IPARC Image Transformation Analysis: Comparing Machine Learning Frameworks for Structuring Element Position Detection

Anonymous authors

Abstract

This study addresses the **IPARC** challenge, which involves deriving interpretable functions for image-to-image transformations using operations such as **Dilation** and **Erosion** with specified Structuring Elements (**SE**). The task is formulated as a multi-class classification problem aimed at identifying the sequence and position of SEs within the transformation pipeline. A range of machine learning models, including Logistic Regression, Decision Trees, Bagging, Boosting, and Support Vector Machines (SVMs), were employed to tackle this problem.

To enhance model performance, the methodology integrates dimensionality reduction via Principal Component Analysis (PCA), feature engineering to incorporate operation and SE-specific attributes, and extensive hyperparameter tuning through cross-validation. Ensemble techniques such as Random Forests and XGBoost were leveraged to address the overfitting challenges posed by a small, imbalanced dataset. Experimental results reveal consistent performance trends across models and demonstrate robustness in SE position prediction, irrespective of operational variations. The comparative analysis further provides insights into the interplay between different SEs and operations. This work highlights the applicability and limitations of machine learning methods in addressing structured image transformation tasks within the **IPARC** framework.

1 Overview of the Challenge

The goal of the **IPARC** challenge is to evaluate whether an inductive program learner can be designed to infer a human-readable function capable of converting an input image into its corresponding output image. This conversion must leverage a predefined library of operations, including ***Dilation*** and ***Erosion***, alongside 8 specified ***Structuring elements (SE)*** and color bands. In **Category A**, the learner's primary objective is to accurately reconstruct the sequence of structuring elements used to perform the transformation. The initial task involves identifying the exact location of the structuring element within the input-output transformation process.

2 Hypothesis

For a given transformation from an input image to an output image, characterized by the use of operations such as ***Dilation*** or ***Erosion*** and specific ***Structuring Elements (SEs)***, it is possible to train a machine learning model to identify the precise position of the SE within the sequence of operations. Furthermore, the model should be capable of determining whether a particular SE was omitted entirely in the transformation process. In addition to this, we hypothesize that a comparative study can be conducted to evaluate how well the model predicts transformations involving either different Structuring Elements within the same operation (e.g., ***Dilation SE4*** vs. ***Dilation SE3***) or the same ***Structuring Element(SE4)*** across different operations (e.g., ***Dilation*** vs. ***Erosion***). Achieving high accuracy in identifying these positions, omissions, and inter- or intra-operation relationships would significantly enhance our ability to reconstruct

the function responsible for mapping the input to the output. This capability would serve as a key step toward solving the **IPARC** challenge, wherein the goal is to uncover the human-understandable logic behind the image transformation pipeline.

3 Methodology

3.1 Problem Formulation

The **IPARC Challenge** aims to uncover human-comprehensible logic in the transformation of input images (I_{input}) to output images (I_{output}) using operations such as **Dilation** and **Erosion** along with specific **Structuring Elements (SEs)**. The objective is to train a machine learning model capable of predicting the usage, sequence, and position of Structuring Elements within these operations, while validating intra-operation and inter-operation relationships.

The transformation process T can be mathematically represented as:

$$I_{\text{output}} = T_8(T_7(\dots T_2(T_1(I_{\text{input}}, O_1(S_1)), O_2(S_2)) \dots, O_7(S_7)), O_8(S_8))$$

where:

- $O_i \in \{\text{Dilation}, \text{Erosion}\}$ is the operation.
- $S_i \in \{\text{SE}_1, \dots, \text{SE}_8\}$ is the Structuring Element.

In **Category A** solutions, apply 4 Dilation operations followed by 4 Erosion operations, each using a unique Structuring Element.

3.2 Task for the Model

Given I_{input} and I_{output}, the machine learning model is tasked to:

- **Predict the position** p of a given Structuring Element S_m and Operation O_l in the sequence:

  $$p = \text{Model}(O_l, S_m), \quad \text{where } O_l \in \{\text{Dilation}, \text{Erosion}\},\ m \in \{1, 2, 3, 4, 5, 6, 7, 8\}$$

  $p \in \{0, 1, 2, 3, 4\}$. The value of p indicates the position of the structuring element in the operation used. A value of $p = 0$ indicates that the pair (O_l, S_m) was not used.

3.3 Experimental Design & Data Pre-Processing

To validate the model and test the hypothesis, the following experiment is conducted:

1. **Hypothesis Validation:** The accuracy of each model is measured to draw conclusions regarding the hypothesis in the context of the IPARC Challenge.
2. **Machine Learning Models:** A variety of machine learning algorithms are evaluated, including Linear Models, Decision Trees, and Support Vector Machines (SVMs).
3. Techniques such as *Cross-Validation*, *Bagging*, *Gradient Boosting*, and *Feature Selection* are employed to ensure robust performance and enhance accuracy.
4. **Class Labels:** Each task involves 4 input-output image pairs with their solutions. It is possible to set the Structural Element and the Operation that we are training and testing the model for. The dataset uses class labels that correspond to the SE position: '0' for the SE absence, and '1' to '4' indicating SE position in Dilation or Erosion sequence, creating 5 distinct classification categories. In case of Erosion, even if the SE position is from '5' to '8', we do position - 4 so that we can use the same model for Erosion as well.

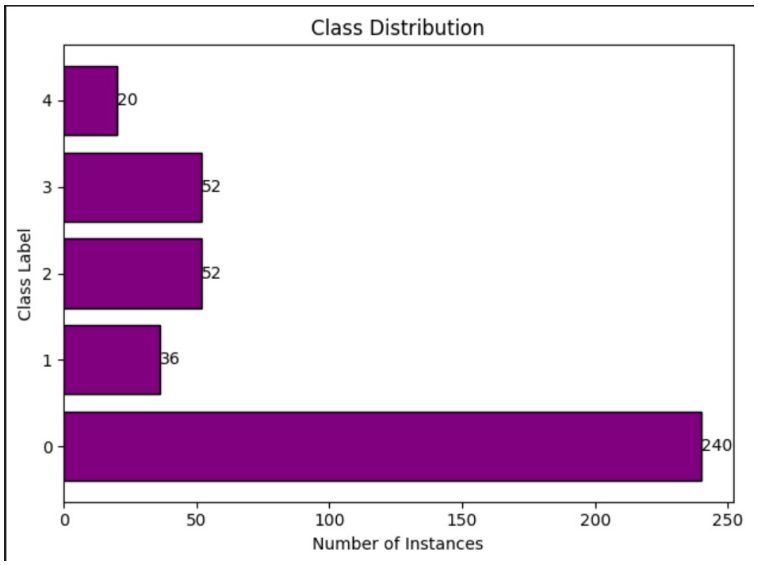

Figure 1: Class Label Distribution

5. **Feature Representation:** The algorithm processes 15×15 binary images, generating 450 pixel features. By adding Structuring Element pixel features and the operation (Dilation or Erosion), the total feature count increases to 459 (450 + 8 SE + 1 Operation). Experimental findings showed a notable accuracy enhancement with this expanded feature set.

4 Methods and Approaches

All experimental results presented are for Erosion as the operation and SE4 as the Structuring Element. The distribution of the class labels is displayed alongside and highlights an imbalanced dataset with limited data relative to the feature space of the input. This imbalance poses challenges for modeling and may lead to biased predictions, necessitating careful handling during model training and evaluation.

4.1 Principal Component Analysis (PCA)

We applied **Principal Component Analysis** (PCA) dimensionality reduction technique that transforms data into a set of uncorrelated components, prioritizing directions of maximum variance) to reduce feature space dimensionality. After mean-centering the data, we selected 94 components that preserve 90% of the original data variance.

Moreover, a **90-10 split** is applied to the input data. A random seed of 60 is used to maintain reproducibility, and the data is shuffled before splitting. The dataset is split into distinct subgroups based on specific characteristics to ensure proper representation or analysis of each subgroup which inturn helps in maintaining same class label proportions among the training and test datasets. This results in the training data having a shape of (360, 94) and the test data a shape of (40,94).

4.2 Support Vector Classifier (SVC)

We trained the Support Vector Classifier (SVC) using class weights, applying cross-validation on the parameters: *kernel*, *tolerance level*, and the L*2 regularization parameter C* for improving performance by class imbalance, hyperparameter tuning, kernel selection and ensuring proper convergence during optimization. After training with regularization and performing 7-fold cross-validation, we achieved an accuracy of 80% on the test dataset.

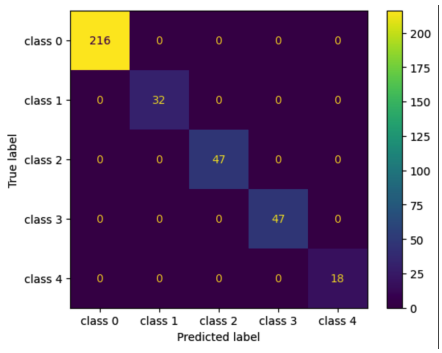

Figure 2: CM for Train

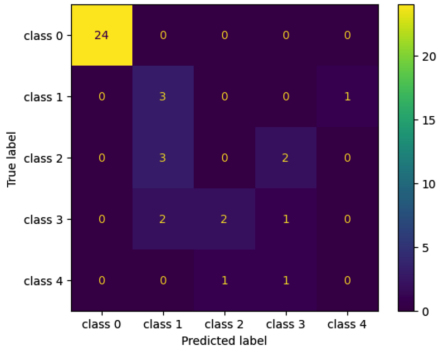

Figure 3: CM for Test

4.3 Logistic Regression

To address class imbalance in the training dataset, inverse class weights proportional to their frequencies were applied to the logistic regression model. L2 regularization was incorporated, and the hyperparameter C was fine-tuned through 7-fold cross-validation. The optimal model was selected based on cross-validation accuracy. This process resulted in a 80% accuracy on the test dataset.

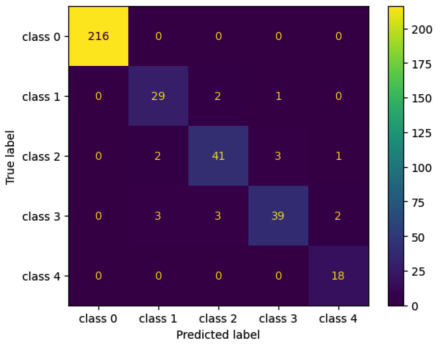

Figure 4: CM for Train

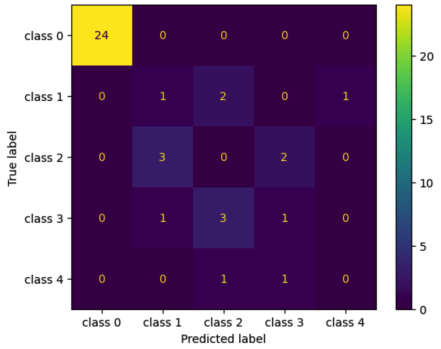

Figure 5: CM for Test

4.4 Decision Trees

We trained a decision tree model, initially allowing it to overfit by growing unconstrained. To address overfitting, we applied pre-pruning by tuning parameters like *maximum depth, minimum samples per split,* and *minimum samples per leaf* through cross-validation. Cost-complexity pruning was then employed, using the **ccp_alpha** parameter to iteratively remove leaves based on impurity until only a single internal node remained. These methods improved test accuracy to **65%**, up from **48%** for the unpruned tree and **40%** with pre-pruning alone.

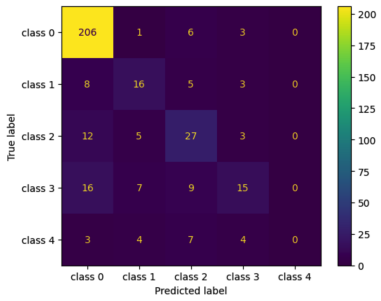

Figure 6: CM for Train

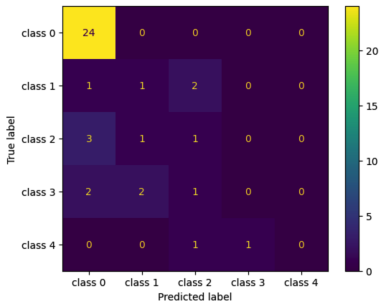

Figure 7: CM for Test

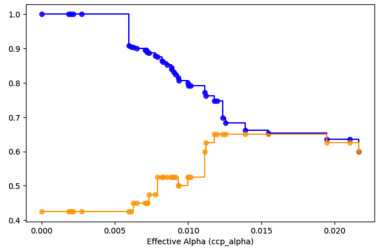

Figure 8: Effective Alpha

4.5 Bagging and Boosting Methods(XGBoost)

To address the high variance of the decision tree model, we employed ensemble methods, including bagging with RandomForest and ExtraTrees classifiers. Hyperparameter tuning was conducted (to improve its performance on a given dataset) using both RandomizedSearch (for initial exploration of the hyperparameter space to identify promising ranges) and GridSearch (for a focused, fine-grained search around the promising hyperparameters identified) by cross-validation across parameters such as $n\_estimators$, $max\_features$, max_depth, $min\_samples\_split$, $min\_samples\_leaf$, and $bootstrap$. For boosting, the **XGBoost** algorithm was utilized, with similar hyperparameter optimization applied through GridSearch. Both bagging and boosting approaches achieved a test accuracy of **60%.**

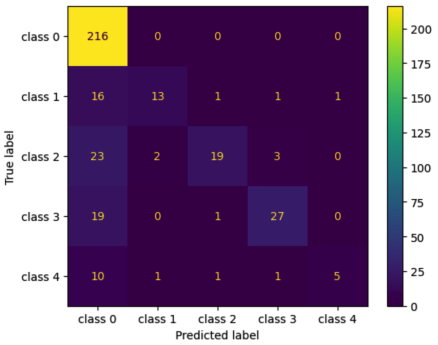

Figure 9: CM for Train

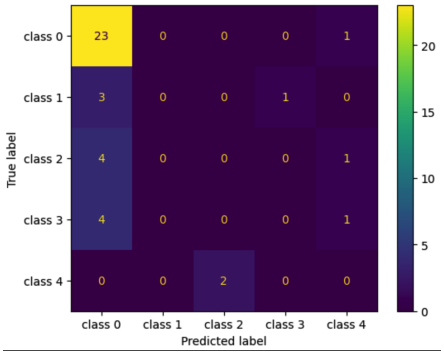

Figure 10: CM for Test

5 Experimental Results

In this section, we have included a comparative study of **Dilation SE4** vs. **Dilation SE6** and **Dilation SE4** vs. **Erosion SE4** to provide additional insights into the model's performance across different Structuring Elements and Operations.

5.1 Comparison of Test Accuracy of Dilation SE4 and Erosion SE4

	Logistic Regression	Decision Tree(pruned)	Bagging & Boosting	SVM
Dilation SE4	0.65	0.60	0.675	0.7
Erosion SE4	0.65	0.65	0.69	0.70

Table 1: Test Accuracy Dilation SE4 vs Erosion SE4

5.2 Comparison of Test Accuracy of Dilation SE4 and Dilation SE7

	Logistic Regression	Decision Tree(pruned)	Bagging & Boosting	SVM
Dilation SE4	0.65	0.60	0.675	0.7
Dilation SE7	0.8	0.65	0.775	0.80

Table 2: Test Accuracy Dilation SE4 vs Dilation SE6

6 Conclusion and Future Work

The models showed significant overfitting due to the small dataset size. The complexity required by the multi-class classification task and the large feature set exceeded what the dataset could support, resulting in low prediction accuracy across all models. Notably, the accuracy was unaffected by the operation type (**Dilation** or **Erosion**) or the specific **Structuring Element**, suggesting that task performance is largely independent of these variations. Furthermore, the accuracy rankings among models—Logistic Regression, Decision Trees, Bagging, Boosting, and SVM—remained stable across experiments, indicating consistent trends in model behavior and robustness when applied to different configurations of the dataset. Future work could focus on addressing dataset limitations through advanced data augmentation techniques, such as rotation, flipping, and transformation-specific synthetic data creation. Simultaneously, leveraging deep learning models like Convolutional Neural Networks could enhance the extraction of complex features, enabling more accurate predictions for intricate image transformations. Furthermore, incorporating transfer learning by utilizing pretrained models on related tasks may significantly reduce computational overhead while improving performance, especially for datasets with limited labeled examples. These approaches collectively offer a promising direction for enhancing the robustness and scalability of solutions for structured image transformation tasks.

7 References

- François Chollet. "On the Measure of Intelligence." *CoRR*, abs/1911.01547 (2019). arXiv:1911.01547. `http://arxiv.org/abs/1911.01547`
- IPARC Challenge V2. `https://github.com/ac20/IPARC_ChallengeV2/tree/main`
- Chen, T., & Guestrin, C. "XGBoost: A Scalable Tree Boosting System." *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, 2016, pp. 785-794.
- He, H., & Garcia, E. A. "Learning from Imbalanced Data." *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263-1284, 2009. DOI: 10.1109/TKDE.2008.239.
- Jolliffe, I. T. *Principal Component Analysis*. Springer Series in Statistics, 2002.