Machine Learning Specific to Climate and Weather Applications

Part 1: Christina Kumler

CIRES University of Colorado Boulder & NOAA/OAR/ESRL/Global Systems Division

Part 2: Imme Ebert-Uphoff

CIRA & Dept. of Electrical and Computer Engineering, Colorado State University

NOAA-STAR seminar - July 18, 2019

PART 1

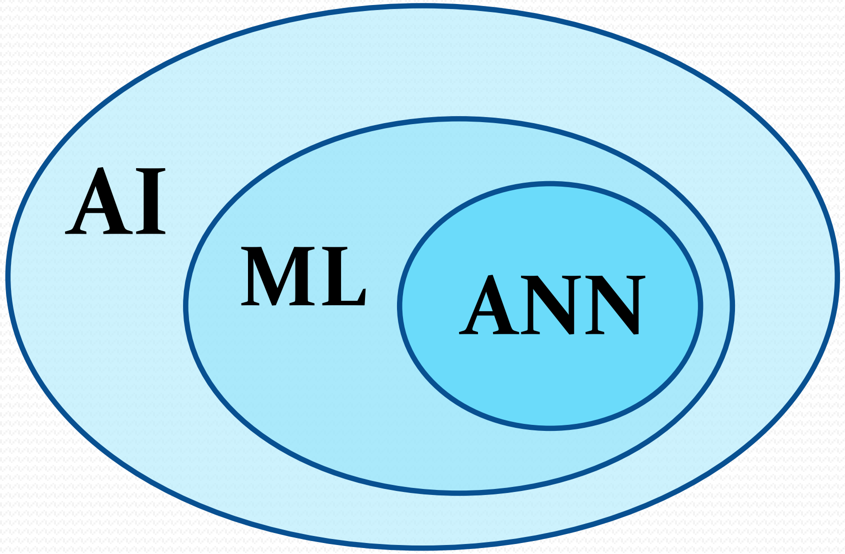
GSD's Machine Learning Cyclone Region of Interest (ROI) Project

Christina Kumler[1,3], Jebb Stewart[2,3], Lidia Trailovic[1,3], Isidora Jankov[2,3], and Mark Govett[3]

Christina.E.Kumler@noaa.gov

[1]CIRES University of Colorado Boulder, [2]CIRA Colorado State University, [3]NOAA/OAR/ESRL/Global Systems Division

Very Beginning - Acronyms



AI = Artificial intelligence:

- Goal: machines that "work and react in intelligent ways".
- Fairly abstract.

ML = Machine learning:

- Algorithms that enable a computer to automatically learn from experience – primarily from data samples - without being explicitly programmed.
- **Subset of AI**. That's what we typically want for weather/climate applications. (But "AI" sounds more impressive, and is technically correct, too.)

ANN = Artificial Neural Network

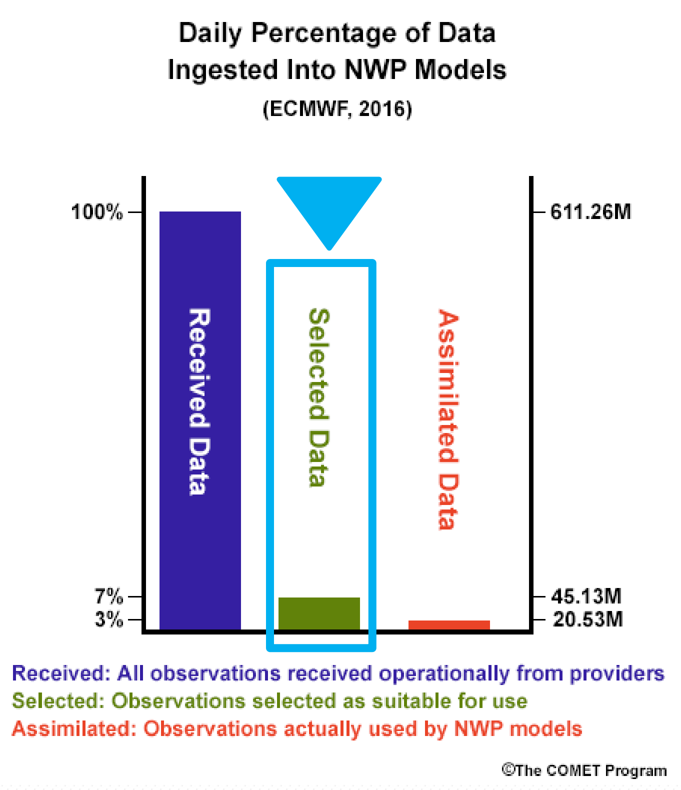
- Very versatile and powerful ML method, especially for image/video processing.

DL = Deep Learning

- ANN with many layers, i.e. of high complexity.

So, Why Machine Learning?

- Machine learning is not new
- Making a comeback because of amounts of data as well as high performance computing (HPC) abilities
- Machine learning is a broad term:
  - Neural networks
  - Deep learning
  - Random Forest
  - Regression Methods
  - Clustering



So, Why Machine Learning?

- Deep Learning has shown to be a very good tool when used on the following:
  - Speed up old methods (ex: heuristics)
  - Identify similar clusters without “supervision”
  - **Identify patterns with “supervision”**
- We need labels!
- We need balanced Datasets!
- We have tons of satellite data!
- We have an understanding of what we expect!

xkcd.com

So, Why Machine Learning?

- Remember HPC? It helps us build models quickly
  - GSD's theia resource of **tremendous** value
- Bottom line: Could not do what we do without it

Theia

- 100 nodes
- Each node has two 10 core Haswell processors
- Each node has 256 GB of memory
- Each node has 8 Tesla P100 (Pascal) GPUs
  - 16 GB Memory Each

	CPU	GPU
Hardware	Two 10 core Haswell Intel	8 Tesla (P100)
Training (per epoch)	11.5 hours	3 minutes
Complete training (~ 70 epochs)	~ 5 weeks	~ 3 hours
Inference for single input	1 second	40 milliseconds

Table to left shows computational comparisons between CPU and GPU for our ROI project training and inference stages

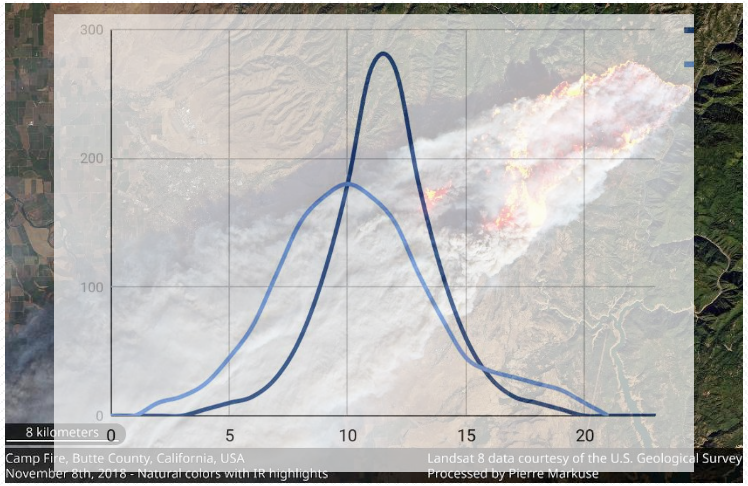
ROI and Deep Learning is Just ONE Area

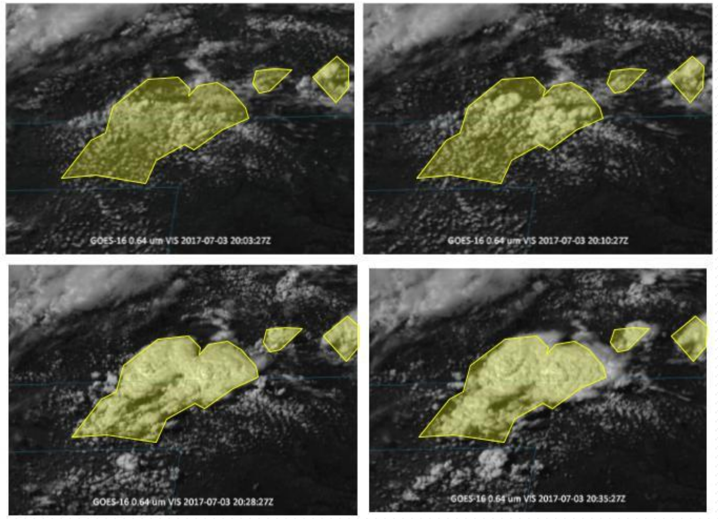
Machine Learning in GSD

Soil Moisture – Random Forest

Convection Initiation – Deep Learning

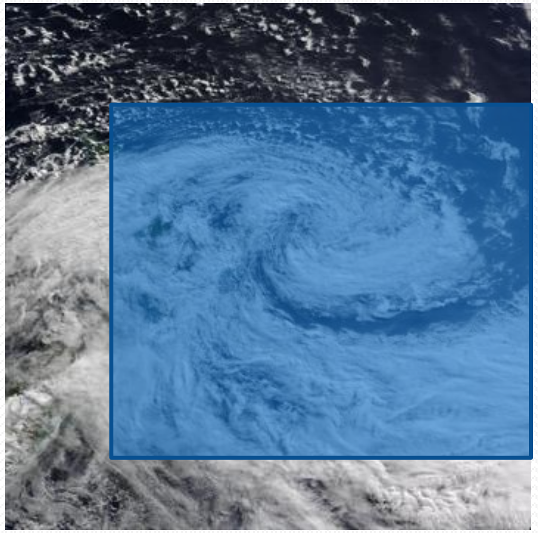
Fire Radiative Power – Random Forest



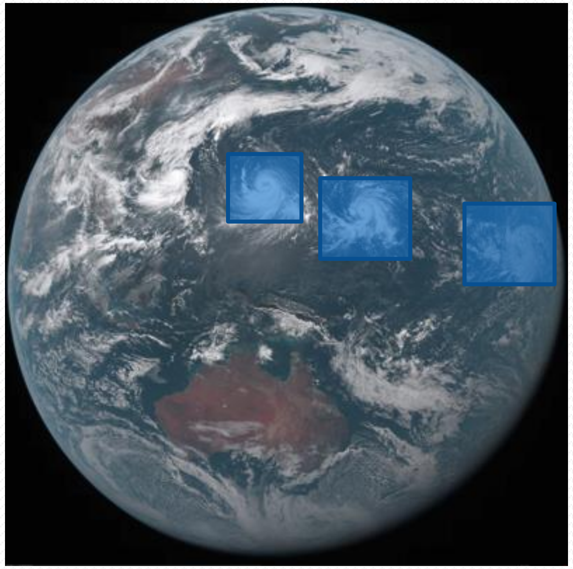


Background into Regions of Interest (ROI)

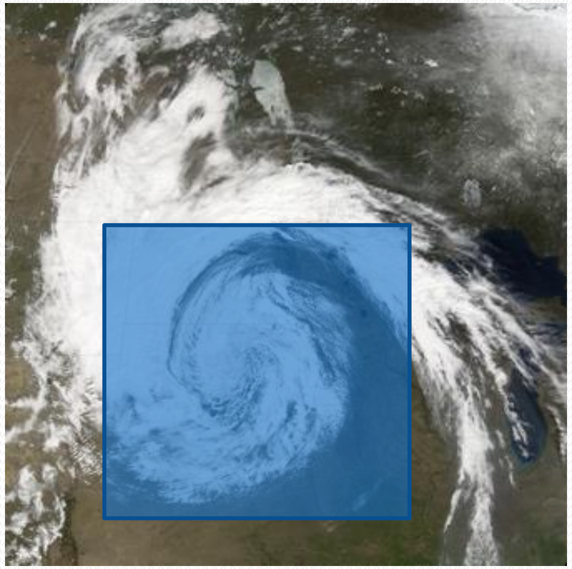
- A ROI can be anything: convection, tornado, hurricane, cirrus clouds, etc.
- Select ROI that meteorologist can identify from within satellite data
  - Cyclones: tropical and extratropical
- Note: Storms have similar characteristics but still look different



Winter Cyclone over New Zealand from MODIS Terra satellite in 2008



Himawari-8 true-color Red/Green/Blue (RGB) full-disk image



Spring snow over the Dakotas from MODIS Terra satellite in 2006

Background into Regions of Interest (ROI)

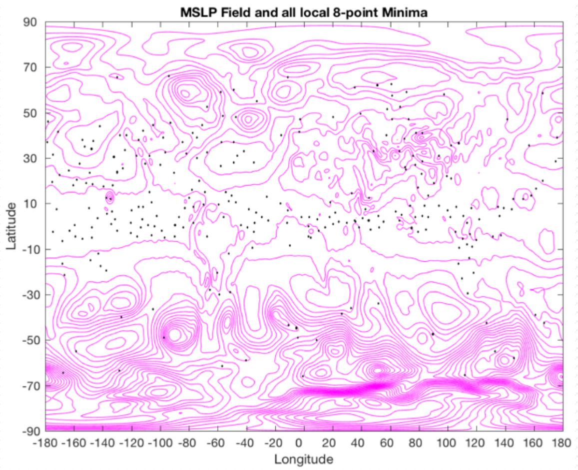
- Many post-processing studies in environmental science already have identified datasets/ROI
  - What about those that don't?
- Image Processing Struggles
  - How do we **validate** our models?
  - Is there a dataset containing images of ocean blooms that's **up-to-date**?
- Real-time ROI processing even bigger challenging
  - Especially with large image files such as satellite

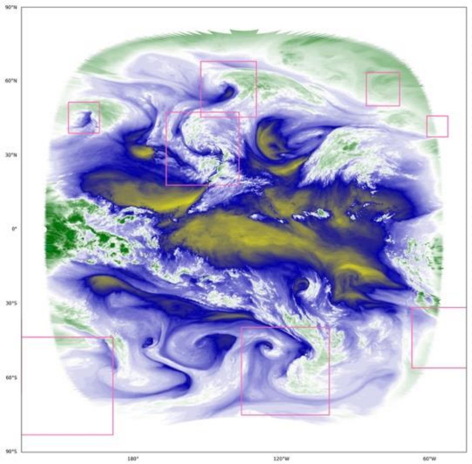
Finding Labeled Data For Cyclone ROI

- International Best Track Archive for Climate Stewardship (IBTrACS) dataset for tropical cyclones
  - Most complete worldwide database
  - Already created
- Engineered a unique automatic *heuristic* labeler for extratropical cyclones
  - Several heuristic models for extratropical cyclones already, but each have different rules and therefore identify different amounts of cyclones
    - IMILAST paper lists most trackers in existence
- Why not just use this and call it a day? → It's slow!

Bonfanti, Christina, et al. "Machine Learning: Defining Worldwide Cyclone Labels for Training." 2018 21st International Conference on Information Fusion (FUSION), 2018, doi:10.23919/icif.2018.8455276.
Neu, Urs, et al. IMILAST: A Community Effort to Intercompare Extratropical Cyclone Detection and Tracking Algorithms. Bulletin of the American Meteorological Society, vol. 94, no. 4, 2013, pp. 529547., doi:10.1175/bams-d-11-00154.1.
https://www.ncdc.noaa.gov/ibtracs/index.php

Making the Labels





GFS Model Outputs

⇩

Into Labeler Model

⇩

Labeled ROI csv

⇩

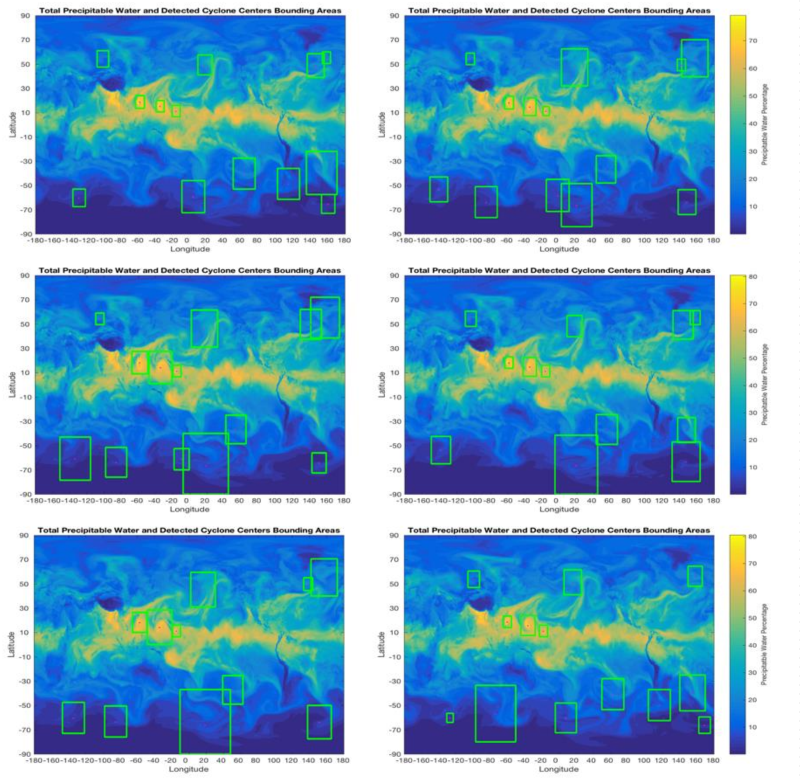
Transposed onto satellite images

⇩

Labeled Satellite Image

⇩

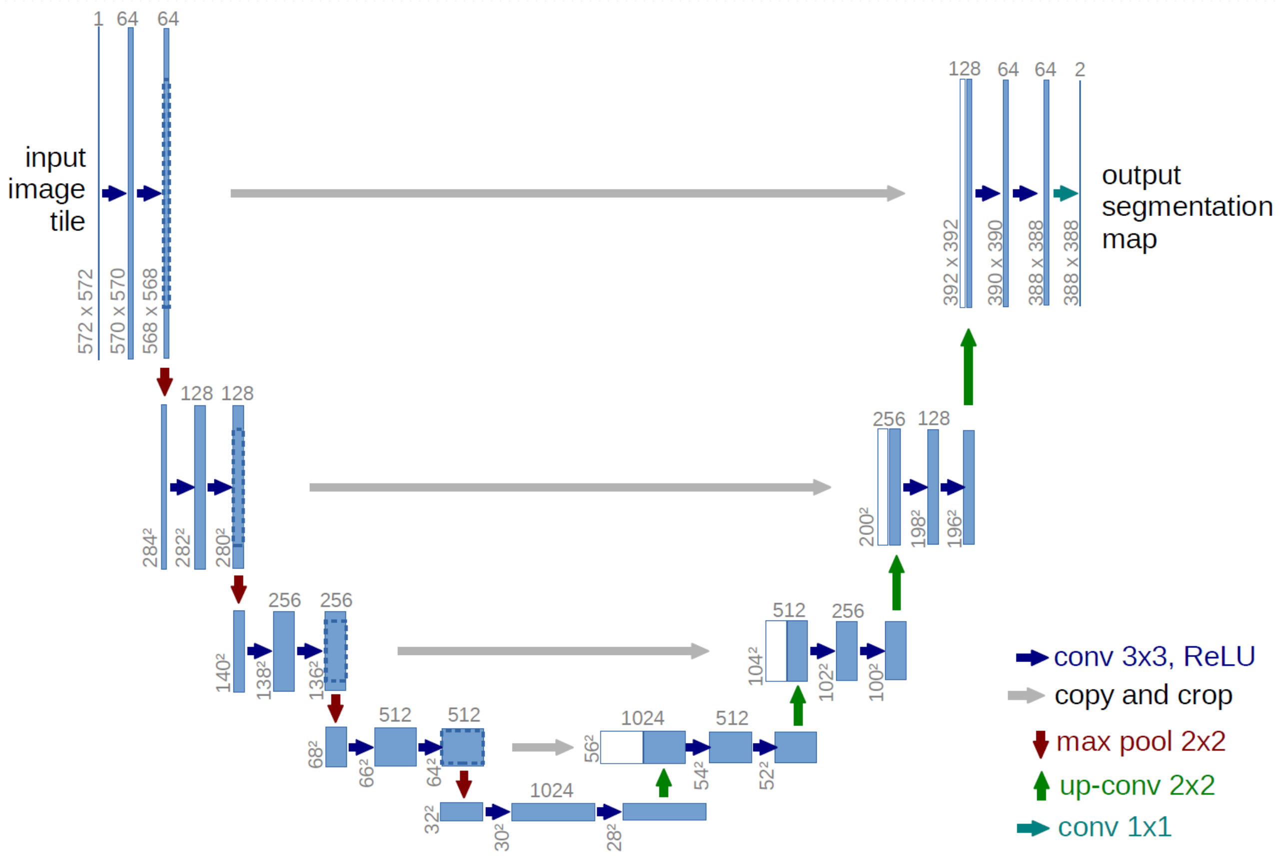
Used in deep learning model



Tropical and Extratropical Cyclone ROI

The UNET: The structure has a contracting path and then an expansion path hence the "U" in the name

- **Connects larger features before compression with smaller ones after compression**
- Image Segmentation Problem
- Heuristic-derived labels
- Keras up top
- Google's tensorflow underneath
- GOES water vapor channel
  - 2010-2016



Tropical and Extratropical Cyclone ROI

- Our UNET:
  - Loss function: DICE
  - Activation: RELU
  - After each convolution, it does a batch normalization
- 22,232 Training Samples
- 4,707 Test Samples
- 40 epochs
- Batch size of 16
- Images resized and cropped to 1024x512

```
from keras.models import Model
from keras.layers import Input, concatenate, Conv2D, MaxPooling2D, Activation, UpSampling2D, BatchNormalization

def unet(img_rows=64, img_cols=64, channels=1, output_channels=1, activation="relu"):
    print ("Creating Unet network with rows: ", img_rows, " cols:", img_cols, " channels:", channels, " output:", output_channels)

    inputs = Input((img_rows, img_cols, channels))


    down0 = Conv2D(32, (3, 3), padding='same')(inputs)
    down0 = BatchNormalization()(down0)
    down0 = Activation(activation)(down0)
    down0 = Conv2D(32, (3, 3), padding='same')(down0)
    down0 = BatchNormalization()(down0)
    down0 = Activation(activation)(down0)
    down0_pool = MaxPooling2D((2, 2), strides=(2, 2))(down0)
    # 128

    down1 = Conv2D(64, (3, 3), padding='same')(down0_pool)
    down1 = BatchNormalization()(down1)
    down1 = Activation(activation)(down1)
    down1 = Conv2D(64, (3, 3), padding='same')(down1)
    down1 = BatchNormalization()(down1)
    down1 = Activation(activation)(down1)
    down1_pool = MaxPooling2D((2, 2), strides=(2, 2))(down1)
    # 64

    down2 = Conv2D(128, (3, 3), padding='same')(down1_pool)
    down2 = BatchNormalization()(down2)
    down2 = Activation(activation)(down2)
    down2 = Conv2D(128, (3, 3), padding='same')(down2)
    down2 = BatchNormalization()(down2)
    down2 = Activation(activation)(down2)
    down2_pool = MaxPooling2D((2, 2), strides=(2, 2))(down2)
    # 32

    down3 = Conv2D(256, (3, 3), padding='same')(down2_pool)
    down3 = BatchNormalization()(down3)
```

Answering: How well did it work?

We looked at the *dice coefficient* to measure how well our model performed against our truth because this compares “likeness” as opposed to accuracy’s binary comparison:

$$\frac{\mathbf{2*|X \cap Y|}}{\mathbf{|X|+|Y|}}$$

This is basically a statistic which shows how well the model and truth match.

Answering: How well did it work?

We looked at the *Tversky coefficient* to measure how well our model performed against our truth because this compares “likeness” as opposed to accuracy’s binary comparison. We set $\alpha = 0.3$ and $\beta = 0.7$:

$$\frac{|\mathbf{X} \cap \mathbf{Y}|}{|\mathbf{X} \cap \mathbf{Y}| + \boldsymbol{\alpha}|\mathbf{X} - \mathbf{Y}| + \boldsymbol{\beta}|\mathbf{Y} - \mathbf{X}|}$$

NOTE: if both $\alpha = \beta = 0.5$ then you have the dice coefficient as they have equal weight.

Results: Labels on GFS Output

IBTrACS Labels versus DL model

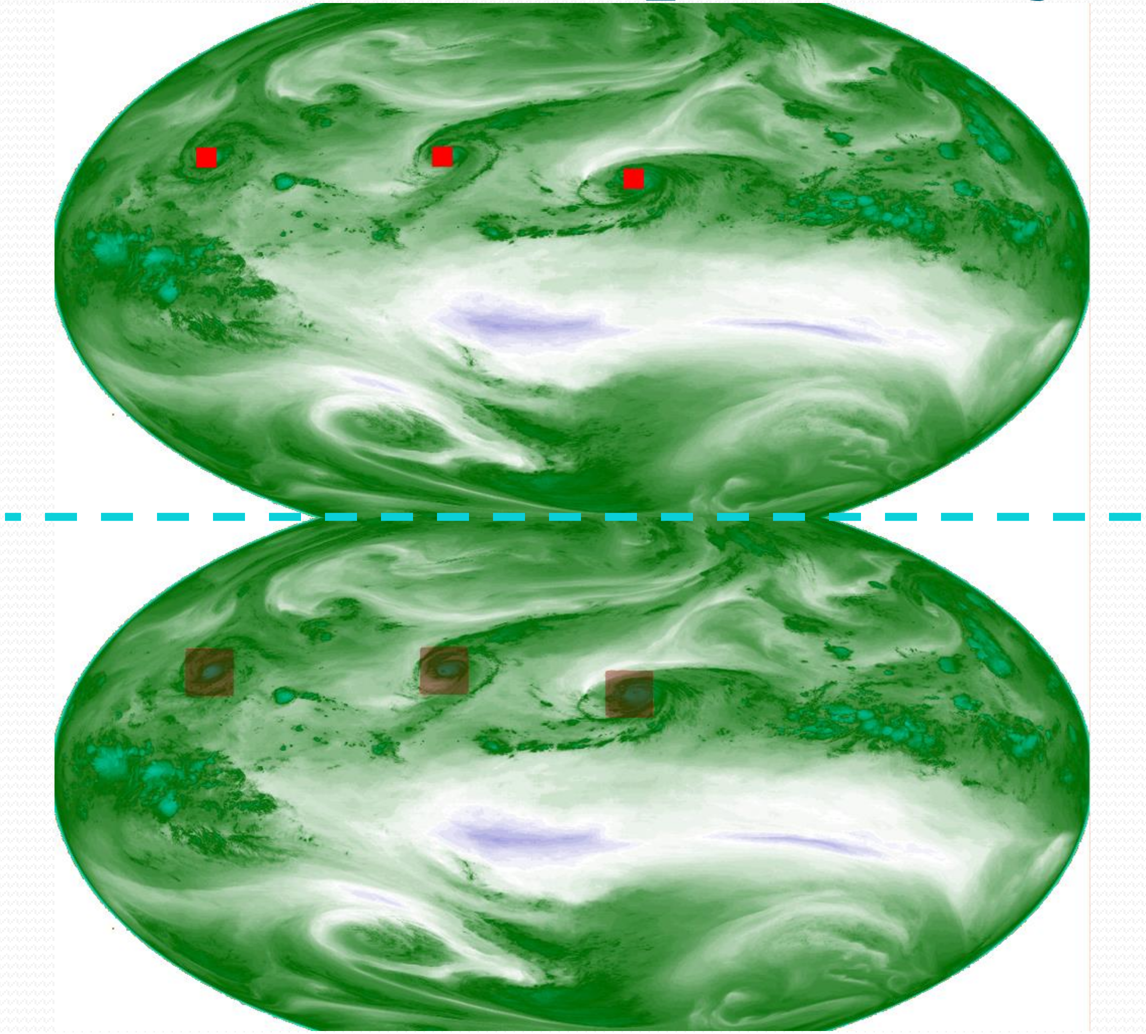
Loss	Dice Coefficient	Accuracy
0.1896	0.8243	0.9979

Heuristic Labels versus DL model

Loss	Dice Coefficient	Accuracy
-----	0.7282	0.9783

Tropical and Extratropical Cyclone ROI

IBTrACS training only

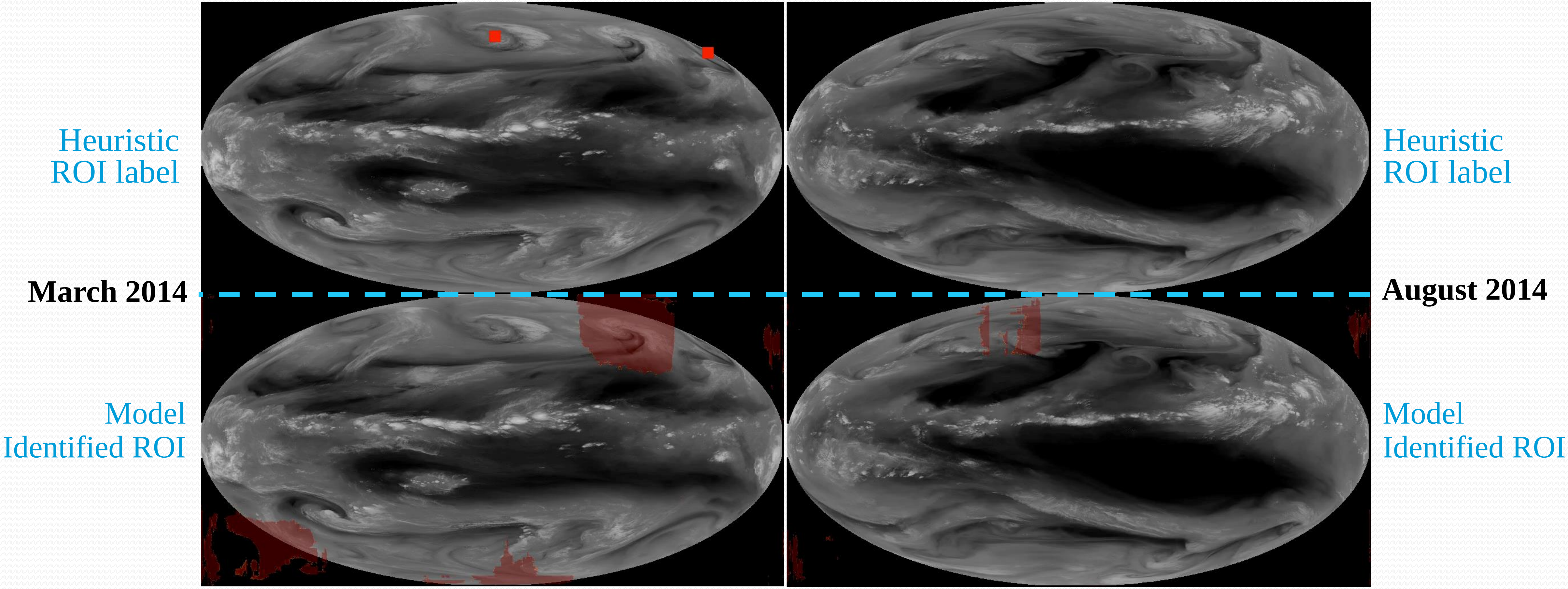


IBTRaCS label

Model Identified Hurricanes

Tropical and Extratropical Cyclone ROI

All cyclones from ROI labeler with 80x80 pixel ROI pixel box at 75% threshold using a dice loss function and relu activation



Results: Heuristic labels on Satellite Data

Heuristic Labels versus DL model

Accuracy	Dice Coefficient	Tversky Coefficient
0.8924	0.4006	0.3626

What needs to be considered in a DL model?

- When and how data is processed matters
  - How is it normalized/scaled? With or without NaN’s?
    - More success with a batch normalization
- Memory is an issue
- Picking the right loss and activation function
- Up-and-coming methods:
  - Curvature of the edges- SphereNet

Thoughts/Conclusions

- Tropical versus Extratropical cyclones
  - Tropical Cyclones are more uniform and deviate less in appearance than extratropical
  - Occur along the equator where projection is less an issue
  - Edges are an issue for both but more so when projection is also a factor
- Impressed by the visual results of extratropical cyclone detection
  - Numerical measurements of performance don’t speak to the results of ROI detection
  - **How do we compare success to another heuristic-based method?**

PART 2

Machine Learning for Climate and Weather – Challenges and Strategies

Imme Ebert-Uphoff
CIRA / Electrical & Computer Engineering
Colorado State University

NOAA-STAR seminar
July 18, 2019

CIRA
Connecting Models and Observations

Challenges for Using ML in Weather/Climate

Earth science applications differ from typical machine learning (ML) applications.
→ Not straight forward to apply standard ML algorithms.

Two Key Challenges (these will be discussed in more detail):

1. **Transparency**: Scientists usually want to **understand how** ML methods get their results. Even if not, transparency is important for debugging, etc.
2. **Generalization**: ML models must be able to handle new regimes, i.e. scenarios they have not seen during model construction/training.

Other challenges (not further discussed here):

- Often large data set, but small sample size. In particular, lack of labeled data, i.e. often not many samples available along with "correct answer".
- Spatio-temporal structure;
- Objects may have fuzzy boundaries (e.g., clouds, atmospheric rivers);
- Heterogeneity in space and time;
- Multi-source, multi-resolution data.

Sample Pitfall

Application: ML applied to Skin cancer detection

Task: Given image of skin lesion, classify whether benign or malignant

On first try: Method had *amazing* success rate - whenever the doctors thought it was benign/malignant, the ML method came to the same conclusion!

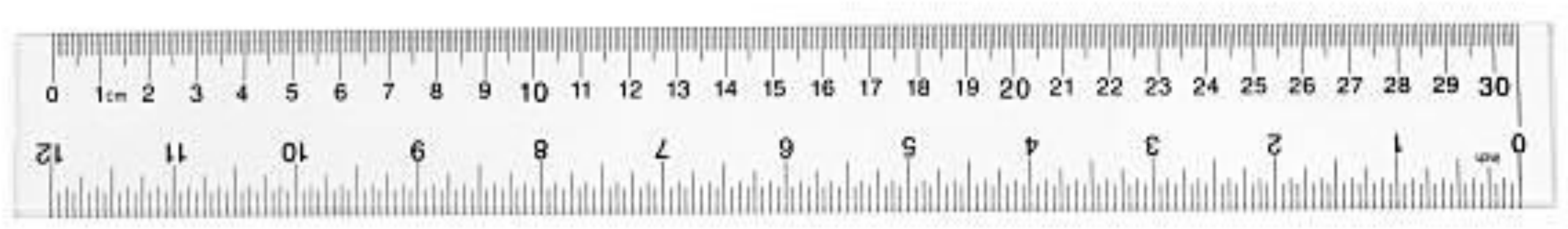
Almost *too good* to be true.
→ Scientists wanted to know: How did the algorithm figure it out?

→ Applied visualization tool to learn about method's reasoning.

Scientists found that …

Skin cancer example

Scientists found that …
… doctors had placed a ruler into the image whenever they thought it was malignant.



The algorithm detected the ruler, then concluded that the growth was malignant. **That's not what folks had intended for the algorithm to do!**
Found problem early thanks to transparency tools.

That’s why transparency is important

Conclusion: The algorithm detected the ruler, didn’t look at the skin lesion at all.

Why did the ML algorithm behave this way?

- An ML algorithm does not “understand” its task.
- It is just **looking for any information/patterns that help it solve its task**.
- ML algorithm did its job perfectly – the ruler was the perfect indicator.

Lack of generalization:

- The algorithm worked perfectly for the training samples, but would not have worked when deployed (no rulers).
- **Including the rulers was an obvious mistake, but there can always be (less obvious) clues that lead to non-generalizable detection algorithms.**

How to avoid such pitfalls: Understand reasoning of ML algorithm as much as possible

- Obviously: Get to know your data well before applying any ML method.
- Visualization/attribution tools can be very helpful.
- Scientists found the problem quickly once they applied visualization tools.

Selected Strategies to overcome these challenges

1. **Physics-guided Machine Learning (PGML)**

2. **Use of Interpretation/visualization tools of ML methods (especially ANNs).**

3. **Insights from Team Science / Interdisciplinary Studies to built innovative + synergistic collaborations between atmospheric and ML researchers.**
   (no time to discuss today)

[1] Newell and Luckie, *Pedagogy for Interdisciplinary Habits of the Mind*, Conference on Interdisciplinary Teaching and Learning, 2012.
[2] Flinterman et al., *Transdisciplinarity: The New Challenge for Biomedical Research*, Bulletin of Science, Technology & Society, Vol. 21, No. 4, 2001.
[3] Ebert-Uphoff, I., and Y. Deng (2017), Three steps to successful collaboration with data scientists, Eos, 98, https://doi.org/10.1029/2017EO079977.

Strategy 1: Physics-guided Machine Learning (PGML)

We have a huge advantage in earth sciences:

hundreds of years of knowledge about underlying processes available!

- PGML Approach: Integrate as much science knowledge as possible into ML algorithms.

Advantage:

- PGML can greatly **improve transparency and generalization capabilities**.

Disadvantage:

- **Requires complex, customized solution** for each project.
- Takes *a lot* of time!

How? Next slide.

Strategy 1: Physics-guided Machine Learning (PGML)

Sample techniques:

a) **Utilize physics whenever you can**
   - Break task into sub-tasks and solve as many sub-tasks as possible with physics, etc. If possible, only go the last mile with ML.
   - Examples: Preprocessing, **feature engineering.** Can I use physics to create information-rich ”features”?

b) **Algorithm & Architecture selection**
   - Example: If using neural networks, can I customize the architecture?

c) **Adding physical constraints**
   - Can we add physical constraints in optimization functions?
   - In parameter space of optimization?

PGML – Adding physical constraints – Basic Idea

Idea: Many ML algorithms are based on minimizing a cost function (aka loss function).

→ **Can use that cost function to add physical constraints.**

Typical cost function (no physics):

loss = (prediction error on test data) + (regularization term)

Adding physics:

loss = (prediction error on test data) + (regularization term) **+ (physics penalty)**

Physics penalty =

term that measures how much the results for test data **violate a given physical constraint**.

→ Physics penalty guides algorithm toward solutions that obey physical constraint.

→ Physics term can greatly improve generalization and accuracy.

PGML – Adding physical constraints - Example

Sample Application: Cloud parametrization

Source: Beucler, Tom, Stephan Rasp, Michael Pritchard, and Pierre Gentine. "Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling." ICML conference, arXiv:1906.06622 (2019).

- **Approach:** Using Artificial Neural Networks (ANNs) to emulate cloud processes.
- **Problem:** ANNs do not intrinsically conserve energy and mass, etc.
  → emulation results do not conserve those either.
- **Solution 1:** Constraints can be defined in terms of a linear function of inputs, x, and outputs, y:

$$[C]\,[\,x^T\,y^T]^T = 0$$

Enforce constraints in loss function

Cost function with physics:

loss = (prediction error on test data) + (regularization term) + $||\,[C]\,[\,x^T\,y^T]^T\,||$

PGML – Customize architecture – Ex. 1

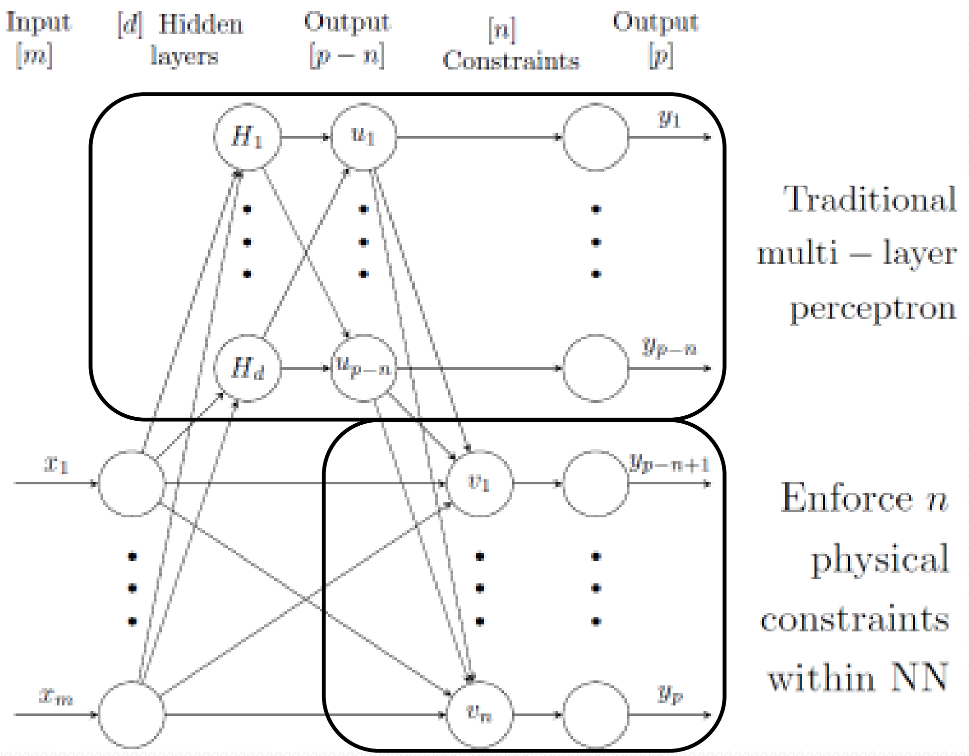
Source: Beucler, Tom, Stephan Rasp, Michael Pritchard, and Pierre Gentine. "Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling." ICML conference, arXiv:1906.06622 (2019).

Sample Application: Cloud parametrization

- **Solution 2:** Same constraints can be enforced through architecture of ANN.

$$[C]\,[\,x^T\,y^T]^T = 0$$

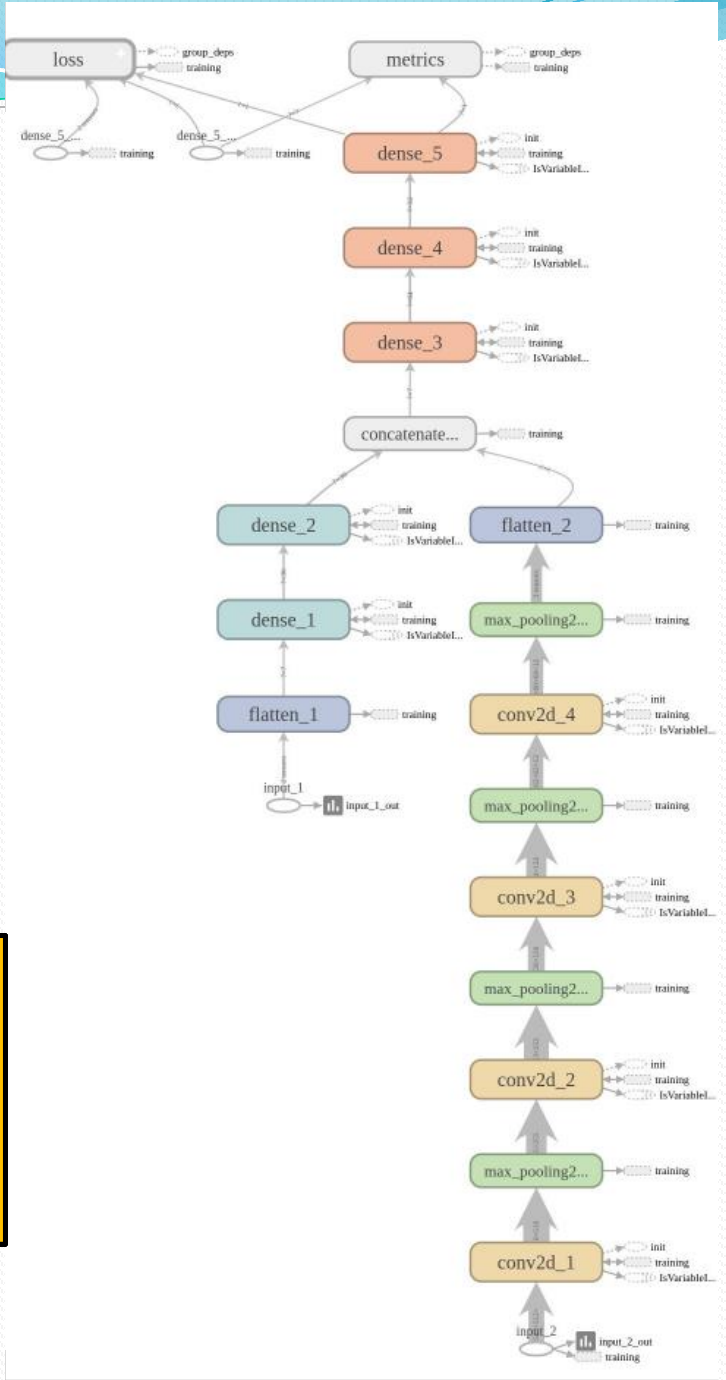
Enforce constraints in architecture



PGML – Customize architecture – Ex. 2

Second example: Complex network

- **Principle**: Decouple interaction of features in architecture based on which context is needed to extract information from them.
- Different types of variables get "different treatment".
- Special features extracted *separately* from each type.
- Resulting features merged later. Simpler model.



Source: Jebb Stewart, Christina Kumler, Isidora Jankov, Lidia Trailovic, Stevan Maksimovic, Mark Govett, *Improving Processing and Extracting Value from Satellite Observations through Deep Learning,* NOAA Workshop on AI, College Park, MD, 4-24-2019.

Strategy 2: Utilize Newest Interpretation tools for ML

Key idea:

- Many new tools exist for interpretation/visualization of ML approaches.
- New field: explainable AI.
- Use these tools as much as possible to increase transparency of ML methods.
- Shine a light inside the black box.

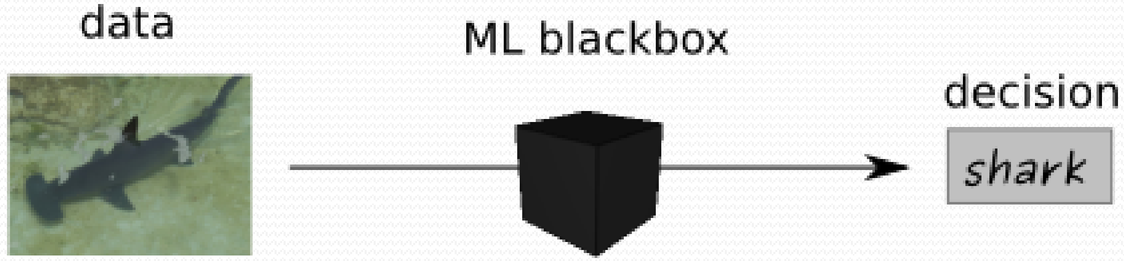
Source:
www.heatmapping.org

Example - Visualization of ANNs

ML tools are often described as a "black box" – with no way to look inside.
But more and more **tools are becoming available to get a glimpse inside**.
Sample method: Visualization to explain classification of a *specific image*

Question 1: **Is this a shark?**

Answer: **Yes!**



Question 2:
But WHY does the method think this is a shark?

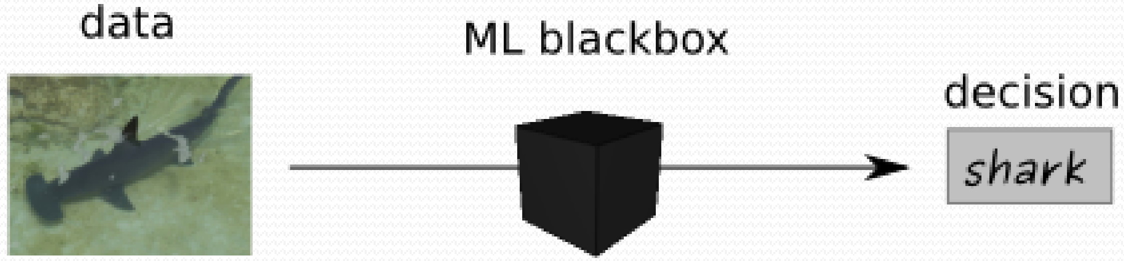
Source:
www.heatmapping.org

Example - Visualization of ANNs

ML tools are often described as a "black box" – with no way to look inside.
But more and more **tools are becoming available to get a glimpse inside**.
Sample method: Visualization to explain classification of a *specific image*

Question 1: **Is this a shark?**

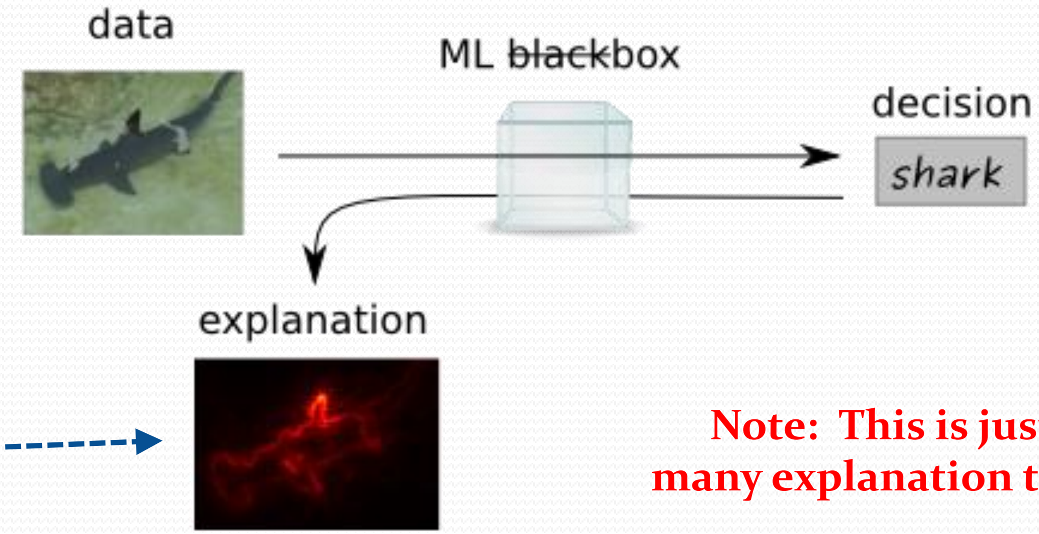
Answer: **Yes!**



Question 2:
Why does the method think this is a shark?

More specific:
Which pixels of the input image are most important to decide that this is a shark?

Answer: *Heatmap*.

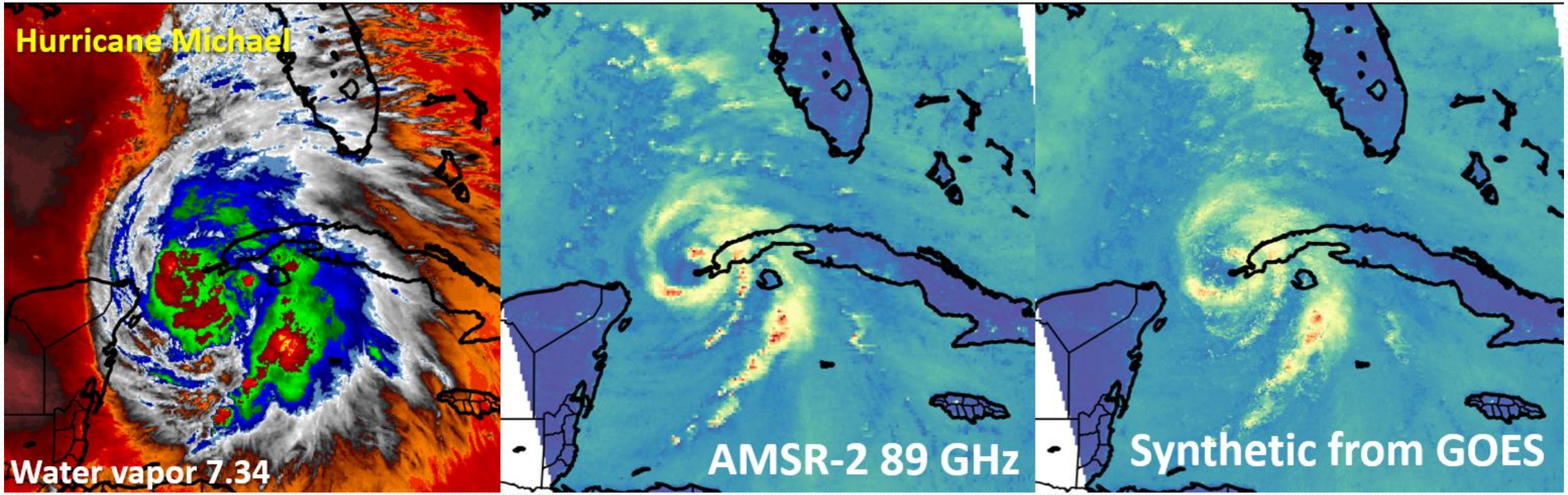


Note: This is just one of many explanation techniques.

Other On-Going ML Projects

Chris Slocum and John Knaff:

- Creating synthetic microwave imagery using GOES-R baseline products for improved hurricane monitoring and rainfall estimation
- Current Method: ANNs



Other On-Going ML Projects

Kyle Hilburn and Steve Miller:

- Machine Learning Techniques Applied to GOES-R ABI for Cloud Morphology- and Evolutionary-Based Airmass Characterization Toward Cloud-Permitting Model Analyses
- Current Method: ANNs

Yoo-Jeong Noh, Steve Miller, John Haynes, John Forsythe, Curtis Seaman:

- Improving the VIIRS Nighttime Cloud Base Height and Cloud Cover Layers Products for High Latitude Weather and Aviation Forecast Applications
- Current Method: Random Forest

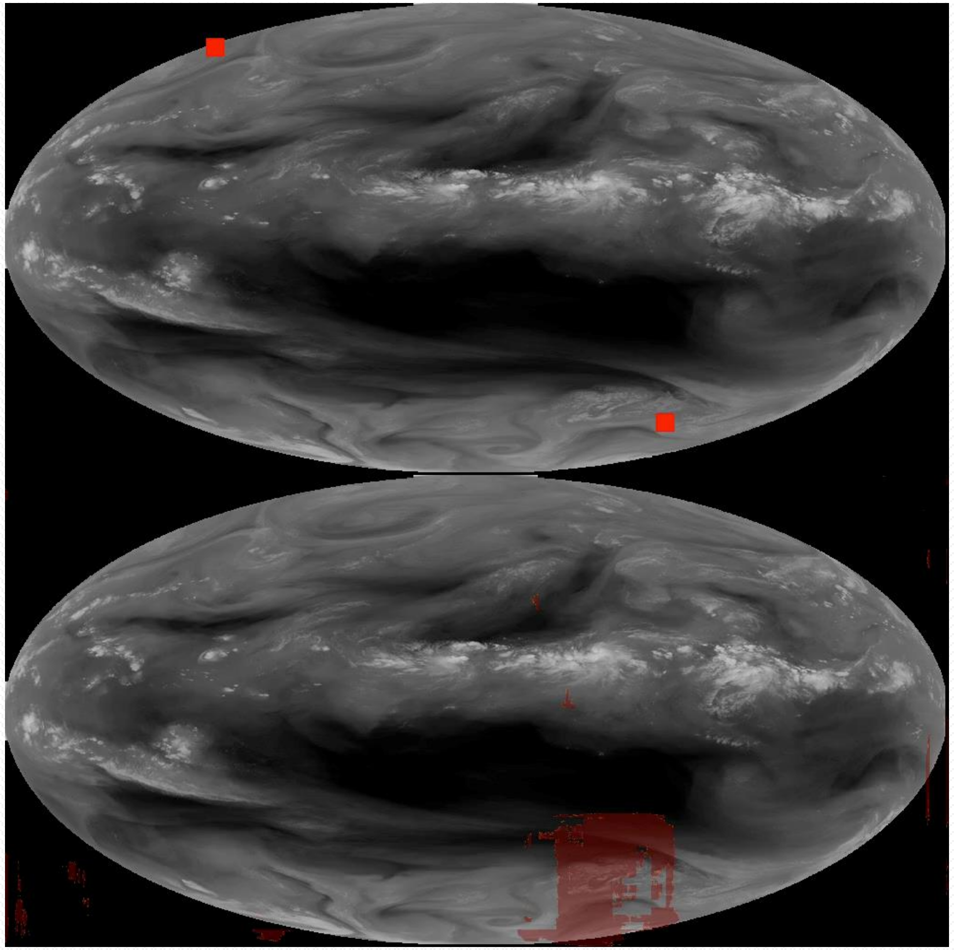
Concluding Comments

Selected strategies discussed:

1. **Physics-guided Machine Learning (PGML) → Atmospheric scientist is very important. → See Item #3 below.**
2. **Use of interpretation/visualization tools of ML methods (especially ANNs).**
3. **Building innovative + synergistic collaborations between atmospheric and ML researchers (team science, interdisciplinary studies).**

These strategies **do *not* provide a magic wand**, but they can be **quite effective** to

- Increase transparency,
- Improve generalization capabilities,
- Deal with small sample size.



[Questions or Suggestions?]

Imme Ebert-Uphoff (iebert@colostate.edu)
Christina Kumler (christina.e.kumler@noaa.gov)