Introduction to Neural Networks 2019

Sudhakaran Prabakaran
IISER Pune & University of Cambridge

How many are here to become this?

Source: Andrew Ng's book

Outline of the Lectures

Lecture 1: Long introduction to Neural Networks
- Why machine learning?
- Examples in biology
- Introduction to perceptron
- Different neural network architectures
- Why biology?

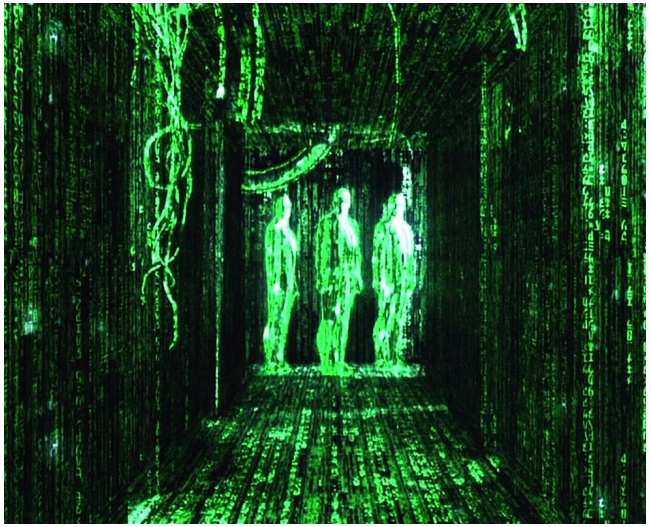
Lecture 2: Introduction to CARET package in R & simple ANN models

Lecture 3: Introduction to Deep Learning

Lecture 4: Examples of Deep Learning models & Brief introduction to Generative Adversarial Networks

For many of us Neural Networks means….

Self-driving cars

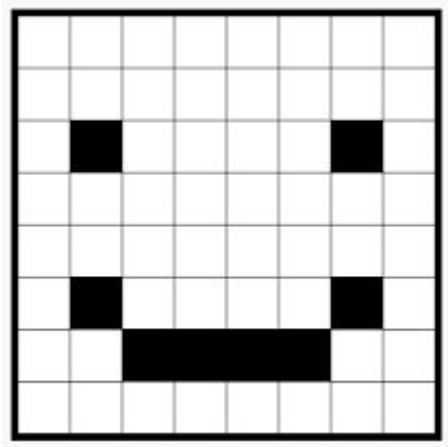


Or the MATRIX!

But real machines are still struggling with this problem

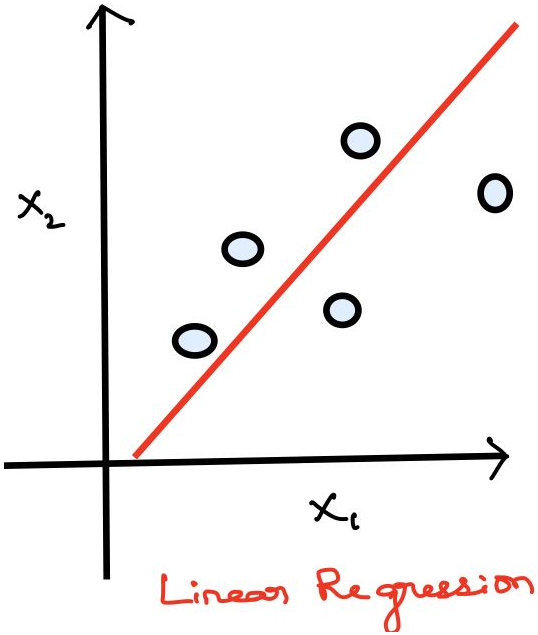
New York Times reported in **1958** that the invention was the beginning of a computer that would "**be able to walk, talk, see, write, reproduce itself and be conscious of its existence.**"

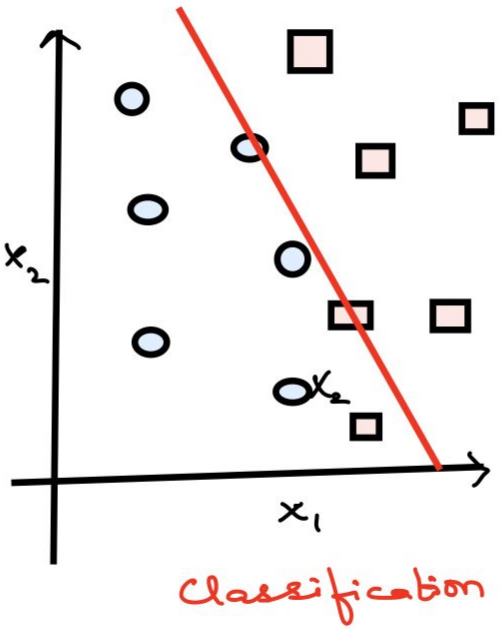
Behind the hood

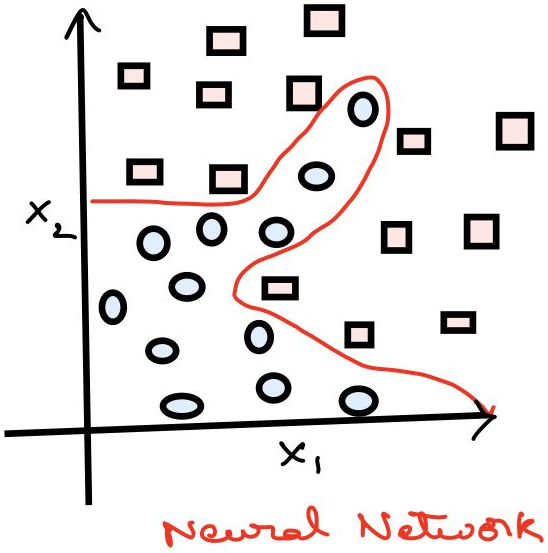


chihuahua

muffin







What is machine learning?

Mathematical hypothesis based on data and use the hypothesis (models) to predict results on new data

Types:

1. Supervised
2. Unsupervised
3. Semi-supervised
4. Reinforcement learning

Common models used in machine learning

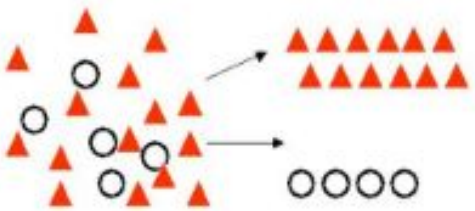
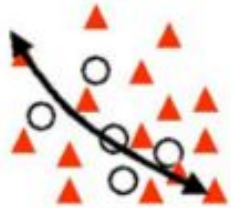
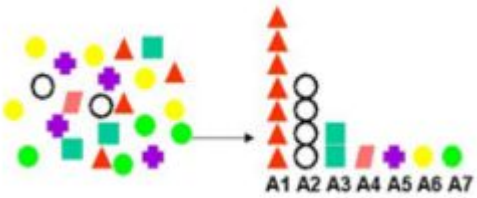
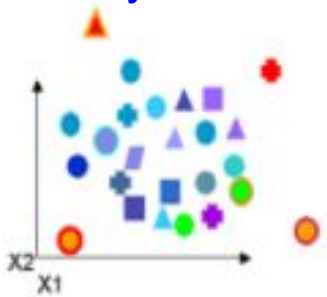
Techniques	Applications
Classification 	Response/ no response yes/ no
Regression 	Continuous numerical outcome
Attribute Ranking 	Finding importance based on strength of relationship with target attribute

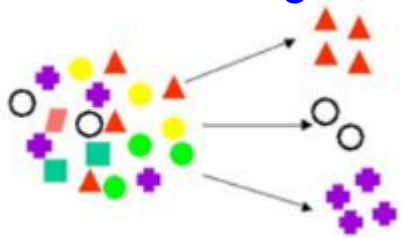
Image source: Futuretext

Anomaly detection



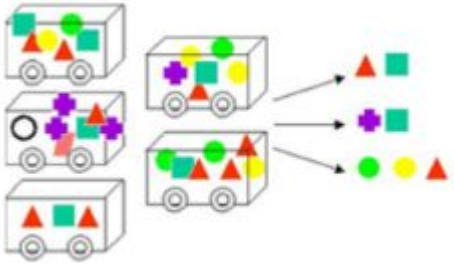
Identifies unusual cases

Clustering



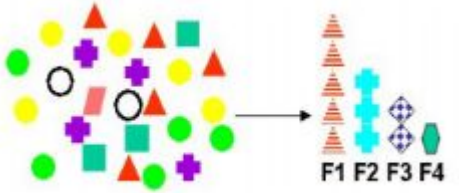
Finding natural groupings

Association



Finds rules associated with naturally co occurring terms

Feature selection



Produces new attributes based on a linear combination of existing attributes

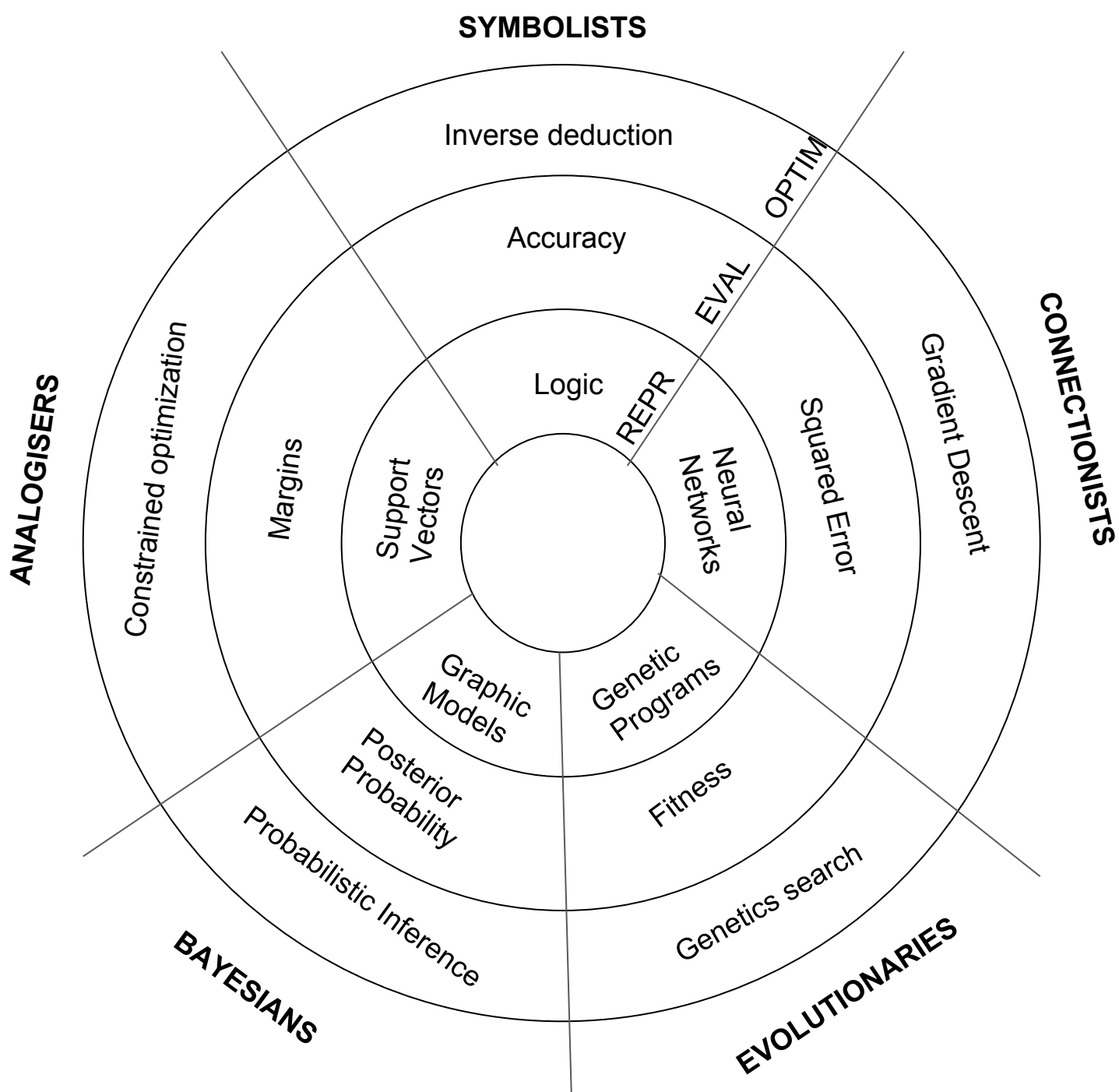
Image source: Futuretext

Different ML models

Dimensionality Reduction
Clustering

Nearest Neighbours
SVM
Random Forest & Decision Trees

Linear Model & Matrix Algebra
Linear & Nonlinear Regression
Artificial Neural Networks
Deep learning


SYMBOLISTS
CONNECTIONISTS
EVOLUTIONARIES
BAYESIANS
ANALOGISERS
Inverse deduction
Accuracy
OPTIM
EVAL
REPR
Logic
Gradient Descent
Squared Error
Neural Networks
Genetic Programs
Fitness
Genetics search
Graphic Models
Posterior Probability
Probabilistic Inference
Support Vectors
Margins
Constrained optimization

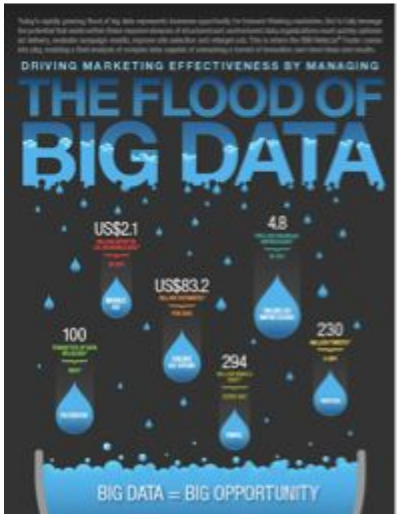
Machine Learning: why now

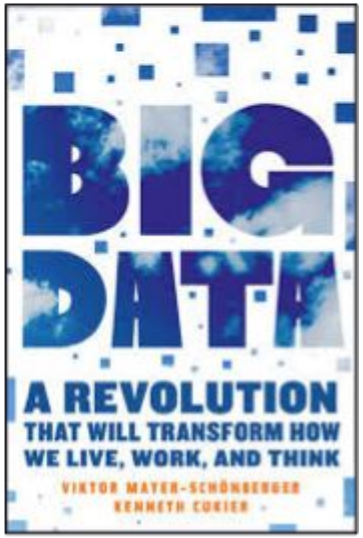
Part II

Big data world

IBM estimates that 90% of world's data has been created in the last two years

Commercial World Data: Financial & Retail Data





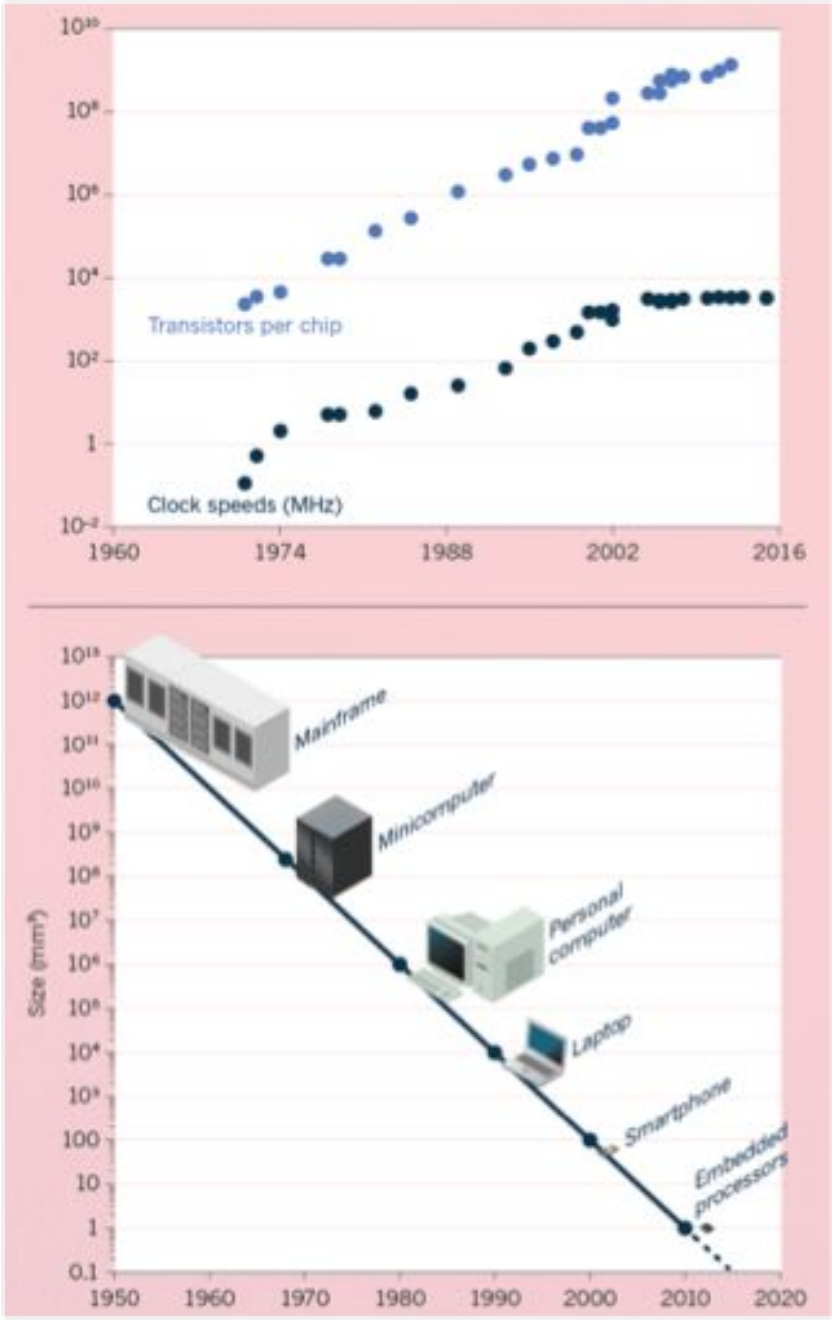
*

Human brain	2.5 PB
Spotify	10 PB
Ebay	90 PB
Facebook	300 PB
Google	15000 PB

***Royal Society website**

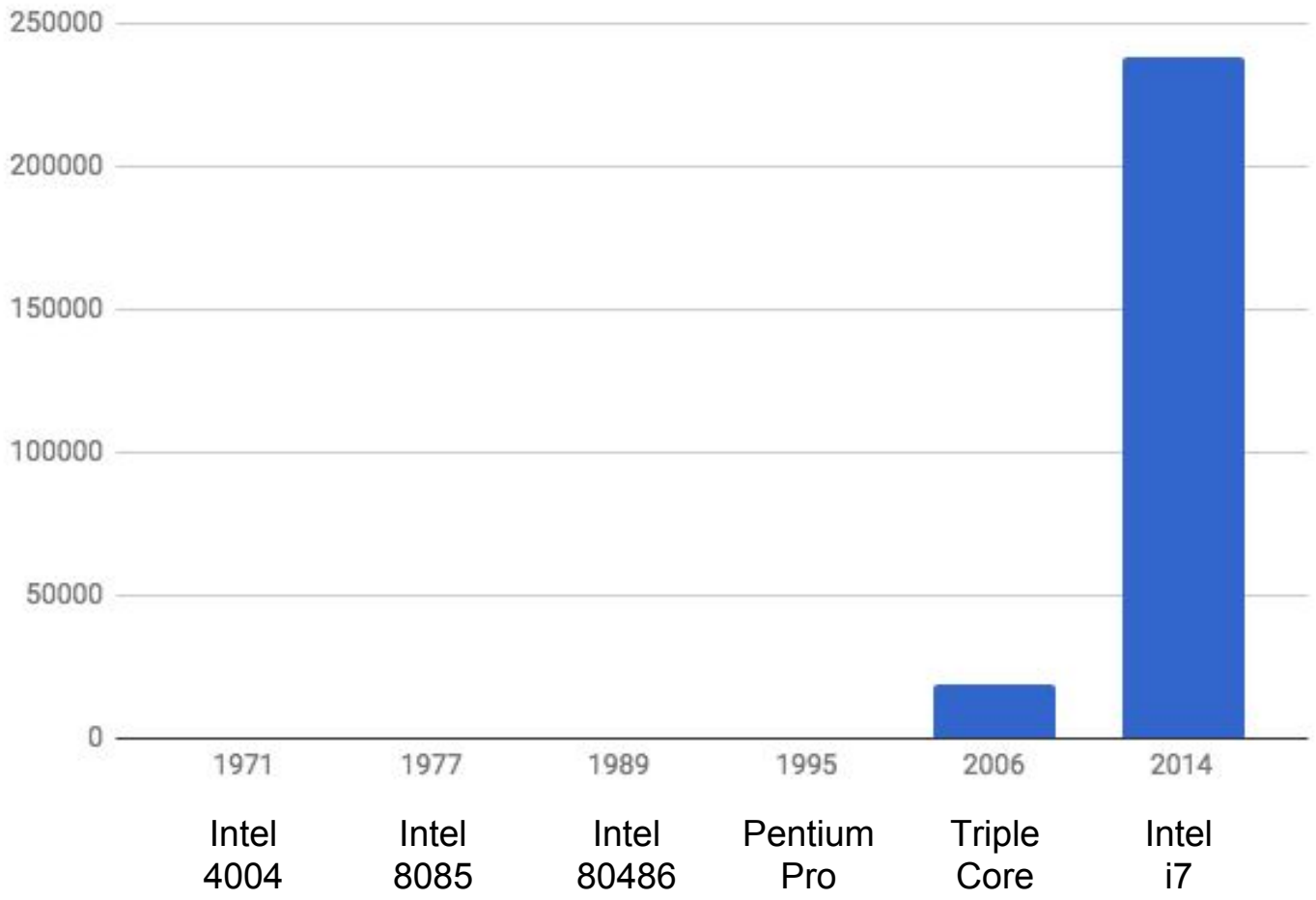
Moore's Law: Exponential Scaling of Computer Technology

- Exponential increase in the number of transistors per chip.
- Led to improvements in speed and miniaturization.
- Drove widespread adoption and novel applications of computer technology.

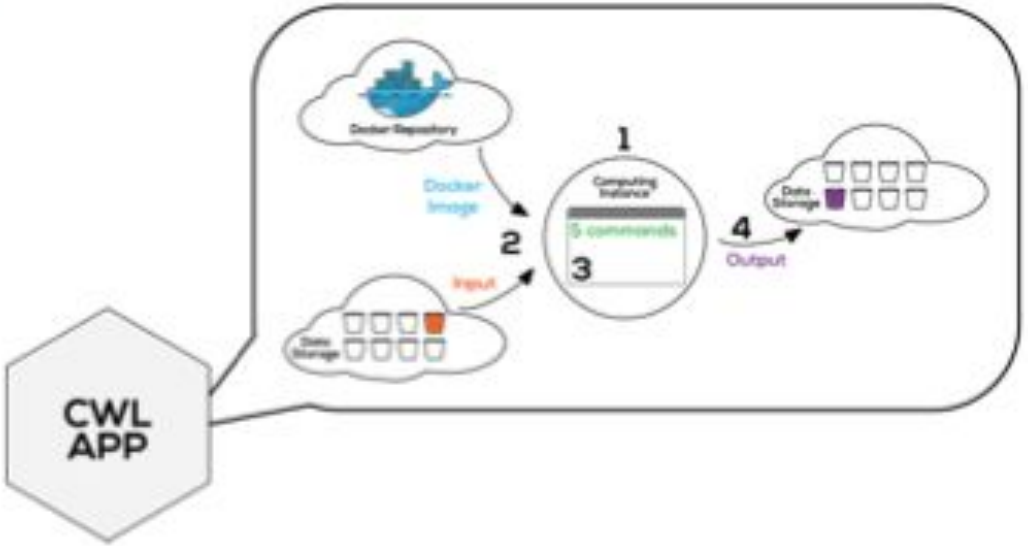


[Waldrop ('15) Nature]

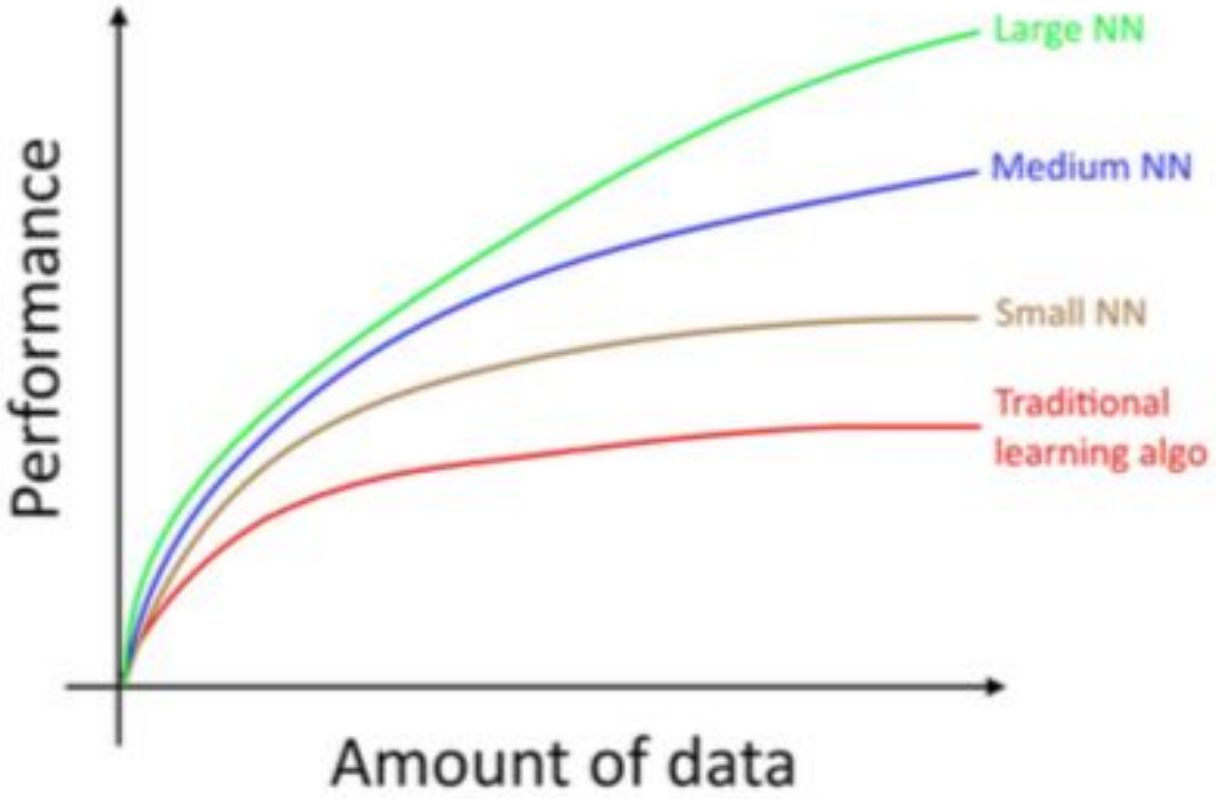
Processing speeds



Cloud computing

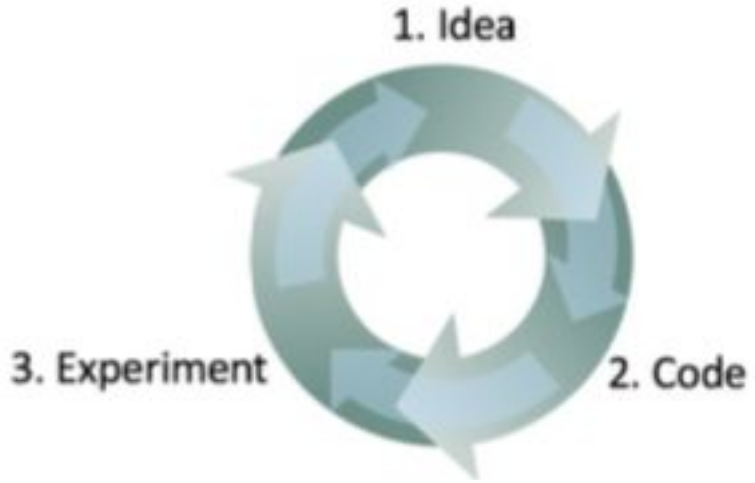


As a result!



Source: Andrew Ng's book

Typical ML algorithm development



Dev/Test set & Metrics

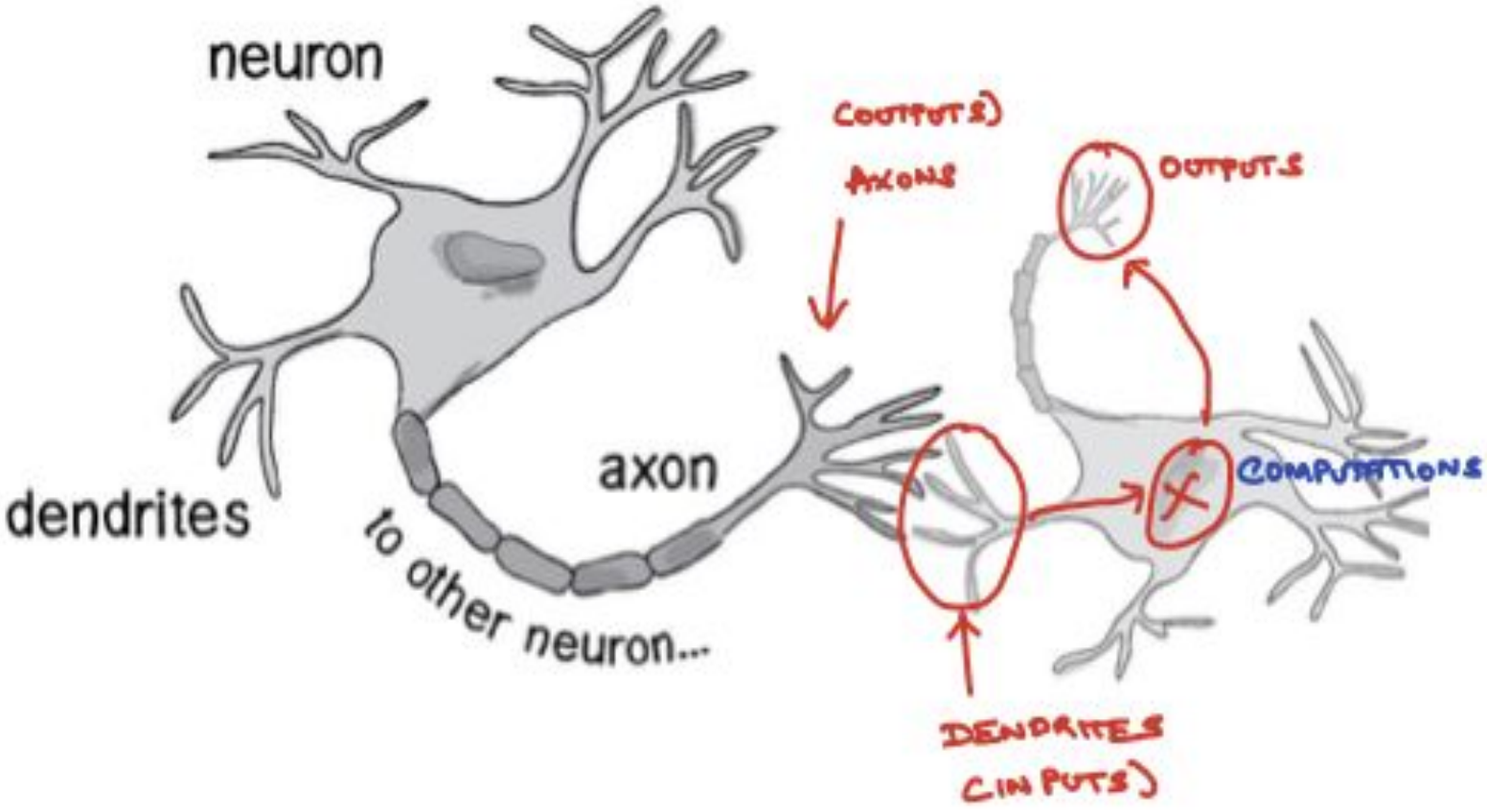
Keep in mind:

1. The distributions should be identical
2. Overfitting dev set
3. The metric is measuring something else
4. Establish metric and dev/test set soon

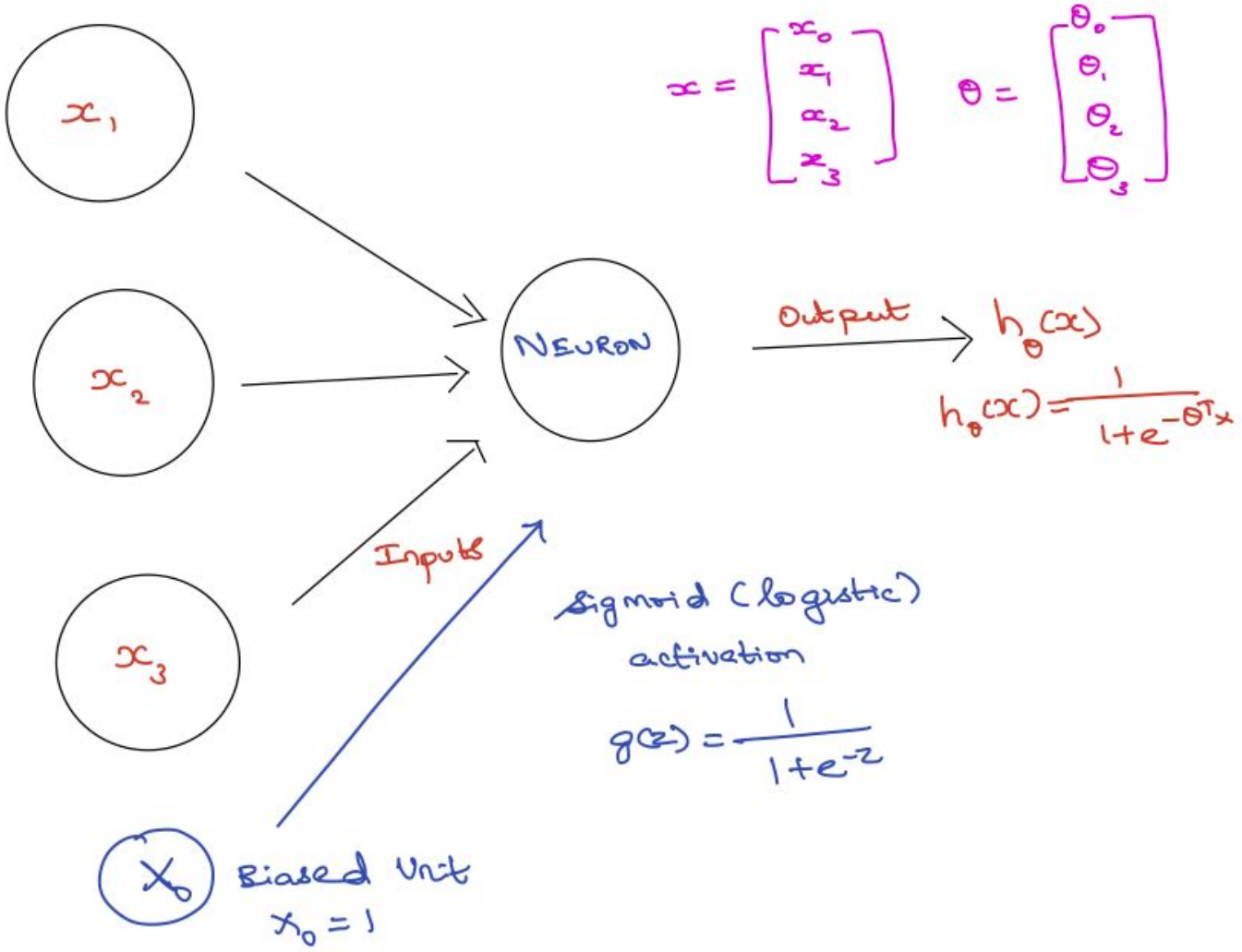
Artificial Neural Networks

Part IV

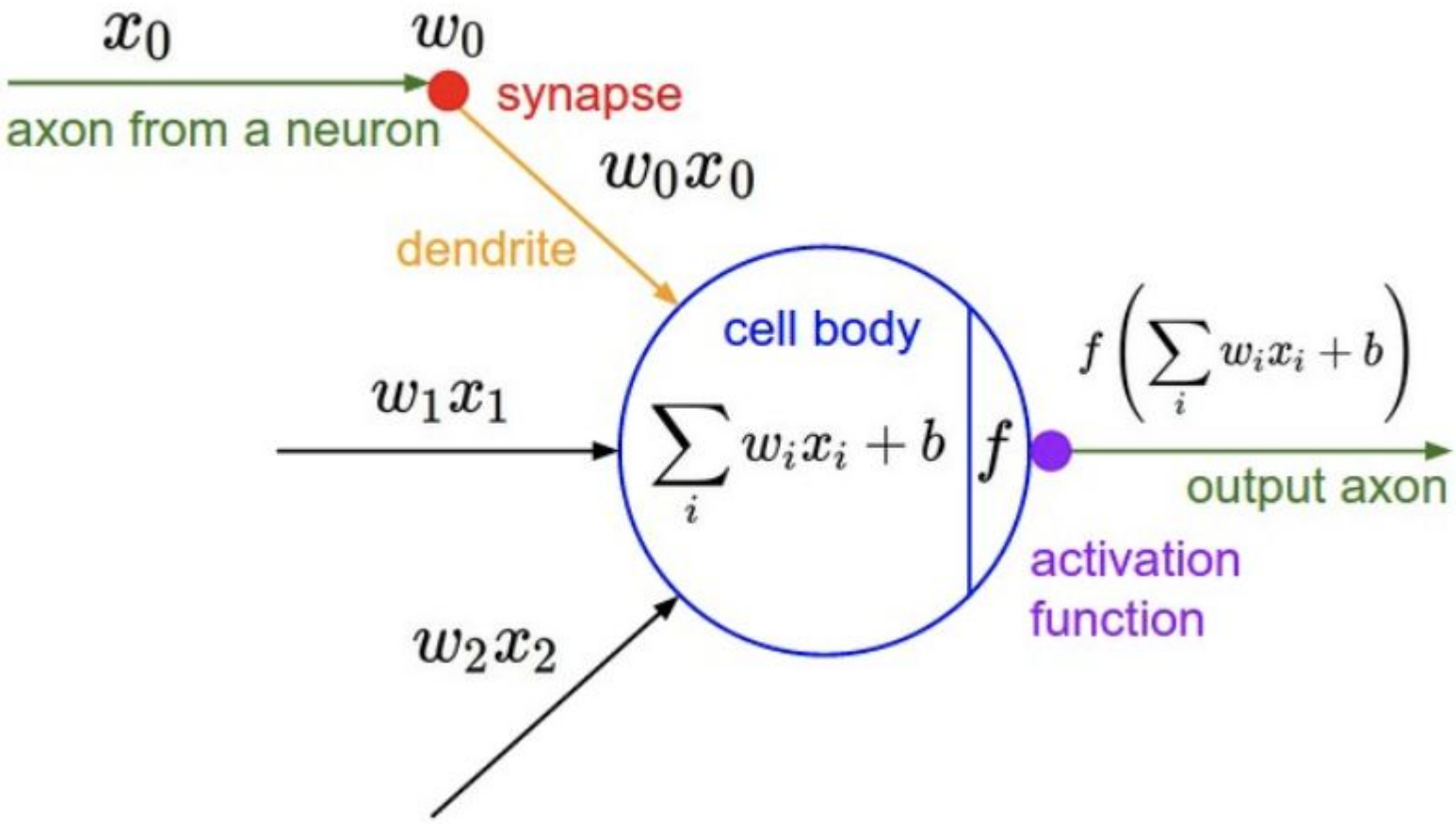
Artificial Neural Networks



Perceptron Model

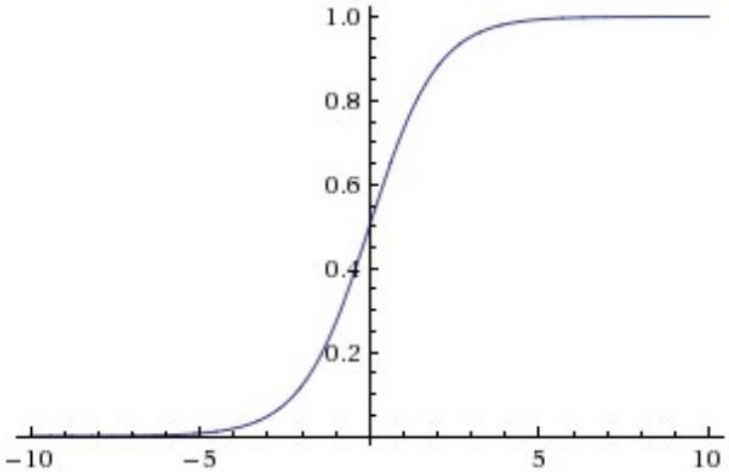


Artificial Neural Network

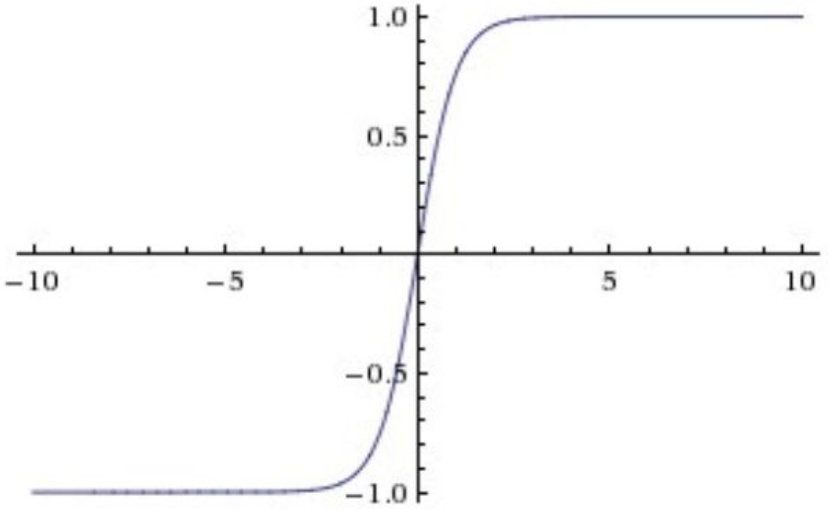


Activation functions

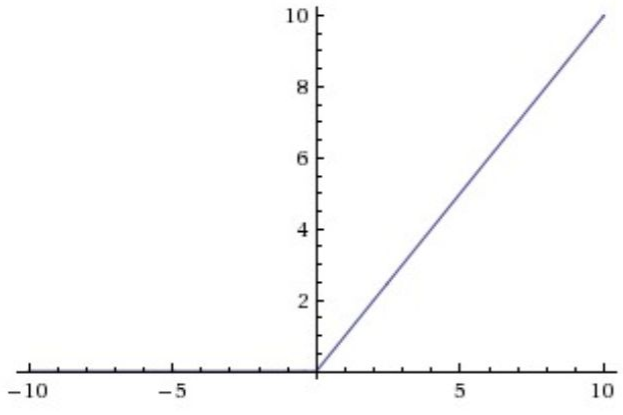
Sigmoid

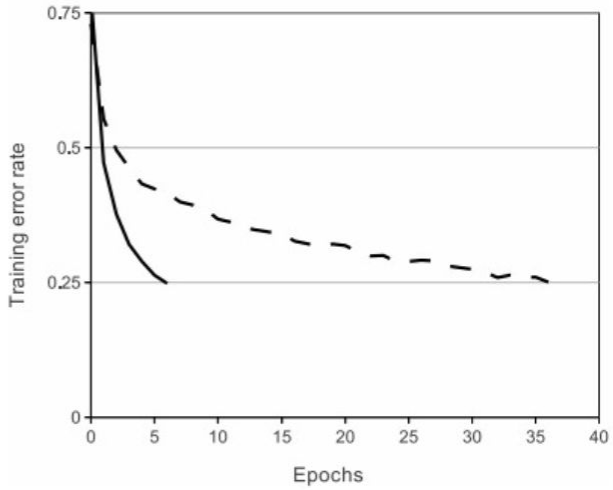


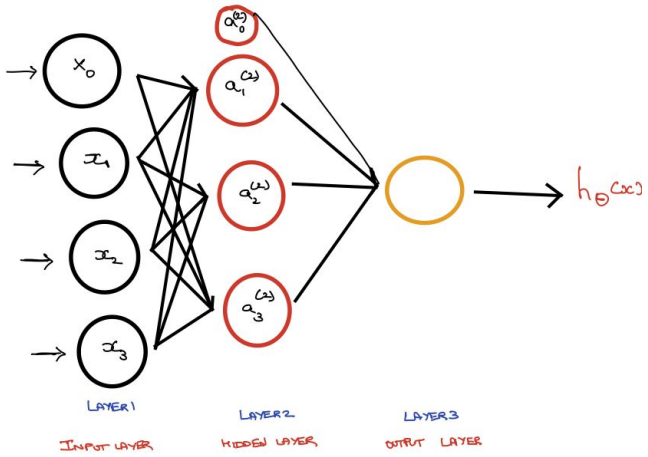
tanh



Rectified Linear Unit (ReLU)







where

$a_i^{(j)}$ = activation of i in layer j

θ^i = matrix of weights controlling function mapping from layer j to layer j+1

$$
\begin{aligned}
a_1^{(2)} &= g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3) \\
a_2^{(2)} &= g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3) \\
a_3^{(2)} &= g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3) \\
h_\theta(x) = a_1^{(3)} &= g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(1)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})
\end{aligned}
$$

Vectorized notations of inputs and activations.

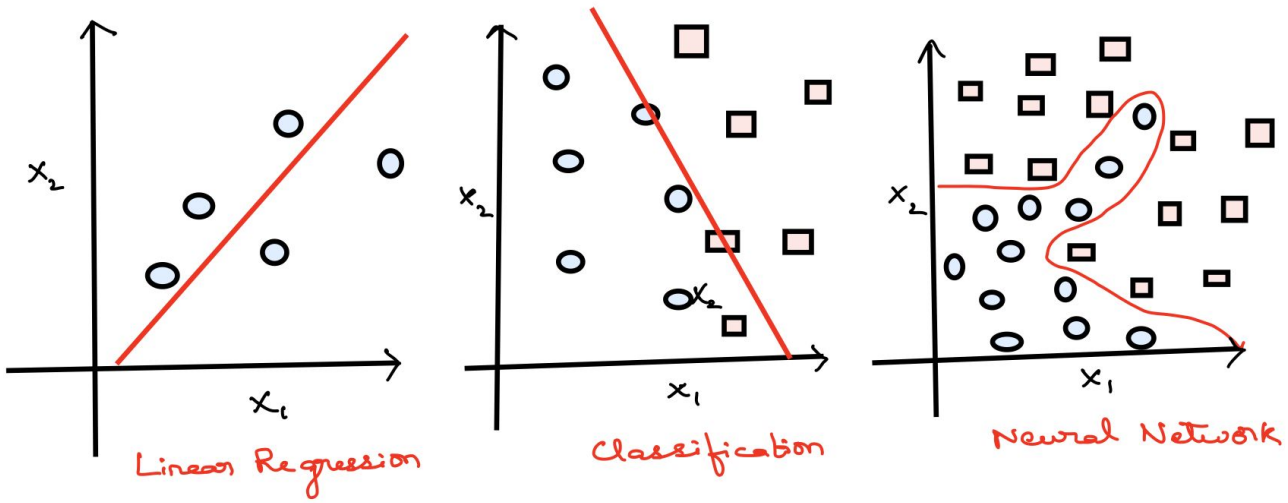
$$
x = \begin{bmatrix} x_o \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}
$$

$$
z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}
$$

Vectorized representation of activation of hidden layer and activation layer.

$$
\begin{aligned}
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)})
\end{aligned}
$$

$$
\begin{aligned}
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
h_\Theta(x) &= a^{(3)} = g(z^{(3)})
\end{aligned}
$$



Whereas for the third we could probably apply a logistic regression with a lot of nonlinear features like this

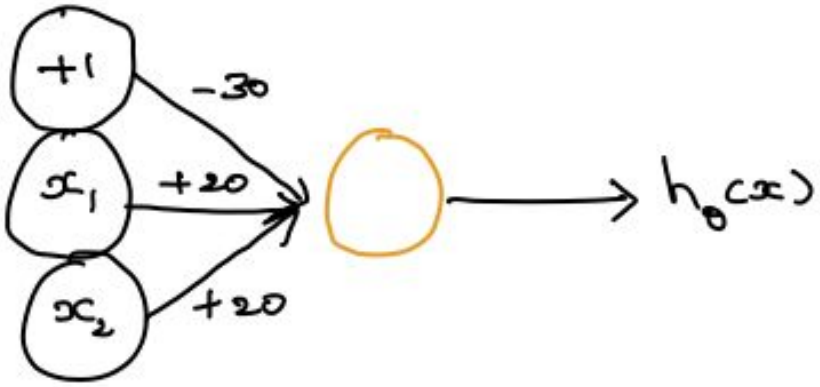
$$Y_i = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_i^2 x_2 + \theta_5 x_i^3 x_2 + \theta_6 x_i^3 x_2^2 \ldots)$$

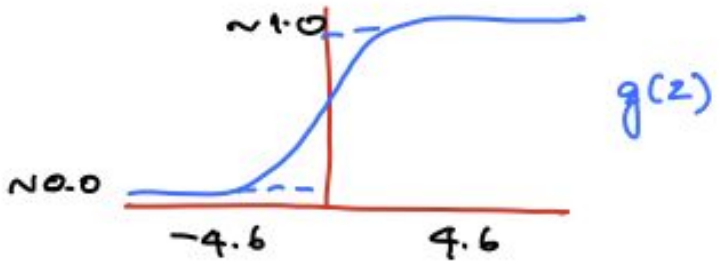
i.e. if we include enough polynomials we could arrive at an hypothesis that will separate the two classes.

This could perfectly work well if we just have two features, such as x_1 and x_2 but for almost all machine learning problems we usually have more than two features. Importantly if the number of features increase the number of quadratic terms increase as a function of $n^2/2$; where n is the number of features.

This would result in overfitting if the number of features increase.

Because ANN has felixibility to derive complex features from each layer of ptrons, it can be applied to any complex functional relationship and more importantly unlike generalized linear models (GLMs) it is not necessary to prespecify the type of relationship between covariates and response variables as for instance as linear combination. This makes ANN a valuable statistical tool.





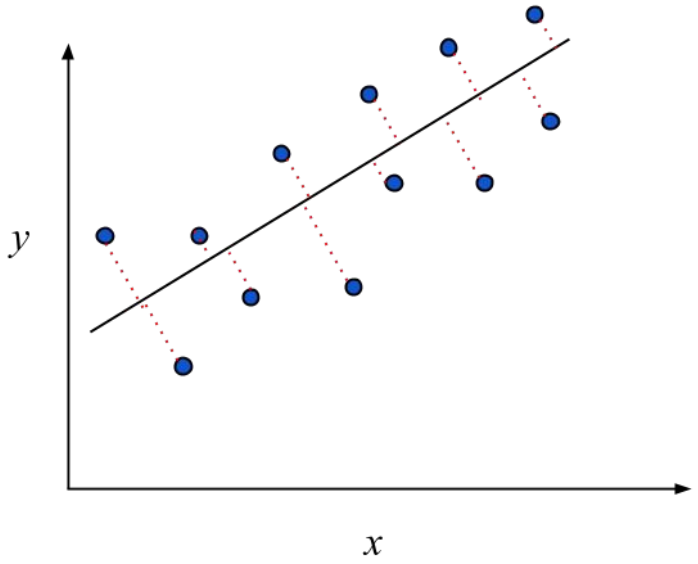
$$h_\theta(x) = g(-30 + 20x_1 + 20x_2)$$

Truth table

x_1	x_2	$h_\theta(x)$
0	0	$g(-30) \approx 0$
0	1	$g(-10) \approx 0$
1	0	$g(-10) \approx 0$
1	1	$g(10) \approx 1$

For linear models

architecture



hypothesis

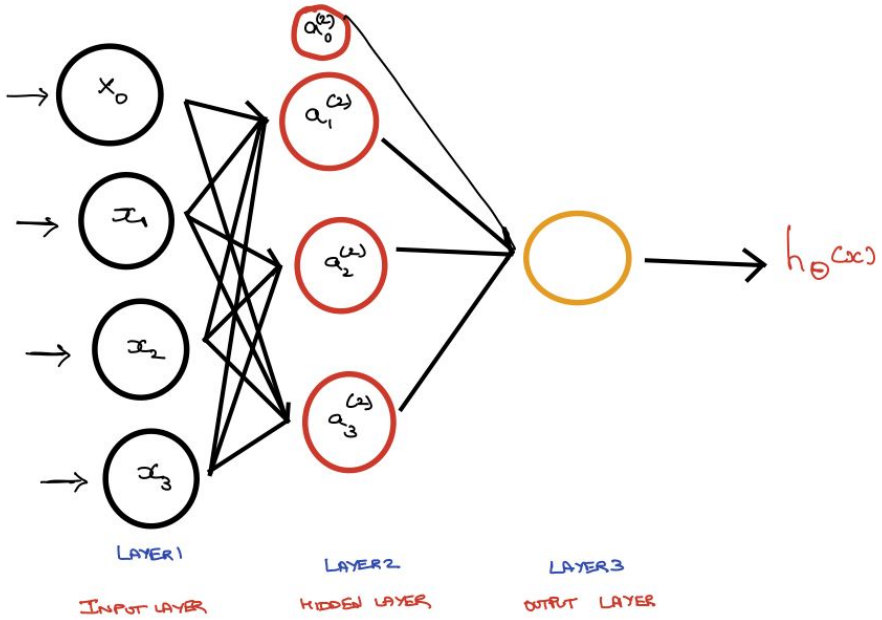
$$h_{\Theta}(x) = \Theta_1 x + \Theta_0$$

Cost function = $$1/2n \sum_{i=1}^{n} (h_\theta(x^i) - y^i)^2$$

Machine learning algorithm tries to **minimise** this Cost Function

For neural network models

architecture



hypothesis

$$a_1^{(2)} = g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3)$$
$$a_2^{(2)} = g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3)$$
$$a_3^{(2)} = g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3)$$
$$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(1)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

Cost function =

$$CF(\Theta) = -1/m \left[\sum_{i=1}^{m} \sum_{k=1}^{K} y_k^{(i)} log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) log(1 - (h_\Theta(x^{(i)}))_k) \right]$$

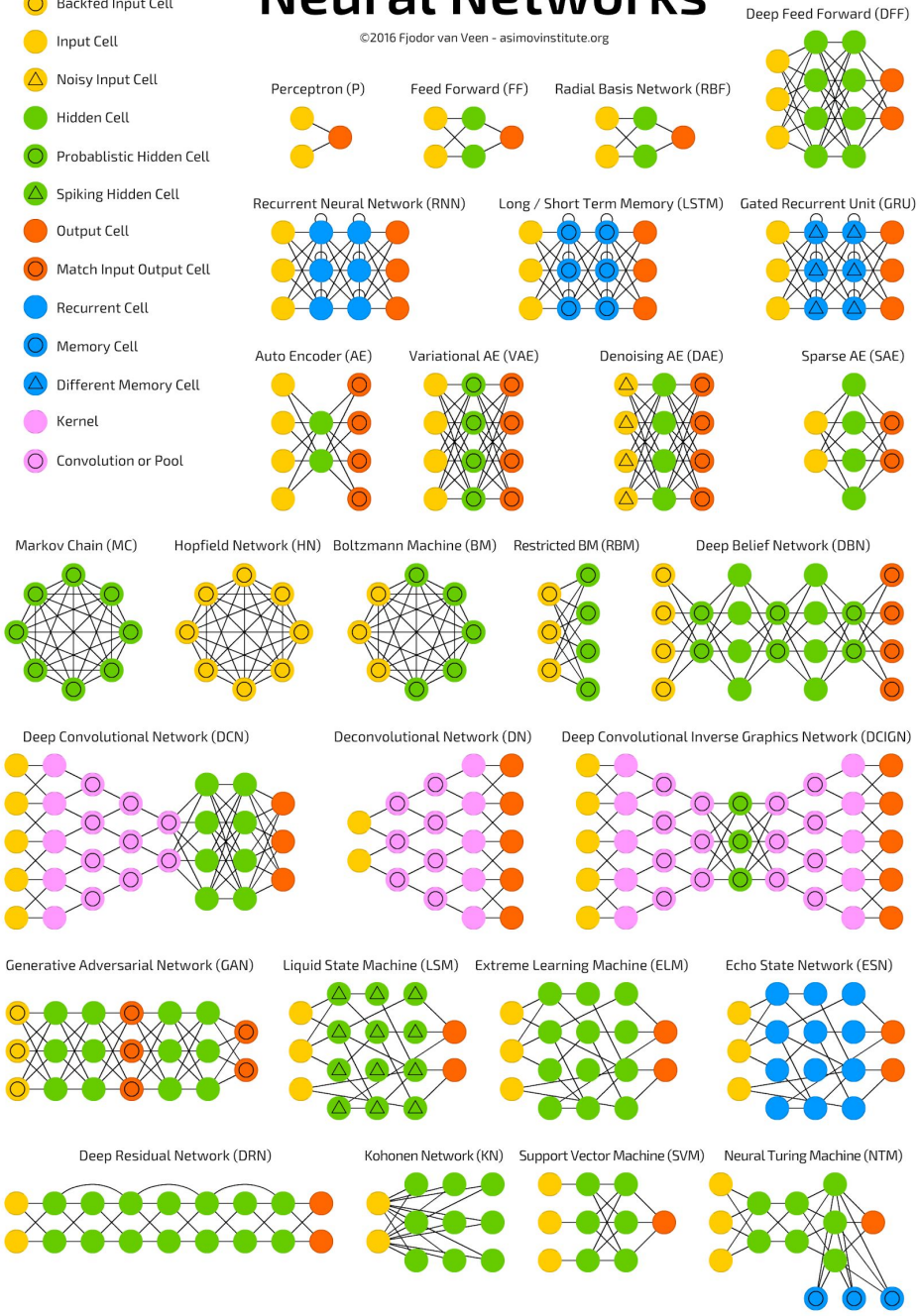
$$+\lambda/2m \sum_{l=1}^{L-1} \sum_{i=1}^{s_l} \sum_{j=1}^{S_{l+1}} (\Theta_{ji}^{(l)})^2$$

A mostly complete chart of

Neural Networks

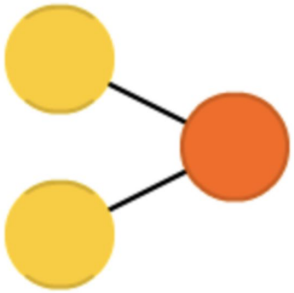
©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- Noisy Input Cell
- Hidden Cell
- Probablistic Hidden Cell
- Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- Different Memory Cell
- Kernel
- Convolution or Pool



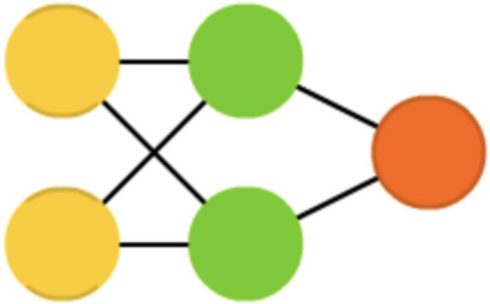
Neural network architectures

Perceptron (P)



Perceptron. The simplest and oldest model of Neuron, as we know it. Takes some inputs, sums them up, applies activation function and passes them to output layer. No magic here.

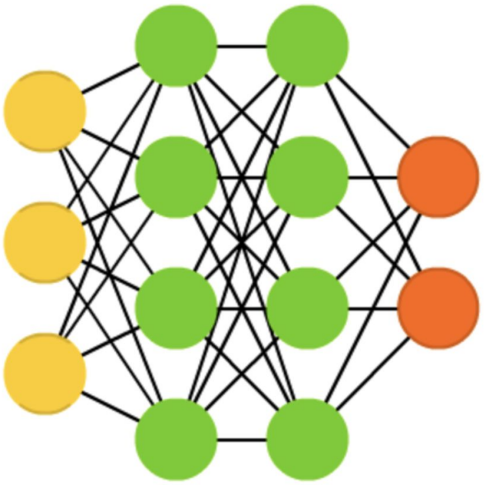
Feed Forward (FF)



1. all nodes are fully connected
2. activation flows from input layer to output, without back loops
3. there is one layer between input and output (hidden layer)
4. In most cases this type of networks is trained using Backpropagation method.

Neural network architectures

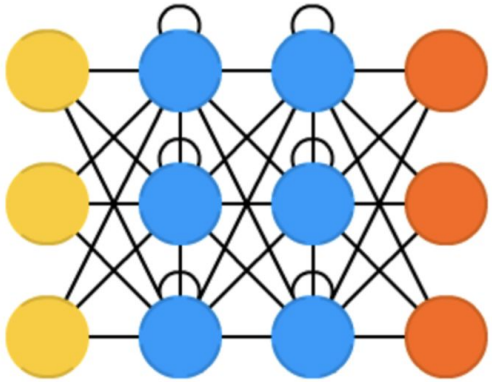
Deep Feed Forward (DFF)



DFF neural networks opened pandora box of deep learning in early 90s. These are just FF NNs, but with more than one hidden layer. So, what makes them so different?

Because of that stacking more layers led to exponential growth of training times, making DFFs quite impractical. Only in early ‘00s we developed a bunch of approaches that allowed to train DFFs effectively; now they form a core of modern Machine Learning systems, covering the same purposes as FFs, but with much better results.

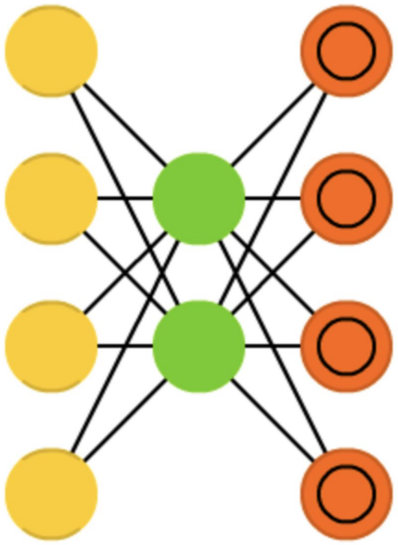
Recurrent Neural Network (RNN)



Recurrent Neural Networks introduce different type of cells—Recurrent cells. The first network of this type was so called Jordan network, **when each of hidden cell received it’s own output with fixed delay**—one or more iterations.

Neural network architectures

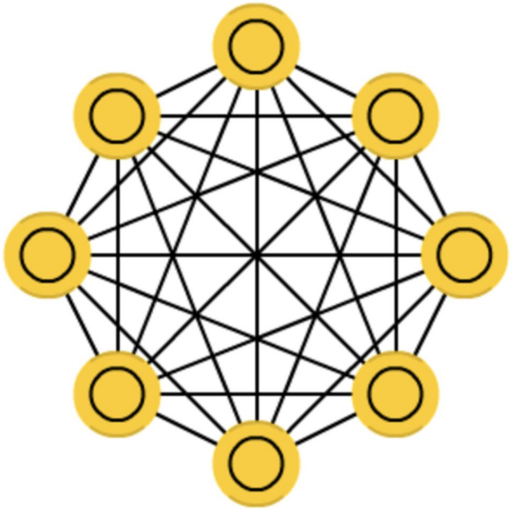
Auto Encoder (AE)



Autoencoders are used for classification, clustering and feature compression.

AEs are trained without supervision. Their structure—when number of hidden cells is smaller than number of input cells (and number of output cells equals number of input cells), and when the AE is trained the way the output is as close to input as possible, forces AEs to generalise data and search for common patterns.

Hopfield Network (HN)

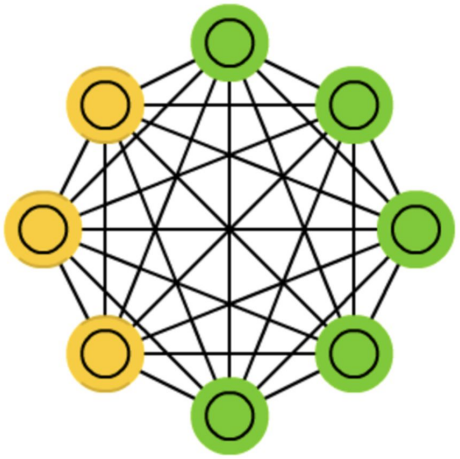


Hopfield networks are trained on a limited set of samples so they respond to a known sample with the same sample.

Each cell serves as input cell before training, as hidden cell during training and as output cell when used.

Neural network architectures

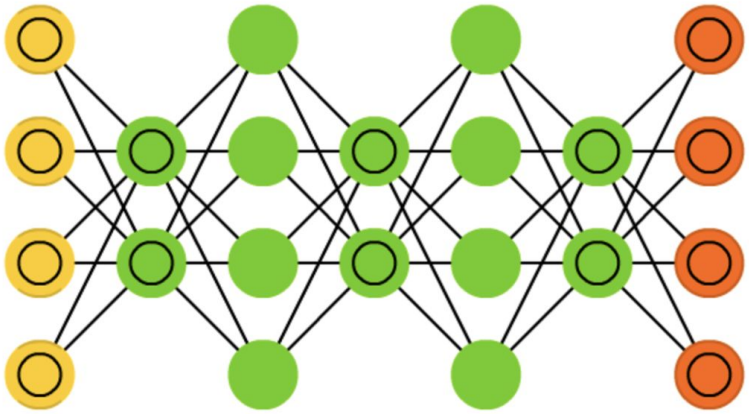
Boltzmann Machine (BM)



Boltzmann machines are very similar to HNs where some cells are marked as input and remain hidden. Input cells become output as soon as each hidden cell update their state (during training, BMs / HNs update cells one by one, and not in parallel).

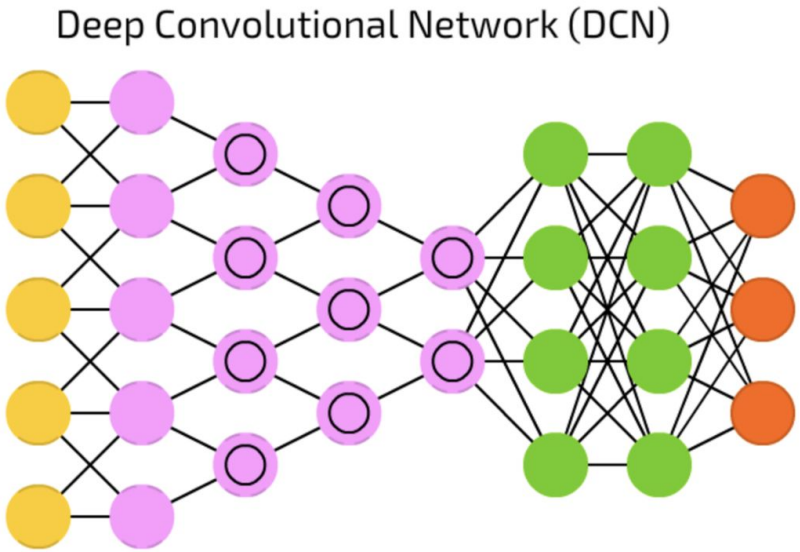
This is the first network topology that was successfully trained using Simulated annealing approach.

Deep Belief Network (DBN)



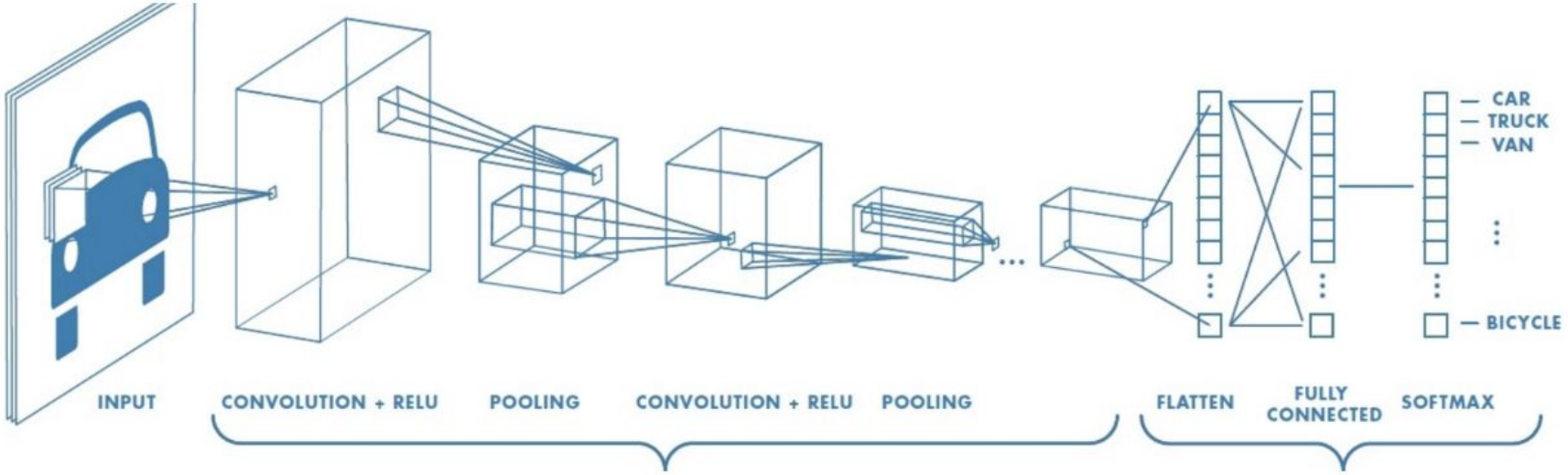
DBNs, mentioned above, are actually a stack of Boltzmann Machines (surrounded by VAEs). They can be chained together (when one NN trains another) and can be used to generate data by already learned pattern.

Neural network architectures



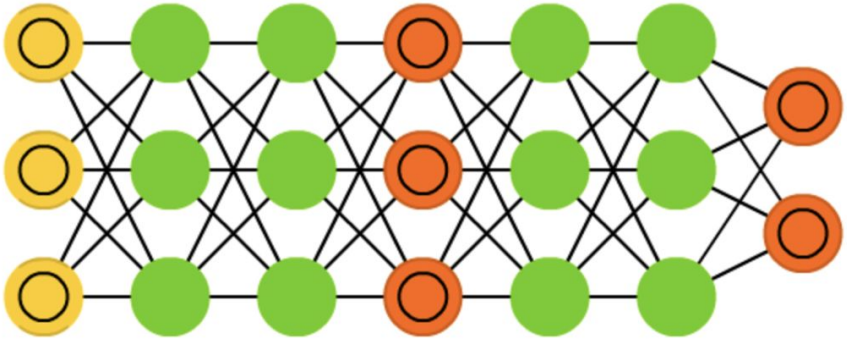
DCN nowadays are stars of artificial neural networks. They feature convolution cells (or pooling layers) and kernels, each serving a different purpose.

Convolution kernels actually process input data, and pooling layers simplify it (mostly using non-linear functions, like max), reducing unnecessary features.



Neural network architectures

Generative Adversarial Network (GAN)



GAN represents a huge family of double networks, that are composed from generator and discriminator. They constantly try to fool each other—**generator tries to generate some data, and discriminator, receiving sample data, tries to tell generated data from samples.** Constantly evolving, this type of neural networks can generate real-life images, in case you are able to maintain the training balance between these two networks.

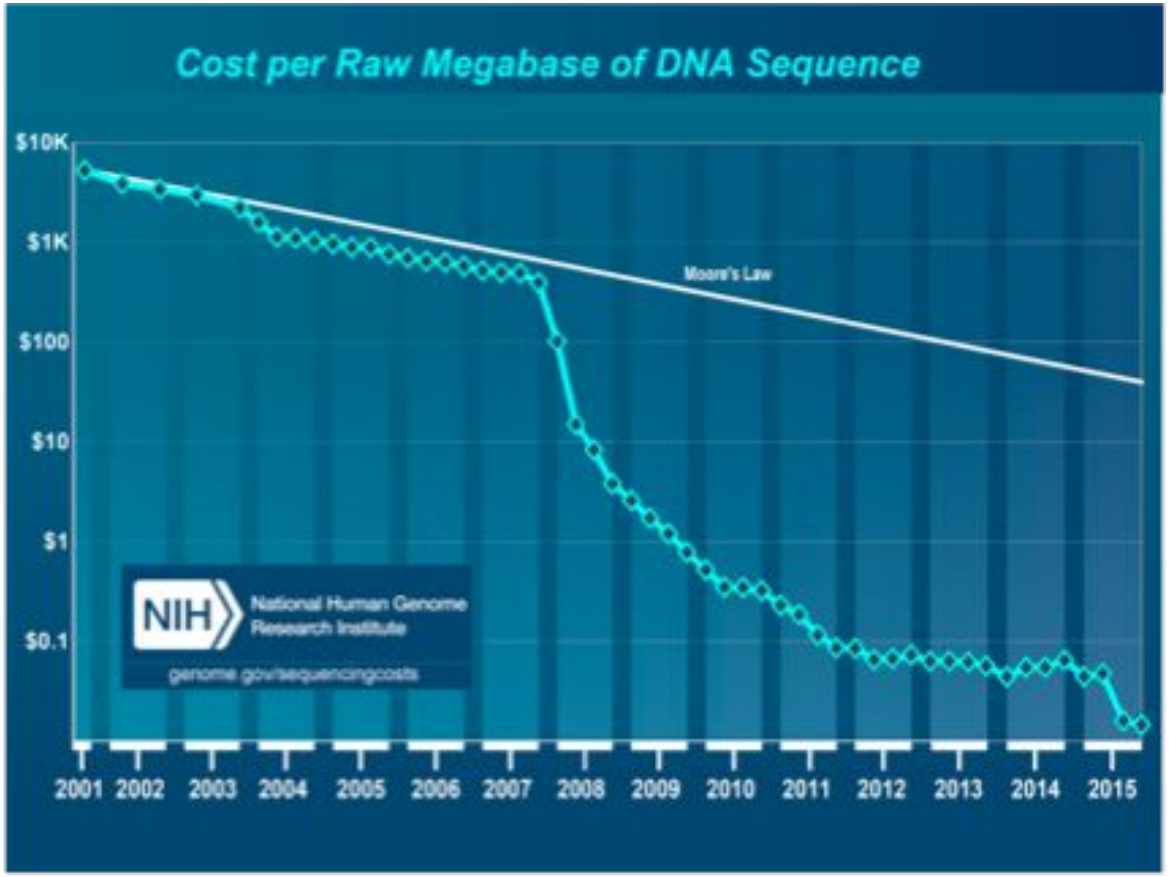
Machine Learning: why biology?

Part IV

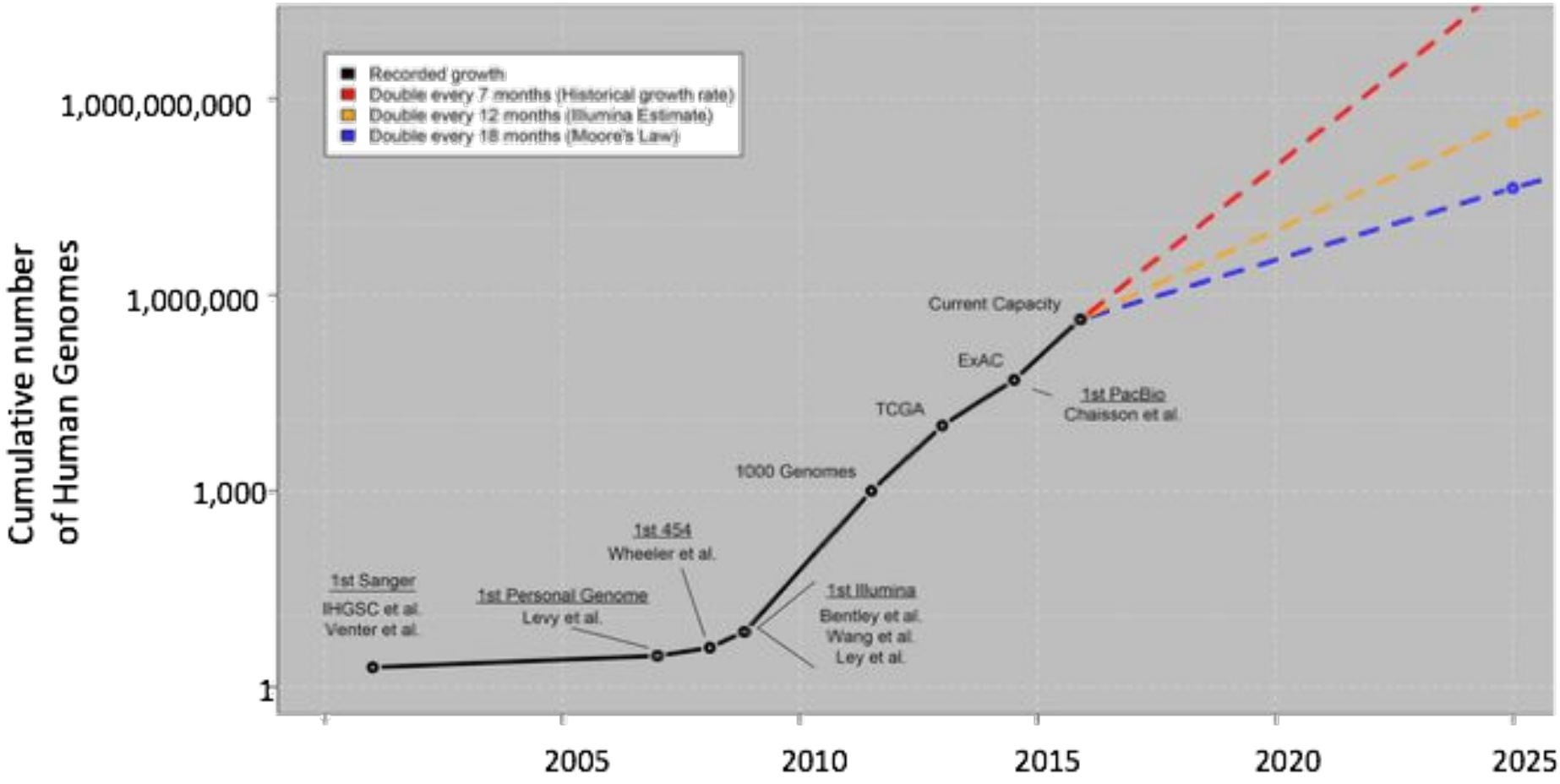
Sequencing data explosion: faster than moore's law over time

DNA sequencing has gone through technological S-curves

- In the early 2000's, improvements in Sanger sequencing produced a scaling pattern similar to Moore's law.
- The advent of NGS was a shift to a new technology with dramatic decrease in cost).

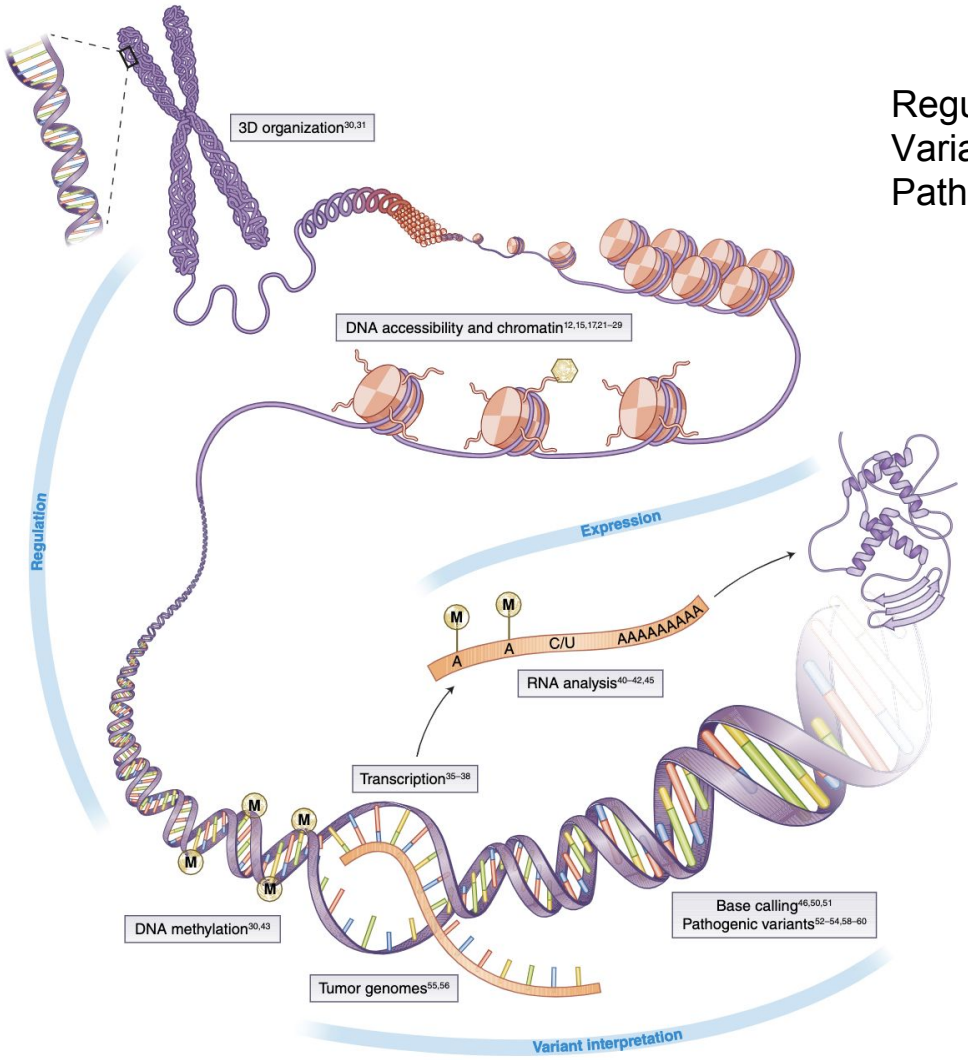


Where are we heading?



By 2019 total NGS data to exceed 2 Exabytes

Applications of Deep learning in genomics

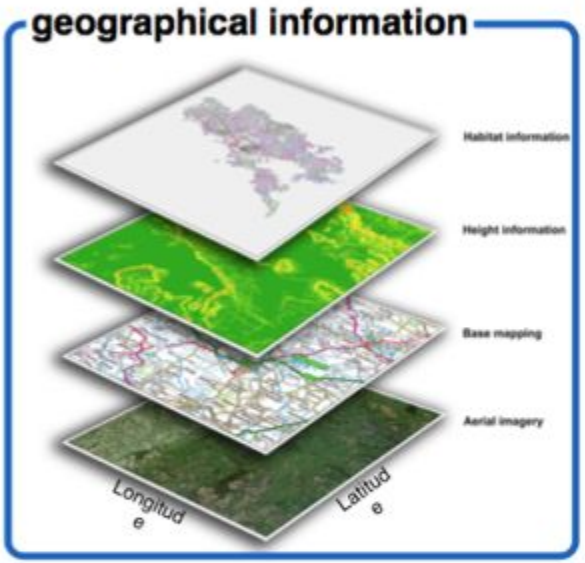


Regulatory genomics,
Variant calling and
Pathogenicity scores.

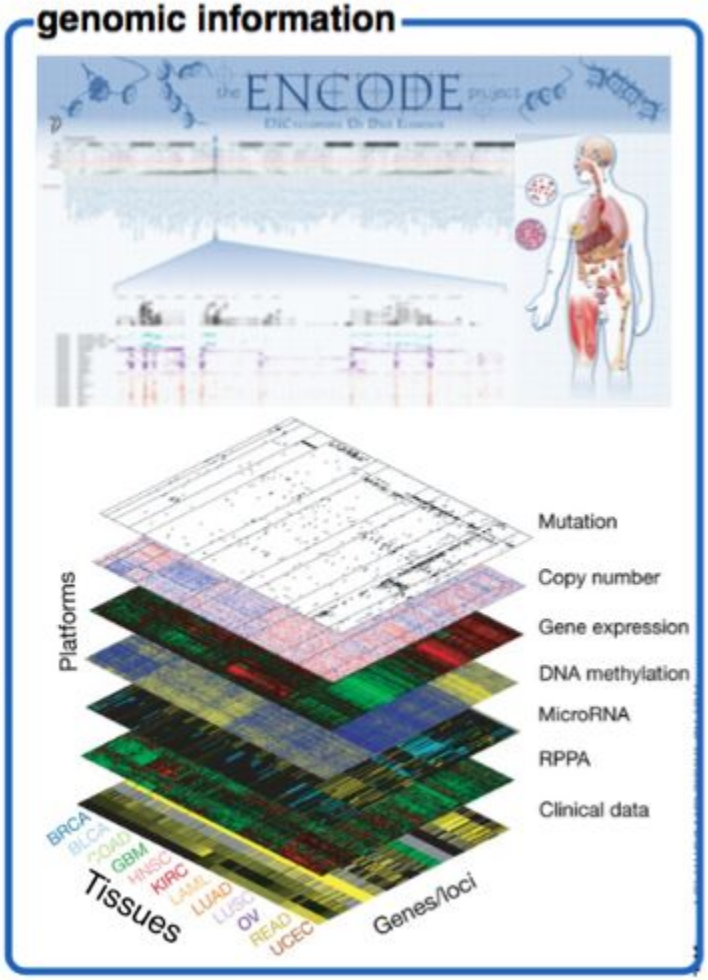
Zou, J. et al., 2018. A primer on deep learning in genomics. *Nature genetics*. Available at: http://dx.doi.org/10.1038/s41588-018-0295-5.

More and more data!

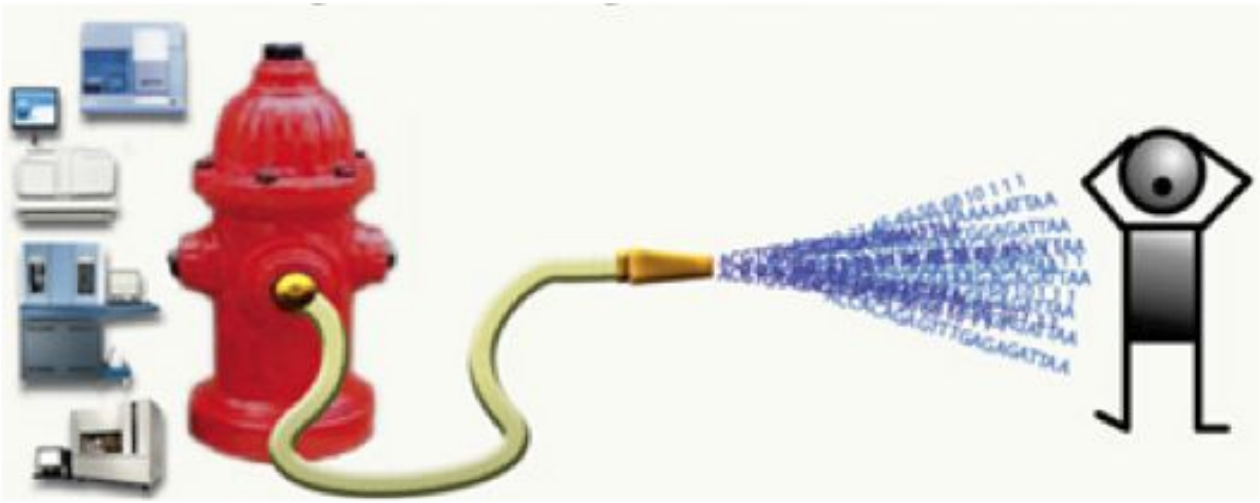
Human genome annotation — a non-intuitive map



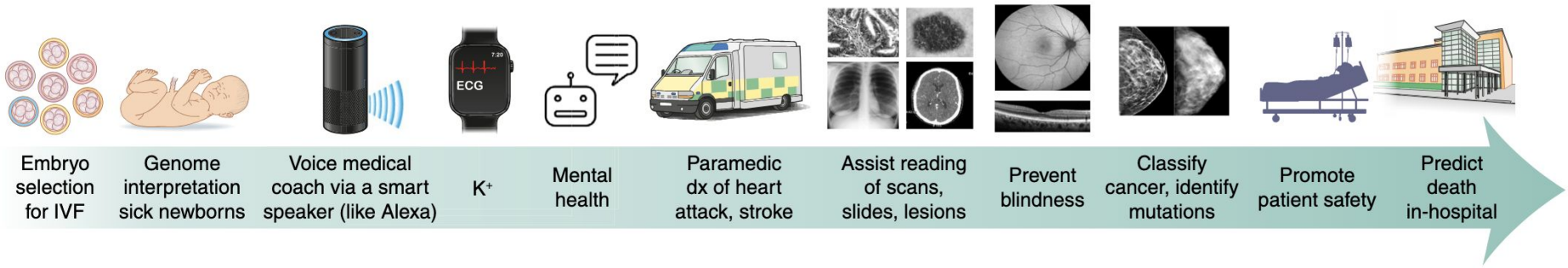
- Large-scale organisation providing an overview of the genome
- Integration of heterogeneous data



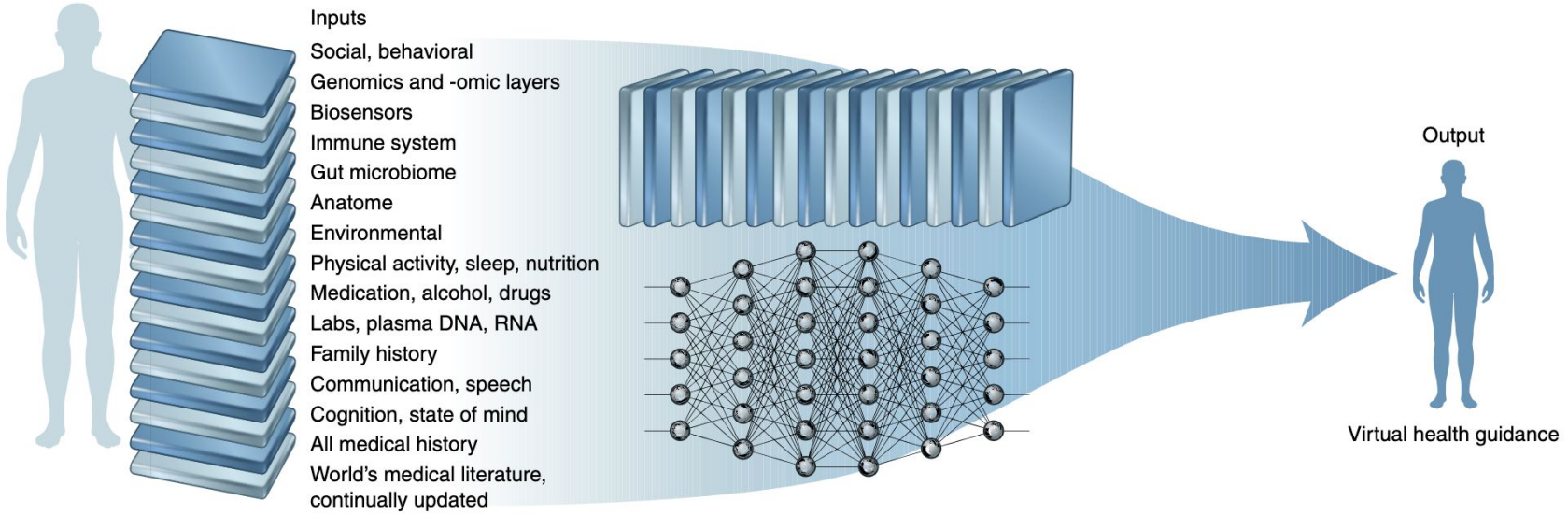
Data overload?



Examples of AI applications across the human life



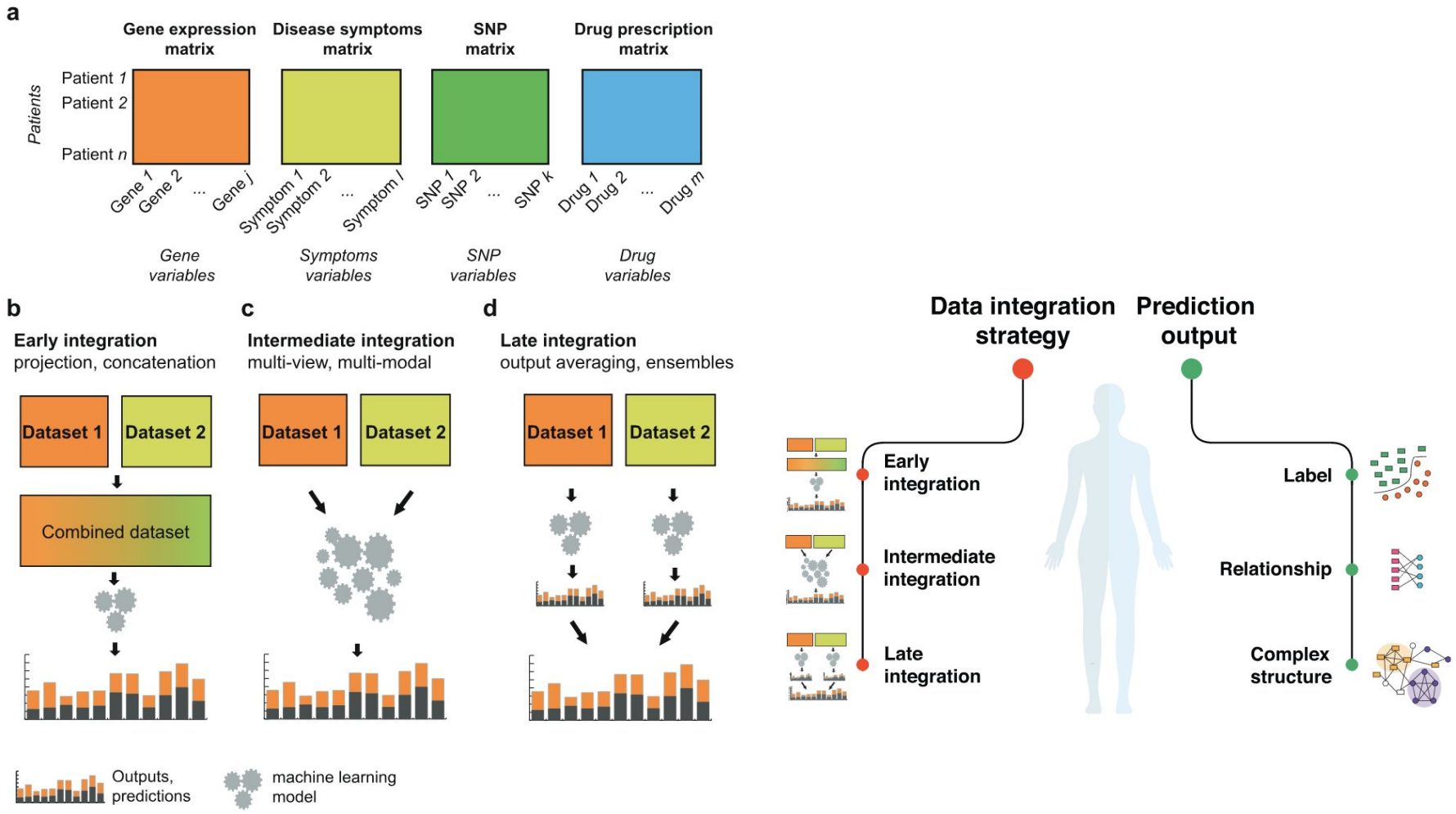
Multi-modal data inputs & algorithms to provide individualized guidance



Topol, E.J., 2019. High-performance medicine: the convergence of human and artificial intelligence. Nature medicine, 25(1), pp.44–56.

Multi-modal data inputs & algorithms to provide individualized guidance

high-dimensional, incomplete, biased, heterogeneous, dynamic, and noisy.



Zitnik, M. et al., 2019. Machine Learning for Integrating Data in Biology and Medicine: Principles, Practice, and Opportunities. An international journal on information fusion, 50, pp.71–91.

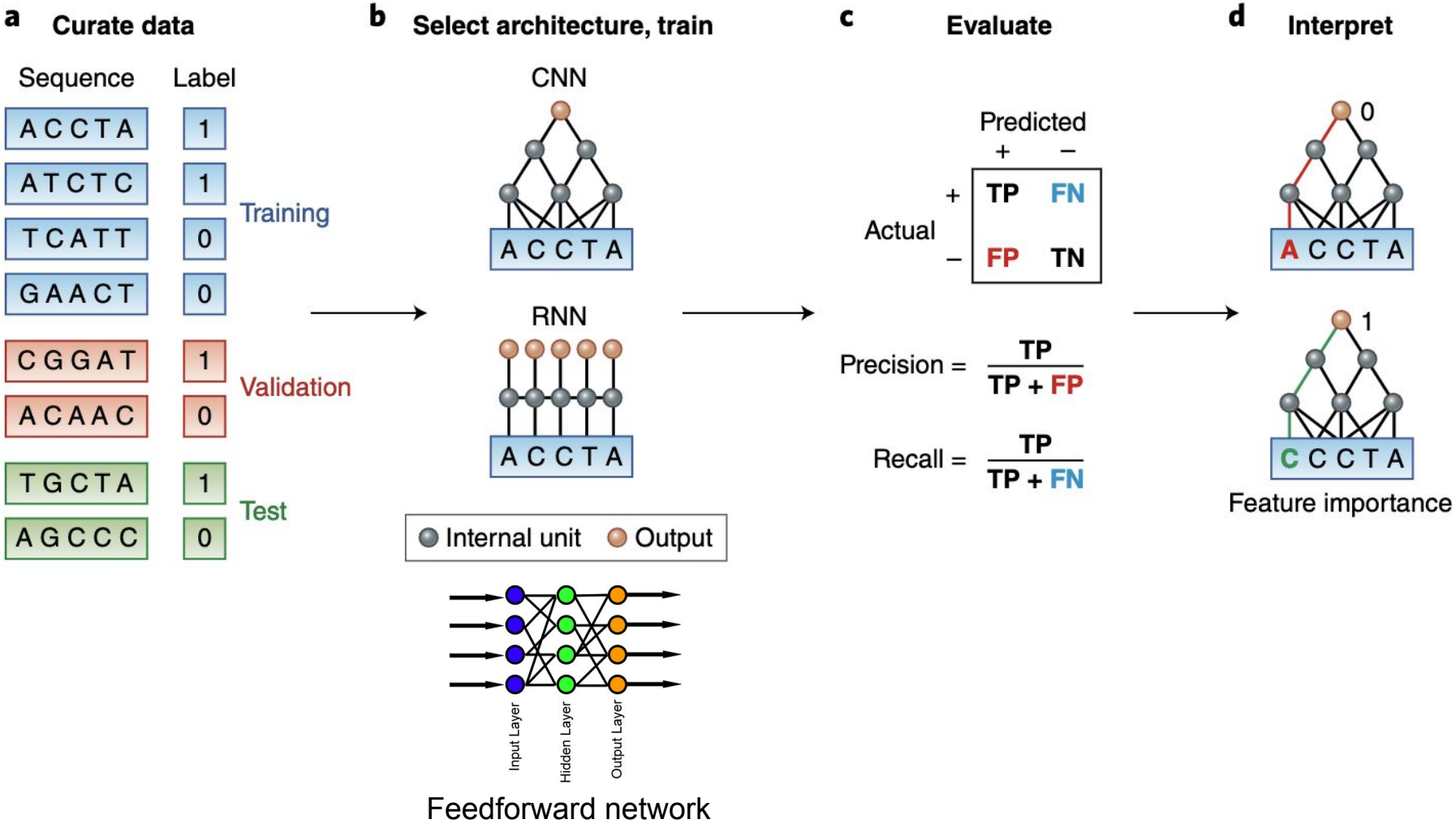
Current and potential AI applications in medicine

Basic biomedical research	Translational research	Clinical practice
Automated experiments	Biomarker discovery	Disease diagnosis
Automated data collection	Drug–target prioritization	Interpretation of patient genomes
Gene function annotation	Drug discovery	Treatment selection
Prediction of transcription factor binding sites	Drug repurposing	Automated surgery
Simulation of molecular dynamics	Prediction of chemical toxicity	Patient monitoring
Literature mining	Genetic variant annotation	Patient risk stratification for primary prevention

Yu, K.-H., Beam, A.L. & Kohane, I.S., 2018. Artificial intelligence in healthcare. *Nature Biomedical Engineering*, 2(10), pp.719–731.

Deep learning workflow in genomics

Deep learning is an umbrella term that refers to the recent advances in neural networks and the corresponding training platforms (e.g.,TensorFlow and PyTorch)



Zou, J. et al., 2018. A primer on deep learning in genomics. *Nature genetics*. Available at: http://dx.doi.org/10.1038/s41588-018-0295-5.

Deep learning resources

All the resources, except for those listed in the 'specific for genomics' block, are relevant for deep learning in general. Because the field is developing rapidly, this information is likely to be considerably different in the future.

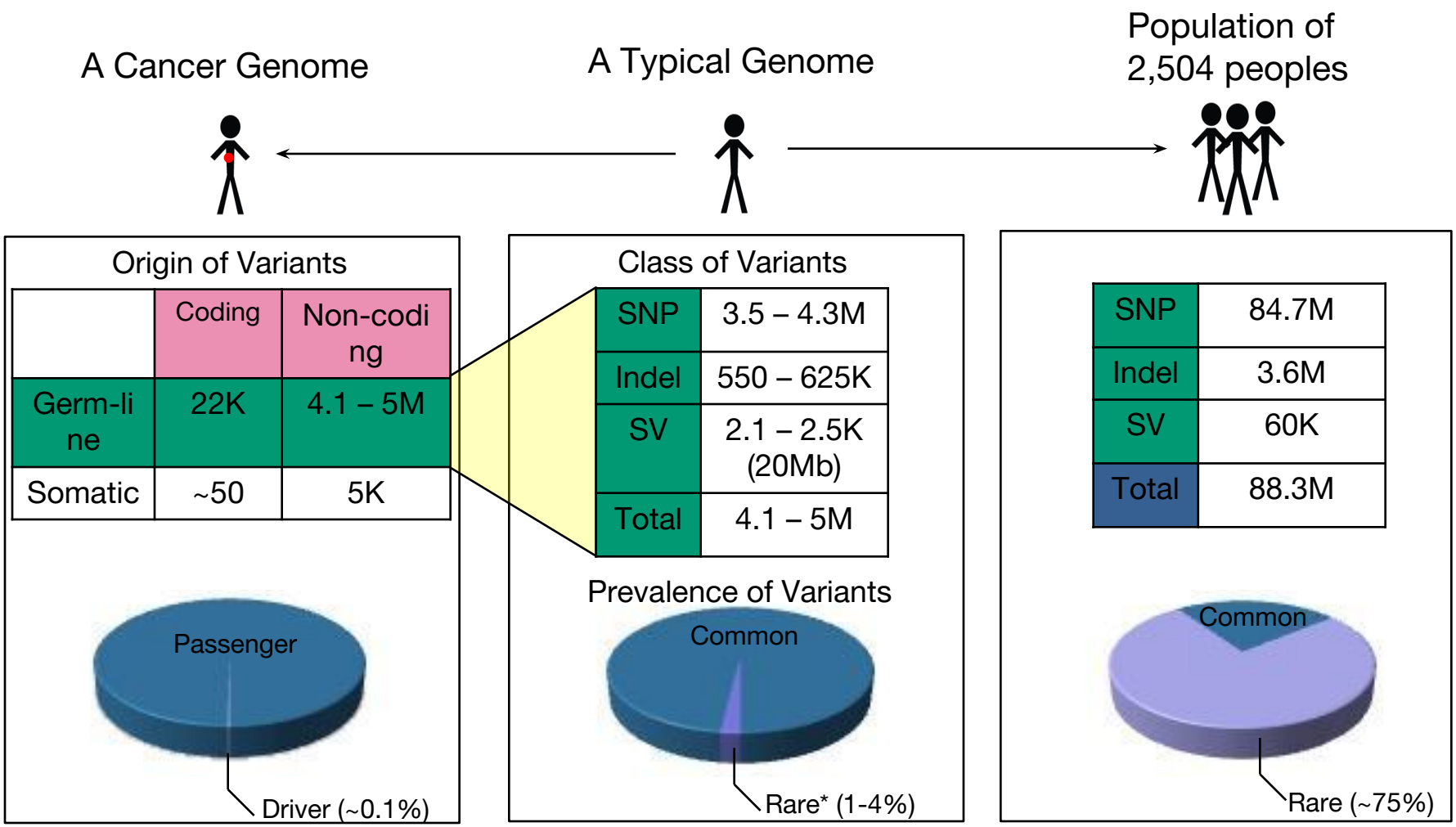
Resource type	Name	URL	Comment
Cloud platform	Amazon EC2	https://aws.amazon.com/ec2/	Most popular cloud platform
	Microsoft Azure	https://azure.microsoft.com/	Second-largest cloud platform
Plug-and-play cloud GPU services	FloydHub	https://www.floydhub.com/	All startups in the GPU service space; pay-by-the-hour model on top of basic monthly subscriptions
	PaperSpace	https://www.paperspace.com/	
	Valohai	https://valohai.com/	
	Google CloudML	https://cloud.google.com/ml-engine/	Can run your own models on Google's hardware, including tensor processor units
	Google Colaboratory	https://colab.research.google.com/	Notebook environment with free GPUs (during 12 h)
Design services for deep learning models	Fabrik	https://github.com/Cloud-CV/Fabrik/	Model export to Keras code; no training
	IBM Data Cloud	https://datascience.ibm.com/	Model export to Keras, PyTorch, TensorFlow or Caffe
	DeepCognition	http://deepcognition.ai/	Training and evaluation included
Prebuilt images with CUDA support	Docker Hub	https://hub.docker.com/r/nvidia/cuda/	Docker images from NVIDIA with CUDA/cuDNN GPU support
	Amazon Deep Learning AMIs	https://aws.amazon.com/machine-learning/amis/	Amazon Machine Images (AMIs) with GPU support
Software libraries (general)	Keras	https://keras.io/	More high-level than TensorFlow (below) but can be integrated with it in many ways
	TensorFlow	https://www.tensorflow.org/	Developed by Google; most popular deep learning framework
	PyTorch	http://pytorch.org/	Developed by Facebook
Software libraries (specific for genomics)	DragoNN	https://kundajelab.github.io/dragonn/	Tutorials included
	Kipoi	http://kipoi.org/	Model zoo for deep learning in genomics
Educational resources	fast.ai	http://www.fast.ai/	E.g., Deep Learning for Coders 1 and 2
	Coursera	https://www.coursera.org/specializations/deep-learning/	Deep-learning-specialization course package
	Textbook	http://neuralnetworksanddeeplearning.com/	Free online textbook with example code
	Fast.ai tips on configuring a deep learning environment	https://github.com/reshamas/fastai_deeplearn_part1/blob/master/README.md#platforms-for-using-fastai-gpu-required/	Instructions for configuring deep learning frameworks for a variety of platforms; from the fast.ai course but general; the details of these procedures change quickly
	Setting up TensorFlow with GPU on Google Cloud Engine	https://medium.com/google-cloud/jupyter-tensorflow-nvidia-gpu-docker-google-compute-engine-4a146f085f17/	Recipe for Docker-based setup of Google Cloud instance with TensorFlow, GPU support and Jupiter Notebooks

Zou, J. et al., 2018. A primer on deep learning in genomics. *Nature genetics*. Available at: http://dx.doi.org/10.1038/s41588-018-0295-5.

Biological problems that we follow and work on

Part V

Human genetic variation



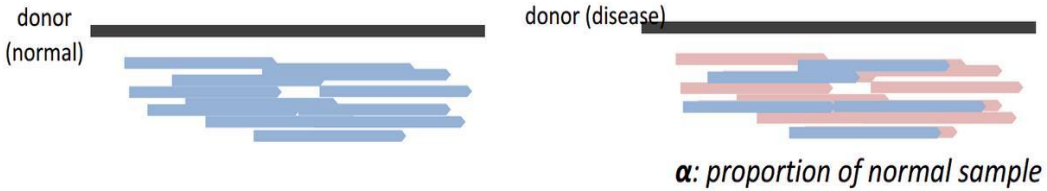
* Variants with allele frequency < 0.5% are considered as rare variants in 1000 genomes project.

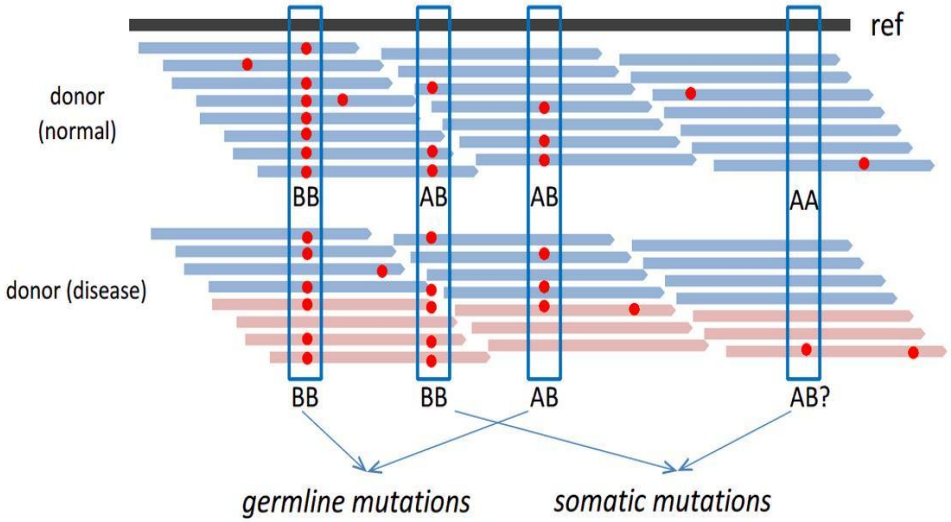
The 1000 Genomes Project Consortium, Nature. 2015. 526:68-74
Khurana E. et al. Nat. Rev. Genet. 2016. 17:93-108

Variant calling

Searching the haystack

3.5 million SNPs



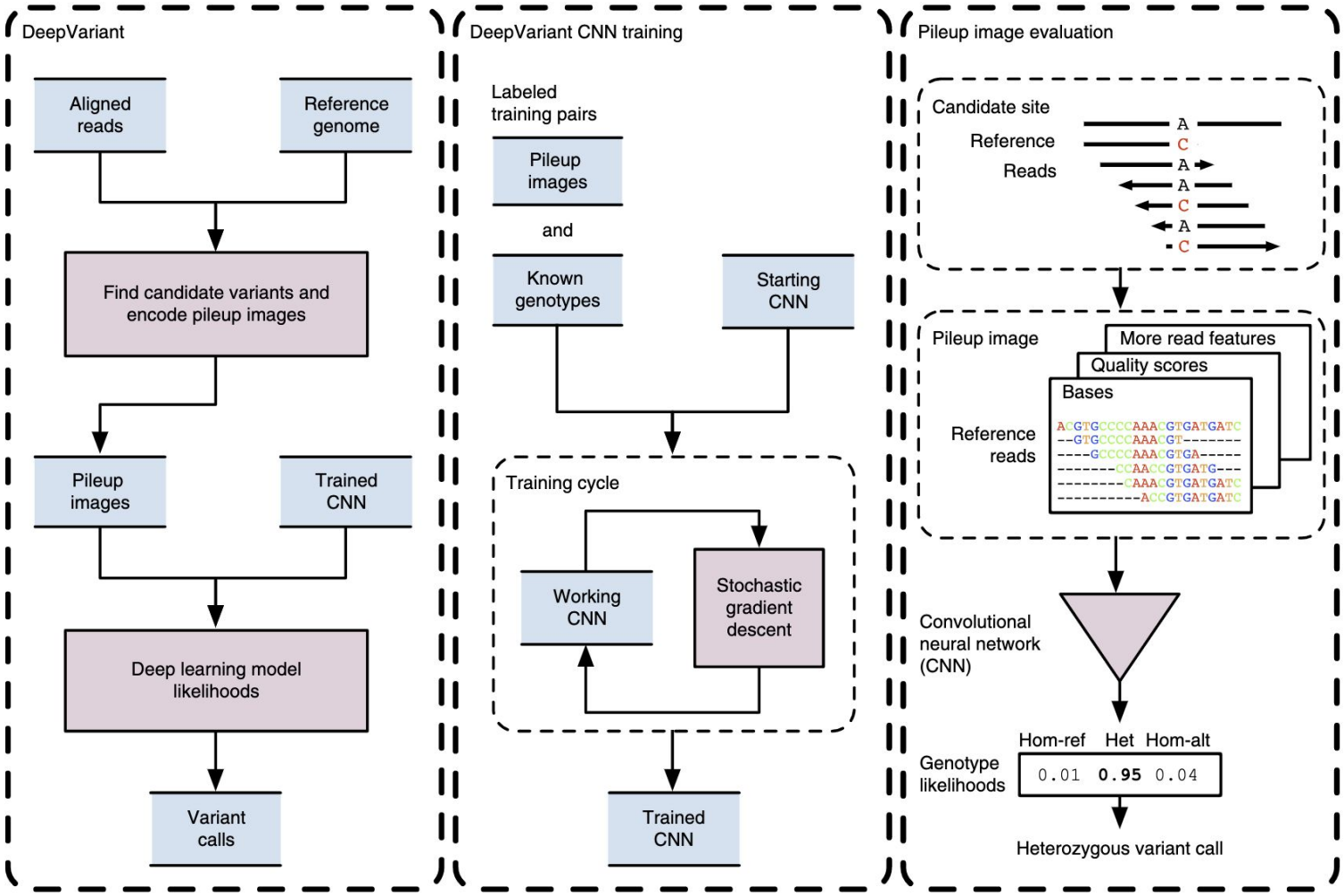


Alignment -> Improving -> Variant calling -> Filtering

Resulting file type: vcf

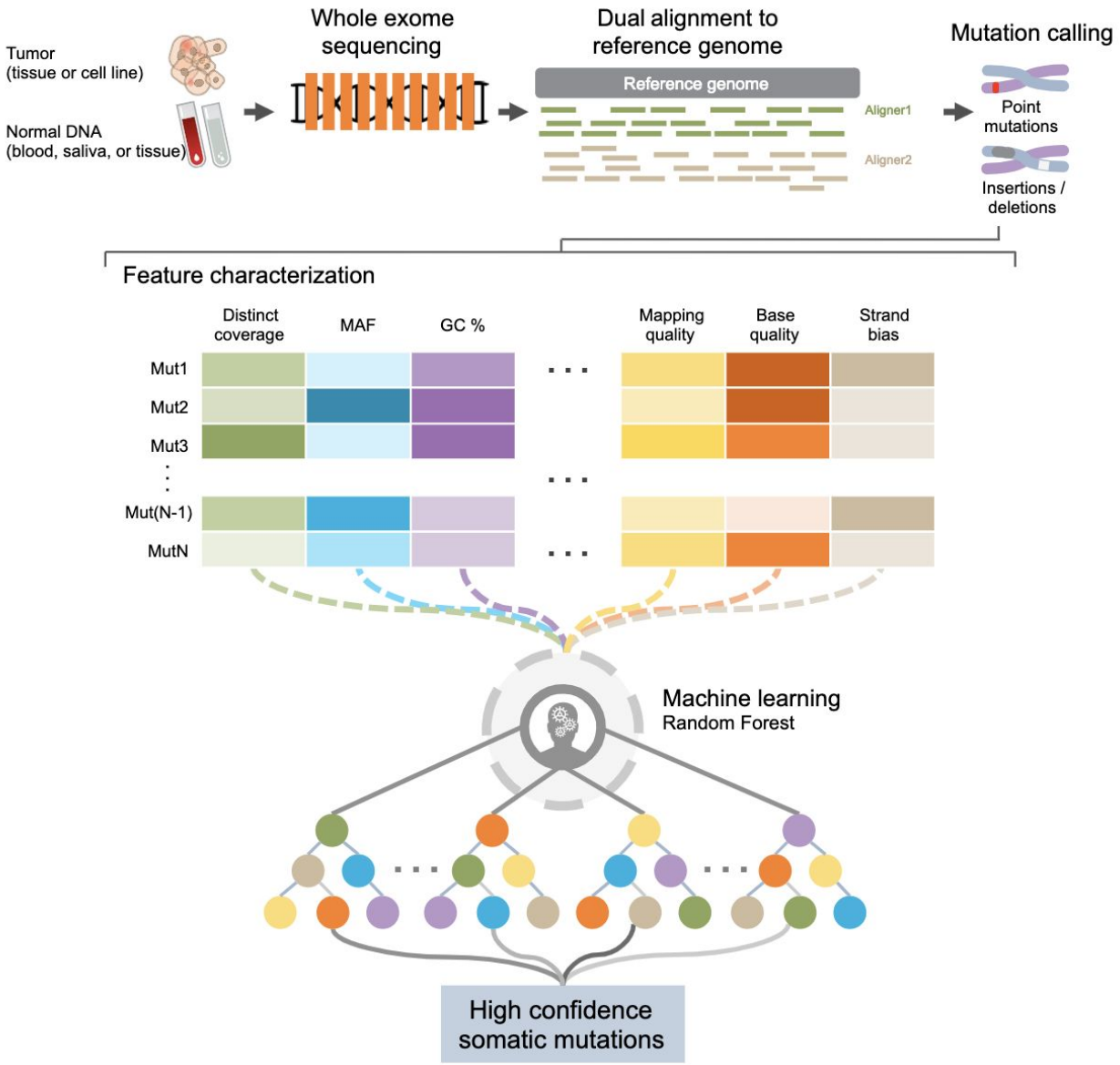
"What are the differences to the reference genome?"

Variant caller using Deep Neural Networks



Poplin, R. et al., 2018. A universal SNP and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10), pp.983–987.

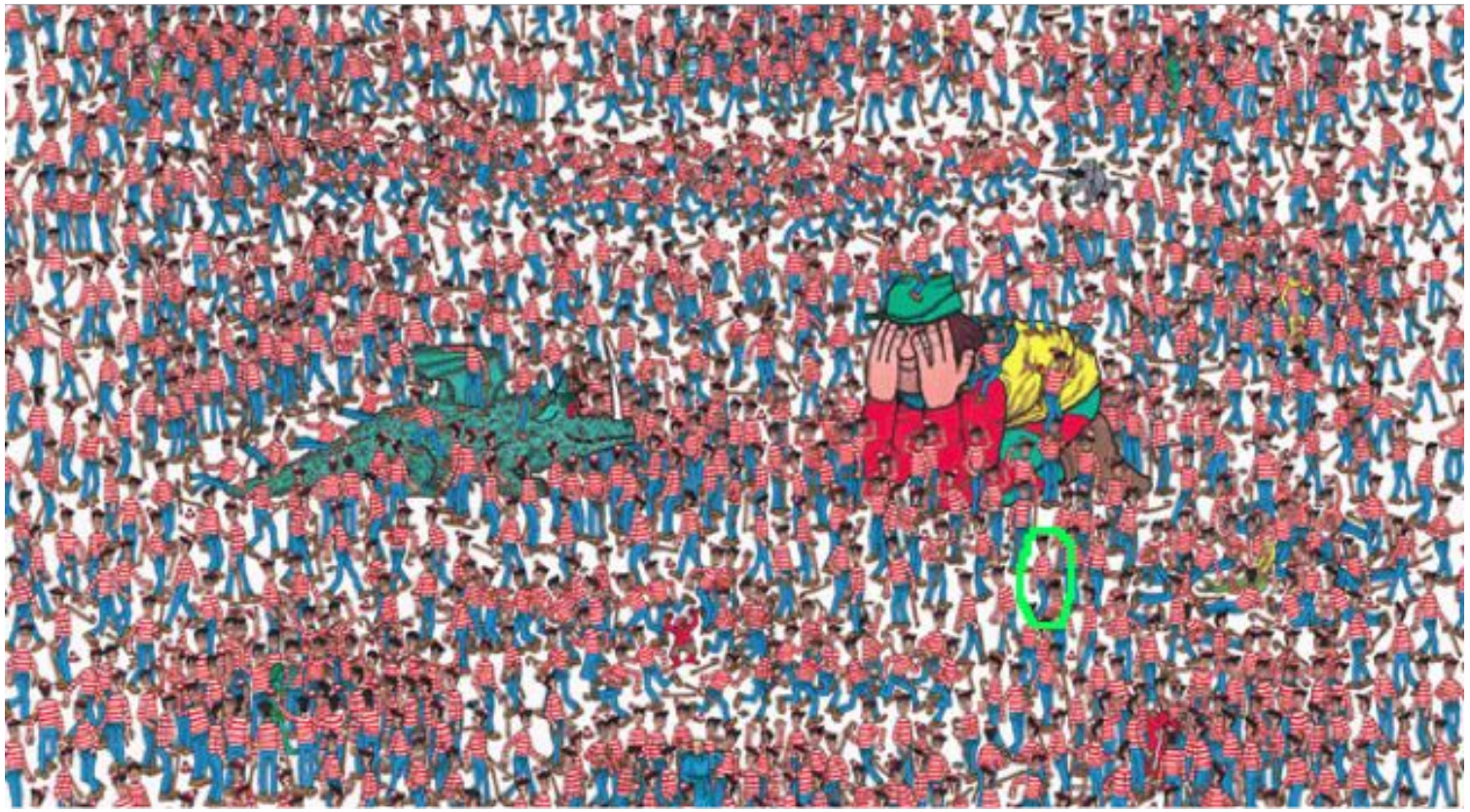
Somatic mutation discovery using Random Forest



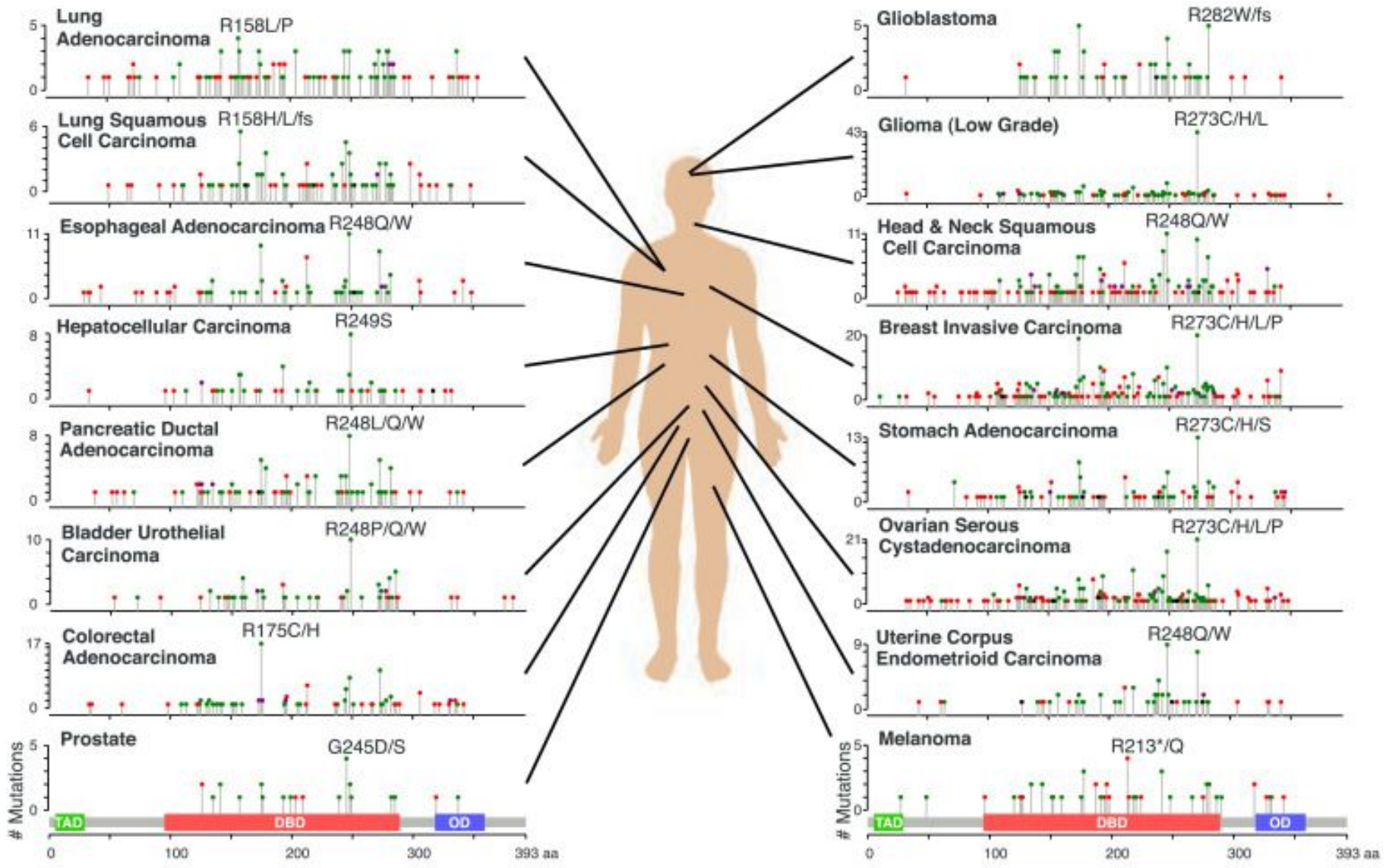
Wood, D.E. et al., 2018. A machine learning approach for somatic mutation discovery. *Science translational medicine*, 10(457), http://dx.doi.org/10.1126/scitranslmed.aar7939.

How does it impact diseases?

(Finding the key mutations in ~3M Germline variants &
~5K Somatic Variants in a Tumor Sample)

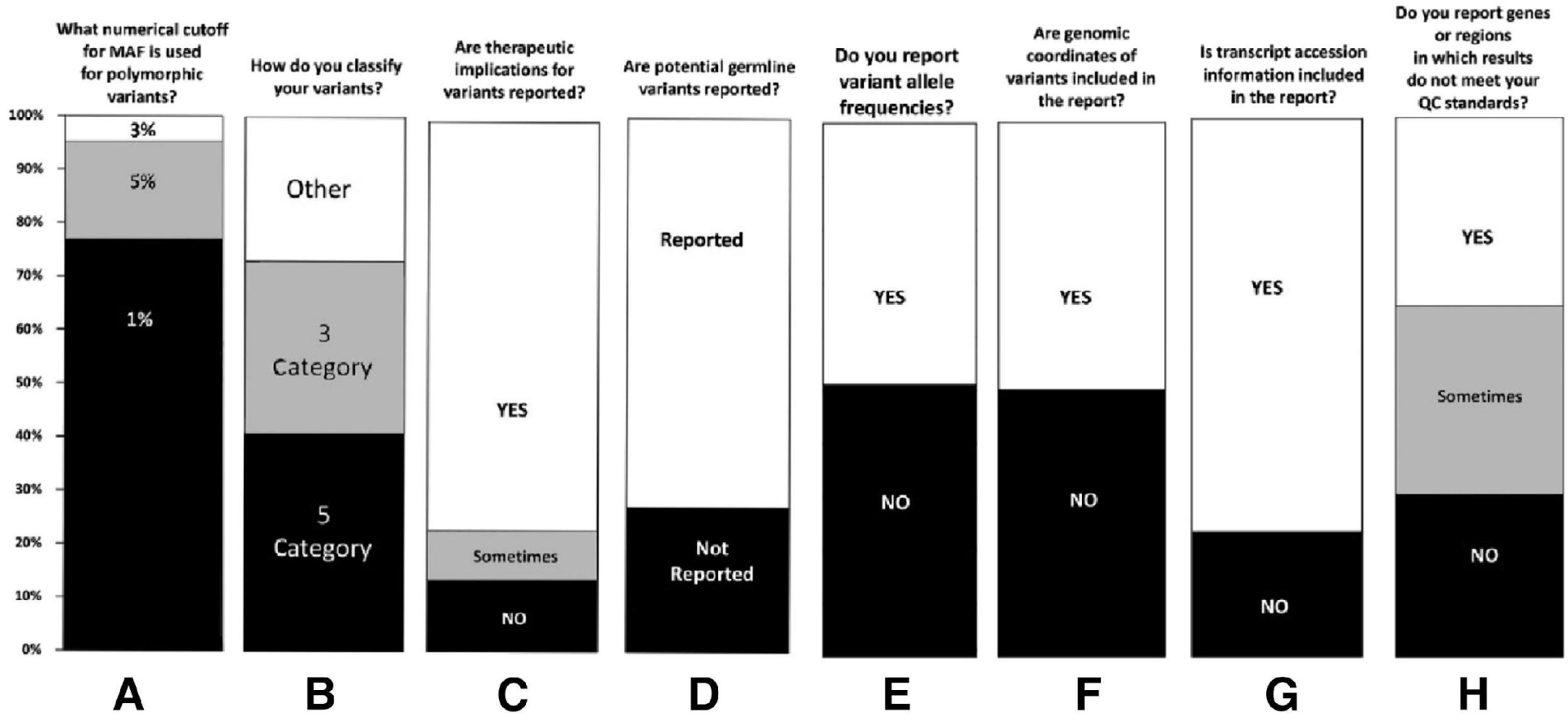


Variant prioritization

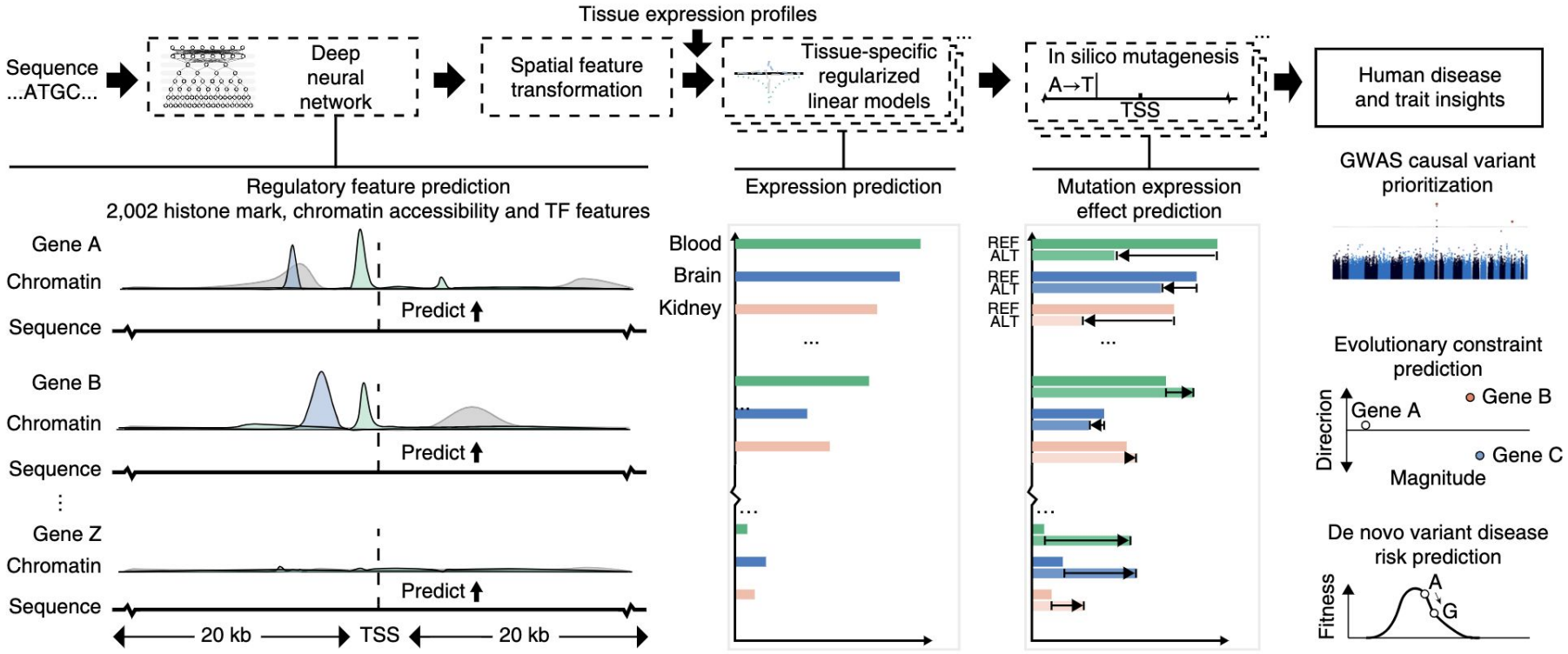


Cell 2017 170, 1062-1078DOI: (10.1016/j.cell.2017.08.028)

Arbitrary classification of pathogenicity of variants

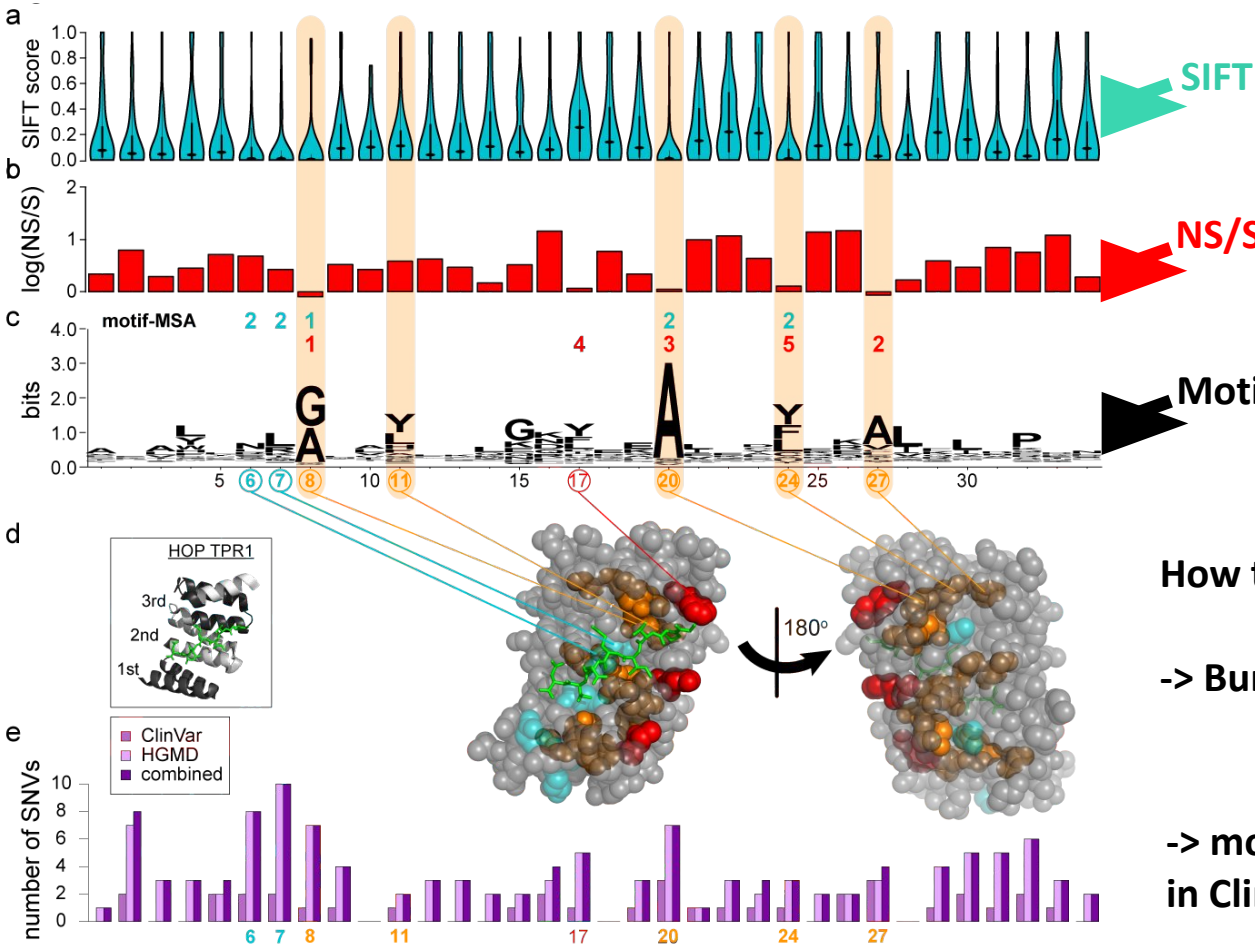


Deep Learning for ab initio prediction of variant effects



Zhou, J. et al., 2018. Deep learning sequence-based ab initio prediction of variant effects on expression and disease risk. *Nature genetics*, 50(8), pp.1171–1179.

Variants in protein coding regions



Selection in PPI motifs

How to check possible significance:

-> Burial within structure

-> more SNVs implicated in diseases in ClinVar and HGMD

Deep Generative Models to capture genetic variations on proteins

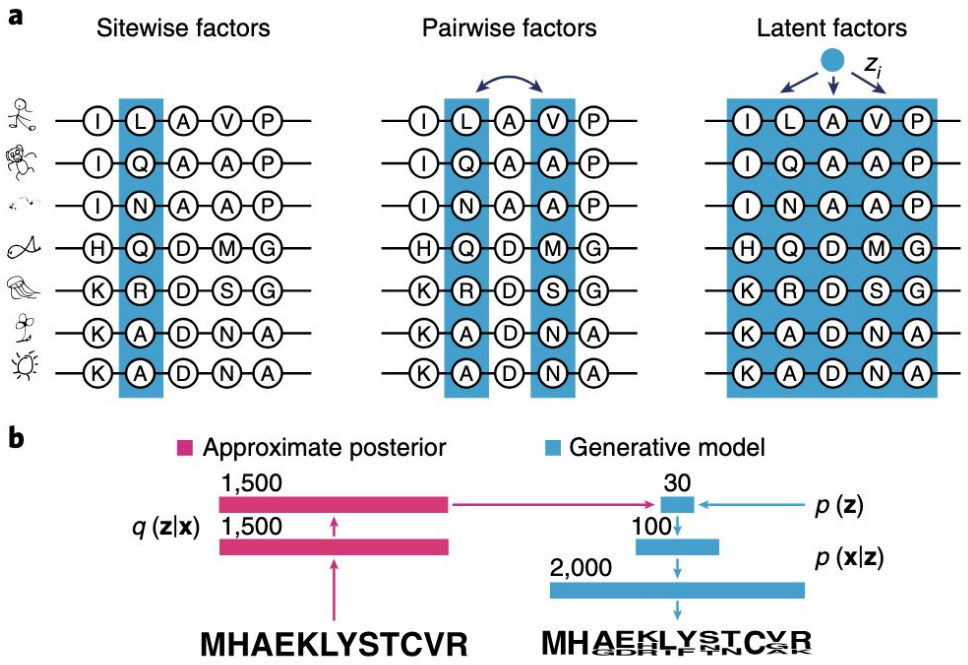


Fig. 1 | A nonlinear latent-variable model captures higher-order dependencies in proteins and RNAs. a, Comparison of a nonlinear latent-variable model with site-independent and pairwise models. **b**, The dependency $p(\mathbf{x}|\mathbf{z})$ (blue) of the sequence **x** on the latent variable **z** is modeled by a neural network, and inference and learning are made tractable by joint training with an approximate inference network $q(\mathbf{z}|\mathbf{x})$ (pink). This combination of model and inference is also known as a variational autoencoder. The size of the latent variables **z** and hidden dimensions of the neural network are shown.

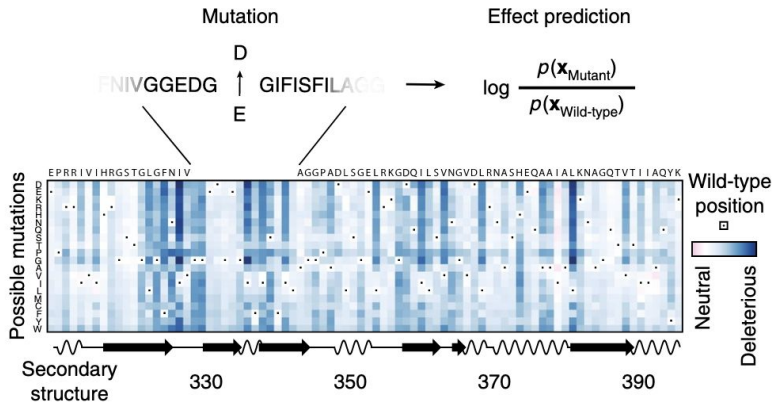


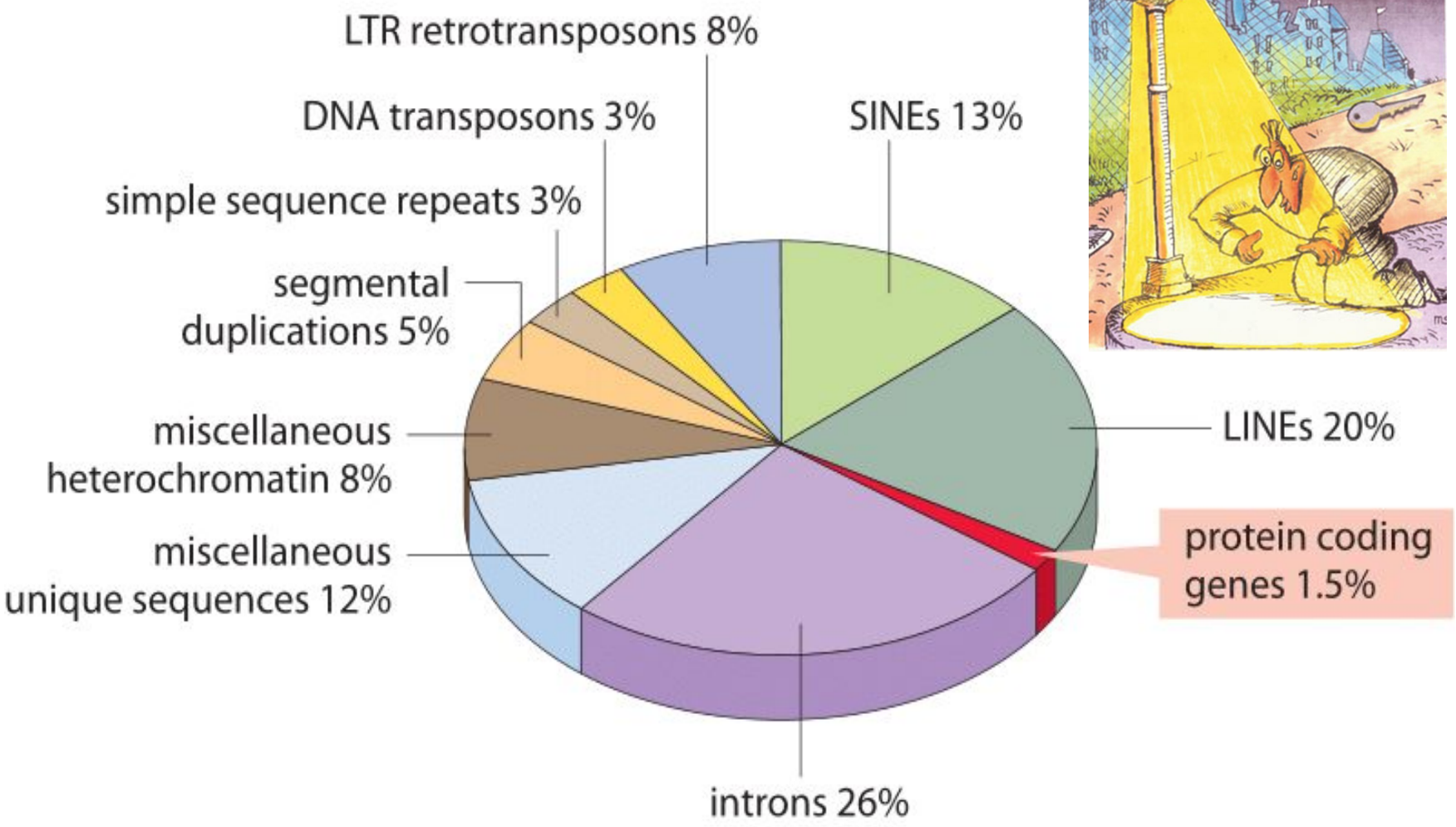
Fig. 2 | Mutation effects can be quantified by likelihood ratios. After fitting a probabilistic model to a family of homologous sequences, we heuristically quantified the effect of mutation as the log-ratio of mutant likelihood to wild-type likelihood (as approximated by the ELBO; Methods). Bottom: mutation-effect scores for positions 310–393 in the PDZ domain.

DeepSequence

Riesselman, A.J., Ingraham, J.B. & Marks, D.S., 2018. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10), pp.816–822.

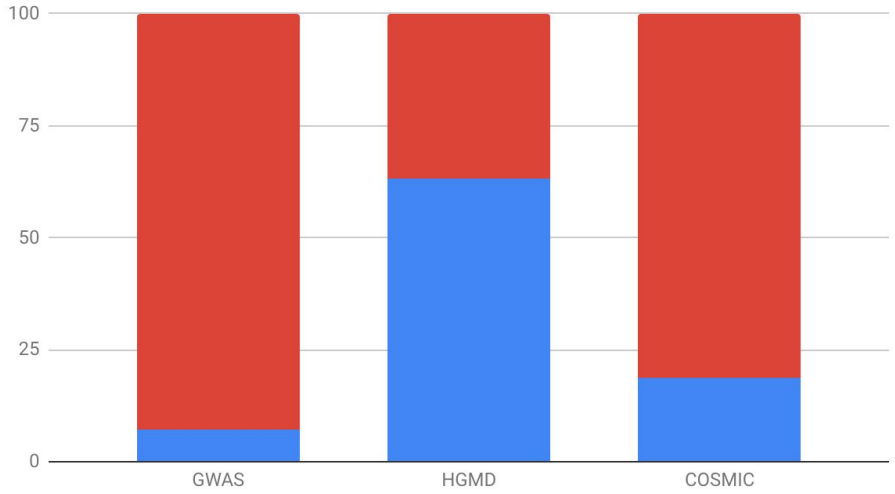
Problem 2

main components of the human genome

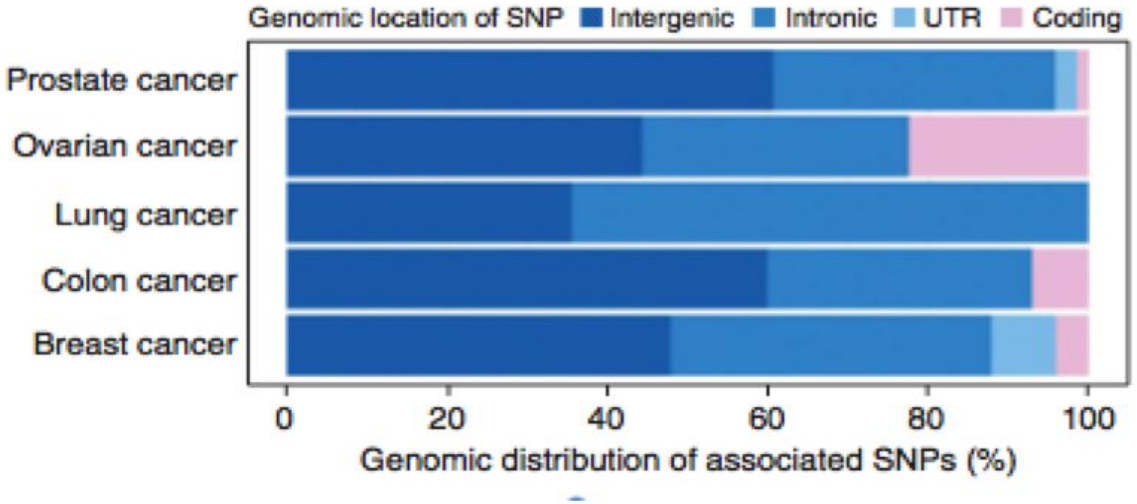


Ref 1

Cancer associated variants

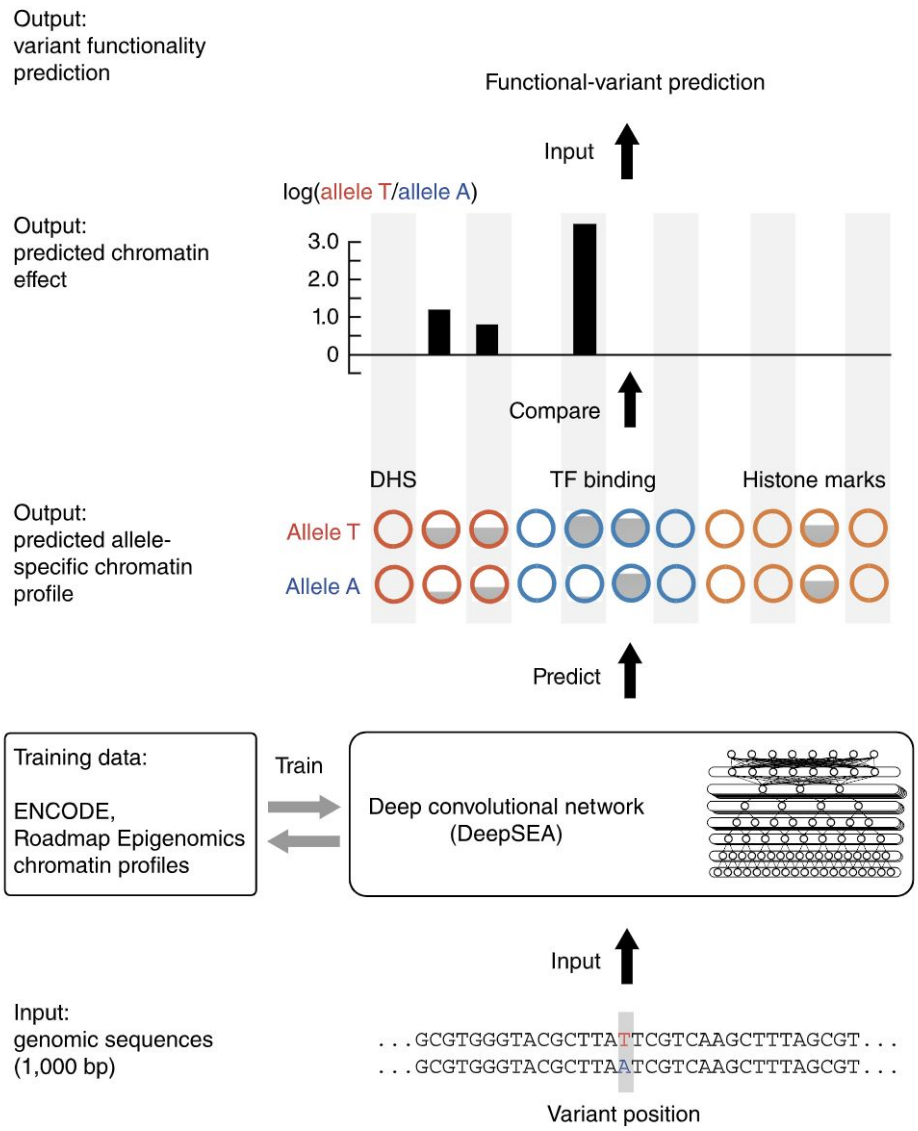


Prabakaran Lab, Manuscript in preparation



S. W. Cheetham et al., Long non coding RNAs and the genetics of Cancer. British Journal of Cancer, 108 (12), 2419-2425 (2013)

DeepSEA pipeline



strategy for predicting chromatin effects of noncoding variants.

Zhou, J. & Troyanskaya, O.G., 2015. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature methods*, 12(10), pp.931–934.

Properties of predictive models of six tools

Table 1 Properties of predictive models for six tools

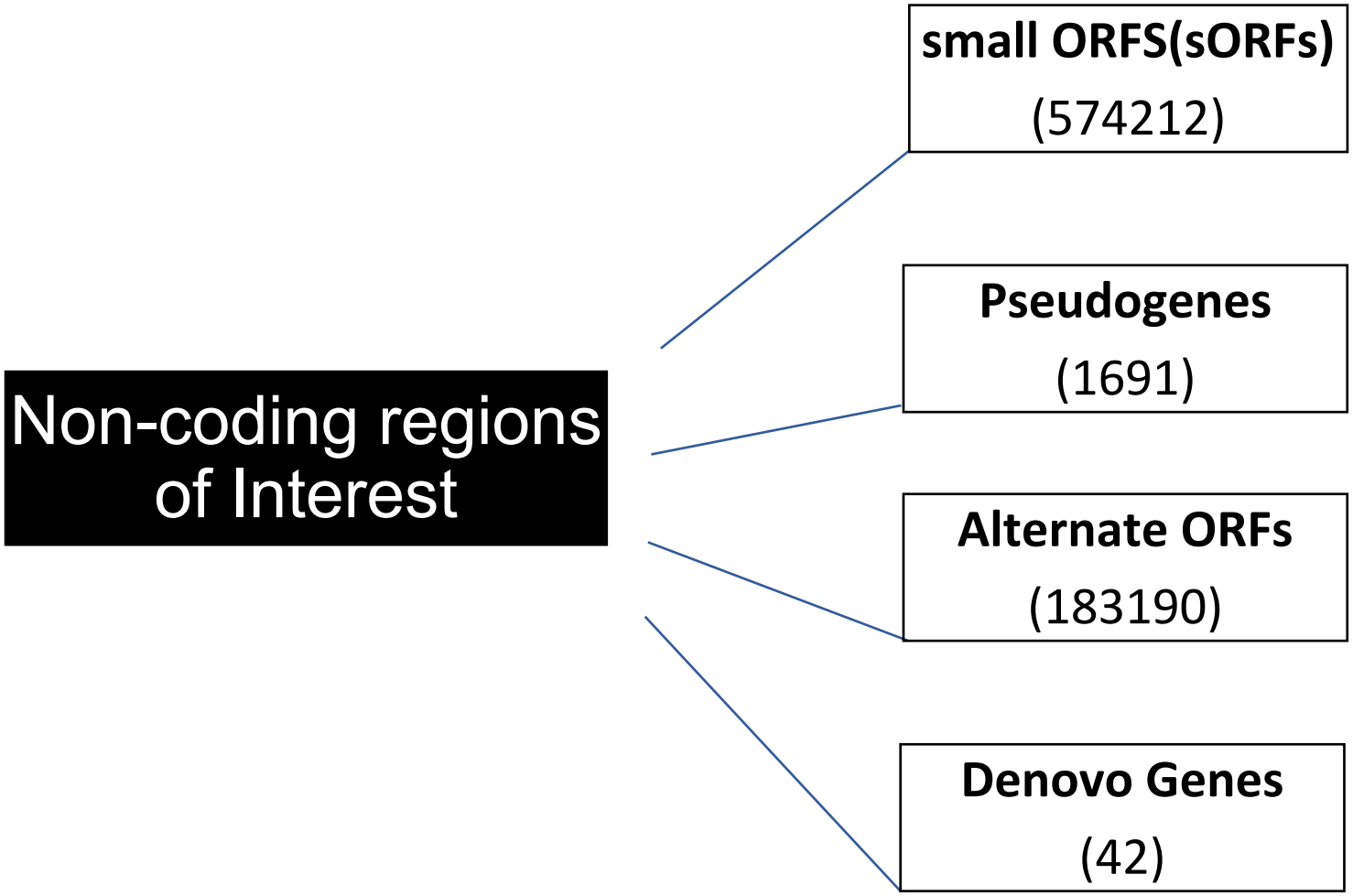
Method	Assumption of pathogenicity	Predictors	Modeling approaches	Performance (AUROC)[a]
CADD	Evolutionary fitness	Evolutionary parameters, ENCODE summaries, functional annotations, population frequencies	Support vector machines	0.92[b]
CATO	Molecular functions	Cell type- and tissue-specific assays, evolutionary parameters, functional annotations	Logistic regression	NA[c]
DeepSEA	Molecular functions	Local sequences, evolutionary parameters	Deep learning, Logistic regression	0.85
EIGEN	None[d]	Evolutionary parameters, ENCODE summaries, population frequencies	Unsupervised learning	0.79
GWAVA	DAVs vs. CPPs	Evolutionary parameters, ENCODE summaries, population frequencies	Random forests	0.97
LINSIGHT	Evolutionary fitness	Evolutionary parameters, ENCODE summaries, functional annotations	Generalized linear model	0.96

AUROC = area under the receiver operator characteristic curve, DAV = disease-associated variant, CPP = common population polymorphism
[a]Highest AUROC values in classifying DAVs and CPPs reported in the original publications
[b]CADD reported AUROC values that mixed coding and noncoding variants
[c]CATO predicts transcription factor occupancy instead of pathogenicity
[d]EIGEN uses an unsupervised learning approach and thus makes no assumption of pathogenicity during training

Liu, L. et al., 2019. Biological relevance of computationally predicted pathogenicity of noncoding variants. *Nature communications*, 10(1), p.330.

Problem 2 questions

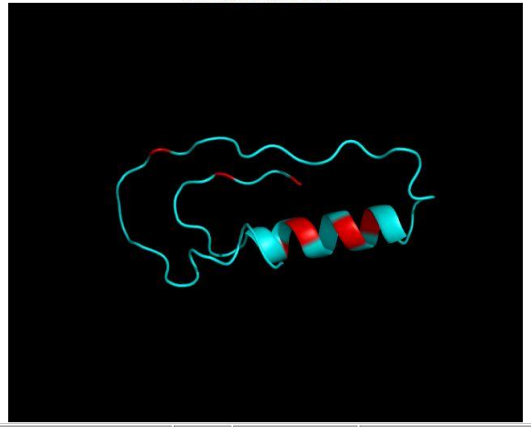
1. Regions of noncoding genome



2. Prioritizing variants in these regions using machine learning approaches

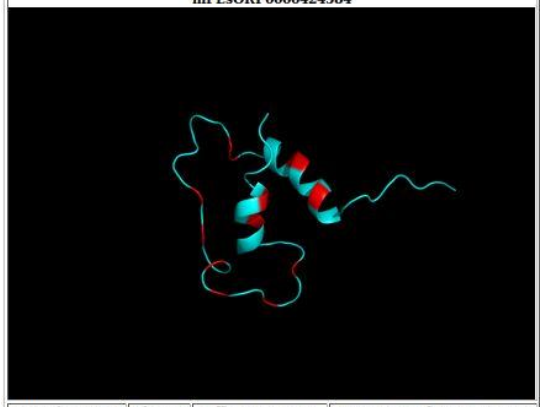
Examples of sORFs with mutations

mPLsORF0000239729



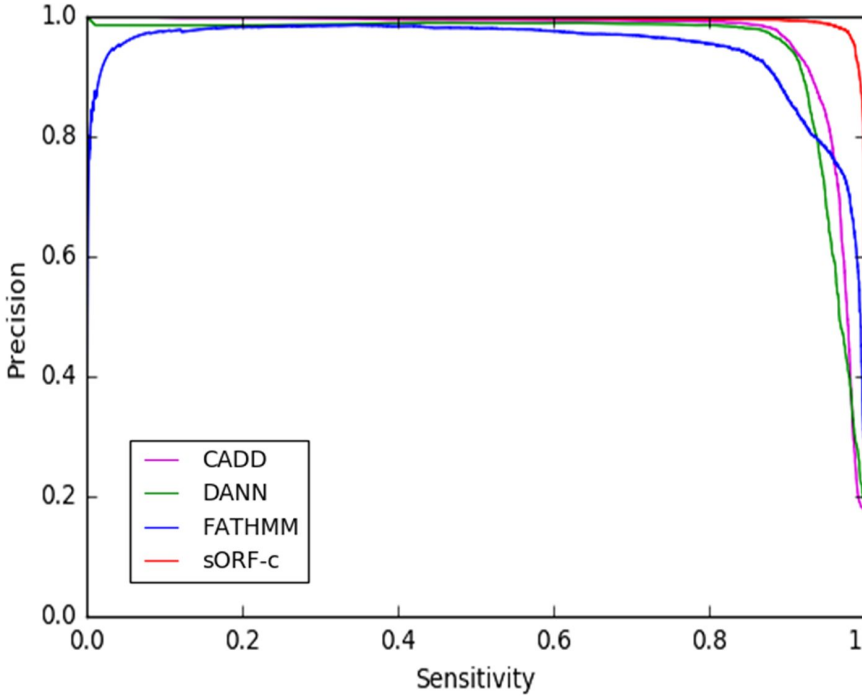
Mutation ID(s)	Change	Effect on sORF	sORF secondary structure
COSM3816594-AA=p.W105* COSM3816595-AA=p.W105*	G>A	W16>* (PDB W17)	H
COSM4056367-AA=p.Y91H COSM4056368-AA=p.Y91H	T>C	Y2>H (PDB Y3)	E
COSM5632701-AA=p.L109V COSM5632702-AA=p.L109V	C>G	L20>V (PDB L21)	H
COSM6324258-AA=p.A94V COSM6324259-AA=p.A94V	C>T	A5>V (PDB A6)	C
COSM6761144-AA=p.E103D COSM6761145-AA=p.E103D	G>T	E14>D (PDB E15)	H
COSM964154-AA=p.D111Y	G>T	D22>Y (PDB D23)	H
COSM964155-AA=p.R129H	G>A	R40>H (PDB R41)	C

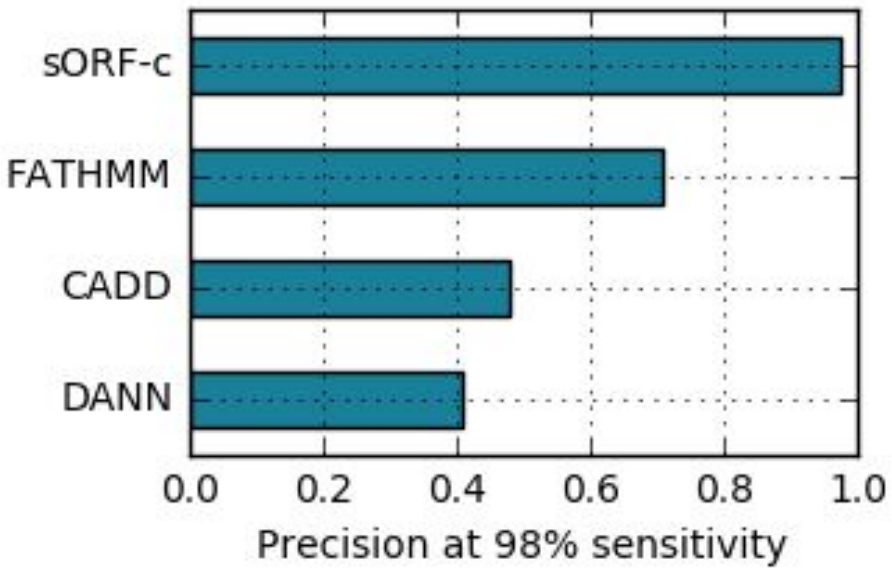
mPLsORF0000424584



Mutation ID(s)	Change	Effect on sORF	sORF secondary structure
COSN1260962 COSN5770834	A>G	Y10>C (PDB Y10)	H
COSN18723188	C>T	P32>L (PDB P32)	C
COSN20080080	C>T	R13>C (PDB R13)	C
COSN20121983	C>T	I28>I (PDB I28)	C
COSN20648546 COSN21264986 COSN23079780	G>A	W50>* (PDB R50)	H
COSN20650206 COSN21677358 COSN22583798 COSN23084029	C>T	G42>G (PDB G42)	E
COSN20650207 COSN22687956 COSN23081188	T>C	W50>R (PDB R50)	H
COSN21264985 COSN22557586 COSN23085391	A>G	K49>K (PDB K49)	H
COSN21675097	T>C	Y43>Y (PDB Y43)	H
COSN23082255	G>A	K34>K (PDB K34)	H
COSN23087874	A>G	K26>E (PDB E26)	H
COSN24397730	G>A	R23>H (PDB R23)	H
COSN26666338	A>T	E67>V	H
COSN27001349 COSN8016762	G>A	T20>T (PDB T20)	C
COSN27001446	C>T	T20>M (PDB T20)	C
COSN27001936	G>T	R70>M	C
COSN27003013	G>A	G54>D (PDB G54)	C

Benchmarking sORF-c perfomance





Jackson, F., Wayland, M.T. & Prabakaran, S., 2017. Identification And Prioritisation Of Variants In The Short Open-Reading Frame Regions Of The Human Genome.
bioRxiv, p.133645. Available at: http://biorxiv.org/content/early/2017/05/03/133645

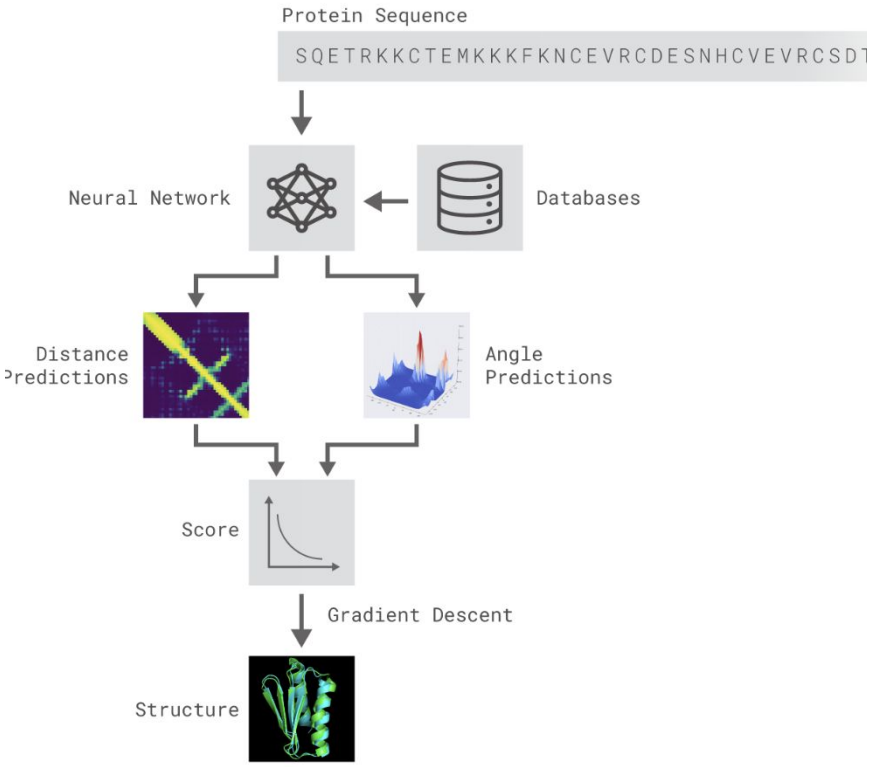
DeepMind Alpha Fold

AlphaFold: Using AI for scientific discovery

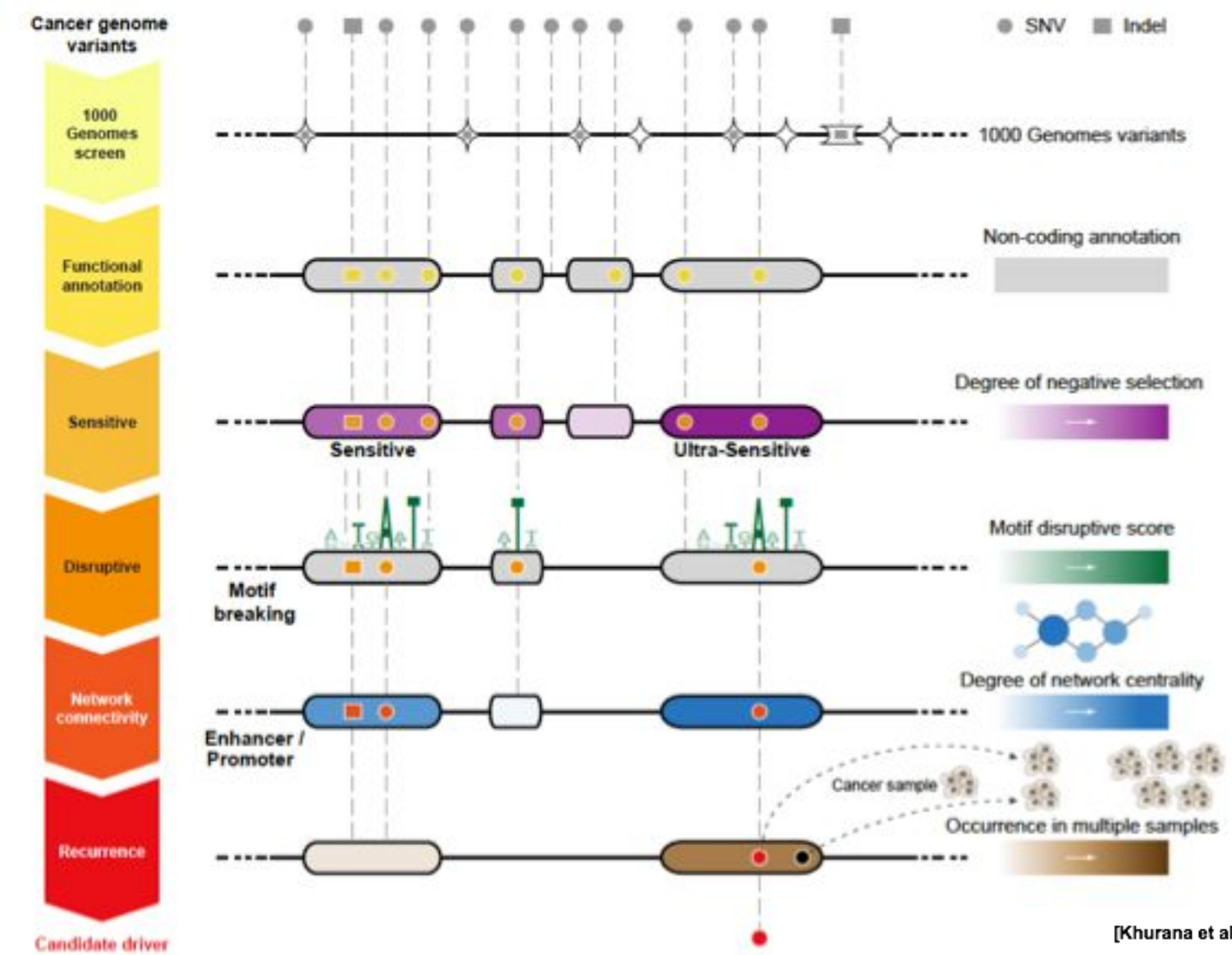
Today we're excited to share DeepMind's first significant milestone in demonstrating how artificial intelligence research can drive and accelerate new scientific discoveries. With a strongly interdisciplinary approach to our work, DeepMind has brought together experts from the fields of structural biology, physics, and machine learning to apply cutting-edge techniques to predict the 3D structure of a protein based solely on its genetic sequence.

Our system, **AlphaFold**, which we have been working on for the past two years, builds on years of prior research in using vast genomic data to predict protein structure. The 3D models of proteins that AlphaFold generates are far more accurate than any that have come before—making significant progress on one of the core challenges in biology.

What is the protein folding problem?

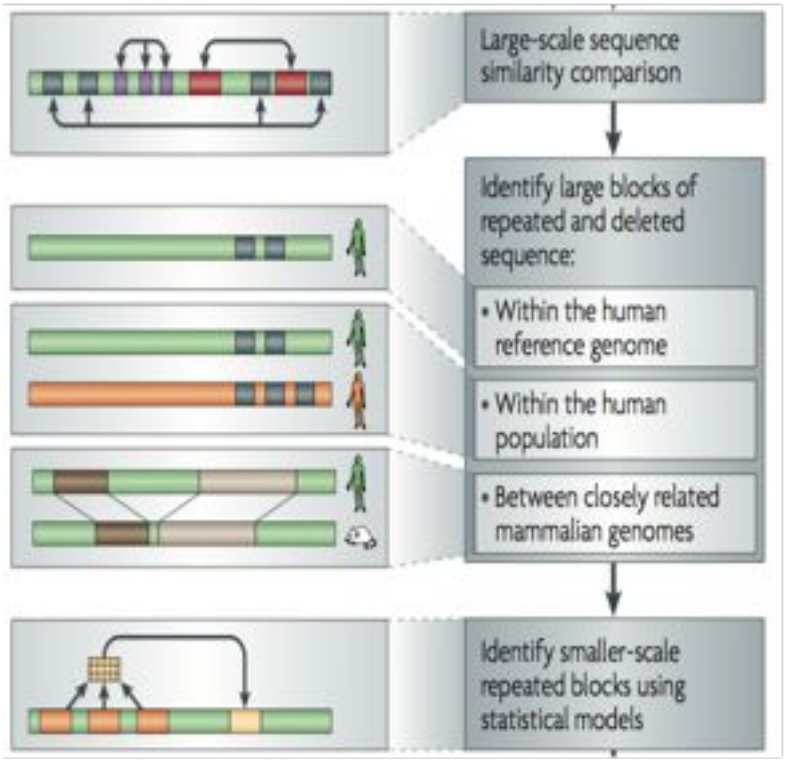


Other attempts are variant prioritization in noncoding regions

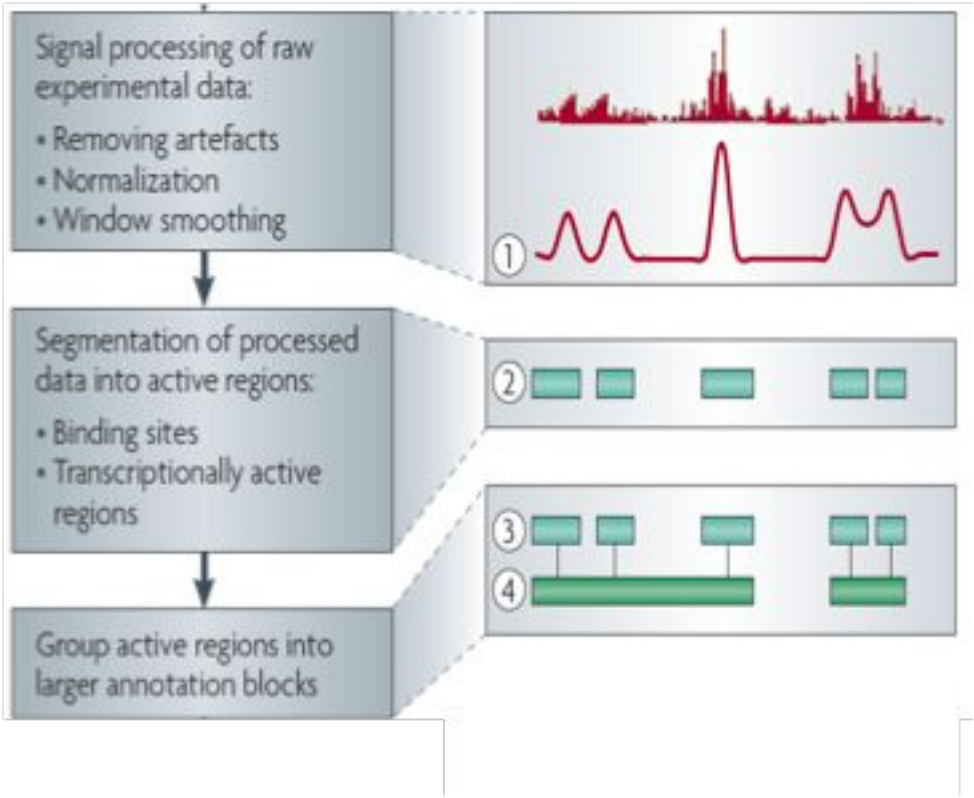


[Khurana et al., *Science* ('13)]

Examples of genome annotation

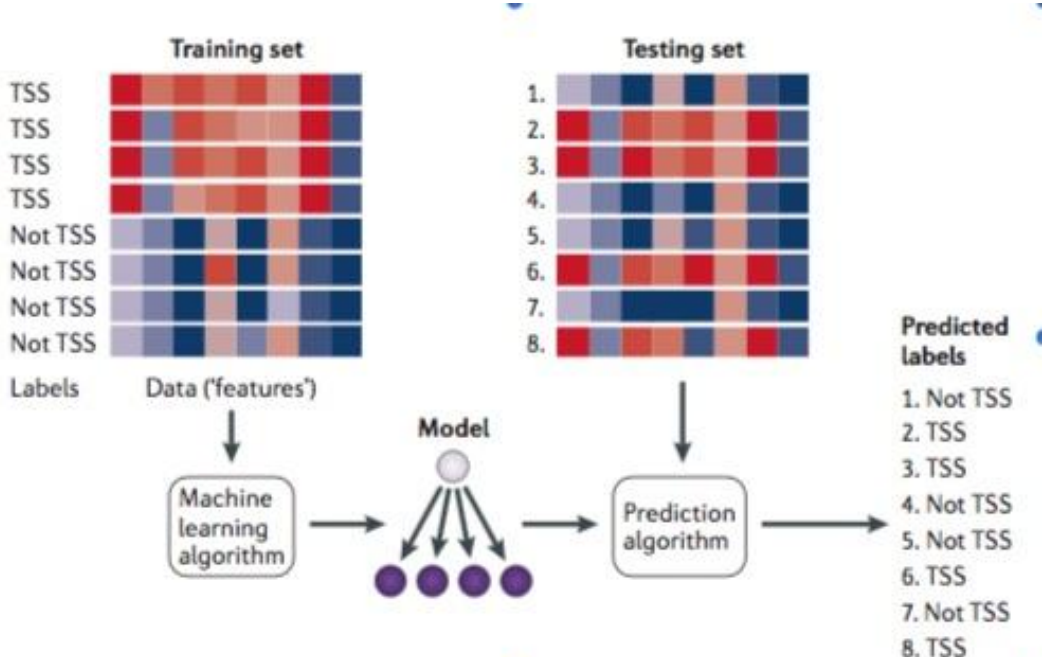


Functional Genomics
Chip-seq (Epigenome & seq. specific TF)
and ncRNA & un-annotated transcription



Annotation of transcription factor binding sites

Typical supervised ML workflow



Step 1: develop an algorithm that will lead to successful learning.

Step 2: the algorithm is provided with a large collection of TSS sequences as well as, optionally, a list of sequences that are known not to be TSSs. The annotation indicating whether a sequence is a TSS is known as the label. The algorithm processes these labelled sequences and stores a model.

Step 3: new unlabelled sequences are given to the algorithm, and it uses the model to predict labels (in this case, 'TSS' or 'not TSS') for each sequence. If the learning was successful, then all or most of the predicted labels will be correct.

Some more examples of ML in biology

1. To identify splice sites
2. To identify promoters, enhancers, or positioned nucleosomes
3. To annotating genes — including their untranslated regions (UTRs), introns and exons — along entire eukaryotic chromosomes
4. Gene expression (microarrays and RNAseq)
5. Gene Onotology term assignments
6. Infer gene networks
7. DNAase-seq, MNase-seq, FAIRE-seq, ChIP-seq analysis

Thanks!