Journal of Information Systems Engineering and Management
2025, 10(13s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Deep Dive into Training Algorithms for Deep Belief Networks

Dr.S. Manohar[1], Raji.N[2],
[1]*Associate Professor, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram Campus Chennai, India. manohars1@srmist.edu.in*
[2]*Research Scholar, Department of Computer Science and Engineering, SRM Institute of Science and Technology, Ramapuram Campus Chennai, India. E-mail: rn5525@srmist.edu.in*

ARTICLE INFO

Received: 30 Nov 2024
Revised: 16 Jan 2025
Accepted: 30 Jan 2025

ABSTRACT

Deep Belief Networks (DBNs) have emerged as powerful tools for feature learning, representation, and generative modeling. This paper presents a comprehensive exploration of the various training algorithms employed in the training of DBNs. DBNs, composed of multiple layers of stochastic hidden units, have found applications in diverse domains such as computer vision, natural language processing, and bioinformatics. The paper begins by delving into the pre-training phase, where Restricted Boltzmann Machines (RBMs) play a central role. We review the Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD) algorithms, shedding light on their strengths and weaknesses in initializing deep networks. Emphasis is placed on their applicability to different data types and scales. Moving to the fine-tuning stage, the paper explores the use of backpropagation with gradient descent, discussing modern optimization techniques, including stochastic gradient descent and adaptive learning rate methods. We also examine regularization techniques like dropout and weight decay to address overfitting concerns. Furthermore, we discuss architectural variants of DBNs, such as Convolutional Deep Belief Networks (CDBNs) for image data and Recurrent DBNs for sequential data. We highlight the adaptation of DBNs for specific tasks, including classification, regression, clustering, and generative modeling.

Keywords: Deep Belief Networks, Contrastive Divergence, Pre-training, fine tuning.

INTRODUCTION

Deep Belief Networks (DBNs) [1,2] are a class of deep learning models that have gained significant attention and popularity in the field of artificial intelligence and machine learning. DBNs are a type of neural network architecture designed to learn hierarchical representations of data, making them particularly well-suited for tasks such as feature learning, dimensionality reduction, and generative modelling.

DBNs are composed of multiple layers of interconnected units, typically organized into two main types of layers: the visible layer and the hidden layers. The visible layer represents the input data, while the hidden layers are responsible for capturing progressively abstract and higher-level features from the input. The hidden layers are where the "belief" part of the network comes into play, as they aim to model complex patterns and relationships in the data.

RBMs[3] are a fundamental component of DBNs. They are used for pre-training the network in an unsupervised manner. RBMs are two-layer neural networks that use a probabilistic approach to model the relationships between the visible and hidden layers. DBNs are typically trained in a layer-wise fashion. This means that each layer is pretrained as an RBM before the entire network is fine-tuned using backpropagation. Layer-wise pre-training helps initialize the network's weights in a way that facilitates the learning of useful features.

DBNs are capable of generative modelling, which means they can generate new data samples that are similar to the training data. This property makes them useful for tasks such as image generation, text generation, and anomaly detection. DBNs are considered "deep" because they have multiple hidden layers. This depth allows them to capture complex, hierarchical patterns in the data, which can be especially advantageous for tasks involving high-dimensional or structured data.

DBNs have found applications in a wide range of domains, including computer vision, natural language processing[4,5], speech recognition, recommendation systems, and bioinformatics. They excel at tasks such as image

and speech recognition, as well as feature learning for subsequent machine learning tasks. Training deep belief networks can be computationally intensive and may require careful hyperparameter tuning. Additionally, the choice of training algorithms and regularization techniques can impact the performance of DBNs.

Overall, Deep Belief Networks have played a significant role in advancing the capabilities of machine learning models, particularly in tasks where hierarchical feature extraction and generative modelling are essential. While they have been somewhat overshadowed by other deep learning architectures like convolutional neural networks (CNNs) and recurrent neural networks (RNNs), DBNs continue to be a valuable tool in the machine learning toolkit for various applications.

TRAINING ALGORITHMS

A comprehensive exploration of the various training algorithms employed in the training of Deep Belief Networks (DBNs) is essential for understanding how these networks learn hierarchical representations from data. DBNs, composed of multiple layers of stochastic hidden units, require specialized training techniques to capture complex patterns and relationships in the data. Let's explore some of the key training algorithms used in DBNs:

Contrastive Divergence[8-10] (CD) is a popular algorithm for training the individual layers of DBNs, particularly the Restricted Boltzmann Machines[6,7] (RBMs) that compose the network. It operates by approximating the gradient of the log-likelihood of the data. CD iteratively updates the model's parameters to minimize the difference between the data distribution and the model's distribution. Persistent Contrastive Divergence (PCD) is an extension of CD and is used to address the slow convergence of CD. PCD maintains a persistent Markov chain of samples from the model's distribution, which speeds up training and improves stability.

After pre-training the individual layers with RBMs, fine-tuning the entire DBN involves using SGD with backpropagation. SGD adjusts the weights of the network in the direction that minimizes a specified loss function (e.g., cross-entropy for classification tasks). Modern variants of SGD, such as mini-batch SGD, help improve convergence and efficiency. Techniques like Adagrad, Adadelta, and Adam adjust the learning rate during training based on the past gradients, helping to accelerate convergence and improve training stability.

Weight Decay (L2 Regularization[11]) is a regularization technique used to prevent overfitting by adding a penalty term to the loss function. It encourages the model to have smaller weights, which can help generalize better to unseen data. Dropout is a regularization technique that randomly deactivates a fraction of neurons during each training iteration. This prevents the network from relying too heavily on any single neuron and can mitigate overfitting.

Batch Normalization[12-15] is a technique that normalizes the activations of each layer, reducing internal covariate shift and speeding up training. It has been found effective in training deep networks, including DBNs. The fine-tuning phase involves training the entire DBN as a deep neural network for the specific task at hand, such as classification or regression. The choice of loss function and performance metrics depends on the task.

In some variations of DBNs, variational inference techniques are used to approximate the posterior distribution over hidden variables. Variational autoencoders[16,17] (VAEs) are an example of DBNs that incorporate variational inference. Bayesian Deep Belief Networks extend DBNs with Bayesian methods to provide uncertainty estimates in predictions. Techniques like dropout can also be used to estimate uncertainty.

PRETRAINING PHASE

The pre-training phase of Deep Belief Networks (DBNs) is a crucial step in the training process that initializes the network's weights and biases in an unsupervised manner. This phase lays the foundation for the subsequent fine-tuning stage using supervised learning. Pre-training with DBNs typically involves the use of Restricted Boltzmann Machines (RBMs) and aims to capture hierarchical features from the data.

A. STEPS IN PRETRAINING

a) Initialization of RBMs: The first step is to initialize one RBM for each pair of consecutive layers in the DBN. If you have a three-layer DBN, you would initialize two RBMs: one between the visible and first hidden layer and another between the first and second hidden layer.

b) Unsupervised Learning with RBMs: Each RBM is trained independently using unsupervised learning. During this training, the RBM learns to capture statistical patterns and dependencies within the data without the need for labeled information.

c) Contrastive Divergence (CD) Training: The most commonly used algorithm for training RBMs in the pre-training phase is Contrastive Divergence (CD). CD is an iterative algorithm that approximates the gradient of the log-likelihood of the data. It involves the following steps:

•Positive Phase: Compute the expected values of the hidden units given the input data (positive phase).

•Negative Phase: Sample from the hidden units to reconstruct the input (negative phase).

•Update Weights: Adjust the weights and biases of the RBM to minimize the difference between the positive and negative phases.

•Multiple RBM Layers: If you have a deep architecture with multiple hidden layers, you repeat the RBM training process layer by layer. The output of one RBM becomes the input to the next one. This is referred to as layer-wise pre-training.

d)Stacking RBMs to Form a DBN: After training all the RBMs, you stack them together to create the deep architecture of the DBN. The connections between layers are set based on the trained RBM parameters.

e)Fine-Tuning: Once the DBN is constructed with initialized weights, it undergoes a fine-tuning phase. Fine-tuning involves training the entire network using supervised learning, typically with backpropagation and gradient descent. The weights are adjusted to minimize a specific loss function related to the task at hand, such as classification or regression.

The primary purpose of pre-training with RBMs is to provide a good initialization for the DBN's weights. This initialization helps overcome the challenges of training deep networks, such as the vanishing gradient problem, by starting with weights that capture useful hierarchical features. The fine-tuning phase further refines the network's weights to make them task-specific.

Here's a tabular review of Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD), highlighting their strengths and weaknesses in initializing deep networks.

Table1. CD vs PCD

Strengths/ Weakness	***CD***	***PCD***
Simplicity	Simple and easy to understand	Relatively simple to implement
Efficiency	Converges quickly in early stages	Faster convergence than CD
Widely Used	Commonly applied in practice	Widely used in training deep networks
Sampling Noise	Sensitive to sampling noise	Mitigates sampling noise through persistence
Slow Convergence	Slower convergence, especially in deep networks	Faster convergence, particularly in deep networks
Initialization Dependency	Sensitive to initialization	Improved weight initialization compared to CD
ComputationCost	Lower computational cost	Higher memory requirements due to persistent chain

FINE TUNING PHASE

The fine-tuning stage of Deep Belief Networks (DBNs) is a critical step in the training process that follows the pre-training phase, where each layer of the DBN is initialized using unsupervised learning techniques like Restricted

Boltzmann Machines (RBMs). In the fine-tuning stage, the entire DBN is trained as a deep neural network using supervised learning to adapt it for a specific task. This is typically a classification or regression task.

•Data Preparation: Before fine-tuning, you need to prepare your dataset. Ensure that it is divided into training, validation, and test sets, with labels or target values provided for supervised learning.

•Architecture Adaptation: Depending on your task, you may need to adapt the architecture of the DBN. For classification tasks, the top hidden layer of the DBN can be converted into a SoftMax layer, while for regression tasks, it can be modified accordingly.

•Objective Function (Loss Function): Define an appropriate loss function that measures the error or discrepancy between the network's predictions and the ground truth labels or target values. Common loss functions include mean squared error (MSE) for regression and cross-entropy for classification.

•Gradient-Based Optimization: Employ gradient-based optimization techniques to update the network's weights and biases. The most common optimization algorithm is stochastic gradient descent[18,19] (SGD). Modern variants like Adam, RMSprop, or Adagrad are often used to accelerate convergence and improve performance.

•Mini-Batch Training: Divide the training dataset into mini-batches to compute gradient updates more efficiently. Each mini-batch is used to update the model's parameters, and this process is repeated iteratively.

•Backpropagation: Perform backpropagation to compute the gradients of the loss function with respect to the model's parameters. The gradients are used to update the weights and biases of the DBN in a direction that minimizes the loss.

•Regularization Techniques: To prevent overfitting, employ regularization techniques such as dropout, weight decay (L2 regularization), or early stopping. These techniques help the model generalize better to unseen data.

•Hyperparameter[22] Tuning: Experiment with hyperparameters like learning rate, batch size, and the number of training epochs to optimize the fine-tuning process. Cross-validation on the validation set can help identify the best hyperparameters.

•Monitoring and Validation: Continuously monitor the performance of the DBN on the validation set during training. This allows you to detect overfitting and adjust regularization or other hyperparameters as needed.

•Evaluation: After training, evaluate the fine-tuned DBN on a separate test set to assess its performance. Common evaluation metrics include accuracy, precision, recall, F1-score for classification tasks, or metrics like RMSE and R-squared for regression tasks.

•Deployment: Once the DBN is fine-tuned and evaluated satisfactorily, it can be deployed for making predictions on new, unseen data in real-world applications.

The fine-tuning stage transforms the DBN, initially trained through unsupervised learning, into a supervised learning model capable of performing specific tasks. It leverages the hierarchical features learned during pre-training and further adapts them to the task at hand through gradient-based optimization and supervised learning techniques.

A. ARCHITECTURAL VARIANTS

Architectural variants of Deep Belief Networks (DBNs) extend the basic DBN structure to address specific types of data and tasks. Two prominent architectural variants are Convolutional Deep Belief Networks (CDBNs) for image data and Recurrent Deep Belief Networks (RDBNs) for sequential data.

Convolutional Deep Belief Networks (CDBNs) are a type of DBN architecture designed for processing structured grid data, particularly images and spatial data. They incorporate convolutional layers inspired by Convolutional Neural Networks (CNNs) to capture spatial hierarchies and local patterns. CDBNs contain one or more convolutional layers, each consisting of multiple learnable filters that scan the input image to detect local patterns. These filters are trained to capture features like edges, textures, and simple shapes.

Similar to CNNs, CDBNs often include pooling layers (e.g., max-pooling) to down sample feature maps and reduce spatial dimensions while preserving essential information. CDBNs use multiple convolutional and pooling layers to learn hierarchical representations of image data. Each layer captures increasingly abstract and complex features.

CDBNs excel in computer vision tasks, such as image classification, object detection, and image generation. They have been instrumental in achieving state-of-the-art results on image-related challenges.

Recurrent Deep Belief Networks (RDBNs) are an architectural variant of DBNs designed for sequential data, where the order of data points matters, such as time series, natural language, and speech. RDBNs incorporate recurrent neural network (RNN) layers, such as Long Short-Term Memory[20] (LSTM) or Gated Recurrent Unit[21] (GRU) cells, instead of traditional feedforward layers. These recurrent layers enable the network to capture temporal dependencies and sequences.

RDBNs consider the temporal relationships between data points, making them suitable for tasks like time series forecasting, speech recognition, natural language processing, and video analysis. Similar to the hierarchical feature extraction in standard DBNs, RDBNs learn hierarchical temporal abstractions by stacking recurrent layers to capture complex patterns over time. RDBNs find applications in various sequential data tasks, including speech recognition, machine translation, sentiment analysis, and predicting stock prices.

In practice, hybrid architectures that combine CDBNs and RDBNs can be used for tasks involving both spatial and temporal data, such as video analysis and action recognition. These hybrid models leverage the strengths of both architectures to capture complex spatiotemporal patterns. Overall, architectural variants of DBNs, such as CDBNs and RDBNs, are tailored to specific data types and domains, allowing them to excel in tasks where the inherent structure and characteristics of the data play a crucial role. Researchers continue to explore and develop new architectural variants to address diverse data modalities and challenges in machine learning and artificial intelligence.

B. TRADE-OFFS

• Depth vs. Width: A trade-off exists between network depth and width. Deeper networks can capture more complex features but are computationally expensive and may suffer from vanishing gradients during training. Wider networks have more parameters, which can lead to overfitting.

•Overfitting vs. Generalization: Striking the right balance between preventing overfitting and achieving good generalization is a critical trade-off. Regularization techniques like dropout and weight decay help mitigate overfitting but may require careful tuning.

•Computational Resources vs. Model Complexity: Training deep networks, including DBNs, demands substantial computational resources, including powerful GPUs or TPUs. Increasing model complexity (e.g., more layers and neurons) often requires even more resources.

C. CHALLENGES

•Vanishing[23-25] and Exploding Gradients: As DBNs get deeper, gradients can become very small (vanishing) or very large (exploding) during backpropagation. Techniques like gradient clipping and careful weight initialization are used to address these issues.

•Data Availability: Deep networks, including DBNs, require large amounts of labeled data for effective training. Obtaining labeled data can be expensive and time-consuming, especially for niche or specialized domains.

•Choice of Hyperparameters: Selecting appropriate hyperparameters, such as learning rates, batch sizes, and regularization strengths, can be challenging. Grid search, random search, and automated hyperparameter tuning methods help in this regard.

D. TRENDS

•Transfer Learning[26,27]: Pre-training DBNs on large datasets and fine-tuning them for specific tasks has been a successful trend. Models like convolutional neural networks (CNNs) and transformer-based architectures are often pre-trained on vast corpora and fine-tuned for various tasks.

•AutoML and Neural Architecture Search: Automated Machine Learning (AutoML) and Neural Architecture Search (NAS) aim to automate the process of finding optimal neural network architectures and hyperparameters, reducing the burden on practitioners.

•Sparse and Efficient Architectures: Research into making deep networks more computationally efficient and compact continues. Sparse activation techniques, quantization, and model compression are explored to deploy deep models on resource-constrained devices.

•Self-Supervised Learning: Self-supervised learning techniques, where networks are trained to predict parts of their input data, have gained attention as a way to leverage unlabeled data for pre-training.

•Explainability and Fairness: As deep learning models become increasingly powerful, there is a growing focus on interpretability, explainability, and fairness in AI systems. Research is ongoing to make DBNs more transparent and to address bias and fairness issues.

•Few-shot Learning: Techniques that enable models to learn from very few examples are a trending research area. Meta-learning and few-shot adaptation methods are being explored to improve the efficiency of model training and application.

ADAPTATION OF DBN

Deep Belief Networks (DBNs) are versatile neural network architectures that can be adapted to various machine learning tasks, including classification, regression, clustering, and generative modelling. DBNs can be customized for each of the following tasks:

A. CLASSIFICATION

a) Adaptation: To use DBNs for classification tasks, you typically replace the top layer of the DBN with a softmax layer or a sigmoid layer (for binary classification). This modified top layer allows the network to output class probabilities.

b) Process: After pre-training the DBN using unsupervised learning, you fine-tune the network using supervised learning. You feed labelled data to the network and train it to minimize a classification-specific loss function (e.g., cross-entropy).

c) Applications: DBNs adapted for classification are widely used in image classification, text categorization, and other tasks where data needs to be assigned to discrete classes.

B. REGRESSION

a) Adaptation: For regression tasks, you can modify the top layer of the DBN to have a single neuron with a linear activation function. This setup allows the network to predict continuous values.

b) Process: Similar to classification, after pre-training, you fine-tune the DBN using labelled data. In this case, you minimize a regression-specific loss function (e.g., mean squared error) during training.

c) Applications: DBNs adapted for regression are employed in tasks like predicting housing prices, stock market analysis, and any problem involving the prediction of continuous numerical values.

C. CLUSTERING

a) Adaptation: To use DBNs for clustering tasks, you can leverage the features learned by the DBN's hidden layers to represent data points. Then, you apply clustering algorithms, such as k-means or Gaussian mixture models, to these feature representations.

b) Process: After pre-training the DBN, you extract the activations of the hidden layers for each data point. We use these feature representations as input to a clustering algorithm to group similar data points into clusters.

c) Applications: DBNs adapted for clustering are useful in unsupervised learning scenarios, including customer segmentation, anomaly detection, and document clustering.

D. GENERATIVE MODELLING

a) Adaptation: DBNs can be used for generative modelling tasks by fine-tuning them as generative models. This typically involves training the network to generate new data samples that are similar to the training data.

b) Process: After pre-training, you fine-tune the DBN using a generative modelling[28,29] approach. One popular approach is to use the contrastive divergence algorithm for fine-tuning RBMs in the DBN.Once fine-tuned, the DBN can generate new samples by sampling from the learned probability distributions.

c) Applications: DBNs adapted for generative modelling are used in image generation, text generation, recommendation systems, and other tasks where generating new data samples is valuable.

Overall, DBNs can be adapted to a wide range of machine learning tasks by modifying their network architecture, loss functions, and fine-tuning procedures. Their hierarchical feature learning capabilities, combined with supervised or unsupervised fine-tuning, make them a flexible choice for various applications, from classification and regression to clustering and generative modelling.

CONCLUSION

DBNs, composed of multiple layers of stochastic hidden units, have found applications in diverse domains such as computer vision, natural language processing, and bioinformatics. The pre-training phase, the Contrastive Divergence (CD) and Persistent Contrastive Divergence (PCD) algorithms, their strengths and weaknesses in initializing deep networks were discussed along with their applicability to different data types and scales. Moving to the fine-tuning stage, we explored the use of backpropagation with gradient descent, discussing modern optimization techniques, including stochastic gradient descent and adaptive learning rate methods. We also examined regularization techniques like dropout and weight decay to address overfitting concerns. Furthermore, we discussed architectural variants of DBNs, such as Convolutional Deep Belief Networks (CDBNs) for image data and Recurrent DBNs for sequential data. We also have highlighted the adaptation of DBNs for specific tasks, including classification, regression, clustering, and generative modeling.

REFRENCES

[1] Yuming Hua, Junhai Guo and Hua Zhao, "Deep Belief Networks and deep learning," Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things, Harbin, 2015, pp. 1-4, doi: 10.1109/ICAIOT.2015.7111524.

[2] N. Senthilkumar, S. Karpakam, M. Gayathri Devi, R. Balakumaresan, P. Dhilipkumar, Speech emotion recognition based on Bi-directional LSTM architecture and deep belief networks,Materials Today: Proceedings, Volume 57, Part 5, 2022, Pages 2180-2184, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2021.12.246.

[3] Wang, Q., Gao, X., Li, X. et al. A precise method for RBMs training using phased curricula. Multimed Tools Appl 82, 8013–8047 (2023). https://doi.org/10.1007/s11042-022-12973-2

[4] Fabien Lareyre, Bahaa Nasr, Arindam Chaudhuri, Gilles Di Lorenzo, Mathieu Carlier, Juliette Raffort, Comprehensive Review of Natural Language Processing (NLP) in Vascular Surgery, EJVES Vascular Forum, 2023,, ISSN 2666-688X, https://doi.org/10.1016/j.ejvsvf.2023.09.002.

[5] Fred Nugen, Diana V. Vera Garcia, Sunghwan Sohn, John P. Mickley, Cody C. Wyles, Bradley J. Erickson, Michael J. Taunton, Application of Natural Language Processing in Total Joint Arthroplasty: Opportunities and Challenges, The Journal of Arthroplasty, Volume 38, Issue 10, 2023, Pages 1948-1953, ISSN 0883-5403, https://doi.org/10.1016/j.arth.2023.08.047.

[6] Xueqin Lü, Liyuan Long, Ruiyu Deng, Ruidong Meng, Image feature extraction based on fuzzy restricted Boltzmann machine, Measurement, Volume 204, 2022, 112063, ISSN 0263-2241, https://doi.org/10.1016/j.measurement.2022.112063.

[7] Nan Zhang, Shifei Ding, Jian Zhang, Yu Xue, An overview on Restricted Boltzmann Machines, Neurocomputing, Volume 275, 2018, Pages 1186-1199, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2017.09.065.

[8] Lin Ning, Randall Pittman, Xipeng Shen, LCD: A Fast Contrastive Divergence Based Algorithm for Restricted Boltzmann Machine, Neural Networks, Volume 108, 2018, Pages 399-410, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2018.08.018.

[9] Asja Fischer, Christian Igel, Training restricted Boltzmann machines: An introduction, Pattern Recognition, Volume 47, Issue 1, 2014, Pages 25-39, ISSN 0031-3203, https://doi.org/10.1016/j.patcog.2013.05.025.

[10] João Paulo Papa, Walter Scheirer, David Daniel Cox, Fine-tuning Deep Belief Networks using Harmony Search, Applied Soft Computing, Volume 46, 2016, Pages 875-885, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2015.08.043.

[11] Mei Yang, Ming K. Lim, Yingchi Qu, Xingzhi Li, Du Ni, Deep neural networks with L1 and L2 regularization for high dimensional corporate credit risk prediction, Expert Systems with Applications, Volume 213, Part A, 2023, 118873, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2022.118873.
[12] Sofia C. Pereira, Joana Rocha, Aurélio Campilho, Pedro Sousa, Ana Maria Mendonça, Lightweight multi-scale classification of chest radiographs via size-specific batch normalization, Computer Methods and Programs in Biomedicine, Volume 236, 2023, 107558, ISSN 0169-2607, https://doi.org/10.1016/j.cmpb.2023.107558.
[13] Guofa Li, Delin Ouyang, Liu Yang, Qingkun Li, Kai Tian, Baiheng Wu, Gang Guo, Cross-subject EEG linear domain adaption based on batch normalization and depthwise convolutional neural network, Knowledge-Based Systems, 2023, 111011, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2023.111011.
[14] Reza Kharghanian, Ali Peiravi, Farshad Moradi, Alexandros Iosifidis, Pain detection using batch normalized discriminant restricted Boltzmann machine layers, Journal of Visual Communication and Image Representation, Volume 76, 2021, 103062, ISSN 1047-3203, https://doi.org/10.1016/j.jvcir.2021.103062.
[15] Jinrui Wang, Shunming Li, Zenghui An, Xingxing Jiang, Weiwei Qian, Shanshan Ji, Batch-normalized deep neural networks for achieving fast intelligent fault diagnosis of machines, Neurocomputing, Volume 329, 2019, Pages 53-65, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2018.10.049.
[16] Ming Ding, The road from MLE to EM to VAE: A brief tutorial, AI Open, Volume 3, 2022, Pages 29-34, ISSN 2666-6510, https://doi.org/10.1016/j.aiopen.2021.10.001.
[17] Rachid Zeghlache, Mohamed Aymen Labiod, Abdelhamid Mellouk, Driver vigilance estimation with Bayesian LSTM Auto-encoder and XGBoost using EEG/EOG data, IFAC-PapersOnLine, Volume 55, Issue 8, 2022, Pages 89-94, ISSN 2405-8963, https://doi.org/10.1016/j.ifacol.2022.08.015.
[18] Alexandre Lemire Paquin, Brahim Chaib-draa, Philippe Giguère, Stability analysis of stochastic gradient descent for homogeneous neural networks and linear classifiers, Neural Networks, Volume 164, 2023, Pages 382-394, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2023.04.028.
[19] Wilfrido Gómez-Flores, Humberto Sossa, Learning smooth dendrite morphological neurons by stochastic gradient descent for pattern classification, Neural Networks, 2023, ISSN 0893-6080, https://doi.org/10.1016/j.neunet.2023.09.033.
[20] Lingxiao Zhao, Zhiyang Li, Leilei Qu, Junsheng Zhang, Bin Teng, A hybrid VMD-LSTM/GRU model to predict non-stationary and irregular waves on the east coast of China, Ocean Engineering, Volume 276, 2023, 114136, ISSN 0029-8018, https://doi.org/10.1016/j.oceaneng.2023.114136.
[21] Zhuoyue Guo, Canyun Yang, Dongsheng Wang, Hongbin Liu, A novel deep learning model integrating CNN and GRU to predict particulate matter concentrations, Process Safety and Environmental Protection, Volume 173, 2023, Pages 604-613, ISSN 0957-5820, https://doi.org/10.1016/j.psep.2023.03.052.
[22] Sergei Manzhos, Manabu Ihara, Rectangularization of Gaussian process regression for optimization of hyperparameters, Machine Learning with Applications, Volume 13, 2023, 100487, ISSN 2666-8270, https://doi.org/10.1016/j.mlwa.2023.100487.
[23] Inas Abuqaddom, Basel A. Mahafzah, Hossam Faris, Oriented stochastic loss descent algorithm to train very deep multi-layer neural networks without vanishing gradients, Knowledge-Based Systems, Volume 230, 2021, 107391, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2021.107391.
[24] Xin Wang, Yi Qin, Yi Wang, Sheng Xiang, Haizhou Chen, ReLTanh: An activation function with vanishing gradient resistance for SAE-based DNNs and its application to rotating machinery fault diagnosis, Neurocomputing, Volume 363, 2019, Pages 88-98, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2019.07.017.
[25] Dong Liu, Yong Wang, Chenhong Luo, Jun Ma, An improved autoencoder for recommendation to alleviate the vanishing gradient problem, Knowledge-Based Systems, Volume 263, 2023, 110254, ISSN 0950-7051, https://doi.org/10.1016/j.knosys.2023.110254.
[26] Xuetong Wu, Jonathan H. Manton, Uwe Aickelin, Jingge Zhu, A Bayesian approach to (online) transfer learning: Theory and algorithms, Artificial Intelligence, Volume 324, 2023, 103991, ISSN 0004-3702, https://doi.org/10.1016/j.artint.2023.103991.
[27] Juhyeon Kim, Joonsuk Huh, Daniel K. Park, Classical-to-quantum convolutional neural network transfer learning, Neurocomputing, Volume 555, 2023, 126643, ISSN 0925-2312, https://doi.org/10.1016/j.neucom.2023.126643.
[28] Uiwon Hwang, Sung-Woo Kim, Dahuin Jung, SeungWook Kim, Hyejoo Lee, Sang Won Seo, Joon-Kyung Seong, Sungroh Yoon, Real-world prediction of preclinical Alzheimer's disease with a deep generative model, Artificial

Intelligence in Medicine, Volume 144, 2023, 102654, ISSN 0933-3657, https://doi.org/10.1016/j.artmed.2023.102654.

[29] Yong Shi, Mengyu Shang, Zhiquan Qi, Intelligent layout generation based on deep generative models: A comprehensive survey, Information Fusion, Volume 100, 2023, 101940, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2023.101940.