Part 1

- Mean average deviations over 10 runs: 1.017, 1.048, 0.836, 1.269, 1.120, 2.535, 2.024, 0.701, 0.291, 0.522

Architecture

- I have chosen two layers because results using three were slightly worse.
- The dense layers have 32 and 16 units each, because the input dimension is rather small and higher values achieved higher loss and mean averaged deviations.
- Loss and metric: mean square error and mean average error respectively. Both are suited for a regression task as this one.
- For training I used 100 epochs because higher values increased MSE and MAE.

Data preparation

- `hour.csv` consists of 17380 rows and 17 columns. The first row describes a header and the first two columns count the hours and list the dates. Both can be omitted because an implicit count is given by reading the file line by line into an array and the dates could be of interested if there were more years given. On some holidays there is surely an differing demand (either higher for a bicycle tour or lower due to less commuting) but two years are to sparse to learn day correlated features, especially when weather conditions are sufficiently documented.

Optimization ideas

- Normalizing input data
- Using Keras Tuner

Documentation for colleagues

- I personally like separating config/control related variables in a separated file because these values determine the behaviour of the program and should be easily accessible, i.e. no opening a large file, no (long) scrolling, not scattered across a file, etc.
  One could argue that every value, for instance the activation functions are important, but in my opinion some are more relevant because they are more inclined to change.

Miscellaneous

- No use of convolutional layers, because the data is not spatially related.
- No recurrent, LSTM or GRU architecture because there is no sequential correlation within an input.

Part 2

- Scaling properties: In principal given. As I mentioned, I decided against more complex layers like CNN and LSTM, so adding additional fully connected layers isn't much of a big deal.

- I would address these problems by adding more dense layers and increase the number of weights in each layer. I would also consider to adapt the network to running on GPUs by following the hints on Keras documentation – Doing preprocessing synchronously on-device vs. asynchronously on host CPU.
  It also possible to use distributed deep learning techniques.

- Limits and drawbacks: Additional hardware is required, which implies further costs and some training on this hardware. More data is a priori not bad but efforts to retrieve new information or improve existing approaches is not guaranteed either. Depending on the performance over-engineering when there is little room for improvement is an outcome which should be avoided. In the bike sharing example, this is the case because the mean average deviation is roughly around 1 and the mean of requested bikes is 190. There is potential for optimization but in my point of view the additional work won't be worth it.

- I have no hands-on experience with technologies processing data in the terabyte range.