(Human) Network Analysis

Lecture 4:
Recurrent visual processing

Ben Harvey

In the last class, we looked at how biological and artificial deep networks scale up from simple responses in input layers to complex abilities that emerge in deep feedforward networks.

We finished by looking at a number of effects that cannot be explained by feedforward processing. These include attention, where our goals influence responses of spatial representations and object representations, and dreams, where our object representations are activated with no visual input.

Today we are going to build on that question to see how higher levels of visual processing and interact with lower levels.

Here we will come back to our original focus on artificial deep networks, but we will see that these networks get pretty complex and have weird emergent properties.

But we will also see how the structure of the artificial networks is closely follows the properties of neural processing in the brain more closely that feedforward networks do.

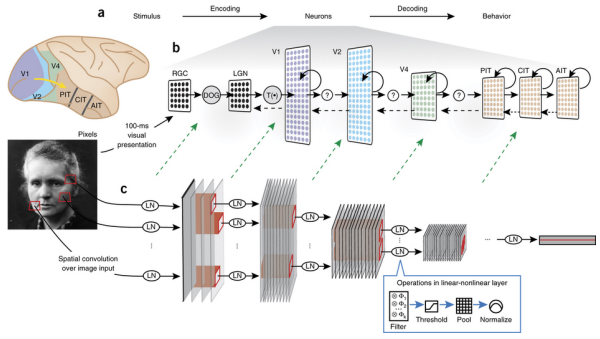
Why recurrent processing?

- Important properties of early and higher visual processing are absent in current artificial DCNNs
- Including these properties may:
  - Give artificial DCNNs new abilities
  - Make artificial DCNNs more brain-like
  - Unite Hebbian learning and backpropagation
  - Reduce the gap between unsupervised (biological) and supervised (artificial) learning

2

These properties are what we will look at today.

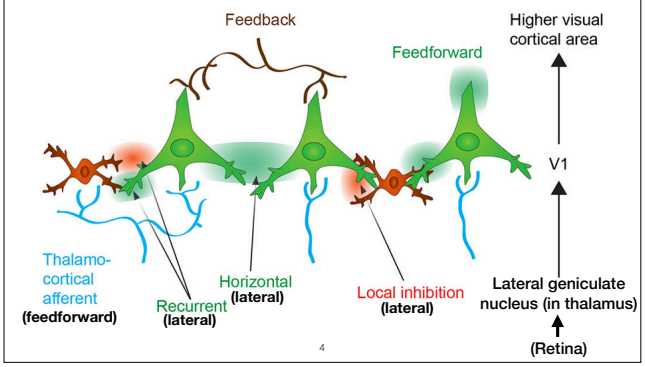
Feed-forward convolutions only?



3

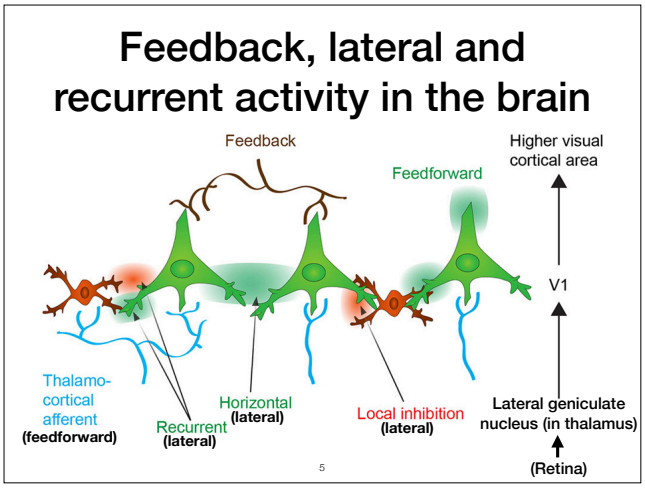
Here we see that a biological network, at the top, has feedforward (or bottom-up) connections, together with feedback (or top-down) connections shown as dashed arrows.
-For these connections to put a precise activity pattern into earlier areas, like in dreams or imagination, each of these feedback arrows must carry a similarly complex pattern of activity to the feedforward arrows that represent our convolutional filters.
-Here we also see inputs to each unit coming from other units in the same layer, called lateral connections.
-In the artificial DCNN at the bottom, both of these are missing
-In many artificial DCNNs, the input is a single static image with no temporal information, so recent activity is always zero and we can ignore it. But in networks with temporal inputs, recurrent connections are common.
-These are particularly important in linguistics, as language is a temporal sequence.

Feedback, lateral and recurrent activity in the brain

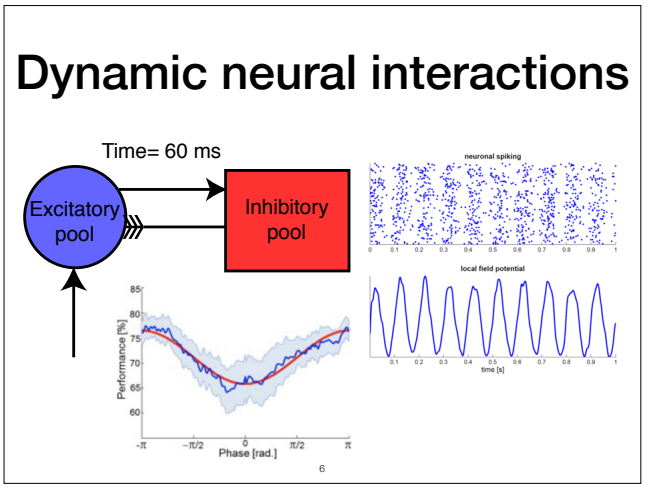


4

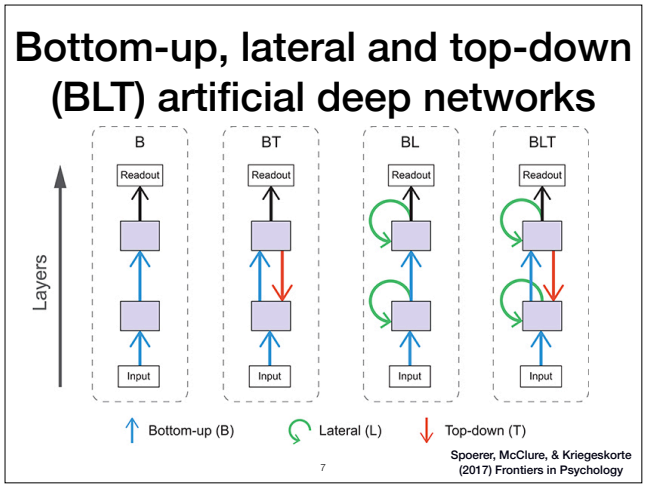
So far, we have only looked at feedforward connections, represented in this case as inputs to V1 from the thalamus, shown in blue at the bottom.
-After some transformation, these will then pass the image representation up to higher visual areas.
-But likewise, feedback connections from these higher areas will affect the activity of the neurons.
-And the neurons will interact with each other, through both excitatory and inhibitory lateral connections (labelled 'horizontal' and 'local inhibition' respectively).
-For example, a cell responding to a vertical orientation at one location may form an excitatory synapse with a cell responding to a vertical orientation at a nearby location, making that more likely to fire.
-Or it may form an inhibitory synapse with a cell responding to a horizontal orientation, making that less likely to fire.



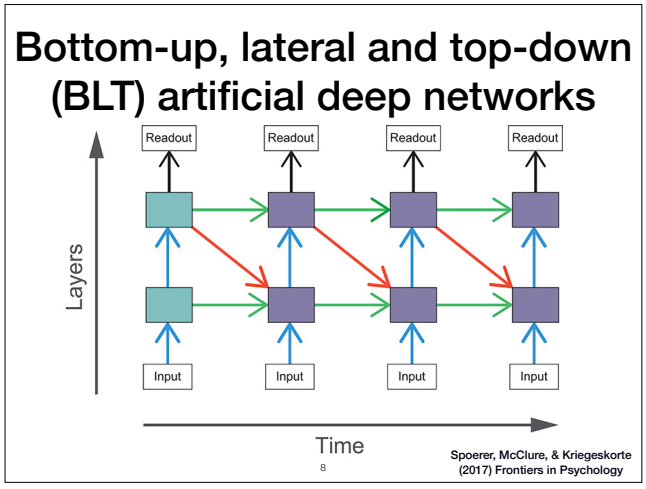
-A neuron can excite or inhibit other neurons in the same brain area, higher or lower areas, changing their activity.
-Importantly, changes in the activity of those connected neurons can in turn excite or inhibit the first neurons (here labelled 'recurrent').
-Note that these neurons never form synapses with themselves, there is always another neuron forming a circuit.
-As these interactions take a moment to happen, the neuron's previous activity will then affect its current activity.
-Indeed, all of these interactions have a temporal component, as each interaction step takes a little time.
-As a result, a feedforward network will reach it's end state with one pass of the information through the network, as information only moves in one direction.
-But networks with interactions like we see here will change the activity in this layer depending on results of surrounding activity and higher level activity.
-The result will in turn change the surrounding activity and the higher level activity, changing their influences back and forth until an equilibrium is reached.
Because of this dynamic interaction, such connections always make biological networks time-dependent.



These dynamic interactions between excitatory and inhibitory neural population produce oscillations in neural population activity, which can be measured using EEG. At oscillation peaks, excitatory population activity is highest, including the spike rate This also affects perception. Just-visible stimuli presented at an oscillation peak are more likely to be perceived than those presented at a trough



Here is a set of artificial network that incorporate the extra connections that exist in biological networks.
So far, the networks we have looked at are of type B, bottom-up connections only.
It is actually pretty straightforward to make the convolutional filter not only sample from the previous layer, but also from surrounding units in the same layer, and from the next layer up.
The convolutional filter needs to be larger, so more weights need the be learned.



Perhaps more complex is that the network's activity will move back and forth as the results of each interaction affect the activity of the interacting elements.
Here the experimenters include a fixed number of cycles, here 4.
Of course, in the brain there are not such discrete time steps and there is no end to the interaction cycles.
Here, the input image remains

the same through these cycles, while the input to the brain will always be changing

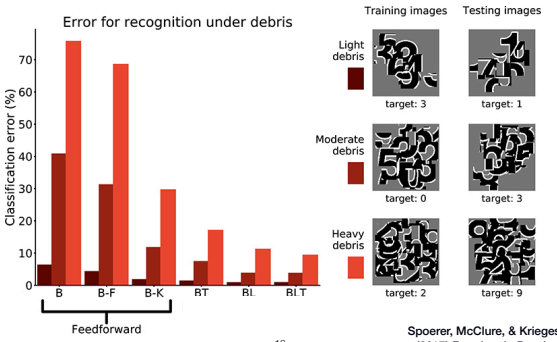
Bottom-up, lateral and top-down (BLT) artificial deep networks

Spoerer, McClure, & Kriegeskorte (2017) Frontiers in Psychology

9

They test this on a digit recognition task.
As you have seen in the labs, this is a pretty easy task and does not require very deep or complex networks.
This gets much harder if we add some parts of other digits to our target digit, but a human can still make out the target digit well enough.

Bottom-up, lateral and top-down (BLT) artificial deep networks



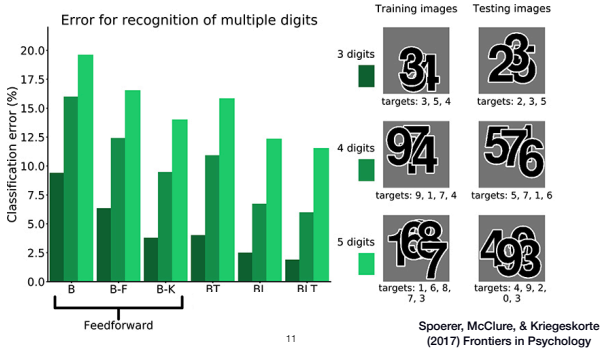
Spoerer, McClure, & Kriegeskorte (2017) Frontiers in Psychology

10

When there is relatively little junk (debris) in the way, a simple bottom-up network does pretty well (far left, dark colours, low error).
But adding a lot of debris (lighter colours) makes performance of a bottom-up network drop quickly.
In a BLT network, we see much better performance in all three levels of debris.
B-F and B-K networks are both bottom-up networks, but the filters are larger or there are more filters so that the same number of weights needs to be learned as in the more complex networks.
These also do well, but clearly the feedback and lateral influences are

improving performance.

Bottom-up, lateral and top-down (BLT) artificial deep networks

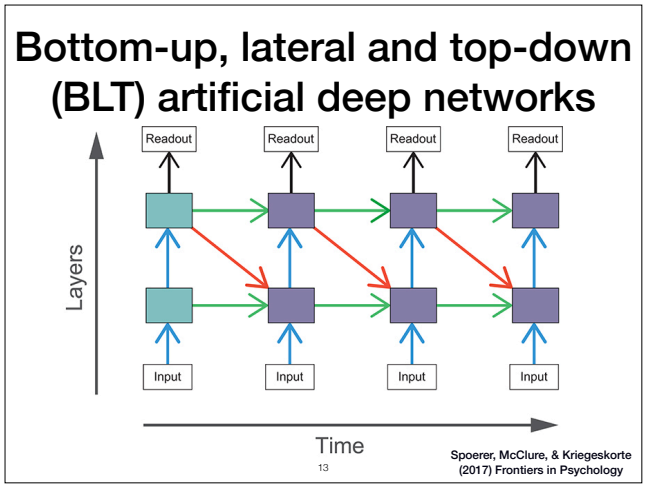


Spoerer, McClure, & Kriegeskorte (2017) Frontiers in Psychology

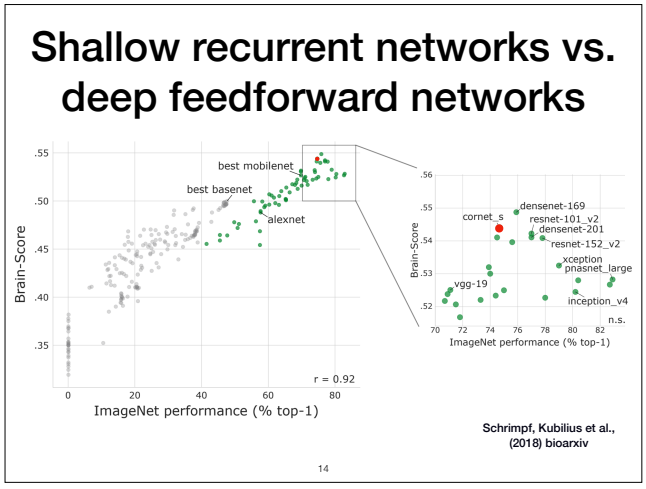
Similarly, when recognising multiple digits, we see an improvement over feed-forward only networks

Feedback, lateral and recurrent activity

- The brain relies heavily on lateral and feedback activity
  - Recurrent activity is implemented through both lateral and feedback connections
  - Results in dynamic neural oscillations
  - No fixed states of activity
- Recent experimental artificial networks are investigating lateral and top-down connections
  - Resulting interactions make the network inherently time-dependent
  - Image classification performance in difficult task improves considerably
  - More computationally intensive
    - More extensive filters
    - Multiple time steps modelled

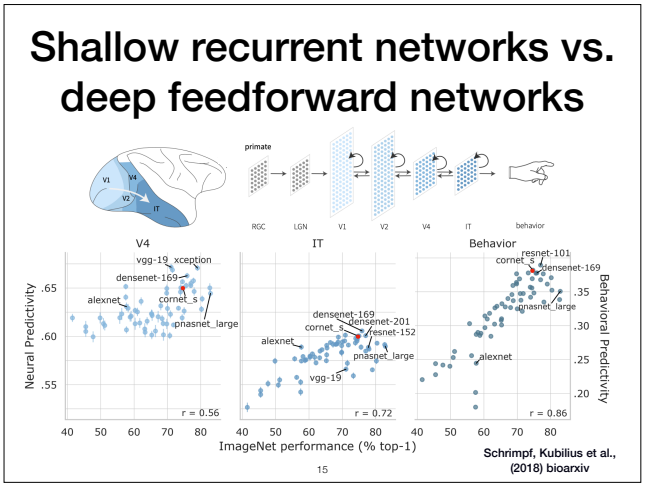


In a network with recurrent connections, the activity is processed by the same layers many times.
This network has only 2 hidden layers, but cycles activation patterns though each 4 times, effectively making 8 layers that responses can be processed through.
Also note that the path of information is not linear, there are longer routes and shorter routes so some of these steps can be skipped.
The layers have the same connection weights each time.
The connection weights will reach some kind of compromise between what is most effective for the first pass, the second pass, and the third and forth.
So a relatively shallow network can effectively be made much deeper using recurrent connections, though some compromises may be necessary.



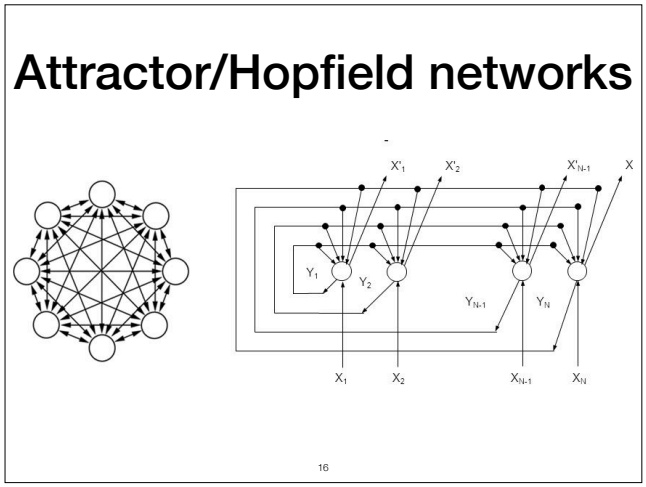
Therefore, shallow recurrent networks can perform much like very deep feedforward networks.
The network ‘cornet-s’ is a 4 layer recurrent network.
Its performance on classifying the ImageNet image set of object photographs is comparable to much deeper, 100 layer networks.
Because recurrent connections allow activity to be analysed over and over again by one layer,

shallow recurrent networks can act like deeper feedforward networks.
But in the context of the massive increase in feedforward network depth, it may more useful to think that a very deep feedforward network can act like a shallow recurrent network like the brain.
But there are far fewer weights to learn, as each cycle through a layer uses the same weights.
The other measure here, Brain-Score, summarises the ability of network units follow the activity of biological neurons and human behaviour. Here cornet does better than most far deeper feedforward networks.

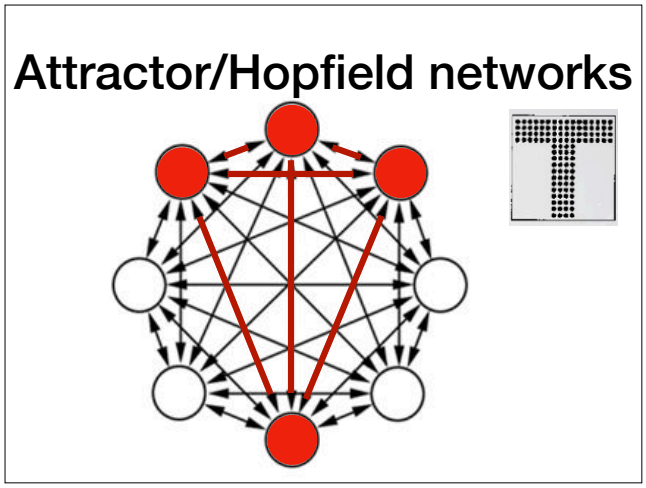


The architecture of cornet is designed to mimic that of biological neural processing's recurrent structure, so perhaps it is not surprising that it mimics the brain's response closely.
The pathway from V1 to inferior temporal cortex only contains around 4 brain areas, so a 100-layer network is clearly not a close match for the brain.
Many networks can predict V4 activity well, often better than

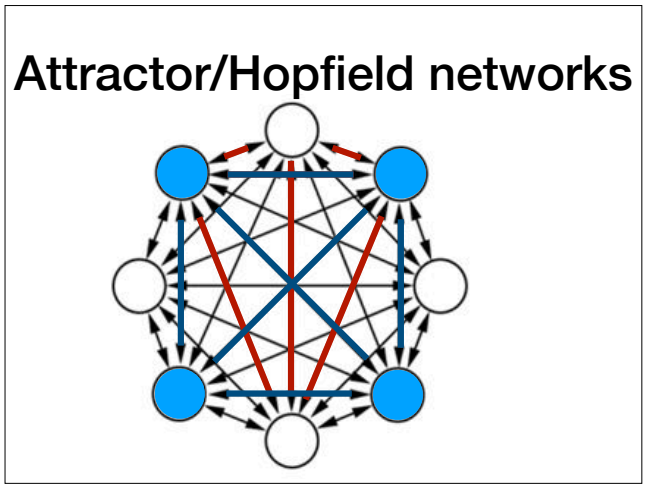
cornet. By IT, only two networks perform better, each with over 100 layers. In predicting human behaviour, only one network performs better.
As well as giving extra depth through recurrency, feedback connections seem to be carrying some useful information. But what does this information look like?



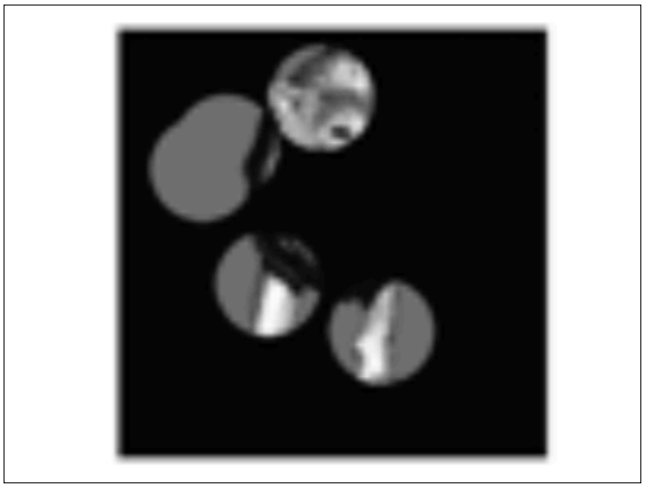
It has become clear that something very interesting is happening in the lateral (within-layer) connections in biological networks.
An incoming signal generates a pattern of neural activity in the higher layer.
Here, all of the neurons in this higher layer are connected to all others, as shown here in two common notations of the same structure.



The result of this organisation is that frequently seen patterns of activity produce strong connections between the activated group of neurons, due to Hebbian learning. The pattern of activity gets built into the connection weights.
After these common patterns are learned, a new incoming pattern will cause a sequence of recurrent activity that attracts the network's state towards a common pattern.
Neurons that are usually activated together in the higher layer all become active, completing the pattern of activity based on what they usually see.
So if an incomplete pattern comes in, it is completed based on experience.

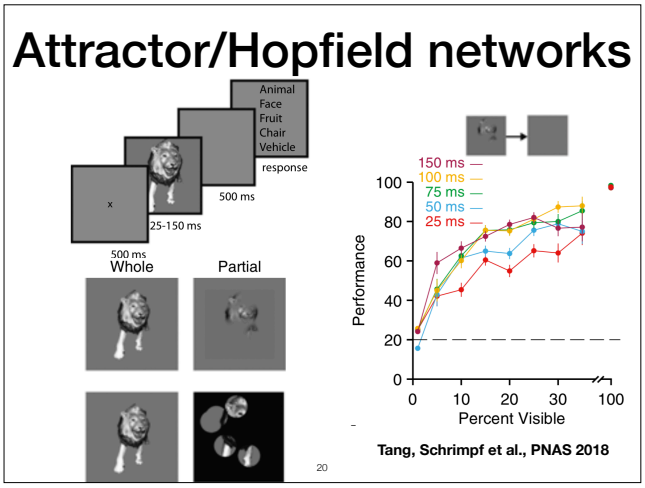


This also means that another pattern can be stored in the same layer using many of the same nodes and connections.
Here, two different units are activated, but one common with the previous T pattern.
That does not reach the threshold (2 active inputs) that would re-activate the T pattern, but it is enough to re-activate the square.

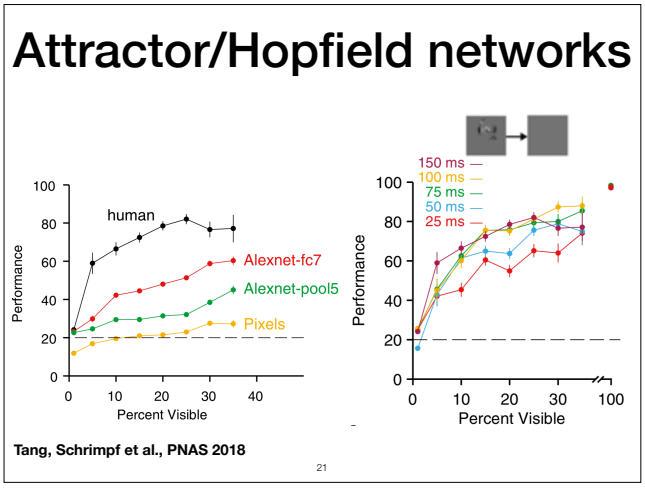


Humans are very good at completing partial inputs like this to recognise objects

---

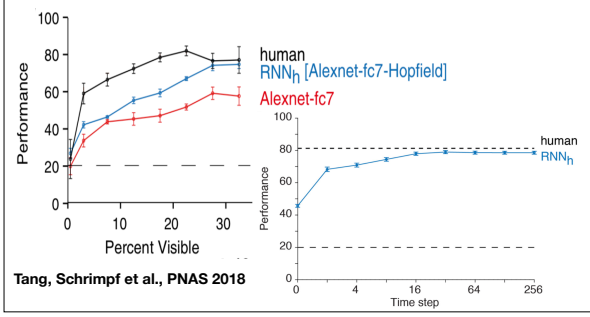


With only 30% visible and only presented for 100 ms, we are 80% correct at seeing that was an animal.



Feedforward networks don't do that: Their performance is poorer, and increases approximately linearly as more of the object is shown.

Attractor/Hopfield networks



Tang, Schrimpf et al., PNAS 2018

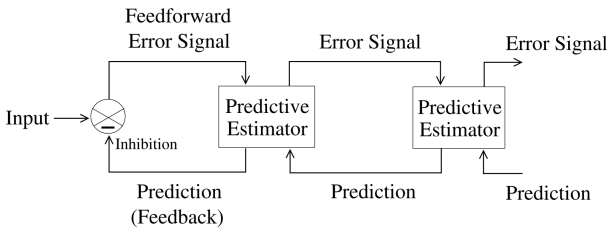
But a recurrent Hopfield attractor network does approach human performance at classifying partial images.
If we track how this performance emerges over different time steps, it relies on this recurrent interaction.

Attractor networks

- Recurrent connections effectively make networks deeper
  - Same analysis repeated by recurrent cycles
  - Each layer performs multiple layer operations
    - But each with the same set of weights
  - Matches neural architecture and activity more closely than deeper networks
- Attractor/Hopfield network among lateral connections
  - Attract activity patterns towards previously common states
  - Complete or predict noisy or incomplete input patterns
  - Requires multiple recurrent cycles

23

Predictive coding

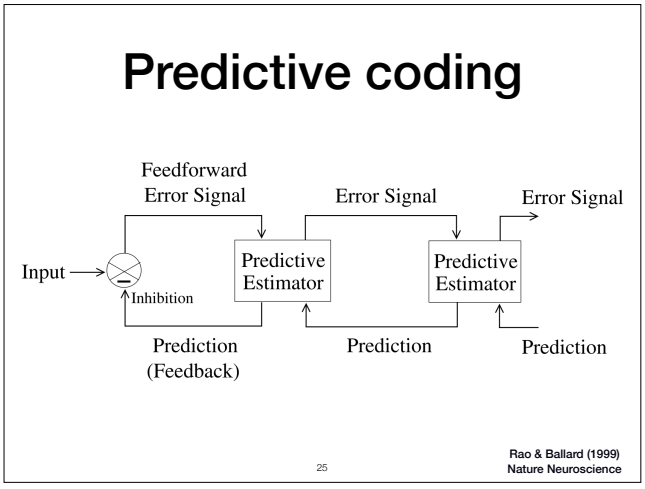


24

Rao & Ballard (1999)
Nature Neuroscience

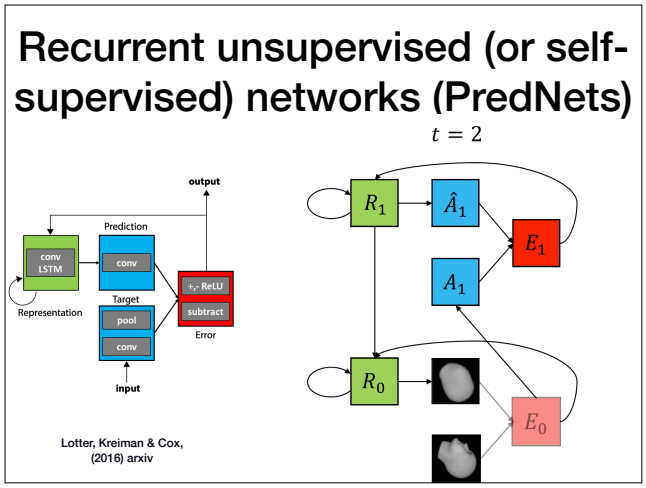
This completed pattern of activity based on experience is often described as a ‘prediction’ of what is coming in. Completing this prediction seems to be the important role of lateral connections.
But what about feed-BACK connections? Here, every feedforward connection seems to have an equivalent feedback connection.
This completed prediction is then

fed back to the lower layer, and INHIBITS activity that fits with this prediction.
As if the higher area was saying 'I think I recognise this, you can ignore it'.
This feedback connection therefore carries the prediction and allows it to interact with incoming signals.
Due to this specific inhibition of activity consistent with the prediction, the feedforward signal becomes only the parts that do not fit with the prediction: the prediction error.
This happens repeatedly in feedforward-feedback interactions between pairs of layers, though again it gets hard to think about beyond the first layer: it becomes the prediction error on the prediction error signal.



So the first feedforward sweep of activity encodes the incoming stimulus and learns common patterns in it, as we have already seen.
But the recurrent feedforward-feedback loops of activity instead represent the error between the incoming signal and what is expected from previous learning.
Because the activity in the network then responds to this error, Hebbian learning can

operate on the error signal: connection weights change depending on the prediction error.
This incorporates new patterns of activity into the predictions that can be made, by making strong connections for unexpected inputs.
This is remarkably similar to backpropagation of error. The main distinction is that the error is based on the difference from something the network has seen before, not the difference from the expected input classification. In other words, the error is analysed at every layer, not just determined from a final output.
In neuroscience, predictive coding and the related concept of Bayesian inference have been important concepts for the last 20 years.
Predictive coding is consistent with many unexpected aspects of neural activity that are difficult to explain without it.



A new type of recurrent networks, PredNets, have recently been introduced to implement predictive coding. These are unsupervised (or self-supervised) networks: they predict future activity based on current activity and previous experience, but no labels.

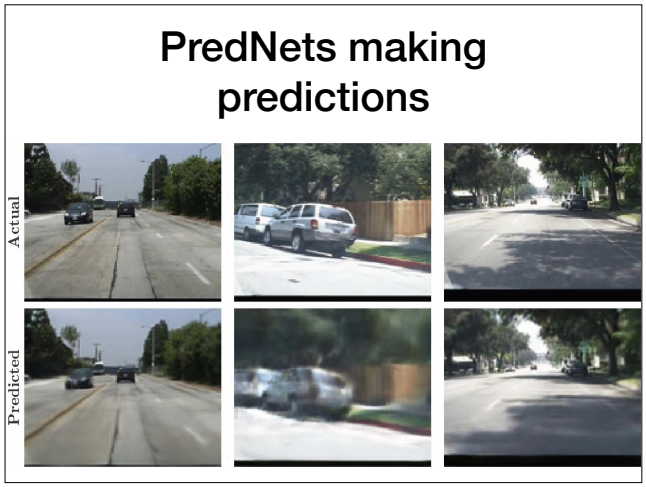
Let's see how this plays out in a simple 2-layer network, with layers 0 (where the input image comes in) and layer 1 (a higher layer).

The network initially has no input and so no activity. So the error state in layer 1 is effectively zero. This feeds into a recurrent circuit of layer 1, but nothing much happens, because the activity is all zero. This feeds into the recurrent circuit of layer zero which also takes an input from the empty error circuit of layer 1 (again little happens).

The prediction at this point is an empty image, but an image of a face comes in. The empty prediction is then subtracted from the face image, and so the image is passed forward unaltered in this first feedforward sweep.

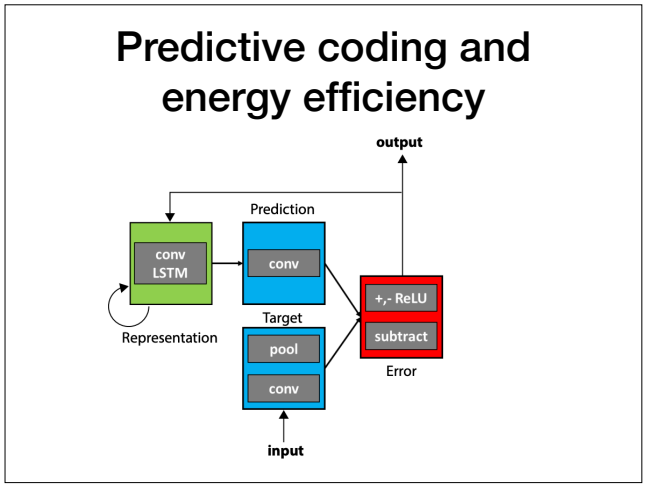
However, at this stage the network starts to generate predictions of what comes next, based on previous patterns of activity in the green attractor circuit that are similar to the incoming image. These get subtracted from the incoming

image, determining the difference and so allowing the prediction to be refined by further steps.

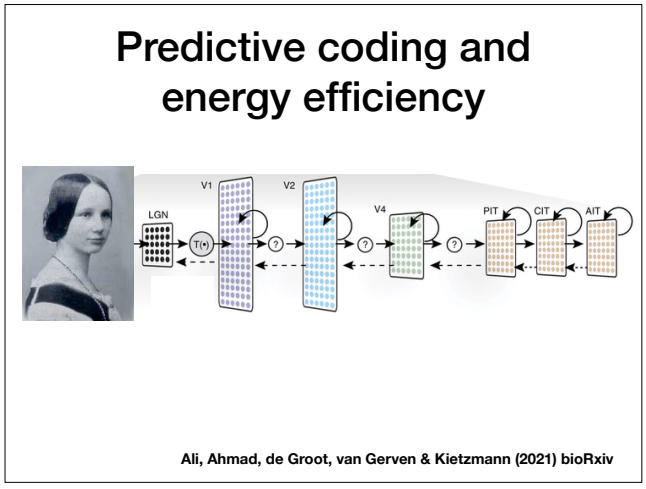


Here we see the layer 0 (image) prediction of the next frame in driving data.
Here the network was trained on a database of driving videos, and is making predictions for videos that it has never seen.
There is a very old criticism of AI systems, that they can only learn what they are taught or do what we can program them to do, and can never understand or create anything.
As Ada Lovelace said, “The Analytical Engine has no pretensions whatever to originate anything. It can do whatever we know how to order it to perform. It can follow analysis; but it has no power of anticipating any analytical relations or truths.”
But to make predictions about what will happen in the future, we need an internal model of the world, an understanding.
This is how human learn: we develop expectations. A child

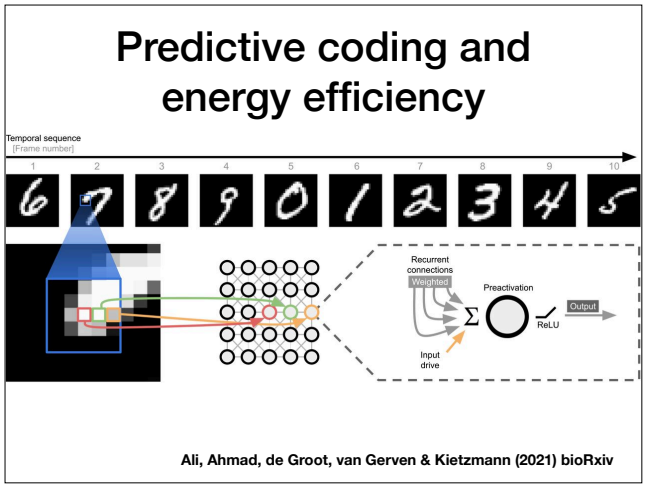
might drop an object to test their exception it will fall. A scientist might perform an experiment to test a hypothesis they have, seeing whether the outcome matches the prediction of their hypothesis, their internal understanding of the world.



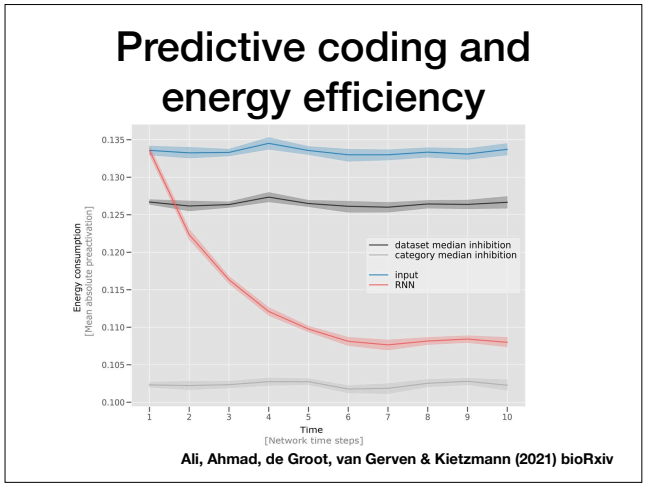
Typically, the organisation of a network is hard wired to achieve predictive coding by specifying convolutional operations, attractor layer (or predictor units) and units that compare the outputs of these operations (error units).



But we can make a network that is simply designed to have lots of layers, each with a full set of internal connections and feedforward and feedback convolution.
Here, each layer has a broad set of connections, and can do lots of different thing: in is not designed to do predictive coding.

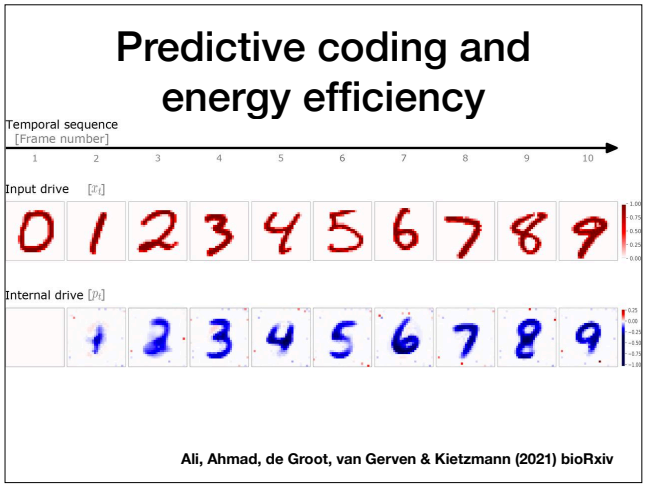


Here the researchers then give that network a predictable input, here the mnist numbers increasing every frame, so that the next image is completely different but predictable from the current image.
Rather than training on accurately identifying that image number, or minimising the difference between the predicted and input images, the researchers simply train it to minimise the summed activation of its inputs.
In biological terms, it is being trained to minimise the generation of postsynaptic potentials.
In other words, it is trained to minimise the generation of network activity, including the responses in the first layer to an image being placed there. So it can't just set everything to zero because that input image will produce a lot of activation.



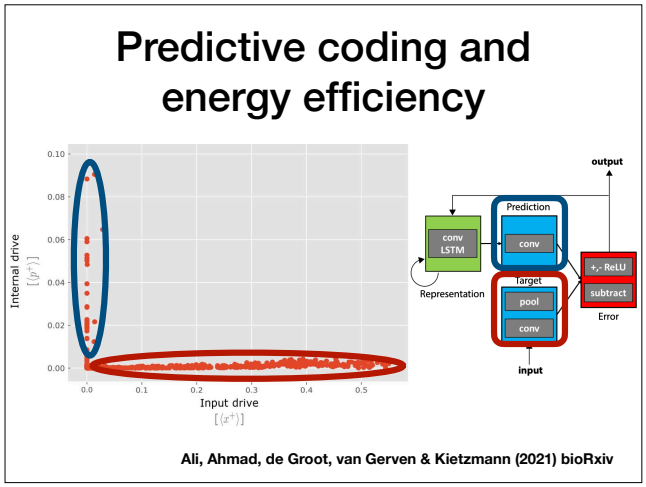
Here we can see that this training is effective: after the new sequence is input, the network reduces energy consumption as it gets more information about its current place in the sequence (red).
It starts from the amount of energy the input activity would consume (blue), and moves towards full inhibition of the average image of this number (category median inhibition,

grey).
Even by the second time point, it is making a specific prediction of this number, as it is consuming less energy than it would if inhibiting the average image of all numbers (black).

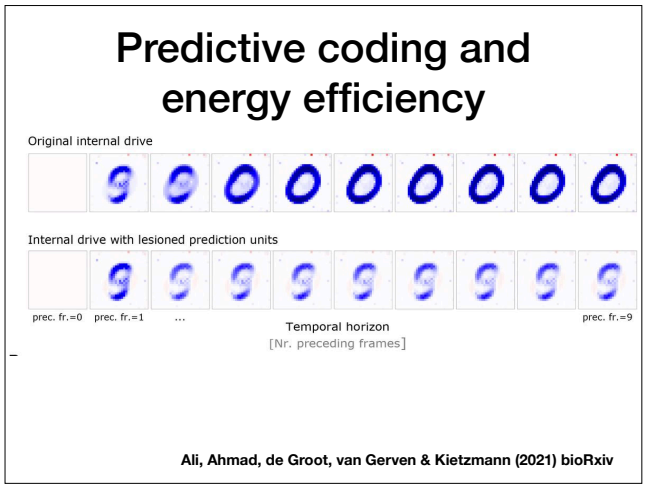


So once the network knows where in the sequence it is starting, it starts generating accurate predictions of what number is next (internal drive), and generating images of the average example image for that number.
As more time steps pass, that predictions gets clearer. These better predictions are more effective in inhibiting input-driven network activity.
Note here that the network architecture was not set up for predictive coding, but that is what is happening.
Also, the network was not optimised to predict the next image, though learning to minimise pre-activation is very similar in the first layer: a good match between the prediction and input will minimise pre-activation.
Of course, it is input with a specific example image of the

number, while the prediction follows an average image of the number, but the average has many pixels in common with the example.



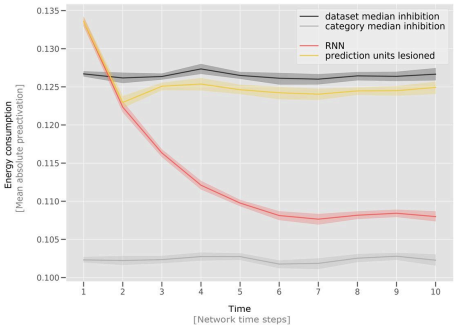
The activity of most units in the trained network depend on the input signal.
But other units show activity that depends on the network's internal state, with no relation to the input signal.
So, even though it isn't hard-wired to make prediction units, these emerge to minimise network activity.



Having identified these prediction units, we can turn them off, by setting all their weights to zero after training.
Here we see the inhibition of the image input when different numbers of previous images have been shown, and in each case the last image was a 9 and the next will be 0.
In the fully-functioning network, the next inhibition of the image layer quickly corresponds to a

zero.
But without working prediction units, that inhibition stops being able to predict upcoming frames, so the the inhibition always stays on what the last frame looked like.

Predictive coding and energy efficiency

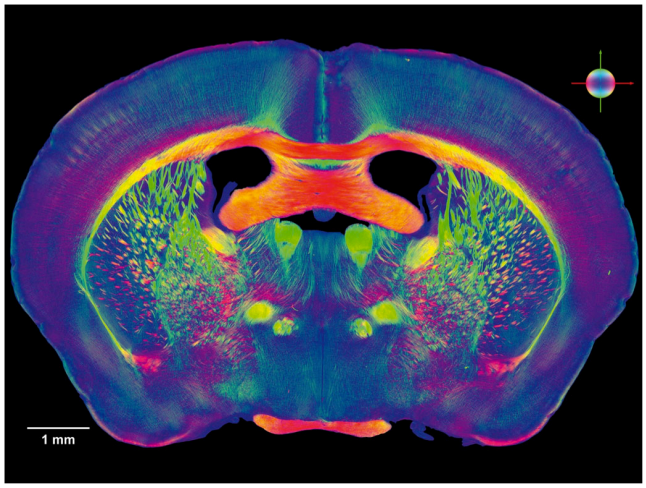


Ali, Ahmad, de Groot, van Gerven & Kietzmann (2021) bioRxiv

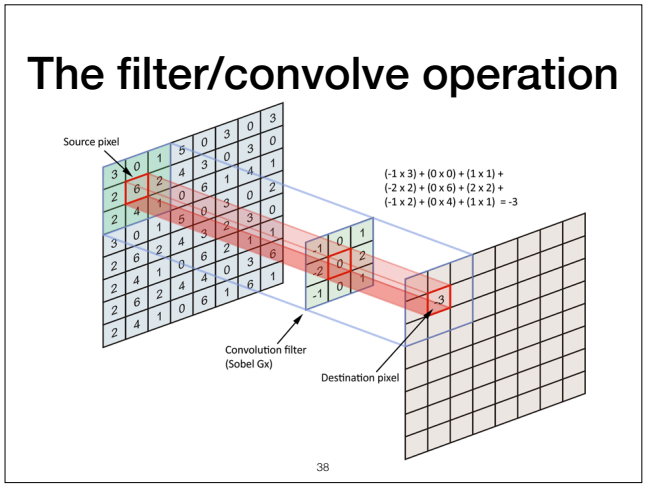
So then this network without functional prediction units stops being able to make effective inhibition of input activity and stops being able to minimise its energy use properly (yellow).

Predictive coding

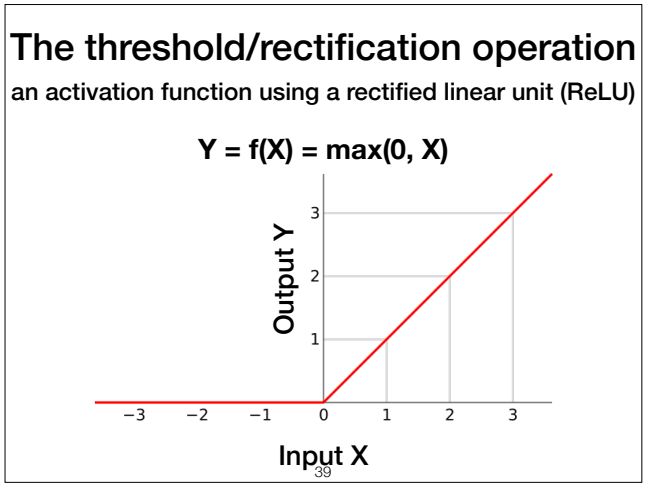
- Predictions of incomplete or upcoming inputs generated in attractor/ Hopfield network layers (lateral connections)
- Fed back to inhibit input activity that matches prediction
  - Initial feedforward sweep that lacks a prediction, extracts image features
  - Prediction generated based on previously seen patterns
  - Inhibition determines difference between prediction and input
    - Then remaining activity signals prediction error
    - Hebbian learning follows this error signal, like backpropagation
- Accurately predicts upcoming images in input
- This network architecture emerges spontaneously if all the connection are available
  - Without blocks explicitly designed to generate predictions
  - Training to minimise pre-activation
  - Minimising energy use is very similar to predicting next input
  - The brain may implement predictive coding to reduce energy demands[36]



The Blue Brain project, running for the last 15 years, recently finished mapping all the neural connections in the whole brain of a transgenic mouse clone.
-The project simulates the processes underlying neural responses at the molecular level, rather than just a single number representing each neuron's activity.



So far, we have looked at a very efficient simulation of synapses and postsynaptic integration of neural activity, which is obviously chosen for speed rather than accurately simulating the underlying processes.



Similarly, the thresholding is a very simple operation that can be achieved quickly over a whole feature map with a single matrix operation. Again, chosen for speed rather than accuracy.

Bluebrain doesn't aim to use this highly efficient, oversimplified approach, it aims to make an accurate simulation of a brain.

Neuron (software toolbox)

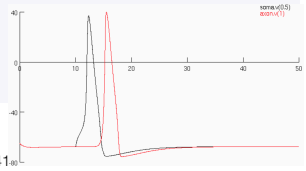
```
//create two sections, the body of the neuron and a very long axon
create soma, axon

soma {
    //length is set to 100 micrometers
    L = 100
    //diameter is set to 100 micrometers
    diam = 100
    //insert a mechanism simulating the standard squid Hodgkin–Huxley channels
    insert hh
    //insert a mechanism simulating the passive membrane properties
    insert pas
}
axon {
    L = 5000
    diam = 10
    insert hh
    insert pas
    //the axon shall be simulated using 10 compartments. By default a single
compartment is used
    nseg = 10
}
```

40

It allows the user to specify the properties of the simulated neuron directly, and bluebrain uses the neural parameters and connection patterns actually measured in the mouse brain.

Neuron (software toolbox)

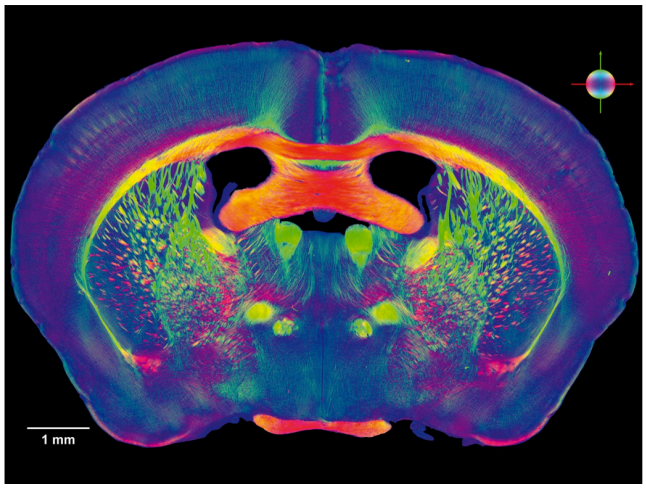
```
//connect the proximal end of the axon to the distal end of the soma
connect axon(0), soma(1)

//declare and insert a current clamp into the middle of the soma
objref stim
soma stim = new IClamp(0.5)

//define some parameters of the stimulus: delay, duration (both in ms) and
amplitude (in nA)
stim.del = 10
stim.dur = 5
stim.amp = 10

//load a default NEURON library file that defines the run routine
load_file("stdrun.hoc")
//set the simulation to run for 50 ms
tstop = 50

//run the simulation
run()
```



41

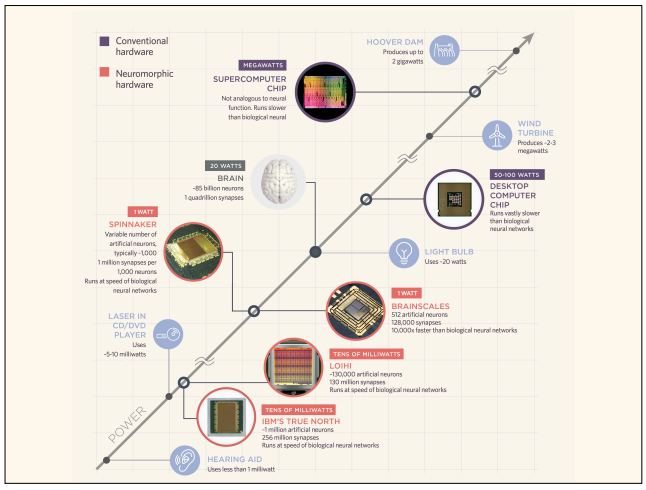
This gives a time-resolved prediction of membrane potentials for each neural compartment

---



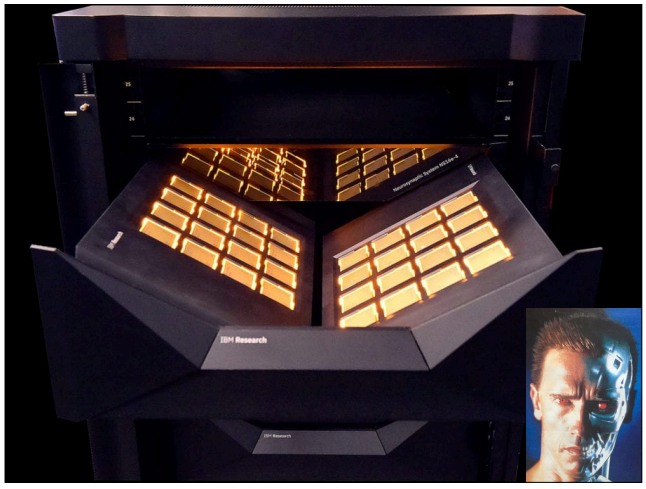
-So if we know the properties and connections of every mouse neuron, we should be able to make a reasonable simulation of a mouse brain.
-Taking a similar approach, we might use the brains of a few thousand genetically engineered human clones to make a similar map of the human brain.
-But in humans, we don't make armies of genetically engineered clones, fill their lives with

experiences and then experiment on their brains (like we do with mice).
-The second problem for the Blue Brain is that the computing power needed to run the simulation of even the mouse brain is not available.
-As a result, a major part of the current Blue Brain project is to get it running as a simplified deep network based on matrix operations.
-These are both practical problems: the first is ethical and the second technological.



The technological problem is being tackled.
Most computer processors are designed for linear mathematics. While they can perform any operation eventually, they are poorly suited to neural simulations: they are slower and more energy-hungry than the brain.
Neuromorphic processors are designed specifically to run neural simulations.

The current state of the art (2019) can simulate 8 million neurons with 2 billion synapses on a single chip, and link 64 of these chips in one computer.
This uses less than 1% of the power of an equivalent conventional graphics processing unit.
To match the human brain, we would need around 10,000 of these chips working together.
This should be possible with only 100 processors that have scaled following Moore's law in about 10 years
So in 10 years we may well match the processing power of the human brain in one desktop-sized computer, or Arnold Schwarzenegger's head, with feasible energy needs.



This is an array of IBM's TrueNorth neurotrophic chips, worryingly designed to pilot drones for the US military. What could possibly go wrong?

Simulating biological brains

- Neuroscience has revealed the basic principles of neural responses and their interactions
  - Recurrent interactions, attractor networks, predictive coding, Hebbian learning from prediction errors
- It is straightforward to implement these in computers
  - Convolution filters can operate on lower, upper and the same layer
  - Simulating feedforward, feedback and lateral connections
  - Makes predictions about external events, some characteristics of biological understanding and intelligence
- The full neural circuitry of mice has been mapped
  - Can be implemented in more realistic biophysical simulation of interacting neurons
  - This simulates neural circuitry accurately, but is slower
    - So currently being implemented in simpler neural models
  - Running these biophysical simulations in real time requires neuromorphic computing technology, developing quickly.

Exam structure

- We cover a lot of details, so you can follow everything from first principles
- The take-home messages are summarised in text slides at the end of each lecture section
  - These are the basis of the exams
  - We will ask you to explain points made on these slides
  - Grades reflect the level of relevant detail and specificity in your answers

A summary slide

- Neurotransmitters (e.g. glutamate and GABA) released from a pre-synaptic cell can excite (depolarise, EPSP) or inhibit (hyperpolarise, IPSP) activity in the post-synaptic neuron
  - This relies on ligand-gated (i.e. neurotransmitter activated) ion channels
- If membrane polarisation reaches a threshold, voltage-gated ion channels open
  - Results from many excitatory inputs
  - Strongly depolarises (fires) the neuron: Action potential