```
# ----------------------------------------
# CBOW (Continuous Bag of Words) Model using Keras
# ----------------------------------------

import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.metrics.pairwise import cosine_similarity

# ----------------------------------------
# 1. Sample Corpus
# ----------------------------------------
corpus = [
    "machine learning models learn patterns from data",
    "deep learning uses neural networks for representation learning",
    "humans learn from experience while machines learn from data",
    "artificial intelligence combines reasoning and learning capabilities",
    "supervised learning requires labeled data to train models"
]

# Context window size
window_size = 2

# ----------------------------------------
# 2. Tokenization and Vocabulary Creation
# ----------------------------------------
tokenizer = Tokenizer(oov_token=None)
tokenizer.fit_on_texts(corpus)

# Convert text to sequences of integers
sequences = tokenizer.texts_to_sequences(corpus)
word_index = tokenizer.word_index
index_word = {i: w for w, i in word_index.items()}

vocab_size = len(word_index) + 1
print("Vocabulary size:", vocab_size)

# ----------------------------------------
# 3. Generate Context–Target Pairs
# ----------------------------------------
contexts, targets = [], []

for seq in sequences:
    for i, target in enumerate(seq):
        # Get context words (window_size to left and right)
        left = seq[max(0, i - window_size):i]
```

```
        right = seq[i+1:i+1+window_size]
        context = left + right

        # Pad context to maintain fixed size (2*window_size)
        context = pad_sequences([context], maxlen=2*window_size, padding='pre')[0]

        contexts.append(context)
        targets.append(target)

contexts = np.array(contexts)
targets = np.array(targets)

print("Training samples:", len(targets))

# ----------------------------------------
# 4. Build CBOW Model
# ----------------------------------------
embedding_dim = 50  # Dimensionality of word embeddings

inputs = layers.Input(shape=(2 * window_size,), dtype='int32')
embed = layers.Embedding(
    input_dim=vocab_size,
    output_dim=embedding_dim,
    input_length=2 * window_size,
    mask_zero=True
)(inputs)
avg = layers.GlobalAveragePooling1D()(embed)
out = layers.Dense(vocab_size, activation='softmax')(avg)

cbow = keras.Model(inputs=inputs, outputs=out)
cbow.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
cbow.summary()

# ----------------------------------------
# 5. Train the Model
# ----------------------------------------
history = cbow.fit(contexts, targets, epochs=100, batch_size=32, verbose=2)

# ----------------------------------------
# 6. Extract Learned Embeddings
# ----------------------------------------
embedding_layer = cbow.get_layer(index=1)
embedding_weights = embedding_layer.get_weights()[0]

# ----------------------------------------
# 7. Function to Find Nearest Words
# ----------------------------------------
```

```
def nearest(word, k=5):
    if word not in word_index:
        return []
    idx = word_index[word]
    vec = embedding_weights[idx].reshape(1, -1)
    sims = cosine_similarity(vec, embedding_weights)[0]
    sims[idx] = -1  # exclude itself
    sims[0] = -1    # exclude padding
    topk = sims.argsort()[-k:][::-1]
    return [(index_word.get(i, "<PAD>"), float(sims[i])) for i in topk]

# ----------------------------------------
# 8. Display Similar Words
# ----------------------------------------
sample_words = ["learning", "data", "models", "humans"]

for w in sample_words:
    print(f"\nNearest to '{w}':")
    for sim_word, score in nearest(w, k=5):
        print(f"  {sim_word:15s}  →  Similarity: {score:.4f}")
```

🔹 Aim:

To implement a **Continuous Bag of Words (CBOW)** model using TensorFlow and Keras for generating **word embeddings** and finding **semantic similarity** between words.

🔹 Objectives:

1. To understand how neural networks learn **semantic representations** of words.
2. To implement the **CBOW model** for word embedding generation.
3. To visualize semantic relationships between words using **cosine similarity**.
4. To demonstrate how context helps in predicting the target word in a sentence.

🔹 Theory:

1. Introduction to Word Embeddings

In Natural Language Processing (NLP), **Word Embeddings** are dense vector representations of words where semantically similar words have similar vector values.
They help convert words into numerical form for input to machine learning models.

Examples of embedding models:

- Word2Vec (CBOW and Skip-gram)
- GloVe
- FastText

---

2. Word2Vec Model

Word2Vec, introduced by Mikolov et al. (2013), is a neural network-based model for learning word embeddings.
It comes in two architectures:

1. **CBOW (Continuous Bag of Words):** Predicts a target word based on its surrounding context words.
2. **Skip-Gram:** Predicts surrounding context words based on a target word.

---

3. CBOW Architecture

In the **CBOW** model:

- **Input:** Context words surrounding a target word.
- **Output:** The target word.
- The model learns to predict the central (target) word from the context.

For example, in the sentence:

> "machine learning models learn patterns from data"
> If the target word is **"models"**, and window size = 2,
> the context words are `[machine, learning, learn, patterns]`.

The neural network learns to associate these context words with the target word.

7. Cosine Similarity

After training, **cosine similarity** is used to measure how close two word vectors are:

cosine_similarity(A,B)=A⋅B∥A∥ ∥B∥\text{cosine_similarity}(A, B) = \frac{A \cdot B}{\|A\| \|B\|}cosine_similarity(A,B)=∥A∥ ∥B∥A⋅B

Words with higher cosine similarity values are semantically related.

🔹 Algorithm / Steps:

1. Import required libraries (`TensorFlow`, `Keras`, `NumPy`, `sklearn`).
2. Define a small **corpus** of sentences.
3. Tokenize the text using `Tokenizer()` and create **word-to-index mappings**.
4. Generate **context–target pairs** based on a chosen window size.
5. Build the CBOW model:
    - Input → Embedding → Average Pooling → Dense (Softmax)
6. Compile the model with **Adam optimizer** and **cross-entropy loss**.
7. Train the model on context-target pairs.
8. Extract the learned embedding weights.
9. Compute **cosine similarity** to find the nearest words.
10. Display semantically similar words for given examples.

🔹 Conclusion:

The **CBOW model** efficiently learns **semantic relationships** among words using contextual information.
It is a simple yet powerful neural approach for building **word embeddings**, forming the foundation for many modern NLP systems such as **Word2Vec, GloVe, and BERT**.

🔹 Applications:

- Text classification and sentiment analysis.
- Machine translation.
- Information retrieval and question answering.
- Chatbots and conversational AI systems.