Faculty of Mechanical Engineering and Robotics

Department of Robotics and Mechatronics

Basics of AI and Deep Learning

Course for Mechatronic Engineering with English as instruction language

Instruction 12:

Advanced AI models and concepts

You will learn a selected topics and problems related to deep learning architecture of your choice – gaining some hands-on experience and, possibly, in-depth understanding of methods' details

Additional materials:

Learning outcomes supported by this instruction:
[Here a list of learning outcomes' codes]

Course supervisor:
Ziemowit Dworakowski, zdw@agh.edu.pl

Instruction authors:
Ziemowit Dworakowski, zdw@agh.edu.pl
Adam Machynia, machynia@agh.edu.pl

Introduction

This laboratory works in a different way than all the others on this course.

First of all, it is designed for two meetings, so you start working on it during one (12th) laboratory, and finish it during the next (13th) laboratory. For this reason it contains parts to do during meeting and parts at home. The idea is that first, you learn basic concepts and have the chance to discuss with the teacher initial approach to the problem, then, you do your own experiments at home, and, finally, discuss and present the outcomes to the teacher and to your colleagues during the final meeting.

Secondly, this instruction contains 8 separate topics to choose from. You are supposed to do tasks ONLY from one of these topics. Get familiar with all of them but this time don't invest any work into them beforehand – as they will be chosen during the 12th meeting and assigned to you by the teacher (probably on the basis of your performance so far – higher grades from tests and laboratories will mean you get to choose your topic first). Note that some topics have two versions – for paid and free generative AI solutions. Picking any of them is based on your personal preference, it will not change your grade in any way. The idea is that the models tend to differ significantly, some tasks are not doable in free software. If you have subscription either way, you might as well test the better model and not force yourself to downgrade.

Thirdly, as the laboratory sometimes requires a lot of experimentation – for some topics it is acceptable to do it in pairs as well – in such a case you will be expected to do a little bit more (e.g. more tests and evaluations). Look for detailed instructions in each topic and look at the topic itself, as some of them do not allow such interaction. Since some of the topics work with many possible datasets, they can be taken multiple times (number of slots is shown next to the topics).

The basic idea of getting grades based on the proportion of tasks done still stands – the more you do, the higher your grade will be. This time, however, you won't be expected to finish the remainder at home after the last laboratory – so the grade you get during 13th laboratory will be considered final (provided it is a pass).

List of topics to choose from

1. **General interaction with ChatGPT (1 student, recommended subscription)**
   You will learn how to build good prompts, test different functionalities of ChatGPT, test how it changes depending on the language and will learn the current limits of its competence. The topic will require studying additional information related to large language models structure and way of operation – and higher grades will require you to acquire and understand a lot of general knowledge.

2. **Interacting with image generation models (1 student, recommended subscription)**
   You will pick and test a selected genAI models in generation of different tasks trying to maintain consistency of outcome and getting as close as possible to the desired results. Note that this topic shines especially if you have paid subscription to at least one model (e.g. Midjourney or Dalle3) – as it usually allows for much less constrained tries.

3. **Data classification using LLMs only (1-2 students)**
   You will again build shallow and deep data classifiers based on datasets used during the course. This time, however, you will have your "LLM minion" that will do all the tasks for you. Your goal will be to define the tasks and evaluate whether the output is correct.

4. **Scientific information acquisition using dedicated LLMs (2 x 1 student)**
   This is a specialized topic in which you will learn how to interact with large language models in search of scientific information, how to research state-of-the-art and how to verify if scientific data you acquire through GenAI tool is reliable and safe to use. This topic will require

5. **Vanilla GAN demonstrator (1-2 students)**
   It all started with basic Generative-Adversarial Networks (GAN) – or at least all in terms of development of modern Generative AI models. In this topic you will try to build your own basic GAN model for generation of images of your choice. This topic is a good starting point to learning about generative AI from a practical perspective, but you should not hope to get as eye-catching results as you are used to see over the internet – it is after all a rather old and simple model, and will probably take a few days to actually train somewhat properly.

6. **Game design using LLMs (2x 1-2 students)**
   You will craft your own game with ascii graphics using LLM and selected programming language (matlab, python, C++, etc.). You will test LLM's capability of maintaining context and building functions compatible with predefined code structure.

7. **Deep engineering time-domain signal interpretation (1-2 students)**
   Do you remember the simulated database we were working on during the 5th instruction? You will now do a similar task, but based on real-life signals and using a deep learning approach. You will design a deep convolution neural network for the task – and compare it with a shallow feature-based classifier. This task will require you to study additional materials related to the context of your work – including most notably condition monitoring in which you will be applying your solution.

8. **Practical image classifier using self-acquired data (2x1-2 students)**
   This one requires almost the same setup as one that we had in laboratory 11 – but you will need to gather your own dataset for the task of your choice. While this may seem trivial at first, practical data acquisition organization is not that obvious, you will learn about practical data leakage sources and ways of providing general and representative data. Note that this topic will require a lot of "field work" and is not a good choice for procrastinators, you won't be able to finish it in just a day before the final laboratory.

DETAILED TOPIC INSTRUCTIONS

Topic 1: General interaction with ChatGPT

While the LLM model is defined in the topic feel free to negotiate with the teacher if you really want to use a different model. Note that this model is required to have memory of your actions – to build a long term working setup.

1. Pick one of the laboratories from our course. Then formulate topics for your ChatGPT evaluation. You will want: - A "documentation" task (e.g. laboratory report preparation, given the data, results and codes that you prepared when you did this instruction "manually") - A "coding task" (e.g. building a function for a selected additional task of the instruction that you did not do in scope of the normal laboratory instruction) - A "learning task" – learning about context of selected methods and topics – e.g. constraints for methods application, risks or potential areas of application. This includes using LLM to gain new knowledge in the area related to the laboratory scope. - A "creative" task – extending the instruction towards new scope, e.g. building your own instruction part or redesigning parts of the instruction. Document these tasks and get teacher's acceptance for them
2. Run the topics using a "fresh" model (in ChatGPT it can be a "temporary" one). Try to construct prompts as well as you can but refrain from multi-prompt approach (let all the necessary information be included in the initial prompt). Store your prompts and the results obtained in the form of a report. Discuss critically the results – and store this discussion as well. Do NOT use a meta-approach where you are testing the LLM and use it to provide report on this very action.
3. Now use model's capability to store information. Describe to the model what are your goals and priorities, what language it should use in further interactions with you, what is its role in these interactions. Solve the topics again, this time using a multi-prompt approach, where you first start by providing context, checking how well model understands the context and then gradually crafting the response (from highlights, through "list of contents" up to full text or task solution). Store the results and your prompts – and compare these results critically with ones obtained in the 2nd task.
4. Evaluate critically the learning task. Find out the answers to your questions the "classical way" and then check (a) how accurate and correct the answers of the model were and (b) how complete they were given the context you provided. Would you explain the material differently?
5. Go back to the learning task with the LLM and start asking for details and clarifications. Evaluate model usefulness in (a) showing you detailed sources for information its providing and (b) providing useful explanations (clarifications of issues raised by you)
6. Go back to the "creative" task – and use the model as a mirror to help you formulate better your own ideas. Try to propose ideas which you know are weird and not really reasonable (like: mixing the topic of the instruction with unrelated themes and convoluted examples). Check how well the model can point out your "mistakes" if it is not prompted to look for them first.

Topic 2: Interacting with image generation models

This task gets conceptually easier with subscription models (just prompt to get image), but can be more rewarding with free ones (require a bit of installation – but then you have your own model to work with). Not all the tasks are doable with every model. For this reason, after the "3.0" part all of the other tasks are marked in green. In order to get 4.0 you will need to solve at least one task more. In order to get 5.0, two more tasks should be sufficient (if done correctly).

1. Pick image generation model of your choice. Easy but paid options include Midjourney or Dalle2 (the latter available through the ChatGPT menu). Free options include https://www.mage.space/, or models found e.g. on https://huggingface.co/ - but these will require a bit of setup. You might want to run ChatGPT for a quick review of the options available within your constraints. Before committing to the model check its availability, you will probably need to generate at least 40 – 50 images to complete this topic
2. Pick themes for the following categories: a logo (e.g. for your future startup), a set of style-consistent pictures illustrating different concepts (e.g. to illustrate a book on particular hobby or provide graphics for your webpage and, finally, a large poster on the particular engineering theme that can fit within the area of mechatronics (e.g. wind turbines, or condition monitoring, or particular type of robotics). Describe your themes and prepare for yourself a set of guidelines and requirements to meet (pretend that you are a person that actually delegates the job for you). Document these and get teacher's acceptance for the documentation.
3. Solve each of the tasks in the image generation engine of your choice. Feel free to change engines if you feel one is better suited for one task and other will be better at another. Store your prompts and prepare a report showing your progress (with partial images, mistakes and problems that you encountered).
4. Take the poster generated in task 2 and try to get rid of particular elements of the picture – or replace them with others. Check how well the model understands what you want it to do – and how well it does the job without affecting other parts of the image. Test the available strategies to solve it – e.g. starting from the same seed but generating image using slightly different prompt, inpainting on the generated image, inpainting on the manually modified image. Note that not for all the models different strategies will be available. Describe your results and findings in a report.
5. Solve again a "poster" task – this time using different models or different configurations available in your setup. Provide a report on your findings and estimate which of the models or setups you find best in completing the task. Note that you should take into consideration the "image quality factor", but also the "merit factor" – does the model grasped the actual look of the things you want to show, or is it rather a "fanfiction" regarding mechatronics area?
6. Pick a LoRA option that is suitable for your choice of the "set of pictures" task definition (note: LoRA are not available for all the setups). Check if this allows to maintain higher consistency in the set of generated images. If you can't use LoRA that fits your initial description, find one that is the closest and check the result. Describe your results and findings in a report.
7. Use your own picture as a basis to generate a robotics-related image (it can be a robot with your face, you assembling a robot, you being chased by the robot, etc.). Try different strategies available in the model (if it allows for that): uploading your photograph as an inspiration or inpainting in your photograph). Check how close the model gets in terms of similarity. Describe your results in a report.

Topic 3: Data classification using LLMs

Take data used in previous classification problems in laboratory. You have: "Artificial clusters" (A) "Engineering signals" (B) and "Image features" (C) sets. In this instruction you will NOT use our instructions and your own matlab codes to solve these classification problems. Instead – you will be using purely LLM of your choice (Recommended: ChatGPT 4.0 or higher). The goal is to upload the data to the LLM and then guide data processing to finally get the classifier and labels for the validation and training dataset.

Save your interactions with the LLM for discussion with the teacher during instruction 13!

1. Take dataset A divided into training, validation and testing subsets. Store safely a testing subset and don't show it to LLM. Pass training and validation dataset to the model, ask it to evaluate data and plot it. Store the generated figures
2. Ask the model to build the classifier based on training dataset. Ask it to evaluate its performance and optimize its structure. When you are satisfied with LLMs actions, feed it the testing dataset WITHOUT LABELS – and ask to do classification and download the result. Then finally manually compare the labels provided by the LLM with your ground-truth labels which you kept without its reach. Check manually accuracy of classification and compare with your own results from laboratories. Prepare a short report showing the results and conclusions.
3. If you aim for higher grade: for datasets B and C start with raw data from training datasets and ask the model to prepare feature extraction algorithms first (you might need to prepare manually codes for data split, so the LLM won't see testing part of your data yet). Try to define task in such a way that the model will calculate the same features we did during laboratories. Don't pass to the model any codes and instruction fragments, only communicate your requests using natural language.
4. Test the possibility of fully-online feature extraction. Is it possible to guide the model to process your raw data, calculate feature matrices and store them further use? If your model allows that – do it. If it does not, test at which point it breaks and explain why.
5. Run the classification routine similarly as in tasks 1 and 2. If you did tasks 3 or 4 – use these features. If not – use features pre-extracted in scope of the laboratory. Then compare the results of running the classifiers on testing dataset with your ground truth (true labels from the original dataset)
6. Ask the model to test different classifiers (SVM, kNN, ensemble models) and find one that fits best to the given task.
7. Test the possibility of using deep learning based on raw data for datasets B and C without label calculation
8. Compare the results obtained through three approaches: manual (results of our laboratories), semi-automatic (LLM produces codes, we run them and store results) and fully-automatic (LLM produces and runs codes, returns vectors of labels for each dataset). Check the accuracy but also your perceived time investment. Try to measure how much time is needed for each method.
9. Prepare a laboratory report comparing accuracy obtained during classes and accuracy obtained with LLM model, discussing findings and conclusions.

Topic 4: Scientific information acquisition using dedicated LLMs

Your goal is to prepare a literature review based on scientific literature. In all cases, save your conversations with LLMs.

1. Pick a topic for your literature research. **Consult your choice with the teacher and get his/her approval.**
2. Refine your search query and prepare initial prompts.
3. Conduct a search using the default ChatGPT model. Remember that you need references to scientific literature. After obtaining the initial review, elaborate on the most important aspects of the topic. Repeat task 3 using Scholar GPT, a GPT model tailored for research.
4. Repeat task 3 using Perplexity.ai.
5. Repeat task 3 using a selected tool from the following: https://scite.ai, https://storm.genie.stanford.edu, or https://www.undermind.ai,
6. Compare the results from tasks 3–6. Identify the main differences and evaluate the quality of the results. Determine which model is most suitable for your needs and explain why.
7. Based on the previously conducted steps, compile a list of keywords related to your query.
8. Conduct a rough manual search using the prepared keywords in available open-access repositories. Keep in mind that you are generally not allowed to upload paywalled articles to AI models. Select a few articles (3–7) that appear to be the most relevant.
9. Using a selected tool (ChatGPT, Scholar GPT, Perplexity, or another approved by the teacher), prepare a brief summary of the articles selected in task 9.
10. Summarize a selected article using https://notebooklm.google. You may also present it as a podcast.
11. Based on the previous results, use a selected tool (ChatGPT, Scholar GPT, Perplexity, or another approved by the teacher) to prepare a literature review chapter for your topic. It should be a brief yet comprehensive introduction, spanning approximately 1 to 4 pages.
12. Create graphics illustrating the main conclusions or the most relevant/interesting parts to include in the report from task 12. Use https://napkin.ai.
13. Prepare a presentation based on your findings, especially focusing on the topics covered in the report generated in task 12. Use https://gamma.app.

Topic 5: Vanilla GAN demonstrator

1. Choose the topic for image data generation. Preferably – these should be your own images (images of your face, images of your pet, images of your loved one, images of your favorite bike, etc.). You will need to provide at least a hundred of such images.
2. Implement vanilla GAN network using matlab tutorial: https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html
3. Train your model to generate data fitting your topic. Note that you should probably start small (with a very low image resolution). Check how long the training lasts and what are the images obtained at different training stages. Note that you should probably expect rather long training times. Consider running training with default parameters for at least entire night.
4. Evaluate influence of several training metaparameters changes. Notice if any change causes significant difference in training
5. Prepare a report showing your findings (graphical form + explanations)
6. Evaluate influence of (a) network size (is depth of generator or discriminator more important?) and (b) weight of generator and discriminator training (which of them should we focus on more?)
7. Look to scientific literature and then pick one change that you think could influence the training in a positive way (either towards higher resolutions of generated data or towards closer resemblance of training examples) – and then implement this change in your model (you can use LLM in this task)
8. If you decided to do task 6 or 7 – include their description and discussion of the obtained results in your report.

Topic 6: ASCII game coded with LLM

This task is devoted to coding and advanced (structurally) program using purely LLMs. For the entirety of this task you are not supposed to write a single line of code yourself – apart from file and function naming. Store your interactions with LLM in a way that allows you to show them to teacher when asked. Note that not all of the "green" tasks are necessary for a 5.0 grade here. Just one "green" task (but well executed) should allow for 5.0

9. Choose the topic for your game. It needs to allow for deep complexity of mechanics while not relying on strong asset usage. Good candidates for your choice are roguelike games, text-based rpg games or ascii tower defence games.
10. Pick the language you would like to use. Recommended option is matlab – just because you are not really "supposed" to write games in it and it is interesting to see how well the LLM will manage this weird task. If you want to go "standard", feel free to choose python, java or C++ as well, but note: you should either be already familiar with the language and the environment, or be prepared to quickly learn how to define functions and how to run and debug codes in the selected language.
11. Choose your "coding companion" – an LLM which is known for a good code-writing skills.
12. Think about a rough structure of the game (is it quest-based? Is it map-based? What will be the major organizational elements of the code (any classes or major functions) – feel free to consult this task with an LLM of your choice (the creative process can also be partially delegated to the model). The structure needs to be

modular and scalable – in the course of this task you will test the limits of the "model coding capabilities" by extending and complicating the initial design. Think about the ways you plan to test these limits.

13. Consult your decisions (Game topic, language, LLM used, basic structure, plans for scalability test) with the teacher during the 11th laboratories. Formulate the "workplan" document storing your goals and constraints.
14. Prepare the "early-alpha-version" of the game – to test the workflow and communication with the model. Refine your structural choices regarding class organization and functions' communication within the model. You can do that also in consultation with the model. Store your findings and conclusions in the form of a short report (what went well? What went wrong?)
15. Build the refined code structure one-function-at-a-time, ensuring that the model remains focused on the task at hand while maintaining compatibility with the planned code structure. Complete the task, run the beta-tests and correct any bugs and mistakes (again, using only your interactions with LLM).
16. Test the limit of structural complexity the model is able to maintain. Add maps, characters, enemies, functions, currencies, functionalities etc. (depending on the topic of your game) until maintenance of the software purely via LLM will no longer be possible. This task is necessary for getting 3.0, but getting a higher grade will require strong commitment here.
17. Describe final state of the game (the working version), its functionalities and your findings and conclusions regarding the task in the form of the report.
18. Think of a new functionality that was not intended during in the 4th task and was not implemented in the 7th task. Let this functionality require changes to the other code parts (e.g. different type of encounter, completely new feature of the characters in game). Implement this change first by allowing the model to "figure it out" based on its already established knowledge of the code and then, if it won't work – by manually tracking parts of code that require changes – and requesting them from LLM
19. Use the model for the balance tests of the game. Your goal is to provide a good user experience – by providing a challenging and rewarding play. Use the model to discuss which game features should be balanced to this end and to design a mathematical model helping in this balance. Then, approve a list of changes and ask the model to include the changes in all the necessary places of your game.
20. Add assets to your game – let the code use a library of graphical, sound or animation assets. Note that these will probably require changes to the "game engine".
21. Transfer final version of your game to a different programming language using LLMs. Check how far you need to influence this process – and what will happen if you just pass files one-by-one. Is it possible to build game for different languages easily?
22. Use the model, to craft game instruction – telling the user how to run the game and what is the strategy that should be incorporated to achieve a good result.
23. Provide second part to your report prepared in step 15 – to incorporate conclusions and findings of points 16 – 20.

Topic 7: Real-life time-domain signal classification

This task requires a bit of coding which we did not cover during classes – but you can find tutorials on similar tasks or use help of chatgpt to code the deep learning part of the task for you if you need.

1. Download Bearing dataset available here: https://engineering.case.edu/bearingdatacenter/download-data-file Look at the data. Since the signals are long, you will need to cut them in parts, so each part would form a "sample" for classification. Decide how long a "part" should be. The longer it is, the less samples you will have at your disposal, but the more consistent features calculated from them will be. Prepare such a dataset and then divide signal parts accordingly into training, validation and testing datasets.
2. Do a shallow-learning-based approach using simple features we've calculated during classes. Check what is the accuracy obtained by your model of choice on the testing dataset.
3. Look into the literature for more advanced signal features or run a more detailed signal processing (e.g. to extract frequency spectra) until you find features that should work good in your setup. Code at least two better features and check if they improve the classification accuracy.
4. Compare both approaches in the report form.
5. Configure a convolutional neural network for signal classification. You will need to modify at least: (a) raw data input. Instead of image datastore you can use signalDatastore or simply construct datasets from cell arrays containing particular signals, (b) replace square convolutional kernels with 1-dimensional ones and (c) change input layer from image to sequence. The rest of the code should work similarly. Use the structure configuration that you feel would be sufficient to solve this task
6. Train and evaluate your deep solution – in classification of different signal faults. Store the results and be able to show them in a graphical form
7. Optimize network structure (including number of layers and filter sizes) and evaluate the network statistically.
8. Look at the data again. Think about what are the possible explanations for the outcome you obtained. Test if different data splits provide a more or less reliable solution. Find information about this dataset in the literature and try to compare your solution to ones reported in the literature. Provide a report explaining steps taken (including the approach to code the solution) and discussing the obtained results.

Topic 8: Practical image classifier using self-acquired data

1. Pick a practical classification problem to solve using your own acquired dataset. During the course of this mini-project you will need to prepare a large database of images, based on which your classifiers will be able to work. Possible examples may include *classification of trees by their leaves*; *classification of dishes by their top-view photograph*; *classification of road signs into categories*. Consider ethical constraints, pick a topic for which you will be allowed to take hundreds of photographs**. Consult your choice with the teacher and get his/her approval**
2. **Plan the experimental part.** Consider how many images are to be acquired, under which conditions, who will acquire them and how will you make sure that the dataset you are preparing is representative enough to train reliable classifiers on. **Consult your choice with the teacher and get his/her approval.** Prepare documentation with plan of the experimental part.
3. Ask the teacher to provide input regarding the "*additional testing dataset*" (teacher will modify your experimental plan by adding a design of a different experiment based on yours to provide testing data). Include that in your plan
4. Acquire data **strictly according to your plan**
5. Acquire data according to teacher's plan
6. If you feel like you missed something important designing experiment or you noticed a risk factor and would like to have more data in the end, feel free to acquire an additional data for further testing and enhancement of your dataset, but separate them clearly from the initially planned experiment.
7. Label your dataset and prepare a document with experimental description
8. Divide data into training/validation/testing parts randomly, exclude teacher's dataset or any other datasets you may have acquired – Lets name it Split 1
9. Divide data into training/validation/testing parts in an informed manner – based on e.g. time of acquisition, experimentator, equipment, place of acquisition, etc. – ensuring independence between datasets – Lets name it Split 2
10. Do a standard optimization of metaparameters and training of your classifier (from instruction 11) using training/validation subsets and test it on testing part – for Split 1
11. Repeat point 10 for Split 2
12. Compare statistically (based on multiple trainings) performance of the classifier on your's testing set and the teacher's testing set. If you have Split 2, test it as well.
13. Compare and discuss different approaches to classifier training and configuration (be prepared to present your findings to your colleagues and the teacher using proper results' visualization).
14. Submit the report with your data, your experimental results and experimental description