Reports on Convolutional Neural Networks

1 Fundamentals of Convolutional Neural Networks

1.1 Convolution Operation

A Convolutional Neural Network (CNN) processes images using the **convolution operation**, which applies a small matrix called a **filter** or **kernel** across an input image to extract meaningful patterns such as edges, textures, and shapes.

Mathematically, for an input image I and kernel K, the convolution output S is:

$$S(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$

1.2 Filters, Feature Maps, and Stride

Each filter produces a **feature map** highlighting a specific visual pattern. The **stride** defines how far the filter moves at each step. Larger strides reduce spatial resolution but improve computational efficiency.

1.3 Padding

Padding controls output size:

- **Valid padding**: No padding, output shrinks.
- **Same padding**: Zero-padding preserves spatial dimensions.

1.4 Pooling Layers

Pooling reduces spatial dimensions while retaining important information.

- **Max Pooling**: Selects maximum value.
- **Average Pooling**: Computes average value.

1.5 Convolutions over Volumes

CNNs handle color images by convolving over volumes (Height $\times$ Width $\times$ Channels). Each filter spans all input channels, producing a single feature map.

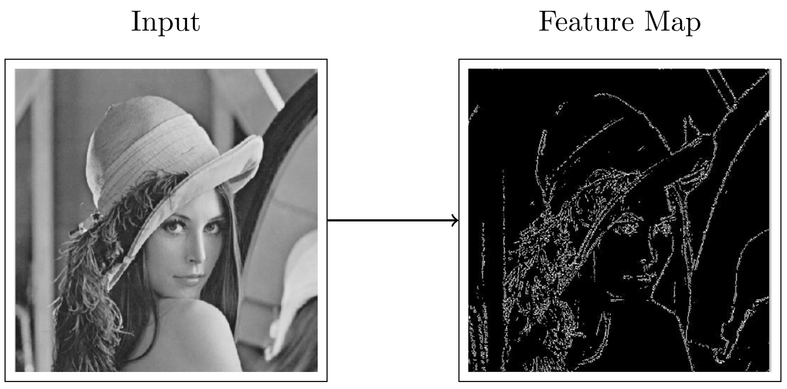

Figure 1: Basic convolution process

2 CNN Architectures and Evolution

2.1 LeNet-5

LeNet-5 pioneered CNNs for handwritten digit recognition, introducing convolution, pooling, and fully connected layers.

2.2 AlexNet

AlexNet significantly improved image classification accuracy by using ReLU activations, dropout, and GPU acceleration, sparking modern deep learning adoption.

2.3 Inception (GoogLeNet)

Inception networks introduced **inception modules**, which apply multiple filter sizes in parallel, capturing multi-scale features efficiently. The use of **1×1 convolutions** reduces computation.

2.4 ResNet

ResNet addressed the **degradation problem** by introducing **skip connections**:

$$y = F(x) + x$$

This allows gradients to flow more easily during training, enabling very deep networks.

2.5 MobileNet

MobileNet uses **depthwise separable convolutions**, splitting convolution into:

- Depthwise convolution
- Pointwise (1×1) convolution

This significantly reduces computational cost.

2.6 EfficientNet

EfficientNet scales network depth, width, and resolution uniformly using compound scaling, achieving high accuracy with fewer parameters.

3 Advanced Training Techniques for CNNs

3.1 Data Augmentation

Data augmentation artificially expands training data using transformations such as rotation, flipping, cropping, and color jittering, improving generalization.

3.2 Transfer Learning and Fine-Tuning

Transfer learning leverages pretrained models. Early layers are often frozen, while later layers are fine-tuned to adapt to the target task.

3.3 Batch Normalization

Batch normalization stabilizes training by normalizing layer inputs:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

3.4 Dropout

Dropout randomly disables neurons during training to prevent overfitting, forcing the network to learn robust features.

3.5 Degradation Problem

As networks deepen, accuracy may saturate or degrade. Residual learning and normalization techniques mitigate this issue.

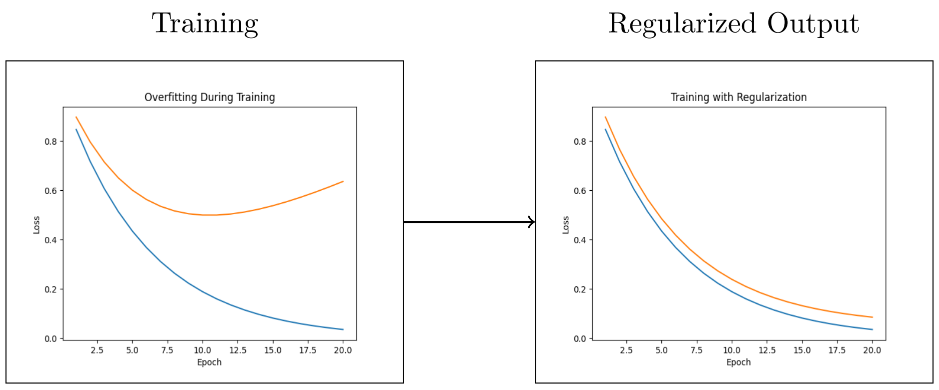

Figure 2: Effect of regularization techniques