HairNet:
A Hair Type CNN Classifier

DSAN 6600: Deep Learning & Neural Networks

Yashwanth Devabathini
Morgan Dreiss
Satomi Ito
Viviana Luccioli

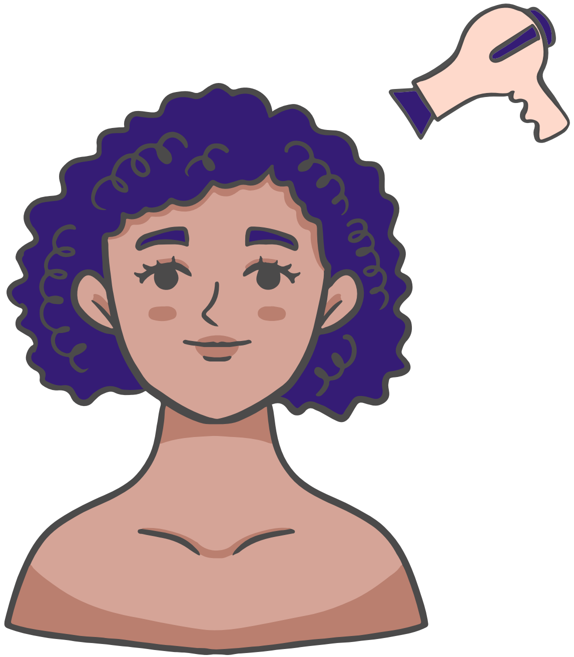


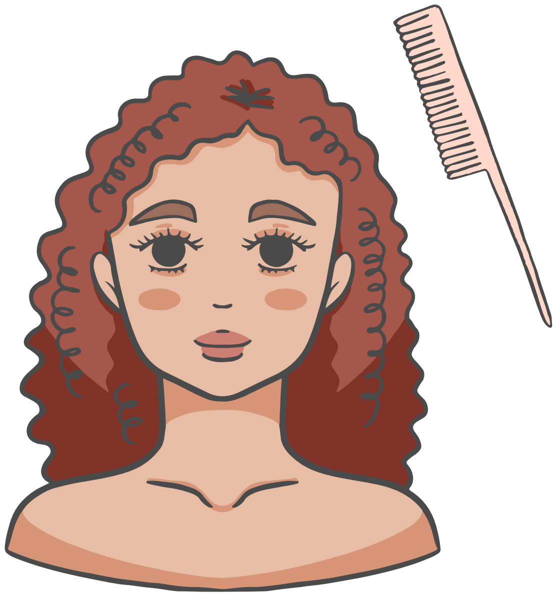
Table of contents

01 Introduction

Hair type determines wash frequency, products, styling tools, and protective practices → knowing your hair type allows you to care for it!

The problem:

- Hair type self classification is difficult
- Current methods (comparison charts, quizzes) can be subjective & inconsistent
- No existing ML models or labeled datasets for this task



Our solution:

- Custom data collection & pipeline
- Application of CNN (EfficientNet) to Andre Walker's 10-class hair typing system


Type of hair
Straight Hair S1
Wavy Hair W2a
Wavy Hair W2b
Wavy Hair W2c
Curly Hair C3a
Curly Hair C3b
Curly Hair C3c
Kinky Hair K4a
Kinky Hair K4b
Kinky Hair K4c

02 Dataset - Source

1. Multi-query Google search (*SerpAPI*)

- 25 unique search queries per hair type to maximize diversity
- ~4,000 target images per category across 10 hair types
- API filters out exact duplicates

2. Manual classification

- Scanned through images to ensure correct assignment
- Re-classified to appropriate hair type if not

02 Dataset - Data Processing

Quality filtering
(*YOLOv8*)

- Only kept images with exactly 1 person

Data augmentation
(*Keras*)

- Rotation, brightness, horizontal flip
- **Minimum 1,200 samples per class**

Hair isolation & image resizing

(*MediaPipe hair segmenter*)

- Background removal
- Bounding box → cropped to hair
- resize to **600×600px**

Example - Data Processing

Quality filtering

Data augmentation

Hair segmentation

03 Methodology

Transfer learning with EfficientNet:

- Pretrained ImageNet weights
- Works with high resolution inputs
- Insufficient dataset size for training from scratch

	EfficientNet V1 - B7	EfficientNet V2-M
ImageNet acc.	84.4% top-1	Comparable
Training speed	~11× slower	~11× faster
Parameters	~66M	~53M

03 Methodology

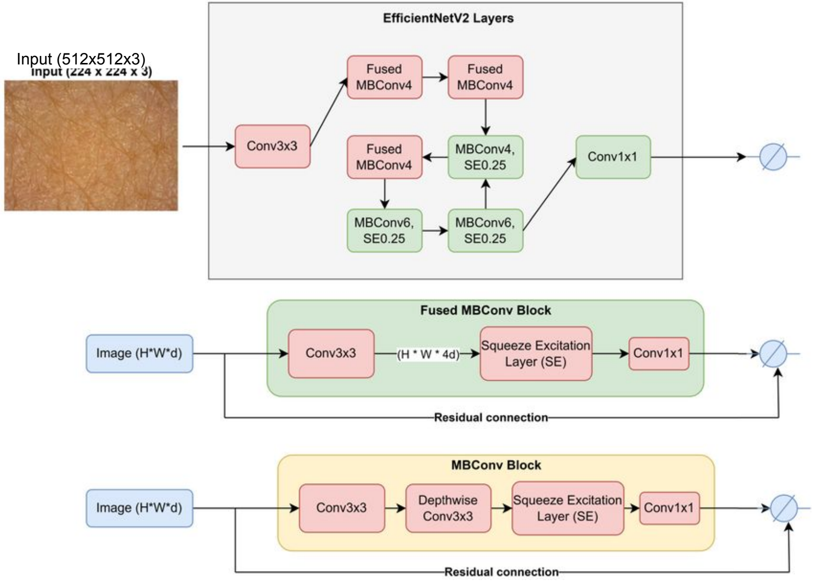
1. Baseline: EfficientNet + standard MCE
    a. Segmented original data
        i. V2-M → 42.18% best acc.
        ii. B7 → 40% best acc
    b. Non-segmented original data
        i. V2-M → 45% best acc.
2. **Frozen vs. fine-tuned layers** → fine-tuning needed
3. Added **CORN ordinal LF**
4. Manual data curation
    a. Segmented data
        i. V2-M → 51% test acc
        ii. B7 → 53% test acc
    b. Non-segmented data
        i. V2-M → 54% test acc

Data quality, not architecture, was the bottleneck!

04 CNN Architecture

EfficientNetV2-M:

- 52.9M parameters (fine-tuned)
- Fused-MBConv blocks→ faster training on GPUs
- Input: 600×600 (segmented) or **512×512** (non-segmented)
- Classification head: 1280 features → 10 classes



05 Model Training

Pytorch

Loss function: **CORN (Ordinal CE)**

- Type 1 → 2a → … → 4c
- Penalizes predictions by distance from true class
- 10 classes → 9 binary classifiers

Hardware/ training time:

- Google Colab T4 GPU (16GB VRAM)
- ~8.5GB per run
- ~14 min/epoch → ~4.5 hours total

Hyperparameters:

- **Optimizer**: AdamW
- **LR**: $3\times10^{-4} \rightarrow 1\times10^{-6}$ (cosine annealing)
- **Batch size:** 8 × 4 accumulation = 32 effective
- **Epochs**: 20
- **Early stopping** patience: 5
- Mixed precision (FP16) via PyTorch AMP

Data split:

- 70% train (~10k images)/ 15% val (~2k) / 15% test (~2k)

06 Results

Accuracy:

0.537 (53.7%)

Within-1 Accuracy:

0.887 (88.7%)

F1 Score:

0.535

	Precision	Recall	F1-Score	Support
1	0.9	0.91	0.9	225
2a	0.71	0.78	0.74	219
2b	0.6	0.55	0.57	186
2c	0.58	0.55	0.57	184
3a	0.5	0.39	0.44	180
3b	0.44	0.46	0.45	192
3c	0.47	0.54	0.5	182
4a	0.35	0.46	0.4	180
4b	0.32	0.27	0.3	260
4c	0.5	0.47	0.48	324

Accuracy			**0.54**	2132
Macro avg	0.54	0.54	0.54	2132
Weighted avg	0.54	0.54	0.53	2132

06 Results

Accuracy:

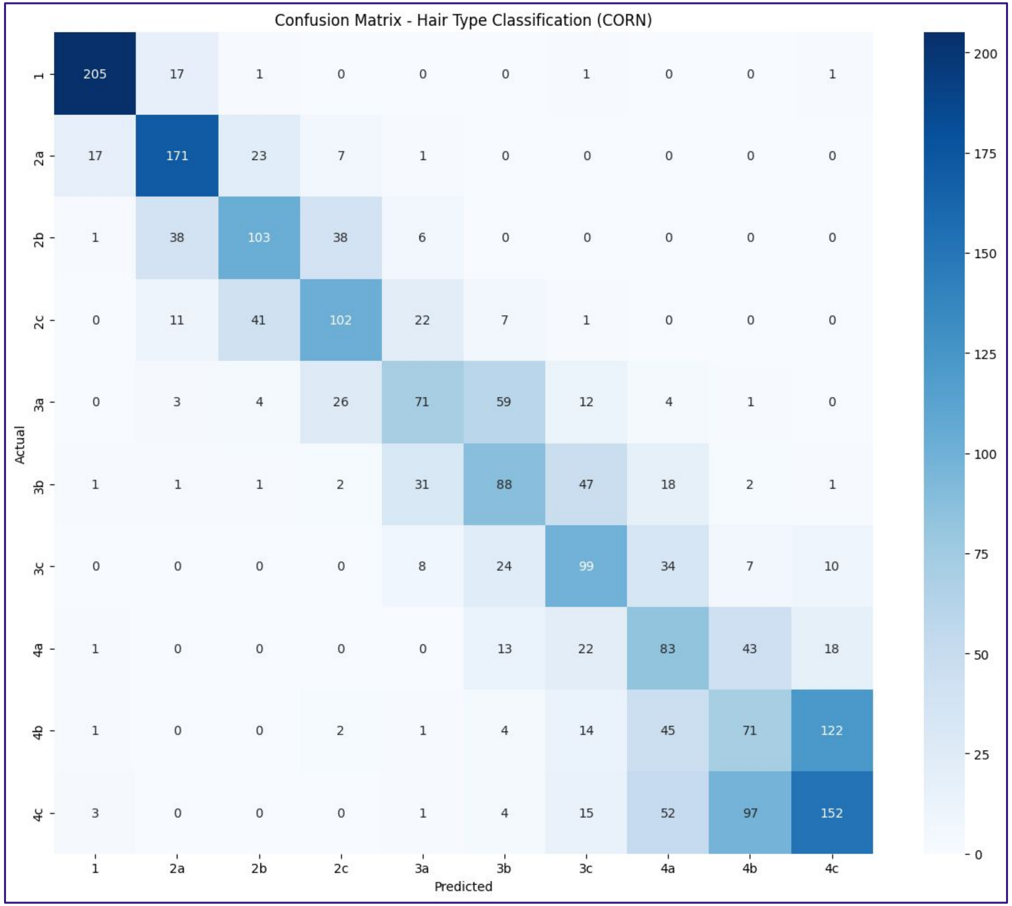
0.537 (53.7%)

Within-1 Accuracy:

0.887 (88.7%)

F1 Score:

0.535



06 Results - Discussion

Key Observations:

- Type 1: Easiest to classify with ~90% F1
- 2a-3c : Consistent confusion with adjacent curl patterns (Moderate Performance)
- 4a-4b: Lowest F1
- 4c: Overlaps heavily with 4a/4b

Model Observations:

- Captures broad categories well, struggles with fine-grained distinctions
- Primary bottleneck: dataset quality - ground truth classification
- Better performance on non-segmented images

07 Conclusion

Built end-to-end data pipeline

SerpAPI scraping
↓
YOLO filtering
↓
augmentation
↓
MediaPipe segmentation

Efficient NET-V2M for fine-grained task

Transfer learning enabled strong performance

Data quality was main bottleneck

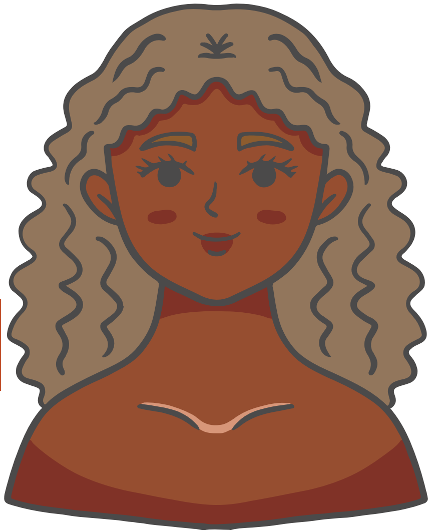
Manual curation and segmentation improved results

Our approach demonstrates strong potential for automated hair type classification, with accuracy poised to improve as data quality improves.

Hair care companies could utilize this to improve specialized product recommendations and sales.

Now, to the Application....

HairNet



CREDITS: This presentation template was created by Slidesgo, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Please keep this slide for attribution