Select Via Proxy: Efficient Data Selection For Training Deep Networks


Reproduced and Presentation by Paola Cascante-Bonilla. October 2019.
Motivation

Sample of cats & dogs images from Kaggle Dataset

https://www.kaggle.com/c/dogs-vs-cats
Motivation

Large amounts of annotated data available in multiple domains!

- Total number of non-empty synsets: 21841
- Total number of images: 14,197,122

http://image-net.org/about-overview
Motivation

Deep models to train all that data!

Top 1-Error of 4.2% on the ILSVRC 2012 classification benchmark [1]

[1] Rethinking the Inception Architecture for Computer Vision

https://cloud.google.com/tpu/docs/inception-v3-advanced
But...

• Training large models over all available data can be computationally expensive

• Training deep networks can incur prohibitively long training times, measured in days, weeks, or even months

• This overhead impedes the development of new machine learning models and uses large amounts of computational resources
Background

- Subsampling training data is a common solution.

- Core-set selection techniques is another solution.

https://www.thenewsletterpro.com/swimming-right-embracing-your-uniqueness/
**Background**

**proxy**  
[prok-see]  
SHOW IPA  

*noun, plural proxies.*

1. the agency, function, or power of a person authorized to act as the deputy or substitute for another.  
2. the person so authorized; substitute; agent.  
3. a written authorization empowering another person to vote or act for the signer, as at a meeting of stockholders.  
4. an ally or confederate who can be relied upon to speak or act in one's behalf.
Related Work

Core-set selection.
Find a representative subset of points to speed up learning or clustering.

K-means, Bayesian Inference, SVM.

Subset selection.
Choose data points whose predictions have changed most over the previous epochs as a lightweight estimate of uncertainty [2].

Using Reinforcement Learning → Student/teacher model.

accuracy and training time

[2] Active bias: Training more accurate neural networks by emphasizing high variance samples
Related Work

Heterogeneous active learning.
Use one model to select points for a different, more expensive model.

NLP Task -> CRF, Naïve Bayes, Maximum Entropy.

Optimization and Importance Sampling.
Based on Gradient norm, loss, focusing on more “hard” examples in later epochs.
Claim / Target Task

Using a **proxy model** reduces the cost of selection by up to a 100×

It can be easily added to a training pipeline without modifying the training procedure of the target model.

The proxy is very fast to train and can substantially improve the training time of large deep models while maintaining the predictive performance.
An Intuitive Figure Showing WHY Claim

Proposed Solution

Creating a Proxy Model.

- Scaling down the target model

Figure 2: Top-1 test error on CIFAR10 for varying model sizes (left) and over the course of training a single model (right), demonstrating a large amount of time is spent on small changes in accuracy.
Proposed Solution

Creating a Proxy Model.

- Training for smaller number of epochs
- Boosting the performance by ensembling small models.

Figure 3: Pearson product-moment correlation of examples ranked by entropy calculated from different models on CIFAR10 using ResNet20 and ResNet164 with pre-activation (left), SVHN using ResNet20 and ResNet152 (center), and Amazon Review Polarity using fastText and VDCNN29 (right). S1 and S2 represent two separate runs of the small proxy model (e.g., ResNet20), while L1 and L2 represent different runs of the large target model (e.g., ResNet164). R gives a random order of points for reference. On all datasets, ensembling multiple small models together through rank combination (SC) increases the Pearson product-moment correlation with the large model.
Proposed Solution

Subset Selection via Proxy.
Use the proxy model to select the most uncertain data points around the decision boundary.

Quantifying uncertainty:
• Confidence, margin, and entropy
• For every data point $x$ that provides $P(y|x)$ for $x$ to belong to class $y$, the uncertainty function $f$ can be defined as:

$$f_{\text{confidence}}(x) = 1 - P(\hat{y}|x)$$

$$f_{\text{margin}}(x) = 1 - \min_{y \neq \hat{y}} (P(\hat{y}|x) - P(y|x))$$

$$f_{\text{entropy}}(x) = - \sum_y P(y|x) \log P(y|x),$$
Proposed Solution

**Training the target model on the subsets selected via proxy.** The set of uncertain data points can be used to train the large target model.

1. Select the data points around the approximate decision boundary learned by the proxy model.

2. Let the target model refine the decision boundary of the proxy model.
Implementation

Algorithm 1 SELECT VIA PROXY (SVP)

**Input:** Data set $D$, cardinality $k$, deep model architecture $M$.

**Output:** Trained deep model $M^t$.

1: Create a proxy model by scaling down the target model as described in section 3.1.
2: Train the small proxy model on the entire dataset $D$.
3: Calculate uncertainty of data points via the proxy model using uncertainty metrics from section 3.2.
4: Sort the examples in a decreasing order based on their uncertainty.
5: Train the target model $M$ on the subset $S$ of top $k$ uncertain examples to get the final output $M^t$.
6: return $M^t$.


Data Summary

**CIFAR-10**

10 classes.
Train set: 50k images
Test set: 10k images
Balanced:
5k images per class

**SVHN**

10 classes.
Train set: 73257
Test set: 26032
Extra training data: 531131

**Amazon Review Polarity**

2 classes
Train samples: 3,600,000
Test samples: 400,000
Experimental Results

Training without data selection via proxy vs SVP algorithm:

(a) CIFAR10

(b) SVHN
Experimental Results

Average Top-1 error and standard deviation for 3 runs of different proxy models across a range of subset sizes of the CIFAR10, SVHN, and Amazon Review Polarity datasets:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Architecture</th>
<th>Proxy</th>
<th>Epochs ($n_p$)</th>
<th>0.4</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR10</td>
<td>3xResNet20</td>
<td>Entropy</td>
<td>50</td>
<td>6.52 ± 0.21</td>
<td>5.46 ± 0.06</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>1xResNet20</td>
<td>Entropy</td>
<td>50</td>
<td>6.83 ± 0.07</td>
<td>5.61 ± 0.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>1xResNet20</td>
<td>Entropy</td>
<td>180</td>
<td>7.09 ± 0.17</td>
<td>5.71 ± 0.22</td>
<td>5.53 ± 0.23</td>
<td>-</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>1xResNet164</td>
<td>Entropy</td>
<td>181</td>
<td>7.83 ± 0.32</td>
<td>6.31 ± 0.15</td>
<td>5.68 ± 0.25</td>
<td>5.48 ± 0.08</td>
</tr>
<tr>
<td>CIFAR10</td>
<td></td>
<td></td>
<td></td>
<td>8.93 ± 0.19</td>
<td>6.87 ± 0.16</td>
<td>6.07 ± 0.10</td>
<td>5.52 ± 0.12</td>
</tr>
<tr>
<td>SVHN</td>
<td>1xResNet20</td>
<td>Entropy</td>
<td>10</td>
<td>1.87 ± 0.03</td>
<td>1.72 ± 0.04</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SVHN</td>
<td>1xResNet20</td>
<td>Entropy</td>
<td>50</td>
<td>1.94 ± 0.20</td>
<td>1.86 ± 0.05</td>
<td>1.79 ± 0.02</td>
<td>-</td>
</tr>
<tr>
<td>SVHN</td>
<td></td>
<td></td>
<td></td>
<td>2.27 ± 0.06</td>
<td>1.98 ± 0.05</td>
<td>1.88 ± 0.04</td>
<td>1.79 ± 0.06</td>
</tr>
<tr>
<td>Amazon Review Polarity</td>
<td>1xfastText</td>
<td>Entropy</td>
<td>5</td>
<td>4.39 ± 0.02</td>
<td>4.23 ± 0.02</td>
<td>4.16 ± 0.02</td>
<td>-</td>
</tr>
<tr>
<td>Amazon Review Polarity</td>
<td></td>
<td></td>
<td></td>
<td>4.89 ± 0.03</td>
<td>4.50 ± 0.05</td>
<td>4.28</td>
<td>4.13 ± 0.04</td>
</tr>
</tbody>
</table>
Experimental Analysis

Comparison of uncertainty metrics:

(a) CIFAR10

(b) SVHN
Experimental Results Reproduced

CIFAR10 – Resnet164 – 181 Epochs

Accuracy achieved: 7.42%  ->  Time: 260 minutes
Reported accuracy: 5.52%  ->  Reported time: 240 minutes

Using a Titan X GPU
- 40k as training set
- 10k as validation set
- 10k as test set
Experimental Results Reproduced

CIFAR10 – Resnet20 – 50 Epochs

Accuracy achieved: ~11.47%  ->  Time: ~7.45 minutes
Reported accuracy: 9.2%  ->  Reported time: 12 minutes

2x for Model Ensemble
Experimental Results Reproduced

Dataset: CIFAR10

<table>
<thead>
<tr>
<th>Proxy</th>
<th>Fraction of Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
<td>Architecture</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>2xResNet20</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>1xResNet20</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>1xResNet164</td>
</tr>
</tbody>
</table>

Details:

Splits:
- 40k training set
- 10k validation set
- 10k testing set

Learning rate: 0.1 with a decay step of 0.1 every 30 epochs
Batch size: 64
SGD: momentum 0.9 – weight decay: 0.0005
Experimental Results Reproduced

SVHN – Resnet152 – 150 Epochs

Accuracy achieved: 4.09%  ->  Time: 294 minutes
Reported accuracy: 1.79%  ->  Reported time: 480 minutes

Using a Titan X GPU
Experimental Results Reproduced

SVHN – Resnet152 – 50 Epochs

Accuracy achieved: 4.81%  ->  Time: 14.40 minutes (50 epochs)
Reported accuracy: 1.79%  ->  Reported time: 13.4 minutes (10 epochs)

Using a Titan X GPU
Experimental Results Reproduced

Dataset: SVHN

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Architecture</th>
<th>Metric</th>
<th>Proxy</th>
<th>Fraction of Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVHN</td>
<td>1xResNet20</td>
<td>Confidence</td>
<td>50</td>
<td>0.4 0.6 0.8 1</td>
</tr>
<tr>
<td>SVHN</td>
<td>1xResNet152</td>
<td>No Proxy</td>
<td>50</td>
<td>5.37 5.07 4.96 4.81</td>
</tr>
</tbody>
</table>

Details:
Splits:
- 66,257 training set
- 7k validation set
- 10k testing set

Learning rate: 0.1 with a decay step of 0.1 every 30 epochs
Batch size: 64
SGD: momentum 0.9 – weight decay: 0.0005
Experimental Analysis

Comparison of uncertainty metrics:

```python
#confidence = 1 - P(y'|x)

def get_correct_values(model, device, data_loader, indices):
    max_correct_results = {}
    model.eval()
    with torch.no_grad():
        for i, (data, target) in enumerate(data_loader):
            data, target = data.to(device), target.to(device)
            output = model(data)
            out_softmax = F.softmax(output, dim=1)
            pred = out_softmax.argmax(dim=1, keepdim=True) # get the index of the max log-probability
            for t, p in enumerate(pred):
                if p == target[t]:
                    max_correct_results[indices[(i*args.batch_size)+t]] = out_softmax[t][pred[t]].item()

    return max_correct_results

#entropy = -sum P(y|x) log P(y|x)

def get_correct_values(model, device, data_loader, indices):
    max_correct_results = {}
    model.eval()
    with torch.no_grad():
        for i, (data, target) in enumerate(data_loader):
            data, target = data.to(device), target.to(device)
            output = model(data)
            out_softmax = F.softmax(output, dim=1)
            pred = out_softmax.argmax(dim=1, keepdim=True) # get the index of the max log-probability
            entropy = (-out_softmax.detach().cpu().numpy() * np.log2(out_softmax.detach().cpu().numpy())).sum(axis=1)
            for t, p in enumerate(pred):
                if p == target[t]:
                    max_correct_results[indices[(i*args.batch_size)+t]] = entropy[t].item()

    return max_correct_results
```
Conclusion and Future Work

• A small proxy model can select a subset of data to train a large architecture while maintaining the predictive performance.

• On CIFAR10 and SVHN, the speed of training the proxy model leads to a 1.6× and 1.8× speed-up in end-to-end training time by selecting 60% and 50% of data respectively to train the target model on.

• Train on larger datasets. More experiments without making multiple passes over all the data.
References

[1] Rethinking the Inception Architecture for Computer Vision

[2] Active bias: Training more accurate neural networks by emphasizing high variance samples

[3] Very Deep Convolutional Networks for Text Classification


[5] Bag of Tricks for Efficient Text Classification
Appendix

```python
model_state = torch.load('resnet20_100percent_best_50Epochs_STEP30_PreActivations_lr0.1_Best.ckpt')
model.load_state_dict(model_state)
print ('100 Percent \nBest ResNet20 model - Test Accuracy:"
acc = test(args, model, device, test_loader, class_criterion_eval)
total_time = end_time - start_time
print ('Time: {}'.format(total_time))
print ('=' * 20)
x_epochs = range(0, args.epochs)
plot_acc_valtest(x_epochs, 100 - np.array(record_val_acc), 100 - np.array(record_test_acc))
```

100 Percent
Best ResNet20 model - Test Accuracy:
Average loss: 0.3406, Accuracy: 8842/10000 (88.42%)

Time: 450.59966111183167

=====================

![Graph showing accuracy over epochs]

0 10 20 30 40 50
epoch

10 20 30 40 50
ACCURACY

Val Acc
Test Acc

0 10 20 30 40 50
epoch
model_state = torch.load('resnet164_100_best_181Epochs_Step30_PreActivations_lr0.1_Best.ckpt')
model.load_state_dict(model_state)

print("100 Percent \nBest ResNet181 model - Test Accuracy:")
acc = test(args, model, device, test_loader, class_criterion_eval)
total_time = end_time - start_time
print('{Time: {}' .format(total_time))
print('--------------------------')
x_epochs = range(0, args.epochs)
plot_acc_valtest(x_epochs, 100 - np.array(record_val_acc), 100 - np.array(record_test_acc))

100 Percent
Best ResNet181 model - Test Accuracy:
Average loss: 0.3159, Accuracy: 9258/10000 (92.58%)

Time: 15605.45658475538
--------------------------
CIFAR10
0.4 of dataset
CIFAR10
0.8 of dataset