NUMBER 4/2021

ISSN 2324-3635

OCCASIONAL & DISCUSSION PAPER SERIES

A Comprehensive Review of Deep Learning Algorithms

Dr Soheil Varastehpour

Dr Hamid Sharifzadeh

Dr Iman Ardekani

ePress

Unitec Institute of Technology

OCCASIONAL AND DISCUSSION
PAPER SERIES 4/2021

A Comprehensive Review of Deep Learning Algorithms

By Dr Soheil Varastehpour, Dr Hamid Sharifzadeh and Dr Iman Ardekani

A Comprehensive Review of Deep Learning Algorithms by Dr Soheil Varastehpour, Dr Hamid Sharifzadeh and Dr Iman Ardekani is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.

Keywords: artificial intelligence, deep learning, convolutional neural network, autoencoder, resticted Boltzmann machine, sparse coding

This publication may be cited as:

Varastehpour, S., Sharifzadeh, H., Ardekani, I. (2021). A Comprehensive Review of Deep Learning Algorithms. *Unitec ePress Occasional and Discussion Papers Series* (2021:4). Retrieved from http://www.unitec.ac.nz/epress.

About this series:

Unitec ePress periodically publishes occasional and discussion papers that discuss current and ongoing research authored by members of staff and their research associates. All papers are blind reviewed. For more papers in this series please visit: www.unitec.ac.nz/epress/index.php/category/publications/epress-series/discussion-and-occasionalpapers.

Cover design by Penny Thomson

Contact:

epress@unitec.ac.nz
www.unitec.ac.nz/epress/

Unitec New Zealand
Private Bag 92025, Victoria Street West
Auckland 1142
New Zealand

ISSN
2324-3635

A Comprehensive Review of Deep Learning Algorithms

Dr Soheil Varastehpour, Dr Hamid Sharifzadeh and Dr Iman Ardekani

Abstract

Deep learning algorithms are a subset of machine learning algorithms that aim to explore several levels of the distributed representations from the input data. Recently, many deep learning algorithms have been proposed to solve traditional artificial intelligence problems. In this review paper, some of the up-to-date algorithms of this topic in the field of computer vision and image processing are reviewed. Following this, a brief overview of several different deep learning methods and their recent developments are discussed.

Introduction

Researchers have been trying to simulate the human brain through machines (computers) for decades, to create computers that can learn, think, and make decisions by themselves. To achieve this goal, artificial intelligence (AI) was born (Brunette et al., 2009). AI is an area of computer science that emphasises the creation of intelligent machines that work and react like humans. AI has become an essential part of the technology industry these days. Currently, AI is widely

used in speech recognition, image recognition, learning, planning and problem solving (Brunette et al., 2009).

Machine learning is an application of AI that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Brunette et al., 2009; Dutton & Conroy, 1997). Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. In machine learning, there is some data (dataset), and then a model (method) is trained using this data in order to predict new data. In other words, the model tries to teach a computer to do something special. For example, a recognition model is designed to distinguish photos of cats from a group of photos that include cats and other animals or humans. After each attempt, the model receives its prediction result and understands how it predicted. The feedback is presented to the model in the form of an error in order for it to adjust itself to get better results and eliminate errors. Machine learning algorithms can now enable computers to communicate with humans, autonomously drive cars, write and publish sports-match reports and find terrorist suspects (Dutton & Conroy, 1997), and has begun to have an impact on most industries and the jobs within them.

Deep learning is an emerging area of research and modern application. It is a very widespread and demanding field, relevant to industry, business and healthcare. Deep learning methods have gained popularity because they often outperform conventional (i.e., shallow) machine learning methods and can extract features automatically from raw data with little or no preprocessing. One of the key reasons deep learning is more powerful than classical machine learning is that it creates transferable solutions (Bengio, 2009). Deep learning algorithms are able to create transferrable solutions through neural networks: that is, layers of neurons/units. In addition, machine learning algorithms almost always require structured data, whereas deep learning networks rely on layers of artificial neural networks (ANN). Furthermore, deep learning algorithms are able to solve complex problems that require discovering hidden patterns in the data and/or a deep understanding of intricate relationships between a large number of interdependent variables (Bengio, 2009). Deep learning combines a number of important research fields, including the internet of things (IoT), e-healthcare, cybersecurity, bioinformatics and optimisation, which are interdependent. Deep learning has the potential to drive innovations in industry, healthcare or business intelligence, and its use will increase in the future.

In recent years, deep learning has been extensively studied in the field of computer science and a number of related approaches have been developed. Generally, these methods are based on the basic method from which they are derived. They are divided into four different categories: convolutional neural networks (CNN), restricted Boltzmann machines (RBM), autoencoder (AE) and sparse coding (SC) (Schmidhuber, 2015; Krizhevsky et al., 2012; K. He et al., 2016; Cireșan et al., 2012; Mikolov et al., 2013). This review paper discusses the categories and sub-categories of these methods, including their pros and cons, and the fields in which they can be used in computer sciences, especially computer vision and image processing fields.

Figure 1 shows the genealogy of deep learning, along with the work

that has been done on each method. To cover the types of deep learning algorithms, the following four sections will: review different algorithms of the CNN; explain the history of RBMs and categorise the different types of RBM algorithms; and explain the AE and SC algorithms respectively. The authors then discuss and conclude this review.

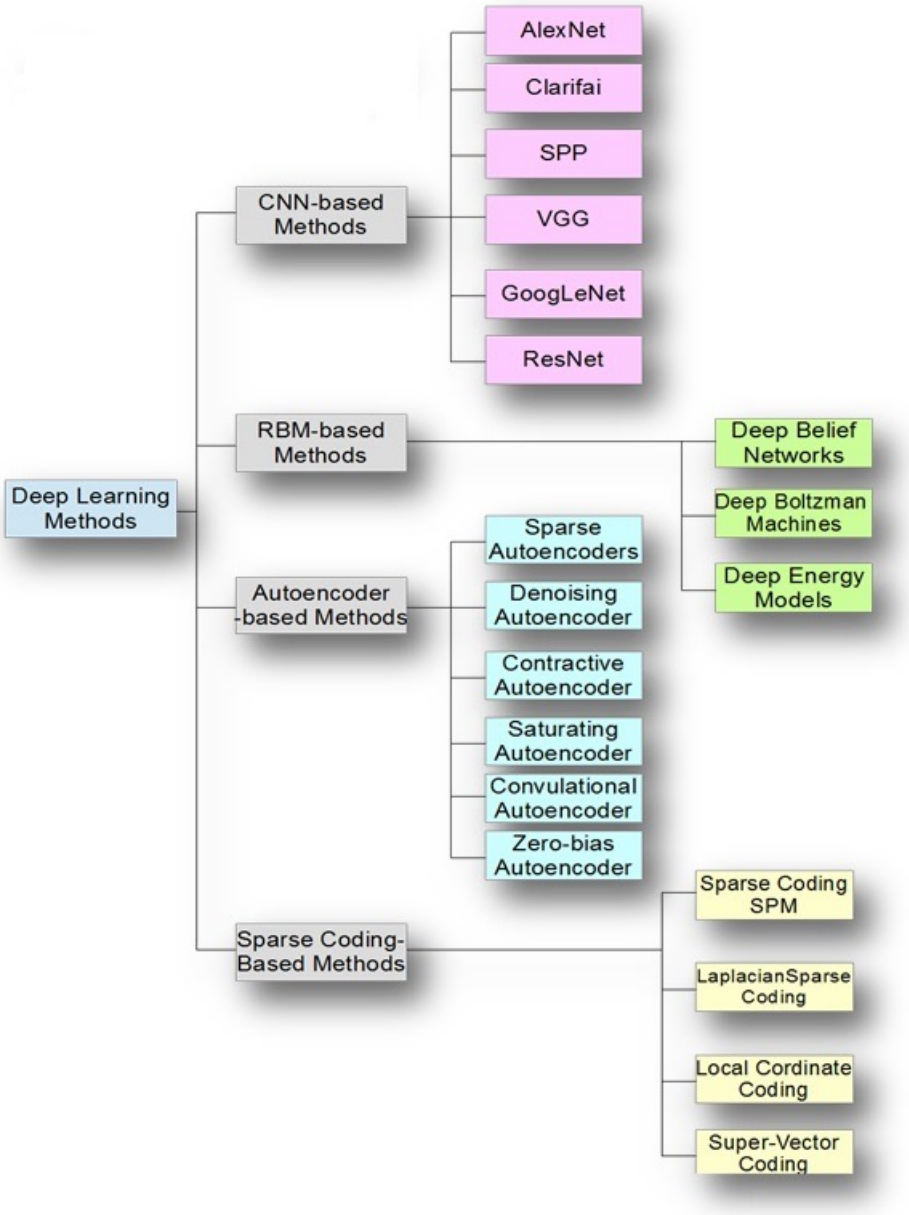

Figure 1. Deep learning genealogy.

Convolutional Neural Networks (CNN)

CNNs are one of the most important deep learning methods, in which multiple layers are trained in a powerful way (Krizhevsky et al., 2012). This method is very efficient and it is one of the most common methods in various computer vision applications. In general, a CNN consists of three main layers: the convolutional layer, the pooling layer and the fully connected layer. Different layers perform different tasks.

The main advantage of using a CNN is that it can extract the spatial features from the data using its kernel, which other networks cannot do (Cao et al., 2018). For example, CNN can detect edges, distribution of colours, etc., in the image, making these networks very robust in image classification and other similar data containing spatial properties. In terms of using CNN, we generally refer to a two-dimensional CNN which is used for image classification. But there are two other types of CNNs used in the real world, which are one-dimensional and three-dimensional CNNs (Cao et al., 2018). One-dimensional CNN (Conv1D) is used for time-series data. In Conv1D, the kernel slides along one dimension. This data is collected from an accelerometer that a person is wearing on their arm. Data represents the acceleration in all three axes. One-dimensional CNN can perform activity-recognition tasks from accelerometer data, such as if the person is standing, walking, jumping, etc. This data has two dimensions. The first dimension is time-steps, and the other is the values of the acceleration in three axes. Similarly, one-dimensional CNNs are also used on audio and text data since we can also represent the sound and texts as time-series data (Cao et al., 2018). Two-dimensional CNN (Conv2D) is the standard CNN that was first introduced in LeNet-5 architecture. Conv2D is generally used on image data. It is called two-dimensional CNN because the kernel slides along two dimensions on the data (Cao et al., 2018). Three-dimensional CNN (Conv3D) is mainly used with 3D image data, such as magnetic resonance imaging (MRI) data. MRI data is widely used for examining the brain, spinal cord, internal organs and many more. A computerised tomography (CT) scan is also an example of 3D data created by combining a series of x-ray images taken from different angles around the body. Conv3D can be used to classify this medical data or extract features from it. One more example of 3D data is video, which is simply a sequence of image frames together. Conv3D can be applied to video as well since it has spatial features (Cao et al., 2018).

There are two steps to training in each CNN: the feed-forward stage and back-propagation or post-propagation. In the first step, the input image is fed to the network, and parameters (the coefficients) are adjusted between each neuron and input data in order to operate the CNN. Then the network output is calculated. To adjust the network parameters (the coefficients), the output result is used to calculate the network error rate (Krizhevsky et al., 2012). In order to calculate the error rate, the network output is compared with the correct answer through a loss function. The next step starts with the back-propagation stage based on the calculated error rate. At this point the gradient of each parameter is calculated according to the chain rule and all the parameters are changed according to their effect on the error in the network. With updated parameters, the next feed-forward phase can be run. After repeating a number of these steps, the network training will be ended.

TYPES OF CNN NETWORK LAYERS

A CNN is a powerful and efficient model which performs automatic feature extraction. Generally, a CNN is a hierarchical neural network with its convolutional layers interconnected with pooling layers, and there are several

fully connected layers. A CNN can be thought of as a combination of two components: feature extraction and classification. The convolution and pooling layers perform feature extraction (automatic feature extraction).

Convolutional layers: the CNN network uses different kernels to convolve the input image as well as the middle feature maps, which create different feature maps. The benefits of convolution operations are listed below (Zeiler, 2013):

- The weight-sharing mechanism in each feature map dramatically reduces the number of parameters.
- Local binding learns the relationship between neighbouring pixels.
- It causes the immutability and stability of the object to shift.

Due to the benefits introduced by convolution operations, some well-known research articles have used it to replace fully connected layers to speed up the learning process (Szegedy et al., 2015; Oquab et al., 2015). One of the interesting ways to manage convolution layers is the Network in Network (NIN) method (M. Lin et al., 2013), in which the main idea is to replace the convolution layer with a small perceptron neural network that includes several layers fully connected to non-linear activation functions. As such, linear filters are replaced by non-linear neural networks. This method produces reasonable results in image classification.

Pooling layers: a pooling layer is usually placed after a layer of convolution and can be used to reduce the size of feature maps and network parameters. Like convolution layers, pooling layers are unchanged (stable) towards the translation because of the neighbouring pixels in their calculations (Szegedy et al., 2015; Oquab et al., 2015). Pooling layer implementations, by using the max pooling function and the average pooling function, are the most common implementation methods. Using a max pooling filter of size 2 * 2 and two strides, a feature map of size 8 * 8 creates an output of size 4 * 4 (Szegedy et al., 2015; Oquab et al., 2015).

Boureau et al. (2010) provide a detailed theoretical analysis of max pooling and average pooling efficiency. Scherer et al. (2010) also make a comparison between the two operations and find that max pooling can lead to faster convergence, better generalisation (generalisation improvement), and selection of invariant features. In recent years, various rapid implementations of different types of CNNs have been performed on graphics processing units (GPUs), most of which use max pooling operations (Krizhevsky et al., 2012; Cireşan et al., 2011).

Much research has been done on pooling layers in comparison to the other three layers on CNN. There are three popular approaches to this layer and each pursues different goals. These approaches are as follows:

- **Stochastic pooling:** one drawback of max pooling is that it is sensitive to over-fitting in the training set (Deng, 2014), making it difficult to generalise (Zeiler, 2013). To solve this problem, Zeiler and Fergus (2013) propose a pooling stochastic method in which

certain pooling operations are replaced by a random procedure. This random procedure is a random selection of values within each pooling area based on a polynomial distribution (Schmidhuber, 2015). The operation is similar to the standard max pooling, with many copies of the input image having deforming local features (Bengio, 2013). The stochastic nature is useful in preventing the problem of over-fitting, which is why it has been used in this method (Bengio, 2009).

- **Spatial pyramid pooling (SPP):** typically, neural-network-based methods require a fixed-size input image. This limitation may reduce detection accuracy for images of arbitrary sizes. In order to remove this limitation, K. He et al. (2015) used a CNN, replacing the last pooling layer with an SPP layer. This layer is capable of extracting fixed-sized symbols (images) from arbitrary images (or areas). This provides a flexible solution for managing different scales, sizes and aspect ratios that can be used in most of the CNN architecture and enhance its performance (Bengio, 2009).
- **Def-pooling:** managing deformation is a major challenge in computer vision (Bengio et al., 2013), especially in the field of object recognition. Max pooling and average pooling are useful for deformation management but they are not capable of learning deformation constraints and geometrical modelling of object components. In order to better deal with deformation, Ouyang et al. (2014) introduced a new deformation-constrained pooling layer. This is known as the def-pooling layer. It enriches the deep model by learning the deformation of visual patterns. This layer can be used instead of the max pooling layer at most of the level of abstraction (Ouyang et al., 2014).

By combining several different types of pooling layers, each developed with a different purpose and method, the efficiency of a neural network can be increased.

Fully connected layer: after the last pooling layer, there are fully connected layers that convert the 2D feature maps into 1D feature vectors to continue the feature representation process. The fully connected layers act like their counterparts in traditional artificial neural networks (ANN) and comprise approximately 90% of the parameters of a CNN network. The fully connected layer allows us to present the grid result in a vector of a specified size. This vector can be used to categorise images (Krizhevsky et al., 2012) or to continue further processing (Girshick et al., 2014). Restructuring of fully connected layers is not common, but one example was carried out in the transferred learning method (Oquab et al., 2014), in which the parameters learned by ImageNet (Krizhevsky et al., 2012) were retained, but the last fully connected layer was replaced with two fully connected layers to allow the network to adapt to new vision recognition activities (Oquab et al., 2014).

The big problem with these types of layers is that they have too many parameters. This results in a very high processing cost to spend on training, so a commonly used method with satisfactory results is to either remove these layers altogether or reduce the number of connections in these layers through different methods. For example, GoogLeNet (Szegedy et al., 2015) designed a deep and extensive network where the computational cost was kept constant. This was done by switching from a fully connected architecture to a scattered connected architecture.

CNN ARCHITECTURE

With the recent advances in the use of CNN in the field of computer vision, several CNN architectures have emerged. In improving and developing the performance of different systems/applications, model architecture is a critical factor. Several developments and modifications have been achieved in CNN architecture over the last years, including structural reformulation, regularisation, parameter optimisations, etc. It should be noted that the key upgrade in CNN performance occurred largely due to the processing-unit reorganisation, as well as the development of novel blocks. In particular, the most novel developments in CNN architectures were performed on the use of network depth. In this section, the most popular CNN architectures, beginning with the AlexNet model and ending with the ResNet model, will be discussed and the characteristics of each of their applications then summarised. Studying these architecture features (such as input size, depth and robustness) is the key to help researchers to choose the most suitable architecture for their target task. The configurations and achievements of several conventional CNN models are presented in Table 1. These methods have been classified based on the input, depth and robustness features with insight into the computer vision and image processing fields.

Table 1. Top CNN models along with their configuration and achievements.

Method	Year	Configuration	Achievement
AlexNet	2012	5 convolution layers + 3 fully connected layers	An important architecture that has attracted many researchers to the field of computer vision
Clarifai	2013	5 convolution layers + 3 fully connected layers	It made what was happening inside the network visible
SPP	2014	5 convolution layers + 3 fully connected layers	By providing spatial pyramid pooling, the image-size limitation was eliminated
VGG	2014	13-15 convolution layers + 3 fully connected layers	Full evaluation of incremental depth network
GoogLeNet	2014	21 convolution layers + 1 fully connected layer	Increases network depth and width without increasing computing requirements
ResNet	2015	152 convolution layers + 1 fully connected layer	Increases grid depth and provides a way to prevent gradient saturation

AlexNet

The history of deep CNNs began with the appearance of LeNet (LeCun et al., 1995). At that time, the CNNs were restricted to handwritten digit recognition tasks, which cannot be scaled to all image classes. In deep CNN architecture, AlexNet is highly respected (Krizhevsky et al., 2012), as it achieved innovative results in the fields of image recognition and classification. Krizhevesky et al. (2012) first proposed AlexNet and consequently improved the CNN learning ability by increasing its depth and implementing several parameter-optimisation strategies.

Clarifai

In 2013, Zeiler and Fergus (2014) introduced a new visual representation method, Clarifai, which could be used to observe activities within layers of a neural network. These visualisations allowed them to find architectures that were far superior to AlexNet in the ImageNet category competition.

Spatial pyramid pooling (SPP)

To solve the constraint requiring a fixed resolution for input images, K. He et al. (2015) proposed a new spatial pyramid pooling (SPP) strategy to eliminate the image-size constraint. The SPP-net architecture obtained by this method, despite the different designs, has been able to improve accuracy in a variety of CNN architectures.

VGGNet and GoogLeNet

In addition to the commonly used configuration of CNN, there are other ways of trying to explore deeper networks. VGGNet (Simonyan & Zisserman, 2014), by adding more layers of convolution and increasing the use of small convolution filters, is able to make deeper networks. Similarly, Szegedy et al. (2015) developed a model called GoogLeNet, which has a very deep structure consisting of 22 layers. Despite the high classification performance achieved by different models, CNN-based models and their various applications are not limited to image classification. Based on these models, new frameworks have been developed that can be used for other difficult tasks such as object detection, semantic segmentation, etc.

ResNet

There are two well-known derivative frameworks: RCNN (regions with CNN features) (Girshick et al., 2014) and FCN (fully convolutional network) (Long et al., 2015), which were originally designed for object detection and semantic segmentation. The main idea behind RCNN is to create multiple object proposals, extract features from each using a CNN, and then categorise each candidate window, with a linear support vector machine (SVM) for each category. Recognition using the regions scheme has received encouraging

performance in object detection and has gradually become a general architecture for recent promising object detection algorithms (Gkioxari et al., 2015; Zhu et al., 2015).

However, RCNN's performance relies too much on object location accuracy, which may limit its power. In addition, the creation and processing of a large number of proposals may reduce its optimality. Recent advances in this area have focused mainly on these two topics (Gkioxari et al., 2015; Ren et al., 2015; Zhu et al., 2015; Hariharan et al., 2014).

RCNN uses CNN models as a feature extractor and makes no changes to the network. However, FCN offers a way to redesign CNN models into convolutional networks and it is capable of producing optimal outputs (Yoo et al., 2015). Although FCN is mainly provided for semantic segmentation, this technique can be used in other applications, such as image segmentation (Yoo et al., 2015), edge detection (Xie & Tu, 2015), etc.

Restricted Boltzmann Machine (RBM)

A restricted Boltzmann machine (RBM) is a generative stochastic neural network, developed by Hinton and Sejnowski in 1986. An RBM is a variant of a Boltzmann machine that has a limit on which visible units and hidden units form a bipartite graph (Hinton & Sejnowski, 1986). This limitation creates more-efficient training algorithms, especially the gradient-based contrastive divergence algorithm (Carreira-Perpinan & Hinton, 2005). Since this model is a bipartite graph, the hidden units (H) and the visible units (V_1) are conditionally independent (Carreira-Perpinan & Hinton, 2005). Therefore, in the following equation:

$$P(HV_1) = P(H_1V_1)P(H_2V_1)...P(H_nV_1), \quad (1)$$

both H and V_1 satisfy the Boltzmann distribution. With input V_1 we can get H through $P(HV_1)$. Similarly, we can obtain the value of V_2 through $P(H_2V_1)$. By adjusting the parameters, we can minimise the difference between V_1 and V_2, and the resulting H will act as a good property of V_1 (Carreira-Perpinan & Hinton, 2005). Hinton (2012) provides a detailed explanation and a practical way to train RBMs. Cho et al. (2011) discuss the main problems of RBM training and their underlying causes, and propose a new algorithm including an adaptive learning rate and improved gradient for fixed problems. A famous example of RBM can be found in Nair and Hinton (2010), in which rectified linear units are used in order to improve the performance of RBMs. This model approximates binary units with noisy rectified linear units in order to preserve information about the relative intensities as the information flows through the layers. This refinement works well in this model, and it is also widely used in various CNN-based methods (Krizhevsky et al., 2012; Zeiler & Fergus, 2013).

Using RBMs as learning modules, we can create deep belief networks (DBNs), deep Boltzman machines (DBMs) and deep energy models (DEMs)

(Ngiam et al., 2011). The comparison between these three models is illustrated in Figure 2.

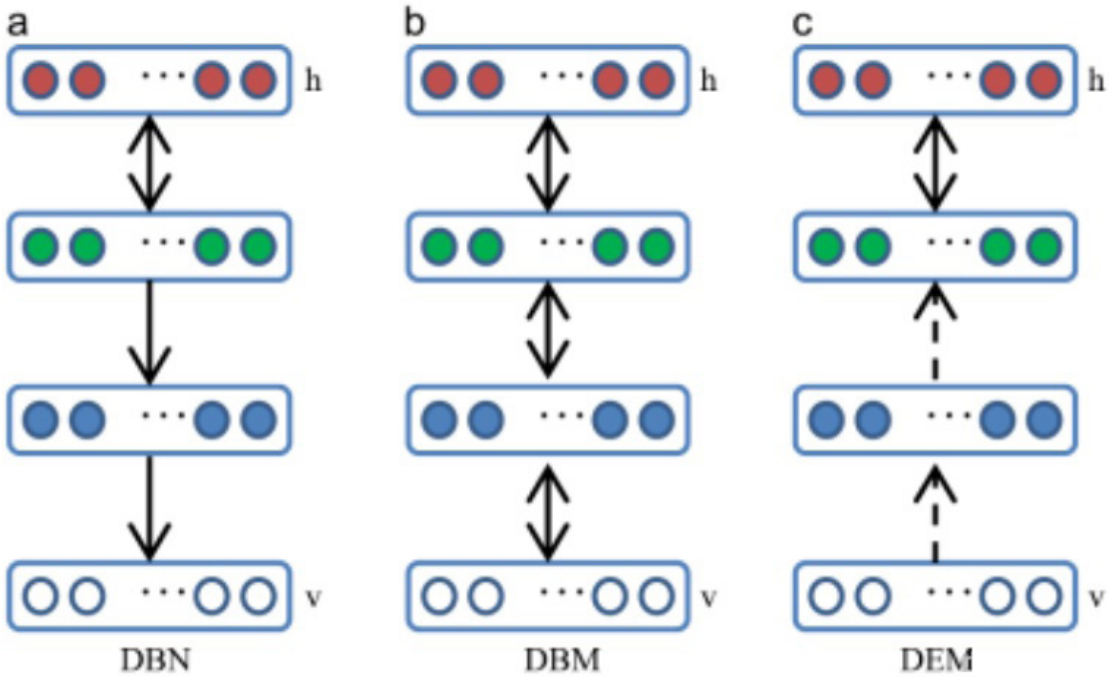

Figure 2. Comparison of three deep models, DBN, DBM and DEM (Ngiam et al., 2011, p. 1).

Table 2 represents a summary of these three models, along with their portfolio (Guo et al., 2016).

Table 2. Summary of RBM-based methods and portfolio completed.

Method	Attributes	Advantage/s	Disadvantage/s
DBN	Directional junction in the upper two layers and directional junctions in the layers	1) Properly initialises the network and partially prevents it from falling into weak local optimality 2) The process of training is unsupervised, which eliminates the need for labelled data for training	Creating a DBN model is computationally costly due to the initialisation process
DBM	Indirect connections between all layers in the network	Handles ambiguous inputs by using top-down feedback	Joint optimisation is time consuming
DEM	Definite hidden units in the bottom layers and random hidden units in the top hidden layers	Creates better productive models by allowing the lower layers to adapt to the upper layers' training	The initial weight gained may not be a good convergence

DEEP BELIEF NETWORKS (DBNS)

The DBN was provided by Hinton (Bengio et al., 2013). This network was a remarkable breakthrough in deep learning. A DBN is a probabilistic generative model that provides a common probability distribution over visible data and tags. A DBN first uses an efficient layer-by-layer greedy learning strategy to initialise the deep network parameters and then carefully align all the weights with expected outputs (fine-tune) (Bengio et al., 2013). The DBN process has

two benefits (Arel et al., 2010):

1) Provides a good initialisation for the network, and thus barely responds to the parameter selection, and may cause local optima to be partially weak.
2) Unsupervised learning procedure with no labels. It does not require a class label so eliminates the need for tagged data for training. Creating a DBN model is computationally costly because it requires multiple RBM training and it is unclear how to maximise the likelihood of training to optimise the model (Bengio et al., 2013). Many successful pieces of research have been done on DBNs and, as a result, many species have been produced (Lee et al., 2008; Lee et al, 2009; Nair & Hinton, 2009).

Nair and Hinton (2009) developed a modified DBN in which the top-layer model used a third-order Boltzmann machine to detect object recognition. The model presented in Lee et al. (2008) is the two-layer model, which captures natural images using sparse coding and RBMs, where the first layer learns local, directional and edge filters, and the second layer receives a variety of contour features along the edges and intersections.

To improve the model's power against occlusion and random noise, Lee et al. (Tang & Eliasmith, 2010) used two strategies. The first strategy was to regularise sparse connections in the first layer of the DBN in order to regularise the model, and the second strategy was the development of a probabilistic denoising algorithm. There is a major limitation if we intend to use this network for computer vision activities. The problem with DBNs is that they do not consider the 2D structure of an input image. To address this problem, convolutional deep belief networks (CDBNs) were introduced (Lee et al., 2009). The CDBN utilises spatial information of neighbouring pixels with the introduction of the RBM canon, and thus creates an invariant translation model that is easily scalable. This algorithm was further extended by Huang et al. (2012).

DEEP BOLTZMANN MACHINES (DBMS)

A DBM was proposed by Salakhutdinov and Hinton (2009), which is another deep learning algorithm in which processing units are placed in layers. The unbiased graphical model and the bottom layers form a directional productive model, the DBM having connections throughout its structure.

Similar to the RBM, the DBM is a subset of the Boltzmann family. The difference is that DBMs have multiple layers containing hidden units, which are the units in individually numbered layers conditionally independent of even-numbered layers, and vice versa.

With visible units, the calculation of posterior distribution over hidden units is no longer traceable, which is the result of interactions between hidden units. During network training, a DBM jointly trains all layers of a specific, unsupervised model, and instead of directly maximising probability, uses the stochastic maximum likelihood algorithm (SML) to maximise lower boundaries in probability (Younes, 1999). In other words, it means using the

Markov chain Monte Carlo (MCMC) method between each parameter. In order to prevent weak local minima from inactivating many hidden units, a DBN training strategy is also applied to layers in the pre-training DBM network, which is very similar to what is done in DBN (Bengio et al., 2013). Co-learning has delivered promising improvements in both the probability and efficiency of feature learner classifications. But the important drawback of DBMs is the complexity of approximation inference, which is much higher than for DBNs. This makes common optimisation of DBM parameters for large datasets impractical. To increase the efficiency of DBMs, some researchers have introduced an approximate inference algorithm (Salakhutdinov & Larochelle, 2010; Salakhutdinov & Hinton, 2012) that uses a recognition model to initialise latent variables in all layers. It effectively speeds up inference. There are many other ways to improve the effectiveness of DBMs. These improvements can occur either at the pre-training stage (Salakhutdinov & Hinton, 2012; Cho et al., 2013) or at the beginning of training (Montavon & Müller, 2012; I. J. Goodfellow et al., 2013). For example, Montavon & Müller (2012) introduced a centring trick to improve the stability of a DBM and make it more discriminative and generative. The multi-prediction training scheme was used to train the DBM joint (I. Goodfellow et al., 2013), which resulted in a far better performance than previous methods in classifying the images outlined in I. J. Goodfellow et al. (2013).

DEEP ENERGY MODELS (DEMS)

The DEM was proposed by Ngiam et al. (2011), and is a new way of teaching deep architecture. Unlike DBNs and DBMs, which share multiple stochastic hidden layers, a DEM has only one layer containing stochastic hidden units for optimal training and inference.

This model uses feed-forward neural networks to model the energy landscape and is able to train all layers at once. By evaluating the performance of this model on natural images, it is shown that multi-layer co-instruction improves the quality and quantity of DBN training (in terms of training samples). Ngiam et al. (2011) used the hybrid Monte Carlo (HMC) to train this model. There are other options, such as contrastive divergence, score matching, etc. Similar work can also be found in Cho et al. (2011) and Elfwing et al. (2015).

Although RBMs, like CNNs, are not suitable for computer vision applications, there are some good examples of RBMs being used for computer vision activities. The shape Boltzmann machine (SBM) was proposed by Eslami et al. (2014) for using in binary shape image modelling work; it also distributes high-quality probability distributions on object shapes in order to realistically derive realism from samples from the distribution and generalisation to new examples with the same shape class (Eslami et al., 2014). Kae et al. (2013) combined conditional random fields (CRF) and RBM models to model both local and global structure of face segmentation. This has led to a steady decline in face-tagging errors. A new in-depth architecture for telephone recognition (Dahl et al., 2010) combines the feature extraction module of an RBM mean-covariance with a standard DBN. This method

addresses both the representational inefficiency issues of Gaussian mixture models (GMMs) and an important limitation in previous work that DBNs used in voice recognition (Dahl et al., 2010).

Autoencoder (AE)

The AE is a special type of ANN that is used to encode learning optimally (Liou et al., 2014). Instead of network training and predicting the target value of Y for X input, you find a training AE to reconstruct your X input. Therefore, the output vectors will have the same dimension as the input vector. During this process, the AE is optimised by minimising the reconstruction error. The corresponding code is the same attribute learned. Generally, a single layer is not capable of receiving distinct features from raw data. Researchers are currently using a deep AE that sends the learned code from the previous AE to the next AE to get the job done.

The deep AE was developed by Hinton and Salakhutdinov (2006) and it is still extensively studied in recent articles (Zhang et al., 2014; Y. Zhou et al., 2014; Jiang et al., 2013). A deep AE is often trained with some kind of back-propagation operation such as the conjugate gradient method. Although this model is often efficient and effective, it can be extremely ineffective if errors occur in the first layers. This may cause the average educational data network to rebuild. A good way to eliminate this problem is to pre-train the network of initial weights that approximate the final solution (Hinton & Salakhutdinov, 2006). There are also variants of AE that suggest keeping representations as consistent as possible with changes in input.

In Table 3, a list of popular AE types is given, along with a summary of their features and benefits.

Table 3. A list of popular AE types along with a summary of their features.

Method	Attributes	Benefits
Sparse AE	Adds s sparsity penalty to force the representation to be sparse	1) Makes categories more separable 2) Makes complex data more meaningful 3) Biological version system
Denoising AE	Recovers the correct input from a corrupted version	More robust to noise
Contractive AE	Adds an analytical contractual penalty to the reconstruction error function	Better captures the local directions of variation dictated by the data
Saturating AE	Raises error reconstruction for inputs not near the data manifold	Limits the ability to reconstruct inputs that are not near the data manifold
Convolutional AE	Shares weights among all locations in the input, preserving spatial locality	Utilises 2D image structure
Zero-bias AE	Utilises proper shrinkage function to train AE without additional regularisation	More powerful in learning representations on data with very high intrinsic dimensionality

SPARSE AE (SAE)

SAEs seek to extract sparse features from raw data. Scattering of the features can be obtained either by penalising hidden unit biases (Varastehpour et al., 2019a, 2019b; Lee et al., 2008; I. Goodfellow et al., 2009; Ranzato et al., 2007) or directly penalising the output of hidden unit values (I. Goodfellow et al., 2009; Le et al., 2011). Sparse representations have several possible benefits (Ranzato et al., 2007):

- Using high-dimensional representations increases the likelihood that different clusters can be easily separated, as in SVM theory.
- Sparse representations provide us with a simple interpretation of complex input data in a number of sections.
- Biological vision uses scattered signals in the primary visual areas.

A very popular variant of the SAE is a locally-based nine-layer model with pooling and contrast normalisation (Le, 2013). This model allows the system to train a face finder without the need to label the images as face free. The resulting feature detector is very powerful for translation, scaling and out-of-plane rotation.

DENOISING AE (DAE)

To increase the power of this model, Vincent introduced a model called the de-noising AE (DAE) (Vincent et al., 2008; Vincent et al., 2010) that is able to retrieve the correct input from the corrupted version. This forces the model to capture the distribution structure of the input. The structure of a DAE is illustrated in Figure 3.

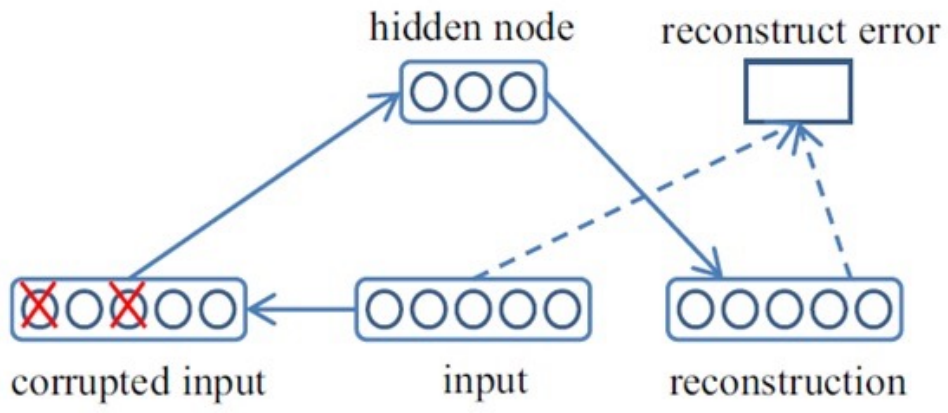

Figure 3. DAE structure (Vincent et al., 2010, p. 3379)

CONTRACTIVE AE (CAE)

CAE was proposed by Rifai et al. (2011) and is a sequence of DAE with a similar motivation for powerful representation learning (Bengio et al., 2013). While DAE powers all mapping operations by injecting noise into the training set, CAE achieves this by adding an analytical contractive penalty to the reconstruction error function.

Bengio et al. (2013) explain significant differences between DAE and CAE, but Alain and Bengio (2014) state that DAE and one form of CAE are very close together. A DAE with small corruption noise can be considered a type of CAE, where the contractive penalty is on the whole renovation function rather than the encoder. Both DAE and CAE have been used successfully in transfer learning and unsupervised problems (Mesnil et al., 2011).

ZERO-BIAS AE

Most deep learning algorithms, such as CNN, RBM, etc., rely on spatial or sequential data attributes to learn. If the data is highly sparse, then the network learns 'zeros.' In essence, there is no real learning happening. There are ways to preprocess, such as oversampling to generate dense data, but this may not always work. By oversampling data, all the information is kept in the training set. On the other hand, the random oversampling may increase overfitting since it makes exact copies of the minority class examples. In this way, a symbolic classifier, for instance, might construct rules that are accurate but cover one replicated example.

Zero bias-AE (Konda et al., 2014) is one of the deep learning algorithms that is more potent in learning representations on data with very high intrinsic dimensionality and sparse features. Zero-bias AE was proposed by Konda et al. (2014) and quantises the input space with tiles proportional in quantity to the data density. They claimed that it is arguably the best way to represent data, given enough training data and enough tiles. It allows us to approximate any function reasonably well using only a subsequent linear layer to summarise regions using responses that are invariant to some changes in the input. Invariance, from this perspective, is a necessary evil and not a goal in itself. But it is increasingly essential for increasingly high-dimensional and sparse inputs (Konda et al., 2014). The response of a hidden unit in their models is defined by multiplying the filter response or squaring it, followed by a non-linearity. To reconstruct the input, the output of the hidden unit is then multiplied by the filter response itself, making the model bi-linear. As a result, reconstructions are defined as the sum of feature vectors, weighted by the active hidden linear coefficients. This may suggest interpreting the fact that these models work well on videos and other high-dimensional and sparse data as a result of using linear, zero-bias hidden units, too (Konda et al., 2014).

Sparse Coding (SC)

SC is an unsupervised method used to learn an over-complete set of basic functions to describe input data (Olshausen & Field, 1997). Advantages of this method are as follows (Raina et al., 2007; Yu et al., 2009; Wang et al., 2010; Yang et al., 2009):

- It is able to rebuild a better descriptor by using multiple base and receive relationships between similar descriptors from those with common foundations.
- The scattering allows the representation to capture the salient features of the images.
- This is in line with the biological vision system, which argues that the scattering characteristics of signals are useful for learning.
- Studying image statistics shows that image patches are scattered signals.
- Patterns are more linearly separable with dispersed features.

In the following section, a number of SC algorithms in the field of computer vision are discussed.

SC SPATIAL PYRAMID MATCHING (SCSPM)

Sparse coding spatial pyramid matching (SCSPM) is one of the important SC algorithms that is an extension of the SPM algorithm (Lazebnik et al., 2006; Wang et al., 2010). Unlike SPM, which uses vector quantisation (VQ) for image representation, SCSPM uses SC with multi-scale spatial max pooling for this purpose. The sparse coding codebook is an over-complete basis and every feature is capable of activating a few of them. Compared to VQ, SC has a much lower reconstruction error due to the less restrictive constraint that considers the assignment action. Coates and Ng (2011) did more research on the reasons for the success of SC than vector quantisation and presented the results in a detailed paper (Coates & Ng, 2011). One drawback of SCSPM is that this method deals with local features separately and thus ignores the two-way dependency between them. This makes the feature variance too sensitive, which means that the sparse codes can change drastically, even for similar features.

LAPLACIAN SC METHOD (LSC) AND HYPERGRAPH LAPLACIAN SPARSE CODING METHOD (HLSC)

To address this problem, Gao et al. (2010) presented the Laplacian SC method (LSC), in which similar features are not only optimally selected for batch centres, but it also ensures that the centres of the selected categories are

the same. By adding locality constraints to SC objectives, the LSC method is able to maintain a two-way dependency on SC. Gao et al. (2012) developed the hypergraph Laplacian sparse coding method (HLSC), which developed the LSC. Their approach was to state the similarities among the samples by a hypergraph. Both the LSC and HLSC methods add SC power.

Another approach that can be used to solve the (over) sensitivity problem is the hierarchical SC (HSC) method developed by Yu et al. (2011). This method introduces a two-layer SC model that encodes the first layer of each patch separately, and the second layer is a joint set of patches belonging to one group. By doing this, the model makes the best use of the neighbouring space structure by modelling high order dependency patches in the same local area of the image. In addition, this is a fully automated way of learning features from the pixel level and no longer needs to manually design the scale-invariant feature transform (SIFT). HSC has been used to learn the unsupervised features in Zeiler et al. (2010). This model was later used by Zeiler et al. (2011).

LOCAL COORDINATE CODING (LCC)

In addition to sensitivity, there is another method used to improve the SCSPM algorithm. This method tries to improve the SCSPM algorithm by considering the locality. Yu et al. (2009) considered how the algorithm works; SCSPM results tend to be localised and mean non-zero coefficients are usually assigned to nearby bases. These results suggest a change in the SCSPM, referred to as local coordinate coding (LCC). This explicitly encourages coding to be localised. They also theorised that locality can enhance sparsity and that SC is useful for learning only when codes are local, so it is best to have non-zero dimensional data in codes. Although LCC is computationally superior to classical SC, it still requires time-consuming L1-norm optimisation (Yu et al., 2009). In order to expedite the learning process, a practical coding method called locality constrained linear coding (LCLC) was introduced by Wang et al. (2010), which can be seen as a quick implementation of the LCC that replaced L1-norm regularisation with L2-norm regularisation. Figure 4 shows the comparison between VQ, SCSPM and LLC.

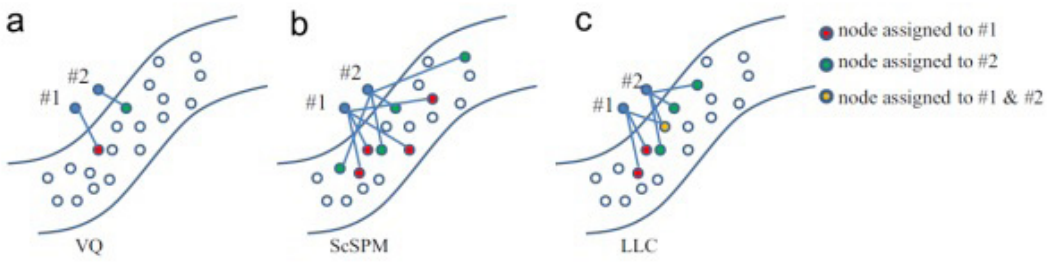

Figure 4. Comparison between VQ, SCSPM and LLC (Wang et al., 2010, p. 3).

SUPER-VECTOR CODING (SVC)

In addition to LLC, there is another model, known as super-vector coding (SVC), which guarantees local SC (X. Zhou et al., 2010). Upon receiving X, the SVC activates the coordinates associated with the neighbourhood of X to achieve sparse representation. SVC is a simple extension of VQ obtained by extending the VQ of local tangent degrees and therefore it is a more uniform coding method (X. Zhou et al., 2010).

An interesting result showed in Y. Lin et al. (2011) that the averaging stochastic gradient descent (ASGD) method is combined with the LLC and SVC algorithms to increase the scalability of the image clustering in the large dataset, and it achieved excellent results.

There is another popular method called smooth sparse coding (SSC), which is presented in Balasubramanian et al. (2013). By combining neighbourhood similarity and temporal information in SC, it was able to obtain codes that represent a neighbourhood rather than an individual sample and have a lower average square error of reconstruction (Balasubramanian et al., 2013).

Recently, He et al. (2014) introduced a new unsupervised feature learning framework called deep sparse coding (DSC), which extended SC using a multilayer architecture and achieved the best performance among sparse coding methods.

Model Selection of Deep Learning Methods

The model selection from deep learning methods (or even machine learning algorithms) is a challenging and critical factor in having the best results. In this section, the challenge of method selection for deep learning will be discussed.

MODEL SELECTION

Model selection is the process of selecting one final deep learning model from among a collection of candidate deep learning models for a training dataset (Murphy, 2012). It is a process that can be applied both across different types of methods, including CNN, AE, etc., and across methods of the same type configured with varying hyperparameters of practice (e.g., different kernels in an SVM). For example, we may have a dataset for which we are interested in developing a classification or predictive regression model. We do not know which model will perform best on this problem, as it is unknowable. Therefore, we fit and evaluate a suite of different models on the problem. Method selection is the process of choosing the final method that addresses the problem.

Model selection is different from model assessment. For example, we evaluate or assess candidate methods to choose the best one, which is method selection. Once a method is chosen, it can be evaluated to communicate how well it is expected to perform in general; this is model assessment (James et al., 2013).

CONSIDERATIONS FOR MODEL SELECTION

Fitting the model is relatively straightforward, although selecting among them is the true challenge of applied machine learning (Murphy, 2012). All models have some predictive error, given the statistical noise in the data, the incompleteness of the data sample, and the limitations of each different model type. Therefore, the notion of a perfect or best model is not useful. Instead, we must seek a model that is "good enough" (Murphy, 2012, pp. 22-23).

Different project stakeholders may have specific requirements, such as maintainability and limited model complexity. As such, a model with lower skill but that is more straightforward and easier to understand may be preferred. Alternately, method skill might be valued above all other concerns. In that case, the model's ability to perform well on out-of-sample data will be preferred regardless of the computational complexity involved. Therefore, a "good enough" method may refer to many things and is specific in different projects, such as (Murphy, 2012, pp. 22-23):

- A model that meets the requirements and constraints of project stakeholders
- A model that is sufficiently skillful given the time and resources available
- A model that is skillful as compared to naive models
- A model that is skillful relative to other tested models
- A model that is skillful comparable to the state of the art

Next, what is being selected must be considered. For example, a fit model is not specified, as all methods will be discarded. This is because once a model is chosen, a new final method will be fitted on all available data and used to start making predictions. Some algorithms require specialised data preparation to best expose the structure of the problem to the learning algorithm. Therefore, model selection should be considered as the process of selecting among model development pipelines. Each pipeline may take in the same raw training dataset and output a model that can be evaluated in the same manner, but may require different or overlapping computational steps, such as data filtering, feature selection, etc.

MODEL SELECTION TECHNIQUES

The best approach to model selection requires 'sufficient' data, which may be nearly infinite depending on the complexity of the problem. In this ideal situation, the data is split into training, validation and test sets. Then, it fits candidate methods on the training set, evaluates and selects them on the validation set, and reports the final model on the test set (Friedman et al., 2001). This is impractical on most predictive modelling problems, given that we rarely have sufficient data or can even judge what would be adequate (Biship, 2007). Instead, there are two main classes of techniques to approximate the ideal case of model selection, which are as follows:

- Probabilistic Measures: Choose a model via in-sample error and complexity
- Resampling Methods: Choose a model via estimated out-of-sample error

Probabilistic measures involve analytically scoring a candidate model using both its performance on the training dataset and the complexity of the technique. It is known that training error is optimistically biased, and therefore is not a reasonable basis for choosing a model. The performance can be penalised based on how optimistic the training error is believed to be. This is typically achieved using algorithm-specific models, often linear, that penalise the score based on the complexity of the technique (Bishop, 2007). Probabilistic measures are appropriate when using simpler linear models like linear regression or logistic regression, where the calculating of model complexity penalty (e.g., in sample bias) is known and tractable.

Resampling models seek to estimate the performance of a method (or, more precisely, the model development process) on out-of-sample data. This is achieved by splitting the training dataset into sub-train and test sets, fitting a model on the sub-train set, and evaluating it on the test set. This process may then be repeated multiple times, and the mean performance across each trial is reported (James et al., 2013). In the next section, the ten features of deep learning methods are compared and discussed.

Discussion

In order to compare these four types of deep learning methods and gain an understanding of them, a summary of the benefits and problems of each according to their various characteristics is presented in Table 4. Please note that in Table 4, 'Yes' indicates that the handle works well in this feature. Otherwise, it is marked with 'No.' '*Yes' refers to elementary or weak ability. Table 4 has a total of ten main features. 'Generalisation' refers to the way the media (image, sound, etc.) is used and various applications such as speech recognition, visual recognition, etc. 'Unsupervised learning' also refers to the ability to automatically learn a deep model without supervisory annotations. 'Feature learning' is the ability to automatically learn features based on a dataset. 'Real-time training' and 'real-time prediction' both refer to the efficiency of learning and the inferential process, respectively. 'Biological understanding' and 'theoretical justification' indicate whether the method has a biological basis or a theoretical basis. 'Invariance' refers to whether the method is resistant to transformations such as rotation, scale and translation. 'Small training set' also refers to learning a deep model using only a few examples. 'Sparse and high-dimensional data' refers to the ability to deal with sparse input data and high-dimensional data for feature extraction purposes. It is important to note that Table 4 presents only general overviews and does not provide a snapshot of future opportunities or specific examples. Table 5 shows the comparison of machine learning and deep neural network.

Deep learning has been used extensively in various areas of computer vision such as image classification, object detection, semantic segmentation, image retrieval, human gesture estimation and feature extraction. Each of these algorithms can be applied for a specific purpose. For example, to provide a solution and overcome some of the limitations on current vein pattern recognition (VPR) methods (Varastehpour et al., 2020), deep learning algorithms are able to improve the performance of the current methods of VPR in the domain of interest, as indicated above, objectively, effectively and efficiently.

Table 4: Comparison of ten features between CNN, RBM, AE and SC.

Features	**CNN**	**RBM**	**AE**	**SC**
Generalisation	Yes	Yes	Yes	Yes
Unsupervised learning	No	Yes	Yes	Yes
Feature learning	Yes	*Yes	Yes	No
Real-time training	No	No	Yes	Yes
Real-time prediction	Yes	Yes	Yes	Yes
Biological understanding	No	No	No	Yes
Theoretical justification	*Yes	Yes	Yes	Yes
Invariance	*Yes	No	No	Yes
Small training set	*Yes	*Yes	Yes	No
Sparse and high-dimensional data	No	No	*Yes	No

Table 5. Comparison of machine learning and deep neural network.

Machine learning	**Deep neural network**
Machine learning uses algorithms to parse data, learn from that data, and make informed decisions based on what it has learned	Deep learning structures algorithms in layers to create an 'artificial neural network' that can learn and make intelligent decisions on its own
Can train on less training data	Requires large data sets for training
Takes less time to train	Takes a longer time to train
Trains on CPU	Trains on GPU and CPU for proper training
The output is in numerical form for classification and scoring applications	The output can be in any form, including free-form elements such as free text and sound
Limited tuning capability for hyperparameter tuning	It can be tuned in various ways

Conclusion

Deep learning algorithms have received great attention over the last decade due to their high performance. They have helped in the fields of computer vision and image processing to improve their efficiency. The four key categories of deep learning for computer vision and image processing that have been reviewed in this literature are CNN, RBM, autoencoder, and sparse coding-based methods. They have been employed with different performance rates in a variety of features, such as generalisation, unsupervised learning, feature learning, biological understanding, etc. However, each category has distinct advantages and disadvantages. CNN and autoencoder-based methods have the unique capability of feature learning; that is, of automatically learning features based on the given dataset. Sparse coding-based methods are invariant to transformations, which is a great asset for certain computer vision applications and it is also useful in terms of biological learning, while RBM, autoencoder, and CNN-based methods are not. Of the models investigated, both CNNs and RBMs are computationally demanding when it comes to training, whereas autoencoder and sparse coding can be trained in real time under certain circumstances. However, these deep learning algorithms need to be investigated more in the future based on the new demand in the field of computer vision to which a specific architecture or algorithm is effective in a given task or not. These algorithms are among the most important topics that will continue to attract the interest of the machine learning research community in the years to come.

References

Alain, G., & Bengio, Y. (2014). What regularised auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, *15*(1), 3743–3773. https://jmlr.csail.mit.edu/papers/volume15/alain14a/alain14a.pdf

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning – a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, *5*(4), 13–18. https://ieeexplore.ieee.org/document/5605630/authors

Balasubramanian, K., Yu, K., & Lebanon, G. (2013). Smooth sparse coding via marginal regression for learning sparse representations. *Proceedings of the 30th International Conference on Machine Learning*, *PMLR*, *28*(3), 289–297. http://proceedings.mlr.press/v28/balasubramanian13.html

Bengio, Y. (2013). Deep learning of representations: Looking forward. *International Conference on Statistical Language and Speech Processing*, 1–37. https://link.springer.com/chapter/10.1007/978-3-642-39593-2_1

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828. https://ieeexplore.ieee.org/document/6472238

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127. https://doi.org/10.1561/2200000006

Biship, C. M. (2007). *Pattern recognition and machine learning (information science and statistics)*. Springer.

Boureau, Y-L., Ponce, J., & Lecun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 111–118. https://www.researchgate.net/publication/221345753_A_Theoretical_Analysis_of_Feature_Pooling_in_Visual_Recognition

Brunette, E. S., Flemmer, R. C., & Flemmer, C. L. (2009). A review of artificial intelligence. *2009 4th International Conference on Autonomous Robots and Agents*, 385–392. https://doi.org/10.1109/ICARA.2000.4804025

Cao, Z., Shaomin, M. U., Yongyu, X. U., & Dong, M. (2018). Image retrieval method based on CNN and dimension reduction. *International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 441–445. https://ieeexplore.ieee.org/abstract/document/8965601

Carreira-Perpinan, M. A., & Hinton, G. E. (2005). On contrastive divergence learning. *Aistats*, *10*, 33–40. https://www.cs.toronto.edu/~hinton/absps/cdmiguel.pdf

Cho, K., Raiko, T., & Ihler, A. T. (2011). Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 105–112. https://dl.acm.org/doi/abs/10.5555/3104482.3104496

Cho, K., Raiko, T., Ilin, A., & Karhunen, J. (2013). A two-stage pretraining algorithm for Deep Boltzmann Machines. *International Conference on Artificial Neural Networks*, 106–113. https://link.springer.com/chapter/10.1007/978-3-642-40728-4_14

Cireșan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. *IEEE Conference on Computer Vision and Pattern Recognition*, 3642–3649. https://ieeexplore.ieee.org/document/6248110

Cireșan, D. C., Meier, U., Masci, J., Gambardella, L. M., & Schmidhuber, J. (2011). High-performance neural networks for visual object classification. https://arxiv.org/pdf/1102.0183.pdf

Coates, A., & Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 921–928. https://dl.acm.org/doi/10.5555/3104482.3104598

Dahl, G., Ranzato, M., Mohamed, A-r., & Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. *Advances in Neural Information Processing Systems*, 469–477. https://www.researchgate.net/publication/221620570_Phone_Recognition_with_the_Mean-Covariance_Restricted_Boltzmann_Machine

Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. *APSIPA Transactions on Signal and Information Processing* (Vol. 3). Cambridge University Press.

Dutton, D. M., & Conroy, G. V. (1997). A review of machine learning. *The Knowledge Engineering Review*, *12*(4), 341–367.

Elfwing, S., Uchibe, E., & Doya, K. (2015). Expected energy-based restricted Boltzmann machine for classification. *Neural Networks*, *64*, 29–38. https://doi.org/10.1016/j.neunet.2014.09.006

Eslami, S. A., Heess, N., Williams, C. K., & Winn, J. (2014). The shape Boltzmann machine: A strong model of object shape. *International Journal of Computer Vision*, *107*(2), 155–176. https://doi.org/10.1007%2Fs11263-013-0669-1

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* (Vol. 1, No. 10). Springer.

Gao, S., Tsang, I. W-H., & Chia, L-T. (2012). Laplacian sparse coding, Hypergraph Laplacian sparse coding, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(1), 92–104. https://doi.org/10.1109/TPAMI.2012.63

Gao, S., Tsang, I. W-H., Chia, L-T., & Zhao, P. (2010). Local features are not lonely – Laplacian sparse coding for image classification. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2010.5539943

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 580–587. https://ieeexplore.ieee.org/document/6909475

Gkioxari, G., Girshick, R., & Malik, J. (2015). Contextual action recognition with r* CNN. *Proceedings of the IEEE International Conference on Computer Vision*, 1080–1088. https://openaccess.thecvf.com/content_iccv_2015/html/Gkioxari_Contextual_Action_Recognition_ICCV_2015_paper.html

Goodfellow, I., Lee, H., Le, Q. V., Saxe, A., & Ng, A. Y. (2009). Measuring invariances in deep networks. *Advances in Neural Information Processing Systems*, 646–654. https://ai.stanford.edu/~ang/papers/nips09-MeasuringInvariancesDeepNetworks.pdf

Goodfellow, I., Mirza, M., Courville, A., & Bengio, Y. (2013). Multi-prediction deep Boltzmann machines. *Advances in Neural Information Processing Systems*, *1*, 548–556. https://proceedings.neurips.cc/paper/2013/file/0bb4aec1710521c12ee76289d9440817-Paper.pdf

Goodfellow, I. J., Courville, A., & Bengio, Y. (2013). Joint training deep Boltzmann machines for classification. https://arxiv.org/pdf/1301.3568.pdf

Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., & Lew, M. S. (2016). Deep learning for visual understanding: A review. *Neuro Computing*, *187*, 27–48. https://doi.org/10.1016/j.neucom.2015.09.116

Hariharan, B., Arbeláez, P., Girshick, R., & Malik, J. (2014). Simultaneous detection and segmentation. *European Conference on Computer Vision*, 297–312. https://link.springer.com/chapter/10.1007/978-3-319-10584-0_20

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1904–1916. https://doi.org/10.1109/TPAMI.2015.2389824

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. https://doi.org/10.1109/cvpr.2016.90

He, Y., Kavukcuoglu, K., Wang, Y., Szlam, A., & Qi, Y. (2014). Unsupervised feature learning by deep sparse coding. *Proceedings of the 2014 SIAM International Conference on Data Mining*, 902–910. https://epubs.siam.org/doi/abs/10.1137/1.9781611973440.103

Hinton, G. E. (2012). A practical guide to training restricted Boltzmann machines. In G. Montavon, G. B. Orr, & K-R. Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 599–619). Springer.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, *1*, 282–317. https://dl.acm.org/doi/10.5555/104279.104291

Huang, G. B., Lee, H., & Learned-Miller, E. (2012). Learning hierarchical representations for face verification with convolutional deep belief networks. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2518–2525. https://ieeexplore.ieee.org/document/6247968

Jiang, X., Zhang, Y., Zhang, W., & Xiao, X. (2013). A novel sparse auto-encoder for deep unsupervised learning. *2013 Sixth International Conference on Advanced Computational Intelligence (ICACI)*, 256–261. https://ieeexplore.ieee.org/document/6748512

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). Springer.

Kae, A., Sohn, K., Lee, H., & Learned-Miller, E. (2013). Augmenting CRFs with Boltzmann machine shape priors for image labeling. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019–2026. https://doi.org/10.1109/CVPR.2013.263

Konda, K., Memisevic, R., & Krueger, D. (2014). *Zero-bias autoencoders and the benefits of co-adapting features*. Cornell University. https://arxiv.org/pdf/1402.3337.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 1097–1105. https://papers.nips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

Lazebnik, S., Schmid, C., & Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognising natural scene categories. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, *2*, 2169–2178. https://ieeexplore.ieee.org/document/1641019

LeCun, Y., Jackel, L. D., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Muller, U. A., Sackinger, E., Simard, P., & Vapnik, V. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. In J. H. Oh, C. Kwon, & S. Cho (Eds.), *Neural networks: The statistical mechanics perspective* (pp. 261–276). World Scientific.

Le, Q. V. (2013). Building high-level features using large scale unsupervised learning. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 8595–8598. https://ieeexplore.ieee.org/document/6639343

Le, Q. V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., & Ng, A. Y. (2011). On optimisation methods for deep learning. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 265–272. https://dl.acm.org/doi/10.5555/3104482.3104516

Lee, H., Ekanadham, C., & Ng, A. Y. (2008). Sparse deep belief net model for visual area v2. *Advances in Neural Information Processing Systems*, 873–880. https://papers.nips.cc/paper/2007/hash/4daa3db355ef2b0e64b472968cb70f0d-Abstract.html

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning*, 609–616. https://doi.org/10.1145/1553374.1553453

Lin, M., Chen, Q., & Yan, S. (2013). Network in network. https://arxiv.org/pdf/1312.4400.pdf

Lin, Y., Lv, F., Zhu, S., Yang, M., Cour, T., Yu, K., Cao, L., & Huang, T. (2011). Large-scale image classification: fast feature extraction and SVM training. *2011 IEEE Conference on Computer Vision and Pattern Recognition*, 1689–1696. https://experts.illinois.edu/en/publications/large-scale-image-classification-fast-feature-extraction-and-svm-

Liou, C-Y., Cheng, W-C., Liou, J-W., & Liou, D-R. (2014). Autoencoder for words. *Neurocomputing*, *139*, 84–96.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440. https://openaccess.thecvf.com/content_cvpr_2015/papers/Long_Fully_Convolutional_Networks_2015_CVPR_paper.pdf

Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., & Bergstra, J. (2011). Unsupervised and transfer learning challenge: A deep learning approach. *Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop*, *27*, 97–110. http://proceedings.mlr.press/v27/mesnil12a.html

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 3111–3119. https://papers.nips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Montavon, G., & Müller, K-R. (2012). Deep Boltzmann machines and the centering trick. In G. Montavon, G. Orr, & K-R Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 621–637). Springer.

Murphy, K. P. (2012). *Machine learning: A probabilistic perspective*. MIT Press.

Nair, V., & Hinton, G. E. (2009). 3D object recognition with deep belief nets. *Advances in Neural Information Processing Systems*, 1339–1347. https://papers.nips.cc/paper/2009/hash/6e7b33fdea3adc80ebd648fffb665bb8-Abstract.html

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 807–814. https://dl.acm.org/doi/10.5555/3104322.3104425

Ngiam, J., Chen, Z., Koh, P. W., & Ng, A. Y. (2011). Learning deep energy models. *Proceedings of the 28th International Conference on Machine Learning*, 1105–1112. https://dl.acm.org/doi/10.5555/3104482.3104621

Olshausen, B. A., & Field, D. J. (1997). Sparse coding with an over complete basis set: A strategy employed by v1? *Vision Research*, *37*(23), 3311–3325. https://doi.org/10.1016/S0042-6989(97)00169-7

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1717–1724. https://ieeexplore.ieee.org/document/6909618

Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2015). Is object localisation for free? Weakly-supervised learning with convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 685–694. https://doi.org/10.1109/CVPR.2015.7298668

Ouyang, W., Luo, P., Zeng, X., Qiu, S., Tian, Y., & Li, H. (2014). DeepID-net: Multi-stage and deformable deep convolutional neural networks for object detection. https://arxiv.org/abs/1409.3505

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. *Proceedings of the 24th International Conference on Machine Learning*, 759–766. https://doi.org/10.1145/1273496.1273592

Ranzato, M., Poultney, C., Chopra, S., & Cun, Y. L. (2007). Efficient learning of sparse representations with an energy-based model. *Advances in Neural Information Processing Systems*, 1137–1144. https://proceedings.neurips.cc/paper/2006/file/87f4d79e36d68c3031ccf6c55e9bbd39-Paper.pdf

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 91–99. https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Rifai, S., Vincent, P., Müller, X., Glorot, X., & Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 833–840. https://icml.cc/2011/papers/455_icmlpaper.pdf

Salakhutdinov, R., & Hinton, G. (2009). Deep Boltzmann machines. *Artificial Intelligence and Statistics*, 448–455. http://proceedings.mlr.press/v5/salakhutdinov09a/salakhutdinov09a.pdf

Salakhutdinov, R. R., & Hinton, G. E. (2012). A better way to pretrain deep Boltzmann machines. *Advances in Neural Information Processing Systems*, *2*, 2447–2455. https://dl.acm.org/doi/10.5555/2999325.2999408

Salakhutdinov, R., & Larochelle, H. (2010). Efficient learning of deep Boltzmann machines. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 693–700. http://proceedings.mlr.press/v9/salakhutdinov10a/salakhutdinov10a.pdf

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In K, Diamantaras, W. Duch, & L. S. Iliadis (Eds.), *International Conference on Artificial Neural Networks* (pp. 92–101). Springer.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. https://arxiv.org/pdf/1409.1556.pdf

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Tang, Y., & Eliasmith, C. (2010). Deep networks for robust visual recognition. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1055–1062. https://icml.cc/Conferences/2010/papers/370.pdf

Varastehpour, S., Sharifzadeh, H., Ardekani, I., & Francis, X. (2019a). An adaptive method for vein recognition enhancement using deep learning. *2019 19th International Symposium on Signal Processing and Information Technology (ISSPIT)*, 1–5.

Varastehpour, S., Sharifzadeh, H., Ardekani, I., & Francis, X. (2019b). Vein pattern visualisation and feature extraction using sparse auto-encoder for forensic purposes. 2019 16th IEEE International *Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–8. https://doi.org/10.1109/AVSS.2019.8909860

Varastehpour, S., Sharifzadeh, H., Ardekani, I., & Sarrafzadeh, A. (2020). Human biometric traits: A systematic review focusing on vascular pat- terns. Unitec ePress Occasional and Discussion Papers Series (2020/3). https://www.unitec.ac.nz/epress/index.php/human-biometric-traits-a-systematic-review-focusing-on-vascular-patterns/

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning*, 1096–1103. https://doi.org/10.1145/1390156.1390294

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., & Manzagol, P-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, *11* (December), 3371–3408. https://dl.acm.org/doi/10.5555/1756006.1953039

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 3360– 3367. https://ieeexplore.ieee.org/document/5540018

Xie, S., & Tu, Z. (2015). Holistically-nested edge detection. *Proceedings of the IEEE International Conference on Computer Vision*, *1395–1403*. https://doi.org/10.1109/ICCV.2015.164

Yang, J., Yu, K., Gong, Y., & Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 1794–1801. https://doi.org/10.1109/CVPR.2009.5206757

Yoo, D., Park, S., Lee, J-Y., & So Kweon, I. (2015). Multi-scale pyramid pooling for deep convolutional representation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 71–80. https://doi.org/10.1109/CVPRW.2015.7301274

Younes, L. (1999). On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, *65*(3-4), 177–228. https://doi.org/10.1080/17442509908834179

Yu, K., Lin, Y., & Lafferty, J. (2011). Learning image representations from the pixel level via hierarchical sparse coding. *CVPR 2011*, 1713–1720. https://doi.org/10.1109/CVPR.2011.5995732

Yu, K., Zhang, T., & Gong, Y. (2009). Nonlinear learning using local coordinate coding. *Advances in Neural Information Processing Systems*, 2223–2231. https://www.researchgate.net/publication/221618703_Nonlinear_Learning_using_Local_Coordinate_Coding

Zeiler, M. D. (2013). *Hierarchical convolutional deep learning in computer vision* [Unpublished doctoral dissertation]. New York University.

Zeiler, M. D., & Fergus, R. (2013). Stochastic pooling for regularisation of deep convolutional neural networks. https://arxiv.org/pdf/1301.3557.pdf

Zeiler, M. D., & Fergus, R. (2014). Visualising and understanding convolutional networks. *European Conference on Computer Vision*, 818–833.

Zeiler, M. D., Krishnan, D., Taylor, G. W., & Fergus, R. (2010). De-convolutional networks. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2528–2535. https://doi.org/10.1109/CVPR.2010.5539957

Zeiler, M. D., Taylor, G. W., & Fergus, R. (2011). Adaptive de convolutional networks for mid and high level feature learning. *Proceedings of the IEEE International Conference on Computer Vision, 1*(2), 6. https://doi.org/10.1109/ICCV.2011.6126474

Zhang, J., Shan, S., Kan, M., & Chen, X. (2014). Coarse-to-fine autoencoder networks (CFAN) for real-time face alignment. *European Conference on Computer Vision*, 1–16. https://link.springer.com/chapter/10.1007/978-3-319-10605-2_1

Zhou, X., Yu, K., Zhang, T., & Huang, T. S. (2010). Image classification using super-vector coding of local image descriptors. *European Conference on Computer Vision*, 141–154. https://link.springer.com/chapter/10.1007/978-3-642-15555-0_11

Zhou, Y., Arpit, D., Nwogu, I., & Govindaraju, V. (2014). Is joint training better for deep auto-encoders? https://arxiv.org/abs/1405.1380

Zhu, Y., Urtasun, R., Salakhutdinov, R., & Fidler, S. (2015). SegDeepM: Exploiting segmentation and context in deep neural networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4703–4710. https://arxiv.org/abs/1502.04275

AUTHORS

Dr Soheil Varastehpour is a Lecturer in the School of Computing, Electrical and Applied Technology at Unitec New Zealand.

Dr Hamid Sharifzadeh is an Associate Professor in the School of Computing, Electrical and Applied Technology at Unitec New Zealand.

Dr Iman Ardekani is an Associate Professor in the School of Computing, Electrical and Applied Technology at Unitec New Zealand.

ACKNOWLEDGMENTS

The authors gratefully acknowledge Associate Professor Marcus Williams, Dr Kerry Kirkland and Marie Shannon for their continuous support.