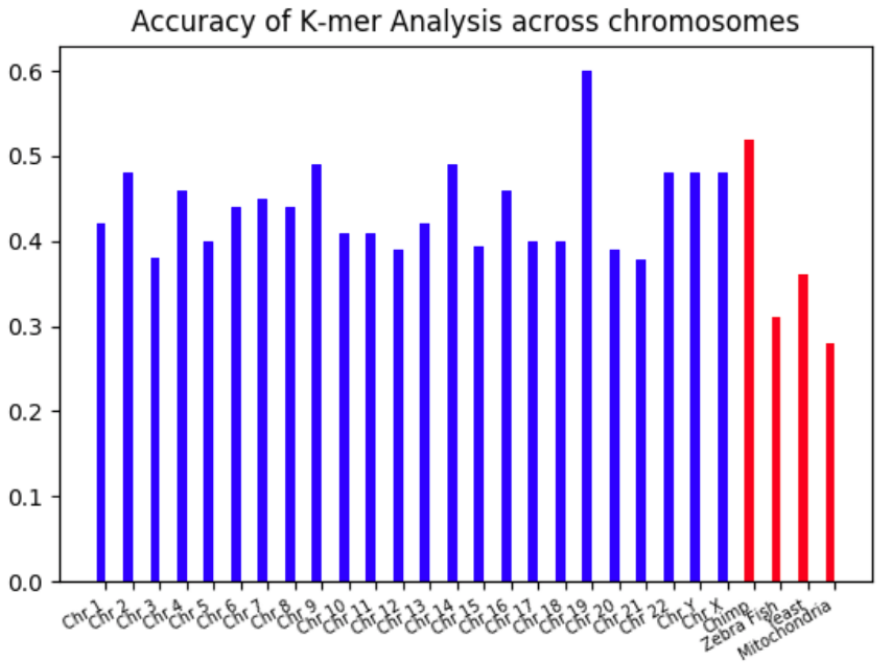
Improving context based prediction of DNA using deep learning (?)

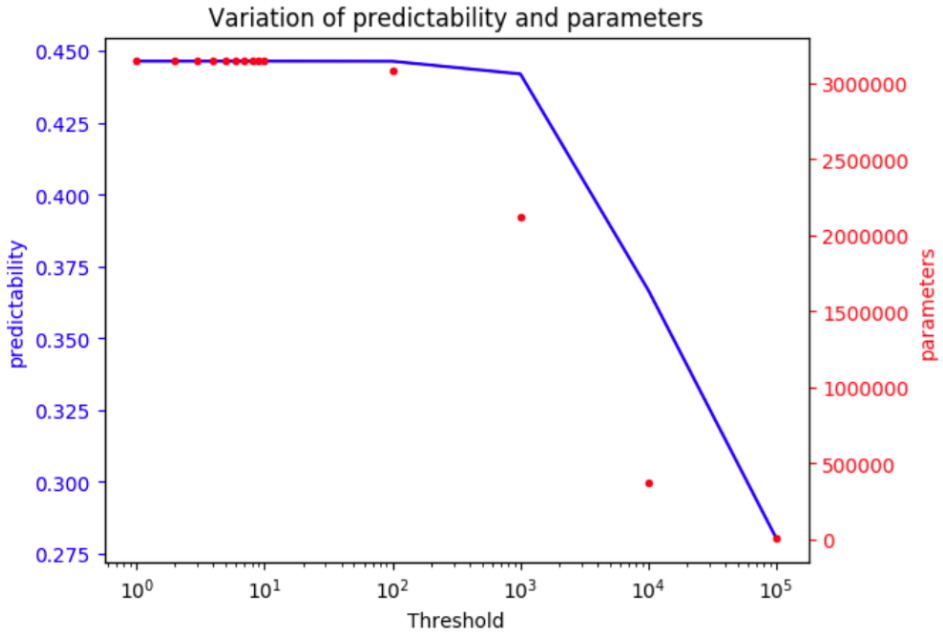
Christian Grønbæk
Section for Computational and RNA Biology,
Department of Biology

Table of Contents

k-mer model

Simply: count!





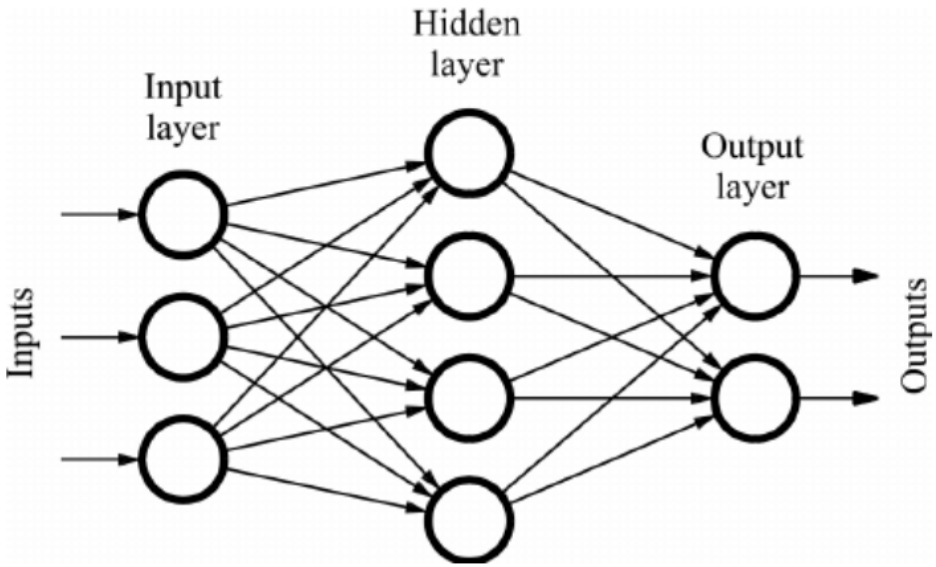
- $k = 5$; "Trained" on chr11, tested on rest

Source: Siobhan's thesis

- Number of parameters is $3 * 4^{2*5} \sim 3$ mio

Deep learning, 1st attempt: MLP

Multi Layer Perceptron ... feed forward NN



Siobhan's best:

- Context: flanks 50
- 3 layers
- 300 units each
- Performance: 46 pct (acc), 1.4 (loss)

Searches//Grid searches

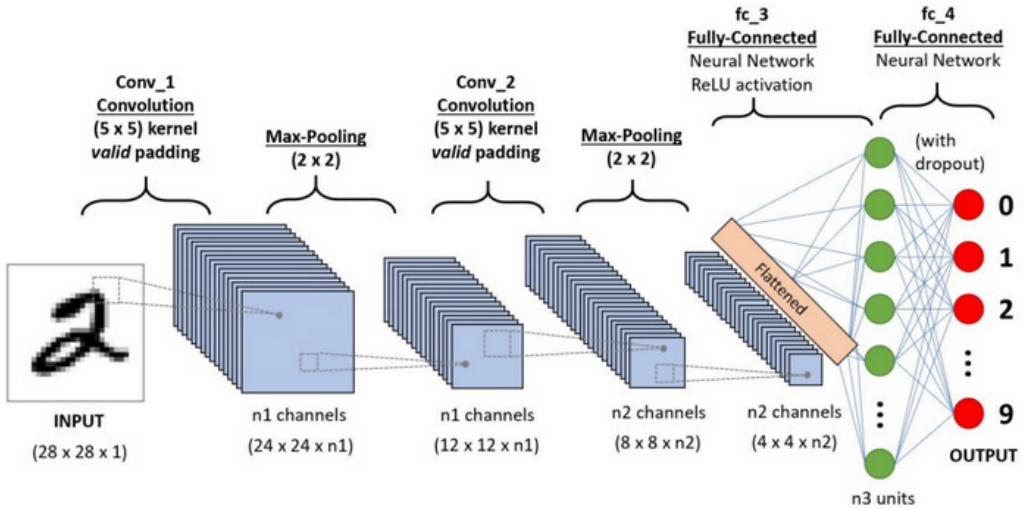
- Some done
- Flanksize: 10 - 200
- Nr of layers: 1- 4
- Nr of units: 50 - 350
- Dropout: 0.15 - 0.35
- Learning rate

What info is used in predictions?

- Shuffle part of the flanks
- Randomize part of the flanks
- Relevance propagation
- $\rightarrow$ inner flanks clearly most important
- $\rightarrow$ some sequential info in outer flanks is used

2nd attempt: Convolutional NN

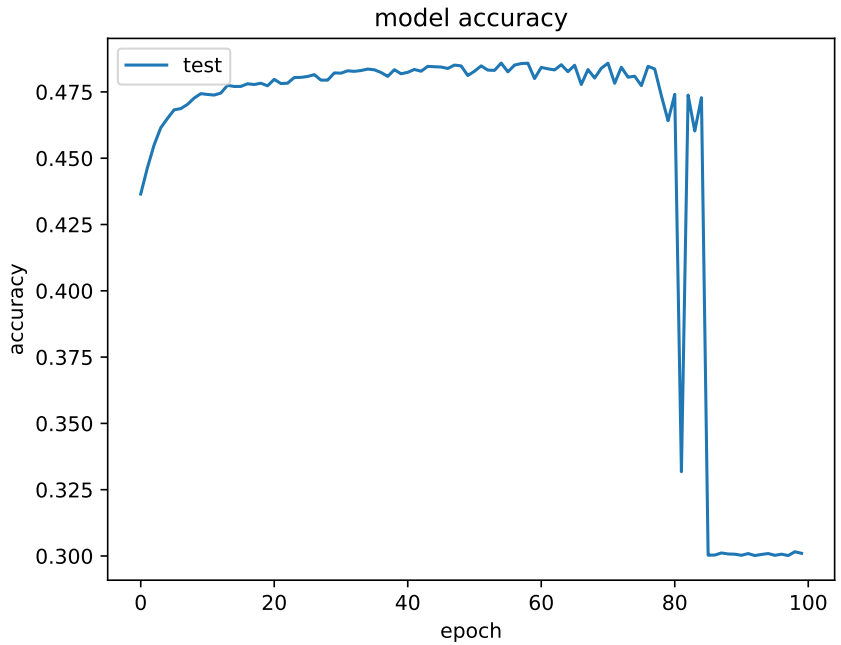
Convolutional NN



- 1-hot encode the 4 letters
- Use a ”large” set of various filters
- And in several layers
- Quite cheap in nr of parameters!

Convolutional, example

- 6 convo layers
- Filter sizes: 2, 3, 4, 6, 8, 10
- 200 of each
- Two dense layers at the end
- About 2.7 mio params
- Accuracy $\sim$ 48 pct



”epoch” equals:

- Block of 100 epochs ...
- each 100 steps of batch size 512

Searches/Grid searches

On:

- Flanksize
- Nr of layers (3 - 6), nr of final dense layers (1 - 2)
- Nr of filters (10 - 200)
- Filter sizes (2 - 10)
- Flat/increasing/decreasing nr of filters, filter sizes

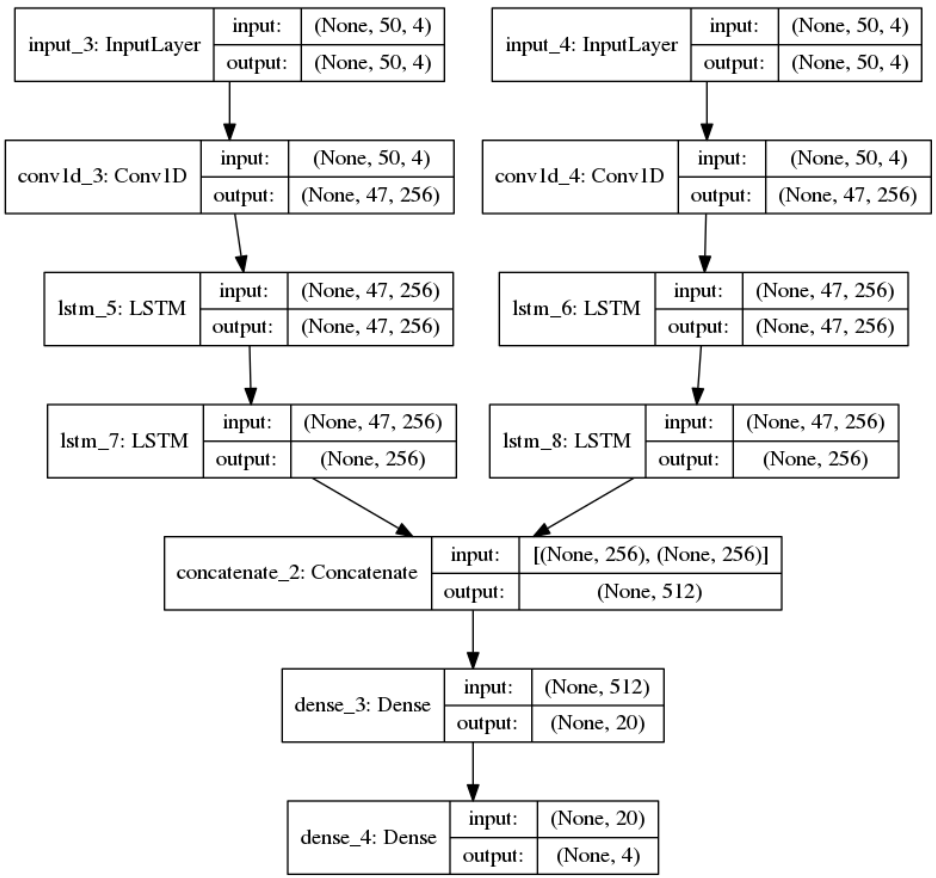
Other variations/options that we have done

- Merge the k-mer model and a convolutional ...
- Eg as input to final dense layer
- Predict base-pair rather than base
- Predict pyrimidine/purine

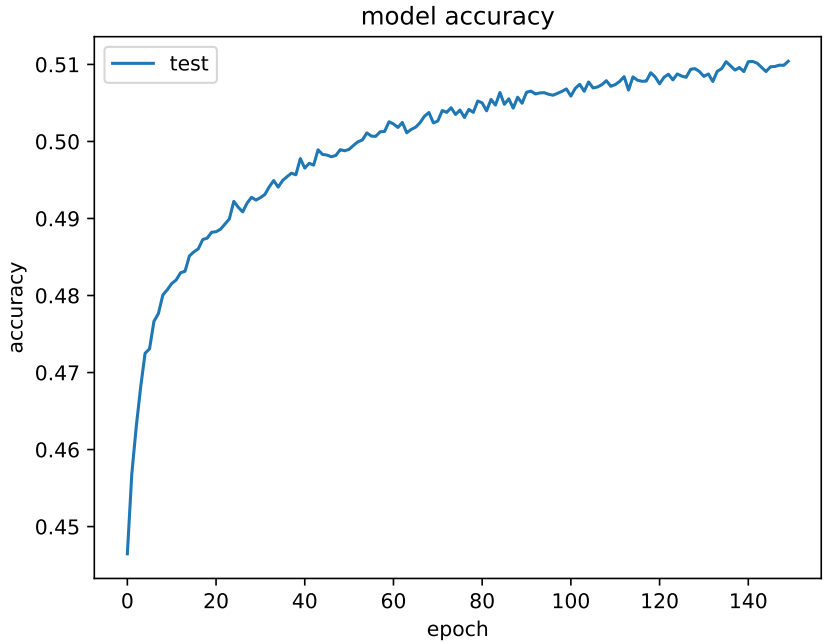
3rd: LSTM

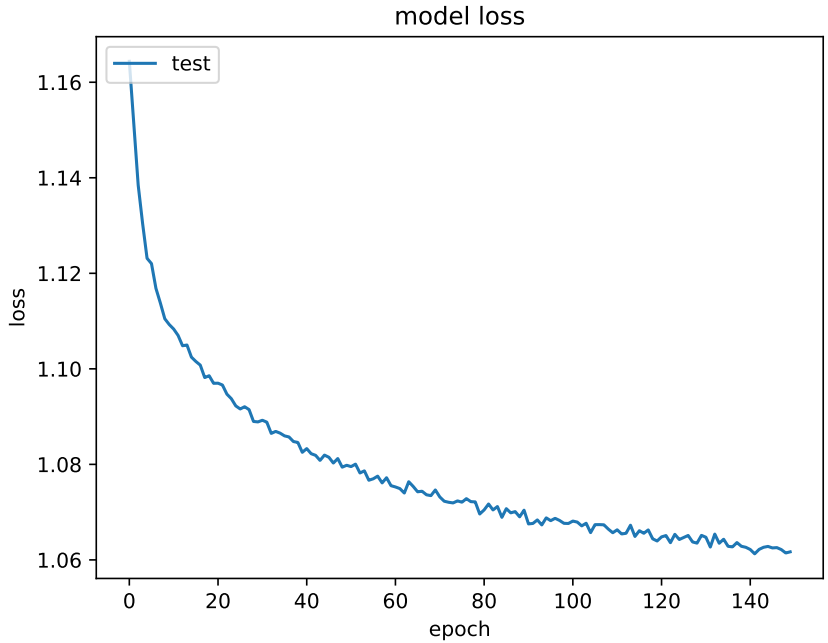
The best so far ...

Bi-directional (Convo + LSTM) plus a final dense layer (for rep'n!):



... and its performance



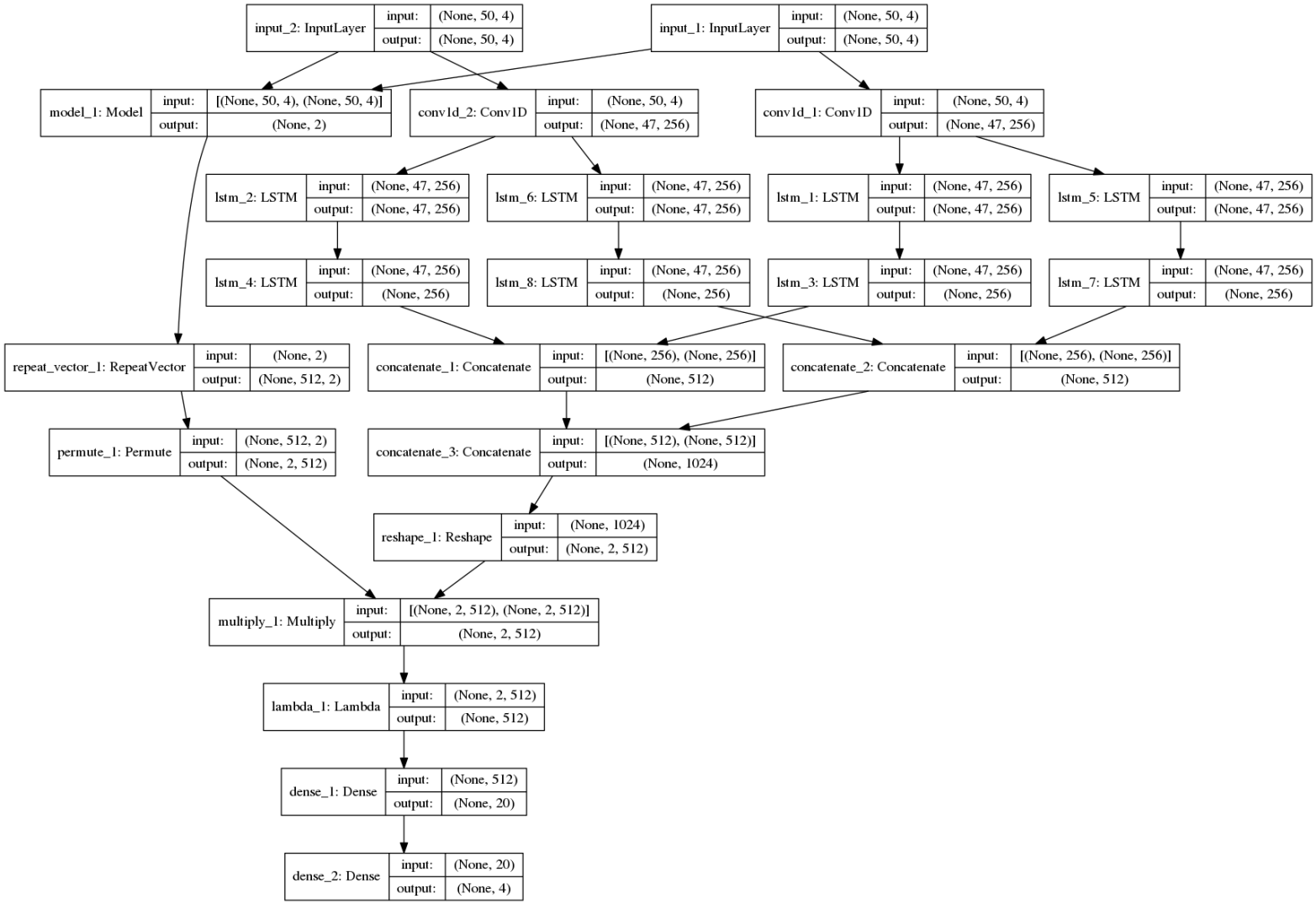


- Took 2-3 weeks to train (on our best GPU, but not idle)
- Has about 2.3 mio parameters
- Obs: last 1pct takes about 100 ”epochs” out of 150!
- Non-/repeat accuracy is roughly split 40 pct/60 pct (near 63 pct really)

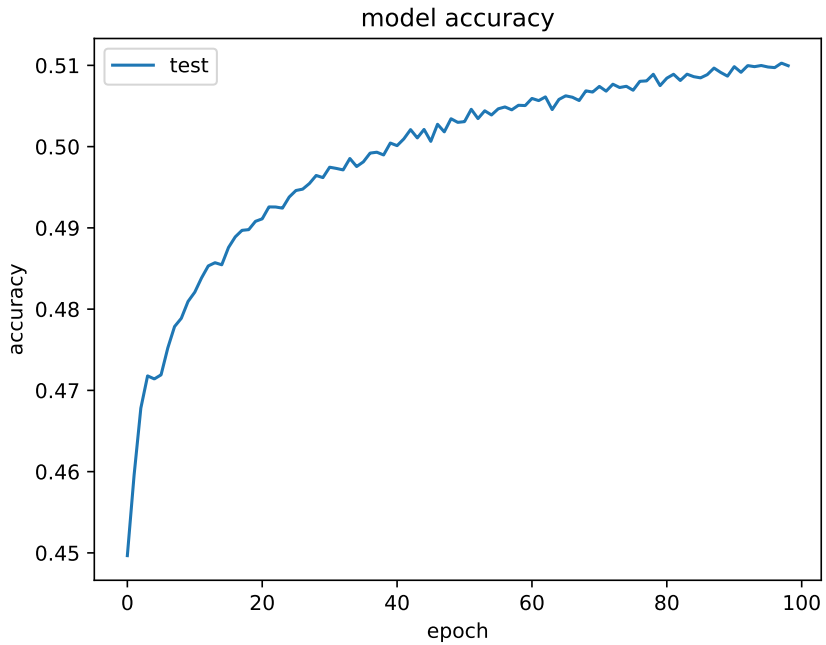
Variations for improvement

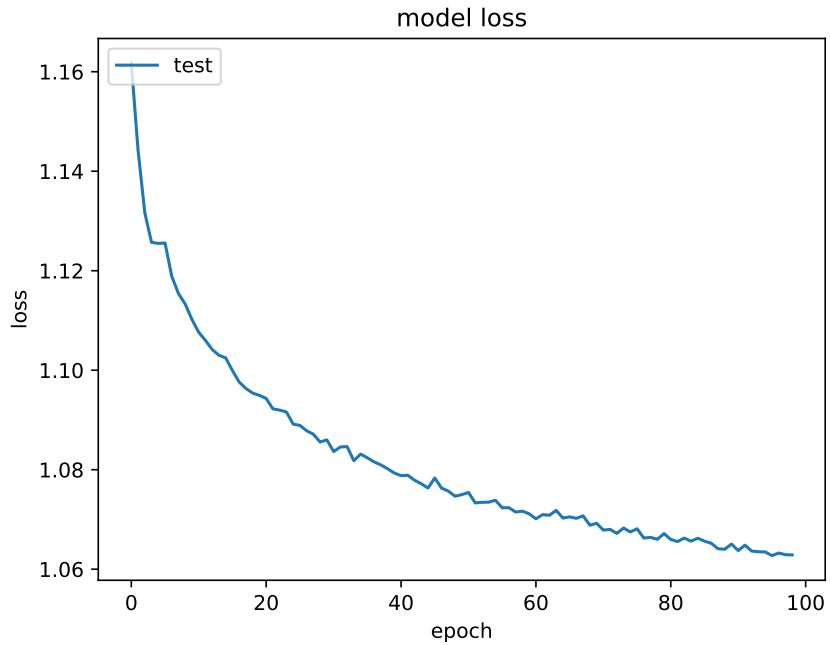
- Improve acc on repeat part?
- Use larger flanks? (more later)
- Other: train one model to predict non-/repeat
- Merge two of the models above ...
- by weighting with the non-/repeat model ... like this:

Two parallel LSTMs merged with repeat model



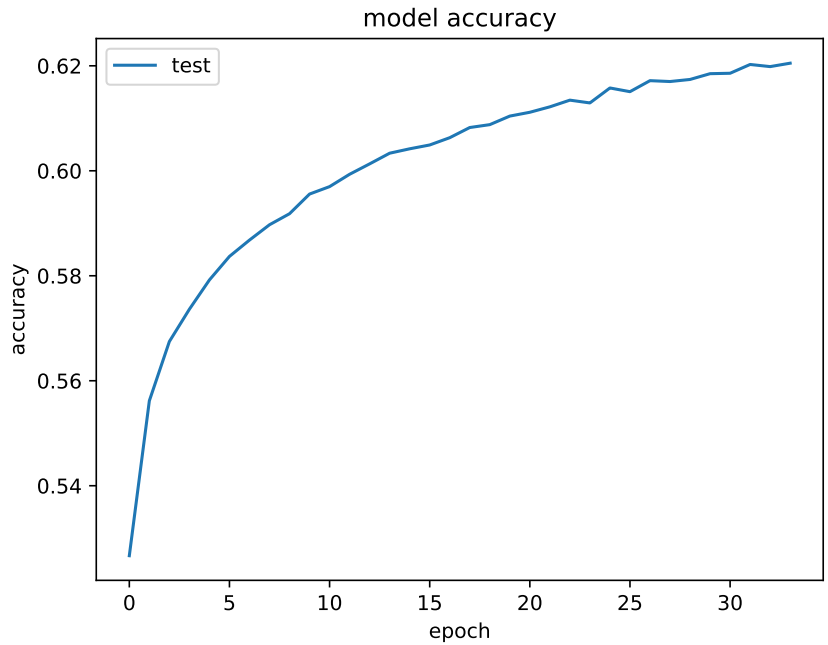
... and its performance





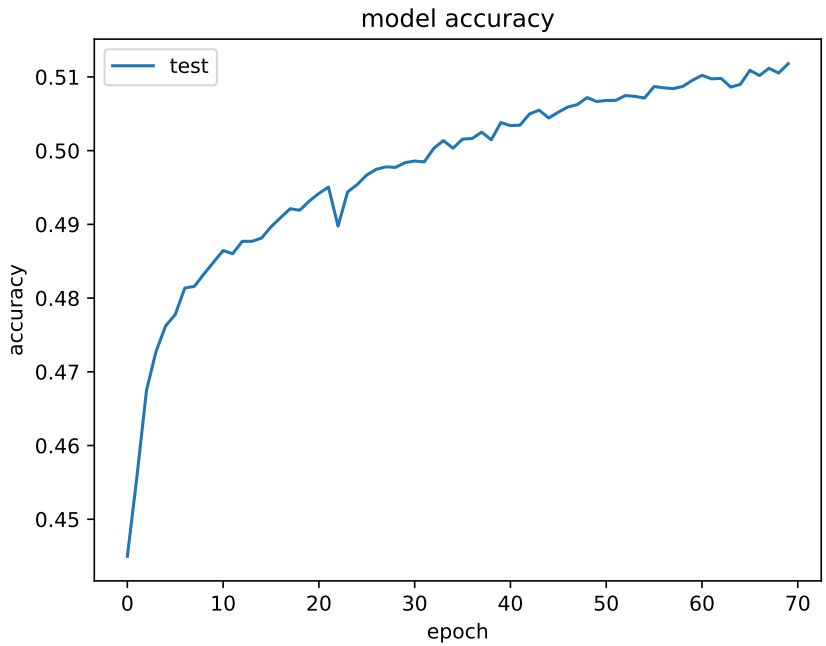
But: best model trained on repeats only ...

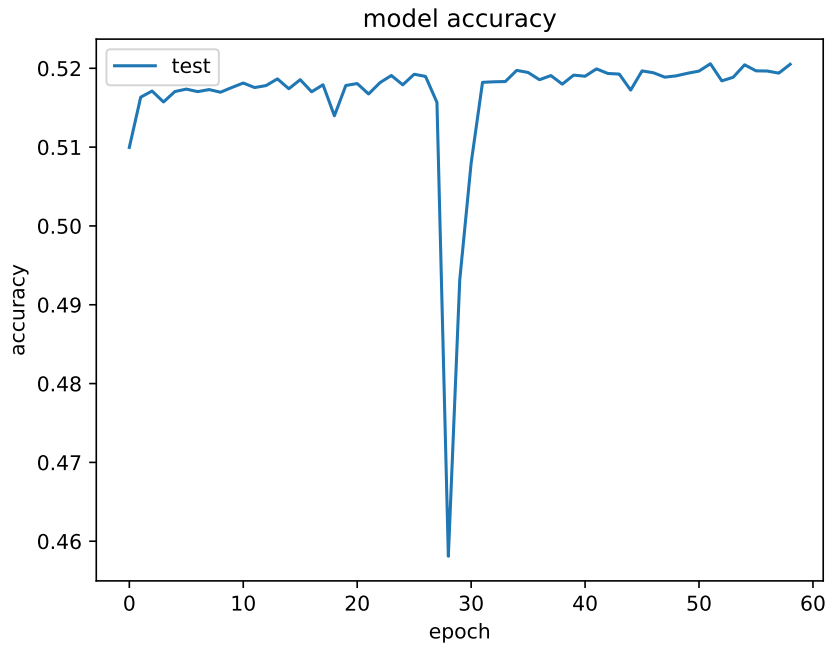
- Model (almost) id to ”best”
- When trained only on repeats ..
- acc approaches the best’s



- Apparently it’s not worthwhile to make a ”fusion-model”
- Obs: the fusion model has about 7.5 mio param’s

Best model but on larger flanks (200)





- So: my new personal best is $\sim$ 52 pct!
- Obs: the number of param's is still $\sim$ 2.3 mio

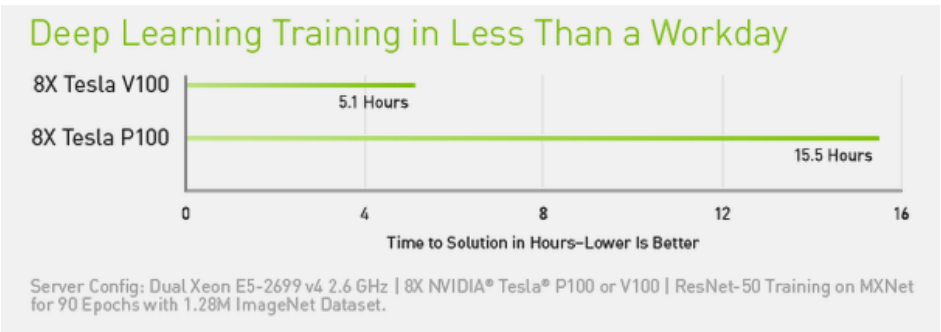
4th: Transformer

Keyword: ”Attention”

- State-of-the-art in language modelling tasks
- Idea is: pay particular attention to certain words (and not all, far from!)
- May not be what we need here, but a must to try it out!
- Have the set-up running now
- Need to get acquainted with it ...
- Training/validation looks strange so far (looks as if overfitting but should not be able to)

GPU-issues!!

- Recently Facebook group ran deep-transformer on 250 mio amino acid seq's
- Took $\sim$ 4 days to train ...
- on 128 Nvidia V100 GPU's!!
- Each is $\sim$ 3 x as fast as our's
- Ie 1day for FB $\sim$1 year for us
- Would cost about 30000 $'s to buy on Google Cloud!



Final words

- Yes, it is possible to improve the k-mer model using (deep) NNs
- Yes, the 50 pct mark can be passed ...
- and without using more param's than the k-mer model
- Yes, there's a hardware challenge ...
- but we can get first signs of good performance
- One aim is to get an ”economic” representation
- And (ab)use it for various biological insights