National College of Ireland

Configuration Manual

MSc Research Project
MSc in Data Analytics

Siddhesh Patil
Student ID: x23309512

School of Computing
National College of Ireland

Supervisor: Prof. Hamilton Niculescu

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Siddhesh Patil

Student ID: X23309512

Programme: MSc Data Analytics **Year:** 2025

Module: MSc Research Project

Lecturer: Prof. Hamilton Niculescu

Submission Due Date: 11-08-2025

Project Title: Optimizing Footwear Recommendations using CNN

Word Count: 1050 **Page Count:** 10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Siddhesh Patil

Date: 11-08-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Siddhesh Patil
Student ID: x23309512

1 Introduction

The present configuration manual is a comprehensive guide to the setup, execution, and replication of the Optimizing Footwear Recommendations Using CNN-Based Visual Similarity and Health-Fit Filtering project. It describes the hardware and software tier, the Python packages, and dataset preparation, after which the three-phase recommendation pipeline is detailed in terms of implementation and evaluation procedures. The project takes the advantage of pretrained CNN architectures, ResNet, VGG, and EfficientNetB0, and FAISS-based retrieval methods as well as multi-label to provide both aesthetically-similar and ergonomically relevant footwear recommendations. Based on this manual, researchers and practitioners can replicate the experiments, know the reason behind each step, and adapt the framework for similar fashion-tech applications involving visual and functional product recommendation.

2 System Configuration

2.1 Software & Coding Tools

The project was primarily developed and executed in Google Colab Pro+, which provided a hosted Jupyter Notebook environment with GPU acceleration for model training. The setup can be fully replicated on local systems using Jupyter Notebook, Anaconda Navigator (Anaconda, 2024), or any compatible Python IDE. Anaconda Navigator can be used to configure the user environment and manage dependencies, while Jupyter Notebook allows for interactive execution and debugging of Python code.

2.2 Libraries Used

```
[ ] # Libraries
    import os, zipfile, cv2
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    from tqdm import tqdm

    # TensorFlow, Keras
    import tensorflow as tf
    from keras.metrics import Precision, Recall, AUC
    from tensorflow.keras.models import Model
    from tensorflow.keras.layers import Input, Dense, Dropout, BatchNormalization, Concatenate, GlobalAveragePooling2D
    from tensorflow.keras.preprocessing import image
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    from tensorflow.keras.callbacks import EarlyStopping
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.applications import ResNet50, VGG16, EfficientNetB0
    from tensorflow.keras.applications.resnet50 import preprocess_input as resnet_preprocess
    from tensorflow.keras.applications.vgg16 import preprocess_input as vgg_preprocess
    from tensorflow.keras.applications.efficientnet import preprocess_input as effnet_preprocess

    # Additional
    !pip install faiss-cpu
    import faiss
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import classification_report, f1_score, precision_score, recall_score, roc_auc_score
```

os – file and directory path operations
zipfile – extraction of dataset archives
pandas – data manipulation and analysis
numpy – numerical operations and array handling
tqdm – progress bar for loops
tensorflow – deep learning framework for CNN model building and training
keras – high-level neural network (via TensorFlow)
tensorflow.keras.applications – pretrained CNN architectures (ResNet50, VGG16, EfficientNetB0)
tensorflow.keras.preprocessing.image – image preprocessing and augmentation (ImageDataGenerator)
PIL (Pillow) – image loading and manipulation
faiss-gpu – efficient similarity search and clustering for embedding retrieval
umap-learn – dimensionality reduction for visualization
sklearn.manifold.TSNE – t-SNE dimensionality reduction for embedding space visualization
scikit-learn (sklearn) – evaluation metrics, classification reports, and preprocessing

Also installed the following for visualizations:
matplotlib – data visualization and plotting
seaborn – advanced statistical visualizations

3 Setting up the Environment

For this project, all deep learning model development and training were carried out using Google Colab, an online cloud-based platform provided by Google (Google Colab, n.d.). Google Colab is particularly suited for resource-intensive tasks such as deep learning, as it offers free access to GPU-accelerated hardware, removing the need for a high-spec local machine.

Instead of running the code locally in Jupyter Notebook, which would require a powerful system, Colab provided cloud-based GPUs that significantly reduced training times. In our case, we leveraged the T4 GPU runtime, which can speed up model execution by up to 10x

compared to a standard CPU. However, availability of T4 GPUs in the free Colab tier is not guaranteed, as it depends on current resource allocation.
Step 1: Go to: https://colab.google/

Step 2: Click New Notebook

Step 3: Navigate to Runtime → Change runtime type

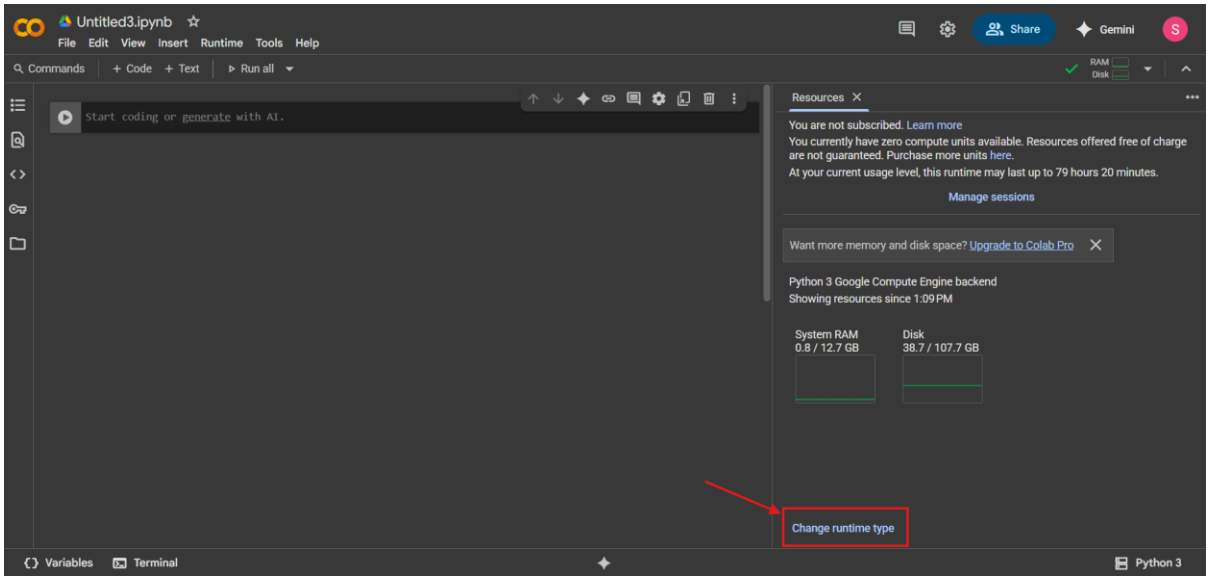

Figure 1: Change runtime

Step 4: Set Runtime type to Python 3 and Hardware accelerator to GPU (T4)

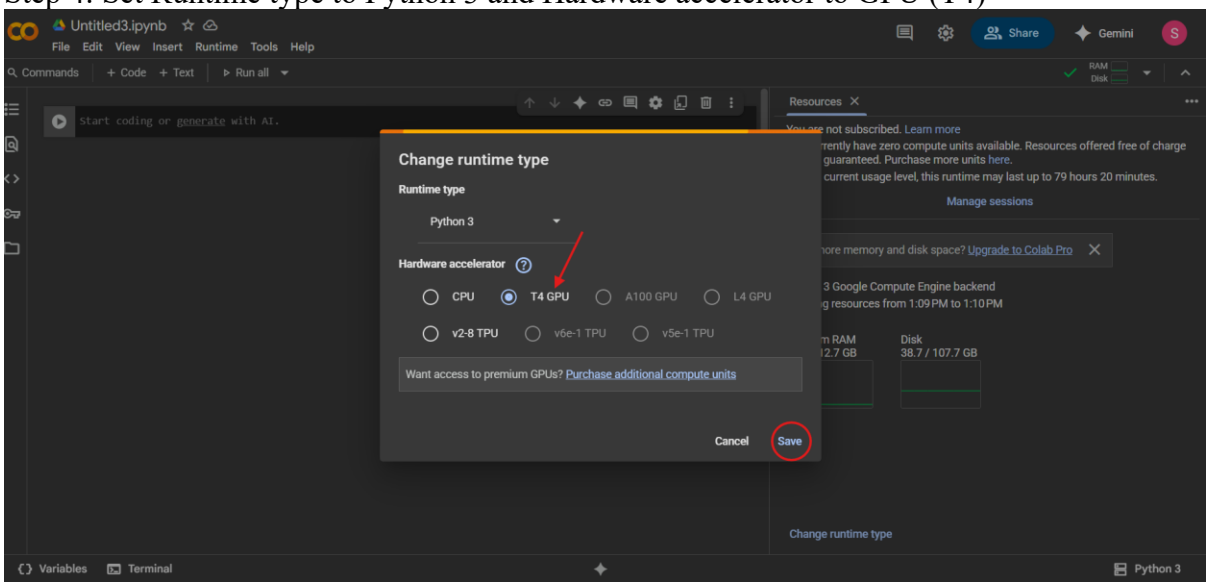

Figure 2: T4 GPU

Step 5: Make sure you SAVE.

4 Dataset – UTZappos50K

4.1 Dataset Creation

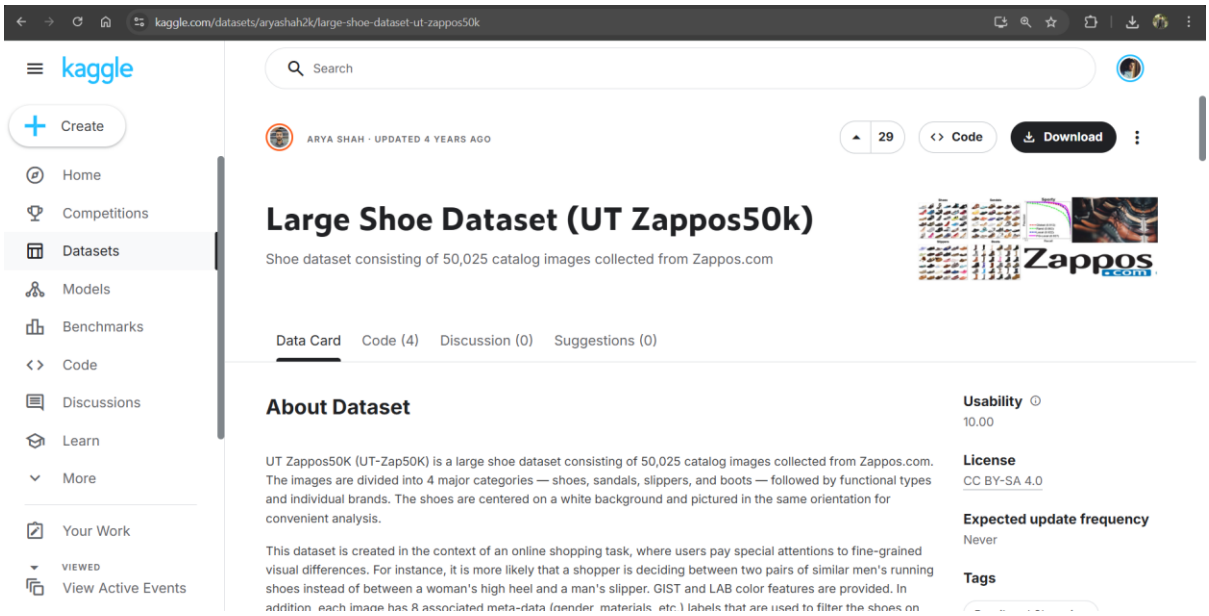

Figure 3: Kaggle Dataset

Step 1: Download the Dataset from the Link - Image Data & Meta Data File (https://www.kaggle.com/datasets/aryashah2k/large-shoe-dataset-ut-zappos50k)

Step 2: Upload it on Google Drive

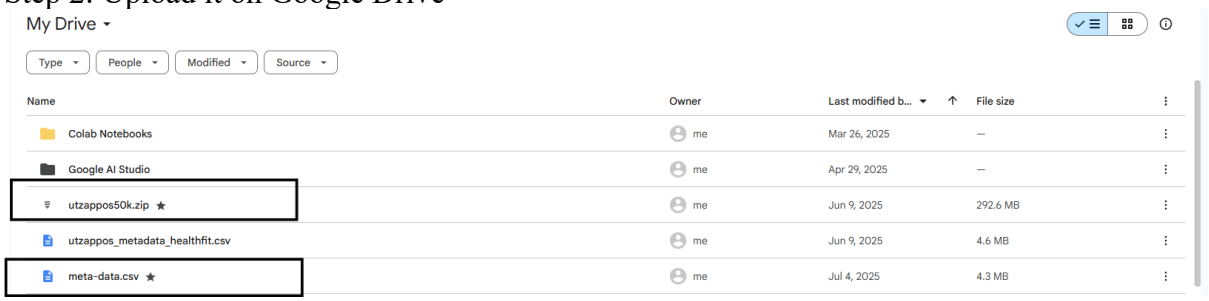

Figure 4: Dataset on Google Drive

Step 3: Extracted the Zip File from Google Colab to access the images. Loaded the Meta Data. Added 'File Name' column to meta data displaying the path to the respective image extracted from the zip folder.

Data Extraction and Metadata Preparation

```
[ ] # Extract zip
    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)

    # Load metadata and generate CID filename
    meta_df = pd.read_csv(meta_path)
    meta_df['cid_filename'] = meta_df['CID'].astype(str).str.replace("-", ".", regex=False) + ".jpg"

    # Build full image path mapping
    image_records = []
    for root, _, files in os.walk(extract_dir):
        for file in files:
            if file.lower().endswith(".jpg"):
                image_records.append({
                    "cid_filename": file,
                    "image_full_path": os.path.join(root, file)
                })
    image_df = pd.DataFrame(image_records)

    # Merge metadata with image paths
    merged_df = pd.merge(meta_df, image_df, on='cid_filename', how='left')
```

Figure 5. Data Extarction

Step 4: Sample Data Generation

```
[ ] # Normalize categorical columns
    for col in ['HeelHeight', 'Insole', 'Material', 'ToeStyle']:
        merged_df[col] = merged_df[col].fillna('').astype(str).str.lower()

    # Define binary health tag features
    merged_df['arch_support'] = merged_df.apply(lambda row: int(
        any(x in row['HeelHeight'] for x in ['1in - 1 3/4in']) or
        any(x in row['Insole'] for x in ['arch', 'poron', 'contour', 'orthotic'])
    ), axis=1)

    merged_df['cushioning'] = merged_df.apply(lambda row: int(
        any(x in row['Insole'] for x in ['padded', 'cushion', 'memory foam', 'gel'])
    ), axis=1)

    merged_df['breathability'] = merged_df.apply(lambda row: int(
        any(x in row['Material'] for x in ['mesh', 'knit', 'canvas', 'perforated', 'woven'])
    ), axis=1)

    merged_df['widefeet'] = merged_df.apply(lambda row: int(
        any(x in row['ToeStyle'] for x in ['square', 'wide', 'round', 'open toe'])
    ), axis=1)
```

Figure 6. Engineered Health Fit Tags for addition in Meta Data

```
[ ] # Sample 1000 per tag
    arch_sample = merged_df[merged_df['arch_support'] == 1].sample(n=1000, random_state=42)
    cushion_sample = merged_df[merged_df['cushioning'] == 1].sample(n=1000, random_state=43)
    breath_sample = merged_df[merged_df['breathability'] == 1].sample(n=1000, random_state=44)
    widefeet_sample = merged_df[merged_df['widefeet'] == 1].sample(n=1000, random_state=45)

    # Remaining 1000 untagged
    tagged_cids = set(pd.concat([arch_sample, cushion_sample, breath_sample, widefeet_sample])['CID'])
    remaining = merged_df[~merged_df['CID'].isin(tagged_cids)].sample(n=1000, random_state=46)

    # Concatenate and label
    final_5k_df = pd.concat([arch_sample, cushion_sample, breath_sample, widefeet_sample, remaining])
    final_5k_df = final_5k_df.drop_duplicates(subset='CID').reset_index(drop=True)
    final_5k_df['source'] = 'tagged_train'
    final_5k_df.loc[final_5k_df['CID'].isin(remaining['CID']), 'source'] = 'random_test'

    print("Final 5K Dataset Shape:", final_5k_df.shape)

    Final 5K Dataset Shape: (4897, 16)
```

Figure 7. Sample Data Generation

Engineered 4 Health Tags, concatenated 1000 images per tag + 1000 random images into a stratified sample of 5000 (which turned out to be 4897 due to overlapping of tags). This

subset is used for entire research theory.

4.2 Data Understanding

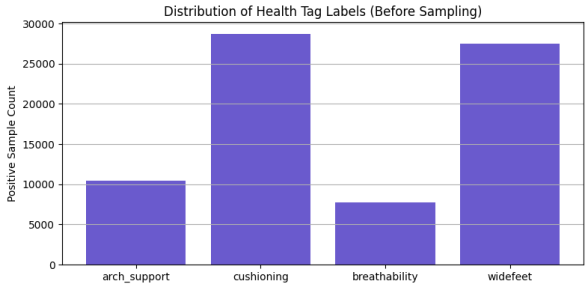

Figure 8. Before Sampling

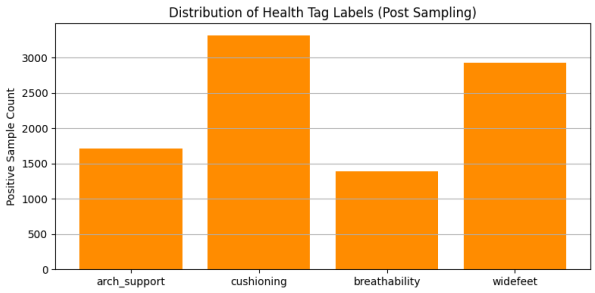

Figure 9. After Sampling

Initially, UT Zappos50K dataset was represented by more than 50,000 footwear images with an extremely uneven distribution of 4 tags of health-fitness (arch support, cushioning, breathability, and wide feet). There were some categories that were overrepresented and others with quite far fewer examples as well, which may cause biased model training.

After sampling, the curated subset of 4870 unique images that gave a better balanced distribution of tags without making the size of the dataset too large to train the models iteratively in a GPU enabled environment.

5 Models & Evaluation

The research is divided into 3 phases wherein, the same CNN models , ResNet50, VGG16, EfficentNetB0 are used with different configurations to compare performance in generating embeddings & recommendations.

5.1 Model

Phase 1 - The pretrained CNNs, initialised with ImageNet weights, were used purely as feature extractors by removing their final classification layers. The extracted embeddings were indexed using FAISS for unimodal (visual-only) similarity search.

Phase 2 - Multimodal Embedding with Health-Tag Fusion
The CNN-generated visual embeddings were concatenated with binary health-tag vectors to form multimodal representations. This fusion enabled the retrieval process to account for both aesthetic and ergonomic attributes.

Phase 3 - Multi-Label Classification
The CNN backbones were fine-tuned with a new dense classification head to directly predict the four health-fit attributes (arch support, cushioning, breathability, and wide feet) in a multi-label setup.

5.2 Model Evaluation

Model performance was assessed using a combination of quantitative metrics and qualitative visualisations across all three phases. For the retrieval tasks in Phase 1 (unimodal) and Phase 2 (multimodal), evaluation was carried out using Precision@5 to measure the proportion of relevant items among the top-5 retrieved results. In Phase 2, an additional HealthTagMatch@5 metric was introduced to quantify the alignment between the retrieved items and the query image in terms of health-fit attributes.

For Phase 3 (multi-label classification), standard classification metrics such as macro-averaged F1-score, precision, and recall were computed both overall and per health-fit label (arch support, cushioning, breathability, and wide feet).

	Model	Label	precision	recall	f1-score	support
0	VGG16	arch_support	0.807	0.715	0.758	316.0
1	EfficientNetB0	breathability	0.824	0.817	0.820	240.0
2	VGG16	cushioning	0.710	0.998	0.829	553.0
3	EfficientNetB0	widefeet	0.780	0.899	0.835	476.0

Figure 10. Example Results : Phase 3

To provide an interpretable view of the embedding quality and to assess how well the models grouped semantically similar footwear items, t-SNE (t-distributed Stochastic Neighbor Embedding) and UMAP (Uniform Manifold Approximation and Projection) were used for dimensionality reduction and plotted in 2D space. These visualisations offered qualitative insight into how embeddings clustered according to health-fit tags, highlighting differences between unimodal and multimodal configurations, as well as between the three CNN backbones.

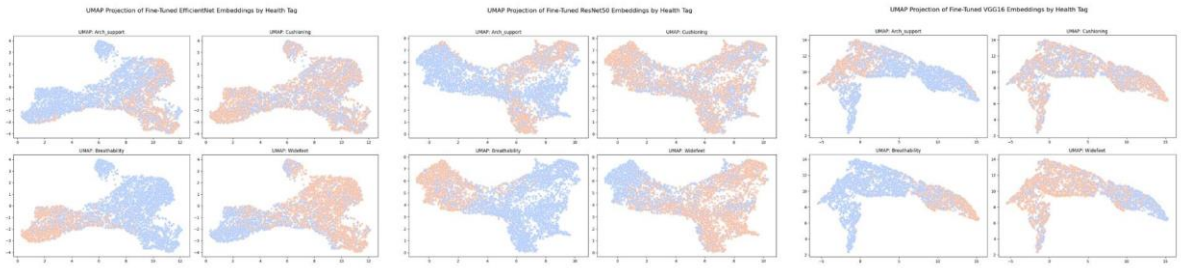

Figure 11. Example UMAP (Phase 3)

References

Shah, A. (2021) UT Zappos50K Dataset. Kaggle. Available at: https://www.kaggle.com/datasets/landrunner/ut-zappos50k

Yu, A. and Grauman, K. (2017) 'Semantic Jitter: Dense Supervision for Visual Comparisons via Synthetic Images', ICCV.