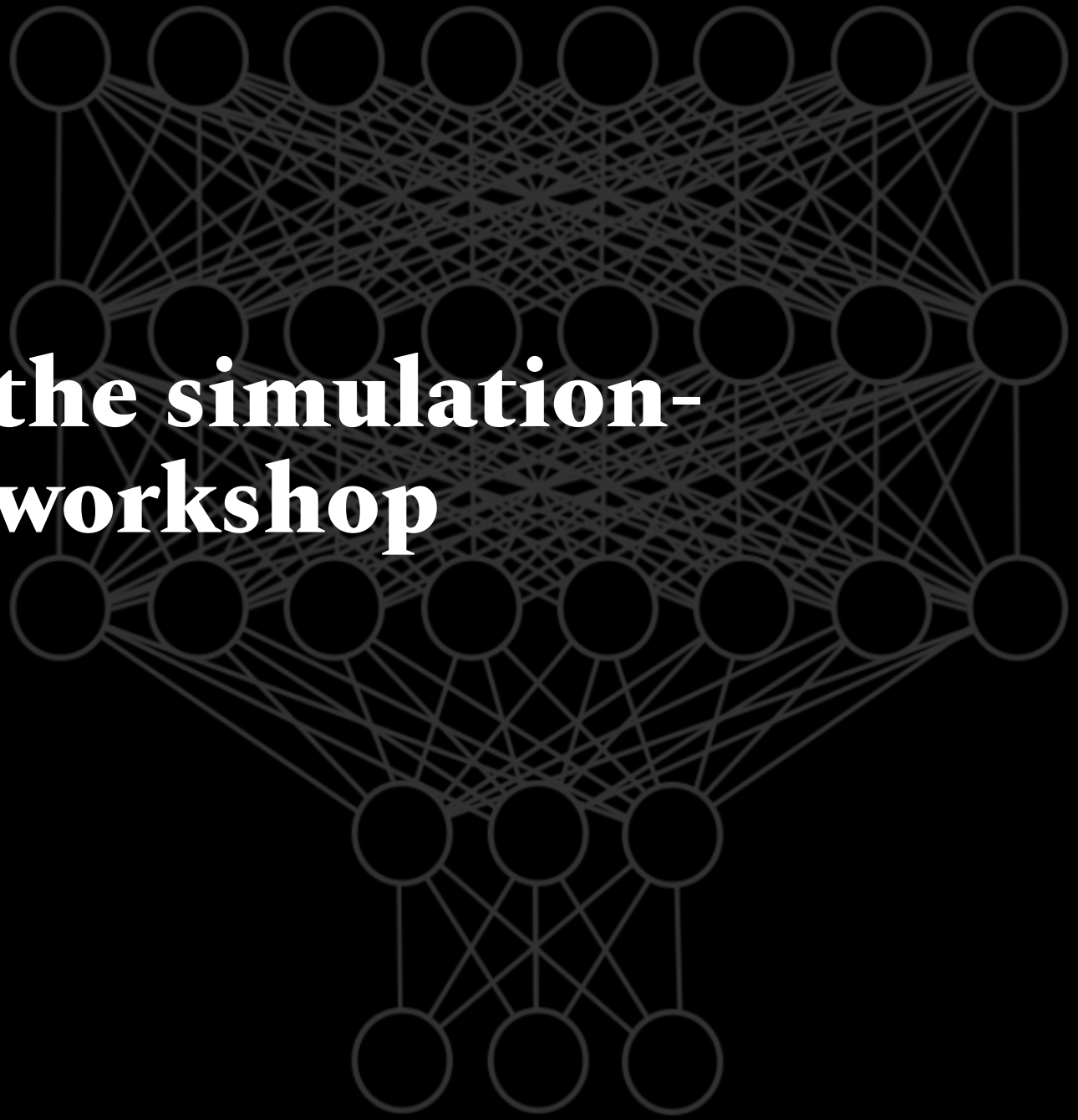


Introduction to the simulation-based inference workshop

Peter Kvam

The Ohio State University

Agenda

- Machine learning fundamentals
  - What are AI, ML, neural networks, and deep learning?
  - Why are they so successful at a variety of tasks?
- Applications and special additions
  - Images / videos, text, sequential data
  - Problems with no correct solution (unsupervised learning)
  - Generative networks
- Potential uses in decision psychology
  - Model fitting, comparison, discovery
  - Additional problems - posterior sampling, input structures

Agenda

- Machine learning fundamentals
  - What are AI, ML, neural networks, and deep learning?
  - Why are they so successful at a variety of tasks?
- Applications and special additions
  - Images / videos, text, sequential data
  - Problems with no correct solution (unsupervised learning)
  - Generative networks
- Potential uses in mathematical psychology
  - Model fitting, comparison, discovery
  - Simulation studies of learning, evolution

Defining AI & ML

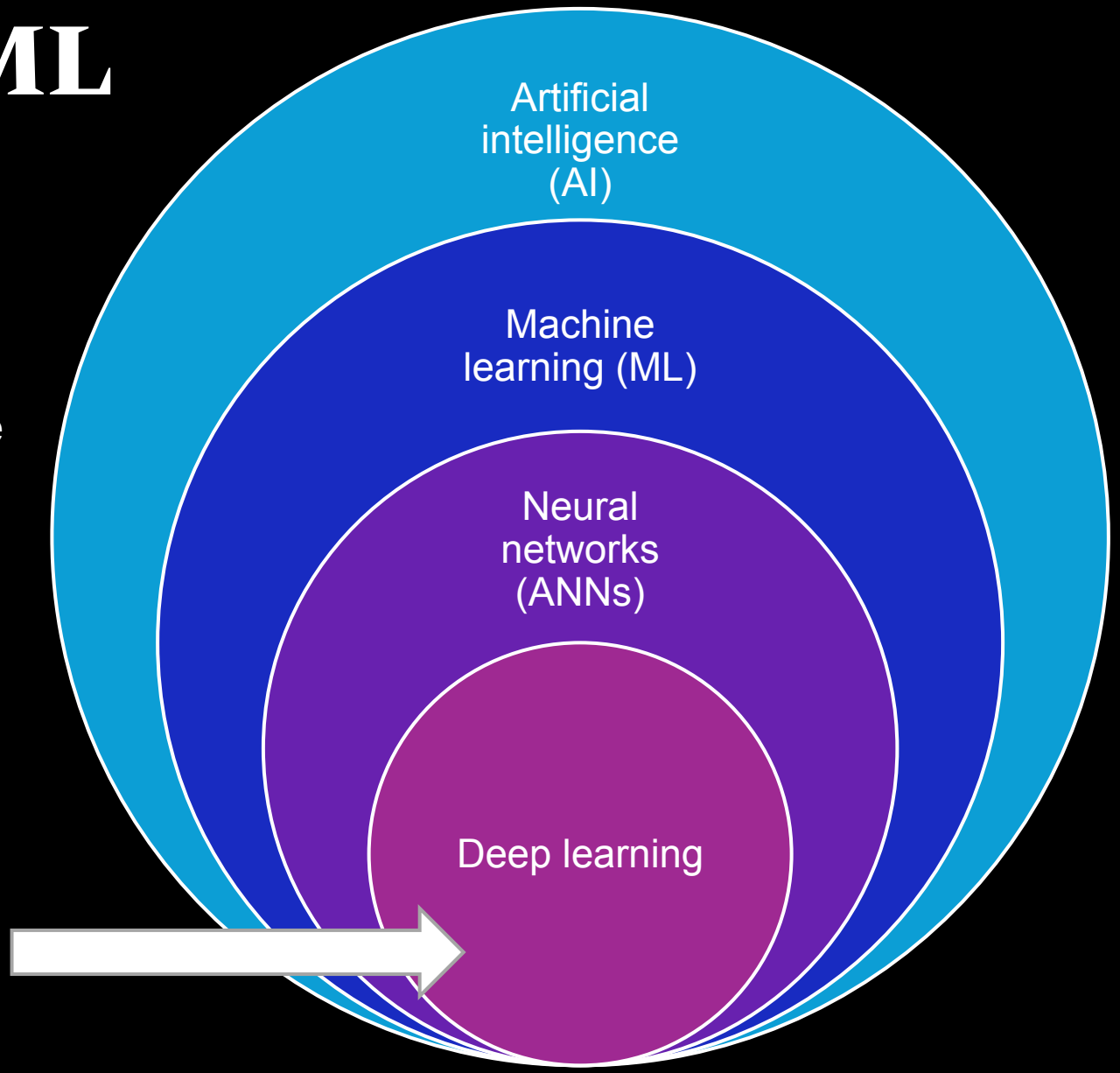
AI: machines that can mimic (some) human tasks or behaviors

ML: A special type of AI where a computer learns to do something outside its direct programming

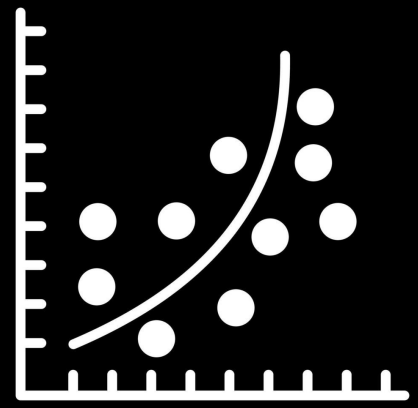
Neural networks: A special type of ML using brain-inspired structures of "neurons" / nodes and connections

Deep learning: Neural networks with multiple layers of nodes

Most applications you see these days fall down here



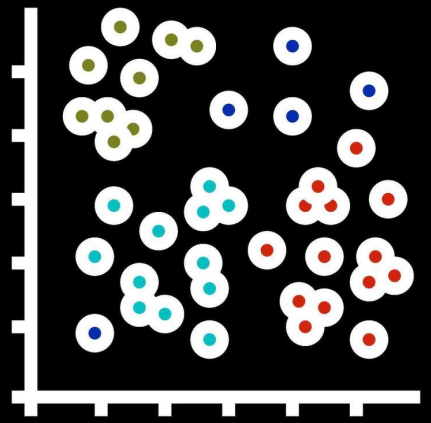
Flavors of machine learning



Prediction

Supervised learning

Approximates the relationship between variables based on known pairs of values



Discovery

Unsupervised learning

An algorithm creates a structure to organize data based on their quantitative attributes

Exploration

Reinforcement learning

An agent interacts with its environment to determine action-response mappings from experience

Regression

- Regression uses inputs (regressors / predictors, b) to predict some output (outcomes, c) using weights (a) and sums:

$$c = a_1 * b_1 + a_2 * b_2 + a_3 * b_3$$

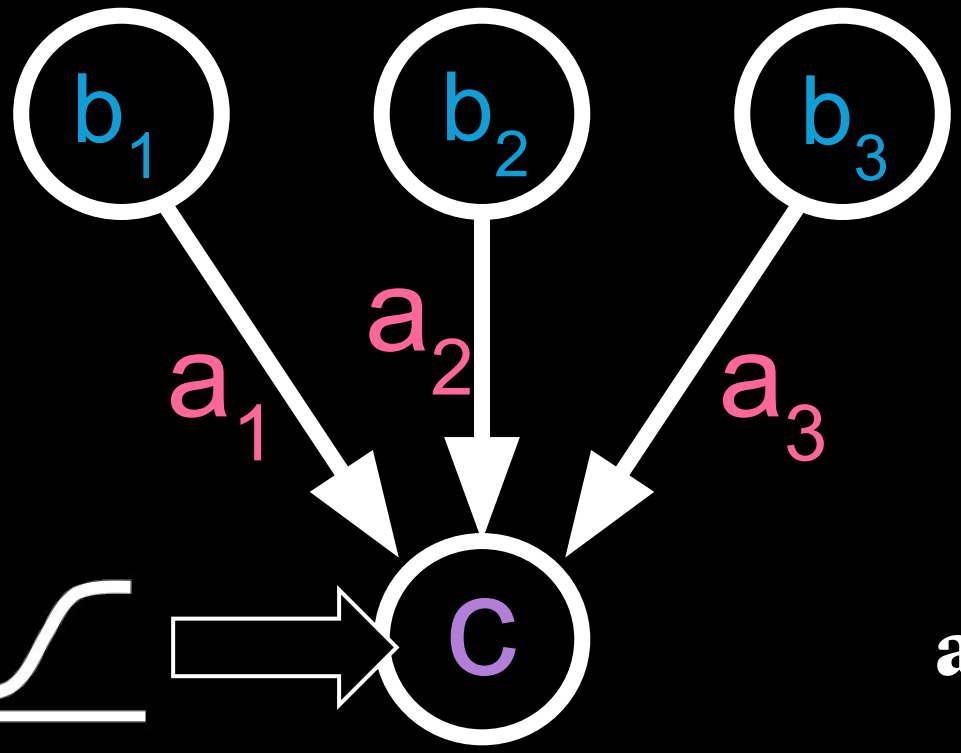
- In matrix form, it might look like this

$$C = A * B \quad A = [a_1 \quad a_2 \quad \quad B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Regression

$$c = a_1 * b_1 + a_2 * b_2 + a_3 * b_3$$

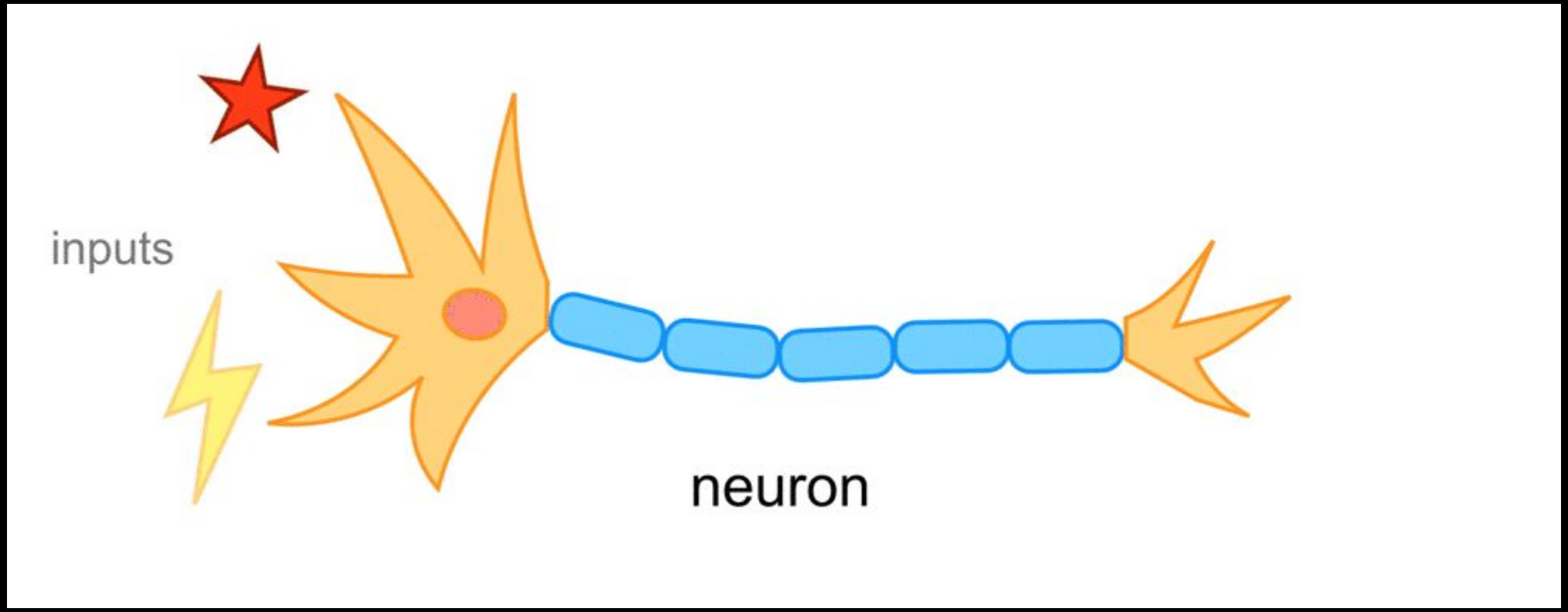
- Put visually, regression might look something like this:



In logistic regression, you might transform this outcome variable

This is now an artificial neural network!

Inspired by neural systems



But: they use a very simplistic picture, and now
ANNs are diverging quite sharply from real anatomy!

Scaling it up

- Start by adding another outcome – maybe we're interested in two things!
- We have our predictors b, weights a, and outcomes c
  - Now have two indices because we have two outcomes
  - Sorry for flipping a & b!
- Again can be represented by equation, matrix, or network

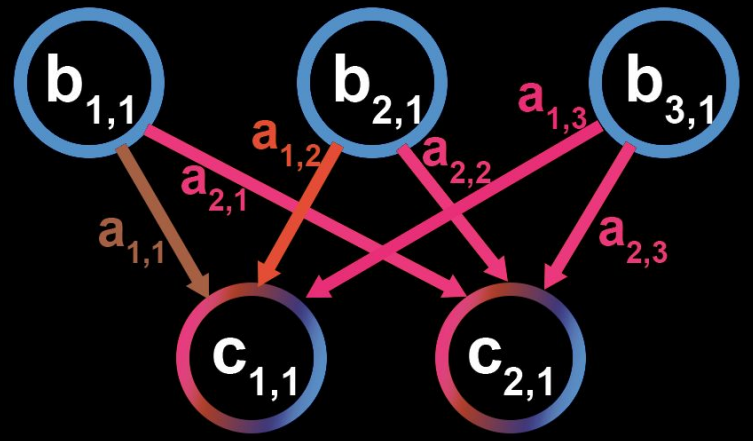
Equation form

$$c_{1,1} = a_{1,1}*b_{1,1} + a_{1,2}*b_{2,1} + a_{1,3}*b_{3,1}$$
$$c_{2,1} = a_{2,1}*b_{1,1} + a_{2,2}*b_{2,1} + a_{2,3}*b_{3,1}$$

Matrix form

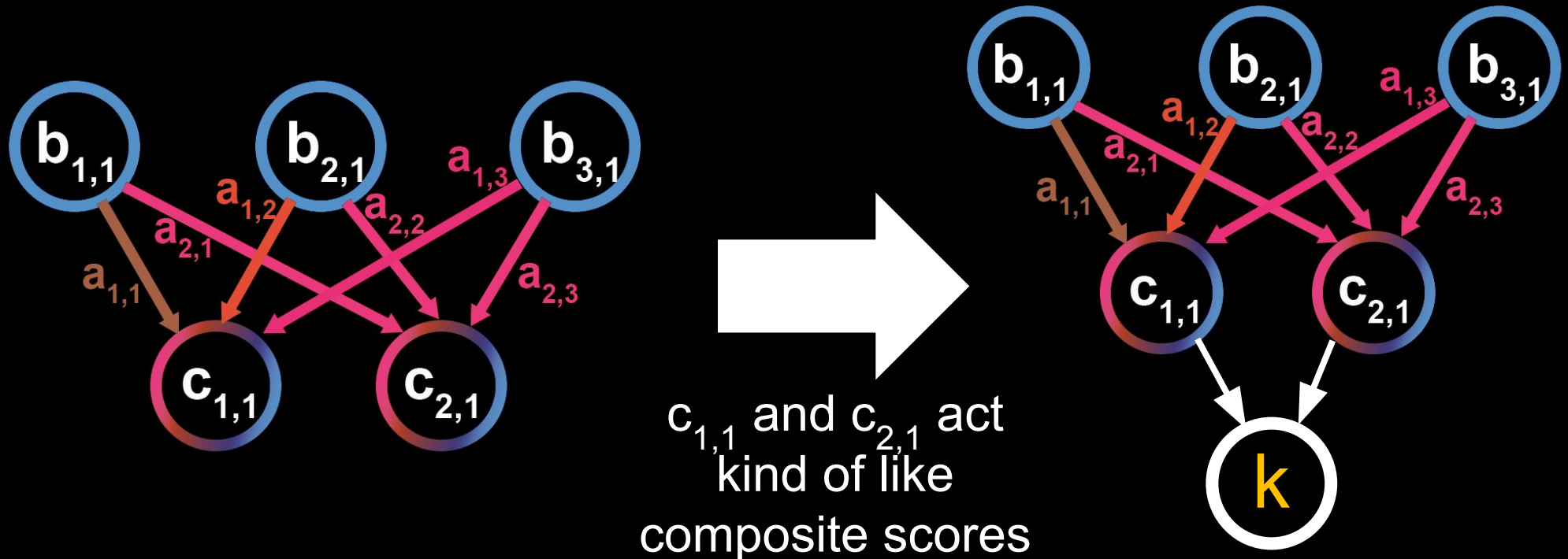
$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix} \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{bmatrix} = \begin{bmatrix} c_{1,1} \\ c_{2,1} \end{bmatrix}$$

Network form



Information in the network

- The nodes c1 and c2 contain some (weighted, summed) information about the inputs
  - Can we use these to better predict another outcome k?



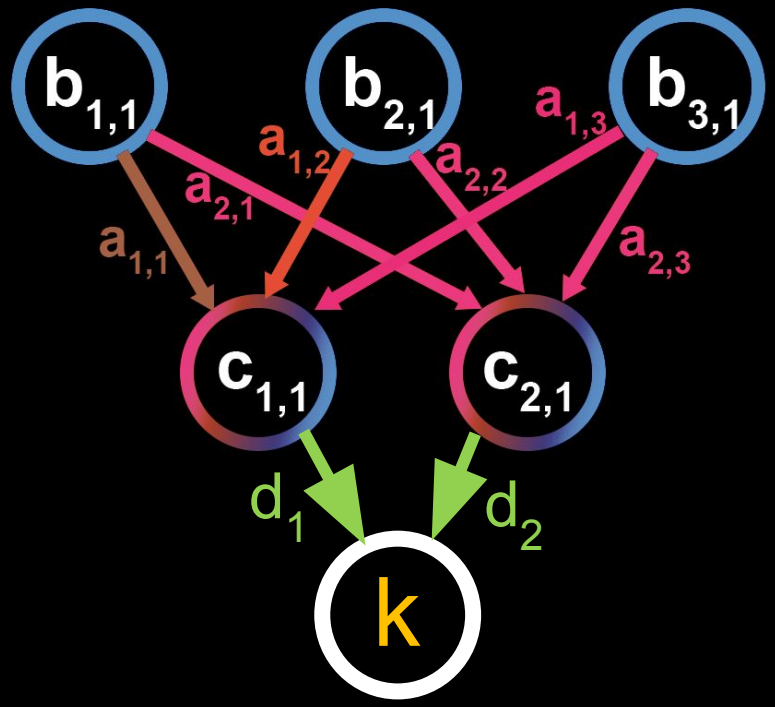
Problem!

- Without activation functions, the outcome k can just as easily be predicted by the initial inputs/ predictors (a)
- In matrix form:

$$C = A*B$$
$$K = D*C$$

$$K = \underbrace{D*A}*B$$

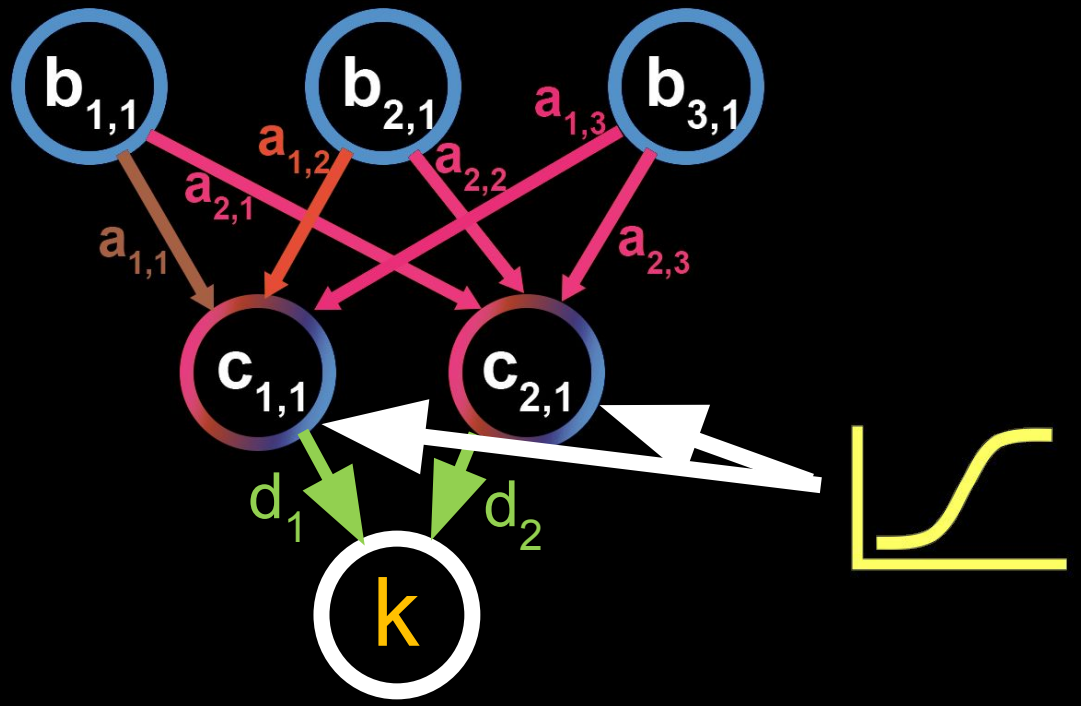
$$K = W*B$$



This is equivalent to a single-layer network – we don't need the composites C!

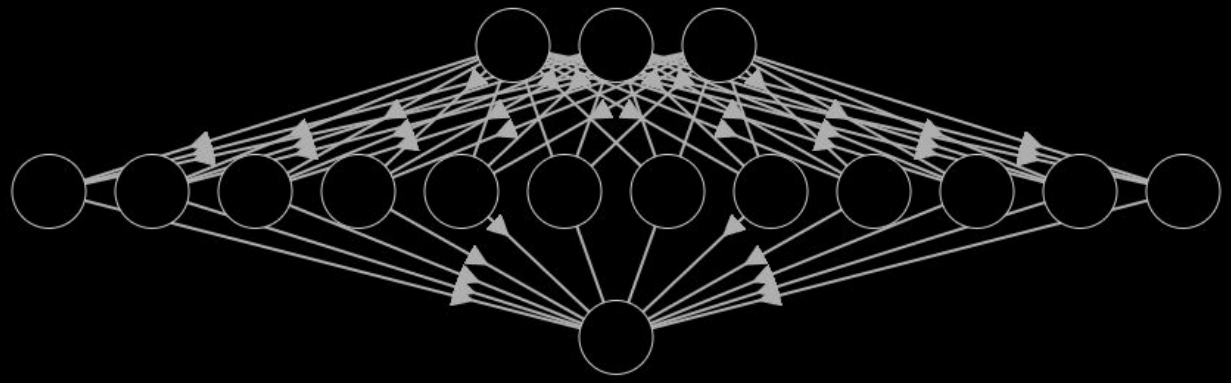
Activation functions

- This can be saved by carrying out some transformation of the composite variables c
  - Makes it so the weight matrices a and d can't just be combined
- This is called an activation function applied to a hidden layer (c)

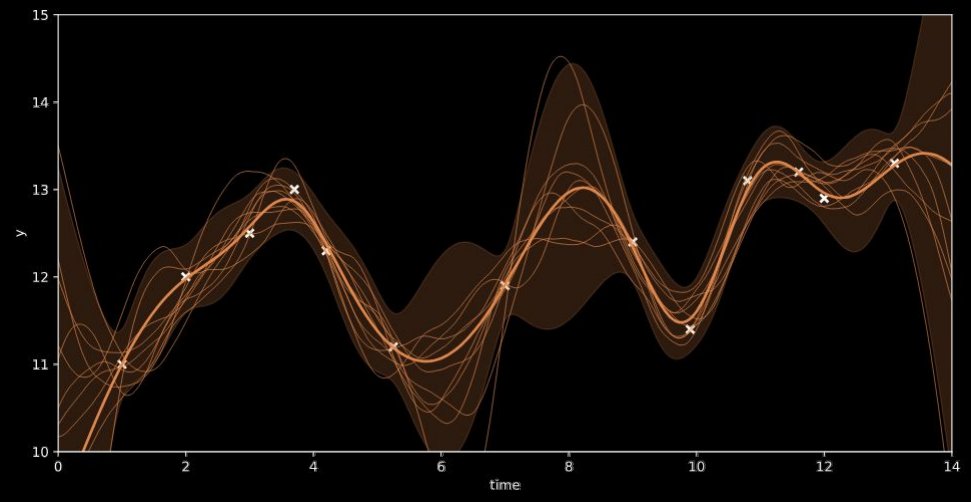


Universal approximation

- It is possible to prove that with enough nodes (c values) in the hidden layer, you can approximate any smooth function relating your predictors to your outcomes
  - Called the universal approximation theorem (Cybenko, 1989)
  - Mimics a Gaussian process

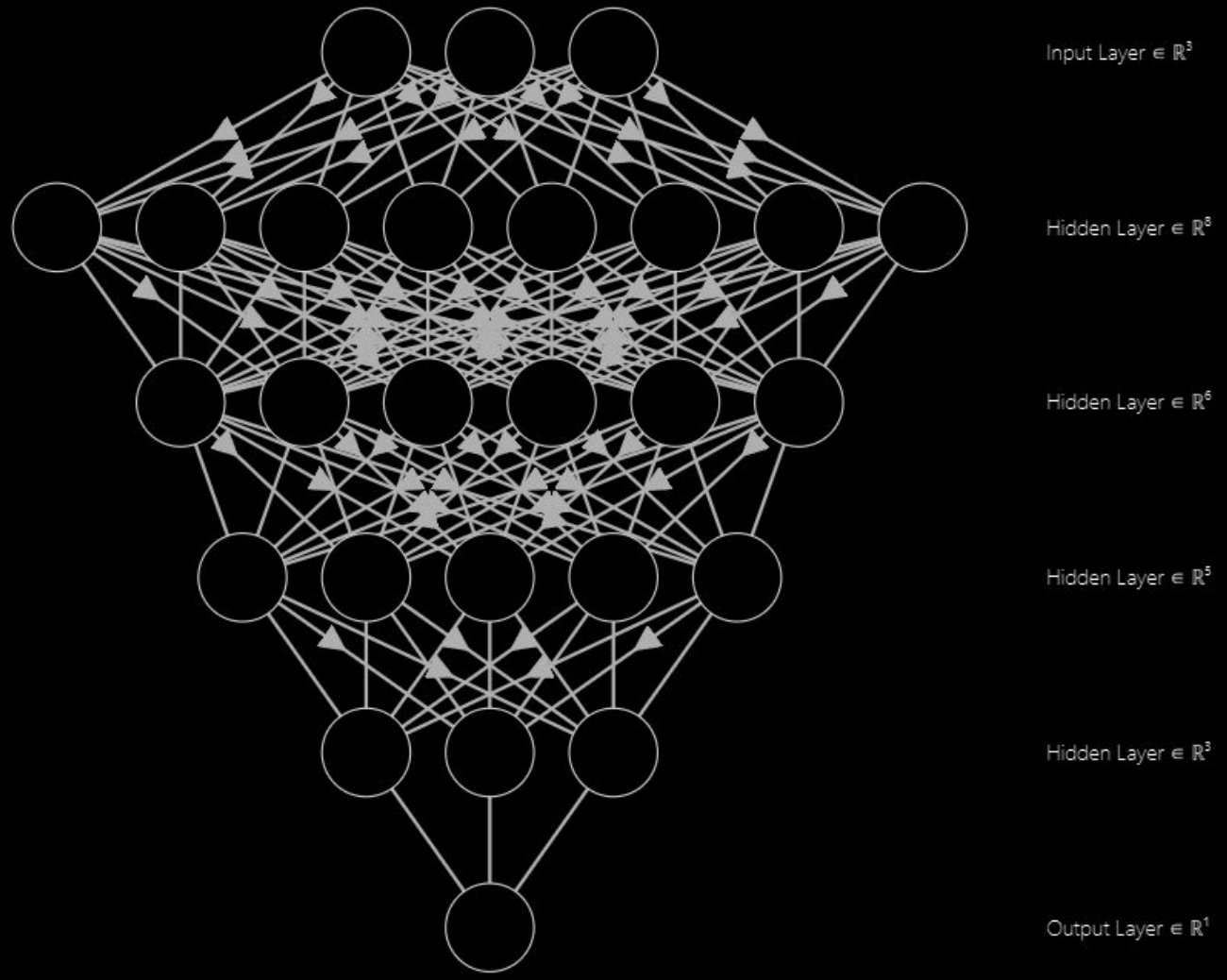


~



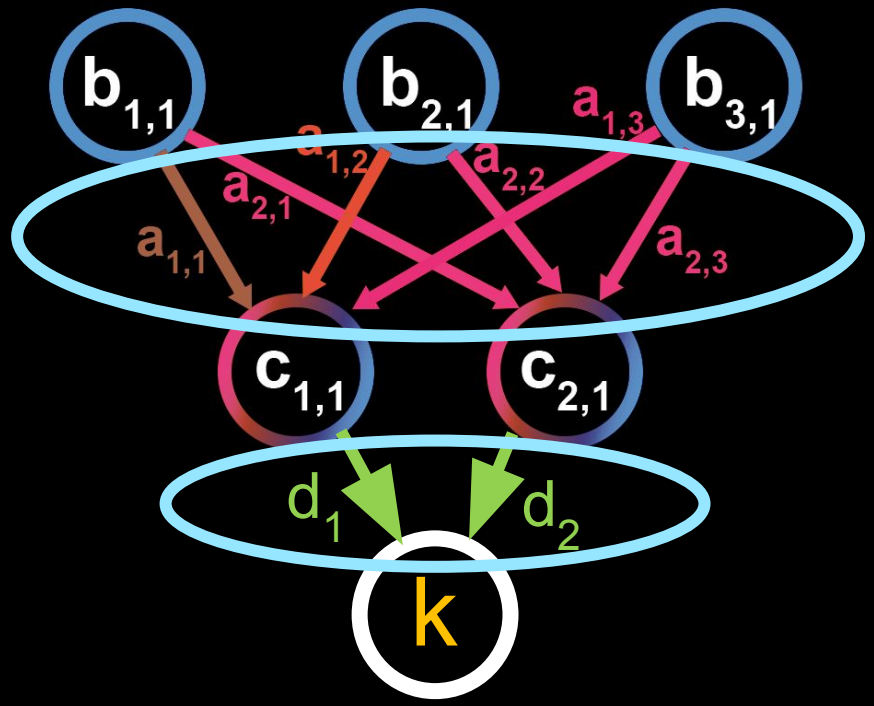
Deep neural networks

- Later, it was also proven that this could be done (often more efficiently) with multiple hidden layers
  - Gripenberg, 2003
  - Subsequent results suggested it could be done with relatively few layers and nodes
- This property ultimately led to deep neural networks becoming *the* dominant machine learning approach



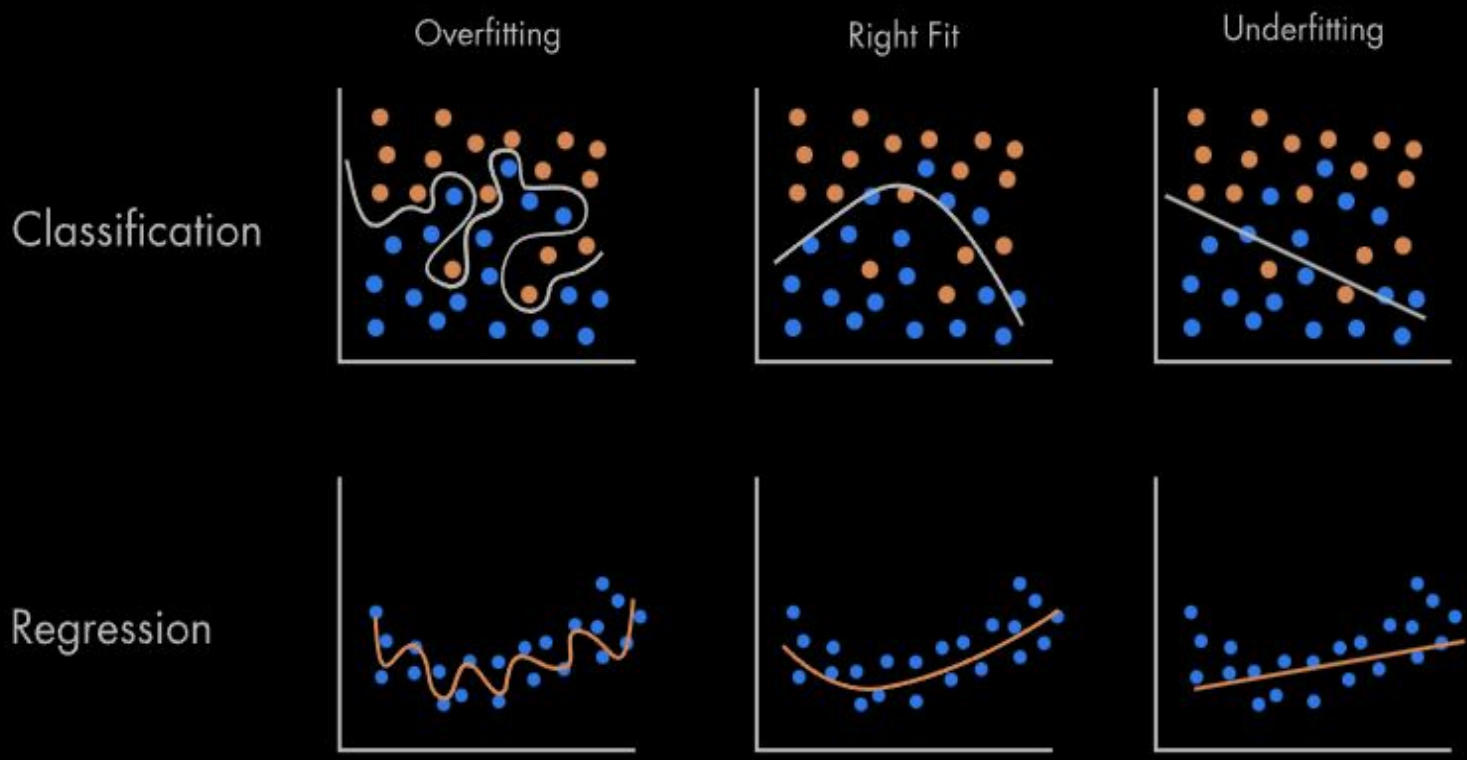
Training a neural network

- The next step is to teach your neural network what the relationships between predictors and outcomes are
- Estimate **weights** in the network
  - Values of all the lines connecting layers
- Try to minimize a **loss function**
  - How bad is your fit to the real data?
  - Often just mean squared error (L2)
  - Optimizers for a variety of losses
    - Cross-entropy (classification)
    - Mutual information
    - Absolute error (L1)



Testing a neural network

- Next, we see if it does what we want it to do
  - Typically scored via **cross-validation** to check for things like **over-fitting**
  - Does it do well in training but not held-out data?
  - Model and parameter recovery studies



Agenda

- Machine learning fundamentals
  - What are AI, ML, neural networks, and deep learning?
  - Why are they so successful at a variety of tasks?
- Applications and special additions
  - Images / videos, text, sequential data
  - Problems with no correct solution (unsupervised learning)
  - Generative adversarial networks
- Potential uses in mathematical psychology
  - Model fitting, comparison, discovery
  - Simulation studies of learning, evolution

Encoding & decoding

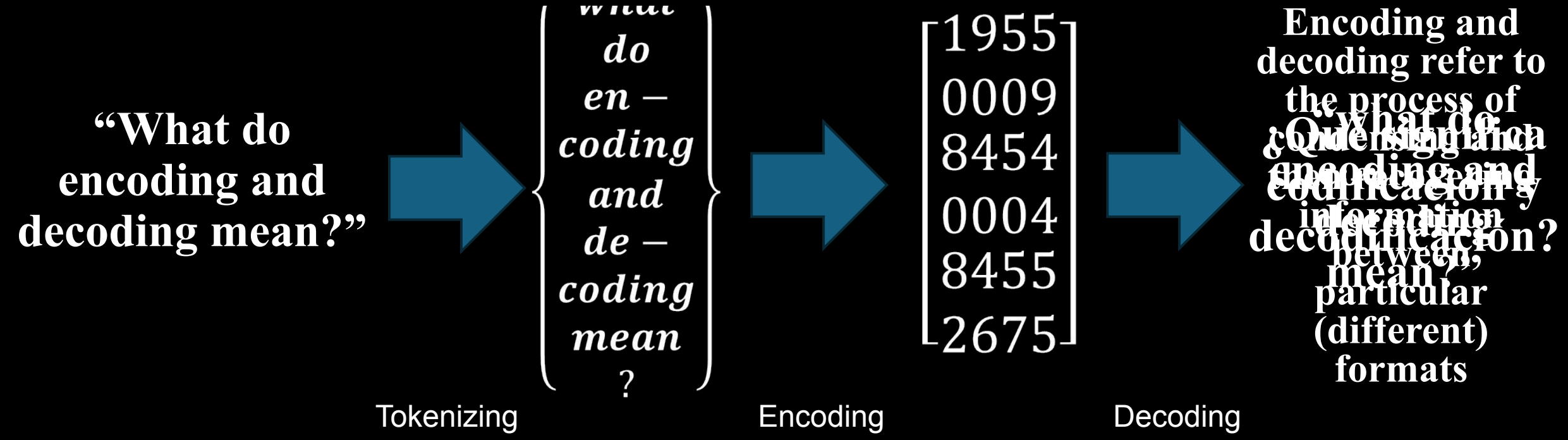
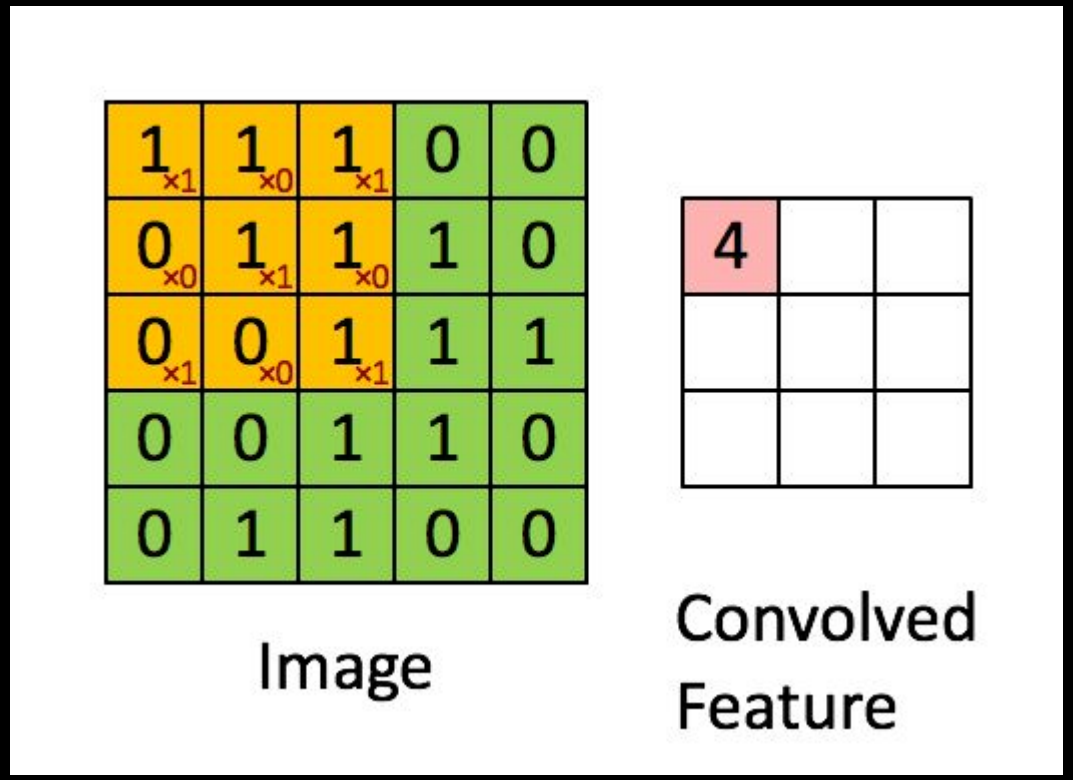


Image data - convolution



For example:

Input image

Convolution Kernel

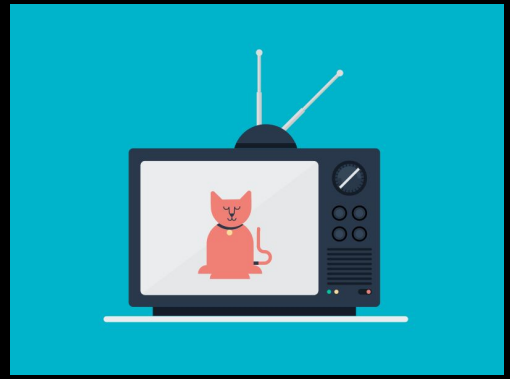
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

Process RGB pixels into (often smaller) data formats that can go into a neural network

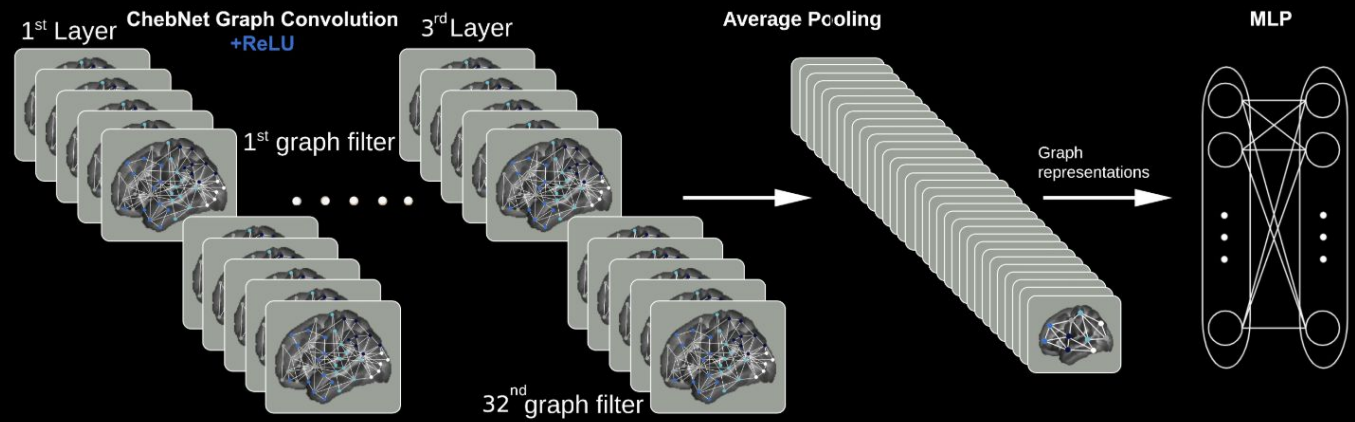
Convolution

- Convolutional, pooling, normalization layers allow the image to be represented efficiently as numbers



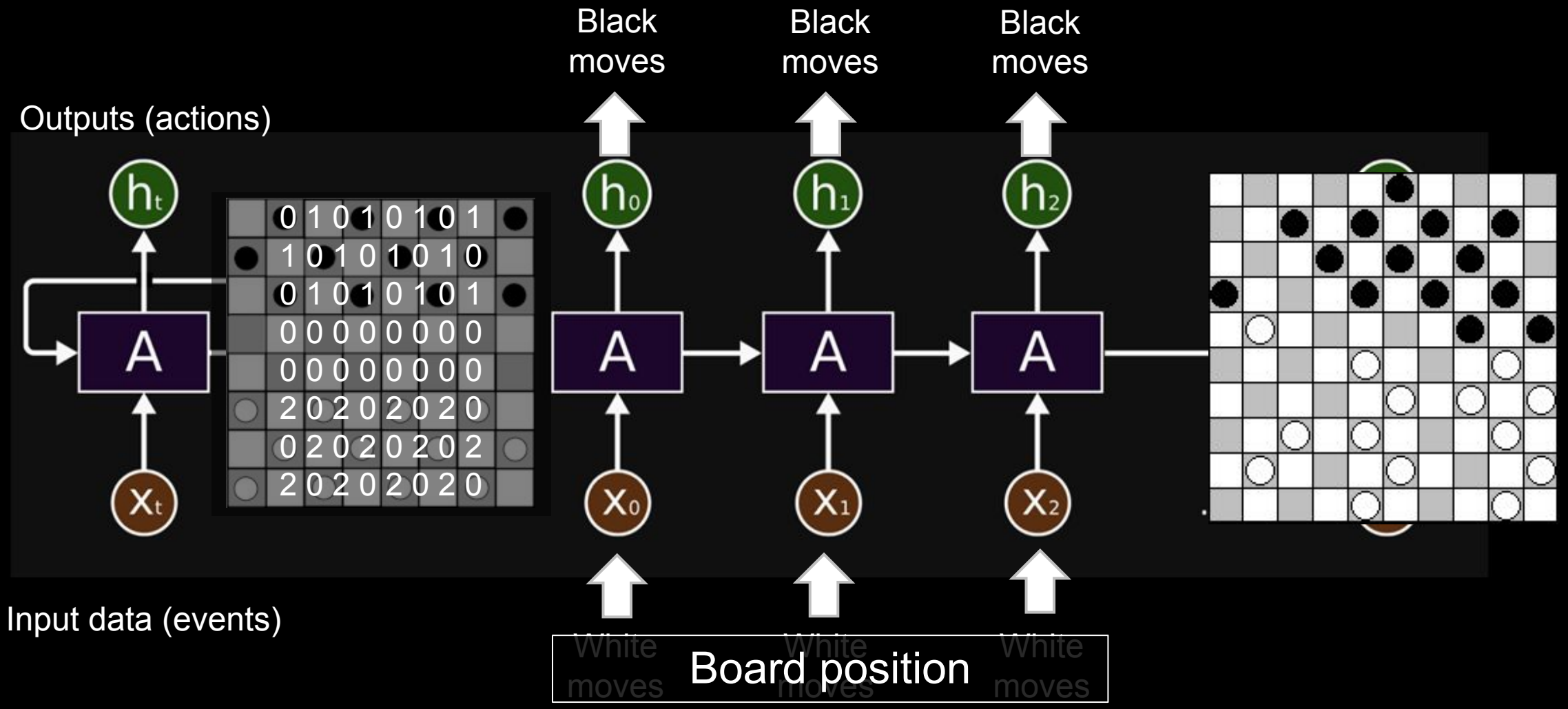
Color video: 4-D image data
Width
Height
Color channel (R, G, B)
Time

- Next, turn the encoded representation into some desire output:
  - Image label
  - Confidence (e.g., medical)
  - Another image (restoration, enhancement)
  - Generative parameters

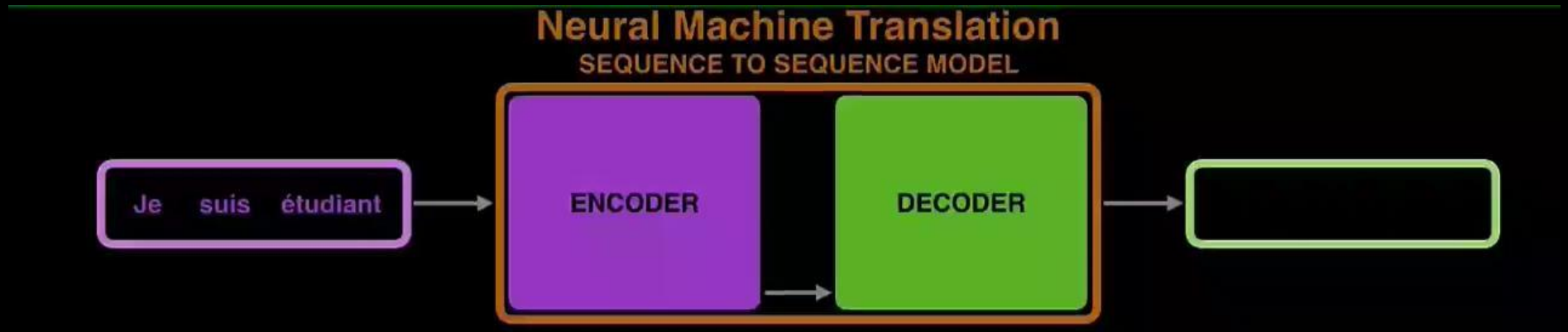


fMRI: 4+D image data

Sequential data – recurrent networks



Transformers / LLMs

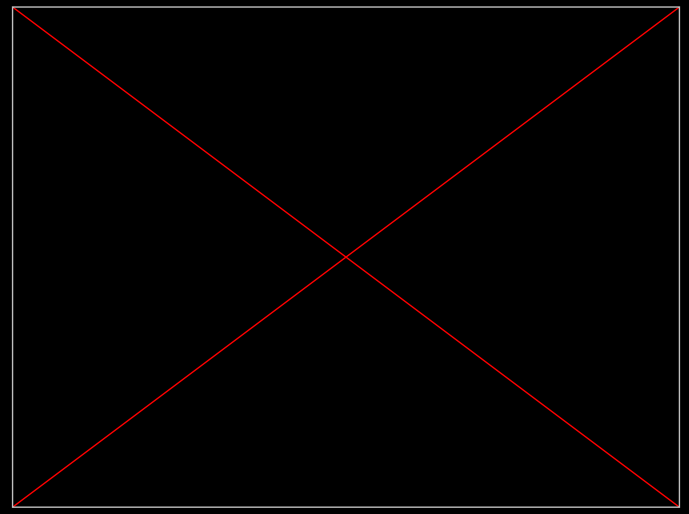


Transformers involve “self-attention” that modifies a word encoding based on its context, considering both input and position information in parallel.

The decoder then uses self-attention and “encoder-decoder attention” to consider different properties (e.g., semantic, grammatical rules) when constructing an output.

Alammar, J (2018). The Illustrated Transformer https://jalammar.github.io/illustrated-transformer/

Transformers / LLMs



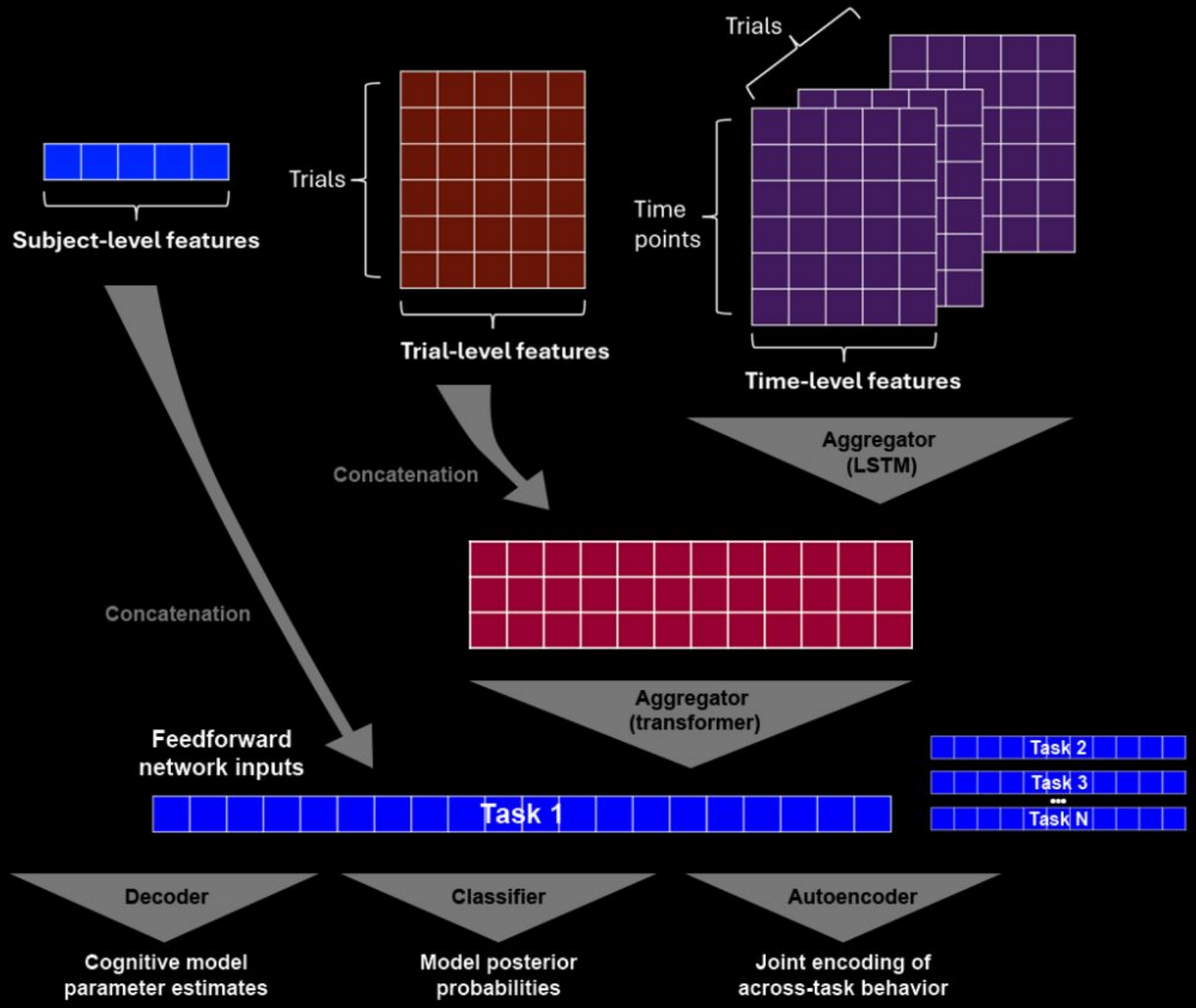
Transformers involve “self-attention” that modifies a word encoding based on its context, considering both input and position information in parallel.

The decoder then uses self-attention and “encoder-decoder attention” to consider different properties (e.g., semantic, grammatical rules) when constructing an output.

Alammar, J (2018). The Illustrated Transformer https://jalammar.github.io/illustrated-transformer/

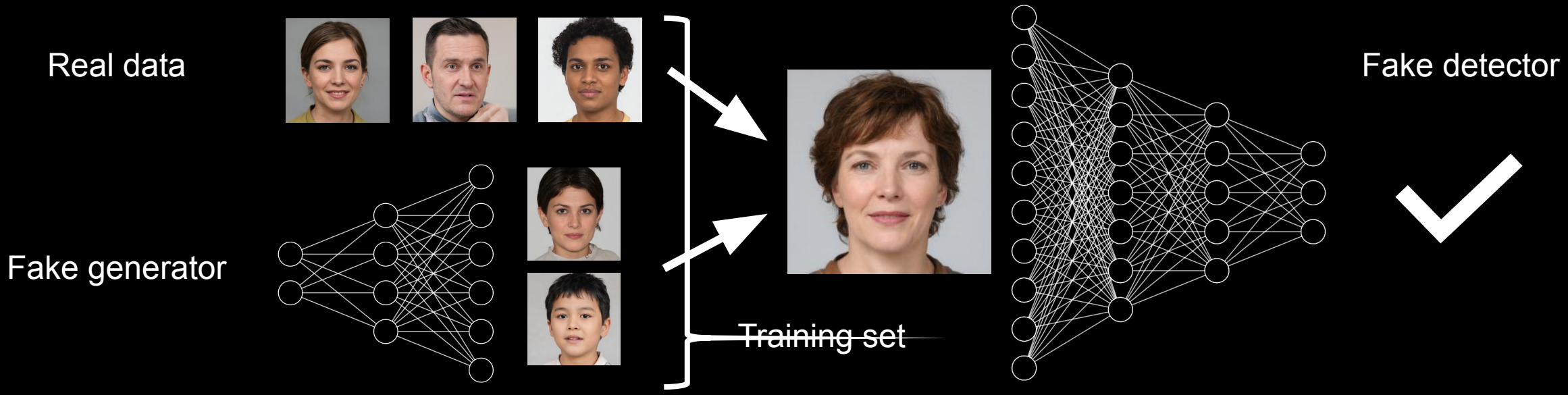
Transformers

Many potential uses in modeling behavioral or neural data, which all have sequential structure



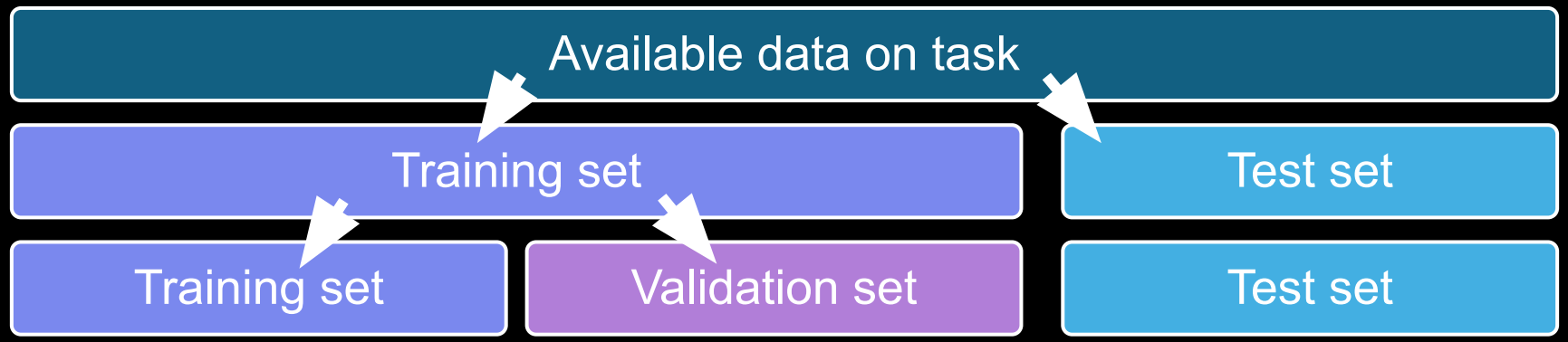
Another way to learn

- AlphaGo learned encodings from human players, but then it improved almost entirely through **self-play**
  - These are called **generative adversarial networks** , which compete against one another to accomplish opposing tasks
  - Compete at a game, Fraud creation vs detection, Protein folding
  - **Invertible networks** allow parameters <-> data (e.g., normalizing flows)



Training sets

- Any algorithm needs data from the world
  - In supervised learning, these are input-output mappings
  - In unsupervised learning, these may be quantifiable attributes / dimensions
  - Split into training sets and test sets (and sometimes validation)



- Training sets are absolutely critical to getting ML to do anything!
  - Examples of the task or structure you want it to learn

Warnings

- We might be able to explain the building blocks of deep learning, but the emergent behavior is often a **"black box"**
  - Explainable AI is a big emerging field

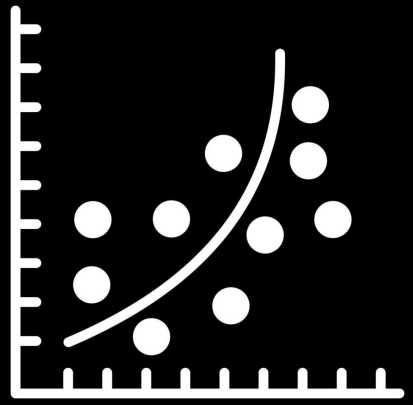
- Any given solution (in terms of weights) is probably not unique
  - There are many ways to do the same thing when you have enough parameters to fit!
  - Often easier to use as a benchmark for how well one could do

- AI / ML are more engineering tools (do they work?) rather than scientific ones (how do they work?)

Agenda

- Machine learning fundamentals
  - What are AI, ML, neural networks, and deep learning?
  - Why are they so successful at a variety of tasks?
- Applications and special additions
  - Images / videos, text, sequential data
  - Problems with no correct solution (unsupervised learning)
  - Generative adversarial networks
- Uses in mathematical psychology
  - Likelihood approximation, model fitting, comparison, discovery
  - Simulation studies of learning, evolution

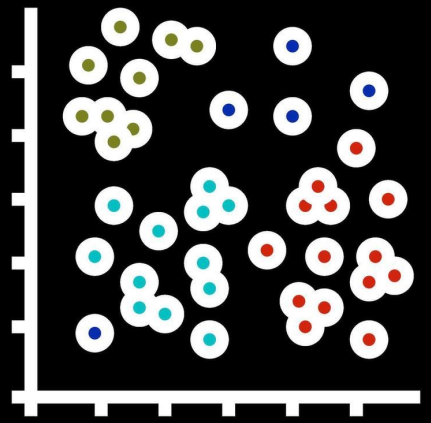
Flavors of machine learning



Prediction

Supervised learning

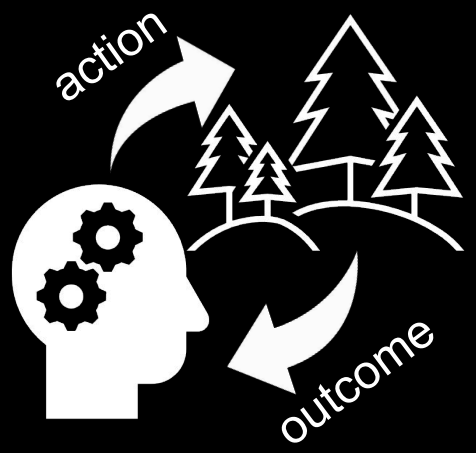
Approximates the relationship between variables based on known pairs of values



Discovery

Unsupervised learning

An algorithm creates a structure to organize data based on their quantitative attributes



Exploration

Reinforcement learning

An agent interacts with its environment to determine action-response mappings from experience

Two approaches

Likelihood Approximation

Goal: Obtain the likelihood of the data given the parameters

Simulate: Combinations of parameters and data

Train: Neural network to circumvent the need for exact likelihood functions

Apply: Combine network with priors to sample from posterior (e.g., MCMC)

Posterior Approximation

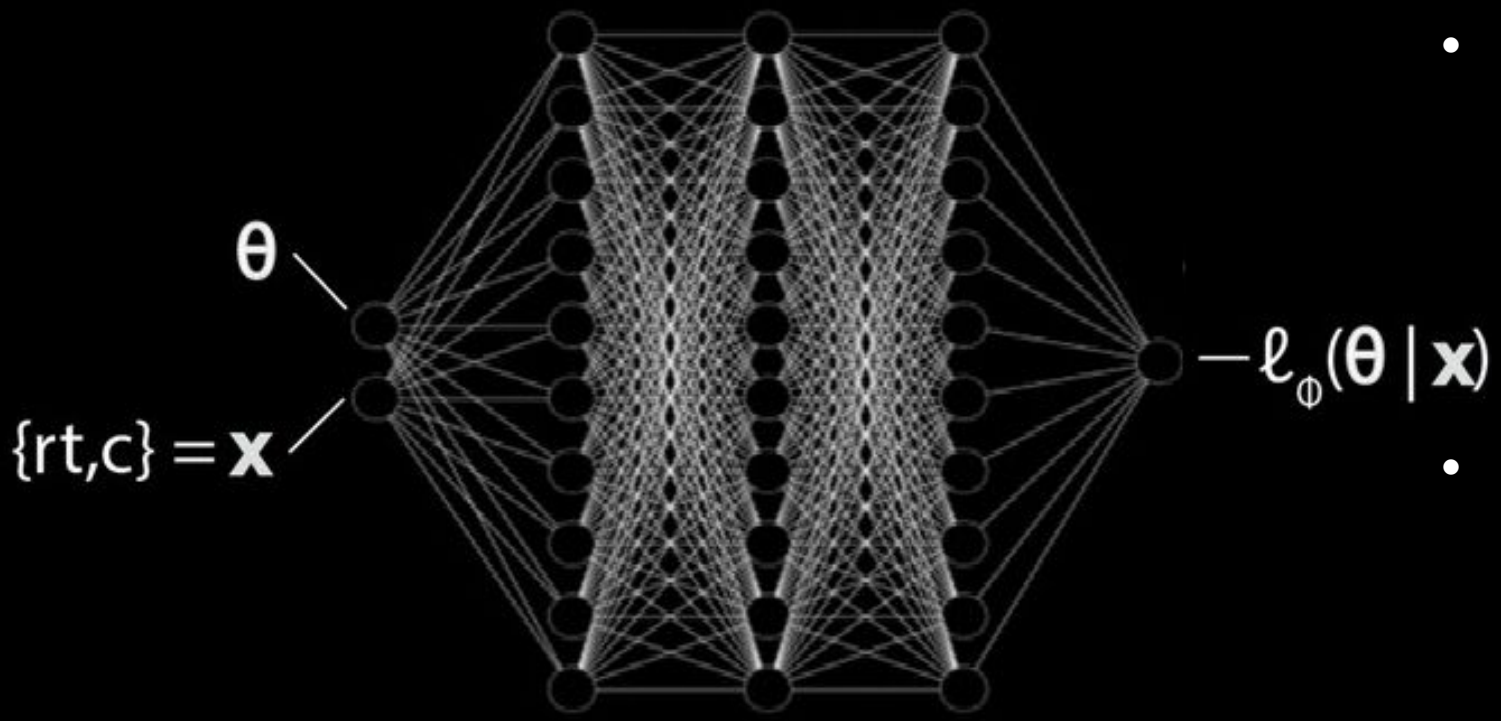
Goal: Obtain directly the posterior parameters / model given the data

Simulate: Data from different parameters / models

Train: Neural network to invert the relationship between model (parameters) and data

Apply: Feed (summarized) data set as inputs in order to obtain the posterior distribution

Likelihood approximation



- Key limitation is the need for a likelihood function
  - MCMC requires this function to evaluated many times for every data point

- **Amortize** this computation by training a network
  - Simulate data (rt, c) from parameters (θ) with known / approximate likelihood (θ|x)
- Learn this relationship, then use your favorite sampler to get posterior

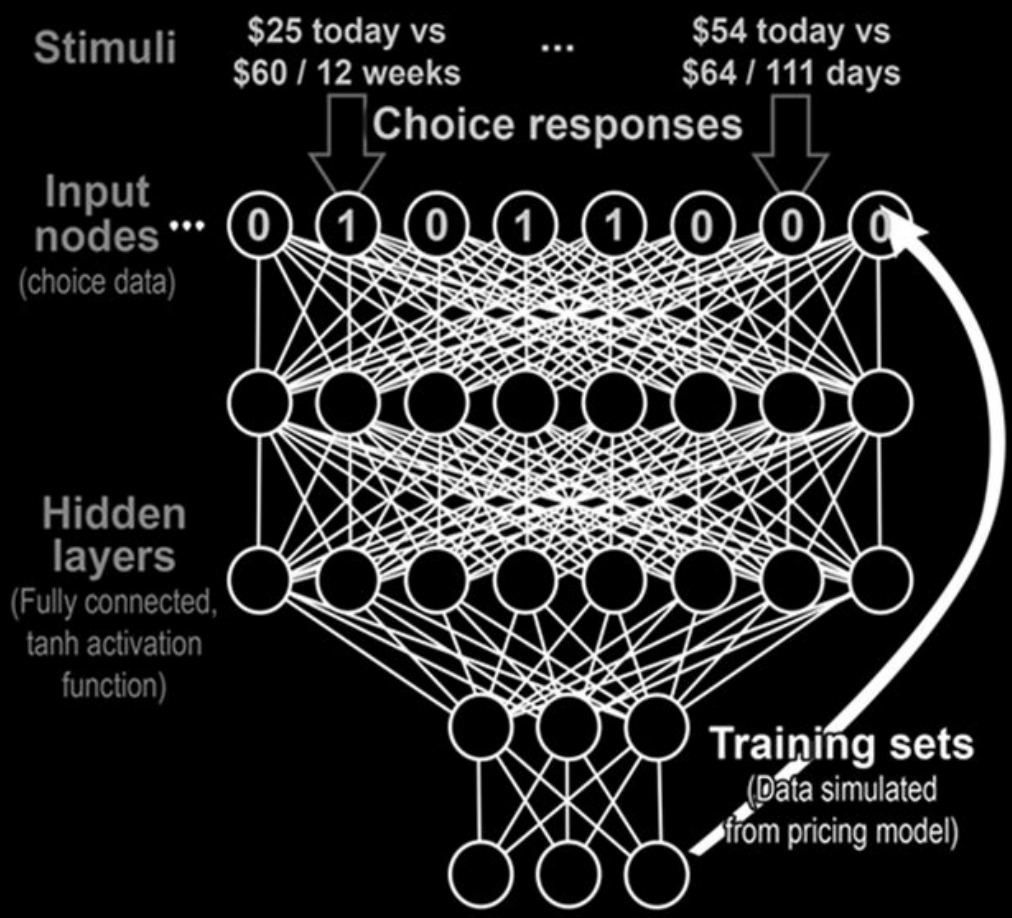
Fengler, A., Govindarajan, L. N., Chen, T., & Frank, M. J. (2021). Likelihood approximation networks (LANs) for fast inference of simulation models in cognitive neuroscience. *Elife*, *10*, e65074.

Direct approximation

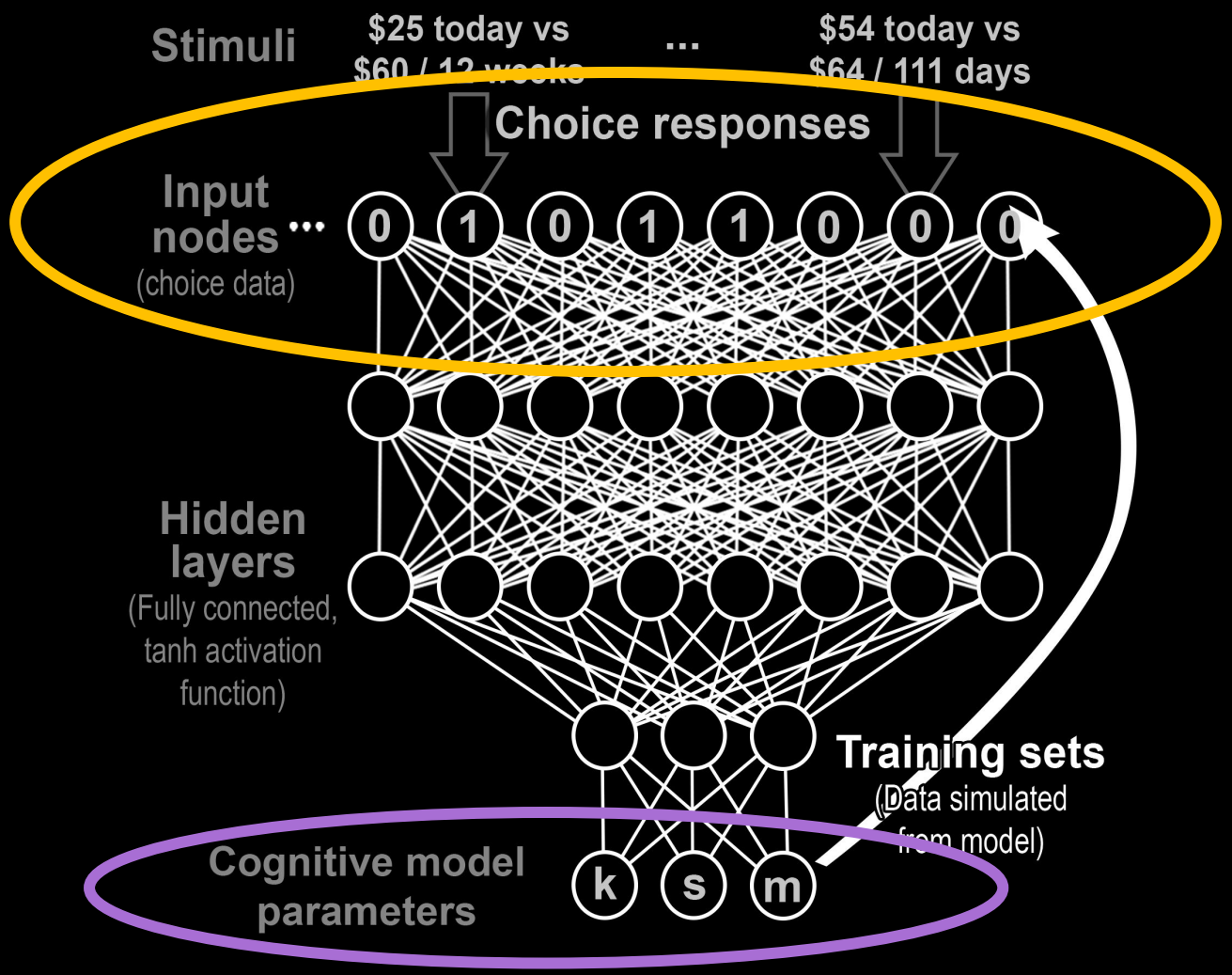
- Teach a machine to do the model fitting for us!
  - **Deep learning** for parameter estimation and model comparison

Simulation
Parameters → Data

Invert it!
Data → Parameters



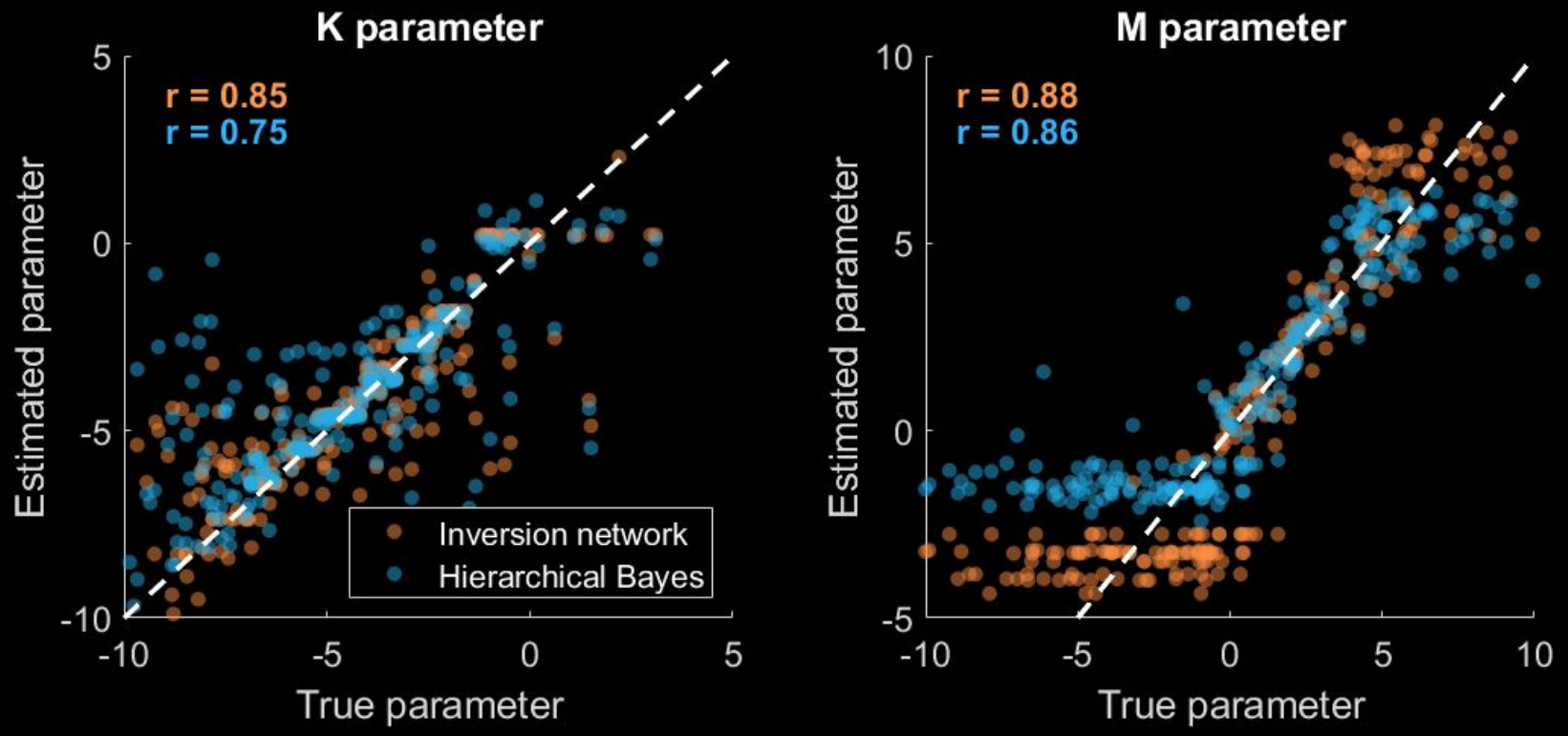
Neural networks for model fitting



- **Inputs:** complete or simple statistical summary of the behavioral data
  - Likelihood approximation: parameters and data
- **Outputs:** best-fit model parameters (what generated the data?)
  - Likelihood approximation: (log) likelihood of the data given the parameters

Parameter estimation

Example: Estimation of parameters of a 2-parameter hyperbolic discounting model using data from the Monetary Choice Questionnaire



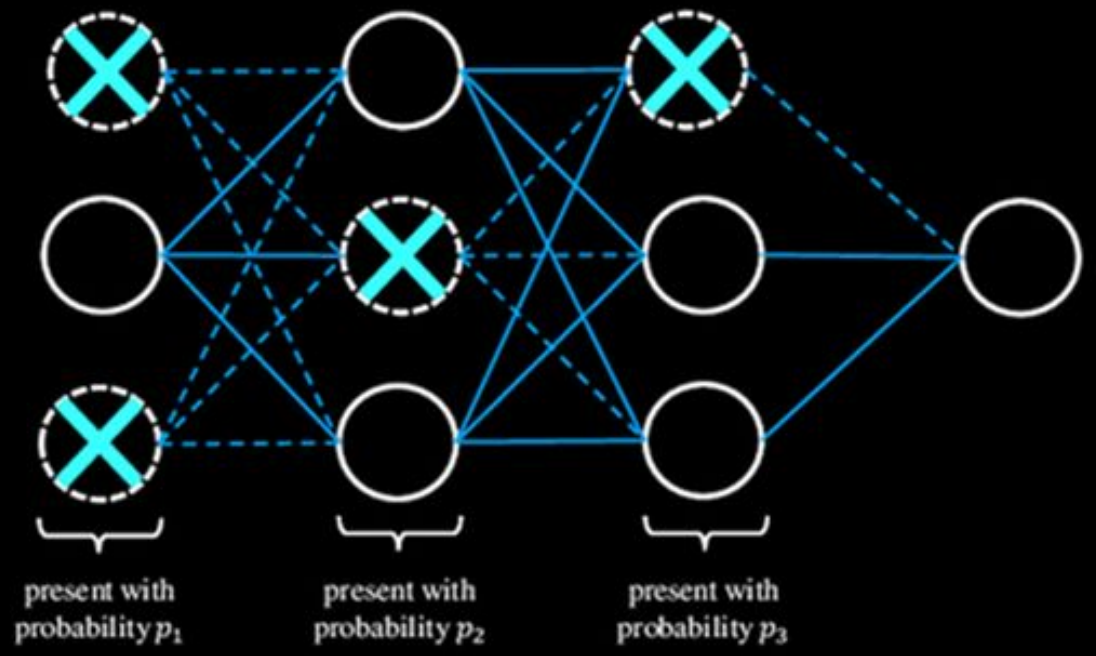
Kvam, P. D., Sokratous, K., Fitch, A. K., & Vassileva, J. (in press). Comparing likelihood-based and likelihood-free approaches to fitting and comparing models of intertemporal choice. *Behavior Research Methods*.

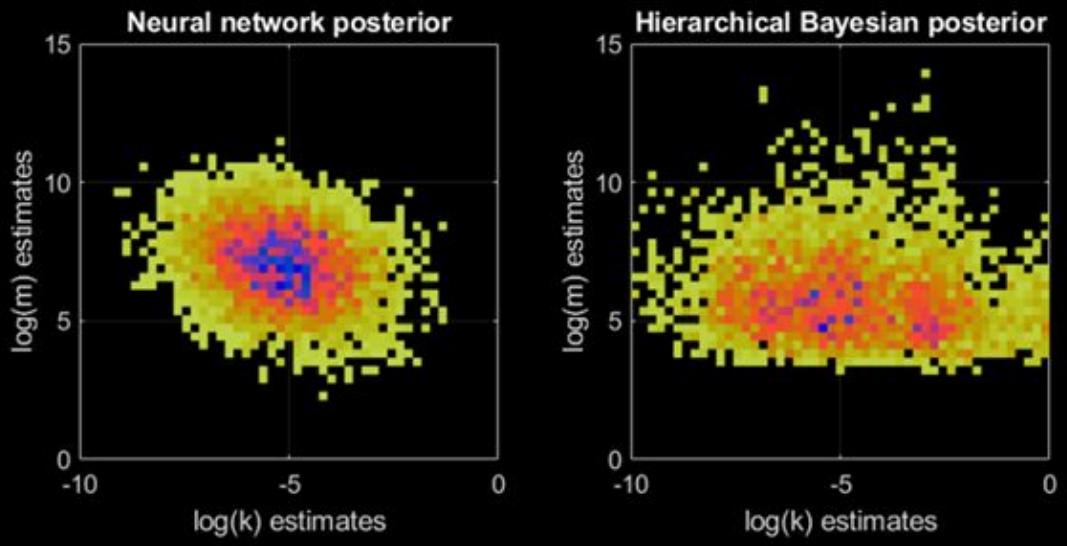
Posterior estimation

- How do we go from a point estimate to a posterior distribution? A few solutions:
  - **Normalizing flows** (Stefan covers this later)
    - Invertible networks also approximate the generative (simulator) model
  - Secondary network to estimate error / posterior variance
    - Train it to estimate expected error based on parameters & data
  - Bayesian neural networks
    - Estimate both a mean and variance for every weight in the network
  - **Dropout sampling**
    - Sample from functions relating data to parameters

Dropout sampling

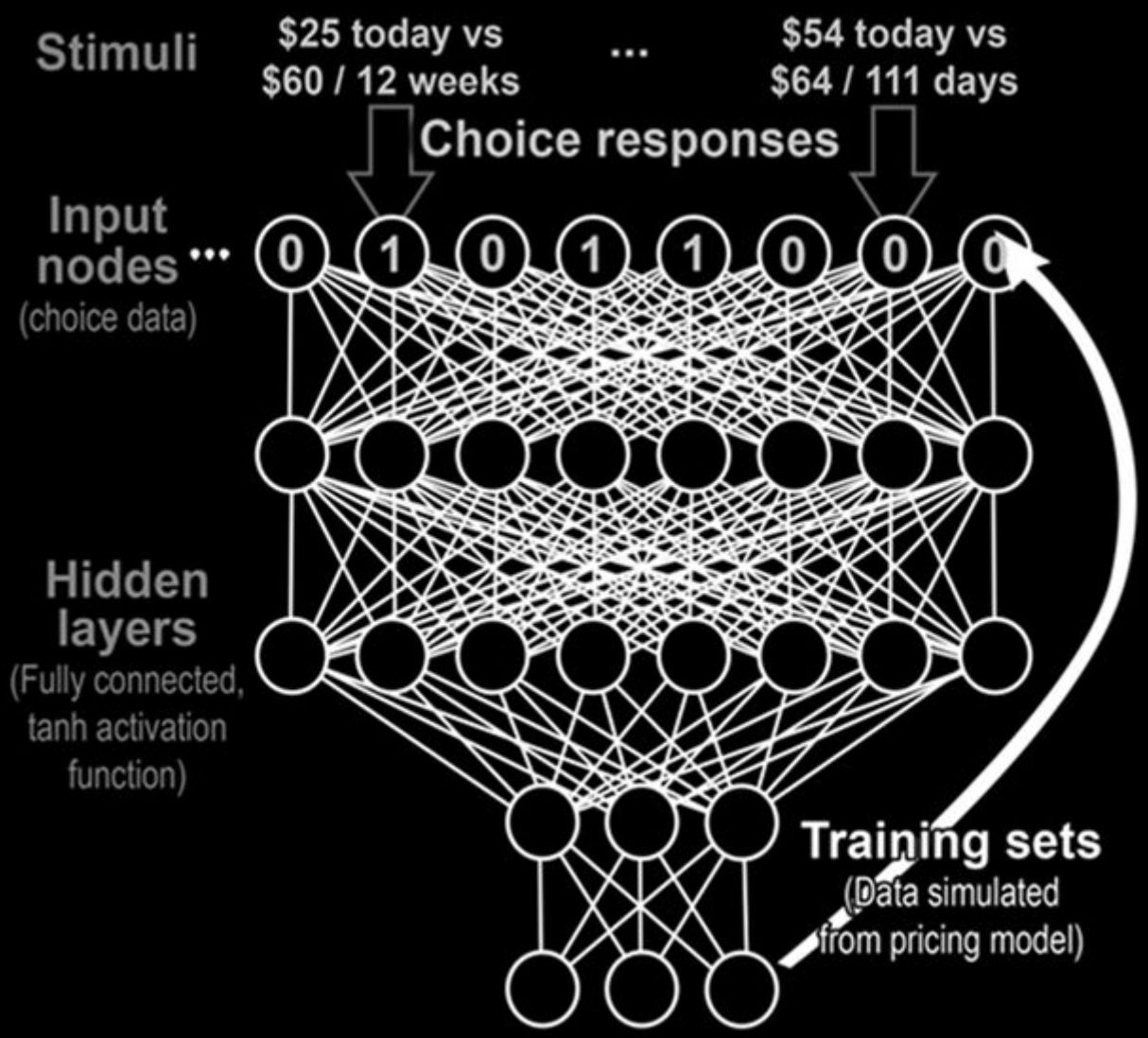
- Dropout layers are usually used to prevent overfitting
  - During training, set node `j` to zero with probability `p`
  - Creates redundant "sub-networks" that approximate the same function
- Applying dropout during testing allows for **sampling from the posterior** of a deep Gaussian process linking data to parameters





Model comparison

- Instead of parameters, the outputs are different models
  - Model comparison as a **classification** problem
  - Trained network assigns a posterior probability to each model given the data



Model comparison

Confusion matrix for neural network-based approach to model comparison.

		Inferred model		
		Direct difference	Hyperbolic	Hyperboloid
True model	Direct difference	72.4[illegible]%	5.37%	[illegible]2.16%
	Hyperbolic	16.0[illegible]%	[illegible]	[illegible]2.82%
	Hyperboloid	12.2[illegible]%	1.66%	[illegible]6.07%
	Total	100.83%	58.12%	141.05%

73.60%

Confusion matrix for DIC-based approach to model comparison.

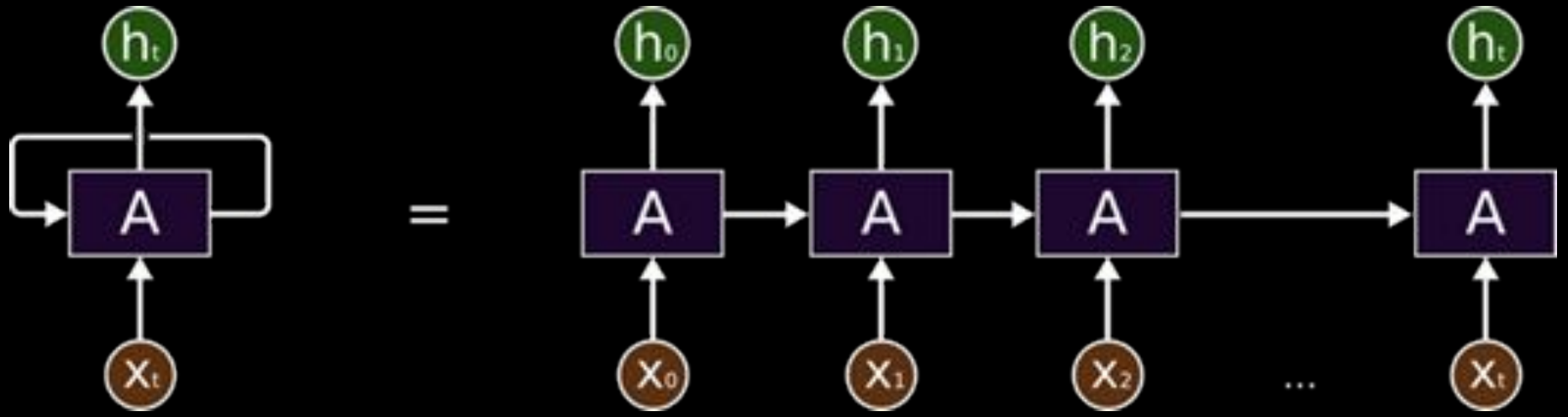
		Inferred model		
		Direct difference	Hyperbolic	Hyperboloid
True model	Direct difference	32.48[illegible]%	40.33%	28[illegible].20%
	Hyperbolic	26.26[illegible]%	68.25%	5[illegible]39%
	Hyperboloid	37.90[illegible]%	36.31%	25[illegible].80%
	Total	95.64%	144.98%	59.38%

42.21%

Kvam, P. D., Sokratous, K., Fitch, A. K., & Vassileva, J. (in press). Comparing likelihood-based and likelihood-free approaches to fitting and comparing models of intertemporal choice. *Behavior Research Methods*.

Additional problems: input structure

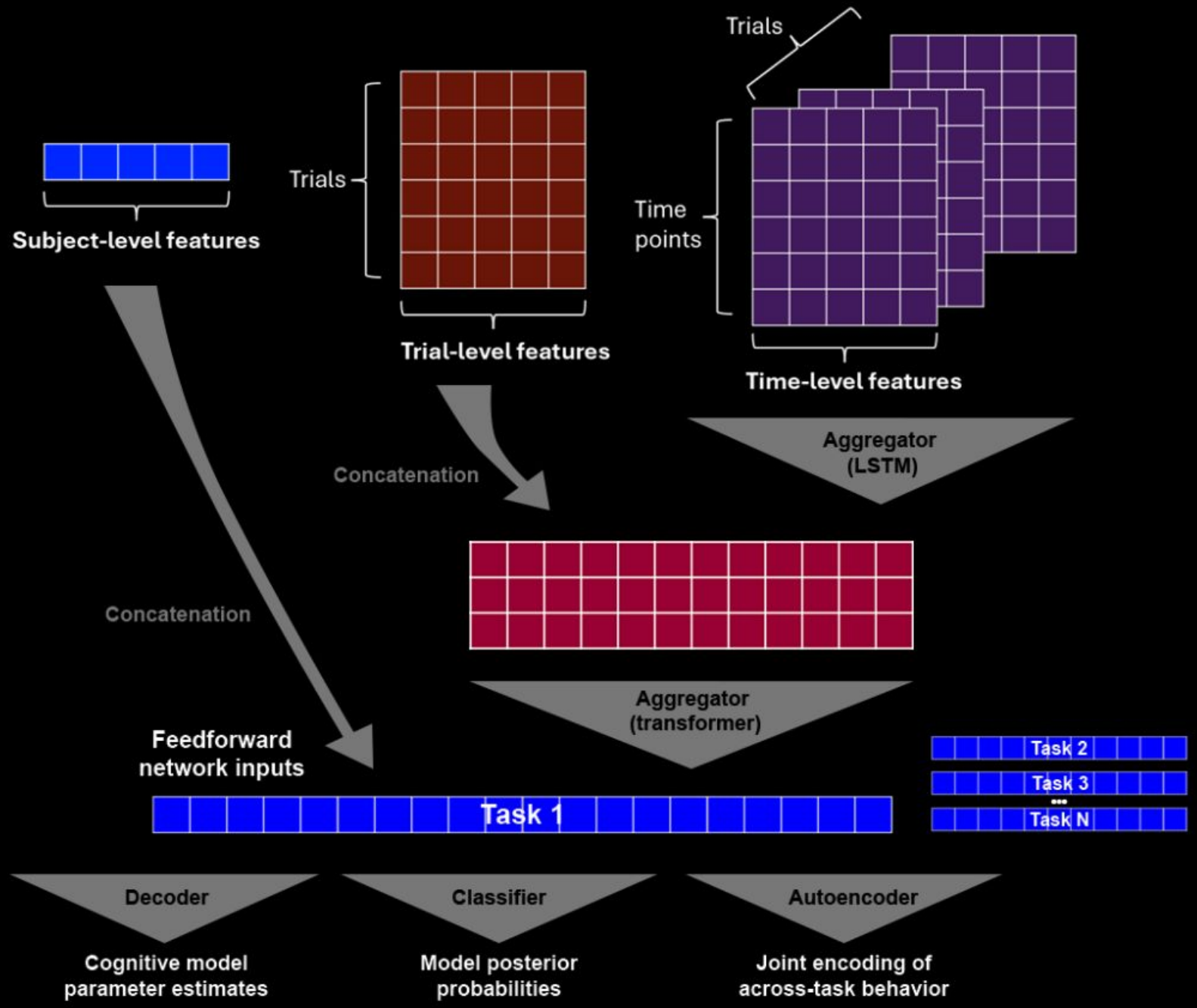
- Different experiments (and even participants) vary in terms of the conditions, # of trials, order, etc.
  - Neural network typically has a fixed number of inputs
- A few options:
  - Summary statistics (as in ABC): manual or automatic, **autoencoders**
  - **Recurrent networks** (RNN, GRU, LSTM) or transformers



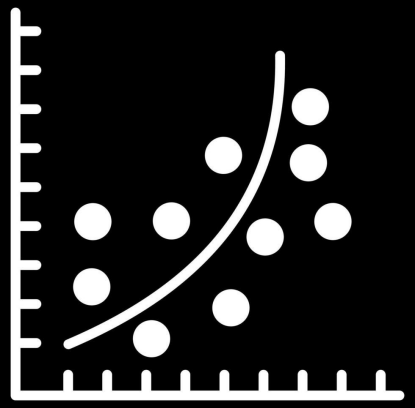
LSTMs & Transformers

Many potential uses in modeling behavioral or neural data, which all have sequence structure

Lets us relax the i.i.d. assumptions that we usually make when modeling



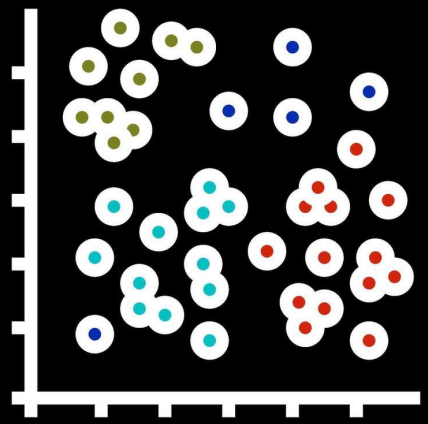
Flavors of machine learning



Prediction

Supervised learning

Approximates the relationship between variables based on known pairs of values



Discovery

Unsupervised learning

An algorithm creates a structure to organize data based on their quantitative attributes

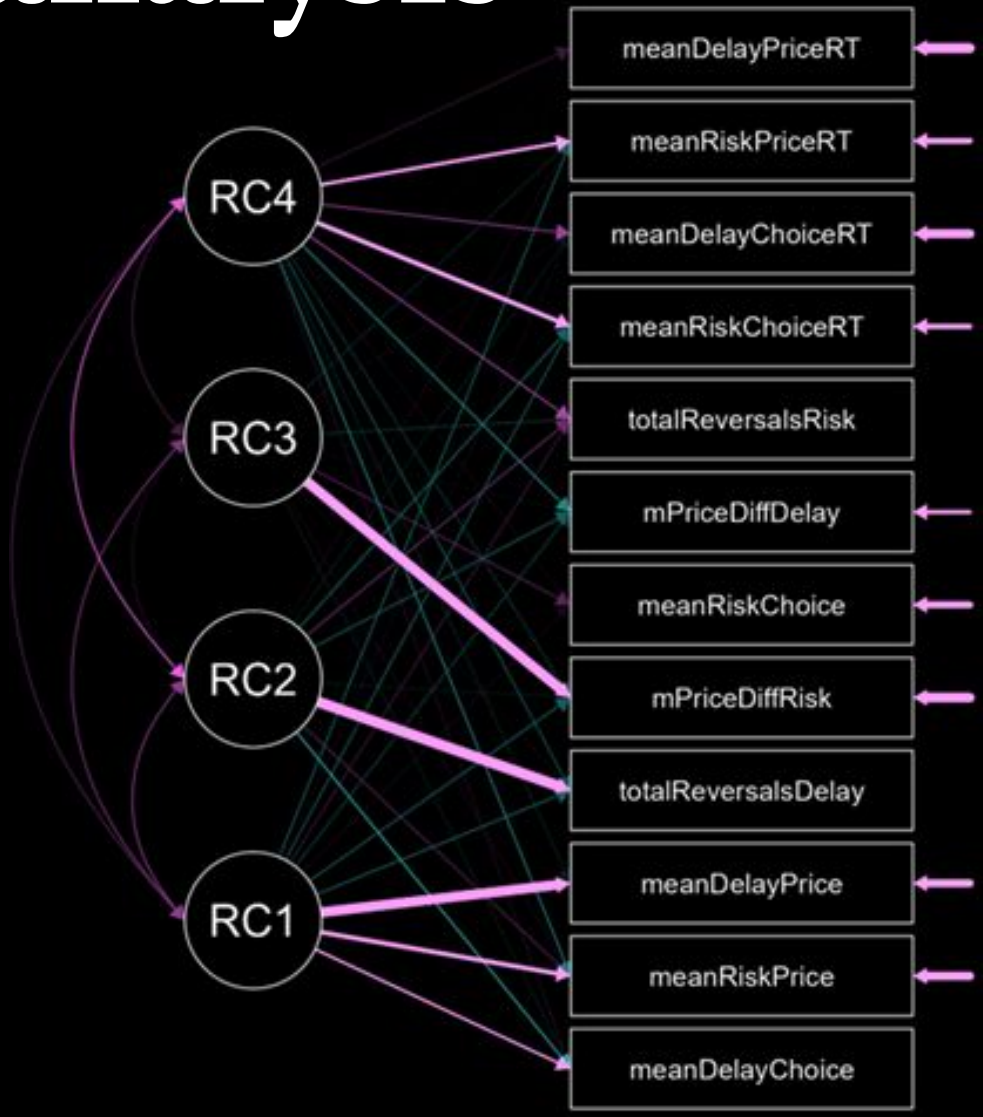
Exploration

Reinforcement learning

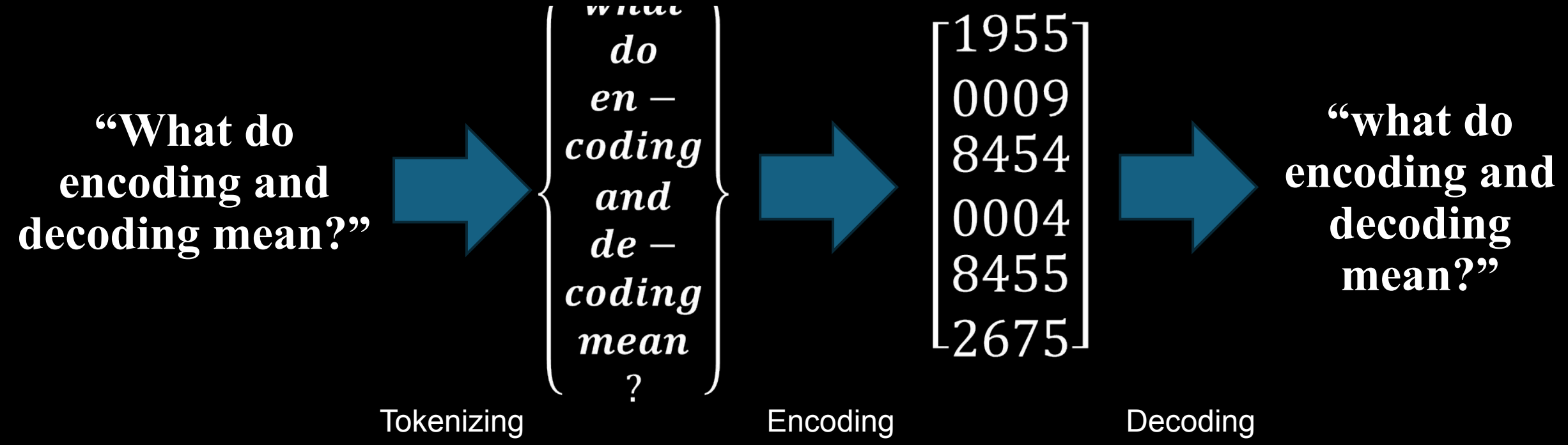
An agent interacts with its environment to determine action-response mappings from experience

Exploratory data analysis

- Traditionally use approaches like **exploratory factor analysis** or principal components analysis to identify latent traits / processes
- However, these assume linear relationships between measures
  - Linear correlation / covariance matrix
- They can also create “phantom oscillations” (Shinn, 2023)

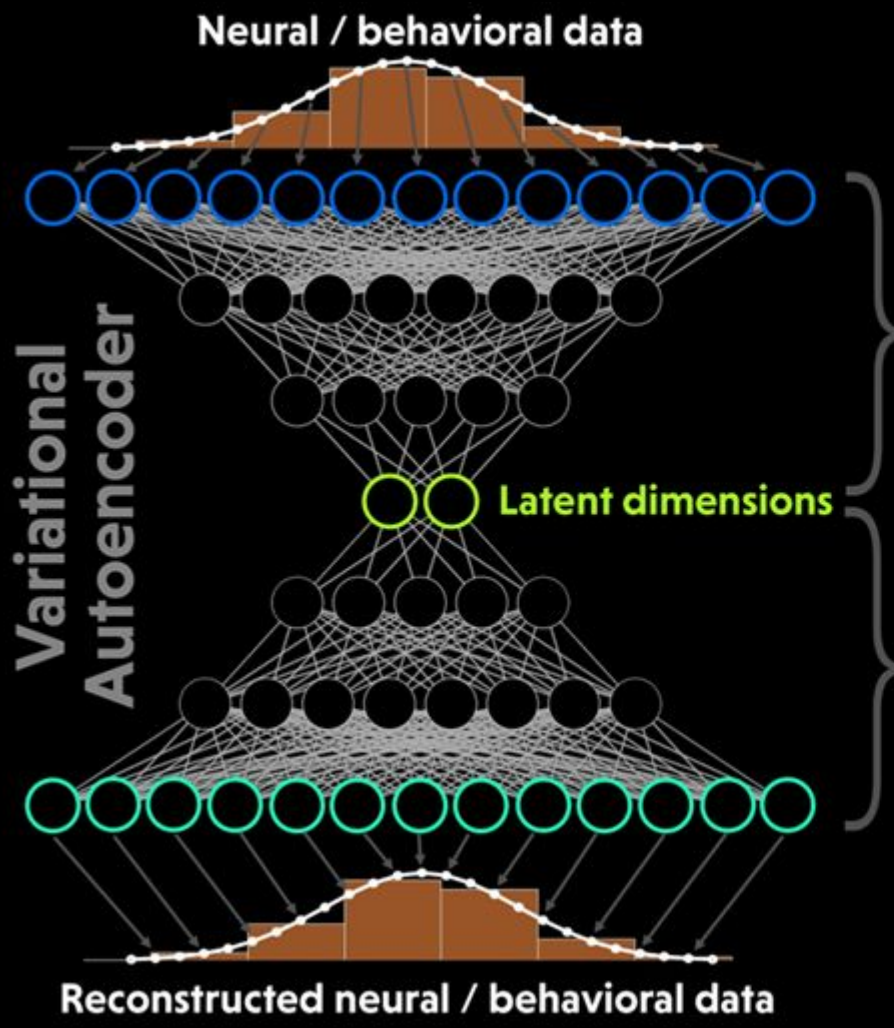


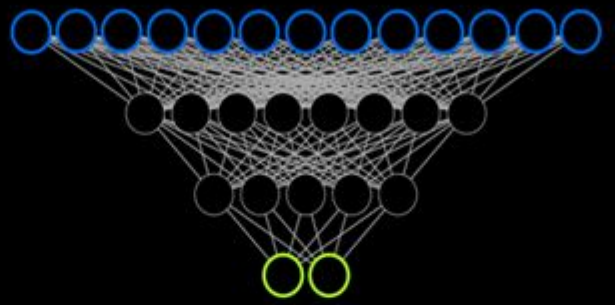
Encoding & decoding the same thing



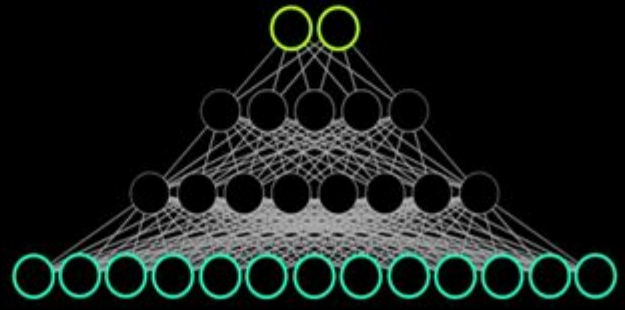
Solution: Autoencoders

Autoencoders



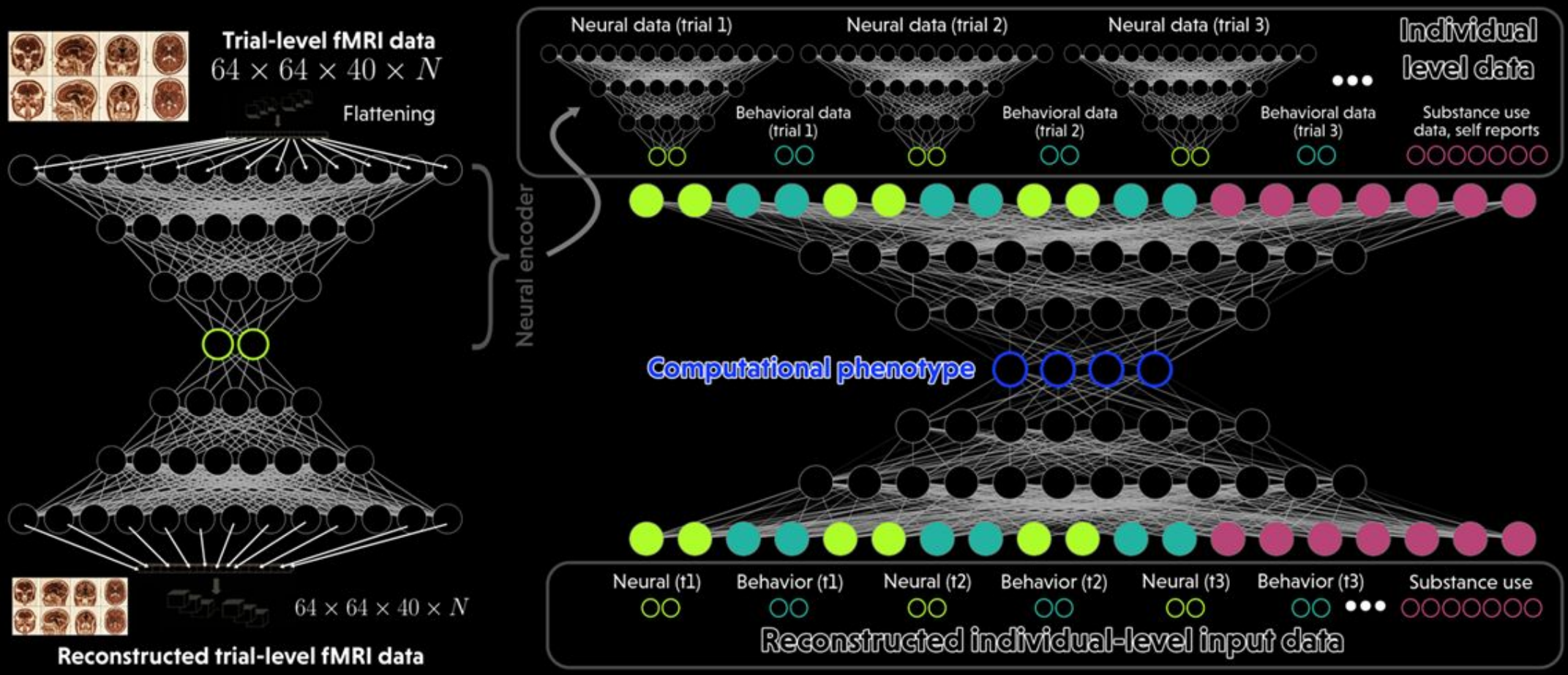


- Condense incoming data while preserving total information
- Identify latent constructs that predict / explain behavior
- Identify latent groups, new clusters (in combination with unsupervised learning)
- Exploratory data analysis



- Simulate expected data by varying latent dimensions
- Evaluate error in predictions
- Optimally impute / reconstruct missing data
- Explore nonlinear relationships between different outcome measures (via latent dimensions)

Encoding & processing data



Conclusions

- Modern machine learning relies on its ability to **approximate relationships between inputs & outputs**
- Tasks involving prediction, discovering structure, or simulation can make good use of existing tools
  - **Likelihood approximation** uses simulation to overcome the need for an analytic likelihood, opening up simulation-based models
  - **Posterior approximation** also skips the MCMC sampling step, directly obtaining parameters / models from the data
- There are now a variety of ways to overcome (previous) limitations on things like posterior estimation and fixed inputs
  - Posterior from normalizing flows, dropout sampling (Gaussian process)
  - Sequential inputs can be handled using recurrent neural networks, transformers, or autoencoder latent dimensions