Introduction to Deep Learning (I2DL)

Exercise 8: CIFAR 10 with Pytorch Lightning

Exercise plan: Recap and Outlook

Exercise 03: Dataset and Dataloader
Exercise 04: Solver and Linear Regression
Exercise 05: Neural Networks
Exercise 06: Hyperparameter Tuning

Numpy
(Reinvent the wheel)

Exercise 07: Introduction to Pytorch
Exercise 08: Cifar10 with Pytorch

Pytorch/Tensorboard

Exercise 09: Convolutional Neural Networks
Exercise 10: Semantic Segmentation
Exercise 11: Recurrent Neural Networks

Applications
(Hands-off)

Today's Outline

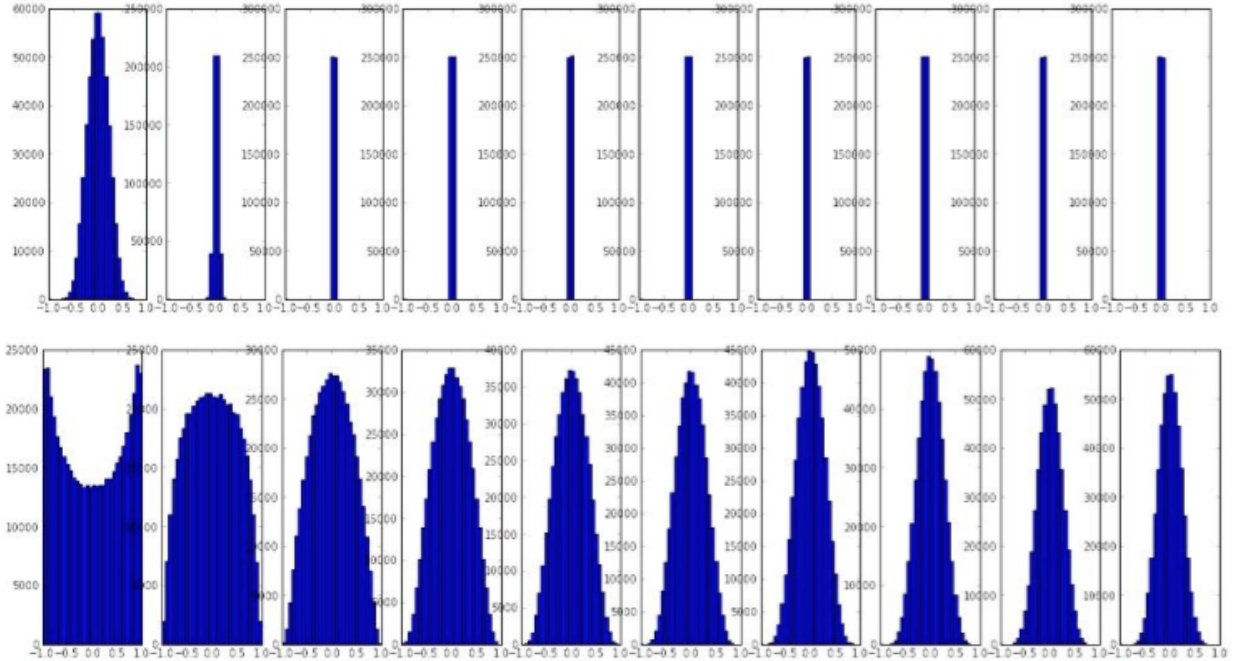
- How to improve your training!
  - Batch Normalization
  - Dropout
  - Data Augmentation
- Submission 8: CIFAR 10 Classification with Pytorch
  - Start: **June 11, 2020 12.00**
  - End: **June 17, 2020 23.59**
- Hyperparameter tuning

Improve your training!

Batch Normalization

(1_BatchNormalization-optional.ipynb)

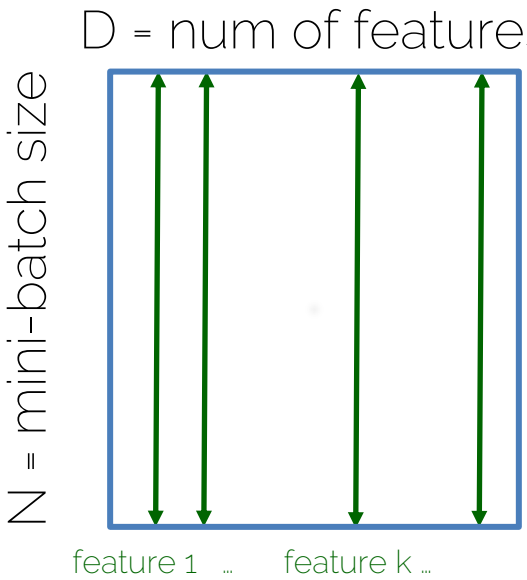
- All we want is that our activations do not die out



Batch Normalization

(1_BatchNormalization-optional.ipynb)

- Wish: Unit Gaussian activations



Mean of your mini-batch examples over feature k

$$\hat{\boldsymbol{x}}^{(k)} = \frac{\boldsymbol{x}^{(k)} - E[\boldsymbol{x}^{(k)}]}{\sqrt{Var[\boldsymbol{x}^{(k)}]}}$$

Unit gaussian

Batch Normalization

(1_BatchNormalization-optional.ipynb)

- 1. Normalize

$$\hat{\boldsymbol{x}}^{(k)} = \frac{\boldsymbol{x}^{(k)} - E[\boldsymbol{x}^{(k)}]}{\sqrt{Var[\boldsymbol{x}^{(k)}]}}$$

- 2. Allow the network to change the range

$$\boldsymbol{y}^{(k)} = \gamma^{(k)}\hat{\boldsymbol{x}}^{(k)} + \beta^{(k)}$$

backprop

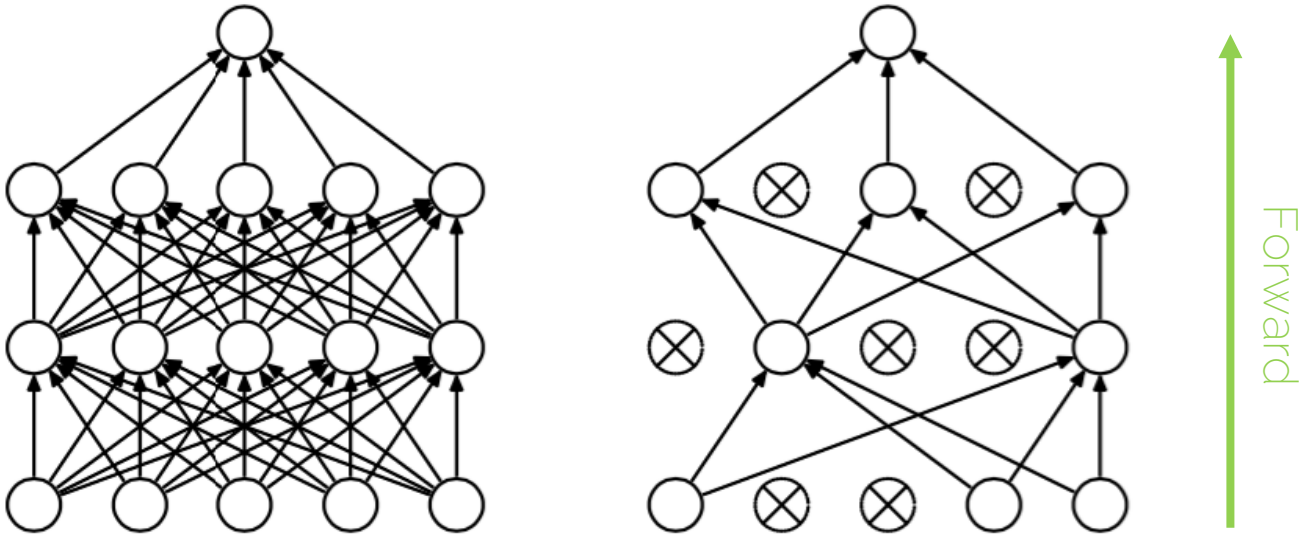
The network *can* learn to undo the normalization

$$\gamma^{(k)} = \sqrt{Var[\boldsymbol{x}^{(k)}]}$$

$$\beta^{(k)} = E[\boldsymbol{x}^{(k)}]$$

Dropout

(2_Dropout-optional.ipynb)



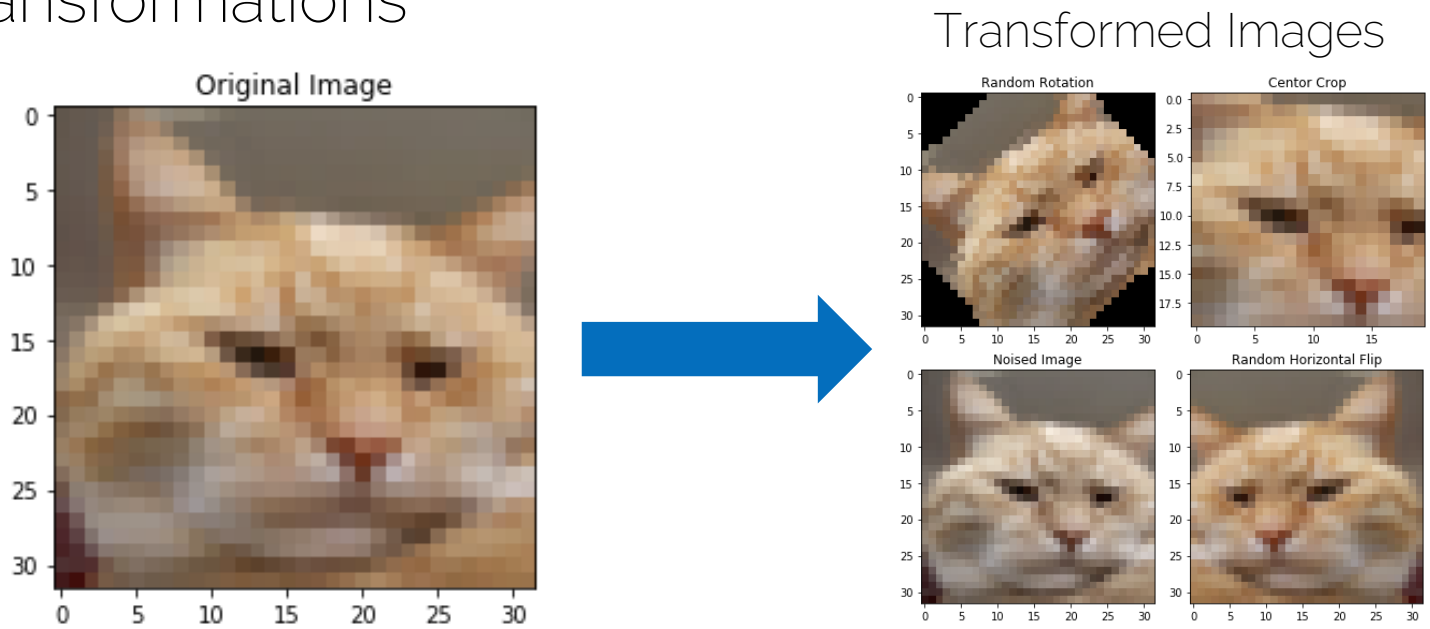
(a) Standard Neural Net

(b) After applying dropout.

- Using half the network = half capacity

Data Augmentation
(3_Data_Augmentation.ipynb)

- A classifier has to be invariant to a wide variety of transformations



Submission 8

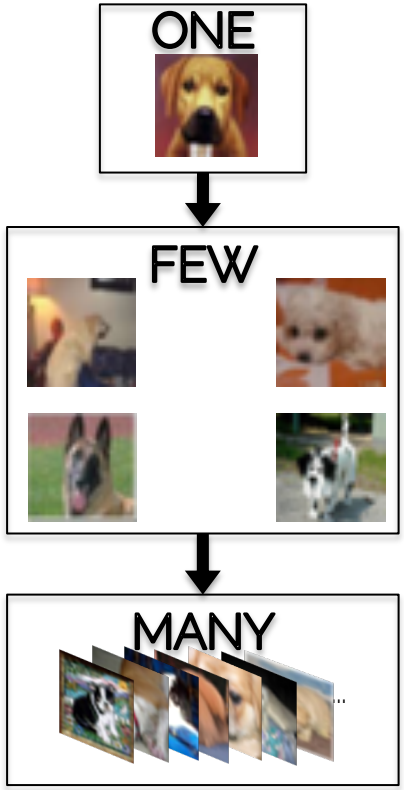
Submission Goal: Cifar10 Classification

(4_Cifar10_PytorchLightning.ipynb)

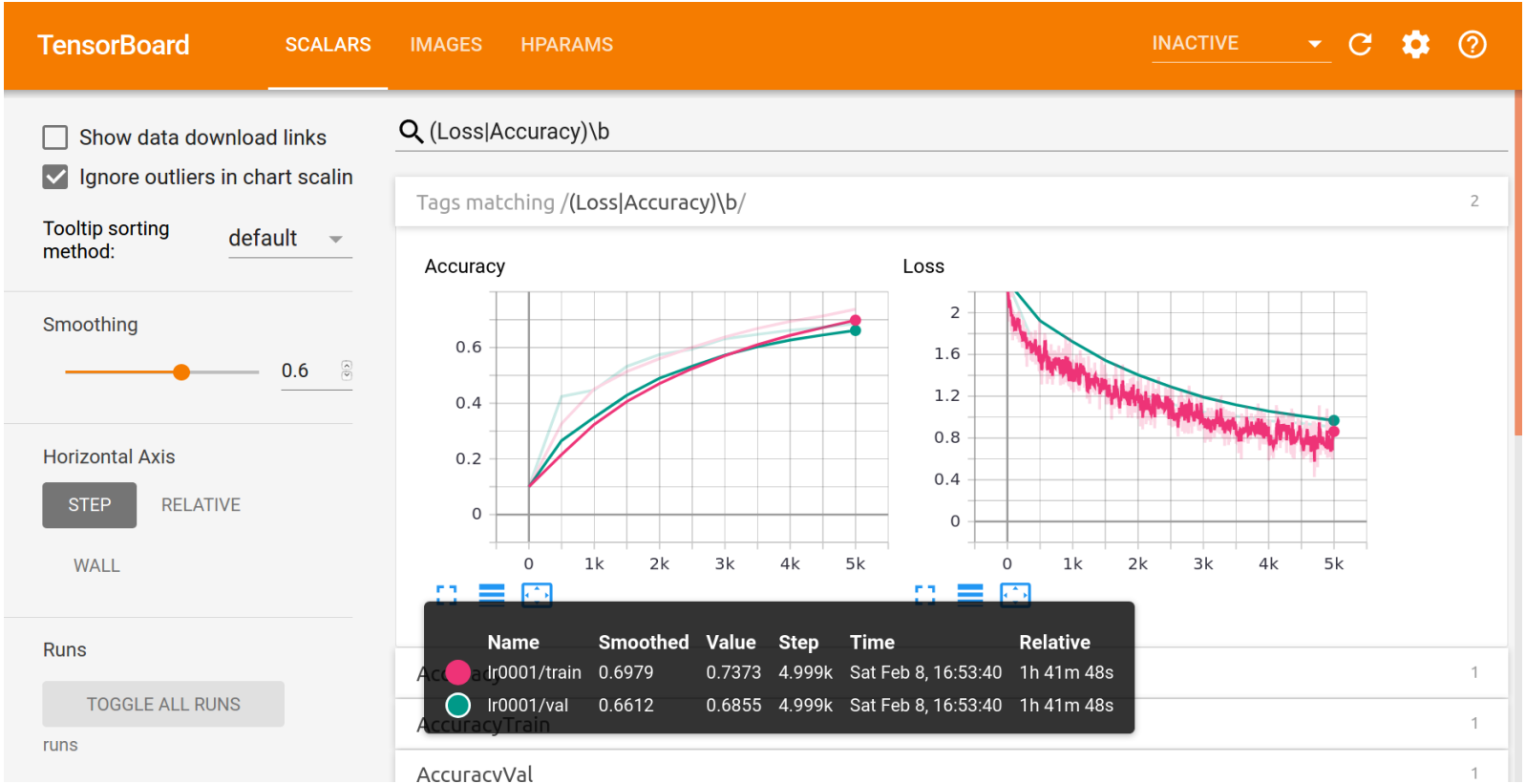
Only affine layers (nn.Linear()), no convolution layers!

How to Start

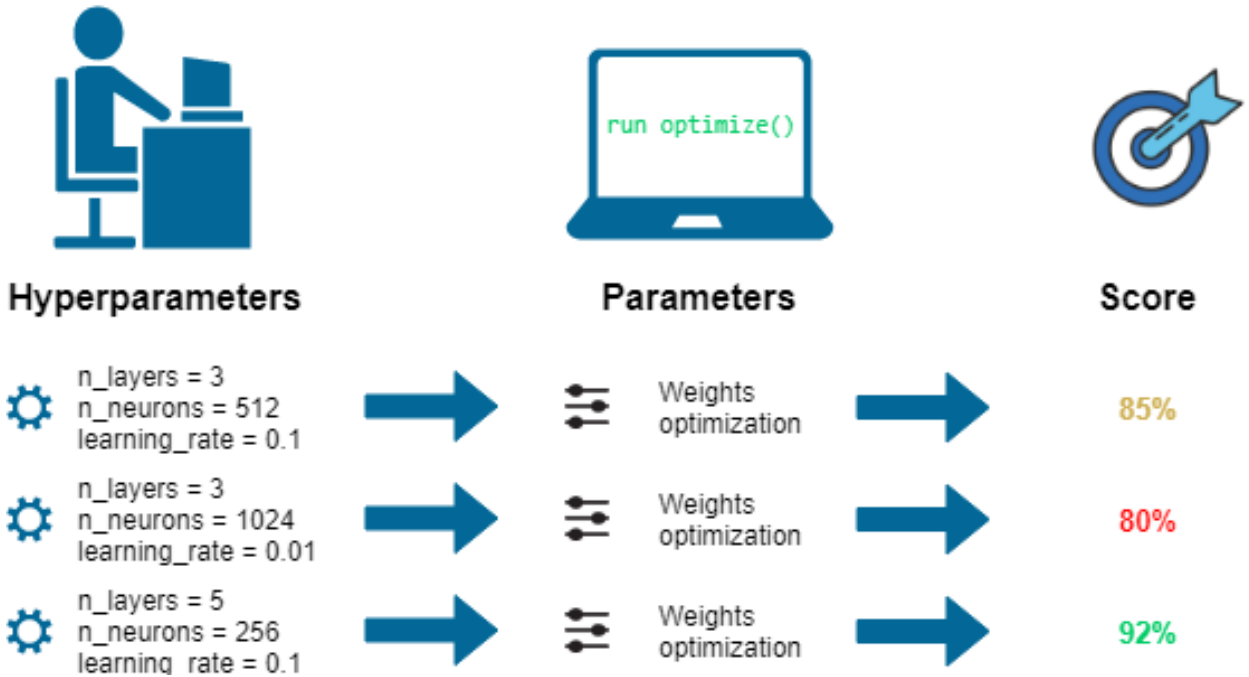
- Start with single training sample
  - Check if output correct
  - Overfit → train accuracy should be 100% because input just memorized
- Increase number samples
  - Augment your data
- Increase complexity of your model
  - Add more layers / neurons
- When overfitting
  - Add regularization (Dropout, Batchnormalization)



Monitor loss and accuracy



Hyperparameter Tuning
(5_HyperParamTuning_Optuna.ipynb)



Source: https://images.deepai.org/glossary-terms/05c646fe1676490aa0b8cab0732a02b2/hyperparams.png

Hyperparameter Tuning
(5_HyperParamTuning_Optuna.ipynb)

- Optuna is an automatic hyperparameter optimization framework: https://github.com/optuna/optuna!

- Hyper parameters:
  - #layers
  - #neurons
  - Learning rate
  - Dropout
  - Batch size
  - ……

OPTUNA

Submission

- Submission **Start: June 11**, 2020 12.00
- Submission **Deadline : June** 17, 2020 23.59
- Submit your trained Pytorch Lightning Model!
- Your model's **accuracy** is all that counts!
  - At least **50%** to pass the submission
  - There will be a **leaderboard** of all students!

Rank	User	Score	Pass
#1	s0270	51.65	✔
#2	s0262	42.98	✘
#3	s0265	10.35	✘

See you next week!