Hybrid Networks: Improving Deep Learning Networks via Integrating Two Views of Images

Sunny Verma[1], Wei Liu[1,*], Chen Wang[2], and Liming Zhu[2]

[1] Advanced Analytics Institute, School of Software, University of Technology Sydney, Sydney, Australia
`Sunny.Verma@student.uts.edu.au, Wei.Liu@uts.edu.au`
[2] CSIRO, Data61, Sydney, Australia
`Chen.Wang@data61.csiro.au, Liming.Zhu@data61.csiro.au`

Abstract. The *principal component* analysis network (**PCANet**) is an unsupervised parsimonious deep network, utilizing *principal components* as filters in the layers. It creates an amalgamated view of the data by transforming it into column vectors which destroys its spatial structure while obtaining the principal components. In this research, we first propose a tensor-factorization based method referred as the **Tensor Factorization Networks** (**TFNet**). The **TFNet** retains the spatial structure of the data by preserving its individual modes. This presentation provides a minutiae view of the data while extracting matrix factors. However, the above methods are restricted to extract a single representation and thus incurs information loss. To alleviate this information loss with the above methods we propose **Hybrid Network** (**HybridNet**) to simultaneously learn filters from both the views of the data. Comprehensive results on multiple benchmark datasets validate the superiority of integrating both the views of the data in our proposed **HybridNet**.

Keywords: Tensor Decomposition, Classification, Feature Extraction

1 Introduction

Features extraction is an important operation in the development of classification tasks. This process has matured with a multitude of developments evolving from the machine learning, computer vision, data mining, and signal processing communities [26]. Today, in the era of deep learning, features are extracted by processing the data through multiple stacked convolution layers. The crux of feature extraction with deep architectures is to perform sophisticated operations with multiple layers in a sequential manner [1]. The features obtained by the deep networks promise better feature representations than the conventional shallow networks. However, these networks are trained via stochastic optimization techniques which necessitates multiple flops of the same data to effectively learn its representation. This leads to longer training time while obtaining feature representations from the data. Furthermore, the fundamental operations utilized in deep networks are expensive regarding memory and space complexities. This limits the usability of deep architectures on micro devices like cellphones. The current research trend focuses on alleviating the above problem associated with the development of deep architectures [11],[14].

PCANet is one such promising architecture: it is an unsupervised deep parsimonious network extracting *principal components* in its cascaded layers [4]. Due to the remarkable performance of **PCANet** on several benchmark face datasets, the network is currently accepted as a simple deep learning baseline for image classification. However, the features extracted by **PCANet** (and its later variant **FANet** [13]) does not achieve similar performance on challenging object recognitions datasets like CIFAR-10 [15]. There is a major reason for this performance degradation: vectorizing image patches (which we call the amalgamated view of the data) while extracting *principal components*. This results in loss of spatial information present in the images. This loss is amplified when one vectorizes an *RGB*-image which incurs the loss of both color and spatial information present in the data. However, this operation is inherent with the *principal components* analysis and necessitates the development of sophisticated techniques to reduce this information loss.

In this paper, we explore the feasibility of reducing the spatial information in **PCANet** by first devising **Tensor Factorization Networks** (**TFNet**). The **TFNet** extracts features from the original multi-mode data (which we call the minutiae view of the data) by utilizing multi-linear algebraic operations in its cascaded deep architecture. Contrary to the **PCANet** the **TFNet** *does not vectorizes the data while learning its convolution filters and hence preserves the spatial information present in the data.* Also, each mode of the multi-mode data is decomposed individually providing several degrees of freedom to the filter learning procedure in the **TFNet**.

We then propose the **Hybrid Network** (**HybridNet**) which integrates the advantages of both the **PCANet** and the **TFNet**. The **HybridNet** utilizes both tensor and matrix decompositions techniques while obtaining features representations from different views of the data. Our hypothesis is that the information from either the amalgamated view or the minutiae view is individually insufficient for classification. Since the information captured from the two views contains complementary information and hence both of them are necessary and their integration can enhance the performance of classification systems. To validate our claims we utilize multiple real world benchmark datasets to extensively evaluate the classification performance of the features obtained through the **PCANet**, the **TFNet**, and the **HybridNet**.

We summarize our contributions in this paper as follows:

- We propose **Tensor Factorized Network** (**TFNet**) which preserves the spatial information present in the data and enables extraction of matrix factors from the minutiae view of the tensorial data.
- We propose **Hybrid Network** (**HybridNet**) which integrates the filter learning procedure from the amalgamated view and the minutiae view of the data and simultaneously extracts features from them.

– We perform comprehensive evaluations with the features obtained via **PCANet**, **TFNet**, and the **HybridNet** on multiple benchmark real world datasets.

The rest of the paper is organized in the following sections: prior work (i.e. the **PCANet**) and tensor preliminaries is presented in Section 2. Our proposed **TFNet** and **HybridNet** are presented in Section 3 and Section 4 respectively. Next we describe the experimental setup, results and discussions in Section 5 and finally the conclusions in Section 6.

2 Background

2.1 PCANet

The **PCANet**'s 3-layer architecture is summarized in this section. Assume that there are N input training images denoted as $\{I_i\}_{i=1}^{N}$ of size m×n. Also, assume learning L_1 and L_2 number of filters in the first and the second layer respectively.

The First Layer The procedure begins by extracting overlapping patches of size $k_1 \times k_2$ around each pixel in the image; the patches from image I_i are denoted as $\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,\tilde{m}\tilde{n}} \in \mathbb{R}^{k_1k_2}$, where $\tilde{m} = m - \lceil \frac{k_1}{2} \rceil$ and $\tilde{n} = n - \lceil \frac{k_2}{2} \rceil$,where $\lceil z \rceil$ gives the smallest integer greater than or equal to z. Then, the obtained patches are *vectorized* and the mean of the image patches is subtracted from them to obtain the patch matrix as $\boldsymbol{X}_i \in \mathbb{R}^{k_1k_2 \times \tilde{m}\tilde{n}}$. Obtaining patch representation for all the images one obtains

$$\boldsymbol{X} \in \mathbb{R}^{k_1k_2 \times N\tilde{m}\tilde{n}} \tag{1}$$

the PCA minimizes the reconstruction error with a family of orthonormal filters known as L_1 principal eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$ calculated as below:

$$\min_{V \in \mathbb{R}^{k_1k_2 \times L_1}} \|\boldsymbol{X} - VV^T\boldsymbol{X}\|_F, \; s.t. \quad V^TV = I_{L_1} \tag{2}$$

where I_{L_1} is an identity matrix of size $L_1 \times L_1$, the filters are then expressed as:

$$W^1_{l_{\mathbf{PCANet}}} = mat_{k_1,k_2}(ql(\boldsymbol{X}\boldsymbol{X}^T)) \in \mathbb{R}^{k_1 \times k_2}, \quad l = 1, 2, ..., L_1 \tag{3}$$

where $mat_{k_1,k_2}(v)$ is a function that maps $v \in \mathbb{R}^{k_1k_2}$ to a matrix $W \in \mathbb{R}^{k_1 \times k_2}$, and $ql(\boldsymbol{X}\boldsymbol{X}^T)$ denotes the l-th principal eigenvector of $\boldsymbol{X}\boldsymbol{X}^T$. Each input image I_i in this layer are then convolved with the L_1 filters obtained as below:

$$I^l_{i_{\mathbf{PCANet}}} = I_i * W^1_{l_{\mathbf{PCANet}}}, \quad i = 1, 2, ..., N, \quad l = 1, 2, ..., L_1 \tag{4}$$

where $*$ denotes the 2D convolution. The boundary of image I_i is zero-padded before convolution to obtain $I^l_{i_{\mathbf{PCANet}}}$ with the same dimensions as in I_i. From Eq. 4, one obtains $N \times L_1$ outputs attributed as the output from the first layer.

The Second Layer In the second layer, the overlapping patches from the input images in this layer (i.e., $I^l_{i_{\mathbf{PCANet}}}$) are collected and then the mean of the patches is subtracted from them. Next the patches are vectorized to obtain the final patch matrix which is factorized to obtain the PCA filters:

$$\boldsymbol{Y} \in \mathbb{R}^{k_1 k_2 \times L_1 N \tilde{m}\tilde{n}} \tag{5}$$

$$W^2_{l_{\mathbf{PCANet}}} = mat_{k_1,k_2}(ql(\boldsymbol{Y}\boldsymbol{Y}^T)) \in \mathbb{R}^{k_1 \times k_2}, \quad l = 1, 2, ..., L_2 \tag{6}$$

we then convolve input images in the this layer with the L_2 filters to obtain the output from this layer and proceed with the next layer of the network

$$O^l_{i_{\mathbf{PCANet}}} = I^l_{i_{\mathbf{PCANet}}} * W^2_{l_{\mathbf{PCANet}}}, \quad i = 1, 2, ..., NL_1, \quad l = 1, 2, ..., L_2 \tag{7}$$

The Output Layer In the final output layer, the convolution outputs from the previous layers of **PCANet** are combined to obtain the final feature vector. First, each of the real-valued outputs from Eq. 7 are binarized by using a Heaviside function $H(O^l_{i_{\mathbf{PCANet}}})$ which converts positive entries to 1 otherwise 0. Then the L_2 outputs in the second layer corresponding to the L_1 outputs in the first layer are combined by summing and multiplying with weights. This converts them back into a single image whose pixel value is in the range $[0, 2^{L_2} - 1]$:

$$\mathcal{I}^l_{i_{\mathbf{PCANet}}} = \sum_{l=1}^{L_2} 2^{l-1} H(O^2_{l_{\mathbf{PCANet}}}) \tag{8}$$

Then, each of the L_1 images from Eq. 8 are partitioned into B blocks and then for each block a histogram is computed with 2^{L_2} bins. Finally the histograms from B blocks are concatenated and denoted as $Bhist(\mathcal{I}^l_{i_{\mathbf{PCANet}}})$. This block-wise encoding process encapsulates the L_1 images from Eq. 8 into a feature vector as:

$$f_{i_{\mathbf{PCANet}}} = [Bhist(\mathcal{I}^1_{i_{\mathbf{PCANet}}}), ..., Bhist(\mathcal{I}^{L_1}_{i_{\mathbf{PCANet}}})]^T \in \mathbb{R}^{(2^{L_2})L_1 B} \tag{9}$$

One can now utilize these feature vectors to perform classification.

2.2 Tensor Preliminaries

Tensors are multi-mode arrays, where the modes (also known as orders) of a tensor are analogous to rows and columns (i.e., the two modes) of a matrix. Vectors are defined as first order tensors denoted as $\boldsymbol{x}$, whereas matrices are defined as second order tensors denoted as $\boldsymbol{X}$. Tensors are of order-3 or higher and are denoted as $\mathcal{X}$. Few important tensors operations utilized in this paper are defined below.

Tensor Unfolding: also known as tensor matriziation, is the way of rearranging the elements of an n-mode tensor $\mathcal{X} \in \mathbb{R}^{i_1 \times i_2 \cdots \times i_N}$ as a matrix in chosen mode n denoted as $\mathbf{X}_{(n)} \in \mathbb{R}^{i_n \times j}$, where $j = i_1 \, ... \times i_{n-1} \times i_{n+1} ... \times i_N$.

Tensor to matrix multiplication: The n-mode tensor product of matrix $\mathbf{A} \in \mathbb{R}^{j \times i_n}$ with tensor $\mathcal{X} \in \mathbb{R}^{i_1 \cdots \times i_{m-1} \times i_m \times i_{m+1} \cdots \times i_n}$ is denoted as $\mathcal{X} \times_n \mathbf{A}$, which results in another tensor $\hat{\mathcal{X}}$ of size $\mathbb{R}^{i_1 \times i_2 \times i_{n-1} \times j \times i_{n+1} \cdots \times i_n}$.

Tensor Decomposition Tensor decomposition is a form of generalized of matrix decomposition for factorizing tensors. The algorithm factorizes an n-mode tensor $\mathcal{X} \in \mathbb{R}^{i_1 \times i_2 \dots \times i_n}$ into two subcomponents: 1) $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \dots \times r_n}$ which is a lower dimensional tensor called the *core-tensor* and, 2) $\mathbf{U}^{(n)} \in \mathbb{R}^{r_n \times i_n}$ which are matrix factors associated with each mode of the tensor. Entries in the *core tensor* $\mathcal{G}$ represents the level of interaction between different components. By contrast, entries in the factor matrices $\mathbf{U}^{(n)}$ can be thought as the *principal components* associated with the mode-n. This form of tensor factorization falls under the *Tucker* family of tensor decomposition [5]. The original tensor can be reconstructed by taking product of the *core-tensor* with the factor matrices as:

$$\mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \dots \times_N \mathbf{U}^{(n)} \approx \mathcal{X} \tag{10}$$

The advantages of *Tucker* based factorization methods has already studied in several domains. In computer vision, [25] applied them to the face recognition problem and popularized them as Tensor faces. In data mining, [20] considered the problem of handwritten digits recognition through tensor factorization. In signal processing, [7],[5] considered the problem of brain signal analysis with tucker decomposition.

3 The Tensor Factorization Network (TFNet)

The development of **TFNet** is motivated by the information loss which occurs while vectorizing image-patches in **PCANet**. This transformation is inherent while extracting the *principal components* and incurs the loss of geometric structure present in the data. Furthermore, the vectorization of the data results in high dimensional vectors which generally requires more computational resources. Motivated by the above shortcomings with **PCANet** we propose **TFNet** which is computationally efficient and extracts information while preserving the spatial structure of the data for obtaining feature representation.

3.1 The First Layer

Similar to the first layer in **PCANet**, we collect all overlapping patches of size $k_1 \times k_2$ around each pixel from the image $\{I_i\}$. However, contrary to **PCANet** here the obtained patches forms a 3-mode tensor $\mathcal{X}_i \in \mathbb{R}^{k_1 \times k_2 \times \tilde{m}\tilde{n}}$ instead of a matrix. The mode-1 and mode-2 of this tensor represents the row-space and column-space spanned by the pixels in the image, while the third mode of this tensors represents the total number of image patches and we obtain

$$\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times N\tilde{m}\tilde{n}} \tag{11}$$

as our final tensor. We decompose the tensor using our custom-designed *LoMOI* algorithm presented in Alg. 1 to obtain the factor matrices corresponding to the first two modes, which are later utilized in our tensorial filter generation.

$$[\hat{\mathcal{X}}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}] \leftarrow LoMOI(\mathcal{X}, r_1, r_2) \tag{12}$$

Algorithm 1 Left One Mode Out Orthogonal Iteration (LoMOI)

```
1: Input: $n$-mode tensor $\mathcal{X} \in \mathbb{R}^{i_1, i_2, \cdots, i_n}$; factorization ranks for each mode of the tensor $[r_1...r_{m-1}, r_{m+1}...r_n]$, where $r_k \le i_k \forall\ k \in 1, 2, ..., n$ and $k \ne m$; factorization error-tolerance $\varepsilon$, and Maximum allowable iterations = $Maxiter$, $m$ = mode to discard while factorizing
2: for $i = 1, 2, ..., n$ and $i \ne m$ do
3:     $\mathbf{X}_i \leftarrow$ unfold tensor $\mathcal{X}$ on mode-$i$
4:     $\mathbf{U}^{(i)} \leftarrow r_i$ left singular vectors of $\mathbf{X}_i$        ▷ extract leading $r_i$ matrix factors
5: $\mathcal{G} \leftarrow \mathcal{X} \times_1 (\mathbf{U}^{(1)})^T ... \times_{m-1} (\mathbf{U}^{(m-1)})^T \times_{m+1} (\mathbf{U}^{(m+1)})^T ... \times_n (\mathbf{U}^{(n)})^T$        ▷ Core tensor
6: $\hat{\mathcal{X}} \leftarrow \mathcal{G} \times_1 (\mathbf{U}^{(1)})^T ... \times_{m-1} (\mathbf{U}^{(m-1)})^T \times_{m+1} (\mathbf{U}^{(m+1)})^T \times_N \mathbf{U}^{(n)}$        ▷ reconstructed tensor obtained by multilinear product of the core-tensor with the factor-matrices; Eq. 10.
7: $loss \leftarrow \|\mathcal{X} - \hat{\mathcal{X}}\|$        ▷ decomposition loss
8: $count \leftarrow 0$
9: while $[(loss \ge \varepsilon)\ Or\ (Maxiter \le count)]$ do        ▷ loop until convergence
10:     for $i = 1, 2, ..., n$ and $i \ne m$ do
11:         $\mathcal{Y} \leftarrow \mathcal{X} \times_1 (\mathbf{U}^{(1)})^T ... \times_{(i-1)} (\mathbf{U}^{(i-1)})^T \times_{(i+1)} (\mathbf{U}^{(i+1)})^T ... \times_n (\mathbf{U}^{(n)})^T$        ▷ obtain the variance in mode-$i$
12:         $\mathbf{Y}_i \leftarrow$ unfold tensor $\mathcal{Y}$ on mode-$i$
13:         $\mathbf{U}^{(i)} \leftarrow \mathbf{r}_i$ left singular vectors of $\mathbf{Y}_i$
14:     $\mathcal{G} \leftarrow \mathcal{X} \times_1 (\mathbf{U}^{(1)})^T ... \times_{(m-1)} (\mathbf{U}^{(m-1)})^T \times_{(m+1)} (\mathbf{U}^{(m+1)})^T ... \times_n (\mathbf{U}^{(n)})^T$
15:     $\hat{\mathcal{X}} \leftarrow \mathcal{G} \times_1 \mathbf{U}^{(1)} ... \times_{(m-1)} (\mathbf{U}^{(m-1)})^T \times_{(m+1)} (\mathbf{U}^{(m+1)})^T ... \times_n \mathbf{U}^{(n)}$
16:     $loss \leftarrow \|\mathcal{X} - \hat{\mathcal{X}}\|$
17:     $count \leftarrow count + 1$
18: Output: $\hat{\mathcal{X}}$ the reconstructed tensor and $[\mathbf{U}^{(1)}...\mathbf{U}^{(m-1)}, \mathbf{U}^{(m+1)}...\mathbf{U}^{(n)}]$ the factor matrices
```

where $\hat{\mathcal{X}} \in \mathbb{R}^{r_1 \times r_2 \times N\tilde{m}\tilde{n}}$, $\mathbf{U}^{(1)} \in \mathbb{R}^{k_1 \times r_1}$, and $\mathbf{U}^{(2)} \in \mathbb{R}^{k_2 \times r_2}$. We discard obtaining the matrix factors from mode-3 i.e. $\mathbf{U}^{(3)}$ of the tensor as the *mode*-3 matricization of tensor $\mathcal{X}$ denoted as $\boldsymbol{X}_3 \in \mathbb{R}^{N\tilde{m}\tilde{n} \times k_1 \times k_2}$ is equivalent to the transpose of the patches matrix $\boldsymbol{X}$ defined in Eq. 1 which is not decomposed in the **PCANet** while obtaining their filters. A total of $L_1 = r_1 \times r_2$ filters (equivalent to the number of filters in the **PCANet**)are obtained from the factor matrices $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$ as:

$$W^1_{l_{\mathbf{TFNet}}} = \mathbf{U}^{(1)}_{(:,i)} \otimes \mathbf{U}^{(2)}_{(:,j)} \in \mathbb{R}^{k_1 \times k_2},\ i = 1...r_1,\ j = 1...r_2,\ l = 1...L_1 \tag{13}$$

where '$\otimes$' is the outer product between two vectors and $\mathbf{U}^{(m)}_{(:,i)}$ represents 'i^{th}' column of the 'm^{th}' factor matrix. Our filters obtained in Eq. 13 does not require any explicit reshaping as the operation *outer*-product between two vectors naturally results in a matrix. Hence, we can straightforwardly convolve our tensorial filters with the input images to obtain output from the first stage as:

$$I^l_{i_{\mathbf{TFNet}}} = I_i * W^1_{l_{\mathbf{TFNet}}}, \quad i = 1, 2, ..., N, \quad l = 1, 2, ..., L_1 \tag{14}$$

When the data are *RGB*-images, every patch $\mathbf{x}_{i,j}$ extracted from image is a 3-order tensor $\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times 3}$ (*RowPixels*$\times$*ColPixels*$\times$Color). After collecting image patches from the training images, we obtain a 4-mode tensor

$$\mathcal{X} \in \mathbb{R}^{k_1 \times k_2 \times 3 \times N\tilde{m}\tilde{n}} \tag{15}$$

decomposing the above tensor to obtain factor matrices are as follows:

$$[\hat{\mathcal{X}}, \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \mathbf{U}^{(3)}] \leftarrow LoMOI(\mathcal{X}, r_1, r_2, r_3) \tag{16}$$

$$W^1_{l_{\mathbf{TFNet}}} = U^{(1)}_{(:,i)} \otimes U^{(2)}_{(:,j)} \otimes U^{(3)}_{(:,k)} \quad \forall i \in 1...r_1,\ j \in 1...r_2,\ k \in 1...r_3 \tag{17}$$

3.2 The Second Layer

Similar to the first layer, we extract overlapping patches from the input images and then subtract the patch mean from the patches and build a 3-mode tensor denoted as $\mathcal{Y}_i$ and then decompose it to obtain our factor matrices as:

$$[\hat{\mathcal{Y}}, \mathbf{V}^{(1)}, \mathbf{V}^{(2)}] \leftarrow LoMOI(\mathcal{Y}, r_1, r_2) \tag{18}$$

where, $\hat{\mathcal{Y}} \in \mathbb{R}^{r_1 \times r_2 \times NL_1 \tilde{m}\tilde{n}}$, $\mathbf{V}^{(1)} \in \mathbb{R}^{k_1 \times r_1}$, and $\mathbf{V}^{(2)} \in \mathbb{R}^{k_2 \times r_2}$. We then generate our tensorial filters from the matrix factors of the first two modes as:

$$W^2_{l_{\mathbf{TFNet}}} = \mathbf{V}^{(1)}_{(:,i)} \otimes \mathbf{V}^{(2)}_{(:,j)} \in \mathbb{R}^{k_1 \times k_2}, \quad i = 1..r_1, \; j = 1..,r_2, \; l = 1...L_2 \tag{19}$$

Now, each of the L_1 input images in the first layer are convolved with tensorial filters obtained in the second layer as:

$$O^l_{i_{\mathbf{TFNet}}} = I^l_{i_{\mathbf{TFNet}}} * W^2_{l_{\mathbf{TFNet}}}, \quad l = 1, 2, ..., L_2 \tag{20}$$

The number of output images obtained from this operation is equal to $L_1 \times L_2$. We now utilize the output layer of **PCANet** to obtain our final feature vectors

$$\mathcal{I}^l_{i_{\mathbf{TDNet}}} = \sum_{l=1}^{L_2} 2^{l-1} H(O^2_{l_{\mathbf{TensorNet}}}) \tag{21}$$

$$f_{i_{\mathbf{TFNet}}} = [Bhist(\mathcal{I}^1_{i_{\mathbf{TFNet}}}), ..., Bhist(\mathcal{I}^{L_1}_{i_{\mathbf{TFNet}}})]^T \in \mathbb{R}^{(2^{L_2})L_1 B} \tag{22}$$

the features vectors obtained in Eq. 22 can now be utilized for classification.

4 The Hybrid Network (HybridNet)

The **PCANet** extracts features from the amalgamated view of the data whereas the **TFNet** extracts features from the minutiae view of the data. Our hypothesis is that both these views are important as they conceal distinct representations of the data and integrating feature representations from these views can enhance the performance of classification systems. Motivated by the above, we propose the **HybridNet** which integrates the filter learning process from the minutiae view and the amalgamated view. We explain the feature extraction procedure in **HybridNet** with the help of Fig. 1.

4.1 The First Layer

The first layer in **HybridNet** consists of image-patches expressed as both as tensors and matrices. In this way, the first layer in **HybridNet** perceives more diverse information from different views of the data while learning its filters. Further, the filters for the tensorised patches were obtained via *LoMOI*, while the filters for the patch-matrices are obtained via the principal components. Since the first layer of **HybridNet** consists of hybrid filters, the output from this layer is obtained by convolving input images with: a) the PCA-filters and b) the tensorial-filters. This injects more diversity to the output from the first

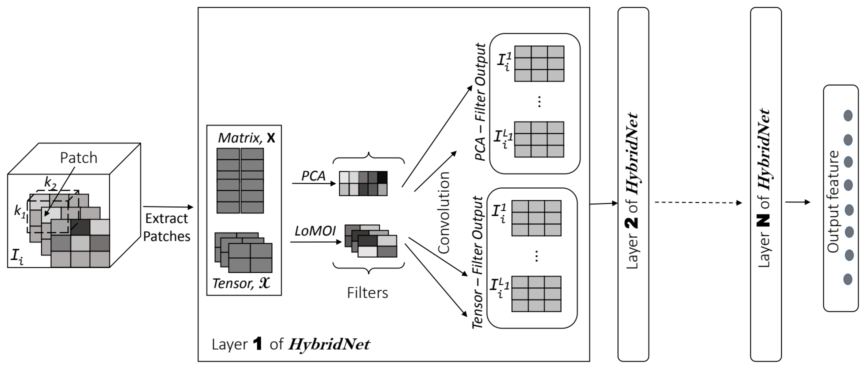


Fig. 1. The Proposed Hybrid Network

layer in **HybridNet** or equivalently to the input of the succeeding layer. Consequently,the covariance matrix in the **HybridNet** captures more variability than that of the covariance matrix obtained in either of the **PCANet** or **TFNet**. Therefore, *the hybrid filters captures more variability in the data which leads to better disentangled representations.* This results in superior performance from the features obtained with the **HybridNet**. Since we obtain L_1-PCA filters from the patch-matrices and L_1-tensor filters from the patch-tensors, a total of $2 \times L_1$ output images are obtained after the convolution of images with these filters.

4.2 The Second Layer

In the second layer of **HybridNet** the filters are learned with the hybrid data obtained from the first layer. Moreover, the output images from this layer are obtained by a) convolving the L_1 images corresponding to the output from the PCA-filters in the first layer with the -filters in the second layer, and b) convolving the L_1 images corresponding to the output from the tensor-filters in the first layer with the tensor-filters in the second layer. The number of output images obtained from the second layer produces a total of $2 \times L_1 \times L_2$ outputs. Finally, the outputs from the second layer of **HybridNet** are processed with the same **Output Layer** as in **PCANet** and **TFNet**, to obtain hybrid features.

5 Experiments and Results

5.1 Experimental Setup

In our experiments, we utilized two-layer architecture for each of the networks while learning their filters and utilized the output layer of the **PCANet** to obtain feature vectors from the networks. Since the number of filters in the first and the second layer are L_1 and L_2 respectively. The feature-length obtained with the **PCANet** and the **TFNet** are equal to $BL_1 2^{L_2}$, while the feature-length with **HybridNet** is equal to $2BL_1 2^{L_2}$; where B is the number of image-blocks obtained while calculating the histograms. Throughout our experiments, we utilized *Linear SVM* [9] as the classifier while performing classification.

5.2 Datasets

We utilize the following datasets and hyper-parameters in our experiments:

1 MNIST variations [16], which consists of 28×28 gray scale handwritten digits with controlled factors of variations such as background noise, rotations, background-images etc. Each variation contains $10K$ training and $50K$ testing images. We set, $L_1 = 9$, L_2 =8, $k_1 = k_2 = 7$, with block size = 7×7.[3]

2 CUReT texture dataset [24], consisting of 61 classes of image textures where each class has images of the same material with different pose, illumination conditions, specularity, shadowing, and surface normals. A subset of 92 cropped images were taken from each category as in [24],[4]. Following the standard procedure in [4], we randomly split the data into train and test set with a split ratio of 50% and classification results are averaged over 10 trails. We set, $L_1 = 9$, L_2 =8, $k_1 = k_2 = 5$, and the block size = 50×50.

3 CIFAR-10 [15] consisting of $50K$ training and $10K$ testing images distributed among 10 classes. The *RGB* images are of dimensions 32×32 and vary significantly in object position, scale, colors, and textures. We vary L_1 as 9 and 27, keep L_2 =8; whereas the patch sizes $k_1 = k_2$ are varied as 5, 7, and 9. Following [4] we also applied spatial pyramid pooling (SPP) [10] to the output layer of **PCANet**, while the block size = 8×8. We additionally applied PCA to reduce the dimension of each pooled feature to 100.[4]

5.3 Results and Discussions

Classification errors obtained on handwritten digits variations and texture recognition datasets are reported in Table 1. The hybrid features obtained with our proposed **HybridNet** outperforms the state of the art results on five out of seven MNIST variations dataset. For texture classification the hybrid features achieves the lowest error among the three networks, however on this dataset they perform slightly lower than the state of the art. For object recognition the classification errors obtained on CIFAR-10 are reported in Table 2, again the hybrid features achieves the lowest error among the three networks studied in this paper. However on this dataset the performance of hybrid features is **14.57**% lower than state of the art - **NIN** i.e. **10.41**% (without data augmentation) [17]. This is because a) **NIN** is comparatively deeper and more importantly b) **NIN** performs supervised nonlinear feature extraction whereas the **HybridNet**

[3] Overlapping regions between the blocks is equal to half of the block size.

[4] Results does not vary significantly on increasing the projection dimensions.

Table 1. Classification Error (%) obtained on MNIST variations and CuReT datasets

Methods	baisc	rot	bg-rnd	bg-im	bg-im-rot	rect	cnvx
CAE-2 [19]	2.48	9.66	10.90	15.50	45.23	21.54	-
TIRBM [22]	-	**4.20**	-	-	35.50	-	-
PGBM [23]	-	-	6.08	12.25	36.76	**8.02**	-
ScatNet [3]	1.27	7.48	12.30	18.40	50.48	15.94	6.50
PCANet	1.07	6.88	6.08	11.16	37.28	13.59	4.15
TFNet	1.07	7.15	6.49	11.44	38.26	16.87	4.98
HybridNet	**1.01**	6.47	**5.46**	**10.08**	**35.15**	12.91	**3.55**

(a) MNIST Variations Datasets

Methods	Error (%)
Textons [12]	1.50
BIF [8]	1.40
Histogram [2]	1.00
ScatNet [3]	**0.20**
PCANet	0.84
TFNet	0.96
HybridNet	0.81

(b) CuReT Dataset

Table 2. Classification Error (%) obtained on CIFAR-10 without Data Augmentation

Parameters				**PCANet**	**TFNet**	**HybridNet**
L_1	L_2	k_1	k_2	Error (%)	Error (%)	Error (%)
8	8	5	5	34.80	32.57	**31.39**
27	8	5	5	26.43	29.25	**24.98**
8	8	7	7	39.92	37.19	**35.24**
27	8	7	7	30.08	32.57	**28.53**
8	8	9	9	43.91	39.65	**38.04**
27	8	9	9	33.94	34.79	**31.36**

(a) Performance of **PCANet**, **TFNet**, and the **HybridNet** by varying hyperparamters

Methods	Error (%)
Tiled CNN [18]	26.90
K-means [6] (1600 dim.)	22.10
Conv. Maxout [21]	11.68
NIN [17]	**10.41**
PCANet	26.43
TFNet	29.25
HybridNet	24.98

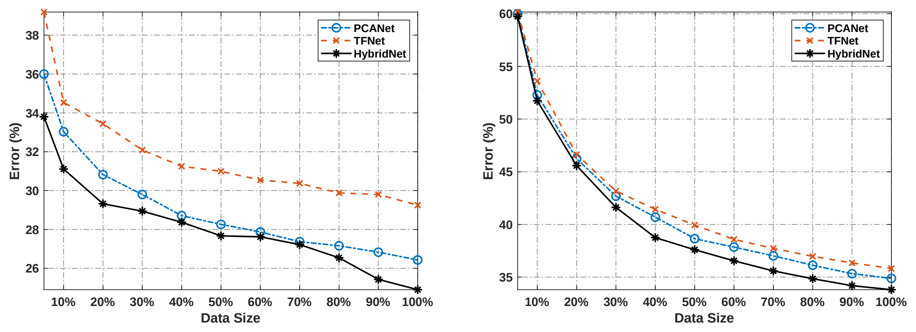
(b) Benchmark comparisons

performs an unsupervised linear feature extraction. However, the classification errors obtained with **HybridNet** are still promising and can be enhanced with more layers and non-linear operations. Besides, the above we have also evaluated the performance of **PCANet**, **TFNet** and the **HybridNet** by varying training data size on CIFAR and MNIST variation[5] dataset, shown in Fig. 2. The above experiments validates our claim of improving the classification accuracy by integrating the information from two views of the data.

6 Conclusion and Future Work

In this paper we have first introduced **Tensor Factorization Networks** which preserves the spatial structure of the data while extracting features from the minutiae view of the data. Since both the amalgamated view and the minutiae view are individually insufficient feature representations, we propose a hybrid parsimonious network called the **Hybrid Network**. The **Hybrid Network** simultaneously learns its hybrid convolution filters by integrating the two views

[5] Random background, images, and rotation is utilized with block size = 4×4.



(a) CIFAR 10 (b) MNIST bg-img-rot

Fig. 2. Performance Comparison by varying size of the training data

of the data. The features obtained through hybrid filters enhances the classification performance on several benchmark datasets. The experiments validates the advantages of obtaining superior feature representation by integrating the two views of the data. In future, we plan to study the effects of unaligned images during the filter learning phase in the **Hybrid Network**.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1798–1828 (2013)
2. Broadhurst, R.E.: Statistical estimation of histogram variation for texture classification. In: Proc. Intl. Workshop on Texture Analysis and Synthesis. pp. 25–30 (2005)
3. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1872–1886 (2013)
4. Chan, T.H., Jia, K., Gao, S., Lu, J., Zeng, Z., Ma, Y.: Pcanet: a simple deep learning baseline for image classification? IEEE Transactions on Image Processing **24**(12), 5017–5032 (2015)
5. Cichocki, A., Mandic, D., De Lathauwer, L., Zhou, G., Zhao, Q., Caiafa, C., Phan, H.A.: Tensor decompositions for signal processing applications: from two-way to multiway component analysis. IEEE Signal Processing Magazine **32**(2), 145–163 (2015)
6. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. pp. 215–223 (2011)
7. Cong, F., Lin, Q.H., Kuang, L.D., Gong, X.F., Astikainen, P., Ristaniemi, T.: Tensor decomposition of eeg signals: a brief review. Journal of Neuroscience Methods **248**, 59–69 (2015)
8. Crosier, M., Griffin, L.D.: Using basic image features for texture classification. International Journal of Computer Vision **88**(3), 447–460 (2010)
9. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: Liblinear: a library for large linear classification. Journal of Machine Learning Research **9**(Aug), 1871–1874 (2008)

10. Grauman, K., Darrell, T.: The pyramid match kernel: discriminative classification with sets of image features. In: Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. vol. 2, pp. 1458–1465. IEEE (2005)
11. Han, S., Mao, H., Dally, W.J.: Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. ICLR (2016)
12. Hayman, E., Caputo, B., Fritz, M., Eklundh, J.O.: On the significance of real-world conditions for material classification. In: European Conference on Computer Vision. pp. 253–266. Springer (2004)
13. Huang, J., Yuan, C.: Fanet: factor analysis neural network. In: International Conference on Neural Information Processing. pp. 172–181. Springer (2015)
14. Kossaifi, J., Khanna, A., Lipton, Z., Furlanello, T., Anandkumar, A.: Tensor contraction layers for parsimonious deep nets. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on. pp. 1940–1946. IEEE (2017)
15. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto (2009)
16. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th International Conference on Machine Learning. pp. 473–480. ACM (2007)
17. Lin, M., Chen, Q., Yan, S.: Network in network. ICLR (2013)
18. Ngiam, J., Chen, Z., Chia, D., Koh, P.W., Le, Q.V., Ng, A.Y.: Tiled convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1279–1287 (2010)
19. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. pp. 833–840. Omnipress (2011)
20. Savas, B., Eldén, L.: Handwritten digit classification using higher order singular value decomposition. Pattern Recognition **40**(3), 993–1003 (2007)
21. Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems. pp. 2951–2959 (2012)
22. Sohn, K., Lee, H.: Learning invariant representations with local transformations. In: Proceedings of the 29th International Coference on International Conference on Machine Learning. pp. 1339–1346 (2012)
23. Sohn, K., Zhou, G., Lee, C., Lee, H.: Learning and selecting features jointly with point-wise gated boltzmann machines. In: International Conference on Machine Learning. pp. 217–225 (2013)
24. Varma, M., Zisserman, A.: A statistical approach to material classification using image patch exemplars. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(11), 2032–2047 (2009)
25. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear image analysis for facial recognition. In: Pattern Recognition, 2002. Proceedings. 16th International Conference on. vol. 2, pp. 511–514. IEEE (2002)
26. Zheng, L., Yang, Y., Tian, Q.: Sift meets CNN: a decade survey of instance retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(5), 1224–1244 (2018)