Neural Bloom Filters for One-Shot and Few-Shot Set Membership and Compression

Bhavik Patel
IIT Gandhinagar
Gandhinagar, Gujarat, India
bhavik.patel@iitgn.ac.in

Jinil Patel
IIT Gandhinagar
Gandhinagar, Gujarat, India
jinilkumar.patel@iitgn.ac.in

Pranav Patil
IIT Gandhinagar
Gandhinagar, Gujarat, India
pranav.patil@iitgn.ac.in

Abstract

We propose a Neural Bloom Filter (NBF) that uses a differentiable memory model to perform one-shot and few-shot set membership queries while achieving effective space compression. The NBF employs a learnable addressing mechanism and integrates a backup classical Bloom filter to ensure 0% false negatives.

Keywords

Bloom Filter, Neural Networks, One-Shot Learning, Few-Shot Learning

1 Introduction

Bloom filters are space-efficient data structures that answer set membership queries with a controlled false positive rate. Recently, learned Bloom filters have combined classical methods with neural networks to compress stored sets more effectively, particularly when data is structured. Traditional recurrent models (e.g., LSTMs) can learn one-shot set membership but do not scale on long sequences, while Memory Networks store separate embeddings per element, limiting compression. Our aim is to develop a Neural Bloom Filter (NBF) that:

- Uses a differentiable memory-augmented model to "hash" inputs via learned addressing.
- Supports one-shot and few-shot learning without backpropagation through time.
- Achieves competitive space compression and low false positive rates with a backup classical Bloom filter.

2 Related Work

Bloom filters, introduced by Burton Bloom[1] in 1970, are widely studied for set membership queries. Variants like counting and scalable Bloom filters address specific needs. Learned Bloom filters, proposed by Kraska et al. (2018)[4], use machine learning to predict membership, reducing space needs. Memory-augmented neural networks, such as Neural Turing Machines and Differentiable Neural Computers, handle external memory tasks but scale poorly for set membership with long sequences. Meta-learning[3] enables fast adaptation in few-shot tasks, yet its use in set membership is limited. This project combines memory-augmented networks and meta-learning for an efficient set membership solution.

3 Methodology

3.1 Core Design and Implementation

- **Controller Network:** Develop an encoder (e.g., a 3-layer CNN or an LSTM) to generate input embeddings.
- **Learnable Addressing:** Map embeddings to memory slots using a learnable address matrix.
- **Additive Write:** Update a shared memory matrix additively, analogous to a bitwise OR.
- **Memory Read:** Use an MLP to compute a scalar logit from the memory content.
- **Backup Filter:** Integrate a classical Bloom filter to correct any false negatives.

3.2 Novel Extensions

- **Alternative Addressing:** Replace softmax with sparsemax[2] or use differentiable top-k selection (via Gumbel-softmax)[5] to induce sparsity.
- **Few-Shot Extension:** Process multiple examples per storage set and aggregate updates to build a joint memory representation.

3.3 Training via Meta-Learning

Task Sampling: Each training episode samples:

- A storage set S from a training distribution.
- A set of queries $\{x_1, \ldots, x_t\}$ with binary labels (1 if $x_j \in S$, 0 otherwise).

Fast Adaptation: Write the entire storage set S into memory in one shot (or aggregate for few-shot).

Loss and Optimization: Compute the cross-entropy loss between predicted logits and true labels, and update parameters using backpropagation (without BPTT over long sequences).

3.4 Ablation Studies and Evaluation

- Compare addressing methods (softmax, sparsemax, differentiable top-k) in terms of computation, memory usage, and false positive/negative rates.
- Evaluate one-shot vs. few-shot settings on structured tasks.
- Use MNIST and synthetic datasets to assess compression efficiency and membership accuracy.

4 Datasets and Libraries

Given are the tentative datasets and libraries that will be used for the analysis.

Datasets:

- **MNIST**: Class-based vs uniform sampling
- **Synthetic**: Controlled set structure

Libraries: PyTorch for deep learning, NumPy for numerical tasks, scikit-learn for metrics.

5 Timeline

Below given is the tentative weekly timeline for the project.

- **Week 1:** Design and implement the controller network and addressing mechanism.
- **Week 2:** Implement memory write/read modules and integrate the backup Bloom filter.
- **Week 3:** Train the model via meta-learning and perform initial experiments on one-shot and few-shot tasks.
- **Week 4:** Conduct ablation studies, evaluate performance, and analyze results.

References

[1] Burton H. Bloom. 1970. Space/Time Trade-Offs in Hash Coding with Allowable Errors. *Commun. ACM* 13, 7 (1970), 422–426. https://doi.org/10.1145/362686.362692

[2] André FT Martins and Ramón F Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. https://arxiv.org/pdf/1602.02068

[3] Nal Rahaman, Amit Arora, Antoine Baratin, Franziska Draxler, Mengye Lin, Fred Hamprecht, and Samuel Schoenholz. 2019. On the Spectral Bias of Neural Networks. *arXiv preprint arXiv:1906.04304* (2019). https://arxiv.org/pdf/1906.04304

[4] Kapil Vaidya, Eric Knorr, Tim Kraska, and Michael Mitzenmacher. 2020. Partitioned Learned Bloom Filter. *arXiv preprint arXiv:2006.03176* (5 June 2020). https://arxiv.org/abs/2006.03176

[5] Herke van Hoof Wouter Kool and Max Welling. 2019. The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. https://arxiv.org/pdf/1903.06059