A Study of Weather Detection in Public Webcams Using Neural Networks

Abraham Wong
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, arwong@andrew.cmu.edu

Abstract

A framework is described for detecting the weather conditions in outdoor public webcams. A neural network is trained using recorded data from cameras, such as traffic cameras, and radar and satellite weather data from NOAA and NASA.

Introduction

Thousands of cameras output their images to public sites on the Internet, capturing fish tanks to public spaces to bedrooms.

This paper describes a framework which could be used to detect weather conditions in outdoor public webcams. Given the density of some traffic cameras, applying the framework to these images could provide weather data at higher spatial and temporal resolutions than that provided by weather satellites and radar.

The hypothesis is therefore that this framework is viable and effective. Experiments were conducted, and results are presented.

Background

District 11 of the Pennsylvania Department of Transportation (PennDOT) encompasses Allegheny, Beaver, and Lawrence counties, and the city of Pittsburgh. It maintains a network of 64 cameras in the Pittsburgh area which watch major roadways such as I-279 and I-376. These are 320x240 color and black-and-white images which update every several minutes, and are of generally poor quality.

Artificial neural networks are inspired by the human nervous system, and are built out of artificial neurons. Neurons are cells which are naturally interconnected, and which have the ability to learn. In biological systems, a neuron receives real-valued inputs from other neurons and computes a weighted sum. If the sum exceeds a certain threshold, the neuron "fires," sending outputs to other neurons. Learning happens when the thresholds and the inputs, outputs, and their weights change.

Artificial neural networks (ANNs) approximate this. Unlike biological neurons which are either on or off, standard artificial neurons have a differentiable output function; an important characteristic which allows a learning system for artificial networks to compute slopes in the error space to perform gradient descent of the weights.

ANNs have been applied to many applications, including computer vision tasks such as face recognition (Rowley, et al.), and steering a motor vehicle (Batavia, et al.) in the ALVINN project. The success of ANNs in these tasks was a strong motivator for their selection for this application.

Methods

Materials

PennDOT Traffic Cameras. A network of traffic cameras in the Pittsburgh area.
NOAA Radar Imagery. Online radar images.
GEOS Satellite Imagery. Online satellite imagery from NASA weather satellites.
Downloader. A Java application which downloads webcam images at a set interval.

JooneInputFileBuilder. A Java application which takes traffic camera images and weather data imagery and outputs text files suitable for Joone.
Joone. An open-source Java implementation of artificial neural networks. It includes both a Java API and a GUI application. (http://www.jooneworld.com)

Operational Definitions

Precipitation. The DBZ as reported by the NWS radar map, which can be easily partitioned into classes such as "light rain" (5-15), "moderate rain" (35-45), etc., according to the scale on NWS radar available at http://radar.weather.gov.
Cloud Cover. The brightness of the NASA GOES infrared satellite map.

Design

The framework ideally works as follows:

1. A Downloader is set to regularly download images from the PennDOT cameras and weather sites.
2. Traffic cameras are manually associated with pixel locations on the weather images.
3. For each weather data image source, an average value is taken for a 3x3 box around each pixel, constituting either the precipitation or the cloud cover.
4. For each image in the test set, scale the image down, and output it as a 1-dimensional array.
5. Train a neural network using the image data as input, and the weather data as desired output.
6. The trained network can then be used to classify new camera images, such as images gathered in real-time from PennDOT.

The downloader is set to a 5 minute interval, which provides a balance between the 3-5 minute interval of the traffic cams, 10 minute interval of radar, and ~15 minute interval of the satellite data.

The ANN training was performed on two machines: a dual Intel Xeon 2.0 GHz with 768 MB RAM, running Windows 2000 and J2SE 1.4.3, and a single-processor 1.7 GHz Pentium M with 1 GB RAM, running Windows XP and J2SE 1.5.

Cameras

To limit the scope of this study, only two cameras were seriously considered: the SR-60 at Robinson Town Center camera (cam06), and the I-279 at Evergreen Rd camera (cam14).

Figure 1. cam06

Figure 2. cam14

Results

All experimental results refute the hypothesis. A wide range of input, output, and network configurations were tried. Configuration variables include:

Image size. The source 320x240 image was scaled to sizes between 16x12 and 160x120. At sizes beyond 64x48, the networks took intractably long to train; over 2 hours for the first 50 epochs.
Camera angles. Images were from either a single or multiple camera angles.
Weather data. Inputs included precipitation alone, cloud cover alone, or both data.
Image time. Input data was taken from portions of the day, ranging from 12pm to 6pm, and all day.
Training images. Between 5 and 1600 training images were used.
Image cropping. Input data was either a cropped portion of a full camera image, a full camera image, or both.
Network structure. The network had either one or two layers of hidden nodes, each of which had between 2 and 128 nodes. Layers were either linear (generally the input layer), sigmoid (generally the hidden and output layers), or sine.

The synapses between layers were either full, Sanger, or Kohonen. The network was feed-forward in all cases, and trained with back propagation.

Network parameters. The learning rate and momentum were varied between 0.05 and 0.9.

Output format. The desired output was expressed either as a single value, as the unary of a scaled datum (e.g., 1, 1, 1, 0, 0, 0 represents 0.5 on 6 output nodes), or as a single "on" node among "off" nodes (0, 0, 1, 0, 0, 0 for 0.5 on 6 nodes)

Networks were trained using the root mean squared error (RMSE). In all cases, the networks' errors never deviated substantially from their original values between 0.38 and 0.54, which corresponded with a single output array for all inputs that was either a DBZ around 0 (no rain), or a cloud cover around 70%. Using a sine output layer caused the RMSE to fluctuate, but it never strayed far from its original value.

Note that it was possible to train identical networks on test training data. In particular, networks were trained to compute the XOR of a checkerboard-style input array, output zero, and extract individual pixels from a full camera image, with little difficulty.

Some features of the images and weather data are worth noting, however, for future work.

1. Image quality of PennDOT images is poor. Comparable detail is visible if the images are scaled to a quarter of their original area.
2. There is variation in overall color between images, such that one might be redder than another, which reinforces the idea of one neural network per camera.
3. Cameras are not fixed; a single camera may change views throughout the day. The angles appear to be limited to a few fixed views per camera. (see figures 3 and 4)
4. Rain most often presented itself as a shiny road and rain on the camera lens. (see figure 5)
5. The precipitation measure seemingly correlates well with the true observable weather, while the cloud cover measure appears to lack sufficient resolution. (see figures 5 and 6)

Figure 3. cam18, 3/31, 9:04am

Figure 4. cam18, 3/31, 2:14pm

Figure 5. cam14, 3/31, 4:29pm, DBZ is 23.0 – "rain"

Figure 6. cam14, 4/2 11:04am, DBZ is -0.6 – "no rain"

Discussion and Future Work

These results are very surprising. Given the effectiveness of neural networks in other image classification tasks, the fact that the networks invariably failed to learn a nontrivial function may very well be due to some flaw in the construction of the networks.

The ALVINN project at CMU used a 25x25 grayscale input image, a 3-layer feed-forward network with 4 hidden layer nodes, and desired output of a single "on" node among "off" nodes (Batavia, et al.) Among methods it trained on, back propagation worked successfully. This would suggest that weather detection would work well, but clearly it did not in these experiments.

Future work should address concerns about flaws in the neural networks, though how to do this is as yet unclear.

It may very well be possible that ANNs are simply poorly suited for this task. If this is the case, other techniques could be examined, though many exist. *k*-nearest neighbor may be the most promising, though naïvely using it may doom a system to unacceptably slow test times. *k*NN requires a good similarity metric, however, which could require much work to develop.

References

H. Rowley, S. Baluja, and T. Kanade. Neural Network-Based Face Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 1, January, 1998, pp. 23-38.

NASA. Interactive GOES-12 (East) Infrared Weather Satellite Image Viewer. http://www.ghcc.msfc.nasa.gov/GOES/goeseastconusir.html as of 30 March 2006.

NOAA. NWS radar image from Pittsburgh, PA. http://radar.weather.gov/ridge/radar.php?rid=pbz as of 30 March 2006.

PennDOT District 11. All Cameras View. http://www.nb.net/~finals/allcams.htm as of 30 March 2006.

P. Batavia, D. Pomerleau, and C. Thorpe. Applying Advanced Learning Algorithms to ALVINN. tech. report CMU-RI-TR-96-31, Robotics Institute, Carnegie Mellon University, October, 1996

Appendix

Additional Images

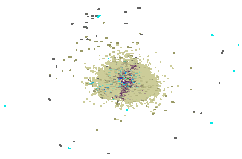

Figure 7: NWS Radar

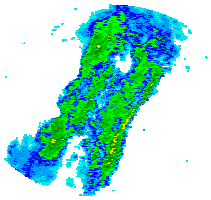

Figure 8: NWS Radar

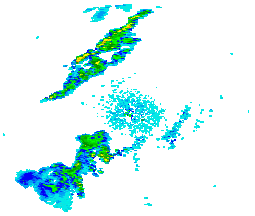

Figure 9: NWS Radar

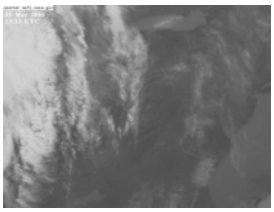

Figure 10: NASA infrared

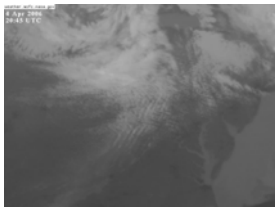

Figure 11: NASA infrared

Figure 12: cam06

Figure 13: cam06

Figure 14: cam22

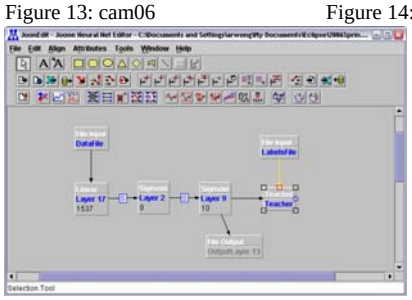

Figure 15: Joone GUI and ANN visualization