A Deep Learning Bootcamp for Engineering & Management Students

Lukas Lodes [*1] **Alexander Schiendorfer** [*1]

Abstract

Students from engineering & management with a focus on vocational activities such as machining or accounting tend to lack the necessary computer science foundations to build, appreciate, and evaluate machine learning solutions. However, they are likely going to have to identify and judge potential use cases in their careers in industrial practice. Therefore, we propose a guided three-day curriculum that goes all the way from manual data inspection to the implementation of several models in Python, including several evaluation metrics. We focus on a computer vision task identifying traffic signs due to its conceptual simplicity and similarity to tasks in vision-based quality assurance. We share our material as OER as well as experiences from four offerings of the bootcamp.

1. Background

Machine learning and especially deep learning are getting increasingly important in classical engineering disciplines due to, e.g., computer vision applications such as automated quality inspection or anomaly detection in process data from machines. Graduates from engineering & management, industrial engineering have to be able to identify and prototypically implement promising ML use cases. At our university, the curriculum involves a fair amount of classical engineering (mechanics, manufacturing processes, CAD construction) and business (accounting, marketing) courses but the computer science background falls short with only two dedicated compulsory lectures.

We designed a short block event called "deep learning bootcamp" that is run over three days, eight hours each day, that walks the students through a hands-on project: recognizing traffic signs from images. The bootcamp has been offered four times so far in different degree programs.

[1]Institute AImotion Bavaria, Technische Hochschule Ingolstadt, Ingolstadt, Germany. Correspondence to: Alexander Schiendorfer <alexander.schiendorfer@thi.de>.

Proceedings of the 3^{rd} Teaching in Machine Learning Workshop, PMLR, 2022.

We share our positive and negative experiences and provide our materials as open education resources (OER) on https://doi.org/10.5281/zenodo.6984734.

We make the following assumptions about our audience

Computer-Operation-Level (COL): The command of basic computer science concepts (handling files programmatically in a shell environment, working a command line, connecting to a server via SSH, ...) is low.

Programming-Level (PL): Similarly, strong programming skills in a language such as Python or R cannot be assumed.

Application-Oriented (AO): The application perspective matters most. Students tend to work part-time in industrial environments (e.g., manufacturing companies) where they might get tasked to experiment with ML.

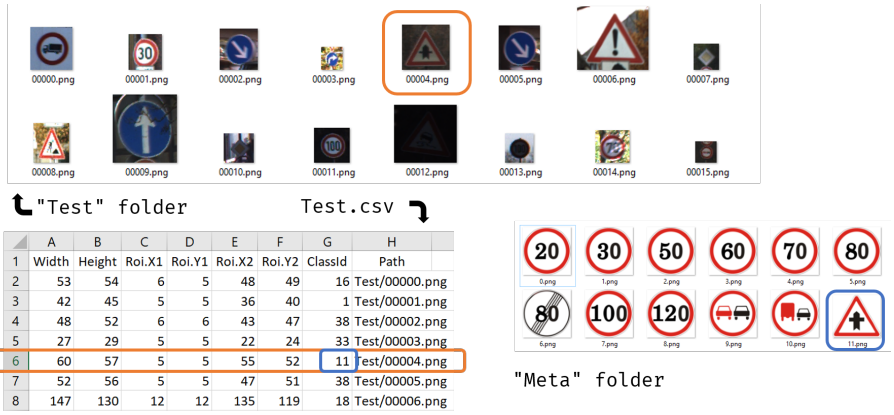
STEM-Background (STEM): The students have a sufficient mathematical foundation to understand the key concepts of deep learning, in particular, linear algebra and multivariate calculus.

The course has been offered four times, each time in a virtual manner via Zoom and in a blocked way. We hope that the provided notebooks are useful for ML teachers. For interactivity, we inserted a couple of demos that can be found in (NVIDIA, 2022) or (Google, 2022).

Related work Several papers propose related courses. (Müller et al., 2022) describes a course of similar duration (5 days) that introduces ML to PhD students in molecular biology research. (Acquaviva, 2022) presents concepts for teaching ML to physisicsts. The courses were much longer than ours, but similar experiences were made. (Shouman et al., 2022) also propose a semester-spanning event, but with similar elements to our course, e.g. a focus on practically relevant examples or block sessions with diverse content delivery. By contrast, (Garcia-Algarra, 2021) targets an audience that lack the STEM background we presuppose.

2. Learning goals

Contrary to many classical ML courses, our main goal is – besides teaching the inner workings of elementary deep



	A	B	C	D	E	F	G	H
1	Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
2	53	54	6	5	48	49	16	Test/00000.png
3	42	45	5	5	36	40	1	Test/00001.png
4	48	52	6	6	43	47	38	Test/00002.png
5	27	29	5	5	22	24	33	Test/00003.png
6	60	57	5	5	55	52	11	Test/00004.png
7	52	56	5	5	47	51	38	Test/00005.png
8	147	130	12	12	135	119	18	Test/00006.png

Figure 1. The data for the GTSRB are not readily available in Keras but can be inspected using tools familiar to the students (Windows Explorer, Microsoft Excel), taking into account COL and PL.

learning models and algorithms – to convey a sense of the tasks that need to be done when approaching a novel problem: (i) data acquisition, (ii) exploratory data analysis, (iii) framing of the learning problem (e.g., if supervised, what are the inputs and targets?) (iv) gathering metrics and evaluating the trained models. It also delibaretly omits many important ML algorithms on tabular data such as gradient boosting to (1) foster motivation for "tangible" use cases in industrial practice such as image recognition (2) show that these use case are within reach for the students due to the impressive quality of software frameworks and modern hardware.

The main learning goals result from the designated audience. After passing the bootcamp students should:

i) Understand that preparing and inspecting the data is a key challenge.

ii) Know how to implement an off-the-shelf convolutional network for image recognition using Keras (Chollet et al., 2015) and scikit-learn (Pedregosa et al., 2011).

iii) Have a coarse understanding for the inner workings of a deep learning system (i.e., autograd, gradient descent, hyperparameters such as the learning rate)

iv) Be able to identify problems that are suitable for transfer learning using pretrained nets as this is a standard approach when dealing with real-world datasets.

With the learning goals in mind, the overall goal of the course is to solve an image recognition task together with the participants – albeit one that is not simply available as a benchmark in Keras/TensorFlow rightaway (e.g., CIFAR10, Fashion MNIST, Imagenet, etc). Since the underlying dataset should keep the students' motivation high, we use the *German Traffic Sign Recognition Benchmark (GTSRB)* (Houben et al., 2013). The task is clear (see fig. 1), classifying a photo taken from a car into 43 classes that correspond to different traffic signs, but it involves a fair amount of data handling (cf. COL) since the 50 000 training and 10 000 test images are organized in a file directory tree and CSV files.

3. Contents

The bootcamp is divided into several blocks that each consist of a set of slides accompanied by a Jupyter notebook and various activation exercises. Besides, the students and teachers *together* complete a shared slideshow file. We used Google docs due to its capability of using *pseudonyms* for the students which made them more confortable in participating but any collaborative tool would do.

3.1. What is a neural network, what is deep learning?

Initially, most of the students in our audience tend to have no idea what a neural network is or what the difference between AI, ML, Deep learning and neural networks are. Therefore, the course starts with a brief introduction to those keywords, establishing that deep learning can be seen as a generalization from classical feedforward nets to more general algebraic circuits of differentiable modules that can be trained using gradient descent on loss functions (Russell & Norvig, 2020). We also include the broader perspective on AI research since the 1960s including symbolic approaches. At this point, we ask the students to write down in the shared slides *if they perceive AI as something positive or negative*, *what they know about AI/ML* and *whether they think it has an impact on their careers and lives*[1]. This will be compared with the results after three days. Since the focus is on deep learning, we then take a quick turn to the Tensorflow Playground (Google, 2022) to have the students see a neural net train as fast as possible in the course – this

[1]Click here for an example

usually happens within the first hour.

Using the playground, students can get a first grasp to concepts such as changing weights and the decision boundaries of a binary classifier evolving (and typically improving) over time. It is also helpful to stop the training process after a couple of epochs, inspect the loss and the visual result. This part is concluded by challenging the students to find the configuration with the smallest loss (highest accuracy resp.) and post screenshots thereof in the shared slides.

+ Gives a fast first impression of neural networks

+ Contains all the major components of a DL system

+ Allows for experimenting with provoked underfits / overfits to build intuition

- Tinkering with the playground may enforce the "wrong" priorities in first-learners: "I. as an ML engineer, must excel at deciding layers/neurons/activation functions" whereas this can often be automated or become less relevant when working with a large enough data set.

In terms of self-study tasks, we had the students watch the neural network series by 3Blue1Brown (Sanderson, 2017) after playing with the TensorFlow playground. This was accompanied by a clear call to action with questions in the shared slides that had to be filled out afterwards and discussed.

3.2. How is a neural network for image recognition programmed?

After the initial exposition to neural networks in the playground setting and video material, we focus on image recognition, keeping AO in mind. To show that the actual DL code can be, nowadays, fairly compressed using Keras or PyTorch, we start with a single-cell Jupyter notebook that trains a convolutional net on CIFAR10.

This segment can then be accompanied by some data exploration, plotting some images using `imshow` along with the target classes. We cover the basics of computer vision (e.g. how an image is represented as a $N \times M \times 3$ tensor). After training, a few calls to `model.predict()` reinforce the impression that the model "has actually learned something" if it produces some accurate predictions.

+ Students can train their first neural network in Python directly in Google Colab (PL), even on a GPU (COL)

+ No data handling required for the first contact, application is clear (AO)

- A ConvNet uses many concepts yet to be introduced but a simple feedforward net does not perform as well

- Cifar10 comes with a vector of normalized logits as outputs, they need to be converted to "human-readable" class names using `argmax` and a list (PL)

- Model predict needs a $(1, 32, 32, 3)$ tensor (including the batch dimension) which requires a `reshape` which is not intuitive for first-time learners (PL)

3.3. What does the GTSRB data look like?

But at the heart of the bootcamp lies our goal of solving a "fresh" image recognition task akin to what students would have to do if they collected their own data in their respective companies (e.g., by scraping or taking photos). We tend to spend more time on this data exploration every running of the course. And in this paper, we truly want to recommend the GTSRB data set for other similar courses:

(1) The problem is very easy to grasp and common in real life (even non-driver pedestrians need to understand traffic signs). Further, there are attractive use cases such as autonomous driving or driver assistance (AO).

(2) Training and test data are organized as PNG images in a ZIP file (see fig. 1). That means, students can browse through instances directly in Windows Explorer (or other gallery tools on a Mac or Unix system) without having to code (PL). This easily enables them to identify difficulties (illuminations, occlusions, similarities between classes such as 30/80 limit).

(3) The 43 classes are represented by "canonical" images in the *Meta* folder which also makes it very easy to work with class indices by image.

(4) The images have different sizes and the training images are sorted in folders according to their classes, which makes additional preprocessing steps necessary. This is very similar to real-world problems.

To conduct data exploration, we again formulate clear questions to address: *Is the dataset hard to learn for a computer? Where could be difficulties between classes? Are some instances harder to classify than others?* The responses are collected in the shared slides.

+ Easy data exploration without specialized tools (PL STEM)

+ Data set contains a non-trivial variety of instances, not all class overlaps are obvious!

+ Handling the files programmatically (e.g., extracting, converting PNGs to numpy arrays, resizing) requires additional code (which we provide in this notebook as a basis)

- Training and test images are organized differently, adding a bit of confusion

- CSV files contain additional information (region of interests, etc) which can be overwhelming (COL)

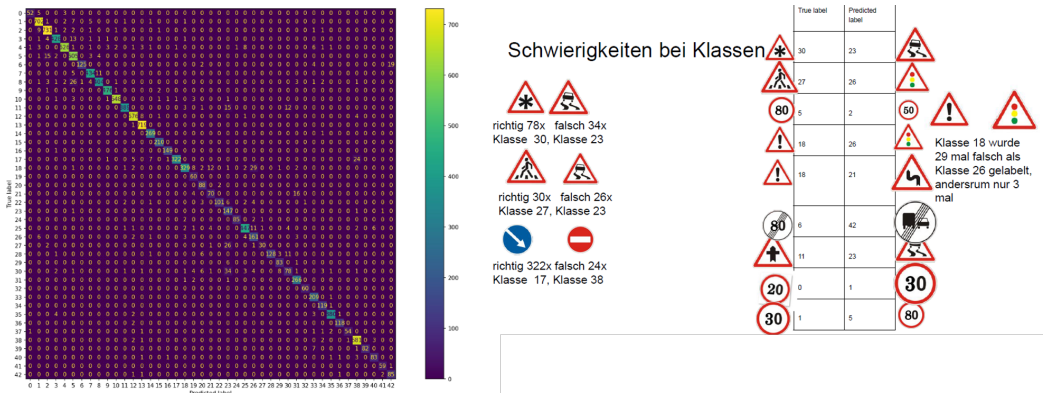


Figure 2. Some slides created and provided with courtesy by the students (in German) using a confusion matrix to determine classes that caused, well, confusion in the model. Besides traffic signs with obvious similarities (e.g. "crosswalk" and "risk of slipping/skidding"), they also observed confusion between a white arrow on blue background and a white line on red background

3.4. How is a neural network actually trained?

After having trained neural networks in the TensorFlow playground and Keras (so far only CIFAR10), the students have a rough understanding of the training mechanism which we then deepen. This draws on the sufficient mathematical foundations (STEM) although discussing partial derivatives of the loss function with respect to weights and biases (i.e. in the application context of neural networks) seemed to make the otherwise abstract concepts of multivariate calculus more tangible to the students. To make gradient descent more intuitive (and appealing), we include a session in the loss landscape explorer (Ideami, 2020), where we task the students to experiment with different initial conditions, learning rates, and levels of momentum. The most interesting screenshots are stored in the shared slides. This part of the course follows a standard exposition to auto-differentiation and gradient descent.

3.5. How to put it all together and do experiments?

Based on the code for training a model on CIFAR10 and the code to load the GTSB data programmatically into a running Jupyter notebook, the students are putting the pieces together – mostly on their own. That involves taking care of the input dimensions (we resize the GTSB images to $45 \times 45 \times 3$ instead of $32 \times 32 \times 3$), encapsulating the model creation for fully connected, convolutional, and pretrained convolutional into functions as well as evaluating the models in terms of training duration and accuracy. We leave enough time for the students to build up these parts by themselves or in groups. An exemplary solution can be found here.

4. Experiences and Results

The predetermined duration of three days is very limited for such a block course. Especially if the students lack basic programming skills in Python, it is hard and distracting for them to develop the models and experiments by themselves. Hence, we offered substantial code bases for students to fill in the gaps – which might bore them. Nevertheless, the fact that they actually built a system that can do something as useful as recognizing traffic signs within three days was reported as motivating.

Despite Python and Keras being beginner friendly, it may still be overwhelming. Moving the data exploration and design of the DL system to tools they are familiar with – Windows Explorer and Excel – turned out to be beneficial. For such an audience, we also consider experimenting with graphical tools offered by Microsoft or Matlab.

In terms of videocalling tools such as Zoom or Teams, while we noticed the typical downsides (less interaction and feedback) they also have a couple of advantages. It was easier to catch breaks for both the students and the lecturers which is a critical factor during eight-hour-days. Additionally, most participants had a computer setup they are comfortable with at home. This avoids limitations of university labs, e.g. not enough power outlets, unstable WIFI or projectors with low resolution. Especially the screen sharing was a big advantage. The shared slides led to a more productive cooperation than leaving the organization up to the students, as evidenced by the results they created in fig. 2.

Finally, the students' impressions about DL and AI changed tremendously throughout all offerings. While the mechanics of training deep networks can take away from the "magic" halo that surrounds the typical media coverage, working with software trained purely from data still fascinated the participants. We hope that our experiences and learning resources are useful for the community of ML (and here DL) teachers.

References

Acquaviva, V. Teaching machine learning for the physical sciences: A summary of lessons learned and challenges. In Kinnaird, K. M., Steinbach, P., and Guhr, O. (eds.), *Proceedings of the Second Teaching Machine Learning and Artificial Intelligence Workshop*, volume 170 of *Proceedings of Machine Learning Research*, pp. 35–39. PMLR, 08–13 Sep 2022. URL https://proceedings.mlr.press/v170/acquaviva22a.html.

Chollet, F. et al. Keras. https://keras.io, 2015.

Garcia-Algarra, J. Introductory machine learning for non stem students. In Bischl, B., Guhr, O., Seibold, H., and Steinbach, P. (eds.), *Proceedings of the First Teaching Machine Learning and Artificial Intelligence Workshop*, volume 141 of *Proceedings of Machine Learning Research*, pp. 7–10. PMLR, 14 Sep 2021. URL https://proceedings.mlr.press/v141/garcia-algarra21a.html.

Google. AI Experiments - Experiments with Google, 2022. URL https://experiments.withgoogle.com/collection/ai.

Houben, S., Stallkamp, J., Salmen, J., Schlipsing, M., and Igel, C. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

Ideami, J. Loss Landscape Explorer, 2020. URL https://losslandscape.com/explorer.

Müller, R., Fasemore, A. M., Elhossary, M., and Förstner, K. U. A lesson for teaching fundamental machine learning concepts and skills to molecular biologists. In Kinnaird, K. M., Steinbach, P., and Guhr, O. (eds.), *Proceedings of the Second Teaching Machine Learning and Artificial Intelligence Workshop*, volume 170 of *Proceedings of Machine Learning Research*, pp. 68–72. PMLR, 08–13 Sep 2022. URL https://proceedings.mlr.press/v170/muller22a.html.

NVIDIA. NVIDIA AI Demos, 2022. URL https://www.nvidia.com/en-us/research/ai-demos/.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Russell, S. and Norvig, P. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020. ISBN 9780134610993.

Sanderson, G. Neural Networks, 2017. URL https://www.3blue1brown.com/topics/neural-networks.

Shouman, O., Fuchs, S., and Wittges, H. Experiences from teaching practical machine learning courses to master's students with mixed backgrounds. In Kinnaird, K. M., Steinbach, P., and Guhr, O. (eds.), *Proceedings of the Second Teaching Machine Learning and Artificial Intelligence Workshop*, volume 170 of *Proceedings of Machine Learning Research*, pp. 62–67. PMLR, 08–13 Sep 2022. URL https://proceedings.mlr.press/v170/shouman22a.html.