Machine Learning and Pattern Recognition

SWIPESTER

Cloth Classification and Quality Testing

GET STARTED →

Agastya Tiwari, Siddhanth Karthikeyan and Sana Arora

Swipester

Table Of Content

Swipester

Problem Statement

Talking about Swipster and why its important

Swipester

Problem Statement

FULL PROBLEM STATEMENT

Swipester is an AI-powered thrift shopping platform that integrates a personalized recommendation system to learn users' evolving style preferences and a computer vision model that detects wear, stains, and defects in clothing to ensure quality and trust in second-hand fashion.

Swipester

Literature Survey

Our work combines adaptive recommendation + automated garment quality verification in a single pipeline, which existing systems typically address separately.

Swipester

Literature Review for Personalised Recommendation

Content-based and multimodal

- **What they solve:** recommend using item attributes and solves cold-start problem
- **Typical solution:** CNN / CLIP embeddings & metadata through “visual similarity”
- **Shortcoming:** often becomes “similar items”, not sustained personalisation, weak user modelling

Sequential based recommendations

- **What they solve:** short-term intent from sequences: what the user clicked/swiped recently
- **Typical solution:** GRU4Rec, SASRec, BERT4Rec
- **Shortcoming for thrift:** needs lots of history; ignores “strength” of interactions; not exploration-aware

Collaborative filtering and implicit feedback

- **What they solve:** learn preferences from likes/swipes/clicks
- **Typical solution:** matrix factorisation for implicit feedback like LightFM, ALS, BPR loss
- **Shortcoming for you:** failure on cold-start items and high churn inventory

Two retrieval and ranking

- **What they solve:** scale and personalisation
- **Typical solution:** two-tower retrieval and ranker like GBDT or neural ranker
- **Shortcoming:** static user embedding updates, batch updates, or weak session context

Swipester

Literature Review for Wear, Tear and Stain Detection

- Most existing research comes from textile quality control and assistive technologies for visually impaired users.
- Defects such as tears, holes, and stains are treated as objects to detect in images.

FabricSpotDefect Dataset

- **Dataset:** 1,014 raw images, 3,286 annotated defects
- Augmented to 2,300 images with 7,641 defect labels
- **Defects:** stains, discoloration, oil marks, rust

Fabric Stain Classification

- **Dataset:** 466 images
- **Composition:** 398 stained fabrics, 68 defect-free
- **Fabrics:** cotton and polyester

Chenab Textile dataset

- **Dataset:** ~2,800 fabric samples with 7 defect classes
- **Annotation:** Roboflow, augmentation used to reduce class imbalance
- **Best Model:** YOLOv8n

Transfer learning and mask R-CNN

- Used for precise stain detection and segmentation
- **Achieved** ~91% F1 score
- **Uses** transfer learning

Clothing Defect detectin for blind people

- **Dataset:** ~340 garment images containing stains and holes
- **Model**: YOLOv5 with data augmentation
- **Performance:** mAP up to 0.747

YOLOv10 for Fabric Defects

- **Model** used for detecting torn paths and defects in fabrics
- **Dataset** included diverse textile types
- **Achieved** 85.6% accuracy, outperforming previous YOLO

Convulational Neural Network

- **Model used:** YOLOv10 (CNN-based one-stage object detector) trained to detect torn paths/fabric defects
- **Dataset:** Custom textile dataset with diverse fabric types

- Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, an Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks.
- Alharbi, Saad, Abdullah Alhindi, and Abdulrahman Alzahrani. 2023. "Bl People: Clothing Category Classification and Stain Detection Using Transfer Learning." Sensors 23 (5): 2609.
- Kim, J., H. Lee, and S. Park. 2021. "Using Object Detection Technology t Identify Defects in Clothing for Blind People." In Proceedings of the IE International Conference on Consumer Electronics.
- Hussain, A., M. Khan, and S. Ahmad. 2024. "Deep Learning Based Fabri Defect Detection in Textile Manufacturing." Applied Sciences 14 (1): 92
- Mendeley Data. 2022. FabricSpotDefect: Annotated Dataset for Identifying Spot Defects in Different Fabric Types.
- Roboflow Universe. 2023. Defect Detection in Cloth Dataset.
- Dataset Ninja. 2023. Fabric Stain Detection Dataset.
- https://datasetninja.com/fabric-stain-detection
- Khichi, Saif. 2022. Fabric Defect Detection Dataset. Kaggle.
- https://www.kaggle.com/datasets/saifkhichi96/fabric-defect-detectior

Swipester

Dataset and Features Preprocessing

how was the features pre-processing done by you?

Swipester

Dataset and Features Preprocessing: Recommendation System

Dataset Overview

Dataset Used: DeepFashion-style fashion image dataset
Total usable images: 44,071

- Train: 35,256
- Validation: 4,407
- Test: 4,408

Final clothing classes: 6 merged categories

- bottoms
- dresses
- knitwear
- outerwear
- rompers/jumpsuits
- tops

Why DeepFashion?

- Large-scale fashion dataset with high visual diversity
- Contains real-world variations in:
  - pose
  - lighting
  - background clutter
  - occlusions
  - garment overlap
- Suitable for learning robust clothing representations

Feature extraction

Feature Type	Method Used
Visual Features	EfficientNet-based image embeddings / CNN features
Text Features	Captions processed using text vectorization
Metadata Features	Attribute/category side information
Behavioral Features	Swipe response, response time, recency, positive swipe count
Image Processing	Resize, normalization, center crop, foreground crop
Recommendation Features	Category preference, embedding similarity, recency-weighted interest

Ethical Consideration

1. The dataset contains human models in some images
2. Potential bias due to fashion trends and brand/style imbalance
3. Recommendation systems can reinforce popularity bias

Swipester

Dataset and Features Preprocessing: Recommendation System

Missing Noise

- Missing timestamps handled using fallback recency estimation
- Missing response times replaced using median imputation
- Invalid embeddings excluded from ranking pipeline

Model Selection Justification

Why LinearSVC Was Selected

- Strong generalization on high-dimensional embeddings
- Computationally efficient
- Robust decision boundaries for sparse multimodal features
- Lower overfitting compared to deeper classifiers

Future Improvements

- Attention-based garment localisation
- Transformer-based multimodal fusion
- Temporal user preference modelling
- Contrastive representation learning

Feature Engineering Challenges

Challenges Faced

- Background clutter affecting garment extraction
- Multiple garments present in single images
- Class imbalance between categories

Solutions Implemented

- Category merging into broader classes
- Foreground crop heuristics
- Multi-view preprocessing pipeline
- Weighted behavioural scoring

Preprocessing Details

- Resize to 224×224
- Image normalization
- Center crop generation
- Foreground-aware crop estimation
- Multi-view augmentation during inference

User Preference Modeling

The recommendation engine incorporates:

- Positive swipe count
- Swipe response
- Response time
- Recency-weighted interactions
- Embedding similarity scores

Swipester

Dataset and Features Preprocessing: Wear, Tear and Stain Detection

Dataset Overview

- Composition: Integrates four distinct datasets (Torn Clothes, Fabric Stain, FabricSpotDefect, and Blind-Clothing Defect) totaling over 18,000 images.
- Real-World Focus: Captures actual smartphone photography of garments featuring everyday defects like tears, stains, holes, and discolorations.
- Distribution Profile: Highly imbalanced—the Torn Clothes dataset (~15k images) heavily dominates the smaller, specialized defect datasets.
- Ethical Compliance: Strictly excludes personally identifiable information (PII, faces, tags) and uses appropriately licensed (CC0) or consented data.

Dataset Processing

- Standardization: Images are cleaned (backgrounds cropped), uniformly resized (e.g., 224x224 px), and pixel values are normalized to a 0 to 1 scale.
- Data Augmentation: Rotations, flips, blurring, and hue/brightness shifts were applied to simulate diverse real-world camera and lighting conditions.
- Feature Extraction & Dimensionality: Leverages lightweight backbones (MobileNetV2/EfficientNet) and YOLO architectures for defect localization, relying on CNN layers for implicit dimensionality reduction.

Swipester

Dataset and Features Preprocessing: Wear and Tear Detection Model

LDA vs PCA

- PCA works blind (Unsupervised): It ignores our labels and might just focus on useless variations, like different camera lighting.
- LDA is targeted (Supervised): It specifically uses our "good," "torn," and "stained" labels to push the three classes as far apart as possible.
- The Presentation Bonus: Because we have exactly 3 classes, LDA perfectly boils the data down to 2 components. That means we get a clean 2D scatter plot to visually prove to the professor that our classes actually separate.

Features Removal

Irrelevant to Detection

- light_pixel_ratio: High bright pixels don't tell us anything useful about damage or stains.
- num_defect_annotations: That's just a label count from the YOLO file, not an actual image feature.
- width_px / height_px: Image dimensions tell us nothing about the condition of the clothing.
- aspect_ratio: Whether a photo is portrait or landscape is irrelevant to damage detection.

Swipester

ML Methodology

Which ML methods did you use to work on the above problem statement and why?

Swipester

ML METHADOLOGY: Personalised recommendation model

Why EfficientNetB0?

- Pretrained CNN with strong image representation capability
- Computationally efficient compared to larger CNNs
- Performs well on fashion-image understanding tasks
- Captures:
  - texture
  - silhouette
  - color patterns
  - garment structure

Output:
High-dimensional visual embedding vectors

Why LinearSVC?

LinearSVC was selected because:

- Strong performance on high-dimensional embeddings
- Fast training and inference
- Better generalization on sparse multimodal features
- Lower overfitting risk compared to deeper classifiers

Role:
Predict garment category from extracted embeddings.

End-to-End Workflow

DeepFashion 2 clothing images → image preprocessing → Visual fingerprint generation → EfficientNetB0 converts each garment into embedding vectors → category prediction → LinearSVC predicts clothing type/category

User interaction collection → User taste profile learning → Positive swipes are converted into user preference embeddings → Similarity and behaviour scoring → Garment embedding is matched with user profiles → Recommendation ranking output → Top users/items ranked using confidence score → Exploration mechanism → Occasionally recommends diverse items to avoid overfitting user taste

Component	Method Used	Purpose
Feature Extraction	EfficientNetB0	Extract deep visual embeddings
Classification	LinearSVC	Predict clothing category
Recommendation Engine	Embedding similarity + weighted ranking	Match users to garments
Behavioral Modeling	Recency & interaction weighting	Personalize recommendations

Swipester

Algorithm: Wear, Tear and Stain Detection

Goal

- Develop a lightweight, efficient, and accurate on-device automated system to classify user-uploaded garments as "Good", "Torn", or "Stained"

Approach

- Base Architecture: MobileNetV2 initialized with ImageNet weights via transfer learning
- Two-Phase Fine-Tuning: Train classifier head first (frozen base), then unfreeze top 30 layers for deeper adaptation
- Classification: Final predictions across three classes: Good, Torn, or Stained
- Mobile Deployment: Model exported to both .keras and TensorFlow Lite (.tflite) with dynamic range quantization for on-device inference

Model Selection & Training Strategy

- Core Architecture: Transfer learning with MobileNetV2 (pretrained on ImageNet) — chosen for its mobile-first design and strong texture/edge feature extraction relevant to fabric defect detection
- Phase 1 (12 epochs): Base layers fully frozen; custom head trained using Adam optimizer (lr = 1e-3) with categorical cross-entropy loss
- Phase 2 (8 epochs): Top 30 MobileNetV2 layers unfrozen, BatchNorm layers kept frozen for stability; fine-tuned at lr = 1e-5
- Regularization: Dropout (0.3 + 0.2), EarlyStopping (patience=5), ReduceLROnPlateau (halves lr every 2 stagnant epochs), ModelCheckpoint saving best weights
- Data Split: 80/10/10 train/val/test; preprocess_input() used for normalization aligned with ImageNet pretrained weights

Alternatives & Mitigations

- No existing dataset for Indian thrift clothing — built and labelled a custom dataset from scratch with automated 80/10/10 splitting
- Class imbalance (fewer torn/stained samples) — addressed using sklearn's compute_class_weight('balanced') to apply weighted loss during training
- Overfitting on small data — mitigated with augmentation: ±15° rotation, ±10% shift, horizontal flip, ±10% zoom, brightness variation (0.8–1.2×)
- Domain gap between ImageNet and real thrift photos — fine-tuning Phase 2 directly adapts deeper layers to clothing-specific textures
- Hardware constraints — trained on Google Colab T4 GPU; code structured to run as .py or .ipynb

Swipester

Challenges faced

Challenge	Solution
Background clutter in images	Multi-view preprocessing & foreground crop
High-dimensional embeddings	Efficient CNN backbone + normalization
Noisy user behavior data	Weighted scoring & median imputation
Class imbalance	Category merging into broader classes
Hardware limitations	EfficientNetB0 selected for lower compute cost
Real-world image variability	Multi-view augmentation pipeline
Class imbalance (fewer torn/stained samples)	Applied compute_class_weight('balanced') to penalise majority class during training

Swipester

Performance Metrics and Deployability of the ML solution

What were the performance metrics and how much were they?

Swipester

Performance Metrics for personal recommendation model

	precision	recall	f1-score	support
bottoms	0.69	0.90	0.78	347
dresses	0.90	0.94	0.92	683
knitwear	0.73	0.79	0.76	500
outerwear	0.68	0.89	0.77	367
rompers_jumpsuits	0.81	0.92	0.86	166
tops	0.96	0.83	0.89	2345
accuracy			0.86	4408
macro avg	0.80	0.88	0.83	4408
weighted avg	0.88	0.86	0.86	4408

	bottoms	dresses	knitwear	outerwear	rompers_jumpsuits	tops
bottoms	312	0	4	14	1	16
dresses	2	645	3	6	19	8
knitwear	14	0	393	56	0	37
outerwear	8	2	15	327	0	15
rompers_jumpsuits	0	12	0	1	152	1
tops	119	54	125	74	16	1957

Why These Metrics Matter

1. Classification Accuracy (~86%)

→ Shows the model can reliably understand and classify clothing categories from uploaded images.

2. Recall@3 of reccommendation (~94.29%)

→ Measures whether relevant users/items appear in the top recommendation results.

How the System Works in Practice

→ The deployed pipeline can:

- accept uploaded garment images
- classify clothing type
- provide personalized recommendations in real time

Current deployment:

- Streamlit-based interactive application
- Real-time inference using EfficientNet embeddings

MODEL BENCHMARKING

EfficientNet embeddings ranked by LinearSVC win across baselines.

MODEL	TEST ACCURACY
Logistic Regression	66.63 %
MLP — baseline	69.58 %
ResNet50 + Gradient Boosting	81.94 %
EfficientNet + Gradient Boosting	82.92 %
ResNet50 + LinearSVC	83.19 %
EfficientNet + LinearSVC	**84.73 %**

OBSERVATION

EfficientNet-based multimodal embeddings consistently outperformed traditional baselines across every ranker tested.

WHY EFFICIENTNET + LINEARSVC

Superior hierarchical feature extraction
Efficient parameter scaling
Strong separation in embedding space
Robust classification margins

Swipester

Performance Metrics for Wear, Tear and Stain detection

	precision	recall	f1-score	support
good	0.89	0.86	0.88	436
stained	0.95	0.98	0.96	447
torn	0.88	0.89	0.89	433
accuracy			0.91	1316
macro avg	0.91	0.91	0.91	1316
weighted avg	0.91	0.91	0.91	1316

Why These Metrics Matter?

- 86% accuracy shows reliable clothing classification
- High Recall indicates strong recommendation quality
- Behavioral + similarity scoring improves personalization
- Model generalizes well on unseen fashion images

MobileNetV2 is most efficient out of the chosen networks.

MODEL	TEST ACCURACY	MACRO-F1
Logistic Regression (24 features)	72.14%	0.681
SVM-RBF (24 features)	78.63%	0.742
Random Forest (24 features)	83.47%	0.812
MobileNetV2 — midsem path	87.21%	0.851
HistGradientBoosting (24 features)	88.09%	0.872
RandomForest + HGB soft-voting	89.44%	0.887
SwipesterQC-v2 (ensemble + calibration + tear guardrail) ★	**91.03%**	**0.909**

OBSERVATION:
→ Engineered image features fed into a calibrated ensemble consistently outperformed end-to-end CNN classifiers when tested on real seller uploads with messy backgrounds and varied lighting.

WHY SWIPESTERQC-V2 + CALIBRATION?

- Ensemble sees structured statistics, not raw pixels alone
- Calibration eliminates false alarms on clean catalog images
- Tear guardrail catches tiny edge-localised damage CNN misses
- No GPU required deployable on a standard CPU server

Swipester

Deployability of the ML solution

→ **Campus Fashion Marketplace**

1. Students upload clothing items for:

- resale
- exchange
- thrift recommendations

2. Student Recommendation Feed

System can personalise:

- clothing suggestions
- thrift listings
- outfit recommendations

→ **Possible Deployment**

- Campus thrift/resale platform
- Personalised student fashion feed
- AI-assisted clothing recommendation system

Advantages of Deployment

- Lightweight inference pipeline
- Real-time prediction capability
- Scalable recommendation logic
- Minimal manual labelling during inference

Swipester

LIVE DEMO

SHOWING OF THE WORKING MODEL

Swipester

Thank You

Please feel free to ask questions!

+91 8383973326