Challenging Traditional Beliefs in Machine Learning

Sebastian Johann Wetzel

PERIMETER INSTITUTE

Canada NRC·CNRC

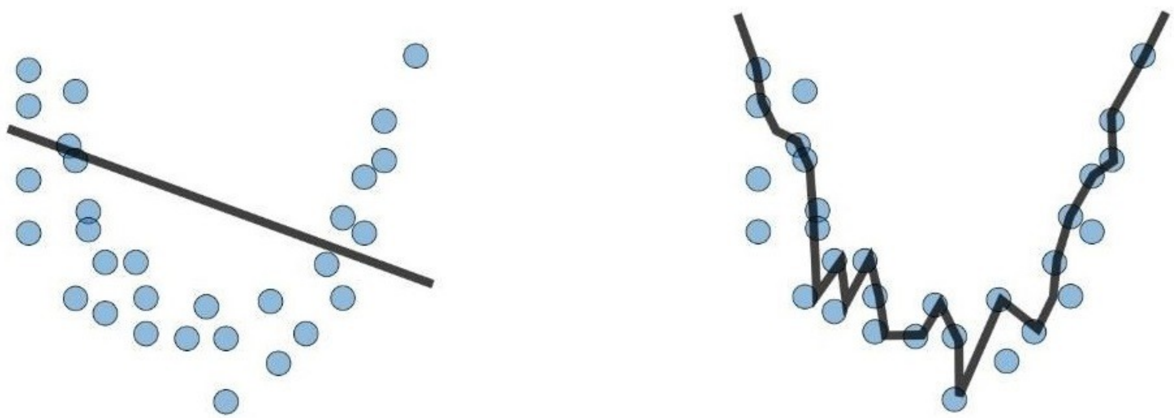
PIQUIL

Common Beliefs

1. Neural Networks are a black box algorithm

Common Beliefs

2. Prediction performance is limited by Bias Variance tradeoff



Common Beliefs

3. Don’t train your supervised model on unlabelled test data

Fewer Labels →

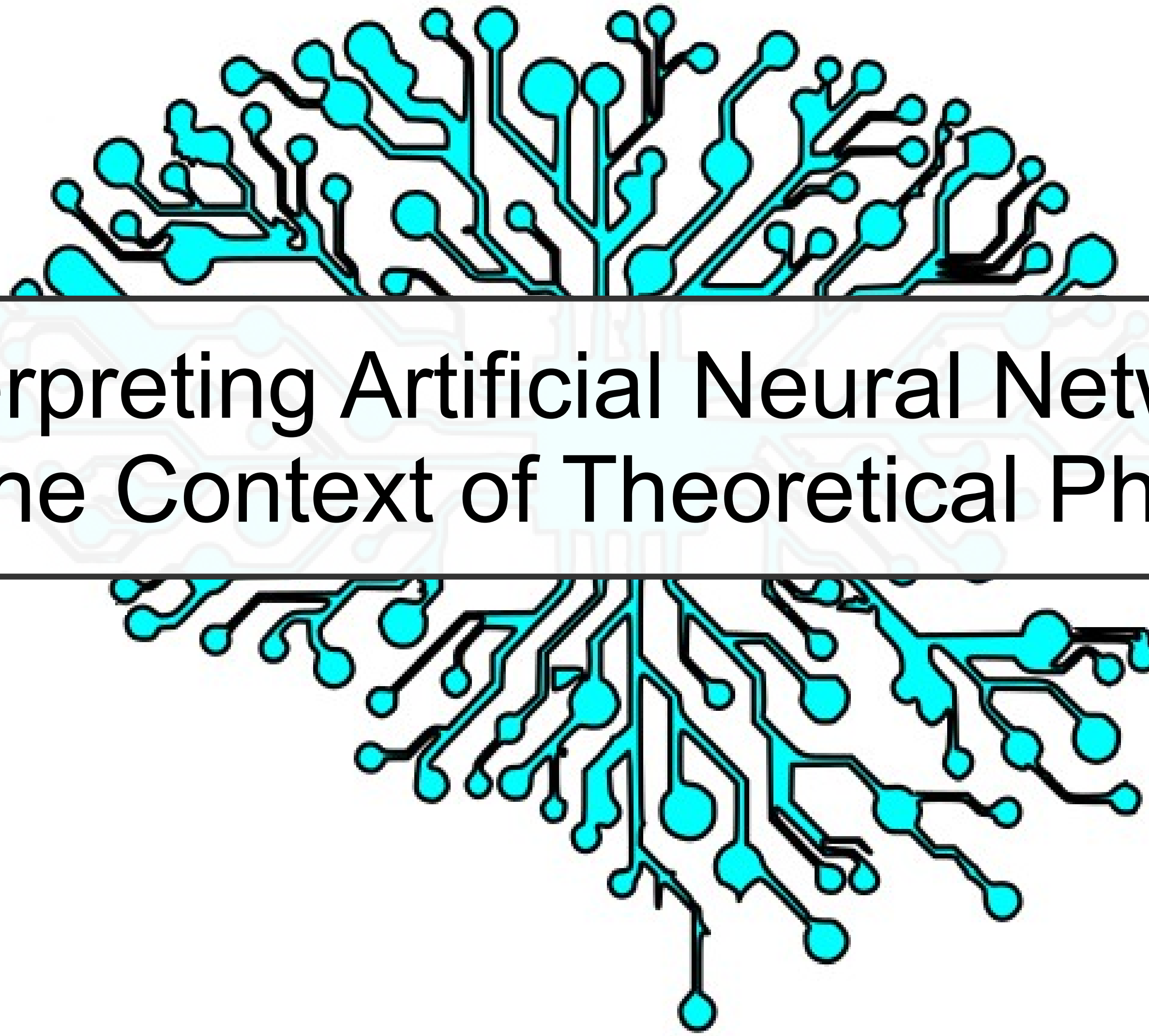
Supervised Classification	?	Unsupervised Classification =?
Supervised Regression	?	Unsupervised Regression =?

Common Beliefs

3. Dont train your supervised model on unlabelled test data

Fewer Labels →

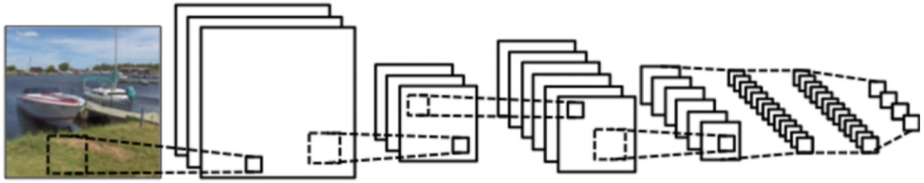
Supervised Classification	Semi-Supervised Classification	Unsupervised Classification =clustering
Supervised Regression	Semi-Supervised Regression	Unsupervised Regression does not exist

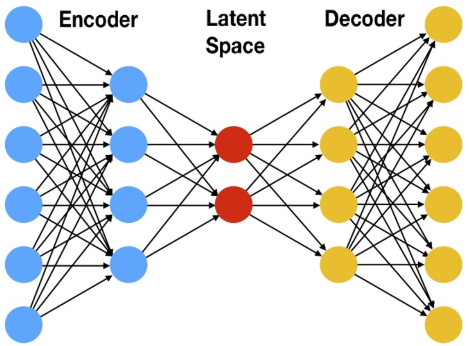


Interpreting Artificial Neural Networks in the Context of Theoretical Physics

Success of Artifical Neural Networks

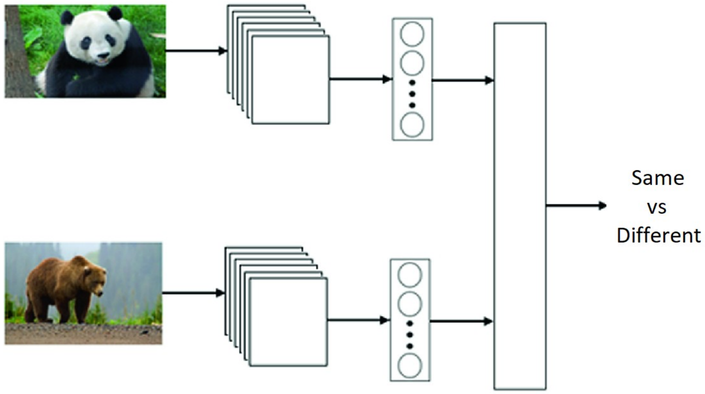
Image Classification
(Convolutional Network)





Generative Modelling /
Anomaly Detection
(Autoencoders)

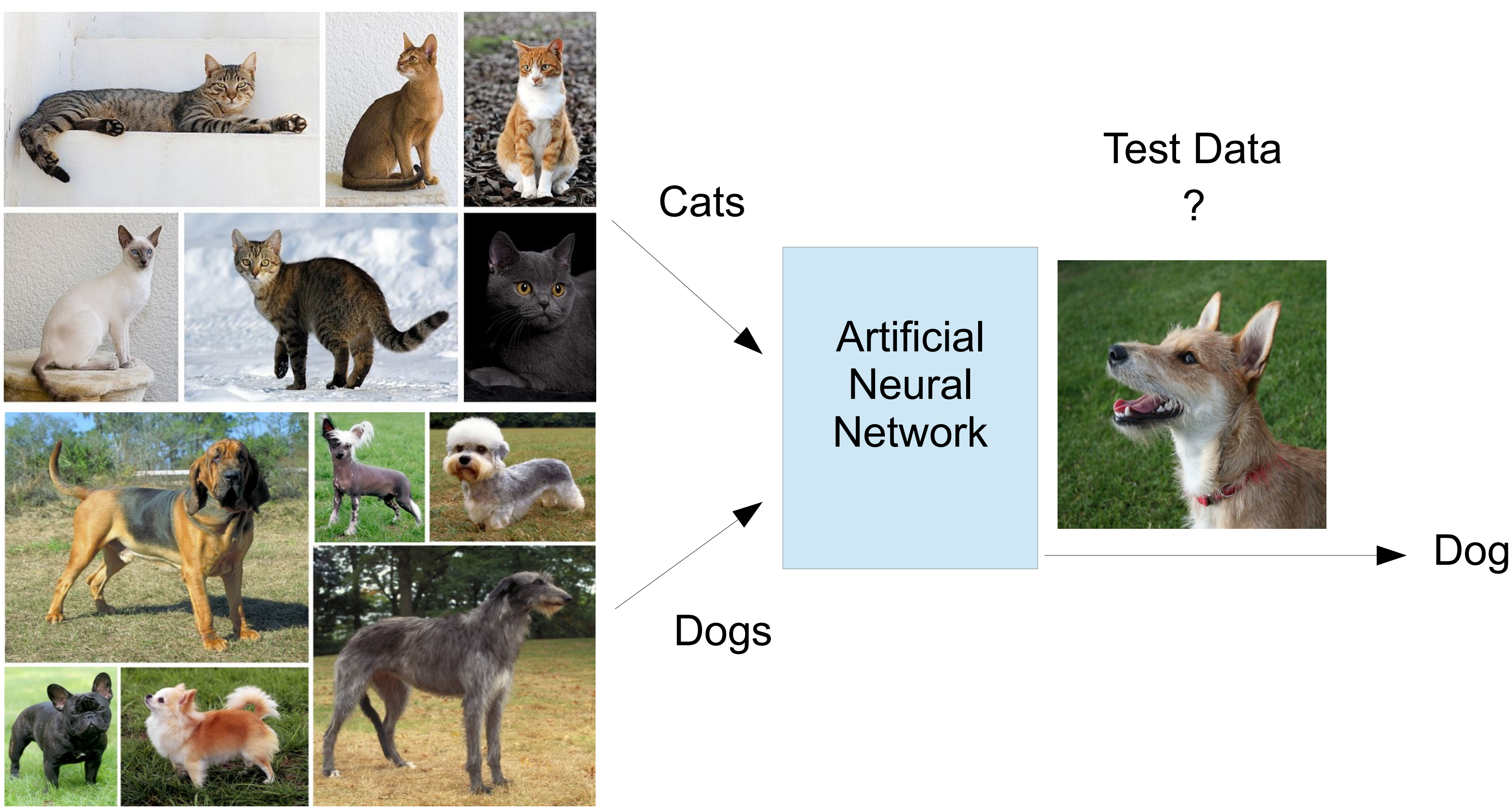
Similarity Detection
(Siamese Network)



(Supervised) Machine Learning with Neural Nets

„Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.“ - Wikipedia

Training Data



(Supervised) Machine Learning with Neural Nets

„Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.“ - Wikipedia

1) What does the Neural Network actually learn?
2) Can this Knowledge help in Scientific Discovery?

Training Data

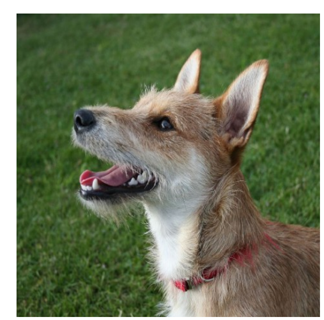
Cats

Dogs

Artificial Neural Network

Test Data

?



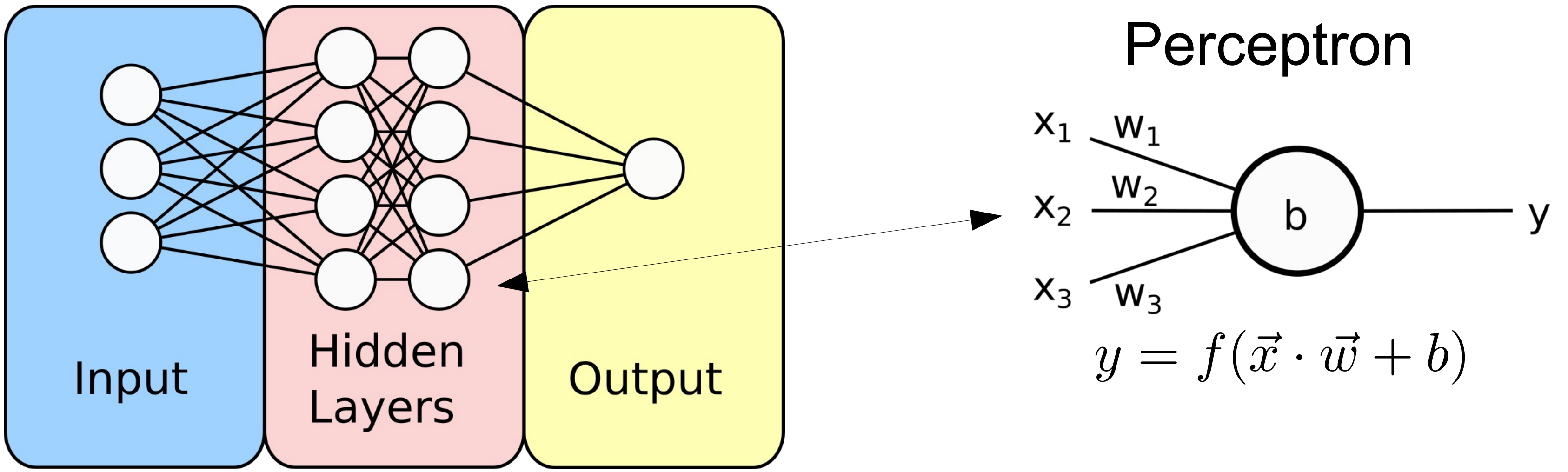
Dog

Overview

- ✗ Artificial Neural Networks
- ✗ Interpretation of Convolutional Neural Networks
- ✗ Interpretation of Autoencoders
- ✗ Interpretation of Siamese Networks

Artificial Neural Networks

Feed forward neural network

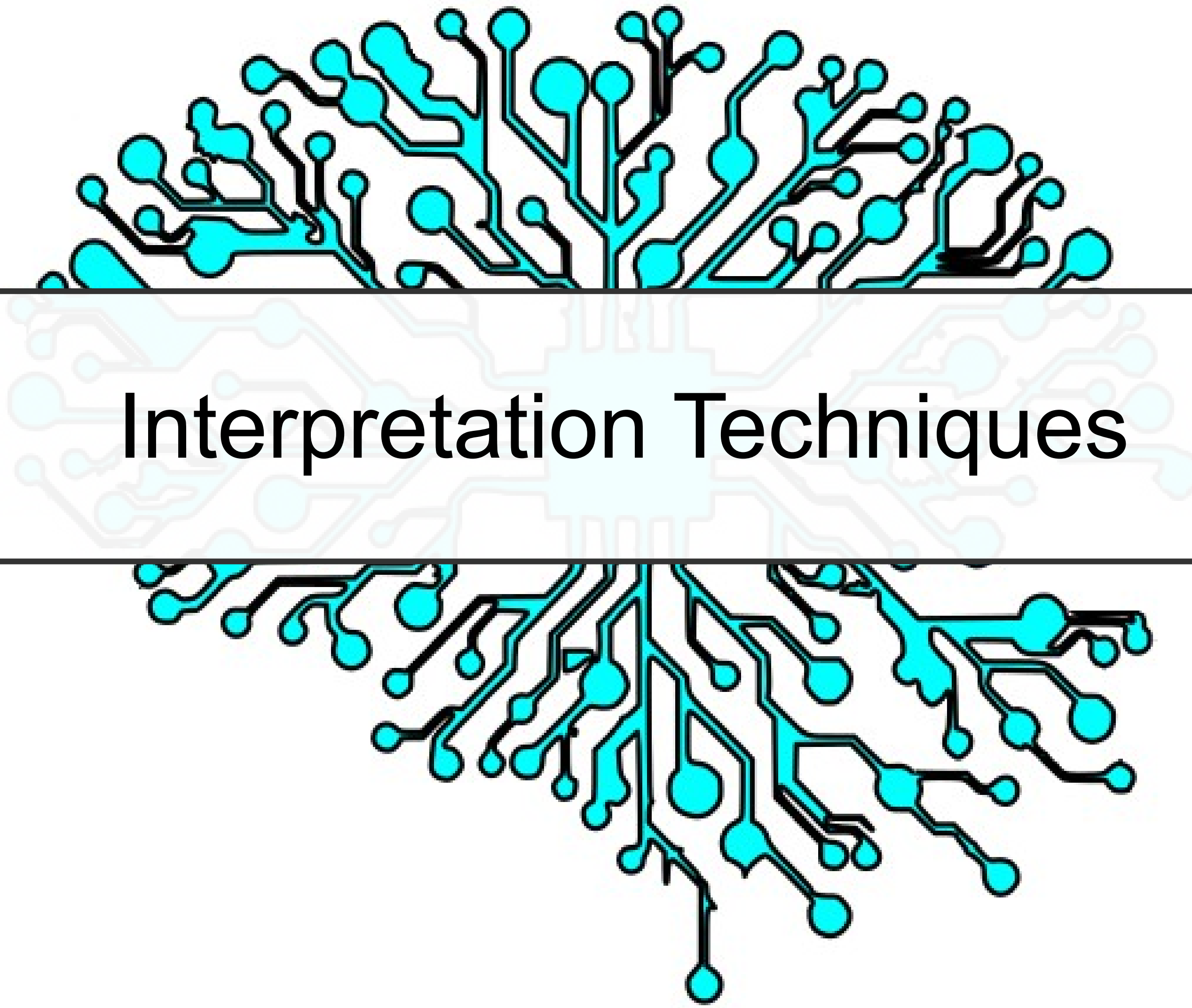


$y = f(\vec{x} \cdot \vec{w} + b)$

Input: Data $X = (\vec{x}_1, ..., \vec{x}_n)$, Label $Y = (y_1, ..., y_n)$

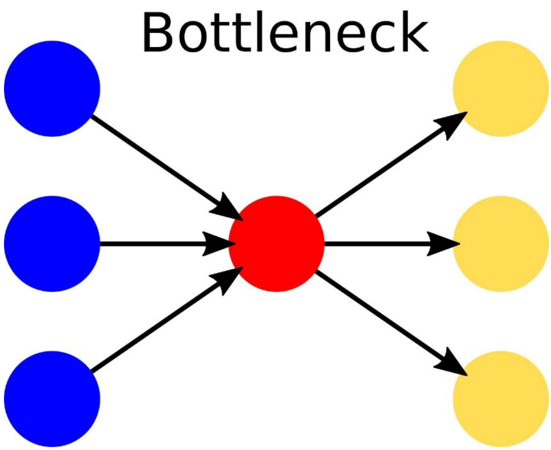
Output: $Y_{pred} = F(X, w_{ij}^L, b_i^L)$

Goal: choose w_{ij}^L and b_i^L such that $Y_{pred} \approx Y$

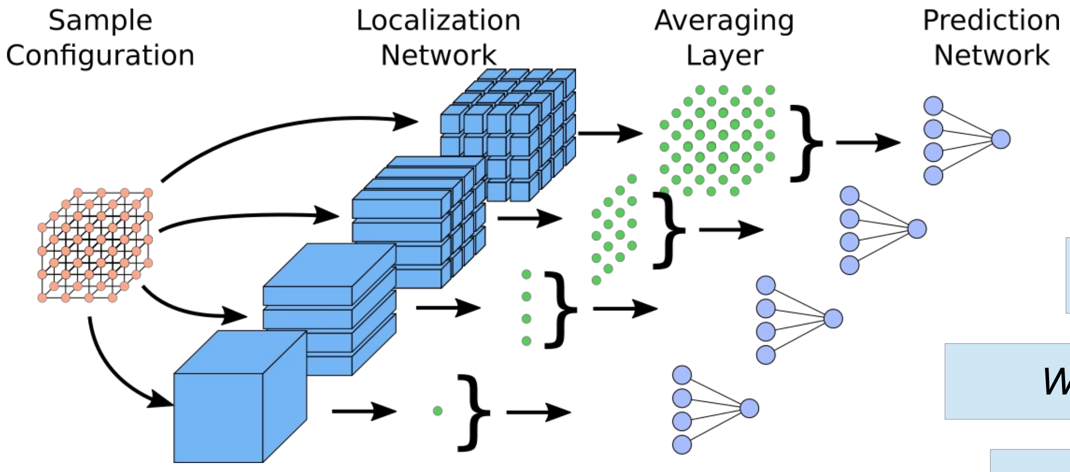


Interpretation Techniques

Bottleneck Interpretation
+Correlation Probing Neural Network



+



Wetzel, PRE 2017

Wetzel, Scherzer, PRB 2017

Wetzel, et al, PRR 2020

Looking at the weights

- ✗ No, works but only for the most simple problems.

Influence Functions

Phase Detection with Neural Networks: Interpreting the Black Box

Anna Dawid,[1,2] Patrick Huembeli,[2] Michał Tomza,[1] Maciej Lewenstein,[2,3] and Alexandre Dauphin[2]

- ✗ Remove specific datapoints or features and measure the effect on the performance
- ✗ Largest change in performance indicates the most influential data point or feature

Dawid et al, New Journal of Pyhsics 2020

Dark Matter

Discovering Symbolic Models from Deep Learning with Inductive Biases

Miles Cranmer[1] Alvaro Sanchez-Gonzalez[2] Peter Battaglia[2] Rui Xu[1]
Kyle Cranmer[3] David Spergel[4,1] Shirley Ho[4,3,1,5]

- ✗ Simulate Dark Matter
- ✗ Apply symbolic regression at the output of a graph neural network to recover force equation

Cranmer et al., Neurips 2020

Condensed Matter+Correlator Network

Correlator Convolutional Neural Networks: An Interpretable Architecture for Image-like Quantum Matter Data

Cole Miles,[1] Annabelle Bohrdt,[2,3,4] Ruihan Wu,[5] Christie Chiu,[2,6,7] Muqing Xu,[2] Geoffrey Ji,[2] Markus Greiner,[2] Kilian Q. Weinberger,[5] Eugene Demler,[2] and Eun-Ah Kim[1]

- ✗ Explicit feature engineering layer that probes for correlations
- ✗ Dominant features correspond to dominant correlations in condensed matter system

Miles et al., Arxiv 2020

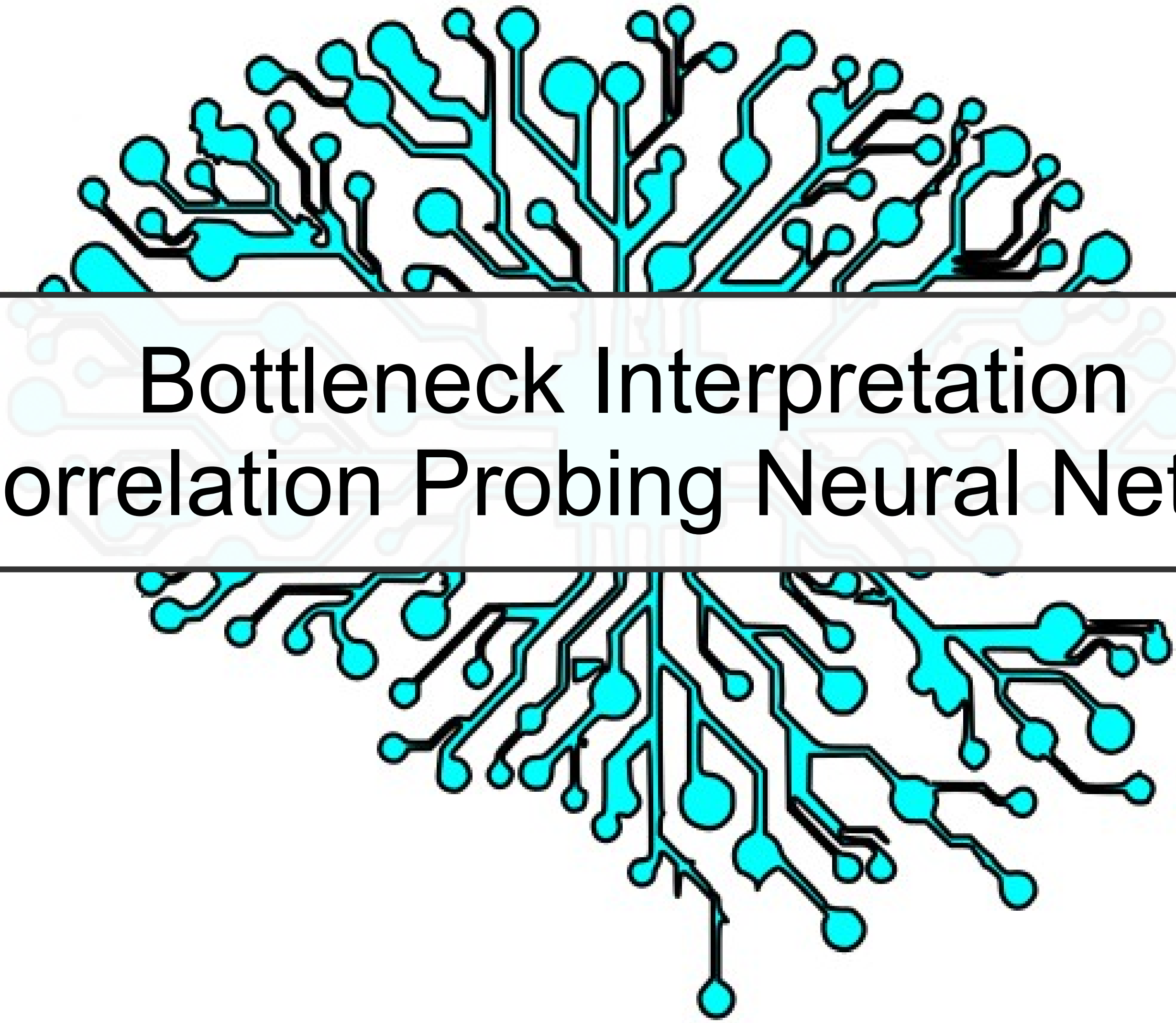
Physical Concepts

Discovering physical concepts with neural networks

Raban Iten,* Tony Metger,* Henrik Wilming, Lídia del Rio, and Renato Renner
ETH Zürich, Wolfgang-Pauli-Str. 27, 8093 Zürich, Switzerland.
(Dated: January 24, 2020)

- ✗ Interpretation of autoencoder latent representation
- ✗ Ask physical questions to be extractable from latent space

Iten et al., PRL 2020



Bottleneck Interpretation
+Correlation Probing Neural Network

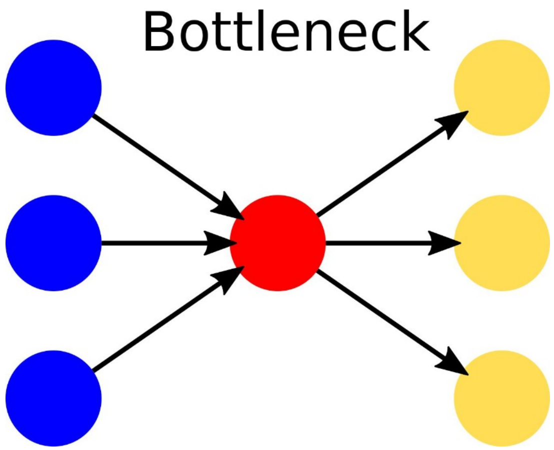
Bottleneck Interpretation

Interpretation is often difficult since information is spread over several neurons and layers

If the neuron contains the information of <u>one</u> single quantity/obervable $Q(S)$ $\longrightarrow$ The output of the neuron can be mapped via a bijective function to the observable

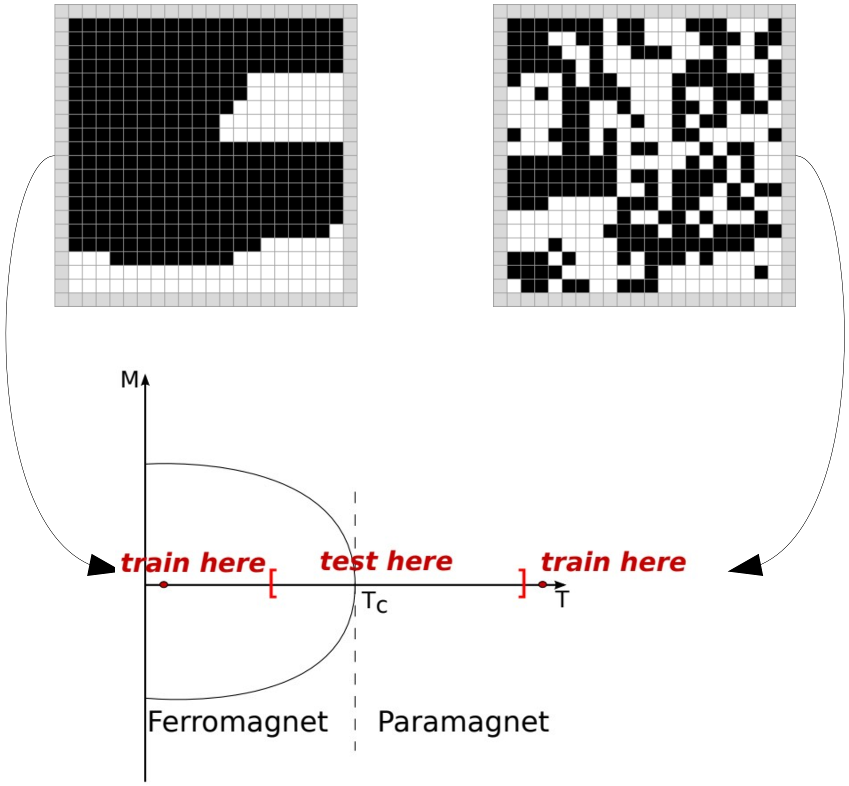
$$F(S) = f(Q(S))$$

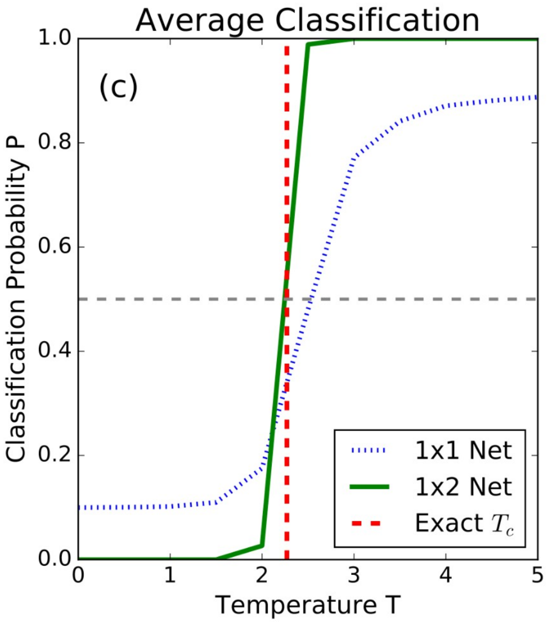
- Idea: identify or enforce bottlenecks in the network
- Perform regression on the output of the bottleneck neuron



Supervised Learning
2d Ising Model

- Data: Monte Carlo samples
- Training at well known points in phase diagram
- Labels: Phase
- Testing in interval containing phase transition
- Estimate within 1% of exact value $T_c = \dfrac{2}{\ln(1+\sqrt{2})}$





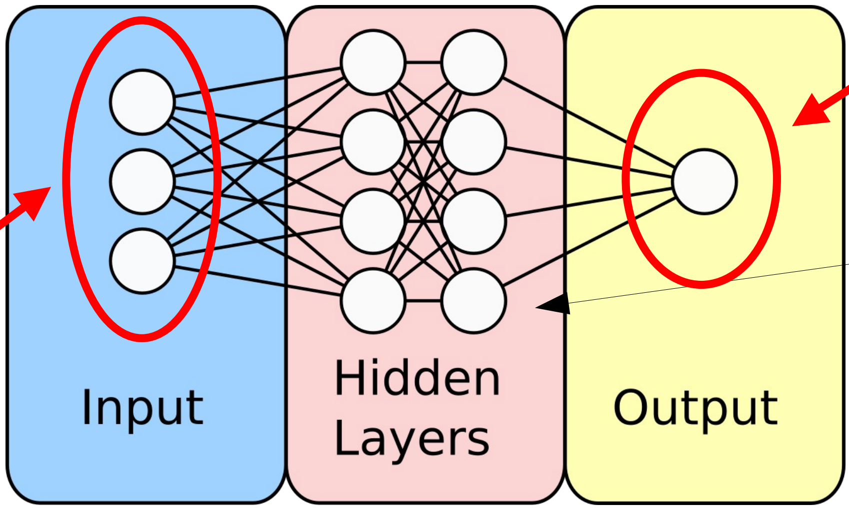
Carrasquila, Melko, Nature 2017

Artificial Neural Networks

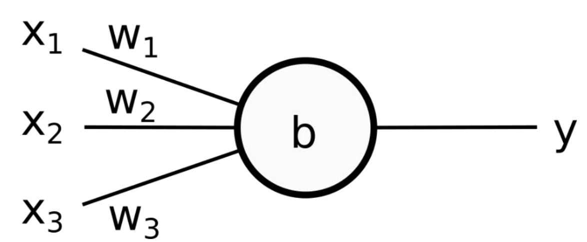
Too Many Features for Regression

Feed forward neural network

Natural Bottleneck



Perceptron



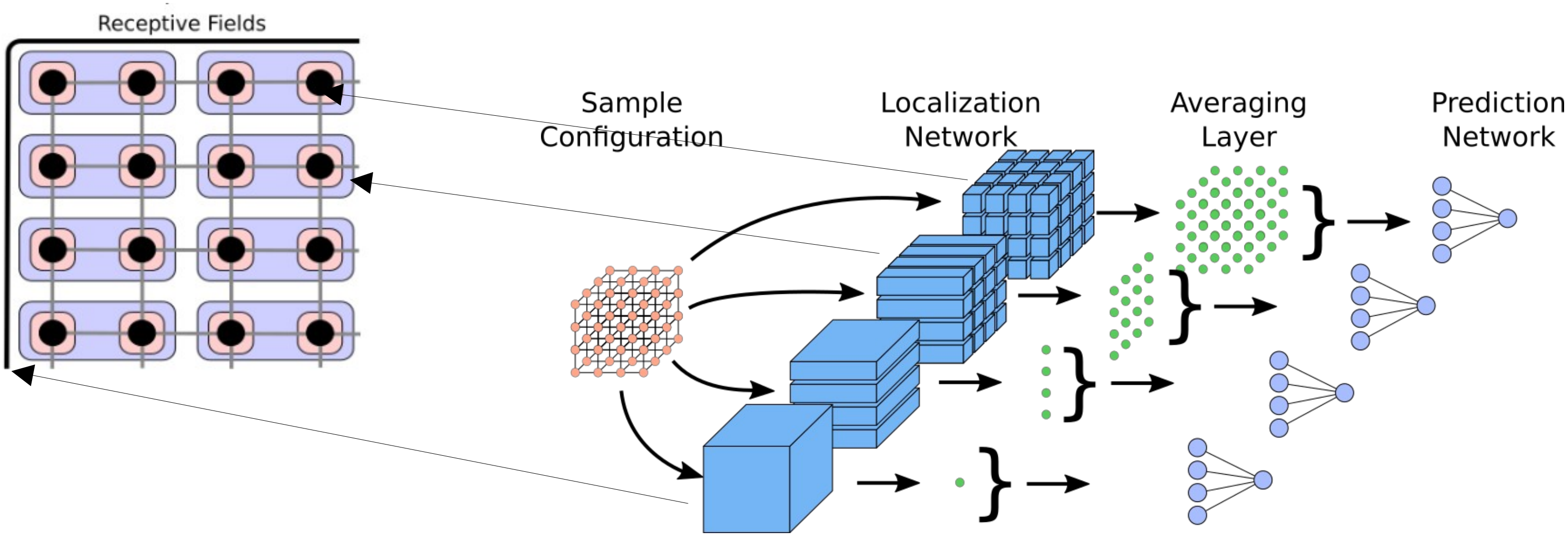
$y = f(\vec{x} \cdot \vec{w} + b)$

Input: Data $X = (\vec{x}_1, ..., \vec{x}_n)$, Label $Y = (y_1, ..., y_n)$

Output: $Y_{pred} = F(X, w_{ij}^L, b_i^L)$

Goal: choose w_{ij}^L and b_i^L such that $Y_{pred} \approx Y$

Interpretation of Neural Network
2d Ising Model



Interpretation Net

Wetzel, Scherzer, PRB 2017

- Interpretation Net interpolates between a general NN and a minimal optimal NN which has the same performance
- Interpretation by reducing the NN capacity in an ordered manner until one observes a performance drop
- Inspired by extensive physical quantities (averaging layer probes for translational invariance of the quantity $Q(S)$)

Interpretation of Neural Network
2d Ising Model

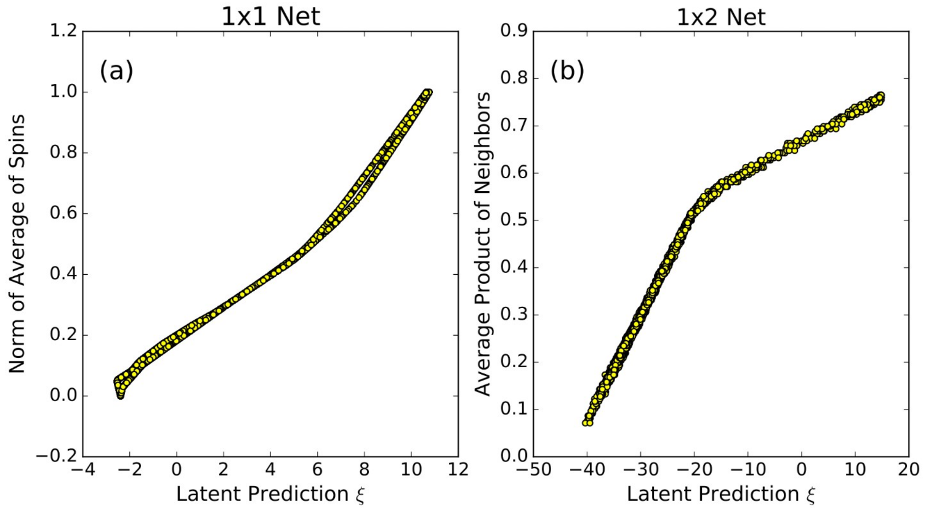
Receptive Field Size	Train Loss	Validation Loss
28×28	$6.1588e-04$	0.0232
1 × 2	**1.2559e-04**	**1.2105e-07**
1 × 1	**0.2015**	**0.1886**
baseline	0.6931	0.6931

Decision functions

$$F(S) = \mathrm{sigmoid}(w\,Q(S) + b)$$

- $Q(S) = |1/N \sum_i s_i|$ Magnetization
- $Q(S) = \dfrac{1}{N} \sum_{<i,j>_{nn}} s_i s_j$ Expected Energy per site

Deduction visually confirmed:

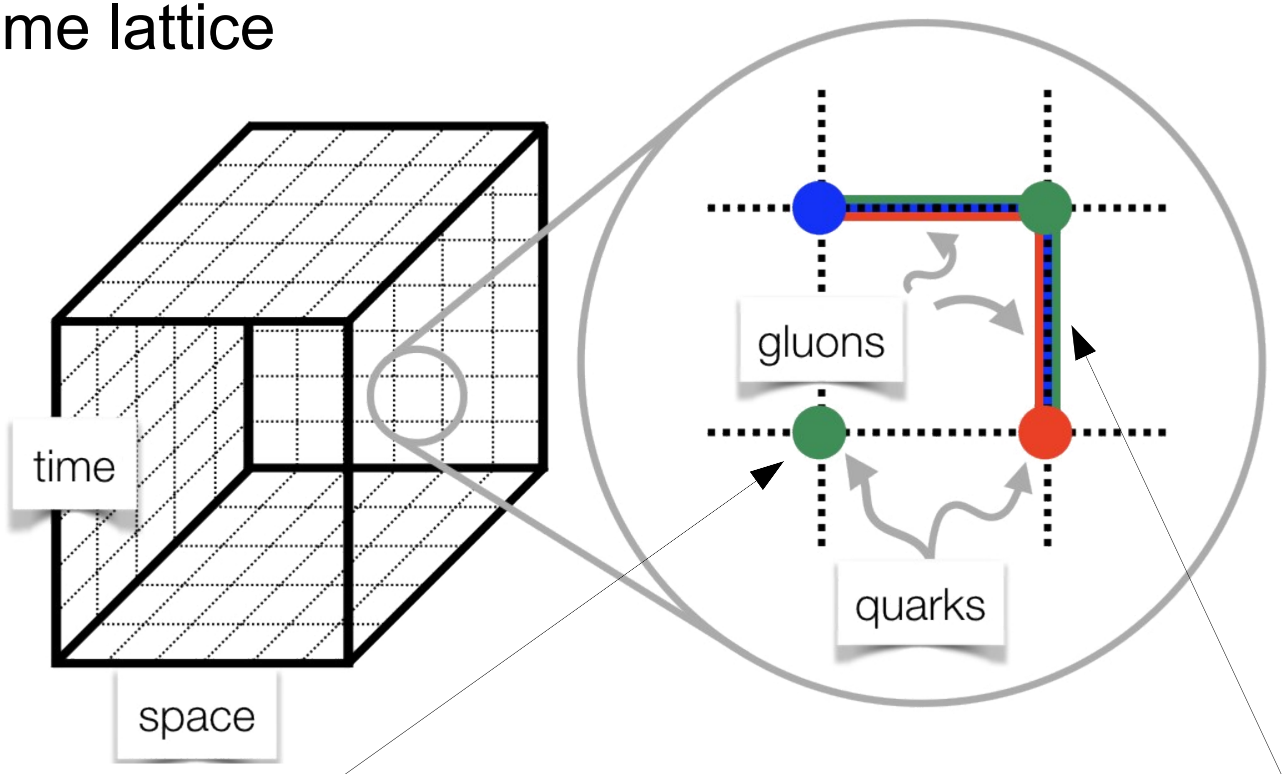


Note:

1x2 Network also has
the Magnetization minimum
which is easier to find!

SU(2) Lattice Gauge Theory

Space time lattice



Quarks on heavy static lattice sites.

Gluons on the connections between lattice sites are described by Matrices

$$\cancel{U_\mu^x \in SU(3)}$$
$$U_\mu^x \in SU(2)$$

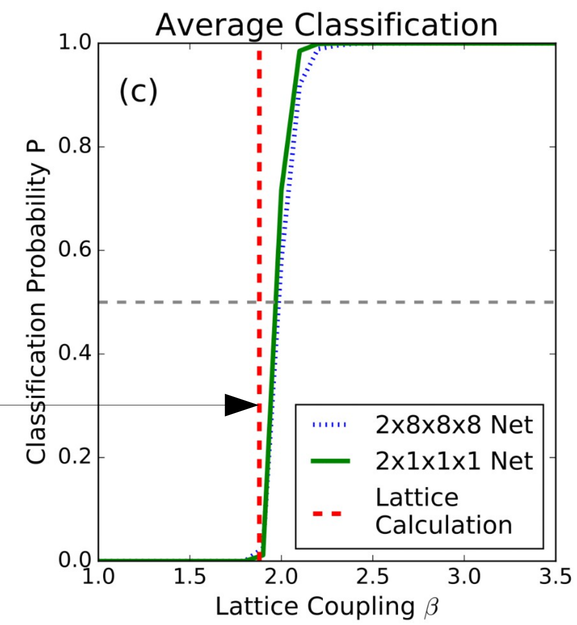
Supervised Learning
SU(2) Lattice Gauge Theory

Data: Monte Carlo samples

$$S_{\text{Wilson}}[U] = \beta_{\text{latt}} \sum_{x} \sum_{\mu<\nu} \text{Re tr}\left(1 - U^x_{\mu\nu}\right)$$

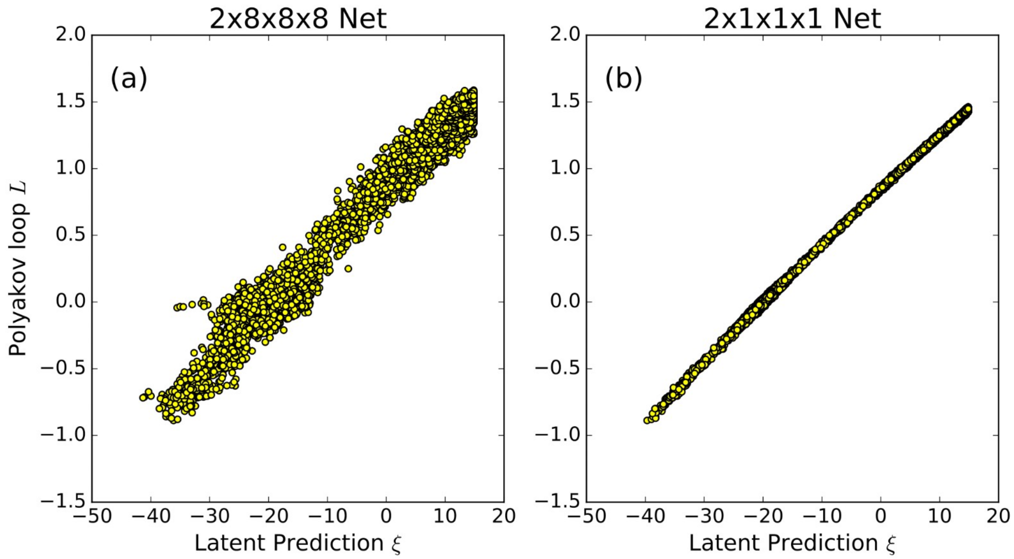
- Training at well known points in phase diagram
- Labels: Phase

Find phase transition close to lattice calculation



Interpretation of Neural Network
SU(2) Gauge Theory

Deduction confirmed by perfect correlation between NN output and Polyakov Loop order parameter

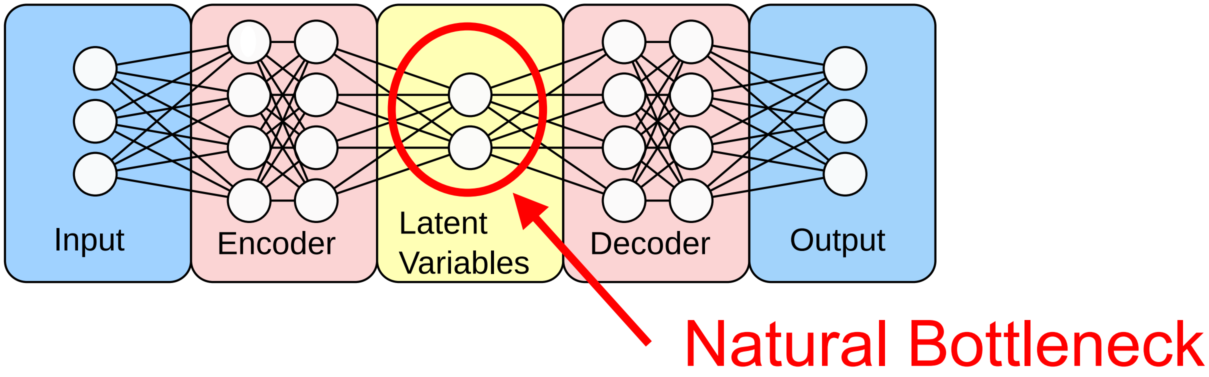


$$F(S) \approx \text{sigmoid}\left(w\left(\frac{2}{N}\sum_{\vec{x}} f(\{U_\mu^{x_0,\vec{x}}\})\right)+b\right)$$

$$f(\{U_\mu^{x_0}\}) = a_\tau^0 a_\tau^1 - b_\tau^0 b_\tau^1 - c_\tau^0 c_\tau^1 - d_\tau^0 d_\tau^1 = \mathrm{tr}\left(U_\tau^0 U_\tau^1\right)$$

- Polyakov Loop

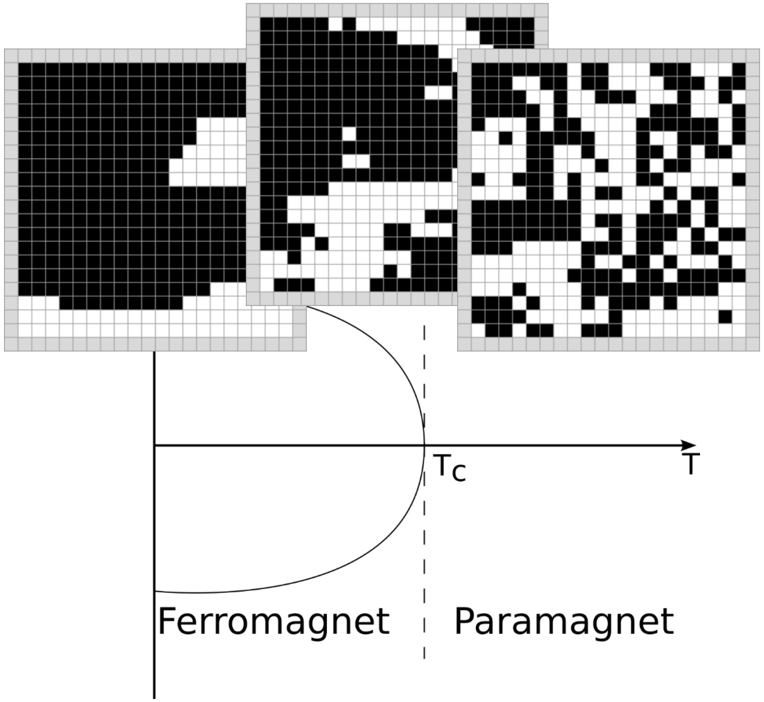
(Variational) Autoencoder
2d Ising Model



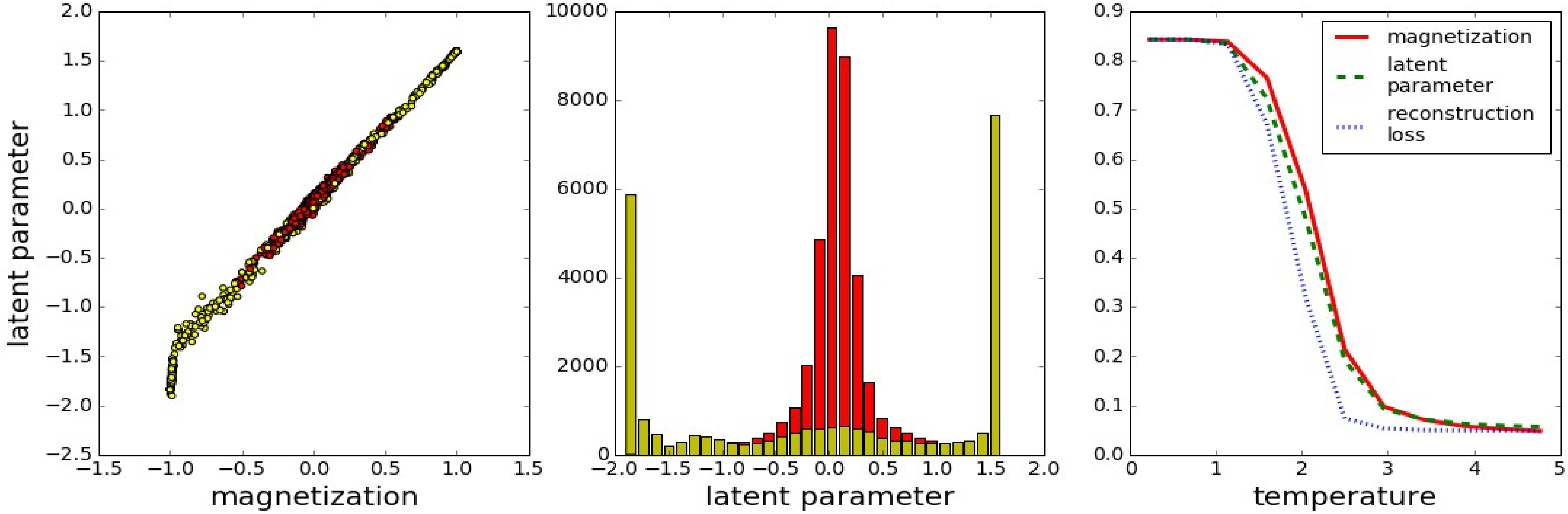
Objective: Minimize Reconstruction error

$$MSE = \frac{1}{N} \sum_k \left\| x_k - F(x_k) \right\|^2$$

- Data: Monte Carlo samples
- Train everywhere in phase diagram
- Labels: None



(Variational) Autoencoder
2d Ising Model

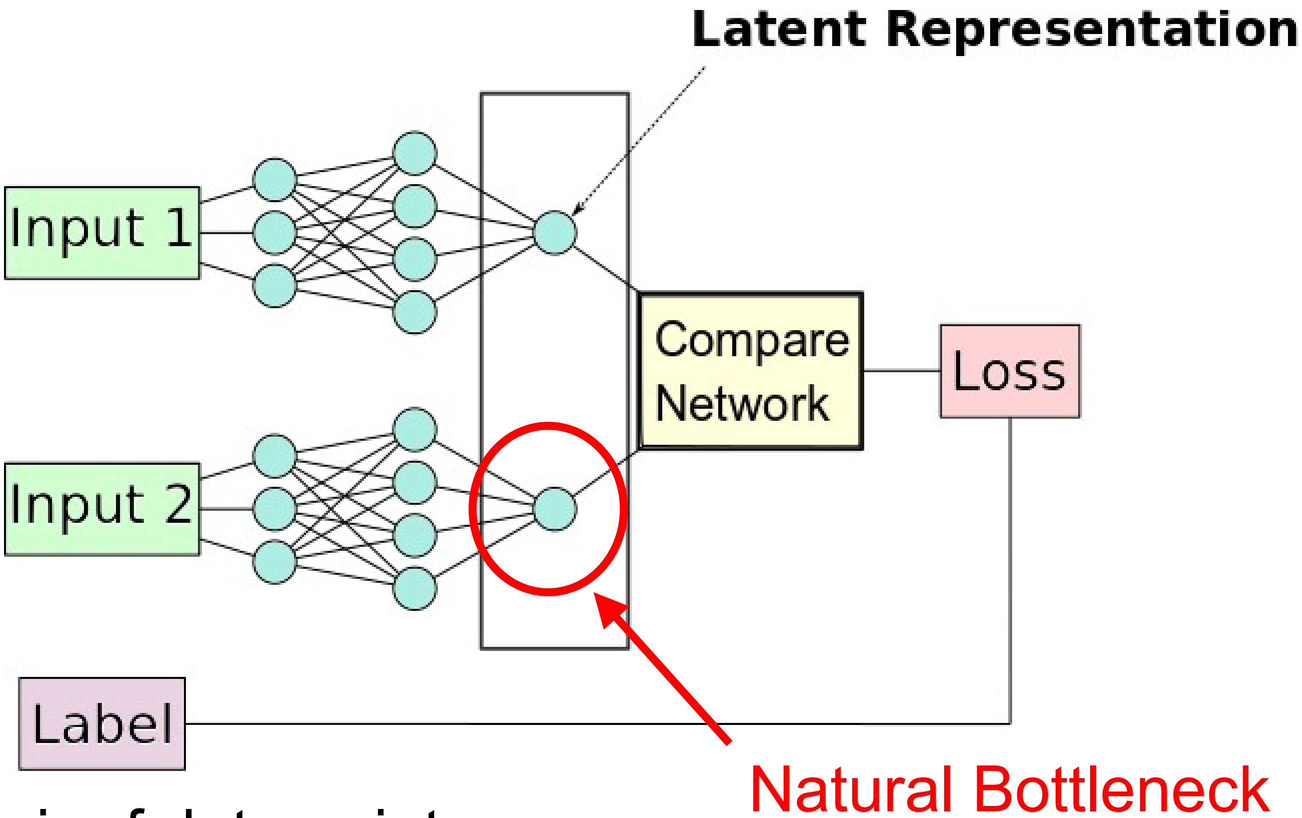


Ferromagnetic Ising model on the square lattice

Wetzel, PRE 2017

- Latent parameter corresponds to magnetization
- Identification of phases: Latent representations are clustered
- Location of phases: Magnetization, latent parameter and reconstruction loss show a steep change at the phase transition.

Siamese Neural Networks

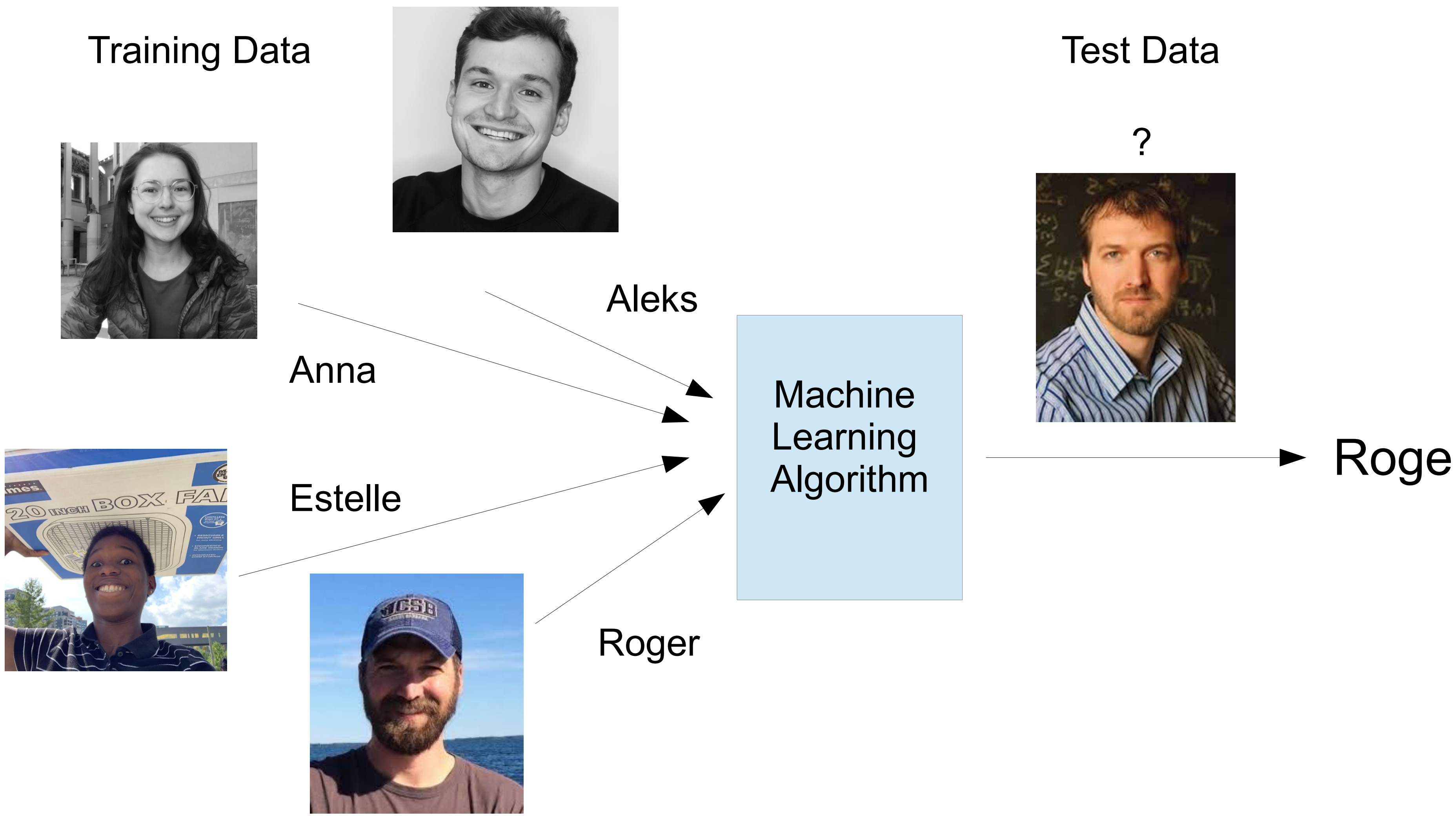


- Input : Pair of data points
- Label : same / different
- Network pair contains identical neural networks with shared weights

Wetzel, et al, PRR 2020

Machine Learning
Multi Class Classification

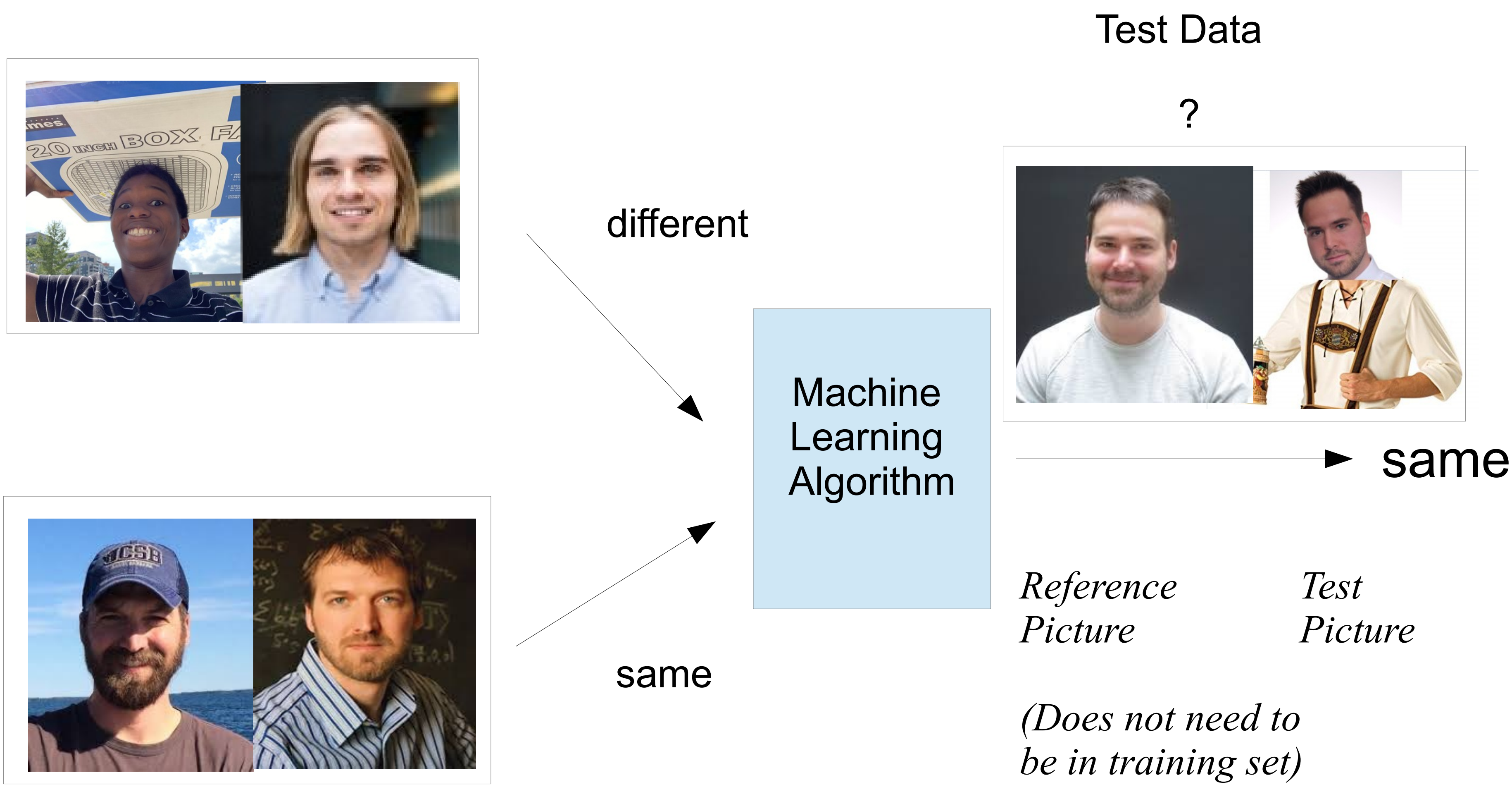
„Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed.“ - Wikipedia



Machine Learning
Infinite Class Classification

Reformulation of the Problem:

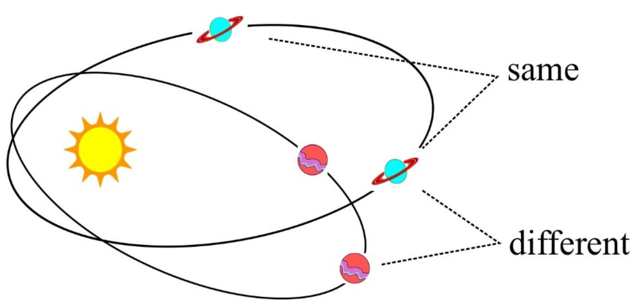
- Teach a maching learning algorithm if two pictures show the same class.



Siamese Neural Networks
Particle in Gravitational Potential

Problem:

- Given two observations of positions and velocities, do they belong to the same particle trajectory?



SNN Solution:

- Prepare Dataset of positive data where the pair is connected by solving the equations of motion

$$((x, y, v_x, v_y), (x', y', v'_x, v'_y))$$

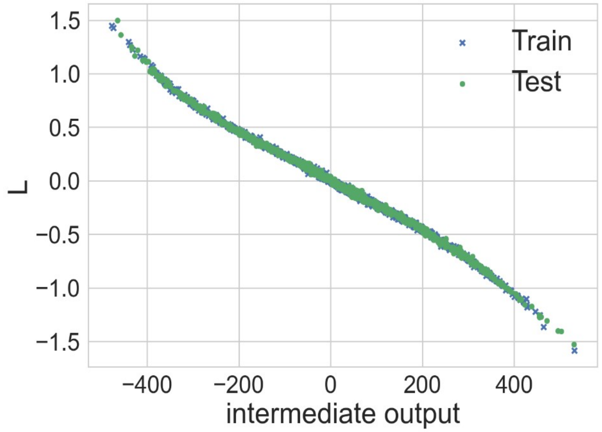
- Prepare Negative Dataset by permuting positive dataset
- Train SNN to distinguish between positive and negative pairs

Siamese Neural Networks
Particle in Gravitational Potential

Results:

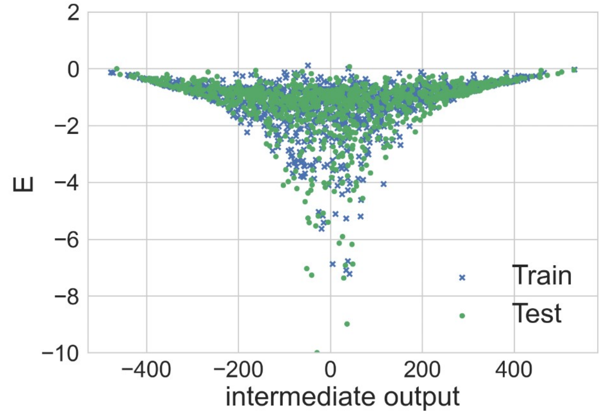
Training accuracy : 98%

Test accuracy : 97%



- Interpretation by polynomial regression on latent representation:

$$\begin{aligned} f(\mathbf{x}) \approx & -403.71 x v_y - 4.85 x - 0.58 x y \\ & -0.17 x v_x - 0.02 v_y^2 - 0.01 v_x v_y \\ & +0.00 v_y^2 + 0.01 v_y + 0.02 v_x \\ & +0.45 x^2 + 0.66 y^2 + 0.74 \\ & +0.99 y v_y + 1.24 y + 402.44 y v_x \\ \approx & -403 \underbrace{\left(x v_y - y v_x\right)}_{=L_z} \end{aligned}$$

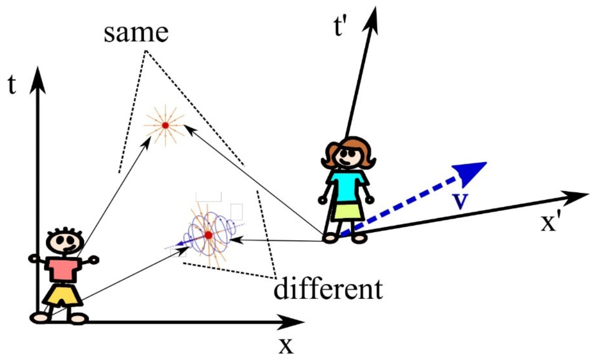


- Network has learned the angular momentum to infer its prediction.

Siamese Neural Networks
Lorentz Transformation of Electromagnetic Fields

Problem:

- Given two field configurations, can they be transformed into each other by a Lorentz transformation?



SNN Solution:

- Prepare Dataset of positive data where the pair is connected by a Lorentz Transformation

$$((E_x, E_y, E_z, B_x, B_y, B_z), (E'_x, E'_y, E'_z, B'_x, B'_y, B'_z))$$

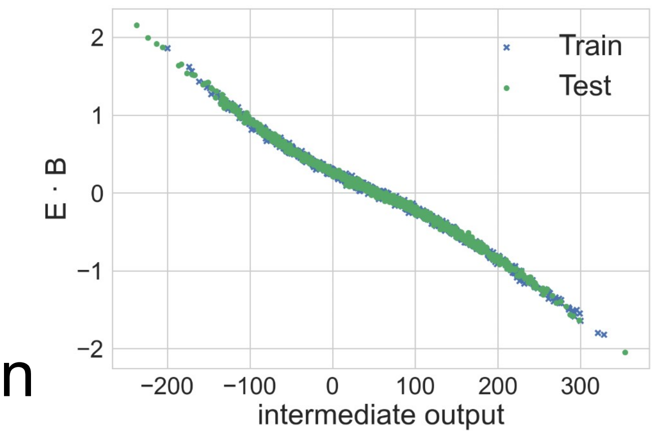
- Prepare Negative Dataset by permuting poitive dataset
- Train SNN to distinguish between positive and negative pairs

Siamese Neural Networks
Lorentz Transformation of Electromagnetic Fields

Results:

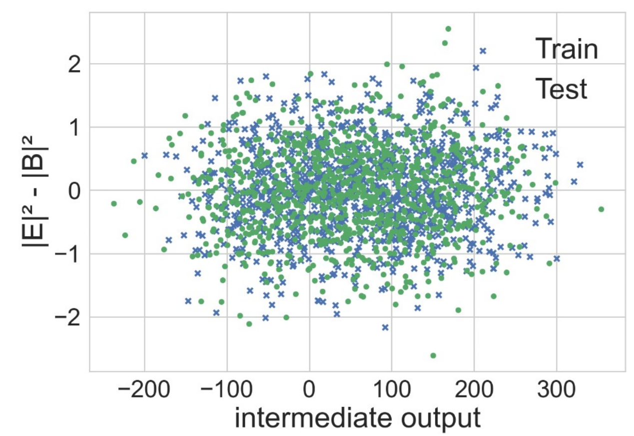
Training accuracy : 95%

Test accuracy : 94%



- Interpretation by polynomial regression on latent representation:

$$f(\mathbf{x}) \approx -170.53E_2B_2 - 170.22E_1B_1 - 170.20E_3B_3 - 4.13B_3^2 + \ldots + 4.92E_2^2 + 53.43$$
$$\approx -170\,\underbrace{(E_1B_1 + E_2B_2 + E_3B_3)}_{=E\cdot B} + 53$$



- Network has learned the determinant of the field strength tensor to infer its prediction.

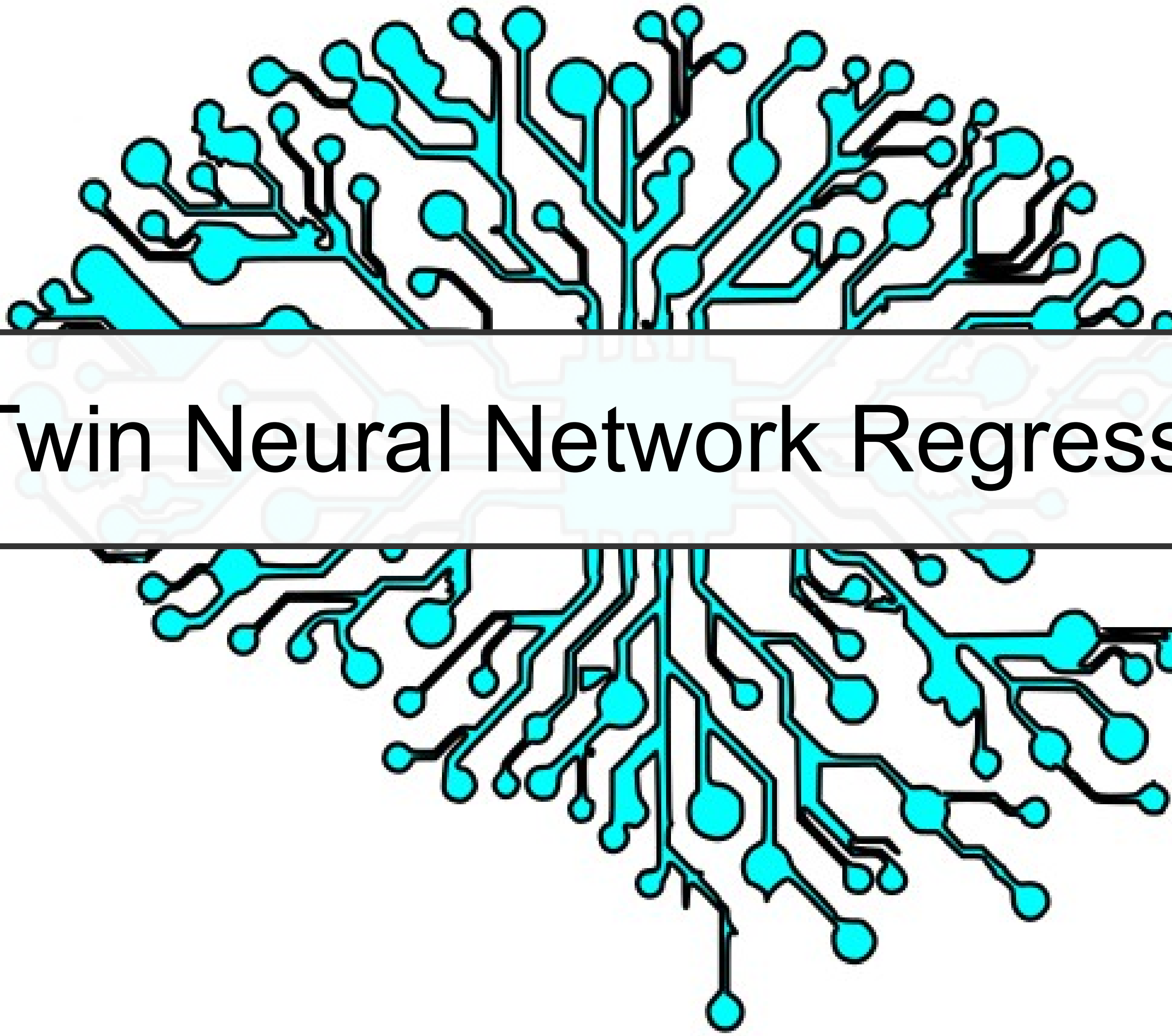
Summary

- Interpretation of Artificial Neural Networks is hard because information is distributed among many layers and neurons
- Interpretation is possible by identifying bottlenecks and performing regression
- Interpretation is constructive and can give insight into the underlying physics:

  Neural Networks applied to phase recognition learn order parameters or energies

  Siamese Networks for similarity detection learn invariants or conserved quantities

Twin Neural Network Regression



Overview

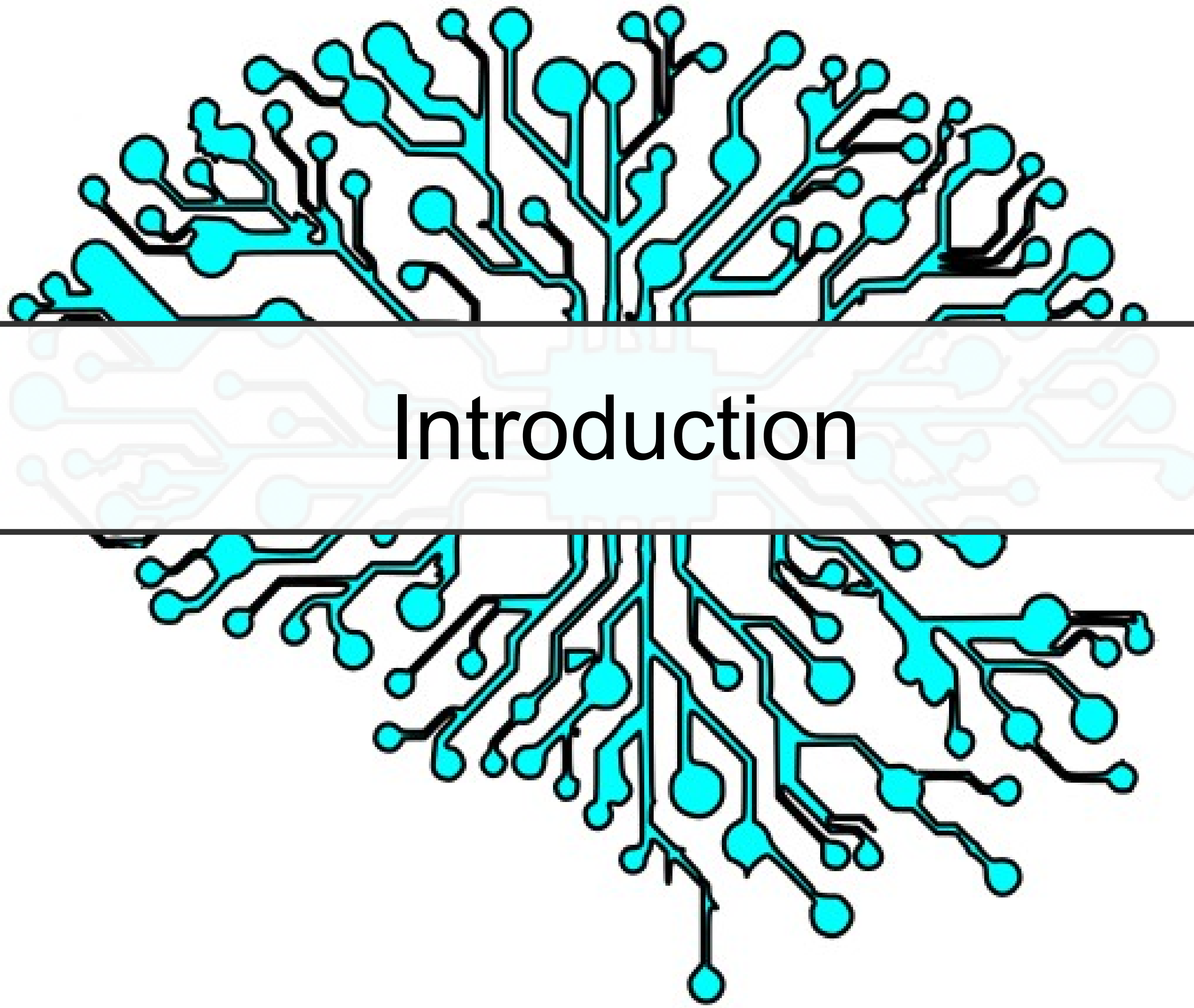
Introduction:

- ✗ Regression
- ✗ Limits of Traditional Algorithms

Twin Neural Network Regression:

- ✗ Circumventing Bias Variance Tradeoff
- ✗ Uncertainty Signal
- ✗ Semi Supervised Training

Introduction



Regression

Regression assumes there exists a true function with noise that models the relation between features and targets.

$$y = f(x) + \varepsilon$$

Using the information contained in a training data set D the goal is to estimate a function

$$\hat{f}\,(x; D)$$

that minimizes the error between prediction and true target

$$(y - \hat{f}\,(x; D))^2$$

on certain unlabelled test data.

Regression Algorithms

Regression Algorithms List

1. Linear Regression
2. Polynomial Regression
3. Poisson Regression
4. Ordinary Least Squares (OLS) Regression
5. Ordinal Regression
6. Support Vector Regression
7. Gradient Descent Regression
8. Stepwise Regression
9. Lasso Regression
10. Ridge Regression
11. Elastic Net Regression
12. Bayesian Linear Regression
13. Least-Angled Regression (LARS)
14. Neural Network Regression
15. Locally Estimated Scatterplot Smoothing (LOESS)
16. Multivariate Adaptive Regression Splines (MARS)
17. Locally Weighted Regression (LWL)
18. Quantile Regression
19. Principal Component Regression (PCR)
20. Partial Least Squares Regression

Regression Wishlist

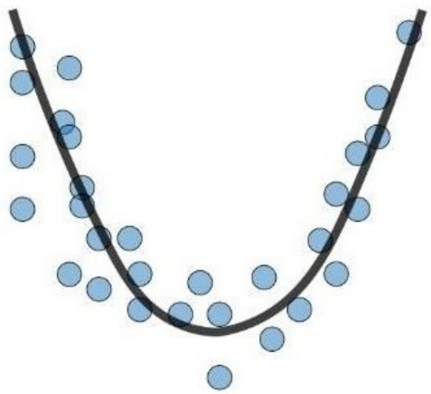
- We have all these nice regression algorithms, why do we need more?
- Who cares if you invent a new algorithm that performs equally well as the mentioned ones?
- Instead identify the limits of these algorithms and overcome them.

Regression Wishlist

People are looking for accurate and reliable solutions

Accurate: low average Mean Squared Error

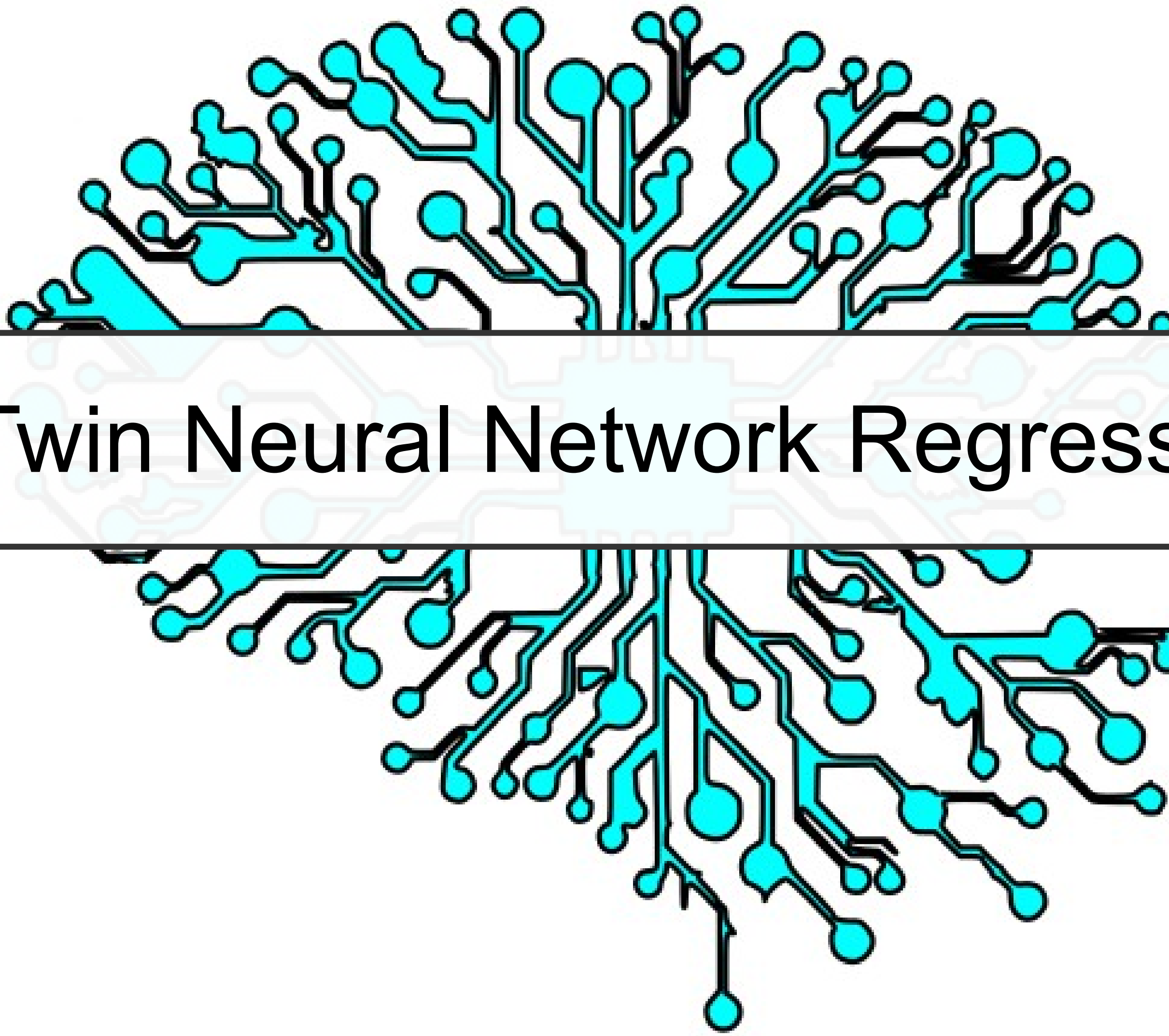
- Limited by Bias-Variance Tradeoff
- Limited by Available Labelled Training Data



Reliable: knowing when the model is incorrect

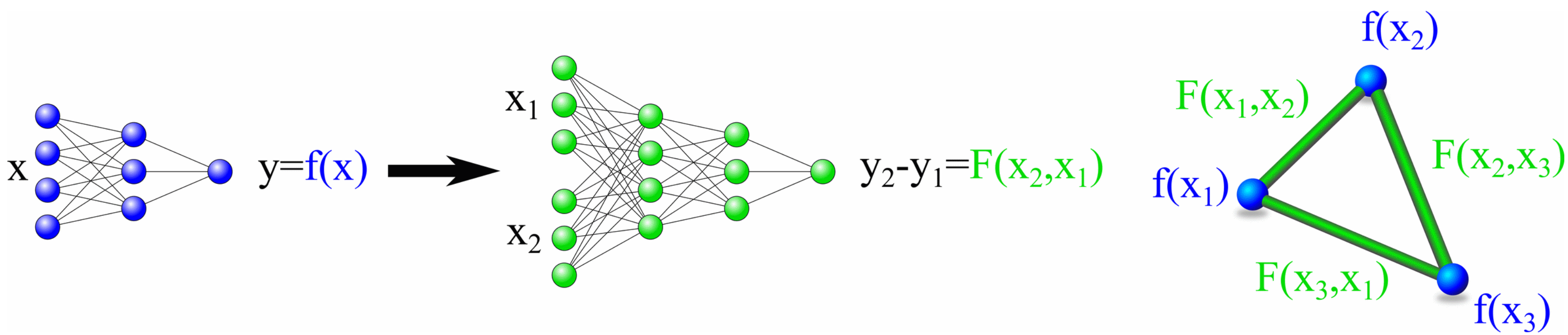
- Requires Uncertainty Measure

Twin Neural Network Regression



Twin Neural Network Regression

Solution to Limitations: Solve different Problem



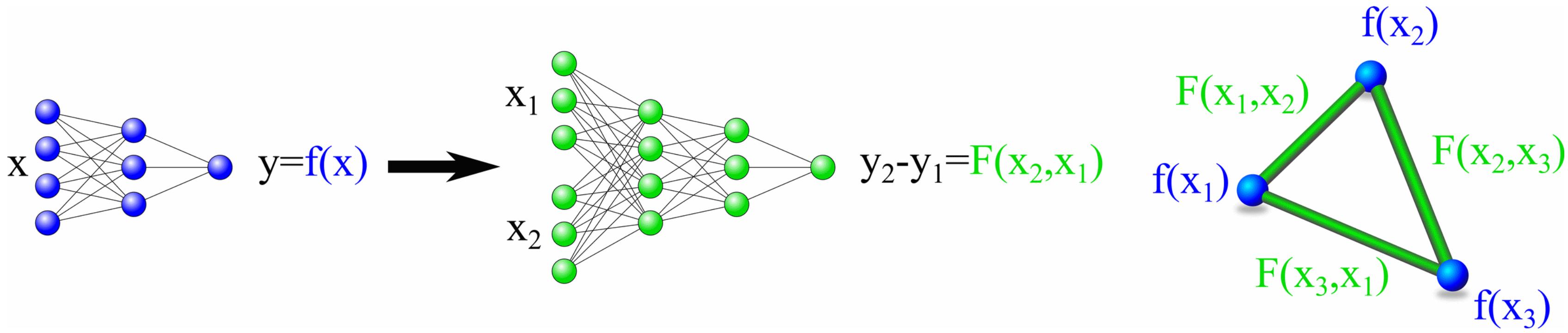
Inputs are Pairs of Data Points

Based on Artificial Neural Networks

- Highest Performance Ceiling (Universal Approximation Theorem)
- Modular Architectures Allow for Adaption to Specific Problem
- Scales well with Number of Input Features (Important for Pairs)
- High Variance + Low Bias

Loop Structure in Predictions

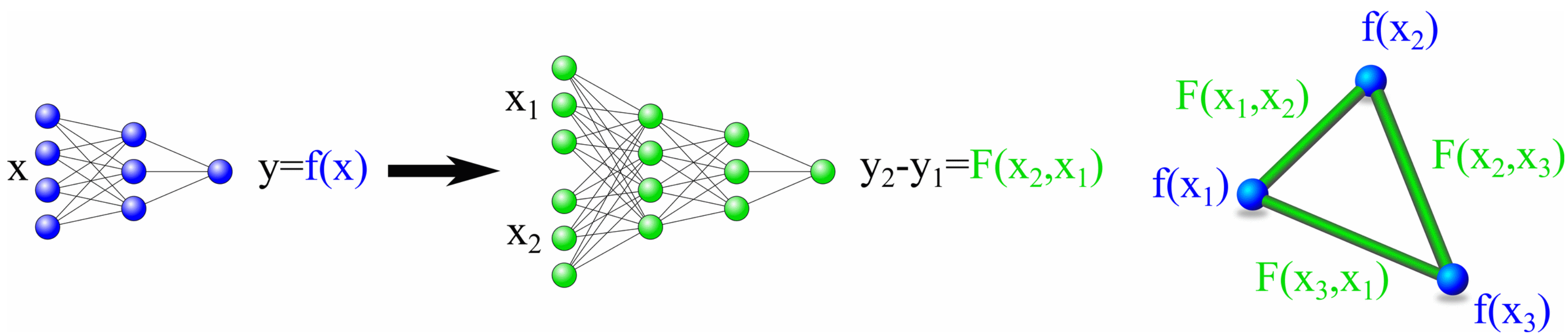
Twin Neural Network Regression



- Solution of the Original Regression Problem:

$$y_2 = F(x_2, x_1) + y_1$$

Twin Neural Network Regression



- Solution of the Original Regression Problem:

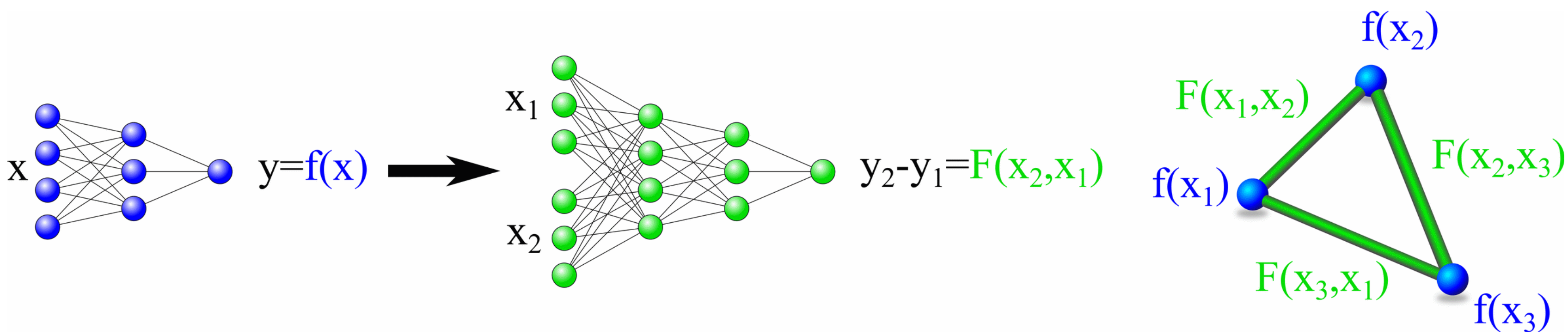
$$y_2 = F(x_2, x_1) + y_1$$

Solution to original problem

Unknown data point

Known training data point

Twin Neural Network Regression



- Solution of the Original Regression Problem:

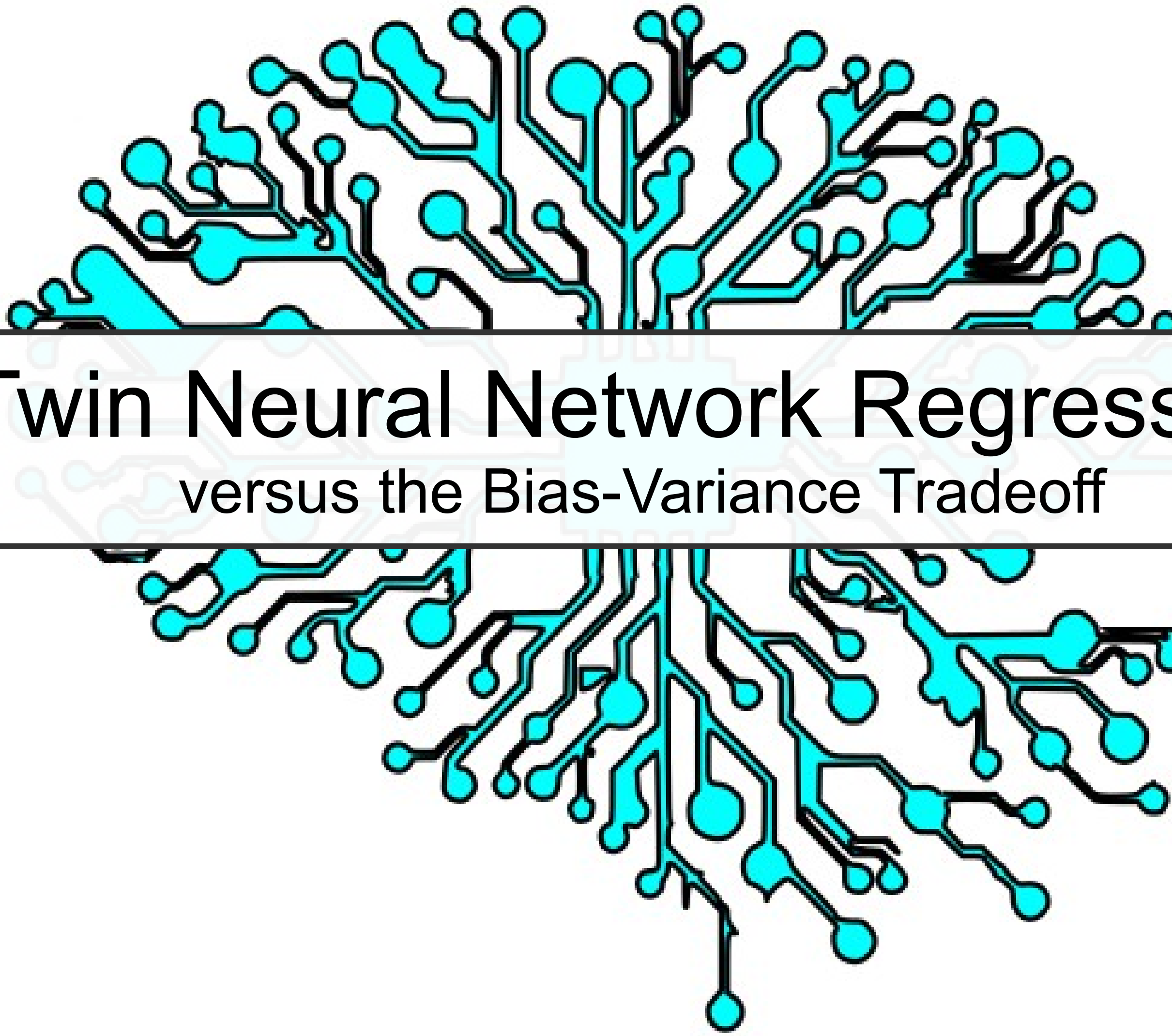
$$y_2 = F(x_2, x_1) + y_1$$

twice the input data size 😟

twice the label noise 😒

square the training data size 🤓

but then training scales quadratically ☹️

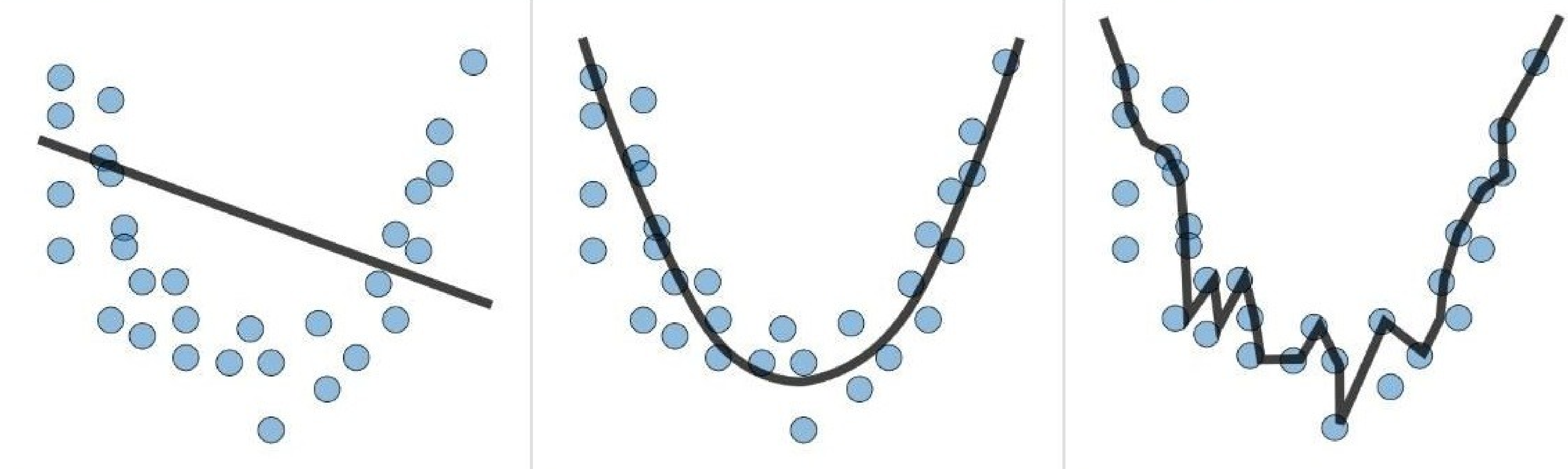


Twin Neural Network Regression

versus the Bias-Variance Tradeoff

Bias-Variance Tradeoff

Underfitting	Just right	Overfitting
• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance



Bias-Variance Tradeoff

In mathematical form provides an expectation for the Mean Square Error.

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right] \right\} + \sigma^2$$

Bias-Variance Tradeoff

In mathematical form provides an expectation for the Mean Square Error.

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right] \right\} + \sigma^2$$

Bias Error

Model too restricted

Too few free parameters

Variance Error

Model too general

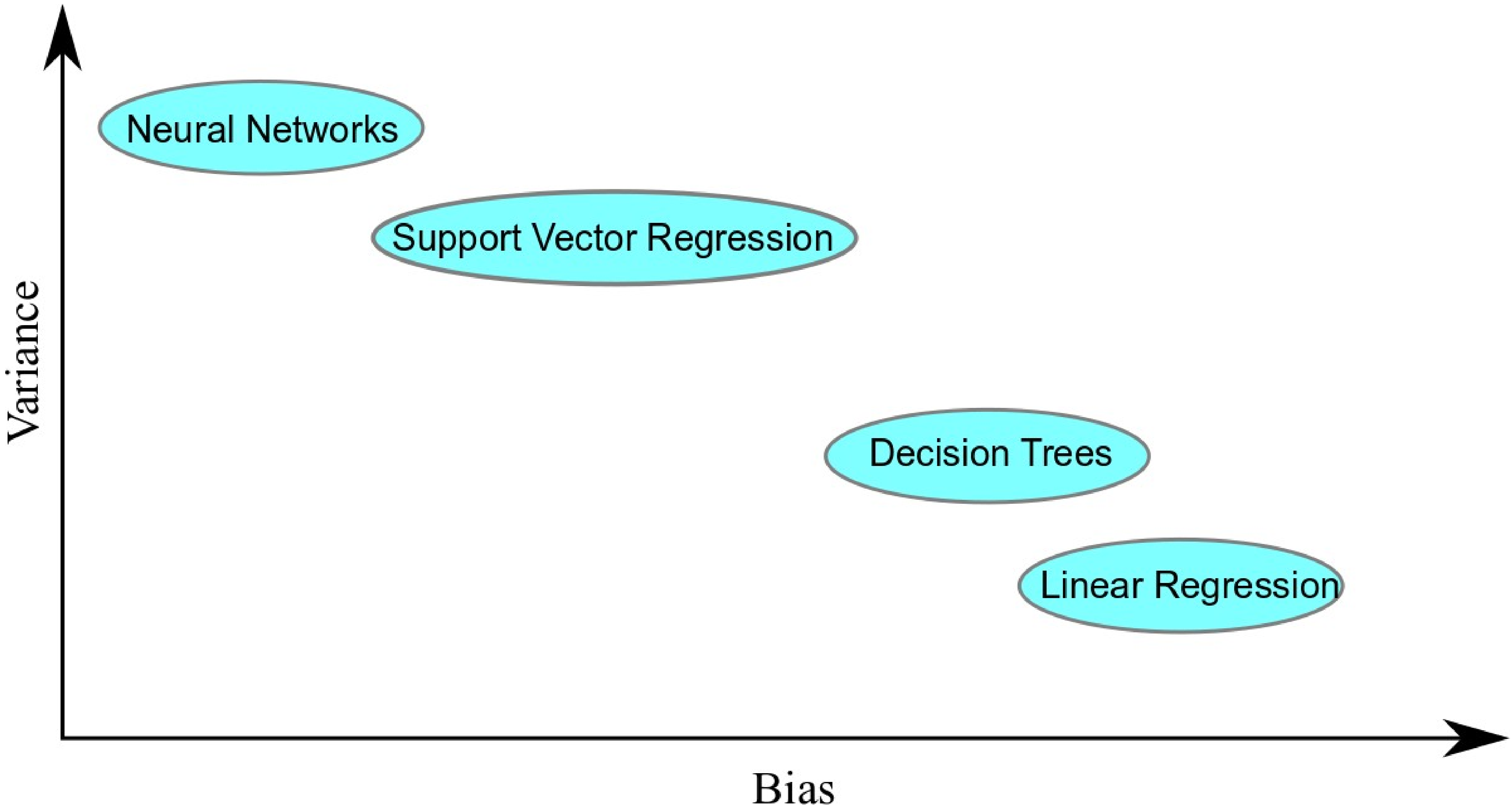
Too many free parameters

Data Error

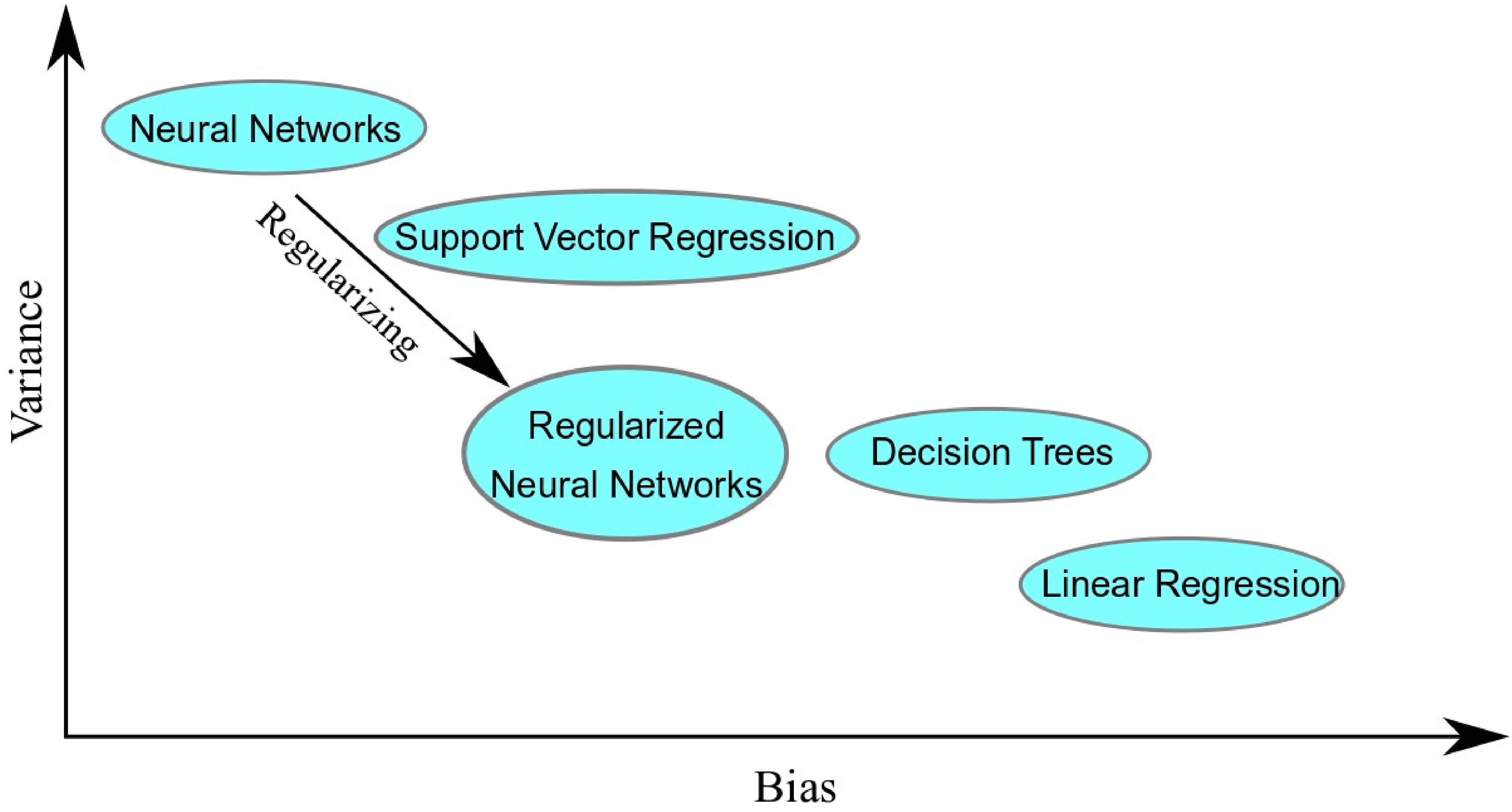
Noise

Intrinsic to Dataset

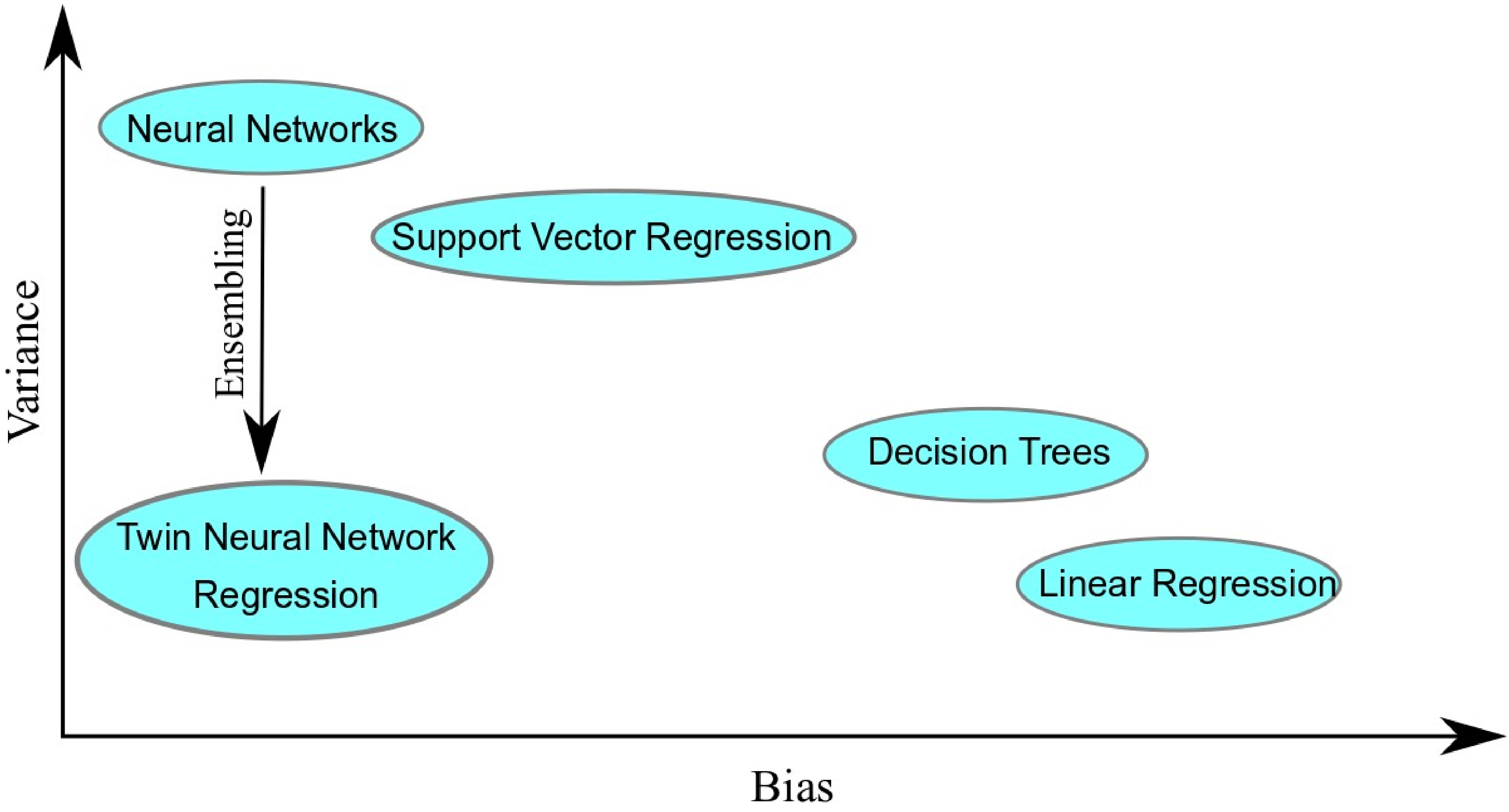
Bias-Variance Tradeoff



Bias-Variance Tradeoff



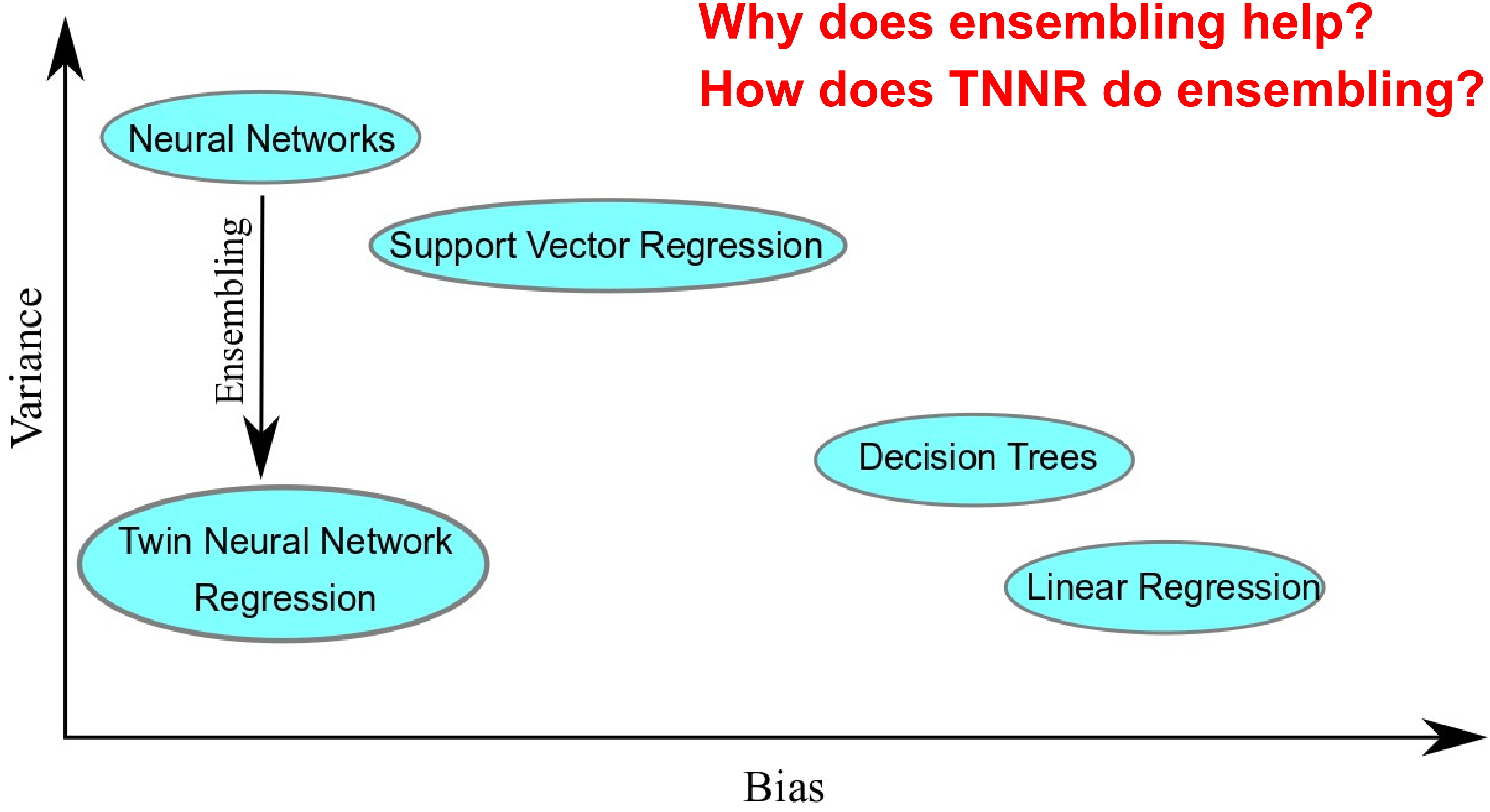
Bias-Variance Tradeoff



Bias-Variance Tradeoff

Why does ensembling help?

How does TNNR do ensembling?



Bias-Variance Tradeoff

Effects of Ensembling on Bias-Variance Tradeoff:

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right] \right\} + \sigma^2$$

Let us assume the final prediction is generated by an ensemble of two different solutions from similar models

$$\hat{f}(x;D) = 1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)$$

This let's us rewrite the Bias-Variance Tradeoff

Bias-Variance Tradeoff

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}[1/2\hat{f}_A + 1/2\hat{f}_B]^2 + \mathrm{Var}\left[1/2\hat{f}_A + 1/2\hat{f}_B\right] \right\} + \sigma^2$$

$$= \mathrm{E}_x \left\{ \mathrm{Bias}[\hat{f}]^2 + \mathrm{Var}\left[1/2\hat{f}_A\right] + \mathrm{Var}\left[1/2\hat{f}_B\right] + 2\,\mathrm{Cov}\left[1/2\hat{f}_A, 1/2\hat{f}_B\right] \right\} + \sigma^2$$

$$= \mathrm{E}_x \left\{ \mathrm{Bias}[\hat{f}]^2 + 1/2\,\mathrm{Var}\left[\hat{f}\right] + 1/2\,\mathrm{Cov}\left[\hat{f}_A, \hat{f}_B\right] \right\} + \sigma^2$$

Bias-Variance Tradeoff

$$\mathrm{MSE} = \mathrm{E}_x \left\{ \mathrm{Bias}[1/2\hat{f}_A + 1/2\hat{f}_B]^2 + \mathrm{Var}\left[1/2\hat{f}_A + 1/2\hat{f}_B\right] \right\} + \sigma^2$$

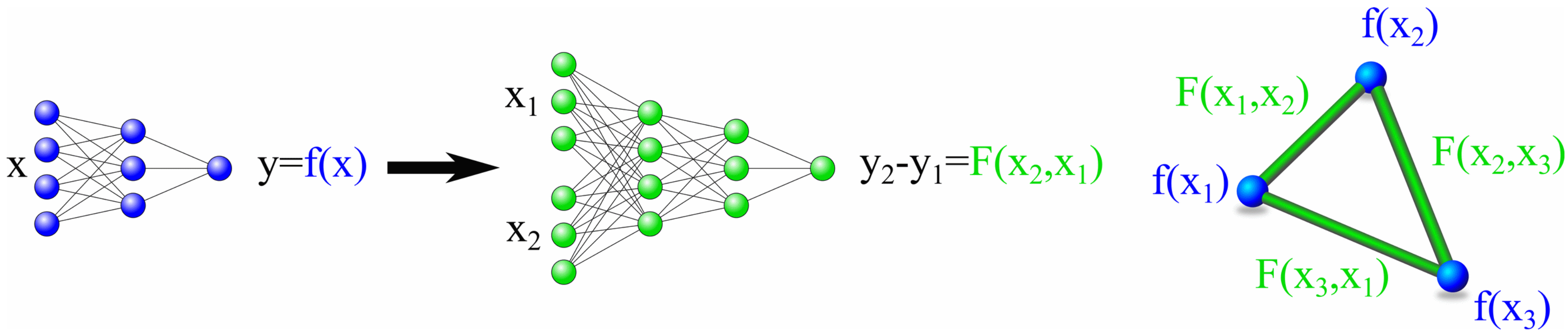
$$= \mathrm{E}_x \left\{ \mathrm{Bias}[\hat{f}]^2 + \mathrm{Var}\left[1/2\hat{f}_A\right] + \mathrm{Var}\left[1/2\hat{f}_B\right] + 2\,\mathrm{Cov}\left[1/2\hat{f}_A, 1/2\hat{f}_B\right] \right\} + \sigma^2$$

$$= \mathrm{E}_x \left\{ \mathrm{Bias}[\hat{f}]^2 + 1/2\,\mathrm{Var}\left[\hat{f}\right] + 1/2\,\mathrm{Cov}\left[\hat{f}_A, \hat{f}_B\right] \right\} + \sigma^2$$

If the ensemble members are uncorrelated the covariance vanishes.

- Typically, ensemble members are correlated.
- Pseudo ensembles can be generated by perturbing weights of a neural network
- Real ensembles can be generated by retraining using different initializations or different parts of the training data

Bias-Variance Tradeoff



TNN implicit ensemble

$$y_i^{pred} = \frac{1}{m}\sum_{j=1}^{m} F(x_i, x_j^{train}) + y_j^{train} = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{2}F(x_i, x_j^{train}) - \frac{1}{2}F(x_j^{train}, x_i) + y_j^{train}$$

- Get huge ensemble of twice the training data set size
- Ensemble is relatively uncorrelated, since the predicted differences are different by construction

Bias Variance Tradeoff

Common Data	BH	CS	EE	YH	WN	BC
RF	4.24 ± 0.29	8.23 ± 0.24	2.22 ± 0.08	2.95 ± 0.46	0.64 ± 0.02	$\mathbf{0.71\pm0.03}$
XGB	2.93 ± 0.18	4.37 ± 0.19	1.17 ± 0.04	$\mathbf{0.42\pm0.06}$	$\mathbf{0.61\pm0.01}$	$\mathbf{0.70\pm0.03}$
ANN	3.09 ± 0.14	5.37 ± 0.17	0.98 ± 0.03	0.52 ± 0.07	0.64 ± 0.01	0.76 ± 0.02
ANNE	3.43 ± 0.32	5.14 ± 0.21	0.89 ± 0.04	$\mathbf{0.43\pm0.05}$	$\mathbf{0.62\pm0.01}$	$\mathbf{0.72\pm0.03}$
MCD	2.95 ± 0.15	6.07 ± 0.21	2.96 ± 0.12	1.42 ± 0.18	0.68 ± 0.01	$\mathbf{0.72\pm0.03}$
TNN	$\mathbf{2.55\pm0.10}$	4.19 ± 0.25	0.52 ± 0.02	0.49 ± 0.07	$\mathbf{0.62\pm0.01}$	0.83 ± 0.03
TNNE	$\mathbf{2.61\pm0.20}$	$\mathbf{3.88\pm0.22}$	$\mathbf{0.46\pm0.02}$	$\mathbf{0.37\pm0.06}$	0.63 ± 0.01	$\mathbf{0.72\pm0.02}$

Science Data	RP	RCL	WSB
RF	0.604 ± 0.013	0.288 ± 0.004	0.141 ± 0.011
XGB	0.229 ± 0.005	0.124 ± 0.002	0.071 ± 0.006
ANN	0.050 ± 0.002	0.019 ± 0.000	0.047 ± 0.004
ANNE	0.032 ± 0.002	0.016 ± 0.001	0.031 ± 0.002
MCD	0.086 ± 0.002	0.033 ± 0.001	0.042 ± 0.003
TNN	0.022 ± 0.001	0.017 ± 0.000	$\mathbf{0.020\pm0.001}$
TNNE	$\mathbf{0.016\pm0.001}$	$\mathbf{0.014\pm0.001}$	0.022 ± 0.002

Image Data	ISING
RF	0.601 ± 0.003
XGB	0.144 ± 0.003
CNN	0.050 ± 0.001
CNNE	0.044 ± 0.001
CMCD	0.052 ± 0.001
CTNN	0.035 ± 0.001
CTNNE	$\mathbf{0.030\pm0.001}$

Table 1. Best estimates for root mean square errors (RMSEs) of different algoritms on the test sets belonging to different data sets. The Lowest RMSEs are in bold for clarity. Our confidence on the RMSEs is determined by their standard error. Data sets: Boston housing (BH), concrete strength (CS), energy efficiency (EE), yacht hydrodynamics (YH), red wine quality (WN), Bio Conservation (BC), random polynomial (RP), RCL circuit (RCL), Wheatstone bridge (WSB) and the Ising Model (ISING). Algorithms: Random Forests (RF), xgboost (XGB), Neural Networks (ANN), Monte-Carlo Dropout networks (MCD), Twin Neural Networks (TNN) and ensembles (E) or convolutional variants (C).

Bias Variance Tradeoff

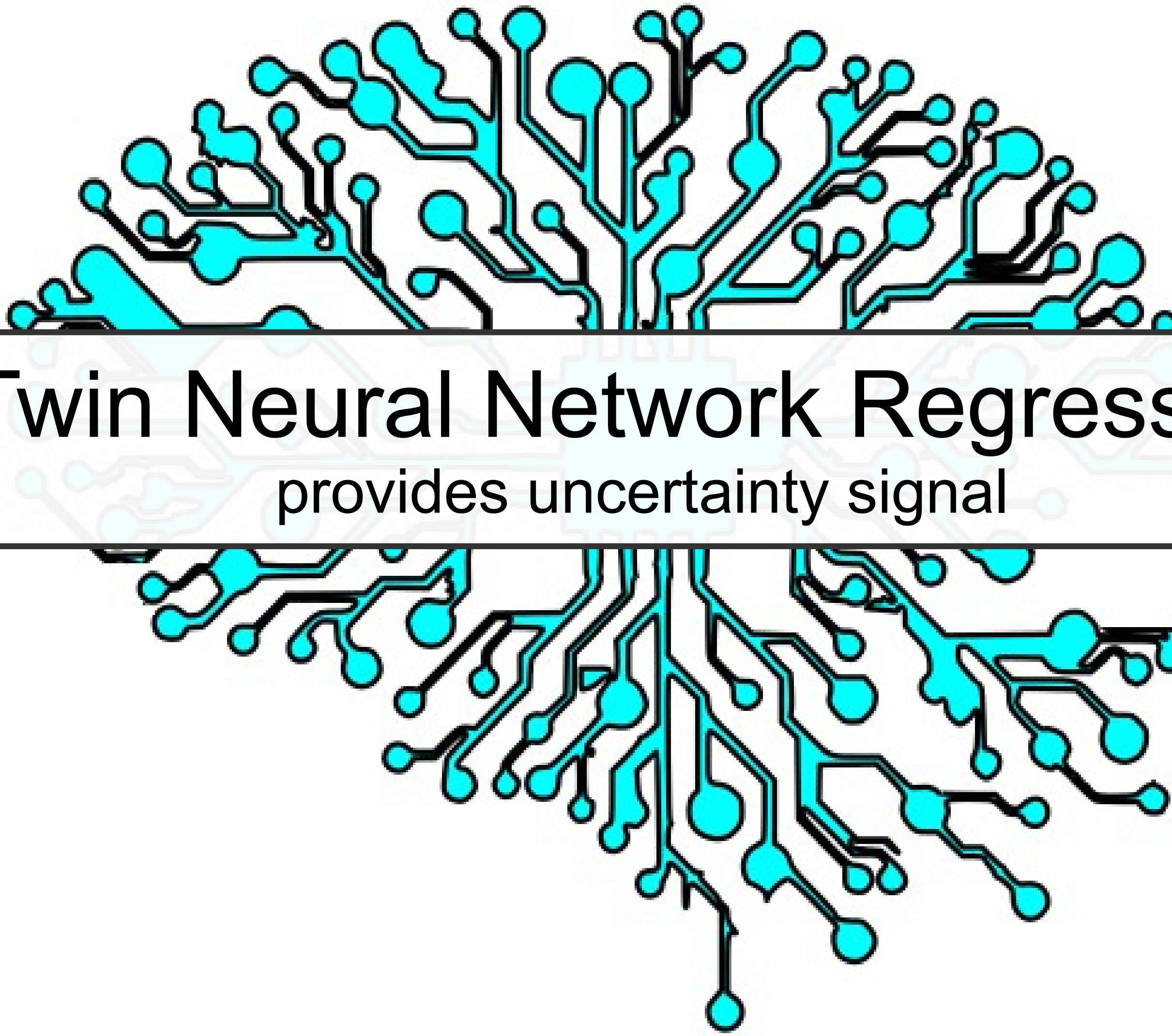
Common Data	BH	CS	EE	YH	WN	BC
RF	4.24 ± 0.29	8.23 ± 0.24	2.22 ± 0.08	2.95 ± 0.46	0.64 ± 0.02	**0.71 ± 0.03**
XGB	2.93 ± 0.18	4.37 ± 0.19	1.17 ± 0.04	**0.42 ± 0.06**	**0.61 ± 0.01**	**0.70 ± 0.03**
ANN	3.09 ± 0.14	5.37 ± 0.17	0.98 ± 0.03	0.52 ± 0.07	0.64 ± 0.01	0.76 ± 0.02
ANNE	3.43 ± 0.32	5.14 ± 0.21	0.89 ± 0.04	**0.43 ± 0.05**	**0.62 ± 0.01**	**0.72 ± 0.03**
MCD	2.95 ± 0.15	6.07 ± 0.21	2.96 ± 0.12	1.42 ± 0.18	0.68 ± 0.01	**0.72 ± 0.03**
TNN	**2.55 ± 0.10**	4.19 ± 0.25	0.52 ± 0.02	0.49 ± 0.07	**0.62 ± 0.01**	0.83 ± 0.03
TNNE	**2.61 ± 0.20**	**3.88 ± 0.22**	**0.46 ± 0.02**	**0.37 ± 0.06**	0.63 ± 0.01	**0.72 ± 0.02**

Remember these

Science Data	RP	RCL	WSB
RF	0.604 ± 0.013	0.288 ± 0.004	0.141 ± 0.011
XGB	0.229 ± 0.005	0.124 ± 0.002	0.071 ± 0.006
ANN	0.050 ± 0.002	0.019 ± 0.000	0.047 ± 0.004
ANNE	0.032 ± 0.002	0.016 ± 0.001	0.031 ± 0.002
MCD	0.086 ± 0.002	0.033 ± 0.001	0.042 ± 0.003
TNN	0.022 ± 0.001	0.017 ± 0.000	**0.020 ± 0.001**
TNNE	**0.016 ± 0.001**	**0.014 ± 0.001**	0.022 ± 0.002

Image Data	ISING
RF	0.601 ± 0.003
XGB	0.144 ± 0.003
CNN	0.050 ± 0.001
CNNE	0.044 ± 0.001
CMCD	0.052 ± 0.001
CTNN	0.035 ± 0.001
CTNNE	**0.030 ± 0.001**

Table 1. Best estimates for root mean square errors (RMSEs) of different algoritms on the test sets belonging to different data sets. The Lowest RMSEs are in bold for clarity. Our confidence on the RMSEs is determined by their standard error. Data sets: Boston housing (BH), concrete strength (CS), energy efficiency (EE), yacht hydrodynamics (YH), red wine quality (WN), Bio Conservation (BC), random polynomial (RP), RCL circuit (RCL), Wheatstone bridge (WSB) and the Ising Model (ISING). Algorithms: Random Forests (RF), xgboost (XGB), Neural Networks (ANN), Monte-Carlo Dropout networks (MCD), Twin Neural Networks (TNN) and ensembles (E) or convolutional variants (C).



Twin Neural Network Regression

provides uncertainty signal

Uncertainty Signal

Reliability: Knowing when the predictions can be trusted.

Even an inaccurate model can be reliably wrong!

Even an accurate model can make mistakes when it is applied to data points that are too different from the training data.

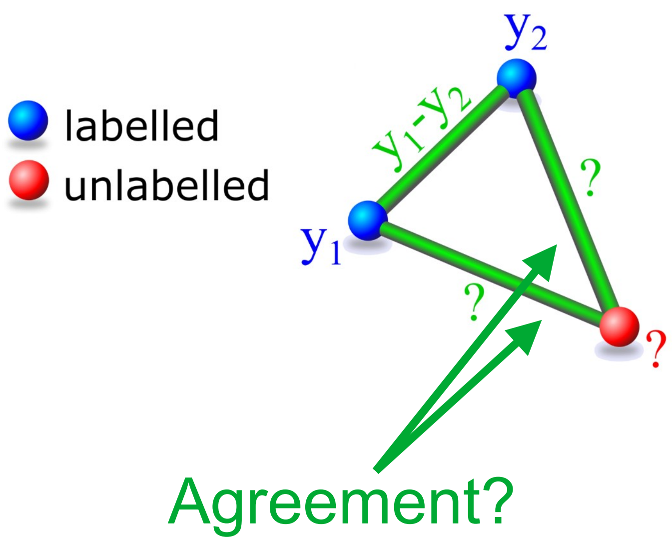
- Adverserial attacks
- Interpolation
- Extrapolation

Uncertainty Signal

Do ensemble members agree?

$$y_i^{pred} = \frac{1}{m}\sum_{j=1}^{m} F(x_i, x_j^{train}) + y_j^{train} = \frac{1}{m}\sum_{j=1}^{m} \frac{1}{2}F(x_i, x_j^{train}) - \frac{1}{2}F(x_j^{train}, x_i) + y_j^{train}$$

- Uncorrelated predictions make different mistakes
- Measure ensemble standard deviation

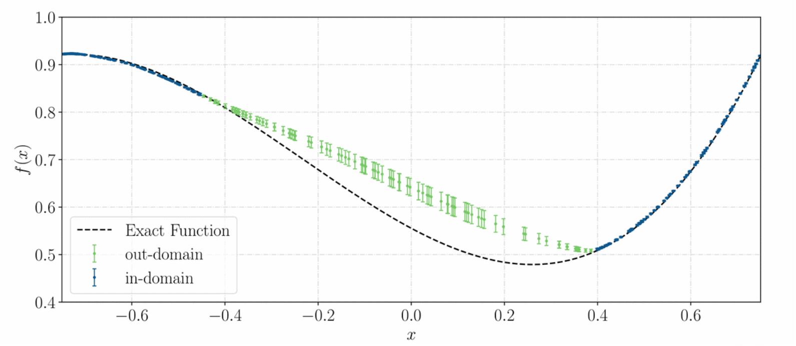


(additional uncertainty signal based on loop consistencies)

Uncertainty Signal

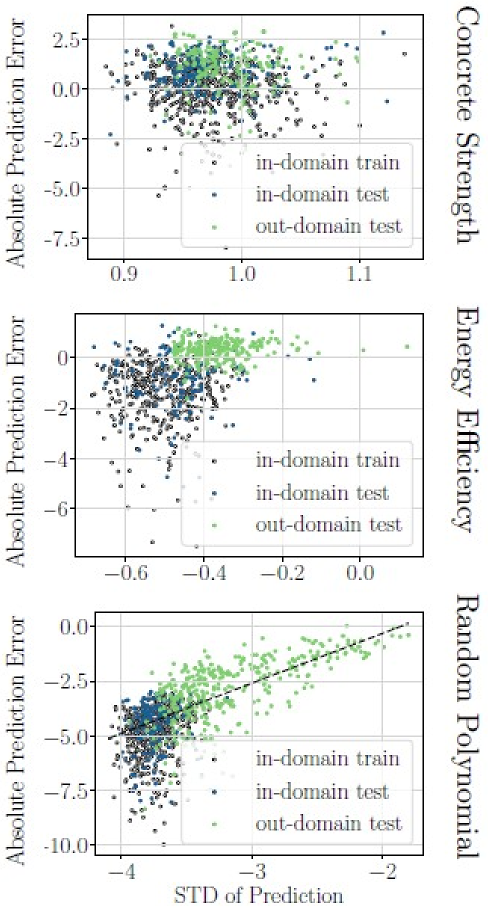
Example 1d Function

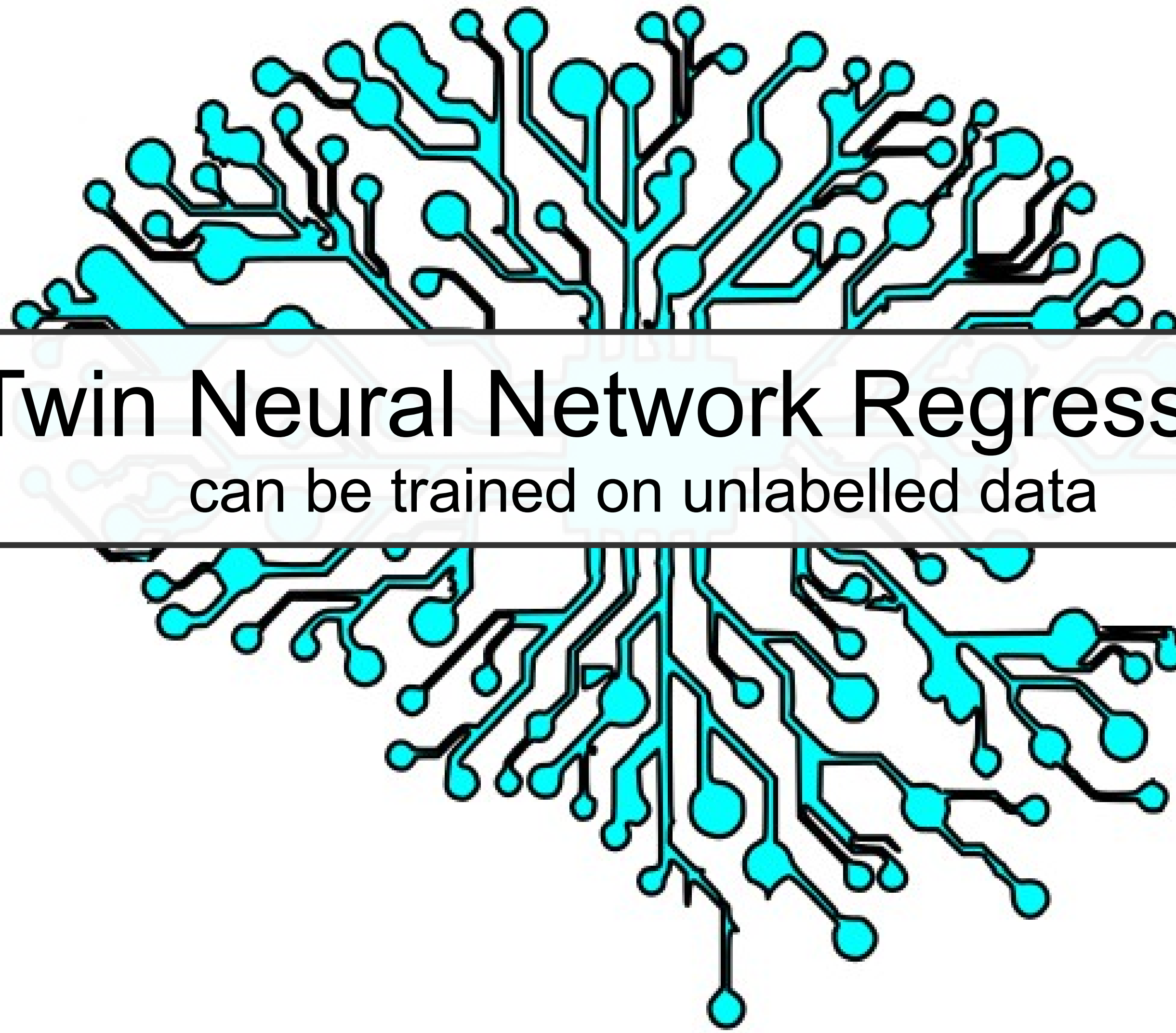
- Uncertainty increases in interpolation regime



Generic Data Sets

- High StD suggests higher prediction error
- In domain test data has lower StD and Error
- Out of domain test data has higher StD and Error





Twin Neural Network Regression

can be trained on unlabelled data

Semi-Supervised Learning

Supervised Learning
Learning from Labelled Data

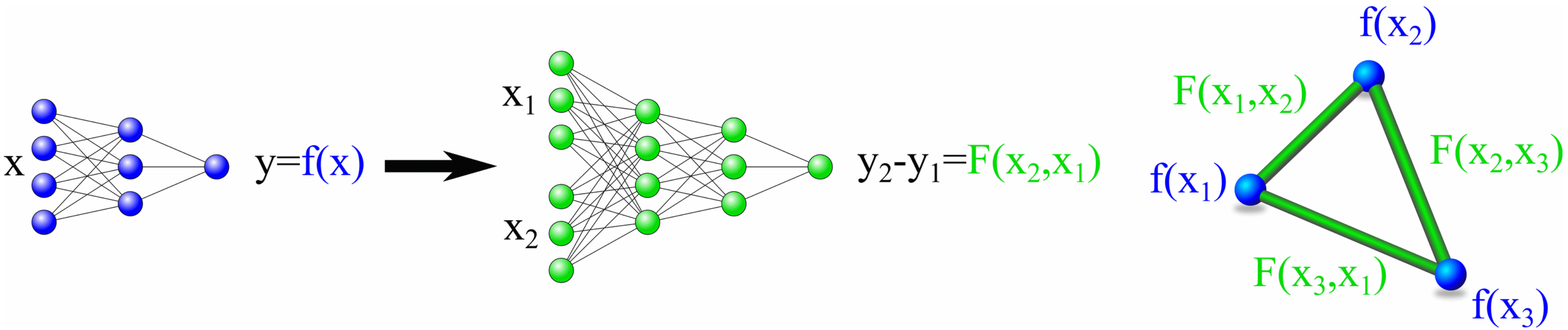
Unsupervised Learning
Learning from Unlabelled Data

Semi-Supervised Learning
Learning from Labelled and Unlabelled Data

- Transductive: The goal of transductive learning is to infer the correct labels for given unlabelled data which is present during the training phase
- Inductive: The goal of inductive learning is to infer the correct mapping from that allows labelling of unlabelled data not present during training

(Semi Supervised Regression is neglected vs Classification)

Semi-Supervised Learning

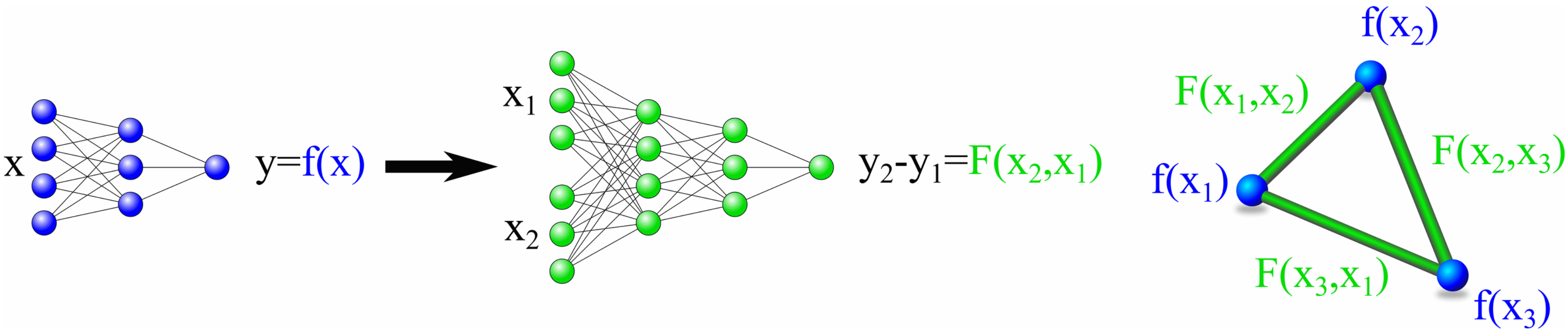


- Train to enforce loop consistency during training
- Loops can be used as training data even if the data points within them are unlabelled

$$0 = F(x_i, x_j) + F(x_j, x_k) + F(x_k, x_i)$$

- It can be viewed as two predictions provide a suggested label for the third.

Semi-Supervised Learning
Loss Function



MSE loss for training on labelled training loops

$$loss_{MSE} = \frac{1}{n^2} \sum_{ij} \left(F(x_i, x_j) - (y_i - y_j)\right)^2$$

Loop loss for training on unlabelled/partially labelled loops

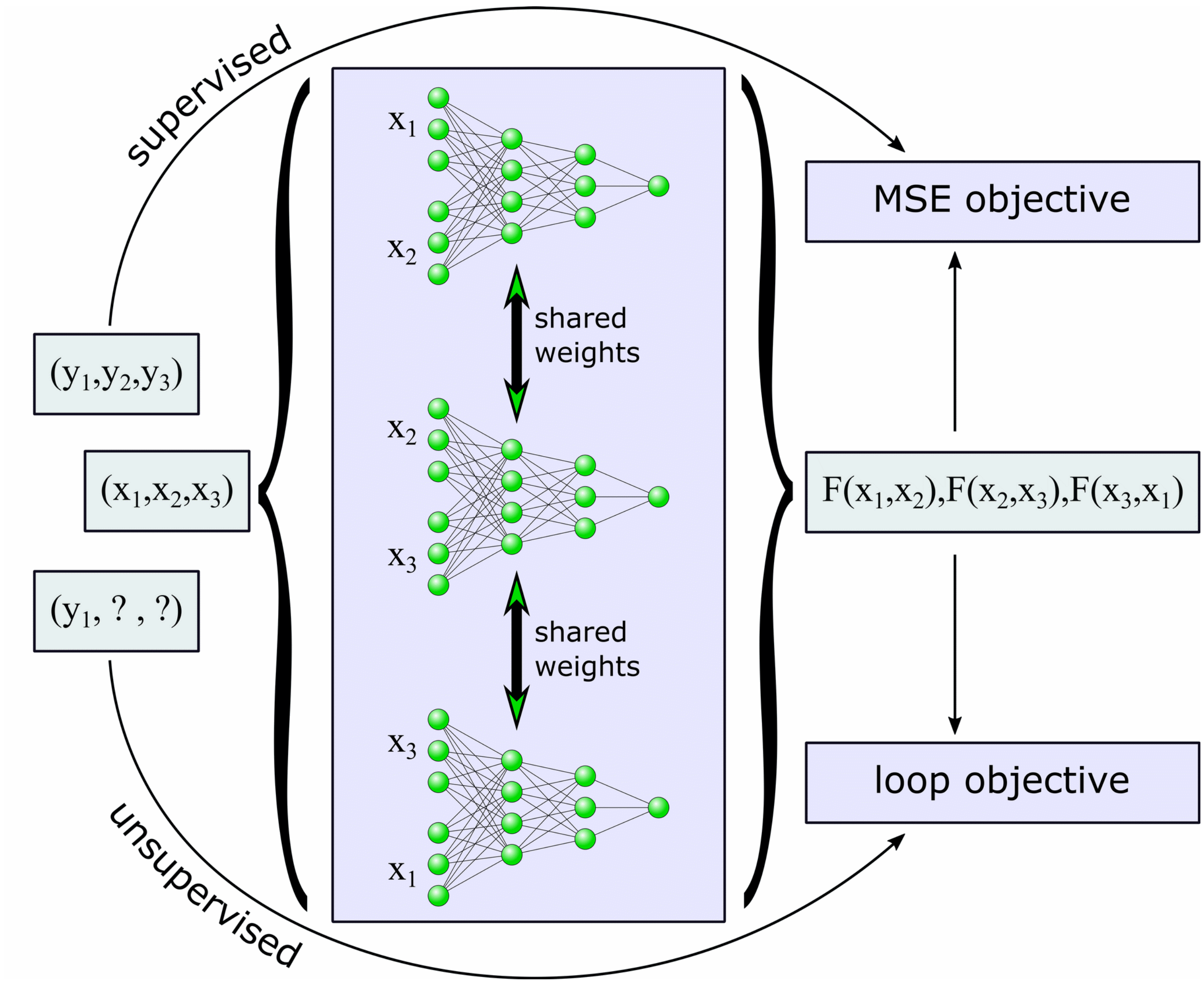
$$loss_{loop} = \frac{1}{(m+n)^3} \sum_{ijk} \left(F(x_i, x_j) + F(x_j, x_k) + F(x_k, x_i)\right)^2$$

Combine loss function with loop weight hyperparameter

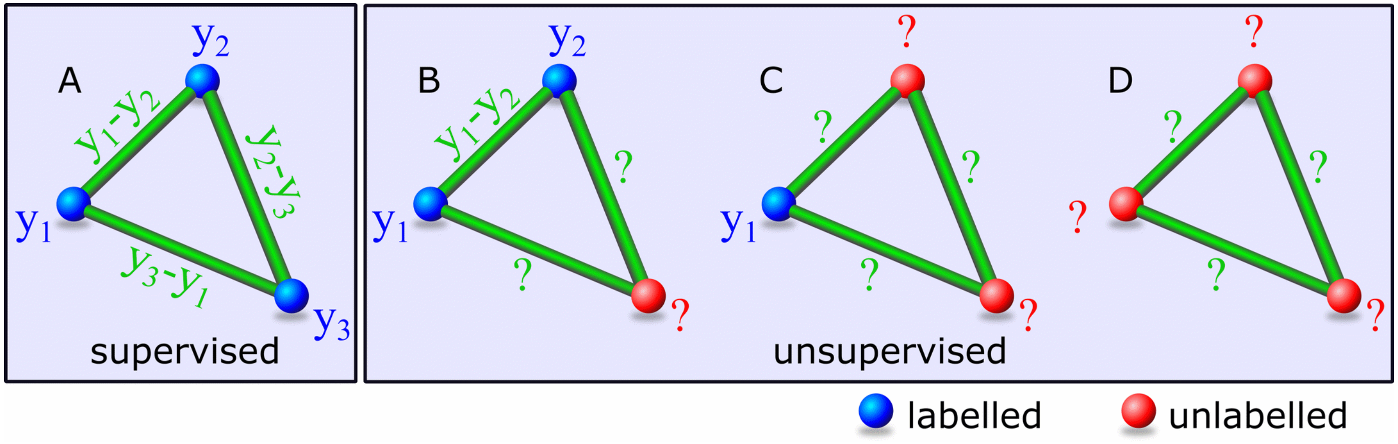
$$loss = loss_{MSE} + \Lambda\, loss_{loop}$$

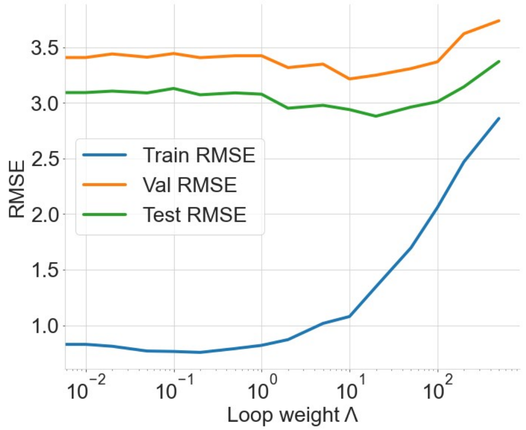
Semi-Supervised Learning

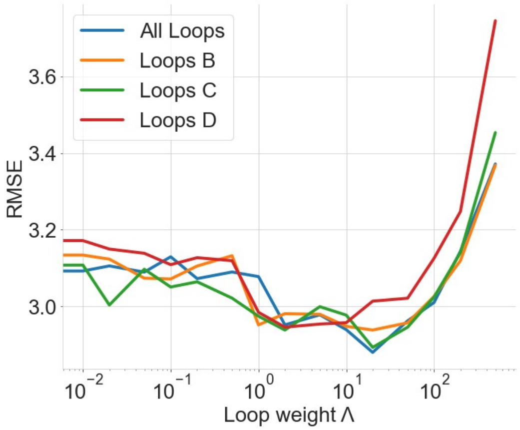
Training Architecture



Semi-Supervised Learning
Loop Types







Compare Loop types on Boston Housing Data (transductive)

- All Loops together seem best

Semi-Supervised Learning
Results

Table 2: Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), red wine quality (WN), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 90% of the available data where 30% is labelled training data, 10% is unlabelled validation data and 50% is unlabelled test data whose labels are predicted using TNNR as a transductive semi-supervised learning method. The labels of the 10% of the data which was not used during training are inferred using TNNR as an inductive semi-supervised learning method.

	30% labelled training data					
	Supervised	Transductive	Gain	Supervised	Inductive	Gain
BC	0.9382 ± 0.0137	0.7960 ± 0.0056	15.2%	0.8996 ± 0.0280	0.7721 ± 0.0175	14.2%
BH	4.1357 ± 0.1229	3.8228 ± 0.0951	7.6%	3.6830 ± 0.2337	3.5521 ± 0.2281	3.6%
CS	6.0777 ± 0.0773	5.9088 ± 0.0616	2.8%	6.0467 ± 0.1412	6.0905 ± 0.1260	−1.0%
EE	1.5084 ± 0.0317	1.4194 ± 0.0409	5.9%	1.4794 ± 0.0416	1.3902 ± 0.0459	6.0%
RCL	0.0200 ± 0.0003	0.0194 ± 0.0003	3.0%	0.0203 ± 0.0004	0.0195 ± 0.0004	3.9%
TF	0.0066 ± 0.0004	0.0063 ± 0.0004	4.5%	0.0064 ± 0.0004	0.0059 ± 0.0004	7.8%
WN	0.7841 ± 0.0047	0.6511 ± 0.0027	17.0%	0.7868 ± 0.0087	0.6534 ± 0.0075	17.0%
WSB	0.0341 ± 0.0012	0.0341 ± 0.0012	0.0%	0.0368 ± 0.0018	0.0368 ± 0.0018	0.0%
YH	1.2203 ± 0.0616	1.2203 ± 0.0616	0.0%	1.1170 ± 0.0910	1.1170 ± 0.0910	0.0%

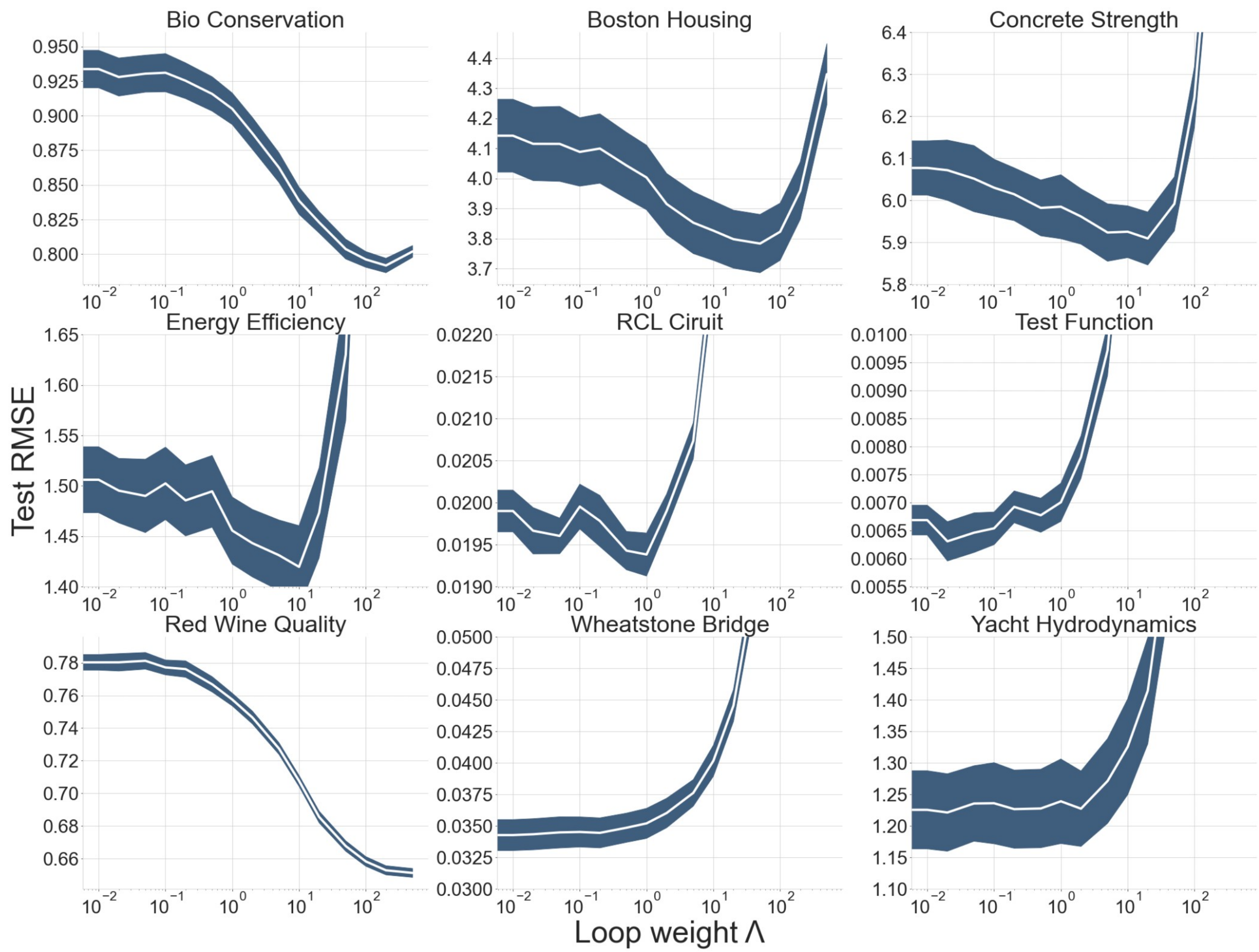
Semi-Supervised Learning
Results

Table 2: Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), red wine quality (WN), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 90% of the available data where 30% is labelled training data, 10% is unlabelled validation data and 50% is unlabelled test data whose labels are predicted using TNNR as a transductive semi-supervised learning method. The labels of the 10% of the data which was not used during training are inferred using TNNR as an inductive semi-supervised learning method.

	30% labelled training data					
	Supervised	Transductive	Gain	Supervised	Inductive	Gain
BC	0.9382±0.0137	0.7960±0.0056	$\boxed{15.2\%}$	0.8996±0.0280	0.7721±0.0175	$\boxed{14.2\%}$
BH	4.1357±0.1229	3.8228±0.0951	7.6%	3.6830±0.2337	3.5521±0.2281	3.6%
CS	6.0777±0.0773	5.9088±0.0616	2.8%	6.0467±0.1412	6.0905±0.1260	−1.0%
EE	1.5084±0.0317	1.4194±0.0409	5.9%	1.4794±0.0416	1.3902±0.0459	6.0%
RCL	0.0200±0.0003	0.0194±0.0003	3.0%	0.0203±0.0004	0.0195±0.0004	3.9%
TF	0.0066±0.0004	0.0063±0.0004	4.5%	0.0064±0.0004	0.0059±0.0004	7.8%
WN	0.7841±0.0047	0.6511±0.0027	$\boxed{17.0\%}$	0.7868±0.0087	0.6534±0.0075	$\boxed{17.0\%}$
WSB	0.0341±0.0012	0.0341±0.0012	0.0%	0.0368±0.0018	0.0368±0.0018	0.0%
YH	1.2203±0.0616	1.2203±0.0616	0.0%	1.1170±0.0910	1.1170±0.0910	0.0%

Remember these?

Semi Supervised Learning
(30% labelled, transductive)



Summary

Twin Neural Network Regression is an Accurate and Reliable State of the Art Regression Algorithm

- ✗ Circumvents Bias Variance Tradeoff
- ✗ Provides Uncertainty Signal
- ✗ Can be Trained on Unlabelled Data for Transductive and Inductive Semi-Supervised Learning
- ✗ Only One Single Neural Network + One Hyperparameter