

## **A Study of Word Embedding Models, CNN Architectures, and Vector Databases**

**SAANVI VENKATESH KULKARNI,1RVU23CSE391**

1) What is CBOW?(continuous bag of words)

CBOW (Continuous Bag of Words) is a Word2Vec model that learns word meanings by predicting a target word from its surrounding context words. For example, consider the sentence: "The cat sits on the mat." If the target word is "sits," CBOW takes the surrounding words "The," "cat," "on," "the," and "mat" as input. It converts these context words into vector representations (embeddings), averages them, and then tries to predict the target word "sits." During training, the model adjusts the word vectors so that words appearing in similar contexts develop similar embeddings. Over time, this allows the model to capture semantic relationships ,for instance, words like "cat" and "dog" may end up with similar vector representations because they appear in similar contexts. CBOW ignores word order (hence "bag of words") and focuses only on the surrounding words to learn meaningful numerical representations of language.

2) What is GloVe (with example)

GloVe (Global Vectors for Word Representation) is a word embedding method that learns word meanings using global word co-occurrence statistics from a corpus. Unlike Word2Vec, which learns from local context windows, GloVe builds a large co-occurrence matrix that counts how often words appear together across the entire dataset.

Example:

Consider the sentences:

"I like ice cream"

"I like chocolate"

A co-occurrence matrix would record how often:

"like" appears with "ice"

"like" appears with "chocolate"

"ice" appears with "cream"

GloVe uses these global counts to learn embeddings such that ratios of co-occurrence probabilities capture meaning. For example, the relationship between "ice" and "steam" relative to "solid" and "gas" helps encode semantic meaning in vector space.

### 3) Difference between Word2Vec and GloVe

Word2Vec:

- Predictive model
- Learns embeddings by predicting context (CBOW or Skip-Gram)
- Uses local context windows
- Neural network-based training

GloVe:

- Count-based model
- Uses global co-occurrence matrix
- Learns by factorizing the matrix
- Captures global statistical structure

Word2Vec learns by predicting words.

GloVe learns by analyzing overall word co-occurrence counts.

### 4) What is LeNet?

LeNet is one of the earliest convolutional neural networks, developed by Yann LeCun in 1998 for handwritten digit recognition (like MNIST).

Architecture:

- Convolution layer
- Pooling layer
- Convolution layer
- Pooling layer
- Fully connected layers

It was mainly used for digit classification (0–9) and laid the foundation for modern CNNs.

### 5) What is LlamaIndex?

LlamaIndex (formerly GPT Index) is a framework used to connect large language models (LLMs) with external data sources. It helps structure, index, and retrieve documents so that LLMs can perform Retrieval-Augmented Generation (RAG).

Example:

If you upload company documents, LlamaIndex:

- Indexes them
- Converts them into embeddings
- Retrieves relevant chunks when you ask a question
- It acts as a bridge between LLMs and your custom data.

## 6) What is VGGNet?

VGGNet is a deep convolutional neural network developed by Oxford (Visual Geometry Group). It became famous in ImageNet competitions.

Key features:

- Uses very small 3x3 convolution filters
- Very deep architecture (16 or 19 layers)
- Simple and uniform design
- Example versions:
- VGG16
- VGG19

It improved performance by increasing network depth systematically.

## 7) Explain AlexNet and VGGNet

AlexNet:

- Developed in 2012
- Won ImageNet competition
- 8 layers (5 convolution + 3 fully connected)
- Used ReLU activation
- Used dropout to reduce overfitting
- Introduced GPU training for CNNs

VGGNet:

- Developed after AlexNet
- Much deeper (16–19 layers)
- Uses repeated 3x3 convolutions
- Simpler but deeper structure
- Higher accuracy but more parameters

Difference:

AlexNet introduced deep CNN success.

VGG improved depth and structure refinement.

## 8) What is Skip-Gram? (with code snippet)

Skip-Gram is a Word2Vec model that predicts surrounding context words from a target word.

Example:

Sentence:

"I love deep learning"

If target = "deep"

Skip-Gram predicts:

"I", "love", "learning"

So:

Target → Context

PyTorch Skip-Gram example:

```
import torch
import torch.nn as nn
import torch.optim as optim

class SkipGram(nn.Module):
    def __init__(self, vocab_size, embedding_dim):
        super(SkipGram, self).__init__()
        self.embeddings = nn.Embedding(vocab_size, embedding_dim)
        self.output = nn.Linear(embedding_dim, vocab_size)

    def forward(self, target_word):
        embed = self.embeddings(target_word)
        out = self.output(embed)
        return out

# Example usage
vocab_size = 1000
embedding_dim = 50
model = SkipGram(vocab_size, embedding_dim)

target = torch.tensor([5]) # example word index
output = model(target)
print(output.shape)
```

## 10) What is a VectorDB?

A Vector Database stores high-dimensional vector embeddings and allows fast similarity search.

Instead of storing normal text, it stores vectors like:

`cat → [0.23, 0.11, 0.89, ...]`

When you query:  
“small furry animal”

It converts the query to a vector and finds nearest vectors using cosine similarity or Euclidean distance.

Used in:

RAG systems, Chatbots, Semantic search