SOME COLLECTIVE FEEDBACK ON GA1

A question:

When do you decide that network A is

- more powerful
- better/worse

than network B?

Network power: poor example

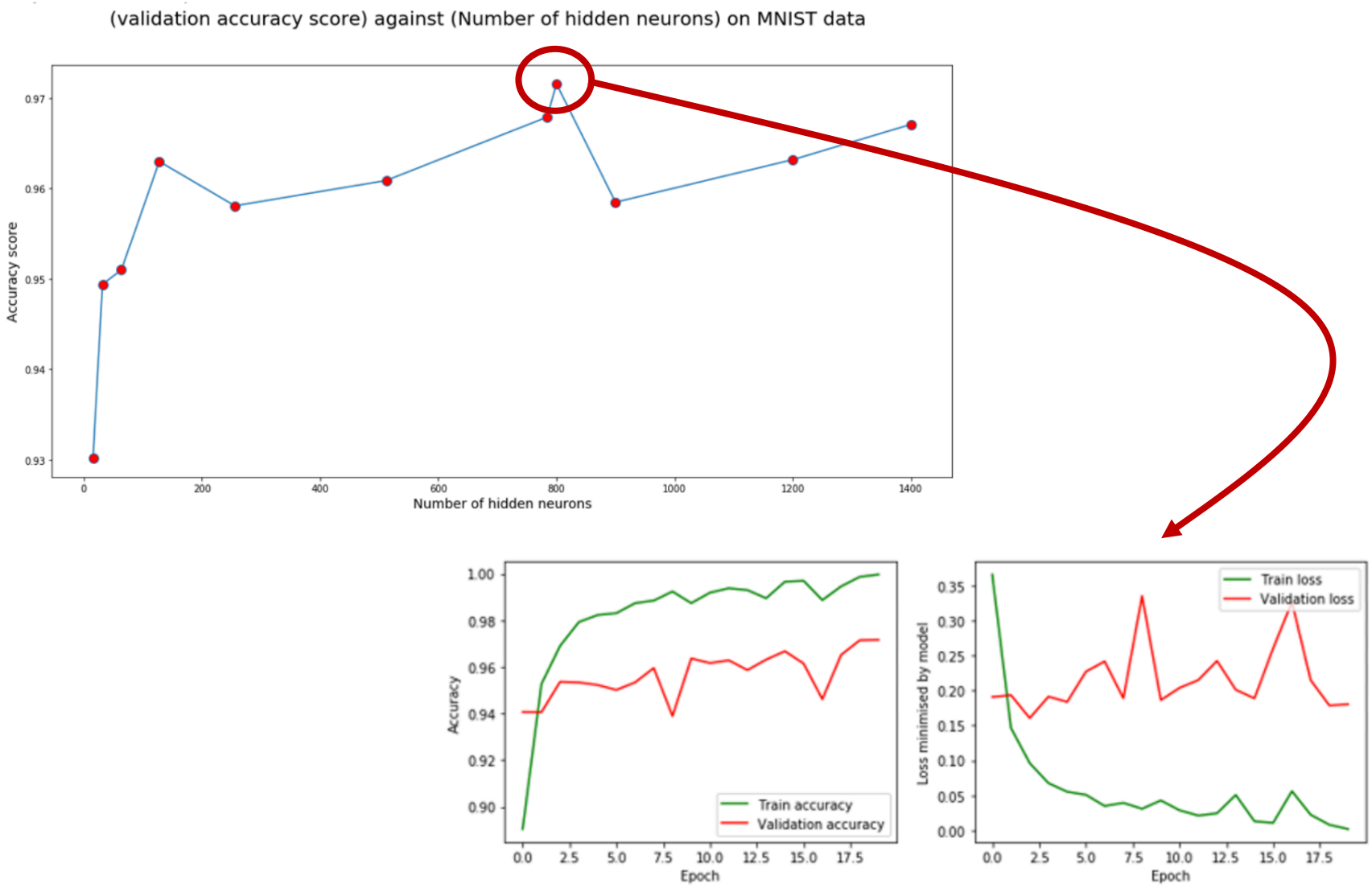


Figure A.3: Validation curve for 800 neurons with 2 hidden layers

Network power: another poor example

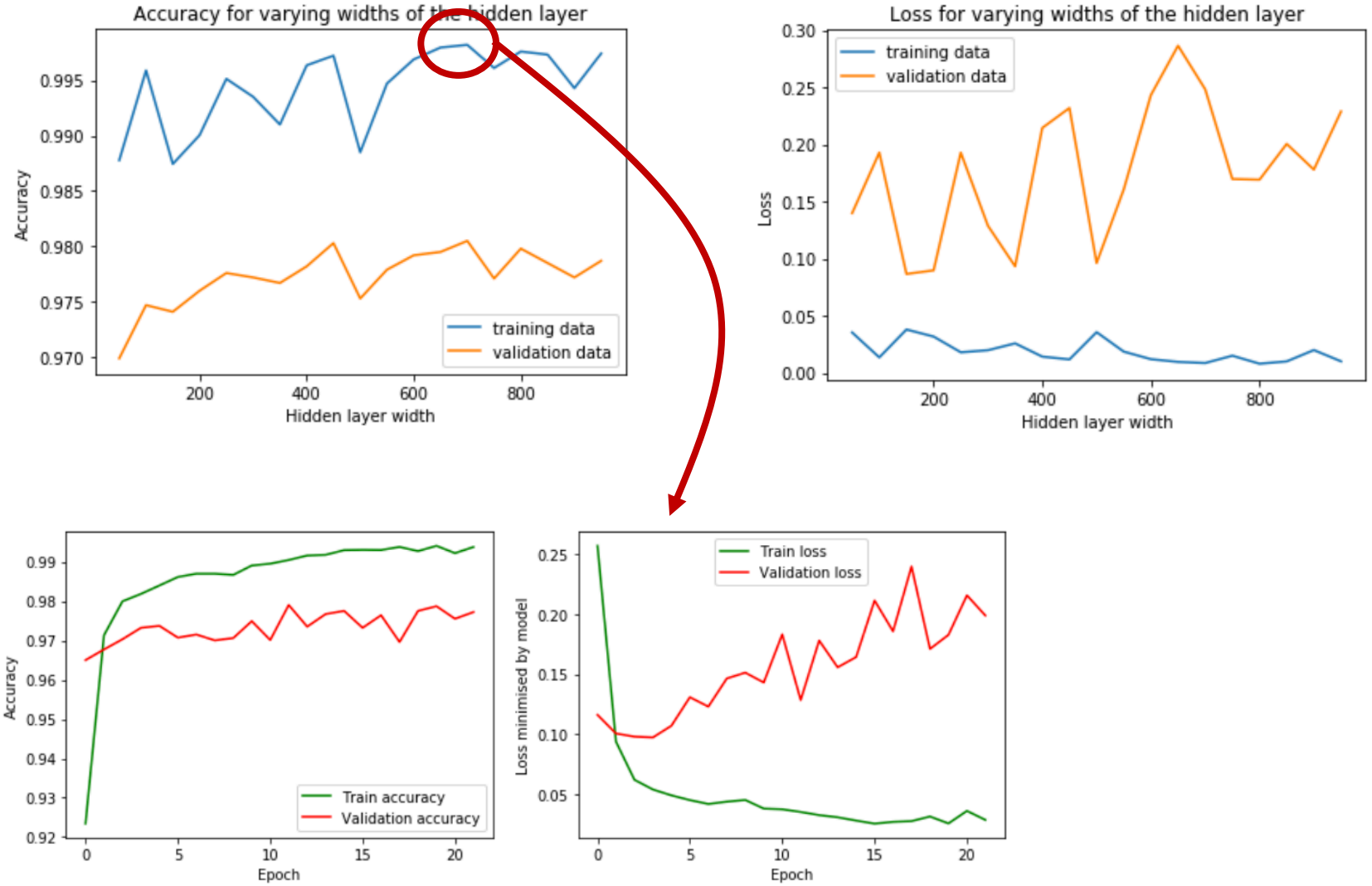
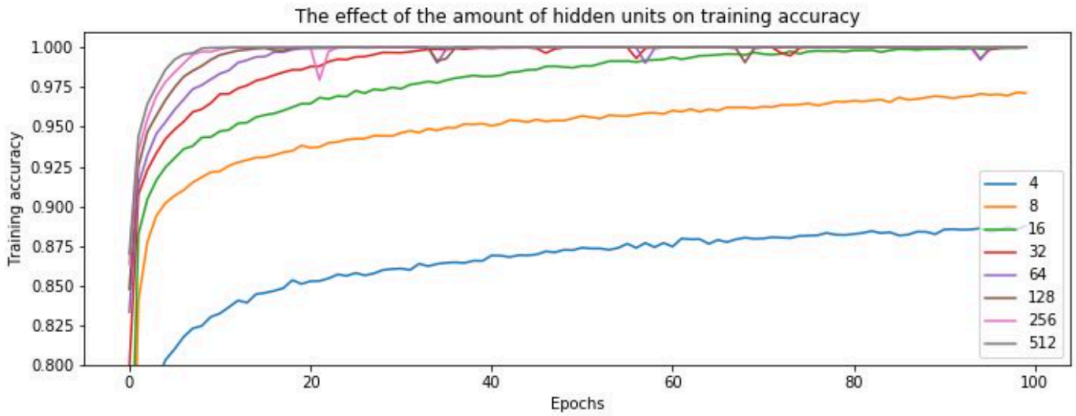


Figure 3: History plot of a single hidden layer model with width equal to 700

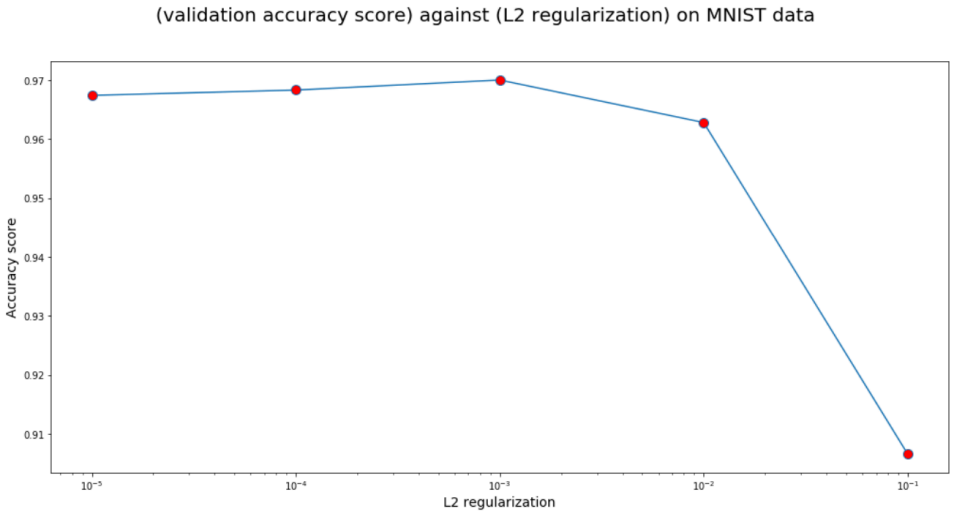
Convergence first!!!

- Must not be perfect, but acceptable:
  - training scores must be sufficiently smooth
  - measure you want to optimise must have converged
  - model power: look at convergence of training scores
  - model generalisation: look at convergence of validation scores
- When tuning model power:
  - not only training accuracy, but speed of convergence
  - can only decide by looking at validation curves (so no gridsearch-based selection)

Gridsearch: good examples



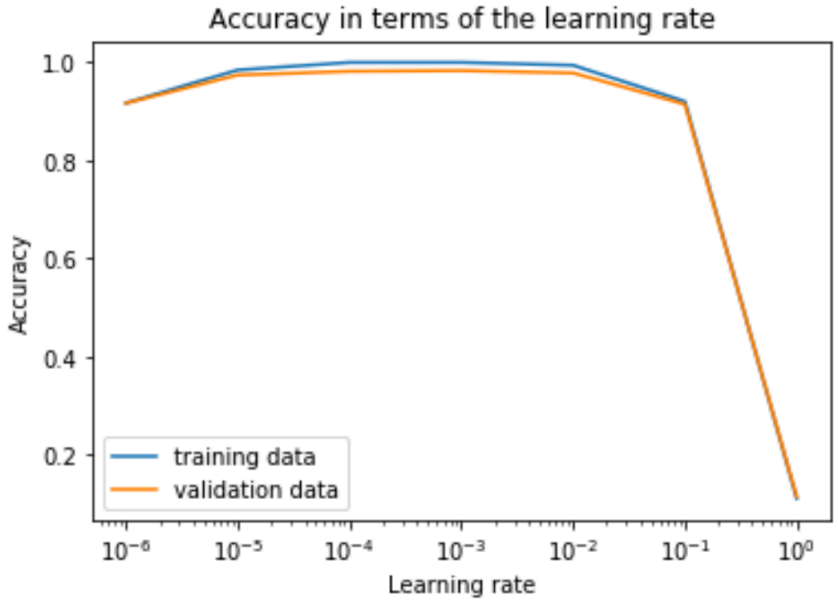
- IF you want to do gridsearch:
  - set convergence parameters based on an intermediate point of your sweep
  - run gridsearch but save histories
  - afterwards: check convergence of outer points in grid (curves)
  - or: write your own loop and select best option after inspecting validation curves

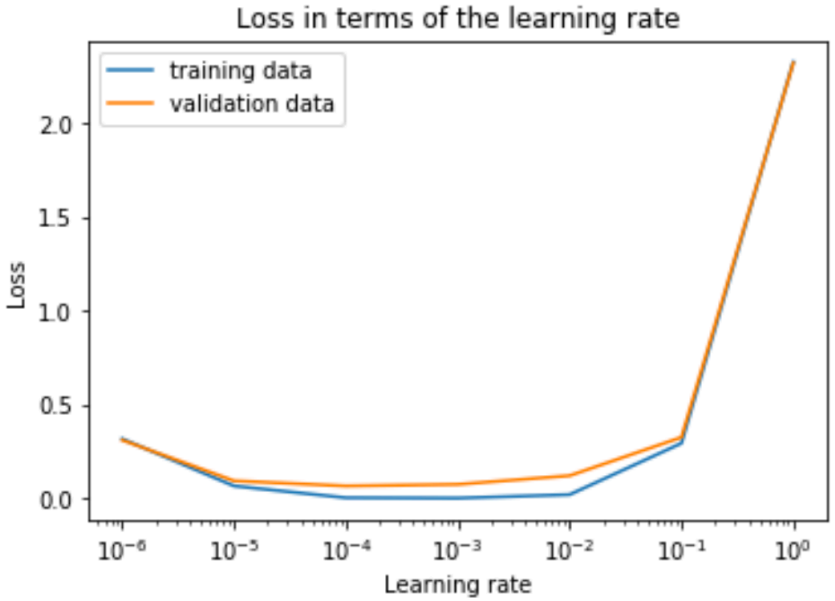


If you do it right, grid search validation curves look like this
But always check convergence!!

Do NOT sweep convergence params

- learning rate, batch size, epochs





- These params are NOT tuned to improve generalisation, but to improve convergence!!!

Tune batch size or learning rate??

- Batch size affects the accuracy of the gradient direction
  - if too small: inaccurate gradients, jaggy curves
  - can also be too large: local minima!!
  - smaller batch sizes may yield “wider minima” which generalise better
  - the “what fits in memory” rule only applies if you are memory-limited! Otherwise: the smallest batch size that does the job
- (Initial) learning rate affects the size of the steps
  - this is related to the complexity of the loss surface
  - many ‘sharp’ local minima: many steep slopes, need lower learning rate
  - since you don’t know what the landscape looks like, it’s worth a try
  - smaller (initial) learning rates usually do not make matters worse, but need more epochs

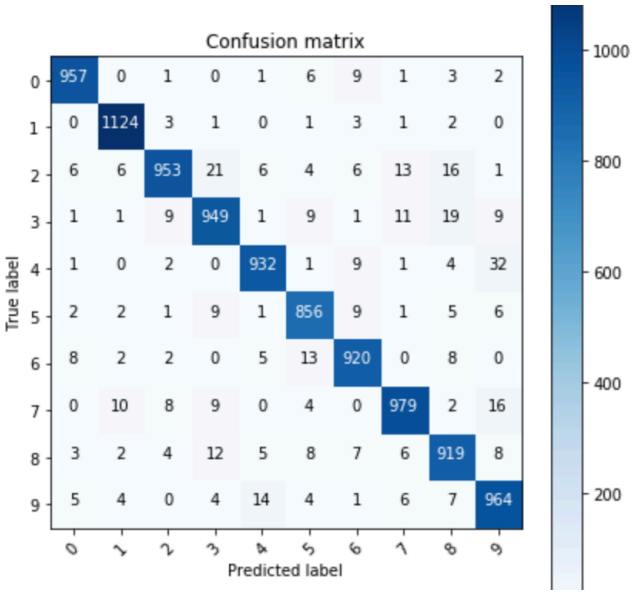
Reporting numbers

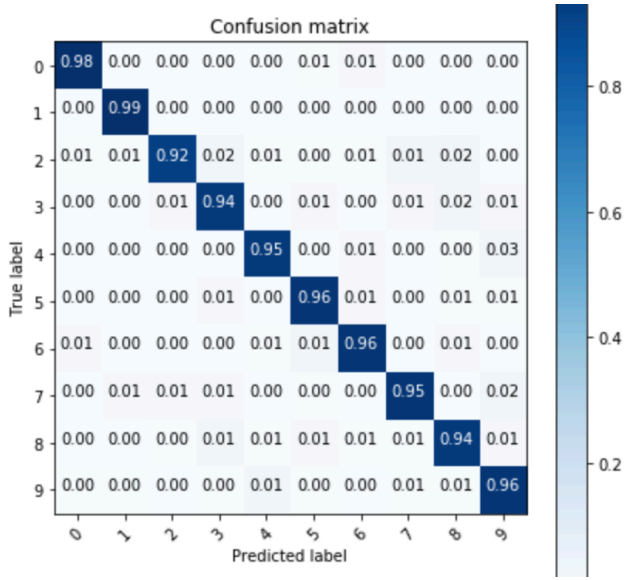
- Increase the precision of the numbers you print if necessary (it doesn't make sense to discuss 10 variants in your report for which you mention a validation accuracy of 0.98)

Number of digits after the comma

```
print("Training set Accuracy:{:7.2f}".format(train_accuracy))
print("Training set Loss:{:7.4f}\n".format(train_loss))
```

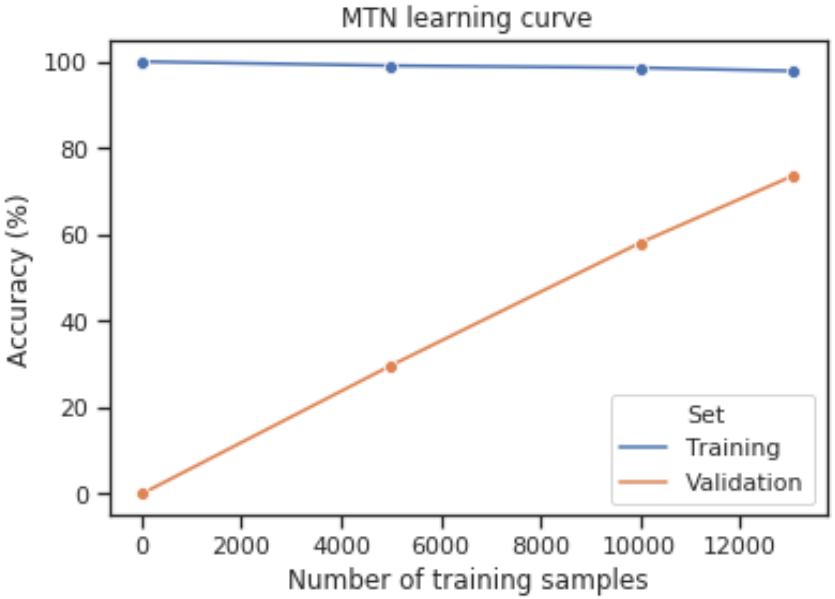
- Confusion matrices: which do you find more informative??





Learning curves??

- A lot of people have been using “learning curves” for the train- and validation curves (versus number of EPOCHS)
- This is a “real” learning curve:



- You could plot a learning curve if you re-train for (a few) different training set sizes. It could give you an indication of whether you are bumping into the ‘data limit’, where you cannot further reduce overfitting without sacrificing valisation accuracy and only more data helps