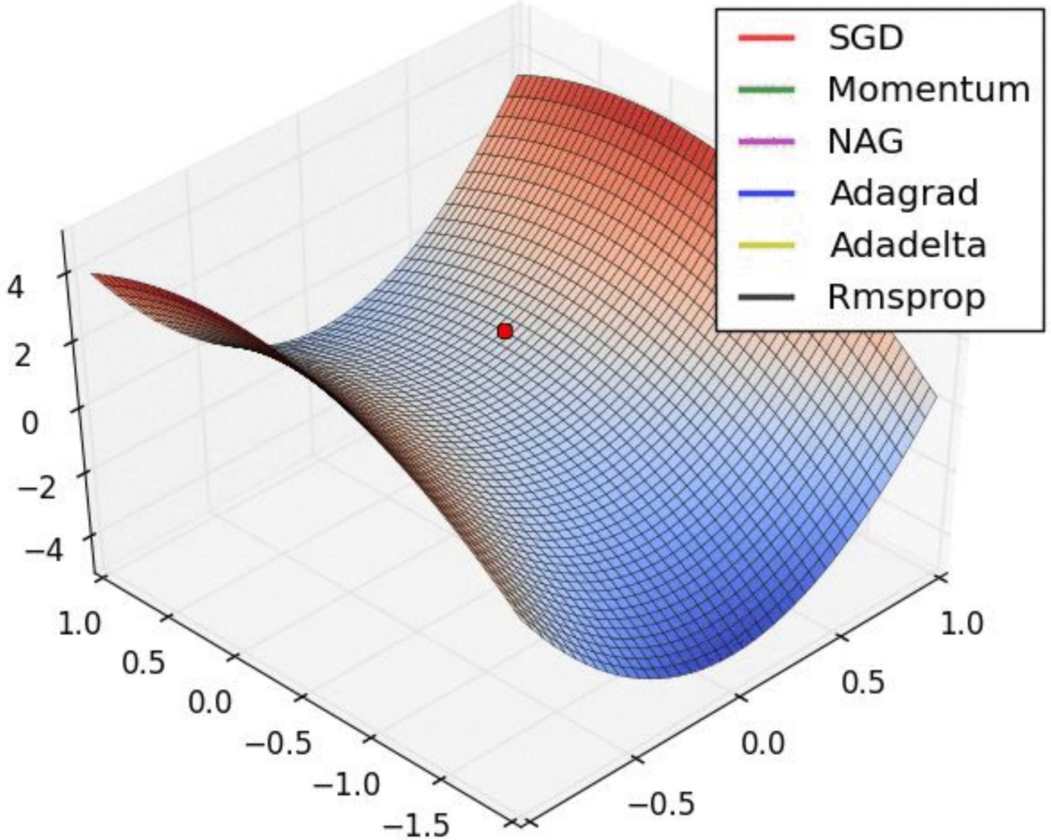
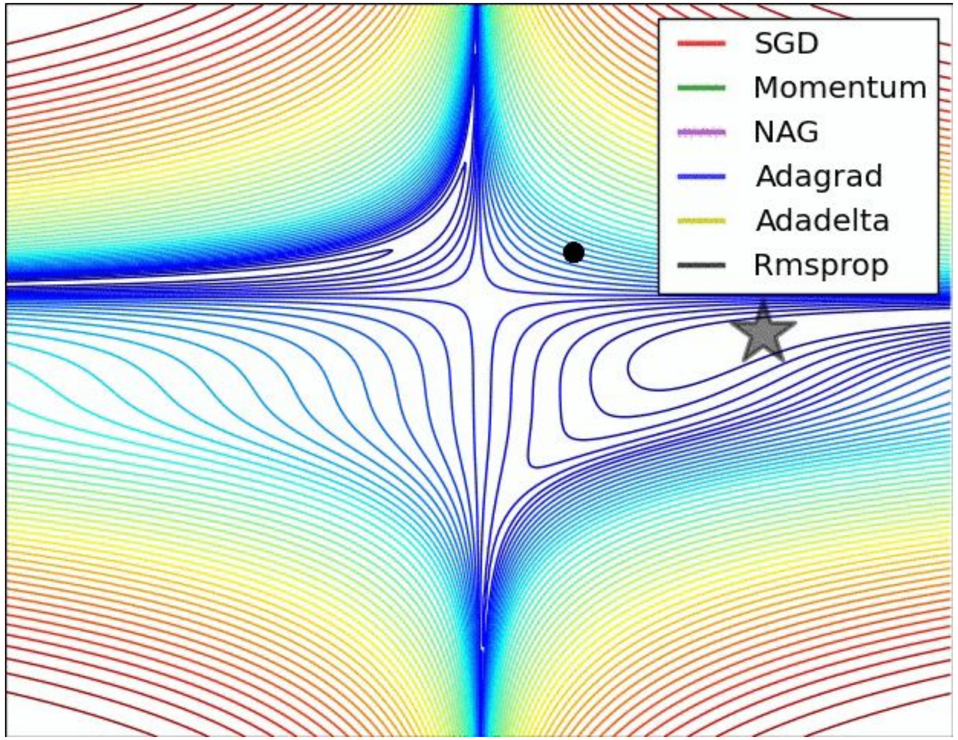
OPTIMIZERS



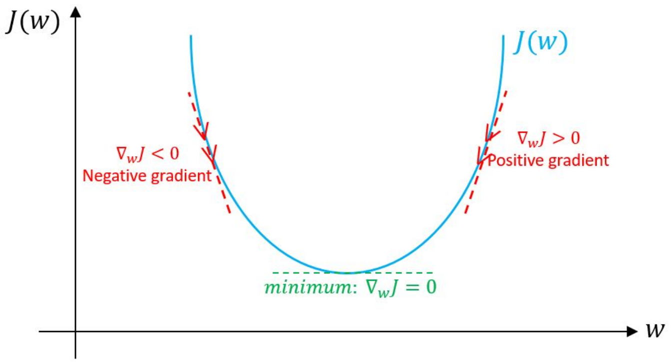


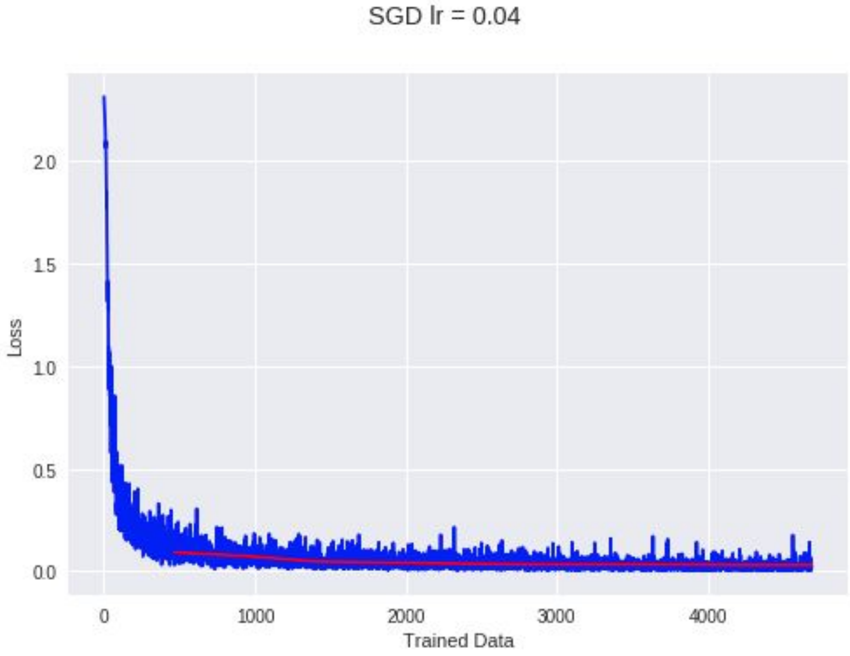
Francesc Cabrera
Hamza Errahmouni
David Molins
Guillem Viladrich

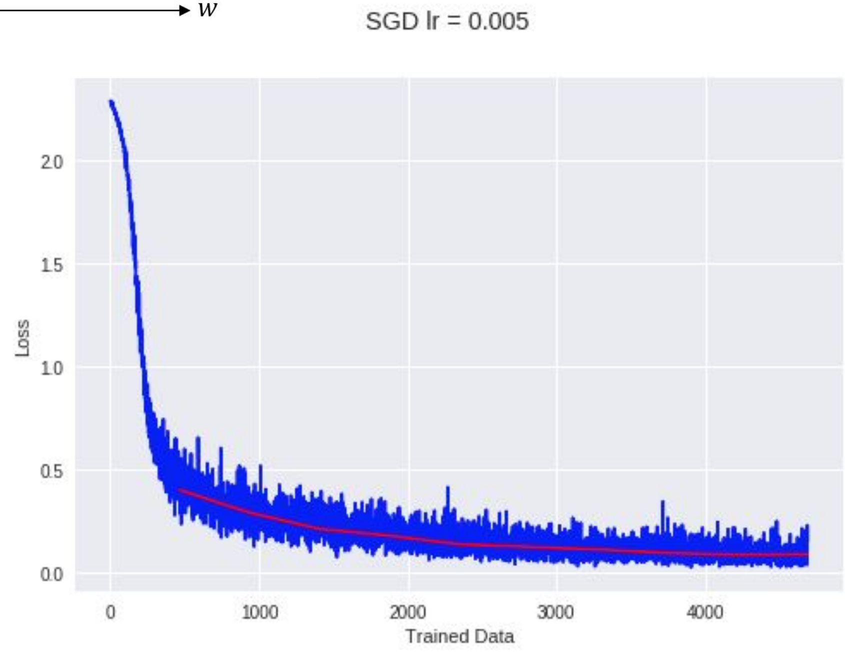
Stochastic Gradient Descent

Learning Rate

$$\theta_t = \theta_{t-1} - \underbrace{\alpha \nabla_\theta \mathcal{L}(\theta_{t-1})}_{\text{Evaluated on a mini-batch}}$$



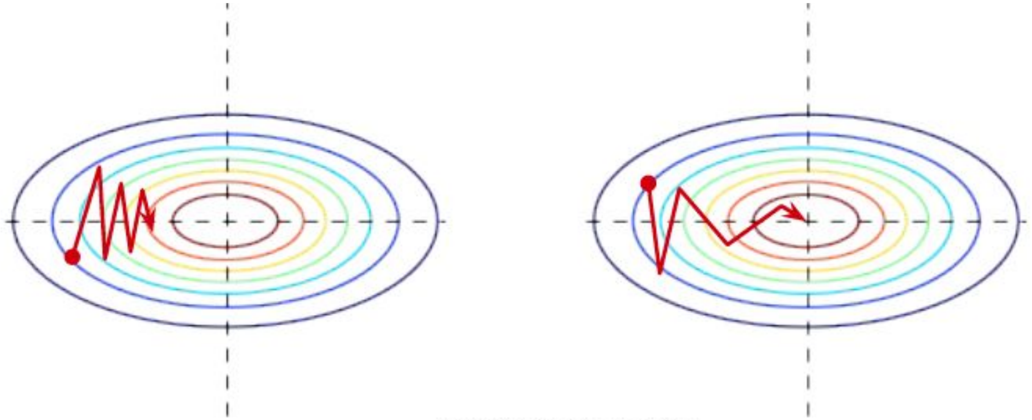




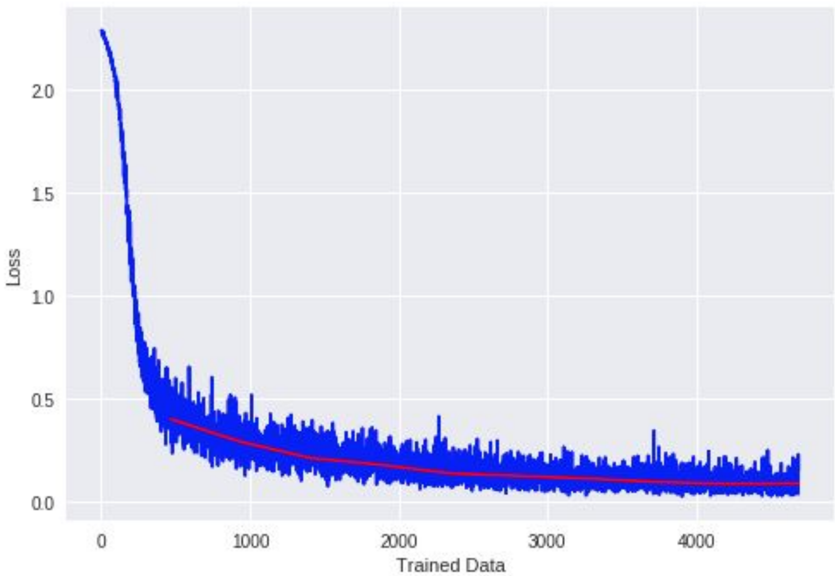
Stochastic Gradient Descent

Momentum

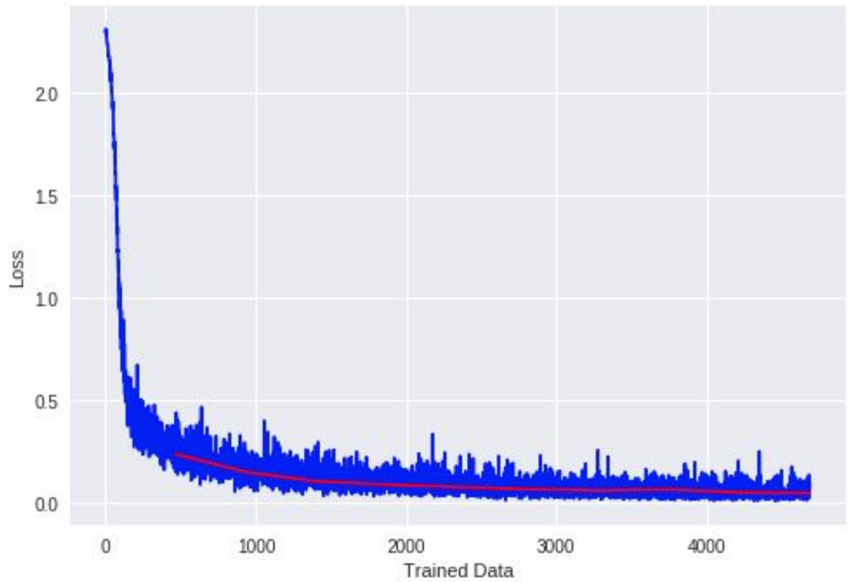
$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \mathcal{L}(\theta_{t-1})$$
$$\theta_t = \theta_{t-1} - v_t$$



SGD lr = 0.005



SGD lr = 0.005 m = 0.5



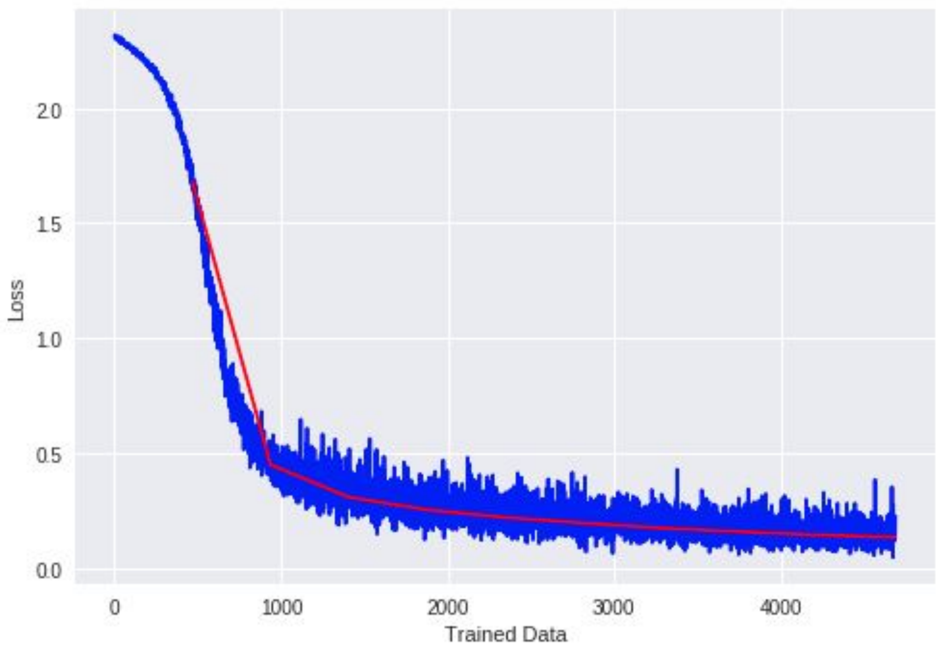
Stochastic Gradient Descent

Nesterov momentum

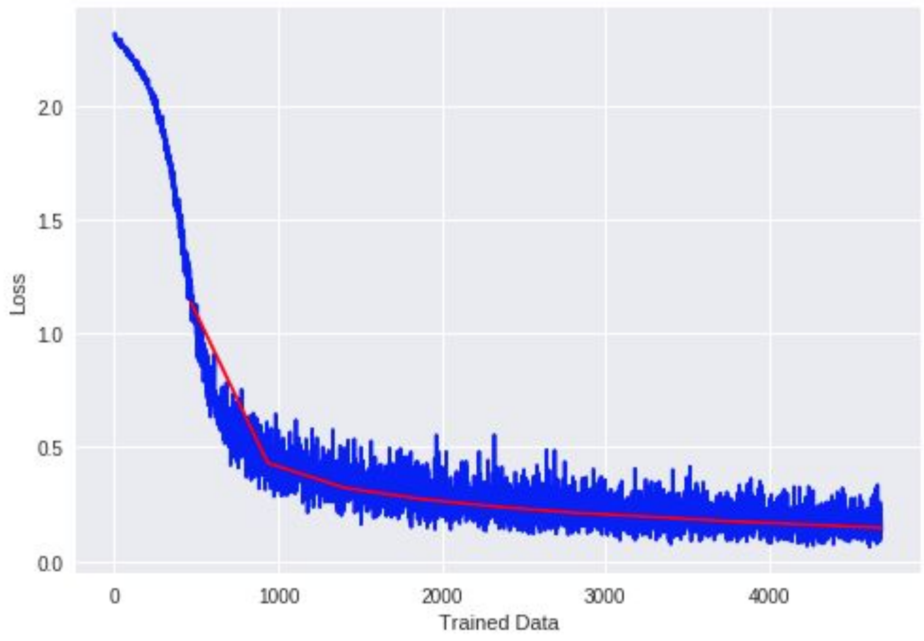
$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta \mathcal{L}(\theta_{t-1} - \gamma v_{t-1})$$

$$\theta_t = \theta_{t-1} - v_t$$

SGD lr = 0.001 m = 0.5 Nesterov = False



SGD lr = 0.001 m = 0.5 Nesterov = True



Adam

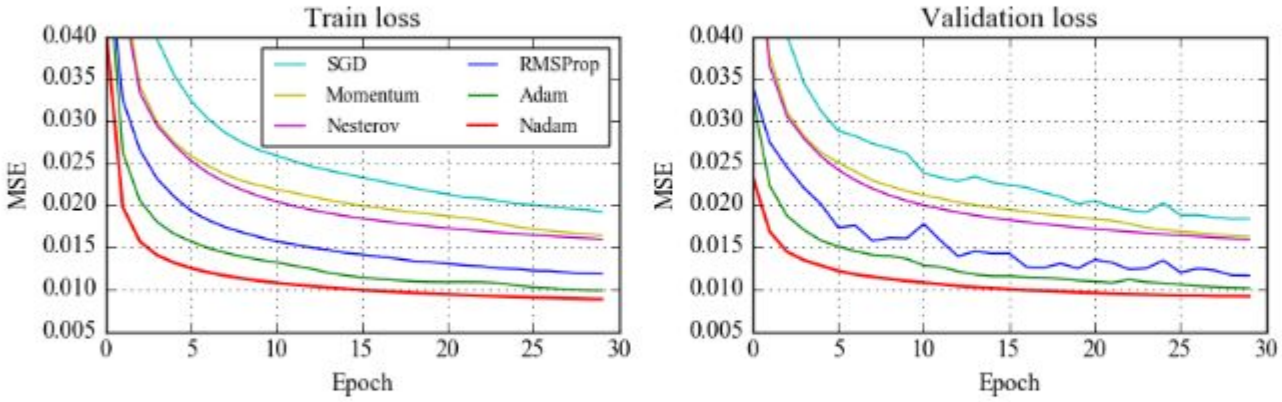
Actual step size taken by the Adam in each iteration is approximately bounded the step size hyper-parameter. This property add intuitive understanding to previous unintuitive learning rate hyper-parameter.

Step size of Adam update rule is invariant to the magnitude of the gradient, which helps a lot when going through areas with tiny gradients (such as saddle points or ravines). In these areas SGD struggles to quickly navigate through them.

However, after a while people started noticing that despite superior training time, Adam in some areas does not converge to an optimal solution.

From the Adam analysis, in general, you must know that:

- Adaptive Moment Estimation (Adam) is an algorithm that computes adaptive learning rates for each parameter.
- Also, it keeps an exponentially decaying average of past gradients similar to momentum.
- Each beta is for one of the 2 estimations that you make, they are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method



$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

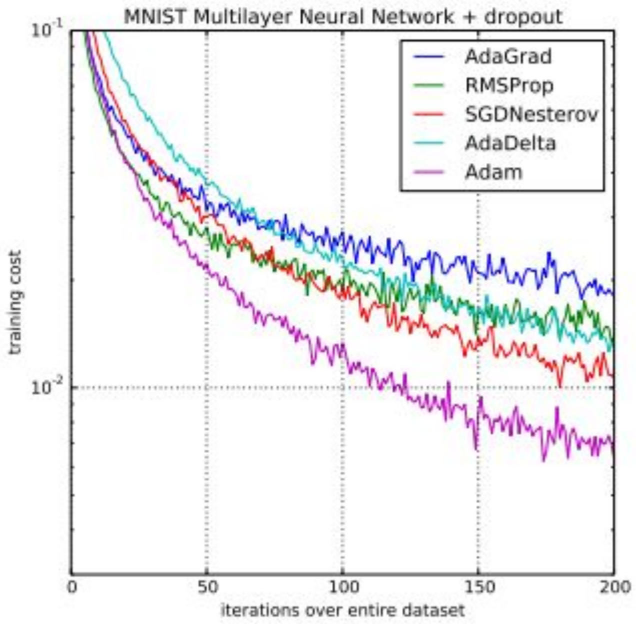
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t.$$

Algorithm 1. Adam Optimization Algorithm

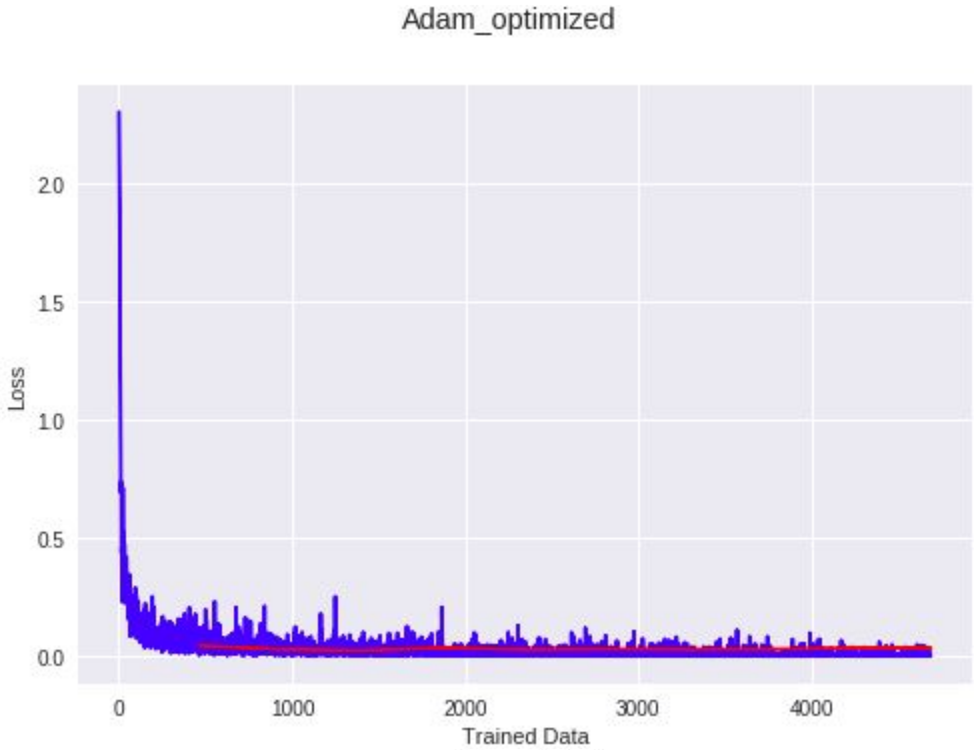
Require: Objective function $f(\theta)$, initial parameters θ_0, stepsize hyperparameter α, exponential decay rates β_1, β_2 for moment estimates, tolerance parameter $\lambda > 0$ for numerical stability, and decision rule for declaring convergence of θ_t in scheme.

procedure $\textsc{Adam}(f, \theta_0\ ;\ \alpha,\ \beta_1,\ \beta_2)$
- $m_0, v_0, t \leftarrow [0, 0, 0]$ # Initialize moment estimates
- # and timestep to zero
- # Begin optimization procedure
- **while** θ_t has not converged **do**
  - $t \leftarrow t+1$ # Update timestep
  - $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ # Compute gradient of objective
  - $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$ # Update first moment estimate
  - $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot (g_t \odot g_t)$ # Update second moment estimate
  - $\widehat{m}_t \leftarrow m_t/(1-\beta_1^t)$ # Create unbiased estimate $\widehat{m}_t$
  - $\widehat{v}_t \leftarrow v_t/(1-\beta_2^t)$ # Create unbiased estimate $\widehat{v}_t$
  - $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t/(\sqrt{\widehat{v}_t} + \lambda)$ # Update objective parameters
- **return** θ_t # Return final parameters

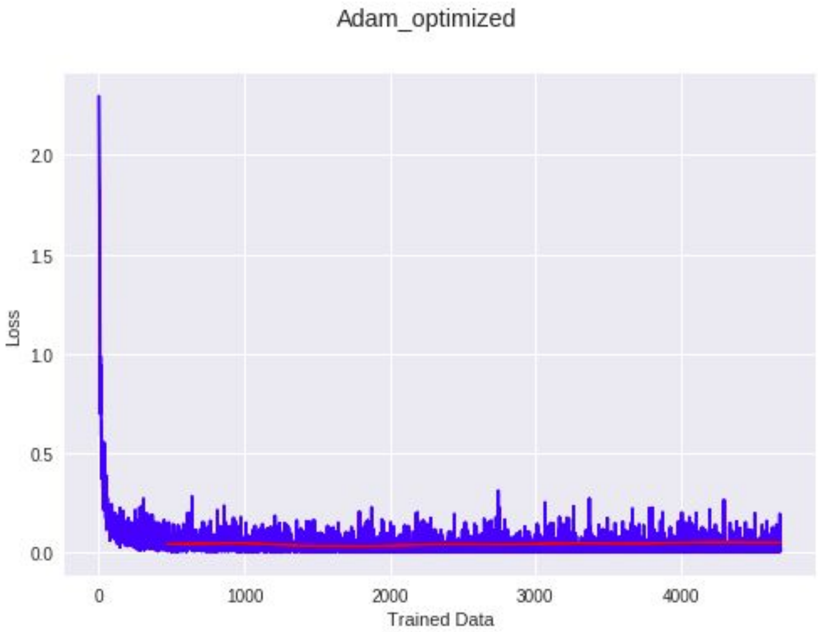


Deep Learning - Optimization

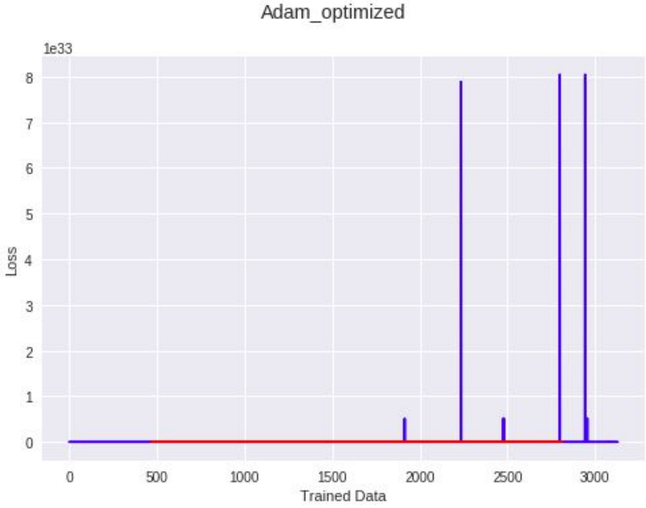
Effect on the betas



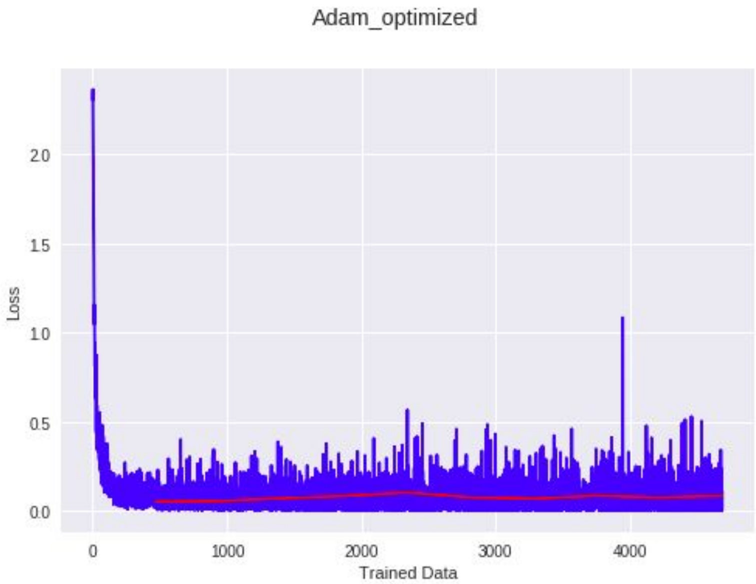
$\beta1 = 0.9$
$\beta2=0.99$
$lr < 0.001$



$\beta1 = 0.9$
$\beta2= 0.99$
$lr = 0.001$



$\beta1 > \beta2$



$\beta1 = 0.01$
$\beta2= 0.01$

Adadelta

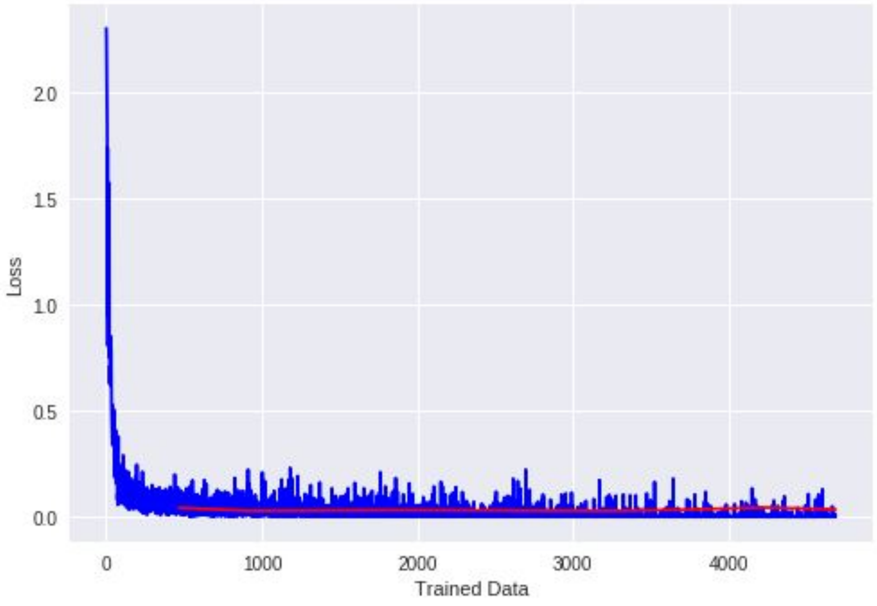
Adadelta is a more robust extension of Adagrad that adapts learning rates based on a moving window of gradient updates, instead of accumulating all past gradients. This way, Adadelta continues learning even when many updates have been done, the recommended initialization values for pytorch are the following:

- rho (*float*, *optional*) – coefficient used for computing a running average of squared gradients (default: 0.9)
- eps (*float*, *optional*) – term added to the denominator to improve numerical stability (default: 1e-6)
- lr (*float*, *optional*) – coefficient that scale delta before it is applied to the parameters (default: 1.0)
- weight_decay (*float*, *optional*) – weight decay (L2 penalty) (default: 0)

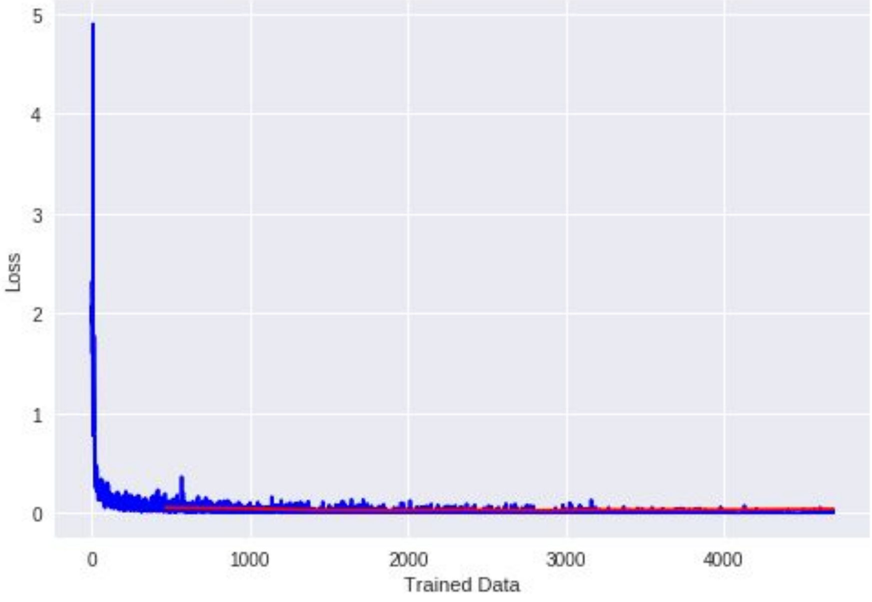
Adadelta

We tested Adadelta changing rho between 0.01 and 0.9 (recommendation):

Delta rho = 0.01



Delta rho = 0.9



As we can see there is not much difference between one case and the other, there's more loss when rho=0.01, but the accuracy is 99% in both cases. So we can assume Adadelta is so robust that can adapt its parameters properly.

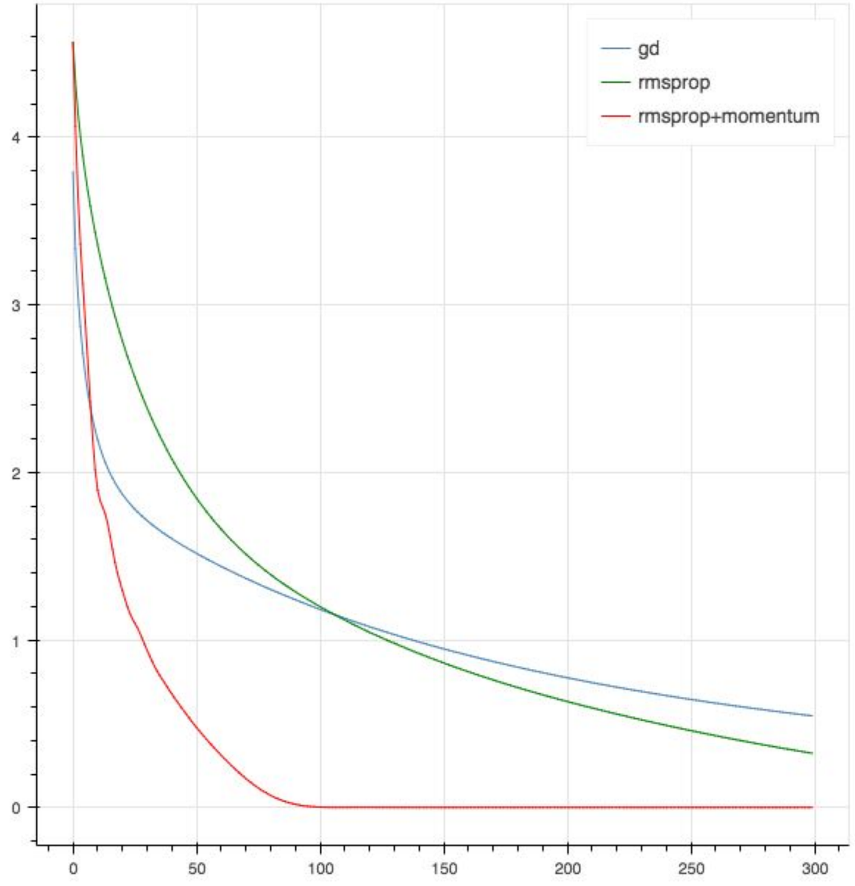
RMSprop

Modification of Adagrad to address aggressively decaying learning rate.

Instead of storing sum of squares of gradient over all time steps so far, use a **decayed moving average** of sum of squares of gradients

$$G_t = \gamma G_{t-1} + (1-\gamma)\mathrm{diag}(\nabla \mathcal{L}(\theta))^2$$

Update rule: $$\theta_t = \theta_{t-1} - \alpha G_t^{-\frac{1}{2}} \nabla \mathcal{L}(\theta_{t-1})$$



The RMSprop optimizer is similar to the gradient descent algorithm with momentum.

RMSprop

The RMSprop optimizer **restricts** the oscillations in the **vertical direction**. Therefore, we can increase our learning rate and our algorithm could take **larger steps** in the **horizontal direction** converging faster.

The difference between RMSprop and gradient descent is on how the gradients are calculated. The value of momentum is denoted by beta and is usually set to 0.9. A good default value for the learning rate is 0.001.

$$v_{dw} = \beta \cdot v_{dw} + (1-\beta) \cdot dw^2$$

$$v_{db} = \beta \cdot v_{dw} + (1-\beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

RMSprop optimizer

$$v_{dw} = \beta \cdot v_{dw} + (1-\beta) \cdot dw$$

$$v_{db} = \beta \cdot v_{dw} + (1-\beta) \cdot db$$

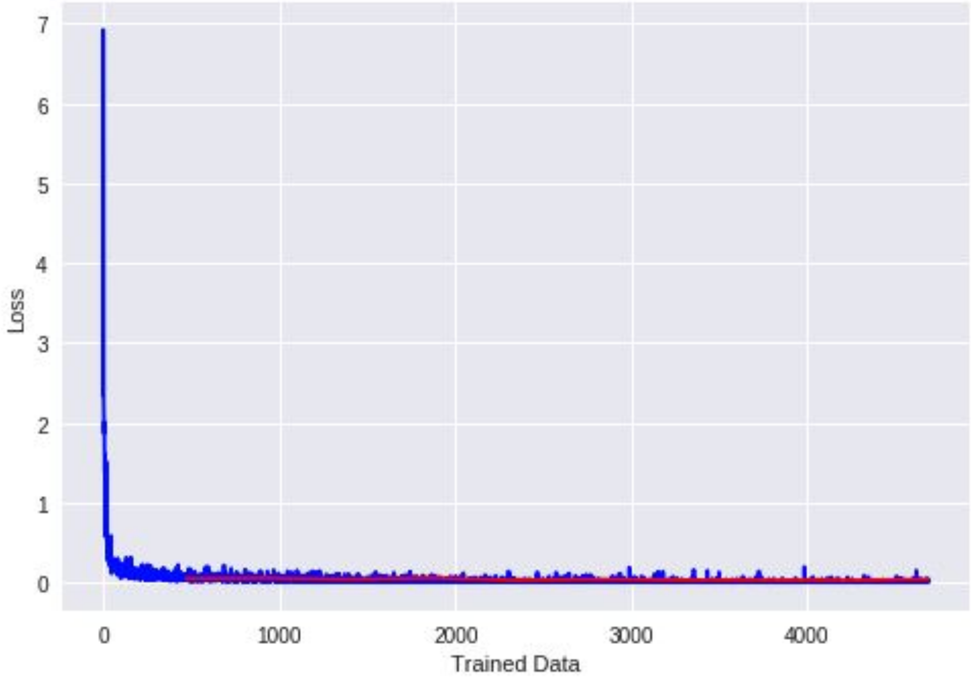
$$W = W - \alpha \cdot v_{dw}$$

$$b = b - \alpha \cdot v_{db}$$

Gradient descent with momenttum

RMSprop

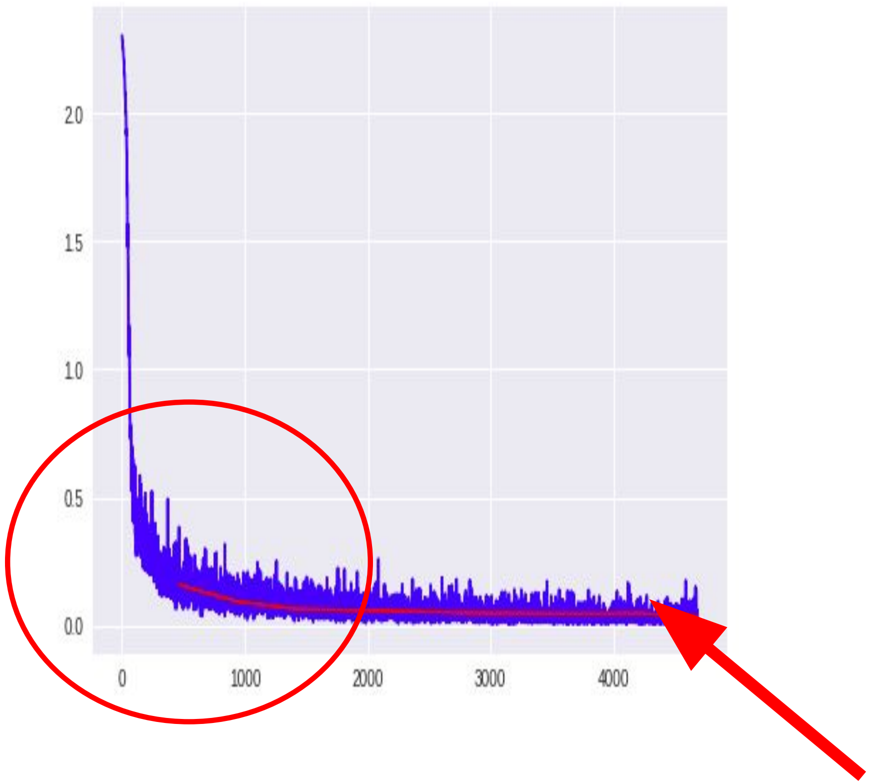
```
RMSprop (
Parameter Group 0
    alpha: 0.99
    centered: False
    eps: 1e-08
    lr: 0.001
    momentum: 0
    weight_decay: 0
)
```



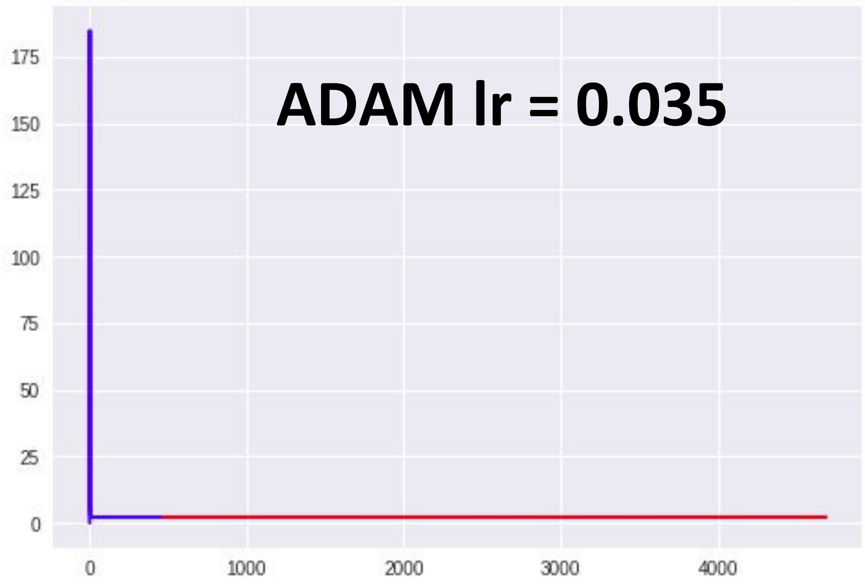

```
Training with opt: RMS -- Train epoch: 1 -- Validation Average loss: 0.0520, Accuracy: 9831/10000 (98%)
Training with opt: RMS -- Train epoch: 2 -- Validation Average loss: 0.0571, Accuracy: 9809/10000 (98%)
Training with opt: RMS -- Train epoch: 3 -- Validation Average loss: 0.0424, Accuracy: 9864/10000 (99%)
Training with opt: RMS -- Train epoch: 4 -- Validation Average loss: 0.0583, Accuracy: 9821/10000 (98%)
Training with opt: RMS -- Train epoch: 5 -- Validation Average loss: 0.0291, Accuracy: 9907/10000 (99%)
Training with opt: RMS -- Train epoch: 6 -- Validation Average loss: 0.0414, Accuracy: 9864/10000 (99%)
Training with opt: RMS -- Train epoch: 7 -- Validation Average loss: 0.0324, Accuracy: 9903/10000 (99%)
Training with opt: RMS -- Train epoch: 8 -- Validation Average loss: 0.0315, Accuracy: 9909/10000 (99%)
Training with opt: RMS -- Train epoch: 9 -- Validation Average loss: 0.0304, Accuracy: 9920/10000 (99%)
Training with opt: RMS -- Train epoch: 10 -- Validation Average loss: 0.0411, Accuracy: 9901/10000 (99%)
```

Deep Learning - Optimization

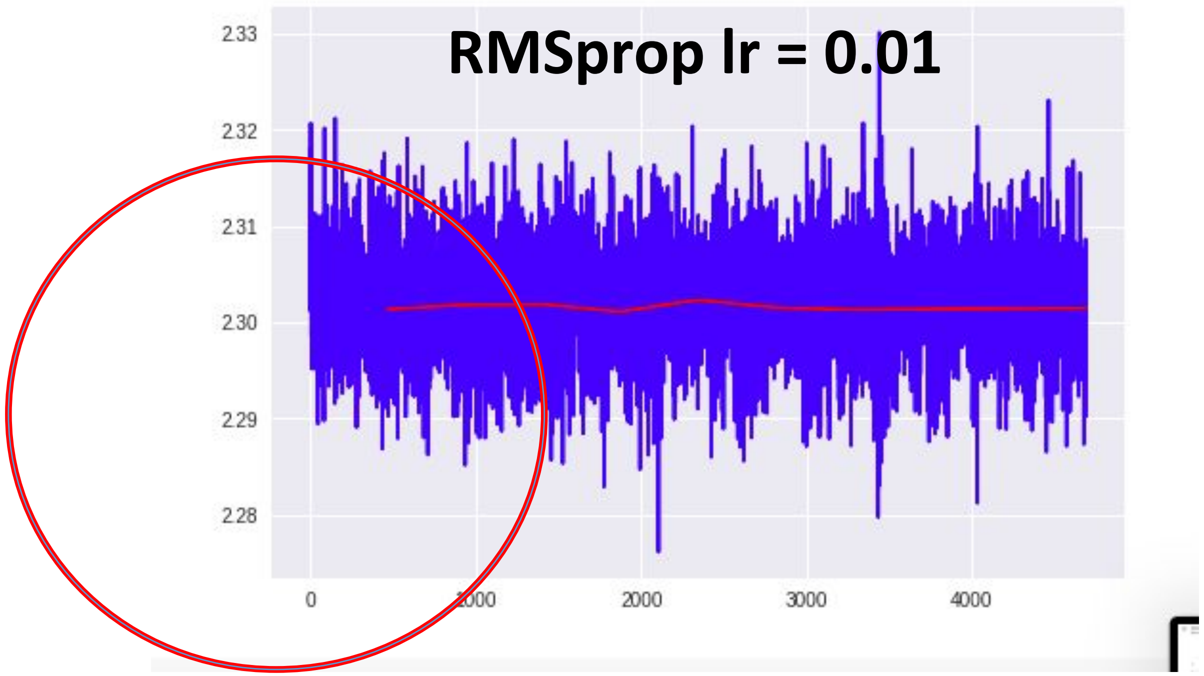
SGD lr = 0.01



ADAM lr = 0.035

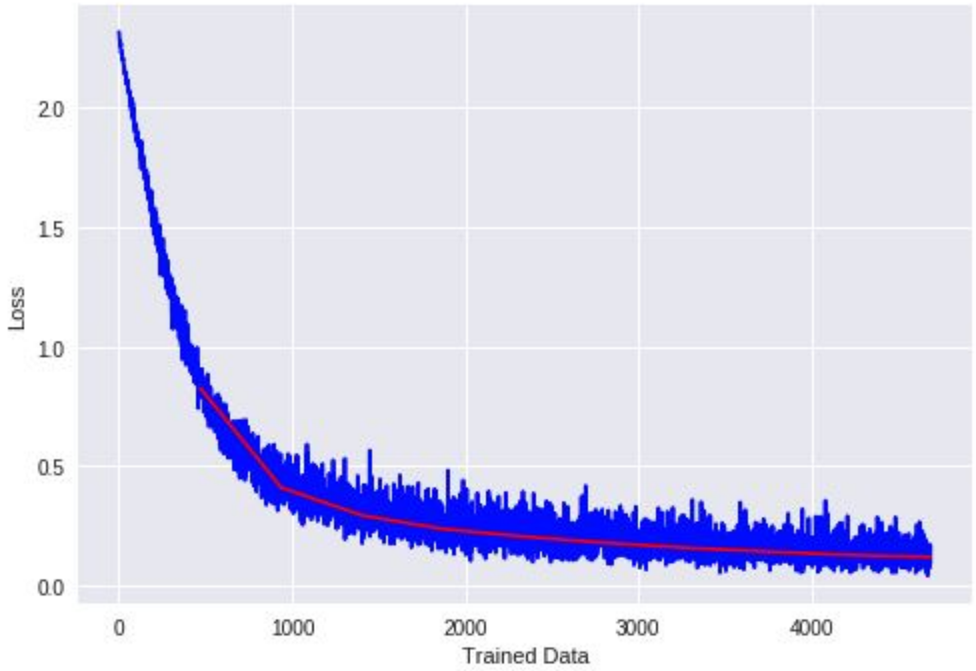


RMSprop lr = 0.01



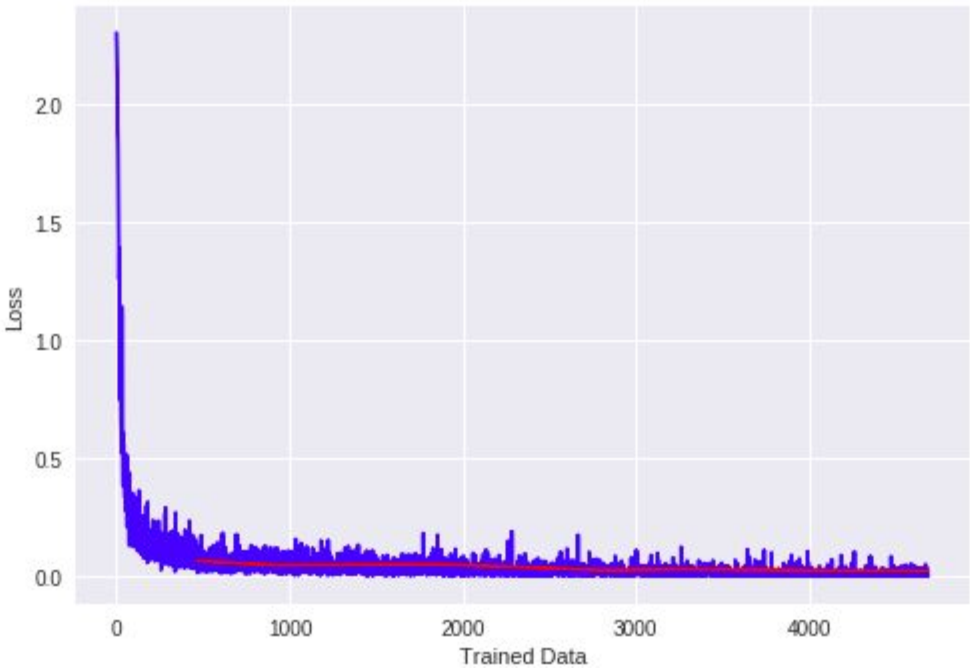
RMSprop

```
RMSprop (
Parameter Group 0
    alpha: 0.99
    centered: False
    eps: 1e-08
    lr: 1e-05
    momentum: 0
    weight_decay: 0
)
```

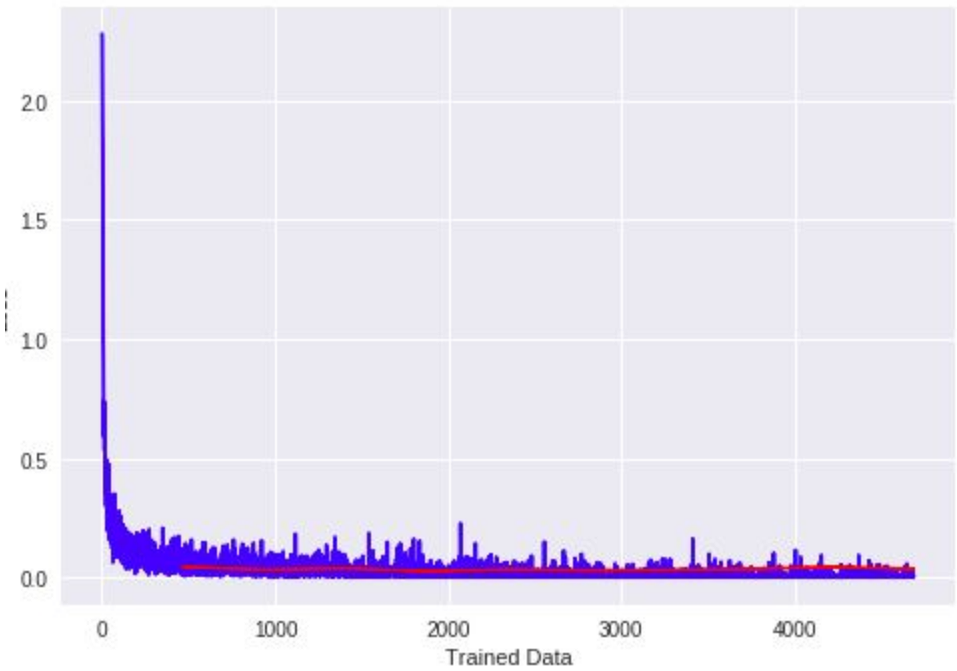



```
Training with opt: RMS -- Train epoch: 1 -- Validation Average loss: 0.8277, Accuracy: 8580/10000 (86%)
Training with opt: RMS -- Train epoch: 2 -- Validation Average loss: 0.4112, Accuracy: 9013/10000 (90%)
Training with opt: RMS -- Train epoch: 3 -- Validation Average loss: 0.2947, Accuracy: 9203/10000 (92%)
Training with opt: RMS -- Train epoch: 4 -- Validation Average loss: 0.2382, Accuracy: 9335/10000 (93%)
Training with opt: RMS -- Train epoch: 5 -- Validation Average loss: 0.2062, Accuracy: 9424/10000 (94%)
Training with opt: RMS -- Train epoch: 6 -- Validation Average loss: 0.1802, Accuracy: 9475/10000 (95%)
Training with opt: RMS -- Train epoch: 7 -- Validation Average loss: 0.1598, Accuracy: 9535/10000 (95%)
Training with opt: RMS -- Train epoch: 8 -- Validation Average loss: 0.1438, Accuracy: 9593/10000 (96%)
Training with opt: RMS -- Train epoch: 9 -- Validation Average loss: 0.1296, Accuracy: 9632/10000 (96%)
Training with opt: RMS -- Train epoch: 10 -- Validation Average loss: 0.1217, Accuracy: 9665/10000 (97%)
```

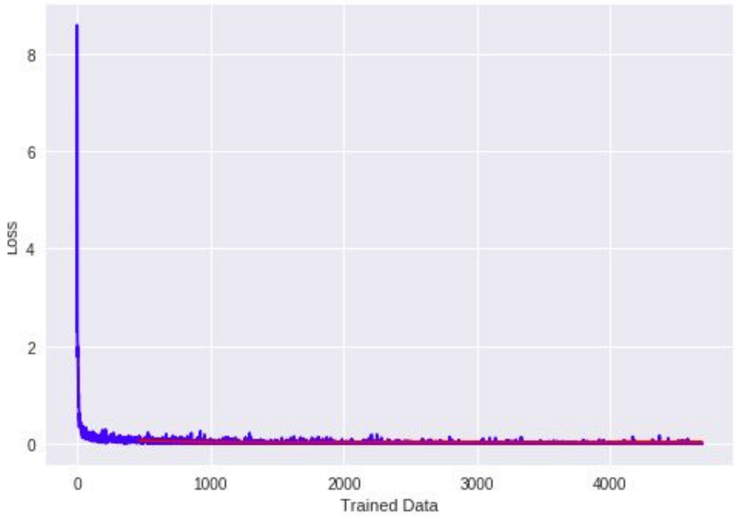
SGD lr = 0.04 m = 0.5 Nesterov = True



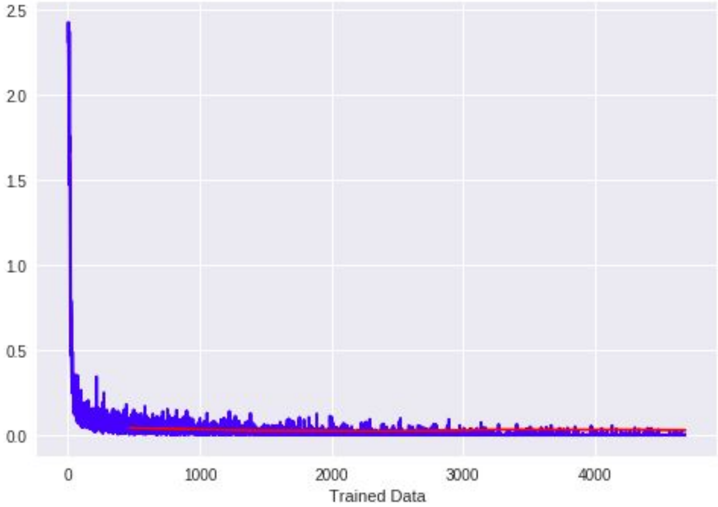
Adam lr = 0.001 Betas = (0.9, 0.99)



RMSprop lr = 0.001



Adadelta rho = 0.95



Analysis:

We'll observe how the optimizers work on the accuracy, convergence and its speed.

These factors vary a lot depending on the learning rate if we talk about SGD, Adam and RMSprop, so we'll analyse them depending on the value of this hyper parameter that we have.

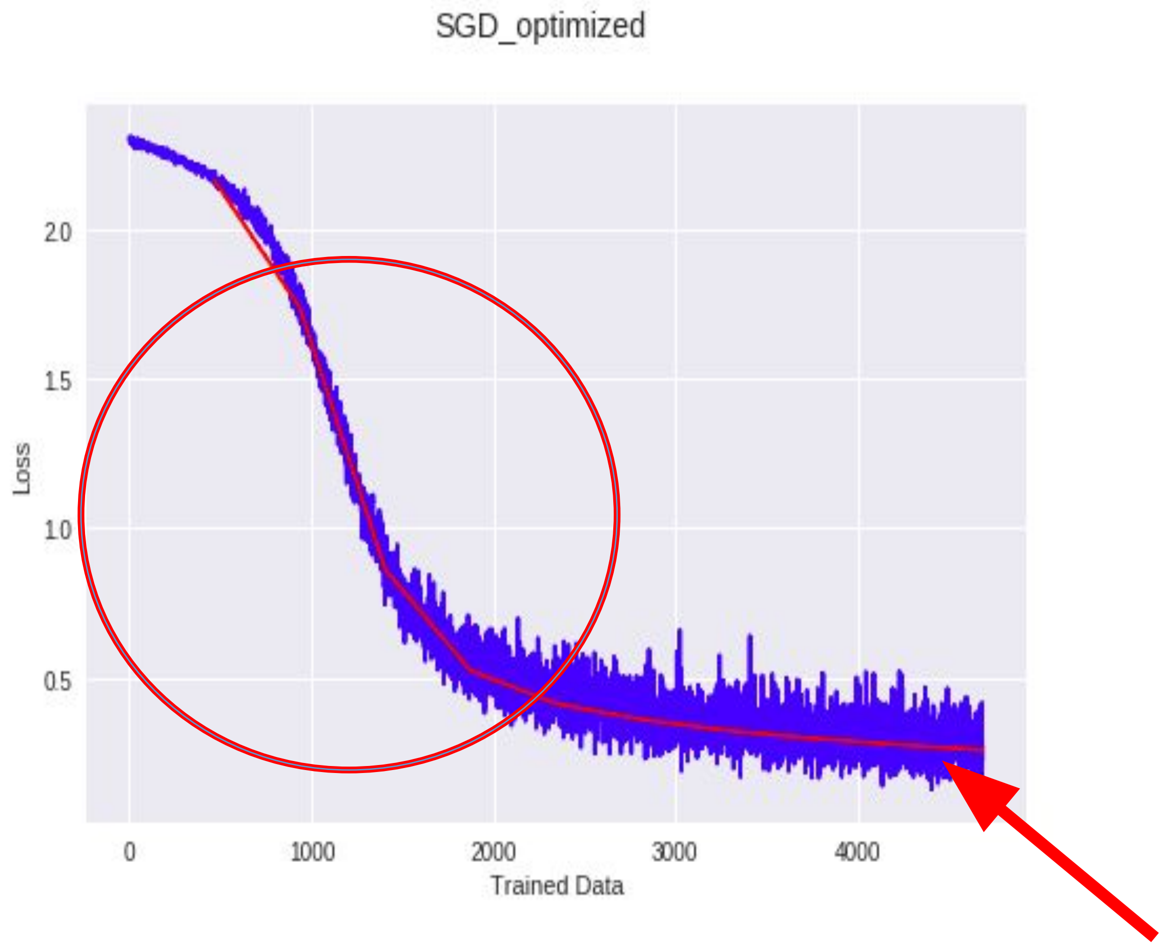
At the same time, the effect on the betas on Adam vary a lot so we'll analyze the general case scenarios

Adadelta depends specifically on the parameter rho.

Deep Learning - Optimization

```
SGD (
Parameter Group 0
    dampening: 0
    lr: 0.0005
    momentum: 0.5
    nesterov: False
    weight_decay: 0
)

Validation set: Average loss: 2.1605, Accuracy: 5517/10000 (55%)

Validation set: Average loss: 1.7345, Accuracy: 7321/10000 (73%)

Validation set: Average loss: 0.8599, Accuracy: 8362/10000 (84%)

Validation set: Average loss: 0.5235, Accuracy: 8700/10000 (87%)

Validation set: Average loss: 0.4134, Accuracy: 8892/10000 (89%)

Validation set: Average loss: 0.3605, Accuracy: 8977/10000 (90%)

Validation set: Average loss: 0.3239, Accuracy: 9069/10000 (91%)

Validation set: Average loss: 0.2977, Accuracy: 9154/10000 (92%)

Validation set: Average loss: 0.2765, Accuracy: 9188/10000 (92%)

Validation set: Average loss: 0.2582, Accuracy: 9240/10000 (92%)
```

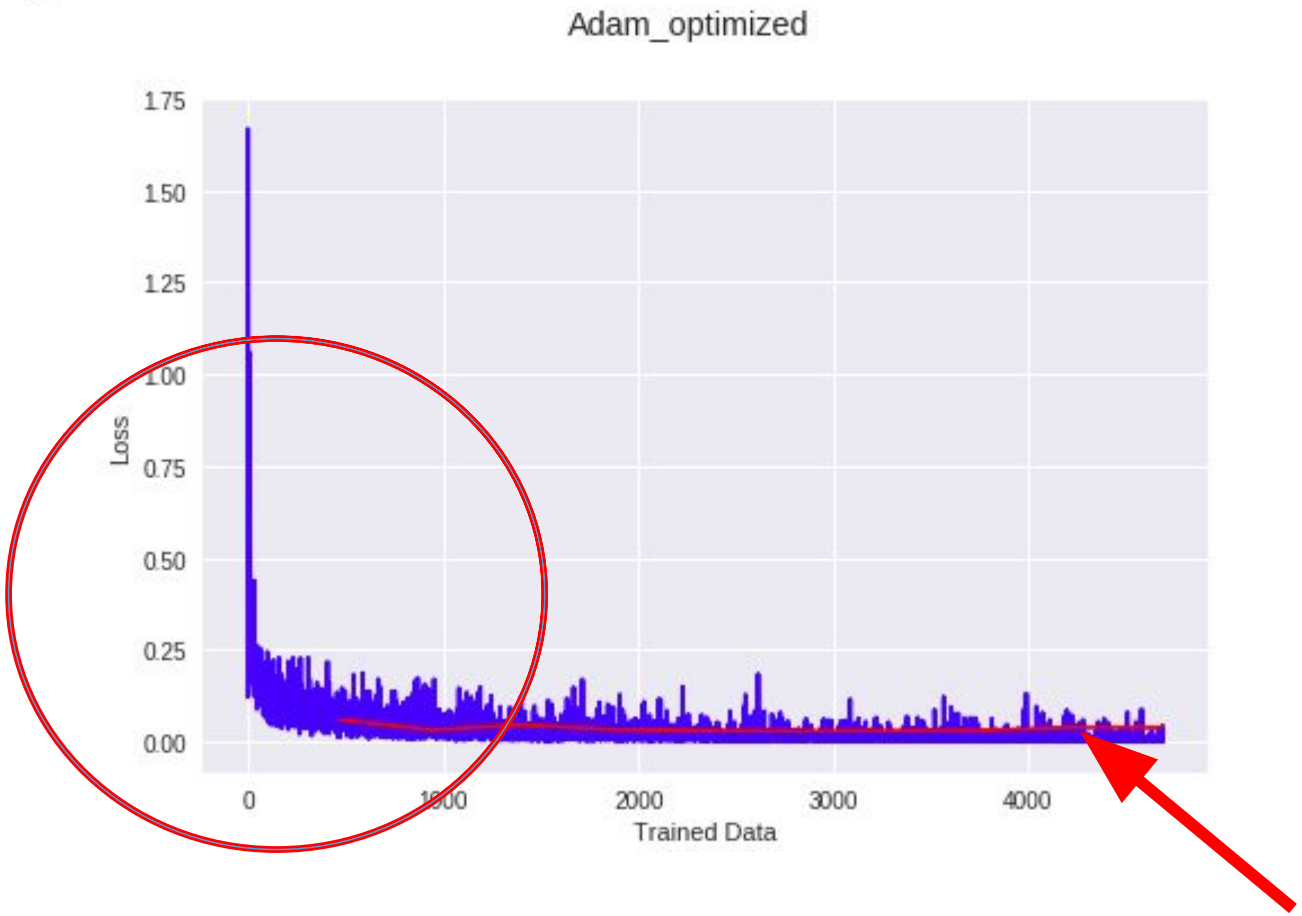
SGD lr = 0.0005



Deep Learning - Optimization

```
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    eps: 1e-08
    lr: 0.0005
    weight_decay: 0
)

Validation set: Average loss: 0.0604, Accuracy: 9809/10000 (98%)

Validation set: Average loss: 0.0331, Accuracy: 9891/10000 (99%)

Validation set: Average loss: 0.0460, Accuracy: 9850/10000 (98%)

Validation set: Average loss: 0.0340, Accuracy: 9880/10000 (99%)

Validation set: Average loss: 0.0308, Accuracy: 9897/10000 (99%)

Validation set: Average loss: 0.0292, Accuracy: 9907/10000 (99%)

Validation set: Average loss: 0.0303, Accuracy: 9908/10000 (99%)

Validation set: Average loss: 0.0310, Accuracy: 9914/10000 (99%)

Validation set: Average loss: 0.0396, Accuracy: 9889/10000 (99%)

Validation set: Average loss: 0.0393, Accuracy: 9900/10000 (99%)
```

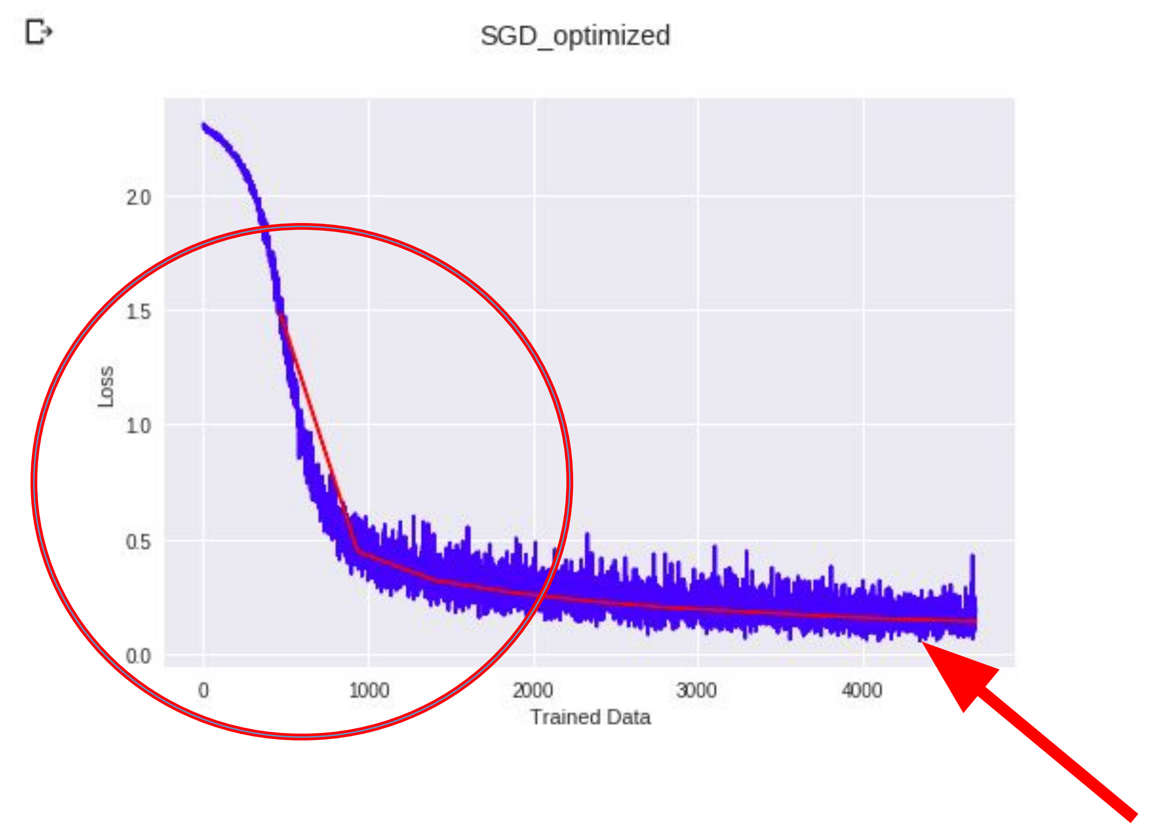
ADAM lr = 0.0005



Deep Learning - Optimization

SGD lr = 0.001

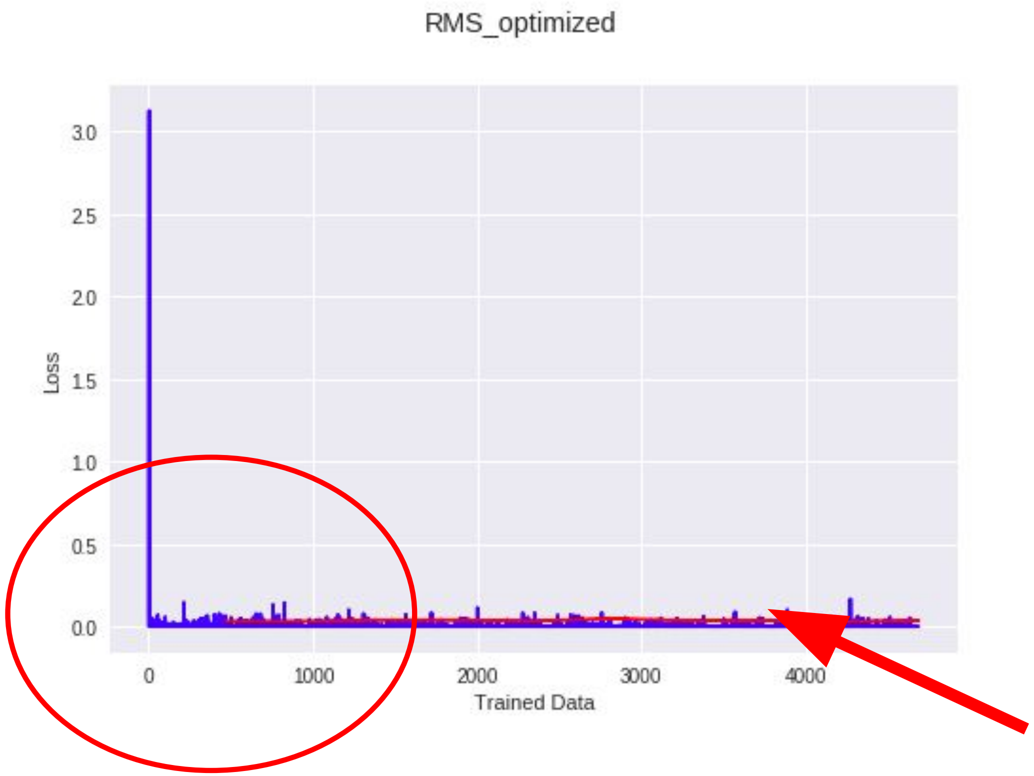
```
[10] SGD (
     Parameter Group 0
         dampening: 0
         lr: 0.001
         momentum: 0.5
         nesterov: False
         weight_decay: 0
     )

     Validation set: Average loss: 1.4816, Accuracy: 7719/10000 (77%)

     Validation set: Average loss: 0.4470, Accuracy: 8855/10000 (89%)

     Validation set: Average loss: 0.3211, Accuracy: 9098/10000 (91%)

     Validation set: Average loss: 0.2686, Accuracy: 9231/10000 (92%)

     Validation set: Average loss: 0.2310, Accuracy: 9320/10000 (93%)

     Validation set: Average loss: 0.2034, Accuracy: 9430/10000 (94%)

     Validation set: Average loss: 0.1857, Accuracy: 9481/10000 (95%)

     Validation set: Average loss: 0.1670, Accuracy: 9524/10000 (95%)

     Validation set: Average loss: 0.1527, Accuracy: 9562/10000 (96%)

     Validation set: Average loss: 0.1426, Accuracy: 9613/10000 (96%)
```



Deep Learning - Optimization

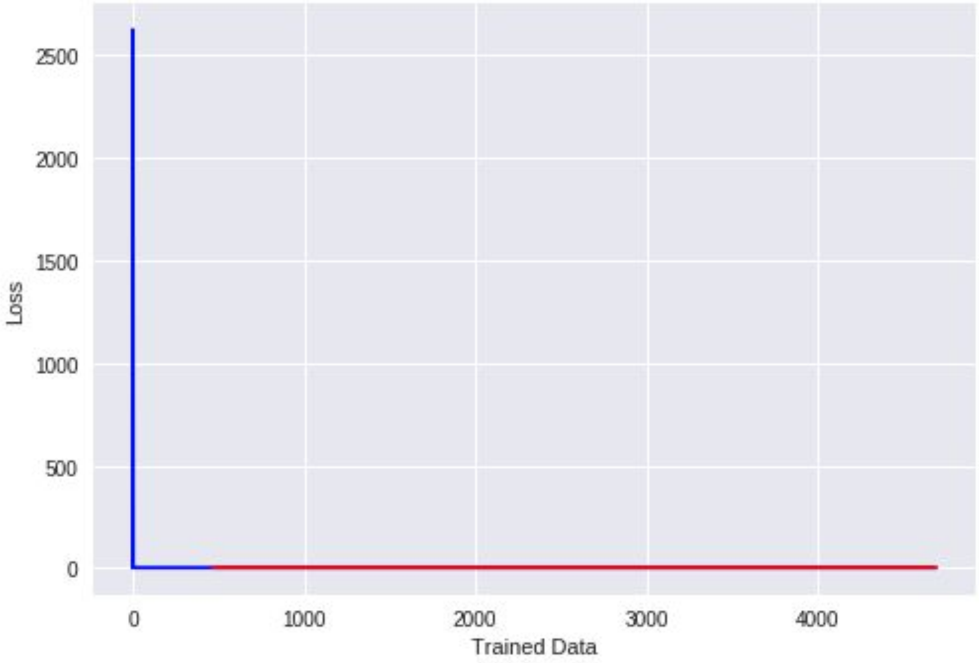
RMSprop lr = 0.001

```
RMSprop (
Parameter Group 0
    alpha: 0.99
    centered: False
    eps: 1e-08
    lr: 0.001
    momentum: 0
    weight_decay: 0
)

Validation set: Average loss: 0.0538, Accuracy: 9868/10000 (99%)

Validation set: Average loss: 0.0702, Accuracy: 9832/10000 (98%)

Validation set: Average loss: 0.0437, Accuracy: 9915/10000 (99%)

Validation set: Average loss: 0.0293, Accuracy: 9934/10000 (99%)

Validation set: Average loss: 0.0354, Accuracy: 9924/10000 (99%)

Validation set: Average loss: 0.0454, Accuracy: 9906/10000 (99%)

Validation set: Average loss: 0.0294, Accuracy: 9937/10000 (99%)

Validation set: Average loss: 0.0357, Accuracy: 9938/10000 (99%)

Validation set: Average loss: 0.0589, Accuracy: 9893/10000 (99%)

Validation set: Average loss: 0.0461, Accuracy: 9923/10000 (99%)
```



RMSprop

```
RMSprop (
Parameter Group 0
    alpha: 0.99
    centered: False
    eps: 1e-08
    lr: 0.01
    momentum: 0
    weight_decay: 0
)
```

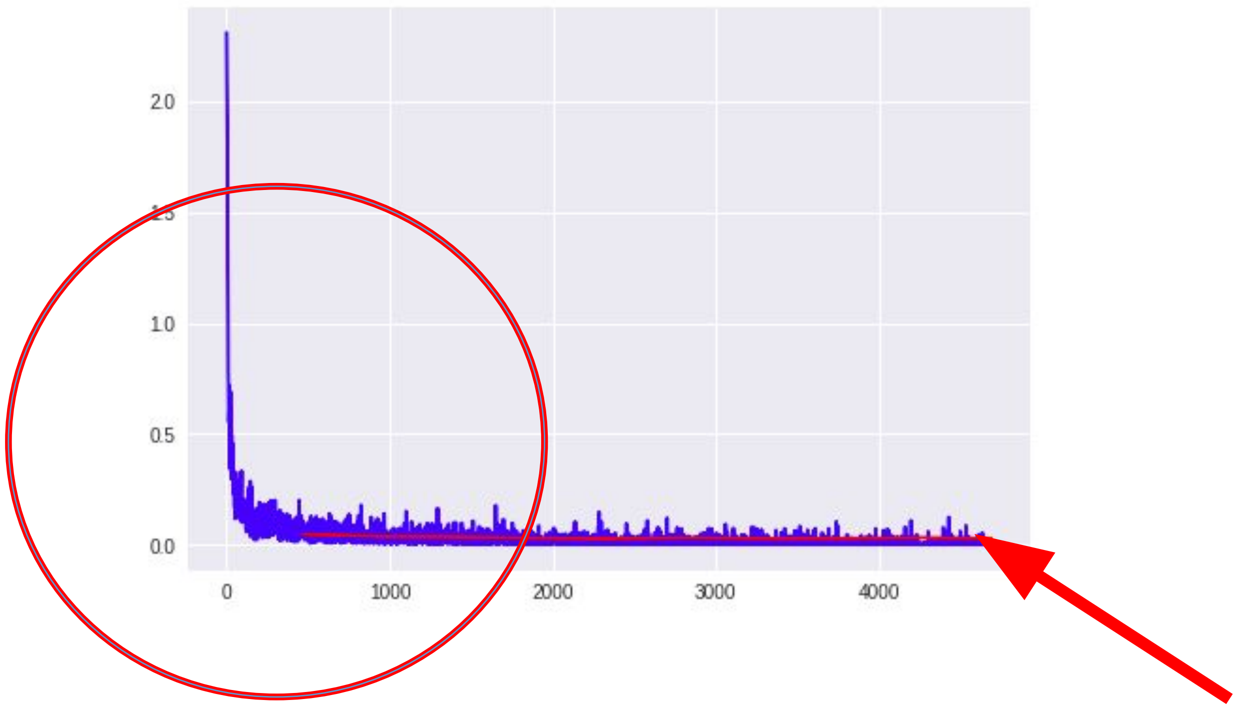



```
Training with opt: RMS -- Train epoch: 1 -- Validation Average loss: 2.3016, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 2 -- Validation Average loss: 2.3026, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 3 -- Validation Average loss: 2.3016, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 4 -- Validation Average loss: 2.3011, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 5 -- Validation Average loss: 2.3015, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 6 -- Validation Average loss: 2.3017, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 7 -- Validation Average loss: 2.3015, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 8 -- Validation Average loss: 2.3016, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 9 -- Validation Average loss: 2.3015, Accuracy: 1135/10000 (11%)
Training with opt: RMS -- Train epoch: 10 -- Validation Average loss: 2.3015, Accuracy: 1135/10000 (11%)
```

Deep Learning - Optimization

```
Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    eps: 1e-08
    lr: 0.001
    weight_decay: 0
)

Validation set: Average loss: 0.0478, Accuracy: 9854/10000 (99%)

Validation set: Average loss: 0.0376, Accuracy: 9863/10000 (99%)

Validation set: Average loss: 0.0357, Accuracy: 9886/10000 (99%)

Validation set: Average loss: 0.0309, Accuracy: 9898/10000 (99%)

Validation set: Average loss: 0.0275, Accuracy: 9917/10000 (99%)

Validation set: Average loss: 0.0328, Accuracy: 9897/10000 (99%)

Validation set: Average loss: 0.0289, Accuracy: 9914/10000 (99%)

Validation set: Average loss: 0.0290, Accuracy: 9901/10000 (99%)

Validation set: Average loss: 0.0325, Accuracy: 9900/10000 (99%)

Validation set: Average loss: 0.0286, Accuracy: 9927/10000 (99%)
```

ADAM lr = 0.001



ADAM lr = 0.00025

```
[18] Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    eps: 1e-08
    lr: 0.00025
    weight_decay: 0
)

Validation set: Average loss: 0.0639, Accuracy: 9807/10000 (98%)

Validation set: Average loss: 0.0472, Accuracy: 9851/10000 (99%)

Validation set: Average loss: 0.0383, Accuracy: 9875/10000 (99%)

Validation set: Average loss: 0.0323, Accuracy: 9898/10000 (99%)

Validation set: Average loss: 0.0333, Accuracy: 9895/10000 (99%)

Validation set: Average loss: 0.0420, Accuracy: 9863/10000 (99%)

Validation set: Average loss: 0.0313, Accuracy: 9898/10000 (99%)

Validation set: Average loss: 0.0285, Accuracy: 9916/10000 (99%)

Validation set: Average loss: 0.0325, Accuracy: 9908/10000 (99%)

Validation set: Average loss: 0.0293, Accuracy: 9905/10000 (99%)
```

RMSprop lr = 0.00025

```
RMSprop (
Parameter Group 0
    alpha: 0.99
    centered: False
    eps: 1e-08
    lr: 0.00025
    momentum: 0
    weight_decay: 0
)

Validation set: Average loss: 0.0295, Accuracy: 9911/10000 (99%)

Validation set: Average loss: 0.0285, Accuracy: 9917/10000 (99%)

Validation set: Average loss: 0.0312, Accuracy: 9911/10000 (99%)

Validation set: Average loss: 0.0316, Accuracy: 9914/10000 (99%)

Validation set: Average loss: 0.0311, Accuracy: 9917/10000 (99%)

Validation set: Average loss: 0.0415, Accuracy: 9898/10000 (99%)

Validation set: Average loss: 0.0414, Accuracy: 9890/10000 (99%)

Validation set: Average loss: 0.0409, Accuracy: 9900/10000 (99%)

Validation set: Average loss: 0.0379, Accuracy: 9917/10000 (99%)

Validation set: Average loss: 0.0344, Accuracy: 9928/10000 (99%)
```

Conclusions