Signal, Image and Video Processing (2019) 13:541–549
https://doi.org/10.1007/s11760-018-1380-z

ORIGINAL PAPER

Deep feature extraction and its application for hailstorm detection in a large collection of radar images

Iksha Gurung[1] · Chao Peng[1] · Manil Maskey[2] · Rahul Ramachandran[2]

Received: 14 April 2018 / Revised: 21 August 2018 / Accepted: 16 October 2018 / Published online: 2 November 2018

Abstract
With the improvement of sensing and storing technologies, a large amount of weather data become available, and the data size will continue growing as radar imaging instruments continuously acquire data. In this work, we develop a deep convolutional neural network with a large collection of radar images as input to train and validate a classification model, and then we use the model to detect hailstorm events. This is interdisciplinary work between the disciplines of computer science and meteorology. We are primarily interested in what hailstorm features the network learns and how it learns as convolving into deeper iterations. The evaluation results show a high classification accuracy in comparison with existing hailstorm detection approaches. The proposed approach can also be used to detect other types of severe weather events with minimal efforts on variable or parameter changes.

Keywords Hailstorm detection · Convolutional neural network · Deep feature extraction

1 Introduction

Radar systems are commonly used to acquire data and measure moving objects for applications such as surveillance, weather analytics, and traffic controls. Data acquired by a radar system are stored as images, which are accumulated as new data is acquired. As a result, the data become a large collection of radar images. Our goal is to discover patterns or correlations (features) that associate to an event in radar images, and then use them for current data in order to predict the occurrence of a similar event. Detecting an event from radar images presents an exceptional challenge because of the large data volume and high complexity to interpret.

In this work, we propose a convolution neural network (CNN) to extract deep features in radar images for severe weather detection. The CNN used in this work is motivated by Krizhevsky et al. [14]. Severe weather detection is a typical research domain that uses radar images as the primary input data source. In particular, we are interested in hailstorm detection using a large collection of radar images that carry space–time precipitation information. A hailstorm is a type of thunderstorms producing small ice balls (hails), which may hurt people, damage buildings, cause small aircrafts to crash, etc.

Existing hailstorm detection approaches use a variety of meteorological data collected from multiple sources such as cloud-top temperatures, severe hail indices, convective available potential energy. A hailstorm expert has to manually tune the threshold values of those variables. Data collected from multiple sources may be in different types, and they are usually obtained at different time intervals. Also, for a given event, some types of data are available while others may be not or only available at a different time points. Thus, such data formatting issues make hailstorm detection with existing approaches a laborious task.

Contribution In this work, the CNN approach learns features automatically from radar images with little human

This research was supported by NASA Grant NNM11AA01A.

✉ Chao Peng
chao.peng@uah.edu

Iksha Gurung
ig0004@uah.edu

Manil Maskey
manil.maskey@nasa.gov

Rahul Ramachandran
rahul.ramachandran@nasa.gov

[1] University of Alabama in Huntsville, Huntsville, USA

[2] National Aeronautics and Space Administration (NASA), Marshall Space Flight Center, Huntsville, USA

interference and provides higher accuracy in comparison with the existing approaches. We visualize and analyze different CNN layers and discover a unique feature called a *nested looping pattern*. This pattern was not discussed in previous hailstorm detection research. We evaluate three activation functions and two pooling operations, which are crucial components in a CNN, with a concentration on their impact on the network's training performance and classification accuracy for hailstorm detection. This work is interdisciplinary and requires the knowledge of computer science and meteorology. To our knowledge, this is the first work that adopts a CNN for hailstorm detection.

The rest of the paper is organized as follows: Sect. 2 reviews some existing approaches for hailstorm detection. We describe our methodology with the CNN in Sect. 3. Data preprocessing and the experimental design are presented in Sect. 4. Section 5 describes the evaluation results. Section 6 concludes our work and discusses the future work.

2 Related work

In the past, approaches with thresholding, network, and machine learning algorithms were proposed to detect hailstorms. Auer [3] used a combination of radar reflectivity at S-band and cloud-top temperatures to detect hails. Bauer-Messmer et al. [4] used satellite data, and a hailstorm can be predicted based on the visual threshold place on metostat data. Witt et al. [16] proposed a Bayesian neural network with S-band NEXRAD radar data and temperature profiles to define a severe hail index.

Marzban et al. [16] developed two neural networks to predict the hail size and classify for hailstorm occurrence. The features used in the neural network were parameters such as VIL, severe hail index, storm top divergence derived from doppler radar as well as some environmental parameters calculated using numerical models such as vertically integrated wet-bulb temperatures. Ravinder et al. [26] presented a k-means clustering algorithm for visual RGB satellite images to obtain cloud textures, and then they applied the haar wavelet transformation to transform the textures with wavelength to detect hails. Merino et al. [17] developed a two-phase hail-detection tool based on logistic regression models, in which multi-threshold techniques were adopted. Shen et al. [30] utilized CNNs for contour detections in natural images using a new loss function. Their approach can detect features similar to those in our application, but it does not consider feature coherence in time-varying sequences.

In recent years, researchers have proposed detection algorithms using microwave soundings. Ferraro et al. [7] developed a simple threshold algorithm utilizing advanced microwave sounding unit and the storm reports from Storm Prediction Center for hailstorm detection. Mroz et al. [18] presented a threshold method for Ku-band reflectivity and evaluated the classification accuracy at different threshold values.

3 Methodology

We preprocess radar images from a storm event database and crop out image regions that contain historical hailstorm events (see the details in Sect. 4.1). Those cropped regions are the input of the network. The network utilizes the CNN layers similar to those in [15,25]. We followed the network design methodology in [14]. Training a CNN is an iterative process that progresses neurons during the forward pass, and updates parameters (e.g., the weights of convolutional filters) during the backward pass, until the loss calculated at an iteration during the forward pass is minimized. As shown in Fig. 1, the network is composed of five convolutional ($conv.$) layers, four pooling ($pool.$) layers, four normalization ($norm.$) layers, three fully connected (FC) layers. More technical details can be found in [14].

A convolutional layer extracts features by using learnable filters across the *feature maps* produced from the previous layer. Filters are randomly initialized using Gaussian noise. The output feature map of each filter is stacked on top of each other to form a three-dimensional output, whose depth is the total number of filters used in the layer. Thus, each feature element (or a neuron) can be represented as $x \in \mathbb{R}^{d\times d\times m}$, where $d\times d$ is the size of the feature map, and m is the number of filters. Given the feature map stacks of the previous layer as input, to obtain a neuron $y_{(i,j)}$ of the current layer, we use the following equation:

$$y^k_{(i,j)} = \left(\sum_{r=0}^{m^{l-1}-1} \sum_{s=0}^{n-1} \sum_{t=0}^{n-1} w^k_{(s,t)} \cdot x_{(i-n/2+s,\, j-n/2+t,\, r)} \right) + b^k \tag{1}$$

Equation 1 originates from the convolution theorem [5], where $i, j \in [0, d)$ and they represent the neuron index in the feature map of the previous layer; $k \in [0, m^l)$ represents the kth filter in the current layer l; n defines the size of the kth filter; $w^k_{s,t}$ is the weight value at the index (s, t) in the kth filter; m^{l-1} represents the number of filters in the previous

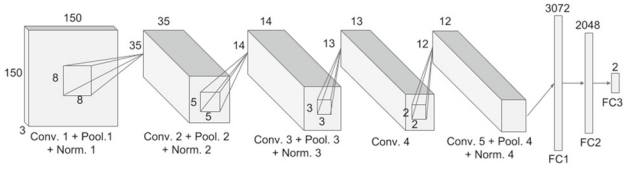

Fig. 1 The design of our network

layer ($l-1$), and it corresponds to the depth of the input feature map stack; b^k is the bias of the kth filter. When $l=0$ (the first convolutional layer), the outermost sum over the depth is ignored. Also, in each convolutional layer, the values of neurons are transformed with a nonlinear activation function. Common activation functions include *sigmoid* [24], *rectified linear unit* (ReLU) [6,19] and *tanh* [12,13]. We studied those activation functions for hailstorm detection (see Sect. 5.2). We eventually chose ReLU as the activation function in the convolutional layer.

A pooling layer uses the max pooling function to downsample the feature maps. The max pooling function selects the maximum neuron value in the pooling region (defined by the size of a pooling filter) and disregards others. Literature has shown the debate about overlapped pooling or non-overlapping pooling [6,14,27,29]. We studied overlapping and non-overlapping pooling operations for hailstorm detection (see Sect. 5.2). We eventually chose the overlapping pooling operation in the pooling layer.

The normalization layers use the local response normalization method [14]. The first fully connected layer flattens the output of the last normalization layer into a single vector. A dropout method [32] is used in the first and second fully connected layers. The third fully connected layer produces two probabilities in correspondence to "Hail" and "No Hail" classes.

4 Experimental design

The network was developed with Google's TensorFlow library [2] accelerated with Nvidia CUDA parallel processing platform. The program was written using Python. The training experiment was performed on a Nvidia's Tesla P100-PCIE GPU device with 12 GB memory.

The experiment consists of three scenarios: training, validation, and testing. Each scenario is assigned with a subset of original radar images. This section first discusses the preprocessing method that crops out the image regions corresponding to the historical hailstorm events. Then, this section discusses the training, validation, and testing scenarios in detail.

4.1 Preprocessing

The storm event database from National Centers for Environmental Information (NCEI) [1] maintains the records of historical storm events. The records were created based on the reports from officials, news, and other trusty sources. Each event consists of the event location (latitude and longitude), date and time, event type, etc. In our experiment, we used hailstorm events recorded from year 2006 to 2016, with an average of 17,000 hailstorm examples available per year.

We used the records of events to locate the hailstorms on NEXRAD images available from Iowa Environmental Mesonet [11]. We cropped the event from the image into the size of 150×150 pixels. The center of the cropped image is at the event's latitude and longitude. It is possible that the records archived multiple reports for a single event. Thus, we considered that the cropped images overlapping within the 150×150 pixel vicinity are duplications of the same event. NEXRAD images were collected by radar instruments every 5 min. We cannot always refer an event to the image with the exact time. We therefore cropped the image within a (± 1) minute range of the event time. The total time to download and crop images is about 5 h.

We also cropped the images that do not have hails. To do this, we randomly selected images and randomly located a cropping place on the images. We labeled them as "No hail" images, if they do not overlap with any portion of a "Hail" image.

As a result, we obtained a total of 133,421 cropped images, which consumes 3.3 GB of memory. Note that the original full-size radar images require a total of 136 GB for storage. The cropped images were divided into three datasets in a ratio of 7:2:1 for training, validation, and testing, respectively. Table 1 lists the sizes of the image subsets we created.

4.2 Training

Due to the large number of input images and the large number of parameters to learn, the GPU does not have enough memory to load them within one iteration of training. We used batches with 600 images in each, so we obtained a total of 156 batches. One iteration of training loaded and processed the images of one batch. Thus, the network took 156 iterations to complete one epoch. As progressing through epochs, the network learned from the feature maps and updated the learnable parameters, as observed with progressively increased accuracy and gradually decreased loss. Tables 2 and 3 give the hyperparameter values used in the network.

We used a less number of filters in early layers. As convolving to later layers, we increased the number of filters and decreased the filter sizes. The *stride* parameter was used to provide neuron shifts in convolving steps. It affects the size of an output feature map. We also specified the *padding* value which adds zero-valued neurons around the input fea-

Table 1 Image sets for training, validation, and testing

Ground truth labels	Training	Validation	Testing	Total
Hail	38,813	12,486	7675	58,974
No Hail	54,580	14,199	5668	74,447
Total	98,393	26,685	13,343	133,421

Table 2 The configuration of hyperparameters in the convolutional layers for the experiment

Hyperparameters	Conv. 1	Conv. 2	Conv. 3	Conv. 4	Conv. 5
Input size	150 × 150	35 × 35	14 × 14	13 × 13	12 × 12
# of filters (m)	64	160	256	256	128
Filter size ($e \times e$)	8 × 8	5 × 5	3 × 3	2 × 2	2 × 2
Stride	2	2	1	1	1
Padding	0	0	1	0	0
# of learnable parameters	12,352	256,160	368,896	262,400	131,200
Output size ($d \times d$)	72 × 72	16 × 16	14 × 14	12 × 12	11 × 11

Table 3 The configuration of hyperparameters in the pooling layers for the experiment

Hyperparameters	Pool. 1	Pool. 2	Pool. 3	Pool. 4
Input size	72 × 72	16 × 16	14 × 14	11 × 11
Filter size ($e \times e$)	3 × 3	3 × 3	2 × 2	3 × 3
Stride	2	1	1	1
Padding	0	0	0	0
Output size ($\tilde{d} \times \tilde{d}$)	35 × 35	14 × 14	13 × 13	9 × 9

ture map. As a result, the size of a feature map in a layer ($d^l \times d^l$) can be calculated as $d^l = \frac{d^{l-1} - e + 2 * \text{padding}}{\text{stride}+1}$.

4.3 Validation

Validation checks the progress of learning in the epochs so that we can tune the hyperparameters to improve the training. We used 20% of the total images to validate the learning results of the network. We checked the accuracy after an epoch was completed. If the training accuracy increases but the validation accuracy is random or decreases, the overfitting occurs. When the validation accuracy increases and the loss computed from the training decreases, the overfitting issue is reduced. The CNN layers used for validation is the same as the one used for training.

4.4 Testing

We used 10% of the total dataset for testing. The test dataset was not presented in the network during the training or validation iterations. The testing used the best performing model after the network was done for learning. Our evaluation results were generated using the test dataset.

5 Evaluation results

The training of the network became stable after 65 epochs. The execution time of a single epoch was an average of 5 min. The training took about 4 h and 30 min. Figure 2 shows that the accuracy increases, and the loss decreases, as the number of epochs increases. We did not experience an overfitting issue because the loss value during the training continuously decreased until it converged to a minimum and did not diverge after that. The accuracies of training, validation, and testing are 86%, 84%, and 83%, respectively.

5.1 Classification accuracy

We used the confusion matrix to statistically analyze the accuracy of the trained classification model. For the test dataset, correctly classified test images are categorized in true positive (TP) and true negative (TN). Misclassified test images are categorized in false positive (FP) and false neg-

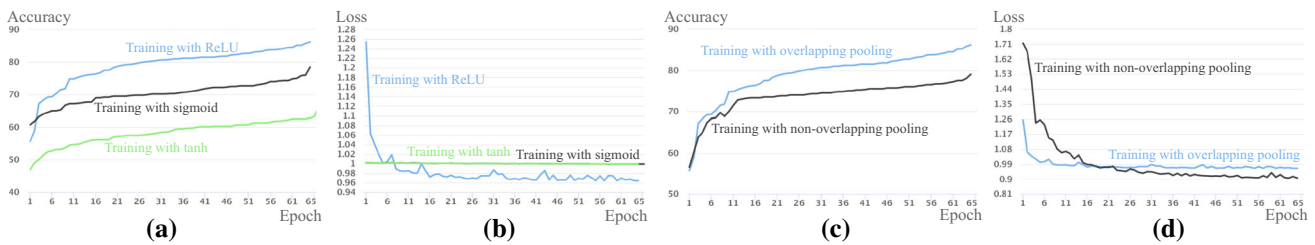

Fig. 2 Accuracy and loss comparisons. **a**, **b** Are the accuracy and loss comparisons of the network with ReLU, sigmoid and tanh activation functions. The overlapping pooling operation is used. **c**, **d** Are the accuracy and loss comparisons of the network with overlapping pooling and non-overlapping pooling. ReLU activation function is used

Table 4 The confusion matrix for "Hail" (positive) and "No hail" (negative) classification

	Predicted hail	Predicted no hail	Total
Actual Hail	6303 (TP)	1372 (FN)	7675
Actual No Hail	830 (FP)	4838 (TN)	5668
Total	7133	6210	13,343

ative (FN). The confusion matrix is shown in Table 4. We calculated the *precision*, *probability of detection* (POD), *false alarm ratio* (FAR), and *critical success index* (CSI). They can be expressed with the following equations:

$$\text{precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})}; \text{POD} = \frac{\text{TP}}{(\text{TP} + \text{FN})}; \quad (2)$$

$$\text{FAR} = \frac{\text{FP}}{(\text{TP+FP})}; \text{CSI} = \frac{\text{TP}}{(\text{TP} + \text{FP} + \text{FN})}; \quad (3)$$

POD is sensitive to FN. If the evaluation considers only POD, the trained classification model may over-predict the occurrence of hailstorm events. FAR is sensitive to TP. If the evaluation considers only FAR, the model may under-predict the occurrence of hailstorm events. CSI implies the confidence level of using the model to detect events. According to the measure of weather forecasting used by the National Weather Service [8], the perfect model would have POD = 1, FAR = 0, and CSI = 1. The CSI can also be expressed as $\frac{1}{\frac{1}{1-\text{FAR}} + \frac{1}{\text{POD}-1}}$ [28]. To compare our approach to existing approaches, if exact CSI values are not provided in the existing papers, the function in [28] is used to compute CSI values.

As shown in Table 5, our approach achieved a high precision (0.821) and a high CSI (0.741). We observed that the approach with the highest CSI was developed by Auer [3] in the year 1994, but that approach requires cloud-top temperatures as an additional parameter. It is usually difficult to find a pair of cloud-top temperature and radar image at the same location and time. For example, a GOES satellite instrument usually produces a cloud-top temperature every 15 min [21]. A radar image is usually produced every 5 min. A hailstorm event usually lasts for a few minutes in duration. Having a 15-min data producing interval to capture cloud-top temperatures leads to a high chance of missing the entire event of a hailstorm. Data augmentation methods [23] such as image flipping, rotating, and scaling may not make up the features missed in uncaptured hailstorm events. Using radar images is a more general and practical method for hailstorm detection, and we will be less likely to miss any of hailstorms within a 5-min radar image producing interval. With the solid state of radar imagery recording and processing techniques, our approach can be used to implement a real-time system to detect hailstorms, without suffering the limitation of event missing due to the use of cloud-top temperature parameters.

Table 5 Comparison with prior approaches developed for hailstorm detection on evaluation terms of precision, POD, FAR, and CSI

Approaches			Precision	POD	FAR	CSI
Ours			0.821	0.884	0.116	0.741
ResNet [10]			0.778	0.897	0.103	0.714
VGG16 [31]			0.795	0.871	0.129	0.712
SVM (with Scikit-learn package [22])			0.779	0.782	0.160	0.640
Auer [3]	A regression model	without cloud-top temperature parameter	–	0.560	0.220	0.480
		with cloud-top temperature parameter	–	0.910	0.120	0.810
Bauer-Messmer and Waldvogel [4]	A three-step area thresholding method		–	0.840	0.660	0.319*
	Thresholding with the highest possible value		–	0.560	0.470	0.374*
Zibert and Zibert [33]	A min & max pixel detection algorithm		0.540	0.660	–	–
Merino14 et al. [17]	A hybrid approach with CM & HM algorithms		–	0.769	0.167	0.667*
Ferraro et al. [7]	BT thresholding		—	0.400	0.700	0.207*
Mroz et al. [18]	Thresholding with Ku-based parameters		–	0.653	0.410	0.449
Ni et al. [20]	Thresholding with developed hail-detection criteria		–	0.450	0.240	0.394*

For the prior approaches that do not provide the CSI, their CSI values are computed with the paper-provided POD and FAR values and marked with asterisks

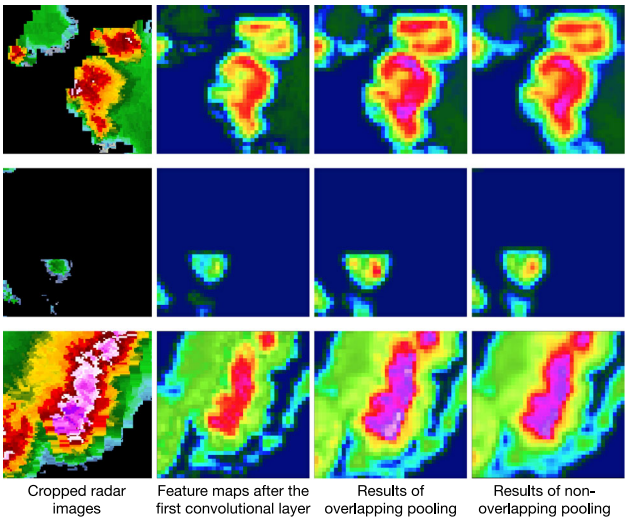

Fig. 3 Comparison of feature preservation between the use of overlapping pooling and the use of non-overlapping pooling with three input radar images. Each row (from left to right) shows the input radar image, the feature map generated from a filter in the first convolutional layer, and the results of the two different pooling operations. The radar image size is 150×150. The feature map size is 72×72. The pooling filter size is set to 3×3. The stride of overlapping pooling is set to 2. The size of the overlapping pooling result is 35×35. The size of the non-overlapping pooling result is 23×23. As shown in the last two columns, the feature regions are sharper after using overlapping pooling. The results from overlapping pooling have higher color contrasts and clearer details

Without adding cloud-top temperatures, the accuracy of Auer's approach becomes worse than ours (26.1% less than our CSI). Comparing with the recent approaches of Mroz et al. [18] and Ni et al. [20] in the year 2017, our approach results in an increase of 29.2% and 34.7% for CSI, respectively.

In addition, we implemented a ResNet network [10], a VGG16 network [31], and a nonlinear SVM [22] for hailstorm detection using the same dataset that we used in our CNN approach. The method of transfer learning based on the knowledge learned in our approach was used to implement ResNet and VGG16 networks. The ResNet network was composed of 177 layers, and the VGG16 was composed of 19 layers. The weights for those layers were kept intact. Three fully connected layers were added at the end of each network. For SVM, we chose the radial basis function (RBF) as the kernel function, and set the two RBF hyperparameters as 10^{-3} and 10^{3}, which are in favor of spacing their values far apart from each other. Their classification statistics are listed as part of Table 5. Our approach performed better than ResNet, VGG16, and SVM. This is evident by the CSI which is better than all other approaches aforementioned.

5.2 Evaluation of activation and pooling functions

We compared sigmoid, tanh, and ReLU activation functions. As shown in Fig. 2a, b, ReLU results in the highest accuracy and the lowest loss value. It preserves the receptive fields of neurons through the CNN layers. In contrast, sigmoid and tanh slow down the process of network learning. The effectiveness of ReLU is significant during early epochs. The loss with ReLU is reduced exponentially as the network convolves over epochs. To complete 65 epochs, the network took about 6 h with sigmoid, and about 7 h and 15 min with tanh; In comparison, the network took about 4 h and 30 min with ReLU.

We found that the use of the overlapping pooling operation results in a better accuracy. As shown in Fig. 2c, d, the network with overlapping pooling starts with a low accuracy, and then the accuracy increases logarithmically over epochs. With non-overlapping pooling, the network starts with a slightly higher accuracy, but over the epochs it does not increase as fast as the network with overlapping pooling. As regards to loss reduction, the network with overlapping pooling starts with a relatively small loss value, but it is not reduced much by the end. The network with non-overlapping pooling reduces the loss exponentially, and eventually results in a lower loss value. We also found that the overlapping pooling method preserves essential feature details when downsampling feature maps. Figure 3 shows the comparison of feature preservation between the use of overlapping pooling and the use of non-overlapping pooling. Also, the CSI of the network with overlapping pooling (0.741) is higher than the CSI with non-overlapping (0.639).

5.3 Hailstorm features

We wanted to understand what features the network learned during the training. For different convolutional layers, we used *Conviz* [9] to generate synthetic visualization of feature maps. As shown in Fig. 4, feature maps of each convolutional layer are visualized as a grid of images, where the number of images is the total number of filters in the layer. We noticed that neurons in feature maps appear to be blobby and smoothly crowding together in early layers (e.g., the first and second convolutional layers), and they tend to become compact and localized as the network convolves. Sometimes ReLU did not activate neurons due to the filters failing to extract features from the input, such as the examples shown in Fig. 4b, f, i, l, o, so the filters corresponding to those zero feature maps became "dead." However, the learning rate was decayed properly and progressively, so dead filters did not cause any training issue.

In Fig. 4a, c–e, g, m, n, we found that the network learned a *nested looping pattern* from the images. Figure 5 shows the example images that are correctly classified as "Hail." According to the domain knowledge of measuring precipitation information, a high value of decibels (dBZ) between 60–65 usually indicates hailstorms. We found that a clear nested looping patten has pink regions (> 60 dBZ) gathering

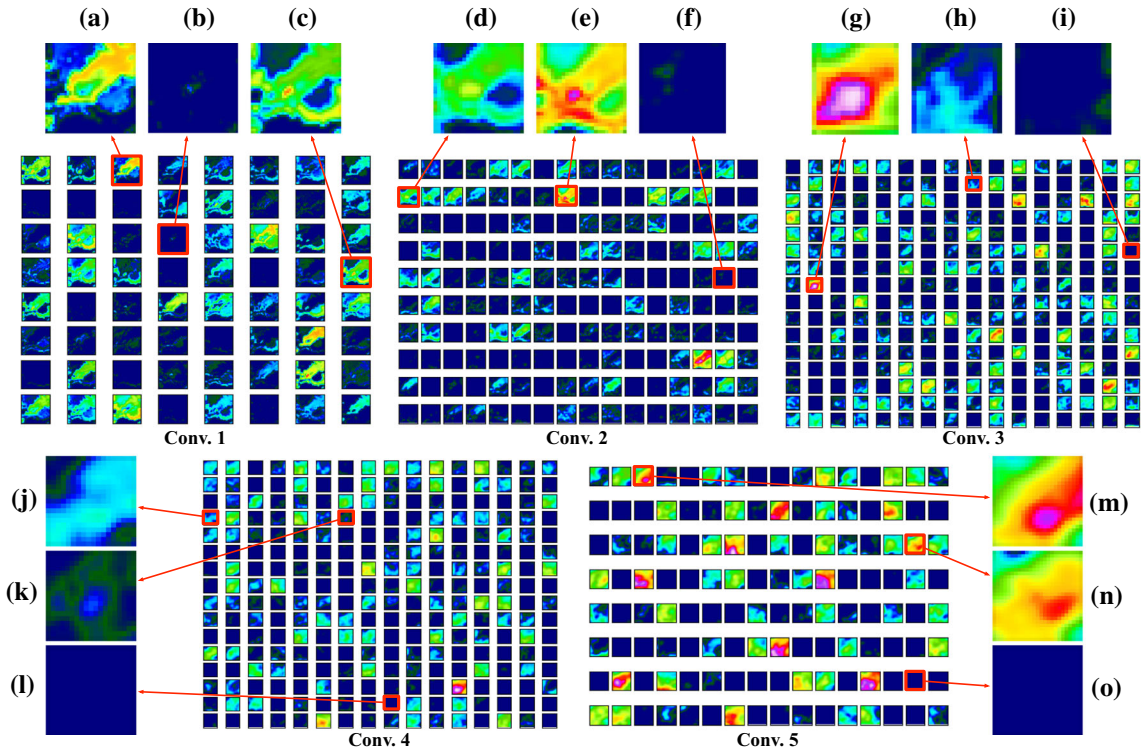

Fig. 4 Feature maps from each of the five convolution layers after the last (the 65th) epoch. The zoom-in images of **(a-o)** show details of the patterns learned by the network

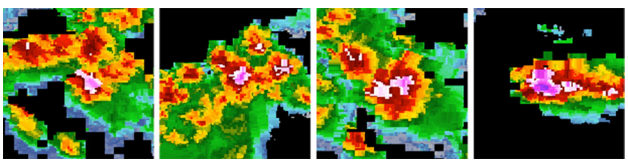

Fig. 5 An example that test images labeled as "Hail" are classified as "Hail" (TP) by our trained model

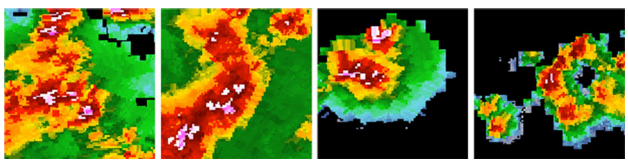

Fig. 6 Test images labeled as "No hail" are classified as "Hail" (FP) by our approach. Those misclassified images are mainly because of the dataset limitations described in Sect. 5.4

at the center of the pattern, and has red and yellow regions fading toward outer loops. This is a unique feature which was not discussed in the existing work.

5.4 Dataset limitations

The reports of hailstorm events we used for this experiment are from different officials or agencies. There are cases that a hailstorm (or a storm with strong involvement of hail features) at a location was reported as a thunderstorm instead. Because of that, "No hail" images used for training may contain hail features. Such cases also exist in the test dataset and may lead to misclassification. As shown in Fig. 6, the test images labeled as "No hail" are classified as "Hail" because hail (or hail-like) features exist in those images.

Also, there are cases that some reported events are extremely localized, so the hail features are only in a small region on the image. Such localized hailstorms usually last a very short period time. It is possible that high dBZ values were not captured by the radar instrument. That means it is possible that a reported "Hail" event may not contain hail features. This mismatch between the report and the actual image may lead to misclassification.

Furthermore, there are cases that the hailstorm and thunderstorm are reported at the same place and same time. We believe it is appropriate to label those images as "Hail" rather than "No hail" for training, because such a storm usually contains hail features and have already evolved to a hailstorm though it may not yet be reported as that. To be cautious, we checked the statuses of the reports in ± 3 min to find out if an actual hailstorm was observed for that storm.

6 Conclusion and future work

We have designed and evaluated a deep convolutional neural network for hailstorm detection. In each convolutional layer, we studied what features the network tries to learn. We discussed the effectiveness of training and classification when using different activation functions and different pooling methods in CNN layers. We provided a comprehensive evaluation of the trained classification model for hailstorm detection. With our approach, hailstorm detection at a given location can be done in seconds with little human interference. Our approach has the potential to be used to classify and detect other types of severe weather events. With the knowledge of current climates of a geographical region, our trained classification model can be used to predict the occurrence of a severe weather event.

The proposed approach will be suitable for the application of local severe weather forecasting. A 150×150 cropped image covers a 33.5×45.63 miles of a geographical region. We consider to increase the cropping size and evaluate if the accuracy could be improved. In this work, hyperparameter values are adjusted based on the intermediate validation results and the analysis of visualized feature maps. In the future, we plan to develop optimization algorithms to determine a suitable combination of hyperparameter values. The goal is to further reduce loss. We will research on optimization algorithms, such as Bayesian optimization algorithm in [15], to automate the hyperparameter optimization.

We will mix radar images with other types of input data, such as infrared images and visual images from satellites, to create multichannel input for network training. Our long-term goal is to predict hailstorms rather than detecting. Creating multichannel input will be the next step toward that goal. We will try to obtain more reliable data for training to eliminate the dataset limitations as described in Sect. 5.4. We will study a mix of hurricane, thunderstorm, tornado events with larger and more complex train and test datasets.

Acknowledgements This research work was supported by NASA Grant NNM11AA01A. We thank Dr. Sundar A. Christopher, Professor of Atmospheric Science at UAH, for his insightful suggestions for this work. We thank Ms. Melinda Pullman who helped us organize the data from National Center for Environmental Information Storm Events Database. We thank the support of Department of Computer Science at UAH and the support of NASA.

References

1. National centers for environmental information. https://www.ncei.noaa.gov (2017)
2. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I.J., Harp, A., Irving, G., Isard, M., Jia, Y., Józefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D.G., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P.A., Vanhoucke, V., Vasudevan, V., Viégas, F.B., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. CoRR. arXiv:1603.04467 (2016)
3. Auer Jr., A.H.: Hail recognition through the combined use of radar reflectivity and cloud-top temperatures. Mon. Weather Rev. **122**(9), 2218–2221 (1994)
4. Bauer-Messmer, B., Waldvogel, A.: Satellite data based detection and prediction of hail. Atmos. Res. **43**(3), 217–231 (1997)
5. Bracewell, R.N.: The Fourier Transform and its Applications. McGraw-Hill Series in Electrical Engineering, Networks and Systems, 2 rev. edn, p. c1986. McGraw-Hill, New York (1986)
6. Chen, M., Shi, X., Zhang, Y., Wu, D., Guizani, M.: Deep features learning for medical image analysis with convolutional autoencoder neural network. IEEE Trans. Big Data **PP**(99), 1–1 (2017)
7. Ferraro, R., Beauchamp, J., Cecil, D., Heymsfield, G.: A prototype hail detection algorithm and hail climatology developed with the advanced microwave sounding unit (AMSU). Atmos. Res. **163**(Supplement C), 24–35 (2015)
8. Gerapetritis, H., Pelissier, J.M.: On the behavior of the critical success index. US Department of Commerce, National Oceanic and Atmospheric Administration, National Weather Service (2004)
9. Gryshkevych, S.: Conviz. https://github.com/grishasergei/conviz (2016)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
11. Herzmann, D., Arritt, R., Todey, D.: Iowa environmental mesonet. http://mesonet.agron.iastate.edu/request/coop/fe.phtml. Verified 27 Sept 2005. Iowa State University, Department of Agronomy, Ames, IA (2004)
12. Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al.: What is the best multi-stage architecture for object recognition? In: 2009 IEEE 12th International Conference on Computer Vision. IEEE, pp. 2146–2153 (2009)
13. Klein, B., Wolf, L., Afek, Y.: A dynamic convolutional layer for short range weather prediction. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4840–4848 (2015)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc, Red Hook (2012)
15. Liu, Y., Racah, E., Prabhat, Correa, J., Khosrowshahi, A., Lavers, D., Kunkel, K., Wehner, M., Collins, W.D.: Application of deep convolutional neural networks for detecting extreme weather in climate datasets. CoRR. arXiv:1605.01156 (2016)
16. Marzban, C., Witt, A.: A bayesian neural network for severe-hail size prediction. Weather Forecast. **16**(5), 600–610 (2001)
17. Merino, A., López, L., Sánchez, J., García-Ortega, E., Cattani, E., Levizzani, V.: Daytime identification of summer hailstorm cells from msg data. Nat. Hazards Earth Syst. Sci. **14**(4), 1017–1033 (2014)
18. Mroz, K., Battaglia, A., Lang, T.J., Cecil, D.J., Tanelli, S., Tridon, F.: Hail-detection algorithm for the gpm core observatory satellite sensors. J. Appl. Meteorol. Climatol. **56**(7), 1939–1957 (2017)
19. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10. Omnipress, USA, pp. 807–814 (2010)
20. Ni, X., Liu, C., Cecil, D.J., Zhang, Q.: On the detection of hail using satellite passive microwave radiometers and precipitation radar. J. Appl. Meteorol. Climatol. **56**(10), 2693–2709 (2017)

21. Oceanic, N., Administration, A.: Goes-R cloud top temperature (2016). https://vlab.ncep.noaa.gov/web/goes-r-end-user-mission-readiness-project/cloud-top-temperature
22. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
23. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. CoRR. arXiv:1712.04621 (2017)
24. Poria, S., Cambria, E., Gelbukh, A.: Aspect extraction for opinion mining with a deep convolutional neural network. Knowl. Based Syst. **108**(C), 42–49 (2016)
25. Pradhan, R., Aygun, R., Maskey, M., Ramachandran, R., Cecil, D.: Tropical cyclone intensity estimation using a deep convolutional neural network. IEEE Trans. Image Process. **PP**(99), 1–1 (2017)
26. Ravinder, A., Reddy, P.K., Prasad, N.: Detection of wavelengths for hail identification using satellite imagery of clouds. In: 2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks (CICSyN). IEEE, pp. 205–211 (2013)
27. Sainath, T.N., Mohamed, A-r., Kingsbury, B., Ramabhadran, B.: Deep convolutional neural networks for LVCSR. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 8614–8618 (2013)
28. Schaefer, J.T.: The critical success index as an indicator of warning skill. Weather Forecast. **5**(4), 570–575 (1990)
29. Scherer, D., Müller, A., Behnke, S.: Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition, pp. 92–101. Springer, Berlin, Heidelberg (2010)
30. Shen, W., Wang, X., Wang, Y., Bai, X., Zhang, Z.: Deepcontour: a deep convolutional feature learned by positive-sharing loss for contour detection. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3982–3991 (2015). https://doi.org/10.1109/CVPR.2015.7299024
31. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR. arXiv:1409.1556 (2014)
32. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
33. Žibert, M.I., Žibert, J.: Monitoring and automatic detection of the cold-ring patterns atop deep convective clouds using meteosat data. Atmos. Res. **123**, 281–292 (2013)