J4.2

A CLOUD DETECTION ALGORITHM APPLIED TO A WHOLE SKY IMAGER INSTRUMENT USING NEURAL NETWORKS

Andy Linfoot* and Randall J. Alliss
Northrop Grumman Information Technology TASC, Chantilly, Virginia

1. abstract

The knowledge of the placement and size of clouds in the atmosphere has many applications in Atmospheric Science. In particular, clouds play a major role in the earth's radiation budget. Given the large amounts of image data and the relative short time this information is relevant requires an automated detection algorithm that runs in real-time on current workstations. We present such an algorithm in this paper.

We show that given daytime images, human analysts agree amongst themselves at most 94% on the placement of clouds within the image. The subjectivity of the problem motivates our approach. The algorithm uses a combination of neural networks to compute the probability of a pixel being clear or cloud. Images are first sorted into neighborhoods via a Self-Organizing map based on the image's gray scale levels. After sorting, each pixel is processed using a Mixture of Experts neural network unique to the neighborhood. The calculation is of the probability that a pixel represents clear or cloudy. A pixel level cloud mask is then constructed by assigning the most likely designation to each pixel.

The algorithm reliably identifies a variety of clouds from images. Relative agreements with human analysts are typically greater than 90%. In addition, the pixel level cloud masks show high visual correlations with unprocessed images as well as very good temporal correlations.

2. Introduction

Images of the sky are taken by a whole sky imager (WSI). This is an instrument developed by the Atmospheric Optics Group at the Marine Physical Laboratory of the Scripps Institution of Oceanography at the University of California, San Diego Keller et al. (2006). It is designed to take a horizon to horizon picture of the sky at user specified times. For a given time it takes three images in quick succession. Each of these images is taken with a different filter. The filters are blue, red, and near infrared. In our discussions we refer to these as BLU, RED, and NIR. The images produced are 16-bit with 512x512 total pixels in each image. Examples of images can be seen in Figure 10. A picture of a typical unit is shown in Figure 1.

Corresponding author address: Andy Linfoot, Northrop Grumman Information Technology TASC, 4801 Stonecroft Blvd, Chantilly, VA 20151. E-mail: andy.linfoot@ngc.com

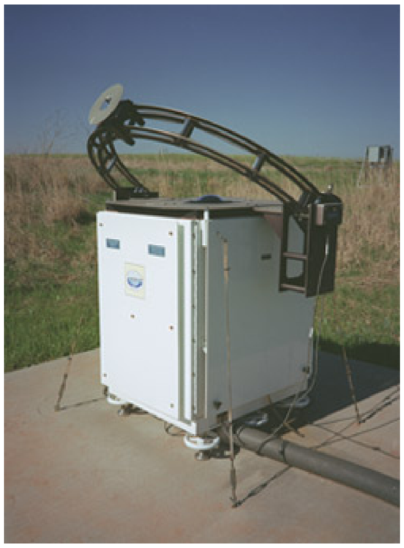
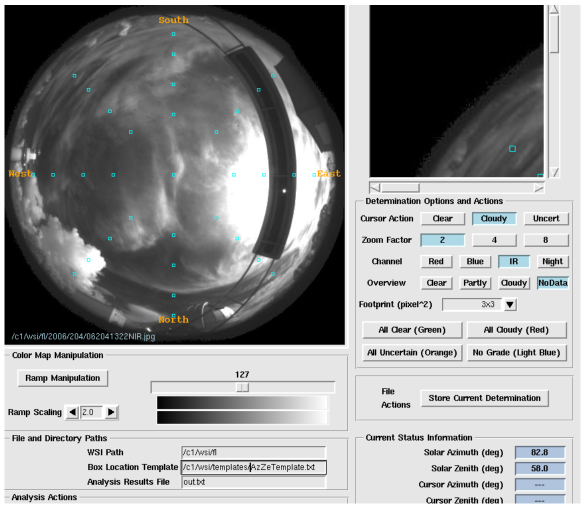


Figure 1: Whole Sky Imager

Our problem is given such an instrument and images to determine the location of clear sky and clouds. Due to the high volume of imaging data this determination needs to be automated and completed relatively quickly. Each group of three images is taken at roughly one minute intervals throughout the day. Unless there is mechanical issues images are taken everyday. The average number of image times for a day is ~ 1000 which translates to $\sim 365,000$ cloud masks per year. More importantly for the information to be relevant the images need to be processed as quickly as possible.

The ultimate goal is to assign to each pixel in an image a value of 0 if it is clear sky and 1 if it is a



WSI Path: /c1/wsi/fl
Box Location Template: /c1/wsi/templates/AzZeTemplate.txt
Analysis Results File: out.txt

Solar Azimuth (deg): 82.8
Solar Zenith (deg): 58.0
Cursor Azimuth (deg): ---
Cursor Zenith (deg): ---

Figure 2: Example of imaging grading.

cloud. Thus each set of images for a given time are used to build a "$0/1$" map from which relevant atmospheric information can be acquired. Some examples are; cloud fraction, overall direction of clouds, clear sky quality, etc.

Without other means of verification we assume that human analysts are the gold standard. We demonstrate through a small study that on average analysts when given the same set of images only agree $\sim 94\%$ with each others assessments. Therefore, identifying clouds in a given image is not entirely objective.

Nevertheless, we have developed an algorithm to produce accurate cloud masks using a combination of neural networks. Our cloud mask algorithm consists of using a combination of neural networks responsible for pre-processing and pixel level mappings. Accuracy here is to be understood as agreement with human analysts. Finally, we show that this level of agreement is achievable with relatively small networks. The total number of network weights is less than 298. This makes the computational expense of our algorithm negligible. Our algorithms are capable of processing a days worth of images in 3 minutes on a 3 GHz Intel Xeon workstation.

3. The Gold Standard

Without other means of verification we considered a human analyst as the gold standard for determining the designations. To evaluate the performance of the various algorithms we had several analysts grade the sample group of images at the same pixel

B\A	Clear	Cloud	Uncertain
Clear	c_{00}	c_{01}	c_{02}
Cloud	c_{10}	c_{11}	c_{12}
Uncertain	c_{20}	c_{21}	c_{22}

Table 1: Example Contingency Table

locations. This sample was then used to judge accuracy for our algorithm. Also similar information was also used to form a training set for the neural networks.

We had a total of six analysts. The analysts evaluated 1095 total images. These images represented an almost uniform sample from the months of May, June, September, October, November, and December for the year 2006. Each image was designated Cloud, Clear, or Uncertain in 33 pixel boxes placed at regular intervals. The total number of designations for each analyst was close to $36,135$ as some boxes were left undetonated by certain analysts. All of the designated boxes were then compared to the various algorithms to generate a percentage agreement or overall score. Figure 2 shows the template on a typical image.

The sampling of images and the placements of evaluation boxes was not random. We decided to only grade images during the day. In addition, each graded image usually corresponded to ten minutes before the hour. The time was chosen to allow for cross validations with other meteorological data. There were a few images that did not correspond to this sampling scheme due to lack of availability. There is also a lot of temporal correlation in images so grouping several images in a small time window is not entirely necessary. The pixel boxes were placed at various azimuth and zenith angles. Azimuth angles ranged between $0°$ and $315°$ with a increment size of $45°$. The center of the image corresponds to zenith angle $0°$. Zenith angles ranged between $0°$ and $70°$ with increment size of $15°$. So for a given image there are approximately 33 boxes.

To get a statistical upper bound on how well a given algorithm could perform with our evaluation scheme we decided to compute the conditional probabilities between analysts. This was done in two ways. The first was to include all designations, i.e. Clear, Cloud, and Uncertain. We also repeated the calculations disregarding any box that was designated as Uncertain by either analyst. The calculation of the probabilities was facilitated by use of contingency tables. An example of a contingency table can be found in Table 1. We computed four main probabilities. It was natural to partition the compu-

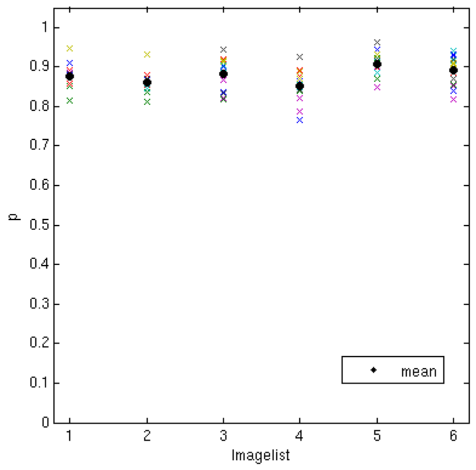


Figure 3: Sample probability of agreement. Each point represents the agreement rate amongst two analysts for all three classifications; Clear, Cloud, Uncertain.

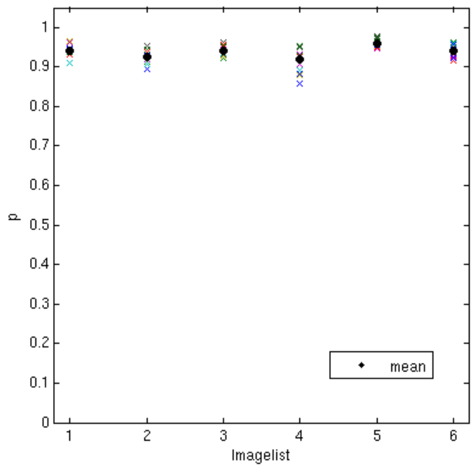


Figure 4: Sample probability of agreement. Each point represents the agreement rate amongst two analysts using only Clear and Cloud classifications.

tation based on image list.

The first was overall agreement. These calculations can be found in Figures 3 and 4. The formula for this calculation is simply the sum of the diagonal in the table divided by the sum of the entire table when counting uncertain and the same for the upper 2x2 table when we disregard Uncertain.

Finally for each designation we computed the agreements. Figures 5, 6, and 7 are the scatter plots of all possible pairings between analysts when we take into account Uncertain. The probabilities when excluding Uncertain are in Figures 8 and 9. Taking x and y to denote the various designations this corresponds to $P($ “A:x” | “B:y” $)$. The probability of each of these is simply the number of times A designated x divided by the number of times B designation y. As an example the probability that A agrees with B's Cloud designation is,

$$P(\text{ “A:Cloud” } | \text{ “B:Cloud” }) = \frac{c_{11}}{c_{10} + c_{11} + c_{12}}$$

Again as in the overall agreement when Uncertain designations are ignored the table is reduced to the upper 2x2 portion.

The best an algorithm can score with regard to our evaluation methodology is the mean of the agreements between the analysts. It does not mean that the best an algorithm can perform in detecting clouds is this mean but rather how well we can measure performance. The main issue is that determining whether a given group of pixels represents a cloud is not entirely objective. There is a small percentage of times that are not straight forward to classify. This is particularly true of boxes that are labeled Uncertain and provided our motivation for computing additional statistics disregarding these classifications.

Studying the various conditional probabilities we see that on Cloud and Clear designations analysts have a high level of agreement. The high level of agreements on these classifications are easily understood. For most of the classifications it is very straight forward to label the box as Clear or Cloudy. Boxes that received the designation of Uncertain vary quite considerably. This is clear from Figure 7. Part of the explanation for the high variances on Uncertain classifications is explained by their relatively small sample size. Another reason is analysts were not given specific instructions on what constituted any of the particular classifications. Uncertain classifications are used to designate a variety of occurrences in images ranging from airplanes to the halo effect around the sun. Classifications involving the halo effects are hard to determine due to limitations of our validation tool and the images themselves. For example the validation tool does not allow for adjustments of brightness nor contrast.

Overall there was a high level of agreement between the analysts with the exception of Uncertain designations. If we include the Uncertain designations, then the average level of agreement is 88.0%.

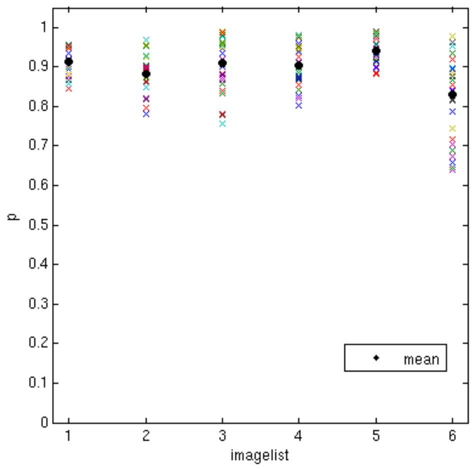


Figure 5: Sample probability of agreement between analysts conditioned on Clear classifications. Each point represents one possible pairing of the analysts. These are calculated using the 3x3 contingency tables.

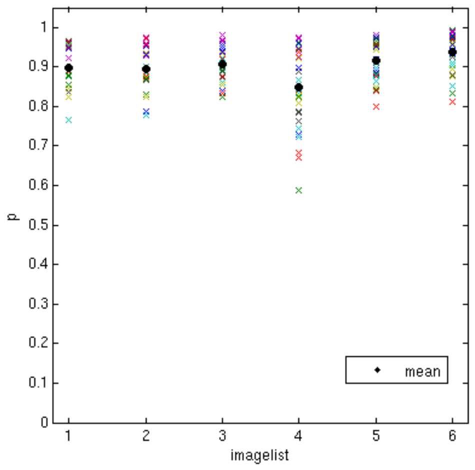


Figure 6: Sample probability of agreement between analysts conditioned on Cloud classifications. Each point represents one possible pairing of the analysts. These are calculated using the 3x3 contingency tables.

The average agreement between analysts moved upto 93.8% when disregarding Uncertains. This is reasonable in our opinion due to the limitations of our evaluation process. Also it is safe to assume that there are a small percentage of designations that were misclassified by the analyst due the error of checking the wrong box. Finally, there are instances when the analyst was not entirely sure but decided to designate the box as either Clear or Cloud. It is our estimate that these factors account for $6 \sim 8\%$ of the $36,135$ possible designations. The majority of these resulting from when analysts are not entirely certain of their designation.

The conclusion we draw is that the performance of any algorithm should be relative to the average level of agreement between analysts. This average gives a statistical upper bound on how well an algorithm can be evaluated. Basically, it is impossible for the algorithm to agree on all designations with all the analysts. The algorithm's performance in relative terms should consider the level of agreement amongst analyst as 100% accuracy. The relative accuracy of an algorithm is then given by,

$$\text{Relative \%} = \frac{\text{Algorithm's \%}}{\text{Average \% Agreement}}$$

We will discuss performance in both absolute and relative terms.

4. Cloud Detection Algorithm

Our algorithm makes use of several neural networks to assign a clear or cloud designation to pixels in images. The first step is to sort the images into neighborhoods using a self-organizing map. Following the assignment of an image to a neighborhood another neural network assigns clear or cloud to each pixel in the image.

As demonstrated in the previous section the identification of clouds in an image is subjective. This lead us to calculate a pixel's probability of being a clear or cloud. The central idea to our algorithm is that a neural network can be used to approximate this probability distribution. To accomplish this we use a *Mixture of Experts* neural network Jacobs et al. (1996). The images are placed into different neighborhoods to partition the problem domain hence reducing spatial complexity. Each of these neighborhoods has a uniquely assigned *Mixture of Experts* that is responsible for pixel level classifications.

a. Building Neighborhoods via a Self-Organizing Map

The ultimate goal of the unsupervised learning was to reduce the spatial complexity of the problem. We desired an automated way to classify images into large scale types. These large scale types

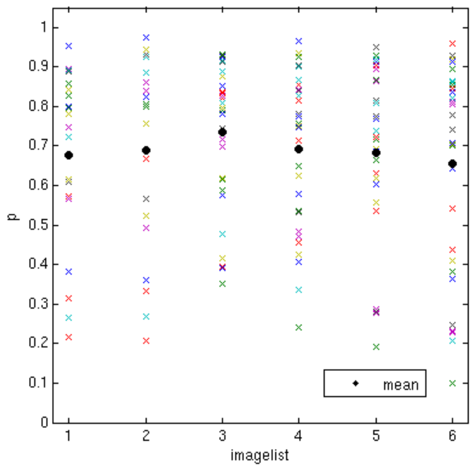


Figure 7: Sample probability of agreement between analysts conditioned on Uncertain classifications. Each point represents one possible pairing of the analysts. These are calculated using the 3x3 contingency tables.

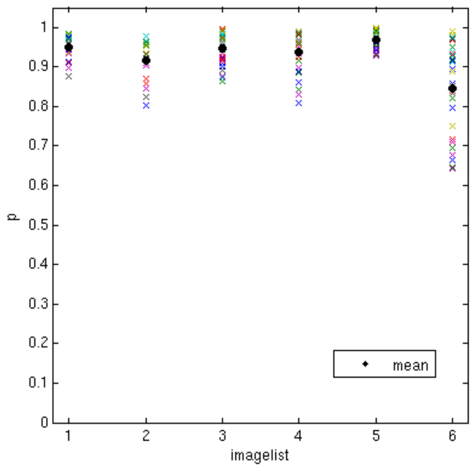


Figure 8: Sample probability of agreement between analysts conditioned on Clear classifications. Each point represents one possible pairing of the analysts. These are calculated disregarding Uncertain.

are thought of as neighborhoods with the images in them having some commonality. For instance, a neighborhood might consists of images classified as clear, cloudy, hazy, etc.. The classification network is trained on a subset of images and then acts as a pre-processor for threshold algorithms.

We used a variation of a Self-Organizing Map. The details of the base algorithms which we used can be found in Kohonen (1990) as well as a general discussion on various properties of these maps. It was hoped that images would be mapped into meaningful neighborhoods. If this was the case, then it would limit the variability in the images. These neighborhoods would then be processed using various threshold algorithms.

Our use of Self-Organizing maps is very similar to a Winner Take All (WTA) Algorithm. WTA train a single layer network via a competitive process. A good introductory discussion of WTA can be found in Brady (1992). We use some slight variations which we discuss below but first we give a brief outline of a generic WTA algorithm.

Generically the algorithm is to start with a training set, an untrained network, an adjustment factor, and a total number of neighborhoods. Each training example is feed through the network. The weights which activate the most with respect to some criteria, i.e. give the biggest response, are adjusted using the input and adjustment factor. All the other weights are kept constant. Iteration of this process trains the network. A complete pass through the training set is typically referred to as an *epoch*. The total number of epochs is decided before hand and training proceeds until this number is reached. The final result is a trained network. The outline of our algorithm is the same. We now give the specifics.

It is convenient to keep the network in a two dimensional array, i.e. matrix form. The rows of this matrix are the different activation groups. The column dimension of the matrix corresponds to the number of components in an input. Since the network is kept in matrix form and given our selection and updating rules feeding an input example through the network corresponds to a matrix multiplication. A function can also be applied to the output matrix component wise, but we have chosen to use the identity function.

We also specify initial weight values, a normalization condition, and a rule for weight adjustment. Each row in the network is initially set to,

$$\mathbf{w^{(k)}} = \frac{\mathbf{1}}{\sqrt{\mathbf{r}}}(\mathbf{1}, \mathbf{1}, \ldots, \mathbf{1}) \qquad (1)$$

as proposed in Hecht-Nielsen (1987) with r equal to the number of components in an input. It is assumed that all the input examples have also been normalized prior to training. After feeding an input through the map we determine the winner by finding the biggest output. The row onto which the input

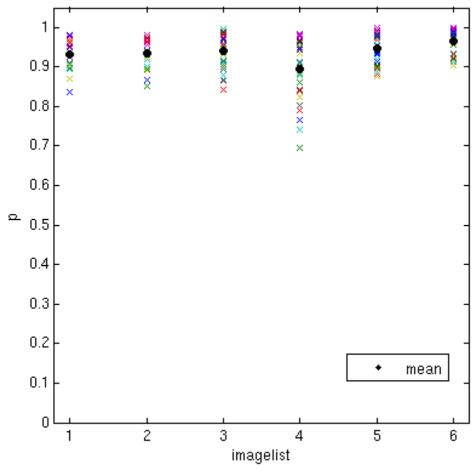


Figure 9: Sample probability of agreement between analysts conditioned on Cloud classifications. Each point represents one possible pairing of the analysts. These are calculated disregarding Uncertain.

has the biggest projection is declared the winner. Explicitly the rule is given by,

$$(\mathbf{x^{(n)}}, \mathbf{w^{(k)}}) \quad = \quad \max_s\{(\mathbf{x^{(n)}}, \mathbf{w^{(s)}})\}. \qquad (2)$$

Once a winning row is determined its weights are updated. We made a slight modification to the update rule presented in Kohonen (1990). The update rule we used is,

$$w_{iv}^{(k)} \quad = \quad w_{iv}^{(k)} + \alpha(t)(x_j^{(n)} - w_{ij}^{(k)}) \qquad (3)$$

with k denoting the winning row. The motivation for forming the difference of $\mathbf{x^{(n)}}$ and $\mathbf{w^{(k)}}$ is to limit the updating of weights. Only weight components that differ considerably from the corresponding input components get a significant update. That is weight components become frozen once they match well with input components. The factor $\alpha(t)$ is a non increasing function of epoch number and whose max value is less than or equal to one. For our training we used piecewise linear functions. After updating the winning row is then re-normalized. The whole process is repeated for a specified number of epochs on the training set.

Although the overall training scheme was the same, we experimented with several different training sets before settling on using histograms of the NIR data. The training sets are built by specifying a given group of files as opposed to random selection. The procedure for building histograms is to first mask out the horizon, this causes the processing to

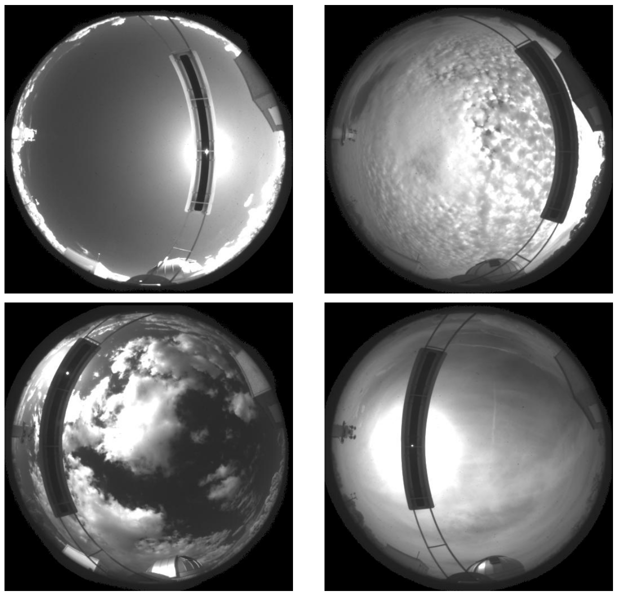


Figure 10: Examples from clustering network training set. Starting from upper left and going clockwise; Clear, Cloudy, Thin Cirrus, and Partially Cloudy.

ignore such things as antennas, buildings, mountains, and trees. The data is then scaled by the inverse of the max value and a histogram is built. These histograms form the training set for the WTA algorithm.

b. *Pixel Classifications*

As stated in the last section we trained a self-organizing map to act as a pre-processor. The goal being to cluster images based upon some coarse grain characteristic. For each of these clusters we would then use a specialized threshold algorithm to build cloud masks. We decided to train independent neural networks for each cluster. These networks would be responsible for the pixel level classifications and generation of cloud masks. The main advantages of using neural networks were; a threshold could be determined on a per pixel basis and meta data could be incorporated into a designation.

Due to the subjective nature of designating pixels we decided to compute the likelihood of designation. This consisted of building and training neural networks to approximate a probability distribution. This is known as a *Mixture of Experts* is presented in Jacobs et al. (1996). An introduction of the concept can be found in Bishop (1995). For clarity we will briefly summarize the general results.

There are two key ideas. The first is that outputs from a neural network can be thought of as continuously varying parameters. This means that the outputs can be used as means and variances to define a Gaussian kernel. The other idea is that by summing Gaussian kernels it is possible to approximate another probability distribution. That is to say for a given probability distribution,

$$p(t|x) \approx \sum_{k=1}^{M} \alpha_k(\mathbf{x})\phi_k(\mathbf{t}|\mathbf{x}). \tag{4}$$

The value of M denotes the number of kernels used in the summation which is the same as the number of experts and C denotes the dimensionality of the encoding. We take the mean $\mu(\mathbf{x})$ to be the kernel's the designation. The coefficient of each kernel is the prior probability of a given input. In the current context it represents the certainty of a particular kernel designation. For the outputs we will use the notation from Bishop (1995). Outputs corresponding to the priors are denoted z_k^{α}, to the variances by z_k^{σ}, and the means by $z_{k,i}^{\mu}$.

Maintaining probability imposes the constraint on the priors,

$$\sum_{k=1}^{M} \alpha_k(\mathbf{x}) = 1. \tag{5}$$

As such we follow convention and take the priors to be defined using a soft-max function, hence

$$\alpha_k(\mathbf{x}) = \frac{\exp z_k^{\alpha}}{\sum_{l=1}^{M} \exp z_l^{\alpha}}. \tag{6}$$

Typically, each kernel corresponds to a different expert. A kernel is given by,

$$\phi(\mathbf{t}|\mathbf{x}) = \frac{1}{(2\pi\sigma^2)^{C/2}} \exp\Big(\frac{-\|\mathbf{t}-\mu(\mathbf{x})\|^2}{2\sigma^2(\mathbf{x})}\Big) \tag{7}$$

with $\mathbf{x}$ and $\mathbf{t}$ representing the network input and answer respectively. Note that $\mathbf{t}$ will only be known on a training set.

The result of using a neural network in this fashion is equivalent to using the error function,

$$E = -\sum_n \log \sum_{k=1}^{M} \alpha_k(\mathbf{x}^n)\phi_k(\mathbf{t}^n|\mathbf{x}^n) \tag{8}$$

as part of a maximum likelihood calculation. That is we minimizes the error with respect to the network weights. We refer to this optimization as network training in later sections. Simple gradient descent with a stochastic updating rule, as outlined in Mitchell (1997), was used to perform the optimizations.

We conclude this section with a synopsis of the weight update rule. For convenience we define the following function of $\mathbf{x}$ and $\mathbf{t}$,

$$\Pi_k = \frac{\alpha_k\phi_k}{\sum_{l=1}^{M} \alpha_l\phi_l}. \tag{9}$$

Output from the i^{uh} unit is denoted z_i. The k^{Te} unit's activation function is $g(a_k)$ with the input defined as the weighted average of the outputs from the previous level,

$$a_k = \sum_l w_{k,i} \cdot z_i. \tag{10}$$

The value of $g(a_k)$ is also the output from unit k. Weight adjustment for hidden units is given by,

$$\delta_k = g'(a_k) \sum_{j\in D(k)} w_{j,k}\delta_j \tag{11}$$

with $D(k)$ denoting the units that are connected to unit k but one level closer to the output layer. This is derived by application of the chain rule when differentiating the error function with respect to the weights.

The output units have more complicated updating rules. There is a separate update rule for each type of output, i.e. one for the prior probabilities, variances, and means. The partials of the error function with respect to the weights for the output units are,

$$\frac{\partial E}{\partial z_k^{\alpha}} = \alpha_k - \Pi_k \tag{12}$$

$$\frac{\partial E}{\partial z_k^{\sigma}} = -\Pi_k\Big(\frac{\|\mathbf{t}-\mu_k(\mathbf{x})\|^2}{\sigma_k^2} - C\Big) \tag{13}$$

$$\frac{\partial E}{\partial z_{k,i}^{\mu}} = \Pi_k\Big(\frac{\mu_{k,i}-t_i}{\sigma_k^2}\Big). \tag{14}$$

These partials are the weight updates for the output layer. They are also used in (11) to calculate the hidden unit values.

We used the stochastic update rule Mitchell (1997) given by,

$$w_{k,i} = w_{k,i} + \eta\, \delta_k z_i \tag{15}$$

with η corresponding to the step size.

A variety of network inputs, predictors, were used. They were limited to the meta information recorded by the WSI and the sky images. The meta data included various camera diagnostics, temperatures, time, day of year, longitude/latitude of the

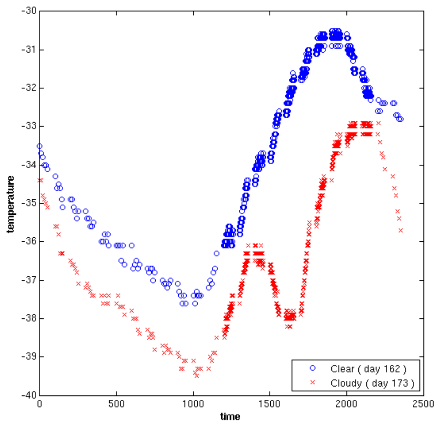


Figure 11: CCD chip temperature vs. time (Z). The blue curve is of a predominately clear day and the red of a predominately cloudy day. Both days are from a summer month and they are only separated by nine days.

WSI, and solar position. Much of this data is not strongly correlated to pixel designations. We eventually made different combinations of the pixel location, pixel intensity, time, solar position, ccd chip temperature, and exposure.

We included time, solar zenith, ccd chip temperature, and exposure to act as global predictors. Time was included due to the fact hat the color of the sky changes through out the day. This is also why the solar zenith angle was used. The ccd chip temperature was indicative of the relative brightness of the sky. On clear days the ccd chip temperature is usually higher than in overcast days. Figure 11 shows a typical plot of chip temperature for a clear and cloudy day. For similar reasons we included the exposure of the camera.

Three network architectures were tried for the *Mixture of Experts*. These were 4x16x9, 6x16x9, and 8x16x9. All hidden layers had hyperbolic tangent activation functions. Cloud designations were encoded as 0.99 and Clear designations were encoded as 0.01. This encoding scheme was chosen for convenience and to help with numerical stability. It limits the dimensionality of the means to one. As our encoding involved only a one dimensional mean then each network had three experts. We also used four experts but difficulties arose during training however either as numerical instability or lack of convergence to a suitable extremal. Also, when we did get satisfactory convergence performance was not noticeably improved.

As noted the instrument takes three whole sky images in quick succession. Each corresponding to the use of a different filter. The combinations of the intensity information were based on several observations. The first was analysts graded images $\sim 99.9\%$ using primarily the NIR images. As such, it was reasonable to attempt to build cloud masks using only these values. Using only NIR intensity values would also act as a baseline. For the larger networks, differences between the NIR and other spectrum were used. On bright days objects in the NIR images are significantly brighter than in the other spectrum. A simple means of using this observation was to have the differences as a predictors.

A variety of predictors for the 4x16x9 networks were used. We experimented with combinations of time, solar zenith angle, row position of pixel, ccd chip temperature, and NIR pixel intensity value from the center of box. After reviewing agreement rates and visual correlations it was clear that time, pixel row position, ccd chip temperature and the pixel intensity gave the best results.

For the 6x16x9 networks we also included pixel column position, ratio of BLU to NIR, ratio of RED to NIR, difference of NIR and BLU, difference of NIR and RED, and max value over all pixels of a given image for BLU, NIR, and RED. Again we used several combinations judging the performance of each. We finally decided upon time, ccd chip temperature, pixel row position, pixel column position, difference of NIR and BLU, and difference of NIR and RED. As this combination provided good performance and an ease of convergence.

Finally, the 8x16x9 used the same predictors as the 6 input networks with the addition of solar zenith angle and camera exposure. Again this was decided after trying several other predictors and weighing ease of convergence, the visual quality of cloud masks, and agreement with our validation set.

5. Training and Results

The clustering network was trained simply by specifying a list of images. The images in this list were picked by inspection with care being given to having an equal quantity of four specific types. We chose predominately clear and cloudy, partially cloudy, and hazy days. Classification was open to the subjectivity of an analyst and some days that were predominately cloudy could also have been considered hazy. An example of each can be found in Figure 10. On average we trained the clustering networks for 12000 iterations. There was not a noticeable improvement beyond this point. Starting and ending α

	% (seed)	% (entire)
#neigh. 1	88.3	82.7
#neigh. 4	90.0	87.0
#neigh. 8	93.3	90.0

Table 2: 4 predictors overall agreement with training set.

	% (seed)	% (entire)
#neigh. 1	95.2	93.9
#neigh. 4	96.7	95.2
#neigh. 8	98.7	96.2

Table 3: 6 predictors overall agreement with training set.

	% (seed)	% (entire)
#neigh. 1	96.0	95.5
#neigh. 4	96.5	96.0
#neigh. 8	99.7	96.6

Table 4: 8 predictors overall agreement with training set.

% agreement	predictors: 4	6	8
#neigh. 1	72.8	83.2	84.4
#neigh. 4	76.4	85.4	84.8
#neigh. 8	78.1	83.4	82.8

Table 5: Average agreement over all lists on validation set.

were 0.999 and 0.001 respectively. The number of neighborhoods ranged from 2 to 64 but as we note later there was difficulty getting our training set to populate more than eight.

To train each *Mixture of Experts* we graded a group of independent images using a completely different template. The template consisted of 265 equally spaced boxes. In addition, the images were graded by a single independent analyst with the caveat that pixels should only be labeled if the designation was certain. This meant that most if not all images were only partially graded, i.e. most boxes for a given image were left ungraded. The final set of graded boxes consisted of 12075 separate examples. This set formed the collection of training examples. An additional training set was constructed to ensure complete independence from the validation set.

Boxes that were labeled uncertain by analyst are usually caused by obstruction from the occulter. We ignored these designations in our training for that reason. This reduced the size of the training set to 10527. As such we did not have the neural networks output this designation. It is a straight forward calculation to track the occulter and mask out the pixels of its location in the cloud masks. This is similar to the procedure for the horizon masking during the WTA clustering calculations. Considering the training process as in Owens and Filkin (1989), it is equivalent to solving a system of coupled ordinary differential equations. It is clear only the output layer has direct external forcing resulting from the error function. Thus restricting the size of the output layer is of considerable help with the numerical stability.

Training began by selecting at random a small sub-sample from the larger training set. The goal was first to get a network trained on this small set and then up train with a larger set of examples. During the course of training we tried several different schemes based on this idea but finally decided that only one small training set per neighborhood was necessary to seed the learning process.

The overall training for each neighborhood consisted of doing a gradient descent on the seed training set until a specified accuracy was reached. After this the network was trained in alternating fash-

	% (seed)	% (entire)	#ex. (seed)	#ex. (entire)
#neigh. 1				
	88.3	82.7	599	10527
#neigh. 4				
	96.6	93.0	89	1356
	95.4	96.2	65	1357
	96.0	91.0	100	1817
	85.5	82.3	345	5997
#neigh. 8				
	100	100	37	745
	100	97.1	80	1533
	100	99.0	11	205
	93.0	91.9	43	754
	91.2	86.6	136	2332
	82.6	79.7	115	2153
	96.7	92.8	90	1377
	97.7	92.9	87	1428

Table 6: 4 predictors agreement with training set by neighborhood.

ion using the larger training set and the seed. This later alternating training was also performed using gradient decent with the same step size as the initial training. The alternating training was terminated when a specified level of accuracy on the seed and larger training sets was reached. The seed set was usually several times smaller. The agreements for the various predictors on the training set can be found in Tables 6, 7, and 8. These are displayed by neighborhood. The overall agreement rates are found in Tables 2, 3, 4. All training was done on-line, i.e. weights were updated after each example.

The number of neighborhoods calculated using a WTA algorithm was restricted by the training set. For more than eight it was difficult to seed each neighborhood. Hence after some initial investigations we decided on one, four, and eight neighborhoods. The initial weights of the neural networks were chosen at random and several different initializations were trained. We then used the one with the highest overall accuracy.

Upon completion of training the various neural networks along with their clustering network are used to build cloud masks. These were then scored using the independently graded images. Sample cloud masks can be found in Figures 12 - 13. White pixels in the cloud masks corresponds to clear sky. It is important to note that these images are completely independent of the training data. These images do not appear in either the training or validation sets. Also each pixel is the most probable designation. The cloud masks correspond to the "$0/1$" mapping defined by the mean of the kernel with highest prior, i.e.

$$\text{arg max}_k\{\alpha_k(\mathbf{x})\}. \tag{16}$$

Relative and absolute accuracies can be found in Table 9, 10, and 11. Also included in these tables is the relative bias of clear and cloud designations. Biases that are less than unity correspond to the under detection of clouds.

6. Conclusion

We have demonstrated that our approach builds reliable cloud masks at a pixel level. As Figures 12-13 show our approach can resolve typical clouds. In addition, the networks are also able to identify objects in images such as the occulter. Note, that the networks were not explicitly trained to identify the occulter. This network generalization was present in most images when using six predictors. The validation results are likely overall too low for other pixels in an image. The images when viewed agree quite well on a macro scale with the cloud masks from our algorithm. Most of the disagreements are from fine structures in the images, e.g. cloud edges. These regions of an image form a low percentage of the overall number of pixels. Given the small sample size of the validation set and the apparent visual similarity the relative agreements are more indicative of performance. Therefore, our best results agree $\sim 90\%$ with human analysts.

Overall judgment of performance was taken to be a combination of visual inspection of a few random

	% (seed)	% (entire)	#ex. (seed)	#ex. (entire)
#neigh. 1				
	95.2	93.9	599	10527
#neigh. 4				
	97.8	98.2	89	1356
	98.5	95.4	65	1357
	100	99.0	100	1817
	95.1	93.1	345	5997
#neigh. 8				
	100	100	37	745
	100	98.2	80	1533
	100	99.0	11	205
	100	98.4	43	754
	98.5	95.9	136	2332
	96.5	92.2	115	2153
	98.9	98.3	90	1377
	98.9	94.7	87	1428

Table 7: 6 predictors agreement with training set by neighborhood.

	% (seed)	% (entire)	#ex. (seed)	#ex. (entire)
#neigh. 1				
	96.0	95.5	599	10527
#neigh. 4				
	98.9	97.9	89	1356
	100	99.6	65	1357
	100	98.2	100	1817
	94.2	94.1	345	5997
#neigh. 8				
	100	99.9	37	745
	100	98.1	80	1533
	100	99.5	11	205
	100	98.3	43	754
	100	95.1	136	2332
	100	94.8	115	2153
	98.9	98.4	90	1377
	98.9	95.7	87	1428

Table 8: 8 predictors agreement with training set by neighborhood.

	% relative	% absolute	bias
#neigh. 1			
image list: 1	81.6	76.8	0.975
2	72.7	67.2	0.742
3	76.2	71.8	0.874
4	81.8	75.2	1.075
5	76.7	73.6	1.024
6	76.4	71.9	0.745
#neigh. 4			
image list: 1	83.3	78.4	0.904
2	83.1	76.9	0.850
3	81.6	76.9	0.898
4	82.5	75.8	1.037
5	79.3	76.1	0.999
6	78.6	74.0	0.756
#neigh. 8			
image list: 1	83.7	78.8	0.824
2	86.4	79.9	0.805
3	86.6	81.5	0.930
4	82.1	75.5	0.917
5	81.2	78.0	0.966
6	79.4	74.7	0.746

Table 9: 4 predictors agreement with validation set.

	% relative	% absolute	bias
#neigh. 1			
image list: 1	87.2	82.1	0.864
2	89.0	82.3	0.809
3	92.2	86.8	0.783
4	90.0	82.7	0.692
5	89.2	85.6	0.740
6	82.7	77.9	0.748
#neigh. 4			
image list: 1	87.4	82.3	0.887
2	92.2	85.3	0.896
3	94.8	89.3	0.851
4	91.3	83.9	0.745
5	92.0	88.3	0.891
6	86.7	81.6	0797
#neigh. 8			
image list: 1	86.7	81.6	0.803
2	91.9	85.0	0.836
3	91.8	86.4	0.820
4	89.9	82.6	0.708
5	87.9	84.4	0.762
6	83.7	78.8	0.778

Table 10: 6 predictors agreement with validation set.

samples and the calculation of agreement with the validation lists. The agreements in Table 5 differ from the worst (4 predictors and 1 neighborhood) to the best (6 predictors and 4 neighborhoods) by only 12.6%. Even though the agreements differ by a few percentage points, cloud masks made with six predictors are markedly better in appearance. Although the use of six predictors instead of four predictors increased overall agreement by $\sim 10\%$, the trend did not continue when we added two additional predictors. In fact, some of the overall agreements decreased as well as the visual correlation between images and their cloud masks.

According to our validation lists and the calculated biases, clouds are being under detected. This is probably a result of our relatively small training set. The training set is not indicative of the seasonal light conditions. Cloud motion between images for a given time is also a cause of error. Recall images are taken in quick succession which means clouds will invariably move causing some cloudy pixels to be take as clear.

Our pre-processing network seems to help most when we use a small number of predictors. The clustering seems to reduce the overall complexity by partitioning the problem domain. Our best re-

	% relative	% absolute	bias
#neigh. 1			
image list: 1	88.4	83.2	0.897
2	90.2	83.4	0.842
3	92.1	86.7	0.802
4	90.5	83.2	0.690
5	90.2	86.6	0.792
6	87.3	82.2	0.847
#neigh. 4			
image list: 1	88.7	83.5	0.821
2	92.0	85.1	0.853
3	93.8	88.3	0.821
4	91.7	84.3	0.692
5	92.3	88.6	0.892
6	80.4	75.7	0.727
#neigh. 8			
image list: 1	86.8	81.7	0.837
2	92.0	85.1	0.844
3	92.0	86.6	0.846
4	87.9	80.8	0.741
5	89.0	85.4	0.833
6	79.4	74.8	0.712

Table 11: 8 predictors agreement with validation set.

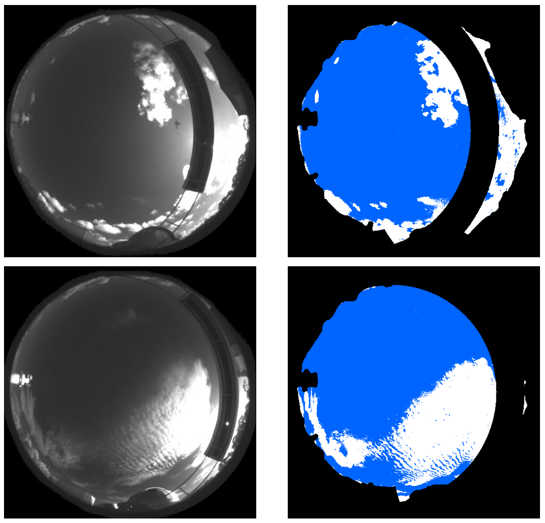


Figure 12: Example cloud masks using 6 predictors and 4 neighborhoods. Black denotes Uncertain, blue denotes Clear, and white denotes Cloud.

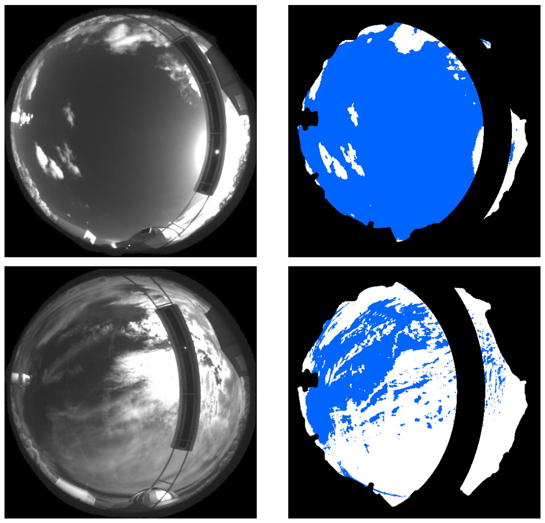


Figure 13: Example cloud masks using 6 predictors and 4 neighborhoods. Black denotes Uncertain, blue denotes Clear, and white denotes Cloud.

sults are when using four clusters. This is likely due to the cluster training set having four types of images. It is also clear that using eight clusters only improves performance when a four predictors are used. This is due to the fact that when the training set is partitioned over eight clusters the neural network for each is being over trained. Note that some of the clusters have only several examples from the seed training sets and only a few hundred considering the entire training set. Using eight or more neighborhoods would likely improve performance if we had a larger and more varied training set.

On a final note, the current approach is usable as a quality control agent for other algorithms. Additional network inputs could be a pixel's designation from other more traditional approaches Keller et al. (2006), Ogler et al. (1991), and Long et al. (2006). In these cases the networks would be post processors and overall quality control agent for these other algorithms. This would provide a straight forward means of incorporating meta data and human knowledge into these traditional imaging algorithms. Thus that with little change to current algorithms and software improved performance could be attained for negligible cost.

References

Bishop, C. M., 1995: *Neural Networks for Pattern Recognition*. Oxford University Press Inc., New York, U.S.A.

Brady, J. M., 1992: *Introduction to Artificial Neural Systems*. West Publishing Company, U.S.A.

Hecht-Nielsen, R., 1987: Counterpropagation networks. *Applied Optics*, **26**, 4979–4984.

Jacobs, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton, 1996: Adaptive mixtures of local experts. *Neural Computation*, **3**, 79–87.

Keller, K., R. Caveney, and R. Link, 2006: Wsi cloud mask generation. Technical report, TASC, 4801 Stonecroft Blvd., Chantilly, VA 20151.

Kohonen, T., 1990: The self-organizing map. *Proceedings of the IEEE*, **78**, 1464–1480.

Long, C. N., J. Sabburg, J. Calbó, and D. Pagès, 2006: Retrieving cloud characteristics from ground-based daytime color all-sky images. *Journal of Atmospheric and Oceanic Technology*, **23**, 633–652.

Mitchell, T. M., 1997: *Machine Learning*. WCB/McGraw-Hill, New York, U.S.A.

Ogler, T. L., R. Johnson, and J. M. Shields, 1991: Status of the whole sky imager database. *Proc. Cloud Impacts on DOD Operations and Systems - 1991 Conference*, Department of Defense, pp. 77-80.

Owens, A. J. and D. L. Filkin, 1989: Efficient training of the back propagation network by solving a system of stiff ordinary differential equations. *Proc. Internationl Joint Conference on Neural Networks - 1989 Conference*, IEEE, pp. 381-386.