Agentic AI with MCP – from Chatbot to Co-Scientist

(taught by Dr. Christian Kauth)

Large language models (LLMs) like ChatGPT, Copilot or Claude are remarkable conversationalists – but out of the box they cannot train a model on your data, run inference on a live webcam feed, or touch a file on your laptop. They can only talk about it. This course closes that gap by teaching you how to turn a general-purpose LLM into a specialized agent: one that can call real code to train and use machine learning models, while staying firmly under your control.

We do this with the Model Context Protocol (MCP), the emerging / established open standard for connecting LLMs to external tools, data and applications. You will build your own MCP server – mostly in Python – that exposes a small set of well-defined tools such as "upload a training set", "train a model" and "run inference". Any MCP-compatible client, e.g. GitHub Copilot in VS Code or Claude Desktop, can then discover these tools and call them on its own initiative, in response to a plain-language request from the user.

Working step by step, you will wire up a tool that trains a model on data supplied by the user (implemented in Python, with a look at how the same tool could equally call R via Rscript or rpy2), and a second tool that runs inference on demand – collecting input through a simple form or, for image data, straight from the webcam. The trick: because your MCP server runs locally, the LLM's tool calls only ever need to carry lightweight references (a file path, a few numbers) rather than the data itself. That means your training set, your trained model and your predictions can all stay on your machine; only that short back-and-forth, plus anything you deliberately choose to show the model, such as a loss-per-epoch plot, travels through the cloud.

This course is designed to bridge the more traditional ML/DL courses of this Winterschool (KNIME, R, Python) with the modern generative-AI era: by the end of the day you will have turned a general-purpose chatbot into a specialized co-pilot for your own machine learning workflows.

Fun fact: Anthropic, who introduced MCP in November 2024, likes to describe it as "a USB-C port for AI applications" – one standard connector that lets any model plug into any tool, instead of everyone building bespoke one-off integrations.

Objectives

- To understand what MCP is, and why it matters for the agentic AI era
- To build and run your own MCP server in Python
- To connect that server to real MCP clients (GitHub Copilot, Claude Desktop)
- To equip an LLM agent with a tool that trains a machine learning model on data you provide
- To equip an LLM agent with a tool that runs inference on demand, including image data from a webcam
- To keep your data and models local while only the LLM itself runs in the cloud

Content

- What MCP is (protocol basics: tools, resources, prompts; servers and clients)
- How to build a Python MCP server and expose custom tools to an agent

- How to add a data-upload tool so participants can hand training data to the agent
- How to add a training tool (Python, with a glance at R via Rscript/rpy2) that returns results such as a loss curve
- How to add an inference tool, using text, forms or webcam input to collect new data
- How to design tool calls around lightweight references (file paths, summary numbers) so data, model and computation stay local while the LLM stays online
- How to combine classical ML/DL know-how with modern LLM agents

Preconditions

- Basic familiarity with Python is helpful, but not required: if you don't have it yet, feel free to enjoy the day as a live demo, and pick up the fundamentals in the Python & SciPy crash-course included in the course "Machine Learning with Python – from Zero to Hero".
- Unlike the other Python courses of this Winterschool, this course does not run on Google Colab: your MCP server, its training and inference tools, your data and your model all run locally on your own machine, with only the LLM itself living in the cloud.
- Please come with Python (3.12+), Visual Studio Code and Claude Desktop already installed on your laptop (Windows, Mac or Linux all work).
- Comfort executing basic commands in a shell/terminal (e.g. running a script, installing a package with pip) on your operating system.

Duration

1 day (roughly 7*45 minutes)

Evaluation

take home exam: project work – build and demonstrate an MCP tool that trains a model on your own data and runs inference with it.

ECTS

0.5