Training Day-74 Report:

Simple Neural Network with TensorFlow, **Word Embedding**, and understanding **CBOW** and **Skip-gram** models.

1. Simple Neural Network with TensorFlow

Goal:

Build a neural network using TensorFlow to solve a binary classification problem (e.g., XOR).

Implementation:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# XOR dataset
X = [[0, 0], [0, 1], [1, 0], [1, 1]]
y = [0, 1, 1, 0]

# Define a sequential model
model = Sequential([
    Dense(8, activation='relu', input_shape=(2,)),  # Hidden layer with 8 neurons
    Dense(1, activation='sigmoid')                  # Output layer for binary classification
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=100, verbose=1)
```

```
# Evaluate the model
print("Evaluation Results:", model.evaluate(X, y))

# Predictions
print("Predictions:", model.predict(X))
```

This simple NN demonstrates how to use TensorFlow to solve small problems with minimal code.

2. Word Embedding

Word embeddings convert words into dense vectors in a continuous vector space, capturing semantic meanings.

Using TensorFlow/Keras:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Flatten, Dense

# Sample vocabulary and corresponding tokenized sentences
vocab_size = 50
embedding_dim = 8
max_length = 10

# Example tokenized input
sentences = [[1, 2, 3, 4], [3, 4, 1, 2]]
padded_sentences = tf.keras.preprocessing.sequence.pad_sequences(sentences, maxlen=max_length)

# Define the embedding layer model
model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length),
    Flatten(),
    Dense(1, activation='sigmoid')
])
```

```
# Compile and summarize the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()

# Dummy output labels
labels = [1, 0]

# Train the model
model.fit(padded_sentences, labels, epochs=10)
```

This creates word embeddings for a vocabulary and uses them in a classification task.

3. CBOW (Continuous Bag of Words) & Skip-gram

CBOW and Skip-gram are two approaches used in Word2Vec for learning word embeddings.

CBOW:

- Predicts a target word from its surrounding context words.
- Suitable for frequent words in a corpus.

Skip-gram:

- Predicts context words from a target word.
- Suitable for infrequent words.

TensorFlow Implementation of Skip-gram:

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Dot, Flatten

# Sample dataset
word_pairs = [(1, 2), (1, 3), (2, 3), (2, 4)]  # (target, context)
vocab_size = 5  # Vocabulary size
embedding_dim = 8
```

```
# Prepare inputs and labels
targets, contexts = zip(*word_pairs)
targets = np.array(targets)
contexts = np.array(contexts)

# Define the model
input_target = Input((1,))
input_context = Input((1,))

embedding = Embedding(vocab_size, embedding_dim, input_length=1)

target_embedding = embedding(input_target)
context_embedding = embedding(input_context)

dot_product = Dot(axes=-1)([target_embedding, context_embedding])
output = Flatten()(dot_product)

model = Model(inputs=[input_target, input_context], outputs=output)
model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit([targets, contexts], np.ones(len(word_pairs)), epochs=100)
```

This implementation focuses on Skip-gram, where the network learns word embeddings by maximizing the similarity between target and context words.

Summary:

- **Simple NN**: Built a basic neural network using TensorFlow for XOR classification.
- **Word Embedding**: Showed how to use TensorFlow's Embedding layer for converting words into dense vectors.

- **CBOW & Skip-gram**: Explained concepts and provided a Skip-gram implementation in TensorFlow.