Connectionist Computing
COMP 30230/41390

Gianluca Pollastri

office: E0.95, Science East.

email: gianluca.pollastri@ucd.ie

Credits

- Geoffrey Hinton, University of Toronto.
  – borrowed some of his slides for "Neural Networks" and "Computation in Neural Networks" courses.

- Ronan Reilly, NUI Maynooth.
  – slides from his CS4018.
- Paolo Frasconi, University of Florence.
  – slides from tutorial on Machine Learning for structured domains.

Lecture notes on Brightspace

- Strictly confidential...
- Slim PDF version will be uploaded later, typically the same day as the lecture.
- If there is demand, I can upload onto Brightspace last year's narrated slides.. (should be very similar to this year's material)

Books

- No book covers large fractions of this course.

- Parts of chapters 4, 6, (7), 13 of Tom Mitchell's "Machine Learning"
- Parts of chapter V of Mackay's "Information Theory, Inference, and Learning Algorithms", available online at:

http://www.inference.phy.cam.ac.uk/mackay/itprnn/book.html

- Chapter 20 of Russell and Norvig's "Artificial Intelligence: A Modern Approach", also available at:

http://aima.cs.berkeley.edu/newchap20.pdf

- More materials later..

Marking

- 3 landmark papers to read, and submit a 10-line summary on Brightspace about: each worth <u>6-7%</u>
- a connectionist model to build and play with on some sets, write a report: <u>30%</u>
- Final Exam in the RDS (<u>50%</u>)

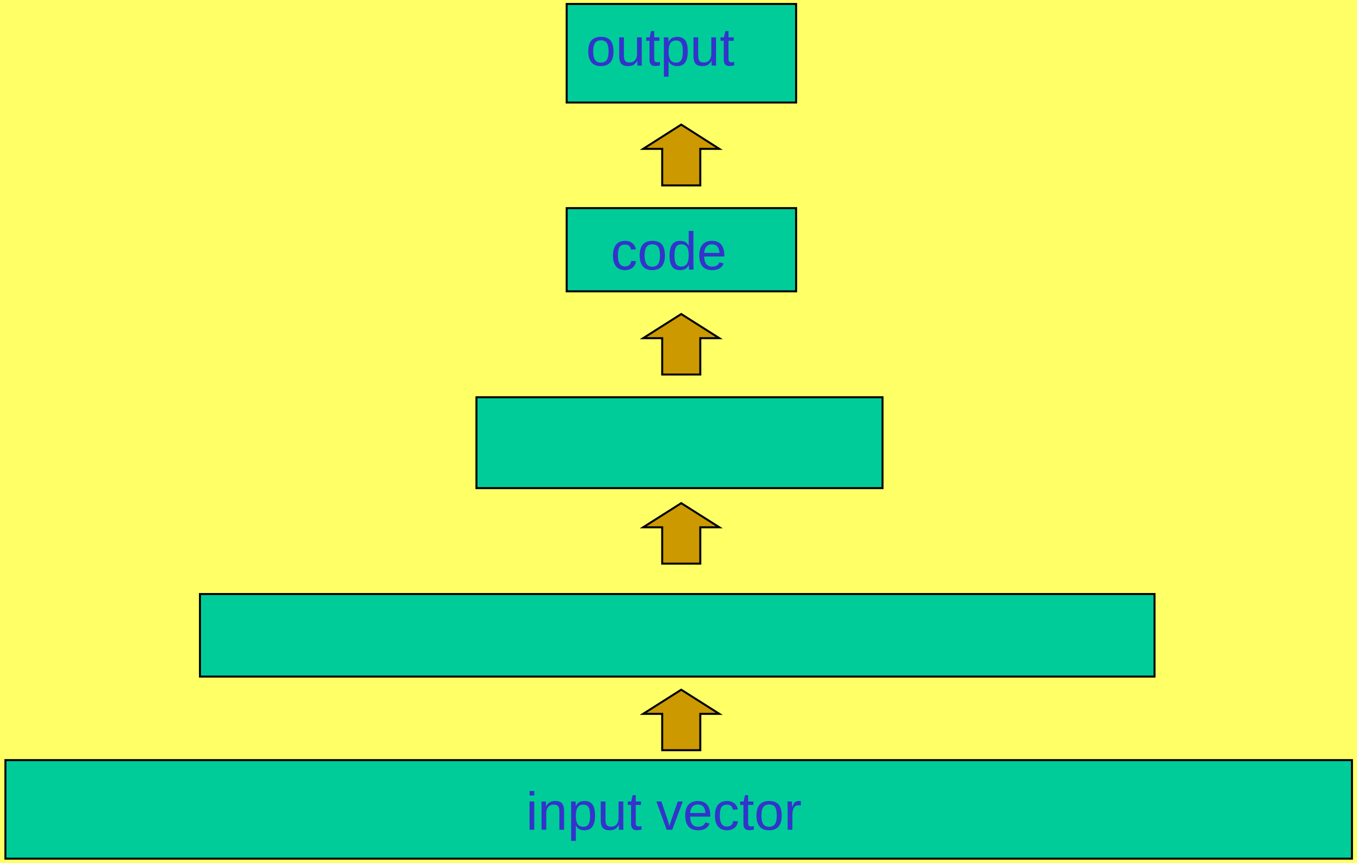
Programming assignment

- Implement a Multi-Layer Perceptron, test it.

- The description on Brightspace.

- Submit through Brightspace code and test results by Dec the 5th at 23:59, any time zone of your choice (Baker Island?).
- 30% of the overall mark
- One third of a grade down every day late, that is: if you deserve an A and you're 1 day late you get an A-, 2 days late a B+, etc.

Deep networks?



Deep is hard

- There are many reasons why deep should work better than shallow.
- But anyone who has trained a neural network with many layers knows the more inner layers you add, the longer the initial plateau lasts.
- By 5 hidden layers, typically, it is long enough that you can't afford to wait..

Deep is hard because gradients vanish

- **The problem is that when networks get deep the gradient *vanishes*.**
- **When a network is untrained, the deeper down a hidden unit is, the more subtle its effect is on the outputs.**
- **If it's subtle, it means it doesn't do much to the error if you change it.**

Deep learning solutions

- These are more principled solutions that were proposed starting circa 2006.
- Essentially they fall into two main categories:
- pre-train
- use artificial inner targets

The rationale of pre-training

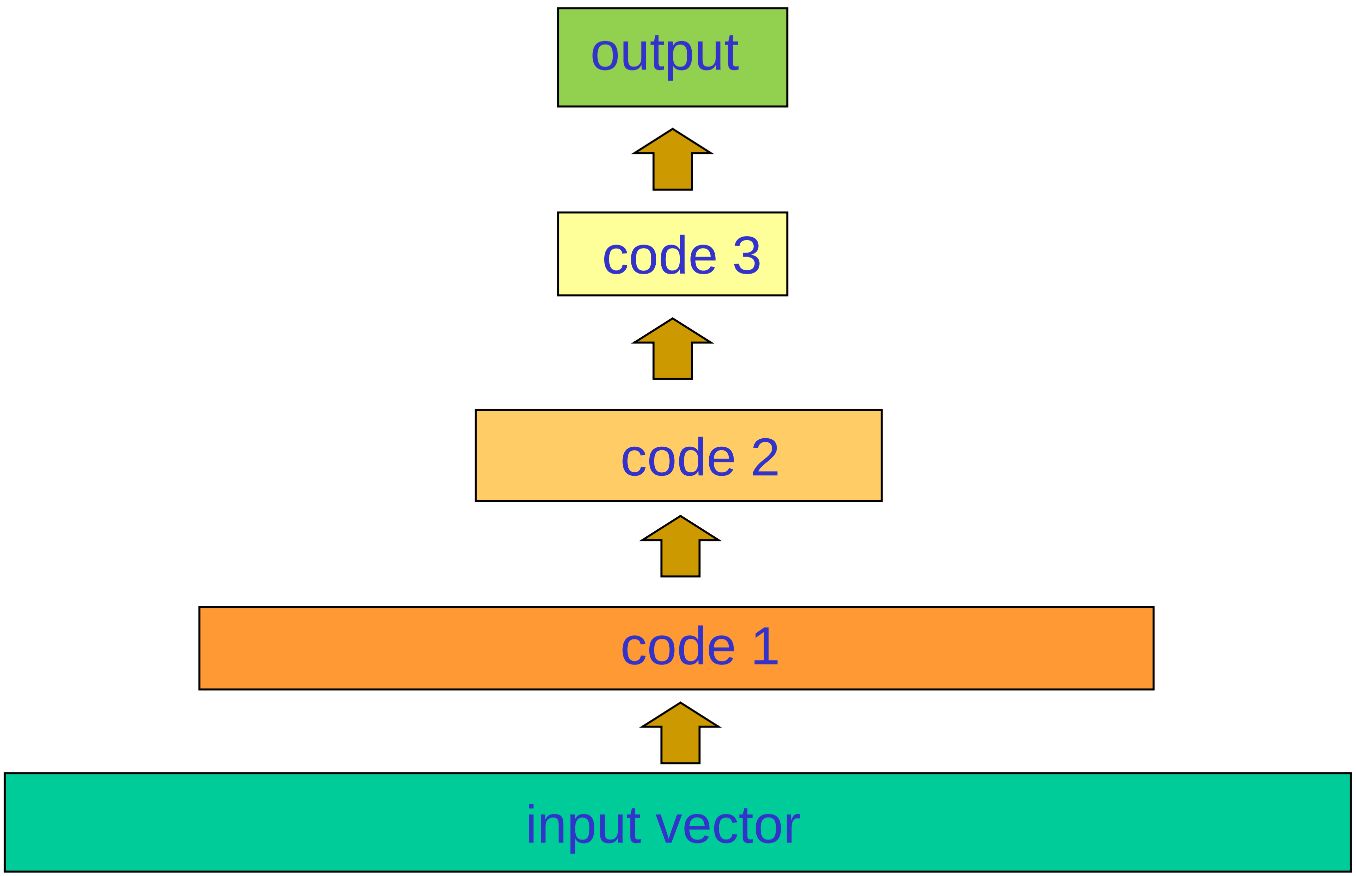
- Stack deep networks layer by layer.
- Make sure your first hidden layer represents the input meaningfully before you add a second layer.
- Then make sure your second layer represents the first layer (thus, the input) meaningfully before you add a third layer.
- And so on…

- This can be done by auto-association

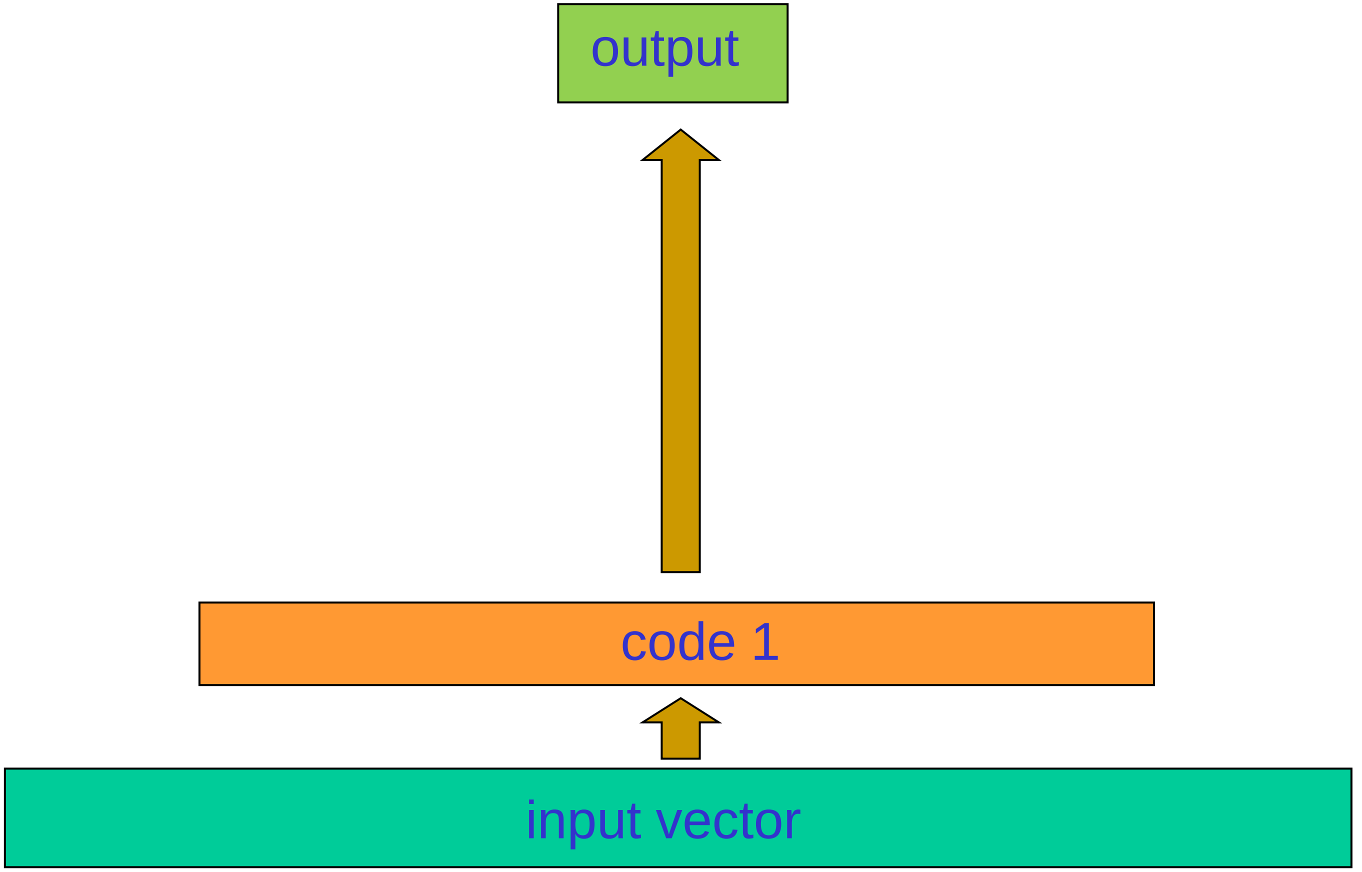
About pre-training by auto-association

- (Massive) advantage:
  - you can use unlabelled data!
- (Potentially disastrous) disadvantage:
  - you aren't considering at all the property you want to predict
  - you compress *regardless* of the property. If it's lossy, the loss can be in the wrong place..

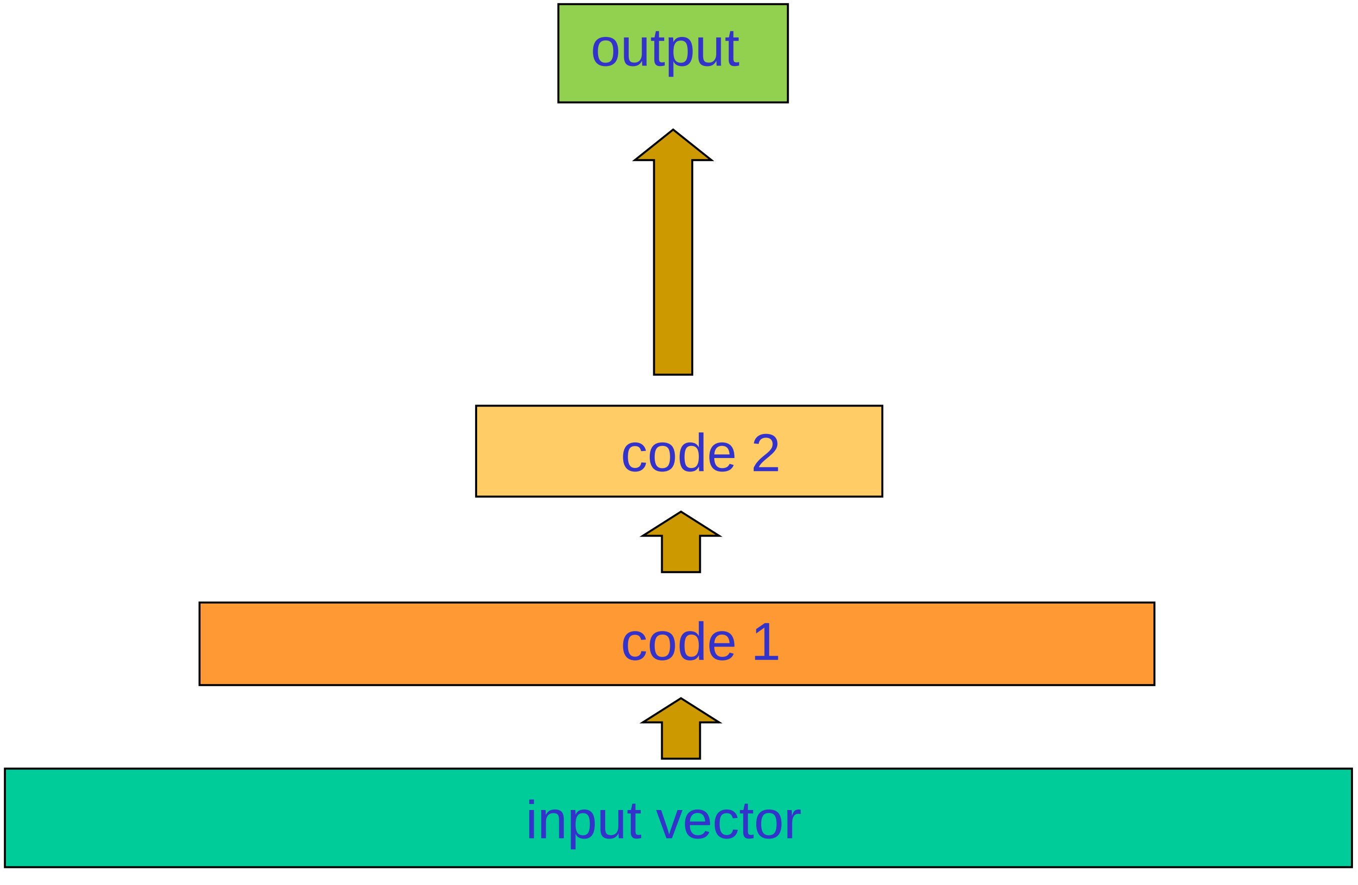
Pre-training without auto-association



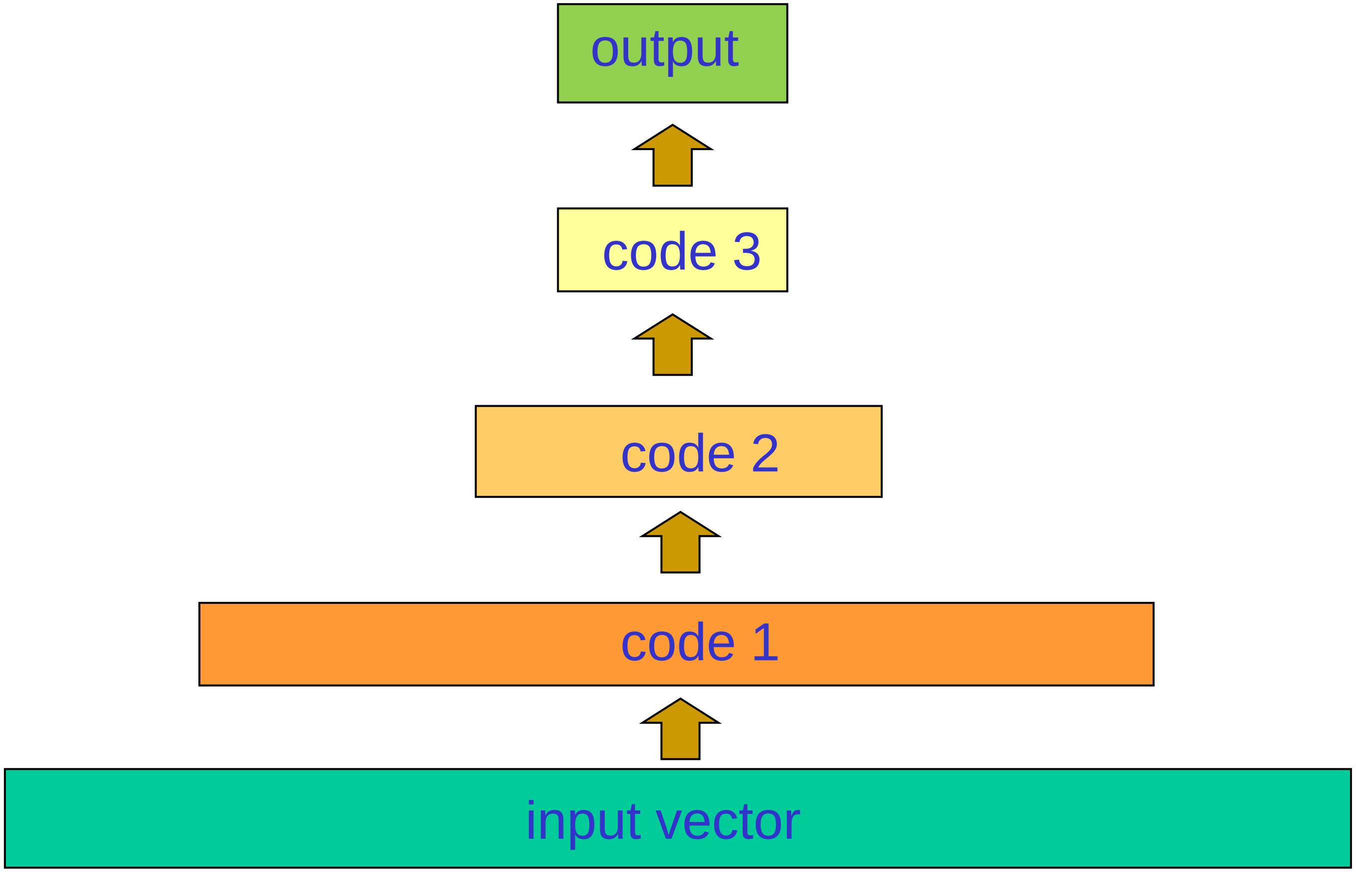
Pre-training without auto-association (2)



Pre-training without auto-association (3)



Pre-training without auto-association (4)



About pre-training *without* auto-association

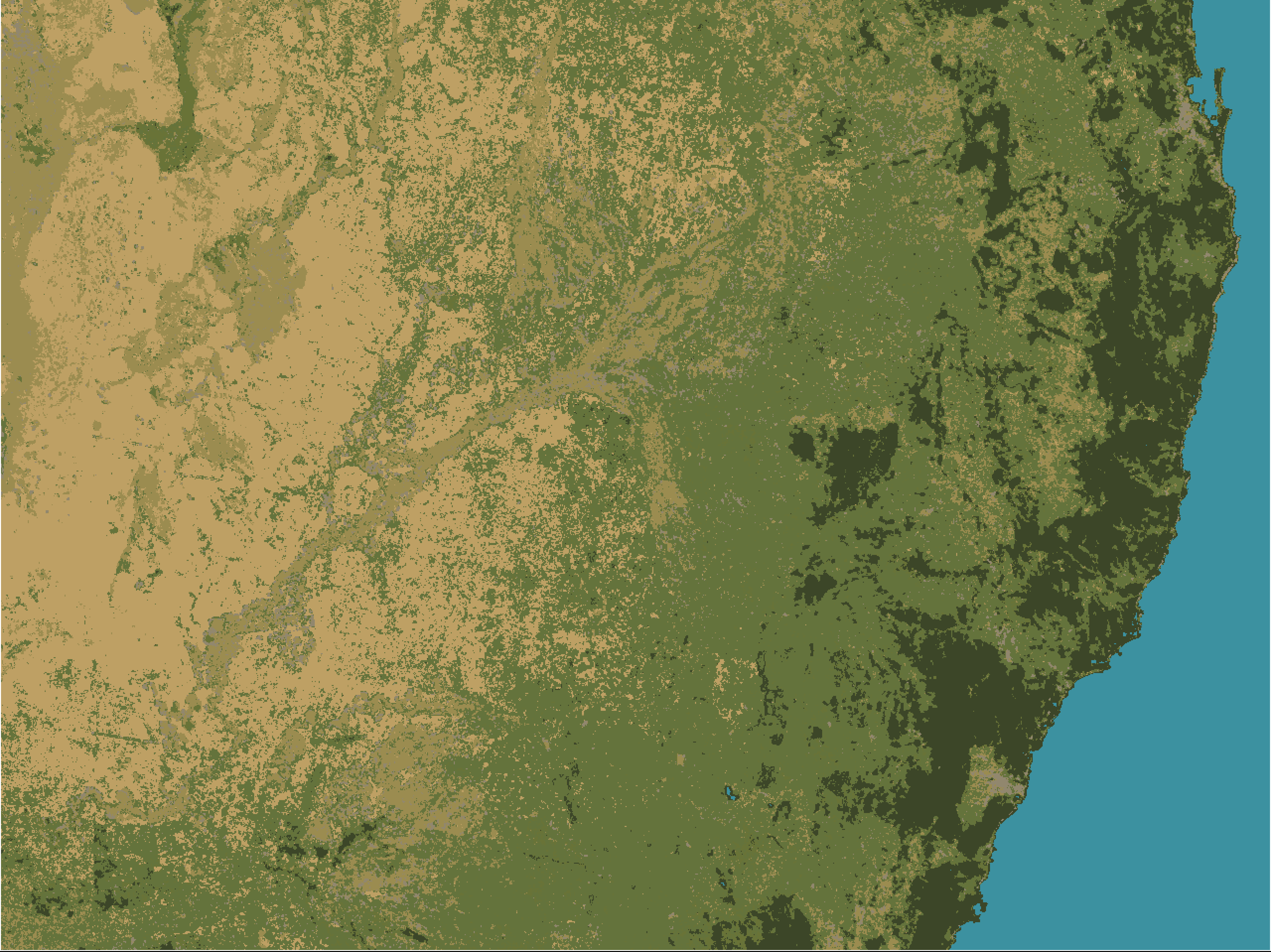
- (Possibly big) disadvantage:
  - you <u>can't</u> use unlabelled data! You need a target at all stages.
  - that said, less data = shorter training
- (Possibly big) advantage:
  - you compress *based* on the property you are trying to predict. If it's lossy, the loss is probably in the right place..

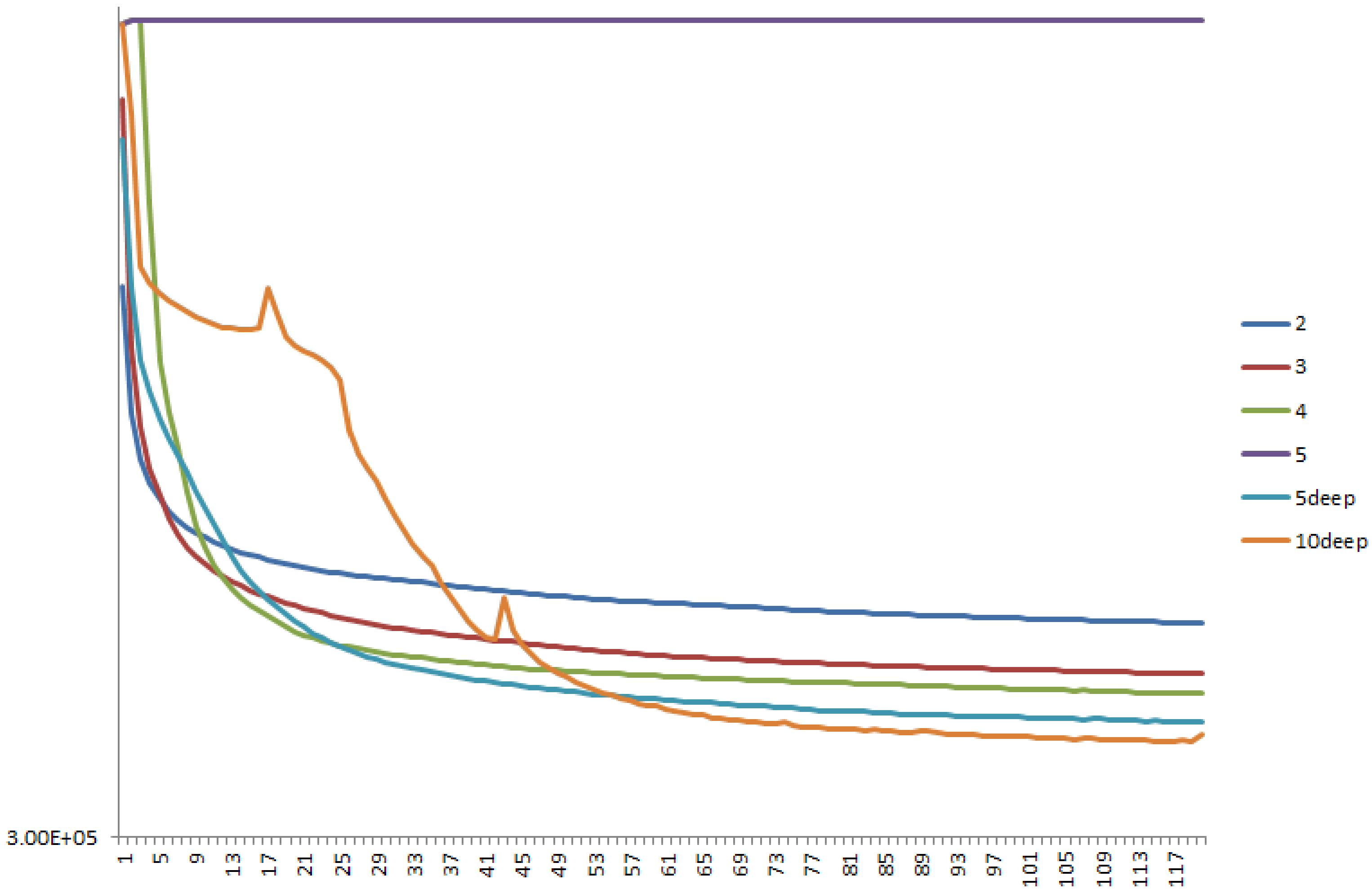
Deep learning: artificial targets

- **The real problem is that inner layers don't get gradient.**
- ***Every now and then*** **use** **<u>hard targets</u>:**
  - **generate a handful of random targets for the layer**
  - **check them all (how do they fare on this example?)**
  - **Use the best one..**

Additional solutions

- Different squashing units (e.g. rectified linear units)
- Initialisation
- Normalisation techniques for inputs to inner layers
- …




3.00E+05
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
101
105
109
113
117
2
3
4
5
5deep
10deep

Protein bioinformatics

- The input is a string (the primary sequence)
- The target is whatever value added property of the protein we are interested in. E.g. protein function, secondary structure, solvent accessibility.

Protein bioinformatics

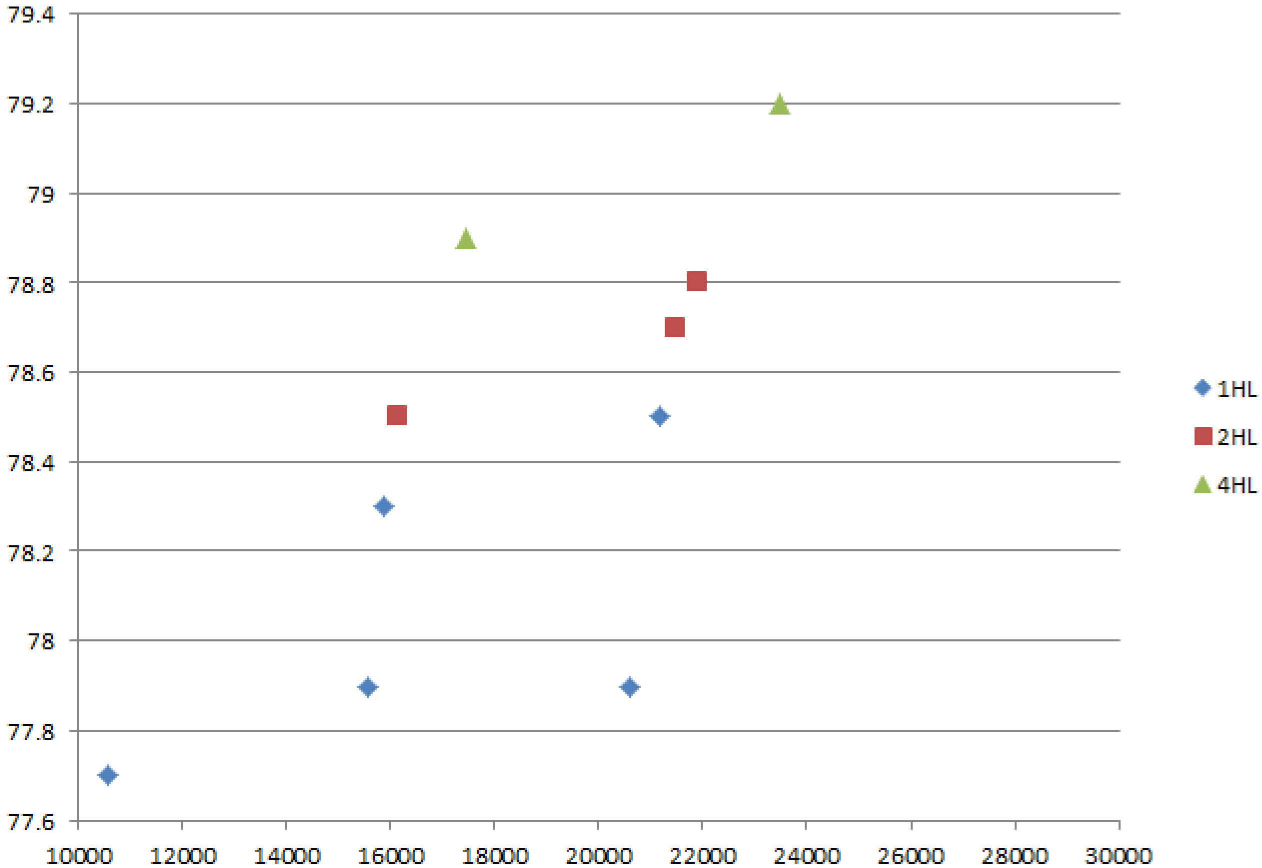
- **The input is a string (the primary sequence)** 0.10$
- **The target is whatever value added property of the protein we are interested in. E.g. protein function, secondary structure, solvent accessibility.**

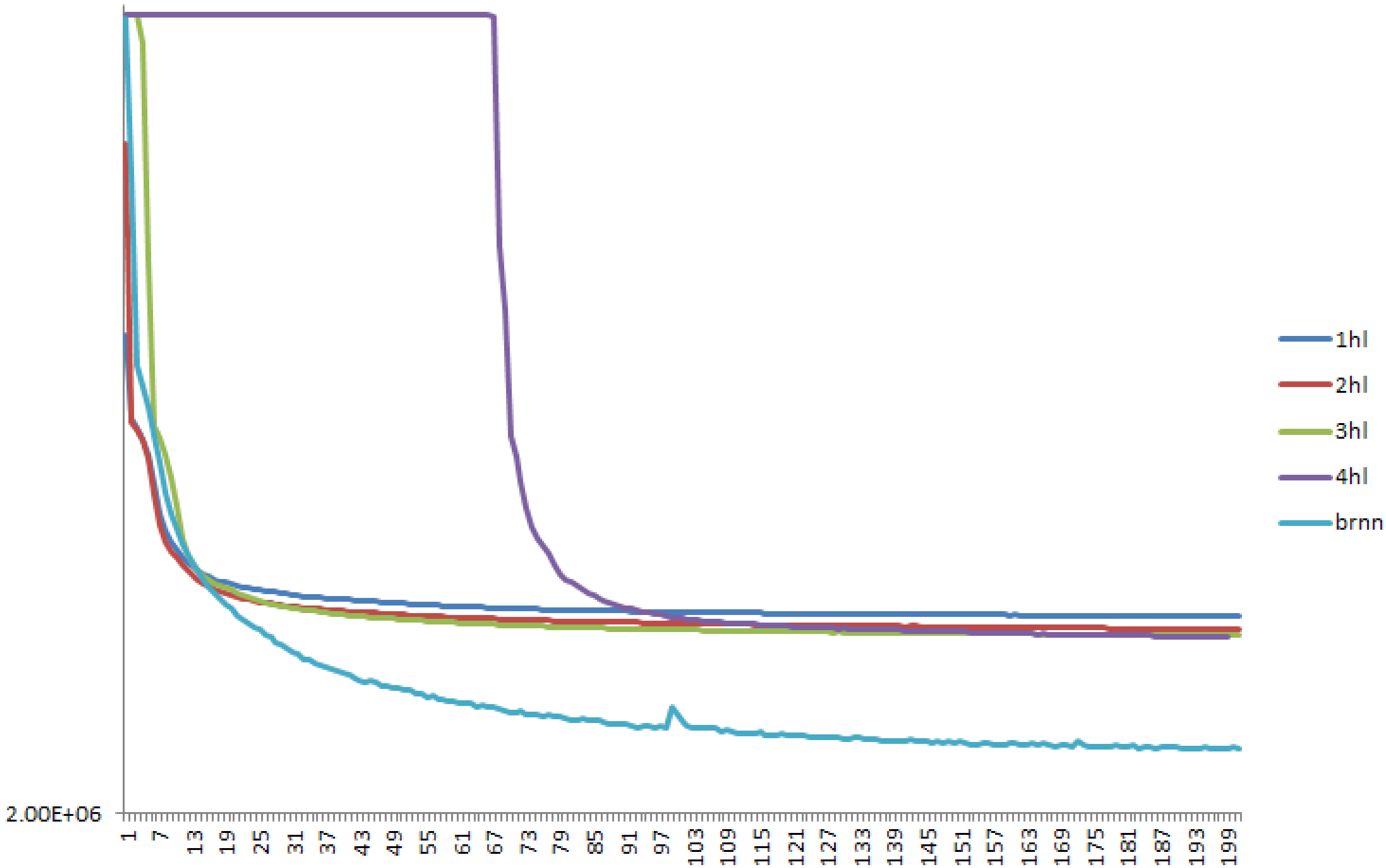
100,000$?

Data

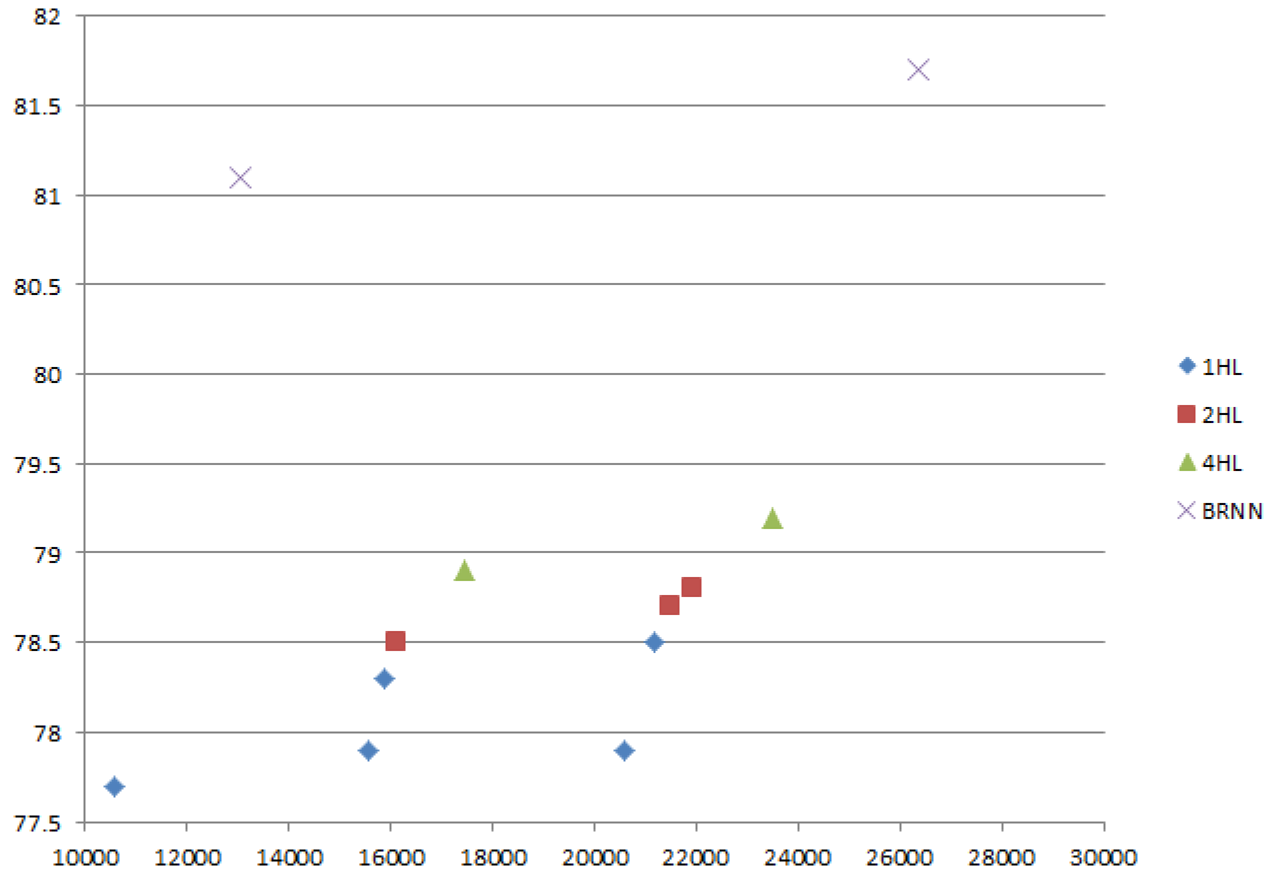
- I dislike small sets. A 1000 point set isn't going to give you the same answer as a 1M point set. The winners will change.

- I built a set of 23,000 proteins (6M examples)


79.4
79.2
79
78.8
78.6
78.4
78.2
78
77.8
77.6
10000
12000
14000
16000
18000
20000
22000
24000
26000
28000
30000
1HL
2HL
4HL



2.00E+06
1
7
13
19
25
31
37
43
49
55
61
67
73
79
85
91
97
103
109
115
121
127
133
139
145
151
157
163
169
175
181
187
193
199
1hl
2hl
3hl
4hl
brnn



82
81.5
81
80.5
80
79.5
79
78.5
78
77.5
10000
12000
14000
16000
18000
20000
22000
24000
26000
28000
30000
1HL
2HL
4HL
BRNN

Data, data, data

- **The more the better..**

set size	2179	3129	4818	7522
date	12/2003	1/2007	11/2009	6/2012
Porter (SS)	79.1%	80.5%	81.8%	82.2%
PaleAle (RSA)	-	54.4% (79.1%)	54.9% (79.5%)	55.3% (80%)

Mirabello & Pollastri 2013, *Bioinformatics*

What did I learn about deep learning?

- Layer-by-layer pre-training
- Artificial targets for inner layers

- They all worked the same for me.
- Deeper nets work (slightly) better than shallow ones. The big gain is between 1 and 2 hidden layers, then it tapers

- BUT: some cleverer deep wirings (BRNN) can yield major improvements.

It depends on the problem!

- Deep learning has produced some pretty stunning results in some fields (e.g. computer vision).
- In other fields, going from shallow to less shallow usually helps, but there is no need (or scope) for true deep learning.

Algorithms plus data plus CPU (or GPU) plus ease of use

- Many algorithms used in deep learning have been around for a while. At most they have been combined and shuffled cleverly.
- The big changes are:
  - immense amounts of data
  - faster computers and, especially, the ability to run training algorithms on graphics cards 1+ orders of magnitude faster, $ for $
  - A number of environments/libraries that have made formerly highly complicated implementations accessible
  - A LOT of buzz..

Next..

- A number of deep architectures I have worked with.
- Interestingly, most of them need only relatively lightweight (or no) deep learning techniques to be trained.