Practical Machine Learning and Deep Learning

Vladimir Ivanov

WEEK 1

Objectives (for today)

* to learn about the course: how to success in it
* to recall neural networks, deep learning
* to understand limitations of deep learning

Course. Team and Communication

* Vladimir Ivanov (@nomemm)
* Youssef Youssry Ibrahim (@YoussefYN)
* Course Channel:

Course. Syllabus

* Review: Definitions, training DNNs
* CNNs
* RBMs, DBNs
* RNNs: LSTM, Attention, Transformers
* Graph networks: Graph convolution networks
* GANs (?) Bayesian Deep Learning (?)
* Software Tools for ML/DL
* Data Science Teams: Processes and Roles

Course. Structure

* Lectures + Quizzes (written, 10 min. on a lecture)
* Labs + Assignments (aka Homework, 4 hours / week)
* Final Project (group project, 4 hours / week)
* Optional: students' presentations, invited lectures

Course. Books

* Goodfellow et al. Deep Learning, MIT Press. 2017.
* Géron A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. – " O'Reilly Media, Inc.", 2017. **SECOND EDITION (!)**
* Osinga, Douwe. Deep Learning Cookbook: Practical Recipes to Get Started Quickly. O'Reilly Media, 2018.
* Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. – Спб.: Питер, 2018.

Course. Assessment

- Quizzes: **10 points**
- Assignments: **30 points**
- Midterm (theory + practice): **20 points**
- Final project: **30 points**
- Final exam (theory + practice): **20 points**

90+: **A**

76-90: **B**

60-75: **C**

0-59: **D**

Course. Tools

* Python (ver. 3+)
* Frameworks: Tensorflow 2.0, Pytorch
* Cloud Services: Collaboratory, AWS, Azure, GCP,...
* Docker, K8s, Kubeflow, ML Flow, Flask

Prerequisites

* Linear Algebra, Calculus courses
* Introduction to ML course
* Prob. Theory and Stat. course (or STDS)
* optionally: any online courses in ML (e.g. https://www.coursera.org/learn/machine-learning/)

How to success?

* Assignments: work hard (individually) + office hours
* Project: work in team + application modern tools
* Exams: reading the books + office hours

Resources

* http://deeplearning.net
* https://www.kaggle.com
* https://codalab.org (especially, if you do a thesis with me)
* https://www.wandb.com
* https://dvc.org

What are your ambitions?

- Answer the questions:
  - Which book I will read during the course?
  - What is the most interesting **project topic** for me?
  - Which tools I would like to study in the course?
  - Which paper(s) I would like to read / implement ?
- Fill the Google Form: **http://bit.ly/pdl2020plan**

Machine Learning

* Problem formulation
* Model families
* Optimization
* Regularization

Intro

* Goal: **to find** a function $\hat{f}$ that can perform some task of interest

* Main idea: to **use data examples** rather to specify program's behaviour

Problem Formulation

* Train set: $D = \{(x_1, y_1), \ldots (x_n, y_n)\}$
* Loss function: $L(f(x), y)$
* Approximation: $\hat{f} = \arg\min_{f \in F} \frac{1}{|D|} \sum_{(x,y) \in D} L(f(x), y)$
* Family F of possible solutions is infinite

Model Families

* Examples of model families: Support Vector Machines, **Neural Networks**, k-NN, k-means
* We assume, the true function f^* exists: $f^*(x) = y$
* **The choice** of family affects underfitting and overfitting (how?)

Empirical risk

* Assuming the samples came from $(x, y) \sim P(x, y)$
* Risk: $R(f) = \int_{x,y} L(f(x), y) P(x, y) dx dy$
* We cannot calculate R, we can only estimate the empirical risk $\hat{R}$ on the dataset D

Optimization

* The empirical risk $\hat{R}$ depends on $f(\theta)$
* Usually, $\hat{R}$ and $f(\theta)$ are differentiable w.r.t. to θ
* Gradient based methods used to iteratively update θ

$$\theta^{t+1} = \theta^t - \eta \frac{\partial \hat{R}(\theta^t)}{\partial \theta^t}$$

Regularization (Occam's razor)

* Regularization is used to prevent overfitting
* Tikhonov regularization (p-norm)

$$\hat{f} = \arg\min_{f_\theta \in F} \left(\hat{R}(f_\theta) + \lambda \, || \, \theta \, ||_p \right)$$

Break, 5 min.

Deep Learning. Review

* What do you already know about DL?
  * https://docs.google.com/forms/d/1T1hNrP0KlXAN0mR0Ji2tsBy9ZxsXuyzX6HLJnVIphrA/edit#responses
* What is the core idea?

Review.

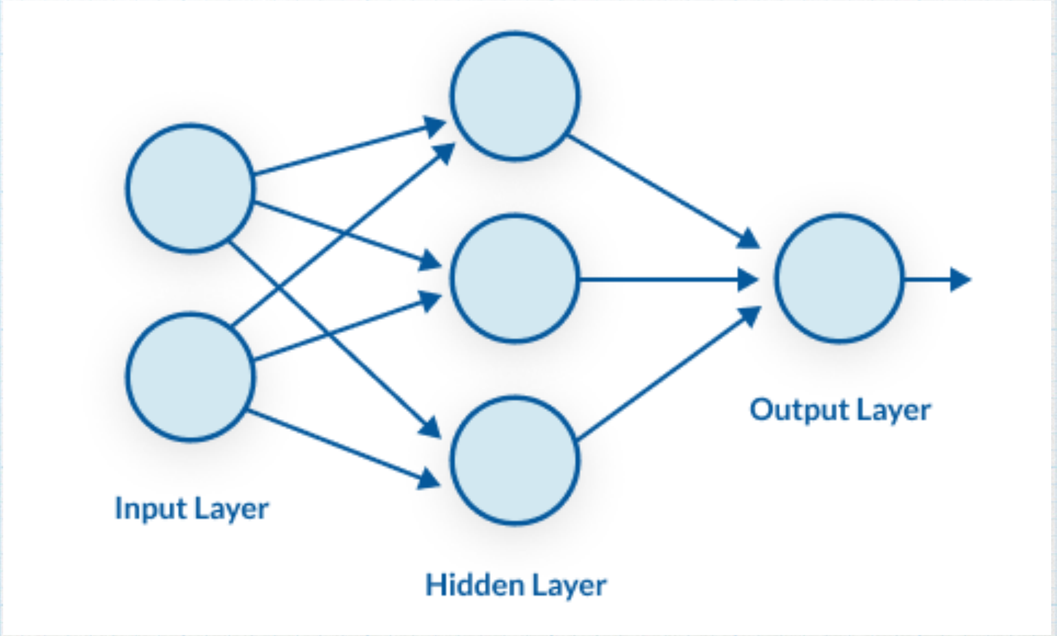
* Core idea of DL:
  * to train deep features hierarchies
  * with some form of regularization
  * to ensure good generalization.

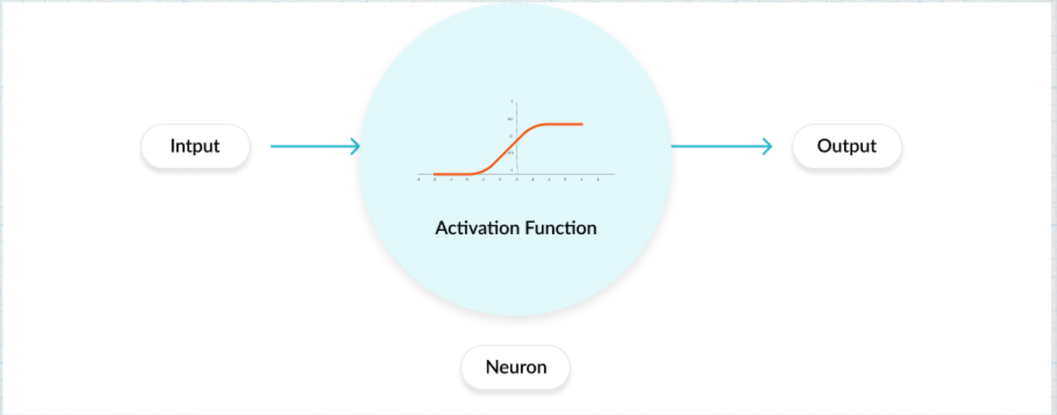
Deep neural networks

* Deep NN is a combination of a logistic regressor with a **learnt non-linear** feature transform Ψ

$$h_1(x) = \psi_1(W^1 x + b^1)$$

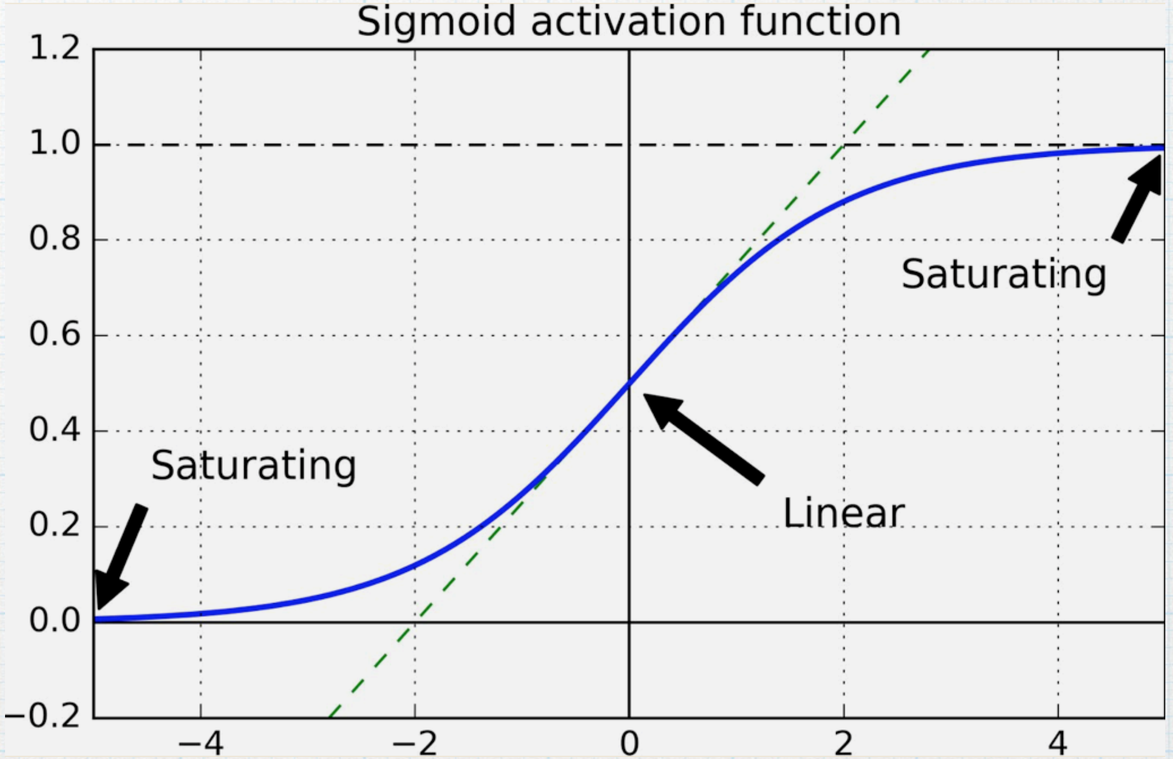
$$o(x) = \psi_2(W^2 h + b^2)$$



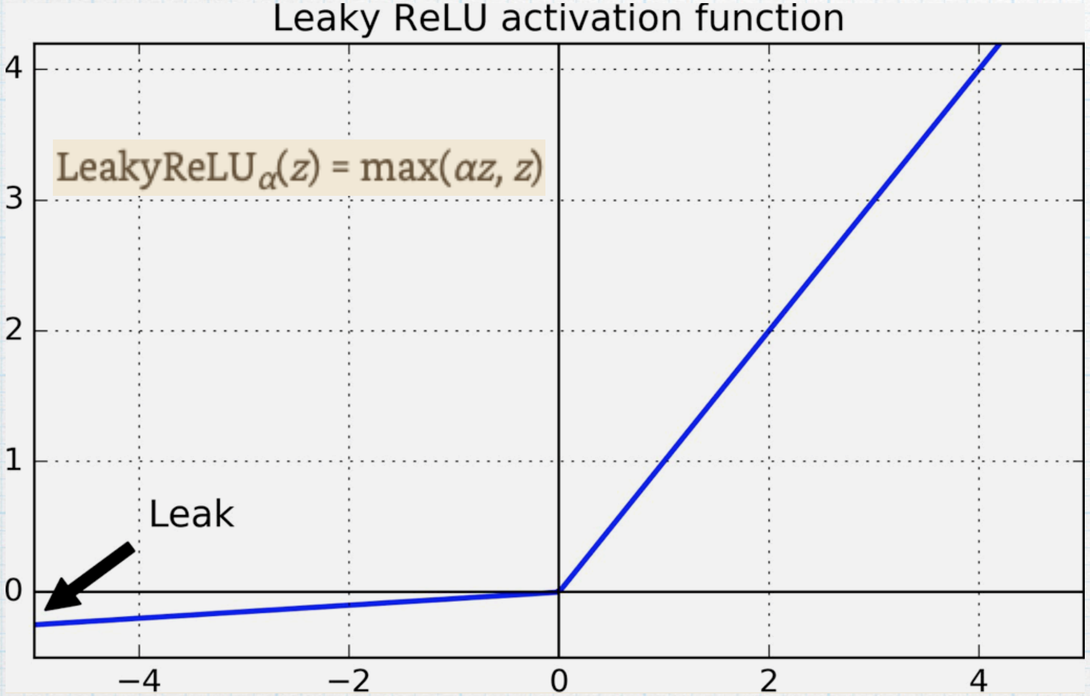


Activation functions

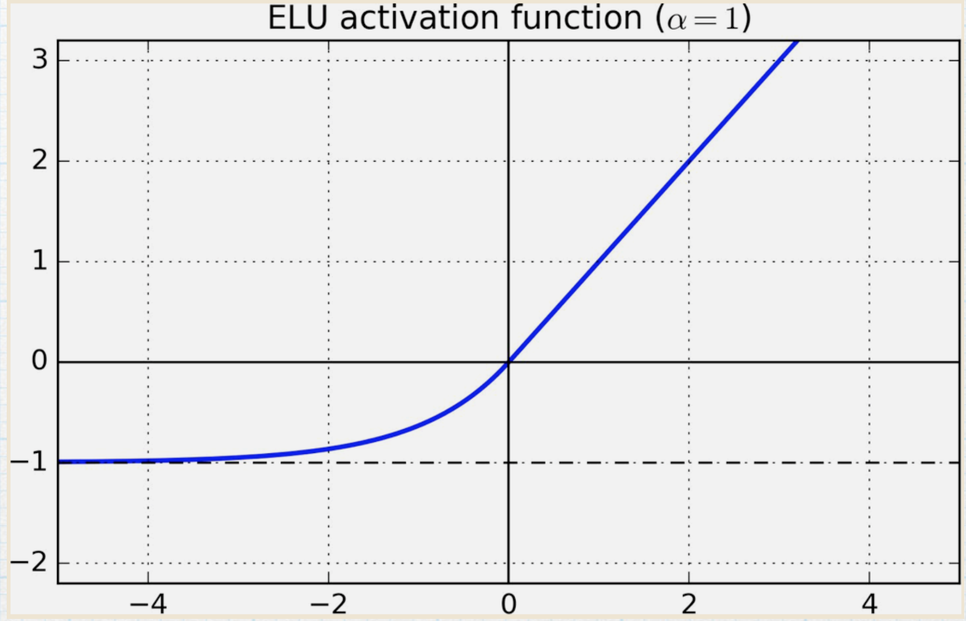
- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- ELU



Nonsaturating activation functions



$$\mathrm{ELU}_\alpha(z) = \begin{cases} \alpha(\exp(z) - 1) & \text{if } z < 0 \\ z & if\, z \geq 0 \end{cases}$$



* Leaky ReLU

* ELU

“Empirical Evaluation of Rectified Activations in Convolution Network,” B. Xu et al. (2015).

Backpropagation

* For each training instance the backpropagation algorithm
  * first makes a prediction (forward pass),
  * measures the error, then
  * goes through each layer in reverse to measure the error contribution from each connection (reverse pass), and
  * finally slightly tweaks the connection weights to reduce the error (Gradient Descent Step).

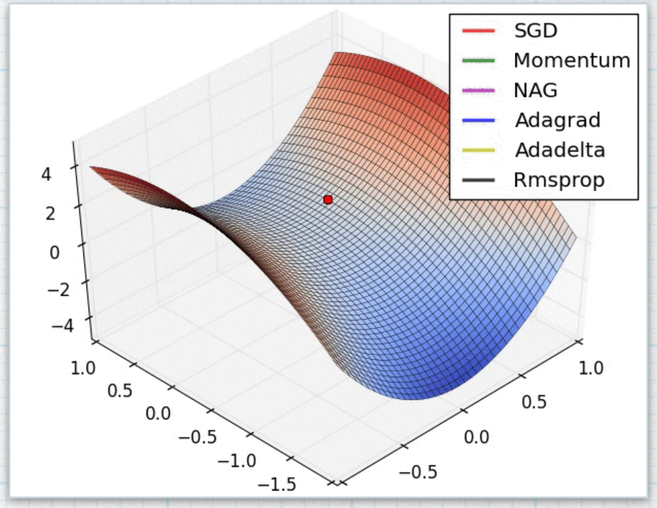
Batch normalization

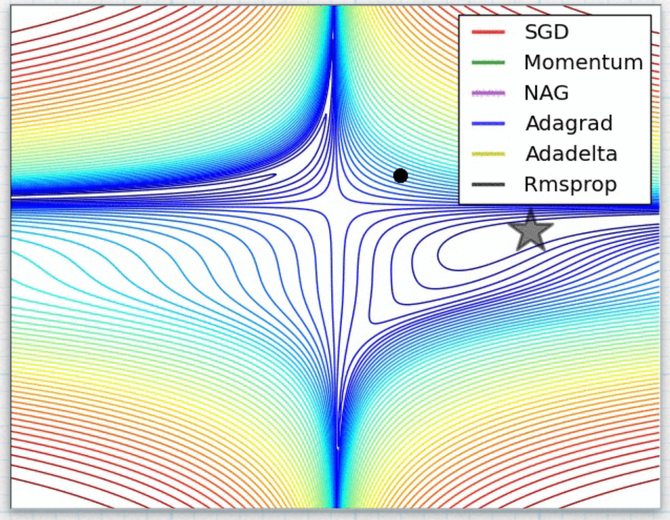
* just before the activation function of each layer,
* simply zero-centering and normalizing the inputs,
* then scaling and shifting the result using two new parameters per layer (one for scaling, the other for shifting)

“Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” S. Ioffe and C. Szegedy (2015).

Faster optimizers

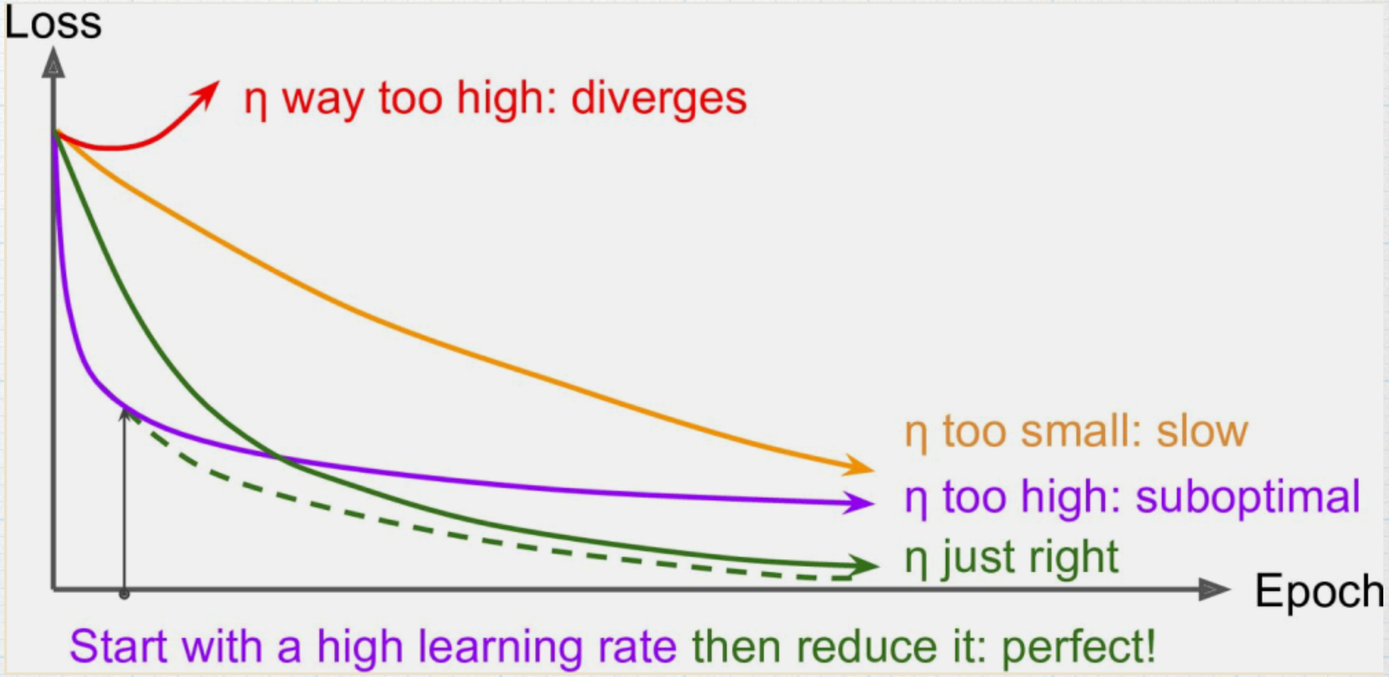
- Momentum optimization
- Nesterov accelerated gradient
- Adagrad
- Rmsprop
- Adam optimization
- Learning rate scheduling



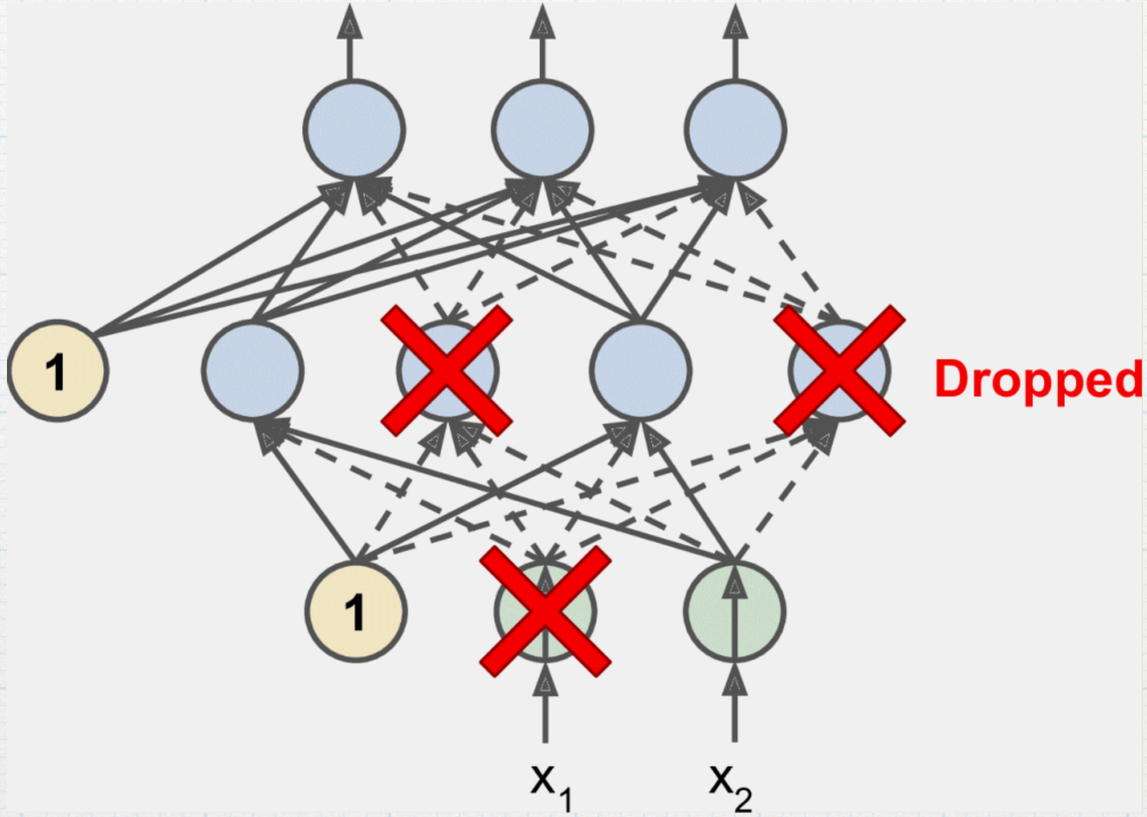


Learning rate scheduling

- constant
- Piecewise constant
- Performance scheduling
- Exponential scheduling
- Power scheduling



Dropout



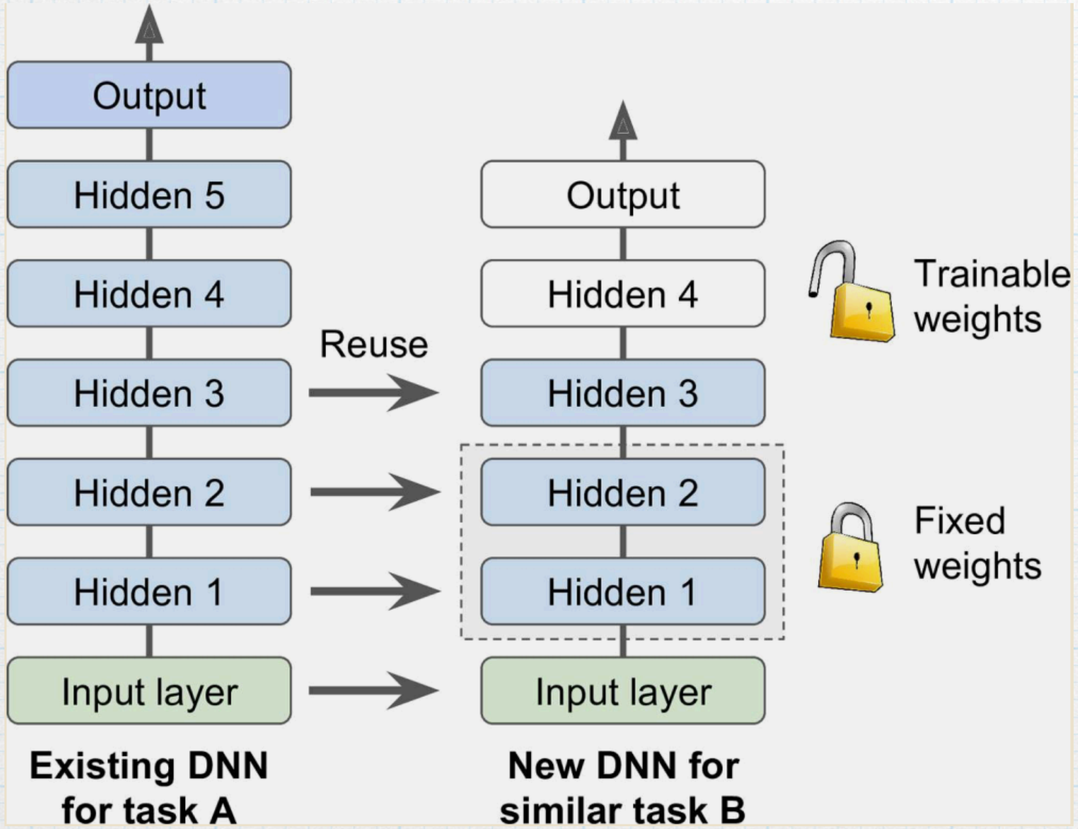
"Improving neural networks by preventing co-adaptation of feature detectors," G. Hinton et al. (2012).

"Dropout: A Simple Way to Prevent Neural Networks from Overfitting," N. Srivastava et al. (2014).

Reusing Pretrained layers

* Freezing the lower layers
* Tweaking, dropping , or replacing the upper layers
* Unsupervized pretraining

Freezing the lower layers

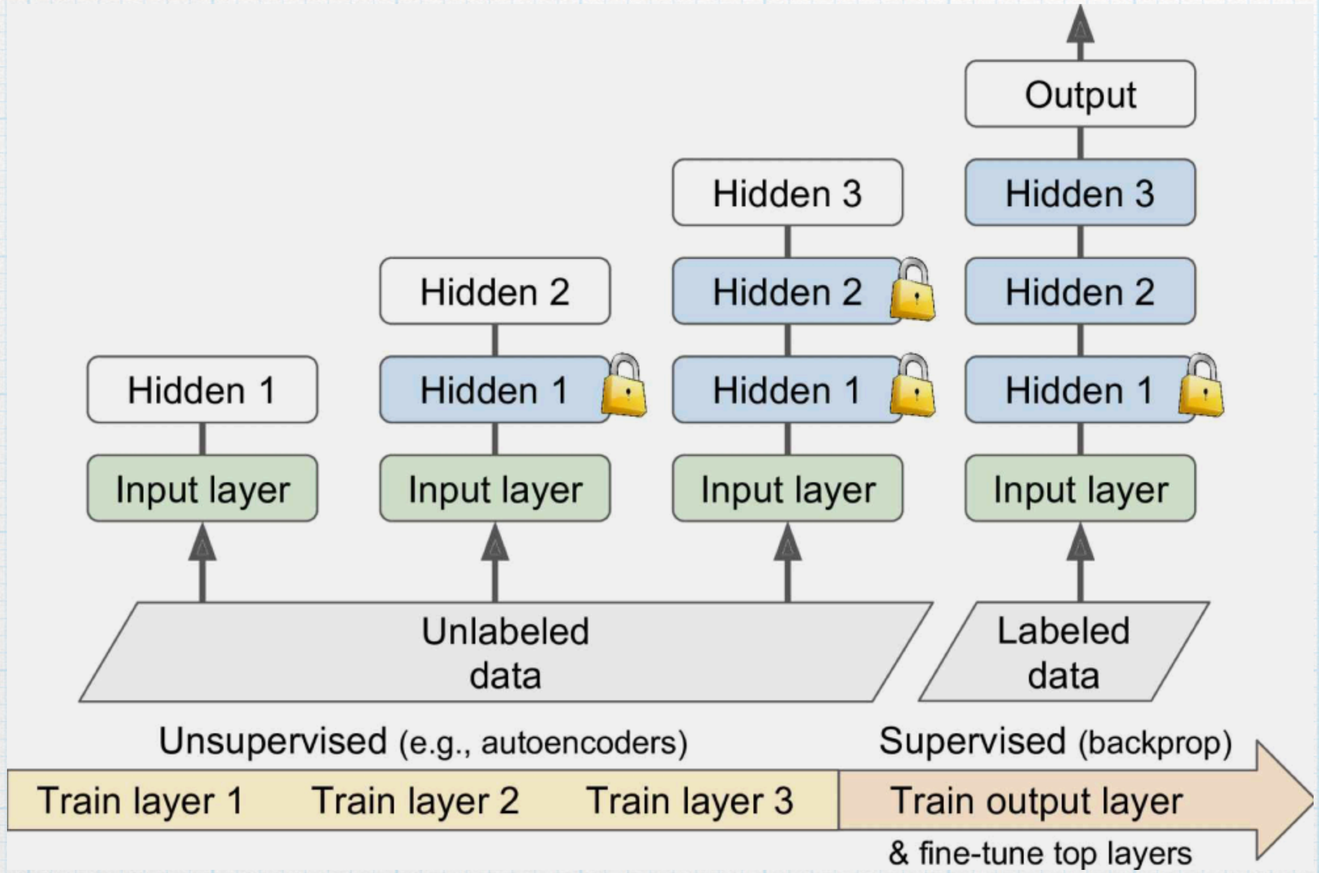


- Cache outputs of the Frozen layers!
- Transfer learning works well if the inputs have similar low-level features

Tweaking, dropping , or replacing the upper layers

- The higher the hidden layer, the more task-specific features are
- Try to gradually unfreeze a layer at a time and retrain a model
- More training data more layers you can unfreeze
- Last resort: drop few top layers

Unsupervized pretraining



* Experiments have shown that pretraining helps mainly
  * as a regularizer and
  * as an aid to optimization to some extent

Assignment: 5 min.

* Write down a formula for Gradient Descent Update of model parameters
* Write down formulas used in forward propagation to get output of a feed-forward deep neural network

Deep Learning. Limitations

* Approximation power
  * Given enough hidden units, NN can approximate any function

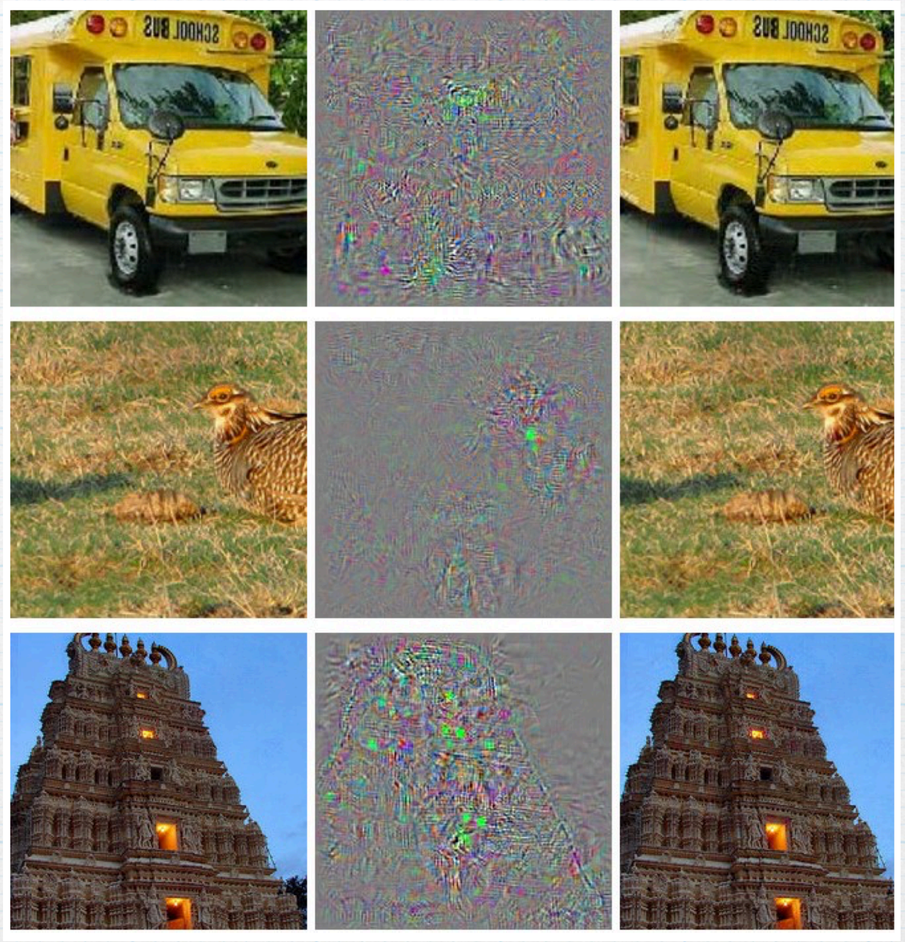
UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

* experiments establish that state-of-the-art convolutional networks for image classification trained with stochastic gradient methods easily fit a **random labeling** of the training data.

https://arxiv.org/pdf/1611.03530.pdf

Adversarial Attacks

- 50 Shades of an Ostrich



Adversarial Attacks

* A rifle from any angle
* https://www.labsix.org/media/2017/10/31/video.mp4

Challenges / Open questions

- There are numerous open questions in deep learning.
  - When to use deep learning?
  - What makes a good representation?
  - How to scale algorithms to large-scale problems?
  - Is unsupervised learning the future of deep learning?
  - How to model uncertainty?

Summary

* Here comes your summary from the class

Homework (in Moodle)

* Write a short list of your personal goals in the course:
  * What knowledge you'd like to obtain?
  * What skill you'd like to learn?

Videos

* About overfitting
  * https://youtu.be/DQWI1kvmwRg
* Bullshit about Neural ODEs:
  * https://www.youtube.com/watch?v=YZ-_E7A3V2w