Automatic Snowfall Detection

Machine Vision for Identification of Snowfall in Images using a Convolutional Neural Network

The Problem

Can a neural network be trained to automatically recognize falling snow via machine vision?

U.S. Department of Transportation:
~ 4% of vehicle crashes are related to snow

Localized snow detection could be useful for transportation and emergency personnel.

Navigation apps could inform users of falling snow.

On a separate note, ski resorts could easily monitor new snowfall on each run.

Presentation Overview

01.
The Data
- Data source
- Overview

02.
Preprocessing
- Preparing the data for modelling

03.
The Models
- Structure
- Performance
- Streamlit app

04.
Conclusions
- Takeaways
- Future Directions

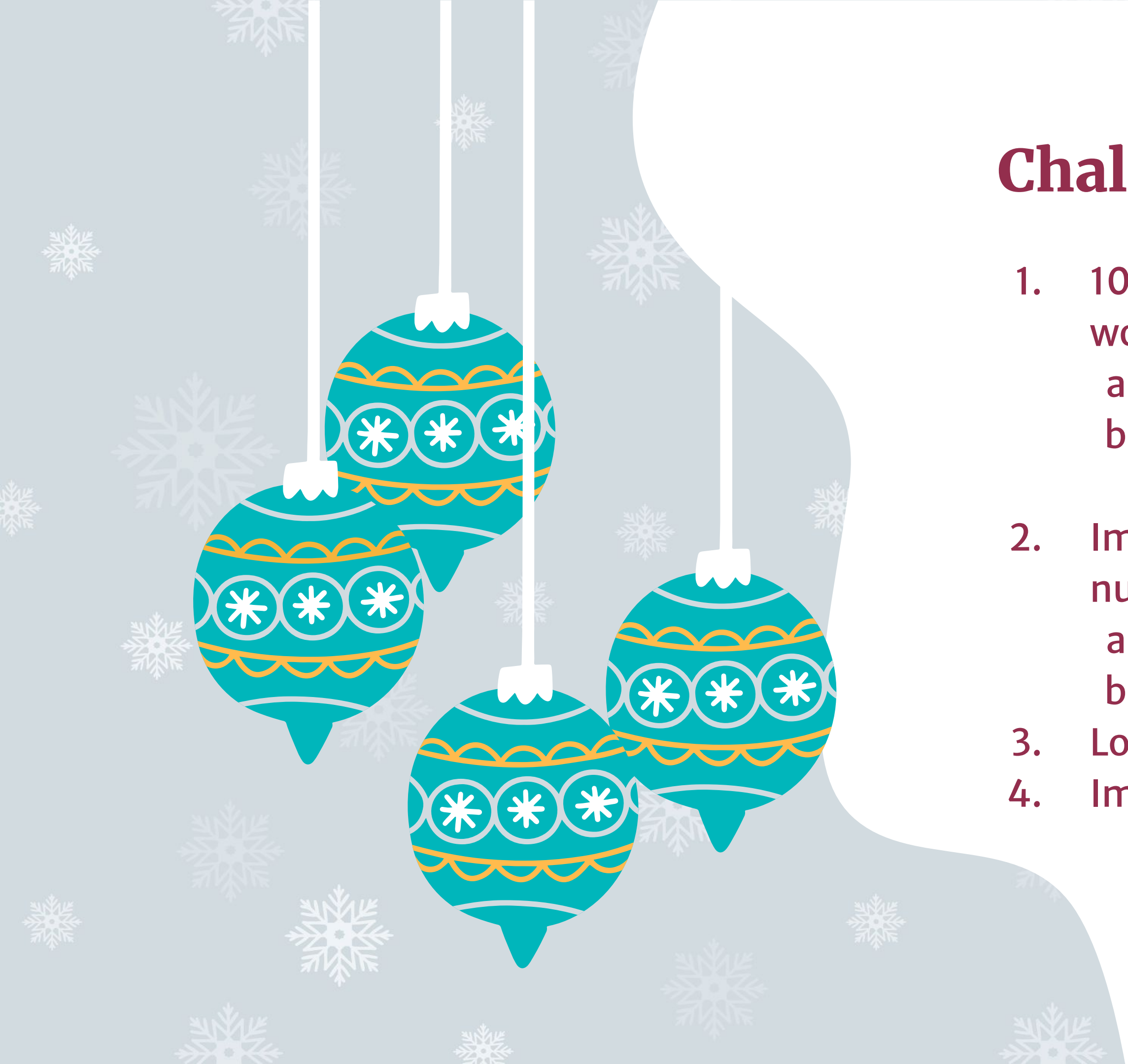
The Data

DesnowNet: Context-Aware Deep Network for Snow Removal

Liu, Yun-Fu and Jaw, Da-Wei and Huang, Shih-Chia and Hwang, Jenq-Neng

- 50,000 images without snow
- A copy of each image with falling snow artificially added
- All RGB-encoded .jpg
- Various resolutions
- Largest dimension for any image = 640 pixels

Kindly provided for download on their website:
https://sites.google.com/view/yunfuliu/desnownet



Challenges of this Dataset

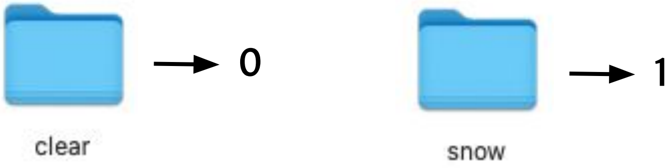
1. 100,000 images is far too large for working memory
   a. How can we shuffle and split?
   b. How can we feed so many images to the model?
2. Images must be paired with a numerical label
   a. 1 for snow
   b. 0 for clear
3. Long time to train
4. Images are different sizes

Preparing the Images for Modelling

1. 100,000 images is far too large for working memory
   a. Start with just the file paths
   b. Shuffle and split before reading in image data
   c. Read in the images in batches

2. Images must be paired with a numerical label

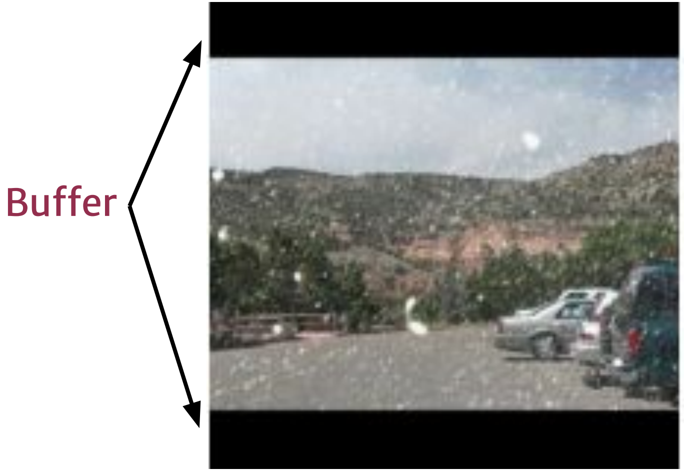
Get the labels from the folder name



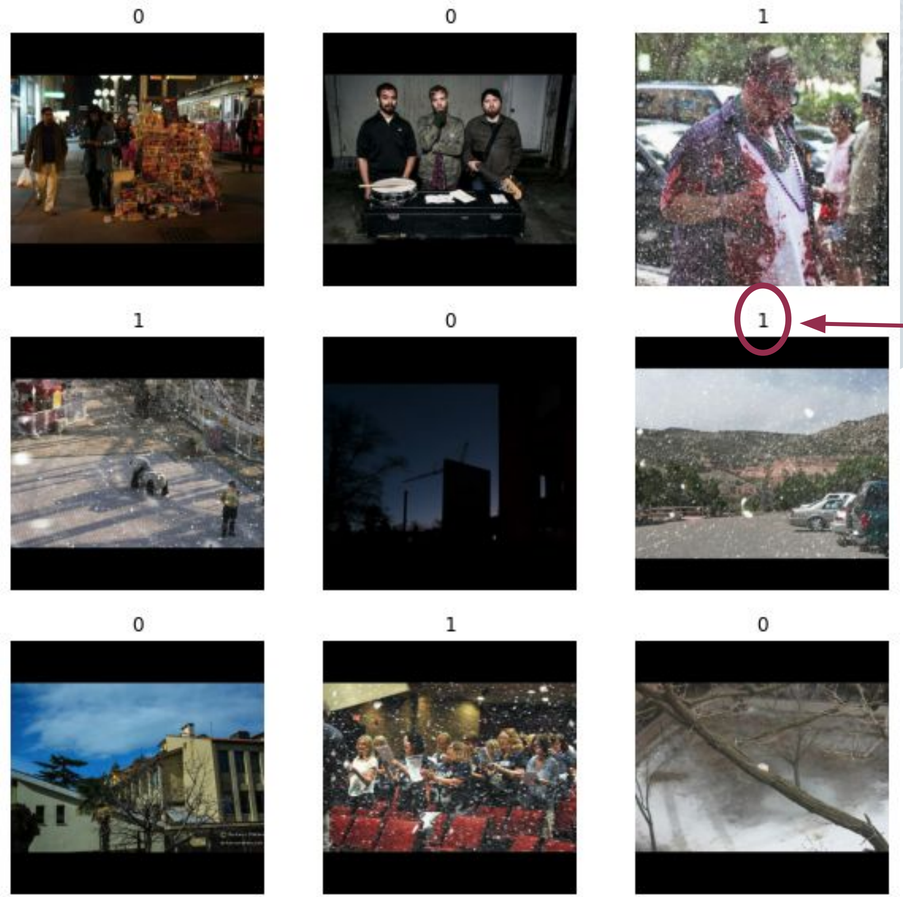
3. Long time to train

Use a GPU → NVIDIA CUDA

4. Images are different sizes


Buffer

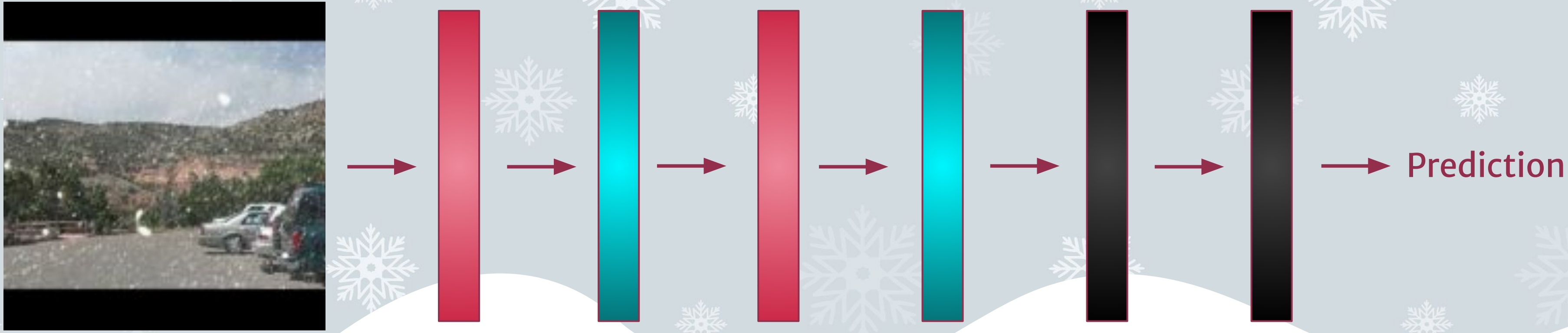
Ready for the Neural Net



1 for snow
0 for clear

Buffers make every
image 640x640

The Models



🟥 Convolutional Layer: Groups pixels near each other, helping identify shapes.

🟦 Max Pooling Layer: Pixels near each other all given the value of the largest.

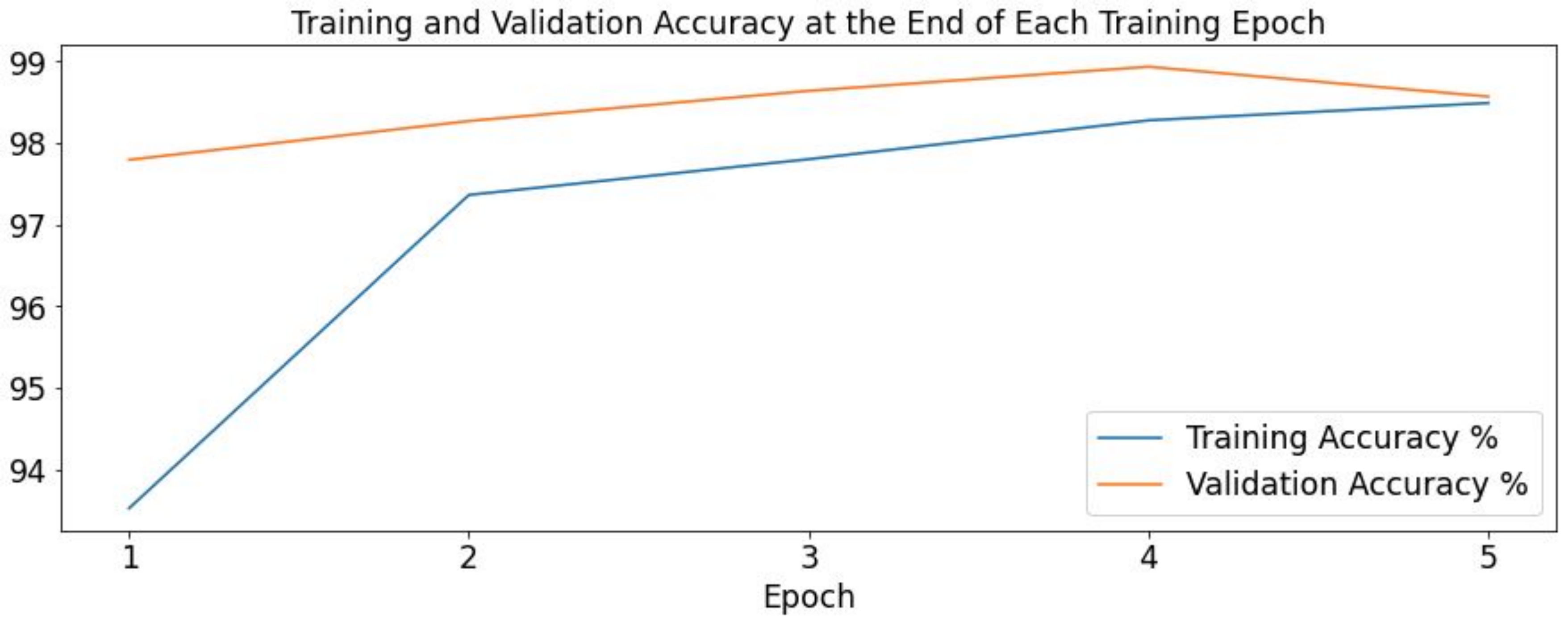
⬛ Dense Layer: Look for patterns. Mostly a black box.

Model 1

Convolution Kernel: 5x5

Max Pooling Kernel: 3x3

Training Passes: 5



Model 1

🟥 Convolution Kernel: 5x5

🟦 Max Pooling Kernel: 3x3

Training Passes: 5

Metrics

Precision: If I say a picture has snow, how often am I right?

- 99.26%

Recall: Of the pictures with snow, how many did I identify?

- 97.80%

Accuracy: Overall, how many of my predictions were correct?

- 98.53%

I don't want to miss snowy pictures.

Can I improve the Recall?

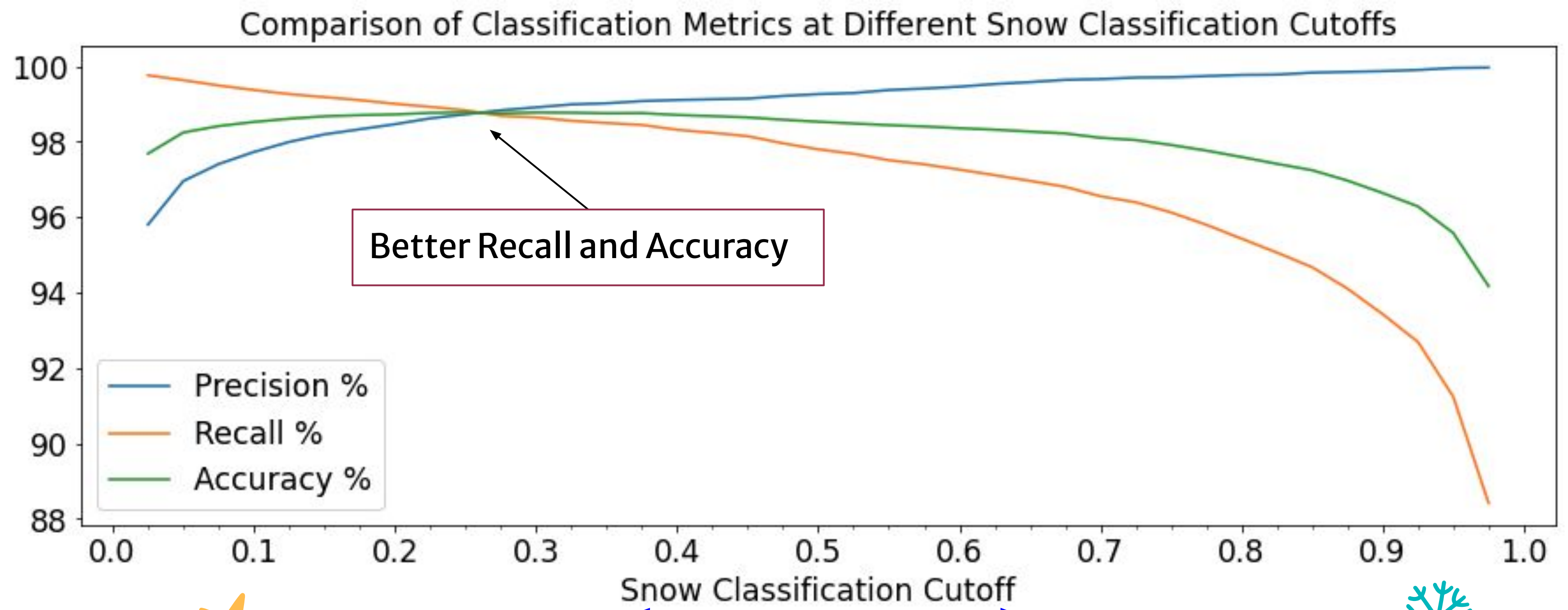
And keep high accuracy?

Model 1

Convolution Kernel: 5x5

Max Pooling Kernel: 3x3

Training Passes: 5



Better Recall and Accuracy

Looks Clear

Looks Snowy

Model 1

🟥 Convolution Kernel: 5x5

🟦 Max Pooling Kernel: 3x3

Training Passes: 5

Updated Metrics

Precision: If I say a picture has snow, how often am I right?

- 99.26% → 98.73%

Recall: Of the pictures with snow, how many did I identify?

- 97.80% → 98.84%

Accuracy: Overall, how many of my predictions were correct?

- 98.53% → 98.78%

Cutoff can be adjusted significantly without much accuracy loss.

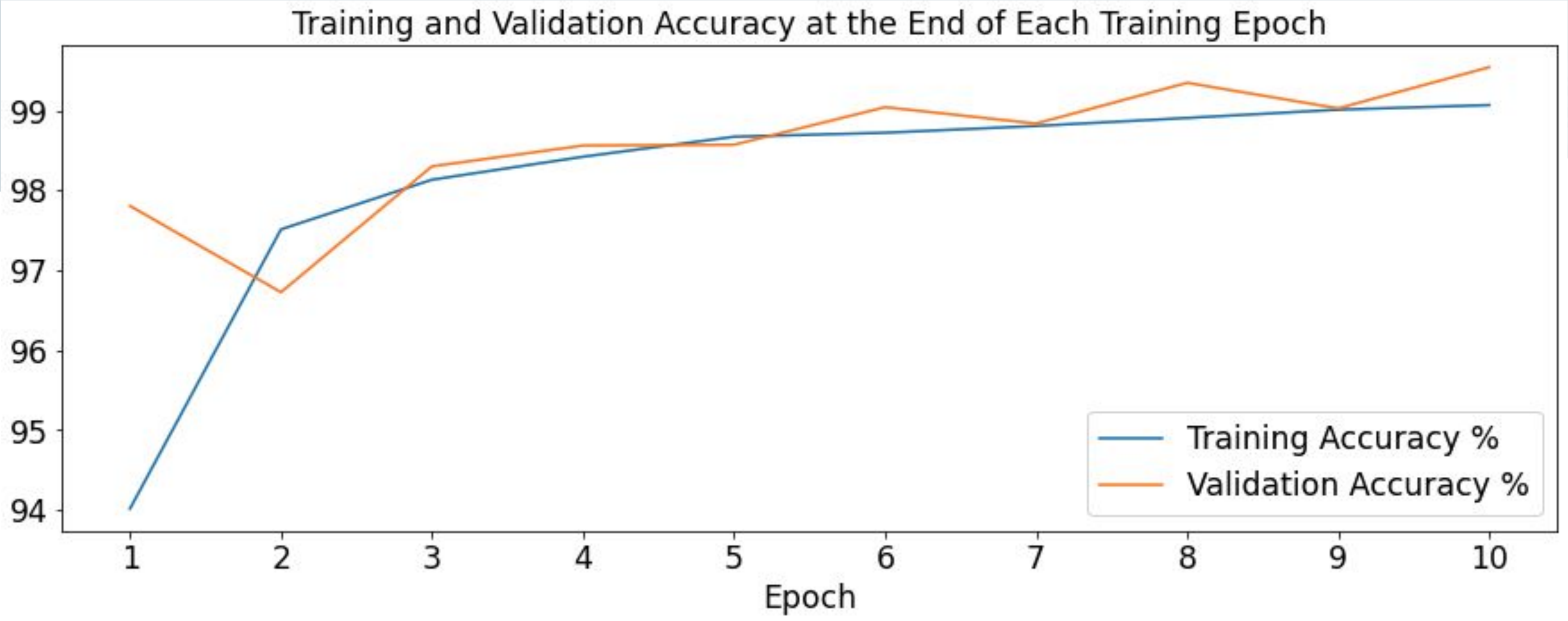
Model is highly tunable.

Model 2

Convolution Kernel: 4x4

Max Pooling Kernel: 4x4

Training Passes: 10



Model 2

🟥 Convolution Kernel: 4x4

🟦 Max Pooling Kernel: 4x4

Training Passes: 10

Metrics

Precision: If I say a picture has snow, how often am I right?
- 99.37%

Recall: Of the pictures with snow, how many did I identify?
- 99.57%

Accuracy: Overall, how many of my predictions were correct?
- 99.47%

This model is already very balanced.

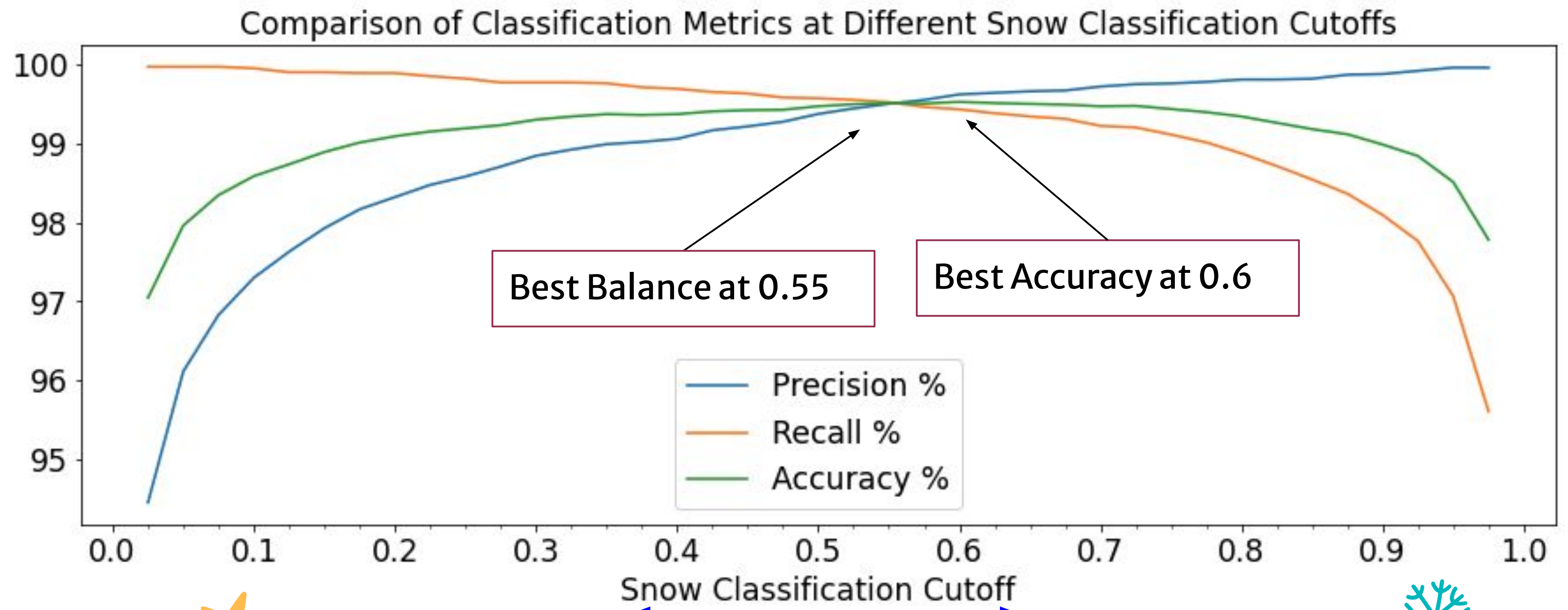
How tuneable is it?

Model 2

Convolution Kernel: 4x4

Max Pooling Kernel: 4x4

Training Passes: 10



Best Balance at 0.55

Best Accuracy at 0.6

Looks Clear ← → Looks Snowy

Model 2

🟥 Convolution Kernel: 4x4

🟦 Max Pooling Kernel: 4x4

Training Passes: 10

Updated Metrics: Cutoff 0.55

Precision: If I say a picture has snow, how often am I right?

- 99.37% → 99.50%

Recall: Of the pictures with snow, how many did I identify?

- 99.57% → 99.52%

Accuracy: Overall, how many of my predictions were correct?

- 99.47% → 99.51%

This model is still highly tunable.

Streamlit

Let’s see the model in action.

Conclusions

Takeaways

- The models presented are very capable of identifying falling snow in images.
- The models can be easily tuned for greater precision or recall depending on the application.
- The models are small, less than 1 MB for model2, making them easily deployable.

Future Directions

- Test on more real snow images.
- Analyze failure cases.
- Further training and tuning possible, but likely unnecessary.
  - Optimize for speed.
- Identify snow in live video or webcam. May require more sophisticated algorithm (YOLO).
- Use a GAN neural net to remove snow from images.
  - Original use for this data.

Thanks!

Do you have any questions?

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**