GLUU WORLDWIDE
DATA SCIENCE AND AI IN RESTORATION SERVICES

Student Team:
Yuan-Zhi Cai (3886716)
Anoop Varghese (3962557)
Sweta Karmacharya (3938458)
Mukesh Lakshmana (3859256)

Industrial Supervisor:
Zane Aburaneh

Academic Supervisor:
Azadeh Alavi

Table of Contents

ABSTRACT

This report examines the application of Artificial Intelligence (AI) in enhancing the restoration services at Gluu Inc., a company specializing in the repair and preservation of vintage artifacts. The project focused on integrating a CLIP (Contrastive Language-Image Pre-Training) model-based image recognition system to identify damage and automate task allocation efficiently. CLIP, which allows for the recognition of objects that have not been explicitly labeled in the training data, was chosen to address the variability and uniqueness of the vintage items handled by Gluu Inc. Utilizing a pre-trained CLIP model, the project developed a system capable of accurately analyzing a wide range of images to facilitate automated task assignments. This paper outlines the project's technological approach, implementation processes, and the outcomes of integrating AI to improve efficiency and quality in restoration services.

The adoption of CLIP is particularly significant given the diverse and often rare nature of vintage artifacts. Traditional machine learning models would require extensive labeled datasets for each specific type of damage, which is not feasible for unique and historical items. By leveraging CLIP, the AI system can generalize from existing data to identify new and unseen forms of damage, making it a robust tool for restoration experts. This capability not only enhances the initial assessment process but also ensures that the allocation of restoration tasks is precise and efficient.

In addition to improving operational efficiency, the project also aimed to preserve the integrity and authenticity of vintage artifacts by minimizing human error in the preliminary stages of restoration. The successful implementation of this AI-driven approach underscores its potential to transform restoration practices, offering a scalable solution that maintains high standards of quality and detail. Future work will explore expanding the dataset and refining the AI model to cover a broader range of artifacts, further solidifying AI's role in the preservation of cultural heritage.

INTRODUCTION

Partner/Client: Zane Aburaneh

INDUSTRY BACKGROUND

Gluu Inc. is a pioneering provider of specialized services dedicated to the restoration and preservation of delicate, hand-made, vintage artifacts. Renowned for its commitment to quality, expertise, and innovation, Gluu Inc. leverages cutting-edge technology alongside traditional craftsmanship to maintain its status as a trailblazer in the industry. Operating primarily within the fashion industry, Gluu Inc. focuses on resource recovery and the localization of repairs, aiming to minimize environmental impact and inefficiencies associated with cross-border shipping.

Gluu Inc. operates in a niche market where the restoration of vintage items requires not only skilled craftsmanship but also a deep understanding of historical preservation techniques. The company's dual approach—merging artisanal skills with modern technological advancements—allows it to restore items with high precision and care, ensuring the longevity and integrity of each piece.

In an era where sustainability and environmental consciousness are increasingly vital, Gluu Inc. addresses these challenges by integrating advanced data science into its restoration processes. The company's innovative approach not only enhances the efficiency of item restoration but also significantly reduces the carbon footprint by localizing repair efforts. Through a sophisticated digital platform, Gluu Inc. connects local craftspeople with restoration opportunities, thus fostering a sustainable and economically viable repair ecosystem.

The platform's ability to harness data science for item restoration highlights Gluu Inc.'s commitment to innovation. This initiative not only mitigates the environmental impact of international shipping but also revitalizes local craftsmanship by providing artisans with consistent work opportunities. This localized model promotes sustainability and supports economic growth within the community, aligning with broader global goals for sustainable development.

PROJECT MOTIVATION AND DATASET SELECTION

Project Motivation

The Challenge of Variability and Uniqueness

The restoration of vintage artifacts presents a unique set of challenges primarily due to the inherent variability and uniqueness of each item. Unlike mass-produced modern goods, vintage artifacts often possess distinctive characteristics that stem from their individual histories, manufacturing processes, and materials used. This variability can include differences in wear and tear patterns, material degradation, color fading, and structural integrity. As a result, creating comprehensive training datasets for machine learning models becomes exceedingly difficult. Each vintage item may exhibit unique forms of damage that are not present in other items, making it nearly impossible to cover all possible variations with traditional labeled datasets.

Traditional Machine Learning Limitations

Traditional machine learning models rely heavily on large, well-labeled datasets to learn to identify specific features and categories. In the context of vintage artifact restoration, this would require extensive labeling of diverse damage types across numerous items, which is impractical. The time and expertise required to label such datasets accurately are considerable, and the likelihood of encountering entirely new forms of damage in future items remains high.

The Need for an Advanced Solution

To address these challenges, our project integrates a CLIP (Contrastive Language-Image Pre-Training) based image recognition system. CLIP is an innovative model developed by OpenAI that leverages a unique approach to learning from image-text pairs. By training on a vast array of images and their corresponding textual descriptions, CLIP can recognize and understand objects and concepts it has not explicitly seen before. This makes it particularly suited to the diverse and unique nature of vintage items handled by Gluu Inc. CLIP's ability to generalize from existing data allows it to identify damage and automate task allocation without the need for extensive labeled datasets.

Data Selection

Limitations of Proprietary Datasets

Given the niche and specialized nature of Gluu Inc.'s work, there is a notable shortage of proprietary datasets specifically tailored to vintage artifact restoration. The proprietary data available is limited in scope and variety, which poses a significant barrier to training an effective machine learning model. Therefore, to develop a robust AI system, it was essential to identify and utilize external datasets that could simulate the conditions and characteristics of vintage artifacts.

We found the CLIP model from OpenAI more suitable for the task . Pre-trained CLIP model trained on vast and diverse set of images and text gives prediction more in line with the requirement by defining output classes as per requirement .This capability is crucial for handling the diverse range of artifacts processed by Gluu Inc. dataset for developing the AI model .Gluu Inc. can also use the same model for predicting other artifacts and materials like shoes ,furniture etc…

PROBLEM DEFINITION

Operational Challenges at Gluu Inc.:

As a provider of high-demand restoration services, Gluu Inc. faces the critical challenge of efficiently managing a large volume of restoration requests. The increasing popularity of vintage and pre-owned shoes has led to a surge in demand for restoration services. This heightened demand has resulted in bottlenecks, particularly in the allocation of restoration tasks to skilled craftspeople.

Current Manual Processes:

The current manual process for assessing the condition of each shoe and assigning restoration tasks involves several labor-intensive steps:
- Visual Inspection: Craftspeople must individually inspect each shoe to assess its condition, identify damages, and determine the necessary restoration procedures.
- Expert Judgment: The process relies heavily on the subjective judgment of experienced craftspeople, which can lead to variability in assessments.
- Task Assignment: Based on the assessment, tasks are manually allocated to craftspeople, taking into account their specific skills and current workload.

This manual approach is not only time-consuming but also prone to human error, resulting in delays and inconsistencies in service delivery. As the customer base continues to grow, it becomes increasingly challenging to ensure that each restoration request is matched with the most suitable craftspeople based on their expertise and availability.

Technological Integration to Address Challenges:

To address these challenges, the integration of advanced AI technologies can significantly enhance operational efficiency. The key components of this technological integration include:

Automated Image Recognition:
- Data Collection: High-resolution images of each shoe are collected and processed.
- Feature Extraction: Advanced image recognition algorithms extract key features from the images, such as wear patterns, discoloration, and structural damages.
- Damage Classification: The AI system classifies the type and extent of damage using machine learning models trained on extensive datasets.

CLIP for Damage Assessment:

- Model Training: Traditional supervised learning models require large, labeled datasets for training. CLIP, however, enables the AI system to recognize and categorize damages it has never seen during training by using semantic descriptions and attribute-based learning.
- Generalization: This approach allows the model to generalize its knowledge to new, unseen types of damage, making it particularly suitable for the diverse range of vintage and pre-owned shoes handled by Gluu Inc.

Automated Task Allocation:
- Skill Matching: The AI system matches restoration tasks with the most suitable craftspeople based on their skill profiles and past performance data from the output predicted by the CLIP Model

- Availability Management: The system dynamically adjusts task assignments based on real-time availability, ensuring an optimal distribution of workload among craftspeople.This requirement is planned for the future .

Benefits of AI Integration:

Increased Efficiency: Automating the assessment and task allocation processes reduces the time and manpower required, significantly increasing operational efficiency.
Consistency and Accuracy: AI-driven assessments are consistent and less prone to errors compared to manual evaluations, ensuring high-quality service delivery.
Scalability: The system can easily scale to accommodate growing customer demands without a proportional increase in manpower, providing a sustainable solution for long-term growth.
Enhanced Customer Satisfaction: Faster turnaround times and consistent restoration quality lead to higher customer satisfaction and loyalty.

DATASET

The dataset utilized for this project is meticulously tailored for an online shopping scenario, where users often need to scrutinize minute visual distinctions between similar items. This requirement is particularly relevant in differentiating between subtle variations of products, such as two pairs of similar men's running shoes, rather than more apparent distinctions like those between women's high heels and men's slippers.

Dataset Composition:

The dataset used in training CLIP includes various sources such as:

1. **Image-Text Pairs**: Images and their corresponding textual descriptions or captions sourced from diverse datasets like COCO (Common Objects in Context), Conceptual Captions, and more.
2. **Internet Data**: CLIP leverages large-scale internet data to gather a wide array of images and their associated textual descriptions from websites, blogs, social media platforms, and other online sources.
3. **Multilingual Sources**: To enable CLIP to understand text in multiple languages, the dataset comprises image-text pairs in different languages, allowing the model to learn cross-lingual associations.
4. **Balanced Representation**: Efforts are made to ensure that the dataset encompasses a balanced representation of various categories, concepts, and contexts to avoid biases and to enable the model to generalize well across different domains.
5. **Preprocessing and Cleaning**: The dataset undergoes preprocessing and cleaning to remove noise, ensure consistency, and enhance the quality of the training data.

By training on such a diverse and extensive dataset, CLIP learns to associate images with corresponding textual descriptions in a way that enables it to perform a wide range of tasks, including image

Feature Extraction:

GIST Features: GIST descriptors capture the spatial structure of the scene, providing a holistic representation of the image. These features are particularly useful for distinguishing between different types of shoes by capturing the overall shape and layout of the footwear.

LAB Color Features: The LAB color space is designed to approximate human vision, with three channels: L for lightness and a and b for the color-opponent dimensions. LAB color features are effective in capturing subtle color variations, which are essential for distinguishing between similar shoe designs.

Application of Dataset in Restoration:

Resource Recovery: In the context of restoration, resource recovery refers to the extraction of useful materials from items that would otherwise be considered waste. This practice supports sustainability by minimizing waste and repurposing materials, ensuring that valuable resources are not discarded unnecessarily.

Localization of Repairs: Performing repairs in the same geographical area as the customer significantly reduces the carbon footprint and supports local economies. This practice not only promotes environmental sustainability but also enhances the efficiency of the repair process by reducing shipping times and costs.

Technological Components:

Image Recognition: AI-driven image recognition technology is employed to identify and classify objects within the images. In the context of restoration, this technology automates the assessment of restoration needs by analyzing the images to detect damages such as scuffs, tears, and discoloration. Advanced convolutional neural networks (CNNs) are typically used for this purpose, given their ability to learn hierarchical features from images.

CLIP (Contrastive Language-Image Pre-training): CLIP is a powerful neural network model developed by OpenAI. It's designed to understand images in the context of natural language descriptions, allowing it to perform a variety of tasks such as image classification, object detection, and more, based on textual prompts.

Dataset Utilization for AI Model Training:

- Training Process

Pre Training Data: CLIP is pretrained on a large dataset consisting of pairs of images and associated text. This dataset is typically diverse and covers a wide range of concepts, objects, scenes, and contexts. OpenAI used a mixture of images and their captions from the internet, including from sources like books and articles, to create such a dataset.

Contrastive Learning: CLIP utilizes a contrastive learning approach during pretraining. In contrastive learning, the model learns by comparing different pairs of data points and adjusting its parameters to make similar pairs more similar and dissimilar pairs more dissimilar. In the case of CLIP, it learns to associate images and corresponding textual descriptions while also distinguishing between different pairs of images and texts.

Vision and Language Encoders: CLIP consists of two main components: a vision encoder and a text encoder. The vision encoder processes images to extract meaningful visual features, while the text encoder processes textual descriptions to extract semantic information. Both encoders are pre trained using contrastive learning.

Joint Training: After pretraining the vision and text encoders separately, CLIP undergoes joint training where the model learns to align visual and textual representations. During this phase, the model is presented with pairs of images and associated text and learns to map them into a shared embedding space where similar images and texts are placed close to each other.

Fine-tuning: Once the model has been pre trained and jointly trained, it can be fine-tuned on specific downstream tasks. Fine-tuning involves training the model on a task-specific dataset to adapt its parameters for the particular task at hand. For example, CLIP can be fine-tuned for tasks like image classification, object detection, text-based image retrieval, and more.

Evaluation Metrics: The performance of the AI model is evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics provide a comprehensive understanding of the model's capability to correctly identify and classify damages, ensuring reliability in real-world applications.

Scalability and Adaptation:

Scalability: The dataset and the AI models developed can be scaled to accommodate other types of vintage artifacts beyond footwear. By fine-tuning the models with additional datasets containing images of different artifact types, the system can be adapted to handle a broader range of restoration tasks.

Continuous Improvement: The system is designed for continuous learning, where it can be updated with new data and feedback from restoration experts. This iterative approach ensures that the model remains accurate and effective as it encounters new and diverse restoration scenarios.

PROJECT AIMS

The primary objective of this project is to develop and integrate an advanced AI-driven image recognition system utilizing the CLIP model into Gluu Inc.'s operational platform. The project aims to address several key challenges in the restoration of vintage artifacts by leveraging state-of-the-art artificial intelligence and machine learning techniques.

- **Enhancing Damage Identification:**

Automated Visual Assessment: The AI system will employ sophisticated convolutional neural networks (CNNs) to analyze high-resolution images of vintage items. These networks are capable of detecting and classifying various types of damage, such as scuffs, tears, discoloration, and structural wear.

Feature Extraction: By extracting intricate visual features from images, such as texture, edge patterns, and color distribution, the AI can provide a detailed assessment of each item's condition. Advanced feature extraction techniques, including Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG), will be employed to ensure robustness against variations in lighting and orientation.

Damage Classification: The system will use multi-label classification techniques to categorize different types of damage. This approach allows for the simultaneous identification of multiple damage types within a single image, providing a comprehensive assessment.

Utilizing the CLIP Model for Enhanced Recognition and Classification:

Semantic Embeddings: The CLIP model leverages semantic embeddings to enable the understanding and classification of items. By mapping both visual features and semantic descriptions into a shared space, the CLIP model can generalize its knowledge to categorize a wide variety of items. This approach allows the AI to comprehend and distinguish features even in artifacts it has not explicitly seen during

its training phase. The shared embedding space facilitates the identification of damage and other characteristics based on the learned relationships between images and textual descriptions.

Attribute-Based Classification: The system employs attribute-based classification to recognize and categorize unique vintage items. Attributes such as material type, era, design style, and specific damage characteristics are utilized to ensure accurate identification. The CLIP model's ability to interpret these attributes from both visual and textual data enhances its capacity to manage the diverse and intricate features of vintage artifacts. This attribute-focused approach allows for precise classification, accommodating the unique variations found in vintage items.

Reduction of Data Labeling Requirements: A significant advantage of using the CLIP model is the reduction in the need for exhaustive labeled data. In the domain of vintage artifacts, where labeled datasets are often scarce, this is particularly beneficial. The CLIP model can leverage existing labeled datasets to learn various attributes and apply this knowledge to new items without requiring additional labeling efforts. This capability not only speeds up the training process but also ensures the model remains adaptable and effective in handling the diverse range of vintage artifacts encountered by Gluu Inc.

Expanding to a Broader Array of Artifacts:

Modular Architecture: The AI system will be designed with a modular architecture, allowing it to be easily extended to handle various types of artifacts beyond shoes. Each module will be responsible for specific tasks such as image preprocessing, feature extraction, damage assessment, and task allocation.

Incremental Learning: The system will support incremental learning, enabling it to continuously improve as new data becomes available. This approach will allow the model to adapt to new types of artifacts and restoration challenges over time.

Scalability: Scalability is a critical aspect of the system design. The architecture will support horizontal scaling, allowing additional computational resources to be added as the volume of restoration requests grows. This ensures that the system can handle increased demand without degradation in performance.

Integration into Gluu Inc.'s Operational Platform:

Seamless Integration: The AI system will be integrated into Gluu Inc.'s existing digital platform. This integration will involve developing APIs and interfaces that allow the AI system to communicate with other components of the platform, such as the customer portal, inventory management system, and artisan scheduling tools.

User-Friendly Interface: A user-friendly interface will be developed for both customers and artisans. Customers will be able to upload images of items needing restoration and receive instant damage assessments. Artisans will receive detailed task descriptions and recommendations for restoration techniques.

Data Security and Privacy: Ensuring data security and privacy is paramount. The system will implement robust security measures, including encryption, access controls, and compliance with data protection regulations such as GDPR.

FINAL DELIVERABLES

Figure 1 illustrates the Gluu project workflow, which involves three inputs: customer image, restoration skill level, and shoe type. The customer image is fed into the CLIP model for classification into three categories: shoe type, material, and damaged areas. Upon receiving the results, the identified damage issues are matched with the user's restoration skill level. When the AI prediction aligns with the user's input, the program displays accurate information and forwards the shoe type, material, and damage details to the craftspeople. This streamlined process reduces the time required to assign tasks to craftspeople effectively.

However, mismatches may occur due to two potential reasons: user input errors or incorrect predictions by the CLIP model. Nonetheless, the information provided to the craftspeople includes both the user's input and the CLIP model's predictions, maximizing the details given to increase matching accuracy and further reduce processing time.

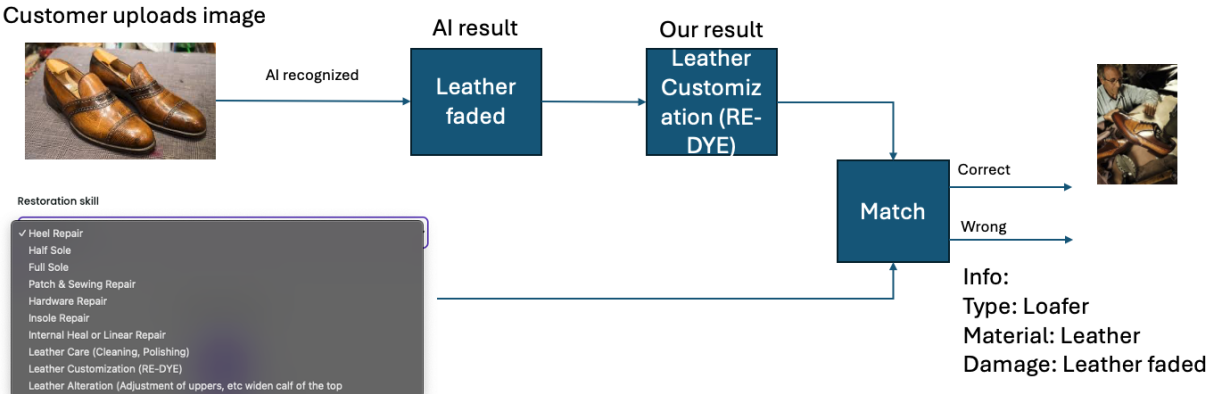

Figure 1: Example of project processing

The project aims to deliver an innovative AI system that includes:

- A robust image recognition system using zero-shot learning, capable of identifying a wide range of damage and restoration needs across various types of shoes and potentially other artifacts.

- Craftsperson profiling system that catalogs the expertise, experience, and availability of each artisan, facilitating intelligent task allocation.

RESULTS, MODEL EVALUATION AND ANALYSIS

CLIP Model:
Figure 2 illustrates the comparison between zero-shot CLIP models and traditional ImageNet models under natural distribution shifts demonstrates significant improvements in effective robustness, reducing the accuracy gap by up to 75%. These findings highlight the potential for zero-shot models to achieve much greater

robustness. However, these results do not necessarily imply that supervised learning on ImageNet alone causes the observed robustness gap. Other factors inherent to CLIP, such as its extensive and diverse pre-training dataset or the use of natural language supervision, could also contribute to the enhanced robustness.
It's important to note that zero-shot models can still exploit coincidental correlations shared between the datasets used during pre-training and evaluation. To explore this further, we conducted initial experiments to gauge how CLIP models' performance changes when adapted to the ImageNet distribution using L2 regularized logistic regression classifiers trained on CLIP features from the ImageNet training set.

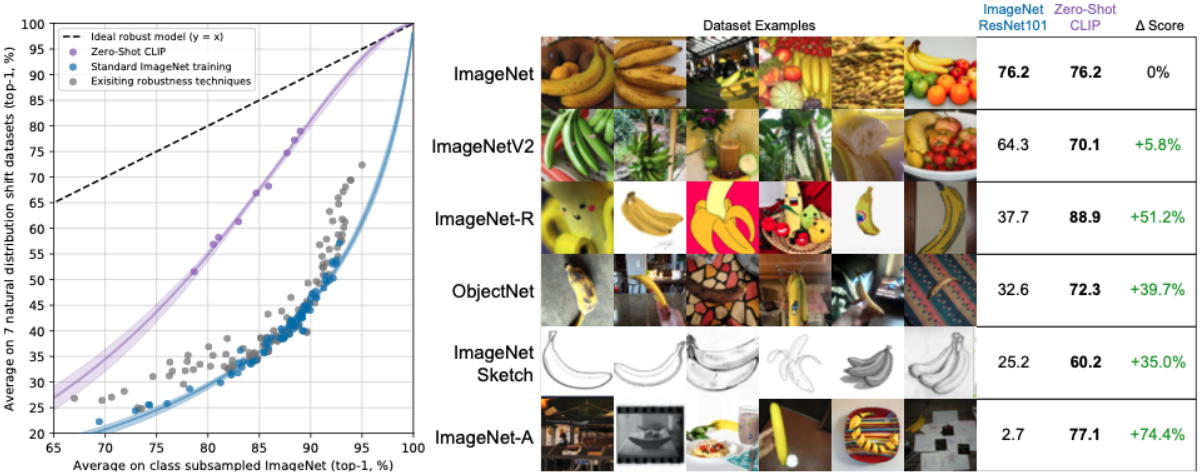

Figure 2: Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models.

Category result:

In this case in Figure 3, there is 100% accuracy for this case, the highest score will be the category output.

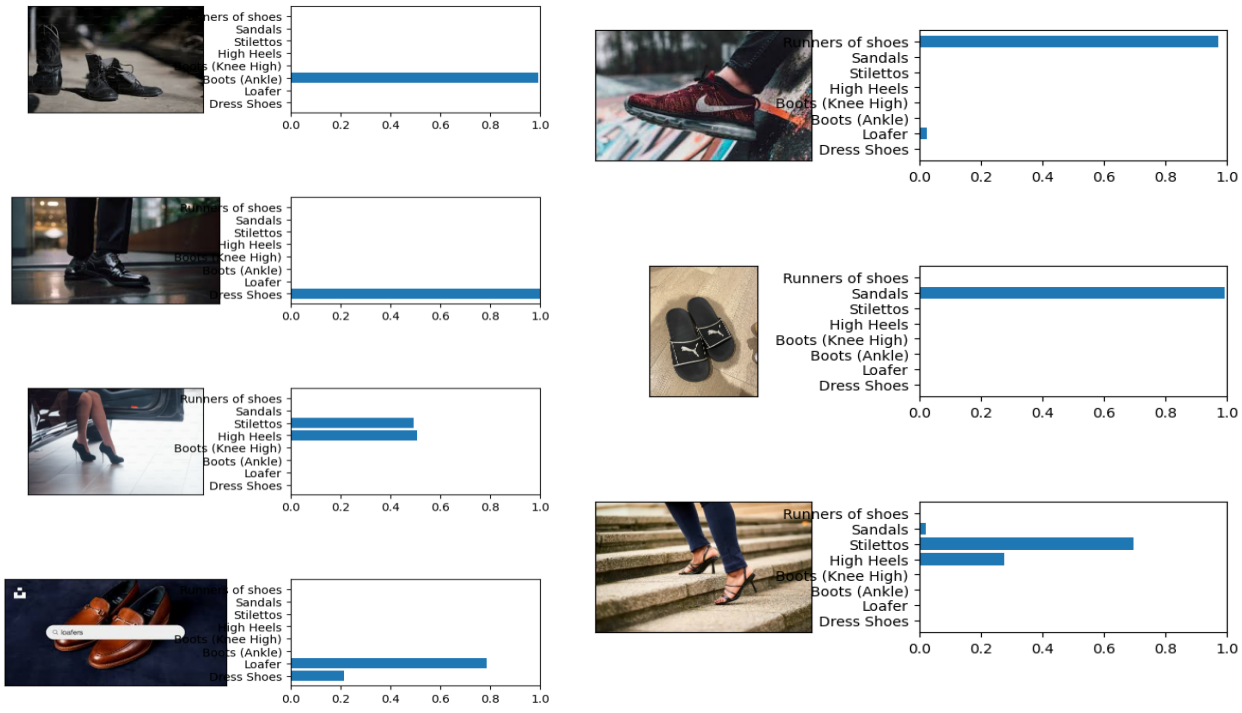

Figure 3: The result of category

Material result:

In this case, accuracy is not perfect, and the images are unlabelled. The information provided on materials serves mainly to enhance craftspeople's understanding without

significantly expediting the matching process between customers and craftspeople, as depicted in Figure 4 below.

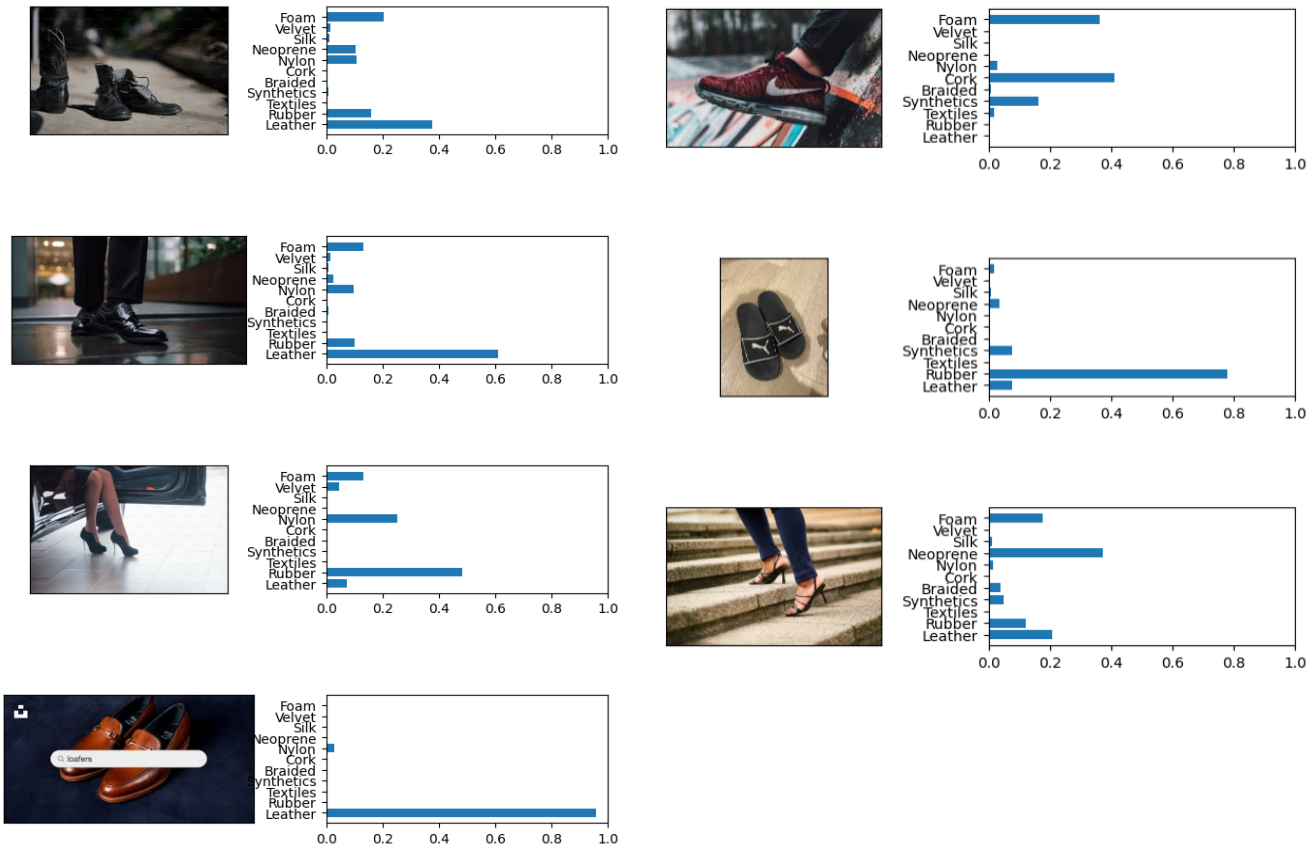

Figure 4: The result of material

Damage result:

In Figure 5 of this scenario, the crucial aspect is its ability to align with user input and direct tasks to craftspeople. However, accurately predicting shoe damage is highly challenging due to various factors, including the angle of the picture, which can lead the model to make incorrect predictions.

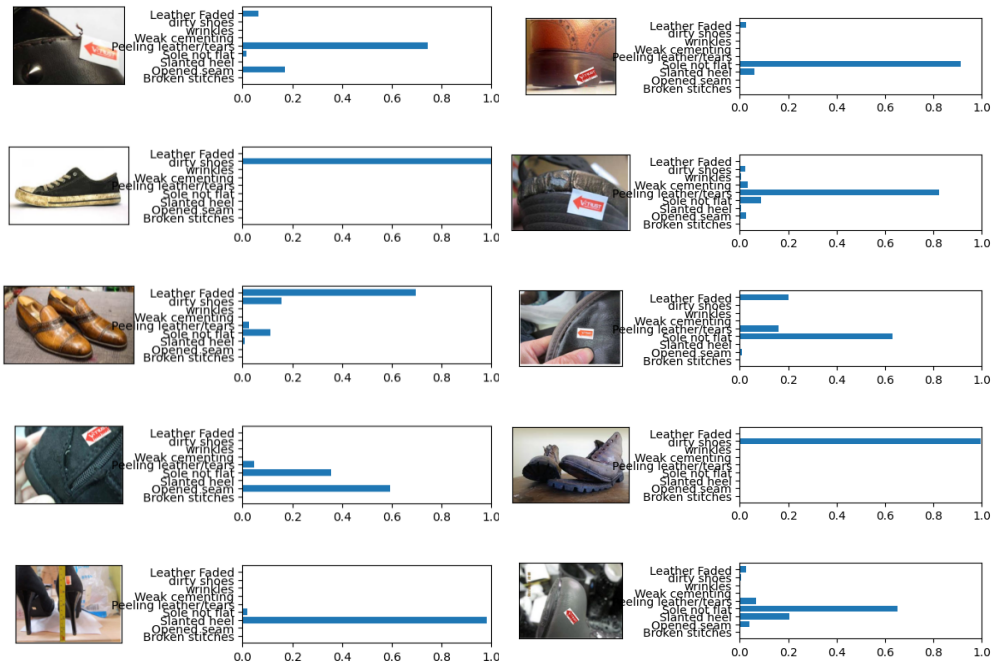

Figure 5: The result of damage

Discussion - Challenges

Integrating zero-shot learning into the restoration process at Gluu Inc. presents several unique challenges:

1. Model Generalization: Zero-shot learning models are designed to handle items they have never seen before by leveraging general descriptions or attributes. However, ensuring these models can accurately generalize to the vast array of unique and diverse vintage artifacts Gluu Inc. handles requires sophisticated modeling and a deep understanding of both AI technology and artifact characteristics.
2. Data Annotation: While zero-shot learning reduces the need for extensive labelled datasets, it relies heavily on rich attribute descriptions and accurate metadata. Developing a comprehensive attribute system that effectively captures the nuances of vintage artifacts is both time-consuming and complex.
3. Integration with Artisan Skills: Merging advanced AI technology with traditional craftsmanship practices is challenging. Artisans must adapt to new workflows that incorporate AI recommendations, which may initially disrupt established restoration processes.
4. Technology Acceptance: There might be resistance from craftspeople who are accustomed to traditional methods of restoration. Encouraging adoption and demonstrating the benefits of AI-supported workflows are critical for successful integration.

FUTURE DIRECTION:

1. **Enhanced Attribute Encoding:** Improving the way attributes are encoded into the zero-shot learning model could enhance its accuracy and applicability to a broader range of artifacts. This might involve leveraging natural language processing to better understand and utilize descriptive metadata.
2. **Broadening AI Applications:** Expanding the zero-shot learning framework to include other restoration categories beyond shoes could significantly increase Gluu Inc.'s market reach and service offerings. Each new category will enrich the model's understanding and capabilities.
3. **Feedback Loops:** Integrating feedback mechanisms that allow artisans to provide input on AI performance can refine and improve the model over time. This feedback can help adjust the model to better align with real-world restoration needs and craftspeople’s expertise.
4. **Global Expansion:** As the model becomes more robust, exploring opportunities for international expansion could provide a larger dataset and a broader range of artifact types, further enhancing the model’s versatility and accuracy.
5. **Sustainability Impact Measurement:** Developing metrics to quantify the environmental impact of AI-driven restoration processes could reinforce Gluu Inc.'s commitment to sustainability and provide valuable data for continuous improvement.

6. **Transfer Learning:** Transfer learning with CLIP for image classification involves utilizing its pre-trained capabilities in both visual and textual domains to classify images into predefined categories. To begin, prepare a labeled dataset containing images corresponding to their respective classes, ensuring it includes training, validation, and potentially test sets. Initiate with the pre-trained CLIP model, which has been trained on a diverse dataset of image-text pairs. Customize the model by modifying or adding final classification layers specific to your classification task. Define a suitable loss function, typically categorical cross-entropy, and select an optimizer like Adam or SGD for training. Iterate through training epochs, adjusting batch size and learning rate based on performance monitoring on the validation set to prevent overfitting. Evaluate the fine-tuned model on the validation set, refining hyperparameters or model architecture as needed. Finally, deploy the fine-tuned CLIP model for inference, benefiting from its efficiency, generalization capabilities, and versatility across various image classification tasks.

Addressing these challenges and pursuing the outlined future directions will enable Gluu Inc. to fully leverage zero-shot learning, optimizing both the quality and efficiency of its restoration services while scaling its operations to meet global demands.

Appendix – Code for Gluu project, original IDE is Google Colab.

```
# -- coding: utf-8 --
"""Gluu.ipynb

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1jy2y3IU8y51I3Jr0RvQcYuYs7YGDjTYJ
"""

# mount your Google Drive files
from google.colab import drive
drive.mount('/content/drive')

# Transformers provides thousands of pretrained models to perform tasks on different modalities such as text, vision, and audio.
!pip install transformers

# Import nessessary libraries
from transformers import CLIPProcessor, CLIPModel
import torch

# get CLIP pre-trained model
model = CLIPModel.from_pretrained("laion/CLIP-ViT-H-14-laion2B-s32B-b79K")
processor = CLIPProcessor.from_pretrained("laion/CLIP-ViT-H-14-laion2B-s32B-b79K") # a wrapper for input data preprocessing. It includes both encoding the text using tokenizer and preparing the images.

# For image displaying
!pip install Pillow

"""### Types test
Types from website:
1. Dress Shoes
2. Loafer
3. Boots (Ankle & Knee High)
4. High Heels & Stilettos
5. Sandals
6. Runners

Types for classification from AI Model
1. Dress Shoes
2. Loafer
3. Boots (Ankle)
4. Boots (Knee High)
5. High Heels & Stilettos
6. Sandals
7. Runners
"""

# Import libraries
```

```
from PIL import Image
from transformers import pipeline
import os

from PIL import Image

def image_grid(imgs, cols):
    # Calculate the number of rows needed
    rows = (len(imgs) + cols - 1) // cols

    # Get the width and height of the first image (assuming all images are the same size)
    w, h = imgs[0].size

    # Create a new blank image with the appropriate size to fit all images in the grid
    grid = Image.new('RGB', size=(colsw, rowsh))

    # Iterate over the list of images and paste each one into the grid
    for i, img in enumerate(imgs):
        # Calculate the position where the current image should be pasted
        x = (i % cols)  w  # Column position
        y = (i // cols)  h  # Row position
        # Paste the image at the calculated position
        grid.paste(img, box=(x, y))

    # Return the final grid image
    return grid

# Local directory containing the images
image_folder = '/content/drive/MyDrive/Colab_Notebooks/Gluu/datasets/test/background'

# List of image filenames in the local directory
image_filenames = [
    'Boots.jpeg',
    'Dress_shoes.jpeg',
    'High_Heels.jpeg',
    'Loafer.jpeg',
    'Runners.jpeg',
    'Sandals.jpg',
    'Stilettos.jpeg'
]

# Append the images together
images = []
for filename in image_filenames:
    img_path = os.path.join(image_folder, filename)
    images.append(Image.open(img_path))

grid = image_grid(images, cols=3)
grid.show()  # Use .show() for displaying the image grid

# List of class names for different types of shoes
classes = [
    'Dress Shoes', 'Loafer', 'Boots (Ankle)', 'Boots (Knee High)',
    'High Heels', 'Stilettos', 'Sandals', 'Runners of shoes'
]
```

```
# Process the text (class names) and images into tensors suitable for the model
inputs = processor(
    text=classes,        # List of class names
    images=images,       # List of images
    return_tensors="pt", # Return PyTorch tensors
    padding=True,        # Pad the inputs to the same length
    do_convert_rgb=False # Assume images are already in RGB format
)

# Pass the processed inputs through the model to get the outputs
outputs = model(inputs)

# Extract the logits for image-text similarity
logits_per_image = outputs.logits_per_image

# Apply softmax to the logits to get probabilities for each class
probs = logits_per_image.softmax(dim=1)

# 'probs' now contains the probabilities of each image belonging to each class

import matplotlib.pyplot as plt

# Create a figure with specified size (width, height)
fig = plt.figure(figsize=(8, 20))

# Loop over each image and its corresponding probabilities
for idx in range(len(images)):

    # Display the original image
    fig.add_subplot(len(images), 2, 2(idx+1)-1)
    plt.imshow(images[idx])  # Show the image
    plt.xticks([])  # Remove x-axis ticks
    plt.yticks([])  # Remove y-axis ticks

    # Display the probabilities as a horizontal bar chart
    fig.add_subplot(len(images), 2, 2(idx+1))
    plt.barh(range(len(probs[0].detach().numpy())), probs[idx].detach().numpy(),
tick_label=classes)
    plt.xlim(0, 1.0)  # Set x-axis limits to range from 0 to 1

    # Adjust subplot parameters for spacing
    plt.subplots_adjust(left=0.1,  # Adjust space to the left
                        bottom=0.1,  # Adjust space at the bottom
                        right=0.9,  # Adjust space to the right
                        top=0.9,  # Adjust space at the top
                        wspace=0.2,  # Adjust horizontal space between subplots
                        hspace=0.8)  # Adjust vertical space between subplots

# Display the figure with all subplots
plt.show()

"""### Materials test"""

# List of material classes
```

```
classes = [
    'Leather', 'Rubber', 'Textiles', 'Synthetics', 'Braided',
    'Cork', 'Nylon', 'Neoprene', 'Silk', 'Velvet', 'Foam'
]

# Process the text (material classes) and images into tensors suitable for the model
inputs = processor(
    text=classes,        # List of material classes
    images=images,       # List of images to classify
    return_tensors="pt", # Convert inputs to PyTorch tensors
    padding=True,        # Pad the inputs to the same length
    do_convert_rgb=False # Assume images are already in RGB format
)

# Pass the processed inputs through the model to get the outputs
outputs = model(inputs)

# Extract the logits for image-text similarity from the model outputs
logits_per_image = outputs.logits_per_image

# Apply softmax to the logits to get probabilities for each material class
probs = logits_per_image.softmax(dim=1)

# Import Matplotlib for visualization
import matplotlib.pyplot as plt

# Create a figure for plotting with specified size (width, height)
fig = plt.figure(figsize=(8, 20))

# Loop over each image and its corresponding probabilities
for idx in range(len(images)):
    # Display the original image
    fig.add_subplot(len(images), 2, 2(idx+1)-1)  # Add subplot for the image
    plt.imshow(images[idx])  # Show the image
    plt.xticks([])  # Remove x-axis ticks
    plt.yticks([])  # Remove y-axis ticks

    # Display the probabilities as a horizontal bar chart
    fig.add_subplot(len(images), 2, 2(idx+1))  # Add subplot for the bar chart
    plt.barh(range(len(probs[0].detach().numpy())), probs[idx].detach().numpy(),
tick_label=classes)  # Create a horizontal bar chart
    plt.xlim(0, 1.0)  # Set x-axis limits to range from 0 to 1

    # Adjust subplot parameters for spacing
    plt.subplots_adjust(
        left=0.1,  # Adjust space to the left
        bottom=0.1,  # Adjust space at the bottom
        right=0.9,  # Adjust space to the right
        top=0.9,  # Adjust space at the top
        wspace=0.2,  # Adjust horizontal space between subplots
        hspace=0.8  # Adjust vertical space between subplots
    )

# Display the figure with all subplots
plt.show()
```

```
import matplotlib.pyplot as plt

# Create a figure for plotting with specified size (width, height)
fig = plt.figure(figsize=(8, 20))

# Loop over each image and its corresponding probabilities
for idx in range(len(images)):

    # Display the original image
    fig.add_subplot(len(images), 2, 2(idx+1)-1)  # Add a subplot for the image
    plt.imshow(images[idx])  # Show the image
    plt.xticks([])  # Remove x-axis ticks for a cleaner look
    plt.yticks([])  # Remove y-axis ticks for a cleaner look

    # Display the probabilities as a horizontal bar chart
    fig.add_subplot(len(images), 2, 2(idx+1))  # Add a subplot for the bar chart
    plt.barh(range(len(probs[0].detach().numpy())), probs[idx].detach().numpy(),
tick_label=classes)  # Create a horizontal bar chart
    plt.xlim(0, 1.0)  # Set x-axis limits to range from 0 to 1, as probabilities range between 0 and
1

    # Adjust subplot parameters for spacing
    plt.subplots_adjust(left=0.1,  # Adjust space to the left
                        bottom=0.1,  # Adjust space at the bottom
                        right=0.9,  # Adjust space to the right
                        top=0.9,  # Adjust space at the top
                        wspace=0.2,  # Adjust horizontal space between subplots
                        hspace=0.8)  # Adjust vertical space between subplots

# Display the figure with all subplots
plt.show()

"""### Damage test:
Reference: https://www.v-trust.com/en/blog/top-10-shoe-defects-every-buyer-should-know
"""

# Import necessary libraries
from PIL import Image
from transformers import pipeline
import os

# Local directory containing the images
image_folder = '/content/drive/MyDrive/Colab_Notebooks/Gluu/datasets/test/Damage'

# List of image filenames in the local directory
image_filenames = [
    'Broken_stitches.jpeg',  # Filename of the image showing broken stitches
    'dirty.jpeg',            # Filename of the image showing a dirty item
    'Leather_faded.jpeg',    # Filename of the image showing faded leather
    'Opened_seam.jpeg',      # Filename of the image showing an opened seam
    'Slanted_heel.jpeg',     # Filename of the image showing a slanted heel
    'Sole_not_flat.jpeg',    # Filename of the image showing a sole that is not flat
    'tears.jpeg',            # Filename of the image showing tears
    'Weak_cementing.jpeg',   # Filename of the image showing weak cementing
```

```
    'Weak_cementing1.jpeg',  # Another filename of the image showing weak cementing
    'wrinkles.jpeg',         # Filename of the image showing wrinkles
]

# Initialize an empty list to store the images
images = []
for filename in image_filenames:
    # Construct the full path to the image file
    img_path = os.path.join(image_folder, filename)

    # Open the image file and append it to the images list
    images.append(Image.open(img_path))

# Function to create an image grid (this should be defined or imported earlier in the script)
def image_grid(imgs, cols):
    # Calculate the number of rows required
    rows = (len(imgs) + cols - 1) // cols

    # Get the size of the first image (assuming all images are the same size)
    w, h = imgs[0].size

    # Create a new blank image with a size to fit all images in a grid
    grid = Image.new('RGB', size=(colsw, rowsh))

    # Loop through each image and paste it into the grid
    for i, img in enumerate(imgs):
        # Calculate the position where the image should be pasted
        grid.paste(img, box=(i % cols  w, i // cols  h))

    # Return the combined image grid
    return grid

# Create an image grid with the specified number of columns
grid = image_grid(images, cols=3)

# Display the image grid
grid.show()  # Use .show() for displaying the image grid

# Define the list of classes (labels) corresponding to different types of damage
classes = [
    'Broken stitches',
    'Opened seam',
    'Slanted heel',
    'Sole not flat',
    'Peeling leather/tears',
    'Weak cementing',
    'wrinkles',
    'dirty shoes',
    'Leather Faded'
]

# Process the text (class labels) and images to create the inputs for the model
inputs = processor(
    text=classes,              # List of class labels
    images=images,             # List of images to be classified
```

```
    return_tensors="pt",       # Return the data as PyTorch tensors
    padding=True,              # Pad the inputs to the same length
    do_convert_rgb=False       # Do not convert images to RGB (assuming they are already in
RGB format)
)

# Pass the inputs through the model to get the outputs
outputs = model(inputs)

# Extract the logits for image-text similarity scores
logits_per_image = outputs.logits_per_image  # This is the image-text similarity score

# Apply softmax to the logits to get the probabilities for each class
probs = logits_per_image.softmax(dim=1)  # Take the softmax to get the label probabilities

# Get the highest scoring label for each image
predicted_labels = torch.argmax(probs, dim=1)

import matplotlib.pyplot as plt

# Create a figure for plotting with specified size (width, height)
fig = plt.figure(figsize=(8, 20))

# Iterate over each image and its corresponding probabilities
for idx in range(len(images)):

    # Show the original image
    fig.add_subplot(len(images), 2, 2(idx+1)-1)
    plt.imshow(images[idx])  # Display the image
    plt.xticks([])  # Remove x-axis ticks for a cleaner look
    plt.yticks([])  # Remove y-axis ticks for a cleaner look

    # Show the predicted probabilities as a horizontal bar chart
    fig.add_subplot(len(images), 2, 2(idx+1))
    plt.barh(range(len(probs[0].detach().numpy())), probs[idx].detach().numpy(),
tick_label=classes)  # Create a horizontal bar chart
    plt.xlim(0, 1.0)  # Set x-axis limits to range from 0 to 1, as probabilities range between 0 and
1


    # Adjust subplot parameters for spacing
    plt.subplots_adjust(left=0.1, bottom=0.1, right=0.9, top=0.9, wspace=0.2, hspace=0.8)

# Display the figure with all subplots
plt.show()

"""### Point to craftperson
1. Heel Repair -> Category with Heel & Slanted Heels
2. Half Sole -> Weak cementing
3. Full sole -> Weak cementing
4. Patch & Sweing Repair -> Broken stitches & Opened seam
5. Hardware Repair -> Sole not flat, wrinkles
6. Insole Repair
7. Internet Heal or Linear Repair
8. Leather Care(Cleaning, Polishing) -> Dirty shoes
9. Leather Customization (RE-DYE) -> Leather faded
```

```
10. Leather Alternation (Adjustment of uppers, etc widen calf of the top) -> Peeling leather/tears


"""

# Print the results
for i, label_idx in enumerate(predicted_labels):
    # Print the filename of the image and the predicted class label
    print(f"Image {image_filenames[i]} is predicted to be: {classes[label_idx]}")

    # Provide specific recommendations based on predicted classes
    if str(classes[label_idx]) == 'Broken stitches':
        print("Should be: Patch & Sewing Repair")
    elif str(classes[label_idx]) == 'Opened seam':
        print("Should be: Patch & Sewing Repair")
    elif str(classes[label_idx]) == 'Slanted heel':
        print("Should be: Heel Repair")
    elif str(classes[label_idx]) == 'Sole not flat':
        print("Should be: Hardware Repair")
    elif str(classes[label_idx]) == 'Peeling leather/tears':
        print("Should be: Leather Alternation")
    elif str(classes[label_idx]) == 'Weak cementing':
        print("Should be: Half/Full sole")
    elif str(classes[label_idx]) == 'wrinkles':
        print("Should be: Hardware Repair")
    elif str(classes[label_idx]) == 'dirty shoes':
        print("Should be: Leather Care")
    elif str(classes[label_idx]) == 'Leather Faded':
        print("Should be: Leather Customization RE-DYE")
    else:
        print("There are no specific recommendations for this class")

"""# Demo
- Demo 1: Correct to match
Structure:
1. User input
2. Image import
3. Classify damage of shoes
4. Classify type of shoes
5. Classify material of shoes
6. Comparing input and model output section

- Demo 2: Fail to match
1. User input
2. Image import
3. Classify damage of shoes
4. Classify type of shoes
5. Classify material of shoes
6. Comparing input and model output section

User input
"""

# Customer input
input_customer_type = "Dress Shoes"
```

```
input_customer_restoration = "Leather Customization RE-DYE"

# string variable for storing outcome
damage = ""
material = ""
shoes_type = ""

"""Pre-definned function to display information for backend"""

"""
print_info function:
this is just print the information to craftperson to support faster process.
"""
def print_info(control, predicted_type, predicted_damage, predicted_material,
input_customer_type, input_customer_restoration):
  # print out the information -> match outcome
  if control == "correct":
    print("the information is correct:")
    print("Shoes type:", predicted_type)
    print("Shoes damage:", predicted_damage)
    print("Shoes material:", predicted_material)
  # print out the information -> unmatch outcome
  elif control == "wrong":
    print("The information:")
    print("The customer input:")
    print("type:" + input_customer_type)
    print("damage:" + input_customer_restoration)
    print("The model output:")
    print("Shoes type:", predicted_type)
    print("Shoes damage:", predicted_damage)
    print("Shoes material:", predicted_material)

"""## Image import"""

# Local directory containing the images
image_folder = '/content/drive/MyDrive/Colab_Notebooks/Gluu/datasets/test/Damage'

# List of image filenames in the local directory
image_filenames = [
    'Leather_faded.jpeg',  # Filename of the image showing faded leather
]

images = []
for filename in image_filenames:
    img_path = os.path.join(image_folder, filename)  # Construct the full path to the image file
    images.append(Image.open(img_path))  # Open the image file and append it to the images list

# Assuming image_grid is defined elsewhere in the script or imported
grid = image_grid(images, cols=3)  # Create an image grid with 3 columns
grid.show()  # Display the image grid using .show()

"""## Classify damage"""

# List of damage classes (labels)
classes = [
```

```
    'Broken stitches',
    'Opened seam',
    'Slanted heel',
    'Sole not flat',
    'Peeling leather/tears',
    'Weak cementing',
    'wrinkles',
    'dirty shoes',
    'Leather Faded'
]

# Process text (classes) and images using a transformer pipeline (processor)
inputs = processor(
    text=classes,              # List of damage classes
    images=images,             # List of images to process
    return_tensors="pt",       # Return PyTorch tensors
    padding=True,              # Pad inputs to the same length
    do_convert_rgb=False       # Assume images are already in RGB format
)

# Pass inputs through the model to get outputs
outputs = model(inputs)

# Extract logits for image-text similarity scores
logits_per_image = outputs.logits_per_image  # Tensor of similarity scores

# Apply softmax to logits to get class probabilities
probs = logits_per_image.softmax(dim=1)  # Probabilities for each class label

# Get the highest scoring label (class) for each image
predicted_damage = torch.argmax(probs, dim=1)  # Tensor of predicted class indices

# Convert predicted class index to damage label string
damage = str(classes[predicted_damage])  # Get the damage label for the predicted class

# Print the predicted damage label
print(damage)

"""## Classify type"""

# List of shoe types (labels)
classes = [
    'Dress Shoes',
    'Loafer',
    'Boots (Ankle)',
    'Boots (Knee High)',
    'High Heels',
    'Stilettos',
    'Sandals',
    'Runners of shoes'
]

# Process text (classes) and images using a transformer pipeline (processor)
inputs = processor(
    text=classes,              # List of shoe types
```

```
    images=images,             # List of images to process
    return_tensors="pt",       # Return PyTorch tensors
    padding=True,              # Pad inputs to the same length
    do_convert_rgb=False       # Assume images are already in RGB format
)

# Pass inputs through the model to get outputs
outputs = model(inputs)

# Extract logits for image-text similarity scores
logits_per_image = outputs.logits_per_image  # Tensor of similarity scores

# Apply softmax to logits to get class probabilities
probs = logits_per_image.softmax(dim=1)  # Probabilities for each shoe type label

# Get the highest scoring label (shoe type) for each image
predicted_type = torch.argmax(probs, dim=1)  # Tensor of predicted shoe type indices

# Convert predicted shoe type index to string label
shoes_type = str(classes[predicted_type])  # Get the shoe type label for the predicted class

# Print the predicted shoe type label
print(shoes_type)

"""## Classify material"""

# List of material classes (labels)
classes = [
    'Leather',
    'Rubber',
    'Textiles',
    'Synthetics',
    'Braided',
    'Cork',
    'Nylon',
    'Neoprene',
    'Silk',
    'Velvet',
    'Foam'
]

# Process text (classes) and images using a transformer pipeline (processor)
inputs = processor(
    text=classes,              # List of material classes
    images=images,             # List of images to process
    return_tensors="pt",       # Return PyTorch tensors
    padding=True,              # Pad inputs to the same length
    do_convert_rgb=False       # Assume images are already in RGB format
)

# Pass inputs through the model to get outputs
outputs = model(inputs)

# Extract logits for image-text similarity scores
logits_per_image = outputs.logits_per_image  # Tensor of similarity scores
```

```
# Apply softmax to logits to get class probabilities
probs = logits_per_image.softmax(dim=1)  # Probabilities for each material label

# Get the highest scoring label (material) for each image
predicted_material = torch.argmax(probs, dim=1)  # Tensor of predicted material indices

# Convert predicted material index to string label
material = str(classes[predicted_material])  # Get the material label for the predicted class

# Print the predicted material label
print(material)

"""## Comparing input and model output section

"""

# Broken Stitches
if damage == 'Broken stitches' :
  if shoes_type == input_customer_type and input_customer_restoration == "Patch & Sewing
Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Open seam
elif damage == 'Opened seam':
  if shoes_type == input_customer_type and input_customer_restoration == "Patch & Sewing
Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Slanted heel
elif damage == 'Slanted heel':
  if shoes_type == input_customer_type and input_customer_restoration == "Heel Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Sole not flat
elif damage == 'Sole not flat':
  if shoes_type == input_customer_type and input_customer_restoration == "Hardware Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Peeling leather/tears
elif damage == 'Peeling leather/tears' :
  if shoes_type == input_customer_type and input_customer_restoration == "Leather
Alternation":
```

```
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Weak cementing
elif damage == 'Weak cementing' :
  if shoes_type == input_customer_type and input_customer_restoration == "Half/Full sole":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Wrinkles
elif damage == 'wrinkles':
  if shoes_type == input_customer_type and input_customer_restoration == "Hardware Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Dirty shoes
elif damage == 'dirty shoes':
  if shoes_type == input_customer_type and input_customer_restoration == "Leather Care":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Leather Faded
elif damage == 'Leather Faded':
  if shoes_type == input_customer_type and input_customer_restoration == "Leather
Customization RE-DYE":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
else:
  print("There is no recommendations")

"""# Demo 2: Unmatch

## User input
"""

# Customer input
input_customer_type = "Boost"
input_customer_restoration = "Hardware Repair"

# string variable for storing outcome
damage = ""
material = ""
shoes_type = ""
```

```
"""## Image import"""

# Local directory containing the images
image_folder = '/content/drive/MyDrive/Colab_Notebooks/Gluu/datasets/test/Damage'

# List of image filenames in the local directory
image_filenames = [
    'wrinkles.jpeg',
]

images = []
for filename in image_filenames:
    img_path = os.path.join(image_folder, filename)
    images.append(Image.open(img_path))

grid = image_grid(images, cols=3)
grid.show()  # Use .show() for displaying the image grid

"""## Classify damage"""

classes = ['Broken stitches',
           'Opened seam',
           'Slanted heel',
           'Sole not flat',
           'Peeling leather/tears',
           'Weak cementing',
           'wrinkles',
           'dirty shoes',
           'Leather Faded'
           ]
inputs = processor(text=classes, images=images, return_tensors="pt", padding=True,
do_convert_rgb=False)

outputs = model(inputs)
logits_per_image = outputs.logits_per_image  # this is the image-text similarity score
probs = logits_per_image.softmax(dim=1)  # we can take the softmax to get the label
probabilities

# Get the highest scoring label for each image
predicted_damage = torch.argmax(probs, dim=1)
damage = str(classes[predicted_damage])
print(damage)

"""## Classify types"""

classes = ['Dress Shoes', 'Loafer', 'Boots (Ankle)',  'Boots (Knee High)', 'High Heels', 'Stilettos',
'Sandals', 'Runners of shoes']
inputs = processor(text=classes, images=images, return_tensors="pt", padding=True,
do_convert_rgb=False)

outputs = model(inputs)
logits_per_image = outputs.logits_per_image  # this is the image-text similarity score
probs = logits_per_image.softmax(dim=1)  # we can take the softmax to get the label
probabilities
```

```
# Get the highest scoring label for each image
predicted_type = torch.argmax(probs, dim=1)
shoes_type = str(classes[predicted_type])
print(shoes_type)

"""## Classify material"""

classes = ['Leather',
           'Rubber',
           'Textiles',
           'Synthetics',
           'Braided',
           'Cork',
           'Nylon',
           'Neoprene',
           'Silk',
           'Velvet',
           'Foam']
inputs = processor(text=classes, images=images, return_tensors="pt", padding=True,
do_convert_rgb=False)

outputs = model(inputs)
logits_per_image = outputs.logits_per_image  # this is the image-text similarity score
probs = logits_per_image.softmax(dim=1)  # we can take the softmax to get the label
probabilities

# Get the highest scoring label for each image
predicted_material = torch.argmax(probs, dim=1)
material = str(classes[predicted_material])
print(material)

"""## Comparing input and model output section"""

# Broken Stitches
if damage == 'Broken stitches' :
  if shoes_type == input_customer_type and input_customer_restoration == "Patch & Sewing
Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Open seam
elif damage == 'Opened seam':
  if shoes_type == input_customer_type and input_customer_restoration == "Patch & Sewing
Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Slanted heel
elif damage == 'Slanted heel':
  if shoes_type == input_customer_type and input_customer_restoration == "Heel Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
```

```
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Sole not flat
elif damage == 'Sole not flat':
  if shoes_type == input_customer_type and input_customer_restoration == "Hardware Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Peeling leather/tears
elif damage == 'Peeling leather/tears' :
  if shoes_type == input_customer_type and input_customer_restoration == "Leather
Alternation":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Weak cementing
elif damage == 'Weak cementing' :
  if shoes_type == input_customer_type and input_customer_restoration == "Half/Full sole":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Wrinkles
elif damage == 'wrinkles':
  if shoes_type == input_customer_type and input_customer_restoration == "Hardware Repair":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Dirty shoes
elif damage == 'dirty shoes':
  if shoes_type == input_customer_type and input_customer_restoration == "Leather Care":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
# Leather Faded
elif damage == 'Leather Faded':
  if shoes_type == input_customer_type and input_customer_restoration == "Leather
Customization RE-DYE":
    print_info("correct", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
  else:
    print_info("wrong", shoes_type, damage, material, input_customer_type,
input_customer_restoration)
```

```
else:
  print("There is no recommendations")
```

Appendix – Self reflection

Yuan-Zhi Cai (3886716)
Anoop Varghese (3962557)
Sweta Karmacharya (3938458)
Mukesh Lakshmana (3859256)

We discussed what we learn for this project together. However, we have started late due to project re allocation ,as the initial project was more aligned to Computer Science professionals with more focus on UI Designing and not related to AI or Data Science .Since the Client is trying to integrate AI or Data Science technology and since it's startup Data was less for designing specific machine learning algorithm for client needs .As per the guidance of tutor we have try to find the dataset by ourselves and after finalizing able to find a broad dataset for shoes . In the next step we tried to do research on multiple papers and was able to find Zero shot learning will be more advanced and suited technology for addressing the Client requirement . Further working on the paper and replicating the code we run into multiple issues mainly computation power and memory related to the advanced model .Due to the challenges and timeline we used pre trained model from OpenAI as it's more suited and tailored for the customer requirements as it can be used to predict other artifacts also by feeding the model required labels .

Learnings

For replicating contents in the advanced research paper needs vast knowledge to identify the challenges beforehand than last stage like computational power ,but we are able to understand most of the codes and was able to replicate most of the program .

Requirement Gathering - We came to know the importance of the requirement gathering as initial meetings with the client we are not clear of the requirement ,but after further conversing on the matter and with guidance of professors we are able to understand what the client needs and to prepare a solution that will address the client's needs by keeping in mind the future requirement the client may have from the meetings (in our case other artifacts like watches,jewelry ,bags etc - OpenAI's CLIP model can address this)

Timeline and Milestone - By setting up TImelines and Milestones we are able to track and by setting a Timeline to work with initial model and setting up a contingency plan we are able to recover from the failure of initial plan and have got enough time to learn about the new model and able to deliver the client on time

Collaboration - We meet up in Teams and Whatsapp and in-person on Campus on various stages and was able to address the challenges and able to make further plans in the road map to address the changes

Data Preparation and Model Selection - Our technical learning also improved on a real world scenario from other than what we learned as part of Course ,we came to understand the learnings we done were able to apply on the scenario and on data preparation and how to select a suitable machine learning model or algorithm to address the issue

Gaining Proficiency in Technology and Tools - We were able to work with latest machine learning Technologies like Zero shot learning and we are able to further improve our knowledge we amassed through various courses as part of our curriculum and were able to address or

handle the requirements and challenges accordingly with help of professors guidance when needed .

Professional Growth - We are able to understand the importance of Communication skills ,Improving the ability to explain complex technical concepts to non-technical stakeholders and
Improve our ability to write clear and concise project documentation and reports .Developing a systematic approach to troubleshooting and debugging models through collaboration and by researching the papers keeping up with the latest advantages and research in AI Field

Group Work:

GLUU WORLDWIDE DATA SCIENCE AND AI IN RESTORATION SERVICES

Contribution Sheet

name	student number	contribution percentage	specific work on assignment	signature
Yuan-Zhi Cai	3886716	26.67%	Coding and DEMO Algorithm and Machine Learning Model Client Communication Research	Yuan-Zhi Cai
Anoop Varghese	3962557	26.67%	Research Report and Presentation	Anoop Varghese
Mukesh Lakshmana	3859256	26.67%	Research Report and Presentation	Mukesh Lakshmana
Sweta Karmacharya	3938458	20%	Research Report and Presentation	Sweta Karmacharya

REFERENCES

1. AI and Image Recognition:

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

2. CLIP Model:

- Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J. and Krueger, G., 2021, July. Learning transferable visual models from natural language supervision. In International Conference on Machine Learning (pp. 8748-8763). PMLR.
- Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. International Conference on Learning Representations.

3. UT-Zappos50K Dataset:

- Yu, A., & Grauman, K. (2014). Fine-grained visual comparisons with local learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 192-199).

4. Feature Extraction Techniques:

- Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.
- Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Vol. 1, pp. 886-893).

5. Resource Recovery and Sustainability:

- McDonough, W., & Braungart, M. (2002). Cradle to Cradle: Remaking the Way We Make Things. North Point Press.
- Ghisellini, P., Cialani, C., & Ulgiati, S. (2016). A review on circular economy: the expected transition to a balanced interplay of environmental and economic systems. Journal of Cleaner Production, 114, 11-32.

6. Localization of Repairs:
- Porter, M. E. (1998). Clusters and the new economics of competition. Harvard Business Review, 76(6), 77-90.
- Gereffi, G., Humphrey, J., & Sturgeon, T. (2005). The governance of global value chains. Review of International Political Economy, 12(1), 78-104.

7. AI in Task Allocation:
- Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370.
- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

8. Scalability and System Design:
- Tanenbaum, A. S., & Van Steen, M. (2007). Distributed systems: principles and paradigms. Prentice-Hall.
- Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.