BAMS Article

# Evaluation, Tuning, and Interpretation of Neural Networks for Working with Images in Meteorological Applications

Imme Ebert-Uphoff and Kyle Hilburn

**ABSTRACT:** The method of neural networks (aka deep learning) has opened up many new opportunities to utilize remotely sensed images in meteorology. Common applications include image classification, e.g., to determine whether an image contains a tropical cyclone, and image-to-image translation, e.g., to emulate radar imagery for satellites that only have passive channels. However, there are yet many open questions regarding the use of neural networks for working with meteorological images, such as best practices for evaluation, tuning, and interpretation. This article highlights several strategies and practical considerations for neural network development that have not yet received much attention in the meteorological community, such as the concept of receptive fields, underutilized meteorological performance measures, and methods for neural network interpretation, such as synthetic experiments and layer-wise relevance propagation. We also consider the process of neural network interpretation as a whole, recognizing it as an iterative meteorologist-driven discovery process that builds on experimental design and hypothesis generation and testing. Finally, while most work on neural network interpretation in meteorology has so far focused on networks for image classification tasks, we expand the focus to also include networks for image-to-image translation.

**KEYWORDS:** Satellite observations; Radars/Radar observations; Artificial intelligence; Machine learning; Neural networks; Deep learning

https://doi.org/10.1175/BAMS-D-20-0097.1
Corresponding author: Imme Ebert-Uphoff, iebert@colostate.edu
In final form 18 August 2020

**AFFILIATIONS: Ebert-Uphoff**—Electrical and Computer Engineering, and Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, Colorado; **Hilburn**—Cooperative Institute for Research in the Atmosphere, Colorado State University, Fort Collins, Colorado

Neural networks (NNs) are increasingly emerging as useful tools for meteorological applications (Boukabara et al. 2019; Reichstein et al. 2019; Lee et al. 2018). However, because of their novelty many questions are yet to be answered for their use, such as the following:

- Evaluation and tuning: Which performance measures are most useful to evaluate and tune a neural network in a meteorological context?
- Interpretability: If an NN is performing well, *how* does it generate meteorologically meaningful results? Can we discover the strategies it uses?

We consider both types of questions here with focus on NNs for working with meteorological imagery. We build on the work of others, most notably of McGovern et al. (2019) and Toms et al. (2020), and also of Lagerquist et al. (2020), McGovern and Lagerquist (2020), and Gagne et al. (2019, 2015).

Developing a better understanding of NN models can benefit meteorological applications in the following ways: (i) identify and eliminate potential failure modes, and, more generally, identify ways to improve the model's overall performance; (ii) improve trust in the model; (iii) gain scientific insights that might lead to the development of simpler, more transparent approaches for such tasks; and (iv) to learn new physics (Toms et al. 2020; Barnes et al. 2019, 2020).

In this paper we seek to extend existing work by the following contributions:

- recognizing the process of NN interpretation for meteorological applications as an iterative discovery process that uses many different tools, and emphasizing the importance of the meteorologist at every step of the process;
- enumerating many available tools for NN evaluation and interpretation;
- expanding the discussion to image-to-image translation networks (such as encoder–decoder networks or U-nets), which have received little attention in the context of interpretation;
- highlighting two important NN concepts that have received little attention in meteorology, namely, receptive fields and layer-wise relevance propagation.

A key lesson we learned in the course of this research was that gaining insights into the neural network consisted of an iterative, meteorologist-driven discovery process, driven by traditional methods of experimental design, and hypothesis generation and testing, with NN visualization tools simply providing additional tools to assist this process.

The remainder of this article is organized as follows. Table 1 provides a list of acronyms. The second section discusses NN basics with emphasis on NNs for working with images. The third section introduces the much overlooked concept of receptive fields. The fourth section highlights strategies for NN evaluation and tuning. The fifth section presents NN interpretation via targeted experiments. The sixth section discusses selected NN visualization methods, and the seventh section presents conclusions.

**Table 1. List of acronyms.**

| Acronym | Expansion | Section/reference |
|---|---|---|
| ANN or NN | Artificial neural network | "From simple neurons to powerful neural networks" section |
| ReLu | Rectified linear unit (sample activation function) | "From simple neurons to powerful neural networks" section |
| D layer | Dense layer (NN layer with trainable weights) | "Layer types for working with images" section |
| C layer | Convolution layer (NN layer with trainable weights) | "Layer types for working with images" section |
| P layer | Pooling layer (fixed NN layer) | "Layer types for working with images" section |
| U layer | Upsampling layer (fixed NN layer) | "Layer types for working with images" section |
| CNN | Convolutional neural network (specific neural network type) | "NN architectures for working with images" section |
| FCN | Fully convolutional neural network (specific neural network type) | "NN architectures for working with images" section |
| GAN | Generative adversarial network (specific neural network type) | "NN architectures for working with images" section |
| GOES | Geostationary Operational Environmental Satellite | Schmit et al. (2017) |
| ABI | Advanced Baseline Imager (primary instrument on GOES) | Schmit et al. (2017) |
| GOES-GLM | GOES Geostationary Lightning Mapper (single channel of GOES) | Goodman et al. (2013) |
| MRMS | Multi-Radar Multi-Sensor product | Smith et al. (2016) |
| GREMLIN | GOES Radar Estimation via Machine Learning to Inform NWP | "Sample application: Image-to-image translation from GOES to MRMS" section, Hilburn et al. (2020) |
| TRF | Theoretical receptive field | "Receptive fields" section, Araujo et al. (2019) |
| ERF | Effective receptive field | "Receptive fields" section, Luo et al. (2016) |
| LRP | Layer-wise relevance propagation | "NN visualization methods" section, Montavon et al. (2017) |
| Grad-CAM | Gradient-weighted class-activation mapping | "NN visualization methods" section, Selvaraju et al. (2017) |

## NN basics for working with images

Neural networks have opened up many new avenues to ingest and utilize images in meteorology, e.g., to identify specific patterns in an image (image classification) or to transform information in an image into a different representation (image-to-image translation). In this section we briefly review the basics of neural networks with emphasis on important concepts for working with images. For further reading on neural networks we suggest the book by Burkov (2019) for a quick introduction, and the books by Géron (2019) and Chollet (2017) for deeper study.

***From simple neurons to powerful neural networks.*** An *artificial neural network* (ANN; or NN for short) is a machine learning method loosely inspired by the human brain. An NN consists of a set of neurons (aka *nodes*) that are connected by synapses which pass signals between the neurons. In *sequential NNs*, which are the primary type considered here, all neurons are arranged in a sequence of layers, and signals pass in a one-directional manner from the input layer through intermediate layers to the output layer. The ultimate goal of the NN is to learn from data samples, which are provided in the form of input–output pairs, how to map the inputs to the outputs.

Each neuron is connected to one or more neurons in the preceding layer, and each neuron's state is represented by its *activation value*, a scalar variable that takes continuous values. A neuron's activation value is calculated from the activation values of its directly preceding neurons through simple regression, followed by application of a fixed, scalar, nonlinear function $f(x)$ known as *activation function*. A simple example is the ReLu activation function which assigns $f(x) = x$ for $x \geq 0$ and $f(x) = 0$ otherwise. While the function $f(x)$ is usually deceptively simple, its use is crucial to allow the NN to approximate nonlinear mappings from input to output.

Although neuron interaction is thus modeled in a highly simplified manner compared to the human brain, the resulting NNs show surprisingly similar behavior and functionality to

key components of the human brain, in particular the visual cortex, so much so that NNs are now considered to be important research tools in neuroscience for the study of the human brain (Schrimpf et al. 2018; Cichy and Kaiser 2019). It is thus not surprising that NNs have shown outstanding skill in interpreting, processing, and emulating images, including for meteorological applications.

***Training neural networks.*** Given enough samples of input–output pairs a neural network can attempt to learn the mapping from input to output from the samples, resulting in an approximate model. Once the model is learned (aka *trained*), it can be applied to new input samples to generate output predictions. Training a model on such input–output pairs is called *supervised learning* (Géron 2019). The input–output pairs that the model is trained on are called *training data*. There are two primary types of supervised learning tasks, namely, *regression tasks* and *classification tasks*. In a regression task the predicted output is a continuous value, such as a predicted rain rate. In contrast, in a classification task the output is restricted to a finite number of discrete states (aka *classes*), such as "rain," "hail," or "snow" for the predicted precipitation type.

Training an NN consists of simultaneously adjusting the regression parameters of the individual neurons (aka *NN weights*) to best match the training data. To do so, one first needs to choose a cost function (aka *loss function*) that continuously measures the NN's performance during training, such as the mean square error of predictions generated for the training samples by the current NN. All NN weights are assigned random values at first. Then the loss function is minimized iteratively using gradient descent, i.e., the gradient of the loss function is calculated with respect to the NN weights and the NN weights are adjusted accordingly. This step is known as *back propagation* and is repeated until some end criterion is reached.

Even very simple NNs trained this way have shown outstanding success as universal function approximators (Cybenko 1989; Hornik 1991) in many applications. A network with few layers is called *shallow* and has limited modeling abilities. In contrast a network with many layers is called *deep*, leading to the term *deep learning*. The power of deep learning comes from the ability of a deep network to recognize and utilize increasingly complex patterns in its deeper layers (Hanin 2019; Lu et al. 2017).

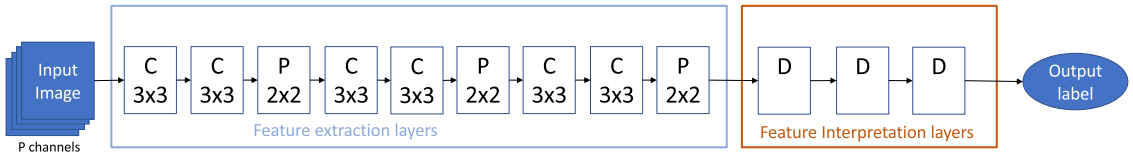
***Layer types for working with images.*** NN architectures utilize different layer types to achieve specific purposes. A *dense* (*D*) *layer* is a general purpose layer where all neurons in one layer are connected to all neurons in the next layer. In contrast the following layers are specific for working with images, namely, *convolution* (*C*), *pooling* (*P*), and *upsampling* (*U*) layers, which all connect to concepts and operations from classic image processing. A key concept in image processing is a *mask* (aka *kernel* or *filter* or *convolution matrix*). A mask is a small matrix that is convolved locally with an image to apply a certain operation to the image, such as smoothing or edge detection. *Convolution layers* in neural networks implement the same type of mask and convolution operation as classic image processing, most commonly using a $3 \times 3$ mask. However, the key difference is that the mask entries (matrix values) in NNs are treated as weights, i.e., NN parameters to be learned during training, while in classic image processing the mask entries are chosen a priori to implement a particular operation, thus fixed. The flexible convolution masks allow NNs to learn and detect important local patterns in images, such as edges and, in deeper layers, much more complex spatial patterns. Convolution layers use the same type of neuron interaction model as dense layers, but there are fewer connections between neurons, and the NN weights of neurons within the same layer are coupled and represent the mask entries.

*Pooling layers* serve the purpose of reducing the resolution of an image, most commonly by a factor of 2 in each dimension. They do so by taking nonoverlapping $2 \times 2$ patches of an

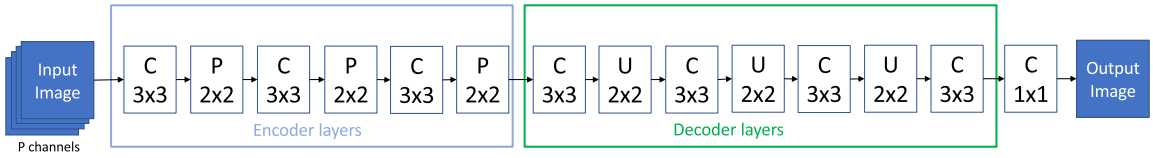
image and representing that patch by a single pixel with its value assigned either the average or maximum value of the original 2 × 2 patch. These layers are denoted as *average pooling* or *max pooling*, depending on whether they assign the average or maximum value. *Upsampling layers* serve the purpose of increasing the resolution of an image, most commonly also by a factor of 2, by expanding each pixel into a 2 × 2 image patch through interpolation. Of course even with the best interpolation scheme a pooling layer cannot be reversed by an upsampling layer as too much information is lost. Note that both pooling layers and upsampling layers are fixed transformations, i.e., they do not contain any NN weights to be trained. Graphic explanations of convolution and pooling layers can be found in the supplement of McGovern et al. (2019).

Last, it should be mentioned that a convolution layer can recognize only one pattern for each convolution mask it uses. To be able to recognize several different patterns several masks are used in each convolution layer, each one with its own set of weights and trained to its own pattern, and each one resulting in a separate output channel of that convolution layer. Thus *channels* in the intermediate layers of an NN represent the presence of different patterns in an image, which is not to be confused with the different channels of the NN input used to feed different channels of satellite imagery [e.g., different Geostationary Operational Environmental Satellite (GOES) channels] into the NN.

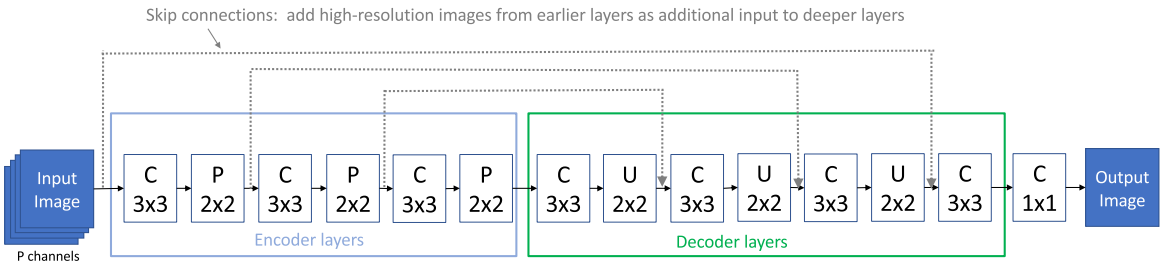
***NN architectures for working with images.*** Figure 1 illustrates common NN architectures for the tasks of image classification and image-to-image translation, using only the four different layer types discussed above, dense (D), convolution (C), max pooling (P), and upsampling (U) layers. All three sample architectures shown in Fig. 1 take as input an image, typically consisting of several channels. The classification model (Fig. 1a) yields as output the most likely image label out of a number of predefined labels (classes). Such networks can be used,




(a) Sample CNN architecture for Image Classification.




(b) Sample Encoder-Decoder architecture for Image-to-Image Translation.




(c) Sample U-net architecture for Image-to-Image Translation. This architecture can also be drawn in the shape of a 'U' so that skip connections appear as straight horizontal connections, thus the name 'U-net'.

**Fig. 1. Samples of three common NN architecture types for (a) classification and (b),(c) image-to-image translation tasks. Here "C" indicates a convolution layer, "P" a max pooling layer, "U" an upsampling layer, and "D" a dense layer. Sample filter dimensions are provided where appropriate.**

for example, to estimate tropical cyclone intensity (Wimmers et al. 2019), identify cyclones (Bonfanti et al. 2018), detect large hail (Gagne et al. 2019), detect synoptic-scale fronts (Lagerquist et al. 2019), anticipate tornadoes (McGovern et al. 2019), and predict precipitation to occur in form of rain, freezing rain, snow, or ice (McGovern et al. 2019).

In contrast the image-to-image translation models (Figs. 1b,c) generate as output an image, typically of the same dimension (but not necessarily the same number of channels) as the input image. Image-to-image translation models can be used to enhance remote sensing images (Tsagkatakis et al. 2019), to detect changes in satellite imagery (Peng et al. 2019), for precipitation forecasting (Sønderby et al. 2020), for weather forecasting (Weyn et al. 2020), to detect tropical and extratropical cyclones (Kumler-Bonfanti et al. 2020), to emulate radar imagery for satellites that only have passive channels (Hilburn et al. 2020), and to identify all locations in a satellite image corresponding to convection (Lee et al. 2020, manuscript submitted to *Atmos. Meas. Tech.*).

All three architectures start with a sequence of convolution and pooling layers, which are enclosed in a blue box in Figs. 1a–c. This sequence typically consists of repeating layer patterns of the form "CP," "CCP," etc., with the repeating pattern and number of repetitions depending on the application. The purpose of this sequence is identical in all three models, namely, to extract spatial patterns of increasing size from the input image. Since convolution filters are limited in size (most efficient filter size tends to be $3 \times 3$), the pooling layers serve the purpose of reducing resolution at their output, thus allowing subsequent convolution filters to act on pixels that represent larger and larger areas of the input space, and thus detecting larger patterns. The increase in size of the detected patterns is studied in the "Receptive fields" section.

In the classification model (Fig. 1a) the feature extraction layers (blue box) first extract spatial patterns, aka *features*. After that, the feature interpretation layers (red box) interpret the presence of the extracted features to generate a suitable output label, i.e., to determine the class the image belongs to. In contrast in the image-to-image translation models (Figs. 1b,c) the blue box is followed by a sequence of convolution and upsampling layers (green box) which increases image resolution to restore original image size, while also unfolding the image typically into a different representation, i.e., translating the detected patterns into a different, learned representation of those patterns to generate the output image. [As an analogy, one may think of the task of translating from one language to another. Given a sequence of words in one language, the encoder first extracts patterns (meaning) from the sequence, then the decoder expands those patterns in the other language.] For images, because of the information lost during the downsampling process, skip layers are often added that allow high-resolution information to be added to the later layers as supplemental channels. This type of architecture is called a U-net and is shown in Fig. 1c.

For notation, a *convolutional neural network* (CNN) can use both dense and any combination of convolution, pooling, and upsampling layers, while a *fully convolutional network* (FCN) uses any combination of convolution, pooling, and upsampling layers, but no dense layers. All three architectures in Fig. 1 are thus CNNs, but only the image-to-image translation architectures (Figs. 1b,c) are FCNs. Furthermore, the first two architectures are sequential (Figs. 1a,b), while the U-net is not sequential due to its skip layers.

A more advanced NN architecture for image-to-image translation are generative adversarial networks (GANs), which are known to produce satellite images that look particularly realistic (Xu and Zhao 2018). Gagne et al. (2020) generate stochastic distributions for meteorological applications and Stengel et al. (2020) show that GANs can successfully emulate small-scale features in meteorological fields. One of the most popular type of GAN for satellite images is Pix2Pix (Isola et al. 2017) which is a general-purpose image-to-image translation tool. Kim et al. (2019) use Pix2Pix to generate nighttime reflectance imagery from visible satellite

bands. While the details of GANs are beyond the scope of this paper, Pix2Pix typically uses either an encoder–decoder or U-net architecture as its image generator, which can be analyzed with the methods described in this article.

Last, the "Training neural networks" section described the training process of an NN model, which determines all of its weights, resulting in a complete NN model. As we saw in this section NN architectures have several additional parameters, such as the types and number of layers to be used, the filter dimensions of convolutional layers, and the activation functions and loss function used. These parameters are called *hyperparameters*, and must be chosen before an NN is trained. However, sometimes one might want to do a search over different architectures, i.e., different hyperparameters. To do that, one can conduct a *hyperparameter search* which forms a wrapper around the NN training process, by trying different sets of hyperparameters, training a complete model for each set, evaluating the resulting model, and then deciding which hyperparameter set results in best performance. Algorithms range from simple exhaustive grid search (as illustrated in the "Using performance measures for NN tuning" section) to sophisticated algorithms (Kasim et al. 2020; Hertel et al. 2020).

***Sample application: Image-to-image translation from GOES to MRMS.*** We demonstrate many of the concepts in this article for a sample application, namely, estimating Multi-Radar Multi-Sensor (MRMS) composite reflectivity (Smith et al. 2016) from GOES imagery (Schmit et al. 2017; Goodman et al. 2013). Since GOES imagery is available throughout the continental United States but MRMS has gaps, it would be useful to emulate MRMS in areas where it is not available. This is a classic image-to-image translation task and we use a classic encoder–decoder architecture of the type shown in Fig. 1b (more details below in the "Receptive fields" section). There are four input channels, namely, GOES-ABI channels 7, 9, and 13 (all infrared; Schmit et al. 2017), and the GOES-GLM channel which indicates presence and location of lightning (Goodman et al. 2013). The resulting model is called GOES Radar Estimation via Machine Learning to Inform NWP (GREMLIN), described in detail in Hilburn et al. (2020). Note that the version used here for demonstration is an earlier version of GREMLIN that has slightly different NN weights than the *final* model in Hilburn et al. (2020).

## Receptive fields

As mentioned earlier the power of deep learning comes from an NN's ability to recognize and utilize increasingly complex patterns. When considering one layer after the other in an NN, starting from the first (input layer) to the last (output layer), the general trend is that each layer can recognize patterns in the input image that are—in comparison to the previous layer—(i) more complex and (ii) larger in size. *Pattern complexity* is very difficult to evaluate for several reasons: 1) patterns can only be evaluated after NN training is completed; 2) techniques for discovering patterns, such as feature visualization (Olah et al. 2017, 2018), to date only provide limited answers; and 3) feature visualization is even more challenging for meteorological imagery, because it tends to have amorphous boundaries (e.g., clouds, atmospheric rivers, ocean eddies) (Karpatne et al. 2019) rather than the crisp boundaries common in many computer vision applications.

On the other hand, the maximal *pattern size* can easily be calculated even before the network is trained, solely based on its architecture. This maximal pattern size is known as the *theoretical receptive field* of the layer and can be used to design NN architectures that can utilize meteorological features up to a certain size. We believe that receptive fields present important information for NN design in meteorology, yet have attracted very little attention to date (Hilburn et al. 2020).

Figure 2 illustrates the concept of the *theoretical receptive field* (TRF). Considered are two NN layers, the input layer, which contains the input image (with one or more channels, only

one channel shown here) and an intermediate layer, layer *L*. Considering a single pixel in layer *L* (red cross in Fig. 2), the theoretical receptive field tells us the maximal area in the input image (red box in Fig. 2) that can impact the value of that pixel. *In other words, layer L can only understand meteorological features in the input image up to the size of the red box.*

Figure 3 illustrates the TRF for all layers of the GREMLIN architecture for a specific sample and for only one channel (ABI channel 13) of the input sample. It shows how the maximal spatial context utilized by the NN grows from layer to layer until it reaches the maximal size of 50 × 50 pixels. For a comprehensive discussion of how to calculate the TRF size for any type of NN, see Araujo et al. (2019).

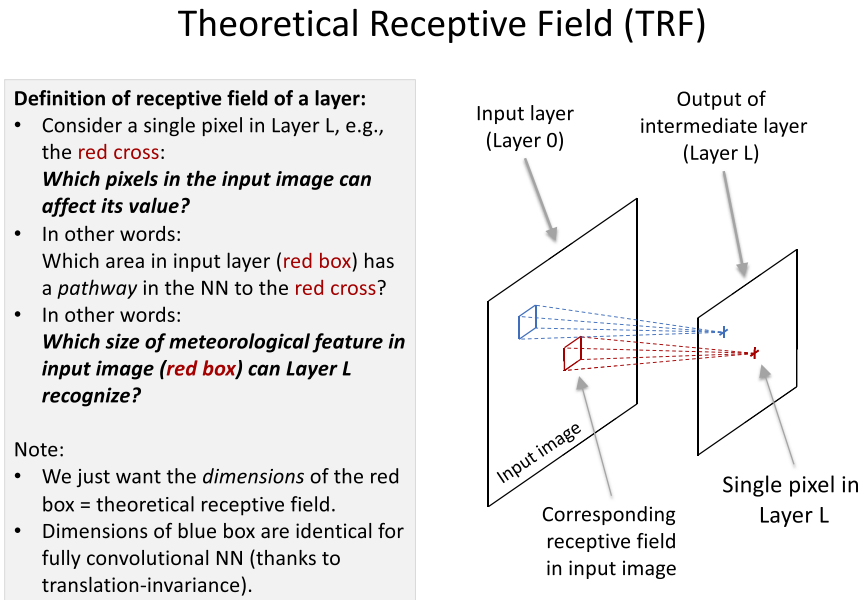



**Fig. 2. The theoretical receptive field of a considered layer (layer *L*): Considering a single pixel in layer *L* (red cross), what is the corresponding receptive field in the input image (red box)? The receptive field tells us the maximal size of meteorological feature in the input image that this layer can recognize (extent of spatial context). The receptive field tends to increase from layer to layer. Thanks to translation invariance in fully convolutional NN, the dimensions of the blue box are identical to those of the red box. The blue box is just translated depending on the location of the corresponding pixel (blue cross) in layer *L*.**

TRF size can serve as a useful guideline when selecting how many repetitions of combinations of convolution, pooling, and upsampling layers to use in NN architectures for both

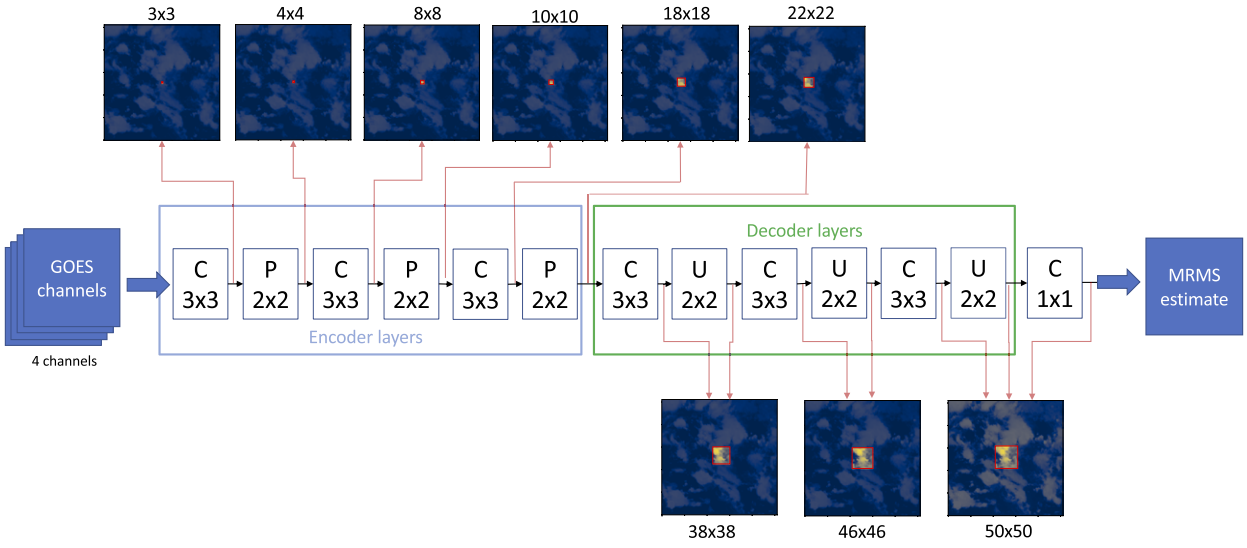



**Fig. 3. Architecture and theoretical receptive field (TRF) of all layers of GREMLIN model visualized for sample 68 and input channel 13 (longwave IR). The numbers on top of each image denote the TRF size in terms of pixels in the input sample. The red square in each image indicates the TRF corresponding to a central pixel in the output map of a considered layer. Thus the red square represents the maximal spatial context in the input image that a layer can utilize. The input image has 256 × 256 pixels and the final layer has a TRF of 50 × 50 pixels. Note that adding skip connections to the model (U-net) would not change the TRF boundaries.**

image-to-image translation tasks or classification tasks. For image-to-image translation architectures the connection is obvious: if meteorologists can provide a rough estimate of spatial context size that they think is important for the NN to consider (a good estimate for that is often the spatial pattern size a *meteorologist* would use to perform the task manually), then the NN architecture should be chosen to have a TRF that is at least as big, preferably already at the end of the encoder layers. In contrast, in image classification architectures (Fig. 1a) one considers the TRF size only up to the end of the feature extraction layers (output of blue box in Fig. 1a). That TRF size determines the maximal spatial context of the individual features used in classification. (There is no point of considering the TRF in Fig. 1a) after that point, as once a dense layer is added the TRF by definition spans the entire input image.)

While the TRF determines the *maximal* spatial context used by each layer, Luo et al. (2016) show that the actual impact of pixels varies drastically within the TRF region. They found that the impact roughly follows a Gaussian distribution, resulting in strongest impact for pixels near the TRF center and declining away from the center. The actual distribution, known as the *effective receptive field* (ERF), depends on the trained NN parameters. Figure 4 shows how much the ERF changes for GREMLIN during training.

While we highly recommend calculating the TRF, we do not think that calculating the ERF is particularly useful. Instead the main take-away from Fig. 4 is simply to realize that the ERF, i.e., *the area that the NN truly uses, might be a bit smaller than the TRF*, which is only a theoretical upper bound of that area. Thus the initial architecture guess is typically followed by some trial-and-error experiments with architectures of larger TRF size. Beyond architecture design, we find calculating the TRF of the different layers (Fig. 3) very helpful to understand the spatial context—and thus the meteorological phenomena—the NN model is able to utilize in the different layers, for the purpose of NN interpretation.

## Using performance measures for NN tuning

The meteorological community has developed many tools to evaluate the performance of algorithms used for weather and climate tasks. It seems obvious to apply such meteorological performance measures also to NN algorithms whenever they are used for the same tasks. However, that is not always happening. The likely reason for that effect is the misalignment between the needs of two disciplines, computer science and meteorology. NN methods are usually developed by computer scientists for computer science applications, which come with their own performance measures, some of which are not suitable for meteorological applications. Furthermore, NN literature illustrates NN methods

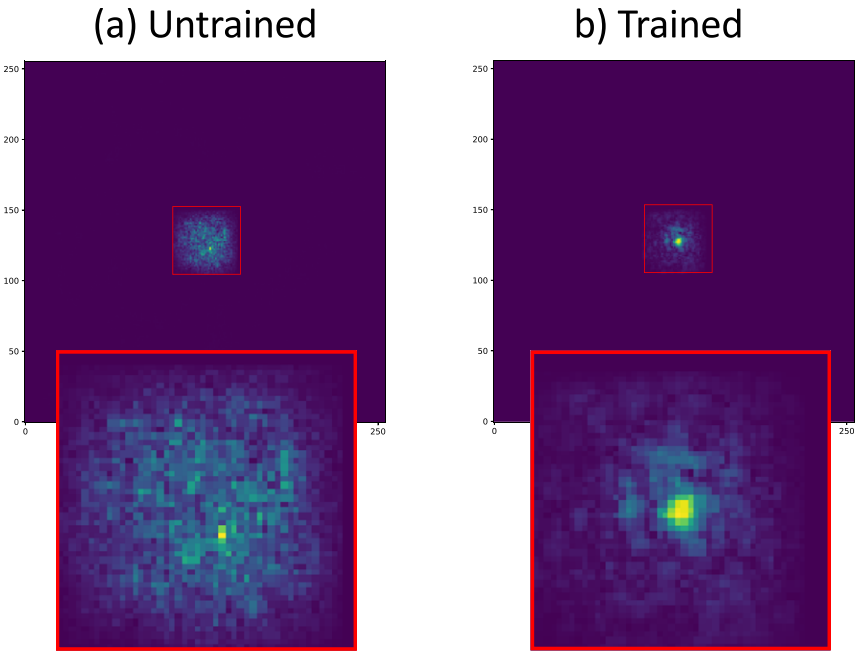



**Fig. 4. Approximation of effective receptive field (ERF) for final output layer of GREMLIN model (a) before training (0 epochs), i.e., using random weights, and (b) after training (100 epochs). This approximation is calculated with SmoothGrad (Smilkov et al. 2017) using validation sample 80 and a central output pixel; see Hilburn et al. (2020) for details. Note that the ERF for the untrained model is much more diffuse, while the trained model for this sample and location is much more focused at the center.**

with computer science-oriented performance measures and NN software packages include them as well. Thus it is only natural that meteorologists carry over not only the methods, but also the typical performance measures that the papers teach one to use. *It is important to be conscious of this potential misalignment, i.e., when transferring NN methods from the computer science literature to meteorology, not to rely solely on the accompanying performance measures, and instead to also consider the full range of meteorological performance measures.*

***Example of underutilized meteorological measure.*** The *categorical performance diagram* developed by Roebber (2009) is commonly used to evaluate algorithms in meteorology, especially for weather forecasting. However, it has only very recently been used in the context of NNs. Lagerquist et al. (2019) and Gagne et al. (2019) use performance diagrams to evaluate NNs, while our group used performance diagrams to tune NNs using a small hyperparameter search (Hilburn et al. 2020), as outlined below. We used performance diagrams to tune the GREMLIN model to estimate MRMS reflectivity on a pixel-by-pixel basis (Hilburn et al. 2020). Evaluating the output pixel-by-pixel allows us to apply classic performance measures from meteorological forecasting problems. (In contrast, measures that evaluate the output image as a whole are discussed in the next section.) A challenge for estimating MRMS reflectivity is that high reflectivity values are quite rare in the training samples (class imbalance), and use of a standard MSE loss function resulted in the NN focusing too much on providing good estimates for low reflectivity but significantly underestimated high MRMS values. This can be corrected by adding weights to the loss function that put more emphasis on the rare (high) values. The weighted function must be chosen as a trade-off to avoid losing too much performance for lower values. We tested the following family of weighted mean square error (MSE) functions as loss functions for an individual sample with true image $y^{\text{true}}$ and estimated image $y^{\text{pred}}$:

$$L\left(y^{\text{true}}, y^{\text{pred}}\right) = \sum_{p=1}^{P} w\left(y_p^{\text{true}}\right)\left(y_p^{\text{pred}} - y_p^{\text{true}}\right)^2, \tag{1}$$

$$w\left(y_p^{\text{true}}\right) = e^{b\left(y_p^{\text{true}}\right)^c}, \tag{2}$$

where $y_p^{\text{true}}$ and $y_p^{\text{pred}}$ are the true and predicted values of output pixel $p$, $P$ is the number of pixels in the output image, and $w\left(y_p^{\text{true}}\right)$ is the weight assigned to the pixel of a sample based on its true $y$ value. Hyperparameters $b$ and $c$ allow us to tune the weighting function $w$.

One can then vary $b$ and $c$ in an exhaustive grid search, for example, with $b$ ranging from 1 to 5 in increments of 0.5 and $c$ ranging from 1 to 5 in increments of 1, resulting in a total of 45 combinations. Training a separate NN for each parameter combination, and plotting the performance of the resulting 45 NNs in a categorical performance diagram, results in Fig. 5. In this example the loss function corresponding to the red line is deemed to perform best (closest to diagonal). As shown in Hilburn et al. (2020), such a choice can greatly enhance performance for high reflectivity values, without significant performance loss for low values.

***Loss functions versus auxiliary metrics.*** Performance measures can be used in neural networks in two ways, as the *loss function* or as *auxiliary metric*. The loss function tells the neural network what to optimize during training, so must be carefully chosen, but must also be differentiable to facilitate NN training via gradient descent methods. When the NN optimizes the loss function, it may do so by sacrificing other desired properties, and additional auxiliary metrics allow us to detect such trade-offs. *Our recommendations are thus 1) to carefully choose the loss function to measure the most important (differentiable) performance property for the meteorological task, and, if needed, to define a customized loss function for that purpose rather than relying on predefined choices; and 2) to define and track one or more*

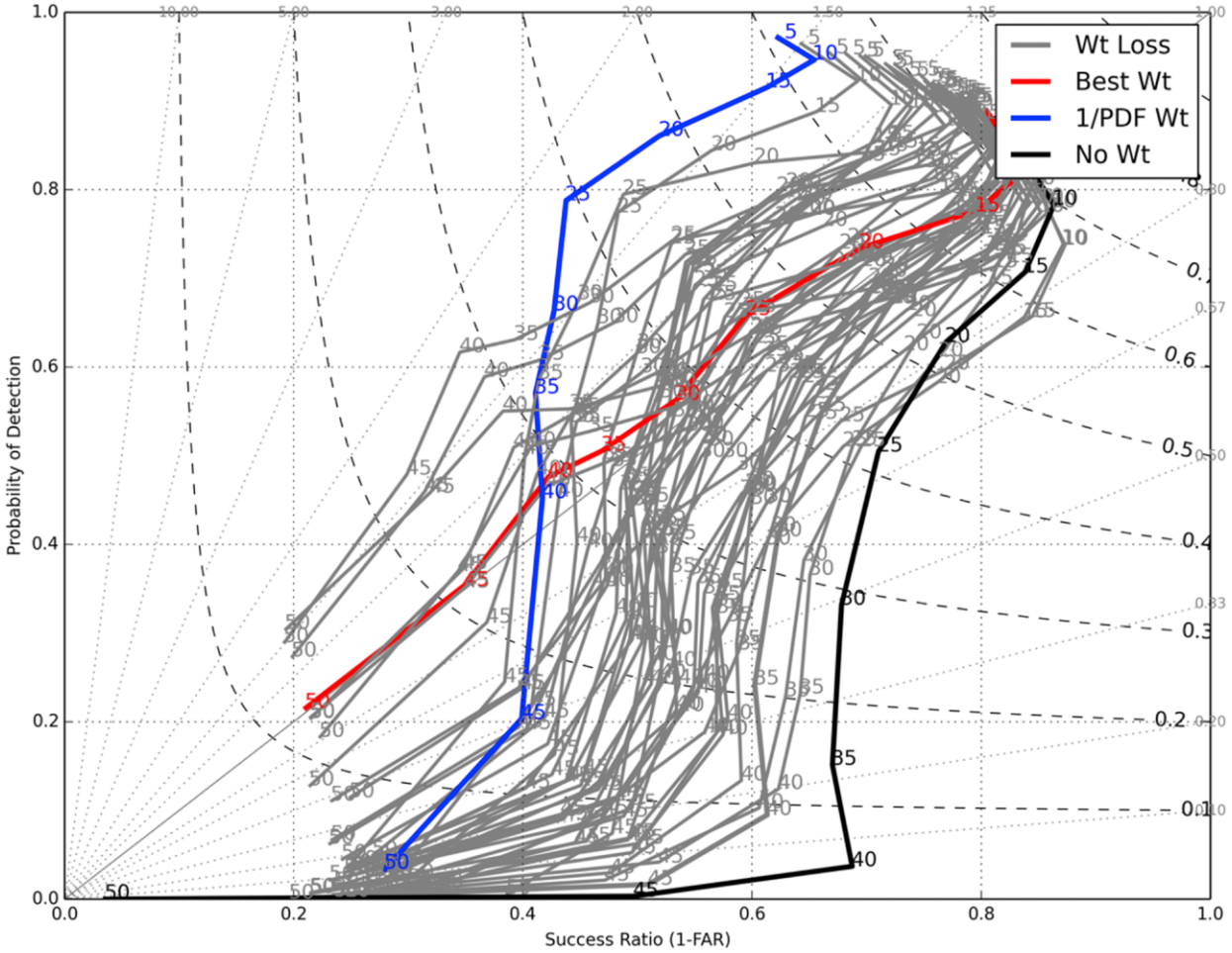



**Fig. 5. Use of categorical performance diagram to select weighted loss function for NN in MRMS estimation application. Each line represents the performance of one NN. Black line represents the unweighted case (plain MSE), blue line (marked as 1/PDF Wt.) uses the inverse of value frequency (thus 1/PDF) as its weight, and red line represents the NN selected to have the best performance. Best performance is defined here as being closest to the diagonal line, i.e., having equal tendency to overestimate as underestimate.**

*additional desired performance criteria as auxiliary metrics to ensure that unacceptable tradeoffs are detected right away.*

Once a loss function has been chosen, one can add as many auxiliary metrics as desired in the NN algorithm, which are then evaluated after every epoch of training. Auxiliary metrics do *not* have to be differentiable, which allows implementing a much bigger variety of measures than for loss functions. For example, in our MRMS reflectivity application above we implemented the categorical performance diagram as an auxiliary metric because it is not differentiable. That allowed us to nevertheless determine the NN with best categorical performance out of a finite number of options through a small hyperparameter search. It also allowed us to track the evolution of categorical performance during the NN training process, which revealed that even for our tuned loss function the NN first learned how to predict lower reflectivity values well, with performance for higher values improving much later in the training process.

A close relative of the categorical performance diagram is the receiver operator characteristic (ROC) curve and the corresponding area under the curve (AUC) is a common performance measure in meteorological applications. The AUC formula can be included as an auxiliary metric in NNs, but it is not differentiable, so cannot be included in the loss function. Researchers in the medical community derived approximations of the AUC formula that are differentiable and thus can be placed in the NN's loss function (Sahiner et al. 2013;

Ramos-Pollán et al. 2011), resulting in NNs that are trained explicitly to maximize AUC. Researchers in meteorology might consider exploring this approach for applications where AUC is of primary importance.

Finally, it is common practice to include additional terms in the loss function, combined as weighted sums. Regularization terms, which avoid NN overfitting by punishing unnecessarily large weights in the network, are a common choice. Another choice, which has only recently been proposed, seeks to incorporate knowledge about the underlying system into the neural network. Namely, if there are any known constraints that the solution must obey and which are differentiable, violation of those constraints can be included in the loss function (Karpatne et al. 2017). For other ideas of how to incorporate knowledge about the physical system into NNs, see Willard et al. (2020). Beucler et al. (2020), motivated by the application of using machine learning approaches for subscale parameterization, developed means to incorporate physical constraints not only in the NN loss function but also in the NN architecture.

***Other relevant measures.*** Categorical performance diagrams and ROC curves are just two examples of a wide spectrum of meteorological performance measures to consider. For a great overview of some interesting performance measures to consider for general machine learning algorithms in meteorology, see Gagne et al. (2015). Of particular interest to image-to-image translation applications are distance measures comparing pairs of images that go beyond pixel-by-pixel comparison, such as the *Wasserstein distance*, aka *Earth mover's distance*, from optimal transport theory (Snow et al. 2016), and the *structural similarity index measure* (SSIM) by Wang et al. (2004) that are already well established in remote sensing applications including in the context of NNs. See Alberga (2009) and Tsagkatakis et al. (2019) for a wide variety of image evaluation measures for remote sensing.

## Interpretation through targeted experiments

This section discusses ways to interpret NNs through targeted experiments. The "NN visualization methods" section describes NN visualization methods that can aid these targeted experiments.

***Designing experiments—Generating sets of input samples to investigate.*** A powerful means to learn about the internals of a neural network is to split the input samples into different groups and to investigate how the NN behaves for each group. This section provides ideas for such groupings, starting with the simplest, fairly automatic grouping and progressing to groupings that increasingly use meteorological expertise and increase in complexity and effort. In fact, the last strategy actually does not group existing input samples, but creates new sets from scratch (synthetic input samples), completely driven by meteorological expertise.

Once a set of input samples is chosen, for example using one of the strategies outlined in this subsection, then steps such as illustrated in Fig. 6 can be used to generate hypotheses. Note that steps 4–6 are all "manual" steps to be performed by a meteorologist in conjunction with a machine learning expert.

**Studying the biggest successes and failures.** Extreme behaviors tend to provide the biggest clues, so an analysis should usually start by studying the NN's biggest successes and failures. This is a very common technique; e.g., see Lagerquist et al. (2020). For example, to study failures for a classification task one can study several input samples that lead to false alarms, as well as several input samples that lead to misses. Likewise, for regression tasks, one should identify and study input samples that led to gross over- or underestimation. Sample questions to ask to analyze failures:

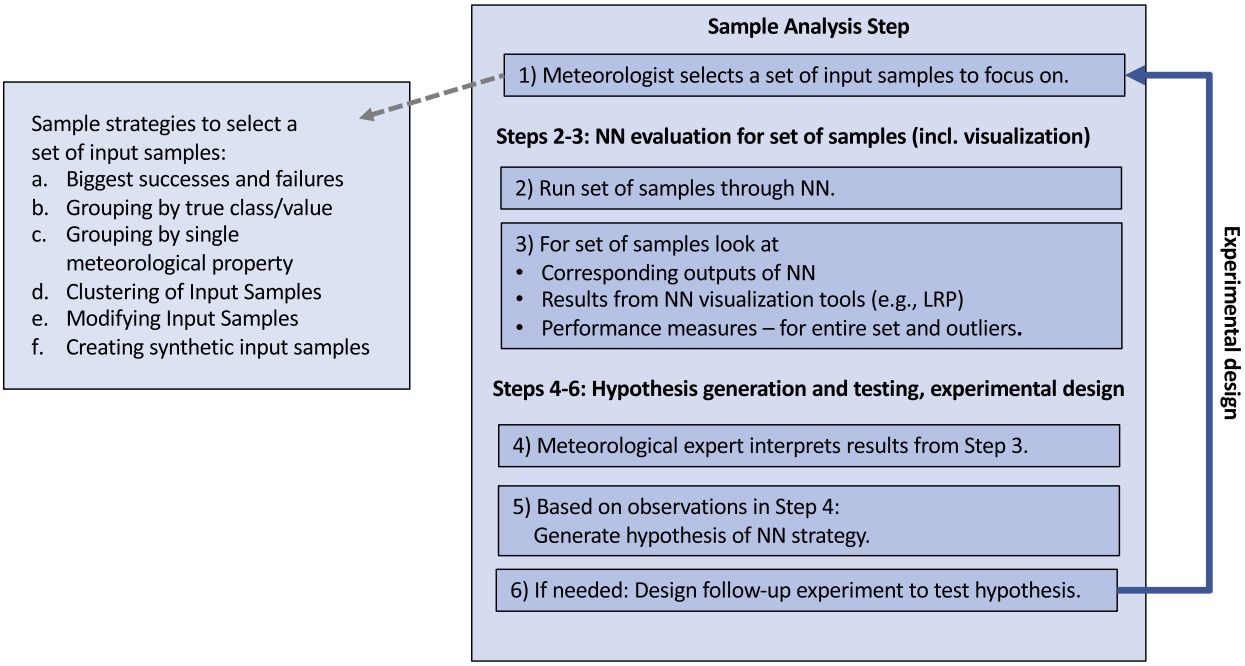



**Fig. 6. Sample analysis step focusing on meteorologically motivated subsets of input samples.**

1) Do those input samples have any obvious commonalities? Can I find a meteorological reason why these cases would be particularly challenging?
2) Where in the input was the neural network looking when it made the (bad) decision? Was it paying attention in the correct places? Several of the NN visualization tools can shed light on that question (see the "NN visualization methods" section).

The study of successes follows analogously. Note that out of all strategies discussed in this section, this is the only one that selects input samples strictly based on the NN's performance.

**Grouping by true class/value.** For a classification task one should always study the performance of each class separately, to see whether performance differs significantly by class. This is of course particularly important if one of the classes represents rare events, because performance for that class might otherwise easily get overlooked. Likewise, for a regression task one may group the input samples into groups of small, medium, or high correct output values, and study how well the NN performs in each group.

**Grouping by single meteorological property.** It is also helpful to split the input samples by properties of the input samples that we know may greatly impact the underlying physical processes. This is also known as *case studies*. These might include location, time, or specific meteorological conditions. One can then investigate the NN's dependence as follows (these are just sample criteria, not an exhaustive list):

1) Location dependence: How does the algorithm perform at different latitudes? Over land versus over ocean? Important insights can often be gained by generating a map showing NN performance by location.
2) Time dependence: Does the algorithm perform better in summer months than in winter months? During the day or at night?
3) Meteorological conditions (of course these should be entirely motivated by the application): Does the algorithm perform better for strong or weak weather events?

The first step above is for meteorologists to choose a categorization that they suspect might make a significant difference for the performance of the algorithm, and that is straight forward to evaluate for the input samples. Then the input samples are assigned to the different categories (choosing value ranges for the chosen property to create discrete categories), and finally the performance of the neural network is visualized for each category. For example, maps of performance across locations often result in interesting insights that can identify information or modeling gaps, e.g., the meteorologist may suspect that certain processes are not properly captured by the NN or that there is insufficient information in the input features for the model to be able to perform its task—thus often leading to additional variables being added to the input (such as latitude and time of day).

**Clustering of input samples.** Sometimes it is useful to group input samples by different meteorological regimes that are not easily described by a meteorological measure. In that case one might apply clustering to the input samples to obtain groups representing common, well separated, meteorological scenarios, e.g., clustering images by cloud type (Haynes et al. 2011; Denby 2020).

**Modifying input samples.** In addition to selecting input samples, we can also modify them. For example, one way to analyze the GREMLIN model for MRMS estimation is to keep the first three channels intact, but to erase all lightning in the GLM channel, to examine the effect of lightning, as demonstrated in the "Layer-wise relevance propagation" section. Of course the type of modification, such as setting a channel to zero versus assigning random values, depends entirely on the application, thus should be designed by the meteorologist.

**Creating synthetic samples.** The most complex option is to create synthetic input samples that represent specific meteorological input scenarios. Creating such examples is challenging, and probably not possible for all applications. However, if achievable this provides a great tool to test the behavior of the NN by providing perfect control of the input scenario, i.e., in isolation of any other factors that might be present in an observed input sample. This can be a powerful tool in hypothesis testing, where the input scenario is specifically designed to test a single aspect.

As example we revisit our MRMS estimation model, GREMLIN, from the "Sample application: Image-to-image translation from GOES to MRMS" section. As we will see in the following section, NN visualization methods reveal that GREMLIN is using radiance gradients to infer locations of high composite reflectivity. However, this is in the context of real samples that have complicated spatial patterns, which makes it difficult to answer questions such as, are sharper gradients associated with higher composite reflectivity? To study such questions in isolation it is helpful to specify inputs with a parametric model that allows us to vary certain properties of the input in order to evaluate the sensitivity of the NN response. One such model we have had success with (Hilburn et al. 2020) is a generalized elliptical Gaussians (GEG) model, which assumes an outer Gaussian $G_o$ that represents the thunderstorm anvil, and an inner Gaussian $G_i$ that represents the overshooting top. The synthetic brightness temperature $T$ is a function of $(x, y)$ with the following parameters: location $x_0$ and $y_0$, amplitude $A$, size $S$, aspect $\alpha$, orientation $\theta$, and sharpness (exponent) $p$ for the outer and inner Gaussians, denoted respectively with subscripts $o$ and $i$:

$$\hat{x}_{o,i} = \left(x - x_{o,o,i}\right)\cos\theta_{o,i} - \left(y - y_{o,o,i}\right)\sin\theta_{o,i}, \tag{3}$$

$$\hat{y}_{o,i} = \left(x - x_{o,o,i}\right)\sin\theta_{o,i} + \left(y - y_{o,o,i}\right)\cos\theta_{o,i}, \tag{4}$$

$$T_{o,i} = \exp\left\{-1\left[\frac{\hat{x}_{o,i}^2}{2s_{o,i}^2} + \frac{\hat{y}_{o,i}^2}{2\left(s_{o,i}\alpha_{o,i}\right)^2}\right]^{p_{o,i}}\right\}, \tag{5}$$

$$T = A_o T_o + A_i T_i. \tag{6}$$

Figure 7a provides the synthetic input for which GREMLIN responds with maximum composite reflectivity (Fig. 7b), based on a grid search of thousands of parameter settings for the synthetic inputs. Other strongly activating patterns are similar to this one and have in common very large $p_o$ and large $p_i$, meaning that the anvil and overshooting top have very sharp $T_B$ gradients. In contrast, the patterns for which the NN responds weakly have a meteorologically unphysical appearance. *These experiments indicate that the NN has learned to recognize realistic overshooting top satellite signatures.*

***Designing experiments—Studying a simpler neural network.*** If interpreting the original network seems intractable it can be helpful to study simpler network architectures first. The general concept of restricting a model's abilities and studying the abilities of the restricted model is also known as *ablation study* (Sønderby et al. 2020; Raghu and Schmidt 2020).

Figure 8 provides an overview of the process. One simplification we find particularly helpful for convolutional neural networks is to reduce the dimensions of all convolution, pooling and upsampling layers down to 1 × 1. This allows us—with only changes to a few lines of code—to turn a neural network that takes spatial context into account into a neural network that analyzes each pixel separately, eliminating any ability to use spatial context. This enables us to study how important the spatial context is for this problem, i.e., how much performance we lose. A visual comparison of the qualitative difference between the NN output with or without spatial context often provides clues for the strategy used by the full network. An example is shown in Fig. 9 which shows the results for four NNs of increasing complexity related to GREMLIN (Figs. 9c–f). As one would expect the biggest improvement increase is between Figs. 9d and 9e, i.e., when we allow the model to use spatial context. Note how poorly the models perform without spatial context. The methods in the next section can shine light on *how* specifically an NN uses spatial context.

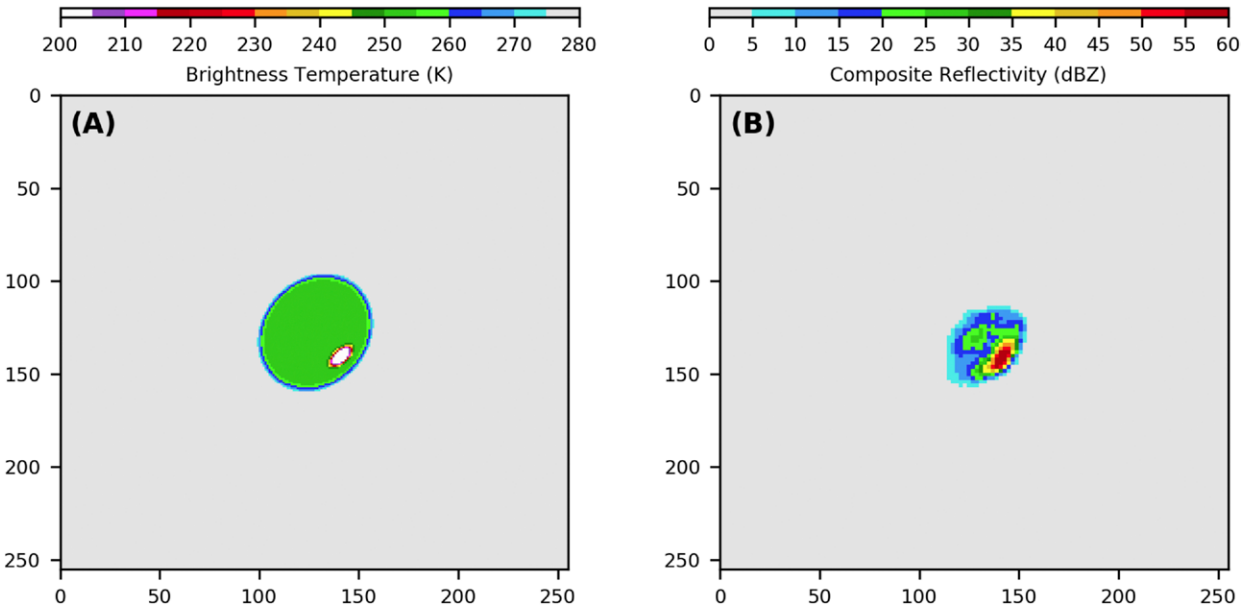



**Fig. 7. (a) Synthetic brightness temperature that produces maximum composite reflectivity from a grid search of over 45,000 parameter settings. (b) Composite reflectivity retrieved from input in (a).**

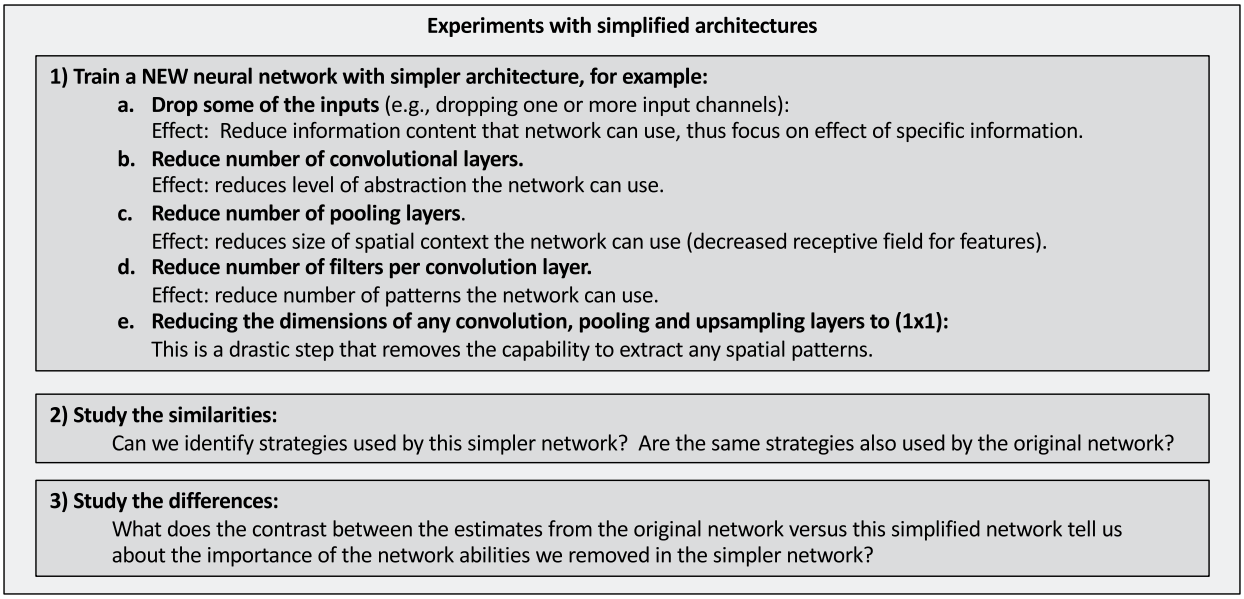



**Fig. 8. Studying a hierarchy of restricted NN architecture to shed light on strategies of the original NN architecture. Key steps are to strategically define which capabilities of original model to restrict and to carefully study both similarities and differences in capabilities between the original model and restricted (simplified) models.**

## NN visualization methods

McGovern et al. (2019) provide an excellent overview of many neural network interpretation and visualization methods and demonstrate their use for several meteorological applications. We refer the reader to that comprehensive overview to learn about the methods of permutation importance, sequential (forward or backward) selection, saliency maps, gradient-weighted class-activation maps (Grad-CAM), backward optimization, and novelty detection. For an overview of general explainable AI methods, see Samek (2019). Here we focus first on the method of layer-wise relevance propagation (LRP), which is not widely known in the meteorological community [and was not studied by McGovern et al. (2019)], yet we find it to be the most promising method in the climate and weather applications we considered. Next, we contrast LRP with saliency maps, and finally we present some pitfalls and solutions for using various visualization tools for image-to-image translation versus classification tasks.

Most methods discussed below focus on analyzing an NN model by focusing on a specific sample, and providing a *heat map* in the input space that can be overlayed as a mask on the input sample. Studying such a heat map, in addition to the input sample and corresponding output, can provide important clues on how the NN derived the output, be that a class or an image.

***Layer-wise relevance propagation.*** LRP answers the question of *where in the input the neural network was paying the most attention* when deriving its result. LRP and its variations were developed by Montavon et al. (2017, 2018) and Lapuschkin et al. (2019). Toms et al. (2020) introduced its use for geoscience applications. We have consistently found LRP to be the most informative NN visualization method for meteorological applications. For example, the authors and their collaborators have used it to uncover strategies used by NNs to discover spatial patterns of climate phenomena (Barnes et al. 2020; Toms et al. 2020), to detect convection in satellite imagery (Lee et al. 2020, manuscript submitted to *Atmos. Meas. Tech.*), and to estimate MRMS (Hilburn et al. 2020). However, to the best of our knowledge, no other research group has so far used LRP for meteorological applications.

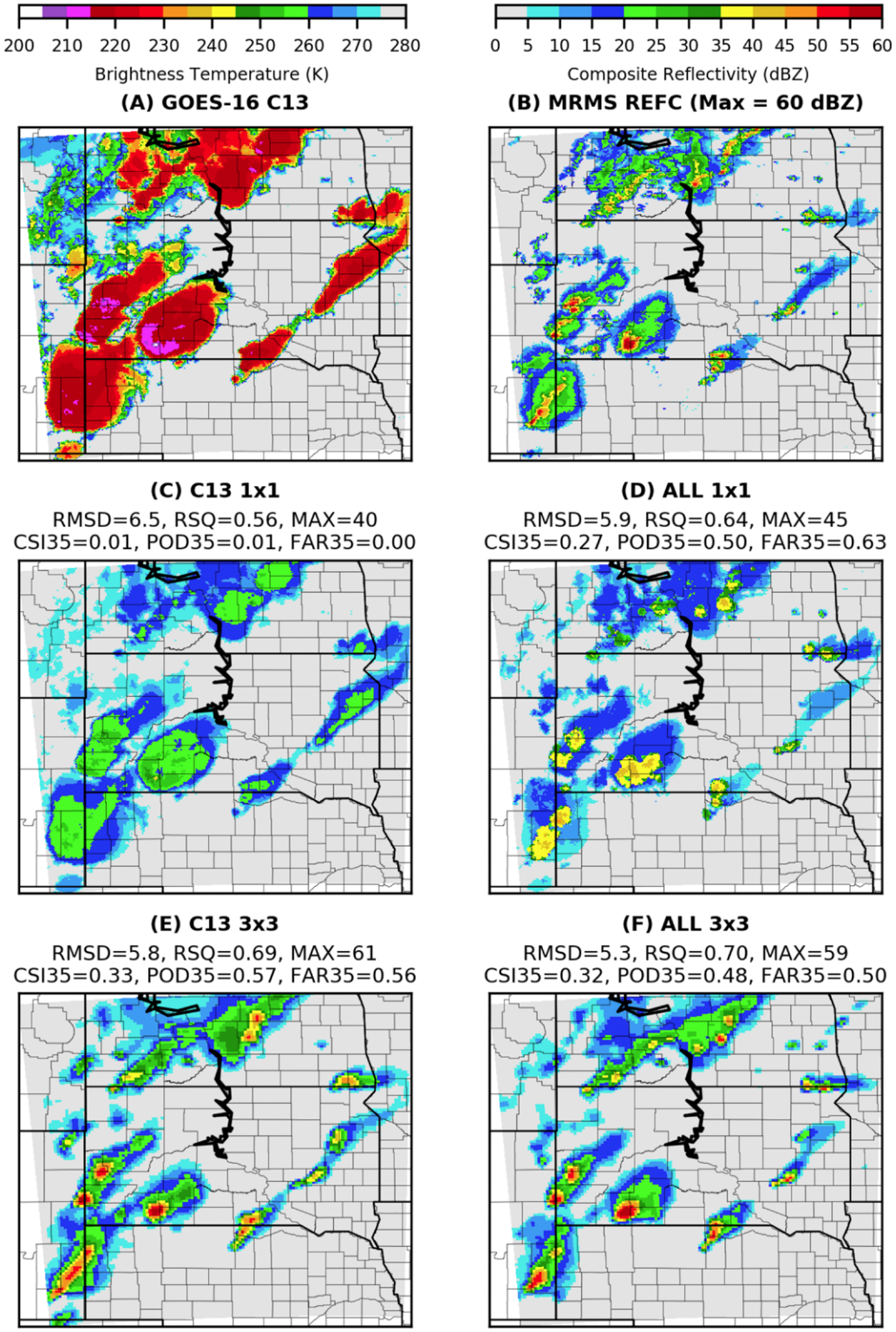



**Fig. 9. Example of studying a hierarchy of simpler NN architectures for GREMLIN. (a),(b) Sample input channel and desired output: (a) GOES-ABI channel 13 (input) and (b) MRMS (true output). (c)–(f) MRMS estimates for four different models: (c),(d) only 1 × 1 convolution and either (c) only one input channel or (d) all input channels; and (e),(f) as in (c) and (d), but using 3 × 3 convolution. Performance without spatial context, i.e., (c) and (d), is poor. Using spatial context but only one input channel, i.e., (e), is much improved, with additional improvement when all channels are used, i.e., (f). Note that (f) represents the full GREMLIN model. Results are for validation sample 80, at 2215 UTC 3 Jul 2019.**

Figure 10 shows an example of LRP for GREMLIN. This example focuses on analyzing for a specific sample how GREMLIN derives the MRMS estimate for a chosen output pixel. The first three rows provide information on the input and output for the sample. The fourth row shows the LRP heat map, which indicates that for this sample the NN was mainly focusing on

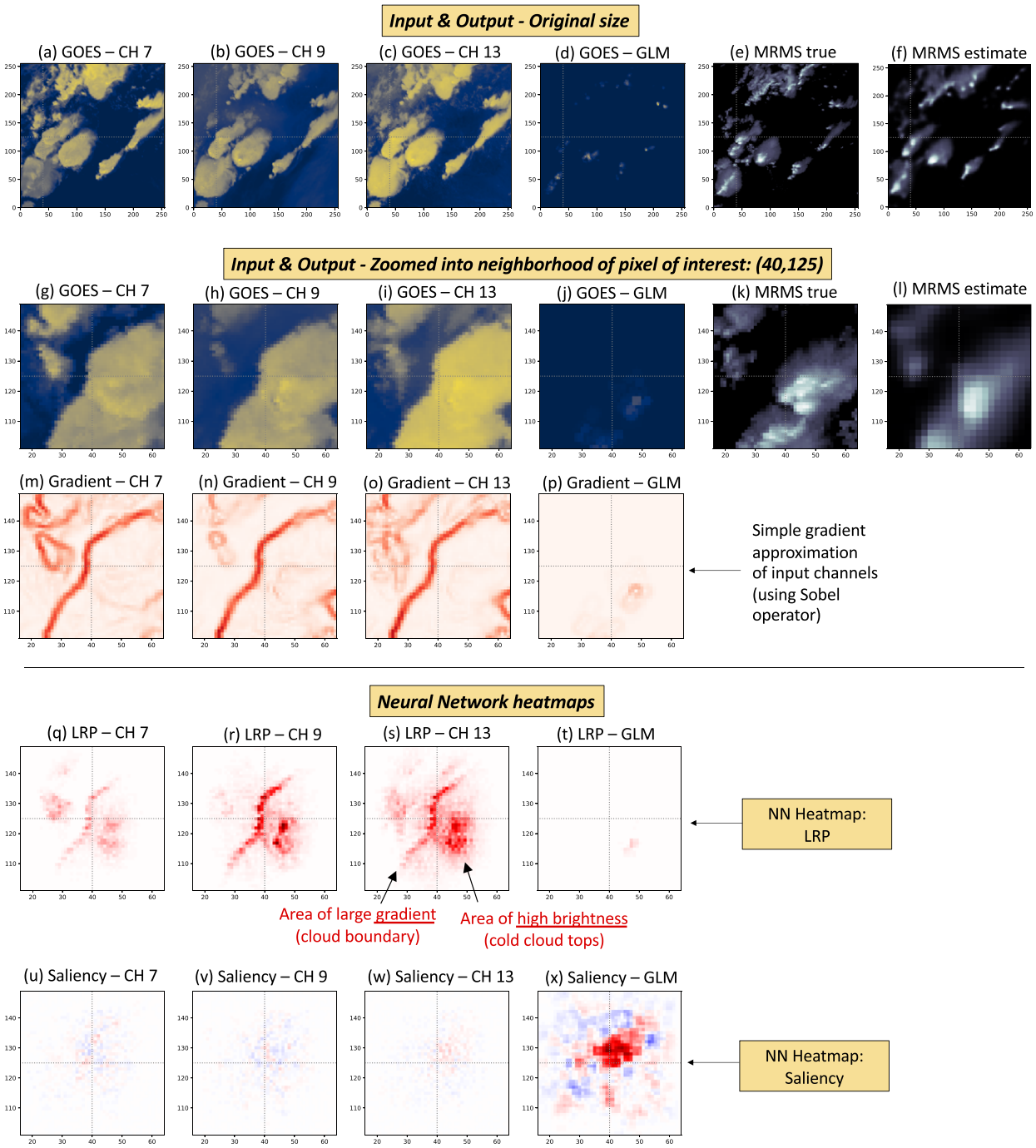



**Fig. 10. Analysis of strategies of NN model GREMLIN for sample 80 at output location ($x$ = 40, $y$ = 125). (a)–(f) Inputs and true/estimated outputs for full image. (g)–(l) As in (a)–(f), but zoomed into neighborhood of desired output pixel. (The zoom level is chosen to match the size of the TRF.) (m)–(p) The gradient of the input channels obtained by applying a Sobel operator (Gonzalez and Woods 2002) to the input channels. This auxiliary information is not used by the NN and only provided to highlight high gradient areas in the input channels for the reader. (q)–(t) Results for LRP heat maps, which indicate that the NN focused here on both the gradient and high brightness regions in the three infrared channels to determine the output value. (u)–(x) Results for saliency heat maps, which indicate that the easiest way to increase the value of the output pixel is to increase lightning. LRP and saliency maps are calculated using the iNNvestigate package (Alber et al. 2019), using the $\alpha$–$\beta$ rule for LRP with $\alpha$ = 1 and $\beta$ = 0.**

locations in the first three input channels, specifically on areas of high brightness (cold cloud tops) and areas of large gradient (primarily cloud boundaries). Analyzing this and many other samples using LRP revealed the following strategies used by our NN at any considered location: 1) Presence of strong lightning nearby takes priority, often shifting the NN's focus even in the first three input channels toward the lightning locations. 2) In the absence of lightning nearby, the NN primarily looks for areas of cold cloud tops or of cloud boundaries nearby. 3) If neither of the above applies, then the MRMS estimate is set to (nearly) zero.

After identifying these general strategies using LRP we then studied these strategies further using our simplified architecture experiments (Figs. 8, 9) to identify the importance of spatial context and of specific channels, and our synthetic experiments (Fig. 7) to quantify the impact of key meteorological properties in the input scenarios.

***Saliency maps.*** Saliency maps, developed by Simonyan et al. (2013), are probably the most popular heat map method for NN visualization. Saliency maps answer the question *which input pixels should be changed to yield a maximal increase in the considered output value with minimal change in the input image?* While this question sounds similar to the question answered by LRP, the results can differ drastically. Consider the example in Fig. 10, where the last row shows saliency results. The saliency maps indicate that the most efficient way to increase GREMLIN's output value for the considered sample through modification of the input channels is to add lightning nearby. This matches strategy 1 discovered by our LRP analysis, namely, strong sensitivity to presence of lightning. However, saliency returns an almost identical result for any sample we considered, i.e., saliency was unable to identify any of the other strategies used by the NN. This is explained by the fact that saliency maps only look at *local gradients* in the input space, which can be limiting, as demonstrated by Montavon et al. (2017), while LRP takes a more global view. On the other hand, saliency is widely available for any network type, while LRP implementations are currently available only for a small set of NN architectures.

***Visualization methods for image-to-image translation versus image classification.*** Finally we discuss some considerations and pitfalls encountered when using heat map methods for image-to-image translation versus classification tasks. Heat maps were developed for classification tasks, so some caution must be applied when using them for other tasks. First of all, when generating heat maps (e.g., for saliency or LRP) the algorithm needs to be fed an input sample and must know which output neuron should be analyzed. For classification tasks it is common practice to let the algorithm use the default neuron, namely, the output neuron with the highest value. For a classification task this default is a meaningful choice, since in that case the output neurons represent the different classes, and the neuron with the highest value corresponds to the label the NN selected as most likely class for the image. That selection is typically exactly what the user wants to analyze. However, for an image-to-image translation task, where the output is an image, this default usually is not meaningful, because in that case the default represents the image pixel with the highest value, which is usually not the pixel the user wants to analyze, thus the algorithm may yield misleading results without providing a warning. Thus the user must ensure that the default mode is turned off and instead specify a desired pixel in the output image to be analyzed. Furthermore, to specify that output pixel the user often has to clear yet another hurdle. As these algorithms are designed for classification tasks they often implicitly assume that the output is a vector, thus the user might have to add a flatten layer to the output to convert the output image into a vector, then specify the desired pixel in that flattened representation. We hope that by working more closely with the computer science community on development of tools specifically for the Earth science community, future tools will not present these additional hurdles.

Second, gradient-weighted class-activation maps (Grad-CAM), a method used with great success by McGovern et al. (2019) for classification tasks, requires the NN to have at least one dense layer. Thus Grad-CAM is not applicable to typical NN architectures for image-to-image translation, namely, fully convolutional networks, which never contain a dense layer. Note that implementations typically do not check for this, and might return heat maps even if the method does not apply. It is thus the responsibility of the user to ensure that an applied method is indeed applicable for the considered NN architecture.

Third, we tried to apply the method of optimal input, aka feature visualization (Olah et al. 2017, 2018), to our MRMS prediction task, but it always yielded adversarial examples rather than physically meaningful results. We suspect that optimal input is better suited for classification tasks, see for example its successful use by McGovern et al. (2019), because different classes might have dominant patterns that act as attractors for the optimal input, and no equivalent patterns might exist in typical image-to-image translation tasks. However, this is a topic of active investigation.

## Conclusions

This article highlights many strategies and practical considerations for the development and interpretation of neural networks for meteorological applications, including the concept of effective receptive fields, discussion of performance measures, and a long list of potential methods for NN interpretation, with synthetic experiments and LRP heat maps emerging as particularly useful tools. A common thread that emerged throughout is that NN interpretation needs to be driven by meteorological experts via specific questions or hypotheses to be investigated for the NN. Meteorologists have a crucial role to play to develop creative, meteorologically motivated standards for all aspects of NN development and interpretation, and such efforts will go a long way to create trustworthy neural networks for operational use in meteorology.

**Acknowledgments.** This work was supported in part by the National Science Foundation through Grant OAC-1934668 (Ebert-Uphoff) and by NOAA's GOES-R Program through Award NA19OAR4320073. We thank NOAA RDHPCS for access to the Fine Grain Architecture System on Hera, without which this research would not have been possible. We are grateful to the developers of the iNNvestigate package (Alber et al. 2019) (available at https://github.com/albermax/innvestigate).

## References


Alber, M., and Coauthors, 2019: iNNvestigate neural networks! *J. Mach. Learn. Res.*, **20**, 93, www.jmlr.org/papers/volume20/18-540/18-540.pdf.

Alberga, V., 2009: Similarity measures of remotely sensed multi-sensor images for change detection applications. *Remote Sens.*, **1**, 122–143, https://doi.org/10.3390/rs1030122.

Araujo, A., W. Norris, and J. Sim, 2019: Computing receptive fields of convolutional neural networks. *Distill*, **4**, e21, https://doi.org/10.23915/distill.00021.

Barnes, E. A., J. W. Hurrell, I. Ebert-Uphoff, C. Anderson, and D. Anderson, 2019: Viewing forced climate patterns through an AI lens. *Geophys. Res. Lett.*, **46**, 13 389–13 398, https://doi.org/10.1029/2019GL084944.

——, B. Toms, J. W. Hurrell, I. Ebert-Uphoff, C. Anderson, and D. Anderson, 2020: Indicator patterns of forced change learned by an artificial neural network. *J. Adv. Model. Earth Syst.*, **12**, e2019MS002002, https://doi.org/10.1029/2020MS002195.

Beucler, T., M. Pritchard, P. Gentine, and S. Rasp, 2020: Towards physically-consistent, data-driven models of convection. arXiv, https://arxiv.org/abs/2002.08525.

Bonfanti, C., L. Trailovic, J. Stewart, and M. Govett, 2018: Machine learning: Defining worldwide cyclone labels for training. *21st Int. Conf. on Information Fusion*, Cambridge, United Kingdom, IEEE, 753–760, https://doi.org/10.23919/ICIF.2018.8455276.

Boukabara, S.-A., V. Krasnopolsky, J. Q. Stewart, E. S. Maddy, N. Shahroudi, and R. N. Hoffman, 2019: Leveraging modern artificial intelligence for remote sensing and NWP: Benefits and challenges. *Bull. Amer. Meteor. Soc.*, **100**, ES473–ES491, https://doi.org/10.1175/BAMS-D-18-0324.1.

Burkov, A., 2019: *The Hundred-Page Machine Learning Book*. Burkov, 160 pp.

Chollet, F., 2017: *Deep Learning with Python.* Manning Publications, 384 pp.

Cichy, R. M., and D. Kaiser, 2019: Deep neural networks as scientific models. *Trends Cognit. Sci.*, **23**, 305–317, https://doi.org/10.1016/j.tics.2019.01.009.

Cybenko, G., 1989: Approximation by superpositions of a sigmoidal function. *Math. Contr. Signals Syst.*, **2**, 303–314, https://doi.org/10.1007/BF02551274.

Denby, L., 2020: Discovering the importance of mesoscale cloud organization through unsupervised classification. *Geophys. Res. Lett.*, **47**, e2019GL085190, https://doi.org/10.1029/2019GL085190.

Gagne, D. J., II, A. McGovern, J. Brotzge, M. Coniglio, J. Correia Jr., and M. Xue, 2015: Day-ahead hail prediction integrating machine learning with storm-scale numerical weather models. *27th Conf. on Innovative Applications of Artificial Intelligence*, Austin, TX, AAAI, 3954–3960.

——, S. E. Haupt, D. W. Nychka, and G. Thompson, 2019: Interpretable deep learning for spatial analysis of severe hailstorms. *Mon. Wea. Rev.*, **147**, 2827–2845, https://doi.org/10.1175/MWR-D-18-0316.1.

——, H. M. Christensen, A. C. Subramanian, and A. H. Monahan, 2020: Machine learning for stochastic parameterization: Generative adversarial networks in the Lorenz '96 model. *J. Adv. Model. Earth Syst.*, **12**, e2019MS001896, https://doi.org/10.1029/2019MS001896.

Géron, A., 2019*: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques To Build Intelligent Systems*. O'Reilly Media, 856 pp.

Gonzalez, R. C., and R. E. Woods, 2002: *Digital Image Processing*. Prentice Hall, 793 pp.

Goodman, S. J., and Coauthors, 2013: The GOES-R Geostationary Lightning Mapper (GLM). *Atmos. Res.*, **125–126**, 34–49, https://doi.org/10.1016/j.atmosres.2013.01.006.

Hanin, B., 2019: Universal function approximation by deep neural nets with bounded width and ReLu activations. *Mathematics*, **7**, 992, https://doi.org/10.3390/math7100992.

Haynes, J. M., C. Jakob, W. B. Rossow, G. Tselioudis, and J. Brown, 2011: Major characteristics of Southern Ocean cloud regimes and their effects on the energy budget. *J. Climate*, **24**, 5061–5080, https://doi.org/10.1175/2011JCLI4052.1.

Hertel, L., J. Collado, P. Sadowski, J. Ott, and P. Baldi, 2020: Sherpa: Robust hyperparameter optimization for machine learning. arXiv, https://arxiv.org/abs/2005.04048.

Hilburn, K. A., I. Ebert-Uphoff, and S. D. Miller, 2020: Development and interpretation of a neural network-based synthetic radar reflectivity estimator using GOES-R satellite observations. *J. Appl. Meteor. Climatol.*, **60**, 3–21, https://doi.org/10.1175/JAMC-D-20-0084.1.

Hornik, K., 1991: Approximation capabilities of multilayer feedforward networks. *Neural Networks*, **4**, 251–257, https://doi.org/10.1016/0893-6080(91)90009-T.

Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros, 2017: Image-to-image translation with conditional adversarial networks. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Honolulu, HI, IEEE, 5967–5976, https://doi.org/10.1109/CVPR.2017.632.

Karpatne, A., W. Watkins, J. Read, and V. Kumar, 2017: Physics-guided neural networks (PGNN): An application in lake temperature modeling. arXiv, https://arxiv.org/abs/1710.11431.

——, I. Ebert-Uphoff, S. Ravela, H. A. Babaie, and V. Kumar, 2019: Machine learning for the geosciences: Challenges and opportunities. *IEEE Trans. Knowl. Data Eng.*, **31**, 1544–1554, https://doi.org/10.1109/TKDE.2018.2861006.

Kasim, M., and Coauthors, 2020: Up to two billion times acceleration of scientific simulations with deep neural architecture search. arXiv, https://arxiv.org/abs/2001.08055.

Kim, K., J.-H. Kim, Y.-J. Moon, E. Park, G. Shin, T. Kim, Y. Kim, and S. Hong, 2019: Nighttime reflectance generation in the visible band of satellites. *Remote Sens.*, **11**, 2087, https://doi.org/10.3390/rs11182087.

Kumler-Bonfanti, C., J. Stewart, D. Hall, and M. Govett, 2020: Tropical and extratropical cyclone detection using deep learning. arXiv, https://arxiv.org/abs/2005.09056.

Lagerquist, R., A. McGovern, and D. J. Gagne II, 2019: Deep learning for spatially explicit prediction of synoptic-scale fronts. *Wea. Forecasting*, **34**, 1137–1160, https://doi.org/10.1175/WAF-D-18-0183.1.

——, ——, C. R. Homeyer, D. J. Gagne, and T. Smith, 2020: Deep learning on three-dimensional multiscale data for next-hour tornado prediction. *Mon. Wea. Rev.*, **148**, 2837–2861, https://doi.org/10.1175/MWR-D-19-0372.1.

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. *Nat. Commun.*, **10**, 1096, https://doi.org/10.1038/s41467-019-08987-4.

Lee, Y.-J., D. Hall, J. Stewart, and M. Govett, 2018: Machine learning for targeted assimilation of satellite data. *Joint European Conf. on Machine Learning and Knowledge Discovery in Databases*, Dublin, Ireland, ECML PKDD, 53–68.

Lu, Z., H. Pu, F. Wang, Z. Hu, and L. Wang, 2017: The expressive power of neural networks: A view from the width. *31st Conf. on Neural Information Processing Systems*, Long Beach, CA, NIPS, 6231–6239.

Luo, W., Y. Li, R. Urtasun, and R. Zemel, 2016: Understanding the effective receptive field in deep convolutional neural networks. *30th Conf. on Neural Information Processing Systems*, Barcelona, Spain, NIPS, 4898–4906.

McGovern, A., and R. A. Lagerquist, 2020: Using machine learning and model interpretation and visualization techniques to gain physical insights in atmospheric science. *AI for Earth Sciences Workshop*, ICLR, https://ai4earthscience.github.io/iclr-2020-workshop/papers/ai4earth16.pdf.

——, ——, D. J. Gagne, G. E. Jergensen, K. L. Elmore, C. R. Homeyer, and T. Smith, 2019: Making the black box more transparent: Understanding the physical implications of machine learning. *Bull. Amer. Meteor. Soc.*, **100**, 2175–2199, https://doi.org/10.1175/BAMS-D-18-0195.1.

Montavon, G., S. Lapuschkin, A. Binder, W. Samek, and K.-R. Müller, 2017: Explaining nonlinear classification decisions with deep Taylor decomposition. *Pattern Recognit.*, **65**, 211–222, https://doi.org/10.1016/j.patcog.2016.11.008.

——, W. Samek, and K.-R. Müller, 2018: Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.*, **73**, 1–15, https://doi.org/10.1016/j.dsp.2017.10.011.

Olah, C., A. Mordvintsev, and L. Schubert, 2017: Feature visualization. *Distill*, **2**, e7, https://doi.org/10.23915/distill.00007.

——, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev, 2018: The building blocks of interpretability. *Distill*, **3**, e10, https://doi.org/10.23915/distill.00010.

Peng, D., Y. Zhang, and H. Guan, 2019: End-to-end change detection for high resolution satellite images using improved UNet++. *Remote Sens.*, **11**, 1382, https://doi.org/10.3390/rs11111382.

Raghu, M., and E. Schmidt, 2020: A survey of deep learning for scientific discovery. arXiv, https://arxiv.org/abs/2003.11755.

Ramos-Pollán, R., N. de Posada, and M. A. G. López, 2011: Optimizing the area under the ROC curve in multilayer perceptron-based classifiers. *Proc. Third Int. Conf. on Future Computational Technologies and Applications*, Rome, Italy, IARIA, 75–81.

Reichstein, M., G. Camps-Valls, B. Stevens, M. Jung, J. Denzler, N. Carvalhais, and Prabhat, 2019: Deep learning and process understanding for data-driven Earth system science. *Nature*, **566**, 195–204, https://doi.org/10.1038/s41586-019-0912-1.

Roebber, P. J., 2009: Visualizing multiple measures of forecast quality. *Wea. Forecasting*, **24**, 601–608, https://doi.org/10.1175/2008WAF2222159.1.

Sahiner, B., X. He, W. Chen, H.-P. Chan, L. Hadjiiski, and N. Petrick, 2013: Neural network training by maximization of the area under the ROC curve: Application to characterization of masses on breast ultrasound as malignant or benign. *Proc. SPIE*, **8670**, 86701M, https://doi.org/10.1117/12.2007615.

Samek, W., 2019: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Vol. 11700. Springer Nature, 435 pp.

Schmit, T. J., P. Griffith, M. M. Gunshor, J. M. Daniels, S. J. Goodman, and W. J. Lebair, 2017: A closer look at the ABI on the GOES-R series. *Bull. Amer. Meteor. Soc.*, **98**, 681–698, https://doi.org/10.1175/BAMS-D-15-00230.1.

Schrimpf, M., and Coauthors, 2018: Brain-score: Which artificial neural network for object recognition is most brain-like? bioRxiv, www.biorxiv.org/content/10.1101/407007v1.

Selvaraju, R. R., M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, 2017: Grad-CAM: Visual explanations from deep networks via gradient-based localization. *Proc. IEEE Int. Conf. on Computer Vision*, Venice, Italy, IEEE, 618–626, https://doi.org/10.1109/ICCV.2017.74.

Simonyan, K., A. Vedaldi, and A. Zisserman, 2013: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv, https://arxiv.org/abs/1312.6034.

Smilkov, D., N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, 2017: SmoothGrad: Removing noise by adding noise. arXiv, https://arxiv.org/abs/1706.03825.

Smith, T. M., and Coauthors, 2016: Multi-Radar Multi-Sensor (MRMS) severe weather and aviation products: Initial operating capabilities. *Bull. Amer. Meteor. Soc.*, **97**, 1617–1630, https://doi.org/10.1175/BAMS-D-14-00173.1.

Snow, M., and Coauthors, 2016: Monge's optimal transport distance for image classification. arXiv, https://arxiv.org/abs/1612.00181.

Sønderby, C. K., and Coauthors, 2020: MetNet: A neural weather model for precipitation forecasting. arXiv, https://arxiv.org/abs/2003.12140.

Stengel, K., A. Glaws, D. Hettinger, and R. N. King, 2020: Adversarial super-resolution of climatological wind and solar data. *Proc. Natl. Acad. Sci. USA*, **117**, 16 805–16 815, https://doi.org/10.1073/pnas.1918964117.

Toms, B. A., E. A. Barnes, and I. Ebert-Uphoff, 2020: Physically interpretable neural networks for the geosciences: Applications to Earth system variability. *J. Adv. Model. Earth Syst.*, **12**, e2019MS002002, https://doi.org/10.1029/2019MS002002.

Tsagkatakis, G., A. Aidini, K. Fotiadou, M. Giannopoulos, A. Pentari, and P. Tsakalides, 2019: Survey of deep-learning approaches for remote sensing observation enhancement. *Sensors*, **19**, 3929, https://doi.org/10.3390/s19183929.

Wang, Z., A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, 2004: Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, **13**, 600–612, https://doi.org/10.1109/TIP.2003.819861.

Weyn, J. A., D. R. Durran, and R. Caruana, 2020: Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *J. Adv. Model. Earth Syst.*, **12**, e2020MS002109, https://doi.org/10.1029/2020MS002109.

Willard, J., X. Jia, S. Xu, M. Steinbach, and V. Kumar, 2020: Integrating physics-based modeling with machine learning: A survey. arXiv, https://arxiv.org/abs/2003.04919.

Wimmers, A., C. Velden, and J. H. Cossuth, 2019: Using deep learning to estimate tropical cyclone intensity from satellite passive microwave imagery. *Mon. Wea. Rev.*, **147**, 2261–2282, https://doi.org/10.1175/MWR-D-18-0391.1.

Xu, C., and B. Zhao, 2018: Satellite image spoofing: Creating remote sensing dataset with generative adversarial networks. *10th Int. Conf. on Geographic Information Science*, Melbourne, Australia, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 67, https://drops.dagstuhl.de/opus/volltexte/2018/9395/pdf/LIPIcs-GISCIENCE-2018-67.pdf.