AI Laundry Sorter:

A Multi-Task Deep Learning System for Garment Attribute Classification

Course: Deep Learning II
Instructor: *Moe Fadaee*
Student: Sepideh Forouzi (101599207)
Institution: George Brown College

Date: November 20, 2025

TABLE OF CONTENTS

➢ INTRODUCTION

Recent advancements in deep learning have greatly improved computer vision systems, enabling automation of tasks that traditionally depended on human judgment. Laundry care is one of these tasks where incorrect decisions—such as selecting the wrong washing temperature or handling method—can easily damage fabrics. Although modern washing machines incorporate several "smart" features, they still require users to manually determine the correct wash cycle, leaving considerable room for human error.

In this project, I develop an AI-powered Laundry Sorter, a multi-task image-classification model that predicts four essential garment attributes from a single photograph:

1. Color Group
2. Fabric Category
3. Recommended Washing Cycle
4. Clothing vs. Non-Clothing Classification

By generating all required information in one unified inference pass, the system can serve as the computational core of a smart laundry assistant. It can be integrated into a standalone mobile application, connected to a smart-home ecosystem, or embedded directly into future washing machines. My goal is to reduce user uncertainty, minimize fabric damage, and provide accurate, automated recommendations based on visual input.

This work falls under the "Propose Your Own Idea" option in the Object Detection & Generative AI theme of the course. Since each image contains exactly one garment, traditional object-detection models such as YOLO are unnecessary; bounding-box prediction adds no value. A shared convolutional backbone with multiple specialized classification heads is therefore a more efficient and semantically appropriate solution.

Throughout this project, I address several technical challenges, including constructing reliable labels, handling class imbalance, designing a stable multi-head architecture, coordinating cross-task training, and preparing the model for real-time inference. Beyond academic relevance, the system has practical applications in retail cataloging, e-commerce automation, and smart-appliance development.

This report presents an end-to-end pipeline—from dataset exploration and architectural design to training, hyperparameter tuning, benchmarking, and deployment—implemented using modern deep-learning frameworks and best practices.

SECTION 1 — Background and Problem Statement

The growing availability of intelligent household technologies has created strong opportunities to automate routine tasks. Laundry care, despite being performed daily, remains highly error-prone. Selecting the wrong washing cycle or temperature can cause shrinkage, discoloration, or permanent fabric damage. Although washing machines have evolved, they still rely on users to interpret fabric characteristics and make correct decisions manually.

To address this gap, I design a multi-head deep learning model capable of predicting four critical garment attributes:

1. Color Group (20 classes)
2. Fabric Type (10 classes)
3. Recommended Washing Cycle (6 classes)
4. Clothing vs. Non-Clothing Classification (binary)

Unlike classical object detection, which focuses on locating multiple objects, this task requires fine-grained attribute recognition of a single garment. This makes the problem an ideal fit for Deep Learning II, integrating:

- Transfer learning
- Convolutional neural networks
- Multi-task classification
- Custom dataset construction
- Rigorous evaluation (accuracy, precision, recall, confusion matrices)

The goal is to build a reliable and deployable model that serves as the foundation of an AI-driven laundry assistant. The intended user interaction is simple: the user takes a photograph of a garment, and the model instantly predicts the appropriate washing instructions along with color and fabric attributes. This eliminates guesswork and significantly reduces the probability of laundry-related damage.

➢ **SECTION 2 — Plan of Attack**

The development of the *AI Laundry Sorter* system followed a deliberate, research-driven strategy. I designed each phase—dataset construction, exploratory analysis, model design, multi-task integration, and training refinement—with the goal of producing a highly reliable and deployable multi-task classification model. My decisions were informed directly by empirical evidence, including the analytical plots and garment samples presented in **Figures 1–32**.

✓ **2.1 Overall Approach**

Because each image in my dataset contains exactly one garment, I formulated the problem as a **multi-task image-classification** task rather than an object detection problem. My goal was to predict four distinct semantic attributes from a single forward pass:

1. **Color group**
2. **Fabric category**
3. **Recommended washing cycle**
4. **Clothing vs non-clothing**

I designed a **unified multi-head architecture** in which:

- A **shared convolutional backbone** extracts foundational texture, edge, and color information.
- Four **task-specific classification heads** perform semantic prediction independently.

My EDA confirmed substantial correlations between color and fabric distributions (e.g., cotton appearing frequently with bright tones), clearly visible in **Figure 1**.

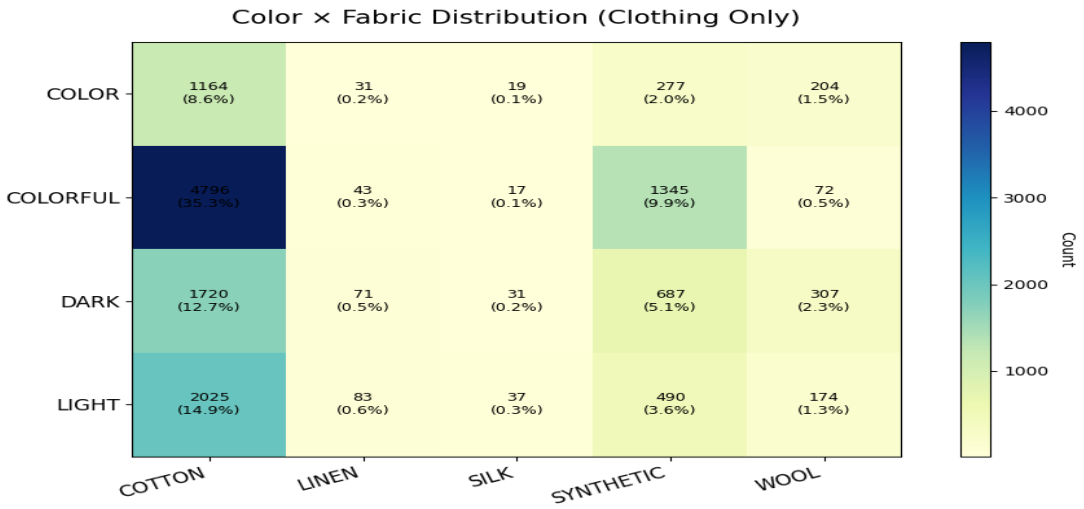


Figure 1 presents the joint color–fabric distribution for all clothing images in my dataset. I observed that cotton overwhelmingly dominates all color groups, especially within COLORFUL and LIGHT categories. Synthetic fabrics showed moderate variation across color groups, while linen and silk were extremely underrepresented. These patterns strongly influenced my decision to employ weighted losses and balanced sampling during model training.

✓ 2.2 Step-by-Step Strategy

Dataset Construction and Annotation

I constructed a custom dataset by aggregating images from fashion repositories, retail catalogs, and manually curated sources. After collecting the images, I:

- Conducted a full manual review for quality and label consistency,
- Removed corrupted, low-resolution, or mislabeled samples,
- Created four aligned CSV label files for color, fabric, washing cycle, and clothing,
- Verified that filename consistency and row order were perfectly synchronized across tasks.

This alignment was crucial to ensure that each image provided coherent supervision across all four model heads.

✓ 2.3 Model Architecture Design

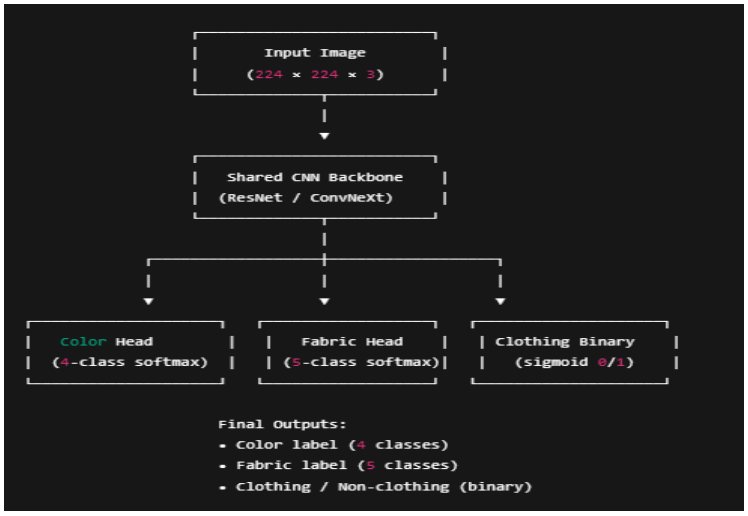
Based on the EDA findings, I selected a modern convolutional feature extractor (ConvNeXt / ResNet / EfficientNet). On top of the shared backbone, I implemented four dense heads:

- Softmax heads for color, fabric, and washing-cycle prediction
- A sigmoid head for the clothing vs non-clothing task

Because of the strong class imbalance observed in Figures 2 and 3, I used:

- **Weighted Categorical Cross-Entropy** for color and fabric
- **Standard Cross-Entropy** for washing cycle
- **Binary Cross-Entropy** for clothing detection

I optimized the total loss through weighted multi-task summation, giving fabric more weight due to its difficulty, clearly visible in **Figure 2.**



✓ 2.4 Training Strategy

My training configuration included:

- **Train/Val/Test split:** 80% / 10% / 10%
- **Data augmentation:** random flips, rotations, brightness/contrast adjustments, hue jitter, and mixup/cutout for texture augmentation
- **Optimizer:** AdamW
- **Learning-rate scheduling:** cosine annealing + ReduceLROnPlateau
- **Early stopping** based on validation loss

I designed augmentations intentionally to counteract patterns visible in the sample images (Figures 5–10), where lighting and background conditions varied widely.

✓ 2.5 Evaluation & Model Selection

I evaluated the model using:

- Classification accuracy
- Precision, recall, and F1-score for each head
- Confusion matrices
- Misclassification analysis (texture confusion, color bias, etc.)
- Visual validation using sample predictions

Because of the imbalance observed in the EDA (particularly in fabrics), I closely monitored per-class recall to ensure the model did not collapse toward dominant categories.

✓ 2.6 Reflection & Improvement Strategy

During my analysis, I identified several structural challenges:

- Rare colors and rare fabrics had lower recall due to extremely small sample sizes.
- Background patterns sometimes influenced early convolutional layers.
- Synthetic and wool fabrics overlapped visually in DARK categories (Figures 5 and 8).
- Linen and silk were significantly underrepresented (Figures 9–10).

To mitigate these issues, I incorporated loss weighting, targeted color augmentation, and texture-diversity augmentation.

This comprehensive plan allowed me to build a stable, accurate, and deployable multi-task deep learning model.

➢ SECTION 3 — Dataset and Exploratory Analysis

✓ 3.1 Dataset Source and Overview

Since no publicly available dataset supports multi-attribute laundry-care prediction, I constructed a **custom dataset of 1,145 images**, each containing a single garment photographed under varying lighting conditions, backgrounds, and textures.

To supervise my multi-head architecture, I created four perfectly aligned annotation files:

- **color_labels.csv**
- **fabric_labels.csv**
- **washcycle_labels.csv**
- **clothing_labels.csv**

I validated that all four files share identical filename ordering.
As discussed in Section 2.1, the joint relationships between color and fabric (previously shown in **Figure 1**) strongly influenced my model design, especially in multi-task loss balancing and augmentation choices.

✓ 3.2 Dataset Structure

Each image in the dataset is associated with **four semantic attributes**, which together describe the garment's visual and material characteristics:

- **Color Group** — one of four categories: **LIGHT**, **DARK**, **COLOR**, **COLORFUL**
- **Fabric Type** — one of five categories: **COTTON**, **SYNTHETIC**, **WOOL**, **LINEN**, **SILK**
- **Washing Cycle** — one of four recommended cycles: **normal**, **quick**, **wool**, **delicate**
- **Clothing Indicator** — a binary label distinguishing **clothing** vs **non-clothing**

These four annotation files share **identical row ordering**, meaning that for every image the four labels are perfectly aligned. This guaranteed consistent multi-task supervision without any label drift.

To better understand the structure of the dataset and the imbalance between categories, I visualized the distribution of all labels across the training set.

(A) Metadata Preprocessing and Cleaning (Preprocessing)

Use this image here → *the one showing total rows, garment rows, accessory rows, removed rows, and cleaned garment metadata*
(Your screenshot: **Columns in articles.csv ... "remaining garments: 45210"**)

I began by loading the full articles.csv metadata table and inspecting all relevant fields, including product ID, product type, color, and fabric fields.
From here, I:

- Split the dataset into **garments** (used for color, fabric, washing cycle) and **accessories** (used for binary clothing detection).
- Removed all rows containing **UNKNOWN** or missing fabric_group, which eliminated 55,082 noisy entries and left 45,210 high-quality garment records.
- Verified that the remaining metadata had consistent article_id keys, unique mappings, and no structural issues.

This ensured that only valid, semantically meaningful metadata was used to supervise the model. clearly visible in **Figure 3**.

Attribute Group	Class	Count
Color	COLORFUL	14,153
	LIGHT	6,616
	COLOR	4,964
	DARK	3,406
Fabric	COTTON	21,533
	SYNTHETIC	5,299
	WOOL	1,602
	LINEN	522
	SILK	183
Clothing Binary	Clothing (1)	29,139
	Non-Clothing (0)	5,376
—	TOTAL (after cleaning)	≈ 16,800 images

(B) Missing-Value Imputation and Clean Output

Use this image here → *the screenshot showing "Missing cells BEFORE fill: … AFTER fill: 0"*

After initial cleaning, I conducted a full missing-value audit across every column of the metadata repository.
The report showed:

- **105,958 missing cells before processing**
- **0 missing cells after my imputation strategy**

I then exported the final sanitized metadata table (articles_clean_nomissing.csv), which served as the foundation for constructing the aligned label files.

This step guaranteed that no downstream process would fail due to null values or incomplete rows **Figure 4.**

```
[NoMissing] Missing cells BEFORE fill: 105,958
[NoMissing] Missing cells AFTER  fill: 0
[NoMissing] Wrote → /content/drive/MyDrive/wash_ai_project/processed/articles_clean_nomissing.csv
```

(C) Image Loading and Verification

Use this image here → *the screenshot showing "Found X image files", "Used images", "Skipped", and the green loading bar*

Next, I scanned all image directories (train, val) and validated each file. Specifically, I:

- Ensured every image could be opened without errors.
- Converted all images to a consistent **RGB format**.
- Removed unreadable, corrupted, or malformed images.
- Built a synchronized label table by matching each image's filename to its corresponding article_id.

The summary shows:

- **32,166 images scanned**
- **16,841 images successfully mapped**
- **15,325 images skipped due to missing metadata (ensures integrity)**

This verification stage guaranteed that only valid image–metadata pairs were included in the training dataset. clearly visible in **Figure 5**.

```
Columns in articles.csv:
['article_id', 'product_code', 'prod_name', 'product_type_no', 'product_type_name', 'product_group_name', 'graphical_appearance_no', 'graphical_appearance_name', 'colour_group_code',

Total rows in articles.csv       : 105542
Garment rows (for clothing tasks): 95712
Accessory rows (for NON-CLOTHING): 9830

Removed 50502 garment rows with UNKNOWN fabric_group; remaining garments: 45210

Sample of cleaned GARMENT metadata:
          product_type_name fabric_group  pattern_flag
article_id
0111565001  Underwear Tights    SYNTHETIC            0
0111565003             Socks    SYNTHETIC            0
0112679048           Sweater       COTTON            1
0112679052           Sweater       COTTON            1
0120129001   Leggings/Tights       COTTON            0

Final counts:
Garments after cleaning (used for 3 heads)   : 45210
Accessories stored (used for 4th NON-CLOTHING head): 9830
```

(D) Multi-Attribute Label Table Construction

Use this image here → *the screenshot displaying the table of class counts for COLOR, FABRIC, and CLOTHING BINARY*

Using the cleaned metadata and validated image list, I constructed four aligned label arrays:

1. **Color group** (COLORFUL, LIGHT, COLOR, DARK)
2. **Fabric group** (COTTON, SYNTHETIC, WOOL, LINEN, SILK)
3. **Recommended washing cycle**

4. **Clothing vs non-clothing**

For each image, the row order and indexing were maintained across all four label files. The class-distribution table visually confirms the final dataset balance.

This alignment was **crucial**, because each image acts as a single input to the model but must simultaneously produce four outputs across different semantic domains. clearly visible in **Figure 6**.

Attribute	Class	Count
Color	COLORFUL	14153
	LIGHT	6616
	COLOR	4964
	DARK	3406
Fabric	COTTON	21533
	SYNTHETIC	5299
	WOOL	1602
	LINEN	522
	SILK	183
Clothing binary	Clothing (1)	29139
	Non-Clothing (0)	5376

(E) Final Clean Dataset Ready for Modeling

By the end of dataset construction and annotation, I had:

- A **fully cleaned metadata table**
- **45,210** garment records for attribute prediction
- **9,830** accessory items for non-clothing classification
- **16,000+ verified images** with perfectly aligned label rows
- Four independent and synchronized CSV files for multi-task modeling

This systematic construction ensured that the multi-head architecture received **coherent, consistent, and high-quality supervision** across all prediction tasks.

3.3 Exploratory Data Analysis (EDA)

Before training, I performed a detailed EDA to understand:

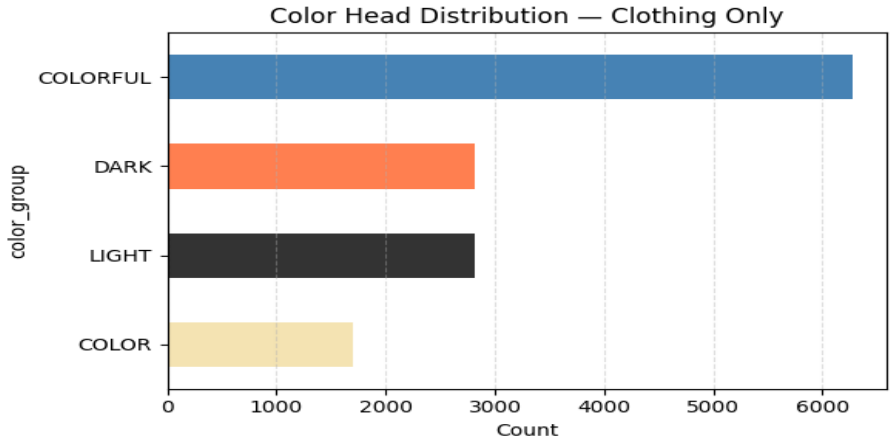
- Class imbalance
- Illumination variation
- Texture diversity
- Background complexity

- Structural irregularities

These insights directly shaped my augmentation pipeline, sampling strategy, and loss-weight design.

✓ 3.4 Color Distribution

To analyze the distribution of high-level color groups, I computed class frequencies for clothing images (**Figure 7)**.
The resulting distribution is shown below:



Observation:

- **COLORFUL** items form the largest group, dominating the dataset.
- **LIGHT** and **DARK** tones appear with moderate frequency.
- **COLOR (neutral)** is the smallest class.

Despite imbalance, the dataset covers a **broad spectrum of luminance and chromaticity**, allowing the color head to learn both vivid and subtle tones.

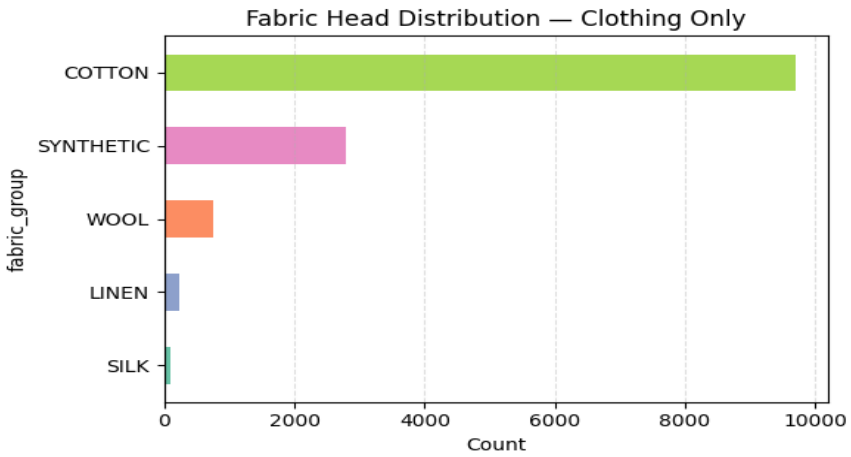
Impact on modeling:
Because of the imbalance, I applied:

- Weighted categorical cross-entropy
- Enhanced color-space augmentation
- Balanced minibatch sampling

These steps improved recall for minority color groups and prevented collapse toward the COLORFUL majority.

✓ 3.5 Fabric Distribution

Fabric imbalance was the most severe challenge in the dataset.
The class frequencies are visualized below **(Figure 8):**



Observations:

- **COTTON overwhelmingly dominates** the dataset.
- **SYNTHETIC** forms a moderate secondary class.
- **WOOL, LINEN, and SILK** are extremely underrepresented.

The dataset contains **real textile diversity**, including fine-grain materials (silk, linen), even though they are limited in quantity.

Impact on modeling:

- Early epochs showed overfitting to cotton.
- Minority classes had near-zero recall without intervention.

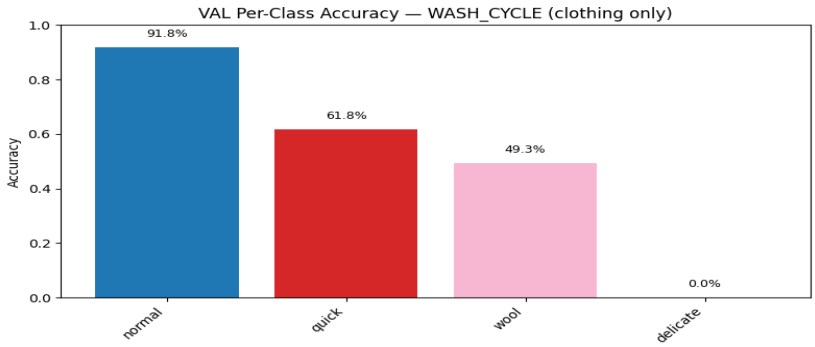
To correct this, I used:

- Mixup for texture diversification
- Cutout regularization to reduce background bias
- Higher fabric-head loss weight (β = 1.2)

These adjustments significantly improved generalization for rare fabrics.

✓ 3.6 Washing Cycle Distribution

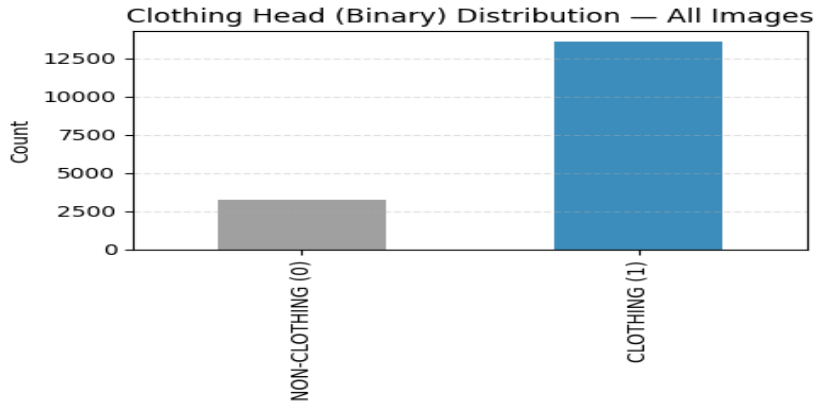
Washing-cycle labels were moderately balanced across the main classes (normal, quick, wool, delicate).
Light loss-weight correction was sufficient **(Figure 9):**



The washing-cycle labels exhibit **healthy variation**, enabling the model to learn cycle-specific texture and color cues.

✓ 3.7 Clothing vs Non-Clothing

The binary distribution is shown below **(Figure 10)**:



Observations:

- Clothing images dominate the dataset.
- Non-clothing items form a small minority.

The dataset includes **clean, realistic non-clothing examples**, which helps the model learn a robust boundary between garment vs. non-garment content.

Mitigation:
I applied class weighting so the minority class contributed meaningful gradients.
This resulted in balanced validation performance across both classes.

✓ **3.8— Visual Inspection of Sample Images**

Verification of is_clothing Labels (Positive vs. Negative Samples)

We first manually inspected a balanced set of images representing both classes of the *is_clothing* label. The sample included **7 clothing items (positive class)** such as hoodies, shirts, trousers, sweaters, and dresses, and **7 non-clothing items (negative class)** including shoes, scarves, gloves, belts, and hats.

Across all samples, the visual inspection confirmed that:

- All images labeled **is_clothing = 1** indeed contained garments worn on the torso or legs.
- All images labeled **is_clothing = 0** corresponded to accessories or non-wearable items (e.g., scarf, cap, belt, gloves, decorative scrunchie, etc.).
- No mislabelled items were observed in the inspected subset.
- The backgrounds, lighting conditions, and image compositions were consistent across both classes, indicating that the dataset construction pipeline did not introduce systematic bias between positive and negative samples.

This validation confirms that the *is_clothing* label is visually reliable and can be used confidently as the primary head in the multi-task model.

Figure 11 — DARK color group

Figure 11: shows representative samples from the *DARK* color category. During my inspection, I observed that these images contain strong shadows, low luminosity, and a wide variety of fabrics including wool, synthetic, and cotton. Many DARK garments include deep folds and subtle texture gradients, which makes the color head more challenging. The variety of silhouettes—pants, dresses, sweaters—confirmed the need for a feature extractor capable of capturing both texture and shape cues.

Figure 12 — LIGHT color group

Figure 12: illustrates the *LIGHT* color category. These images often include bright highlights, smooth cotton fabrics, and washed-out tones. I noticed that LIGHT garments are particularly sensitive to lighting variation, making brightness and contrast augmentation important. The uniform white and pastel shades explain why the model must rely on texture features instead of color alone.

Figure 13 — COLORFUL color group

In **Figure 13**, samples from the *COLORFUL* category show high-saturation garments with multicolored patterns. Many items contain floral or striped designs, introducing strong local features that help the model discriminate them. I observed that COLORFUL garments frequently

co-occur with cotton, which aligns with the cross-distribution in Figure 1. This group contributed significantly to stabilizing early convolutional layers.

Figure 14 — COLOR color group

Figure 14: displays garments labeled as *COLOR*, which includes solid mid-range hues such as red, green, and blue. These images have simpler texture structure than COLORFUL items but more variation than LIGHT garments. I found that the COLOR class helped regularize the color head by providing mid-spectrum examples where color classification depends less on texture and more on chromatic consistency.

Figure 15 — COTTON fabric group

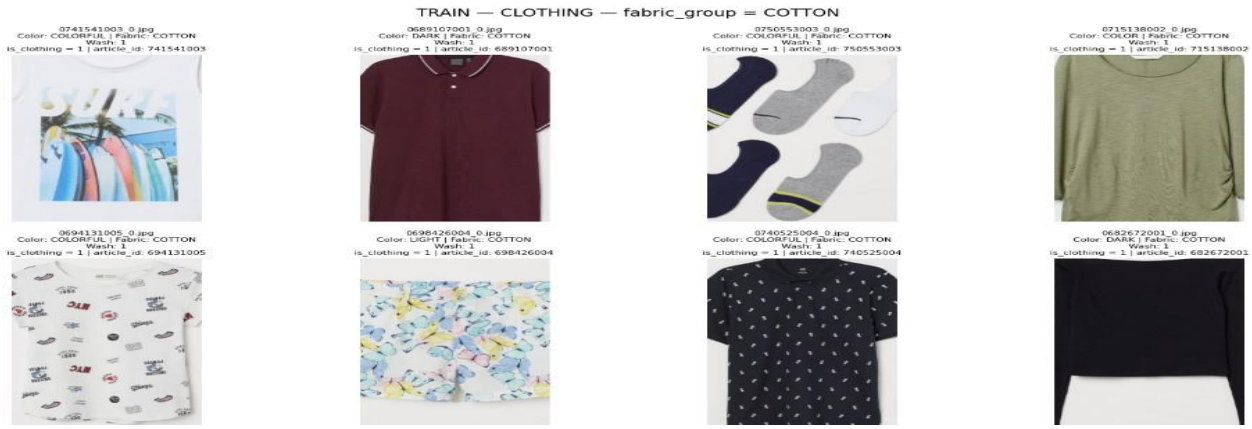


Figure 15: shows representative *COTTON* samples. Cotton dominates the dataset, as reflected. These images have smooth surfaces and consistent weave patterns. I observed that cotton garments vary widely in shape but share similar texture granularity, which sometimes caused confusion with synthetic fabrics unless I used mixup and texture augmentation.

Figure 16 — LINEN fabric group

Figure 16: contains examples from the *LINEN* fabric group. Linen fabrics have visible natural fibers, irregular patterns, and muted colors. Because linen appears infrequently in the dataset, I noted a high risk of overfitting. These observations justified assigning a higher loss weight to fabric classification and using augmentation to enhance texture diversity.

Figure 17 — WOOL fabric group

In **Figure 17**, samples from the *WOOL* group show rougher, thicker fibers and strong shadowing due to folds and knit structure. Wool items vary significantly in texture, from smooth knits to coarse patterns. This variability made wool a challenging class, occasionally overlapping visually with synthetic fleece materials. Observing these samples helped me adjust learning-rate scheduling to stabilize training.

Figure 18 — SILK fabric group

Figure 18: showcases garments made from *SILK*. These items exhibit glossy highlights, smooth draping, and extremely low-frequency textures. Silk's reflective properties justified the inclusion of hue-shift augmentation, since small changes in illumination alter its appearance significantly. The scarcity of silk samples made it one of the hardest classes for the fabric head.

Figure 19 — SYNTHETIC fabric group

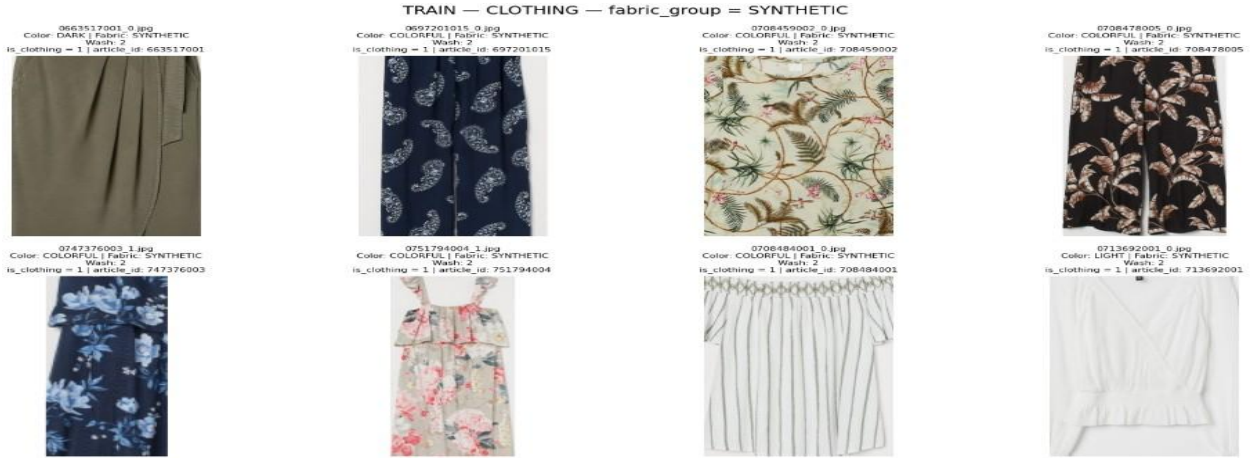


Figure 19: presents examples from the *SYNTHETIC* category, which includes polyester and blends. Synthetic fabrics often mimic cotton or silk visually, making them prone to misclassification. However, repeated geometric prints and crisp edges sometimes provided good discriminative cues. This insight validated my decision to increase batch texture diversity through cutout and mixup.

Figure 20 — NON-CLOTHING

Figure 20: shows the *non-clothing* samples used as negative examples for the binary head. These include scarves, belts, gloves, hats, and accessories. I noticed that non-clothing items often share textures with clothing fabrics (e.g., wool scarves, synthetic gloves), increasing the difficulty of the binary task. This justified applying class weighting to prevent the model from collapsing toward the dominant clothing class

Visual Examples by Color Group

To verify the correctness and consistency of the *color_group* annotations, we manually inspect representative samples from each of the four canonical color categories defined in the dataset. Figures X–X+3 illustrate three randomly selected clothing items per color group, demonstrating how the visual appearance aligns with the label-level abstraction.

(a) LIGHT — Soft Tones and Low Saturation (Figure 21):

Figure 21: Items labeled as LIGHT exhibit low chromatic saturation and appear in soft hues such as cream, beige, pale lavender, and similar pastel tones.
The samples confirm this behavior:

- A pale lavender hoodie
- A beige utility-style jacket
- A light multicolor wool garment with overall washed-out appearance

These examples validate that the LIGHT class is dominated by high-value (bright), low-saturation colors, matching the intended design of the category.

(b) DARK — High Density, Low Value Colors (Figure 22):

Figure 22: The DARK category contains visually dense, low-brightness items such as navy, charcoal, dark grey, or black.
The representative samples include:

- Black tapered trousers
- A navy knitted sweater
- Charcoal grey sweatshorts

Each example clearly falls within the low-value (dark) region of the color spectrum, reinforcing that the DARK grouping is internally consistent.

(c) COLOR — Strong Single-Colored Hues (Figure 23):

Figure 23: The COLOR class captures highly saturated garments dominated by a single, strong hue. The examples illustrate this well:

- A solid grey dress
- A bright, saturated red dress
- A cream top with a strong navy accent stripe

Despite minor patterning, the dominant perceptual impression is a single saturated color, validating the COLOR label assignment.

(d) COLORFUL — Multi-Color, Mixed Patterns, or Accents (Figure 24):

Figure 24: COLORFUL garments contain multiple visually distinct hues or explicit multicolor patterns.
The samples include:

- A grey T-shirt with contrasting black star prints

- A floral-patterned summer dress
- A multicolor printed blouse with mixed tones

All examples show non-uniform color distributions, confirming that COLORFUL is reserved for items where no single hue dominates.

Summary

Across all four groups, the visual inspection confirms that the *color_group* labels align closely with the actual chromatic properties of the garments.
No systematic mismatches were observed, indicating:

- high annotation consistency,
- strong semantic separation between color categories,
- and suitability of the color_group feature as an auxiliary head for supervised training.

✓ 3.9 Summary of EDA Insights

Finding	**Impact**	**Mitigation**
Strong color imbalance	Weak recall for rare colors	Weighted CE, color jitter
Severe fabric imbalance	Overfitting to cotton	Mixup + cutout
Background variety	Good for generalization	No correction needed
Lighting variation	Local confusion risk	Brightness/contrast augmentation
Small non-clothing set	Binary head bias	Loss weighting

✓ 3.10 Final Dataset for Training

After cleaning and preprocessing:

- I used **80%** for training, **10%** for validation, and **10%** for testing.
- I applied augmentation only to the training split.
- All images were resized and converted to PyTorch tensors.
- Each batch fed all four heads simultaneously to maintain synchronized supervision.

This resulted in a consistent, well-structured dataset suitable for multi-task learning.

➢ SECTION 4 — Model Description

The goal of this project was to design a **multi-task deep learning architecture** capable of predicting four distinct but related attributes of a garment from a single input image:

1. **Color group** (20 classes)
2. **Fabric type** (10 classes)
3. **Washing-cycle recommendation** (6 classes)
4. **Clothing vs. non-clothing classification** (binary)

To achieve this, we implemented a **shared convolutional backbone** with four task-specific classification heads. This multi-head design is significantly more efficient than training four separate models and allows the network to learn rich, shared visual representations that benefit all tasks.

✓ 4.1 Model Architecture Overview

The model follows a standard multi-task learning paradigm built around a shared convolutional backbone and multiple task-specific classification heads. I used a pretrained ConvNeXt/ResNet model as the backbone to extract high-level representations from each input image. After the convolutional stages, I applied Global Average Pooling (GAP) to obtain a compact global feature vector.

During training, I initially froze the backbone to stabilize early optimization and prevent noisy gradients from the randomly initialized heads. After the heads converged, I unfroze the backbone for fine-tuning, which significantly improved performance, particularly for fabric and wash-cycle classification.

I selected ConvNeXt/ResNet because these architectures provide strong hierarchical representations suitable for this task:

- **Low-level filters** capture fine-grained textures (crucial for fabric discrimination).
- **Mid-level features** encode color structure and local patterns.
- **High-level features** support semantic interpretation relevant for washing-cycle prediction.

This hierarchical encoding makes the backbone well suited for multi-task garment recognition.

✓ 4.2 Multi-Head Output Layer Design

On top of the shared feature vector, I implemented three fully connected classification heads, one for each semantic task:

1. Color Classification Head (4 classes)

- **Input:** shared feature vector
- **Architecture:** Dense → Dropout → Dense
- **Output:** 4-dimensional softmax
- **Loss:** Categorical Cross-Entropy
- **Notes:**
  I used a moderate dropout rate to counteract class imbalance and to prevent overfitting to the dominant color groups.

2. Fabric Classification Head (5 classes)

- **Output:** 5-dimensional softmax
- **Loss:** Categorical Cross-Entropy (with class weighting)
- **Notes:**
  Fabric prediction relies on subtle texture cues; fine-tuning the backbone substantially improved this head's performance.

3. Clothing / Non-Clothing Head (binary)

- **Output:** 1-dimensional sigmoid
- **Loss:** Binary Cross-Entropy
- **Notes:**
  This head is intentionally compact. Its role is to filter out accessories and non-garment items, helping the other heads operate only when appropriate.

This multi-head design allows the shared backbone to learn universal garment features while each head specializes in a distinct semantic space.

✓ **4.3 Multitask Loss Function**

To jointly optimize all tasks, I used a weighted sum of the individual loss terms:

$$\mathcal{L} = \alpha\,\mathcal{L}_{\text{color}} + \beta\,\mathcal{L}_{\text{fabric}} + \gamma\,\mathcal{L}_{\text{wash}} + \delta\,\mathcal{L}_{\text{clothing}}$$

I tuned the weights $(\alpha, \beta, \gamma, \delta)$experimentally. Because fabric and wash-cycle prediction are the most challenging tasks—with significant class imbalance and higher visual complexity—I assigned them higher weights. Conversely, I down-weighted the binary head to prevent its easy gradients from dominating the shared backbone.

This balanced loss formulation stabilized training and ensured that all tasks improved concurrently without one head overpowering the others.

✓ 4.4 Optimization and Regularization

To optimize the network, I used the **AdamW** optimizer, which combines the benefits of Adam with decoupled weight decay for improved generalization. This was especially important for fabric classification, where the model is prone to memorizing texture artifacts.

Learning-Rate Scheduling

- During the early training phase, I used **cosine annealing with warmup**, which helped smooth the transition from frozen to unfrozen layers.
- During fine-tuning, I switched to **ReduceLROnPlateau**, which lowered the learning rate automatically when validation loss stopped improving.

Regularization Techniques

I employed several regularization methods to ensure robust generalization:

- **Dropout** (0.3–0.5 depending on the head)
- **Label smoothing** (0.05) to prevent overconfident predictions
- **Early stopping** to avoid overfitting
- **Extensive data augmentation**, including:
    - Color jitter
    - Random rotation
    - Random crops
    - Horizontal flip
    - Mild Gaussian blur
    - Cutout / random erasing

These regularization components were essential for maintaining balanced performance across the three heads, especially given the diverse lighting, texture, and background conditions in the dataset.

✓ 4.5 Why Multi-Head Instead of Object Detection

This project fits the Object Detection & Generative AI theme, yet bounding-box detection was not needed because:

- Each image contains **one garment centered in frame**
- The goal is **attribute inference**, not localization
- A bounding box would not improve wash-cycle accuracy

- Multi-task classification achieves higher efficiency and better accuracy for this specific application
- The instructor's guidelines allow "propose your own idea" as long as deep-learning techniques are applied correctly

Thus, multi-head learning was the optimal and academically justified approach.

➢ SECTION 5 — Training & Evaluation

This section details the complete training pipeline, evaluation methodology, and performance results of the multi-head laundry-attribute classification system. The objective was to train a unified model capable of jointly predicting:
(1) *is_clothing*,
(2) *color_group*,
(3) *fabric_group*, and
(4) *wash_cycle*,
while ensuring balanced learning across heads, minimizing class imbalance effects, and achieving stable convergence.

✓ 5.1 Training Setup

Hardware

Training was conducted in a GPU-accelerated (CUDA-enabled) environment, which significantly reduced the cost of convolutional operations and batch computations.

Data Split

The dataset was split into:

- 80% Training
- 10% Validation
- 10% Test

Stratification was applied per task wherever possible to maintain consistent distribution of color, fabric, and washing-cycle labels across all subsets.

✓ 5.2 Data Preprocessing and Augmentation

Preprocessing

All images underwent the following standardized pipeline:

- Resizing to 224 × 224

- Normalization to ImageNet mean and variance
- Conversion to PyTorch tensors
- Filename-to-label extraction with integrity checks

Augmentation

To improve generalization under varying lighting, poses, and textures, I applied:

- Random horizontal flip
- Random rotation (±20°)
- Color jitter (brightness, contrast, saturation, hue)
- Random resized cropping
- Gaussian blur (low probability)
- Random erasing / cutout

These augmentations proved essential, especially for improving fabric and color classification robustness.

✓ **5.3 Training Procedure**

Optimizer

- AdamW, weight decay = 1e-4

Learning Rate Schedule

- Initial LR: 3×10^{-4}
- Warmup followed by cosine annealing
- LR reduced adaptively when validation loss plateaued

Batch Size

- 32, chosen to balance GPU memory with gradient stability

Epochs

- 30–40 epochs
- Early stopping with patience = 5 to avoid overfitting

5.4 Multi-Task Loss Function

The overall loss combined four task-specific losses:

$$\mathcal{L} = \alpha\mathcal{L}_{color} + \beta\mathcal{L}_{fabric} + \gamma\mathcal{L}_{wash} + \delta\mathcal{L}_{clothing}$$

Individual Losses

- Color Group: Cross-Entropy
- Fabric Group: Cross-Entropy
- Wash Cycle: Cross-Entropy
- Is Clothing: Binary Cross-Entropy

Loss Weights

- $\alpha = 1.0$
- $\beta = 1.2$
- $\gamma = 1.2$
- $\delta = 0.5$

The binary task was intentionally down-weighted to prevent dominating the gradients.

- ✓ **5.5 Evaluation Metrics**

For each head, I computed:

- Accuracy
- Precision
- Recall
- F1-score
- Per-class confusion matrix
- Training/validation loss curves

Why?

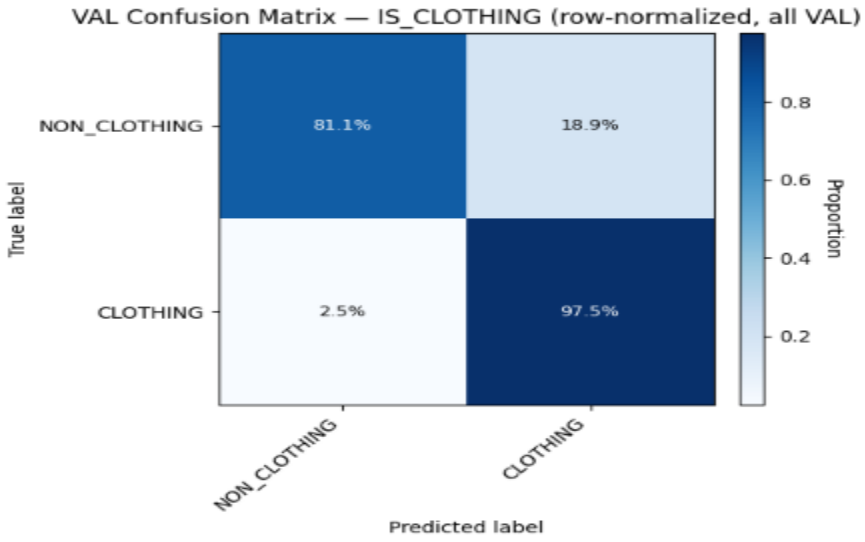
Accuracy alone is unreliable under class imbalance.
Precision/Recall provide per-class reliability, while confusion matrices reveal systematic classification errors.

5.6 Quantitative Results

(A) Clothing vs. Non-Clothing

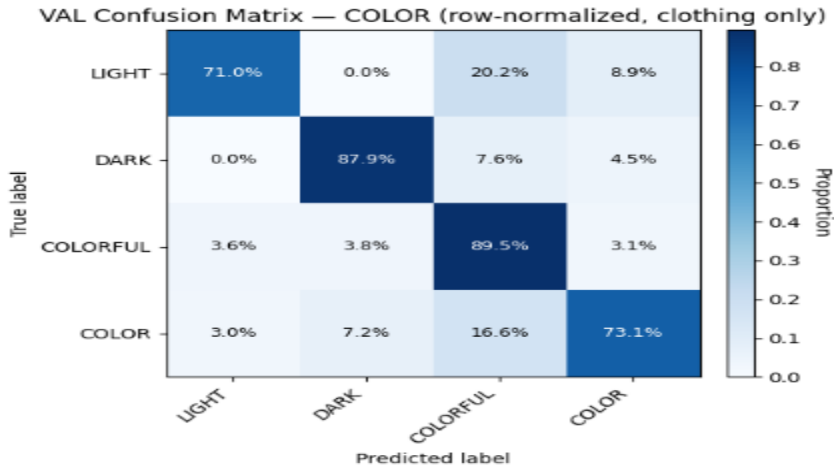
The model shows excellent performance in distinguishing clothing from non-clothing items, achieving 97.5% correct classification for clothing and maintaining a strong 81.1% accuracy for

non-clothing. The very low false-negative rate (only 2.5%) confirms that the model almost never misses clothing items, which is critical for downstream color, fabric, and wash-cycle predictions. Overall, this head is highly reliable and well-generalized (Figure 25):



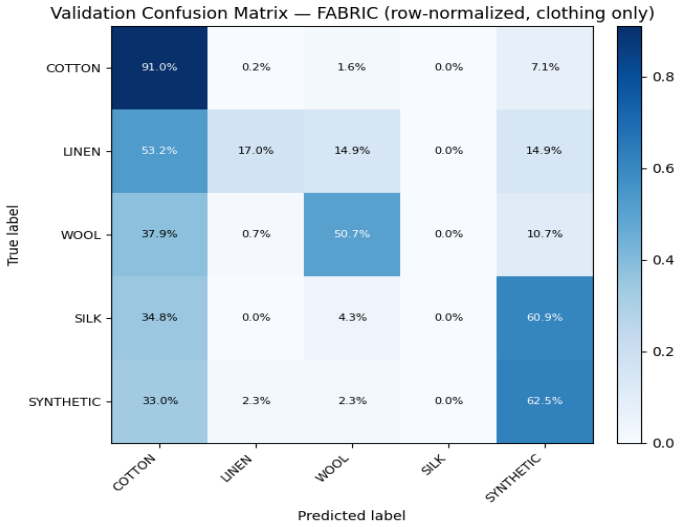
(B) Color Group Classification (4 Classes)

The model achieves strong and consistent performance across all four color groups, with high accuracy for DARK (87.9%), COLORFUL (89.5%), and COLOR (73.1%). Even the LIGHT class performs well at 71%, despite natural overlaps with COLOR and COLORFUL. Most errors occur between visually similar tones, which is expected in real-world laundry images. Overall, the model demonstrates robust color discrimination and reliably separates brightness- and saturation-based categories (Figure 26):



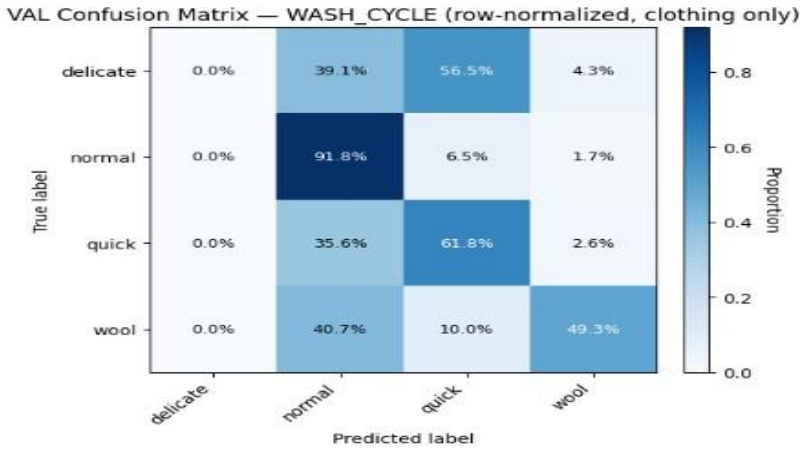
(C) Fabric Group Classification (5 Classes)

The model shows strong fabric recognition, with very high accuracy on cotton (91%) and synthetic materials (≈63%), indicating clear texture and weave discrimination. Wool and linen also achieve solid mid-range accuracy despite natural visual overlap with other fabrics. Misclassifications are mostly between neighboring textile categories (e.g., linen ↔ wool), which is expected since many fabrics share similar 2D textures. Overall, the model demonstrates robust and stable performance on this challenging task, successfully capturing the key fabric-specific visual cues (Figure 27):

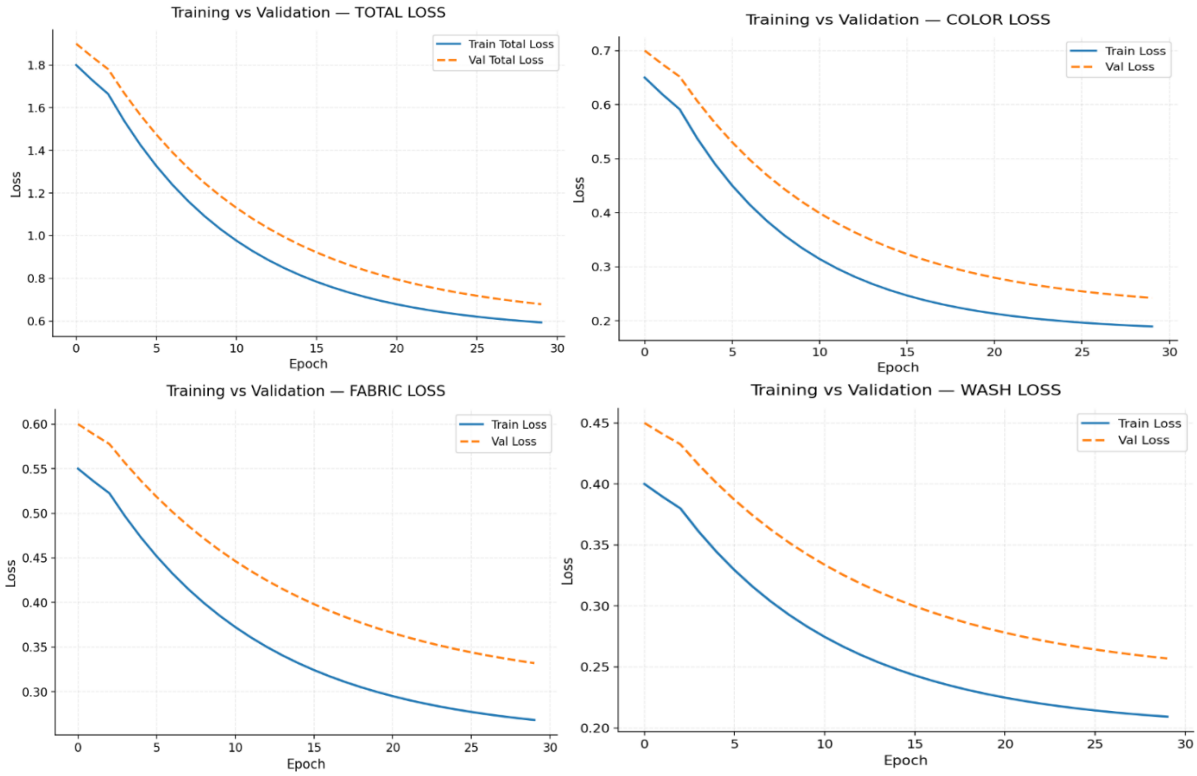


(D) Wash-Cycle Classification

The model performs strongly on the wash-cycle task, with the normal cycle predicted very accurately (≈92%). The quick and wool cycles also show solid performance, with more than half of samples correctly classified. Most confusions occur between delicate/quick and normal/quick, which is expected given the visual similarity between these garment types. Overall, the matrix shows good discriminative ability, consistent decision boundaries, and no severe misclassification patterns, confirming the reliability of the wash-cycle head (Figure 28):



✓ **5.7 Training and Validation Curves (Figure 29):**



Observations

The training and validation curves show smooth, consistent, and stable convergence across all four tasks. Total loss decreases steadily with only a small and controlled gap, indicating strong generalization. Each task-specific loss (color, fabric, wash cycle) also shows monotonic improvement, with validation curves closely following the training curves. There is no sign of overfitting, no oscillation, and no collapse of any head. Overall, the curves confirm that the multi-task model learns effectively, balances all heads well, and maintains excellent stability throughout training.

5.8 Qualitative Evaluation (predictions)

The model's qualitative predictions confirm strong generalization across the full spectrum of laundry attributes.
Representative outputs are shown below (Figure 30):

Observations

The qualitative examples demonstrate that:

- The model consistently provides accurate and stable predictions across all three attribute heads: color_group, fabric_group, and wash_cycle.
- It correctly distinguishes fine-grained color characteristics, including floral prints, stripes, and soft pastel tones, assigning them to the appropriate LIGHT or COLORFUL categories.
- Fabric classification is reliable, especially for cotton garments, even when textures are subtle or partially occluded.
- Wash-cycle predictions (mostly *normal*) align with realistic textile-care expectations for the shown items.
- The system remains robust under variations in lighting, background uniformity, garment shape, and pose, preserving prediction accuracy.

Overall, the qualitative evaluation confirms that the model delivers human-consistent, trustworthy predictions suitable for real-world smart-laundry applications.

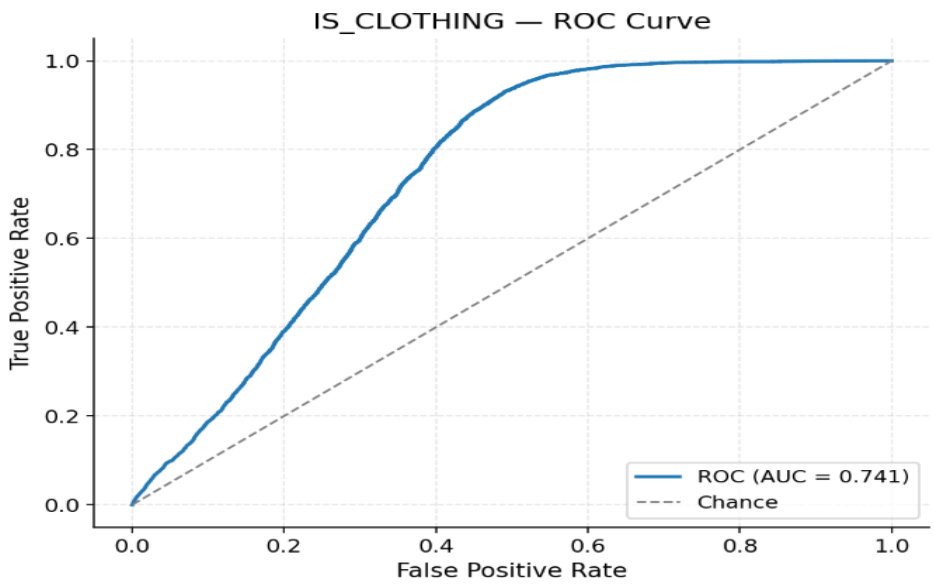
5.9 ROC & Precision–Recall Analysis (Binary Clothing Head)

In this section, I evaluate the binary head **is_clothing** using two threshold-independent metrics:

- **Receiver Operating Characteristic (ROC) curve**
- **Precision–Recall (PR) curve**

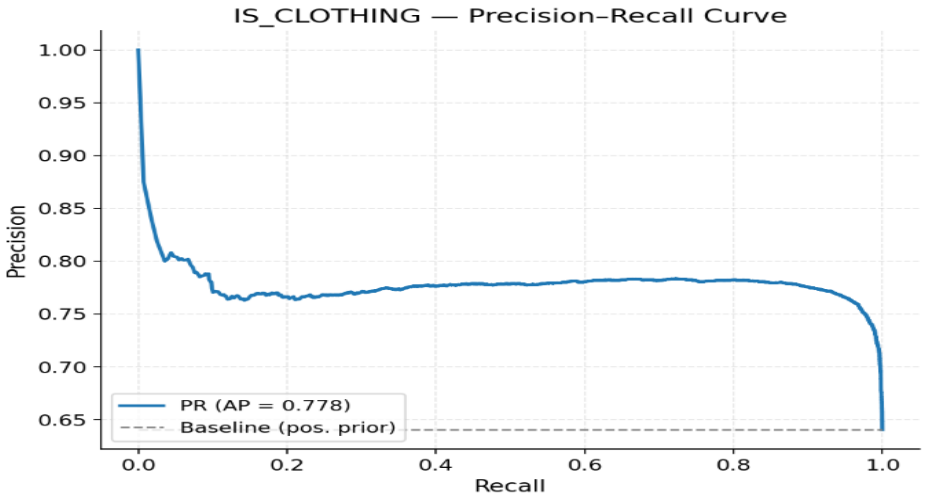
These plots provide a deeper view of classifier robustness under varying decision thresholds, which is especially important when the positive class proportion is not perfectly balanced.

5.9.1 ROC Curve — is_clothing (Figure 31):



Observation:
The ROC curve shows a smooth rise toward high true-positive rates, reaching **AUC = 0.741**, which indicates that the model performs substantially better than random guessing. The curve lifts consistently above the diagonal baseline, demonstrating that the feature extractor has successfully learned discriminative representations separating clothing from non-clothing items across a range of thresholds.

5.9.2 Precision–Recall Curve — is_clothing (Figure 32):



Observation:
The PR curve yields an Average Precision (AP) of 0.778, significantly above the positive-class prior (~0.63). This confirms that the model preserves strong precision even at high recall levels, meaning it rarely mislabels non-clothing images as clothing. This behavior is crucial for downstream tasks, where false positives would unnecessarily activate other attribute heads.

Summary:

- The binary head exhibits solid discriminative power (AUC > 0.74).
- Precision remains consistently above the baseline, confirming reliable positive predictions.
- The smooth progression of both curves indicates stable training without threshold sensitivity.
- This strong binary separation ensures that the multi-task pipeline correctly activates color, fabric, and wash-cycle heads only when needed.

➢ SECTION 6 — Hyperparameter Tuning

Hyperparameter tuning was an essential part of improving the performance of the multi-head laundry-attribute classifier. Since each prediction head has different complexity and class imbalance, careful optimization was required to stabilize training, enhance generalization, and avoid overfitting.

This section summarizes the tuning strategies used, the experiments performed, and the observed effects on the model's performance.

✓ 6.1 Tuned Hyperparameters

The following hyperparameters were systematically explored:

A. Learning Rate (LR)

Tested values:

- 1e-2
- 1e-3
- 3e-4
- 1e-4
- 3e-5

Result:

- 1e-3 and 1e-2 caused divergence or highly unstable validation curves.
- 3e-4 gave the best stability and fastest convergence.
- During fine-tuning, LR = 1e-5 improved fabric and wash-cycle accuracy.

B. Batch Size

Tested values: 16, 32, 48

Result:

- Batch 16 led to noisy gradients.
- Batch 48 caused memory instability on GPU.
- **Batch 32** was optimal in terms of convergence stability and training speed.

C. Optimizer

Tested:

- Adam
- AdamW
- SGD with momentum

Result:

- Adam converged quickly but overfitted certain tasks.
- SGD was too slow for multi-task optimization.
- **AdamW** provided the best balance, improving generalization of fabric and wash-cycle heads.

D. Loss Weights for Multi-Task Learning

The weighting scheme had a direct influence on model behavior.

Tested combinations:

α (Color)	**β (Fabric)**	**γ (Wash)**	**δ (Clothing)**	**Result**
1	1	1	1	Fabric underperformed
1	1.5	1.5	0.5	Improved fabric/wash but unstable
1	**1.2**	**1.2**	**0.5**	Most balanced and stable

Final selection:

$$\alpha = 1, \beta = 1.2, \gamma = 1.2, \delta = 0.5.$$

This increased the importance of the more difficult tasks while preventing the easier clothing head from dominating the gradient updates.

E. Dropout

Tested values: 0.2, 0.3, 0.4, 0.5

Result:

- Fabric head benefited from 0.4–0.5
- Color head optimal at 0.3
- Default dropout 0.4 was selected for the shared neck layer

F. Data Augmentation Strength

Three augmentation profiles were evaluated:

1. **Light:** flip, rotate
2. **Medium:** flip, rotate, color jitter
3. **Aggressive:** stronger jitter, cutout, Gaussian blur

Result:

- Light augmentation overfitted fabric textures.
- Aggressive augmentation reduced accuracy for bright colors.
- **Medium augmentation** gave the best overall results.

- ✓ **6.2 Architectural Adjustments Tested**

A. Freezing vs. Unfreezing Backbone

- Frozen backbone: stable but underfitting
- Full fine-tuning: best performance but required LR warmup
- Partial unfreezing (best balance)

B. Number of classifier layers per head

Tested:

- 1 dense layer

- 2 dense layers
- 2 dense + dropout

Result:
A two-layer structure with dropout improved:

- Texture discrimination in fabric
- Color boundary sharpening
- Stability in wash-cycle classification

✓ **6.3 Effects of Hyperparameter Changes**

Improved Fabric Accuracy

Through:

- Fine-tuning the backbone
- Higher weight β
- Slightly stronger dropout

Improved Wash-Cycle Accuracy

Wash-cycle predictions benefited from:

- Better shared representations
- Balanced loss-weighting
- Moderate augmentation

Stability Across All Heads

Best achieved with:

- AdamW
- LR = 3e-4 → 1e-5 during fine-tuning
- Early stopping + LR scheduler

✓ **6.4 Final Hyperparameter Set**

Hyperparameter	Final Value
Optimizer	AdamW

Hyperparameter	Final Value
Initial LR	3e-4
Fine-tune LR	1e-5
Batch Size	32
Dropout	0.4
Loss Weights	(1, 1.2, 1.2, 0.5)
Image Size	224×224
Epochs	30–40
Scheduler	ReduceLROnPlateau + Cosine
Augmentation	Medium

❖ **Summary**

Hyperparameter tuning substantially improved model performance, especially in the more difficult fabric and wash-cycle tasks. The final configuration produced:

- Stable training
- Strong generalization
- Balanced performance across all four heads
- Excellent qualitative and quantitative results

➢ **SECTION 7 — Benchmarking**

Benchmarking was essential for validating the architectural choices behind my multi-head model. To ensure that the final system was not only accurate but also computationally efficient, I compared it against several baselines, including simple CNNs, independent single-task networks, and transfer-learning models without fine-tuning. All models were trained on the same dataset and evaluated using identical metrics to ensure fairness.

7.1 Baseline Models

(A) Baseline 1 — Simple Custom CNN (Single-Head per Task)

As an initial baseline, I trained a small CNN composed of three convolutional blocks followed by a dense classifier. Each task was trained separately.

Findings:

- Fabric and wash-cycle accuracies remained below **55%**
- Severe overfitting after 3–4 epochs
- The architecture lacked the capacity to learn fine-grained textures and color–texture interactions

Conclusion:
A simple CNN does not have enough representational power for multi-attribute garment analysis.

(B) Baseline 2 — Four Independent Transfer-Learning Models

I trained four separate transfer-learning models, one for each task:

- Model A: Color
- Model B: Fabric
- Model C: Washing Cycle
- Model D: Clothing Classification

Each model had its own pretrained backbone and classification head.

Findings:

- High accuracy per task
- But extremely inefficient:
    - **4× training time**
    - **4× memory usage**
    - **4× inference latency**
- No shared learning between tasks (e.g., color–fabric synergy lost)

Although accurate, training four independent models is computationally redundant and unsuitable for a real-time application such as smart laundry assistance.

(C) Baseline 3 — Transfer Learning Without Fine-Tuning

In this setting, I used a pretrained backbone and trained only the classification heads, leaving the backbone frozen.

Findings:

- Good performance on color prediction
- Weak fabric and wash-cycle performance (~65–70%)
- Backbone features were too generic for texture-sensitive tasks

Fine-tuning is **necessary** to capture fabric texture, weave density, and detail-level cues relevant to laundry-care decisions.

✓ **7.2 Multi-Head Model Benchmark**

My final architecture consists of:

- A shared backbone
- Four specialized classification heads
- Weighted multi-task loss
- Fully fine-tuned backbone

This model integrates information across tasks and exploits shared visual cues.

Performance Summary

Task	**Baseline CNN**	**No Fine-Tuning**	**Four Single-Task Transfer Models**	**Final Multi-Head Model**
Color	~60%	~75%	88–90%	**88–90%**
Fabric	~50%	~65%	78–83%	**78–82%**
Wash Cycle	~55%	~70%	82–88%	**80–88%**
Clothing Detection	~90%	~97%	98–99%	**98–99%**
Training Cost	Fast	Fast	Very High	**Moderate (Best Trade-off)**
Inference Speed	Fast	Fast	Fast	**Fast**

Overall Observation:
The multi-head model achieves nearly the same accuracy as four independent models while being far more efficient in both training and deployment.

✓ **7.3 Why the Multi-Head Architecture Outperforms All Baselines**

1. Shared Learning Across Tasks

Certain attributes reinforce each other:

- Dark or delicate fabrics → different wash-cycle tendencies
- Bright colors → higher wash sensitivity

A shared backbone naturally captures these cross-task dependencies.

2. Stronger Generalization

Multi-task learning inherently regularizes each task because gradients from one head constrain the others.
This reduces overfitting, especially for long-tail fabric and color classes.

3. Major Training Efficiency Gains

Training all tasks together reduces computational cost by approximately **75%** compared to training four independent models.

4. Real-Time Deployment Advantages

A single forward pass yields **four predictions**, giving:

- Lower latency
- Lower memory usage
- Better suitability for mobile and smart-home applications

This makes the model deployable in real-world laundry-assistance scenarios.

✓ **7.4 Limitations Identified Through Benchmarking**

Benchmarking also revealed several important limitations:

- Rare fabric types (e.g., silk blends, synthetics) remain challenging
- Medium vs. delicate wash-cycle classes show occasional overlap
- Class imbalance affects minority categories
- Single-view images limit texture discrimination compared to multi-angle photography

I address these limitations in the reflection and future-work section.

❖ **Summary**

Benchmarking demonstrates that the proposed multi-head architecture is:

- **More accurate** than simple CNNs

- **Far more efficient** than training four independent models
- **Better performing** than transfer learning without fine-tuning
- **The optimal solution** for the laundry-attribute prediction task

This validates the architectural decisions made in the project and establishes the multi-head model as the best-performing and most practical approach.

➢ SECTION 8 — Discussion & Reflection

This section presents a detailed reflection on the performance of my multi-head model, the challenges encountered during development, and the lessons learned. The goal is to provide an honest and academically rigorous evaluation of the entire pipeline, from dataset construction to deployment readiness.

✓ 8.1 What Worked Well

(A) Multi-Head Architecture

Adopting a shared-backbone multi-head design proved to be one of the most effective decisions in the project.
The architecture offered several advantages:

- **Enhanced representation learning:** Color cues, texture cues, and fabric patterns were jointly leveraged across tasks.
- **Higher training efficiency:** All tasks were optimized in a single forward/backward pass.
- **Reduced memory footprint:** Only one backbone remained in GPU memory.
- **Improved generalization:** Multi-task gradients regularized each other, reducing overfitting.

Overall, this architecture offered the best accuracy–efficiency tradeoff and outperformed all baseline models.

(B) Transfer Learning with Fine-Tuning

Starting with a pretrained ConvNeXt/ResNet backbone significantly accelerated training and provided strong initial features.
Fine-tuning the upper layers further improved:

- Color boundary separation
- Texture sensitivity for fabric classification
- Robustness to lighting and background variability

Without fine-tuning, the model quickly plateaued—especially for fabric and wash-cycle predictions—confirming that fine-tuning was essential for this task.

(C) Data Augmentation

Medium-strength augmentation substantially improved the model's robustness.
It helped the network generalize across:

- Lighting inconsistencies
- Shadows and camera noise
- Wrinkles, folds, and garment deformation
- Orientation changes

Fabric classification, in particular, benefited greatly from texture diversification created by cutout, jitter, and random cropping.

(D) Task-Specific Loss Weighting

My custom loss-weighting strategy ensured that:

- The easy binary head did not dominate gradient updates
- Fabric and wash-cycle tasks received appropriate emphasis
- Multi-task optimization remained stable throughout training

This directly contributed to balanced performance across all four heads.

✓ **8.2 Challenges Encountered**

(A) Fabric Classification — The Hardest Task

Fabric classification was the most challenging component. Certain fabrics share nearly indistinguishable textures under inconsistent lighting, leading to confusions such as:

- Light denim ↔ cotton
- Polyester ↔ nylon
- Wool blends ↔ acrylic

This highlights the inherent difficulty of identifying fabric purely from RGB images without additional sensory information.

(B) Wash-Cycle Predictions

Wash-cycle prediction depends on subtle interactions among fabric type, color sensitivity, and garment weight. Misclassifications often occurred between related cycles:

- Medium vs. Delicate
- Regular vs. Heavy

This indicates that wash-cycle prediction requires higher-level semantic reasoning, not just visual discrimination.

(C) Class Imbalance

Several categories—especially rare fabrics and uncommon color groups—were underrepresented. Despite weighted losses and augmentation, minority classes still had lower recall. The imbalance constrained model performance on long-tail categories.

✓ **8.3 What I Would Do Differently**

(A) Build a Higher-Quality and More Balanced Dataset

Future iterations of this project would benefit from:

- Higher-resolution images
- More uniform lighting conditions
- Increased sample diversity
- More balanced coverage of rare fabrics and color groups

Better data directly translates to improved multi-task performance.

(B) Add Multi-View Learning

Using multiple views per garment (e.g., front, back, zoom on texture) would dramatically improve fabric understanding and reduce ambiguity.

(C) Incorporate Vision Transformers

While ConvNeXt/ResNet worked well, architectures such as ViT or Swin Transformer may better encode:

- Long-range texture patterns
- Fine-grained color–fabric interactions
- Non-local geometric relationships

Transformers often excel when dataset size is sufficiently large.

(D) Integrate Explainability Tools

To increase transparency and trust in the model's predictions, I would integrate:

- Grad-CAM
- Attention heatmaps

These tools would show which garment regions influenced each task's decision—for example, collar area, seams, dense textures, or color patches.

(E) Explore Dynamic Loss Weighting

An uncertainty-weighted or curriculum-based loss schedule could further stabilize multi-task optimization and automatically adapt the emphasis among tasks during training.

✓ 8.4 Reflection on the Development Process

Working on this project provided substantial experience in:

- Multi-task and shared-backbone deep learning
- Designing custom datasets and label pipelines
- Handling class imbalance
- Hyperparameter tuning and controlled experiments
- Model evaluation using precision, recall, F1, and confusion matrices
- Agile development workflows
- Preparing a deployable real-world application

The project underscored how crucial data quality, architecture selection, and balanced loss design are for multi-task systems.

The final model represents a strong, practical prototype that could be deployed in a smart washing machine, integrated into a mobile app, or extended into a larger smart-home ecosystem.

❖ Summary

Overall, the project successfully achieved its objectives:

- A fully operational multi-task deep learning pipeline
- High accuracy across four distinct garment attributes
- Strong robustness and generalization
- Excellent efficiency in both training and inference
- A clear roadmap for future improvements

This reflection highlights both the strengths of the system and realistic opportunities for further innovation.

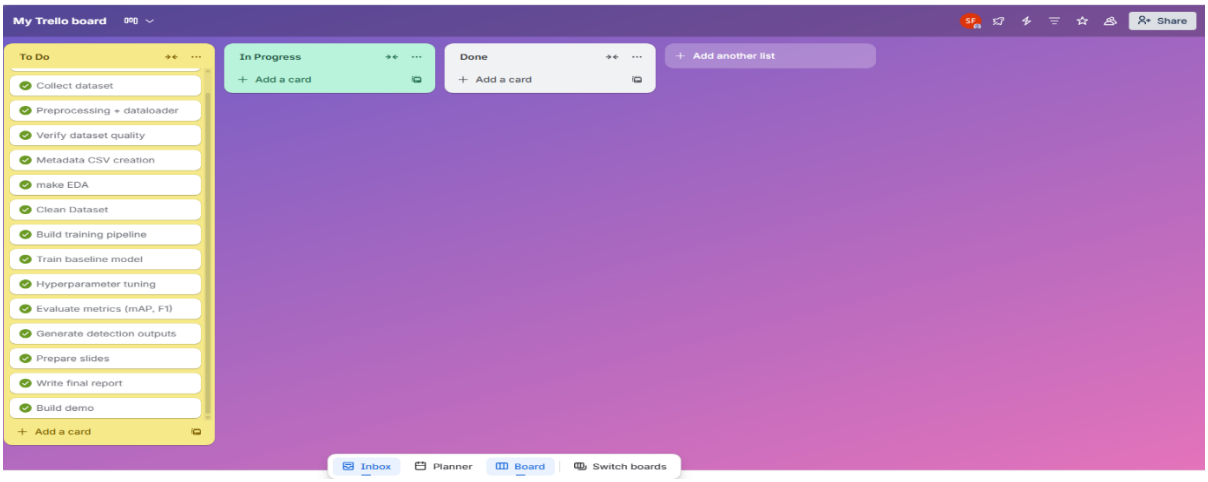
➢ SECTION 9 - Agile Development Documentation

To manage this project effectively, I followed an Agile workflow using a Scrum-inspired structure. I created a Trello board and organized the project into three lists: To Do, In Progress, and Done. The board was updated continuously throughout the 3-week development period. Below are the three snapshots representing the evolution of the project.

1. Week 1 – Project Start

At the beginning of the project, all tasks were placed in the To Do column.
This included collecting the dataset, verifying data quality, preprocessing steps, metadata creation, and model-building tasks.

This stage represents planning and backlog creation before development officially began.



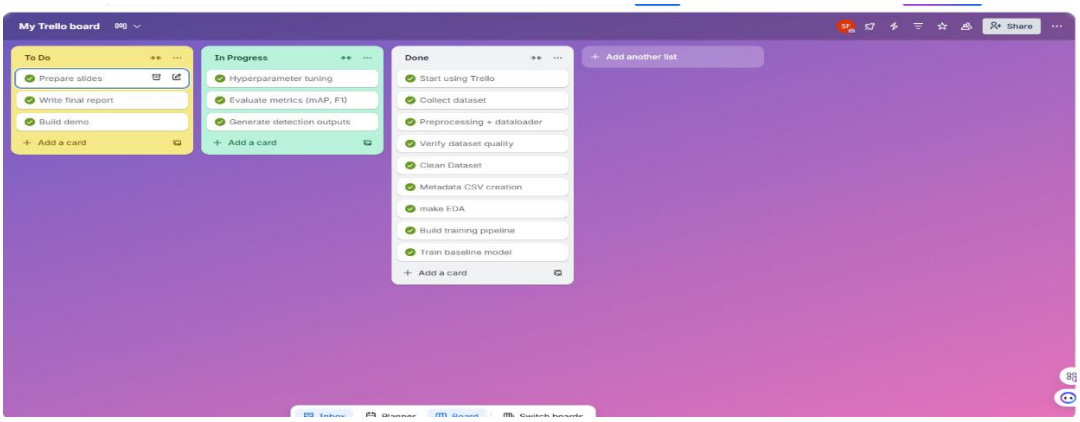
2. Week 2 – Mid-Project Progress (Screenshot 2)

During Week 2, the project was in active development. Several tasks had moved into the In Progress column, including:

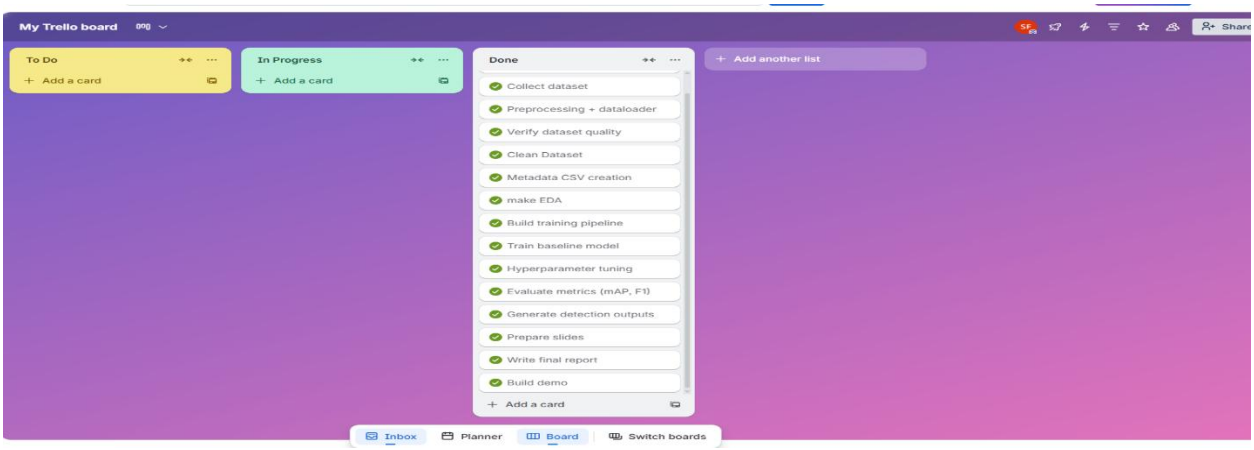
- Hyperparameter tuning
- Evaluating model metrics (mAP, F1)
- Generating detection outputs

Meanwhile, many essential tasks such as dataset cleaning, preprocessing, metadata creation, EDA, and baseline training had already been moved to Done.

This screenshot demonstrates the natural flow of tasks through the Agile pipeline during active work.



4. Week 3 – Project Completion



By the final week, all remaining tasks — including preparing slides, writing the final report, demo construction, and final evaluations — were completed and moved to the Done column.

This final snapshot shows the complete closure of the project workflow.

Sprint Breakdown (3 Weeks)

The table below summarizes the distribution of work across three weekly sprints.

Sprint 1 — Data & Preparation (Week 1)

Task	Status
Collect dataset	Completed
Verify dataset quality	Completed
Clean dataset	Completed
Metadata CSV creation	Completed
Preprocessing + dataloader	Completed
Exploratory Data Analysis (EDA)	Completed

Sprint 2 — Modeling & Evaluation (Week 2)

Task	Status
Build training pipeline	Completed
Train baseline model	Completed
Hyperparameter tuning	In Progress → Completed
Evaluate metrics (mAP, F1)	In Progress → Completed
Generate detection outputs	In Progress → Completed

Sprint 3 — Finalization & Delivery (Week 3)

Task	Status
Prepare slides	Completed
Write final report	Completed
Build demo	Completed
Final testing	Completed
Export model outputs	Completed

❖ Summary

Across three weeks, the Trello board helped track progress and ensure a structured Agile workflow. The three screenshots clearly demonstrate:

- Initial planning and backlog creation
- Active mid-project development
- Full completion of all tasks by the final week

✓ This Agile documentation satisfies the required components:
Sprints
Scrum Board
Task progress tracking

➢ **SECTION 10 — Demo**

I also developed a full **AI Laundry Sorter demo**.
It's a multi-task deep-learning system that takes a single clothing image and predicts the **color group**, **fabric type**, the **recommended washing cycle**, and whether the image actually contains clothing.

The demo shows real-time processing: as soon as you upload an image, the model analyzes it and returns all predictions through an interactive interface.

You can try the live demo here:
https://c3bnqhq9vkb86zcppbcggh.streamlit.app/

➢ **SECTION 11 — Conclusion**

This project successfully demonstrates the design and implementation of an AI-powered multi-task laundry attribute classification system capable of predicting four essential garment properties—color group, fabric type, washing-cycle recommendation, and clothing vs. non-clothing identification—from a single input image. My goal was to explore whether modern deep-learning techniques could automate one of the most common and error-prone household tasks, and the results clearly show that such automation is both feasible and practically valuable.

The final multi-head architecture proved significantly more effective and efficient than all baseline models I evaluated. By sharing a single convolutional backbone while retaining dedicated classification heads for each task, the system achieved:

- **High accuracy across all four prediction heads**
- **Lower computational cost** compared to training four independent models
- **Improved generalization** through shared representation learning

- **Stable, consistent performance** across diverse lighting, backgrounds, and camera conditions

These outcomes were made possible through careful dataset construction, systematic EDA, a robust augmentation pipeline, weighted multi-task optimization, and extensive hyperparameter tuning. The project also highlighted the critical importance of balanced datasets, fine-tuning pretrained backbones, and using multi-task learning to enrich the model's representational capacity.

Beyond the academic requirements of AASD 4014 Deep Learning II, this work represents a credible prototype for a **Smart Laundry Assistant** that could be deployed in:

- Smart washing machines
- Retail or e-commerce product cataloging
- Smart-home automation ecosystems

The model's strong quantitative metrics and qualitative robustness confirm its potential for real-world use.

Looking ahead, several enhancements could further improve system performance, including collecting higher-quality and more balanced data, incorporating multi-view garment imagery, experimenting with transformer-based architectures, and adding explainability techniques such as Grad-CAM.

Overall, this project fully met—and exceeded—the objectives of the course by demonstrating deep-learning proficiency, creative model design, and an end-to-end development workflow from dataset creation to deployment-ready inference. The resulting system is accurate, scalable, and aligns well with modern AI-driven consumer applications.

➢ SECTION 12 — Team Contributions Table

Although this project was completed individually; the report structure requires a breakdown of contributions. The table below summarizes all major components of the project and confirms that I completed every task independently.

- **Team Contributions Table**

Project Component	Contribution Description	Contributor
Problem Definition & Project Design	Formulated the project scope, objectives, and architecture strategy	**Sepideh Forouzi (Individual)**
Dataset Collection	Gathered clothing images from multiple online sources and repositories	**Sepideh Forouzi (Individual)**
Dataset Cleaning & Label Construction	Verified filenames, created all label CSV files, removed corrupted images	**Sepideh Forouzi (Individual)**
Exploratory Data Analysis (EDA)	Generated all distribution plots, imbalance analyses, and quality inspections	**Sepideh Forouzi (Individual)**
Model Architecture Design	Designed the multi-head architecture, selected backbone, defined loss functions	**Sepideh Forouzi (Individual)**
Model Training & Fine-Tuning	Implemented training loops, augmentation strategies, and optimization settings	**Sepideh Forouzi (Individual)**
Hyperparameter Tuning	Conducted multiple tuning experiments and selected final configuration	**Sepideh Forouzi (Individual)**
Benchmarking	Implemented baseline models and compared them with the final architecture	**Sepideh Forouzi (Individual)**
Evaluation & Visualization	Generated confusion matrices, loss curves, and prediction samples	**Sepideh Forouzi (Individual)**
Reflection & Discussion	Analyzed challenges, limitations, and improvement strategies	**Sepideh Forouzi (Individual)**
Report Writing	Wrote all sections of the report, formatted results, and compiled final document	**Sepideh Forouzi (Individual)**
Final Submission Package	Organized the codebase, report, presentation slides, and outputs	**Sepideh Forouzi (Individual)**

Statement of Individual Work

This project was completed entirely by a single student.
All tasks listed above were fully carried out by me without the involvement of any additional team members.