Utilization of Convolutional Neural Networks in the Classification of Snowflakes Based on Images by a Multi-Angle Snowflake Camera

Adam Hicks

Advisor: Branislav M. Notaroš

Committee Members:
Christine Chiu
Ali Pezeshki
Overview of Snowflake Classification

- Motivation
  - Different classes of snowflake may display unique scattering properties
  - Different snowflake formations are indicative of specific environmental conditions
  - “Automatic” classification is crucial for processing large data sets (storms) and supporting in-situ measurements

- Background
  - Multi-Angle Snowflake Camera (MASC)
  - Previous work in Snowflake Classification

- Set up
  - Data Processing
  - Classification Schemes
    - Geometric
    - Rimming Degree Estimation

- Method
  - Back-Propagation Neural Network Algorithm
  - Convolutional Neural Networks (CNNs)
    - ResNet-50

- Software Implementation

- Results
  - Geometric Classification
  - Rimming Degree Estimation

- Future Work
Why Classify Snowflakes?

- **Multi-Angle Snowflake Camera and Visual Hull**
  - MASC captures snowflakes in freefall for 3D reconstruction
    - Able to determine scattering parameters based on particle fall-speed and volume
    - Snowflake features are directly related to scattering parameter calculations and environmental conditions
    - Hundreds of thousands of flakes in a given storm
MASCRAD site
Multi-Angle Snowflake Camera

- Simultaneous acquisitions of 5 high resolution images and fall-speed
- 2Hz acquisition rate
- 30cm² measurement area
- 50µm resolution
- Particle-by-particle electromagnetic scattering analysis
Previous work in Snowflake Classification

- **Backpropagation Neural Network**
  - Feind used this method on data taken from Oklahoma at S.D.S.o.M.[1]
  - Achieved an overall accuracy of 85%
  - Minimum distance and Fuzzy logic methods were considered with inferior results

- Determined 8 geometric classes based on 7 features

- Training set of 2000 images

- Better performance possible, limited data through capture method
Previous work in Snowflake Classification

- **Multinomial Logistic Regression (MLR)**
  - Praz et al. used this method in the Swiss Alps [2]
  - Achieved an overall accuracy of 94.7%
- Determined 10 geometric classes based on 72 features
- Training set of 3500 images
- Classified degree of riming per snowflake
- Seems to require a lot of overhead to implement
Set Up

- **Threshold Filtering**
  - Removes dim or empty images

- **Image Cropping**
  - Separates multi-flake images

- **Resize images for network input**
Set Up - Classification Schemes

- **Geometric Classification Scheme**
  - 6 classes defined per Praz et al. [2] and Magono and Lee [3]
  - 5 classes are represented
  - Clearly identifiable images

- **Images taken during weather events in Greely Colorado on December 26th 2014 and February 22nd 2015.**

- **Total Data set includes 1453 images.**
Set Up - Classification Schemes

- **Riming Degree Estimation Scheme**
  - Physical descriptors defined by Mosimann et al. [4] are adapted to 3 numerical classes
  - Probability estimate $R_{l,c#}$ as output of classifier applies estimate to riming degree

- **Total Data set includes 300 images.**
Method – Back-Propagation Neural Networks

- Artificial Neural Networks (NN)
  - Attempts to model how a brain learns through neuron interactions expressed as an algorithm

- Back-Propagation Feed Forward networks are popular for classification
Method – Back-Propagation Neural Networks

\[ u_k^{(1)}(p) = \sum_{j=1}^{N+1} w_{kj}^{(1)}(p)x_j(p), \forall k \in [1, K] \]
Method – Back-Propagation Neural Networks

\[ u_k^{(1)}(p) = \sum_{j=1}^{N+1} w_{kj}^{(1)}(p)x_j(p), \forall k \in [1, K] \]

\[ f(u) = \frac{1}{1 + \exp(-u)} \text{ or } \text{ReLU}(u) = \max(0, u) \]
Method – Back-Propagation Neural Networks

\[ u_k^{(1)}(p) = \sum_{j=1}^{N+1} w_{kj}^{(1)}(p)x_j(p), \forall k \in [1,K] \]

\[ f(u) = \frac{1}{1 + \exp(-u)} \text{ or } ReLU(u) = \max(0, u) \]

\[ o_k^{(1)}(p) = f\left(u_k^{(1)}(p)\right), \forall k \in [1,K] \]
Method – Back-Propagation Neural Networks

\[ u_k^{(1)}(p) = \sum_{j=1}^{N+1} w_{kj}^{(1)}(p)x_j(p), \forall k \in [1,K] \]

\[ f(u) = \frac{1}{1 + \exp(-u)} \text{ or } ReLU(u) = \max(0, u) \]

\[ o_k^{(1)}(p) = f\left(u_k^{(1)}(p)\right), \forall k \in [1,K] \]

\[ E_p = \frac{1}{2} \sum_{m=1}^{M} \left(d_m^{\text{output}}(p) - o_m^{\text{output}}(p)\right)^2 \]

\( H = \text{number of layers} \)
Method – Back-Propagation Neural Networks

\[ E_p = \frac{1}{2} \sum_{m=1}^{M} \left( d_m^{(output)}(p) - o_m^{output}(p) \right)^2 \]
Method – Back-Propagation Neural Networks

\[ E_p = \frac{1}{2} \sum_{m=1}^{M} \left( d_m^{(\text{output})}(p) - o_m^{\text{output}}(p) \right)^2 \]

\[ \Delta w_{ml}^{(\text{output})}(p) = -\mu \nabla E_p(p) = \mu e_m^{\text{output}}(p) f'(u_m^{(\text{output})}(p)) o_i^{H-1}(p), \]

\[ e_m^{\text{output}}(p) = d_m^{(\text{output})}(p) - o_m^{\text{output}}(p) \]

\[ \mu = \text{learning factor} \]
Method – Back-Propagation Neural Networks

\[
\Delta w_{ml}^{(output)} = \mu e_m^{output}(p)f'(u_m^{(output)}(p))o_l^{H-1}(p),
\]

\[
\Delta w_{lk}^{(H-1)} = \mu f'(u_l^{(H-1)}(p))o_k^1(p) \sum_{m=1}^M w_{ml}^{(output)}(p)e_m^{output}(p)f'(u_m^{(output)}(p)),
\]

\[
\Delta w_{kj}^{(1)} = \mu f'(u_k^{(1)}(p))x_j(p) \sum_{l=1}^L w_{lk}^{(H-1)}(p)f'(u_l^{(H-1)}(p)) \left[ \sum_{m=1}^M w_{ml}^{(output)}(p)e_m^{output}(p)f'(u_m^{(output)}(p)) \right]
\]
**Convolutional Neural Networks**

- **Convolutional Layer**
  - Introduces sparse interaction and parameter sharing

- **Pooling Layer**
  - Introduces equivariance to translation

- **Fully Connected Layers**
  - Similar to BPNN
  - Softmax operator for high level decision making
Convolutional Neural Networks

**Pros**
- Ideal for Image Classification
- Automatic feature extraction with minimal preprocessing of images required
- Can be retrained to different applications

**Con**
- Requires Large Training Sets
Convolutional Neural Networks

- **Pretraining**
  - Due to translation invariance, CNNs can take advantage of pretraining on related data sets for similar tasks

- **ResNet-50**
  - Developed by Microsoft Research[5]
  - Balanced between speed and accuracy
  - Pretrained on ImageNet database[7]

*Image provided from Mathworks [6]. Shows accuracy vs speed relative to fastest network.*
Convolutional Neural Networks

- **ResNet-50**
  - Addresses accuracy saturation issues that occur with deep networks
  - Introduces an identity matrix to keep the network on track
Convolutional Neural Networks

- **Software Implementation**
  - Developed using Matlab™ 2018b with the Deep Learning and Machine Learning Toolboxes
  - "Dropout" is employed to reduce overfitting
  - Training and validation sets are created and random distortion is introduced
  - Learning rate is set at 0.0003
Results

Geometric Classification
## Results

### Geometric Classification

<table>
<thead>
<tr>
<th>Output Class</th>
<th>AG</th>
<th>CC</th>
<th>GR</th>
<th>PC</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target Class</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AG</td>
<td><strong>65</strong> (16.5%)</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>2</strong> (0.5%)</td>
<td><strong>1</strong> (0.3%)</td>
<td><strong>0</strong> (0.0%)</td>
</tr>
<tr>
<td>CC</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>76</strong> (19.2%)</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>1</strong> (0.3%)</td>
<td><strong>0</strong> (0.0%)</td>
</tr>
<tr>
<td>GR</td>
<td><strong>1</strong> (0.3%)</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>75</strong> (19.0%)</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>0</strong> (0.0%)</td>
</tr>
<tr>
<td>PC</td>
<td><strong>13</strong> (3.3%)</td>
<td><strong>1</strong> (0.3%)</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>75</strong> (19.0%)</td>
<td><strong>1</strong> (0.3%)</td>
</tr>
<tr>
<td>SP</td>
<td><strong>0</strong> (0.0%)</td>
<td><strong>2</strong> (0.5%)</td>
<td><strong>2</strong> (0.5%)</td>
<td><strong>2</strong> (0.5%)</td>
<td><strong>78</strong> (19.7%)</td>
</tr>
<tr>
<td></td>
<td>82.3% (17.7%)</td>
<td>96.2% (3.8%)</td>
<td>94.9% (5.1%)</td>
<td>94.9% (5.1%)</td>
<td>98.7% (1.3%)</td>
</tr>
</tbody>
</table>
Results

- Geometric Classification

Confusion Matrix:

<table>
<thead>
<tr>
<th>Target Class</th>
<th>AG</th>
<th>PC</th>
<th>SP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>65</td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td>PC</td>
<td>0</td>
<td>1</td>
<td>78</td>
</tr>
<tr>
<td>SP</td>
<td>0</td>
<td>0</td>
<td>82.3%</td>
</tr>
</tbody>
</table>

- Geometric Classification:
  - GR, 92%
  - AG, 79.7%
  - PC, 97.4%
  - SP, 91.9%
Results

- Riming Degree Estimation
Results

Riming Degree Estimation

Network Label: 2
$R_{t_c}$ values:
C1: 57%
C3: 43%
C5: 0%

Network Label: 3
$R_{t_c}$ values:
C1: 02%
C3: 97%
C5: 01%

Network Label: 4
$R_{t_c}$ values:
C1: 01%
C3: 41%
C5: 58%

Network Label: 4
$R_{t_c}$ values:
C1: 0%
C3: 63%
C5: 37%

Network Label: 4
$R_{t_c}$ values:
C1: 0%
C3: 37%
C5: 63%

Network Label: 5
$R_{t_c}$ values:
C1: 0%
C3: 0%
C5: 100%
Results

➢ Riming Degree Estimation - Human Error

Human label: 5
Network Label: 4
\( R_{IC} \) values:
C1: 05%
C3: 10%
C5: 85%

Human label: 1
Network Label: 2
\( R_{IC} \) values:
C1: 37%
C3: 62%
C5: 01%

Human label: 3
Network Label: 2
\( R_{IC} \) values:
C1: 26%
C3: 74%
C5: 0%

Human label: 4
Network Label: 5
\( R_{IC} \) values:
C1: 00.0%
C3: 00.5%
C5: 99.5%
Results

- Riming Degree Estimation - Class Bias

a) Network Label: 1
   $R_{l,c}$ values:
   C1: 99%
   C3: 00%
   C5: 00%

b) Network Label: 1
   $R_{l,c}$ values:
   C1: 97%
   C3: 03%
   C5: 00%

c) Network Label: 2
   $R_{l,c}$ values:
   C1: 44%
   C3: 38%
   C5: 18%
Future Work

➢ Growth of Geometric Training Set
  o Apply the network to different data sets
  o Introduce additional class types

➢ Refine Riming Degree Estimation
  o Develop larger training set (with Geometric classifier)
  o Test accuracy/validity

➢ Classify Wet/Dry snowflakes
Conclusion

- In-situ measurement devices would benefit from automatic data processing
  - Whether that is data classification or intelligent filtering, both lead to longer deployment time and faster results

- Convolutional Neural Networks
  - Viable candidate for preprocessing front end
  - Fast and accurate
  - Easy to implement
  - Versatile for both image recognition and classification tasks

- CNNs may prove viable for Rimming Degree Estimation
  - Removes guess work or need for complicated quantization

- Let it snow, let it snow, let it snow!
References


