CNN (Convolutional Neural Network) Cheat Sheet

Codeium

2025-02-15

CNN (Convolutional Neural Network) Cheat Sheet

1. Basic Concepts

Convolution Operation

Takes an input image and applies a filter/kernel to produce a feature map

Example:

```
Input Image:         Filter:          Feature Map:
[1  2  3]           [1  0]           [8   11]
[4  5  6]     *     [2  1]    =      [17  20]
[7  8  9]
```

Step-by-step calculation: - Top-left (8) = 1×1 + 2×0 + 4×2 + 5×1 = 1 + 0 + 8 + 5 = 8 - Top-right (11) = 2×1 + 3×0 + 5×2 + 6×1 = 2 + 0 + 10 + 6 = 11 - Bottom-left (17) = 4×1 + 5×0 + 7×2 + 8×1 = 4 + 0 + 14 + 8 = 17 - Bottom-right (20) = 5×1 + 6×0 + 8×2 + 9×1 = 5 + 0 + 16 + 9 = 20

Padding

Types: Valid (no padding), Same (output size = input size)

Example:

```
Original:            With Zero Padding:
[1  2  3]           [0  0  0  0  0]
[4  5  6]    →      [0  1  2  3  0]
[7  8  9]           [0  4  5  6  0]
                    [0  7  8  9  0]
                    [0  0  0  0  0]
```

Stride

Controls filter movement across input. Example with stride 2:

```
Input:               Filter:          Output:
[1   2   3   4]     [1  0]
[5   6   7   8]     [0  1]          [12  16]
[9   10  11  12]                     [28  32]
[13  14  15  16]
```

2. Pooling Layers

Max Pooling (2×2, stride 2)

```
Input:                 Output:
```

```
[1   2   3   4]       [6    8]
[5   6   7   8]  →    [14   16]
[9   10  11  12]
[13  14  15  16]
```

Average Pooling (2×2, stride 2)

```
Input:                 Output:
[1   2   3   4]       [3.5  5.5]
[5   6   7   8]  →    [11.5 13.5]
[9   10  11  12]
[13  14  15  16]
```

3. Activation Functions

ReLU (Rectified Linear Unit)

f(x) = max(0, x)

```
Input: [-2, -1, 0, 1, 2]
Output: [0, 0, 0, 1, 2]
```

Softmax (for classification)

```
Raw scores: [2.0, 1.0, 0.1]
Softmax probabilities: [0.67, 0.24, 0.09]
```

4. Common Architecture Patterns

Basic CNN Structure

```
Input → Conv → ReLU → Pool → Conv → ReLU → Pool → Flatten → Dense → Output
```

Typical Layer Sizes

- Input: 224×224×3 (RGB image)
- Conv1: 32 filters of 3×3
- Pool1: 2×2 max pooling
- Conv2: 64 filters of 3×3
- Pool2: 2×2 max pooling
- Dense: 512 units
- Output: Number of classes (e.g., 10 for MNIST)

5. Training Concepts

Learning Rate

- Typical values: 0.1, 0.01, 0.001, 0.0001
- Example:

  ```
  weight_new = weight_old - learning_rate × gradient
  If gradient = 0.5 and learning_rate = 0.01:
  weight_new = weight_old - 0.01 × 0.5 = weight_old - 0.005
  ```

Batch Size

- Common values: 32, 64, 128, 256
- Example with batch size 4:

```
Batch 1: [image1, image2, image3, image4]
Batch 2: [image5, image6, image7, image8]
```

6. Common Problems & Solutions

Overfitting

Solutions: - Dropout (e.g., p=0.5 drops 50% of neurons) - Data Augmentation - L1/L2 Regularization

Vanishing Gradients

Solutions: - Batch Normalization - ResNet Connections - Proper initialization

7. Performance Metrics

Accuracy Calculation

```
Accuracy = Correct Predictions / Total Predictions
Example: 90 correct out of 100 = 90%
```

Confusion Matrix (Binary Classification)

```
          Predicted
Actual    0    1
0         TN   FP
1         FN   TP

Example:
          Predicted
Actual    0    1
0         45   5
1         10   40
```

8. Tips for Training

1. Start with a simple architecture
2. Use transfer learning when possible
3. Monitor validation loss
4. Use early stopping
5. Implement learning rate scheduling

9. Popular CNN Architectures

- LeNet-5: 7 layers
- AlexNet: 8 layers
- VGG-16: 16 layers
- ResNet-50: 50 layers
- Inception/GoogleNet: 22 layers

10. Advanced CNN Concepts

1. Different Types of Convolutions

- Standard Convolution
- Depthwise Separable Convolution

  ```
  Input → Depthwise Conv → Pointwise Conv (1×1)
  Reduces parameters by ~9x
  ```

- Dilated/Atrous Convolution

  ```
  Regular:     Dilated (rate=2):
  1 2 3        1 _ 2 _ 3
  4 5 6   →    _ _ _ _ _
  7 8 9        4 _ 5 _ 6
               _ _ _ _ _
               7 _ 8 _ 9
  ```

- Transposed Convolution (Deconvolution)

2. Advanced Architectures

- Inception Modules:

```
                   [1×1 Conv]
Input → Split →    [3×3 Conv]    → Concatenate
                   [5×5 Conv]
                   [MaxPool]
```

- ResNet Skip Connections:

```
Input
   ↓                       ↓
   → Conv → ReLU → Conv → Add → ReLU → Output
```

3. Loss Functions for CNNs

- Cross-Entropy Loss:

```
L = - (y_i * log(p_i))
where:
y_i = true label
p_i = predicted probability
```

- Focal Loss (for imbalanced datasets)
- Contrastive Loss (Siamese networks)
- Triplet Loss (face recognition)

4. Advanced Training Techniques

- Learning Rate Scheduling:

```
Step Decay:
lr = lr_initial × 0.1^(epoch/N)

Cosine Annealing:
lr = lr_initial × (1 + cos(  × epoch/total_epochs))/2
```

- Progressive Resizing
- Mixed Precision Training

- Knowledge Distillation

5. Model Optimization

- Quantization:

```
32-bit → 8-bit
Size ↓ ~4x
Speed ↑ ~3x
```

- Pruning
- Model Distillation
- Weight Sharing

6. Visualization Techniques

- Gradient-weighted Class Activation Mapping (Grad-CAM)
- Filter Visualization
- t-SNE for Feature Space
- Attention Maps

7. Advanced Applications

- Object Detection
  - YOLO
  - SSD
  - Faster R-CNN
- Semantic Segmentation
  - U-Net
  - DeepLab
  - FCN
- Instance Segmentation
  - Mask R-CNN
- Style Transfer
  - Content/Style Loss
  - AdaIN

8. Recent Developments

- Vision Transformers (ViT)
- MobileNet Architectures
- EfficientNet Family
- Self-Attention in CNNs

11. Troubleshooting Guide

Common Issues

1. Checkerboard Artifacts
   - Solution: Use proper stride in transposed convolutions
2. Feature Map Collapse
   - Solution: Check initialization, learning rate
3. Mode Collapse in GANs
   - Solution: Use conditional inputs, proper architecture

Debugging Tips

1. Feature Map Visualization
2. Gradient Flow Analysis
3. Learning Rate Range Test
4. BatchNorm Statistics Monitoring

12. Best Practices

Architecture Design

1. Start with proven architectures
2. Use skip connections for deep networks
3. Consider computational budget
4. Balance model capacity

Training Strategy

1. Progressive learning
2. Curriculum learning
3. Multi-task learning
4. Domain adaptation

Deployment Considerations

1. Model quantization
2. Platform-specific optimization
3. Latency vs accuracy tradeoff
4. Memory constraints

Remember: - Start small and scale up - Monitor training/validation curves - Use GPU acceleration when possible - Save checkpoints regularly