Improved Data Sampling Techniques for Training Neural Networks

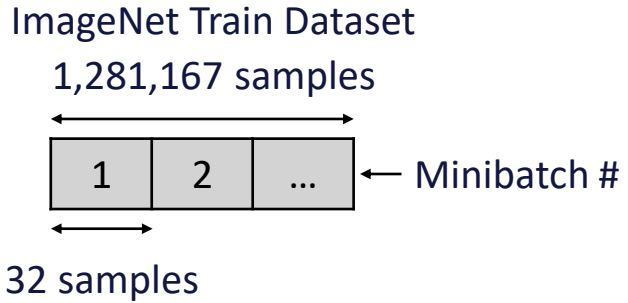
Karthik Rao Aroor

Al Engineer

Outline

- Minibatch gradient descent
- Minibatch sampling technique currently used, and its limitations
- Propose a new minibatch sampling technique
- Results
- Benefits
- Discussion
- Conclusions
- Future Work

Introduction



- Minibatch gradient descent for classification
- Random sampling, currently used
  - Randomly shuffle the training dataset
  - Pick minibatches sequentially
  - Desired outcome of shuffling
    - Samples from a single class are not clustered together (LeCun et al.)
    - Any minibatch has even representation from all classes, not just a single class
  - Actual outcome
    - Classes are not evenly represented in minibatches

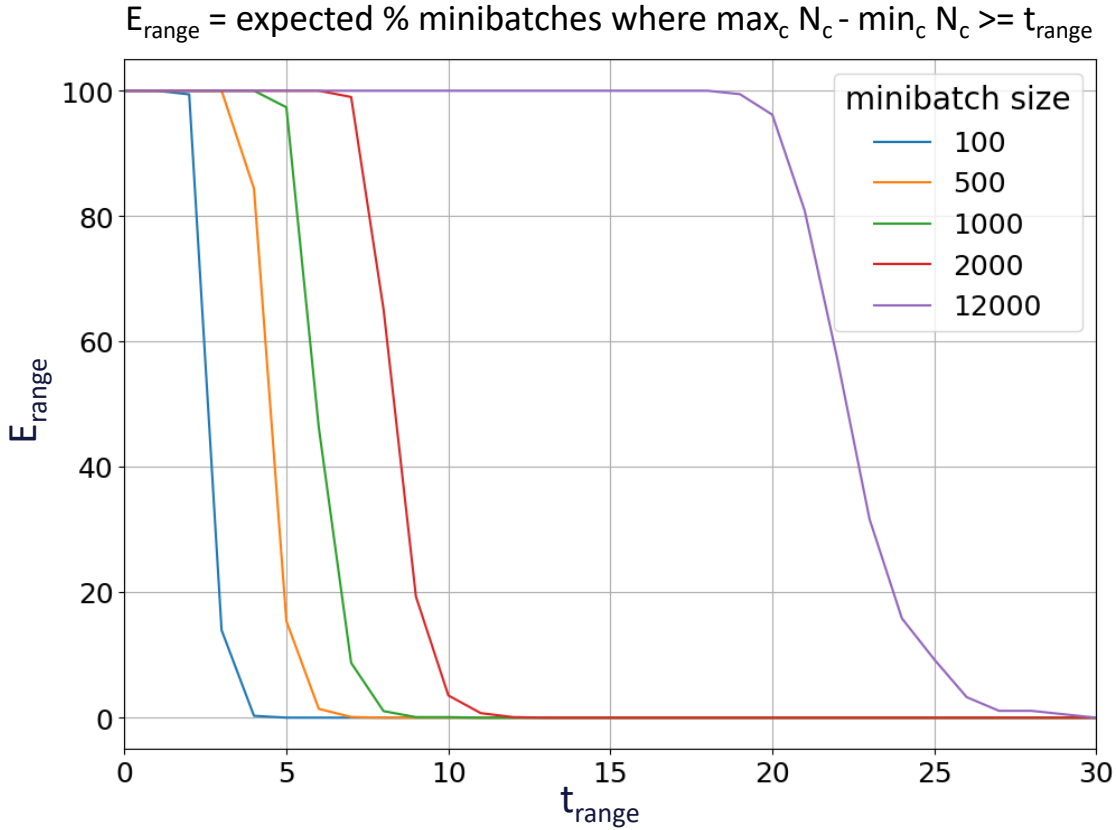
Introduction

- Proposed sampling
  - Classes are more evenly represented in minibatches
- Benefits of proposed sampling vs. random sampling
  - Higher accuracy on train, valid, test datasets, for a given # of training epochs
  - Faster training time for a given training accuracy
- Assumptions
  - Classification
  - All classes in the training dataset have the same number of samples

Random Sampling

- N_c = # of samples from class c in a minibatch
- Claim: N_c does not have uniform distribution
- Measure of uniformity
  - Uniform distribution means: N_c = constant for all c => $max_c\ N_c - min_c\ N_c = 0$
    - Only achieved if minibatch size = integer * # classes
  - If N_c != constant for all c, then $max_c\ N_c - min_c\ N_c$ should not be >> 0
  - E_{range} = expected % minibatches where $max_c\ N_c - min_c\ N_c >= t_{range}$
    - Where t_{range} = 1, 2,..
    - For uniform distribution, E_{range} should be 0 for $t_{range} > 1$

Random Sampling



Proposed Sampling

- Choose m, same for all minibatches
- Sort classes based on # samples remaining in the training dataset in descending order
  - For ex., # samples remaining in the training dataset - Class 1: 4, Class 2: 7, Class 3: 6
  - After sorting: Class 2, Class 3, Class 1
- Select q + 1 samples from each of the r classes with the largest # samples remaining in the training dataset
- Select q samples from each of the remaining C - r classes

Notation

Parameter	Symbol
# training samples	N
# classes	C
minibatch size	m = q C + r
quotient	q = floor(m/C)
remainder	r = mod(m, C)

Proposed Sampling Technique

Class i_1		Class i_r	Class i_{r+1}		Class i_C
q + 1	…	q + 1	q	…	q

r classes with the largest # samples remaining in the training dataset

C - r classes

Proposed Sampling

- Samples are chosen without replacement
- If r = 0, then m = q C
  - Choose q samples from each of the C classes
  - For ex., if m = 30, C = 10, then choose q = 3 samples from every class
- Once # samples per class are determined, samples for any given class are selected uniformly at random
  - For ex., if Class 1 has 100 samples, and we need to choose 4 samples, then choose any 4 out of the 100 samples

Proposed Sampling Example

- C = 4, N = 20*4 = 80, m = 11, q = 2, r = 3

q+1 samples

q samples

 # samples remaining in the training dataset

Batch #	Class 1	Class 2	Class 3	Class 4
Initial	20	20	20	20
1	17	17	18	17
2	15	14	15	14
3	12	11	12	12

 # samples in a minibatch

Batch #	Class 1	Class 2	Class 3	Class 4
Initial				
1	3	3	2	3
2	2	3	3	3
3	3	3	3	2

- If >= r classes have the largest # samples
  - Pick any r classes at random
  - For ex. ‘Initial’ row: we can choose any one of the following classes - (1, 2, 3); (1, 2, 4); (1, 3, 4); (2, 3, 4);
  - We choose (1, 2, 4) at random

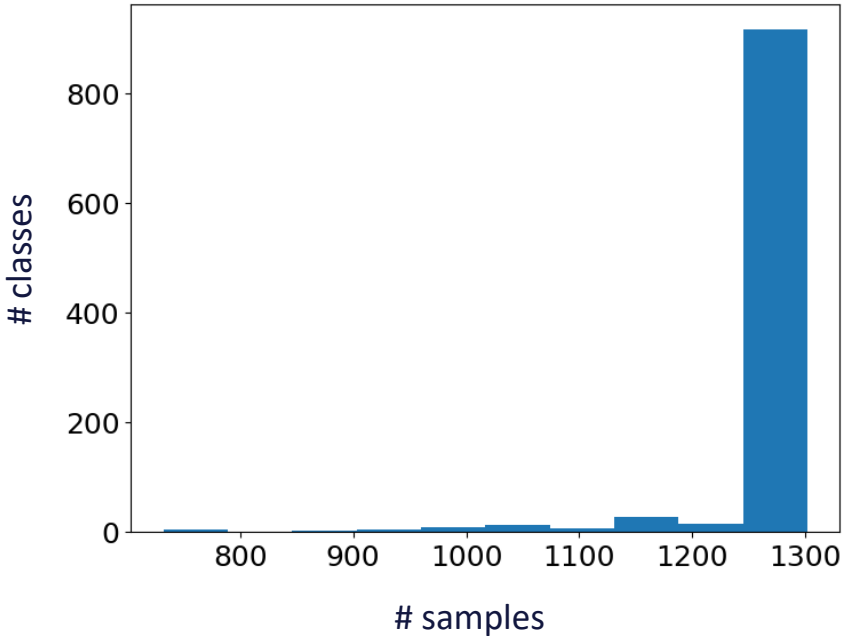
Evaluation Dataset

- ImageNet (ILSVRC2012)
- 1000 classes
- Train dataset
  - 1,281,167 samples
  - Not all classes have the same # samples : Max = 1300, Min = 732
  - Prune dataset to get 732 samples per class
  - 732,000 total samples after pruning
- Validation and Test dataset
  - 25,000 samples in each
  - No pruning required

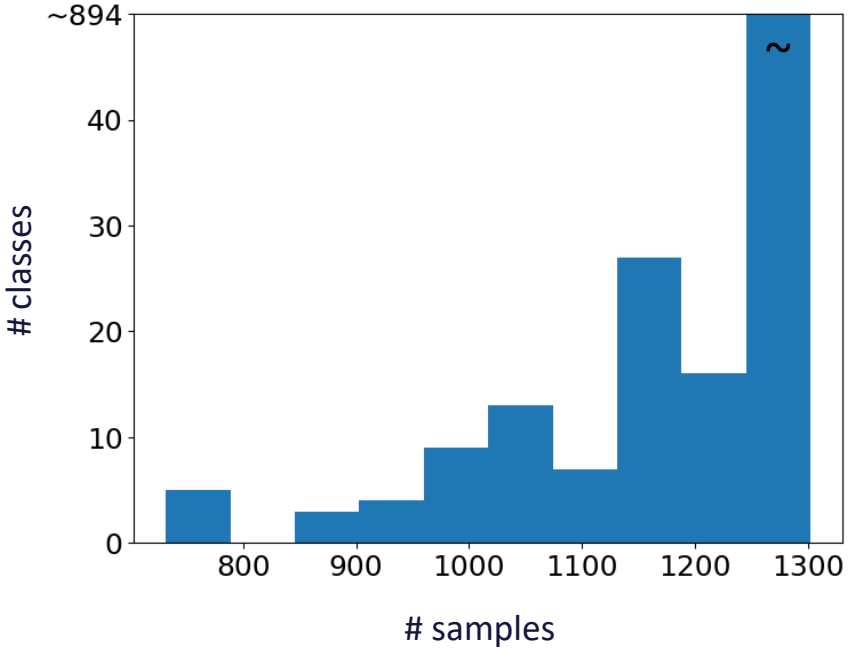
ImageNet, Training Dataset, # Samples

Original, unpruned dataset

histogram of
samples in different classes in the training dataset

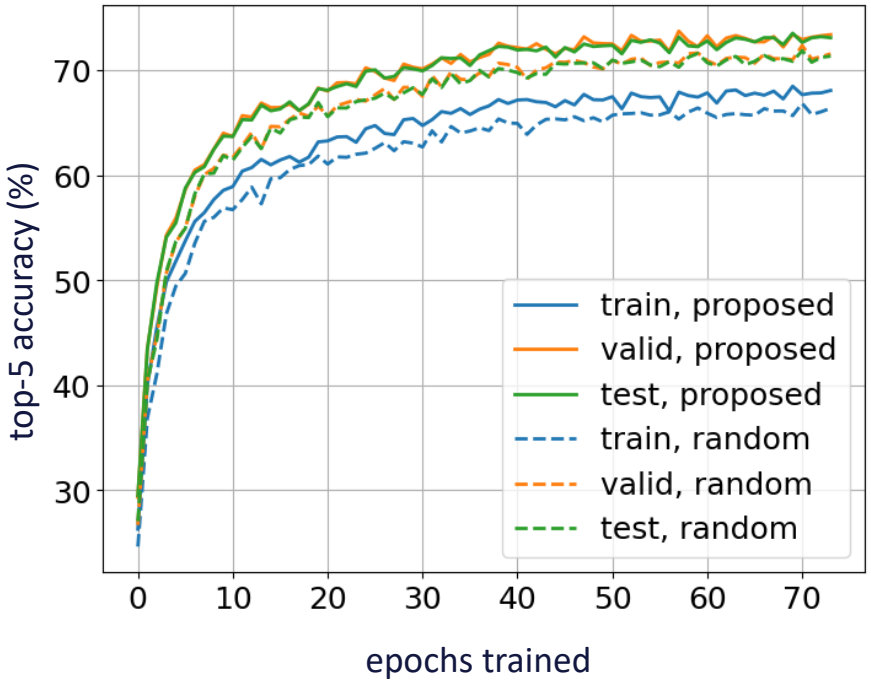


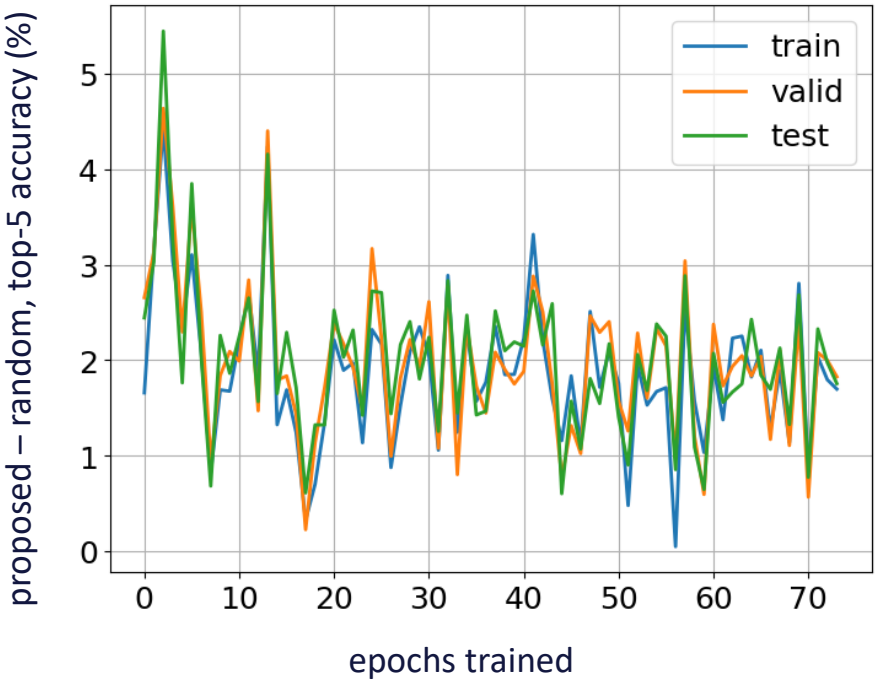
histogram zoomed



ImageNet, Accuracy vs. Epochs Trained

ResNet34, minibatch size = 100





ImageNet, Accuracy

For all cases, minibatch size = 100, learning rate = 0.0003, weight decay = 0.0005

Model	Random Train Accuracy (%)	Proposed Train Accuracy (%)	Random Valid Accuracy (%)	Proposed Valid Accuracy (%)	Random Test Accuracy (%)	Proposed Test Accuracy (%)
ResNet18	57.37	59.50	62.72	64.74	62.70	64.81
ResNet34	66.15	67.98	71.46	73.24	71.21	73.11
ResNet50	72.81	75.05	78.69	79.82	78.47	79.52
ResNet101	60.90	62.19	66.41	67.40	66.55	67.51

ImageNet, Number of Epochs

epochs needed to achieve a particular accuracy, i.e., random train accuracy from the previous slide

For all cases, minibatch size = 100, learning rate = 0.0003, weight decay = 0.0005

Model	Accuracy (%)	Random # epochs	Proposed # epochs
ResNet18	57.37	30	20
ResNet34	66.15	74	38
ResNet50	72.81	94	74
ResNet101	60.90	40	26

ImageNet, 80 Classes, Accuracy

Randomly choose 80 out of 1000 classes in ImageNet

Minibatch size = 8, learning rate = 0.0003, weight decay = 0.001

Model	Random Train Accuracy (%)	Proposed Train Accuracy (%)	Random Valid Accuracy (%)	Proposed Valid Accuracy (%)	Random Test Accuracy (%)	Proposed Test Accuracy (%)
ResNet18	79.19	80.02	83.87	84.64	81.77	82.61

Model	Accuracy (%)	Random # epochs	Proposed # epochs
ResNet18	79.19	24	20

Discussion

- Why is the proposed sampling technique better?
- Large minibatch sizes: uniformity in distribution of classes within a minibatch
- Small minibatch sizes: uniformity in distribution of classes across neighboring minibatches
  - Example, minibatch size = 1

Random Sampling

Minibatch 1	**Minibatch 2**		**Minibatch x**		
C_1	C_1	…	C_2	…	…

Proposed Sampling

Minibatch 1	**Minibatch 2**	**Minibatch 3**		**Minibatch 1000**	**Minibatch 1001**
C_1	C_7	C_{11}	…	C_{784}	C_1

Discussion

- Minibatch sizes
  - Analyzed sizes: 8, 10, 16, 20, 32, 60
  - Accuracy with proposed sampling is 1 % to 2 % higher than random sampling
    - Ran only about 10 – 20 epochs
  - Did not analyze size < 8 or size > 3000
    - < 8, too slow
    - > 3000, resource constraints

Discussion

- Datasets with different # of classes
  - ImageNet with 80 classes
    - Accuracy with Proposed sampling is 1 % to 2 % higher than random sampling
  - If # of classes is low, say 2 to 20
    - No significant improvement
- Ratio of minibatch size / # classes
  - Low values gives larger improvement for the first few epochs
  - Similar improvement for the later epochs

Discussion

- PyTorch
  - Typical training steps with only one function changed
  - Dataloader
    - Input: sampler
    - Output: a minibatch of samples
    - Create a custom sampler for the proposed sampling technique
      - Derived class from *torch.utils.data.Sampler*
      - Determine which train samples are present in each minibatch
      - Return indices of these samples

Conclusions

- Proposed a new minibatch sampling technique that addresses a limitation of random sampling
- Even representation of classes in a minibatch
  - Monte Carlo simulations
- On the ImageNet dataset, across various neural network architectures and minibatch sizes:
  - For a given number of epochs, proposed sampling has an accuracy that is 1% – 2% higher than random sampling
  - For a given accuracy, proposed sampling uses 10 – 30 fewer epochs than random sampling
- Limitations
  - Classification problems
  - All classes must have the same number of samples in the training dataset

Future Work

- Classification problems
  - Unequal number of samples in all classes in the training dataset
- Regression problems

Resources

ImageNet dataset

https://www.image-net.org/

References

Olga Russakovsky et al., ImageNet Large Scale Visual Recognition Challenge (2015)

https://link.springer.com/article/10.1007/s11263-015-0816-y

LeCun et al., Efficient BackProp (2012)

https://link.springer.com/chapter/10.1007/978-3-642-35289-8_3