WIKIPEDIA

Convolutional neural network

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of artificial neural network (**ANN**), most commonly applied to analyze visual imagery.[1] CNNs are also known as **Shift Invariant** or **Space Invariant Artificial Neural Networks** (**SIANN**), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps.[2][3] Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the downsampling operation they apply to the input.[4] They have applications in image and video recognition, recommender systems,[5] image classification, image segmentation, medical image analysis, natural language processing,[6] brain–computer interfaces,[7] and financial time series.[8]

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks make them prone to overfitting data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

Convolutional networks were inspired by biological processes[9][10][11][12] in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

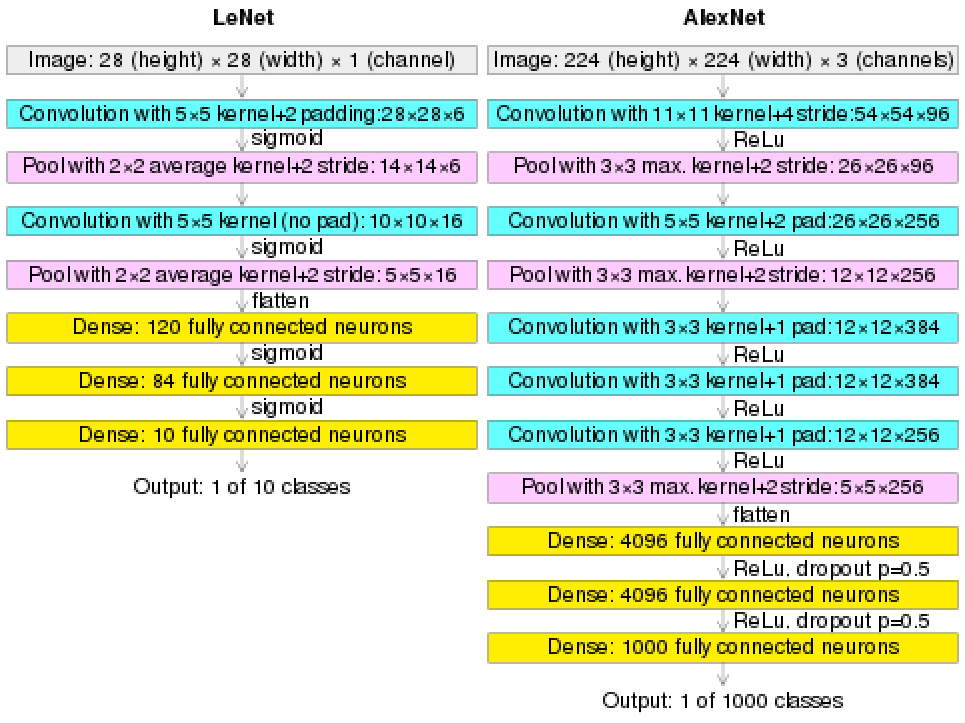
Contents

Definition

Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers.[13] They are specifically designed to process pixel data and are used in image recognition and processing.

Architecture

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.


LeNet
Image: 28 (height) × 28 (width) × 1 (channel)
Convolution with 5×5 kernel+2 padding:28×28×6
sigmoid
Pool with 2×2 average kernel+2 stride: 14×14×6
Convolution with 5×5 kernel (no pad): 10×10×16
sigmoid
Pool with 2×2 average kernel+2 stride: 5×5×16
flatten
Dense: 120 fully connected neurons
sigmoid
Dense: 84 fully connected neurons
sigmoid
Dense: 10 fully connected neurons
Output: 1 of 10 classes
AlexNet
Image: 224 (height) × 224 (width) × 3 (channels)
Convolution with 11×11 kernel+4 stride:54×54×96
ReLu
Pool with 3×3 max. kernel+2 stride: 26×26×96
Convolution with 5×5 kernel+2 pad:26×26×256
ReLu
Pool with 3×3 max. kernel+2 stride: 12×12×256
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×384
ReLu
Convolution with 3×3 kernel+1 pad:12×12×256
ReLu
Pool with 3×3 max. kernel+2 stride:5×5×256
flatten
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 4096 fully connected neurons
ReLu, dropout p=0.5
Dense: 1000 fully connected neurons
Output: 1 of 1000 classes

Comparison of the LeNet and AlexNet convolution, pooling and dense layers (AlexNet image size should be 227x227x3, instead of 224x224x3, so the math will come out right. The original paper said different numbers, but Andrej Karpathy, the head of computer vision at Tesla, said it should be 227x227x3 (he said Alex didn't describe why he put 224x224x3). The next convolution should be 11x11 with stride 4: 55x55x96 (instead of 54x54x96). It would be calculated, for example, as: [(input width 227 - kernel width 11) / stride 4] + 1 = [(227 - 11) / 4] + 1 = 55. Since the kernel output is the same length as width, its area is 55x55.)

Convolutional layers

In a CNN, the input is a tensor with a shape: (number of inputs) x (input height) x (input width) x (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) x (feature map height) x (feature map width) x (feature map channels).

Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus.[14] Each convolutional neuron processes data only for its receptive field. Although fully connected feedforward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs such as high-resolution images. It would require a very high number of neurons, even in a shallow architecture, due to the large input size of images, where each pixel is a relevant input feature. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10,000 weights for *each* neuron in the second layer. Instead, convolution reduces the number of free parameters, allowing the network to be deeper.[15] For example, regardless of image size, using a 5 x 5 tiling region, each with the same shared weights, requires only 25 learnable parameters. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during backpropagation in traditional neural networks.[16][17] Furthermore, convolutional neural networks are ideal for data with a grid-like topology (such as images) as spatial relations between separate features are taken into account during convolution and/or pooling.

Pooling layers

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 x 2 are commonly used. Global pooling acts on all the neurons of the feature map.[18][19] There are two common types of pooling in popular use: max and average. *Max pooling* uses the maximum value of each local cluster of neurons in the feature map,[20][21] while *average pooling* takes the average value.

Fully connected layers

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Receptive field

In neural networks, each neuron receives input from some number of locations in the previous layer. In a convolutional layer, each neuron receives input from only a restricted area of the previous layer called the neuron's *receptive field*. Typically the area is a square (e.g. 5 by 5 neurons). Whereas, in a fully connected layer, the receptive field is the *entire previous layer*. Thus, in each convolutional layer, each neuron takes input from a larger area in the input than previous layers. This is due to applying the convolution over and over, which takes into account the value of a pixel, as well as its surrounding pixels. When using dilated layers, the number of pixels in the receptive field remains constant, but the field is more sparsely populated as its dimensions grow when combining the effect of several layers.

Weights

Each neuron in a neural network computes an output value by applying a specific function to the input values received from the receptive field in the previous layer. The function that is applied to the input values is determined by a vector of weights and a bias (typically real numbers). Learning consists of iteratively adjusting these biases and weights.

The vectors of weights and biases are called *filters* and represent particular features of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons can share the same filter. This reduces the memory footprint because a single bias and a single vector of weights are used across all receptive fields that share that filter, as opposed to each receptive field having its own bias and vector weighting.[22]

History

CNN are often compared to the way the brain achieves vision processing in living organisms.[23]

Receptive fields in the visual cortex

Work by Hubel and Wiesel in the 1950s and 1960s showed that cat visual cortices contain neurons that individually respond to small regions of the visual field. Provided the eyes are not moving, the region of visual space within which visual stimuli affect the firing of a single neuron is known as its receptive field.[24] Neighboring cells have similar and overlapping receptive fields. Receptive field size and location varies systematically across the cortex to form a complete map of visual space. The cortex in each hemisphere represents the contralateral visual field.

Their 1968 paper identified two basic visual cell types in the brain:[10]

- simple cells, whose output is maximized by straight edges having particular orientations within their receptive field
- complex cells, which have larger receptive fields, whose output is insensitive to the exact position of the edges in the field.

Hubel and Wiesel also proposed a cascading model of these two types of cells for use in pattern recognition tasks.[25][24]

Neocognitron, origin of the CNN architecture

The "neocognitron"[9] was introduced by Kunihiko Fukushima in 1980.[11][21][26] It was inspired by the above-mentioned work of Hubel and Wiesel. The neocognitron introduced the two basic types of layers in CNNs: convolutional layers, and downsampling layers. A convolutional layer contains units whose receptive fields cover a patch of the previous layer. The weight vector (the set of adaptive parameters) of such a unit is often called a filter. Units can share filters. Downsampling layers contain units whose receptive fields cover patches of previous convolutional layers. Such a unit typically computes the average of the activations of the units in its patch. This downsampling helps to correctly classify objects in visual scenes even when the objects are shifted.

In a variant of the neocognitron called the cresceptron, instead of using Fukushima's spatial averaging, J. Weng et al. introduced a method called max-pooling where a downsampling unit computes the maximum of the activations of the units in its patch.[27] Max-pooling is often used in modern CNNs.[28]

Several supervised and unsupervised learning algorithms have been proposed over the decades to train the weights of a neocognitron.[9] Today, however, the CNN architecture is usually trained through backpropagation.

The neocognitron is the first CNN which requires units located at multiple network positions to have shared weights.

Convolutional neural networks were presented at the Neural Information Processing Workshop in 1987, automatically analyzing time-varying signals by replacing learned multiplication with convolution in time, and demonstrated for speech recognition.[29]

Time delay neural networks

The time delay neural network (TDNN) was introduced in 1987 by Alex Waibel et al. and was one of the first convolutional networks, as it achieved shift invariance.[30] It did so by utilizing weight sharing in combination with backpropagation training.[31] Thus, while also using a pyramidal structure as in the neocognitron, it performed a global optimization of the weights instead of a local one.[30]

TDNNs are convolutional networks that share weights along the temporal dimension.[32] They allow speech signals to be processed time-invariantly. In 1990 Hampshire and Waibel introduced a variant which performs a two dimensional convolution.[33] Since these TDNNs operated on spectrograms, the resulting phoneme recognition system was invariant to both shifts in time and in frequency. This inspired translation invariance in image processing with CNNs.[31] The tiling of neuron outputs can cover timed stages.[34]

TDNNs now achieve the best performance in far distance speech recognition.[35]

Max pooling

In 1990 Yamaguchi et al. introduced the concept of max pooling, which is a fixed filtering operation that calculates and propagates the maximum value of a given region. They did so by combining TDNNs with max pooling in order to realize a speaker independent isolated word recognition system.[20] In their system they used several TDNNs per word, one for each syllable. The results of each TDNN over the input signal were combined using max pooling and the outputs of the pooling layers were then passed on to networks performing the actual word classification.

Image recognition with CNNs trained by gradient descent

A system to recognize hand-written ZIP Code numbers[36] involved convolutions in which the kernel coefficients had been laboriously hand designed.[37]

Yann LeCun et al. (1989)[37] used back-propagation to learn the convolution kernel coefficients directly from images of hand-written numbers. Learning was thus fully automatic, performed better than manual coefficient design, and was suited to a broader range of image recognition problems and image types.

This approach became a foundation of modern computer vision.

LeNet-5

LeNet-5, a pioneering 7-level convolutional network by LeCun et al. in 1998,[38] that classifies digits, was applied by several banks to recognize hand-written numbers on checks (British English: cheques) digitized in 32x32 pixel images. The ability to process higher-resolution images requires larger and more layers of convolutional neural networks, so this technique is constrained by the availability of computing resources.

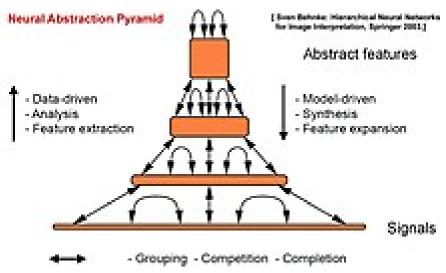
Shift-invariant neural network

Similarly, a shift invariant neural network was proposed by W. Zhang et al. for image character recognition in 1988.[2][3] The architecture and training algorithm were modified in 1991[39] and applied for medical image processing[40] and automatic detection of breast cancer in mammograms.[41]

A different convolution-based design was proposed in 1988[42] for application to decomposition of one-dimensional electromyography convolved signals via de-convolution. This design was modified in 1989 to other de-convolution-based designs.[43][44]

Neural abstraction pyramid

The feed-forward architecture of convolutional neural networks was extended in the neural abstraction pyramid[45] by lateral and feedback connections. The resulting recurrent convolutional network allows for the flexible incorporation of contextual information to iteratively resolve local ambiguities. In contrast to previous models, image-like outputs at the highest resolution were generated, e.g., for semantic segmentation, image reconstruction, and object localization tasks.


Neural Abstraction Pyramid
[Sven Behnke: Hierarchical Neural Networks for Image Interpretation, Springer 2003]
Abstract features
- Data-driven
- Analysis
- Feature extraction
- Model-driven
- Synthesis
- Feature expansion
Signals
- Grouping - Competition - Completion
Neural abstraction pyramid

GPU implementations

Although CNNs were invented in the 1980s, their breakthrough in the 2000s required fast implementations on graphics processing units (GPUs).

In 2004, it was shown by K. S. Oh and K. Jung that standard neural networks can be greatly accelerated on GPUs. Their implementation was 20 times faster than an equivalent implementation on CPU.[46][28] In 2005, another paper also emphasised the value of GPGPU for machine learning.[47]

The first GPU-implementation of a CNN was described in 2006 by K. Chellapilla et al. Their implementation was 4 times faster than an equivalent implementation on CPU.[48] Subsequent work also used GPUs, initially for other types of neural networks (different from CNNs), especially unsupervised neural networks.[49][50][51][52]

In 2010, Dan Ciresan et al. at IDSIA showed that even deep standard neural networks with many layers can be quickly trained on GPU by supervised learning through the old method known as backpropagation. Their network outperformed previous machine learning methods on the MNIST handwritten digits benchmark.[53] In 2011, they extended this GPU approach to CNNs, achieving an acceleration factor of 60, with impressive results.[18] In 2011, they used such CNNs on GPU to win an image recognition contest where they achieved superhuman performance for the first time.[54] Between May 15, 2011 and September 30, 2012, their CNNs won no less than four image competitions.[55][28] In 2012, they also significantly improved on the best performance in the literature for multiple image databases, including the MNIST database, the NORB database, the HWDB1.0 dataset (Chinese characters) and the CIFAR10 dataset (dataset of 60000 32x32 labeled RGB images).[21]

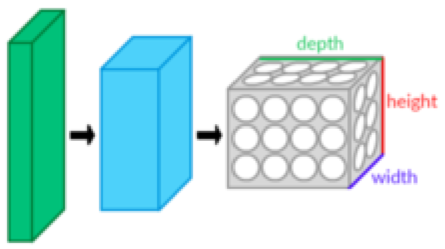
Subsequently, a similar GPU-based CNN by Alex Krizhevsky et al. won the ImageNet Large Scale Visual Recognition Challenge 2012.[56] A very deep CNN with over 100 layers by Microsoft won the ImageNet 2015 contest.[57]

Intel Xeon Phi implementations

Compared to the training of CNNs using GPUs, not much attention was given to the Intel Xeon Phi coprocessor.[58] A notable development is a parallelization method for training convolutional neural networks on the Intel Xeon Phi, named Controlled Hogwild with Arbitrary Order of Synchronization (CHAOS).[59] CHAOS exploits both the thread- and SIMD-level parallelism that is available on the Intel Xeon Phi.

Distinguishing features

In the past, traditional multilayer perceptron (MLP) models were used for image recognition. However, the full connectivity between nodes caused the curse of dimensionality, and was computationally intractable with higher-resolution images. A 1000×1000-pixel image with RGB color channels has 3 million weights per fully-connected neuron, which is too high to feasibly process efficiently at scale.



CNN layers arranged in 3 dimensions

For example, in CIFAR-10, images are only of size 32×32×3 (32 wide, 32 high, 3 color channels), so a single fully connected neuron in the first hidden layer of a regular neural network would have 32*32*3 = 3,072 weights. A 200×200 image, however, would lead to neurons that have 200*200*3 = 120,000 weights.

Also, such network architecture does not take into account the spatial structure of data, treating input pixels which are far apart in the same way as pixels that are close together. This ignores locality of reference in data with a grid-topology (such as images), both computationally and semantically. Thus, full connectivity of neurons is wasteful for purposes such as image recognition that are dominated by spatially local input patterns.

Convolutional neural networks are variants of multilayer perceptrons, designed to emulate the behavior of a visual cortex. These models mitigate the challenges posed by the MLP architecture by exploiting the strong spatially local correlation present in natural images. As opposed to MLPs, CNNs have the following distinguishing features:

- 3D volumes of neurons. The layers of a CNN have neurons arranged in 3 dimensions: width, height and depth.[60] Where each neuron inside a convolutional layer is connected to only a small region of the layer before it, called a receptive field. Distinct types of layers, both locally and completely connected, are stacked to form a CNN architecture.
- Local connectivity: following the concept of receptive fields, CNNs exploit spatial locality by enforcing a local connectivity pattern between neurons of adjacent layers. The architecture thus ensures that the learned "filters" produce the strongest response to a spatially local input pattern. Stacking many such layers leads to nonlinear filters that become increasingly global (i.e. responsive to a larger region of pixel space) so that the network first creates

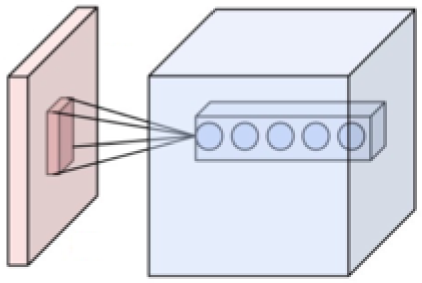
representations of small parts of the input, then from them assembles representations of larger areas.

- Shared weights: In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map. This means that all the neurons in a given convolutional layer respond to the same feature within their specific response field. Replicating units in this way allows for the resulting activation map to be equivariant under shifts of the locations of input features in the visual field, i.e. they grant translational equivariance - given that the layer has a stride of one.[61]
- Pooling: In a CNN's pooling layers, feature maps are divided into rectangular sub-regions, and the features in each rectangle are independently down-sampled to a single value, commonly by taking their average or maximum value. In addition to reducing the sizes of feature maps, the pooling operation grants a degree of local translational invariance to the features contained therein, allowing the CNN to be more robust to variations in their positions.[4]

Together, these properties allow CNNs to achieve better generalization on vision problems. Weight sharing dramatically reduces the number of free parameters learned, thus lowering the memory requirements for running the network and allowing the training of larger, more powerful networks.

Building blocks

A CNN architecture is formed by a stack of distinct layers that transform the input volume into an output volume (e.g. holding the class scores) through a differentiable function. A few distinct types of layers are commonly used. These are further discussed below.


Neurons of a convolutional layer (blue), connected to their receptive field (red)

Convolutional layer

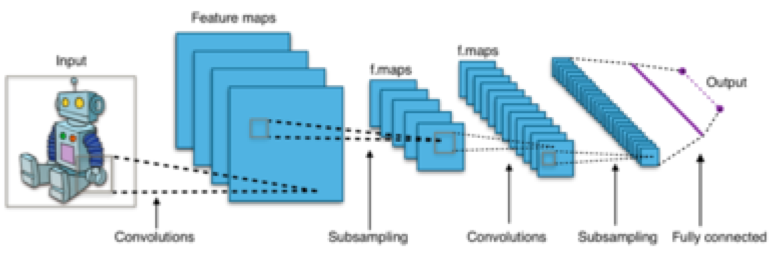
The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters (or kernels), which have a small receptive field, but extend through the full depth of the input volume. During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the filter entries and the input, producing a 2-dimensional activation map of that filter. As a result, the network learns filters that activate when it detects some specific type of feature at some spatial position in the input.[62][nb 1]

Stacking the activation maps for all filters along the depth dimension forms the full output volume of the convolution layer. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same activation map.

Local connectivity

When dealing with high-dimensional inputs such as images, it is impractical to connect neurons to all neurons in the previous volume because such a network architecture does not take the spatial structure of the data into account. Convolutional networks exploit spatially local correlation by enforcing a sparse local connectivity pattern between neurons of adjacent layers: each neuron is connected to only a small region of the input volume.

The extent of this connectivity is a hyperparameter called the receptive field of the neuron. The connections are local in space (along width and height), but always extend along the entire depth of the input volume. Such an architecture ensures that the learnt filters produce the strongest response to a spatially local input pattern.



Typical CNN architecture

Spatial arrangement

Three hyperparameters control the size of the output volume of the convolutional layer: the depth, stride, and padding size:

- The *depth* of the output volume controls the number of neurons in a layer that connect to the same region of the input volume. These neurons learn to activate for different features in the input. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.
- *Stride* controls how depth columns around the width and height are allocated. If the stride is 1, then we move the filters one pixel at a time. This leads to heavily overlapping receptive fields between the columns, and to large output volumes. For any integer $S > 0$, a stride S means that the filter is translated S units at a time per output. In practice, $S \geq 3$ is rare. A greater stride means smaller overlap of receptive fields and smaller spatial dimensions of the output volume.[63]
- Sometimes, it is convenient to pad the input with zeros (or other values, such as the average of the region) on the border of the input volume. The size of this padding is a third hyperparameter. Padding provides control of the output volume's spatial size. In particular, sometimes it is desirable to exactly preserve the spatial size of the input volume, this is commonly referred to as "same" padding.

The spatial size of the output volume is a function of the input volume size W, the kernel field size K of the convolutional layer neurons, the stride S, and the amount of zero padding P on the border. The number of neurons that "fit" in a given volume is then:

$$\frac{W-K+2P}{S}+1.$$

If this number is not an integer, then the strides are incorrect and the neurons cannot be tiled to fit across the input volume in a symmetric way. In general, setting zero padding to be $P=(K-1)/2$ when the stride is $S=1$ ensures that the input volume and output volume will have the same size spatially. However, it is not always completely necessary to use all of the neurons of the previous layer. For example, a neural network designer may decide to use just a portion of padding.

Parameter sharing

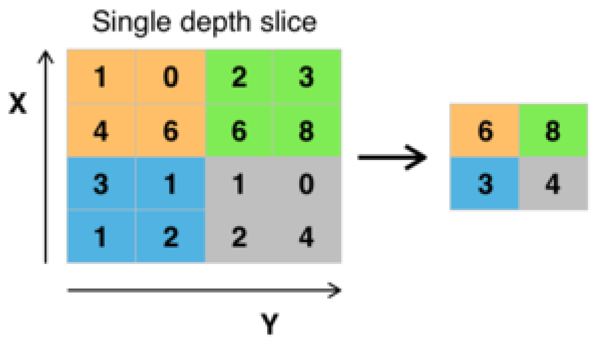
A parameter sharing scheme is used in convolutional layers to control the number of free parameters. It relies on the assumption that if a patch feature is useful to compute at some spatial position, then it should also be useful to compute at other positions. Denoting a single 2-dimensional slice of depth as a *depth slice*, the neurons in each depth slice are constrained to use the same weights and bias.

Since all neurons in a single depth slice share the same parameters, the forward pass in each depth slice of the convolutional layer can be computed as a convolution of the neuron's weights with the input volume.[nb 2] Therefore, it is common to refer to the sets of weights as a filter (or a kernel), which is convolved with the input. The result of this convolution is an activation map, and the set of activation maps for each different filter are stacked together along the depth dimension to produce the output volume. Parameter sharing contributes to the translation invariance of the CNN architecture.[4]

Sometimes, the parameter sharing assumption may not make sense. This is especially the case when the input images to a CNN have some specific centered structure; for which we expect completely different features to be learned on different spatial locations. One practical example is when the inputs are faces that have been centered in the image: we might expect different eye-specific or hair-specific features to be learned in different parts of the image. In that case it is common to relax the parameter sharing scheme, and instead simply call the layer a "locally connected layer".

Pooling layer

Another important concept of CNNs is pooling, which is a form of non-linear down-sampling. There are several non-linear functions to implement pooling, where *max pooling* is the most common. It partitions the input image into a set of rectangles and, for each such sub-region, outputs the maximum.



Max pooling with a 2x2 filter and stride = 2

Intuitively, the exact location of a feature is less important than its rough location relative to other features. This is the idea behind the use of pooling in convolutional neural networks. The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters, memory footprint and amount of computation in the network, and hence to also control overfitting. This is known as down-sampling. It is common to periodically insert a pooling layer between successive convolutional layers (each one typically followed by an activation function, such as a ReLU layer) in a CNN architecture.[62]: 460–461 While pooling layers contribute to local translation invariance, they do not provide global translation invariance in a CNN, unless a form of global pooling is used.[4][61] The pooling layer commonly operates independently on every depth, or slice, of the input and resizes it spatially. A very common form of max pooling is a layer with filters of size 2×2, applied with a stride of 2, which subsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations:

$$f_{X,Y}(S) = \max_{a,b=0}^{1} S_{2X+a,2Y+b}.$$

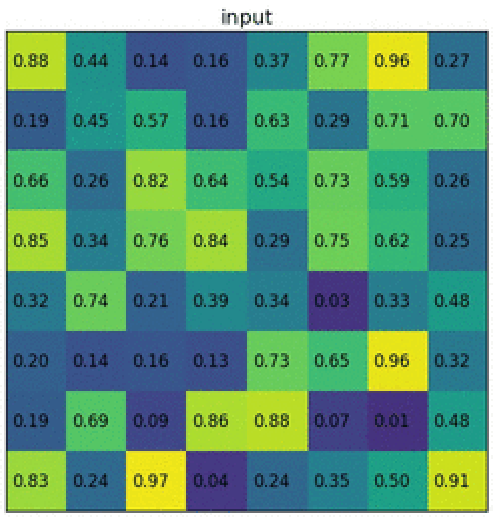
In this case, every max operation is over 4 numbers. The depth dimension remains unchanged (this is true for other forms of pooling as well).

In addition to max pooling, pooling units can use other functions, such as average pooling or ℓ_2-norm pooling. Average pooling was often used historically but has recently fallen out of favor compared to max pooling, which generally performs better in practice.[64]

Due to the effects of fast spatial reduction of the size of the representation, there is a recent trend towards using smaller filters[65] or discarding pooling layers altogether.[66]

"Region of Interest" pooling (also known as RoI pooling) is a variant of max pooling, in which output size is fixed and input rectangle is a parameter.[67]

Pooling is a downsampling method and an important component of convolutional neural networks for object detection based on the Fast R-CNN[68] architecture.



0.88	0.44	0.14	0.16	0.37	0.77	0.96	0.27
0.19	0.45	0.57	0.16	0.63	0.29	0.71	0.70
0.66	0.26	0.82	0.64	0.54	0.73	0.59	0.26
0.85	0.34	0.76	0.84	0.29	0.75	0.62	0.25
0.32	0.74	0.21	0.39	0.34	0.03	0.33	0.48
0.20	0.14	0.16	0.13	0.73	0.65	0.96	0.32
0.19	0.69	0.09	0.86	0.88	0.07	0.01	0.48
0.83	0.24	0.97	0.04	0.24	0.35	0.50	0.91

RoI pooling to size 2x2. In this example region proposal (an input parameter) has size 7x5.

Channel Max Pooling

A CMP operation layer conducts the MP operation along the channel side among the corresponding positions of the consecutive feature maps for the purpose of redundant information elimination. The CMP makes the significant features gather together within fewer channels, which is important for fine-grained image classification that needs more discriminating features. Meanwhile, another advantage of the CMP operation is to make the channel number of feature maps smaller before it connects to the first fully connected (FC) layer. Similar to the MP operation, we denote the input feature maps and output feature maps of a CMP layer as F ∈ R(C×M×N) and C ∈ R(c×M×N), respectively, where C and c are the channel numbers of the input and output feature maps, M and N are the widths and the height of the feature maps, respectively. Note that the CMP operation only changes the channel number of the feature maps. The width and the height of the feature maps are not changed, which is different from the MP operation.[69]

ReLU layer

ReLU is the abbreviation of rectified linear unit, which applies the non-saturating activation function $f(x)=\max(0,x)$.[56] It effectively removes negative values from an activation map by setting them to zero.[70] It introduces nonlinearities to the decision function and in the overall network without affecting the receptive fields of the convolution layers.

Other functions can also be used to increase nonlinearity, for example the saturating hyperbolic tangent $f(x)=\tanh(x)$, $f(x)=|\tanh(x)|$, and the sigmoid function $\sigma(x)=(1+e^{-x})^{-1}$. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.[71]

Fully connected layer

After several convolutional and max pooling layers, the final classification is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer, as seen in regular (non-convolutional) artificial neural networks. Their activations can thus be computed as an affine transformation, with matrix multiplication followed by a bias offset (vector addition of a learned or fixed bias term).

Loss layer

The "loss layer", or "loss function", specifies how training penalizes the deviation between the predicted output of the network, and the true data labels (during supervised learning). Various loss functions can be used, depending on the specific task.

The Softmax loss function is used for predicting a single class of K mutually exclusive classes.[nb 3] Sigmoid cross-entropy loss is used for predicting K independent probability values in $[0,1]$. Euclidean loss is used for regressing to real-valued labels $(-\infty,\infty)$.

Hyperparameters

Hyperparameters are various settings that are used to control the learning process. CNNs use more hyperparameters than a standard multilayer perceptron (MLP).

Kernel size

The kernel is the number of pixels processed together. It is typically expressed as the kernel's dimensions, e.g., 2x2, or 3x3.

Padding

Padding is the addition of (typically) 0-valued pixels on the borders of an image. This is done so that the border pixels are not undervalued (lost) from the output because they would ordinarily participate in only a single receptive field instance. The padding applied is typically one less than the corresponding kernel dimension. For example, a convolutional layer using 3x3 kernels would receive a 2-pixel pad, that is 1 pixel on each side of the image.[72]

Stride

The stride is the number of pixels that the analysis window moves on each iteration. A stride of 2 means that each kernel is offset by 2 pixels from its predecessor.

Number of filters

Since feature map size decreases with depth, layers near the input layer tend to have fewer filters while higher layers can have more. To equalize computation at each layer, the product of feature values v_a with pixel position is kept roughly constant across layers. Preserving more information about the input would require keeping the total number of activations (number of feature maps times number of pixel positions) non-decreasing from one layer to the next.

The number of feature maps directly controls the capacity and depends on the number of available examples and task complexity.

Filter size

Common filter sizes found in the literature vary greatly, and are usually chosen based on the data set.

The challenge is to find the right level of granularity so as to create abstractions at the proper scale, given a particular data set, and without overfitting.

Pooling type and size

Max pooling is typically used, often with a 2x2 dimension. This implies that the input is drastically downsampled, reducing processing cost.

Large input volumes may warrant 4×4 pooling in the lower layers.[73] Greater pooling reduces the dimension of the signal, and may result in unacceptable information loss. Often, non-overlapping pooling windows perform best.[64]

Dilation

Dilation involves ignoring pixels within a kernel. This reduces processing/memory potentially without significant signal loss. A dilation of 2 on a 3x3 kernel expands the kernel to 7x7, while still processing 9 (evenly spaced) pixels. Accordingly, dilation of 4 expands the kernel to 15x15.[74]

Translation equivariance and aliasing

It is commonly assumed that CNNs are invariant to shifts of the input. Convolution or pooling layers within a CNN that do not have a stride greater than one are indeed equivariant to translations of the input.[61] However, layers with a stride greater than one ignore the Nyquist-Shannon sampling theorem and might lead to aliasing of the input signal[61] While, in principle, CNNs are capable of implementing anti-aliasing filters, it has been observed that this does not happen in practice [75] and yield models that are not equivariant to translations. Furthermore, if a CNN makes use of fully connected layers, translation equivariance does not imply translation invariance, as the fully connected layers are not invariant to shifts of the input.[76][4] One solution for complete translation invariance is avoiding any down-sampling throughout the network and applying global average pooling at the last layer.[61] Additionally, several other partial solutions have been proposed, such as anti-aliasing before downsampling operations,[77] spatial transformer networks,[78] data augmentation, subsampling combined with pooling,[4] and capsule neural networks.[79]

Evaluation

The accuracy of the final model based on a sub-part of the dataset set apart at the start, often called a test-set. Other times methods such as *k*-fold cross-validation are applied. Other strategies include using conformal prediction.[80][81]

Regularization methods

Regularization is a process of introducing additional information to solve an ill-posed problem or to prevent overfitting. CNNs use various types of regularization.

Empirical

Dropout

Because a fully connected layer occupies most of the parameters, it is prone to overfitting. One method to reduce overfitting is dropout.[82][83] At each training stage, individual nodes are either "dropped out" of the net (ignored) with probability $1-p$ or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights.

In the training stages, p is usually 0.5; for input nodes, it is typically much higher because information is directly lost when input nodes are ignored.

At testing time after training has finished, we would ideally like to find a sample average of all possible 2^n dropped-out networks; unfortunately this is unfeasible for large values of n. However, we can find an approximation by using the full network with each node's output weighted by a factor of p, so the expected value of the output of any node is the same as in the training stages. This is the biggest contribution of the dropout method: although it effectively generates 2^n neural nets, and as such allows for model combination, at test time only a single network needs to be tested.

By avoiding training all nodes on all training data, dropout decreases overfitting. The method also significantly improves training speed. This makes the model combination practical, even for deep neural networks. The technique seems to reduce node interactions, leading them to learn more robust features that better generalize to new data.

DropConnect

DropConnect is the generalization of dropout in which each connection, rather than each output unit, can be dropped with probability $1-p$. Each unit thus receives input from a random subset of units in the previous layer.[84]

DropConnect is similar to dropout as it introduces dynamic sparsity within the model, but differs in that the sparsity is on the weights, rather than the output vectors of a layer. In other words, the fully connected layer with DropConnect becomes a sparsely connected layer in which the connections are chosen at random during the training stage.

Stochastic pooling

A major drawback to Dropout is that it does not have the same benefits for convolutional layers, where the neurons are not fully connected.

In stochastic pooling,[85] the conventional deterministic pooling operations are replaced with a stochastic procedure, where the activation within each pooling region is picked randomly according to a multinomial distribution, given by the activities within the pooling region. This approach is free of hyperparameters and can be combined with other regularization approaches, such as dropout and data augmentation.

An alternate view of stochastic pooling is that it is equivalent to standard max pooling but with many copies of an input image, each having small local deformations. This is similar to explicit elastic deformations of the input images,[86] which delivers excellent performance on the MNIST data set.[86] Using stochastic pooling in a multilayer model gives an exponential number of deformations since the selections in higher layers are independent of those below.

Artificial data

Because the degree of model overfitting is determined by both its power and the amount of training it receives, providing a convolutional network with more training examples can reduce overfitting. Because these networks are usually trained with all available data, one approach is to either generate new data from scratch (if possible) or perturb existing data to create new ones. For example, input images can be cropped, rotated, or rescaled to create new examples with the same labels as the original training set.[87]

Explicit

Early stopping

One of the simplest methods to prevent overfitting of a network is to simply stop the training before overfitting has had a chance to occur. It comes with the disadvantage that the learning process is halted.

Number of parameters

Another simple way to prevent overfitting is to limit the number of parameters, typically by limiting the number of hidden units in each layer or limiting network depth. For convolutional networks, the filter size also affects the number of parameters. Limiting the number of parameters restricts the predictive power of the network directly, reducing the complexity of the function that it can perform on the data, and thus limits the amount of overfitting. This is equivalent to a "zero norm".

Weight decay

A simple form of added regularizer is weight decay, which simply adds an additional error, proportional to the sum of weights (L1 norm) or squared magnitude (L2 norm) of the weight vector, to the error at each node. The level of acceptable model complexity can be reduced by increasing the proportionality constant('alpha' hyperparameter), thus increasing the penalty for large weight vectors.

L2 regularization is the most common form of regularization. It can be implemented by penalizing the squared magnitude of all parameters directly in the objective. The L2 regularization has the intuitive interpretation of heavily penalizing peaky weight vectors and preferring diffuse weight vectors. Due to multiplicative interactions between weights and inputs this has the useful property of encouraging the network to use all of its inputs a little rather than some of its inputs a lot.

L1 regularization is also common. It makes the weight vectors sparse during optimization. In other words, neurons with L1 regularization end up using only a sparse subset of their most important inputs and become nearly invariant to the noisy inputs. L1 with L2 regularization can be combined; this is called elastic net regularization.

Max norm constraints

Another form of regularization is to enforce an absolute upper bound on the magnitude of the weight vector for every neuron and use projected gradient descent to enforce the constraint. In practice, this corresponds to performing the parameter update as normal, and then enforcing the constraint by clamping the weight vector $\vec{w}$ of every neuron to satisfy $\|\vec{w}\|_2 < c$. Typical values of c are order of 3–4. Some papers report improvements[88] when using this form of regularization.

Hierarchical coordinate frames

Pooling loses the precise spatial relationships between high-level parts (such as nose and mouth in a face image). These relationships are needed for identity recognition. Overlapping the pools so that each feature occurs in multiple pools, helps retain the information. Translation alone cannot extrapolate the understanding of geometric relationships to a radically new viewpoint, such as a different orientation or scale. On the other hand, people are very good at extrapolating; after seeing a new shape once they can recognize it from a different viewpoint.[89]

An earlier common way to deal with this problem is to train the network on transformed data in different orientations, scales, lighting, etc. so that the network can cope with these variations. This is computationally intensive for large data-sets. The alternative is to use a hierarchy of coordinate frames and use a group of neurons to represent a conjunction of the shape of the feature and its pose relative to the retina. The pose relative to the retina is the relationship between the coordinate frame of the retina and the intrinsic features' coordinate frame.[90]

Thus, one way to represent something is to embed the coordinate frame within it. This allows large features to be recognized by using the consistency of the poses of their parts (e.g. nose and mouth poses make a consistent prediction of the pose of the whole face). This approach ensures that the higher-level entity (e.g. face) is present when the lower-level (e.g. nose and mouth) agree on its prediction of the pose. The vectors of neuronal activity that represent pose ("pose vectors") allow spatial transformations modeled as linear operations that make it easier for the network to learn the hierarchy of visual entities and generalize across viewpoints. This is similar to the way the human visual system imposes coordinate frames in order to represent shapes.[91]

Applications

Image recognition

CNNs are often used in image recognition systems. In 2012 an error rate of 0.23% on the MNIST database was reported.[21] Another paper on using CNN for image classification reported that the learning process was "surprisingly fast"; in the same paper, the best published results as of 2011 were achieved in the MNIST database and the NORB database.[18] Subsequently, a similar CNN called AlexNet[92] won the ImageNet Large Scale Visual Recognition Challenge 2012.

When applied to facial recognition, CNNs achieved a large decrease in error rate.[93] Another paper reported a 97.6% recognition rate on "5,600 still images of more than 10 subjects".[12] CNNs were used to assess video quality in an objective way after manual training; the resulting system had a very low root mean square error.[34]

The ImageNet Large Scale Visual Recognition Challenge is a benchmark in object classification and detection, with millions of images and hundreds of object classes. In the ILSVRC 2014,[94] a large-scale visual recognition challenge, almost every highly ranked team used CNN as their basic framework. The winner GoogLeNet[95] (the foundation of DeepDream) increased the mean average precision of object detection to 0.439329, and reduced classification error to 0.06656, the best result to date. Its network applied more than 30 layers. That performance of convolutional neural networks on the ImageNet tests was close to that of humans.[96] The best algorithms still struggle with objects that are small or thin, such as a small ant on a stem of a flower or a person holding a quill in their hand. They also have trouble with images that have been distorted with filters, an increasingly common phenomenon with modern digital cameras. By contrast, those kinds of images rarely trouble humans. Humans, however, tend to have trouble with other issues. For example, they are not good at classifying objects into fine-grained categories such as the particular breed of dog or species of bird, whereas convolutional neural networks handle this.

In 2015 a many-layered CNN demonstrated the ability to spot faces from a wide range of angles, including upside down, even when partially occluded, with competitive performance. The network was trained on a database of 200,000 images that included faces at various angles and orientations and a further 20 million images without faces. They used batches of 128 images over 50,000 iterations.[97]

Video analysis

Compared to image data domains, there is relatively little work on applying CNNs to video classification. Video is more complex than images since it has another (temporal) dimension. However, some extensions of CNNs into the video domain have been explored. One approach is to treat space and time as equivalent dimensions of the input and perform convolutions in both time and space.[98][99] Another way is to fuse the features of two convolutional neural networks, one for the spatial and one for the temporal stream.[100][101][102] Long short-term memory (LSTM) recurrent units are typically incorporated after the CNN to account for inter-frame or inter-clip dependencies.[103][104] Unsupervised learning schemes for training spatio-temporal features have been introduced, based on Convolutional Gated Restricted Boltzmann Machines[105] and Independent Subspace Analysis.[106]

Natural language processing

CNNs have also been explored for natural language processing. CNN models are effective for various NLP problems and achieved excellent results in semantic parsing,[107] search query retrieval,[108] sentence modeling,[109] classification,[110] prediction[111] and other traditional NLP tasks.[112] Compared to traditional language processing methods such as recurrent neural networks, CNNs can represent different contextual realities of language that do not rely on a series-sequence assumption, while RNNs are better suitable when classical time serie modeling is required [113] [114] [115] [116]

Anomaly Detection

A CNN with 1-D convolutions was used on time series in the frequency domain (spectral residual) by an unsupervised model to detect anomalies in the time domain.[117]

Drug discovery

CNNs have been used in drug discovery. Predicting the interaction between molecules and biological proteins can identify potential treatments. In 2015, Atomwise introduced AtomNet, the first deep learning neural network for structure-based drug design.[118] The system trains directly on 3-dimensional representations of chemical interactions. Similar to how image recognition networks learn to compose smaller, spatially proximate features into larger, complex structures,[119] AtomNet discovers chemical features, such as aromaticity, sp^3 carbons, and hydrogen bonding. Subsequently, AtomNet was used to predict novel candidate biomolecules for multiple disease targets, most notably treatments for the Ebola virus[120] and multiple sclerosis.[121]

Health risk assessment and biomarkers of aging discovery

CNNs can be naturally tailored to analyze a sufficiently large collection of time series data representing one-week-long human physical activity streams augmented by the rich clinical data (including the death register, as provided by, e.g., the NHANES study). A simple CNN was combined with Cox-Gompertz proportional hazards model and used to produce a proof-of-concept example of digital biomarkers of aging in the form of all-causes-mortality predictor.[122]

Checkers game

CNNs have been used in the game of checkers. From 1999 to 2001, Fogel and Chellapilla published papers showing how a convolutional neural network could learn to play **checker** using co-evolution. The learning process did not use prior human professional games, but rather focused on a minimal set of information contained in the checkerboard: the location and type of pieces, and the difference in number of pieces between the two sides. Ultimately, the program (Blondie24) was tested on 165 games against players and ranked in the highest 0.4%.[123][124] It also earned a win against the program Chinook at its "expert" level of play.[125]

Go

CNNs have been used in computer Go. In December 2014, Clark and Storkey published a paper showing that a CNN trained by supervised learning from a database of human professional games could outperform GNU Go and win some games against Monte Carlo tree search Fuego 1.1 in a fraction of the time it took Fuego to play.[126] Later it was announced that a large 12-layer convolutional neural network had correctly predicted the professional move in 55% of positions, equalling the accuracy of a 6 dan human player. When the trained convolutional network was used directly to play games of Go, without any search, it beat the traditional search program GNU Go in 97% of games, and matched the performance of the Monte Carlo tree search program Fuego simulating ten thousand playouts (about a million positions) per move.[127]

A couple of CNNs for choosing moves to try ("policy network") and evaluating positions ("value network") driving MCTS were used by AlphaGo, the first to beat the best human player at the time.[128]

Time series forecasting

Recurrent neural networks are generally considered the best neural network architectures for time series forecasting (and sequence modeling in general), but recent studies show that convolutional networks can perform comparably or even better.[129][8] Dilated convolutions[130] might enable one-dimensional convolutional neural networks to effectively learn time series dependences.[131] Convolutions can be implemented more efficiently than RNN-based solutions, and they do not suffer from vanishing (or exploding) gradients.[132] Convolutional networks can provide an improved forecasting performance when there are multiple similar time series to learn from.[133] CNNs can also be applied to further tasks in time series analysis (e.g., time series classification[134] or quantile forecasting[135]).

Cultural Heritage and 3D-datasets

As archaeological findings like clay tablets with cuneiform writing are increasingly acquired using 3D scanners first benchmark datasets are becoming available like *HeiCuBeDa*[136] providing almost 2.000 normalized 2D- and 3D-datasets prepared with the GigaMesh Software Framework.[137] So curvaturebased measures are used in conjunction with Geometric Neural Networks (GNNs) e.g. for period classification of those clay tablets being among the oldest documents of human history.[138][139]

Fine-tuning

For many applications, the training data is less available. Convolutional neural networks usually require a large amount of training data in order to avoid overfitting. A common technique is to train the network on a larger data set from a related domain. Once the network parameters have converged an additional training step is performed using the in-domain data to fine-tune the network weights, this is known as transfer learning. Furthermore, this technique allows convolutional network architectures to successfully be applied to problems with tiny training sets.[140]

Human interpretable explanations

End-to-end training and prediction are common practice in computer vision. However, human interpretable explanations are required for critical systems such as a self-driving cars.[141] With recent advances in visual salience, spatial attention, and temporal attention, the most critical spatial regions/temporal instants could be visualized to justify the CNN predictions.[142][143]

Related architectures

Deep Q-networks

A deep Q-network (DQN) is a type of deep learning model that combines a deep neural network with Q-learning, a form of reinforcement learning. Unlike earlier reinforcement learning agents, DQNs that utilize CNNs can learn directly from high-dimensional sensory inputs via reinforcement learning.[144]

Preliminary results were presented in 2014, with an accompanying paper in February 2015.[145] The research described an application to Atari 2600 gaming. Other deep reinforcement learning models preceded it.[146]

Deep belief networks

Convolutional deep belief networks (CDBN) have structure very similar to convolutional neural networks and are trained similarly to deep belief networks. Therefore, they exploit the 2D structure of images, like CNNs do, and make use of pre-training like deep belief networks. They provide a generic structure that can be used in many image and signal processing tasks. Benchmark results on standard image datasets like CIFAR[147] have been obtained using CDBNs.[148]

Notable libraries

- Caffe: A library for convolutional neural networks. Created by the Berkeley Vision and Learning Center (BVLC). It supports both CPU and GPU. Developed in C++, and has Python and MATLAB wrappers.
- Deeplearning4j: Deep learning in Java and Scala on multi-GPU-enabled Spark. A general-purpose deep learning library for the JVM production stack running on a C++ scientific computing engine. Allows the creation of custom layers. Integrates with Hadoop and Kafka.

- Dlib: A toolkit for making real world machine learning and data analysis applications in C++.
- Microsoft Cognitive Toolkit: A deep learning toolkit written by Microsoft with several unique features enhancing scalability over multiple nodes. It supports full-fledged interfaces for training in C++ and Python and with additional support for model inference in C# and Java.
- TensorFlow: Apache 2.0-licensed Theano-like library with support for CPU, GPU, Google's proprietary tensor processing unit (TPU),[149] and mobile devices.
- Theano: The reference deep-learning library for Python with an API largely compatible with the popular NumPy library. Allows user to write symbolic mathematical expressions, then automatically generates their derivatives, saving the user from having to code gradients or backpropagation. These symbolic expressions are automatically compiled to CUDA code for a fast, on-the-GPU implementation.
- Torch: A scientific computing framework with wide support for machine learning algorithms, written in C and Lua.

See also

- Attention (machine learning)
- Convolution
- Deep learning
- Natural-language processing
- Neocognitron
- Scale-invariant feature transform
- Time delay neural network
- Vision processing unit

Notes

1. When applied to other types of data than image data, such as sound data, "spatial position" may variously correspond to different points in the time domain, frequency domain, or other mathematical spaces.
2. hence the name "convolutional layer"
3. So-called categorical data.

References

1. Valueva, M.V.; Nagornov, N.N.; Lyakhov, P.A.; Valuev, G.V.; Chervyakov, N.I. (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". *Mathematics and Computers in Simulation*. Elsevier BV. **177**: 232–243. doi:10.1016/j.matcom.2020.04.031 (https://doi.org/10.1016%2Fj.matcom.2020.04.031). ISSN 0378-4754 (https://www.worldcat.org/issn/0378-4754). S2CID 218955622 (https://api.semanticscholar.org/CorpusID:218955622). "Convolutional neural networks are a promising tool for solving the problem of pattern recognition."
2. Zhang, Wei (1988). "Shift-invariant pattern recognition neural network and its optical architecture" (https://drive.google.com/file/d/1nN_5odSG_QVae54EsQN_qSz-0ZsX6wA0/view?usp=sharing). *Proceedings of Annual Conference of the Japan Society of Applied Physics*.

3. Zhang, Wei (1990). "Parallel distributed processing model with local space-invariant interconnections and its optical architecture" (https://drive.google.com/file/d/0B65v6Wo67Tk5ODRzZmhSR29VeDg/view?usp=sharing). *Applied Optics*. **29** (32): 4790–7. Bibcode:1990ApOpt..29.4790Z (https://ui.adsabs.harvard.edu/abs/1990ApOpt..29.4790Z). doi:10.1364/AO.29.004790 (https://doi.org/10.1364%2FAO.29.004790). PMID 20577468 (https://pubmed.ncbi.nlm.nih.gov/20577468).
4. Mouton, Coenraad; Myburgh, Johannes C.; Davel, Marelie H. (2020). Gerber, Aurona (ed.). "Stride and Translation Invariance in CNNs" (https://link.springer.com/chapter/10.1007%2F978-3-030-66151-9_17). *Artificial Intelligence Research*. Communications in Computer and Information Science. Cham: Springer International Publishing. **1342**: 267–281. arXiv:2103.10097 (https://arxiv.org/abs/2103.10097). doi:10.1007/978-3-030-66151-9_17 (https://doi.org/10.1007%2F978-3-030-66151-9_17). ISBN 978-3-030-66151-9. S2CID 232269854 (https://api.semanticscholar.org/CorpusID:232269854).
5. van den Oord, Aaron; Dieleman, Sander; Schrauwen, Benjamin (2013-01-01). Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; Weinberger, K. Q. (eds.). *Deep content-based music recommendation* (https://proceedings.neurips.cc/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf) (PDF). Curran Associates, Inc. pp. 2643–2651.
6. Collobert, Ronan; Weston, Jason (2008-01-01). *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning*. *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. New York, NY, USA: ACM. pp. 160–167. doi:10.1145/1390156.1390177 (https://doi.org/10.1145%2F1390156.1390177). ISBN 978-1-60558-205-4. S2CID 2617020 (https://api.semanticscholar.org/CorpusID:2617020).
7. Avilov, Oleksii; Rimbert, Sebastien; Popov, Anton; Bougrain, Laurent (July 2020). "Deep Learning Techniques to Improve Intraoperative Awareness Detection from Electroencephalographic Signals" (https://ieeexplore.ieee.org/document/9176228). *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. Montreal, QC, Canada: IEEE. **2020**: 142–145. doi:10.1109/EMBC44109.2020.9176228 (https://doi.org/10.1109%2FEMBC44109.2020.9176228). ISBN 978-1-7281-1990-8. PMID 33017950 (https://pubmed.ncbi.nlm.nih.gov/33017950). S2CID 221386616 (https://api.semanticscholar.org/CorpusID:221386616).
8. Tsantekidis, Avraam; Passalis, Nikolaos; Tefas, Anastasios; Kanniainen, Juho; Gabbouj, Moncef; Iosifidis, Alexandros (July 2017). "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks". *2017 IEEE 19th Conference on Business Informatics (CBI)*. Thessaloniki, Greece: IEEE: 7–12. doi:10.1109/CBI.2017.23 (https://doi.org/10.1109%2FCBI.2017.23). ISBN 978-1-5386-3035-8. S2CID 4950757 (https://api.semanticscholar.org/CorpusID:4950757).
9. Fukushima, K. (2007). "Neocognitron" (https://doi.org/10.4249%2Fscholarpedia.1717). *Scholarpedia*. **2** (1): 1717. Bibcode:2007SchpJ...2.1717F (https://ui.adsabs.harvard.edu/abs/2007SchpJ...2.1717F). doi:10.4249/scholarpedia.1717 (https://doi.org/10.4249%2Fscholarpedia.1717).
10. Hubel, D. H.; Wiesel, T. N. (1968-03-01). "Receptive fields and functional architecture of monkey striate cortex" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912). *The Journal of Physiology*. **195** (1): 215–243. doi:10.1113/jphysiol.1968.sp008455 (https://doi.org/10.1113%2Fjphysiol.1968.sp008455). ISSN 0022-3751 (https://www.worldcat.org/issn/0022-3751). PMC 1557912 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1557912). PMID 4966457 (https://pubmed.ncbi.nlm.nih.gov/4966457).

11. Fukushima, Kunihiko (1980). "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position" (https://www.cs.princeton.edu/courses/archive/spr08/cos598B/Readings/Fukushima1980.pdf) (PDF). *Biological Cybernetics*. **36** (4): 193–202. doi:10.1007/BF00344251 (https://doi.org/10.1007%2FBF00344251). PMID 7370364 (https://pubmed.ncbi.nlm.nih.gov/7370364). S2CID 206775608 (https://api.semanticscholar.org/CorpusID:206775608). Retrieved 16 November 2013.
12. Matusugu, Masakazu; Katsuhiko Mori; Yusuke Mitari; Yuji Kaneda (2003). "Subject independent facial expression recognition with robust face detection using a convolutional neural network" (http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/sparse/matsugo_etal_face_expression_conv_nnet.pdf) (PDF). *Neural Networks*. **16** (5): 555–559. doi:10.1016/S0893-6080(03)00115-1 (https://doi.org/10.1016%2FS0893-6080%2803%2900115-1). PMID 12850007 (https://pubmed.ncbi.nlm.nih.gov/12850007). Retrieved 17 November 2013.
13. Ian Goodfellow and Yoshua Bengio and Aaron Courville (2016). *Deep Learning* (https://www.deeplearningbook.org/). MIT Press. p. 326.
14. "Convolutional Neural Networks (LeNet) – DeepLearning 0.1 documentation" (http://deeplearning.net/tutorial/lenet.html). *DeepLearning 0.1*. LISA Lab. Retrieved 31 August 2013.
15. Habibi, Aghdam, Hamed (2017-05-30). *Guide to convolutional neural networks : a practical application to traffic-sign detection and classification*. Heravi, Elnaz Jahani. Cham, Switzerland. ISBN 9783319575490. OCLC 987790957 (https://www.worldcat.org/oclc/987790957).
16. Venkatesan, Ragav; Li, Baoxin (2017-10-23). *Convolutional Neural Networks in Visual Computing: A Concise Guide* (https://books.google.com/books?id=bAM7DwAAQBAJ&q=vanishing+gradient). CRC Press. ISBN 978-1-351-65032-8.
17. Balas, Valentina E.; Kumar, Raghvendra; Srivastava, Rajshree (2019-11-19). *Recent Trends and Advances in Artificial Intelligence and Internet of Things* (https://books.google.com/books?id=XRS_DwAAQBAJ&q=exploding+gradient). Springer Nature. ISBN 978-3-030-32644-9.
18. Ciresan, Dan; Ueli Meier; Jonathan Masci; Luca M. Gambardella; Jurgen Schmidhuber (2011). "Flexible, High Performance Convolutional Neural Networks for Image Classification" (https://people.idsia.ch/~juergen/ijcai2011.pdf) (PDF). *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence-Volume Volume Two*. **2**: 1237–1242. Retrieved 17 November 2013.
19. Krizhevsky, Alex. "ImageNet Classification with Deep Convolutional Neural Networks" (https://image-net.org/static_files/files/supervision.pdf) (PDF). Retrieved 17 November 2013.
20. Yamaguchi, Kouichi; Sakamoto, Kenji; Akabane, Toshio; Fujimoto, Yoshiji (November 1990). *A Neural Network for Speaker-Independent Isolated Word Recognition* (https://www.isca-speech.org/archive/icslp_1990/i90_1077.html). First International Conference on Spoken Language Processing (ICSLP 90). Kobe, Japan.
21. Ciresan, Dan; Meier, Ueli; Schmidhuber, Jürgen (June 2012). *Multi-column deep neural networks for image classification*. *2012 IEEE Conference on Computer Vision and Pattern Recognition*. New York, NY: Institute of Electrical and Electronics Engineers (IEEE). pp. 3642–3649. arXiv:1202.2745 (https://arxiv.org/abs/1202.2745). CiteSeerX 10.1.1.300.3283 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.300.3283). doi:10.1109/CVPR.2012.6248110 (https://doi.org/10.1109%2FCVPR.2012.6248110). ISBN 978-1-4673-1226-4. OCLC 812295155 (https://www.worldcat.org/oclc/812295155). S2CID 2161592 (https://api.semanticscholar.org/CorpusID:2161592).
22. LeCun, Yann. "LeNet-5, convolutional neural networks" (http://yann.lecun.com/exdb/lenet/). Retrieved 16 November 2013.

23. Mahapattanakul, Puttatida (November 11, 2019). "From Human Vision to Computer Vision — Convolutional Neural Network(Part3/4)" (https://becominghuman.ai/from-human-vision-to-computer-vision-convolutional-neural-network-part3-4-24b55ffa7045). *Medium*.
24. Hubel, DH; Wiesel, TN (October 1959). "Receptive fields of single neurones in the cat's striate cortex" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130). *J. Physiol*. **148** (3): 574–91. doi:10.1113/jphysiol.1959.sp006308 (https://doi.org/10.1113%2Fjphysiol.1959.sp006308). PMC 1363130 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130). PMID 14403679 (https://pubmed.ncbi.nlm.nih.gov/14403679).
25. David H. Hubel and Torsten N. Wiesel (2005). *Brain and visual perception: the story of a 25-year collaboration* (https://books.google.com/books?id=8YrxWojxUA4C&pg=PA106). Oxford University Press US. p. 106. ISBN 978-0-19-517618-6.
26. LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). "Deep learning". *Nature*. **521** (7553): 436–444. Bibcode:2015Natur.521..436L (https://ui.adsabs.harvard.edu/abs/2015Natur.521..436L). doi:10.1038/nature14539 (https://doi.org/10.1038%2Fnature14539). PMID 26017442 (https://pubmed.ncbi.nlm.nih.gov/26017442). S2CID 3074096 (https://api.semanticscholar.org/CorpusID:3074096).
27. Weng, J; Ahuja, N; Huang, TS (1993). "Learning recognition and segmentation of 3-D objects from 2-D images". *Proc. 4th International Conf. Computer Vision*: 121–128. doi:10.1109/ICCV.1993.378228 (https://doi.org/10.1109%2FICCV.1993.378228). ISBN 0-8186-3870-2. S2CID 8619176 (https://api.semanticscholar.org/CorpusID:8619176).
28. Schmidhuber, Jürgen (2015). "Deep Learning" (http://www.scholarpedia.org/article/Deep_Learning). *Scholarpedia*. **10** (11): 1527–54. CiteSeerX 10.1.1.76.1541 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.76.1541). doi:10.1162/neco.2006.18.7.1527 (https://doi.org/10.1162%2Fneco.2006.18.7.1527). PMID 16764513 (https://pubmed.ncbi.nlm.nih.gov/16764513). S2CID 2309950 (https://api.semanticscholar.org/CorpusID:2309950).
29. Homma, Toshiteru; Les Atlas; Robert Marks II (1988). "An Artificial Neural Network for Spatio-Temporal Bipolar Patters: Application to Phoneme Classification" (https://proceedings.neurips.cc/paper/1987/file/98f13708210194c475687be6106a3b84-Paper.pdf) (PDF). *Advances in Neural Information Processing Systems*. **1**: 31–40.
30. Waibel, Alex (December 1987). *Phoneme Recognition Using Time-Delay Neural Networks*. Meeting of the Institute of Electrical, Information and Communication Engineers (IEICE). Tokyo, Japan.
31. Alexander Waibel et al., *Phoneme Recognition Using Time-Delay Neural Networks (http://www.inf.ufrgs.br/~engel/data/media/file/cmp121/waibel89_TDNN.pdf)* IEEE Transactions on Acoustics, Speech, and Signal Processing, Volume 37, No. 3, pp. 328. - 339 March 1989.
32. LeCun, Yann; Bengio, Yoshua (1995). "Convolutional networks for images, speech, and time series" (https://www.researchgate.net/publication/2453996). In Arbib, Michael A. (ed.). *The handbook of brain theory and neural networks* (Second ed.). The MIT press. pp. 276–278.
33. John B. Hampshire and Alexander Waibel, *Connectionist Architectures for Multi-Speaker Phoneme Recognition (https://proceedings.neurips.cc/paper/1989/file/979d472a84804b9f647bc185a877a8b5-Paper.pdf)*, Advances in Neural Information Processing Systems, 1990, Morgan Kaufmann.
34. Le Callet, Patrick; Christian Viard-Gaudin; Dominique Barba (2006). "A Convolutional Neural Network Approach for Objective Video Quality Assessment" (https://hal.archives-ouvertes.fr/file/index/docid/287426/filename/A_convolutional_neural_network_approach_for_objective_video_quality_assessment_completefinal_manuscript.pdf) (PDF). *IEEE Transactions on Neural Networks*. **17** (5): 1316–1327. doi:10.1109/TNN.2006.879766 (https://doi.org/10.1109%2FTNN.2006.879766). PMID 17001990 (https://pubmed.ncbi.nlm.nih.gov/17001990). S2CID 221185563 (https://api.semanticscholar.org/CorpusID:221185563). Retrieved 17 November 2013.

35. Ko, Tom; Peddinti, Vijayaditya; Povey, Daniel; Seltzer, Michael L.; Khudanpur, Sanjeev (March 2018). *A Study on Data Augmentation of Reverberant Speech for Robust Speech Recognition* (https://www.danielpovey.com/files/2017_icassp_reverberation.pdf) (PDF). The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017). New Orleans, LA, USA.
36. Denker, J S, Gardner, W R, Graf, H. P, Henderson, D, Howard, R E, Hubbard, W, Jackel, L D, BaIrd, H S, and Guyon (1989) Neural network recognizer for hand-written zip code digits (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.852.5499&rep=rep1&type=pdf), AT&T Bell Laboratories
37. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition (http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf); AT&T Bell Laboratories
38. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner (1998). "Gradient-based learning applied to document recognition" (http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf) (PDF). *Proceedings of the IEEE*. **86** (11): 2278–2324. CiteSeerX 10.1.1.32.9552 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.32.9552). doi:10.1109/5.726791 (https://doi.org/10.1109%2F5.726791). S2CID 14542261 (https://api.semanticscholar.org/CorpusID:14542261). Retrieved October 7, 2016.
39. Zhang, Wei (1991). "Error Back Propagation with Minimum-Entropy Weights: A Technique for Better Generalization of 2-D Shift-Invariant NNs" (https://drive.google.com/file/d/0B65v6Wo67Tk5dkJTcEMtU2c5Znc/view?usp=sharing). *Proceedings of the International Joint Conference on Neural Networks*.
40. Zhang, Wei (1991). "Image processing of human corneal endothelium based on a learning network" (https://drive.google.com/file/d/0B65v6Wo67Tk5cm5DTlNGd0NPUmM/view?usp=sharing). *Applied Optics*. **30** (29): 4211–7. Bibcode:1991ApOpt..30.4211Z (https://ui.adsabs.harvard.edu/abs/1991ApOpt..30.4211Z). doi:10.1364/AO.30.004211 (https://doi.org/10.1364%2FAO.30.004211). PMID 20706526 (https://pubmed.ncbi.nlm.nih.gov/20706526).
41. Zhang, Wei (1994). "Computerized detection of clustered microcalcifications in digital mammograms using a shift-invariant artificial neural network" (https://drive.google.com/file/d/0B65v6Wo67Tk5Ml9qeW5nQ3poVTQ/view?usp=sharing). *Medical Physics*. **21** (4): 517–24. Bibcode:1994MedPh..21..517Z (https://ui.adsabs.harvard.edu/abs/1994MedPh..21..517Z). doi:10.1118/1.597177 (https://doi.org/10.1118%2F1.597177). PMID 8058017 (https://pubmed.ncbi.nlm.nih.gov/8058017).
42. Daniel Graupe, Ruey Wen Liu, George S Moschytz."Applications of neural networks to medical signal processing (https://www.researchgate.net/profile/Daniel_Graupe2/publication/241130197_Applications_of_signal_and_image_processing_to_medicine/links/575eef7e08aec91374b42bd2.pdf)". In Proc. 27th IEEE Decision and Control Conf., pp. 343–347, 1988.
43. Daniel Graupe, Boris Vern, G. Gruener, Aaron Field, and Qiu Huang. "Decomposition of surface EMG signals into single fiber action potentials by means of neural network (https://ieeexplore.ieee.org/abstract/document/100522/)". Proc. IEEE International Symp. on Circuits and Systems, pp. 1008–1011, 1989.
44. Qiu Huang, Daniel Graupe, Yi Fang Huang, Ruey Wen Liu."Identification of firing patterns of neuronal signals (http://www.academia.edu/download/42092095/graupe_huang_q_huang_yf_liu_rw_1989.pdf)." In Proc. 28th IEEE Decision and Control Conf., pp. 266–271, 1989. https://ieeexplore.ieee.org/document/70115
45. Behnke, Sven (2003). *Hierarchical Neural Networks for Image Interpretation* (https://www.ais.uni-bonn.de/books/LNCS2766.pdf) (PDF). Lecture Notes in Computer Science. Vol. 2766. Springer. doi:10.1007/b11963 (https://doi.org/10.1007%2Fb11963). ISBN 978-3-540-40722-5. S2CID 1304548 (https://api.semanticscholar.org/CorpusID:1304548).

46. Oh, KS; Jung, K (2004). "GPU implementation of neural networks". *Pattern Recognition*. **37** (6): 1311–1314. Bibcode:2004PatRe..37.1311O (https://ui.adsabs.harvard.edu/abs/2004PatRe..37.1311O). doi:10.1016/j.patcog.2004.01.013 (https://doi.org/10.1016%2Fj.patcog.2004.01.013).
47. Dave Steinkraus; Patrice Simard; Ian Buck (2005). "Using GPUs for Machine Learning Algorithms" (https://www.computer.org/csdl/proceedings-article/icdar/2005/24201115/12OmNylKAVX). *12th International Conference on Document Analysis and Recognition (ICDAR 2005)*. pp. 1115–1119. doi:10.1109/ICDAR.2005.251 (https://doi.org/10.1109%2FICDAR.2005.251).
48. Kumar Chellapilla; Sid Puri; Patrice Simard (2006). "High Performance Convolutional Neural Networks for Document Processing" (https://hal.inria.fr/inria-00112631/document). In Lorette, Guy (ed.). *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft.
49. Hinton, GE; Osindero, S; Teh, YW (Jul 2006). "A fast learning algorithm for deep belief nets". *Neural Computation*. **18** (7): 1527–54. CiteSeerX 10.1.1.76.1541 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.76.1541). doi:10.1162/neco.2006.18.7.1527 (https://doi.org/10.1162%2Fneco.2006.18.7.1527). PMID 16764513 (https://pubmed.ncbi.nlm.nih.gov/16764513). S2CID 2309950 (https://api.semanticscholar.org/CorpusID:2309950).
50. Bengio, Yoshua; Lamblin, Pascal; Popovici, Dan; Larochelle, Hugo (2007). "Greedy Layer-Wise Training of Deep Networks" (https://proceedings.neurips.cc/paper/2006/file/5da713a690c067105aeb2fae32403405-Paper.pdf) (PDF). *Advances in Neural Information Processing Systems*: 153–160.
51. Ranzato, MarcAurelio; Poultney, Christopher; Chopra, Sumit; LeCun, Yann (2007). "Efficient Learning of Sparse Representations with an Energy-Based Model" (http://yann.lecun.com/exdb/publis/pdf/ranzato-06.pdf) (PDF). *Advances in Neural Information Processing Systems*.
52. Raina, R; Madhavan, A; Ng, Andrew (2009). "Large-scale deep unsupervised learning using graphics processors" (http://robotics.stanford.edu/~ang/papers/icml09-LargeScaleUnsupervisedDeepLearningGPU.pdf) (PDF). *ICML*: 873–880.
53. Ciresan, Dan; Meier, Ueli; Gambardella, Luca; Schmidhuber, Jürgen (2010). "Deep big simple neural nets for handwritten digit recognition". *Neural Computation*. **22** (12): 3207–3220. arXiv:1003.0358 (https://arxiv.org/abs/1003.0358). doi:10.1162/NECO_a_00052 (https://doi.org/10.1162%2FNECO_a_00052). PMID 20858131 (https://pubmed.ncbi.nlm.nih.gov/20858131). S2CID 1918673 (https://api.semanticscholar.org/CorpusID:1918673).
54. "IJCNN 2011 Competition result table" (https://benchmark.ini.rub.de/gtsrb_results.html). *OFFICIAL IJCNN2011 COMPETITION*. 2010. Retrieved 2019-01-14.
55. Schmidhuber, Jürgen (17 March 2017). "History of computer vision contests won by deep CNNs on GPU" (https://people.idsia.ch/~juergen/computer-vision-contests-won-by-gpu-cnns.html). Retrieved 14 January 2019.
56. Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E. (2017-05-24). "ImageNet classification with deep convolutional neural networks" (https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf) (PDF). *Communications of the ACM*. **60** (6): 84–90. doi:10.1145/3065386 (https://doi.org/10.1145%2F3065386). ISSN 0001-0782 (https://www.worldcat.org/issn/0001-0782). S2CID 195908774 (https://api.semanticscholar.org/CorpusID:195908774).
57. He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing; Sun, Jian (2016). "Deep Residual Learning for Image Recognition" (https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf) (PDF). *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*: 770–778. arXiv:1512.03385 (https://arxiv.org/abs/1512.03385). doi:10.1109/CVPR.2016.90 (https://doi.org/10.1109%2FCVPR.2016.90). ISBN 978-1-4673-8851-1. S2CID 206594692 (https://api.semanticscholar.org/CorpusID:206594692).

58. Viebke, Andre; Pllana, Sabri (2015). "The Potential of the Intel (R) Xeon Phi for Supervised Deep Learning" (http://lnu.diva-portal.org/smash/record.jsf?pid=diva2%3A877421&dswid=4277). *2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cyberspace Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*. *IEEE Xplore*. IEEE 2015. pp. 758–765. doi:10.1109/HPCC-CSS-ICESS.2015.45 (https://doi.org/10.1109%2FHPCC-CSS-ICESS.2015.45). ISBN 978-1-4799-8937-9. S2CID 15411954 (https://api.semanticscholar.org/CorpusID:15411954).
59. Viebke, Andre; Memeti, Suejb; Pllana, Sabri; Abraham, Ajith (2019). "CHAOS: a parallelization scheme for training convolutional neural networks on Intel Xeon Phi". *The Journal of Supercomputing*. **75** (1): 197–227. arXiv:1702.07908 (https://arxiv.org/abs/1702.07908). doi:10.1007/s11227-017-1994-x (https://doi.org/10.1007%2Fs11227-017-1994-x). S2CID 14135321 (https://api.semanticscholar.org/CorpusID:14135321).
60. Hinton, Geoffrey (2012). "ImageNet Classification with Deep Convolutional Neural Networks" (https://dl.acm.org/doi/10.5555/2999134.2999257). *NIPS'12: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. **1**: 1097–1105 – via ACM.
61. Azulay, Aharon; Weiss, Yair (2019). "Why do deep convolutional networks generalize so poorly to small image transformations?" (https://jmlr.org/papers/v20/19-519.html). *Journal of Machine Learning Research*. **20** (184): 1–25. ISSN 1533-7928 (https://www.worldcat.org/issn/1533-7928).
62. Géron, Aurélien (2019). *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. Sebastopol, CA: O'Reilly Media. ISBN 978-1-492-03264-9., pp. 448
63. "CS231n Convolutional Neural Networks for Visual Recognition" (https://cs231n.github.io/convolutional-networks/). *cs231n.github.io*. Retrieved 2017-04-25.
64. Scherer, Dominik; Müller, Andreas C.; Behnke, Sven (2010). "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition" (http://ais.uni-bonn.de/papers/icann2010_maxpool.pdf) (PDF). *Artificial Neural Networks (ICANN), 20th International Conference on*. Thessaloniki, Greece: Springer. pp. 92–101.
65. Graham, Benjamin (2014-12-18). "Fractional Max-Pooling". arXiv:1412.6071 (https://arxiv.org/abs/1412.6071) [cs.CV (https://arxiv.org/archive/cs.CV)].
66. Springenberg, Jost Tobias; Dosovitskiy, Alexey; Brox, Thomas; Riedmiller, Martin (2014-12-21). "Striving for Simplicity: The All Convolutional Net". arXiv:1412.6806 (https://arxiv.org/abs/1412.6806) [cs.LG (https://arxiv.org/archive/cs.LG)].
67. Grel, Tomasz (2017-02-28). "Region of interest pooling explained" (https://deepsense.io/region-of-interest-pooling-explained/). *deepsense.io*. Retrieved 5 April 2017.
68. Girshick, Ross (2015-09-27). "Fast R-CNN". arXiv:1504.08083 (https://arxiv.org/abs/1504.08083) [cs.CV (https://arxiv.org/archive/cs.CV)].
69. Ma, Zhanyu; Chang, Dongliang; Xie, Jiyang; Ding, Yifeng; Wen, Shaoguo; Li, Xiaoxu; Si, Zhongwei; Guo, Jun (2019). "Fine-Grained Vehicle Classification With Channel Max Pooling Modified CNNs". *IEEE Transactions on Vehicular Technology*. Institute of Electrical and Electronics Engineers (IEEE). **68** (4): 3224–3233. doi:10.1109/tvt.2019.2899972 (https://doi.org/10.1109%2Ftvt.2019.2899972). ISSN 0018-9545 (https://www.worldcat.org/issn/0018-9545).
70. Romanuke, Vadim (2017). "Appropriate number and allocation of ReLUs in convolutional neural networks" (https://doi.org/10.20535%2F1810-0546.2017.1.88156). *Research Bulletin of NTUU "Kyiv Polytechnic Institute"*. **1**: 69–78. doi:10.20535/1810-0546.2017.1.88156 (https://doi.org/10.20535%2F1810-0546.2017.1.88156).

71. Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012). "Imagenet classification with deep convolutional neural networks" (https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf) (PDF). *Advances in Neural Information Processing Systems*. **1**: 1097–1105.
72. "6.3. Padding and Stride — Dive into Deep Learning 0.17.0 documentation" (https://d2l.ai/chapter_convolutional-neural-networks/padding-and-strides.html). *d2l.ai*. Retrieved 2021-08-12.
73. Deshpande, Adit. "The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)" (https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html). *adeshpande3.github.io*. Retrieved 2018-12-04.
74. Seo, Jae Duk (2018-03-12). "Understanding 2D Dilated Convolution Operation with Examples in Numpy and Tensorflow with…" (https://towardsdatascience.com/understanding-2d-dilated-convolution-operation-with-examples-in-numpy-and-tensorflow-with-d376b3972b25). *Medium*. Retrieved 2021-08-12.
75. Ribeiro,Schon, Antonio,Thomas (2021). "How Convolutional Neural Networks Deal with Aliasing". *IEEE International Conference on Acoustics, Speech and Signal Processing*: 2755–2759. arXiv:2102.07757 (https://arxiv.org/abs/2102.07757). doi:10.1109/ICASSP39728.2021.9414627 (https://doi.org/10.1109%2FICASSP39728.2021.9414627). ISBN 978-1-7281-7605-5. S2CID 231925012 (https://api.semanticscholar.org/CorpusID:231925012).
76. Myburgh, Johannes C.; Mouton, Coenraad; Davel, Marelie H. (2020). Gerber, Aurona (ed.). "Tracking Translation Invariance in CNNs" (https://link.springer.com/chapter/10.1007%2F978-3-030-66151-9_18). *Artificial Intelligence Research*. Communications in Computer and Information Science. Cham: Springer International Publishing. **1342**: 282–295. arXiv:2104.05997 (https://arxiv.org/abs/2104.05997). doi:10.1007/978-3-030-66151-9_18 (https://doi.org/10.1007%2F978-3-030-66151-9_18). ISBN 978-3-030-66151-9. S2CID 233219976 (https://api.semanticscholar.org/CorpusID:233219976).
77. Richard, Zhang (2019-04-25). *Making Convolutional Networks Shift-Invariant Again* (https://www.worldcat.org/oclc/1106340711). OCLC 1106340711 (https://www.worldcat.org/oclc/1106340711).
78. Jadeberg, Simonyan, Zisserman, Kavukcuoglu, Max, Karen, Andrew, Koray (2015). "Spatial Transformer Networks" (https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf) (PDF). *Advances in Neural Information Processing Systems*. **28** – via NIPS.
79. E, Sabour, Sara Frosst, Nicholas Hinton, Geoffrey (2017-10-26). *Dynamic Routing Between Capsules* (https://worldcat.org/oclc/1106278545). OCLC 1106278545 (https://www.worldcat.org/oclc/1106278545).
80. Matiz, Sergio; Barner, Kenneth E. (2019-06-01). "Inductive conformal predictor for convolutional neural networks: Applications to active learning for image classification" (https://www.sciencedirect.com/science/article/abs/pii/S003132031930055X). *Pattern Recognition*. **90**: 172–182. Bibcode:2019PatRe..90..172M (https://ui.adsabs.harvard.edu/abs/2019PatRe..90..172M). doi:10.1016/j.patcog.2019.01.035 (https://doi.org/10.1016%2Fj.patcog.2019.01.035). ISSN 0031-3203 (https://www.worldcat.org/issn/0031-3203). S2CID 127253432 (https://api.semanticscholar.org/CorpusID:127253432).

81. Wieslander, Håkan; Harrison, Philip J.; Skogberg, Gabriel; Jackson, Sonya; Fridén, Markus; Karlsson, Johan; Spjuth, Ola; Wählby, Carolina (February 2021). "Deep Learning With Conformal Prediction for Hierarchical Analysis of Large-Scale Whole-Slide Tissue Images" (https://ieeexplore.ieee.org/document/9103229). *IEEE Journal of Biomedical and Health Informatics*. **25** (2): 371–380. doi:10.1109/JBHI.2020.2996300 (https://doi.org/10.1109%2FJBHI.2020.2996300). ISSN 2168-2208 (https://www.worldcat.org/issn/2168-2208). PMID 32750907 (https://pubmed.ncbi.nlm.nih.gov/32750907). S2CID 219885788 (https://api.semanticscholar.org/CorpusID:219885788).

82. Srivastava, Nitish; C. Geoffrey Hinton; Alex Krizhevsky; Ilya Sutskever; Ruslan Salakhutdinov (2014). "Dropout: A Simple Way to Prevent Neural Networks from overfitting" (http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf) (PDF). *Journal of Machine Learning Research*. **15** (1): 1929–1958.

83. Carlos E. Perez. "A Pattern Language for Deep Learning" (http://www.deeplearningpatterns.com).

84. "Regularization of Neural Networks using DropConnect | ICML 2013 | JMLR W&CP" (http://proceedings.mlr.press/v28/wan13.html). *jmlr.org*: 1058–1066. 2013-02-13. Retrieved 2015-12-17.

85. Zeiler, Matthew D.; Fergus, Rob (2013-01-15). "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks". arXiv:1301.3557 (https://arxiv.org/abs/1301.3557) [cs.LG (https://arxiv.org/archive/cs.LG)].

86. Platt, John; Steinkraus, Dave; Simard, Patrice Y. (August 2003). "Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis – Microsoft Research" (https://www.microsoft.com/en-us/research/publication/best-practices-for-convolutional-neural-networks-applied-to-visual-document-analysis/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2F%3Fid%3D68920). *Microsoft Research*. Retrieved 2015-12-17.

87. Hinton, Geoffrey E.; Srivastava, Nitish; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan R. (2012). "Improving neural networks by preventing co-adaptation of feature detectors". arXiv:1207.0580 (https://arxiv.org/abs/1207.0580) [cs.NE (https://arxiv.org/archive/cs.NE)].

88. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting" (https://jmlr.org/papers/v15/srivastava14a.html). *jmlr.org*. Retrieved 2015-12-17.

89. Hinton, Geoffrey (1979). "Some demonstrations of the effects of structural descriptions in mental imagery". *Cognitive Science*. **3** (3): 231–250. doi:10.1016/s0364-0213(79)80008-7 (https://doi.org/10.1016%2Fs0364-0213%2879%2980008-7).

90. Rock, Irvin. "The frame of reference." The legacy of Solomon Asch: Essays in cognition and social psychology (1990): 243–268.

91. J. Hinton, Coursera lectures on Neural Networks, 2012, Url: https://www.coursera.org/learn/neural-networks Archived (https://web.archive.org/web/20161231174321/https://www.coursera.org/learn/neural-networks) 2016-12-31 at the Wayback Machine

92. Dave Gershgorn (18 June 2018). "The inside story of how AI got good enough to dominate Silicon Valley" (https://qz.com/1307091/the-inside-story-of-how-ai-got-good-enough-to-dominate-silicon-valley/). *Quartz*. Retrieved 5 October 2018.

93. Lawrence, Steve; C. Lee Giles; Ah Chung Tsoi; Andrew D. Back (1997). "Face Recognition: A Convolutional Neural Network Approach". *IEEE Transactions on Neural Networks*. **8** (1): 98–113. CiteSeerX 10.1.1.92.5813 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.92.5813). doi:10.1109/72.554195 (https://doi.org/10.1109%2F72.554195). PMID 18255614 (https://pubmed.ncbi.nlm.nih.gov/18255614).

94. "ImageNet Large Scale Visual Recognition Competition 2014 (ILSVRC2014)" (https://image-net.org/challenges/LSVRC/2014/results). Retrieved 30 January 2016.

95. Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott E.; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew (2015). "Going deeper with convolutions". *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015*. IEEE Computer Society. pp. 1–9. arXiv:1409.4842 (https://arxiv.org/abs/1409.4842). doi:10.1109/CVPR.2015.7298594 (https://doi.org/10.1109%2FCVPR.2015.7298594).

96. Russakovsky, Olga; Deng, Jia; Su, Hao; Krause, Jonathan; Satheesh, Sanjeev; Ma, Sean; Huang, Zhiheng; Karpathy, Andrej; Khosla, Aditya; Bernstein, Michael; Berg, Alexander C.; Fei-Fei, Li (2014). "Image *Net* Large Scale Visual Recognition Challenge". arXiv:1409.0575 (https://arxiv.org/abs/1409.0575) [cs.CV (https://arxiv.org/archive/cs.CV)].

97. "The Face Detection Algorithm Set To Revolutionize Image Search" (https://www.technologyreview.com/2015/02/16/169357/the-face-detection-algorithm-set-to-revolutionize-image-search/). *Technology Review*. February 16, 2015. Retrieved 27 October 2017.

98. Baccouche, Moez; Mamalet, Franck; Wolf, Christian; Garcia, Christophe; Baskurt, Atilla (2011-11-16). "Sequential Deep Learning for Human Action Recognition". In Salah, Albert Ali; Lepri, Bruno (eds.). *Human Behavior Unterstanding*. Lecture Notes in Computer Science. Vol. 7065. Springer Berlin Heidelberg. pp. 29–39. CiteSeerX 10.1.1.385.4740 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.385.4740). doi:10.1007/978-3-642-25446-8_4 (https://doi.org/10.1007%2F978-3-642-25446-8_4). ISBN 978-3-642-25445-1.

99. Ji, Shuiwang; Xu, Wei; Yang, Ming; Yu, Kai (2013-01-01). "3D Convolutional Neural Networks for Human Action Recognition". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **35** (1): 221–231. CiteSeerX 10.1.1.169.4046 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.169.4046). doi:10.1109/TPAMI.2012.59 (https://doi.org/10.1109%2FTPAMI.2012.59). ISSN 0162-8828 (https://www.worldcat.org/issn/0162-8828). PMID 22392705 (https://pubmed.ncbi.nlm.nih.gov/22392705). S2CID 1923924 (https://api.semanticscholar.org/CorpusID:1923924).

100. Huang, Jie; Zhou, Wengang; Zhang, Qilin; Li, Houqiang; Li, Weiping (2018). "Video-based Sign Language Recognition without Temporal Segmentation". arXiv:1801.10111 (https://arxiv.org/abs/1801.10111) [cs.CV (https://arxiv.org/archive/cs.CV)].

101. Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks (https://www.cv-foundation.org/openaccess/content_cvpr_2014/papers/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.pdf)." IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2014.

102. Simonyan, Karen; Zisserman, Andrew (2014). "Two-Stream Convolutional Networks for Action Recognition in Videos". arXiv:1406.2199 (https://arxiv.org/abs/1406.2199) [cs.CV (https://arxiv.org/archive/cs.CV)]. (2014).

103. Wang, Le; Duan, Xuhuan; Zhang, Qilin; Niu, Zhenxing; Hua, Gang; Zheng, Nanning (2018-05-22). "Segment-Tube: Spatio-Temporal Action Localization in Untrimmed Videos with Per-Frame Segmentation" (https://qilin-zhang.github.io/_pages/pdfs/Segment-Tube_Spatio-Temporal_Action_Localization_in_Untrimmed_Videos_with_Per-Frame_Segmentation.pdf) (PDF). *Sensors*. **18** (5): 1657. Bibcode:2018Senso..18.1657W (https://ui.adsabs.harvard.edu/abs/2018Senso..18.1657W). doi:10.3390/s18051657 (https://doi.org/10.3390%2Fs18051657). ISSN 1424-8220 (https://www.worldcat.org/issn/1424-8220). PMC 5982167 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5982167). PMID 29789447 (https://pubmed.ncbi.nlm.nih.gov/29789447).

104. Duan, Xuhuan; Wang, Le; Zhai, Changbo; Zheng, Nanning; Zhang, Qilin; Niu, Zhenxing; Hua, Gang (2018). "Joint Spatio-Temporal Action Localization in Untrimmed Videos with Per-Frame Segmentation". *2018 25th IEEE International Conference on Image Processing (ICIP)*. 25th IEEE International Conference on Image Processing (ICIP). pp. 918–922. doi:10.1109/icip.2018.8451692 (https://doi.org/10.1109%2Ficip.2018.8451692). ISBN 978-1-4799-7061-2.

105. Taylor, Graham W.; Fergus, Rob; LeCun, Yann; Bregler, Christoph (2010-01-01). *Convolutional Learning of Spatio-temporal Features* (https://dl.acm.org/doi/10.5555/1888212). Proceedings of the 11th European Conference on Computer Vision: Part VI. ECCV'10. Berlin, Heidelberg: Springer-Verlag. pp. 140–153. ISBN 978-3-642-15566-6.
106. Le, Q. V.; Zou, W. Y.; Yeung, S. Y.; Ng, A. Y. (2011-01-01). *Learning Hierarchical Invariant Spatio-temporal Features for Action Recognition with Independent Subspace Analysis*. *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '11. Washington, DC, USA: IEEE Computer Society. pp. 3361–3368. CiteSeerX 10.1.1.294.5948 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.294.5948). doi:10.1109/CVPR.2011.5995496 (https://doi.org/10.1109%2FCVPR.2011.5995496). ISBN 978-1-4577-0394-2. S2CID 6006618 (https://api.semanticscholar.org/CorpusID:6006618).
107. Grefenstette, Edward; Blunsom, Phil; de Freitas, Nando; Hermann, Karl Moritz (2014-04-29). "A Deep Architecture for Semantic Parsing". arXiv:1404.7296 (https://arxiv.org/abs/1404.7296) [cs.CL (https://arxiv.org/archive/cs.CL)].
108. Mesnil, Gregoire; Deng, Li; Gao, Jianfeng; He, Xiaodong; Shen, Yelong (April 2014). "Learning Semantic Representations Using Convolutional Neural Networks for Web Search – Microsoft Research" (https://www.microsoft.com/en-us/research/publication/learning-semantic-representations-using-convolutional-neural-networks-for-web-search/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2Fdefault.aspx%3Fid%3D214617). *Microsoft Research*. Retrieved 2015-12-17.
109. Kalchbrenner, Nal; Grefenstette, Edward; Blunsom, Phil (2014-04-08). "A Convolutional Neural Network for Modelling Sentences". arXiv:1404.2188 (https://arxiv.org/abs/1404.2188) [cs.CL (https://arxiv.org/archive/cs.CL)].
110. Kim, Yoon (2014-08-25). "Convolutional Neural Networks for Sentence Classification". arXiv:1408.5882 (https://arxiv.org/abs/1408.5882) [cs.CL (https://arxiv.org/archive/cs.CL)].
111. Collobert, Ronan, and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning (https://thetalkingmachines.com/sites/default/files/2018-12/unified_nlp.pdf)."Proceedings of the 25th international conference on Machine learning. ACM, 2008.
112. Collobert, Ronan; Weston, Jason; Bottou, Leon; Karlen, Michael; Kavukcuoglu, Koray; Kuksa, Pavel (2011-03-02). "Natural Language Processing (almost) from Scratch". arXiv:1103.0398 (https://arxiv.org/abs/1103.0398) [cs.LG (https://arxiv.org/archive/cs.LG)].
113. Yin, W; Kann, K; Yu, M; Schütze, H (2017-03-02). "Comparative study of CNN and RNN for natural language processing". arXiv:1702.01923 (https://arxiv.org/abs/1702.01923) [cs.LG (https://arxiv.org/archive/cs.LG)].
114. Bai, S.; Kolter, J.S.; Koltun, V. (2018). "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". arXiv:1803.01271 (https://arxiv.org/abs/1803.01271) [cs.LG (https://arxiv.org/archive/cs.LG)].
115. Gruber, N. (2021). "Detecting dynamics of action in text with a recurrent neural network" (https://www.semanticscholar.org/paper/Detecting-dynamics-of-action-in-text-with-a-neural-Gruber/cd6c9da2e8c52b043faf05ccc2511a07c54ead0c). *Neural Computing and Applications*. **33** (12): 15709–15718. doi:10.1007/S00521-021-06190-5 (https://doi.org/10.1007%2FS00521-021-06190-5). S2CID 236307579 (https://api.semanticscholar.org/CorpusID:236307579).
116. Haotian, J.; Zhong, Li; Qianxiao, Li (2021). "Approximation Theory of Convolutional Architectures for Time Series Modelling". *International Conference on Machine Learning*. arXiv:2107.09355 (https://arxiv.org/abs/2107.09355).

117. Ren, Hansheng; Xu, Bixiong; Wang, Yujing; Yi, Chao; Huang, Congrui; Kou, Xiaoyu; Xing, Tony; Yang, Mao; Tong, Jie; Zhang, Qi (2019). *Time-Series Anomaly Detection Service at Microsoft* | *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. arXiv:1906.03821 (https://arxiv.org/abs/1906.03821). doi:10.1145/3292500.3330680 (https://doi.org/10.1145%2F3292500.3330680). S2CID 182952311 (https://api.semanticscholar.org/CorpusID:182952311).
118. Wallach, Izhar; Dzamba, Michael; Heifets, Abraham (2015-10-09). "AtomNet: A Deep Convolutional Neural Network for Bioactivity Prediction in Structure-based Drug Discovery". arXiv:1510.02855 (https://arxiv.org/abs/1510.02855) [cs.LG (https://arxiv.org/archive/cs.LG)].
119. Yosinski, Jason; Clune, Jeff; Nguyen, Anh; Fuchs, Thomas; Lipson, Hod (2015-06-22). "Understanding Neural Networks Through Deep Visualization". arXiv:1506.06579 (https://arxiv.org/abs/1506.06579) [cs.CV (https://arxiv.org/archive/cs.CV)].
120. "Toronto startup has a faster way to discover effective medicines" (https://www.theglobeandmail.com/report-on-business/small-business/starting-out/toronto-startup-has-a-faster-way-to-discover-effective-medicines/article25660419/). *The Globe and Mail*. Retrieved 2015-11-09.
121. "Startup Harnesses Supercomputers to Seek Cures" (https://www.kqed.org/futureofyou/3461/startup-harnesses-supercomputers-to-seek-cures). *KQED Future of You*. 2015-05-27. Retrieved 2015-11-09.
122. Tim Pyrkov; Konstantin Slipensky; Mikhail Barg; Alexey Kondrashin; Boris Zhurov; Alexander Zenin; Mikhail Pyatnitskiy; Leonid Menshikov; Sergei Markov; Peter O. Fedichev (2018). "Extracting biological age from biomedical data via deep learning: too much of a good thing?" (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5980076). *Scientific Reports*. **8** (1): 5210. Bibcode:2018NatSR...8.5210P (https://ui.adsabs.harvard.edu/abs/2018NatSR...8.5210P). doi:10.1038/s41598-018-23534-9 (https://doi.org/10.1038%2Fs41598-018-23534-9). PMC 5980076 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5980076). PMID 29581467 (https://pubmed.ncbi.nlm.nih.gov/29581467).
123. Chellapilla, K; Fogel, DB (1999). "Evolving neural networks to play checkers without relying on expert knowledge". *IEEE Trans Neural Netw*. **10** (6): 1382–91. doi:10.1109/72.809083 (https://doi.org/10.1109%2F72.809083). PMID 18252639 (https://pubmed.ncbi.nlm.nih.gov/18252639).
124. Chellapilla, K.; Fogel, D.B. (2001). "Evolving an expert checkers playing program without using human expertise". *IEEE Transactions on Evolutionary Computation*. **5** (4): 422–428. doi:10.1109/4235.942536 (https://doi.org/10.1109%2F4235.942536).
125. Fogel, David (2001). *Blondie24: Playing at the Edge of AI*. San Francisco, CA: Morgan Kaufmann. ISBN 978-1558607835.
126. Clark, Christopher; Storkey, Amos (2014). "Teaching Deep Convolutional Neural Networks to Play Go". arXiv:1412.3409 (https://arxiv.org/abs/1412.3409) [cs.AI (https://arxiv.org/archive/cs.AI)].
127. Maddison, Chris J.; Huang, Aja; Sutskever, Ilya; Silver, David (2014). "Move Evaluation in Go Using Deep Convolutional Neural Networks". arXiv:1412.6564 (https://arxiv.org/abs/1412.6564) [cs.LG (https://arxiv.org/archive/cs.LG)].
128. "AlphaGo – Google DeepMind" (https://web.archive.org/web/20160130230207/http://www.deepmind.com/alpha-go.html). Archived from the original (https://www.deepmind.com/alpha-go.html) on 30 January 2016. Retrieved 30 January 2016.
129. Bai, Shaojie; Kolter, J. Zico; Koltun, Vladlen (2018-04-19). "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling". arXiv:1803.01271 (https://arxiv.org/abs/1803.01271) [cs.LG (https://arxiv.org/archive/cs.LG)].
130. Yu, Fisher; Koltun, Vladlen (2016-04-30). "Multi-Scale Context Aggregation by Dilated Convolutions". arXiv:1511.07122 (https://arxiv.org/abs/1511.07122) [cs.CV (https://arxiv.org/archive/cs.CV)].

131. Borovykh, Anastasia; Bohte, Sander; Oosterlee, Cornelis W. (2018-09-17). "Conditional Time Series Forecasting with Convolutional Neural Networks". arXiv:1703.04691 (https://arxiv.org/abs/1703.04691) [stat.ML (https://arxiv.org/archive/stat.ML)].
132. Mittelman, Roni (2015-08-03). "Time-series modeling with undecimated fully convolutional neural networks". arXiv:1508.00317 (https://arxiv.org/abs/1508.00317) [stat.ML (https://arxiv.org/archive/stat.ML)].
133. Chen, Yitian; Kang, Yanfei; Chen, Yixiong; Wang, Zizhuo (2019-06-11). "Probabilistic Forecasting with Temporal Convolutional Neural Network". arXiv:1906.04397 (https://arxiv.org/abs/1906.04397) [stat.ML (https://arxiv.org/archive/stat.ML)].
134. Zhao, Bendong; Lu, Huanzhang; Chen, Shangfeng; Liu, Junliang; Wu, Dongya (2017-02-01). "Convolutional neural networks for time series classi". *Journal of Systems Engineering and Electronics*. **28** (1): 162–169. doi:10.21629/JSEE.2017.01.18 (https://doi.org/10.21629%2FJSEE.2017.01.18).
135. Petneházi, Gábor (2019-08-21). "QCNN: Quantile Convolutional Neural Network". arXiv:1908.07978 (https://arxiv.org/abs/1908.07978) [cs.LG (https://arxiv.org/archive/cs.LG)].
136. Hubert Mara (2019-06-07), *HeiCuBeDa Hilprecht – Heidelberg Cuneiform Benchmark Dataset for the Hilprecht Collection* (in German), heiDATA – institutional repository for research data of Heidelberg University, doi:10.11588/data/IE8CCN (https://doi.org/10.11588%2Fdata%2FIE8CCN)
137. Hubert Mara and Bartosz Bogacz (2019), "Breaking the Code on Broken Tablets: The Learning Challenge for Annotated Cuneiform Script in Normalized 2D and 3D Datasets", *Proceedings of the 15th International Conference on Document Analysis and Recognition (ICDAR)* (in German), Sydney, Australien, pp. 148–153, doi:10.1109/ICDAR.2019.00032 (https://doi.org/10.1109%2FICDAR.2019.00032), ISBN 978-1-7281-3014-9, S2CID 211026941 (https://api.semanticscholar.org/CorpusID:211026941)
138. Bogacz, Bartosz; Mara, Hubert (2020), "Period Classification of 3D Cuneiform Tablets with Geometric Neural Networks", *Proceedings of the 17th International Conference on Frontiers of Handwriting Recognition (ICFHR)*, Dortmund, Germany
139. Presentation of the ICFHR paper on Period Classification of 3D Cuneiform Tablets with Geometric Neural Networks (https://www.youtube.com/watch?v=-iFntE51HRw) on YouTube
140. Durjoy Sen Maitra; Ujjwal Bhattacharya; S.K. Parui, "CNN based common approach to handwritten character recognition of multiple scripts" (https://ieeexplore.ieee.org/document/7333916), in Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, vol., no., pp.1021–1025, 23–26 Aug. 2015
141. "NIPS 2017" (http://interpretable.ml/). *Interpretable ML Symposium*. 2017-10-20. Retrieved 2018-09-12.
142. Zang, Jinliang; Wang, Le; Liu, Ziyi; Zhang, Qilin; Hua, Gang; Zheng, Nanning (2018). "Attention-Based Temporal Weighted Convolutional Neural Network for Action Recognition". *IFIP Advances in Information and Communication Technology*. Cham: Springer International Publishing. pp. 97–108. arXiv:1803.07179 (https://arxiv.org/abs/1803.07179). doi:10.1007/978-3-319-92007-8_9 (https://doi.org/10.1007%2F978-3-319-92007-8_9). ISBN 978-3-319-92006-1. ISSN 1868-4238 (https://www.worldcat.org/issn/1868-4238). S2CID 4058889 (https://api.semanticscholar.org/CorpusID:4058889).

143. Wang, Le; Zang, Jinliang; Zhang, Qilin; Niu, Zhenxing; Hua, Gang; Zheng, Nanning (2018-06-21). "Action Recognition by an Attention-Aware Temporal Weighted Convolutional Neural Network" (https://qilin-zhang.github.io/_pages/pdfs/sensors-18-01979-Action_Recognition_by_an_Attention-Aware_Temporal_Weighted_Convolutional_Neural_Network.pdf) (PDF). *Sensors*. **18** (7): 1979. Bibcode:2018Senso..18.1979W (https://ui.adsabs.harvard.edu/abs/2018Senso..18.1979W). doi:10.3390/s18071979 (https://doi.org/10.3390%2Fs18071979). ISSN 1424-8220 (https://www.worldcat.org/issn/1424-8220). PMC 6069475 (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6069475). PMID 29933555 (https://pubmed.ncbi.nlm.nih.gov/29933555).

144. Ong, Hao Yi; Chavez, Kevin; Hong, Augustus (2015-08-18). "Distributed Deep Q-Learning". arXiv:1508.04186v2 (https://arxiv.org/abs/1508.04186v2) [cs.LG (https://arxiv.org/archive/cs.LG)].

145. Mnih, Volodymyr; et al. (2015). "Human-level control through deep reinforcement learning". *Nature*. **518** (7540): 529–533. Bibcode:2015Natur.518..529M (https://ui.adsabs.harvard.edu/abs/2015Natur.518..529M). doi:10.1038/nature14236 (https://doi.org/10.1038%2Fnature14236). PMID 25719670 (https://pubmed.ncbi.nlm.nih.gov/25719670). S2CID 205242740 (https://api.semanticscholar.org/CorpusID:205242740).

146. Sun, R.; Sessions, C. (June 2000). "Self-segmentation of sequences: automatic formation of hierarchies of sequential behaviors". *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*. **30** (3): 403–418. CiteSeerX 10.1.1.11.226 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.11.226). doi:10.1109/3477.846230 (https://doi.org/10.1109%2F3477.846230). ISSN 1083-4419 (https://www.worldcat.org/issn/1083-4419). PMID 18252373 (https://pubmed.ncbi.nlm.nih.gov/18252373).

147. "Convolutional Deep Belief Networks on CIFAR-10" (http://www.cs.toronto.edu/~kriz/conv-cifar10-aug2010.pdf) (PDF).

148. Lee, Honglak; Grosse, Roger; Ranganath, Rajesh; Ng, Andrew Y. (1 January 2009). *Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations*. *Proceedings of the 26th Annual International Conference on Machine Learning – ICML '09*. ACM. pp. 609–616. CiteSeerX 10.1.1.149.6800 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.149.6800). doi:10.1145/1553374.1553453 (https://doi.org/10.1145%2F1553374.1553453). ISBN 9781605585161. S2CID 12008458 (https://api.semanticscholar.org/CorpusID:12008458).

149. Cade Metz (May 18, 2016). "Google Built Its Very Own Chips to Power Its AI Bots" (https://www.wired.com/2016/05/google-tpu-custom-chips/). *Wired*.

External links

- CS231n: Convolutional Neural Networks for Visual Recognition (https://cs231n.github.io/) — Andrej Karpathy's Stanford computer science course on CNNs in computer vision
- An Intuitive Explanation of Convolutional Neural Networks (https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/) — A beginner level introduction to what Convolutional Neural Networks are and how they work
- Convolutional Neural Networks for Image Classification (https://www.completegate.com/2017022864/blog/deep-machine-learning-images-lenet-alexnet-cnn/all-pages) — Literature Survey