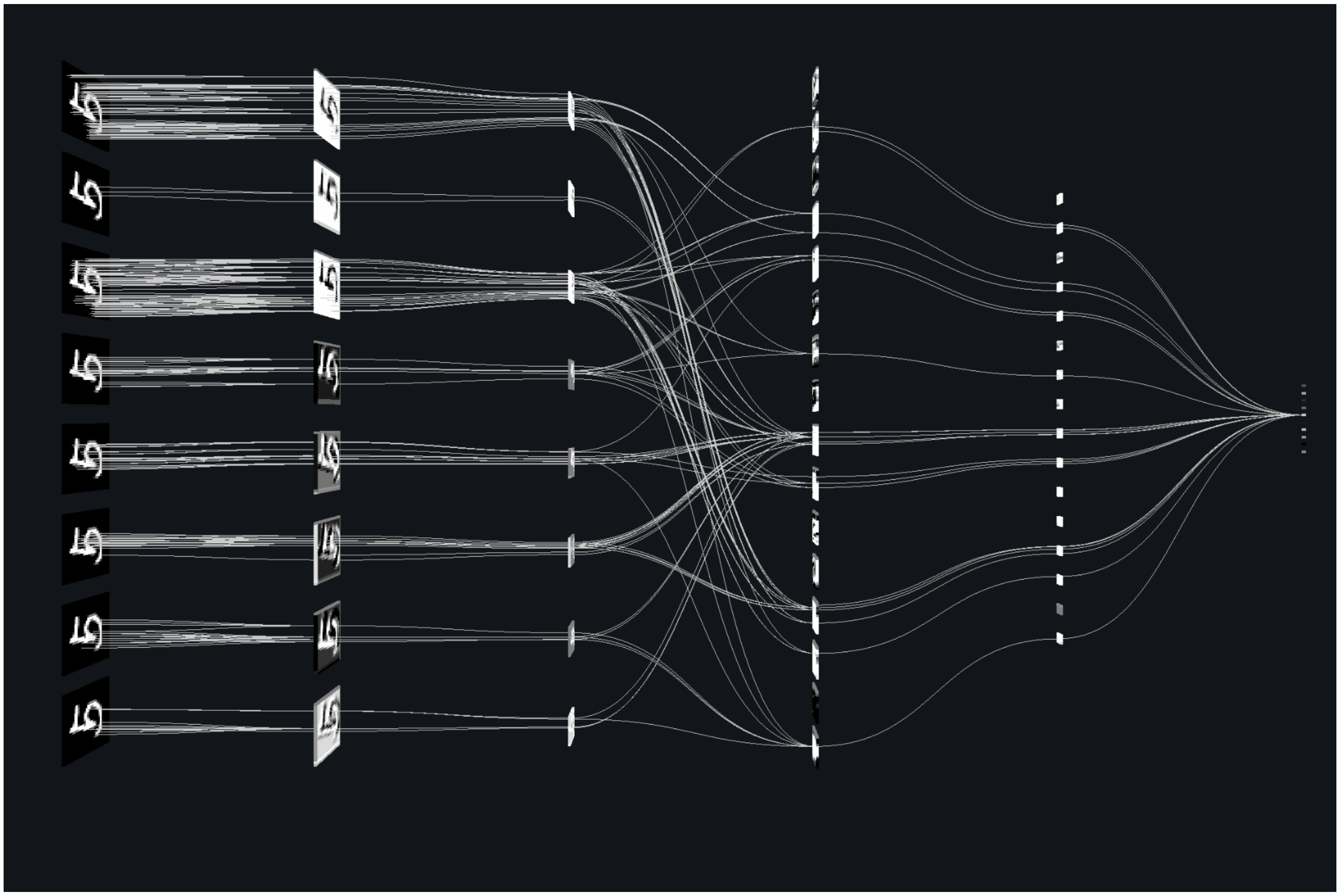


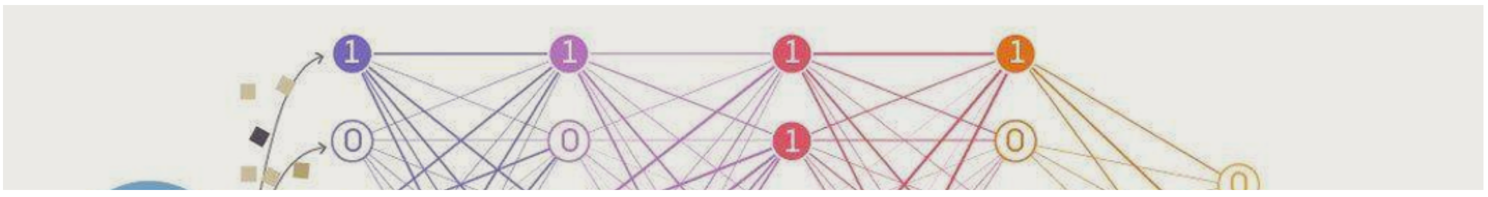
Advancements of convolutional neural networks part 1: Introduction and Main operations

Dashanka Nadeeshan Follow
May 24, 2018 · 8 min read

Today, machine learning technologies in artificial intelligence has become a key aspect of the modern technology. Deep leaning is an already established major machine learning technology in AI that have shown promising results in decision making and problem-solving in variety of applications. This article particularly interested in one of the main deep learning architecture, convolutional neural networks (CNNs) and their recent advancements.

CNNs are a type of artificial neural networks inspired by animal visual perception system mostly used to solve computer vision problems. They have performed well in vision-based recognition tasks, understanding and inference. Yann LeCun, a pioneer mathematician introduced the basic structure of modern CNN and Alex Krizhevsky proposed the first successful CNN architecture, AlexNet in 2012. The basic CNN architecture comprises multiple processing layers that are capable of learning feature representations starting directly from raw inputs. These learned representations at each layer are used to develop multiple levels of abstractions that allow CNNs to successfully applied in a variety of tasks. However, due to the lack of large training data and computing power at that time, deep CNNs couldn't perform well on more complex problems. But with the development of high-performance computing hardware such as GPUs and availability of a large amount of data thanks to the internet, now this is a rapidly developing filed.

The impact of the AlexNet has made CNNs a very active field of study and many researchers have become interested in developing CNNs. As a result, a lot of research work has been done over the past decade in terms of improving performance, new architectures, optimization techniques, data processing, and various applications. This study is to briefly review the recent advancements and developments of CNNs. There have been many variations of CNN architectures purposed over the time but the basic framework of the architecture remains similar. The concept of deep CNN comes with multiple hidden layers and have more processing capabilities. Each fed input processed through a series of hidden layers. The general multilayer architecture is a sequence of three types of alternating data processing layers: convolutional layers, pooling layers, and fully connected layers. Typical order of a CNN is alternately layered convolution and pooling layers followed by fully connected layers and an output layer which is also known as softmax layer at the end of the network which is the final fully connected layer. Following sections briefly discusses about main operational components of CNNs.



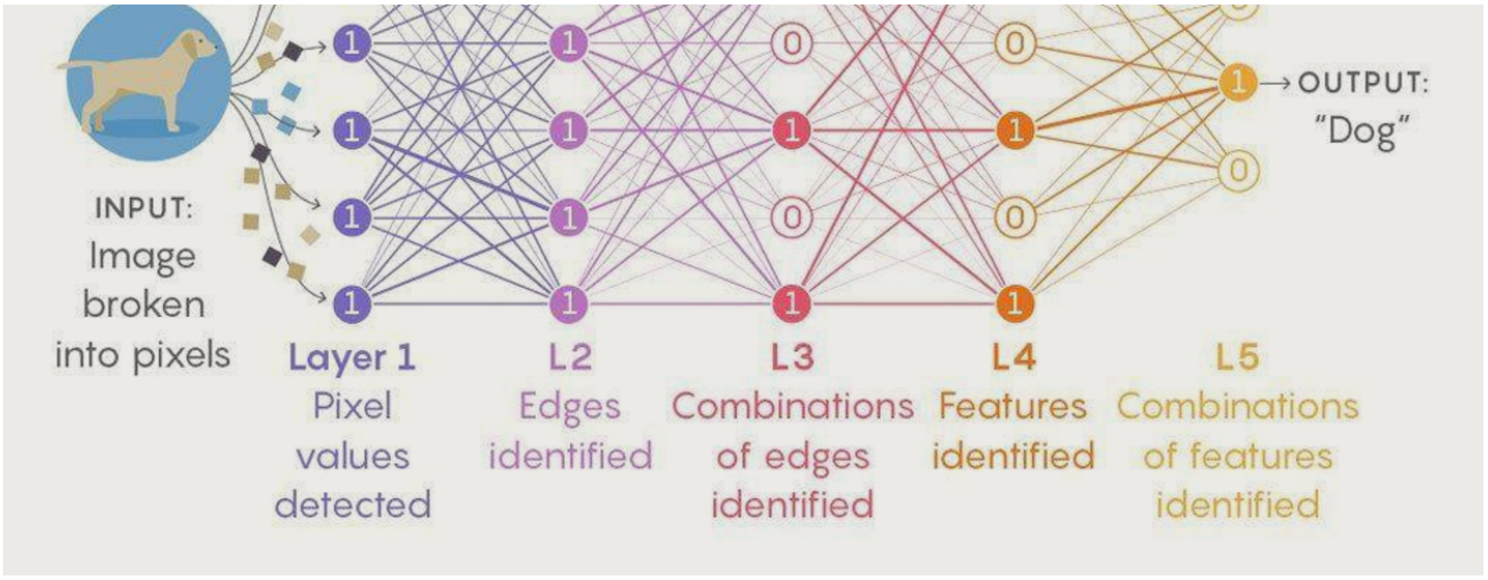
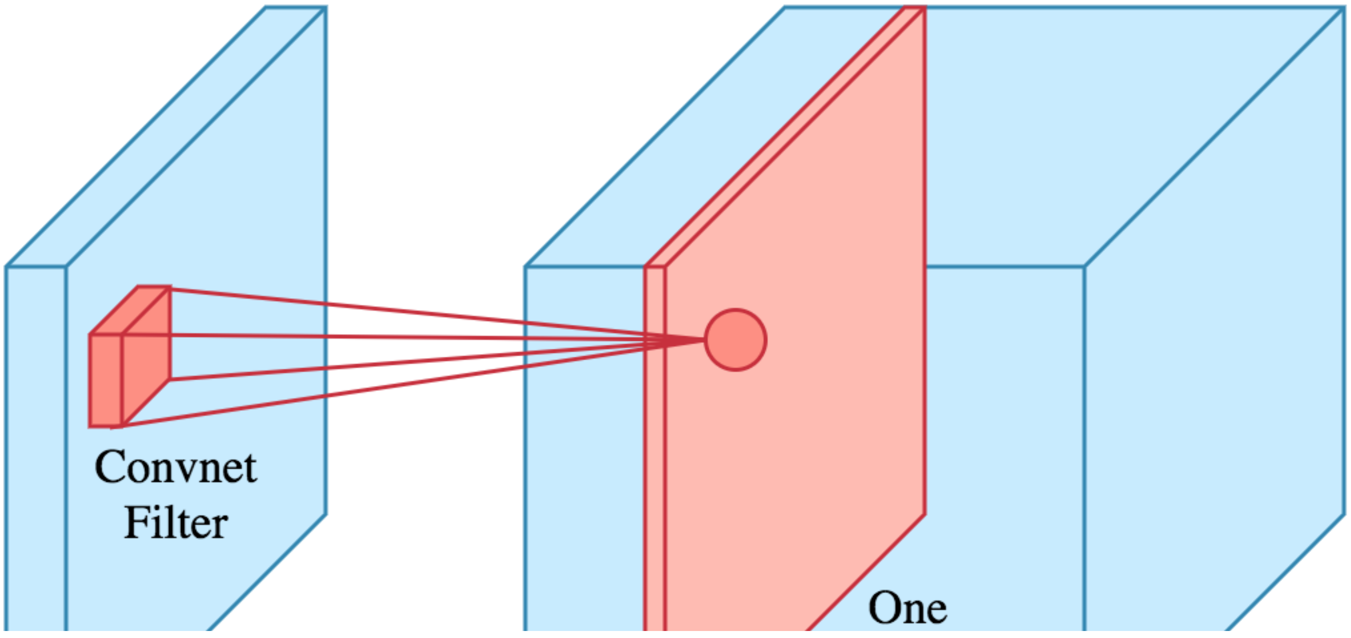

INPUT: Image broken into pixels
Layer 1 Pixel values detected
L2 Edges identified
L3 Combinations of edges identified
L4 Features identified
L5 Combinations of features identified
OUTPUT: "Dog"

Figure 2: How convolutional neural network works

Convolution layer

The convolutional layer is responsible for feature learning from inputs and develops abstractions at each level. The basic feature learning is done by convolving the filter kernels with inputs. A localized region of an input map is captured per convolution which is known as local receptive fields and hence it captures the entire input area. These filter kernels have learnable parameters which are known as weights that are systematically tuned using process called backpropagation during the training phase. The process generates output feature maps depending on number of filters for each convolution layer by transforming the input features from a one level to next level of abstraction and then these feature maps become the inputs to the following layer.



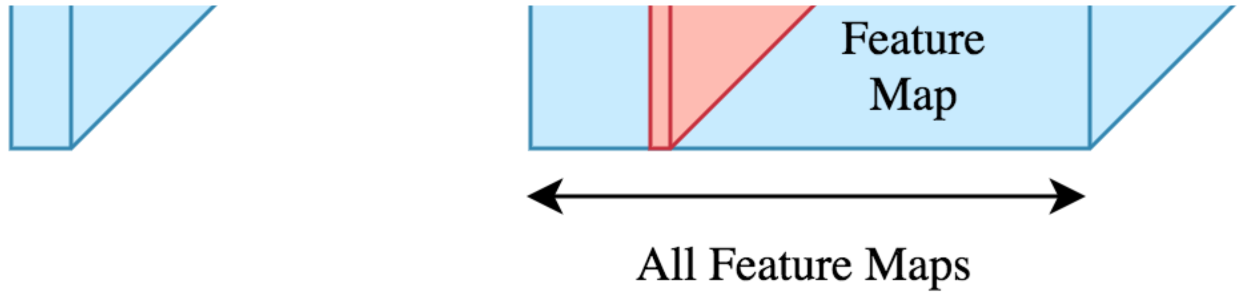


Figure 2: How convolution layer works

The concept of weight shearing is referred as sharing the same weights over a convolution layer. This also reduce the number of parameters to be trained. But it restricts the learning of different invariances, such as scale and rotational invariant features. Tiled convolutions[1] are a purposed solution to address this problem by inserting the many separate filters within the same convolution layer to learn diverse features maps. This method involves parameter k, that refers to the tile size and by varying k the degree of weight sharing changes. If the k is 1, the filter units in each particular map share same weights which gives the traditional CNN. Tiled CNNs provides the advantage of representing complex invariances and a less number of parameters to be trained. Convolution with dilated filter kernels is referred as dilated convolution. Dilated convolutions[2] are used to increase area of the receptive field by simply applying the same convolution filters to inputs at different ranges using a hyperparameter called dilation factor. This increased local view helps to capture more information and exponentially expands the global view of the network. The idea of network in network[3] is to replace the convolution filters with a small-scale neural network known as micro networks. A simple multi-layer perceptron is used here as the micro network. The convolution layer and its micro network layer extract features from the corresponding local receptive field. This enables the main network to abstract richer feature representations.

Activation function

Each and every convolution layer in a CNN is followed by an activation function which introduces a non-linearity to outputs. Without nonlinear activation, the network will be merely a bunch of linear transformations from inputs to outputs. There are 3 main types of activation functions are in practice: sigmoid, hyperbolic tangent and rectified linear units(ReLU). However, recently ReLU[4] has become more popular because they are bounded by a minimum value and it causes the networks to suffer less from diminished gradient flow. The basic ReLU is a piecewise linear function which prunes the negative part to zero and retains the positive part of its input. It also computes faster than sigmoid and tangent activations while providing a better sparsity to the network. Most importantly ReLU activation function suffer less from the problem of vanishing gradients and become back-propagation friendly. There are several variations of ReLU have been introduced. In some cases, ReLU can suffer from the disadvantage of having zero gradients when the unit is inactive. At initial levels, this can cause gradient-based optimization to not update their weights since the units are not active and could remain inactive. In addition, constant zero gradients could slow down the entire training process. Leaky ReLU[5] has been introduced to prevent this by encompassing the negative part of inactive units, so they will not become zero but keeping an allowance for the small non-zero gradients. Parametric ReLU[6] is then purposed an extension to leaky ReLU by introducing learnable parameter instead of predefined parameter for the negative encompassing part. Another variant of leaky ReLU is introduced as randomized ReLU[7] which randomly samples the parameters of the negative part from a uniform distribution in training but uses fixed values for testing. Exponential linear units(ELU) [8] introduced by Clevert et al. to achieve a faster learning and higher classification accuracies. ELU employs a saturation function as negative part and its function is defined as it will decrease the variation of units if they are deactivated.

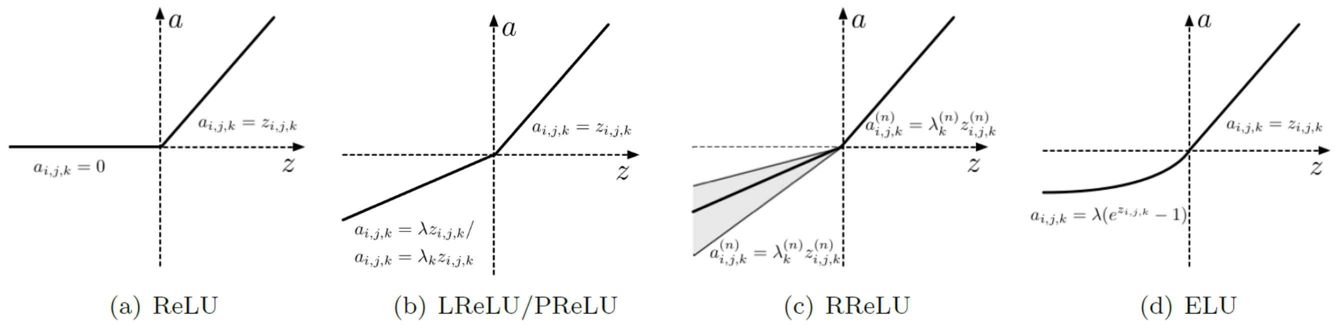


Figure 3: Variations of ReLU activation function

Pooling layer

The pooling layer responsible for lower the computational cost by spatial dimensional reduction of the feature maps and hence the number of connections between convolution layers. Depending on the function of pooling, the process preserves the important information from input feature maps to outputs while reducing the dimensionality. It also helps to achieve translational invariances in features by providing robustness to noise and small distortions. The typical pooling functions are max pooling and average pooling. Max pooling returns the maximum values from locally captured pooling window across each feature map. Average pooling computes the value within each local pooling window and returns as the output across feature maps. Compared to the max pooling, average pooling does not ignore any value in the feature map but takes all values in a local window into account and average over.

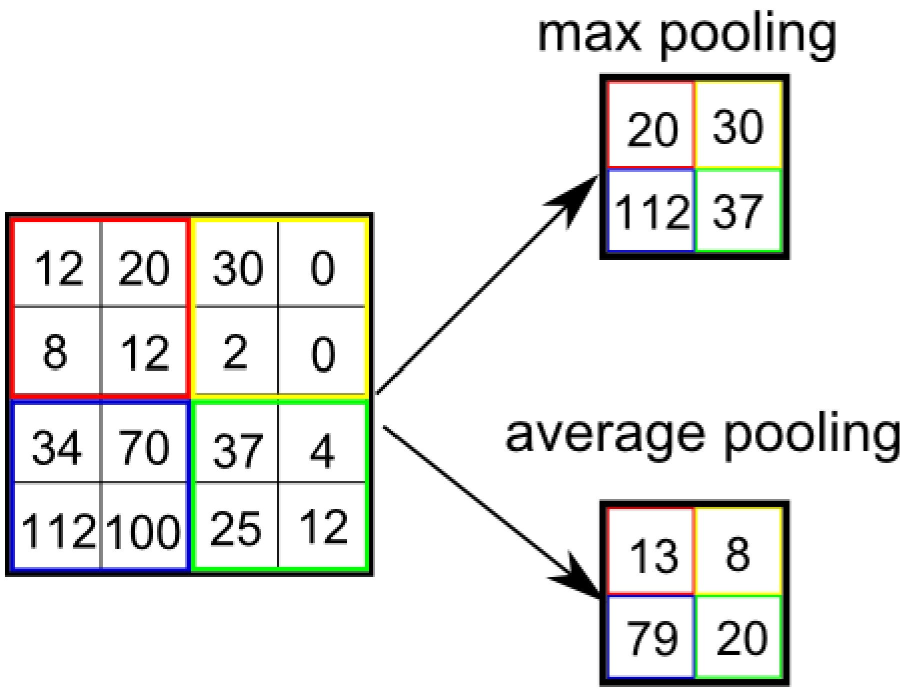


Figure 4: Max pooling and average pooling

In addition to main 2 methods, mixed pooling[9] is introduced by Yu et al. As its name means, mixed pooling are a combination of max pooling and average pooling. The pooling function contains both max pooling, average pooling and additional parameter to trade-off the operation between both methods. Inspired from dropout, stochastic pooling[10] randomly picks the values from feature maps according to a multinominal distribution. This method ensures that the non-maximal values are also recognized. Furthermore, spectral pooling[11] method follows the pooling process by cropping the feature map in frequency domain. The frequency domain conversion is performed by discrete Fourier transform. Once the cropping is done inverse transformed back to the spatial domain.

Recent advancements in main operations of a CNN are briefly discussed throughout this article. Next article will be discussing most influential CNN architectures that have shaped current state of the art.

References

[1] J. Ngiam, Z. Chen, D. Chia, P. W. Koh, Q. V. Le, A. Y. Ng, Tiled convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2010, pp. 1279–1287.
[2] F. Yu, V. Koltun, Multi-scale context aggregation by dilated convolutions, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.
[3] M. Lin, Q. Chen, S. Yan, Network in network, in: Proceedings of the International Conference on Learning Representations (ICLR), 2014.
[4] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proceedings of the International Conference on Machine Learning (ICML), 2010, pp. 807–814.
[5] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proceedings of the International Conference on Machine Learning (ICML), Vol. 30, 2013.
[6] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the International Conference on Computer Vision (ICCV), 2015, pp. 1026–1034.
[7] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, in: Proceedings of the International Conference on Machine Learning (ICML) Workshop, 2015.
[8] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.
[9] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, CoRR abs/1207.0580.
[10] M. D. Zeiler, R. Fergus, Stochastic pooling for regularization of deep convolutional neural networks, in: Proceedings of the International Conference on Learning Representations (ICLR), 2013.
[11] O. Rippel, J. Snoek, R. P. Adams, Spectral representations for convolutional neural

networks, in: Proceedings of the Advances in Neural Information Processing Systems (NIPS), 2015, pp. 2449–2457.

Machine Learning Convolutional Network Deep Learning Neural Networks

Artificial Intelligence