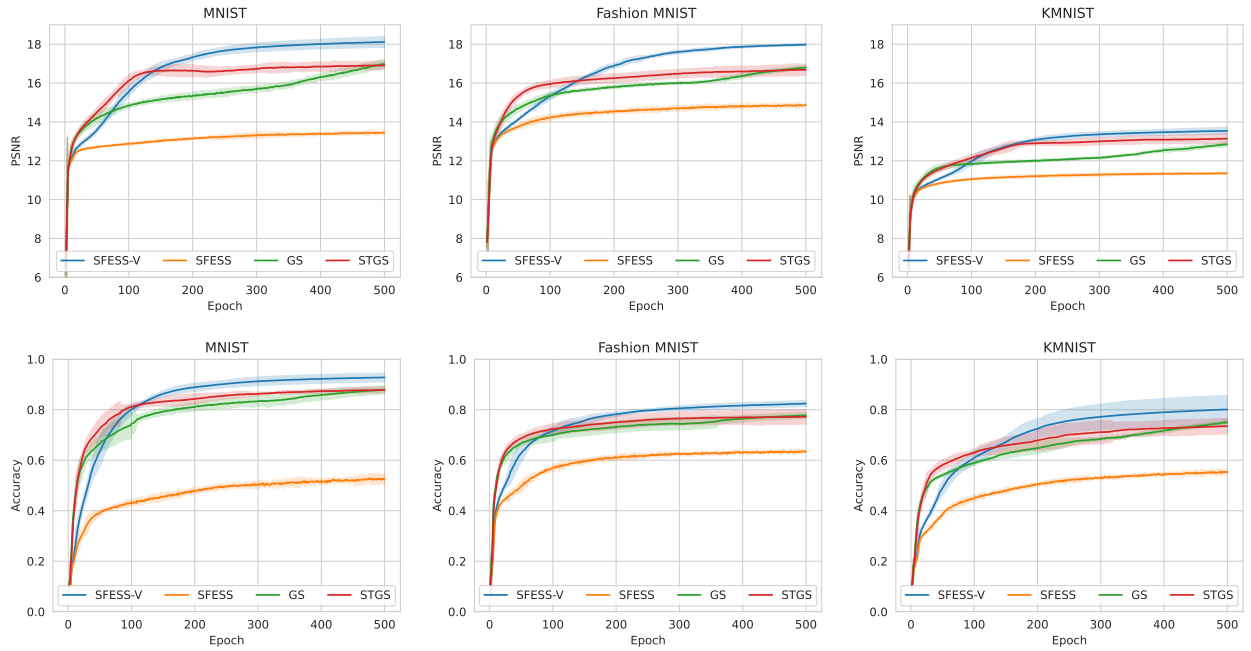

Figure 2: **Convergence plots**. Convergence of training metrics with $k = 30$ selections. The top row shows reconstruction PSNR and the bottom row classification accuracy. The confidence intervals show two standard deviations computed for 5 repeated runs with different random seeds. See Appendix C for the corresponding plot with validation data.

Table 1: **Feature selection results**. Results for feature selection for reconstruction and classification across three datasets with $k = 30$ selections. The confidence intervals show two standard deviations computed for 5 repeated runs with different random seeds. The best mean is shown in **bold** and the second best mean is underlined.

Metric	Dataset	GS	STGS	SFESS	SFESS-V
PSNR ↑	MNIST	17.402 ± 0.194	17.186 ± 0.319	13.686 ± 0.412	**17.775 ± 0.209**
	Fashion-MNIST	16.922 ± 0.289	16.642 ± 0.358	15.308 ± 0.303	**17.805 ± 0.075**
	KMNIST	12.641 ± 0.083	12.561 ± 0.140	11.300 ± 0.281	**12.696 ± 0.120**
SSIM ↑	MNIST	0.771 ± 0.011	0.759 ± 0.319	0.416 ± 0.412	**0.796 ± 0.209**
	Fashion-MNIST	0.586 ± 0.017	0.578 ± 0.031	0.456 ± 0.016	**0.642 ± 0.011**
	KMNIST	0.428 ± 0.021	**0.464 ± 0.048**	0.230 ± 0.026	0.460 ± 0.022
Accuracy ↑	MNIST	0.898 ± 0.014	0.898 ± 0.019	0.627 ± 0.077	**0.921 ± 0.015**
	Fashion-MNIST	0.777 ± 0.012	0.774 ± 0.027	0.643 ± 0.112	**0.809 ± 0.009**
	KMNIST	0.604 ± 0.029	0.591 ± 0.032	0.425 ± 0.023	**0.634 ± 0.059**

Table 2: **Runtime**. Average runtime across all experiments in Table 1, all of which have $k = 30$ selections. The measured time includes all surrounding execution, logging, etc. The shortest mean time is shown in **bold** and the second shortest time is underlined.

Metric	GS	STGS	SFESS	SFESS-V
Total time	28.7 min	29.1 min	**17.0 min**	17.4 min
Time per epoch	3.45 s	3.49 s	**2.04 s**	2.09 s

5. Conclusion

In this work, we derived a computationally efficient, unbiased score function estimator for k-subset distributions. We then showed how it can be computed using a discrete Fourier transform and how its variance can be reduced using control variates. We emphasize that our proposed estimator is complementary to prior works. An interesting direction for future work is combining score function and pathwise gradient estimators for k-subset sampling, as was done for categorical distributions (Tucker et al. 2017).

Acknowledgements

This work was supported by the Swedish e-Science Research Centre (SeRC). The computations were enabled by the Berzelius resource provided by the Knut and Alice Wallenberg Foundation at the National Supercomputer Centre. Klas Wijk thanks Rickard Maus, Sai bharath chandra Gutha, and Ricky Molén for helpful discussions. We also thank the anonymous reviewers for their valuable feedback and suggestions.

References

Ahmed, Kareem et al. (2023). "SIMPLE: A Gradient Estimator for k-Subset Sampling". *International Conference on Learning Representations*.

Balın, Muhammed Fatih, Abubakar Abid, and James Zou (2019). "Concrete Autoencoders: Differentiable Feature Selection and Reconstruction". *International Conference on Machine Learning*.

Bengio, Yoshua, Nicholas Léonard, and Aaron Courville (2013). "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation". *arXiv preprint arXiv:1308.3432*.

Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz (2016). "Discovering Governing Equations from Data by Sparse Identification of Nonlinear Dynamical Systems". *Proceedings of the National Academy of Sciences* 113, pp. 3932–3937.

Chen, Jianbo et al. (2018). "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation". *International Conference on Machine Learning*.

Clanuwat, Tarin et al. (2018). "Deep Learning for Classical Japanese Literature". *NeurIPS Workshop on Machine Learning for Creativity and Design*.

Cooley, James W. and John W. Tukey (1965). "An Algorithm for the Machine Calculation of Complex Fourier Series". *Mathematics of Computation* 19, pp. 297–301.

De Smet, Lennert, Emanuele Sansone, and Pedro Zuidberg Dos Martires (2023). "Differentiable Sampling of Categorical Distributions Using the CatLog-Derivative Trick". *Advances in Neural Information Processing Systems*.

Dimitriev, Aleksandar and Mingyuan Zhou (2021). "ARMS: Antithetic-REINFORCE-Multi-Sample Gradient for Binary Variables". *International Conference on Machine Learning*.

Fernandez, Manuel and Stuart Williams (2010). "Closed-Form Expression for the Poisson-Binomial Probability Density Function". *IEEE Transactions on Aerospace and Electronic Systems* 46, pp. 803–817.

Glynn, Peter W. (1990). "Likelihood Ratio Gradient Estimation for Stochastic Systems". *Communications of the ACM* 33, pp. 75–84.

Grover, Aditya et al. (2019). "Stochastic Optimization of Sorting Networks via Continuous Relaxations". *International Conference on Learning Representations*.

Gu, Shixiang et al. (2016). "MuProp: Unbiased Backpropagation for Stochastic Neural Networks". *International Conference on Learning Representations*.

Huijben, Iris A. M., Bastiaan S. Veeling, and Ruud J. G. van Sloun (2019). "Deep Probabilistic Subsampling for Task-Adaptive Compressed Sensing". *International Conference on Learning Representations*.

Jang, Eric, Shixiang Gu, and Ben Poole (2017). "Categorical Reparameterization with Gumbel-Softmax". *International Conference on Learning Representations*.

Kingma, Diederik P. and Max Welling (2014). "Auto-Encoding Variational Bayes". *International Conference on Learning Representations*.

Kleijnen, Jack P. C. and Reuven Y. Rubinstein (1996). "Optimization and Sensitivity Analysis of Computer Simulation Models by the Score Function Method". *European Journal of Operational Research* 88, pp. 413–427.

Kool, Wouter, Herke van Hoof, and Max Welling (2019). "Buy 4 REINFORCE Samples, Get a Baseline for Free!" *ICLR Workshop on Deep Reinforcement Learning Meets Structured Prediction*.

Le Cam, Lucien (1960). "An approximation theorem for the Poisson binomial distribution". *Pacific Journal of Mathematics* 10, pp. 1181–1197.

LeCun, Yann, Corinna Cortes, and Chris Burges (1998). *The MNIST database of handwritten digits*.

Liu, Liyuan et al. (2023). "Bridging Discrete and Backpropagation: Straight-Through and Beyond". *Advances in Neural Information Processing Systems*.

Maddison, Chris J., Andriy Mnih, and Yee Whye Teh (2017). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". *International Conference on Learning Representations*.

Manohar, Krithika et al. (2018). "Data-Driven Sparse Sensor Placement for Reconstruction: Demonstrating the Benefits of Exploiting Known Patterns". *IEEE Control Systems Magazine* 38, pp. 63–86.

Mnih, Andriy and Karol Gregor (2014). "Neural Variational Inference and Learning in Belief Networks". *International Conference on Machine Learning*.

Mnih, Andriy and Danilo J. Rezende (2016). "Variational inference for Monte Carlo objectives". *International Conference on Machine Learning*.

Mohamed, Shakir et al. (2020). "Monte Carlo Gradient Estimation in Machine Learning". *Journal of Machine Learning Research* 21, pp. 1–62.

Niepert, Mathias, Pasquale Minervini, and Luca Franceschi (2021). "Implicit MLE: Backpropagating Through Discrete Exponential Family Distributions". *Advances in Neural Information Processing Systems*.

Nilsson, Alfred et al. (2024). "Indirectly Parameterized Concrete Autoencoders". *International Conference on Machine Learning*.

Pervez, Adeel, Phillip Lippe, and Efstratios Gavves (2023). "Scalable Subset Sampling with Neural Conditional Poisson Networks". *International Conference on Learning Representations*.

Tucker, George et al. (2017). "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". *Neural Information Processing Systems*.

Wadycki, Walter J. et al. (1973). "Letters to the Editor". *The American Statistician* 27, pp. 123–127.

Williams, Ronald J. (1992). "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". *Machine Learning* 8, pp. 229–256.

Xiao, Han, Kashif Rasul, and Roland Vollgraf (2017). "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". *arXiv preprint arXiv:1708.07747*.

Xie, Sang Michael and Stefano Ermon (2019). "Reparameterizable Subset Sampling via Continuous Relaxations". *International Joint Conference on Artificial Intelligence*.

Yamada, Yutaro et al. (2020). "Feature Selection using Stochastic Gates". *International Conference on Machine Learning*.

A. Details of Experiments

This appendix gives additional information on network architecture, initialization, hyperparameters and the computing environment used in the experiments.

A.1. Network Architecture

The network architectures used in the main feature selection experiments for reconstruction and classification are shown in Tables 3 and 4 respectively. We used a convolutional neural network for reconstruction and a fully connected network with dropout for classification.

Table 3: **Reconstruction network**. All convolutions have stride 1.

Type	#Features in	#Features out	Activation
Reshape	$28 \times 28 \times 1$	748	–
Linear	748	748	ReLU
Linear	748	748	–
Reshape	748	$28 \times 28 \times 1$	–
Conv (3×3)	$28 \times 28 \times 1$	$28 \times 28 \times 16$	ReLU
Conv (3×3)	$28 \times 28 \times 16$	$28 \times 28 \times 16$	ReLU
Conv (3×3)	$28 \times 28 \times 16$	$28 \times 28 \times 1$	Sigmoid

Table 4: **Classification network**

Type	#Features in	#Features out	Activation
Reshape	$28 \times 28 \times 1$	748	–
Linear	748	256	ReLU
Dropout (20%)	–	–	–
Linear	256	256	ReLU
Dropout (20%)	–	–	–
Linear	256	256	ReLU
Dropout (20%)	–	–	–
Linear	256	10	Softmax

A.2. Initialization

The subset distribution parameters were uniformly initialized to k/n. For the network parameters, we used the default initialization in PyTorch.

A.3. Optimization and Hyperparameters

We use the Adam optimizer with the momentum parameters $\alpha_{\boldsymbol{\theta}} = (0.99, 0.999)$ for the selector and $\alpha_{\boldsymbol{\phi}} = (0.9, 0.999)$ with weight decay 10^{-4} for the downstream network. We set the batch size to 1024. Learning rates were set to 10^{-2} for the selector and 10^{-4} for the downstream network and all models are trained for 500 epochs. We use cross-entropy loss for classification and binary cross-entropy loss for reconstruction. For methods with a temperature parameter (GS and STGS), the temperature is annealed exponentially from 1 to 0.01.

A.4. Compute Environment

The experiments were run on a single Nvidia A100 GPU in a cluster environment. Mixed precision floating point operations were used wherever possible. Less than 2 GB of GPU memory is needed to train the model and we expect that our results can be reproduced on significantly less powerful hardware with reasonable training times.

B. Image Reconstruction

This appendix includes reconstruction plots from the MNIST, Fashion MNIST, and KMNIST test sets, which are shown in Figure 3.

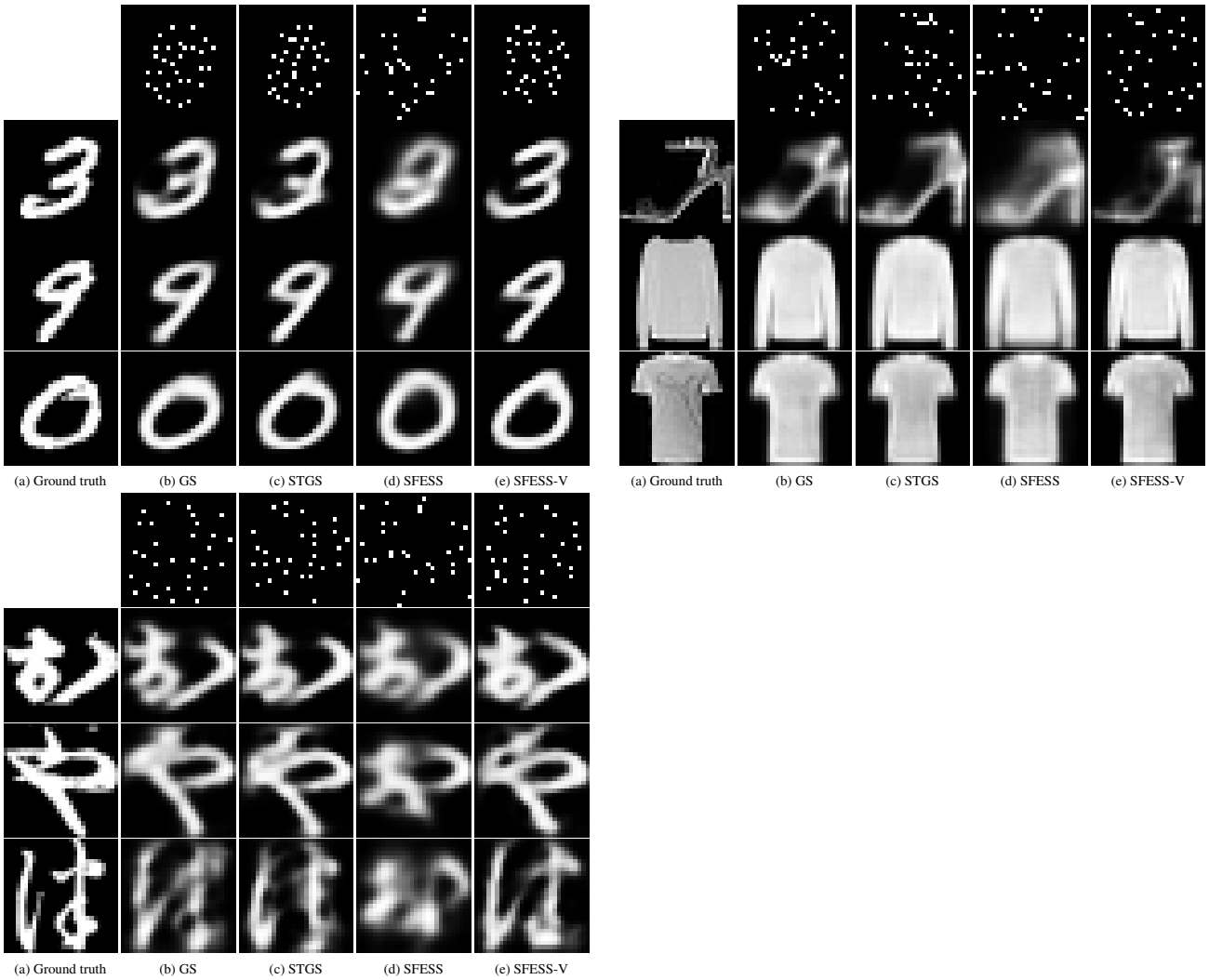

Figure 3: **Reconstruction plots**. For each dataset, the first row shows the learned selection mask and the following rows different samples from the test data. The leftmost column (a) shows the ground truth images and the following (b–e) show reconstructions from the jointly trained decoder. From top to bottom, left to right the datasets shown are MNIST, Fashion MNIST, and KMNIST.

C. Convergence of Validation Metrics

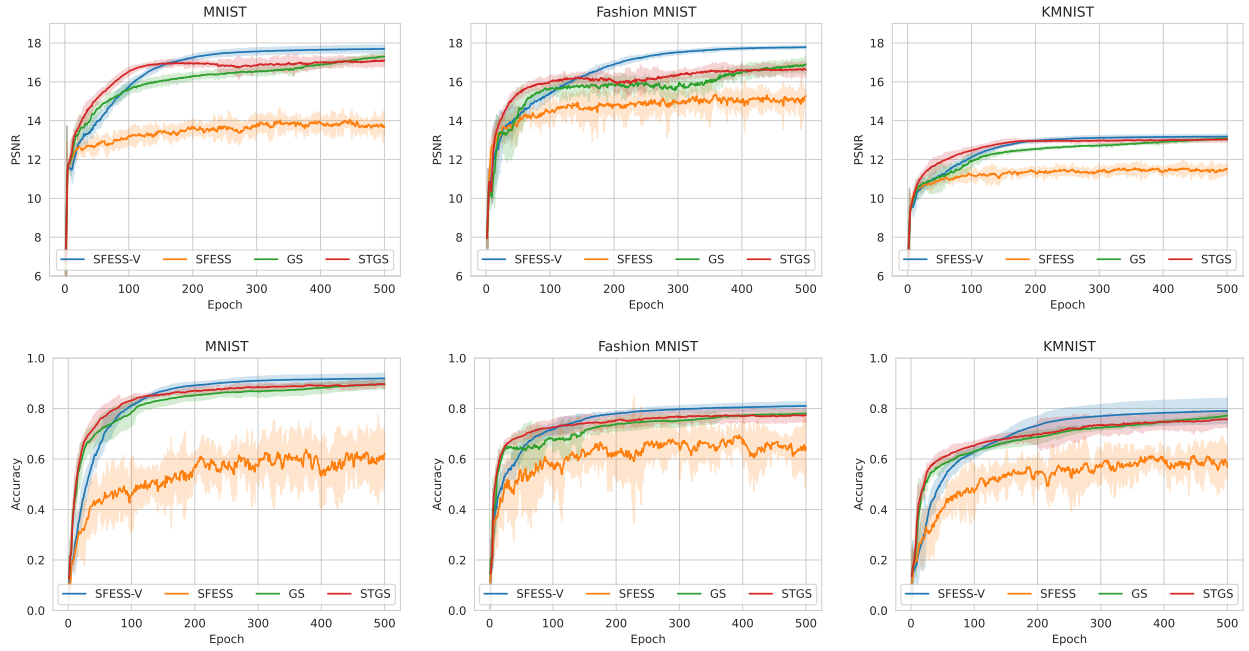

Figure 4: **Convergence plots (validation)**. Convergence of validation metrics with $k = 30$ selections. The top row shows reconstruction PSNR and the bottom row classification accuracy. The confidence intervals show two standard deviations computed for 5 repeated runs with different random seeds.

D. Deriving the Score Function Estimator

In this appendix, we derive the score function estimator (R. J. Williams 1992) which provides a Monte-Carlo estimate of the objective's gradient. We adapt the proof from Mohamed et al. (2020) (with annotations added):

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}[f_{\boldsymbol{\phi}}(\boldsymbol{z})] = \nabla_{\boldsymbol{\theta}} \sum_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{z}) f_{\boldsymbol{\phi}}(\boldsymbol{z}) \qquad \text{By definition of } \mathbb{E} \tag{3}$$

$$= \sum_{\boldsymbol{z}} \nabla_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\boldsymbol{z}) f_{\boldsymbol{\phi}}(\boldsymbol{z}) \qquad \text{Interchange gradient and summation}$$

$$= \sum_{\boldsymbol{z}} p_{\boldsymbol{\theta}}(\boldsymbol{z}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}) f_{\boldsymbol{\phi}}(\boldsymbol{z}) \qquad \text{By } \log \text{ derivative rule}$$

$$= \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{z})}[f(\boldsymbol{z}; \boldsymbol{\phi}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z})] \qquad \text{By definition of } \mathbb{E} \tag{4}$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} f_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)}) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)}) \qquad \text{Monte-Carlo estimate} \tag{5}$$

By the law of large numbers, the Monte-Carlo estimator in Equation (5) converges to the expected value in Equation (4) as $N \to \infty$, which is exactly the value of the true gradient in Equation (3). Hence, the estimator is an unbiased estimator of the true gradient.

E. Score Function Calculation

A key component of SFESS is calculating the score function. The unconditional independent Bernoulli distribution is renormalized by the Poisson-Binomial distribution. This renormalization factor is calculated following Fernandez and S. Williams (2010). Listing 1 outlines this calculation in pseudocode. Figure 5 shows an example of the resulting scores.

Listing 1 PyTorch-style pseudocode for calculating the Poisson-Binomial PMF (Fernandez and S. Williams 2010).

```
import torch
import cmath

def poibin_prob(theta, k):
    n = theta.size(0)
    i = torch.arange(n + 1).unsqueeze(-1)
    c = cmath.exp(2j * torch.pi / (n + 1))
    prod = torch.prod(theta * c**i + (1 - theta), dim=1)
    probs = torch.fft.fft(prod).real / n
    return probs[k]
```

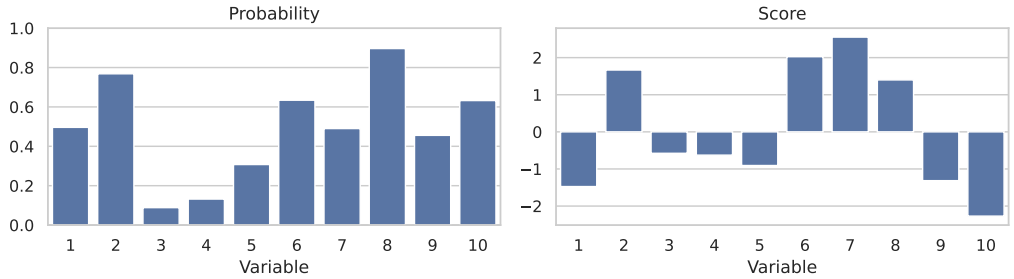

Figure 5: **Scores**. An example of scores $\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta},k}(\boldsymbol{z})$ (right) for the parameters (probabilities) $\boldsymbol{\theta}$ (left) for $n = 10$ and $k = 4$. It is easy to identify the sample that was evaluated – included elements have a positive score. The scores were calculated using Equation (1), where the second term was computed using a DFT.