Assignment -2

Utkarsh Pal
2020144

Assumptions:-
I have removed the images and were reporting a 404 not found. Following this if there is any product_id with 0 valid images the whole has been dropped.
Moreover there is one row with no review and i have replaced it with <no-review> placeholder.

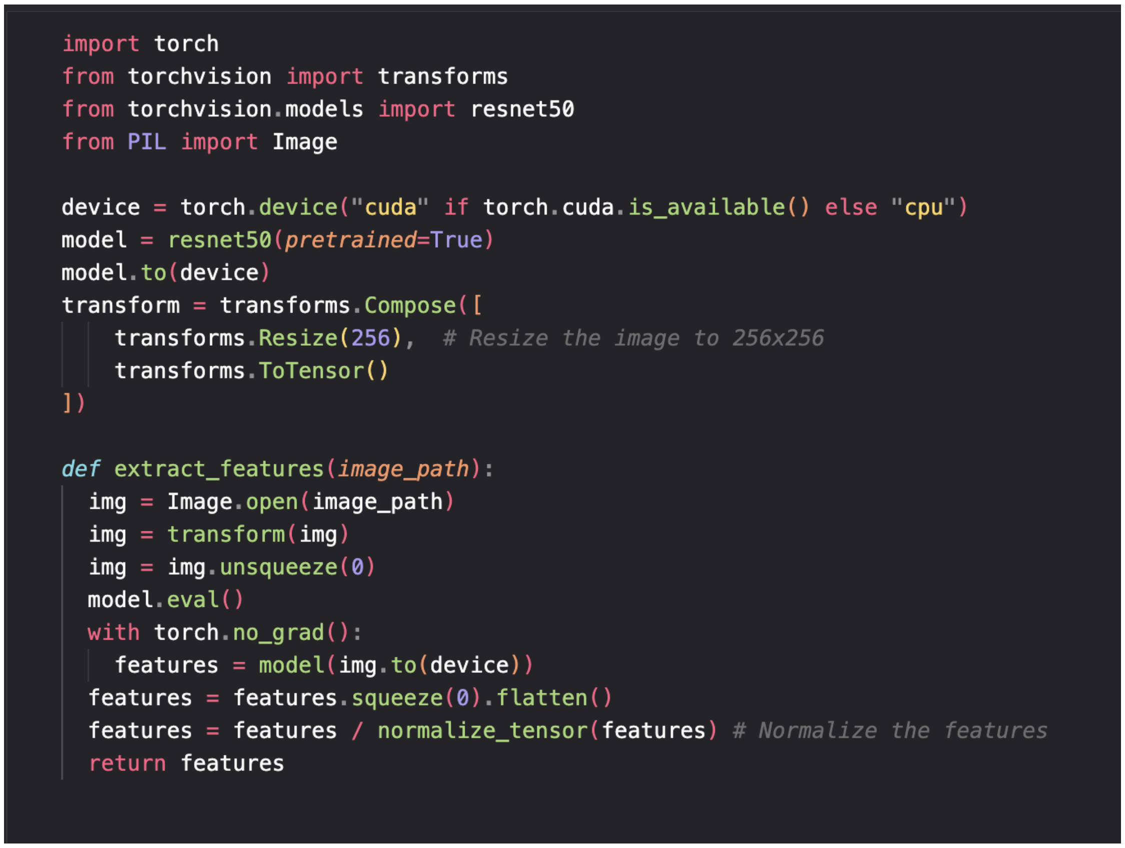
Q-1 a)

The code aims to create variations of existing images by applying random transformations to them. I used the following 3 transformations:-

- rotate_image(image, angle): Rotates an image by a specified angle.
- flip_image(image, flip): Flips an image horizontally or vertically.
- alter_brightness(image, factor): Adjusts the brightness of an image by a specified factor.

Q-1 b/c)

The code snippet is used for extracting image features using a pre-trained ResNet50 model.
The output features were also normalized by dividing each feature with its norm.

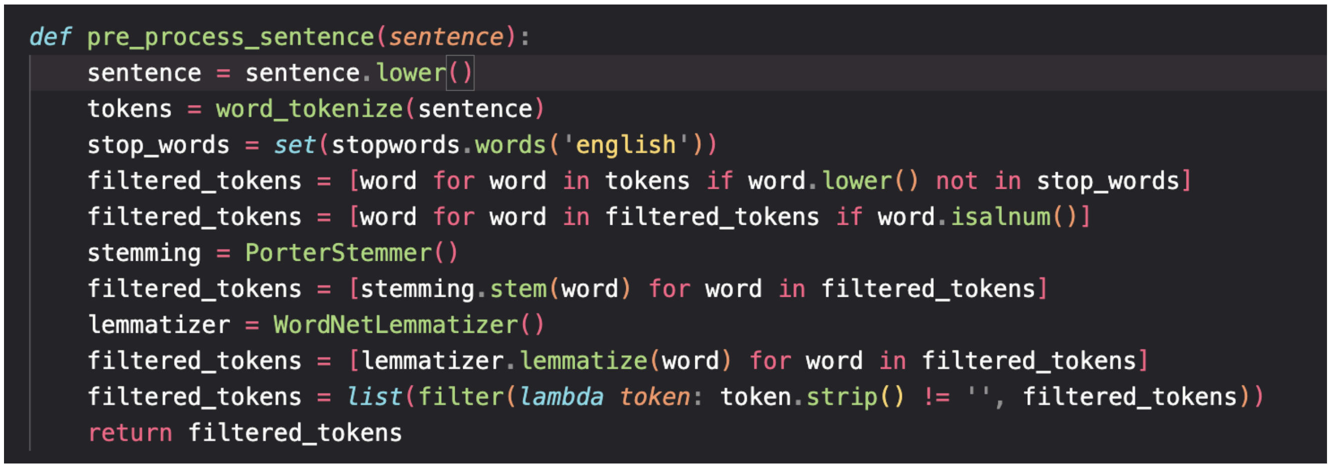


```python
import torch
from torchvision import transforms
from torchvision.models import resnet50
from PIL import Image

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = resnet50(pretrained=True)
model.to(device)
transform = transforms.Compose([
    transforms.Resize(256),  # Resize the image to 256x256
    transforms.ToTensor()
])

def extract_features(image_path):
  img = Image.open(image_path)
  img = transform(img)
  img = img.unsqueeze(0)
  model.eval()
  with torch.no_grad():
    features = model(img.to(device))
  features = features.squeeze(0).flatten()
  features = features / normalize_tensor(features) # Normalize the features
  return features
```

Q-2 a)

This code snippet (pre_process_sentence) cleans text data for NLP tasks. It performs the following:

- Lowercases the sentence for consistency.
- Splits the sentence into words (tokens).
- Removes stop words (common, uninformative words like "the").
- Keeps alphanumeric characters (letters and numbers) only.
- Optionally applies stemming or lemmatization (reducing words to base forms).
- Removes empty tokens.



```python
def pre_process_sentence(sentence):
    sentence = sentence.lower()
    tokens = word_tokenize(sentence)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
    filtered_tokens = [word for word in filtered_tokens if word.isalnum()]
    stemming = PorterStemmer()
    filtered_tokens = [stemming.stem(word) for word in filtered_tokens]
    lemmatizer = WordNetLemmatizer()
    filtered_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]
    filtered_tokens = list(filter(lambda token: token.strip() != '', filtered_tokens))
    return filtered_tokens
```

Q-2 b)

This code calculates the TF-IDF (Term Frequency-Inverse Document Frequency) for a dataset of product descriptions.

1. Building Vocabulary:

    Creates a set of unique words (vocabulary) across all product descriptions.

2. Constructing Document-Term Matrix:

    Initializes a dictionary df to store document frequency (DF) for each word in the vocabulary.

    For each product, Increments the DF count for each unique word encountered in a product description.

3.Iterates through product IDs and their associated text.

    Calculates TF-IDF for each word in the vocabulary for each product:

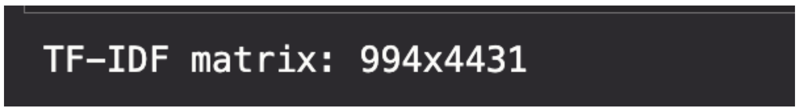
    Term Frequency (TF): Ratio of a word's count in the product description to the total words (with smoothing to avoid division by zero).

    Inverse Document Frequency (IDF): Logarithm of the total number of products divided by the number of products containing the word.

    TF-IDF: Product of TF and IDF.

    Returns a list of all TF-IDF vectors (one per product).

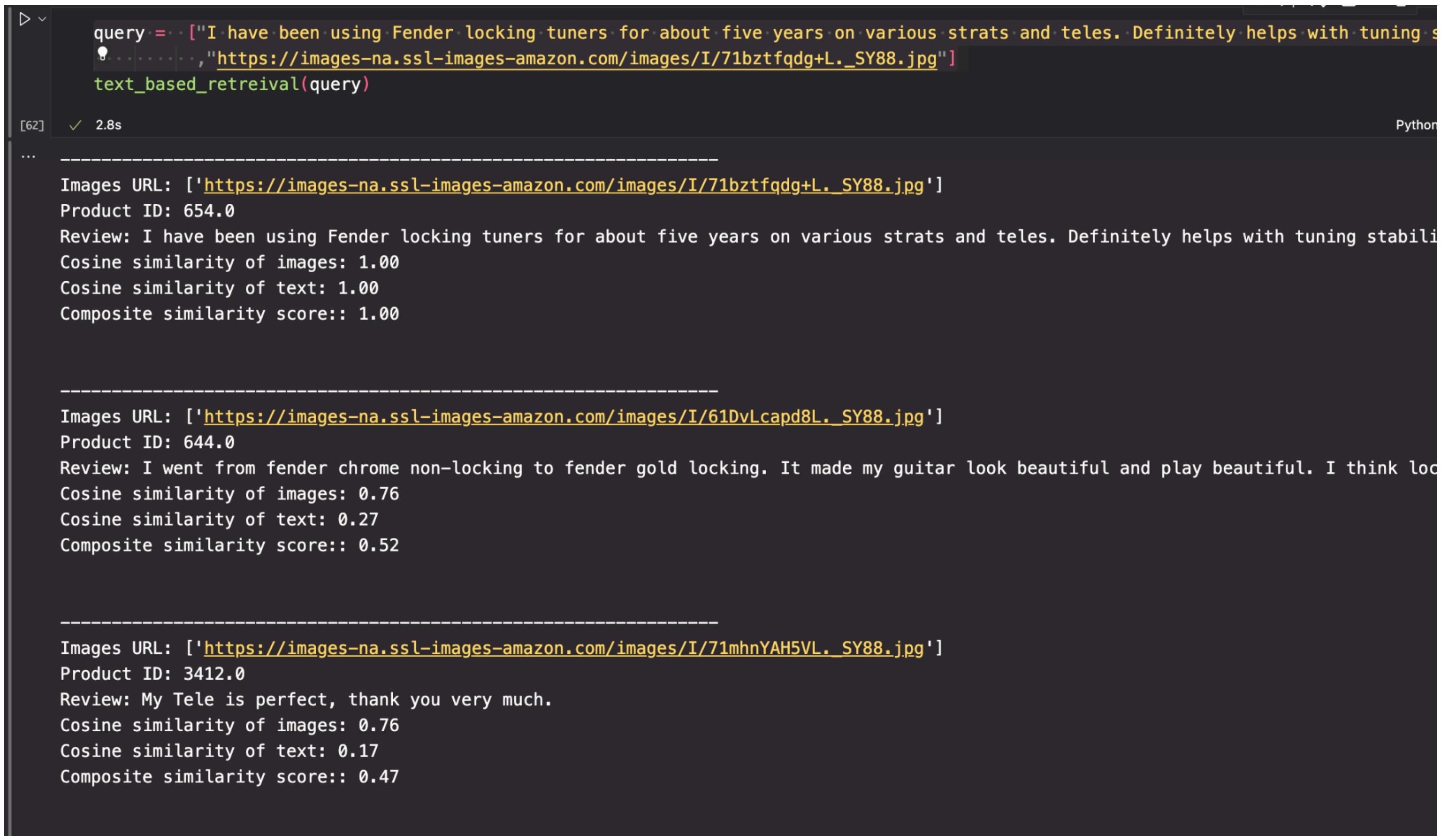
Size of TF-IDF matrix



```
TF-IDF matrix: 994x4431
```

Q-3)
b)
This code implements a **text-based** product retrieval system with an optional image similarity component.

- For a query_text we first calculate tf-idf vector for the given query
- For every product_id we calculate tf-idf vector for its review
- Then we calculate and store the cosine similarity for the above vectors.
- Take the top 3 entries and print the answer.
- For the image cosine similarity take the average of cosine similarities for a product.
- Print the combined cosine similarity by taking arithmetic mean of image cosine similarity and text similarity.



```
query = ["I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning s
        ,"https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg"]
text_based_retreival(query)
```

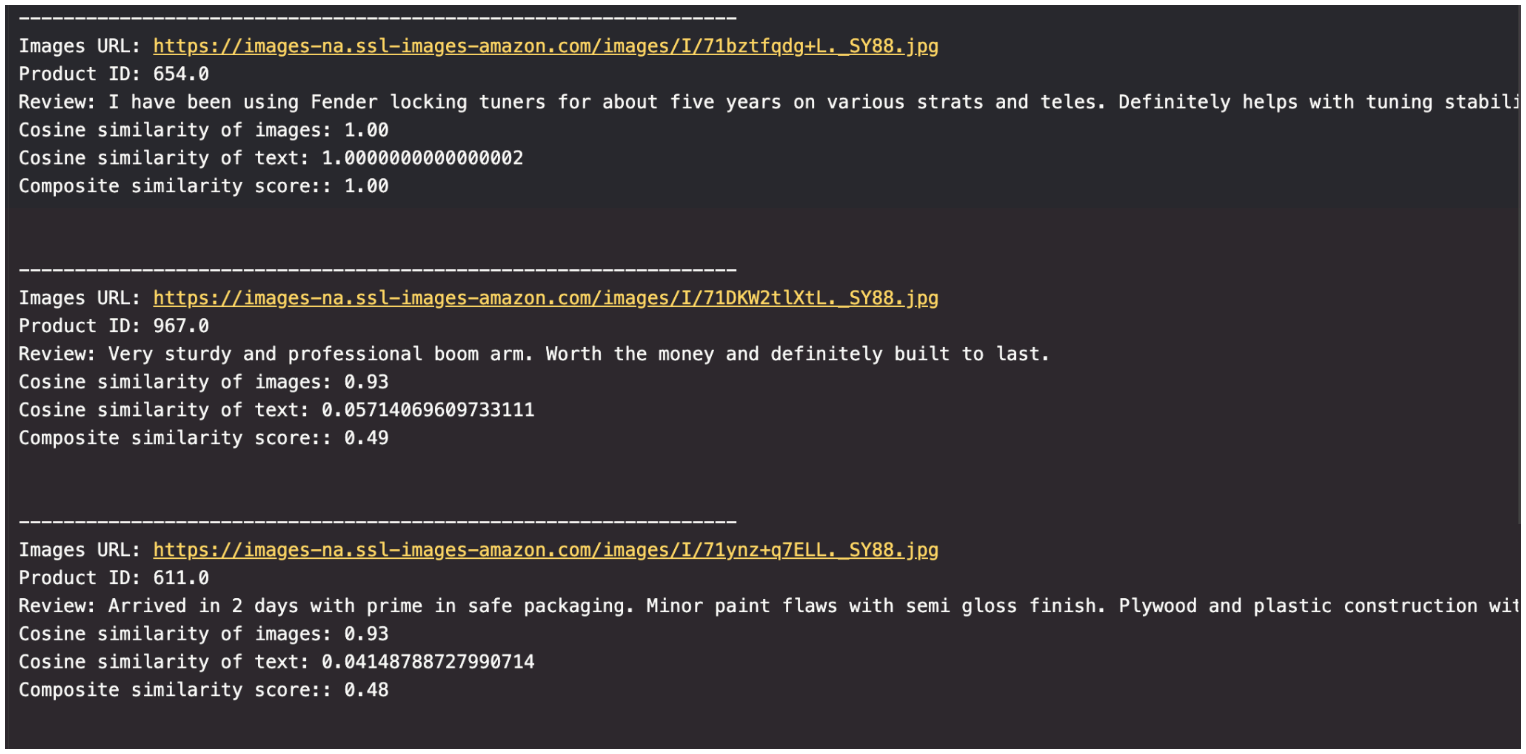
```
[62]  ✓ 2.8s                                                                                     Python
```

```
-----------------------------------------------------------------
Images URL: ['https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg']
Product ID: 654.0
Review: I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning stabili
Cosine similarity of images: 1.00
Cosine similarity of text: 1.00
Composite similarity score:: 1.00


-----------------------------------------------------------------
Images URL: ['https://images-na.ssl-images-amazon.com/images/I/61DvLcapd8L._SY88.jpg']
Product ID: 644.0
Review: I went from fender chrome non-locking to fender gold locking. It made my guitar look beautiful and play beautiful. I think loc
Cosine similarity of images: 0.76
Cosine similarity of text: 0.27
Composite similarity score:: 0.52


-----------------------------------------------------------------
Images URL: ['https://images-na.ssl-images-amazon.com/images/I/71mhnYAH5VL._SY88.jpg']
Product ID: 3412.0
Review: My Tele is perfect, thank you very much.
Cosine similarity of images: 0.76
Cosine similarity of text: 0.17
Composite similarity score:: 0.47
```

Q-3 a)
This code implements an image-based product retrieval system with an optional text similarity component.

- For a query_image we first calculate feature vector for the given query
- For every product_id we calculate feature vector for its image
- Then we calculate and store the maximum cosine similarity for the above vectors.
- Take the top 3 entries and print the answer.
- For the text cosine similarity just take cosine similarities for a product review and query text.

Print the combined cosine similarity by taking arithmetic mean of image cosine similarity and text similarity.



```
----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg
Product ID: 654.0
Review: I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning stabili
Cosine similarity of images: 1.00
Cosine similarity of text: 1.0000000000000002
Composite similarity score:: 1.00


----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/71DKW2tlXtL._SY88.jpg
Product ID: 967.0
Review: Very sturdy and professional boom arm. Worth the money and definitely built to last.
Cosine similarity of images: 0.93
Cosine similarity of text: 0.05714069609733111
Composite similarity score:: 0.49


----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/71ynz+q7ELL._SY88.jpg
Product ID: 611.0
Review: Arrived in 2 days with prime in safe packaging. Minor paint flaws with semi gloss finish. Plywood and plastic construction wit
Cosine similarity of images: 0.93
Cosine similarity of text: 0.04148788727990714
Composite similarity score:: 0.48
```

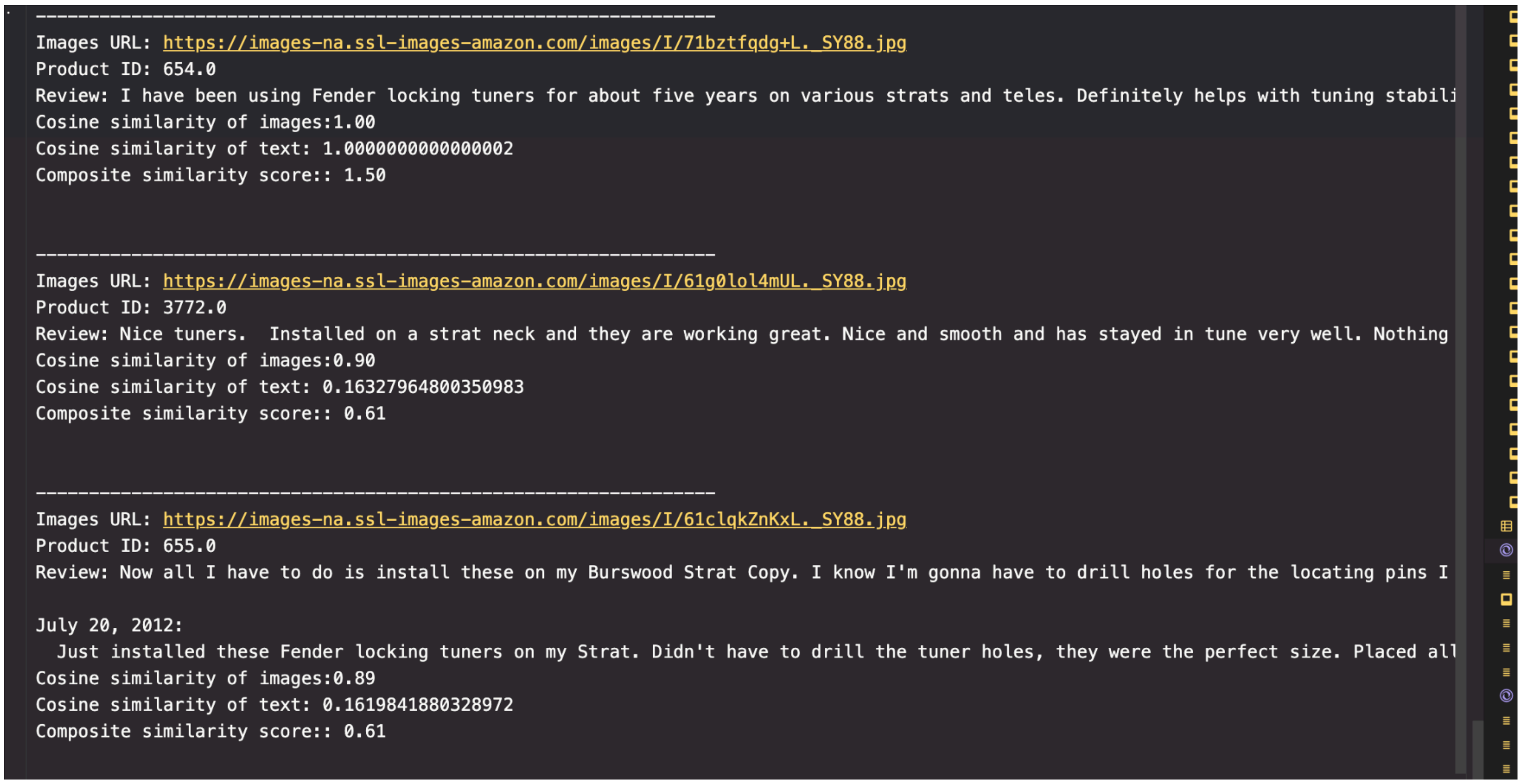
Q-4)
This code snippet (`search_combined_query`) implements a search function that retrieves products based on a combined text query and image similarity:

1. Retrieving Most Similar Image:
   - Extracts features for the query image from the `features` dictionary.
   - Iterates through products in the data.
   - For each product, calculate cosine similarity between the query image features and features of each image associated with the product.
   - Sorts image similarities within each product in descending order (most similar image first).

2. Calculating Combined Similarity:
   - Combines the image similarity score with the text similarity score using a weighted average (weights set to 0.5 each).

3. Presenting Results:
   Sorts products by their combined similarity scores (highest to lowest).
   Prints details for the top `k` results (configurable parameter):

Overall, this function enhances the product search by incorporating image similarity alongside text matching.



```
-----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/71bztfqdg+L._SY88.jpg
Product ID: 654.0
Review: I have been using Fender locking tuners for about five years on various strats and teles. Definitely helps with tuning stabili
Cosine similarity of images:1.00
Cosine similarity of text: 1.0000000000000002
Composite similarity score:: 1.50


-----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/61g0lol4mUL._SY88.jpg
Product ID: 3772.0
Review: Nice tuners.  Installed on a strat neck and they are working great. Nice and smooth and has stayed in tune very well. Nothing
Cosine similarity of images:0.90
Cosine similarity of text: 0.16327964800350983
Composite similarity score:: 0.61


-----------------------------------------------------------------
Images URL: https://images-na.ssl-images-amazon.com/images/I/61clqkZnKxL._SY88.jpg
Product ID: 655.0
Review: Now all I have to do is install these on my Burswood Strat Copy. I know I'm gonna have to drill holes for the locating pins I

July 20, 2012:
  Just installed these Fender locking tuners on my Strat. Didn't have to drill the tuner holes, they were the perfect size. Placed all
Cosine similarity of images:0.89
Cosine similarity of text: 0.1619841880328972
Composite similarity score:: 0.61
```

Q-5)

b. Text-based vs. Combined Retrieval: Similarity Score Comparison

It's difficult to definitively say which retrieval technique (text-based or combined) yields a better similarity score without evaluating them on a specific dataset and task. Here's a breakdown of factors to consider:

Text-based Retrieval:

- Advantages:
    - Efficient for text-heavy searches where textual descriptions are the primary source of information.
    - May be more interpretable, as text similarity scores directly reflect the alignment between query and product descriptions.
- Limitations:

  Might miss relevant products if images contain significant information not captured in the text.

Combined Retrieval:

- Advantages:
    - Potentially retrieves more relevant products by incorporating image content alongside textual information.
    - Can be beneficial for searches where visual aspects are crucial (e.g., fashion, furniture).
- Limitations:
    - Requires pre-computed image features, which can be computationally expensive to generate.
    - Image similarity scores might be less interpretable compared to text similarity.
    - The chosen weights for combining text and image similarity scores can impact the final results.

In our case image-based retrieval give best scores, because tf-idf isn't a very intelligent form of retrieval.

c. Challenges and Improvements in Retrieval Process

Challenges:

- **Data Quality:** Inaccurate or incomplete product descriptions and noisy image features can hinder retrieval accuracy.
- **Semantic Gap:** Bridging the gap between the low-level image features and high-level concepts in text descriptions remains a challenge.
- **Weighting Text and Image Similarity:** Determining the optimal weights for combining text and image similarity scores requires careful consideration based on the specific task and dataset.

Potential Improvements:

- **Text Preprocessing:** Techniques like named entity recognition or part-of-speech tagging can enhance text understanding.
- **Image Feature Extraction:** Exploring more advanced image feature extraction methods that capture semantic information better.
- **Learning-based Ranking:** Utilizing machine learning models to learn optimal weights for combining text and image similarity scores based on training data.
- **Relevance Feedback:** Allowing users to provide feedback on retrieved results to improve the ranking model over time.

By addressing these challenges and incorporating potential improvements, the retrieval process can be refined to deliver more accurate and relevant search results.