Understanding Key Components of the Atmospheric Science Machine Learning Pipeline

David John Gagne

Machine Learning Scientist

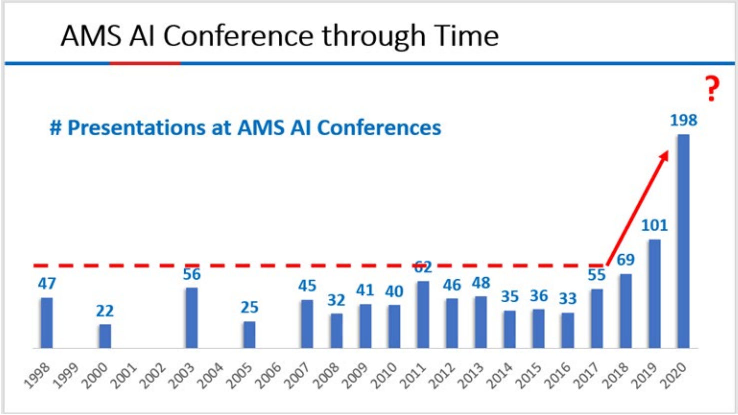
National Center for Atmospheric Research

NCAR
UCAR

April 08, 2020

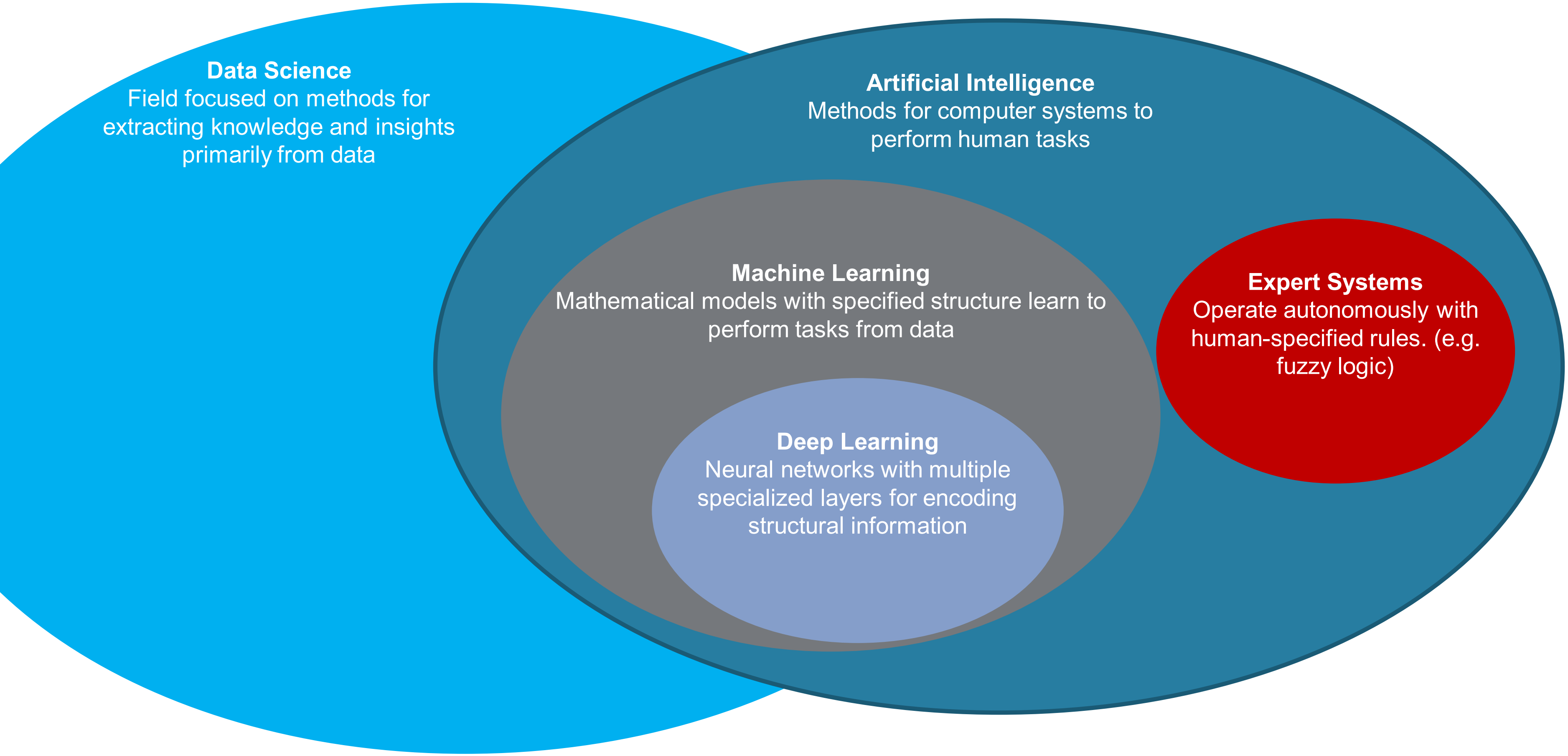
Motivation

- Interest in AI and machine learning in the atmospheric sciences has exploded in the past three years
- Much of the attention has been focused on the algorithms
- However, choosing the right ML algorithm is not sufficient for creating a successful AI/ML system
- Successful deployment of AI/ML requires making smart choices throughout the Machine Learning Pipeline
- Goal: Discuss the components of the ML pipeline and how to construct an effective AI/ML system for atmospheric science problems

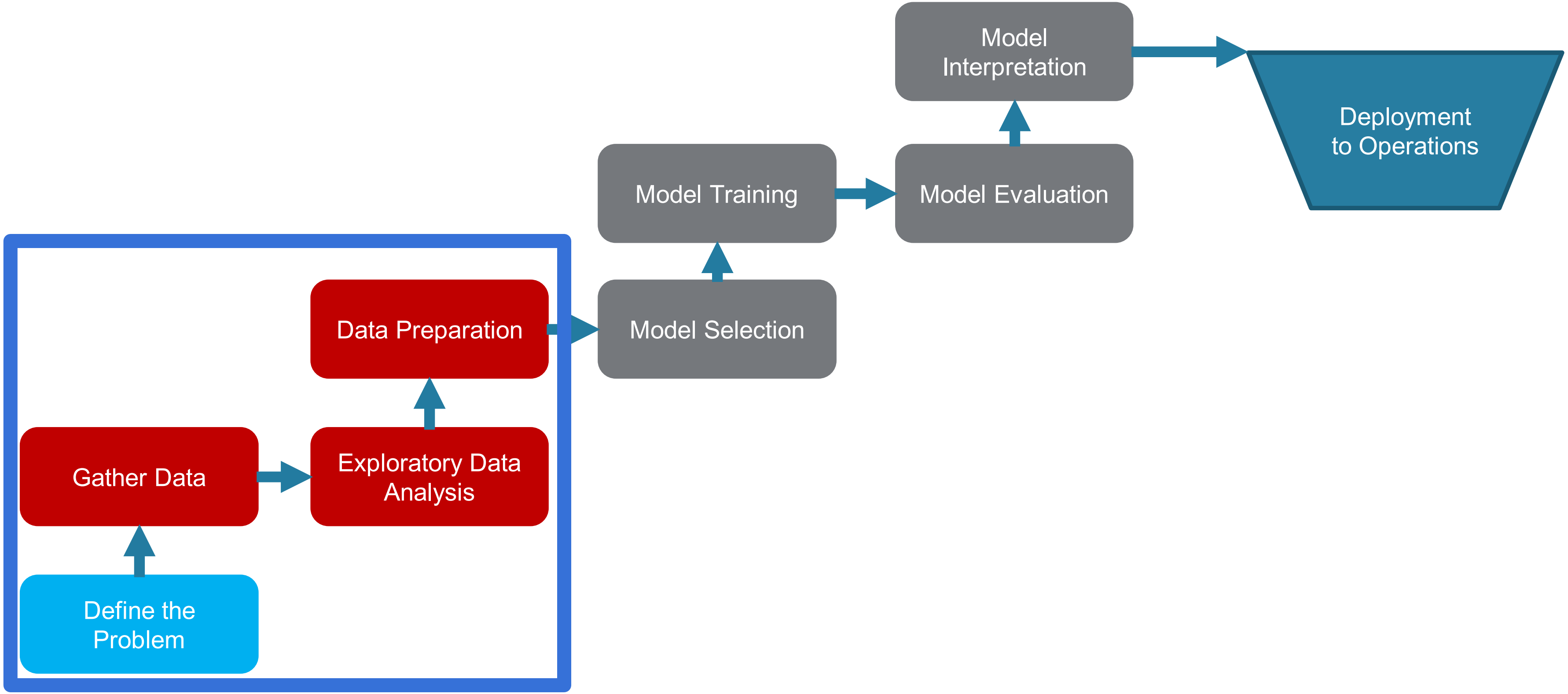


Courtesy Philippe Tissot

The Data Science Taxonomy



The Machine Learning Pipeline

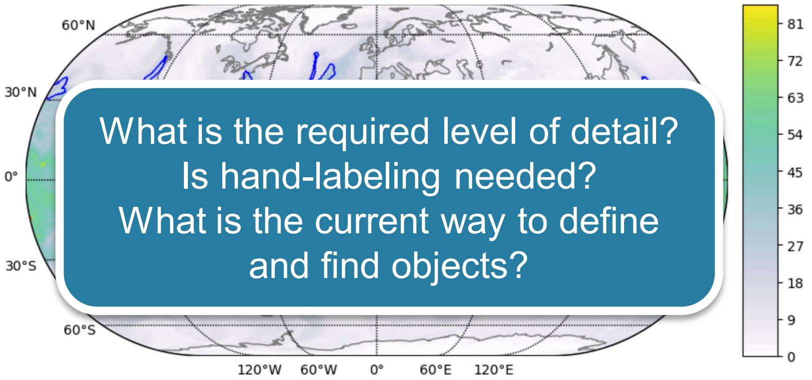


Defining the Problem

- The most important part of any machine learning project is defining the problem properly
- Questions to ask:

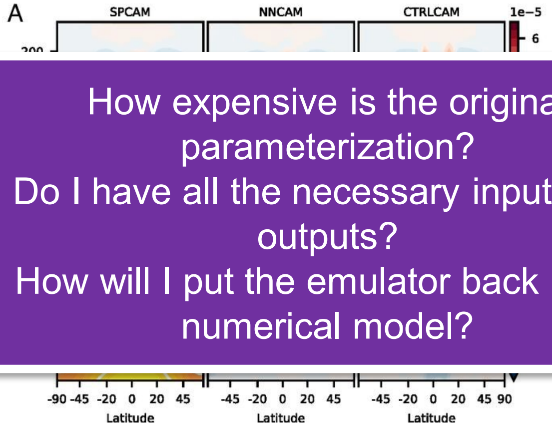
1. What are the ultimate goals of this project?
2. What are the specific inputs and outputs needed to achieve the goals?
3. What data are available for the inputs and outputs? What are the data limitations?
4. What are the problem constraints (time, space, latency, physical)?
5. How is the problem currently solved, and what are the limitations of those methods?

Machine Learning Problem Examples



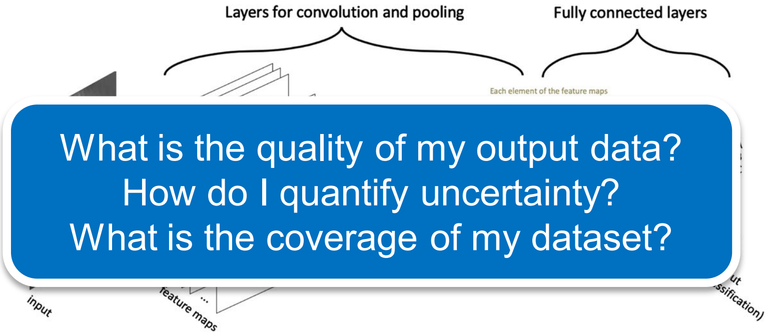
What is the required level of detail?
Is hand-labeling needed?
What is the current way to define and find objects?

Object Segmentation (Kurth et al. 2018)



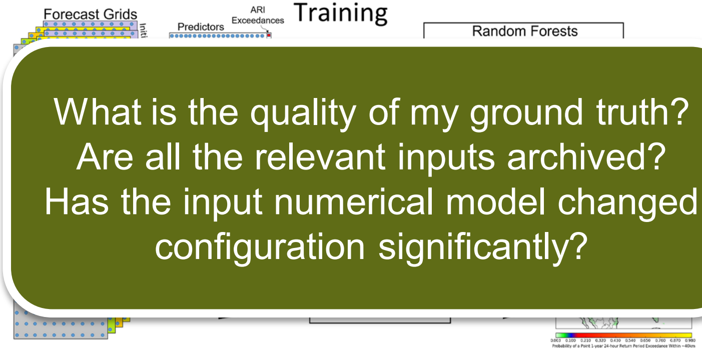
How expensive is the original parameterization?
Do I have all the necessary inputs and outputs?
How will I put the emulator back in the numerical model?

Parameterization Emulation (Rasp et al. 2018)



What is the quality of my output data?
How do I quantify uncertainty?
What is the coverage of my dataset?

Observation Diagnosis (Wimmers et al. 2019)



What is the quality of my ground truth?
Are all the relevant inputs archived?
Has the input numerical model changed configuration significantly?

Model Post-Processing (Herman and Schumacher 2018)

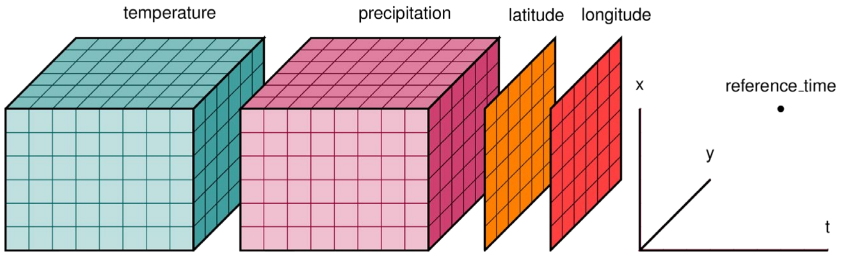
Data Gathering

Choose your data gathering adventure

	Use Existing Data	Gather Your Own Data	Generate Synthetic Data
Benefits	Long archive Freely available Retrieve necessary subsets Can compare different versions	Gather exactly what you need Control experiment design	Control properties of data Repeatable
Perils	File formats Lack of metadata/ provenance Inappropriate variables or pre-processing for problem Biased sampling	Expensive Quality of data gathering No access to past Your responsibility to avoid bad data sampling and processing practices	May be computationally expensive Not from real world Setting up infrastructure is time-consuming

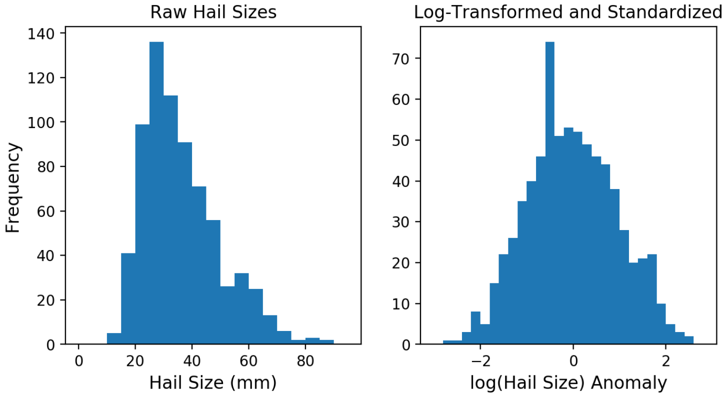
Data Preparation: Transformations

Reshaping and Sampling

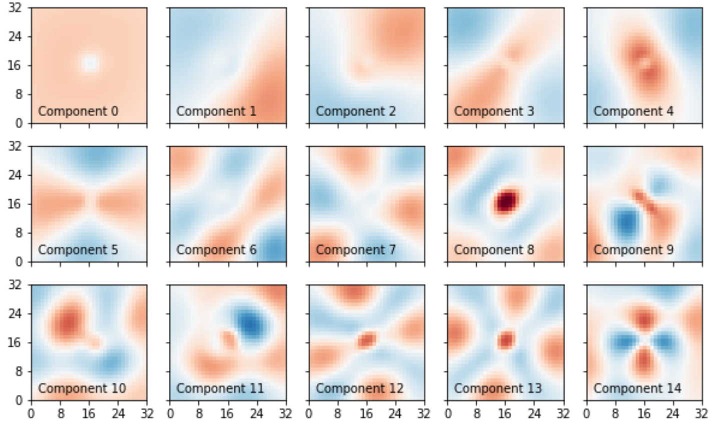


Time	Lat	Lon	Temp	Precip
0	35	-124	28	0
1	32	-94	15	24
2	45	-53	-2	5
…	…	…	…	…

Data Scaling and

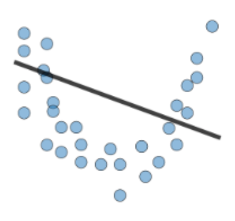
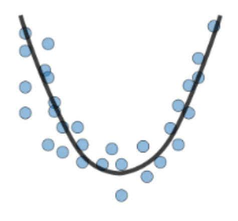
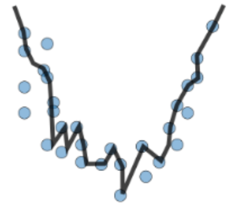
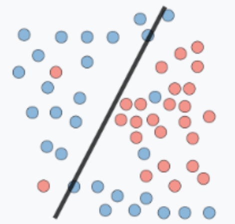
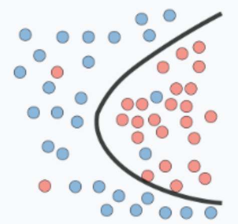
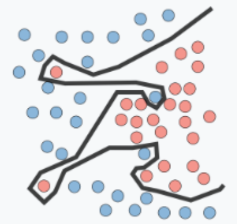


Dimensionality Reduction



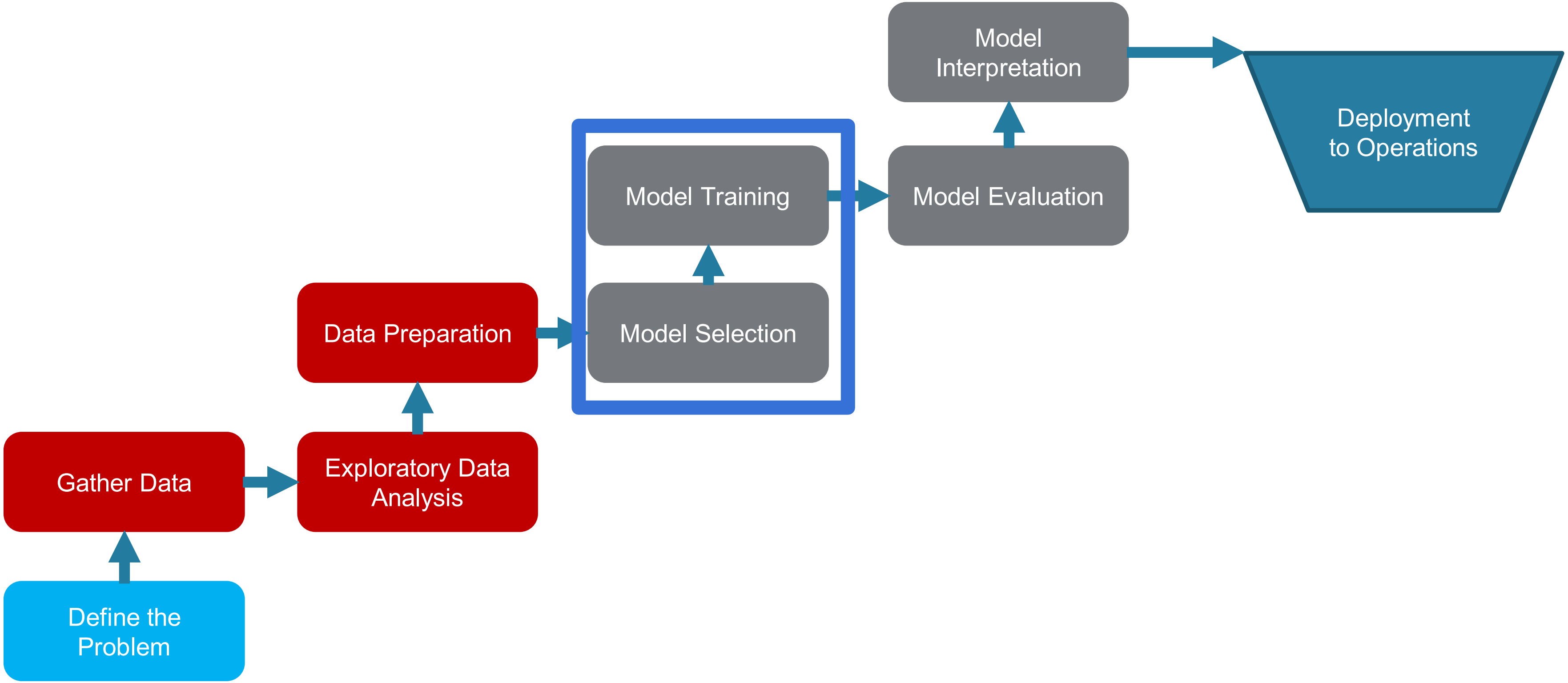
Pre-Processing: Training/Validation/Test Sets

- Goal: produce a ML model that will generalize, or perform well operationally.
- How do we estimate generalization ability?
- Training Set
  - Used to optimize a model's weights or structure for one set of hyperparameters
  - More complex models will almost always improve on training set scores
- Validation Set
  - Used to assess the performance of one or more models
  - Can be used to choose hyperparameters
  - Should be independent of training data unless cross-validation is used
- Test Set
  - Data unseen during training and validation
  - Should be used for final assessment and not model selection
- How to split the data
  - If data points are independent, random splits are fine
  - Splitting process should account for spatial and temporal dependencies

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-machine-learning-tips-and-tricks#diagnostics

The Machine Learning Pipeline



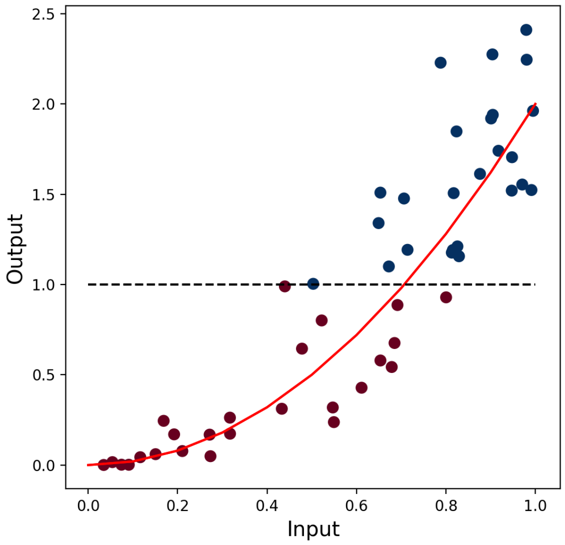
Models for Different Situations

Tabular Data

	Small Num. Features	Large Num. Features
Small Num. Examples	Linear Regression K-Nearest Neighbors	PCA+Linear Regression Decision Tree
Large Num. Examples	Neural Network Linear Regression Random Forest	Random Forest Gradient Boosting

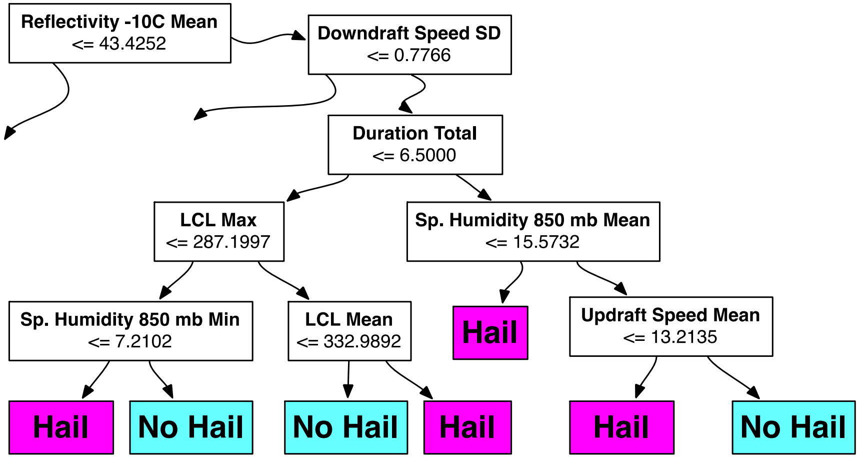
Spatio-Temporal Data

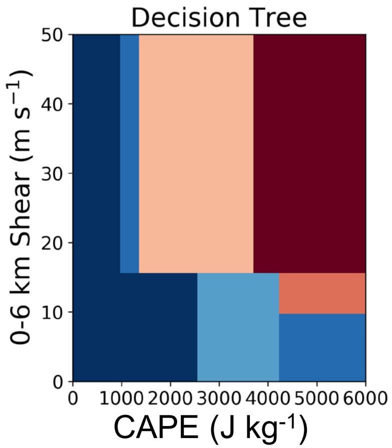
	Small Num. Features	Large Num. Features
Small Num. Examples	Gaussian Process	PCA+Gaussian Process
Large Num. Examples	Convolutional Neural Network	Convolutional Neural Network



Decision Trees

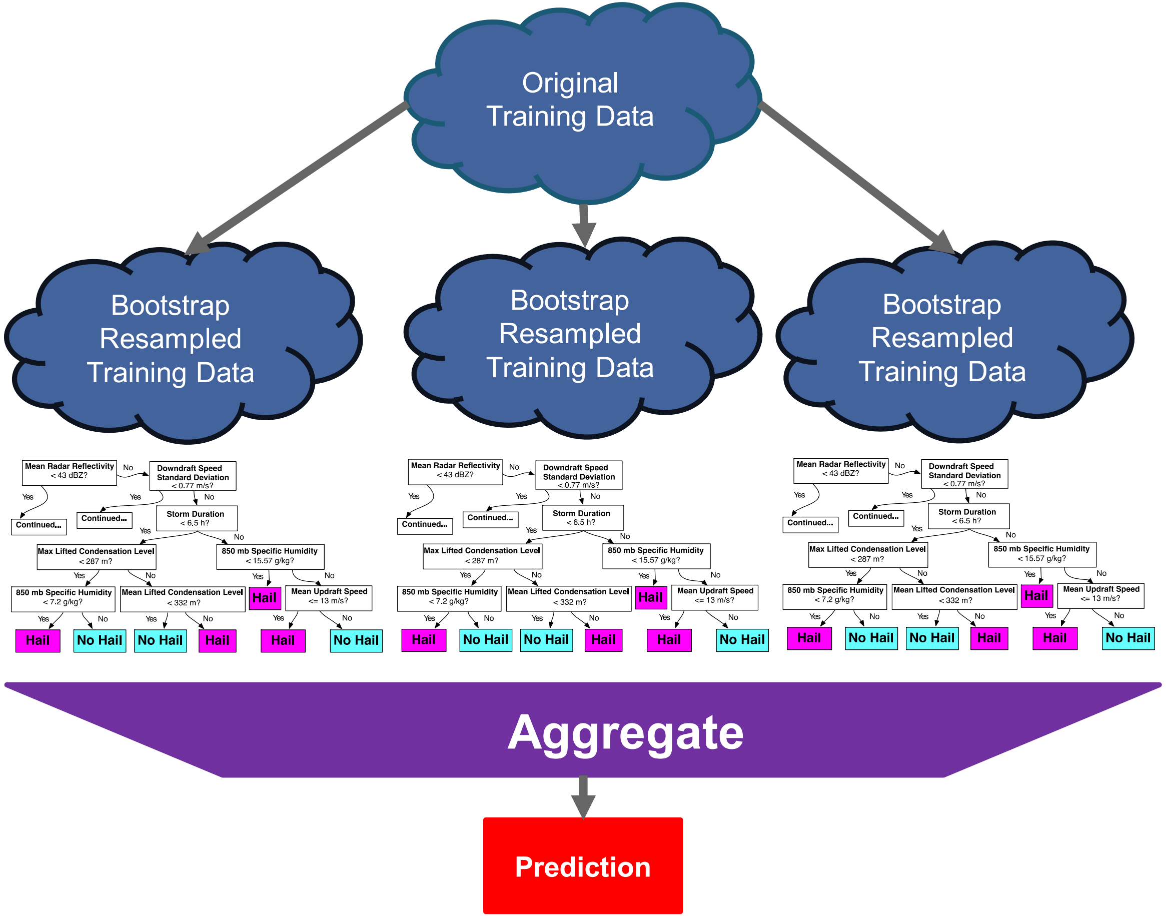
- Model that recursively partitions feature space into smaller, more similar regions
- Assign single prediction value to each subregion
- Decision Tree Training:
    - For each feature in data
        - For each unique split value
            - Split the data into two subsets based on candidate feature and value
            - Calculate error
    - Pick feature and threshold with largest decrease in error
    - Repeat for each branch until splitting is no longer possible or no longer decreases error
- Pros: Interpretable, automatic feature selection
- Cons: brittle, prone to overfitting, low accuracy





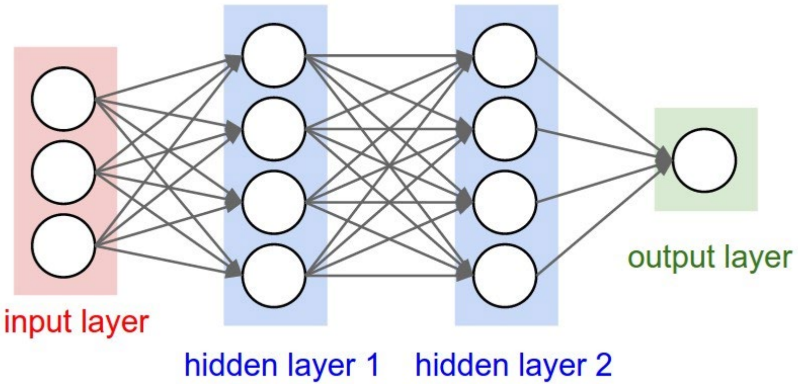
Random Forest

- Ensemble of randomized decision trees (Breiman 2001)
- Two forms of randomness
  - Bootstrap resample training data for each tree
  - Select random subset of input variables for evaluation at each node during training
- Special features
  - High prediction accuracy
  - Automatic feature selection
  - Fast and parallelizable
  - Requires little tuning

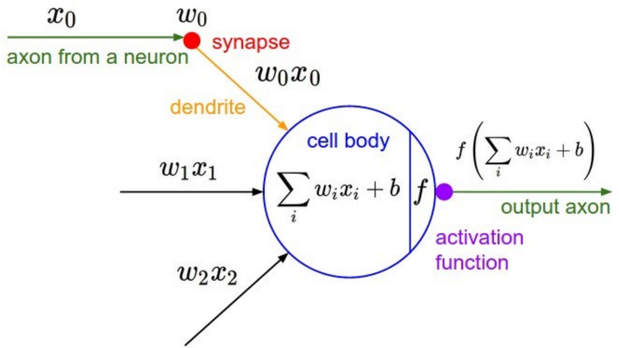


Neural Network Basics

Artificial Neural Network Structure



Perceptron (artificial neuron)

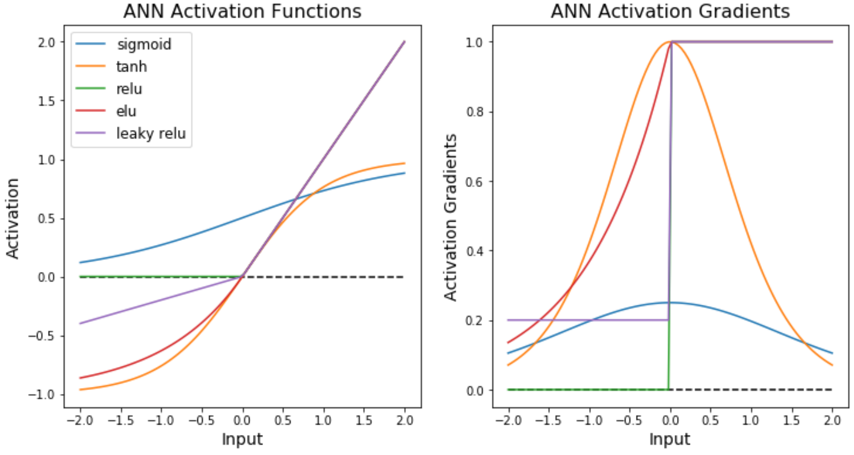


Training Procedure

1. Send batch of training examples through network
2. Calculate prediction error
3. Calculate error gradients back through layers and update weights
4. Repeat over all training examples until errors are satisfactory

Definitions

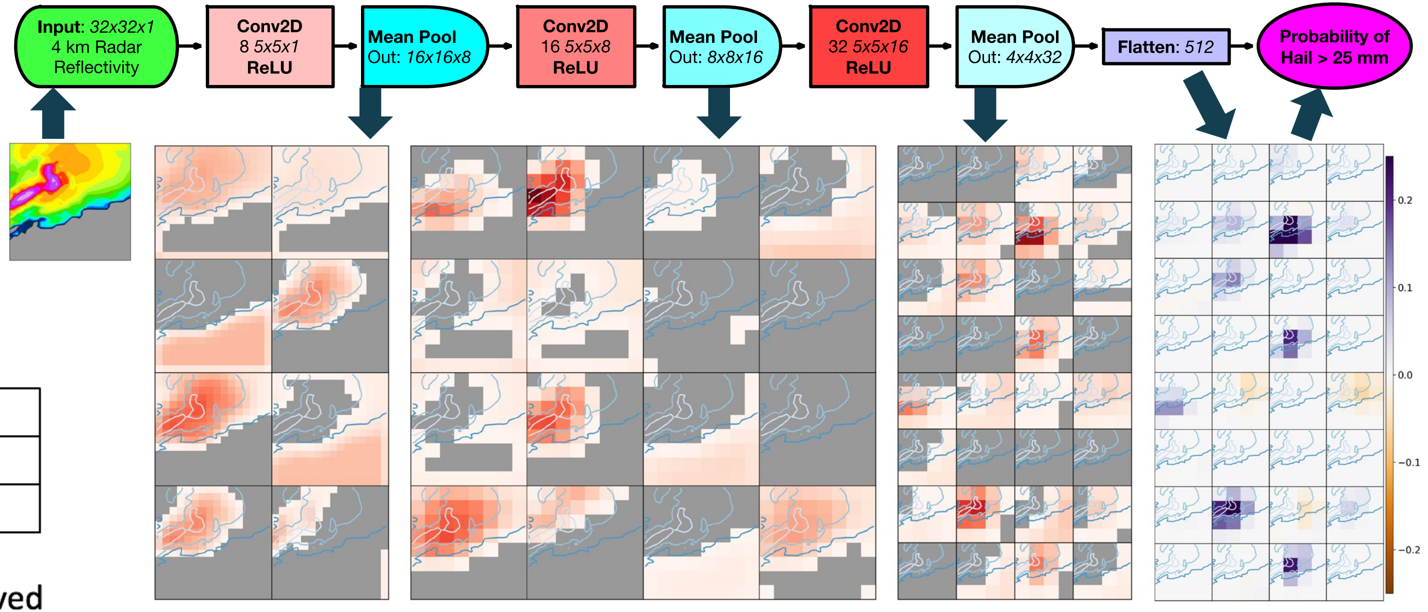
Batch: subset of training examples used to update weights
Epoch: One pass through all examples in training set

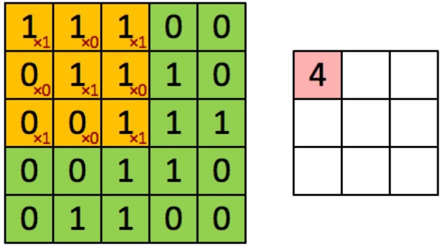


Images from http://cs231n.github.io/convolutional-networks/

Convolutional Neural Network

Deep neural network that encodes spatial information with iteratively optimized convolutional filters





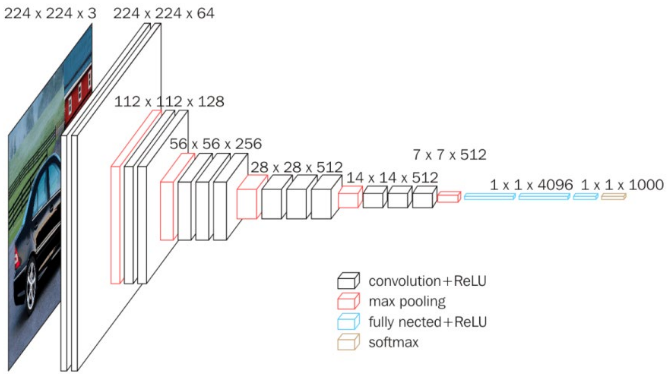
Image

Convolved Feature

Source: https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html

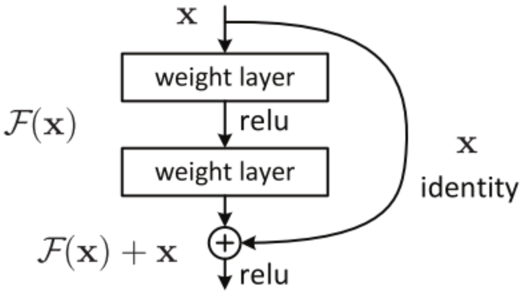
Convolutional Neural Network Model Zoo

VGG-16: A baseline CNN pattern



The VGG16 architecture of cycles of convolution and pooling layers is a good starting point for applying CNNs to many weather problems.

Residual Blocks



Residual blocks can replace regular convolutional layers and enable the training of extremely deep neural networks

U-Net

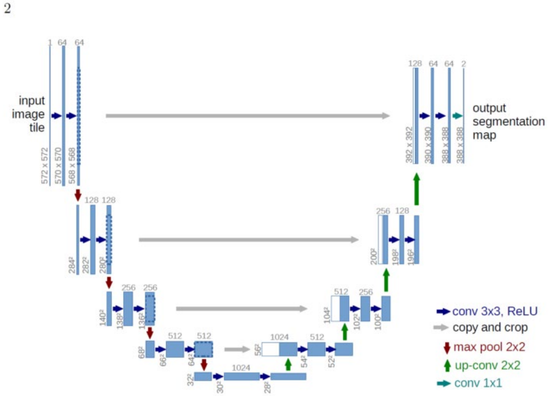
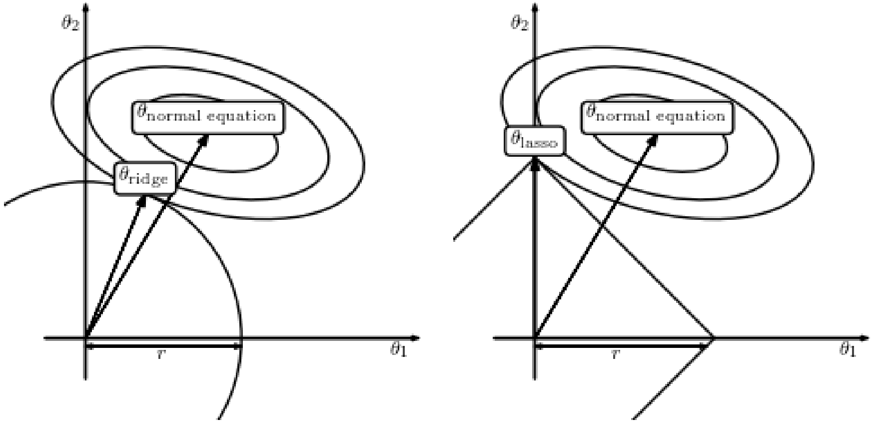
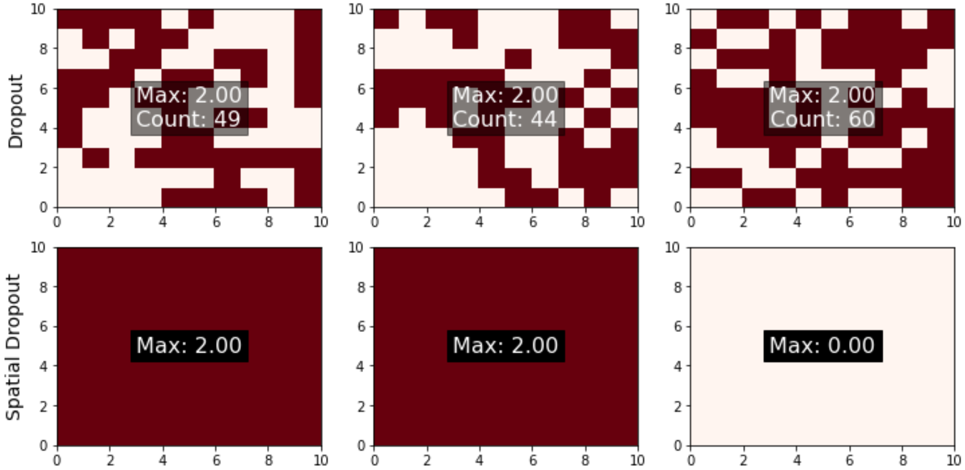


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

U-Nets perform image-to-image translation and can propagate features across different scales. Great for object segmentation and downscaling.

Regularization





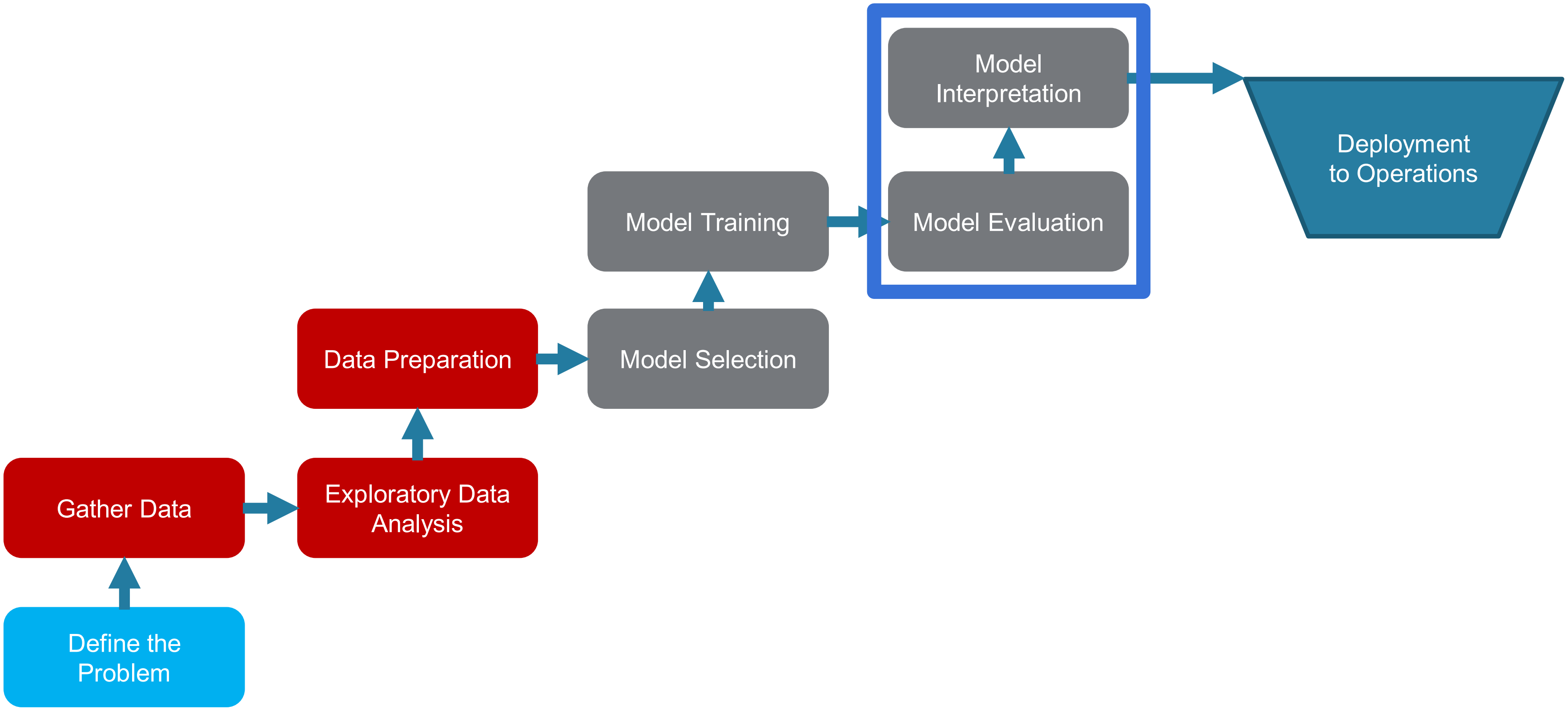
Ridge and Lasso Weight Decay

- Ridge: penalize with L2 norm, reducing all weight magnitudes
- Fits to noisy data more robustly
- Lasso: penalize with L1 norm, setting smaller weights to 0
- Performs feature selection

Dropout

- Randomly set input values to 0 with a fixed probability
- Can be applied to individual neurons or whole spatial channels
- Effectively creates a bootstrapped ensemble within one model

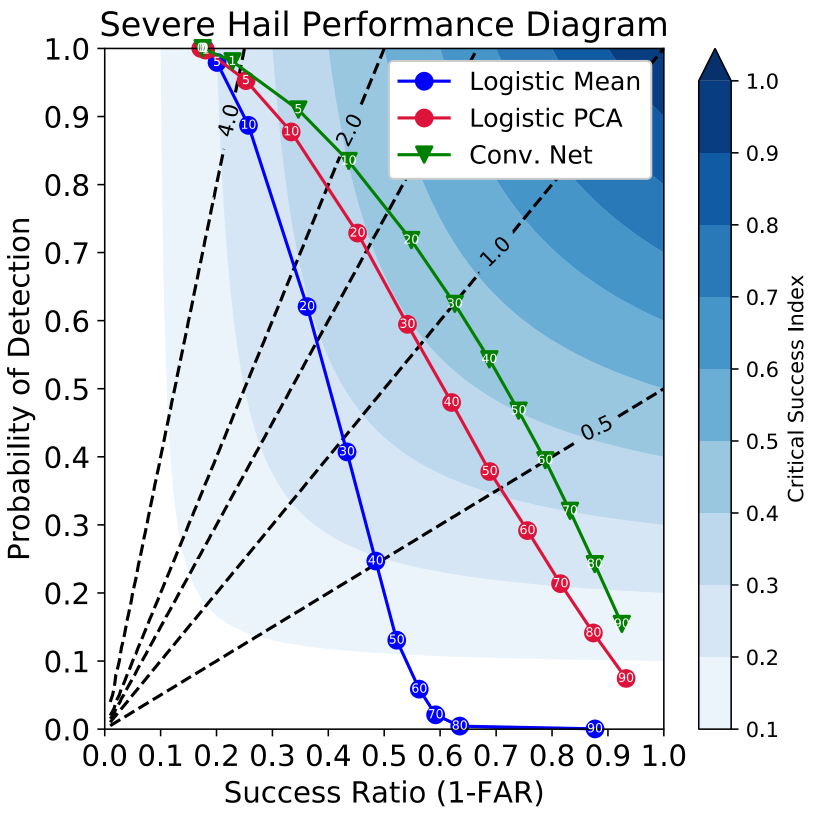
The Machine Learning Pipeline

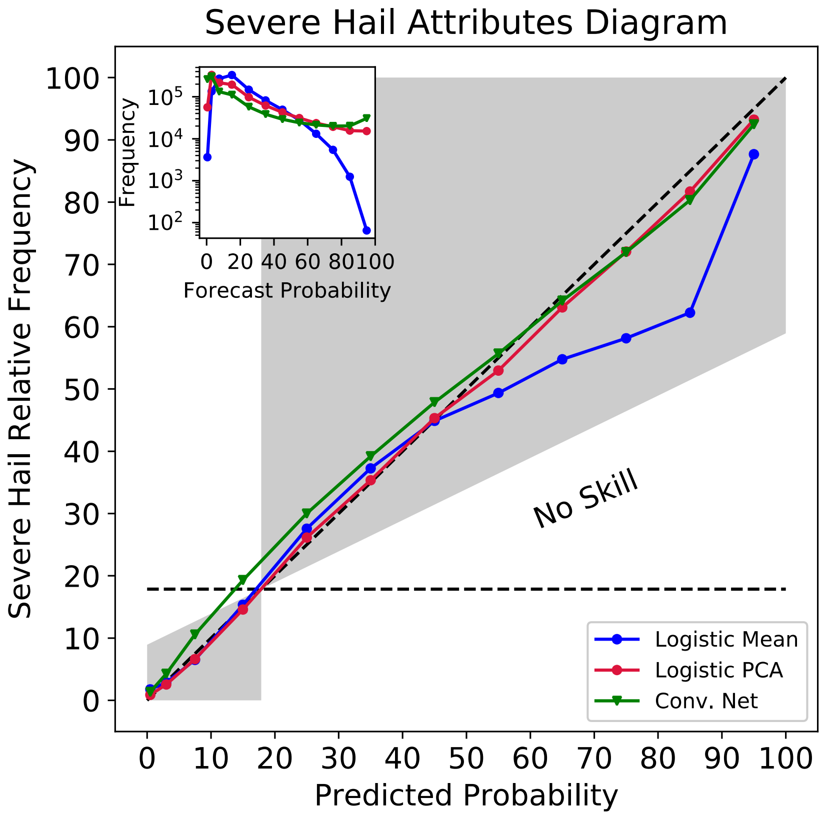


Evaluation Loss Functions and Metrics

	Classification	Regression	Distributions
Loss Functions (smoothly optimize the model)	Cross-Entropy Brier Score	Mean Squared Error Mean Absolute Error	Continuous Ranked Probability Score
Metrics (validate different aspects of performance)	POD, FAR, CSI, etc. Heidke Skill Score ROC Curves Brier Skill Score Reliability Diagrams	Correlation Coefficient Mean Error Skill Score 2D Histograms/Scatter plots	KL-Divergence Hellinger Distance Rank Histograms Quantile-Quantile plot

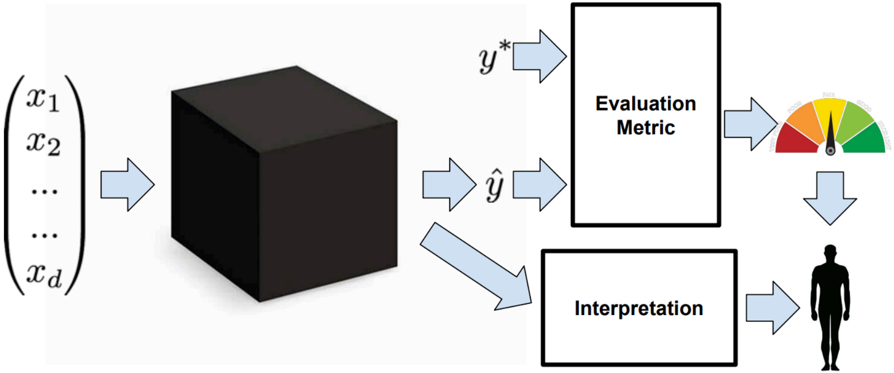
Verification Diagrams





Interpretable Machine Learning

- Machine learning models are sometimes considered "black-box" methods
- Prediction-generation is not transparent for many methods
- Interpretation methods provide additional information about *why* a ML model generates certain predictions
- Interpretation methods are a lower-order model of the full ML model



Source: Z. Lipton, 2016: The Mythos of Model Interpretability. https://arxiv.org/pdf/1606.03490.pdf

Gagne II, D.J., S.E. Haupt, D.W. Nychka, and G. Thompson, 2019: Interpretable Deep Learning for Spatial Analysis of Severe Hailstorms. *Mon. Wea. Rev.*, **147**, 2827–2845, https://doi.org/10.1175/MWR-D-18-0316.1

McGovern, A., R. Lagerquist, D. J. Gagne, G. E. Jergensen, K. L. Elmore, C. R. Homeyer, and T. Smith, 2019: Making the Black Box More Transparent: Understanding the Physical Implications of Machine Learning. *Bull. Amer. Meteor. Soc.*, **100**, 2175–2199, https://doi.org/10.1175/BAMS-D-18-0195.1

Spectrum of Interpretation Techniques

	Model-Agnostic	**Model-Specific**
Global	Permutation Variable Importance Partial Dependence Plots	Impurity Variable Importance Backwards Optimization
Local	LIME Sufficient Input Subsets	Saliency Maps Grad-CAM

Permutation Variable Importance

- Model-agnostic method for ranking input variables to model
- The set of values for each input variable is permuted, and the permuted data are sent through the model
- The change in a verification metric between the unpermuted and permuted data is the importance
- Can be calculated for multiple metrics
- Computationally intensive but parallelizable

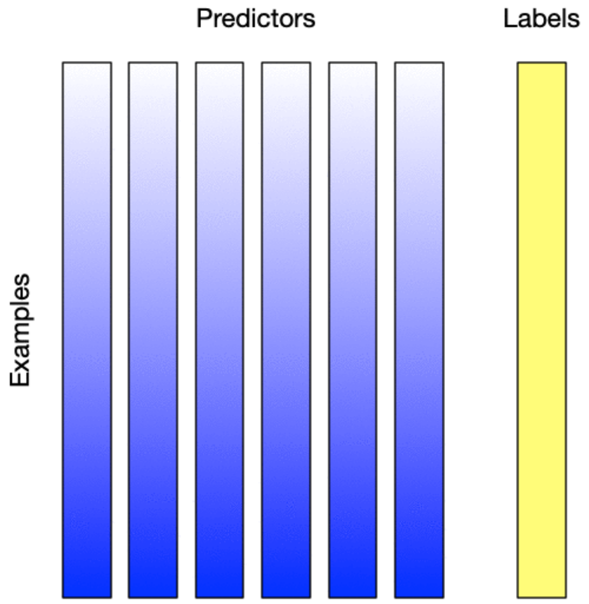
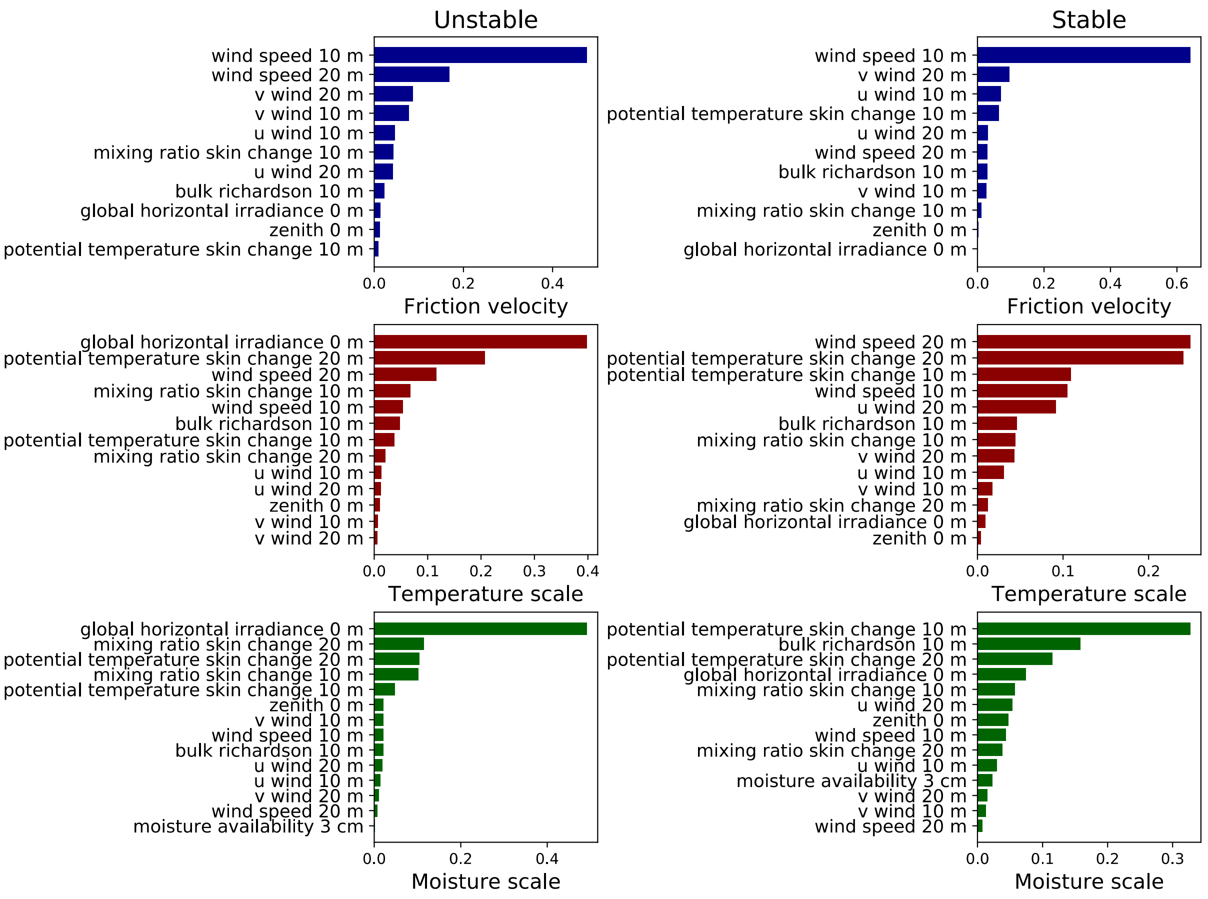


Illustration of single-pass permutation importance. Each predictor is permuted one at a time (blue boxes) and ranked by the difference in score from the original model and the model with permuted data (red shaded values at end).

Variable Importance Example: Surface Layer



Calculate permutation variable importance rankings with training data from stable and unstable regimes from Bulk Richardson.

Friction velocity

- Near surface wind speed most important
- Stable 20 m wind speed less important
- Potential temperature gradient more important at night

Temperature scale

- Diurnal cycle variables more important in unstable
- Wind speed more important in stable

Moisture scale

- Mixing ratio gradient more important in unstable regime
- Temperature gradient more important in stable regime

Partial Dependence Plots

1. Set all instances for one variable in a dataset to a single value

Temperature	Dewpoint	Pressure
280	10	986
280	14	1014
280	2	992
280	25	1025
280	6	950

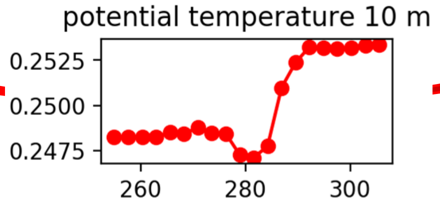
2. Feed fixed data through model

Machine Learning or Physical Model

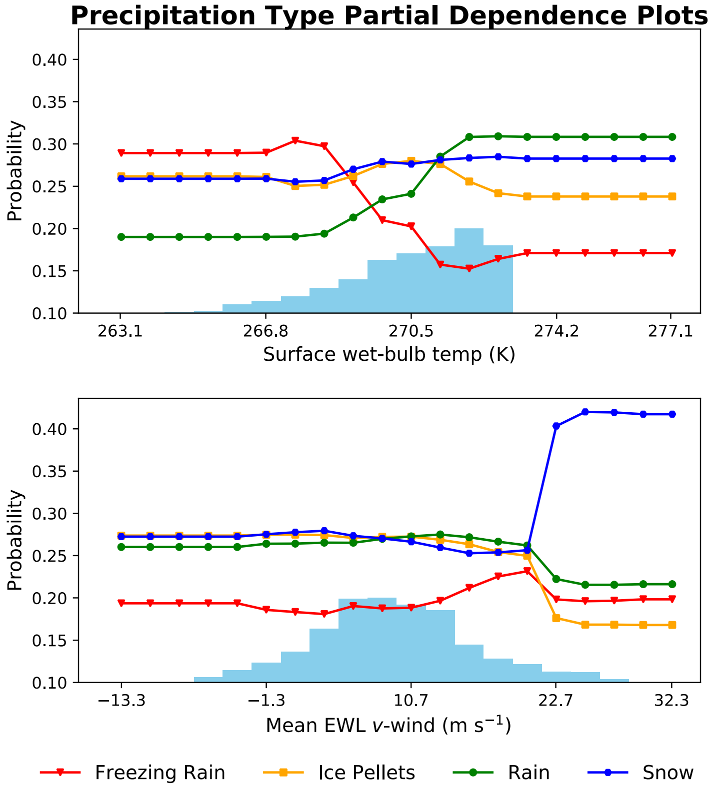
3. Calculate mean prediction for fixed value

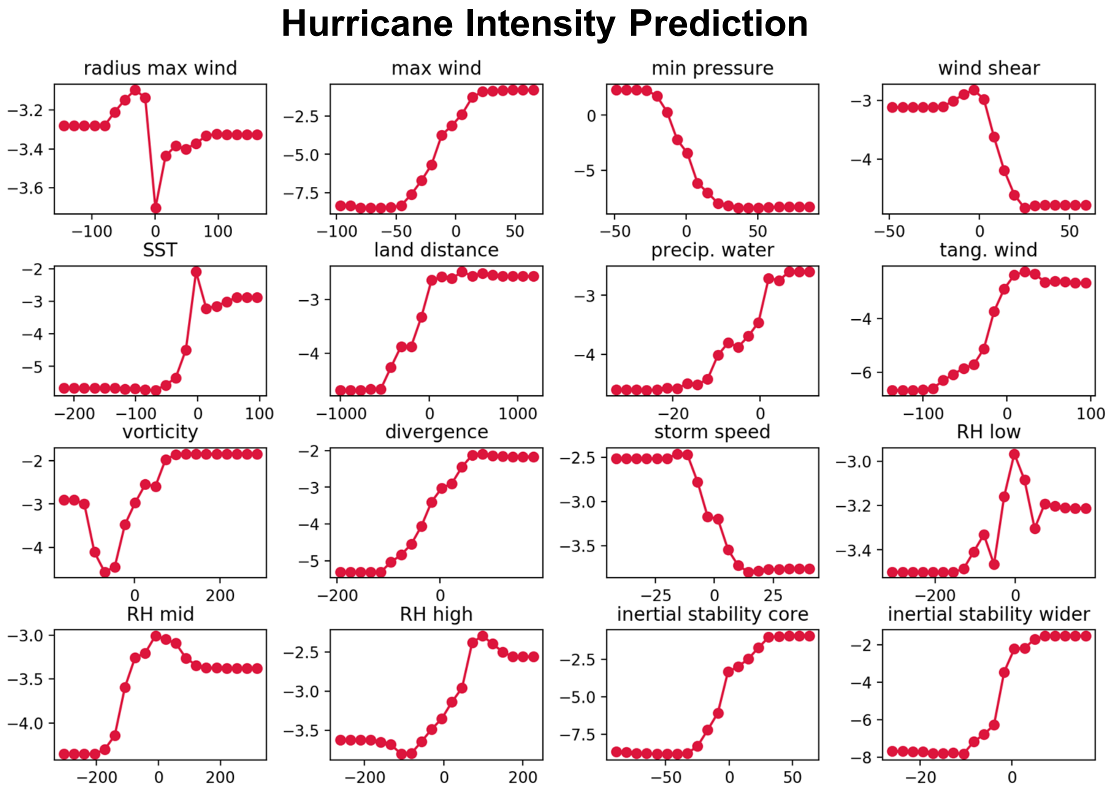
Mean Prediction

4. Repeat process for range of input values



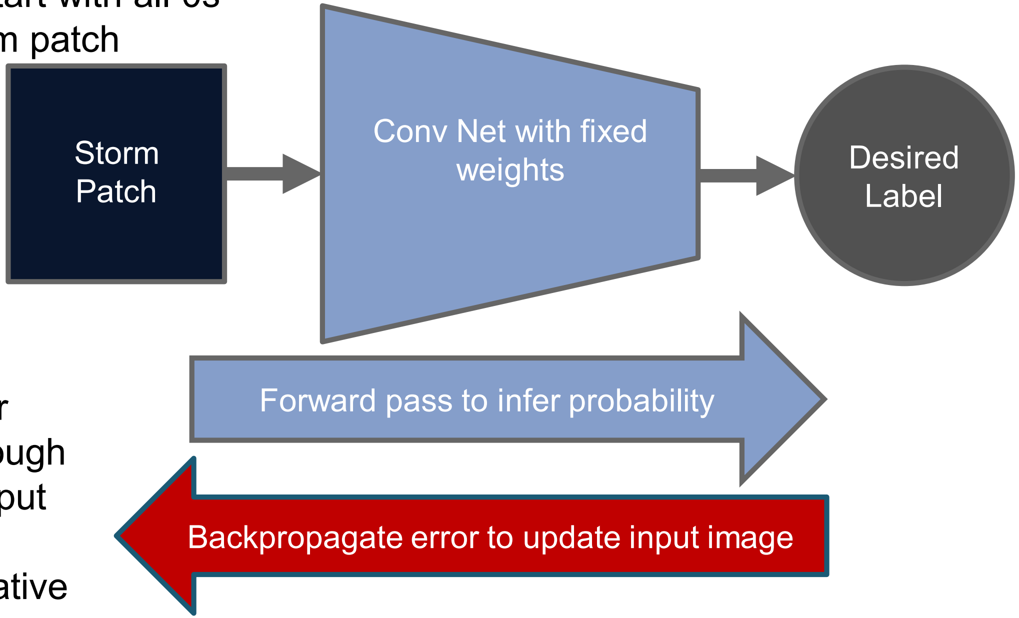
Partial Dependence Examples





Feature Visualization by Optimization

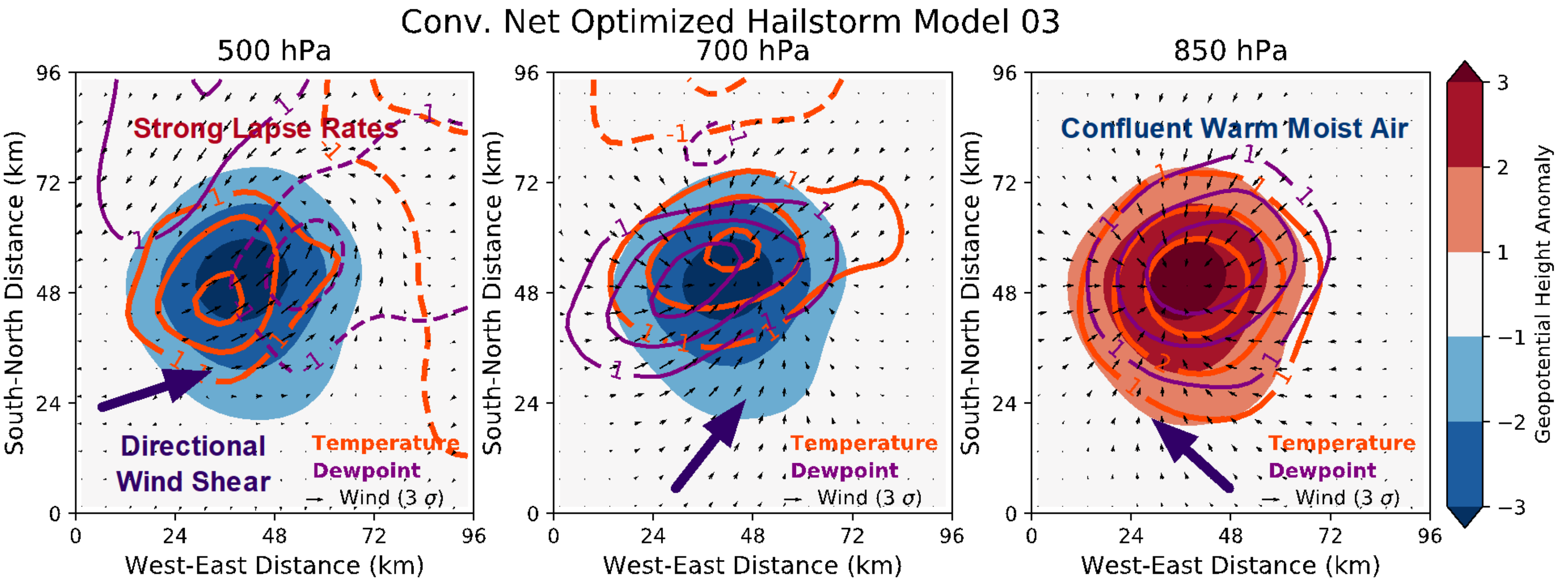
1. Start with all 0s storm patch



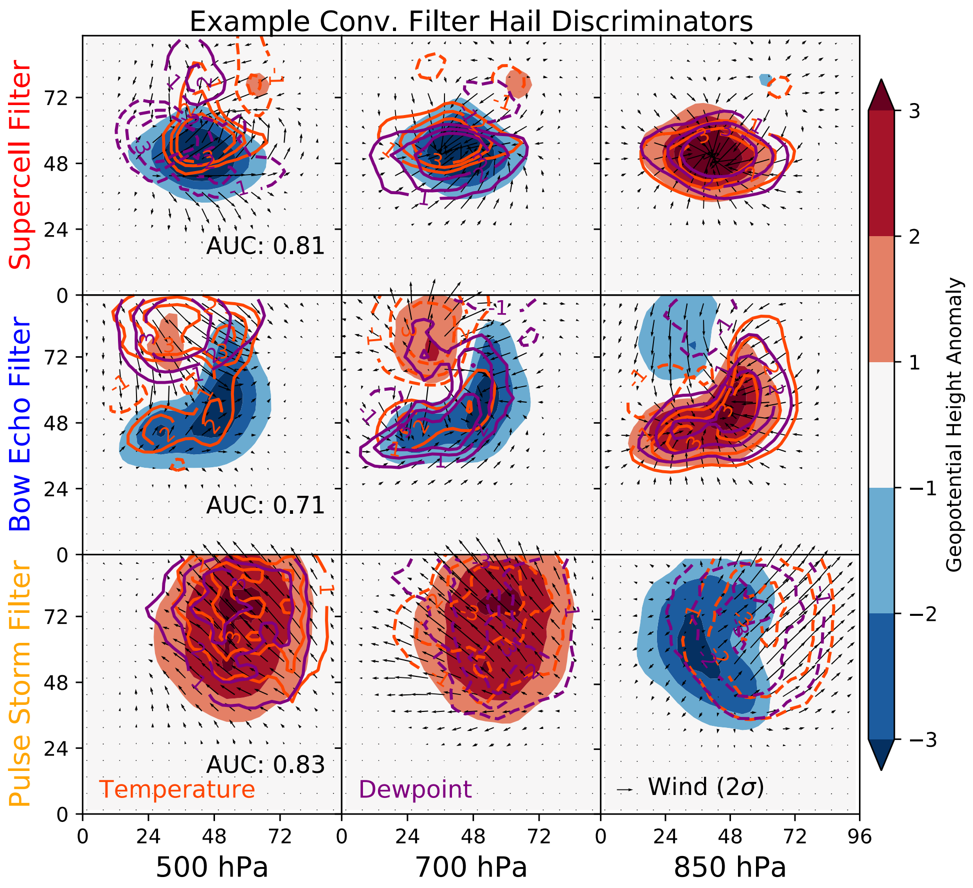
2. Send input forward through net to get error
3. Pass error back through net to get change in input
4. Update input by subtracting error derivative

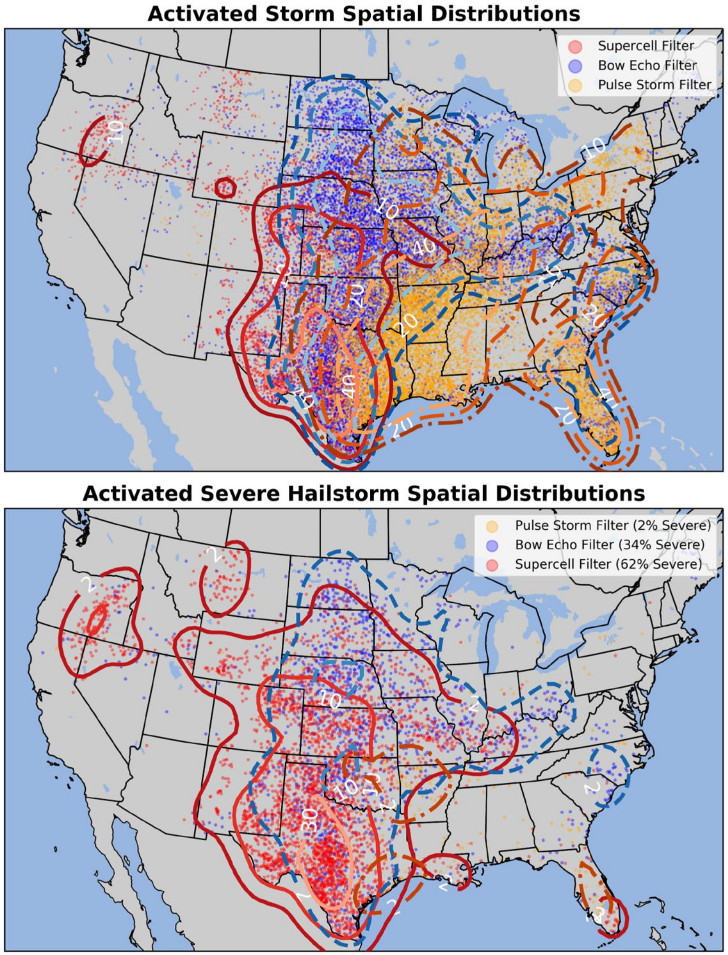
Repeat steps 2-4 until prediction matches desired output

Conv Net Optimal Hailstorms

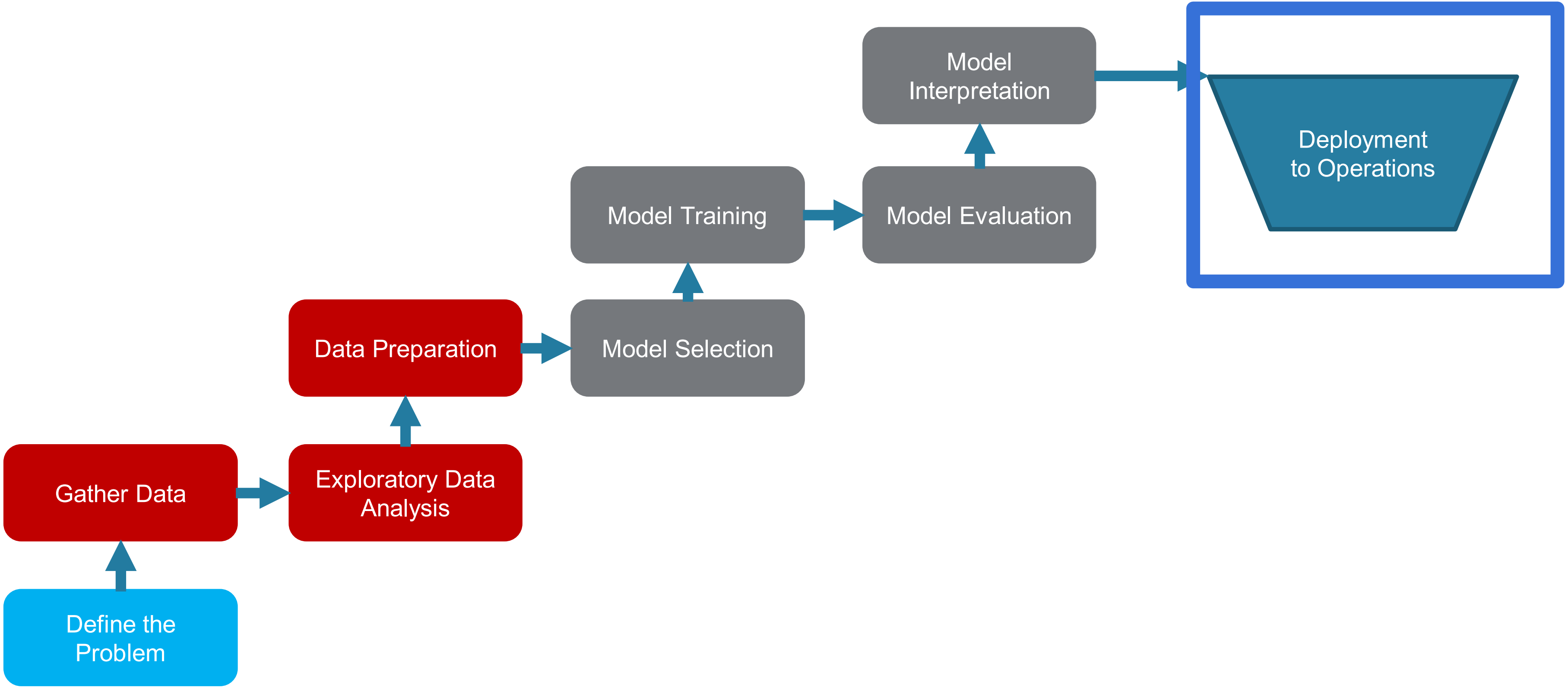


Storm Neurons





The Machine Learning Pipeline



Transitioning Machine Learning to Operations

- "Performance is not outcomes" (https://lukeoakdenrayner.wordpress.com/2019/01/21/medical-ai-safety-doing-it-wrong/)
  - A high test set score does not guarantee improved outcomes in practice
  - A new tool may worsen outcomes by negatively impacting other parts of the process
  - E.g., increased use of automation degrades diagnostic skills (Snellman 1977)
- Operational machine learning satisfies different constraints than research
  - Low latency: machine learning systems should run as fast as possible, including processing real-time data
  - Reliable: should handle data delays and outages, not have too many data dependencies
  - Consistent: output should generally complement other sources of information and provide justification for disagreements
  - Actionable output: output should be easily interpreted by end users and assist in their decision-making process

How can we maximize the potential of ML in Earth System Science?

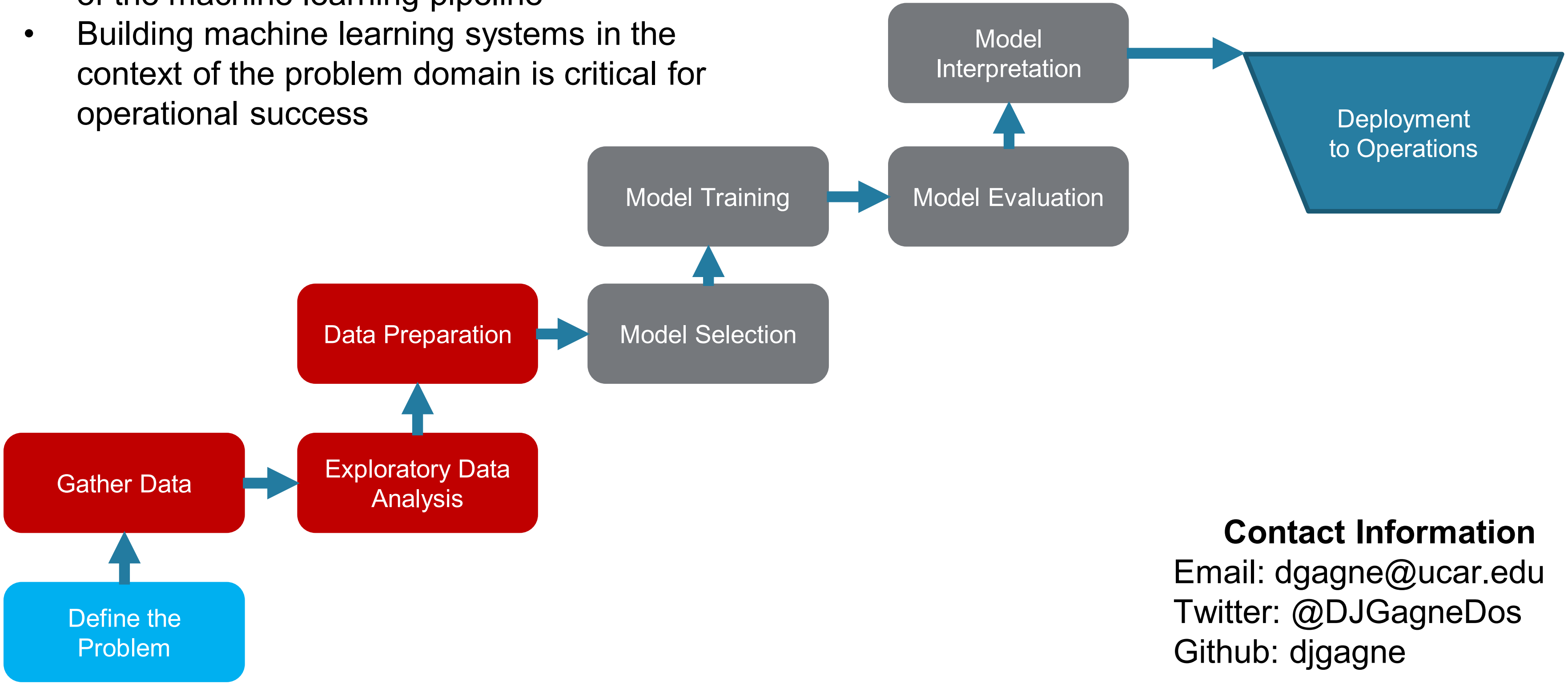
Invest in people
Building ML systems is very labor intensive
Need people with dual expertise in Earth System Science and ML
Few ready but many in the pipeline
Need more classes, tutorials

Invest in infrastructure
HPC workloads becoming more bursty, read-heavy, interactive
Model codes need to be more modular, accessible
Rethink how and how much model output we store
Need more tools to debug ML/identify feedbacks

Invest in patience
Quality research takes years
Essential infrastructure still being built/envisioned
Conference and journal papers are lagging indicators
ML won't change everything tomorrow, but could be an essential in 5-10 years

Summary

- Machine learning practitioners should rigorously decide how to implement each step of the machine learning pipeline
- Building machine learning systems in the context of the problem domain is critical for operational success



Contact Information
Email: dgagne@ucar.edu
Twitter: @DJGagneDos
Github: djgagne