Comprehensive Computer Vision Interview Preparation Guide

Prepared for Deep Learning Interview

June 21, 2025

Contents

1 Introduction

This guide is tailored for candidates preparing for a deep learning interview with a focus on Computer Vision (CV). It assumes basic familiarity with neural networks from Kaggle competitions and provides an exhaustive overview of CV concepts, including foundational neural network components (optimizers, callbacks) and advanced CV architectures like CNNs and Vision Transformers. Each section details the evolution, mathematical foundations, applications, and practical considerations for when to use each technique.

2 Foundational Neural Network Concepts

Neural Networks (NNs) are the backbone of CV. This section covers core components relevant to CV tasks.

2.1 Neural Network Architecture

A neural network consists of:

- **Input Layer**: Receives image data (e.g., pixel values in RGB channels).
- **Hidden Layers**: Extract features using weights, biases, and activation functions.
- **Output Layer**: Produces predictions (e.g., class probabilities for classification).

Each neuron computes:

$$z = \sum (w_i \cdot x_i) + b, \quad a = \sigma(z)$$

where w_i are weights, x_i are inputs, b is the bias, and σ is the activation function.

2.2 Activation Functions

- **ReLU** (2010): $f(z) = \max(0, z)$, fast, mitigates vanishing gradients, default in CV.
- **Leaky ReLU** (2013): $f(z) = \max(0.01z, z)$, prevents dying ReLU, used in deeper networks.
- **ELU** (2015): $f(z) = z$ if $z > 0, \alpha(e^z - 1)$ otherwise, smoother gradients.
- **GELU** (2016): $f(z) = z \cdot \Phi(z)$, used in Transformers, approximates biological neurons.

When to Use: ReLU for most CV tasks; Leaky ReLU for deep CNNs; GELU for Vision Transformers.

2.3 Loss Functions

- **Cross-Entropy Loss**: $-\sum y_i \log(\hat{y}_i)$, for image classification.
- **Mean Squared Error (MSE)**: $\frac{1}{n}\sum (y_i - \hat{y}_i)^2$, for regression (e.g., bounding box coordinates).
- **Dice Loss**: $1 - \frac{2|y \cap \hat{y}|}{|y| + |\hat{y}|}$, for segmentation, handles class imbalance.

When to Use: Cross-Entropy for classification, Dice for segmentation.

2.4 Backpropagation

Backpropagation (1986) computes gradients of the loss w.r.t. weights using the chain rule, enabling optimization in CV models.

2.5 Optimizers

- **Stochastic Gradient Descent (SGD) with Momentum** (1988):

$$v_t = \gamma v_{t-1} + \eta \nabla L(w), \quad w \leftarrow w - v_t$$

  Stable for CV but slow.

- **AdaGrad** (2011): Adapts learning rate, good for sparse image features.
- **RMSProp** (2012): Uses moving average of squared gradients, better for non-convex problems.
- **Adam** (2014): Combines Momentum and RMSProp, default for CV:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

- **AdamW** (2017): Adds weight decay, improves generalization in CV.

When to Use: Adam for most CV tasks; SGD with Momentum for large datasets; AdamW for regularization.

2.6 Callbacks

- **Early Stopping**: Stops if validation loss plateaus, prevents overfitting.
- **Model Checkpoint**: Saves best model based on validation metrics.
- **Learning Rate Scheduler**: Reduces learning rate dynamically (e.g., cosine annealing).
- **Data Augmentation Callback**: Applies random transformations (e.g., flips, rotations) during training.

When to Use: Early Stopping and Checkpoint for all CV models; Augmentation for small datasets.

3 Computer Vision Architectures

CV focuses on tasks like image classification, object detection, and segmentation. Key architectures:

3.1 Convolutional Neural Networks (CNNs)

Introduced in 1989 (LeNet), CNNs are designed for grid-like data (images) due to:

- **Convolution Layers**: Extract features using filters:

$$(f * g)(i,j) = \sum_m \sum_n f(m,n) g(i-m, j-n)$$

- **Pooling Layers**: Downsample (e.g., Max Pooling reduces spatial dimensions by taking maximum).
- **Batch Normalization** (2015): Normalizes layer inputs, accelerates training.
- **Dropout** (2012): Randomly deactivates neurons, prevents overfitting.

Evolution:

- **LeNet** (1989): 5 layers, for digit recognition.
- **AlexNet** (2012): 8 layers, won ImageNet, introduced ReLU, Dropout, GPU acceleration.
- **ZFNet** (2013): Visualized CNN features, improved AlexNet.

- **VGG** (2014): 16-19 layers, small 3x3 filters, deeper networks.
- **Inception (GoogLeNet)** (2014): Inception modules, efficient multi-scale feature extraction.
- **ResNet** (2015): Residual connections:

  $$y = f(x) + x$$

  Enables training of 100+ layer networks.
- **DenseNet** (2017): Dense connections, reuses features, parameter-efficient.
- **EfficientNet** (2019): Compound scaling of depth, width, resolution.

Applications: Classification, detection, segmentation. **When to Use**: CNNs for most image tasks, especially when computational resources are limited.

3.2 Vision Transformers (ViT)

Introduced in 2020, ViTs apply Transformers to images by:

- Splitting images into patches (e.g., 16x16 pixels).
- Embedding patches into vectors.
- Using self-attention:

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

Advantages: Captures global context, outperforms CNNs on large datasets. **Challenges**: Requires large datasets, computationally expensive. **Variants**:

- **Swin Transformer** (2021): Hierarchical structure, efficient for dense tasks.
- **DeiT** (2020): Data-efficient ViT, uses distillation.

When to Use: ViT for large-scale classification; Swin for detection/segmentation.

3.3 Object Detection Architectures

- **R-CNN** (2014): Region proposals + CNN, slow.
- **Fast R-CNN** (2015): Shared convolution, faster.
- **Faster R-CNN** (2015): Region Proposal Network, end-to-end.
- **YOLO** (2016-2023): Single-stage, real-time detection, predicts bounding boxes and classes.
- **SSD** (2016): Single-shot, multi-scale feature maps.

When to Use: YOLO for real-time detection; Faster R-CNN for high accuracy.

3.4 Segmentation Architectures

- **FCN** (2014): Fully convolutional, end-to-end segmentation.
- **U-Net** (2015): Encoder-decoder with skip connections, ideal for medical imaging.
- **DeepLab** (2016-2018): Atrous convolutions for multi-scale context.
- **Mask R-CNN** (2017): Instance segmentation, extends Faster R-CNN.

When to Use: U-Net for medical segmentation; Mask R-CNN for instance segmentation.

4 Practical Considerations in CV

- **Data Augmentation**: Random crops, flips, color jitter to increase dataset size.
- **Transfer Learning**: Pre-trained models (e.g., ResNet on ImageNet) for small datasets.
- **Class Imbalance**: Use weighted loss or oversampling for segmentation.
- **Evaluation Metrics**:
  - Classification: Accuracy, F1-score.
  - Detection: mAP (mean Average Precision).
  - Segmentation: IoU (Intersection over Union).

5 Evolution Timeline

- 1989: LeNet (CNNs).
- 2012: AlexNet.
- 2014: VGG, Inception, R-CNN, FCN.
- 2015: ResNet, Faster R-CNN, U-Net.
- 2016: YOLO, SSD, DeepLab.
- 2017: DenseNet, Mask R-CNN.
- 2019: EfficientNet.
- 2020: ViT, DeiT.
- 2021: Swin Transformer.

6 Interview Tips

- Explain CNNs as hierarchical feature extractors (edges textures objects).
- Discuss trade-offs (e.g., YOLO speed vs. Faster R-CNN accuracy).
- Relate to Kaggle (e.g., used ResNet with transfer learning).
- Be ready to sketch a CNN or ViT architecture.

7 Conclusion

This guide equips you with a deep understanding of CV concepts for your interview. Focus on practical applications and clear explanations. Good luck!