WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

A Systematic Review of Convolutional Neural Network Architectures

[1]Sri Narayana Tejaji

[1]St Paul's English School, J.P Nagar 2nd Phase, Bangalore – 560078, Karnataka, India

A B S T R A C T

Convolutional Neural Networks (CNNs) have revolutionized the field of deep learning and computer vision. This paper explores the fundamental architecture of CNNs, detailing each component, its role, and how CNNs have evolved over time. We also explore how CNNs are used in various real-world applications such as image classification, object detection, and medical imaging. The paper concludes by discussing future trends and challenges in CNN development.

1. Introduction

Deep learning models have demonstrated remarkable performance in complex pattern recognition tasks, with CNNs at the forefront of visual data analysis. Originally inspired by the human visual cortex, CNNs automatically and adaptively learn spatial hierarchies of features from input images. Since the landmark LeNet-5 architecture, CNNs have evolved into powerful tools capable of outperforming humans on some vision tasks.

2. CNN Architecture Overview

CNN architecture consists of several key layers:

2.1 Input Layer

The input layer receives raw pixel values of an image. The input dimensions are typically height × width × channels, e.g., 224×224×3 for a color image.

2.2 Conventional Layer

This is the core building block of a CNN. It applies a set of learnable filters (kernels) that slide across the input to produce feature maps. Each filter captures a specific kind of feature, like edges, textures, or shapes.

Equation:

$$y(i,j) = \sum_m \sum_n x(i+m, j+n) \cdot w(m,n) \quad (1)$$

where x is the input, w is the kernel, and y is the output.

2.3 Activation Layer (ReLU)

The Rectified Linear Unit (ReLU) introduces non-linearity by applying f(x) = max(0, x). This helps CNNs learn complex patterns.

2.4 Pooling Layer

Pooling layers (usually Max Pooling) reduce the spatial size of feature maps and computation, helping with overfitting and making the model invariant to small translations.

Common pooling size: 2×2 with a stride of 2

2.5 Fully Connected Layer (FC)

After several convolution and pooling layers, the high-level reasoning is done via fully connected layers. These act like traditional neural networks and generate the final output (e.g., classification scores).

2.6 Output Layer

For classification tasks, a softmax activation function is applied to produce class probabilities.

3. Evolution of CNN Architectures

CNNs have advanced dramatically. Some key architectures include:

- LeNet-5 (1998) – Introduced by Yann LeCun for digit recognition.
- AlexNet (2012) – Won the ImageNet competition; used ReLU and GPU training.
- VGGNet (2014) – Used 3×3 filters; deeper but simpler architecture.
- GoogLeNet/Inception (2014) – Introduced Inception modules for efficiency.
- ResNet (2015) – Used skip connections (residuals) to train very deep networks.
- EfficientNet (2019) – Balances depth, width, and resolution using compound scaling.z

4. Applications of CNNs

CNNs are not just theory—they're transforming industries:

- Image Classification – Identifying objects in images (e.g., dogs vs. cats).
- Object Detection – Locating multiple objects using YOLO, SSD, Faster R-CNN.
- Medical Imaging – Detecting tumors, abnormalities in MRIs, X-rays.
- Autonomous Vehicles – Recognizing pedestrians, traffic signs, lanes.
- Facial Recognition – Powering social media, surveillance, and authentication.

5. Performance Metrics

CNN models are typically evaluated using:

- Accuracy – Percentage of correct predictions.
- Precision/Recall/F1 Score – For class imbalance.
- Confusion Matrix – Helps analyze errors.
- Training vs. Validation Loss – Monitors overfitting.

6. Results and Comparative Analysis

The following table summarizes top-1 and top-5 error rates on ImageNet and parameter counts for selected models.

Model	Year	Params (M)	Top-1 Err (%)	Top-5 Err (%)
LeNet-5	1998	0.06	—	—
AlexNet	2012	61	42.8	15.3
VGG-16	2014	138	28.5	9.6
GoogLeNet (Inception-V1)	2014	6.8	30.2	10.1
ResNet-50	2015	25.6	23.9	7.1
EfficientNet-B0	2019	5.3	24.0	7.8

7. Challenges in CNNs

- Computational Cost – Training large CNNs requires GPUs and time.

- Overfitting – Especially on small datasets.
- Interpretability – CNNs are often 'black boxes.'
- Adversarial Attacks – Small image perturbations can fool CNNs.

8. Future Directions

- Explainable AI (XAI) – Making CNNs more interpretable.
- Lightweight Models – CNNs for edge devices (e.g., MobileNet).
- Self-supervised Learning – Reducing need for labeled data.
- 3D CNNs – For volumetric data (e.g., in medicine and AR).

9. Conclusion

CNNs are the powerhouse behind modern computer vision. Their layered structure—designed to progressively capture complexity—has set the foundation for image-driven AI applications. As architectures evolve and become more efficient and interpretable, CNNs will likely continue to lead in AI research and deployment across sectors.

Acknowledgements

Appendix A - CNN Architecture Diagrams

A1. Conventional Layer Diagram

Imagine a small filter sliding over the image to detect features like edges.

Input Image (5x5):

[1 2 3 0 1]

[0 1 2 3 4]

[1 0 1 2 3]

[2 1 0 1 0]

[3 2 1 0 1]

Filter (3x3):

[1 0 -1]

[1 0 -1]

[1 0 -1]

Sliding filter across input → Output feature map (3x3)

A 2. Activation Layer (ReLU) Diagram

ReLU = max(0, x). So any negative number becomes zero, positives stay the same.

| Input feature map | After ReLU |

|---|---|

| -1 2 -3 | 0 2 0 |

| 4 -2 1 | 4 0 1 |

| -1 0 3 | 0 0 3 |

Pooling Layer Diagram (Max Pooling 2x2)

Pooling downsamples, keeping the max from each region.

Input feature map (4x4):

[1 3 2 4]

[5 6 7 8]

[4 3 2 1]

[1 2 3 4]

Pooling with 2x2 window, stride 2:

- Top-left block: max(1,3,5,6) = 6

- Top-right block: max(2,4,7,8) = 8

- Bottom-left block: max(4,3,1,2) = 4

- Bottom-right block: max(2,1,3,4) = 4

Output (2x2):

[6 8]

[4 4]

A3 Fully Connected Layer Diagram

Think of flattening the output feature maps into a vector and connecting to neurons like a classic NN.

Feature Maps (e.g., 2 maps of 2x2):

Map1: [6, 8]

[4, 4]

Map2: [1, 3]

[2, 5]

Flattened vector: [6,8,4,4,1,3,2,5]

Connected to fully connected layer neurons with weights and biases → Outputs

A 4 Overall CNN Flow Diagram

Input Image

↓

Convolution + ReLU

↓

Pooling

↓

Convolution + ReLU

↓

Pooling

↓

Flatten

↓

Fully Connected Layer(s)

↓

Output (Softmax for classification)

References

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE.

Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition.

Tan, M., & Le, Q.V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks.