Deep Learning for Computer Vision: Practical Training

Shai Bagon
My Neural Net

SGD tricks
Regularization
BatchNorm
Initialization

...
Agenda

• SGD
  • Momentum, Adam, LR policies, initialization

• Regularization
  • DropOut
  • Weight Decay
  • Augmentation
  • Early stopping

• Batch normalization

• Transfer learning
Stochastic Gradient Descent (SGD)

\[ \mathcal{L}(\theta; \{(x_i, y_i)\}_{i \in 1..N}) \approx \frac{1}{|B|} \sum_{i \in \{B\}} \mathcal{L}(f(x_i; \theta), y_i) \]

Update rule:

\[ \theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t; \{(x_i, y_i)\}_{i \in \{B\}}) \]
Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$
Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$

High learning rate
Vanilla SGD

\[ \theta^{t+1} = \theta^t - \eta \nabla L(\theta^t) \]
Vanilla SGD

\[ \theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t) \]
Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla L(\theta^t)$$
SGD with Momentum

\[ \theta^{t+1} = \theta^t + \mu \nu^t - \eta \nabla L(\theta^t) \]
\[ \nu^{t+1} = \mu \nu^t - \eta \nabla L(\theta^t) \]
SGD with Momentum

\[
\theta^{t+1} = \theta^t + \mu \nu^t - \eta \nabla L(\theta^t) \\
\nu^{t+1} = \mu \nu^t - \eta \nabla L(\theta^t)
\]
Adam

\[ v^{t+1} = \beta_1 v^t + (1 - \beta_1) \nabla L(\theta^t) / (1 - (\beta_1)^t) \]

\[ m^{t+1} = \beta_2 m^t + (1 - \beta_2) \nabla L(\theta^t)^2 / (1 - (\beta_2)^t) \]

\[ \theta^{t+1} = \theta^t - \eta v^{t+1} / (\sqrt{m^{t+1}} + \epsilon) \]

1st moment (like SGD)

2nd moment ("variance")

Unbiased estimation

Weight the change by the variance
Adam

$$\theta^{t+1} = \theta^{t} - \eta \cdot \nu^{t+1} / \left( \sqrt{m^{t+1}} + \varepsilon \right)$$
Other SGD Update Rules

A good review can be found [here](#).
Do we want fast convergence?

\[ \mathcal{L}(\theta) \]

Train loss

Test loss
Vanilla SGD

$$\theta^{t+1} = \theta^t - \eta \nabla \mathcal{L}(\theta^t)$$

High learning rate
Learning Rate Decay

\[ \theta^{t+1} = \theta^t - \eta_t \nabla \mathcal{L}(\theta^t) \]
Learning Rate Decay

\[
\eta_t
\]

\[
\eta_t
\]

\[
\eta_t
\]
SGD - Remarks

• Guarantees?

No!
$y = \sigma(W_2 \cdot \sigma(W_1 \cdot x))$

$y = \sigma(W_2 P^T \cdot \sigma(PW_1 \cdot x))$
SGD – Starting point

What if we init $\theta = 0$?

$$y = W^T x$$

$$\frac{\partial y}{\partial x} = W$$

$$\frac{\partial y}{\partial x} = x^T$$
SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

$w_{ij} \sim N(0, 0.02^2)$
SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

\[ w_{ij} \sim N(0, 0.01^2) \]
SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

\[ y = W^T x \]

\[ \text{var}(y_i) = \text{var}(W_i^T x) \]

\[ \text{var}(w) = \frac{1}{D_{in}} \rightarrow \sigma = \sqrt{\frac{1}{D_{in}}} \]
SGD – Starting point

MLP, 6 layers, hidden dim=4096, no activation

\[ w_{ij} \sim N \left( 0, \frac{1}{D_{in}} \right) \]
SGD – Starting point

MLP, 6 layers, hidden dim=4096, ReLU activation

\[ w_{ij} \sim N \left( 0, \frac{1}{D_{in}} \right) \]
SGD – Starting point

MLP, 6 layers, hidden dim=4096, ReLU activation

\[ w_{ij} \sim N \left( 0, \frac{2}{D_{in}} \right) \]
SGD – Starting point

- Where we start the SGD is crucial
- Init depends on the architecture
- Xavier/Kaiming can be easily extended to Conv layers
- Inputs should be normalized to $\sim N(0, 1)$
Agenda

• SGD
  • Momentum, Adam, LR policies, initialization

• Regularization
  • DropOut
  • Weight Decay
  • Augmentation
  • Early stopping

• Batch normalization

• Transfer learning
Optimization’s Pitfalls

“Space” of all predictors

All deep networks

Networks of specific architecture

Approximation error

Estimation/Generalization error

Optimization error

𝜃∗optimization

𝜃∗training

𝜃∗world

“ideal” predictor

DL4CV @ Weizmann
Optimization’s Pitfalls

Networks of specific architecture

- Optimization error
- Estimation/Generalization error
- Approximation error

- Change Architecture
- Add more examples...
- Tweak optimization meta parameters

"ideal" predictor

\[ \theta^*_\text{world} \]

\[ \theta^*_\text{optimization} \]

\[ \theta^*_\text{training} \]
Optimization’s Pitfalls

• How to look at the loss-vs-iterations for train/test set?

• Overfitting
Regularization

\[ \mathcal{L}(\theta; X) = CE(\theta; X) + \lambda \|\theta\|_p \]
Regularization

- Data augmentation

Dog

Torchvision: transforms
Albumenations: https://github.com/albumentations-team/albumentations
Regularization

Dropout
Regularization

Early Stopping

\[ L \]

-- Train loss
-- Test loss
Optimization’s Pitfalls

Exploding gradients

\[ L \]

-- Train loss
-- Test loss

$t$
Optimization’s Pitfalls

Vanishing gradients
Optimization pitfalls

Too small learning rate

\[ \mathcal{L} \] -- Train loss
--- Test loss
Agenda

• SGD
  • Momentum, Adam, LR policies, initialization

• Regularization
  • DropOut
  • Weight Decay
  • Augmentation
  • Early stopping

• Batch normalization

• Transfer learning
Optimization’s Pitfalls

- Covariate shift

\[ x \xrightarrow{\partial L \over \partial \theta} y \]
Optimization’s Pitfalls

• Batch Norm!

\[ x \xrightarrow{\text{BN}} y \]
Agenda

• SGD
  • Momentum, Adam, LR policies, initialization

• Regularization
  • DropOut
  • Weight Decay
  • Augmentation
  • Early stopping

• Batch normalization

• Transfer learning
Transfer learning

\[ \mathbb{R}^{1000} \]
Transfer learning

\[ x \in \mathbb{R}^D \]

\[ W \in \mathbb{R}^{D \times 1000} \]

\[ \mathcal{L} \]
Transfer learning

$\mathbf{x} \in \mathbb{R}^D$

$W \in \mathbb{R}^{D \times L}$

Fixed

Softmax + CE loss

New Linear

Conv2D

ReLU

Max Pooling
Transfer learning

• Train only the last “task specific” layer
• Train SVM on deep features of new data
• Train several layers close to classification
• Train all layers

“Fine tuning”

```
In [1]: import torchvision
In [2]: model = torchvision.models.alexnet(pretrained=True)
In [3]: 
```
Agenda

• SGD
  • Momentum, Adam, LR policies, initialization
• Regularization
  • DropOut
  • Weight Decay
  • Augmentation
  • Early stopping
• Batch normalization
• Transfer learning
What’s coming up...

• Tutorial – “landmark” deep architectures
• Lecture (next week) – object detection and segmentation