Autonomous Systems (WS25/26)

Team Projects | November 14th, 2025

Project 1: Unsupervised Learning

As an alternative to image-based regression, we can learn a potentially more efficient feature representation of the input using unsupervised learning.

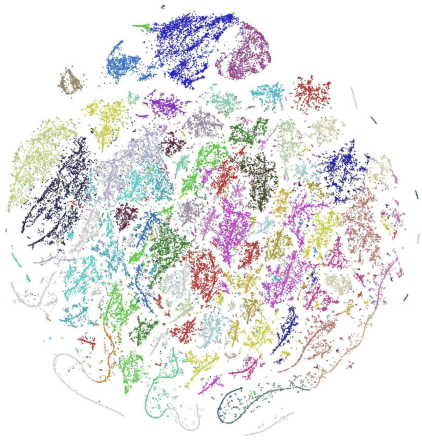

Figure 1: Feature Representation.

(a) Introduce the concept of unsupervised learning.

(b) Describe unsupervised learning using auto-encoders (not: *variational* auto-encoders).

(c) Implement a convolutional auto-encoder that learns a new feature representation of your image dataset. *You can use a deep learning framework like PyTorch to define and optimize the network architecture, but the implementation must be your own.*

(d) Experiment with the dimensionality of the new feature representation and visualize different latent codes in 2D using t-SNE.[1] Interpret the results.

(e) Compare the performance of your car trained with (i) raw images and (ii) the new feature representation.

[1] https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf

Project 2: Backpropagation on Directed Acyclic Graphs

Basic automatic differentiation makes the assumption that the output of a node in a computation graph is connected to a single successor node. Let $\mathrm{FC}(D_o)$ denote a fully-connected layer with D_o output dimensions. Similarly, let $\mathrm{conv}(C_o, s)$ denote a padded convolution with C_o output channels and an $s \times s$ kernel matrix. The input dimensions/channels are assumed to be inferred from the input volume. Given an input image $x \in \mathbb{R}^{224\times224\times3}$, consider the following network architecture f:

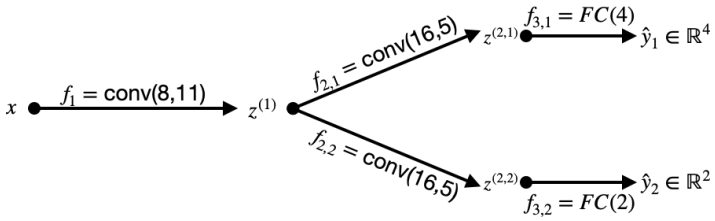

Figure 2: Network Architecture f.

(a) Calculate the gradient of a convolution $\mathrm{conv}(C_o, s)$ for an input of size $d_1 \times d_2 \times C_i$. Can you write the gradient in an elegant form?

(b) Adapt the basic automatic differentiation algorithm such that it works for computation graphs with multiple successor nodes (e.g., node $z^{(1)}$ in Figure **??**).

(c) Assume a mean squared error loss $\mathcal{L}_{\mathrm{MSE}}(y, \hat{y}) = \|y - \hat{y}\|^2$ on $\hat{y}_1$ and $\hat{y}_2$. Implement the adapted algorithm to allow parameter learning in f.

Your implementation must be from scratch and not use any third-party libraries (exception: low-level functions in `numpy`, `scipy`*). It is acceptable, if the algorithm only works for f and not for an arbitrary network architecture.*

(d) Given an input image $x \in \mathbb{R}^{224\times224\times3}$ of your choice, use your implementation to learn f such that it maps x to a bounding box $y_1 = (10, 20, 30, 40) \in \mathbb{R}^4$ and control parameters $y_2 = (5, 6) \in \mathbb{R}^2$.

Project 3: Virtual Cameras

In the theory sessions on camera intrinsics/extrinsics we learned how to use a calibrated camera together with a ground plane to reproject a captured image into a virtual camera.

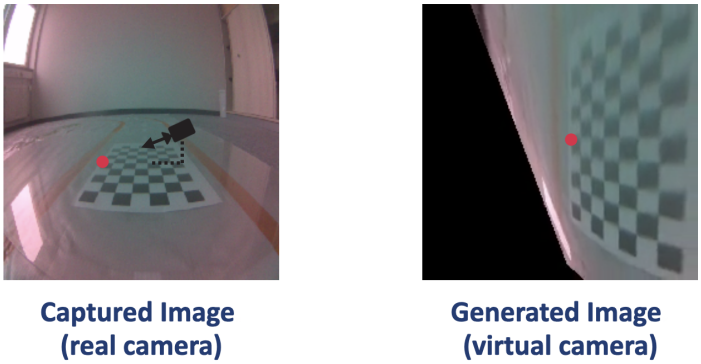

Figure 3: Virtual Cameras.

(a) Use OpenCV[2] to calibrate the camera on your car and take a photo I of the race track.

(b) Compute the world coordinates $p \in \mathbb{R}^3$ of the image coordinates $I(112, 112) \in \mathbb{R}^2$.

(c) Construct a virtual camera C_{virtual} that is positioned $20\,cm$ above p and looks at p.

(d) Reproject I into C_{virtual} and discuss your result and the geometric mathematics required to compute it.

[2] https://opencv.org

Project 4: Object Detection

In an upcoming theory session on object detection, we are going to discuss the YOLO framework.

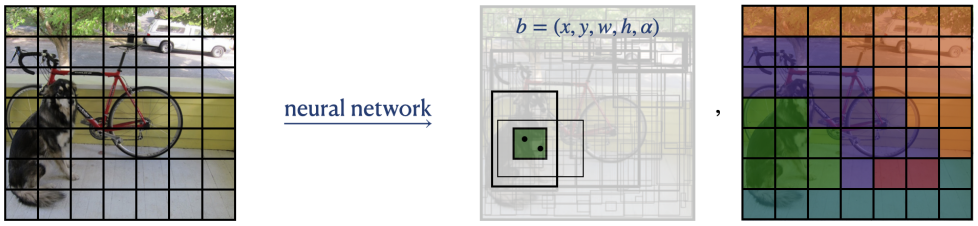

Figure 4: YOLO Framework.

(a) Implement YOLO.

You can use a deep learning framework like PyTorch to define and optimize the network architecture, but the implementation must be your own.

(b) Place a stop sign on the race track and take a photo x of it with the camera on your car. The native resolution of the camera is 224×224, so scaling by a factor of 2 results exactly in the input size required by YOLO.

(c) Label the image x with a ground truth bounding box $y = (\text{row}, \text{col}, \text{width}, \text{height})$ and train your implementation with the single-instance dataset $\mathcal{D} = \{(x, y)\}$.

(d) Demonstrate that you can detect the stop sign by visualizing the image x together with the ground truth bounding box y and your prediction $\hat{y}$.

(e) Evaluate the robustness of your object detection framework.

(e.1) Place the stop sign in a different location.

(e.2) Use a different stop sign (e.g., one with a scratch).

(e.3) Place two more stop signs in the scene.

(e.4) Adjust the brightness of the input image.

(e.5) Rotate and/or translate the input image.

At which point does your model break?

Project 5: Reinforcement Learning

Reinforcement Learning is defined as a machine learning method that is concerned with how software agents should take actions in an environment. The book by Barto and Sutton has become a standard[3].

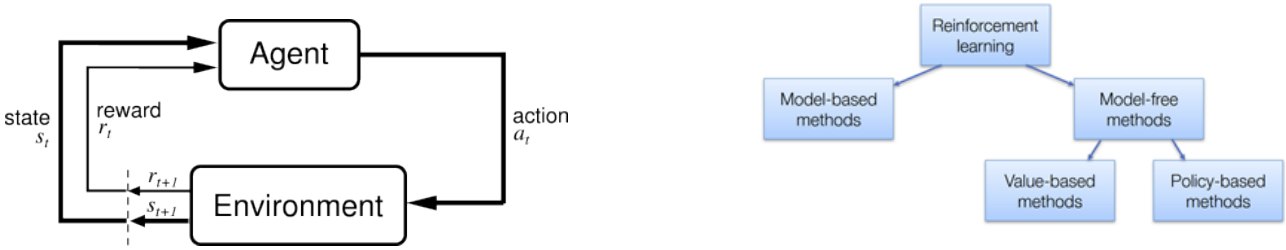

Figure 5: RL Agent environment and Algorithms

(a) Introduce the concept of reinforcement learning.

(b) Shortly describe Model-Based and Model-Free (Value-Based and Policy-Based) algorithms and dicuss closer Q-Learning and the Bellman Equations

(c) Demonstrate how RL works by e.g. the "Taxi Cab Problem"[4]

(d) Train the JetRacer with e.g. Stable Baselines (or OpenAi Gym)
→ There is students work availabe from here[5] and here[6]

Remark: Do not only explain how the algorithms work but also indicate pros and cons and the fields of application.

[3] http://incompleteideas.net/book/ebook/the-book.html
[4] https://medium.com/data-science/reinforcement-learning-teach-a-taxi-cab-to-drive-around-with-q-learning-9913e611028f
[5] https://confluence.student.fiw.fhws.de:8443/spaces/MFES2/pages/140116068/Presentations
[6] https://confluence.student.fiw.fhws.de:8443/spaces/SS2022/pages/77041415/Seminar+SmS+2022

Project 6: Supervised Imitation Learning

When having a look at the Waveshare Wiki[7] the proposed (supervised) learning works as following: "... put the mobile robot on the track ... control the car to run along the track, every time you move a small position, use the mouse to move to the ideal running path of the car in the picture and click to save the picture ..." see fig. 6. ("After collecting the data, select the value of epochs as 10, and then click train to train for 10 rounds.")

Figure 6: Waveshare Wiki: SL with JatRacer

A diverent approach is proposed by NVIDIA in the publication: "End to End Learning for Self-Driving Cars"[8]. In brief: "...We trained a convolutional neural network (CNN) to map raw pixels from a single front-facing camera directly to steering commands. This end-to-end approach proved surprisingly powerful ..."

(a) Present the ideas of of the publication. How do they differ from the Waveshare approach?

(b) Adopt the solution by NVIDIA to the JetRacer Pro and evaluate it on the given track.
→ There is students work (presentation and elaboration) availabe from here[9]

[7] https://www.waveshare.com/wiki/JetRacer$_{Pro_AI_Kit}$

[8] https://arxiv.org/abs/1604.07316

[9] https://confluence.student.fiw.fhws.de:8443/spaces/SS24/pages/155516955/Seminar+SmS+2024+Startseite