Deep Network Ice Crystal Classification Using Spatial Pyramid Pooling For Inconsistent Image Dimensions

Riley Conlin and Dr. Anne Denton
Department of Computer Science
North Dakota State University
Fargo, 58105
riley.conlin@ndsu.edu

Abstract

The process of ice crystal classification is currently hindered by the manual effort required to classify each image into one category. This paper looks at a way to classify images of ice crystals into one of four categories using convolutional neural networks. The dataset given has several issues with it, primarily being that each ice crystal image has a different dimension. Fully-connected layers of neural networks require the image input to all share the same dimensions and rather than further compromise the dataset by cropping or distorting it, a process called spatial pyramid pooling is applied to standardize the size. This process is done in-between the convolutional layers of the CNN and the fully-connected layers. The results gained showed that from a 40-image training set it could correctly get over 50% accuracy for 2 of the categories for training images. This shows potential for the dataset and the process of spatial pyramid pooling though additional work and a larger dataset is required to draw a more stable conclusion.

1 Introduction

Ice crystal classification is a potentially useful and informative practice for meteorology and meteorological prediction, such as determining cloud temperature. The primary roadblocks to widespread practice of classification is the expense of obtaining imagery and the effort required to manually classify the images. This paper approaches the topic of classifying these ice crystal images by applying deep neural network learning processes to them. It also deals in implementing a spatial pyramid pooling layer to allow training of images of variable dimensions without cropping or distortion.

2 Method

2.1 Dataset

The received dataset is a series of images of ice crystals. These images are categorized by the type of crystal they are. The categories are: Bullets/Bullet Rosettes, Columns, Dendrites, and Plates. However, these images are not the ice crystals themselves but are rather sheets of sub-images, those being the ice crystals. All these ice crystal sub-images are of variable dimensions and locations. Alongside them are chunks of text that exist both in the whitespace surrounding the sub-images as well as some that lie in the lower left-hand corner of the sub-images. An example of an image sheet is shown in Figure 1.

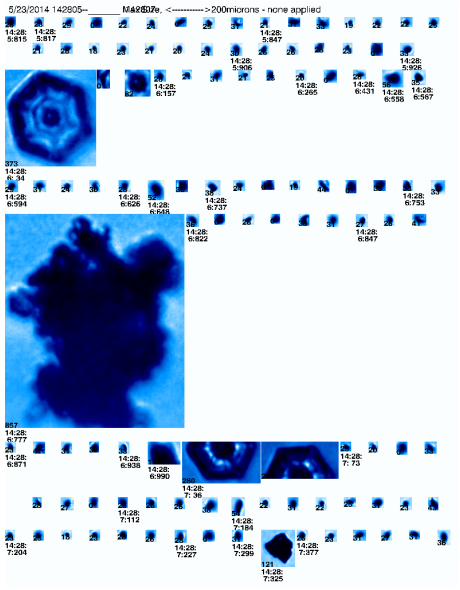

Figure 1: Plate image sheet (not full size).

The sub-images are not necessarily whole ice crystals and the larger ones are more often than not partials. To prevent unnecessary whitespace and text from being extraneous data that would throw off a neural network, the sub-images were separated into their own individual images to be used. The new images are shown in Figure 2. A total of 40 images evenly balanced between categories was manually compiled by cropping images in Paint.net. These 40 images constitute the training set used.

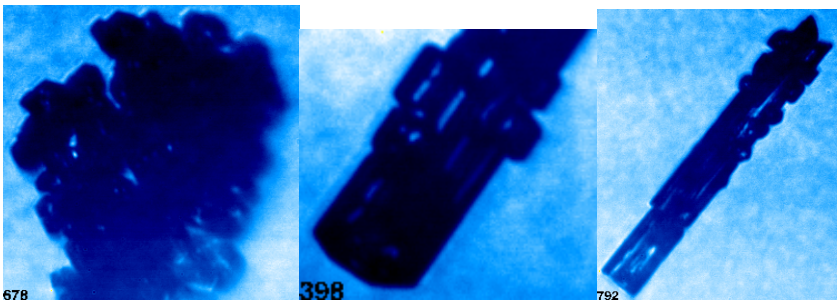

Figure 2: Three separated Column ice crystals (not full size).

2.2 Neural Network

The convolutional neural network (CNN) was constructed using the Tensorflow library for Python 3. It further implements Keras for the basic network structure. The first problem encountered when attempting to use these images together is that they are not guaranteed to share the same dimensions. This means the fully-connected layers won't work. However, the base convolution and pooling layers prior to it do not require images of the same dimension to function. Prior to being put through the network, the images are converted to grayscale because color is not important to classification whereas the structure of the ice crystal is. The constructed CNN is a basic seven-layer model. The first layer is a convolutional layer of pool size 2 and stride size of 2, the second is a max pool with pool size of 2x2, next is another convolutional layer of pool size 4 and stride size of 2, followed by another max pool with pool size of 2x2. The final layers condense the multi-dimensional tensor into a single dimension shape before it is then compressed again into the fully-connected softmax layer with 4 outputs, one for each category. In order to get multi-dimensional images in, a spatial pyramid pooling layer is added after the last max pooling layer but before the dense layers.

2.2.1 Spatial Pyramid Pooling

As described in *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition* [1], spatial pyramid pooling is a type of pooling layer that claims to enhance the Bag-of-Words model's ability to generate the fixed-length vectors required for the fully-connected layers by "maintain[ing] spatial information by pooling in local spatial bins." By sorting into bins of proportional size, the number of bins remain constant, so the supplied vector has a fixed size in the third dimension. Figure 3 illustrates the layer. The pyramid chosen for the layer is of shape 1x1, 2x2, 3x3, 4x4, for a total fixed-length representation of 30. This shape allows for a decent amount of data accuracy to be maintained which is especially useful for the smaller images.

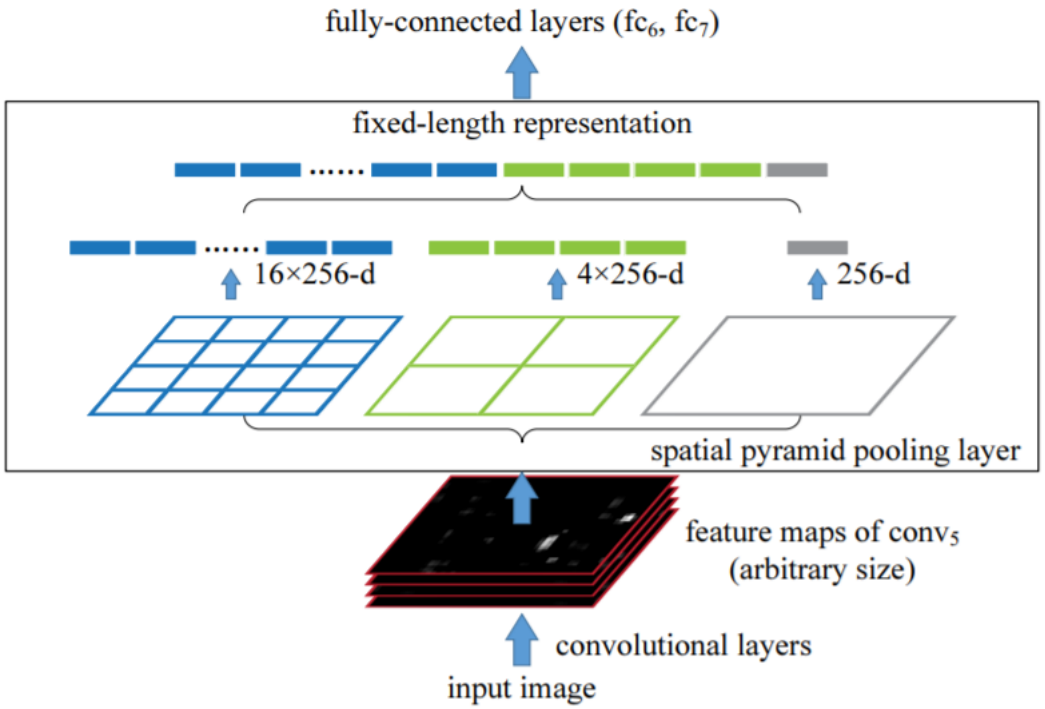

Figure 3: Spatial pyramid pooling layer where 256 is the filter number of $conv_5$. Image credit to *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition* [1].

The current working code base is available on a GitHub repository linked in the references [3].

3 Results

Due to lack of input data and the difficulty of working with the dataset, the training data was tested against itself. One trained model after several epochs had achieved over 50% accuracy in identifying Bullets/Bullet Rosettes and Dendrites. The full test results are listed in Table 1. This shows there is potential in using the dataset and the training methods though often it would be extreme in either giving 100% confidence in one singular category or by giving an equal 25% chance to each category. Both situations would result in that categorization being applied to all the test data. Based on observations during training, it appeared that under-trained models would result in the 100% confidence result, where over-trained models result in the 25% confidence evenly distributed.

Overall the results indicate further tuning of the network and pyramid structure could result in a higher level of accuracy and plausibly over 50% accuracy for all categories rather than just for half. It is also possible that it is overly difficult a task for this CNN to tell the difference between several of the categories. More data or a simpler way to

remove the sub-images from the image sheets would be required to draw a more stable conclusion, though this implies a good start.

Category	Correct	Incorrect	Accuracy
Bullets/Bullet Rosettes	6	4	60%
Columns	3	7	30%
Dendrites	7	3	70%
Plates	3	7	30%
Total	19	21	47.5%

Table 1: Results given when testing training data.

4 Future Work

Efforts will be made towards programmatically extracting the sub-images from the image sheets so more data can be used in training the network. Mask R-CNN [2] will be looked at first for that purpose. *Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition* also claims to be capable of equal accuracy in object detection so there is another possibility. There may also be some value in pursuing a way to remove the text in the lower left-hand corner of the images without compromising the integrity of the image, as that text takes up a large portion of the image on the smaller-sized images and likely interferes with the neural network's ability to see the ice crystal's structure.

5 Acknowledgements

This material is based upon work supported by the National Science Foundation through grant IIA-1355466.

References

[1] Kaiming He et al. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *arXiv*: arXiv:1406.4729v4 [cs.CV], April 2015
[2] Kaiming He et al. Mask R-CNN. *arXiv*: arXiv:1703.06870v3 [cs.CV], Jan 2018.
[3] Code Repository. https://github.com/topoftheyear/snowflake-classification