Improving optimization

Training Deep Networks from Zero to Hero: avoiding pitfalls and going beyond

Gabriel Cavallari
gabriel.cavallari@usp.br

SIBGRAPI 2021

Improving optimization

1. Algorithm (optimizer)
2. Learning rate (step size)
3. Batch size
4. Normalization
5. Regularization
6. Data augmentation

1 - Optimizer

- The original Gradient Descent computes the gradient at one iteration using all training data

- Stochastic Gradient Descent (SGD) is an approximation that allows calculating the gradient of the cost function based on a random example or a small subset of examples (mini-batch)

- The regular SGD is a conservative but fair choice

1 - Optimizer

- SGD is less sensitive to batch size choice and the LR choice is often around 0.01.
  - momentum can be used to accelerate convergence of regular SGD, however it adds another hyperparameter (the velocity weight) to be set.

- Adam and RAdam are good alternatives
  - requiring smaller learning rate values (0.001 or lower) and larger batch sizes when compared to SGD

1 - Optimizer

Available optimizers (tf.keras.optimizers):

- SGD
- RMSprop
- Adam
- Adadelta
- Adagrad
- Adamax
- Nadam
- Ftrl

2 - Learning Rate

- Controls how much to change the model in response to the estimated error each time the model weights are updated

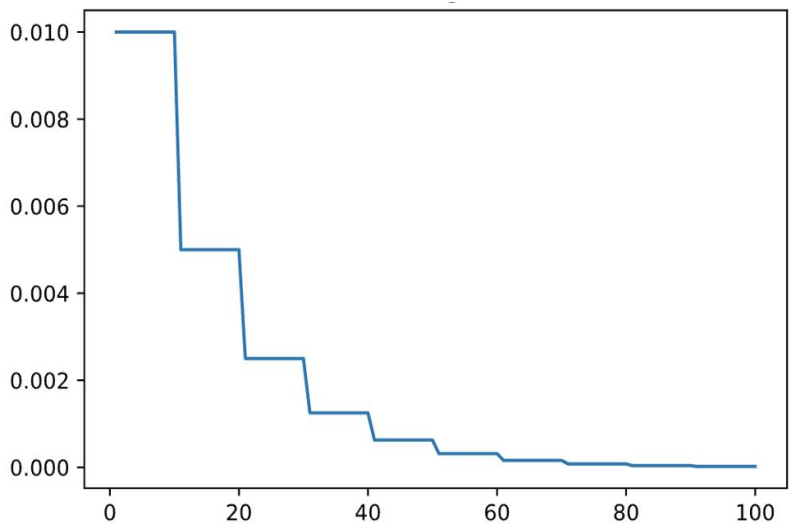
- A bad learning rate choice may ruin all other choices.
  - parameter adjustment is not uniform along the training process.
  - a learning rate/step adaptation using scheduling should always be considered

1. Step Decay
2. Exponential Decay
3. Cosine Annealing

2 - Learning Rate

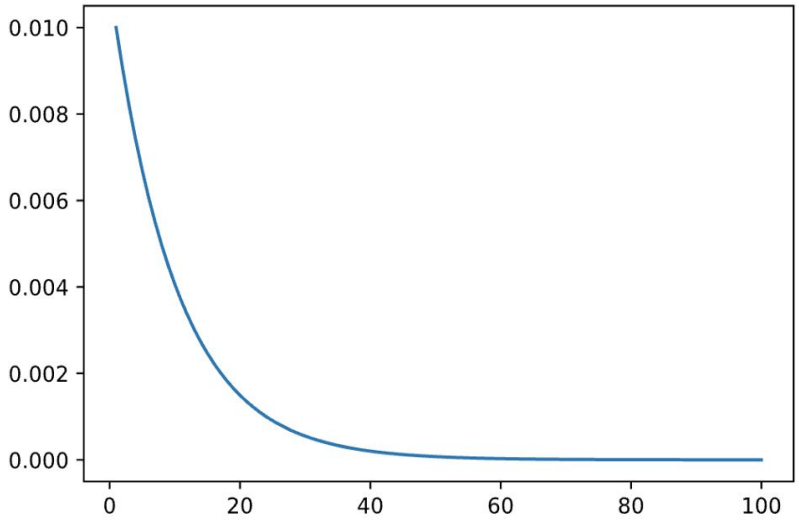
1. **Step Decay:** decreases the learning rate by some factor along the epochs or iterations.

For example: halving the value every 10 epochs



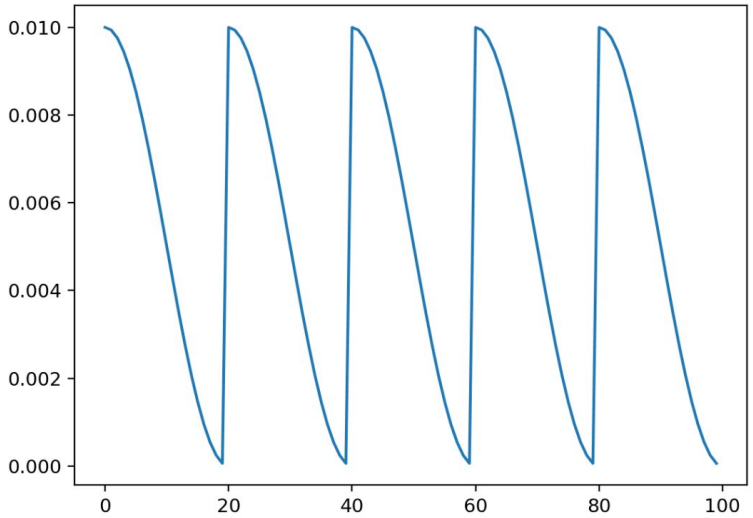
2 - Learning Rate

2. **Exponential Decay**: reduces the learning rate exponentially



2 - Learning Rate

3. Cosine Annealing: continuously decreases to a minimum value and increase it again, which acts like a simulated restart of the learning process



3 - Batch Size

- 32 or 64 is a good starting point

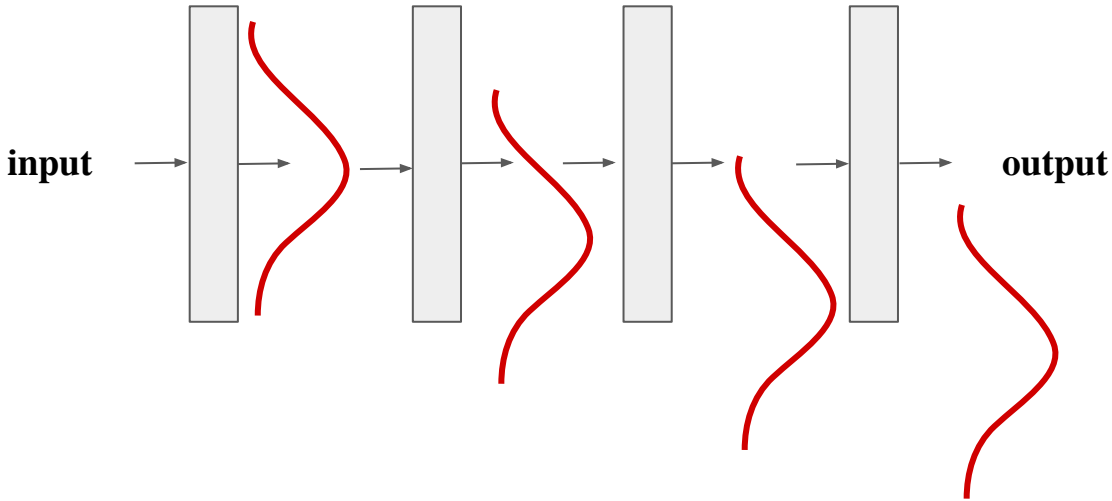
- Depends on the task/model
  - contrastive learning usually require bigger batches sizes:
  - (512, 1024, 2048, 4096)

4 - Normalization

- Normalizing inputs and targets is a staple of classic machine learning. Within the deep learning literature there are however many different ways of introducing normalization within the architecture.

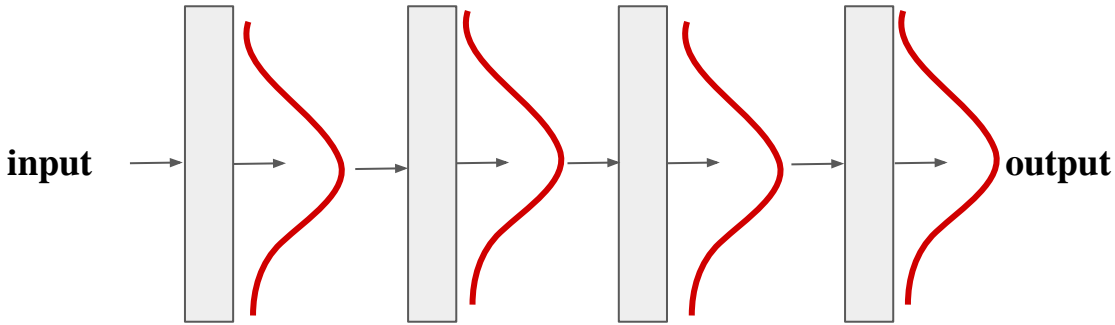
- Since deep models are composite functions with each taking in the input of the previous one, normalization is beneficial to training to keep those intermediary feature maps within some range.

4 - Normalization



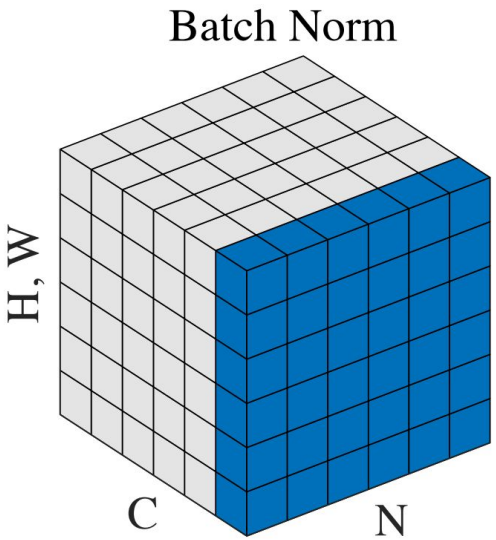
Internal Covariate Shift Problem

4 - Normalization



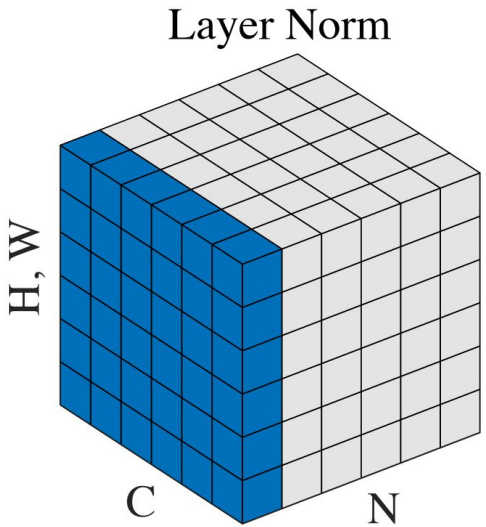
4 - Normalization

1. Batch Norm. (BN): works like a layer that standardizes the feature maps across each input in a mini-batch. Also accelerates training of deep networks



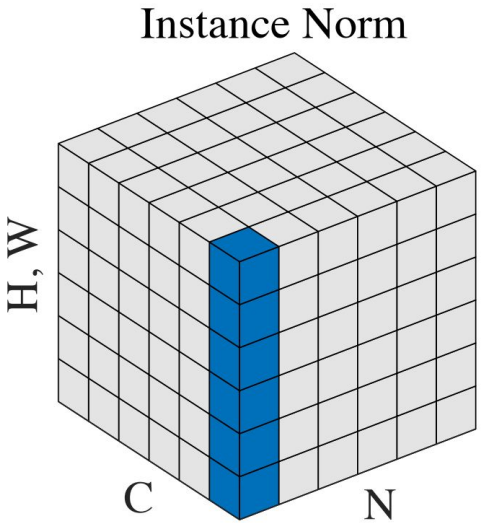
4 - Normalization

2. Layer Norm: performs standardization for each individual sample but takes mean and standard deviation from all feature maps.



4 - Normalization

3. Instance Norm: can be also designed as a layer, but instead of performing standardization across input samples, it does so for each channel of each individual sample. It's performance is worse than BN for recognition.



4 - Normalization

3. Instance Norm

- It was designed specifically for generative and style transfer models.

- Introduced because in BN batch-wise information normally leads these models to collapse, or suffer from the difference in contrast in different input images

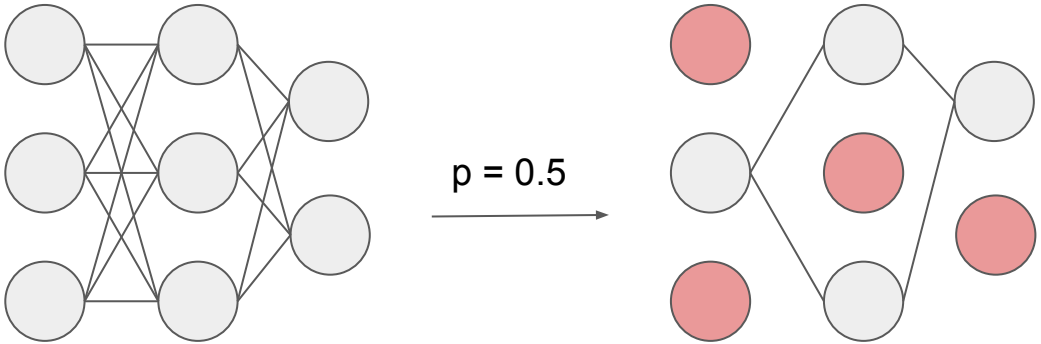
5 - Regularization

- Comprises mechanisms to help find the best parameters while minimizing the loss function.

- During the convergence process of deep networks, several combinations of parameters can be found to correctly classify the training examples.

5 - Regularization

- Dropout works by deactivating p% of neurons, mainly after dense layers.
  - This avoids some neurons to overspecialize/memorize specific data.
  - At each iteration of training, dropout provides different subsamples of activations, at different stages of the network. Consequently, this mechanism prevents overfitting during training. During inference dropout is turned off so that all neurons are activated.



6 - Data Augmentation

- Data augmentation techniques focus exclusively on increasing the size and variability of the training set.
  - Conceptually, it generates new instances derived from the original training set by manipulating the features and replicating the original label to the generated example.
  - Thus the training set becomes more variable and larger.
  - Data augmentation can also be used to balance datasets, controlling one of the drawbacks of deep learning.

6 - Data Augmentation

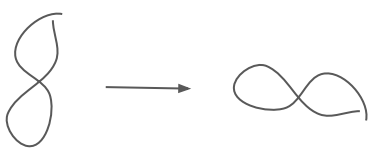
1. Image manipulation
2. Learned-based approaches
3. Meta-learning

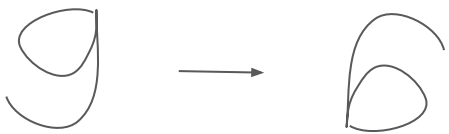
6 - Data Augmentation

1. Image manipulation: techniques for generating modified versions of an image contained in the training set

   - rotations
   - crop
   - color space transformation
   - mixing images
   - noise injection

A recurrent concern in these techniques is to ensure that the transformation performed does not alter the correct image label.





6 - Data Augmentation

2. Learning-based methods

   - use secondary neural networks to generate images
   - adversarial learning
   - generate synthetic images that reduce the possibility of attack

6 - Data Augmentation

3. Meta-learning: optimizing neural networks with neural networks.
    - find which data augmentation provide the lowest classification loss for a given dataset

6 - Data Augmentation

- The more examples available, the better will be the learning.
  - larger datasets provide better results .

- This resource provides a way to increase variability. More than increasing dataset size, data augmentation can be used to balance datasets, controlling one of the drawbacks of deep learning.

Let’s practice