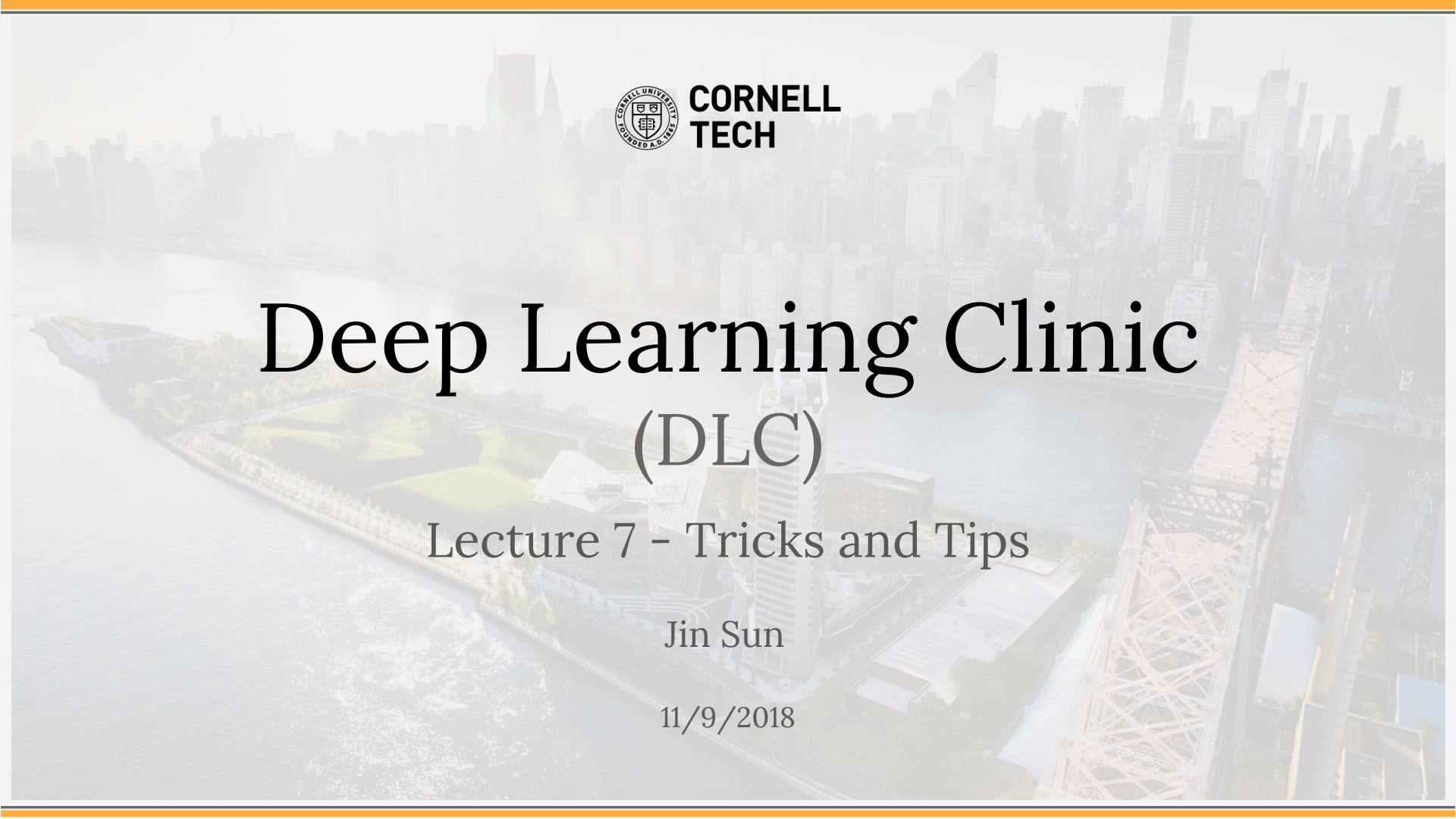


CORNELL TECH

Deep Learning Clinic (DLC)

Lecture 7 - Tricks and Tips

Jin Sun

11/9/2018

Today - Tricks and Tips in DL Training

- **Overview**
- Data Preprocessing
- Network Design
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- Optimization
- Transfer Learning

Slides from http://cs231n.stanford.edu/syllabus.html

Overview

Training a neural network is an art!

But there are tricks people found useful.

Overview

Network Design

Activation, preprocessing, initialization, regularization

Training Dynamics

Monitor the changing of train/val loss, parameter updates, hyperparameters

Recap: Training A Network

Training Loop:

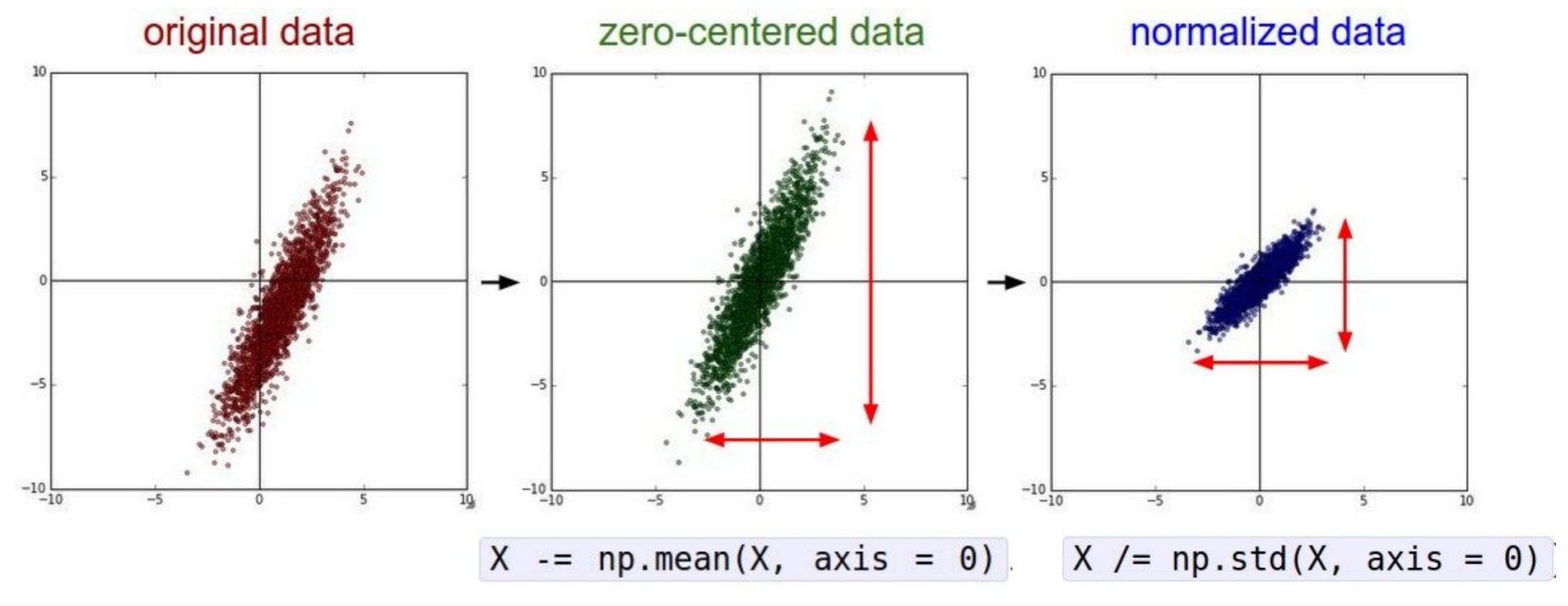
1. Sample a **batch** of data.
2. **Forward** pass the data through the network and calculate the loss.
3. **Backprop** the loss to calculate gradients of the network.
4. **Update** parameters using gradient based optimization method.

Today - Tricks and Tips in DL Training

- Overview
- **Data Preprocessing**
- Network Design
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- Optimization
- Transfer Learning

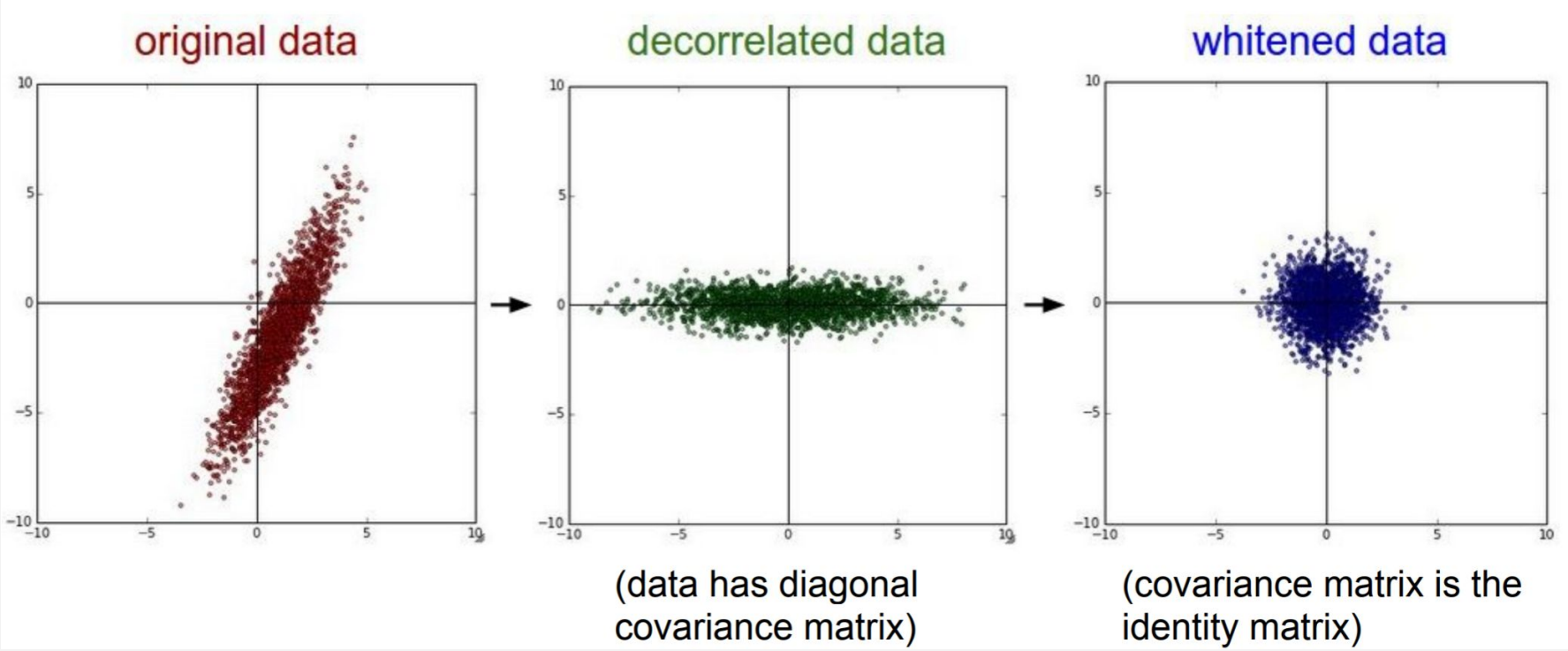
Slides from http://cs231n.stanford.edu/syllabus.html

Data Preprocessing



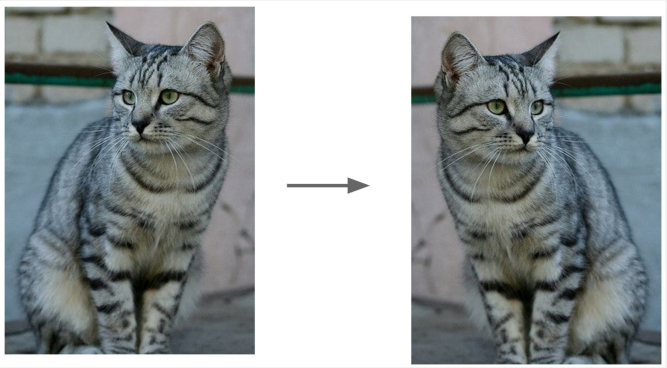
In practice, only do zero-centering for images.

Data Preprocessing - PCA and Whitening

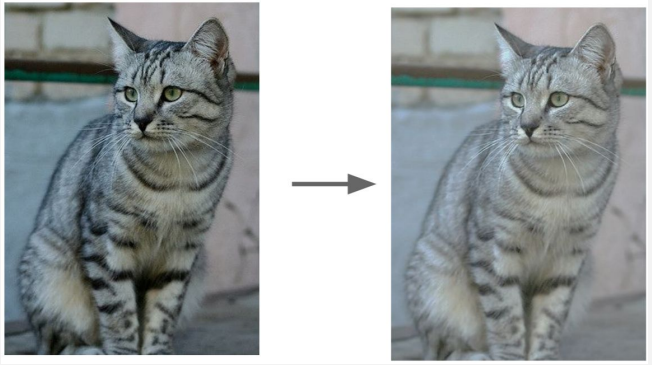


Data Augmentation

Horizontal Flip:

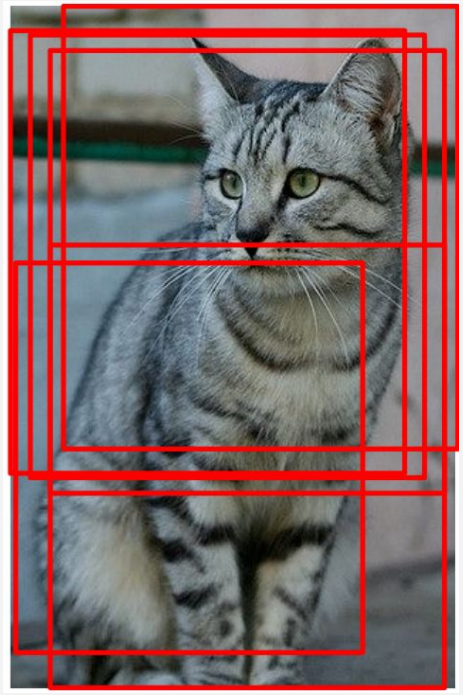


Random Contrast and Brightness

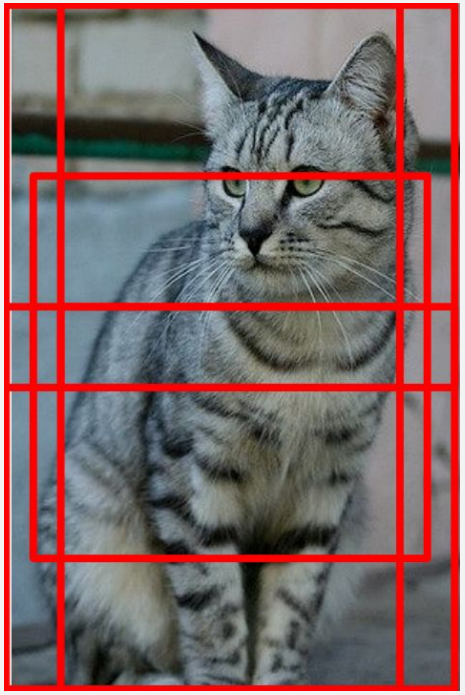


Data Augmentation

Random Crops and Scales (Train)



Fixed Crops and Scales (Test)

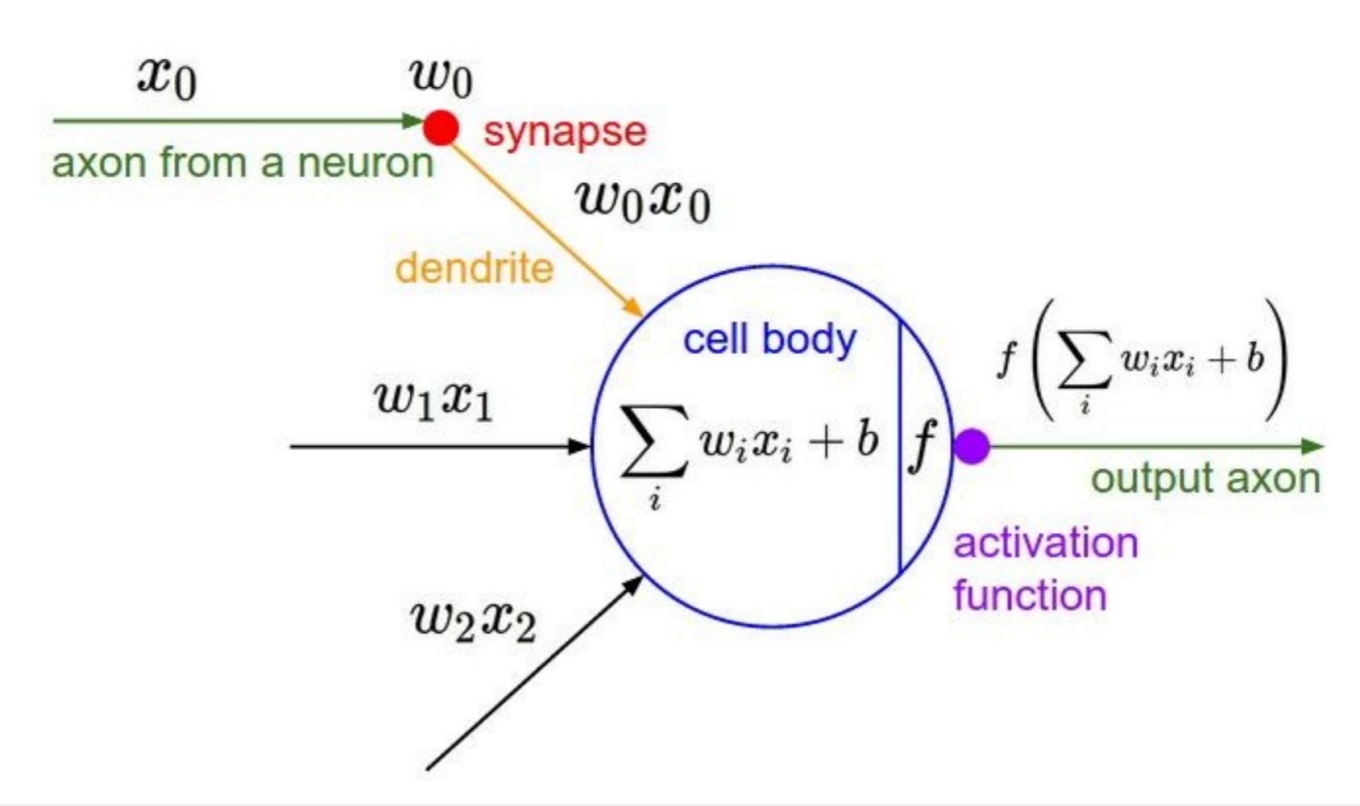


Today - Tricks and Tips in DL Training

- Overview
- Data Preprocessing
- **Network Design**
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- Optimization
- Transfer Learning

Slides from http://cs231n.stanford.edu/syllabus.html

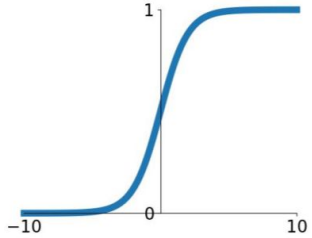
Activation Functions



Activation Functions - the Gallery

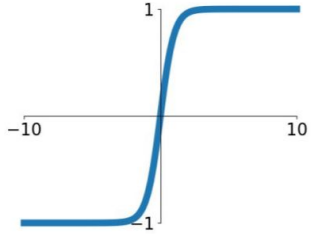
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



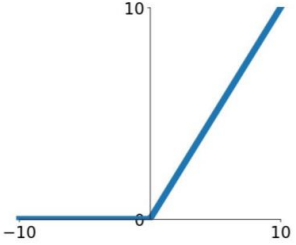
tanh

$$\tanh(x)$$



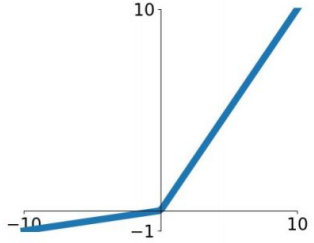
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

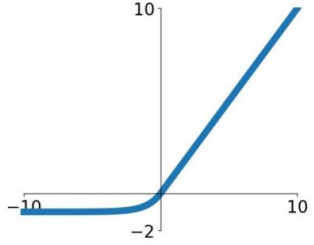


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions - Summary

Use **ReLU** as the default

Try Leaky ReLU / Maxout / ELU for alternatives

Try tanh

Sigmoid is usually not recommended.

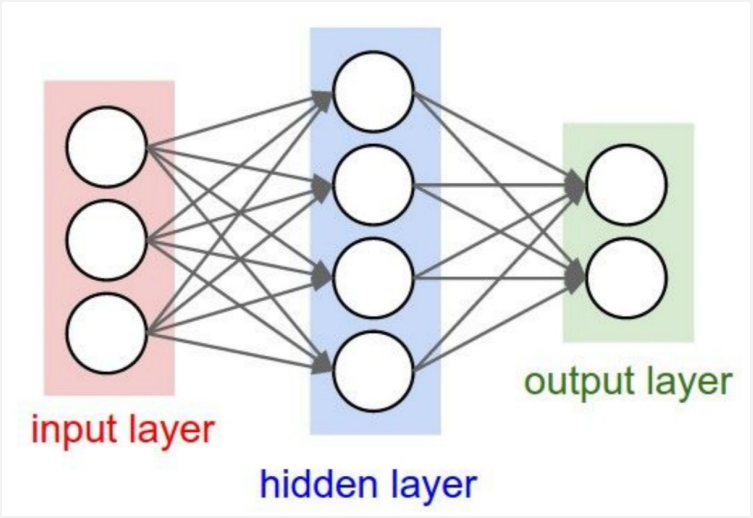
Weight Initialization

Do not use constant weights.

Small Gaussian random numbers not working well for deep networks..

Recommended:

Xavier initialization, and its variants.



Still an active research area.

Batch Normalization

Idea:

Make a batch of activations to have zero-mean and unit-variance.

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

Works as a layer in neural networks.

Put it after fully connected or convolutional layers, before nonlinearity.

Batch Normalization

Benefit:

Improve gradient flow in the network.

Allows higher learning rate.

Works also like a regularizer.

During Testing:

Use a single fixed mean/std obtained from training.

Monitoring the Learning Process

Some simple things to try and observe:

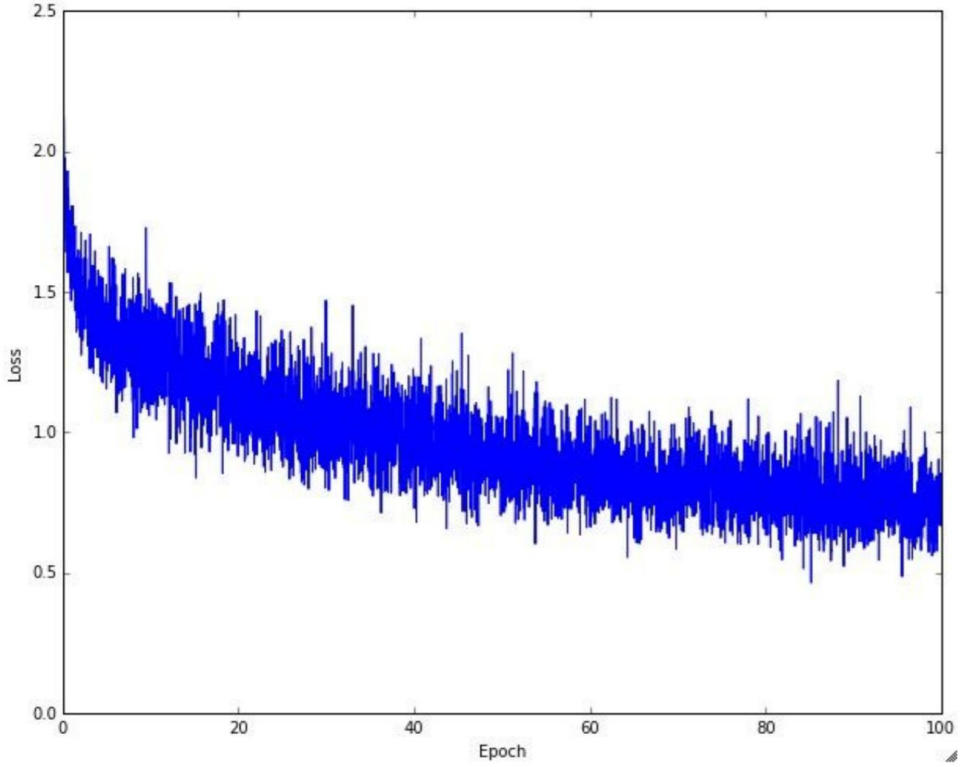
You should be able to overfit your model to a small portion of the data. Do this as a sanity check to make sure the pipeline works.

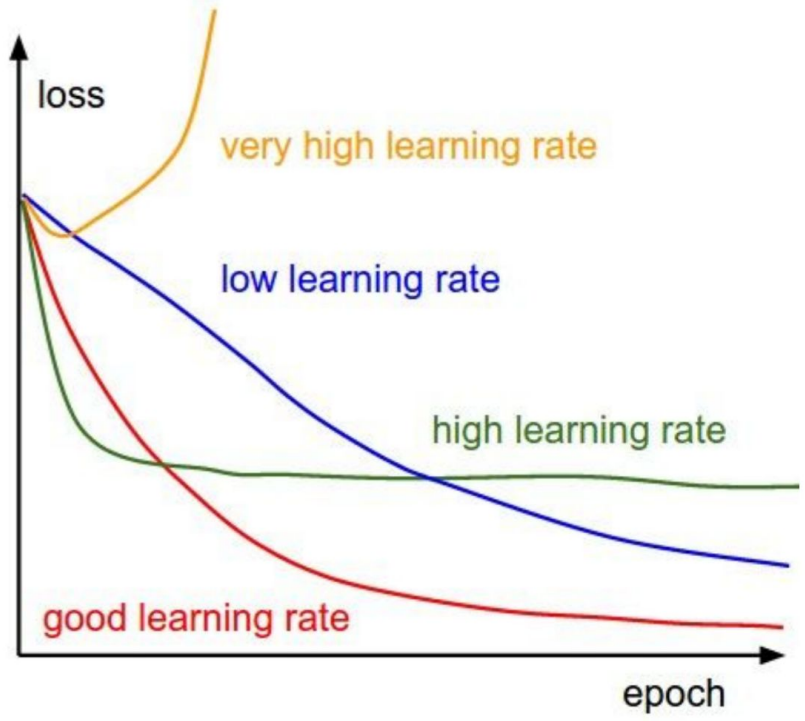
Train loss is not going down: try higher learning rate

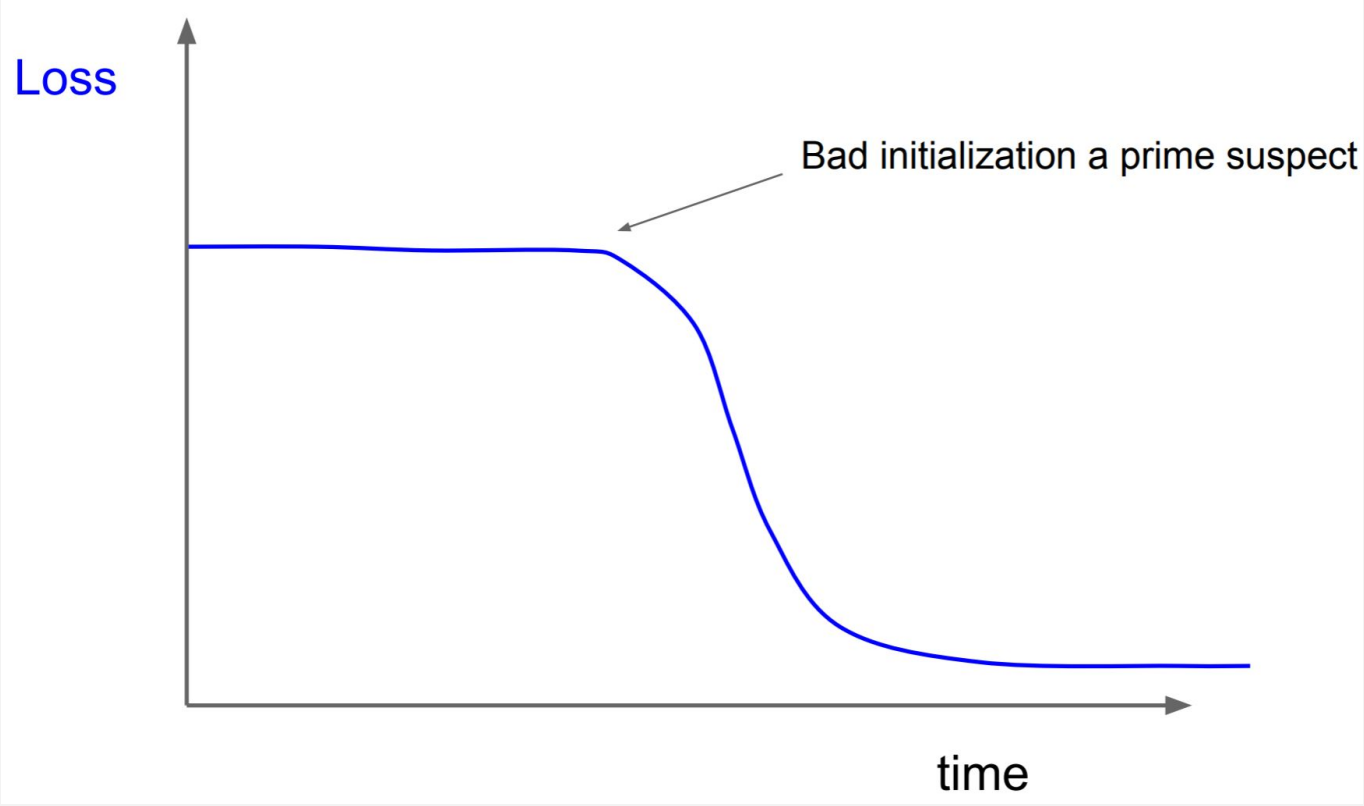
Train loss explodes: try lower learning rate

Find a range of learning rate by doing a simple set of experiments.

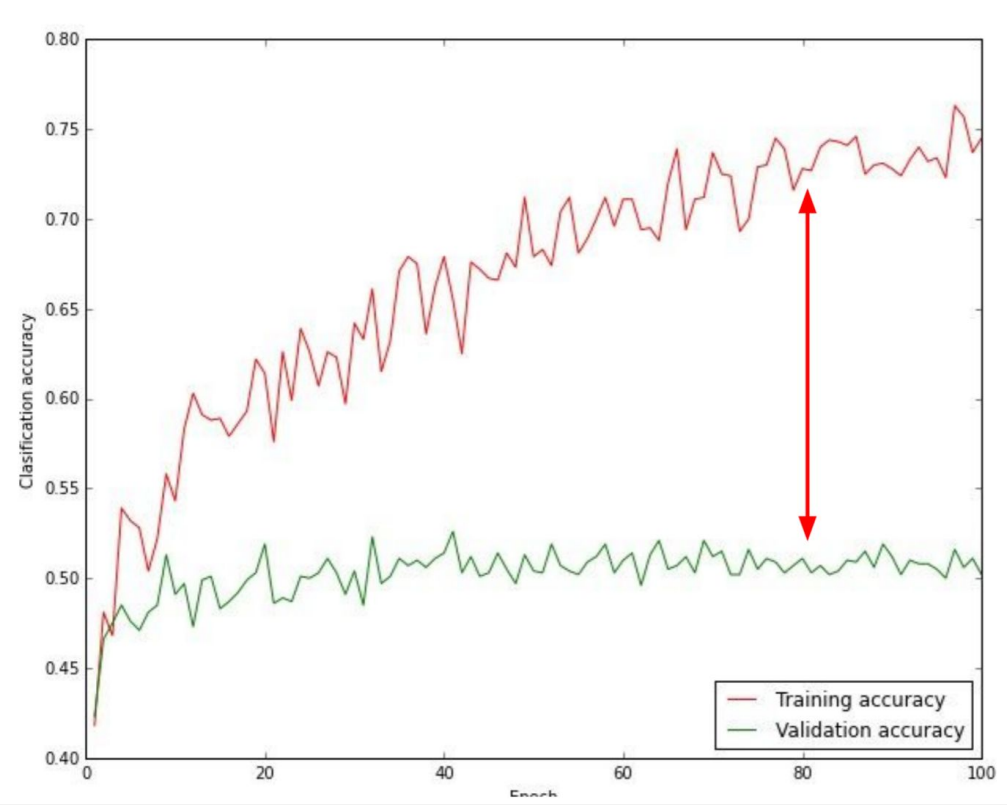
Learning Rate vs Curves






Loss
Bad initialization a prime suspect
time

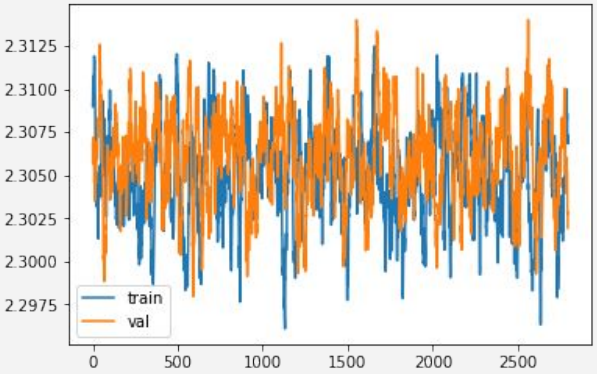
Sign of Overfitting



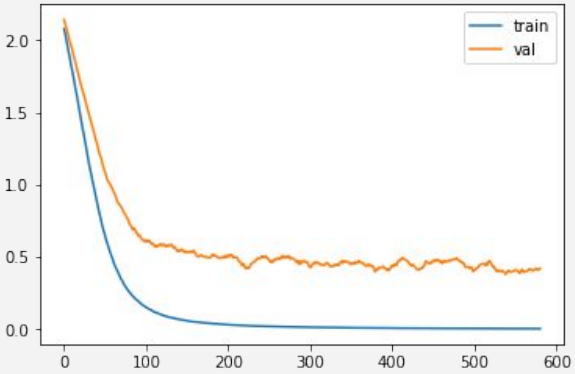
Big gap: Overfitting

No gap: Maybe model can be more complex

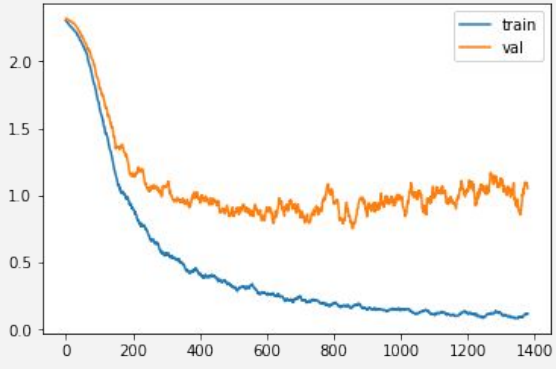
Early Stopping



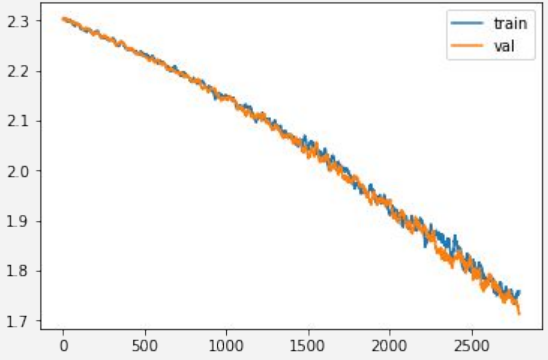
Not learning: gradients not applied to weights



Overfit: model too large/dataset too small

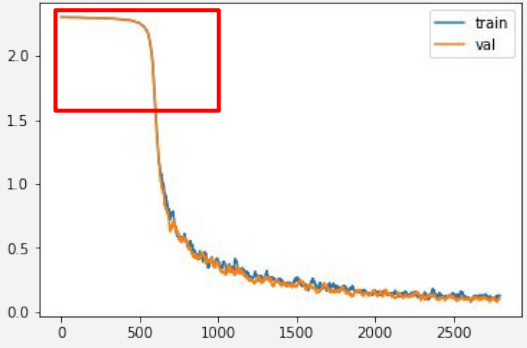


More extreme case of overfitting

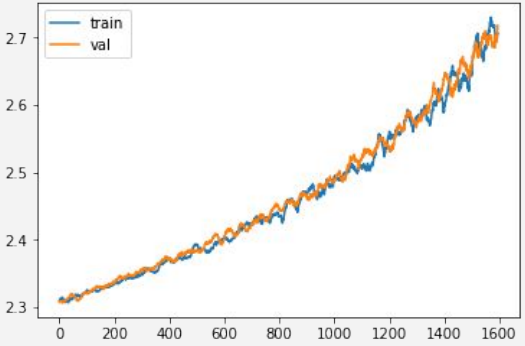


Not converged yet: need longer training

slow start



Slow start: initialization weights too small



Applied the negative of gradients

Hyperparameters

Definition:

All the setup variables that not changing during the learning process. For example: number of layers.

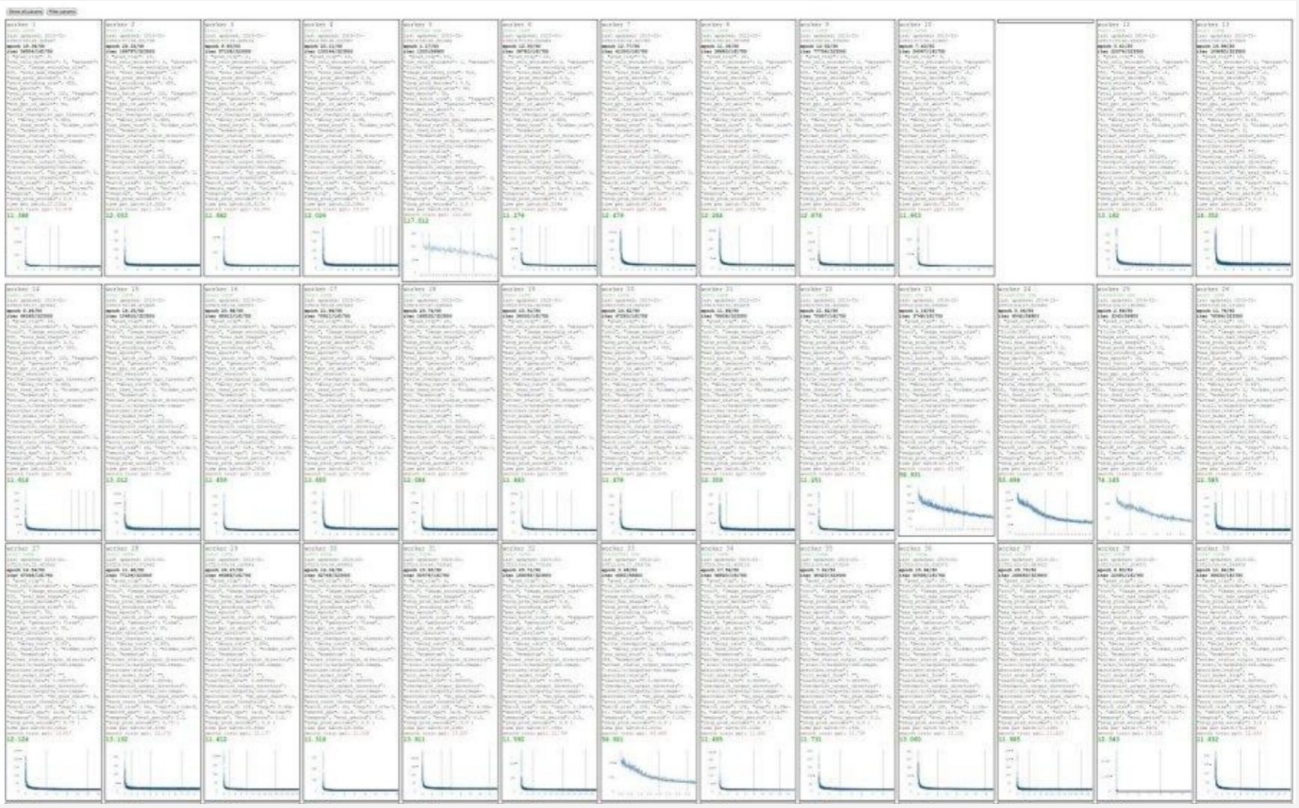
Strategy:

Cross-validate different models on train/val set.

Grid search on the parameters (e.g., learning rate, regularizer factors).

Try log-space.

Hyperparameters



Today - Tricks and Tips in DL Training

- Overview
- Data Preprocessing
- Network Design
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- **Optimization**
- Transfer Learning

Slides from http://cs231n.stanford.edu/syllabus.html

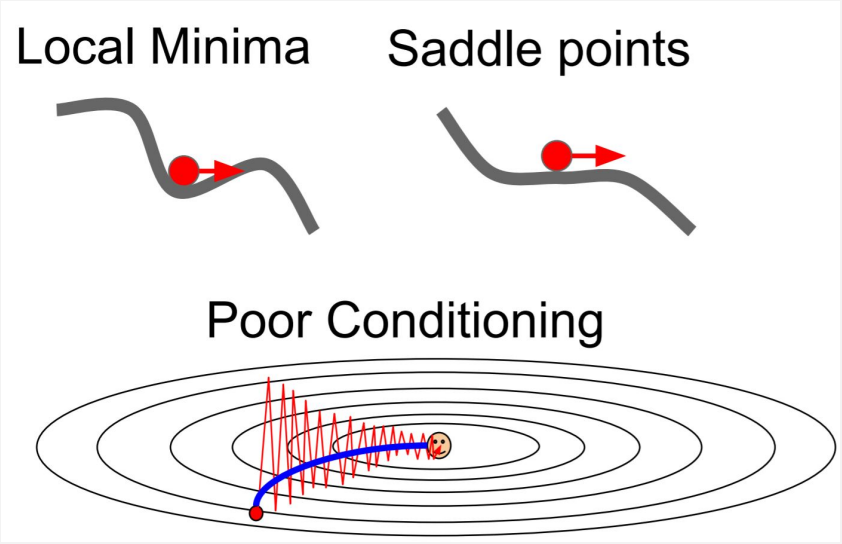
Optimization

Stochastic Gradient Descent (**SGD**)

Usually works with momentum.

Adagrad, **RMSProp**, **Adadelta**

Recommended: **Adam**

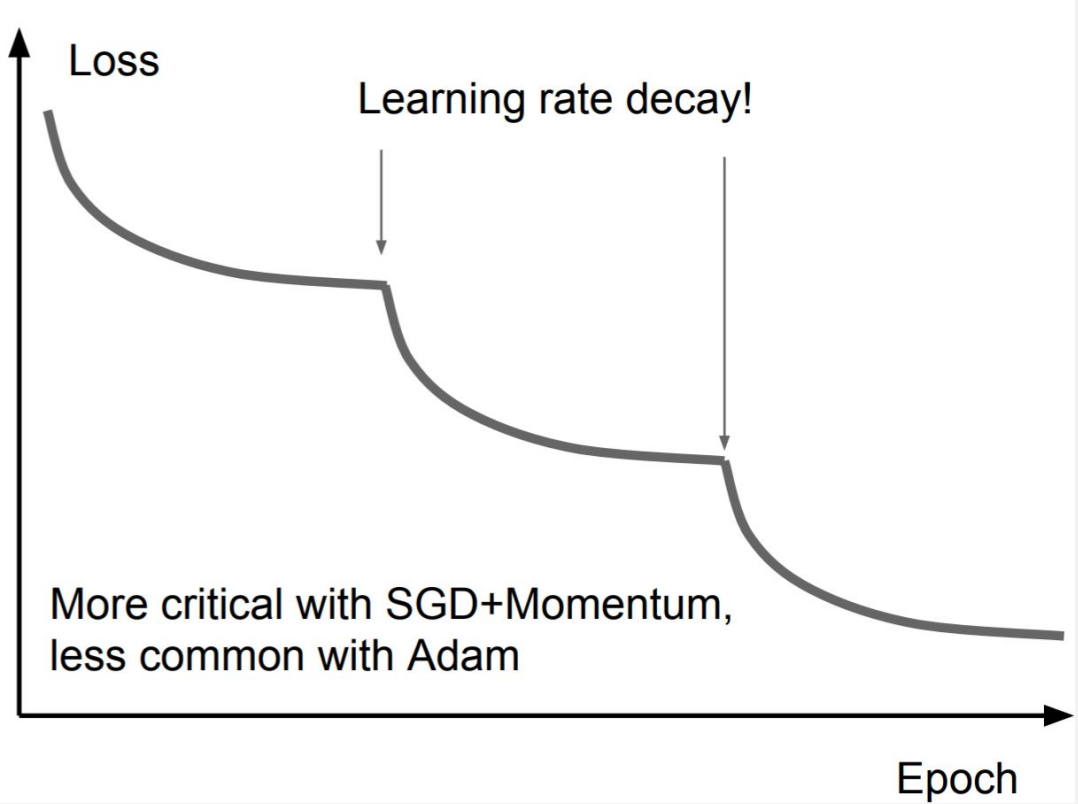


Learning Rate Decay Strategy

Step Decay

Exponential Decay

1/t Decay

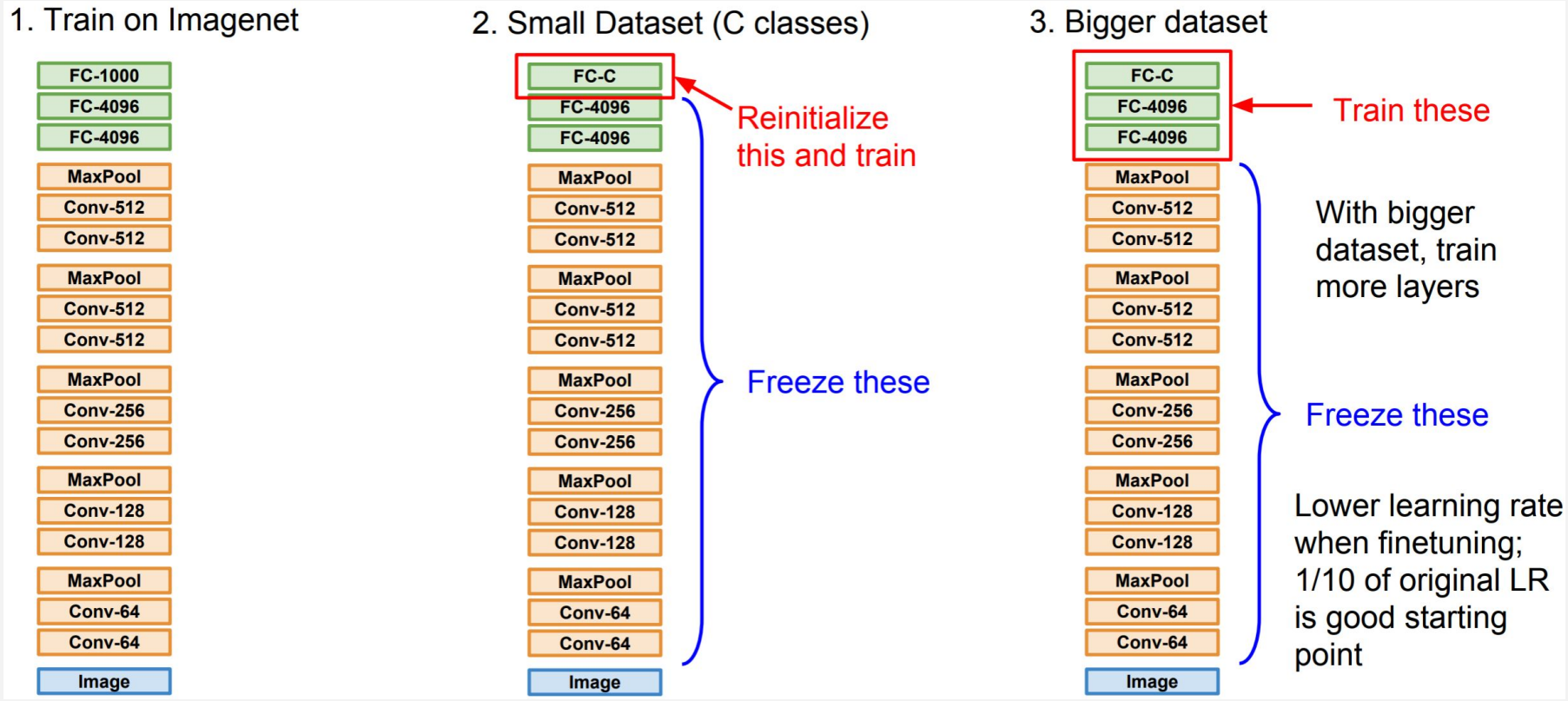


Today - Tricks and Tips in DL Training

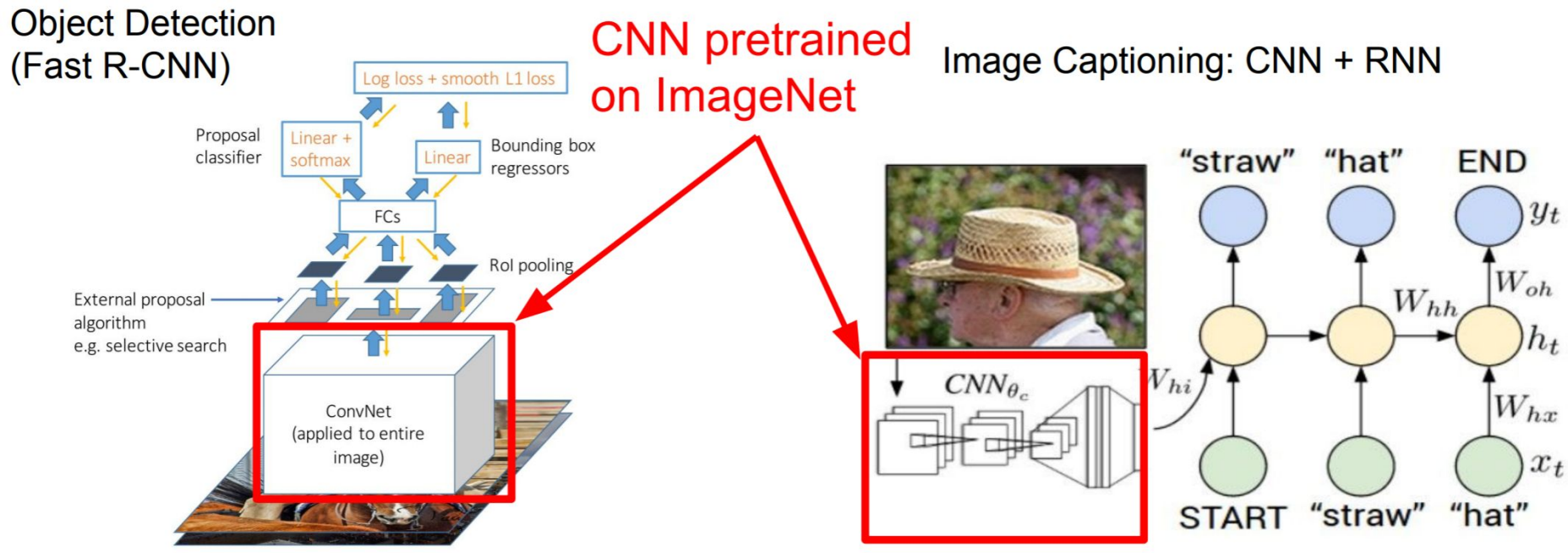
- Overview
- Data Preprocessing
- Network Design
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- Optimization
- **Transfer Learning**

Slides from http://cs231n.stanford.edu/syllabus.html

Transfer Learning



Transfer Learning Is the Norm



Where to Find Pretrained Models

Caffe:

https://github.com/BVLC/caffe/wiki/Model-Zoo

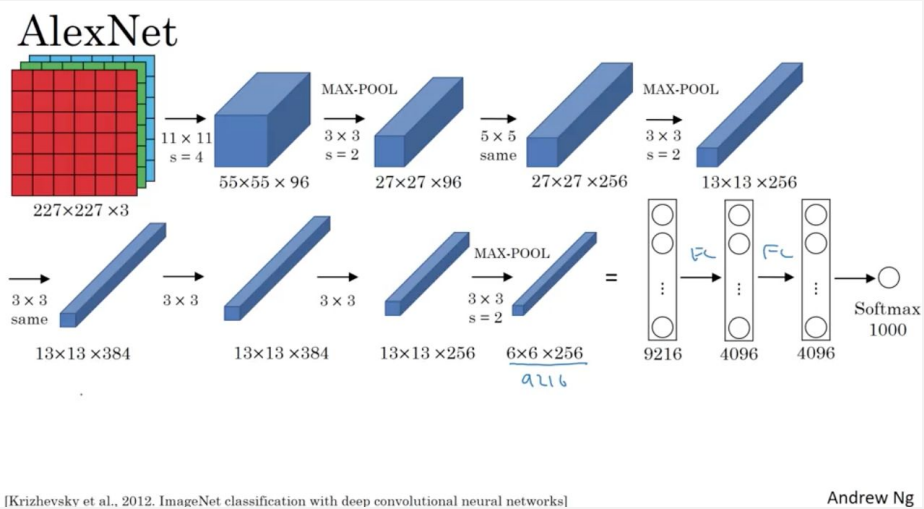
Tensorflow:

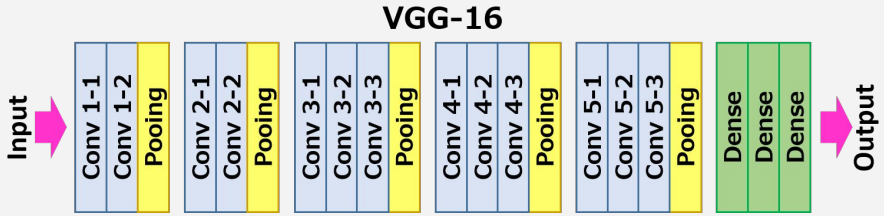
https://github.com/tensorflow/models

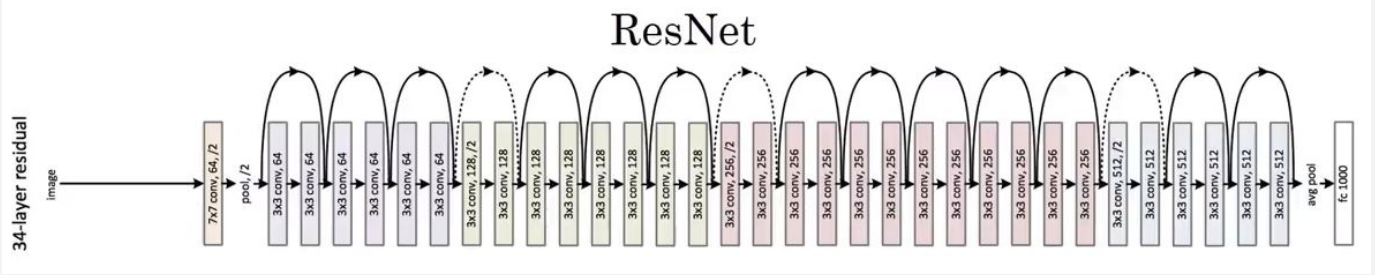
Pytorch:

https://pytorch.org/docs/master/torchvision/

Popular Models in Vision







https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgg-googlenet-and-resnet-7c81c017b848

Today - Tricks and Tips in DL Training

- Overview
- Data Preprocessing
- Network Design
  - Activation Functions
  - Weight Initialization
  - Normalization
  - Monitoring the Learning Process
  - Hyperparameters
- Optimization
- Transfer Learning

Slides from http://cs231n.stanford.edu/syllabus.html