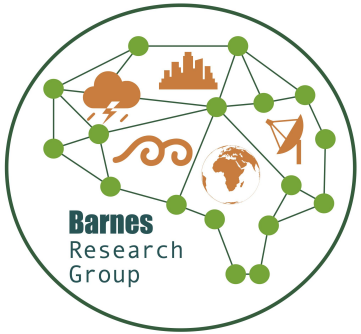
Applied Machine Learning for Earth Scientists



Elizabeth Barnes, Marybeth Arcodia, Charlotte Connolly, Frances Davenport, Zaibeth Carlo Frontera, Emily Gordon, Daniel Hueholt, Antonios Mamalakis, Elina Valkonen

Please cite these slides with the doi provided by Zenodo. Linked here.

Topics Covered

Basics of Machine Learning Applications to Earth Science

- Machine Learning for Science
- Foundational Concepts of ML
- Decision Trees and Random Forests
- Artificial Neural Networks (ANNs)
- ANN Coding Examples
- Advanced ANN Techniques

Visualization and Explainability of Machine Learning in Earth Science

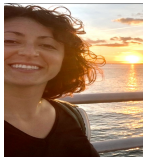
- Ethical Use of AI in Earth Science
- Implementation and Assessment of an ANN
- Methods of Explainable AI (XAI) for ANNs
- Simple Uncertainty Quantification

Resources

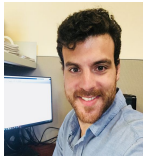
- Github: https://github.com/eabarnes1010/ml_tutorial_csu
  - Open-access code for ML examples included here and some other common ML applications
- Additional ML Resources
  - Google doc with online tutorials, videos, books, papers, etc.

Organizers

Elizabeth Barnes: Professor
- Research: climate and data science
- Website: https://barnes.atmos.colostate.edu
- Email: eabarnes@colostate.edu



Marybeth Arcodia: Postdoc
- Research: Understanding sources of S2S predictability using artificial neural networks
- Email: marcodia@rams.colostate.edu



Antonios Mamalakis: Postdoc
- Research: Dr Mamalakis' research focuses on the application of machine learning (ML) and explainable AI (XAI) methods to climate problems and on climate predictability and teleconnections.
- Website: https://amamalak.wixsite.com/antonios
- Email: amamalak@colostate.edu

Elina Valkonen: Postdoc
- Research: Utilizing "segmentation framework" to detect weather patterns in global climate models to help improve model evaluations. Specific focus on Arctic regions.
- Email: elina.valkonen@colostate.edu

Frances Davenport: Postdoc
- Research: using "transfer learning" to train neural networks with both climate model data and observations to make better real-world S2S predictions
- Website: https://fdavenport.github.io
- Email: f.davenport@colostate.edu

Barnes Research Group

COLORADO STATE UNIVERSITY

Organizers

Emily Gordon: PhD Student
- Research: Decadal variability and prediction with and without machine learning
- Website: https://sites.google.com/view/emilygordon
- Email: emgordy@colostate.edu

Charlie Connolly: MS Student
- Research: Internal variability and climate change
  Website: https://sites.google.com/view/connolly-climate/home
- Email: cconn@rams.colostate.edu

Daniel Hueholt: MS Student
- Research: I study potential methods to intervene in the Earth system in order to reduce risks and impacts from climate change.
- Website: hueholt.earth
- Email: dhueholt@rams.colostate.edu

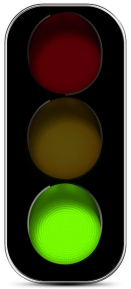
Zaibeth Carlo Frontera: MS Student
- Research: Hurricane number prediction with 2-3 forecast lead in the East Pacific using Random Forests.
- Email: zaibeth.carlo-frontera@colostate.edu

Barnes Research Group

COLORADO STATE UNIVERSITY

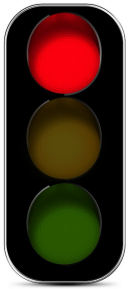
Materials also sourced from Wei-Ting Hsiao, Jamin Rader, Aaron Hill, Imme Ebert-Uphoff, Kirsten Mayer, Ben Toms, Yoonjin Lee @ CSU

What you will learn here...



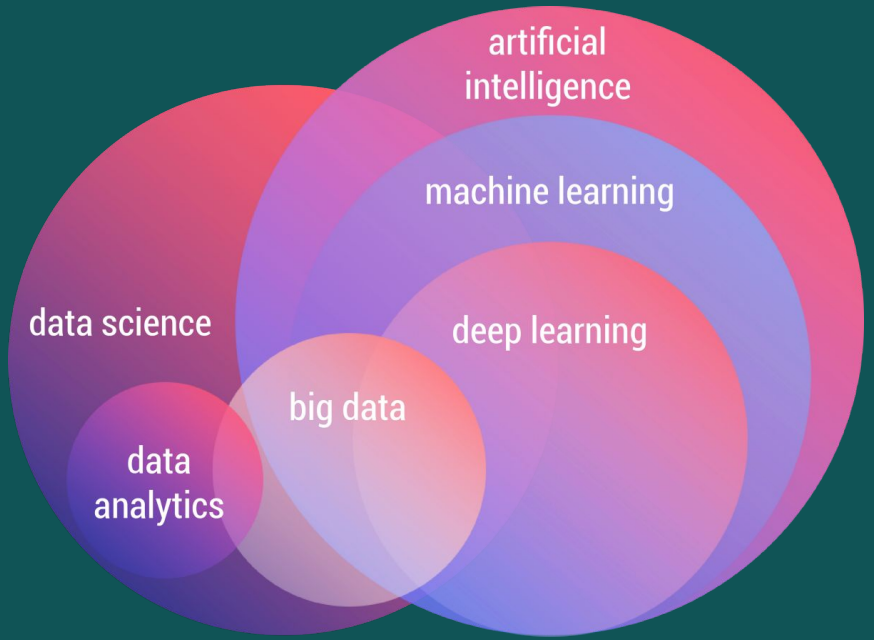
- Neural networks are not magic!
- Basic concepts and terminology
- Simple code to get started
- Examples of topics being explored in the field

What you will not learn…

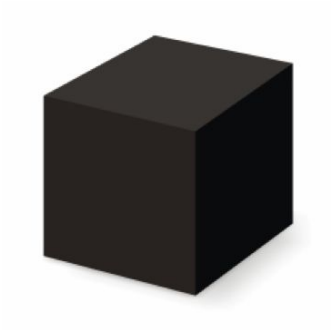


- A perfect formula for choosing the ML method right for your application
- How to choose/optimize specific parameters *a priori*
  (no one really knows anyway)
- Other unsupervised learning methods (e.g. clustering)
- All of the pitfalls associated with every choice

Machine Learning for Science



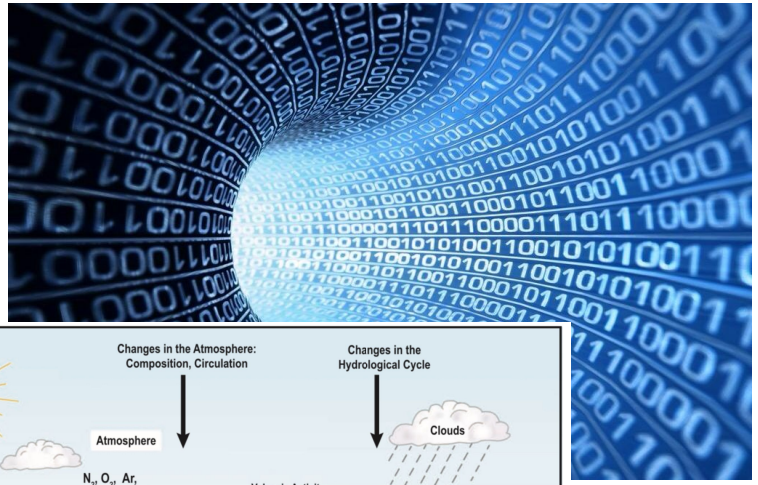
The “black box”

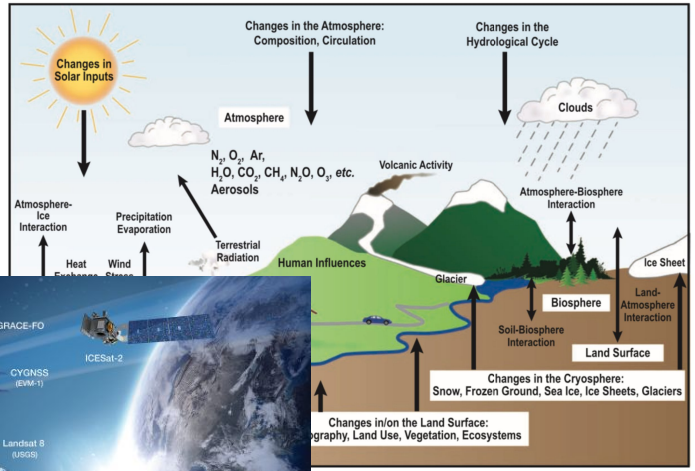
data →  → prediction

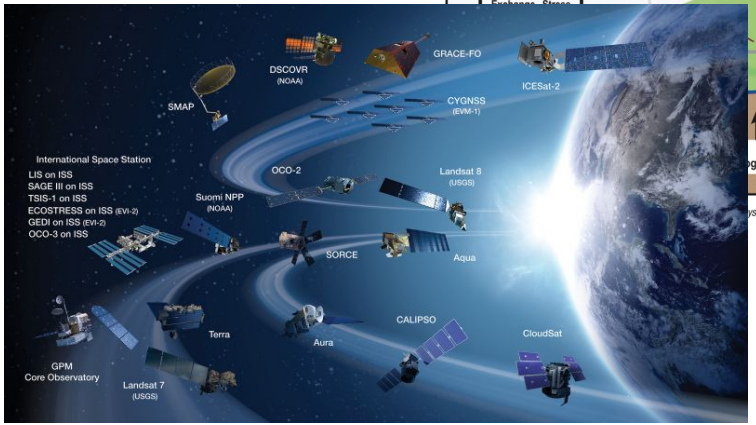
xkcd.com

We have a lot of data...

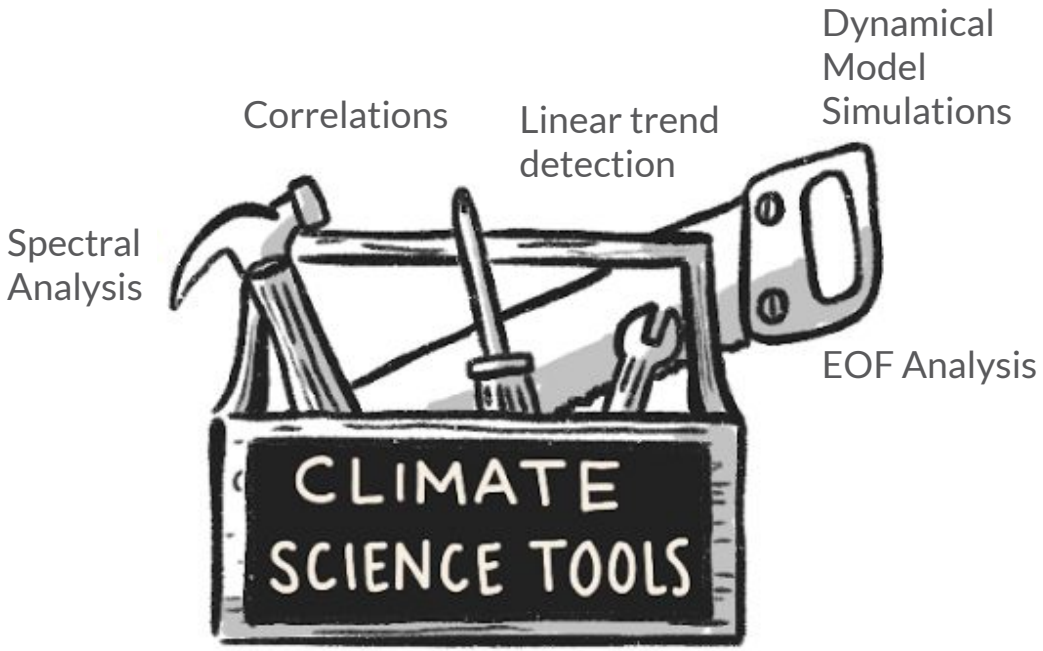
We are creating data faster than we can process.







Our field has a big toolbox…



Machine Learning offers an additional set of tools for the job.

One of our jobs as scientists is to sift through piles of data and try to extract useful relationships that apply elsewhere, i.e. that are applicable “out of sample”.

This is what many machine learning methods are designed to do.

Machine Learning*

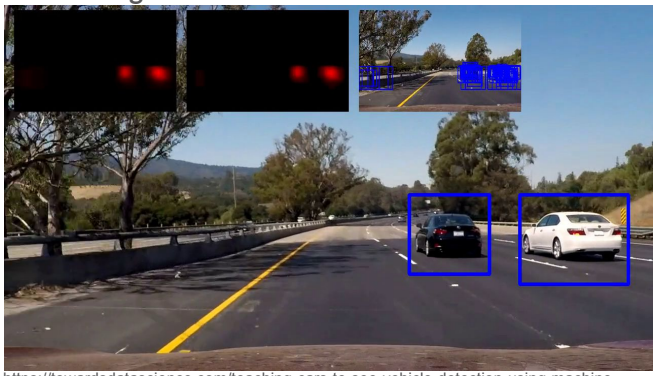
Commercial Applications

Machine learning has made huge inroads for commercial applications

For example, by the early 2000s convolutional neural networks processed 10-20% of all checks in the U.S.

The concept of machine learning has been around since the early 1950's. The explosion in the past decade can be attributed to advances in training deep networks and the explosion of available data

Self-driving vehicles

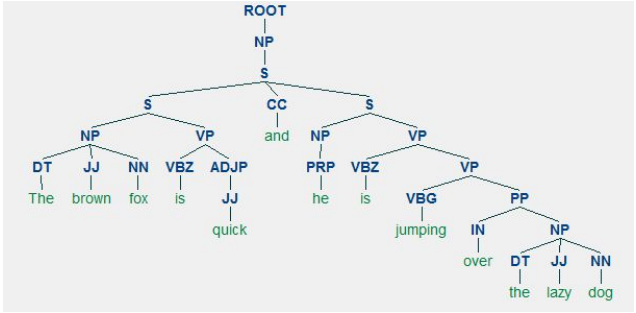


https://towardsdatascience.com/teaching-cars-to-see-vehicle-detection-using-machine-learning-and-computer-vision-54628888079a

Chihuahua or Blueberry Muffin?

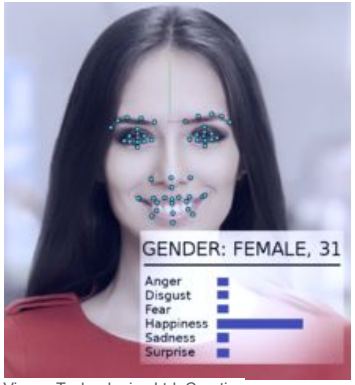
https://blog.cloudsight.ai/chihuahua-or-muffin-1bdf02ec1680

Natural language processing



https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72

Facial recognition



Visage Technologies Ltd, Creative Commons License, Wikimedia.org

Commercial Applications

Machine learning has made huge inroads for commercial applications

For example, by the early 2000s convolutional neural networks processed 10-20% of all checks in the U.S.

The concept of machine learning has been around since the early 1950's. The explosion in the past decade can be attributed to advances in training deep networks and the explosion of available data

Takes a winter image and turns it into summer

Commercial Applications

Machine learning has made huge inroads for commercial applications

For example, by the early 2000s convolutional neural networks processed 10-20% of all checks in the U.S.

The concept of machine learning has been around since the early 1950's. The explosion in the past decade can be attributed to advances in training deep networks and the explosion of available data

It can fail spectacularly too!

Adding carefully crafted noise to a picture can create a new image that people would see as identical, but which a DNN sees as utterly different.

Heaven, D., 2019: Why deep-learning AIs are so easy to fool. Nature, 574, 163–166.

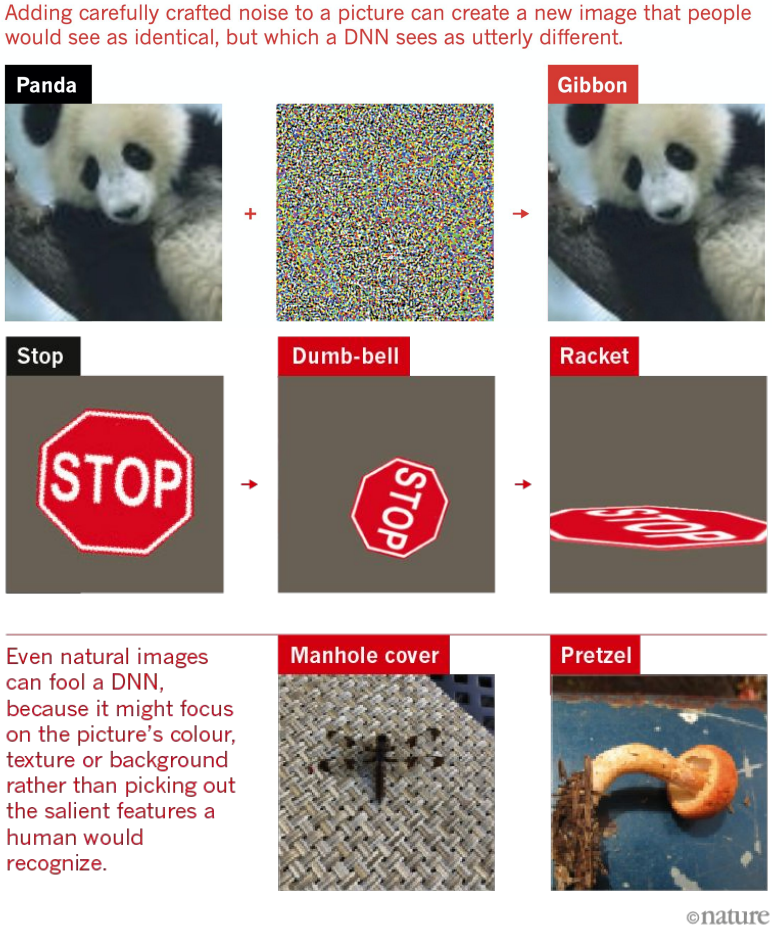
Commercial Applications

Machine learning has made huge inroads for commercial applications

For example, by the early 2000s convolutional neural networks processed 10-20% of all checks in the U.S.

The concept of machine learning has been around since the early 1950's. The explosion in the past decade can be attributed to advances in training deep networks and the explosion of available data

It can fail spectacularly too!



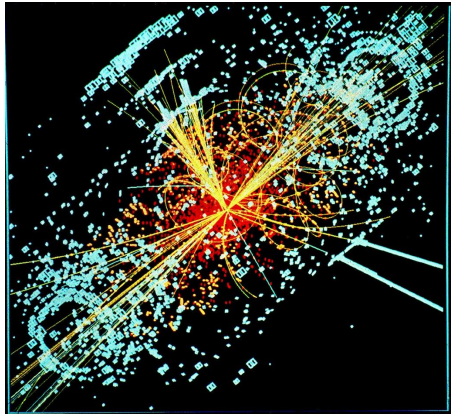
Heaven, D., 2019: Why deep-learning AIs are so easy to fool. Nature, 574, 163–166.

Science!

Even with its “black box” persona, ML has already aided significant scientific advances

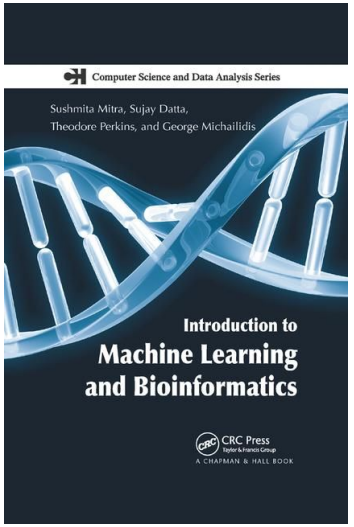
e.g., the area of bioinformatics has exploded in recent years due to machine learning and data mining

Distinguishing high-energy particles in the Large Hadron Collider

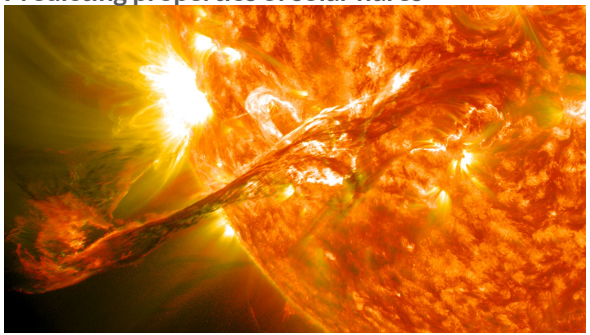


By Lucas Taylor / CERN - http://cdsweb.cern.ch/record/628469, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1433671

Gene prediction and sequencing



Predicting properties of solar flares



Identifying drug-drug interactions

Science!

Even with its “black box” persona, ML has already aided significant scientific advances

e.g., the area of bioinformatics has exploded in recent years due to machine learning and data mining

The number of science articles using supervised learning have seen large trends in recent year

However, AMS papers with supervised machine learning have lagged behind those across all of geoscience

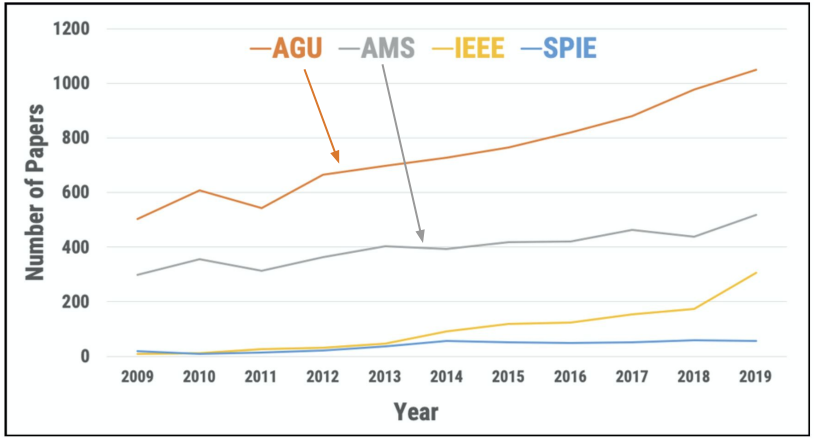


Fig. 1. The yearly number of Earth science papers from major publishers that discuss or use supervised machine learning. AMS = American Meteorological Society, IEEE = Institute of Electrical and Electronics Engineers, SPIE = International Society for Optics and Photonics. Credit: Katrina Virts and Rahul Ramachandran

Maskey, M., H. Alemohammad, K. J. Murphy, and R. Ramachandran (2020), Advancing AI for Earth science: A data systems perspective, Eos, 101, https://doi.org/10.1029/2020EO151245

ML for Climate Science

The field's interest, and research, has exploded in the past ~3 years!

Applications of ML for atmospheric science dates back as far as the 1960's!

Nonlinear Principal Component Analysis by Neural Networks: Theory and Application to the Lorenz System

ADAM H. MONAHAN

Oceanography Unit, Department of Earth and Ocean Sciences, and Crisis Points Group, Peter Wall Institute for Advanced Studies, University of British Columbia, Vancouver, British Columbia, Canada

1998

(Manuscript received 28 October 1998, in final form 7 May 1999)

ABSTRACT

A nonlinear generalization of principal component analysis (PCA), denoted nonlinear principal component analysis (NLPCA), is implemented in a variational framework using a five-layer autoassociative feed-forward neural network. The method is tested on a dataset sampled from the Lorenz attractor, and it is shown that the NLPCA approximations to the attractor in one and two dimensions, explaining 76% and 99.5% of the variance,

An Application of Adaptive Logic to Meteorological Prediction

H. R. GLAHN

U. S. Weather Bureau, Washington, D. C.

1964

(Manuscript received 12 May 1964, in revised form 25 July 1964)

ABSTRACT

Two adaptive logic models and a training algorithm for each are described. These models are tested on a meteorological prediction problem with the use of large developmental and test data samples. Discriminant analysis is used for comparison. It is found that ceiling heights at Washington National Airport could be forecast better with the discriminant analysis technique than with the adaptive logic models.

New Approach to Calculation of Atmospheric Model Physics: Accurate and Fast Neural Network Emulation of Longwave Radiation in a Climate Model

VLADIMIR M. KRASNOPOLSKY

NOAA/NCEP/SAIC, Camp Springs, and Earth System Science Interdisciplinary Center, University of Maryland, College Park, College Park, Maryland

MICHAEL S. FOX-RABINOVITZ AND DMITRY V. CHALIKOV

Earth System Science Interdisciplinary Center, University of Maryland, College Park, College Park, Maryland

2004

(Manuscript received 26 April 2004, in final form 25 October 2004)

ABSTRACT

A new approach based on a synergetic combination of statistical/machine learning and deterministic modeling within atmospheric models is presented. The approach uses neural networks as a statistical or machine learning technique for an accurate and fast emulation or statistical approximation of model physics parameterizations. It is applied to development of an accurate and fast approximation of an atmospheric longwave radiation parameterization for the NCAR Community Atmospheric Model, which is the most

1. Introduction

In recent years there has been an increase in the research effort concerning adaptive logic systems. These systems, including their associated hardware, are also known as learning machines or trainable networks. The desire is to create a system that can be trained to give the correct response when it is presented certain input data or stimuli in somewhat the same way a living organism might adjust to its environment or learn from past experience. It is also hoped that the system would be capable of certain generalizations so that when it is presented a data pattern that it has never encountered before it will still tend to give the correct response.

Adaptive logic models can be used for pattern recognition and, since the forcasting of certain weather elements involves, to some extent, the recognition of patterns, an application of adaptive systems to objective weather forecasting is thereby suggested.

2. Adaptive logic models

One adaptive model that has been described with minor variations by several authors, including Mattson,[1] Sebestyen (1962), Widrow (1962), Hu[2], Duda and Machanik[3] and Hu and Root (1964) is described below.

In Fig. 1 each "S" unit represents one of the $n+1$ numerical inputs to the system. Each "A" unit is an adjustable weight which is altered during the training procedure so that the process of summation, performed in the "$\sum$" unit, of A_iS_i, $i=0, 1, 2, \ldots, n$, and quantization into a binary output will provide the correct dichotomous classification. S_0 is held constant at $+1$ while S_i, $i=1, 2, \ldots, n$ may be different for different training sample points. This complete element has been called by Widrow (1962) an ADALINE for "adaptive linear neuron."

A training program for this model is indicated in Fig. 2. The value D is a positive constant which permits $\sum_{i=0}^{n} A_iS_i$ to have a value of D, rather than zero, immediately after an alteration of the A_i for the sample point which occasioned the alteration. After the output is made to agree with the desired output for a particular sample point, the A_i may again be changed for another sample point. This latter alteration may in turn cause the output to be incorrect for the former sample point. Repeated iterations over the data sample can be made and eventually the correct response may be given for all sample points if such distinction is possible with this model.

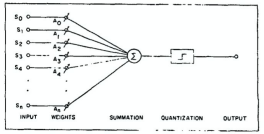


FIG. 1. A simple adaptive logic element.

[1] Mattson, R. L., 1959: The design and analysis of an adaptive system for statistical classification. S.M. Thesis, Massachusetts Institute of Technology, Cambridge, Mass., 60 pp.

[2] Hu, M. J., 1963: A trainable weather-forecasting system. Tech. Rep. No. 6759–1, Contract No. AF33(616)–7726, Systems Theory Laboratory, Stanford University, Stanford, Calif., 19 pp.

[3] Duda, R. O., and J. W. Machanik, 1963: An adaptive prediction technique and its application to weather forecasting. Paper presented at the Western Electronic Show and Convention, San Francisco, Calif.

ML for Climate Science

The field's interest, and research, has exploded in the past ~3 years!

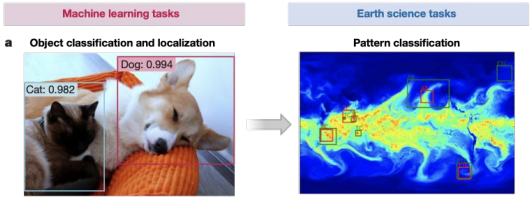
Applications of ML for atmospheric science dates back as far as the 1960's!

Range of applications:

- global weather prediction
- convective & radiative parameterizations
- downscaling
- extreme event detection
- data reconstruction
- weather prediction
- processing of remote sensing data

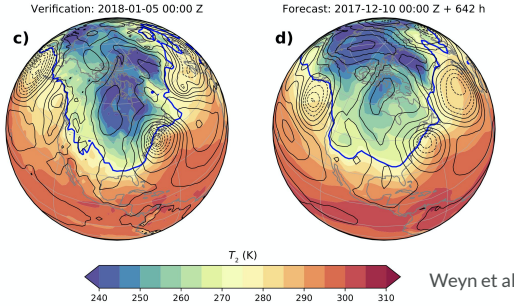
Extreme event detection

e.g. Reichstein et al. (2019)



Weather Prediction

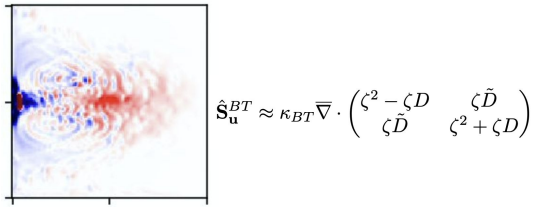
e.g. Gagne et al. (2019); Gagne et al. (2017); Chattopadhyay et al. (2019); Lagerquist et al. (2020)



Weyn et al. (2020)

Equation discovery

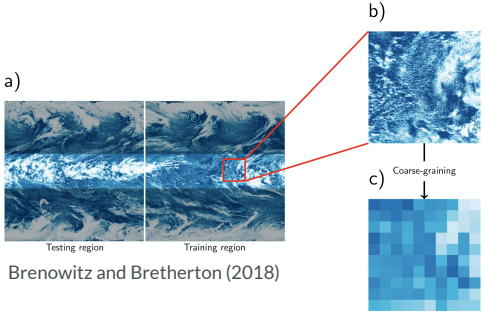
e.g. Zanna and Bolton (2020)



$$\hat{\mathbf{S}}_{\mathbf{u}}^{BT} \approx \kappa_{BT} \overline{\nabla} \cdot \begin{pmatrix} \zeta^2 - \zeta D & \zeta \tilde{D} \\ \zeta \tilde{D} & \zeta^2 + \zeta D \end{pmatrix}$$

Convective parameterizations

e.g. Rasp et al. (2018; PNAS); Schneider et al. (2017; GRL); O'Gorman and Dwyer (2018); Beucler et al. (2020; PRL)



Brenowitz and Bretherton (2018)

ML for Climate Science

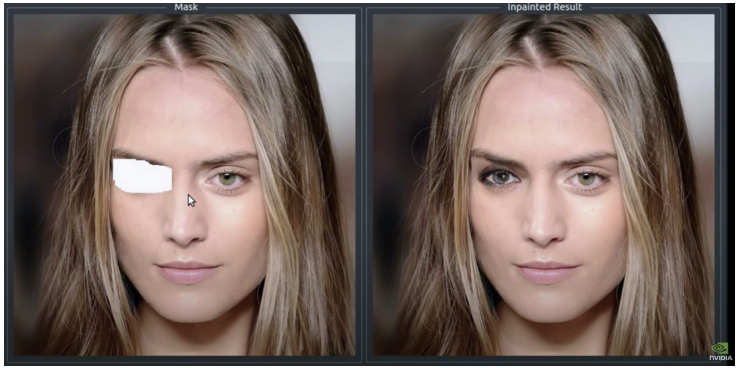
The field's interest, and research, has exploded in the past ~3 years!

Applications of ML for atmospheric science dates back as far as the 1960's!

Range of applications:

- global weather prediction
- convective & radiative parameterizations
- downscaling
- extreme event detection
- data reconstruction
- weather prediction
- processing of remote sensing data

Commercial application: Infilling an image



NVIDIA Research
https://www.youtube.com/watch?v=gg0F5JjKmhA

ML for Climate Science

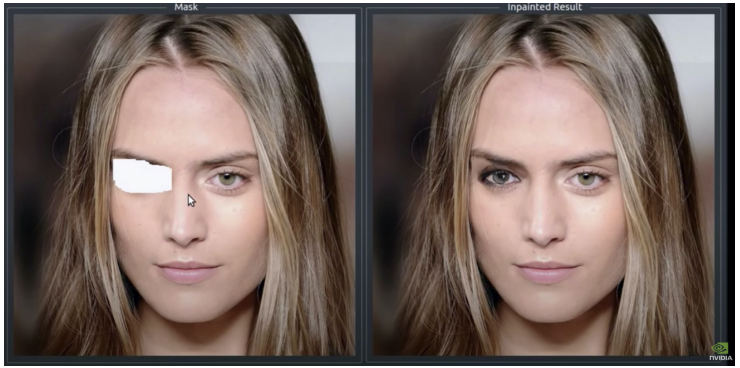
The field's interest, and research, has exploded in the past ~3 years!

Applications of ML for atmospheric science dates back as far as the 1960's!

Range of applications:

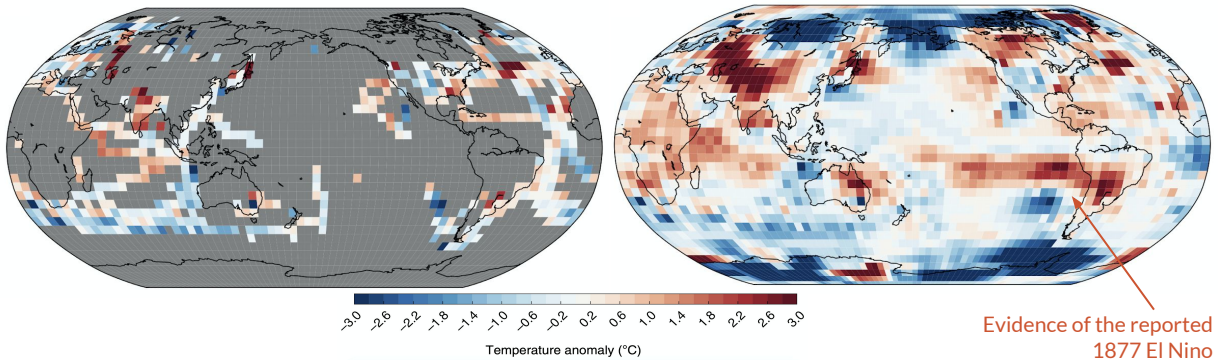
- global weather prediction
- convective & radiative parameterizations
- downscaling
- extreme event detection
- data reconstruction
- weather prediction
- processing of remote sensing data

Commercial application: Infilling an image



NVIDIA Research
https://www.youtube.com/watch?v=gg0F5JjKmhA

Climate Application: Temperature reconstruction (e.g. 1877)



Evidence of the reported 1877 El Nino

Kadow et al. (2020; Nature Geoscience)
Also see for other reconstruction applications DelSole and Nedza (2020)

ML for Climate Science

Communicating climate change is another area with great promise for ML

Groups are working on using deep learning to produce **accurate and vivid renderings of the future outcomes of climate change**

Reasons to use ML for Science

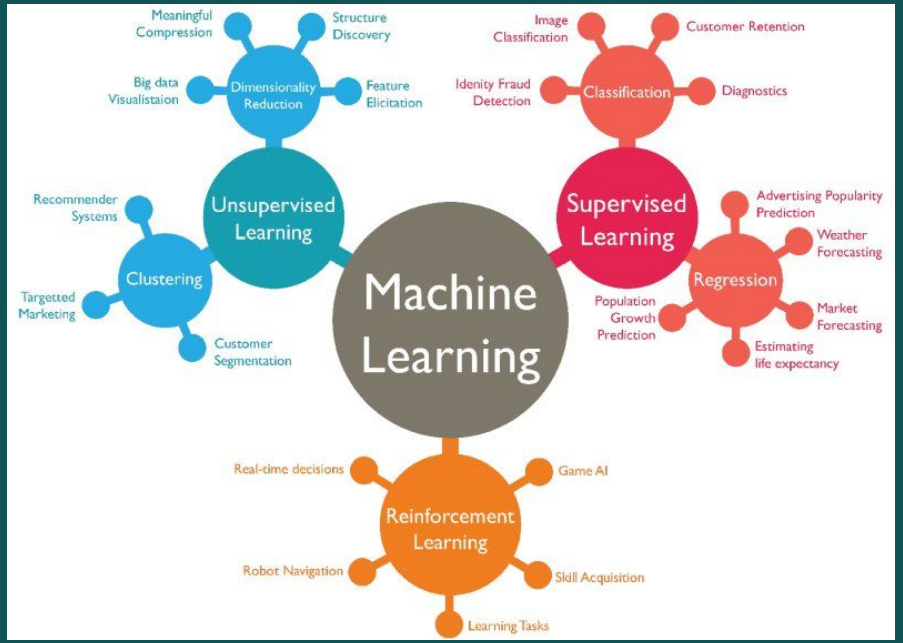
- Do it **better**
  - e.g. Convective parameterizations in models are not perfect, use ML to make them more accurate
- Do it **faster**
  - e.g. Radiation code in models is very slow (but we know the right answer) - use ML methods to speed things up
- Do **something new**
  - e.g. Go looking for non-linear relationships you didn't know were there

Very relevant for research: may be slower and worse, but can still learn something

Example uses in Climate Science

- Making climate models better
  - Parameterizations
  - Speed-up computations
- Better predictions
  - Post-processing dynamical (physics-based) forecasts
  - Purely data-driven forecasts
- Sources of predictability
  - process-based understanding
  - identifying dynamical model errors/biases
- Feature identification
  - Counting clouds, labeling cloud types in images
  - Counting/identifying extreme events
- Summarizing a lot of data
  - Dimension reduction
  - Cluster/group behaviour

Foundational Concepts of ML



What is Machine Learning?

Field of study that gives computers the ability to learn **without being explicitly programmed**.
Defined by Arthur Samuel (1959)

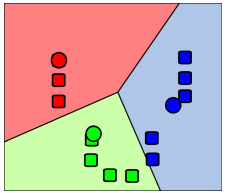
Practically speaking:

Techniques for fitting data.

It is not restricted by a single or a set of simple mathematical expressions.

It can be as complicated by putting in procedures, judgement statements, even randomness...

Types of machine learning

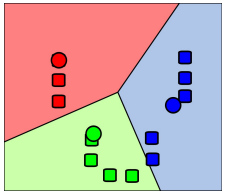


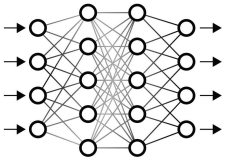
Unsupervised learning

looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision, e.g.

- principal component analysis
- clustering methods (k-means, self-organizing maps)
- autoencoders

There are other types, but these are the two main flavors we will discuss





Types of machine learning

Unsupervised learning

looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum of human supervision, e.g.

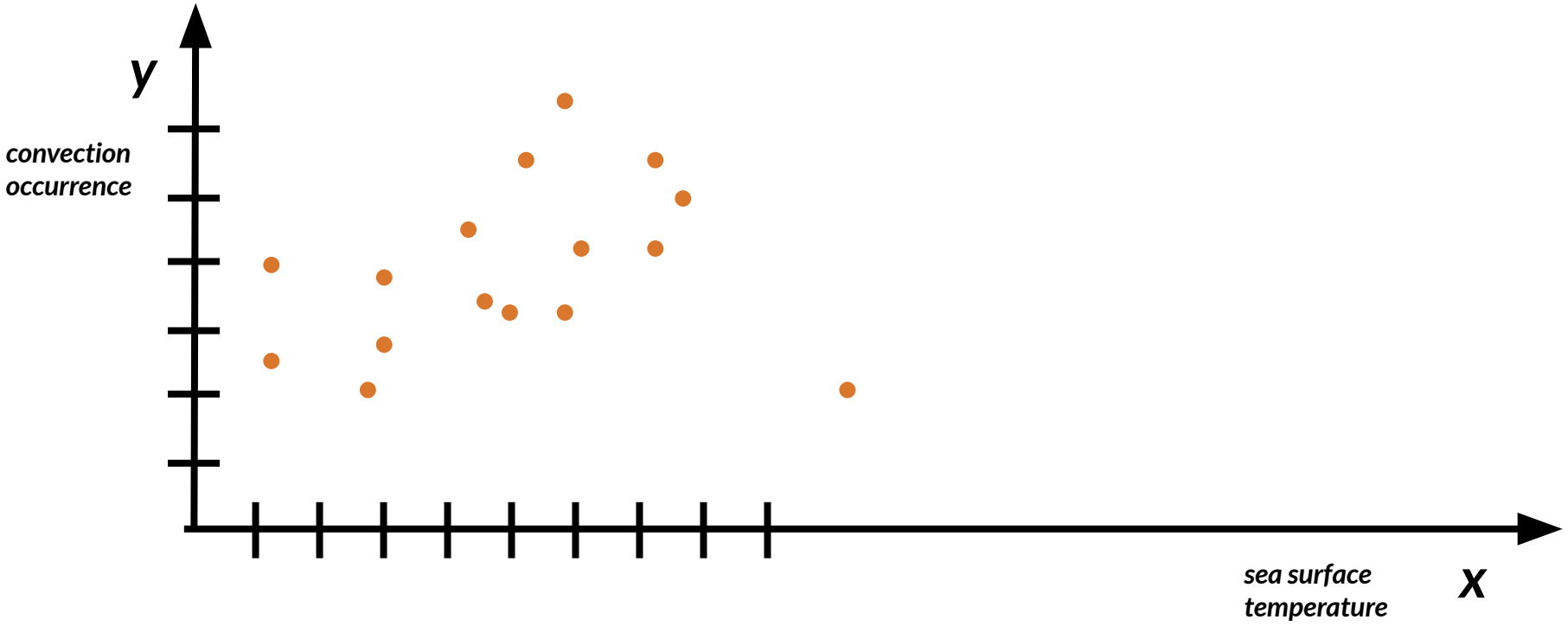
- principal component analysis
- clustering methods (k-means, self-organizing maps)
- autoencoders

Supervised learning

maps an input to an output based on example input-output pairs, e.g.

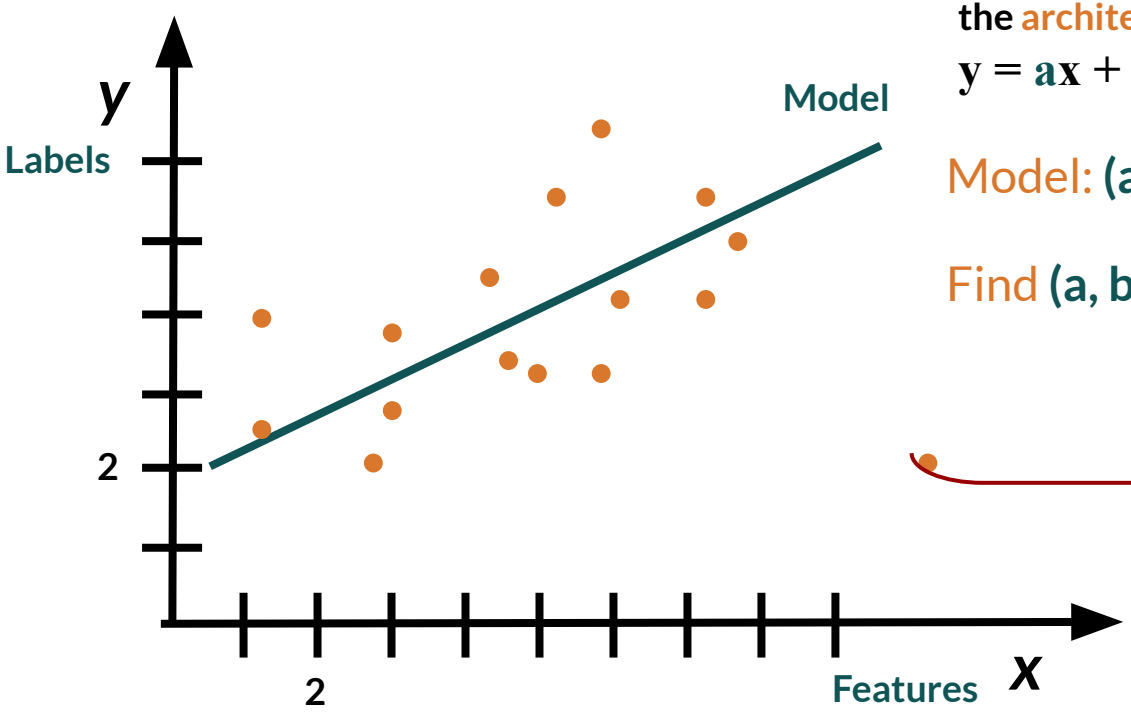
- regression
- decision trees and random forests
- support vector machines
- artificial neural networks

How do you predict y from x?



Linear regression model

With the assumption (or, architecture) that the model is a 1-degree polynomial, we describe this model as: (a, b) = (0.5, 2)



the architecture:

$$y = ax + b$$

Model: (a, b)

Root-mean-square error

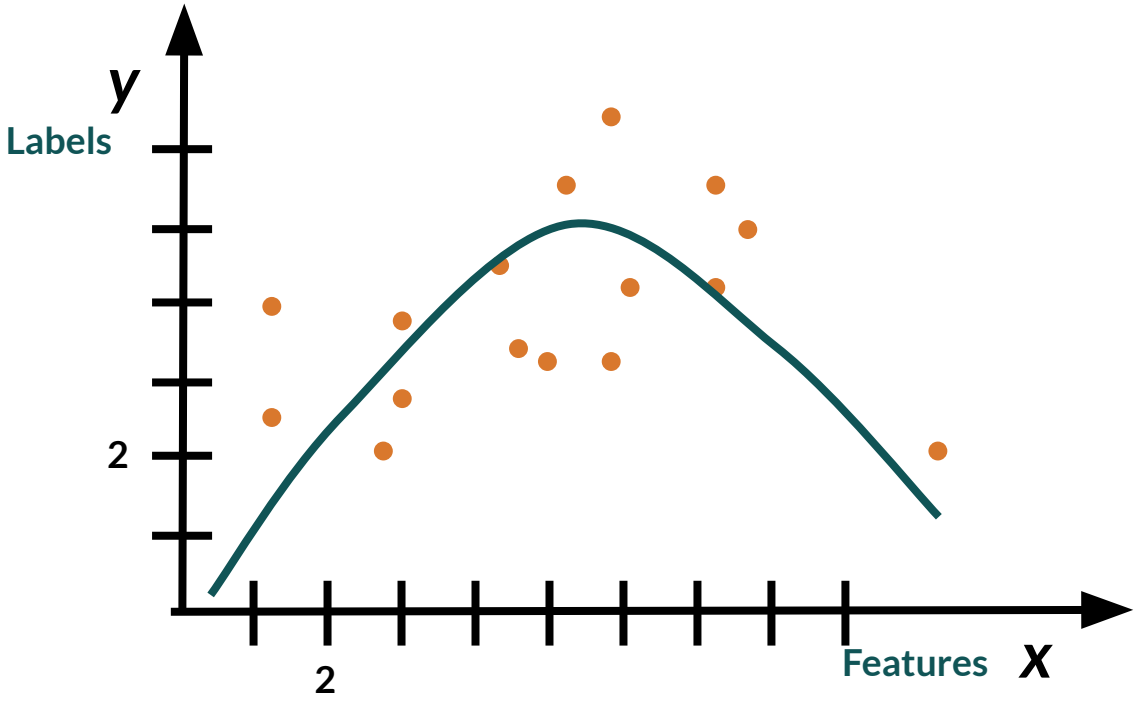
Find (a, b) that minimizes the RMSE.

Loss Function

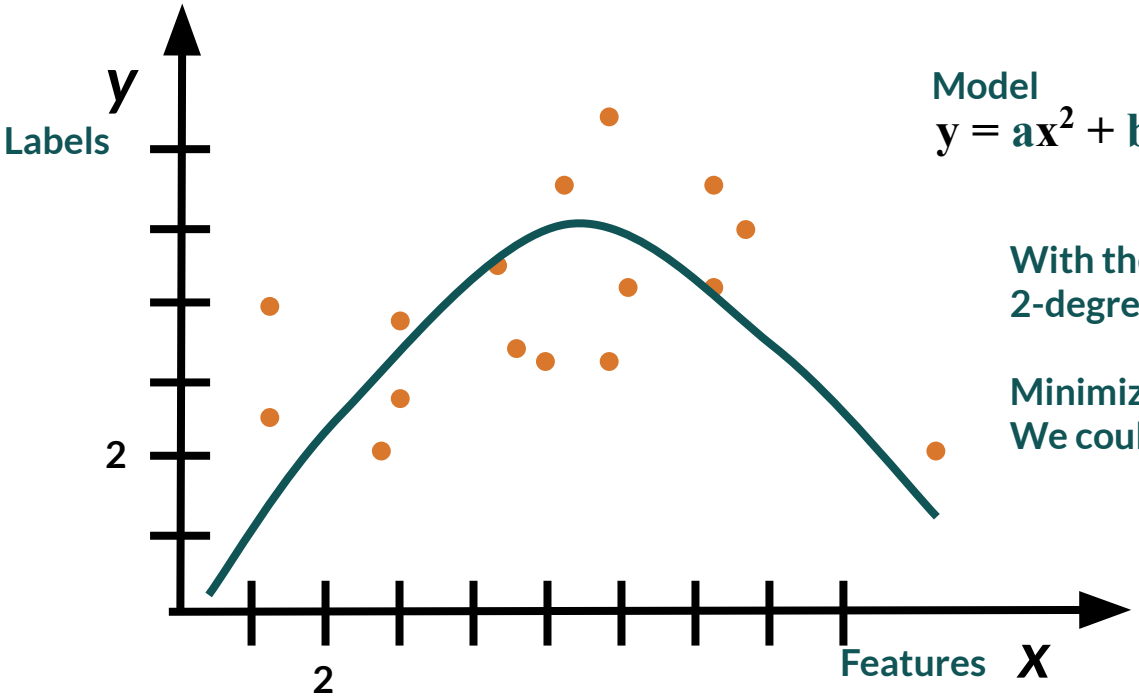
We always need a metric to define how "good" a model is.

Optimization

Nonlinear regression model



Nonlinear regression model



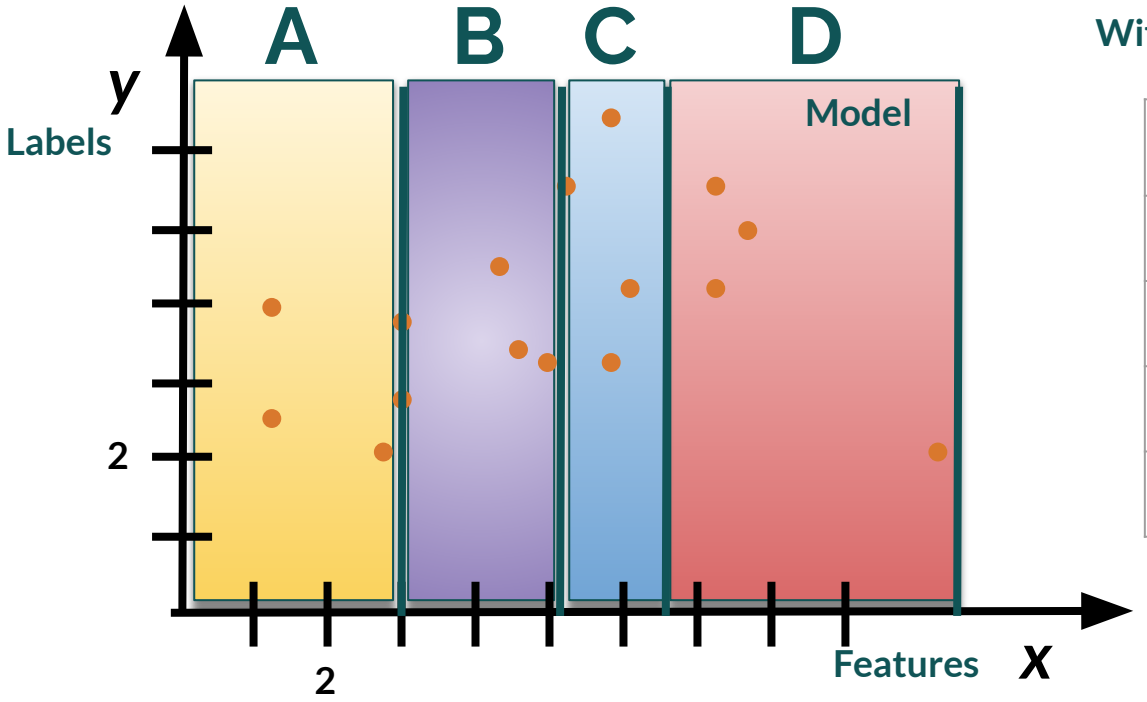
Model

$$y = ax^2 + bx + c$$

With the architecture that the model is a 2-degree polynomial:

Minimize the loss function: RMSE
We could find the model: (a, b, c) = (-0.2, 2, 0)

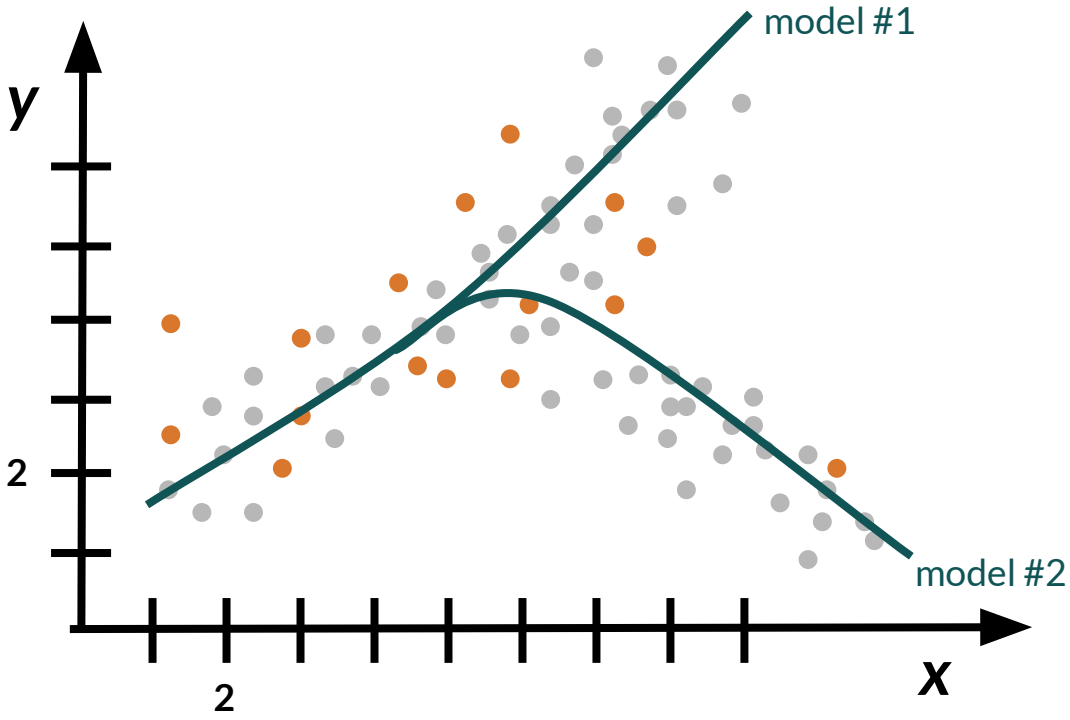
Classification-like model



With this architecture, the model is:

x	y
0.0 < x < 3.0	A
3.0 < x < 4.1	B
4.1 < x < 5.6	C
5.6 < x < 9.7	D

Selection of architectures



Which model is right?
Neither.

Which architecture should be used?
It depends.

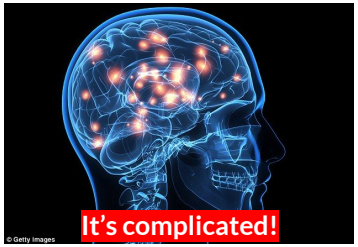
FACT:
Perfect Learning is impossible.
(Since not "all" the data is available.)

1) have a good quality of the data set
2) wisely choose architectures according to data properties
3) wisely use your data
increase the possibility for us to find good models (based on your purpose)!

Machine Learning Approach

Selecting Predictors

Question: Will Marybeth catch the bus to campus?

All possible factors → 

→ **Output:** Catch the bus?

Machine Learning Approach

Selecting Predictors

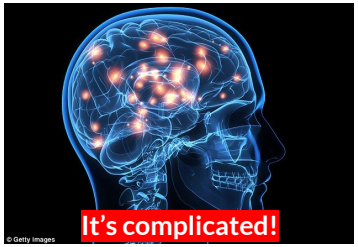
Question: Will Marybeth catch the bus to campus?

Inputs/Predictors:

- Time she woke up
- Trash day
- Weather conditions
- Current record for the Yankees
- The color of the sunrise
- ...

Truth/Predictand:

- Yes or No

All possible factors → 

→ **Output:** Catch the bus?

Machine Learning Approach

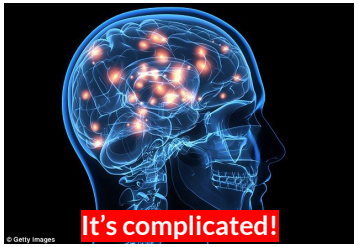
Selecting Predictors

Past Data: Will Marybeth catch the bus to campus?

Marybeth caught the bus to campus 108 of 150 times that it was sunny.

Marybeth never caught the bus when she woke up after 9:00am (sample size of 780 days).

On the one day the sunrise was neon green, Marybeth caught the bus!
Wild, right? Wouldn't anyone wake up early to watch a neon green sunrise?

All possible factors → 

→ **Output:** Catch the bus?

Machine Learning Approach

Selecting Predictors

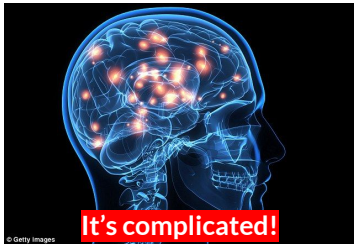
Past Data: Will Marybeth catch the bus to campus?

Marybeth caught the bus to campus 108 of 150 times that it was sunny.

Marybeth never caught the bus when she woke up after 9:00am (sample size of 780 days).

On the one day the sunrise was neon green, Marybeth caught the bus!

Should we use <u>all</u> possible factors in machine learning?

All possible factors → 

→ **Output:** Catch the bus?

Machine Learning Approach

Selecting Predictors

Question: Will Marybeth catch the bus to campus?

Inputs/Predictors:

- Time she woke up
- Trash day
- Weather conditions
- ~~Current record for the Yankees~~
- ~~The color of the sunrise~~
- ...

Truth/Predictand:

- Yes or No

Machine Learning Approach

Data Splitting is a very important aspect of ML design

Machine Learning Approach

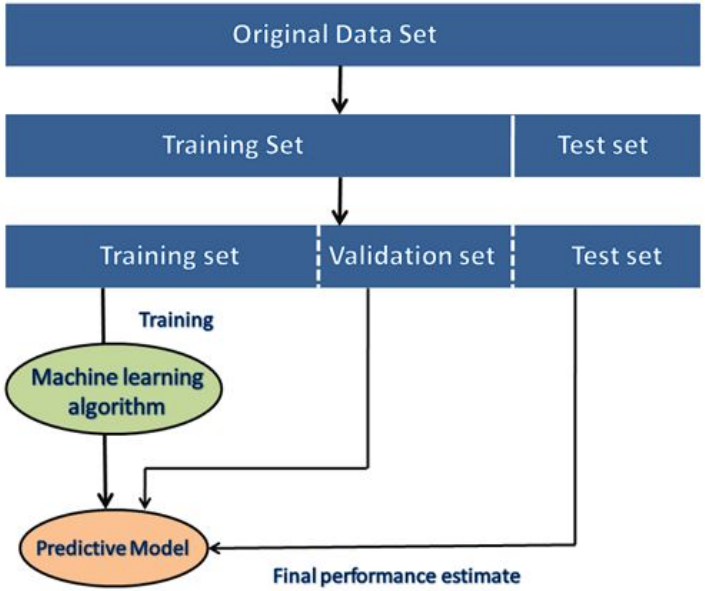
Data Splitting is a very important aspect of ML design

Data split into 3 parts: *training*, *validation*, *testing*

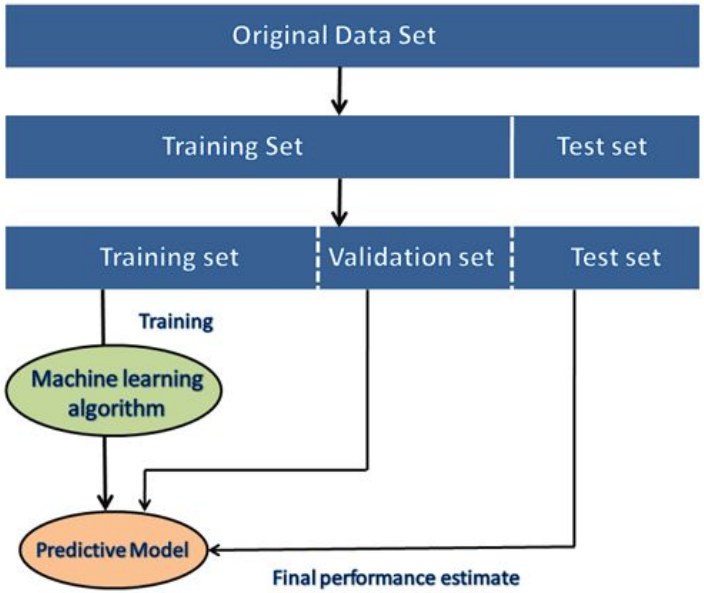
Why?

- Reduce overfitting
- Optimize hyperparameters

*Often need to standardize your data!



Machine Learning Approach



Training Data

Data subset used to fit the ML model; data the model uses to *learn* and optimize (i.e. minimize loss)

Validation Data

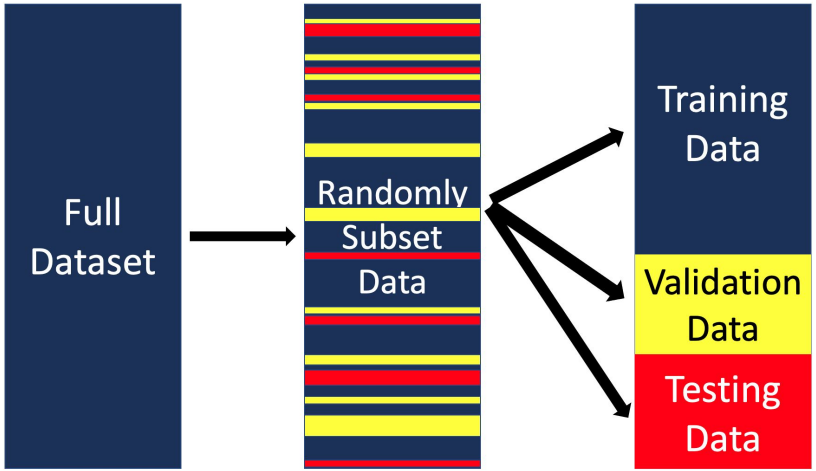
Data subset used to tune model hyperparameters via unbiased evaluation of model fit

Testing Data

Data subset used to evaluate the final ML model on data not seen before

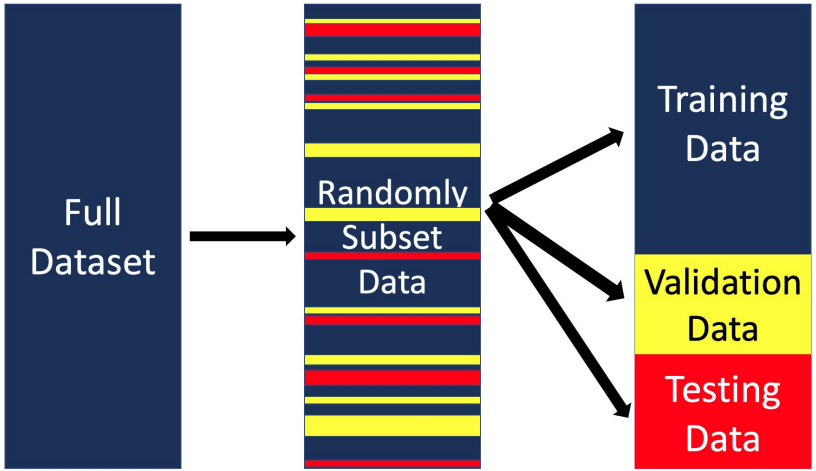
How to Split Data

Random Splitting of Data



How to Split Data

Random Splitting of Data



Split full dataset into 80% training and 20% testing

```
from sklearn.model_selection import train_test_split

x, x_test, y, y_test = train_test_split (x_train,labels,
test_size=0.2, train_size=0.8 )

x_train, x_cv, y_train, y_cv = train_test_split(x,y,test_size = 0.25,
train_size =0.75)
```

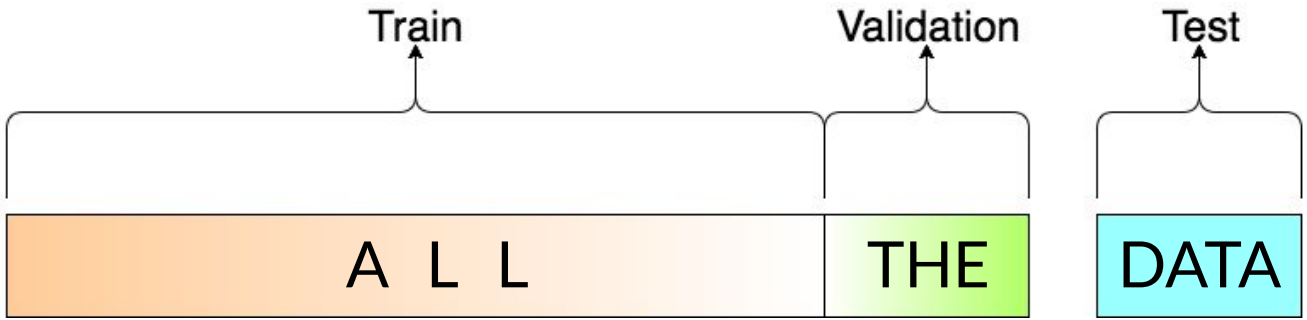
Split training subset into 75% training and 25% validation

How to Split Data

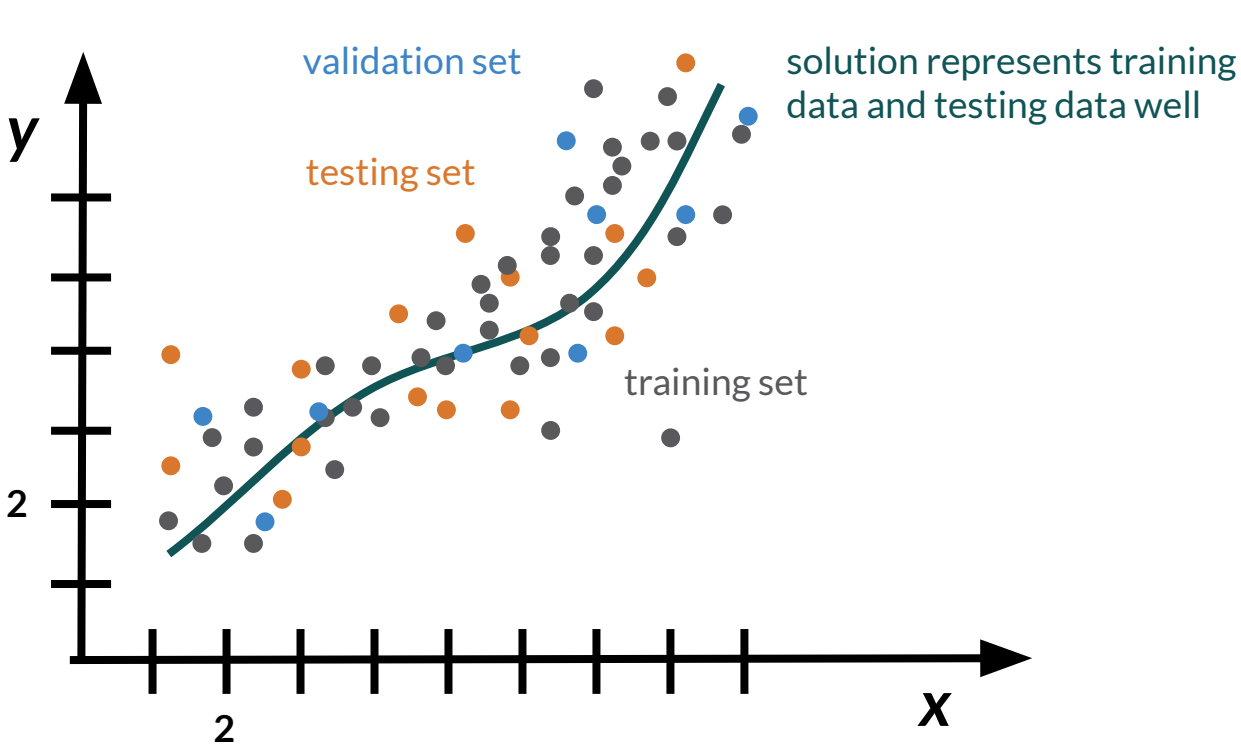
Climate data is often autocorrelated, so we split data chronologically

Example: Full dataset from 1900-2000

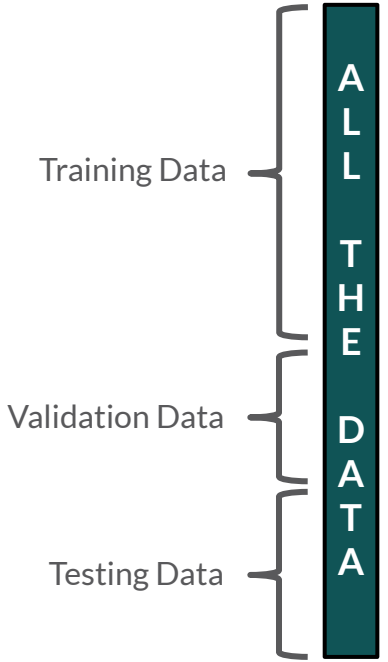
- Training: 1900-1980
- Validation: 1981-1990
- Testing: 1991- 2000



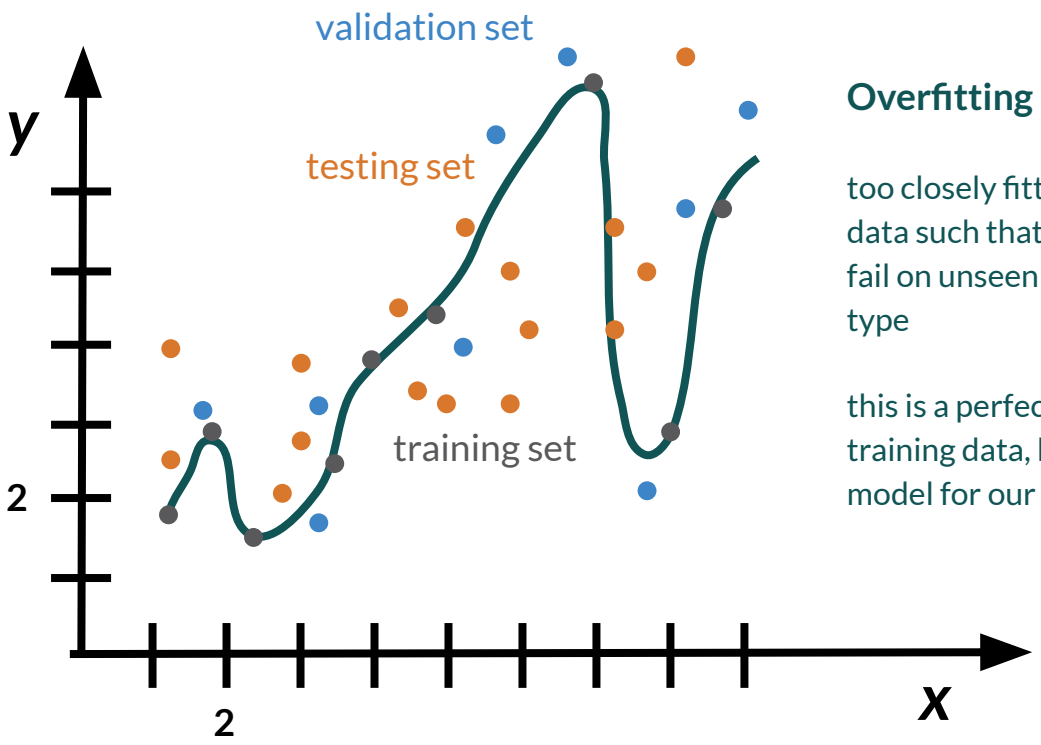
Machine Learning Approach



Groups of Data



Machine Learning Approach

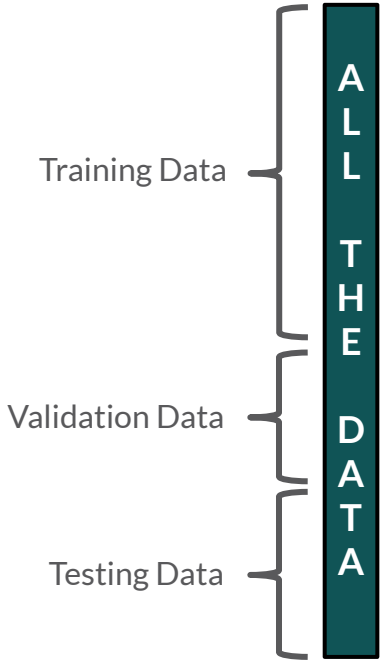


Overfitting

too closely fitting the training data such that the model will fail on unseen data of the same type

this is a perfect model for the training data, but a very poor model for our testing data

Groups of Data



Machine Learning Approach

Training the Model

Training is an iterative process.

Learning happens by training on past data.

Each iteration (epoch) of training, the machine learning model should better fit the testing data.

At the beginning of training, the machine learning model has no skill.

Machine Learning Approach

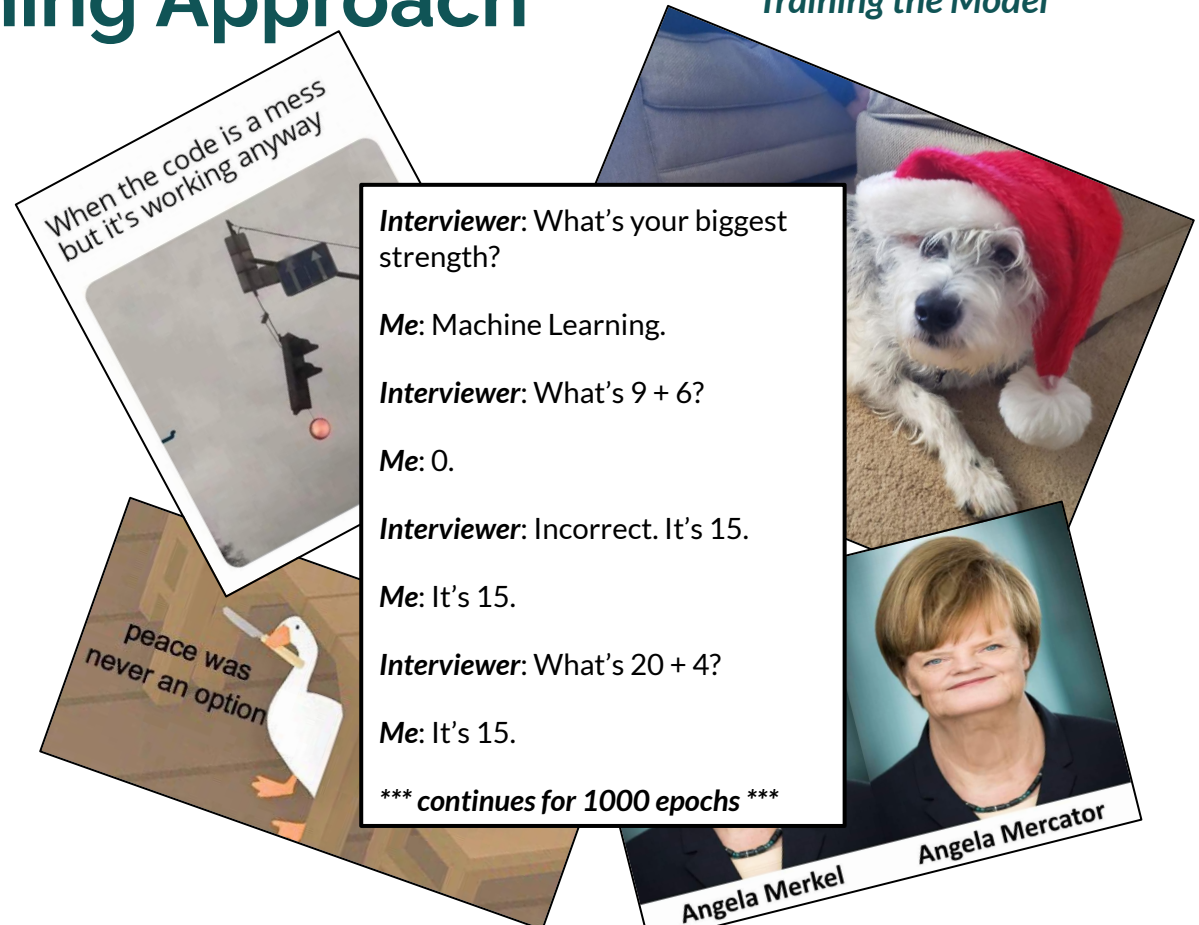
Training the Model

Training is an iterative process.

Learning happens by training on past data.

Each iteration (epoch) of training, the machine learning model should better fit the testing data.

At the beginning of training, the machine learning model has no skill.



Interviewer: What's your biggest strength?

Me: Machine Learning.

Interviewer: What's 9 + 6?

Me: 0.

Interviewer: Incorrect. It's 15.

Me: It's 15.

Interviewer: What's 20 + 4?

Me: It's 15.

****** continues for 1000 epochs ******

Machine Learning Approach

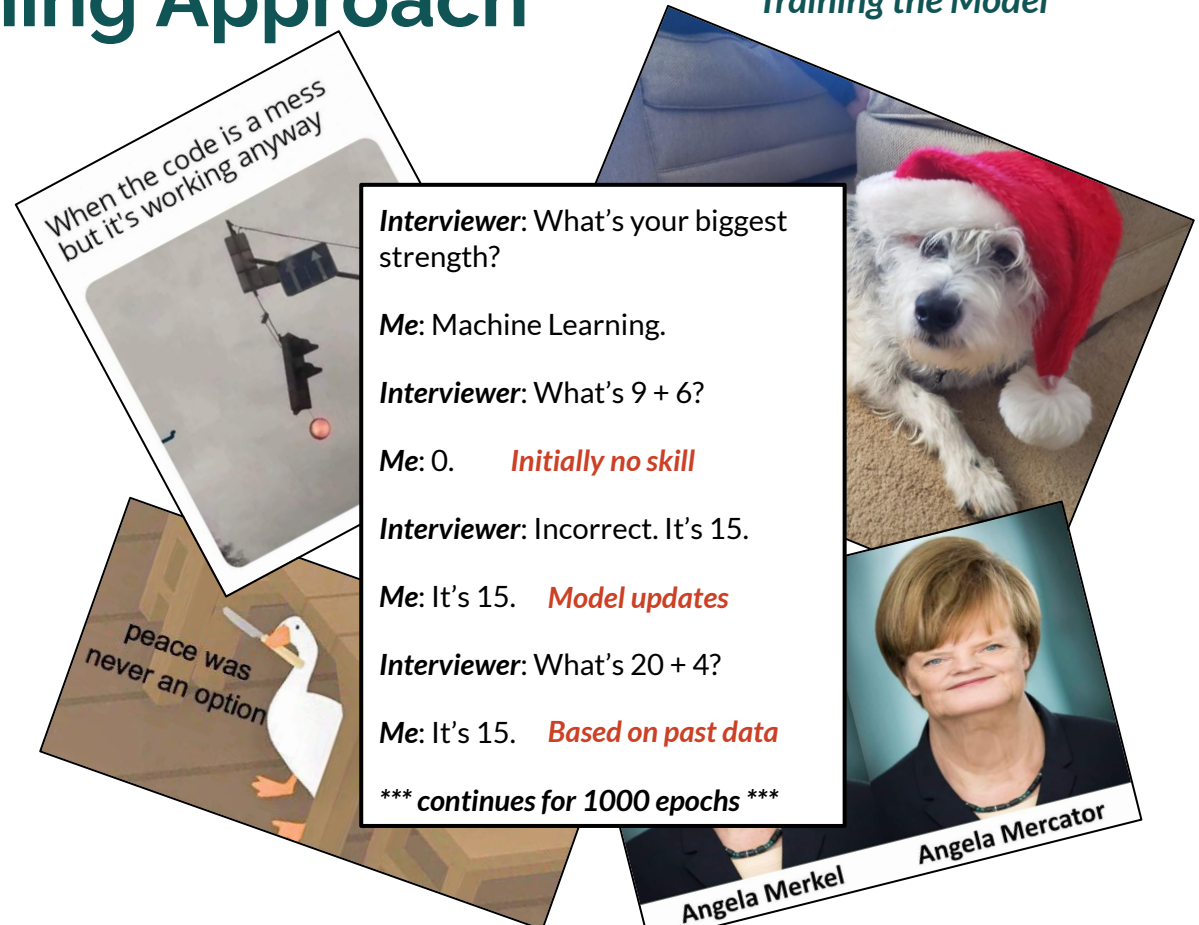
Training is an iterative process.

Learning happens by training on past data.

Each iteration (epoch) of training, the machine learning model should better fit the testing data.

At the beginning of training, the machine learning model has no skill.

Training the Model



Interviewer: What's your biggest strength?

Me: Machine Learning.

Interviewer: What's 9 + 6?

Me: 0. ***Initially no skill***

Interviewer: Incorrect. It's 15.

Me: It's 15. ***Model updates***

Interviewer: What's 20 + 4?

Me: It's 15. ***Based on past data***

****** continues for 1000 epochs ******

Machine Learning Approach

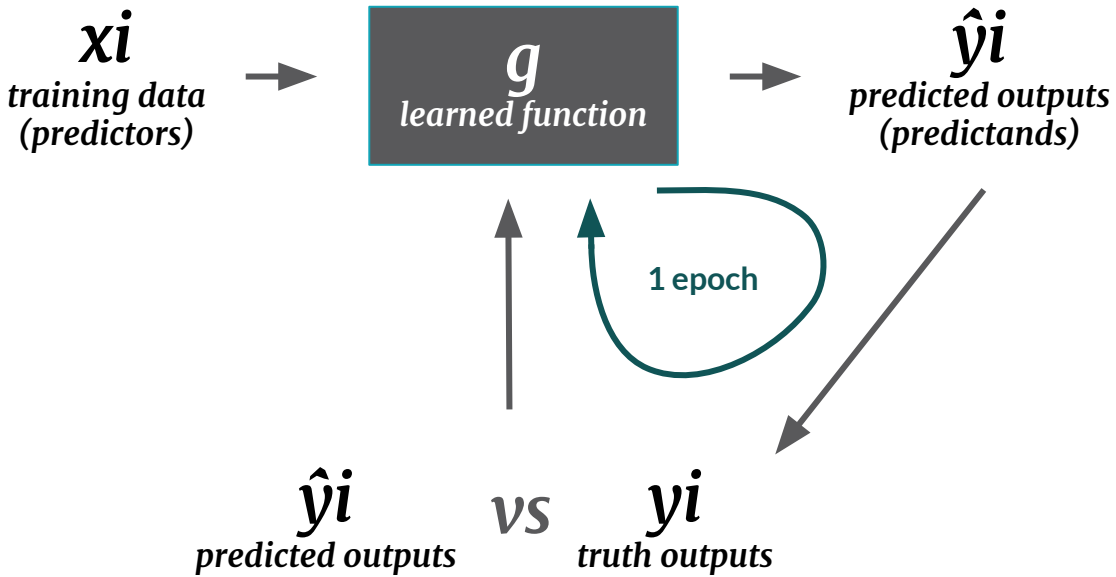
Training the Model

Training data is put into the machine learning model's learned function.

The model outputs its predictions. These predictions are compared with their truth values.

The function is updated to decrease the error between predicted and truth outputs.

This cycle continues.



Some terminology

WHAT WE CALL IT	WHAT ML CALLS IT
Dependent Variable/Right Answer	Label
Variable/Predictor	Feature
Slopes/Regression Coefficients	Weights
Y-intercepts/Constant factor	Bias

Summary

What is ML?

- Machine learning is a technique to fit a function to data
- Is it not constrained by simple mathematical expressions

The Practical Procedure of ML

- Select predictors intelligently
- Split data into training, validation, and testing sets
- The model learns by training iteratively to optimize a loss function

Vocab

- Architecture: y = ax+b Loss Function: RMSE Features/Labels: x / y
- Overfitting: Fitting a model too closely to training data such that it does poorly with unseen data

Decision Trees and Random Forests

A brief overview

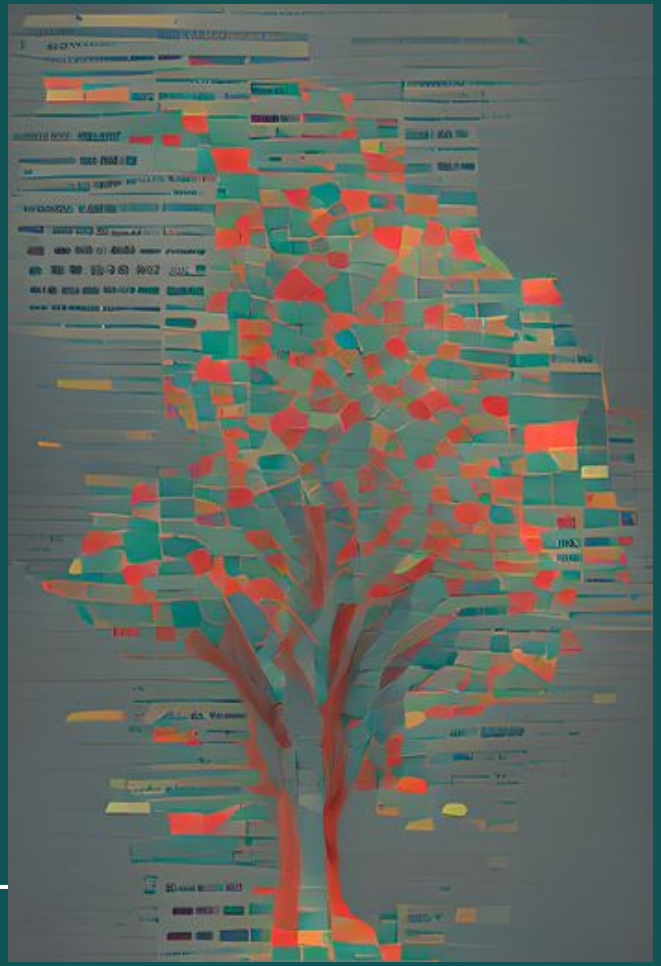


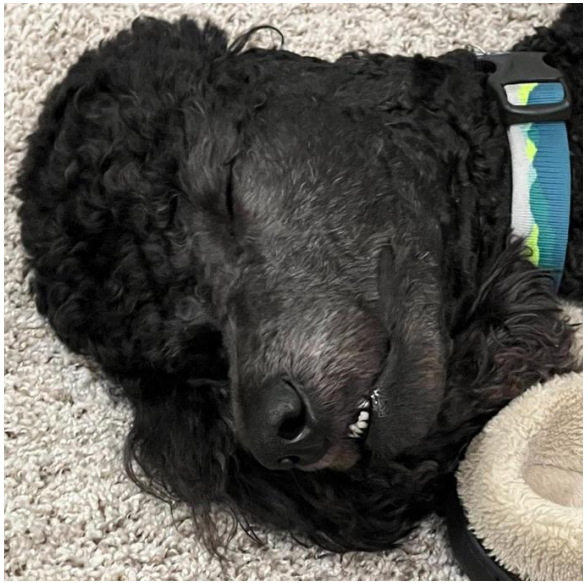
Image generated by Wombo AI

Meet Atlas!

Will Atlas want to play outside?

Playful!

or



Sleepy 💤

Goal: design questions to classify events so predictions with new data are accurate

Observed data/events:
Will Atlas want to play outside?

Predictors:
Temperature, tiredness, boredom, hunger, fur length, snow, etc.

A **decision tree** is a series of questions
Is temperature above 80?
Is it snowy?
Did he play yesterday?

Output
Yes! Atlas is playful
No! Atlas is sleepy

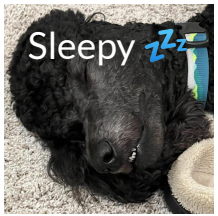
Forest icon made by Freepik from flaticon.com

Will Atlas want to play outside?

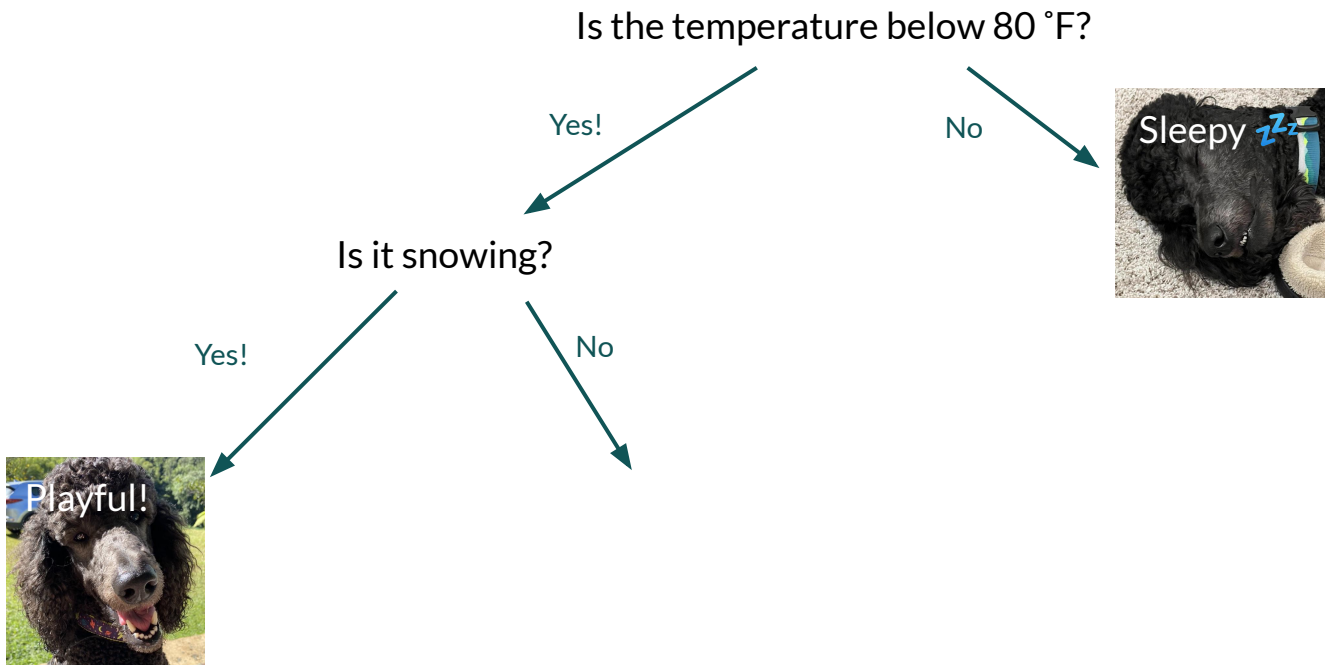
Is the temperature below 80 ˚F?

Yes!

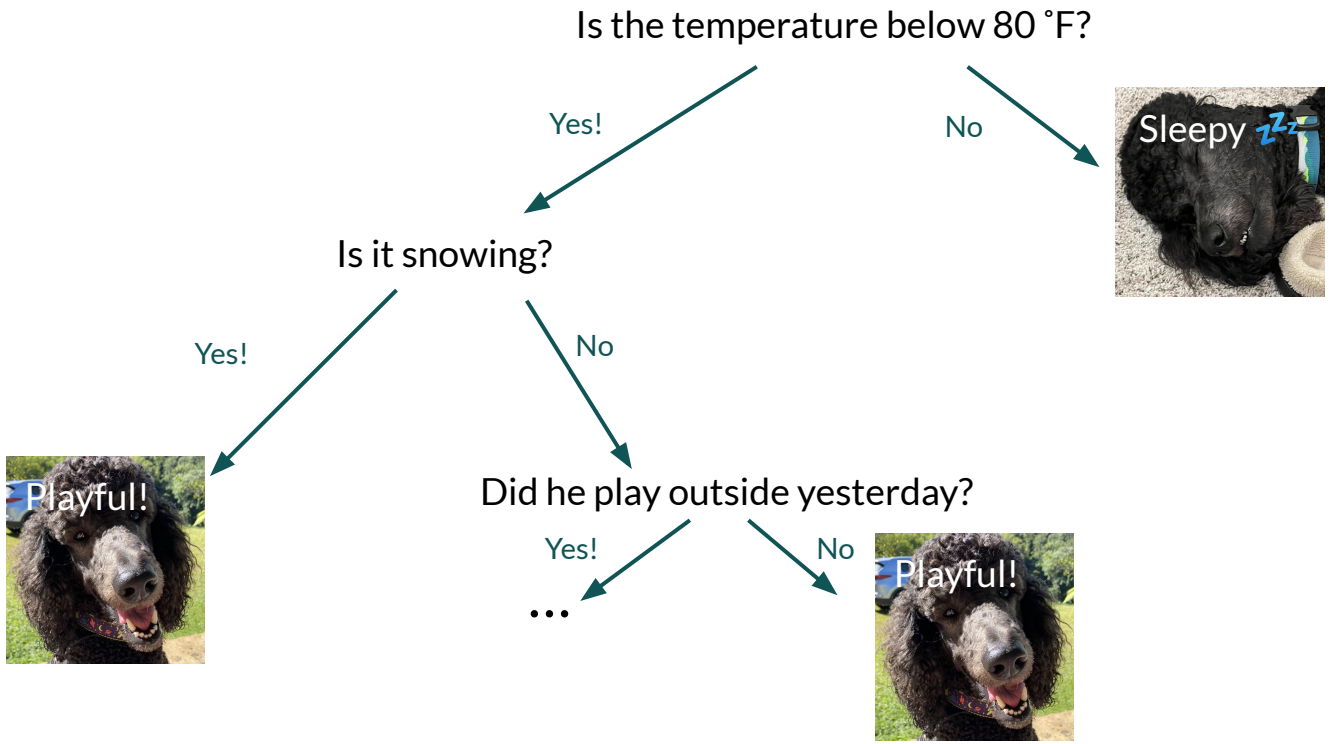
No



Will Atlas want to play outside?



Will Atlas want to play outside?

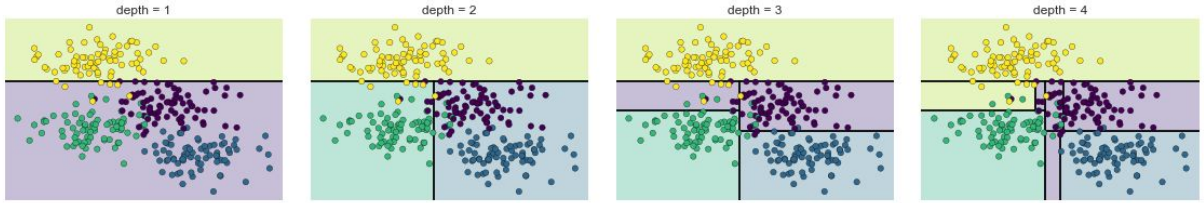


What is a decision tree?

An intuitive supervised learning method for classification (discrete) and regression (continuous variable) tasks

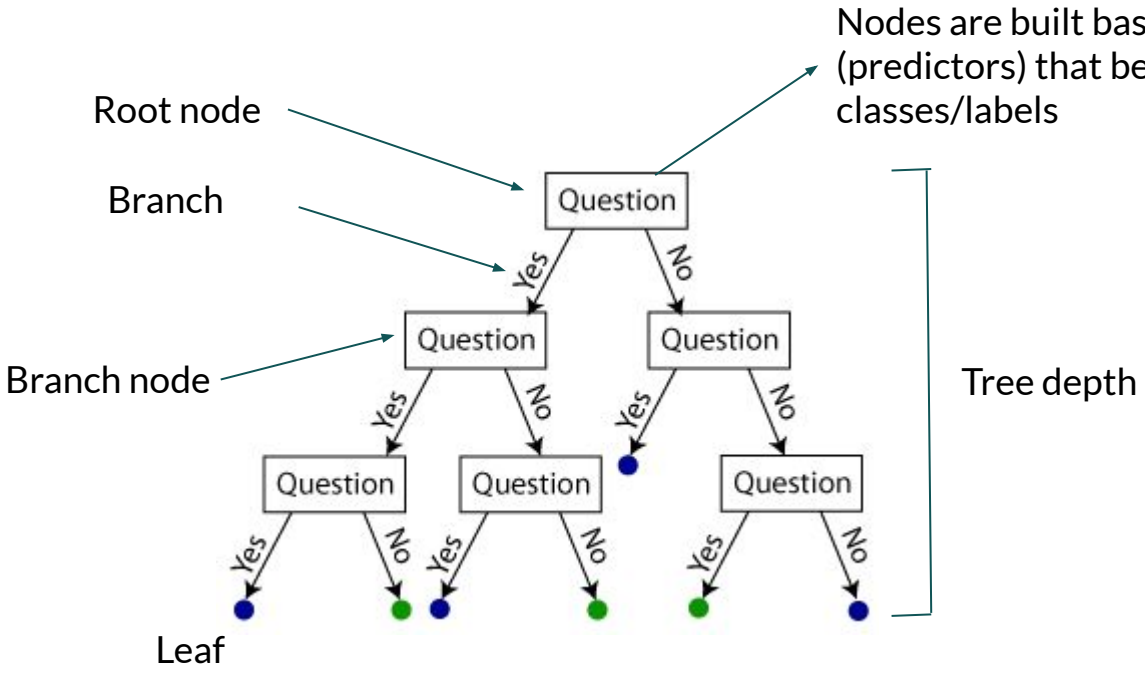
Ask a series of questions to discriminate ***labels*** (i.e., classifications)

Increasing depth of tree →



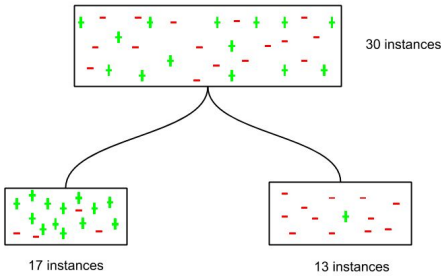
Courtesy: Python Data Science Handbook

Decision tree structure



shapeofdata.wordpress.com/2013/07/09/random-forests/

Splitting and stopping nodes



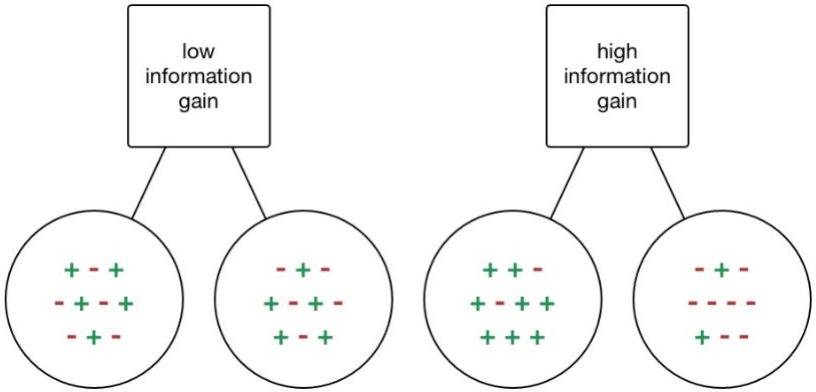
Splitting a node: Child nodes are constructed from additional features that best separate the classes/labels through the depth of the tree

- Use a “gain” metric to assess which feature is best for splitting
- Maximize ***purity*** of node

Stopping at leaf node: Once a stopping criteria is reached (e.g., number of samples at the node, purity), branch is complete

https://www.hackerearth.com/practice/machine-learning/machine-learning-algorithms/ml-decision-tree/tutorial/

There are three common gain metrics



Maximize information gain = **minimizing *impurity:***

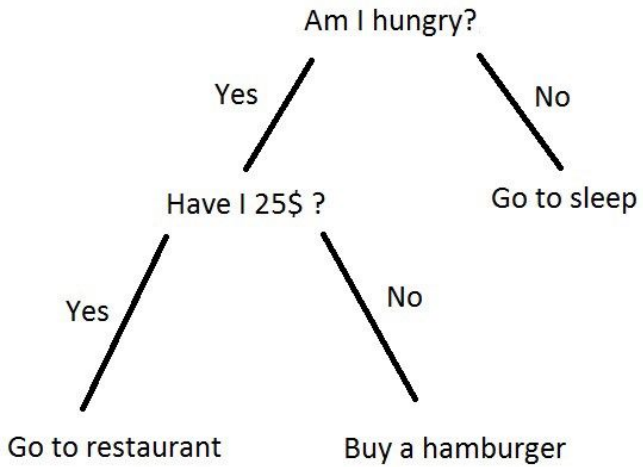
- Gini
- Entropy
- Variance reduction – only for regression

Greedy algorithm: Make the locally optimal decision at each node

https://becominghuman.ai/decision-trees-in-machine-learning-f362b296594a

Advantages of Decision Trees

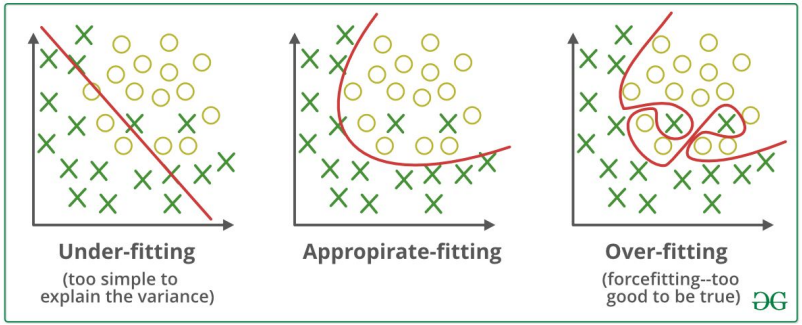
- Easy to understand (interpretable)
- Categorical and continuous variables
- Implicit feature selection
- Fewer tunable parameters than many other ML methods



codeproject.com/Articles/4047359/Step-by-Step-Guide-To-Implement-Machine-Learning-2

Disadvantages of Decision Trees

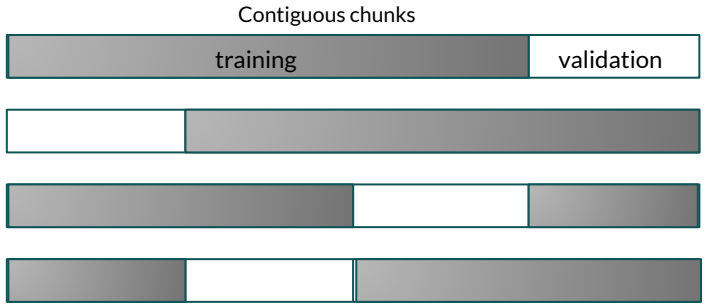
- Easy to overfit with complex trees
- Greedy algorithm is not always globally optimal
- Deterministic – will always produce the same tree for a given set of training data



https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

Reduce overfitting and improve predictions

- Tree depth
- Minimum number of samples to split
- Minimum number of samples for leaf
- Training dataset length and selection
- K-fold cross-validation
- Ensemble methods: **random forest!**



Random forests are decision tree ensembles!

Observed data/events:
Other meteorological data

Predictors:
Temperature, humidity, insolation, wind speed, wind direction, precipitation, date of year, etc.

Each **tree** in the **forest** is a series of questions
Is temperature above 80?
Is the wind from the west?
Is it raining?

Output
Ozone class (good, fair, poor)

Relative frequency of a label across all trees describes probabilistic occurrence (i.e. forecast)

Forest icon made by Freepik from flaticon.com

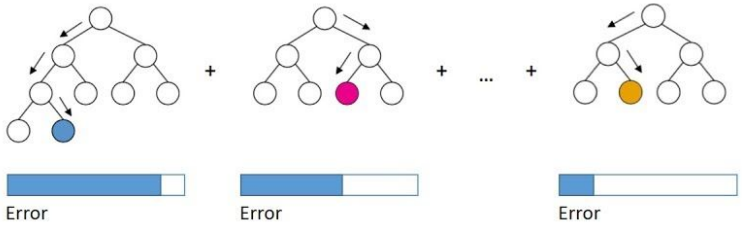
What is a random forest?

-An ensemble of unique decision trees that become uncorrelated by selecting random subsets of features at each node

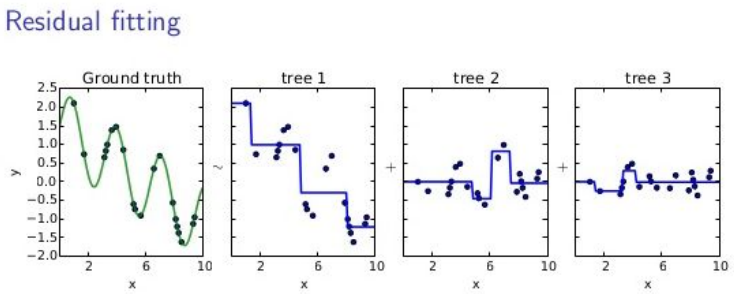
-Bagging instead of boosting

-Random sampling of training data

-Aggregated decision trees can significantly decrease the prediction variance with small increases to model bias (Herman and Schumacher 2016)



medium.com/@shvbajpai/how-to-master-python-for-machine-learning-from-scratch-a-step-by-step-tutorial-8c6569895cb0



https://www.slideshare.net/DataRobot/gradient-boosted-regression-trees-in-scikitlearn

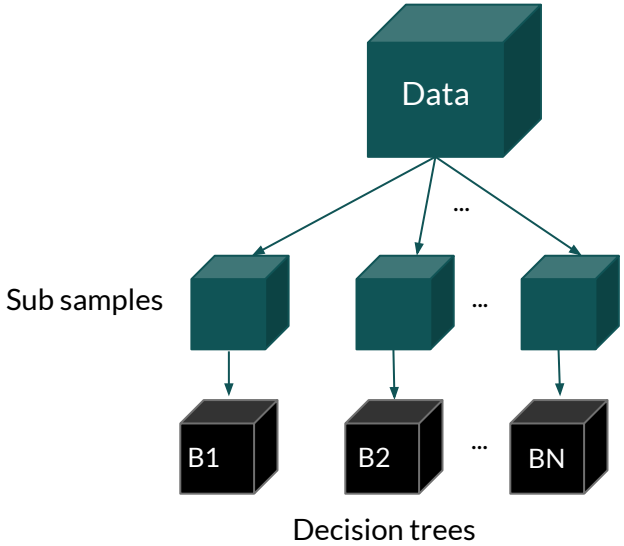
Random Forest Development

Overfitting can still occur, particularly if a small subset of features are great predictors

Bagging, aka bootstrapping:

1) Sample with replacement ***n*** training examples for each tree, ***B*** times
2) Random subset of ***m*** features used at each decision node

Unique decision trees that are then uncorrelated



Random Forest Tunable Parameters

- Number of trees – smaller forests may lead to poor performance, but asymptotes to a limit
- Tree depth – deeper trees can fit more complex behavior, but can lead to overfitting
- Leaf samples – can help prevent overfitting

This is not an exhaustive list but some of the most notable in our experience!

Why are RFs good for the job?

- Probabilistic prediction versus deterministic
- Maintain the same intuitive structure for users as decision trees
- Relatively simple to implement and tweak (few tunable parameters compared to other ML techniques)
- Computationally cheap

Where are they used?

AGU100 ADVANCING EARTH AND SPACE SCIENCE

https://doi.org/10.1029/2019EA001058

Earth and Space Science

RESEARCH ARTICLE
10.1029/2019EA001058

Key Points:
- The RF model exhibits high

Daily Global Solar Radiation in China Estimated From High-Density Meteorological Observations: A Random Forest Model Framework

Table 1
List of Predictor Variables for Estimating DGSR

Variable		Unit	Selected[a]	Description
Geographical factors	Longitude	°	Y	—
	Latitude	°	Y	—
	Altitude	m	Y	—
Time factor	DOY	day	Y	Day of year
Estimated factor	DGSR	MJ/m^2	Y	Daily global solar radiation
Meteorological factors	PRS-mean	hPa	Y	Daily average atmospheric pressure
	PRS-min	hPa	Y	Daily minimum atmospheric pressure
	RHU	%	Y	Daily average relative humidity
	SSD	h	Y	Daily sunshine duration
	SAT-mean	°C	Y	Daily mean surface air temperature
	DTR	°C	Y	Diurnal temperature range (SAT-max minus SAT-min)
	WIN	m/s	Y	Daily average wind speed
	LST-mean	°C	Y	Daily mean land surface temperature
	LST-max	°C	Y	Daily maximum land surface temperature
	LST-min	°C	Y	Daily minimum land surface temperature
	EVP	mm	N	Daily evaporation capacity
	PRS-max	hPa	N	Daily maximum atmospheric pressure
	SAT-min	°C	N	Daily minimum surface air temperature
	SAT-max	°C	N	Daily maximum surface air temperature
	PRE-2008	mm	N	Precipitation from 20:00 to 8:00
	PRE-0820	mm	N	Precipitation from 8:00 to 20:00
	PRE-2020	mm	N	Precipitation from 20:00 to 20:00

[a]Y: Included in the reduced model after variable selection.

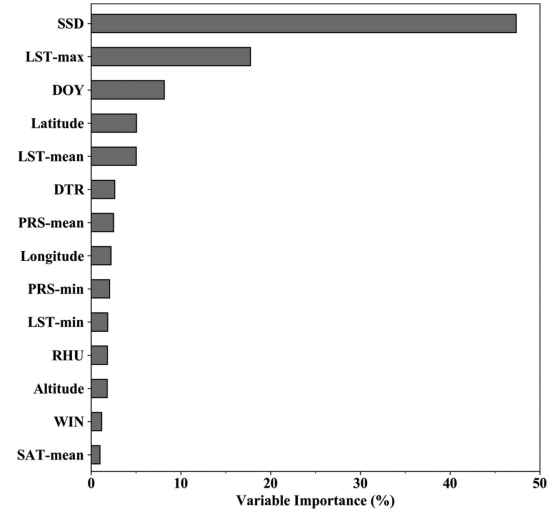


Figure 5. Variable importance plot for the RF model estimating the DGSR in China.

Where are they used?

Environmental Science & Technology

Article

Cite This: *Environ. Sci. Technol.* 2018, 52, 4173−4179

pubs.acs.org/est

Predicting Daily Urban Fine Particulate Matter Concentrations Using a Random Forest Model

Cole Brokamp,*[,†,‡] Roman Jandarov,[§] Monir Hossain,[†] and Patrick Ryan[†,§,‡]

[†]Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center, Cincinnati, Ohio 45229, United States
[‡]Department of Pediatrics, University of Cincinnati, Cincinnati, Ohio 45267, United States
[§]Department of Environmental Health, University of Cincinnati, Cincinnati, Ohio 45267, United States

https://doi.org/10.1021/acs.est.7b05381

Table 1. Summary of Data Used To Train Spatiotemporal Random Forest Models

		resolution	
category	variables	spatial	temporal
AOD	Terra and Aqua Aerosol 5Min 3km L2	3 km nadir	daily
PM2.5	AQS and CCAAPS mean 24-h $PM_{2.5}$	exact location	daily
weather	visibility, planetary boundary height, temperature, relative humidity, total precipitation and rate, pressure, wind speed and direction	none	daily
land cover	percent imperviousness	30 × 30 m	none
roadways	total length of major roadways	exact location	none
greenspace	normalized differential vegetation index	30 × 30 m	none
spatiotemporal	grid identifier, year, day of year	1 × 1 km	daily
convolution layer	median $PM_{2.5}$ from 3 nearby days	none	daily

Where are they used?

Money Doesn't Grow on Trees, but Forecasts Do: Forecasting Extreme Precipitation with Random Forests

DOI: 10.1175/MWR-D-17-0250.1

GREGORY R. HERMAN AND RUSS S. SCHUMACHER

Department of Atmospheric Science, Colorado State University, Fort Collins, Colorado

TABLE 2. List of background predictors used in this study and their associated symbols and descriptions.

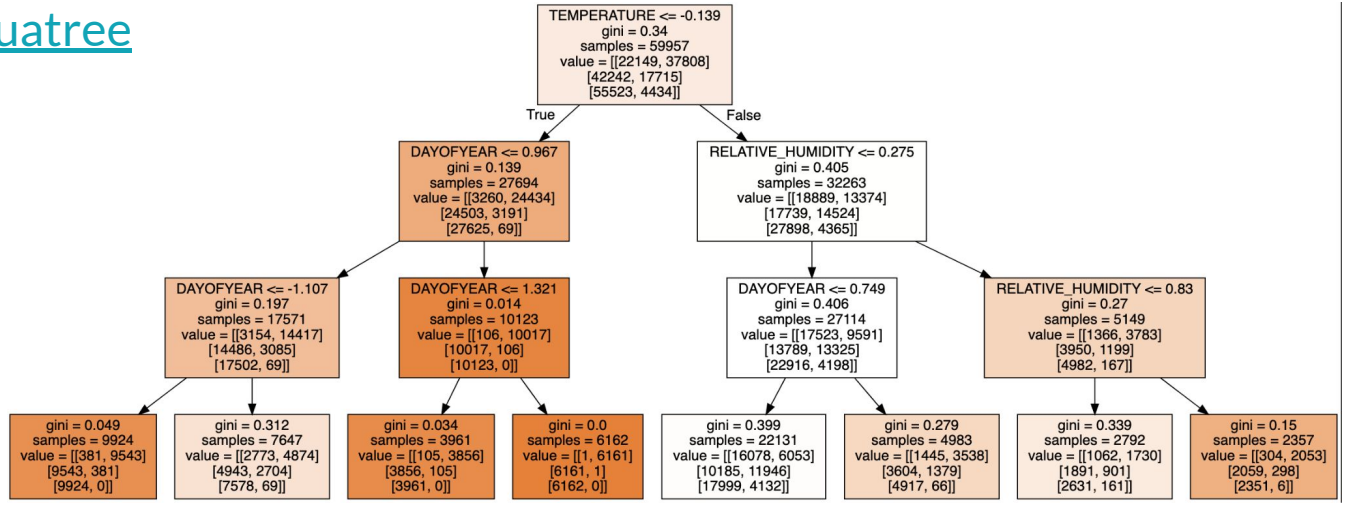
Symbol	Description
ARI1_LOCAL_MEDIAN	Median of 1-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI1_LOCAL_MIN	Minimum of 1-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI1_LOCAL_MAX	Maximum of 1-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI10_LOCAL_MEDIAN	Median of 10-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI10_LOCAL_MIN	Minimum of 10-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI10_LOCAL_MAX	Maximum of 10-yr ARIs whose closest GEFS/R grid point is the forecast point.
ARI1_REGIONAL_MEDIAN	Median of 1-yr ARIs that lie within the domain from which model predictors are drawn.
ARI1_REGIONAL_MIN	Minimum of 1-yr ARIs that lie within the domain from which model predictors are drawn.
ARI1_REGIONAL_MAX	Maximum of 1-yr ARIs that lie within the domain from which model predictors are drawn.
ARI10_REGIONAL_MEDIAN	Median of 10-yr ARIs that lie within the domain from which model predictors are drawn.
ARI10_REGIONAL_MIN	Minimum of 10-yr ARIs that lie within the domain from which model predictors are drawn.
ARI10_REGIONAL_MAX	Maximum of 10-yr ARIs that lie within the domain from which model predictors are drawn.
LAT	Latitude of forecast point
LON	Longitude of forecast point

Now we will play with sample code!

Estimate ozone air quality class at Joshua Tree National Park using meteorological data

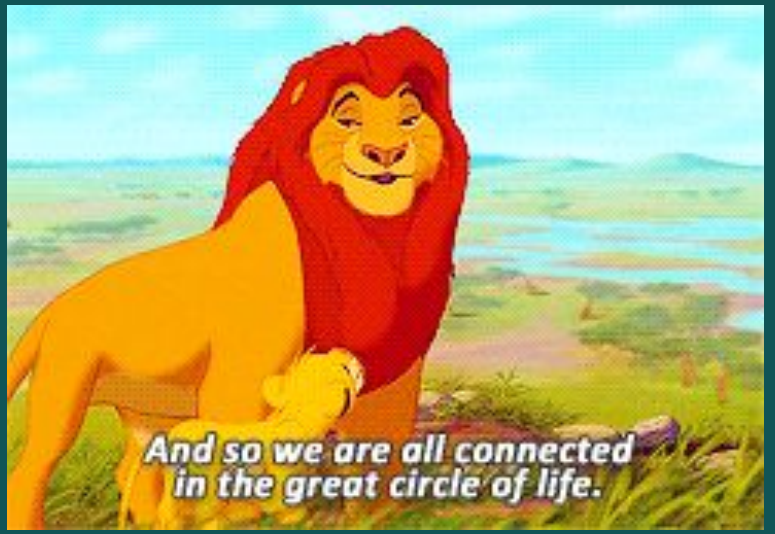
Try to beat the baseline! Validation weighted categorical accuracy = 0.537

Link: [rf_ozone_joshuatree]

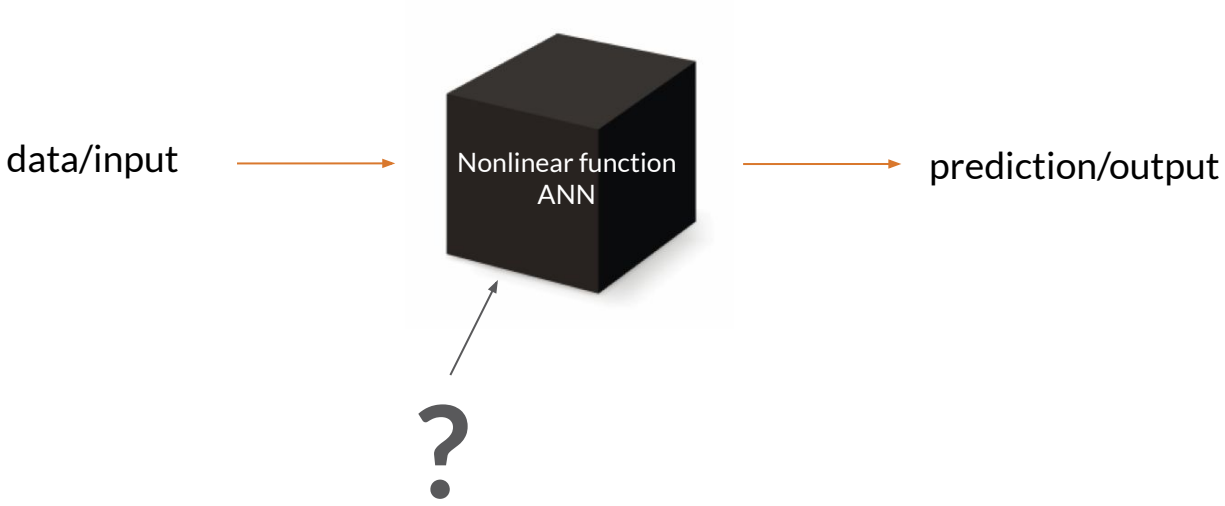


Artificial Neural Networks (ANN)

"It's all connected"



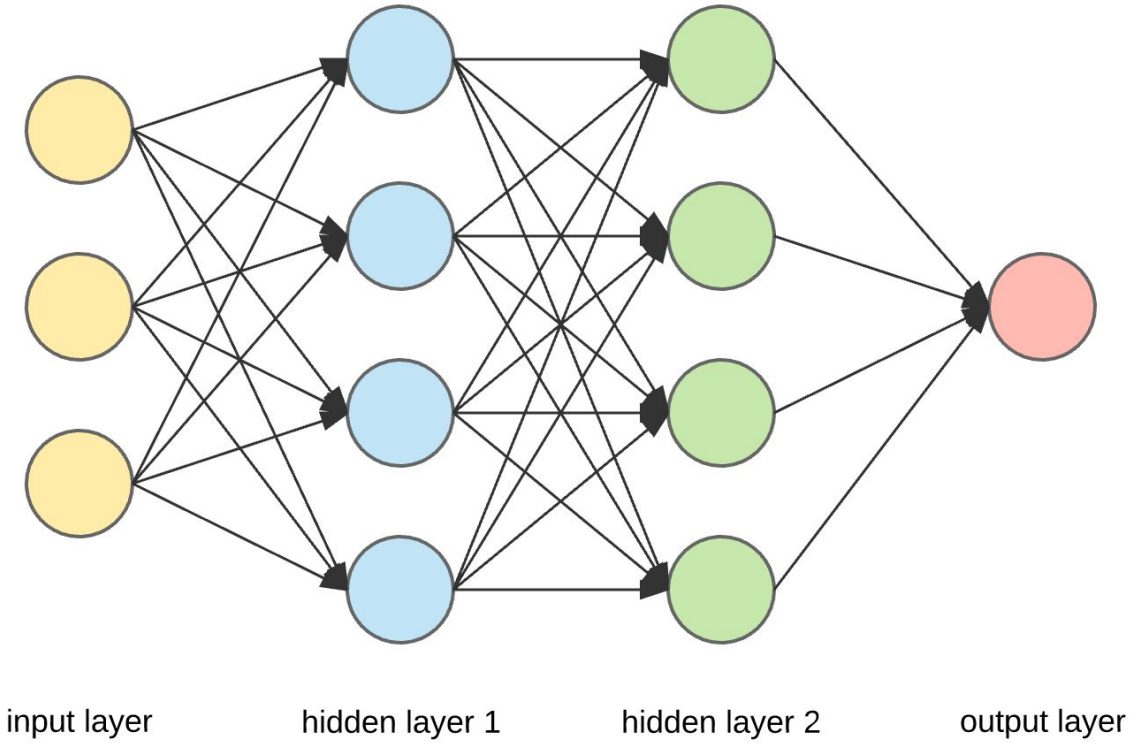
ANNs allow us to utilize a lot of information easily



So how does an ANN work?

ANNs consists of multiple nodes (the circles) and layers that are all connected and using basic math gives out a result.

These are called feed forward networks!



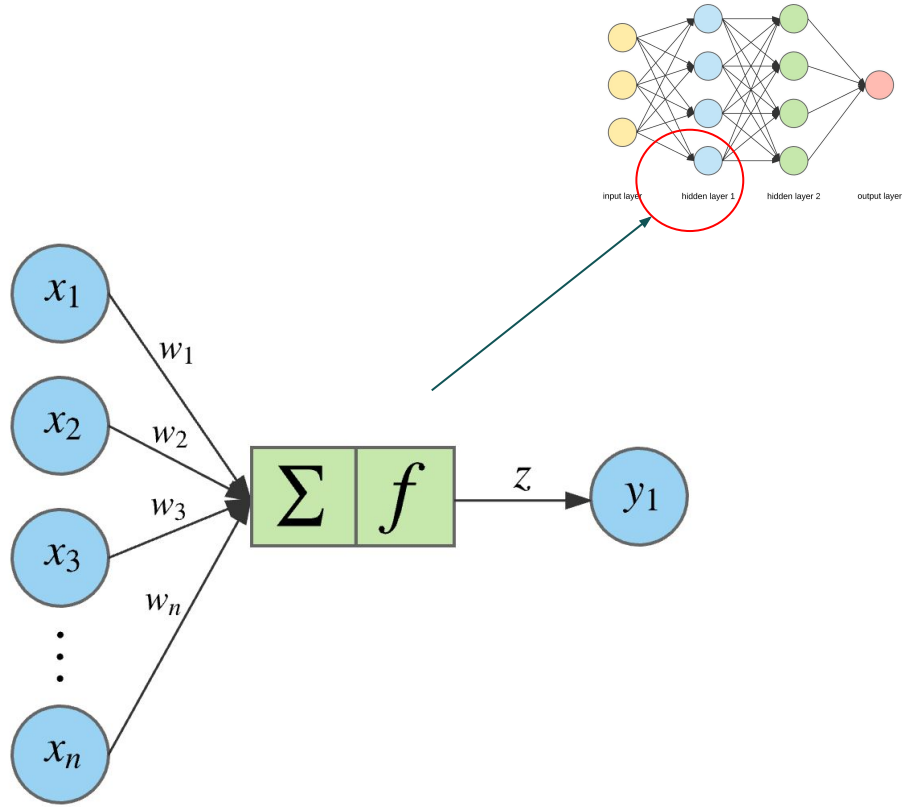
https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

So what happens in the hidden layers?

In each individual node the values coming in are weighted and summed together and bias term is added and activation.

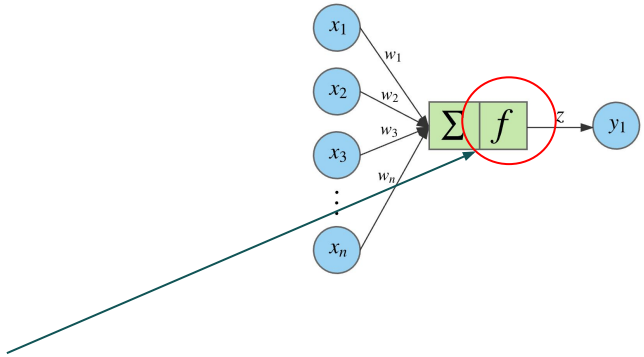
$$z = f_{activation}(w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b)$$

Linear regression with non-linear mapping by an “activation function”



https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Activation function?



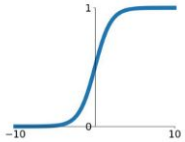
Activation function determines, if information is moving forward from that specific node.

This is the step that allows for nonlinearity in these algorithms, without activation all we would be doing is linear algebra!

Activation Functions

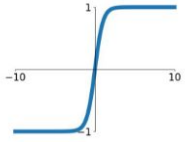
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



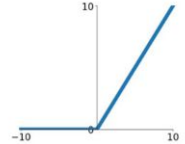
tanh

$$\tanh(x)$$



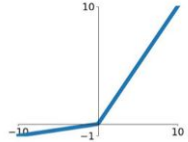
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

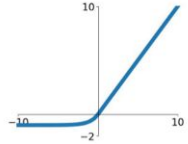


Maxout

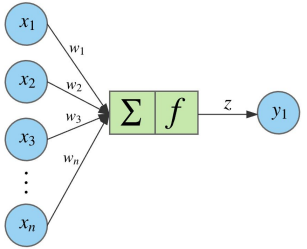
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



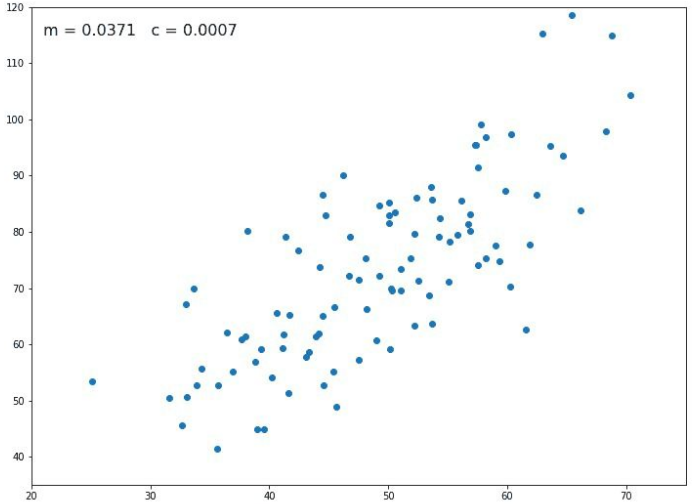
So how do we train a network?



So training of the network is merely determining the weights “w” and bias/offset “b" with the addition of nonlinear activation function.

$$z = f_{activation}(w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_nx_n + b)$$

Goal is to determine the **best function** so that the output is as correct as possible; typically involves choosing “**weights**”



How do we define “best” function?

This is where domain scientist can be very helpful!

You know the data and the goal you’re working towards, so you know the best, which loss function to use.

Basic MSE or MAE works well for regression tasks!

$$\text{error} = \frac{1}{N}\sum_{t=1}^{N}(\tilde{y}_t - y_t)^2$$

Cost/Loss Function

Find a quantity you want to minimize (the “loss”) to help determine the weights

$$\text{error} = \frac{1}{N}\sum_{t=1}^{N}(\tilde{y}_t - y_t)^2$$

loss/cost

ANN prediction

truth/actual

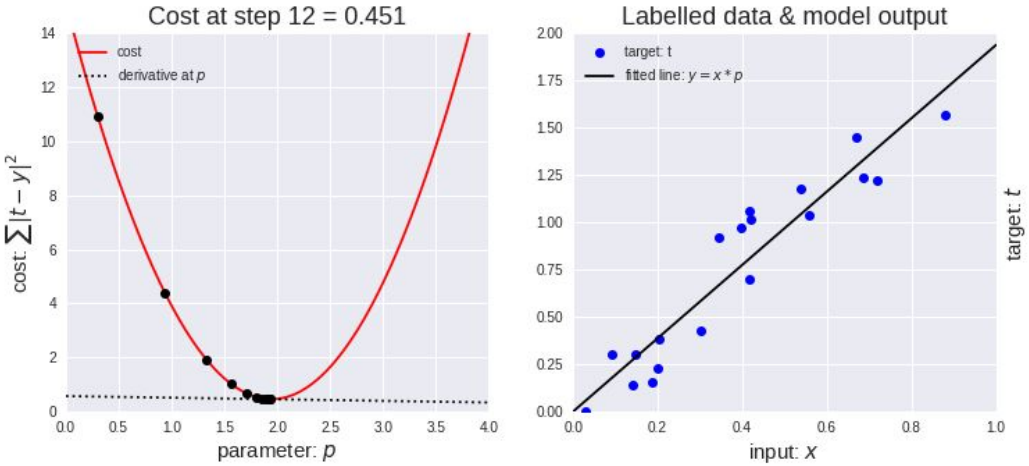
How do we find the minima in real life?

Let's start with an easy linear example

$$\tilde{y}_t = a_0 + a_1 x_t$$

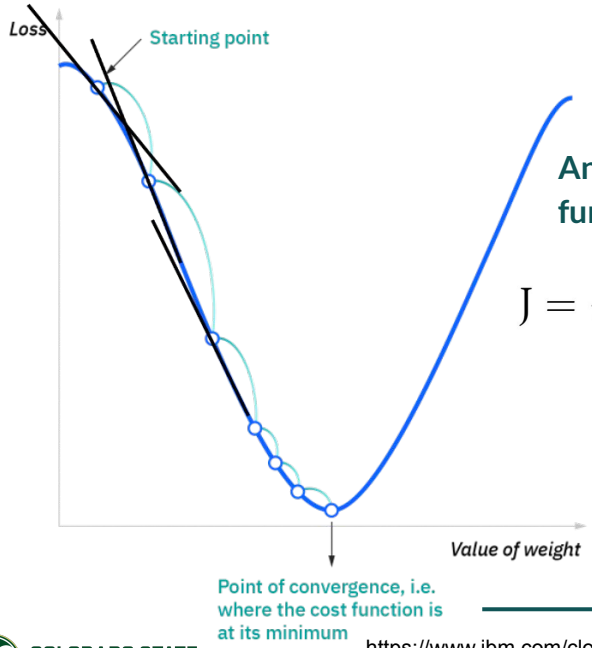
An example loss function (MSE)

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$$



When hiking in Colorado, if the path up isn't clear you choose the steepest incline, and you will find the top. Gradient descent is the same principle but reversed.

Gradient Descent



An example loss function (MSE)

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$$

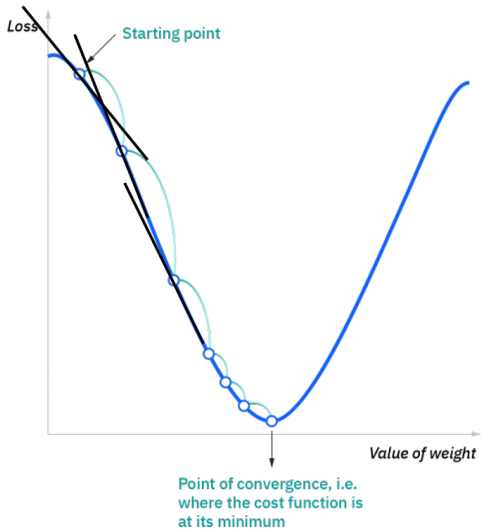
Is a technique to find the weight that minimizes the loss function.

This is done by starting with a random point, the gradient (the black lines) is calculated at that point. Then the negative of that gradient is followed to the next point and so on. This is repeated until the minimum is reached.

https://www.ibm.com/cloud/learn/gradient-descent

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$$

$$\tilde{y}_t = a_0 + a_1 x_t$$



Gradient descent mathematically

So, let's think our example loss function J. The gradient descent formula tells us that the next location depends on the negative gradient of J multiplied by the learning rate λ.

$$J_{t+1} = J_t - \lambda \nabla J_t$$

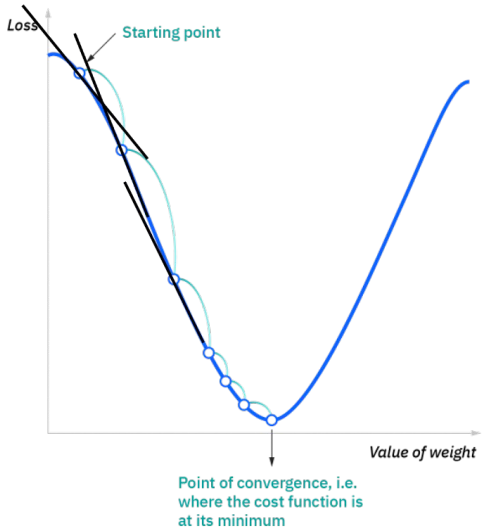
As the RMSE (our loss function) depends on the linear function and its weights a0 and a1, the gradient is calculated as partial derivatives with relation to the weights

$$J = J(a_0, a_1) \qquad \nabla J_t = \begin{bmatrix} \frac{\partial J}{\partial a_0} \\ \frac{\partial J}{\partial a_1} \end{bmatrix}$$

https://www.ibm.com/cloud/learn/gradient-descent

Partial derivatives for RMSE (based on linear regression)

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2 = \frac{1}{N}\sum_{i=1}^{N}((a_0 + a_1 x_i) - y_i)^2$$



From this, we can calculate

$$\frac{\partial J}{\partial a_0} = \frac{\partial}{\partial a_0}\left(\frac{1}{N}\sum_{i=1}^{N}((a_0 + a_1 x_i) - y_i)^2\right) = \frac{2}{N}\sum_{i=1}^{N}((a_0 + a_1 x_i) - y_i)$$
$$= \frac{2}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{\partial}{\partial a_1}\left(\frac{1}{N}\sum_{i=1}^{N}((a_0 + a_1 x_i) - y_i)^2\right) = \frac{2}{N}\sum_{i=1}^{N}((a_0 + a_1 x_i) - y_i)x_i$$
$$= \frac{2}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)x_i$$

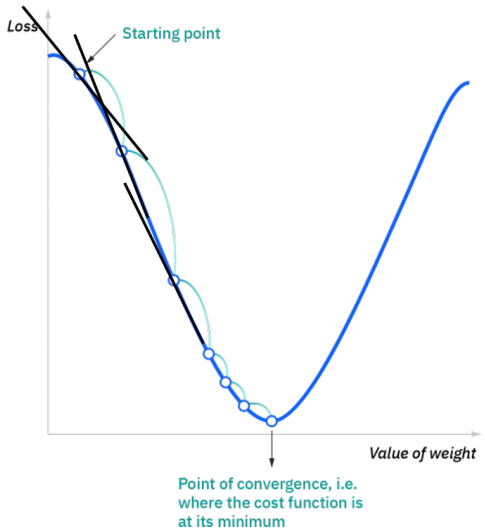
https://www.ibm.com/cloud/learn/gradient-descent

Gradient descent mathematically

After taking the derivatives, the rest is easy!

$$a_{0,new} = a_0 - \lambda \frac{\partial \text{Loss}}{\partial a_0}$$

$$a_{1,new} = a_1 - \lambda \frac{\partial \text{Loss}}{\partial a_1}$$



The only other thing one must pay attention to is the learning rate λ (how big of a step to take). Too small and finding the right weights takes forever, too big and you might miss the minimum.

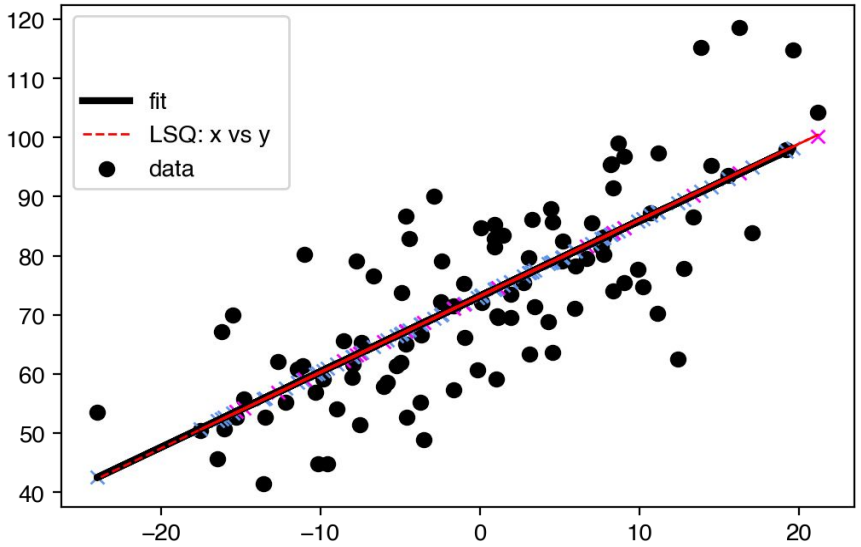
https://www.ibm.com/cloud/learn/gradient-descent

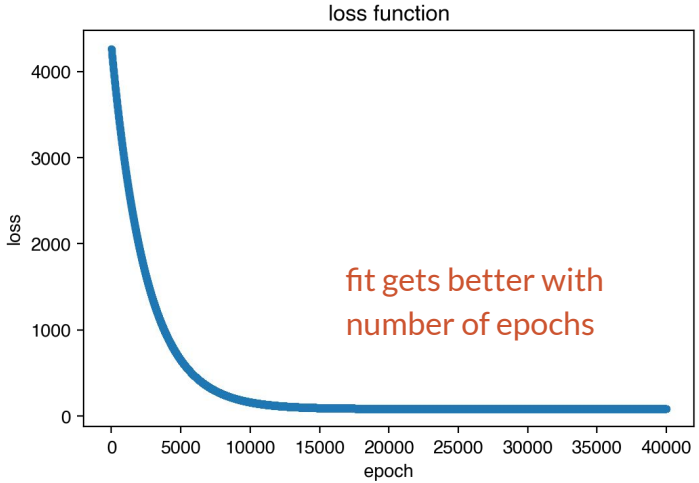
Code it up!

```
# Building the model
m = np.random.uniform()
c = np.random.uniform()

L = 0.0001  # The learning Rate
epochs = 40000  # The number of iterations to perform gradient descent

print('  slope (m)             y-int (c)')
print('-----------------------------------')

errorHistory = np.empty((epochs,))
# Performing Gradient Descent
for i in range(epochs):

    Y_pred = my_fit(m,c,Xtrain)  # The current predicted value of Y
    gradLoss_m, gradLoss_c = my_gradLoss(Xtrain, Ytrain, Y_pred)
    m = m - L * gradLoss_m  # Update m
    c = c - L * gradLoss_c  # Update c

    errorHistory[i] = 1/float(len(Y))*np.sum((Ytrain - Y_pred)**2)

print (m, c)
print('done training')
```

```
# define functional fits
def my_fit(m,c,x):
    # m is the slope of the line
    # c is the y-intercept
    # x is the values to evaluate
    # output: y, the evaluated values
    y = m*x + c
    return y

def my_gradLoss(xtrain,ytrain,ypred):
    n = float(len(xtrain))
    gradLoss_m = 2/n * sum(xtrain * (ypred-ytrain)) #derivative wrt m
    gradLoss_c = 2/n * sum((ypred-ytrain)) #derivative wrt c

    return gradLoss_m, gradLoss_c
```

Code it up!

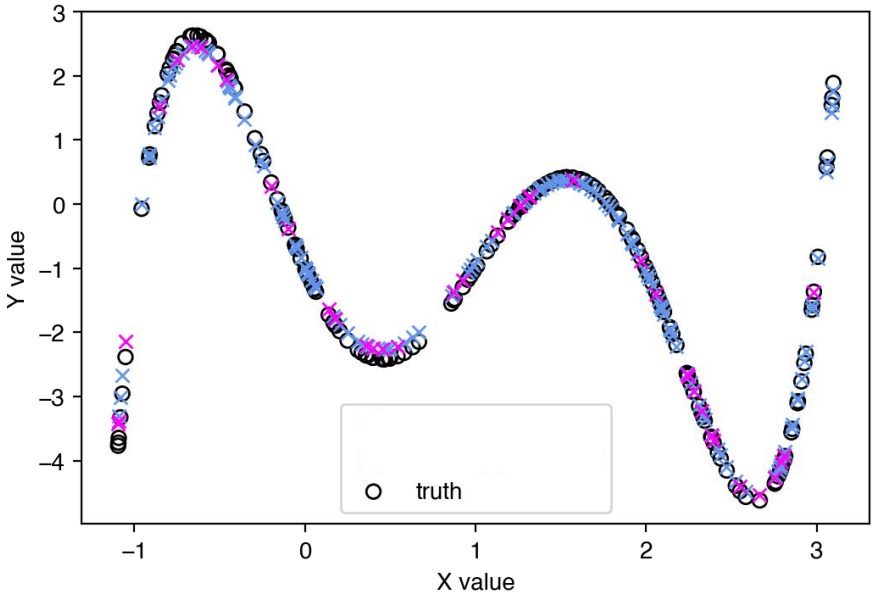


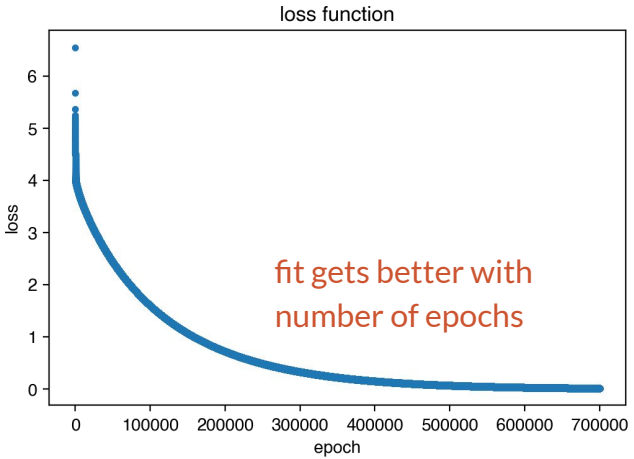


# of iterations through the data

Code it up!

You can do this for polynomials too!





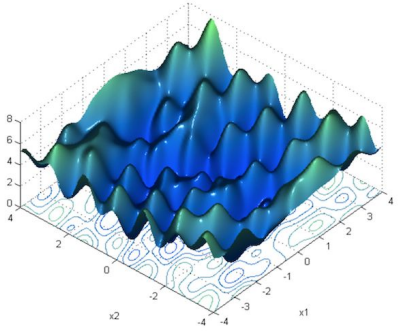
# of iterations through the data

So far so good - this all looks super easy! What's the catch?

We calculated the gradients by hand because we knew the functional form we wanted to fit (**a polynomial**).

So far so good - this all looks super easy! What's the catch?

We calculated the gradients by hand because we knew the functional form we wanted to fit (**a polynomial**).

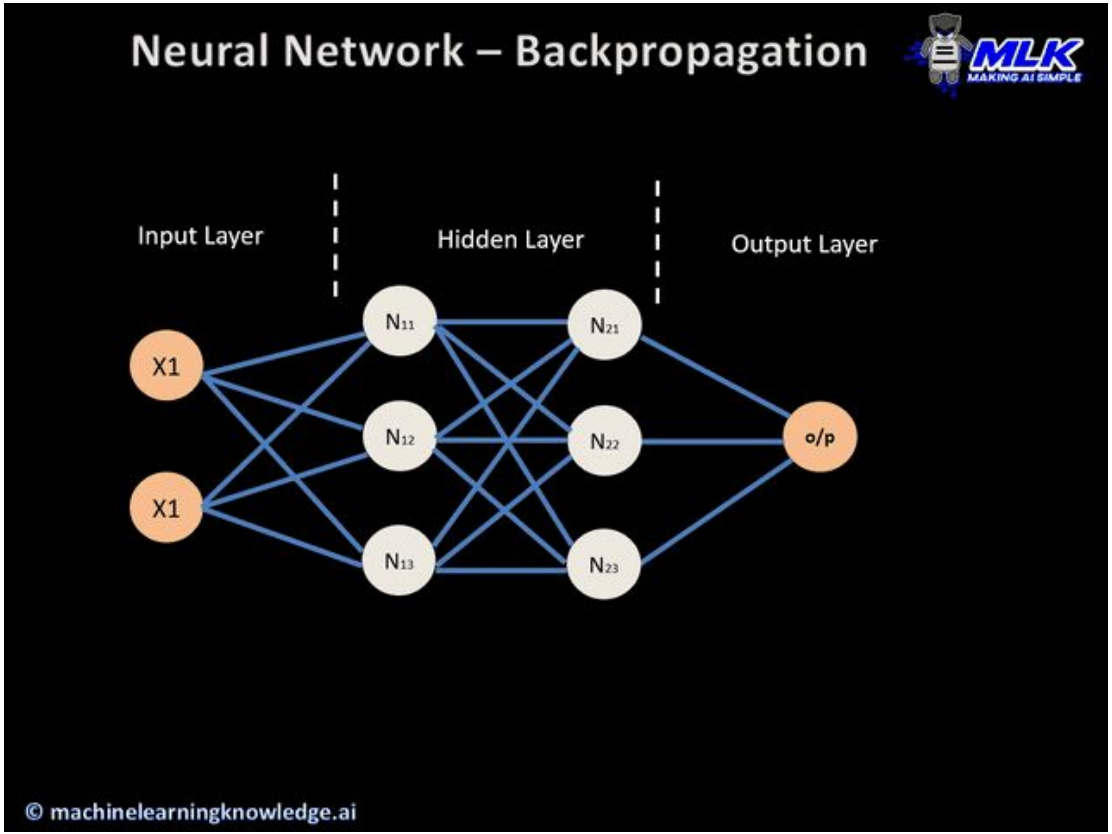


But the whole point of an ANN is that you do not need to assume the optimal functional form of the equation ahead of time - it's probably really complex!

So what do we do now…?

We train the model!

- **Step 1- Forward Pass:** weights/biases are frozen the ANN ingests the input and makes a prediction
- **Step 2 - Calculate Error/Loss:** compute the error in the ANN's prediction
- **Step 3 - Backward Pass:** update the weights via gradient descent and backpropagation to move downgradient
- Rinse and repeat



Backpropagation

- Short for “back propagation of errors”
- A method for determining the gradient of the loss function when you don’t know the functional form
- The method calculates the gradient of the loss function with respect to the neural network's weights.
- **Why are only certain activation functions are allowed - must be differentiable!**

$$J = \frac{1}{N}\sum_{i=1}^{N}(\tilde{y}_i - y_i)^2$$

$$w_{k,new} = w_k - \lambda \frac{\partial \text{Loss}}{\partial w_k}$$

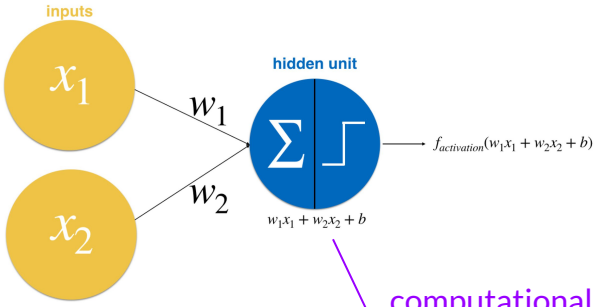
The partial derivative of the PREDICTED y with respect to the weights w.

$$\frac{\partial \text{Loss}}{\partial w_k} = \frac{\partial}{\partial w_k}\left(\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2\right) = \frac{2}{n}\sum_{i=1}^{n}(\tilde{y}_i - y)\frac{\partial \tilde{y}_i}{\partial w_k}$$

What we need to know!

Backpropagation

Relies on **chain rule**

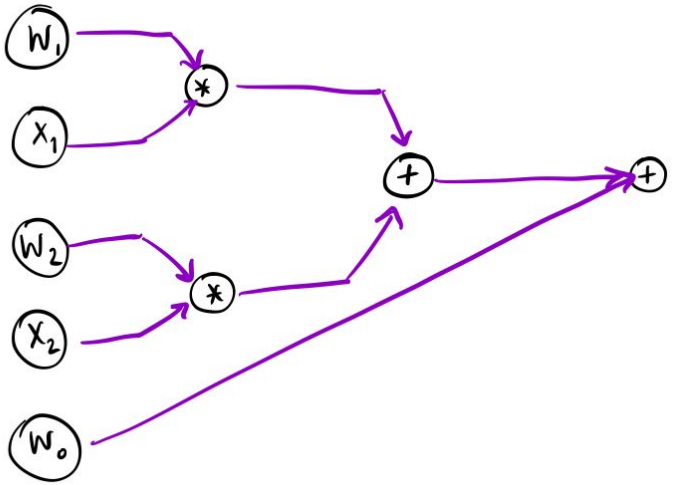


computational graph

$$\frac{\partial}{\partial w}(h(f(g(w)))) = \frac{\partial h}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial w}$$

$$\tilde{y} = h(f(g(w_k x_{input})))$$

$$\Rightarrow \frac{\partial \tilde{y}}{\partial w_k} = \frac{\partial \tilde{y}}{\partial f}\frac{\partial f}{\partial g}\frac{\partial g}{\partial w_k}$$



Backpropagation “by hand”

We encourage practicing backpropagation with pen and paper to help with visualizing the “black box” of ML calculations

[link to example problem with solutions]

want this

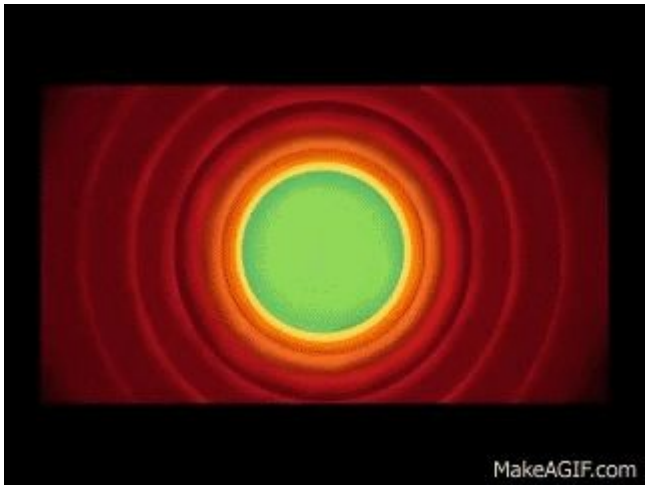
$$w_{k,new} = w_k - \lambda \frac{\partial \text{Loss}}{\partial w_k}$$

$$\frac{\partial \text{Loss}}{\partial w_k} = \frac{\partial}{\partial w_k}\left(\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2\right) = \frac{2}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)\frac{\partial \tilde{y}_i}{\partial w_k}$$

You hopefully never have to do that again!

Hurray for computers!

Great! Now you know how an ANN works!



Well…

Next we will talk about ANN architecture and decisions related to how to design an ANN

Some terminology

WHAT ML CALLS IT

Number of samples: number of individual realizations you have

Batch size: number of samples in a "chunk" of training data that is iterated before updating the weights and biases

Epoch: number of times you train on all batches (number of times you go through your entire training set)

Deep Learning: training an ANN with two or more layers

Lions, and Tigers and Parameters… Oh My!

While this may seem like a lot of parameters, we deal with parameter choices every day…

- How to define the seasonal cycle
- What convective scheme to use
- Width of a moving average

Choices you need to make…

Feed-forward ANNs require choices by the user:

- Architecture (number of nodes and layers)
- Gradient descent algorithm
- Learning Rate
- Activation Function
- Batch size
- Number of epochs
- Choice of regularization and associated parameters
- …

ANN considerations

Train the data the right number of times (epochs).
If we iterate too few times, our model will not have time to find the optimal fit. If we iterate too many times, we will overfit the training data.

Choosing the right number of layers and nodes for the given job is crucial.
Fewer layers/nodes allow for easier interpretability. More layers/nodes allow for more nonlinearity.

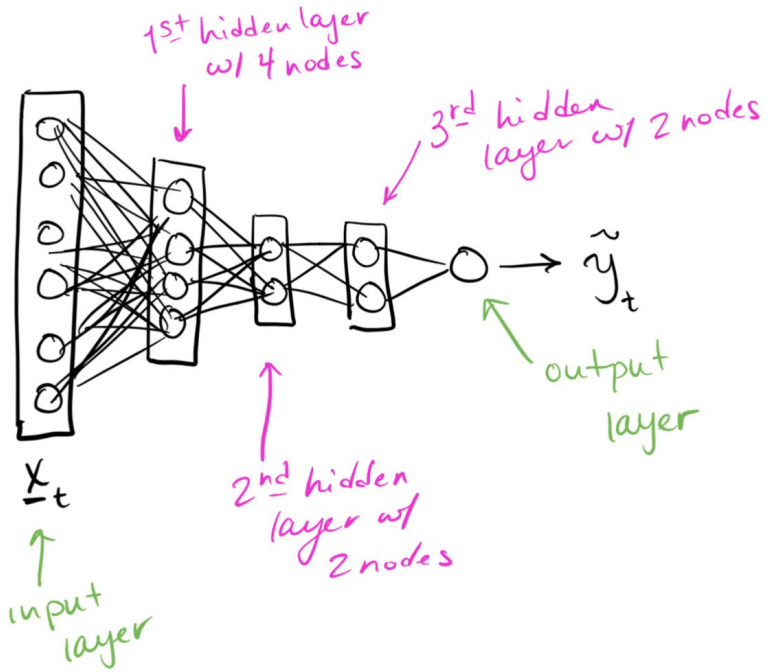
Choose an activation function most appropriate for your solution.
ReLu is a popular option.

Network Architectures

The way we put the ingredients, or pieces, together is called our network "architecture".

- Input units/nodes/neurons
- Output units/nodes/neurons
- Weights
- Bias
- Activation function
- Layers

Regression Problem

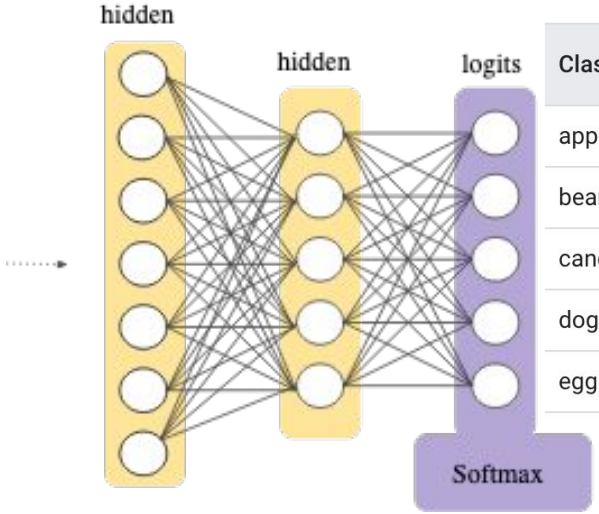


Network Architectures

The way we put the ingredients, or pieces, together is called our network “architecture”.

- Input units/nodes/neurons
- Output units/nodes/neurons
- Weights
- Bias
- Activation function
- Layers

Classification Problem

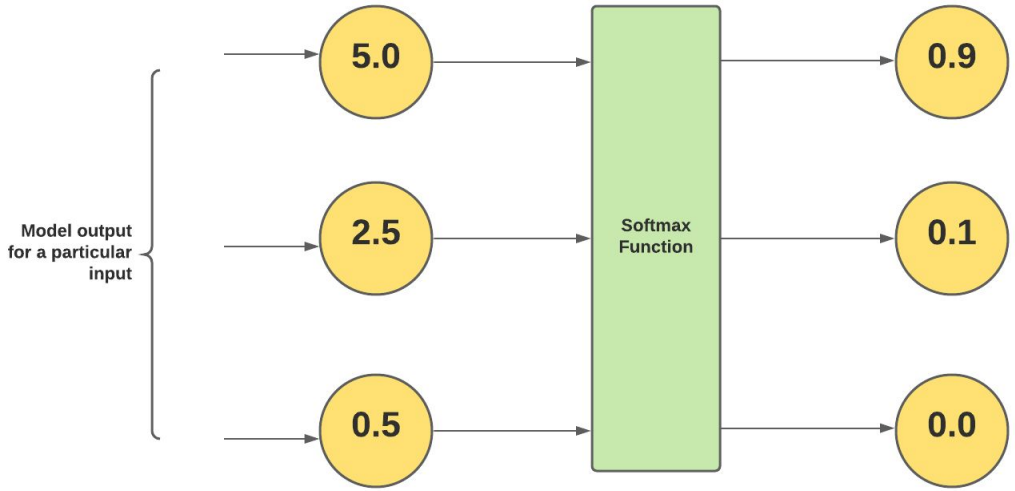


Class	Probability
apple	0.001
bear	0.04
candy	0.008
dog	0.95
egg	0.001

https://developers.google.com/machine-learning/crash-course/multi-class-neural-networks/softmax

Softmax: Let's make the output a probability!

- Converts "raw" output to probabilities for classification tasks
- After the softmax, output values are between 0 and 1
- The class with the highest probability gives your answer



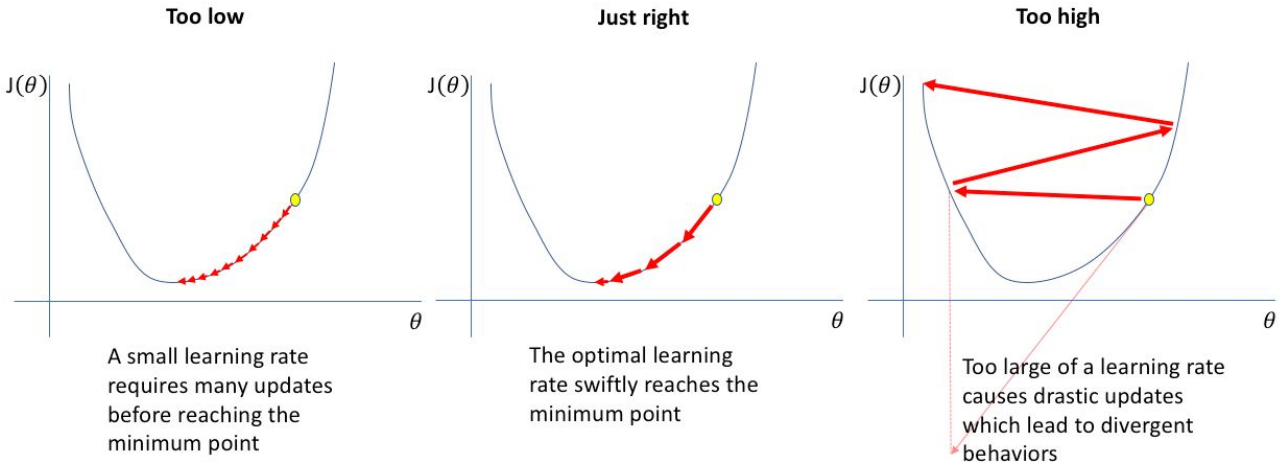
https://vitalflux.com/what-softmax-function-why-needed-machine-learning/

Gradient Descent in ML, Learning rate

Identify the right learning rate.

A learning rate too large may cause the solution to skip between local minima.

A learning rate too small may get stuck in one local minimum, and also may take a longer to learn.



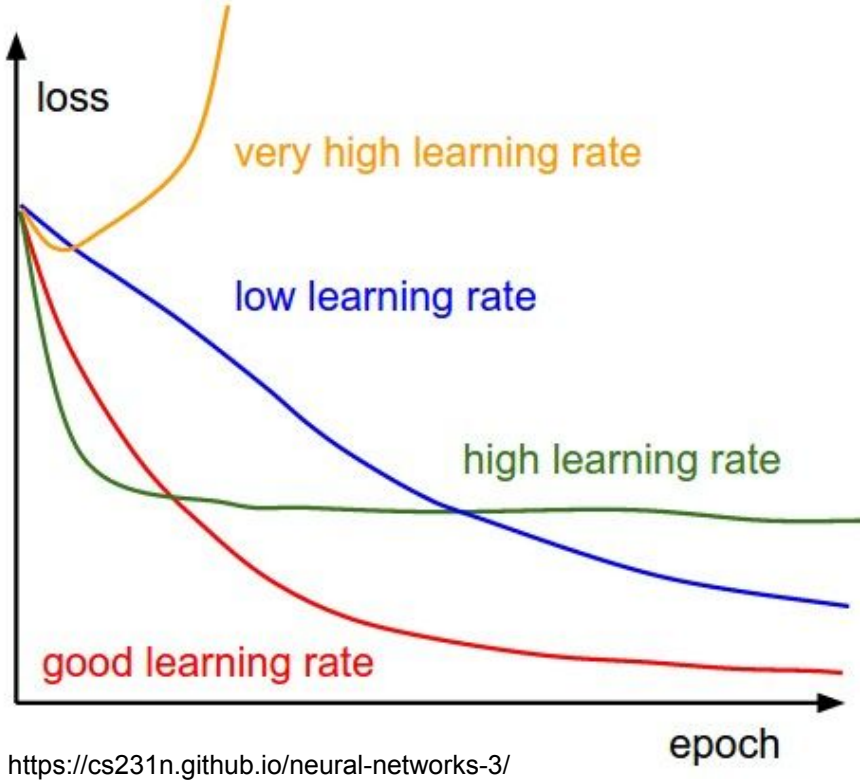
https://www.jeremyjordan.me/nn-learning-rate/

Gradient Descent in ML, Learning rate

Identify the right learning rate.

A learning rate too large may cause the solution to skip between local minima.

A learning rate too small may get stuck in one local minimum, and also may take a longer to learn.



https://cs231n.github.io/neural-networks-3/

Gradient Descent in ML, choices

- Stochastic Gradient Descent (SGD): determine gradient estimate from a small **mini-batch** of the data to speed up computation
- Many choices of exactly how to do this exist
- This is something the user much choose (a “parameter” of the model)

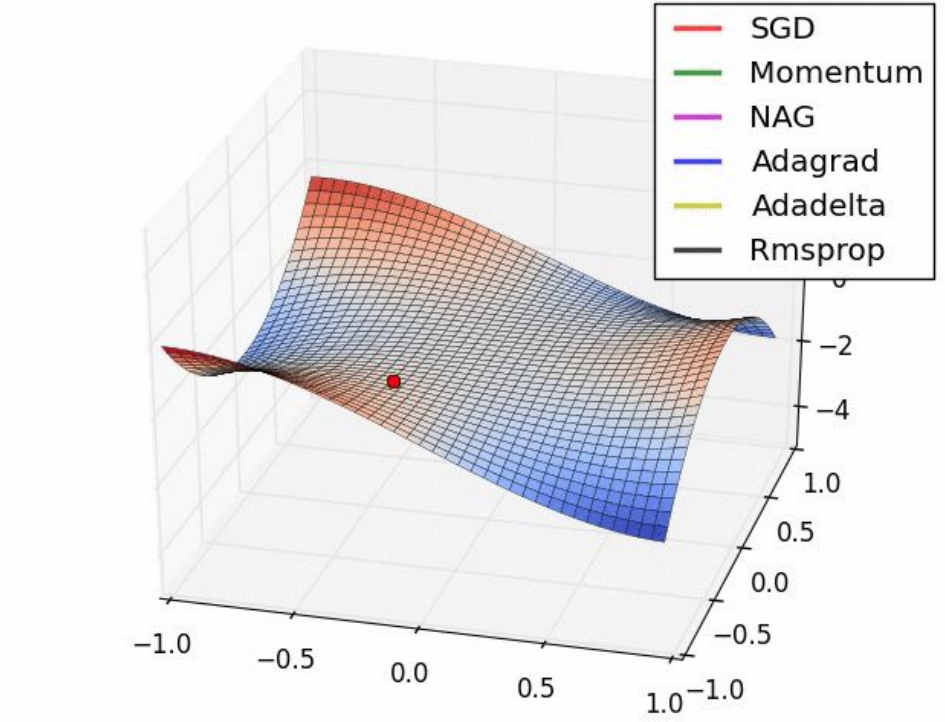
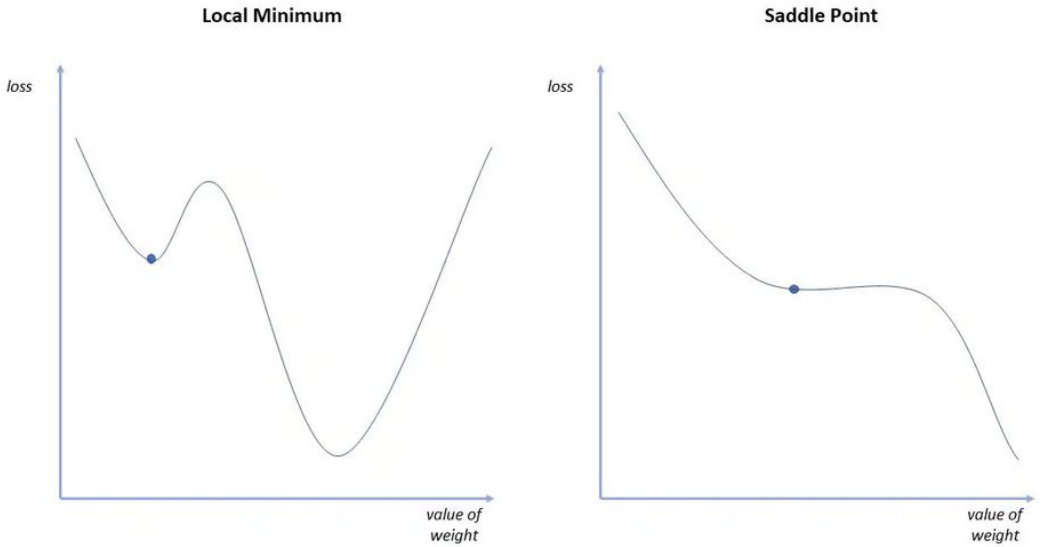


Figure by Alec Radford

Gradient Descent in ML, choices

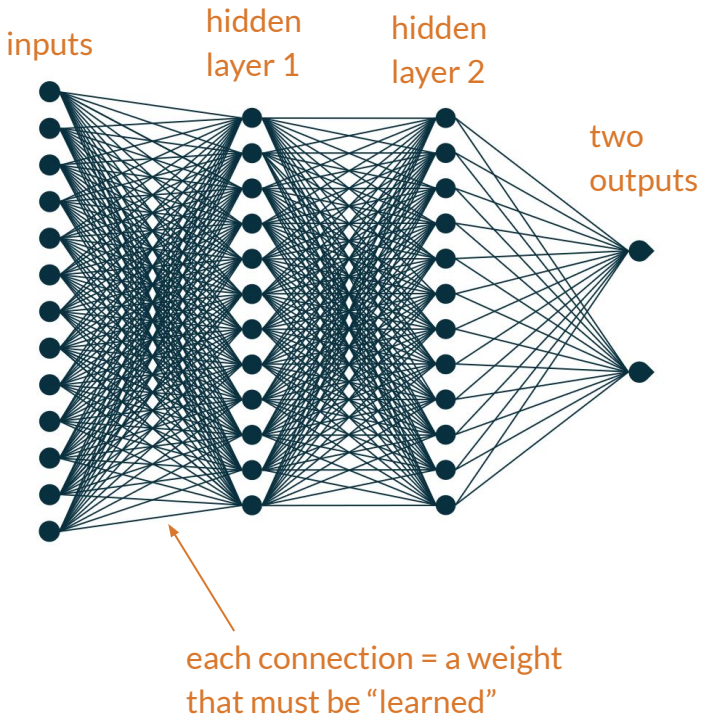
- Utilizing Stochastic Gradient Descent will help in getting close to global minimum instead of getting stuck at a local one
- Also faster than Batch Gradient Descent



https://www.ibm.com/cloud/learn/gradient-descent

Let's Talk About Overfitting...

An ANN often has thousands (or millions) of weights to adjust



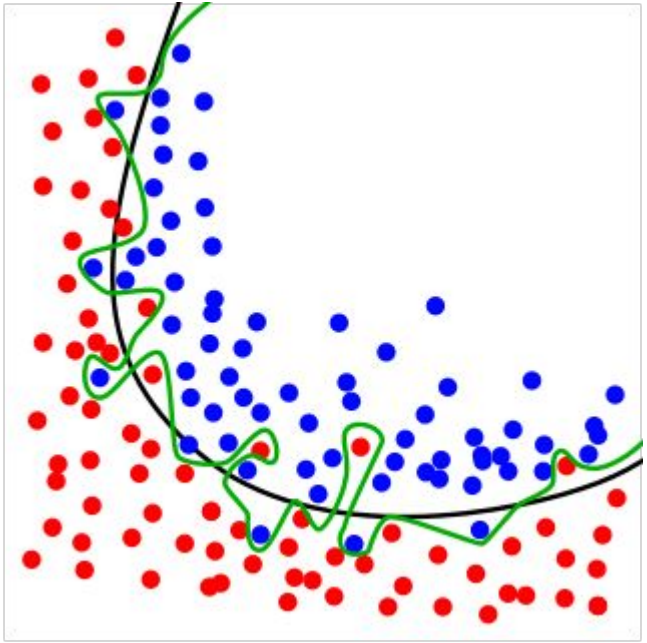
Let's Talk About Overfitting...

An ANN often has thousands (or millions) of weights to adjust

This often leads to egregious overfitting

Two common remedies:

- L1 & L2 Regularization
- Drop-out



The black line fits the data well, whereas the green one is overfitting.

https://elitedatascience.com/overfitting-in-machine-learning

Regularization: L2 aka Ridge Regression

A regression tool to help prevent overfitting.

Adds a term to the loss/error function that penalizes the weights if they are too large.

Keeps the weights small.

Ridge Regression ("L2 Regularization")

$$\text{Loss} = \text{RMSE} + \lambda \sum_{i=1}^{p} w_i^2$$

Ridge Parameter
How much to penalize large weights

Shrinkage Term
Forces the individual coefficients to be small

>> sharing of weight across coefficients

$$w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 = \tilde{y}$$

will force coefficients to share importance

Regularization: L1 aka LASSO Regression

A regression tool to help prevent overfitting.

Adds a term to the loss/error function that penalizes the weights if they are too large.

Keeps the weights small.

LASSO Regression ("L1 Regularization")
(least absolute shrinkage and selection operator)

$$\text{Loss} = \text{RMSE} + \lambda \sum_{i=1}^{p} |w_i|$$

LASSO Parameter
How much to penalize large weights

Shrinkage Term
Punishes high values but actually sets them to exactly zero if not important

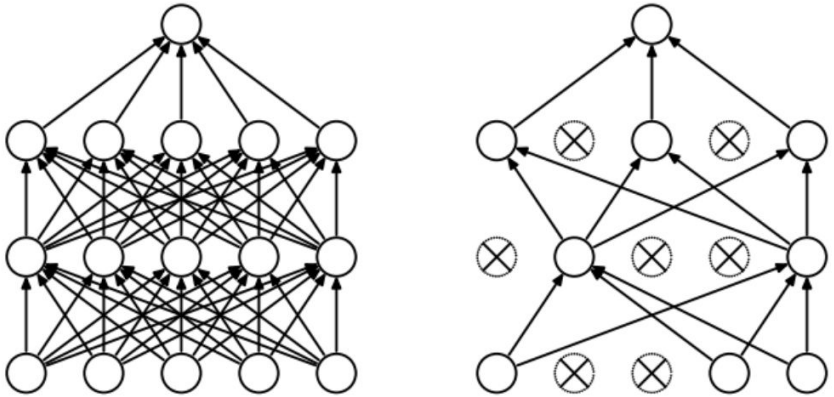
>> helpful for determining which features are most important

$$\cancel{w_1}^{0} x_1 + \cancel{w_2}^{0} x_2 + w_3 x_3 + w_4 x_4 + \cancel{w_5}^{0} x_5 + w_6 x_6 + \cancel{w_7}^{0} x_7 + \cancel{w_8}^{0} x_8 + \cancel{w_9}^{0} x_9 = \tilde{y}$$

will force some coefficients to zero

Dropout

During training (and training only!), randomly remove or "drop out" nodes, requiring that the ANN learn to still make accurate predictions in spite of losing these nodes. Dropout has the effect of forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.



(a) Standard Neural Net (b) After applying dropout.

- *requires the user chose the percent to drop-out (often ~50-80%)*
- *requires more epochs during training*

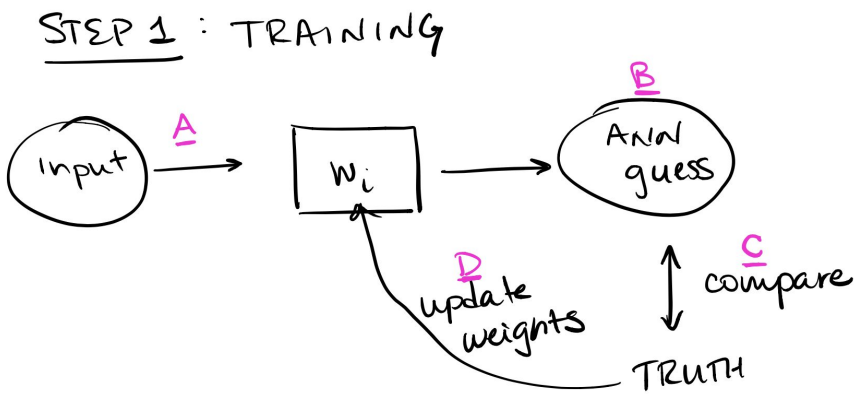
At last, testing

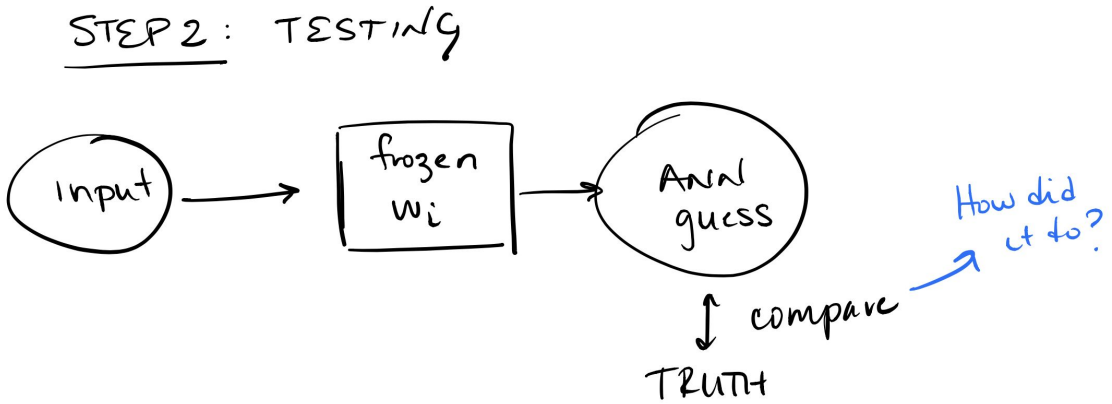
Training an ANN is all about **updating the weights and biases**.

This is done **iteratively** by comparing to the "truth".

Once this is done and you are happy with the performance of your model, it's time for testing!

Testing then involves **freezing** the weights, and using the ANN as a predictor function.





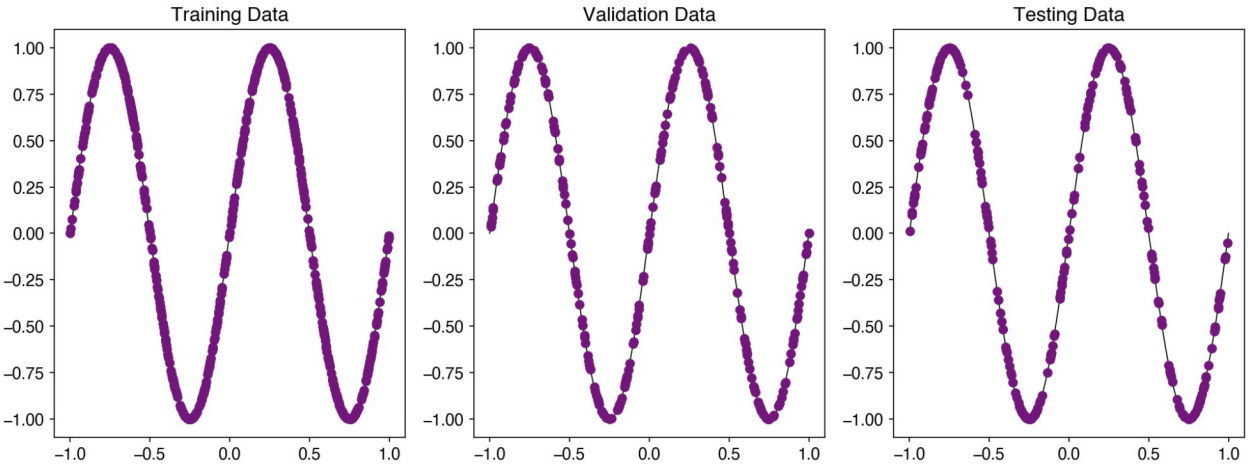
ANN Coding Examples

Two examples

Predicting a Sine Wave with an ANN

We are going to train an ANN to predict a sine wave

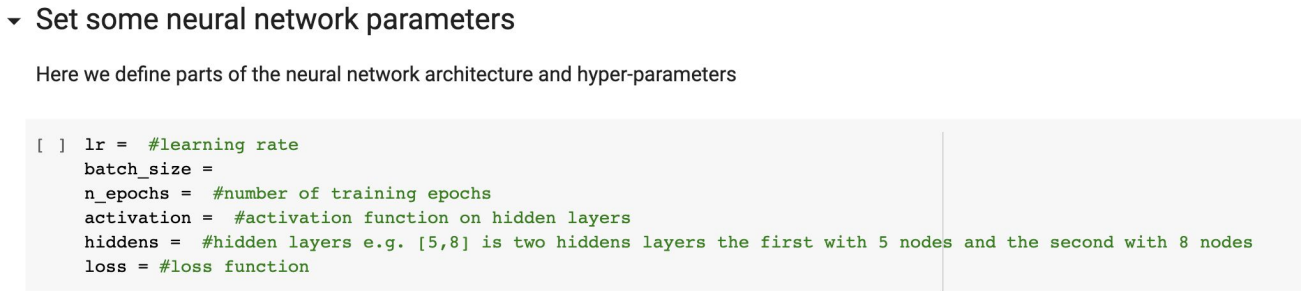
- Input *x* between -1 and 1
- Output sin(*x*)



The function we are predicting is the solid line, and the points we use indicated in purple

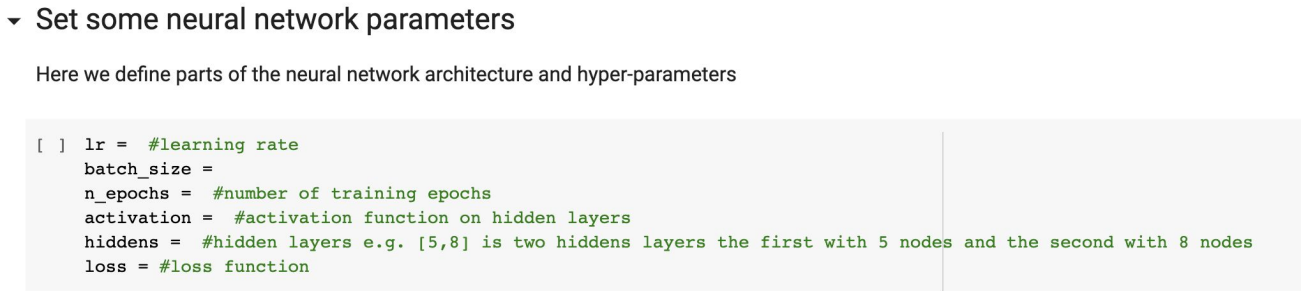
Predicting a Sine Wave with an ANN

Now open the code in google colab, and find the cell titled *Set some neural network parameters*



Predicting a Sine Wave with an ANN

Now open the code in google colab, and find the cell titled *Set some neural network parameters*

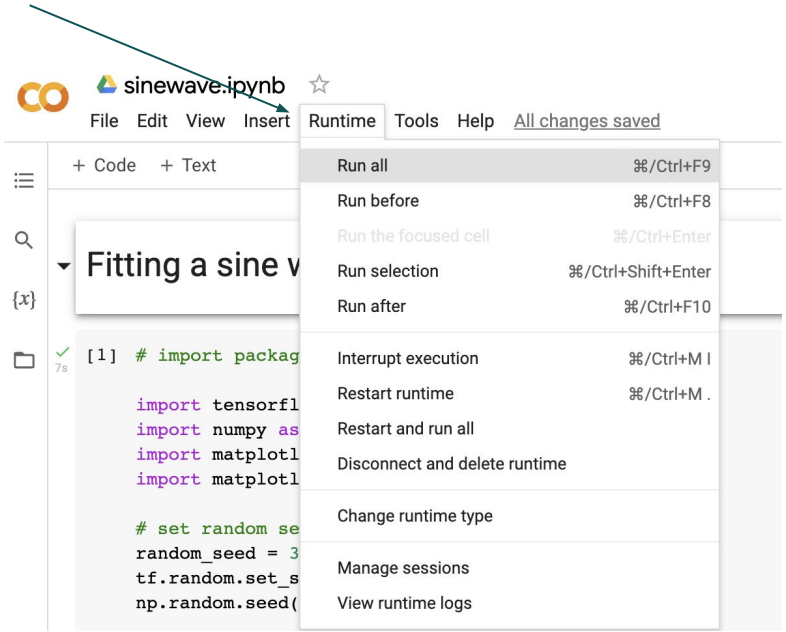


This is where we set our architecture and hyper-parameters. To start with, enter the following values

```
lr = 0.01
batch_size = 32
n_epochs = 400
activation = ‘sigmoid’
hiddens = [10,100]
loss = ‘mae’
```

Predicting a Sine Wave with an ANN

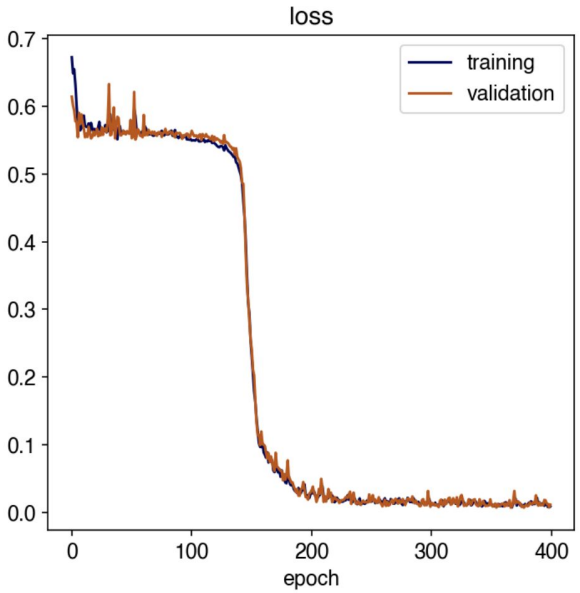
Run the code!

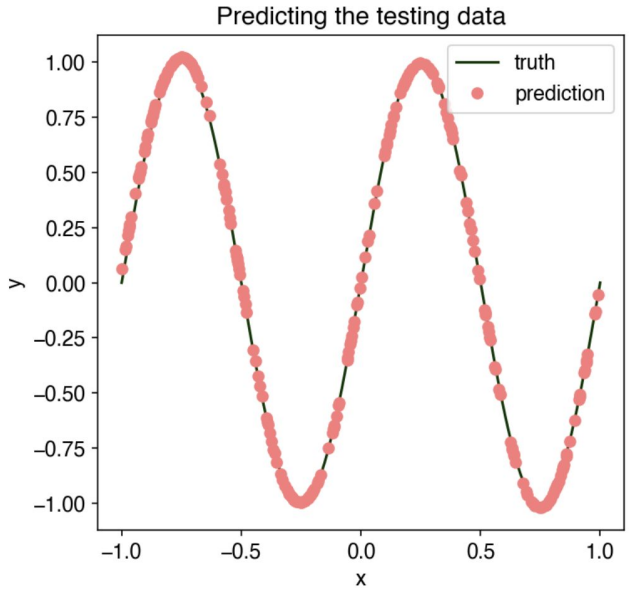


Predicting a Sine Wave with an ANN

How does the model do?

- On the left we plot the model loss during training, i.e. the loss at the end of each epoch

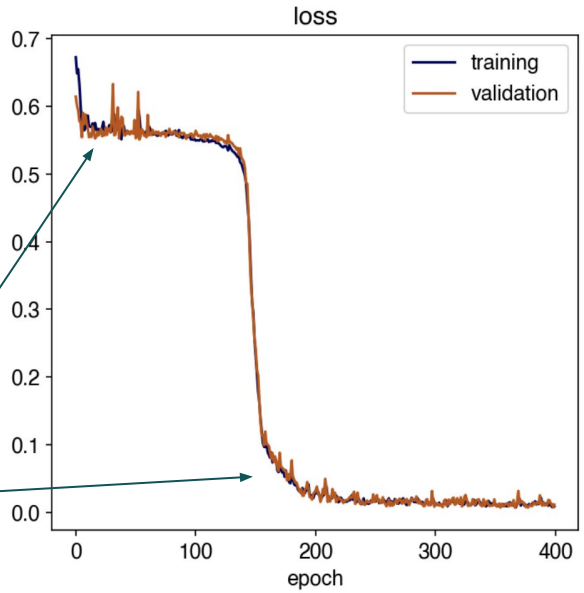


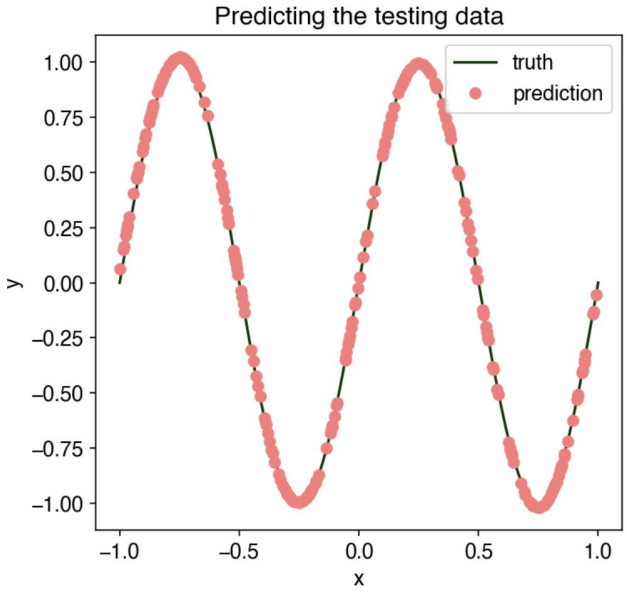


Predicting a Sine Wave with an ANN

How does the model do?

- On the left we plot the model loss during training, i.e. the loss for at the end of each epoch
- Model finds a local minimum
- Model finds a better local minimum

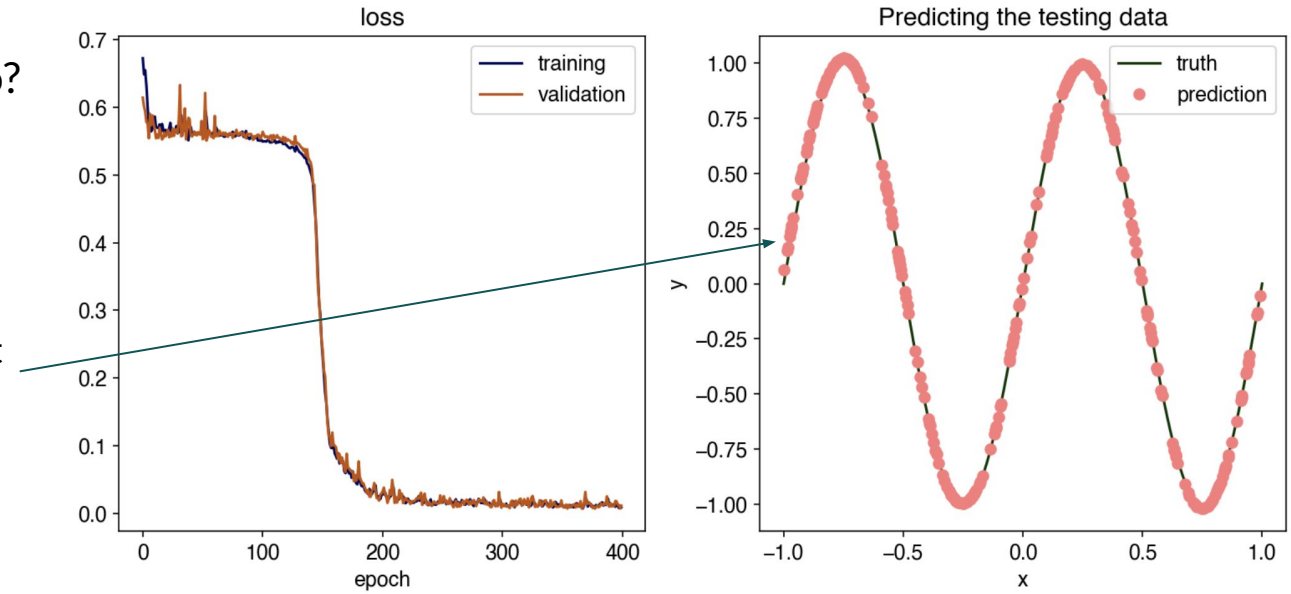




Predicting a Sine Wave with an ANN

How does the model do?

- On the right we plot the network's prediction of the testing data
- Network is able to fit this unseen data



Predicting a Sine Wave with an ANN

Play with the network parameters

▾ Set some neural network parameters

Here we define parts of the neural network architecture and hyper-parameters

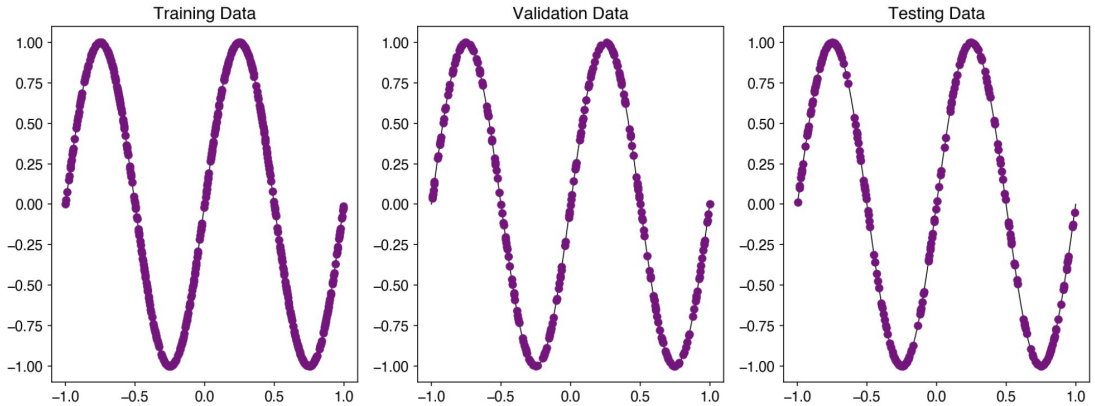
```
[ ]  lr = 0.01 #learning rate
     batch_size = 32
     n_epochs = 400 #number of training epochs
     activation = 'sigmoid' #activation function on hidden layers
     hiddens = [10,100] #hidden layers e.g. [5,8] is two hiddens layers the first with 5 nodes and the second with 8 nodes
     loss = 'mae'
```

- Navigate to the cell titled *Set some neural network parameters*
- You can adjust anything in this cell and then run the code to see how it affects the network
- Suggestions:
  - Change activation to ‘relu’
  - Increase batch size up to 256
  - Decrease learning rate to 0.001 (or even smaller)

Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

- In the previous example, we randomly grabbed the training, validation and testing sets from the data

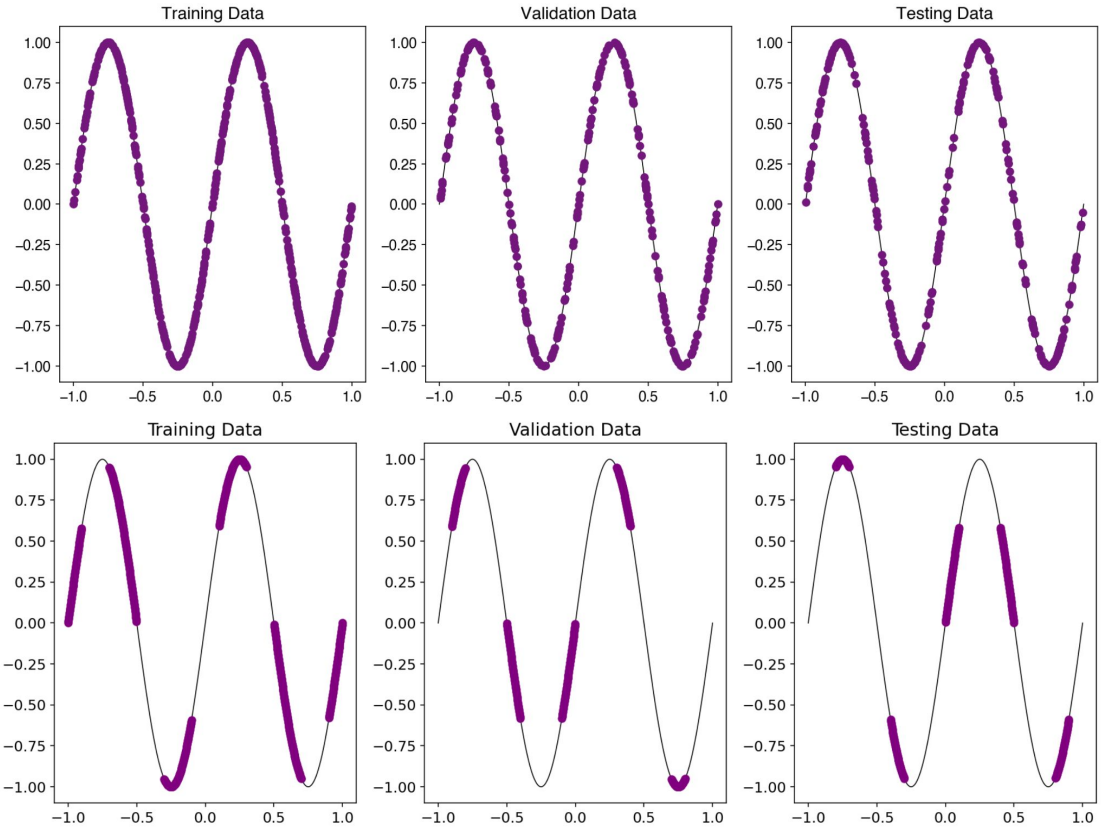


Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

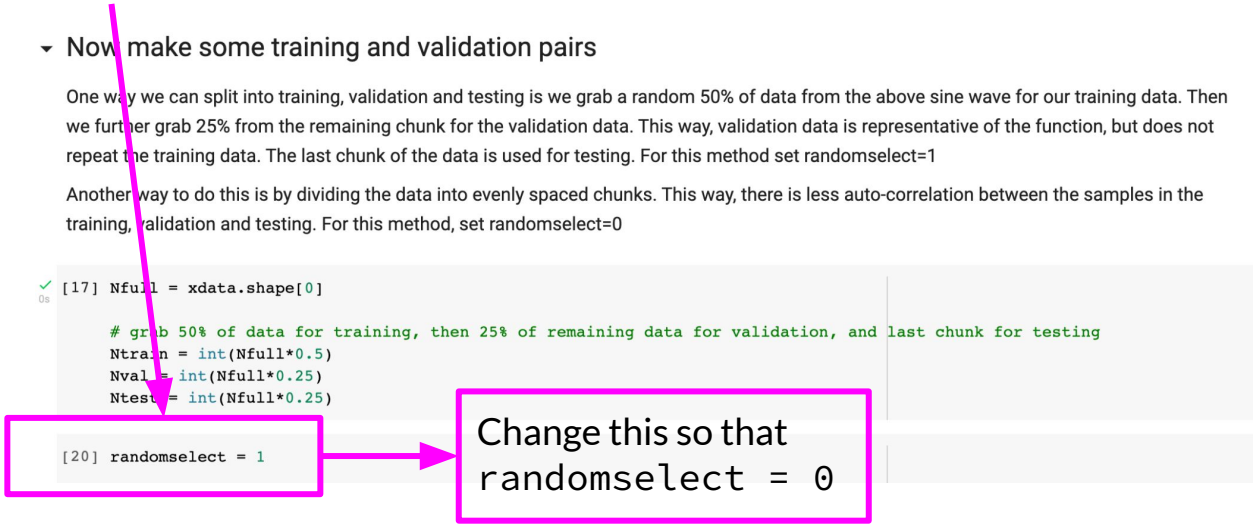
- In the previous example, we randomly grabbed the training, validation and testing sets from the data

- What happens if we change this so the data is picked in chunks?



Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing



Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

- Now run the notebook again with the original parameters

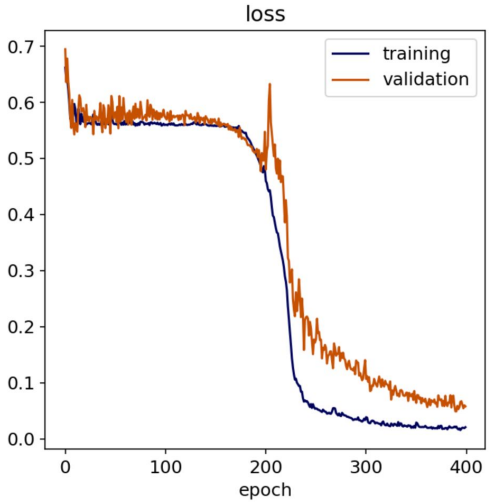
```
lr = 0.01
batch_size = 36
n_epochs = 400
activation = ‘sigmoid’
hiddens = [10,100]
loss = ‘mae’
```

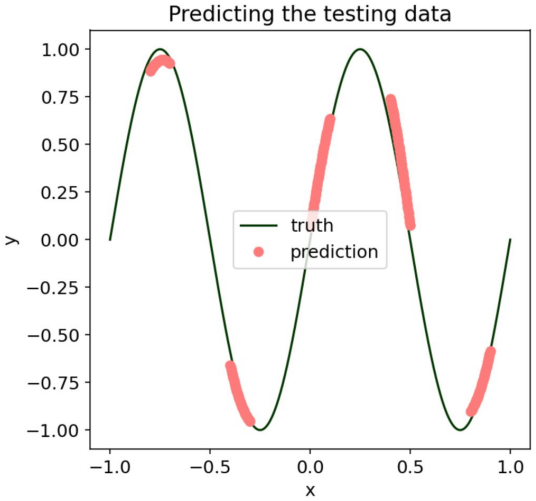
Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

- Now run the notebook again with the original parameters

```
lr = 0.01
batch_size = 36
n_epochs = 400
activation = ‘sigmoid’
hiddens = [10,100]
loss = ‘mae’
```



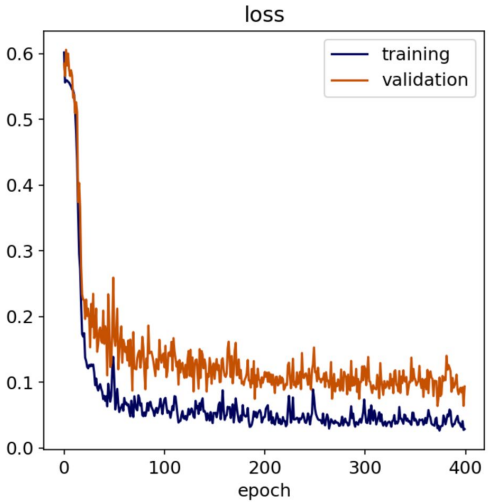


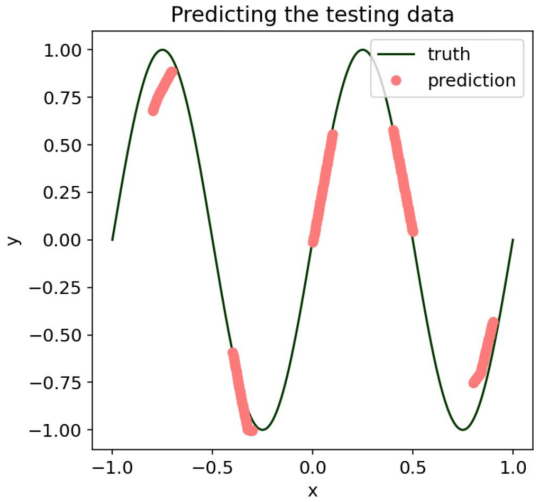
Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

- Now run the notebook again with the original parameters
- Change activation to ‘relu’

```
lr = 0.01
batch_size = 36
n_epochs = 400
activation = ‘relu’
hiddens = [10,100]
loss = ‘mae’
```



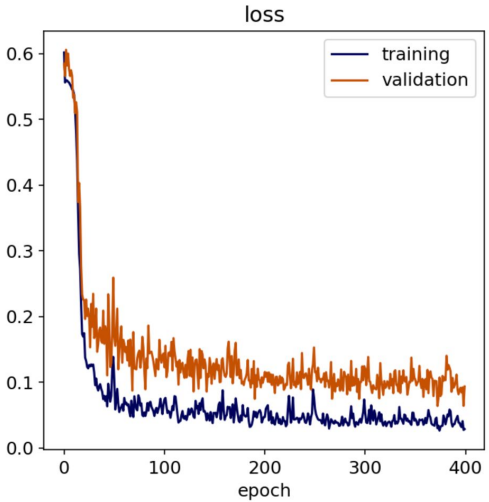


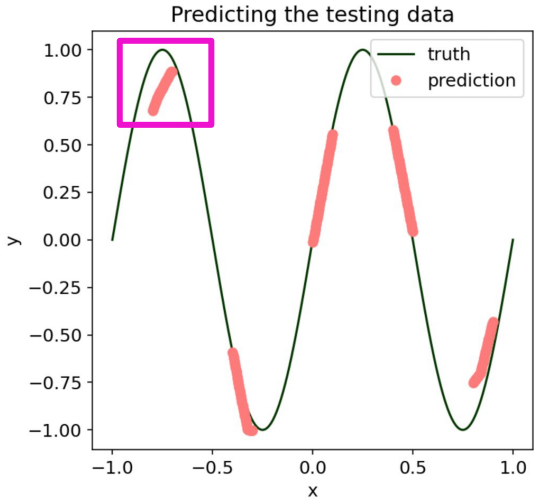
Predicting a Sine Wave with an ANN

A fun example of the effect of auto-correlation between training validation and testing

- Now run the notebook again with the original parameters
- Change activation to ‘relu’

```
lr = 0.01
batch_size = 36
n_epochs = 400
activation = ‘relu’
hiddens = [10,100]
loss = ‘mae’
```





Predicting ENSO with an ANN

Another example we know the answer to.

Defining ENSO…

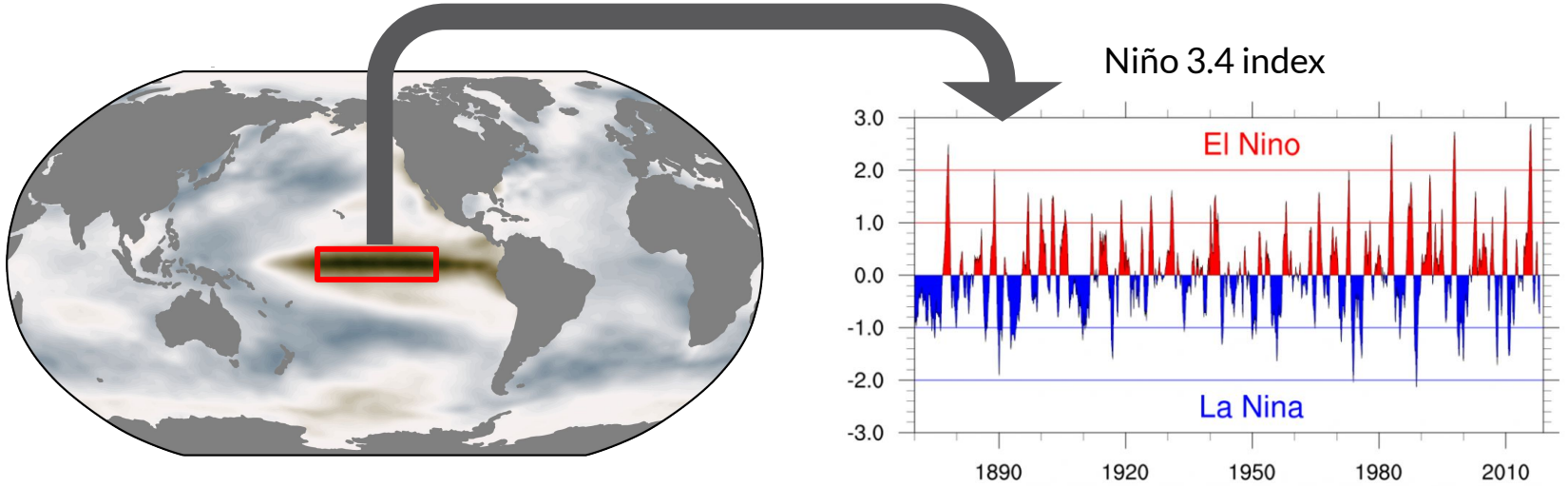
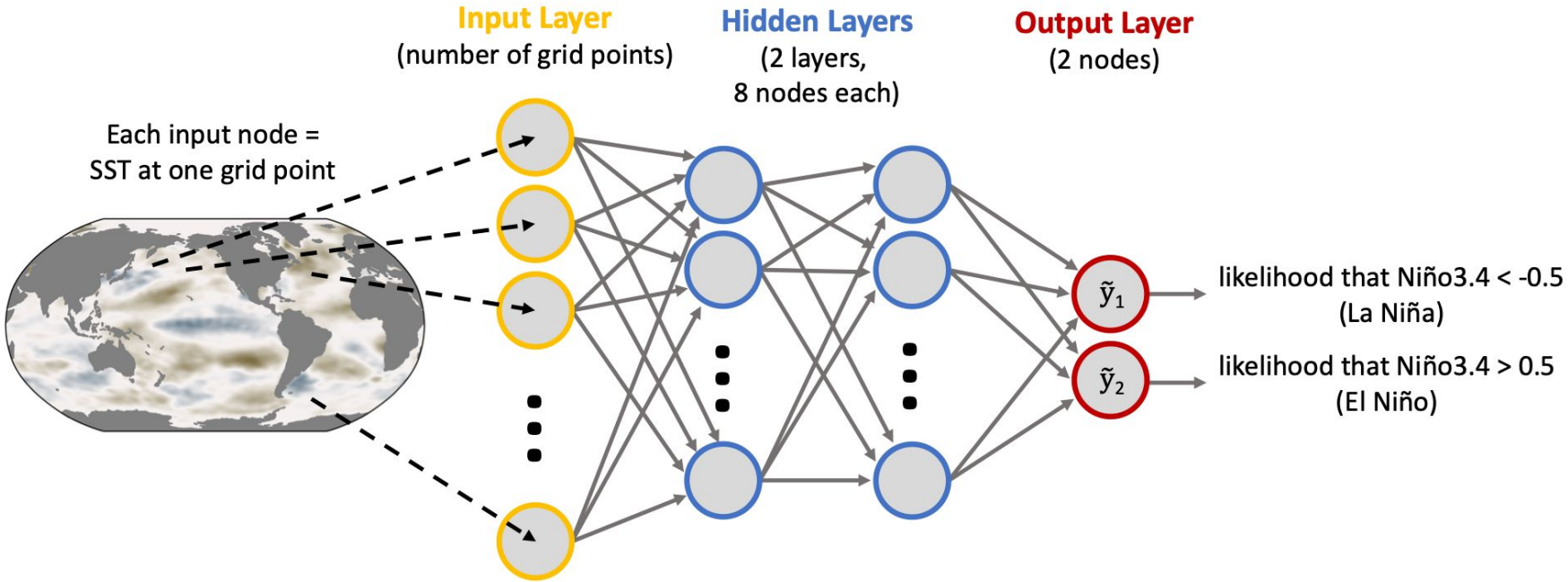


Figure courtesy of NCAR Climate Data Guide

ENSO is commonly defined according to average sea-surface temperatures within the central tropical Pacific.

Toms et al., *JAMES* (2020)

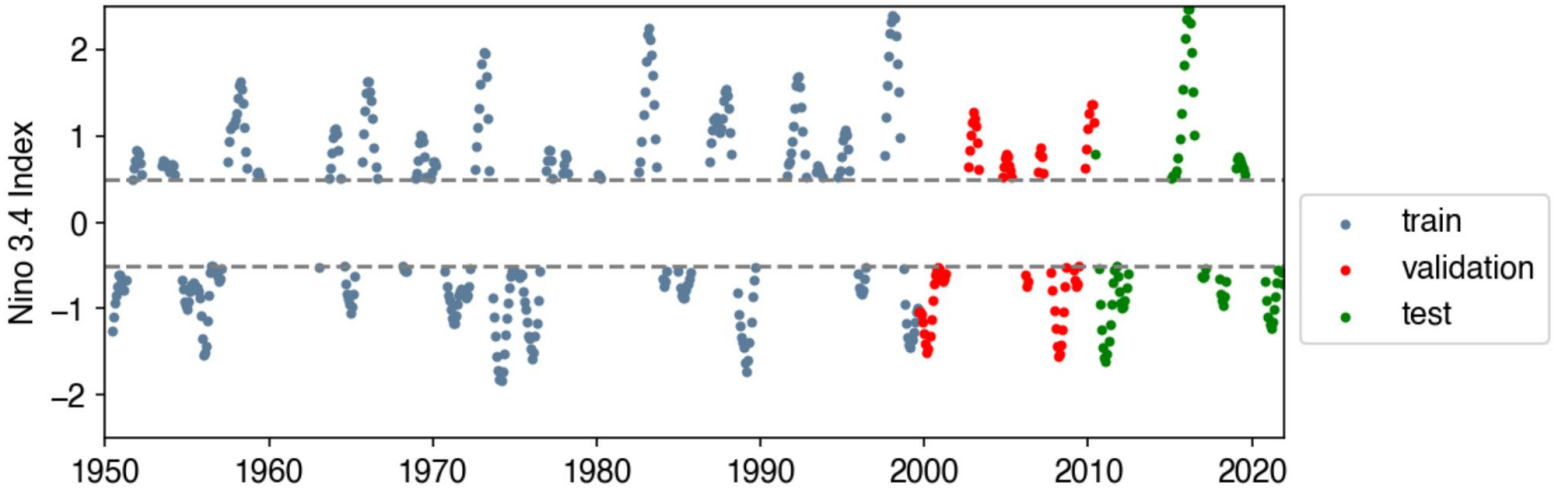
ENSO + Neural Networks



Toms et al., *JAMES* (2020)

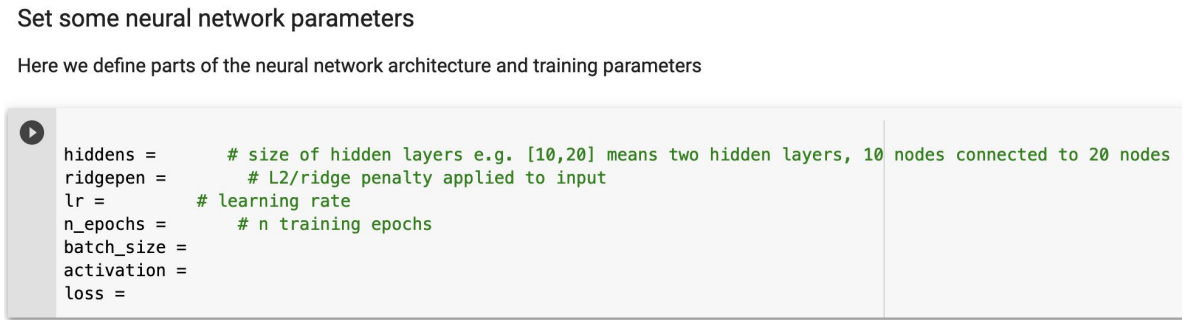
Predicting ENSO with an ANN

Include only samples where nino event is occuring (i.e. nino3.4>0.5 or nino3.4<-0.5)
Split into train, validation, and test sets by date:



Predicting ENSO with an ANN

Now open the code in google colab, and find the cell titled *Set some neural network parameters*



To start with, use the following values:

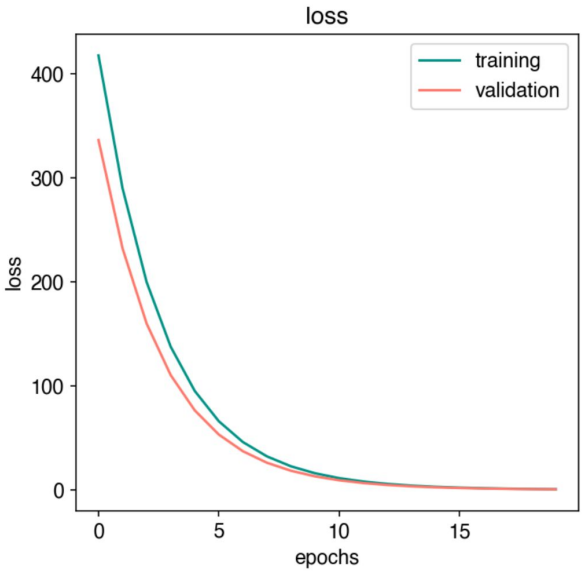
```
hiddens = [12, 12]
ridgepen = 1
lr = 1e-3
n_epochs = 20
batch_size = 32
activation = ‘relu’
loss = ‘categorical_crossentropy’
```

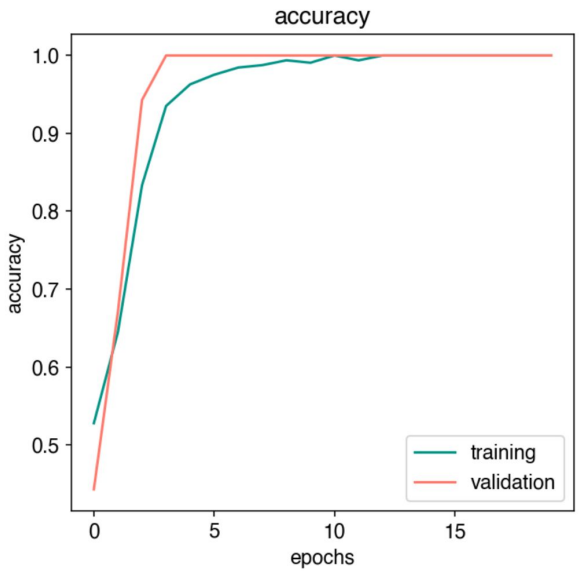
Once, you have set the parameters, run the code!

Predicting ENSO with an ANN

How does the model do?

- Left: loss during training
  - *decreases over time - good!*
- Right: accuracy during training
  - *similar accuracy for training and validation data - good!*





Predicting ENSO with an ANN

How does the model do on test data?

```
nino_pred = model.predict(sst_test) # make predictions for test data

nino_pred = np.argmax(nino_pred,axis=1)
nino_true = np.argmax(nino_test,axis=1)

modelcorr = nino_pred==nino_true
nmodelcorr = modelcorr[modelcorr].shape[0]
ntest = nino_true.shape[0]
print('Model accuracy on testing is %f%%' %(100*nmodelcorr/ntest))
```

```
Model accuracy on testing is 95.714286%
```

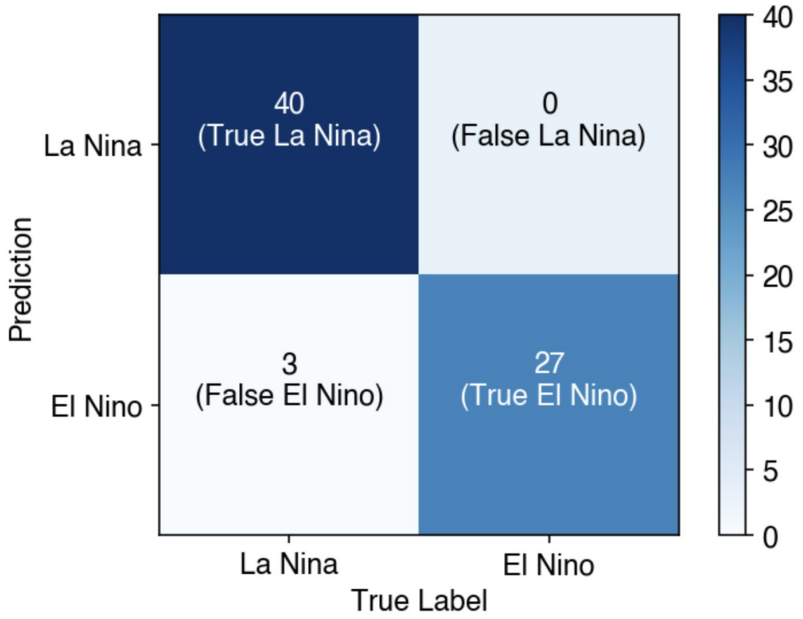
Predicting ENSO with an ANN

How does the model do on test data?

```
nino_pred = model.predict(sst_test) # make predictions for test data

nino_pred = np.argmax(nino_pred,axis=1)
nino_true = np.argmax(nino_test,axis=1)

modelcorr = nino_pred==nino_true
nmodelcorr = modelcorr[modelcorr].shape[0]
ntest = nino_true.shape[0]
print('Model accuracy on testing is %f%%' %(100*nmodelcorr/ntest))
```

```
Model accuracy on testing is 95.714286%
```

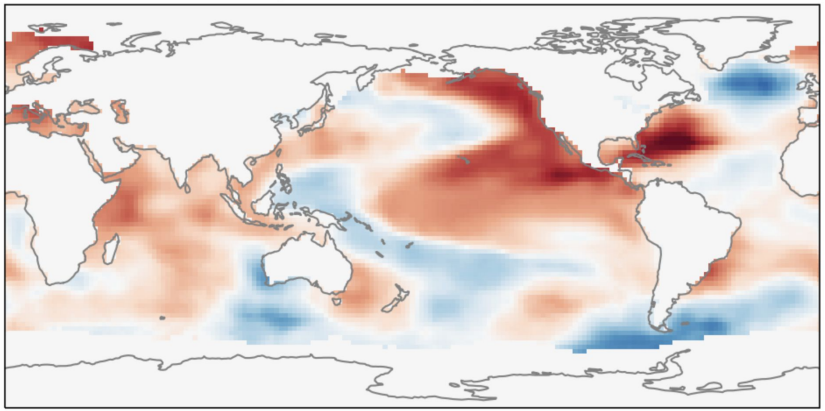
Confusion matrix:



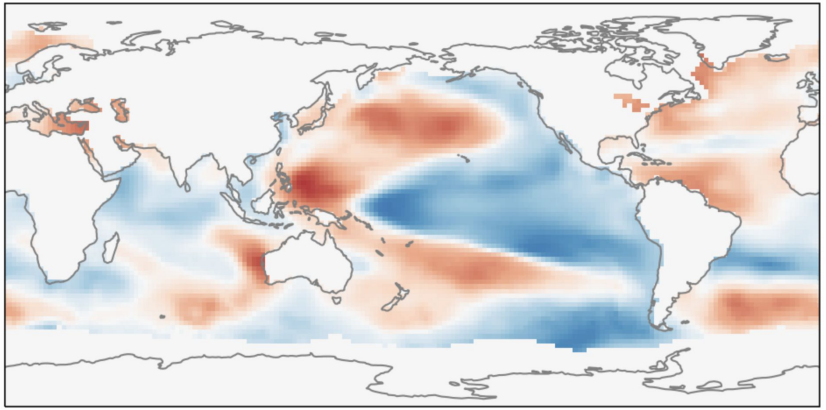
Predicting ENSO with an ANN

How does the model do on test data?

Composite of El Nino predictions



Composite of La Nina predictions



Predicting ENSO with an ANN

Now go back to the neural network parameters.
Re-train the network using different parameters.

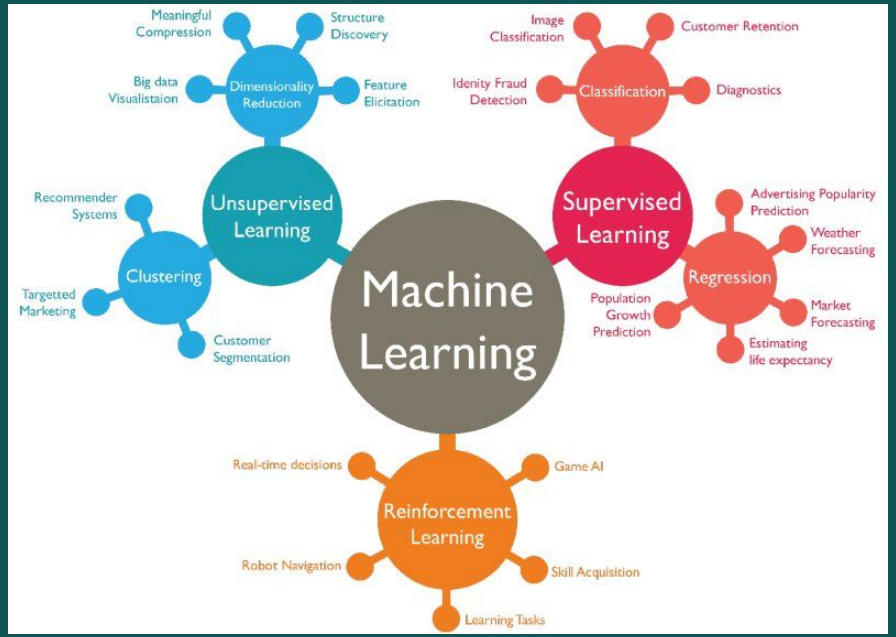
Set some neural network parameters

Here we define parts of the neural network architecture and training parameters

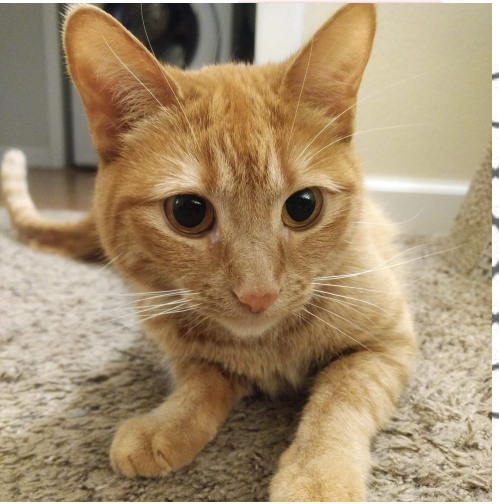
```
hiddens =       # size of hidden layers e.g. [10,20] means two hidden layers, 10 nodes connected to 20 nodes
ridgepen =        # L2/ridge penalty applied to input
lr =          # learning rate
n_epochs =       # n training epochs
batch_size =
activation =
loss =
```

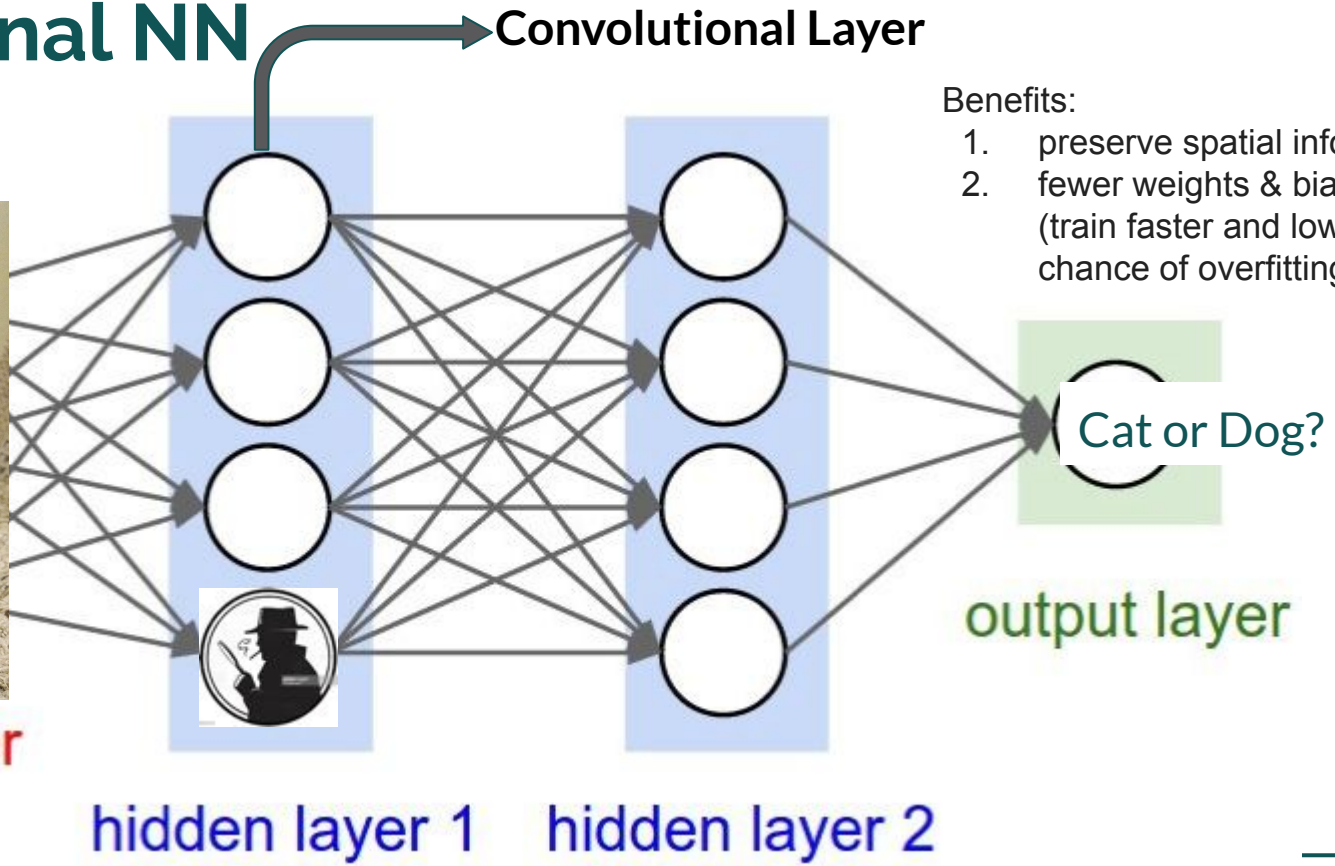
How do the predictions change?

Advanced ANN Techniques

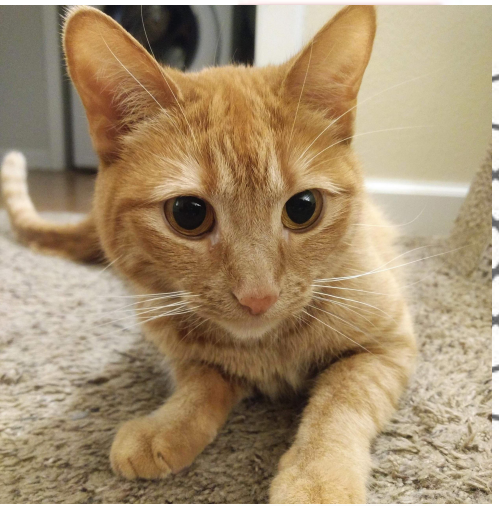


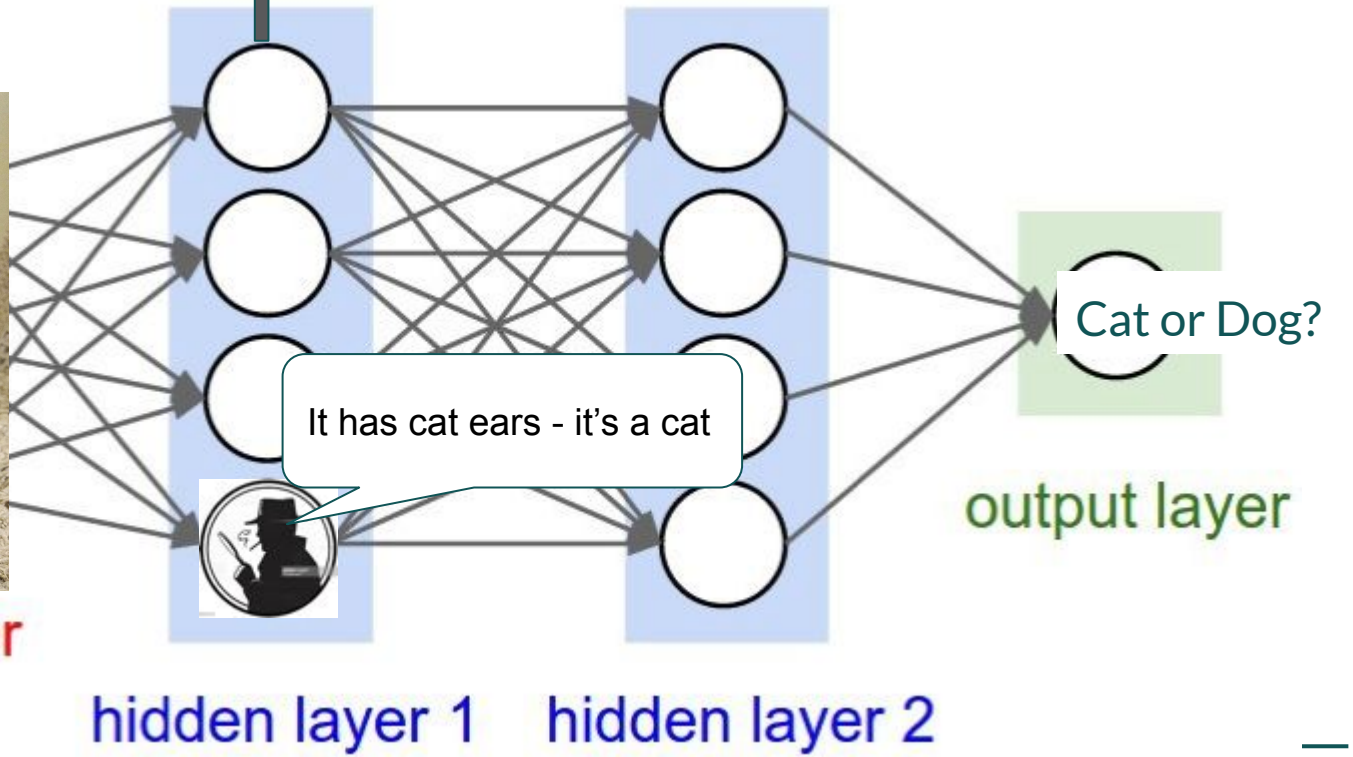
Convolutional NN





Convolutional NN





Convolutional NN

Used for **image** detection

- Neurons/Nodes are now 2D matrices i.e. FILTERS
- These FILTERS detect patterns in the input image

Initial Training:

•Edges, Corners, Shapes (circles, squares)

More Training:

•Objects (Ears, Eyes)

Even More Training:

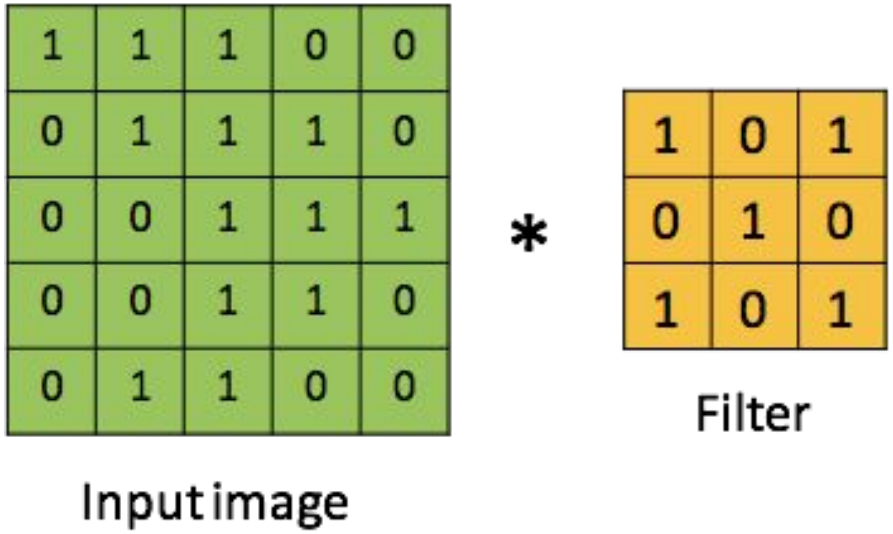
•Complex objects (cats, dogs)

Filters

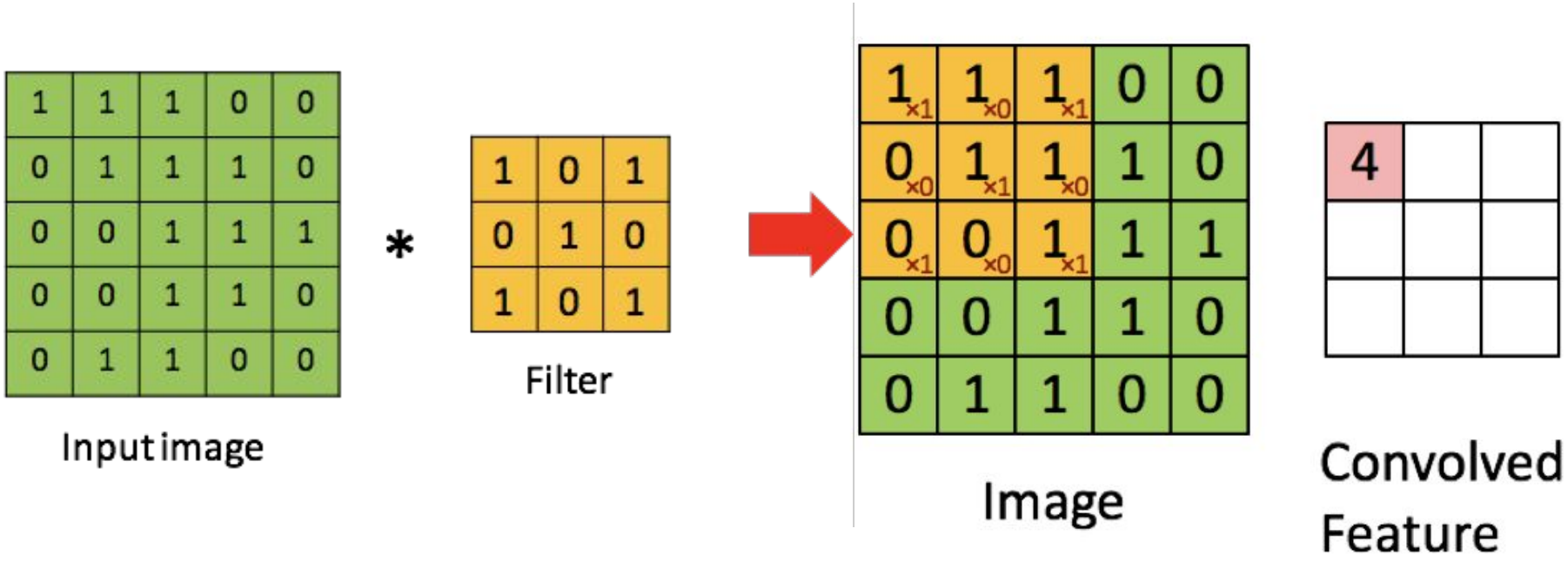
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input image

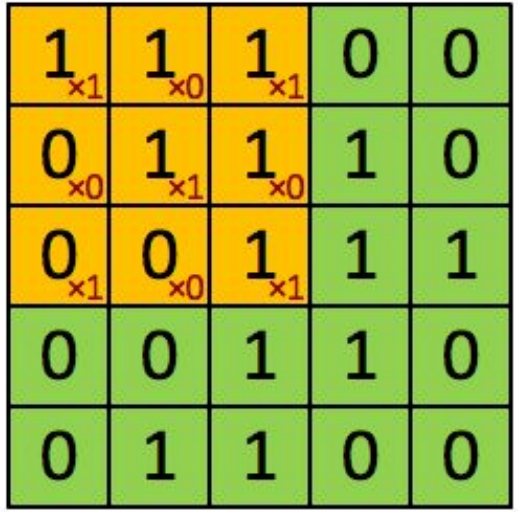
Filters


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
*
1 0 1
0 1 0
1 0 1
Filter
Input image

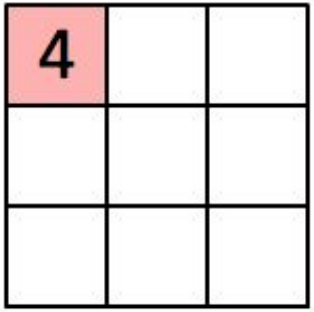
Convolving


1 1 1 0 0
0 1 1 1 0
0 0 1 1 1
0 0 1 1 0
0 1 1 0 0
Input image
*
1 0 1
0 1 0
1 0 1
Filter
1x1 1x0 1x1 0 0
0x0 1x1 1x0 1 0
0x1 0x0 1x1 1 1
0 0 1 1 0
0 1 1 0 0
Image
4
Convolved Feature

Convolving



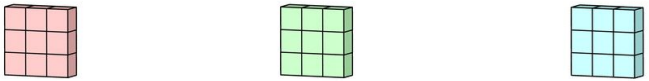
Image



Convolved Feature

CNN layer for a single multi-channel kernel

Perform convolution across channels

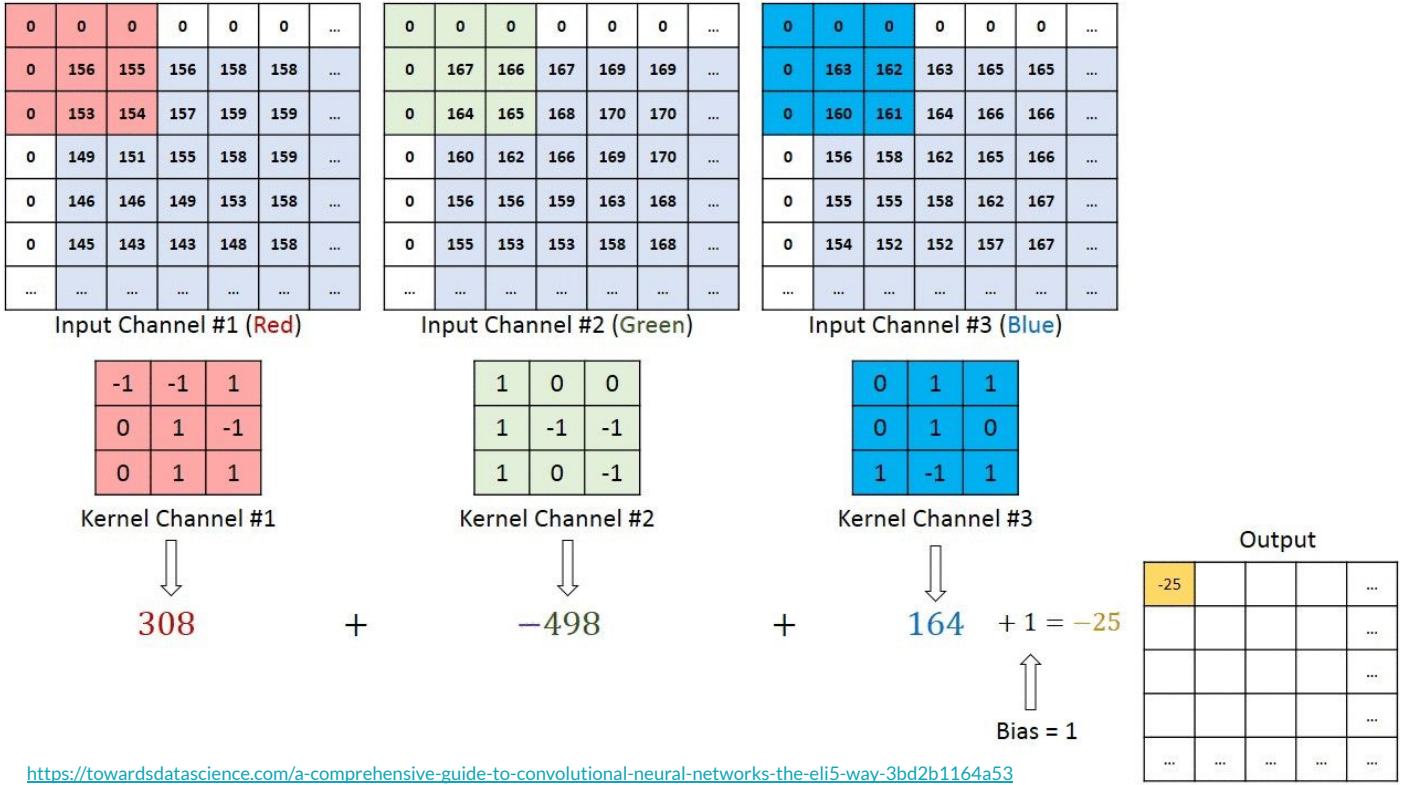


Sum across channels

Add bias term

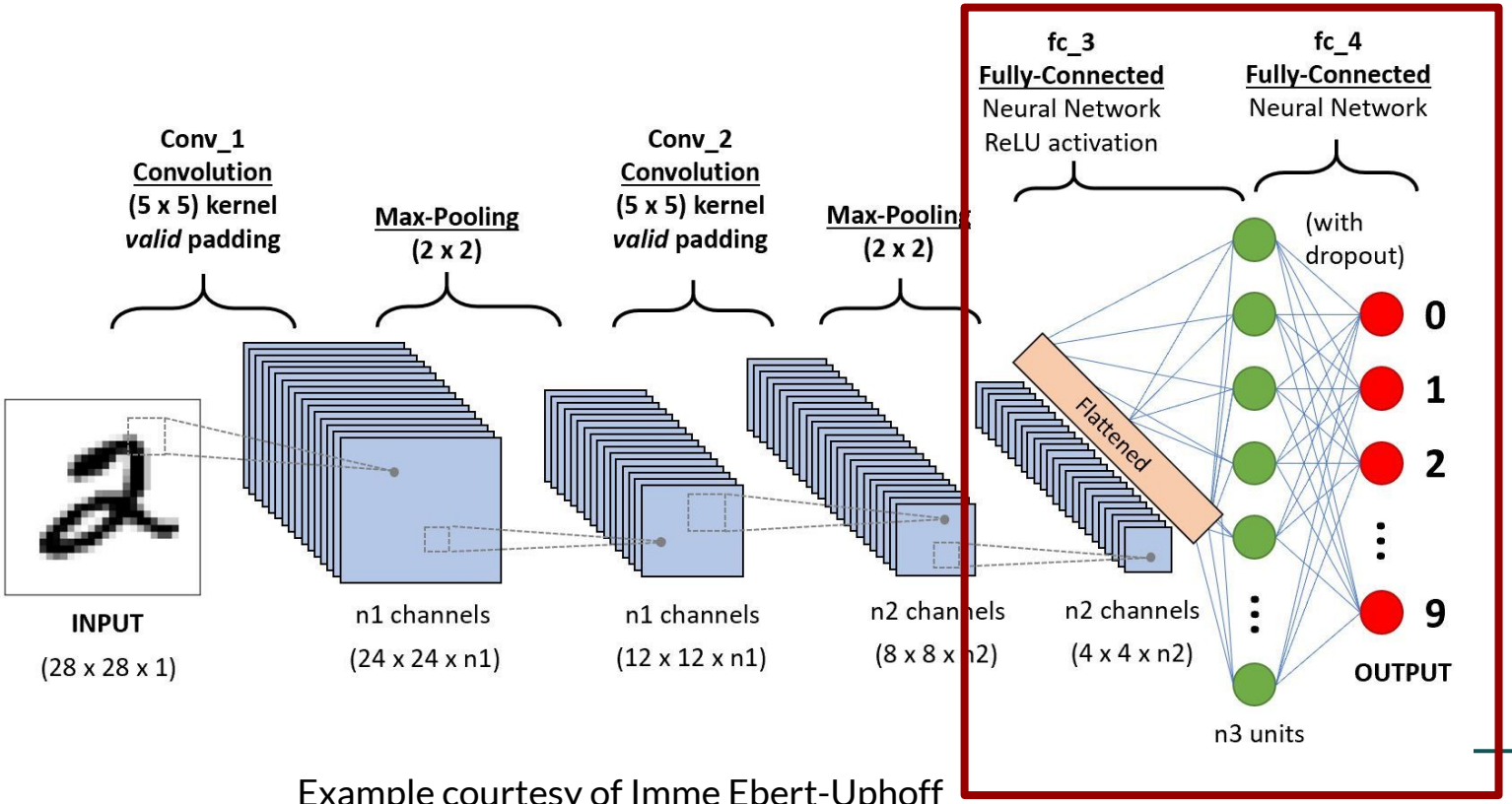
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Zero padding in action



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

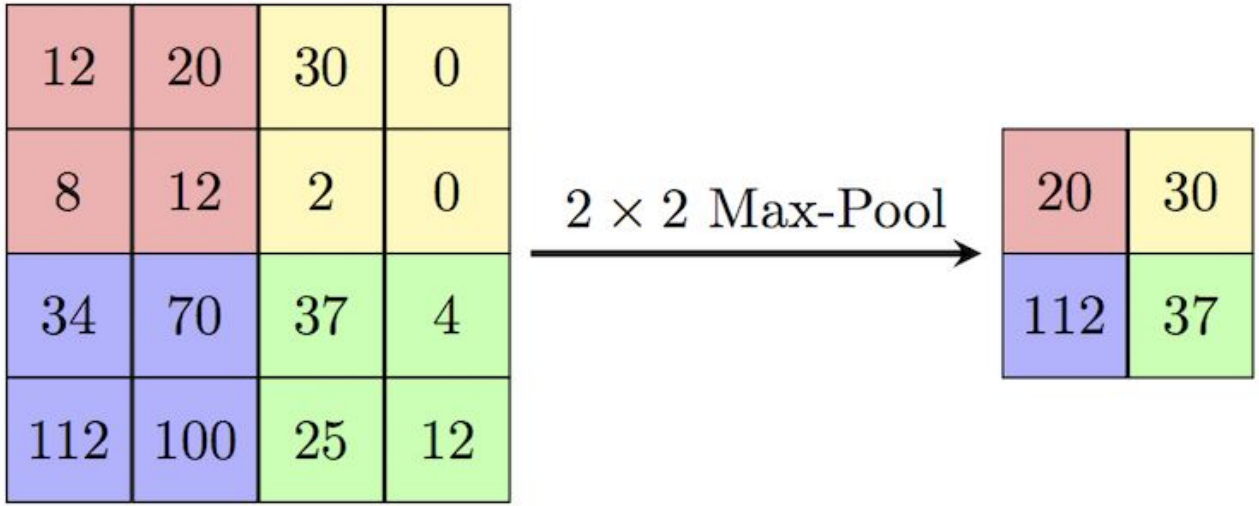
Connecting CNNs to Fully Connected Networks



Example courtesy of Imme Ebert-Uphoff

Pooling Layer

- **Reduce spatial size**
- Noise suppressant
- Max Pooling/Average/Sum Pooling



Seasonal ENSO prediction

Task: Predict Nino3.4 index __ months into the future using maps of the ocean state

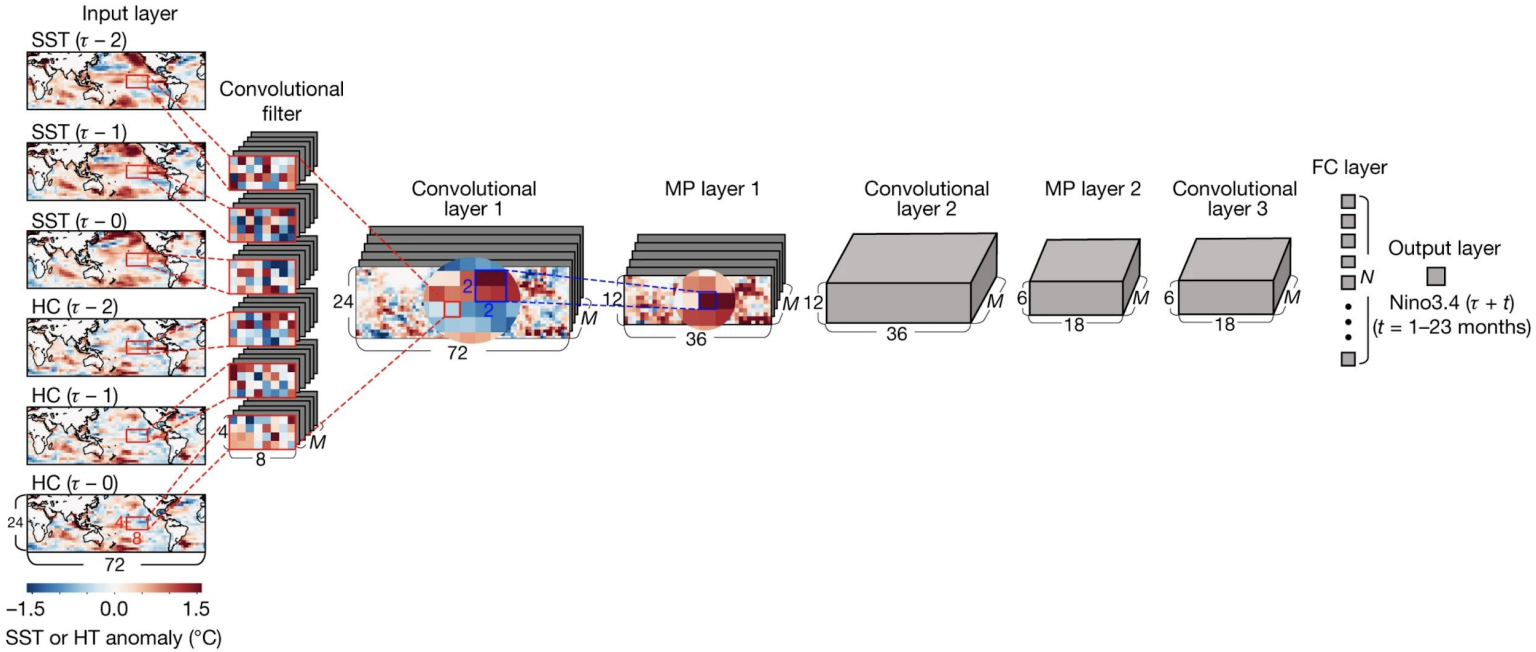


Figure from Ham et al. (2019)

Advantages of CNNs

- Easily retains the spatial relationships between adjacent pixels
  - CNNs are very common for machine learning tasks that use images as inputs
- Fewer weights and biases which speeds up training and reduces overfitting
  - The network only has to learn a small set of kernels and then apply them to *every part of the image* (rather than different weights/kernels for each part)

Satellite Image -> Synthetic Radar Image

Kyle Hilburn
[CSU/CIRA]

Next Article >

Journal of Applied Meteorology and Climatology

Volume 60: Issue 1

Sections

References

Figures

Cited By

Editorial Type: Article

Open access

Development and Interpretation of a Neural-Network-Based Synthetic Radar Reflectivity Estimator Using GOES-R Satellite Observations

Kyle A. Hilburn[1], Imme Ebert-Uphoff[2], and Steven D. ...

View More +

Published-online: 23 Dec 2020

Print Publication: 01 Jan 2021

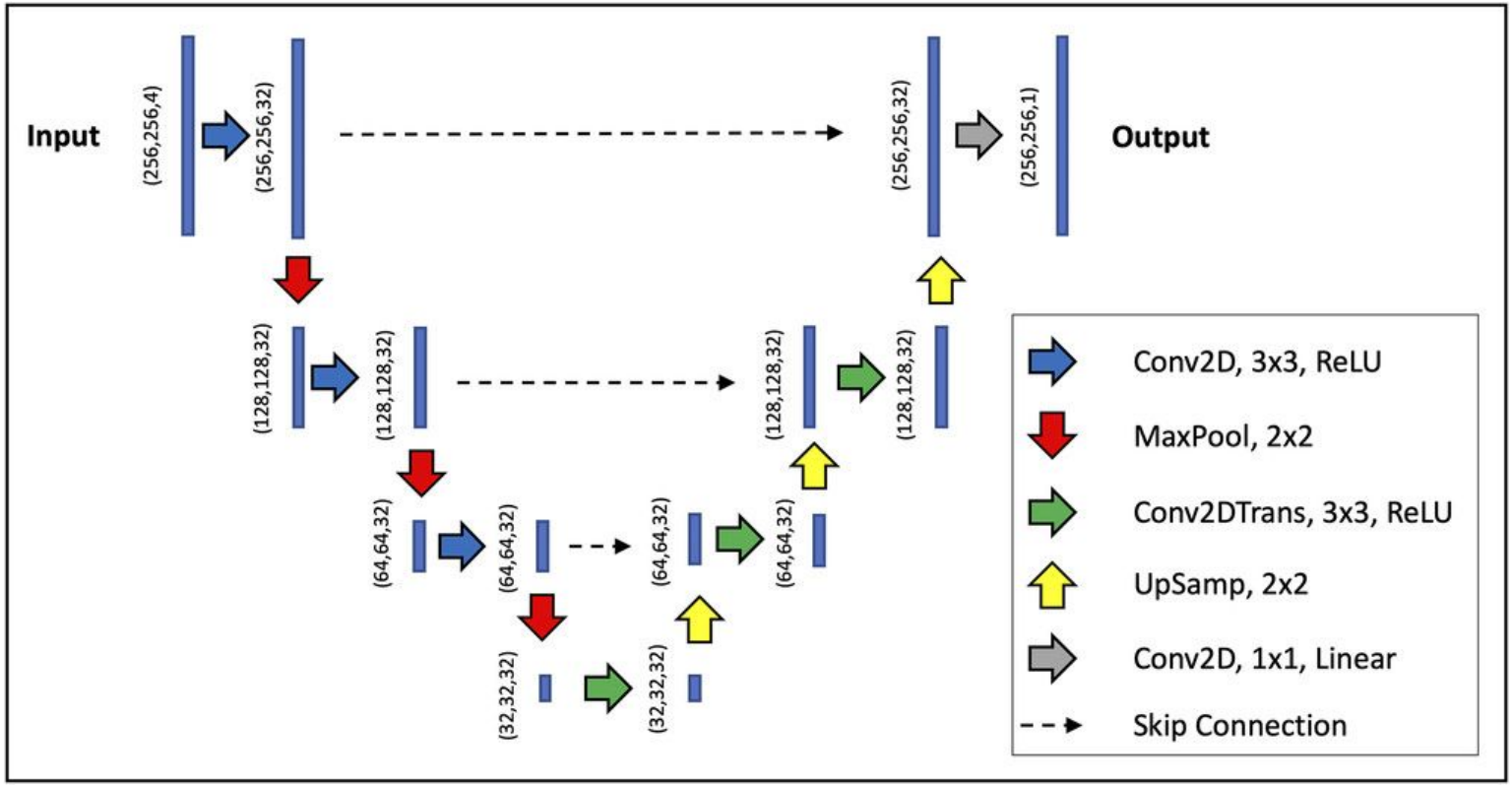
Collections: The 1st NOAA Workshop on Leveraging AI in the Exploitation of Satellite Earth Observations & Numerical Weather Prediction

DOI: https://doi.org/10.1175/JAMC-D-20-0084.1

Page(s): 3–21

U-Net Architecture

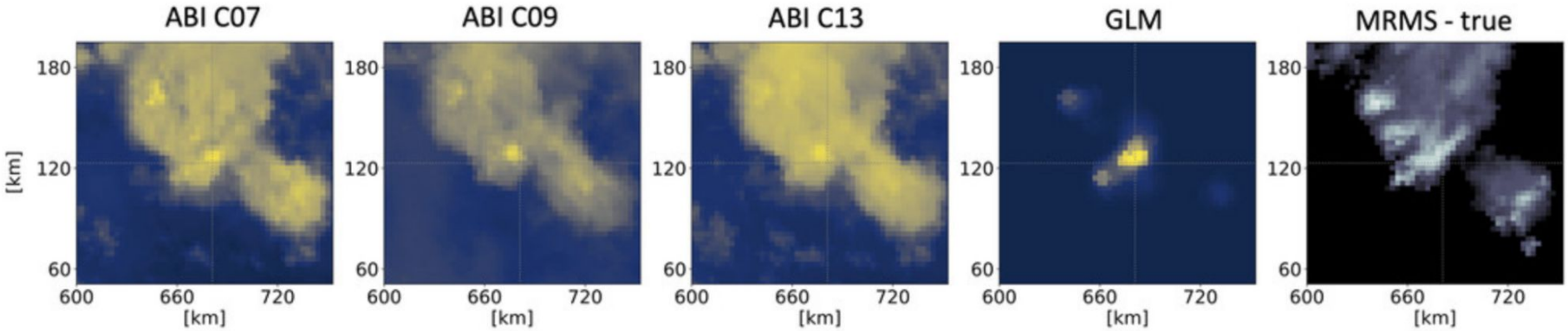
Kyle Hilburn
[CSU/CIRA]



https://journals.ametsoc.org/view/journals/apme/60/1/jamc-d-20-0084.1.xml

Inputs + Labels

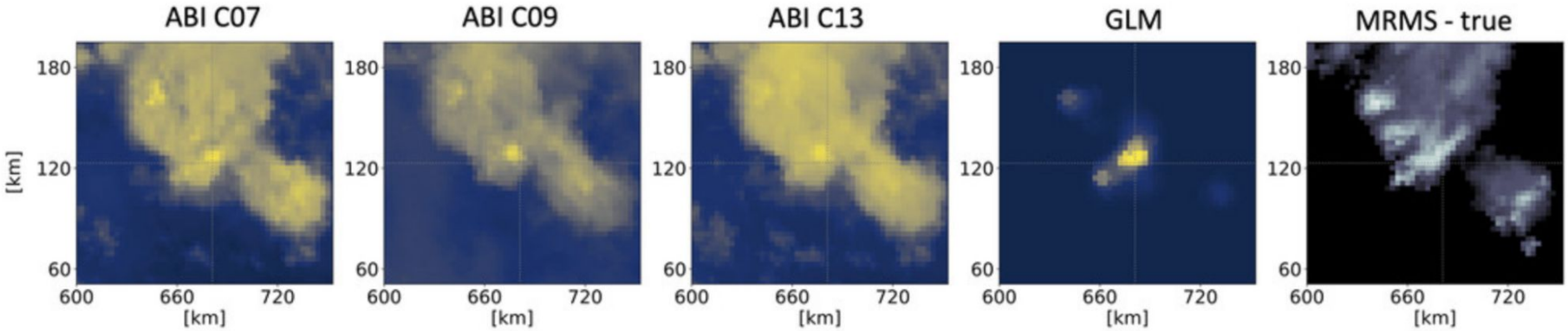
Kyle Hilburn
[CSU/CIRA]

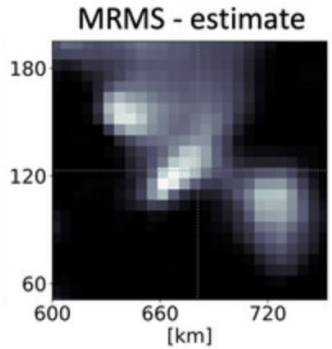


https://journals.ametsoc.org/view/journals/apme/60/1/jamc-d-20-0084.1.xml

Inputs + Labels

Kyle Hilburn
[CSU/CIRA]

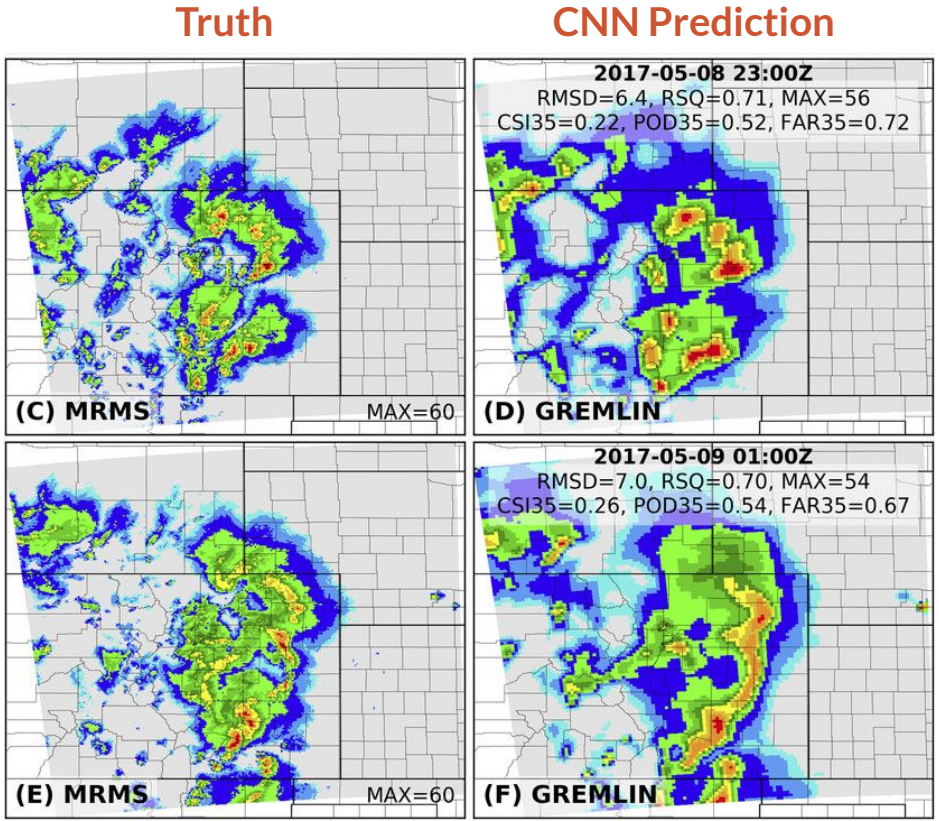




https://journals.ametsoc.org/view/journals/apme/60/1/jamc-d-20-0084.1.xml

Results!

Kyle Hilburn
[CSU/CIRA]

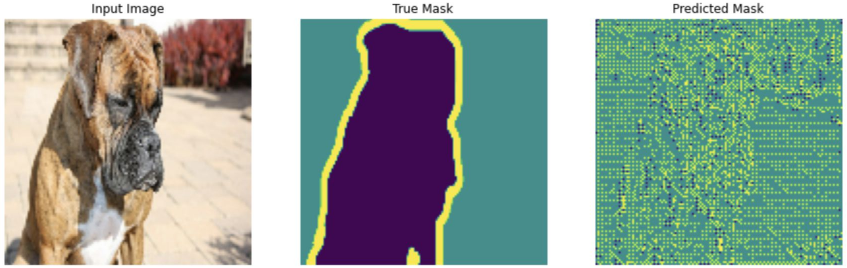


- MRMS = Satellite
- GREMLIN = CNN output

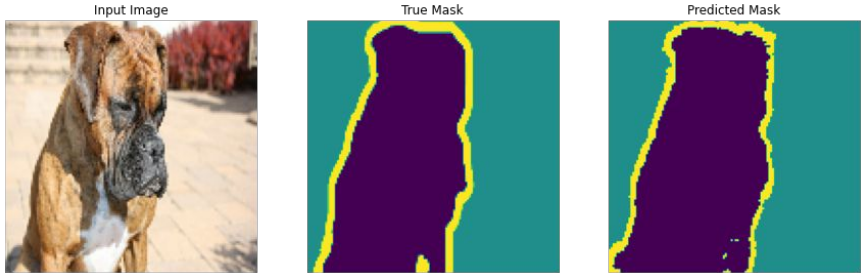
https://journals.ametsoc.org/view/journals/apme/60/1/jamc-d-20-0084.1.xml

Image Segmentation with a CNN

Before training...



After training...

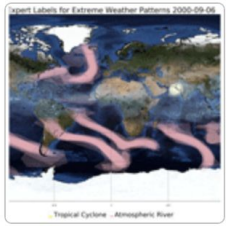


- Class 1: Pixel belonging to the pet.
- Class 2: Pixel bordering the pet.
- Class 3: None of the above/a surrounding pixel.

https://www.tensorflow.org/tutorials/images/segmentation

ClimateNet - using experts to label

ClimateNet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather



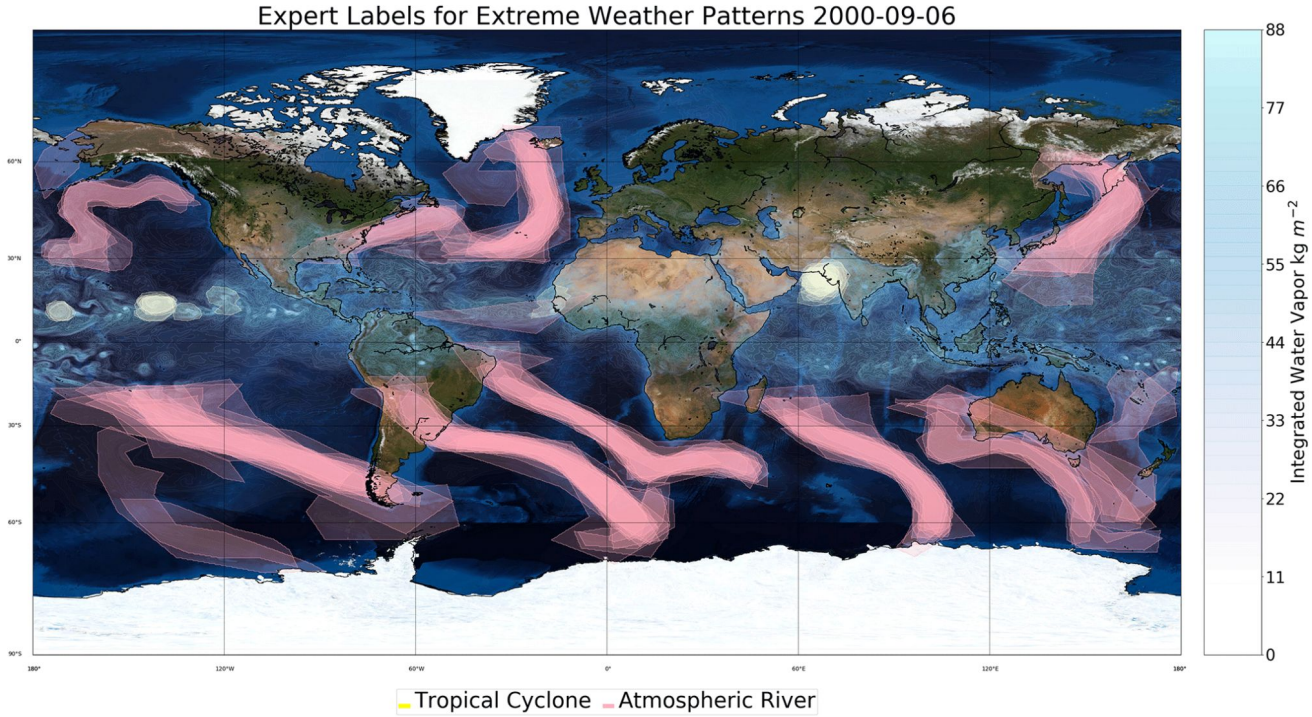
Prabhat[1,2,★], Karthik Kashinath[1,★], Mayur Mudigonda[10,★], Sol Kim[2], Lukas Kapp-Schwoerer[3], Andre Graubner[3], Ege Karaismailoglu[3], Leo von Kleist[3], Thorsten Kurth[4], Annette Greiner[1], Ankur Mahesh[2,1], Kevin Yang[2], Colby Lewis[2], Jiayi Chen[2], Andrew Lou[2], Sathyavat Chandran[5], Ben Toms[6], Will Chapman[7], Katherine Dagon[8], Christine A. Shields[8], Travis O'Brien[9,1], Michael Wehner[1], and William Collins[1,2]

https://gmd.copernicus.org/articles/14/107/2021/
https://portal.nersc.gov/project/ClimateNet/

ClimateNet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather

Prabhat[1,2*], Karthik Kashinath[1*], Mayur Mudigonda[3S*], Sol Kim[4], Lukas Kapp-Schwoerer[5], Andre Graubner[5], Ege Karaismailoglu[5], Leo von Kleist[5], Thorsten Kurth[6], Annette Greiner[7], Ankur Mahesh[2,1], Kevin Yang[2], Colby Lewis[4], Jiaqi Chen[2], Andrew Lou[2], Sathyavat Chandran[2], Ben Toms[8], Will Chapman[9], Katherine Dagon[10], Christine A. Shields[9], Travis O'Brien[11,1], Michael Wehner[12], and William Collins[13,2]

ClimateNet - using experts to label

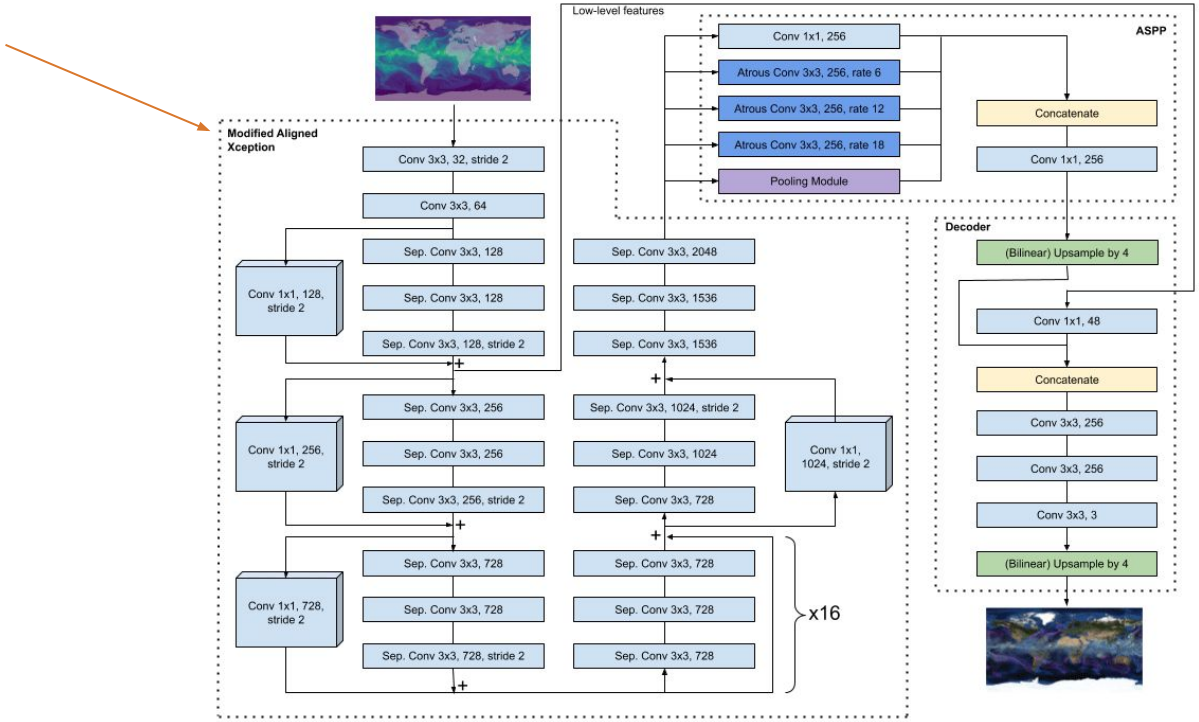


https://gmd.copernicus.org/articles/14/107/2021/

ClimateNet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather

Prabhat[1,2*], Karthik Kashinath[1*], Mayur Mudigonda[3,4*], Sol Kim[1], Lukas Kapp-Schwoerer[5], Andre Graubner[5], Ege Karaismailoglu[5], Leo von Kleist[5], Thorsten Kurth[1], Annette Greiner[1], Ankur Mahesh[1,2], Kevin Yang[1], Colby Lewis[1], Jiaqi Chen[1], Andrew Lou[1], Sathyavat Chandran[1], Ben Toms[6], Will Chapman[7], Katherine Dagon[8], Christine A. Shields[9], Travis O'Brien[10,11], Michael Wehner[12], and William Collins[13,2]

Image Segmentation w/ ClimateNet

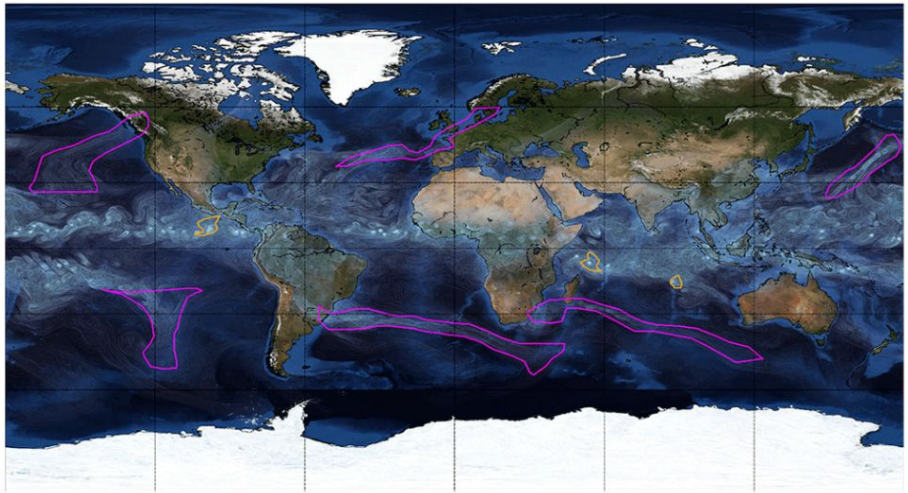


https://gmd.copernicus.org/articles/14/107/2021/

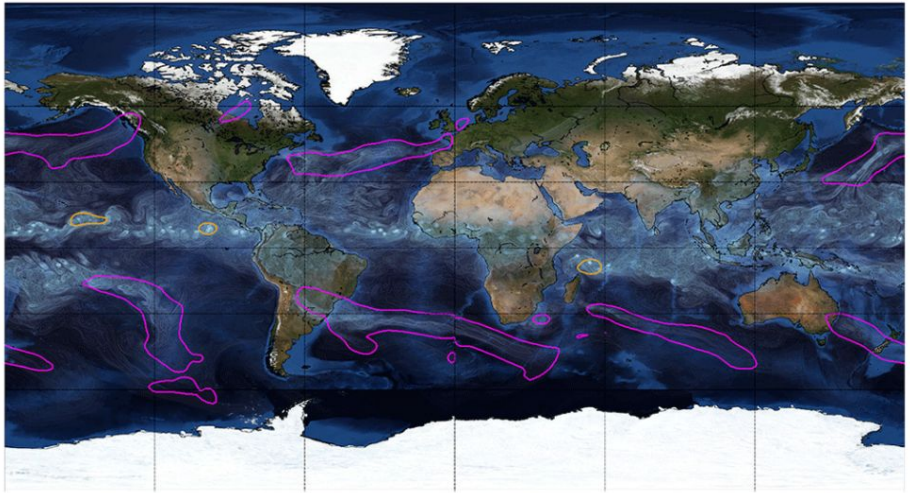
ClimateNet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather

Prabhat[1,2*], Karthik Kashinath[1*], Mayur Mudigonda[3a*], Sol Kim[2], Lukas Kapp-Schwoerer[2], Andre Graubner[2], Ege Karaismailoglu[2], Leo von Kleist[2], Thorsten Kurth[4], Annette Greiner[2], Ankur Mahesh[2,1], Kevin Yang[2], Colby Lewis[4], Jiaqi Chen[2], Andrew Lou[2], Sathyavat Chandran[2], Ben Toms[2], Will Chapman[2], Katherine Dagon[4], Christine A. Shields[4], Travis O'Brien[4,5], Michael Wehner[3], and William Collins[3,2]

Image Segmentation w/ ClimateNet



Truth (Labeled)



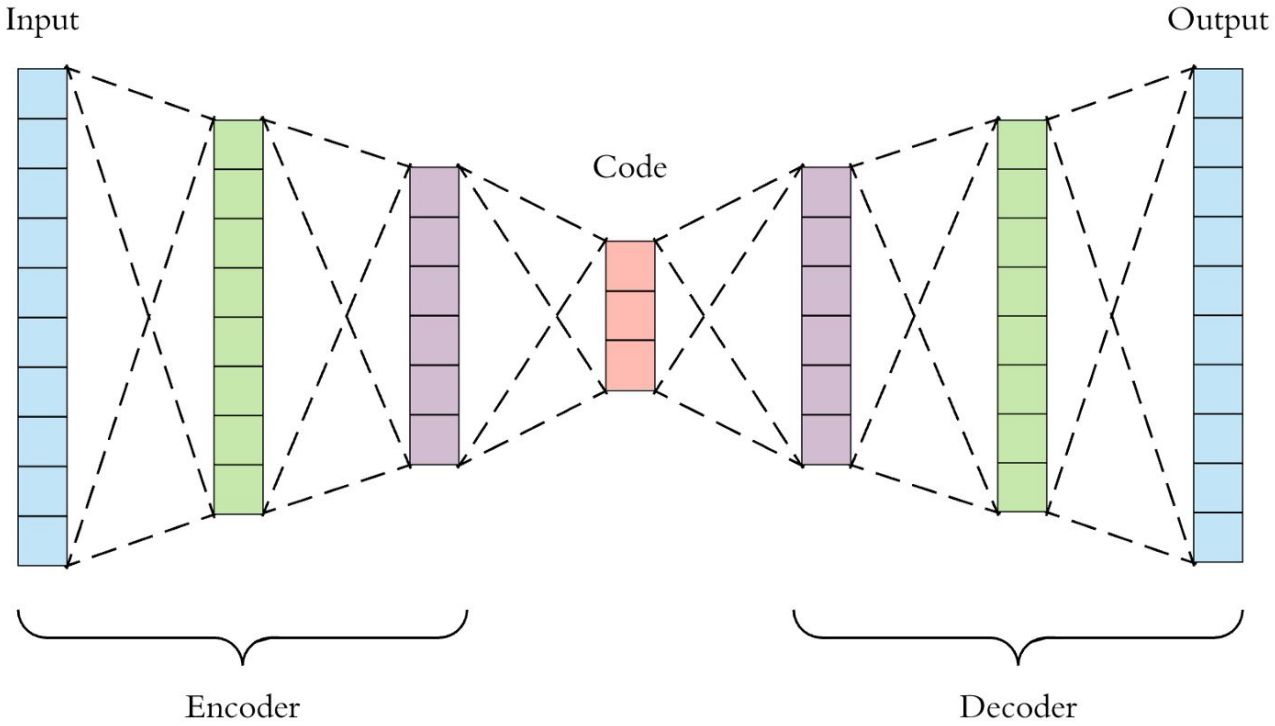
Predicted by Trained CNN

Trained on ClimateNet

https://gmd.copernicus.org/articles/14/107/2021/

Autoencoders

Even more advanced techniques

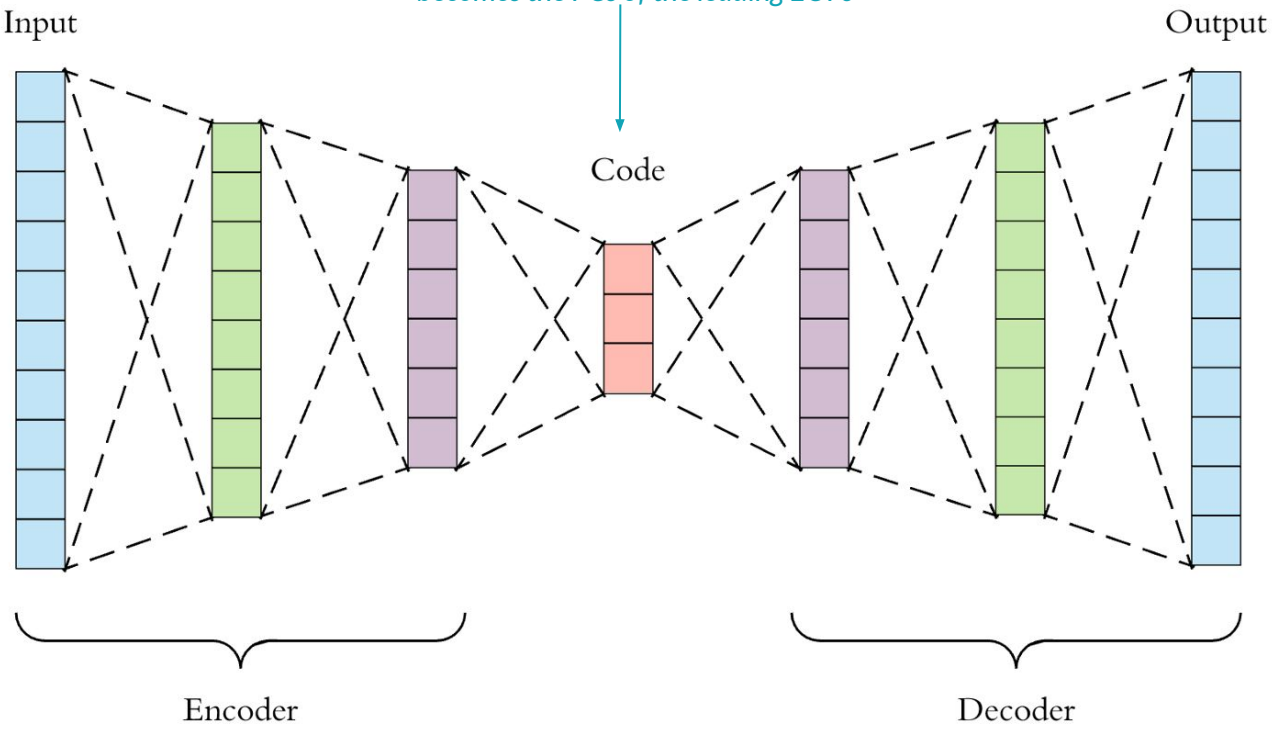


https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Autoencoders

Even more advanced techniques

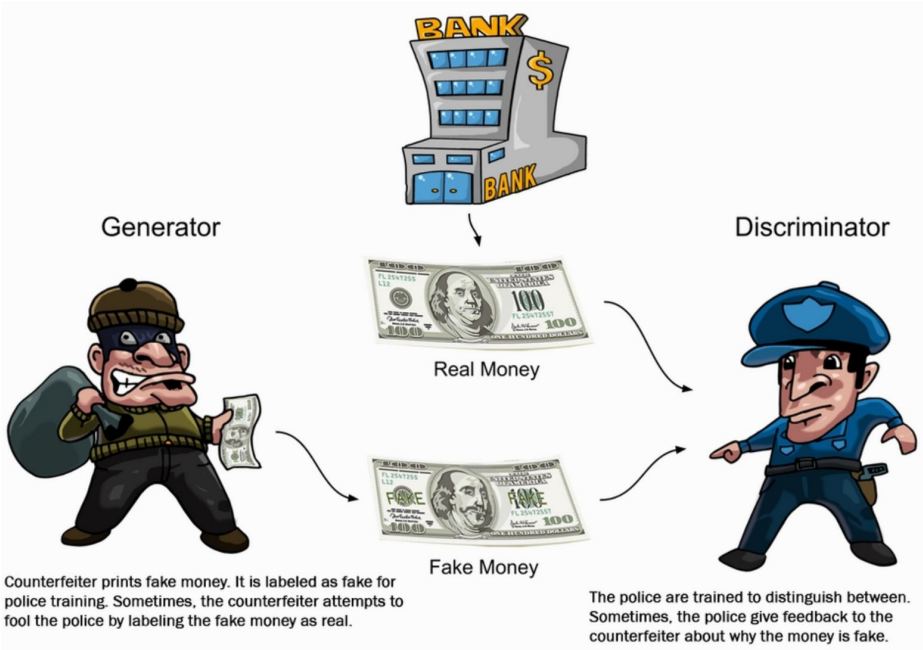
If your activation function is linear, this becomes the PCs of the leading EOFs



https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798

Generative Adversarial Networks

Even more advanced techniques



- First introduced by Goodfellow et al. (2014)
- Competition between **generator** and **discriminator**

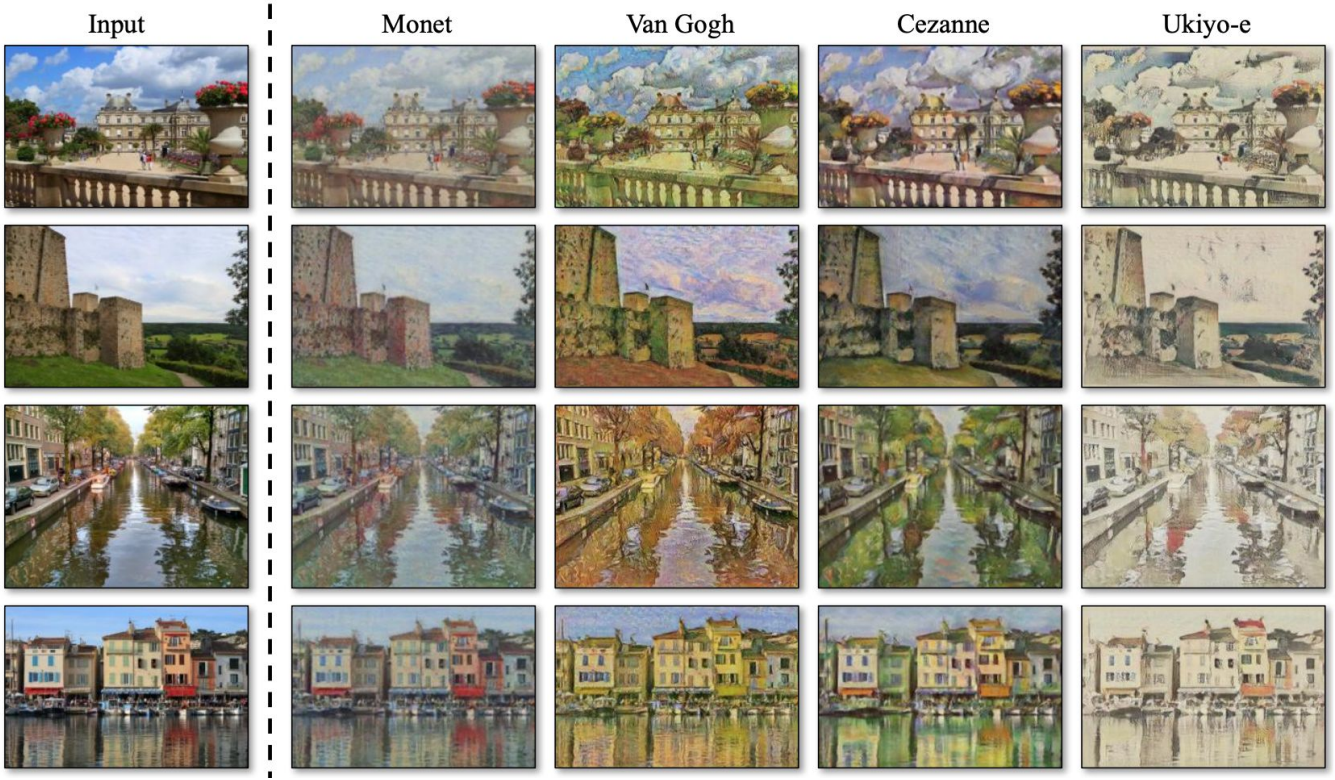
Figure by Atienza, Rowel. Advanced Deep Learning with Keras: Apply deep learning techniques, autoencoders, GANs, variational autoencoders, deep reinforcement learning, policy gradients, and more. Packt Publishing Ltd, 2018.

Visualizing Climate Change & Associated Hazards

https://mila.quebec/en/article/this-climate-does-not-exist-picturing-impacts-of-the-climate-crisis-with-ai-one-address-at-a-time/
https://thisclimatedoesnotexist.com/home

Style GANs generated these faces. They are not real.

https://arxiv.org/abs/1406.2661



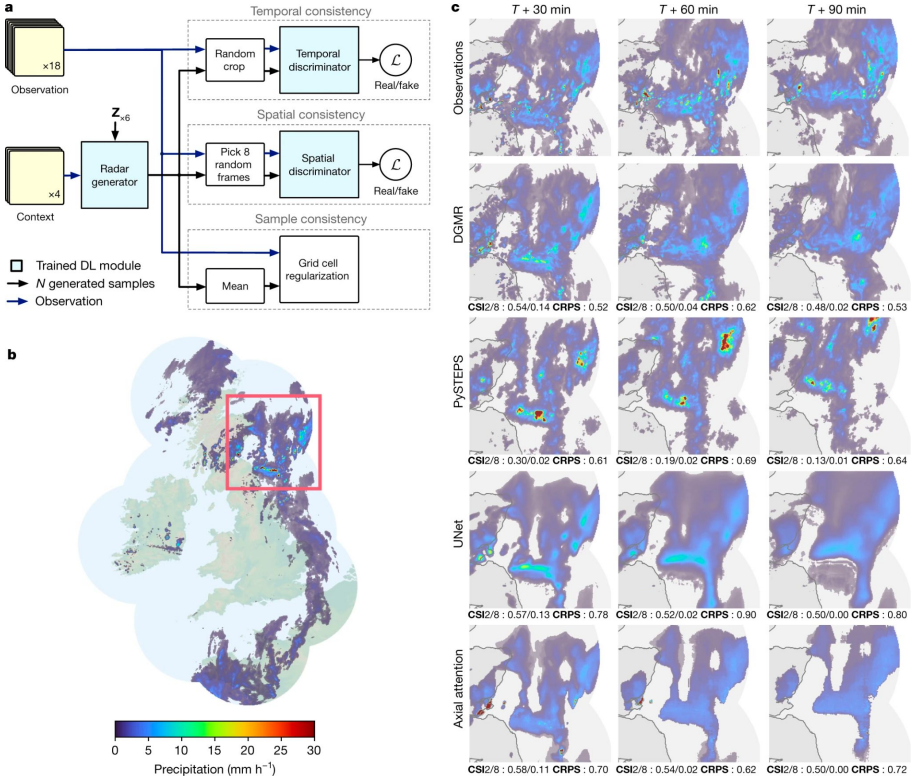
Cycle GAN

https://arxiv.org/abs/1703.10593

Cycle GAN

https://arxiv.org/abs/1703.10593

Fig. 1: Model overview and case study of performance on a challenging precipitation event starting on = 24 June 2019 at 16:15 UK, showing convective cells over eastern Scotland.



DGMR is better able to predict the spatial coverage and convection compared to other methods over a longer time period, while not over-estimating the intensities, and is significantly preferred by meteorologists (93% first choice, $n = 56$, $P < 10^{-4}$). **a**, Schematic of the model architecture showing the generator with spatial latent vectors **Z**. **b**, Geographic context for the predictions. **c**, A single prediction at $T + 30$, $T + 60$ and $T + 90$ min lead time for different models. Critical success index (CSI) at thresholds 2 mm h^{-1} and 8 mm h^{-1} and continuous ranked probability score (CRPS) for an ensemble of four samples shown in the bottom left corner. For axial attention we show the mode prediction. Images are 256 km × 256 km. Maps produced with Cartopy and SRTM elevation data[46].

nature

Explore content | About the journal | Publish with us

nature > articles > article

Article | Open Access | Published: 29 September 2021

Skilful precipitation nowcasting using deep generative models of radar

Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, Rachel Prudden, Amol Mandhane, Aidan Clark, Andrew Brock, Karen Simonyan, Raia Hadsell, Niall Robinson, Ellen Clancy, Alberto Arribas & Shakir Mohamed

Nature **597**, 672–677 (2021) | Cite this article

95k Accesses | **24** Citations | **855** Altmetric | Metrics

GANs for Nowcasting

https://arxiv.org/abs/2005.10374

https://makeoptim.com/en/deep-learning/tensorflow-metal
https://developer.apple.com/metal/tensorflow-plugin/
https://github.com/apple/tensorflow_macos
https://github.com/conda-forge/miniforge

In a nutshell:

- Install **Xcode** and **Command Line Tools**
- Install **miniforge** (Anaconda does not run on the M1)
- Then, open a terminal and do the following:
    - `conda create --name env-name python=3.9`
    - `conda activate env-name`
    - `conda install -c apple tensorflow-deps==2.7`
    - `python -m pip install tensorflow-macos==2.7`
    - `----------`
    - `pip install tensorflow-probability==0.15`
    - `pip install --upgrade numpy scipy pandas statsmodels matplotlib seaborn palettable progressbar2 tabulate icecream flake8 keras-tuner jupyterlab black isort jupyterlab_code_formatter`
    - `pip install -U scikit-learn`
    - `pip install silence-tensorflow tqdm`
    - `conda install -c conda-forge cmocean cartopy xarray dask netCDF4`

Instructions for installing Tensorflow on an Apple M1 Chip

Ethical Use of AI in Earth Science

A fun motivational story

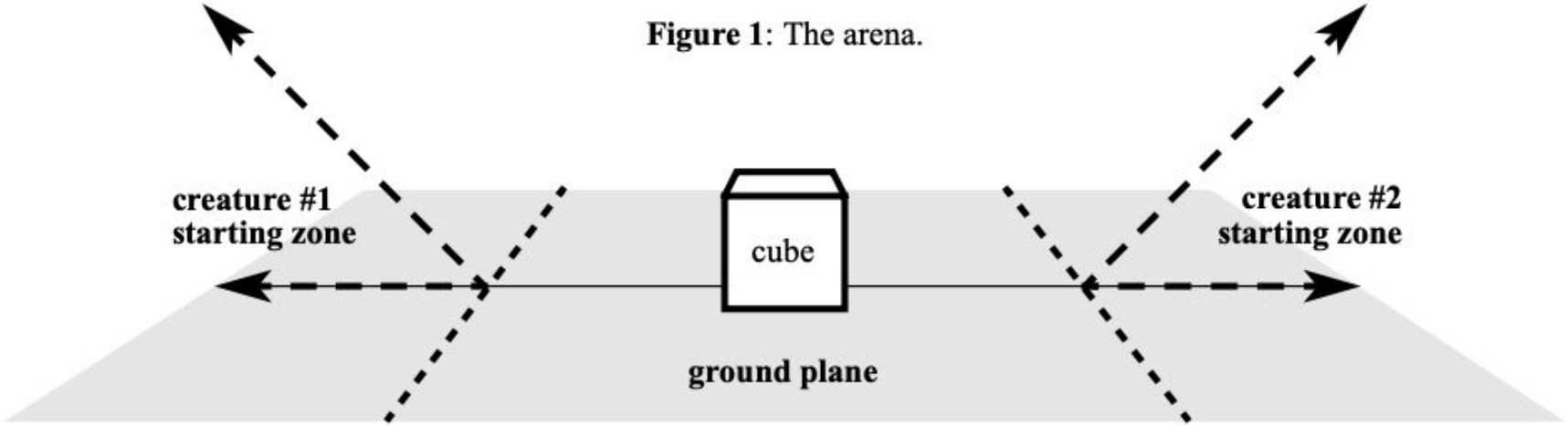
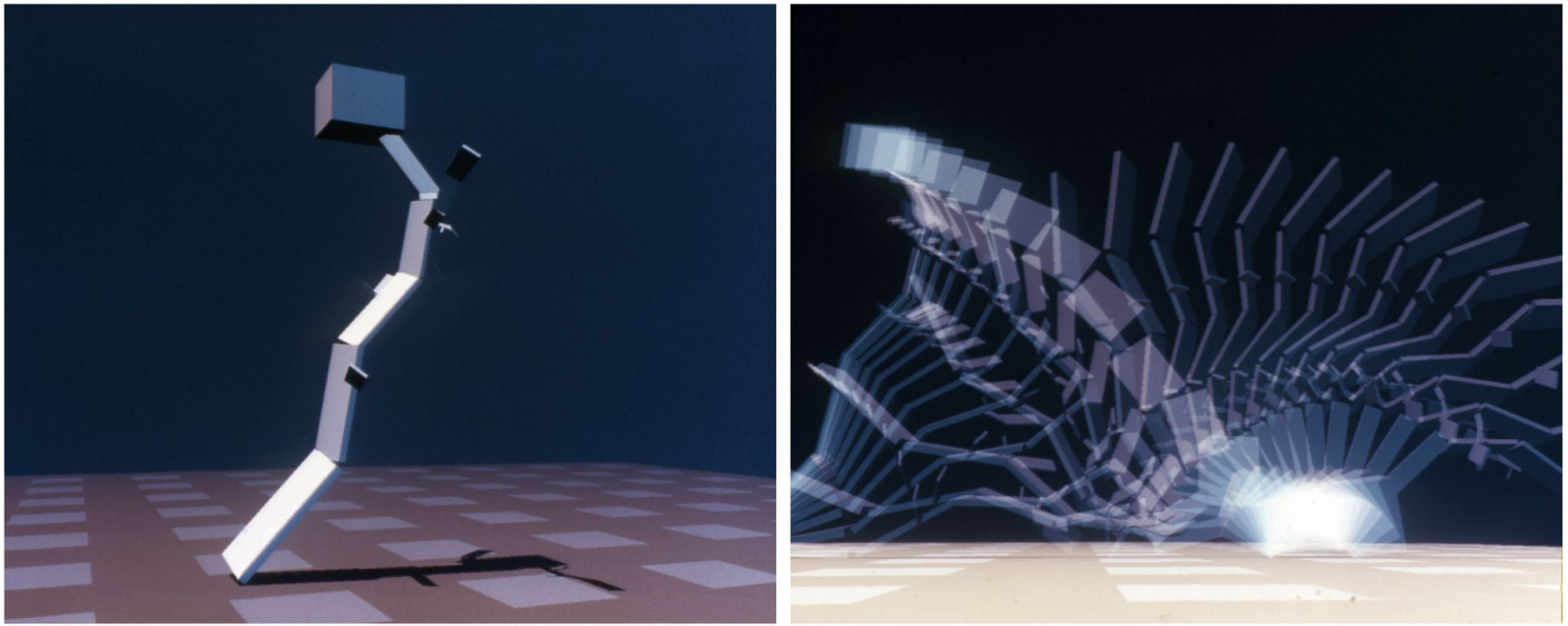


Figure 1: The arena.

Sims (1994) https://www.karlsims.com/papers/alife94.pdf

A fun motivational story



Lehman et al. (2020) https://doi.org/10.1162/artl_a_00319

Using AI to hire new workers

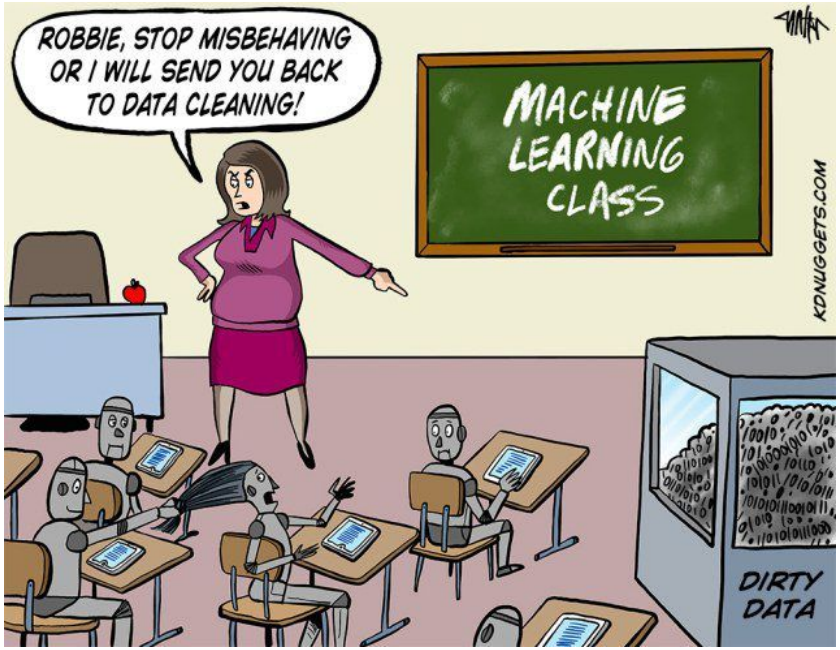
Goal: Feed an AI all resumes and have the AI return the top candidates

Using AI to hire new workers

Goal: Feed an AI all resumes and have the AI return the top candidates

Problem: AI had a gender bias no matter how it was trained

AI Ethics: Using AI methods in an ethical manner as to not cause harm to others. Accomplished through careful thought in data processing, AI development, and deployment.



The Need for Ethical, Responsible, and Trustworthy Artificial Intelligence for Environmental Sciences

Amy McGovern[1]*, Imme Ebert-Uphoff[2,3], David John Gagne II[4] and Ann Bostrom[5]

Impact Statement

This position paper discusses the need for the environmental sciences community to ensure that they are developing and using artificial intelligence (AI) methods in an ethical and responsible manner. This paper is written at a general level, meant for the broad environmental sciences and earth sciences community, as the use of AI methods continues to grow rapidly within this community.

https://arxiv.org/pdf/2112.08453.pdf

Ways in which AI can go wrong for environmental sciences

Issues related to training data:

1. Non-representative training data, including lack of geo-diversity
2. Training labels are biased or faulty
3. Data is affected by adversaries

Issues related to AI models:

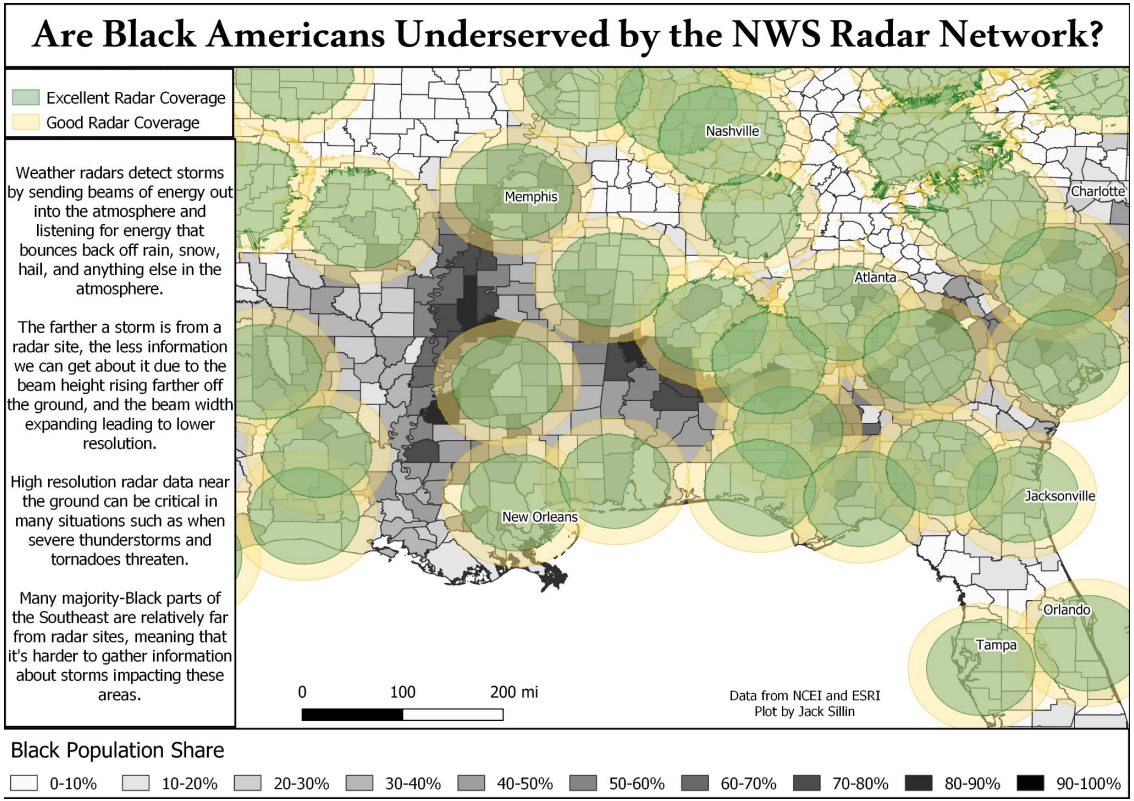
1. Model training choices
2. Algorithm learns faulty strategies
3. AI learns to fake something plausible
4. AI model used in inappropriate situations
5. Non-trustworthy AI model deployed
6. Lack of robustness in the AI model

Other issues related to workforce and society:

1. Globally applicable AI approaches may stymie burgeoning efforts in developing countries.
2. Lack of input or consent on data collection and model training
3. Scientists might feel disenfranchised.
4. Increase of CO_2 emissions due to computing

Example of biased data in Earth science

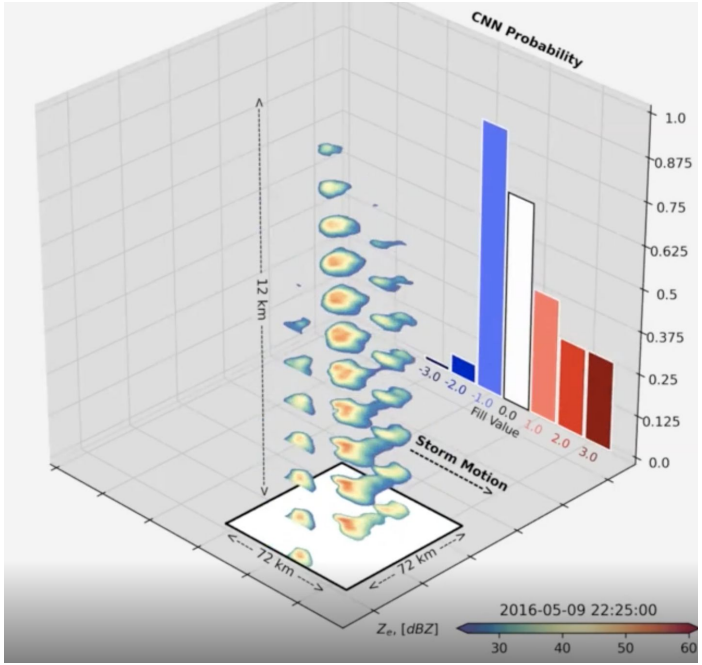
Radar coverage is unequal



McGovern et al. 2021 https://doi.org/10.1175/BAMS-D-18-0195.1

Bias can be introduced unintentionally!

- The fill value used for missing data can alter the model's prediction

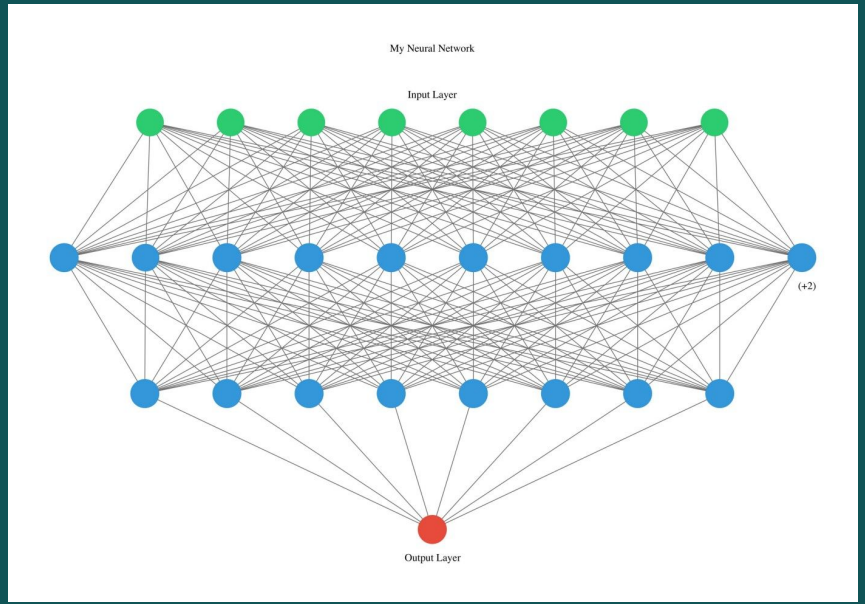


Chase and McGovern (2022) "Deep Learning Parameter Considerations When Using Radar and Satellite Measurements"

What can you do?

- Get to know your data
  - How was it collected?
  - Unequal samples?
  - Fully representative?
- Consider how you create and train your model
  - What assumptions are your model based on?
- After training spend time figuring out what your model has learned and why it is making its decisions
  - Explainability methods can help
- Be intentional!!!

Implementation and Assessment of an ANN



Different regularization techniques

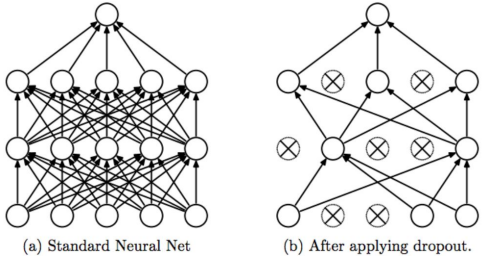
- **L2 (Ridge) Regression**
  - Forces weights to share importance

$$\text{Loss} = \text{RMSE} + \lambda \sum_{i=1}^{p} w_i^2$$

- **L1 (LASSO) Regression**
  - Penalizes high weights; sets low weights to zero

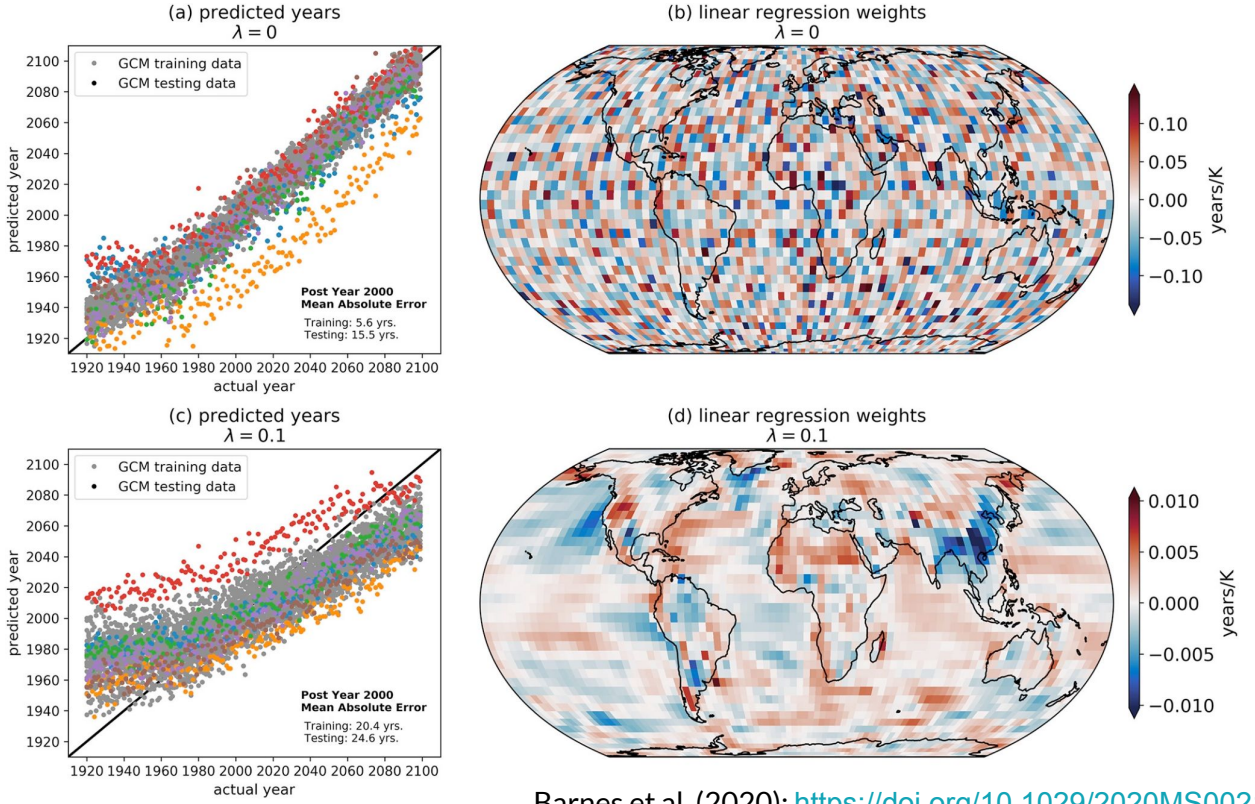
$$\text{Loss} = \text{RMSE} + \lambda \sum_{i=1}^{p} |w_i|$$

- **Dropout**



(a) Standard Neural Net (b) After applying dropout.

L2 Regularization: an example



Barnes et al. (2020): https://doi.org/10.1029/2020MS002195

Performance Measures

*Note- these are only used for **classification** problems!

Confusion Matrix: summary of prediction results

Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Assessment Scores:

Precision = TP / (TP + FP); (between 0 and 1)
Accuracy of positive predictions

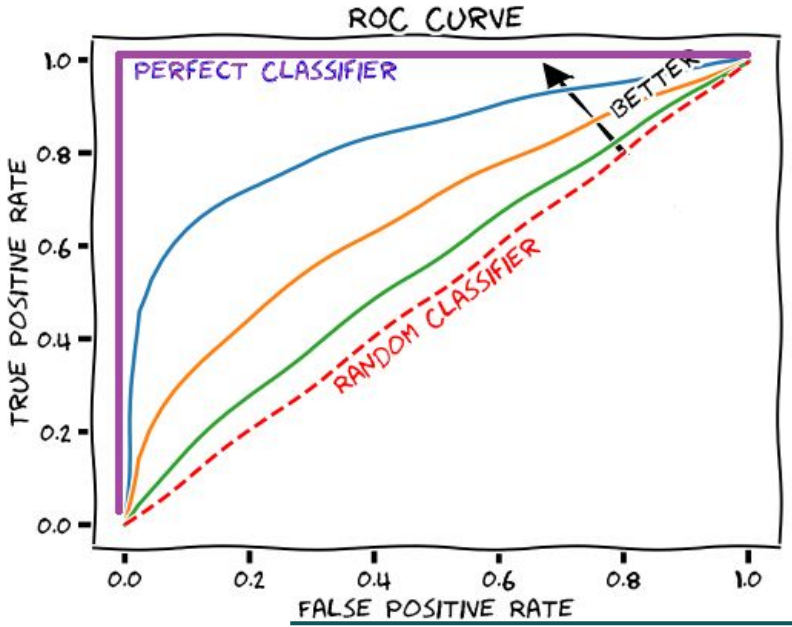
Recall = TP/ (TP + FN); (between 0 and 1)
Ratio of positive instances correctly detected by classifier

F1 score: harmonic mean of precision and recall; (between 0 and 1)

$$F1\ score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 * (Precision * Recall)}{(Precision + Recall)}$$

Performance Measures

Receiver Operating Characteristic (ROC)

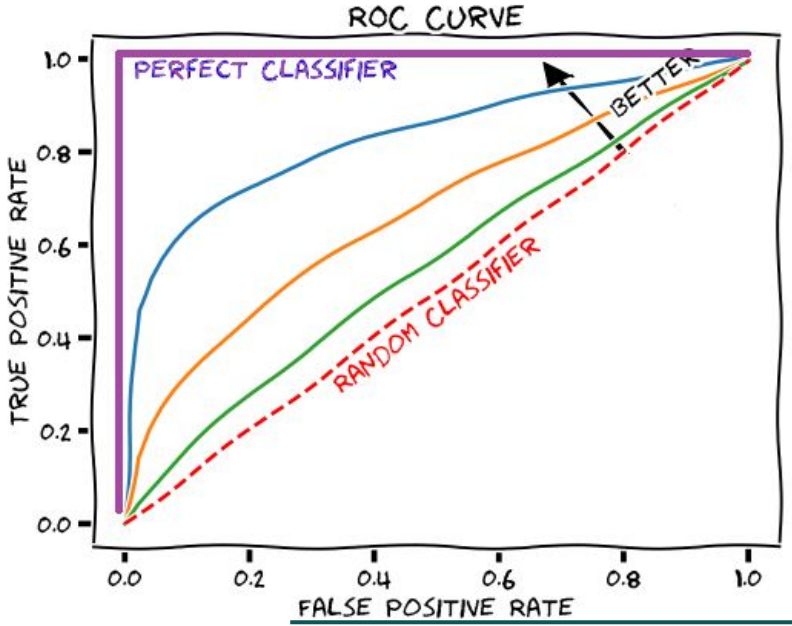


- Plots the false positive rate against the true positive rate for all possible thresholds
- Further from red dotted line, the better the model

- True positive rate = ratio of positive instances correctly detected by classifier (i.e. recall)
- False positive rate = ratio of negative instances that are *incorrectly* classified as positive

Performance Measures

Receiver Operating Characteristic (ROC)



Assessment Scores: measure the ***Area Under the Curve (AUC)***

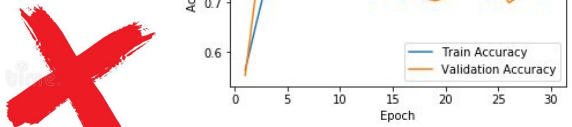
- Ranges from 0 to 1
- Higher AUC → better* model

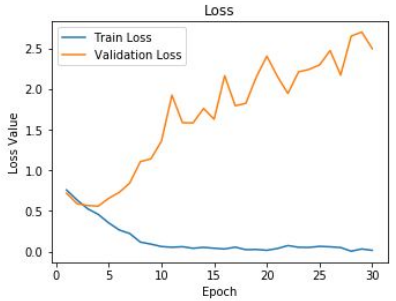
*remember, you (the scientist) decides what better means!

Performance Measures

Loss & Accuracy Assessment: User- Defined Loss Function

Overfit Model: training outperforms validation

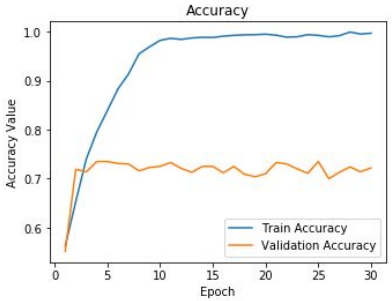


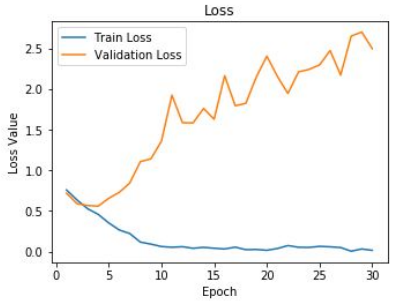


Performance Measures

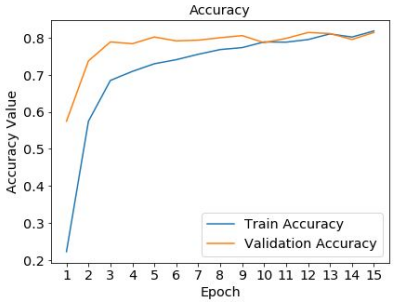
Loss & Accuracy Assessment: User- Defined Loss Function

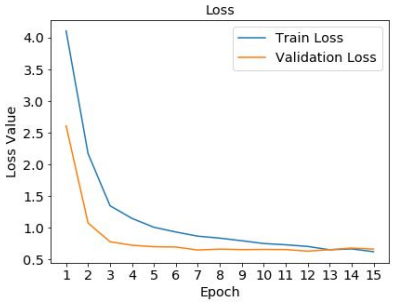
Overfit Model: training outperforms validation





Generalized Model! training matches validation





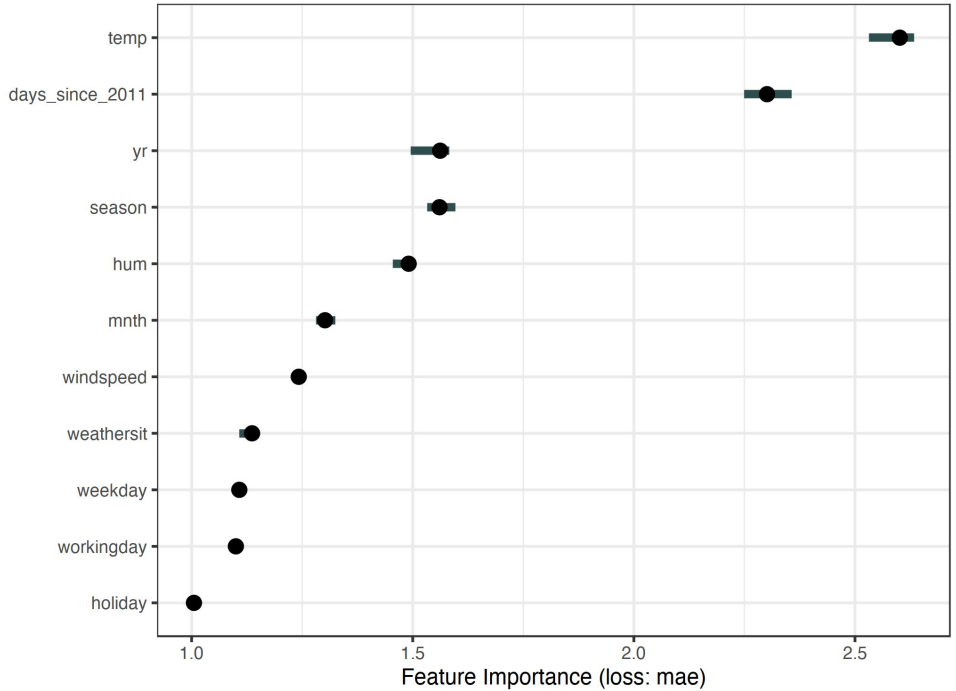
Permutation Importance

A way to detect features that are more important for accurate predictions, *Brieman (2001)*

For each feature, the feature values are shuffled, and new predictions made (with the shuffled feature values included).

If the model error increases after the shuffling, this feature is considered important. If there is no change in the model error, the feature in question adds no value to the model prediction.

Model error in predicting the number of rented bikes, given different weather and calendar related variables.



https://christophm.github.io/interpretable-ml-book/feature-importance.html

Permutation Importance

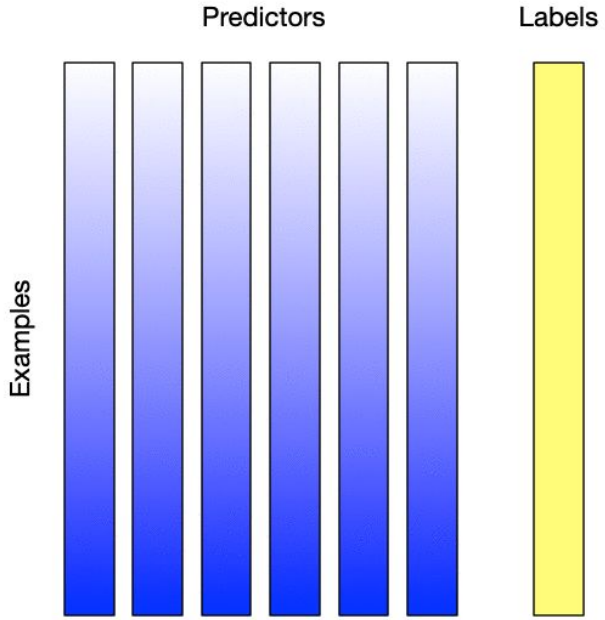
What if x is highly correlated with another predictor? → Multi-pass permutation

Freeze initially permuted predictor that degrades model performance the most, and re-permute other predictors to select second frozen predictor, and continue until all features are permuted and frozen

In comparison to the baseline model skill, can determine which predictors impact skill the most

Rank predictors based on error increase after permutation

McGovern et al. (2019)



Lakshmanan et al. (2015)

Time to code!

https://github.com/eabarnes1010/ml_tutorial_csu

From Github, open ann_ozone_joshuatree_metrics.ipynb

Methods of Explainable AI (XAI) for ANNs

Overview

1) Introduction to XAI:

i) Motivation for XAI

ii) The general idea of how XAI works

iii) Opportunities that XAI brings

iv) Representative methods and categories of XAI

2) Popular XAI methods:

i) Gradient

ii) Input*Gradient

iii) Layer-wise Relevance Propagation

ii) SHAP – SHapley Additive exPlanations

3) Benchmarking XAI:

i) Motivation - General idea

ii) Examples

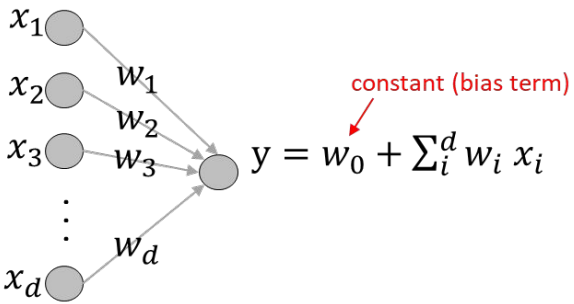
4) Summary

Introduction to XAI

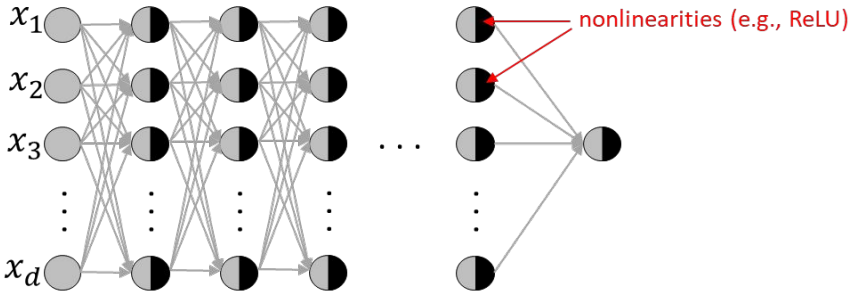
Why is XAI necessary?

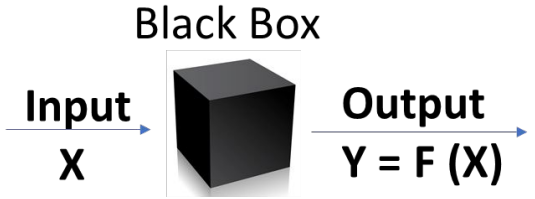
Scientists need to understand what the AI model is doing; what the decision-making process is.

Linear model: inherently interpretable



Neural Network: not inherently interpretable

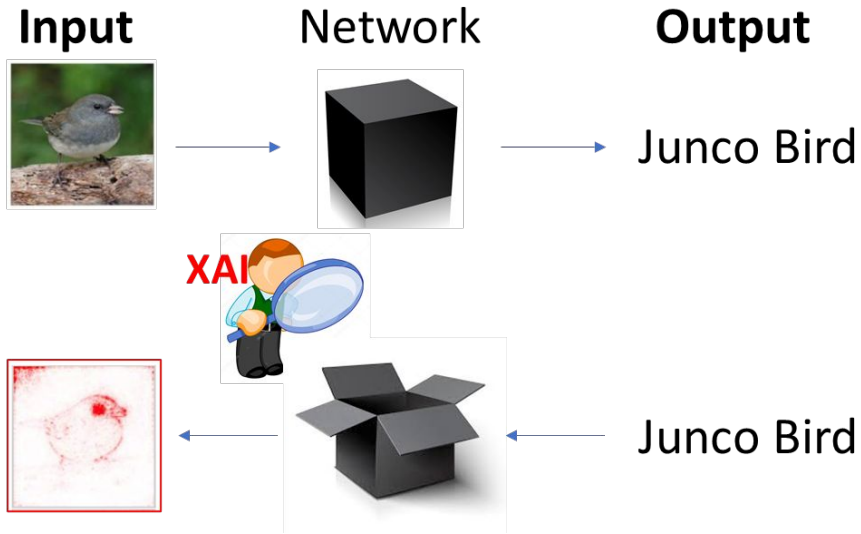




Why is XAI necessary?

Methods of eXplainable Artificial Intelligence (XAI) aim to explain how a Neural Network makes predictions, i.e., what the *decision strategy* is.

XAI methods highlight which features in the input space are important for the prediction: They produce the so-called *explanation/relevance heatmaps*.



Explanation (or relevance) Heatmap

From *Adebayo et al. (2020)*

Why is XAI necessary?

site: https://lrpserver.hhi.fraunhofer.de/image-classification

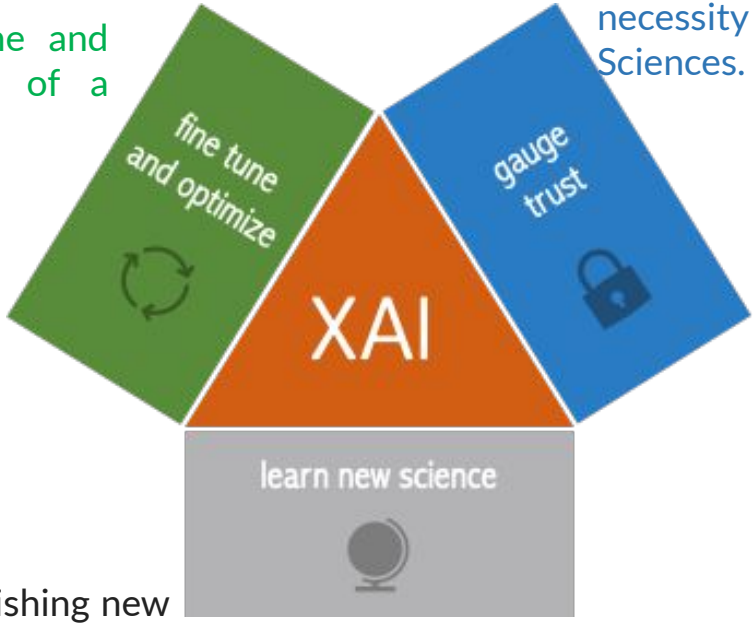
Methods of eXplainable Artificial Intelligence (XAI) aim to explain how a Neural Network makes predictions, i.e., what the *decision strategy* is.

XAI methods highlight which features in the input space are important for the prediction: They produce the so-called *explanation/relevance heatmaps*.

XAI: A potential *game changer* for prediction in Earth Sciences

XAI helps calibrate model trust and physically interpret the network, which is a necessity in many applications in Earth Sciences.

XAI may help fine-tune and optimize the strategy of a flawed model



XAI may help accelerate establishing new science, like investigating new climate teleconnections and gaining new insights.

From *Mamalakis et al. (2022)*

XAI methods and categories

AI models

Interpretable models
(e.g., linear models, decision trees)

Post-hoc Explainable models

Global XAI methods
(e.g., optimal input, permutation importance)

Local XAI methods

sensitivity
(e.g., Gradient)

attribution
(e.g., LRP, SHAP)

. . .

Table 3 Glossary of Interpretability Methods With Abbreviations Referenced Throughout Our Review

Method		Abbrv.	Method		Abbrv.
Anchors	[148]	ANCH	Layer-wise Relevance Propagation (full)	[13]	LRP
ApproShapley (Shapley Value Sampling)	[29]	AS	LRP (composite strategy)	[103], [126]	LRP-CMP
Class Activation Mapping	[206]	CAM	LRP (specific variants)	[13], [126]	LRP-$*$
Contextual Prediction Difference Analysis	[56]	CPDA	Local Interpretable Model-agnostic Explanations	[147]	LIME
DeconvNet	[201]	DCN	Meaningful Perturbation	[47]	MP
DeepLIFT	[170]	DL	NeuronConductance	[36]	NC
DeepLIFT (Rescale)	[170]	DLR	NeuronGuidedBackprop	[178]	NGB
DeepLIFT SHAP	[116]	DLSHAP	NeuronIntegratedGradients	[172]	NIG
Deep Taylor Decomposition	[127]	DTD	Occlusion Analysis	[201]	OCC
ExcitationBackprop	[202]	EB	PatternAttribution	[90]	PA
ExtremalPerturbation	[46]	EP	PatternNet	[90]	PN
GNNExplainer	[198]	GNNEXP	Prediction Difference Analysis	[208]	PDA
GNN-LRP	[162]	GLRP	Randomized Input Sampling for Explanation	[142]	RISE
GradCAM	[167]	GC	Saliency Analysis / Gradient	[14], [174]	SA
Gradient SHAP	[116]	GSHAP	SHapley Additive exPlanations	[116]	SHAP
Gradient $\times$ Input	[170]	GI	SHAP Interaction Index	[115]	SHAPIDX
GuidedBackprop	[178]	GB	SmoothGrad	[176]	SG
Guided GradCam	[167]	GGC	SmoothGrad^2	[76]	SG-SQ
Integrated Gradients	[183]	IG	Spectral Relevance Analysis	[104]	SpRAy
Internal Influence	[110]	II	TreeExplainer	[115]	TEXP
Kernel SHAP	[116]	KSHAP	VarGrad	[1]	VG
LayerConductance	[172]	LC	Testing with Concept Activation Vectors	[89]	TCAV
Local Rule-based Explanations	[58]	LORE	TotalConductance	[36]	TC

From *Samek et al. (2021)*

Popular XAI Methods

Gradient (sensitivity)

- ***Sensitivity*** refers to how much sensitive the value of the output is to a specific input feature. It is essentially the gradient (i.e., the first derivative if we think the network as a function) of the output with respect to the input. **[units output/units input]**

Relevance of feature i for prediction n

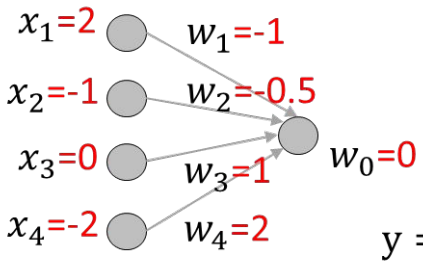
$$R_{i,n} = \frac{\partial \hat{F}}{\partial X_i}\Big|_{X_i = x_{i,n}}$$

Network

Value of feature i in sample n

Partial derivative

Simple case: Linear model



$$y = w_0 + \sum_i^4 w_i\, x_i$$

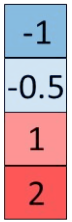
$$= 0+(-2)+0.5+0+(-4)= -5.5$$

Explanation of -5.5:

- The ***sensitivity*** of -5.5 to the feature x_i is $\frac{\partial y}{\partial x_i} = \boldsymbol{w_i}$.

(sensitivity is NOT dependent on x point; not true in nonlinear models)

heatmap



Input*Gradient (attribution)

- ***Attribution*** refers to the relative contribution of a specific input feature to the output. **[units output]**

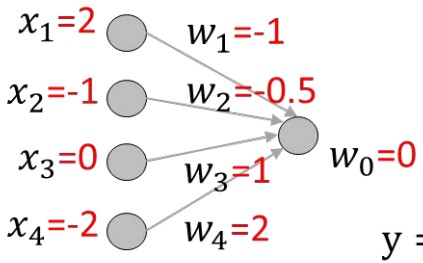
Relevance of feature i for prediction n

Input

Gradient

$$R_{i,n} = x_{i,n} * \frac{\partial \hat{F}}{\partial X_i}\bigg|_{X_i = x_{i,n}}$$

Simple case: Linear model



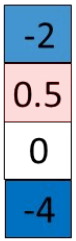
$$y = w_0 + \sum_i^4 w_i\, x_i$$

= 0+(-2)+0.5+0+(-4)= -5.5

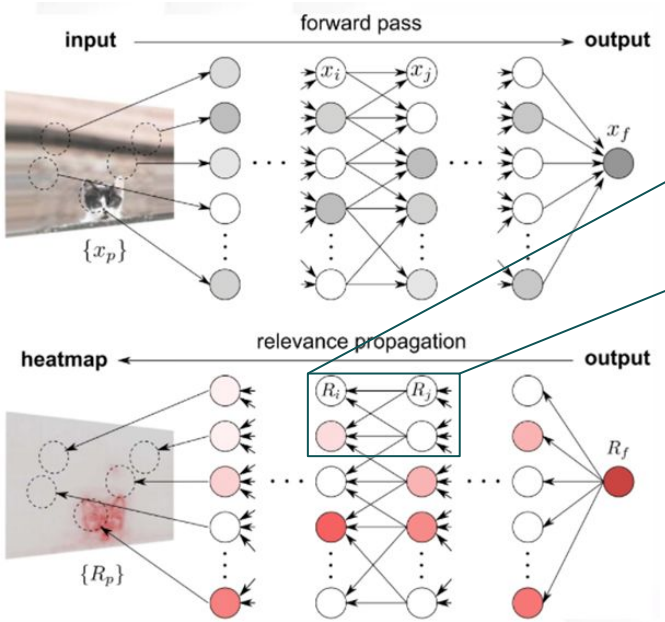
Explanation of -5.5:

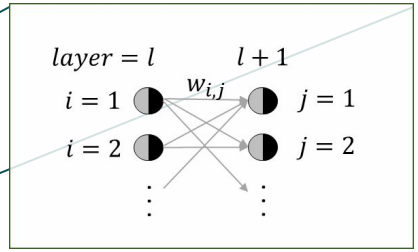
- The ***attribution*** of -5.5 to the feature x_i is $\boldsymbol{w_i x_i}$.

heatmap



LRP: Layerwise Relevance Propagation (attribution)





Relevance of neuron i in layer l

Preactivation from i to j

$$\text{LRP}_z: \quad R_i^l = \sum_j \frac{w_{i,j}x_i}{\sum_i w_{i,j}x_i + w_0} R_j^{l+1}$$

Other popular LRP rules :

Any questions?

$$\text{LRP}_{\alpha\beta}: \quad R_i^l = \sum_j \left(\alpha \frac{(w_{i,j}x_i)^+}{\sum_i (w_{i,j}x_i)^+ + w_0{}^+} + \beta \frac{(w_{i,j}x_i)^-}{\sum_i (w_{i,j}x_i)^- + w_0{}^-} \right) R_j^{l+1}$$

LRP_{comp} : a combination of LRP_z and $\text{LRP}_{\alpha\beta}$

From *Bach et al., (2015)*

SHAP: SHapley Additive exPlanations (attribution)

Consider the general class of explanation models:

$$f(\mathbf{x}) = R_0 + \sum_i R_i \qquad (1)$$

network — input — attribution to feature i

Any XAI method that can be represented as in Eq. (1), we will say it is an ***additive feature attribution method***.

➡ LRP and other popular XAI methods (e.g., LIME, DeepLIFT) are essentially different solutions to Eq. (1).

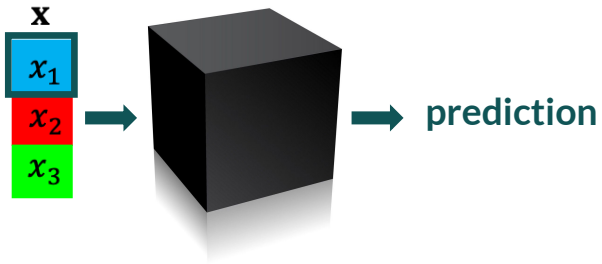
Theorem:
The only solution to Eq. (1) that satisfies the desirable properties of local accuracy, missingness, consistency emerges when R_i are equal to the Shapley values.

$$R_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right].$$

From *Lundberg and Lee (2017)*

SHAP (attribution)

Let's say I want to calculate the Shapley value of x_1



Step 1: Consider all the subsets of the input that contain x_1.
Step 2: For all sets in step 1, calculate the importance of x_1, as the difference between the model output when x_1 is present and when it is missing.
Step 3: The Shapley value of x_1 is the weighted average of the quantities calculated in step 2.

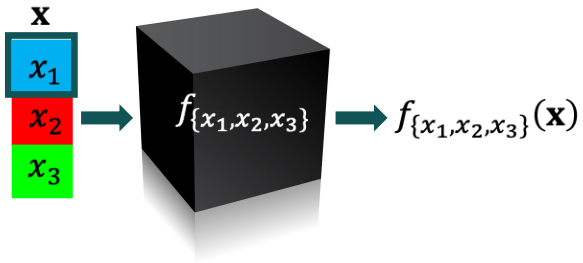
SHAP (attribution)

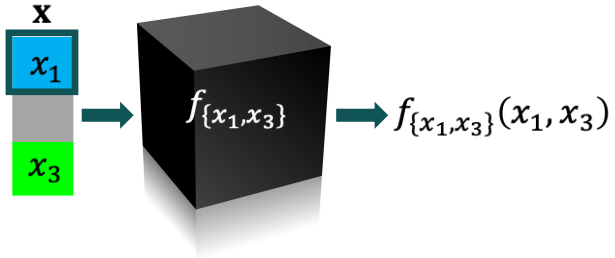
Let's say I want to calculate the Shapley value of x_1

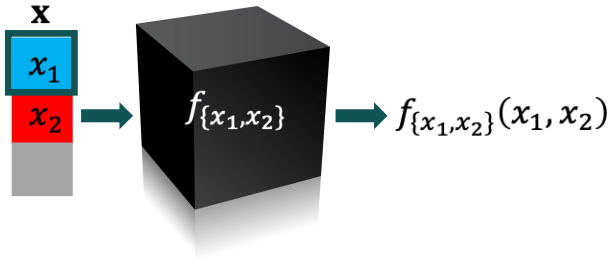
Step 1: Consider all the subsets of the input that contain x_1.

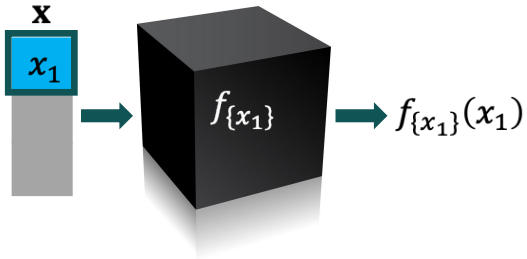
Step 2: For all sets in step 1, calculate the importance of x_1, as the difference between the model output when x_1 is present and when it is missing.

Step 3: The Shapley value of x_1 is the weighted average of the quantities calculated in step 2.









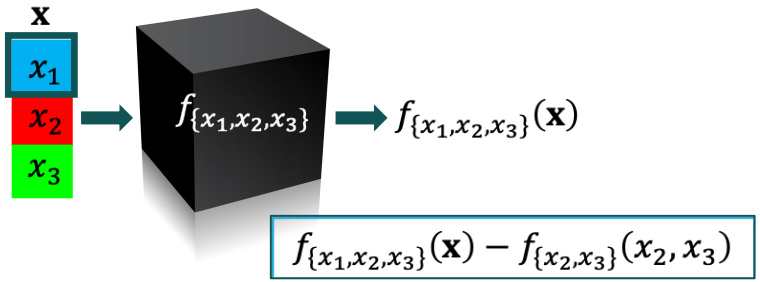
SHAP (attribution)

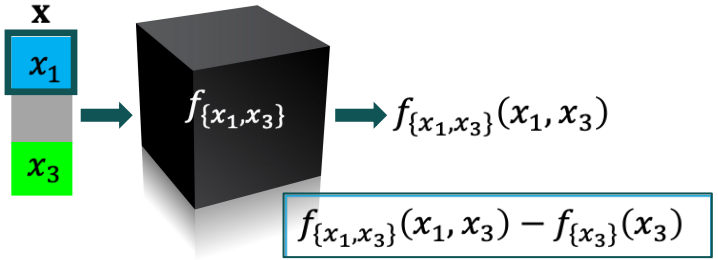
Let's say I want to calculate the Shapley value of x_1

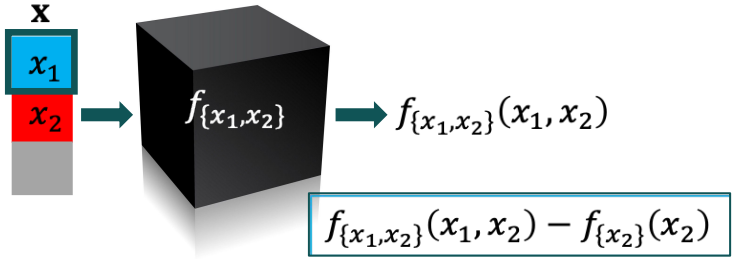
Step 1: Consider all the subsets of the input that contain x_1.

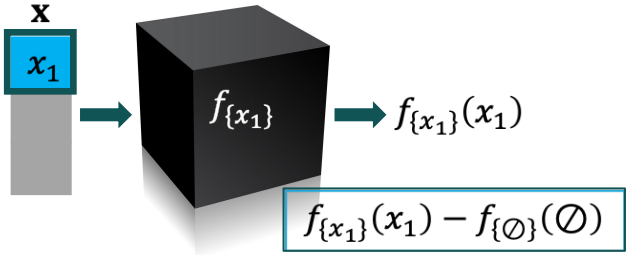
Step 2: For all sets in step 1, calculate the importance of x_1, as the difference between the model output when x_1 is present and when it is missing.

Step 3: The Shapley value of x_1 is the weighted average of the quantities calculated in step 2.









SHAP (attribution)

Step 1: Consider all the subsets of the input that contain x_1.
Step 2: For all sets in step 1, calculate the importance of x_1, as the difference between the model output when x_1 is present and when it is missing.
Step 3: The Shapley value of x_1 is the weighted average of the quantities calculated in step 2.

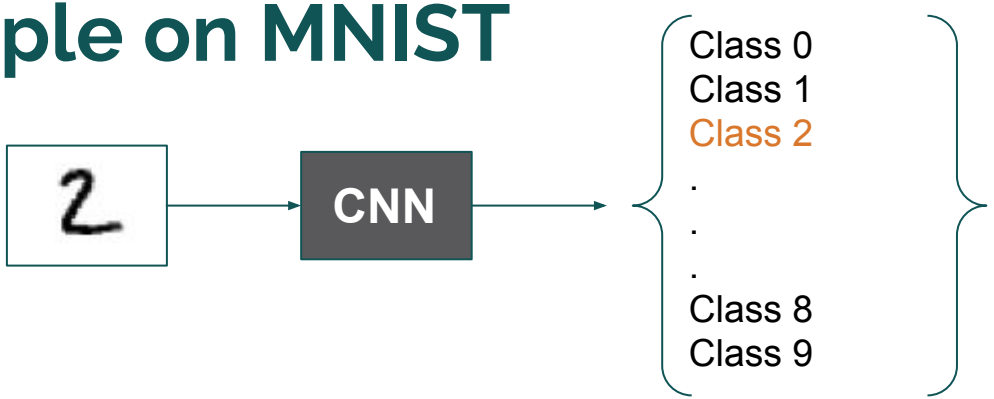
$$R_1 = \frac{1}{3}\left(f_{\{x_1,x_2,x_3\}}(\mathbf{x}) - f_{\{x_2,x_3\}}(x_2,x_3)\right) + \frac{1}{6}\left(f_{\{x_1,x_3\}}(x_1,x_3) - f_{\{x_3\}}(x_3)\right)$$
$$+ \frac{1}{6}\left(f_{\{x_1,x_2\}}(x_1,x_2) - f_{\{x_2\}}(x_2)\right) + \frac{1}{3}\left(f_{\{x_1\}}(x_1) - f_{\{\emptyset\}}(\emptyset)\right)$$

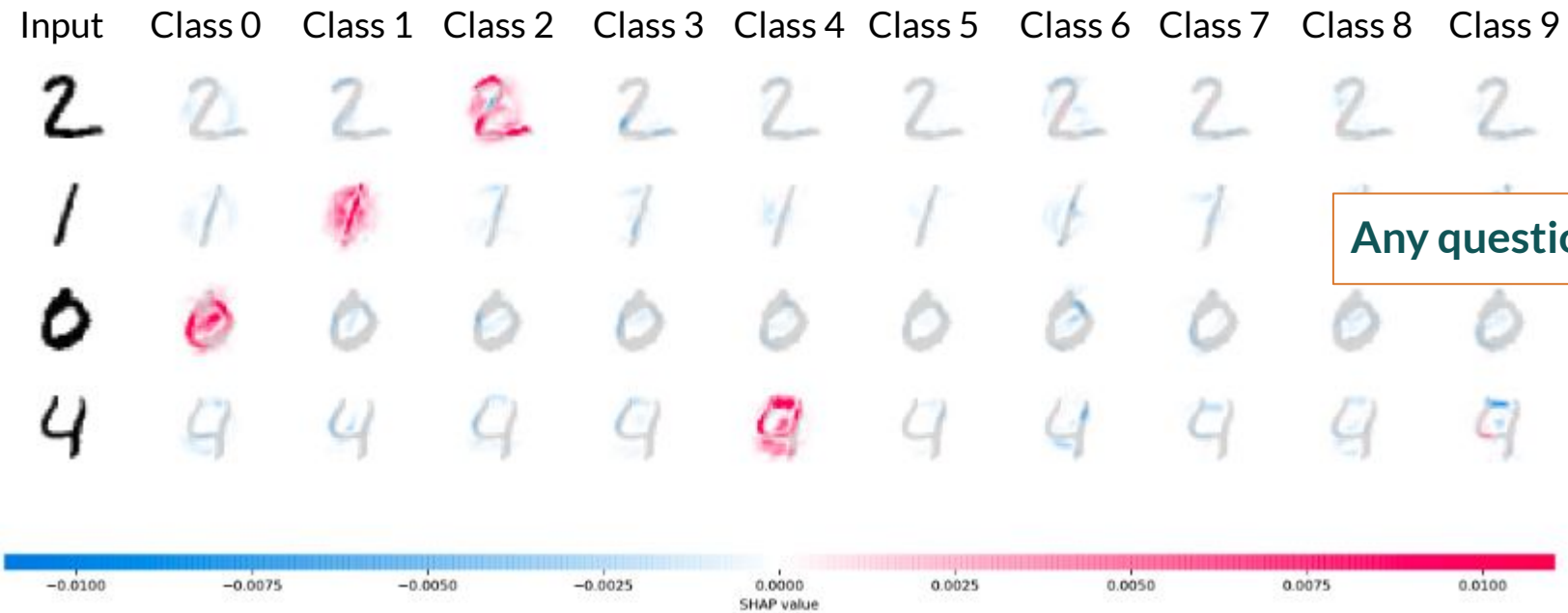
This was only for x_1. The same needs to be done for x_2 and x_3 to get a "heatmap".

For problems with high dimensions (e.g., more than 10 features), the number of times one would need to retrain and evaluate the model in order to calculate the shapley values is extremely high. Computationally, this is not efficient, so the SHAP method uses an approximate algorithm (Deep SHAP), specifically designed for deep neural networks. Deep SHAP is similar to LRP, except that instead of propagating the relevance, it propagate the Shapley values.

Also, there is no model retraining in Deep SHAP. When a specific feature (or neuron output) needs to be withheld (i.e., to be considered missing), it is replaced with a background value, which is usually the average value in the training dataset.

SHAP Example on MNIST





Benchmarking XAI

The need for *objectivity* in assessing XAI

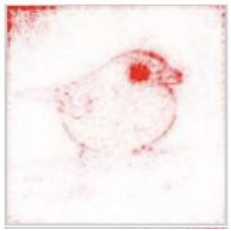
Which input features were important for this classification?

Original Image

Guided BackProp

XAI method

Junco Bird



XAI heatmap

Issues : 1) No ground truth to assess the estimated explanations.

➡ **Debagging the phrase: "*The explanation looks reasonable*"**

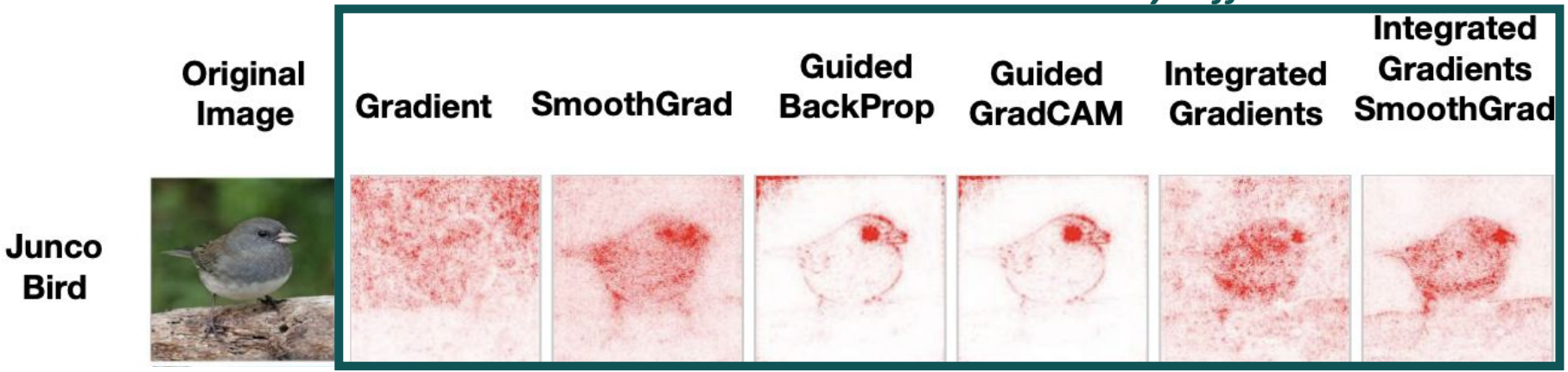
➡ **Remember: The human perception of the explanation alone is NOT a solid criterion for its trustworthiness.**

From *Adebayo et al. (2020)*

The need for *objectivity* in assessing XAI

Which input features were important for this classification?

Many Different XAI methods



Issues : 1) No ground truth to assess the estimated explanations.
2) Different methods provide different answers.

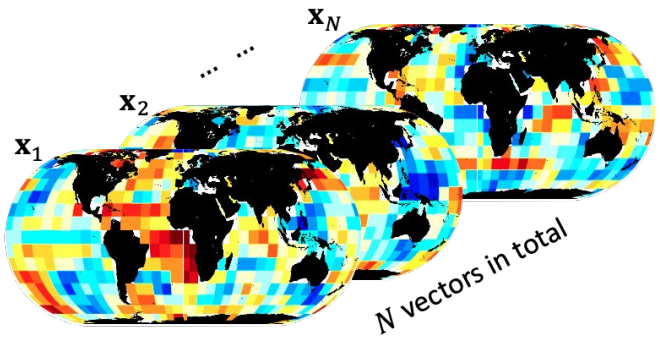
➡ **This is problematic: The uncertainty on how the network decides, leads to limited trust when using neural networks in environmental problems.**

➡ **We need objective frameworks to rigorously assess XAI methods and gain insights about relative strengths and weaknesses.**

From *Adebayo et al. (2020)*

Attribution benchmarks for XAI

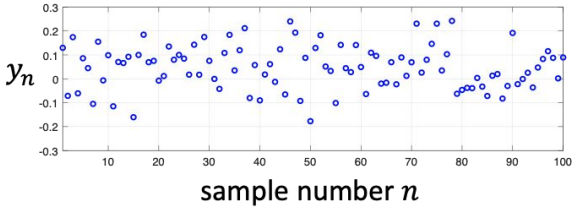
Step 1: Generate N samples of $\mathbf{X} \in \mathbb{R}^d$ from a MVN



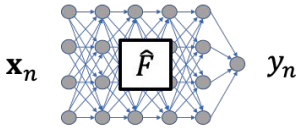
$y_n = F(\mathbf{x}_n)$

Known $F: \mathbb{R}^d \rightarrow \mathbb{R}$

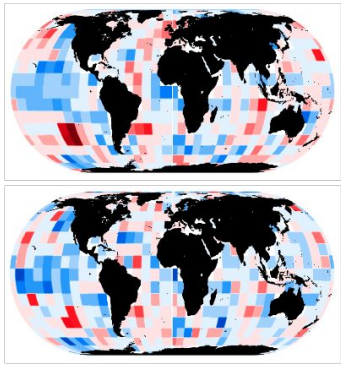
Step 2: Use a known function F that maps each vector $\mathbf{x}_n$ into a scalar y_n



Step 3: Pretend function F is not known and train a NN using inputs $\mathbf{x}_n$ and outputs y_n



Step 4: Use XAI methods to explain the NN and compare with the ground truth from F



F: ground truth

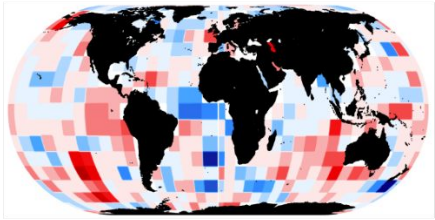
$\hat{F}$: from XAI method

From *Mamalakis et al. (2021)*

Regression Benchmark - Fully Connected Network

y_n : 0.0660
NN prediction: 0.0802

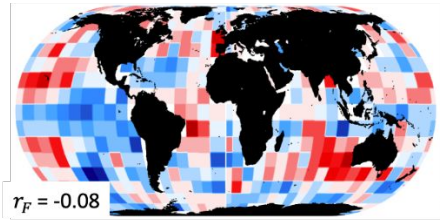
Ground Truth of Attribution for F



Positive contribution
Negative contribution

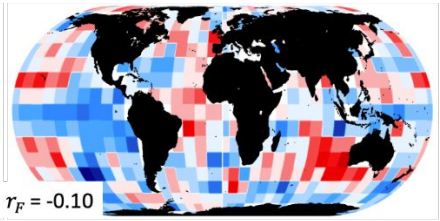
Any questions?

Gradient



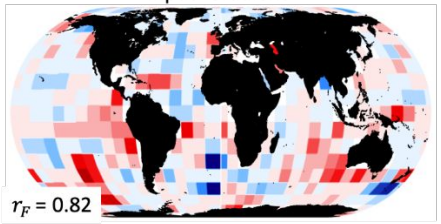
r_F = -0.08

Smooth Gradient



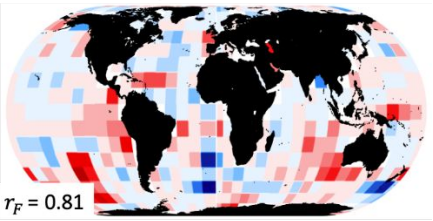
r_F = -0.10

Input*Gradient



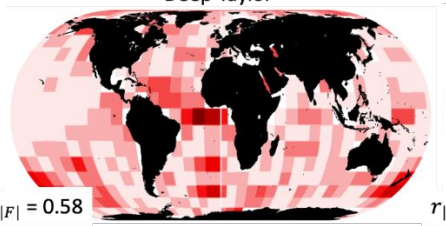
r_F = 0.82

Integrated Gradients



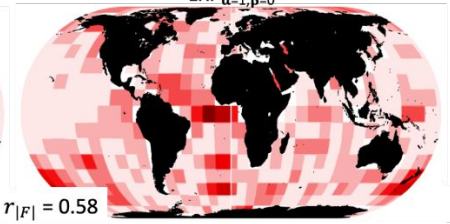
r_F = 0.81

Deep Taylor



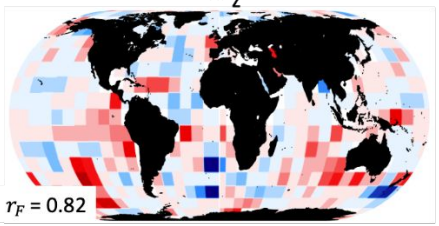
$r_{|F|}$ = 0.58

$LRP_{\alpha=1;\beta=0}$



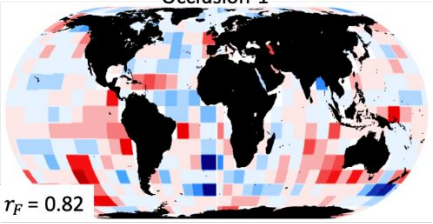
$r_{|F|}$ = 0.58

LRP_z



r_F = 0.82

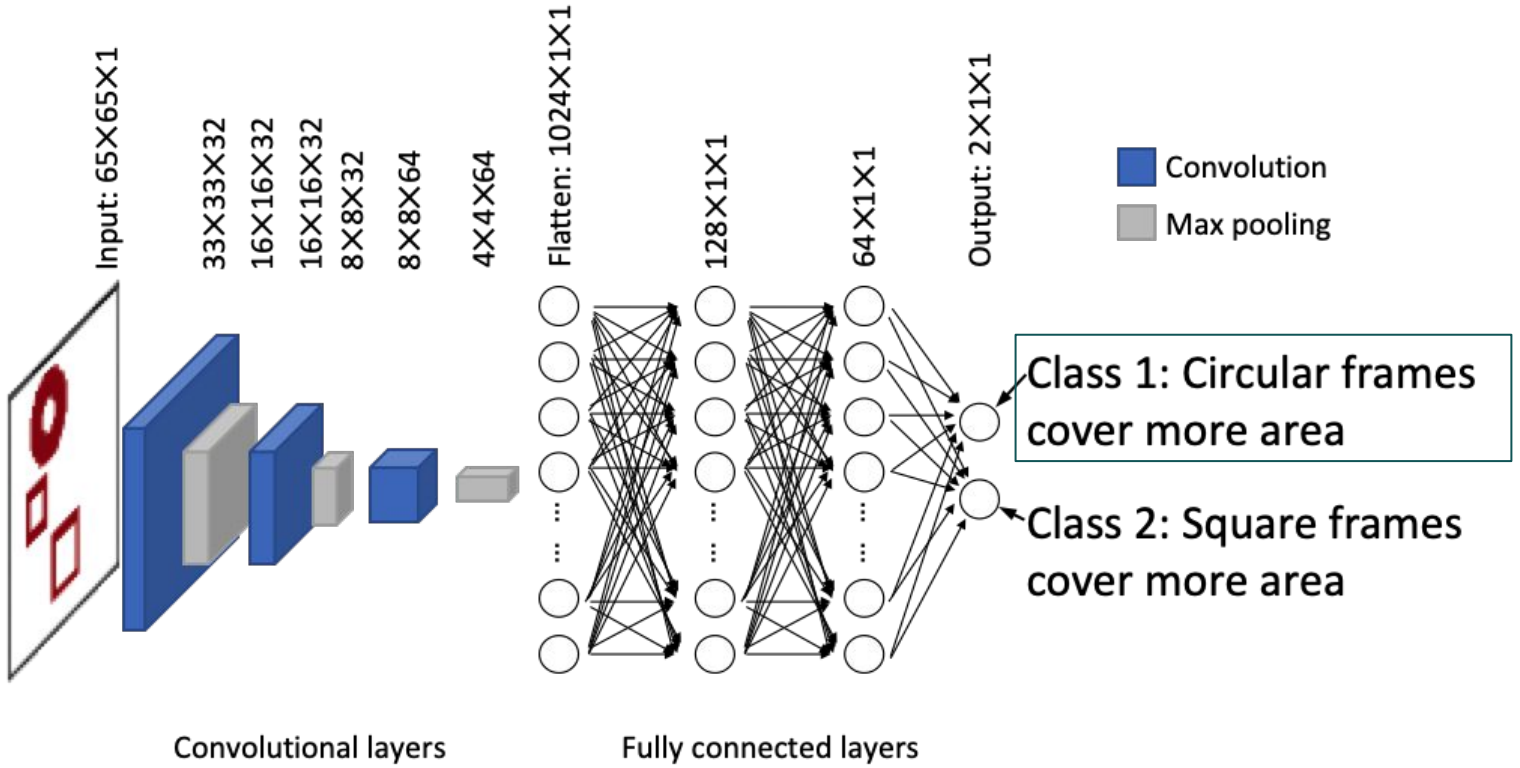
Occlusion-1



r_F = 0.82

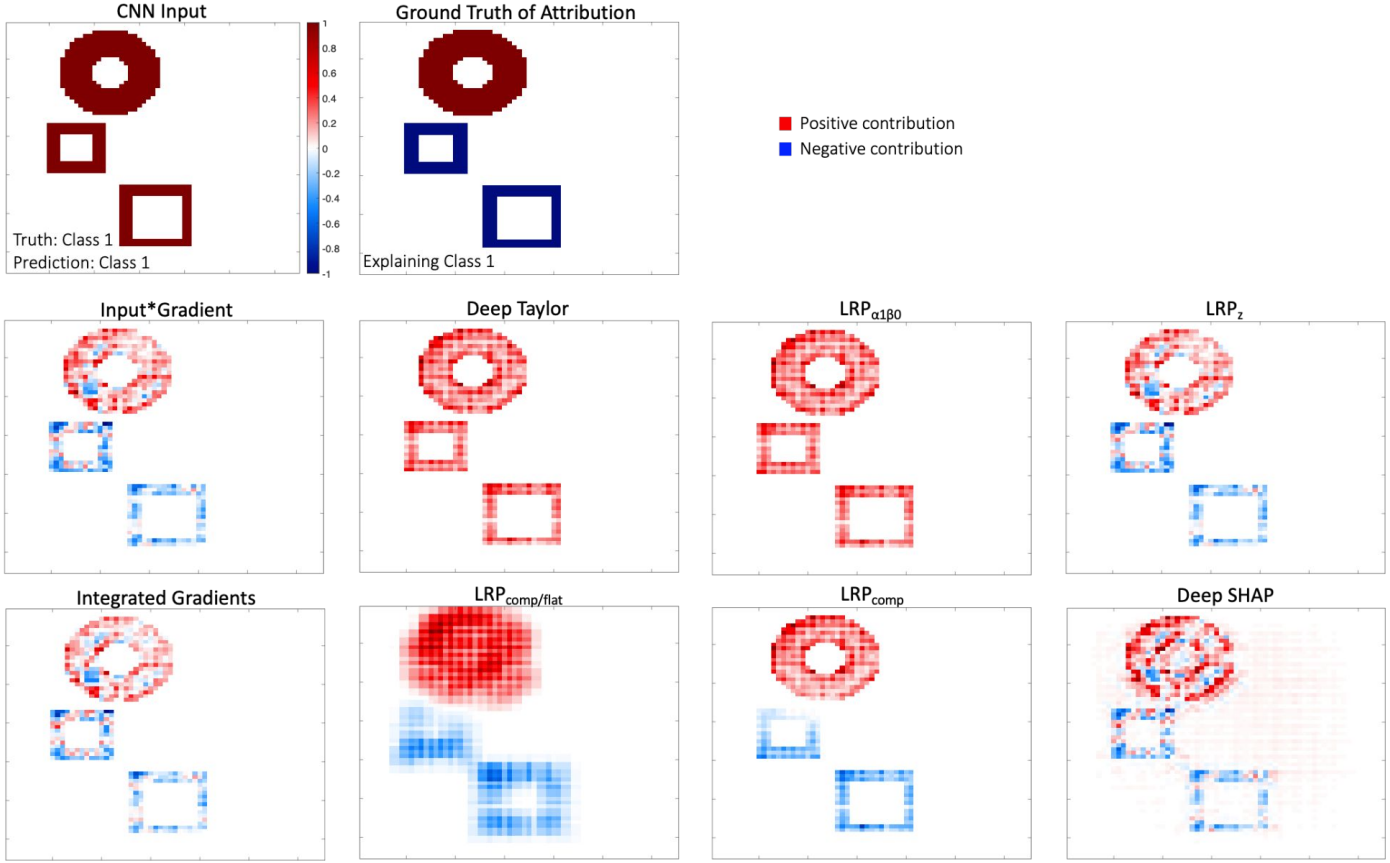
From *Mamalakis et al. (2021)*

Classification Benchmark - Convolutional Network



From *Mamalakis et al. (2022)*

Classification Benchmark - Convolutional Network



From *Mamalakis et al. (2022)*

Best practices of XAI

Table 1. Summary of XAI methods considered in this study. Practical strengths (✓) and weaknesses (×) of each method are also reported.

XAI method		Brief summary of the method	Desired property for CNN applications as explored in this study			Extra comments/insights
			disentangles the sign of relevance	insensitive to gradient shattering	not ignorant to zero input	
Gradient (Simonyan et al., 2014)		Calculates the first partial derivative of the model output with respect to the input. (sensitivity)	✓	×	✓	Estimates the sensitivity of the output to the input, which is not the same as the attribution; see Appendix B
Smooth Gradient (Smilkov et al., 2017)		Calculates the average gradient across many perturbed inputs. (sensitivity)	✓	×	✓	
Input*Gradient (Shrikumar et al., 2017)		Multiplies the input with the gradient. (attribution)	✓	×	×	
Integrated Gradients (Sundararajan et al., 2017)		Multiplies the average gradient along the straight line between the input point and a reference point with the corresponding distance between the two points. (attribution)	✓	×	×	
LRP	**α1β0** (Bach et al., 2015)	Layer-wise back propagation of each neuron's relevance based on the α1β0-rule. (attribution)	×	✓	×	Considers only positive preactivations
	z (Bach et al., 2015)	Layer-wise back propagation of each neuron's relevance based on the z-rule (attribution)	✓	×	×	Equivalent to Input*Gradient for networks using ReLU activations
	comp (Kohlbrenner et al., 2020)	Layer-wise back propagation of each neuron's relevance by combining the α1β0-rule and the z-rule. (attribution)	✓	✓	×	Combines the strengths of LRPz and $LRP_{\alpha1\beta0}$
	comp/flat (Kohlbrenner et al., 2020)	Layer-wise back propagation of each neuron's relevance by combining the α1β0-rule, the z-rule and the flat rule. (attribution)	✓	✓	✓	Provides a coarser picture of attribution; not suitable if local accuracy necessary
Deep Taylor (Montavon et al., 2017)		Applies Taylor decomposition of the relevance function for each neuron recursively. (attribution)	×	✓	×	Equivalent to $LRP_{\alpha1\beta0}$ for networks using ReLU activations; not defined for negative predictions
PatternNet (Kindermans et al., 2017a)		Calculates the signal in the input for each neuron recursively. (signal)	×	✓	✓	Estimates the signal (not the same as the attribution)
PatternAttribution (Kindermans et al., 2017a)		Calculates the attribution in the direction of the signal for each neuron recursively. (attribution)	×	✓	✓	
Deep SHAP (Lundberg and Lee, 2017)		Approximates Shapley values for each neuron recursively (attribution)	✓	×	✓	Based on well-founded theory; computationally expensive

From *Mamalakis et al. (2022)*

Summary

Key take home messages

- **XAI methods show potential to be a game-changer in how we predict/detect patterns in Earth Sciences. We can use these tools to calibrate model trust, fine-tune models and learn new science.**

- **We briefly went through some popular XAI methods in Earth Sciences: Gradient, Input*Gradient, Layer-wise Relevance Propagation (LRP) and SHapley Additive exPlanations (SHAP).**

- **Given the plethora and the diversity of methods out there, the lack of a ground truth to assess their fidelity has the risk of allowing subjective assessment, and cherry-picking certain methods. It is important to introduce *objectivity* in XAI assessment and shed light to relative strengths and weaknesses.**

- ***Engagement* of attribution benchmarks may lead to a more *cautious* and *successful* implementation of XAI methods.**

References

J. Adebayo et al. (2020) "Sanity checks for saliency maps," arXiv preprint, https://arxiv.org/abs/1810.03292.

Bach, et al. (2015) "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation", *PLOS ONE*, https://doi.org/10.1371/journal.pone.0130140

Lundberg, S. M. and S. I. Lee (2017) "A unified approach to interpreting model predictions," *Proc. Adv. Neural Inf. Process. Syst.*, pp. 4768-4777.

Mamalakis, A., I. Ebert-Uphoff and E.A. Barnes (2021) "Neural Network Attribution Methods for Problems in Geoscience: A Novel Synthetic Benchmark Dataset", arXiv preprint, https://arxiv.org/abs/2103.10005, accepted in *Environmental Data Science*.

Mamalakis, A., I. Ebert-Uphoff, E.A. Barnes "Explainable Artificial Intelligence in Meteorology and Climate Science: Model fine-tuning, calibrating trust and learning new science," in *Beyond explainable Artificial Intelligence* by Holzinger et al. (Editors), Springer Lecture Notes on Artificial Intelligence, open access at: *https://link.springer.com/chapter/10.1007/978-3-031-04083-2_16*

Mamalakis, A., E.A. Barnes, I. Ebert-Uphoff (2022) "Investigating the fidelity of explainable artificial intelligence methods for applications of convolutional neural networks in geoscience", arXiv preprint, https://arxiv.org/abs/2202.03407, accepted in *Artificial Intelligence for the Earth Systems*.

Samek, et al. (2021), "Explaining Deep Neural Networks and Beyond: A review of Methods and Applications", in *Proceedings of the IEEE*, vol. 109, no. 3, pp. 247-278, March 2021, doi: 10.1109/JPROC.2021.3060483

Other Resources

INNVESTIGATE	https://github.com/albermax/innvestigate https://innvestigate.readthedocs.io/en/latest/modules/analyzer.html
SHAP	https://github.com/slundberg/shap
My GitHub page	https://github.com/amamalak

Simple Uncertainty Quantification

Can a network predict how right it is?

Uncertainty Quantification

For prediction problems in the Earth sciences, we usually want some sort of uncertainty or likelihood associated with the prediction.

E.g. in forecasting we generate ensembles

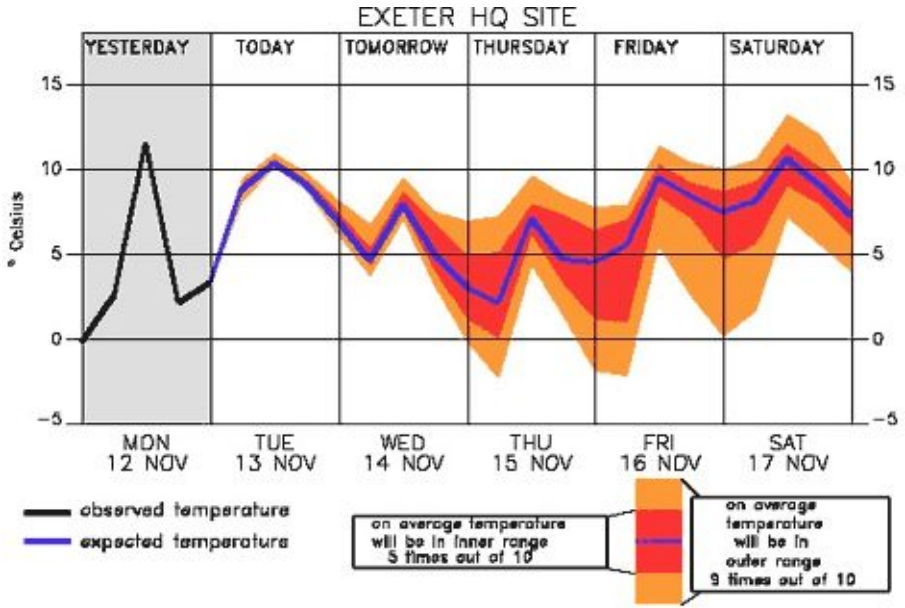


Image from https://www.metoffice.gov.uk/research/weather/ensemble-forecasting/decision-making

Uncertainty Quantification

For prediction problems in the Earth sciences, we usually want some sort of uncertainty or likelihood associated with the prediction.

E.g. in forecasting we generate ensembles

This allows us to quantify the likelihood of an event, as well as build trust in our models

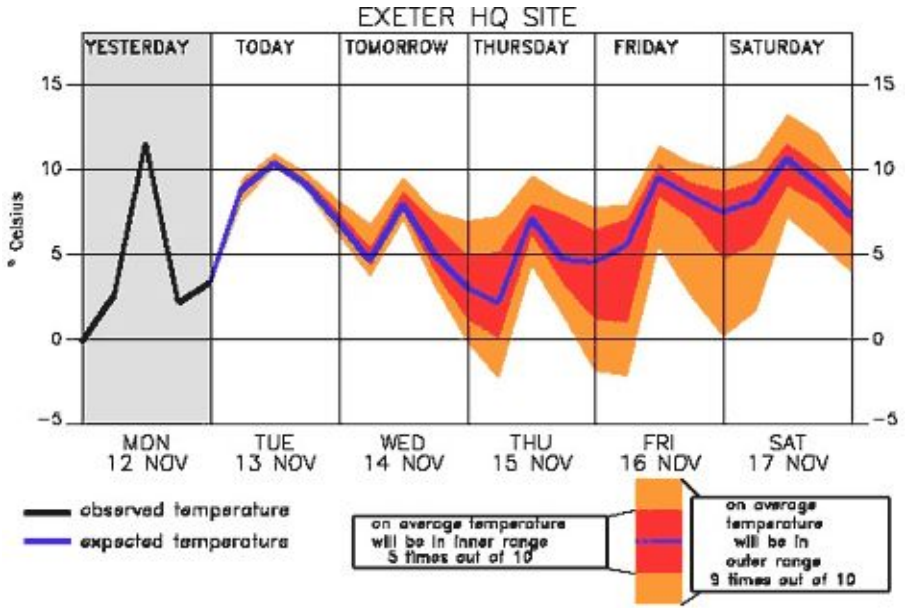
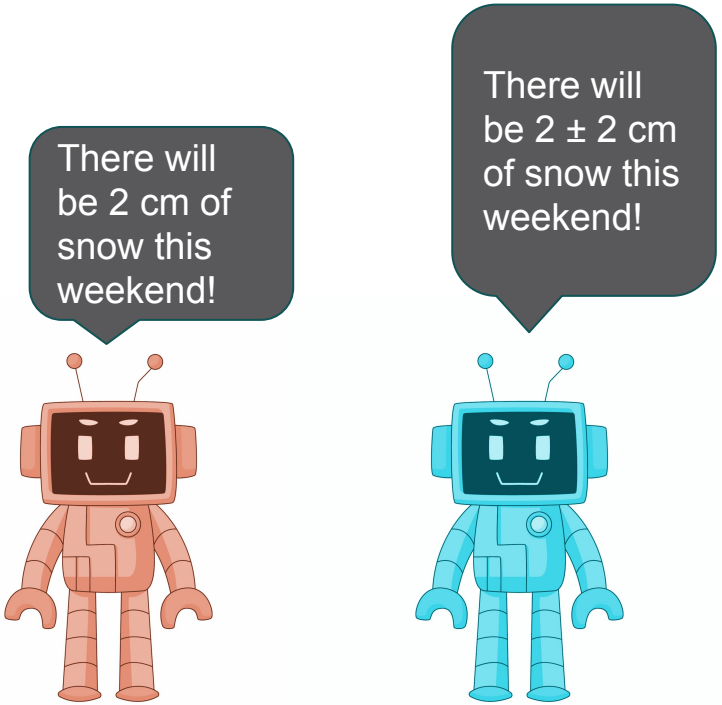


Image from https://www.metoffice.gov.uk/research/weather/ensemble-forecasting/decision-making

Uncertainty Quantification in ML

We want to build uncertainty quantification into machine learning models for the same reasons:

- Quantify the likelihood of an event
- Build trust in our models

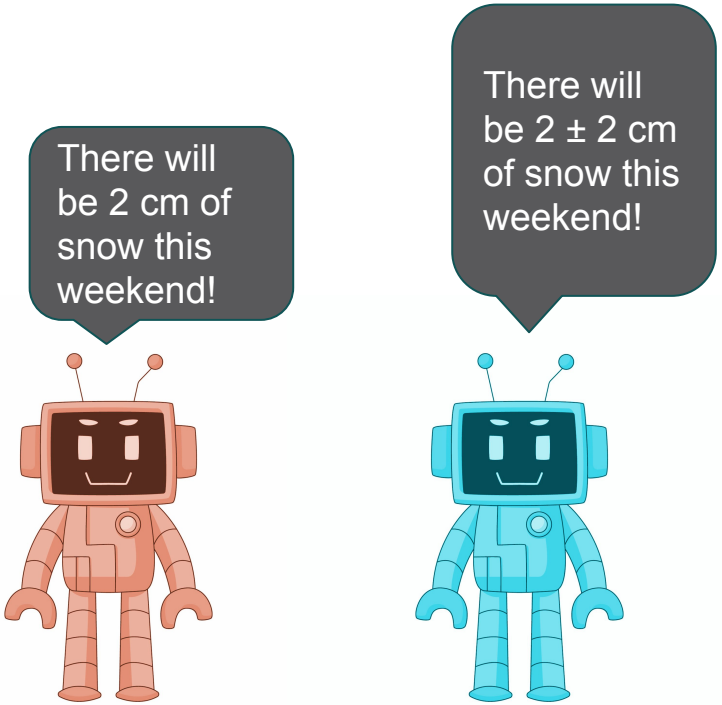


Uncertainty Quantification in ML

We want to build uncertainty quantification into machine learning models for the same reasons:

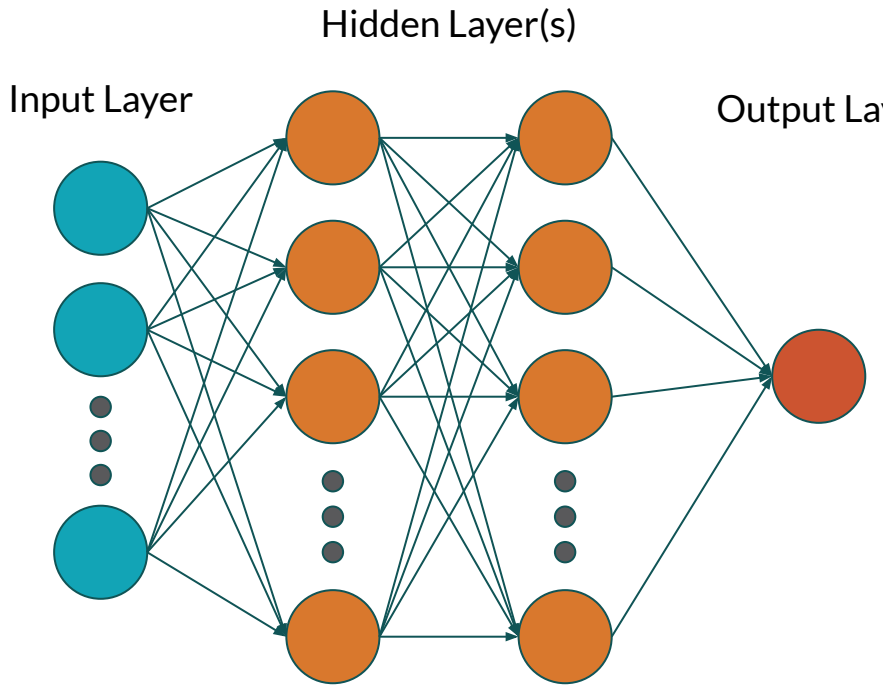
- Quantify the likelihood of an event
- Build trust in our models

Here we will go through one way of adding uncertainty to NNs but there are all sorts of other methods you may encounter



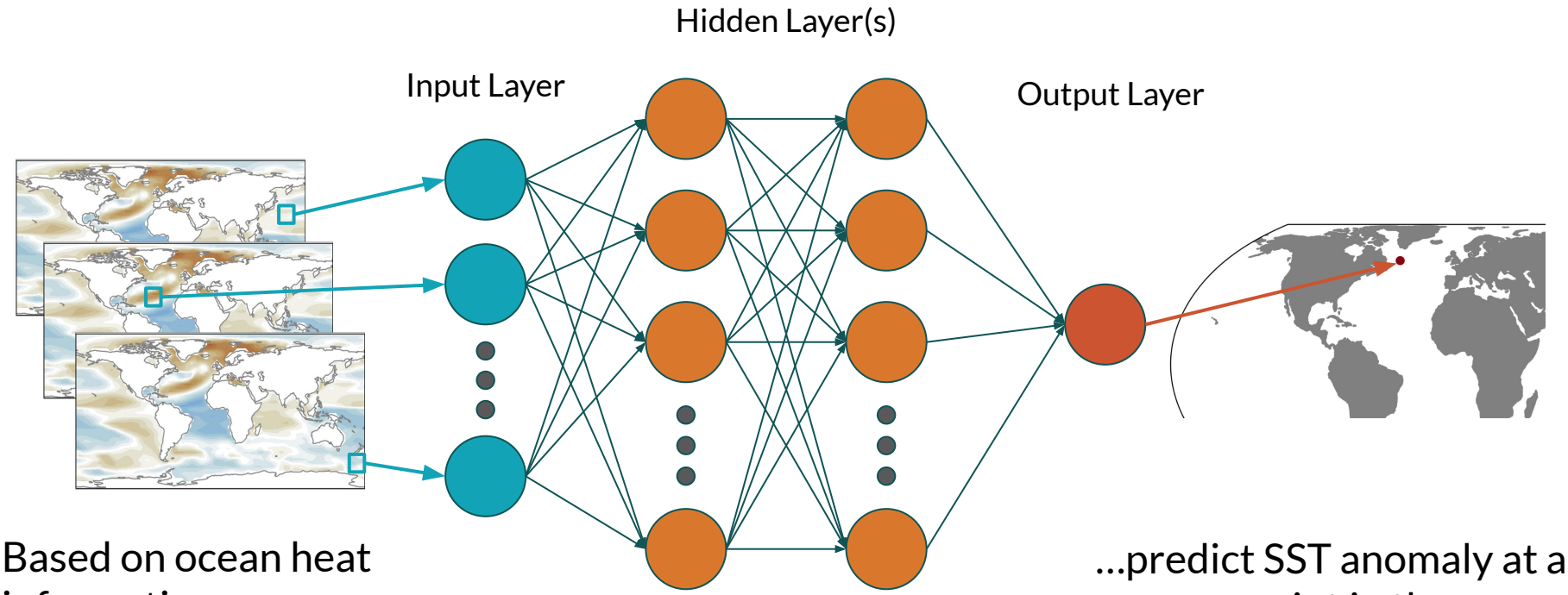
Consider a simple regression task…

Predictands, x_i



Prediction, a specific number, y

Consider a simple regression task…



Based on ocean heat information…

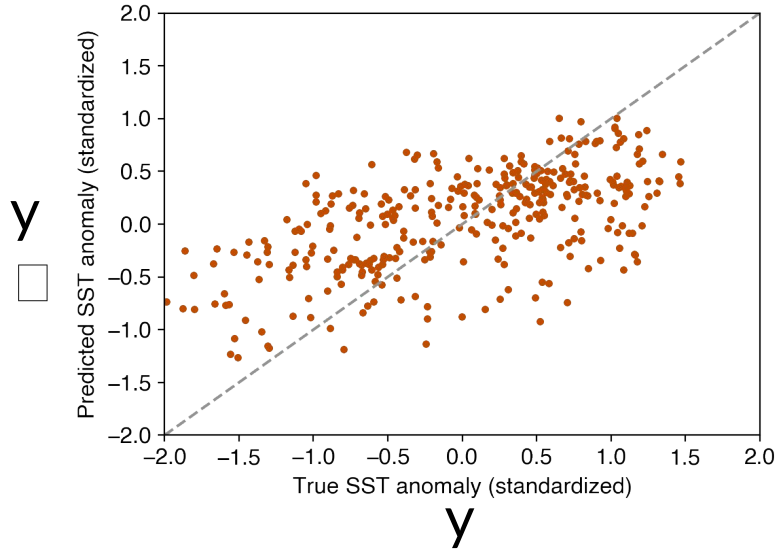
…predict SST anomaly at a point in the ocean

Consider a simple regression task…

In a regression model, we want to train a network to minimize the error between its predictions and the truth, Loss function could be mean absolute error, or mean squared error, e.g.

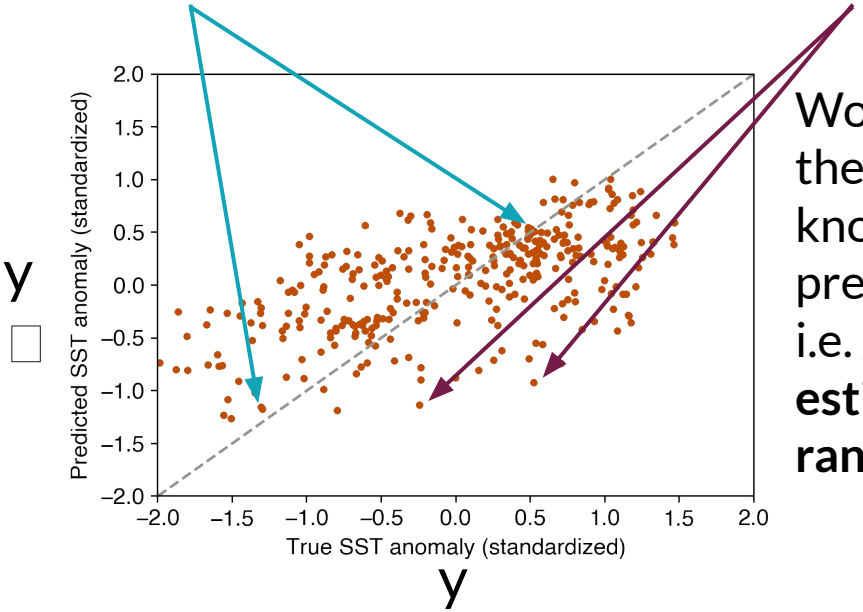
$$\mathcal{L} = |y\square - y|$$

And we can make a scatter plot of the truth (y) vs predictions (y□)



Consider a simple regression task…

Obviously some predictions are better than others

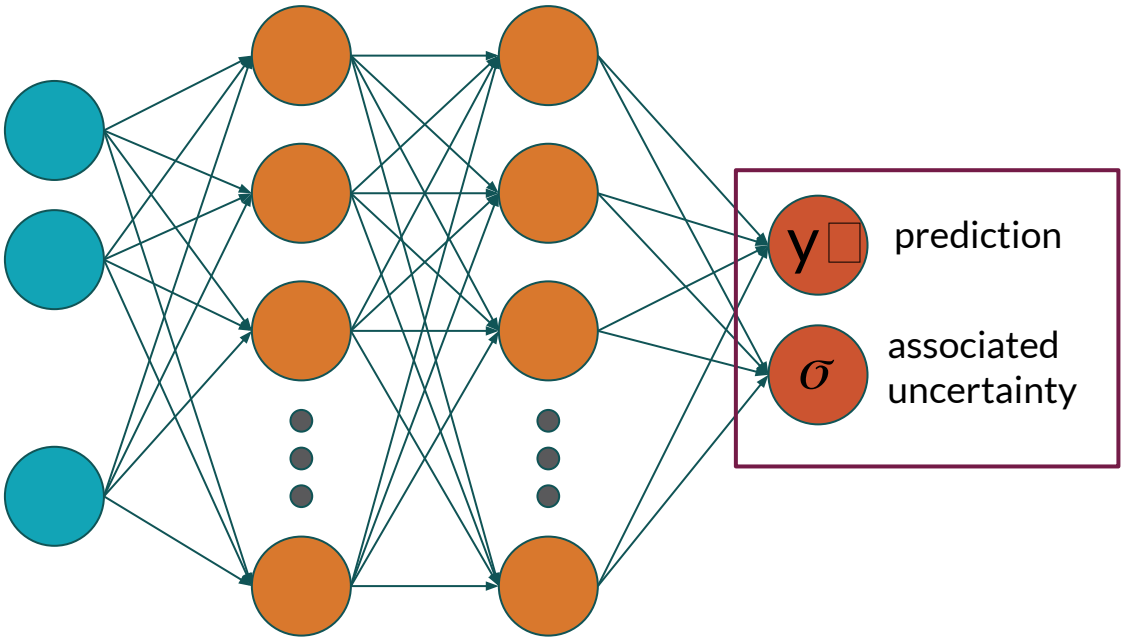


Wouldn't it be great if there was some way of knowing how good a prediction is
i.e. get the **network to estimate an uncertainty range**

Adding Uncertainty to Regression Tasks

Rather than the network outputting a single number as its estimate….

We want the network to output an estimate and an uncertainty range

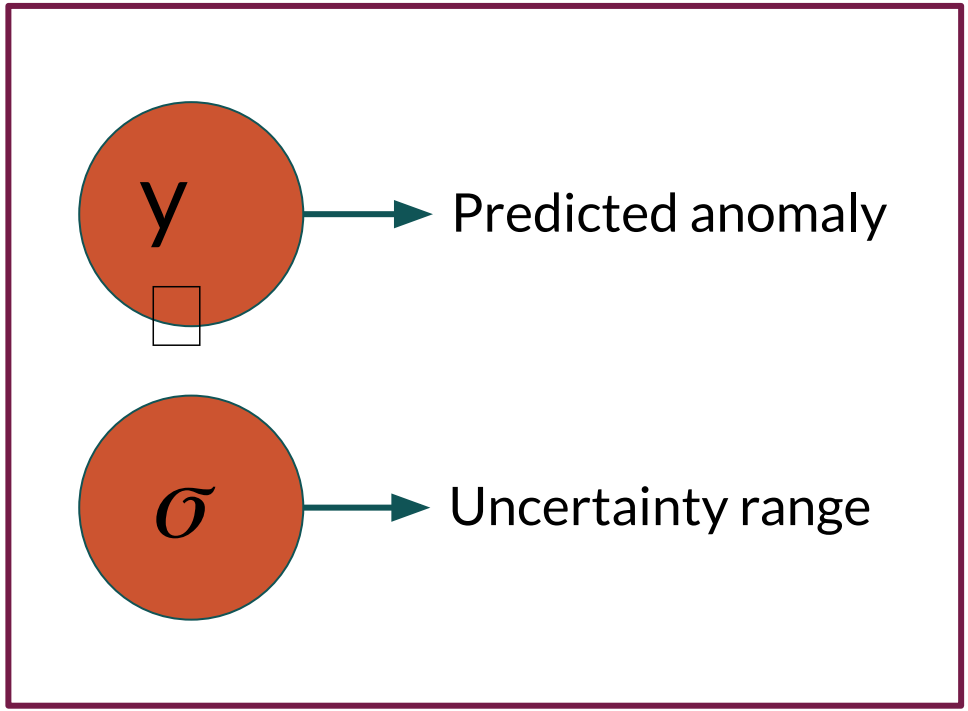


Adding Uncertainty to Regression Tasks

Rather than the network outputting a single number as its estimate….

We want the network to output an estimate and an uncertainty range

We train the neural network to predict conditional distributions, i.e. **each prediction is the parameters of a probability distribution**



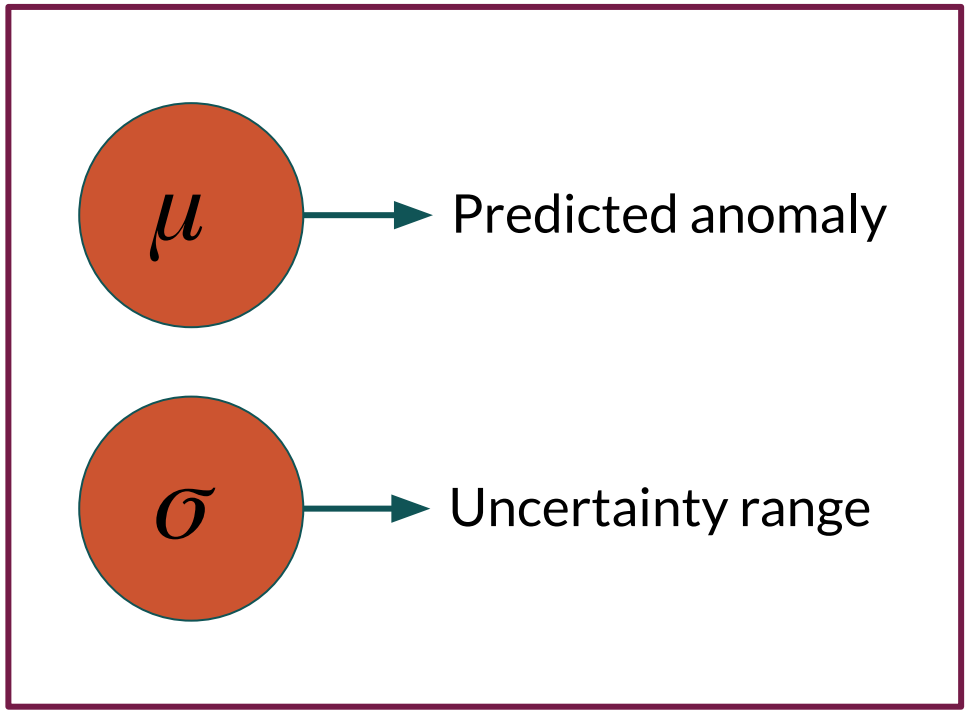
Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

Adding Uncertainty to Regression Tasks

Rather than the network outputting a single number as its estimate….

We want the network to output an estimate and an uncertainty range

The simplest version of this is predicting a Gaussian – so predicting a mean (μ) and standard deviation (σ)



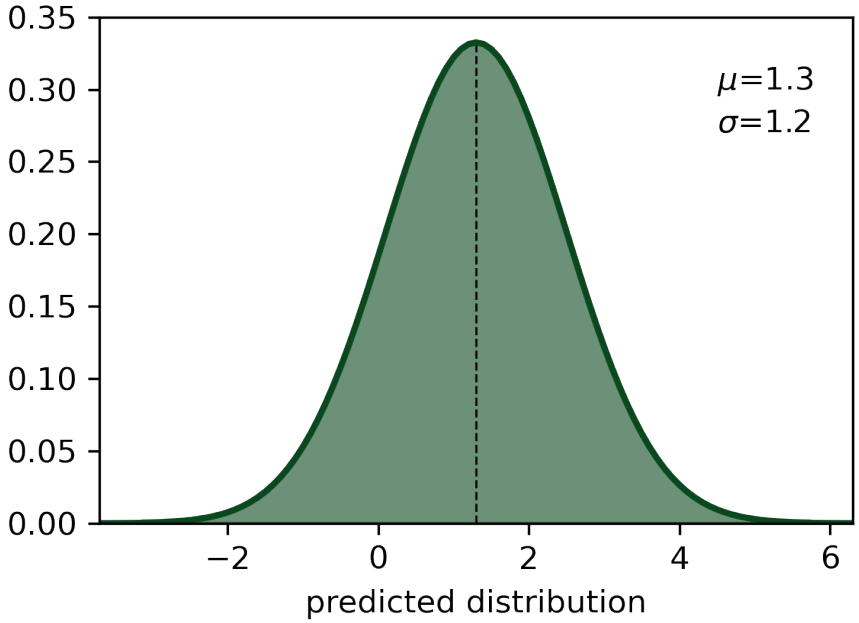
Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

Adding Uncertainty to Regression Tasks

μ → Predicted anomaly
e.g. 1.3

σ → Uncertainty
e.g. 1.2

Predicted μ and σ are used to construct a normal distribution



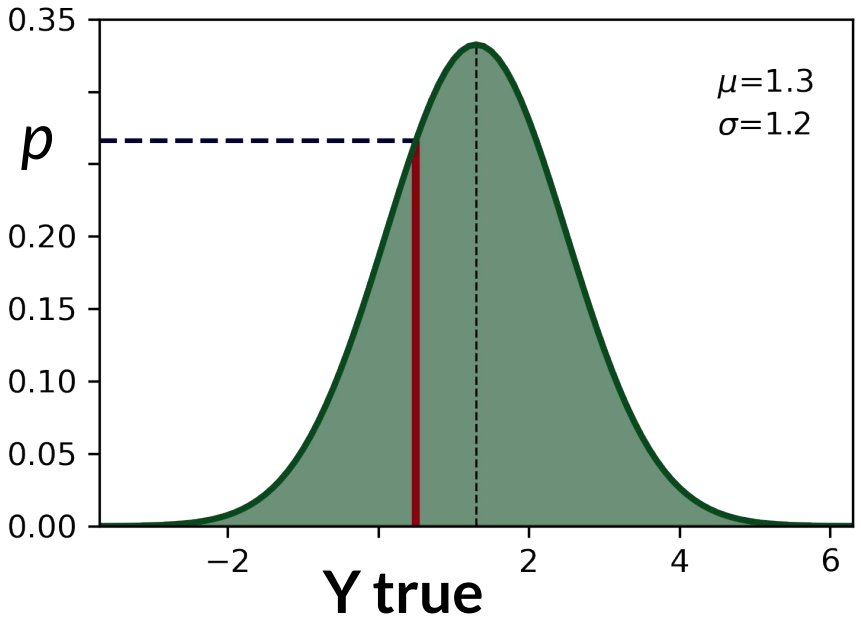
Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

Adding Uncertainty to Regression Tasks

μ → Predicted anomaly
e.g. 1.3

σ → Uncertainty
e.g. 1.2

Evaluate *p*, probability distribution function at the true value of the anomaly



Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

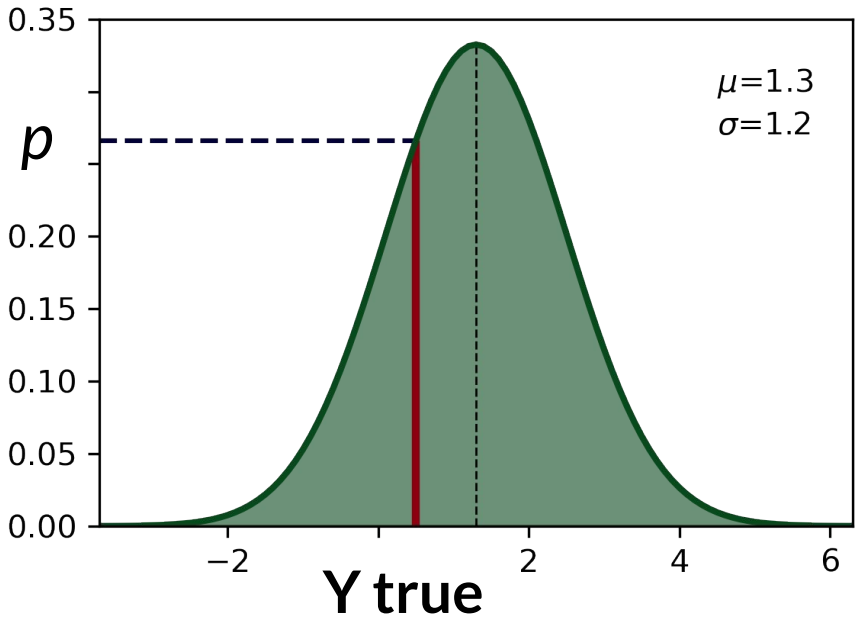
Adding Uncertainty to Regression Tasks

μ → Predicted anomaly
e.g. 1.3

σ → Uncertainty
e.g. 1.2

Loss function is defined as

$$\mathcal{L} = -\log(p)$$



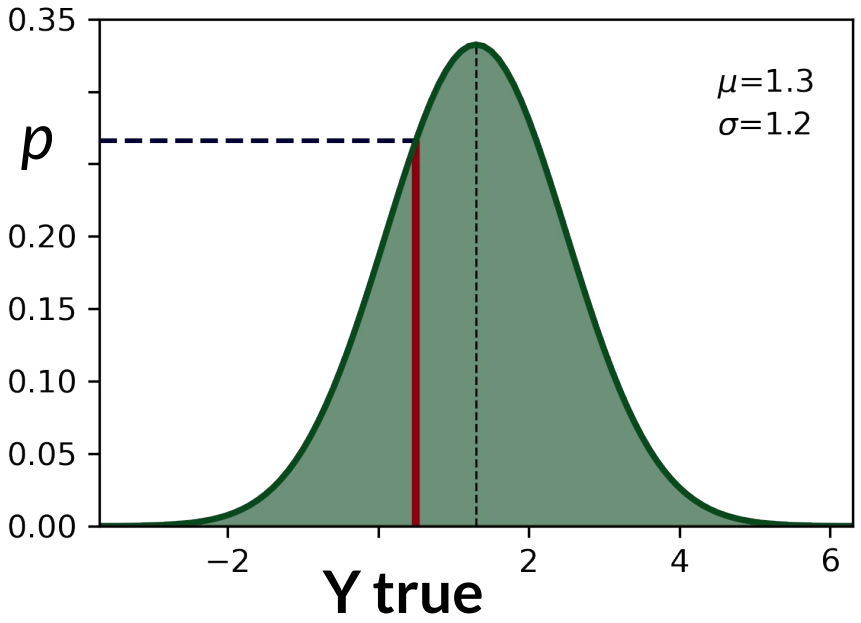
Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

Adding Uncertainty to Regression Tasks

Loss function is defined as

$$\mathcal{L} = -\log(p)$$

To minimize loss, network attempts to maximise p



Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

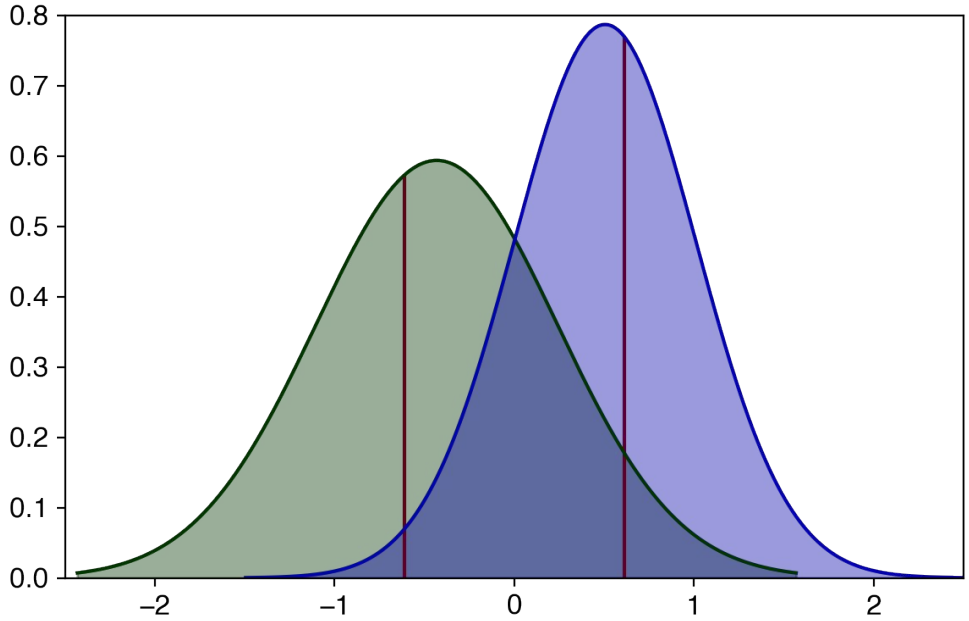
Adding Uncertainty to Regression Tasks

Loss function is defined as

$$\mathcal{L} = -\log(p)$$

To minimize loss, network attempts to maximise p

The network must hence learn to make good anomaly predictions, but also reasonable uncertainty estimates

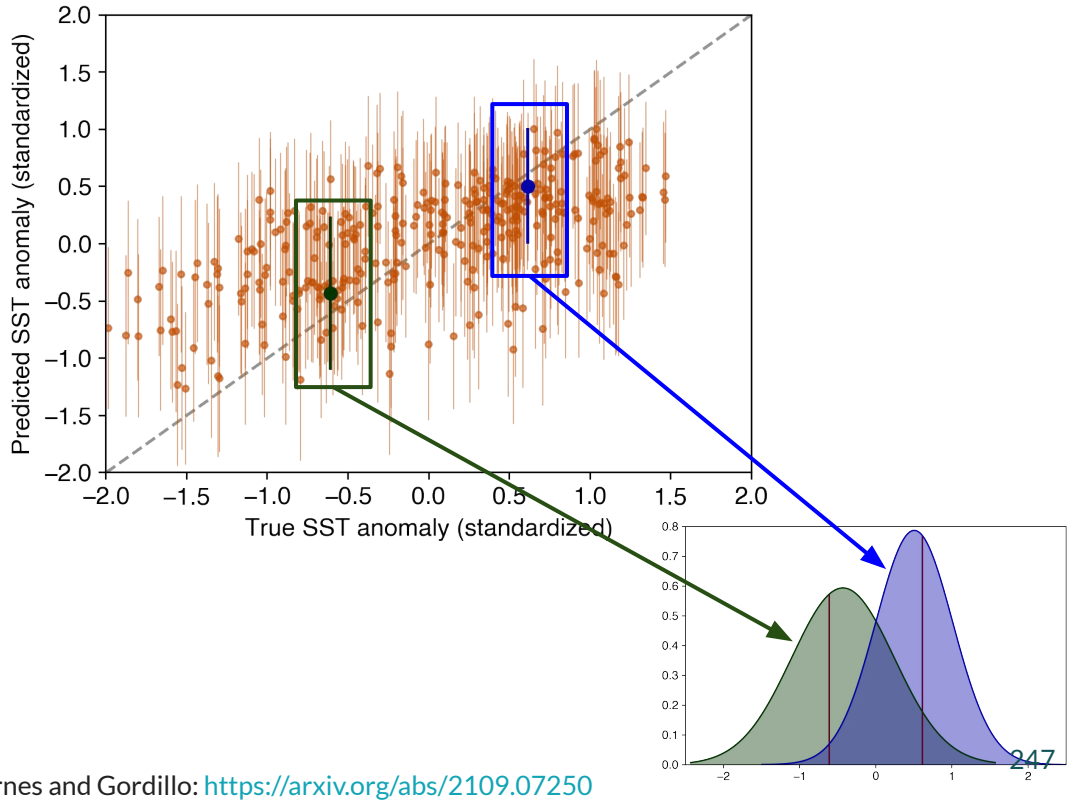


Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

Adding Uncertainty to Regression Tasks

Now predictions have an uncertainty range

Here we plot the 1σ range associated with each prediction



Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

A Fun and Important Point!

Loss function:

$$\mathcal{L} = -\log(p)$$

This is simply a log likelihood!
The predicted distribution does not have to be a Gaussian!!

Consider for example the sinh-arcsinh (SHASH) normal distribution…

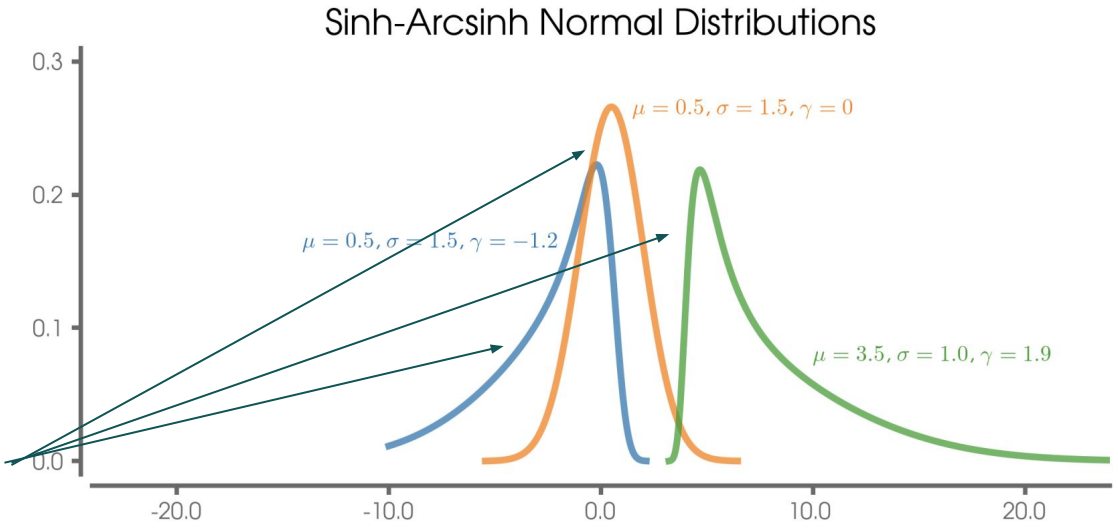
Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

A Fun and Important Point!

The SHASH normal distribution

- A distribution described by parameters
  - location (μ)
  - scale (σ)
  - skewness (γ)
  - and tail weight (τ)

Examples of SHASH with tail weight set to 1 (τ=1)



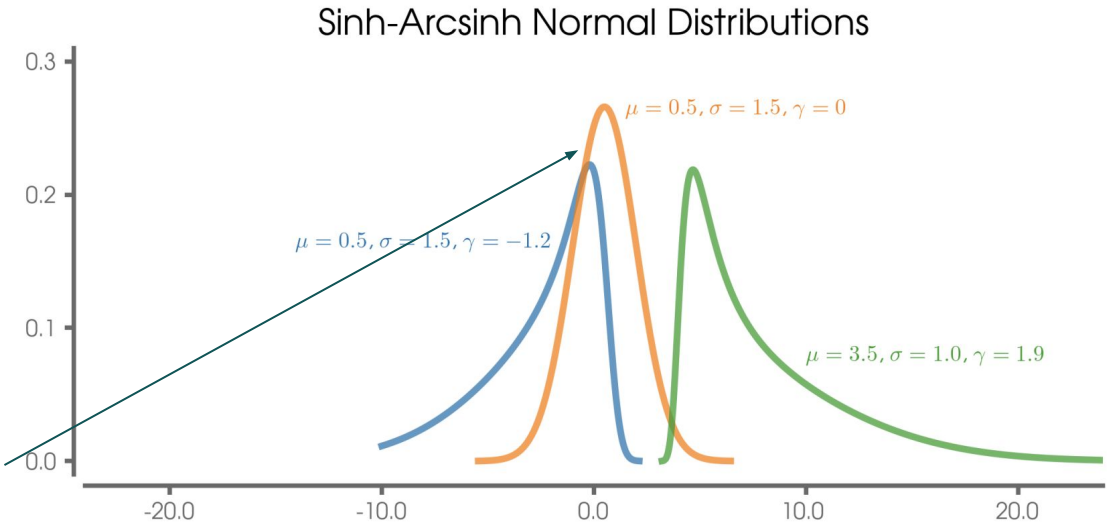
Jones and Pewsey: doi.org/10.1111/j.1740-9713.2019.01245.x

A Fun and Important Point!

The SHASH normal distribution

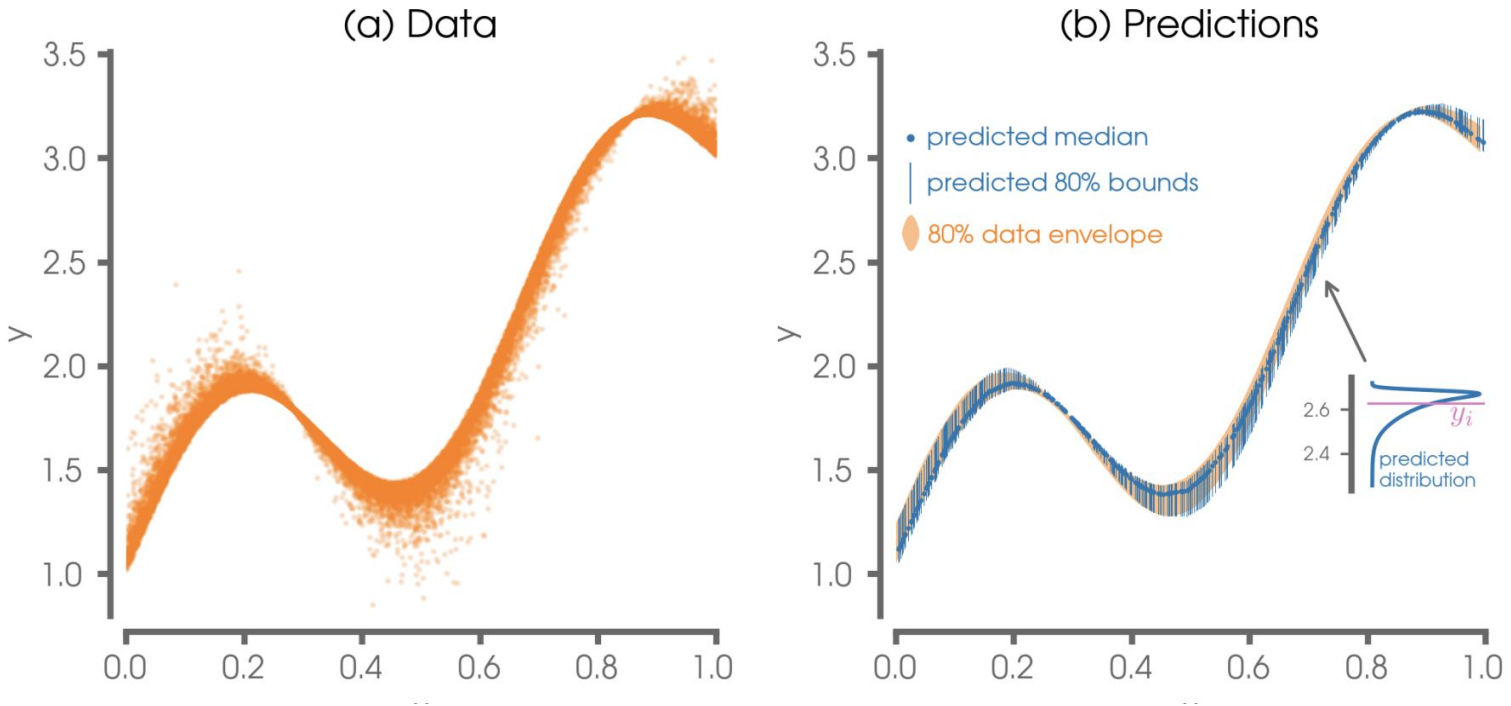
- A distribution described by parameters
  - location (μ)
  - scale (σ)
  - skewness (γ)
  - and tail weight (τ)

Note if skewness=0, then we have a normal distribution!



Jones and Pewsey: doi.org/10.1111/j.1740-9713.2019.01245.x

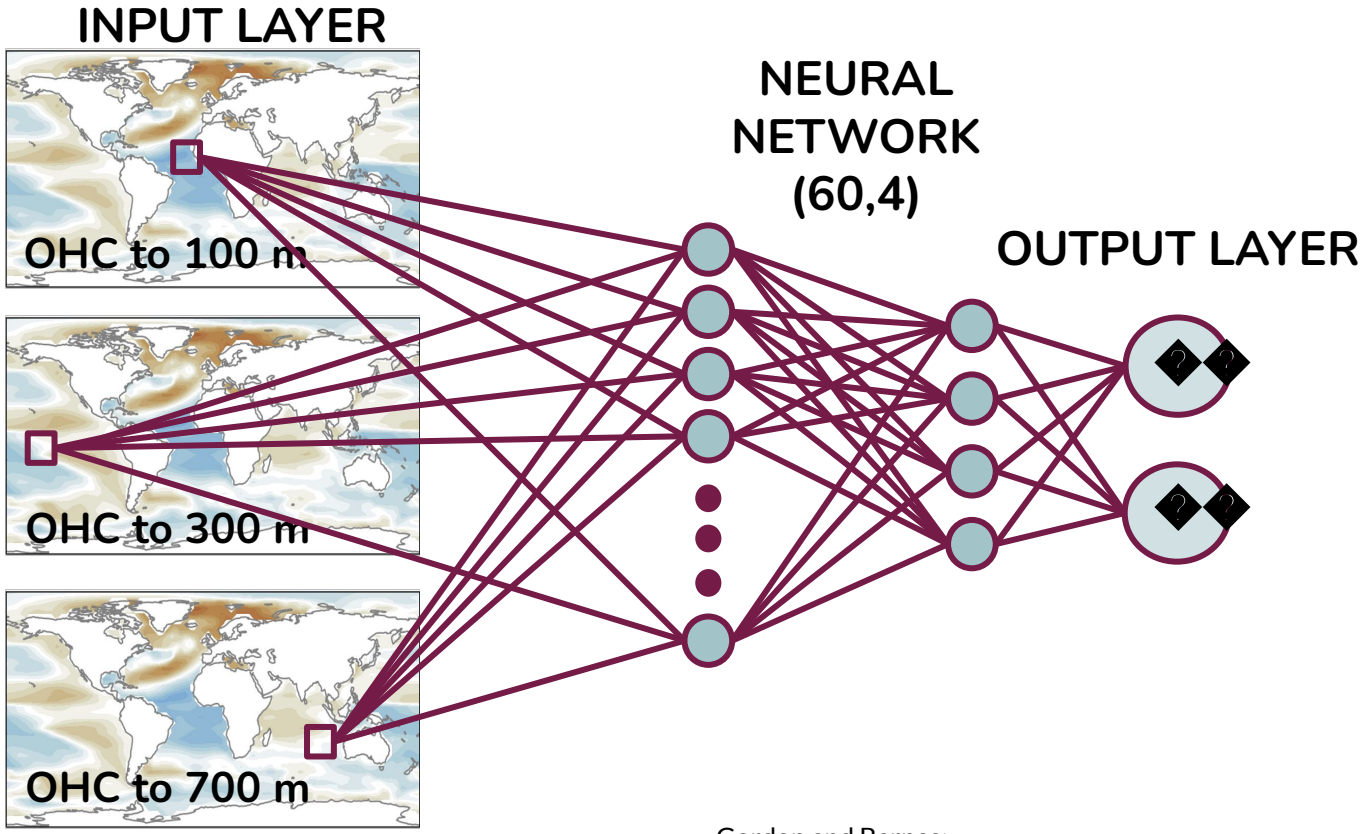
Using SHASH



Barnes, Barnes and Gordillo: https://arxiv.org/abs/2109.07250

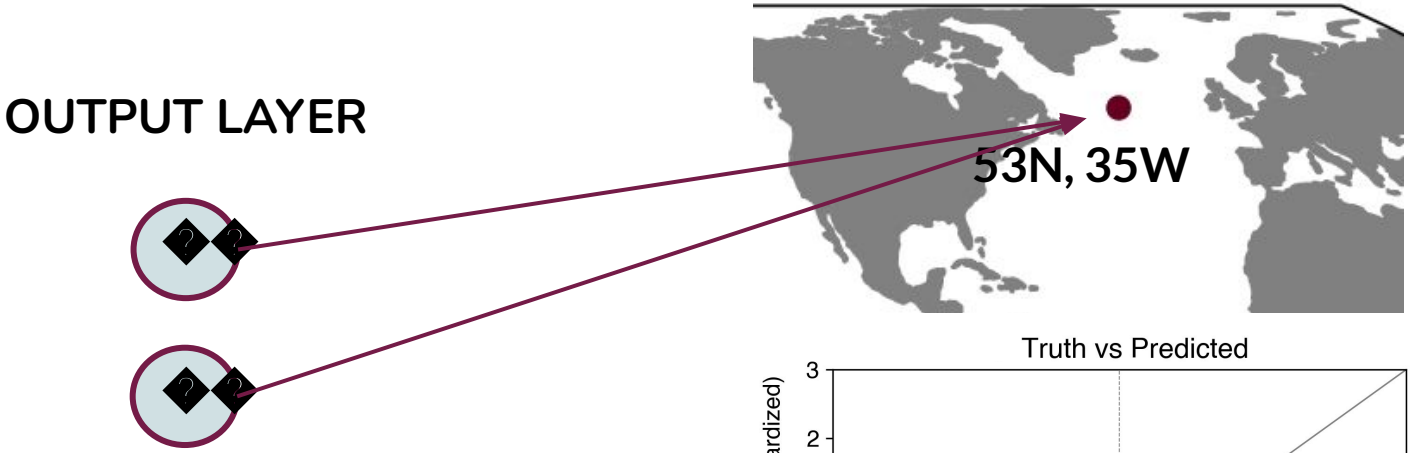
Application: Predicting SST on decadal timescales

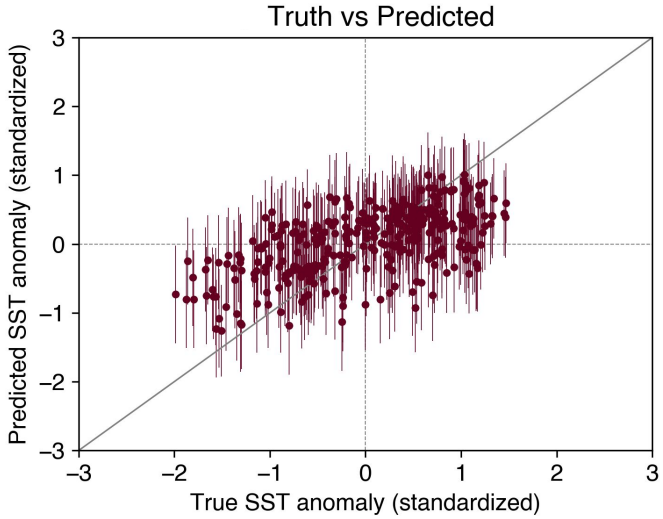
Predicting SST on decadal timescales



Gordon and Barnes:
essoar.org/doi/abs/10.1002/essoar.10510836.1

Predicting SST on decadal timescales





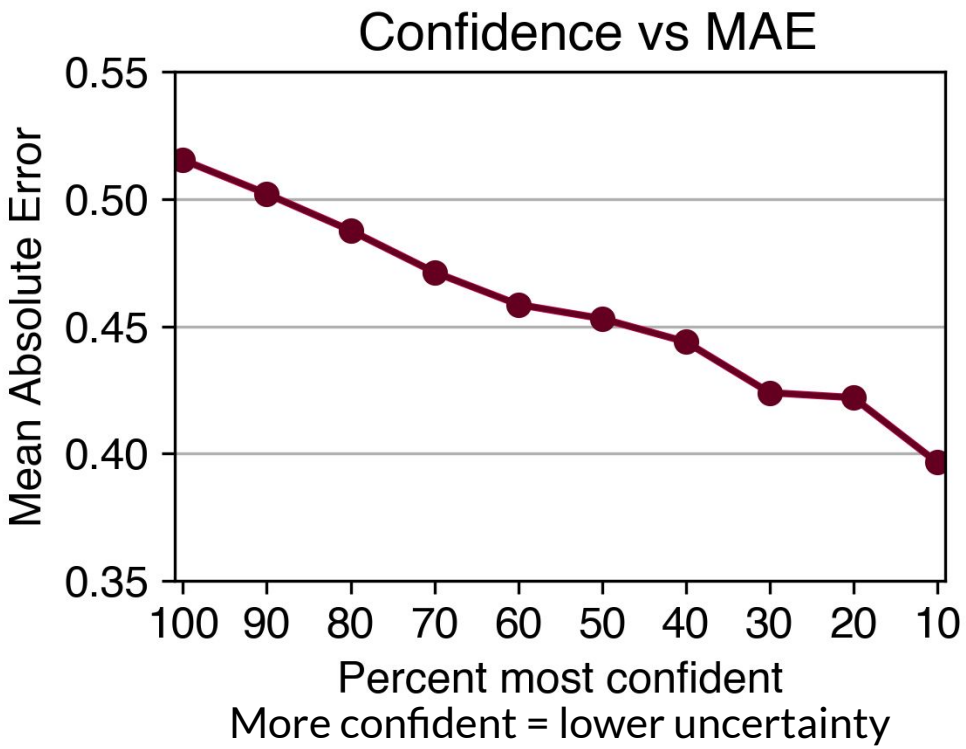
Prediction of SST anomaly with uncertainty at some point in the ocean, 1-5 years later, e.g. North Atlantic Subpolar Gyre

Identifying State-Dependent Predictability

The neural network identifies more predictability by assigning **lower uncertainty values to input patterns that result in more predictable outcomes**

For low uncertainty predictions, the ANN is more confident its prediction is closer to the truth

Prediction error (difference between truth and predicted anomaly) **decreases as we sort by increasing confidence**

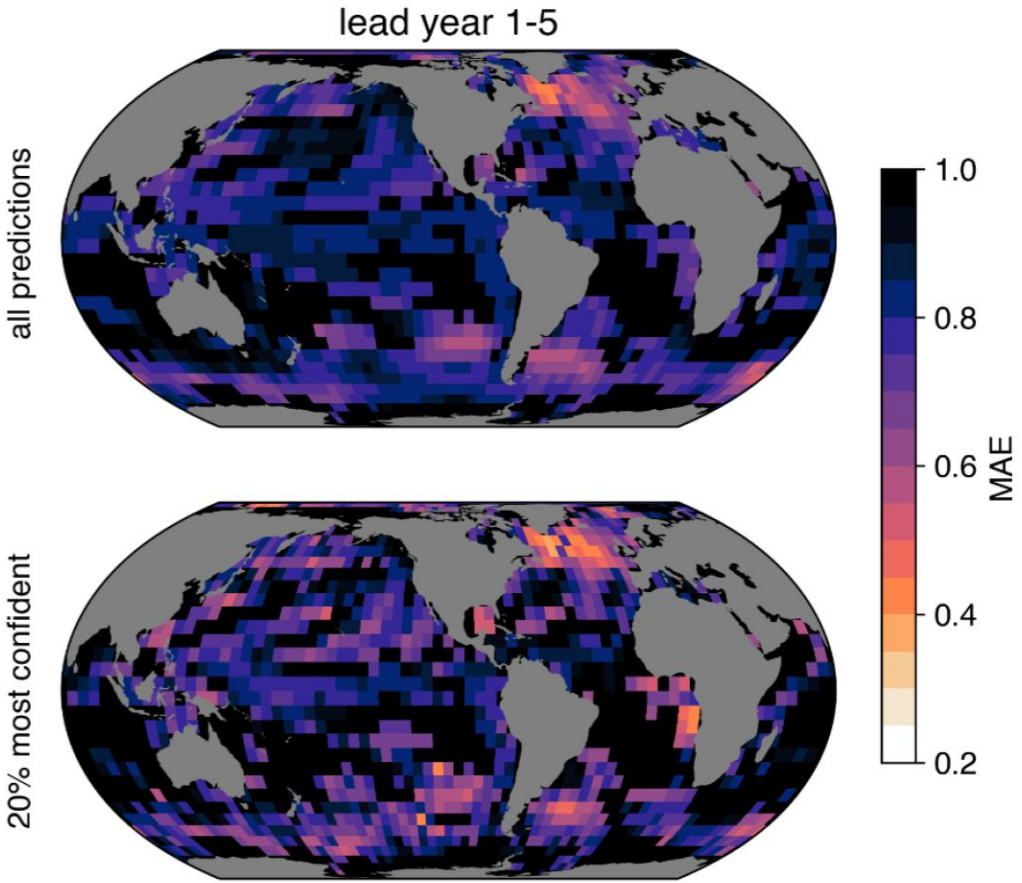


What can we learn from this?

We train a neural network to predict SST in 1-5 years at every grid point in the ocean (i.e. one* neural network per grid point).

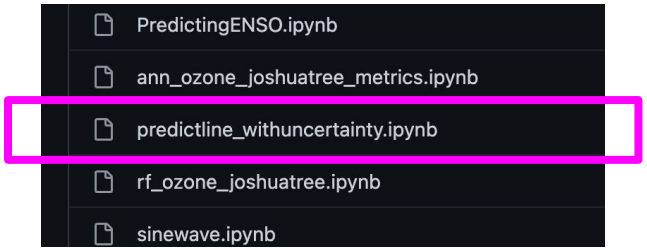
We can then compare the skill of our predictions across all samples in the testing…

And we can leverage the uncertainty/σ predictions to identify increased predictability



Resources if you want more UQ

Code! There is a coding example in the github to go through in your own time.



Papers! We have a quick explainer on this method

- **Barnes, Elizabeth A.,** Randal J. Barnes and **Nicolas Gordillo**: Adding Uncertainty to Neural Network Regression Tasks in the Geosciences, https://arxiv.org/abs/2109.07250

And I have a paper in review on adapting this in my own (Emily Gordon) research

- **Gordon, Emily M.** and **Elizabeth A. Barnes:** Incorporating Uncertainty into a Regression Neural Network Enables Identification of Decadal State-Dependent Predictability, https://www.essoar.org/doi/abs/10.1002/essoar.10510836.1

Main Takeaways

- ML has many applications to Earth Science and can help us learn NEW things about the Earth system and solve problems faster and cheaper
  - We need lots of data and *to think like a scientist*!
- We can open the black box of ML through visualization and understanding of the ML process to learn what the model learned
  - Explainable Artificial Intelligence (XAI) techniques are a potential game changer in Earth Science predictions
    - They aim to explain how a neural network makes predictions, i.e., what the *decision strategy* is

END