ARL-TR-9172 ● Apr 2021

On Data-Driven Saak Transform: Theory and Applications

by Suya You

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9172 ● Apr 2021

On Data-Driven Saak Transform: Theory and Applications

Suya You
Computational and Information Sciences Directorate,
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
April 2021	Technical Report	1 October 2018–30 September 2020

4. TITLE AND SUBTITLE
On Data-Driven Saak Transform: Theory and Applications

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Suya You

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD- RLC-CI
2800 Powder Mill Road, Adelphi, MD 20783-1138

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9172

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This US Army Combat Capabilities Development Command Army Research Laboratory's External Collaboration Initiative is a joint effort with the University of Southern California. Combining the team members' experience in machine learning, signal analysis, computer vision, and perceptual communication and computation led to insights on machine-learning mechanism and the impetus to develop an innovative theory and mathematical framework, the Subspace Approximation with Augmented Kernels (Saak) transform for deep neural-network architecture. The Saak transform is an entirely new signal theory based on insightful interpretations of the deep-learning mechanism. We conducted the fundamental research on Saak transform theory and its two important extensions: Subspace Approximation with Adjusted Bias transform and Successive Subspace Learning. We unified them under a common framework of "interpretable subspace learning". We applied developed theories to several militarily relevant tasks and scenarios including image classification, recognition, defense against adversarial attacks, and biometric facial-data processing.

15. SUBJECT TERMS
machine learning, deep neural network, subspace learning, computer vision, object recognition

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Suya You
Unclassified	Unclassified	Unclassified	UU	49	19b. TELEPHONE NUMBER (Include area code) (310) 448-5395

Contents

List of Figures

List of Tables

Executive Summary

This US Army Combat Capabilities Development Command Army Research Laboratory's External Collaboration Initiative is a joint effort with the University of Southern California. Combining the team members' experience in machine learning, signal analysis, computer vision, and perceptual communication and computation led to insights on machine-learning mechanism and the impetus to develop an innovative theory and mathematical framework, the Subspace Approximation with Augmented Kernels (Saak) transform for deep neural-network architecture. The Saak transform is an entirely new signal theory based on insightful interpretations of the deep-learning mechanism. We conducted the fundamental research on Saak transform theory and its two important extensions: Subspace Approximation with Adjusted Bias transform and Successive Subspace Learning. We unified them under a common framework of "interpretable subspace learning". We applied developed theories to several militarily relevant tasks and scenarios including image classification, recognition, defense against adversarial attacks, and biometric facial-data processing.

This research is *Soldier oriented*. We expect Soldiers to carry intelligent yet extremely low size–weight–power vision-based devices on the battlefield. Today's machine-learning solution is too sensitive to a specific data environment. When data are acquired in a different setting, the network needs to be retrained, which is difficult to conduct in an embedded system. The Saak transform has significantly lower complexity and it is very suitable for portable devices. Furthermore, the development of the joint data compression and learning technique can create a single bit stream that includes the high-level semantic interpretation as well as the low-level visual information. This can lead to game-change solutions with far-reaching applications to the future Army.

1. Introduction

The research mainly focuses on the deep neural network (DNN) since DNN represents the vital technology behind many current artificial intelligence (AI) breakthroughs.

Solutions based on DNNs have been dominant in many fields that involve the learning, modeling, and processing of complex data. Yet, the DNN solution has its own weaknesses in terms of robustness, scalability, and portability. A DNN is built on the choice of end-to-end network architecture and its optimization requires rich labeled data. Furthermore, DNN working principle remains mysterious. The biggest challenge lies in the fact that today's neural-network-based approach works like a black box and there is no proven theory behind it. Although attempts have been made to give intuitions to the underlying mechanism, there is no formal theory. As the complexity of neural-network architectures goes higher, their behavior is mathematically intractable. Therefore, there is an urgent need to understand what is behind the amazing performance of neural networks on one hand and, on the other hand, build an alternative paradigm that achieves similar or better performance with solid theoretical support.

This research is to develop the Subspace Approximation with Augmented Kernels (Saak) transform as an innovative mathematical framework for deep neural-network architecture that could lead to revolutionary computational algorithms. The Saak transform is an entirely new signal theory based on insightful interpretations of the deep-learning (DL) mechanism. The Saak and inverse Saak transforms provide signal analysis and synthesis tools, respectively. The core of the Saak transform consists of three steps: 1) building optimal linear-subspace approximation using the covariance function of input vectors, 2) augmenting each transform kernel with its negative, and 3) applying the rectified linear unit to the transform output. The integration of Steps 2 and 3 removes the rectification loss, resolves the sign-confusion problem, and allows a straightforward implementation of the inverse Saak transform. The mechanism of the Saak transform is based on the properties of input data; thus, it can be computed in a feed-forward, unsupervised manner (no labeled data needed).

Multiple Saak transforms can be cascaded to transform images of a larger size from the spatial domain to a joint spatial–spectral domain. The applications of the Saak transform demonstrate higher recognition accuracy comparable to state of the art. Furthermore, the Saak transform has several additional advantages including lower complexity, weaker supervision, more robustness to noise, and better scalability against different settings. A Saak-transform-based solution offers the potential to

radically change the way of data learning, representation, and processing with far-reaching applications to the future Army.

The overall objectives of this research are to develop of Saak transform as 1) a new DL paradigm that lends itself to white box access and interpretation, and 2) a new methodology for Army-critical applications, especially for scene perception, representation, and processing with emphasis at the tactical edge.

We have conducted the fundamental research on the Saak transform theory and its two important extensions: Subspace Approximation with Adjusted Bias (Saab) transform and Successive Subspace Learning (SSL). We unified them under a common framework, "interpretable subspace learning". We applied developed theories to several militarily relevant tasks and scenarios including image classification, recognition, defense against adversarial attacks, and biometric facial-data processing. The following sections will detail these activities and achievements.

2. Research—Theory

In this section, we describe the fundamental research on Saak transform theory and its two important extensions, Saab transform and SSL.

2.1 Saak Transform

The Saak transform[1] defines a mapping from a real-valued function defined on a 3-D cuboid consisting of spatial and spectral dimensions to a 1-D rectified spectral vector. It consists of two main ideas: subspace approximation and kernel augmentation. For the former, we build the optimal linear subspace approximation to the original signal space via principal component analysis (PCA) or the truncated Karhunen–Loeve Transform (KLT). For the latter, we augment each transform kernel with its negative and apply the rectified linear unit (ReLU) to the transform output. This is equivalent to the sign-to-position (S/P) format conversion. Saak transform is a process of converting the spatial variation to the spectral variation and the inverse Saak transform is a process of converting the spectral variation to the spatial variation.

2.1.1 Forward Saak Transform

This process comprises several steps:

1) Kernel Selection and Augmentation

Collect a representative set of input samples f and determine its KLT basis functions, which are denoted by b_k, k = 1, 2… N. The DC kernel is denoted by a_0. The remaining 2(N–1) AC kernels are obtained using the augmentation rule:

$$a_{2k\_1} = b_k,\ a_{2k} = -b_k, \qquad k = 1, 2, \dots, N\text{-}1. \tag{1}$$

2) Projection onto the Augmented Kernel Set

Project input f on the augmented kernel set from Step 1:

$$p_k = a^T_k\, f \tag{2}$$

and

$$p = (p_0, p_1, \dots, p_{2(N-1)})^T \tag{3}$$

is the projection vector of input f.

3) Apply the ReLU to the projection vector except the first element to yield the final output:

$$g = (g_0, g_1, \dots, g_{2(N-1)})^T \tag{4}$$

where $g_0 = p_0$ and

$$g_{2k-1} = p_{2k\,-1} \text{ and } g_{2k} = 0 \qquad \text{if } p_{2k\,-\,1} > 0 \tag{5}$$

$$g_{2k-1} = 0 \text{ and } g_{2k} = p_{2k} \qquad \text{if } p_{2k} > 0 \tag{6}$$

for k = 1, 2, … , N–1.

The kernel-augmentation scheme described in this passage is a way to motivate the Saak transform. In practical implementation, we can offer another view to the cascade of kernel augmentation and ReLU. As projection values, p_k on KLT basis functions b_k can be positive or negative. It is called the sign format of the projection output and denoted by

$$g_s = (g_{s\,0}, g_{s\,1}, \dots g_{s\,n-1})^T \tag{7}$$

where

$$g_{s\,k} = b^T_k \tag{8}$$

In contrast, the position of each AC element in Eq. 4 is split into two consecutive positions. Its magnitude is recorded in the first and second positions, respectively, depending on whether it is positive or negative. This is called the position format of the projected output. The cascade of kernel augmentation and ReLU is equivalent to the S/P format conversion as shown in Fig. 1. Note that we also demand the P/S format conversion for the inverse Saak transform.

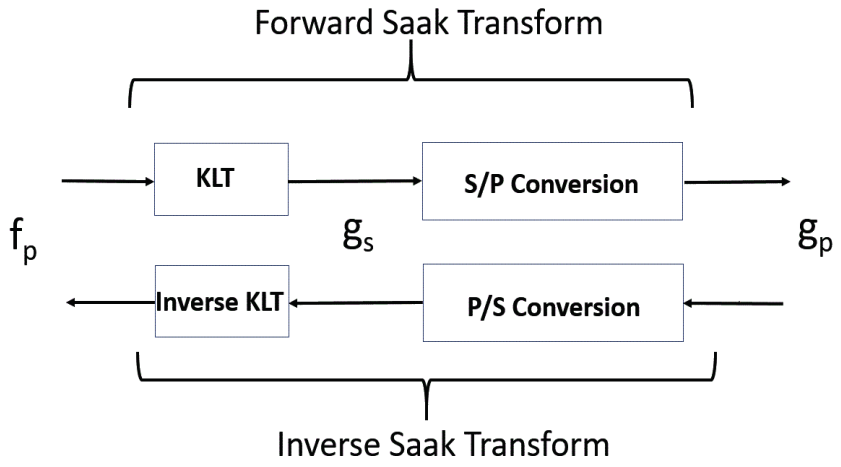

Fig. 1 Block diagram of forward and inverse Saak transforms, where f_p, g_s, and g_p are (respectively) input in position format, output in sign format, and output in position format

2.1.2 Inverse Saak Transform

The inverse Saak transform can be written as

$$f_p = \sum_{k=0}^{N-1} g_{k\,s} b_k \tag{9}$$

where b_k is a basis function. Given position format g_p, we perform P/S format conversion to get sign format g_s and then feed it to the inverse KLT.

2.1.3 Multistage Saak Transforms

Multistage Saak transforms are developed to transform images of a larger size. To begin with, we decompose an input into four quadrants recursively to form a quad-tree structure with its root being the full image and its leaf being a small patch of size 2 × 2. Then, we conduct the Saak transform by merging four child nodes into one parent node stage by stage and from the leaf to the root. The whole process terminates when we reach the last stage (or the root of the tree) that has a spatial dimension of 1 × 1. The signed KLT coefficients in each stage are called the Saak coefficients that can serve as discriminant features of the input image. Multistage Saak transforms provide a family of spatial–spectral representations (Fig. 2). They are powerful representations to be used in many applications.

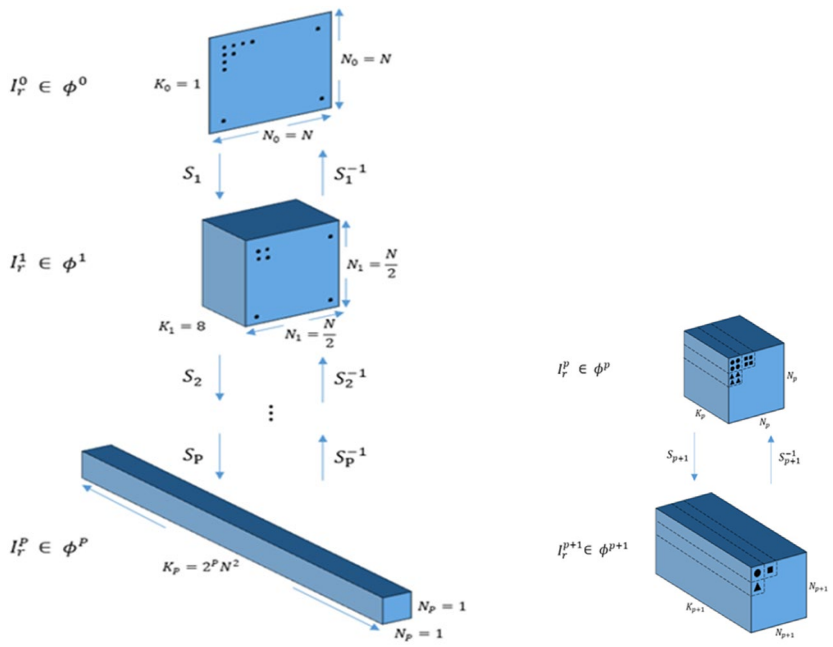

Fig. 2 Illustration of forward and inverse multistage Saak transforms, where S_p and $S^{-1}{}_p$ are the forward and inverse Saak transforms between stages (p – 1) and p, respectively

2.1.4 Saak Transform Properties

The Saak transform provides a mathematical framework for interpretable DNN systems. It is rooted in probability theory and statistics and has very attractive properties:

- The Saak transform is a brand new signal-processing tool. Its orthonormal transform kernels facilitate the computation in both forward and inverse transforms. It is a complete data-driven transform in which neither data labels nor back-propagation is demanded in kernels (weights) determination.
- Saak features (coefficients) at the final stage are almost uncorrelated, and Saak-feature distributions are approximated as the Gaussian Mixture Model, which allows employment of any linear classifier.
- Saak features have strong power of energy compaction, which means we can represent data using few features for data compression and synthesis.
- The Saak transform concentrates image energy well in a low-frequency spectral component. This is a useful property for many tasks such as denoise and defense against adversarial attacks.

- Saak features demonstrate strong discriminability, which is key for visual-perception tasks such as image classification, recognition, and detection.
- The computational requirement of the Saak transform is much lower in comparison with DNN. There is no back-propagation. The Saak transform allows parallel processing for further computational speedup.

2.2 Saab Transform

To resolve the sign-confusion problem, the Saak transform augments a transform kernel with its negative, leading to a large number of transform kernels in total. Any input vector will have a positive/negative correlation pair with a kernel pair. When a correlation is followed by ReLU, one of the two will go through while the other will be blocked. In other words, the Saak transform splits positive/negative correlations into two positive/negative channels. To resolve the sign-confusion problem, it pays the price of doubled spectral dimensions. In this section, we introduce another extension called the Saab transform. The Saab transform can address the sign-confusion problem and avoid the spectral dimension-doubling problem at the same time.

2.2.1 Computational Neuron

The computational neuron serves as the basic building element of neural networks.[2,3] As shown in Fig. 3, it consists of two stages: 1) affine computation and 2) nonlinear activation. The input is an N-dimensional random vector $\mathbf{x}$. The $\mathbf{k}^{th}$ neuron has N filter weights that can be expressed in vector form as $\mathbf{a_k}$ and one bias term $\mathbf{b_k}$. The affine computation is

$$y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k, \quad k = 0, 1, \cdots, K-1 \tag{10}$$

With ReLU nonlinear activation function, the output after ReLU can be written as

$$z_k = \phi(y_k) = \max(0, y_k) \tag{11}$$

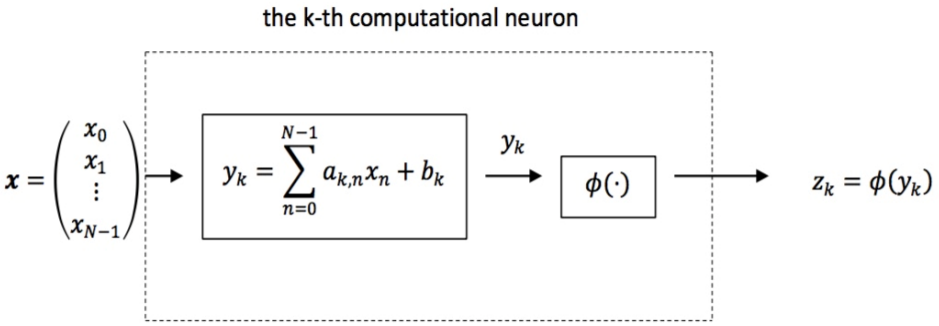

Fig. 3 Illustration of a computational neuron

2.2.2 Saab Transform and Bias Selection

The Saab transform is nothing but a specific way in selecting anchor vector a_k and bias term b_k. We first set $b_k = 0$ and divide anchor vectors into DC and AC categories. The terms "DC" and "AC" are borrowed from circuit theory and denote the "direct current" and the "alternating current", respectively.[4] Based on the two categories of anchor vectors, we decompose the input vector space, $S = R^N$, into the direct sum of two subspaces:

$$\mathcal{S} = \mathcal{S}_{DC} \oplus \mathcal{S}_{AC} \tag{12}$$

where S_{DC} is the subspace spanned by the DC anchor and S_{AC} is the subspace spanned by the AC anchors. They are called the DC and AC subspaces accordingly. For any vector, we can project it to S_{DC} to get its DC component and get its AC subspace component S_{AC} that is the orthogonal complement to S_{DC} in S.

Each bias term, b_k, in Eq. 10, provides one extra degree of freedom per neuron for the end-to-end penalty minimization in the back-propagation design. The bias term is leveraged to overcome the sign-confusion problem in the Saab transform. We impose two constraints on the bias terms.

One is "positive response constraint" to keep neuron output y_k positive. It is easy to derive that this constraint means the selected bias value is always larger than the input values.

$$y_k = \sum_{n=0}^{N-1} a_{k,n} x_n + b_k = \mathbf{a}_k^T \mathbf{x} + b_k \geqslant 0, \tag{13}$$

The second constraint is to make all of the bias terms equal to keep the thing simple.

$$b_0 = b_1 = \cdots = b_{K-1} = b \tag{14}$$

We can see the first constraint is actually equivalent to the ReLU in DNN. Therefore, the Saab transform removes the nonlinearity introduced by the nonlinear activation function. Reducing the rectified loss also makes the feed-forward learning possible without needing back-propagation.[5,6] So, the Saab transform is another way to control rectification loss without increased dimension. It is a variant of Saak transform and keeps all of the nice properties of the Saak transform.[1]

2.3 SSL

As mentioned previously, multilayers or multistage process is one of the success keys for DNN. Deeper is better because the network covers a larger receptive field and information grows from local to global.[7] The same idea already is used for Saak or Saab transforms that cascade multiple transform units to successively transfer the learned information.

Here we extend the situation where subspace is not fixed but growing from one stage to the other. For example, we can take the union of an image pixel and its eight nearest neighbors to form an input space in the first stage. Afterward, we enlarge the neighborhood of the center pixel by a factor in the second stage. Clearly, the first input space is a proper subset of the second input space. By generalizing it to multiple stages, it gives rise to a "successive subspace growing" process. This process exists naturally in the neural-network architectures, where the response in a deeper layer has a larger receptive field. It corresponds to an input of a larger neighborhood. Instead of analyzing these embedded spaces independently, it is advantageous to find a representation of a larger neighborhood using those of its constituent neighborhoods of smaller sizes in computation and storage efficiency. Second, special attention should be paid to the cascade interface of two consecutive stages, as is elaborated on in Section 2.3.1. We named the new extension the SSL.

2.3.1 SSL Principle

Retrospectively, the work in Saak and Saab transforms[1,6] has contained the SSL design idea although the SSL term was not explicitly introduced therein. SSL can be applied, but not limited to, parameters design of any DNN. To illustrate the flexibility and generalizability of SSL, we use an SSL-based machine-learning (ML) system called the PixelHop method.[8–10] The block diagram of the PixelHop system deviates from the standard DNN architecture completely since it is not a network any longer. The word "Hop" is borrowed from graph theory. For a target node in a graph, its immediate neighboring nodes connected by an edge are called its one-hop neighbors. Its neighboring nodes connected to the target node through

n consecutive edges via the shortest path are the n-Hop neighbors. The PixelHop method begins with a localized region; namely, a single pixel. It is called the 0-Hop input. We concatenate the attributes of a pixel and attributes of its one-Hop neighbors to form a one-Hop neighborhood. We can keep enlarging the input by including larger neighborhood regions. This idea applies to structured data (e.g., images) as well as unstructured data (e.g., 3-D point-cloud sets).

Technically, SSL contains four key ingredients: 1) successive near-to-far neighborhood expansion; 2) unsupervised dimension reduction via subspace approximation; 3) supervised dimension reduction via label-assisted regression (LAG); and 4) feature concatenation and decision-making. Fig. 4 illustrates the block diagram of the SSL-based PixelHop method. Its input can be gray or color images. They are fed into a sequence of I PixelHop units in cascade to obtain the attributes of the i^{th} PixelHop unit, as will be shown in Module 1. The attributes in spatial locations of each PixelHop unit are aggregated in multiple forms and then fed into the LAG unit for further dimension reduction to generate M attributes per unit, as in Module 2. Finally, these attributes are concatenated to form the ultimate feature vector of dimension $M \times I$ for image classification, as in Module 3.

Module 1: A sequence of PixelHop units in cascade, to compute attributes of near-to-far neighborhoods of selected pixels through I PixelHop units.

Module 2: Aggregation and supervised dimension reduction via the LAG unit, to aggregate, compress, and extract a diversified set of features at each PixelHop unit.

Module 3: Feature concatenation across all PixelHop units and classification, to concatenate features from all PixelHop units applied to the downstream task (i.e., image classifications).

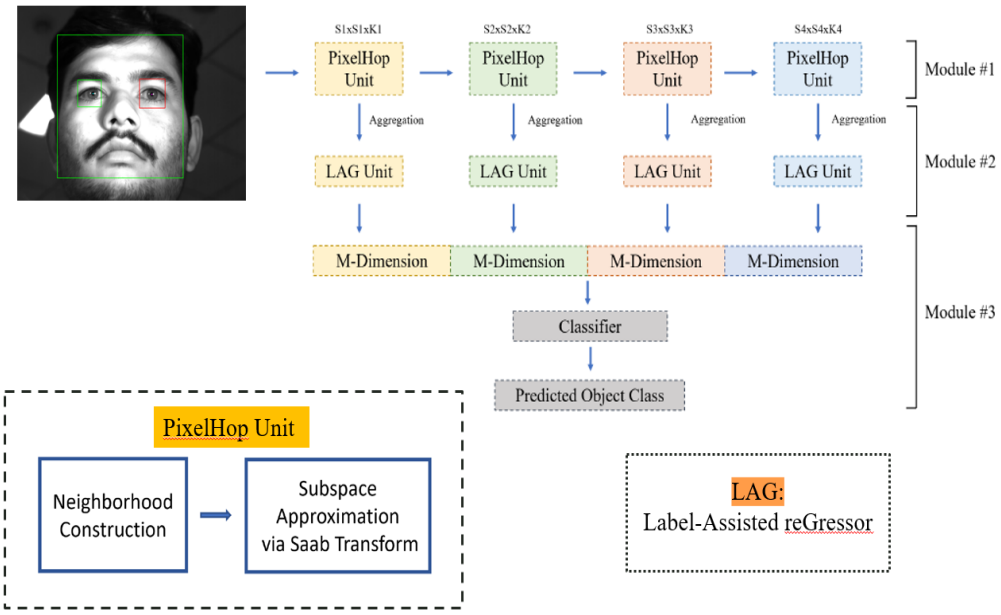

Fig. 4 Algorithmic diagram of SSL-based PixelHop method

2.3.2 SSL Properties and Comparison of SSL and DNN

SSL and DNN have some high-level concepts in common, yet they are fundamentally different in their models, training processes, and training complexities.

Similarities reside in the high-level principle. Both collect attributes in the pixel domain by employing successively growing neighborhoods. Both trade spatial-domain patterns for spectral components using convolutional filters. As the neighborhood becomes larger, the dimension of spectral components becomes larger. Due to significant neighborhood overlapping, there exists strong redundancy among neighborhoods of adjacent pixels. The spatial pooling is adopted by reducing such redundancy.

Next, we show differences between SSL and DNN and elaborate on them:

- *Model expandability*: DNN is a parametric learning method. One selects a fixed network architecture to begin with. Its superior performance is attributed to a very large model size, where the number of model parameters is typically larger than the number of training samples, leading to an over-parameterized network. This could be a waste of resource. Traditional parametric-learning methods do not have enough model parameters to deal with data sets of a larger number of samples with rich diversity. SSL adopts a nonparametric model. It is flexible to add and/or delete filters at various units depending on the size of the input data set. SSL can handle small and

large data sets using an expandable model. This is especially attractive for edge computing. Its model complexity can be adjusted flexibly based on hardware constraints with graceful performance trade-off.

- *Incremental learning*: It is challenging to adapt a trained DNN model to new data classes and samples. Since SSL employs a nonparametric model, we can check whether existing Saab filters can express the new data well. If not, we can add more Saab filters in the unsupervised dimension-reduction part. Furthermore, we can expand the regression matrix to accommodate new classes.

- *Model architecture*: DNN demands a network architecture that has an end node at which a cost function has to be defined. This is essential to allow backpropagation (BP) to train the network parameters. In contrast, the architecture of an SSL design is more flexible. We can extract rich features from processing units in multiple stages. Furthermore, we can conduct ensemble learning on these features.

- *Model interpretability*: The DNN model is a black-box tool. Many of its properties are not well understood. The SSL model is a white box, which is mathematically transparent.

- *Model parameter search*: DNN determines model parameters using an end-to-end optimization approach, which is implemented by BP. SSL adopts unsupervised and supervised dimension-reduction techniques to zoom into an effective subspace for feature extraction. The whole pipeline is conducted in a one-pass feed-forward fashion.

- *Training and testing complexity*: DNN demands many computing resources in model training due to BP. As the number of layers goes extremely deep (say, 100 and 150 layers), the inference can be very expensive as well. The training complexity of SSL is significantly lower since it is a one-pass feed-forward design. Its testing complexity is determined by the stage number. If the number of stages is small, inference can be done effectively.

- *Spectral dimension reduction*: While DNN and SSL both use convolutional operations, they have different meanings. Convolutions in DNN are used to transform one representation to another aiming at end-to-end optimization of the selected cost function for the network. Convolutions in SSL are used to find projections onto principal components of the subspace.

- *Task-independent features*: DNN uses both input images and output labels to determine system parameters. The derived features are task dependent. SSL contains two feature types: task-independent features and

task-dependent features. The features obtained by unsupervised dimension reduction are task independent while those obtained by supervised dimension reduction are task (subspace approximation with adjusted bias) dependent.

- *Multitasking*: DNN can integrate the cost functions of multiple tasks and define a new joint cost function. This joint cost function may not be optimal with respect to each individual task. SSL can obtain a set of task-independent features and feed them into different LAG units and different classifiers to realize multitasking.
- *Incorporation of priors and constraints*: DNN may add new terms to the original cost function, which corresponds to priors and constraints. The impact of the modified cost function on the learning system is implicit and indirect. SSL can use priors and constraints to prune attributes of small and large neighborhoods directly before they are fed into the classifier.
- *Weak supervision*: A large number of labeled data is needed to train DNN models. Data augmentation is often used to create more training samples. SSL outperforms DNN in the weak-supervision case. This could be attributed to the fact the unsupervised dimension-reduction process in successive PixelHop units does not demand labels. Labels are only needed in the LAG units and the training of a classifier. Besides, we may adopt a smaller SSL model in the beginning. Then, we can grow the model size by adding more confident test samples to the training data set.
- *Adversarial attacks*: It is well known that one can exploit the DNN model to find a path from the output decision space to the input data space. Then, a decision outcome can be changed by adding small perturbations to the input. The perturbation can be so small that humans may not be able to see. As a result, two almost identical images will result in different predictions. This is one major weakness of DL networks. In SSL, we expect that weak perturbation can be easily filtered out, and it is challenging for attackers to conduct similar attacks.

3. Research—Applications

A variety of applications could benefit from the new ML technologies of Saak/Saab transforms and SSL. Several typical militarily relevant tasks and scenarios are presented in this section. They involve the areas of image classification, defense against adversarial attacks, and biometric facial-data processing.

3.1 Image Classification

An earlier work[11] studied the power of Saak features as an effort toward interpretable DL. Being inspired by the operations of convolutional layers of convolutional neural networks, multistage Saak transform was suggested. Based on this foundation, we provided an in-depth examination of Saak features, which are coefficients of the Saak transform, by analyzing their properties through visualization and demonstrating their applications in image classification.

Being similar to DNN features, Saak features at later stages have larger receptive fields, yet they are obtained in a one-pass feed-forward manner without back-propagation. The whole feature-extraction process is transparent and is of extremely low complexity. The discriminant power of Saak features is demonstrated, and their classification performance in three well-known data sets—Modified National Institute of Standards and Technology (MNIST),[12] Canadian Institute for Advanced Research (CIFAR)-10,[13] and STL-10[14]—is shown by experimental results.

3.1.1 Saak Features and Discriminability

Saak features demonstrate strong discriminability, being able to capture and encode the unique structures across different images and classes. If a certain Saak kernel gives high response to important locations in an image and if it is consistent across all images of that same class, the coefficients corresponding to that kernel can be considered as a discriminant feature.

To demonstrate the discriminant capability, Fig. 5 shows the Saak feature responses to different objects using carefully selected kernels. Specific kernels are chosen for a particular class and they are convoluted with the images from those classes for visualization. We show that the chosen filters cater high responses in certain locations of images, helping us discriminate one class from other classes.

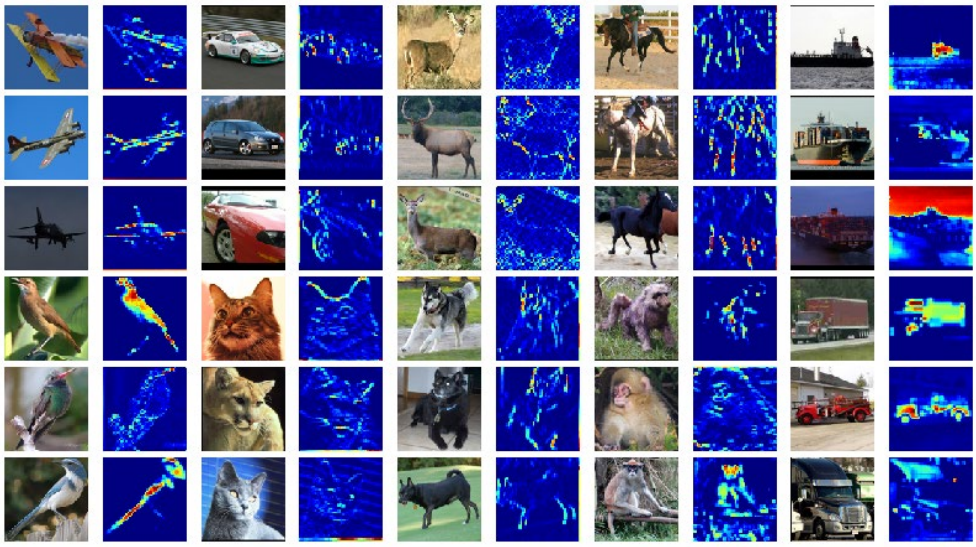

Fig. 5 Visualization of feature responses from STL-10 (three images from each class); each pair contains the original image and its first-stage Saak response; discriminant feature for every class is highlighted

We carefully choose a kernel that is consistent across all images of these classes (e.g., bird, deer, and horse classes). Thus, a particular kernel for deer class gives high response at locations of the deer's ears and horns. Similarly, a unique kernel for horse class gives high response for its legs. Two different kernel outputs are shown for bird class, and we can see that one corresponds to feathers and another corresponds to beak. From these feature visualizations we can recognize these responses are discriminant. Some important features for the airplane are its body and wings; for the ship, its stern and mast; for the cat, its face; for the car, its wheels; and for the monkey, dog, and truck, it is the body itself. This demonstrates the discriminant power of Saak features in both classifying the image and localizing class-specific image regions in a single forward pass.

Discriminant Saak kernels that work best for one class give a poor response for other classes. Saak features are not shared by many classes and hence can be used for many computer-vision applications. Careful feature-selection techniques can help us choose the right set of features for a desired application.

3.1.2 Saak-Based Classification Approach

We revisit the image-classification problem using the Saak transform theory. There is a major difference between the DNN and the Saak transform methodology. For example, the DNN converts the image-classification problem to multiple object-segmentation problems using object proposals. The Saak transform does not need any bounding boxes. As stated before, the Saak transform offers a family of joint spatial–spectral representations that have capabilities for both classifying images

and localizing class-specific image regions. It tackles the challenging segmentation task directly based on the spatial–spectral information. After the segmentation task, it provides a semantic label to each region.

The designed Saak-based algorithm, shown in Fig. 6, mainly consists of 1) extracting local cuboids (LCs) from the images, 2) obtaining KLT components, 3) convoluting the images with the extracted kernels, 4) calculating the cross-entropy measures, and 5) selecting the best spatial–spectral components. Images are convoluted with all the kernels obtained. The best coefficients are then chosen based on the lowest cross-entropy values calculated. This is done on both spectral and spatial dimensions. The responses corresponding to the best kernels are then fed as inputs to the next stage of Saak transform. The architecture in Fig. 6 shows the multistage Saak transform using 3 × 3 Saak kernels applied to input images. After three stages of Saak transform, we end up with a spatial dimension of 4 × 4 and a spectral dimension K_3. These responses are then reshaped and used for classification.

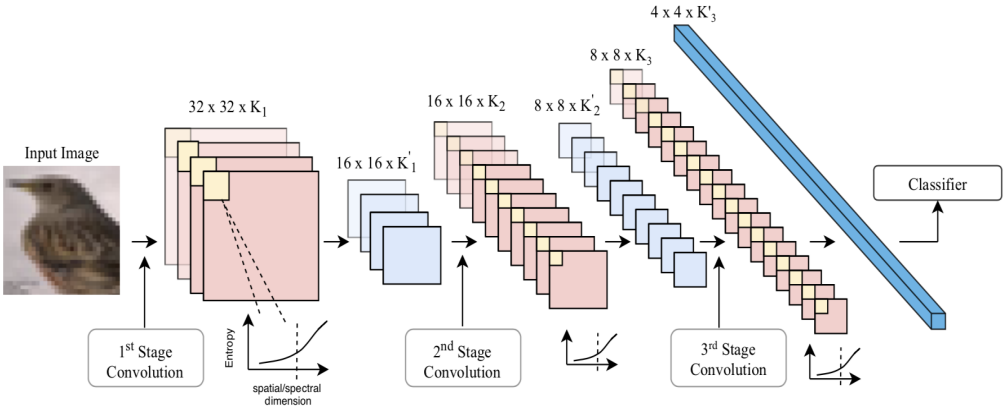

Fig. 6 3-stage Saak architecture using input image of size 32 × 32 × 3; spatial dimension reduces every stage and final stage has response of size 4 × 4 × K_0, reshaped and fed as input to classifier

3.1.3 Experiments and Evaluations

We extensively conduct experiments aimed at deeply understanding the Saak feature representation and demonstrating its benefits and utilities to computer-vision tasks. We study classification performance under different settings to provide in-depth examinations on the unique properties of Saak features in terms of discriminant power, robustness, and complexity.[11]

- *Effect of overlapping LCs*: In the original Saak architecture, nonoverlapping LCs were extracted. In the new architecture, we use overlapping kernels to maximize the information contained in each LC. Table 1 shows the classification performance using overlapping cuboids

and non-overlapping cuboids. As we can see, the overlapping cuboids with max-pooling have enhanced the classification performance; for example, the accuracy for MNIST is 99.3% and for CIFAR-10 and STL-10 the accuracy has increased by 13.3% and 8.75% respectively.

- *Effect of kernel dimension*: We evaluate the impact of Saak kernel dimension on classification performance. We experiment with different kernel dimensions of 2×2, 3×3, and 5×5 to change the receptive fields. Table 2 contains test accuracy obtained for different kernel dimensions for all three data sets. The change of kernel dimension has provided a significant boost in the performance for CIFAR-10 and STL-10. Larger kernel dimension gives better performance when objects in the images are at different locations. For MNIST, since the digits are mostly located in the center, there is a very small increase in the performance when kernel dimension is increased. In the case of CIFAR-10 and STL-10, the objects are not always in the center. Thus, an increased receptive field captures the discriminant regions and improves performance. Saak transforms capability of localizing the discriminative regions in the image is verified.

- *Robustness*: To study the robustness of Saak features we conduct experiments using images attacked by state-of-the-art methods: Basic Iterative Method (BIM),[15] DeepFool (DF),[16] and Fast Gradient Sign Method (FGSM).[17] Adversarial images are created by applying a small perturbation to an image in a way that changes the predictions made by a pre-trained model. Classification-accuracy results for attacked images are shown in Table 3 for different data sets. Many DL methods are vulnerable to adversarial images. Unlike other methods, the Saak transform-based classification is robust and the accuracies of clean and attacked images are very close.

- *Classifier*: We flexibly combine different classifiers with the extracted Saak features intending to evaluate the overall performance of a complete classification system. Four classifiers are evaluated—Random Forest (RF), Support Vector Machines (SVM), Logistic Regression (LR), and Multilayer Perceptron (MLP)—though the technique is applicable to any classifier that accepts feature representations. Table 4 shows the test accuracy corresponding to each classifier.

- *Complexity*: Complexity analysis is gaining popularity in developing practical ML frameworks. The main aim is to understand if added complexity is worth the benefits. In similar lines, we conduct complexity analysis of the Saak-transform-based feature representation. KLT

components of different-stage Saak kernels are considered for this experiment. Images are selected on the basis of stratified sampling. Table 5 displays cosine similarity of the components between subsets of images and the whole data set. We experiment with different data-set sizes. We can see the components extracted at each stage with different number of images are very stable. The KLT components from smaller data-set sizes are very similar to the components extracted from the entire training set. This is very advantageous in constrained environments where huge number of training images cannot be used.

Table 1 Effect of overlapping LCs and max-pooling on classification performance

	Accuracy	
Data set	**Nonoverlapping (%)**	**Overlapping (%)**
MNIST	98.81	99.3
CIFAR-10	61.3	74.6
STL-10	54.3	63.05

Table 2 Effect of kernel dimension on classification performance

	Accuracy		
Data set	**2×2 (%)**	**3×3 (%)**	**5×5 (%)**
MNIST	99.30	98.94	99.03
CIFAR-10	65.68	74.6	73.06
STL-10	55.30	63.05	59.50

Table 3 Performance comparison of different adversarial attacks

Attack	**FGSM (%)**	**BIM (%)**	**DeepFool (%)**
MNIST	94.52	94.13	95.51
CIFAR-10	49.50	48.50	70.44
STL-10	47.69	50.55	58.5

Table 4 Performance comparison of different classifiers

Data set	Accuracy			
	LR (%)	RF (%)	SVM (%)	MLP (%)
MNIST	98.77	96.90	98.5	99.30
CIFAR-10	64.43	55.76	58.5	74.60
STL-10	54.70	45.90	49.43	63.05

Table 5 Cosine similarity among KLT coefficients obtained via different subsets of images and the entire CIFAR-10 data set

Size	Stage 1	Stage 2	Stage 3
5000	0.9999	0.9885	0.9632
10,000	0.9999	0.9877	0.9851
20,000	0.9999	0.9881	0.98492
30,000	0.9999	0.9914	0.9816
40,000	0.9999	0.9999	0.9999

3.2 Defense against Adversarial Attacks

It has been shown that DNNs are vulnerable to adversarial attacks. These attacks come in the form of adversarial inputs with carefully crafted perturbations added to the input samples. Such perturbations are small and imperceptible to humans, but can drastically cause the classification systems to misinterpret adversarial inputs—with potentially disastrous consequences where safety and security are crucial.

In previous work[18] we investigated the robustness of the Saak transform against adversarial attacks on high-performance image classification. We developed a complete image-classification system based on the multistage Saak transform. We took advantage of the ocean of Saak coefficients available at every stage of multistage Saak transform. Careful selection of these features using cross-entropy led us to build a new Saak feature representation. The whole feature extraction and selection process is completely transparent and of extremely low complexity. In the Saak transform domain, clean and adversarial images demonstrate different distributions at different spectral dimensions. Selection of the spectral dimensions at every stage can be viewed as an automatic denoising process. Motivated by this observation, we design new strategies of feature extraction, representation, and classification that increase adversarial robustness. The performances with

well-known benchmark data sets and attacks are demonstrated by extensive experimental results.

3.2.1 Adversarial Attacks

We consider three major adversarial attacks against which we will evaluate our defensive technique:

- FGSM[17]: This method computes adversarial images by adding a pixel-wide perturbation of magnitude in the direction of gradient. The perturbation is computed so each pixel is modified. This value can be computed using back propagation. There is no bound on the modified value; hence, the quality of the adversarial image greatly decreases.
- BIM[16]: This method is an extension of FGSM but with a limit on the value that a pixel can be modified. The change is limited, but the number of iterations of the attack are increased. Hence, to the human eye the BIM-attacked images look less noisy when compared with FGSM attacks. The adversarial images are generated after multiple iterations.
- DeepFool (DF)[16]: This attack is more carefully crafted when compared with FGSM and BIM. It computes the closest L_2 projection distance to the decision-boundary hyperplane of adversarial example and input image. The perturbation is a function of this distance. The perturbation is applied iteratively with smaller steps; hence, the produced adversarial images do not look noisy to the human eye.

3.2.2 Saak-Based Defense Approach

Effective defense against adversarial attacks has been of great concern in designing machine-learning-based vision systems. Methods to defend against adversarial attacks have been done through adversarial training, adversarial detection, and gradient-masking methods. Adversarial training becomes specific to attack methods and fails to generalize while adversarial detectors still possess the risk of being fooled by the attacker. We show how our Saak–feature-based method is robust to such small perturbations in an image. Figure 7 is a sample collection of images from the CIFAR-10 data set.

In the Saak transformation domain, clean and adversarial images have different distributions at different spectral dimensions. Careful selection of the spectral dimensions at every stage can be viewed as an automatic noise-filtering technique. Fig. 8 shows distribution of Saak components belonging first to few spectral dimensions, followed by the distribution for higher spectral dimensions. Saak spectral components differ for both clean and adversarial images at higher

dimension. We also show the normalized and the original root-mean-squared-error (RMSE) values between clean and FGSM adversarial samples in different spectral components. We can observe from Figure 8's plots (c) and (d) that clean and adversarial samples have different Saak coefficient values in high spectral dimensions. These results were obtained from first stage Saak transform of CIFAR10 images using 3 × 3 LCs.

We classify adversarial images using Saak transform. We extract kernels using clean training images and follow the same procedure we use to classify clean images. As shown in Fig. 8, Saak kernels are used to extract the coefficients from attacked images. We classify adversarially attacked images after selecting features using a cross-entropy-based method. Figure 9 depicts the algorithmic architecture of a Saak-based defense approach.

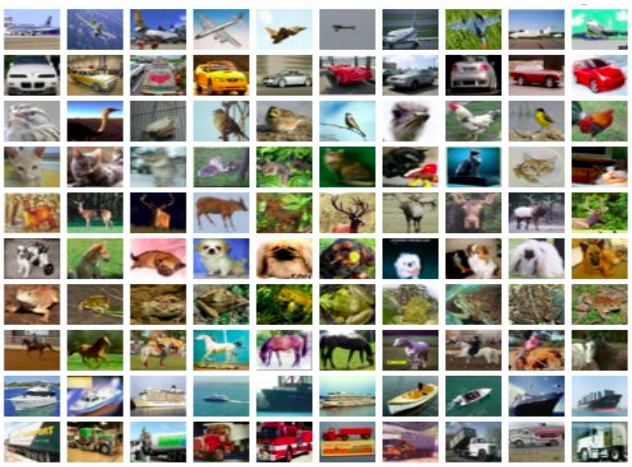

Fig. 7 Sample images from CIFAR-10 data set containing 50,000 training images and 10,000 test images for 10 object categories

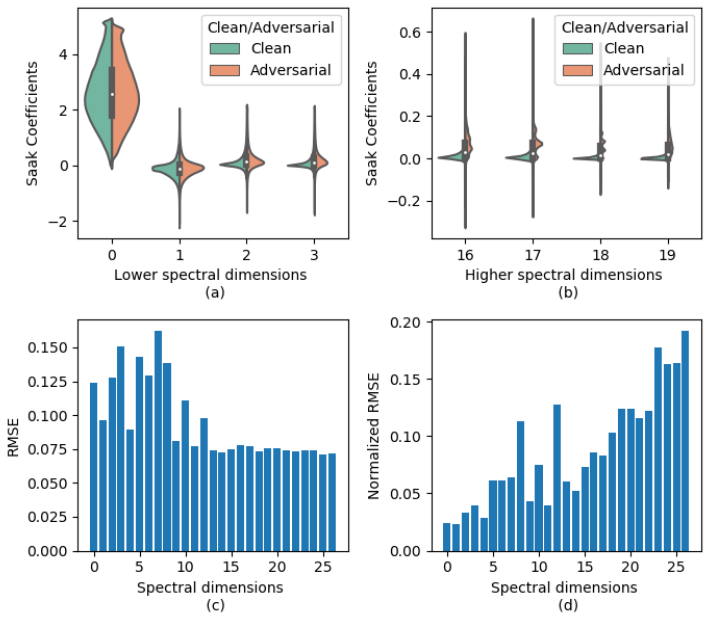

Fig. 8 Images (a) and (b) show lower and higher spectral distribution of Saak coefficients extracted from CIFAR-10—at higher dimensions, distributions from clean and attacked images differ; (c) and (d) show RMSE and normalized RMSE between clean and FGSM-attacked Saak coefficients in different spectral dimensions.

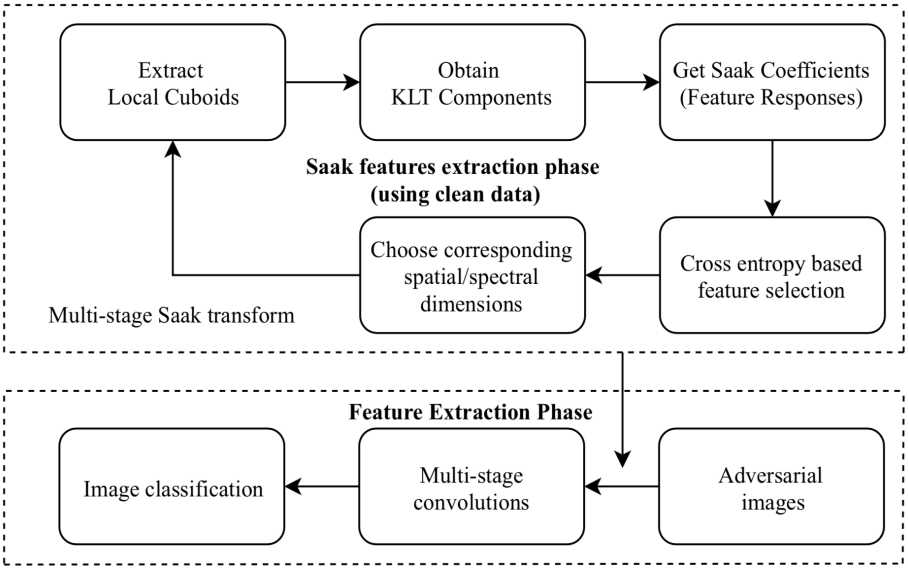

Fig. 9 Algorithmic architecture of Saak-based defense approach uses clean training images to extract kernels followed by feature extraction; feature responses can used for defending against adversarial attacks

3.2.3 Experiments and Evaluation

Extensive experiments are conducted on data sets MNIST, CIFAR-10, and STL-10. We provide in-depth experimental results for adversarial images' classification and prove that classification using the proposed Saak features is adversarially robust in comparison with state-of-the-art defense mechanisms.

Table 6 shows extensive comparison results of our Saak-transform-based classification for various attacks on the MNIST data set with other state-of-art defense methods. The values across the table indicate the drop in classification accuracy from clean and adversarially attacked images (i.e., drop in classification accuracy). The lower the drop value is, the better is the robustness of the classification model. Similarly, Tables 7 and 8 shows the results for CIFAR-10 and STL-10 data sets.

From the results, we see our approach outperforms other adversarial defense methods. The classification-accuracy drop for Saak transform features is much less, thanks to the robustness of the Saak transform to adversarial perturbations. As previously stated, the images classified with Saak transform are not subjected to any specially crafted adversarial defense method. The drop obtained for DF-attacked images is less when compared with the other attacks. For the MNIST data set, the Saak transform's drop in classification accuracy is lower than all other defenses for all the three attacks. Also, the range of drop is much less for MNIST when compared with CIFAR-10 and STL-10, mainly because the attacks are more effective in complex data sets. Even in the complex data sets, the drop in accuracy using Saak features is less and much lower than most of the defenses. The results clearly show that adversarial perturbations can be effectively and efficiently defended against using the state-of-the-art Saak transform.

Table 6 Robustness comparison on MNIST: accuracy on clean images using modified LeNet architecture is 99.2% and 99.4% using Saak transform; table's values are the drop in classification accuracy from clean and adversarially attacked images

Adversarial defense	FGSM (%)	BIM (%)	DF (%)
No defense	13.86	18.95	13
JPEG (Q = 90%)	4.61	9.52	14.14
Bit depth reduction (4-bit)	5.66	9.65	12.83
Bit depth reduction (5-bit)	3.63	8.87	13.06
Median filtering (2×2)	4.04	9.41	8.77
Median filtering (3×3)	5.14	10.2	9.29
Nonlocal (NL) means	5.27	9.54	14.52
TVM	**2.92**	8.81	5.81
Pixel deflection (w/o R-CAM)	4.54	9.55	16.44
Pixel deflection (w/ RCAM)	5.16	9.63	17.3
Saak transform	4.78	**5.17**	**3.79**

Q = JPEG quality; TVM = Total Variance Minimization; R-CAM = Robust Activation Map

Table 7 Robustness comparison on CIFAR-10; accuracy on clean images using pretrained VGG-16 model is 93.95% and 74.6% using Saak transform

Adversarial defense	FGSM (%)	BIM (%)	DF (%)
No defense	83.95	83.95	83.95
JPEG (Q = 90)	74.69	76.26	**2.97**
Bit depth (Subspace approximation with adjusted bias) reduction (4-bit)	82.08	82.94	60.35
Bit depth reduction (5-bit)	81.95	82.93	60.8
Median filtering (2×2)	77.87	77.19	71.03
Median filtering (3×3)	82.55	78.44	79.99
NL means	77.41	75.27	3.15
TVM	76.18	76.49	72.35
Pixel deflection (w/o R-CAM)	83.02	83.61	60.06
Pixel deflection (w/ R-CAM)	82.8	83.67	60.01
Saak transform	**25.1**	**26.1**	4.16

Table 8 Robustness comparison on STL-10; accuracy on clean images using pretrained network is 74.86% and 63.5% using Saak transform

Adversarial defense	FGSM (%)	BIM (%)	DF (%)
No defense	53.4	32.77	44.64
JPEG (Q = 90)	53.75	34.39	10.86
Bit depth reduction (4-bit)	53.76	33.21	27.86
Bit depth reduction (5-bit)	53.63	33.03	24.86
Median filtering (2×2)	53.22	34.91	28.94
Median filtering (3×3)	53.8	35.08	27.30
NL means	53.79	34.24	22.94
TVM	52.53	31.72	30.90
Pixel deflection (w/o RCAM)	53.46	32.26	27.54
Pixel deflection (w/ RCAM)	53.5	32.25	25.90
Saak transform	**15.36**	**12.5**	**4.55**

3.3 Face Clustering

This application is for classifying human attributes from face images. Face attributes classification is an important topic in biometrics.[19,20] The ancillary information of faces such as gender, age, and ethnicity is referred to as "soft biometrics" in forensics. The face–gender-classification problem has been extensively studied for more than two decades. Before the resurgence of DNNs, the problem was treated using the standard pattern-recognition paradigm. We have seen a rapid progress on this area due to the application of DL technology in recent years. Yet, the DL-driven methods rely on large learning models consisting of several hundreds of thousands or even millions of model parameters. The superior performance is contributed by factors such as higher input image resolutions, more and more training images, and abundant computational/memory resources.

Edge/mobile computing in a resource-constrained battlefield environment cannot meet the previously mentioned conditions. The technology of our interest finds applications in rescue missions and/or field operational settings in remote locations. The accompanying face-inference tasks are expected to execute inside a poor computing and communication infrastructure. It is essential to have a smaller learning-model size, lower training and inference complexity, and lower input image resolution. The last requirement arises from the need to image individuals at farther standoff distances, which results in faces with fewer pixels.

In previous work[19–21] we develop an SSL-based interpretable nonparametric ML solution called the FaceHop method. FaceHop has quite a few desired characteristics, including a small model size, a small training-data amount, low training complexity, and low-resolution input images. FaceHop follows the traditional pattern-recognition paradigm that decouples the feature extraction module from the decision module. However, FaceHop automatically extracts statistical features instead of handcrafted features. It is developed with the SSL principle (Section 2.3) and built upon the foundation of the Saak/Saab transform theory (Section 2.1, 2.2).

3.3.1 SSL-Based Gender-Classification Method

An overview of the proposed FaceHop system is shown in Fig. 10. It consists of four modules: 1) preprocessing, 2) PixelHop method, 3) feature extraction, and 4) attributes classification.

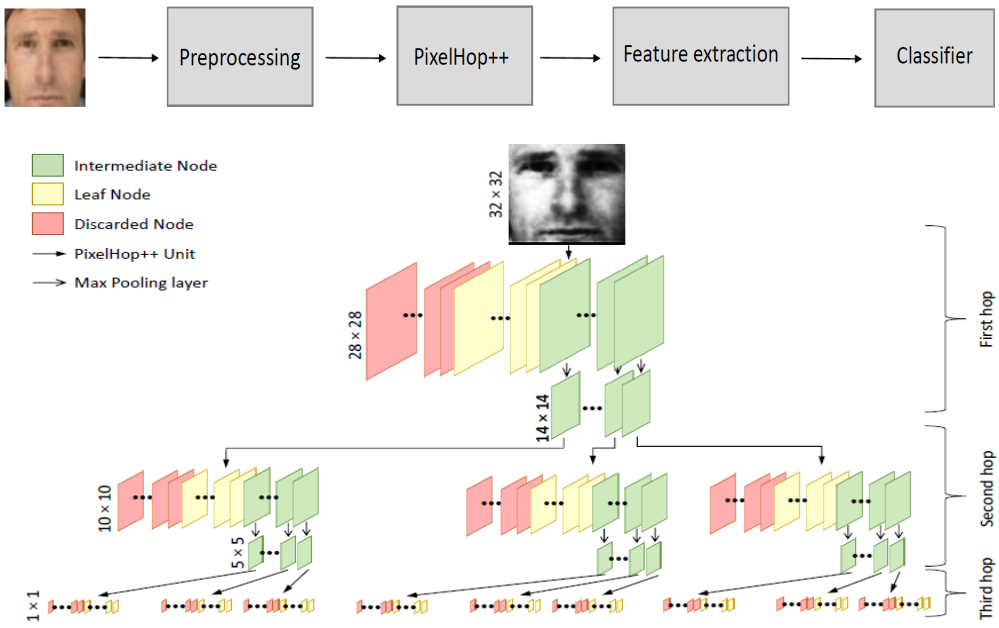

Fig. 10 Overview of proposed 3-hop FaceHop system

Preprocessing: Face images have to be well aligned in the preprocessing module to facilitate their processing in the following pipeline. Based on detected landmarks, we apply a proper 2-D rotation to each face image to reduce the effect of pose variation. Then, all face images are centered and cropped to remove background. Afterwards, we apply histogram equalization to each image to reduce the effect of different illumination conditions. Finally, all images are resized to a low resolution of 32 × 32 pixels.

PixelHop: As described in Section 2.3, PixelHop is used to describe local neighborhoods of a pixel efficiently and successively. The size of a neighborhood is characterized by the hop number. One-Hop neighborhood is the neighborhood of the smallest size. Its actual size depends on the filter size. For example, if we use a convolutional filter of size 5 × 5, then the Hop-1 neighborhood is of size 5 × 5. The Saab filter weights are obtained by performing dimension reduction on the neighborhood of a target pixel using PCA. The Saab filters in PixelHop serve as an equivalent role of convolutional filters in DNNs. We use the Saab transform to reduce its original dimension to a significantly lower one. The neighborhood concept is analogous to the receptive field of a certain layer of DNNs. As we go to deeper layers, the receptive field becomes larger in DNNs. In the SSL context, we say the neighborhood size becomes larger as the hop number increases.

Fig. 10 shows a 3-hop PixelHop system. The input is a low-resolution face image of size 32 × 32. Each hop consists of a PixelHop unit followed by a (2 × 2)-to-(1 × 1) max-pooling operation. A PixelHop system typically has three ingredients: successive neighborhood construction, channel-wise (c/w) Saab transform, and tree-decomposed feature representation.

Feature extraction: Responses at each of the three hops of the FaceHop system have different characteristics. Hop-1 responses give a spatially detailed representation of the input. Hop-2 responses give a coarser view of the entire face so that a small set of them can cover a larger spatial region. Hop-3 responses lose all spatial details but provide a single value at each frequency channel that covers the full face. The traditional eigenface approach can only capture responses of the full face and cannot obtain the information offered by Hop-1 and Hop-2 responses in the FaceHop system. We extract features based on responses in all three hops.

Classifiers: We train four classifiers in Hop-1, another three classifiers in Hop-2, and one classifier in Hop-3. Each classifier is a binary classifier. It takes a long feature vector as the input and makes a soft decision, which is the probability for the face to be a male or a female. Since the two probabilities add to unity, we only need to record one of them. Then, at the next stage, we feed these eight probabilities into a meta classifier for final decision. The choice of classifiers can be RF, SVM, and LR. Although the SVM and the RF classifiers often give higher accuracy, they have a larger number of model parameters. Since our interest lies in a smaller model size, we adopt the LR classifier only in our experiments.

3.3.2 Experiments and Evaluations

We extensively evaluate the proposed FaceHop gender classification. We compare the FaceHop solution with a variant of LeNet-5 in model sizes and verification performance. The reason of choosing the LeNet-5 for performance benchmarking is due to its small model size and good testing accuracy. Figure 11 has samples from the face-image data sets LFW and CMU Multi-PIE.

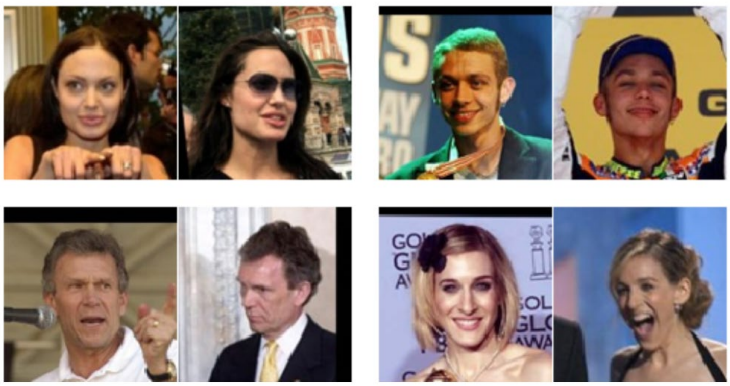

Fig. 11 Sample images from LFW and CMU Multi-PIE data sets

We use the two standard face-image data sets in our experiments: LFW data set,[22] which consists of 13,233 face images of 5,749 individuals, and CMU Multi-PIE data set,[23] which contains more than 750,000 images of 337 subjects recorded in four sessions. Since both data sets have significantly fewer female images, we use data augmentation to increase the number of female faces.

We evaluate the impact of model size on classification performance. We randomly partition male and original plus augmented female images in the LFW data set into 80% for training and 20% for testing two sets individually. Then, they are mixed again to form the desired training and testing data sets. This is done to ensure the same gender percentages in training and testing. We train eight individual hop/region LR classifiers and one meta LR classifier for ensembles. Then, we apply them to the test data to find out their performance. We repeat the same process four times to get the mean testing accuracy and the standard deviation value and report the testing performance of each individual hop/region in Table 9.

Table 9 Performance evaluation of each individual hop/region classifiers for LFW data set

Classifier	Accuracy(%)	Classifier	Accuracy (%)
Hop-1 (left eye)	86.70 ± 0.65	Hop-2 (upper stripe)	92.25 ± 0.22
Hop-1 (right eye)	86.14 ± 0.66	Hop-2 (lower stripe)	89.70 ± 0.73
Hop-1 (nose)	82.90 ± 0.61	Hop-2 (vertical strip)	92.42 ± 0.56
Hop-1 (mouth)	83.42 ± 0.74	Hop-3	91.22 ± 0.46

The mean testing accuracy ranges from 82.90% (Hop-1/nose) to 92.42% (Hop-2/vertical stripe). The standard deviation is relatively small. Furthermore, we see that Hop-2 and Hop-3 classifiers perform better than Hop-1 classifiers. Based on this observation, we consider two ensemble methods. In the first scheme, called FaceHop I, we fuse soft decisions of all eight hop/region classifiers with a meta classifier. In the second scheme, called FaceHop II, we only fuse soft decisions of four hop/region classifiers from Hop-2 and Hop-3 only. The testing accuracy and the model sizes of LeNet-5, FaceHop I, and FaceHop II are compared in Table 10. FaceHop I and FaceHop II outperform LeNet-5 in terms of classification accuracy by 1.49% and 1.65%, respectively, where their model sizes are only about 33.7% and 22.2% of LeNet-5. Clearly, FaceHop II is the favored choice among the three for its highest testing accuracy and smallest model size.

Table 10 Performance comparison of LeNet-5, FaceHop I, and FaceHop II for LFW data set

Method	Accuracy (%)	Model Size
LeNet-5	92.98	75,846
FaceHop I (all three hops)	94.47 ± 0.54	25,543
FaceHop II (hop-2 & hop-3 only)	94.63 ± 0.47	16,895

Tables 11 and 12 show the results on CMU Multi-PIE data set. This data set is much more challenging than the LFW data set. In the experiment, we consider two ensemble schemes as before; FaceHop I uses all eight soft decisions while FaceHop II takes only four soft decisions from Hop-2 and Hop-3. It is interesting to see FaceHop I and II have slightly better ensemble results of CMU Multi-PIE than of LFW, respectively. The performance of LeNet-5 also increases from 92.98% (LFW) to 95.08% (CMU Multi-PIE). As far as the model size is concerned, the model sizes of FaceHop I and FaceHop II are about 38.4% and 23.2% of LeNet-5, respectively. Again, FaceHop II is the most favored solution among the three for its highest testing accuracy and smallest model size.

Table 11 Performance evaluation of each individual hop/region classifiers for CMU Multi-PIE data set

Classifier	Accuracy (%)	Classifier	Accuracy (%)
Hop-1 (left eye)	79.33±0.33	Hop-2 (upper stripe)	91.95±0.18
Hop-1 (right eye)	78.64±0.25	Hop-2 (lower stripe)	87.00±0.15
Hop-1 (nose)	65.19±0.36	Hop-2 (vertical strip)	91.34±0.22
Hop-1 (mouth)	63.02±0.41	Hop-3	84.55±0.77

Table 12 Performance comparison of LeNet-5, FaceHop I, and FaceHop II for CMU Multi-PIE data set

Method	Accuracy (%)	Model Size
LeNet-5	95.08	75,846
FaceHop I (all three hops)	95.09 ± 0.24	29,156
FaceHop II (hop-2 and hop-3 only)	95.12 ± 0.26	17,628

In conclusion, FaceHop is a highly effective, lightweight, low-resolution face-gender classification method. This solution finds applications in resource-constrained environments with limited networking and computing. Built on the SSL principle, FaceHop provides an interpretable nonparametric ML model. It has several desired characteristics, including a small model size, a small training-data amount, low training complexity, and low resolution input images. Furthermore, we would like to extend the SSL principle and develop methods in identifying heterogeneous and correlated face attributes such as gender, age, and race. It is particularly interesting to develop a multitask learning approach.

3.4 Deepfake Detection

The creation of sophisticated fake images and videos has been demonstrated for decades through the use of various visual effects in the entertainment industry, but recent advances in AI and ML have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created. One of the most recent developments contributing to the problem is the emergence of "Deepfake", which is AI-synthesized hyper-realistic video content falsely depicting individuals saying or doing something. While Deepfake is a relatively new phenomenon, it poses a significant threat to our society and national security. For example, there could be a fake video of a military leader confessing to an illegal activity leading to a constitutional crisis or a political leader saying something racially insensitive leading to civil unrest. Significant concerns have been raised about the possible *weaponization* of such technology. There is an urgent need for technology to understand the mechanism behind the amazing performance of Deepfakes and develop effective solutions to defend against the Deepfake attacks with solid theoretical support.

This research is to pursue innovative techniques and solutions for detecting and defending against the Deepfake attacks. We developed a lightweight high-performance Deepfake-detection method, called DefakeHop.[24] State-of-the-art Deepfake-detection methods are built upon DNNs. DefakeHop extracts features automatically using the SSL principle from various parts of face images. The features are extracted by c/w Saab transform and further processed by a feature-

distillation module using spatial-dimension reduction and soft classification for each channel to get a more concise description of the face. Extensive experiments demonstrate the effectiveness of the proposed DefakeHop method. With a small model size of 42,845 parameters, DefakeHop achieves the state-of-the-art performance with the area under the ROC [receiver operating characteristic] curve (AUC) of 100%, 97.45%, 94.95%, and 90.56% for UADFV, FaceForensic++ (FF++), Celeb-DF v1, and Celeb-DF v2 data sets, respectively. As compared with existing DL methods,[25–27] DefakeHop has significantly fewer model parameters and demands much lower training complexity in solving the Deepfake video-detection problem while yielding state-of-the-art detection performance (see Fig. 12).

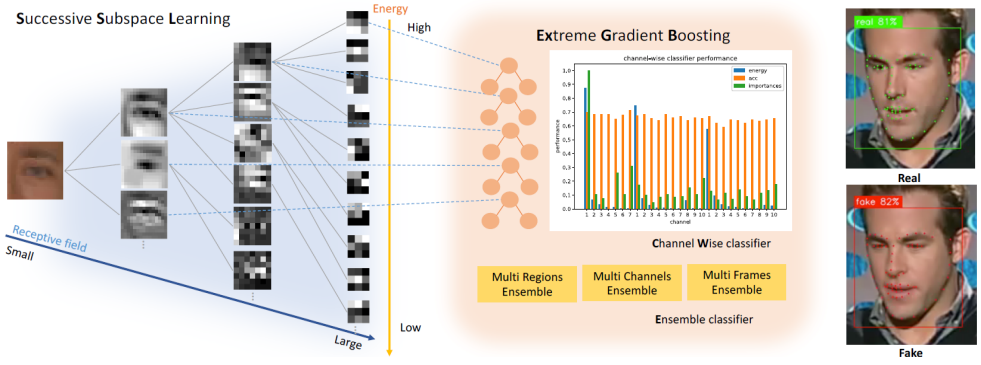

Fig. 12 DefakeHop uses SSL and extreme gradient boosting (XGBoost) to detect Deepfake videos; SSL extracts discriminant features by exploiting pixel correlations while multiple XGBoosts learn the distributions of real and fake videos for classification

3.4.1 SSL-Based DefakeHop Method

Besides the face-image preprocessing step, the developed DefakeHop method consists of two main modules: 1) feature extraction and 2) binary classification and ensemble classification. Figure 13 shows the block diagram of the DefakeHop.

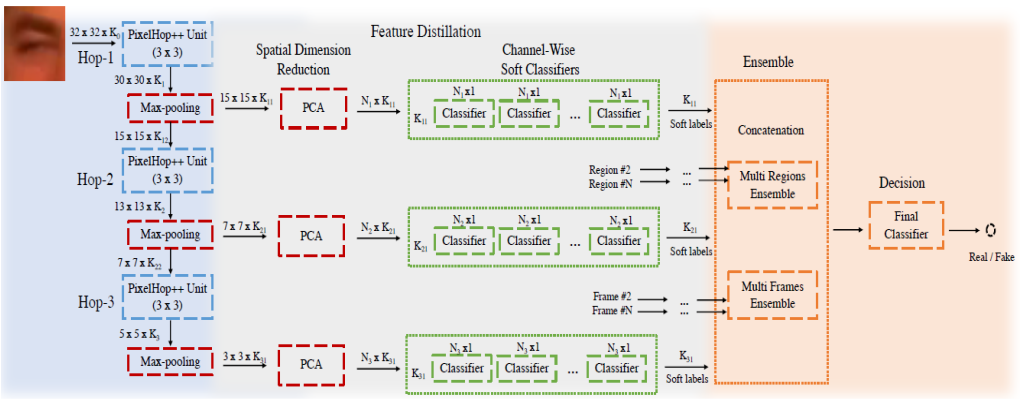

Fig. 13 Algorithmic architecture of developed DefakeHop method

Face-image preprocessing: This initial step to align and normalize input faces to ensure proper and consistent inputs are fed to the following processing modules. Preprocessing allows DefakeHop to handle various imaging conditions in resolutions, frame rates, and postures.

Feature-extraction module: It extracts rich and discriminant features from local blocks. It consists of the cascade of three PixelHop units. As shown in Fig. 14, the input is a color image of spatial resolution 32 × 32 that focuses on a particular part of a human face such as the left or the right eye region. The outputs are features obtained from each of the three PixelHop units. Since the dimension of total features is too high to feed to classifiers directly, we apply 3-D reduction techniques—neighborhood spectral dimension reduction, spatial max-pooling, and c/w spatial dimension reduction—to obtain lower-dimension features. Intuitively speaking, the three PixelHop units automatically extract a set of joint spatial–spectral features that covers different spatial resolutions. That is, the first, second, and third PixelHop units extract features in small-, mid-, and large-spatial ranges. The difference between real and fake videos can be well captured by the change of these joint spatial–spectral features.

Benchmarking Datasets

Datasets	Real	Fake	Train	Test
UADFV	49	49	78	20
FaceForensics++	1000	1000	1440	280
Celeb-DF v1	408	795	1103	100
Celeb-DF v2	890	5639	6011	518

Fig. 14 Deepfake-detection data sets used in our experiment; real, fake, train, and test video numbers for each data set are shown

Ensemble-classification module: This integrates soft decisions from all facial regions (different face patches) and selected frames to offer the final decision whether a video clip is fake or real according to following multilayers procedure:

- **Multichannel ensemble**. We concatenate the probabilities of all channels for a facial region together to get a feature vector representing this region. It is fed into an ensemble classifier to determine the probability of being fake for this region.
- **Multiregion ensemble**. Since different regions might have their strength and weakness against different Deepfake manipulations, we concatenate the

probabilities of different channels and different regions together. In experiments, we focus on three facial regions: left eye, right eye, and mouth.

- **Multiframe ensemble**. Besides concatenating different channels and regions, we concatenate the current frame and its adjacent frames so as to incorporate the temporal information. The feature vector is then fed into an ensemble classifier to determine the probability of the target frame being a fake one. The XGBoost classifier is used as the ensemble classifier.

Finally, for the whole video clip, we compute its probability of being fake by averaging the probabilities of all frames from the same video. Different approaches to aggregate frame-level probabilities can be used to determine the final decision.

3.4.2 Experiments and Evaluations

We evaluate the performance of DefakeHop on four standard Deepfake video data sets—UADFV,[25] FF++,[26] Celeb-DF v1,[27] and Celeb-DF v2[27]—and compare the results with state-of-the-art Deepfake-detection methods.

UADFV does not specify the train/test split. We randomly select 80% for training and 20% for testing. FF++ and Celeb-DF provide the test set, and all other videos are used for training the model.

Deepfake video data sets are categorized into two generations based on the data sets' size and Deepfake methods used. The first generation comprises UADFV and FF++. The second generation comprises Celeb-DF Version 1 and Version 2. Fake videos of the second generation are more realistic, which makes their detection more challenging.

DefakeHop is benchmarked with several other methods using the AUC metric. We report both frame-level and video-level AUC values for DefakeHop.[24] For the frame-level AUC, we follow the standard evaluation method that considers key frames only (rather than all frames). DefakeHop achieves the best performance on UADFV, Celeb-DF v1, and Celeb-DF v2 among all methods. On FF++/DF, its AUC is only 2.15% lower than the best result ever reported. DefakeHop outperforms other non-DL methods by a significant margin. It is also competitive against DL methods. The ROC curve of DefakeHop with respect to different data sets is shown in Fig. 15.

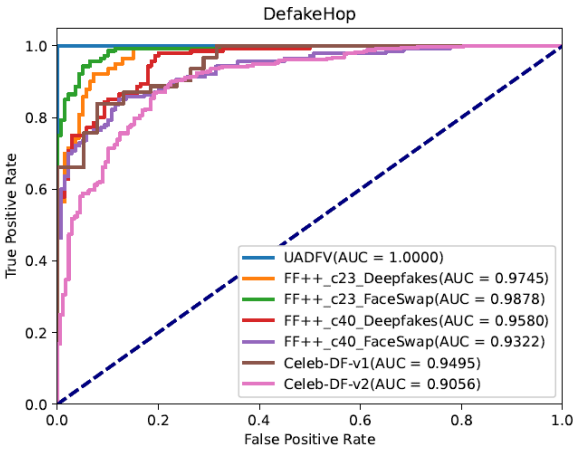

Fig. 15 Deepfake-detection data sets used in our experiment; real, fake, train, and test video numbers for each data set are shown

We compare the detection performance by considering following factors.

Facial regions: This is the performance of DefakeHop with respect to different facial regions and their ensemble results. The ensemble of multiple facial regions can boost the AUC values by up to 5%. Each facial region has different strengths on various faces, and their ensemble gives the best result.

Video quality: In the experiment, we focus on compressed videos since they are challenging for Deepfake-detection algorithms. We evaluate the AUC performance of DefakeHop on videos with different qualities. It has shown the performance of DefakeHop degrades by 5% as video quality becomes worse. Thus, DefakeHop can reasonably handle videos with different qualities.

Weak supervision: We observe that DefakeHop demands fewer training videos. An example is given to illustrate this point. Celeb-DF v2 has 6011 training videos in total. We train DefakeHop with an increasing video number in three facial regions and see that DefakeHop can achieve nearly 90% AUC with only 1424 videos (23.6% of the whole data set), where the AUC of test videos is plotted as a function of training video number in Celeb-DF v2. It is shown that DefakeHop can achieve about 85% AUC with less than 5% (250 videos) of the entire training data. It validates the claim that DefakeHop is a weakly supervised method.

Model-size computation: DefakeHop uses three PixelHop units for feature extraction. Filters in all three hops have a size of 3 × 3. For Hop-1, each input has RGB channels, leading to a 27-D input vector. For Hop-2 and Hop-3, since we use the c/w method, each input only has one channel, leading to a 9-D input vector. By limiting all c/w Saab transform outputs to 10 channels, we have at most 27 × 10, 9 × 10, 9 × 10 parameters for Hop-1, Hop-2, and Hop-3 filters, respectively.

The spatial dimension of each region of each Hop is further reduced by the PCA. Hop-1 is reduced from 225 (15 × 15) to 45, Hop-2 is reduced from 49 (7 × 7) to 30, and Hop-3 is reduced from 9 (3 × 3) to 5. The parameters for these three PCA filters are 225 × 45, 49 × 25, and 9 × 5, respectively.

4. Discussion and Conclusions

This report describes our Director's Research Awards–External Collaboration Initiative project: On Data-Driven Saak Transform: Theory and Applications. It summarizes the major achievements in every research aspect. We discussed the fundamental research on Saak transform theory and its two important extensions: Saab transform and SSL. We unified them under a common framework: "interpretable subspace learning". We applied developed theories to several *militarily relevant tasks and scenarios* including image classification, recognition, defense against adversarial attacks, and biometric facial-data processing. We achieved our research objectives: development of Saak transform as a new DL paradigm that lends itself to *white box* access and interpretation, and production of a new methodology for Army-critical applications, especially for scene perception, representation, and processing with emphasis at the tactical edge.

The DNN has underpinned state-of-the-art empirical results in numerous applied AI and ML tasks. Nonetheless, theoretical analyses of network learning are still lacking in several regards. A major concern in academia and the research community in general is that good performance numbers outweigh all other factors such as robustness, scalability, and generalizability while these numbers could be obtained with fine-tuning and trials and errors. The biggest challenge is that today's neural network-based approach works like a *black box* and there is no proven theory behind it. Our research offers an alternative paradigm that achieves similar or better performance with solid theoretical supports.

Our research is *Soldier oriented*. We expect Soldiers to carry intelligent yet extremely size–weight–power–time-efficient, vision-based devices on the battlefield. Today's ML solution is too sensitive to a specific data environment. When data are acquired in a different setting, the network needs to be retrained, which is difficult to conduct in an embedded system. The Saak transform has significantly lower complexity (estimated to be 1% of DNN training) and it is very suitable for portable devices. Furthermore, the development of a joint data compression and learning technique can create a single bit stream that includes the high-level semantic interpretation as well as the low-level visual information.[28–30] This can lead to game-changing solutions with far-reaching applications to the future Army.

5. References

1. Kuo C-CJ, Chen Y. On data-driven Saak transform. J Vis Comm Image Rep. 2018;50:237–246.
2. Soltanolkotabi M, Javanmard A, Lee JD. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. 2018; arXiv preprint arXiv:1707.04926.
3. Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: visualising image classification models and saliency maps. 2014; arXiv preprint arXiv:1312.6034.
4. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. Computer Vision–European Conference on Computer Vision; 2014. Lecture notes in computer science, vol. 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53.
5. Kuo C-CJ. Understanding convolutional neural networks with a mathematical model. J Vis Comm Image Rep. 2016;41:406413.
6. Kuo C-CJ, Zhang M, Li S, Duan J, Chen Y. Interpretable convolutional neural networks via feedforward design. J Vis Comm Image Rep. 2019;60:346–359.
7. Wang Y, Su H, Zhang B, Hu X. Interpret neural networks by identifying critical data routing paths. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 8906–8914.
8. Rouhsedaghat M, Wang Y, Ge X, Hu S, You S, Kuo C-C J. FaceHop: a light-weight low-resolution face gender classification method. arXiv preprint arXiv:2007.09510.
9. Chen Y, Rouhsedaghat M, You S, Rao R, Kuo C-C J. PixelHop++: a small successive-subspace-learning-based (SSL-based) model for image classification. arXiv preprint arXiv:2002.03141.
10. Chen Y, Kuo C-C J. PixelHop. A successive subspace learning (SSL) method for object recognition. J Vis Comm Image Rep. 2020;70:102749.
11. Manimaran A, Ramanathan T, You S, Kuo C-C J. Visualization, discriminability and applications of interpretable Saak features. J Vis Comm Image Rep. 2019;66:102699.

12. Le Cun Y, Jackel LD, Boser B, Denker JS, Graf HP, Guyon I, Henderson D, Howard RE, Hubbard W. Handwritten digit recognition: applications of neural network chips and automatic learning. IEEE Comm Mag. 1989;27(11):41–46.

13. A. Krizhevsky. Learning multiple layers of features from tiny images [master's thesis]. Department of Computer Science, University of Toronto; 2019.

14. Coates A, Ng A, Lee H. An analysis of single-layer networks in unsupervised feature learning. Proceedings of Machine Learning Research; 2011. p. 215–223.

15. Kurakin A, Goodfellow IJ, Bengio S. Adversarial examples in the physical world. 2016; preprint arXiv:1607.02533v4.

16. Moosavi-Dezfooli S, Fawzi A, Frossard P. Deepfool: a simple and accurate method to fool deep neural networks. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016. p. 2574–2582.

17. Goodfellow I, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. 2014; arXiv preprint arXiv:1412.6572.

18. Ramanathan T, Manimaran A, You S, Kuo C-C J. Robustness of Saak transform against adversarial attacks. IEEE International Conference on Image Processing; 2019.

19. Rouhsedaghat M, Wang Y, Ge X, Hu S, You S, Kuo C-CJ. FaceHop: a light-weight low-resolution face gender classification method. arXiv preprint arXiv:2007.09510.

20. Rouhsedaghat M, Wang Y, Hu S, You S, Kuo C-CJ. Low-resolution face recognition in resource-constrained environments. arXiv preprint arXiv:2011.11674.

21. Chen Y, Rouhsedaghat M, You S, Rao R, Kuo C-CJ. PixelHop++: a small successive-subspace-learning-based (SSL-based) model for image classification. IEEE International Conference on Image Processing (ICIP); 2020.

22. Huang GB, Ramesh M, Berg T, Learned-Miller E. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. University of Massachusetts; 2007 Oct. Tech Report No.: 07-49.

23. Gross R, Matthews I, Cohn J, Kanade T, Baker S. Multi-PIE. Image Vis Comp. 2010;28:807–813.

24. Chen HS, Hu S, You S, Kuo C-CJ. DefakeHop: a light-weight high-performance Deepfake detection method. IEEE International Conference on Multimedia and Expo (ICME); 2021.

25. Li Y, Lyu S. Exposing deepfake videos by detecting face warping artifacts. 2018; arXiv preprint arXiv:1811.00656.

26. Rossler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Nießner M. Faceforensics++: learning to detect manipulated facial images. Proceedings of the IEEE International Conference on Computer Vision; 2019.

27. Li Y, Yang X, Sun P, Qi H, Lyu S. Celeb-df: a large-scale challenging dataset for Deepfake forensics. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020.

28. Rouhsedaghat M, You S, Kuo C-CJ. A non-parametric and weakly-supervised face verification method for resource-constrained environments. J Vis Comm Image Rep. Article submitted 2020.

29. Lin R, Zhou Z, You S, Rao R, Kuo C-CJ. From two-class linear discriminant analysis to interpretable multilayer perceptron design. IEEE Trans Neural Net Learn Sys. Article submitted 2020.

30. Lin R, You S, Rao R, Kuo C-CJ. Constructing multilayer perceptrons as piecewise low-order polynomial approximators: a signal processing approach. IEEE Sig Proc Lett. Article submitted 2020.

List of Symbols, Abbreviations, and Acronyms

1-D	one-dimensional
2-D	two-dimensional
3-D	three-dimensional
AC	alternating current
AI	artificial intelligence
AUC	area under ROC curve
BIM	Basic Iterative Method
BP	backpropagation
CIFAR	Canadian Institute for Advanced Research
CMU	Carnegie Mellon University (data set)
c/w	channel-wise
DC	direct current
DF	DeepFool
DL	deep learning
DNN	deep neural network
FF++	FaceForensic++
FGSM	Fast Gradient Sign Method
KLT	Karhunen–Loeve Transform
LAG	label-assisted regression
LC	local cuboid
LFW	Labeled Faces in the Wild (data set)
LR	Logistic Regression
ML	machine learning
MLP	Multilayer Perceptron
MNIST	Modified National Institute of Standards and Technology
NL	nonlocal

PCA	principal component analysis
P/S	position-to-sign
R-CAM	robust activation map
ReLU	rectified linear unit
RF	Random Forest
RMSE	root mean squared error
ROC	receiver operating characteristic
Saab	Subspace Approximation with Adjusted Bias
Saak	Subspace Approximation with Augmented Kernels
S/P	sign-to-position
SSL	Successive Subspace Learning
SVM	Support Vector Machine
TVM	Total Variance Minimization
v	Version

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
3 (PDF)	DEVCOM ARL FCDD RLC CI S YOU S HU R RAO