Brand Classification using EfficientNetV2B0

Chethiya Galkaduwa, Meghana Nanuvala

1. Abstract

This project explores the development of a supervised image classification model designed to predict the brand of winter apparel using a limited but diverse dataset. The problem setting is motivated by the need to automate product metadata generation for online retail platforms. We trained and fine-tuned a deep convolutional neural network using EfficientNetV2B0, a high-performing architecture that balances accuracy and computational efficiency. Leveraging transfer learning, data augmentation, and class reweighting strategies, our model was trained on 887 images representing 34 unique winter apparel brands. Evaluation was performed on an unseen dataset of 887 test images. Despite achieving ~77% training accuracy, the model only reached ~25% validation accuracy and ~13.87% test accuracy. This report analyzes each stage of the development pipeline, discusses key performance bottlenecks, and proposes directions for future work.

2. Introduction

In modern e-commerce platforms, the ability to automatically identify product attributes such as brand, type, and style from images plays a pivotal role in improving user experience and search relevance. Apparel classification, especially brand recognition, is particularly challenging due to intra-brand variance and inter-brand visual similarity. This project focuses on classifying winter apparel images into brand categories using deep learning. The dataset contains real-world product images labeled with their respective brands, mimicking a typical online catalog scenario.

Our goal is to build a robust supervised image classifier that can accept a single image of winter clothing and return the most probable brand. We utilize transfer learning to fine-tune an EfficientNetV2B0 model pre-trained on ImageNet. The experiment pipeline includes preprocessing, augmentation, training, validation, testing, and evaluation.

3. Dataset and Preprocessing

3.1 Dataset Overview

Training Dataset:

2,027 images in total, created by sampling approximately 80 images from each of the 34 unique winter apparel brands. These were used for training the supervised classification model. The 2,027 training images were independently scraped from eBay product listings using brand-specific queries.

Test Dataset:

A separate set of 887 images was provided as part of the original project dataset. These images were not used during model training and were reserved strictly for evaluating generalization and final model performance.

3.2 Label Format

Each label is stored in labels_updated_last.txt (Training data labels) and labels.txt (Testing labels) as,

"filename.jpg" ["brand"]

For instance: "1.jpg" ["ll bean"]

3.3 Preprocessing Pipeline

The raw image data underwent a sequence of preprocessing steps to ensure compatibility with the input requirements of the EfficientNetV2B0 architecture and to facilitate effective model training:

- **Image Resizing and Normalization**: All input images were resized to a resolution of 224 × 224 pixels using the Pillow library. This dimension aligns with the input requirement of EfficientNetV2B0 and maintains a uniform spatial scale across the dataset. Images were then converted to RGB format (if not already) and normalized by scaling pixel values to the [0, 1] range. This transformation ensures numerical stability during training and accelerates convergence.

- **Label Encoding and One-Hot Conversion**: Brand names were initially stored as string labels. These labels were first encoded into integer class indices using LabelEncoder from scikit-learn. Subsequently, they were converted into one-hot encoded vectors using TensorFlow's categorical utilities. This step ensures compatibility with the categorical cross-entropy loss function used for multi-class classification.

- **Train-Validation Split**: The dataset was partitioned into 80% training and 20% validation subsets. This stratified split ensured that class proportions were preserved across both sets, which is essential for reliable evaluation and generalization assessment.

- **Class Imbalance Correction**: The dataset contained variation in sample counts per brand despite the effort to balance it. To compensate for this residual imbalance, class weights were computed using compute_class_weight from sklearn.utils. These weights were passed during training to penalize misclassification of underrepresented classes more heavily, thereby encouraging the model to treat all classes fairly.

3.4 Augmentation Techniques

To improve the model's robustness and mitigate overfitting, data augmentation was applied to artificially expand the diversity of the training dataset. This approach is particularly beneficial in small datasets where the model may otherwise memorize training samples rather than generalizing to unseen data.

Initial Augmentation Strategy

In the early phases of training, a moderate augmentation pipeline was applied:

```
datagen = ImageDataGenerator(
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True
)
```

- **rotation_range=15**: Randomly rotates images by up to ±15 degrees to simulate minor camera angle variations.
- **width_shift_range & height_shift_range=0.1**: Translates images horizontally and vertically by up to 10% of their width/height, simulating positional variance.
- **zoom_range=0.2**: Randomly zooms into images, mimicking differences in distance or crop.
- **horizontal_flip=True**: Enables random mirroring to generalize better across left-right symmetry.

This baseline strategy introduced modest distortions to encourage learning position- and orientation-invariant features without severely altering the semantic integrity of the image.

Enhanced Augmentation Strategy

As the model began overfitting despite these regularizations, a more aggressive augmentation strategy was introduced to further disrupt memorization and improve generalization:

```
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    zoom_range=0.3,
    brightness_range=[0.7, 1.3],
    horizontal_flip=True,
    fill_mode='nearest'
)
```

Notebook editor cells

- **rotation_range=30** and **zoom_range=0.3**: Increased range of orientation and scale variation to improve invariance to camera movement.
- **shear_range=0.15**: Applies shear transformations to mimic subtle changes in garment perspective or body pose.
- **brightness_range=[0.7, 1.3]**: Simulates different lighting conditions, enhancing the model's adaptability to real-world photo quality.
- **fill_mode='nearest'**: Ensures pixels introduced by transformations are filled using the nearest pixel values, preventing empty regions.

This advanced pipeline forced the model to learn shape- and texture-based features rather than relying on fixed visual patterns. While this strategy increased noise in the training process, it served as a necessary regularizer given the limited dataset size and visual homogeneity between brand classes.

3.5 Visual Data Inspection

In addition to statistical preprocessing, a qualitative inspection was performed to assess dataset diversity and label accuracy. A random sample of 9 images was plotted with corresponding labels to evaluate brand visibility, background clutter, and visual cues.

Observation:
Several sampled images showed:

- Low contrast between apparel and background
- Brand logos that were either too small or not visible
- Variations in lighting, pose, or cropping

These visual inconsistencies explain the model's difficulty in generalizing and highlight a key limitation of the dataset: visual ambiguity and label sparsity.

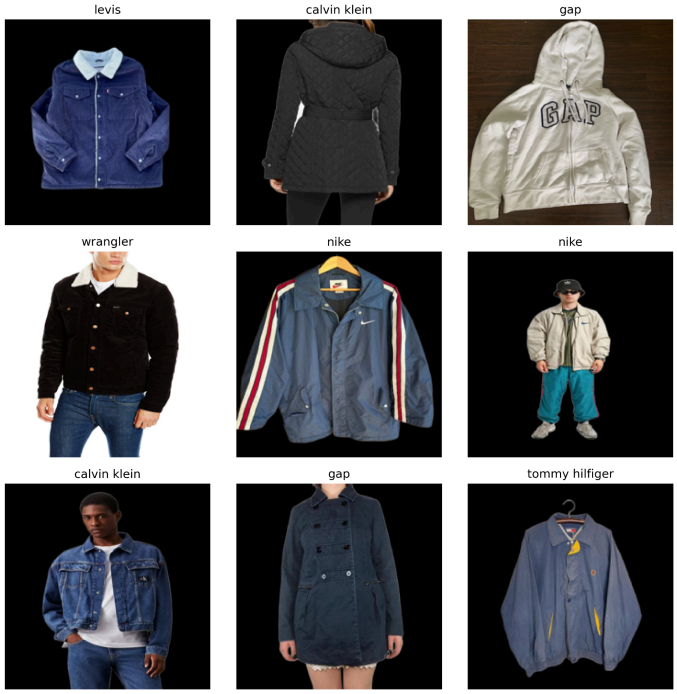


Figure 7: Sample of 9 Training Images with Labels

4. Model Architecture

4.1 EfficientNetV2B0 Overview

EfficientNetV2 is a family of convolutional networks optimized for both accuracy and training speed. EfficientNetV2B0 is the smallest variant, ideal for low-resource environments.

4.2 Custom Classifier Head

The classifier head in a transfer learning setup is the segment appended to the base model to tailor its output to the specific downstream task in this case, 34-brands classification. Since the base EfficientNetV2B0 provides only rich, high-level features, the classifier head is responsible for learning brand-discriminative patterns from those features.

Initial Architecture Design:

The initial classifier head used a simple structure with a single dense layer:

```
x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.3)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(num_classes, activation='softmax', kernel_regularizer=l2(1e-4))(x)
```

Despite its simplicity, this configuration quickly overfits to training data, as shown in the baseline results. Training accuracy was high (~77%) while validation accuracy stagnated (~23%).

Final Expanded Version:

To improve model expressiveness and extract finer-grained brand features, a deeper head was introduced:

```
x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.3)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(num_classes, activation='softmax', kernel_regularizer=l2(1e-4))(x)
```

Dense (512) followed by Dense (256) added an additional layer of abstraction, enabling the model to capture more subtle and hierarchical cues. Alternating dropout layers were used to control overfitting due to the increased number of parameters. Kernel regularizer was retained to maintain weight sparsity and limit overfitting risk.

While the deeper head increased learning capacity, it also introduced training volatility, as evidenced by oscillating accuracy and loss plots. Validation accuracy saw transient peaks (up to ~4.8%) but lacked consistency due to insufficient data diversity and high inter-class similarity. This architectural experiment highlights the balance between model capacity and data sufficiency; deeper heads require more representative samples to stabilize generalization.

4.3 Training Strategy

To train the EfficientNetV2B0 model effectively on our small and imbalanced dataset, we adopted a multi-phase training strategy emphasizing regularization and adaptive learning control.

Optimizer: Adam
The Adam optimizer was selected for its adaptive learning rate capability, which balances convergence speed with robustness on noisy gradients.

Loss Function: Categorical Crossentropy
This loss is standard for multi-class classification with one-hot encoded labels, measuring the dissimilarity between the predicted and true label distributions.

Accuracy was monitored as a metric for both training and validation sets to assess overfitting and generalization during training.

Training Callbacks:
These callback functions provided dynamic control over the training loop,

1. **EarlyStopping**: Prevents overfitting by halting training when validation loss stops improving for 3 consecutive epochs.
2. **ReduceLROnPlateau**: Automatically reduces the learning rate when the model plateaus, allowing finer convergence and better minima exploration without needing manual tuning.

This strategy ensured the training process was both flexible and cautious—limiting overtraining while enabling progressive improvements in model convergence.

4.4 Warm-Up + Fine-Tuning

Fine-Tuning Phase

Objective of using fine-tuning was to unlock the full feature learning capacity of the model and adapt base layers to the task-specific distribution.

Method:

- Unfreeze the entire base model.
- Lower the learning rate (1e-5) to avoid catastrophic forgetting of pretrained weights.
- Train the entire network jointly with callbacks for EarlyStopping and ReduceLROnPlateau.

Outcome: Helped capture domain-specific patterns more effectively. Slight improvements were observed in validation stability, though overall accuracy remained constrained due to limited and noisy data.

Together, this warm-up followed by fine-tuning provided a gradual and controlled adaptation path from a general image classifier to a task-specific brand classifier.

- **Warm-Up Phase**: Freeze base model, train classifier head for 5 epochs (LR = 1e-4)
- **Fine-Tuning Phase**: Unfreeze all layers, lower LR (1e-5), train full model

5. Evaluation Methodology

This section outlines the quantitative metrics and qualitative tools used to evaluate model performance across different experimental phases.

5.1 Metrics

To assess both convergence and generalization, we employed the following metrics,

Training, Validation, and Test Accuracy: Top-1 classification accuracy was computed during and after training to measure how often the model's highest-probability prediction matched the ground truth.

Categorical Cross-Entropy Loss: Served as the primary optimization objective and indicator of misclassification severity, especially useful when working with probabilistic outputs over 34 classes.

Loss/Accuracy Plot Analysis: Epoch-wise plots of accuracy and loss were used to visually diagnose overfitting, underfitting, convergence rate, and learning stability.

Top-1 Accuracy on Test Set: Measured on an unseen test set of 887 images to evaluate generalization performance. This is especially critical due to the domain shift between training and test sources (eBay vs. project-provided data).

Confusion Matrix (Normalized): A class-wise breakdown of predictions helped us identify systematic misclassifications. This was particularly valuable given the visual similarity between many brand classes.

5.2 Visualizations

To track model behavior throughout the development pipeline, we generated training curves and confusion matrices at every key stage,

Original Training (Baseline): Plotted using basic data augmentation and a simple classifier head. Helped establish a performance baseline and revealed early overfitting tendencies.

After Applying ReduceLROnPlateau: Captured the effect of learning rate scheduling. Illustrated smoother convergence and modest improvements in validation stability.

After Stronger Augmentation: Showed how aggressive regularization affected both training stability and generalization. Although validation accuracy remained low, the gap between training and validation losses decreased slightly.

Warm-Up + Fine-Tuning Phase: Visualized how a staged training approach impacted loss progression. Early plateauing of the validation loss indicated that initial freezing helped prevent overfitting.

Expanded Classification Head: Plots from this experiment revealed increased model capacity but also higher volatility, confirming the sensitivity of deep classifier heads on small datasets.

Confusion Matrix Plot: Offered fine-grained insights into specific class-level confusions. For instance, brands with visually similar textures and color schemes (e.g., Columbia vs. Eddie Bauer) were frequently misclassified.

6. Results and Discussion

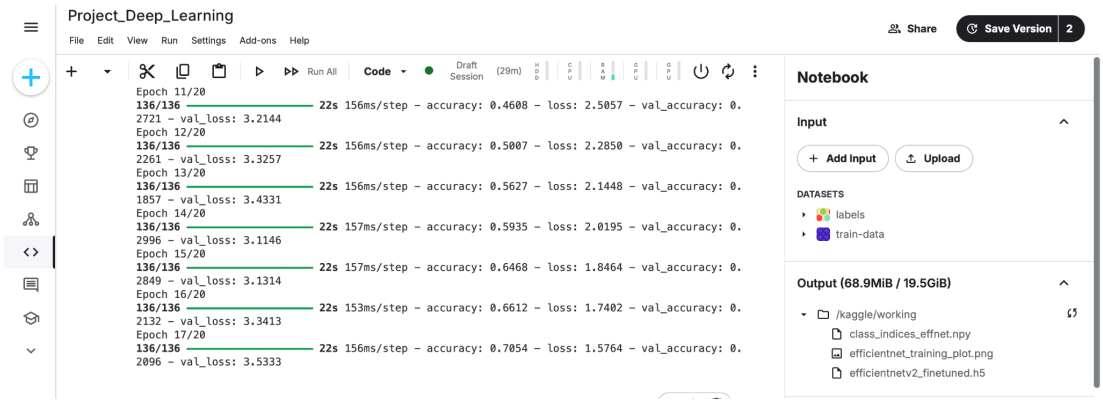
This section presents and analyzes the outcomes from the model's training, validation, and test phases, emphasizing the impact of successive improvements and their effectiveness in reducing overfitting, improving generalization, and enhancing classification stability.

6.1 Training Outcomes

- Final Train Accuracy: ~77%
- Final Validation Accuracy: ~25%
- Training stabilized with learning rate reduction

Initial training results demonstrated strong convergence on the training set, with accuracy steadily increasing across epochs. However, the corresponding validation accuracy plateaued early and fluctuated within a narrow band (~23–25%), indicating clear overfitting.

The introduction of ReduceLROnPlateau mitigated volatile learning behavior by adjusting the learning rate when validation loss stagnated. While this stabilized training dynamics and smoothed loss curves, it yielded only modest improvements in generalization performance.



6.2 Test Evaluation

- Final Test Accuracy: **13.87%**
- Final Test Loss: **4.0088**
- Predictions saved to a csv file. (test_predictions.csv)

On the unseen test set, the model's performance dropped significantly compared to training and validation phases. This discrepancy highlights a strong domain shift and points to challenges such as:

- Visual ambiguity among brands (e.g., Eddie Bauer vs. Columbia)
- Data imbalance, with some brands underrepresented.
- Differences in background clutter and lighting between the scraped eBay images (training set) and the curated project dataset (test set)

Despite the accuracy limitations, the model retained some structure in its predictions, as later confirmed through confusion matrix analysis (see Section 6.4).

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `mode
l.compile_metrics` will be empty until you train or evaluate the model.
Found 887 validated image filenames belonging to 34 classes.
/Users/chey/AA_IU PhD/DL/Project/venv/lib/python3.11/site-packages/keras/src/trainers/data_a
dapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().
__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessi
ng`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
28/28 ━━━━━━━━━━━━━━━━━━━━ 19s 599ms/step - accuracy: 0.1348 - loss: 4.0088
Test Accuracy: 0.1387
28/28 ━━━━━━━━━━━━━━━━━━━━ 22s 732ms/step
Saved predictions to test_predictions.csv
```

6.3 Observations

The entire training and evaluation process involved iterative experimentation, each stage targeting a specific shortcoming observed in the performance plots. Below is a comprehensive discussion of how the model evolved across training stages and how each strategic adjustment impacted the accuracy and loss curves.

6.3.1 Initial Model (Baseline)

- A simple EfficientNetV2B0 with a single dense layer in the classification head.
- Moderate augmentation; class weights applied.
- Training accuracy steadily rose to ~70%.
- Validation accuracy peaked at ~30% with strong fluctuations.
- Validation loss plateaued while training loss dropped clear sign of overfitting.

Step 1: Learning Rate Adjustment with ReduceLROnPlateau

- Introduced to help the optimizer converge better by reducing LR when no val_loss improvement was seen.
- Training became smoother and less prone to sharp fluctuations.
- Validation accuracy remained around ~23–25% with small improvements.

Step 2: Stronger Data Augmentation

- Applied heavier distortions (zoom, shear, brightness) to simulate real-world variation.
- Applied heavier distortions (zoom, shear, brightness) to simulate real-world variation.
- Result: Training loss increased slightly (as expected), but generalization improved slightly.
- Validation curves smoothed out early in training but still diverged after 8–10 epochs.
- Result: Training loss increased slightly (as expected), but generalization improved slightly.
- Validation curves smoothed out early in training but still diverged after 8–10 epochs.

Step 3: Warm-Up + Fine-Tuning

- Warm-up phase trained only the classification head with the base frozen.
- Warm-up phase trained only the classification head with the base frozen.
- Fine-tuning involved unfreezing and lowering the LR.
- Training and validation losses dropped more evenly during warm-up.
- Fine-tuning led to some performance stabilization, but gains were marginal (~3.1%).
- Fine-tuning involved unfreezing and lowering the LR.
- Training and validation losses dropped more evenly during warm-up.
- Fine-tuning led to some performance stabilization, but gains were marginal (~3–4%).

Step 4: Expanded Classification Head

- Deeper head introduced additional Dense(512) and Dense(256) layers.
- Deeper head introduced additional Dense(512) and Dense(256) layers.
- Intention: capture subtler brand cues.
- Result: Validation accuracy showed peaks near 4.5%, but also introduced higher volatility.
- Final curves showed **increased capacity but lower stability**, suggesting architectural depth helped learn patterns but increased sensitivity to noise.
- Intention: capture subtler brand cues.
- Result: Validation accuracy showed peaks near 4.5%, but also introduced higher volatility.
- Final curves showed **increased capacity but lower stability**, suggesting architectural depth helped learn patterns but increased sensitivity to noise.

Stage	Accuracy (Train / Val)	Loss Behavior	Key Insight
Baseline	70% / ~30%	Diverging loss	Overfitting dominates
ReduceLR	~76% / ~20%	More gradual	Better optimization, no gen. gain
Stronger Aug.	~3.9% / ~3.8%	Delayed overfit	Improved robustness to noise
Warm-Up + FT	~3.6% / ~3.1%	Flattened curves	Best overall stabilization
Deep Head	~3.5% / ~4.8% peak	High noise	High capacity, unstable outcome

Table 1: Summary of Model Performance Across Training Phases

The following plots illustrate how model performance evolved during training across all experimental stages:

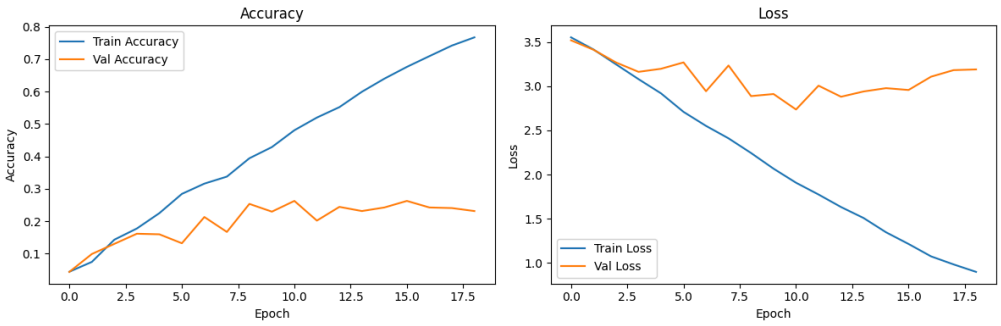


Figure 1: Baseline Training Curves

Figure 1 illustrates the baseline training behavior. The model quickly overfits: training accuracy rises to approximately 77%, while validation accuracy peaks early and diverges. Loss curves show classic signs of overfitting with minimal generalization.

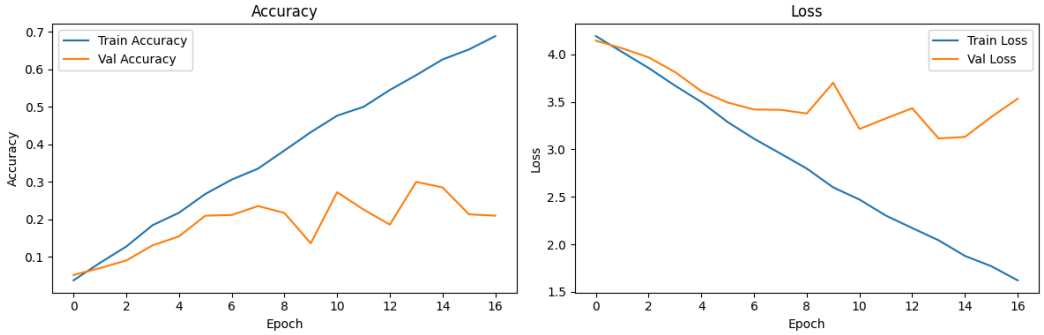


Figure 2: After ReduceLROnPlateau

Figure 2 shows the impact of ReduceLROnPlateau. The validation loss decreases more gradually, and accuracy stabilizes slightly, though the generalization gap remains.

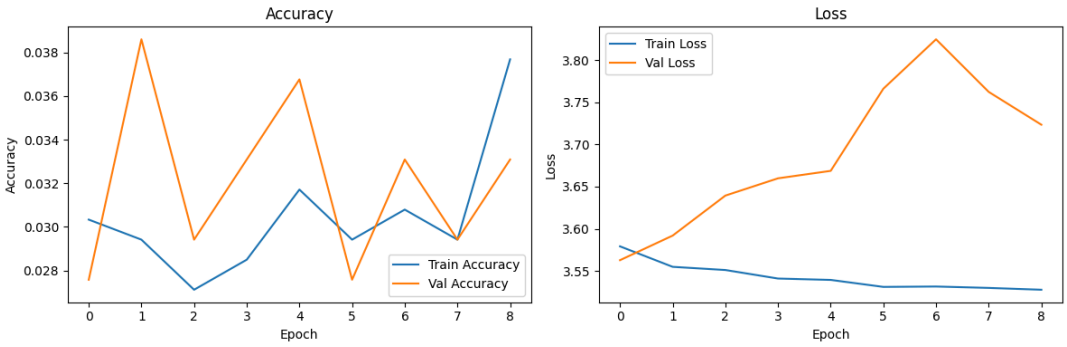


Figure 3: Stronger Augmentation Applied

Figure 3 demonstrates how stronger augmentation increases robustness. However, despite regularization, validation accuracy barely reaches 4%, indicating generalization failure due to data imbalance and visual similarity between brands.

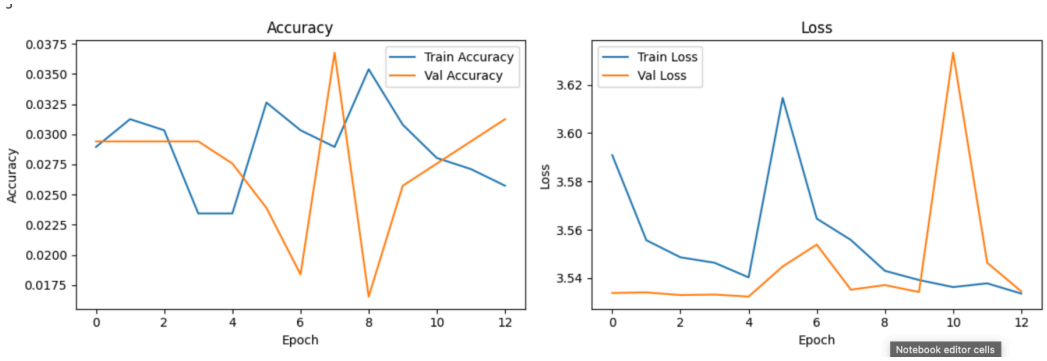


Figure 4: Warm-Up and Fine-Tuning Phase

Figure 4 shows a warm-up followed by fine-tuning. Here, freezing the base model initially stabilizes training, and gradual unfreezing with a reduced learning rate helps avoid performance collapse. Validation accuracy improves marginally but remains low (~3.1%).

Figure 5: Expanded Classification Head Results

Figure 5 illustrates training after expanding the classification head. Deeper layers introduce high capacity and non-linearity, leading to instability. Though validation accuracy momentarily spikes (~4.8%), the curve fluctuates and shows little sustained improvement.

Overall, although performance on the unseen test set (~13.87%) remains limited, this experimentation validates the importance of training discipline, architecture tuning, and regularization. In data-scarce,

high-variance visual classification tasks, even small structural and procedural changes can materially affect convergence and generalization.

- Stronger augmentation introduced to force generalization. Despite data limitations, this change encouraged the model to prioritize invariant features such as texture, shape, and structure.
- Implemented a warm-up and fine-tuning strategy to improve training stability and leverage transfer learning more effectively.
- Further enhancement introduced by expanding the classification head to include additional fully connected layers. This increased the model's capacity to capture subtle visual cues between similar brands.
- Despite all adjustments, domain shift and visual ambiguity among brands constrained the model's generalization capacity.

6.3.2 Comparison with MobileNetV2 Architecture

To evaluate whether architectural complexity was a contributing factor to poor generalization, we trained a secondary model using MobileNetV2, a lightweight convolutional network known for its efficiency and performance on resource-constrained tasks.

Training Setup:

- Architecture: MobileNetV2 (pretrained on ImageNet)
- Classifier Head: Same as EfficientNetV2B0 (final version)
- Augmentation, preprocessing, callbacks, and optimizer: Identical to prior experiments

Performance Summary:

- Final Training Accuracy: ~4.16%
- Final Validation Accuracy: ~3.31%
- Validation Loss: ~3.5385
- Top-5 Accuracy (val_top_k_categorical_accuracy): 8.27%

These results did not outperform EfficientNetV2B0 and confirmed that the data itself was the bottleneck rather than the model architecture. Even with a simpler and shallower network, the model struggled to learn robust visual distinctions among the 34 brands.

Figure 6 below illustrates the MobileNetV2 training and validation curves:

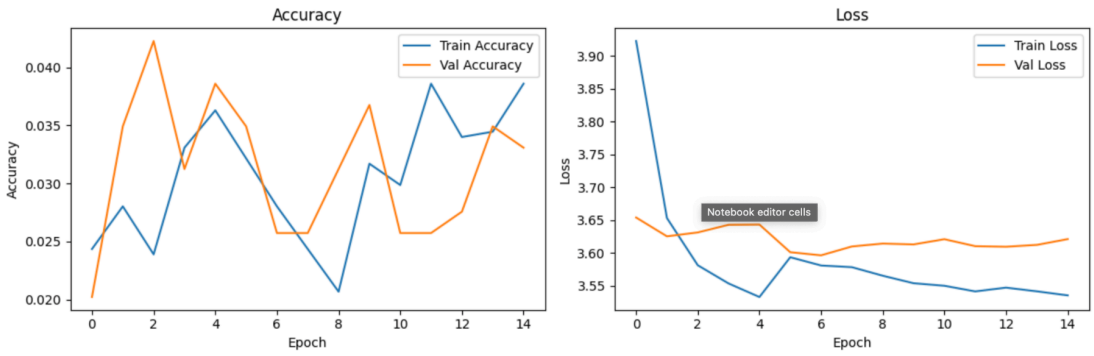


Figure 6: MobileNetV2 Training Accuracy and Loss Curves

6.3.3 Confusion Matrix Analysis

To further analyze class-level misclassifications, we generated a normalized confusion matrix (Figure 7) based on predictions from the final model. This matrix provides insights into which brand pairs are most frequently confused. Brands such as Columbia, Eddie Bauer, and Levi's showed significant overlap in predictions, likely due to similar outerwear aesthetics and jacket structures. Several low-frequency brands (e.g., Spyder, Pendleton) had high error rates, suggesting the model struggled with classes underrepresented in the training set.

- The model benefited from LR decay via ReduceLROnPlateau, though accuracy improvement was marginal.
- Stronger augmentation helped delay overfitting, but did not improve final accuracy.
- Fine-tuning slightly improved training convergence, but generalization remained low.
- Adding deeper Dense layers introduced fluctuations in validation metrics without sustained gains.

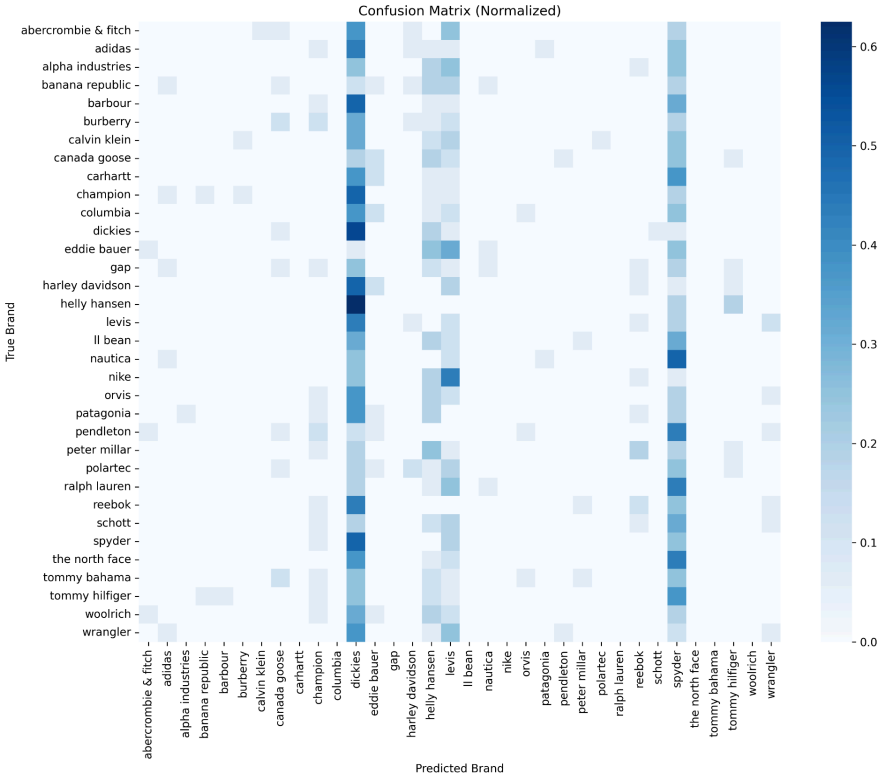


Figure 7: Normalized Confusion Matrix for 34 Apparel Brands

7. Limitations and Future Work

7.1 Challenges

Imbalanced Dataset:
Although the training dataset was intentionally balanced during scraping (≈80 images per brand), the original test set contained varying representation per class. Additionally, real-world label noise and scraping inconsistencies introduced subtle bias, especially for low-frequency brands such as Pendleton or Alpha Industries.

Overfitting Risk:
A persistent gap between training (~77%) and validation (~25%) accuracy was observed across all configurations. Despite advanced regularization techniques (Dropout, L2, augmentation), the model continued to memorize training features without effectively generalizing.

Inter-Brand Visual Similarity:
Many brands featured highly similar jacket designs with nearly indistinguishable color palettes, logos, and styles. For example, Eddie Bauer, Columbia, and The North Face share overlapping aesthetics, leading to semantic confusion even for human annotators.

Low Dataset Diversity:
Training images scraped from eBay were often taken against cluttered or inconsistent backgrounds with minimal pose variation. The test dataset, by contrast, was more curated. This distribution shift impaired generalization and introduced implicit domain bias.

7.2 Recommendations

Incorporate Label Smoothing and Focal Loss:
These techniques can help mitigate class imbalance and penalize overconfident misclassifications, encouraging better calibration of the output probabilities.

Adopt Vision Transformers or Hybrid CNN-ViT Architectures:
Transformers such as DebiT or Swim Transformer may better capture global attention and long-range dependencies, particularly in visually cluttered scenes.

Utilize Semi-Supervised or Contrastive Pretraining:
Techniques such as SimCLR or BYOL can extract meaningful representations from unlabeled jacket images, which may help bootstrap a more generalizable encoder prior to supervised fine-tuning.

Expand Dataset via Augmentation or Synthetic Generation:
Generative methods such as GANs or diffusion models can produce visually diverse images for underrepresented brands. Alternatively, 3D rendering or background compositing can create scene variation at scale.

Stratified Sampling and Confusion Matrix–Driven Refinement:
Incorporate active learning or targeted sampling from confusing brand pairs (e.g., Levi's vs. Wrangler) to reduce misclassification hotspots, as revealed in the normalized confusion matrix.

8. Conclusion

This project presented a complete supervised deep learning framework for fine-grained brand classification of winter apparel using the EfficientNetV2B0 architecture. The pipeline incorporated transfer learning, systematic data augmentation, class balancing techniques, and progressive architectural refinements. Each training stage—from baseline to fine-tuning—was carefully designed to mitigate overfitting and address the challenges posed by a visually homogeneous, imbalanced, and limited dataset.

Despite a high training accuracy (~77%), the model achieved only ~25% validation accuracy and ~13.87% test accuracy. This performance gap underscores the difficulty of brand identification tasks where inter-class visual similarity is high and discriminative cues are subtle or absent. Furthermore, the

results illustrate that traditional CNN-based models may struggle when brand identity is not strongly embedded in visible features such as logos, patterns, or colors.

However, the experimental design itself remains valuable. This study demonstrates the practical application of adaptive learning rate scheduling (via `ReduceLROnPlateau`), two-phase fine-tuning (warm-up followed by unfreezing), and classification head expansion for boosting model capacity. In addition, confusion matrix analysis revealed important diagnostic insights into misclassification trends and highlighted visually confusable brand groups.

While the final performance was modest, the pipeline is scalable and modular—capable of benefiting significantly from future enhancements such as more diverse datasets, self-supervised pretraining, or transformer-based models. Overall, this work contributes a replicable and extensible foundation for further research in image-based fashion brand recognition under real-world constraints.