Ensembling Deep Features with LightGBM for Perceptual Quality Assessment of Filtered Images

Sergio Sanz-Rodríguez
Berlin, Germany
sergio.sanz.rodriguez@gmail.com

***Abstract*— The increasingly popular use of aesthetic filters on social media has introduced new challenges in the field of Image Quality Assessment (IQA), as traditional distortion-based metrics are not designed to capture the subjective and content-aware nature of filter-based enhancements. This paper presents a no-reference IQA method that combines deep feature extraction with gradient-boosted decision trees to assess the perceptual quality of filtered images. Specifically, the proposed model employs three pre-trained deep learning networks to extract an image feature vector, which is fed into an ensemble of eight LightGBM models. The final Mean Opinion Score (MOS) prediction is obtained by averaging the outputs of these individual models. The model outperforms the baseline in terms of PLCC (0.862 vs. 0.543), SROCC (0.853 vs. 0.516), and RMSE (0.073 vs. 0.121), while offering lower memory usage and faster training times. Due to its reduced complexity and native GPU support, the proposed approach is well-suited for deployment scenarios where computational efficiency is essential.**

Keywords—LightGBM, deep learning, ensemble learning, image quality assessment, perceptual quality, filtered images

I. Introduction

The widespread sharing of images on social media platforms has popularized the use of aesthetic filters, making them a dominant form of visual expression. Unlike traditional distortions such as compression artifacts, noise, or blurring, these filter-based modifications are often subjective and content-aware, aiming to enhance visual appeal rather than preserve fidelity. Consequently, assessing the perceived quality of such images presents a significant challenge for computational Image Quality Assessment (IQA) methods. Conventional approaches—such as the Video Multimethod Assessment Fusion (VMAF) metric [1], the Structural Similarity Index (SSIM) and its variants [2], or the ITU-T P.1024.3 Recommendation [3]—are primarily optimized for distortion-related degradations and are less effective for perception-driven alterations.

Wu et al. [4] proposed a state-of-the-art, no-reference, learning-based IQA model specifically designed to predict the perceived quality of filter-manipulated images by human observers. The architecture employs a dual ConvNeXt backbone [5] with two parallel feature extraction pathways: one emphasizing distortion-aware (image quality) features, and the other focusing on filter-aware features. These representations are then fused through local and global contextual attention mechanisms, followed by a regression head that applies a weighted pooling strategy over the spatially fused features to generate a Mean Opinion Score (MOS) aligned with human perceptual judgments.

In this paper, presented as part of the VCIP 2025 Image Manipulation Quality Assessment Challenge [6], a novel no-reference, multi-model IQA model is described. The proposed approach combines feature extraction from multiple deep learning backbones with an ensemble of LightGBM regressors [7] to estimate subjective image quality. Specifically designed for filter-altered images, the method leverages tailored architectural and training strategies to enhance prediction performance, achieving third place in the challenge.

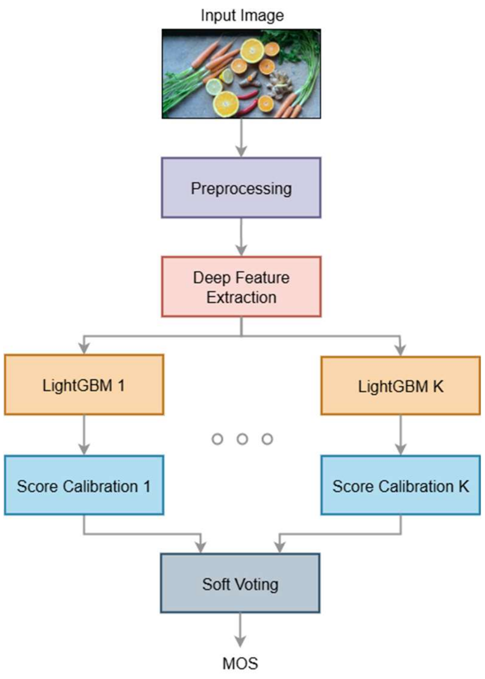

Fig. 1. The proposed no-reference, multi-model IQA pipeline

This alternative IQA model is inspired by the standardized approach described in [3] for no-reference perceptual quality assessment of compressed video, which employs Random Forests rather than relying solely on deep learning. The method is evaluated on the official challenge dataset and benchmarked against the baseline method described in [4], whose reference implementation is available at [6].

The remainder of this paper is organized as follows. Section II provides an overview of the proposed IQA model. Sections III–VI detail the key components of the processing pipeline. Section VII presents the experimental setup, evaluation metrics, and performance results. Finally, Section VIII concludes the paper and discusses a potential direction for future work.

II. General Overview

Fig. 1 illustrates the proposed multi-model IQA pipeline. First, the input image is preprocessed to match the format expected by the deep feature extractor model, which then generates a feature vector describing the image’s properties.

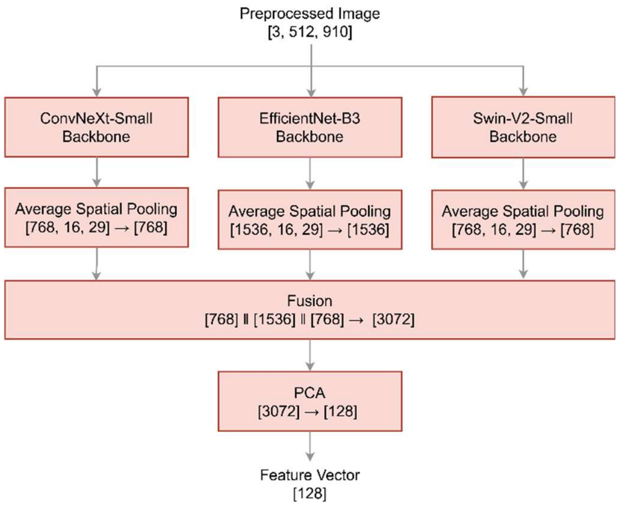

Fig. 2. Deep feature extraction module

The feature vector is then processed by K LightGBM models, each generating a MOS prediction. K is set to 8 in the experiments. These predictions are aggregated through soft voting by calculating their arithmetic mean, and the final MOS estimate is calibrated to enhance prediction accuracy. The implementation of this pipeline is publicly available at [8].

III. Image Preprocessing

Due to the high resolution of the training images (1920×1080 pixels), they are first downscaled to 910×512 pixels to reduce computational load and better accommodate the CPU and GPU resources. After resizing, the image is normalized to the [0, 1] range and standardized using the mean and standard deviation values from the ImageNet dataset [9]: mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225].

IV. Deep Feature Extraction

The goal of this module is to analyze the preprocessed image and extract a feature vector that effectively represents its properties. Deep learning models, such as Convolutional Neural Networks (CNNs) and Transformers [10], offer an efficient means of extracting rich information from images. Numerous such models are available in the literature, many of which have existing implementations in PyTorch [9].

The process for feature extraction is illustrated in Fig. 2 and proceeds as follows:

1. Three pre-trained backbone networks process the image, each producing a tensor of shape $[C, H, W]$, where C is the number of feature channels and $[H, W]$ are the spatial dimensions. This approach leverages *transfer learning*—i.e., no training is required. The backbones output feature maps with the same spatial resolution of 16×29 pixels, although their channel dimensions differ: ConvNeXt-Small (CNN) outputs 768 features, EfficientNet-B3 (CNN) yields 1536, and Swin-V2-Small (Transformer) produces 768.
2. A spatial pooling step then averages the features across the spatial dimensions to produce a single 1D feature vector per backbone.
3. The feature vectors from the three backbones are concatenated into a single 3072-dimensional vector, integrating complementary information from each model to improve feature richness and robustness.
4. Finally, the concatenated feature vector is reduced in dimensionality using Principal Component Analysis (PCA), which transforms the data into a smaller set of uncorrelated components while preserving as much variance as possible. In this case, 128 components were selected, as they capture around 99.6% of the total variance in the data.

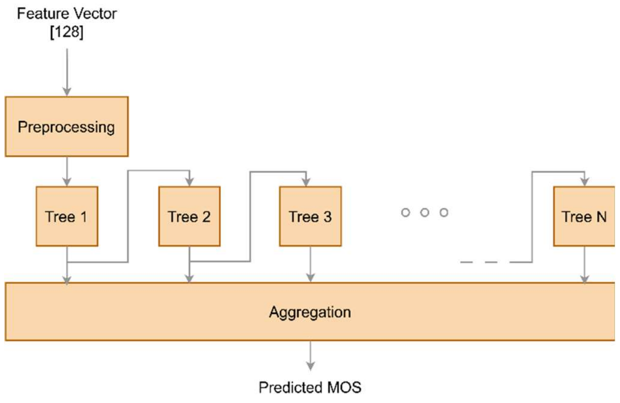

Fig. 3. General overview of the LightGBM architecture

V. The LightGBM Ensemble

LightGBM is a state-of-the-art gradient boosting model that combines multiple weak learners—typically decision trees—into a strong predictor. As shown in Fig.3, LightGBM first applies preprocessing operations to the input feature vector such as binning, which discretizes continuous features, and row and feature subsampling to reduce overfitting. Then, N weak learners are arranged sequentially, each one correcting the errors of its predecessors. This iterative boosting process improves performance by focusing on hard-to-predict samples. The MOS estimate is obtained by aggregating the partial predictions from all trees.

The predictive capabilities of this model are particularly beneficial for this task, as the training dataset is imbalanced, specifically exhibiting a relative underrepresentation of high and low MOS labels. LightGBM's intrinsic focus on hard-to-predict regions helps mitigate this potential issue. Moreover, its fast training speed and low memory usage enable efficient and extensive hyperparameter tuning, allowing for deeper exploration and more precise model optimization.

Better generalization is achieved by ensembling multiple LightGBM models trained under different cross-validation folds (see Section VII), which enhances robustness and reduces variance in the final prediction.

VI. Score Calibration

Before applying soft voting, the predicted MOS from each model *n* is adjusted using the following linear transformation:

$$\widehat{MOS}_n = a_n \times MOS_n + b_n \quad (1)$$

with coefficients a_n and b_n, resulting in a 6.85% reduction in Root Mean Squared Error (RMSE) in the experiments.

VII. Experiments and Results

The LightGBM models were trained on image feature vectors extracted from the three backbone networks. Each feature vector represents the characteristics of one of the 288 filter images used for training [4].

TABLE I. LIGHTGBM HYPERPARAMETER SEARCH SPACE

Hyperparameter	Minimum Value	Maximum Value
No. estimators	50	200
Learning rate	0.01	1.0
L1 regularization	0.0001	1.0
L2 regularization	0.0001	1.0
Max. depth	3	8
No. leaves	4	24
Max. bins	255	768
No. studies	16	
No. trials / study	4000	

The following subsection describes the training and cross-validation process aimed at promoting generalization, followed by an analysis of the experimental results.

A. Experimental Setup

An 8-fold cross-validation strategy was employed, where the data was divided into eight parts. In each iteration, seven folds were used for training and one—referred to as the Out-Of-Fold (OOF) set—was used for validation. This process was repeated eight times, ensuring that each fold served as the validation set once.

In problems with limited training data (for example, 288 samples), using a relatively large number of folds helps ensure a sufficient amount of data for training in each iteration. However, increasing the number of folds also results in higher computational cost, longer training time, and a larger pipeline.

Another challenge when working with small datasets is the increased risk of overfitting. This risk is mitigated by carefully selecting the hyperparameters of each LightGBM model. To this end, the Optuna tool [11] was used to automatically search for effective combinations of hyperparameters that either minimize or maximize a given performance criterion. Optuna is an iterative optimization tool that uses Bayesian optimization to find to efficiently explore the hyperparameter space. For each hyperparameter, a well-defined search range must be chosen to avoid both underfitting and overfitting.

Table I summarizes the search space used for the most important hyperparameters. The number of estimators defines how many trees are built in the LightGBM architecture. The Learning Rate (LR) scales the contribution of each tree to the final prediction. L1 and L2 regularization terms help reduce overfitting by penalizing model complexity. Maximum depth limits how deep each tree can grow, while the number of leaves sets an upper bound on the complexity of each individual tree. The maximum bin parameter controls how finely continuous input features are discretized into bins; higher values allow more precise splits, but at the cost of increased memory usage and computational overhead. The number of trials refers to how many hyperparameter combinations are evaluated in each study, and the number of studies corresponds to how many times the entire search process is repeated to ensure robust results.

Training was performed on a workstation equipped with an Intel Core i9-9900K CPU and an NVIDIA GeForce RTX 4070 GPU. For each fold k, two sets of eight Optuna studies were conducted. In the first set, each study optimized hyperparameters to minimize the Mean Squared Error (MSE), while in the second set, studies aimed to maximize the coefficient of determination (R^2). This resulted in a total of 16 candidate models per fold. Each model was then evaluated using the average of the Pearson Linear Correlation Coefficient (PLCC) and the Spearman Rank-Order Correlation Coefficient (SROCC) between the ground-truth and OOF predictions. The best model for fold n was the one achieving the highest PLCC-SROCC average among the 16 candidates.

TABLE II. BEST TRAINING SETUP PER CROSS-VALIDATION FOLD

Fold No.	No. Estimators	Learning Rate	Max. Depth	No. Leaves	Max. Bins
1	189	0.472	7	7	512
2	160	0.594	6	6	512
3	80	0.267	7	18	512
4	151	0.682	5	18	512
5	128	0.616	7	24	255
6	193	0.308	6	21	255
7	141	0.233	6	19	255
8	137	0.650	4	20	255

TABLE III. COMPARISON OF BASELINE AND PROPOSED METHODS ON OOF SETS USING PLCC, SROCC, AND RMSE

	Baseline			Proposal		
Fold No.	PLCC	SROCC	RMSE	PLCC	SROCC	RMSE
1	0.608	0.609	0.115	0.875	0.888	0.067
2	0.435	0.447	0.131	0.874	0.874	0.070
3	0.514	0.445	0.118	0.842	0.778	0.074
4	0.673	0.573	0.106	0.858	0.835	0.073
5	0.489	0.471	0.127	0.835	0.860	0.079
6	0.466	0.558	0.134	0.810	0.798	0.088
7	0.503	0.307	0.114	0.917	0.877	0.051
8	0.628	0.563	0.123	0.877	0.835	0.076
Ave.	0.539	0.497	0.121	0.861	0.843	0.072
Glob.	**0.543**	**0.516**	**0.121**	**0.862**	**0.853**	**0.073**

For comparison purposes, the reference implementation of the baseline model described in [4], available at [6], was also trained using 8-fold cross-validation on the same workstation. The training process followed the default deep learning configuration: input image size of 480×270 pixels, 80 training epochs, Adam optimizer, batch size of 4, MSE loss, an initial LR of $5e^{-6}$ with a decay factor of 0.9, and early stopping.

B. Experimental Results

Table II presents the training configurations that produced the best-performing model for each of the eight cross-validation folds. Each row corresponds to the configuration that achieved the highest PLCC-SROCC average in its respective fold, based on the evaluated checkpoints.

As shown in the table, each fold results in a different LightGBM configuration and, consequently, a unique set of model parameters. This diversity arises from the use of cross-validation, where the training data and validation splits vary for each fold. Such variability enhances the overall robustness of the system, as averaging the predictions across folds helps reduce overfitting and improves the generalization of the final MOS prediction. Notably, a few of these models use close to 200 estimators, highlighting the effectiveness of restricting the search range to prevent overfitting.

Table III presents a comparison of the baseline and proposed methods on the OOF sets, evaluated in terms of PLCC, SROCC, and RMSE. The results are reported per fold, as an average across folds, and globally after concatenating the OOF data points.

As shown, the proposed method outperforms the baseline, achieving a PLCC of 0.862 compared to 0.543, an SROCC of 0.853 versus 0.516, and an RMSE of 0.073 versus 0.121.

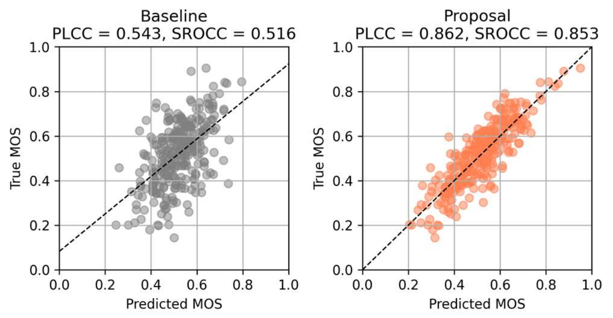

Fig. 4. Correlation between predicted and true MOS on the OOF sets for the baseline (left) and proposed (right) models

Fig. 4 shows the correlation between the predicted and true MOS for both methods on the OOF sets. As observed, the scatter plot corresponding to the proposed method exhibits fewer outliers compared to the baseline, which reflects its improved PLCC and SROCC performance. Furthermore, the linear regression curve (dashed line) closely aligns with the identity function, indicating that the model captures the general trend well.

When evaluating the models on the test dataset, which consists of 72 unlabeled samples, the resulting predicted MOS distributions are illustrated in Fig. 5. The figure indicates that the proposed IQA model produces a broader distribution than the baseline, suggesting that it captures greater variance in the data. This may enable improved predictions, particularly at the lower and higher ends of the MOS scale, which were underrepresented in the training set. The achieved PLCC and SROCC values are 0.674 and 0.654, respectively.

From the storage and memory efficiency perspective, the proposed pipeline occupies approximately 419 MB on disk and requires about 2.1 GB of CPU memory and 573 MB of GPU memory during inference. In contrast, the baseline's dual ConvNeXt-Large architecture (without an 8-fold ensemble) occupies 1.5 GB on disk and uses approximately 3.0 GB of CPU memory and 1.6 GB of GPU memory during inference.

VIII. Conclusions and Further Work

This paper presented a no-reference, multi-model IQA method for predicting the MOS produced by filters aimed at enhancing image visual quality. The approach leverages three pre-trained deep learning networks to extract a total of 3072 features from images, which are subsequently reduced to 128 via PCA. The resulting feature vector is fed into an ensemble of eight LightGBM models, whose individual outputs are averaged to produce the final MOS prediction. This ensemble strategy enhances the model's robustness and reduces the risk of overfitting. The proposed approach demonstrates improved performance over the baseline during cross-validation.

In addition, the method's relatively low complexity, memory requirements, and fast training times make it well-suited for real-world machine learning workflows. It is also worth highlighting that LightGBM provides native GPU acceleration [12], allowing both the deep feature extractor and LightGBM ensemble to run on the same device, thereby reducing data transfer overhead between CPU and GPU.

Special attention was also given to the training process, which was carefully designed with cross-validation to fully exploit LightGBM's learning capabilities and to improve generalization through ensembling.

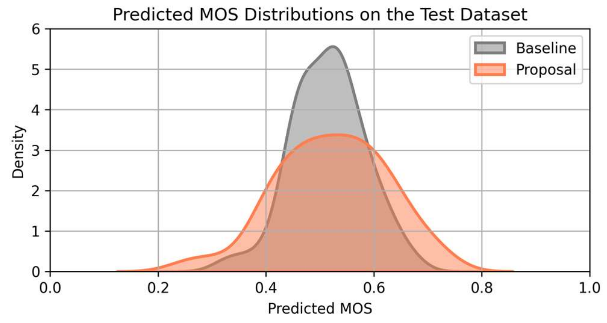

Fig. 5. Predicted MOS distribution on the test dataset for the baseline and the proposed models

As future work, the deep feature extraction stage could be made trainable, at least partially. This modification would enable the generation of problem-specific image feature vectors, potentially enhancing predictive accuracy. However, the current pipeline shown in Fig. 1 would not support this approach, as it would introduce data leakage during training. Instead, the feature extraction module should be integrated within each branch of the ensemble, allowing for joint training of the backbones and LightGBM models under a K-fold cross-validation strategy. This adjustment would significantly increase computational cost and memory usage, as each ensemble branch would require its own dedicated deep feature extractor. Nonetheless, in real-world scenarios where higher predictive accuracy is critical, this trade-off may be justified.

References

[1] Z. Li, A. Aaron, I. Katsavounidis, A. Moorthy, and M. Manohara, "Toward a Practical Perceptual Video Quality Metric," Netflix Tech Blog, 2016.

[2] Z. Wang, E.P. Simoncelli, and A.C. Bovik, "Multi-scale Structural Similarity for Image Quality Assessment," Proc. 37th Asilomar Conf. Signals, Syst. Comput., Pacific Grove, USA, vol. 2, pp. 1398-1402, 2003.

[3] ITU-T Recommendation P.1204.3, "Video Quality Assessment of Streaming Services over Reliable Transport for Resolutions up to 4K with Access to Full Bitstream Information," International Telecommunication Union, Jan. 2020.

[4] X. Wu, J. Lou, Y. Wu, W. Liu, P.L. Rosin, G.B. Colombo, S. Allen, R. Whitaker, and H. Liu, "Image Manipulation Quality Assessment," IEEE Transactions on Circuits and Systems for Video Technology, vol. 35, no. 4, pp. 3450-3461, April 2025.

[5] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), New Orleans, LA, USA, pp. 11976–11986, 2022.

[6] P.L. Rosin, H. Liu, J. Liu, Y. Liang, Y. Li, Y. Ma, X. Wu, and W. Zou, "Image Manipulation Quality Assessment Challenge," VCIP 2025, 2025. [Online]. Available: https://jiangliu5.github.io/imqac.github.io/

[7] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly Efficient Gradient Boosting Decision Tree," Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NeurIPS), Long Beach, CA, USA, pp. 3149–3157, Dec. 2017.

[8] S. Sanz, "LightGBM Ensemble IQA," 2025. [Online]. Available: https://github.com/sergio-sanz-rodriguez/LightGBM-Ensemble-IQA

[9] PyTorch, "Model and Pre-trained Weights," PyTorch website. 2025. [Online]. Available: https://docs.pytorch.org/vision/main/models.html.

[10] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows," Proc. IEEE/CVF Int. Conf. Comp. Vis. (ICCV), Montreal, QC, Canada, pp. 10012–10022, Oct. 2021.

[11] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," Proc. 25th ACM SIGKDD Int. Conf. Knowledge Discovery & Data Mining (KDD), Anchorage, AK, USA, Aug. 2019, pp. 2623–2631.

[12] LightGBM, "LightGBM GPU Tutorial," LightGBM website, 2025. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/GPU-Tutorial.html