Temporal Updating Scheme for Probabilistic Neural Network with Application to Satellite Cloud Classification

Bin Tian, Mahmood R. Azimi-Sadjadi, *Senior Member, IEEE,* Thomas H. Vonder Haar, and Donald Reinke

***Abstract*—In cloud classification from satellite imagery, temporal change in the images is one of the main factors that causes degradation in the classifier performance. In this paper, a novel temporal updating approach is developed for probabilistic neural network (PNN) classifiers that can be used to track temporal changes in a sequence of images. This is done by utilizing the temporal contextual information and adjusting the PNN to adapt to such changes. Whenever a new set of images arrives, an initial classification is first performed using the PNN updated up to the last frame while at the same time, a prediction using Markov chain models is also made based on the classification results of the previous frame. The results of both the old PNN and the predictor are then compared. Depending on the outcome, either a supervised or an unsupervised updating scheme is used to update the PNN classifier. Maximum likelihood (ML) criterion is adopted in both the training and updating schemes. The proposed scheme is examined on both a simulated data set and the Geostationary Operational Environmental Satellite (GOES) 8 satellite cloud imagery data. These results indicate the improvements in the classification accuracy when the proposed scheme is used.**

***Index Terms*—Cloud classification, Markov chain models, maximum likelihood, probabilistic neural networks, temporal updating.**

I. Introduction

SATELLITE imagery has provided us with both global and local views of our planet and the atmosphere. The Geostationary Operational Environmental Satellite (GOES)-8 is sending back five spectral channel images of the earth at intervals as short as 1 min. These images generally capture prominent changes of clouds and the earth. Traditionally, these data are inspected visually by meteorologists to determine cloud types and a wide range of significant weather patterns such as fronts, cyclones and thunderstorms. Owing to the huge volume of data (25 GB per GOES 8 satellite) received every day, manual interpretation of all of these images becomes a very tedious and sometimes impractical task. Consequently, the potential of the satellite imagery may not be fully exploited. Therefore, highly efficient and robust cloud detection/classification schemes are needed for automatic processing of the spatial–temporal satellite cloud imagery for climatological and many other relevant applications.

In recent years, considerable research has been focused on cloud classification. A good overview of such efforts is provided by Pankiewicz [1]. Various feature extraction approaches have been examined. The spectral information, which is comprised of a set of radiance measurements of clouds at different bands, was used in [2], [3]. Textural features, which are often distinct and tend to be less sensitive to the effects of atmospheric attenuation or detector noise, have received more attention in recent years. Welch *et al.* [4] calculated several statistical textural features based on gray level co-occurrence matrix (GLCM) while Lamei *et al.* [5] used Gabor filters to extract cloud textural features. Several comparative studies of both the spectral and textural features for cloud and other satellite imagery classifications have been conducted by Parikh [6], Gu [7] and Ohanian [8] and Augusteijin [9] which led to the conclusion that no consistent optimal feature extraction scheme can be devised for this problem. The other important issue in the literature is the choice of the appropriate classifier for the cloud classification problem. Both the traditional statistical and neural network classifiers have been employed for this application. Simpson and Gobat [10], [11] used a nested hierarchical partitional clustering algorithm to segment GOES images and an adaptive thresholding to label clusters as cloudy or cloud free classes. Welch [4] used linear discrimination techniques while Lee *et al.* [12] tested a three-layer back-propagation neural network (BPNN). A Probabilistic neural network (PNN) was also examined by Bankert *et al.* [13]. In [14], these three classifiers were benchmarked for the polar cloud and surface classification. The results showed that BPNN-based solution achieves the highest classification accuracy, while PNN falls behind within a very small accuracy measure but requires much less training time compared to the BPNN-based solution. Owing to the fact that in most of the situations the truth maps of clouds and background areas may not be available or reliable, and further a large volume of satellite images is generally encountered, an unsupervised neural network solution was also exploited in [15]. In spite of the previous and on-going research efforts in this area, automatic cloud classification schemes are still far from being practical. This is due, mainly, to the fact that the characteristics of clouds are highly variable and difficult to define. Moreover, most of the studies have only examined one or several images that were obtained

Manuscript received October 11, 1997; revised August 3, 1998 and June 17, 1999. This work was supported by the Department of Defense under the Contract DAAH04 94 G0420.

B. Tian and M. R. Azimi-Sadjadi are with the Department of Electrical Engineering, Colorado State University, Fort Collins, CO 80523 (e-mail: azimi@engr.colostate.edu).

T. H. Vonder Haar and D. Reinke are with the Cooperative Institute for Research in the Atmosphere (CIRA), Colorado State University, Fort Collins, CO 80523.

Publisher Item Identifier S 1045-9227(00)06004-5.

at a certain time of the day (generally noon time). Only a few studies considered processing a series of cloud images, where the temporal variation of the data must be taken into account.

The temporal factor is extremely important in satellite cloud imagery classification and other remote sensing applications. Generally speaking, there are two kinds of the temporal factors. The first one is the temporal contextual information (short-term). Since clouds and background areas are unlikely to move or change significantly over short time intervals (one hour), there is a strong correlation between two consecutive images. It is widely known in remote sensing that proper utilization of this temporal contextual information can help to improve the classification accuracy [16]. A number of temporal contextual-based classifiers have been proposed such as cascade classifier [17] and stochastic model based schemes [18]. The second type of temporal factor that must be considered corresponds to longer term temporal changes. As time elapses, certain types of clouds will "look" different in the visible channel due to changes in the sun angle. At the same time, the ground and low level clouds will be heated during the daytime and cooled at night producing textural and radiative changes in the infrared (IR) channel. All of these changes will be reflected in the satellite imagery and hence affect the feature vectors. Although these variations may not be very prominent in short term, they can accumulate over time. Thus, a fixed neural network may not be able to deal with a sequence of images obtained at different time of the day. There are basically three broad categories of solutions to alleviate these problems caused by temporal changes. The first category of approaches attempts to find the features that are somewhat insensitive to temporal changes. However, this itself is a difficult task. The second class of solutions introduces the temporal factor to the neural network classifier. For example, the time at which the image is obtained can be used as an input parameter to the classifier. One can also design a number of neural networks that correspond to different times and seasons. However, one obvious drawback for these solutions is that a substantial amount of data must be included in the training set in order to accurately represent the trend of all possible temporal changes. Moreover, the useful temporal context information is neglected. The third class of solutions involves the design of a neural network classifier that can update itself to accommodate the temporal changes. Unlike the global classification schemes that use AVHRR, this type of classifier is primarily designed to work in a specific regional area where there are high time interval (geostationary) data and, so far, only during daylight hours when both visible and IR data are available. The idea behind this approach is to identify the changes and then make suitable adjustments to the neural network classifier. The main difficulty in this updating process is that "truth maps" are not available. This problem is addressed in this paper by developing a novel temporally adaptive neural network-based cloud classification scheme exploiting the temporal contextual information. A PNN is used as the classifier due to its good generalization ability and fast learning capability, which are crucial for on-line updating [19]. A Markov chain-based predictor is also designed, which makes the initial guess based on the temporal contextual information. The results of the PNN, updated to the last frame, and the predictor results are then compared. Depending on the match between the results of the classifier and the predictor, either a supervised or an unsupervised learning scheme is used to update the PNN. The well-known Expectation-Maximization (EM) method, which can implement both the supervised and unsupervised learning into one procedure, is employed for the updating process. The proposed temporal updating scheme can also be applied to a number of other important applications where spatial–temporal classification of a sequence of images is needed.

The organization of this paper is as follows. A brief review of the PNN is given in Section II. In Section III, the proposed temporal updating scheme for PNN is discussed. Experimental results on both the simulated data and GOES 8 satellite data are presented in Section IV. Section V provides the conclusion and comments.

II. Probabilistic Neural Network (PNN)

PNN is a supervised neural network that is widely used in the area of pattern recognition, nonlinear mapping, and estimation of the probability of class membership and likelihood ratios [20]. It is closely related to Bayes classification rule and Parzen nonparametric probability density function (PDF) estimation theory [21].

Consider an input feature vector $\boldsymbol{x}$, where $\boldsymbol{x}$ is a d-dimensional vector which belongs to one of the K classes, c_i, $i = 1, 2, \cdots, K$. A classifier can be regarded as a mapping, C: $R^d \rightarrow \{c_1, c_2, \cdots, c_K\}$ that classifies the given pattern $\boldsymbol{x}$ to class $C(\boldsymbol{x})$. Suppose that the class conditional distribution, $p(\boldsymbol{x}|c_i)$, and the *a priori* class probability $P(c_i)$ are known, then the best classifier which can minimize the defined cost function is given by the fundamental Bayesian decision rule [22]. When the 0-1 cost function, which implies minimum classification error rate, is adopted, the Bayes classifier becomes the maximum *a posterior* (MAP) classifier, i.e.,

$$C(\boldsymbol{x}) = \arg\max_{c_i} P(c_i)p(\boldsymbol{x}|c_i) \qquad i = 1, 2, \cdots, K \quad (1)$$

One main concern when implementing the above optimal Bayesian classifier is to estimate the class-conditional probability density function $p(\boldsymbol{x}|c_i)$ and the *a priori* class distribution $P(c_i)$ from the training data set and then use the resultant estimates as if they were the true values. Generally, the *a priori* class distribution is highly dependent on the specific task and should be decided by the physical knowledge of the problem. It is often assumed to be uniformly distributed when no physical knowledge is available. For the sake of convenience, uniform distribution assumption for $P(c_i)$ is adopted here and thus only the estimation of class-conditional probability density will be discussed in the sequel. However, all the neural network structures and training schemes proposed later can easily be extended to accommodate the situations where the a prior class distribution is not uniform.

There are basically two categories of schemes for the traditional density estimation: *parametric* approaches which model the class-conditional densities as multivariate Gaussian and then estimate the necessary parameters from the training data set; and the *nonparametric* density estimation. In [23], Parzen proved that $p(\boldsymbol{x}|c_i)$'s can be estimated from all the

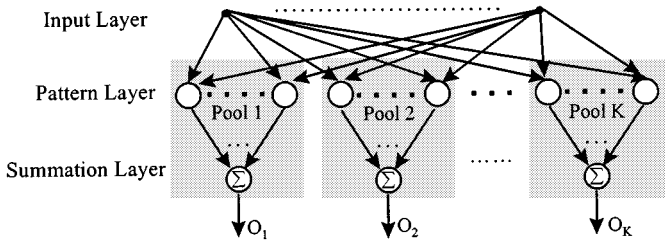

Fig. 1. Structure of probabilistic neural network.

samples in the training set which belong to the class c_i. When a Gaussian kernel function is adopted, the Parzen probability density function estimator can be represented by [21]

$$p(\boldsymbol{y}|c_i) = \frac{1}{N_i(2\pi)^{d/2}\sigma^d} \sum_{j=1}^{N_i} \cdot \exp\left[-\frac{\left(\boldsymbol{y}-\boldsymbol{x}_i^{(j)}\right)^T\left(\boldsymbol{y}-\boldsymbol{x}_i^{(j)}\right)}{2\sigma^2}\right] \quad (2)$$

where

- N_i is the number of samples in the training set that belong to class c_i,
- $\boldsymbol{x}_i^{(j)}$ represents the jth sample belonging to class c_i,
- d is the input vector dimension and is called the "smoothing factor."

The general PNN structure, which was proposed by Specht in [21], is a direct implementation of the above PDF estimator and Bayesian decision rule. It consists of three feed-forward layers: input layer, pattern layer, and summation layer [21] which are shown in Fig. 1. The input layer accepts the feature vectors and supplies them to all of the neurons in the pattern layer. The pattern layer consists of K pools of pattern neurons corresponding to K classes. In each pool, i, there are N_i number of pattern neurons. For any input feature vector $\boldsymbol{y}$, the output of each pattern neuron is given by

$$f\left(\boldsymbol{y}; \boldsymbol{w}_i^{(j)}, \sigma\right) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\left[-\frac{\left(\boldsymbol{y}-\boldsymbol{w}_i^{(j)}\right)^T\left(\boldsymbol{y}-\boldsymbol{w}_i^{(j)}\right)}{2\sigma^2}\right] \quad (3)$$

where $\boldsymbol{w}_i^{(j)}$ is the weight vector of the jth neuron in the ith pool, and the nonlinear function $f(\cdot)$ represents the activation function of the neurons. There are totally K neurons in the summation layer where the ith neuron, $i = 1, \cdots, K$, forms the weighed sum of all the outputs from the ith pool in the pattern layer. The weights are determined by the decision cost function and the *a priori* class distribution. For the "0-1" cost function and uniform *a priori* distribution, all the weights are $1/N_i$ for the ith neuron. Comparing (2) and (3), it is very easy to see that the output of the ith summation neuron in the PNN is simply the estimated class-conditional PDF for class c_i, when $\boldsymbol{w}_i^{(j)} = \boldsymbol{x}_i^{(j)}$. Furthermore, the PNN becomes the Bayesian classifier in (1) if the classification decision is made by simply comparing the outputs of the summation neurons.

From the above analysis, the training of the PNN is very straightforward. For each new training sample $\boldsymbol{x}$ belonging to class c_i, the training process adds a new neuron in the ith pool of the pattern layer, with the weight vector which is $\boldsymbol{x}$. Although, this noniterative training procedure is very fast, a very large network may be formed since every training pattern needs to be stored. This leads to extensive storage cost and computation time during the testing phase.

One technique for improving the PNN is to reduce the number of neurons, i.e., use fewer kernels but place them at the optimal places. Several schemes using Kohonen and learning vector quantization (LVQ) have been proposed for clustering the training samples [24]. In [25], [26], Streit *et al.* improved the PNN by using finite Gaussian mixture models. This neural network structure is adopted in this paper and briefly described below.

For any class c_i, $i = 1, \cdots, K$, suppose the class conditional distribution is modeled approximately by a Gaussian mixture, i.e.,

$$p(\boldsymbol{x}|c_i) = \sum_{j=1}^{M_i} \pi_{ji} p_{ji}(\boldsymbol{x}; \mu_{ji}, \Sigma_{ji}) \quad (4)$$

where M_i is the number of Gaussian components in class c_i and π_{ji}'s are the weights of the components which satisfy the constraint $\sum_{j=1}^{M_i} \pi_{ji} = 1$, $p_{ji}(\boldsymbol{x}; \mu_{ji}, \Sigma_{ji})$ denotes the multivariate Gaussian density function of the jth component in class c_i. Further, we have

$$p_{ji}(\boldsymbol{x}; \mu_{ji}, \Sigma_{ji}) = \frac{1}{(2\pi)^{d/2}|\Sigma_{ji}|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-\mu_{ji})^t \Sigma_{ji}^{-1}(\boldsymbol{x}-\mu_{ji})\right\} \quad (5)$$

where μ_{ji} and Σ_{ji} are the mean vector and covariance matrix of the jth Gaussian component for class c_i, respectively. The Gaussian mixture model described in (4) and (5) can also be easily mapped to the PNN structure. For the ith pool in the pattern layer, only M_i neurons are needed. The weight set associated with each pattern neuron is $\{\mu_{ji}, \Sigma_{ji}\}$, $i = 1, \cdots, K$, $j = 1, \cdots, M_i$, and the input–output relation is specified by (5). In the summation layer, the weight from jth neuron in pool i of pattern layer to the ith neuron in the summation layer is π_{ji}. By configuring the PNN in this way, the output of the PNN will be the same as the Gaussian mixture model output given by (4) and (5). Since generally M_i is much smaller than the number of training samples that belong to class i, N_i, the pattern layer of the PNN is therefore substantially simplified from its original version. The price paid for this simplification is the elimination of the noniterative training procedure. Instead, the weights of the PNN, i.e. the parameter sets of the mixture model for each class, need to be estimated from the training data set.

Let $\lambda_i = \{\pi_{ji}, \mu_{ji}, \Sigma_{ji}\}_{j=1}^{M_i}$ denote the parameter set used to describe the mixture model of class c_i and $\Lambda = \{\lambda_i\}_{i=1}^{K}$ denote the whole parameter space for the PNN. There are several methods available that can be used to estimate Λ. If we assume that the parameters in Λ are unknown fixed quantities, the maximum likelihood (ML) estimation method is a suitable choice.

Now suppose that the training samples drawn independently from the feature space form the set X, which can be further separated into K subsets X_i, $i = 1, \cdots, K$, in which all the samples belong to class c_i. The ML estimation of parameter set Λ is then given by

$$\Lambda^* = \arg\max_{\Lambda} \prod_{i=1}^{K} \prod_{\boldsymbol{x}\in X_i} p(\boldsymbol{x}|c_i;\, \Lambda) \tag{6}$$

For the computational efficiency, generally we will maximize the equivalent log-likelihood, i.e.,

$$\begin{aligned}\Lambda^* &= \arg\max_{\Lambda} \sum_{i=1}^{K} \sum_{\boldsymbol{x}\in X_i} \log[p(\boldsymbol{x}|c_i;\, \Lambda)] \\ &= \arg\max_{\Lambda} \sum_{i=1}^{K} \sum_{\boldsymbol{x}\in X_i} \log[p(\boldsymbol{x}|c_i;\, \lambda_i)]\end{aligned} \tag{7}$$

The last step in (7) is arrived at based upon the assumption that the conditional probability of class c_i is solely decided by the parameter set of that class, λ_i and not by the parameter set of the other classes. The maximization of the log-likelihood function can be done using a gradient descent scheme. However, a more efficient way is to use the well-known expectation-maximization (EM) approach, which was proposed by several researchers including Dempster [27]. The EM approach can help to achieve the maximum-likelihood estimation via iterative computation when the observations are viewed as incomplete data. There are two major steps in this approach: the expectation (E) step and maximization (M) step. The E step extends the likelihood function to the unobserved variables, then computes an expectation with respect to them using the current estimate of the parameter set. In the M step, the new parameter set is obtained by maximizing the resultant expectation function. These two steps are iterated until convergence is reached. The reader is referred to [27] for the detailed description on the EM algorithm. For the ML training of PNN, it is shown [26] that the parameter set can be estimated iteratively using the following E and M steps:

- E Step

$$\begin{aligned}&E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right] \\ &\quad = \frac{p_{ji}\left(\boldsymbol{x};\, \mu_{ji}^{\text{old}},\, \Sigma_{ji}^{\text{old}}\right)\pi_{ji}^{\text{old}}}{\sum_{m=1}^{M_i} p_{mi}\left(\boldsymbol{x};\, \mu_{mi}^{\text{old}},\, \Sigma_{mi}^{\text{old}}\right)\pi_{mi}^{\text{old}}} \\ &\qquad i = 1, \cdots, K, \text{ and } j = 1, \cdots, M_i\end{aligned} \tag{8a}$$

where $z_{j|i}(\boldsymbol{x})$ is a random variable indicating which Gaussian component generates the observation pattern. For a sample $\boldsymbol{x}$ belonging to class c_i, variable $z_{j|i}(\boldsymbol{x})$ is defined by

$$z_{j|i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if feature } \boldsymbol{x} \text{ comes from } j\text{th Gaussian component in class } c_i \\ 0, & \text{otherwise} \end{cases}$$

which is called unobserved variable in the EM approach [27], [28].

- M Step: The new parameter set can be estimated by

$$\pi_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right]}{\sum_{m=1}^{M_i}\sum_{\boldsymbol{x}\in X_i} E\left[z_{m|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right]} \tag{8b}$$

$$\mu_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X_i} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right]} \tag{8c}$$

and

$$\Sigma_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right](\boldsymbol{x}-\mu_{ji})(\boldsymbol{x}-\mu_{ji})^t}{\sum_{\boldsymbol{x}\in X_i} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\, \Lambda^{\text{old}}\right]} \tag{8d}$$

The convergence property of this two-step iterative procedure is proven in [27]. Although the EM scheme may also end up with a local minimum, like the gradient descent based scheme, it generally converges much faster. There is one important observation from (8a)–(8d): the estimation for parameter set of class c_i is only dependent on the training samples in this class, i.e, the optimization process can be solved separately for each class without considering the effect of the others. This is especially suitable for the cloud classification application since a new cloud type can easily be added to the system without affecting the other classes. Moreover, in the updating process, we have the choice of only updating those classes that are affected by the temporal changes. Another benefit of this property is the reduced training time due to the fact that each class can be trained separately, thus requiring a small number of neurons and training samples. On the other hand, the price one pays for this property is that the classification accuracy is dependent on the real distribution of the feature space. If the Gaussian mixture model is a good assumption, the classifier will achieve high accuracy. Otherwise, the trained PNN may not be optimal in the sense of minimum classification error. Although the number of Gaussian components can be increased so that the mixture model can be approximated to any distribution (it becomes a Parzen window PDF estimation when the number of components is equal to the number of training samples), the computational cost may become unacceptable. The number of Gaussian components is generally decided experimentally.

III. Temporal Updating of the Probabilistic Neural Network (PNN)

Once the parameter set is estimated using EM, the PNN can be applied to the satellite cloud image classification problem [29]. However, the temporal factor must be taken into account when processing a sequence of satellite images. Considering two satellite images of the same area but obtained at different times, three kinds of changes are generally observed in these images. The first one is the spatial movement of certain clouds that can be mainly due to wind effects. This change can be modeled by a Markov process. The second type of variation is the class

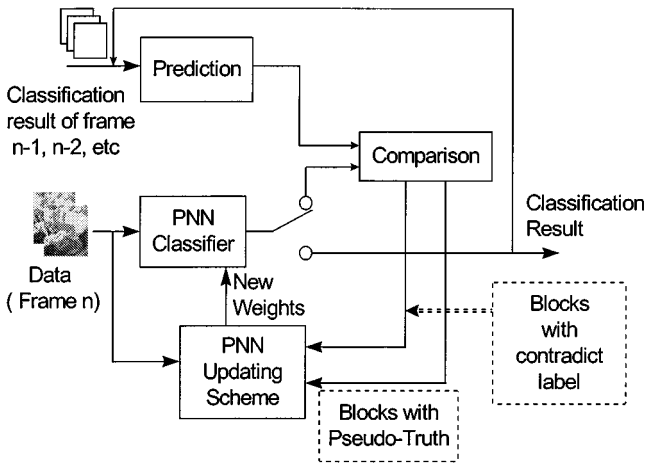

Fig. 2. Block diagram of the proposed temporal updating scheme.

transition, e.g., certain clouds may be generated, terminated or evolve to other classes. Although these variations commonly occur in sequences of satellite cloud images, they will not usually affect the performance of the classifier unless a new class is generated. Nonetheless, if they are modeled properly, they can even help to improve the classification accuracy. The third type of variation is due to the temporal changes of the features. A number of factors such as sun angle and ground heating/cooling effects impact the features of clouds and background areas. Although the temporal changes of the feature space changes are not so prevalent over a short time period, say 1/2 or 1 hours, this effect accumulates and finally will degrade the performance of the classifier significantly. One solution to this problem is to frequently update the classifier to accommodate such changes. In this section, a novel temporal adaptation scheme for the PNN is proposed. The block diagram of the proposed scheme is shown in Fig. 2.

Suppose that the cloud images at the previous time frame, up to frame $n-1$, are correctly classified and the weights of the neural network have been updated to frame $n-1$. Now, the new frame n which includes both visible (ch1) and IR (ch4) channels arrives. The new images go through the feature extraction stage [29] and will then be applied to the PNN classifier. If the interval between the adjacent frames is short enough (1/2 or 1 h for GOES 8 satellite data), the changes of these features will be minimal, hence the old PNN can still correctly classify most of the data. Due to the rich temporal class contextual information between adjacent frames, i.e., most of the cloud and background type (land/water) won't move or change abruptly to other types, a prediction can be made based on the classification results of the previous frame. The initial classification result of the PNN and the output of the predictor are then compared. If the classification label of an image block is the same for both the PNN and predictor, then that block is classified with a high level of confidence. We refer to this type of class information as "pseudo truth." All the blocks of this kind will form the set $X^{(1)}$. On the other hand, all of the blocks for which the old PNN and the predictor provide different class labels form the data set $X^{(2)}$. Both data sets, $X^{(1)}$ and $X^{(2)}$ are used for the PNN updating even though the learning mechanisms are different. After the temporal adjustment, the new weights of the PNN will be used to classify the image again and generate the final result for this frame n. This process will be repeated whenever a new frame has arrived. In the following sub-section, both the prediction process and the PNN updating schemes will be described in detail.

A. Prediction

The prediction block in Fig. 2 is designed to provide an initial guess of the class of the current data based on the previous classification results. Its feasibility lies in the fact that there is rich temporal contextual information between adjacent frames. In [16], the contextual information is classified into two main categories: "class dependent" and "correlation-based," both of which exist spatially and temporally. The first category includes all the information on class distribution. For example, certain positions in the image tend to belong to the same class in the adjacent frame (temporal class dependency context). In the meantime, most of the classes are likely to cover a relatively large area in one frame instead of appearing in the isolated blocks (spatial class dependency context). There will be a rich class dependency context in satellite imagery series as long as the time interval between frames is not too long compared to cloud movement and changes. This assumption is typically true for most of the GOES imagery data where the time interval between frames is relatively short i.e. 30 min to 1 h. The second kind of contextual information, i.e., correlation-based, refers to the characteristics of the feature distribution in the adjacent (spatial) block or (temporal) frames. This information can help to differentiate between classes. However, this is more difficult to model and generally incurs significant computational cost. Since the initial guess is sufficient in the prediction block, only the spatial–temporal class dependency information is used here.

The spatial–temporal class contextual information is generally modeled by a Markov chain. In order to simplify the computations, 1st order Markov chain is considered here, i.e, the class of the current frame is solely dependent on the previous one. Moreover, for a block in the current frame n, we define its spatial temporal neighborhood in frame $n-1$ and assume that all the temporal class contextual information for that block is conveyed by its spatial–temporal neighborhood in frame $n-1$.

Assume $\boldsymbol{x}(\boldsymbol{r};\, n)$ denotes the feature vector of block $\boldsymbol{r}$ in frame n, where $\boldsymbol{r} = (k, l)$ is the location vector of that block in the image and $c(\boldsymbol{r};\, n)$ refers to the physical class of that block. The spatial–temporal neighborhood of block $\boldsymbol{r}$ in frame n is defined in frame $n-1$ as $H(\boldsymbol{r};\, n) = \{\boldsymbol{x}(\boldsymbol{r}+\boldsymbol{v};\, n-1) | \boldsymbol{v} \in \Psi\}$ where Ψ is the neighborhood defining set. An example of such neighborhood is shown in Fig. 3. Furthermore, $C_H(\boldsymbol{r};\, n) = \{c(\boldsymbol{r}+\boldsymbol{v};\, n-1) | \boldsymbol{v} \in \Psi\}$ represents the class label of $H(\boldsymbol{r},\, n)$.

Now let us assume that the previous frames up to frame $n-1$ have correctly been classified, i.e., $C_H(\boldsymbol{r};\, n)$ is known for block $\boldsymbol{r}$. For the current frame n, we want to predict the label of block $\boldsymbol{r}$ given the classification result of its spatial–temporal neighborhood, $C_H(\boldsymbol{r};\, n)$. If the *a posteriori* conditional probability $P(c(\boldsymbol{r};\, n) = c_i | C_H(\boldsymbol{r};\, n))$ is known, then the MAP predictor can be simply implemented as

$$\hat{c}(\boldsymbol{r};\, n) = \arg\max_{c_i} P(c(\boldsymbol{r};\, n) = c_i | C_H(\boldsymbol{r};\, n)) \quad i = 1, \cdots, K \tag{9}$$

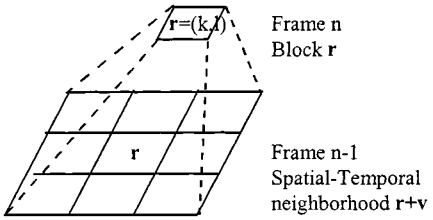

Fig. 3. Example of a temporal neighborhood. In this case, $\boldsymbol{v} \in \Psi = \{(-1,-1), (-1,0), (-1,1), (0,-1), (0,0), (0,1), (1,-1), (1,0), (1,1)\}$.

Although the idea is quite simple, it is very difficult to find the conditional probability, $P(c(\boldsymbol{r};n) = c_i | C_H(\boldsymbol{r};n))$, which specifies the spatial–temporal class dependent context. One straightforward scheme is to estimate it from the training set. However, this approach will rarely produce meaningful results in practice. This is mainly due to the fact that there are so many different combinations of $C_H(\boldsymbol{r};n)$ and $c(\boldsymbol{r};n)$ that there may not be enough training samples. Furthermore, some combinations may not even appear in the training set. Therefore, it is more feasible to specify a model for this conditional probability based on the physical background of the problem. We assume that there are two underlying Markov chains. The first one describes the spatial movement in the image. For the object in block $\boldsymbol{r}$ in the current frame may come from any block in its spatial–temporal neighborhood in the previous frame. A random vector $\boldsymbol{\xi}$ is defined to represent this spatial movement, i.e., the object in block $\boldsymbol{r}+\boldsymbol{\xi}, \boldsymbol{\xi} \in \Psi$ in frame $n-1$ will move into block $\boldsymbol{r}$ in frame n. On the other hand, the second Markov chain describes the possible class change of that object. The class transition Markov chain is needed otherwise the current block will always be one of the types appeared in its spatial temporal neighborhood, which is not the case in real situations. Applying this model and using total probability, we can write

$$\begin{aligned} P(c(\boldsymbol{r};n) = c_i | C_H(\boldsymbol{r};n)) &= \sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r};n) = c_i, \boldsymbol{\xi} = \boldsymbol{v} | C_H(\boldsymbol{r};n)) \\ &= \sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r};n) = c_i | \boldsymbol{\xi} = \boldsymbol{v}, C_H(\boldsymbol{r};n)) \\ &\quad \cdot P(\boldsymbol{\xi} = \boldsymbol{v} | C_H(\boldsymbol{r};n)) \end{aligned} \tag{10}$$

For the sake of simplification, two assumptions are made as follows:

- For the spatial transition Markov chain, we assume that

$$P(\boldsymbol{\xi} = \boldsymbol{v} | C_H(\boldsymbol{r};n)) = P(\boldsymbol{\xi} = \boldsymbol{v}) \tag{11}$$

  i.e., the spatial transition probability is only decided by the relative position of that block in the spatial–temporal neighborhood and is independent of the class label in its spatial–temporal neighborhood.
- When the content of block $\boldsymbol{r}+\boldsymbol{xi}$ in frame $n-1$ moves to the block r of the current frame, its class may also change. It is reasonable to assume that such class transition is solely dependent on the class of that object in frame $n-1$, i.e, $c(\boldsymbol{r}+\boldsymbol{\xi};n-1)$ and is not related to the label of other blocks in the spatial–temporal neighborhood. Thus,

$$P(c(\boldsymbol{r};n) | \boldsymbol{\xi}, C_H(\boldsymbol{r};n)) = P(c(\boldsymbol{r};n) | c(\boldsymbol{r}+\boldsymbol{\xi};n-1)) \tag{12}$$

In fact, the above assumptions imply that the two Markov chains are statistically independent of each other. Under these assumptions, the conditional probability $P(c(\boldsymbol{r};n) = c_i | C_H(\boldsymbol{r};n))$ in (10) becomes

$$\begin{aligned} &P(c(\boldsymbol{r};n) = c_i | C_H(\boldsymbol{r};n)) \\ &\quad = \sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r};n) = c_i | c(\boldsymbol{r}+\boldsymbol{v};n-1)) P(\boldsymbol{\xi} = \boldsymbol{v}) \end{aligned} \tag{13}$$

and the predictor in (9) can be computed as

$$\begin{aligned} \hat{c}(\boldsymbol{r};n) = \arg\max_{c_i} &\sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r};n) = c_i | c(\boldsymbol{r}+\boldsymbol{v};n-1)) \\ &\cdot P(\boldsymbol{\xi} = \boldsymbol{v}) \end{aligned} \tag{14}$$

The position and class transition probabilities can be computed from the training data or decided based on the physical background of the problem. For example, for the case of the spatial–temporal neighborhood in Fig. 3, the spatial transition probability can take on the following values:

$$P(\boldsymbol{\xi} = \boldsymbol{v}) = \begin{cases} 0.2, & \text{for } \boldsymbol{v} = (0,0) \\ 0.1, & \text{for } \boldsymbol{v} \in \{(0,\pm1), (\pm1,0), (\pm1,\pm1)\} \end{cases} \tag{15}$$

Since the objects are generally not likely to change positions in adjacent frames, a relatively large position transition probability is given to this situation, i.e. for $\boldsymbol{v} = (0,0)$. The class transition probability can be heuristically specified in a similar way, i.e., retaining the same class has a higher probability, while the transition probability to another class is much lower. This kind of distribution can be represented by

$$P(c(\boldsymbol{r};n) = c_i | c(\boldsymbol{r}+\boldsymbol{v};n-1) = c_j) = \begin{cases} \alpha, & \text{if } c_i = c_j \\ \dfrac{1-\alpha}{K-1}, & \text{otherwise} \end{cases} \tag{16}$$

where K is the total number of classes and α is a user defined number between 0 and 1 that satisfies $\alpha > (1-\alpha)/(K-1)$. It is proved in Appendix B that the value of α is not important for the final prediction result as long as the class transition probability takes this form. Furthermore, the predictor in (16) is equivalent to a simpler form

$$\hat{c}(\boldsymbol{r};n) = \arg\max_{c_i} \sum_{\boldsymbol{v}\in\Psi} \delta(c(\boldsymbol{r}+\boldsymbol{v};n-1), c_i) P(\boldsymbol{\xi} = \boldsymbol{v}) \tag{17}$$

where

$$\delta(c_i, c_k) = \begin{cases} 1, & c_i = c_k \\ 0, & \text{otherwise,} \end{cases}$$

is similar to the Kronecker delta function.

It is clear that accurate prediction can not be achieved only based on the temporal contextual information. However, if the output of the PNN achieved the same classification result for the same block, then a much higher confidence can be assured. All the blocks that have the same labels from both the predictor and

the PNN updated to frame $n-1$ form the data set $X^{(1)}$ while the others form the set $X^{(2)}$. In the following section, these two data sets will be used to update the PNN.

B. PNN Temporal Updating Scheme

The updating process of PNN is a type of "on-line training." There are basically two requirements for the updating process. First, the updating process must be stable, i.e., the updated PNN must maintain good classification capability for those previously established categories. Second, the updating must be plastic to accommodate temporal changes of the data and new class generation. Note that the only truth available for comparison in this updating process is the "pseudo truth" obtained by utilizing the temporal contextual information and the old classifier's results. However, this class information will be used as if it is the truth.

Assume that the training samples drawn independently from the current feature space form the training set X which is basically the set of features for frame n. This training set can be further separated into two sets: $X^{(1)}$ and $X^{(2)}$, where $X^{(1)}$ includes all of the samples for which the class label is assumed to be known, while all the samples of unknown types belong to $X^{(2)}$. Moreover, let $X_i^{(1)}$, $i=1,\cdots,K$, denote a subset of $X^{(1)}$ in which all the samples are known to belong to c_i. The neural network structure is the same as that discussed in Section II and does not change in the updating process. The goal is to re-estimate the parameter set for PNN so that it can more accurately represent the distribution of the temporally changed feature space. Maximum likelihood (ML) criterion is adopted in this PNN updating process.

Three types of cost functions are considered based on whether or not the pseudo truth information and the whole data set are used in the training process:

1) Cost Function 1: Exclusively Unsupervised Training:

$$\begin{aligned} F_1(X;\Lambda) &= \sum_{\boldsymbol{x}\in X}\log(p(\boldsymbol{x};\Lambda)) \\ &= \sum_{\boldsymbol{x}\in X}\log\left[\sum_{i=1}^{K} p(\boldsymbol{x},c_i;\Lambda)\right] \\ &= \sum_{\boldsymbol{x}\in X}\log\left[\sum_{i=1}^{K} p(\boldsymbol{x}|c_i;\Lambda)P(c_i)\right] \end{aligned} \tag{18}$$

In this cost function, no class information from training samples was used. The training result is completely decided upon by the distribution of the features, which may not guarantee a good classifier and the stability requirement. As a result, this cost function is not a suitable choice.

2) Cost Function 2: Exclusively Supervised Training:

$$\begin{aligned} F_2(X;\Lambda) &= \sum_{\boldsymbol{x}\in X^{(1)}}\log(p(\boldsymbol{x};\Lambda)) \\ &= \sum_{i=1}^{K}\sum_{\boldsymbol{x}\in X_i^{(1)}}\log(p(\boldsymbol{x}|c_i;\lambda_i)) \end{aligned} \tag{19}$$

In this cost function, only those samples in the set $X^{(1)}$ are used, for which the class of samples are known. This supervised training will lead to an updated neural network which will generally perform at least as well, on this data set, as the previous one. Thus, the stability requirement is likely to be satisfied. However, since the $X^{(1)}$ set is only a subset of the whole feature set X and their distributions are not generally the same, the resultant neural network may not be able to reflect the distribution of the whole feature space, X, hence degrading the classification performance.

3) Cost Function 3: Combination of Supervised and Unsupervised Learning:

$$\begin{aligned} F_3(X;\Lambda) &= \sum_{\boldsymbol{x}\in X}\log(p(\boldsymbol{x};\Lambda)) \\ &= \sum_{\boldsymbol{x}\in X^{(1)}}\log(p(\boldsymbol{x};\Lambda)) + \sum_{\boldsymbol{x}\in X^{(2)}}\log(p(\boldsymbol{x};\Lambda)) \\ &= \sum_{i=1}^{K}\left[\sum_{\boldsymbol{x}\in X_i^{(1)}}\log(p(\boldsymbol{x}|c_i;\lambda_i))\right] \\ &\quad + \sum_{\boldsymbol{x}\in X^{(2)}}\log(p(\boldsymbol{x};\Lambda)) \end{aligned} \tag{20}$$

This cost function is a combination of the two aforementioned cost functions. All the samples and available class information are used. Maximizing the first part corresponds to the supervised learning process which can help to keep the stability of the training, while maximizing the second part leads to unsupervised learning which can help to form a more accurate representation of the distribution for the whole feature space, thus providing the plasticity needed for this problem. Fortunately, this is still a maximum likelihood estimation problem and the EM approach can be used to maximize this cost function as a whole. This approach is similar, in principle, to that in [28]. The detail derivation to maximize this cost function is given in Appendix A. It can be proven that the local maxima can be achieved by the iterative use of the following 2-steps until convergence is reached. The parameter set of the old PNN can be used as the initial values.

- E step

$$E\left[z_{j|i}(\boldsymbol{x})\middle|X;\Lambda^{\text{old}}\right] = \frac{p_{ji}\left(\boldsymbol{x};\mu_{ji}^{\text{old}},\Sigma_{ji}^{\text{old}}\right)\pi_{ji}^{\text{old}}}{\sum_{m=1}^{M_i} p_{mi}\left(\boldsymbol{x};\mu_{mi}^{\text{old}},\Sigma_{mi}^{\text{old}}\right)\pi_{mi}^{\text{old}}} \quad \text{where } \boldsymbol{x}\in X^{(1)},\ i=1,\cdots,K, \text{ and } j=1,\cdots,M_i \tag{21a}$$

$$E\left[z_{ji}(\boldsymbol{x})|X,\Lambda^{\text{old}}\right] = \frac{p_{ji}\left(\boldsymbol{x};\mu_{ji}^{\text{old}},\Sigma_{mi}^{\text{old}}\right)\pi_{ji}^{\text{old}}}{\sum_{k=1}^{K}\sum_{m=1}^{M_k} p_{mk}\left(\boldsymbol{x};\mu_{mk}^{\text{old}},\Sigma_{mk}^{\text{old}}\right)\pi_{mk}^{\text{old}}} \quad \text{where } \boldsymbol{x}\in X^{(2)} \tag{21b}$$

where $z_{j|i}(\boldsymbol{x})$ and $z_{ji}(\boldsymbol{x})$ are random variables indicating which Gaussian component generates the observation pattern. However, they are defined on a different set: $z_{j|i}$ is defined on the $X^{(1)}$ set where the class label of the input is known while z_{ji} is defined on the $X^{(2)}$ set where

neither class nor mixture component information is available. Please refer to Appendix A for further information.

- M step [see (21c)–(21e), shown at the bottom of the page.]

Several issues must be carefully considered in the implementation of this scheme. The main idea of this updating scheme is to re-estimate the parameter set of the PNN so that it can more accurately represent the changed space. This, generally requires the availability of substantial amount of data, which may not be possible in many real-life applications. This data poverty not only prevents the accurate estimation of the parameters for certain Gaussian component but also makes it difficult to update the proportional weights among different components. Considering these issues, we choose to update only those Gaussian components which have enough samples in $X^{(1)}$ set. Furthermore, only the mean vectors are updated, while the covariance matrices and weights are assumed to be unchanged over time since generally accurate estimation of these variables requires larger data set. Another important issue in updating is to balance the contributions between the unsupervised and supervised training. Let us rewrite (21d) as shown in (22a) where

$$\mu_{ji}^{\sup} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(x)\middle|X,\,\Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]}$$

and

$$\mu_{ji}^{\text{un sup}} = \frac{\sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]} \tag{22b}$$

It can be shown that if we only perform supervised learning on the $X^{(1)}$ set, then the updated mean vector will be $\mu_{ji}^{\sup}$. On the hand, $\mu_{ji}^{\text{un sup}}$ is the result of the unsupervised learning on the data set $X^{(2)}$. Moreover,

$$\beta = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]} \tag{22c}$$

controls the contributions of the two types of training. Notice that the terms $\sum_{\boldsymbol{x}\in X_i^{(1)}} E[z_{j|i}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]$ and $\sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]$ represent the expected number of samples in $X^{(1)}$ and $X^{(2)}$ sets that belonging to component j in class c_i. Thus, the combination factor β is totally decided by the newly arrived data set. Also, the result in (22a) implies that the solutions to (20) can be expressed in terms of the weighted combination of the supervised and unsupervised solutions while the weights, β and $(1-\beta)$ are determined by number and distribution of the samples in the sets of $X^{(1)}$ and $X^{(2)}$. For the supervised learning to dominate the final results, i.e., satisfy the stability requirement, we define a constant factor $\beta_{\min}$ and require that the combination factor $\beta > \beta_{\min}$.

Based on the above discussion, we improved the updating scheme by adding some necessary inspection steps. The new scheme is given as follows:

1) The parameter set updated to the last frame is used as the initial value. For the jth Gaussian component in class i, $i = 1, \cdots, K$ and $j = 1, \cdots, M_i$
2) Use (21a) and (b) to calculate $E[z_{j|i}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]$ (for $\boldsymbol{x} \in X^{(1)}$) and $E[z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]$ (for $\boldsymbol{x} \in X^{(2)}$).
3) Check to see if there is enough number of samples available for updating the mean of this Gaussian component.

   If $\sum_{\boldsymbol{x}\in X_i^{(1)}} E[z_{j|i}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}] < N_1$, where N_1 is called the "updating threshold," then no updating will be performed on this Gaussian component because of data poverty. Otherwise go to the next step.

$$\pi_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]}{\sum_{m=1}^{M_i}\left\{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{m|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[z_{mi}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]\right\}} \tag{21c}$$

$$\mu_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]\boldsymbol{x} + \sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]} \tag{21d}$$

$$\Sigma_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right](\boldsymbol{x}-\mu_{ji})(\boldsymbol{x}-\mu_{ji})^t + \sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right](\boldsymbol{x}-\mu_{ji})(\boldsymbol{x}-\mu_{ji})^t}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[Z_{j|i}(\boldsymbol{x})\middle|X,\,\Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[Z_{ji}(\boldsymbol{x})|X,\,\Lambda^{\text{old}}]} \tag{21e}$$

4) Calculate the supervised and unsupervised learning results, μ_{ji}^{sup} and $\mu_{ji}^{\text{un sup}}$, as well as combination factor β.
5) If $\beta < \beta_{\min}$, let $\beta = \beta_{\min}$. This step ensures that the supervised training will play more important role in the updating than that of the unsupervised learning.
6) Updating: If $\sum_{\boldsymbol{x}\in X_i^{(1)}} E[z_{j|i}(\boldsymbol{x})|X, \Lambda^{\text{old}}] > N_2$, ($N_2$ is a pre-defined factor called as "sufficient threshold"), which indicates that there are enough samples for accurate estimation, then we choose the newly estimated parameter to replace the old one, i.e.,

$$\mu_{ji} = \beta\mu_{ji}^{\text{sup}} + (1-\beta)\mu_{ji}^{\text{un sup}}$$

Otherwise, the newly estimated value may not be reliable hence we choose to combine it with the corresponding parameter of the last frame, i.e.

$$\mu_{ji} = \gamma\mu_{ji}^{ini} + (1-\gamma)\left[\beta\mu_{ji}^{\text{sup}} + (1-\beta)\mu_{ji}^{\text{un sup}}\right]$$

where μ_{ji}^{ini} is the initial value (the parameters of the last frame) and the factor γ satisfies $0 \le \gamma \le 1$ and can be calculated by

$$\gamma = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X, \Lambda^{\text{old}}\right] - N_1}{N_2 - N_1}$$

The above steps are performed iteratively until convergence is reached.

IV. Experimental Results

In this section, two experiments were designed to examine the effectiveness of the proposed temporal updating scheme. In the first case, a simulated data set was used. In spite of its simplicity, this data set can help to demonstrate the properties of the proposed scheme very well. The second test was done on two sequences of GOES-8 satellite images. A cloud classification system was implemented and its performance was examined in this case.

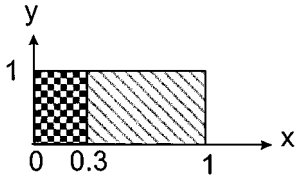

Fig. 4. Data set D for testing.

A. Experiment 1: Simulated Data Set

In this experiment, the database consists of only two classes. The data for these classes are uniformly distributed in the 2-D space [0, 0.3] × [0, 1] and [0.3, 1] × [0, 1], respectively, as shown in Fig. 4. These data sets have several unique features. First, the distribution is not Gaussian thus posing a big challenge for the proposed method which is based upon Gaussian mixture models. Second, the simple unsupervised-based training scheme is not likely to achieve good results since the distribution is uniform and the areas of the two classes are not the same.

The PNN described in Section II was adopted and only one Gaussian component (worst case scenario) was assigned to each class. A total of 400 samples for each class were used in the training phase. The classification error rate of the resultant PNN was 1.38%. This data set, D, was then exposed to four kinds of temporal changes T_i, $i = 1, \cdots, 4$, and four new sets D_i were generated by T_i: $D \to D_i$. Specifically, they are defined as follows:

1) Data set $D_1 = T_1(D)$, where T_1: $(x^{\text{old}}, y^{\text{old}}) \in D \to (x^{\text{old}} * 1.5 + w_1, y^{\text{old}} + w_2) \in D_1$
2) Data set $D_2 = T_2(D)$, where T_2: $(x^{\text{old}}, y^{\text{old}}) \in D \to (x^{\text{old}} * 0.8 + w_1, y^{\text{old}} + w_2) \in D_2$
3) Data set $D_3 = T_3(D)$, where T_3: $(x^{\text{old}}, y^{\text{old}}) \in D \to (x^{\text{old}} + 0.1 + w_1, y^{\text{old}} + w_2) \in D_3$
4) Data set $D_4 = T_4(D)$, where T_4: $(x^{\text{old}}, y^{\text{old}}) \in D \to (x^{\text{old}} - 0.1 + w_1, y^{\text{old}} + w_2) \in D_4$

where w_1 and w_2 are Gaussian white noise with zero mean and standard deviation 0.05. Data set D_1 was designed to simulate the expansion trend of the features while data set D_2 was generated to represent the shrinkage case. Moreover, D_3 and D_4 correspond to the case where the changes are translational. All these four kinds of changes are very common.

$$\begin{aligned}\mu_{ji} =& \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X, \Lambda^{\text{old}}\right]}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X, \Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}]} \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(x)\middle|X, \Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X, \Lambda^{\text{old}}\right]} \\ &+ \frac{\sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\right]}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\left[z_{j|i}(\boldsymbol{x})\middle|X, \Lambda^{\text{old}}\right] + \sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}]} \frac{\sum_{\boldsymbol{x}\in X^{(2)}} E\left[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\right]\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X^{(2)}} E[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}]} \\ =& \beta\mu_{ji}^{\text{sup}} + (1-\beta)\mu_{ji}^{\text{un sup}} \end{aligned} \tag{22a}$$

TABLE I
THE CLASSIFICATION ERROR RATES (%) OF THE THREE PNN

	Ideal PNN	Non-updated PNN	Updated PNN
Data set D1	3.0	12.4	3.75
Data set D2	5.25	7.75	6.63
Data set D3	4.13	11.25	5.50
Data set D4	4.13	8.38	6.25

The proposed temporal adaptation scheme was then applied to these four new data sets. Since there is no complicated spatial temporal context for this example, for a sample $\boldsymbol{z} = (x, y)$ in the current data set D_i, $i = 1, \cdots, 4$, its spatial–temporal neighborhood is defined to include only one sample in D, which is determined by the inverse mapping of T_i, i.e. $T_i^{-1}(\boldsymbol{z})$. After the updating process, the new PNN is used to re-classify the temporally changed data set and the resultant classification error rates are given in Table I.

For the sake of comparison, the accuracies for two other schemes are also provided. The first one corresponds to the results of the PNN trained on the new data set. The second one corresponds to the nonupdated case where the PNN that was trained on the old data set, D, was used. From Table I, it can be found that the performance of the nonupdated PNN degraded significantly when temporal changes are present. The classification error rate jumped from 3–5% for the ideal case to more than 7.7%. After temporal updating, the PNN performed much better. For the data set D_1 and D_3, the improvement on the accuracy rate for the updated PNN is around 6–8% while for the data set D_2 and D_4, this is around 1.4–2.1%. It is interesting to mention that the updating improvement varies for each data set. There are several factors that contribute to this phenomenon, such as the number of samples in the updating sets $X^{(1)}$ and $X^{(2)}$, and the parameter set of the old PNN, but perhaps the most important one is that the unsupervised learning plays a different role for each case. For data set D_1 and D_3, the actual decision boundary is shifted to the right compared to that of the original data set D. Thus most of the samples in the set $X^{(2)}$ belong to class 1. Owing to the initial value of PNN, the unsupervised training on $X^{(2)}$ will help to move the decision boundary to the right, which is the correct direction. For the data set D_2 and D_4, the situation is just the opposite, so the updating results are inferior to the other cases. Overall, the proposed updating scheme worked well on this simulated data set.

B. Experiment 2: GOES 8 Satellite Image Classification

In order to test the proposed temporal updated scheme for cloud classification application, a GOES 8 satellite imagery database was used for this purpose. The GOES 8 satellite carries five channel sensors. However, only two channels, namely visible (channel 1) and IR (channel 4), were used since most of the other meteorological satellites only carry these two channels. Two image series acquired on May 1st and 5th, 1995 between 15:45–20:45 UTC* at one hour interval, were chosen for this study. Figs. 5 and 6 show the image pair (visible and IR) obtained at the beginning, 15:45 UTC and at the end, 20:45 UTC, respectively, for the sequence collected on May 1st, 1995. These images of size 512 × 512 pixels (spatial resolution of 4 km/pixel) cover the mid-west and most of the eastern part of the U.S., extending from the Rocky Mountains to the Atlantic coast. The images cover mountains, plains, lakes and coastal areas where clouds have some specific features that are tied to topography. Lake Michigan is in the upper right corner and Florida is located in the lower right, with Gulf of Mexico in the lower center of the image. These sequences are of particular interest because of the presence of a variety of cloud types. For example, in Fig. 5 one can find thin cirrus in the left middle part, cirrostratus in the right middle part and low/middle-level clouds (stratocumulus and altostratus) in the center part as well as water and land areas. Since ground truth maps are not available and/or reliable, two meteorologists were asked to identify all the possible cloud types as well as the background areas based on the visual inspection and other related information. This was accomplished with the aid of a computer software package developed solely for this purpose. Approximately 50% of each image was analyzed and classified into ten cloud/background classes which are: Cold Land (Clnd), Land (Lnd), Water (Wtr), Stratus (St), Cumulus (Cu), Altostratus (As), Stratocumulus (Sc), Cirrus, and CirroStratus (Cs).

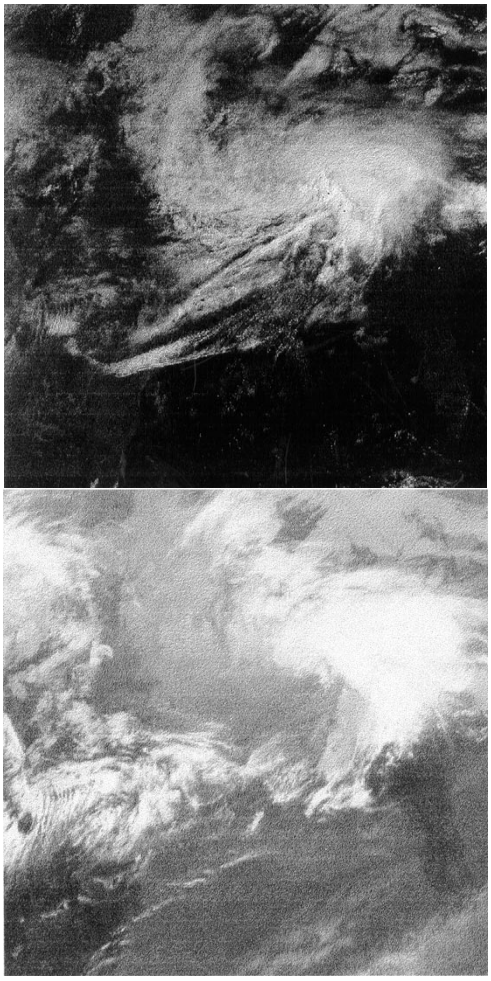

Fig. 5. GOES 8 satellite images obtained in May 1st, 1995 at time 15:45 UTC.

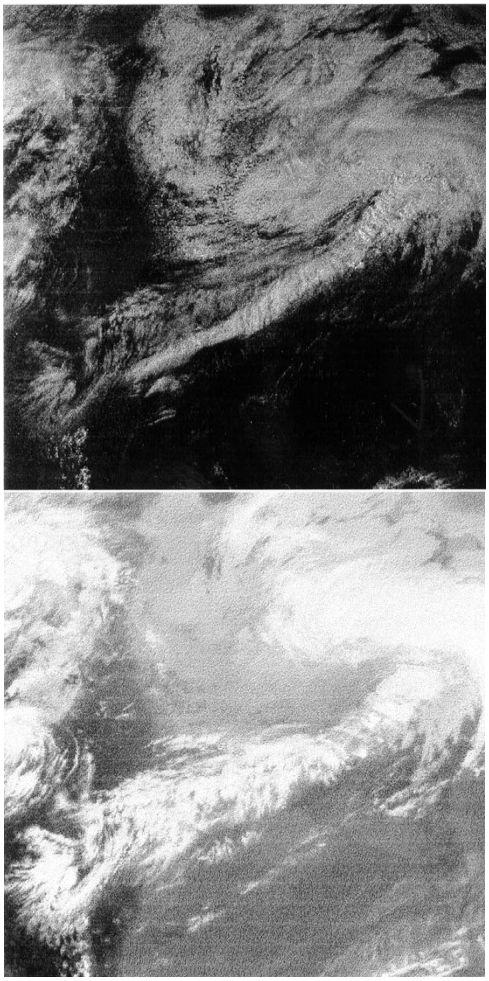

Fig. 6. GOES 8 satellite images obtained in May 1st, 1995 at 20:45 UTC.

The visible and IR images were first partitioned into small blocks of size 8×8 corresponding to an area of size 32×32 km. A feature extraction operation was then performed on these blocks to extract pertinent features for classification. Various textural feature extraction schemes for the cloud classification application were studied and compared in [29]. These studies indicated that the Singular Value Decomposition (SVD), 2-D Wavelet Transform and Gray Level Concurrence Matrix (GLCM) textural features achieved almost similar result. Nonetheless, since SVD scheme is much less computationally demanding, it was adopted in this study. Using the SVD approach, a total of 16 singular values were first extracted, 8 from every 8×8 block in each channel. To remove the redundancy among these features, a sequential forward feature selection process was employed in which the Bhattacharya distance is used to measure the class separability. After the feature selection process, only 6 features with good discriminatory ability were chosen for the subsequent classification process. These features correspond to the 1st, 3rd, 5th singular values in the visible channel and the 1st, 3rd and 6th in the IR channel. For a detail discussion on feature extraction/extraction, the reader is referred to [29].

The initial PNN was trained on six image pairs obtained from 15:45 UTC to 17:45 UTC on both May 1st and May 5th, 1995. Half of the labeled blocks in these image pairs made up the training set while the rest formed the test set. The Cirrus type has the largest number of training samples (2629) while the Cold Water type has the least number of samples (46). The detailed information on the training of this PNN including the number of training samples for each class and the performance evaluation is described in [29]. Fig. 7(a) and (b) show the classification results of the PNN on the image pairs collected at 15:45 UTC, May 1st (shown in Fig. 5) and the areas labeled by the meteorologists, respectively. Visual inspection of the color-coded image in Fig. 7(a) indicates that different cloud/background areas have been well-separated and the results agree very well with the meteorologist labeling in Fig. 7(b). However, when this PNN was applied to the images obtained at 20:45 UTC of the same day, the results were very poor. The color-coded image in Fig. 8(a) shows the result while the corresponding labeled image (meteorologists) is shown in Fig. 8(b). Comparing to the results in Fig. 7(a), many of the cloud areas have been incorrectly assigned to different classes although the clouds may not change a lot. The large area of Cumulus cloud over the Gulf of Mexico (lower center) has been misclassified to a mixture of Cirrus and Stratocumulus. The Stratocumulus clouds in the upper central part have been misclassified as Altostratus. In addition, several other regions have been misclassified into Cirrus type, for example, the Stratocumulus clouds in the upper right corner and the Altocumulus cloud in the central right part. Several factors may possibly contribute to this poor performance. The training sets are not large or representative enough for some classes. The major factor, however, has to do with the temporal changes in the data. Fig. 9(a) and (b) provide the scatter plots of the original images at time 15:45 UTC and 20:45 UTC, May 1st, respectively. The y-axis represents the normalized pixel intensity in the IR channel that reflects the temperature. Generally, land is the warmest class at day time in May leading to the largest pixel value, followed by water, low level clouds, middle level clouds, while the cirrus is the coolest class. On the other hand, the values on the x-axis describe the reflectivity in the visible channel. Water has the lowest reflectivity (at low sun angles—due to radiation being reflected away from the satellite sensor rather than scattered toward it) giving rise to the smallest value, followed by ground and then clouds that have a high scattering component. Careful comparison of the two scatter plots can easily reveal the differences. The most obvious distinction is in the upper left corner that corresponds to the land areas. Due to the heating effects, this region has expanded to cover a larger area after five hours. In the mean time, due to sun angle changes, clouds are not as bright in the visible channel as they were before. There are many pixels whose values exceed 0.5 in the visible channel at 15:45 UTC while very few of them exist at 20:45 UTC (the lower sun angle causes more energy to be reflected from the cloud, and away from the satellite).

The proposed temporal adaptation approach was then applied to this image series. The updating process starts from 18:45 UTC and happens every hour. The network structure as well as the weights and covariance matrices of the Gaussian components are kept unchanged while the mean vectors are updated when there are enough samples available [i.e., $\sum_{\boldsymbol{x} \in X_i^{(1)}} E[z_{j|i}(\boldsymbol{x})|X, \Lambda^{\text{old}}] > N_1$]. All the constants needed in the training algorithm are decided experimentally.

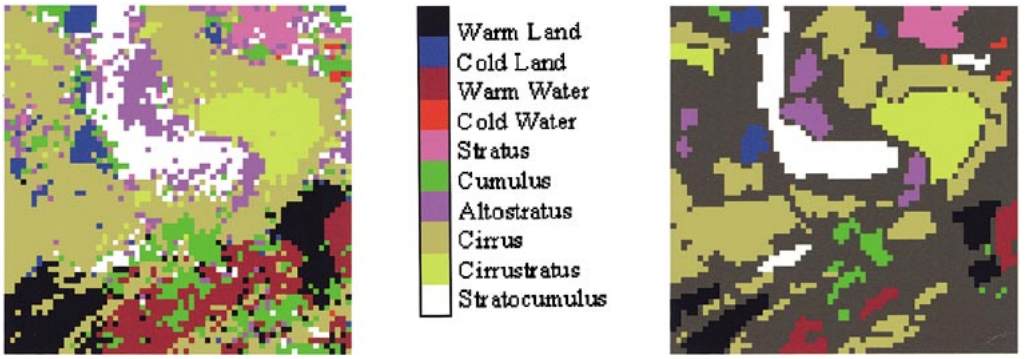

Fig. 7. (a) Classification results at time 15:45 UTC, May 1st, 1995. (b) Cloud/background classes labeled by meteorologists.

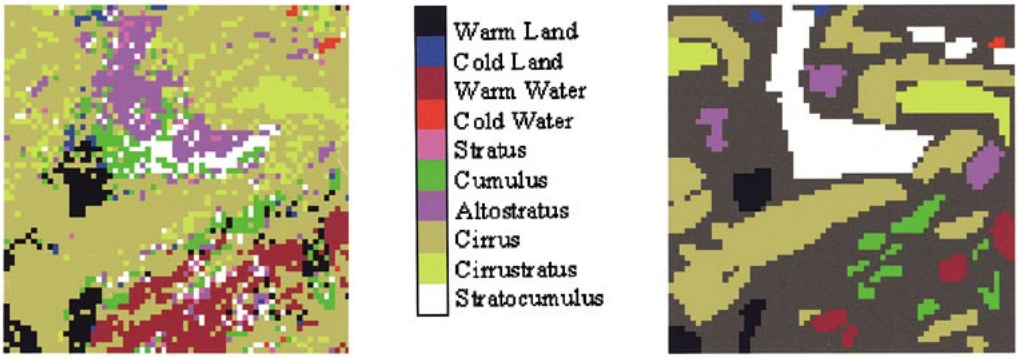

Fig. 8. (a) Classification results of the nonupdated PNN at time 20:45 UTC, May 1st, 1995. (b) Cloud/background classes labeled by meteorologists.

Specifically, the updating threshold N_1 was chosen to be 5 and the sufficient threshold N_2 was set at 10. The combination factor β was 0.9 for the background classes and 0.5 for the cloud types. This difference on β settings is due to the observation that the unsupervised training tends to degrade the separation between land and water classes, so large β was chosen to reduce the effect of unsupervised learning. The size of the spatial–temporal neighborhood for the predictor was 3×3 for this example.

Fig. 10 shows the classification result of the image pair obtained at time 20:45 UTC, May 1st using the updated PNN. Comparing with the nonupdating results in Fig. 8(a), significant improvements can be observed. For example, the classification errors made by nonupdated PNN on the Cumulus clouds over the Mexico Gulf are corrected and most of the clouds in the upper center area have been classified to the correct Stratocumulus type. Furthermore, the classification accuracy improvements are also observed in both the upper right corner and the Altostratus clouds in the center right part. On the other hand, for those regions that were not changed, the updated PNN made the same decisions as the nonupdated PNN that demonstrates the stability of the updating approach.

Besides the visual inspections, the classification accuracy rates for both the nonupdated and updated PNN are examined. There are totally nine types of clouds identified in the image pair at 20:45 UTC (Stratus type is missing). The number of labeled blocks for each class is given in Table II. It should be mentioned that the way meteorologists label images differs from that of the neural network classification system, i.e., instead of labeling each block individually, they first try to identify certain regions and then assign that whole area into one category. As a result, it is possible that some blocks in that region may belong to different classes since the labeling is done based upon global information. Also not all the blocks have been labeled and some regions have mixed cloud types hence making the classification task difficult. Due to all these factors, the accuracy rate may not fully represent the performance of the classifier and consequently the visual inspection of the results would be the best way to evaluate the performance.

The confusion tables for the PNN classifier before and after the updating are given in Tables III and IV, respectively. It was found that the overall classification rate increased from 65.8% to 75.4% after the updating process. Furthermore, the accuracy rate are significantly improved for the Warm Water (WWtr) and almost all the cloud types, namely Cumulus (Cu), Altostratus (As), CirroStratus (Cs) and Stratocumlus (Sc) classes, while the accuracy is not changed for Cold Land (CLnd) type. Warm Land (WLnd), Cold Water (CWtr) and Cirrus (Ci) are the three types that the accuracy degrades after the updating. For the Cold Water (CWtr) class, although 20% accuracy degradation seems a lot, it can easily be explained since it is just caused by one block misclassified as the Warm Water (WWtr) type (there are only a total of 5 labeled blocks available). Similarly, the slight accuracy degradation on the Warm Land (WLnd) class is mainly caused by two blocks that were wrongly classified, one as Cold Land (CLnd) type. The degradation of accuracy on the Cirrus (Ci) class, on the other hand, is primarily due to the fact that the result of the nonupdated PNN for this class is seriously biased because of the predominant number of blocks in this class. On the contrary, the accuracy improvements for some of the cloud types are significant. For example, for the Cumulus (Cu), Altostratus

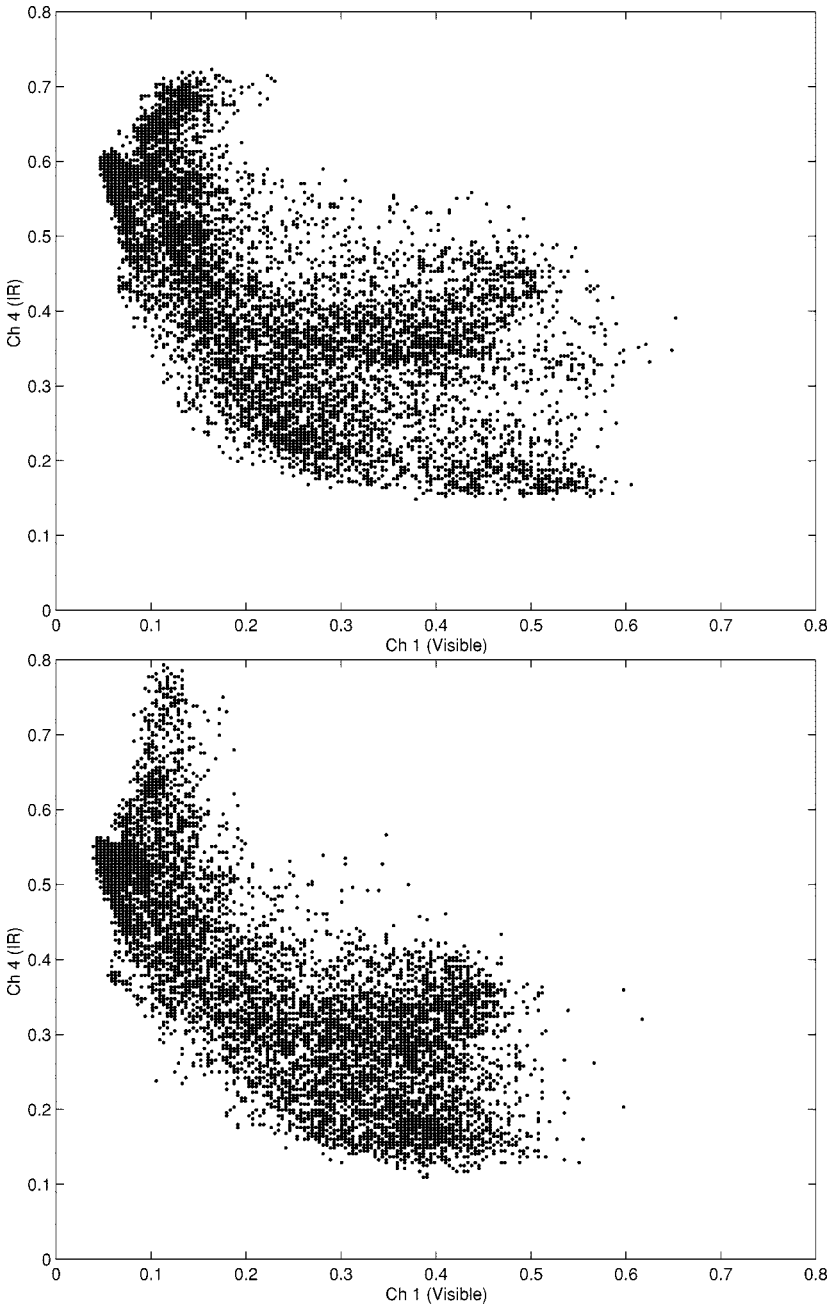

Fig. 9. Scatter plots of the cloud images got in May 1st, 1995.

(As) and Stratocumulse (Sc) class, the accuracy rates after the updating go up 42.2%, 20.7% and 45.7%, respectively. These results clearly demonstrate the effectiveness of the proposed updating scheme.

The same tests have also been applied to the image sequence collected on the May 5th. Very similar results were achieved. Overall the proposed updating-based PNN achieved better and more consistent classification results than those of the nonupdated ones. Not only the color-coded results conform very well with the meteorologist's labeling, but also the classification rates were improved. The price paid for this improvement is the extra computation cost due to the updating process. The proposed scheme was implemented in MATLAB running on a Pentium 266 MHz. The CPU time for processing the two image

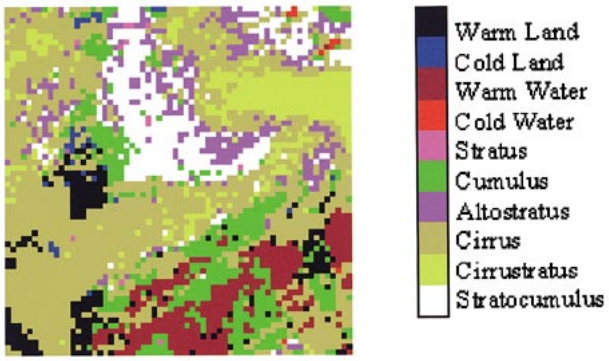

Fig. 10. Cloud classification result of the updated PNN at time 20:45 UTC, May 1st.

TABLE II
NUMBER OF LABELED BLOCKS FOR EACH CLASS IN THE IMAGE PAIR OF 20:45 UTC, MAY 1st, 1995.

Warm Land (WLnd)	Cold Land (CLnd)	Warm Water (WWtr)	Cold Water (CWtr)	Stratus (St)	Low Cumulus (Cu)	Strato-Cumulus (Sc)	Alto-Stratus (As)	Cirro-Stratus (Cs)	Cirrus (Ci)
101	10	84	5	0	102	399	97	158	901

TABLE III
CONFUSION MATRIX FOR THE NON-UPDATED PNN CLASSIFIER (%). OVERALL CLASSIFICATION RATE IS 65.8%. (THE RESULTS IN EACH ROW OF THE TABLE REPRESENT THE NEURAL NETWORK CLASSIFICATION ACCURACY FOR EACH CLASS DETERMINED BASED ON THE RESULTS OF EXPERT LABELING.)

	WLnd	CLnd	WWtr	CWtr	St	Cu	As	Ci	Cs	Sc
WLnd	86.1	0	0	0	0	0	0	13.9	0	0
CLnd	0	60	0	0	0	10	0	30	0	0
WWtr	0	0	81	0	0	0	0	19	0	0
CWtr	0	0	0	80	0	0	0	20	0	0
St	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Cu	3.9	1	2	0	6.9	48	8.8	12.7	0	16.7
As	0	0	0	0	1	2.1	24.7	53.6	17.5	1
Ci	0.2	0.2	0.3	0	0.2	0.1	0.9	87	10.7	0.3
Cs	0	0	0	0	0	0	0	36.7	63.3	0
Sc	0	0	0	0	4.8	12.3	43.9	13.5	0.5	25.1

TABLE IV
CONFUSION MATRIX FOR THE UPDATED PNN CLASSIFIER USING SVD FEATURES (%). OVERALL CLASSIFICATION RATE IS 75.4%. (THE RESULTS IN EACH ROW OF THE TABLE REPRESENT THE NEURAL NETWORK CLASSIFICATION ACCURACY FOR EACH CLASS DETERMINED BASED ON THE RESULTS OF EXPERT LABELING.)

	WLnd	CLnd	WWtr	CWtr	St	Cu	As	Ci	Cs	Sc
WLnd	84.2	1	0	0	0	2	0	12.9	0	0
CLnd	0	60	0	0	0	30	0	10	0	0
WWtr	2.4	0	95.2	0	0	1.2	0	1.2	0	0
CWtr	0	0	20	60	0	20	0	0	0	0
St	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Cu	3.9	0	2	0	0	90.2	0	3.9	0	1
As	0	0	0	0	0	2.1	45.4	20.6	7.2	24.7
Ci	0.3	0.1	0.4	0	0.2	1.4	1.9	76	18.6	0.9
Cs	0	0	0	0	0	0	0	23.4	76.6	0
Sc	0	0	0	0	2	4	20.1	2.3	0.3	71.4

sequences is 943 and 827 s, respectively. Since the neural network has to be updated three times for each image sequence (18:45 to 20:45 UTC), the computational cost is approximately 5 min per frame. We expect that the processing time can further be reduced when the proposed approach is optimized and implemented using a more efficient computer language and on faster processor.

V. CONCLUSIONS

Many meteorological applications would benefit from an automated cloud classification system. However, the study of cloud classification is still in its infancy. There are several factors that contribute to the difficulties in cloud classification. These include: high variability of cloud features and lack of reliable ground truth. Furthermore, when dealing with sequences of consecutive satellite images, the temporal changes of the features in those images must be considered.

In this paper, a PNN-based cloud classification system with a novel temporal updating capability has been proposed. When a new image arrives, the system first performs initial classification as well as a prediction based upon the temporal class dependency context. This information is then used to update the PNN classifier. ML criterion is adopted in the updating process while the EM algorithm is used to estimate the new parameter set of the PNN. The input image is classified again by the updated PNN. The proposed scheme is examined on both a simulated data set and two sequences of GOES-8 satellite images, and promising results are achieved. Future research would involve a more thorough analysis of this scheme. In particular, among those issues to be studied are: how to further overcome possible data poverty problem, how to present the error propagation. Additionally, other on-line updating approaches may be explored.

APPENDIX A
DERIVATION OF THE UPDATING PROCESS USING EM APPROACH

For the updating process, our goal is to estimate the parameter set that will maximize the cost function in (20), i.e, $\Lambda^{\text{opt}} = \arg\max_{\Lambda} F_3(X;\,\Lambda)$ where

$$F_3(X;\,\Lambda) = \sum_{i=1}^{K}\left[\sum_{\boldsymbol{x}\in\boldsymbol{X}_i^{(1)}} \log(p(\boldsymbol{x}|c_i;\,\lambda_i))\right] + \sum_{\boldsymbol{x}\in\boldsymbol{X}^{(2)}} \log(p(\boldsymbol{x};\,\Lambda)) \tag{A.1}$$

Direct analytical solution for (A.1) is difficult to obtain. Instead, we will apply the EM approach. The basic idea of the EM algorithm is that the maximization process may be simplified if

the missing variable is added to the data set. In this application, the missing information is identification of the Gaussian component that generates the observation pattern. Notice that for the samples in set $X^{(1)}$ and $X^{(2)}$, the missing information is not the same. For a sample $\boldsymbol{x}$ in $X^{(1)}$ set belonging to class i, we can define a variable $z_{j|i}(\boldsymbol{x})$ by

$$z_{j|i}(\boldsymbol{x}) = \begin{cases} 1, & \text{if feature } \boldsymbol{x} \text{ comes from } j\text{th Gaussian component in class } i \\ 0, & \text{otherwise,} \quad i = 1, \cdots, K \text{ and } j = 1, \cdots, M_i \end{cases} \tag{A.2}$$

On the other hand, for the samples in set $X^{(2)}$, for which the class information is missing, we can similarly define variable $z_{ji}(\boldsymbol{x})$ by

$$z_{ji}(\boldsymbol{x}) = \begin{cases} 1, & \text{if feature } \boldsymbol{x} \text{ comes from class } i \text{ and the } j\text{th Gaussian component generates it.} \\ 0, & \text{otherwise,} \quad i = 1, \cdots, K \text{ and } j = 1, \cdots, M_i \end{cases} \tag{A.3}$$

All $z_{j|i}$ and z_{ji} form the set Z. The observation feature set X is generally called the incomplete data set while the set $Y = \{X, Z\}$ is called the complete data set, since the missing information has been added to it. We can extend the likelihood function to include the unobserved variable and compute the log–likelihood of the complete data set as

$$\begin{aligned} F(Y|\Lambda) &= \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \log p(\boldsymbol{y}|c_i;\, \lambda_i) + \sum_{\boldsymbol{x} \in X^{(2)}} \log p(\boldsymbol{y};\, \Lambda) \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \log \left(\sum_{j=1}^{M_i} z_{j|i}(\boldsymbol{x}) p(\boldsymbol{x}|c_i,\, z_{j|i}(\boldsymbol{x}) = 1;\, \lambda_i) \cdot p(z_{j|i}(\boldsymbol{x}) = 1) \right) \\ &\quad + \sum_{\boldsymbol{x} \in X^{(2)}} \log \left(\sum_{i=1}^{K} \sum_{j=1}^{M_i} z_{ji}(\boldsymbol{x}) p(\boldsymbol{x}|c_i z_{ji}(\boldsymbol{x}) = 1;\, \lambda_i) \cdot p(z_{ji}(\boldsymbol{x}) = 1) \right) \end{aligned} \tag{A.4}$$

Using the Gaussian mixture model in (4), we have

$$\begin{aligned} &F(Y|\Lambda) \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \log \left(\sum_{j=1}^{M_i} z_{j|i}(\boldsymbol{x}) p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji}) \pi_{ji} \right) \\ &\quad + \sum_{\boldsymbol{x} \in X^{(2)}} \log \left(\sum_{i=1}^{K} \sum_{j=1}^{M_i} z_{ji}(\boldsymbol{x}) p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji}) \pi_{ji} P(c_i) \right) \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \sum_{j} z_{j|i}(\boldsymbol{x}) \log(\pi_{ji} p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji})) \\ &\quad + \sum_{\boldsymbol{x} \in X^{(2)}} \sum_{i=1}^{K} \sum_{j} z_{ji}(\boldsymbol{x}) \log(\pi_{ji} p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji}) P(c_i)) \end{aligned} \tag{A.5}$$

Since both $z_{j|i}$ and z_{ji} are delta type functions, the summations and log operation can be interchanged in the last step.

One important property of the EM scheme is that the likelihood function of the incomplete data set X in (A.1) can be maximized by iterative application of the E and M-steps to the complete data likelihood function, $F(Y|\Lambda)$ [25].

In the E-Step, we take the expectation of the complete data likelihood based on current parameter set, Λ^{old} and observation set X. This expectation is denoted by $Q(\Lambda|\Lambda^{\text{old}})$, i.e.,

$$\begin{aligned} &Q\big(\Lambda\big|\Lambda^{\text{old}}\big) \\ &:= E\big(F(Y|\Lambda)|X,\, \Lambda^{\text{old}}\big) \\ &= E\left[\sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \sum_{j} z_{j|i}(\boldsymbol{x}) \log(\pi_{ji} p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji})) \right. \\ &\qquad \left. + \sum_{\boldsymbol{x} \in X^{(2)}} \sum_{i=1}^{K} \sum_{j} z_{ji}(\boldsymbol{x}) \log(\pi_{ji} p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji})) \,\middle|\, X, \Lambda^{\text{old}} \right] \\ &= \sum_{i=1}^{K} \sum_{\boldsymbol{x} \in X_i^{(1)}} \sum_{j} E\big[z_{j|i}(\boldsymbol{x})\big|X,\, \Lambda^{\text{old}}\big] \log(\pi_{ji} p_{ji}(\boldsymbol{x};\, \mu_{ji},\, \Sigma_{ji})) \\ &\qquad + \sum_{\boldsymbol{x} \in X^{(2)}} \sum_{i=1}^{K} \sum_{j} E\big[z_{ji}(x)|X,\, \Lambda^{\text{old}}\big] \\ &\qquad \cdot \log(\pi_{ji} p_{ji}(x;\, \mu_{ji},\, \Sigma_{ji})) \end{aligned} \tag{A.6}$$

According to the definition of $z_{j|i}$ in (A.2), for the sample x belonging to $X^{(1)}$ set, we can calculate

$$\begin{aligned} &E\big[z_{j|i}(\boldsymbol{x})\big|X;\, \Lambda^{\text{old}}\big] \\ &\quad = P\big(z_{j|i}(\boldsymbol{x}) = 1\big|X;\, \Lambda^{\text{old}}\big) \\ &\quad = \frac{p\big(\boldsymbol{x}\,\big|z_{j|i}(\boldsymbol{x}) = 1;\, \lambda_i^{\text{old}}\big)\, p\big(z_{j|i}(\boldsymbol{x}) = 1;\, \lambda_i^{\text{old}}\big)}{p\big(\boldsymbol{x};\, \lambda_i^{\text{old}}\big)} \\ &\quad = \frac{p_{ji}\big(\boldsymbol{x};\, \mu_{ji}^{\text{old}},\, \Sigma_{ji}^{\text{old}}\big)\, \pi_{ji}^{\text{old}}}{\sum_{m=1}^{M_i} p_{mi}\big(\boldsymbol{x};\, \mu_{mi}^{\text{old}},\, \Sigma_{mi}^{\text{old}}\big)\, \pi_{mi}^{\text{old}}} \end{aligned} \tag{A.7}$$

Similarly, for the samples x from set $X^{(2)}$, we can get

$$\begin{aligned} &E\big[z_{ji}(\boldsymbol{x})|X;\, \Lambda^{\text{old}}\big] \\ &\quad = P\big(z_{ji}(\boldsymbol{x}) = 1|X;\, \Lambda^{\text{old}}\big) \\ &\quad = \frac{p_{ji}\big(\boldsymbol{x};\, \mu_{ji}^{\text{old}},\, \Sigma_{ji}^{\text{old}}\big)\, \pi_{ji}^{\text{old}} p(c_i)}{\sum_{k=1}^{K} \sum_{m=1}^{M_k} p_{mk}\big(\boldsymbol{x};\, \mu_{mk}^{\text{old}},\, \Sigma_{mk}^{\text{old}}\big)\, \pi_{mk}^{\text{old}} p(c_k)} \end{aligned} \tag{A.8}$$

If we assume that the *a priori* class distribution, $p(c_i)$, are uniformly distributed, then

$$\begin{aligned} &E\big[z_{ji}(\boldsymbol{x})|X;\, \Lambda^{\text{old}}\big] \\ &\quad = P\big(z_{ji}(\boldsymbol{x}) = 1|X;\, \Lambda^{\text{old}}\big) \\ &\quad = \frac{p_{ji}\big(\boldsymbol{x};\, \mu_{ji}^{\text{old}},\, \Sigma_{ji}^{\text{old}}\big)\, \pi_{ji}^{\text{old}}}{\sum_{k=1}^{K} \sum_{m=1}^{M_k} p_{mk}\big(\boldsymbol{x};\, \mu_{mk}^{\text{old}},\, \Sigma_{mk}^{\text{old}}\big)\, \pi_{mk}^{\text{old}}} \end{aligned} \tag{A.9}$$

The second step in the EM algorithm is to maximize the expectation function $Q(\Lambda|\Lambda^{\text{old}})$with respect to the parameter set Λ. This M-step can be done by taking the derivative of $Q(\Lambda|\Lambda^{\text{old}})$and setting it to zero. The optimal parameters will be obtained by solving the resultant equations. This process is rather straightforward, since the parameters $\{\pi_{ji}, \mu_{ji}, \Sigma_{ji}\}$ are decoupled from each other after taking the derivative. The Langrange multiplier is used to solve for parameter π_{ji} in order to satisfy the constraint $\sum_{j=1}^{M_i} \pi_{ji} = 1$. The detailed mathematics is omitted here and the final result is given in (A.10)–(A.12) where $i = 1, \cdots, K$ and $j = 1, \cdots, M_i$.

Appendix B
Proof of (17) for Temporal Predictor

The MAP temporal predictor is given in (9), i.e.,

$$\hat{c}(\boldsymbol{r}; n) = \arg\max_{c_i} P(c(\boldsymbol{r}; n) = c_i|C_H(\boldsymbol{r}; n)) \quad i = 1, \cdots, K \tag{B.1}$$

where the *a posteriori* conditional probability can be computed using (13), which is rewritten here for reader's convenience.

$$P(c(\boldsymbol{r}; n) = c_i|C_H(\boldsymbol{r}; n)) = \sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r}; n) = c_i|c(\boldsymbol{r}+\boldsymbol{v}; n-1))P(\boldsymbol{\xi} = \boldsymbol{v}) \tag{B.2}$$

Furthermore, we assume that the class transition probability is in the form of

$$P(c(\boldsymbol{r}; n) = c_i|c(\boldsymbol{r}+\boldsymbol{v}; n-1) = c_j) = \begin{cases} \alpha, & \text{if } c_i = c_j \\ \dfrac{1-\alpha}{K-1}, & \text{otherwise} \end{cases} \tag{B.3}$$

where the transition probability between the same class, α, is generally much larger than the transition probability to the other class, $(1-\alpha)/(K-1)$. Now we want to prove that (16) is an equivalent predictor to that in (B.1). Substituting (B.3) into (B.2), we get

$$\begin{aligned} &P(c(\boldsymbol{r}; n) = c_i|C_H(\boldsymbol{r}; n)) \\ &\quad = \sum_{\boldsymbol{v}\in\Psi} P(c(\boldsymbol{r}; n) = c_i|c(\boldsymbol{r}+\boldsymbol{v}; n-1))P(\boldsymbol{\xi} = \boldsymbol{v}) \\ &\quad = \sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi} = \boldsymbol{v})\Big[\alpha\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i) \\ &\qquad + \frac{1-\alpha}{K-1}(1-\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i))\Big] \end{aligned} \tag{B.4}$$

where

$$\delta(c_i, c_k) = \begin{cases} 1, & c_i = c_k \\ 0, & \text{otherwise} \end{cases}$$

is similar to the Kronecker delta function. For any two classes c_i, and c_j, the difference between conditional probability is thus given by

$$\begin{aligned} &P[c(\boldsymbol{r}; n) = c_i|C_H(\boldsymbol{r}; n)] - P\lfloor c(\boldsymbol{r}; n) = c_j|C_H(\boldsymbol{r}; n)\rfloor \\ &\quad = \sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi} = \boldsymbol{v})\Big[\alpha\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i) \\ &\qquad + \frac{1-\alpha}{K-1}(1-\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i))\Big] \\ &\qquad - \sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi} = \boldsymbol{v})\Big[\alpha\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_j) \\ &\qquad + \frac{1-\alpha}{K-1}(1-\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_j))\Big] \\ &\quad = \sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi} = \boldsymbol{v})\Big[\alpha(\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i) \\ &\qquad - \delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_j)) \\ &\qquad - \frac{1-\alpha}{K-1}(\delta(c(\boldsymbol{r}+\boldsymbol{v}; n-1), c_i) \end{aligned}$$

$$\pi_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[z_{j|i}(\boldsymbol{x})\big|X, \Lambda^{\text{old}}\big] + \sum_{\boldsymbol{x}\in X^{(2)}} E\big[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\big]}{\sum_{m=1}^{M_i}\left\{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[z_{m|i}(\boldsymbol{x})\big|X, \Lambda^{\text{old}}\big] + \sum_{\boldsymbol{x}\in X^{(2)}} E\big[z_{mi}(\boldsymbol{x})|X, \Lambda^{\text{old}}\big]\right\}} \tag{A.10}$$

$$\mu_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[z_{j|i}(x)\big|X, \Lambda^{\text{old}}\big]\,\boldsymbol{x} + \sum_{\boldsymbol{x}\in X^{(2)}} E\big[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\big]\,\boldsymbol{x}}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[z_{j|i}(\boldsymbol{x})\big|X, \Lambda^{\text{old}}\big] + \sum_{\boldsymbol{x}\in X^{(2)}} E\big[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\big]} \tag{A.11}$$

$$\Sigma_{ji} = \frac{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[z_{j|i}(\boldsymbol{x})\big|X, \Lambda^{\text{old}}\big]\,(\boldsymbol{x}-\mu_{ji})(\boldsymbol{x}-\mu_{ji})^t + \sum_{\boldsymbol{x}\in X^{(2)}} E\big[z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}\big]\,(\boldsymbol{x}-\mu_{ji})(\boldsymbol{x}-\mu_{ji})^t}{\sum_{\boldsymbol{x}\in X_i^{(1)}} E\big[Z_{j|i}(\boldsymbol{x})\big|X, \Lambda^{\text{old}}\big] + \sum_{\boldsymbol{x}\in X^{(2)}} E[Z_{ji}(\boldsymbol{x})|X, \Lambda^{\text{old}}]} \tag{A.12}$$

$$
\begin{aligned}
&- \delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_j))\Big] \\
&= \left(\alpha - \frac{1-\alpha}{K-1}\right) \sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi}=\boldsymbol{v}) \\
&\quad \cdot [\delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_i) - \delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_j)]
\end{aligned} \tag{B.5}
$$

Since $\alpha > (1-\alpha)/(K-1)$, the sign of $P[c(\boldsymbol{r};\, n) = c_i | C_H(\boldsymbol{r};\, n)] - P[c(\boldsymbol{r};\, n) = c_j | C_H(\boldsymbol{r};\, n)]$ is solely decided by

$$
\sum_{\boldsymbol{v}\in\Psi} P(\boldsymbol{\xi}=\boldsymbol{v})\{\delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_i) - \delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_j)\}
$$

and not related to the value of α. So instead of using the *a posteriori* conditional probability in the temporal predictor of (B.1), it is equivalent to make prediction using

$$
\hat{c}(\boldsymbol{r};\, n) = \arg\max_{c_i} \sum_{\boldsymbol{v}\in\Psi} \delta(c(\boldsymbol{r}+\boldsymbol{v};\, n-1),\, c_i) P(\boldsymbol{\xi}=\boldsymbol{v}) \tag{B.6}
$$

REFERENCES

[1] G. S. Pankiewicz, "Pattern recognition techniques for identification of cloud and cloud systems," *Meteorol. Appl.*, vol. 2, pp. 257–271, Sept. 1995.

[2] W. E. Shenk and R. J. Holub, "A multi-spectral cloud type identification method developed for nimbus 3 mrir measurements," *Mon. Weather Rev.*, vol. 104, pp. 284–291, Mar. 1976.

[3] D. W. Reynolds and T. H. Vonder-Haar, "Bi-spectral method for cloud parameter determination," *Mon. Weather Rev.*, vol. 105, pp. 446–457, Mar. 1977.

[4] R. M. Welch, K. S. Kuo, S. K. Sengupta, and D. W. Chen, "Cloud field classification based upon high spatial resolution textural feature (I): Gray-level cooccurrence matrix approach," *J. Geophys. Res.*, vol. 93, pp. 12,663–12 681, Oct. 1988.

[5] N. Lamei, "Cloud-type discrimination via multispectral textural analysis," *Opt. Eng.*, vol. 33, pp. 1303–1313, Apr. 1994.

[6] J. A. Parikh, "A comparative study of cloud classification techniques," *Remote Sens. Environ.*, vol. 6, pp. 67–81, Mar. 1977.

[7] Z. Gu and C. Duncan, *et al.*, "Texture and spectral features as an aid to cloud classification," *Int. J. Remote Sensing*, vol. 12, no. 5, pp. 953–968, 1991.

[8] P. P. Ohanian and R. C. Dubes, "Performance evaluation of four classes of texture features," *Pattern Recognit.*, vol. 25, pp. 819–833, 1992.

[9] M. F. Augusteijin, "Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier," *IEEE Trans. Geosci. Remote Sensing*, vol. 33, pp. 616–625, May 1995.

[10] J. J. Simpson and J. I. Gobat, "Improved cloud detection in GOES scenes over land," *Remote Sens. Environ.*, vol. 52, pp. 36–54, 1995.

[11] ——, "Improved cloud detection in GOES scenes over the oceans," *Remote Sens. Environ.*, vol. 52, pp. 79–94, 1995.

[12] J. Lee *et al.*, "A neural network approach to cloud classification," *IEEE Trans. Geosci. Remote Sensing*, vol. 28, no. 5, pp. 846–855, Sept. 1990.

[13] R. L. Bankert *et al.*, "Cloud classification of AVHRR imagery in maritime regions using a probabilistic neural network," *J. Appl. Meteorol.*, vol. 33, pp. 909–918, Aug. 1994.

[14] R. M. Welch *et al.*, "Polar cloud and surface classification using AVHRR imagery: An intercomparison of methods," *J. Appl. Meteorol.*, vol. 31, pp. 405–420, May 1992.

[15] M. A. Shaikh and B. Tian, *et al.*, "Neural network-based cloud detection/classification using textural and spectral features," in *Proc. IGRASS'96*, Lincoln, NE, May 1996, pp. 1105–1107.

[16] B. Jeon and D. A. Landgrebe, "Classification with spatio-temporal interpixel class dependency contexts," *IEEE Trans. Geosci. Remote Sensing*, vol. 30, pp. 663–672, July 1992.

[17] P. H. Swain, "Bayesian classification in a time-varying environment," *IEEE Trans. Syst. Man., Cybern.*, vol. SMC-8, pp. 879–883, Dec. 1978.

[18] H. M. Kalayeh and D. A. Landgrebe, "Utilizing multi-temporal data by a stochastic model," *IEEE Trans. Geosci. Remote Sensing*, vol. GRS-24, pp. 792–795, Sept. 1986.

[19] D. F. Specht and P. D. Shapiro, "Generalization accuracy of probabilistic neural networks compared with backpropogation networks," in *Proc. IJCNN'91*, July 1991, pp. 458–461.

[20] D. F. Specht and H. Romsdahl, "Experience with adaptive probabilistic neural network and adaptive general regression neural network," in *ICNN'94*, 1994, pp. 1203–1208.

[21] D. F. Specht, "Probabilistic neural network," *Neural Networks*, vol. 3, pp. 109–118, 1990.

[22] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[23] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, pp. 1065–1076, 1962.

[24] P. Burrascano, "Learning vector quantization for the probabilistic neural network," *IEEE Trans. Neural Networks*, vol. 2, pp. 458–461, July 1991.

[25] R. L. Streit and T. E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," *IEEE Trans. on Neural Networks*, vol. 5, no. 5, pp. 764–783, 1994.

[26] S. Lin, S. Y. Kung, and L. Lin, "Face recognition/detection by probabilistic decision-based neural network," *IEEE Trans. Neural Networks*, vol. 8, pp. 114–132, 1997.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc.*, ser. B, vol. 39, pp. 1–38, 1977.

[28] R. A. Redner and H. F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," *SIAM Review*, vol. 26, pp. 195–240, 1984.

[29] B. Tian, M. A. Shaikh, M. R. Azimi, T. H. Vonder Haar, and D. Reinke, "An study of neural network-based cloud classification using textural and spectral features," *IEEE Trans. Neural Networks*, vol. 10, pp. 138–151, 1999.

Bin Tian received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 1991 and 1993, respectively. He has recently completed the Ph.D. degree at the Department of Electrical Engineering, Colorado State University, Fort Collins.

His main research interests include signal/image processing, telecommunication, and theory and applications of neural networ

Mahmood R. Azimi-Sadjadi (S'81–M'81–SM'89) received the B.S. degree from University of Tehran, Iran, in 1977, the M.Sc. and Ph.D. degrees from the Imperial College, University of London, U.K., in 1978 and 1982, respectively, all in electrical engineering.

He served as an Assistant Professor in the Department of Electrical and Computer Engineering, University of Michigan-Dearborn. Since July 1986, he has been with the Department of Electrical Engineering, Colorado State University, Fort Collins, where he is now a Professor. He is also the director of the Multisensory Computing Laboratory (MUSCL) at Colorado State University. His areas of interest include digital signal/image processing, target detection and tracking, multidimensional system theory and analysis, adaptive filtering, system identification, and neural networks. His contributions in these areas resulted in more than 100 journal and refereed conference publications. He is a coauthor of the book Digital Filtering in One and Two Dimensions (New York: Plenum, 1989).

Dr. Azimi-Sadjadi is the recipient of 1993 ASEE-Navy Senior Faculty Fellowship Award, 1991 CSU Dean's Council Award, 1990 Battelle Summer Faculty Fellowship Award, and the 1984 DOW chemical Outstanding Young Faculty Award of the American Society for Engineering Education. He is an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING.

Thomas H. Vonder Haar received the B.S. degree in aeronautics from Parks College of St. Louis University, MO, in 1963, the M.S. and Ph.D. degrees in meteorology from the University of Wisconsin, Madison, in 1964 and 1968, respectively.

He has taught in the Atmospheric Science Department at Colorado State University, Fort Collins, since 1970, and became a University Distinguished Professor in 1994. He became the Director of the Cooperative Institute for Research in the Atmosphere (CIRA) at Colorado State University in 1980. He has been the senior author or coauthor on more than 400 scientific publications, more than 100 in refereed journals. The range of titles covers several different research areas, with the use of measurements from meteorological satellites as a most common factor. Research work has been published in such journals as *Science*, *Solar Energy*, *Space Research*, *Journal of Atmospheric Science*, *Bulletin of the American Meteorological Society*, *Monthly Weather Review*, *Journal of Applied Meteorology*, *Journal of Geophysical Research*, *Journal of Atmospheric and Oceanic Technology*, etc. He is coauthor of *Satellite Meteorology: An Introduction* (New York: Academic, 1995). In addition, some original research work is now being included in textbooks and summaries of atmospheric scinece and atmospheric physics by authors both in the United States and abroad.

Donald Reinke received the B.S. degree in meteorology from the University of Oklahoma, Norman, in 1975 and the M.S. degree in atmospheric science from Colorado State University, Fort Collins, in 1982.

He has extensive experience in automated processing of meteorological satellite data, satellite derived cloud analyzes, scientific programming, and real-time meteorological support systems management. He is presently a Research Associate at Colorado State University managing a research group that provides data, data analysis computer applications, and technical support to CIRA, as well as management of the computer support resources for the 50-plus person research facility. His group also has the responsibility for 24-h/day operation of real-time satellite earthstation to collect environmental data from eight worldwide meteorological satellite platforms and two data service networks.His professional experience includes duty as a Research Scientist and Project Manager for a number of multiyear research programs related to the processing and analysis of digital meteorological satellite imagery and related ancillary data sets. He has been a lead or coauthor on more than 60 published papers.