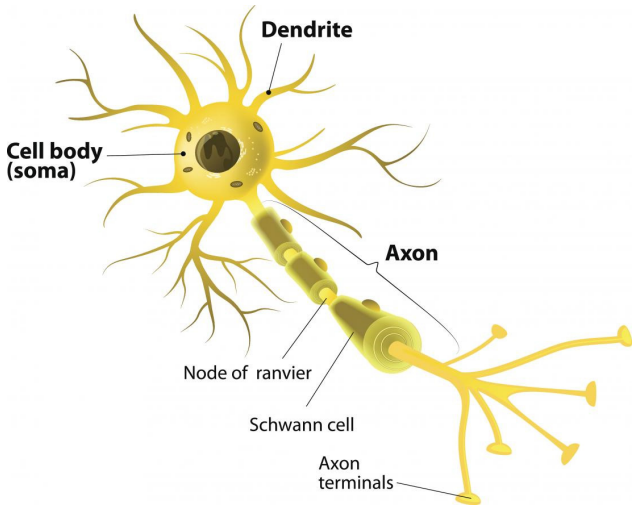
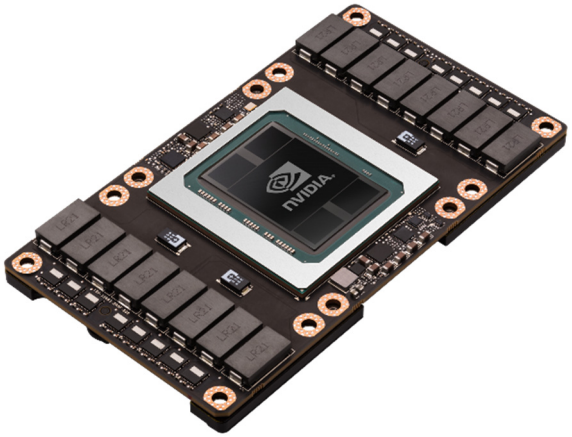




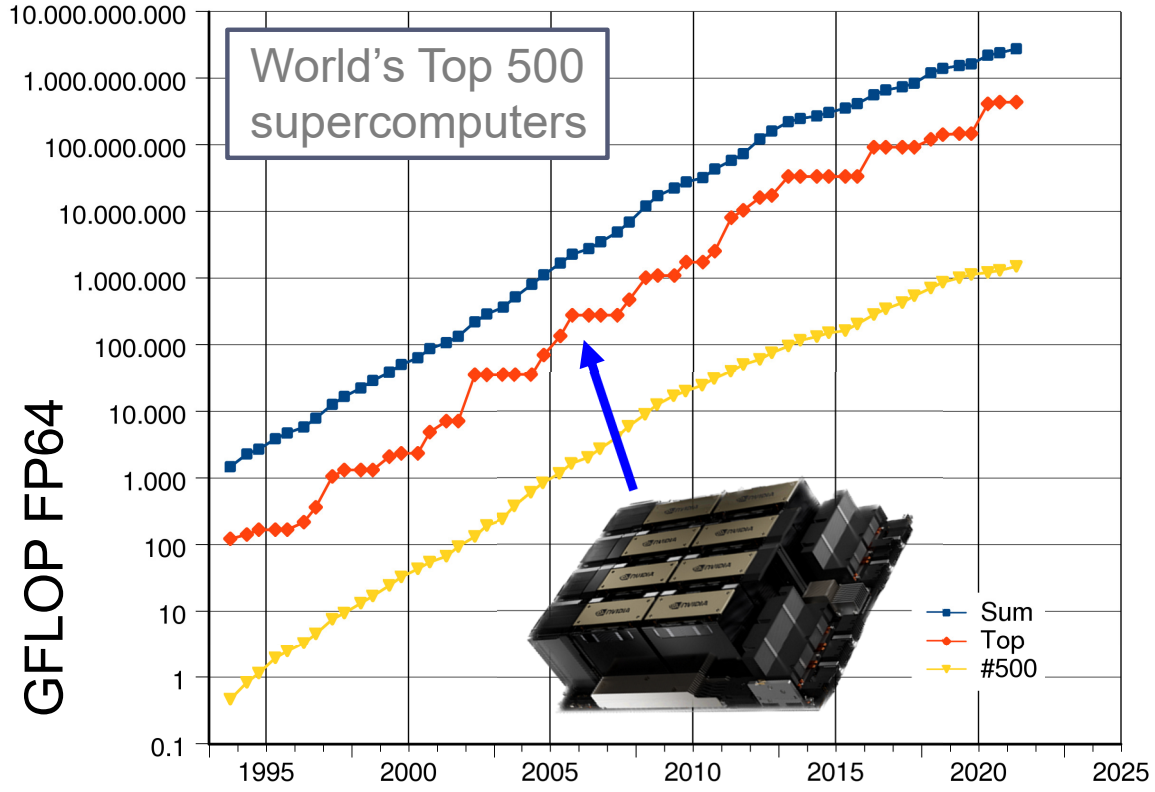
Machine Learning:
the algebra behind the hype

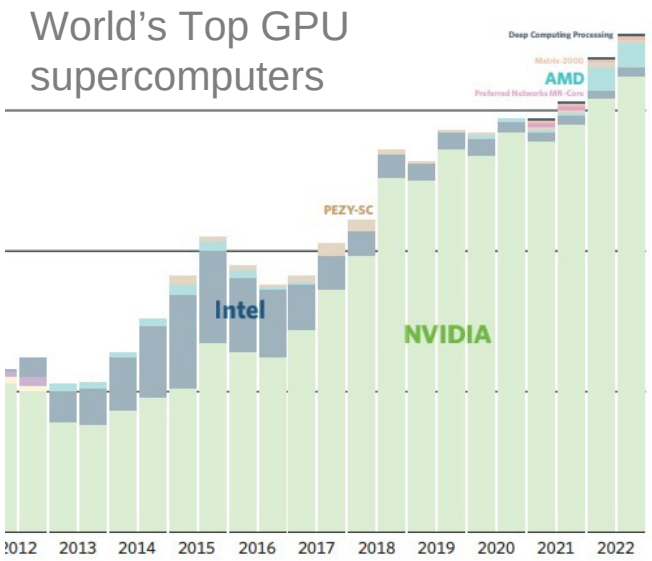
a jargon-busting tutorial on common machine learning building blocks and a few examples

Prof Ilya Kuprov, Weizmann Institute of Science, 2025

For all lecture notes and video records see https://spindynamics.org

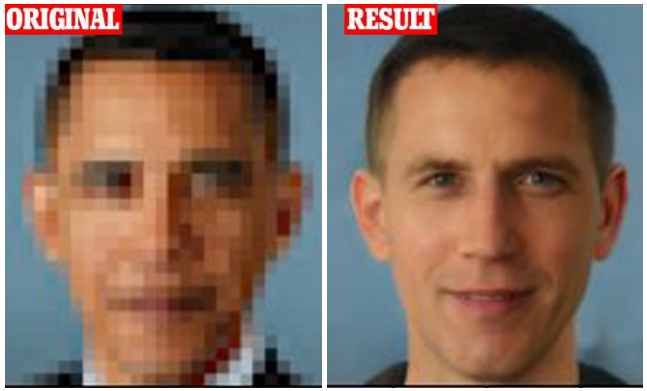
Nvidia GPU revolution (2010 - ...)

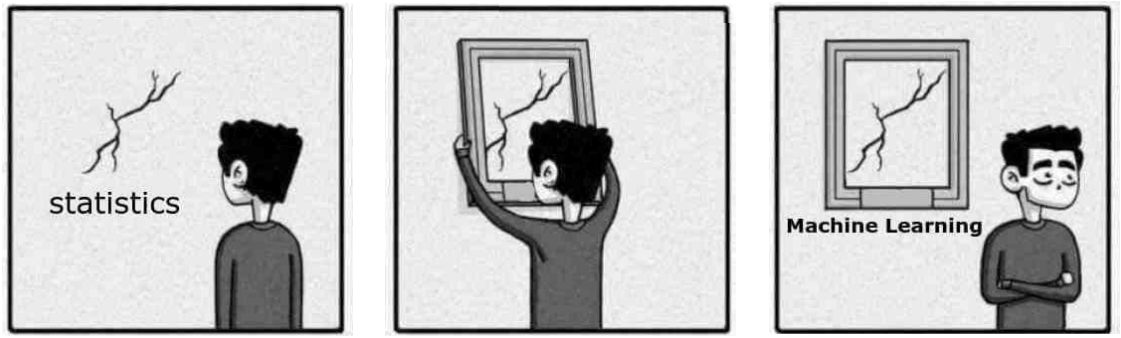




MRI people: *here's over 9000 papers on AI resolution enhancement!*

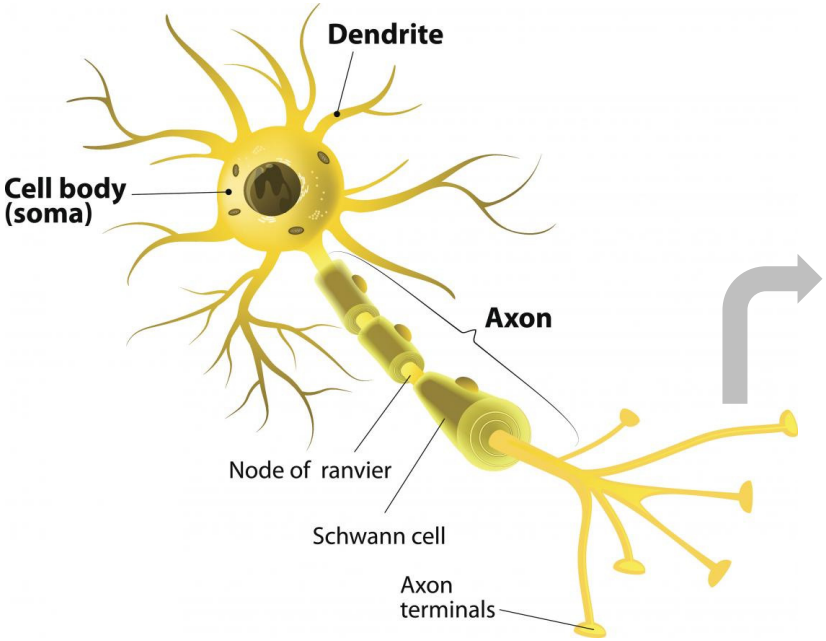
AI resolution enhancement:

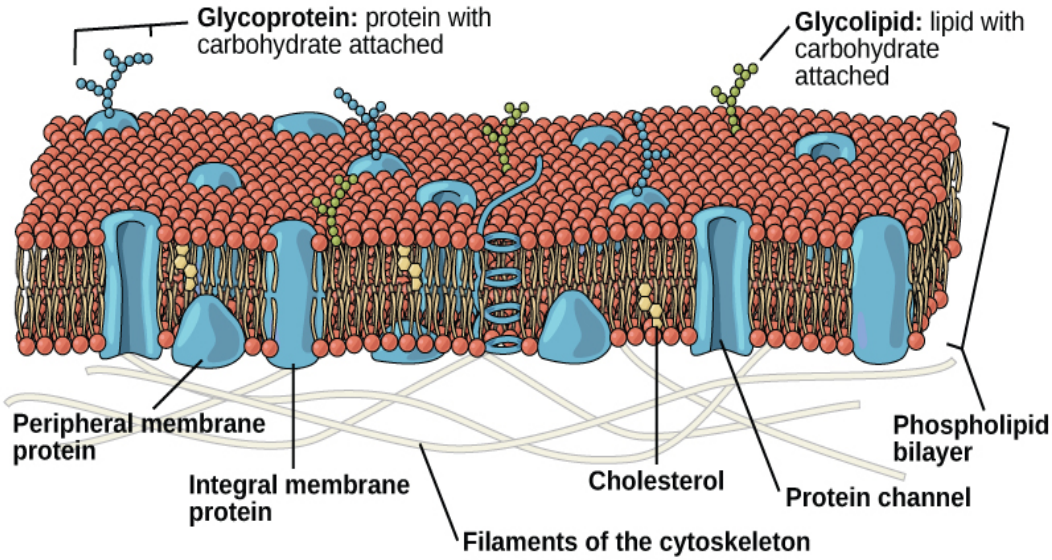




- 250 TFLOP × 1 month = 1 millimole of multiplications!

Inspiration: biological neuron





ION	CONCENTRATION (mM) INTRACELLULAR	EXTRACELLULAR
Squid neuron		
Potassium (K^+)	400	20
Sodium (Na^+)	50	440
Chloride (Cl^-)	40–150	560
Calcium (Ca^{2+})	0.0001	10
Mammalian neuron		
Potassium (K^+)	140	5
Sodium (Na^+)	5–15	145
Chloride (Cl^-)	4–30	110
Calcium (Ca^{2+})	0.0001	1–2

$$V_m = \frac{RT}{F} \ln\left(\frac{P_{Na}[Na^+]_{out} + P_K[K^+]_{out} + P_{Cl}[Cl^-]_{in}}{P_{Na}[Na^+]_{in} + P_K[K^+]_{in} + P_{Cl}[Cl^-]_{out}} \right)$$

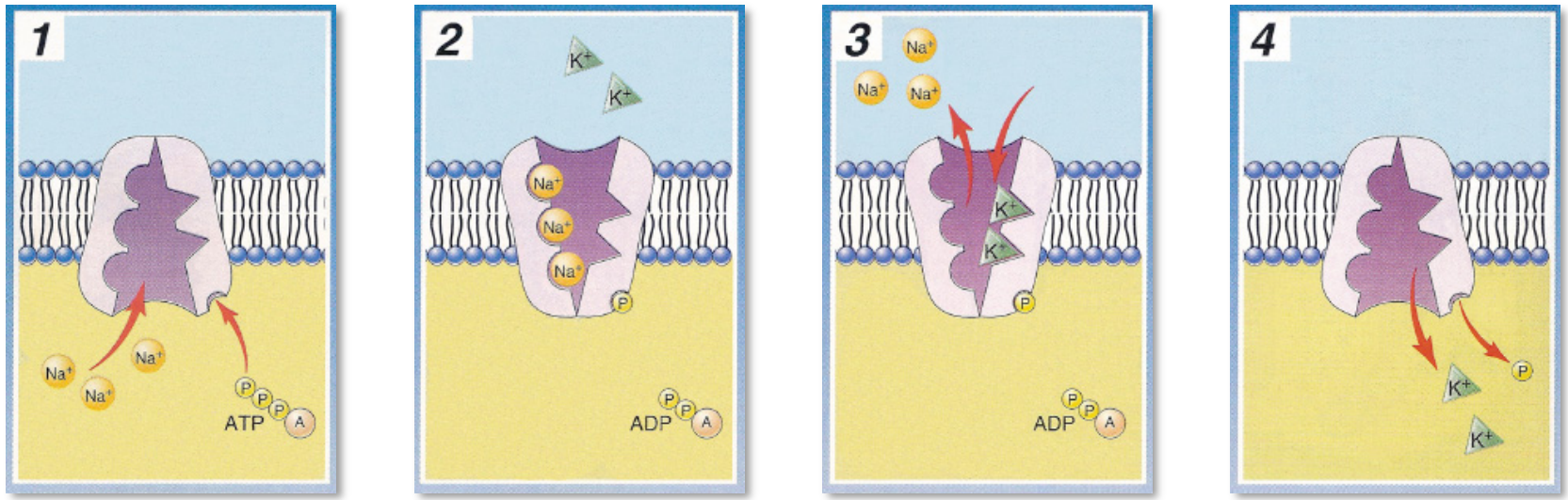
Goldman equation for transmembrane potential

For a mammalian neuron at rest, $V_m \approx -60$ mV, corresponding to $E \approx 6 \times 10^6$ V/m.

- There are ~100 billion neurons and ~100 trillion synapses in the human brain.

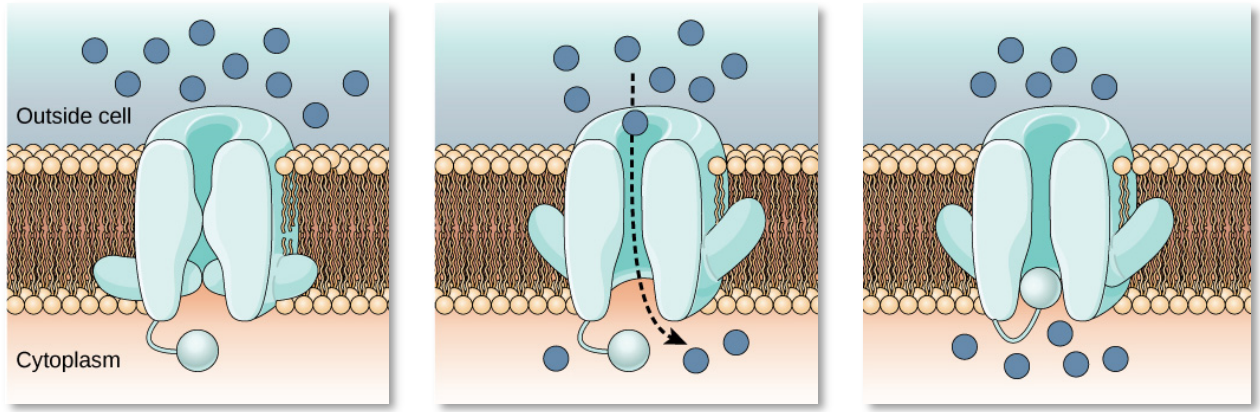
Inspiration: biological neuron

Transmembrane potential is maintained by sodium-potassium ion pumps:



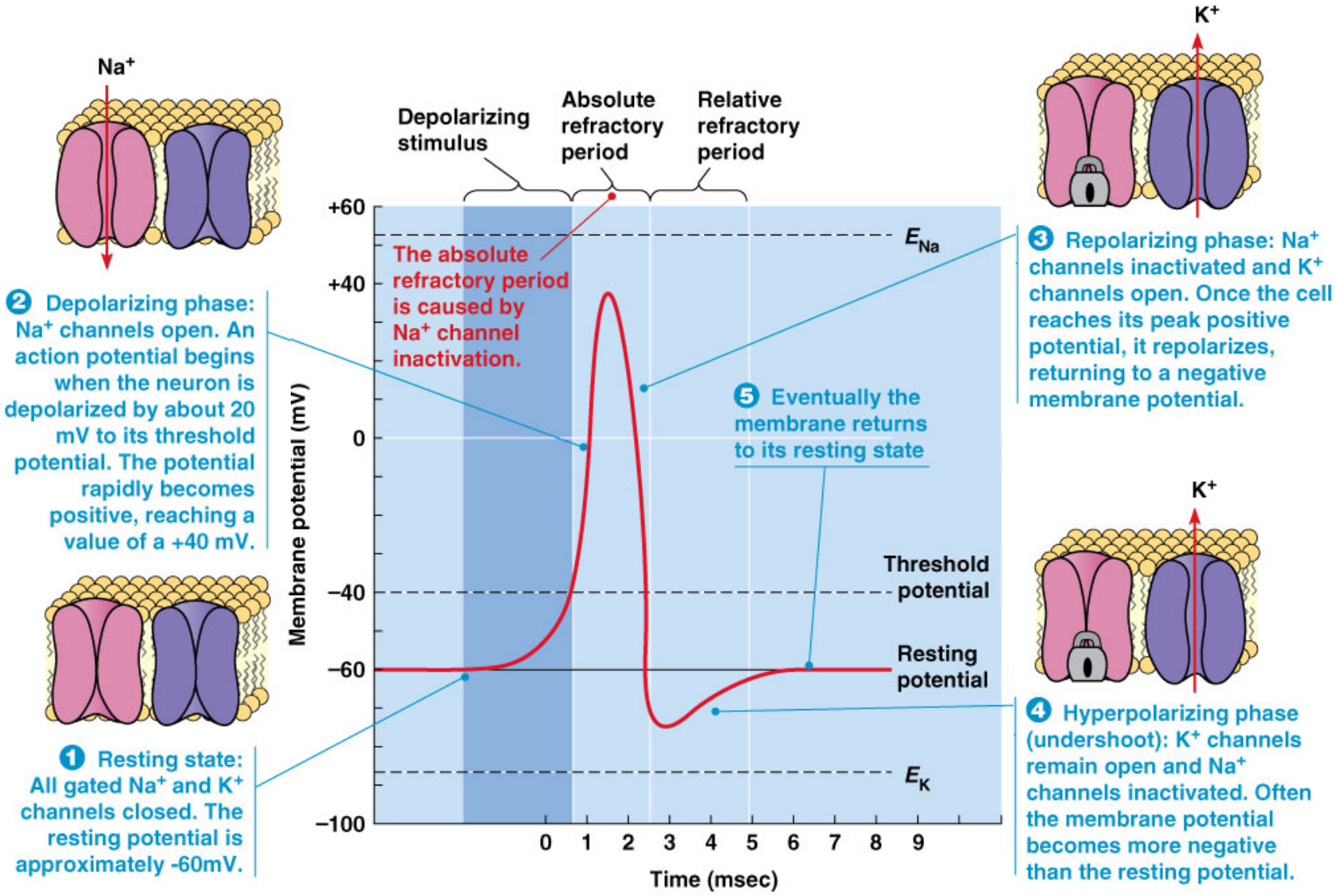
Voltage-gated sodium channels open in response to a drop in V_m.

This causes a further drop in V_m.



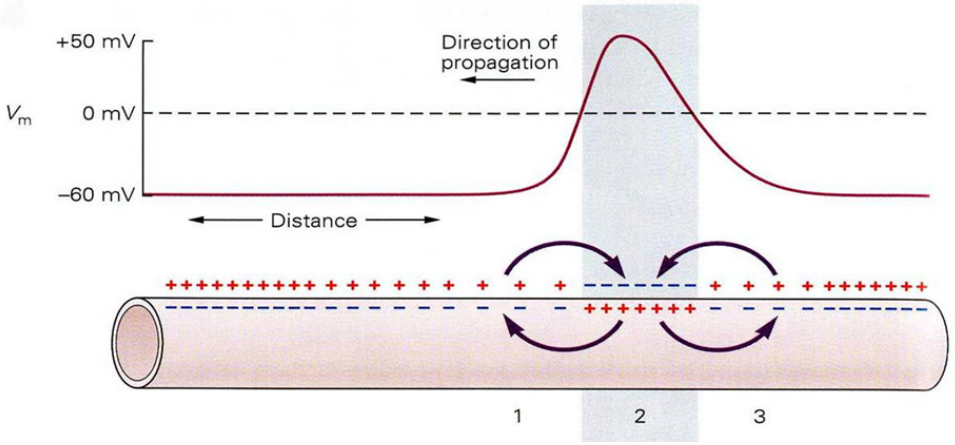
- Ion pumps consume most of the energy budget of the neuron.

Inspiration: biological neuron



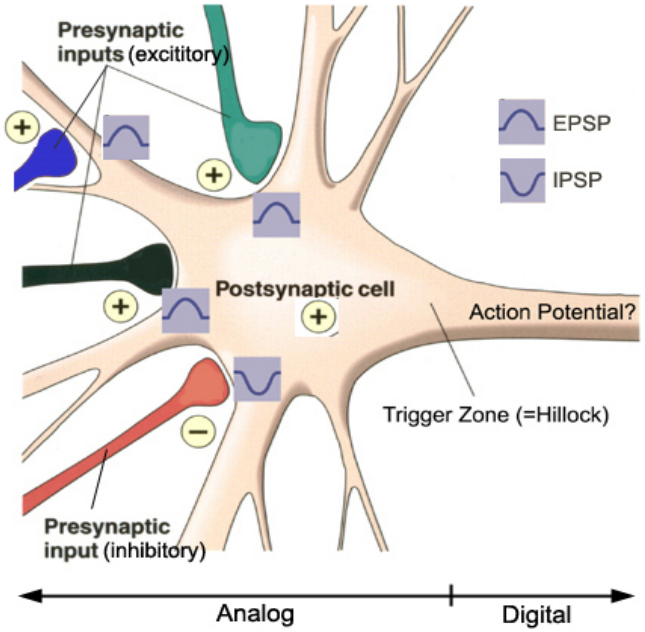
- The result is a travelling wave – a depolarisation event triggers the same process nearby.

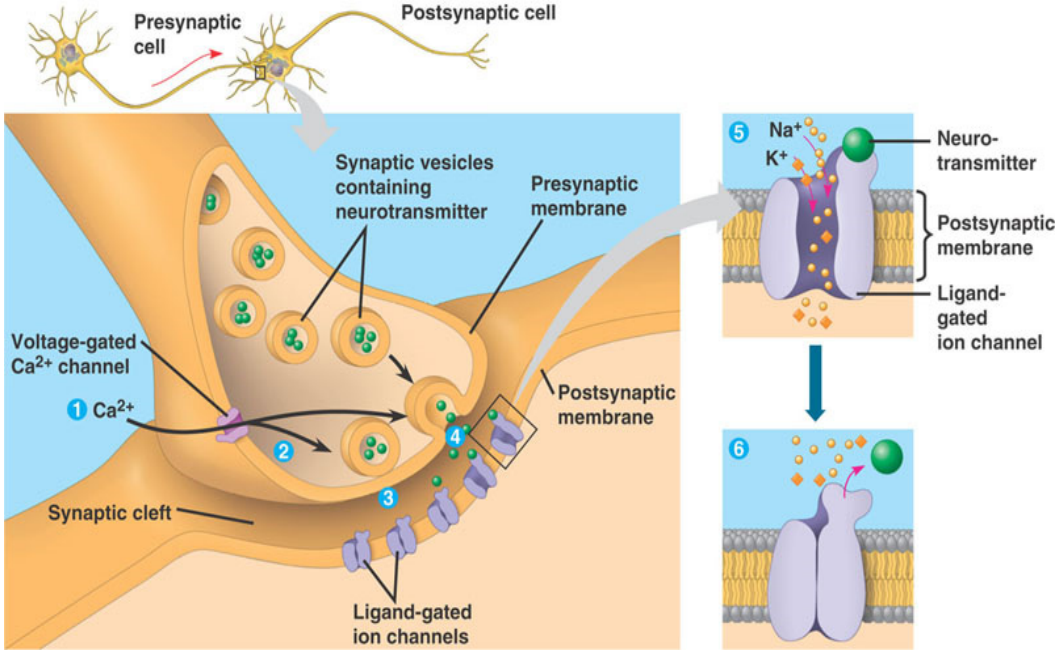
Inspiration: biological neuron



Membrane depolarisation wave travels at 1-100 m/s. It takes several ms to restore the trans-membrane potential.

Signalling between neurons is chemical: voltage gated release of neuromediators.





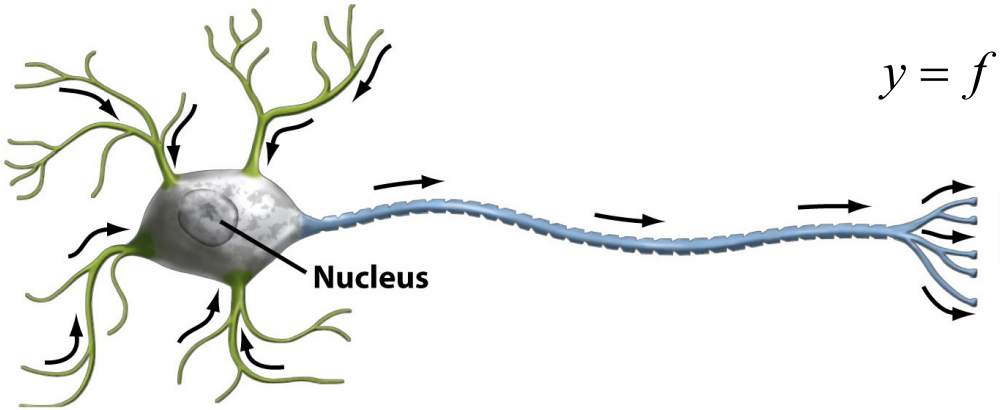
A neuron collects, moderates, and processes electrical signals; then it passes a signal on.

Artificial neuron and layers of neurons (1943)

The artificial "neuron" was inspired by the biological neuron:

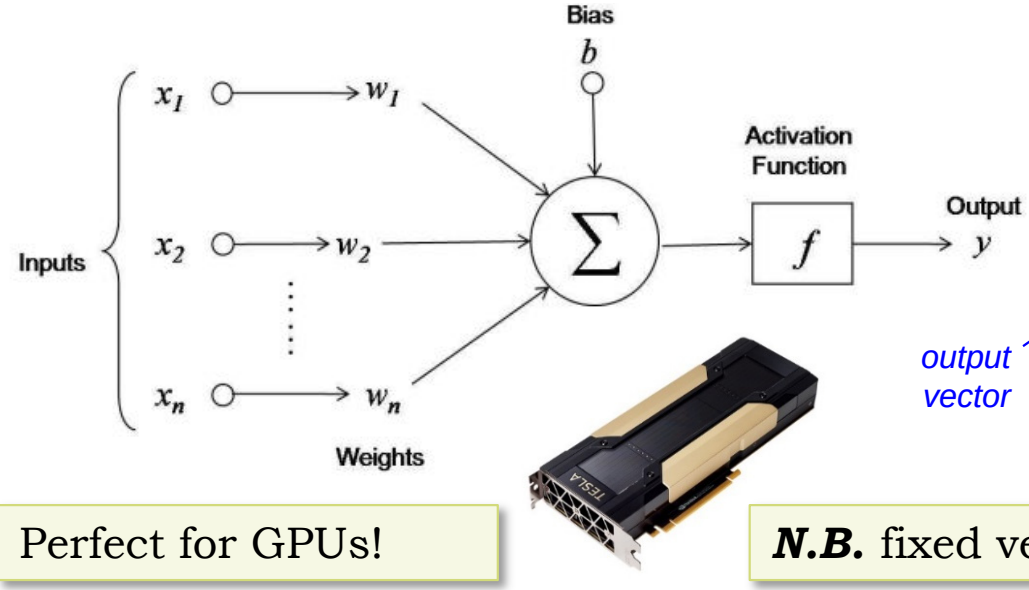
The mathematics maps nicely onto matrix-vector arithmetic and array operations:

$$y = f\left(\sum_k w_k x_k + b\right) \quad \Rightarrow \quad y = f\left(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b\right)$$



Stack multiple neurons vertically:

$$\begin{cases} y_1 = f\left(\mathbf{w}_1^{\mathrm{T}}\mathbf{x} + b_1\right) & \leftarrow \text{neuron 1} \\ \cdots \\ y_n = f\left(\mathbf{w}_n^{\mathrm{T}}\mathbf{x} + b_n\right) & \leftarrow \text{neuron n} \end{cases} \quad \text{layer}$$



$$\Downarrow$$

$$\mathbf{y} = f\left(\mathbf{W}\mathbf{x} + \mathbf{b}\right)$$

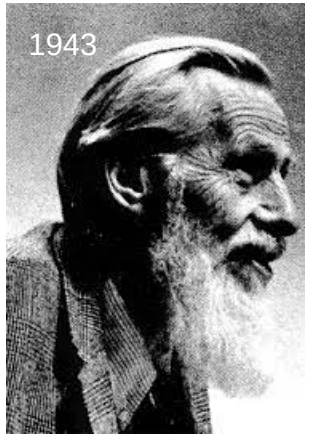
input vector

output vector

activation function

weight matrix

bias vector



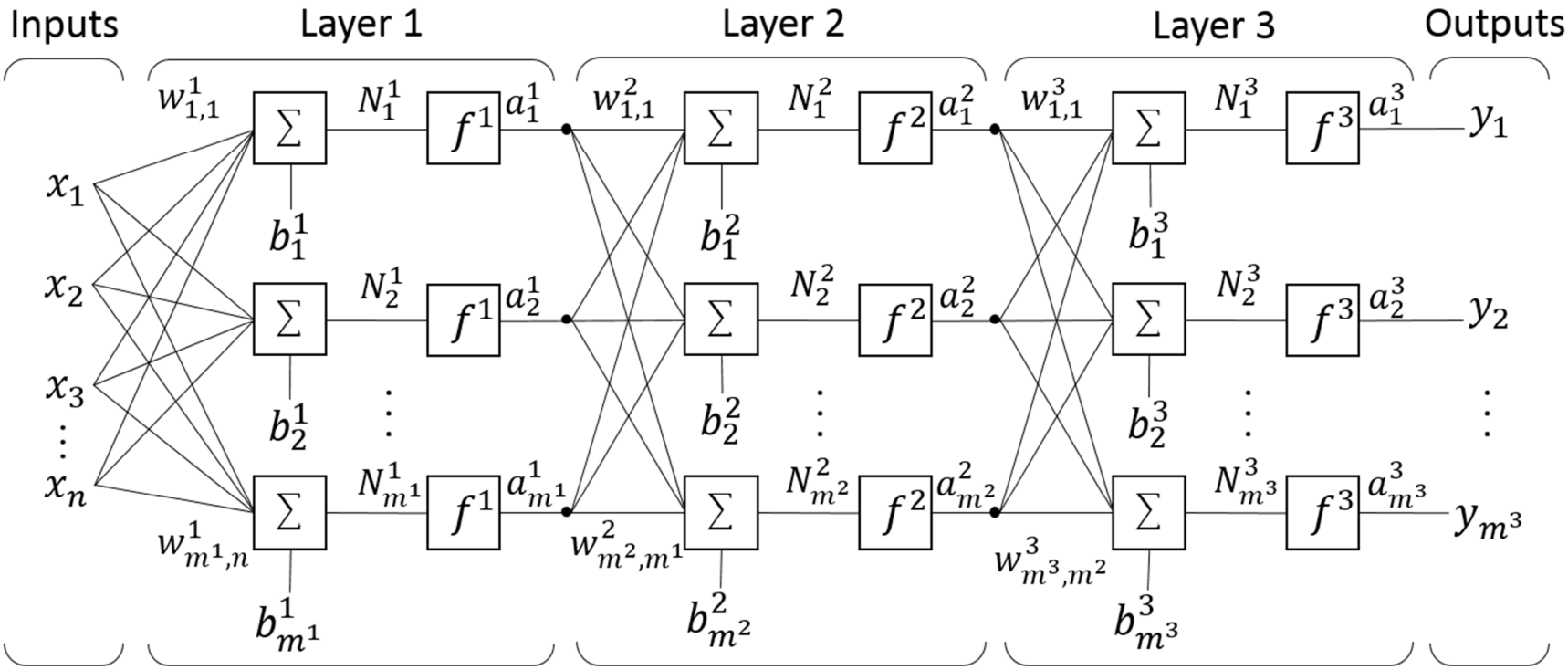
Warren McCulloch

Perfect for GPUs!

N.B. fixed vector dimensions!

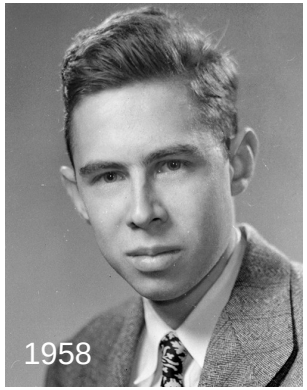
A deep theorem (1989): networks of such "neurons" can model any function to any accuracy.

Networks of artificial neurons (1958 - ...)



- Neural networks are made of connected layers of neurons.
- Output of the previous layer is the input of the next layer.
- For the three-layer network shown above:

$$\mathbf{y} = f_3\left(\mathbf{W}_3 f_2\left(\mathbf{W}_2 f_1\left(\mathbf{W}_1\mathbf{x}+\mathbf{b}_1\right)+\mathbf{b}_2\right)+\mathbf{b}_3\right)$$



Frank Rosenblatt

Weights and biases are the parameters of the neural network, activation functions decided by user.

Efficient neural network training (1970)

Neural network "training" is glorified least squares fitting. Need the gradient…

$$y = f\left(\sum_k w_k x_k + b\right)$$

neural network layer

$$\left\{\mathbf{x}^{(n)}, y^{(n)}\right\}$$ ← *training database*

$$\Omega(\mathbf{w}, b) = \sum_n \left[y^{(n)} - f\left(\sum_k w_k x_k^{(n)} + b\right)\right]^2$$

least squares error

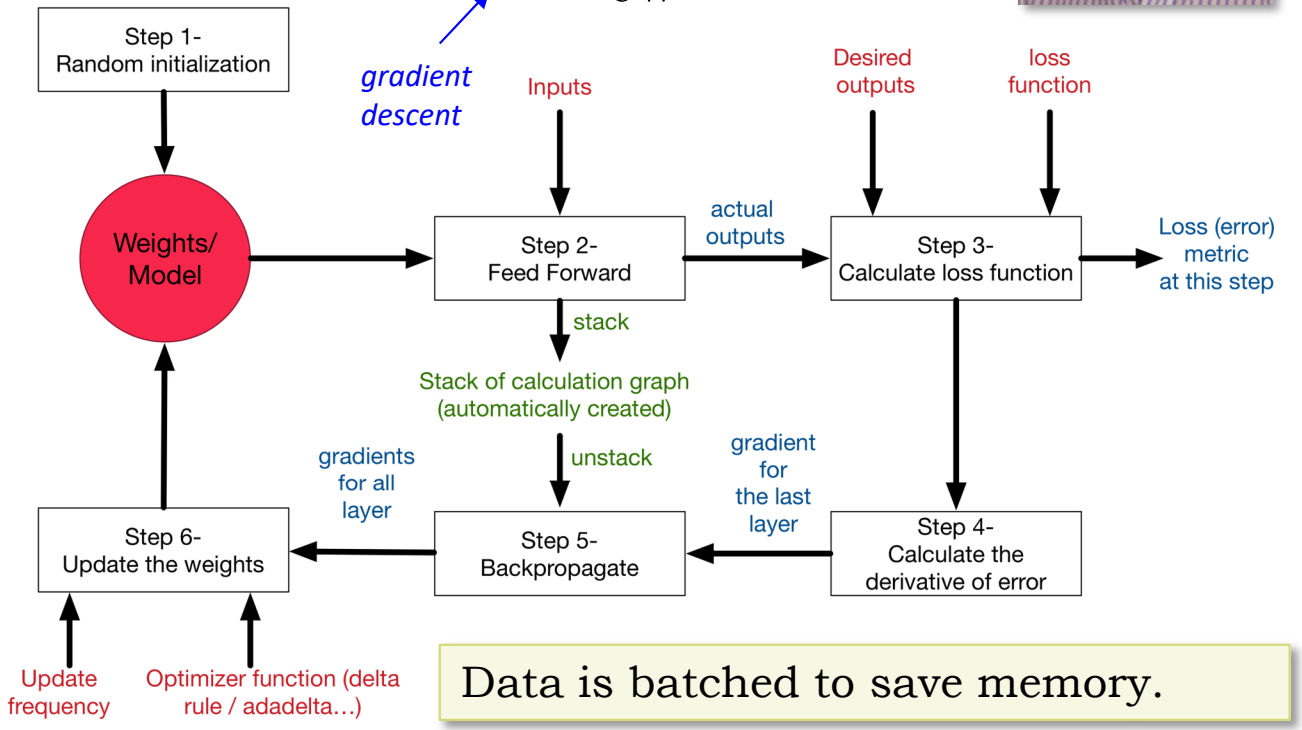
$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \Omega(\mathbf{w}, b)}{\partial \mathbf{w}}$$

gradient descent

Seppo Linnainmaa

training set
millions of question-answer pairs needed, even for small nets (may be obtained by simulation)

validation set
separate dataset that is used to benchmark the performance of the neural network

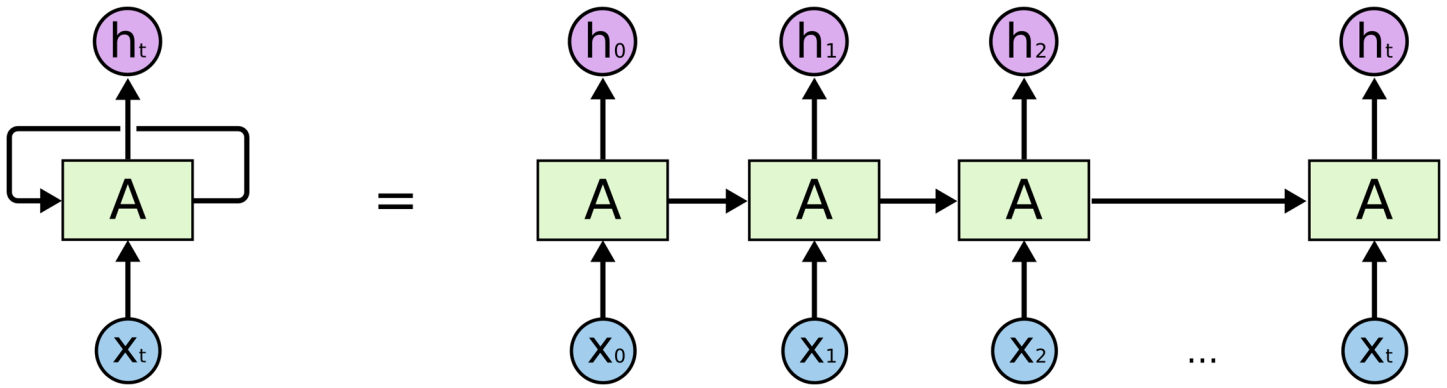


Data is batched to save memory.

Training is time-consuming, GPUs are essential. There are thorny numerical technicalities.

Recurrent neural nets and LSTM layers (1997)

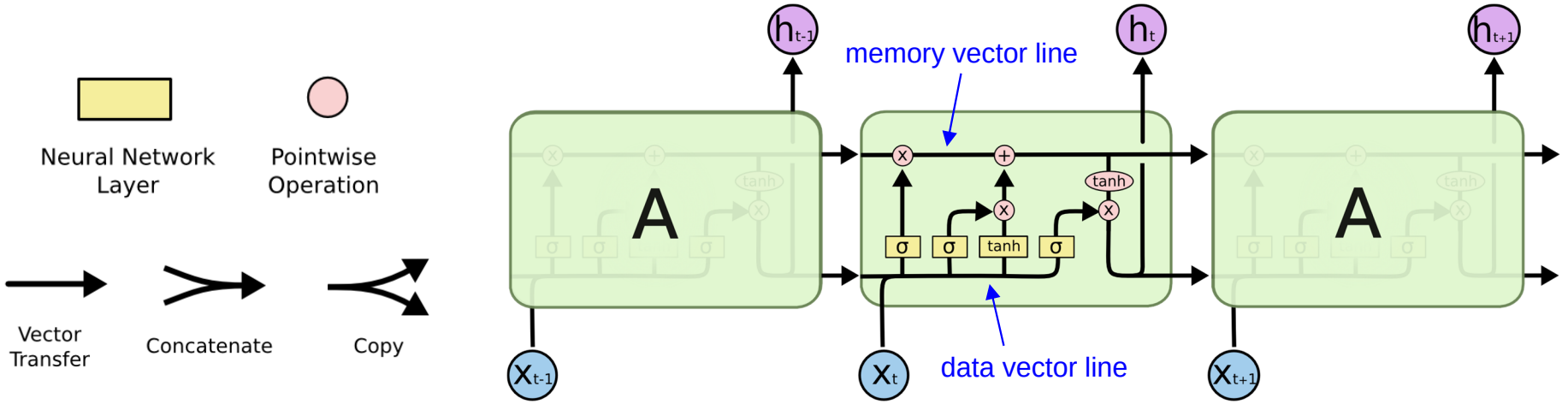
Input order does matter (text, amino acid sequence, time domain signals, *etc.*):



BUT

training for long-range dependences is too slow

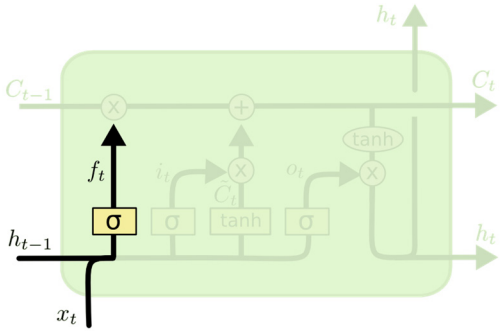
Long short-term memory (LSTM) layers are designed to mitigate the problem:



Layer schematics by Christopher Olah (https://colah.github.io)

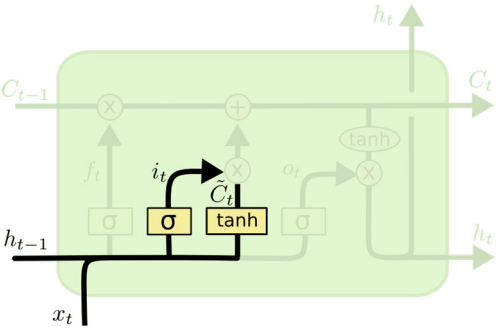
The anatomy of an LSTM layer (1997)

forget gate



$$\mathbf{f}_t = \sigma\left(\mathbf{W}_\mathrm{f}\left[\mathbf{h}_{t-1};\mathbf{x}_t\right]+\mathbf{b}_\mathrm{f}\right)$$

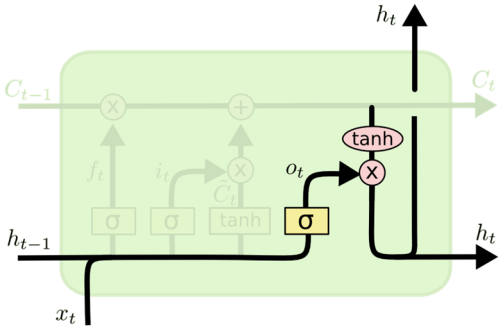
input gate and input



$$\mathbf{i}_t = \sigma\left(\mathbf{W}_\mathrm{in}\left[\mathbf{h}_{t-1};\mathbf{x}_t\right]+\mathbf{b}_\mathrm{in}\right)$$

$$\tilde{\mathbf{c}}_t = \tanh\left(\mathbf{W}_\mathrm{C}\left[\mathbf{h}_{t-1};\mathbf{x}_t\right]+\mathbf{b}_\mathrm{C}\right)$$

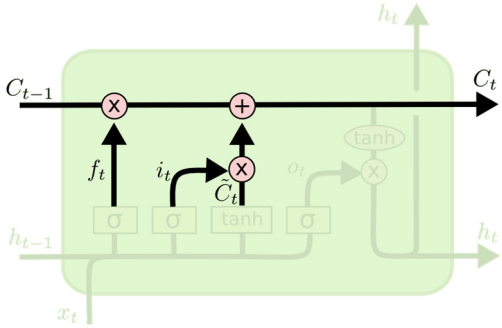
filter and output



$$\mathbf{o}_t = \sigma\left(\mathbf{W}_\mathrm{out}\left[\mathbf{h}_{t-1};\mathbf{x}_t\right]+\mathbf{b}_\mathrm{out}\right)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)$$

internal state update



$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

Remaining issues:

weak parallelisation

slow training

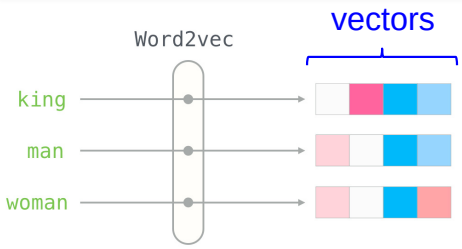
very long range dependencies still a problem

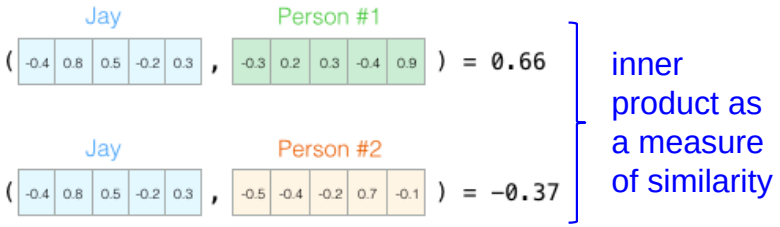
DEAD END

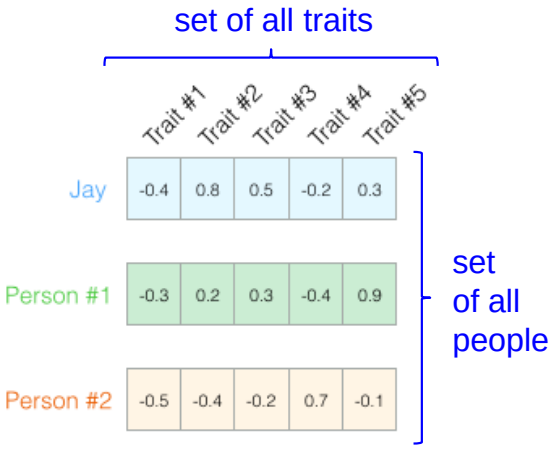
Layer schematics by Christopher Olah (https://colah.github.io)

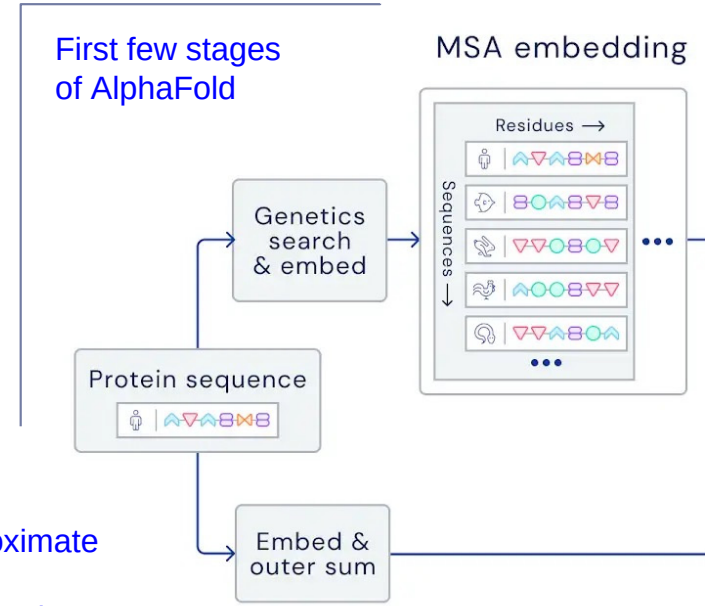
Road to transformers: embedding (1957)

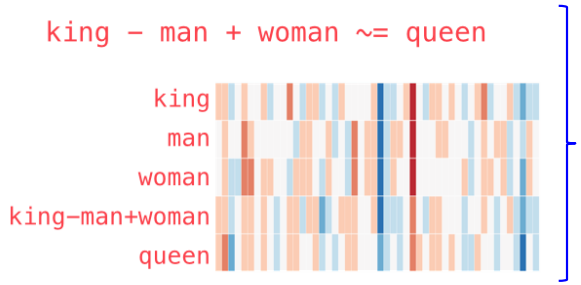
An **embedding** is a mapping of a discrete variable (*e.g.* words of a language) to vectors in a continuous space.

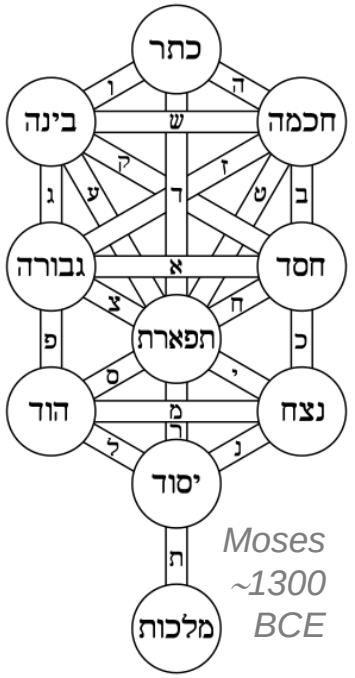


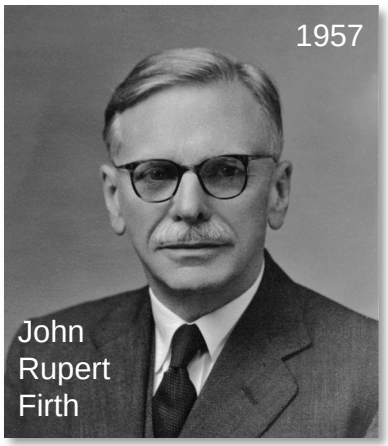










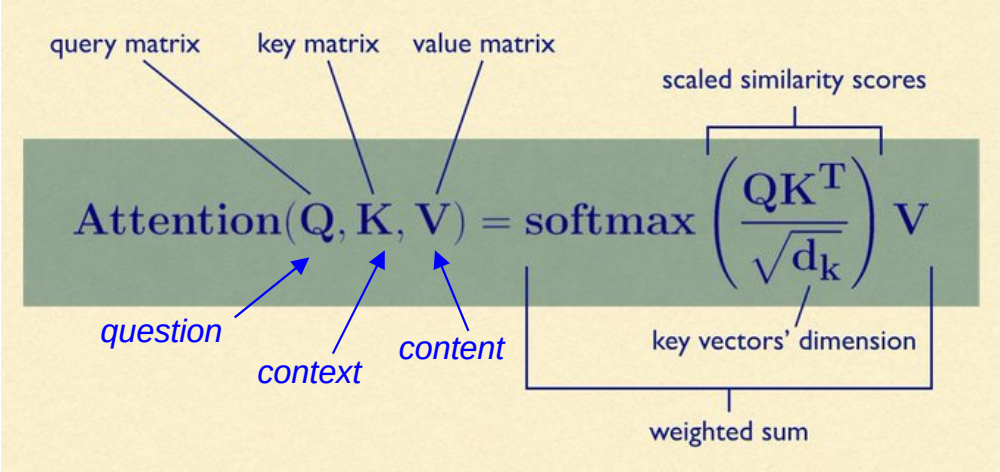


- Words are now mapped into elements of a vector space; we are really good with vector spaces!

Road to transformers: attention mechanism (2017)

Vasvani et al. (132,000 citations):

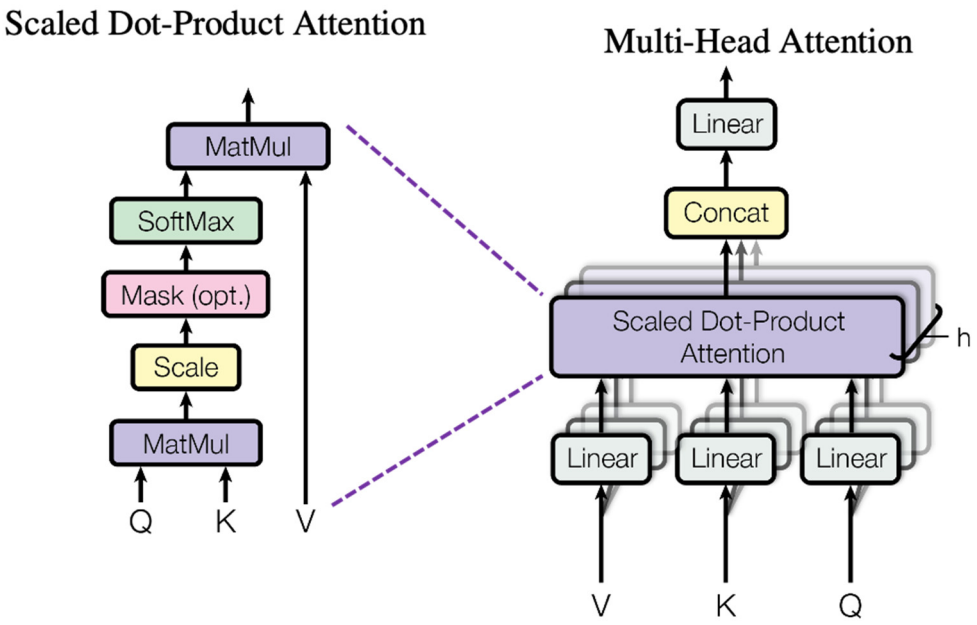
"We propose a novel, simple network architecture based solely on an attention mechanism, dispensing with recurrence and convolutions..."



Query vectors: after embedding, one vector per word ("token") in the input, this is the question being asked.

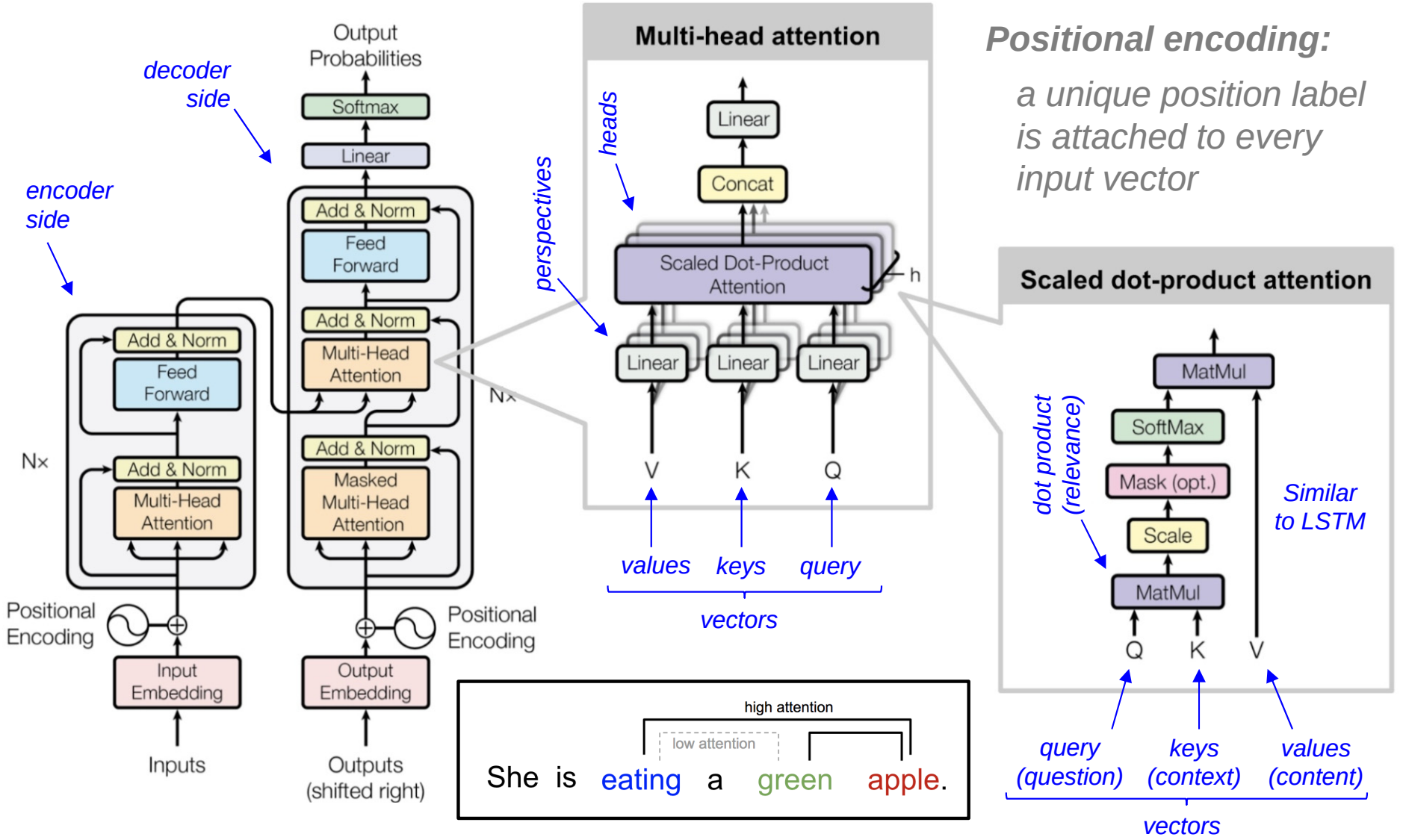
Key vectors: a basis set of contexts; the inner product with a query vector determines the relevance of a query within each context.

Value vector: a basis set of outputs; a weighted linear combination is returned based on the query-context inner product outcome.



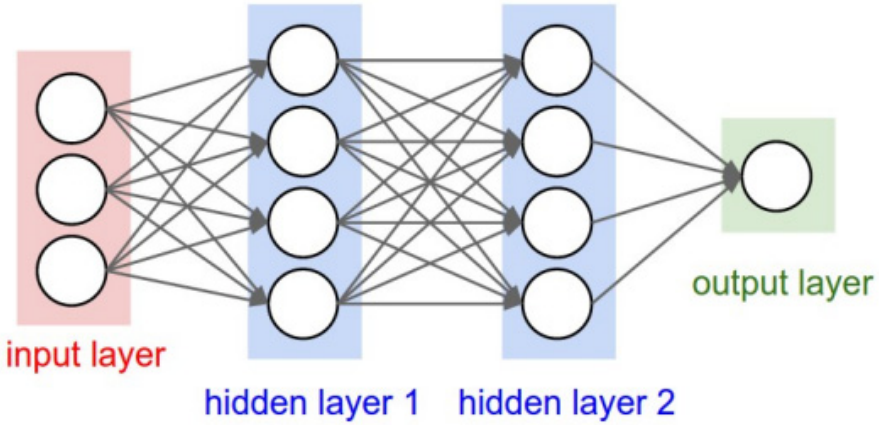
- Transformers: no recurrence, no convolutions (https://doi.org/10.48550/arXiv.1706.03762)

Transformers (AlphaFold, ChatGPT: today)



- Transformers: no recurrence, no convolutions (https://doi.org/10.48550/arXiv.1706.03762)

Interpretability problem



In fully connected nets, intermediate signals are... ENCRYPTED!

Recital 71 of GDPR Law (EU):

"[...] the right not to be subject to a decision [...] based solely on automated processing"

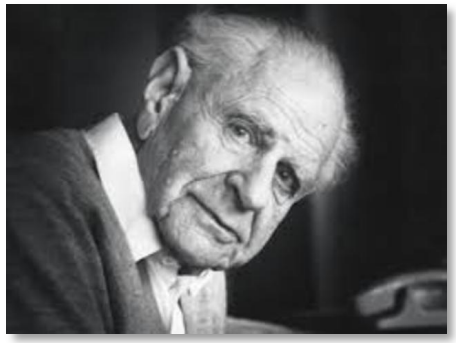
Equal Credit Opportunity Act (US):

"The statement of reasons for adverse action [...] must be specific and indicate the principal reason(s) for the adverse action."

René Descartes
(in *The Discourse*, 1637):

1. Doubt everything.
2. Break every problem into smaller parts.
3. Solve simplest problems first and build from there.

Karl Popper
(in *Objective Knowledge*):



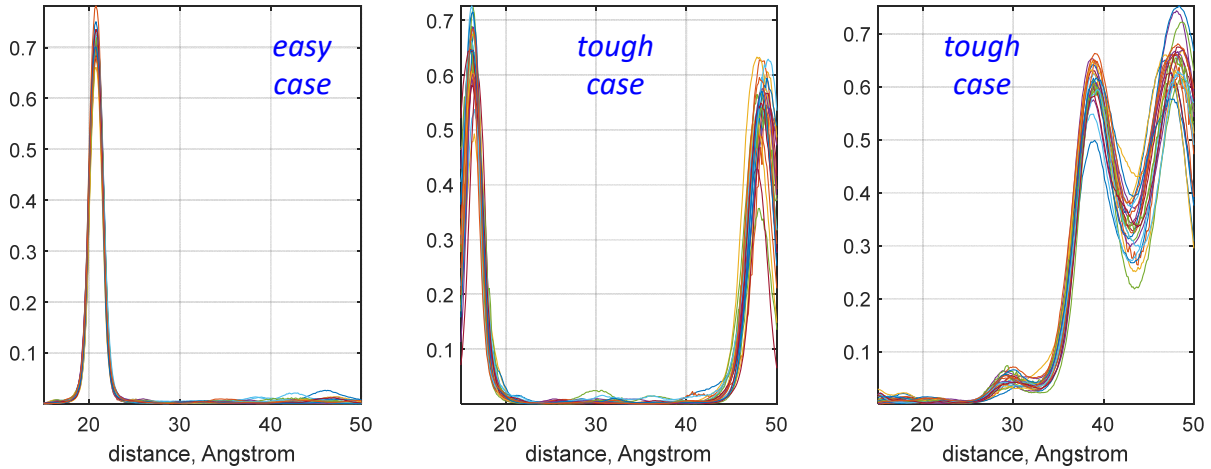
1. Propose a theory.
2. Test the predictions of the theory; the best theories are those that survive.

not testable = not science

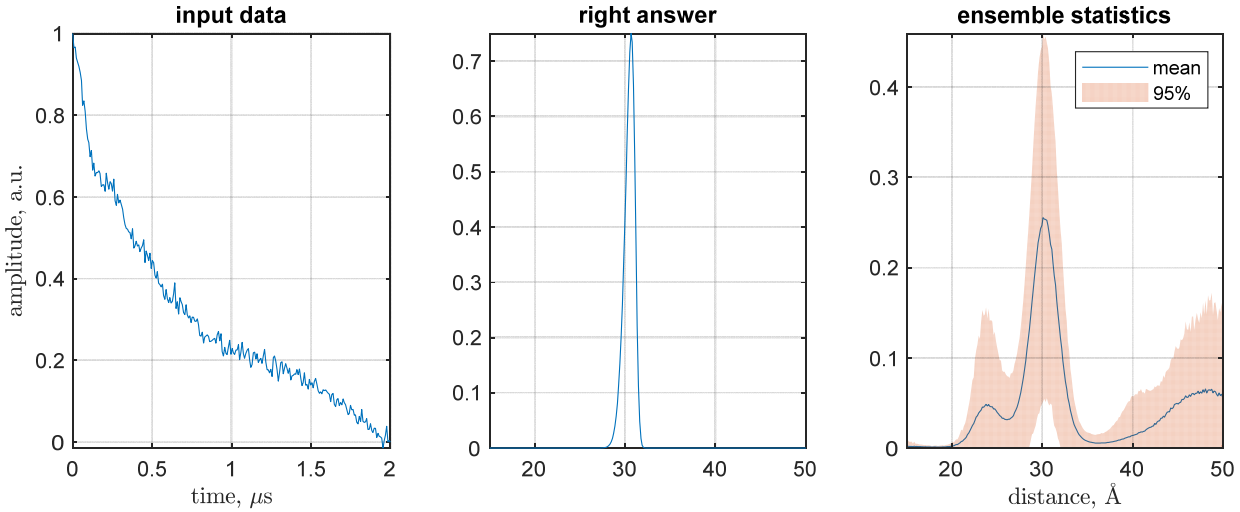
Further discussion: https://doi.org/10.1073/pnas.2016917118

Uncertainty quantification

Train an ensemble of networks and do the statistics:



Each net trained on a different database.

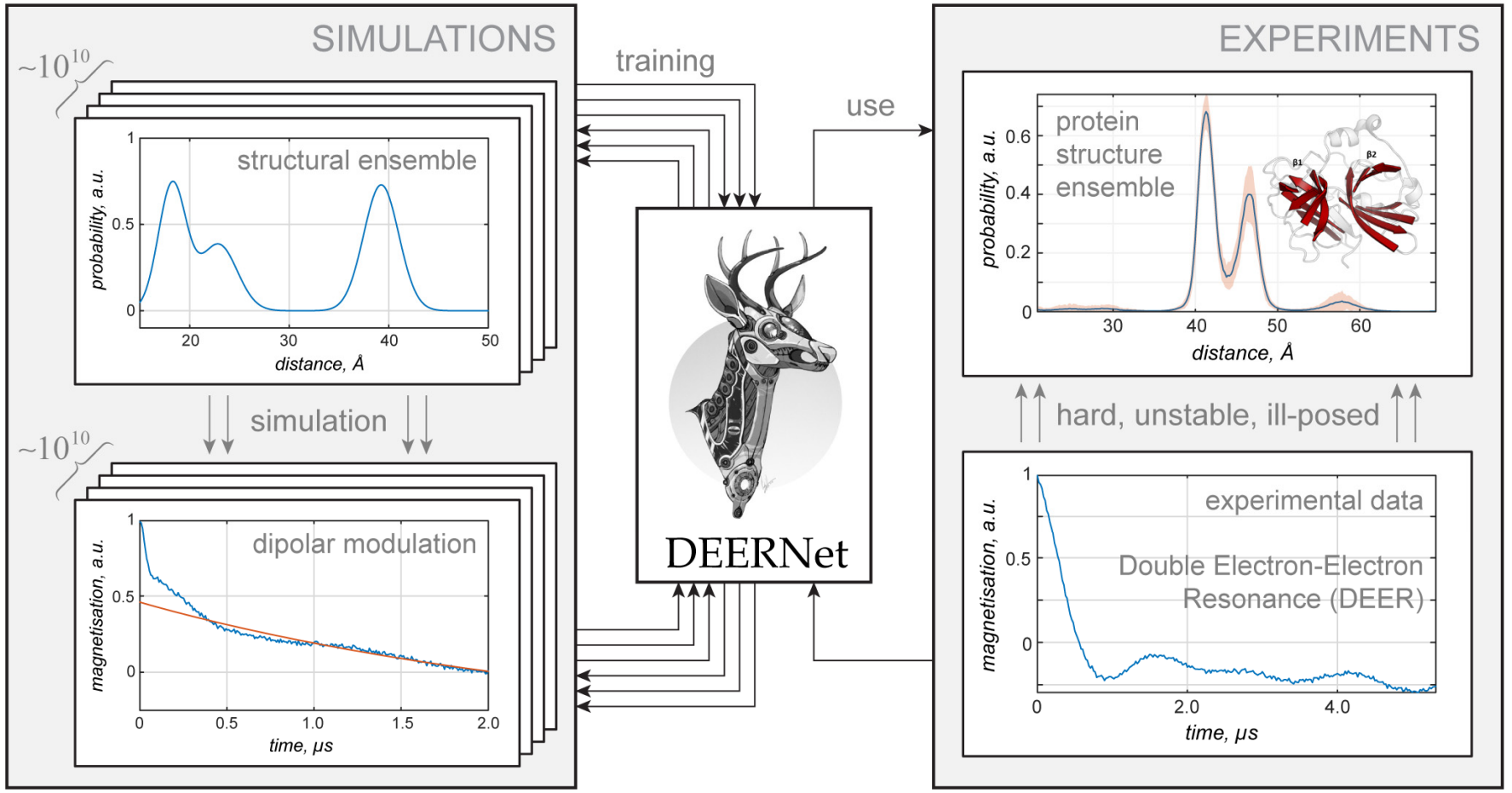


When the S/N is too bad, the nets fail gracefully.

Further discussion: https://doi.org/10.1016/j.jmr.2022.107186

Training set generation, DEERNet

Not enough DEER data in the world. Try to train a neural net on simulations?



"Hmm… this creates effectively an infinite training dataset!" – Jake Keeley

Further discussion: https://doi.org/10.1016/j.jmr.2022.107186

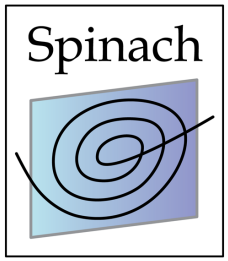
Training set generation, PyRuv8

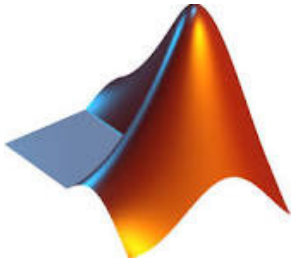
3D HNCA acquisition and labelling:

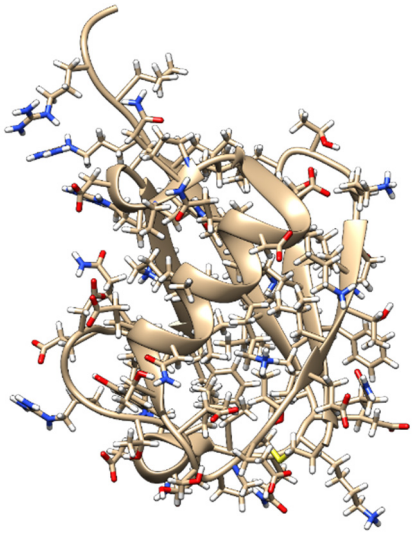
To acquire (sparse sampling): **48 hours**

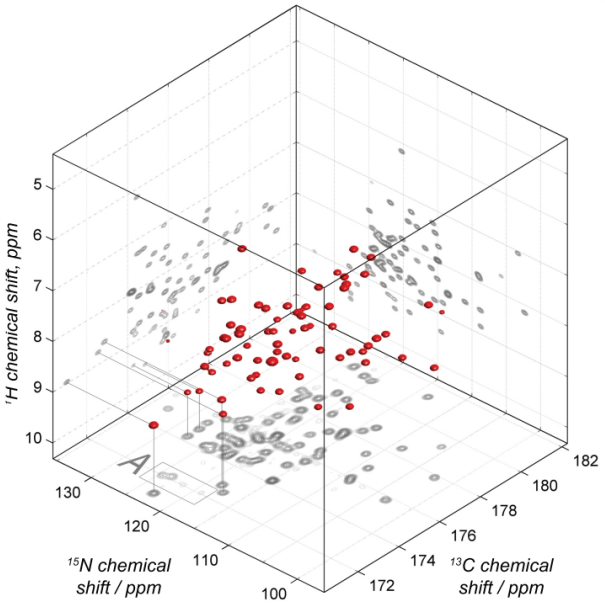
To process, assign, and label: **2 weeks**

must use synthetic NMR data



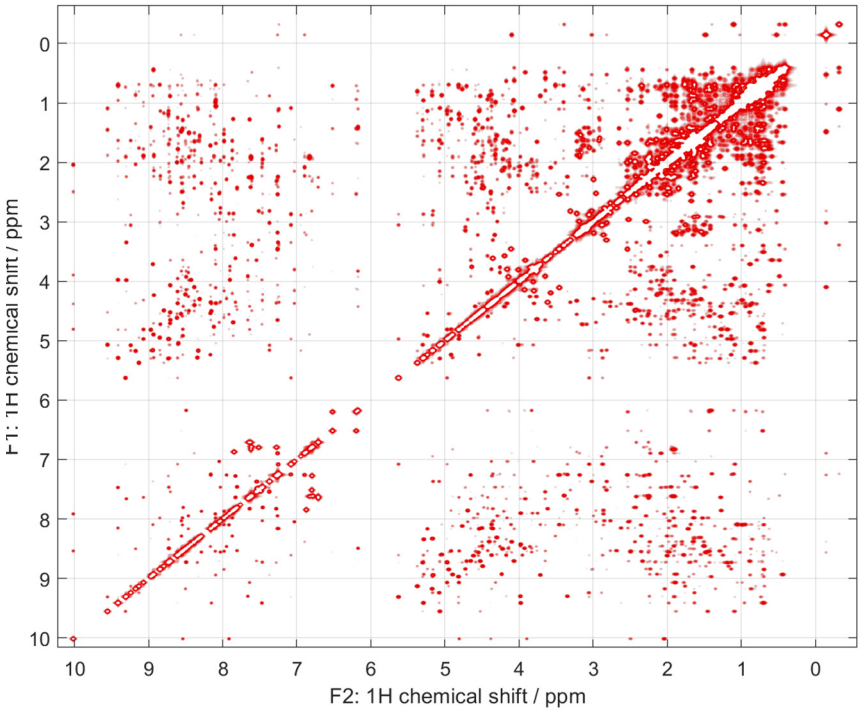






Storage cost of *one* HNCA NMR spectrum:

$$1024\left({}^{1}\mathrm{H}\right) \times 2048\left({}^{13}\mathrm{C}\right) \times 256\left({}^{15}\mathrm{N}\right) \times$$
$$\times\, 2\left(\mathrm{Re}+\mathrm{Im}\right) \times 8\left(\mathrm{bytes}\right) = \mathbf{16\,GB}$$



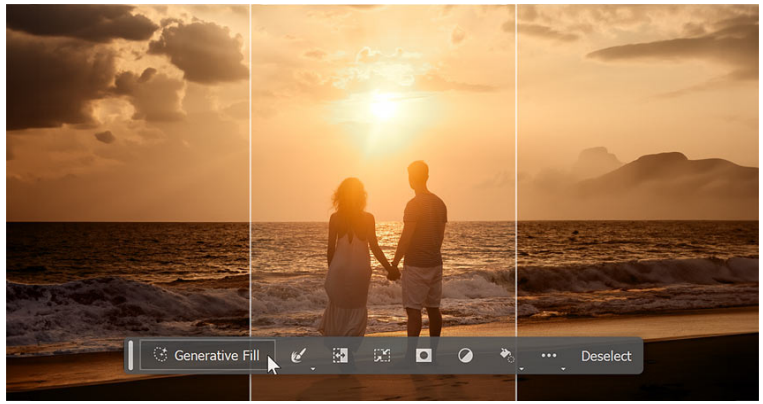
STILL IMPOSSIBLE TO STORE!

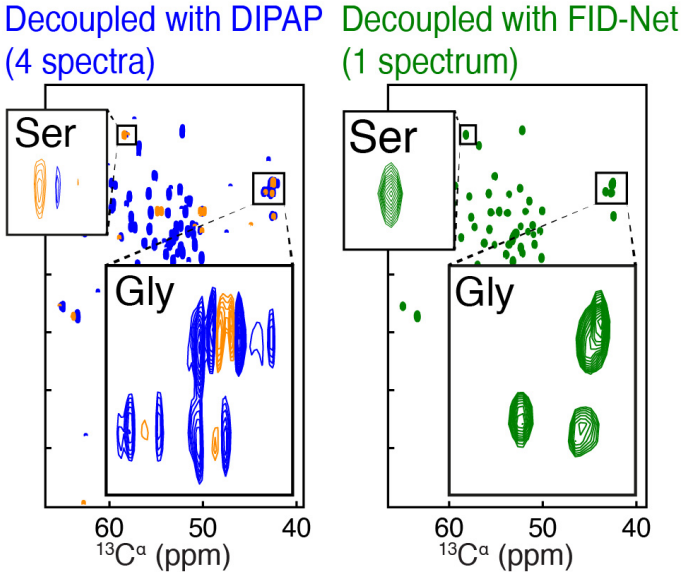
Further discussion: https://doi.org/10.1126/sciadv.ado0403

Examples of Magnetic Resonance AI

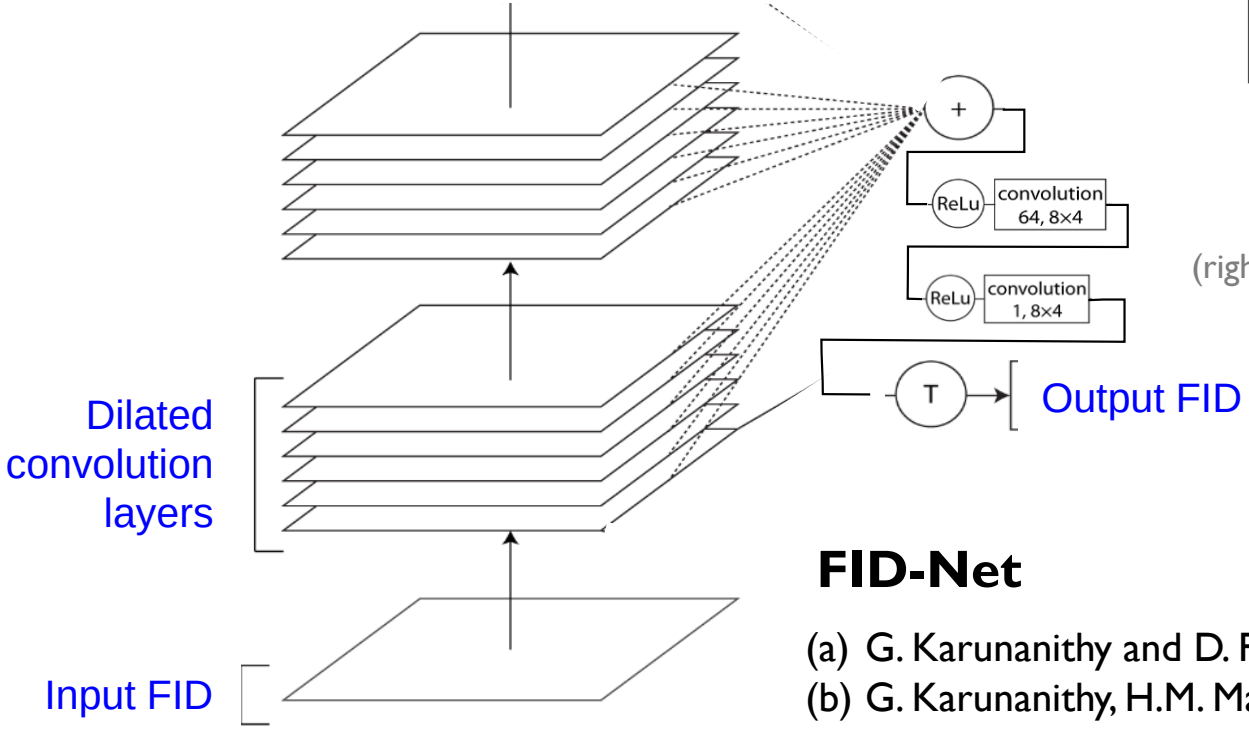
nD FID may be viewed as an image:

paint in the missing pixels!





$^{13}C^{\alpha}$-detected $^{13}C^{\alpha}$-^{13}CO spectrum
(left) Traditional DIPAP with 4 input spectra
(right) FID-Net spectrum with 1 input spectrum



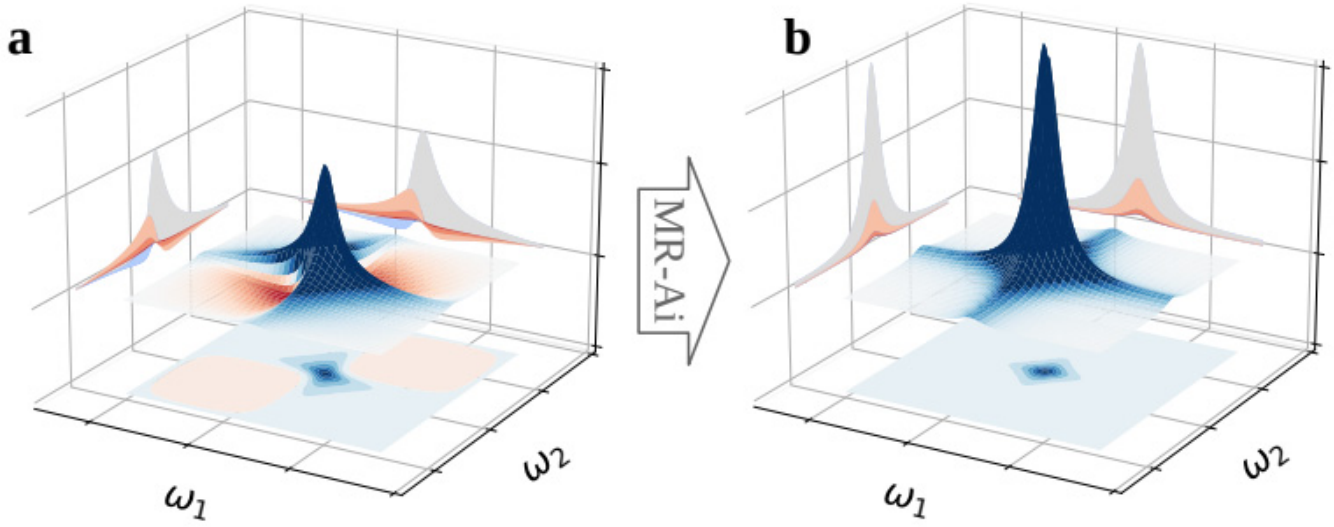
FID-Net

(a) G. Karunanithy and D. F. Hansen, *J. Biomol NMR* (2021)
(b) G. Karunanithy, H.M. Mackenzie, D. F. Hansen, JACS (2021)

A special case of DNN image processing: huge amount of infrastructure already in place.

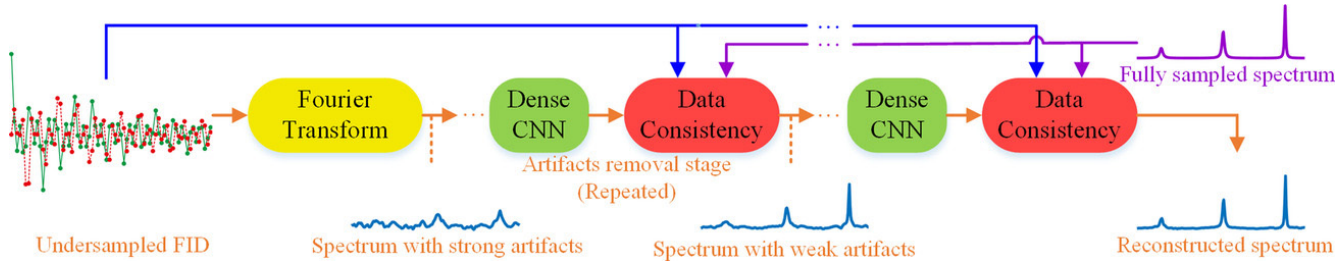
Examples of Magnetic Resonance AI

Data processing algorithms may be trained by example:



Amir Jahangiri

Missing redundant data points may be painted in:

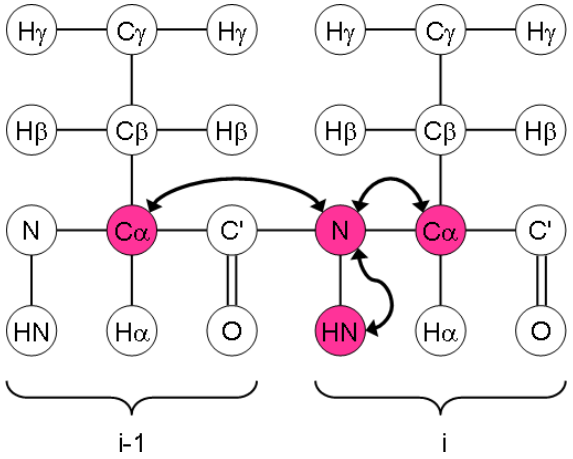


+ peak picking, multiplet identification, baseline correction, pulse design, …

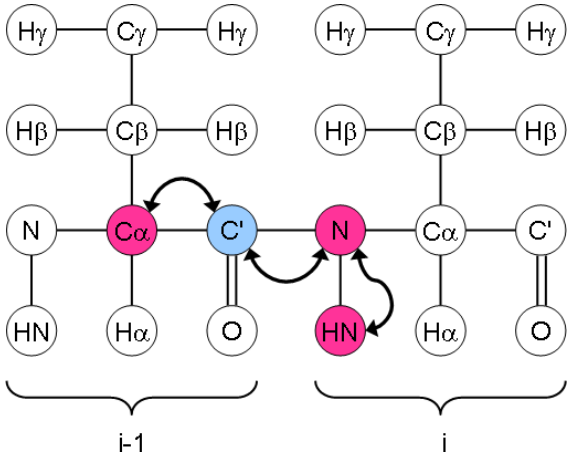
Further discussion: https://doi.org/10.1016/j.jmr.2022.107342, https://arxiv.org/abs/2405.07657

Examples of Magnetic Resonance AI

3D HNCA sequence:

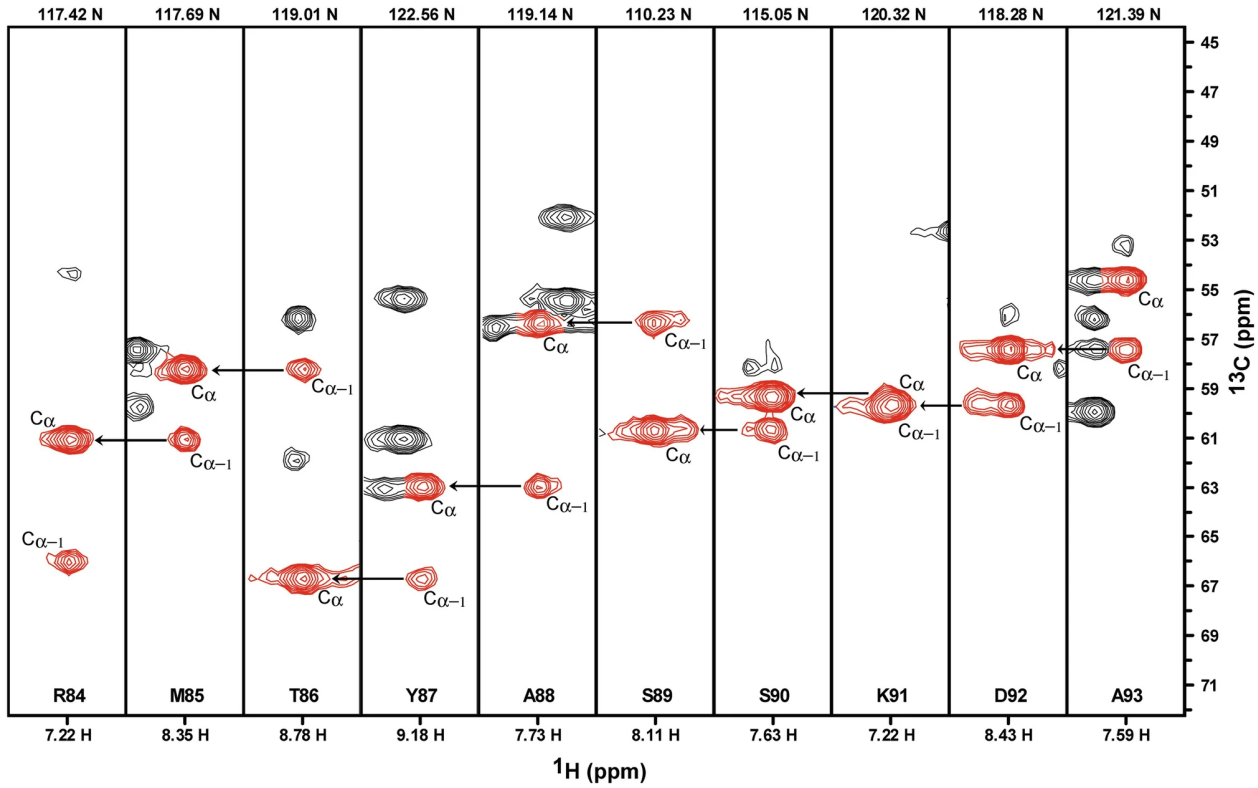


3D HN(CO)CA sequence:



Exceedingly laborious:

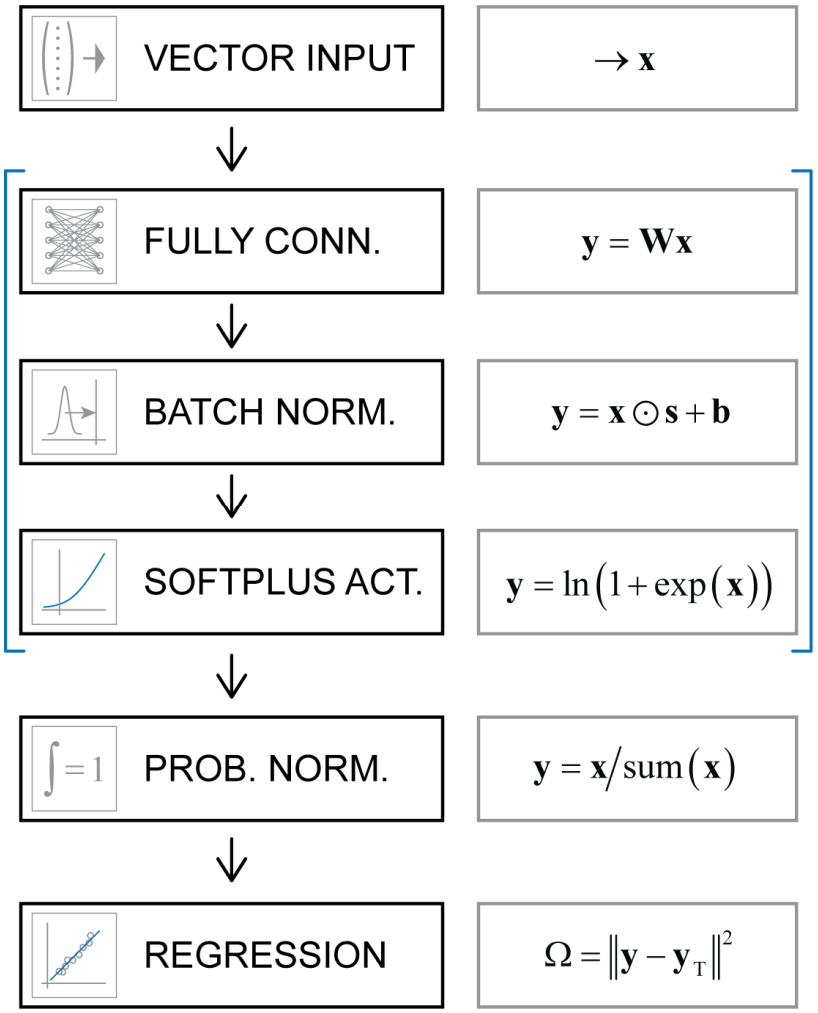
1. Cut 3D HNCA into planes by amide ^{15}N chemical shift.
2. Cut each plane into strips by amide ^{1}H chemical shift.
3. Match ^{13}C chemical shift manually or semi-automatically.
4. Amino acid type cannot be reliably identified at this stage.

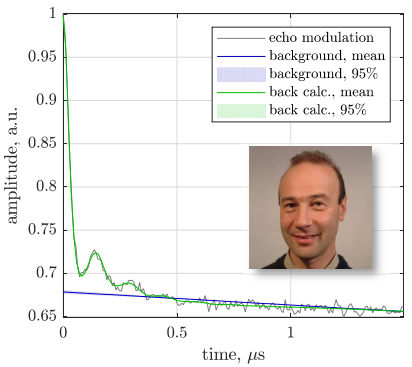


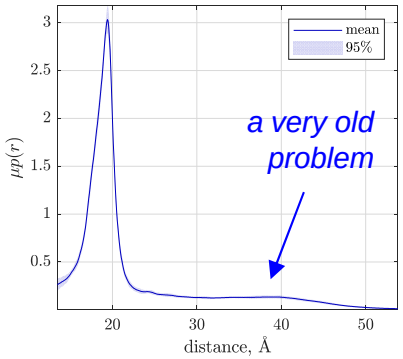
Sequence schematics by Vicky Higman. Further discussion: https://doi.org/10.1126/sciadv.ado0403

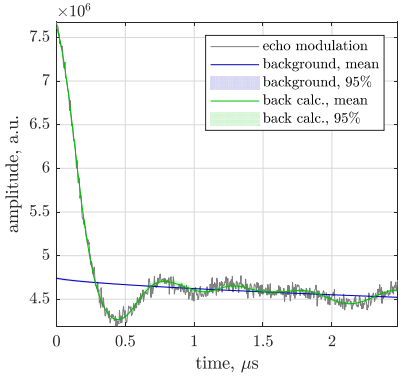
Examples of Magnetic Resonance AI

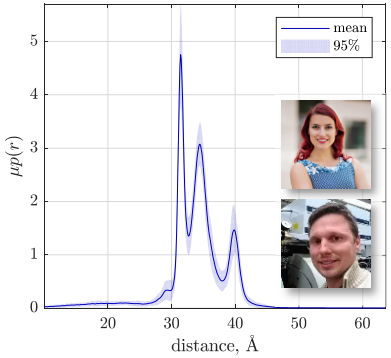
DEERNet (electron spin resonance):











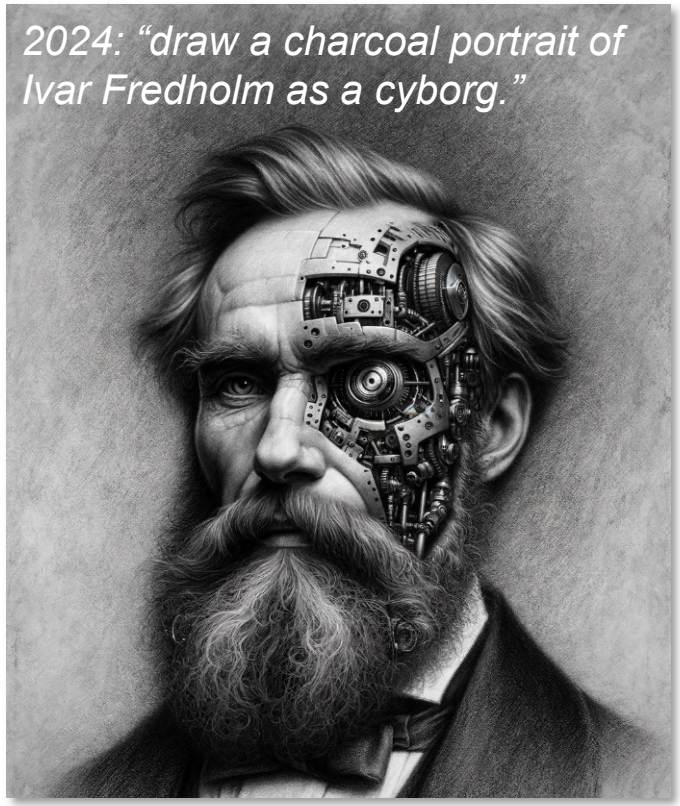
Jake Amey *Jake Keeley* *Tajwar Choudhury* *Gunnar Jeschke* *Steven Worswick*

Further discussion: https://doi.org/10.1126/sciadv.aat5218

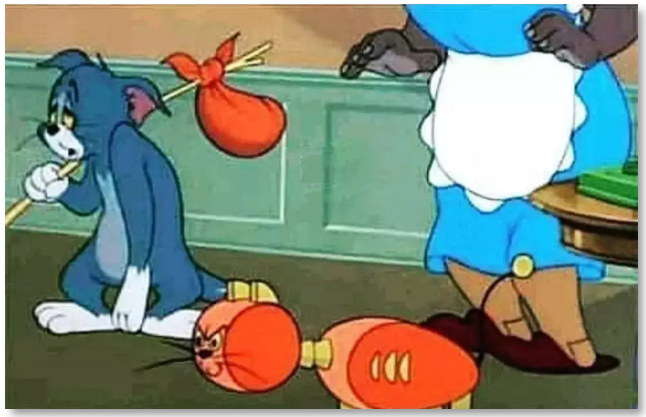
Where is it all going?

1994: "no artificial intelligence can ever replace human creativity – of artists, poets, musicians..."

2024: "draw a charcoal portrait of Ivar Fredholm as a cyborg."



Sun rises. Spins align.
In sunny fields they dance.
Must write a grant.



Finite Volume Method Setup

Consider a 2D domain divided into Voronoi cells. For each cell i, let C_i be the concentration, $\mathbf{V}_i$ the velocity vector, and D the diffusion coefficient.

Governing Equation

ChatGPT on FVM

The general transport equation for the concentration C in a cell i is:

$$\frac{dC_i}{dT} + \nabla \cdot (\mathbf{V}C_i) = \nabla \cdot (D\nabla C_i).$$

Using the finite volume method, integrate this equation over a control volume V_i (the Voronoi cell i):

$$\int_{V_i} \frac{dC_i}{dT}\, dV + \int_{V_i} \nabla \cdot (\mathbf{V}C_i)\, dV = \int_{V_i} \nabla \cdot (D\nabla C_i)\, dV.$$

Applying the divergence theorem, this becomes:

$$\frac{d}{dT}\left(\int_{V_i} C_i\, dV\right) + \int_{S_i} (\mathbf{V}C_i) \cdot \mathbf{n}\, dS = \int_{S_i} (D\nabla C_i) \cdot \mathbf{n}\, dS,$$

where S_i is the boundary of the Voronoi cell i and $\mathbf{n}$ is the outward normal vector on S_i.

Discretization

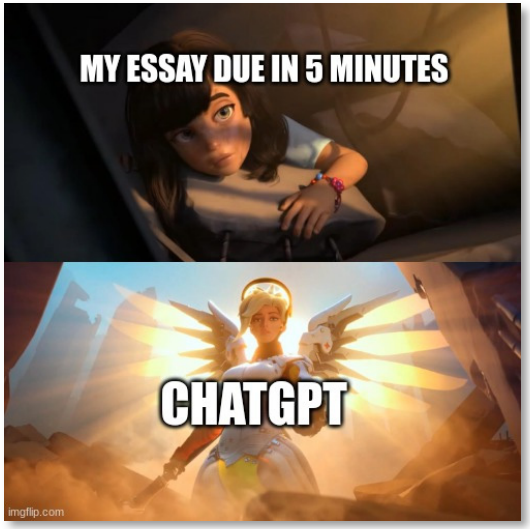
Assume the concentration C_i is constant within each Voronoi cell i:

$$V_i \frac{dC_i}{dT} + \sum_{j \in \mathcal{N}(i)} \int_{S_{ij}} (\mathbf{V}C_i) \cdot \mathbf{n}_{ij}\, dS = \sum_{j \in \mathcal{N}(i)} \int_{S_{ij}} (D\nabla C_i) \cdot \mathbf{n}_{ij}\, dS,$$

where $\mathcal{N}(i)$ denotes the set of neighboring cells to cell i, and S_{ij} is the shared boundary between cells i and j.

Advection Term

For the advection term, approximate the flux across the shared boundary S_{ij}:



"For centuries, the philosophical approach to science has been to find fundamental laws that govern reality, to test those laws, and to use their predictive power. **Black-box neural networks amount to blasphemy within that school, but they are irresistible because they... just work.**"

Further discussion: https://doi.org/10.1073/pnas.2016917118

Summary and outlook

1. **It works. How's your math? :)**
2. **Neural networks are easy, but training databases are not.**
3. **NMR spectroscopy is a giant image analysis problem.**
4. **ML is useless without uncertainty analysis and robust treatment of corrupted input data.**
5. **Much of machine learning is about artefacts in the data.**

Two PhD studentships and a 4-year postdoc position available

```
% Start with image input layer
layers=imageInputLayer([65 33 1],...
       'Normalization','none');

% Tapered fully connected layers
layers=[layers; fullyConnectedLayer(1024);
                batchNormalizationLayer();
                softplusLayer('Name','SP0')];
layers=[layers; fullyConnectedLayer(512);
                batchNormalizationLayer();
                softplusLayer('Name','SP1')];
layers=[layers; fullyConnectedLayer(256);
                batchNormalizationLayer();
                softplusLayer('Name','SP2')];
layers=[layers; fullyConnectedLayer(64);
                batchNormalizationLayer();
                softplusLayer('Name','SP3')];
layers=[layers; fullyConnectedLayer(18);
                batchNormalizationLayer();
                softplusLayer('Name','SP4')];

% Probnorm followed by regression
layers=[layers; regressionLayer()];

% Set up the datastore
fds=fileDatastore('D:\GB1','ReadFcn',readfcn);

% Train network
net=trainNetwork(fds,layers,options);
```

See DEERNet and PyRuv8 papers in *Science Advances*, the code is on GitHub (MIT License).