ML Challenge 2025: Smart Product Pricing Solution

**Team Name: TensorTitans**  
**Team Members:**

RAHUL BALAJI  
P Vamsi Krishna  
Sachin Uke  
Subhasis

**Submission Date:** 11 October 2025

Executive Summary

Our approach leverages a multi-modal deep learning pipeline that integrates text, image, and tabular data to predict optimal e-commerce product prices. We extract semantic embeddings from product descriptions using Sentence Transformers, visual embeddings from product images via MobileNetV2, and engineered features like brand and item pack quantity from text. These features are fused into a unified MLP model trained with MAE loss, optimized for both accuracy and runtime efficiency — achieving competitive performance while running in under 20 minutes on Google Colab.

**Methodology**

## 1. Overview

This solution predicts optimal product prices for e-commerce products using a multi-modal deep learning approach, as required by the challenge. We extract and use features from:

* Product text (catalog\_content)
* Product images (image\_link)
* Tabular features (extracted Item Pack Quantity, brand label encoding)

## 2. Feature Engineering

### Tabular Features

* **Item Pack Quantity (IPQ):** Extracted using regex from catalog\_content (e.g., "Pack of 5", "2 pcs").
* **Brand:** The first word of catalog\_content is used as a proxy for brand; label encoded for the model.

### Text Features

* **Text Embeddings:** Used [sentence-transformers/all-MiniLM-L6-v2](https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2) to generate 384-dimensional semantic embeddings for catalog\_content. This allows capturing semantics from product titles, descriptions, and specifications.

### Image Features

* **Image Embeddings:** Used [MobileNetV2](https://keras.io/api/applications/mobilenet/) pretrained on ImageNet to extract 1,280-dimensional pooled embeddings from product images. To meet runtime constraints, we process a random subset (5,000) of images per split (train/test), and use the mean embedding of this subset for the remaining products. This balances speed with multi-modal compliance.

## 3. Model Architecture

* **Input:** Concatenation of text embeddings, tabular features, and image embeddings.
* **Neural Net:** A simple but effective MLP:
  + Dense(512) + BatchNorm + Dropout(0.2)
  + Dense(256) + BatchNorm + Dropout(0.15)
  + Dense(64)
  + Dense(1, relu) for price prediction (enforces non-negative prices)
* **Optimizer:** Adam (lr=1e-3)
* **Loss:** Mean Absolute Error (MAE)
* **Early Stopping:** On validation MAE, patience=2.

## 4. Training and Validation

* **Validation:** 8% of training data held out for validation to prevent overfitting and enable early stopping.
* **Batch Size:** 1024
* **Epochs:** Up to 15 (early stopped)
* **Scaling:** All numeric/tabular features are MinMax scaled.

## 5. Output Generation

* **Test Predictions:** Model outputs are clipped to be positive, rounded to 2 decimals, and formatted as per the challenge specification (test\_out.csv with columns: sample\_id, price).

## 6. Computational Considerations

* **Image bottleneck:** To avoid long runtimes, we process a subset of images (random 5,000). The rest use the mean image embedding, a common competition trick.
* **Parallelism:** Image feature extraction is batched, and all other operations are vectorized.
* **Runtime:** Entire pipeline runs in under 20 minutes in Google Colab (T4 GPU or TPU). Most time is spent on image downloading and embedding.

## Files Provided

* fast\_multimodal\_submission.py: Main pipeline, can be run in Colab.
* test\_out.csv: Final output file for test set, ready for submission.

## Reproducibility Instructions

1. **Install requirements** (in Colab or local):pip install pandas numpy tensorflow pillow tqdm sentence-transformers gdown
2. **Upload and run fast\_multimodal\_submission.py** in your runtime.
3. **Download test\_out.csv** from your workspace after execution.

## Notes

* **No external price lookup**: This solution strictly uses only provided data and open-source models.
* **All models are MIT/Apache 2.0 licensed**.
* **All outputs are formatted as per sample output.**

## Improvements & Further Work

* Ensemble with gradient boosting (e.g., CatBoost) on tabular/text features.
* Use larger image and text models if more compute is available.
* Experiment with multimodal transformers for further improvement.