IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Multi-view Temporal Ensemble for Classification of Non-Stationary Signals

B. H. D. Koh[1] and W. L. Woo[1, 2]

[1]Newcastle University, Newcastle upon Tyne , England, United Kingdom NE1 7RU
[2]Northumbria University, Newcastle upon Tyne, England, United Kingdom NE1 8ST

Corresponding author: B. H. D. Koh (e-mail: b.h.d.koh@newcastle.ac.uk).

ABSTRACT In the classification of non-stationary time series data such as sounds, it is often tedious and expensive to get a training set that is representative of the target concept. To alleviate this problem, the proposed method treats the outputs of a number of deep learning sub-models as the views of the same target concept that can be linearly combined according to their complementarity. It is proposed that the view's complementarity be the contribution of the view to the global view, chosen in this work to be the Laplacian eigenmap of the combined data. Complementarity is computed by alternate optimization, a process that involves the cost function of the Laplacian eigenmap and the weights of the linear combination. By blending the views in this way, a more complete view of the underlying phenomenon can be made available to the final classifier. Better generalization is obtained, as the consensus between the views reduces the variance while the increase in the discriminatory information reduces the bias. Data experiment with artificial views of environment sounds formed by deep learning structures of different configurations shows that the proposed method can improve the classification performance.

INDEX TERMS deep learning, data fusion, time series classification

I. INTRODUCTION

In multi-view learning, more than one feature set is used for learning. The features may be redundant, but they are not entirely similar. As such, besides learning the patterns in the features, the relationship among the feature sets can be used for learning too.

Multi-view learning was first introduced as a framework by Blum and Mitchell [1] for the semi-supervised learning of web page classification. The text of the web pages and the anchor text in the hyperlinks of the web pages were used as the feature sets in this two-view setting. Using co-training, two separate models built on the two disjoint views were used to predict the unlabeled data. This was used to decide on which of the unlabeled data to add to the training set. In this way, the training set can be enlarged for further training.

A survey on multi-view learning by Sun [2] reviews the theories, properties and behaviors of multi-view learning. It shows that multi-view learning, as an emerging and rapidly growing field in machine learning, has been used in all branches of machine learning, from unsupervised learning [3], semi-supervised learning, active learning [4], supervised learning, transfer learning [5], and ensemble training [6]. Some examples of applications include the sentiment analysis of the attitude or opinion of a user [7], and speech analysis for phonetic recognition [8].

An important part of multi-view learning is the construction of the views. The views may be naturally distinct, as in the text of the web page and the anchor text in the hyperlink of the web page, or the video and audio signals of a multimedia content [9]. They may be distinct due to the feature extraction methods used on the raw data, such as the CELP features and the MFCC features of an audio signal [10]. They may be subsets that are split from a single feature set, based on the ordered importance of the features in the feature set.

When multi-view features are not available, random feature split of a single view can be used to construct artificial views. The use of the artificial views in multi-view learning can still improve the generalization performance. This is because multi-view learning is robust to the violated assumptions of its underlying classifiers [11].

The architectures for the two types of multi-view data, namely natural and artificial, are shown in Fig. 1 below:

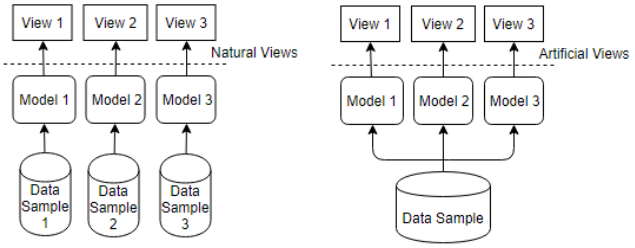

FIGURE 1. Natural (left) and artificial (right) views for multi-view learning.

In this work, we utilize deep learning to create the artificial views, and then make use of the artificial views in multi-view learning for the classification of time series data, in particular sounds. The framework makes use of *A.* the linear relationship of an ensemble of deep learning sub-models, the output of each sub-model is seen as a view, *B.* the computation of the complementarity of the views, and *C.* the formation of the input for the sub-models so that the views can be features in the time-frequency domain.

Fig. 2 shows the architecture of the proposed network. The time series data are first decomposed in the time-frequency domain to expose the spectral aspect of the time series to the deep learning sub-models. The features extracted by the sub-models form the views. These views are obviously redundant, but they are not entirely similar, due to the different configurations of the deep learners. The views from the deep learners can be combined according to their complementarity. The combined data, being more representative of the target concept, will result in better performance by the final classifier.

The proposed framework addresses the problem of the strong dependency of the performance of a trained model on the representativeness of the data. As is well known, it is tedious and expensive to construct a representative training set, due to the extensive manual curation and annotation that are needed. This is particularly true for time series data, as clear segmentation is not readily available. By treating the outputs of the deep learning sub-models as the views of the same target concept, the dependency on any one of the views could be weakened through the appropriate use of the views' complementarity. This helps reduce the need for clear segmentation and improve the generalization performance of the classifier.

A. CONSTRUCTION OF MULTIPLE VIEWS BY DEEP LEARNING

In a traditional single-view classifier, the training set consists of a single feature set $\boldsymbol{V}$. By contrast, in the multi-view setting, there are M views, denoted as $\boldsymbol{V}^{(i)}, i \in \{1, \dots, M\}$. Each of these views is sufficient for the learning of the target concept. However, in this work, the alternative approach, that of fusing the M views according to their complementarity, is proposed for use instead. This will result in a common feature set that is more representative of the target concept, compared to the individual views.

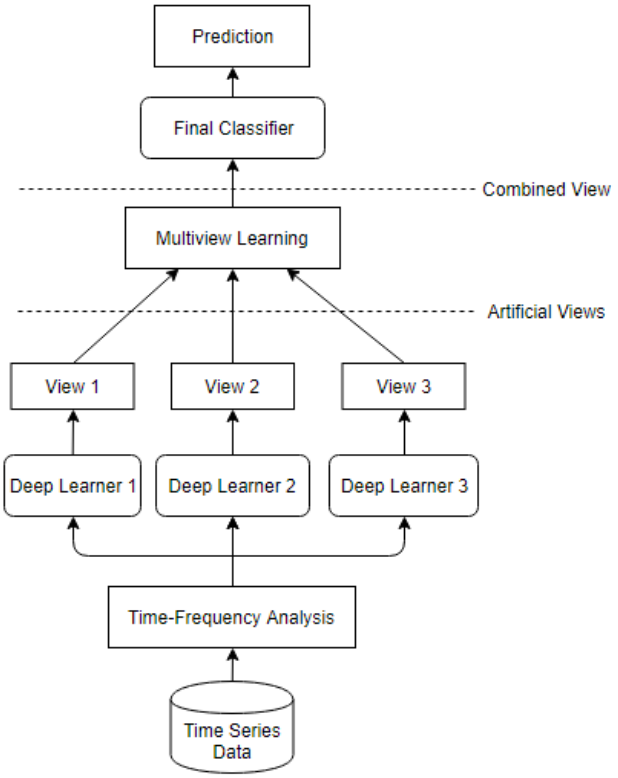

FIGURE 2. Architecture of the proposed multi-view temporal ensemble.

The proposed way to construct the views is to subject each of the input data segments, $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{d}}$ of length d, to a number of deep learning sub-models that are configured differently in terms of the number of hidden nodes. With different configurations, it is tantamount to the random split of the input data. This will result in views that are distinct from each other.

The view to be retrieved from the sub-model is the penultimate layer of the sub-model, rather than the final softmax layer. The penultimate layer can be thought of as the feature set that is extracted by deep learning from the input data. It can be represented as the approximate function $f(\boldsymbol{x})$ of the input data $\boldsymbol{x}$. As there are M different sub-models, represented as $f^{(1)}(\cdot), f^{(2)}(\cdot), \dots, f^{(M)}(\cdot)$, so M views will be available, i.e. $f^{(1)}(\boldsymbol{x}), f^{(2)}(\boldsymbol{x}), \dots, f^{(M)}(\boldsymbol{x})$.

According to the Representer theorem [12], the approximate function of a machine learning model is the linear combination of the basis functions. Thus, assuming that each of the views is a basis function, the views can be combined linearly, with appropriate weights assigned to the linear combination, as shown in (1) below.

$$\boldsymbol{V}_{combined} = \sum_{i=1}^{M} \alpha^{(i)} \boldsymbol{V}^{(i)} \tag{1}$$

The weights $\alpha^{(i)}, i \in \{1, \dots, M\}$, are the mixing coefficients of the ensemble. The restriction on $\alpha^{(i)}$ is according to the linear sum as shown in (2) below.

$$\sum_{i=1}^{M} \alpha^{(i)} = 1, \alpha^{(i)} > 0 \quad (2)$$

The value of $\alpha^{(i)}$ is the complementarity of the i-th view. It is the probability of the i-th view being compatible with the common target concept. An example of the construction of 3 different views by deep learning is shown in Fig. 3 below.

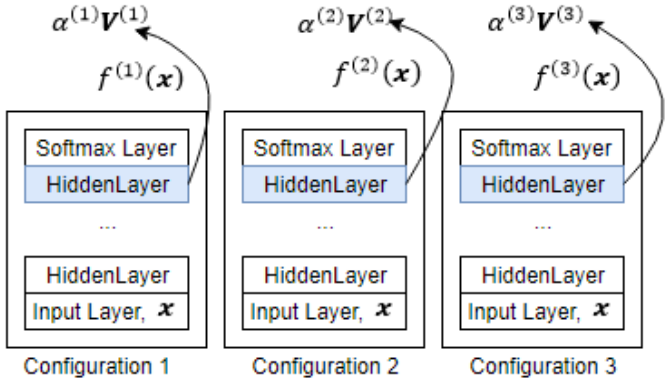

FIGURE 3. Construction of 3 different views by deep learning.

B. COMPLEMENTARITY OF MULTIPLE VIEWS

Intuitively, views that are independent and supplemental will contribute equally to the global view of the combined data. The weight of each of these views is the average weight $1/M$. This will result in a less noisy combined output. When the combined output is used as the input of the final classifier, the overall system performance will have a lower variance [13].

On the other hand, if a view contains complementary information, it will contribute more to the global view, and its weight will be higher than $1/M$. This is at the expense of the view that contains less complementary information.

So, instead of using the average weight $1/M$ for $\alpha^{(i)}$, it is proposed in this work that the complementarity of the views be used as the weights instead. The purpose of this is for the combined output to have a higher probability of a lower generalization error. Thus, the larger the contribution of the view to the global view, the more complementary it is, and the higher should be its weight.

However, the global view of a linear mixture is actually latent, given the individual views. In other words, although the global view can be obtained from the weighted sum of the individual views, it begs the question of what the weight values should be for the linear combination.

The candidate method to solve the minimization problem with two unknowns (the weights and the global view) is alternate optimization. An example of alternate optimization is the expectation maximization (EM) method used in the Gaussian mixture [14].

A similar approach is proposed for the multi-view temporal ensemble. The cost function, which has to be defined in alternate optimization, is based on that of Laplacian eigenmap [15], a non-linear data reduction technique. It will be modified in this work so that it can be used in the multi-view setting. This will be described later in Section II where the computation method for complementarity is explained.

C. FEATURES IN THE TIME-FREQUENCY DOMAIN

Time-frequency decomposition exposes the spectral changes in the time series data to the sub-model $f(\cdot)$ and is useful for the analysis of signals that are non-stationary. While there are many time-frequency analysis techniques (for example, Wigner-Ville decomposition [16], empirical mode decomposition [17], wavelet transform etc.), the most common practice, particularly for multivariate signals, is still the short time analysis method, such as the spectrogram and the Mel-frequency cepstrum [18], where the signal is split into overlapping segments and transformed to their time-frequency representation.

The sub-model, as a machine learning model, can be a generalized linear model, decision tree, k nearest neighbor, or neural network. In the past decade, deep learning, which is the composition of layers of models, has been found to be effective in the classification of raw signals.

Deep learning, as a feature extractor, has a smooth output in the feature space that can be classified easily by the final classifier. Not only can it approximate the function with an exponentially lower number of training parameters compared to a shallow network, it is also more immune to overfitting [19].

The workhorses of deep learning are deep belief net [20], convolutional neural network (CNN) [21] and long short-term memory (LSTM) recurrent neural network [22]. These models can be combined in different ways to form practical models for signal classification.

In this work, the CNN-LSTM model is proposed for use in the multi-view temporal ensemble. The reason for using the CNN-LSTM model is to extract the temporal and spectral patterns from the two-dimensional time-frequency domain. The lower CNN layer takes in the input data in two-dimension, while the LSTM works on the subsequently flattened layer in one-dimension.

The fully-connected layer before the final softmax layer is the penultimate layer. It contains the features that form the view of the sub-model. By linearly combining the penultimate layers of the CNN-LSTM sub-models, a new input will be formed for the final classifier. This qualifies the proposed multi-view temporal ensemble as an intermediate data fusion technique, rather than a late data fusion technique. This is because the penultimate layer represents the feature extracted by the sub-model, not the decision made by the sub-model.

II. METHOD

This section first provides the overview of the method to compute complementarity, followed by the details in the sub-sections.

The input, output, and initial weight values are as shown below:

Input: A set of M data matrices, each with N data points of length d, $\boldsymbol{X} = \{\boldsymbol{X}^{(i)} \in \mathbb{R}^{N\times d}\}_{i=1}^{M}$

Output: A set of M mixing coefficients, $\boldsymbol{\alpha} = \{\alpha^{(i)}\}_{i=1}^{M}$

Initialize $\boldsymbol{\alpha} = \left[\frac{1}{M}, \dots, \frac{1}{M}\right]$

The set of N data points, $\boldsymbol{X}^{(i)}$, is a mini-batch in the i-th view. N is typically a small number less than 32. For a given data set, the complementarity will be computed in many such mini-batches across the views.

A summary of the terms used in this section are shown below:

$\boldsymbol{W}$ – Weighted adjacency matrix of a view, $\boldsymbol{W} \in \mathbb{R}^{N\times N}$
$\boldsymbol{L}$ – Laplacian matrix of a view, $\boldsymbol{L} \in \mathbb{R}^{N\times N}$
$\boldsymbol{Y}$ – Spectral embedding of a view, $\boldsymbol{Y} \in \mathbb{R}^{N\times m}$, $m < N$
$\boldsymbol{W}^{(i)}$ – Weighted adjacency matrix of the i-th view
$\boldsymbol{L}^{(i)}$ – Laplacian matrix of the i-th view
$\boldsymbol{Y}^{(i)}$ – Spectral embedding of the i-th view
$\boldsymbol{L}^{(G)}$ – Laplacian matrix of the global view
$\boldsymbol{Y}^{(G)}$ – Spectral embedding of the global view
$\alpha^{(i)}$ – Complementarity (i.e. the mixing coefficient, or weight) of the i-th view

To compute complementarity, a set of N data points in the i-th view are first represented as an adjacency matrix $\boldsymbol{W}^{(i)}$ [23]. This matrix describes the distance between pairs of data points. It can be seen as the information about the local proximity of the data points in the data manifold formed by the data points.

From the adjacency matrix of the i-th view, the Laplacian matrix $\boldsymbol{L}^{(i)}$ of the i-th view can be computed quite easily. The individual views $\boldsymbol{L}^{(i)}, i \in \{1, \dots, M\}$ are then linearly combined to form the global view $\boldsymbol{L}^{(G)}$, using the initial weight values. Once the global view $\boldsymbol{L}^{(G)}$ is obtained by linear combination, its spectral encoding $\boldsymbol{Y}^{(G)}$ can be computed directly by eigen-decomposition, based on the solution of Laplacian eigenmap [24].

The global spectral embedding $\boldsymbol{Y}^{(G)}$, in turn, can be used with the Laplacian matrices $\boldsymbol{L}^{(i)}$, $i \in \{1, \dots, M\}$, to compute the weights $\alpha^{(i)}$. These weights are used to update the global view $\boldsymbol{L}^{(G)}$. In this way, through the alternate updates of the global view $\boldsymbol{L}^{(G)}$ and the weights $\alpha^{(i)}$, the global spectral embedding $\boldsymbol{Y}^{(G)}$ will converge to $\boldsymbol{Y}^{(G)*}$. At this point in time, the weights $\alpha^{(i)}$ will represent the complementarity of the i-th view, relative to the other views. This process is illustrated in Fig. 4 below.

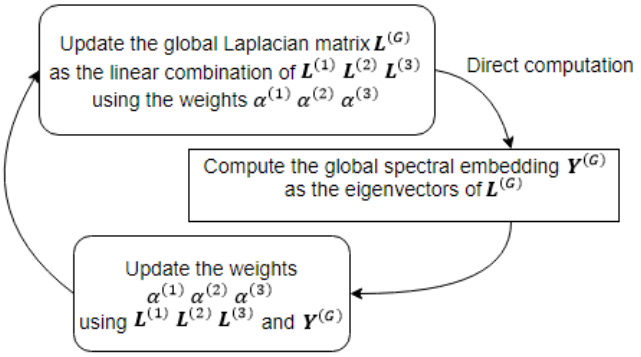

FIGURE 4. Alternate optimization of $\boldsymbol{L}^{(G)}$ and $\alpha^{(i)}$.

The iterative process in Fig. 4 can be summarized by the following steps:

(1) Obtain $\boldsymbol{L}^{(i)}$ from a set of N co-occurring data vectors of the same class from the i-th view
(2) Align the individual $\boldsymbol{L}^{(i)}$ to the global spectral embedding in 2 steps:
   a. obtain $\boldsymbol{L}^{(G)}$ from $\boldsymbol{L}^{(i)}$ by linear combination, according to the weights $\alpha^{(i)}$
   b. obtain $\boldsymbol{Y}^{(G)}$ from $\boldsymbol{L}^{(G)}$ by eigen-decomposition, formed by the m eigen-vectors that correspond to the m smallest eigenvalues other than λ_0, where $0 = \lambda_0 \leq \lambda_1 \leq \cdots \lambda_1 \leq \cdots \leq \lambda_{N-1}$, and $m < N$
(3) Update the values of $\alpha^{(i)}$, which is the inverse of the trace of $\boldsymbol{Y}^{(G)^T}\boldsymbol{L}^{(i)}\boldsymbol{Y}^{(G)}$

Iterate through (2) if the norm of the change in $\boldsymbol{\alpha}$ is bigger than a user-defined threshold.

The above is the overview of how complementarity, in sets of N-point mini-batches, is computed. The details will now be elaborated in the following sub-sections: *A.* Adjacent Matrix and Laplacian Matrix, *B.* Spectral Embedding of the Data Manifold, *C.* Multi-view Laplacian Eigenmaps, *D.* Complementarity, *E.* Co-occurrence and Class-Specificity, and *F.* CNN-LSTM Sub-Model.

A. ADJACENCY MATRIX AND LAPLACIAN MATRIX

For a set of N data points $\{\boldsymbol{x}_i \in \mathbb{R}^{\boldsymbol{d}}\}_{i=1}^{N}$, the weighted adjacency matrix $\boldsymbol{W}$ is a square symmetric matrix of size $N \times N$. The (i,j)-th entry of $\boldsymbol{W}$ can be computed according to (3), as shown below.

$$[\boldsymbol{W}]_{i,j} = \begin{cases} exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{\sigma^2}\right) & if\ x_i, x_j\ connected \\ 0 & otherwise \end{cases} \quad (3)$$

According to (3) above, the entry $[\boldsymbol{W}]_{i,j}$ is cleared to 0 if the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $i, j \in \{1, \dots, N\}$, are not connected. Whether $\boldsymbol{x}_i$ is connected to $\boldsymbol{x}_j$ depends on whether $\boldsymbol{x}_j$ is in the k-nearest neighbourhood of $\boldsymbol{x}_i$, where $k < N$ is a user-defined hyper-parameter. The value of $[\boldsymbol{W}]_{i,j}$ represents the

proximity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the data manifold formed by the set of N data points. The closer the points, the higher the value of the proximity.

B. SPECTRAL EMBEDDING OF THE DATA MANIFOLD

The spectral embedding $\boldsymbol{Y}^*$ of N data points in a single view can be obtained by a method called Laplacian eigenmap [12]. Laplacian embedding is a data reduction technique that projects the data points onto the alternative spectral view while preserving the local proximity of the data points in the new view. Conceptually, this preservation is achieved by the minimization of the cost function $J(\boldsymbol{Y})$ as shown in (4) below:

$$J(\boldsymbol{Y}) = \sum_{i,j\in\{1,\dots,N\}} \left\|\boldsymbol{y}_i - \boldsymbol{y}_j\right\|^2 [\boldsymbol{W}]_{i,j} \tag{4}$$

As seen from (4) above, the cost function $J(\boldsymbol{Y})$ is the total amount of differences between two embedded vectors ($\boldsymbol{y}_i$ and $\boldsymbol{y}_j$, $i,j \in \{1,\dots,N\}$), modulated by $[\boldsymbol{W}]_{i,j}$. When the data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in close proximity in the data manifold, the value of the adjacency matrix $[\boldsymbol{W}]_{i,j}$ will be large, thus contributing more to the cost function. This helps to promote the preservation of the local proximity in the resultant spectral embedding.

The solution $\boldsymbol{Y}^*$ of the above minimization problem [24] can be shown to reduce to

$$\boldsymbol{Y}^* = arg \min_{\boldsymbol{Y}^T\boldsymbol{D}\boldsymbol{Y}=1, \boldsymbol{Y}^T\boldsymbol{D}\boldsymbol{1}=0} tr(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y}) \tag{5}$$

In (5) above, $\boldsymbol{L}$ is the Laplacian matrix that can be computed as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where the diagonal matrix $\boldsymbol{D}$ is the degree of connectedness in the data manifold, i.e. $[\boldsymbol{D}]_{i,i} = \sum_{j=1}^{N} [\boldsymbol{W}]_{i,j}$.

Importantly, finding $\boldsymbol{Y}^* = arg \min_{\boldsymbol{Y}^T\boldsymbol{D}\boldsymbol{Y}=1, \boldsymbol{Y}^T\boldsymbol{D}\boldsymbol{1}=0} tr(\boldsymbol{Y}^T\boldsymbol{L}\boldsymbol{Y})$ is equivalent to finding the eigenvectors $\boldsymbol{Y}^*$ of the generalized eigenvalue problem $\boldsymbol{L}\boldsymbol{Y}^* = \boldsymbol{\lambda}\boldsymbol{D}\boldsymbol{Y}^*$. Thus, given the Laplacian matrix $\boldsymbol{L}$, the spectral embedding $\boldsymbol{Y}^*$ can be found easily.

With $\boldsymbol{L}$ and $\boldsymbol{Y}^*$ known, the cost value $tr(\boldsymbol{Y}^{*T}\boldsymbol{L}\boldsymbol{Y}^*)$ can be computed. The lower the cost value, the closer it is to reach the objective of preserving the local proximity of the data points in the spectral embedding.

It is interesting to note that the spectral embedding $\boldsymbol{Y}^*$ can be computed directly by the eigen-decomposition of $\boldsymbol{L}$, even though it is a minimization problem that comes with a cost function. Also, the Laplacian eigenmap as described above is meant for single view only. It will have to be modified for use in the multi-view setting.

C. MULTI-VIEW LAPLACIAN EIGENMAP

With multiple views, say M views, the solution of the minimization problem in (5) becomes not useful. Not only is the global view $\boldsymbol{L}^{(G)}$ unknown, the weights for forming $\boldsymbol{L}^{(G)}$ is also unknown.

Following the method of patch alignment with multi-view spectral embedding for image and video [25], it is proposed that the global view $\boldsymbol{L}^{(G)}$ be formed by linearly combining the individual view $\boldsymbol{L}^{(i)}$, based on the weights $\alpha^{(i)}$ (initialized as $1/M$).

$$\boldsymbol{L}^{(G)} = \sum_{i=1}^{M} \left(\alpha^{(i)}\right)^r \boldsymbol{L}^{(i)}, r > 1 \tag{6}$$

The minimization problem in (5) will then become (7) as shown below:

$$\boldsymbol{Y}^{(G)*} = arg \min_{\boldsymbol{Y}^T\boldsymbol{Y}=\boldsymbol{1}} \sum_{i=1}^{M} \left(\alpha^{(i)}\right)^r tr(\boldsymbol{Y}^{(G)^T}\boldsymbol{L}^{(i)}\boldsymbol{Y}^{(G)}) \tag{7}$$

The hyper-parameter r has been introduced in (6) with $r > 1$. It is a trick to induce each view to contribute unequally to the global spectral embedding $\boldsymbol{Y}^{(G)*}$. If $r = 1$, the alternate optimization will end up with only the best view instead of the complementary views [26].

$\boldsymbol{Y}^{(G)*}$ can be computed directly as the set of m eigenvectors of the global Laplacian matrix $\boldsymbol{L}^{(\boldsymbol{G})}$. They corresponds to the m smallest eigenvalues other than $\lambda_0 = 0$.

The eigenvectors are arranged in order of the eigenvalue, from the smallest eigenvalue to the largest value, up to the specified dimension $m < N$, where N is the dimension of the Laplacian matrix.

The eigenvectors with the smallest eigenvalues are selected because a compact representation in the projection space is desired. However, since the eigenvector associated with the smallest eigenvalue is likely to represent the noise, it will have to be discarded. Thus, only column vectors $\boldsymbol{Y}_{\cdot j}, j \in \{2, \cdots m+1\}$ are used. The shape of $\boldsymbol{Y}$ is $(N \times m)$, where N is the number of data points in the mini-batch and m is the user-defined hyper-parameter value for the number of selected eigenvectors.

D. COMPLEMENTARITY

The complementarity of the i-th view, or the weight $\alpha^{(i)}$, is defined as shown in (8) below. It is the inverse of the cost value of the i-th view, normalized across the M views.

$$\alpha^{(i)} = \left(1/tr\left(\boldsymbol{Y}^{(G)^T}\boldsymbol{L}^{(i)}\boldsymbol{Y}^{(G)*}\right)\right)^{\frac{1}{r-1}} \Big/ \sum_{i=1}^{M} \left(1/tr\left(\boldsymbol{Y}^{(G)^T}\boldsymbol{L}^{(i)}\boldsymbol{Y}^{(G)*}\right)\right)^{\frac{1}{r-1}} \tag{8}$$

The L_2 norm in (9) below can be used as the criterion for the convergence of the alternate optimization.

$$\sqrt{\sum_{i=1}^{M} \left(\alpha_k^{(i)} - \alpha_{k-1}^{(i)}\right)^2} < \varepsilon \tag{9}$$

In (9) above, $\alpha_k^{(i)}$ is the weight at the k-th iteration and $\alpha_{k-1}^{(i)}$ is the weight at the $(k-1)$-th iteration. ε is a user-defined threshold that is set to a value much smaller than 1. The iteration continues until the change in the norm of $\boldsymbol{\alpha}$ in successive iterations is smaller than ε.

At convergence, $\alpha^{(i)}$ will have a value that is different from $1/M$. A value larger than $1/M$ means that the view is more

complementary and contributes more to the global spectral embedding (compared to the other views). Conversely, a value smaller than $1/M$ means that the view is less complementary and contributes less to the global spectral embedding (compared to the other views).

E. CO-OCCURRENCE AND CLASS-SPECIFICITY

The computation of complementarity is a way to produce data that are more representative of the target concept. Thus, when computing complementarity, the data points across the views must describe the same target concept. For time series data, this translates to the following rules:

1) The data points across the views must be aligned in time, i.e. co-occurring.
2) The data points must belong to the same class, i.e. class-specific.

1) CO-OCCURRENCE

Co-occurrence does not preclude the shuffling of the data points in the individual views, which is often a necessary operation to achieve independent and identical distribution of the input data for model training. It merely states that the same shuffled order must be used across the views so that the data points across the views will occur at the same time point and thus describe the same target concept.

To ensure co-occurrence at the penultimate layers of the deep learning sub-models, the same data set (shuffled and so random in order) will have to be used as the inputs for all the sub-models. As long as there is no randomization of the data vectors in the sub-models, the outputs at the penultimate layers of the sub-models will be co-occurring too. These outputs, which are co-occurring, can then be used as the views for multi-view learning.

The rule of co-occurrence has to be enforced during both the training and the testing process.

2) CLASS-SPECIFICITY

The rule of class-specificity applies to multi-view learning because complementarity can only be determined among data of the same class. It cannot be used for classes that are different.

The cats and dogs analogy illustrates this idea. For a set of dog images and a set of cat images, it is meaningful to define the complementarity of the images within the sets (either the cats or the dogs) but not across the sets. This is because complementarity is ill defined for a combined data set that has different concepts.

The proposed solution to satisfy the requirement of class-specificity is to re-arrange the outputs of the sub-models by class, yet without disturbing the time order necessary for co-occurrence. Complementarity is then computed on the class-specific data, which is then combined across the views.

This process, when carried out separately for the classes, will result in linearly combined data that are class-specific. The data of these classes will have to be stacked together as one single feature set and then shuffled so that they can be used as the input by the final classifier.

The rule of class-specificity seems to contradict the testing requirement in machine learning, where the class in the test set is assumed unknown. Class-specific data seems impossible when the class information is not available in the test set.

Actually, this is not a problem in the proposed multi-view temporal ensemble. This is because the sub-models can predict the class during testing. The predicted class, instead of the actual class, can be used to re-arrange the outputs of the sub-models. The linearly combined data, based on the predicted classes, are then used by the final classifier for the final prediction.

F. CNN-LSTM SUB-MODEL

The multi-view temporal ensemble, when applied to time series data, entails some considerations as shown below:

1) In general, it is a good idea to decompose the time series into the time-frequency representation of the signals so that spectral features are exposed to the learner.
2) The sub-model will need to be a good learner because a good learner is able to produce data that are smooth with respect to their target class labels, thus making the criterion of local proximity in the spectral embedding achievable.
3) Different configurations of the same sub-model could be used to generate artificial views of the input data that may not be segmented well.

The use of CNN as the front end has been verified to be an effective method for time series data in previous works [27]. Thus, instead of using the raw data of a time series segment as the input of the classifier, it is proposed that the data be transformed into its time-frequency domain. The CNN will accept the data in the time-frequency domain in the tensor format of channel × height × width, where height is the time steps and width is the frequency bins.

With the 2-dimensional CNN as the front end, a 1-dimensional CNN can be added on top of it to extract the temporal features across the feature maps. This is then followed by an LSTM to extract the remaining high-level temporal features. Different configurations of such CNN-LSTM models can be used as the sub-models of the multi-view temporal ensemble to produce the views that are needed by multi-view learning.

III. DATA EXPERIMENT AND RESULT

This section will describe the data experiment done on the ESC-50 data set [28]. The ESC-50 data set is chosen for this work because the signals are non-stationary with no obvious time-dependent structure. The purpose is to validate the performance of the multi-view temporal ensemble on a time series data set without curation or manual segmentation.

The work is presented in four parts: (1) the description of the data set, (2) the spot-checking to get the general benchmark of the data set, (3) the performance evaluation of the individual views, each of which is a CNN-LSTM model configured in a particular way, and (4) the performance

evaluation of the multi-view temporal ensemble, based on the penultimate outputs of the CNN-LSTM sub-models.

A. ESC-50 DATA SET

The ESC-50 data set is a univariate numeric time series data set with 2,000 audio recordings constructed from the sound clips in the Freesound project [29]. There are 50 classes, of which 22 classes are the sounds of animals and humans (dog, rooster, etc.), and the rest natural or mechanical sounds (door knock, siren, etc.).

Each of the 50 classes has 40 recordings. Each recording is a 5 second long .wav file (110,250 samples at 22,050 Hz). They can be decoded by the *avconv* library package and processed using the *LibROSA* library package in the Pyhton programming environment.

According to [28], the human capabilities in recognizing the sounds in the data set is estimated at 81.3%. The performance varies across the sounds, with a low of 34.1% for the washing machine noise and almost 100% for crying babies. It is postulated that trained and attentive listeners could reach 90% accuracy for the data set.

With just 40 recordings per class in this data set, there is hardly enough training instances per class for deep learning. To overcome this problem, each of the 5-second audio clips is split into 9 overlapping segments, with 20,992 samples per segment (0.952 second). The content of each segment is arbitrarily segmented with no curation, other than the removal of segments that have very low power, likely to be due to silence.

Within each segment, 41 time-consecutive frames, each with 512 samples (0.023 second), are formed. Each frame is subjected to Fourier transform and converted to the energy values of a 60-bin Mel-frequency cepstrum.

As a result, the data of each segment is a 2-D matrix with 41 time steps and 60 coefficients. The 2-D matrix has a total of 2,460 coefficients in it and is associated with a particular sound class.

B. SPOT-CHECKING

Previous work [15] shows that using a deep learning approach with two convolutional layers with max-pooling followed by two fully connected layers can produce a classification accuracy of 64.5%.

It is also interesting to note that not all deep learning will yield good result on the ESC-50 data set. To show this, a deep learning model with two LSTM layers, a dense layer, and a softmax layer, as shown in Fig. 5 below, was used on the time-frequency representation of the ESC-50 data set.

The result (60.9% accuracy) is less than appealing despite the use of dropout as regularization. This is likely due to the spectral features not being extracted by the LSTMs as well as the CNNs.

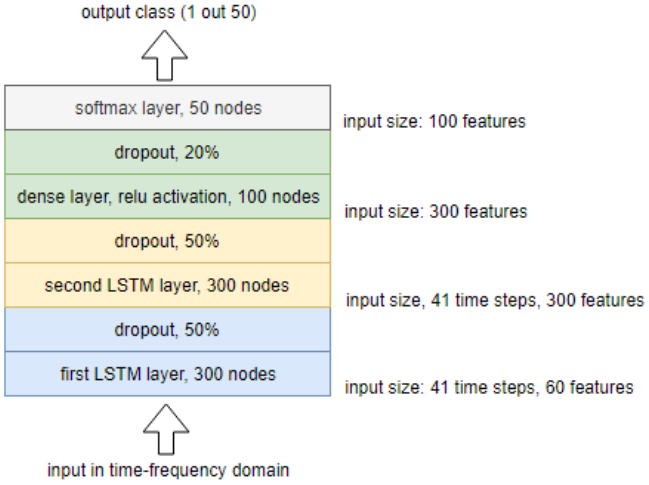

FIGURE 5. Deep learning with two layers of LSTM, ESC-50.

C. PERFORMANCE OF THE INDIVIDUAL VIEWS

Three configurations of the CNN-LSTM model are used in this work, referred to here as View 1, 2, and 3.

The CNN-LSTM model used for View 1 is shown in Fig. 6 below. It consists of two groups of 2-D CNN layers, one group of 1-D CNN layer, one group of LSTM layer, a fully connected dense layer, and a softmax layer. It has 1,454,226 trainable parameters.

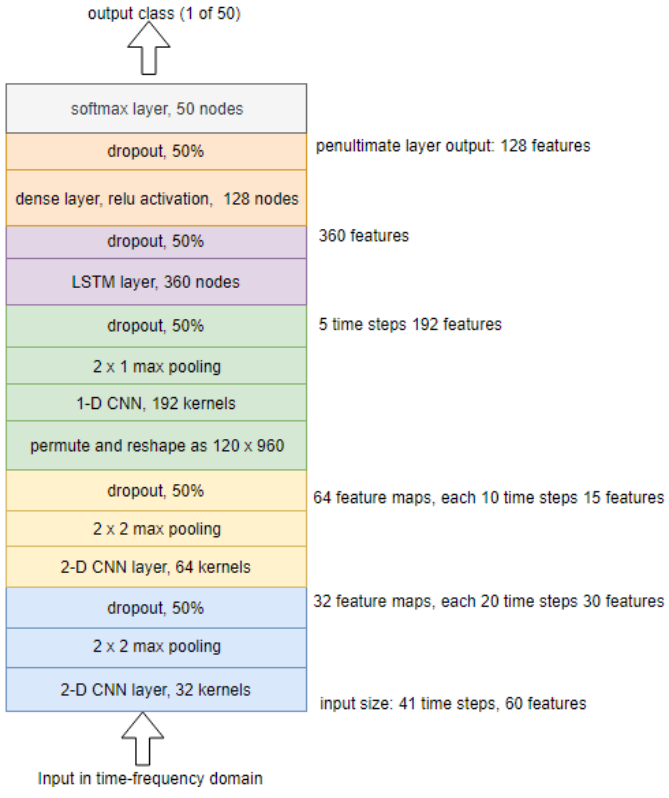

FIGURE 6. Configuration of CNN-LSTM for View 1.

The input of the CNN-LSTM model is a tensor of size (1, 41, 60), where the number of channels is 1, the number of time

steps is 41, and the number of attributes is 60. As Fig. 6 shows, the input is filtered by 32 kernels in the first CNN layer. This will result in 32 feature maps. After max pooling by a 2×2 region, the tensor output of the first 2-D CNN group is (32, 20, 30).

The 2-D CNN group (CNN, max pooling and dropout) is then repeated, this time with 64 kernels, giving rise to the second 2-D CNN group. Together, the two 2-D CNN groups serve as a deep learner to capture the invariant features across the time-frequency structure of the audio segment.

The features are then re-organized as a matrix of 10 time-steps of 960 features. This is used as the input for the 1D-CNN layer. The kernels in the 1D-CNN layer have a size of 3 time steps by 960 features, covering all the 960 features in one dimension.

The output from the 1D-CNN layer is fed to an LSTM layer to extract the remaining high-level features. Thereafter, a fully connected layer with ReLU activation is used with a softmax layer to implement multi-class classification.

Validation of the performance of the CNN-LSTM model for View 1 is done by 66/33 training/test splitting. The classification accuracy is used as the performance metrics, since the data set is a balanced one. Table I shows the View 1 result (classification accuracy) over 20 epochs. It shows that the result has converged over the epochs. The final result, at 83.94%, is close to the reported top scores for this data set.

TABLE I.
CLASSIFICATION ACCURACY (%) OVER 20 EPOCHS, VIEW 1 (ESC-50)

19.27	35.90	45.18	51.73	60.45	64.44	67.61	70.65	72.32	74.16
75.09	77.62	78.99	79.89	81.19	82.17	82.23	82.98	83.94	83.94

The configurations of the CNN-LSTM models for View 2 and View 3 differ from that for View 1 in terms of the number of kernels used in the two 2-D CNN groups. Instead of 32 and 64 kernels for View 1, they are 8 and 16 for View 2, and 16 and 32 for View 3. As a result, the number of trainable parameters of the CNN-LSTM models for View 2 and View 3 are 990, 834 and 1, 138, 898 respectively.

There is no sure way of knowing which set of configurations is better for the given data set. The purpose here is not to select a good configuration for the given data set. Rather, it is to use the different configurations to generate random split of the features so that multiple views can be generated so that they can be linearly combined based on their complementarity to boost the generalization performance.

Based on the CNN-LSTM model for View 2, the results over 20 epochs are shown in Table II below. At 82.64%, it is close to the results of View 1.

TABLE II.
CLASSIFICATION ACCURACY (%) OVER 20 EPOCHS, VIEW 2 (ESC-50)

21.00	35.48	47.56	54.30	60.91	64.94	67.84	69.27	72.03	72.55
75.49	75.36	76.85	78.45	79.50	79.62	79.72	81.64	81.58	82.64

The results produced by the CNN-LSTM model for View 3 are shown in Table III below. At 83.06%, it is, again, similar to the results of View 1 and View 2.

TABLE III.
CLASSIFICATION ACCURACIES (%) OVER 20 EPOCHS, VIEW 3 (ESC-50)

18.21	35.21	43.62	52.57	57.24	62.24	65.69	68.48	70.98	73.68
73.51	76.82	78.37	78.39	79.47	79.39	80.64	81.71	82.31	83.06

The results in Table I, Table II and Table III show that the single views from the CNN-LSTM models are sufficient for state-of-the-art performance for sound classification. The purpose of this work, however, is to show that when a number of such views are available from the same set of time series data, multi-view learning based on the proposed complementarity can boost the performance further.

D. PERFORMANCE OF THE MULTI-VIEW TEMPORAL ENSEMBLE

The penultimate layer of the CNN-LSTM model has 128 nodes. These are the features as extracted by the model. They form the view of the time series data as seen by the model.

There are three CNN-LSTM models in the ensemble, each configured differently from the rest. As such, the ensemble has three complementary views.

In both training and testing, the views will have to be computed for complementarity in small mini-batches of N data points according to the rules of co-occurrence and class-specificity. After linear combination, the combined data will become the input data for the final classifier, which is the classifier placed on top of the ensemble for multi-class classification.

The final classifier used here is a neural network with a hidden layer and a 50-output softmax layer. As for all classifiers, it will have to be trained before it can be used for prediction. Since its training data are now more representative of the target concept, compared to the single view of the CNN-LSTM models, better performance is expected.

Validation of the performance of the proposed multi-view temporal ensemble is done by 66/33 training/test splitting. It was found that the classification accuracy improved to 85.5%, which is better than that of any of the single view.

Fig. 7 below shows the performance of the multi-view temporal ensemble versus those of the individual views. It shows that the complementary data in the individual views

boost the system performance after they were blended according to their complementarity.

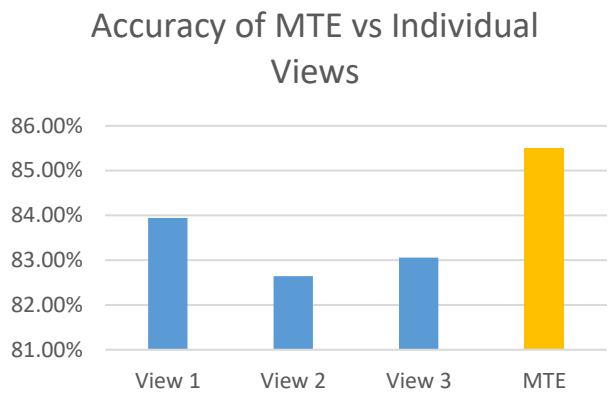

FIGURE 7. Comparison of MTE vs individual view (ESC-50) in terms of classification accuracies.

The mean and standard deviation of 10-times bootstrap resampling is used to compare the effect of the models. This is shown in Table IV below. It shows that the accuracy for the individual sub-models (83.59%, 81.05%, and 82.58%) improves to 85.97% when the multi-view temporal ensemble is used to blend the views and then reclassified by the final classifier.

TABLE IV
MEAN AND STANDARD DEVIATION OF 10-TIMES BOOTSTRAP RESAMPLING OF INDIVIDUAL VIEWS AND MTE (ESC-50)

	Mean	Std Dev
View 1	83.59%	2.46%
View 2	81.05%	2.44%
View 3	82.58%	2.85%
MTE	85.97%	1.84%

The better performance is due to the final classifier having a more complete view of the underlying phenomenon. This can be explained from the perspective of the bias-variance dilemma [30]. With complementary views, the consensus between the views reduces the variance while the increase in the discriminatory information reduces the bias.

IV. CONCLUSION

In this paper, the exploration of deep learning was extended to the field of ensemble technique and multi-view learning. An intermediate data fusion technique, called the multi-view temporal ensemble, is proposed for use with time series data such as sound to boost the generalization performance of classification. In the proposed method, the outputs of the sub-models in the ensemble are linearly combined according to their complementarity so that the features, used as the input by the final classifier, can be more representative of the target concept.

It is proposed that the cost function of the Laplacian eigenmap be adopted for alternate optimization to solve the two-fold problem: (1) the mixing coefficients are unknown, and (2) the global view (i.e. the weighted sum of the individual views) is also unknown. The alternate update of the two unknowns will result in the minimization of the cost function, resulting in the convergence of the mixing coefficients. This technique can be used with time series data with two rules: (1) co-occurrence, and (2) class-specificity.

A CNN-LSTM ensemble framework was described and tested with a time series data set. The result shows that without manual segmentation and curation, the time series data can be classified with greater generalization performance in the multi-view setting, compared to deep learning based on single view alone.

REFERENCES

[1] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*, 1998.
[2] S. Sun, "A survey of multi-view machine learning," *Neural Computing and Applications*. 2013.
[3] J. Liu, C. Wang, J. Gao, and J. Han, "Multi-View Clustering via Joint Nonnegative Matrix Factorization," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, 2013.
[4] I. Muslea, S. Minton, and C. A. Knoblock, "Active learning with multiple views," *J. Artif. Intell. Res.*, 2006.
[5] B. Tan, E. Zhong, E. W. Xiang, and Q. Yang, "Multi-transfer: Transfer learning with multiple views and multiple sources," *Stat. Anal. Data Min.*, 2014.
[6] C. O. Sakar, O. Kursun, and F. Gurgen, "Ensemble canonical correlation analysis," *Appl. Intell.*, 2014.
[7] T. Niu, S. Zhu, L. Pang, and A. Elsaddik, "Sentiment analysis on multi-view social data," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016.
[8] W. Wang, R. Arora, K. Livescu, and J. A. Bilmes, "Unsupervised learning of acoustic features via deep canonical correlation analysis," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2015.
[9] T. Bänziger, D. Grandjean, and K. R. Scherer, "Emotion Recognition From Expressions in Face, Voice, and Body: The Multimodal Emotion Recognition Test (MERT)," *Emotion*, 2009.
[10] Yan Zhang, Danjv Lv, and Yili Zhao, "Multiple-View Active Learning for Environmental Sound Classification.," *International Journal of Online Engineering*. 2016.
[11] K. Nigam and R. Ghani, "Analyzing the effectiveness and applicability of co-training," in *Proceedings of the ninth international conference on Information and knowledge management - CIKM '00*, 2000.
[12] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bull. Am. Math. Soc.*, vol. 39, no. 1, pp. 1–49, 2002.
[13] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Netw. Comput. Neural Syst.*, 1997.
[14] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Stat. Soc. Ser. B Methodol.*, vol. 39, no. 1, pp. 1–38, 1977.
[15] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," *NIPS*, 2001.
[16] W. Martin and P. Flandrin, "Wigner-Ville Spectral Analysis of Nonstationary Processes," *IEEE Trans. Acoust.*, 1985.
[17] N. Rehman and D. P. Mandic, "Multivariate empirical mode decomposition," *Proc. R. Soc. A Math. Phys. Eng. Sci.*, 2010.
[18] V. Tiwari, "MFCC and its applications in speaker recognition," *Int. J. Emerg. Technol.*, 2010.
[19] Y. Bengio, "Learning Deep Architectures for AI," *Found. Trends Mach. Learn.*, 2009.
[20] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A Fast Learning Algorithm for Deep Belief Nets," *Neural Comput.*, 2006.
[21] K. Fukushima, "Artificial vision by multi-layered neural networks: Neocognitron and its advances," *Neural Networks*, 2013.
[22] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.*, 1997.
[23] L. Hogben, "Spectral graph theory and the inverse eigenvalue problem of a graph," in *Electronic Journal of Linear Algebra*, 2005.
[24] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, 2003.
[25] T. Xia, D. Tao, T. Mei, and Y. Zhang, "Multiview spectral embedding," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 40, no. 6, pp. 1438–1446, 2010.
[26] M. Wang, X. S. Hua, X. Yuan, Y. Song, and L. R. Dai, "Optimizing multi-graph learning: towards a unified video annotation scheme," *Proc. 15th Int. Conf. Multimed.*, pp. 862–871, 2007.
[27] N. Hatami, Y. Gavet, and J. Debayle, "Classification of Time-Series Images Using Deep Convolutional Neural Networks," 2017.
[28] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," *Proc. 23rd ACM Int. Conf. Multimedia, MM 2015*, pp. 1015–1018, 2015.
[29] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proc. of the ACM Int. Conf. on Multimedia (ACM MM)*, 2013.
[30] S. Geman, E. Bienenstock, and R. Doursat, "Neural Networks and the Bias/Variance Dilemma," *Neural Comput.*, 1992.