Tornado Prediction Using Environmental Sounding Data: Comparing Random Forest to CNN

Brice Coffer[1], Michaela Kubacki[2], Yixin Wen[3], Ting Zhang[4]

[1]North Carolina State University, [2]Middlebury College, [3]University of Oklahoma, [4]McDaniel College

Research Assistant: Carlos A. Barajas[5]

Faculty Mentor: Matthias K. Gobbert[5]

[5] University of Maryland, Baltimore County

UMBC

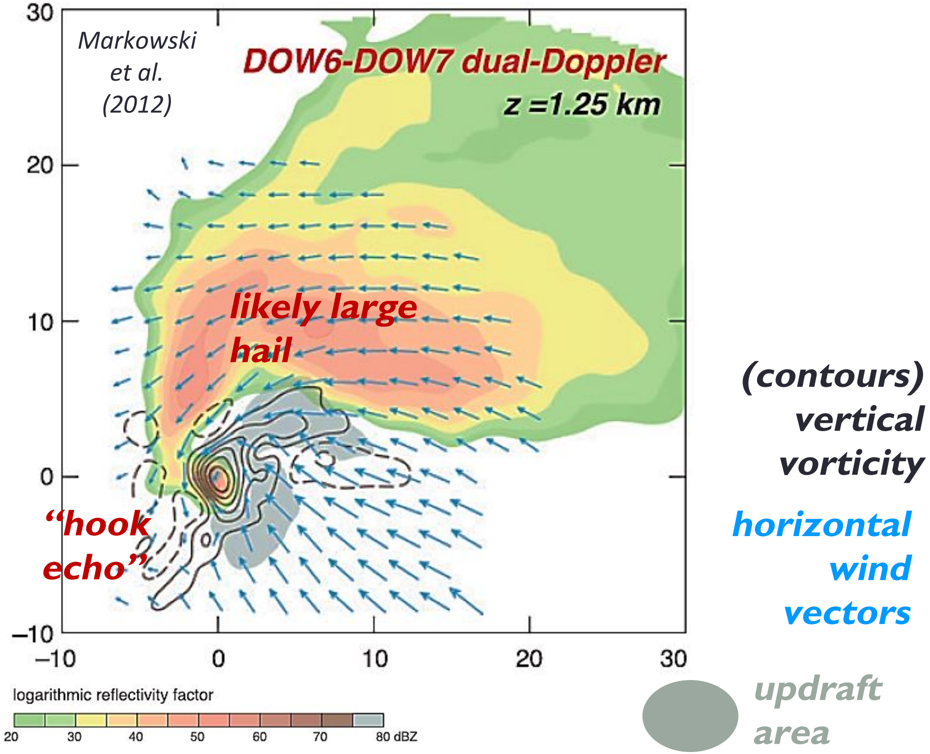
Supported through NSF OAC-1730250

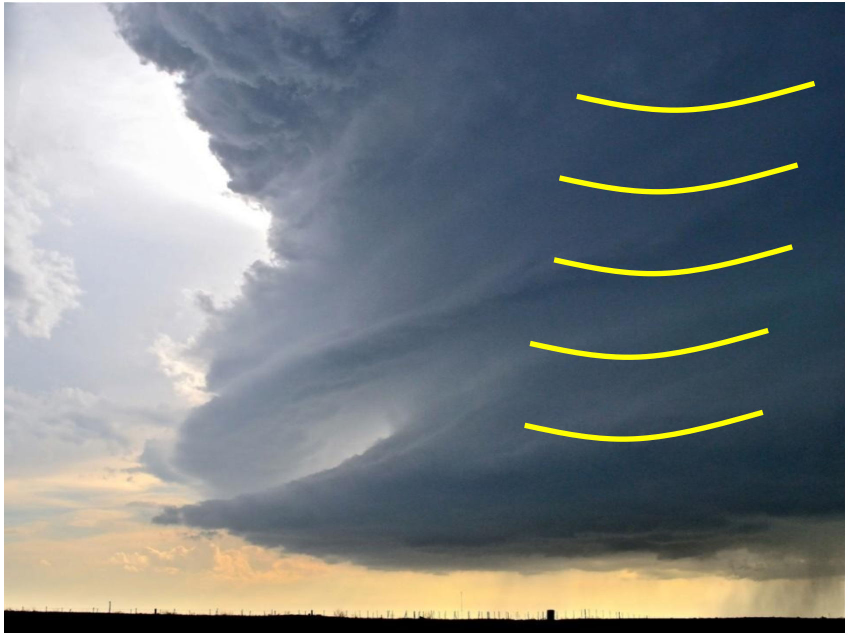
Outline

- Part I: Background & Motivation
  - Research problem & motivation
  - RUCsounding Data
- Part II: Data Preprocessing
- Part III: Random Forest
  - Results
  - Feature Importance Analysis
- Part IV: Convolutional Neural Network
  - Model Description
  - Results
- Part V: Over and Undersampling Applied to RF and CNN
  - Imbalanced Data
  - Results

What is a supercell (dynamically)?

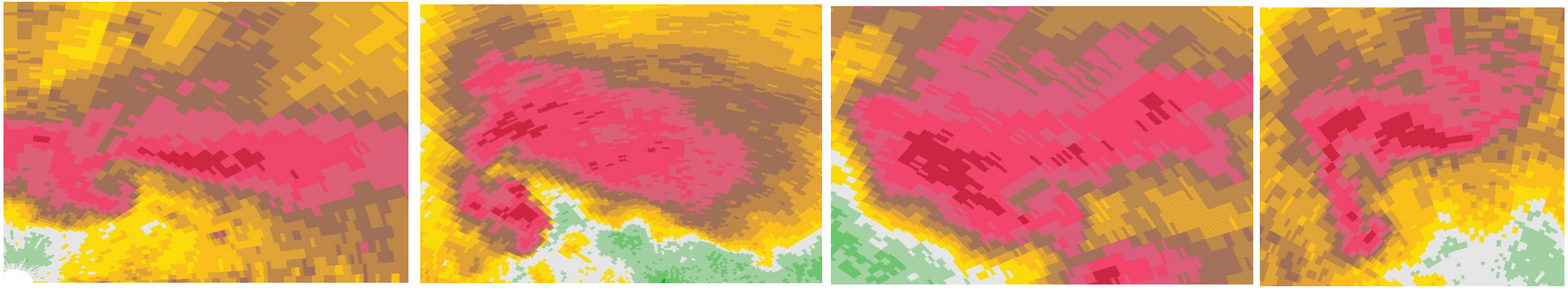
A thunderstorm with a rotating updraft (typically, updraft vertical velocities > 10 m s^{-1} correlated with vertical vorticity $\zeta > 0.01\ s^{-1}$)





What motivates supercell research…

- **Although most significant tornadoes are associated with supercell thunderstorms, most supercells are *not* tornadic**
  - **Perhaps the biggest motivation for supercell research is discriminating between tornadic and nontornadic supercells**



nontornadic ***nontornadic*** ***tornadic*** ***nontornadic***

Markowski and Richardson (2010)

Current forecasting methods: Significant tornado parameter

Storm Prediction Center
Norman, Oklahoma

$$STP = \frac{MLCAPE}{1500\ J\ kg^{-1}} \times \frac{2000 - MLLCL}{1000\ m} \times \frac{200 + MLCIN}{150\ J\ kg^{-1}} \times \frac{EBWD}{20\ m\ s^{-1}} \times \frac{\mathbf{SRH500}}{\mathbf{75\ m^{2}\ m^{-2}}}$$

CAPE	convective available potential energy
LCL	lifted condensation level
CIN	convective inhibition
EBWD	effective bulk wind difference
SRH500	**0 – 500 m storm relative helicity**

ML = mixed layer (i.e., lowest 100 mb of the atmosphere)

Thompson et al. 2003, 2007, 2012

Coffer et al. 2019

Current Status of Machine Learning in Weather Forecasting

- The use of machine learning in severe weather forecasting is recently starting to take off (McGovern et al. 2019).
- Emerging efforts attempting to recreate the National Weather Service's tornado warning process primarily uses **radar data.** (Lagerquist et al. 2020)
  - ***This requires the storm to have already formed.*** (lead time ~10-15 min)

Our Research Approach:

Using **RUC sounding data**, we construct several **Random Forest** (**RF**) and a **Convolutional Neural Network** (**CNN**) Models to predict, ***in advance***, whether a supercell storm will produce a significant tornado (F2-F5), a weak tornado (F0-F1), or no tornado.

Novel aspects of this research:

- By focusing on pre-storm environment from **soundings**, predictions could be useful for forecasters many days ***in advance.***
- **Long term goal** is to use advantages of machine learning to identify novel structures in data to guide future research questions in severe storm modeling and forecasting.

Hardware and Software

Hardware: UMBC High Performance Computing Facility (www.hpcf.umbc.edu)

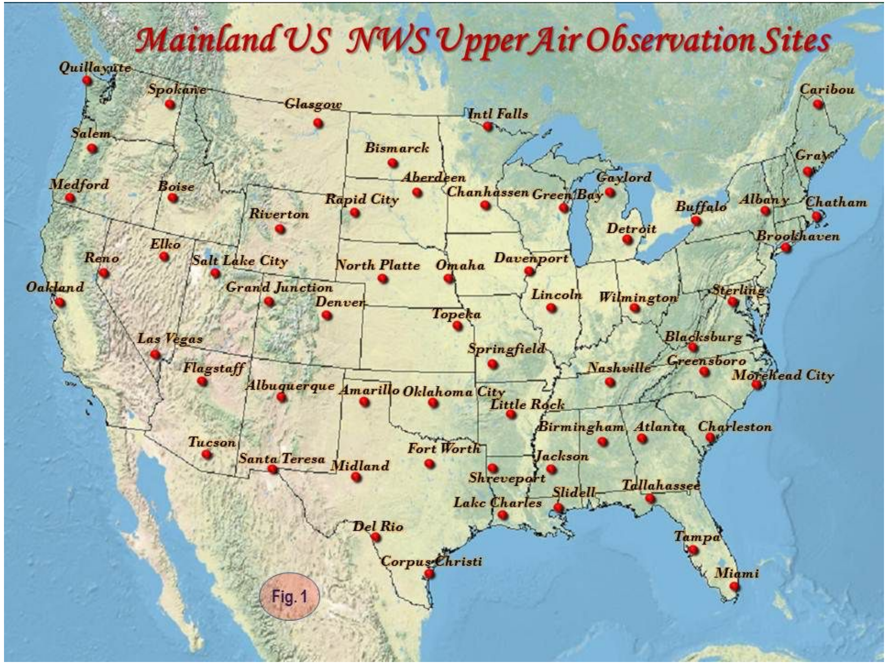
- Studies used a CPU node with two 18-core Intel Xeon Gold 6140 Skylake CPUs (2.3 GHz clock speed, 24.75 MP L3 cache, 6 memory channels)

Software: Python 3.7.6 along with the following packages

- scikit-learn (v. 0.23.dev0), imbalanced-learn (v. 0.6.2), TensorFlow (v. 2.1.0), Keras (v. 1.1.0)

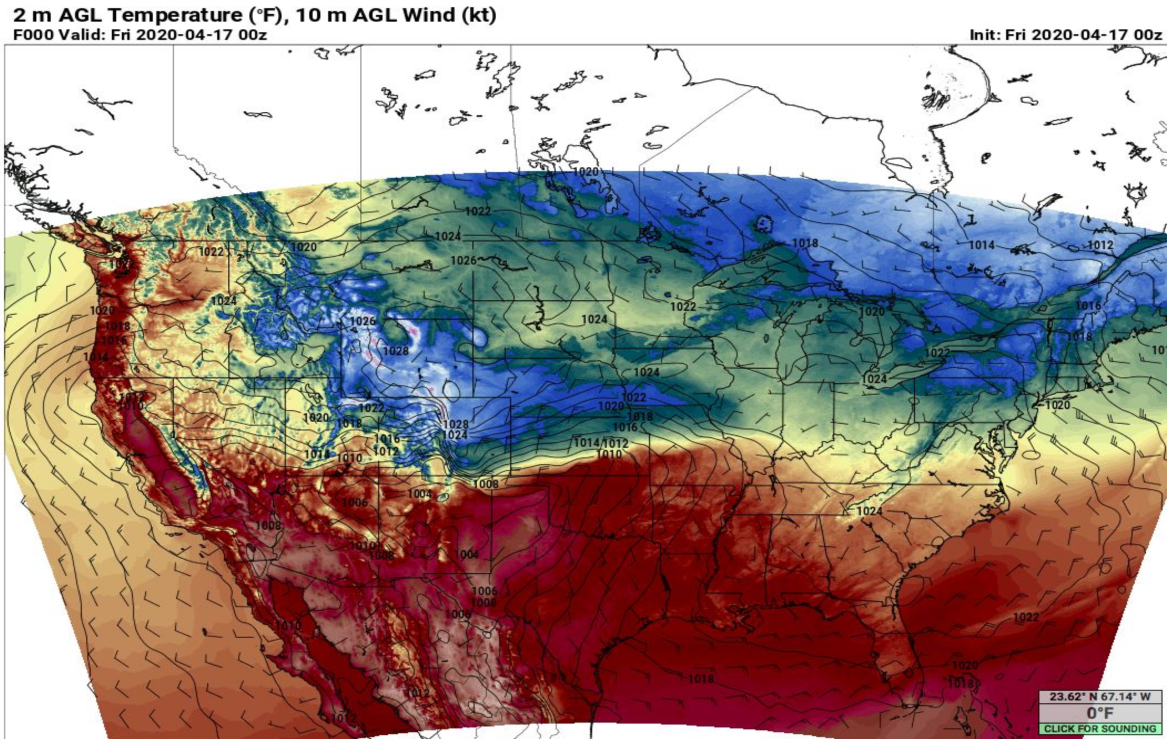
WHAT DATA DO WE HAVE?

Observed soundings



- Balloons only launched twice a day from 70 locations across the United States
- Would take **centuries** to sample enough storms to generate required dataset

Model based soundings



- **Objective analysis schemes** compile observations from many different sources to create a high-resolution, "best guess" of the current state of the atmosphere, in order to initialize forecast models

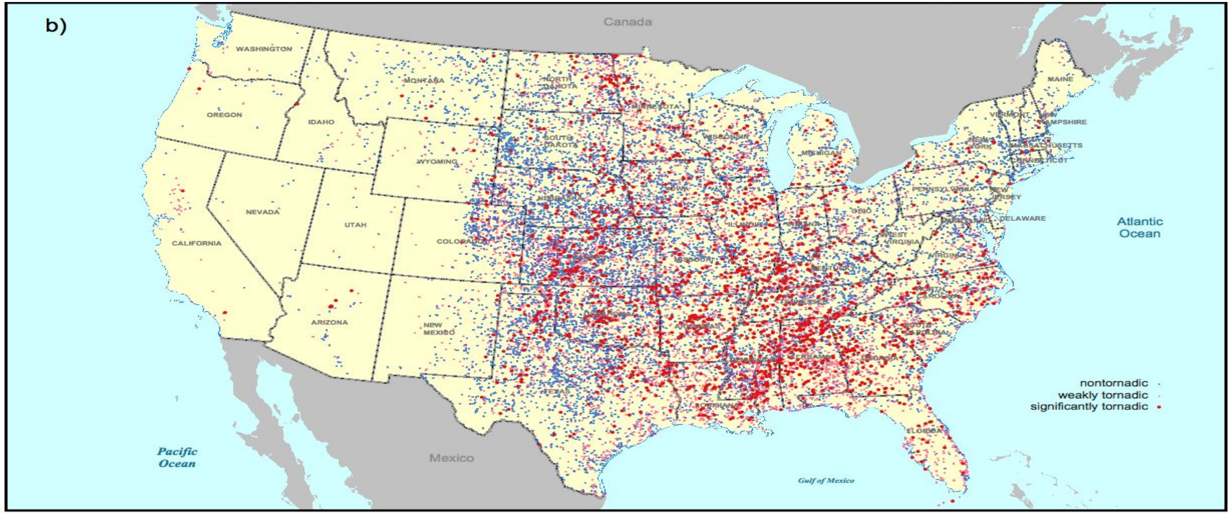
- We can then **extract vertical soundings** from these analyses for any severe weather report in the United States, instead of relying solely on the twice a day sounding network

20194 supercell vertical profiles from 2005-2017

These were manually identified by looking over 100,000+ severe weather reports and filtered to only include supercell storms

<u>Sample Size</u>
Tornadic: 9,355 cases
- 7,743 “weak tornadoes”
- 1,612 “significant tornadoes”

Nontornadic: 10,839 cases
- 7,051 severe hail reports
- 3,788 severe wind reports



Each vertical profile has 7 variables:
- pressure (hPa)
- temperature (C)
- dewpoint temperature (C)
- height above ground level (m)
- relative humidity (%)
- u-component of wind (m/s)
- v-component of wind (m/s)

Data from Smith et al. (2012) and Coffer et al. (2019)

Outline

- Part I: Background & Motivation
  - What is a supercell?
  - RUCsounding Data
  - Research problem & motivation
- Part II: Data Preprocessing
- Part III: Random Forest
  - Results
  - Feature Importance Analysis
- Part IV: Convolutional Neural Network
  - Model Description
  - Results
- Part V: Over and Undersampling Applied to RF and CNN
  - Imbalanced Data
  - Results

Data Structure

- **Variables:** Temperature, Dewpoint, Humidity, U-wind, V-wind, Pressure, and Height
- **Height** = height above ground-level at 25hPa increments up to 100 hPa, converted to meters
- Severe storm reports at higher elevations correspond with lower surface pressure, so there are a variable number of vertical (height) levels at variable heights in the 20,194 samples

	lat	lon	event type	…	height	temp	etc
report 1	35.21	-104.3	'nontornadic'		[10, 253, …]	[24.56, 23.12,…]	
report 2	38.63	-107.68	'weakly tornadic'		[10, 115, …]	[22.31, 20.35,…]	
….							
report 20194	33.46	-99.98	'significantly tornadic'		[10, 198, …]	[19.46, 18.91, …]	

Height Data Variability:

- All storms have at least 26 vertical levels
- 8,479 storms have 37 vertical levels
- Heights at each level vary by storm

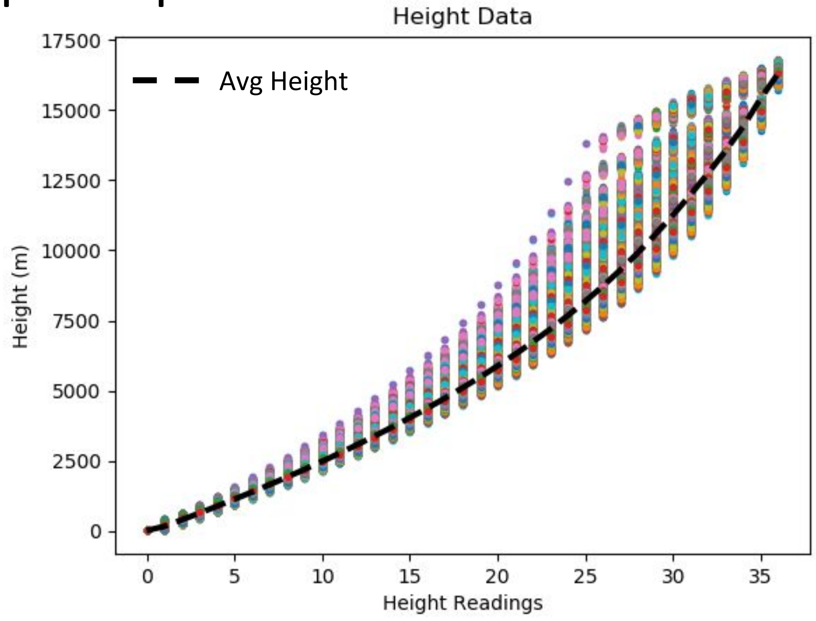
temperature, dewpoint, humidity, U-wind, V-wind, and pressure measurements correspond to recorded heights

Standardizing Height Levels

<u>Underlying motivation:</u> Can RF and CNN models find novel structures in the data that help discriminate between nontornadic and tornadic storms without preconceived computed parameters?

<u>Need uniformly structured data with respect to height:</u>

- Current height levels have wide ranges: Vertical Level 20 has height range 4831m-8085m
- Compute average heights at each vertical level
- Interpolate variable measurements (temperature, dewpoint, etc.) over average heights



Outline

- Part I: Background & Motivation
  - What is a supercell?
  - RUCsounding Data
  - Research problem & motivation
- Part II: Data Preprocessing
- Part III: Random Forest
  - Results
  - Feature Importance Analysis
- Part IV: Convolutional Neural Network
  - Model Description
  - Results
- Part V: Over and Undersampling Applied to RF and CNN
  - Imbalanced Data
  - Results

RANDOM FOREST

Random Forest Classification (RF)

Motivation for using this method:

- Can do multiclass classification
- Can handle large number of input features
- **Feature Importance:** Can estimate which features contribute most in the classification, potentially revealing important structures in the data.

How it works:

- It is an ensemble method that uses randomized decision trees as its base models.
- The algorithm trains a (user specified) set of decision trees separately, and aggregates votes from the trees to give a final prediction for each training object.
- Reduces variance in the predictions, and thus boosts testing performance.
- "Randomness" is injected into the algorithm in two ways:
  1. On each iteration, it takes a new random subsample of the given dataset so that it uses a different training set.
  2. It uses a random subset of features to determine how to split on each tree node.

Data Format for Random Forest (RF)

height	10m	154m	...	16317m	10m	...	16317m	
	Temp 1	**Temp 2**	**...**	**Temp 37**	**Dewpoint 1**	**...**	**Pressure 37**	**Event**
1	16.71	17.00	...	-67.24	15.83	...	100.17	weak
2	20.46	19.99	...	-69.09	18.12	...	100.26	weak
3	20.53	19.69	...	-70.53	18.08	...	100.44	nontornadic
...	...	...	...	...	...	...	...	...
20194	21.12	20.26	...	-69.09	19.98	...	100.11	weak

Total Number of Features = 6*37 =222

(37 each of temperature, dewpoint, humidity, U-wind, V-wind, and pressure)

Random Forest (RF) Model Current Results

Input: RUCSoundings
(Train 80%, Test 20%)

Create RF Model and Evaluate:
(Python sklearn)
Parameter Settings:
n_estimators = 200
max_depth = 200
class_weights ='balanced'

Output Classifications:
[0] - Nontornadic
[1] - Weakly Tornadic (F0-F1)
[2] - Signif. Tornadic (F2-F5)

Output Data:
Confusion Matrix
Feature Importance Scores

Confusion Table

Predicted / Actual	[0]	[1]	[2]
[0]	1837	307	28
[1]	550	986	32
[2]	62	168	69

Accuracy Scores (by class)

Class	Total	Predicted	Accuracy
[0]	2172	1837	84.58%
[1]	1568	986	62.88%
[2]	299	69	23.08%

Overall Accuracy 71.6%

Random Forest Results: Skill Scores

Contigency Table		
Actual **Predicted**	**Yes [2]**	**No [0]**
Yes [2]	69 (Hits)	28 (False alarms)
No [0]	62 (Misses)	1837 (Correct Rejects)

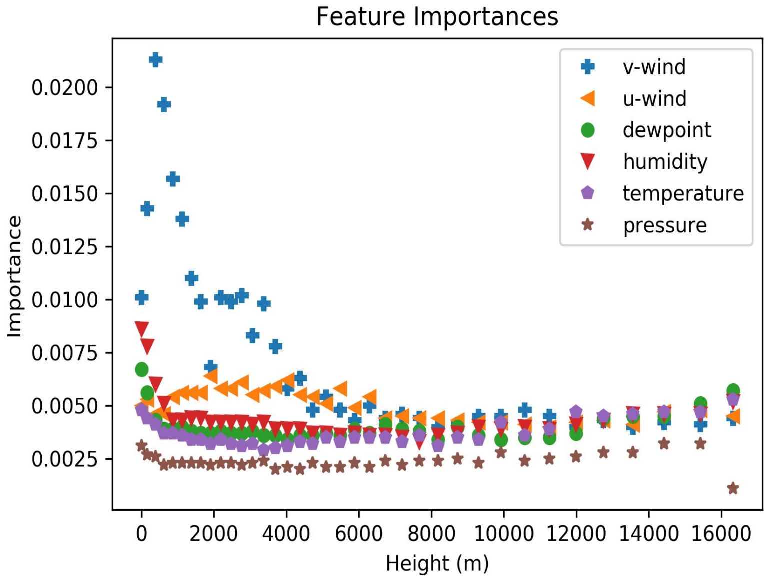
Sig.-Tornadic/Nontornadic ([2] vs [0])

Score	RF Prediction	Forecasting
POD	0.53	0.69
FAR	0.29	0.26
CSI	0.44	0.55
TSS	0.51	0.44

POD = Probability of Detection= Hits/(Observed)
FAR = False Alarm Ratio= (False Alarms)/(Predicted)
CSI = Critical Success Index = Hits/(All)
TSS = True Skill Statistic = (Hits*Rejects-False*Misses)/(Observed*None)

Observed= hits + misses
Predicted= hits + false alarms
All = hits + false alarms + misses
None = false alarms + correct rejects

Random Forest Feature Importance Analysis



Accuracy Scores with and without pressure		
Class	**Accuracy (w/pressure)**	**Accuracy (w/o pressure)**
[0]	84.58%	83.28%
[1]	62.88%	60.94%
[2]	23.08%	23.84%
Total	71.6%	70.14%

RF Feature Importance Analysis Notable Observations:

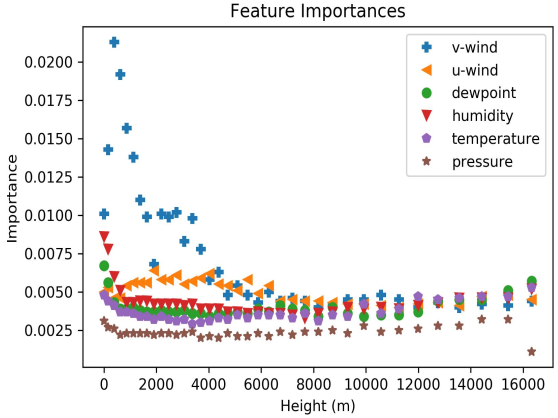
- ***Pressure*** is consistently the lowest ranked feature
- ***V-wind*** below 4 km is of highest importance overall
- ***U-wind*** is most importance between 1-6 km
- ***Humidity and dewpoint*** have highest importance scores at their lowest levels

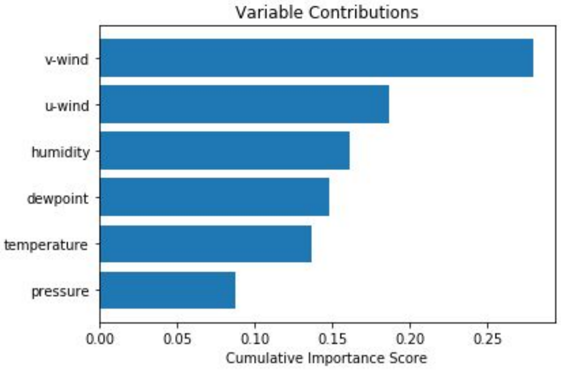
Known Key Features of Tornado Formation

V-wind importance at lower altitudes:
Low-altitude vertical wind shear correlates with the strength of the convergence and stretching by the supercell on developing vortices

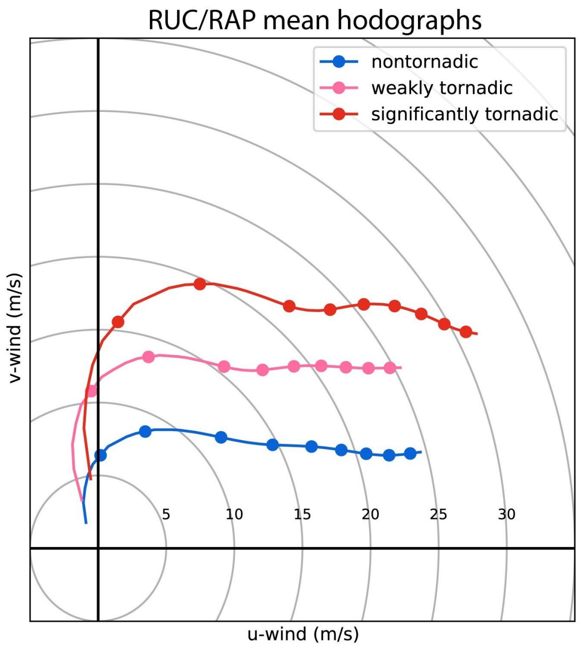
Relative Humidity Importance at Lowest Altitudes:
Low-altitude relative humidity measurements can predict downdraft coldness. Storms with colder downdrafts are less likely to undergo tornadogenesis

Storms with high low-altitude relative humidity and high vertical wind shear are more likely form tornadoes





Random Forest Feature Importance Analysis



Wind profiles for nontornadic, weakly tornadic, and significantly tornadic supercells. Dots represent 500 m, 1 km, 2 km, .etc above ground level.

- Hodograph depicts how wind velocities change with respect to height.
- Moving left to right on each curve, each dot represents the mean velocities at increasing heights above ground level.

Importance of v-wind feature

Larger v-wind results in “curved” hodographs and these curved hodographs lead to more intense supercells capable of producing tornadoes

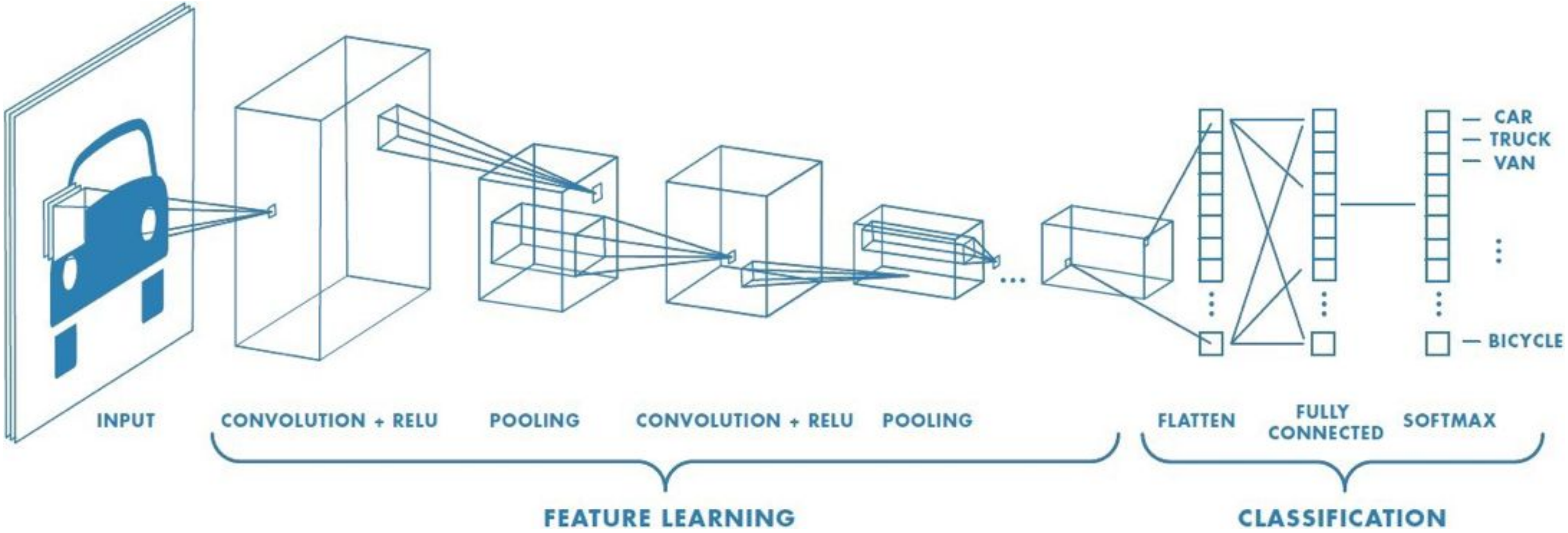
Outline

- Part I: Background & Motivation
  - What is a supercell?
  - RUCsounding Data
  - Research problem & motivation
- Part II: Data Preprocessing
- Part III: Random Forest
  - Results
  - Feature Importance Analysis
- Part IV: Convolutional Neural Network
  - Model Description
  - Results
- Part V: Over and Undersampling Applied to RF and CNN
  - Imbalanced Data
  - Results

Convolutional Neural Network

What is CNN?

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers and then followed by one or more fully connected layers as in a standard multilayer neural network.



https://towardsdatascience.com/a-comprehensive-guide-to-convolutional

Convolutional Layer

The first layer in a CNN is always a Convolutional layer which is to extract features from the input. Convolution preserves the spatial relationship between pixels by learning features using small portion of input data.

Original Image

1 1/9	6 1/9	3 1/9	2	9
2 1/9	11 1/9	3 1/9	10	0
5 1/9	10 1/9	6 1/9	9	7
3	1	0	2	8
4	4	2	9	10

Filtered Image

0	0	0	0	0
0	5			0
0				0
0				0
0	0	0	0	0

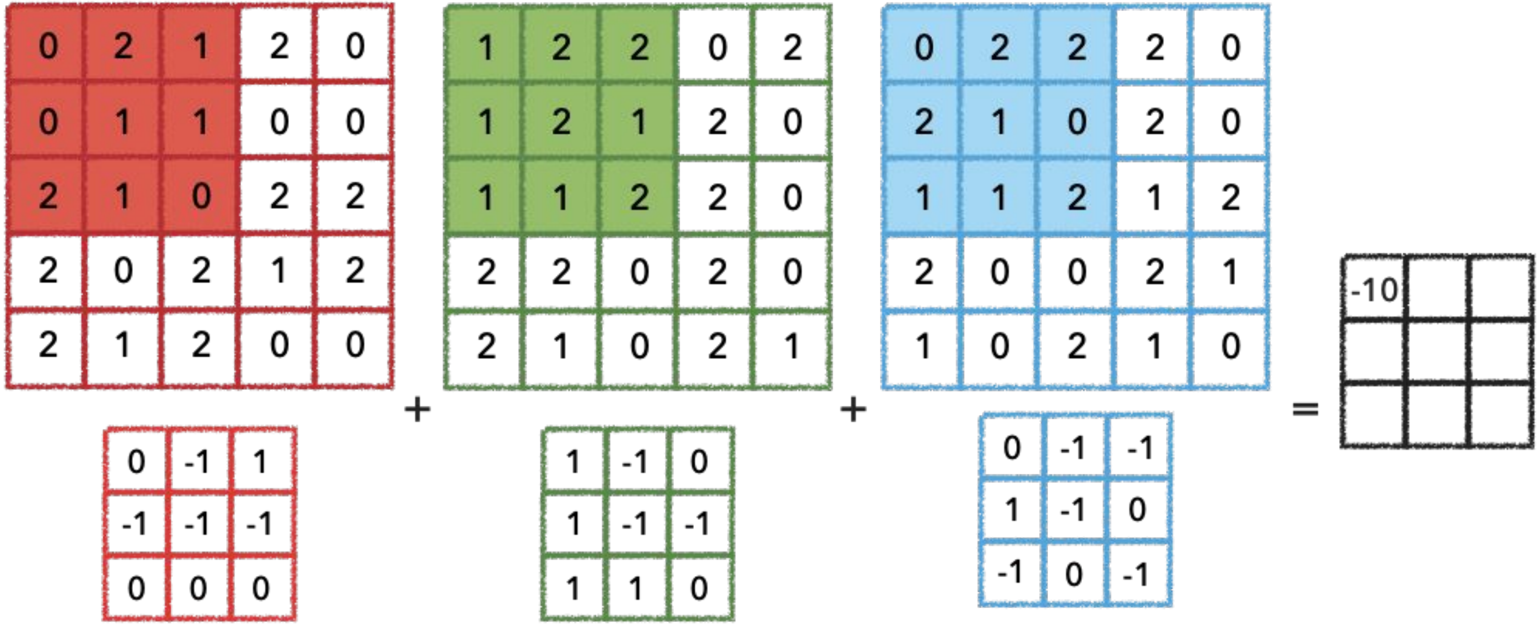
Channels

Channel is used to refer to a certain component of the input.
For example, an image from a standard digital camera has three channels - red, green and blue.

http://www.adsell.com/scanning101.html

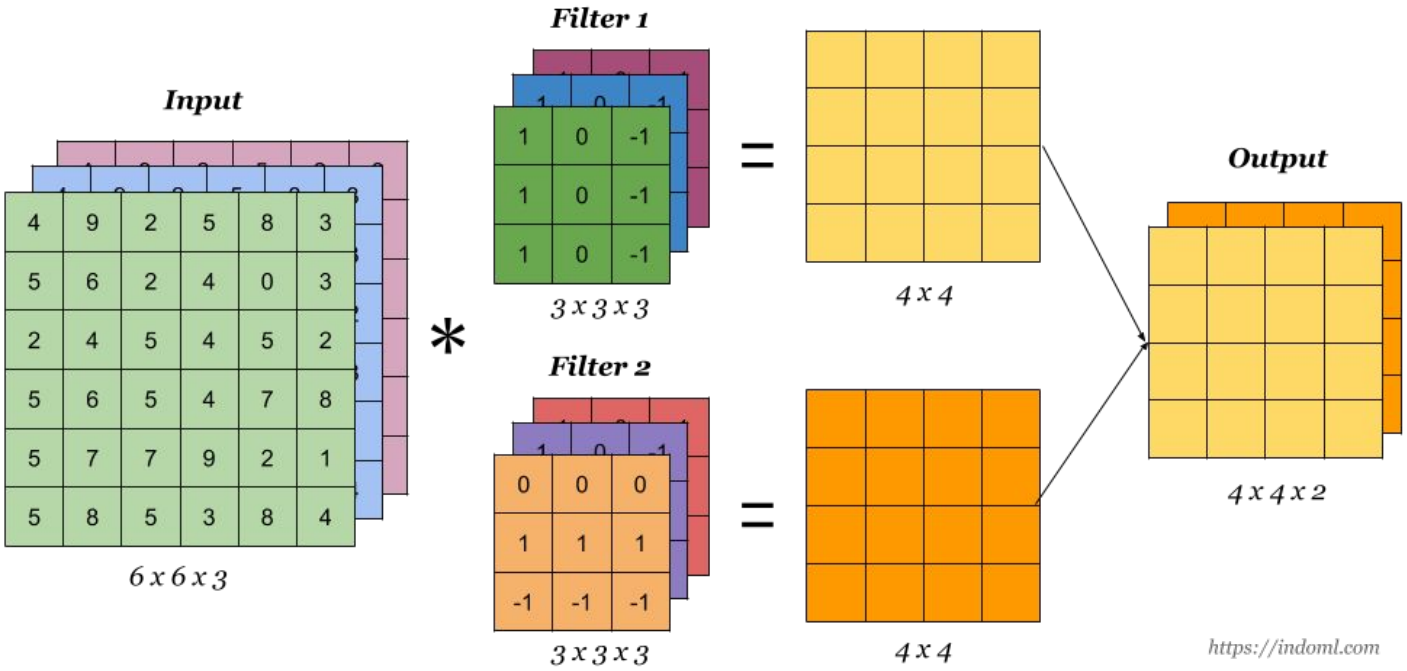
Convolution Operation on Channels

When the input has more than one channels, the filter should have the same number of channels. To calculate one output cell, perform convolution on each channel, then add the result together.



Convolution with Multiple Filters

Multiple filters can be used in a convolutional layer to detect multiple features. The output of the layer then will have the same number of channels as the number of filters applied in the layer.



Other Layers

Pooling: pooling layer is used to reduce the size of the feature representations and to speed up calculations.

- E.g. Max pooling

Fully connected (FC) layer: it is used to learn non-linear function from the output of the previous layers.

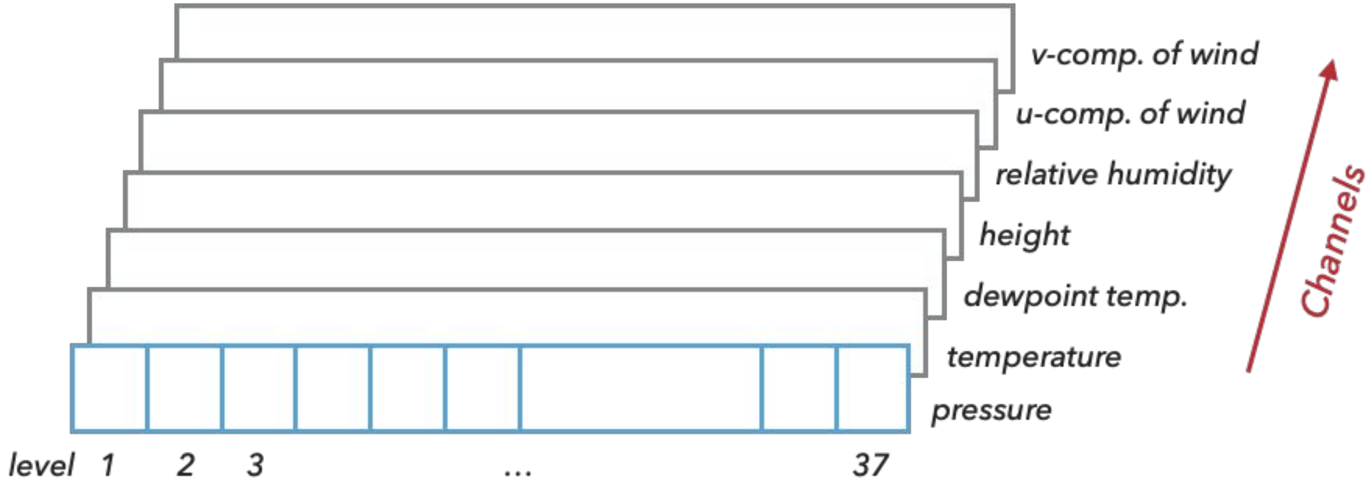
- Input to a FC layer is a 1D feature vector. So need to flat the 3D volume feature representation to a 1D vector.

Data Format for CNN

Special Feature: the data has 37 vertical levels. Thus, we are only interested in features in this dimension.

Channels: each vertical profile has 6 variables (components). So there will be 6 channels in our data: temperature, dewpoint, humidity, u-wind, and v-wind

Input Format: *20194 x 37 x 6*



CNN Model Updates: Model Structure

Input: interpolated RUCSounding Data (Train 80%, Test 20%)

Create CNN Model and Evaluate (Python keras)

Output Classifications:
[0] - Nontornadic
[1] - Weakly Tornadic (F0-F1)
[2] - Signif. Tornadic (F2-F5)

Output:
Accuracy

Basic model structure

Conv1D
64 filters

Conv1D
128 filters

MaxPooling
pool size = 2

Flatten

Dense Network
100 filters

Dense Network
softmax output

Output Classification

Accuracy:
65.365% +/-0.775%

CNN Model Structure

Simple model

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_26 (Conv1D)           (None, 64, 64)            1024
_________________________________________________________________
conv1d_27 (Conv1D)           (None, 62, 128)           24704
_________________________________________________________________
max_pooling1d_16 (MaxPooling (None, 31, 128)           0
_________________________________________________________________
flatten_11 (Flatten)         (None, 3968)              0
_________________________________________________________________
dense_22 (Dense)             (None, 100)               396900
_________________________________________________________________
dense_23 (Dense)             (None, 3)                 303
=================================================================
Total params: 422,931
Trainable params: 422,931
Non-trainable params: 0
_________________________________________________________________
```

Complicated model

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_23 (Conv1D)           (None, 64, 32)            512
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 32)            128
_________________________________________________________________
max_pooling1d_13 (MaxPooling (None, 32, 32)            0
_________________________________________________________________
conv1d_24 (Conv1D)           (None, 30, 64)            6208
_________________________________________________________________
batch_normalization_3 (Batch (None, 30, 64)            256
_________________________________________________________________
max_pooling1d_14 (MaxPooling (None, 15, 64)            0
_________________________________________________________________
conv1d_25 (Conv1D)           (None, 13, 128)           24704
_________________________________________________________________
batch_normalization_4 (Batch (None, 13, 128)           512
_________________________________________________________________
max_pooling1d_15 (MaxPooling (None, 6, 128)            0
_________________________________________________________________
flatten_10 (Flatten)         (None, 768)               0
_________________________________________________________________
dense_20 (Dense)             (None, 100)               76900
_________________________________________________________________
dense_21 (Dense)             (None, 3)                 303
=================================================================
Total params: 109,523
Trainable params: 109,075
Non-trainable params: 448
_________________________________________________________________
```

CNN Results: Skill Scores

Tornadic/Nontornadic (Classes [1,2] vs [0])

Score	Prediction	Forecasting
POD	0.80	0.71
FAR	0.23	0.34
CSI	0.65	0.52
TSS	0.43	0.40

Contingency Table		
Actual **Predicted**	**Yes [1,2]**	**No [0]**
Yes [1,2]	1299 (Hits)	676 (False alarms)
No [0]	533 (Misses)	1531 (Correct Rejects)

POD = Probability of Detection= Hits/(Observed)
FAR = False Alarm Ratio= (False Alarms)/(Predicted)
CSI = Critical Success Index = Hits/(All)
TSS = True Skill Statistic = (Hits*Rejects-False*Misses)/(Observed*None)

Observed= hits + misses
Predicted= hits + false alarms
All = hits + false alarms + misses
None = false alarms + correct rejects

RF vs. CNN: Significantly-Tornadic/Nontornadic

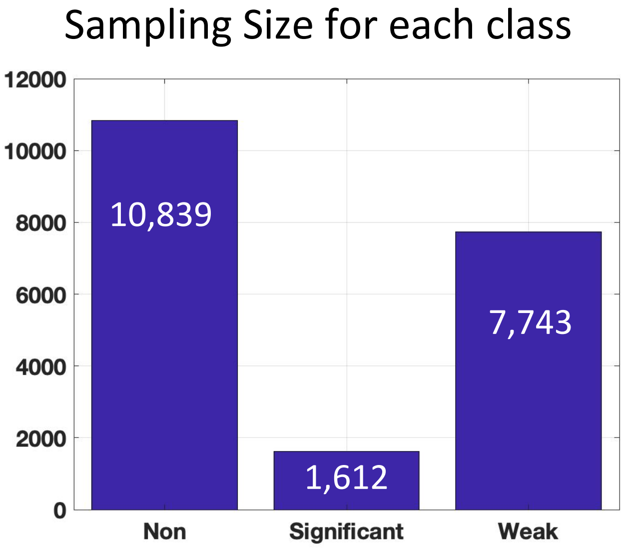
Simple CNN Model		
Actual Predicted	Yes [2]	No [0]
Yes [2]	6 (Hits)	11 (False alarms)
No [0]	211 (Misses)	1873 (Correct Rejects)

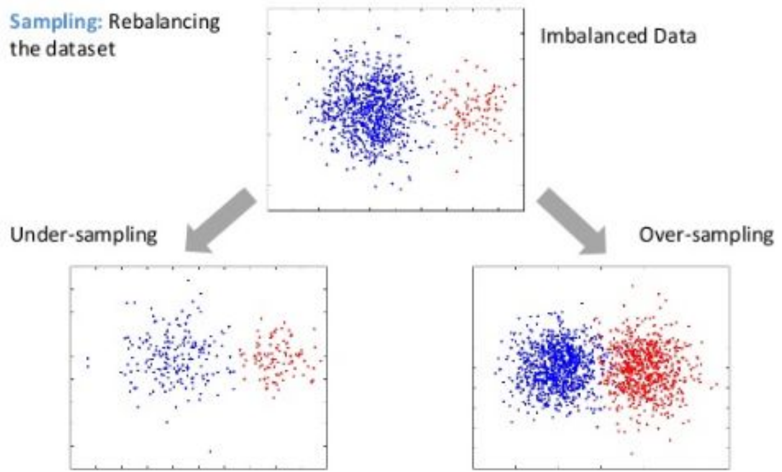
Complex CNN Model		
Actual Predicted	Yes [2]	No [0]
Yes [2]	75 (Hits)	43 (False alarms)
No [0]	83 (Misses)	1774 (Correct Rejects)

Score	Random Forest	Simple CNN	Complex CNN	Forecasting
POD	0.53	0.03	0.53	0.69
FAR	0.29	0.65	0.34	0.26
CSI	0.44	0.03	0.41	0.55
TSS	0.51	0.02	0.50	0.44

POD = Probability of Detection= Hits/(Observed)
FAR = False Alarm Ratio= (False Alarms)/(Predicted)
CSI = Critical Success Index = Hits/(All)
TSS = True Skill Statistic =
(Hits*Rejects-False*Misses)/(Observed*None)

Imbalanced dataset issue





imblearn.under_sampling.RandomUnderSampler
imblearn.over_sampling.RandomOverSampler

How to solve the imbalanced sampling issue:

- **Undersample the majority** class by deleting some samples at random in this class;
- **Oversample the minority** class by duplicating some samples at random;
- **Generate synthetic data** to adjust the ratio of majority and minority classes.

CNN Results: Undersampling/Oversampling

- Both undersampling and oversampling increase the class accuracy for the significantly tornadic class

- Undersampling severely impacts the accuracy of the nontornadic class

- Oversampling increases the accuracy of significantly tornadic class without decreasing the nontornadic class considerably

Class Accuracy	Before	Undersample	Oversample
Nontornadic	80.38%	56.59%	77.03%
Weak Tornadic	58.19%	58.19%	57.46%
Sig. Tornadic	26.69%	37.94%	36.01%

RF Results: Undersampling/Oversampling

- Undersampling severely impacts the accuracy of the nontornadic and weak tornadic classes, but increases the class accuracy for the significantly tornadic class greatly

- Oversampling increases the accuracy of significantly tornadic class without decreasing the nontornadic class considerably

Class Accuracy	Before	Undersample	Oversample
Nontornadic	83.28%	69.78%	83.58%
Weak Tornadic	60.94%	47.02%	55.43%
Sig. Tornadic	23.84%	65.85%	33.53%

We also applied oversampling using SMOTE() function which gave accuracies 78.4%, 58.6%, and 42.81% for nontornadic, weak tornadic, and significantly tornadic classes respectively.

RF and CNN with Under/Oversampling

Class Accuracy	RF Before	RF Undersample	RF Oversample	CNN Before	CNN Undersample	CNN Oversample
Nontornadic	83.28%	69.78% (-13.5%)	83.58% (+0.3%)	80.38%	56.59% (-24%)	77.03% (-3.3%)
Weakly-Tornadic	60.94%	47.02% (-14%)	55.43% (-5.5%)	58.19%	58.19% (0%)	57.46% (-0.7%)
Sig. Tornadic	23.84%	65.85% (+42%)	33.53% (+9.7%)	26.69%	37.94%(+11.2%)	36.01% (+9.3%)
Total Accuracy	70.14%	60.48%	68.43%	67.84%	58.26%	65.12%

- When apply oversampling strategy for imbalanced issue, CNN outperforms RF.
- When apply undersampling strategy, RF obtained better accuracies in both nontornadic and significantly tornadic classes than CNN does.

Detail Predictions of Best Models

RF with undersampling			
Predicted Actual	**[0]**	**[1]**	**[2]**
[0]	1478	403	237
[1]	380	749	464
[2]	31	81	216

CNN with oversampling			
Predicted Actual	**[0]**	**[1]**	**[2]**
[0]	1700	431	76
[1]	511	874	136
[2]	75	124	112

- In both RF and CNN models, the most missed significantly tornadoes are predicted as weak tornadoes.
- In RF model, the most missed weak tornadoes are predicted as significantly tornadoes while those are predicted as nontornadic in CNN model.

Conclusions and Future Work

Using **RUC sounding data,** we built and tested several **RF** and **CNN** models for predicting, ***in advance,*** when a supercell will generate a tornado.

- **RF Feature Analysis:** minimal contribution from pressure. Low-level v-wind, humidity, and dewpoint had high importance. Results agree with known key features of tornado formation.
- **RF Performance:** Undersampling nontornadic and weak tornadic significantly increases the accuracy of significant tornadoes prediction. However, it severely impacts the accuracy of the other two classes.
- **CNN Performance:** Oversampling significant tornado data gave best results for identifying significant tornadoes (36.01%) without drastically underdetecting nontornadic storms (77.03%).

Future Work

- Investigate and apply more alternative methods of training with imbalanced datasets.
- Use a <u>G</u>enerative <u>A</u>dversarial <u>N</u>etwork (**<u>GAN</u>**) to expand significant tornado sample size.
- Perform feature importance analysis on CNN. Compare to RF and current knowledge of tornado formation.
- Expand dataset to include 3D data surrounding tornado event, instead of just 1D profile