Google AI

Neumann Optimizer

A Practical Optimization Algorithm for Deep Neural Networks

Shankar Krishnan, Ying Xiao, Rif A. Saurous
Google AI Perception Research

Training Deep Networks

- Loss function is non-convex
- Finite sample size
- Lower loss doesn't necessarily imply better model performance
- Computational hurdles

Iterative Methods

Create function approximation around the current iterate

Different methods result depending on information used for the approximation

Iterative Methods

Create function approximation around the current iterate

Different methods result depending on information used for the approximation

1st order: $\mathcal{F}(w_t) + \nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2\eta} ||z - w_t||^2$

Iterative Methods

Create function approximation around the current iterate

Different methods result depending on information used for the approximation

1st order: $\mathcal{F}(w_t) + \nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2\eta} ||z - w_t||^2$

2nd order: $\nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2}(z - w_t)^T \nabla^2 \mathcal{F}(w_t)(z - w_t)$

Iterative Methods

Create function approximation around the current iterate

Different methods result depending on information used for the approximation

1st order: $\mathcal{F}(w_t) + \nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2\eta} \|z - w_t\|^2$

2nd order: $\nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2}(z - w_t)^T \nabla^2 \mathcal{F}(w_t)(z - w_t)$

Cubic reg.: $\nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2}(z - w_t)^T \nabla^2 \mathcal{F}(w_t)(z - w_t) + \frac{\alpha}{3} \|z - w_t\|^3$

Neumann Optimizer

- Designed for large batch setting
- Incorporates second order information, without explicitly representing the Hessian
- Based on cubic regularized approximation
- Total per step cost is same as Adam

Neumann Optimizer - Results

- Linear speedup with increasing batch size
- 0.8 - 0.9% test accuracy improvements over baseline on image models
- 10-30% faster than current methods for comparable computational resources

Two Loop Algorithm

- For each mini-batch: solve the quadratic problem of the mini-batch to "good" accuracy.

$$\mathcal{G}(z) = \mathcal{F}(w_t) + \nabla \mathcal{F}(w_t)^T (z - w_t) + \frac{1}{2}(z - w_t)^T \nabla^2 \mathcal{F}(w_t)(z - w_t),$$

- Use the solutions of the quadratic problem as updates.

$$w_{t+1} = w_t - \left[\nabla^2 \mathcal{F}(w_t)\right]^{-1} \nabla \mathcal{F}(w_t).$$

- Hypothesis: with very large mini-batches, the cost-benefit is in favour of extracting more information.

Inner loop: solving a quadratic via power series

How to solve the quadratic?

$$Az = b$$

Use the geometric (Neumann) series for inverse, if $A \succ 0, \|A\|_2 < 1$

$$A^{-1} = \sum_{i=0}^{\infty} (I_n - A)^i.$$

This implies the iteration:

$$z_0 = b \qquad \text{and} \qquad z_{t+1} = (I_n - A) z_t + b,$$

Eliminate the Hessian calculation (too expensive)

Back to the mini-batch problem:

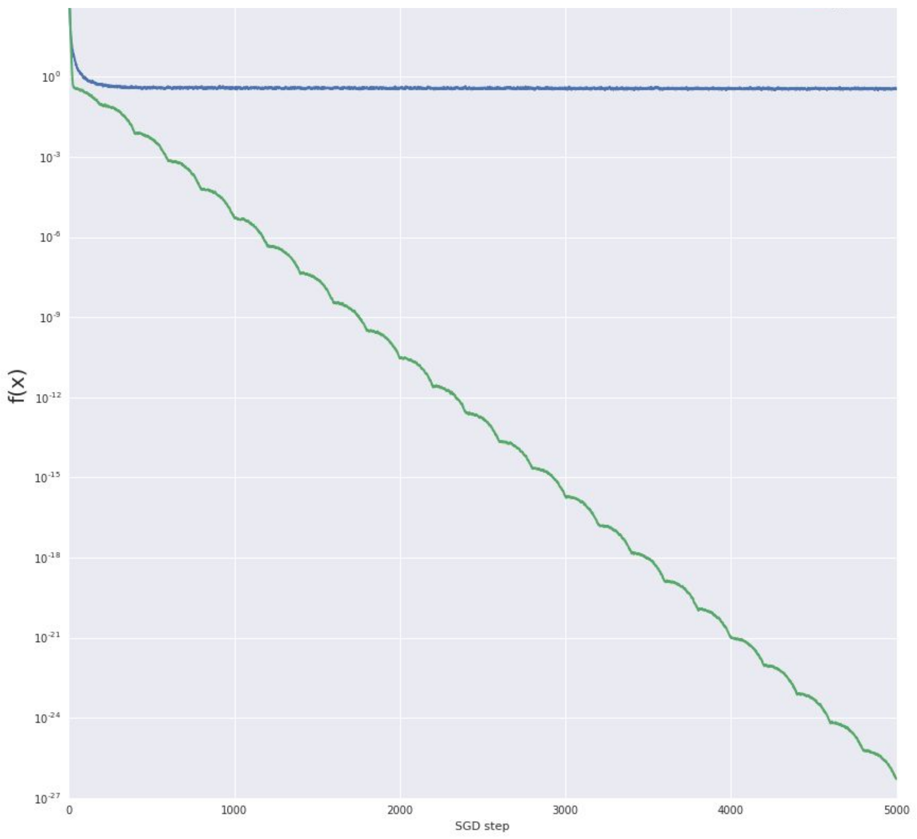
$$\nabla^2 \mathcal{F}(w_t)(w - w_t) = -\nabla \mathcal{F}(w_t)$$

This implies an iteration:

$$\begin{aligned} m_{t+1} &= (I_n - \eta \nabla^2 \hat{f}) m_t - \nabla \hat{f}(w_t) \\ &= m_t - (\boldsymbol{\nabla} \hat{\boldsymbol{f}}(\boldsymbol{w_t}) + \boldsymbol{\eta} \boldsymbol{\nabla}^{\mathbf{2}} \hat{\boldsymbol{f}} \boldsymbol{m_t})) \\ &\approx m_t - \boldsymbol{\nabla} \hat{\boldsymbol{f}}(\boldsymbol{w_t} + \boldsymbol{\eta} \boldsymbol{m_t}). \end{aligned}$$

Google

Convex Quadratics



Some tweaks for actual neural networks

Cubic regularizer (convexification of overall problem) + Repulsive regularizer

$$\hat{g}(w) = \hat{f}(w) + \frac{\alpha}{3}\|w - v_t\|^3 + \beta / \|w - v_t\|$$

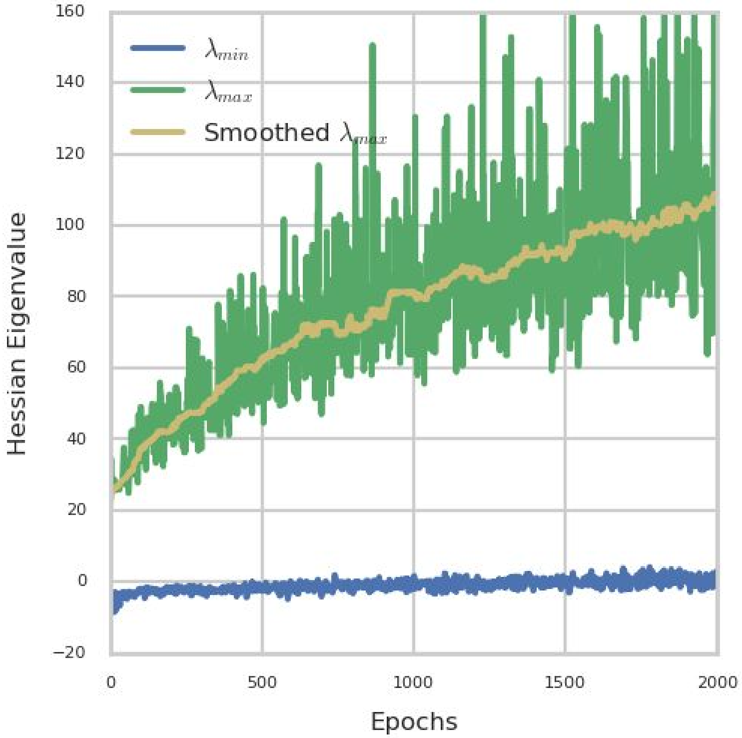
Convexification of each mini-batch.

$$\mu = \frac{\lambda_{\max}}{|\lambda_{\min}| + \lambda_{\max}} \quad \text{and} \quad \eta = \frac{1}{\lambda_{\max}}.$$

$$m_k = \mu m_{k-1} - \eta \nabla \hat{g}(w_t + \mu m_{k-1}).$$

Spectrum of the Hessian

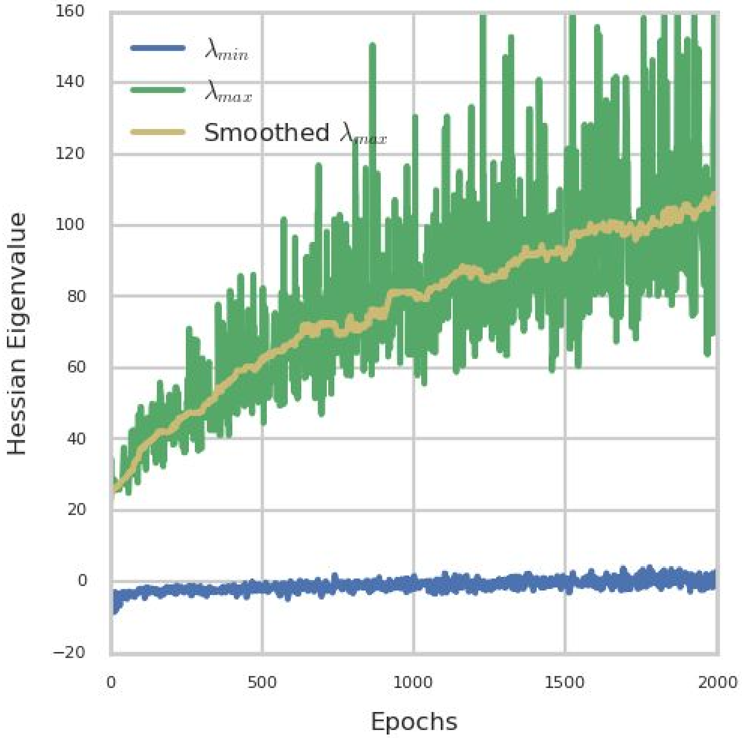
- Studied on various image models
- Lanczos method during optimization
- Similar behavior of extreme eigenvalues
- Consistent with other studies [Sagun ‘16, Chaudhari ‘16, Dauphin ‘14]



Spectrum of the Hessian

- Studied on various image models
- Lanczos method during optimization
- Similar behavior of extreme eigenvalues
- Consistent with other studies [Sagun ‘16, Chaudhari ‘16, Dauphin ‘14]

$$\mu \propto 1 - \frac{1}{1+t} \qquad \eta \propto 1/t$$



Our Algorithm

for $t = 1, 2, 3, \ldots, T$ **do**

Draw a sample $(x_{t_1}, y_{t_1}), \ldots, (x_{t_B}, y_{t_B})$.

Compute derivative $\nabla \hat{f} = (1/B) \sum_{i=1}^{B} \nabla \ell(y_{t_i}, g(x_{t_i}, w_t))$

Compute update $d_t = \nabla \hat{f} + \left(\alpha \|w_t - v_t\|^2 - \frac{\beta}{\|w_t - v_t\|^2} \right) \frac{w_t - v_t}{\|w_t - v_t\|}$

Update Neumann iterate: $m_t = \mu(t) m_{t-1} - \eta(t) d_t$.

Update weights: $w_t = w_{t-1} + \mu(t) m_t - \eta(t) d_t$.

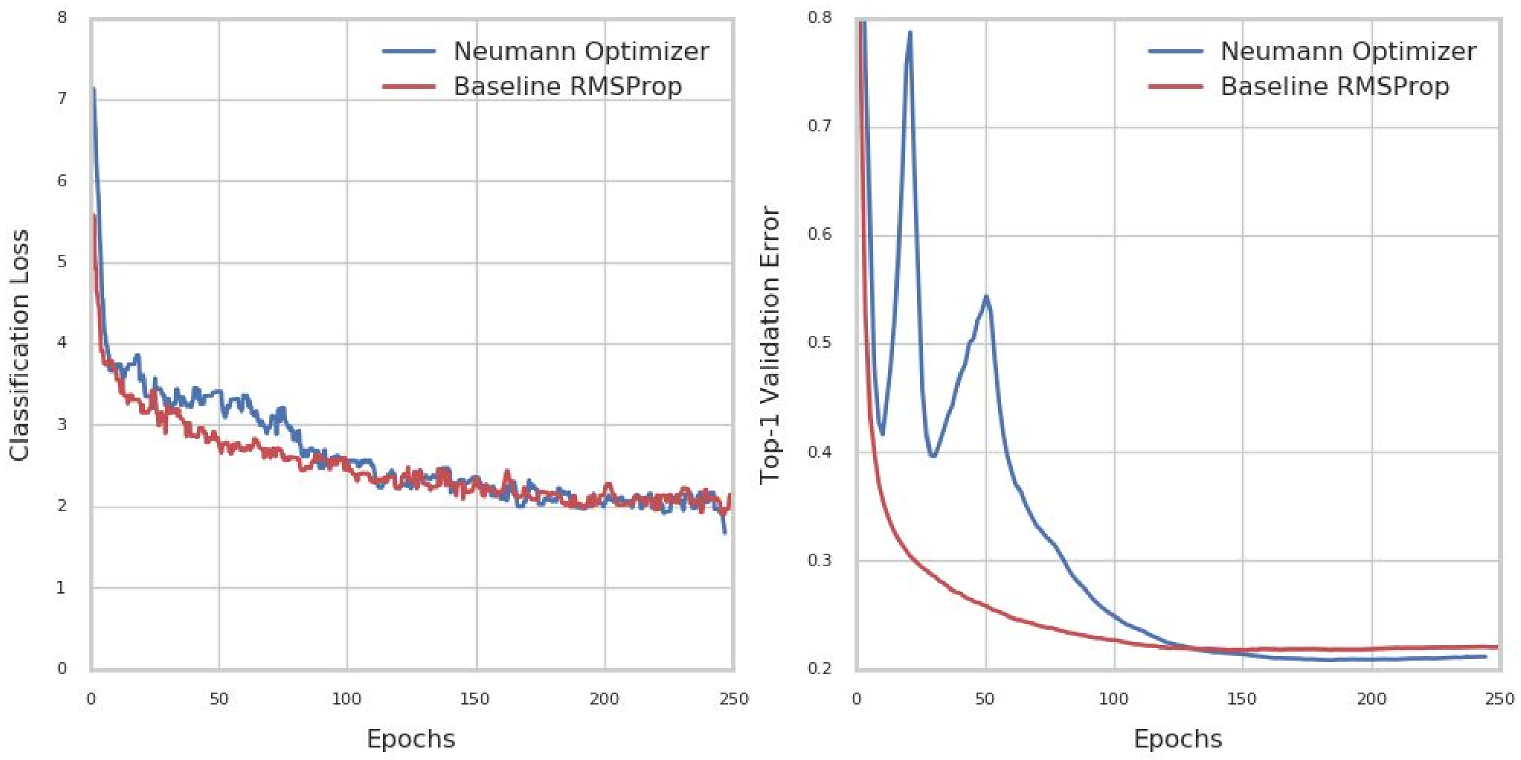
Update moving average of weights: $v_t = w_t + \gamma(v_{t-1} - w_t)$

return $w_T - \mu(T) m_T$

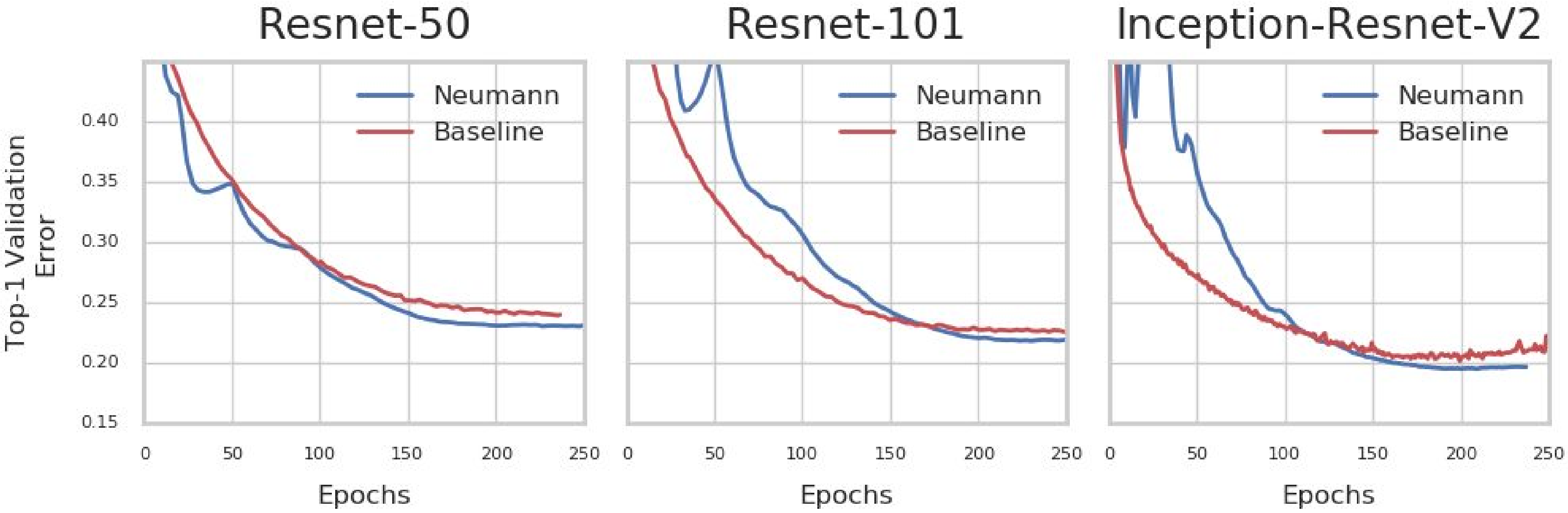
Experimental Results

- Used image models for experimentation
    - Cifar10, Cifar100, Imagenet
- Various network architectures
    - Resnet-V1, Inception-V3, Inception-Resnet-V2
- Training on GPUs (P100, K40)
    - Multiple GPUs in sync mode for large batch training
- Update steps/epochs to measure performance

Experiments on ImageNet (Inception V3)



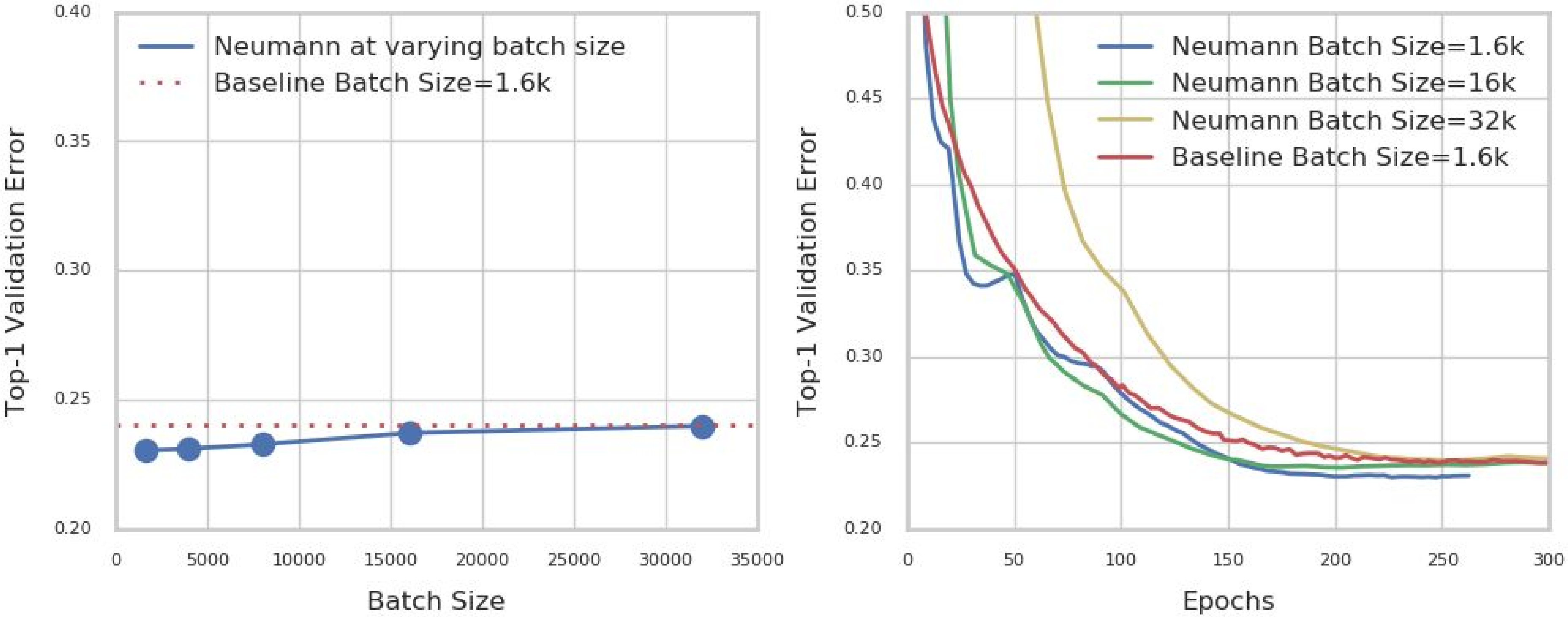
Experiments on ImageNet (Resnet Architectures)



Final Top-1 Validation Error

	Baseline	Neumann	Improvement
Inception-V3	21.7 %	20.8 %	**0.91%**
Resnet-50	23.9 %	23.0 %	**0.94 %**
Resnet-101	22.6 %	21.7%	**0.86 %**
Inception-Resnet-V2	20.3 %	19.5 %	**0.84 %**

Large batch training



Scaling Performance on Resnet-50

Batch Size	Top-1 Validation Error	# Epochs
1600	23.0 %	**226**
4000	23.0 %	**230**
8000	23.1 %	**258**
16000	23.5 %	**210**
32000	24.0 %	**237**

Mini-batch Training

Stochastic methods use unbiased estimates to carry out optimization

For first order methods with batch size B, $\hat{f}(w) = \frac{1}{B} \sum_{i=1}^{B} f_{t_i}(w)$

Mini-batch Training

Stochastic methods use unbiased estimates to carry out optimization

For first order methods with batch size B, $\hat{f}(w) = \frac{1}{B}\sum^{B} f_{t_i}(w)$

$$\mathbb{E}\left[\nabla \hat{f}(w)\right] = \nabla \mathcal{F}(w), \;\; \mathbb{E}\left[\left\|\nabla \hat{f}(w) - \nabla \mathcal{F}(w)\right\|^2\right] < \sigma^2$$

Mini-batch Training

Stochastic methods use unbiased estimates to carry out optimization

For first order methods with batch size B, $\hat{f}(w) = \frac{1}{B}\sum^{B} f_{t_i}(w)$

$$\mathbb{E}\left[\nabla \hat{f}(w)\right] = \nabla \mathcal{F}(w), \ \mathbb{E}\left[\left\|\nabla \hat{f}(w) - \nabla \mathcal{F}(w)\right\|^2\right] < \sigma^2$$

After T steps: $\mathbb{E}\left[\|\nabla \mathcal{F}(w)\|^2\right] \leq O\left(\frac{LD}{T} + \frac{\sigma\sqrt{LD}}{\sqrt{BT}}\right)$

Mini-batch Training

Stochastic methods use unbiased estimates to carry out optimization

For first order methods with batch size B, $\hat{f}(w) = \frac{1}{B}\sum^{B} f_{t_i}(w)$

$$\mathbb{E}\left[\nabla \hat{f}(w)\right] = \nabla \mathcal{F}(w),\ \ \mathbb{E}\left[\left\|\nabla \hat{f}(w) - \nabla \mathcal{F}(w)\right\|^2\right] < \sigma^2$$

After T steps: $\mathbb{E}\left[\|\nabla \mathcal{F}(w)\|^2\right] \leq O\left(\frac{LD}{T} + \frac{\sigma\sqrt{LD}}{\sqrt{BT}}\right)$

$$B \approx O\left(\frac{\sigma^2 T}{LD}\right)$$

Conclusion and Future Work

- Stochastic optimization algorithm for training deep nets
- Exhibits linear speedup with batch size
- Gives better model performance
- Total per step cost on par with existing popular optimizers

Conclusion and Future Work

- Stochastic optimization algorithm for training deep nets
- Exhibits linear speedup with batch size
- Gives better model performance
- Total per step cost on par with existing popular optimizers

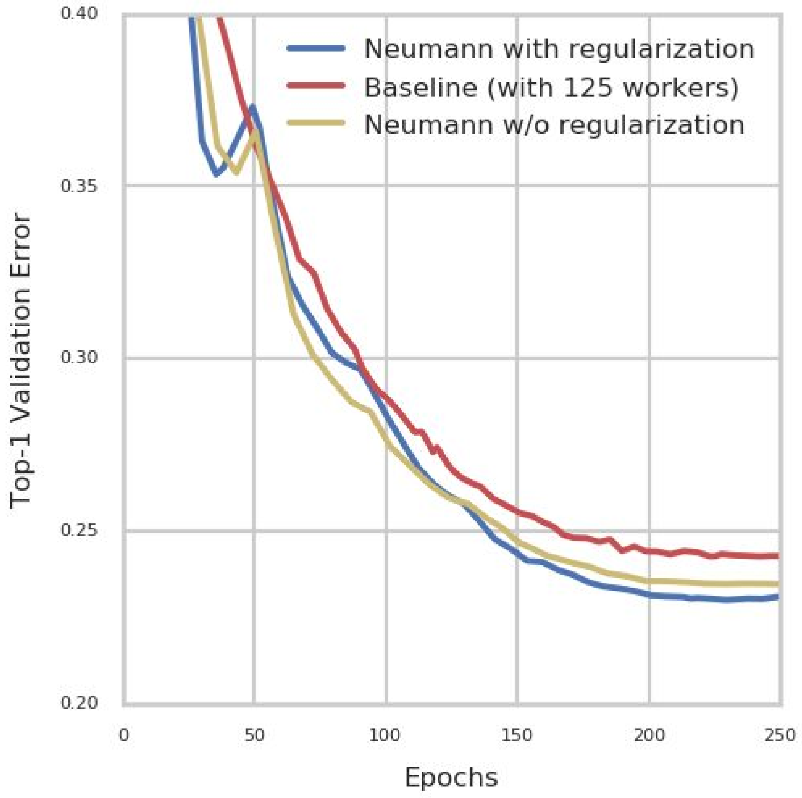
- Other model architectures and models
  - Preconditioned Neumann
- Experiment on TPUs

Thank you!

Paper: https://goo.gl/7M9Avr

Code: Tensorflow implementation coming soon

Ablation Experiment



Google