The final project is part of the oral exam, which means you are *not allowed to work in groups.* The purpose of this project is that you get hands-on experience on most topics of the course and to show that you can present and explain the results of your work. To this end, please submit your code, plots, tables etc. to a bitbucket repository to which at least one organizer of the lecture has access.[1]

In the first 15 minutes of the exam, you will present your approach and the results such that we can discuss it together. It is important that *your evaluation builds the basis for discussion and scientifically analyzes which are the important aspects and characteristics of your approach*—you should present your findings in a convincing manner.

To give the project presentation some structure, you will have to prepare a few presentation slides. Your slides should consist of a motivation slide, slides detailing your approach (2-3) as well as slides for your results (2-3). You are allowed to submit at most 5 slides. *Don't go overboard with your slides.* They are intended to make your presentation coherent.

Optimization of a Convolutional Neural Network

You are tasked with the optimization and analysis of the performance of a neural network on the Kuzushiji-MNIST[2] dataset(s). How you optimize the given network is up to you. For example, you could optimize the hyperparameter of the networks optimizer, apply neural architecture search or a joint approach. In the end, you should convince us that you indeed improved the performance of the network when compared to a provided default configuration. To this end, you could consider the following tasks:

- Measure the default performance of the given network;
- Apply hyperparameter optimization or algorithm configuration to determine a well-performing configuration;
- Determine the importance of the algorithms parameters;
- Extend the configuration space to include preprocessing steps/dropout/skip connections/...;
- Use multi-fidelity optimization;
- Plot the confusion matrices;
- Plot the performance of automated algorithm design approaches over time;
- Construct an EPM from observed data and predict well performing configurations;
- Use a learning to learn approach to learn how to optimize the network.

Please note that you do not have to necessarily apply all these methods – pick the ones that you think are most appropriate. We provide a repository (`https://bitbucket.org/biedenka/ml4aad_ws18_project`) for you to fork[3], in which we will upload the following:

- A pytorch implementation to access the *KMNIST* and *K49* datasets
- A baseline parameterized network to optimize (also written in pytorch)
- An example script to show you how to train and evaluate an network based on the default configuration

You are allowed to use all scripts and tools you already know from the exercises; however, you are not limited to them.

[1] For bitbucket invite user `biedenka`.
[2] `https://github.com/rois-codh/kmnist`
[3] `https://help.github.com/articles/fork-a-repo/`

You should respect the following constraints:

- **Metric:**
  - The final performance has to be measured in terms of classification accuracy. We require that you report your methods final performance on at least the *KMNIST* dataset.
- **Experimental Constraints:**
  - Your code for making design decisions should run no longer than 86400 seconds (without validation).
  - Fully training a network can not train longer than 20 epochs.
  - Your approach uses at most 16GB memory per CPU on the provided remote machines.
- **Implementation Constraints:**
  - You can freely extend the baseline implementation provided to you. However, the code should always be able to run the given default network.
  - All changes to the network have to be (hyper-)parameterized in the given configuration space. You are not allowed to use manual tuning to achieve better performance. All improved design decisions have to be made (somehow) automatically.
- **Grading Guidelines:**
  - The *KMNIST* data set is by far easier than *K49*. You can freely choose between these two datasets, but if you work only on *KMNIST* you can achieve at best a 2.0 as a grade for the project part.
  - The *KMNIST* website[4] provides some results of baseline implementations. We expect that you achieve an accuracy of at least 90%. You get bonus points by achieving better results close to the results on the homepage.

We provide a Google spreadsheet[5] in which you can upload your current progress. This sheet will not be monitored by us but gives you the opportunity to compare your results. This might help you identify early if your approach is working well or not.

This project is due on 24.02.19 (23:59 CET). Submit your presentation for the exam by sending a PDF of your slides to `biedenka@cs.uni-freiburg.de`. **No teams are allowed for the final project.**

[4] `https://github.com/rois-codh/kmnist`

[5] `https://docs.google.com/spreadsheets/d/1dVxWquJ0OXmZTHFQV0wrsNYPsCZdgcfjrlvaNtIBqRw/edit?usp=sharing`