Hands-on Deep Learning for Computer Vision and Biomedicine

Practical Course
Winter Semester 2017/2018

Vladimir Golkov
Prof. Dr. Daniel Cremers
Learning Goals

• Theory & Practice:
  – Basics, machine learning, deep learning
  – Standard architectures (e.g. convolutional networks)
  – Standard applications (e.g. supervised learning)
  – Advanced architectures, advanced applications

• Deep learning craftsmanship
  – Understanding real data and practical problems, finding solutions
  – Designing architectures, choosing/designing loss functions
  – Designing the training procedures
  – Tuning hyperparameters
  – Evaluating and understanding results, iterating, drawing conclusions

• Practical project experience with real-world open problems
  – The projects are geared towards scientific publications

• Presentation skills
Structure of Practical Course

• Three lectures in the beginning of the semester (Tuesday 2-4pm)
• Practical project
  – Own project for each group
  – 1 or 2 students per group
  – Most projects: Python, NumPy, deep learning frameworks
  – You will get access to computers and GPUs in Garching and remotely
  – Deep learning requires early and regular attention
  – Regular discussions with supervisors (important for progress of learning and project success)
• Final presentations
  – Introduction, Methods, Results, Conclusions
  – 20min presentation + 10min Q&A
  – Groups can learn from each other and discuss
  – Presentation dates will be determined by voting (end of semester)
Next Steps

• 14-19 July: 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 until 21 July, e.g.:
  – Your programming skills
  – Some code you wrote in any context
  – Your interests, learning goals
  – Your courses, grades
• If you require project info in advance, contact us
• If you want to propose own projects ideas, they should be discussed with us until 21 July
• Places in the course will be assigned on 26 July
After 26 July

• If you get a place in matching system, you have two options:
  – Either form a group of two and submit project preferences together
  – Or submit your individual project preferences, and whether you have a preference for being assigned a partner, or working on your own, or both is fine

• Projects will be announced, discussed and assigned as soon as possible

• Example of project preferences that a student or group sends us:
  1. Project “C” (most preferred)
  2. Project “T” or “U”
  3. Project “A”
  4. Project “G”

• We will consider your preferences, and also our knowledge about which of your preferred projects match your programming skills
Other Options

• Also consider applying for:
  – Bachelor Thesis
  – Master Thesis
  – Interdisciplinary Project
  – Guided Research

• If you don’t get a place in the practical course:
  – Email us, enter the waiting list
  – Apply in subsequent semesters
  – Apply for Thesis, Interdisciplinary Project, Guided Research
Prerequisites

• Good programming skills
  – Python
  – Array programming in NumPy (or Matlab or similar)
• Curiosity
• Time for regular hard work

• Prior knowledge (deep learning, computer vision, biomedicine) is not required
  – You will learn from your supervisors
• But good programming skills are important
Literature

• http://www.deeplearningbook.org/
• http://neuralnetworksanddeeplearning.com/
• 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)
**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 $\text{ReLU}(z) := \max\{0, z\}$)
- 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

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

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

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

\[ W^{(l)} x^{(l-1)} + b^{(l)} = \]

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

\[ = \begin{pmatrix} -2.8 \\ 4 \end{pmatrix} \]