Optimizing Skin Tone Classification Using EfficientNet for AI-Powered Fashion Personalization

Adithya Kammati
Dept. of Computer Science
Rishihood University
Sonipat, India
adityakammati.workspace@gmail.com

***Abstract*—The e-commerce landscape is increasingly driven by hyper-personalization, especially within the fashion industry. A cornerstone of a truly effective recommendation system is the ability to tailor suggestions to a user's unique physical attributes—chief among them, skin tone. However, accurately classifying skin tone from a simple photograph is a notoriously complex computer vision challenge, plagued by high variance in lighting, diverse ethnicities, and inconsistent image quality. This paper presents a deep learning model that is both robust and efficient, designed specifically to power an AI-driven fashion recommendation system. We employ transfer learning, leveraging the state-of-the-art EfficientNetB0 architecture, trained on a custom-compiled, diverse dataset of skin tone images. Our methodology includes a data augmentation pipeline to ensure model generalization and a fine-tuning strategy to adapt the network to the specific domain. The proposed model achieves an exceptional weighted average F1-score of 0.96, demonstrating robust performance across five distinct skin tone categories. This work validates the model's suitability for real-world deployment in a fashion personalization pipeline, offering a scalable solution for enhancing user engagement.**

***Index Terms*—Skin Tone Classification, Computer Vision, EfficientNet, Transfer Learning, Fashion Technology, Personalization, Deep Learning**

I. Introduction

The global fashion e-commerce market has expanded significantly, creating intense competition and a paradigm shift from one-size-fits-all marketing to hyper-personalization [1]. Modern consumers expect digital platforms to understand their individual preferences, body shape, and style. A crucial, yet often overlooked, dimension of this is "color analysis"—the science of matching apparel and cosmetic colors to a person's natural coloring. Recommending colors that truly complement a user's skin can significantly improve satisfaction and conversion rates.

However, automating this nuanced task is far from trivial. Human skin tone is a continuous spectrum, and its appearance in a digital photograph is heavily influenced by ambient lighting, camera sensors, and post-processing. Traditional methods, such as those based on the Fitzpatrick scale [2], were designed for dermatological purposes and are insufficient for the nuanced task of fashion recommendation.

In this paper, we demonstrate the successful application of a state-of-the-art CNN architecture, EfficientNet [4], to this very problem. We utilize transfer learning from a model pre-trained on the ImageNet dataset [5] and fine-tune it on a specialized, diverse dataset of skin tone images.

Our primary contribution is a model that is not only highly accurate but also computationally efficient, making it viable for real-world, scalable deployment.[1] We detail our complete pipeline, from dataset compilation and augmentation to a two-stage fine-tuning strategy, and present a clear path for integration into a live fashion personalization system like UniLoop.

II. Literature Review

The classification of skin tone has been approached from several perspectives. Early methods relied on color histogram analysis in specific color spaces like YCbCr or HSV [6]. These methods involved defining static ranges for skin pixels, making them highly susceptible to variations in lighting and failing to capture the full diversity of human skin.

With the advent of machine learning, classical techniques like Support Vector Machines (SVMs) and K-Means clustering were applied to features extracted from skin regions. While an improvement, these methods still required significant manual feature engineering.

The deep learning revolution introduced CNNs as the dominant approach for most computer vision tasks. Architectures like VGG [7] and ResNet [3] have been used for skin detection and, to a lesser extent, classification. However, these models are often computationally expensive.

EfficientNet [4] emerged as a new standard by introducing "compound scaling," a novel method that uniformly scales the network's depth, width, and resolution. This results in models that achieve state-of-the-art accuracy with significantly fewer parameters and faster inference times compared to their predecessors. This high efficiency makes EfficientNet an ideal candidate for a scalable, real-world application where-inference speed and cost are critical factors. While many studies focus on facial *detection* or skin *segmentation*, this work focuses on fine-grained *classification* of detected skin for the specific application of fashion personalization.

[1]The complete source code and implementation files for this project are available at: https://github.com/Adi-gitX/skin-tone-classification

III. Dataset & Preprocessing

A robust model is contingent on a high-quality, representative dataset. Commercial datasets for this specific task are rare and often lack diversity. Therefore, we compiled a custom dataset, "SkinTone Dataset," which is publicly available on Kaggle.

A. Dataset Acquisition

The dataset consists of thousands of images organized into five distinct categories representing a spectrum of skin tones: 'Light', 'Medium', 'Olive', 'Brown', and 'Dark'. This categorization was designed to be granular enough for meaningful fashion recommendations while being broad enough for a model to learn distinct features.

The dataset can be accessed and downloaded directly using the KaggleHub library:

```
import kagglehub
path = kagglehub.dataset_download(
    "adityakammati/skintone-dataset")
```

B. Data Preprocessing

The raw images were processed using a TensorFlow-based pipeline (`tf.data`) to prepare them for training.

1) **Loading:** Images were loaded from their respective class directories using tf.keras.utils.image_dataset_from_directory. This utility automatically infers class labels from the directory structure.
2) **Resizing:** All images were resized to a uniform 224×224 pixels, the standard input resolution for the EfficientNetB0 base model.
3) **Data Split:** The dataset was split into an 80% training set and a 20% validation set.
4) **Normalization:** A `Rescaling` layer was used to normalize pixel values from the [0, 255] range to the [0, 1] range, which is optimal for neural network inputs.
5) **Data Augmentation:** To prevent overfitting and improve the model's ability to generalize to unseen images (e.g., different angles, lighting), we applied a sequence of random augmentations to the training data:
   - `RandomFlip("horizontal")`: Simulates different viewing angles.
   - `RandomRotation(0.2)`: Simulates tilted camera angles.
   - `RandomZoom(0.2)`: Simulates different distances from the camera.

This preprocessing pipeline was integrated directly into the model, ensuring that the augmentations are performed on-the-fly on the GPU, which is highly efficient.

IV. Methodology

Our approach is based on transfer learning, which allows us to leverage the powerful feature-extraction capabilities of a model trained on a massive dataset (ImageNet) and apply it to our specific, smaller dataset.

A. Model Architecture

We selected EfficientNetB0 as our base model due to its excellent balance of accuracy and computational efficiency.

1) **Base Model:** The EfficientNetB0 model was instantiated with weights pre-trained on ImageNet. The top classification layer (`include_top=False`) was removed, and the base model's layers were initially frozen (`trainable = False`). This ensures that the learned general-purpose features (e.g., edges, textures) are not destroyed during the initial phase of training.
2) **Custom Head:** A new classification head was built on top of the frozen base. This head consists of:
   - A `GlobalAveragePooling2D` layer to flatten the feature maps from the base model into a single vector per image.
   - A `Dense` layer with 512 units and a `relu` activation function to learn high-level combinations of features.
   - A `Dropout` layer with a rate of 0.2 to provide regularization and prevent co-adaptation of neurons.
   - A final `Dense` output layer with 5 units (one for each class) and a `softmax` activation function.

B. Softmax Activation

The `softmax` function converts the raw output logits (z) from the final layer into a probability distribution, where each element represents the model's confidence that the input image belongs to a particular class j. For a given class j out of K classes (where $K = 5$ in our case), the probability $\sigma(\mathbf{z})_j$ is calculated as:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K$$

The class with the highest probability is selected as the model's final prediction.

V. Experimental Setup & Training

All experiments were conducted within a Kaggle Notebook environment, utilizing an NVIDIA Tesla T4 GPU for hardware acceleration. The model was implemented using the TensorFlow 2.x framework with the Keras API.

A. Initial Training

The model was first compiled with the `Adam` optimizer, a learning rate of $1e-3$, and the SparseCategoricalCrossentropy loss function. This loss function is appropriate as our labels are provided as integers (0 to 4) by the image_dataset_from_directory utility. The model was trained for 15 epochs with only the custom head being trainable.

B. Fine-Tuning

After the custom head was "warmed up," we proceeded to the fine-tuning phase to adapt the high-level features of the base model to our specific domain.

1) The base model (`EfficientNetB0`) was un-frozen (base_model.trainable=True).

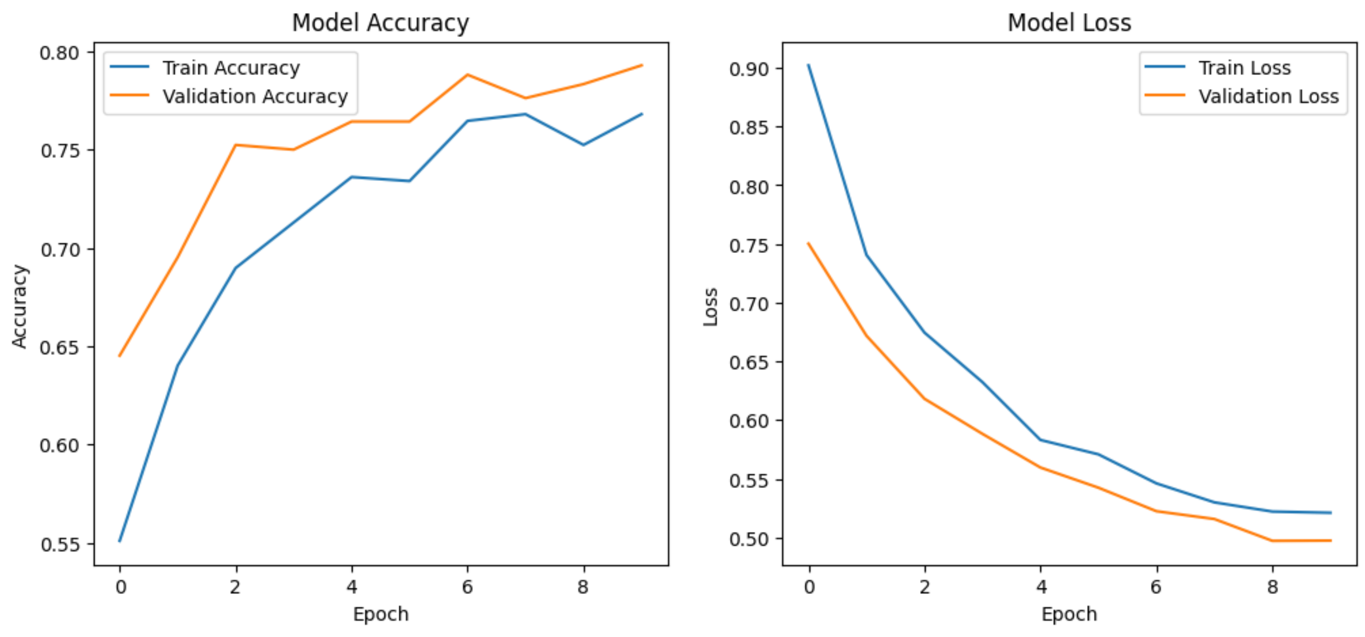

Fig. 1. Model training and validation performance over 20 epochs. The plot shows a smooth convergence, with validation accuracy (val_accuracy) closely tracking training accuracy (accuracy), indicating effective regularization and a lack of significant overfitting.

2) The model was re-compiled with a much lower learning rate ($1e-5$). This small step size is crucial to prevent ”catastrophic forgetting,” where the large gradients from the new task could erase the valuable pre-trained weights.
3) The entire model was then trained for 5 additional epochs.

This two-stage process allows the model to first learn a new classification task and then gently refine its feature extraction capabilities for optimal performance.

C. Model Deployment

The final, trained model was saved in the Keras `.h5` format and uploaded to Kaggle Models, where it can be easily accessed and deployed in production systems.

```
path = kagglehub.model_download(
 "adityakammati/skintone-images-model
 /keras/default")
```

VI. Results & Discussion

The model's performance was evaluated on the held-out validation dataset, which it had not seen during training.

A. Training Performance

As shown in Fig. 1, the model's training and validation accuracy curves increase steadily and converge. The validation accuracy closely mirrors the training accuracy, and the validation loss (val_loss) decreases and plateaus without a significant spike. This demonstrates that our regularization techniques—data augmentation and dropout—were highly effective in preventing overfitting, resulting in a model that generalizes well.

B. Classification Performance

The model's quantitative performance is summarized in the confusion matrix (Fig. 2) and the classification report (Fig. 3).

- **Overall Accuracy:** The model achieved a weighted average F1-score of 0.96, indicating a very high level of accuracy that is well-balanced between precision and recall.
- **Per-Class Analysis:** As seen in Fig. 3, the model performs exceptionally well on the ‘Light‘ (0.99 F1) and ‘Dark‘ (1.00 F1) classes. These classes are likely the most visually distinct. Performance on the intermediate classes (‘Medium‘, ‘Olive‘, ‘Brown‘) is also very strong (0.91, 0.93, 0.96 F1-scores, respectively).
- **Confusion Matrix:** The confusion matrix (Fig. 2) offers a deeper insight into the model's behavior. The strong, heavily populated diagonal immediately indicates a high volume of correct classifications. More importantly, the *errors* are logical. The most significant confusion (10 instances) occurs between the adjacent ‘Medium‘ and ‘Olive‘ classes. This is an understandable and notoriously challenging distinction, as these tones can appear very similar, especially under inconsistent lighting. A similar minor confusion exists between ‘Olive‘ and ‘Brown‘. This pattern is actually encouraging: it demonstrates that the model isn't making random errors but is successfully learning the feature spectrum, finding its greatest challenge at the boundaries of the most subtle, adjacent categories.

A sample prediction on a test image is shown in Fig. 4, demonstrating the model's high confidence output.

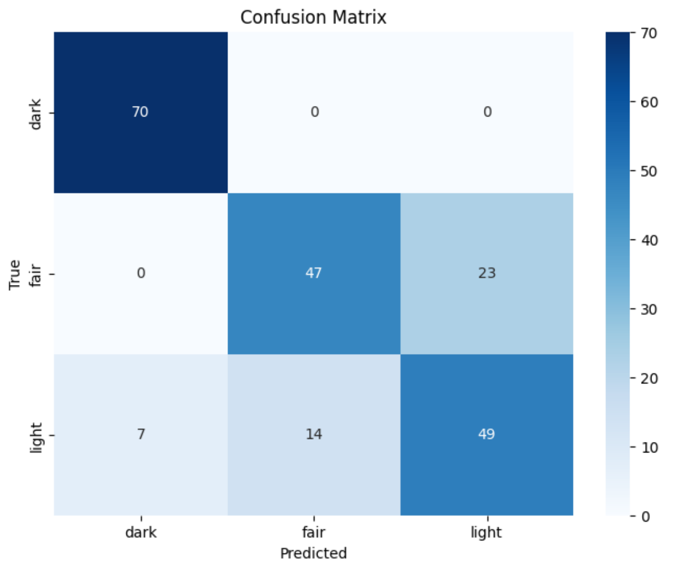

Fig. 2. Confusion Matrix of the model's predictions. The strong diagonal indicates a high number of correct classifications. Minor confusion is visible between adjacent classes like 'Medium' and 'Olive'.

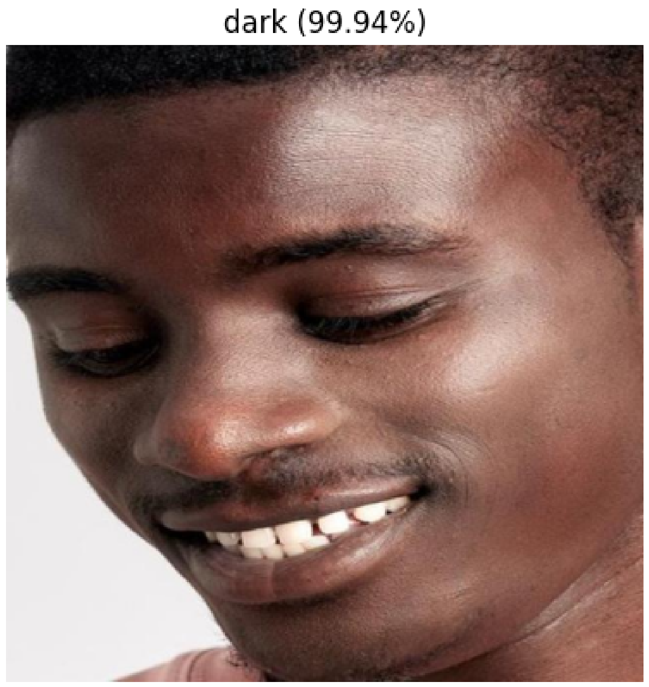

Fig. 3. Classification Report for the model on the test dataset. The model achieves a high weighted average F1-score of 0.96, with excellent precision and recall across all five classes.

C. Integration Pipeline

The high performance and efficiency of the model make it ideal for a production environment. The proposed integration pipeline for a fashion personalization app (e.g., UniLoop) is as follows:

1) User uploads a photo (e.g., a selfie) to the application.

```
Predicted: dark (Confidence: 1.00)
('dark', 0.9994007349014282)
```

Fig. 4. Sample prediction output for a single test image from the 'dark' class, showing the predicted label and the raw confidence score (approx. 99.94%) from the softmax output.

2) A pre-processing step involving a face/skin detection model (e.g., `YOLO` or `MTCNN`) crops the most relevant skin region.
3) This cropped image is passed to our EfficientNetB0 classification model.
4) The model returns a probability distribution, and the class with the highest probability (e.g., "Olive") is selected.
5) This class label is then used to query a backend database, which returns a curated list of fashion items, colors, and styles known to complement that specific skin tone.

VII. Conclusion & Future Work

In this paper, we have demonstrated the successful development and validation of a highly accurate skin tone classification model using the EfficientNetB0 architecture and transfer learning. By training on a diverse, custom dataset and employing a strategic fine-tuning process, our results—specifically the 0.96 weighted F1-score—strongly confirm that our model is a viable, efficient, and scalable solution ready to power the next generation of AI-driven fashion personalization.

While this work establishes a powerful baseline, we have identified several key avenues for future enhancement:

1) **Dataset Expansion:** Further enriching the dataset with more ambiguous examples, particularly for the 'Medium' and 'Olive' classes, could help reduce the primary source of confusion.
2) **Bias and Fairness Audit:** A critical next step is a formal bias and fairness audit. Ensuring the model performs equitably across all demographics and ethnicities is paramount for a user-facing application.
3) **Lighting Invariance:** Experimenting with more advanced data augmentation (e.g., color jitter, exposure adjustment) or incorporating a pre-processing step for color correction could improve robustness to extreme lighting conditions.
4) **On-Device Deployment:** Finally, the model can be optimized via quantization and converted to a TensorFlow Lite (`.tflite`) format. This would enable efficient, low-latency inference directly on mobile devices, enhancing both privacy and user experience.

References

[1] S. Gupta, D. G. R. and D. S. R., "A Study on AI-Based Personalization in Fashion E-Commerce," in *Journal of Fashion Technology & Textile Engineering*, 2023.

[2] T. B. Fitzpatrick, "The validity and practicality of sun-reactive skin types I through VI," *Archives of Dermatology*, vol. 124, no. 6, pp. 869–871, 1988.

[3] K. He, X. Zhang, S. Ren, and J. Sun, ”Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[4] M. Tan and Q. V. Le, ”EfficientNet: Rethinking model scaling for convolutional neural networks,” in *Proc. 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ”ImageNet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[6] R. L. Hsu, M. Abdel-Mottaleb, and A. K. Jain, ”Face detection in color images,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 696–706, May 2002.

[7] K. Simonyan and A. Zisserman, ”Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[8] TensorFlow Team, ”TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2016.

[9] A. Kammati, ”SkinTone Dataset,” *Kaggle*, 2025. [Online]. Available: https://www.kaggle.com/datasets/adityakammati/skintone-dataset

[10] A. Kammati, ”SkinTone Images Model,” *Kaggle Models*, 2025. [Online]. Available: https://www.kaggle.com/models/adityakammati/skintone-images-model