Comprehensive Neural Networks, Computer Vision, and NLP Interview Preparation Guide

Prepared for Deep Learning Interview

June 21, 2025

Contents

1 Introduction **2**

2 Basic Neural Network Concepts **2**
- 2.1 Neural Network Architecture . . . 2
- 2.2 Activation Functions . . . 2
- 2.3 Loss Functions . . . 2
- 2.4 Backpropagation . . . 3
- 2.5 Optimizers . . . 3
- 2.6 Callbacks . . . 4

3 Computer Vision (CV) Concepts **4**
- 3.1 Convolutional Neural Networks (CNNs) . . . 4
- 3.2 Other CV Architectures . . . 4

4 Natural Language Processing (NLP) Concepts **4**
- 4.1 Recurrent Neural Networks (RNNs) . . . 5
- 4.2 Long Short-Term Memory (LSTM) . . . 5
- 4.3 Gated Recurrent Unit (GRU) . . . 5
- 4.4 Transformers . . . 5

5 Evolution Timeline **5**

6 Interview Tips **6**

7 Conclusion **6**

1 Introduction

This guide is designed for candidates preparing for a deep learning interview with limited time. It covers foundational neural network concepts, including optimizers and callbacks, followed by advanced topics in Computer Vision (CV) and Natural Language Processing (NLP), such as CNNs, RNNs, LSTMs, and Transformers. Each section details the evolution of these technologies, their applications, and when to use them. The material is structured to be comprehensive yet concise, assuming basic familiarity with neural networks from Kaggle competitions.

2 Basic Neural Network Concepts

Neural Networks (NNs) are computational models inspired by biological neurons, used for tasks like classification, regression, and pattern recognition. This section covers core components, optimizers, and callbacks.

2.1 Neural Network Architecture

A neural network consists of:

- **Input Layer**: Receives raw data (e.g., pixel values for images).
- **Hidden Layers**: Perform feature extraction using weights, biases, and activation functions.
- **Output Layer**: Produces predictions (e.g., class probabilities).

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 (e.g., ReLU, Sigmoid).

2.2 Activation Functions

Activation functions introduce non-linearity:

- **Sigmoid** (1950s): $\sigma(z) = \frac{1}{1+e^{-z}}$, outputs [0,1], used for binary classification but suffers from vanishing gradients.
- **Tanh** (1980s): $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$, outputs [-1,1], better than Sigmoid but still has vanishing gradient issues.
- **ReLU** (2010): $f(z) = \max(0, z)$, fast and mitigates vanishing gradients, used in most modern NNs.
- **Leaky ReLU** (2013): $f(z) = \max(0.01z, z)$, allows small gradients for negative inputs, prevents "dying ReLU" problem.
- **ELU** (2015): $f(z) = z$ if $z > 0, \alpha(e^z - 1)$ otherwise, smoother than ReLU, improves convergence.

When to Use: ReLU for most cases due to speed; Leaky ReLU/ELU if ReLU neurons die frequently.

2.3 Loss Functions

Loss functions measure prediction errors:

- **Mean Squared Error (MSE)**: $\frac{1}{n}\sum(y_i - \hat{y}_i)^2$, for regression.
- **Cross-Entropy Loss**: $-\sum y_i \log(\hat{y}_i)$, for classification.

- **Hinge Loss**: $\max(0, 1 - y_i \cdot \hat{y}_i)$, for SVM-like models.

When to Use: MSE for regression, Cross-Entropy for classification.

2.4 Backpropagation

Backpropagation (1986) computes gradients of the loss function w.r.t. weights using the chain rule, enabling weight updates via optimization.

2.5 Optimizers

Optimizers update weights to minimize loss. Their evolution:

- **Gradient Descent (GD)** (1847, formalized in NNs 1986):

  $$w \leftarrow w - \eta \nabla L(w)$$

  Updates weights using the full dataset gradient. Slow for large datasets.

- **Stochastic Gradient Descent (SGD)** (1951, popularized 1980s): Updates weights using single samples, faster but noisy.

- **Mini-Batch SGD** (1980s): Uses small batches, balances speed and stability.

- **Momentum** (1988): Adds a fraction of previous updates:

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

  Accelerates convergence.

- **AdaGrad** (2011): Adapts learning rate per parameter:

  $$w \leftarrow w - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla L(w)$$

  Good for sparse data, but learning rate shrinks over time.

- **RMSProp** (2012): Uses exponential moving average of squared gradients:

  $$E[g^2]_t = \rho E[g^2]_{t-1} + (1 - \rho) g_t^2, \quad w \leftarrow w - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$$

  Fixes AdaGrads diminishing learning rate.

- **Adam** (2014): Combines Momentum and RMSProp:

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

  $$w \leftarrow w - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t$$

  Widely used due to robustness.

- **AdamW** (2017): Adds weight decay to Adam, improves generalization.

- **Lookahead** (2019): Maintains fast and slow weights, improves stability.

When to Use:

- SGD with Momentum: Simple tasks, when computation is limited.
- Adam: Most deep learning tasks due to fast convergence.
- RMSProp: When Adam overfits or for RNNs.
- AdamW: When regularization is critical.

2.6 Callbacks

Callbacks are functions executed during training to monitor or control the process. Common callbacks:

- **Early Stopping**: Stops training if validation loss stops improving, prevents overfitting.
- **Model Checkpoint**: Saves the best model based on metrics.
- **Learning Rate Scheduler**: Adjusts learning rate (e.g., reduce on plateau).
- **TensorBoard**: Visualizes training metrics.

When to Use: Early Stopping for all models; Checkpoint for long training; Scheduler for fine-tuning.

3 Computer Vision (CV) Concepts

CV involves processing and understanding images/videos. Key architectures:

3.1 Convolutional Neural Networks (CNNs)

Introduced in 1989 (LeNet), CNNs excel in image tasks due to:

- **Convolution Layers**: Apply filters to extract features (edges, textures).

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

- **Pooling Layers**: Downsample feature maps (e.g., Max Pooling).
- **Fully Connected Layers**: Perform classification.

Evolution:

- **LeNet** (1989): Handwritten digit recognition.
- **AlexNet** (2012): Deep CNN, won ImageNet, used ReLU, Dropout.
- **VGG** (2014): Deeper networks with small filters.
- **ResNet** (2015): Residual connections to train very deep networks:

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

- **EfficientNet** (2019): Scales depth, width, resolution efficiently.

When to Use: CNNs for image classification, object detection, segmentation.

3.2 Other CV Architectures

- **Vision Transformers (ViT)** (2020): Apply Transformers to image patches, outperform CNNs on large datasets.
- **YOLO** (2016): Real-time object detection.
- **U-Net** (2015): Image segmentation, especially medical imaging.

When to Use: ViT for large-scale image tasks; YOLO for real-time detection; U-Net for segmentation.

4 Natural Language Processing (NLP) Concepts

NLP involves processing and understanding text. Key architectures:

4.1 Recurrent Neural Networks (RNNs)

Introduced in 1980s, RNNs process sequences by maintaining a hidden state:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b)$$

Issue: Vanishing/exploding gradients for long sequences. **When to Use**: Simple sequential tasks with short dependencies.

4.2 Long Short-Term Memory (LSTM)

Introduced in 1997, LSTMs address vanishing gradients using gates:

- **Forget Gate**: Decides what to discard.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- **Input Gate**: Decides what to store.
- **Output Gate**: Decides what to output.

When to Use: Tasks with long-term dependencies (e.g., text generation).

4.3 Gated Recurrent Unit (GRU)

Introduced in 2014, GRUs simplify LSTMs with fewer gates, faster training. **When to Use**: When LSTMs are too slow, similar performance.

4.4 Transformers

Introduced in 2017 ("Attention is All You Need"), Transformers use self-attention:

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

Advantages: Parallel processing, captures long-range dependencies. **Evolution**:

- **BERT** (2018): Bidirectional context, excels in classification.
- **GPT** (2018): Autoregressive, excels in generation.
- **T5** (2020): Text-to-text framework.

When to Use: Most modern NLP tasks due to superior performance.

5 Evolution Timeline

- 1980s: RNNs, SGD, Backpropagation.
- 1989: CNNs (LeNet).
- 1997: LSTMs.
- 2011: AdaGrad.
- 2012: AlexNet, RMSProp.
- 2014: Adam, GRUs.
- 2015: ResNet, U-Net.

- 2016: YOLO.
- 2017: Transformers, AdamW.
- 2018: BERT, GPT.
- 2020: ViT, T5.

6 Interview Tips

- Explain concepts intuitively (e.g., CNNs as feature extractors).
- Relate to Kaggle experience (e.g., used Adam for faster convergence).
- Discuss trade-offs (e.g., RNN vs. Transformer for sequence length).
- Be ready to write pseudocode for backpropagation or attention.

7 Conclusion

This guide covers all essential NN, CV, and NLP concepts for your interview. Focus on understanding when to use each architecture and optimizer, and practice explaining them clearly. Good luck!