Convolutional Neural Networks:

Fischer, Manfred M.

DOI:
10.57938/2076fa6b-8225-433c-b3e6-c96de2e46735
10.57938/2076fa6b-8225-433c-b3e6-c96de2e46735

Published: 01/01/2025

Document Version
Publisher's PDF, also known as Version of record

Convolutional Neural Networks: Key Components and Architectures

Manfred M. Fischer

Department of Socioeconomics, Vienna University of Economics and Business, Welthandelsplatz 1, 1020 Vienna, Austria, e-mail: manfred.fischer@wu.ac.at

Abstract. Convolutional Neural Networks (CNNs) are a specialized class of deep neural networks tailored for processing high-dimensional data, excelling in tasks like image classification, object detection, and facial recognition. Their architecture is built on convolutional layers interspersed with pooling layers, which efficiently extract hierarchical features while reducing computational complexity. Convolutional layers utilize filters to detect patterns such as edges or textures, employing shared weights to enhance translational invariance and reduce the number of parameters. Pooling layers further simplify feature maps by downsampling, preserving critical information while minimizing spatial dimensions. Prominent architectures, including LeNet, AlexNet, VGGNet, and ResNet, have set benchmarks in image recognition and inspired advancements in deep learning. The careful tuning of hyperparameters, such as filter size, stride, and padding, plays a pivotal role in optimizing performance, balancing accuracy with computational efficiency. CNNs continue to drive innovation, expanding their applications across diverse fields like natural language processing, speech recognition, and autonomous systems.

Index Terms: Deep convolutional neural networks, LeNet, AlexNet, VGGNet, ResNet

The architectural approach known as convolutional neural network (CNN) is a specialized type of deep neural networks designed to handle high-dimensional data, such as images and videos. CNNs are widely used in tasks such as image classification and segmentation (Krizhevsky, Sutskever, and Hinton 2012; Maggiori et al. 2017), object detection and facial recognition (Hinton et al. 2012; Sermanet et al. 2014; Simonyan and Zisserman 2015). CNNs leverage spatial hierarchies in data through learnable filters, enabling them to automatically extract features across multiple levels of abstraction without the need for manual feature engineering.

Image data is structured as two-dimensional arrays of pixels, often encompassing multiple channels. For instance, color images include pixel intensities in red, green, and blue (RGB) channels, each with its own intensity value, while grayscale images have only one channel. Some specialized images, such as satellite images, might have additional channels (e.g., infrared), while medical imaging modalities such as magnetic resonance imaging (MRI) scans involve three-dimensional grids of voxels.

One motivation for developing CNNs was the challenge posed by the high-dimensional nature of images, which makes standard fully connected networks computationally expensive due to the vast number of parameters required. The use of weight sharing, multiple layers, and hierarchical processing enables CNNs to efficiently analyze (Bishop and Bishop 2024, pp. 287-321). Beyond image-related tasks, CNNs are also applied in fields such as natural language processing, time series analysis, and speech recognition, as well as in technologies like autonomous mobile robots and self-driving cars (LeCun, Bengio, and Hinton 2015).

CNN Architecture

A convolutional neural network consists of an input layer, multiple hidden layers, and an output layer. The core architecture of a CNN consists of three primary components: convolutional layers, pooling layers, and fully connected layers. Variations in the design and training CNNs arise from how these layers are arranged and interrelated.

Convolutional Layers

Convolutional layers use convolutional operations to apply filters (or kernels) to the input data, extracting spatial features such as edges, textures, or more abstract patterns at deeper levels. Filters operate using shared weights by reducing the number of parameters compared to fully connected networks. This weight sharing exploits spatial locality and ensures translational invariance.

A filter is a small matrix of weights used to extract features from localized regions (patches) of the input image, emphasizing the property of locality. A feature map is the output generated when a filter is applied to the input layer, highlighting regions where specific patterns are detected. Neurons within a feature map share the same weight values, enabling parameter sharing, a defining characteristic of convolutional layers.

Each neuron in a convolutional layer is connected to a small patch of the image input, known as its receptive field. The filter size, often a 3x3 or 5x5, is much smaller than the input image dimensions. The convolution operation involves computing the dot product between the filter and the input data, followed by the application of a non-linear activation function, such as Rectified Linear Unit (ReLU). The RELU function has become the standard for introducing non-linearity, as it avoids vanishing gradient problems common in earlier activation functions like sigmoid or tanh (Glorot, Bordes and Bengio 2011; Nair and Hinton 2010).

Filters represent the network's weights, which are learned through backpropagation training. Optimizing algorithms like stochastic gradient descent (SGD) adjust these weights to minimize a loss function, as described by LeCun et al. (1998) and Schmidhuber

(2014). Leveraging the parallel processing capabilities of graphics processing units (GPUs) enhances the efficiency of convolution operations.

As filters slide across the input, they produce feature maps. Each filter generates a distinct map, which is essentially a two-dimensional matrix of activation values reflecting how well the filter matches different regions of the image. A complete convolutional layer is composed of several feature maps.

A key advantage of convolutional layers is parameter sharing, which drastically reduces the number of learnable parameters. This architecture reduces the risk of overfitting, narrowing the gap between test and training errors (Hinton et al. 2012; LeCun et al. 1998). When processing multi-channel images, filters span all channels of the input. For example, when analyzing a color image, a neuron considers a patch across all RGB channels simultaneously, enabling the network to discern complex patterns such as color textures.

Convolutional Hyperparameters

Convolutional hyperparameters control the behavior of convolutional layers. Critical hyperparameters include depth, stride, and padding. Depth refers to the number of filters, which determines the number of feature maps generated. Reducing depth significantly minimizes the total number of neurons but also reduces the network's pattern recognition capabilities.

Stride and padding determine how the filter is applied to an input, affecting the output size and how features are extracted. Stride controls the step size of the filter (kernel) during convolution, influencing the resolution of feature maps. A stride of one results in overlapping receptive fields, while a larger stride produces smaller output dimensions. Padding ensures that feature map dimensions remain consistent or reduced proportionally. Zero-padding is widely used to preserve spatial dimensions, and prevent excessive reduction in feature map size (O'Shea and Nash 2015).

Pooling Layers

Pooling layers, also known as downsampling layers, reduce the spatial resolution of feature maps by summarizing information within local regions, thereby reducing computational complexity and introducing invariance to small translations (Hinton et al. 2012). Pooling operates on small receptive fields, typically 2x2 or 3x3, and is applied independently to each channel of the feature map.

Common pooling techniques include average-pooling and max-pooling. Average-pooling computes the mean of the values within the receptive field and is often used for tasks requiring fine-grained representations. Max-pooling, the most widely used technique,

retains the maximum value in the pooling window, emphasizing prominent features while reducing computational complexity and overfitting (Ciresan et al. 2011).

Fully Connected Layers

CNNs conclude with fully connected layers. These layers aggregate the extracted features and perform the final classification or regression task. They connect every neuron in one layer to every neuron in the next, enabling complex decision boundaries. In classification tasks, these layers predict class probabilities using a softmax activation function.

Prominent Architectures

CNNs, the first deep neural networks with three or more parameter layers, have demonstrated exceptional performance in fields like image classification (Dodge and Karam 2017). An early example, LeNet (LeCun et al. 1998), was designed for handwritten digit classification. Its simple structure included two convolutional layers (with six and 26 filters, respectively, each with a 5x5 filter size), followed by 2x2 pooling layers and three fully connected layers. This architecture showcased the power of CNNs for image recognition and laid the groundwork for advancements in deep learning.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), held annually from 2010 to 2017, significantly advanced CNN research. Based on a subset of the ImageNet dataset (Deng et al. 2009), the challenge involved recognizing and classifying objects and scenes across 1,000 categories, with around 1.2 million RGB training images and 50,000 validation images.

In 2012, AlexNet (Krizhevsky, Sutskever, and Hinton 2012) won the challenge by a large margin, marking a paradigm shift in computer vision. Its success was attributed to its deeper and more complex architecture. AlexNet is a deep neural network architecture with approximately 60 million parameters and 650,000 neurons, five convolutional layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1,000-way softmax. The use of multiple layers allows the network to learn complex hierarchical representations of the input images that are critical for accurate classification. AlexNet was among the first to leverage GPUs for efficient training.

The architecture revolutionized deep learning by using, first, parallelized matrix operations making GPUs essential for high-throughput computations, second, ReLU activations to enhance training and mitigate the vanishing gradient problem, third, data augmentation techniques to artificially increase the size of the training data set, and, finally, a new regularization called dropout to mitigate overfitting. Dropout randomly deactivates a fraction of neurons during training, preventing co-adaptation of features (Srivastava et al. 2014; Hinton et al. 2012).

VGGNet (Simonyan and Zisserman 2015), developed by the Visual Geometry Group at the University of Oxford, deepened CNN architectures with smaller receptive fields. Earlier CNNs typically had fewer convolutional layers as they had larger receptive fields. VGGNet comes in configurations with different depths, known as VGG-16 and VGG-19. The first consists of 16 weight layers (13 convolutional and three fully connected layers), while the second is deeper, with 19 learnable layers (16 convolutional and three fully connected layers).

VGGNet relies on some simple design principles resulting in a relatively uniform architecture that minimizes the number of hyperparameter choices to be made. The network takes input images of size 224 (height) x 224 (width) x 3 (color channels), and ends with fully connected layers for classification, employing softmax activation to generate class probabilities. All convolutional layers have filters of size 3x3 with a stride one, same padding, and a ReLU function while all max-pooling operations use filter size 2x2 and stride two, and thereby downsampling the number of neurons by a factor of four. The uniform design of VGGNet minimized hyperparameter tuning and simplified the architecture while achieving state-of-the-art performance on the ImageNet Challenge data set. Its principles – including the use of small filters – influenced subsequent CNN architectures.

Despite VGGNet's success, training deeper networks proved challenging due to issues like vanishing and shattered gradients. Batch normalization, along with careful initialization of the weights and biases, can help to address the notorious problem of vanishing gradients in deep networks, but not that one of shattered gradients (Balduzzi et al. 2017). ReNet (He et al. 2015), developed by Microsoft Research, addressed this with residual connections. Residual connections are a particular form of skip-layer connections. These skip-layer connections allow networks to bypass one or more layers, enabling information to flow effectively across layers, when using identity mappings as skip connections (He et al. 2016).

ReNet introduced residual learning, where the network learns residual functions rather than directly approximating the desired mapping function. A residual function is defined as the difference between input and output of a block. The architecture is built on blocks with two convolutional layers and identity shortcut connections. Inspired by the design philosophy of VGGNets, ResNet uses 3x3 filters and follows two rules: layers producing the same output feature size use the same number of filters, and halving the feature map size doubles the number of filters. Downsampling is performed using convolutional layers with a stride of two. The network ends with a global average-pooling layer and a 1,000-way fully connected layer with softmax activations.

ResNet variants, such as ResNet-18, ResNet-34, ResNet-50, ResNet-101 and ResNet-152, demonstrate improved performance on complex tasks. The architecture's ability to

train very deep networks has enabled applications beyond pattern recognition, including natural language processing and time series forecasting.

Closing Remarks

CNNs excel at feature extraction, hierarchical representation, and adaptability, making them versatile across domains. In addition to computer vision, CNNs are applied in:

Natural Language Processing: Tasks like sentiment analysis and language translation,

Speech Recognition: Feature extraction from audio signals,

Autonomous Vehicles: Real-time object detection,

Medical Diagnostics: Magnetic resonance imaging, image segmentation and registration,

Predictive Maintenance: Anomaly detection in manufacturing.

The success of CNNs is rooted in their innovative architectures, regularization methods, and training techniques. Advances like residual networks have made CNNs more robust and computationally efficient. These developments provide a strong foundation for exploring new applications in robotics, urban planning, and environmental monitoring, enabling transformative impacts across industries.

References

Bishop CM and Bishop H (2024): *Deep Learning: Foundations and Concepts*. Cham: Springer

Balduzzi D, Frean M, Leary L, Lewis JP, Wan-Duo Ma K and McWilliams B (2017): The shattered gradients problem: If resnets are the answer, then what is the question? *Proceedings of the 27th International Conference on Machine Learning*, pp. 342-350

Ciresan DC, Meier U, Masci J, Gambardella LM and Schmidhuber J (2011): Flexible, high performance convolutional neural networks for image classification. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pp. 1237-1242

Deng J, Dong W, Socher R, Li L-J, Li K and Fei-Fei L (2009): ImageNet: A large-scale hierarchical image database. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 248-255

Dodge S and Karam L (2017): A study and comparison of human and deep learning recognition performance under visual distortions. arXiv: 1705.02498v1 [cs.CV]

Glorot X, Bordes A and Bengio Y (2011): Deep sparse rectifier neural networks. *Proceedings of Machine Learning Research* 15, 315-323

He K, Zhang X, Ren S and Sun J (2016): Identity mapping in deep residual networks. arXiv: 1603.05027v3 [cs.CV]

He K, Zhang X, Ren S and Sun J (2015): Deep residual learning for image recognition. arXiv: 1512.03385v1 [cs.CV]

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I and Salakhutdinov RR (2012): Improving neural networks by preventing co-adaptation of feature detectors. arXiv: 1207.0580v1[cs.NE]

Krizhevsky A, Sutskever I and Hinton GE (2012): ImageNet classification with deep convolutional neural networks. *Communications of the ACM* 60, 84-90

LeCun Y, Bengio Y and Hinton G (2015): Deep learning, *Nature* 512, 436-444

LeCun Y, Bottou L, Bengio Y and Haffner P (1998): Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, pp. 2278-2324

Maggiori E, Tarabalka Y, Charpiat G and Alliez P (2017): Convolutional neural networks for large-scale remote-sensing image classification*, IEEE Transactions on Geoscience and Remote Sensing* 55(2), 645-657

Nair N and Hinton GE (2010): Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*, pp. 807-814

O'Shea K and Nash R (2015): An introduction to convolutional neural networks. arXiv: 1511.0845802 [cs.NE]

Schmidhuber J (2014): Deep learning in neural networks: An overview, *Neural Networks* 61, 85-117

Sermanet P, Eigen S, Zhang X, Mathieu M, Fergus R and LeCun Y (2014): OverFeat: Integrated recognition, localization and detection using convolutional networks. arXiv: 1312.6229v4 [cs.CV]

Simonyan K and Zisserman A (2015): Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556v6 [cs.CV]

Srivastava N, Hinton G, Krizhevsky A, Sustskever I and Slakhutdinov R (2014): Dropout: A simple way to prevent neural networks from overfitting, *Journal of Machine Learning Research* 15, 1929-1958