Technische Universität München

Hands-on Deep Learning for Computer Vision and Biomedicine

Practical Course

Summer Semester 2019

Vladimir Golkov
Qadeer Khan
Patrick Wenzel
Prof. Dr. Daniel Cremers

Learning Goals

- Theory & Practice:
  - Basics <u>and</u> advanced techniques
- Deep learning craftsmanship
  - Understanding practical problems
  - Designing solutions
- Practical project experience with real-world open problems
  - The projects are geared towards producing scientific publications
  - Topics include biomedicine, autonomous driving, etc.
- Presentation skills

Prerequisites

- Good programming skills
  - Python
  - Array programming in NumPy (or Matlab or similar)
  - Deep learning framework (e.g. PyTorch or TensorFlow)
- Curiosity
- Passion for mathematics
- Time for regular hard work
- Proactivity
  - Project success depends on a two-way communication between the students and supervisors
  - If you expect to just passively receive detailed instructions and directions rather than also establishing communication and asking questions, then this practical course is not for you
- Prior knowledge in deep learning (and for some projects in computer vision) is **required**
- Prior knowledge in biomedicine is **not** required
  - You will learn from your supervisor

Structure of Practical Course

- Three lectures in the beginning of the semester (Tuesday 2-4pm)
- Practical project
  - Each student gets assigned to one project (or a few very similar projects)
  - Each project consists of a “pool” of tasks
    - Requirements elicitation and agreeing upon solutions
  - Usually 1 or 2 students per task
  - Most projects: Python, NumPy, deep learning frameworks (mostly PyTorch)
  - Access to computers and GPUs in Garching and remotely
  - Deep learning requires early and regular efforts
  - Regular communication with supervisors (important for progress of learning and project success)
    - Depending on the project, there may be a short weekly meeting/presentation discussing progress and challenges
    - Emailing skills are also important
- Final presentations
  - Groups can learn from each other and discuss
  - Presentation dates will be determined by voting (end of semester)

Next Steps

- 8-13 February: Apply for a place at https://matching.in.tum.de/
- There are many applicants
- Sending info about yourself is crucial to get matched and to get assigned a project with appropriate difficulty
- Email us info ideally several days before you fill in your priorities on the matching website, and at the very latest until 15 February:
  - Your programming skills
  - Some code you wrote in any context
  - Your interests, learning goals
  - Your courses, all grade transcripts
- If you require project info in advance, contact us
- If you want to propose own projects ideas, they should be discussed with us until 15 February
- Places in the course will be assigned on 20 February

After 20 February

- Projects will be announced, discussed and assigned as soon as possible
- We will consider your preferences, and also our knowledge about which of your preferred projects match your programming skills

Most Imporantly

- Most importantly:
  - Read project descriptions very carefully, ask as soon as possible whenever something is unclear, select projects wisely
  - Follow all announced recommendations

Other Options

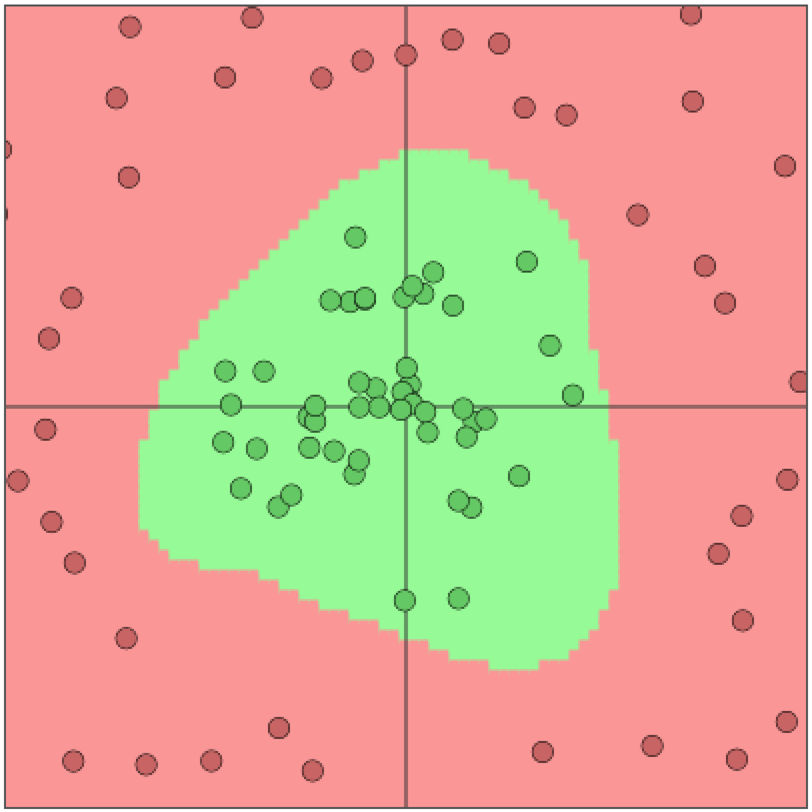
- If you don't get a place in the practical course:
  - Email us, enter the waiting list
  - Apply in subsequent semesters

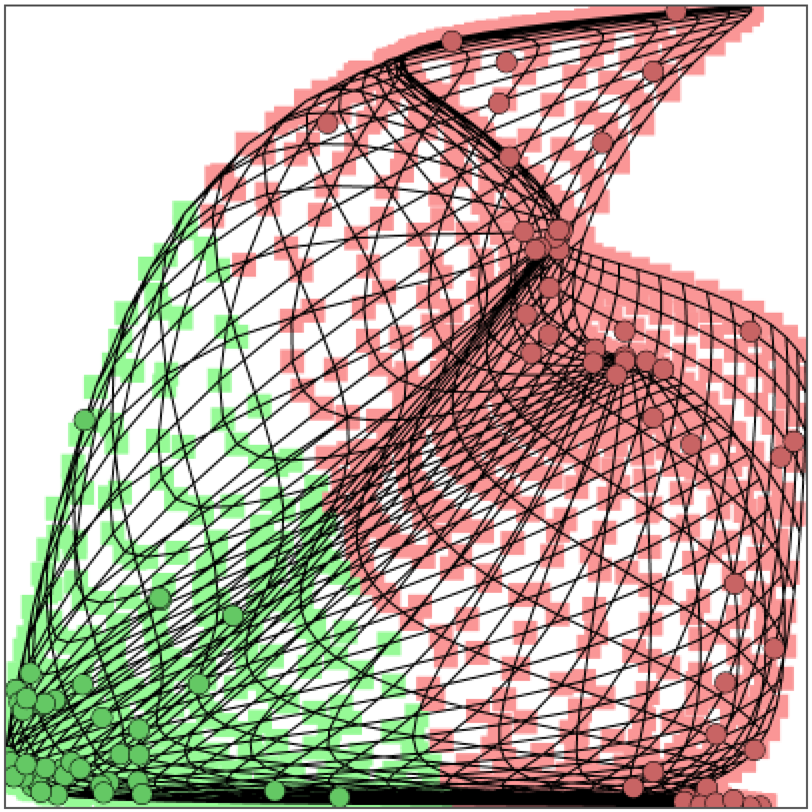
- Whether you get a place or not, also consider applying for:
  - Bachelor Thesis
  - Master Thesis
  - Interdisciplinary Project
  - Guided Research
  - etc.

Literature

- Christopher M. Bishop: "Pattern Recognition and Machine Learning", Springer, 2006. (Skim the Chapters 1, 2, 5.)
- http://www.deeplearningbook.org/
- http://neuralnetworksanddeeplearning.com/
- http://www.mlyearning.org/
- NumPy: Advanced Array Indexing
  https://docs.scipy.org/doc/numpy/reference/arrays.indexing.html

Nonlinear Coordinate Transformation

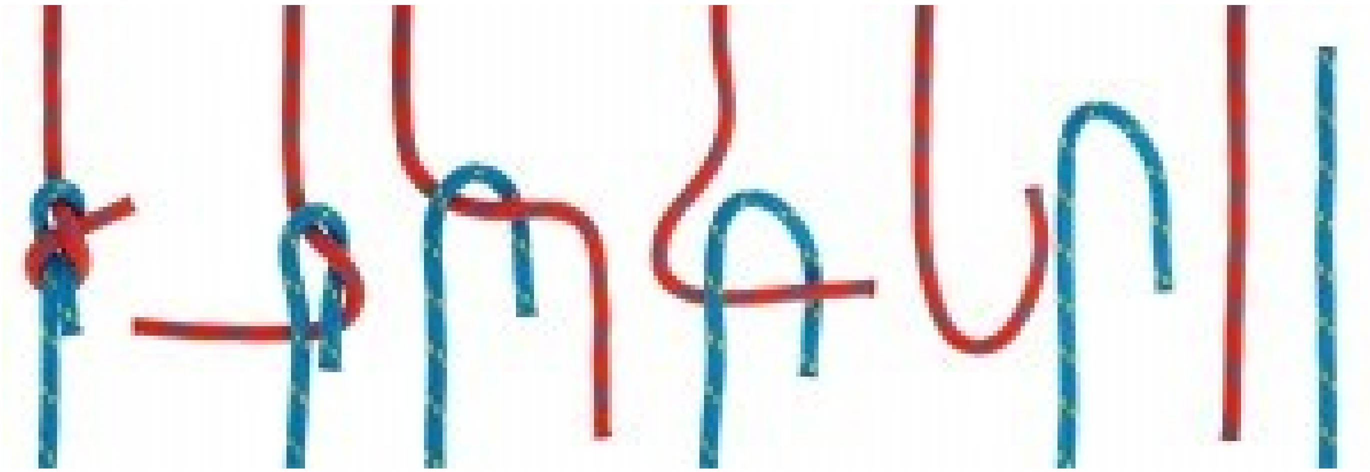




http://cs.stanford.edu/people/karpathy/convnetjs/

Dimensionality may change! (Here: 2D to 2D)

Deep Neural Network: Sequence of Many Simple Nonlinear Coordinate Transformations that “disentangle” the data (by transforming the entire coordinate system)



Data is sparse (almost lower-dimensional)

Linear separation of red and blue classes

Fully-Connected Layer a.k.a. Dense Layer

$x^{(0)}$ is input feature vector for neural network (one sample).

$x^{(L)}$ is output vector of neural network with L layers.

Layer number l has:

- Inputs (usually $x^{(l-1)}$, i.e. outputs of layer number $l-1$)
- Weight matrix $W^{(l)}$, bias vector $b^{(l)}$ - both trained (e.g. with stochastic gradient descent) such that network output $x^{(L)}$ for the training samples minimizes some objective (loss)
- Nonlinearity s_l (fixed in advance, for example $\mathrm{ReLU}(z) \coloneqq \max\{0, z\}$)
- Output $x^{(l)}$ of layer l

Transformation from $x^{(l-1)}$ to $x^{(l)}$ performed by layer l:

$$x^{(l)} = s_l\left(W^{(l)} x^{(l-1)} + b^{(l)}\right)$$

One Layer: Graphical Representation

$$W^{(l)} = \begin{pmatrix} 0 & 0.1 & -1 \\ -0.2 & 0 & 1 \end{pmatrix}$$

$$x^{(l-1)} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$$

$$b^{(l)} = \begin{pmatrix} 0 \\ 1.2 \end{pmatrix}$$

$$W^{(l)} x^{(l-1)} + b^{(l)} =$$

$$= \begin{pmatrix} 0 \cdot 1 + 0.1 \cdot 2 - 1 \cdot 3 + 0 \\ -0.2 \cdot 1 + 0 \cdot 2 + 1 \cdot 3 + 1.2 \end{pmatrix}$$

$$= \begin{pmatrix} -2.8 \\ 4 \end{pmatrix}$$