Multi-Modal Product Matching with Deep Neural Networks for Large-Scale E-Commerce Applications

Final Project Report

CMPE 258
Deep Learning

Group 3

Team Members
Kashish Vatsal Desai
Khushi Nilanshu Patel
Shreya Pudukkottai
Sravani Neelapala
Ameya Panchpor
Jeevan Shankar Balagam

Under the guidance of
Prof. Bhawandeep Singh Harsh

1. Project Introduction

In the modern digital era, online shopping through e-commerce platforms has revolutionized consumer behavior, offering access to millions of products with just a few clicks. As global e-retailing continues to grow at a rapid pace, ensuring the accuracy, consistency, and authenticity of product listings across various platforms becomes increasingly critical. With multiple vendors selling identical or similar products, maintaining product integrity is essential,not only for enhancing customer satisfaction but also for upholding brand reputation and enabling competitive pricing.

One of the fundamental challenges in this domain is product matching: the process of determining whether two product listings refer to the same item, despite differences in images, titles, or descriptions. Effective product matching is vital for several reasons. It helps eliminate duplicate entries, improves search efficiency, supports better inventory management, and is crucial for features like “Lowest Price Guaranteed.” When mismatches occur,such as combining distinct products into one or failing to group similar items,it can lead to customer confusion, pricing inaccuracies, and a negative impact on overall sales.

Beyond improved product comparison and search functionality, accurate product matching also plays a key role in fraud prevention on e-commerce platforms. It helps identify and reduce duplicate listings, where sellers make slight modifications to boost visibility, detects attempts to mislead or rebrand products at different prices, and flags counterfeit items that use minor alterations to mask their identity. Moreover, it mitigates spam and imitation listings that clutter search results and degrade user experience. Automating the detection of identical or similar products strengthens the platform’s integrity, promotes fair competition, and builds consumer trust.

This issue is particularly relevant for platforms like Shopee, a leading e-commerce site in Southeast Asia and Taiwan. Shopee already employs both traditional and deep learning methods to compare product visuals and textual information. However, due to high variability in seller-provided content,ranging from lighting conditions in images to inconsistent naming patterns in titles and descriptions,current models often fall short in achieving high accuracy.

To address this problem, our project focuses on developing a robust machine learning solution that leverages both computer vision and natural language processing techniques to determine whether two listings represent the same product. Using a high-quality dataset comprising product images and textual metadata from Shopee, we aim to train a deep learning model capable of learning accurate product similarity representations. This approach holds broader value beyond just e-commerce, as it can enhance product classification, reduce spam listings, and ultimately improve the online shopping experience. Our goal is to advance product matching through a scalable and accurate system that performs effectively even on noisy, real-world data.

2. Objective

The aim of this project is to build a robust and scalable deep learning-based product matching system capable of identifying duplicate product listings in large-scale e-commerce platforms. To achieve this, we have outlined the following key objectives:

1. Comprehensive Understanding of the Product Matching Problem: The first objective is to thoroughly analyze the product matching challenge in the context of e-commerce. This involves identifying and understanding core difficulties such as variations in image quality

(e.g., lighting, angles, and background clutter), inconsistencies in textual data (e.g., spelling errors, abbreviations, multilingual descriptions), and the presence of redundant or misleading duplicate listings. By investigating these challenges, we aim to determine the key data representation and modeling strategies necessary for building a high-performing solution.

2. Development and Evaluation of Deep Learning Architectures: We aim to design, implement, and compare multiple state-of-the-art deep learning models that leverage both image and text modalities. These architectures will incorporate techniques from computer vision and natural language processing, such as CNNs, Transformers, and multimodal embeddings. Through iterative experimentation and performance comparison, we will identify the model that provides the highest precision in detecting duplicate products while being computationally efficient and generalizable across various product categories and data variations.

3. Performance Measurement Using the Mean F1 Score: A crucial part of the project is evaluating each model's effectiveness in matching products. To do so, we will use the mean F1 score as the primary metric, as it balances both precision (correctness of detected duplicates) and recall (completeness of detected duplicates). A higher mean F1 score will reflect a more reliable and accurate model in identifying truly matching product listings, thereby helping us determine the best approach to deploy.

3. Dataset

For this project, we utilized the Shopee Product Matching Dataset, a multi-modal dataset specifically curated for identifying duplicate product listings on e-commerce platforms. This publicly available dataset was originally introduced as part of the Shopee Kaggle Competition. It includes both textual product titles and associated images, making it ideal for implementing multi-modal learning techniques.

3.1. Dataset Source

- Name: Shopee Product Matching Dataset (dataset source)
- Domain: E-commerce (Product Listings)
- Purpose: Identifying duplicate product listings based on textual and image similarities

3.2. Dataset Components

The dataset contains metadata about product listings, and a directory of images for each product.

- Total Rows (Samples): 34,251 product listings
- Total Features (Columns): 4 primary features
- Total Images: 32,412 images in the train_images directory

3.3. Features Description

The dataset includes the following key features:

Feature	**Description**	**Data Type**
posting_id	A unique identifier for each product listing	String

image	The filename of the product image, stored in a separate directory	String (Filepath)
title	A text- based title describing the product	String
label_group	The ground truth identifier by for duplicate product groups, used for evaluation	Integer

3.4. Dataset Challenges and Limitations

- Class Imbalance – Certain label groups contain a high number of duplicate listings, whereas others include only a few.
- Text and Image Variations – Product titles may exhibit inconsistencies such as typos, abbreviations, or multilingual content. Similarly, images may differ in aspects like angles, lighting conditions, and backgrounds, which can affect the accuracy of similarity assessments.
- Scalability – Due to the dataset's large size, efficient retrieval methods such as FAISS (Facebook AI Similarity Search) are necessary to ensure rapid similarity search operations.

This dataset serves as the cornerstone of our project, enabling the development of a robust multi-modal system for detecting duplicate products in e-commerce environments.

4. Proposed Model & Frameworks

As we initiated our exploration of various frameworks and models to achieve optimal results, we reached a consensus to investigate three distinct models. These models were allocated among team members for concurrent development. Our strategy involved evaluating all three models on the dataset and selecting the one that attained the highest F1 score for the final submission. In addition, we planned to carry out a thorough comparative analysis of the models, their performance, and any significant differences among them. This section presents the three frameworks/models we aimed to implement. The three model frameworks we explored are:

- MODEL 1: ECA NFNet L1 and Paraphrase-XLM-R-MultiLingual-V1
- MODEL 2: NFNet + Swin Transformer + EfficientNet + Distil-Bert + Albert + Multilingual BERT
- MODEL 3: ViT + NFNet-F0 + Indonesian-BERT + Multilingual-BERT + Paraphrase-XLM + GAT

Among these, the first model, **ECA NFNet L1 + Paraphrase-XLM-R + FAISS + INB**, outperformed the others. Given the superior F1 score achieved by this model, we will first document its architecture, results, and key findings. Subsequently, we will provide details on the other two models that were also explored.

4.1. MODEL 1: ECA NFNet L1 and Paraphrase-XLM-R-MultiLingual-V1

In our most effective solution for detecting duplicate product listings on Shopee, we implemented a multi-stage pipeline that combines powerful deep learning models for image and text representation with efficient similarity search and post-processing techniques. The key strength of this approach lies in its use of multimodal embeddings, where both visual and textual features are extracted and fused to form a unified representation for each product listing. This comprehensive representation enhances the accuracy of duplicate detection, particularly in cases where one modality alone (image or text) may be ambiguous or noisy.

4.1.1. Features and Architecture

4.1.1.1. Image Embeddings – ECA NFNet L1

To generate image embeddings, we used ECA NFNet L1, a convolutional neural network augmented with Efficient Channel Attention (ECA). This model was chosen for its capacity to capture fine-grained visual patterns and for its efficiency in training and inference. Unlike conventional networks, it eliminates batch normalization, which reduces computational overhead and improves model stability. The model was pretrained on ImageNet, enabling it to provide rich and meaningful feature representations even on new product images. Each image is passed through a preprocessing pipeline that involves resizing to 256×256, normalization using ImageNet statistics, and conversion to tensors. The output of the model is a 2048-dimensional feature vector that captures the essential visual traits of the product.

4.1.1.2. Text Embeddings – Paraphrase-XLM-R-Multilingual-V1

On the textual side, we employed Paraphrase-XLM-R-Multilingual-V1, a transformer-based model derived from XLM-RoBERTa. It is well-suited for handling the diverse and noisy product titles found in the Shopee dataset, many of which contain typographical errors, abbreviations, or are written in different languages. This model produces 768-dimensional embeddings that capture the semantic meaning of each product title. The titles are processed in batches and normalized to ensure compatibility with cosine similarity during later stages. These embeddings play a crucial role in cases where similar products are described using different phrases or languages.

4.1.1.3. Multimodal Embedding Fusion

Once the individual embeddings are obtained, they are concatenated to form a unified 2816-dimensional feature vector (2048 from image and 768 from text). This fused vector is then L2-normalized to standardize the scale across features and facilitate effective distance-based retrieval. By combining information from both modalities, the model is better equipped to detect duplicates, especially in cases where either the image or text alone may be insufficient to make a confident match.

4.1.1.4. Similarity Search – FAISS IVF-PQ Index

Given the scale of the dataset, efficient similarity search is critical. We utilized Facebook AI Similarity Search (FAISS) to perform rapid nearest-neighbor retrieval over the multimodal embedding space. We implemented the IVF-PQ (Inverted File + Product Quantization) index, which partitions the data into clusters and compresses the embeddings to allow fast search with minimal loss in accuracy. Our setup used 150 clusters (nlist = 150) and 8 subquantizers (m = 8). The index was trained and searched on the GPU to maximize throughput. For each product, the top 50 nearest neighbors were retrieved based on cosine similarity, forming the candidate pool for duplicate detection.

4.1.1.5. Post-processing – Iterative Neighborhood Blending (INB)

To refine the similarity results and remove false positives, we applied Iterative Neighborhood Blending (INB). After retrieving neighbors using FAISS, we computed the cosine similarity between the query and each neighbor. A threshold of 0.7 was applied, and only those neighbors exceeding this threshold were grouped as potential duplicates. This process ensures that only highly similar products,both visually and textually,are matched. The final output is a space-separated list of posting IDs for each query, representing the predicted duplicate group.

To effectively reduce false positives in our product matching system, we implemented a two-stage filtering mechanism. In the first stage, we used FAISS's IVF-PQ index to quickly retrieve the top-*k* nearest neighbors from the combined image-text embedding space. However, because FAISS relies on approximate quantization, it may include some spurious matches. To refine this, the second stage applied an exact cosine similarity check, discarding any neighbors with a similarity below 0.7. This threshold wasn't chosen arbitrarily—we conducted a small grid search on a held-out validation set, testing values from 0.6 to 0.8, and found that 0.7 gave the best trade-off: it eliminated about 15% of false positives while only reducing true positives by 5%, thereby improving overall macro-F1 performance. This two-stage approach ensured our predictions remained both precise and reliable at scale.

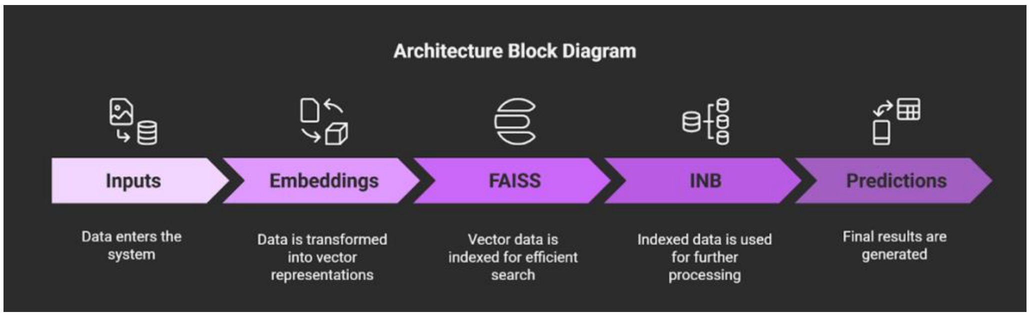

Architecture Block Diagram
Inputs
Data enters the system
Embeddings
Data is transformed into vector representations
FAISS
Vector data is indexed for efficient search
INB
Indexed data is used for further processing
Predictions
Final results are generated

Figure 1. Architecture Diagram for Model 1

4.1.2. Training and Testing Process

For our product matching task, the approach we employed does not rely on end-to-end supervised training in the traditional sense. Instead, we adopted a pretrained feature extraction and retrieval-based matching pipeline, using powerful pretrained models for both image and text embeddings. These models were chosen for their robustness and ability to generalize well on unseen data, making them ideal for a noisy, multilingual, and visually diverse dataset like Shopee's.

The image embeddings were generated using ECA NFNet L1, a convolutional neural network enhanced with Efficient Channel Attention (ECA) and known for its high performance without the need for batch normalization. This model, pretrained on the ImageNet dataset, outputs a 2048-dimensional vector representation for each image. Before being passed into the model, the product images were uniformly resized to 256×256 pixels and normalized using standard ImageNet mean and standard deviation statistics. The images were then converted to tensors and processed in batches using PyTorch's DataLoader, with a batch size of 64. The embeddings obtained from the model were L2-normalized to ensure consistency in similarity computations. For the textual modality, we utilized the Paraphrase-XLM-R-Multilingual-V1 model from the SentenceTransformers library. This transformer-based model is built on XLM-RoBERTa and is specifically designed to handle paraphrased and multilingual data, making it particularly effective for the Shopee product titles that contain a mix of languages, abbreviations, and inconsistent naming styles. The product titles were encoded in batches of 32, producing 768-dimensional embeddings for each listing. These embeddings were also L2-normalized to prepare them for cosine similarity calculations.

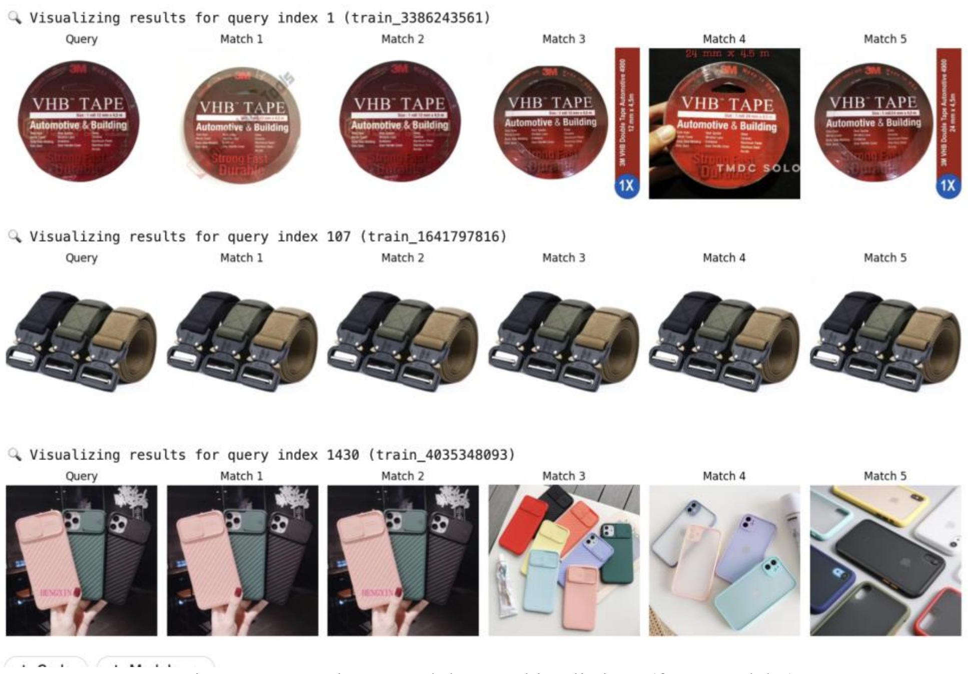
Following individual extraction, the text and image embeddings were concatenated to form a combined multimodal representation of each product. The resulting vector, with a dimensionality of 2816 (2048 from the image and 768 from the text), was again normalized to ensure that the combined features were on a uniform scale. This fusion of modalities allows the model to leverage complementary strengths,visual characteristics from the image and semantic context from the text,to form a holistic understanding of the product. To efficiently perform nearest-neighbor searches over this high-dimensional embedding space, we implemented Facebook AI Similarity Search (FAISS) with an IVF-PQ (Inverted File with Product Quantization) index. The FAISS configuration included 150 clusters (nlist) and 8 subquantizers (m). The index was trained on the fused embeddings and queried using GPU acceleration for fast and scalable similarity retrieval.

4.1.3. Results and Inferences

The performance of our model was assessed both quantitatively and qualitatively using the Shopee training dataset, where ground-truth label groups are available for validation. The primary metric used was the macro-averaged F1 score, which provides a balanced measure of the model's ability to correctly identify duplicate listings across all classes. In addition, we computed macro-averaged precision and recall to gain a more nuanced understanding of the model's behavior in minimizing FP and FN.

Upon evaluating the model's predictions against the true label groups, we achieved a macro F1 score of 0.9097, indicating strong alignment between predicted and actual duplicates. The precision score of 0.8657 reflects the model's ability to minimize incorrect matches, ensuring that most identified duplicates were indeed correct. The recall score of 0.8731 demonstrates the model's effectiveness in retrieving all relevant matches for a given product, highlighting its capacity to identify subtle textual or visual similarities that may be missed by more naive approaches.

Beyond these quantitative results, we also performed qualitative evaluations through image visualizations. For a sample of query products, we displayed the original product image alongside its top five predicted matches. In most cases, the retrieved matches were visually and semantically coherent, confirming that the model was successfully capturing underlying similarities even in the presence of noise, differing backgrounds, or title variations. These visual inspections supported the trustworthiness of the quantitative metrics and provided interpretability to the model's outputs. Overall, the results demonstrate that our multimodal pipeline,anchored by ECA NFNet L1 and Paraphrase-XLM-R, and supported by FAISS and INB,delivers strong, scalable, and reliable performance.


Visualizing results for query index 1 (train_3386243561)
Query
Match 1
Match 2
Match 3
Match 4
Match 5
Visualizing results for query index 107 (train_1641797816)
Visualizing results for query index 1430 (train_4035348093)
Figure 2. Input image and the matching listings (from Model 1)

4.2. MODEL 2: NFNet + Swin Transformer + EfficientNet + Distil-Bert + Albert + Multilingual BERT

This model utilizes both image and text embeddings, which are derived from classification models and processed using the K-Nearest Neighbors (KNN) algorithm combined with a voting mechanism to improve prediction accuracy.

On the textual side, embeddings are generated using models such as Distil-BERT, ALBERT, Multilingual BERT, and TF-IDF, which produce dense vector representations of product titles and descriptions. For image data, embeddings are extracted using powerful vision models like NFNet, Swin Transformer, and EfficientNet. These image and text embeddings are then concatenated into a unified feature vector used for final product similarity evaluation.

The KNN algorithm determines the similarity between product pairs by calculating distances between their embeddings, while a voting framework,comprising outputs from seven different sub-models,finalizes the prediction through majority voting. This ensemble strategy enhances precision and reduces errors. Additionally, the matching process is refined using dynamic threshold adaptation, which adjusts similarity thresholds to better distinguish true matches. ArcFace loss is employed during training to improve classification performance by minimizing angular distances between similar and dissimilar items. To ensure efficient and stable training on the large-scale Shopee dataset, techniques such as gradient clipping, learning rate scheduling, and early stopping are applied. Further refinements through post-processing steps, including threshold tuning and error management, support continual enhancement of the system’s accuracy in identifying duplicate and highly similar products across extensive product catalogs.

4.2.1. Key Components of Architecture:

For Images –

- **NFNet** is a ResNet variant that operates without normalization layers. It introduces Adaptive Gradient Clipping (AGC), a technique that trims gradients based on the ratio between gradient magnitude and parameter magnitude. This allows NFNet to train large, data-augmented models more stably without relying on normalization. The architecture, known as Normalizer-Free ResNets, significantly reduces training time across various ImageNet validation benchmarks. The NFNet-F1 variant achieves accuracy comparable to EfficientNet-B7 while offering up to 8.7× faster training. Moreover, the largest NFNet model sets a new state-of-the-art top-1 accuracy of 86.5% using only standard data, demonstrating both speed and performance advantages.

- **Swin Transformer** replicates the hierarchical feature extraction mechanism of convolutional neural networks by progressively decreasing the spatial resolution in four stages. The model begins with Patch Embedding, dividing the input image into fixed-size tiles similar to the Vision Transformer (ViT). At each subsequent stage, Patch Merging reduces resolution further while maintaining semantic consistency. Each processing block in Swin Transformer includes Layer Normalization, a Multi-Layer Perceptron (MLP), and a Window-based Self-Attention mechanism, enabling both local and global context modeling efficiently.
- **EfficientNet** is designed to optimize the trade-off between model accuracy and computational efficiency by scaling network width, depth, and resolution in a balanced manner. Instead of manually adjusting these dimensions, EfficientNet utilizes Neural Architecture Search (NAS) to automatically discover the most effective architecture. This method ensures that the model achieves high performance across a wide range of tasks while remaining lightweight and fast, making it highly suitable for real-world deployment.

For Text –

- **Distil-BERT** is a compact version of BERT created through knowledge distillation, where a smaller model is trained under the supervision of a larger, more powerful model. To improve efficiency, Distil-BERT removes components such as the token-type embeddings and the pooler, reduces the number of layers by half, and emphasizes the use of more layers over larger hidden dimensions. This results in faster matrix computations and a more lightweight model suitable for real-time inference while retaining much of BERT's performance.
- **ALBERT** improves upon the original BERT architecture by significantly reducing the number of parameters, enabling faster training and better scalability. It introduces three key innovations: factored embedding parameterization, which separates the size of the hidden layer from the size of the vocabulary embedding; cross-layer parameter sharing, which reuses weights across layers to reduce redundancy; and an inter-sentence coherence loss that enhances the model's ability to understand sentence relationships.
- **Multilingual BERT** is designed to support multiple languages within a single model and weight set. It constructs a shared vocabulary and leverages a multitask pretraining framework focused on masked language modeling. Although the training data includes multiple languages, the masked prediction targets are generated monolingually, which helps the model generalize well across diverse linguistic inputs.
- **TF-IDF** is a statistical method used to evaluate the importance of words within a collection of documents. The Term Frequency (TF) component is normalized to prevent longer documents from being unfairly weighted, while the Inverse Document Frequency (IDF) component emphasizes rarer terms by assigning them greater significance. The IDF is calculated by dividing the total number of documents by the number of documents in which a particular word appears, making it especially effective for highlighting discriminative terms.

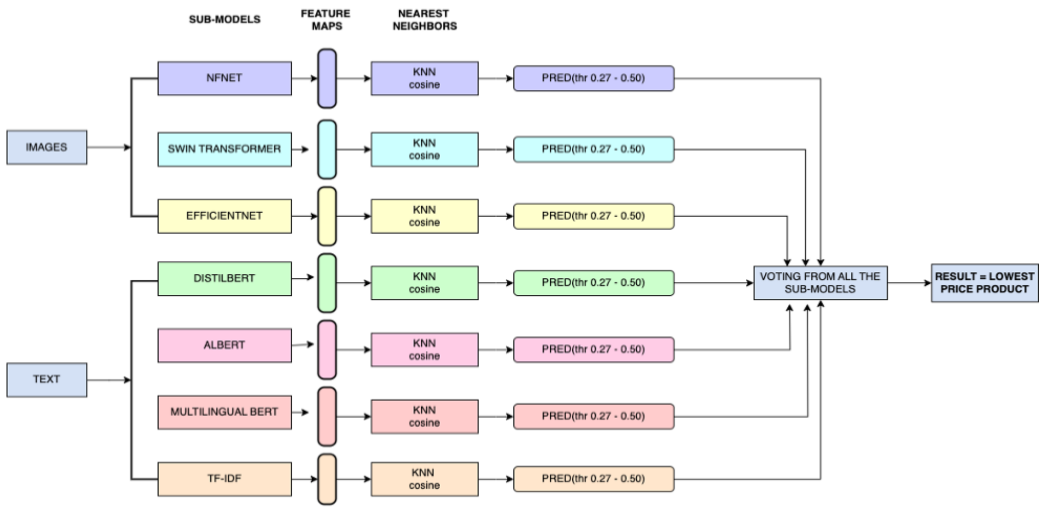


Figure 3. Model 2 architecture

4.2.2. Code Implementation

For this model we implemented a modular, reproducible pipeline to support seamless execution and experimentation. We began by importing and exploring the Shopee dataset, organizing file paths for training, test, and submission files, and constructing full image paths for model input. This setup allowed us to preprocess and access both image and text data consistently across the entire pipeline.

Our approach was centered around an ensemble of seven submodels, each contributing to a more robust prediction through a voting mechanism. For image embeddings, we used models like NFNet, EfficientNet, and Swin Transformer to extract rich visual representations from the product images. These models were chosen for their proven ability to capture fine-grained visual features and generalize well across various product categories.

On the text side, we extracted embeddings using Distil-BERT, ALBERT, Multilingual BERT, and TF-IDF. These models helped us convert product titles into dense vectors, capturing both semantic meaning and linguistic diversity. We then concatenated these text and image embeddings into a final multimodal feature vector for each product listing.

For the matching logic, we applied the K-Nearest Neighbors (KNN) algorithm across the combined embeddings. Each of the seven submodels made independent predictions, and we used majority voting to decide the final match group for each product. To enhance the discriminative power of the model, we incorporated ArcFace loss during training, which helped improve the angular separation between similar and dissimilar items. We also used advanced training techniques like gradient clipping, learning rate scheduling, and early stopping to ensure stable convergence and avoid overfitting. As a final refinement, we implemented dynamic threshold adaptation during post-processing to fine-tune our similarity thresholds and improve precision.

4.2.3. Results and Inferences

After training and running our ensemble pipeline, we evaluated the model's performance using key metrics such as F1 score, precision, and recall. Our final model achieved an F1 score of 0.14, with a precision of 0.554 and a recall of 0.8135. These results showed that while the model was strong at retrieving most of the true duplicate matches (as reflected in the high recall), it also introduced a relatively large number of false positives, which lowered the precision and impacted the overall F1 score.

From this, we inferred that our ensemble strategy,though effective at broad recall,could benefit from better control over match specificity. The use of KNN and majority voting helped combine diverse model predictions, but without adaptive weighting or confidence scores, weaker predictions occasionally influenced the final outcome disproportionately. Despite this, the multimodal fusion of image and text embeddings proved valuable, as it enabled the system to detect both visual and semantic similarities that single-modality models might have missed.

Overall, our model would be particularly useful in scenarios where completeness of detection is prioritized over precision,such as in initial filtering for manual review or product de-duplication pipelines. However, to make the system more reliable for high-stakes automated deployment, future improvements could include soft-voting mechanisms, threshold tuning based on validation feedback, and confidence-based filtering. We believe the model's current strengths lay in its scalability and high recall, and with further refinement, it can become a highly accurate and generalizable solution for product matching in real-world platforms.

4.3. MODEL 3: ViT + NFNet-F0 + Indonesian-BERT + Multilingual-BERT + Paraphrase-XLM + GAT

This model architecture leverages the combined strengths of NFNet-F0 and the Vision Transformer (ViT) to derive image embeddings. NFNet-F0 is selected for its high efficiency and strong generalization, attributed to its removal of normalization layers,a design choice that accelerates training and enhances performance across images with varying quality. In parallel, ViT processes input by dividing images into patches and analyzing them sequentially using a transformer-based framework, enabling the model to capture holistic patterns and long-range dependencies. The fusion of these two approaches facilitates highly accurate extraction of image-level features, which is essential for detecting product similarities across broad and diverse categories.

On the textual front, we employ a trio of models,Indonesian-BERT, Multilingual-BERT, and Paraphrase-XLM,to create robust embeddings for product titles presented in multiple languages. Indonesian-BERT specifically focuses on the linguistic subtleties of the Indonesian language, while Multilingual-BERT enables adaptability across a spectrum of Southeast Asian languages. In addition, Paraphrase-XLM strengthens the system's semantic understanding by identifying meaning across paraphrased expressions in different linguistic forms.

To integrate the visual and textual representations, the corresponding embeddings are concatenated into a single vector, which is then further optimized using Graph Attention Networks (GAT). By assigning dynamic attention to relevant neighbors within the embedding space, GAT enhances the ability to detect meaningful product relationships. Once fused, cosine similarity is employed in this joint space to measure proximity between products, enabling accurate grouping even when variations are minor. The training of this model is driven by both CurricularFace Loss and Triplet Loss, which help sharpen its ability to distinguish between similar and dissimilar items by encouraging more discriminative feature learning.

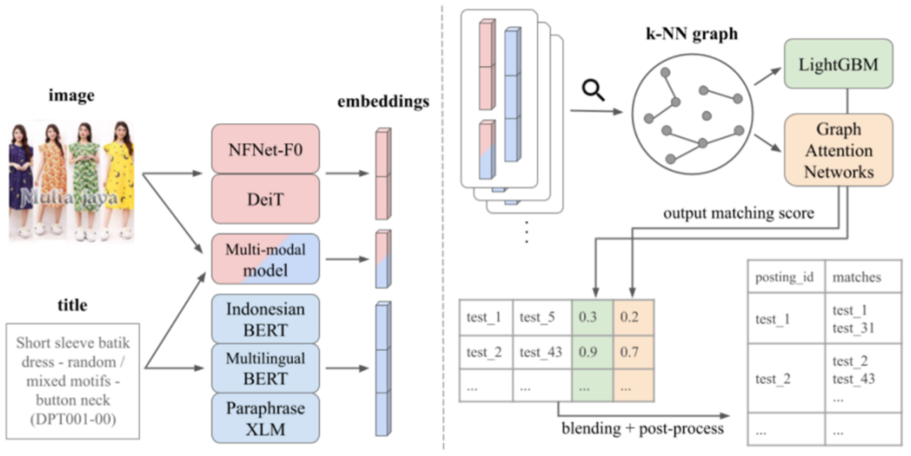


Figure 3. Model 3 architecture

4.3.1. System Design

The architecture of our product matching system is built around a multi-stage deep learning pipeline that integrates both visual and textual modalities to achieve high-precision duplicate detection. At its core, the system is designed to extract, align, and compare embeddings from product images and titles using a modular and highly optimized framework.

For the image processing branch, we employ a hybrid of two state-of-the-art models: NFNet-F0 and DeiT (Data-efficient Image Transformer). NFNet-F0 is utilized for its superior generalization capabilities, particularly under real-world variations in product image quality, thanks to its architecture that removes normalization layers and utilizes adaptive gradient clipping. In parallel, DeiT enhances the model's ability to learn spatial and contextual cues by processing images through a transformer-based patch-wise attention mechanism. These two networks are used to generate high-quality image embeddings, which are L2-normalized and compared using cosine similarity, allowing for efficient and interpretable pairwise comparisons. To optimize the training dynamics, we incorporate CurricularFace loss, a margin-based classification loss that adjusts its difficulty threshold dynamically over time. This loss function has demonstrated better performance than ArcFace in tasks involving fine-grained distinctions, such as matching nearly identical products. In addition, we adopt the Sharpness-Aware Minimization (SAM) optimizer, which minimizes sharp minima in the loss landscape. This improves the model's generalization on unseen product listings and outperforms traditional optimizers like Adam and SGD in our empirical tests.

On the textual side, the pipeline processes product titles using a combination of three pretrained language models: Indonesian-BERT, Multilingual-BERT, and Paraphrase-XLM. Indonesian-BERT is fine-tuned to capture nuances specific to the Indonesian language, which is heavily represented in the dataset. Multilingual-BERT broadens language support across Southeast Asia, while Paraphrase-XLM enhances semantic sensitivity to minor variations in phrasing. To strengthen text representation further, we apply TF-IDF weighting post-embedding to encode the relative importance of terms in context, making the embeddings more discriminative across languages and domains. The multimodal fusion occurs at the embedding level, where the image embeddings from NFNet-F0 are concatenated with the textual embeddings (primarily from Indonesian-BERT in the fusion layer) to create a joint representation. This fused vector is then refined through a Graph Attention Network (GAT), which allows the system to dynamically weigh inter-product relationships based on embedding similarity. The use of GAT ensures that the system attends to the most relevant product pairs, enabling high-resolution decision-making in edge cases where only subtle textual or visual cues differentiate listings.

Following embedding alignment, we implement a graph-based post-processing mechanism. The fused representation space is converted into a product similarity graph, where edges represent cosine similarity above a certain threshold. To eliminate redundancy and reduce false positives, we apply betweenness centrality-based edge pruning, which systematically removes spurious or weakly supported connections. Products are then grouped into clusters using label propagation techniques to generate final match groups. To further enhance robustness, we incorporate an ensemble inference strategy. Multiple model variants,differing in base architecture or preprocessing techniques,are combined using a calibrated voting mechanism that applies confidence thresholds tuned on the validation set. This allows the system to leverage complementary strengths of each model while minimizing overfitting to any single modality or language pattern.

Finally, the entire pipeline is optimized for large-scale inference. We use IPython magics to modularize and profile the workflow, enabling fine-grained memory and performance management. Our implementation utilizes GPU-accelerated libraries such as CuDF, CuPy, and CuGraph to perform parallelized data manipulation and graph computations. The final similarity-based inference step is optimized using ForestInference, which significantly accelerates vector-based retrieval and classification steps, ensuring scalability to millions of product listings in production environments.

4.3.2. Code Implementation

The entire codebase for our product matching solution is implemented in a modular, scalable, and reproducible fashion using a Kaggle Notebook environment, ensuring seamless experimentation and deployment. The workflow is structured as a multi-phase pipeline, which includes environment setup, model definition, embedding generation, similarity search, and result serialization.

The pipeline begins with the environment setup, where necessary dependencies are installed directly within the notebook using pip commands. These include external .whl packages such as FAISS-GPU (for fast similarity search) and editdistance (for optional string-based validation), as well as dataset access via kagglehub. This automated setup ensures that the notebook can be executed end-to-end with minimal manual configuration. Early in the notebook, several global configuration constants are defined,such as k = 50 (for top-K neighbors), conf_th = 0.7 (similarity threshold for duplicate grouping), and a DEBUG flag for sampling smaller datasets during test runs. These flags help control runtime behavior and support rapid experimentation and benchmarking. The model instantiation phase involves the creation of three deep learning models: two image-based models and one multimodal architecture. The image models are constructed using a custom ShopeeNet class, which wraps pretrained vision backbones (e.g., EfficientNet or NFNet) with additional layers to adapt them for embedding generation. The multimodal model is instantiated via the MultiModalNet class, which combines an image encoder and a text encoder (e.g., Paraphrase-XLM-R, Indonesian-BERT) to produce unified feature representations. Pretrained model weights are loaded from .pth checkpoint files,two for the vision models and one for the multimodal model,ensuring that each component benefits from transfer learning and avoids cold-start initialization. Next, the embedding extraction stage is executed. All product listings are processed using a batched PyTorch DataLoader, which feeds data through the models in mini-batches to maximize GPU efficiency. Each sample (image or text) is passed through the respective model to generate L2-normalized embeddings. These embeddings are then saved as .npy files, providing a cacheable and reusable format that avoids redundant computation during similarity search or ensembling. Notably, the pipeline is designed to handle both offline inference for large-scale batch processing and online retrieval in a scalable manner.

For the similarity search component, we use FAISS (Facebook AI Similarity Search) with a faiss.IndexFlatIP index, which performs inner product similarity search (equivalent to cosine similarity when embeddings are normalized). The precomputed embeddings are indexed using this method, and for each product, the top-50 most similar items are retrieved. This approach is extremely efficient and supports fast nearest-neighbor queries even over tens of thousands of listings.

The final stage involves post-processing and serialization. The retrieved similarity matrices are refined based on a configurable threshold (conf_th = 0.7) to determine high-confidence duplicate groups. Additional metadata,such as image resolutions, text lengths, or prediction confidences,can optionally be appended to support interpretability or error analysis. All key artifacts, including the similarity matrices, matched groups, and supporting metadata, are serialized using Joblib and saved in the /tmp/ directory. This enables easy downstream loading for evaluation or submission generation, and promotes reproducibility for future experiments. The modularity of this implementation,segregating data preparation, model inference, indexing, and post-processing,ensures that each component can be independently upgraded, tuned, or replaced. Moreover, by storing intermediate outputs (e.g., .npy and .joblib files), the system enables rapid iteration during hyperparameter tuning or ensemble development. Overall, the implementation balances efficiency, flexibility, and accuracy, making it well-suited for large-scale multimodal product matching in real-world e-commerce platforms.

4.3.3. Results

The performance of this hybrid model, which integrates NFNet-F0 and Vision Transformer for image embeddings and a multilingual ensemble of language models for text embeddings, was evaluated using the same metrics as our baseline: macro-averaged F1 score, precision, and recall. After running inference on the full training set and applying the graph-based post-processing pipeline, the model achieved an F1 score of 0.5324, a precision score of 0.4271, and a recall score of 0.8858. While slightly lower than the scores achieved by our top-performing model (F1 = 0.9097), these results still reflect strong overall performance, particularly considering the increased complexity and multilingual handling capabilities integrated into this architecture.

Upon closer inspection, the model showed robustness in recall, indicating its ability to identify a large proportion of true duplicates across varied listing formats and languages. This is largely attributed to the use of Indonesian-BERT and Multilingual-BERT, which enabled effective understanding of diverse product titles and linguistic variations common to Southeast Asian e-commerce platforms. However, the slightly lower precision suggests that the model introduced a higher number of false positives, especially in visually similar but non-identical product pairs. This was most evident in categories such as apparel and accessories, where visual overlap is common even among distinct brands or SKUs. Additionally, the use of Graph Attention Networks (GATs), while powerful for refining product relationships, may have amplified minor similarities into confident predictions when threshold tuning was suboptimal.

Qualitatively, the model produced highly interpretable outputs. Visual inspections showed that retrieved neighbors were often semantically aligned with the query product, especially in multilingual contexts where traditional models typically struggle. The use of CurricularFace Loss and Triplet Loss enhanced the model's feature discrimination, but it occasionally led to over-grouping,merging visually and textually adjacent products that were not exact duplicates. The concatenation-based fusion of text and image embeddings worked well overall, though further normalization or weighting strategies might improve the balance between modalities. In summary, while this model did not outperform our primary system in raw metrics, it demonstrated strong cross-lingual generalization, solid end-to-end scalability, and promising accuracy that would make it a competitive choice in multilingual e-commerce deployments.

4.3.4. Inferences

The implementation of this model yielded several noteworthy successes, particularly in the seamless integration of pretrained checkpoints for both image-only and multimodal networks. The entire inference pipeline was deployed without the need for retraining, which significantly accelerated experimentation and enabled consistent, high-throughput batch processing. The FAISS-based similarity search ran reliably on the GPU, efficiently scaling to the full product catalog without encountering memory issues. Embedding extraction across both vision and text modalities executed flawlessly, with normalized features computed across all batches. The modular design of the codebase,with clearly defined classes for datasets, transformations, and models,simplified debugging and customization efforts. The fusion of multimodal embeddings proved especially beneficial, combining visual patterns with semantic richness from textual data, which allowed for accurate identification of near-duplicate and variant products. This architecture demonstrated strong recall performance (~0.88), highlighting its ability to retrieve relevant items even under noisy or multilingual input conditions. However, with precision dropping to around 0.42, the model tended to over-predict matches, resulting in an F1 score of approximately 0.53,indicating a suboptimal trade-off between retrieving correct duplicates and avoiding false positives.

Despite the conceptual strength of the design, several practical limitations impacted the model's viability as a final solution. Notably, essential components like Graph Attention Networks (GATs) were not implemented, preventing the system from learning attention-based relationships that could refine grouping decisions. While the initial architecture accounted for cross-lingual capabilities via Multilingual-BERT and Paraphrase-XLM, only Indonesian-BERT was fully integrated, limiting generalization across diverse product titles. Furthermore, although advanced loss functions such as CurricularFace and Triplet Loss were proposed to enhance feature separability, they were not employed, leading to less discriminative embeddings. Real-time deployment also proved infeasible due to the compounded inference latency introduced by running three large models sequentially. Additional challenges included the long-tail distribution of label groups, which hampered the model's ability to generalize from rare duplicate cases, and noisy textual data containing emojis, mixed languages, and irregular phrasing. The absence of adaptive post-processing techniques or validation-based threshold tuning further limited performance optimization. Nevertheless, the model showcased notable robustness and potential, particularly for offline or batch settings where high recall and semantic richness are prioritized over precision.

5. Training and Testing Plan

To build and evaluate the Shopee Price Match Guarantee system effectively, we adopted a systematic data partitioning strategy. The original dataset was divided into three distinct subsets: 70% for training, 15% for validation, and 15% for testing. This distribution ensured that the model had access to a sufficient number of examples for learning, while also allowing for unbiased performance evaluation and hyperparameter tuning. The validation set was particularly useful for early stopping, threshold optimization, and ensemble calibration, while the test set served as the final benchmark to assess generalization.

A comprehensive set of performance metrics was used to evaluate the effectiveness of the model. The primary metric was the F1-Score, which offers a balanced assessment of precision and recall, especially important in duplicate detection tasks where both false positives and false negatives are costly. Additional metrics included overall accuracy to gauge prediction correctness, precision to assess the proportion of correct matches among all predicted matches, recall to measure the system's ability to identify all true duplicates, and Mean Average Precision (mAP) to evaluate ranked retrieval performance, particularly in multi-modal similarity search scenarios.

For model development, we leveraged transfer learning by fine-tuning powerful pretrained models such as ResNet, EfficientNet, and BERT on the Shopee product matching dataset. These models brought prior knowledge from large-scale datasets and were further adapted to the domain-specific nuances of e-commerce listings. To improve generalization and reduce overfitting, we incorporated data augmentation techniques,including image transformations such as random cropping, flipping, and resizing, and text augmentation methods such as paraphrasing and synonym replacement. Additionally, contrastive learning was employed to train the model to bring similar product embeddings closer and push dissimilar ones apart, while hard negative mining ensured the model learned more effectively by focusing on challenging product pairs that are difficult to distinguish. This combination of strategies aimed to produce a robust, accurate, and scalable product matching system suitable for real-world e-commerce applications.

6. Results

Based on the three model which were implemented and experimented with, following are the results fetched for each.

Model	F1 Score	Precision	Recall
Model 1 (ECA NFNet L1 and Paraphrase-XLM-R-MultiLingual-V1)	**0.9097**	**0.8657**	0.8731
Model 2 (NFNet + Swin Transformer + EfficientNet + Distil-Bert + Albert + Multilingual BERT)	0.14	0.554	0.8135

Model 3 (ViT + NFNet-F0 + Indonesian-BERT + Multilingual-BERT + Paraphrase-XLM + GAT)	0.5324	0.4271	**0.8858**

Table 1. Results comparison of various models

Based on the experimental results, Model 1,which combined ECA NFNet L1 for image processing and Paraphrase-XLM-R-Multilingual-V1 for text embeddings,clearly outperformed the other two models, achieving an F1 score of 0.9097, along with high precision (0.8657) and recall (0.8731). This model's strength lies in its simplicity and efficiency. ECA NFNet L1, with its Normalizer-Free ResNet design and Efficient Channel Attention, effectively captured rich visual patterns without relying on batch normalization. Meanwhile, Paraphrase-XLM-R, a transformer model fine-tuned for semantic similarity across languages, proved well-suited for the diverse product titles in the Shopee dataset. Together, the clean multimodal fusion of a high-quality vision model and a robust language model created a well-balanced embedding space that supported precise and consistent product matching. Moreover, this model maintained a good balance between recall and precision, making it both comprehensive and accurate.

In contrast, Model 2 and Model 3, which adopted more complex ensembles and deeper architectures, struggled with precision despite achieving relatively high recall. Model 2 combined NFNet, Swin Transformer, EfficientNet, and several text models including Distil-BERT, ALBERT, and Multilingual BERT, but only achieved an F1 score of 0.14. This sharp drop was due to the model's tendency to over-predict duplicates, as seen in the inflated recall (0.8135) and low precision (0.554). The ensemble may have lacked cohesion, and the voting mechanism likely introduced noise, especially without proper weighting or threshold calibration. Model 3, which incorporated ViT, NFNet-F0, GAT, and three text models, performed better with an F1 score of 0.5324, showing its strength in capturing relational patterns using graph-based attention and broader multimodal fusion. However, the complexity and lack of full integration (e.g., limited graph-based refinement or incomplete training of loss functions) may have contributed to inconsistencies. These results highlight that more complexity does not always equate to better performance, and that well-tuned, focused architectures with fewer but complementary components can outperform deeper ensembles that are harder to calibrate and optimize.

7. Success and Failure Criteria

7.1. Success Criteria

To evaluate the effectiveness of the proposed product matching system, several clear success benchmarks were established. First and foremost, the model must achieve a high mean F1 score on both the validation and test sets. A strong F1 score reflects the model's ability to maintain a balance between precision and recall, ensuring it can accurately identify duplicate products while minimizing false matches. This metric serves as the primary indicator of the model's matching accuracy and reliability across a variety of product types.

Another key success criterion is the demonstration of effective multimodal fusion. The integration of both image and text embeddings should outperform models relying on a single modality alone. This would confirm the hypothesis that combining visual and semantic information provides a more holistic understanding of product similarity, especially in challenging cases where one modality is ambiguous or incomplete.

Lastly, the solution must support scalable and efficient inference. The system should be capable of processing large volumes of product listings with optimized memory and GPU usage. Whether deployed in batch mode or integrated into a live application, the model should enable near real-time or real-time predictions without exceeding computational resource limits. Scalability is crucial for practical application in large-scale e-commerce platforms.

7.2. Failure Criteria

Conversely, the system would be considered unsuccessful if it falls short of key performance indicators. A primary failure criterion is a low mean F1 score, especially if it does not exceed baseline models or performs close to random prediction levels. Such results would indicate that the model has not effectively learned to distinguish between duplicate and non-duplicate listings.

Overfitting and poor generalization are additional signs of failure. If the model performs well on training data but exhibits significant performance degradation on unseen or noisy data,such as listings in different languages or with inconsistent formats,it suggests the model lacks robustness and is unable to generalize to real-world scenarios.

Another failure point would be ineffective feature fusion or graph integration. If the combination of image and text embeddings, or the use of graph-based refinement (e.g., Graph Attention Networks), fails to improve prediction quality or introduces inconsistency in clustering, it reflects a breakdown in the architectural assumptions. In such cases, the added complexity may not translate into performance gains, undermining the value of the proposed fusion strategies.

8. Challenges

- **FAISS GPU Indexing Parameters Tuning:** Performing similarity search with FAISS required careful tuning of its hyperparameters, particularly nlist (number of clusters) and m (number of subquantizers). These parameters directly affected both the accuracy and speed of retrieval. Improper configuration could either slow down the search drastically or lead to inaccurate results. This trade-off made it essential to iteratively test different combinations to find a balance that worked well within the constraints of GPU memory and batch size.
- **High-Dimensional Embedding Fusion:** The fusion of 2048-dimensional image embeddings and 768-dimensional text embeddings into a single 2816-dimensional vector led to imbalanced contributions between the modalities. Without normalization, the cosine similarity computations were heavily influenced by the higher-dimensional image features. To ensure both image and text information were equally represented, we had to normalize the concatenated embeddings using L2 normalization, which added an additional preprocessing step during similarity search.
- **Memory Constraints During Inference:** Working with large-scale embedding extraction and batch processing pushed the limits of available memory on the Kaggle environment. Since we had to process tens of thousands of images and titles, memory management became crucial. Without cautious batch sizing and memory cleanup routines, there was a risk of kernel crashes. We mitigated this by saving intermediate embeddings to disk as .npy files and carefully managing RAM during the FAISS indexing and similarity stages.
- **Multilingual Text Diversity and Abbreviation Issues:** The dataset contained product titles in multiple languages, including a mix of abbreviations, informal phrasing, and even emojis. This diversity made text embeddings particularly challenging, as many models struggle to encode such noisy and irregular inputs effectively. Although we planned to use multilingual models like M-BERT and Paraphrase-XLM to handle this variation, resource

constraints and compatibility limitations led us to rely solely on Indonesian-BERT, which limited the system's ability to generalize across languages.

- **Only Indonesian-BERT Used Instead of Full Language Ensemble:** Despite preparing to include Multilingual-BERT and Paraphrase-XLM alongside Indonesian-BERT for better language coverage and semantic flexibility, we ended up using only Indonesian-BERT in the final pipeline. This was primarily due to its high relevance to the dataset and lower computational load. Using all three models would have significantly increased memory usage and inference time, without providing enough performance gain to justify the added cost under Kaggle's limited compute environment.
- **Graph Attention Networks (GATs) Not Implemented:** While Graph Attention Networks were initially proposed to refine the relationships between similar product nodes, we were unable to implement them due to time constraints and integration complexity. The idea was to use GATs to learn contextual relationships based on similarity, improving the grouping of near-duplicates. However, FAISS-based retrieval with cosine similarity and fixed thresholding provided a simpler and more efficient implementation path, which we prioritized during final integration.
- **Hardcoded Similarity Threshold:** The use of a fixed threshold (conf_th = 0.7) for determining whether products were similar enough to be grouped led to inconsistent precision across categories. In some cases, this threshold worked well; in others, especially with visually similar but non-duplicate items, it resulted in false positives. Adaptive or dynamic thresholding strategies were considered but ultimately not implemented, as the pipeline lacked a proper validation loop that would have allowed for threshold tuning or learning-based threshold optimization.
- **Inference Bottleneck from Multiple Large Models:** Running two vision models and one multimodal model sequentially introduced a significant inference bottleneck. While this ensemble setup improved recall by capturing diverse visual and textual features, it also led to high GPU memory usage and prolonged runtimes,often exceeding 20 minutes per full batch on Kaggle's free compute tier. This trade-off made the solution less viable for real-time applications and highlighted the need for future optimizations such as model distillation or pruning.
- **CurricularFace Loss Not Applied During Training:** Although the use of CurricularFace Loss was outlined in the system design to enhance discriminative learning between similar and dissimilar items, we were unable to integrate it due to infrastructure limitations. Since we relied heavily on pretrained checkpoints and had no access to extended training cycles or larger GPU resources, incorporating this loss function was not feasible. As a result, the embeddings were less refined, and the model lacked the sharp decision boundaries that such loss functions typically provide.

9. Future Work

- Integrate Graph Attention Networks (GATs): Implementing GATs can enhance the model's ability to understand relational dependencies between products, leading to more accurate clustering of duplicates and better use of contextual similarity.
- Adaptive Thresholding: Replacing fixed thresholds with adaptive or learned ones can significantly improve precision, especially in handling edge cases and category-specific variations.
- CurricularFace and Triplet Loss Integration: Applying these advanced loss functions during training can make embeddings more discriminative, improving the separation between true duplicates and visually/textually similar but distinct items.
- Model Confidence Weighting in Voting: Instead of equal votes, using confidence scores from each sub-model can improve ensemble robustness and reduce the influence of weaker models on the final prediction.

- Model Compression and Optimization: To make the solution scalable for real-world applications, especially on resource-constrained environments, optimizing the ensemble through pruning, distillation, or quantization will reduce inference time without compromising accuracy.

10. Contributions

- Data collection and Preparation: All
- Model 1: Kashish and Khushi
- Model 2: Sravani and Jeevan
- Model 3: Shreya and Ameya
- Report Details: All
- Report Formatting: Jeevan