

MLDB Presentation

SLIDE: Sub-Linear Deep Learning Engine

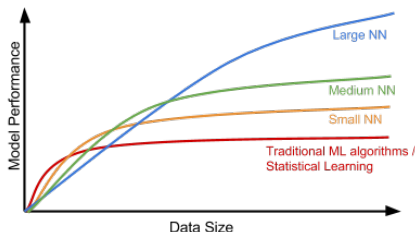
Beidi Chen Tharun Medini James Farwell Sameh Gabriel
Charlie Tai Anshumali Shrivastava

June 17, 2020

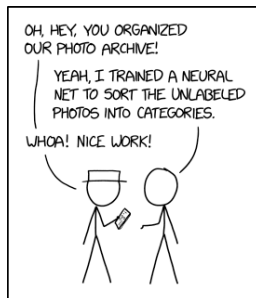
- 1 Motivation
 - Existing Approaches
 - Problem Setting
- 2 Contributions
- 3 Locality Sensitive Hashing
 - Sampling Approach to LSH
 - Additional LSH tricks
- 4 Implementation
- 5 Results

Motivation

Era of Deep Learning



(a) Model Performance wrt Dataset size



ENGINEERING TIP:
WHEN YOU DO A TASK BY HAND,
YOU CAN TECHNICALLY SAY YOU
TRAINED A NEURAL NET TO DO IT.

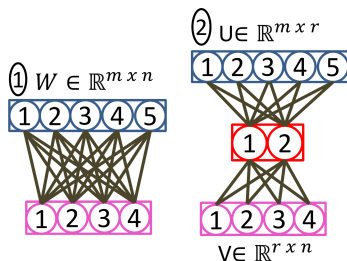
(b) Fun xkcd comic

- Large datasets → More Data
- Big models (Eg, 17B parameter NLP models)
- Improvements in optimizations and gradient descent
- **Matrix multiplication** is a computational bottleneck
- Many approaches exists such as **GPUs**

Existing Approaches

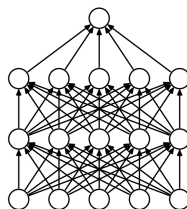
Low Rank structure

- $W \in \mathbb{R}^{m \times n}$ is weight matrix
- W has a low-rank structure $W = UV$
- $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$, where $r \ll \min(m, n)$
- Equivalent representation with $/$ activation function is better
- $\mathcal{O}(mn)$ becomes $\mathcal{O}(mr + rn)$
- Better storage of parameters as well
- But still needs dense gradient update, cannot parallelise asynchronously

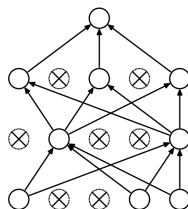


Dropout and Sparsity

- Well known regularization method for Neural Networks
- With probability p neurons in each layer is **turned off**
- Used during training to ensure model generalizes
- Sparsity above 50% tends to begin hurting performance



(a) Standard Neural Net



(b) After applying dropout.

Adaptive Dropout

- p is **chosen adaptively** based on activation of neurons in a layer
- And perform forward and backward propagation only on the **active neurons** in each layer
- Reduces the number of neurons to compute in each layer
- RELU is sparse in nature and filters negative activations
- [Ba and Frey, 2013] use a Bernoulli distribution proportional to activation of each neuron
- Still requires computing **all activations** to identify active neurons

Problem Setting

Sampling Problem

- Each layer has N time-varying sampling weights
- Weights a function of activations of neurons, $w_1^t, w_2^t, \dots, w_N^t$
- At each time t , we need to sample x_i with probability w_i^t
- Sampling cost is $\mathcal{O}(N)$ for each time step t

Sampling Problem

- Each layer has N time-varying sampling weights
- Weights a function of activations of neurons, $w_1^t, w_2^t, \dots, w_N^t$
- At each time t , we need to sample x_i with probability w_i^t
- Sampling cost is $\mathcal{O}(N)$ for each time step t

Problem

Can we efficiently sample the neurons in a layer to find the **active neurons** given the input to the layer?

Contributions

Main Contributions

[Chen et al., 2020] Contributions

- C++ OpenMP implementation for "standard" CPU
- Sparsity inspired, LSH based backpropagation algorithm
- Rigorous evaluation with TensorFlow(TF)-GPU and CPU
- Further optimizations using Hugepages and SIMD instructions

Main Contributions

[Chen et al., 2020] Contributions

- C++ OpenMP implementation for "standard" CPU
- Sparsity inspired, LSH based backpropagation algorithm
- Rigorous evaluation with TensorFlow(TF)-GPU and CPU
- Further optimizations using Hugepages and SIMD instructions

[Spring and Shrivastava, 2017] Contributions

- LSH and hash tables based identification of active neurons
- Sparse gradient updates in backpropagation
- Gains in computations using Asynchronous SGD (ASGD)

Locality Sensitive Hashing

Definition: LSH Family

A family \mathcal{H} is called (S_0, cS_0, p_1, p_2) -sensitive if for any two points $x, y \in \mathbb{R}^D$ and h chosen uniformly from \mathcal{H} satisfies the following:

- if $\text{Sim}(x, y) \geq S_0$ then $\Pr(h(x) = h(y)) \geq p_1$
 - if $\text{Sim}(x, y) \leq cS_0$ then $\Pr(h(x) = h(y)) \leq p_2$
-
- For approx nearest neighbour search, $p_1 > p_2$ and $c < 1$
 - $\Pr_{\mathcal{H}}(h(x)h(y)) \propto \text{Sim}(x, y)$ is sufficient for \mathcal{H} to be a LSH family
 - A LSH family is sufficient to solve nearest-neighbour search in **sub-linear time**
 - There is a **preprocessing** and **query** phase

LSH Preprocessing phase

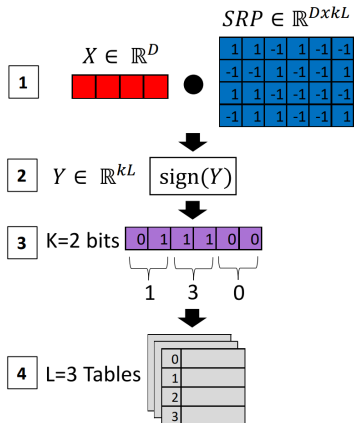


Figure 1: Locality Sensitive Hashing: (1) Compute the projection using a signed, random matrix $\mathcal{R}^{D \times kL}$ and the item $x \in \mathbb{R}^D$. (2) Generate a bit from the sign of each entry in the projection \mathbb{R}^{kL} . (3) From the kL bits, we create L integer fingerprints with k bits per fingerprint. (4) Add the item x into each hash table using the corresponding integer key

LSH query phase

- Now, given a query Q , we first compute the K hashes for each table
- Then we retrieve all the elements in the corresponding bucket of that table
- The union of all the elements from the L buckets is the **candidate set**
- for nearest-neighbour we then further filter the candidate set to get the possible answers
- K reduces false positives and L decreases false negatives
- An item x is in the candidate set with probability $1 - (1 - p^K)^L$
- Here $p \propto \text{Sim}(Q, x)$

Sampling Approach to LSH

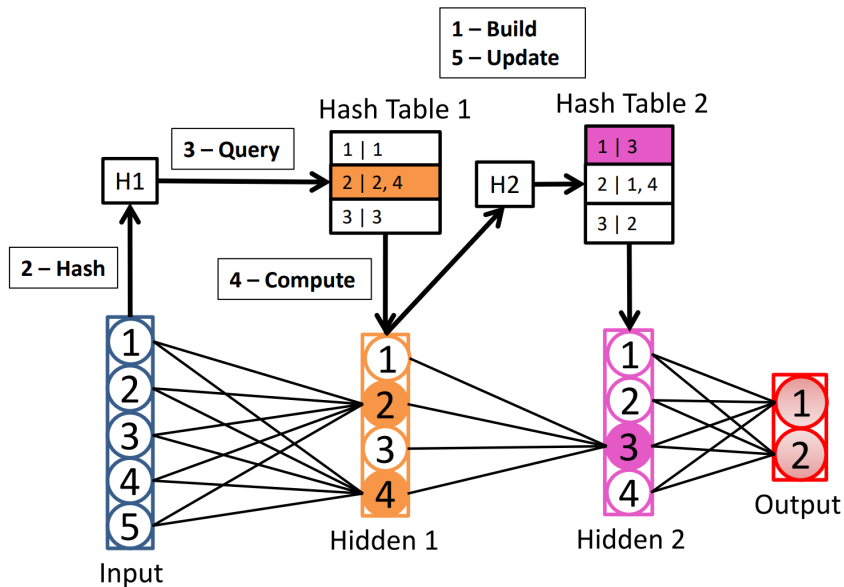
Hashing-Based Sampling

- Essentially LSH samples, in sub-linear time, elements with probability $1 - (1 - p^K)^L$
- Therefore, we can use the candidate set returned by LSH as the result of an adaptive sampling algorithm

Intuition

- Consider a node i with weight w_i and an input x
- Activation is monotonic function of $w_i^T \cdot x$
- Finding the **active neurons** is searching each w_i and finding the ones with largest inner product
- Can do this efficiently using LSH and hash tables

Intuition



Additional LSH tricks

① Hashing inner products

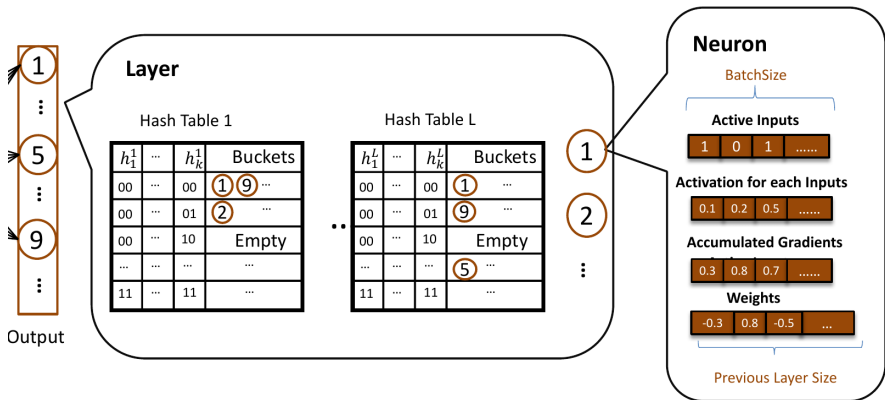
- Maximum Inner Product Search (MIPS): $p = \operatorname{argmax}_{x \in \mathcal{S}} q^T x$
- Nearest neighbour search (NNS): $p = \operatorname{argmin}_{x \in \mathcal{S}} \|q - x\|_2^2$
- Equivalent if norm of every element $x \in \mathcal{S}$ is constant. This is not the case for neural network weights.
- In regular LSH, $\Pr[h(x) = h(x)] = 1$ is closest neighbour for query x
- [Shrivastava and Li, 2014] proposes using an asymmetric transformation to solve MIPS using LSH

② Large L for good estimates in vanilla LSH

- [Lv et al., 2007] proposes Multi-probe LSH which reduces number of tables
- When querying, probe nearby buckets as well
- Do this by flipping a few bits in the hash for that table

Implementation

LSH Implementation



Overview of Implementation

- Each batch is **processed in parallel** using simple OpenMP
- Only active neurons take part in forward and backpropagation
- The **output softmax** is also computed only on active neurons
- After gradients are updates the hash tables are updated
- Can use Simhash or Densified Winner Take All (DWTA) hash families

- Highly sparse active set, as small 5% of all neurons
- Each update is computationally efficient
- Gradient updates are sparse \rightarrow parallel ASGD
- Can choose different sampling

Results

Questions or Comments



Ba, L. J. and Frey, B. (2013).

Adaptive dropout for training deep neural networks.

In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3084–3092, Red Hook, NY, USA. Curran Associates Inc.



Chen, B., Medini, T., Farwell, J., gobl, s., Tai, C., and Shrivastava, A. (2020).

Slide : In defense of smart algorithms over hardware acceleration for large-scale deep learning systems.

In *Proceedings of Machine Learning and Systems 2020*, pages 291–306.



Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. (2007).

Multi-probe lsh: efficient indexing for high-dimensional similarity search.

In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961.



Shrivastava, A. and Li, P. (2014).

Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips).

In *Advances in Neural Information Processing Systems*, pages 2321–2329.



Spring, R. and Shrivastava, A. (2017).

Scalable and sustainable deep learning via randomized hashing.

In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 445–454, New York, NY, USA. Association for Computing Machinery.