J. Vis. Commun. Image R. 90 (2023) 103685

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

Review

Green learning: Introduction, examples and outlook ☆

C.-C. Jay Kuo *, Azad M. Madni

University of Southern California, Los Angeles, CA, USA

ARTICLE INFO

Keywords:
Machine learning
Green learning
Trust learning
Deep learning

ABSTRACT

Rapid advances in artificial intelligence (AI) in the last decade have been largely built upon the wide applications of deep learning (DL). However, the high carbon footprint yielded by larger and larger DL networks has become a concern for sustainability. Furthermore, DL decision mechanism is somewhat obscure in that it can only be verified by test data. Green learning (GL) is being proposed as an alternative paradigm to address these concerns. GL is characterized by low carbon footprints, lightweight model, low computational complexity, and logical transparency. It offers energy-efficient solutions in cloud centers as well as mobile/edge devices. GL also provides a more transparent, logical decision-making process which is essential to gaining people's trust. Several statistical tools such as unsupervised representation learning, supervised feature learning, and supervised decision learning, have been developed to achieve this goal in recent years. We have seen a few successful GL examples with performance comparable with state-of-the-art DL solutions. This paper introduces the key characteristics of GL, its demonstrated applications, and future outlook.

1. Introduction

The last decade has seen rapid advances in artificial intelligence (AI) and machine learning (ML) technology. In large part, the impressive advances have been built upon the ability to construct increasingly large datasets that enable the design of increasingly complex neural networks. Representative neural networks today include the convolutional neural network (CNN) [56], the recurrent neural network (RNN), [134,143,157] the long short-term memory network (LSTM) [55,62,156], etc. In recent years, deep neural networks (DNNs) have attracted the most attention from academia and industry since their resurgence in 2012 [85]. As networks continue to grow in size and depth, the discipline of deep learning (DL) has emerged [92]. DL has already made significant impact in various application domains, including computer vision, natural language processing, autonomous driving, robot navigation, and more.

DL is characterized by two specific design attributes: the network architecture and the loss function. Once these two attributes are specified, model parameters are automatically determined using an end-to-end optimization algorithm called backpropagation. When the number of training samples is less than the number of model parameters, it is common to adopt pre-trained networks (e.g., ResNet [60] or DenseNet [71] pretrained by ImageNet [37]) to construct larger networks to achieve superior performance. Another emerging trend is the adoption of the transformer architecture [59,72,79,165,178], which trades a model of higher complexity for further performance boosting.

Despite its rapid advance, the DL paradigm faces several technical and practical challenges. DL networks are mathematically intractable, vulnerable to adversarial attacks [3], and demand heavy supervision. Efforts have been made in explaining the behavior of DL networks with limited success, e.g., [19,36,113,133,153,179]. Adversarial training has been developed to provide a tradeoff between robustness and accuracy [208]. Self-supervised [124] and semi-supervised learning [151,164] have been explored to reduce the supervision burden.

Two further concerns over the DL technology are less addressed. The first is its high carbon footprint [90,146,181,187]. Since the training of DL networks is computationally intensive, the training of larger complex networks on huge datasets adversely affects sustainability goals [144,148,155]. The second relates to its trustworthiness. The use of blackbox DL models in high stakes decisions continues to be questioned [7,135,141]. Conclusions drawn from a set of input–output relationships can potentially be misleading and counter-intuitive. Therefore, it is essential to justify ML predictions by complementing them with logical reasoning to gain people's trust.

To address the first problem, it may be possible to optimize DL systems by taking performance and complexity into account jointly. An alternative approach is to build a new learning paradigm that strives to achieve low carbon footprint from scratch. Since the latter specifically targets minimizing carbon footprint of ML systems by design, we call it the green learning (GL) paradigm. The early development of GL was

☆ This paper has been recommended for acceptance by Zicheng Liu.

* Corresponding author.
E-mail address: cckuo@sipi.usc.edu (C.-C.J. Kuo).

https://doi.org/10.1016/j.jvcir.2022.103685
Received 11 October 2022; Accepted 29 October 2022
Available online 11 November 2022

initiated by an effort to understand the operation of computational neurons of CNNs in [86–89]. Through a series of focused investigations, building blocks of GL have been gradually developed, with additional applications being demonstrated in recent years. With regard the second problem, a clear and logical description of the decision-making process to create modularized design is emphasized in the development of GL. In other words, GL is based on the principle of modularized design. By rooting each module on local and statistical optimization, GL avoids end-to-end global optimization to achieve logical transparency and computational efficiency. Importantly, GL extensively exploits ensembles in its learning system to boost overall decision performance. Ultimately, GL yields probabilistic ML models that enable trust and risk assessment with certain performance guarantees.

GL addresses the following questions to make the learning process both efficient and effective:

1. How to remove redundancy among the data source (e.g., pixels in images) for concise representations?
2. How to generate more expressive representations?
3. How to select discriminant/relevant features based on labels?
4. How to achieve feature and decision combinations in the design of powerful classifiers/regressors?

New, powerful tools have been developed to address each of these questions in the last several years, e.g., the Saak [88] and Saab transforms [89] for question 1, PixelHop [23], PixelHop++ [25] and IPHop [188] learning systems for question 2, the discriminant and relevant feature tests [194] for question 3, and the subspace learning machine [50] for question 4. The original ideas in these papers are scattered in different publications over the years. In this paper, we systematically introduce these concepts for the benefit of the readers.

In this overview paper, we elaborate on GL's development, building modules and demonstrated applications. We will also provide an outlook for future R&D opportunities. The rest of this paper is organized as follows. The genesis of GL is reviewed in Section 2. A high-level sketch of GL is presented in Section 3. GL's methodology and its building tools are detailed in Section 4. Illustrative application examples of GL are shown in Section 5. Future technological outlook is discussed in Section 6. Finally, concluding remarks are offered in Section 7.

2. Genesis of Green learning

The proven success of DL in a wide range of applications gives a clear indication of its power. However, the reasons of its success appear to be somewhat of a mystery. Research on GL was initiated by providing a high-level understanding of the superior performance of DL [86,87,185]. The goal was not on a rigorous treatment of the subject but on obtaining insights into a set of basic questions such as:

- What is the role of nonlinear activation [86]?
- What are individual roles played by the convolutional layers and the fully connected (FC) layers [89]?
- Is it possible to avoid the expensive backpropagation optimization process in filter weight determination [88,89,108]?

With increasing understanding of the subject matter, it becomes apparent that learning can be achieved without the use of nonlinear activation and backpropagation.

2.1. Anatomy of computational neurons

As shown in Fig. 1, a computational neuron consists of two stages in cascade: (1) an affine transformation that maps an n-dimensional input vector to a scalar, and (2) a nonlinear activation operator. The affine transformation is relatively easy to understand. However, nonlinear activation obscures the function of a computational neuron. If the function of nonlinear activation is well understood, it becomes possible to remove it and replace its function with another mechanism. As

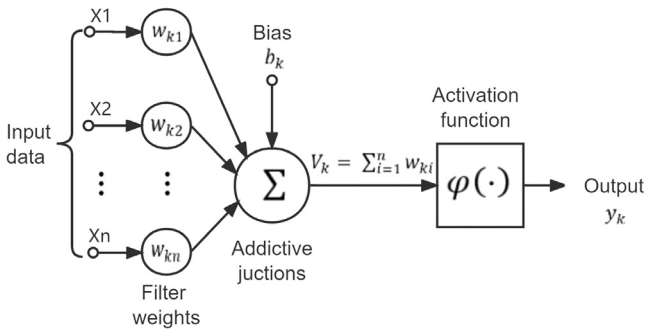

Fig. 1. Illustration of a computational neuron.

explained in Section 2.2, one neural network achieves two objectives simultaneously [89]: (1) finding an expressive embedding of the input data, and (2) making decision (i.e., classification or regression) based on data embedding.

An affine transformation contains n weights and bias b for an input vector of dimension n. According to the analysis in [50,108], weights and biases play different roles. The determination of proper weights of an affine transformation is equivalent to finding a discriminant 1D projection for partitioning. For example, one can use the linear discriminant analysis (LDA) to find a hyper-plane that partitions the whole space into two. The weight vector is the surface normal of the hyper-plane. In other words, the affine transformation defines a projection onto a line through the inner product of an input vector and the weight vector. The bias term defines the split point in the projected 1D space (i.e., greater, equal and less than zero). The role of nonlinear activation was investigated for CNNs in [86] and multilayer perceptrons (MLPs) in [108]. It is used to avoid the sign confusion problem caused by two neurons in cascade.

2.2. Anatomy of neural networks

MLPs have been commonly used as classifiers. The convolutional neural networks (CNNs) can be viewed as the cascade of two subnetworks: the convolutional layers and the FC layers. One simple example called the LeNet-5 [93] is illustrated in Fig. 2. It is convenient to have a rough breakdown of their functions as done in [89]. That is, the first sub-network is used to find powerful embeddings while the second sub-network is used for decision making. Admittedly, while this anatomy is too simplistic for more complicated networks such as ResNet [60], DenseNet [71] and transformers [165], this viewpoint is helpful in deriving the GL framework. We will elaborate on the first and the second sub-networks in Sections 2.2.1 and 2.2.2.

2.2.1. Feature subnet

Modern DL networks have inputs, outputs, and intermediate embeddings. Their dimensions for image/video data are generally large. Consider the two examples below.

Example 1. An image in the MNIST dataset has gray-scale pixels of spatial resolution 28×28. Its raw dimension is 784. LeNet-5 [93] was designed for the MNIST dataset. It has two cascaded convolutional-pooling layers. At the output of the 2nd convolutional-pooling layers, one obtains an embedding space of dimension $5 \times 5 \times 16 = 400$.

Example 2. After image size normalization, an image in the ImageNet dataset has color pixels of spatial resolution 224×224. It has a raw dimension of $224 \times 224 \times 3 = 150{,}528$. AlexNet [85] and VGG-16 [150] were proposed to solve the object classification problem in the ImageNet dataset. The last convolutional layers of AlexNet and VGG have an embedding space of $13 \times 13 \times 256 = 43{,}264$ and $7 \times 7 \times 512 = 25{,}088$ dimensions, respectively.

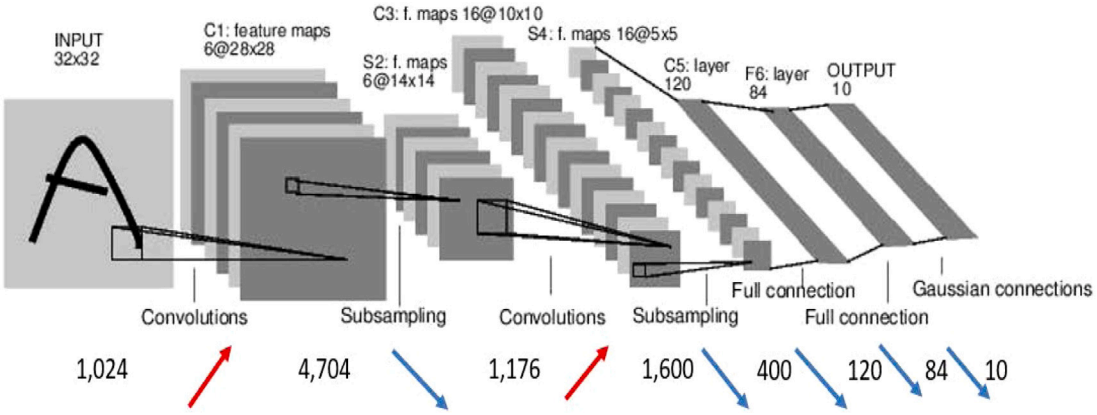

Fig. 2. Illustration of the LeNet-5 architecture [93], where the dimensions of intermediate layers are shown under the network.

The embedding dimension of the last convolutional layer is significantly smaller than the input dimension from the above two examples. Dimension reduction is essential to the simplification of a decision pipeline. It is well known that there are spatial correlations between neighboring pixels of images, which can be exploited for dimension reduction. However, dimension reduction is only one of two main functions of the feature subnet. The feature subnet needs to find discriminant dimensions at the same time.

The dimension variation at different stages of LeNet-5 is shown in Fig. 2, where a red upward arrow denotes dimension expansion and a blue downward arrow denotes dimension reduction. Dimensions increase with convolution operations in the first two layers, and dimension decreases with the (2×2)-to-(1×1) pooling operations. The increase in the dimension of an intermediate layer is determined by the network architecture. By adding neurons and reducing the stride number, the intermediate layer will have more dimensions. More embedding variables allow the generation of more expressive embeddings. The design of network architecture highly depends on specific applications (or datasets). The search of an optimal network architecture [42] and the associated loss function for new problems is costly.

2.2.2. Decision subnet

MLPs have the universal approximation capability to an arbitrary function [34,64]. Without loss of generality, consider the mapping from an n-dimensional input to an arbitrary 1D function. The latter can be approximated by a union of piece-wise lower-order polynomials. The basic unit of a piece-wise constant (or linear) approximation is a box (or a triangular-shaped) function of finite support. The form of activation determines the shape, e.g., two step activations can be used to synthesize a box function. Users are referred to [107] for more detail. For each partitioned interval, the mean value of observed output samples (or the labels of the majority class) can be assigned in a regression problem (or a classification problem). Decision making with space partitioning is essential to all ML classifiers/regressors, e.g., the support vector machine/regression (SVM/R), decision trees, random forests, gradient boosting classifiers/regressors. Yet, unlike for neural networks, no nonlinear activation is required by them. This difference will be explained in Section 4.4.

2.2.3. Filter weight determination

One characteristics of DL systems is that filter weights can be automatically adjusted by propagating decision errors backwards (i.e., backpropagation). As long as there exist negative gradients to lower the loss function, the training loss will keep decreasing. Many advanced network architectures are devised to allow more paths to avoid the vanishing gradient problem. Weight adjustment has different implications in the feature subnet and the decision subnet. For the feature subnet, weights are adjusted to find the most expressive embeddings under the architecture constraint. For the decision subnet, weights are fine-tuned to find discriminant 1D projections for space partitioning. Sometimes, there may not be sufficient labeled data to train DL networks. To boost overall performance, one may build a modularized DL system, where filter weights of some modules are pre-trained by other datasets. One example is ResNet/DenseNet pretrained by ImageNet. These pre-trained modules offer embeddings to interact with other modules of the DL system.

2.3. Interpretable feedforward-designed CNN

The feedforward-designed convolutional neural network (FF-CNN) [89] plays a transitional role from DL to GL. It follows the standard CNN architecture but determines the filter weights in a feedforward one-pass manner. Filter weights in the convolutional layers of FF-CNN are determined by the Saab transform without supervision.

Although a bias term is adopted by the Saab transform to address the sign confusion problem in [89], this term is removed in the later Saab transform implementation for the following reason. The sign confusion problem actually only exists in neural network training because of the use of backpropagation optimization. In this context, both the embeddings from the previous layer and the filter weights of the current layer have to be simultaneously determined based on the desired output. In contrast, the input to the current layer is already given in the feedforward design while filter weights are determined by the statistics of the current input. Only filter responses are computed in FF-CNN. As a result, it does not have the sign confusion problem.

Filter weights in the fully connected (FC) layers of FF-CNN are determined by linear least-squared regression (LAG). That is, in FF-CNN training, training samples of the same class are clustered into multiple sub-clusters and a pseudo-label for each sub-cluster is created. For example, there are 10 labels (i.e. 0, 1, …, 9) for each image. One can have 12 sub-clusters for each digit and create 120 pseudo-labels. The first FC layer maps from 400 latent variables in the last convolution layer to 120 pseudo-labels. The determination of filter weights can be formulated as a least-squared regression problem. FF-CNN has been used in the design of a privacy preservation framework in [68,171] and integrated with ensemble learning [27] and semi-supervised learning [28].

3. High-level sketch of GL

3.1. Overview

As mentioned in Section 1, GL tools have been devised to achieve the following objectives:

Table 1
A set of developed GL techniques.

Learning techniques	Need of supervision	Linear operation	Examples
Subspace Approximation	No	Yes	Saak transform [88], Saab transform [89]
Expressive Representation generation	Maybe	Maybe	Attention, Multi-Stage transform [25]
Ensemble-enabled Architecture	Maybe	Maybe	Pixelhop [23] Pixelhop++ [25]
Discriminant Features selection	Yes	No	Discriminant Feature test [194]
Feature space Partitioning	Yes	No	Subspace learning Machine [50]

1. Remove redundancy among the data source for concise representations.
2. Generate expressive representations.
3. Propose a system architecture that enables ensembles for performance boosting.
4. Select discriminant/relevant features based on supervision (i.e., training labels).
5. Allow feature and decision combinations in classifier/regressor design.

These techniques are summarized in Table 1.

At this point, it is worth commenting on differences and relationship between GL, DL and classical ML. Traditional ML consists of two building blocks: feature design and classification. Feature design is typically based on human intuition and domain knowledge. Feature extraction and decision are integrated without a clear boundary in DL. Once the parameters of DL networks are determined by end-to-end optimization, feature design becomes a byproduct. Techniques No. 1–3 in Table 1 correspond to the feature design in GL. Techniques 1 and 2 can be automated with little human involvement. Only hyper-parameters are provided by humans, which is similar to the network architecture design in DL. Technique 3 provides feedback from labels to the learned representations so as to zoom into the most powerful subset. Unlike traditional ML, it does not demand human intuition or domain knowledge. Finally, the last module in GL is the same as the classifier in traditional ML.

The reason to decompose the feature design module in classical ML into three individual steps in GL is for the purpose of automation. For simplicity and yet without loss of generality, we use an illustrative example to explain the four modules of GL below. Consider the ML problem of recognizing 10 handwritten digits (i.e. 0, 1, …, 9) with the MNIST dataset. It has 60,000 training images and 10,000 test images. Each image has a spatial resolution of 28×28 pixels, where each pixel has 256 gray scales. Typically, we treat each pixel as one dimension so that the input space has $28 \times 28 = 784$ dimensions. Since there are 10 output classes, we assign 10 dimensions to the output space, denoted by $(p_0, p_1, \ldots, p_9)^T$, where p_i is the probability of the test image in class i. Note that this problem can be well solved by a neural network solution called LeNet-5 [93]. Although LeNet-5 is a shallow network, its architecture can be generalized to deeper networks such as AlexNet [85] and VGG-16 [150] in a straightforward manner.

3.2. Subspace approximation

Each pixel in an image is too weak to be a discriminant for any image content. Thus, we group a set of neighboring pixels to form a basic unit. Typical units are squared blocks of size 3×3 or 5×5. There exists correlation among pixels in PixelHop units. The correlation can be reduced or removed by signal transforms. Transform kernels can be either predefined or derived from the input data. The discrete cosine transform (DCT) and the wavelet transform use predefined kernels while the Karhunen–Loéve transform (KLT) adopts the data-driven transform kernel. Two new data-driven transforms have been introduced to tackle Subproblem 1 in the GL setting. They are the Saab (Subspace approximation via adjusted bias) transform [89] and the Saak (Subspace approximation via augmented kernels) transform [88]. These new transforms will be reviewed in Section 4.2. No supervision is needed in learning the approximating subspace.

3.3. Generation of expressive representations

There are many ways to conduct the 2D transform for images in the MNIST dataset. We compare the following two designs.

A One-stage transform.
The transform has a kernel of size $28 \times 28 = 784$. The transform input and output have the same dimension; namely, 784. Since the element-wise energy is not uniformly distributed for the 784D output, we drop those elements of extremely low energy to form a representation vector.

B Two-stage transform.
The first-stage transform has a kernel size of 5×5. It is applied to 28×28 interior pixels with stride equal to one. Then, we take the absolute value of each response and conduct the maximum pooling over non-overlapping 2×2 blocks. This leads to a tensor output of dimension $14 \times 14 \times 25$, where 14×14 indicates the number of spatial locations and 25 is the channel response of the first-stage transform. Finally, we conduct the second-stage transform to 14×14 locations for each channel. The output of the second-stage transform still have a dimension of $14 \times 14 \times 25$. Again, we can drop those elements of extremely low energy to form a representation vector.

Computationally, the one-stage transform is less efficient than the two-stage transform due to its large kernel size. Here, we pay attention to another issue — the expressiveness of derived representations. For fair comparison, we select the same element number (e.g., 512) of the highest energy from designs A and B as the representation vector. We may ask which representation set is more useful in the classification problem. For design A, only a few low-frequency transform coefficients are helpful since a vast majority of high-frequency transform coefficients are not stable for images of the same label. For design B, the first-stage transform captures the local variation while the second-stage transform characterizes the global variation. It has more useful representations since we can obtain more stable associations between transform coefficients and the image class. We use expressiveness to differentiate these two representation sets.

In this example, no supervision is used to derive expressive representations. However, it is sometimes useful to exploit image labels to derive the expressive representation set. One example is attention detection. Consider the classification of dog and cat images. Both dog and cat images have background, which is irrelevant to the classification task. It is desired to remove the background effect and focus on the main object, which is called the attention. To obtain attention, we can decompose images into smaller blocks, extract block-level representations and use image labels to train the probability of a block to be a dog or a cat. For the foreground object, a test cat (or dog) block is expected to have a higher probability for the cat (or dog) decision. For a background block, if has the same likelihood to be with cat and dog images, its test probability will be about equal (i.e., 50%) for the cat and the dog. On the other hand, if it has a higher likelihood to be with cat (or dog) images, then its test probability will be higher for the cat (or dog) class. As a result, we can use blocks with more skewed probability distribution as attention. Clearly, attention offers expressive representations that have to be trained by image labels.

Subspace approximation is used to reduce redundancy in representation. However, it is still desired to find more expressive representations that are valuable in solving the target learning problem.

The former is straightforward while the latter is not. This makes GL challenging yet interesting. Since they are closely related to each other in finding an effective representation of a data source, we treat them as a whole. Details are presented in Section 4.2.

3.4. Ensemble-enabled architectures

Filter responses of DL architectures are latent variables. Their values keep evolving during the end-to-end optimization process since the filter weights are adjusted through backpropagation. The exact values of these individual latent variables are not important since they are affected by many factors such as filter weight initialization and the detailed implementation of the stochastic gradient descent optimization (e.g., the order of mini-batches and dropout). People are mainly concerned with the converged neural networks as a whole.

In contrast, filter weights in GL are determined in a feedforward one-pass manner. Once computed, these weights stay the same (i.e., they do not change any longer). The same GL architecture will have the same filter weights and responses. GL needs other mechanisms to boost the learning performance. One idea is the "ensemble-enabled architectures" as demonstrated in PixelHop, PixelHop++, IPHop, PointHop, and PointHop++. Filter responses at various stages define joint spatial–spectral representations in form of 3D tensors. The 3D tensors in earlier (or later) stages have higher (or lower) spatial dimensions and smaller (or higher) spectral dimensions. They have different physical meanings. GL can conduct feature ensembles and/or decision ensembles to achieve better performance.

3.5. Discriminant features selection

Feature selection methods can be categorized into unsupervised [152], semi-supervised [149], and supervised [69] three types. Unsupervised methods focus on the statistics of each feature dimension while ignoring the target class or value. However, their power is limited and less effective than supervised methods. Recently, Yang et al. [194] proposed a supervised learning tool in selecting discriminant features and call it the discriminant feature test (DFT). DFT can rank the discriminant power of each dimension from the highest to the lowest and determine a threshold that separates discriminant dimensions automatically. This capability, which offers an immensely powerful tool for discriminant feature selection, will be reviewed in Section 4.3.

3.6. Feature space partitioning

After discriminant feature selection, the last step is classification that maps a sample in the feature space to a label. There are two main ideas to implement the classification task:

A *Change of variables from input features to output label-probability vector.* Examples include the logistic regression classifier [41], the linear least-squares regression (LLSR) classifier [111], and the multilayer perceptron (MLP) [137].

B *Partitioning of the input feature space into purer cells whose samples are mostly from the same class.* There are two major families: (1) the support vector machine (SVM) [33] and (2) the decision tree (DT) [16], and its variants such as random forests [15] and XGBoost [20,21].

Recently, Lin et al. [108] provides a geometrical interpretation to MLP that builds a bridge between the above two idea. Furthermore, Fu et al. [50] proposed an enhanced version of DT known as the subspace learning machine (SLM) tree as well as SLM Forest and SLM Boost. They will be described in Section 4.4.

4. GL methodologies

4.1. Overview of GL systems

The architecture of a GL system is a radical departure from that of neural networks. A GL system does not have the computational neuron as its basic unit. Its global architecture is not a network, either. As illustrated in Fig. 3, a generic GL system consists of three learning modules in cascade: (1) unsupervised representation learning, (2) supervised feature learning and (3) supervised decision learning. The input and the output of the system are data sources and decisions, respectively. The two intermediate results are representations and features. They are not latent but explicit and observable. The entire system is characterized by three traits: feedforward design, individually optimized module, and statistical ensembles.

A concrete example of the GL system, called PixelHop [23], is shown in Fig. 4. Its three modules are elaborated below.

- Module #1: Derive a rich set of representations from image sources through a sequence of PixelHop units in cascade.
  The main purpose of this module is to derive attributes from near-to-far pixel neighborhoods of source images without supervision (i.e. labels). Each PixelHop unit takes the neighborhood of any pixel as the input and learns its spectral representation. The Saab transform is used for dimension reduction. Furthermore, spatial pooling can be used to reduce the dimension of the whole image. The pooling process helps enlarge the neighboring size in the following PixelHop unit.
- Module #2: Select useful features from a large set of representations.
  Each PixelHop unit yields new spatial–spectral representations of the input. Their dimensions are still high. The main purpose of this module is to find a smaller set of discriminant features from a larger set of representations based on the label information for the desired task. The Label-Assisted reGressor (LAG) unit was proposed in PixelHop to achieve this objective. A more powerful tool was recently developed to replace the LAG unit in [194], which will be discussed in Section 4.3.
- Module #3: Conduct feature ensembles and the final classification task.
  All features across multiple PixelHop units are concatenated to form an ensemble feature set. Then, they are fed into an ML classifier for the final classification task.

4.2. Unsupervised representation learning

4.2.1. Single-stage data-driven transforms

Signal transforms are widely used in signal processing to reveal the spectral properties of underlying signals. Many signal transforms are data independent such as the Fourier transform, the discrete cosine transform (DCT), and the wavelet transform. However, there are transforms that are data-driven. One famous example is KLT, also known as the principal component analysis (PCA). KLT can decorrelate elements of an input signal vector and generate a compact description. In the context of computer vision, the eigenface for face recognition [163] was developed based on the 2D KLT.

Saak and Saab transforms were proposed to define filter weights of a computational neuron in convolutional layers of CNNs. Let us use the neuron in Fig. 1 as an example. The dimension of its input vector $\mathbf{x}$ is n. Its KLT kernel is in form of $\mathbf{h}_k = (h_{k1}, \ldots, h_{k,n})^T$, $k = 1, \ldots, n$. However, we may choose a smaller kernel number for lossy approximation. Typically, eigenvalue λ_k is ordered in a decreasing order of k. If $k = n$, this leads to the lossless Saab transform. If the kernel number is reduced from k to $k' < k$, this leads to the lossy Saab transform. The Saak transform demands that filter weights appear in

Fig. 3. An overview of a generic GL system, which consists of three modules in cascade: (1) unsupervised representation learning, (2) supervised feature learning and (3) supervised decision learning.

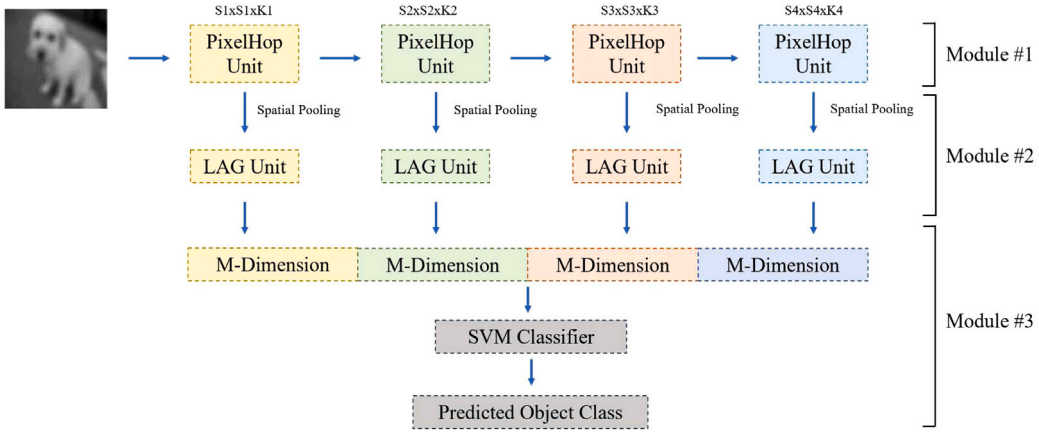

Fig. 4. The block diagram of the PixelHop learning system [23].

pairs; namely, both $\mathbf{h}_k$ and $-\mathbf{h}_k$ co-exist so that it demands $2k'$ neurons. It is desired to reduce the neuron number from $2k'$ back to k'.

Also, input $\mathbf{x}$ has to be a mean-removed vector in PCA. Although one can remove the mean of the input in the training, there is no guarantee that the training and testing data will have the same mean values. To address the mean mismatch problem, the Saab transform introduce a DC kernel whose elements have the same value. The operation of the DC kernel on the input yields a local patch mean. For images that satisfy the ergodic property, the local mean can provide a good approximation to the ensemble mean. Then, one can remove the local-mean of the input and conduct PCA to generate AC kernels. For the example in Fig. 1, one DC kernel and $(k'-1)$ AC kernels serve as the filter weights of k' neurons. To remove the effect of nonlinear activation, all filters at the same layer add the same bias such that all filter responses are positive. This filter weight design methodology is computationally efficient. Besides, it allows a simple interpretation of the role of convolutional layers.

4.2.2. Multi-stage transforms in cascade

There exist long-, mid-, and short-distance correlations in images. Such correlations are handled by multiple convolutional layers in cascade in CNNs. To enlarge the receptive field of a deeper layer in CNNs, there are two choices: (1) adopting a larger stride number, and (2) applying the max-pooling operation at the output of each neuron. The same idea is followed by Module #1 in the PixelHop system.

Saab coefficients are weakly correlated in the spectral domain due to the use of PCA in their derivation. As show in Fig. 5, the weak spectral correlation enables the decomposition of a 3D (i.e., 2D spatial and 1D spectral) input tensor of dimension $S_i \times S_i \times K_i$ into K_i spatial tensors of neighborhood size $S_i \times S_i$ (i.e., one for each spectral component) for the ith PixelHop unit, respectively. Then, instead of performing the standard 3D Saab transform as given in the top subfigure, one applies K_i channel-wise (c/w) Saab transforms to each of the spatial tensors as shown in the bottom subfigure. There are two advantages for c/w Saab transform. First, for the lossless Saab transform, the model size of the c/w Saab transform is smaller than that of the standard 3D Saab

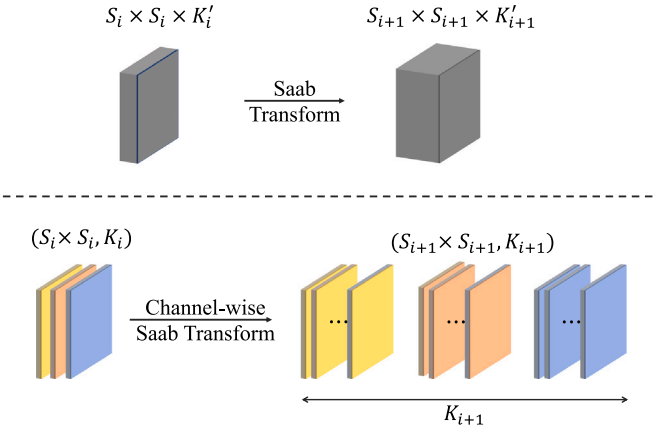

Fig. 5. Illustration of one-stage channel-wise (c/w) Saab transform, which decomposes a 3D tensor input into multiple 2D tensor inputs and multiple 2D Saab transforms (denoted by different colors individually) are conducted accordingly [25].

transform. Second, for the lossy Saab transform, the model size saving of the c/w Saab transform can be even more substantial.

To check the first point, let $n_{3D} = S_i^2 K_i$ and $n_{C/W} = S_i^2$ denote the dimensions of the inputs to the standard and c/w Saab transforms. Then, their model sizes are:

$$\text{Standard Saab:} \quad n_{3D}^2 = S_i^4 K_i^2, \tag{1}$$

$$\text{C/W Saab:} \quad K_i n_{C/W}^2 = S_i^4 K_i. \tag{2}$$

To see the second point, we show the multi-stage lossy c/w Saab transform for the whole image of size 32×32 as an example in Fig. 6. For high frequency components, the spatial correlation of pixels in a local patch is so weak that its covariance matrix is a nearly diagonal matrix, and there is no need to conduct the Saab transform afterwards. These channels are colored in pink in Fig. 6 and called leaf nodes. On the other hand, for low frequency components, the spatial correlation

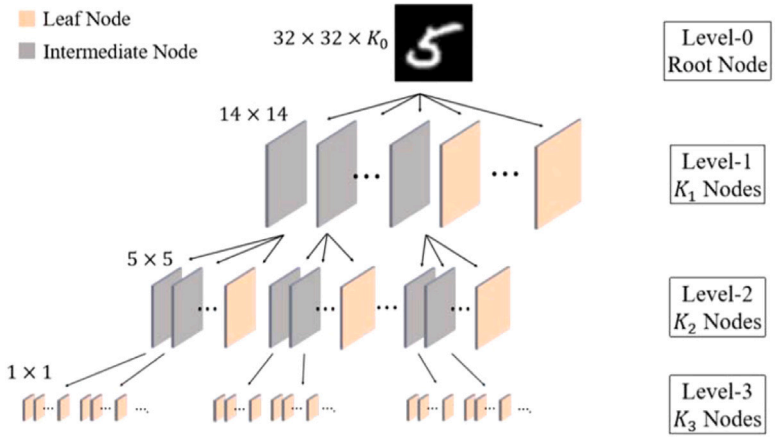

Fig. 6. Illustration of multi-stage lossy c/w Saab transforms, where 2D Saab transforms are only conducted at selected low frequency channels [25].

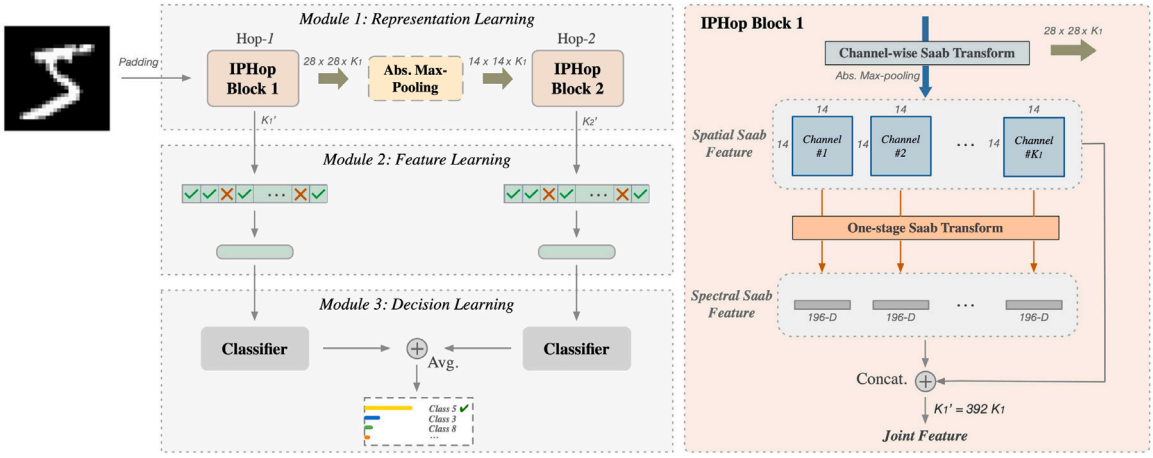

Fig. 7. Block diagram of the IPHop learning system [188], where IPHop is the acronym of the improved PixelHop.

of pixels in a local patch is still strong so that the c/w Saab transform is conducted. They are colored in gray and called intermediate nodes.

4.2.3. Enrichment of expressive representations

There are several means to generate expressive representations. Two means were proposed in IPHop [188]. IPHop is an improved PixelHop system. Its system diagram is shown in Fig. 7, which has two hops.

The first module of IPHop is similar to that of PixelHop++ except that max-pooling in PixelHop++ is changed to absolute-max-pooling. That is, the responses from the previous stage can be either positive or negative (without adding the bias term). Instead of clipping negative values to zero as done in the ReLU unit, we take the absolute value of all responses followed by the maximum pooling operation. The absolute-max-pooling operation gives a small gain over the straightforward max-pooling since the former preserves the low frequency components of the spatial responses at the output of any single Saab filter.

The filter responses extracted at Hop-1 and Hop-2 only have a local view on a larger object due to the limited receptive field size. Thus, they are not discriminant enough. For a given Saab filter, its responses are spatially correlated. To decorrelate them, one can conduct another Saab transform across them at each channel. This operation provides channel-wise spectral Saab representations at Hop-1 and Hop-2 as shown in the right subfigure of Fig. 7. As compared to representations learned by enlarging the neighborhood range gradually, the channel-wise spectral Saab representations can capture the long-range correlation at a fine scale. The spatial and spectral Saab representations are concatenated at Hop-1 and Hop-2 to form joint-spatial–spectral Saab representations.

4.3. Supervised feature learning

The leverage of labels (i.e., supervision) effectively to boost the performance of a learning system is a key question in ML. While classical ML exploits labels only in classifier design, DL uses labels to also adjust filter weights in the feature subnet and the decision subnet and thereby achieves better performance. The linear least-squared regression adopted by FF-CNN is also an example of supervised feature learning. An effective supervised learning tool is discussed in this subsection.

Existing semi-supervised and supervised feature selection methods can be classified into wrapper, filter and embedded three classes [149]. Wrapper methods [84] create multiple models with different subsets of input features and select the model containing the features that yield the best performance. One example is recursive feature elimination (RFE) [57]. This process can be computationally expensive. Filter methods involve evaluating the relationship between input and target variables using statistics and selecting those variables that have the strongest relation with the target ones. One example is the analysis of variance (ANOVA) [145]. This approach is computationally efficient with robust performance. Embedded methods perform feature selection in the process of training and are usually specific to a single learner. One example is "feature importance" (FI) obtained from the training process of the XGBoost classifier/regressor [20,21], which is also known as "feature selection from model" (SFM). Recently, the discriminant feature test (DFT) and the relevant feature test (RFT) were proposed in [194] for the classification and the regression problems, respectively. They belong to the filter class.

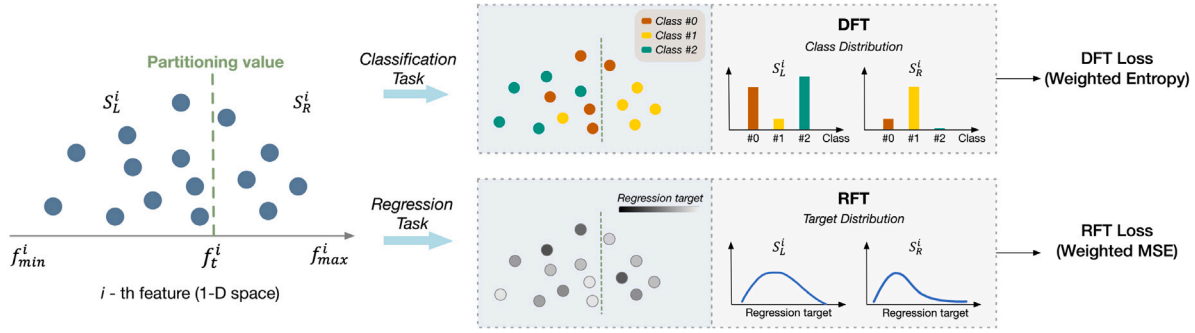

Fig. 8. Block diagram of two supervised feature selection methods: the discriminant feature test (DFT) and the relevant feature test (RFT). For the ith feature, DFT measures the class distribution in S_L^i and S_R^i to compute the weighted entropy as the DFT loss while RFT measures the weighted estimated regression MSE in both sets as the RFT loss. They are used to select discriminant/relevant features from a large set of representations learned from the source without labels [194].

Consider a classification problem with N data samples, P features and C classes. Let f^i, $1 \leqslant i \leqslant P$, be a feature dimension and its minimum and maximum are $f^i_{\min}$ and $f^i_{\max}$, respectively. DFT is used to measure the discriminant power of each feature dimension out of a P-dimensional feature space independently. If feature f^i is a discriminant one, one expects that data samples projected to it should be classified more easily. To check it, an idea is to partition $[f^i_{\min}, f^i_{\max}]$ into M non-overlapping subintervals and adopt the maximum likelihood rule to assign the majority class label to samples inside each sub-interval. Then, one can compute the percentage of correct predictions. The higher the prediction accuracy, the higher the discriminant power. Although prediction accuracy may serve as an indicator for purity, it does not tell the distribution of the remaining $C-1$ classes if $C > 2$. Since the weighted entropy measure provides more valuable information, it was adopted in [194].

By following the practice of a binary decision tree, the work in [194] chose $M = 2$ as shown in the left subfigure of Fig. 8, where f_t^i denotes the threshold position of two sub-intervals. For a sample with its ith dimension, $x_n^i < f_t^i$, it goes to the subset associated with the left subinterval. Otherwise, it will go to the subset associated with the right subinterval. DFT and RFT uses the weighted entropy and the weighted mean-squared-error (MSE) as the cost functions, respectively. The optimal cost values are computed for each feature dimension. A representation is more discriminant (or relevant) if its optimal cost value is lower. To select a set of discriminant features, one can sort representations based on their optimal cost values from the lowest to the highest and plot the curve accordingly. A representative DFT curve is shown in Fig. 9. An elbow point can be identified in most applications. Both early and late elbow points were considered in [194]. A late elbow point was illustrated in Fig. 9, which contains around 150 sorted representation indices. The set of representations with optimal cost values lower than the elbow point are selected as features. They are enclosed by a red box in Fig. 9.

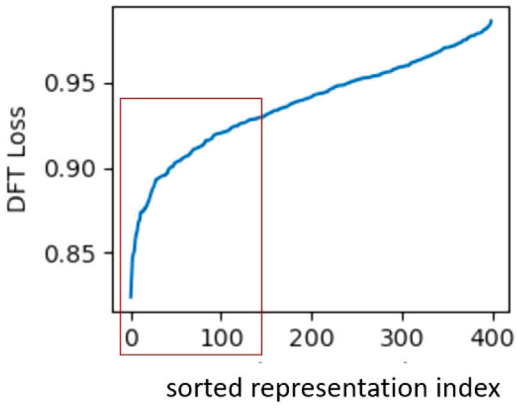

Fig. 9. The plot of a representative DFT curve, where the x-axis denotes sorted representations, and the y-axis shows the optimal cost value of an individual representation. The red box encloses representations that are selected as features.

4.4. Supervised decision learning

Consider a classification problem with N-dimensional feature vector, denoted by $\mathbf{f} = (f_1, \ldots, f_N)^T$, and M-dimensional decision vector, denoted by $\mathbf{d} = (d_1, \ldots, d_M)^T$. The latter means that there are M classes and d_m, $m = 1, \ldots, M$, is the probability of choosing class m. To optimize the performance of a classifier, it is often to consider two types of manipulation:

- Feature Combination:

$$F : \mathbf{f} \rightarrow \mathbf{w}^T\mathbf{f} + w_0 \tag{3}$$

where $\mathbf{w} = (w_1, \ldots, w_N)^T$ is an N-dimensional weight vector.

- Decision Combination:

$$D : (\mathbf{d}_1, \ldots, \mathbf{d}_P)^T \rightarrow \sum_{p=1}^{P} \alpha_p \mathbf{d}_p \tag{4}$$

where $\mathbf{d}_1, \ldots, \mathbf{d}_P$ are P decision vectors.

Table 2
Categorization of classifiers based on whether a feature combination operation (along columns) or a decision combination operation (along rows) is conducted in the classification procedure.

Decision combination	Feature Combination		
	None	One-stage	Multi-stage
None	Decision tree	SVM	MLP, SLM tree
Ensemble	Random forest		SLM forest
Boosting	XGboost		SLM boost

The decision learning problem is to find optimal feature weight vector $\mathbf{w}$ and/or decision weight vector $\alpha = (\alpha_1 \cdots \alpha_P)^T$ to optimize the classification performance. Table 2 lists several representative classifiers. They are categorized based on whether a feature processing operation (along columns) or a decision processing operation (along rows) is exploited in the classification procedure.

4.4.1. Classifiers built upon feature combinations

The use of feature combinations to optimize classification performance is examined in this section. Only SVM, SLM and MLP in Table 2 exploit feature combinations.

To introduce the SVM classifier, one can re-write Eq. (3) and set it to zero:

$$\mathbf{w}^T\mathbf{f} - \phi = 0, \tag{5}$$

where $\phi = -w_0$ is a bias. There is a physical meaning associated with Eq. (3). It corresponds to an N-dimensional hyper-plane that partitions the R^N space into two nonoverlapping half-spaces:

$$R_+ = \left\{\mathbf{f} \mid \mathbf{w}^T\mathbf{f} - \phi \geqslant 0\right\}, \tag{6}$$

$$R_- = \left\{\mathbf{f} \mid \mathbf{w}^T\mathbf{f} - \phi < 0\right\}. \tag{7}$$

For a simple binary classification problem, SVM chooses parameters $\mathbf{w}^T$ and ϕ to maximize the separation, or margin, between the two classes. Such a hyper-plane is called the maximum-margin hyper-plane. The location of the hyper-plane is determined by a couple of specific data points called support vectors. Generally, a set of hyper-planes is determined to partition the feature space.

SVM and a single SLM tree use different criteria to search for the optimal parameters. The optimization of an SLM tree consists of two steps. The first step is to find the most discriminant 1D direction, denoted by $\mathbf{w}_{opt}$, that minimizes the weighted entropy of a decision under an optimal split point $-w_0$ (or ϕ). The second step is to project N-dimensional feature vectors onto 1D feature vectors and then find the optimal split (bias) in the 1D space. A probabilistic search and an adaptive particle swarm optimization algorithm were proposed in [49,50] to implement the SLM, respectively. Actually, the stochastic gradient descent (SGD) algorithm can also be used to speed up the search of optimal combination parameters.

It is important to emphasize that the feature combination operations in SVM and SLM only yield partitioning hyper-planes. They do not change the feature vectors of input samples. The sign of $\mathbf{w}^T\mathbf{f} - \phi$ can be evaluated to decide the location of a sample against a specific partitioned hyper-plane. When there are multiple partitioning planes, the information can be binary coded against each of the planes as explained in [108]. Two samples belong to the same region if they share the same binary code. However, this is not the case in MLP.

The feature combination in Eq. (3) transforms the embeddings of all samples from their original space into a new scalar variable, denoted by g, at a single neuron. If there are Q neurons, the embeddings of all samples evolve in the following manner:

$$\mathbf{f} = (f_1, \ldots, f_N)^T \rightarrow \mathbf{g} = (g_1, \ldots, g_Q)^T, \tag{8}$$

where

$$g_q = \sum_{i=1}^{N} w_{q,i} f_i + w_{q,0}, \quad q = 1, \ldots, Q. \tag{9}$$

The embedding combination occurs at least at two layers (e.g., one hidden layer and the output layer). A nonlinear activation unit is needed after each embedding combination step to resolve the sign confusion [86,108].

4.4.2. Classifiers built upon decision processing

For traditional DTs, the random forest (RF) is the most popular bagging ensemble algorithm. It consists of a set of tree predictors, where each tree is built based on the values of a random vector sampled independently and with the same distribution for all trees in the forest. With the Strong Law of Large Numbers, the performance of RF converges as the tree number increases. As compared to the individual DTs, significant performance improvement is achieved with the combination of many weak decision trees.

Motivated by RF, SLM Forest is developed by learning a series of single SLM tree models to enhance the predictive performance of each individual SLM tree. SLM is a predictive model stronger than DT. Besides, the probabilistic projection provides diversity between different SLM models. Following RF, SLM Forest takes the majority vote of the individual SLM trees as the ensemble result for classification tasks, and it adopts the mean of each SLM tree prediction as the ensemble result for regression tasks.

With standard DTs as weak learners, GBDT [48] and XGBoost [20,21] can deal with a large amount of data efficiently and achieve the state-of-the-art performance in many machine learning problems. They take the ensemble of standard DTs with boosting, i.e. by defining an objective function and optimizing it with learning a sequence of DTs. By following the gradient boosting process, SLM Boost was proposed to ensemble a sequence of SLM trees in [50]. All the above discussion can be extended to the design of the subspace learning regressor (SLR).

4.5. Historical notes

1. Early Exploration (2015–2017)
   In the early exploration stage, research efforts focused on the understanding of the roles played by the filter weights and the nonlinear activation unit in a computational neuron of CNNs. The main objective was to shed light on the superior performance from the signal processing viewpoint since it was difficult to get engineering insights from the end-to-end non-convex optimization problem formulation. The necessity of nonlinear activation was explained in [86] as a mechanism to resolve the sign confusion problem in neural network training. The filter weights were explained as the memory of a CNN learning system [87] and used to capture local, mid-level and global intrinsic representations of objects [185]. Although the explanation of the role of filter weights was preliminary, it did inspire the later work on unsupervised representation learning.
2. FF-CNN, Saab Transforms and channel-wise Saab Transforms (2018–2019)
   Based on the foundation laid by [87,185], novel ways to determine filter weights of the convolutional layers in CNNs in a feedforward manner were proposed in [89]. The design is called the feedforward CNN (FF-CNN). The filter weights in convolutional layers of FF-CNN are determined by the Saab (or Saak) transforms. The transform operations are unsupervised since no class labels are used. The filter weights in FC layers of FF-CNN are determined by the label-guided least-squared regression (LAG). The main objective was to lower the training complexity with an interpretable design.
3. PixelHop, PixelHop++, PointHop, PointHop++, and GL Applications (2020–2021)
   PixelHop [23] and PixelHop++ [25] were two first-generation GL systems. Their designs are a total departure from that of neural networks. Instead of demanding end-to-end connectivity of neurons, GL systems adopt feedforward and ensemble learning principle. Both PixelHop and PixelHop++ have three modules. They use multi-stage Saab transforms and pooling operation to yield joint spatial–spectral representations with no supervision in the first module. The LAG operation is exploited to reduce the representation dimension before being fed into classifiers in the second stage. A traditional ML classifier is used in the third stage. The main difference between PixelHop and PixelHop++ lies in the multi-stage "standard" and "channel-wise" Saab transforms. While PixelHop and PixelHop++ target at the image classification application, PointHop [207] and PointHop++ [205] were developed for the point cloud classification. Due to the irregular spatial structure of a 3D point cloud scan, a novel octant representation was introduced before the Saab transform in PointHop. We have witnessed many successful applications of GL since 2020. Their performance is comparable to that of state-of-the-art DL solutions with significantly less resource. Examples include image classification [192], image enhancement [9], image quality assessment [121,203], deepfake image/video detection [22,24,29,218], point cloud classification, segmentation, registration [74,75,77,110,201,202], face biometrics [139,140], texture analysis and synthesis [95,96,200], image generation [8,94] graph node classification [183,184], data compression [26,206], joint image compression and classification [162], 3D medical image analysis [39,109], etc. We will present several successful examples to provide insights for future development of GL in Section 5.

Table 3
The history on the development of various GL techniques.

Year	Technology	Description	Ref.
2016	Sign confusion	Identified the sign confusion problem and explained	[86]
	Resolution	The use of nonlinear activation to resolve it in CNNs	
2017	RECOS	Attempted to give a physical meaning to filter weights	[87]
	Transform	In CNNs	
2018	Saak	Determined filter weights in CNNs with Saak transform	[88]
	Transform	Which is a variant of principal component analysis	
2019	Saab	Determined filter weights in CNNs with Saab transform	[89]
	Transform	Which is another variant of principal component analysis	
2020	Pixelhop	Proposed Successive Subspace Learning (SSL),	[23]
		Which deviates from CNN architecture completely	
2020	Label-Assisted	Attempted to link unsupervised representations with labels	[23]
	Regression		
2020	Pointhop	Applied Saab transform to octant representations	[207]
		Targeting at point cloud data	
2020	Channel-Wise	Proposed an effective multi-stage Saab transform	[25,205]
	Saab Trans.	Applicable to Pixelhop and Pointhop	
2022	DFT/RFT	Proposed effective supervised feature selection methods:	[194]
		Discriminant/Relevant Feature Tests	
2022	SLM/SLR	Proposed a new classifier/regressor that allows	[49,50]
		Both feature and decision combinations	
2022	IPhop	Proposed an improved Pixelhop system that has new Unsupervised representations and replaces LAG with DFT	[188]

4. Advanced GL Tools and Broader Applications (2022-)
   Two advanced GL tools were developed in 2022: (1) DFT/RFT for supervised discriminant/relevant feature selection [194] (see Section 4.3) and (2) SLM/SLR as a new classifier/regressor that exploits both feature combinations and decision combinations [50] (see Section 4.4).
   Recently, a more advanced GL system called IPHop was proposed in [188]. It replaces the traditional max-pooling operation with the absolute-max-pooling operation and yields more unsupervised representations by introducing the spatial Saab transform at each channel in the first module of PixelHop. Furthermore, it replaces the LAG operation with DFT in the second module of PixelHop.

The hostorical notes as described above are summarized in Table 3.

5. Demonstrated examples

GL has been successfully applied to quite a few application problems. In this section, we choose five representative examples to illustrate the power of GL.

5.1. Deepfake detection

The fast-growing Generative Adversarial Network (GAN) technology has been applied to image forgery in recent years. GANs are effective in reducing manipulation traces detectable by human eyes. It is challenging for humans to distinguish Deepfake images from real images because of the level of sophistication of deepfake methods.

There are three generations of deepfake video datasets, which shows the evolution of deepfake techniques. The UADFV dataset [99] belongs to the first-generation. It has 50 video clips generated by one deepfake method. Its real and fake videos can be easily detected by human eyes. FaceForensics++ (FF++) [138], Celeb-DF-v1 and Celeb-DF-v2 [103] belong to the second generation. They have more video clips with more subjects. It is difficult for humans to distinguish real and fake faces. DFDC [40] is an example of the third-generation dataset. It contains more than 100k fake videos generated by 8 deepfake techniques and perturbed by 19 distortion types.

State-of-the-art deepfake detectors are based on DNNs. Although they offer high detection performance, their model sizes are exceptionally large so that they cannot be deployed on mobile phones. For example, the winning team of the DFDC contest [147] used seven pre-trained EfficientNets that contain 432 million parameters. In contrast, two lightweight deepfake detectors called DefakeHop [24] and DefakeHop++ [22] were developed based on the GL principle. Their model sizes are significantly smaller as shown in the last column of Table 4.

The detection performances of several deepfake detectors on the second-generation datasets under cross-domain training are compared in Table 4, where AUC is used as the performance metric. For the cross-domain setting, deepfake detectors were trained on FF++ and tested on Celeb-DF-v1 and Celeb-DF-v2. As shown in the table, video-level DefakeHop++ gives the best AUC score while frame-level DefakeHop++ gives the second best for Celeb-DF-v1. As to Celeb-DF-v2, Multi-attentional has the best AUC score while Xception-raw and Xception-c23 offer the second best scores. DefakeHop++ is slightly inferior to them. Furthermore, the performance of DefakeHop and DefakeHop++ under the same domain training is shown in the last four rows of Table 4. Their performance has improved significantly. Video-level DefakeHop++ outperforms video-level DefakeHop by 2.5% in Celeb-DF v1 and 6.1% in Celeb-DF v2.

Fig. 10 compares the detection AUC performance of MobileNet v3 and DefakeHop++ on the third-generation dataset, DFDC. MobileNet v3 is a lightweight CNN model of 1.5M parameters targeting at mobile applications, DefakeHop++ has an even smaller model size of 238k parameters (i.e., 16% of MibileNet v3). We see from Fig. 10 that DefakeHop++ outperforms MobileNet v3 without data augmentation and leverage of pre-trained models. DefakeHop++ and pre-trained MobileNet v3 have similar detection performance under weak supervision.

5.2. Blind image/video quality assessment

Image/video quality assessment (QA) aims at evaluating image/video quality to facilitate image streaming. Based on the availability

Table 4
Comparison of the detection performance of several deepfake detectors on the second-generation datasets under cross-domain training and with AUC as the performance metric. The AUC results of DefakeHop and DefakeHop++ in both frame-level and video-level are given. The best and the second-best results are shown in boldface and with underbar, respectively. Furthermore, we include results of DefakeHop and DefakeHop++ under the same-domain training in the last 4 rows. The AUC results of benchmarking methods are taken from [103] and the number of parameters are from https://keras.io/api/applications. Also, [a] and [b] denote DL and GL methods, respectively [22].

Methods	Model	2nd Generation		Param No.
		Celeb-DF v1	Celeb-DF v2	
Two-stream [214]	InceptionV3[a]	55.7%	53.8%	23.9M
Meso4 [2]	Designed CNN[a]	53.6%	54.8%	28.0k
MesoInception4 [2]	Designed CNN[a]	49.6%	53.6%	28.6k
HeadPose [191]	SVM[b]	54.8%	54.6%	–
FWA [99]	ResNet-50[a]	53.8%	56.9%	25.6M
VA-MLP [119]	Designed CNN[a]	48.8%	55.0%	–
VA-LogReg [119]	Logistic regression[b]	46.9%	55.1%	–
Xception-raw [138]	XceptionNet[a]	38.7%	48.2%	22.9M
Xception-c23 [138]	XceptionNet[a]	–	65.3%	22.9M
Xception-c40 [138]	XceptionNet[a]	–	65.5%	22.9M
Multi-task [128]	Designed CNN[a]	36.5%	54.3%	–
Capsule [130]	CapsuleNet[a]	–	57.5%	3.9M
DSP-FWA [100]	SPPNet[a]	–	64.6%	–
Multi-attentional [212]	Efficient-B4[a]	–	**67.4%**	19.5M
Defakehop++ [22] (Frame)	DefakeHop++[b]	56.30%	60.5%	238k
Defakehop++ [22] (Video)	DefakeHop++[b]	**58.15%**	62.4%	238k
Defakehop [24] (Trained on Celeb-DF, Frame)	DefakeHop[b]	93.1%	87.7%	42.8k
Defakehop [24] (Trained on Celeb-DF, Video)	DefakeHop[b]	95.0%	90.6%	42.8k
Defakehop++ [22] (Trained on Celeb-DF, Frame)	Defakehop++[b]	95.4%	94.3%	238k
Defakehop++ [22] (Trained on Celeb-DF, Video)	Defakehop++[b]	**97.5%**	**96.7%**	238k

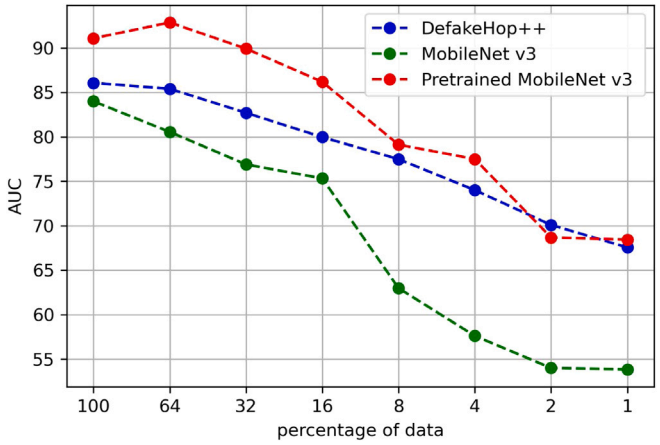

Fig. 10. Comparison of the detection AUC performance as a function of training data percentages of the DFDC dataset: (1) DefakeHop++, (2) MobileNet v3 with pre-training by ImageNet, and (3) MobileNet v3 without pre-training [22].

of undistorted reference images/videos, objective QA can be classified into three categories [105]: full-reference (FR), reduced-referenced (RR) and no-reference (NR). The last one is also known as blind QA (BQA). FR-QA metrics have achieved high consistency with human subjective evaluation. Many FR-QA methods have been well developed in the last two decades such as SSIM [168], FSIM [209] and VMAF [97]. RR-QA metrics only utilize features of reference images/videos for quality evaluation. In some application scenarios (e.g., image receivers), users cannot access reference images/videos so that NR-QA is the only choice. BQA methods attract growing attention in recent years.

The GL technology has been applied to BIQA (blind image quality assessment) as summarized in this subsection. Conventional BIQA methods has two steps: (1) extraction of quality-aware features and (2) adoption of a regression model for quality score prediction. Recently, as the amount of user generated images grows rapidly, DNN-based BIQA methods have been proposed with significant performance improvement [190]. DNN-based solutions often rely on huge pre-trained networks which are trained using large datasets. The large model size demands high computational complexity and memory requirement. In [121], a lightweight BIQA method, called GreenBIQA, was proposed to achieve high performance that is competitive with DNN-based solutions yet demands much less computing power and memory. To this end, for a video source, GreenBIQA can predict perceptual quality scores frame by frame in real time.

The performance of GreenBIQA is evaluated on four IQA datasets. CSIQ [91] and KADID-10K [104] are two synthetic-distortion datasets. They were created by applying multiple distortions of various levels to a set of reference images. LIVE-C [52] and KonIQ-10K [66] are two authentic-distortion datasets. They contain a wide range of distorted images captured by cameras. The performance metrics are the Pearson Linear Correlation Coefficient (PLCC) and the Spearman Rank Order Correlation Coefficient (SROCC). PLCC is used to measure the linear correlation between predicted scores and subjective quality scores. SROCC is adopted to measure the monotonicity between predicted and subjective quality scores. Each dataset is split into 80% for training and 20% for testing. Besides, 10% of training data is used for validation.

The performance of GreenBIQA is compared with four conventional and five DL-based methods in Table 5. The results of experiments run 10 times and the median PLCC and SROCC values are reported. The benchmarking methods can be classified into four categories.

- NSS-based handcrafted features: NIQE [126] and BRISQUE [125]
- Codebook-learning methods: CORNIA [195] and HOSA [186]
- DL methods without pre-training: BIECON [80] and WaDIQaM [14]
- DL methods with pre-training: PQR [198], DBCNN [204], and HyperIQA [158]

Methods in the first two categories are called conventional BIQA methods.

As shown in Table 5, GreenBIQA outperforms conventional BIQA methods in all four datasets. It also outperforms two DL methods without pre-training in both authentic-distortion datasets. As compared with DL methods with pretraining, GreenBIQA achieves competitive or even better performance in synthetic-distortion datasets (i.e., CSIQ and KADID-10K). For authentic-distortion datasets, GreenBIQA performs better than DL methods without pretraining, which demonstrates that it can generalize well to diverse distortions. There is performance gap between GreenBIQA and DL models with pre-training. Yet, these pre-trained models have much larger model sizes as a tradeoff.

Table 5
Comparison of SROCC/PLCC performance and model sizes of various blind image quality assessment methods on several IQA databases [121].

BIQA method	CSIQ		LIVE-C		KADID-10K		KonIQ-10K		Model size (MB)
	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC	SROCC	PLCC	
NIQE [126]	0.627	0.712	0.455	0.483	0.374	0.428	0.531	0.538	–
BRISQUE [125]	0.746	0.829	0.608	0.629	0.528	0.567	0.665	0.681	–
CORNIA [195]	0.678	0.776	0.632	0.661	0.516	0.558	0.780	0.795	7.4
HOSA [186]	0.741	0.823	0.661	0.675	0.618	0.653	0.805	0.813	0.23
BIECON [80]	0.815	0.823	0.595	0.613	–	–	0.618	0.651	35.2
WaDIQaM [14]	**0.955**	**0.973**	0.671	0.680	–	–	0.797	0.805	25.2
PQR [198]	0.872	0.901	0.857	0.882	–	–	0.880	0.884	235.9
DBCNN [204]	0.946	0.959	0.851	0.869	0.851	**0.856**	0.875	0.884	54.6
HyperIQA [158]	0.923	0.942	**0.859**	**0.882**	**0.852**	0.845	**0.906**	**0.917**	104.7
GreenBIQA [121]	0.925	0.936	0.673	0.689	0.847	0.848	0.812	0.834	1.9

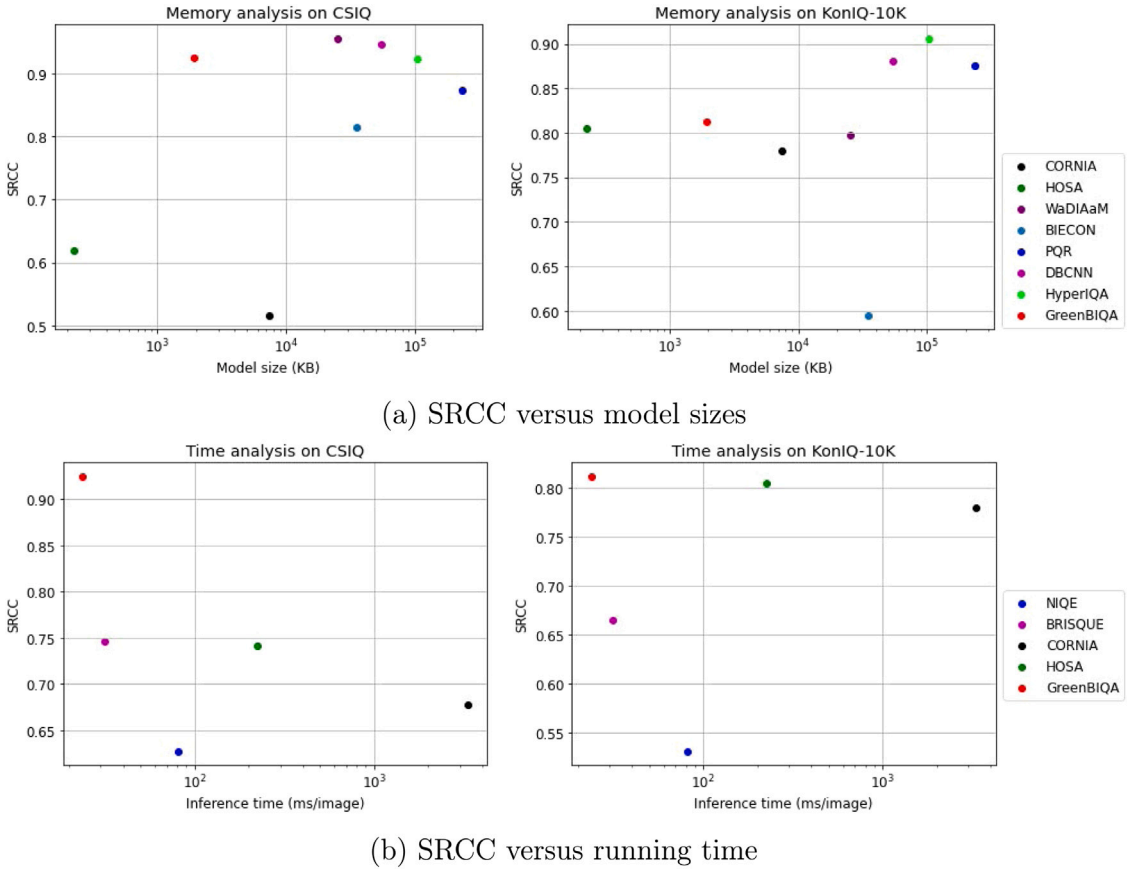

Fig. 11. Comparison of (a) SRCC vs. model sizes and (b) SRCC vs. running time of several BIQA methods on the CSIQ and KonIQ-10K datasets [121].

A small memory size is critical to real-world applications such as mobile phones. To illustrate the tradeoff between performance and memory sizes, we show the SROCC performance versus the memory size (under the assumption that one parameter takes one byte memory) of several benchmarking methods against CSIQ and KonIQ-10K in Fig. 11(a). We see that GreenBIQA can achieve similar performance on CSIQ and competitive performance on KonIQ-10K compared to DL methods with around 54x smaller model size.

For BIQA, another factor to consider at the client is running time in MOS prediction. GreenBIQA can process around 43 images per second with a single CPU. Thus, it meets the real-time processing requirement. We see from Fig. 11(b) that GreenBIQA has the smallest inference time among all benchmarking methods. It can be executed faster than conventional methods (i.e. NSS-based and codebook-learning methods) with significantly better SROCC performance on both synthetic- and authentic-distortion datasets. Note that we do not include DL methods in Fig. 11(b) since they are all implemented on GPUs.

5.3. Point clouds registration

GL technologies have been developed for point cloud classification, segmentation and registration tasks. Point cloud registration is used as an example to demonstrate GL techniques in this subsection. Given a pair of point cloud scans, registration attempts to find a rigid transformation that optimally aligns them. The quality of registration affects downstream tasks such as classification, segmentation, object detection, pose estimation and odometry. They are commonly encountered in autonomous driving, robotics, computer graphics, localization, AR/VR, etc. Present focus of point cloud registration research is to develop learning models that can handle challenges such as robustness to noise, varying point densities, outliers, occlusion and partial views.

To develop an accurate 3D correspondence solution, it is essential to have a good representation for points. Earlier solutions, e.g., [73,160], used 3D descriptors to capture local geometric properties such as

Fig. 12. Registration of seven point clouds from the ModelNet40 dataset using R-PointHop. The first row shows source point clouds (in black) and their target point clouds (in red), respectively. The second row shows registration results. Both the source and the target are complete point clouds in columns #1 and #2. The source in columns #3-#5 contains only part of the target. Both the source and the target are partial in columns #6 and #7 [76].

surface normals, tangents and curvatures. They are derived based on the first- or higher-order statistics of neighboring points, histogram, angles, etc. Recently, the trend is to learn features from an end-to-end optimization setting with DNNs. DL methods are often run on GPUs since they demand larger model sizes and longer training/inference time. DL-based point cloud processing is no exception. Here the intent is to look for green solutions that demand much less power consumption. This implies a smaller model size and less training/inference time. Yet, its performance should still be on par with that of DNNs.

A GL-based method, called R-PointHop, was proposed in [76] to meet the need. It learns representations in an unsupervised manner for point correspondence. Afterwards, the correspondences are used to find the 3D transformation for registration. R-PointHop provides a new way to extract representations that are invariant to point cloud rotation and translation. Rotation invariance is achieved by adopting a local reference frame (LRF) defined at each point. LRF enables R-PointHop to find robust point correspondences for larger rotation angles. Seven source (in black) and target (in red) point clouds and their registered results are illustrated in Fig. 12. The first two columns give registration results of full point clouds. columns 3, 4 and 5 provide results where only the source is partial. columns 6 and 7 show results where both the source and target are partial.

Registration of partial point clouds occurs in practical scenarios, where the source and the target have only a subset of points in common. R-PointHop can handle partial-to-partial registration which is often encountered in real world problems. A critical element in registering partial point clouds is to find correspondences between overlapping points. To test the capability of partial point clouds' registration, R-PointHop is trained on the ModelNet40 dataset [182] in the reported experiment. ModelNet40 is a synthetic dataset consisting of 40 categories of CAD models of common objects such as car, chair, table, airplane, person, etc. In this experiment, the initial point cloud has 1024 point, the partial point cloud has 768 points, and the overlapping region between the source and target is random and between 512 and 768. The results of partial-to-partial registration are given in Table 6 with two settings: (1) registration on unseen point clouds and (2) registration on unseen classes. R-PointHop gives the best performance in both.

In Table 6, ICP [10], Go-ICP [189] and FGR [216] are three model-free methods, while PointNetLK [5], DCP [172] and PR-Net [173] are supervised DL-based methods. The model size of R-PointHop is only 200 kB as compared with 630 kB of PointNetLK and 21.3 MB of DCP. The use of transformer makes the model size of DCP significantly larger. Although the model free methods are most favorable in terms of model sizes and training time, their registration performance is much worse. R-PointHop offers a good balance when all factors are considered.

Table 6
Comparison of the registration performance for various methods on partial point clouds [76].

Method	Registration errors on unseen objects					
	MSE(R)	RMSE(R)	MAE(R)	MSE(t)	RMSE(t)	MAE(t)
ICP [10]	1134.55	33.68	25.05	0.0856	0.2930	0.2500
Go-ICP [189]	195.99	13.99	3.17	0.0011	0.0330	0.0120
FGR [216]	126.29	11.24	2.83	0.0009	0.0300	0.0080
PointNetLK [5]	280.04	16.74	7.55	0.0020	0.0450	0.0250
DCP [172]	45.01	6.71	4.45	0.0007	0.0270	0.0200
PR-Net [173]	10.24	3.12	1.45	0.0003	0.0160	0.0100
R-PointHop [76]	**2.75**	**1.66**	**0.35**	**0.0002**	**0.0149**	**0.0008**
	Registration errors on unseen classes					
ICP [10]	1217.62	34.89	25.46	0.0860	0.293	0.251
Go-ICP [189]	157.07	12.53	2.94	0.0009	0.031	0.010
FGR [216]	98.64	9.93	1.95	0.0014	0.038	0.007
PointNetLK [5]	526.40	22.94	9.66	0.0037	0.061	0.033
DCP [172]	95.43	9.77	6.95	0.0010	0.034	0.025
PR-Net [173]	15.62	3.95	1.71	0.0003	0.017	0.011
R-PointHop [76]	**2.53**	**1.59**	**0.37**	**0.0002**	**0.0148**	**0.0008**

5.4. Graph node classification

Semi-supervised learning that exploits both labeled and unlabeled data is in high demand for real-world applications because of the expensive data labeling cost and the availability of a large number of unlabeled samples. For graph problems with few labels, the geometric or manifold structure of unlabeled data can be leveraged to improve the performance of classification, regression, and clustering algorithms. Graph convolutional networks (GCNs) have been widely accepted as the *de facto* tool in addressing semi-supervised graph learning problems [58,82,154]. GCN parameters are learned via label supervision through backpropagation [142]. The joint attributes encoding and propagation as smoothening regularization enable GCNs to yield superior performance.

There are however challenges in GCN-based solutions. First, GCNs demand a sufficient number of labeled samples for training. Instead of deriving label embeddings from graph regularization as done in [213,217], GCNs learns a series of projections from the embedding space to the label space. The transformations rely on ample labeled samples for supervision. The lack of sufficient labeled samples often makes training unstable (or even divergent). To address this problem, one may integrate other semi-supervised learning techniques (e.g., self- and co-training [98]) with GCN training or enhance the filter power to strengthen regularization [102]. Second, GCNs usually consist of two convolutional layers so that the information from a small neighborhood of each node is exploited [1,58,82,166]. The learning ability is handicapped by ignoring correlations from nodes of longer distance. Although increasing the number of layers could be a remedy, it leads to an oversmoothing problem and results in inseparable node representations [98,199]. Furthermore, a deeper model needs to train more parameters, which in turn requires even more labeled samples.

Two enhanced label propagation methods, called GraphHop [184] and GraphHop++ [183], was recently proposed. They are GL solutions

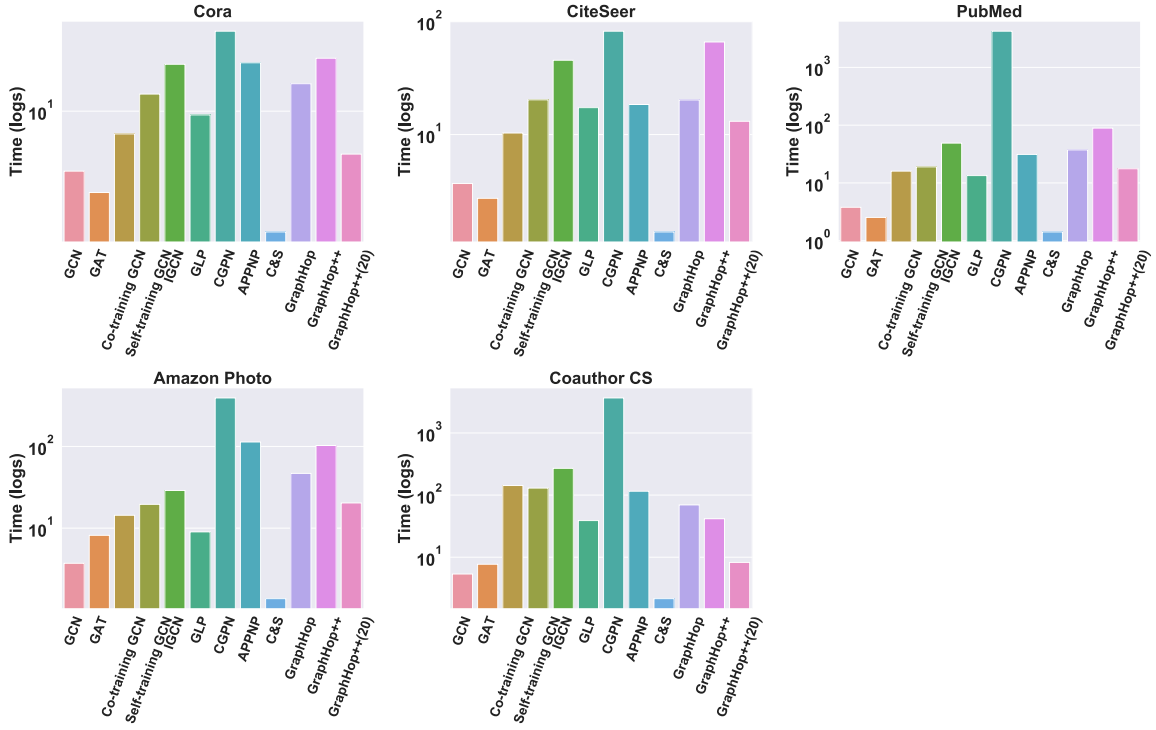

Fig. 13. Comparison of computational efficiency of different methods measured by log(second) for Cora, CiteSeer dataset, PubMed, Amazon Photo and Coauthor C&S datasets, where the label rate is 20 labeled samples per class and GraphHop++(20) is the result of GraphHop++ with 20 alternate optimization rounds.

Table 7
Test accuracy for the Coauthor CS dataset with extremely low label rates measured by "the mean accuracy (%) ± standard deviation". The highest mean accuracy is in **bold** while the second and third ones are underlined [183].

Coauthor CS						
# of labels per class	1	2	4	8	16	20
GCN [82]	64.42 ± 2.14	74.07 ± 1.09	82.62 ± 0.74	87.88 ± 0.55	89.86 ± 0.15	90.19 ± 0.17
GAT [166]	72.81 ± 2.01	80.76 ± 1.25	85.37 ± 0.94	88.06 ± 0.61	89.83 ± 0.16	89.82 ± 0.17
LP [213]	52.63 ± 0.00	59.34 ± 0.00	61.77 ± 0.00	68.60 ± 0.00	73.50 ± 0.00	74.65 ± 0.00
APPNP [83]	71.06 ± 17.48	86.18 ± 0.56	86.57 ± 0.58	89.66 ± 0.30	90.65 ± 0.21	90.55±0.17
C&S [70]	35.63 ± 13.20	67.11 ± 3.50	78.57 ± 1.22	83.92 ± 1.15	87.09 ± 0.95	87.97 ± 0.56
Co-training GCN [98]	75.78 ± 1.00	**86.94 ± 0.70**	87.40 ± 0.78	88.81 ± 0.32	89.22 ± 0.42	89.01 ± 0.55
Self-training GCN [98]	69.69 ± 3.27	82.79 ± 4.10	87.62 ± 1.49	88.82 ± 1.00	89.53 ± 0.54	89.07 ± 0.70
IGCN [102]	62.16 ± 2.81	59.56 ± 2.91	65.82 ± 4.86	86.57 ± 0.78	87.76 ± 0.74	88.10 ± 0.52
GLP [102]	43.56 ± 7.06	50.74 ± 7.55	46.61 ± 9.85	76.61 ± 3.39	81.75 ± 2.81	82.43 ± 3.31
CGPN [167]	67.66 ± 0.00	64.49 ± 0.00	71.00 ± 0.00	77.09 ± 0.00	78.75 ± 0.00	79.71 ± 0.00
Graphhop [184]	65.03 ± 0.01	77.59 ± 3.17	83.79 ± 1.13	86.69 ± 0.62	89.47 ± 0.38	89.84 ± 0.00
Graphhop++ [183]	**82.46 ± 1.28**	86.37 ± 0.37	**88.45 ± 0.35**	**89.87 ± 0.48**	**90.69 ± 0.13**	**90.87 ± 0.06**

to graph learning problems. They achieve state-of-the-art performance as compared with GCN-based methods [82,98,166] and other classical propagation-based algorithms [131,174,213]. They perform particularly well at extremely low label rates. To give an example, the performance of GraphHop and GraphHop++ on the Coauthor CS graph dataset is shown in Table 7. Each column in the table presents the classification accuracy (%) of all benchmarking methods on test data under a specific label rate.

Overall, GraphHop++ has the best performance in most cases. In particular, for cases with extremely limited labels, GraphHop++ outperforms other methods by a large margin. This is because GCNs are difficult to train with a small number of labels. The lack of supervision prevents them from learning the transformation from the embedding space to the label space with nonlinear activation at each layer. Instead, GraphHop++ applies regularization to the label space. It is more effective since it relieves the burden of learning the transformation. A small number of labeled nodes also restricts the efficacy of message passing on graphs of GCN-based methods.

In general, only two convolutional layers are adopted by GCN methods. It means that only messages in the two-hop neighborhood of each labeled node can be supervised. Apparently, the two-hop neighborhood of a limited number of labeled nodes cannot cover the whole network effectively. Thus, a large number of nodes do not have supervised training from labels. It explains the inferior performance of all GCN methods. Some label-efficient GCN methods try to alleviate this problem by exploiting pseudo-labels as supervision (e.g., self-training GCN) or improving message passing capability (e.g., IGCN and CGPN). Yet, they are still handicapped by deficient message passing in the graph convolutional layers. In contrast, label propagation methods (e.g., APPNP, GraphHop and GraphHop++) achieve better performance in most datasets at low label rates since the messages from labeled nodes can pass over a longer distance on graphs through an iterative process. It is worth noting that, although classical LP and C&S are also propagation-based, their performance is poorer for the following reasons. Classical LP fails to encode the rich node attribute information in model learning. C&S, which was originally designed for supervised learning, is affected by the lack of labeled samples.

The running time performances of GraphHop++ and its several benchmarking methods are compared in Fig. 13. It shows the results at a label rate of 20 labeled samples per class with respect to five

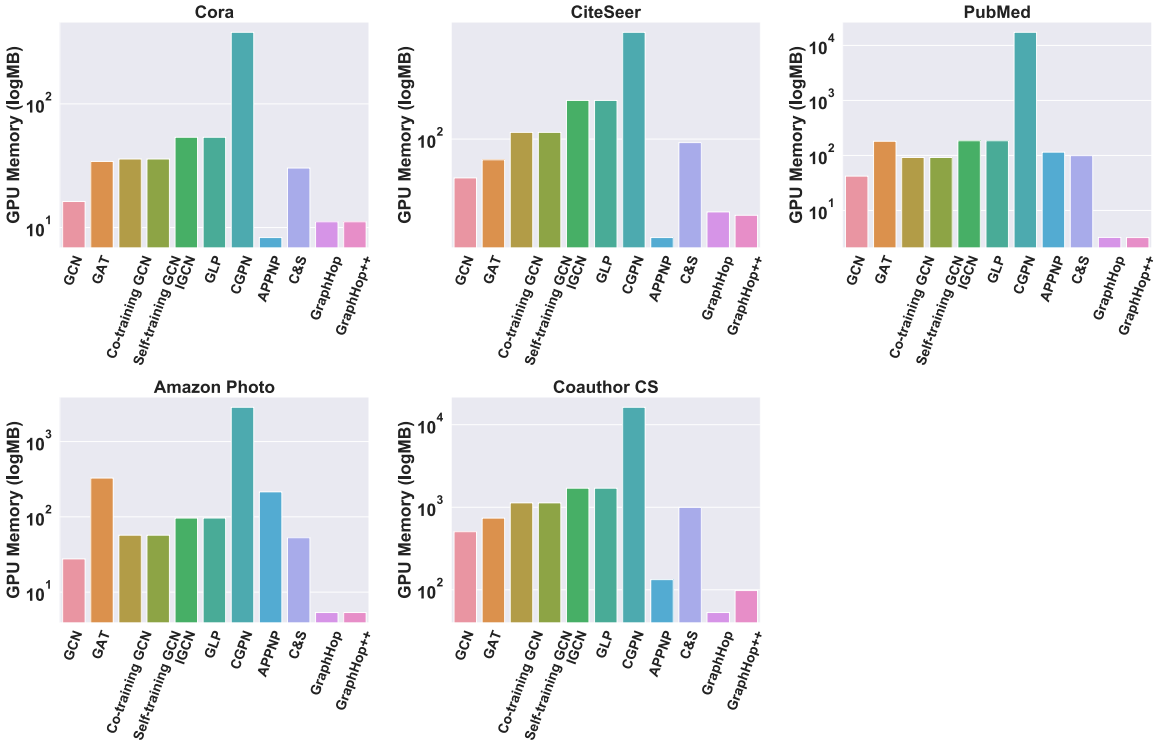

Fig. 14. Comparison of GPU memory usages of different methods measured by log(Mega Bytes) for Cora, CiteSeer dataset, PubMed, Amazon Photo and Coauthor C&S datasets, where the label rate is 20 labeled samples per class.

datasets. For GraphHop++, results are reported with 100 rounds and 20 rounds, respectively, since it can converge within 20 rounds. As shown in Fig. 13, GraphHop++(20) can achieve an average running time. The most time-consuming part of GraphHop++ is the training of the LR classifiers. It demands multiple loops of all data samples. However, we find that incorporating classifiers is essential to the superior performance of GraphHop++ under limited label rates. In other words, GraphHop++ trades time for effectiveness in the case of extremely low label rates. Among all datasets, GraphHop++ achieves the lowest running time for Coauthor C&S due to its uncomplicated design as LP.

Next, the GPU memory usage of GraphHop++ and its several benchmarking methods is compared in Fig. 14, where the values of GraphHop++ are taken from the same experiment of running time comparison as given in Fig. 13. It is measured by `torch.cuda.max_memory_allocated()` for PyTorch and `tf.contrib.memory_stats.MaxBytesInUse()` for TensorFlow. Generally, GraphHop and GraphHop++ achieves the lowest GPU memory usage among all benchmarking methods. This is because GraphHop and GraphHop++ allow minibatch training. The only parameters to be stored in the GPU are classifier parameters and one minibatch of data. In contrast, since the number of neighborhood nodes increases exponentially regarding deeper layers, GCN methods cannot simply conduct minibatch training. Their GPU memory consumption increases significantly.

5.5. Knowledge graph completion

Knowledge graph completion (KGC) aims to discover missing relationships between entities in knowledge graphs (KGs). Knowledge graph embedding (KGE) techniques offer state-of-the-art solutions to this problem. In KGE models, embeddings for entities and relations are determined by optimizing triple score functions. That is, scores among positive triples are maximized while those among negative triples are minimized. The entity and relation embeddings are stored as model parameters. The number of parameters in a KGE model is linearly proportional to the embedding dimension and the number of entities and relations in KGs, i.e. $O((|E| + |R|)d)$, where $|E|$ is the number of entities, $|R|$ is the number of relations, and d is the embedding dimension.

Most prior KGC methods focus on learning representations for entities and relations. A higher-dimensional embedding space is usually required for better reasoning capability, which leads to a larger model size. It hinders the applicability of KGE methods to real-world problems (e.g., large-scale KGs or mobile/edge computing). It is critical to obtain a KGC method that has good reasoning capability in low dimensions. A lightweight modularized KGC solution, called GreenKGC, was proposed in [169] to meet the requirement.

GreenKGC consists of three modules: (1) representation learning, (2) feature pruning, and (3) decision learning. Each module is optimized by itself. In Module 1, a KGE method, called the baseline method, is leveraged to learn high-dimensional entity and relation representations. In Module 2, a feature pruning process is applied to the high-dimensional entity and relation representations to yield discriminant low-dimensional features for triples. Since logical patterns for different relations vary a lot, triples are partitioned into multiple relation groups of homogeneous logical patterns. As a result, relations in the same group can share the same low-dimensional features. It helps improve the performance and reduces the model size. In Module 3, a binary classifier is trained for each relation group so that it predicts triple's score in inference. The predicted score is a soft value between 0 and 1, indicating the likelihood of a certain triple.

We examine the performance of GreenKGC on three link prediction datasets. They are FB15k-237 [13,161], WN18RR [13,38] and YAGO3-10 [38]. FB15k-237 is a subset of Freebase [12] on real-world relationships. WN18RR is a subset of WordNet [122] on lexical relationships between word senses. YAGO3-10 is a subset of YAGO3 [117] on person's attributes. For link prediction, the goal is to predict the missing entity given a query triple, i.e. $(h, r, ?)$ or $(?, r, t)$. The correct entity should be ranked higher than other candidates. To this end, several common ranking metrics are MRR (Mean Reciprocal Rank) and Hits@k (k = 1, 3, 10). Following the convention in [13], we adopt the

Table 8
Performance comparison on link prediction, where the performance gain (or loss) in terms of percentages with an up (or down) arrow and the ratio of the model size are shown within the parentheses against those of respective 500D models [169].

Baseline	Dim.	FB15k-237			WN18RR			YAGO3-10		
		MRR	H@1	#P (M)	MRR	H@1	#P(M)	MRR	H@1	#P (M)
TransE	500	0.325	0.228	7.40	0.223	0.013	20.50	0.416	0.271	61.60
	100	0.274 ↓ 15.7%	0.186 ↓ 18.5%	1.48 (0.2x)	0.200 ↓ 10.3%	0.009 ↓ 30.8%	4.10 (0.2x)	0.377 ↓ 2.0%	0.269 ↑ 0.7%	12.32 (0.2x)
	GreenKGC 100	0.316 ↓ 2.8%	0.232 ↑ 1.8%	2.04 (0.28x)	0.407 ↑ 82.5%	0.361 ↑ 176.9%	4.30 (0.2x)	0.455 ↑ 9.4%	0.358 ↑ 32.1%	12.88 (0.21x)
RotatE	500	0.333	0.237	14.66	0.475	0.427	40.95	0.495	0.402	123.20
	100	0.296 ↓ 11.1%	0.207 ↓ 12.7%	2.93 (0.2x)	0.452 ↓ 3.8%	0.427 –	8.19 (0.2x)	0.482 ↓ 2.0%	0.393 ↑ 0.7%	24.64 (0.21x)
	GreenKGC 100	0.348 ↑ 4.5%	0.266 ↑ 12.2%	3.49 (0.24x)	0.476 ↑ 0.2%	0.430 ↑ 0.7%	8.75 (0.21x)	0.485 ↓ 2.0%	0.405 ↑ 0.7%	25.20 (0.21x)

Table 9
Performance comparison on link prediction in low dimensions ($d = 32$), where the best and the second-best numbers are in bold and with an underbar, respectively [169].

Model	FB15k-237				WN18RR				YAGO3-10			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Knowledge Graph Embedding Methods												
TransE [13]	0.270	0.177	0.303	0.457	0.221	0.020	0.388	0.517	0.324	0.221	0.374	0.524
RotatE [159]	0.290	0.208	0.316	0.458	0.387	0.330	0.417	0.491	0.419	0.321	0.475	0.607
Classification-based KGC Methods												
ConvKB [129]	0.232	0.157	0.255	0.377	0.346	0.300	0.374	0.422	0.311	0.194	0.368	0.526
ConvE [38]	0.282	0.201	0.309	0.440	0.405	0.377	0.412	0.453	0.361	0.260	0.396	0.559
Low-dimensional KGE Methods												
AttH [18]	0.324	0.236	0.354	0.501	0.466	0.419	0.484	0.551	0.397	0.310	0.437	0.566
DualDE [219]	0.306	0.216	0.338	0.489	**0.468**	**0.419**	**0.486**	**0.560**	–	–	–	–
GreenKGC [169]	**0.326**	**0.248**	**0.351**	**0.479**	0.411	0.367	0.430	0.491	**0.453**	**0.361**	**0.509**	**0.629**

filtered setting, where all entities serve as candidates except for the ones that have been seen in training, validation, or testing sets.

One novel characteristic of GreenKGC is that it prunes a high-dimensional KGE to low-dimensional triple features and make prediction with a binary classifier as a powerful decoder. The capability of GreenKGC in saving the number of parameters and maintaining the performance by pruning original 500D KGE to 100D triple features is shown in Table 8. As shown in the table, GreenKGC can achieve comparable or even better performance with around 4 times smaller model size. Especially, its Hits@1 metric is improved in most datasets. Furthermore, negative sampling adopted in classifier training can correct some failed cases in original KGE methods. For all datasets, 100D GreenKGC could generate better results than 100D KGE models.

The performance of GreenKGC with KGE, classification-based, and low-dimensional KGC methods of 32D is compared in Table 9. KGE methods, such as TransE and RotatE, cannot yield good performance in low dimensions due to over-simplified score functions. Classification-based methods outperform KGE methods as they adopt DNNs as complex decoders. Low-dimensional KGC methods are specifically designed for low dimensions. They provide state-of-the-art KGC solutions in low dimensions. Yet, GreenKGC outperforms all of them in FB15k-237 and YAGO3-10. For WN18RR, KGE methods perform poorly. Since GreenKGC relies on KGE as its baseline, its performance is affected. In spite of this fact, GreenKGC still outperforms all KGE and classification-based methods for WN18RR.

6. Future outlook

6.1. Robustness

Today's DL networks are dominated by feedforward neural networks, i.e., information moves in only one forward direction during the inference (i.e., from input nodes, through hidden nodes, and to output nodes). They do not have cycles or loops. Their filter weights are obtained by end-to-end optimization using the backpropagation algorithm. One drawback of feedforward neural networks is that it is easy to add small perturbations to the input to fool DL. This is known as the adversarial attack [3]. Adversarial training [116] has been proposed to improve the robustness of DL networks with clean data albeit at the expense of poorer performance.

There is little research on the robustness of GL systems. To attack an arbitrary input to a GL system, one needs to modify its representation, the feature learner (i.e., DFT) or the classifier. The latter two are related to statistics of training data. They cannot be controlled by a single input. With regard the first, we argue that adversarial attacks designed for DL systems would not harm GL systems. This is because GL systems adopt signal transforms to reduce representation's dimension, where signal transforms are obtained by statistics of training samples. Small perturbations to the input obtained by adversarial attacks to DL systems are filtered out by signal transforms. Furthermore, the ensemble learning technique can be leveraged by GL to enhance its accuracy since each individual GL decision is lightweight.

Apparently, if the distributions of sampled training and test data are sufficiently different, the trained GL systems will not predict well. Yet, this is a common concern to all ML systems. Generally speaking, it is an open research problem to conduct adversarial attacks that are specifically effective against GL systems. In this regard, it is also not clear how to design more resilient GL systems.

Building large AI/ML datasets is a laborious task. Once data are collected, they are partitioned into training and testing sets in accord with a certain ratio. However, the methodology for collecting fewer yet representative data for certain applications is rarely addressed in the current AI/ML literature. As a result, most datasets tend to be built in a brute force, ad hoc fashion. Future work in this area needs to address how to build relevant datasets for ML systems.

6.2. Trust and risk assessment

The application of blackbox ML models to high stakes' decisions such as those involved in healthcare, legal judgment, critical infrastructure management has been challenged by researchers, e.g., [7,135,141]. Today's evaluations conducted on DL models are based on a certain split of training and test data, which are collected using the same protocol. Such an evaluation methodology is likely to be too simplistic to be trustworthy. Also, conclusions drawn from a set of input–output relationships can be misleading and counter-intuitive. It is unclear how to quantify prediction performance for previously unseen data. It is difficult to provide a logical description of the decision process to experts in the field to earn their trust.

Along this line, the debate between Yann LeCunn and Gary Marcus on whether AI needs more innate machinary is inspiring.[1] If a learning system does not have a semantic structure, it does not have properties, such as compositionality, which are required for interpretation. However, it was stated in [46,114,115] that, as long as the attributes on which the network is being trained on along with their respective weights are comprehensible to the user, this is sufficient for interpretability. It is this fact that engenders trust in users, and not the ability to articulate the reasoning chain or strive for full model transparency.

For GL to be well-accepted by society, especially for problems with dire consequences for erroneous interpretations, it is essential to have human interpretable results backed by scientific rigor. Since GL is rooted in both statistics and optimization, it offers probabilistic models with performance guarantees, which also enable risk assessment. The risk of GL's decisions can be attributed to three factors: (1) sampling bias and fairness [120,196]; (2) labeling errors [132]; (3) analysis errors. Some domains have tighter quality assurance and error control such as new drug development and medical treatment procedure. Others may have lower standards. For example, autonomous driving should belong to the high stakes decision category. Due to the dominance of DL in the field of computer vision and the blackbox nature of DL, we do not see significant error analysis on the methodology. This in turn affects the standards on the first two factors. Furthermore, while GL is still in its infancy, it is important to include risk analysis in this emerging discipline. GL should also be tolerant to labeling errors in the training process. Finally, it should be able to estimate errors for different data types, knowing that inliers tend be more robust while outliers tend to be less reliable.

6.3. Relationship between human cognition and explainable AI/ML

An important discussion today revolves around explainable AI for DNNs and counterfactual examples [47]. The basic idea is this: counterfactual inputs to produce some desired outputs are first created and "reading of the hidden units would then, presumably, explain why the network produced some other output". However, Browne and Swift [17] showed that "counterfactual examples are just slightly more meaningful adversarial examples, which are examples that are produced by performing small and unobservable perturbations on inputs causing the network to misclassify them with very high confidence". Moreover, it is claimed that counterfactual examples explain what some feature should have been to get the right prediction but without opening the black box, i.e., without explaining how the algorithm works. It was argued in [17] that counterfactual examples do not provide a solution to explainability, and that there can be no explanation without semantics. Also, the property of compositionality, which is irreversible in DNNs, poses a real problem to explanation generation. Simply put:

- There can be no explanation without semantics [17].
- There can be no semantics without invertible compositionality [45].
- Compositionality in DNNs is irreversible [45].

In the light of the foregoing, it appears that DL models that lack a semantic structure are not capable of generating any meaningful explanations. The challenge applies to the data-driven ML discipline in general. GL is no exception.

The relationship between human cognition and explainable AI/ML remains to be an open problem. In a different vein, past researchers have shown that full explanation capability is not necessary in many cases to engender trust in neural network results so long as the key attributes being considered and their relative weighting can be transparently communicated to users [46,114,115].

6.4. Generative models

Given a family of images of similar characteristics, one ML problem is to generate more images that share the same characteristics. DL has achieved a significant amount of progress in this area. DL generative models can be categorized into two types: non-adversarial and adversarial ones. The former includes variational auto-encoders (VAEs) [81], GLO [11], IMLE [101], GLANN [65], and diffusion models [61] while the latter includes the original GAN [53], LSGAN [118], WGAN [6], StyleGAN [78], etc. DL generative models take an arbitrary white Gaussian random vector as their input and generate an image of the target family via a DL generator (or decoder). All outputs at intermediate layers are latent variables. It is difficult to give them a physical meaning.

There are preliminary efforts in the development of GL generative models, e.g., NITES [95], TGHop [96], Pager [8], GENHOP [94]. GL generative models contains two modules:

- Module 1: Fine-to-Coarse Image Analysis
  The dimension of color images of $N \times N$ pixels is $3N^2$. As N becomes large, the diversity is too high to model with a reasonable mathematical model. To reduce the diversity, one straightforward idea is to partition one large image into non-overlapping smaller blocks, each of which has a significantly lower dimension. However, these blocks cannot be generated independently since they are correlated with each other. To take the block correlation into account, one can down-sample fine-resolution images to coarse-resolution ones and compute the residual image between interpolated coarse-resolution images and fine-resolution images. Then, to synthesize the fine-resolution images, the GL generative model only needs to model residual images as well as the coarse-resolution images. This process can be done recursively to get residual images at multiple resolutions as well as the coarsest-resolution images. This pipeline is called the fine-to-coarse image analysis.
- Module 2: Coarse-to-Fine Image Synthesis
  Usually, the space formed by coarsest-resolution images is called the core subspace. A sample in the core subspace corresponds to a coarsest-resolution image. The core subspace was modeled using the independent component analysis (ICA) in [96] and the Gaussian mixture model (GMM) in [8]. After a sample is generated in the core subspace, we need to add more details to it. To achieve this objective, we partition coarse-resolution images into smaller blocks and use their content as the condition for detail generation. In principle, these conditional probabilities can be learned from Module 1. Then, one can generate high-resolution images through the core generation as well as a sequence of detail generations.

Although the high-level idea of GL generation can be described by words easily above, it demands good mathematical tools, mature programming skills and strong engineering insights to implement a high-performance GL generation system. A recent paper [54] points to a non-DL single-image generative solution and its applications in retargeting, conditional inpainting, collage, etc.

[1] Readers can find the information from a YouTube video and a Quora blog.

6.5. Structures of high-dimensional representation space

The representation space of GL is derived by source data statistics (e.g., image pixel correlations) and a set of features that minimizes a cost function defined by labels (or tasks). The process from the source space to the representation space is deterministic and the resulting representation is unique for the same representation. This is different from the case of DL. Even for the same network, DL's embeddings are not uniquely defined. They depend on the initialization of the filter weights and the order of input samples in the stochastic gradient descent optimization process. Since the representation space of a GL system can be uniquely specified, its geometrical analysis will facilitate the development of GL-based learning algorithms.

Geometrical analysis of low-dimensional feature spaces was exploited in [108] to design the multilayer perceptrons for cases. Analysis of high-dimensional representation spaces remain to be a challenge. The advancement in this area will contribute to development of GL technologies. For example, we need to model the distributions of representations in the core and residual subspaces as discussed in Section 6.4. Although ICA and GMM have been used as generic modeling tools, they could be fine-tuned to match distribution's geometry to yield simpler yet more powerful generative models.

Another application scenario is the separation of inliers and outliers in the representation space. The number of inliers is significantly higher than that of outliers. The learning of inliers can be done in the weakly supervised fashion without data augmentation since the sample density is higher. On the other hand, the learning of outliers should be done in the heavily supervised fashion with data augmentation since the sample density is lower. To this end, some preliminary research on this topic has been conducted in [188]. Generally speaking, inliers are easy samples while outliers are hard samples. The performance of ML algorithms is largely determined by the ratio of easy and hard samples. It is difficult to separate inliers and outliers in DL networks in the training pipeline. Both have to be included in the backpropagation process. In contrast, by separating inliers and outliers, we can zoom in outliers, conduct data augmentation, and design a tailored feature extractor and classifier in GL systems.

6.6. Object detection, classification and semantic segmentation

A few large-scale datasets have been built for object detection, classification, and semantic segmentation problems. These include:

- Object Detection Datasets: Pascal VOC [43], ImageNet [37], MS CoCo [106], etc.
- Object Classification Datasets: ImageNet [37], Places [215], etc.
- Semantic Segmentation Datasets: Pascal VOC [43], Cityscapes [32], etc.

These problems have been well solved by DL methods with quite a few survey papers on these topics, e.g., [197] for scene classification, [211] for object detection, and [123] for semantic segmentation. One common challenge of the three closely related problems is the multi-scale-size and multi-aspect-ratio of underlying objects in given input images. For DL-based solutions, object proposals of multiple sizes and aspect ratios are adopted, e.g. in the YOLO object detection system [136].

The concept of flexible object proposals is challenging to implement in the GL setting. Instead, we may revisit an old framework known as "feature pyramids + sliding windows". This methodology was intensively studied before 2014 [220]. To derive a GL solution, we may handle objects of various sizes through feature pyramids so as to have a rough idea about object's location, size and class based on detected salient regions. Then, we can exploit prior statistics on the roughly estimated object to finetune the object proposal and/or refine the classification result. Once a rough location of an object is detected, its precise segmentation (i.e., pixel-based classification) can be solved more easily.

There are several constraints in classical object detection, classification and segmentation solutions before the DL era. For example, it is often to use histograms of oriented gradients (HoG) [35] as the local representation and the deformable part model [44] as the global representation. Both of them are neither expressive nor general enough in characterizing a large number of object classes and appearance variations. Their performance does not scale well with the dataset size due to the limited power of the associated representations.

A possible GL solution is outlined below. First, we can apply a sliding window of a fixed size to different levels of a pyramid, which is created by the Lanczos down-sampling operation, and apply IPHop-II [188] to each block to learn a more effective local representation. The main purpose is to capture distinctive blocks that are associated with certain object classes. Next, we can assemble neighboring blocks of the same predicted class, analyze their relation, and build a global representation. For example, to detect a horse, we can see the whole horse in a sliding window at a higher level of a pyramid and observe parts of the same horse in multiple sliding windows at lower levels of the same location. The local representation can be 3D Saab coefficients (2D from the space and 1D from the scale). The above sketched concept remains to be tried.

6.7. Multi-domain data integration

ML with multi-domain data such as images, videos, point clouds, text, speech, audios, graphs is an emerging trend that can be expected to provide significant dividends in the future. It is straightforward to apply GL to a single data domain for representation learning. Non-image-domain applications of GL can be found in some exploratory work — videos [175], point clouds [74,75,77,110,201,202,205,207], texts [176,177], graphs [183,184] and knowledge graphs [51,170]. Generally, we can concatenate representation vectors learned from each domain into "hybrid" representation vectors for multi-domain input samples. The main challenge is that the dimension of hybrid representation vectors is extremely high. The training data will be too sparse in the representation space to yield meaningful learning results. We need some mechanism to zoom into a lower dimensional space. To this end, we may leverage co-occurrence (e.g., joint or conditional probabilities), priors, constraints.

Co-occurrence of representations and labels has been exploited in attention localization in [193] as explained. We use the problem of dog/cat image classification to illustrate it. Each image can only have one label (i.e., dog or cat) despite the existence of diverse backgrounds (e.g., indoor or outdoor). We partition input images into local patches, assign the source image label to resulting patches, and group patches of visual similarity into clusters. There are clusters of background regions (say, wall, window, furniture, floor, sky, grass, street, etc.). They are typically shared by dog and cat images. As a result, background patches with dog and cat labels co-exist in these clusters. Such clusters have a higher entropy value. In contrast, clusters contain regions specific to dog/cat images (e.g., their facial regions) are dominated by one class. Such clusters have a lower entropy value. Patches in low entropy clusters can be chosen as attention regions. This idea could be extended to object tracking and video object segmentation, and further exploration along this direction could be fruitful.

To train an automatic GL-based image/video captioning system may leverage the co-occurrence of multi-domain representations and labels. First, we need build a link between simple visual data units (e.g., objects and scenes) and their texts. This training can be conducted based on the ImageNet or the Places dataset [215]. The scene recognition could be challenging due to the occlusion of the foreground objects. On the other hand, the correlation between an object and its environment can be exploited. For example, a giraffe is most likely to be in the wild or zoo. In the inference stage, we can first identify main objects

and the background in an input image using their image representations. Next, we need to describe interactions between objects and/or interactions between object/environment. They could be learned from the image/video captioning training datasets. Interaction in the text domain is easier to capture. One way is to build a dependency parsing tree [176,177] whose child nodes contain main objects and/or background. Various interactions between child nodes define "actions". We group actions into similar types, and go back to the image domain to find similarities among their associated images with an objective to connect image- and text-based action representations.

6.8. Lightweight DL versus GL

Development of lightweight DL networks of smaller sizes and faster computation is an active research topic in the DL community. One approach is to compress large DL networks [4,30,31,63,112,127,187] to smaller ones. This can be achieved by quantization and/or model pruning. The former lowers the precision of model parameters while the latter reduces the number of model parameters. They can be applied on a given model one after the other. Another approach is to design smaller DL architectures targeting at mobile/edge computing platforms from scratch. Several well-known networks belong to this class, e.g., MobileNets [67], ShuffleNet [210] and Light CNN [180]. Usually, the lightweight DL networks trade degraded performance for less memory and computation requirement.

Although little performance benchmarking between GL models and lightweight DL models has been conducted, we do find one example in [22] for deepfake video detection. The work compares a GL system called DefakeHop++ and MobileNet v3. The numbers of model parameters of DefakeHop++ and MobileNet v3 are 238k and 1.5M, respectively. Furthermore, DefakeHop++ outperforms MobileNet V3 (without pre-training) in detection performance. The performance gap is higher as the percentages of training samples go lower. This preliminary study demonstrates the enormous potential of GL models over simplified DL models. More extensive evaluations of DL and GL comparison in a wider range of settings and applications are needed to draw more definite conclusion.

7. Conclusion

This overview paper introduced a novel GL methodology with illustrative examples. GL adopts a modularized design, in which each module is optimized independently for implementation efficiency and logical transparency. The initial research on GL was conducted to provide a better understanding of CNNs. The study led to the design of an interpretable FF-CNN. FF-CNN derives network parameters of the current layer based on data statistics from the output of the previous layer in a one-pass feedforward manner. Filter weights in the convolutional layers are conveniently computed through the Saab and c/w Saab transforms without any labels. Filter weights in the FC layers are calculated with labels or pseudo-labels based on linear least squared regression. To reduce the performance gap between FF-CNN and the classical BP-CNN (backpropagation-based CNN), ensemble learning is incorporated in the FF design, leading to more powerful ML systems such as PixelHop, PixelHop++ and IPHop. Recently, discriminant and relevant feature selection tools and new classifiers/regressors that leverage both feature and decision fusions have been devised to make GL a more mature technology.

The authors hope that this overview paper will garner the attention and interest of researchers in this emerging learning paradigm that is critical to the sustainability of our living environment and the reliability and consistency of high stakes decisions. The differences between GL and DL in terms of computational complexity (or carbon footprint), storage requirement (or model size), and logical transparency were compared using several examples throughout the paper. We have seen a few successful applications of GL with performance comparable with state-of-the-art DL solutions demanding significantly less resources.

Admittedly, while GL is still in its infancy, it has several key characteristics that make it a productive area of research. Several problems that are well solved by DL do not have a corresponding GL solution yet. We believe that it is worth re-examining these problems from the GL angle. Such initiatives can be expected to lead to fruitful results. In this regard, several future research and development topics pertaining to GL were discussed in Section 6. The opportunities are plentiful. Commitment to further advancement of GL is not only intellectually rewarding but also practically meaningful.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The authors acknowledge the Center for Advanced Research Computing (CARC) at the University of Southern California, United States for providing computing resources that have contributed to the research results reported in this work. The first author would also like to thank his former and current Ph.D. students in the Media Communications Laboratory at the University of Southern California for their collaborative efforts in exploring green learning technologies. Special thanks go to Hao Xu, Yueru Chen, Yijing Yang, Min Zhang, Pranav Kadam, Mozhdeh Rouhsedaghat, Hong-Shuo Chen, Ruiyuan Lin, Hongyu Fu, Xuejing Lei, Zohreh Azizi, Yifan Wang, Tian Xie, Zhiruo Zhou, Wei Wang, Yao Zhu, Kaitai Zhang, Bin Wang, Chengwei Wei, Yun Cheng Wang, Xiou Ge, Siyang Li, Jiali Duan, Vasileios Magouliantis, Zhanxuan Mei, Ganning Zhao, and Xinyu Wang.

Appendix. Glossary of terms

AC	Alternate Current
AI	Artificial Intelligence
CNN	Convolutional Neural Network
c/w Saab	Channel-wise Saab
DC	Direct Current
DCT	Discrete Cosine Transform
DFT	Discriminant Feature Test
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
FC	Fully Connected
FF-CNN	Feedforward-designed Convolutional Neural Network
GAN	Generative Adversarial Network
GCN	Graph Convolutional Network
GL	Green Learning
KLT	Karhunen-Loéve transform
LAG	Label-Assisted Regression
LDA	Linear Discriminant Analysis
ML	Machine Learning
MLP	Multi-Layer Perceptron
MSE	Mean-Squared Error

PCA	Principal Component Analysis
RF	Random Forest
RFT	Relevant Feature Test
RNN	Recurrent Neural Network
Saab	Successive Approximation with Adjusted Bias
Saab	Successive Approximation with Augmented Kernels
SLM	Subspace Learning Machine
SLR	Subspace Learning Regressor/Regression
SSL	Successive Subspace Learning
SVM	Support Vector Machine
SVR	Support Vector Regressor/Regression

References

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, Aram Galstyan, Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing, in: International Conference on Machine Learning, PMLR, 2019, pp. 21–29.

[2] Darius Afchar, Vincent Nozick, Junichi Yamagishi, Isao Echizen, Mesonet: a compact facial video forgery detection network, in: 2018 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, 2018, pp. 1–7.

[3] Naveed Akhtar, Ajmal Mian, Threat of adversarial attacks on deep learning in computer vision: A survey, IEEE Access 6 (2018) 14410–14430.

[4] Jose M. Alvarez, Mathieu Salzmann, Compression-aware training of deep networks, Adv. Neural Inf. Process. Syst. 30 (2017).

[5] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, Simon Lucey, PointNetLK: Robust & efficient point cloud registration using PointNet, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7163–7172.

[6] Martin Arjovsky, Soumith Chintala, Léon Bottou, Wasserstein generative adversarial networks, in: International Conference on Machine Learning, 2017, pp. 214–223.

[7] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al., Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Inf. Fusion 58 (2020) 82–115.

[8] Zohreh Azizi, C.-C. Jay Kuo, PAGER: Progressive attribute-guided extendable robust image generation, 2022, arXiv preprint arXiv:2206.00162.

[9] Zohreh Azizi, Xuejing Lei, C.-C. Jay Kuo, Noise-aware texture-preserving low-light enhancement, in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), IEEE, 2020, pp. 443–446.

[10] Paul J. Besl, Neil D. McKay, Method for registration of 3-D shapes, in: Sensor Fusion IV: Control Paradigms and Data Structures, Vol. 1611, International Society for Optics and Photonics, 1992, pp. 586–606.

[11] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, Arthur Szlam, Optimizing the latent space of generative networks, in: International Conference on Machine Learning, 2018, pp. 600–609.

[12] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor, Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 1247–1250.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, Oksana Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Process. Syst. 26 (2013).

[14] Sebastian Bosse, Dominique Maniry, Klaus-Robert Müller, Thomas Wiegand, Wojciech Samek, Deep neural networks for no-reference and full-reference image quality assessment, IEEE Trans. Image Process. 27 (1) (2017) 206–219.

[15] Leo Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[16] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, CRC, Boca Raton, FL, 1984.

[17] Kieran Browne, Ben Swift, Semantics and explanation: why counterfactual explanations produce adversarial examples in deep neural networks, 2020, arXiv preprint arXiv:2012.10076.

[18] Ines Chami, Adva Wolf, Da-Cheng Juan, Frederic Sala, Sujith Ravi, Christopher Ré, Low-dimensional hyperbolic knowledge graph embeddings, 2020, arXiv preprint arXiv:2005.00545.

[19] Kwan Ho Ryan Chan, Chong You, Haozhi Qi, John Wright, Yi Ma, ReduNet: A white-box deep network from the principle of maximizing rate reduction, J. Mach. Learn. Res. 23 (114) (2022) 1–103.

[20] Tianqi Chen, Carlos Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[21] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al., Xgboost: extreme gradient boosting, R Pack. Ver. 0.4-2 1 (4) (2015) 1–4.

[22] Hong-Shuo Chen, Shuowen Hu, Suya You, C.-C. Jay Kuo, DefakeHop++: An enhanced lightweight deepfake detector, 2022, arXiv preprint arXiv:2205.00211.

[23] Yueru Chen, C.-C. Jay Kuo, Pixelhop: A successive subspace learning (SSL) method for object recognition, J. Vis. Commun. Image Represent. (2020) 102749.

[24] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suya You, C-C Jay Kuo, DefakeHop: A light-weight high-performance deepfake detector, in: 2021 IEEE International Conference on Multimedia and Expo (ICME), IEEE, 2021, pp. 1–6.

[25] Yueru Chen, Mozhdeh Rouhsedaghat, Suya You, Raghuveer Rao, C-C Jay Kuo, Pixelhop++: A small successive-subspace-learning-based (SSL-based) model for image classification, in: 2020 IEEE International Conference on Image Processing (ICIP), IEEE, 2020, pp. 3294–3298.

[26] Yueru Chen, Yiting Shao, Jing Wang, Ge Li, C-C Jay Kuo, Point cloud attribute compression via successive subspace graph transform, in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), IEEE, 2020, pp. 66–69.

[27] Yueru Chen, Yijing Yang, Wei Wang, C.-C. Jay Kuo, Ensembles of feedforward-designed convolutional neural networks, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 3796–3800.

[28] Yueru Chen, Yijing Yang, Min Zhang, C.-C. Jay Kuo, Semi-supervised learning via feedforward-designed convolutional neural networks, in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 365–369.

[29] Hong-Shuo Chen, Kaitai Zhang, Shuowen Hu, Suya You, C-C Jay Kuo, Geo-DefakeHop: High-performance geographic fake image detection, 2021, arXiv preprint arXiv:2110.09795.

[30] Yu Cheng, Duo Wang, Pan Zhou, Tao Zhang, A survey of model compression and acceleration for deep neural networks, 2017, arXiv preprint arXiv:1710.09282.

[31] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, Jagannathan Sarangapani, A comprehensive survey on model compression and acceleration, Artif. Intell. Rev. 53 (7) (2020) 5113–5155.

[32] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, Bernt Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.

[33] Corinna Cortes, Vladimir Vapnik, Support-vector networks, Mach. Learn. 20 (3) (1995) 273–297.

[34] George Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.

[35] Navneet Dalal, Bill Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol. 1, IEEE, 2005, pp. 886–893.

[36] Alexandru Damian, Jason Lee, Mahdi Soltanolkotabi, Neural networks can learn representations with gradient descent, in: Conference on Learning Theory, PMLR, 2022, pp. 5413–5452.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2009, pp. 248–255.

[38] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, Sebastian Riedel, Convolutional 2d knowledge graph embeddings, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[39] Yichen Ding, Varun Gudapati, Ruiyuan Lin, Yanan Fei, René R Sevag Packard, Sibo Song, Chih-Chiang Chang, Kyung In Baek, Zhaoqiang Wang, Mehrdad Roustaei, et al., Saak transform-based machine learning for light-sheet imaging of cardiac trabeculation, IEEE Trans. Biomed. Eng. 68 (1) (2020) 225–235.

[40] Brian Dolhansky, Joanna Bitton, Ben Pflaum, Jikuo Lu, Russ Howes, Menglin Wang, Cristian Canton Ferrer, The deepfake detection challenge (dfdc) dataset, 2020, arXiv preprint arXiv:2006.07397.

[41] Stephan Dreiseitl, Lucila Ohno-Machado, Logistic regression and artificial neural network classification models: a methodology review, J. Biomed. Inform. 35 (5–6) (2002) 352–359.

[42] Thomas Elsken, Jan Hendrik Metzen, Frank Hutter, Neural architecture search: A survey, J. Mach. Learn. Res. 20 (1) (2019) 1997–2017.

[43] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, Andrew Zisserman, The pascal visual object classes (voc) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.

[44] Pedro Felzenszwalb, David McAllester, Deva Ramanan, A discriminatively trained, multiscale, deformable part model, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.

[45] Jerry A. Fodor, Zenon W. Pylyshyn, Connectionism and cognitive architecture: A critical analysis, Cognition 28 (1–2) (1988) 3–71.

[46] Amos Freedy, Azad Madni, Michael Samet, Adaptive user models: Methodology and applications in man-computer systems, in: Advances in Man-Machine Systems Research, Vol. 2, JAI Press London, 1985, pp. 249–293.

[47] Timo Freiesleben, The intriguing relation between counterfactual explanations and adversarial examples, Minds Mach. 32 (1) (2022) 77–109.

[48] Jerome H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Statist. (2001) 1189–1232.

[49] Hongyu Fu, Yijing Yang, Yuhuai Liu, Joseph Lin, Ethan Harrison, Vinod K Mishra, C-C Jay Kuo, Acceleration of subspace learning machine via particle swarm optimization and parallel processing, 2022, arXiv preprint arXiv:2208.07023.

[50] Hongyu Fu, Yijing Yang, Vinod K. Mishra, C.-C. Jay Kuo, Subspace learning machine (SLM): Methodology and performance, 2022, arXiv preprint arXiv:2205.05296.

[51] Xiou Ge, Yun-Cheng Wang, Bin Wang, C.-C. Jay Kuo, Compounde: Knowledge graph embedding with translation, rotation and scaling compound operations, 2022, arXiv preprint arXiv:2207.05324.

[52] Deepti Ghadiyaram, Alan C. Bovik, Massive online crowdsourced study of subjective and objective picture quality, IEEE Trans. Image Process. 25 (1) (2015) 372–387.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[54] Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, Michal Irani, Drop the gan: In defense of patches nearest neighbors as single image generative models, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 13460–13469.

[55] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, Jürgen Schmidhuber, LSTM: A search space odyssey, IEEE Trans. Neural Netw. Learn. Syst. 28 (10) (2016) 2222–2232.

[56] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al., Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377.

[57] Isabelle Guyon, Jason Weston, Stephen Barnhill, Vladimir Vapnik, Gene selection for cancer classification using support vector machines, Mach. Learn. 46 (1) (2002) 389–422.

[58] Will Hamilton, Zhitao Ying, Jure Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, 2017, pp. 1024–1034.

[59] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al., A survey on vision transformer, IEEE Trans. Pattern Anal. Mach. Intell. (2022).

[60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[61] Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.

[62] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[63] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, Alexandra Peste, Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, J. Mach. Learn. Res. 22 (241) (2021) 1–124.

[64] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.

[65] Yedid Hoshen, Ke Li, Jitendra Malik, Non-adversarial image synthesis with generative latent nearest neighbors, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5811–5819.

[66] Vlad Hosu, Hanhe Lin, Tamas Sziranyi, Dietmar Saupe, Koniq-10k: An ecologically valid database for deep learning of blind image quality assessment, IEEE Trans. Image Process. 29 (2020) 4041–4056.

[67] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

[68] Yuhang Hu, Jinyan Wang, et al., SLFB-CNN: An interpretable neural network privacy protection framework, in: 2020 16th International Conference on Computational Intelligence and Security (CIS), IEEE, 2020, pp. 298–302.

[69] Samuel H. Huang, Supervised feature selection: A tutorial, Artif. Intell. Res. 4 (2) (2015) 22–37.

[70] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, Austin R Benson, Combining label propagation and simple models out-performs graph neural networks, 2020, arXiv preprint arXiv:2010.13993.

[71] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, Kilian Q Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[72] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al., Spatial transformer networks, Adv. Neural Inf. Process. Syst. 28 (2015).

[73] Andrew E. Johnson, Spin-Images: A Representation for 3-D Surface Matching (Ph.D. thesis), Robotics Institute, Carnegie Mellon University, 1997.

[74] Pranav Kadam, Min Zhang, Shan Liu, C.-C. Jay Kuo, Unsupervised point cloud registration via salient points analysis (SPA), in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), IEEE, 2020, pp. 5–8.

[75] Pranav Kadam, Min Zhang, Shan Liu, C.-C. Jay Kuo, R-PointHop: A green, accurate and unsupervised point cloud registration method, 2021, arXiv preprint arXiv:2103.08129.

[76] Pranav Kadam, Min Zhang, Shan Liu, C.-C. Jay Kuo, R-PointHop: A green, accurate, and unsupervised point cloud registration method, IEEE Trans. Image Process. 31 (2022) 2710–2725.

[77] Pranav Kadam, Qingyang Zhou, Shan Liu, C.-C. Jay Kuo, PCRP: Unsupervised point cloud object retrieval and pose estimation, 2022, arXiv preprint arXiv:2202.07843.

[78] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila, Analyzing and improving the image quality of stylegan, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8110–8119.

[79] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, Mubarak Shah, Transformers in vision: A survey, ACM Comput. Surv. (2021).

[80] Jongyoo Kim, Sanghoon Lee, Fully deep blind image quality predictor, IEEE J. Sel. Top. Sign. Proces. 11 (1) (2016) 206–220.

[81] Diederik P. Kingma, Max Welling, Auto-encoding variational bayes, 2013, arXiv preprint arXiv:1312.6114.

[82] Thomas N. Kipf, Max Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2016.

[83] Johannes Klicpera, Aleksandar Bojchevski, Stephan Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank, in: International Conference on Learning Representations, 2018.

[84] Ron Kohavi, George H. John, Wrappers for feature subset selection, Artificial Intelligence 97 (1–2) (1997) 273–324.

[85] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet classification with deep convolutional neural networks, in: F. Pereira, C.J.C. Burges, L. Bottou, K.Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 25, Curran Associates, Inc., 2012, pp. 1097–1105.

[86] C.-C. Jay Kuo, Understanding convolutional neural networks with a mathematical model, J. Vis. Commun. Image Represent. 41 (2016) 406–413.

[87] C.-C. Jay Kuo, The CNN as a guided multilayer RECOS transform [lecture notes], IEEE Signal Process. Mag. 34 (3) (2017) 81–89.

[88] C.-C. Jay Kuo, Yueru Chen, On data-driven saak transform, J. Vis. Commun. Image Represent. 50 (2018) 237–246.

[89] C-C Jay Kuo, Min Zhang, Siyang Li, Jiali Duan, Yueru Chen, Interpretable convolutional neural networks via feedforward design, J. Vis. Commun. Image Represent. 60 (2019) 346–359.

[90] Loïc Lannelongue, Jason Grealey, Michael Inouye, Green algorithms: quantifying the carbon footprint of computation, Adv. Sci. 8 (12) (2021) 2100707.

[91] Eric Cooper Larson, Damon Michael Chandler, Most apparent distortion: full-reference image quality assessment and the role of strategy, J. Electron. Imaging 19 (1) (2010) 011006.

[92] Yann LeCun, Yoshua Bengio, Geoffrey E. Hinton, Deep learning, Nature 521 (2015) 436–444.

[93] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[94] Xuejing Lei, Wei Wang, C.-C. Jay Kuo, GENHOP: An image generation method based on successive subspace learning, 2022, arXiv preprint arXiv:2210.03689.

[95] Xuejing Lei, Ganning Zhao, C.-C. Jay Kuo, NITES: A non-parametric interpretable texture synthesis method, in: 2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2020, pp. 1698–1706.

[96] Xuejing Lei, Ganning Zhao, Kaitai Zhang, C.-C. Jay Kuo, Tghop: an explainable, efficient, and lightweight method for texture generation, APSIPA Trans. Signal Inf. Process. 10 (2021).

[97] Zhi Li, Christos Bampis, Julie Novak, Anne Aaron, Kyle Swanson, Anush Moorthy, JD Cock, VMAF: The journey continues, Netflix Technol. Blog 25 (2018).

[98] Qimai Li, Zhichao Han, Xiao-Ming Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[99] Yuezun Li, Siwei Lyu, Exposing deepfake videos by detecting face warping artifacts, 2018, arXiv preprint arXiv:1811.00656.

[100] Yuezun Li, Siwei Lyu, Exposing DeepFake videos by detecting face warping artifacts, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019.

[101] Ke Li, Jitendra Malik, Implicit maximum likelihood estimation, 2018, arXiv preprint arXiv:1809.09087.

[102] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, Zhichao Guan, Label efficient semi-supervised learning via graph filtering, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9582–9591.

[103] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, Siwei Lyu, Celeb-df: A large-scale challenging dataset for deepfake forensics, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3207–3216.

[104] Hanhe Lin, Vlad Hosu, Dietmar Saupe, KADID-10k: A large-scale artificially distorted IQA database, in: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), IEEE, 2019, pp. 1–3.
[105] Weisi Lin, C.-C. Jay Kuo, Perceptual visual quality metrics: A survey, J. Vis. Commun. Image Represent. 22 (4) (2011) 297–312.
[106] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, C Lawrence Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[107] Ruiyuan Lin, Suya You, Raghuveer Rao, C.-C. Jay Kuo, On relationship of multilayer perceptrons and piecewise polynomial approximators, IEEE Signal Process. Lett. 28 (2021) 1813–1817.
[108] Ruiyuan Lin, Zhiruo Zhou, Suya You, Raghuveer Rao, C-C Jay Kuo, Geometrical interpretation and design of multilayer perceptron design, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[109] Xiaofeng Liu, Fangxu Xing, Chao Yang, C-C Jay Kuo, Suma Babu, Georges El Fakhri, Thomas Jenkins, Jonghye Woo, VoxelHop: Successive subspace learning for ALS disease classification using structural MRI, 2021, arXiv preprint arXiv:2101.05131.
[110] Shan Liu, Min Zhang, Pranav Kadam, C.-C. Jay Kuo, 3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods, Springer, 2021.
[111] Wei-Yin Loh, Classification and regression trees, Wiley Interdiscip. Rev.: Data Min. Knowl. Discov. 1 (1) (2011) 14–23.
[112] Christos Louizos, Karen Ullrich, Max Welling, Bayesian compression for deep learning, Adv. Neural Inf. Process. Syst. 30 (2017).
[113] Yi Ma, Doris Tsao, Heung-Yeung Shum, On the principles of parsimony and self-consistency for the emergence of intelligence, Front. Inf. Technol. Electron. Eng. (2022) 1–26.
[114] Azad M. Madni, Michael G. Samet, Amos Freedy, A trainable on-line model of the human operator in information acquisition tasks, IEEE Trans. Syst. Man Cybern. 12 (4) (1982) 504–511.
[115] Azad M. Madni, Michael G. Samet, D. Purcell, Adaptive Models in Information Management, in: Applications in Artificial Intelligence, Petrocelli, Princeton, New Jersey, 1985, pp. 279–294.
[116] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, Adrian Vladu, Towards deep learning models resistant to adversarial attacks, 2017, arXiv preprint arXiv:1706.06083.
[117] Farzaneh Mahdisoltani, Joanna Biega, Fabian Suchanek, Yago3: A knowledge base from multilingual wikipedias, in: 7th Biennial Conference on Innovative Data Systems Research, CIDR Conference, 2014.
[118] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, Stephen Paul Smolley, Least squares generative adversarial networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2794–2802.
[119] Falko Matern, Christian Riess, Marc Stamminger, Exploiting visual artifacts to expose deepfakes and face manipulations, in: 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW), IEEE, 2019, pp. 83–92.
[120] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, Aram Galstyan, A survey on bias and fairness in machine learning, ACM Comput. Surv. 54 (6) (2021) 1–35.
[121] Zhanxuan Mei, Yun-Cheng Wang, Xingze He, C.-C. Jay Kuo, GreenBIQA: A lightweight blind image quality assessment method, 2022, arXiv preprint arXiv:2206.14400.
[122] George A. Miller, WordNet: a lexical database for English, Commun. ACM 38 (11) (1995) 39–41.
[123] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, Demetri Terzopoulos, Image segmentation using deep learning: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 44 (7) (2022) 3523–3542.
[124] Ishan Misra, Laurens van der Maaten, Self-supervised learning of pretext-invariant representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 6707–6717.
[125] Anish Mittal, Anush Krishna Moorthy, Alan Conrad Bovik, No-reference image quality assessment in the spatial domain, IEEE Trans. Image Process. 21 (12) (2012) 4695–4708.
[126] Anish Mittal, Rajiv Soundararajan, Alan C. Bovik, Making a completely blind image quality analyzer, IEEE Signal Process. Lett. 20 (3) (2012) 209–212.
[127] MG Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, Faraz Hussain, Machine learning at the network edge: A survey, ACM Comput. Surv. 54 (8) (2021) 1–37.
[128] Huy H Nguyen, Fuming Fang, Junichi Yamagishi, Isao Echizen, Multi-task learning for detecting and segmenting manipulated facial images and videos, 2019, arXiv preprint arXiv:1906.06876.
[129] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, Dinh Phung, A novel embedding model for knowledge base completion based on convolutional neural network, 2017, arXiv preprint arXiv:1712.02121.
[130] Huy H. Nguyen, Junichi Yamagishi, Isao Echizen, Use of a capsule network to detect fake images and videos, 2019, arXiv preprint arXiv:1910.12467.
[131] Feiping Nie, Shiming Xiang, Yun Liu, Changshui Zhang, A general graph-based semi-supervised learning with novel class discovery, Neural Comput. Appl. 19 (4) (2010) 549–555.
[132] Curtis G. Northcutt, Anish Athalye, Jonas Mueller, Pervasive label errors in test sets destabilize machine learning benchmarks, 2021, arXiv preprint arXiv:2103.14749.
[133] Samet Oymak, Mingchen Li, Mahdi Soltanolkotabi, Generalization guarantees for neural architecture search with train-validation split, in: International Conference on Machine Learning, PMLR, 2021, pp. 8291–8301.
[134] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, PMLR, 2013, pp. 1310–1318.
[135] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, Hanna Wallach, Manipulating and measuring model interpretability, in: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, 2021, pp. 1–52.
[136] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[137] Frank Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain., Psychol. Rev. 65 (6) (1958) 386.
[138] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner, Faceforensics++: Learning to detect manipulated facial images, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1–11.
[139] Mozhdeh Rouhsedaghat, Yifan Wang, Xiou Ge, Shuowen Hu, Suya You, C-C Jay Kuo, Facehop: A light-weight low-resolution face gender classification method, in: International Conference on Pattern Recognition, Springer, 2021, pp. 169–183.
[140] Mozhdeh Rouhsedaghat, Yifan Wang, Shuowen Hu, Suya You, C-C Jay Kuo, Low-resolution face recognition in resource-constrained environments, Pattern Recognit. Lett. 149 (2021) 193–199.
[141] Cynthia Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nat. Mach. Intell. 1 (5) (2019) 206–215.
[142] David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.
[143] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, Shahrokh Valaee, Recent advances in recurrent neural networks, 2017, arXiv preprint arXiv:1801.01078.
[144] Victor Sanh, Lysandre Debut, Julien Chaumond, Thomas Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019, arXiv preprint arXiv:1910.01108.
[145] Henry Scheffe, The Analysis of Variance, Vol. 72, John Wiley & Sons, 1999.
[146] Roy Schwartz, Jesse Dodge, Noah A. Smith, Oren Etzioni, Green AI, Commun. ACM 63 (12) (2020) 54–63.
[147] Selim Seferbekov, A prize winning solution for DFDC challenge, 2020, https://github.com/selimsef/dfdc_deepfake_challenge.
[148] Or Sharir, Barak Peleg, Yoav Shoham, The cost of training NLP models: A concise overview, 2020, arXiv preprint arXiv:2004.08900.
[149] Razieh Sheikhpour, Mehdi Agha Sarram, Sajjad Gharaghani, Mohammad Ali Zare Chahooki, A survey on semi-supervised feature selection methods, Pattern Recognit. 64 (2017) 141–158.
[150] Karen Simonyan, Andrew Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[151] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, Chun-Liang Li, Fixmatch: Simplifying semi-supervised learning with consistency and confidence, Adv. Neural Inf. Process. Syst. 33 (2020) 596–608.
[152] Saúl Solorio-Fernández, J Ariel Carrasco-Ochoa, José Fco Martínez-Trinidad, A review of unsupervised feature selection methods, Artif. Intell. Rev. 53 (2) (2020) 907–948.
[153] Mahdi Soltanolkotabi, Adel Javanmard, Jason D. Lee, Theoretical insights into the optimization landscape of over-parameterized shallow neural networks, IEEE Trans. Inform. Theory 65 (2) (2018) 742–769.
[154] Zixing Song, Xiangli Yang, Zenglin Xu, Irwin King, Graph-based semi-supervised learning: A comprehensive review, IEEE Trans. Neural Netw. Learn. Syst. (2022).
[155] Emma Strubell, Ananya Ganesh, Andrew McCallum, Energy and policy considerations for deep learning in NLP, 2019, arXiv preprint arXiv:1906.02243.
[156] Yuanhang Su, C.-C. Jay Kuo, On extended long short-term memory and dependent bidirectional recurrent neural network, Neurocomputing 356 (2019) 151–161.
[157] Yuanhang Su, C.-C. Jay Kuo, et al., Recurrent neural networks and their memory behavior: A survey, APSIPA Trans. Signal Inf. Process. 11 (1) (2022).
[158] Shaolin Su, Qingsen Yan, Yu Zhu, Cheng Zhang, Xin Ge, Jinqiu Sun, Yanning Zhang, Blindly assess image quality in the wild guided by a self-adaptive hyper network, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3667–3676.
[159] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, Jian Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, 2019, arXiv preprint arXiv:1902.10197.

[160] Federico Tombari, Samuele Salti, Luigi Di Stefano, Unique signatures of histograms for local surface description, in: Proceedings of the European Conference on Computer Vision, Springer, 2010, pp. 356–369.

[161] Kristina Toutanova, Danqi Chen, Observed versus latent features for knowledge base and text inference, in: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, 2015, pp. 57–66.

[162] Tzu-Wei Tseng, Kai-Jiun Yang, C-C Jay Kuo, Shang-Ho Tsai, An interpretable compression and classification system: Theory and applications, IEEE Access 8 (2020) 143962–143974.

[163] Matthew Turk, Alex Pentland, Eigenfaces for recognition, J. Cogn. Neurosci. 3 (1) (1991) 71–86.

[164] Jesper E. Van Engelen, Holger H. Hoos, A survey on semi-supervised learning, Mach. Learn. 109 (2) (2020) 373–440.

[165] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[166] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, Yoshua Bengio, Graph Attention Networks, in: International Conference on Learning Representations, 2018.

[167] Sheng Wan, Yibing Zhan, Liu Liu, Baosheng Yu, Shirui Pan, Chen Gong, Contrastive graph Poisson networks: Semi-supervised learning with extremely limited labels, Adv. Neural Inf. Process. Syst. 34 (2021).

[168] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, Image quality assessment: from error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.

[169] Yun-Cheng Wang, Xiou Ge, Bin Wang, C.-C. Jay Kuo, Greenkgc: A lightweight knowledge graph completion method, 2022, arXiv preprint arXiv:2208.09137.

[170] Yun-Cheng Wang, Xiou Ge, Bin Wang, C.-C. Jay Kuo, KGBoost: A classification-based knowledge base completion method with negative sampling, Pattern Recognit. Lett. 157 (2022) 104–111.

[171] Jinyan Wang, Qiyu Li, Yuhang Hu, Xianxian Li, et al., A privacy preservation framework for feedforward-designed convolutional neural networks, Neural Netw. (2022).

[172] Yue Wang, Justin M. Solomon, Deep closest point: Learning representations for point cloud registration, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3523–3532.

[173] Yue Wang, Justin M. Solomon, Prnet: Self-supervised learning for partial-to-partial registration, in: Advances in Neural Information Processing Systems, 2019, pp. 8814–8826.

[174] Fei Wang, Changshui Zhang, Label propagation through linear neighborhoods, IEEE Trans. Knowl. Data Eng. 20 (1) (2007) 55–67.

[175] Chengwei Wei, C.-C. Jay Kuo, Rafael Liuz Testa, Ariane Machado-Lima, L. S. Nunes Fatima, ExpressionHop: A lightweight human facial expression classifier, in: Proceedings of the IEEE Conference on Multimedia Information Processing and Retrieval, 2022.

[176] Chengwei Wei, Bin Wang, C.-C. Jay Kuo, SynWMD: Syntax-aware word mover's distance for sentence similarity evaluation, 2022, arXiv preprint arXiv:2206.10029.

[177] Chengwei Wei, Bin Wang, C.-C. Jay Kuo, Task-specific dependency-based word embedding methods, Pattern Recognit. Lett. (2022).

[178] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al., Transformers: State-of-the-art natural language processing, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 38–45.

[179] John Wright, Yi Ma, High-Dimensional Data Analysis with Low-Dimensional Models: Principles, Computation, and Applications, Cambridge University Press, 2022.

[180] Xiang Wu, Ran He, Zhenan Sun, Tieniu Tan, A light CNN for deep face representation with noisy labels, IEEE Trans. Inf. Forensics Secur. 13 (11) (2018) 2884–2896.

[181] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al., Sustainable AI: Environmental implications, challenges and opportunities, Proc. Mach. Learn. Syst. 4 (2022) 795–813.

[182] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, Jianxiong Xiao, 3D shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1912–1920.

[183] Tian Xie, Rajgopal Kannan, C.-C. Jay Kuo, GraphHop++: New insights into GraphHop and its enhancement, 2022, arXiv preprint arXiv:2204.08646.

[184] Tian Xie, Bin Wang, C.-C. Jay Kuo, Graphhop: An enhanced label propagation method for node classification, IEEE Trans. Neural Netw. Learn. Syst. (2022).

[185] Hao Xu, Yueru Chen, Ruiyuan Lin, C.-C. Jay Kuo, Understanding CNN via deep features analysis, in: 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2017, pp. 1052–1060.

[186] Jingtao Xu, Peng Ye, Qiaohong Li, Haiqing Du, Yong Liu, David Doermann, Blind image quality assessment based on high order statistics aggregation, IEEE Trans. Image Process. 25 (9) (2016) 4444–4457.

[187] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, Lei Li, A survey on green deep learning, 2021, arXiv preprint arXiv:2111.05193.

[188] Yijing Yang, Hongyu Fu, C.-C. Jay Kuo, Design of supervision-scalable learning systems: Methodology and performance benchmarking, 2022, arXiv preprint arXiv:2206.09061.

[189] Jiaolong Yang, Hongdong Li, Dylan Campbell, Yunde Jia, Go-ICP: A globally optimal solution to 3D ICP point-set registration, IEEE Trans. Pattern Anal. Mach. Intell. 38 (11) (2015) 2241–2254.

[190] Xiaohan Yang, Fan Li, Hantao Liu, A survey of DNN methods for blind image quality assessment, IEEE Access 7 (2019) 123788–123806.

[191] Xin Yang, Yuezun Li, Siwei Lyu, Exposing deep fakes using inconsistent head poses, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2019, pp. 8261–8265.

[192] Yijing Yang, Vasileios Magoulianitis, C.-C. Jay Kuo, E-pixelhop: An enhanced pixelhop method for object classification, in: 2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), IEEE, 2021, pp. 1475–1482.

[193] Yijing Yang, Vasileios Magoulianitis, Xinyu Wang, C-C Jay Kuo, Statistical attention localization (SAL): Methodology and application to object classification, 2022, arXiv preprint arXiv:2208.01823.

[194] Yijing Yang, Wei Wang, Hongyu Fu, C.-C. Jay Kuo, On supervised feature selection from high dimensional feature spaces, 2022, arXiv preprint arXiv:2203.11924.

[195] Peng Ye, Jayant Kumar, Le Kang, David Doermann, Unsupervised feature learning framework for no-reference image quality assessment, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2012, pp. 1098–1105.

[196] Bianca Zadrozny, Learning and evaluating classifiers under sample selection bias, in: Proceedings of the Twenty-First International Conference on Machine Learning, 2004, p. 114.

[197] Delu Zeng, Minyu Liao, Mohammad Tavakolian, Yulan Guo, Bolei Zhou, Dewen Hu, Matti Pietikäinen, Li Liu, Deep learning for scene classification: A survey, 2021, arXiv preprint arXiv:2101.10531.

[198] Hui Zeng, Lei Zhang, Alan C. Bovik, Blind image quality assessment with a probabilistic quality representation, in: 2018 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 609–613.

[199] Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, Ren Chen, Decoupling the depth and scope of graph neural networks, Adv. Neural Inf. Process. Syst. 34 (2021).

[200] Kaitai Zhang, Hong-Shuo Chen, Ye Wang, Xiangyang Ji, C.-C. Jay Kuo, Texture analysis via hierarchical spatial-spectral correlation (HSSC), in: 2019 IEEE International Conference on Image Processing (ICIP), IEEE, 2019, pp. 4419–4423.

[201] Min Zhang, Pranav Kadam, Shan Liu, C.-C. Jay Kuo, Unsupervised feedforward feature (UFF) learning for point cloud classification and segmentation, in: 2020 IEEE International Conference on Visual Communications and Image Processing (VCIP), IEEE, 2020, pp. 144–147.

[202] Min Zhang, Pranav Kadam, Shan Liu, C.-C. Jay Kuo, GSIP: Green semantic segmentation of large-scale indoor point clouds, 2021, arXiv preprint arXiv:2109.11835.

[203] Xinfeng Zhang, Sam Kwong, C.-C. Jay Kuo, Data-driven transform-based compressed image quality assessment, IEEE Trans. Circuits Syst. Video Technol. 31 (9) (2020) 3352–3365.

[204] Weixia Zhang, Kede Ma, Jia Yan, Dexiang Deng, Zhou Wang, Blind image quality assessment using a deep bilinear convolutional neural network, IEEE Trans. Circuits Syst. Video Technol. 30 (1) (2018) 36–47.

[205] Min Zhang, Yifan Wang, Pranav Kadam, Shan Liu, C-C Jay Kuo, PointHop++: A lightweight learning model on point sets for 3D classification, 2020, arXiv preprint arXiv:2002.03281.

[206] Xinfeng Zhang, Chao Yang, Xiaoguang Li, Shan Liu, Haitao Yang, Ioannis Katsavounidis, Shaw-Min Lei, C-C Jay Kuo, Image coding with data-driven transforms: Methodology, performance and potential, IEEE Trans. Image Process. 29 (2020) 9292–9304.

[207] Min Zhang, Haoxuan You, Pranav Kadam, Shan Liu, C-C Jay Kuo, PointHop: An explainable machine learning method for point cloud classification, IEEE Trans. Multimed. (2020).

[208] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, Michael Jordan, Theoretically principled trade-off between robustness and accuracy, in: International Conference on Machine Learning, PMLR, 2019, pp. 7472–7482.

[209] Lin Zhang, Lei Zhang, Xuanqin Mou, David Zhang, FSIM: A feature similarity index for image quality assessment, IEEE Trans. Image Process. 20 (8) (2011) 2378–2386.

[210] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.

[211] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, Xindong Wu, Object detection with deep learning: A review, IEEE Trans. Neural Netw. Learn. Syst. 30 (11) (2019) 3212–3232.

[212] Hanqing Zhao, Wenbo Zhou, Dongdong Chen, Tianyi Wei, Weiming Zhang, Nenghai Yu, Multi-attentional deepfake detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2185–2194.

[213] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, Bernhard Schölkopf, Learning with local and global consistency, Adv. Neural Inf. Process. Syst. 16 (2003) 321–328.

[214] Peng Zhou, Xintong Han, Vlad I. Morariu, Larry S. Davis, Two-stream neural networks for tampered face detection, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2017, pp. 1831–1839.

[215] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, Antonio Torralba, Places: A 10 million image database for scene recognition, IEEE Trans. Pattern Anal. Mach. Intell. (2017).

[216] Qian-Yi Zhou, Jaesik Park, Vladlen Koltun, Fast global registration, in: Proceedings of the European Conference on Computer Vision, Springer, 2016, pp. 766–782.

[217] Xiaojin Zhu, Zoubin Ghahramani, John D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 912–919.

[218] Yao Zhu, Xinyu Wang, Hong-Shuo Chen, Ronald Salloum, C-C Jay Kuo, A-PixelHop: A green, robust and explainable fake-image detector, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 8947–8951.

[219] Yushan Zhu, Wen Zhang, Mingyang Chen, Hui Chen, Xu Cheng, Wei Zhang, Huajun Chen, DualDE: Dually distilling knowledge graph embedding for faster and cheaper reasoning, in: Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, 2022, pp. 1516–1524.

[220] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, Jieping Ye, Object detection in 20 years: A survey, 2019, arXiv preprint arXiv:1905.05055.