Significance-tested and physically constrained interpretation of a deep-learning model for tornadoes

- **Ryan Lagerquist**
  (ryan.lagerquist@noaa.gov, @ralager_Wx)
- Collaborators: Amy McGovern, David John Gagne II, Cameron Homeyer

- ECMWF-ESA Workshop on Machine Learning, Oct 7 2020

1. Motivation

- Tornadoes are too small to be resolved by operational dynamical models.
- Though many new datasets (*e.g.,* high-resolution satellite and dual-polarization radar) have potential to improve tornado prediction, they are not easily assimilated into dynamical models.
- Thus, machine learning (ML) is becoming a popular approach.
- **However, end users (including meteorologists) often do not trust ML, which has caused a push for interpretable ML.**

Image source:
https://en.wikipedia.org/wiki/2011_Joplin_tornado#/media/File:Joplin_2011_tornado_damage.jpg

1. Motivation

- Many interpretation tools have been developed in the ML literature, but they often produce noise that does not reflect true physical processes.
- **We have developed significance tests and physical constraints for 4 ML-interpretation methods.**
- **We apply these "augmented" interpretation methods to a convolutional neural network for tornado prediction.**

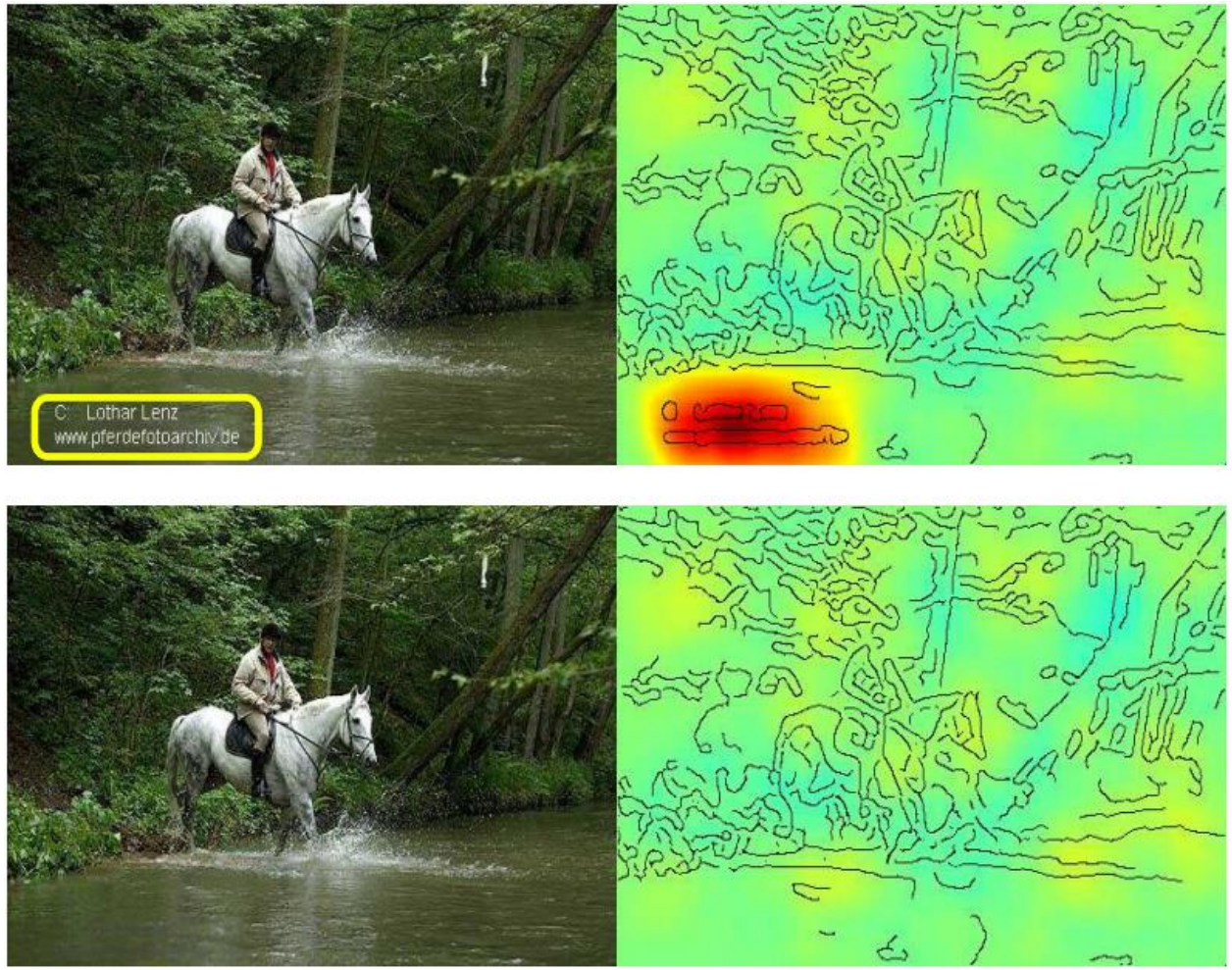


Image source: Figure 2a of Lapuschkin *et al.* (2019)

2. Input data

- **We use three datasets as input to the CNN:**
  - GridRad (Homeyer and Bowman 2017)
  - Rapid Refresh (RAP) dynamical weather model
  - Tornado reports in the Severe Weather Data Inventory (SWDI)

- **We use each dataset for a different purpose:**
  - GridRad: create storm-centered radar images to use as predictors
  - RAP: create near-storm (proximity) soundings to use as predictors
  - SWDI: create labels (“yes” if the storm is tornadic in the next hour, “no” otherwise)

2. Input data

- **GridRad is a multi-radar dataset**, created by combining and quality-controlling data from all WSR-88D radars in the continental United States.

- Resolution:
  - 5-minute time steps
  - 0.0208° horizontal spacing (~2 km)
  - 0.5-km vertical spacing from 0-7 km
  - 1.0-km vertical spacing aloft

- We acquired data for 154 days:
  - Training: 2012-14
  - Validation: 2015-18
  - Testing: 2011

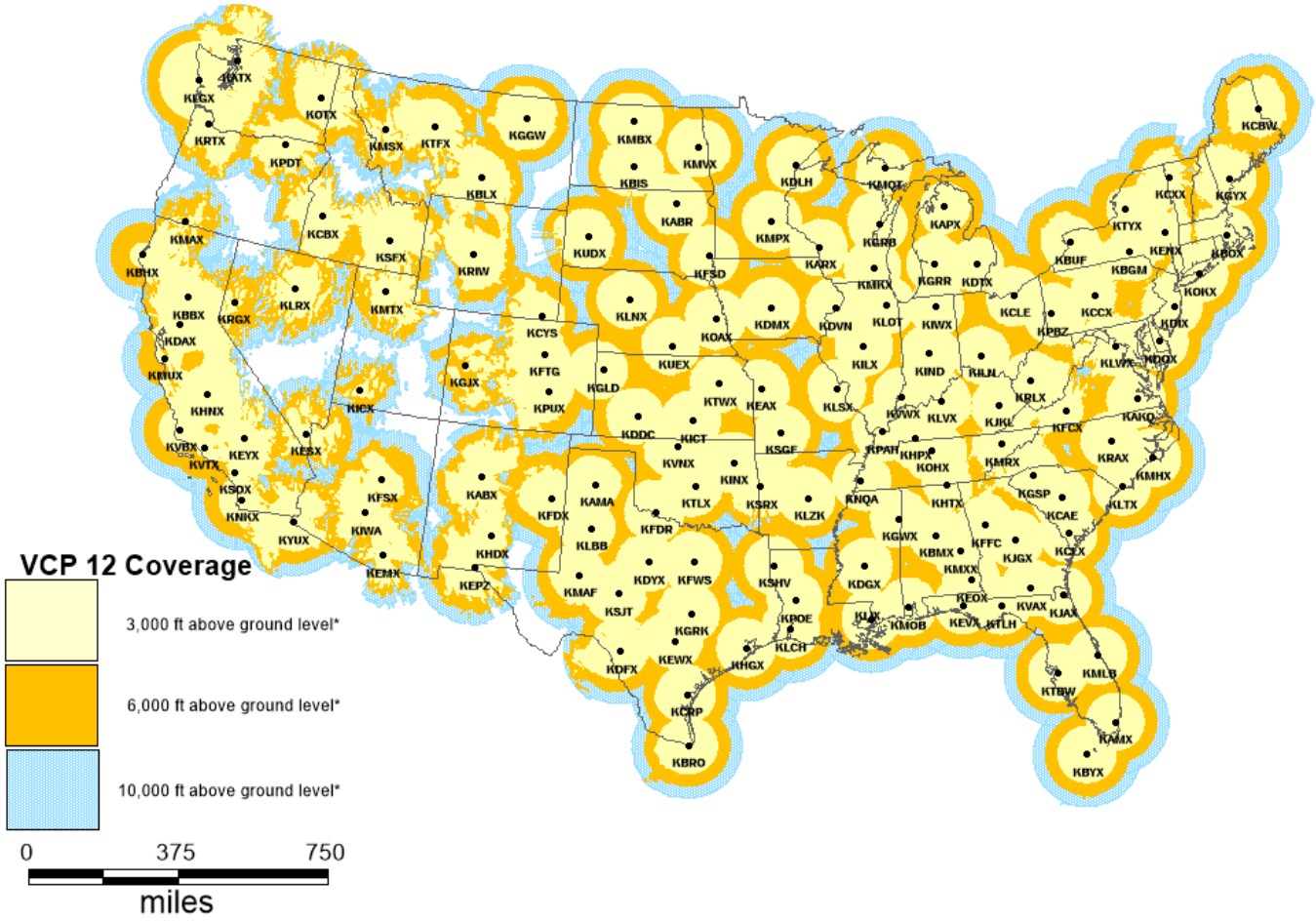
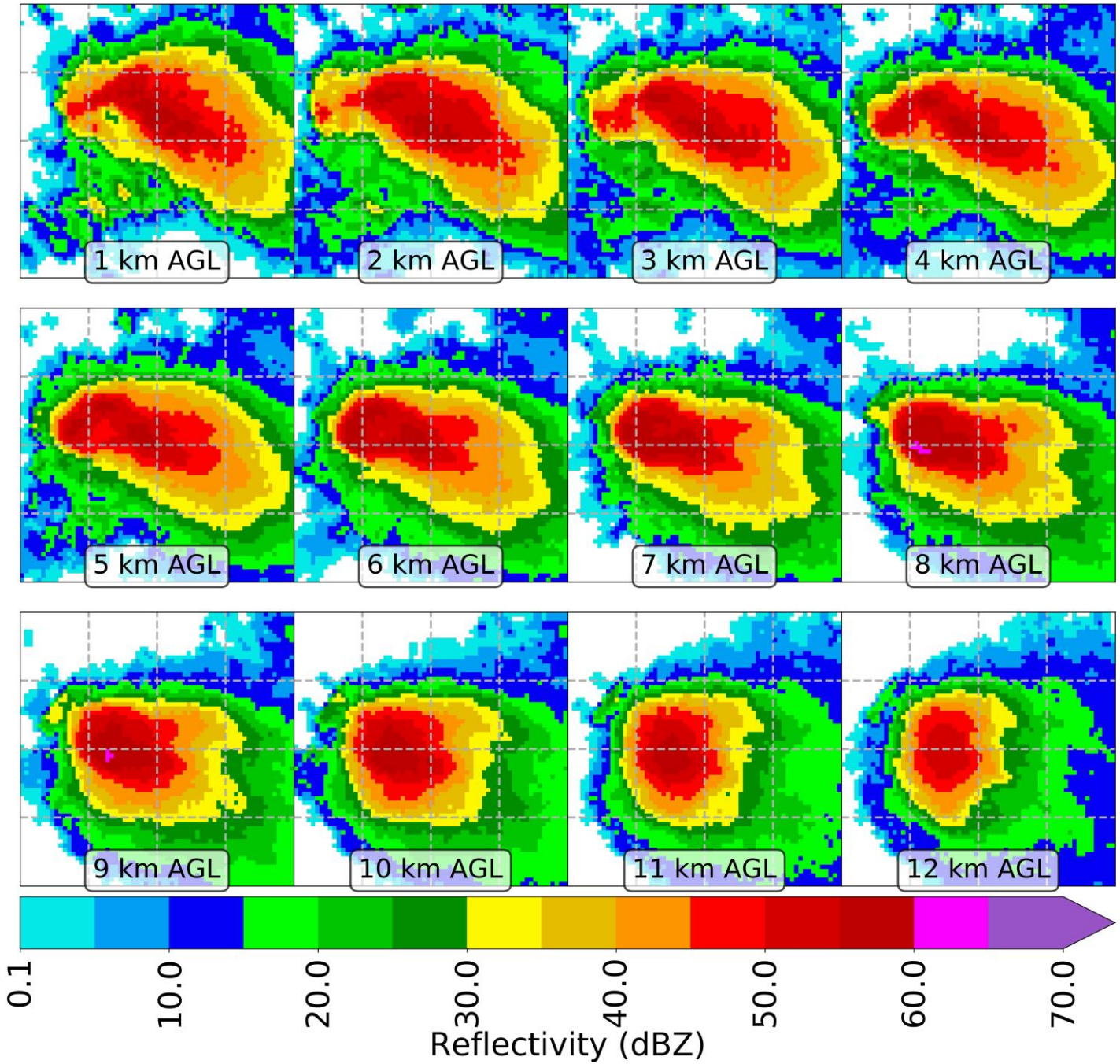


Image source: https://www.roc.noaa.gov/WSR88D/Maps.aspx

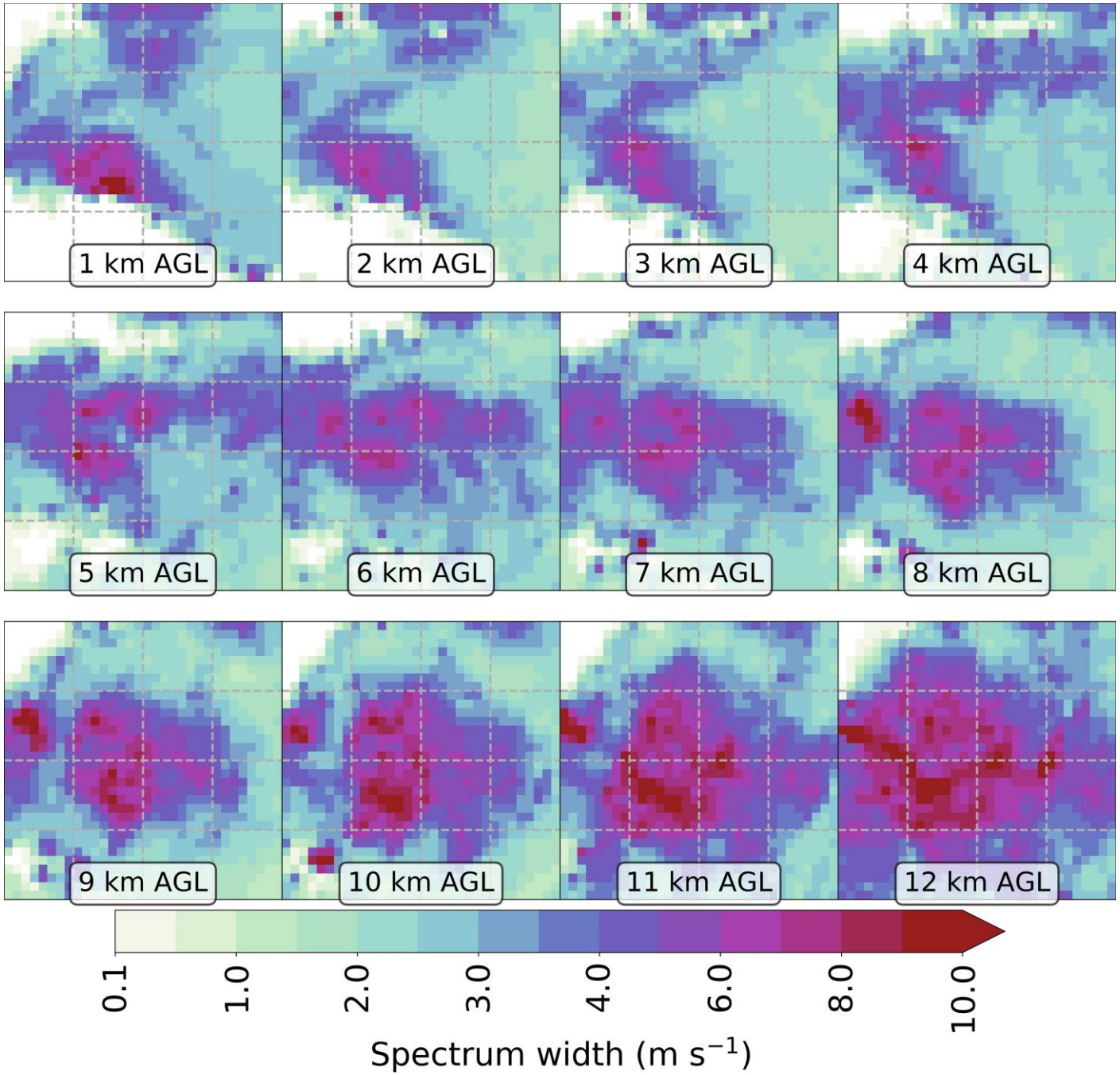
2. Input data

- **GridRad contains the following variables:**
  - **Reflectivity**
  - Velocity-spectrum width (increases with mean wind speed and turbulence)
  - Vorticity
  - Divergence



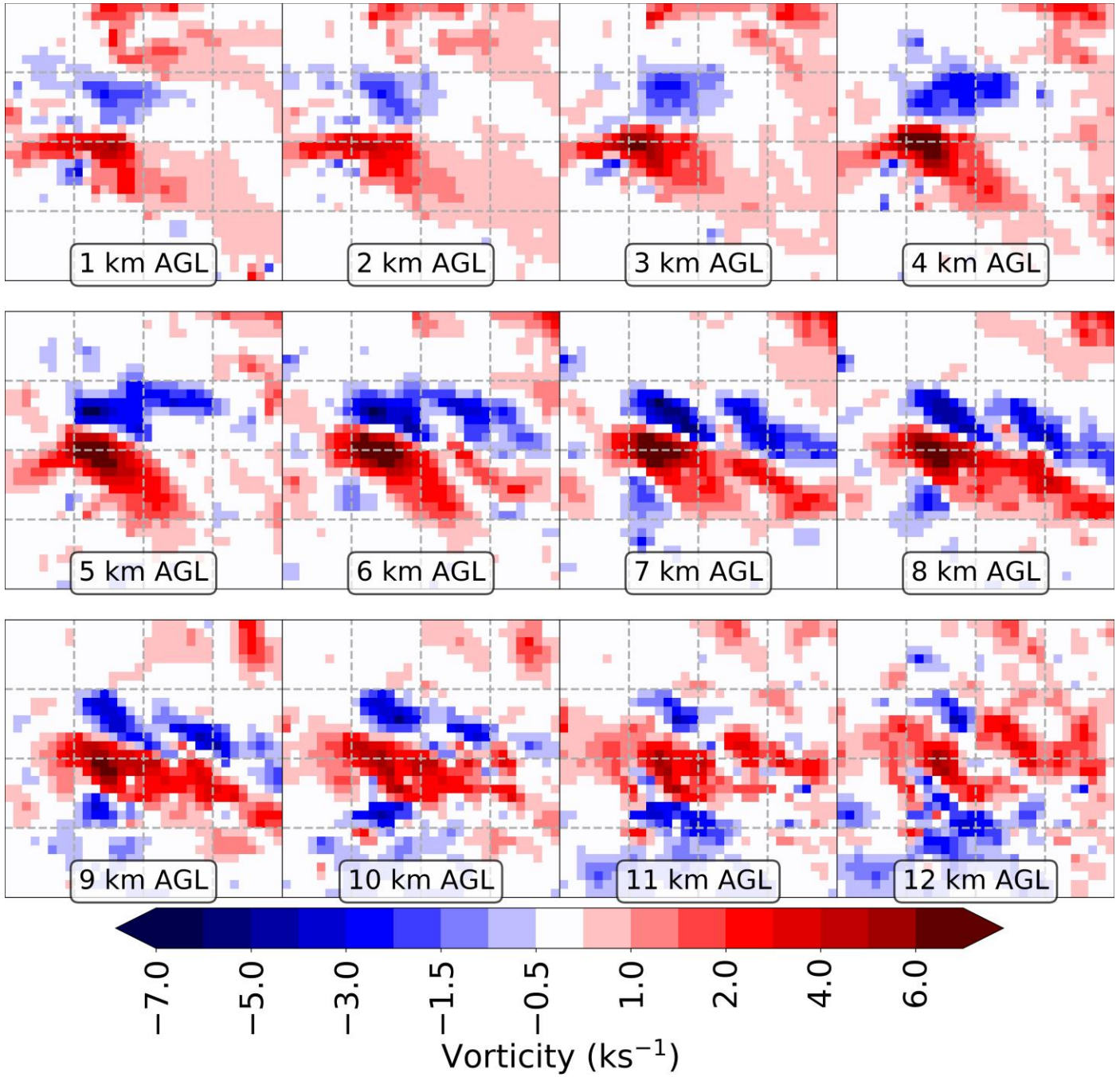
2. Input data

- **GridRad contains the following variables:**
  - Reflectivity
  - **Velocity-spectrum width** (increases with mean wind speed and turbulence)
  - Vorticity
  - Divergence



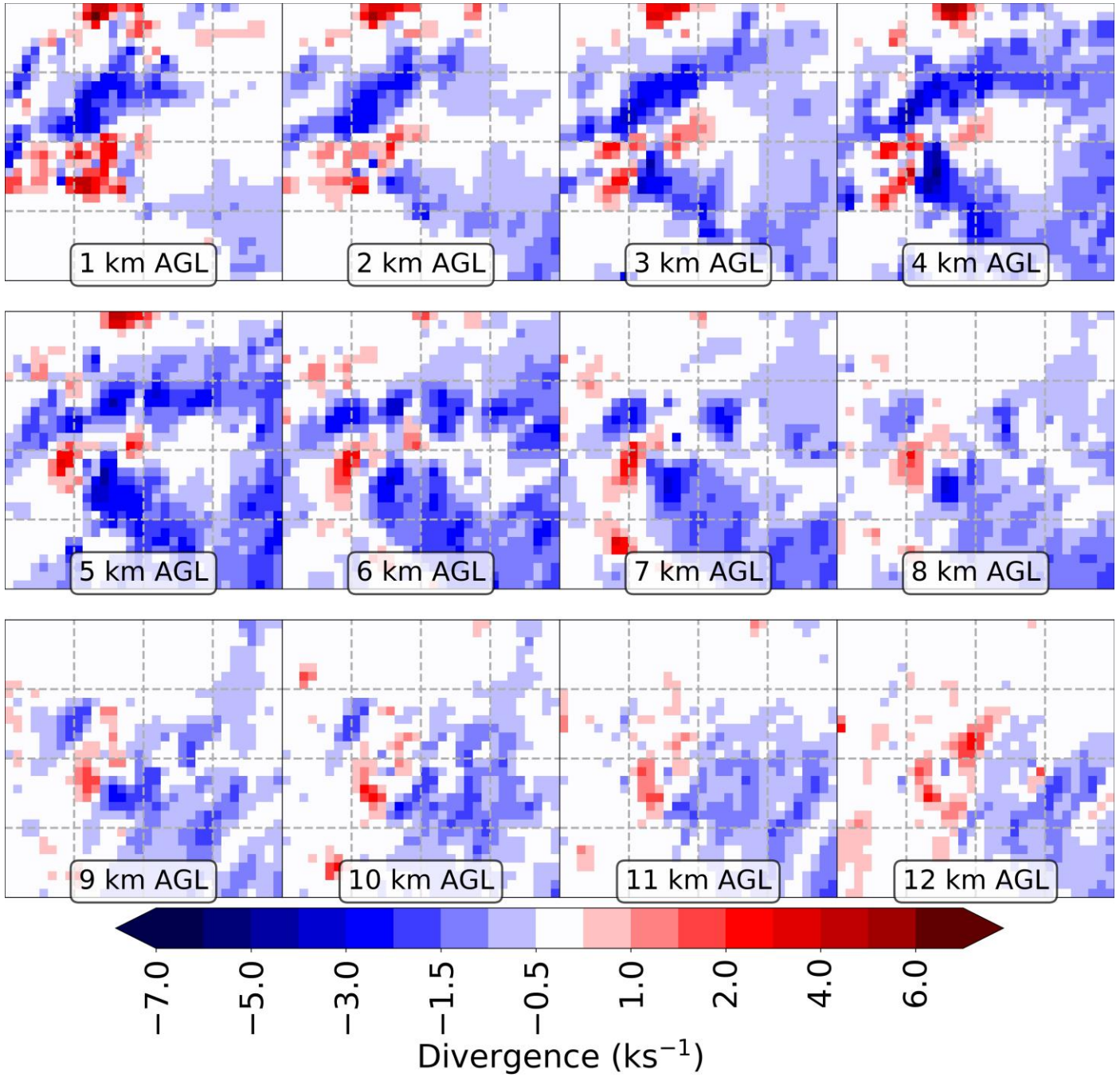
2. Input data

- **GridRad contains the following variables:**
  - Reflectivity
  - Velocity-spectrum width (increases with mean wind speed and turbulence)
  - **Vorticity**
  - Divergence



2. Input data

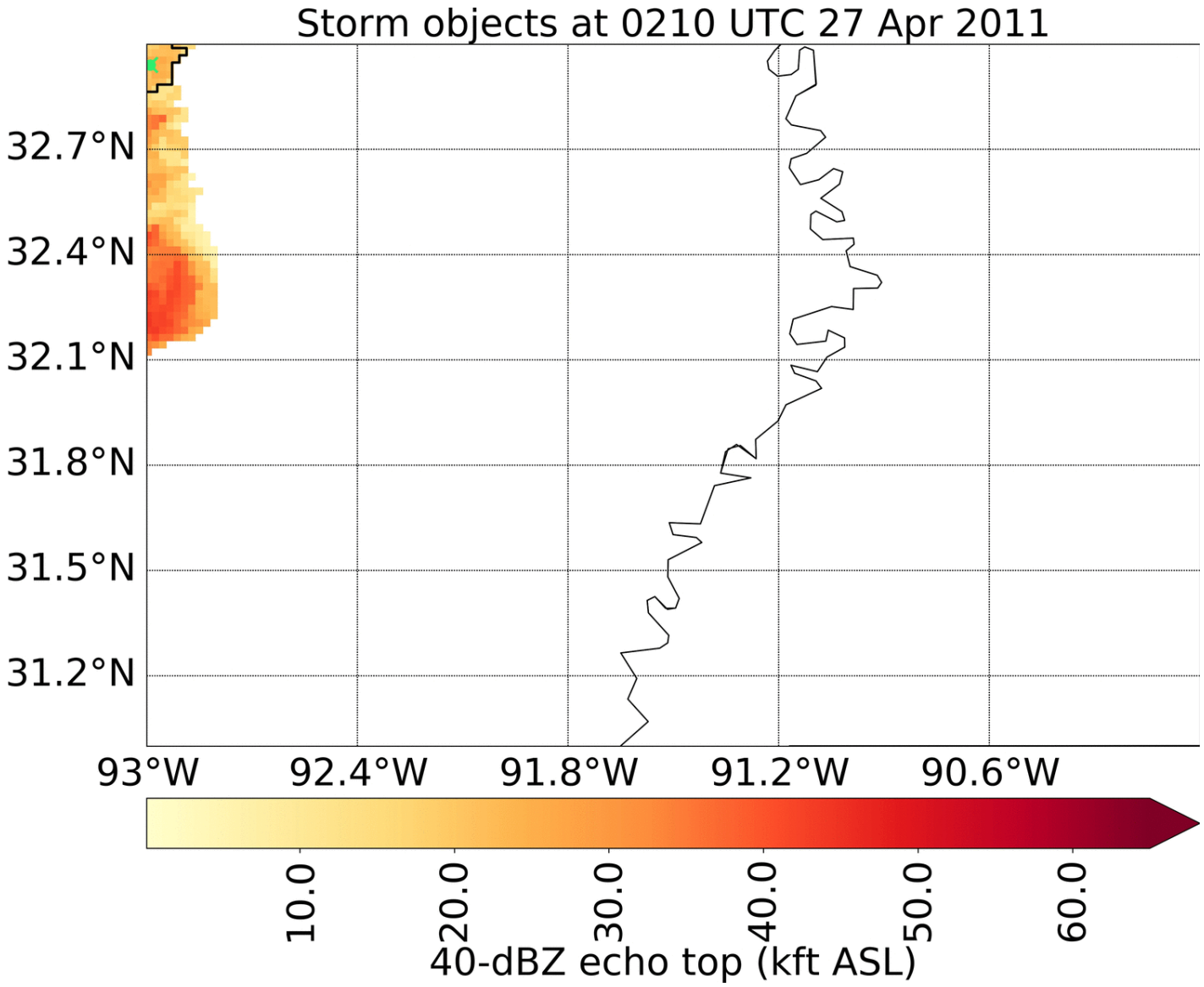
- **GridRad contains the following variables:**
  - Reflectivity
  - Velocity-spectrum width (increases with mean wind speed and turbulence)
  - Vorticity
  - **Divergence**



2. Input data

- **Pre-processing is summarized below** (details in Lagerquist *et al.* 2020).
- Each CNN example is one "storm snapshot" (one storm cell at one time).

1. **Outline storm cells at each time step**
2. **Track storm cells over time**
3. **Create storm-centered radar images** (to use as predictors)
   - One per storm snapshot
   - On equidistant grid with storm motion towards the right

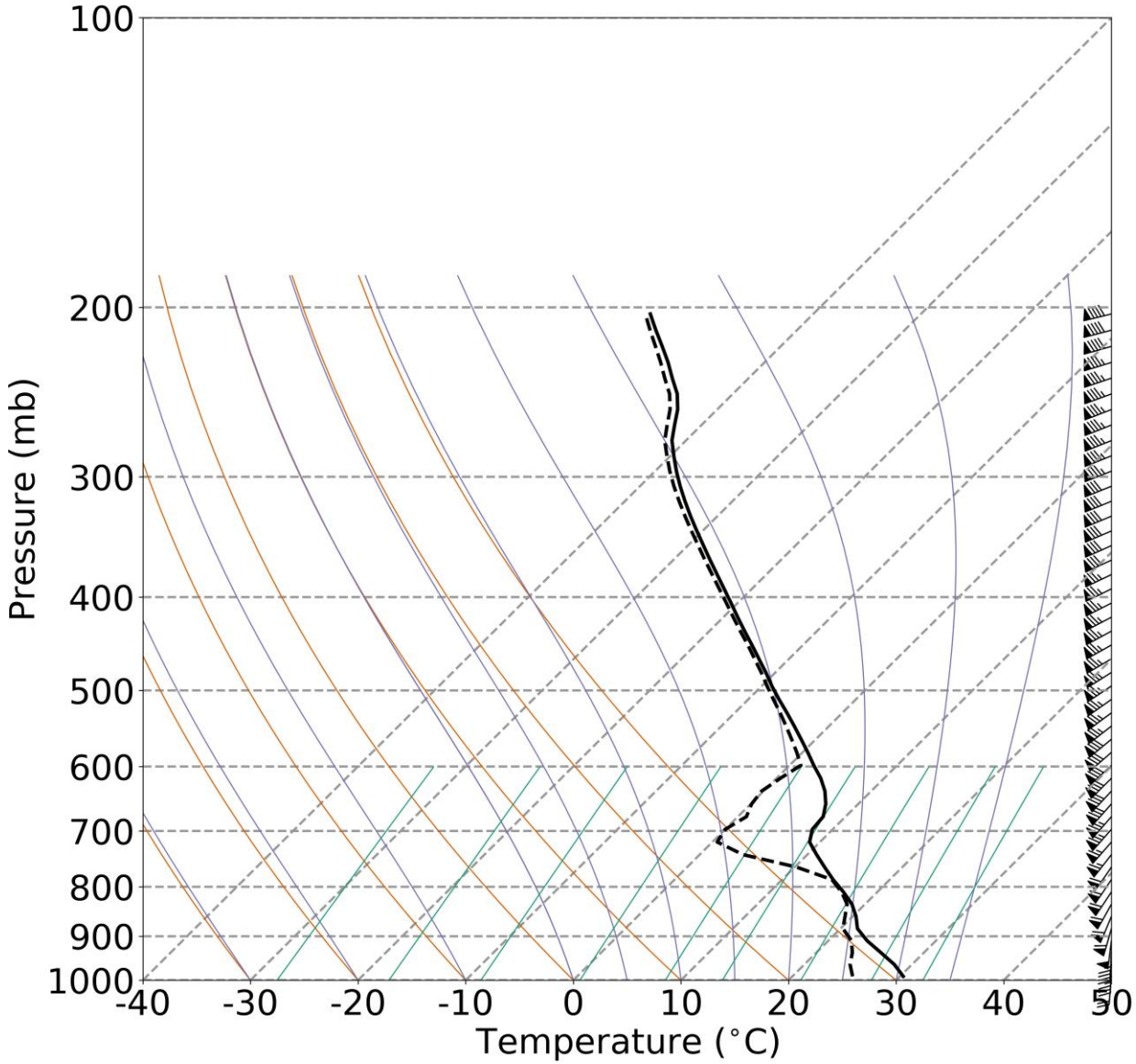


2. Input data

4. **Create proximity soundings (to use as predictors)**
   - One per storm snapshot
   - Represents near-storm environment over the next hour

5. **Link tornado reports to storms**

6. **Create labels (to use as targets)**
   - One per storm snapshot
   - “Yes” if storm is tornadic in the next hour, “no” otherwise



3. Model architecture and evaluation

- CNNs use three main types of layers:
  - Convolutional: to detect spatial features
  - Pooling: to decrease spatial resolution, so that further convolutional layers can detect larger-scale features
  - Fully connected: to map spatial features to predictions (here, tornado probabilities)

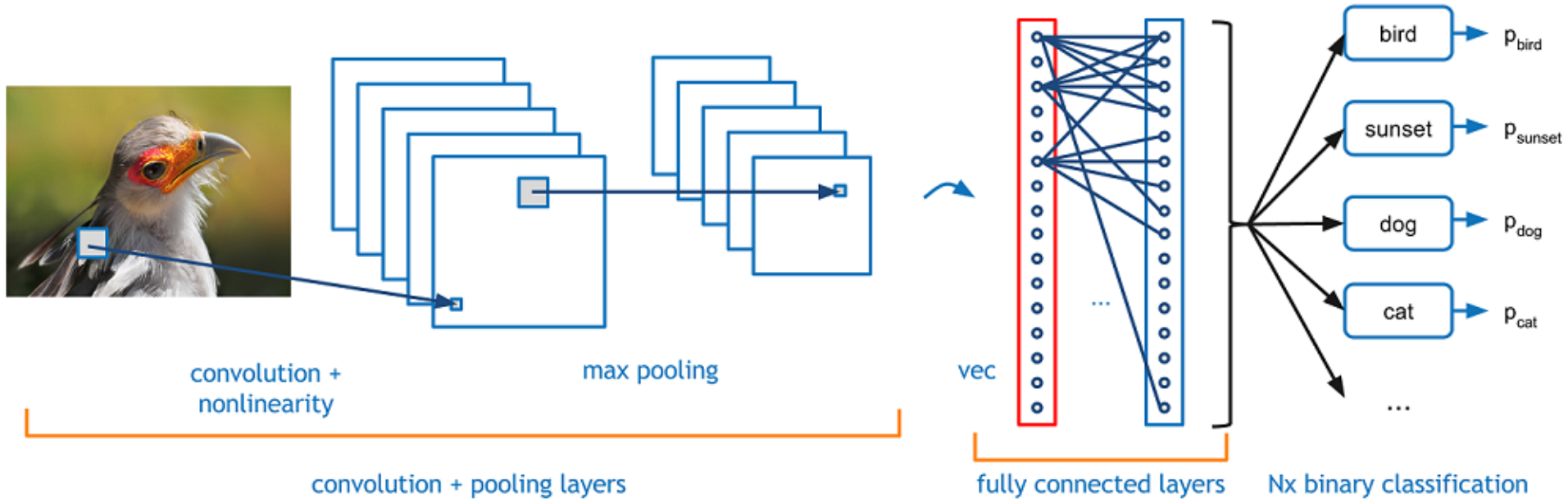
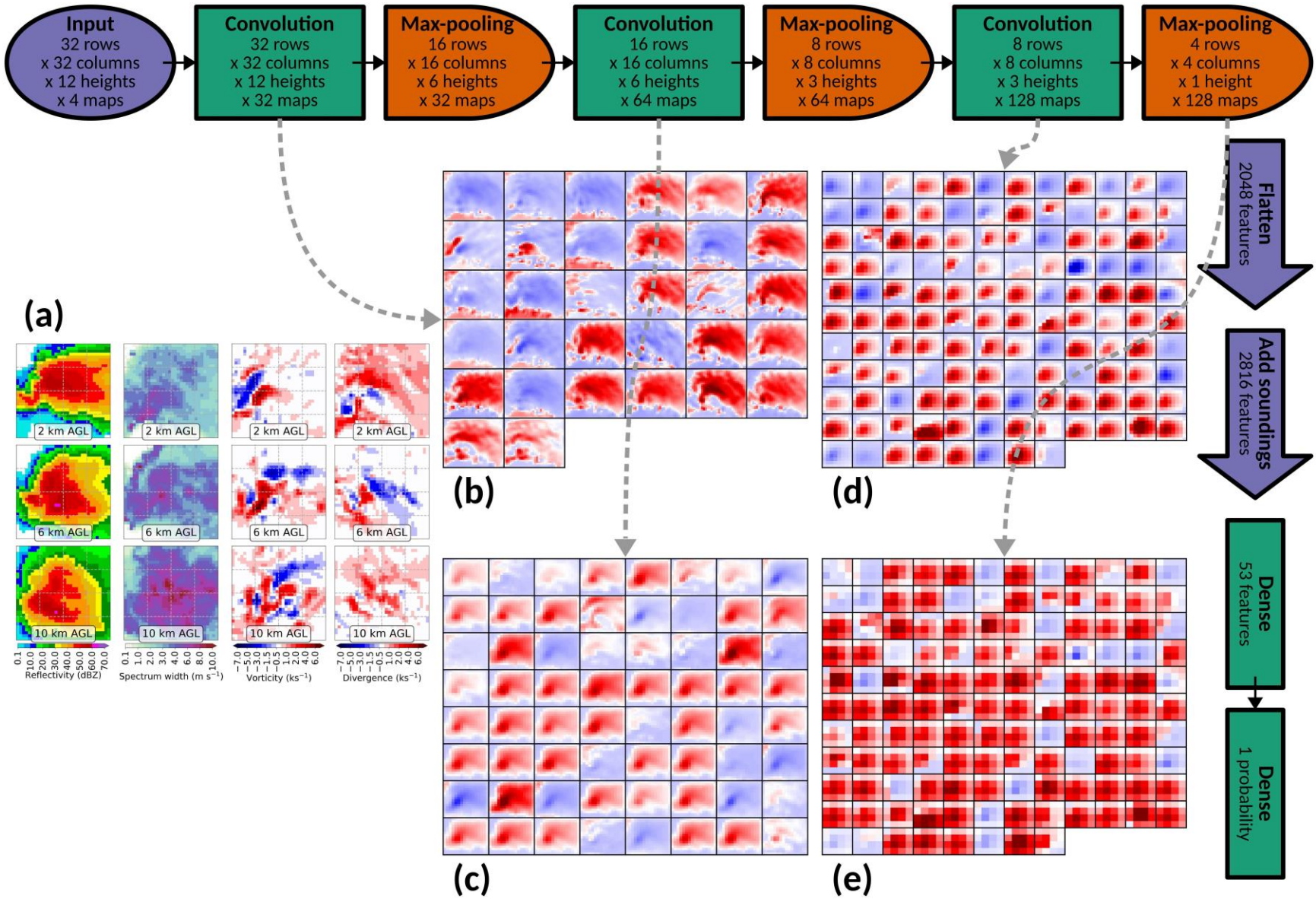


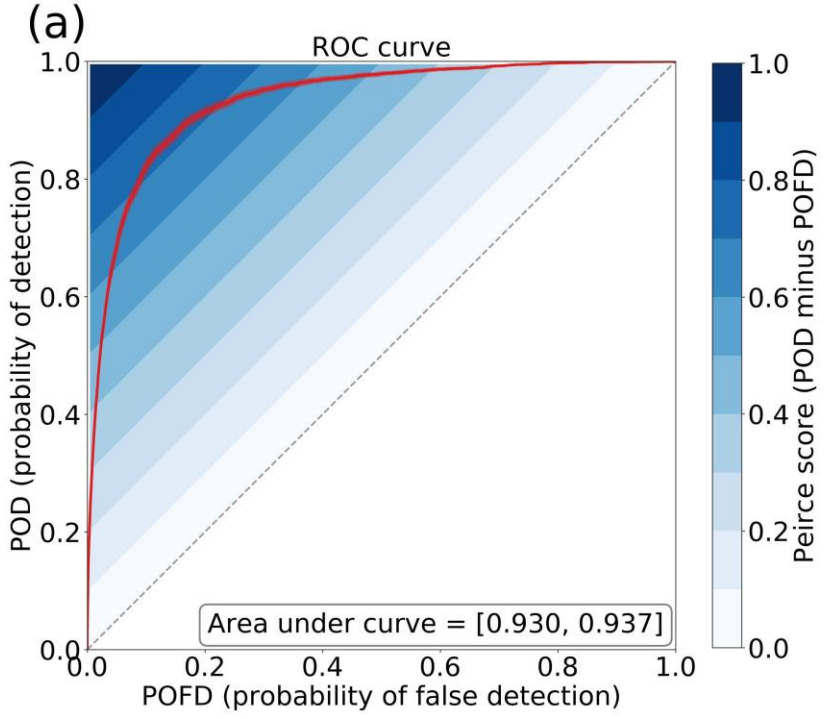
Image source:
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

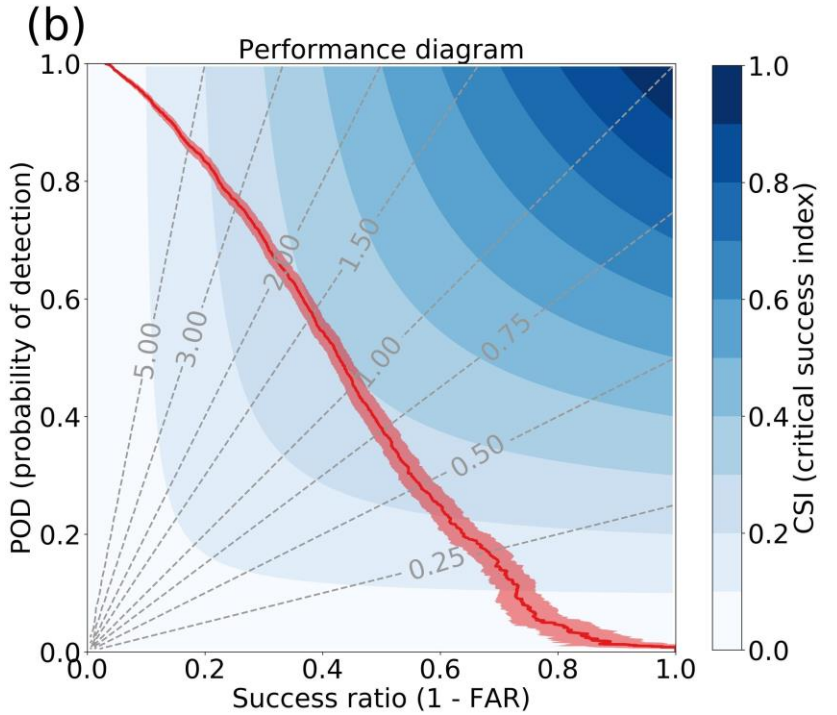
- **Right: architecture of our CNN for tornado prediction.**
- (a) Storm-centered radar image. Only three of the twelve heights (1, 2, …, 12 km AGL) are shown.
- (b-e) Feature maps produced by convolutional and pooling layers, at the lowest height only.
- **Another branch of the CNN uses 1-D convolution and pooling to detect spatial features in proximity soundings (not shown).**
- Pooling layers successively double grid spacing of radar image.
  - Horizontal spacing goes from 1.5 → 3 → 6 → 12 km.
  - Vertical spacing goes from 1 → 2 → 4 → 8 km.



3. Model architecture and evaluation

- **Right: results on testing data.**
- Testing set contains 137 270 examples with 3.43% event frequency.
  - In other words, only 3.43% of storms are tornadic in the next hour.
- Area under ROC curve > 0.9, generally considered “excellent” performance.
- Maximum CSI is low (0.30), but this is typical for rare events (difficult to achieve high POD with low FAR).
- **Upshot: model performance is good enough to motivate model interpretation.**





4. The permutation test

- **The permutation test measures the overall importance of each predictor variable.**
  - "Overall" = averaged over all testing examples
  - Applied to the trained CNN; does not involve retraining the CNN

- "Importance" of predictor x_j = how much CNN performance declines when x_j is permuted.

- **"Permutation" means randomly shuffling maps of x_j, so that they are assigned to the wrong examples.**

- **If performance declines significantly, x_j is important.**

- If performance does not decline significantly, x_j is either unimportant or too strongly correlated with other predictors.

4. The permutation test

- **There are 4 versions of the permutation test:**
  - Single-pass forward (Breiman 2001)
  - Multi-pass forward (Lakshmanan *et al.* 2015)
  - Single-pass backwards
  - Multi-pass backwards

- **The 4 versions give different results when predictors are strongly correlated, similar results otherwise.**
  - For a detailed explanation, see Section 7.1.1 of Lagerquist (2020).

- Thus, we run all 4 versions and compare the results.

4. The permutation test

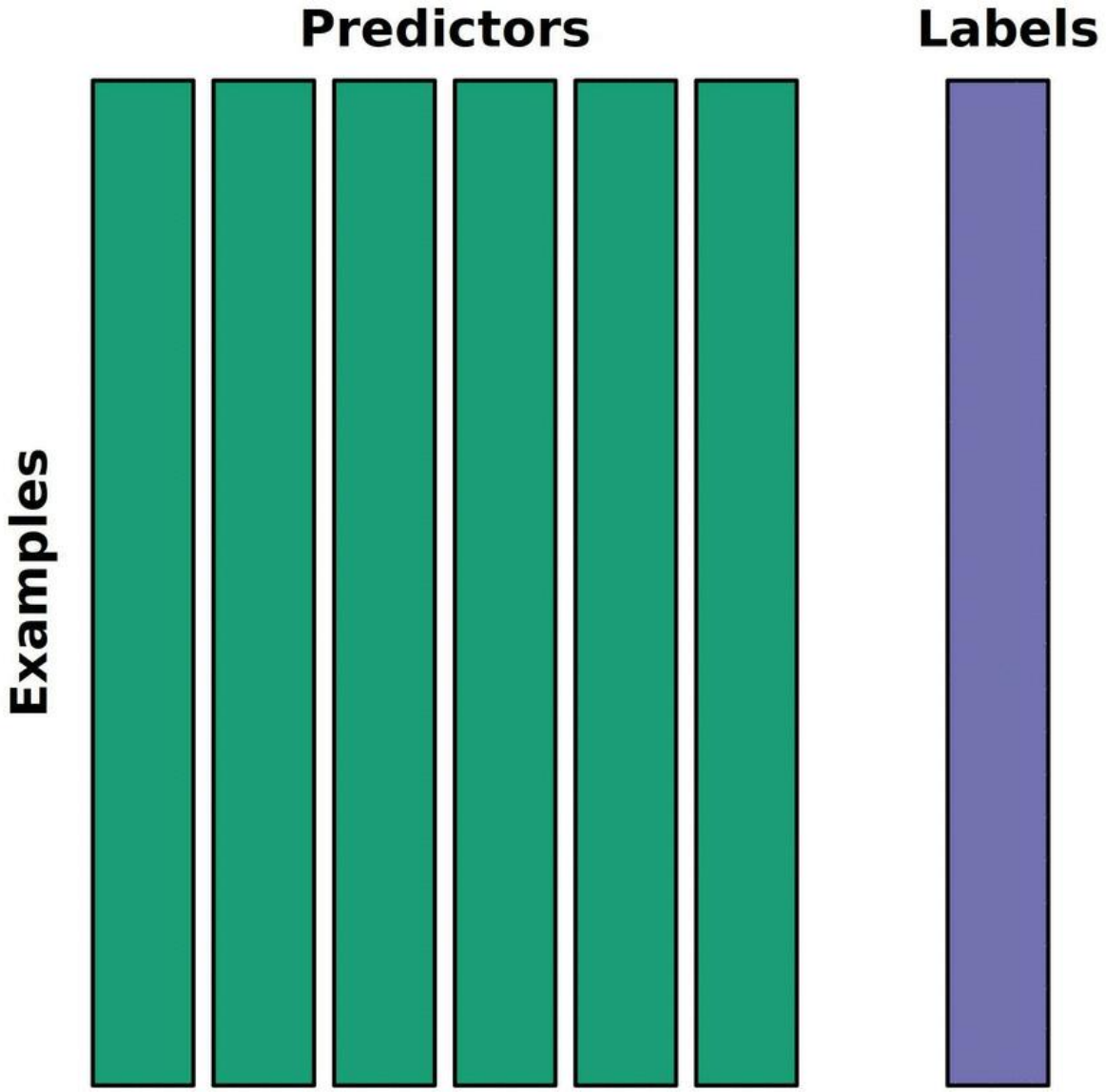
- **GIF at right: single-pass forward test.**

- **The procedure is below**, letting $\mathcal{F}$ be the trained model and $\boldsymbol{X}$ be the clean predictor set.
  - "Clean" means no predictors are permuted.

1. Repeat the following for each predictor x_j:
   a) Copy the dataset $\boldsymbol{X}$ to a new variable, $\boldsymbol{X}'$.
   b) Permute values of x_j over all examples in $\boldsymbol{X}'$.
   c) Pass $\boldsymbol{X}'$ through $\mathcal{F}$ and record the new loss.

2. Rank predictors by loss after permutation.

- **The *k*th-most important predictor is that whose permutation causes the *k*th-highest loss.**



4. The permutation test

- **GIF at right: multi-pass forward test.**

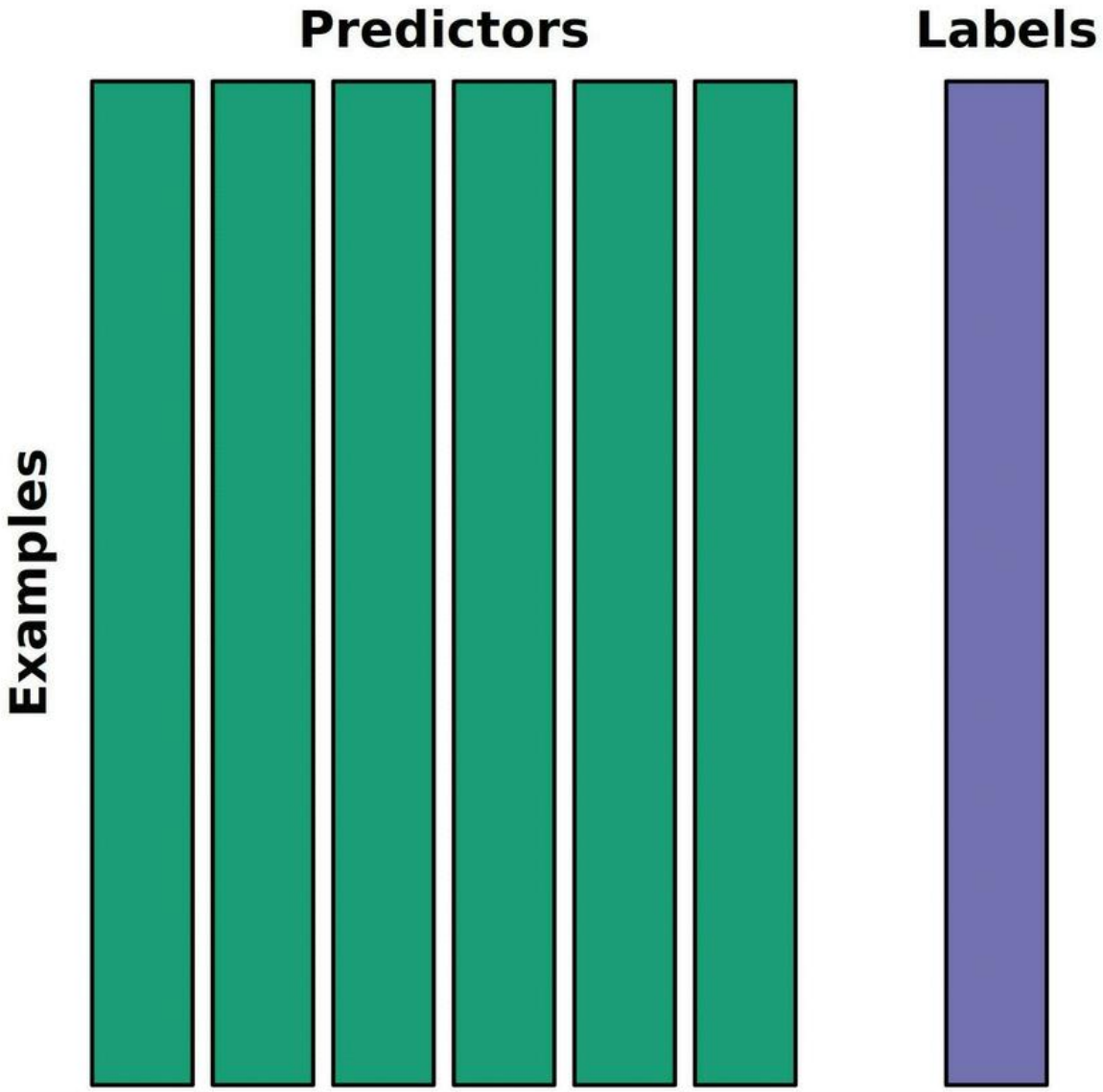
- **The procedure is below**, letting $\mathcal{U}$ be the set of predictors that are not permanently permuted.

1. $\mathcal{U} = \{\text{all predictors}\}$

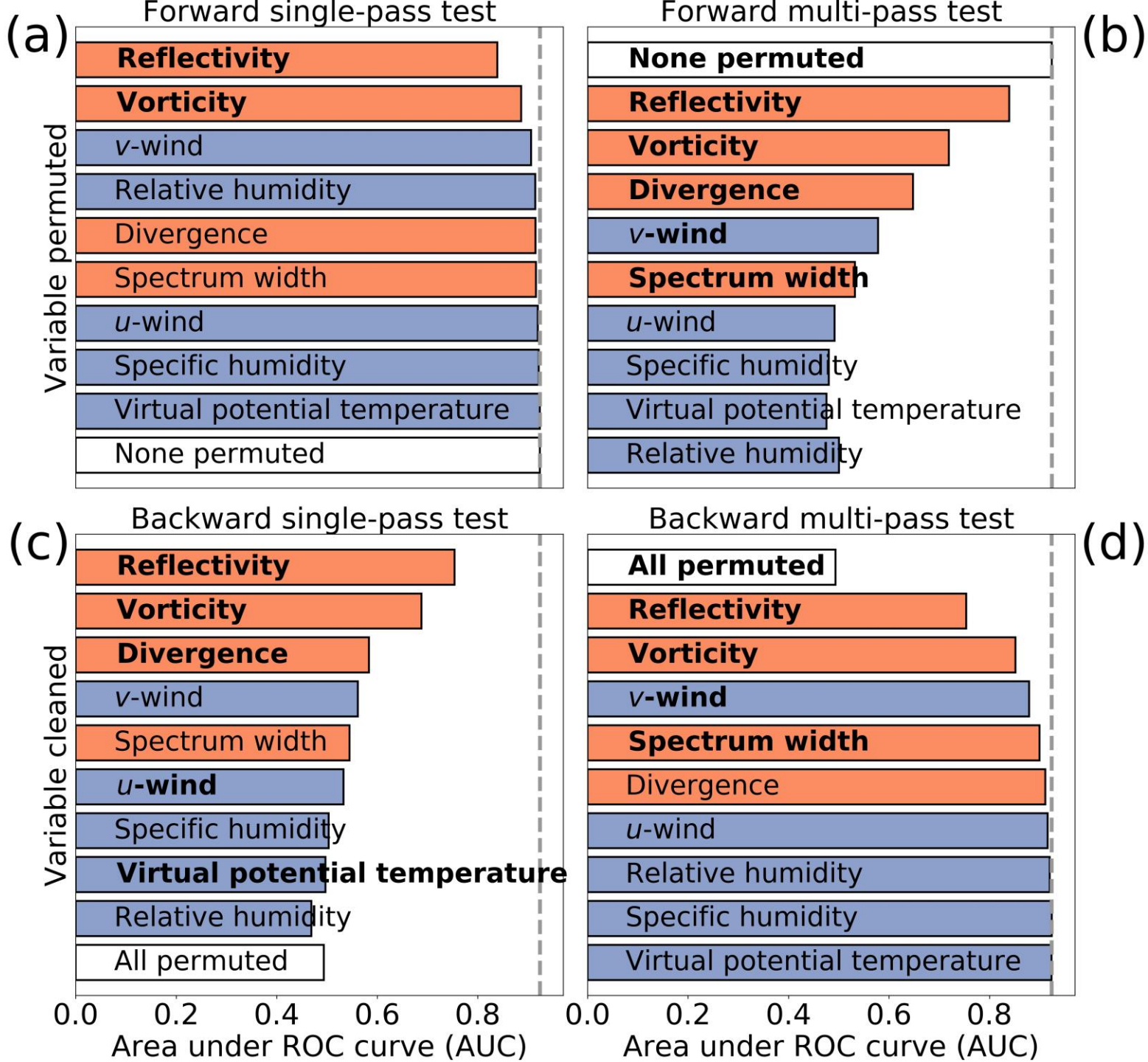
2. Run the single-pass forward test for all predictors in $\mathcal{U}$. Permanently permute the one that causes the highest loss.

3. Repeat step 2 until $\mathcal{U}$ is empty.

- **The *k*th-most important predictor is the *k*th to be permanently permuted.**



- **Results shown at right.**
- Radar variables in orange, sounding variables in purple.
- In each panel, the most (least) important predictor is at the top (bottom).
- **Bold font means that the predictor is significantly (at 95% level) more important than the one below it.**
- This is determined by bootstrapping the difference between post-permutation losses 1000 times.
- **According to all 4 versions of test:**
  - Reflectivity and vorticity are most important overall
  - *v*-wind is most important sounding variable



5. Saliency maps

- Saliency (Simonyan *et al.* 2014), or "sensitivity," is defined below:

$$s = \left.\frac{\partial p}{\partial x}\right|_{x=x_0}$$

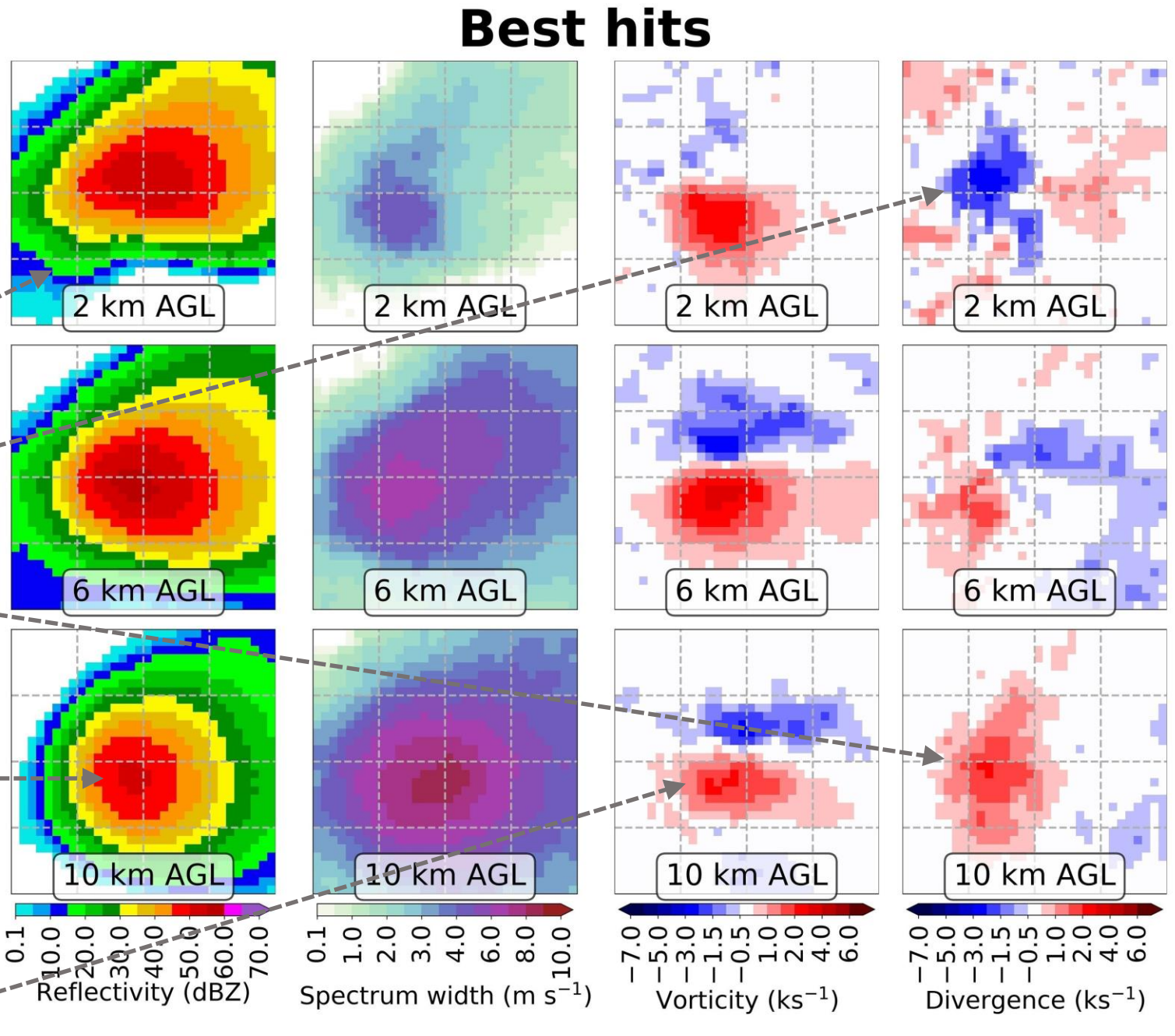
- p = activation of output neuron = tornado probability
- x = one input (one predictor variable at one grid point)
- x_0 = x-value in a real example

- In plain English: saliency is a linear approx to $\frac{\partial p}{\partial x}$, valid around the x-value found in a real example.

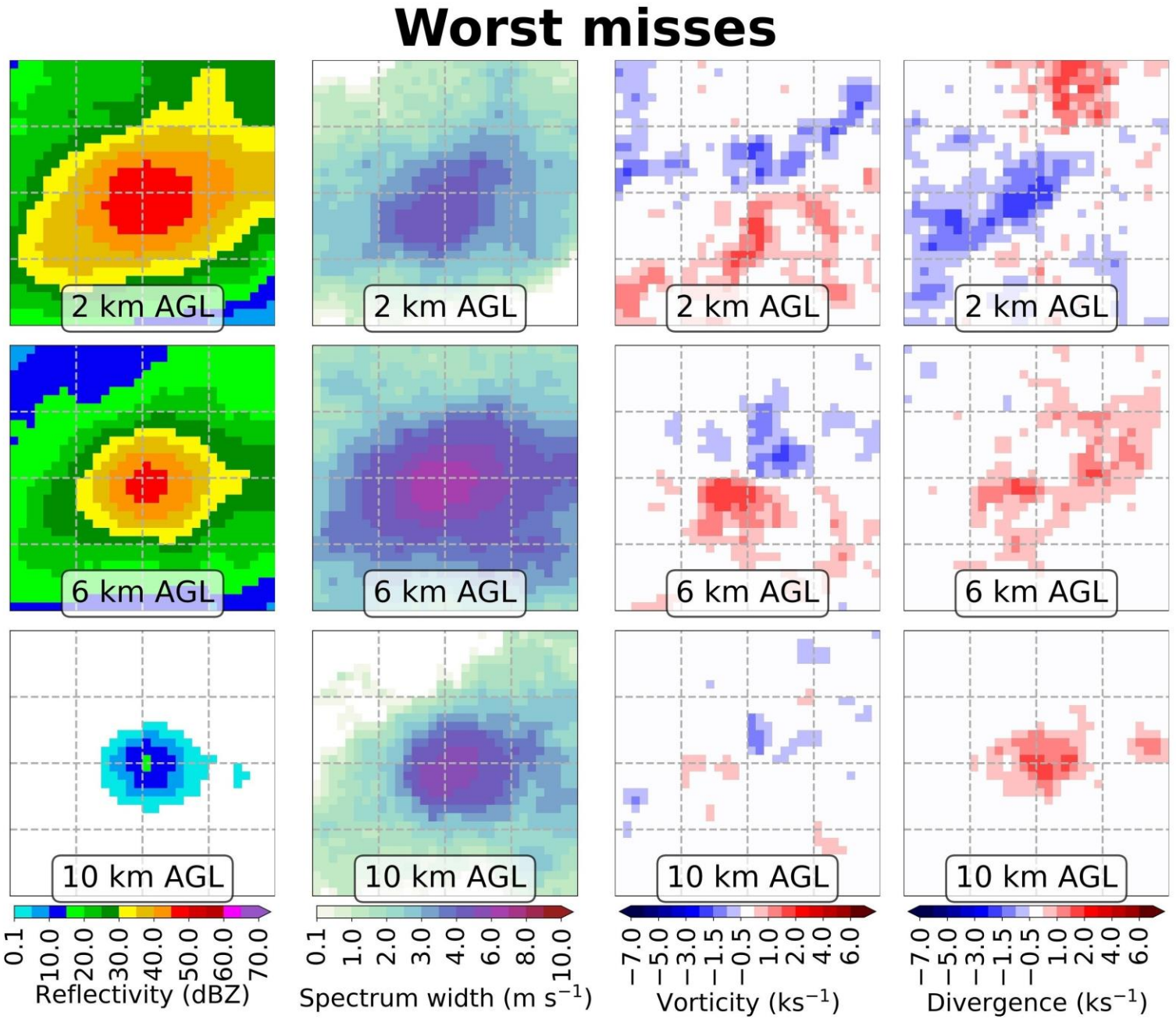
- **Thus, our saliency maps answer the following question (from the CNN's perspective):**

"How would you change the storm to increase/decrease tornado probability?"

- **Right: composite (average) radar image for the 100 best hits.**
- "Best hits" = positive cases with highest CNN probabilities (average of 99.2%).
- **Composite looks like a supercell.**



- **Right: composite (average) radar image for the 100 worst misses.**
- "Worst misses" = positive cases with lowest CNN probabilities (average of 8.6%).
- **These storms have weak rotation and shallow, elongated reflectivity core.**
- This makes sense, as 67 of the 100 storms are part of quasi-linear convective systems (**QLCS**).
- QLCS storms generally produce weaker tornadoes, and these tornadoes are commonly missed by humans and other forecasting methods (Brotzge *et al.* 2013; Anderson-Frey *et al.* 2016).



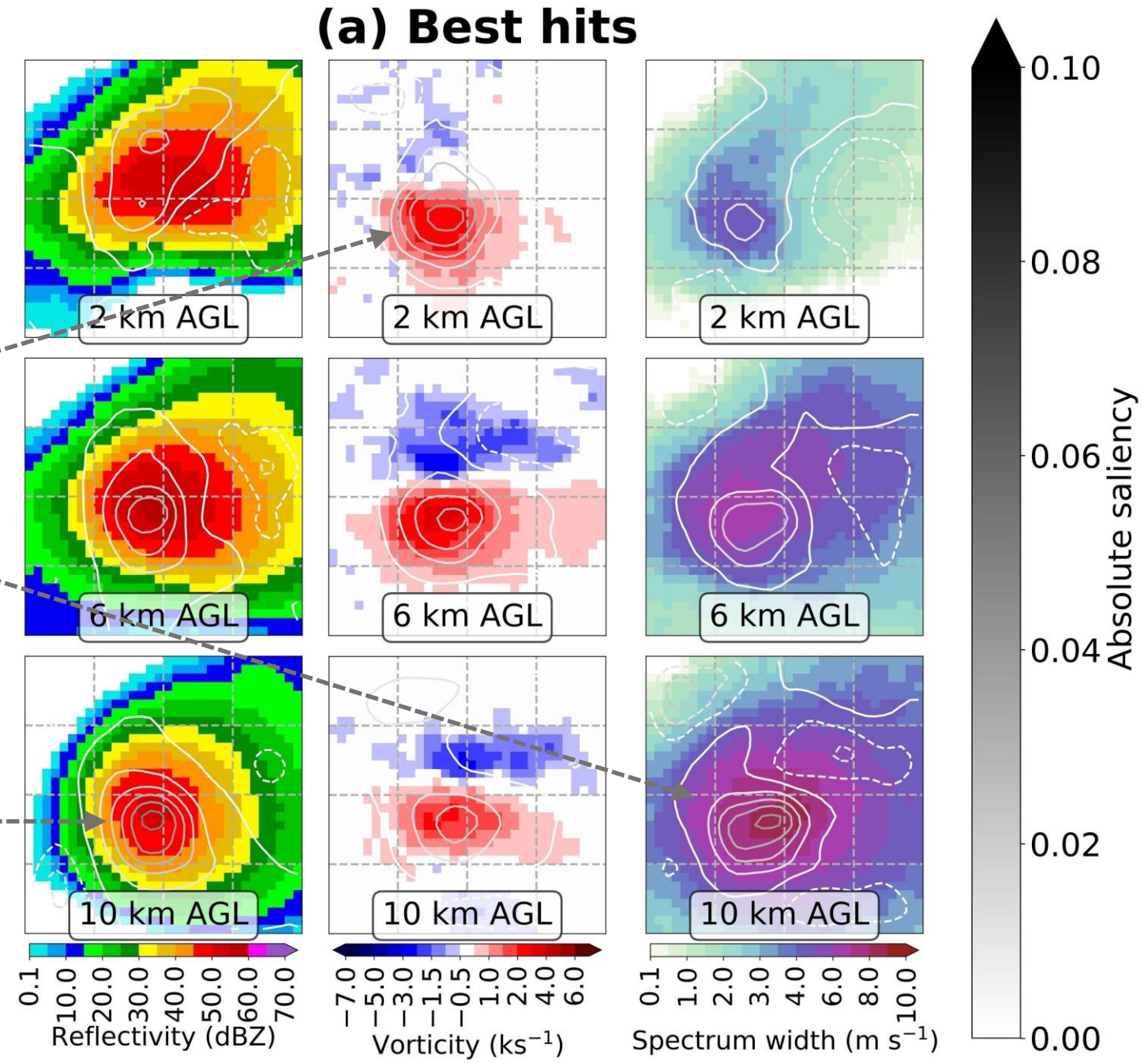
- **Right: average saliency map for best hits.**

- Solid (dashed) contours mean positive (negative) saliency.

$p_{tornado}$ increases with vorticity in mesocyclone, especially at lower levels

$p_{tornado}$ increases with spectrum width

$p_{tornado}$ increases with reflectivity in core, especially at upper levels

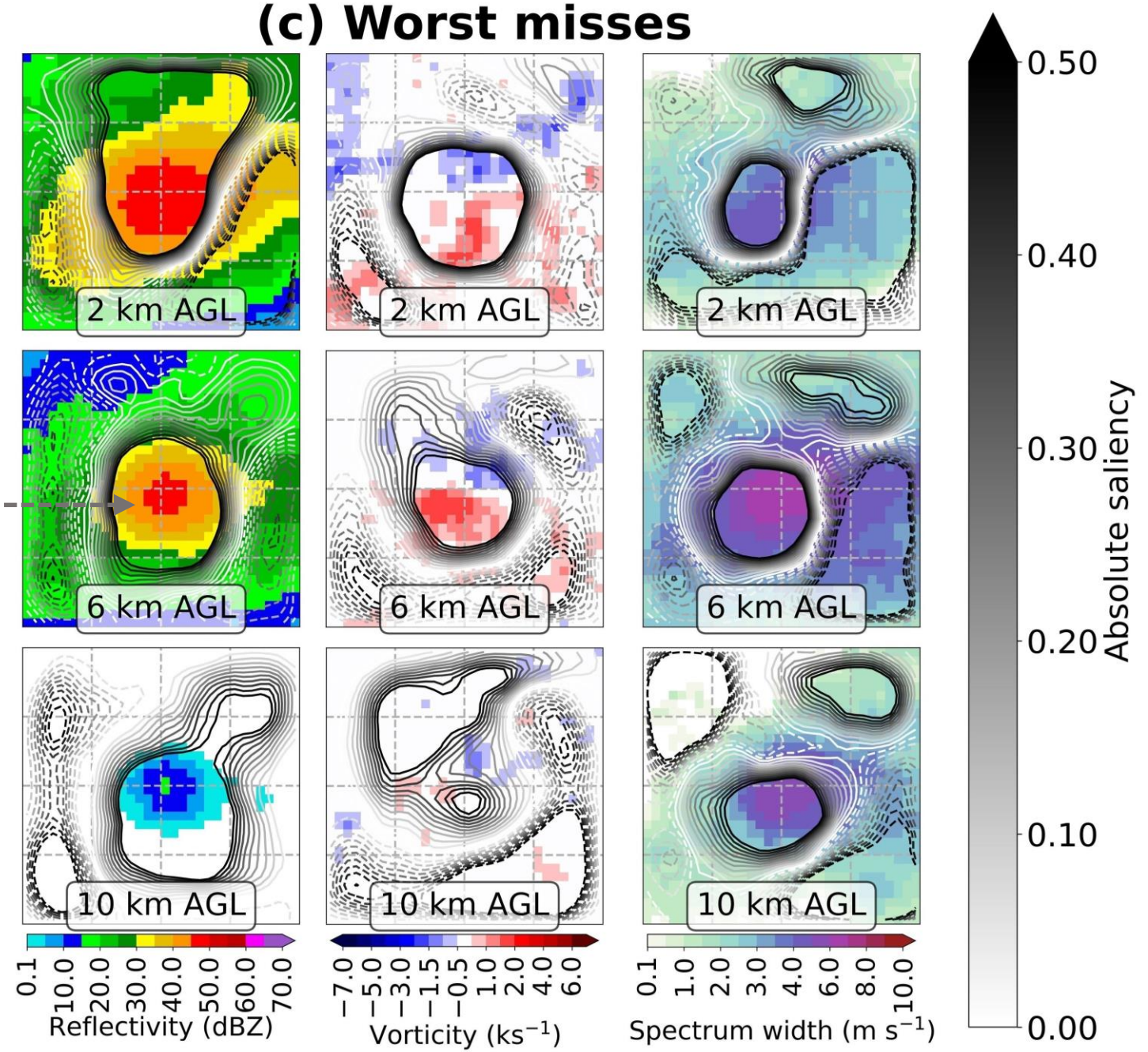


- **Right: average saliency map for worst misses.**

- Solid (dashed) contours mean positive (negative) saliency.

$p_{tornado}$ increases with all variables inside the storm, decreases with all variables around the storm

- Thus, $p_{tornado}$ increases as the storm becomes stronger and more discrete (isolated).



5. Saliency maps

- To ensure that saliency maps are not noise, we apply the "sanity checks" proposed by Adebayo *et al.* (2019):
  1. **Edge-detector test**: "can an untrained edge-detector reproduce the CNN's saliency maps?"
  2. **Model-parameter-randomization test**: "does the CNN produce similar saliency maps when weights in one layer are randomized?"
  3. **Data-randomization test**: "can a CNN trained with randomized labels produce similar saliency maps?"

- Whereas the sanity checks proposed by Adebayo *et al.* are visual, **we include a formal significance test, based on Monte Carlo resampling.**
  - Details in Section 7.1.2 of Lagerquist (2020).

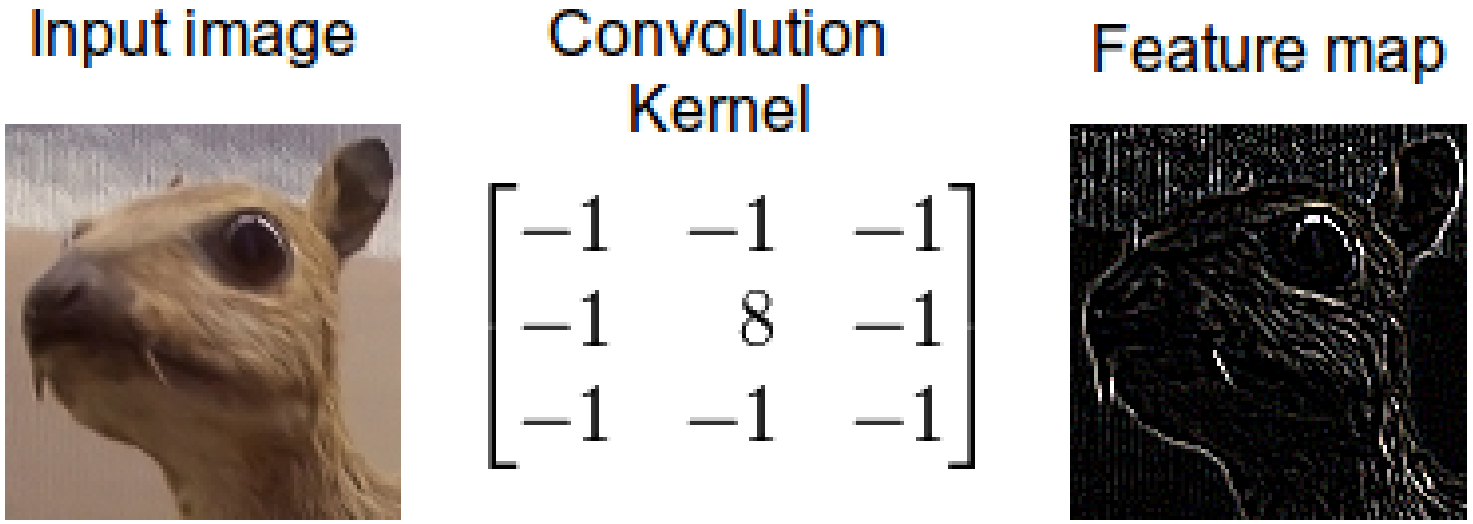
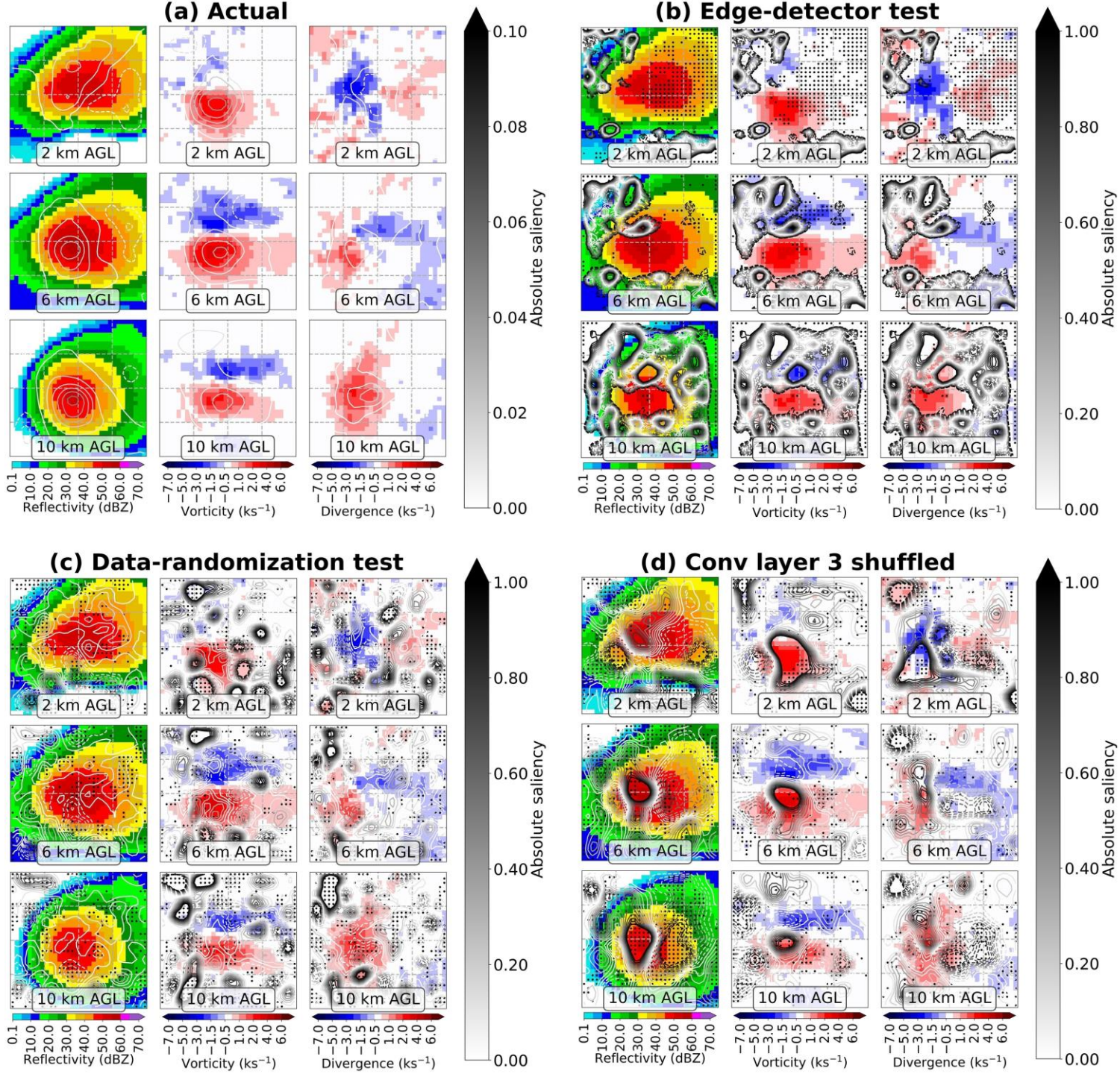
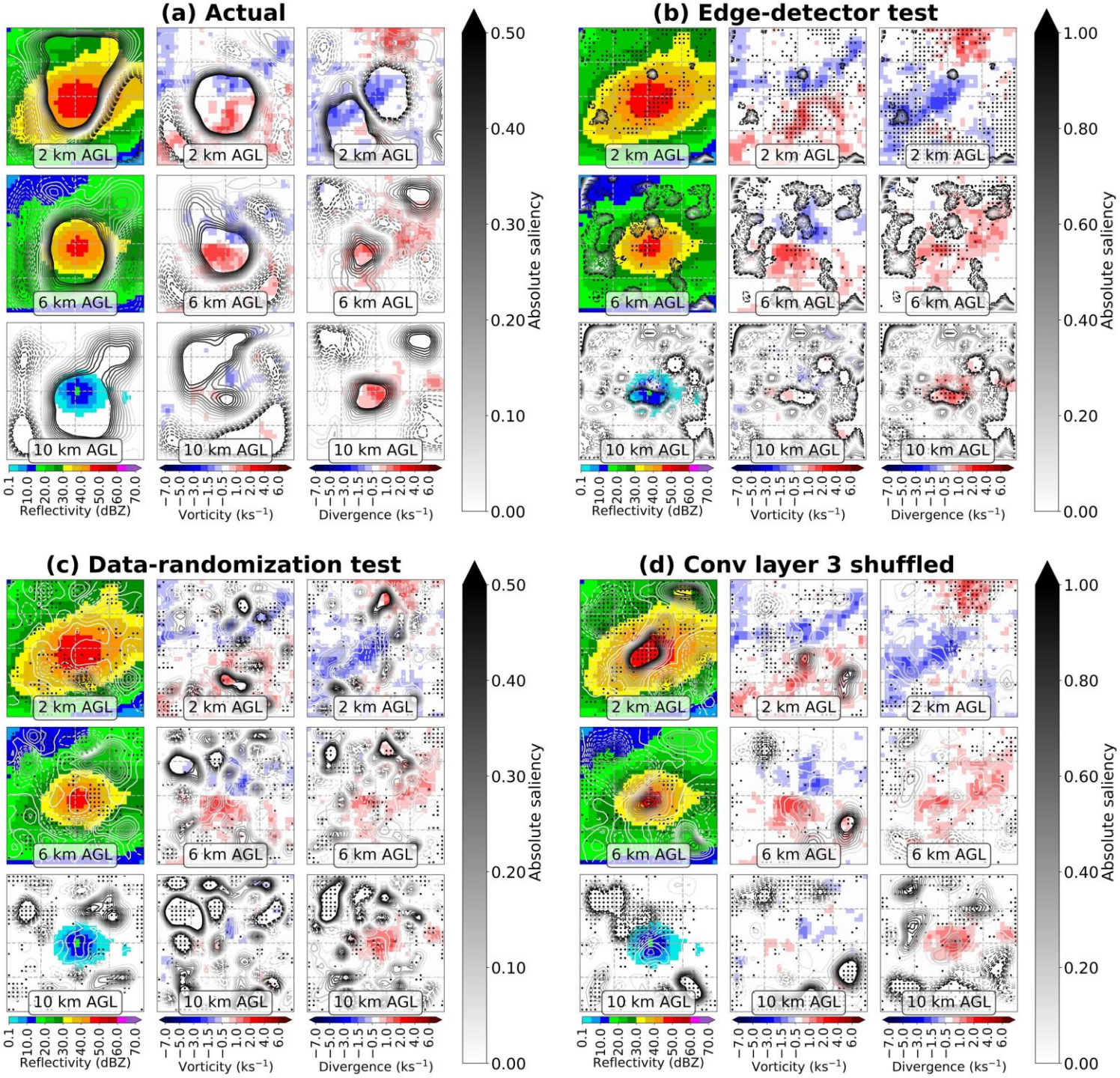


Image source: https://developer.nvidia.com/discover/convolution

- **Right: sanity checks for best hits.**
- **In panels b-d, stippling shows where difference between test and actual saliency map is significant (at 95% level).**
- Our Monte Carlo procedure accounts for false discovery due to multiple comparisons (one per grid point).
- We make the test more conservative by comparing scaled saliency values (the rank over all grid points), rather than raw saliency values.
- Percentage of significant differences: 20% in panel b, 23% in panel c, 16% in panel d.



- **Right: sanity checks for worst misses.**
- **In panels b-d, stippling shows where difference between test and actual saliency map is significant (at 95% level).**
- Percentage of significant differences: 22% in panel b, 21% in panel c, 20% in panel d.



6. Class-activation maps

- **Class activation (Zhou *et al.* 2016) is the amount of evidence for a particular class, defined at each grid point.**
- **We consider only the positive class (tornado in next hour).**
- We use Grad-CAM (Selvaraju *et al.* 2017), because the original algorithm works only for CNNs ending with a global-max-pooling layer.

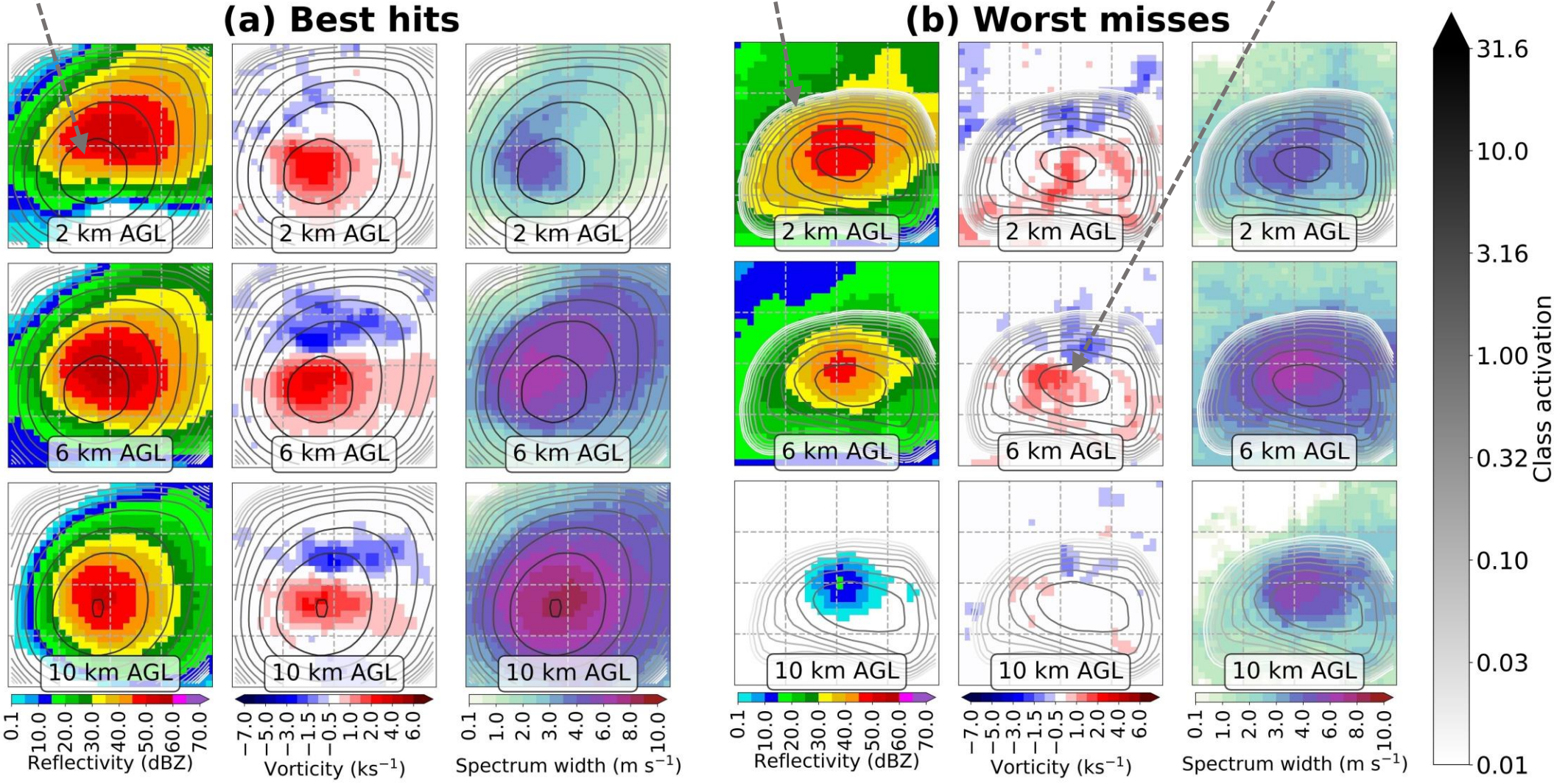
Saliency	Class activation
One value for each scalar predictor (each predictor at each grid point)	Only one value per grid point
Can be positive or negative	Cannot be negative
Highlights most important values for changing model's prediction	Highlights most important values for yielding model's actual prediction
Not layer-specific	Specific to one convolutional layer (here, always the third/deepest layer)

- **Below: average CAM for best hits and worst misses.**

Tornado evidence maxxed on right-rear flank, near hook echo and rear-flank downdraft

Area with zero evidence is the part that makes quasi-linear structure obvious

Tornado evidence maxxed with max reflectivity and vorticity



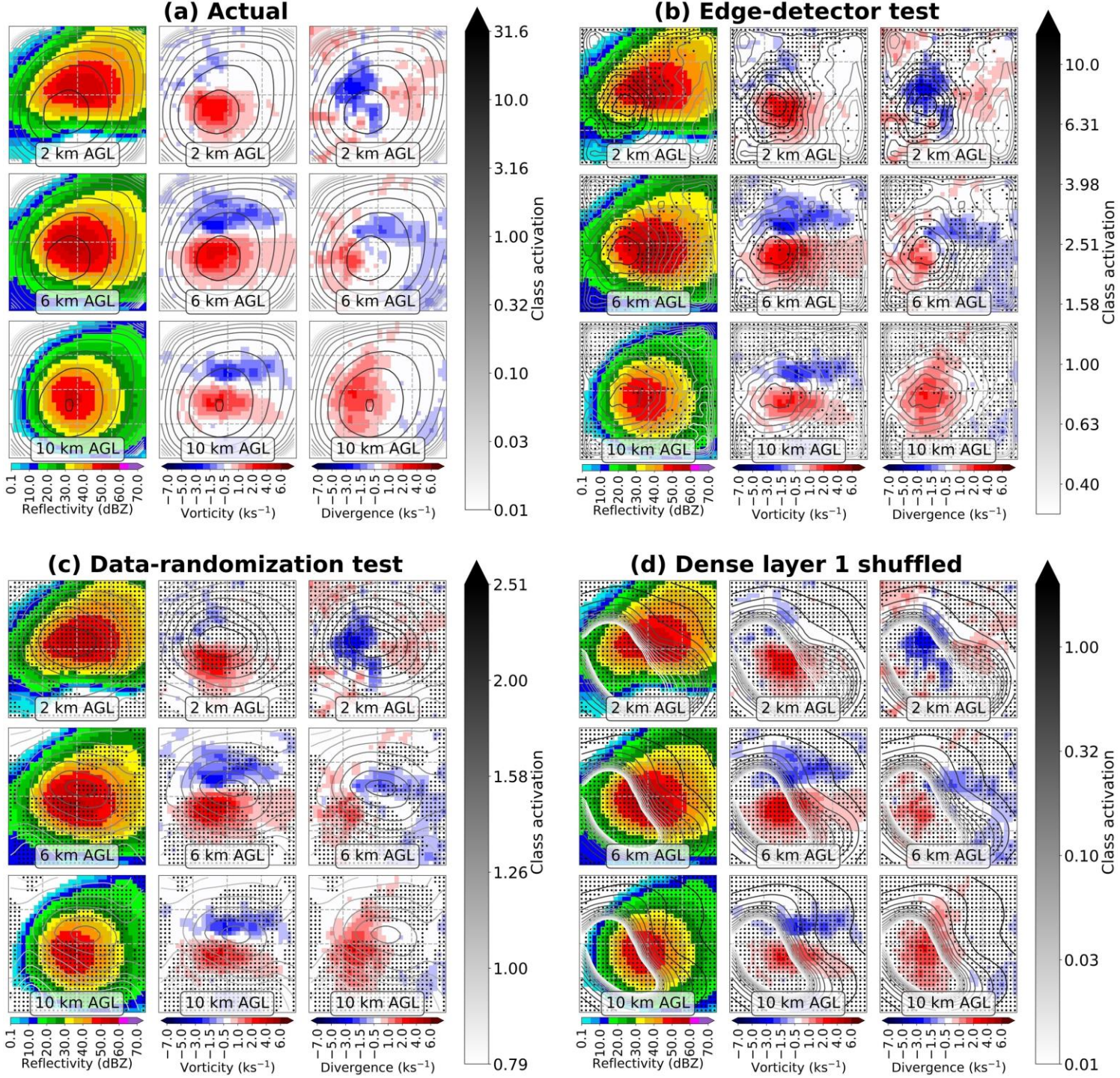
6. Class-activation maps

- **The sanity checks in Adebayo *et al.* (2019)** were originally proposed for saliency maps, but they **can be extended to class-activation maps.**

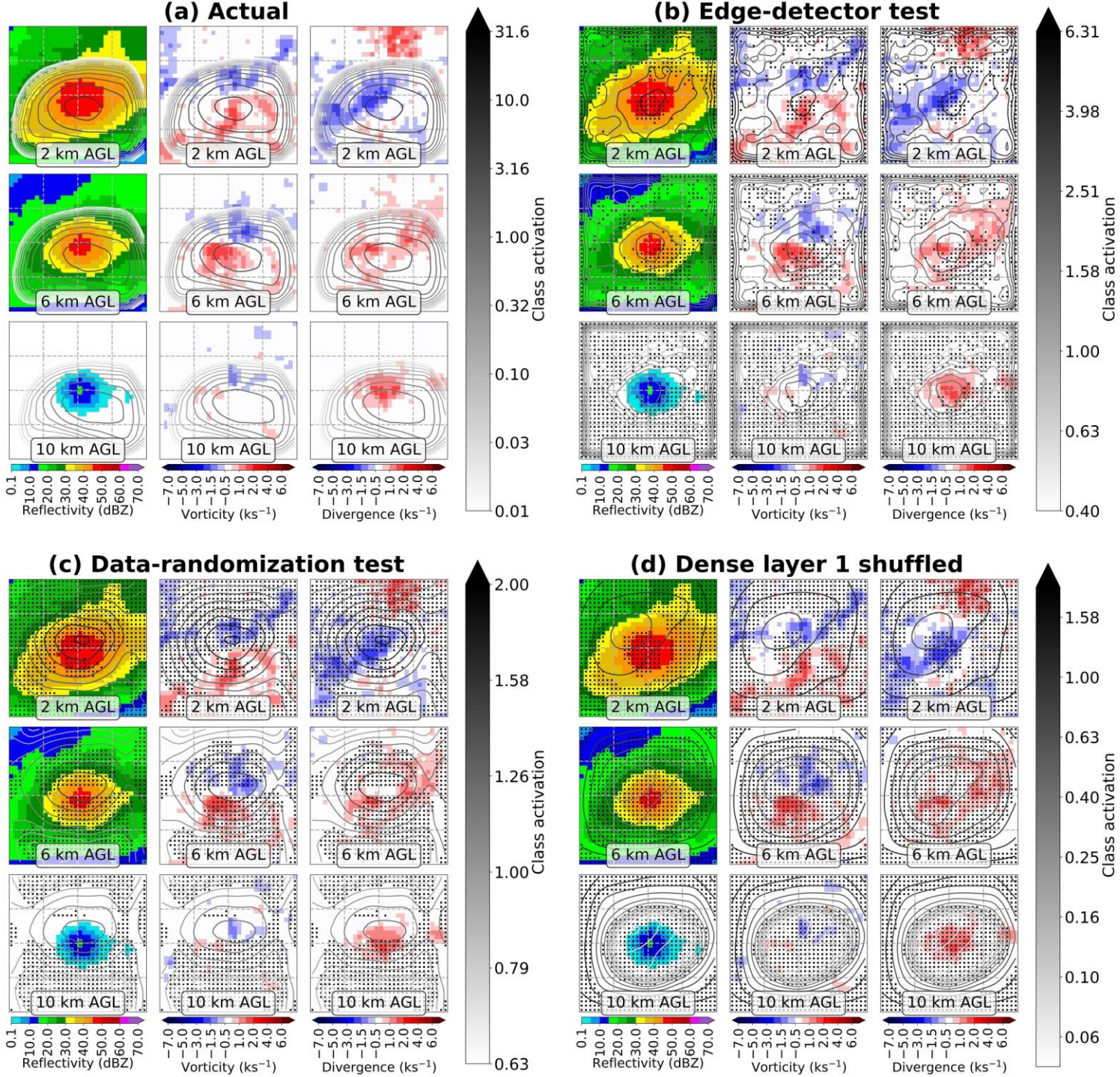
- We apply the same three sanity checks with the same Monte Carlo significance test.

- Again, we make the test more conservative by comparing scaled class activations (the rank over all grid points), rather than raw values.

- **Right: sanity checks for best hits.**
- **In panels b-d, stippling shows where difference between test and actual saliency map is significant (at 95% level).**
- Percentage of significant differences: 66% in panel b, 69% in panel c, 81% in panel d.



- **Right: sanity checks for worst misses.**
- **In panels b-d, stippling shows where difference between test and actual saliency map is significant (at 95% level).**
- Percentage of significant differences: 65% in panel b, 65% in panel c, 74% in panel d.



7. Backwards optimization

- Backwards optimization (Erhan *et al.* 2009) creates a synthetic input that minimizes or maximizes activation of the output neuron.

- **Here, backwards optimization creates a synthetic storm snapshot that minimizes or maximizes tornado probability.**

- This is done by gradient descent, the algorithm used during training to optimize CNN weights.

- Gradient descent requires a starting point ("initial seed"), for which there are many options:
  - Constant image (*e.g.,* all zeros)
  - Random image
  - Real data example (this is our choice)

7. Backwards optimization

- The default loss function for backwards optimization is $(p - p^*)^2$, where p is the current tornado probability and p^* is the desired probability.
- **However, we add some constraints to make the CNN produce more realistic storm snapshots:**

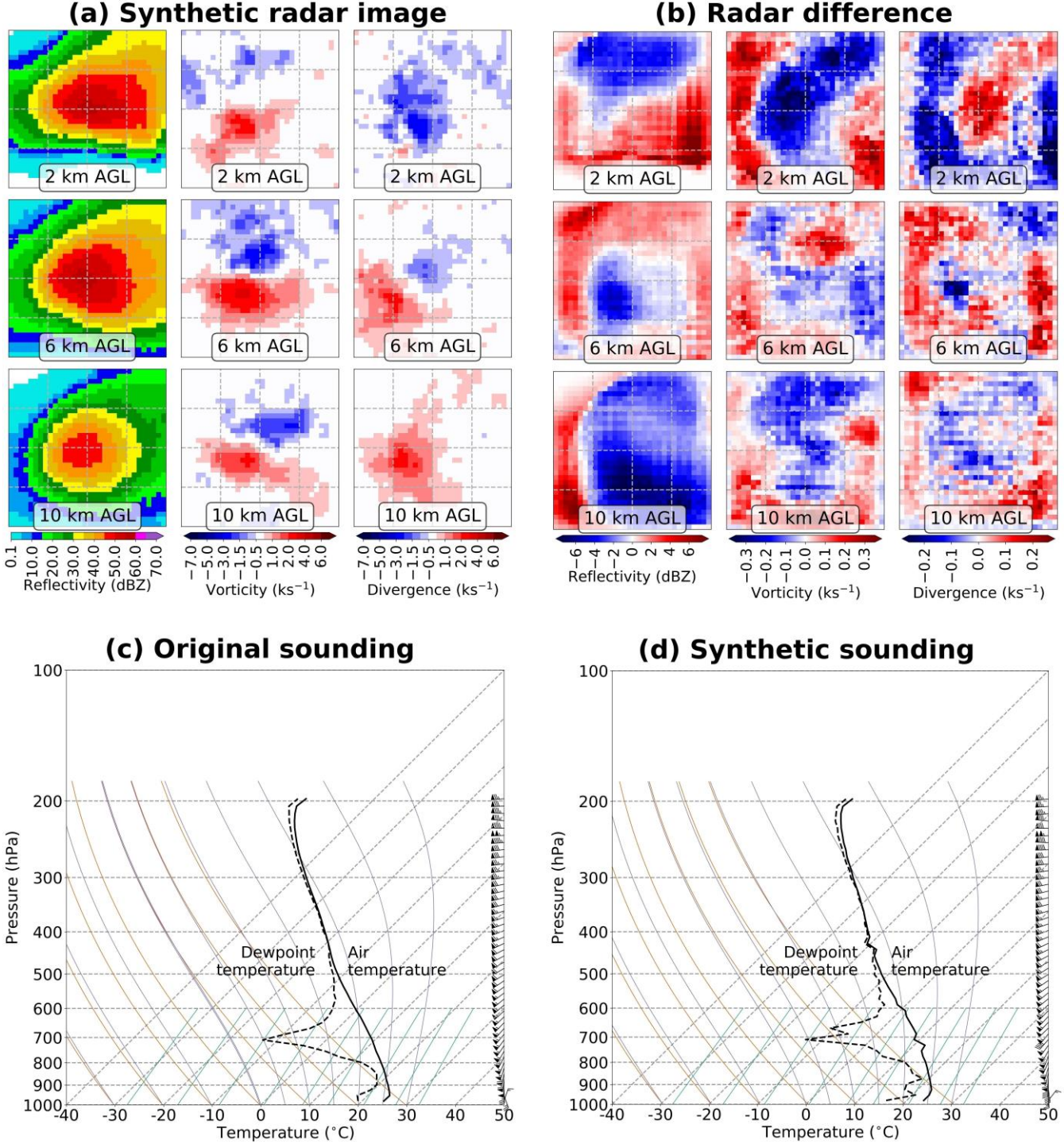
$$
\begin{aligned}
J &= (p - p^*)^2 \\
&+ \lambda_2 \|\boldsymbol{X} - \boldsymbol{X_0}\|^2 \\
&+ \lambda_{\text{minmax}} \sum_j \left\| \max\left(\boldsymbol{X_j^{min}} - \boldsymbol{X_j}, \boldsymbol{0}\right) \right\|^2 \\
&+ \lambda_{\text{minmax}} \sum_j \left\| \max\left(\boldsymbol{X_j} - \boldsymbol{X_j^{max}}, \boldsymbol{0}\right) \right\|^2
\end{aligned}
$$

- **Second term (L_2 penalty) penalizes difference between original and synthetic storm snapshot.**
- **Third term penalizes violation of minimum-constraints.**
  - Example: reflectivity cannot be < 0 dBZ
- **Fourth term penalizes violation of maximum-constraints.**
  - Example: relative humidity cannot be > 100%
- More details in Section 7.1.4 of Lagerquist (2020).

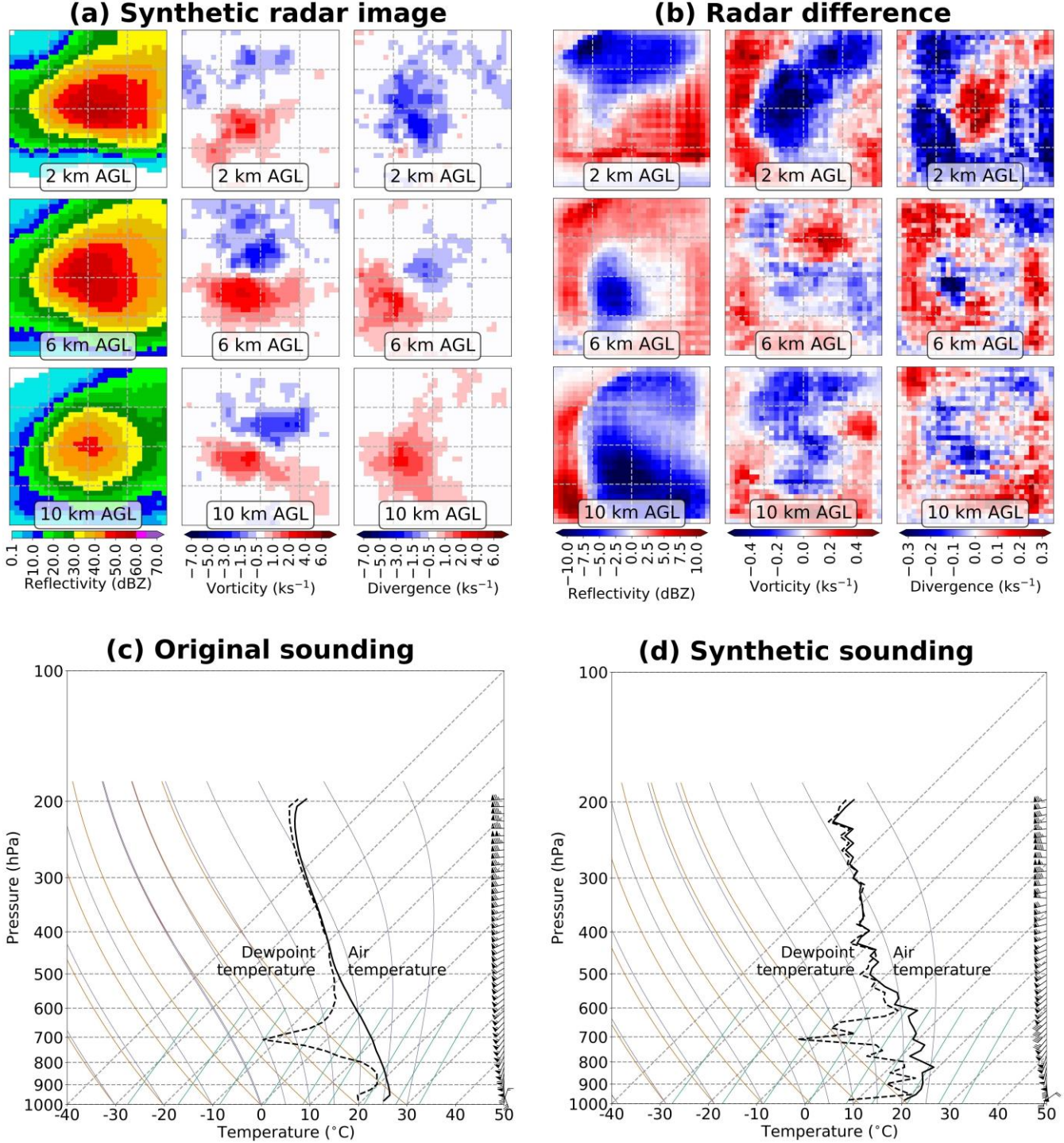
- **Right: results for best hits.**

- We use backwards optimization to alter each storm snapshot, with the goal of reducing p_{tornado} to 0.

- **Main effects:**
  - Decreases mean p_{tornado} from 99.2% to 6.9%
  - Decreases all three radar variables in the core and mesocyclone (makes storms weaker)
  - Increases all three radar variables around the edge (makes storms less discrete)
  - Decreases low-level moisture, thus decreasing CAPE for surface-based convection
  - Rotates low-level wind clockwise, thus decreasing low-level wind shear



- **Right: same but without constraints in loss function.**
- Radar image and sounding both contain a lot of high-frequency noise.
- However, the ones created with physical constraints (last slide) were also not perfect.
- The “fudge factors” used on the last slide ($\lambda_2 = 1$ and $\lambda_{\text{minmax}} = 10$) were the result of extensive experimentation, and it’s not clear that choosing different values would lead to more realistic storms.
- **In general, more work is needed on generating realistic weather data.**



8. Conclusions

- Our CNN predicts next-hour tornado probability, based on a radar image and proximity sounding.
- We used four interpretation methods to understand physical relationships learned by the CNN.
- Interpretation methods often produce noise, so we developed a formal significance test or physical constraints for each method.
- **Main findings of "augmented" interpretation methods are generally consistent with dynamical-modeling and observational studies. Examples:**
  - Most important part of sounding is low-level wind and thermal profile.
  - Most important of storm (especially for supercells) is right-rear flank, where a tornado would be expected
  - Tornadoes are more likely for discrete storms
- **To our knowledge, this is one of few studies to use formal significance tests for ML interpretation.**
- **Robust interpretation is crucial in building ML systems that are properly understood and trusted.**
- For more details, see:
  - Lagerquist (2020)
  - Upcoming BAMS paper (hopefully early 2021)

References

Adebayo, J., J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, 2018: “Sanity checks for saliency maps.” *Conference on Neural Information Processing Systems,* Montréal, Canada, Neural Information Processing Systems Foundation, http://papers.nips.cc/paper/8160-sanity-checks-for-saliency-maps.

Anderson-Frey, A., Y. Richardson, A. Dean, R. Thompson, and B. Smith, 2016: “Investigation of near-storm environments for tornado events and warnings.” *Weather and Forecasting,* **31 (6)**, 1771–1790, https://doi.org/10.1175/WAF-D-16-0046.1.

Breiman, L., 2001: "Random forests." *Machine Learning,* **45**, 5–32, https://doi.org/10.1023/A:1010933404324.

Brotzge, J., S. Nelson, R. Thompson, and B. Smith, 2013: “Tornado probability of detection and lead time as a function of convective mode and environmental parameters.” *Weather and Forecasting,* **28 (5)**, 1261–1276, https://doi.org/10.1175/WAF-D-12-00119.1.

Erhan, D., Y. Bengio, A. Courville, and P. Vincent, 2009: “Visualizing higher-layer features of a deep network.” Tech. rep., link.

Homeyer, C., and K. Bowman, 2017: “Algorithm Description Document for Version 3.1 of the Three-Dimensional Gridded NEXRAD WSR-88D Radar (GridRad) Dataset.” Tech. rep., University of Oklahoma, http://gridrad.org/pdf/GridRad-v3.1-Algorithm-Description.pdf.

Lagerquist, R., 2020: “Using deep learning to improve prediction and understanding of high-impact weather.” Doctoral dissertation, School of Meteorology, University of Oklahoma, https://shareok.org/handle/11244/324145.

Lagerquist, R., A. McGovern, C. Homeyer, D. Gagne, and T. Smith, 2020: “Deep learning on three-dimensional multiscale data for next-hour tornado prediction.” *Monthly Weather Review,* **148 (7)**, 2837-2861, https://doi.org/10.1175/MWR-D-19-0372.1.

Lakshmanan, V., C. Karstens, J. Krause, K. Elmore, A. Ryzhkov, and S. Berkseth, 2015: "Which polarimetric variables are important for weather/no-weather discrimination?" *Journal of Atmospheric and Oceanic Technology,* **32**, 1209-1223, https://doi.org/10.1175/JTECH-D-13-00205.1.

Lapuschkin, S., S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.R. Müller, 2019: “Unmasking Clever Hans predictors and assessing what machines really learn.” *Nature Communications,* **10 (1096)**, https://doi.org/10.1038/s41467-019-08987-4.

Selvaraju, R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, 2017: "Grad-CAM: Visual explanations from deep networks via gradient-based localization." *International Conference on Computer Vision,* Venice, Italy, IEEE, http://openaccess.thecvf.com/content_ICCV_2017/papers/Selvaraju_Grad-CAM_Visual_Explanations_ICCV_2017_paper.pdf.

Simonyan, K., A. Vedaldi, and A. Zisserman, 2014: “Deep inside convolutional networks: Visualizing image classification models and saliency maps.” *arXiv pre-prints,* 1312, https://arxiv.org/abs/1312.6034.

Zhou, B., A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, 2016: “Learning deep features for discriminative localization.” *Conference on Computer Vision and Pattern Recognition,* Las Vegas, Nevada, IEEE, https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Zhou_Learning_Deep_Features_CVPR_2016_paper.html.