Neural Networks Class 6

CNN Architectures & Advanced Topics

Learning Objectives:

- Master complete CNN architectures and mathematical flow
- Understand famous CNN models and their innovations
- Learn advanced CNN concepts and modern developments

Duration: 50 minutes

Complete CNN Architecture

Conv → ReLU → Pool → FC Pipeline

Typical CNN Structure:

$$\text{Input} \to [\text{Conv} \to \text{ReLU} \to \text{Pool}]^n \to [\text{FC} \to \text{ReLU}]^m \to \text{FC} \to \text{Softmax}$$

Mathematical Flow Example (32×32×3 → 10 classes):

1. **Conv1:** $32 \times 32 \times 3 \to 32 \times 32 \times 32$ (5×5 filters, pad=2)
2. **Pool1:** $32 \times 32 \times 32 \to 16 \times 16 \times 32$ (2×2 max pool)
3. **Conv2:** $16 \times 16 \times 32 \to 16 \times 16 \times 64$ (5×5 filters, pad=2)
4. **Pool2:** $16 \times 16 \times 64 \to 8 \times 8 \times 64$ (2×2 max pool)
5. **FC1:** $8 \times 8 \times 64 = 4096 \to 128$
6. **FC2:** $128 \to 10$ (softmax output)

Mathematical Flow Through CNN

Complete Forward Pass

Layer-by-layer Mathematical Description:

Convolutional Layer l:

$$\mathbf{Z}^{(l)}_{i,j,k} = \sum_{c=1}^{C^{(l-1)}} \sum_{u=1}^{F} \sum_{v=1}^{F} \mathbf{W}^{(l)}_{u,v,c,k} \mathbf{A}^{(l-1)}_{i+u-1,j+v-1,c} + b^{(l)}_k$$

Activation:

$$\mathbf{A}^{(l)}_{i,j,k} = \mathrm{ReLU}(\mathbf{Z}^{(l)}_{i,j,k})$$

Mathematical Flow Through CNN

Complete Forward Pass

Pooling Layer:

$$\mathbf{P}_{i,j,k}^{(l)} = \max_{u,v \in \text{pool}} \mathbf{A}_{2i+u,2j+v,k}^{(l)}$$

Flattening for FC:

$$\mathbf{h} = \text{flatten}(\mathbf{P}^{(L)}) \in \mathbb{R}^{H \times W \times C}$$

Receptive Field Through Network

Growing Perception

Receptive Field Growth:

Each layer sees increasingly larger input regions

Calculation Through Network:

- **Conv1 (3×3):** RF = 3×3
- **Pool1 (2×2, s=2):** RF = 4×4
- **Conv2 (3×3):** RF = 8×8
- **Pool2 (2×2, s=2):** RF = 10×10
- **Conv3 (3×3):** RF = 18×18

Receptive Field Through Network

Growing Perception

General Formula:

$$RF_l = RF_{l-1} + (K_l - 1) \times \prod_{i=1}^{l-1} S_i$$

Insight: Deep networks see large input regions while maintaining parameter efficiency

Parameter Counting Exercise

Complete Network Analysis

Example Network Architecture:

```
Input: 224×224×3
Conv1: 64 filters, 7×7, stride=2, pad=3 → 112×112×64
Pool1: 2×2, stride=2 → 56×56×64
Conv2: 128 filters, 3×3, stride=1, pad=1 → 56×56×128
Pool2: 2×2, stride=2 → 28×28×128
FC1: 1024 neurons
FC2: 10 classes (output)
```

Parameter Counting Exercise

Complete Network Analysis

Parameter Calculations:

- **Conv1:** $7 \times 7 \times 3 \times 64 + 64 = 9,472$
- **Conv2:** $3 \times 3 \times 64 \times 128 + 128 = 73,856$
- **FC1:** $28 \times 28 \times 128 \times 1024 + 1024 = 103,219,200$
- **FC2:** $1024 \times 10 + 10 = 10,250$

Total: ~103.3M parameters (FC layers dominate!)

LeNet-5 Architecture

The Pioneer (1998)

Yann LeCun's LeNet-5 for MNIST:

Architecture:

$$32 \times 32 \times 1 \to C1 \to S2 \to C3 \to S4 \to C5 \to F6 \to \text{Output}$$

LeNet-5 Architecture

The Pioneer (1998)

Layer Details:

- **C1:** 6 filters, 5×5 → 28×28×6 (156 params)
- **S2:** 2×2 avg pool → 14×14×6
- **C3:** 16 filters, 5×5 → 10×10×16 (2,416 params)
- **S4:** 2×2 avg pool → 5×5×16
- **C5:** 120 filters, 5×5 → 1×1×120 (48,120 params)
- **F6:** FC layer → 84 neurons (10,164 params)
- **Output:** 10 classes (850 params)

Total Parameters: ~62K (tiny by modern standards!)

AlexNet Architecture

The ImageNet Breakthrough (2012)

Key Innovations:

1. **ReLU activation** instead of tanh/sigmoid
2. **Dropout regularization** in FC layers
3. **Data augmentation** and multiple crops
4. **GPU implementation** for parallel training

AlexNet Architecture

The ImageNet Breakthrough (2012)

Architecture Highlights:

- **Input:** 224×224×3 (ImageNet scale)
- **Conv1:** 96 filters, 11×11, stride=4 → 55×55×96
- **Conv2:** 256 filters, 5×5 → 27×27×256
- **Conv3-5:** Multiple 3×3 convolutions
- **FC layers:** 4096 → 4096 → 1000 neurons
- **Total:** ~60M parameters

Mathematical Impact: Showed CNNs could scale to large datasets

AlexNet Mathematical Improvements

Why It Dominated

ReLU vs Sigmoid/Tanh:

$$\text{ReLU}(x) = \max(0, x) \text{ vs } \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Advantages:

- **No saturation** for positive inputs
- **Sparse activation** (many zeros)
- **Fast computation** (simple threshold)
- **Better gradient flow** (derivative is 0 or 1)

AlexNet Mathematical Improvements

Why It Dominated

Dropout Mathematics:

$$\tilde{\mathbf{h}} = \mathbf{r} \odot \mathbf{h} \text{ where } \mathbf{r} \sim \text{Bernoulli}(0.5)$$

Regularization Effect: Prevents co-adaptation of neurons

VGG Architecture

Deeper with Smaller Filters (2014)

Key Innovation: Small filters (3×3) instead of large ones

Mathematical Advantage:

Two 3×3 convolutions = One 5×5 convolution in receptive field

- **Parameters:** $2 \times (3^2 \times C^2) = 18C^2$ vs $5^2 \times C^2 = 25C^2$
- **Non-linearity:** 2 ReLU activations vs 1
- **28% parameter reduction** with more expressiveness

VGG Architecture

Deeper with Smaller Filters (2014)

VGG-16 Structure:

```
Conv3-64 × 2 → Pool → Conv3-128 × 2 → Pool → Conv3-256 × 3 → Pool
Conv3-512 × 3 → Pool → Conv3-512 × 3 → Pool → FC-4096 × 2 → FC-1000
```

Total: ~138M parameters (mostly in FC layers)

ResNet Innovation

Solving the Degradation Problem

Problem: Deeper networks perform worse than shallow ones

- Not due to overfitting (training error also increases)
- **Degradation problem:** Optimization difficulty

Solution: Residual connections

$$\mathbf{H}(\mathbf{x}) = \mathbf{F}(\mathbf{x}) + \mathbf{x}$$

ResNet Innovation

Solving the Degradation Problem

Mathematical Insight:

Instead of learning $\mathbf{H}(\mathbf{x})$, learn residual $\mathbf{F}(\mathbf{x}) = \mathbf{H}(\mathbf{x}) - \mathbf{x}$

Identity Mapping:

If optimal function is identity, easier to learn $\mathbf{F}(\mathbf{x}) = 0$ than $\mathbf{H}(\mathbf{x}) = \mathbf{x}$

ResNet Mathematical Analysis

Gradient Flow Through Skip Connections

Forward Pass:

$$\mathbf{x}_{l+1} = \mathbf{x}_l + \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)$$

Backward Pass:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_{l+1}} \left(1 + \frac{\partial \mathcal{F}(\mathbf{x}_l, \mathcal{W}_l)}{\partial \mathbf{x}_l}\right)$$

Key Insight: The "+1" ensures gradient flow even if $\frac{\partial \mathcal{F}}{\partial \mathbf{x}_l} \to 0$

ResNet Mathematical Analysis

Gradient Flow Through Skip Connections

Multi-hop Gradient:

$$\frac{\partial \mathcal{E}}{\partial \mathbf{x}_l} = \frac{\partial \mathcal{E}}{\partial \mathbf{x}_L} \left(1 + \sum_{i=l}^{L-1} \frac{\partial \mathcal{F}_i}{\partial \mathbf{x}_l} \right)$$

Result: Enables training of 152+ layer networks

Transfer Learning Mathematics

Feature Reuse and Fine-tuning

Pre-trained Feature Representation:

$$\mathbf{f}_{\text{pre}} = CNN_{\text{pre}}(\mathbf{x}; \boldsymbol{\theta}_{\text{pre}})$$

Transfer Learning Strategies:

1. Feature Extraction:

$$\mathbf{y} = \text{Classifier}(\mathbf{f}_{\text{pre}}; \boldsymbol{\theta}_{\text{new}})$$

Only train $\boldsymbol{\theta}_{\text{new}}$, freeze $\boldsymbol{\theta}_{\text{pre}}$

Transfer Learning Mathematics

Feature Reuse and Fine-tuning

2. Fine-tuning:

$$\mathbf{y} = CNN_{\text{full}}(\mathbf{x}; [\boldsymbol{\theta}_{\text{pre}}, \boldsymbol{\theta}_{\text{new}}])$$

Train all parameters with smaller learning rate for $\boldsymbol{\theta}_{\text{pre}}$

Mathematical Justification: Lower layers learn general features (edges, textures), higher layers learn task-specific features

Data Augmentation for Computer Vision

Mathematical Transformations

Geometric Transformations:

Rotation:

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Translation:

$$\mathbf{T}_{t_x,t_y}(\mathbf{x}) = \mathbf{x} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Data Augmentation for Computer Vision

Mathematical Transformations

Scaling:

$$\mathbf{S}_s(\mathbf{x}) = s \cdot \mathbf{x}$$

Horizontal Flip:

$$\text{Flip}(I[i, j]) = I[i, W - 1 - j]$$

Mathematical Effect: Increases effective dataset size by factor of augmentations used

Batch Normalization in CNNs

Channel-wise Normalization

For Convolutional Layers:

Normalize across batch and spatial dimensions, per channel

Statistics Computation:

$$\mu_c = \frac{1}{N \cdot H \cdot W} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} x_{n,h,w,c}$$

$$\sigma_c^2 = \frac{1}{N \cdot H \cdot W} \sum_{n=1}^{N} \sum_{h=1}^{H} \sum_{w=1}^{W} (x_{n,h,w,c} - \mu_c)^2$$

Batch Normalization in CNNs

Channel-wise Normalization

Normalization:

$$\hat{x}_{n,h,w,c} = \frac{x_{n,h,w,c} - \mu_c}{\sqrt{\sigma_c^2 + \epsilon}}$$

Scale and Shift (per channel):

$$y_{n,h,w,c} = \gamma_c \hat{x}_{n,h,w,c} + \beta_c$$

Modern CNN Innovations

Depthwise Separable Convolutions

Standard Convolution:

$$\text{Params} = K \times K \times C_{\text{in}} \times C_{\text{out}}$$
$$\text{FLOPs} = K \times K \times C_{\text{in}} \times C_{\text{out}} \times H \times W$$

Depthwise Separable:

1. **Depthwise:** Each input channel convolved separately

$$\text{Params} = K \times K \times C_{\text{in}}$$

2. **Pointwise:** 1×1 convolution across channels

$$\text{Params} = 1 \times 1 \times C_{\text{in}} \times C_{\text{out}}$$

Modern CNN Innovations

Depthwise Separable Convolutions

Total Reduction:

$$\frac{K \times K \times C_{\text{in}} + C_{\text{in}} \times C_{\text{out}}}{K \times K \times C_{\text{in}} \times C_{\text{out}}} = \frac{1}{C_{\text{out}}} + \frac{1}{K^2}$$

Attention in Computer Vision

Spatial Attention Mechanisms

Spatial Attention Map:

$$\mathbf{A} = \text{sigmoid}(\text{Conv}(\text{AdaptiveAvgPool}(\mathbf{F}) + \text{AdaptiveMaxPool}(\mathbf{F})))$$

Channel Attention:

$$\mathbf{M}_c = \sigma(\mathbf{W}_2(\text{ReLU}(\mathbf{W}_1(\text{GAP}(\mathbf{F}_c)))))$$

Attention in Computer Vision

Spatial Attention Mechanisms

Self-Attention for Images:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$$

Where $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are linear projections of spatial features

Vision Transformer (ViT): Pure attention-based architecture for images

Performance Metrics

Beyond Accuracy

Classification Metrics:

Top-1 Accuracy:

$$\mathrm{Acc}_1 = \frac{\text{correct predictions}}{\text{total predictions}}$$

Top-5 Accuracy:

$$\mathrm{Acc}_5 = \frac{\text{samples where true label in top 5 predictions}}{\text{total predictions}}$$

Performance Metrics

Beyond Accuracy

Precision and Recall (per class):

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c}, \quad \text{Recall}_c = \frac{TP_c}{TP_c + FN_c}$$

F1-Score:

$$F1_c = \frac{2 \times \text{Precision}_c \times \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c}$$

Model Efficiency Metrics

Speed and Memory Considerations

FLOPs (Floating Point Operations):

For convolution: $\text{FLOPs} = K^2 \times C_{\text{in}} \times H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}}$

Parameters vs FLOPs:

- **Parameters:** Storage/memory requirement
- **FLOPs:** Computational requirement

Model Efficiency Metrics

Speed and Memory Considerations

Inference Time:

- **Latency:** Time for single prediction
- **Throughput:** Predictions per second

Memory Usage:

- **Model size:** Parameter storage
- **Activation memory:** Intermediate feature maps
- **Peak memory:** Maximum during forward/backward pass

CNN Design Principles

Architecture Guidelines

Depth vs Width Trade-offs:

- **Deeper networks:** More representational power, harder to train
- **Wider networks:** More parameters per layer, easier to train

Filter Size Evolution:

- **Early layers:** Larger filters (7×7, 5×5) for low-level features
- **Later layers:** Smaller filters (3×3, 1×1) for complex combinations

CNN Design Principles

Architecture Guidelines

Channel Progression:

- **Input → Output:** Gradually increase channels while decreasing spatial size
- **Typical pattern:** 3 → 64 → 128 → 256 → 512 → 1024

Mathematical Intuition: Preserve information flow while reducing spatial dimensions

Future Directions

Beyond Traditional CNNs

Neural Architecture Search (NAS):

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{\text{val}}(\mathbf{w}^*(\alpha), \alpha)$$

EfficientNet Scaling:

$$\text{depth} = \alpha^\phi, \quad \text{width} = \beta^\phi, \quad \text{resolution} = \gamma^\phi$$

Subject to: $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$ and $\alpha, \beta, \gamma \geq 1$

Future Directions

Beyond Traditional CNNs

Vision Transformers:

Moving from convolution-based to attention-based architectures

3D CNNs:

Extension to video: (T, H, W, C) with temporal convolutions

Summary: Class 6 Key Concepts

Complete CNN Architectures:

- **Full pipeline:** Conv → ReLU → Pool → FC with mathematical flow
- **Parameter counting:** Understanding computational complexity
- **Receptive field growth:** How networks see larger input regions

Famous Architectures:

- **LeNet-5:** Pioneer for digit recognition
- **AlexNet:** ImageNet breakthrough with ReLU and dropout
- **VGG:** Deeper networks with small filters
- **ResNet:** Skip connections solve degradation problem

Summary: Class 6 Key Concepts

Advanced Topics:

- **Transfer learning:** Feature reuse and fine-tuning mathematics
- **Batch normalization:** Stabilizing training in CNNs
- **Modern innovations:** Attention, efficient architectures, NAS

The journey from simple perceptrons to sophisticated CNN architectures represents the power of mathematical foundations combined with computational innovation.