
Architectures and Applications

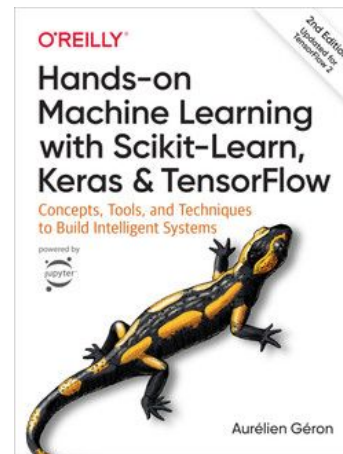
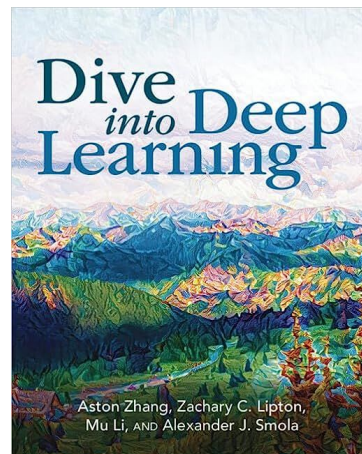
Machine Learning for Climate Scientists

4th-5th March 2024

Paul Keil

Resources

- Dive into Deep Learning:
<https://d2l.ai/index.html>
- Hands-on Machine Learning
- youtube
- <https://towardsdatascience.com/>



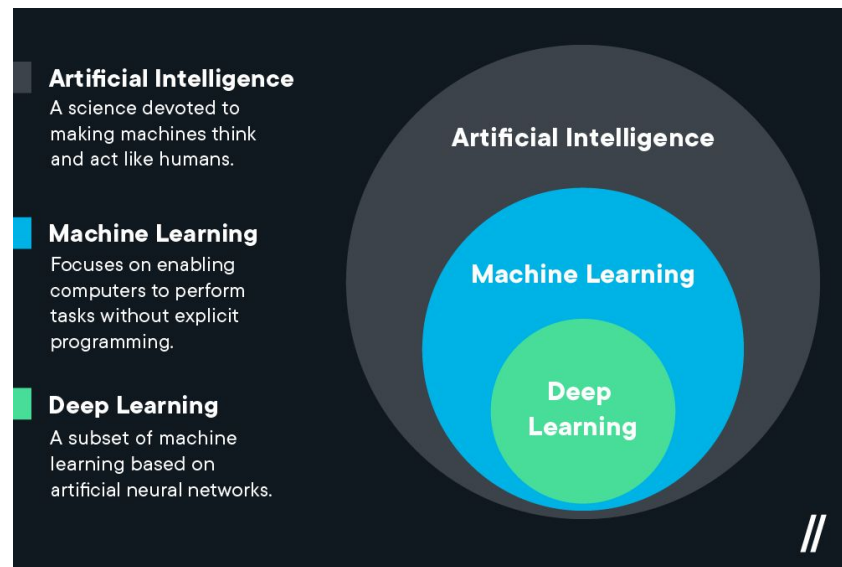
What to expect

- A look into the toolbox
- We won't be going deep into the math
- It will be overwhelming
- Focus is on Deep Learning
- Examples from weather and climate science
- probably some overlap to earlier presentations
- break after around 40mins

Outline

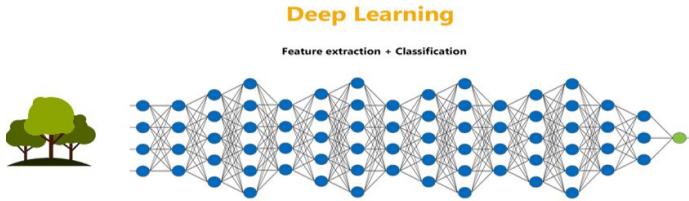
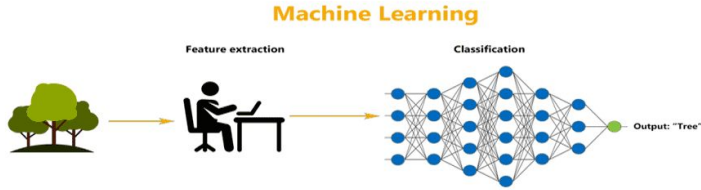
- Machine Learning (but not deep learning)
- Deep Learning:
 - Multilayer Perceptron
 - Convolutional Neural Networks
 - Recurrent Neural Networks
 - Transformers
 - Graph Neural Networks
- Autoencoders
- Probabilistic Deep Learning
- Physics-Informed Deep Learning
- Coupling general circulation models with ML model
- AI weather prediction

Machine Learning (but not deep learning)

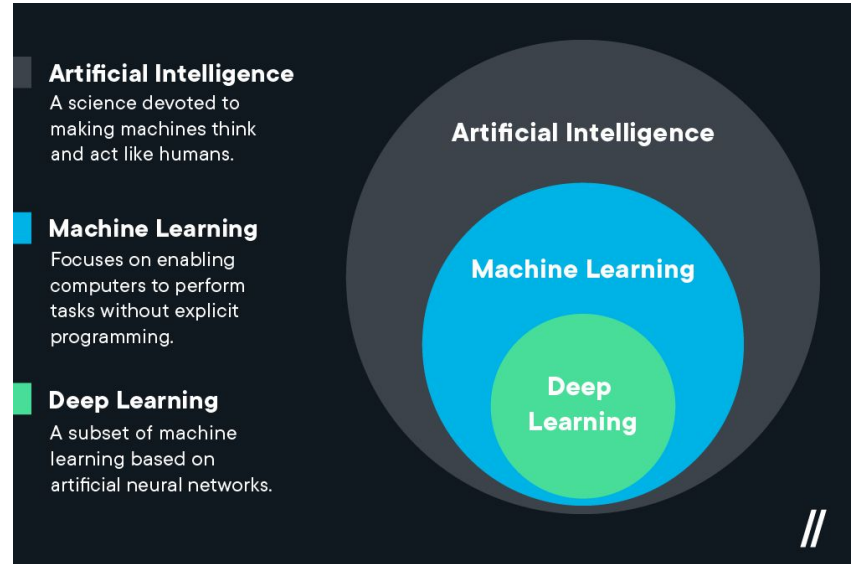


<https://flatironschool.com/blog/deep-learning-vs-machine-learning/>

Machine Learning (but not deep learning)



<https://www.baeldung.com/cs/machine-learning-vs-deep-learning>

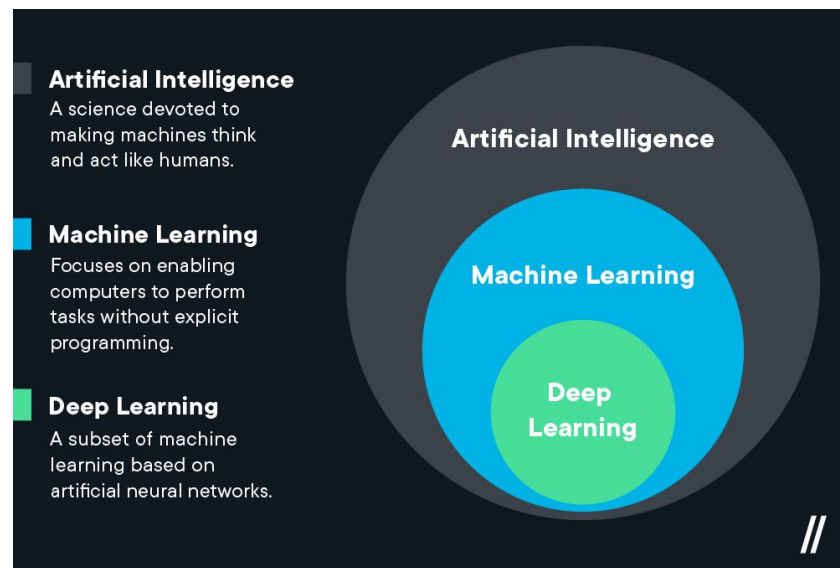


<https://flatironschool.com/blog/deep-learning-vs-machine-learning/>

Machine Learning (but not deep learning)

- Linear Regression
- Support Vector Machines (SVM)
- Random Forests
- k-means clustering
- Principal Components Analysis (PCA)
- ...

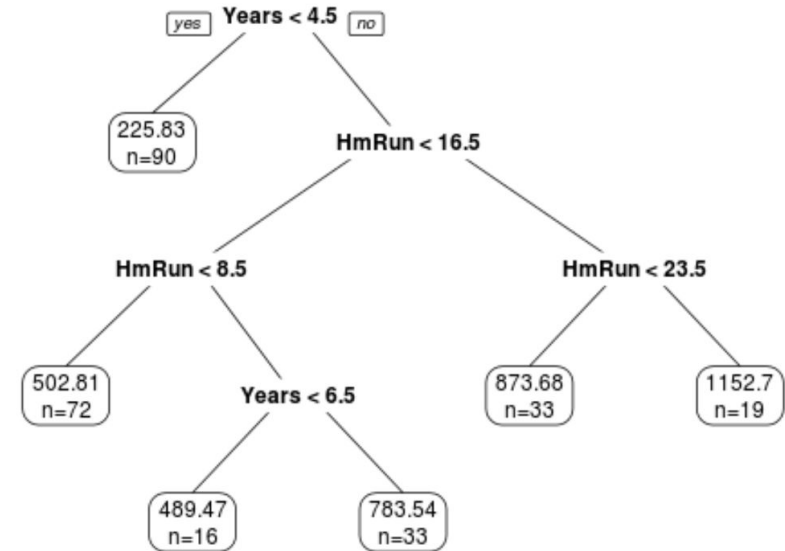
For many tasks, these algorithms are sufficient.



<https://flatironschool.com/blog/deep-learning-vs-machine-learning/>

Random Forest

- A random forest is an ensemble of decision trees
- regression or classification
- easy to train, works with little data
- returns feature importance by design -> Interpretability and Explainability out of the box!
- predicting convective downdrafts:
<https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2022MS003048>



Predicting baseball player's salary using average home runs and years of experience

<https://www.statology.org/random-forests/>

Deep Learning

Multilayer Perceptrons

- The “standard” neural net
- “fully connected layers”, or “dense layers”
- calculate hidden state H using weights (W) and biases (b) and activation function

W : weight matrix

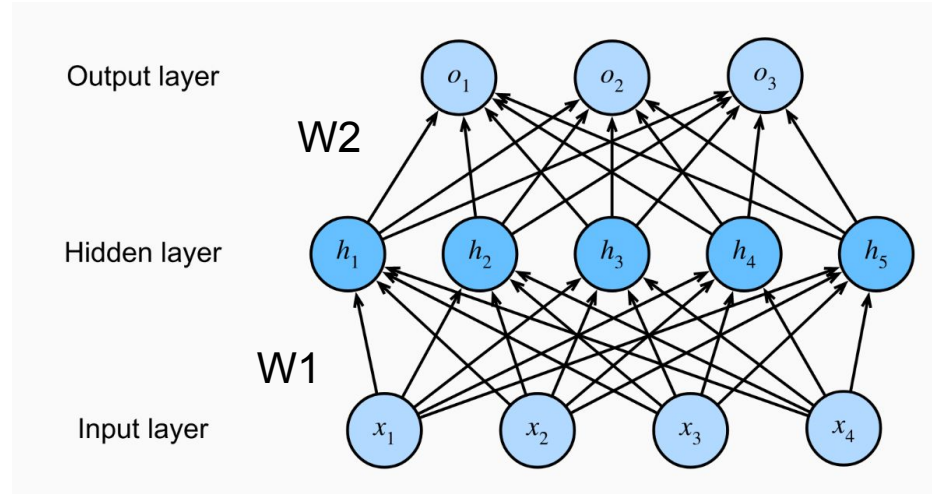
b : biases

σ : nonlinear function

X : input vector

H : hidden or “latent” state

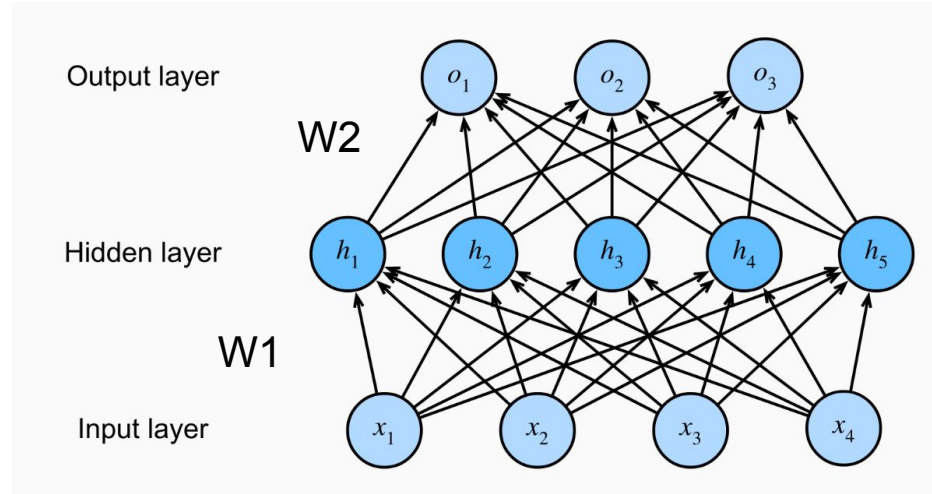
O : output



$$\mathbf{H} = \sigma(\mathbf{XW}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{HW}^{(2)} + \mathbf{b}^{(2)}.$$

Multilayer Perceptrons

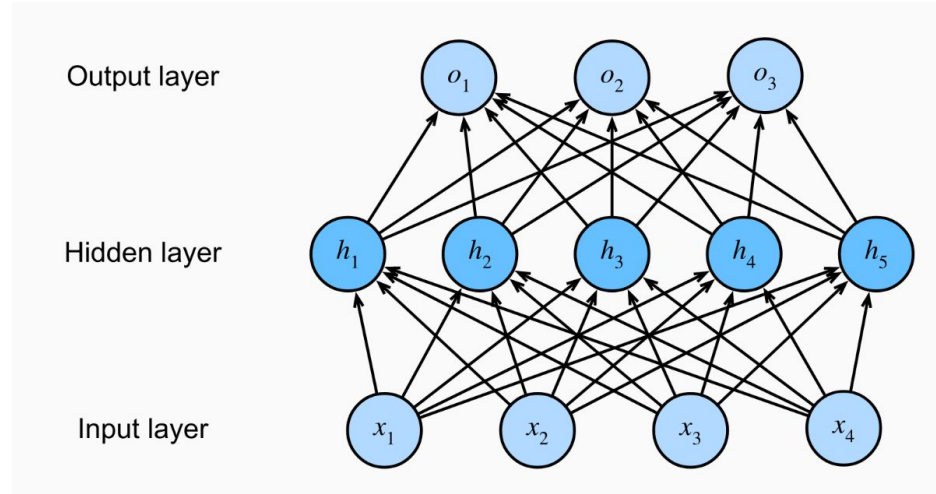
- Weights and biases are the parameters that are “learned”
- input x has “4 features”
- The number of hidden layers and the amount of neurons can be chosen



$$\mathbf{H} = \sigma(\mathbf{XW}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{HW}^{(2)} + \mathbf{b}^{(2)}.$$

Multilayer Perceptrons

- The “standard” neural net
- “fully connected layers”, or “dense layers”
- Connections between nodes are weights
- Weights are the parameters that are “learned”
- input x has “4 features”
- The number of hidden layers and the amount of neurons can be chosen



Multilayer Perceptrons

Nodes are calculated from the previous layer's nodes using the weights, biases and activation function

W : weight matrix

b : biases

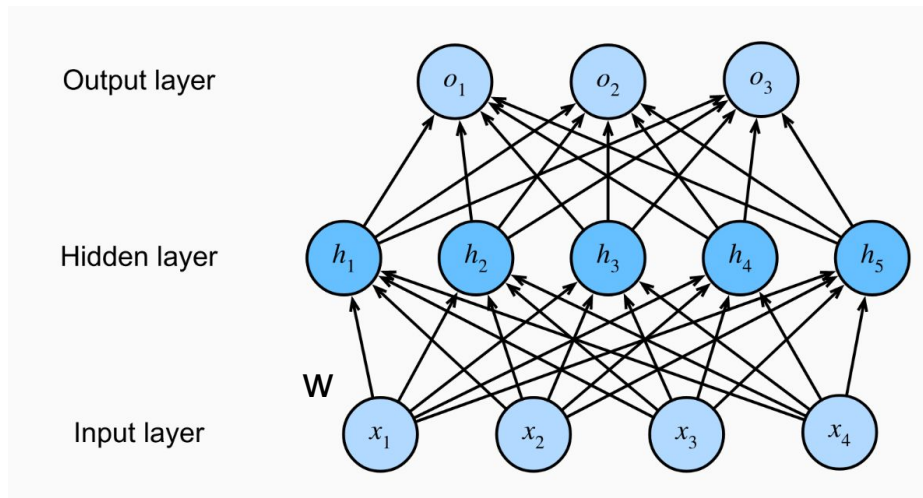
σ : nonlinear function

X : input vector

H : hidden or "latent" state

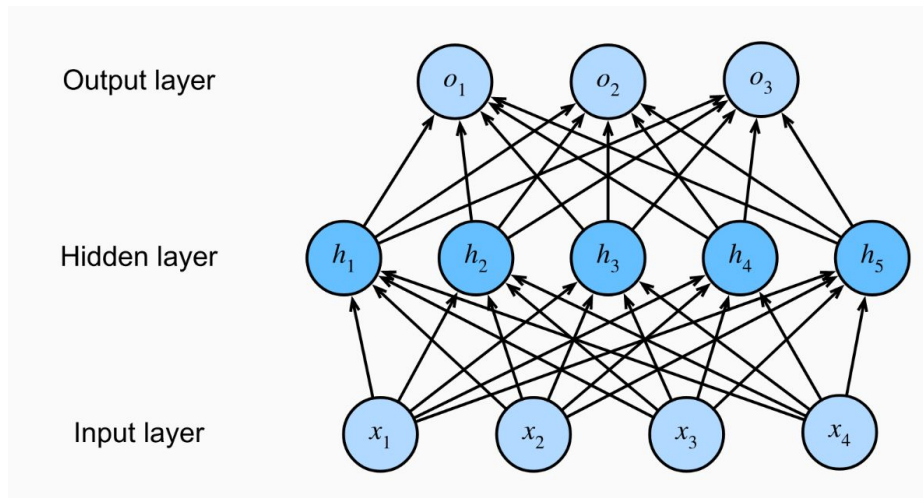
O : output

W , b are the trainable parameters



$$\mathbf{H} = \sigma(\mathbf{XW}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{HW}^{(2)} + \mathbf{b}^{(2)}.$$

Multilayer Perceptrons



Deep Learning is mostly just linear
Algebra and some Calculus! No need
to be scared :)

$$\mathbf{H} = \sigma(\mathbf{XW}^{(1)} + \mathbf{b}^{(1)}),$$
$$\mathbf{O} = \mathbf{HW}^{(2)} + \mathbf{b}^{(2)}.$$

Activation Functions

“Add non-linearity”. Neurons are activated (“they fire”) like brain neurons

- ReLu

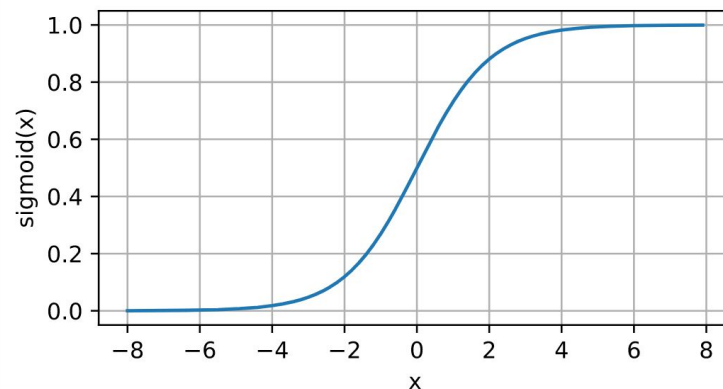
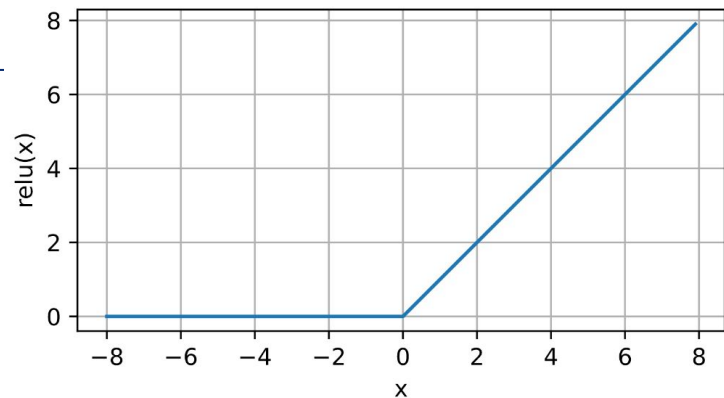
$$\text{ReLU}(x) = \max(x, 0).$$

simple and good performance
behaves “well” during backpropagation

- Sigmoid

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}.$$

important for some architectures

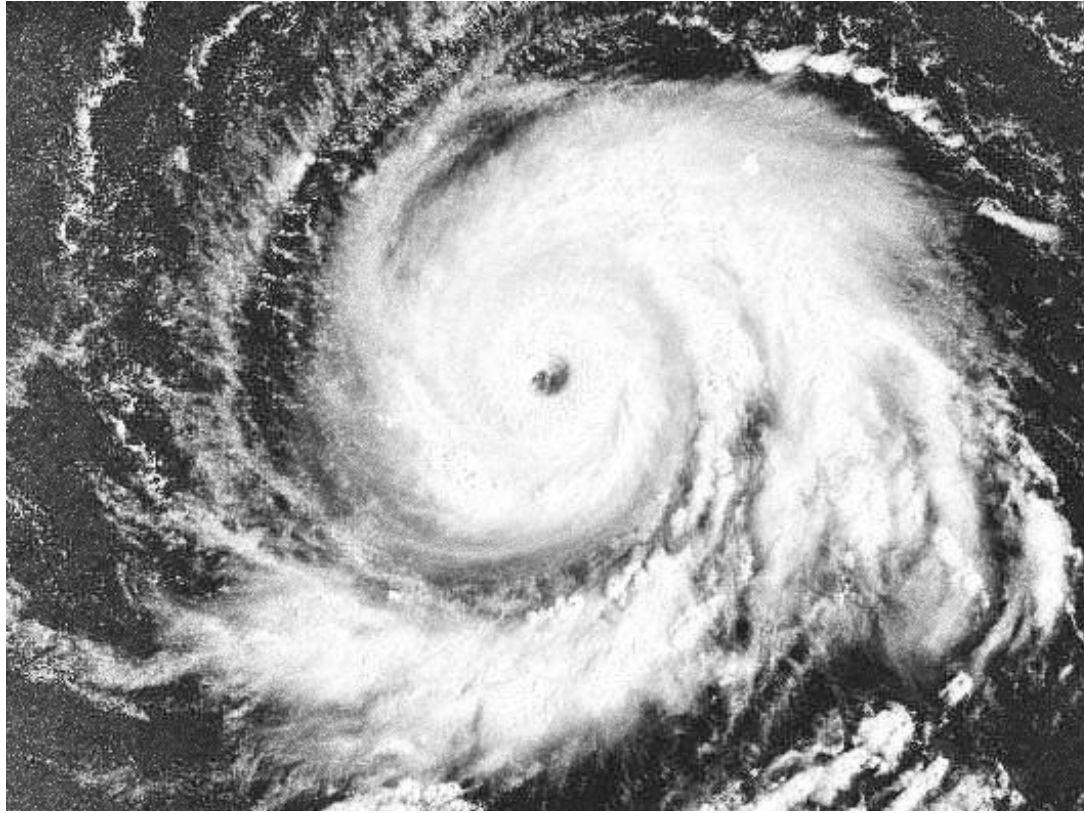


Let's go over this again...



Inputs

Source data fed into the neural network, with the goal of making a decision or prediction about the data.





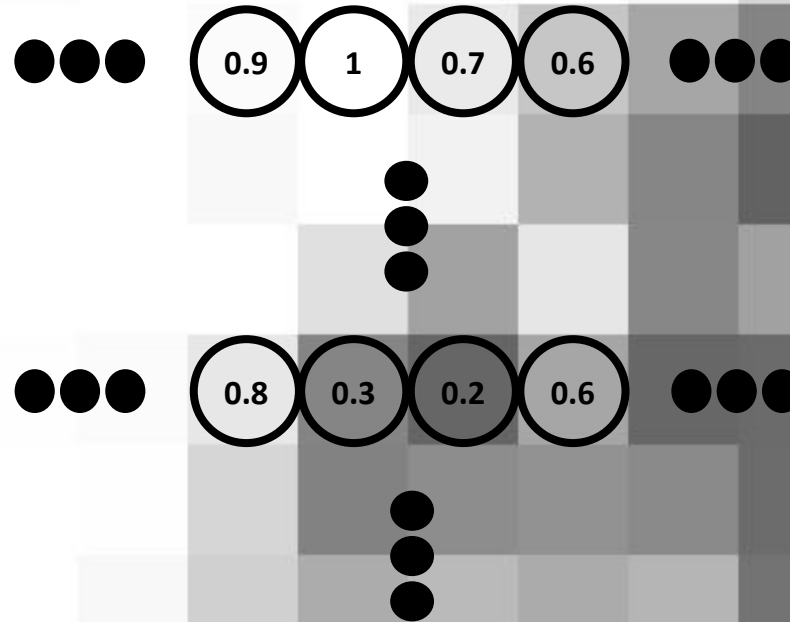
Inputs

Source data fed into the neural network, with the goal of making a decision or prediction about the data. The data is broken down into binary signals, to allow it to be processed by single neurons—for example an image is input as individual pixels.



Training, Validation, Test Set

A set of outputs for which the correct outputs are known, which can be used to train the neural networks.



General



Inputs

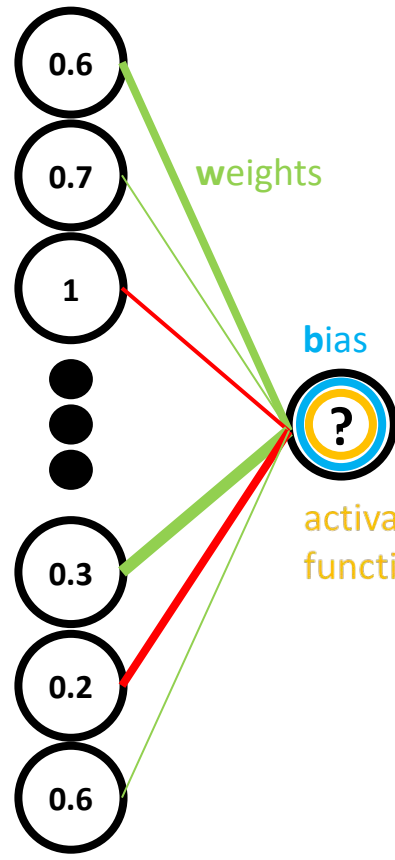


Training, Validation, Test Set



Nodes, Weights, Biases

activation



$$\sigma(w_1 a_1 + w_2 a_2 + \dots + w_n a_n + b)$$

activation function σ

General



Inputs



Training, Validation, Test Set



Nodes, Weights, Biases



Activation Function

Each neuron accepts part of the input and passes it through the activation function. Commonly used functions are the sigmoid function, tanh and ReLu.

input a

0.6

0.7

1

⋮

⋮

⋮

0.3

0.2

0.6

weights

bias

activation
function σ

$$\sigma(w_1 a_1 + w_2 a_2 + \dots + w_n a_n + b)$$

General



Inputs



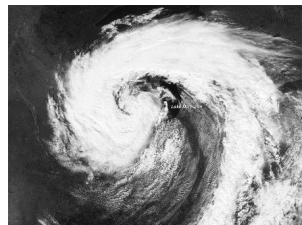
Training, Validation, Test Set



Nodes, Weights, Biases

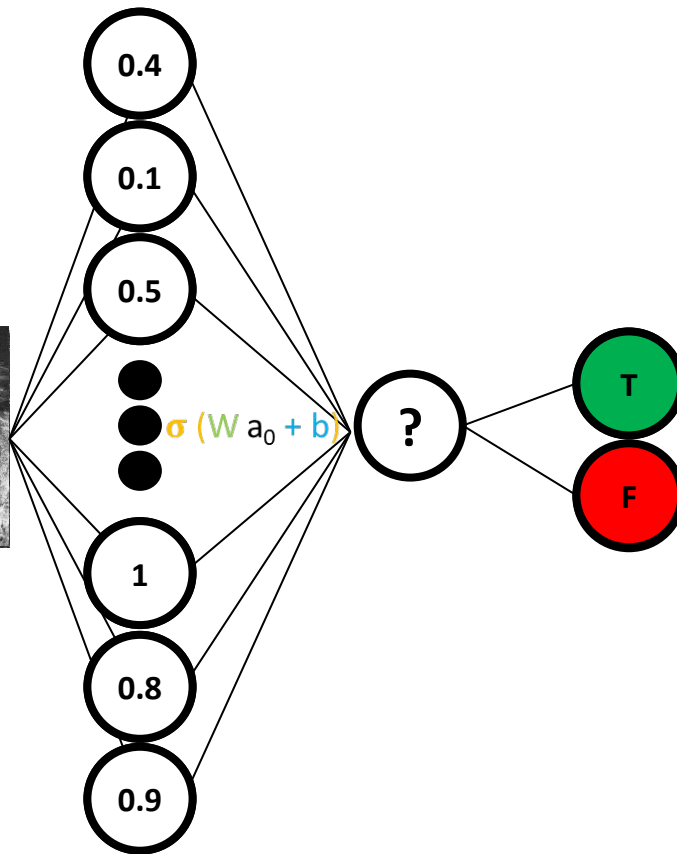


Activation
Function



Outputs

The output of the neural network can be a real value between 0 and 1, a boolean, or a discrete value (for example, a category ID).



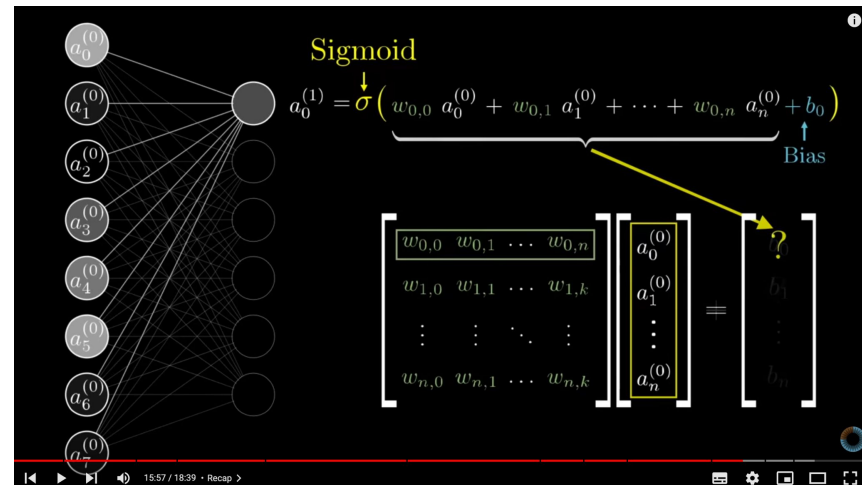
Questions

we'll be needing this again...

Multilayer Perceptrons

A great visual introduction:

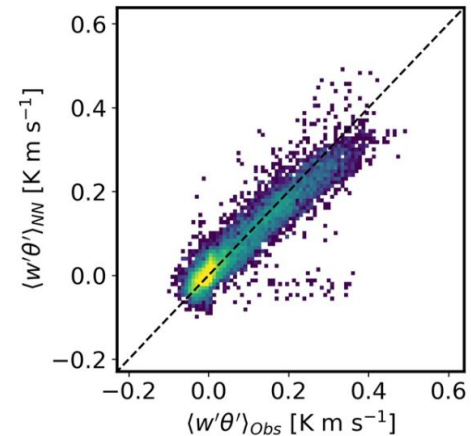
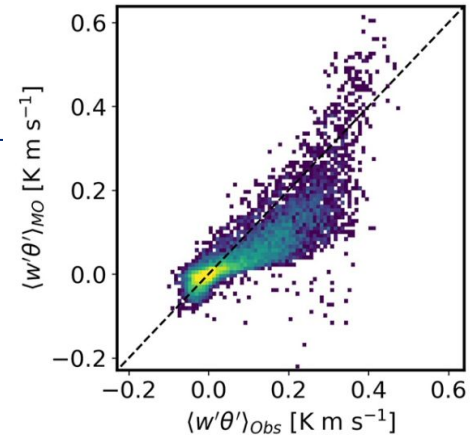
https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi



Predicting surface heat fluxes

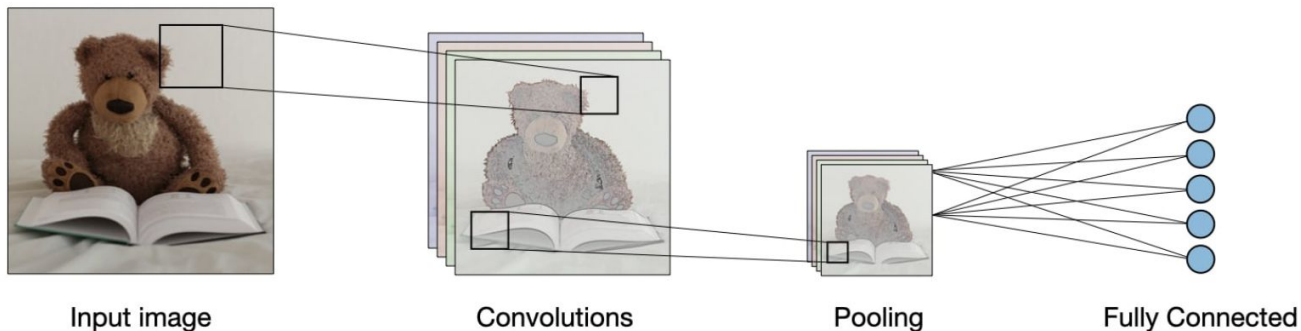
- Predicting surface heat fluxes using a neural network trained on observations
- input: wind speed, temperature, Richardson number
- two hidden layers with 64 neurons each
- Top: Traditional Approach
Bottom: Neural Network

Muñoz-Esparza, Domingo, et al. "On the Application of an Observations-Based Machine Learning Parameterization of Surface Layer Fluxes Within an Atmospheric Large-Eddy Simulation Model." *Journal of Geophysical Research: Atmospheres* 127.16 (2022): e2021JD036214.



Convolutional Neural Networks

- Multilayer Perceptrons do not account for data structure
- Convolutional Neural Networks solve this problem
- Popular for any 2-dimensional data, especially image classification tasks
- you will be programming one tomorrow



1. Convolution

$$[\mathbf{H}]_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$$

i,j : pixel location

\mathbf{V} : weight matrix of kernel

u : bias

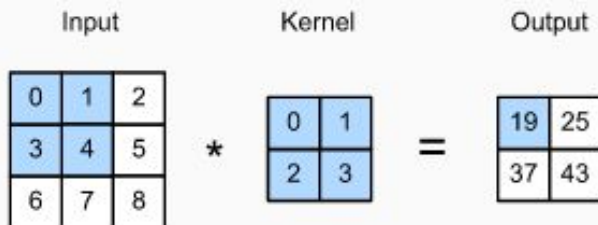
Δ : kernel size

a,b : kernel indices

\mathbf{X} : input

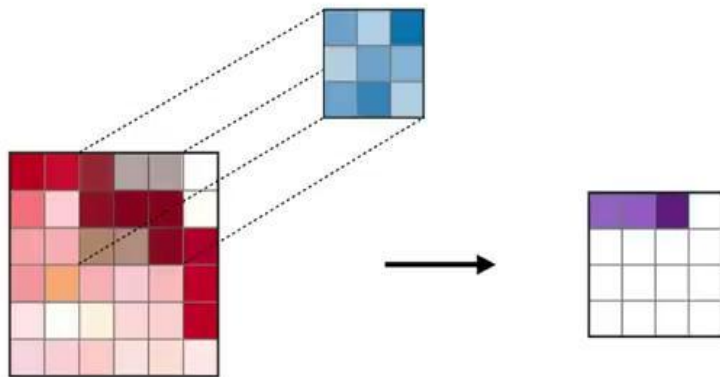
\mathbf{H} : hidden or “latent” state

(not strictly a convolution, but a cross-correlation)



1. Convolution

$$[\mathbf{H}]_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$$



i, j : pixel location

\mathbf{V} : weight matrix of kernel

u : bias

Δ : kernel size

a, b : kernel indices

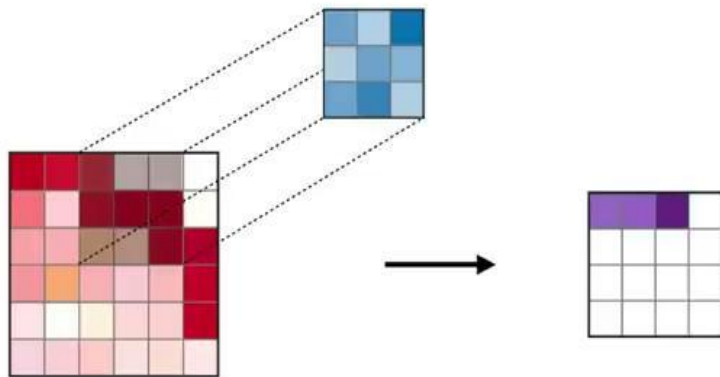
\mathbf{X} : input

\mathbf{H} : hidden or “latent” state

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks>

1. Convolution

$$[\mathbf{H}]_{i,j} = u + \sum_{a=-\Delta}^{\Delta} \sum_{b=-\Delta}^{\Delta} [\mathbf{V}]_{a,b} [\mathbf{X}]_{i+a,j+b}.$$



i, j : pixel location

\mathbf{V} : weight matrix of kernel

u : bias

Δ : kernel size

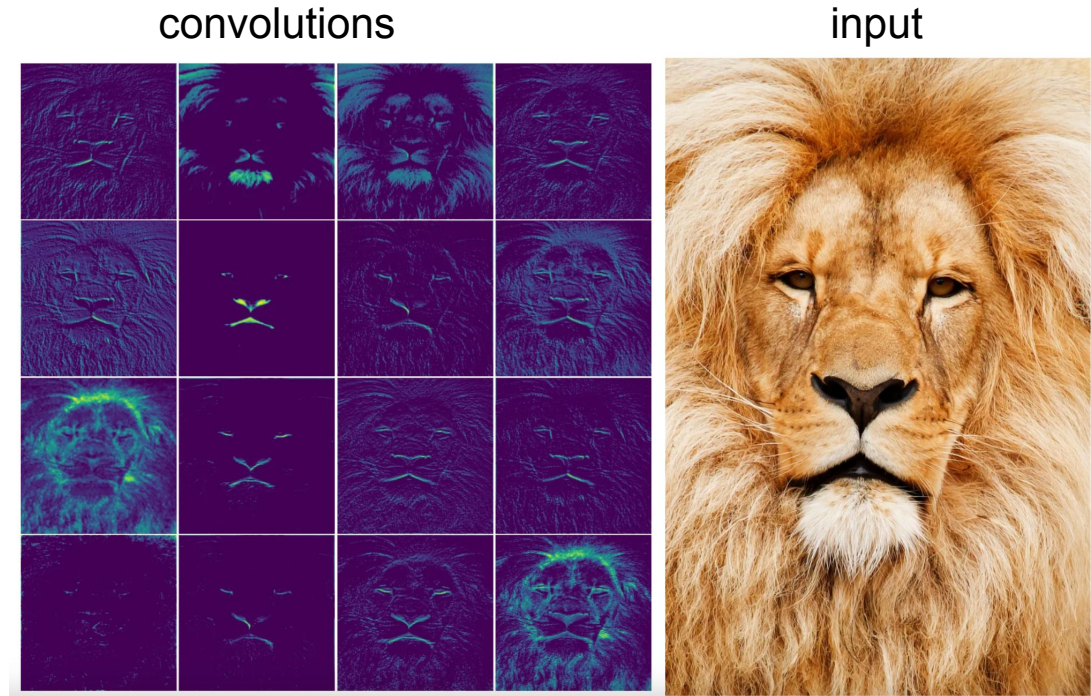
a, b : kernel indices

\mathbf{X} : input

\mathbf{H} : hidden or “latent” state

Typically, there are multiple kernels \mathbf{V} , and as a result \mathbf{H} typically has three dimensions. Kernels are trained.

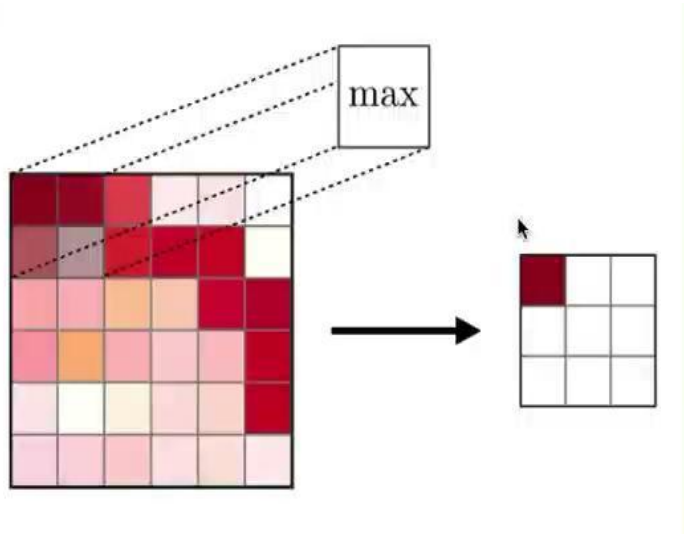
How does a CNN learn?



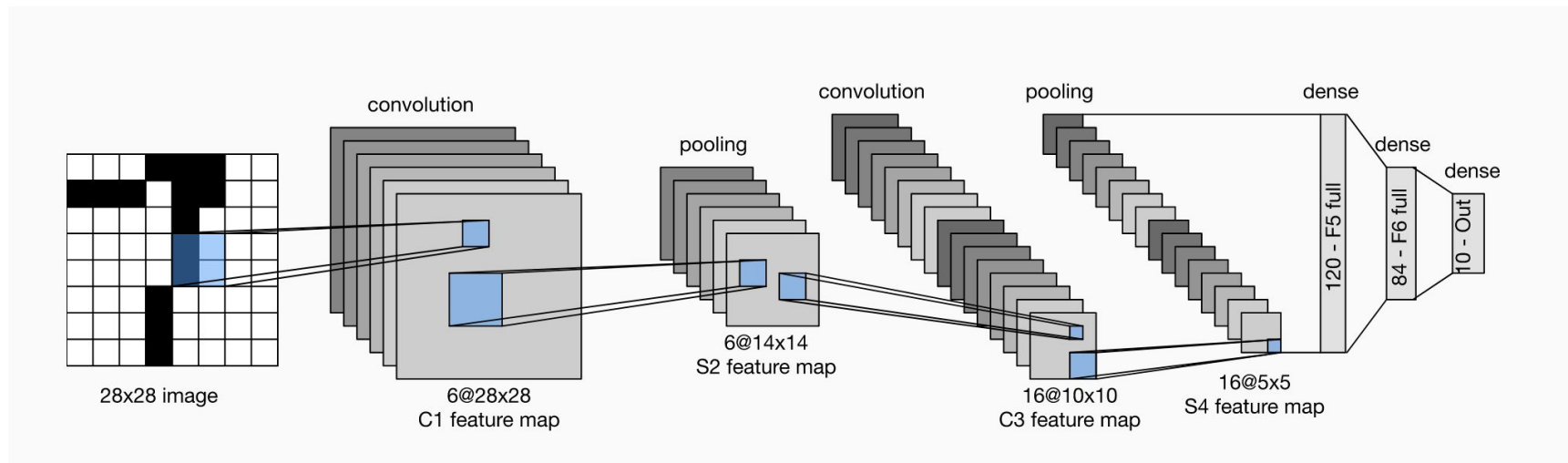
<https://towardsdatascience.com/exploring-feature-extraction-with-cnns-345125cefc9a>

2. Pooling

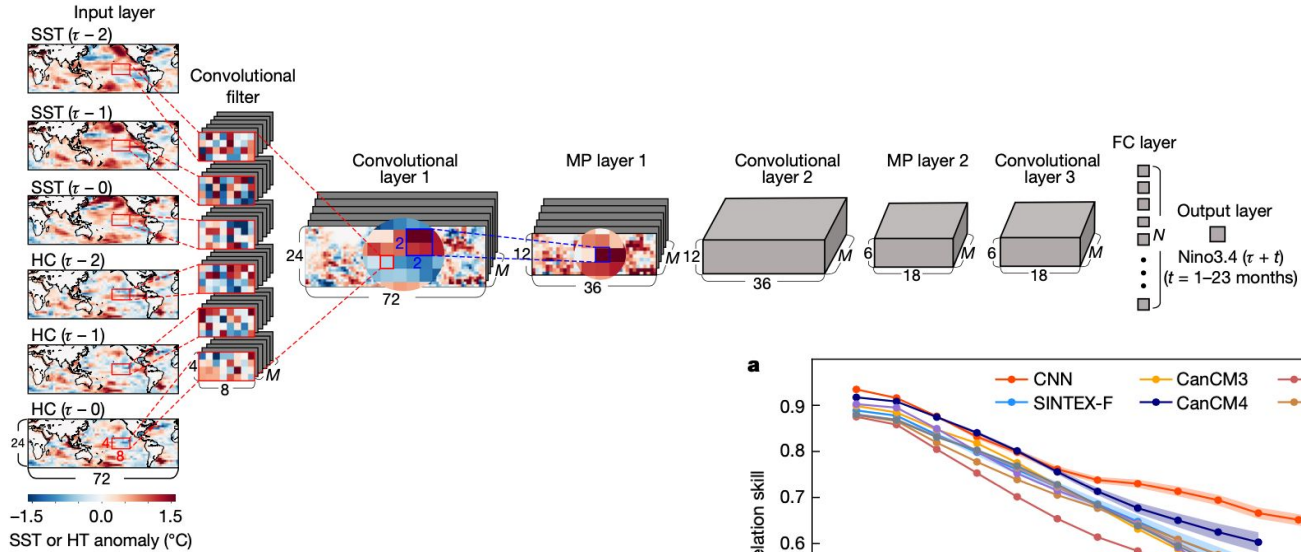
A downsampling operation typically after the convolution layer



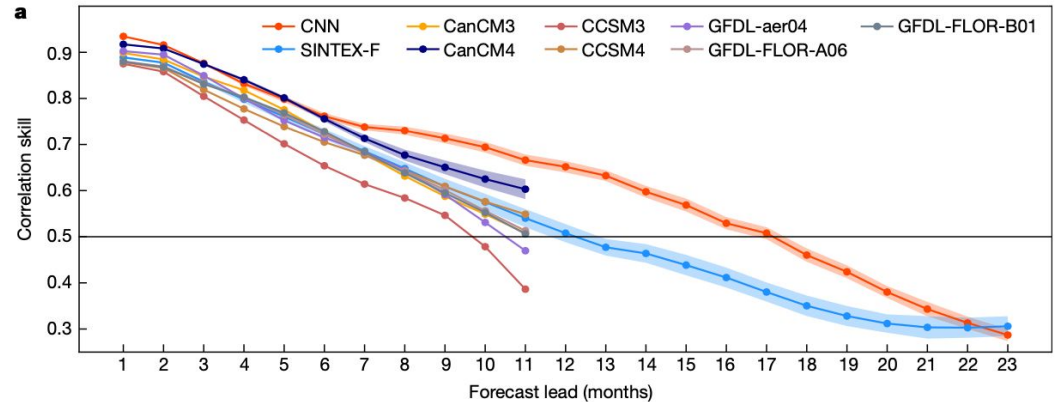
Putting it all together



Deep learning for multi-year ENSO forecasts

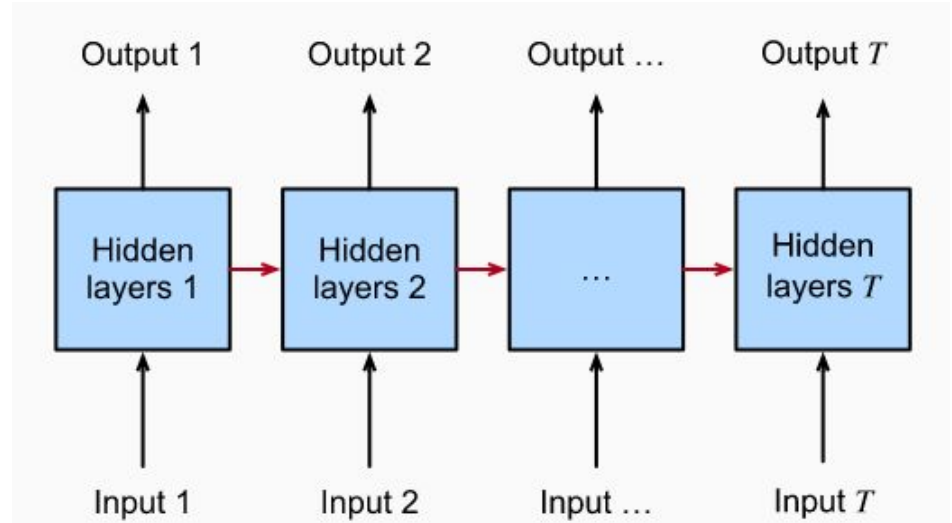


Ham, Yoo-Geun, Jeong-Hwan Kim, and Jing-Jia Luo. "Deep learning for multi-year ENSO forecasts." *Nature* 573.7775 (2019): 568-572.



Recurrent Neural Networks

- Used for sequential data: time series prediction, language processing
- memory mechanism
- problem: exploding or vanishing gradients during learning

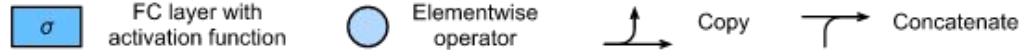
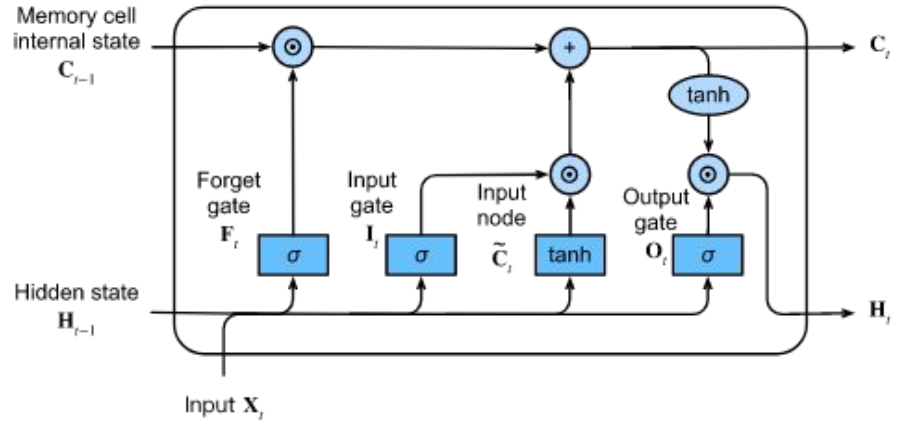


Long short-term memory (LSTM)

Can learn what important data is,
and remembers this data for a
long time

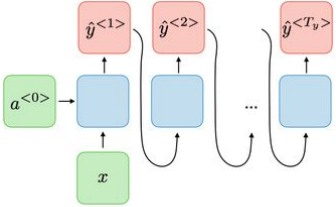
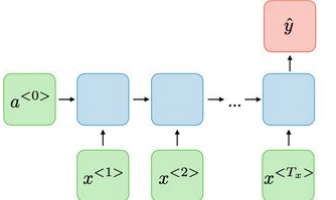
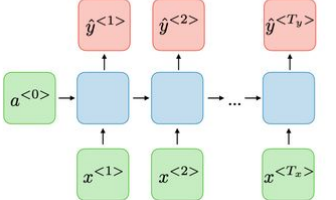
Hidden State: Short term memory
Memory Cell: Long term memory

“Gated Recurrent Units (GRU)”:
streamlined version of the LSTM



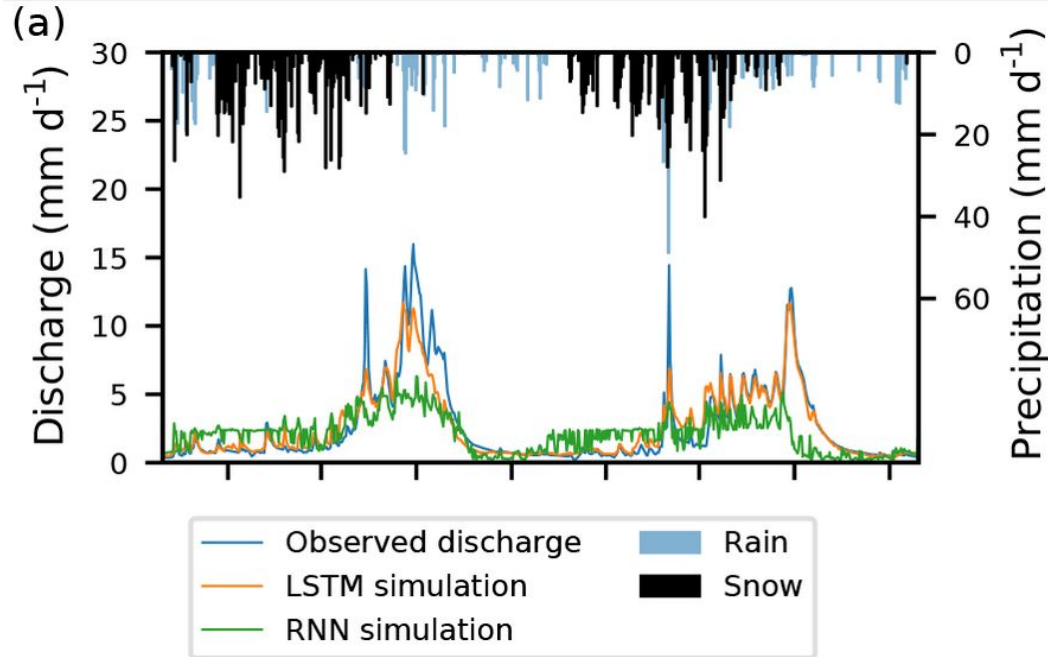
$$\begin{aligned} I_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i), \\ F_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f), \\ O_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o), \end{aligned}$$

Different ways to employ them

<p>One-to-many $T_x = 1, T_y > 1$</p>		<p>Music generation</p>
<p>Many-to-one $T_x > 1, T_y = 1$</p>		<p>Sentiment classification</p>
<p>Many-to-many $T_x = T_y$</p>		<p>Name entity recognition</p>

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

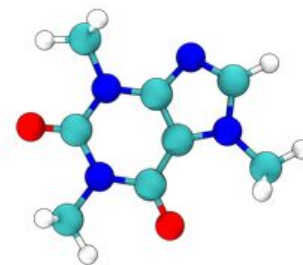
LSTM for rainfall-runoff modelling



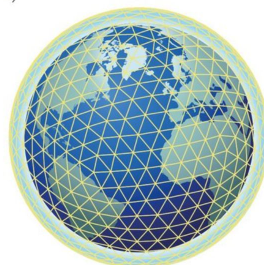
Kratzert, F., Klotz, D., Brenner, C., Schulz, K., and Herrnegger, M.: Rainfall-runoff modelling using Long Short-Term Memory (LSTM) networks, *Hydrol. Earth Syst. Sci.*, 22, 6005–6022, <https://doi.org/10.5194/hess-22-6005-2018>, 201

Graph Neural Networks

- Similar to CNNs, but for arbitrary structures
- Molecules, social networks, climate model grid, weather stations
- map your data to a graph: nodes, edges
- Used to predict node characteristics, edge characteristics, or even new graph structures



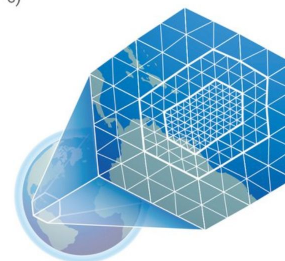
a)



b)

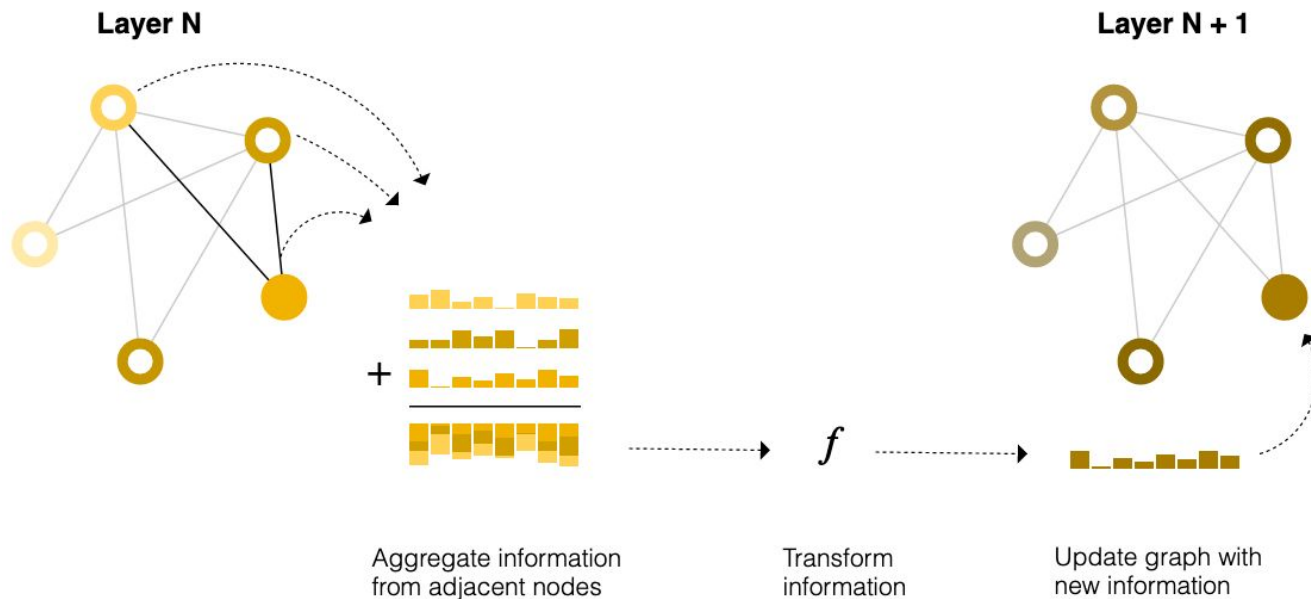


c)



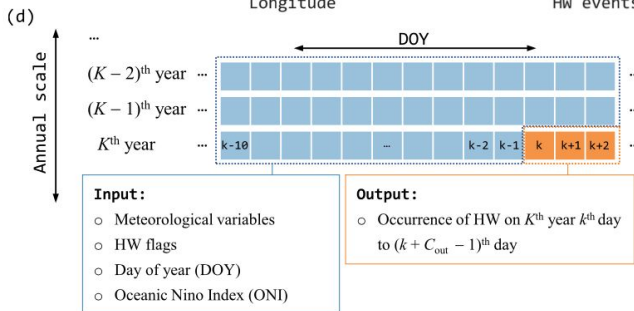
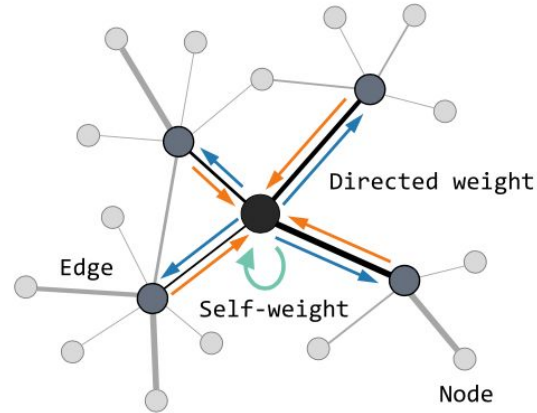
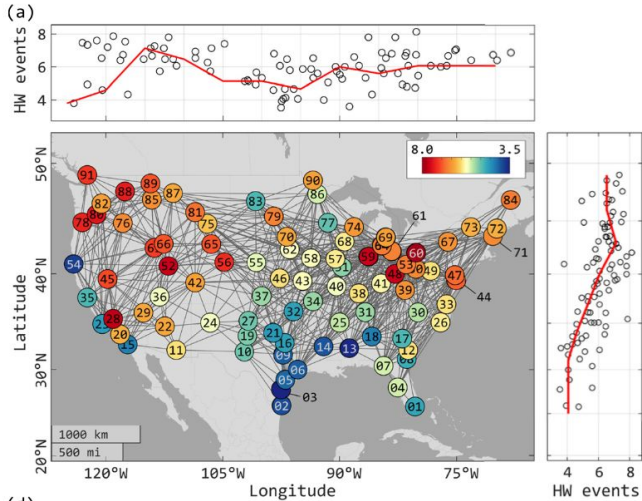
<https://mpimet.mpg.de/forschung/modellierung>

Graph Neural Networks: Message Passing



For a great introduction to GNNs: <https://distill.pub/2021/gnn-intro/>

GNNs for predicting heat waves



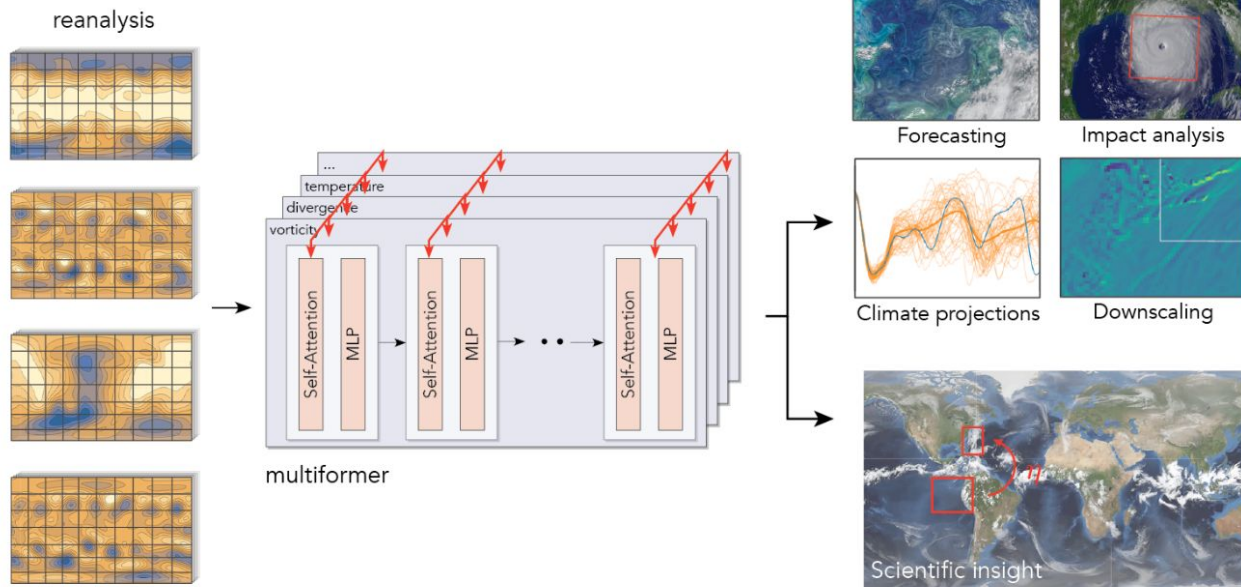
Li, Peiyuan, et al. "Regional heatwave prediction using Graph Neural Network and weather station data." *Geophysical Research Letters* 50.7 (2023): e2023GL103405.

Break

State of the Art Deep Learning: Transformers

- Most (all?) large language models are based on the transformer architecture
- Vision Transformers for diverse vision tasks
- Core idea: Attention Mechanism
- Long Term memory
- Data is positionally encoded and embedded into tokens
 - Can handle any data (text, images, 4D data,...)
- Used to build “foundation models”
 - EU AI act: “AI model that is trained on broad data at scale, is designed for generality of output, and can be adapted to a wide range of distinctive tasks”.
- Further reading:
 - <https://jalammar.github.io/illustrated-transformer/>
 - Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

Transformers: Atmorep

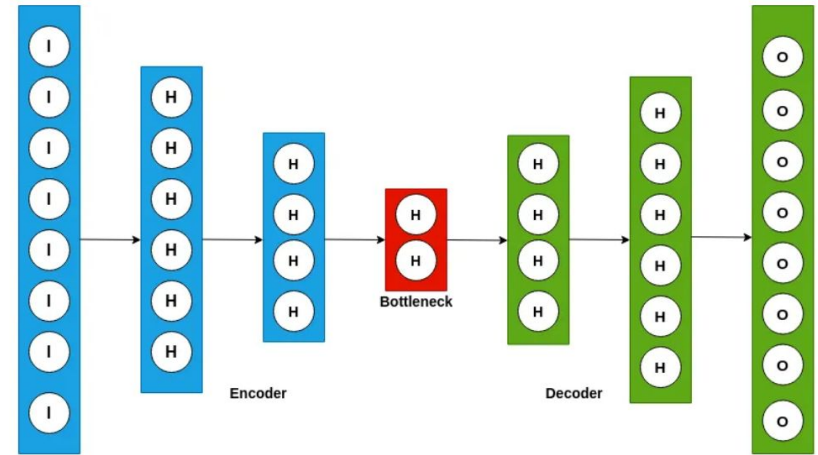


Lessig et al, under review at Nature

<https://arxiv.org/abs/2308.13280>, <https://www.atmorep.org/>

Autoencoders

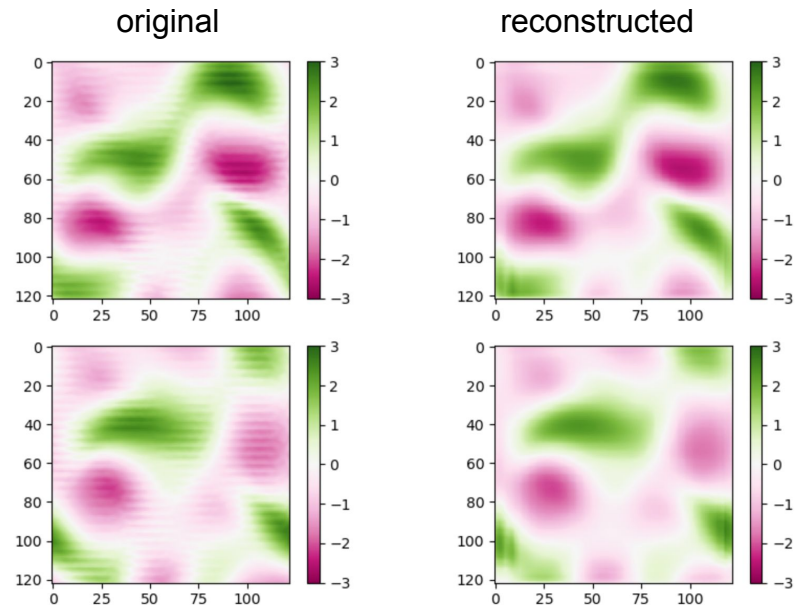
- The input is transformed and its dimensionality reduced by an encoder
- The decoder tries to reconstruct the input
- the bottleneck is a lower dimensional representation of your data
- exploits underlying correlations among data
- Applications include compression, filtering, denoising, ...
- Can consist of elements from different deep learning architectures (CNNs, Attention, GNNs,...)



<https://towardsdatascience.com/introduction-to-autoencoders-7a47cf4ef14b>

Denoising/Filtering with Autoencoders

- Numerical Simulations sometimes have unphysical artifacts that arise from the discretisation and the the numerical methods
- CNN-based autoencoder extracts the important features and filters the “random” noise



Raw and filtered zonal wind. Taken from an ongoing project at AWI with V. Sidorenko

Probabilistic Deep Learning

Common problems in deep learning:

- overfitting
- overconfidence

→ Probabilistic Approaches to deep learning can help

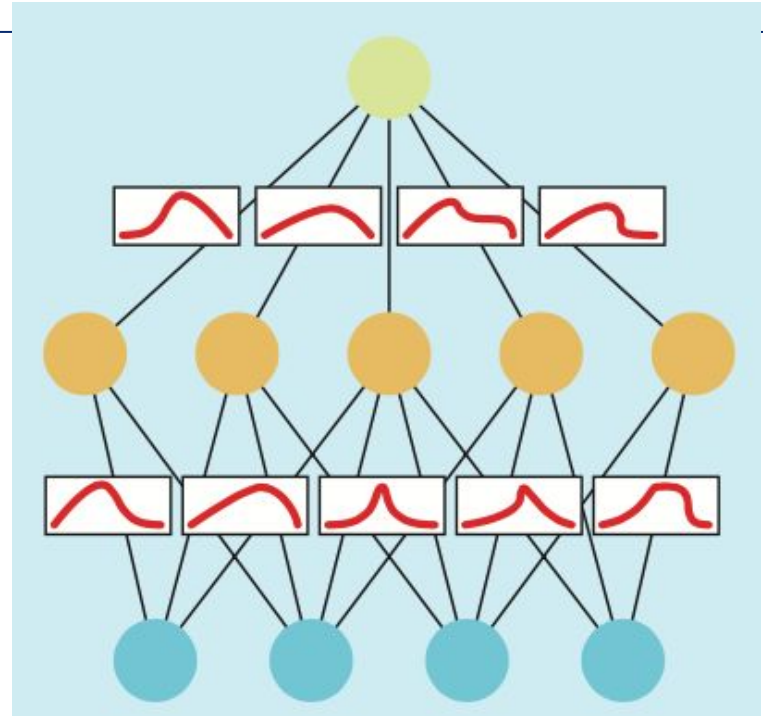
The main goal is not to be better than point-estimate methods, although this might be the case, but to provide an uncertainty estimate.

- Gaussian Processes
- Bayesian Deep Learning

Bayesian Deep Learning

- weights are stochastic
- the output is also stochastic and therefore allows an uncertainty estimate for the prediction
- weights and biases are sampled based on Bayes theorem using e.g. Markov-Chain Monte Carlo methods

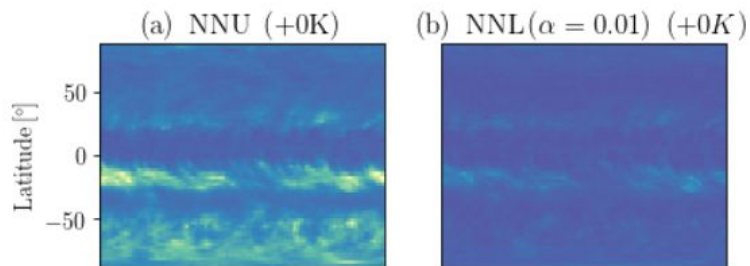
$$p(\theta|D) = \frac{p(D_y|D_x, \theta)p(\theta)}{\int_{\theta} p(D_y|D_x, \theta')p(\theta')d\theta'}$$



Jospin, Laurent Valentin, et al. "Hands-on Bayesian neural networks—A tutorial for deep learning users." *IEEE Computational Intelligence Magazine* 17.2 (2022): 29-48.

Physics-Informed Deep Learning

- Idea: Force your Deep Learning model to conserve energy/mass or apply another physical constraint
- Typically achieved by
 - constraining the model architecture
 - modifying the loss function



$$\mathcal{L}(\alpha) = \alpha \mathcal{P}(x, y_{\text{NN}}) + (1 - \alpha) \text{MSE}(y, y_{\text{NN}})$$

“Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling”, Beucler et al 2019, <https://arxiv.org/abs/1906.06622>

Coupling General Circulation Models with ML

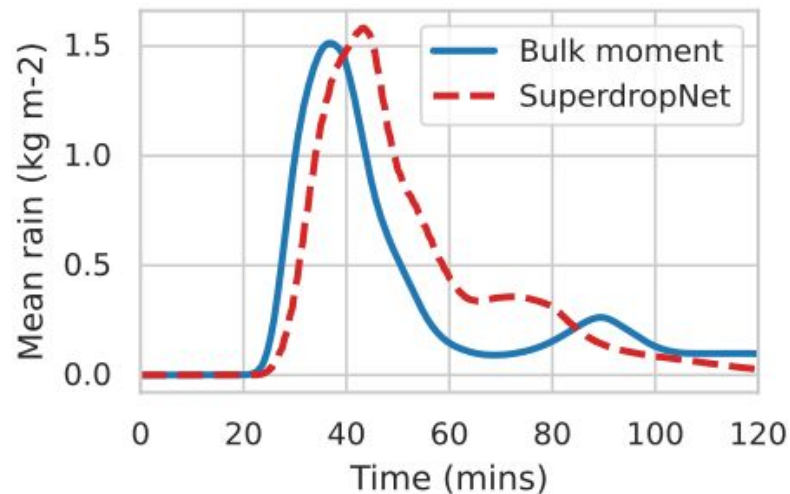
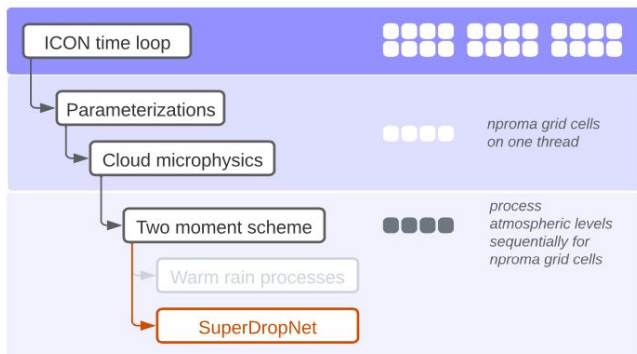
Potential:

- Replace parameterisations that rely on empirical assumptions with ML-based relationships that are learned from observations.
- Replace computationally expensive components with ML-based model

Challenges:

- Efficient Technical implementation of python code into FORTRAN or C++
- Offline vs online behaviour of ML model
- Application to problems outside of training data

Simulating rain in ICON with a neural network trained on superdroplet simulations



Arnold, C., Sharma, S., Weigel, T., and Greenberg, D.: Efficient and Stable Coupling of the SuperdropNet Deep Learning-based Cloud Microphysics (v0.1.0) to the ICON Climate and Weather Model (v2.6.5), EGU sphere [preprint], <https://doi.org/10.5194/egusphere-2023-2047>, 2023.

Deep Learning for Weather Prediction

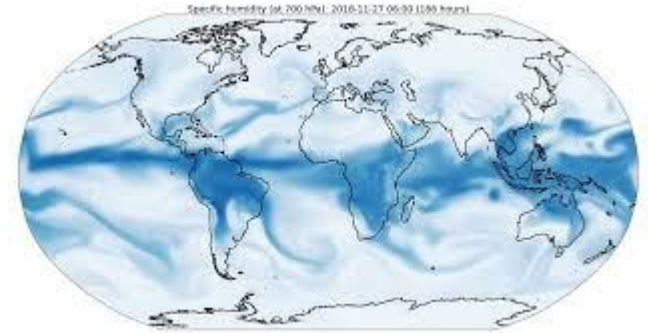
Several deep learning models now beat the numerical weather forecast by ECMWF for several days. These models are purely based on machine learning and do not take any physical constraints into account.

- GraphCast by google (GNNs)
- Pangu-Weather by HUAWEI (Transformers)
- FourcastNet by NVIDIA (Vision Transformers, Fourier Neural Operators)
- ...

check out <https://sites.research.google/weatherbench/> for an overview

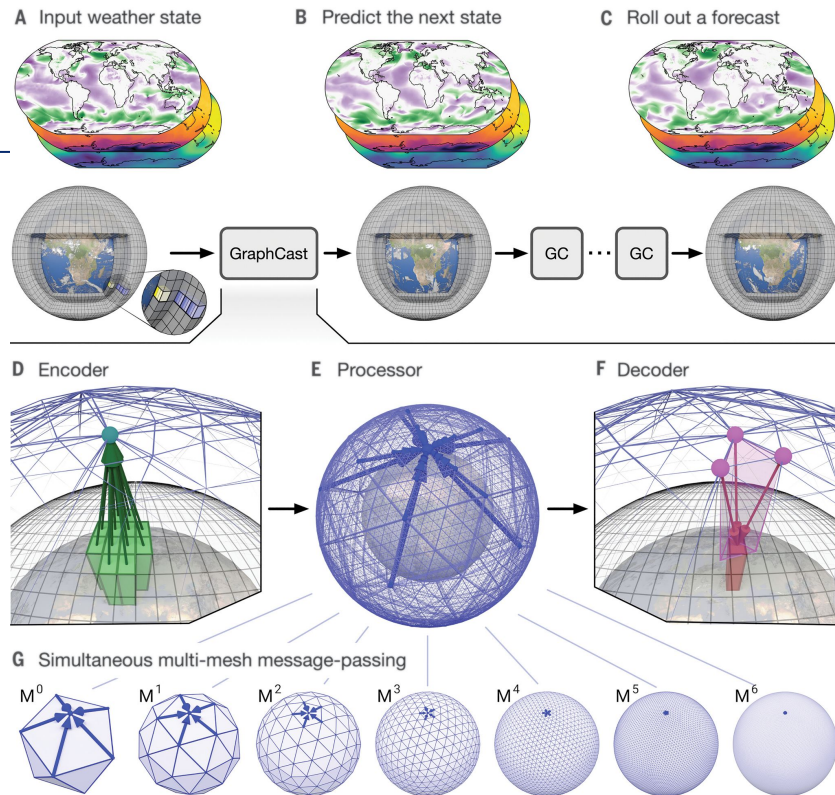
Google Graph Cast

https://www.youtube.com/watch?v=Q6fOIW-Y_Ss&t=13s



Google Graph Cast

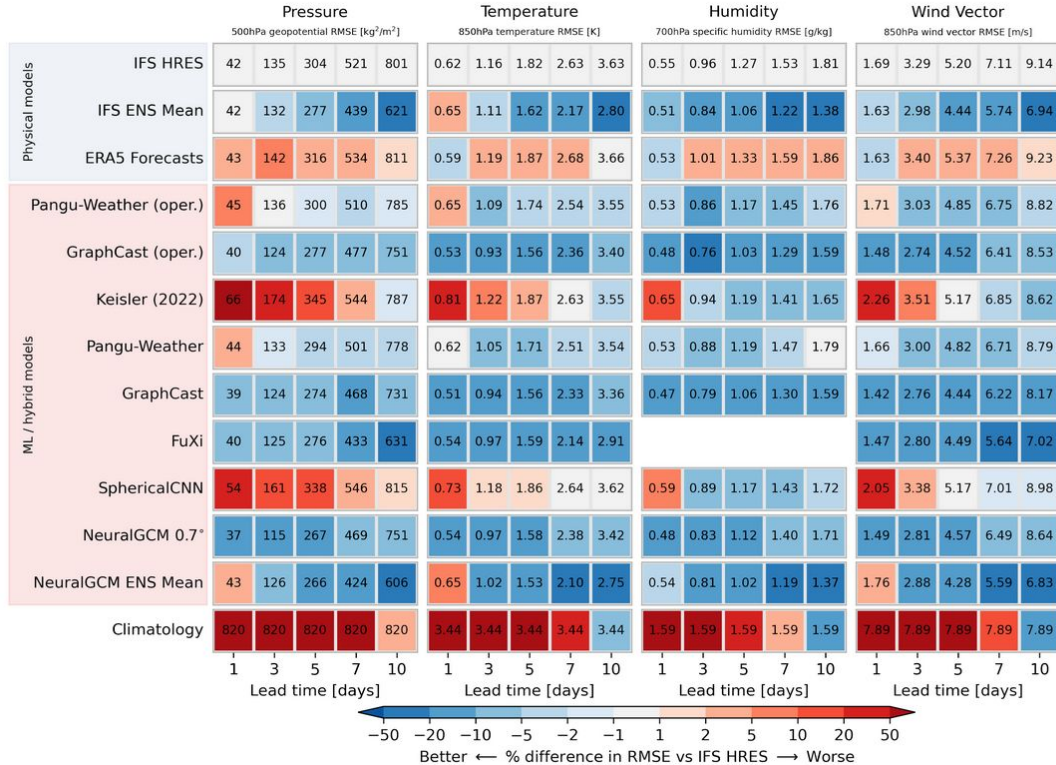
- Does predictions in 6-hour timesteps
- Processing on graph structure
- Processor takes various spatial scales into consideration
- open source



Remi Lam *et al.*, Learning skillful medium-range global weather forecasting. *Science* **382**,1416-1421 (2023).

DOI:[10.1126/science.adi2336](https://doi.org/10.1126/science.adi2336)

Deep Learning for Weather Prediction



Weatherbench:

<https://sites.research.google/weatherbench/>

Benefits

- Accuracy
- Performance
 - FOURCASTNET as example:
 - Training: 16 hours wall-clock time on 64 Nvidia A100 GPUs
 - Inference (forecasting): 24-hour 100-member ensemble forecast time reduced from 984,000 to 12.4 node-seconds
→ factor 80000 speedup!
 - extremely large ensembles possible (>10000)

The end of NWP? No (at least not yet)

- All ML models are trained on ERA5. A physically based training dataset is necessary
- no ML-model for high resolution, high quality regional forecasts
- Projections of future climate are currently not possible (what should the training data be?)

New Kid on the block: NeuralGCM

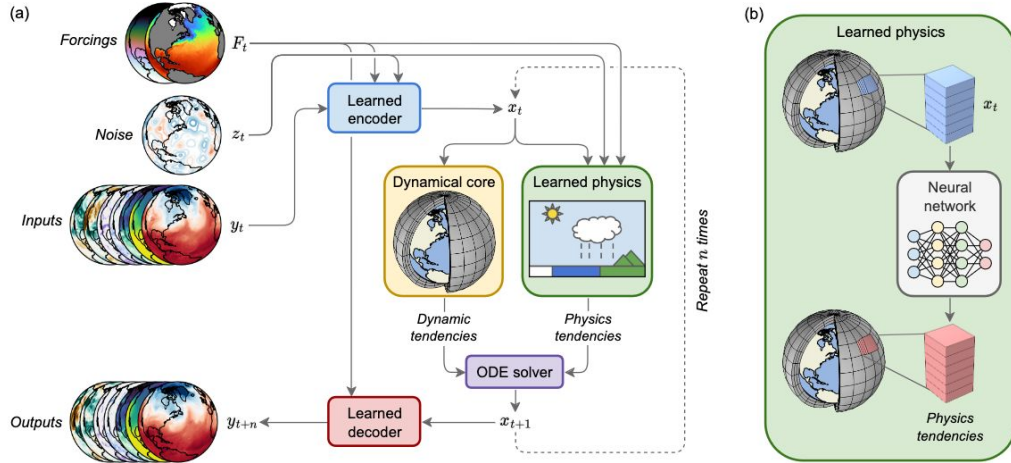


Fig. 1 Structure of the NeuralGCM model. (a) Overall model structure, showing how forcings F_t , noise z_t (for stochastic models), and inputs y_t are encoded into the model state x_t . Model state is fed into the dynamical core, and alongside forcings and noise into the learned physics module. This produces tendencies (rates of change) used by an implicit-explicit ODE solver to advance the state in time. The new model state x_{t+1} can then be fed back into another time step, or decoded into model predictions. (b) Inset of the learned physics module, which feeds data for individual columns of the atmosphere into a neural network used to produce physics tendencies in that vertical column.

preprint published 13 Nov 2023

<https://arxiv.org/abs/2311.07222>

Conclusion

- Fast moving science, state of the art changes every year
- Find what works best for you
- Play around with these tools and combine them

Conclusion

- Fast moving science, state of the art changes every year
- Find what works best for you
- Play around with these tools and combine them

Thank you.

Multilayer Perceptrons

Universal Approximation Theorem (Barren, 1993)

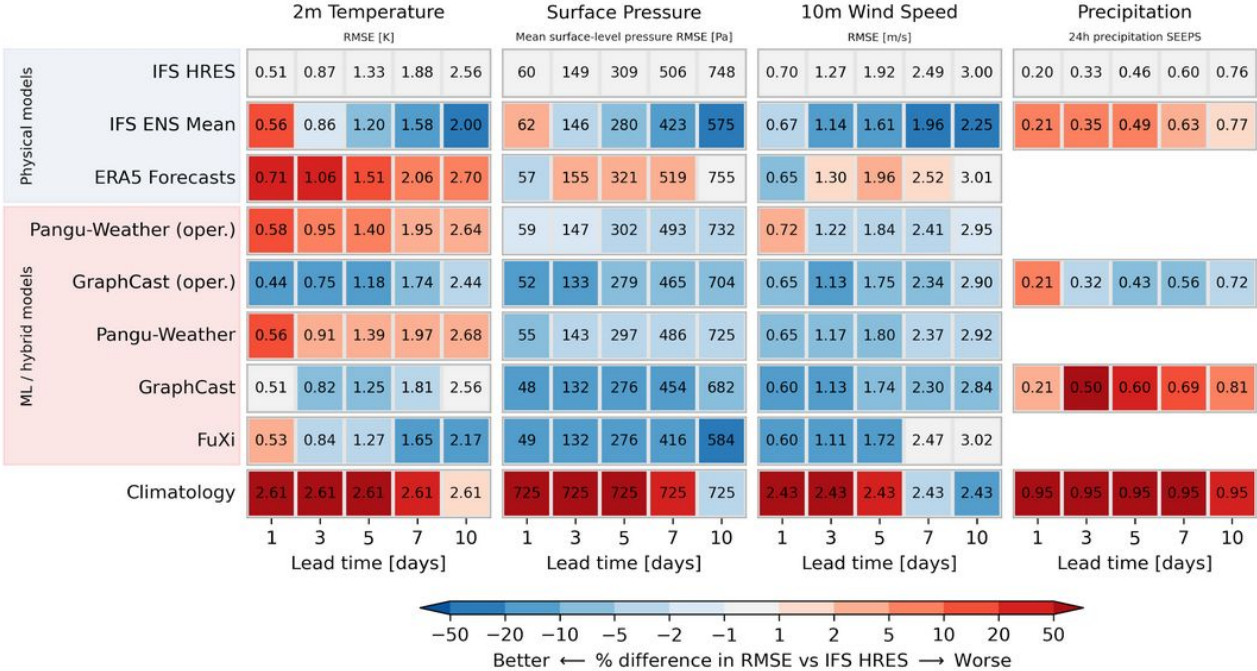
Any continuous function on a compact domain can be approximated by a neural network with only one hidden layer provided the activation functions used are bounded, continuous and monotonically increasing.

Possible in theory, but not feasible in practice.

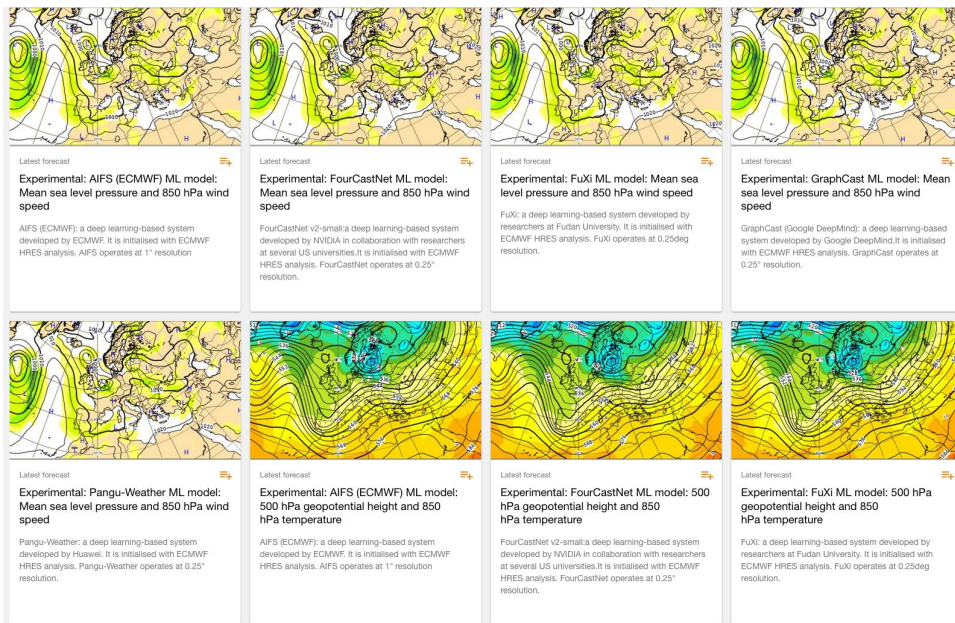
Other practical things you will need

- Regularisation to combat Overfitting
 - dropout layers
 - weight decay
 - ...
- Hyperparameter Tuning
- Different Loss functions
- Preprocessing data
 - shuffling
 - normalising
 - splitting into training, testing, validation datasets

Deep Learning for Weather Prediction



Already (kind-of) operational



<https://charts.ecmwf.int/>