Machine learning model interpretation and visualization focusing on meteorological domains are introduced and analyzed.

Machine learning (ML) and deep learning (DL; LeCun et al. 2015) have recently achieved breakthroughs across a variety of fields, including the world’s best Go player (Silver et al. 2016, 2017), medical diagnosis (Rakhlin et al. 2018), and galaxy classification (Dieleman et al. 2015). Simple forms of ML (e.g., linear regression) have been used in meteorology since at least the 1950s (Malone 1955), and ML has been used extensively to forecast convective hazards since the mid-1990s. Kitzmiller et al. (1995) use linear regression to forecast the probability of tornadoes, large hail, or damaging wind; Billet et al. (1997) use linear regression to forecast hail probability and size; Marzban and Stumpf (1996, 1998) use neural networks to forecast the probability of tornadoes and damaging wind, respectively; and Marzban and Witt (2001) use neural networks to forecast hail size. Gagne et al. (2013, 2017a) use random forests to forecast hail probability at 1-day lead time; McGovern et al. (2014) and Williams (2014) use random forests to forecast convectively induced aircraft turbulence; while Cintineo et al. (2014, 2018) use naïve Bayes to forecast the probability of tornadoes, large hail, and damaging wind. DL is also beginning to be used in meteorology, with applications including hail prediction (Gagne et al. 2019) and detection of extreme weather patterns such as tropical cyclones, atmospheric rivers, and synoptic-scale fronts (Liu et al. 2016; Mahesh et al. 2018; Kunkel et al. 2018; Lagerquist et al. 2019b). The authors have extensive
experience using ML to improve forecasting and understanding of weather phenomena (Gagne et al. 2017a,b; Lagerquist et al. 2017; McGovern et al. 2017; Gagne et al. 2019; Lagerquist et al. 2018). Many of these products have been used by human meteorologists in experiments and day-to-day operations.

Despite its wide adoption in meteorology, ML is often criticized by forecasters and other end users as being a “black box” because of the perceived inability to understand how ML makes its predictions. This phenomenon is not exclusive to meteorology, and many ML practitioners and users have recently begun to focus on this interpretability problem (Olah et al. 2017; Lipton 2016; NeurIPS Foundation 2018; Molnar 2018).

The main contribution of this paper is to synthesize and analyze multiple approaches to model interpretation and visualization (MIV) for meteorology. MIV is useful in all phases of ML development (Selvaraju et al. 2017). Initially, MIV can be used to aid with debugging, enabling the domain scientists and data scientists to ensure that the models are focusing on the most physically relevant aspects of the problem. During deployment, MIV can be used to help the users gain trust in the model and to identify ideal scenarios as well as shortcomings. If ML surpasses human predictions, interpretation methods could be used to improve the humans’ skills (Silver et al. 2016; Johns et al. 2015) and MIV could be used to identify new scientific hypotheses.

To ensure general results, we use both traditional ML and DL for meteorological phenomena at different spatiotemporal scales. At the synoptic scale, we apply DL to predict the probability of severe hail 24–48 h in advance across the continental United States (CONUS). At the mesoscale, we apply traditional ML to soundings from a mesoscale numerical model to predict winter precipitation type. At the storm scale, we use DL to predict the probability that a storm will produce a tornado within the next hour and traditional ML methods to classify a storm’s convective mode. Lagerquist et al. (2018, 2019a), Gagne et al. (2019), McGovern et al. (2018), and Jergensen et al. (2019) focus on the training and evaluation of these models, while we focus on MIV.

MACHINE LEARNING. We briefly review ML as needed for the MIV explanations, giving more detail on DL since it is newer to meteorology. McGovern et al. (2017) provides a more in-depth review of traditional ML methods for meteorology.

**Decision trees.** Decision trees, which can be understood as a flowchart where the decision points have been automatically learned by a computer, have been used in meteorology since the 1960s (e.g., Chisholm et al. 1968). Decision trees were built subjectively by human experts until the 1980s, when an objective learning algorithm was developed (Quinlan 1986). Their human readability has helped contribute to their popularity in many scientific domains. During training, at each branch node, the algorithm considers a number of potential questions that can split the data (e.g., is dewpoint $\geq 60^\circ$F?). Splits are chosen to minimize error on the predictions, which can be classifications, probabilities (Provost and Domingos 2003), or real valued (Breiman 1984).

Decision trees are brittle, meaning that small changes in the data can cause large changes in the final model. Ensemble approaches, such as random forests (RF; Breiman 2001) and gradient-boosted regression trees (GBRT; Friedman 2002), mitigate this problem by training ensembles of trees but this minimizes human readability. In RF, diversity is maintained by training each tree with a different subset of examples. In GBRT, the $k$th tree is fit to the error of the first $k - 1$ trees, rather than being fit to the target value. RF and GBRT are used successfully in meteorology (Williams et al. 2008a,b; Gagne et al. 2009; McGovern et al. 2014; Williams 2014; McGovern et al. 2015; Clark et al. 2015; Elmore and Grams 2016; Lagerquist et al. 2017).

**Support-vector machines.** Support-vector machines (SVM; Vapnik 1963) find a hyperplane in predictor space that can be used to linearly separate the data. SVMs can also be used for nonbinary classification (Franc and Hlavac 2002) or regression (Drucker et al. 1997). SVMs often use a kernel to transform the predictor space into another space where the problem is more easily separated. One such kernel, the radial basis function (RBF; Schölkopf et al. 1997), uses a Gaussian to transform the space. Linear SVMs, like decision trees, are human readable when there are few predictors, but high-dimensional linear SVMs and nonlinear SVMs are not easily interpreted by a human. Recent SVM applications in meteorology include Radhika and Shashi (2009), Rao et al. (2012), and Rajasekhar and Rajinikanth (2014).

**Deep learning.** DL is a subset of ML that specializes in leveraging spatiotemporal structure in the input data. DL is well suited for meteorology, because it can be applied directly to spatiotemporal grids and identify salient features at different spatiotemporal scales. The DL models we use here are convolutional neural networks (CNN; Fukushima and Miyake 1972) and recurrent neural networks (RNN; Williams 1989).
1982). The main components of a CNN are convolutional, pooling, and dense layers (Fig. 1). Each convolutional layer passes many convolutional filters over the input maps, where an input map consists of multiple “channels” (e.g., red, green, and blue channels for an image), creating one output map for each filter. The output maps (feature maps) are passed through a nonlinear activation function, such as the rectified linear unit (ReLU; Nair and Hinton 2010). The activation function must be nonlinear; otherwise, the net can learn only linear relationships. Supplemental Fig. ES1 is an animation of one filter.

During training, weights in the convolutional layers are updated via stochastic gradient descent (SGD; section 4.4.3 of Mitchell 1997) to minimize the loss function. The typical loss function for classification, adopted in this work, is cross entropy, defined as

$$-\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \left( \hat{y}_{ik} \right),$$

where $N$ is the number of examples, $K$ is the number of classes, $y_{ik}$ is the true value (1 if the $i$th example belongs to the $k$th class, 0 otherwise), and $\hat{y}_{ik}$ is the predicted probability that the $i$th example belongs to the $k$th class. Cross entropy varies from $[0, \infty)$, and the optimal score is zero.

Pooling layers reduce the resolution of feature maps, which allows deeper convolutional layers (further to the right in Fig. 1) to learn larger-scale features along with some amount of translational invariance. Deeper layers learn higher-level abstractions, because the feature maps are larger in scale and have passed
through more nonlinear transformations (section 5.2.1 of Chollet 2018). Supplemental Fig. ES2 is an animation of maximum pooling.

Most CNNs end with one or more dense layers (section 3.1.1 of Chollet 2018; our Fig. 1), which are identical to hidden layers in a traditional neural net (Haykin 2001). Weights in the dense layers are learned via SGD, simultaneously with those in the convolutional layers. For binary classification, the final prediction is one value (probability of event). For K-class problems (K > 2), the final prediction consists of K values (one probability for each class).

For traditional ML methods presented in this work, we use scikit-learn (Pedregosa et al. 2011). For DL, we use Keras (Chollet 2015). All models are trained on the training data; hyperparameters (parameters not adjusted during training) are selected by which model performs best on the validation data; and final results are reported on the testing data.

**INTERPRETATION AND VISUALIZATION METHODS FOR TRADITIONAL MACHINE LEARNING.** The most general classes of MIV methods are filter and wrapper methods (Kohavi and John 1997). Filter methods consider only the data themselves, whereas wrapper methods incorporate (wrap around) the model. We focus on wrapper methods, as the goal is to understand what the ML models have learned from the data. This section describes interpretation methods designed for traditional ML, though all but impurity importance can be generalized to DL.

**Impurity importance.** Impurity importance (Louppe et al. 2013; Breiman 2001) is one of the most popular importance methods because it can be computed during training and is implemented in scikit-learn. However, this method works only for tree-based models. It is animated in supplemental Fig. ES3 and defined as follows:

$$I(p) = \frac{1}{T} \sum_{t=1}^{T} \sum_{s \in S_p} \left( \frac{N_s}{N} \right) \Delta i(s),$$

where $T$ is the number of trees in the ensemble, $S_p$ is the set of all splits in tree $t$ that involve predictor $p$, $\Delta i(s)$ is the decrease in the impurity score (e.g., Gini or entropy) achieved by split $s$ on the training data, $N$ is the total number of training examples, $N_s$ is the number of training examples passed to split $s$, and $I(p)$ is the resulting importance for predictor $p$. The most important predictors are those that affect the most examples (higher in tree) and that split the data more effectively (decrease impurity more).

**Permutation importance.** The permutation method is another approach for ranking predictor importance. Although initially proposed by Breiman (2001) for RFs, it can be used for any traditional or DL model. There are two variations of permutation importance that we denote as single pass (Breiman 2001) and multipass (Lakshmanan et al. 2015). In both variations, the goal is to determine how much performance deteriorates when the statistical link between a predictor $x_j$ and the target variable is broken. This is achieved by randomly permuting $x_j$ over all examples, then comparing the performance of the trained model on unpermuted data to performance on the permuted data. If performance deteriorates significantly when $x_j$ is permuted, this indicates that $x_j$ is important. If performance does not deteriorate significantly, either 1) $x_j$ is generally unimportant or 2) information in $x_j$ is redundant with information contained in the other predictors. For example, if there are two predictors $x_j$ and $x_k$, such that $x_k = 2x_j$, they are fully linearly dependent (Pearson correlation = 1.0), so either predictor on its own could be deemed unimportant. Multipass importance aims to address this issue. The single-pass and multipass algorithms are precisely stated in supplemental Fig. ES4. Supplemental Figs. ES5 and ES6 describe the algorithms with animations. Both versions are implemented with publicly available code in Jergensen (2019).

**Sequential (forward and backward) selection.** Sequential selection is another method for ranking predictor importance, but unlike impurity and permutation importance, it can be used to explicitly add or remove predictors from the model. The algorithms for sequential forward and backward selection (SFS and SBS) are shown in supplemental Fig. ES7 and animated in supplemental Figs. ES8 and ES9. The two algorithms are very similar: SFS begins with a climatological model (one that always predicts the mean target value in the training data), containing zero predictors, and adds one predictor at a time. Conversely, SBS begins with a model trained on all predictors and removes one at a time.

SFS may be likened to the song “99 Bottles of Pop on the Wall,” where each pop bottle is a predictor. At the $k$th iteration of SFS, $k-1$ predictors (bottles) have been selected (taken off the wall and put on the table). The goal is to find the one remaining predictor that, when added to those already selected, yields the best $k$-predictor model. The stopping criterion in the algorithm (whether cost function $J$ decreases significantly) is purposely vague, as there are many ways to implement this. Basically, $J$ should decrease enough to
justify adding another predictor to the model, which increases model complexity.

SBS can be thought of as SFS in reverse. At the kth iteration, k–1 predictors (bottles) have been removed from the model (put back on the wall). The goal is to find the worst remaining predictor (i.e., the one whose removal from the model yields the smallest increase in J). Again, the stopping criterion is purposely vague. Basically, J should increase enough to justify no longer removing the worst predictor from the model.

Generalized versions of SFS and SBS have been proposed (e.g., Stracuzzi and Utgoff 2004; chapter 9 of Webb 2003), which allow both forward and backward steps in the same algorithm or removing several predictors at each step. Also, genetic algorithms can be used to iteratively improve (evolve) the set of selected predictors (e.g., Siedlecki and Sklansky 1993; Leardi 1996). Last, a method called “sufficient input subsets” (Carter et al. 2018) can be used to find the most relevant predictors for one or a subset of examples, without retraining the model.

Partial-dependence plots. While the above methods can reveal the most important predictors for a given problem, they do not indicate how or why each predictor is important. One way to address this would be to visualize the average prediction for each possible value of $x_j$, the predictor of interest. However, this does not account for nonlinear interactions between $x_j$ and the other predictors. Partial-dependence plots (PDP; Friedman 2001) address this problem by fixing the value of one or more predictors $X^*$ for all examples, passing these new data through the trained model, and averaging the resulting predictions. This averages out the effects of the other predictors. To make the full PDP, the entire process is repeated for a range of values for $X^*$. Regions of the PDP with nonzero slope indicate where the ML model is sensitive to $X^*$. Note that this method provides one plot for each predictor, which can be overwhelming with hundreds of predictors.

Related MIV methods attempt to explain the model’s prediction for an individual example. Individual conditional expectation (ICE; Goldstein et al. 2015) is the PDP for a specific example. The ICE plot (which can be shown on the same axes as the PDP) identifies clusters of model behavior, which are regions of the predictor space where the model treats examples similarly. Another method is locally interpretable model-agnostic explanation (LIME; Ribeiro et al. 2016), which fits a simple model, such as linear regression, to a set of slightly perturbed examples. The perturbed examples are similar to the example of interest but with slightly altered predictor values. Predictor weights in the simple altered model are used to explain the prediction.

INTERPRETATION METHODS FOR DEEP LEARNING. All methods presented in the previous section, with the exception of impurity importance, can be adapted for DL. In this work, we demonstrate how the permutation method can be adapted for DL. Although SFS and SBS can also be adapted, they require significant additional computation for the retraining of each model, making them infeasible for most DL applications. For permutation, images from one channel are permed over all examples. Thus, each example consists of a set of spatially intact maps, but the maps are temporally shuffled (e.g., the temperature field from 1 January 1970 may be matched with the humidity field from 5 April 2012). All interpretation methods presented in this section are implemented in Lagerquist and Gagne (2019); saliency maps are also implemented in Lagerquist (2018). We discuss primarily CNNs in this section, because this is the DL model used in our results. However, the methods presented herein can be applied to other DL models, such as convolutional long-short-term memory (LSTM) and recurrent neural nets.

Saliency maps. Saliency maps (Simonyan et al. 2014) quantify the influence of each input value (i.e., each predictor at each grid point) on the activation of some part of the CNN. This could be the activation of a particular neuron, a group of neurons, or the final prediction from the CNN. Most often it is with respect to an output neuron, whose activation is a predicted probability, p. The saliency of predictor $x$ at grid point $(i,j,k)$, with respect to prediction $p$, is $\delta p / \delta x(i,j,k)$. Thus, positive (negative) saliency means that the prediction increases (decreases) as $x(i,j,k)$ changes.

One advantage of saliency maps is that they share the dimensions of the input data, which allows them to be viewed as images (the way meteorologists generally prefer to query data) and overlaid with the input data. One disadvantage is that saliency does not necessarily imply importance: salient values are those with which the prediction changes most dramatically, but they are not necessarily most important for the original prediction (Samek et al. 2017). This disadvantage is alleviated by methods such as layerwise relevance propagation (Samek et al. 2017; Montavon et al. 2018) and class-activation maps (“Gradient-weighted class-activation maps” section). Another disadvantage is that saliency is a linear approximation around the actual value of $x(i,j,k)$, meaning saliency indicates...
how the model reacts when $x$ is perturbed slightly from the actual value, but not when it is perturbed drastically.

**Gradient-weighted class-activation maps.** Class-activation maps (CAM; Zhou et al. 2016) quantify the influence of each grid point, rather than each predictor at each grid point, on the predicted probability of a given class $p_k$. However, CAM works only on a specific type of CNN architecture, so we use a generalization, gradient-weighted class-activation maps (Grad-CAM; Selvaraju et al. 2017). Grad-CAM quantifies the influence of each grid point on $p_k$, filtered through a given convolutional layer in the network. In other words, at a given depth in the network, Grad-CAM indicates which spatial locations support the prediction of the $k$th class. For deeper convolutional layers, the class-activation map tends to be smoother (with less small-scale variation) and more localized (with nonzero values in a smaller part of the physical space), reflecting the tendency for deeper layers to learn higher-level abstractions. The ability to leverage representations at different layers is an advantage of Grad-CAM over saliency maps.

**Backward optimization.** Backward optimization (BWO; or “feature optimization” in Olah et al. (2017)) creates a synthetic input example that maximizes the activation of particular neuron(s), using SGD (“Machine learning” section). Whereas SGD is used during training to update the network weights in a way that minimizes the loss function, it is used during BWO to update input values in a way that maximizes the activation of the given neuron(s). For example, if the task is tornado prediction and we choose to maximize the activation of the output neuron, BWO will create an “optimal tornadic storm.” Conversely, if we choose to minimize the activation, BWO will create an “optimal nontornadic storm.” Supplemental Fig. ES10 shows an animation of backward optimization, where the goal is to decrease tornado probability for a tornadic storm that initially had very high forecast probability.

Because SGD only adjusts the values in an array, rather than creating the array from scratch, it requires a starting point or “initial seed.” Some options are all zeros, Gaussian noise, or a real-dataset example. The advantage of all zeros and Gaussian noise is that the initial seed almost never resembles a real example, so the synthetic example created by BWO is more novel with respect to the initial seed. The advantage of real-data initialization is that the output of BWO is usually more physically realistic. Another way to make the output more physically realistic is to integrate BWO with a generative model (e.g., Goodfellow et al. 2014), which learns to create novel but representative dataset examples [Montavon et al. (2018), who use the term “activation maximization” instead of BWO]. The sensitivity of BWO to the initial seed can be seen as a disadvantage since it does not yield one “perfect” answer, but it can also be seen as an advantage, since BWO can be run with many initial seeds to obtain different answers.

**Novelty detection.** Novelty detection (Wagstaff and Lee 2018) finds the most novel, or unexpected, image $X^*$ in a set of images (trial set) with respect to all images in another set (baseline set), then quantifies the novelty of each value in $X^*$ (i.e., each predictor at each grid point). The algorithm is detailed in supplemental Fig. ES11. As an example of its use, the baseline set could contain nontornadic storms, while the trial set contains tornadic storms. In this case, novelty detection would quantify which tornadic storms are most novel, and which parts of these storms are most novel, with respect to the nontornadic ones. The algorithm involves “features,” which are inputs to the first dense layer of the CNN (Fig. 1). It is crucial to remember that the CNN extracts only features that aid in the prediction task, so the results of novelty detection are always with respect to the prediction task.

Novelty detection works by using singular-value decomposition (SVD; section 9.3.5 of Wilks 2006) to create a lower-dimensional representation of the image data (feature vector) and an up-convolutional network (upconvnet; Dosovitskiy and Brox 2016) to transform the feature vector back to image space. Loosely, an upconvnet is a backward CNN. The upconvnet allows novelty to be viewed in image space, which is easier for humans to interpret than feature space. The main outputs of novelty detection are the reconstructed image from the CNN’s feature space, the reconstructed image from an SVD approximation to the same feature space, and the difference between the two.

**Meteorological domains.** We briefly summarize each domain (prediction task) for which the MIV methods are used. Full descriptions are found in Lagerquist et al. (2018, 2019a), Gagne et al. (2019), McGovern et al. (2018), and Jergensen et al. (2019). We chose deliberately to present results on a wide variety of meteorological domains to show the wide applicability of the MIV methods. This section is kept brief, as specific details are not needed to understand the results fully. Rather, we wanted to highlight the broad applicability of the results.
We use traditional ML methods (RF, GBRT, and SVMs) to classify storms into three categories: supercell, part of a quasi-linear convective system (QLCS), and discrete storms. The categories and the human-labeled data both come from Thompson et al. (2012) and Smith et al. (2012). We use data from the years 2003–11. Predictors for this task include radar statistics derived from the Multi-Year Reanalysis of Remotely Sensed Storms (MYRORSS; Ortega et al. 2012) and environmental data from the Rapid Update Cycle (RUC; Benjamin et al. 2004). Models are trained with ninefold cross validation, where each fold is 1 year.

**Precipitation type.** We apply the same traditional ML methods for predicting winter precipitation type. The four precipitation types are rain, freezing rain, snow, and ice pellets. Labels are Meteorological Phenomena Identification Near the Ground (mPING; Elmore et al. 2014) reports from October 2014 to March 2015. The predictors are statistics derived from RUC proximity soundings. Specifically, soundings are taken from the nearest grid point to the mPING report at 6-, 12-, and 18-h lead times. Although soundings from the three lead times are not independent, they increase the size and variability of the dataset, which is crucial given that the time period is only one winter. The three forecasts have the same valid time, so differences among them are due solely to differences among the three model runs. The results in this paper come from training on the classic warm-nose sounding, characterized by an elevated warm (melting) layer above a cold (freezing) layer at the surface. This type of sounding contains two freezing layers, one above the elevated warm layer and one below, and is the type most commonly associated with freezing rain and ice pellets.

**Tornado prediction.** We use a CNN to forecast tornadogenesis. Specifically, for each storm object (one thunderstorm cell at one time), the CNN is applied to a storm-centered radar image and proximity sounding, with the goal of predicting whether or not the storm will generate a tornado in the next hour. Radar images come from the GridRad dataset (Homeyer and Bowman 2017), a mosaic of all Next Generation Weather Radar (NEXRAD) scans in the CONUS. The GridRad data used here have a horizontal resolution of 0.02°, vertical resolution of 0.5 km up to 7 km above sea level and 1.0 km aloft, and temporal resolution of 5 min. Storm-centered 2D grids (e.g., Fig. 9) are 32 × 32, interpolated to 1.5-km horizontal resolution, and rotated so that storm motion is toward the right. The composites contain 12 variables: minimum, mean, and maximum reflectivity from 1 to 3 km above ground level; minimum, mean, and maximum 1–3-km radial velocity spectrum width; minimum, mean, and maximum 2–4-km vorticity; and minimum, mean, and maximum 5–8-km vorticity. The choice of these variables is based on previous work by Sandmael and Homeyer (2018), showing that they discriminate well between tornadic and severe nontornadic storms.

Soundings come from the RUC model before 1 May 2012 and the Rapid Refresh (RAP; Benjamin et al. 2016) otherwise. In general, interpretation results are shown only for radar data, as results for soundings have been noisy. Tornado reports from Storm Data (National Weather Service 2016) are used to determine when/if a storm undergoes tornadogenesis.

**Hail prediction.** We use CNNs to predict large hail in simulated thunderstorms (Gagne et al. 2019) from the National Center for Atmospheric Research (NCAR) convection-allowing ensemble (CAE; Schwartz et al. 2015). The target variable is based on the storm’s maximum future hail size (“yes” if ≥25 mm in diameter, “no” otherwise), according to the Thompson microphysics scheme (Thompson et al. 2004, 2008). This is a perfect-model experiment, because the target variable comes from a simulation rather than true observations. The goal is to identify storm-scale and environmental features that promote simulated hail growth.

The predictors are storm-centered grids of five variables (temperature, dewpoint, u wind, v wind, and geopotential height) at three pressure levels (850, 700, and 500 hPa). The grids are 32 × 32 and share the 3-km grid spacing of the NCAR CAE. Each CNN trained for this problem has three strided-convolution layers (which combine the convolution and pooling operations into one layer), with either ReLU or leaky-ReLU activation (Maas et al. 2013), followed by a dense layer with sigmoid activation (cf. Fig. 1).

**RESULTS.** We organize results by MIV method, beginning with traditional ML and moving to DL.

**Ranking and selecting important predictors.** Figure 2 compares importance rankings for tree-based models, using both impurity and permutation importance for the tasks of storm-mode classification and winter...
precipitation. It is important to understand both the meteorological significance of the predictors and the difference in importance rankings among the methods. This is especially important for researchers who may have previously used only one importance ranking.

For winter precipitation type, the two versions of permutation agree on the top four predictors (with a difference in ordering for the second and third): surface wet-bulb temperature, mean $u$ wind in the lowest cold layer (LCL), $u$ wind difference between the LCL and elevated warm layer (EWL), and mean...
u wind from the EWL to 5 km above ground level (AGL). The importance of these predictors makes sense meteorologically. For example, (isobaric) wet-bulb temperature is the temperature that an air parcel would be if it were cooled by evaporating water into it at constant pressure. If the surface temperature is >273.15 K, freezing rain is impossible and rain becomes much more likely. There are more substantial differences between impurity importance and permutation importance. For example, the top predictor (surface wet-bulb temperature) using permutation is fifth-most important according to impurity, while the most important predictor for impurity (mean u wind in the EWL) is fifth- or seventh-most important according to permutation.

For storm-mode classification, the two versions of permutation importance agree on three of the top four predictors (with a slight difference in ordering): the y component of lifting condensation level-to-equilibrium level (LCL–EL) wind shear, perimeter, and compactness. In general, the most important predictors are reflectivity statistics (spatial statistics based only on grid cells inside the storm), environmental wind shear (in the proximity sounding), and shape parameters. These results are broadly consistent with what we know about storm mode, especially the difference between supercells and other modes: supercells tend to be less elongated (lower eccentricity) than QLCS storms, with higher reflectivity and higher wind shear. Again, the two versions of permutation agree more with each other than with impurity importance. However, impurity importance still emphasizes shape parameters, reflectivity, environmental wind shear, and low-level shear. This last variable is radar-derived azimuthal shear from 0 to 2 km above ground level and is greater in supercells, due to rotation in the mesocyclone.

Permutation can also be used to rank predictors for non-tree-based models, as shown in Fig. 3. For
winter precipitation, the two versions agree on three of the top four (and five of six) predictors for SVMs: mean $u$ wind from the EWL to 5 km AGL, mean $v$ wind in the EWL, and $v$-wind difference between the two cold layers (below and above the EWL). However, these are mostly disparate from the top predictors in Fig. 2, for the RF. This suggests that the RF and SVM “care about” different predictors. This is expected, since the two models internally are very different. According to the error bars in Figs. 2 and 3, the performance of the two models is statistically similar, either before permutation or after any number of permutations. Thus, there is no reason to give more credence to the permutation-importance results of one ML model over the other. This underscores the importance of using several ranking methods and considering general types of predictors (e.g., sounding and shape statistics) rather than just single predictors. The different rankings would also be easily explainable if there were many linearly dependent predictors. However, the predictors for this task were preprocessed to remove any absolute Pearson correlation > 0.7 in the data.

For tornadogenesis (Fig. 3), the most important predictor [ranked by area under the receiver operator curve (AUC); Metz 1978] is $u$ wind (meridional wind in the proximity sounding). The third- to fifth-most important predictors in the single-pass method (maximum low-level reflectivity, midlevel vorticity, and low-level vorticity) match the second- to fourth-most important predictors, respectively, in the multipass method. Perhaps the most striking difference is that RH (relative humidity in the sounding) is ranked second by the single-pass method but seventh by the multipass method. This suggests that in the multipass method, after the first predictor ($u$ wind) has been

**Fig. 4.** Sequential-selection results for winter precipitation type for (top) a random forest and (bottom) a radial-basis-function SVM. (left) For SFS, the first predictor selected is at the top and the last selected is at the bottom. (right) For SBS, the first predictor removed is at the top and the last removed is at the bottom.
permuted, RH is no longer
the most important predic-
tor. This is counterintuitive,
as RH is less dependent on
wind (Pearson correlation
of 0.11) than maximum
low-level reflectivity on
wind (0.43). However, Pear-
son correlation is linear
and based on the entire
dataset. It is possible that,
for a small partition of the
dataset with a strong ef-
fect on AUC, there is more
dependence between RH
and wind, causing the
two variables to be more
redundant than indicated
by the Pearson correlation.

In the single-pass method, after the top few predictors,
there is very little difference in importance among the
rest. This is because the other 16 predictors, which
contain the vast majority of useful information, are
still intact. Conversely, in the multipass method, AUC
decreases substantially with each successive predictor
permuted until it reaches ~0.5 after the ninth predictor
(minimum low-level vorticity), which is the AUC for
a completely random model. Overall, both methods
suggest that reflectivity and vorticity are the most
important radar predictors, while wind and relative
humidity are the most important sounding predictors.

SFS and SBS are used to explicitly select predictors
to keep in an ML model. Figure 4 shows results for
winter precipitation type, based on both the RF and
SVM models. Results for storm mode are not shown,
because SBS would take hundreds of hours with the
hundreds of predictors used for this task. For the RF,
while forward and backward selection do not agree
precisely, their ranking of predictors is similar. For
the SVM, forward and backward selection agree on
only two of the top five (and five of the top eight) pre-
dictors. Regularization of the SVM may affect these
results, something that requires further investigation.
RFs perform predictor selection internally (by choos-
ing the best predictor at each split point; “Machine
learning” section), but SVMs do not: if it receives 17
predictors it must fit a 17-dimensional hyperplane,
which can lead to unstable results (Vapnik 1995).

Partial-dependence plots. A partial-dependence plot for
precipitation type for the most important predictors
in the RF is shown in Fig. 5. The easiest curves to in-
terpret are those for rain and freezing rain. As surface
wet-bulb temperature increases from approximately
267 to 273 K, rain probability increases sharply while
freezing-rain probability decreases sharply. Freezing
rain is impossible when surface temperature exceeds
273.15 K, because it requires that rain fall as liquid and
freeze upon contact with the surface. The fact that the
RF and SVM create freezing rain at $T_w$ well below
freezing and freezing rain at $T_w$ well above freezing is
a particular characteristic of the NWP model errors.

Saliency maps. Saliency maps for tornado prediction
are shown in Fig. 6. For the sake of brevity, these
maps include only four of the 12 radar variables list-
ed in the “Tornado prediction” section. Each row is a
composite over 100 examples (one example indicates
one storm at one time) in the validation period: the
best true positives (tornadic examples with the high-
est forecast probabilities), worst false alarms (non-
tornadic examples with the highest probabilities),
worst misses, and best correct nulls. Composites
are created with the method of probability-matched
means (PMM; Ebert 2001). We initially tried using
simple means, but the resulting composites were
unrealistic due to spatial offsets among storms in
the composite. We have examined interpretation
outputs for individual storms (e.g., supplemental
Fig. ES10), and the results are conceptually similar
to the composites, so we are not overly concerned
about artifacts introduced by PMM. In the future
we will add local PMM (Clark 2017).

For the best true positives, the composite radar im-
age looks like a supercell (e.g., Kumjian and Ryzhkov
2008), with a large core of reflectivity > 55 dBZ; a slight
hook echo on the right flank (more visible in the top-left

![Precipitation Type Partial Dependence Plots](image)

**Fig. 5.** Partial-dependence plots for the probability of each precipitation
type, given two predictors in the random forest. Blue histograms show the
distribution of each predictor in the training data.
panel of Fig. 9); and large maxima of low-level reflectivity, low-level spectrum width, low-level vorticity, and midlevel vorticity in the mesocyclone. According to the composite saliency map, tornado probability increases strongly with all four of these maxima and decreases strongly with reflectivity behind the storm, especially near the mesocyclone. This latter relationship suggests that tornadoes are more likely when the storm is more isolated from surrounding deep convection and the rear-flank downdraft is not too cold (due to evaporative cooling; e.g., Markowski et al. 2002), concepts familiar to human meteorologists.

Adebayo et al. (2018) discuss three “sanity checks,” which ensure that saliency maps reflect meaningful
relationships learned by the model, rather than patterns that exist in all data, such as the “Buell patterns” that often appear in principal-component analysis (Richman 1986). We have implemented the edge-detector test and supplemental Fig. ES12 shows that the “saliency maps” produced by an untrained edge detector are markedly different than those produced by the trained model.

For the worst false alarms, the composite radar image is very similar to the best hits, except that there is no discernible hook echo, a smaller reflectivity core, and smaller maxima of all four variables. The composite saliency map is also similar to the best hits, which indicates that if these maxima were increased to their levels in the best hits, tornado probability would increase strongly. For the worst misses, the composite radar field looks very different than the best hits and worst false alarms. The reflectivity core has a more linear structure, which suggests that many of the 100 storms are part of a QLCS. This makes sense, given that QLCS tornadoes are often missed by human forecasters (Table 2; Brotzge et al. 2013). Minimum low-level spectrum width is near zero throughout most of the domain, possibly because these storms tend to be more elevated (with bases higher aloft than the best hits and worst false alarms). Also, maxima of the two vorticity fields are only 0.5–1.0 ks⁻¹, about 10 times smaller than for the best hits or worst false alarms. Tornado probability increases strongly with reflectivity, slightly with spectrum width, and moderately with vorticity, in the reflectivity core. Tornado probability also decreases strongly with reflectivity behind the core and moderately with reflectivity in front of the core, indicating a preference for isolated convection. For the best correct nulls, the composite radar image and saliency map look roughly similar to the worst misses. The main differences are that the reflectivity core is weaker and the storm is more elongated in the direction of motion.

For the hail-prediction task, composite saliency maps for selected CNN neurons are shown in Fig. 7 (inset). The top neuron is activated by supercell-like...
storms, with a rounded shape and rotational wind field. The middle neuron is activated by bow-echo-like storms, with an elongated shape, strong low-level convergence, and little rotation. The bottom neuron is activated by pulse-type storms, with neither an exceptionally rounded nor exceptionally elongated shape and an outflow-dominant low-level wind field. The geographic map in Fig. 7 shows the spatial

![Composite Grad-CAM](https://via.placeholder.com/150)

**Fig. 8.** Composite Grad-CAM (gradient-weighted class-activation maps) for the 100 best hits, 100 worst false alarms, and 100 worst misses, according to different convolution layers. Storm motion is to the right. Layer 1 is the shallowest, and layer 4 is the deepest (Fig. 1). Heat maps represent input fields (predictors), as in Fig. 6, while line contours represent class activation. Darker colors indicate that the underlying spatial location has a greater positive influence on “yes” (tornado) predictions. Negative influences cannot be shown with Grad-CAM. For the 100 worst misses, output is shown for layer 3 rather than layer 4, because class-activation maps produced by layer 4 are all zeros.
distributions of storms that strongly activate the three neurons.

**Class-activation maps.** CAMs for tornado prediction, produced by Grad-CAM, are shown in Fig. 8. These maps show the most important grid cells for tornado prediction—that is, those that most strongly support a “yes” forecast. The first CAM for the best true positives, produced by the first (shallowest) convolution layer, suggests that the most important locations are in the mesocyclone, collocated with the slight hook echo and maxima in spectrum width and vorticity. Outside of this region, class activation decreases sharply. Contours are elongated along the right flank, indicating that class activation decreases less sharply along the right flank than perpendicular to it. This makes sense, as the right flank is adjacent to the storm’s inflow environment, to which tornadogenesis is highly sensitive [e.g., review in first paragraph of Wade et al. (2018)]. The second CAM for the best hits, produced by the fourth (deepest) convolution layer, is similar to the first CAM, except that contours are smoother and nonzero activations are more confined to the mesocyclone. This makes sense, given that (i) inputs to the fourth convolution layer have coarser resolution (3 vs 1.5 km) and (ii) deeper layers learn higher-level abstractions, increasing their ability to selectively focus on a small part of the image (e.g., the

---

**Fig. 9.** Composite backward-optimization results for the 100 best hits. Storm motion is to the right. In this case the goal of backward optimization was to minimize tornado probability. (top) Input fields (before optimization); (middle) output fields (after optimization), and (bottom) the increments made by backward optimization (after minus before).
main vorticity maximum in the reflectivity core and not the secondary one in the forward flank). There is a slight offset between class-activation maxima for the first and fourth convolution layers, which is probably due to upsampling from dimensions of $10 \times 10$ to $32 \times 32$.

The CAM for the worst false alarms is similar to the best hits, with two main exceptions. First, nonzero activations cover a smaller area, consistent with the reflectivity core covering a smaller area. Second, contours are elongated in the direction of storm motion, rather than along the right flank. This indicates that the right flank is less supportive of “yes” forecasts in the worst false alarms than in the best hits, which makes sense, as vorticity maxima for the best hits extend farther along the right flank. Finally, the CAM for the worst misses is produced for the third (second deepest) convolution layer (Fig. 1). We show the third convolution layer, instead of the fourth, because CAMs for the fourth layer are all zeros. This makes sense, given that forecast probabilities for the worst misses (by definition) are very low. According to the CNN, the radar image as a whole does not support a “yes” forecast. Progressing deeper in the network, from the first to fourth convolution layer (not shown), the area of nonzero class activations is pushed away from the center until all activations become zero.

**Backward optimization.** BWO results for tornado prediction are shown in Figs. 9 and 10. In Fig. 9, BWO is used to adjust each of the 100 best true positives (as the seed) with the goal of decreasing tornado probability. BWO decreases the CNN’s forecast probability from near one to near zero. Conversely, in Fig. 10,
BWO is used to adjust the worst misses (as the seed), with the goal of increasing tornado probability. Here, BWO increases the probability from near zero to near one. In general, BWO has the greatest effect on low-level reflectivity and spectrum width (making changes up to 40 dBZ and 4 m s\(^{-1}\), respectively), with a much subtler effect on the vorticity fields (making changes of \(~10^{-3}\) s\(^{-1}\)). However, as shown in the reflectivity fields, BWO does not necessarily produce realistic output. It is possible that more realistic output could be encouraged by adding physical constraints to the loss function, as is sometimes done in objective analysis [e.g., the geostrophic constraint used in Panofsky (1949), Bergthórsson and Döös (1955), and Cressman (1959)].

BWO for hail prediction is shown in Fig. 11. The initial seed (an array of all zeros) has been adjusted by the CNN to maximize large-hail probability. The output (synthetic storm) includes a positive height anomaly at 850 hPa, with negative height anomalies at 700 and 500 hPa. This height-anomaly gradient is associated with a high lapse rate (strong increase of temperature with pressure), which can lead to more instability and a stronger updraft. The synthetic storm also includes positive temperature and dewpoint anomalies at 850 and 700 hPa, along with confluent winds, indicating that warm and moist air is flowing into the storm. Finally, winds in the inflow region (bottom of the map) rotate clockwise with height, which is favorable for right-moving supercells (Bunkers et al. 2000).

**Novelty detection.** Novelty detection for tornado prediction is shown in Fig. 12. Each row is a PMM composite over the 100 most novel examples for which the storm undergoes tornadogenesis in the next hour in the validation period. Both the actual feature vector and its SVD reconstruction are projected to image space by the upconvnet, which allows the input (radar image) and output (novelty map) to be viewed in the same space. The most novel or unexpected parts of the examples shown are low reflectivity to the storm’s right, which indicates a lack of deep convection in the inflow environment; high spectrum width in the mesocyclone and reflectivity core; high low- and midlevel vorticity to the storm’s right; and low midlevel vorticity on the left side of the reflectivity core.

Although the upconvnet cannot exactly map storms from feature space to image space, it can highlight novel or interesting areas of the input for further examination. One must be careful to ensure that artifacts of the upconvnet, such as the positive low-level vorticity anomalies to the storm’s right and in the forward flank, are recognized as such. The novel regions can be used for knowledge discovery and further hypothesis testing.

**DISCUSSION AND FUTURE WORK.** This paper synthesizes and analyzes ML MIV methods and demonstrates their use for various meteorological domains. Table 1 provides a high-level summary, listing the advantages and disadvantages of each method and Table 2 summarizes when a user should choose each method. As ML continues to gain popularity in meteorology and other physical sciences, it is crucial for practitioners to understand the trade-offs inherent in the models themselves and the MIV methods used to explain them. It is also important to understand the computational trade-offs of these methods. Some
methods are efficient, while some may take additional supercomputing time, meaning that users need to decide if additional computational effort is worth the potential insights gained.

For example, if the user wants to identify the most important predictors for the ML model, the most computationally efficient approach is impurity importance. However, impurity importance has a
<table>
<thead>
<tr>
<th>Method</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Impurity importance</strong> (<em>“Impurity importance” section</em>)</td>
<td>Succinct list of predictors  Computed at training time</td>
<td>Does not explicitly select predictors (underlying tree does selection) Works only for tree-based methods</td>
</tr>
<tr>
<td>Permutation importance (<em>“Permutation importance” section</em>)</td>
<td>Succinct Model agnostic (can be applied to any ML model) Single-pass method can be easily parallelized</td>
<td>Greedy algorithm that chooses one predictor at each step, which limits the search of solution space and also make it difficult to compare across runs if the underlying ML model is brittle Multipass is computationally expensive Does not explain why something is important</td>
</tr>
<tr>
<td>Sequential selection (<em>“Sequential (forward and backward) selection” section</em>)</td>
<td>Succinct Model agnostic Selects most relevant predictors Generalized versions allow algorithm to be somewhat nongreedy</td>
<td>Computationally inefficient (model is retrained many times) Usually implemented as a greedy algorithm Does not explain why something is important</td>
</tr>
<tr>
<td>Partial-dependence plots (<em>“Partial-dependence plots” section in “Interpretation and visualization methods for traditional machine learning” section</em>)</td>
<td>Model agnostic Explains how predictor x is important: i.e., how output changes with x over range of x</td>
<td>Difficult to extend to deep learning Inefficient for multivariate interactions Potentially overwhelming for the human when used for large numbers of predictors</td>
</tr>
<tr>
<td>Saliency maps (<em>“Saliency maps” section in “Interpretation methods for deep learning” section</em>)</td>
<td>Results can be presented in image space (often easier for humans to examine) Example-by-example and multiexample explanations: meaningful results can be presented for single examples; can also be a disadvantage but can be alleviated by compositing Explains how neuron activation changes with each input value (i.e., each predictor at each grid point) Can be used for neurons, channels, or other groupings of neurons</td>
<td>Differentiable models only Simulates only slight change to input data (linear approximation to derivative)</td>
</tr>
<tr>
<td>Grad-CAM (<em>“Gradient-weighted class-activation maps” section</em>)</td>
<td>Results can be presented in image space Example-by-example and multiexample Identifies important locations in the images</td>
<td>Deep learning only Does not explain how values at these locations influence neuron activation</td>
</tr>
<tr>
<td>Backward optimization (<em>“Backward optimization” section in “Interpretation methods for deep learning” section</em>)</td>
<td>Results can be presented in image space Example-by-example and multiexample Extends saliency maps (uses derivatives to optimize image for desired neuron activation) Shows model behavior for extreme case</td>
<td>Deep learning and differentiable models only Can produce physically unrealistic output Answer depends heavily on initial seed</td>
</tr>
<tr>
<td>Novelty detection (<em>“Novelty detection” section in “Interpretation methods for deep learning” section</em>)</td>
<td>Results can be presented in image space Example-by-example and multiexample Finds interesting images and/or image subsets for further analysis</td>
<td>Deep learning only Depends on upconvnet, which can be difficult to train</td>
</tr>
</tbody>
</table>

Major disadvantage: it can be used only for tree-based models. Permutation is more general (can be applied to traditional and DL models) but more computationally expensive, especially for the multipass version. Sequential selection is also general but even more computationally expensive, as it requires retraining the model potentially thousands or millions of times. Also, none of these methods explain how or why a
predictor is important, nor do they differentiate between situations where the predictor is important and is not. This question can be answered by partial-dependence plots and some of the DL-based methods.

DL-based interpretation methods can identify important spatial locations and spatial multivariate patterns, create synthetic data that minimize or maximize a certain prediction, identify novel examples in the dataset, and identify the novel parts of each example. Most DL-based methods can be applied to different parts of the model, for example, a neuron in any layer, a group of neurons in any layer, or the final prediction, which allows these methods to explain what the model “sees” at different depths (e.g., Fig. 8). Compared to the permutation test and sequential selection, most of the DL-based methods discussed are computationally efficient. This is primarily because these methods do not involve retraining the DL model. However, training the DL model in the first place can be computationally expensive.

It is crucial to understand the trade-offs between predictability and interpretability. Since ML models are not inherently modeling a physical problem, they may find solutions with better predictive skill at the cost of a less interpretable model. For example, different ML models inherently learn different types of solutions. If one method has better predictive skill and chooses a different set of important predictors, this does not necessarily mean that the other predictors are physically unimportant. This is a common pitfall in recent MIV papers in physical science, and we caution new users to understand the limitations of and differences among MIV methods before making physical conclusions.

One aspect of the problem not discussed in this paper is formal hypothesis testing. To conclude that ML has confirmed existing knowledge or discovered something new, would require robust hypothesis testing. We are currently determining the best ways to incorporate this into our work.

In addition to explaining the behavior of the model, one potential use of MIV in meteorology is to identify new hypotheses for scientists to explore. This potential is becoming more attractive as datasets grow and prediction tasks become harder, while ML accordingly becomes more sophisticated and more integrated into our workflow. Since ML can process data quickly, it could be used to “flag” interesting data (e.g., patterns or interactions among predictors) for further analysis. This has already been done with novelty detection, which is used to flag images taken by the Mars rovers as targets for future exploration by the rovers (Wagstaff and Lee 2018). In meteorology, such methods could be used, for example, to identify observations that need to be collected more often or processes that need to be better resolved in physical models. This would allow for data science to feed back on and enrich physical science.

ACKNOWLEDGMENTS. This material is based upon work supported by the National Science Foundation under Grant EAGER AGS 1802627. Funding was also provided by

| **Table 2. Mapping MIV methods to tasks.** |
|---------------------------------|----------------------------------|
| **Method**                      | **Tasks**                        |
| Impurity importance             | Quick computation of predictor importance for tree-based methods |
| Permutation importance          | Ranks and quantifies importance of each predictor; works for any model (not only trees) |
| Sequential selection            | Identify a minimal set of predictors to build a model |
| Partial-dependence plots        | Identify sensitivity of a model to a predictor over its full range |
| Saliency maps                   | Visualize local gradient of predictions with respect to the predictors in the input space |
| Grad-CAM                        | Visualize most important spatial regions of the predictor space; deep learning only |
| Backward optimization           | Create synthetic examples that activate the model in a certain way (e.g., minimize or maximize prediction); deep learning only |
| Novelty detection               | Identify novel/unexpected examples and what region of each example makes it novel; deep learning only |
REFERENCES


Eltmore, K., and H. Grams, 2016: Using mPING data to generate random forests for precipitation type


Franc, V., and V. Hlavac, 2002: Multi-class support vector machine. Int. Conf. on Pattern Recognition, Quebec City, QC, Canada, IEEE.


——, ——, and M. Xue, 2013: Severe hail prediction within a spatiotemporal relational data mining framework. Int. Conf. on Data Mining, Dallas, TX, IEEE, https://doi.org/10.1109/ICDMW.2013.121.


Johns, E., O. Aodha, and G. Brostow, 2015: Becoming the expert: Interactive multi-class machine teaching. Conf. on Computer Vision and Pattern Recognition, Boston, MA, IEEE.


Attention AMS Student Members

Stay connected to AMS after graduation for half the regular membership rate

AMS
American Meteorological Society

Let AMS help you build your expertise, your network, your career. There’s never been a more important time to be a member.

http://www.ametsoc.org/earlycareer