Vol 7, Number 2, 2017
European Journal of Technic
EJT

AN OVERVIEW OF POPULAR DEEP LEARNING METHODS

Musab COŞKUN[1*], Özal YILDIRIM[2,] Ayşegül UÇAR[3], Yakup DEMIR[4]

[1,4]Department of Electrical and Electronics Engineering, Firat University, Elazığ, Turkey
[2]Department of Computer Engineering, Munzur University, Tunceli, Turkey
[3]Department of Mechatronics Engineering, Firat University, Elazığ, Turkey
[*]musabcoskun@firat.edu.tr

This paper offers an overview of essential concepts in deep learning, one of the state of the art approaches in machine learning, in terms of its history and current applications as a brief introduction to the subject. Deep learning has shown great successes in many domains such as handwriting recognition, image recognition, object detection etc. We revisited the concepts and mechanisms of typical deep learning algorithms such as Convolutional Neural Networks, Recurrent Neural Networks, Restricted Boltzmann Machine, and Autoencoders. We provided an intuition to deep learning that does not rely heavily on its deep math or theoretical constructs.

Key words: *Deep Learning, Convolutional Neural Networks, Recurrent Neural Networks, Autoencoders, Restricted Boltzmann Machines,*

1. Introduction

Machine learning technology supports the modern society in many ways. This technology becomes widespread in many products such as cameras and smart phones and is also used in many applications such as content filtering in social network search. Moreover, it is especially beneficial for object recognition [1,2], speech recognition [3], edge detection [4], and many other areas as addressed in references [5,6,7].

Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Deep learning has enabled many practical applications of machine learning and by extension the overall field of Artificial Intelligence. Compared to shallow learning deep learning has the advantage of building deep architectures to learn more abstract information. The most important property of deep learning methods is that it can automatically learn feature representations thus avoiding a lot of time-consuming engineering. Better chip processing abilities, considerable advances in the machine learning algorithms, and affordable cost of computing hardware are primarily crucial reasons for the booming of deep learning [8].

Traditional machine learning relies on shallow networks which are composed of one input and one output layer, and no more than one hidden layer between input and output layers. Deep learning is qualified when more than three layers exist in a network including input and output layers. Therefore, the more the number of hidden layers is increased, the more the network gets deeper as shown in Fig.1.

The remainder of this paper discusses typical deep learning algorithms which are Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Restricted Boltzmann Machine (RBM), and Autoencoders respectively. We offer our paper in a way that each section can be read independently.

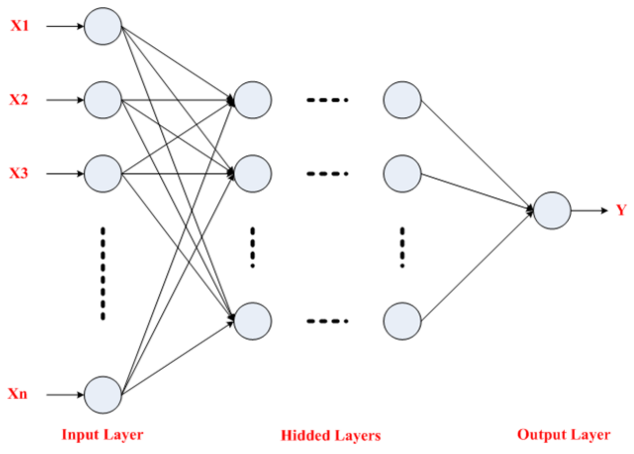

Fig. 1. An example of deep neural network

2. Deep learning methods

2.1. Convolutional Neural Network

CNN was firstly introduced by Kunihiko Fukushima [9]. It was later proposed by Yann LeCun. He combined CNN with back-propagation theory to recognize handwritten digits and document recognition [10,11]. His system was eventually used to read hand-written checks and zip codes. CNN uses convolutional layers and pooling layers. Convolutional layers filter inputs for useful information. They have parameters that are learned so that filters are adjusted automatically to extract the most useful information for a certain task. Multiple convolutional layers are used that filter images for more and more abstract information after each layer. Pooling layers are used for limited translation and rotation invariance. Pooling also reduces the memory consumption and thus allows for the usage of more convolutional layers.

2.1.1. Convolution Operation

Convolution is just a mathematical operation that describes a rule of how to mix two functions and produces a third function. This third function is an integral that expresses the amount of overlap of one function as it is shifted over the other function. In other words, an input data and a convolution kernel are subjected to particular mathematical operation to generate a transformed feature map. Convolution is often interpreted as a filter, where the kernel filters the feature map for information of a certain kind. Convolution is described formally as follows:

$$h(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{1}$$

CNN typically works with two-dimensional convolution operation as summarized in Fig. 2. The leftmost matrix is input data. The matrix in the middle is convolution kernel and the rightmost matrix is a feature map. The feature map is calculated by sliding convolution kernel over the entire input matrix. The convolution process is an element-wise operation followed by a sum. For example, when the right upper 3×3 matrix is convoluted with convolution kernel, the result is 77.

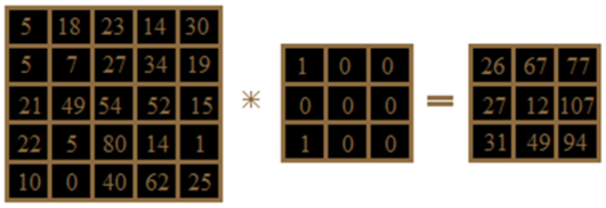

Fig. 2. A simple depiction of 2-dimensional convolutional operation

The convolution operation is usually known as kernels. By different choices of kernels, different operations of the images can be obtained. Operations are typically including edge detection, blurring, sharpening etc. By introducing random matrices as convolution operator, some interesting properties might be discovered. As a result of convolution in neural networks, the image is split into perceptrons, creating local receptive fields and finally compressing the perceptrons in feature maps. All in all, learning a meaningful convolutional kernel is one of the central tasks in CNN when applied to computer vision tasks.

2.1.2. Convolution Layers

A typical CNN architecture consists of convolutional and pooling (or subsampling or downsampling) layers as depicted in Fig. 3. A convolutional layer is primarily a layer that performs convolution operation. Its main task is to map. The result of staging convolutional layers in conjunction with the following layers is that the information of the image is classified like in vision. That means that the pixels are assembled into edges, edges into motifs, motifs into parts, parts into objects, and objects into scenes. Convolutional layer introduces the Rectified Linear Unit (ReLU) the non-linearity transform after convolution to assist the simulation to be more successful. There are other non-linear functions such as Hyperbolic Tangent or Sigmoid that can also be used instead of ReLU, however ReLU has been found to perform better in most situations. ReLU is a special implementation that combines non-linearity and rectification layers in CNNs. It is a piecewise linear function defined as follows:

$$f(x) = \max(0, x) \tag{2}$$

which is a transform that replaces all negative pixel values in the feature map by zero
and therefore solves the cancellation problem as well as results in a much more sparse activation volume at its output. The sparsity is useful for multiple reasons but mainly provides robustness to small changes in input such as noise [12].

The pooling layer is responsible for reducing the spacial size of the activation maps. Although it reduces the dimensionality of each feature map, it retains the most important information. There are different strategies of the pooling which are max-pooling, average-pooling and probabilistic pooling. Max-pooling takes the maximum of the input data. Average-pooling takes the averged value of the input data. Probabilistic pooling takes a random value of the input data [13]. Pooling makes the input representations or feature dimension smaller and more manageable. It helps the network to be invariant to small transformations, distortions,and translations in the input image. It also reduces the number of parameters and computations in the network as well as minimizes the likelihood of overfitting.

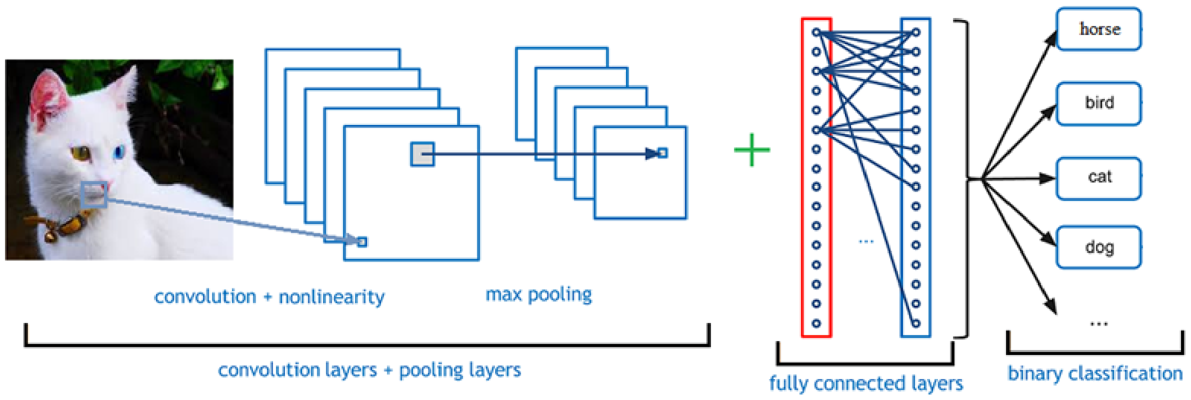

Fig. 3. A typical CNN architecture

After several convolutional and max pooling layers, the high-level reasoning in the neural network is done via Fully Connected Layers (FCLs). A FCL takes all neurons in the previous layer and connects it to every single neuron it has. FCLs are not spatially located anymore, that means they can be visualized as one-dimensional. Therefore there can be no convolutional layers after an FCL.

The output from the convolutional layers represents high-level features in the data. While that output could be flattened and connected to the output layer, adding a fully-connected layer is a cheap way of learning non-linear combinations of these features. The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the Softmax as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

2.1.3. CNN Architectures

CNNs have recently enjoyed a great success in large-scale image and video recognition. The influential architectures of CNNs can be listed as below and are presented in chronological order with better accuracy from the earlier ones from LeNet to DenseNet.

LeNet is a pioneering work was named LeNet-5 by Yann LeCun after previous successful iteration [14,15]. At that time the LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc. With the introduction of LeNet, LeCun et al. [15] also introduced the MNIST database, which is known as the standard benchmark in digit recognition field.

AlexNet made CNNs popular in Computer Vision. It is composed of 5 convolutional layers followed by 3 fully connected layers. It was developed by Alex Krizhevsky et al. and won ImageNet ILSVRC challenge in 2012 [16]. During this competition it produced the best results, top-1 and top-5 error rates of 37.5% and 17.0%.

ZFNet won the ILSVRC 2013. It was proposed by Matthew Zeiler and Rob Fergus [17]. It became known as the ZFNet. It was an improvement on AlexNet by tweaking the architecture hyper-parameters, in particular by expanding the size of the middle convolutional layers and making the stride and filter size on the first layer smaller.

VGGNet was the runner-up in ILSVRC 2014 from VGG group, Oxford [18]. It makes the improvement over AlexNet and has 19 layers in total. Its main contribution was in showing that the depth of the network or the number of layers is a critical component for good performance. Although VGGNet achieves a phenomenal accuracy on ImageNet dataset, its deployment on even the most modest sized Graphics Pprocessing Units (GPUs) is a problem because of huge computational requirements, both in terms of memory and time. It becomes inefficient due to large width of convolutional layers.

GoogLeNet was invented by Szegedy et al. from Google that was the winner of ILSVRC 2014 [19]. Its main contribution was the development of an inception module that dramatically reduced the number of parameters in the network. Inception module approximates a sparse CNN with a normal dense construction. Since only a small number of neurons are effective as mentioned earlier, width/number of the convolutional filters of a particular kernel size is kept small. Additionally, it uses convolutions of different sizes to capture details at varied scales. Another salient point about the module is that it has a so-called bottleneck layer. It helps in massive reduction of the computation requirement. Another change that GoogLeNet made, was to replace the FCLs at the end with a simple global average pooling which averages out the channel values across the 2D feature map, after the last convolutional layer. This drastically reduces the total number of parameters. This can be understood from AlexNet, where FCLs contain approximately 90% of parameters. Use of a large network width and depth allows GoogLeNet to remove the FCLs without affecting the accuracy. It achieves 93.3% top-5 accuracy on ImageNet and is much faster than VGG.

ResNet (Residual Network) developed by Kaiming He et al. was the winner of ILSVRC 2015 [20]. ResNet is a 152 layer network, which was ten times deeper than what was usually seen during the time when it was invented It features special skip connections and a heavy use of batch normalization. It uses a global average pooling followed by the classification layer. It achieves better accuracy than VGGNet and GoogLeNet while being computationally more efficient than VGGNet. ResNet-152 achieves 95.51% top-5 accuracies.

DenseNet was published by Gao Huang et al and won best paper award in CVPR 2017 [21]. It has each layer directly connected to every other layer in a feed-forward fashion. The DenseNet has been shown to obtain significant improvements over previous state-of-the-art architectures on four highly competitive object recognition benchmark tasks(CIFAR-10, CIFAR-100, SVHN, and ImageNet).

2.2. Recurrent Neural Network

RNNs are a family of neural networks for processing sequential data. RNNs are popular models that have shown great promises in a variety of problems such as speech recognition, language modeling, translation, image captioning [22-26]. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. In other words they have a memory which captures information about what has been calculated till the moment. In theory RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps.

A simple example of RNN was firstly proposed by Elman[27]. Its diagram is shown in Fig. 4. If RNN in Fig. 4 is unfolded, it turns out to be like in Fig. 5. A chunk of neural network looks at some inputs and outputs a value. A loop allows information to be passed from one step of the network to the next. A RNN can be thought of as multiple copies of the same network, each passing a message to a successor. This chain-like nature reveals that RNNs are intimately related to sequences and lists.

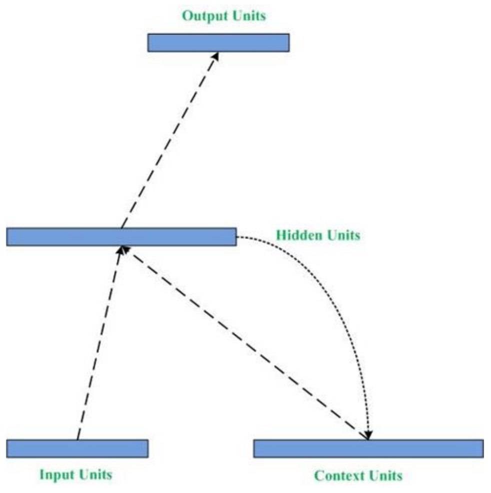

Fig. 4. A simple example of RNN

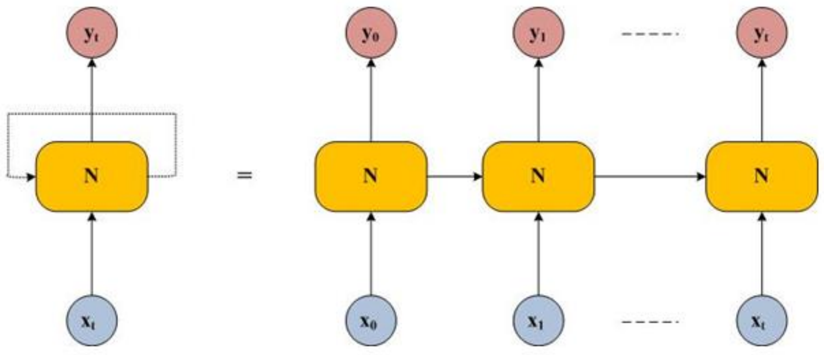

Fig. 5. Unfolding of a RNN in time of the computation involved in its forward computation

The big deal about RNN is its memory capability for modeling sequential patterns. It was plagued with gradients that die after a few steps till Long Short Term Memory (LSTM), the most commonly used type of RNNs, was invented [28] . It was much better at capturing long-term dependencies than vanilla RNNs. LSTMs have a different way of computing the hidden state.

An LSTM is an architecture that solves the vanishing gradient problem of plain vanilla RNN, so unless there are other considerations, there is no reason not to choose LSTM. The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell [29].

2.3. Restricted Boltzmann Machine

Boltzmann machines have been proposed in 1985 [30]. Compared to the times when they were first introduced, RBMs can be applied to more interesting problems due to the increase in computational power and the development of new learning algorithms in many domains such as image classification, texture synthesis, medical image processing, and denoising [31-36].

An RBM is structually a shallow neural net with just two layers that are the visible layer (input layer) and the hidden layer [37] as shown in Fig. 6. It is a method that can automatically find patterns in data by reconstructing the input. An RBM is considered restricted because of the fact that neurons in each layer have no connections between them and are connected to all other neurons in other layer. In RBM networks, connections between neurons are bidirectional and symmetric. This means that information flows in both directions during the training and during the usage of the network and those weights are the same in both directions. During forward pass, an RBM takes the inputs and translates them into a set of numbers that encode the inputs. In the backward pass, it takes this set of numbers and translates them back to form the reconstructed inputs. A well-trained RBM network is able to perform the backward translation with a high degree of accuracy. In both steps, the weights and biases have a crucial role. They allow the RBM to decipher the interrelationships among the input features and they also help RBM decide which input features are the most important when detecting patterns.

Through several forward and backward passes, an RBM is trained to reconstruct the input data. There are three steps repeated over and over through the training process as below:

- With a forward pass every input is combined with an individual weight and one overall bias, and the result is passed to the hidden layer which may or may not activate.
- Each activation function is combined with an individual weight and an overall bias, and the result is passed to the visible layer for reconstruction in a backward pass.
- In the last step, the construction is compared against the original input to determine the quality of the result.

An interesting aspect of an RBM is that the data does not need to be labeled. This turns out to be very important for the real-world data sets like photos, videos, and sensor signals. These are all tending to be unlabeled. By reconstructing the input, the RBM must also decipher the building blocks and patterns that are inherent in the data.

RBMs have received a lot of attention recently after being proposed as building blocks of multi-layer learning architectures called Deep Belief Networks (DBNs) [31, 39].

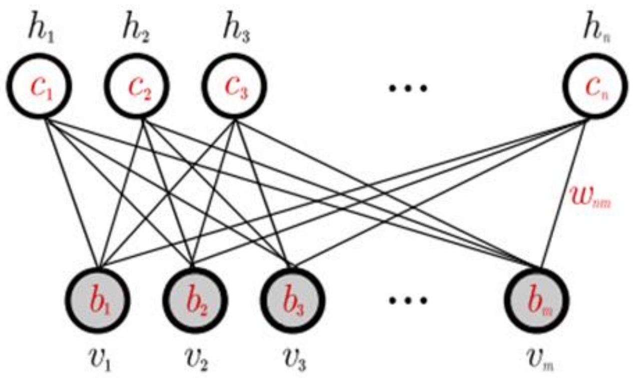

Fig. 6. The network graph of an RBM with n hidden and m visible units [38]

DBNs are multi-layer belief networks. Each layer in DBN is an RBM and they are stacked each other to construct DBN. DBNs were conceived by Hinton as an alternative to backpropagation. It showed that it is possible to learn a deep, densely connected, belief network one layer at a time. Their architecture demonstrated successful results on the MNIST dataset [40].

The difference of a DBN from a multilayer perceptron comes from the way it is being trained. Training method of the DBN is the key factor that it can outperform its shallow counterparts. A DBN can be viewed as a stack of RBMs, where the hidden layer of one RBM is the visible layer of the one above it. It can be illustrated as depicted in Fig. 7. A DBN is trained as follows:

- RBM1 is trained to reconstruct its input as accurately as possible.
- The hidden layer of RBM1 is treated as the visible layer for RBM2 and RBM2 is trained using the outputs from RBM1.
- This process is repeated until output layer in the network is trained.

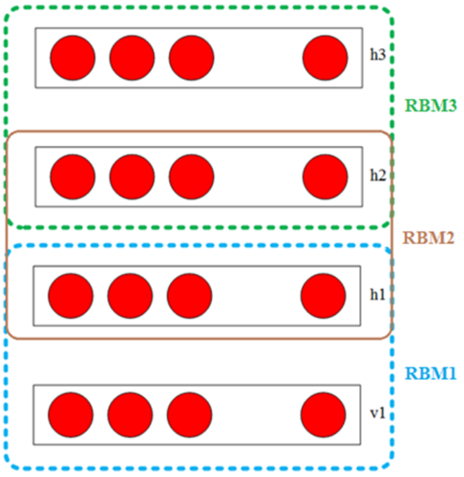

Fig. 7. An architecture of DBN

An important point about a DBN is that each RBM layer learns the entire input. It works globally by fine tuning the entire input in succession as the model slowly improves. After initial training, the RBMs create a model that can detect inherent patterns in the data. However what those patterns are called is still unknown. To finish the training, it is required to introduce labels to the patterns and fine-tune the network with supervised learning. In order to do this, a small set of labeled samples is needed so that the features and patterns can be associated with a name. The weights and biases are changed slightly, resulting in a small change in the network's perception of the patterns, and often a small increase in the total accuracy.

All in all, an RBM can extract features and reconstruct features. However, the vanishing gradient problem is still waiting to be solved. A DBN only needs a small labeled data set, which is important for real-world applications. The training process can also be completed in a reasonable amount of time through the use of Graphical Processing Units (GPUs). Furthermore, the resulting network will be very accurate compared to a shallow network. Therefore a DBN can be regarded as a solution to the vanishing gradient problem.

2.4. Autoencoders

Autoencoders (also called Autoassociators) are a family of neural networks for which the input layer is the same as the output layer, as well as an unsupervised learning algorithm[41,42]. They work by compressing the input into a latent-space representation, and then reconstructing the output from this representation as illustrated in Fig. 8. In more terms, autoencoding is a data compression algorithm where the compression and decompression functions are data-specific, lossy and learn automatically from examples. They have been used as building blocks to build a deep multi-layer neural network [43] as well as reducing the dimensionality of the data [31]. An autoencoder takes a set of typically unlabeled inputs, and after encoding them, tries to reconstruct them as accurately as possible. As a result of this, the network must decide which of the data features are the most important, essentially acting as a feature extraction engine. Autoencoders are typically very shallow, and are usually comprised of an input layer, an output layer and a hidden layer. Some of autoencoder networks have only two layers instead of three like the RBM. It can also be thought of as a 2-way translator like the RBM. Input signals are encoded along the path to the hidden layer, and these same signals are decoded along the path to the output layer.

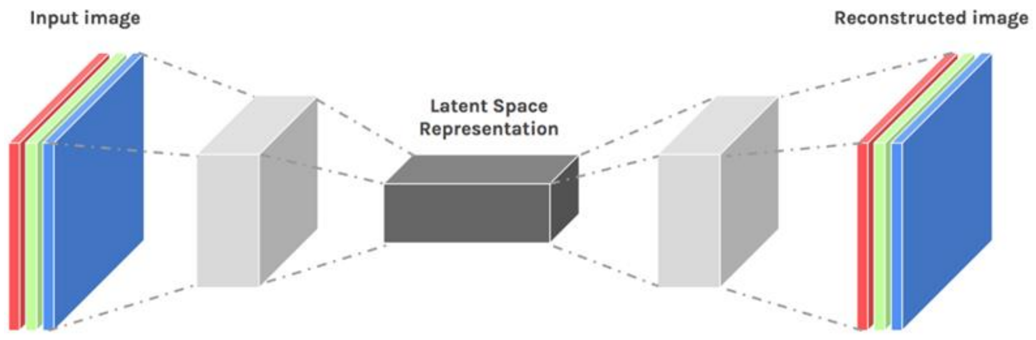

Fig. 8. Autoencoder architecture [44]

Deep autoencoders are extremely useful tools for dimensionality reduction [31]. For example, these networks can transform an image containing 28x28 grid of pixels into a representation with only 30 numbers. The image can then be reconstructed with the appropriate weights and bias. Additionally, some networks also add random noise at this stage in order to enhance the robustness of the discovered patterns. The reconstructed image would not be perfect. However the result would be a decent approximation depending on the strength of the network. The purpose of this compression is to reduce the input size on a set of data before feeding it to a deep classifier. Smaller inputs lead to large computational speedups, so this preprocessing step is worth the effort.

Data denoising and dimensionality reduction for data visualization are considered as two main interesting practical applications of autoencoders. With appropriate dimensionality and sparsity constraints, autoencoders can learn data projections that are more interesting than Principal Component Analysis (PCA) or other basic techniques.

3. Conclusions

In this paper, we particularly consider deep models such as CNNs, RNNs, RBMs, and Autoencoders. Due to the prominence and more problem spaces of CNNs in recent years, we mainly focused on their structure and gave more details about their structures and architectures.

Since deep learning inception, the last decade has been the blooming of Artificial Intelligence. Deep learning takes hand-crafted techniques out of the scene when there is enough data and good network architectures in order to learn abstract features. With recent improvements in GPU technology a lot of matrix computations can be done very efficiently in parallel and this helps training a deep network not consuming time as it used to be a decade ago. This is also one of the reasons why deep learning is growing to prominence. While still nascent, it is deep learning getting closer to the ultimate goal of Artificial Intelligence which is closed to a human intelligence level that helps solving harder and more significant problems that truly affect humanity.

4. References

[1] M. Liang, and X. Hu, Recurrent convolutional neural network for object recognition, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3367–3375.
[2] P. Pinheiro, and R. Collobert, Recurrent convolutional neural networks for scene labeling, Proceedings of the 31st International Conference on Machine Learning, 2014, vol. 32, pp. 82–90.
[3] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, Natural Language Processing (Almost) from Scratch, J. Mach. Learn. Res., 2011, vol. 12, pp. 2493–2537.
[4] Mohamed A. El-Sayed, Yarub A. Estaitia, and Mohamed A. Khafagy, Automated Edge Detection Using Convolutional Neural Network, Int. J. Adv. Comput. Sci. Appl., 2013, vol. 4, no. 10, pp. 11–17.
[5] Dan Cireşan, Deep Neural Networks for Pattern Recognition.
[6] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang, DeepContour: A deep convolutional feature learned by positive-sharing loss for contour detection, in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2015, vol. 07–12–June, pp. 3982–3991.
[7] E. Shelhamer, J. Long, and T. Darrell, Fully Convolutional Networks for Semantic Segmentation, IEEE Trans. Pattern Anal. Mach. Intell., 2017, vol. 39, no. 4, pp. 640–651.
[8] Y. LeCun, Y. Bengio, G. Hinton, L. Y., B. Y., and H. G., Deep learning, 2015, Nature, vol. 521, no. 7553, pp. 436–444.
[9] Kunihiko Fukushima, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, 1980, Biol. Cybernetics 36, pp. 193-202.
[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Handwritten digit recognition with a back-propagation network, in NIPS'89.
[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, 1998, Proceedings of the IEEE.
[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio, Deep Sparse Rectifier Neural Networks, 2011, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics, vol. 15 of JMLR. Pp. 315-323.
[13] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, 2009, In Proceedings of the 26th annual international conference on machine learning, pp. 609–616.
[14] Yan Lecun B Boser, John S Denker, D Henderson, Richard E Howard, W Hubbard, and Lawrence D Jackel, Handwritten digit recognition with a back-propagation network, 1990, In Advances in neural information processing systems. Citeseer.
[15] Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner, Gradient-based learning applied to document recognition, 1998a, Proceedings of the IEEE, pp. 2278–2324.
[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, Imagenet classification with deep convolutional neural networks, In Advances in neural information processing systems, 2012, pages 1097–1105.
[17] Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. CoRR, abs/1311.2901, 2013, Published in Proc. ECCV, 2014.
[18] Karen Simonyan, and Andrew Zisserman, Very Deep Convolutional Networks For Large-Scale Image Recognition, 2014, arXiv preprint arXiv:1409.1556.

[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, Going Deeper with Convolutions, 2014, Computer Vision and Pattern Recognition (CVPR 2015).
[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep Residual Learning for Image Recognition, 2015, arXiv:1512.03385v1.
[21] Gao Huang, Zhuang Liu, Kilian Q. Weinberger, and Laurens van der Maaten, Densely Connected Convolutional Networks, 2016, arXiv:1608.06993v4.
[22] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, Speech recognition with deep recurrent neural networks, 2013, IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).
[23] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition, 2009, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 31(5):855–868.
[24] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Honza Cernock, Sanjeev Khudanpur, Recurrent neural network based language model, 2010, INTERSPEECH 2010.
[25] Biao Zhang, Deyi Xiong, and Jinsong Su, Recurrent Neural Machine Translation, 2016, EMNLP 2016, arXiv:1607.08725v1.
[26] Justin Johnson, Andrej Karpathy, and Li Fei-Fei, DenseCap: Fully Convolutional Localization Networks for Dense Captioning, 2015, arXiv:1511.07571v1.
[27] Jeffrey L. Elma, Finding Structure in Time, 1990, Cognitive Science 14, pp. 179-211.
[28] Sepp Hochreiter, and Jürgen Schmidhuber, Long Short-Term Memory, 1997, Neural Computation 9(8):1735-1780.
[29] Klaus Greff, Rupesh K. Srivastava, Jan Koutnık, Bas R. Steunebrink, and Jurgen Schmidhuber, LSTM: A Search Space Odyssey, 2016, IEEE Transactions On Neural Networks And Learning Systems, vol.28, Issue: 10, pp. 2222-2232.
[30] David H Ackley, Geoffrey E Hinton, and Terrence J Sejnowski, 1985, A learning algorithm for boltzmann machines. Cognitive science, 9(1):147–169.
[31] G. E. Hinton and R. R. Salakhutdinov, 2006, Reducing the dimensionality of data with neural networks, Science, 313(5786):504–507.
[32] J. Kivinen, and C. Williams, Multiple texture Boltzmann machines, 2012, JMLR W&CP: AISTATS 2012, 22:638–646.
[33] H. Larochelle and Y. Bengio, Classification using discriminative restricted Boltzmann machines, 2008, International Conference on Machine learning(ICML), pp. 536-543.
[34] A. Mohamed, and G. E. Hinton, Phone recognition using restricted Boltzmann machines, 2010, IEEE International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 4354–4357.
[35] T. Schmah, G. E. Hinton, R. S. Zemel, S. L. Small, and S. C. Strother, Generative versus discriminative training of RBMs for classification of fMRI images, 2009, Advances in Neural Information Processing Systems (NIPS 21), pp. 1409–1416.
[36] Y. Tang, R. Salakhutdinov, and G. E. Hinton, Robust Boltzmann machines for recognition and denoising, 2012, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2264–2271.
[37] Hinton, G.E, Training products of experts by minimizing contrastive divergence, 2002, Neural Computation 14, pp. 1771–1800.

[38] Asja Fischer, and Christian Igel, Training Restricted Boltzmann Machines: An Introduction, 2014, Pattern Recognition 47:25-39.

[39] [Hinton, G.E, Learning multiple layers of representation, 2007, Trends in Cognitive Sciences 11(10), pp. 428–434.

[40] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, A Fast Learning Algorithm for Deep Belief Nets, 2006, Neural Computation 18, pp. 1527-1554.

[41] Yoshua Bengio, Learning Deep Architectures for AI, 2009, Dept. IRO, Universite de Montreal, Technical Report 1312.

[42] P. Vincent, H. Larochelle Y. Bengio and P.A. Manzagol, Extracting and Composing Robust Features with Denoising Autoencoders, 2008, Proceedings of the 25th International Conference on Machine Learning (ICML2008), pp. 1096-1103.

[43] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle, Greedy Layer-Wise Training of Deep Networks, 2007, Advances in neural information processing systems, pp. 153-160.

[44] Autoencoders-Bits and Bytes of Deep Learning, [Online], *https://medium.com/towards-data-science/autoencoders-bits-and-bytes-of-deep-learning-eaba376f23ad*, [Accessed: 09-Oct-2017].