UNIVERSITÄT BAYREUTH

Faculty of Mathematics, Physics & Computer Science

Nikita Agrawal
Research Assistant, Chair of Data Systems

Prof. Dr. Ruben Mayer
Head of the Chair of Data Systems

Concept and Sketch for the Bachelor Internship: Federated Learning

Table of Contents

1 Project Phase 1: Setup and Concept

Task 1: 27 October 2025 – 03 November 2025

1.1 Goal

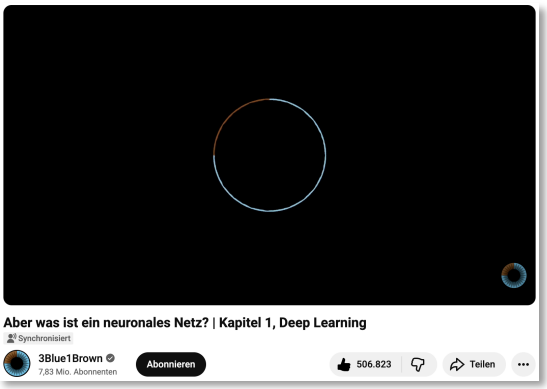
The objective of this initial phase is to design and prepare the theoretical and technical concept for the entire internship project.

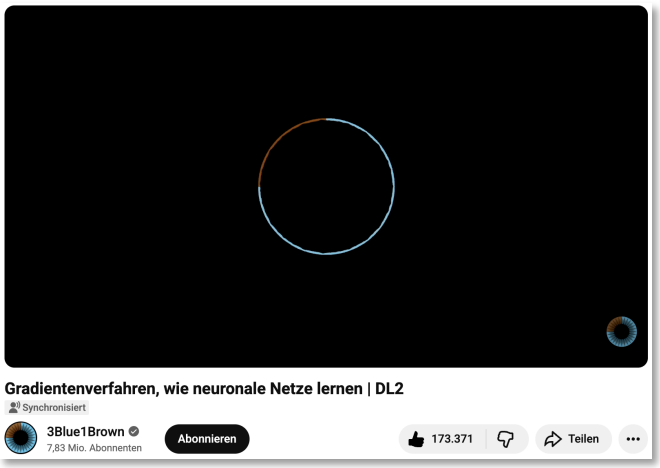
The aim of this phase is to lay the groundwork for all subsequent development steps by defining the project structure, outlining the methodological approach, and preparing all technical components that will be required later.

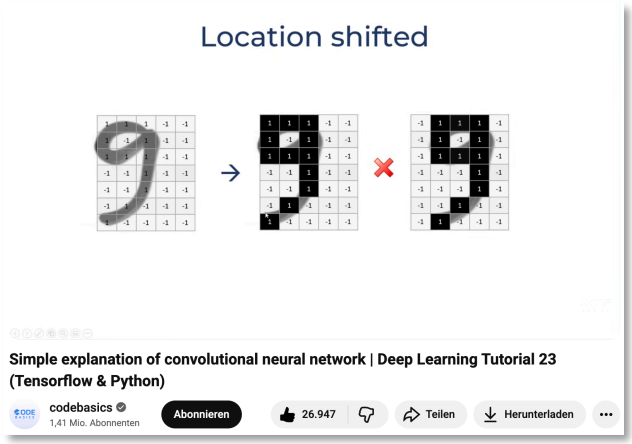
This includes creating a detailed conceptual sketch that defines how each main aspect (*Research, Coding, Visualization, Documentation* and *Experimentation*) will be approached in the following phases. The project will progress from a centralized training baseline (Task 2) towards a federated setup (Task 3) and finally to a non-IID simulation (Task 4), all within one consistent Python codebase.

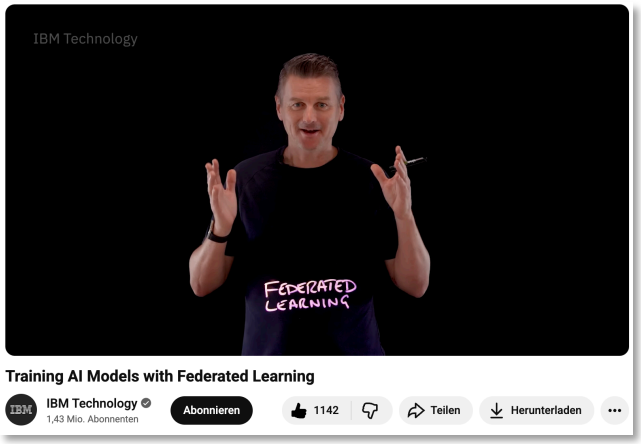
1.2 Research

In this first research phase, I focused on understanding the core technologies and theoretical foundations of the project. I reviewed the *Module Slides* and the *Extra Reference Material* on federated learning, machine learning, neural networks and optimization that were provided in the e-Learning course.

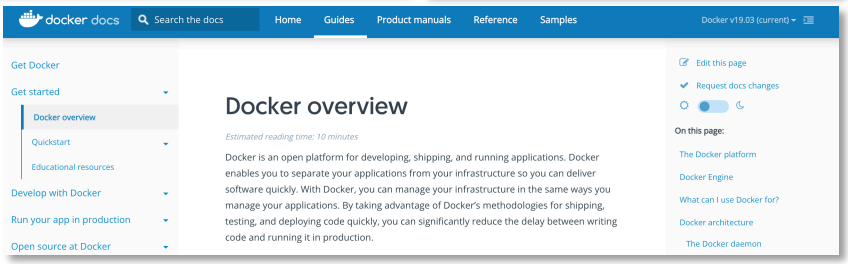




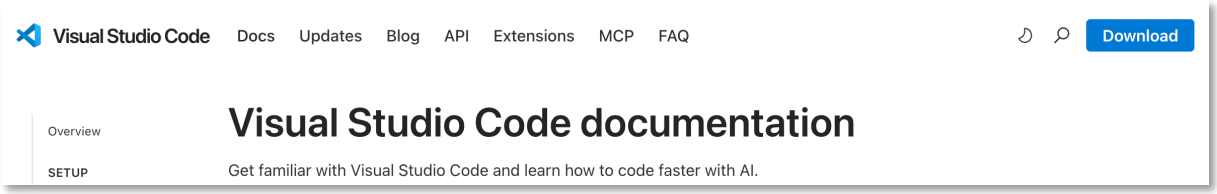


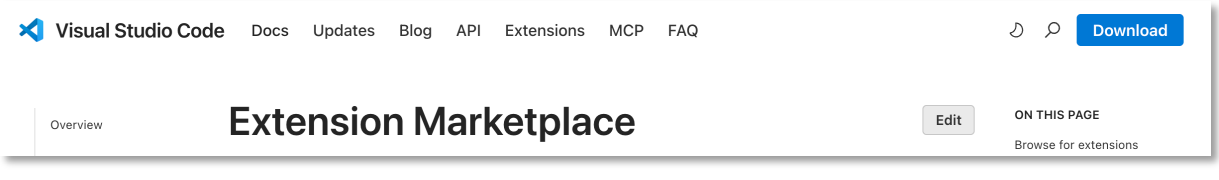


Additionally, I studied several video tutorials and official documentation on Docker and Docker Compose to understand containerization principles and best practices *(see images below)*.



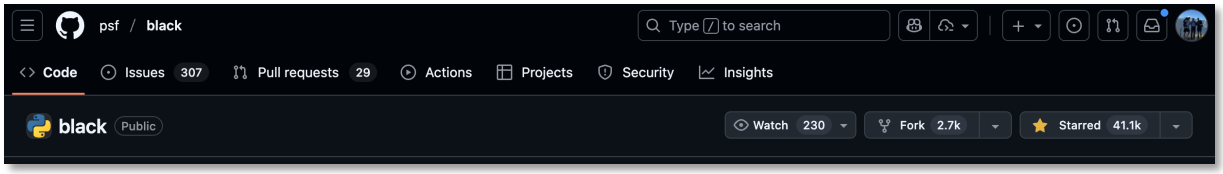
Although several code editors, such as PyCharm (JetBrains) and Eclipse, were suggested, I decided to use Visual Studio Code, as I am already familiar with it. I also reviewed the official documentation and explored useful extensions.



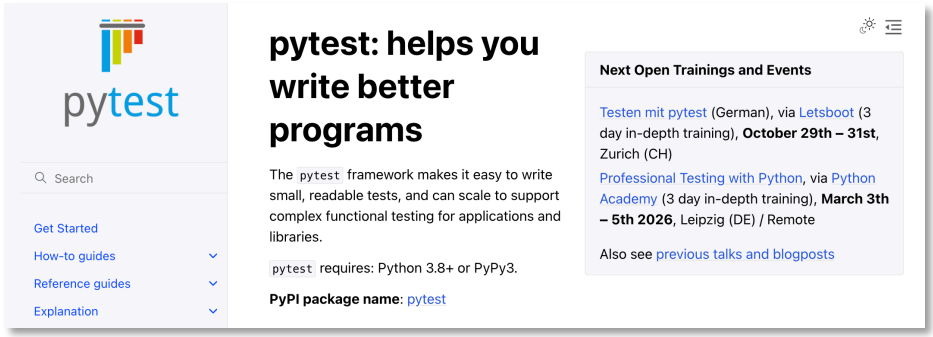


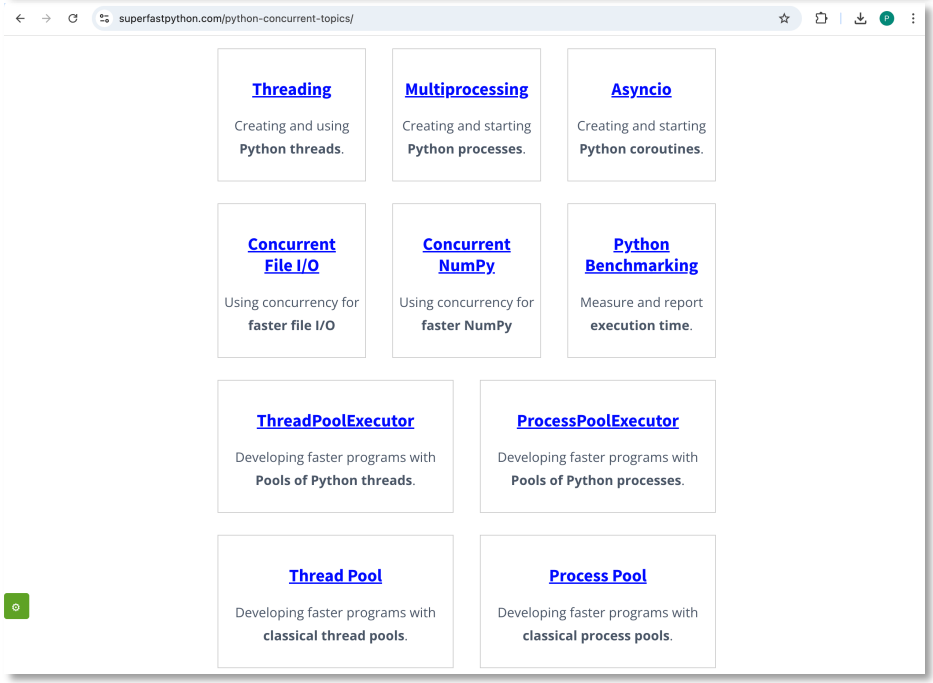
Having gained prior experience with Python during my bachelor's thesis, I was already familiar with its syntax and thus did not need to learn it again.

For this project, I will enforce consistent code formatting with Black and perform linting with Pylint to maintain a clean, standards-compliant codebase.



During this initial phase, I further deepened my understanding by studying testing frameworks such as *pytest* and exploring more advanced Python concurrency concepts like *asyncio*, *coroutines*, *multiprocessing* and *threading*. These concepts may be useful for setting up parallel clients in later tasks *(see images below)*.





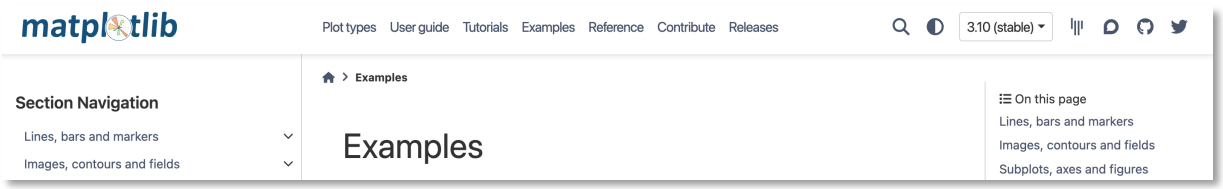
1.3 Documentation

I use Git for version control in this bachelor project, and the repository is hosted on GitHub. The repository contains all key theoretical components. All materials are structured in Markdown format, and the following tree provides an overview of the summarized files:

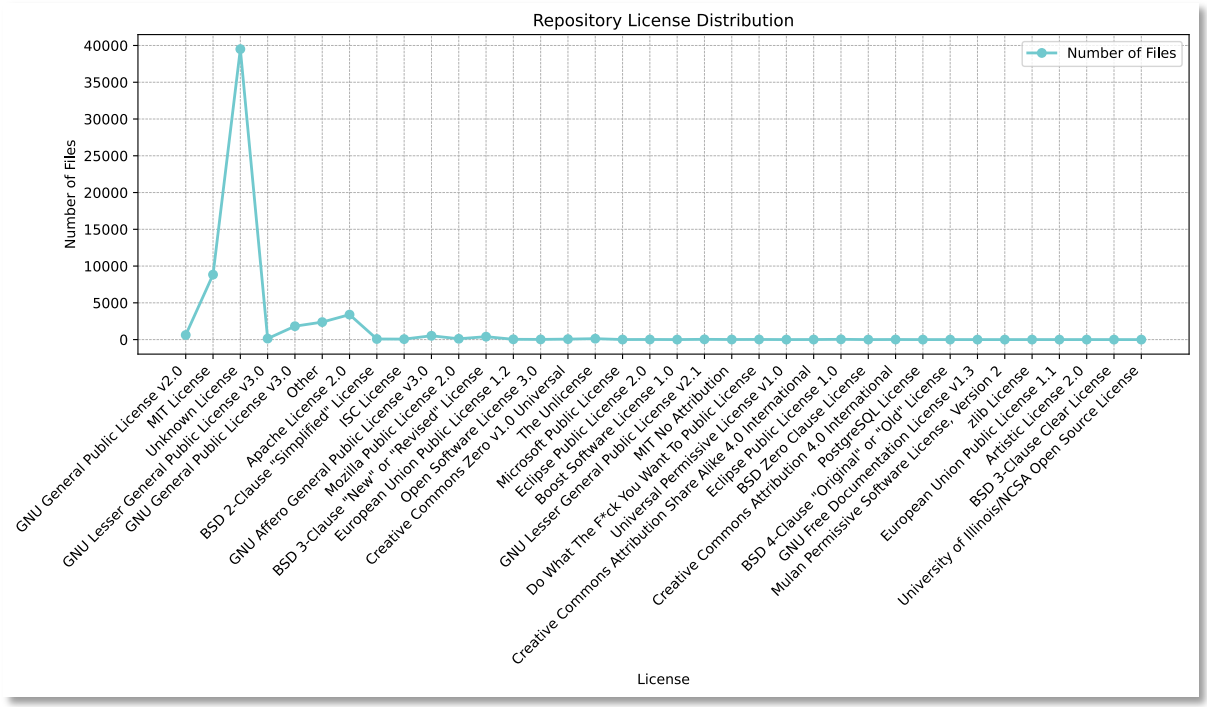
```
.
├── docker
│   ├── docker-commands.md
│   ├── docker-compose.md
│   ├── docker-file.md
│   └── docker.md
├── federated-learning.md
├── machine-learning.md
├── neural-networks.md
├── optimization.md
├── python
│   ├── concurrency
│   │   ├── asyncio.md
│   │   ├── concurrency.md
│   │   ├── coroutines.md
│   │   ├── multiprocessing.md
│   │   └── threads.md
│   ├── formatting
│   │   └── black.md
│   ├── linting
│   │   └── pylint.md
│   ├── syntax
│   │   ├── _.md
│   │   ├── classes.md
│   │   ├── decorators.md
│   │   ├── dictionaries.md
│   │   ├── error-handling.md
│   │   ├── functions.md
│   │   ├── iterators.md
│   │   ├── lists.md
│   │   ├── logging.md
│   │   ├── loops.md
│   │   ├── operators.md
│   │   ├── sets.md
│   │   ├── strings.md
│   │   ├── tuples.md
│   │   └── variables-and-data-types.md
│   └── testing
│       └── pytest.md
├── tensors.md
└── torch.md
```

1.4 Experimentation

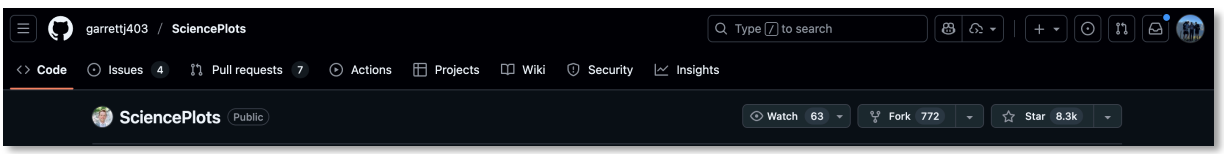
As part of the initial experimentation, I examined the *requirements.txt* file provided to gain an overview of the technologies required for the upcoming tasks. As well as Flower and Torch, which will play a key role in Tasks 2 and 3 respectively, the file also included Matplotlib.



I therefore conducted some initial experiments with Matplotlib to familiarize myself with its plotting functions. To this end, I used an existing GitHub repository database that I had created for my bachelor's thesis and used Matplotlib to generate a few initial visualizations from this data.



During the FAQ meeting, it was also recommended to consider SciencePlots as an alternative visualization library. Following this suggestion, I briefly reviewed the official documentation to understand its capabilities and potential use for later phases.



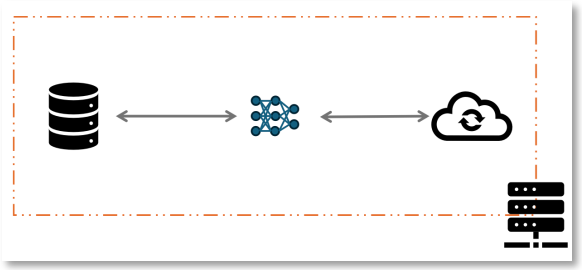
2 Project Phase 2: Machine Learning

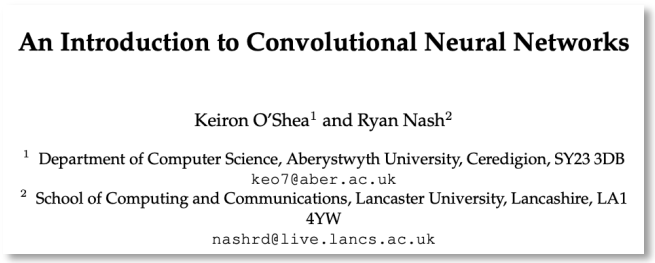
Task 2: 04 November 2025 – 17 November 2025

2.1 Goal

The second phase will establish the foundational environment and toolchain for the centralized machine learning setup (Task 2). During the research phase, I will familiarize myself with the underlying concepts and technologies, and prepare a clean, extensible development environment to support the subsequent federated-learning phases.

The goal of this phase is to implement and evaluate a classic centralized machine learning pipeline as a baseline for later federated learning experiments. I will train a simple Convolutional Neural Network (CNN) on the EMNIST dataset (Extended MNIST) for handwritten character recognition, using a standard optimizer (SGD) and a small number of epochs (e.g. 1–3) as a proof of concept. This provides a reference point in terms of model accuracy and loss, against which the performance of federated approaches (in Tasks 3 and 4) can be compared.

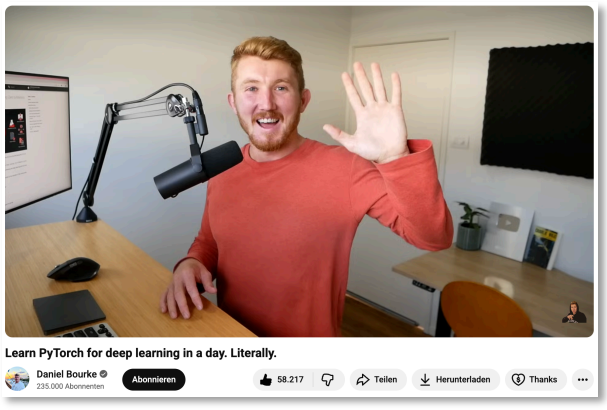


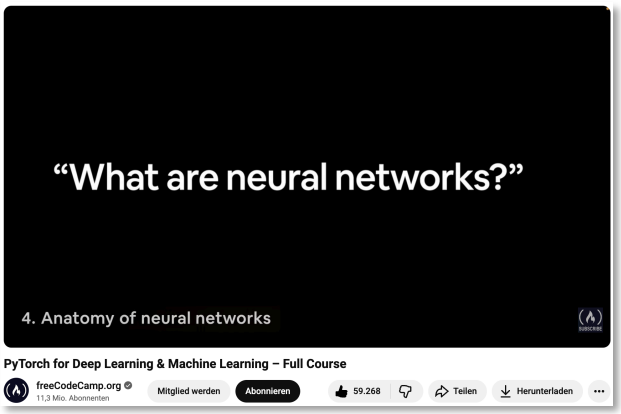


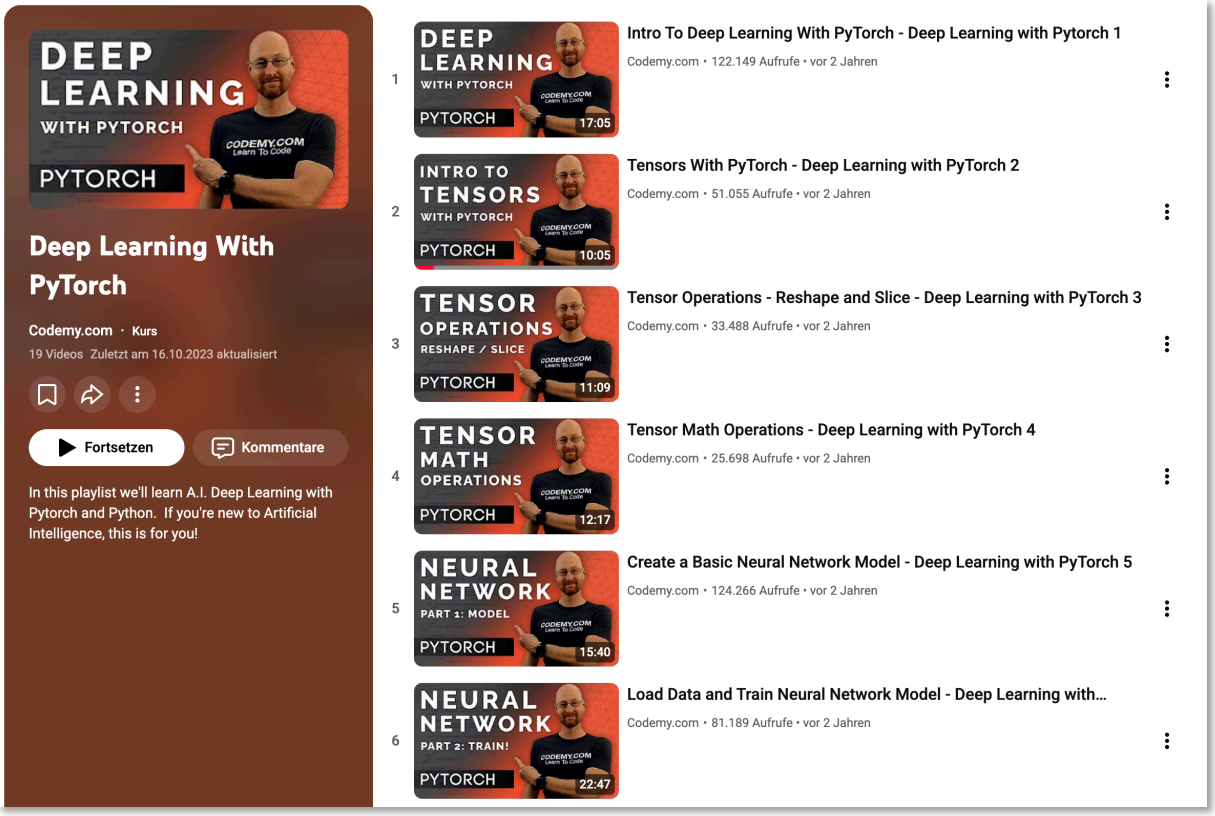
Additionally, this phase aims to ensure all tools and components (data loading, model definition, training loop, evaluation metrics) work correctly in a single-node scenario before moving to a distributed setting. Visualization of the training results (loss and accuracy curves) will be performed to verify that the model learns as expected.

2.2 Research

In preparation for this phase, most of my research will focus on PyTorch, specifically dataset loading, transforms, CNN modules and the training loop. I have also collected a few YouTube videos and playlists to study alongside the PyTorch documentation.

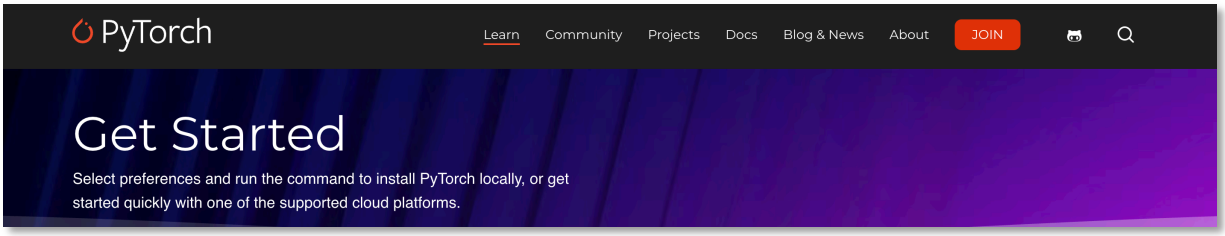


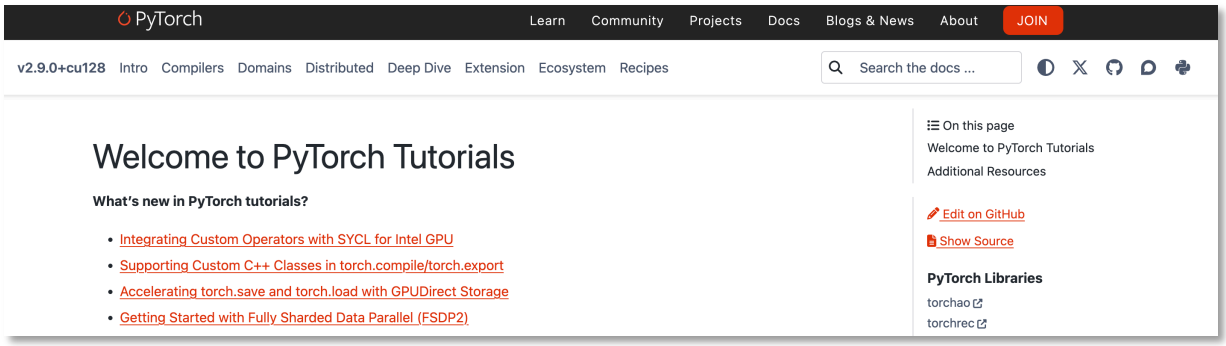


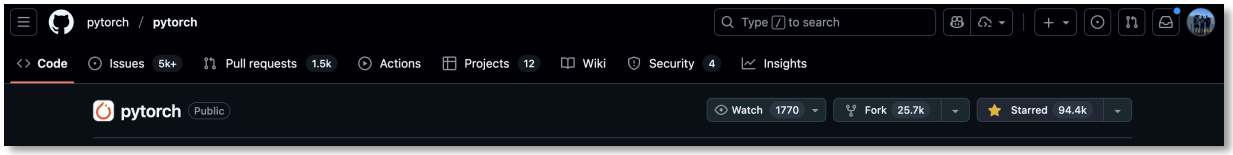


I will prioritize the most relevant videos during the two-week Task 2 window.

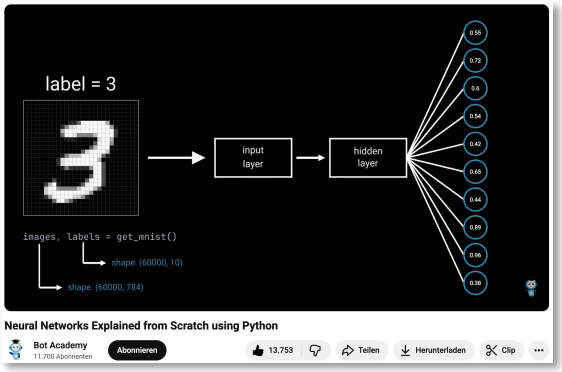
In addition, I will explore the official PyTorch documentation and will examine the PyTorch GitHub repository to familiarize myself with its implementation details and underlying architecture.

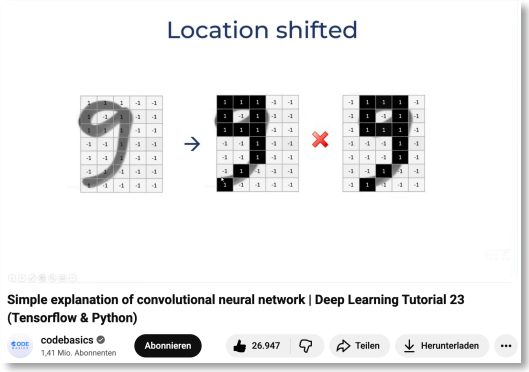


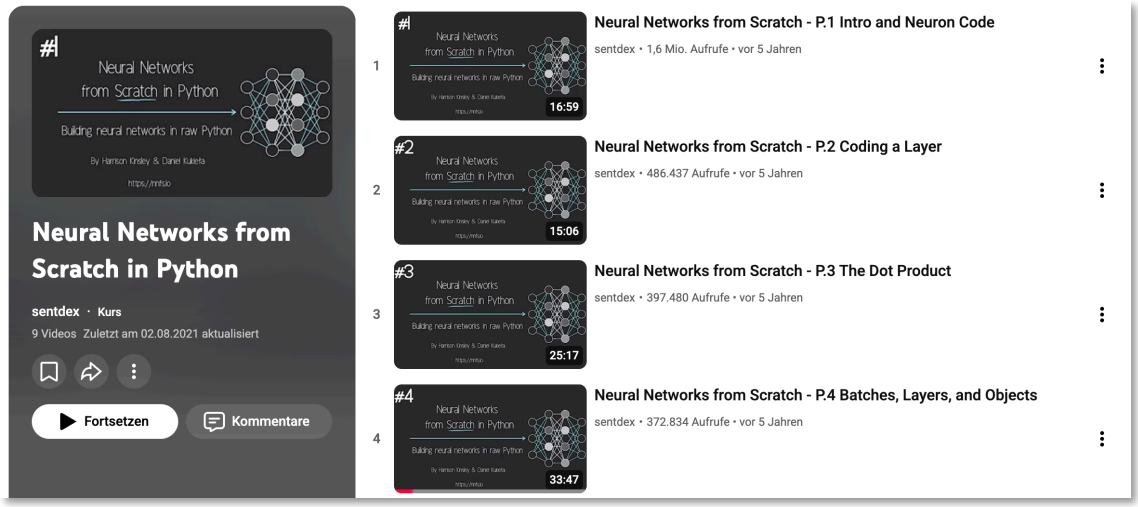


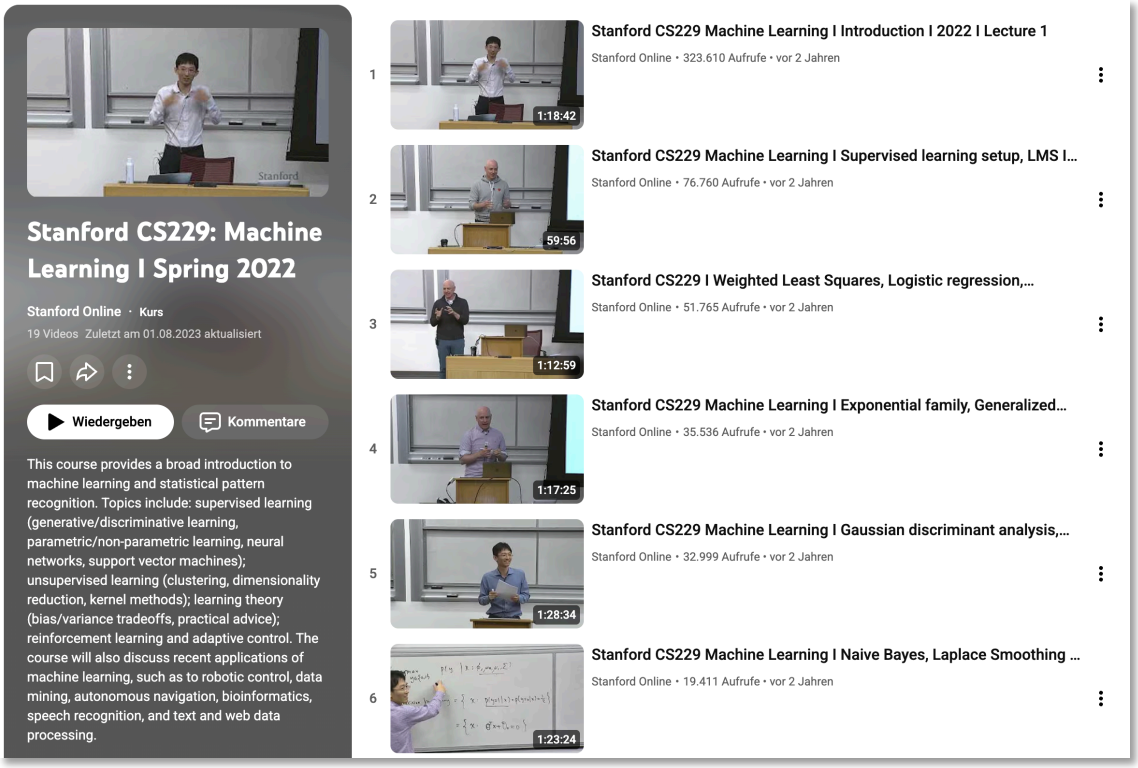


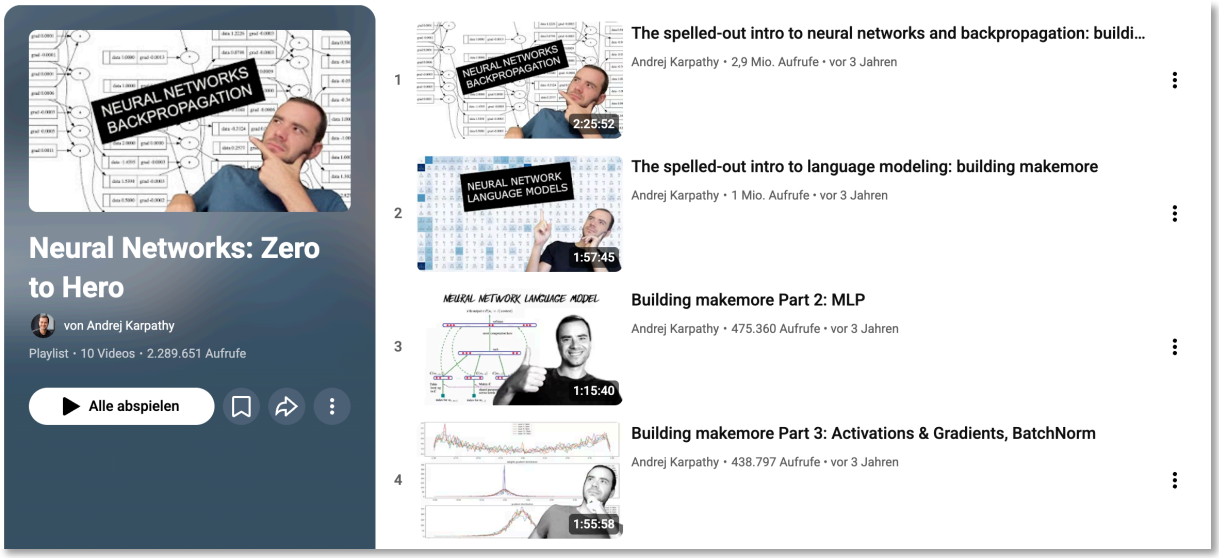
I will do the same for neural networks, which will include re-watching and applying the *Extra Reference Material* e-Learning videos on CNN-based image recognition as well as additional resources.



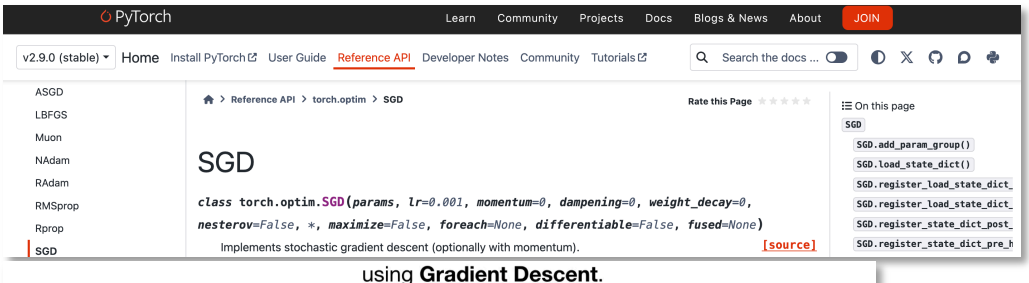


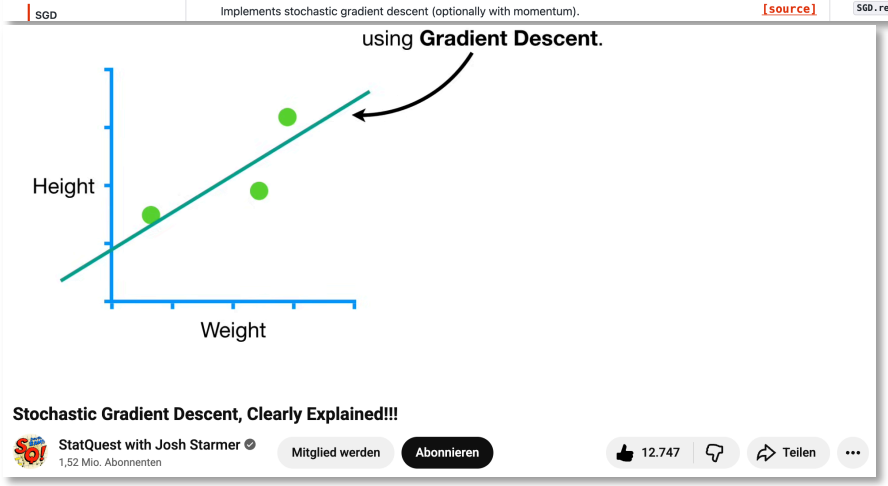


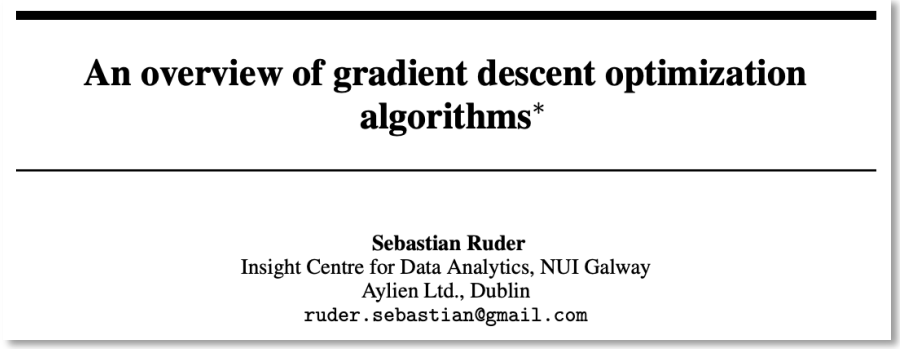




Similarly, I will research the SGD optimizer, as Task 2 explicitly requires it. This includes reviewing its mechanics (learning rate and momentum), the PyTorch API (torch.optim.SGD) and ensuring that my setup enforces the constraints of the assignment precisely (batch size = 30, gradient steps = 3 and epochs = 1).







I will also research the EMNIST dataset and suitable model architectures because Task 2 explicitly requires using EMNIST for centralized training. EMNIST contains handwritten characters derived from NIST Special Database 19 and is converted to 28×28 grayscale images with a dataset structure that directly matches MNIST. I will review the official documentation and confirm input/output shapes and transforms to comply with the assignment.

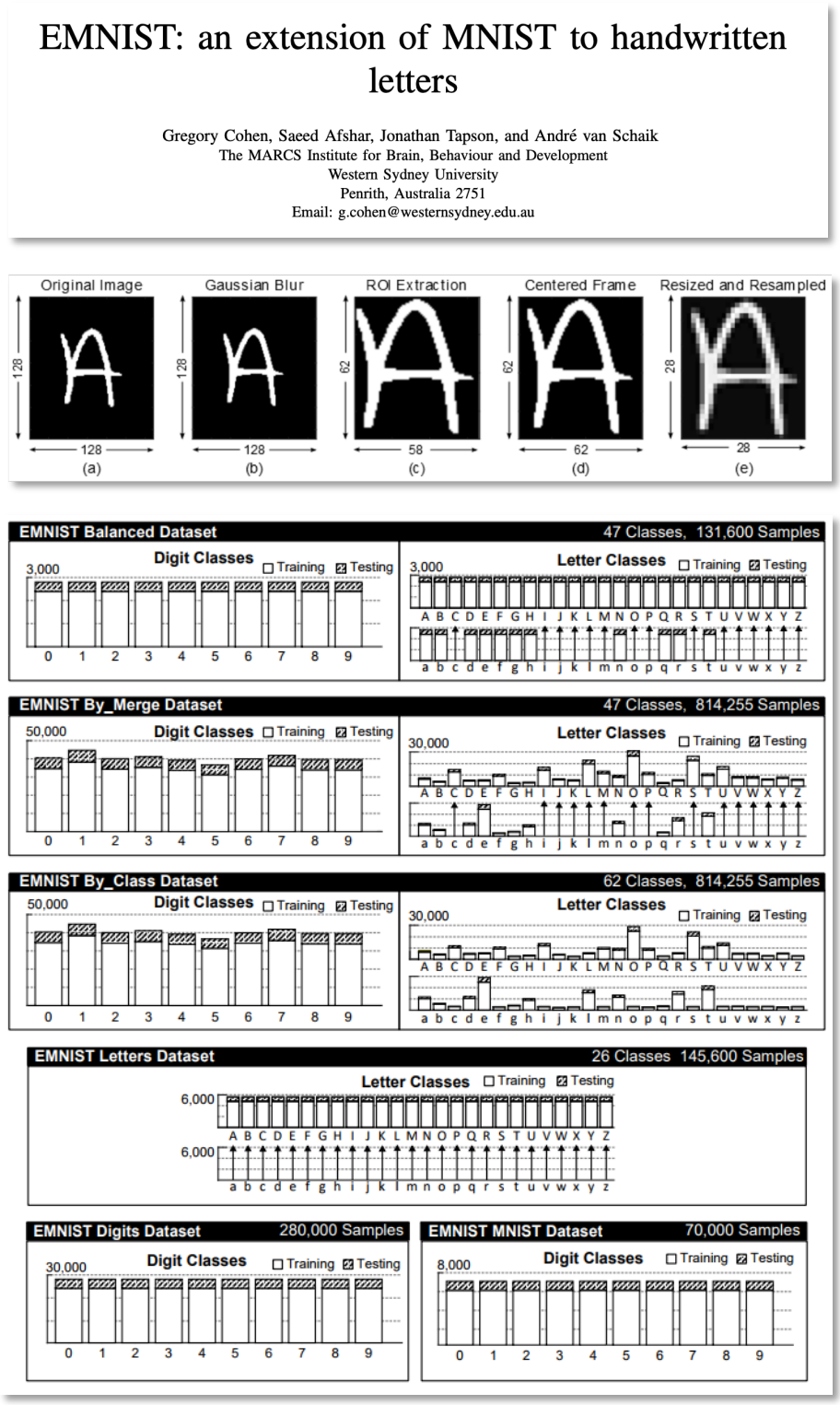
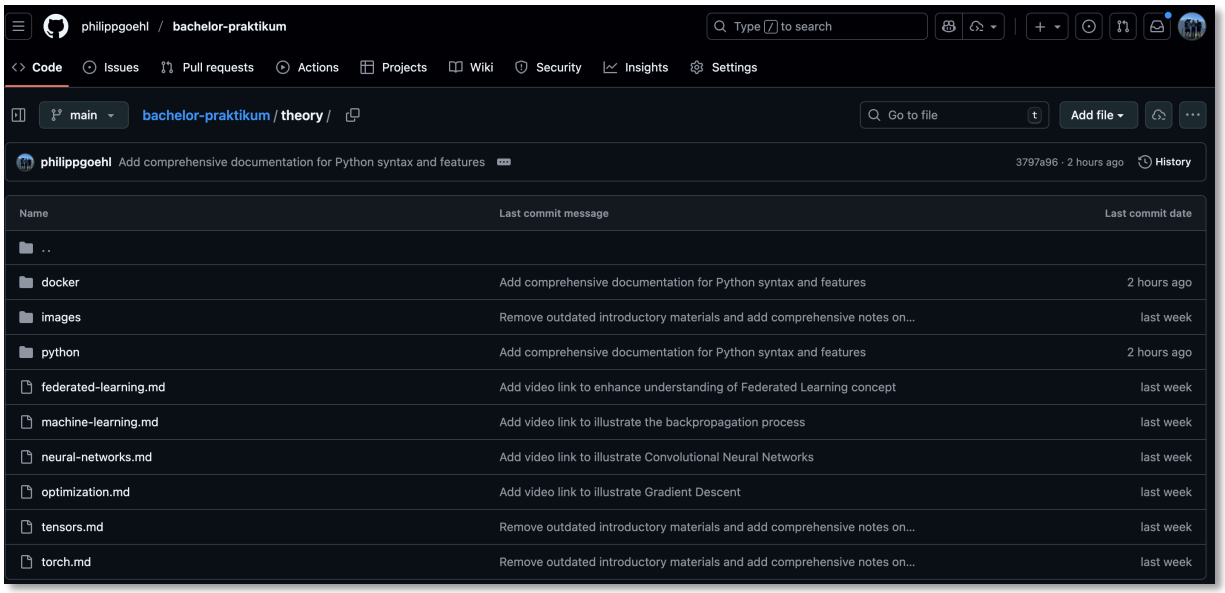


Figure 1: Cohen, Gregory & Afshar, Saeed & Tapson, Jonathan & van Schaik, André. (2017). EMNIST: an extension of MNIST to handwritten letters. arXiv preprint arXiv:1702.05373.

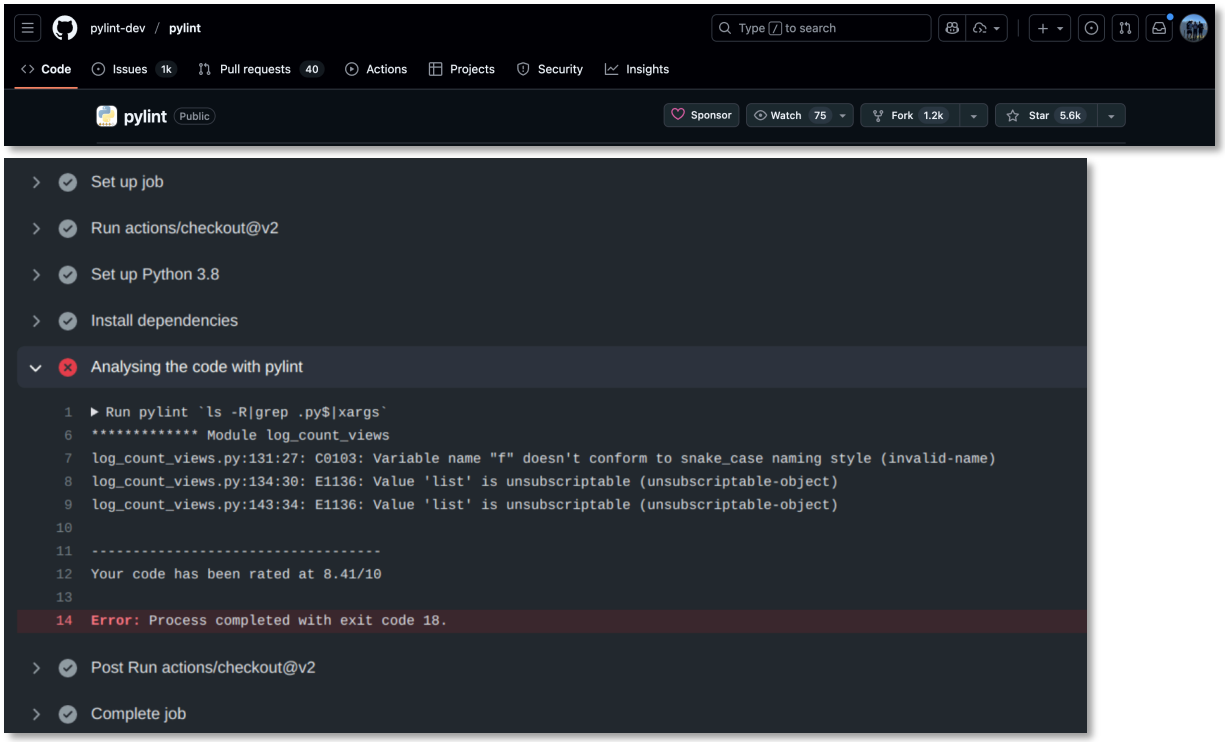
2.3 Documentation

I will keep all theory in a dedicated *theory* folder in the main bachelor's internship repository and document the remaining materials there in concise Markdown.



To keep the code readable and maintainable, I will follow standard Python practices, such as clear naming, focused comments, and comprehensive docstrings for modules, classes, and functions.

I will also set up a GitHub Actions workflow that runs Pylint on every push to identify missing docstrings and potential issues, ensuring that I am notified immediately and can address any problems promptly, thereby maintaining a clean and reviewer-friendly codebase for the correction and grading.



2.4 Visualization

Proper visualization is key to interpreting the performance of the model. During this phase, I will use Matplotlib to plot the training and evaluation metrics. Specifically, after training the CNN on EMNIST, I plan to generate a line plot showing the training loss (and potentially validation loss) over each epoch, as well as the model's accuracy. These plots will confirm whether the model is converging (e.g. decreasing loss, increasing accuracy) and will serve as a baseline graph to compare with federated training in the next phases.

The following illustrations depict the potential configuration of these plots:

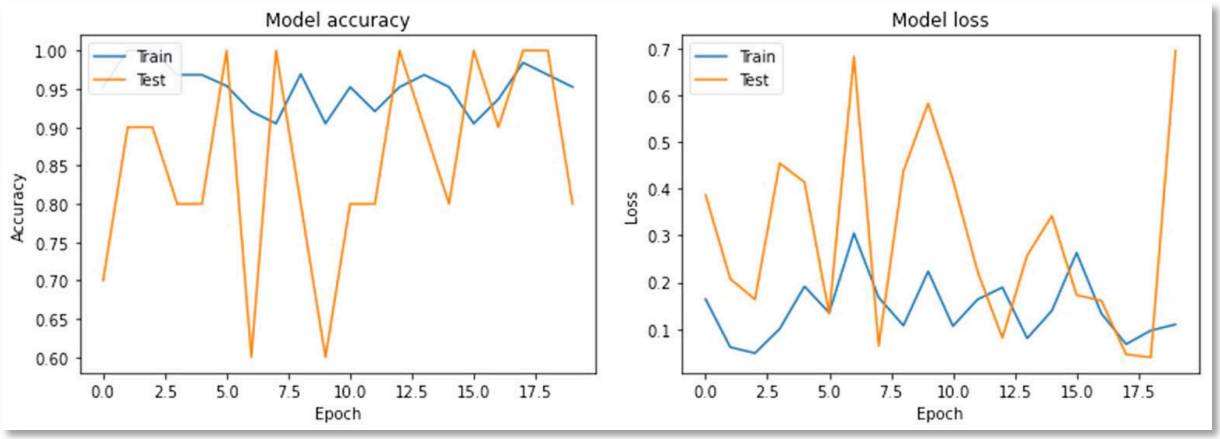


Figure 2: Serel, Ahmet & Ozturk, Sefa & Soyupek, Sedat & Serel, Huseyin. (2022). Deep Learning in Urological Images Using Convolutional Neural Networks: An Artificial Intelligence Study. Türk Üroloji Dergisi/Turkish Journal of Urology. 48. 299-302. 10.5152/tud.2022.22030.

In addition, I intend to visualize the final model performance in more detail: for example, by plotting a confusion matrix of the model's predictions on the test set to see which character classes are often misclassified. This can highlight any weaknesses in the baseline model (for instance, certain letters that the model confuses).

The following plots depict a possible configuration of these confusion matrices:

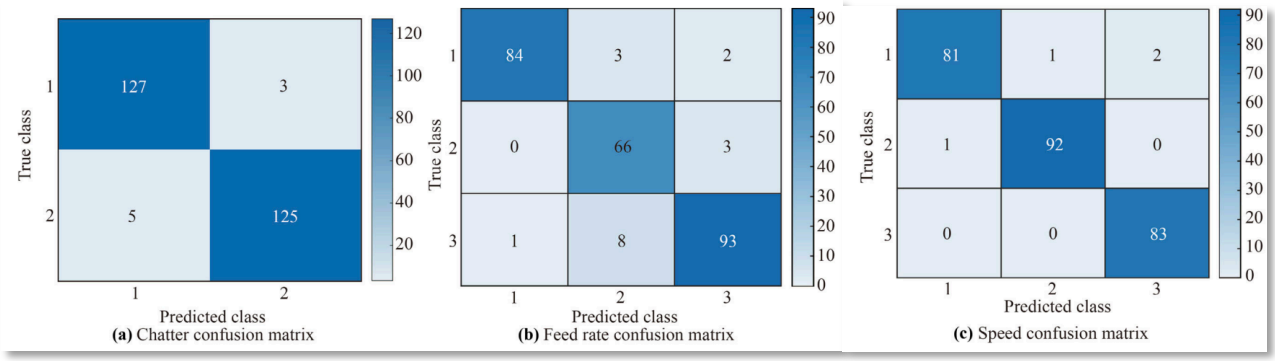
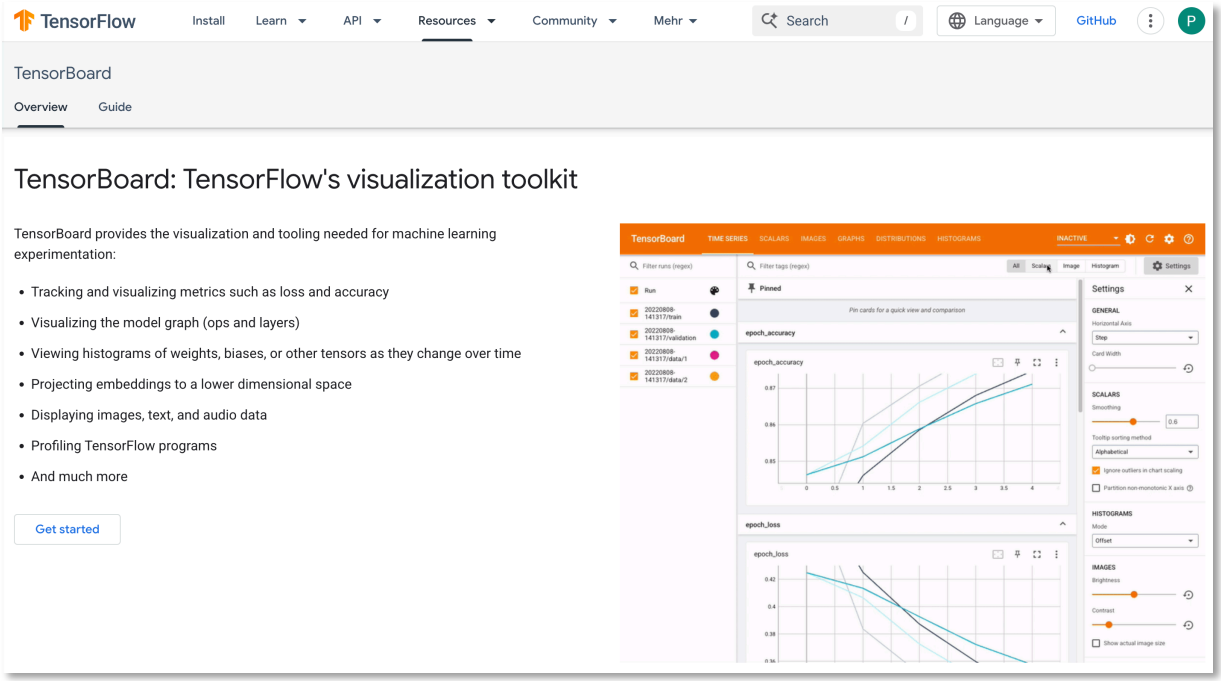


Figure 3: Yang, Ruoyu & Rai, Rahul. (2019). Machine auscultation: enabling machine diagnostics using convolutional neural networks and large-scale machine audio data. Advances in Manufacturing. 7. 10.1007/s40436-019-00254-5.

If time permits, I may also experiment with tools like TensorBoard for real-time monitoring of training metrics, although the initial focus is on static plots for simplicity. The emphasis is on creating clear, concise charts that can be included in the documentation, making it easy to communicate how well the centralized model learned from the data.



2.5 Experimentation

For experimentation in Task 2, I will first run the official baseline exactly as required: EMNIST with SGD, batch size = 30, gradient steps = 3, and 1 epoch, and I will save the results and simple plots (loss and accuracy).

After that, I will try a few small, safe variations to learn what helps: for example, I will test a slightly different learning rate (a bit lower or higher) and do a quick “can it learn?” check by training on a very small subset to see if the model can quickly fit those samples.

If time allows, I will also try a tiny rotation on the images (only a few degrees) to see if it changes the result, but I will keep the official baseline unchanged for fair comparison.

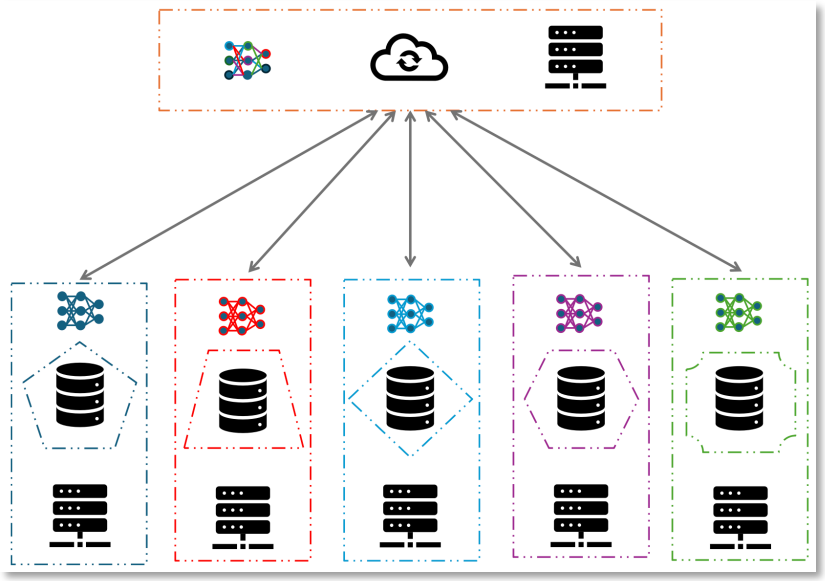
I will use a fixed random seed so the runs are repeatable, and I will write down short notes (what I changed and what I saw) together with the saved CSV metrics and plots. This keeps the work simple and beginner-friendly but still shows that I explored the task a bit beyond the basic run.

3 Project Phase 3: Federated Learning

Task 3: 18 November 2025 – 01 December 2025

3.1 Goal

The goal of this phase is to extend the previously built centralized machine learning pipeline into a federated learning setup. In this federated scenario, the EMNIST dataset will be split identically and randomly across 10 clients, and model training will occur locally on each client's subset of data.



A central parameter server will be introduced to coordinate the training: it will collect model updates from the clients and aggregate them to form a new global model, which is then sent back to all clients for the next round.

I will implement the Federated Averaging (FedAvg) strategy as required, meaning after each local training step (each client processes one mini-batch of data), the server will average the clients updated weights to produce the global model for the next round. In effect, one communication round consists of every client performing a single gradient update on its mini batch (e.g. batch size 30 as before), and then the server averaging these updates. Therefore, one round is roughly equivalent to processing a "global mini-batch" composed of 10 local mini-batches (one per client).

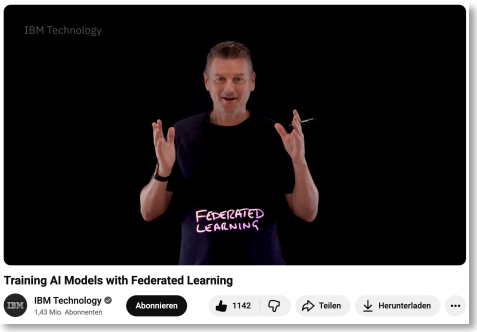
Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan **Eider Moore** **Daniel Ramage** **Seth Hampson** **Blaise Agüera y Arcas**
Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

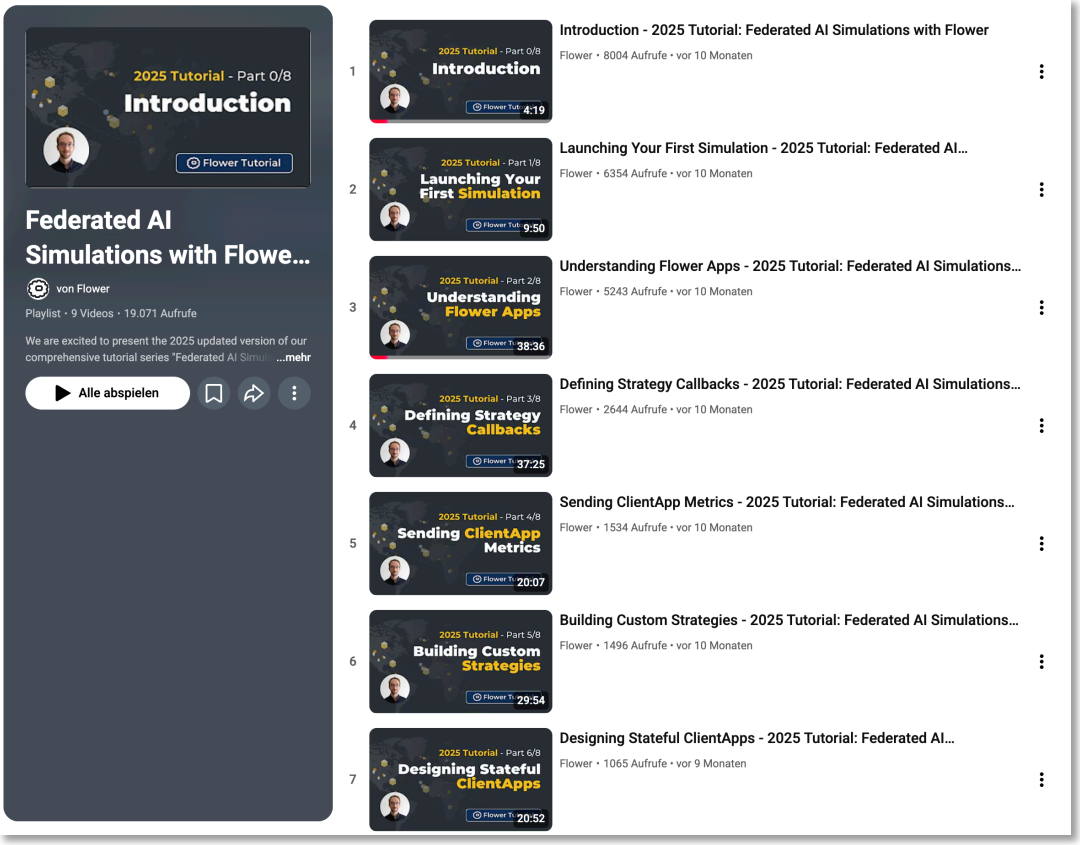
By the end of this phase, I intend to have a working federated training loop that mirrors the centralized baseline in functionality, allowing us to compare performance (e.g. accuracy and convergence speed) between the centralized and federated approaches. An overarching objective is to confirm that with identical data distributions on all clients, the federated learning approach can reach a similar accuracy to the centralized model, while laying the groundwork for the more challenging non-IID scenario in Task 4.

3.2 Research

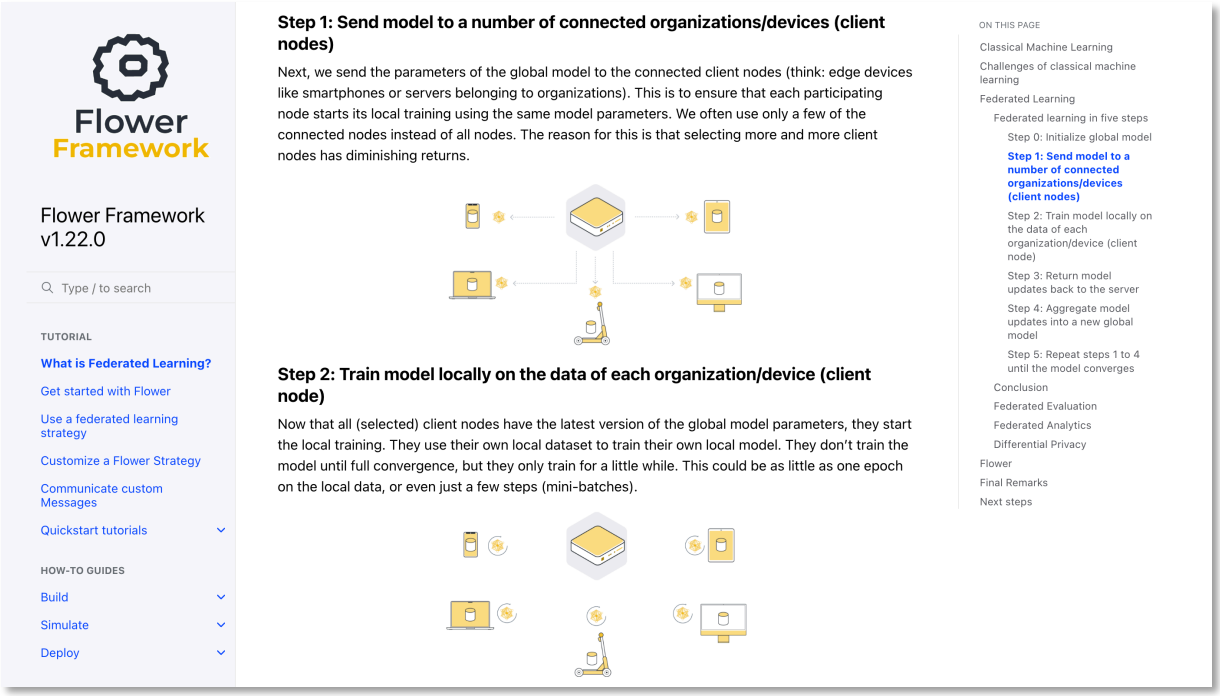
In preparation for implementing federated learning, I will revisit the fundamental concepts and materials on FL that were introduced in the e-Learning course. This includes reviewing the *Module Slides* and the *Extra Reference Material* on federated learning and the FedAvg algorithm, to solidify my understanding of how local training and global aggregation should be orchestrated.



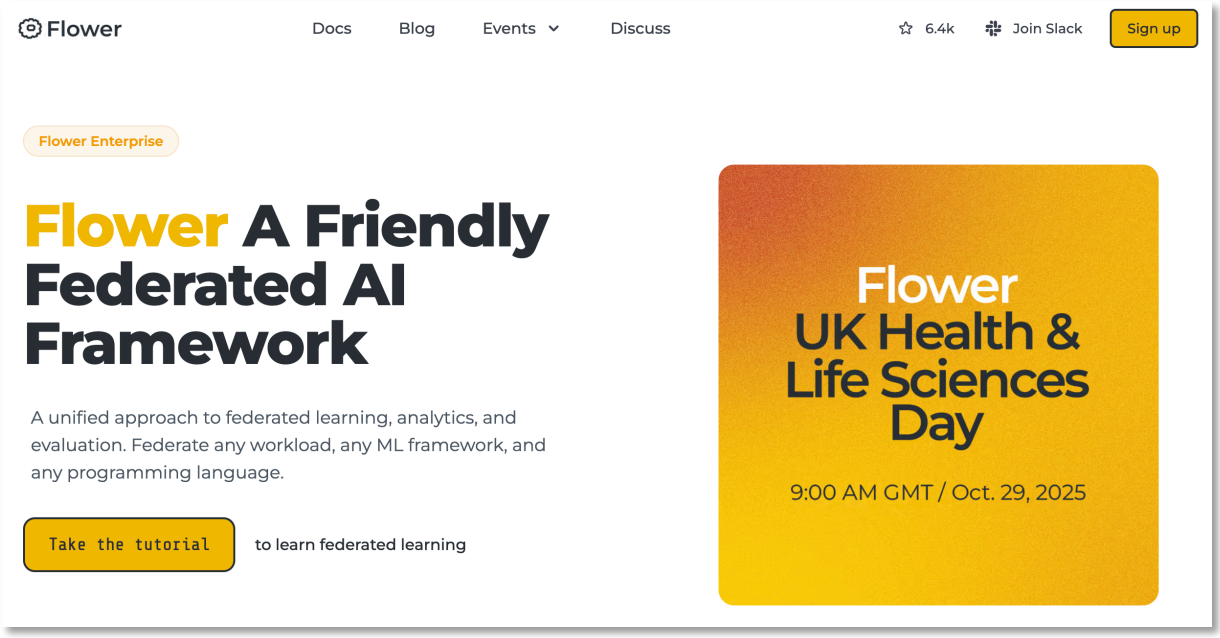
Furthermore, I will watch various YouTube videos on this topic to gain a more practical and intuitive understanding of the concepts and their implementation.

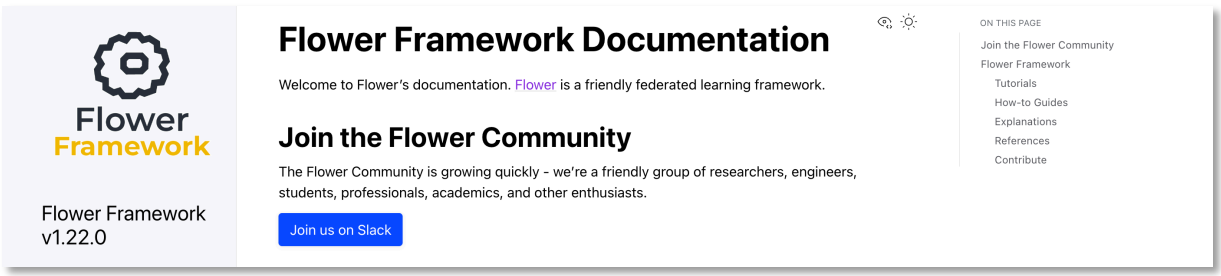


In addition, I will explore practical resources on federated learning frameworks, particularly Flower, which is a federated learning framework mentioned in our requirements.



Besides this, I will go through the official Flower tutorials on federated learning as well as the documentation on implementing federated learning systems with Flower.





In conjunction with this, I will research how to partition the EMNIST dataset for our purposes. Since Task 3 specifies that data should be distributed identically across clients (i.i.d. distribution), I plan to perform a random split of the EMNIST training set into 10 equal parts. Each client will thus receive roughly the same number of samples and a similar distribution of classes, ensuring that any performance differences are not due to data skew (that aspect will be explored in Task 4).

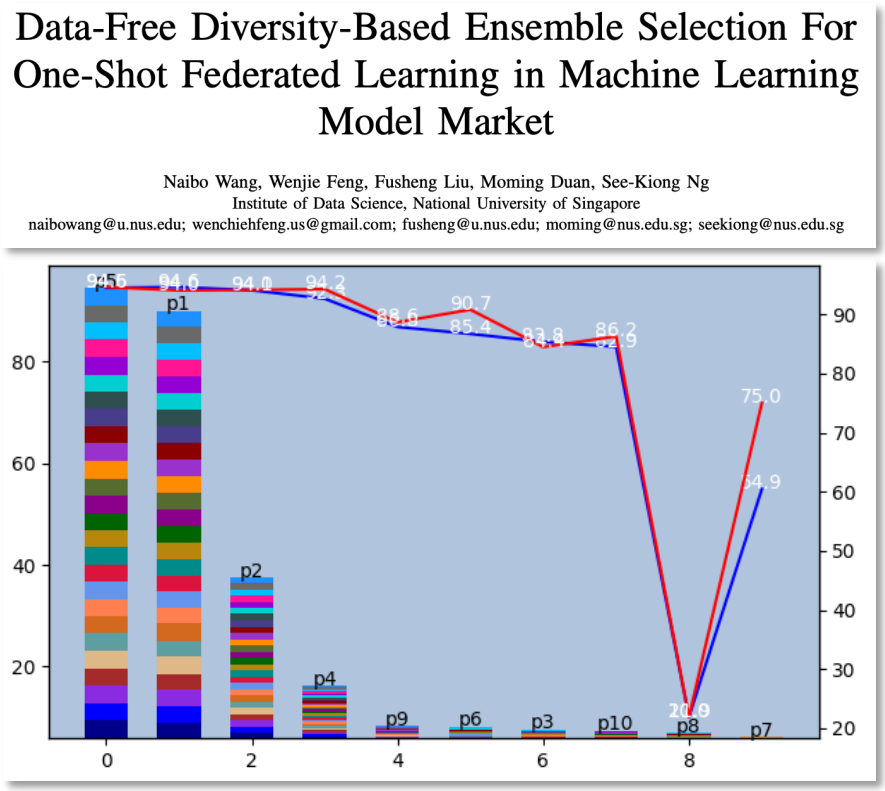


Figure 4: Data distribution for the iid-dq partition of the EMNIST letters dataset.
Wang, Naibo & Feng, Wenjie & Liu, Fusheng & Duan, Moming & Ng, See-Kiong. (2023). Data-Free Diversity-Based Ensemble Selection For One-Shot Federated Learning in Machine Learning Model Market. 10.48550/arXiv.2302.11751.

3.3 Documentation

As in Task 3, I will continue to maintain thorough documentation of both the theoretical background and the code changes. In the project's *theory* folder, I will update the *federated-learning.md* file with notes specific to this phase. For example, describing the federated learning workflow, the FedAvg strategy, and how data is partitioned among clients. Any insights gained from the research (such as definitions of new terminology like "rounds" or details on how FedAvg works) will be concisely noted there for future reference.

On the code side, I will ensure that all new modules and functions related to the client-server setup are well documented with docstrings and comments. This includes documenting how the federated training loop is structured, how the server aggregates weights, and how clients are simulated (especially if using a framework like Flower, I will document the configuration and usage of its API in our context).

3.4 Visualization

Visualization remains crucial in this phase to interpret the outcomes of federated learning and to compare them with our previous results. After implementing the federated training, I plan to plot several metrics to evaluate how the global model is learning over the communication rounds.
Specifically, I will use Matplotlib (as in the previous phase) to create line charts for metrics such as the global model's accuracy and loss on a validation or test set versus the number of rounds. This will show whether the federated training is converging (e.g. decreasing loss and improving accuracy over rounds) and how rapidly it does so in comparison to the centralized training baseline.
A plot for the *Global Model Accuracy vs. Number of Rounds* could look as follows:

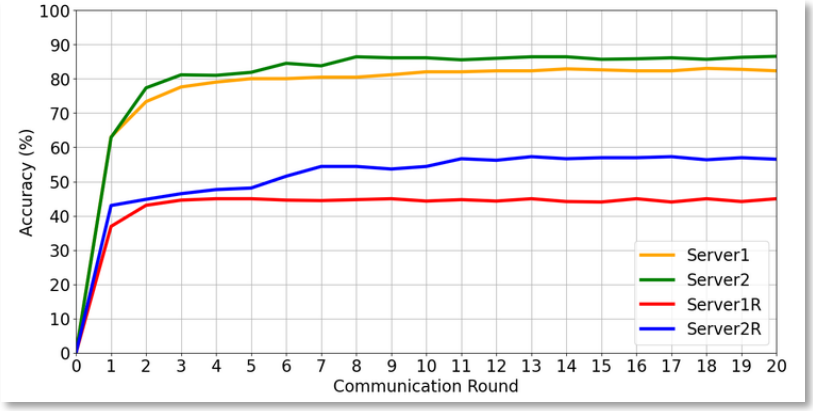


Figure 5: Wehbi, Osama & Wahab, Omar & Mourad, Azzam & Otrok, Hadi & Alkhzaimi, Hoda & Guizani, Mohsen. (2023). Towards Mutual Trust-Based Matching For Federated Learning Client Selection. 1112-1117. 10.1109/IWCMC58020.2023.10182581.

One visualization I intend to include is a direct comparison of the centralized vs. federated learning curves. For example, I might overlay the accuracy curve from the Task 2 (centralized 1-epoch training) with the accuracy curve from Task 3's federated training on the same axes or simply place them side by side. This can illustrate differences in convergence speed or final accuracy. I anticipate that due to the federated approach (with one mini-batch per client per round), I might need several rounds for the federated model to reach the accuracy that the centralized model achieved in just one epoch. Plotting these will confirm if that's the case.
A plot comparing Centralized vs. Federated Learning Accuracy over Rounds could look as follows:

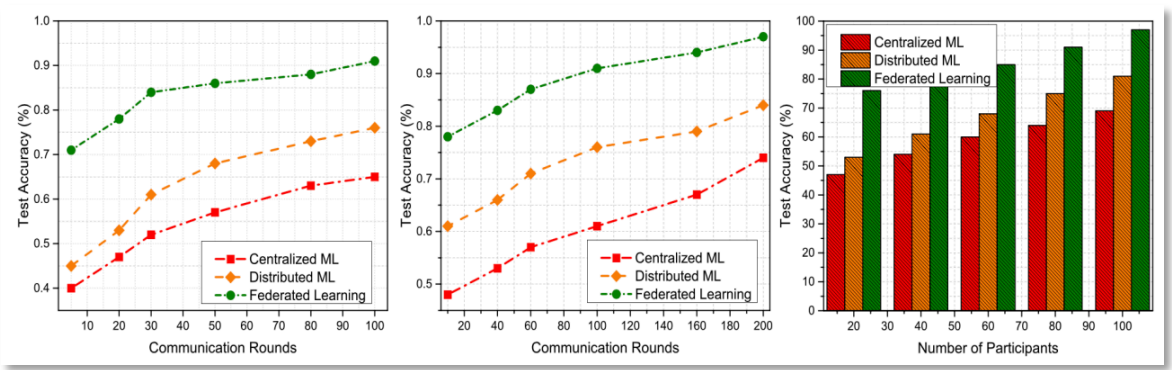


Figure 6: Asad, Muhammad & Moustafa, Ahmed & Ito, Takayuki. (2020). Federated Learning Versus Classical Machine Learning: A Convergence Comparison.

Besides performance over time, I will also visualize the concept of the federated setup itself in the documentation. If possible, I may also include a small schematic in the report showing how one round works (for instance, arrows from server to clients distributing the model, and arrows back from clients to server sending updates).

3.5 Experimentation

For experimentation in Task 3, I will begin by running the federated training with the exact specifications given: 10 clients, each performing one local gradient step per round (with the same batch size of 30 as used in Task 2), using the FedAvg strategy to aggregate, and continue this for a suitable number of rounds.

I will record the performance metrics (such as the global test accuracy after each round) to see how the model improves. This first run will serve as the primary result to analyze and as a correctness check that the federated pipeline is functioning as expected.

After establishing the baseline federated run, I plan to try a few minor variations to deepen my understanding. One experiment will be to adjust the number of communication rounds: for instance, if the initial run used a fixed number of rounds, I might try increasing the rounds to see if the global model's accuracy continues to improve and how it eventually compares to the centralized model's accuracy. Conversely, I might test what happens if I limit the number of rounds (though likely the accuracy would drop, it's still insightful to observe how many rounds are needed to reach a certain performance level).

Another variation could involve the local training workload on each client. While the assignment specifies one mini-batch per round (which I will follow for the main result), I'm curious to see the effect if each client did slightly more work locally (for example, one full epoch on its local data per round, turning FedAvg into a more typical scenario). If time permits, I will run an experiment where instead of 1 mini-batch, each client processes its entire local dataset for a couple of epochs before averaging and then compare the outcomes. This can highlight the differences in convergence when clients do more local training (though this deviates from the exact task requirements, it would be an interesting learning exercise for me).

Throughout these experiments, I will use a fixed random seed to ensure reproducibility, and I will carefully log the settings of each run. I will also monitor if all clients are contributing as expected. Since the data is evenly split, I don't anticipate any one client dominating the updates, but it's worth verifying that the weight updates from all 10 clients are being properly aggregated. If any anomalies appear (e.g. learning stalls or a client's update seems to negatively affect the model), I will investigate by printing intermediate metrics per client.

4 Project Phase 4: Non-IID Dataset

Task 4: 02 December 2025 – 15 December 2025

4.1 Goal

The objective of this final phase is to simulate a more realistic federated learning scenario by introducing non-IID data distribution and observing its impact on training. In real-world federated settings, each client's data is not identically distributed. Some devices may have predominantly different classes or quantities of data. To emulate this, I will distribute the EMNIST dataset unevenly by label across the 10 clients. In practice, this means each client will receive a subset of the training data that is biased towards certain character classes (for example, one client might mostly have images of specific letters, while another has different ones). This label-skewed partitioning will create disparate local datasets and test the robustness of our federated algorithm under statistical heterogeneity.

Another planned change in this phase is to increase the amount of local training per client before aggregation. In Task 3, each client performed a single mini-batch update (batch size 30) per communication round. Now, each client will perform training on 3 local mini-batches (essentially a small local epoch of 3 batches) before sending its update to the server. The reason for this change is to simulate a scenario where clients might do a bit more work locally (which is often the case to reduce communication frequency), and to observe how this influences convergence when data is non-IID.

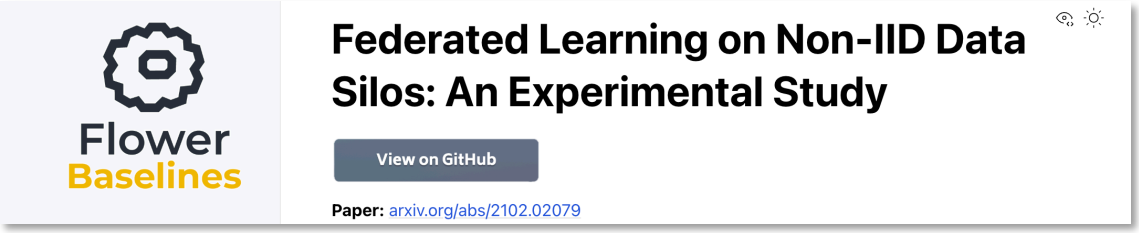
Non-IID data in Federated Learning: A Survey with Taxonomy, Metrics, Methods, Frameworks and Future Directions

Daniel M. Jimenez G., David Solans, Mikko A. Heikkilä, Andrea Vitaletti, Nicolas Kourtellis, Aris Anagnostopoulos, Ioannis Chatzigiannakis, *Senior Member, IEEE*

The overall goal is to examine the performance of the federated learning algorithm (FedAvg) under these more challenging conditions and compare it with the previous (IID data, fewer local updates) scenario. Key metrics of interest will be the global model's accuracy and convergence behavior: I will specifically compare how the accuracy achieved in this non-IID setup differs from the accuracy in the prior IID setup (Task 3), given the same model and similar total training effort.

Ultimately, this phase will validate whether our federated system can still learn effectively when facing uneven data distribution, and how much of an accuracy gap exists relative to the idealized IID case. The results will be visualized to provide a clear comparison between the federated learning performance in IID versus non-IID scenarios.

4.2 Research



I will be researching practical methods to simulate a non-IID split of the EMNIST dataset. One straightforward approach (and the one suggested by the assignment) is to divide the dataset by label. I will likely implement this by sorting or grouping the EMNIST training samples by their class label and then splitting this grouped list into ten parts for the ten clients.

This could mean, for example, each client gets a disproportionate concentration of certain character classes (Client 1 might get mostly the first few letters of the alphabet, Client 2 the next few, and so on). I want to ensure that each client still has a sufficient number of samples to train on, but the distribution of classes will be highly uneven between clients. While this *label-partitioning* is one common way to create a non-IID scenario, I'm aware that there are multiple degrees and types of non-IID. For a thorough understanding, I will have to look into alternative partitioning schemes as well.

4.3 Documentation

Since Task 4 builds directly on the existing codebase and concepts, I will focus on inline code comments and concise explanations of any new or changed components (for example, describing the non-IID data partition function or any new parameters in the training loop). The theoretical background of federated learning and the FedAvg process has already been documented in earlier phases, so here I will only add notes specific to the non-IID scenario as needed. Any observations about the non-IID effects or decisions (such as how data was split) will be noted in the final report and code comments.

4.4 Visualization

Visualization will be crucial for comparing the federated learning performance between the IID and non-IID scenarios. As with the previous phases, I will use Matplotlib to create clear and informative plots of the results. The primary plot will likely be a line graph of the global model accuracy over communication rounds, similar to what was done in Task 3. On this graph, I intend to overlay the accuracy curve from Task 3 (IID data, 1 batch per round) with the accuracy curve from Task 4 (non-IID data, 3 batches per round). By plotting these together (with appropriate labels or a legend), I can directly visualize differences in learning dynamics. I expect this comparison to highlight whether the non-IID setup learns more slowly, reaches a lower peak accuracy, or exhibits more fluctuations in accuracy between rounds.

For instance, if the Task 3 (IID) curve showed a smooth improvement to a certain accuracy after a number of rounds, the Task 4 curve might show a slower rise or a plateau at a slightly lower accuracy. Seeing them on one plot will make such differences immediately evident.

In addition to the accuracy-over-rounds plot, I will likely plot the loss over rounds for the global

model as well, since loss can provide insight into convergence (and potential divergence) that accuracy alone might miss. A smoothly decreasing loss in the IID case versus a possibly oscillating or higher loss in the non-IID case would reinforce the impact of data heterogeneity.

Beyond performance metrics, I am considering adding a visualization to illustrate the data distribution across clients in the non-IID scenario. For example, a bar chart or histogram could show how many samples of each class each client has. This would visually confirm the degree of inequality in the data (e.g. Client 1 might have mostly class 'A' and 'B' images, Client 2 mostly 'C' and 'D', etc.). Including such a chart in the report can help to grasp why the learning might be harder.

Finally, once the federated training in the non-IID setting is done, I might also visualize the final model performance in a similar way to Task 2's analysis. For example, I could generate a confusion matrix of the global model's predictions on the EMNIST test set. This could be particularly interesting to see if certain characters are now misclassified more frequently than in the IID case, potentially because those classes were underrepresented or isolated on particular clients during training.

Comparing such a confusion matrix to that of the IID-trained model (Task 3's global model) might highlight specific weaknesses introduced by the non-IID training. All visualizations will be saved and included in the documentation or presentation.

4.5 Experimentation

In this experimental phase, I will implement the federated learning setup with the non-IID data partition and observe the results step by step.

First, I will run the baseline experiment as specified: 10 clients, each with an unequal, label-skewed portion of the EMNIST data, performing 3 local batch updates per round, using the FedAvg aggregation on the server after each round. I will keep the other parameters consistent with Task 3 for a fair comparison. For instance, the same CNN model architecture, the same optimizer (SGD) and learning rate, and the same batch size of 30 for each local update. The number of communication rounds will be chosen to allow the model to train roughly an equivalent amount as before (or until I see the accuracy leveling off).

For example, since each round now processes

```
3 batches × 10 clients × 30 images each = 900 images in total
```

it might take on the order of

```
60,000 samples / 900 images ≈ 67 rounds
```

to equal one epoch across the entire dataset (which has 60,000 samples), whereas in Task 3 with

```
10 clients × 30 images each = 300 images per round
```

it took

```
60,000 samples / 300 images = 200 rounds
```

for an epoch.

I will likely run the training for a sufficient number of rounds (perhaps 100 or more) to see clear trends, and I will log the global accuracy and loss after each round. This main run will show how the model performance evolves under non-IID conditions. I will then compare the final accuracy from this experiment to the final accuracy from Task 3's IID experiment (and even to Task 2's centralized baseline). This direct comparison addresses the core question: *how much does non-IID data with limited local updates affect the model's accuracy?*

After obtaining the baseline results, I plan to conduct a few additional experiments and variations to deepen the analysis (time permitting). One variation will be to adjust the number of communication rounds to see if the non-IID model can eventually reach the same accuracy as the IID case given more training. If, for instance, the non-IID run of 100 rounds results in slightly lower accuracy than the IID run did, I might extend the training to 200 rounds to see if it catches up or if it asymptotes below the IID performance. This will inform whether the accuracy gap is due to slower convergence (needing more rounds) or a fundamental limitation of the FedAvg approach under non-IID data. Conversely, I might also check what happens if I limit rounds (though likely it would just have lower accuracy, it's mostly to observe the trend).

Another experimental angle is exploring different degrees of non-IID distribution. The assignment allowed flexibility ("this is up to us"), so I could try a couple of approaches:

- Extreme label partitioning (which is the main case): each client gets a disjoint set of labels. This is expected to be the most challenging scenario for FedAvg.
- Partial label overlap: for comparison, I could create a scenario where clients have some overlap in labels (e.g., each client has data from, say, 5 out of the 10 total classes, with some classes present on multiple clients). This would still be non-IID but slightly less extreme. Running the training in this setting and comparing accuracy could show how performance improves as data distribution becomes less skewed.
- Quantity skew: an alternative non-IID scenario is when clients have vastly different amounts of data (even if label proportions are similar). Although the assignment emphasizes uneven by label, for curiosity I might simulate one client with a large chunk of the data and others with very small amounts, to see how that impacts the FedAvg updates (e.g., does the dominant client's data overwhelm the results?).

Additionally, I am interested in the effect of multiple local batches (3 in our case) on the training dynamics. To isolate this effect, I could run a mini-experiment where I revert to 1 local batch per round *but still with non-IID data* and compare it to the 3-batch case. This would help determine if the 3 local updates per round are helping or hurting the global model when data is skewed. Intuitively, doing more local batches might increase the divergence between client models each round (since they train longer on their biased data before averaging), potentially making the global aggregation less effective.

On the other hand, it also means more training is done per round, which might speed up reaching a certain accuracy if divergence doesn't get too large. Observing these outcomes will be enlightening. If 3 local batches prove too unstable, one might try a smaller learning rate or fewer local batches; if it works well, it shows FedAvg can tolerate a bit of local training even in non-IID settings.