Intro to Deep Learning (Fall 2025) – Final Project

The final project is designed to emphasize your **practical understanding** of the material learned throughout the course.
Each group must consist of **2–3 students** (exceptions may be approved in special cases).

We offer **five predefined project options**, as well as an additional option to **propose your own idea** (subject to my approval).

On **Wednesday at 18:30**, a project registration form will open on Moodle.
Each project has a **limited number of slots**, and registration will be on a **first-come, first-served** basis.

Submission Requirements

Each group must submit:

- A **final report**
- A **live presentation** (attendance is mandatory) during the last 1–2 weeks of the semester
- A **GitHub repository** containing all your code
- A **README file** with installation instructions and a **demo script** for running (inference) your model on **train, validation and new test images**.

These are **real**, **practical projects** built on **real data**, giving you **hands-on experience** that **can directly strengthen your portfolio** with a meaningful additional project.

Grading Breakdown

Grading is based on a weighted combination of several criteria.
While the weighting is fixed, the score in each category includes a degree of subjective evaluation.

Criterion	Weight
Originality & creativity of your solution, integration of new ideas, and execution quality	**20%**
Effort and amount of work – 20%	**20%**
Final report quality (Abstract, Introduction, Related Work, Method, Experiments, Ablation Study, Limitations)	**20%**
Reproducibility & accessibility (Installation, GitHub code, optional webpage/app demo)	**10%**
Presentation quality on the project presentation days	**15%**
Performance on my test data, compared to other groups	**15%**

The list of the projects:

1. Project 1: Chessboard square Classification and Board-State Reconstruction (Real Data)

2. Project 2: Chess Synthetic-to-Real Generalization for Chessboard Classification.

3. Project 3: Chess Synthetic-to-Real Image Translation for Chessboard real-data generation.

4. Project 4: Campus Image-to-GPS Regression for localization and navigation tasks.

5. Project 5: CUDA based Wavelet Convolution Implementation.

6. Open Project (subject to my approval).

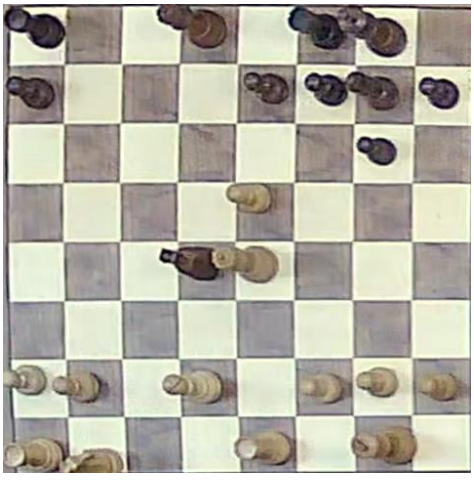
Project 1: Chessboard Square Classification and Board-State Reconstruction (Real Data)

Description:
Given real chessboard images, the goal is to classify each of the 64 board squares into one of the possible piece classes (white pawn, black knight, empty, occluded/unknown etc.). The system should then reconstruct the chess diagram (FEN notation) based on the per-square predictions.
You will receive labeled frames with the corresponding chessboard state (piece-square positions only, occlusions are not labeled), as well as PGN labeled games.
The PGN files contain the full game state information, which can be used to easily generate additional frame level labels.

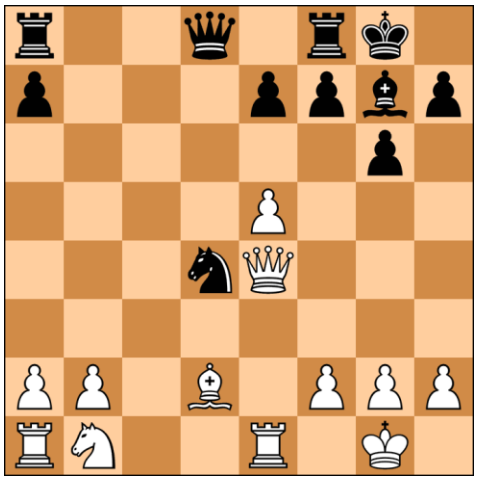
Key Components:

- Multi-class classification per cell
- Handling occlusions and uncertainty (Out of Distribution / dustbin class)
- Classifying the entire board state and producing a (FEN) and board image
- Robustness to new games.

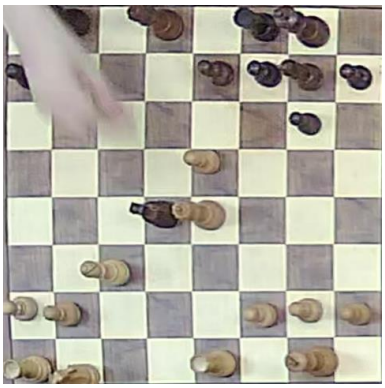
Example:



--> Classification -->



Occluded examples:



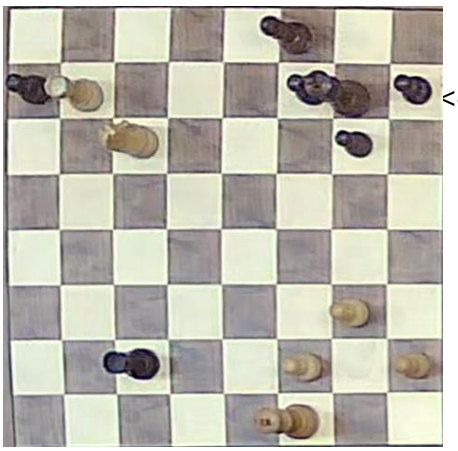
Project 2: Synthetic-to-Real Generalization for Chessboard square and board-state Classification.

Description:
In this project, you will train a chessboard square classifier (similar to Project 1) **mainly on synthetic chessboard images** generated from a simulated environment (e.g., Blender). The main objective is to evaluate how well a model trained in simulation can **generalize real chessboard images**.
You will test both **zero-shot performance** (no real training data used) and **fine-tuned performance** on small amount of data and **combined synthetic & real data** training for overall performance
This project focuses on understanding domain shift and measuring sim-to-real transfer effectiveness.

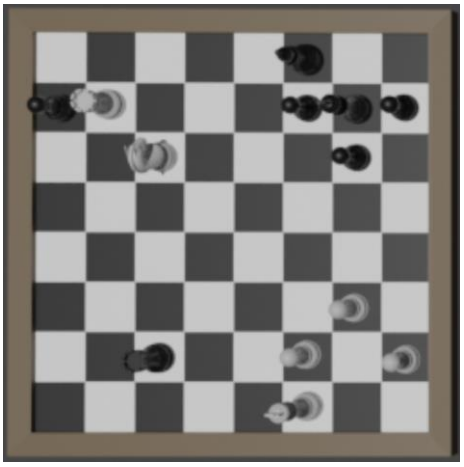
You will receive board-generation code (that I created) using Blender and PyBlender, along with a predefined set of 3D chess pieces and a chessboard. If you need additional synthetic variation (e.g., different piece designs), you may adapt the code and download more Blender chess sets (a website will be provided).

Key Components:

- Giving board generation code (that I created to some level) using Blender and PyBlender you will need to generate the data you need.
- Training the classifier using **synthetic images only**
- Evaluating **zero-shot generalization** on real images
- Performing **fine-tuning / transfer learning** with a small real dataset
- Training the classifier using **synthetic and real** data.
- Analyzing the **domain gap** between synthetic and real images



<-- Synthetic Image --> using the ground truth state of the original image.



Project 3: Synthetic-to-Real Image Translation for Chessboard Rendering

Description:

In this project, you will train a **chessboard image generation model** whose goal is to transform synthetic chessboard renders into images that visually match real-world appearance while **preserving** the geometry and board layout (piece positions, camera pose, etc.).
This is an **image-to-image translation** task focused on closing the **visual** domain gap between synthetic and real images.

Unlike **Project 2**, which trains a classifier on synthetic images and evaluates its ability to generalize to real images, **Project 3 is not about classification at all**.
Instead, your task here is to **generate** realistic images from synthetic inputs, improving the synthetic data quality itself.
The output is an *image*, not a set of labels.

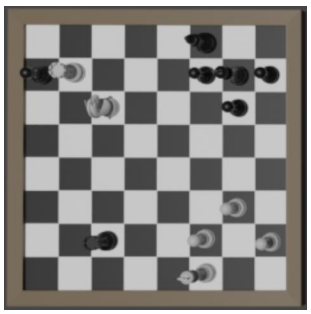
You will receive chessbpard generation code (that I created) using Blender and PyBlender, along with a predefined set of 3D chess pieces and a chessboard. If you need additional synthetic variation (e.g., different piece designs), you may adapt the code and download more Blender chess sets (a website will be provided).

Key Components:

- Using the provided PyBlender generation script to create your synthetic dataset
- Converting synthetic chessboard renders into **realistic-looking** images.
- Preserving geometric structure: piece identity, board layout, etc.
- Exploring image-to-image translation methods (GANs, CycleGAN, diffusion-based approaches, etc.)
- Evaluating how well the translated images match the real domain visually (qualitatively and/or quantitatively)

Input **Image-to-Image generation** **Output**

---------->



Project 4: Campus Image-to-GPS Regression for Localization and Navigation.

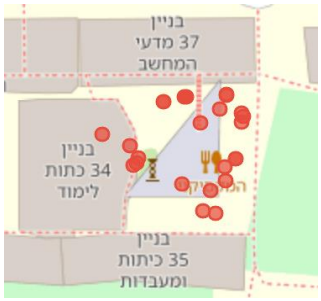
Description:
In this project, you will build a model that predicts the **GPS coordinates** of the camera given a single image on the campus.
You will collect images across a predefined area (that I will choose) using your smartphones. Modern phones automatically store GPS coordinates in the image metadata (EXIF), allowing you to use these coordinates as ground-truth labels.
The goal is to train a regression model that maps the visual appearance of the scene to its geographical location, enabling coarse localization and navigation on campus.

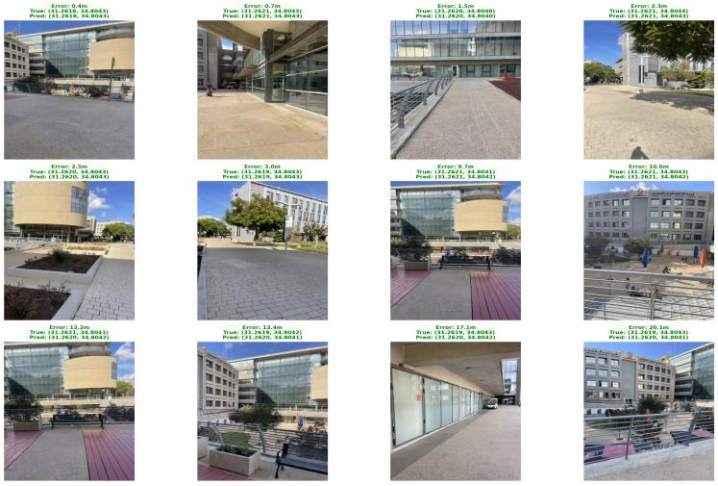
You can see a example here from an earlier project I conducted using **classical computer vision methods only**, where the task was to **classify** the building in the image and then provide a navigation map to the user (note: that older project performed classification, not regression).

Key Components:

- Collect your own dataset of campus images with **embedded GPS metadata** in the predefined area.
- Extracting latitude and longitude labels from EXIF data
- Training a neural network (you need to explore the models, loss functions, data augmentations etc.) to **regress GPS coordinates from images.**
- Evaluating the model by computing GPS prediction error in meters.
- Visualizing predicted vs. real locations on a map.

Example of a very small, predefined area. The real predefined area for the project will be much larger and will be provided later.





Project 5: CUDA-Based Wavelet Convolution Implementation

Description:
Wavelet Convolution (WTConv) for Large Receptive Fields recently published in ECCV 2024, by Shahaf Finder, Roy Amoyal, Eran Treister, Oren Freifeld.

WTConv provides several practical advantages, including consistent performance improvements over standard baselines. As a result, it is already being used in a variety of practical applications. **I will present the work in the lecture on 16.12.25.**

In this project, you will implement the **forward and backward passes** of the Wavelet Convolution operator in **CUDA (C++)**, and expose the implementation to Python using **pybind11**. You will then integrate your custom CUDA kernel into modern deep-learning architectures such as **ResNet, ConvNeXt, or MobileNet**.

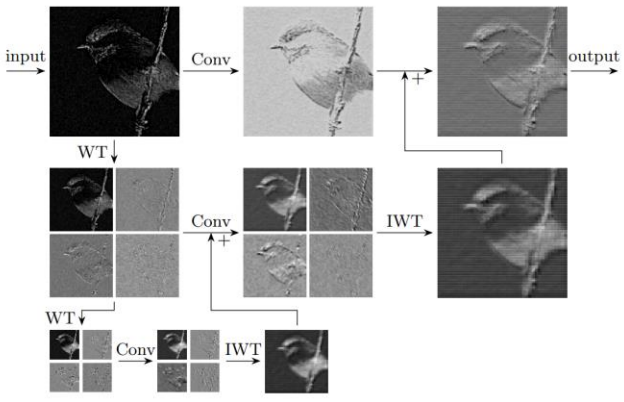
The main objective is to speed up Wavelet Convolution, which is currently implemented in an inefficient manner.

Key Components:

- Implementing the **forward Wavelet Convolution** in CUDA
- Implementing the **backpropagation (gradient)** pass in CUDA
- Binding the CUDA/C++ implementation to Python using **pybind11**
- Integrating the operator into PyTorch models (ResNet / ConvNeXt / MobileNet)
- Benchmarking runtime improvements vs. the existing inefficient implementation
- Validating correctness using unit tests and numerical gradient checks

In any case, I can help you understand the paper and possibly guide you to some level (depending on how many people choose the project).

If you significantly speed up the operation, we will allow you to submit a pull request to our official implementation, contributing to a real research project.



Project 6: Optional Open Project

Description:
This project allows you to propose your own idea, as long as it clearly fits within the domain of **deep learning for computer vision** and involves working with **images or video**.

Your proposed project should define a meaningful vision task, the dataset you plan to use or collect, the deep learning approach, and the expected results.

The project must also match the overall complexity and workload of the other predefined projects.

To proceed, you must present your idea and obtain approval in advance.
You can do this by coming to my office hours or scheduling a Zoom meeting with me **no later than the end of this weekend (13.12.25).**

While your proposal is under review, you should temporarily register for one of the existing predefined projects to ensure you have a reserved slot.

Key Components:

- Proposing a project that must involve deep learning and computer vision
- Using or collecting images or video as the core data modality
- Ensuring the project matches the complexity and scope of the predefined projects
- Meeting with me (in person or via Zoom) to receive approval
- Registering for an existing project until approval is granted
- Providing a clear project scope, dataset plan, method, and deliverables

Good luck!