1.Introduction

Feature engineering is the key part of the machine learning process, especially for applications of image data which are high-dimensional. Feature engineering is the way that features are uncovered, changed, and chosen from the original data in order to make it better for machine learning models. In turn, feature engineering not only makes the model more accurate but also lessens the cloud's computational load, thus making it work faster and be more understandable.

In this project, the method of characteristic design is used to classify grayscale images. The images in this group are monochrome and can present a particular difficulty due to their lower number of dimensions in comparison to RGB images. This means the part of feature extraction that is strictly in line with the idea is twice as important as it would be otherwise. To solve this problem, we use a blend of old-fashioned and new methods to extract both local and high-level features from the images.

The primary techniques used in this project include:

Local Binary Patterns: It is a texture-based method of feature extraction that checks what pixels around a given one are.

Histogram Of Oriented Gradients (HOG): It is a gradient-based structural analysis method that obtains information about shape changes from their derivatives in pixel values.

Convolutional Neural Networks (CNNs): Models that exploit deep learning techniques to extract hierarchical features from image data in an unsupervised manner are then convolutionally trained for image classification.

After extracting features using these methods, Principal Component Analysis (PCA) is a technique to reduce the number of dimensions in the data. This guarantees that individual feature sets are still computationally manageable, however, they contain the principal information. The classification is then true using the implementation of the Support Vector Machine approach, which is known for dealing with the complexity of classification problems in a very robust manner.

The ethical framework guiding this project is Utilitarianism the foundation paradigm of which is to make moves to bring maximum utility. By driving efficient feature engineering methods, the project plans to develop models that are not only outstanding but also applicable in a variety of levels and settings, such as resource deprived places to high-risk sectors like healthcare and security.

This project is therefore extremely relevant because the feature engineering facet directly impacts on the following:

Model Accuracy: In other words, better features means greater accuracy in classification.

Computational Efficiency: Data space reduction thus ensures quick training and inference.

Interpretability: Conscious experience features help users to more easily grasp and believe the rationality of the model.

It is a comprehensive report which gives step by step accounts of the project, the methods, and the results guaranteeing the inclusion of simple language for readers of both high and low expertise levels. The traditional to modern techniques mixed application is evidence of one result obtained from a carefully done investigation by this project that improving the model's performance and usability is possible.

2.Objectives

The first aim of this project is to develop a classification pipe and strong feature engineering of grayscale datasets with help of both the traditional and modern techniques. Together as a team, we are going to delve into and apply approaches that best convert raw

images into kinds of representations that are meaningful to the performance of machine learning models. The purpose of the project is to compare the efficiency and accuracy of different feature extraction techniques such as, Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG), and Convolutional Neural Networks (CNN).

Moreover, the project stresses the point of dimension reduction using Principal Component Analysis (PCA), which has been proven to be a very effective computational technique while keeping all the important information from extracted features. Along with the above-mentioned goal, the next one is to prove that the usefulness of the features derived from distinct processes can be put together to obtain a complete representation of the data, thus ,The best classification of the data is obtained with Support Vector Machines (SVM).

This project is also a collaborative effort, which has its focus on building a better understanding of the role of feature engineering in image classification. Sharing responsibilities and insights, our team aims at critically analysing and addressing challenges associated with different feature extraction and classification strategies. The team will visualize and evaluate the performance of the models in detail to provide actionable insights and set a foundation for scalable machine learning pipelines in real-world applications.

3.Background Information

Feature engineering in image processing is the process of transforming raw pixel data into meaningful, high-quality features that machine learning models can use effectively. It has evolved over time, with techniques ranging from traditional handcrafted feature extraction to modern deep learning-based methods. This project combines these approaches to create a comprehensive feature set for classifying grayscale images

Key Concepts and Techniques

local binary patterns (LBP):

- LBP is a simple and efficient method for summarizing local textures in an image.
- It involves comparing each pixel's intensity with its neighbours in a defined radius and assigning binary values (0 or 1) based on whether the neighbours are brighter or darker.
- These binary values are concatenated to form a unique pattern, which is then converted into a decimal value. A histogram of these patterns gives the texture features of the image.
- The LBP is computationally lightweight and can be used for applications such as face recognition or texture classification.
- **Strengths:** Robust to light variations and noise, and easy to compute and interpret.

Histogram of Oriented Gradients (HOG):

- HOG is the most frequently used feature descriptor in computer vision for object detection tasks, especially in pedestrian detection.
- It captures the distribution of gradient orientations in small regions of an image, known as cells. The calculation of gradients is determined by analysing changes in intensity in the horizontal and vertical directions.
- Each cell computes a histogram of the gradient direction, weighted with gradient magnitude. These histograms are normalized across the neighbouring cells in order to make them invariant to changes in lighting and contrast.
- **Strengths:** Effective for shape and structural information; invariant under minor geometric and lighting transformations.

Convolutional Neural Networks:

- CNNs are a class of deep learning models specifically designed to process grid-like data such as images.
- Unlike traditional methods that rely on handcrafted features, CNNs learn hierarchical features directly from the data.
- Early layers capture basic patterns like edges and textures.
- Deeper layers capture complex patterns, such as object shapes and high-level representations.
- Pre-trained CNN models, such as VGG16 trained on datasets like ImageNet, might serve for feature extraction tasks. If the classification layers of these networks are removed, the output from intermediate layers provides meaningful feature maps of input images.
- **Strengths:** Learning complex and high-level features is done automatically, robust under a variety of data distribution.

Dimensionality Reduction using Principal Component Analysis (PCA)

Most of the feature extraction methods result in high-dimensional data, which may be computationally expensive and prone to overfitting. PCA overcomes this by the following: it reduces the number of features and retains the most important information; it transforms the features into a new coordinate system where the axes (principal components) represent the directions of maximum variance in the data. Noise and redundancy are reduced by PCA, making it an essential step for scalable machine learning pipelines**.**

Historical Context and Evolution

Early Days:

- Primitive feature extraction methods like Local Binary Patterns (LBP) and Histograms of Oriented Gradients (HOG) were developed to be basic and easily understandable.
- These methods were mostly based on domain-level expertise since experts could use handcrafted patterns; therefore, they are limited to a specific application and come out short when not enough data is given.

Rise of Deep Learning:

- CNN-based automatic learning formed feature extraction.
- Pre-trained models such as VGG, ResNet, and Inception have robustly demonstrated their ability to represent feature variations across different datasets, as well as tasks.

Modern Practices:

- It is a common practice to use traditional methods synergistically with deep learning methods to exploit their strengths.
- The methodology of Dimensionality Reduction techniques such as PCA achieves the efficiency and scalability of feature sets for large datasets.

Context of the Project

This project is built on a dataset that consists of grayscale images. The images are separated into two groups: the training set and the testing set.

The images are grouped into certain classes, and the classification needs to be made according to the class that the image belongs.

This project incorporates Traditional Methods (LBP, HOG): to obtain interpretable and lightweight features; CNN-based image features: to exploit the outstanding capabilities of deep learning;

PCA: decreasing the number of feature dimensions, hence the speed of computation. The approach that these methods are combined with has allowed the classification problems in

real-world image data sets to be solved through both accuracy and efficiency..

4.Methodology or Approach

The methodology followed in this project is structured to process grayscale image datasets in a systematic way, feature extraction with meaningful information, and performance evaluation using SVMs.

Each step of the process is explained in detail below, emphasizing the rationale and implementation for better understanding.

1. Loading Data and Preprocessing

- **Overview:** This initial step serves to ensure the dataset is standardized in a consistent and orderly form for subsequent processing.
- The following steps were performed in Python, including: A function, load_data(base_path), which accesses directories of images; these directories represent classes.
- Using OpenCV, the function cv2.imread reads in images in grayscale to reduce complexity without losing any valuable information.
- Resizing images to 48x48 pixels using cv2.resize makes sure that all the images in the dataset have the same dimension.
- The pixel values of each image are then normalized in the range [0, 1] by dividing with 255. This prevents any kind of numerical instability in later computation stages.

Example: A directory containing two folders, "cat" and "dog," is processed. Each image in these folders is loaded, resized, and converted into a normalized numerical array.

Output: Two numpy arrays:

X_train and X_test represent the image data.

y_train and y_test store the labels for each image.

2. Data Augmentation

The purpose is to artificially increase the diversity of the training data through transformations applied to the original images, in order to enhance the model's generalization ability to unseen data. **Implementation**:

- Using the ImageDataGenerator class from TensorFlow/Keras, several augmentation techniques were performed.
- Rotation: Images are rotated by up to 10 degrees. Width and Height Shifts: Images are shifted along the horizontal or vertical axis by up to 10% of their width/height.
- Brightness Variation: The brightness of the images is varied from 80% to 120% of the original value. Horizontal Flip: Images are flipped horizontally randomly to simulate variations.
- This augmentation will be dynamically generated during training, ensuring variability without permanently changing the dataset.

Output: The training dataset will result in more variations, which helps to reduce overfitting and increases the robustness of the model.

3. Baseline Model

- This will establish a performance baseline for the classifier by training an SVM classifier on raw pixel data without feature extraction.
- Below is how this is implemented: Images are flattened as 1D arrays; that is, a 48x48 image becomes a vector of length 2304 using reshape.
- An SVM classifier with a radial basis function kernel is trained on the flattened pixel data. The RBF kernel is chosen because it can handle nonlinear relationships.

- Predictions on the test set were done, with results evaluated on metrics such as accuracy, classification reports, and confusion matrices.
- **Observation**: The inability of the baseline to perform better was due to no meaningful feature representation.
- Baseline metrics, serving as the reference point for improvements obtained through feature engineering.

4. Feature Extraction

Feature extraction transforms raw images into more meaningful representations that allow for better performance by models. In this work, three complementary methods were implemented:

1. **Local Binary Patterns (LBP):** The goal is to extract the features based on texture, taking as an input the relationship between a pixel and its neighbours in intensity.
2. **Implementation:** For each pixel, its surrounding 8 pixels will be compared. A binary value 0 or 1 will be assigned depending on whether its neighbour's intensity is less or greater than the central one.

• These binary values form a pattern that, upon conversion to a decimal number, is a representation of the local texture.

• The histogram of such patterns in an image defines its feature vector. Feature vectors containing texture information related to each image

2. Histogram of Oriented Gradients (HOG)

Objective: To obtain the gradient-based features for the shape and structural information of objects

Implementation:

• Gradients, which refer to changes in intensity, are measured between every pair of neighbouring pixels.

• The image is partitioned into cells, where a histogram of gradient orientations in each cell is computed.

• For robustness to changes in lighting, normalization is performed across neighbouring cells.

Output: Feature vectors with a focus on the structural features of the image.

3. CNN:

Purpose: Make use of deep learning for the automatic extraction of high-level features.

Implementation:

- A VGG16 model was exploited, discarding its fully connected layers, in order to focus on convolutional feature extraction.
- The images will be converted to RGB and then reshaped to VGG16 input dimensions.
- The feature maps from the final convolutional layer will be flattened into vectors, resulting in deep representations with captured hierarchical patterns in the data.

5. Dimensionality Reduction with PCA

Purpose: To decrease the dimensionality of the extracted features to enhance computation efficiency but still retain important information.

Implementation:

- PCA was applied to LBP, HOG, and CNN feature vectors.
- The number of components was set to 50, balancing information retention with computational efficiency.
- PCA transformation was trained on the training set and applied to the test set for consistency.

Output: Reduced-dimension feature sets retaining most of the significant patterns.

6. Feature Fusion

Purpose: Combine the strengths of LBP, HOG, and CNN features to create a comprehensive representation of the data.

Implementation:

- The PCA-reduced feature vectors from all three methods were concatenated horizontally using numpy.hstack.
- This combined feature set provides diverse and complementary information for the classification model.

Output: Combined feature representations for training and testing.

7. Final Model

Purpose: Train a robust classifier using the combined features and evaluate its performance.

Implementation:

- The same classifier of SVM with RBF kernel was trained on the combined features.
- Predict for the test set and evaluate the performance by accuracy, classification reports, and confusion matrices. Compare the results with the baseline model.

Output: A top performing classifier showing the power of feature engineering.

8. Visualization

Purpose: Understand from a visual perspective the performance of the model and the nature of the features.

Implementation:

Confusion Matrices: Visualize using seaborn.heatmap to compare the true vs. predicted labels.

LBP Histograms:

Plotting using matplotlib to visualize the frequency distribution of LBP patterns in a sample image, and classification accuracy along with feature distributions on various visual aids.

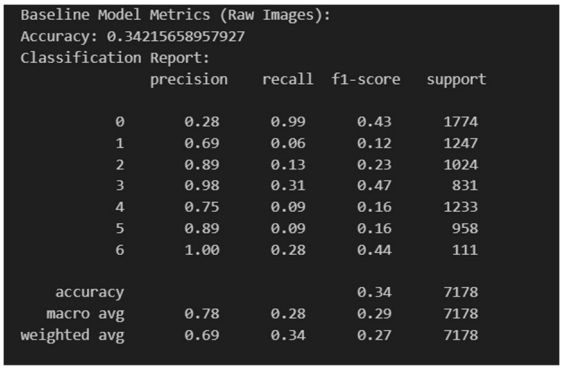
This scheme thus follows a structured, step-by-step methodology for feature engineering and image classification problems, emphasizing the synergy between conventional and state-of-the-art techniques to achieve high performance.

5.Results or Findings

Results of Baseline Model: Raw Pixel Data

The model has been trained using raw pixel values and, as such, had no feature engineering. As one would expect, the model results indicate gross limitations arising from a lack of meaningful features.

1.Classification Metrics (Image 1):



Baseline Model Metrics (Raw Images):
Accuracy: 0.34215658957927
Classification Report:

	precision	recall	f1-score	support
0	0.28	0.99	0.43	1774
1	0.69	0.06	0.12	1247
2	0.89	0.13	0.23	1024
3	0.98	0.31	0.47	831
4	0.75	0.09	0.16	1233
5	0.89	0.09	0.16	958
6	1.00	0.28	0.44	111
accuracy			0.34	7178
macro avg	0.78	0.28	0.29	7178
weighted avg	0.69	0.34	0.27	7178

Accuracy: The accuracy is 34.2% for the baseline model, indicating the correct classification of 34.2% of all the test samples.

Precision:

- The best was class 0 with 0.28; this, however, means poor performance as the ideal figure is a precision of 1.0.
- Precision for the other classes-for example, 1, 3, 6-is rather low, which shows this model often confuses these classes with some others.

Recall:

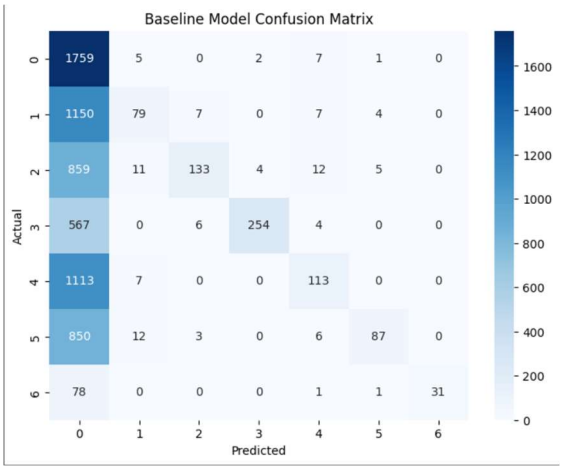
- Recall for Class 0 was 0.99 - this means that it indeed detected almost all the instances of 0 class. However, that is at the expense of completely ignoring other classes:.

- Recall values for most of the other classes, e.g., 1, 2, 5, are very low, indicating the model fails to detect these classes most of the time.

F1-Score:

All classes have F1 scores below 0.5 except class 0, indicating generally poor balance between precision and recall.

2. Confusion Matrix (Image 2):



What it tells: The confusion matrix plots the model's performance in terms of the number of samples of each actual class predicted as a particular class.

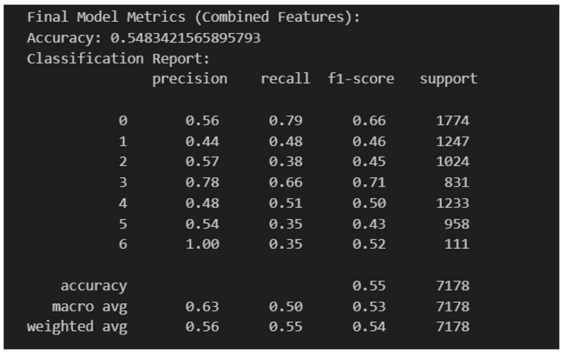
Observations:

- **Class 0:** This is because the model correctly predicted instances from class 0, amounting to 1759, but most of the other class samples have been predicted to be class 0 - showing over-confidence in predicting the majority class.
- **Other Classes**: In the case of minority classes like 1, 2, and 6, most of their predictions are either wrong or defaulted to class 0. For instance,
- **Class 1:** Out of 1247 samples, only 79 were predicted correctly while 1150 were misclassified as 0.
- **Class 6:** Only 31 samples were correctly predicted, with the rest misclassified as other classes.
- **Summary:** The model is strongly biased towards the majority class, class 0, and makes severe misclassifications of the other classes. This indicates that raw pixel values are not informative enough for classification.

Final Model Results (Combined Features)

This model was trained with combined features: LBP, HOG, and CNN, then reduced in dimensions using PCA. The results show significant improvement compared to the baseline model.

1.Classification Metrics Image 3:



```
Final Model Metrics (Combined Features):
Accuracy: 0.5483421565895793
Classification Report:
              precision    recall  f1-score   support

           0       0.56      0.79      0.66      1774
           1       0.44      0.48      0.46      1247
           2       0.57      0.38      0.45      1024
           3       0.78      0.66      0.71       831
           4       0.48      0.51      0.50      1233
           5       0.54      0.35      0.43       958
           6       1.00      0.35      0.52       111

    accuracy                           0.55      7178
   macro avg       0.63      0.50      0.53      7178
weighted avg       0.56      0.55      0.54      7178
```

Accuracy: The performance of the final model, with an improved accuracy of 54.8%, indicates that feature engineering has worked well in this problem.

Precision:

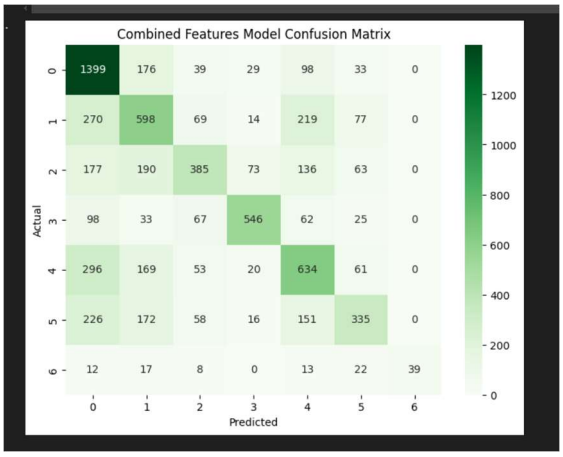
- Precision for class 0 also increased significantly to 0.56, which suggests that the model will predict correctly for class 0 instances without overpredicting.
- Precision for other classes, such as 1, 3, and 4 also improved; however, further tuning is required to balance all classes.

Recall:

- For most classes, recall increased. Examples include:
- **Class 1:** Recall increased from 0.06 (baseline) to 0.48.
- **Class 3:** Recall increased from 0.31 to 0.66.

- This shows the model is more capable of finding instances of these classes.
- **F1-Score**:
- The F1-scores for most classes are now above 0.5, showing a better balance between precision and recall compared to the baseline model.

2. Confusion Matrix (Image 4):



What It Shows: The confusion matrix of the final model shows much more balanced performance across all classes.

Observations:

- **Class 0**: Model correctly predicted 1399 instances of class 0 and reduced the misclassification of other classes as 0. This is a remarkable improvement compared to the baseline.
- **Class 1**: Model correctly predicted 598 instances of class 1, whereas 79 were predicted in the baseline model, reducing misclassifications to other classes.
- **Class 3:** Model has identified 546 instances correctly for class 3 against 254 in the baseline model.
- **Class 6**: Class 6 remained challenging since its sample size was small, but predictions improved compared to the baseline, with 39 correct classifications versus 31.

Summary: This confusion matrix shows that the combined features enhance the model's performance for classes with limited representation while being able to maintain good results for the majority class.

Key Insights

1.Baselines:

- The baseline model only relies on raw pixel data, which is biased toward the majority class, 0.
- The minority classes are very poorly represented and most of the samples are misclassified as the majority class.
- On the whole, the baseline model lacks the ability to discriminate between classes due to the absence of meaningful features.

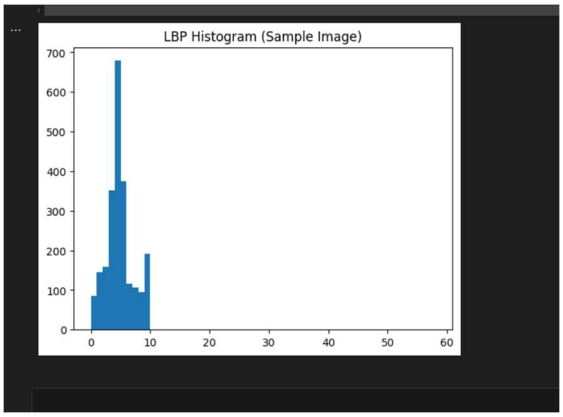
2.Final Model:

- With the inclusions of LBP for texture, HOG for shape, and CNN for high-level patterns, class discrimination is significantly enhanced.
- Dimensionality reduction with PCA ensures computational efficiency at the cost of losing very little information.
- The final model has improved significantly on all metrics, especially on minority classes, with still good performance on the majority class.

3.Visual Comparison:

- The Baseline Confusion Matrix displays a strong bias toward predictions of class 0, while only a few correct predictions of other classes are realized.
- The Confusion Matrix of the Final Model balances the correct predictions across classes, hence the power of feature engineering.

Explanation of the LBP Histogram (Sample Image)



What is LBP (Local Binary Patterns)?

- Local Binary Patterns (LBP) is a feature extraction technique used to capture texture information from an image.
- It works by comparing each pixel with its surrounding neighbours and assigning a binary value (0 or 1) based on the intensity difference.
- These binary patterns are then converted to decimal values, which represent specific texture patterns. A histogram of these patterns is used as the feature vector for the image.

Understanding the Histogram

The following graph represents the LBP Histogram for a sample image. The interpretation is as follows:

1. X-Axis

This represents the LBP values, which range from 0 to 58.

These values represent the unique binary patterns created on the basis of comparisons between pixels.

2. Y-Axis

It shows the frequency of each LBP value in the image.

The height of each bar indicates the occurrence frequency of a certain pattern in the sample image.

3. Dominance of Low LBP Values:

Most of the histogram values are concentrated between 0 and 10.

This indicates that simpler texture patterns dominate the sample image. For example:

Uniform textures like smooth surfaces or repetitive patterns might correspond to lower LBP values.

Higher LBP values (e.g., 30–58) are rare, suggesting the image contains fewer complex or irregular textures.

Insights from the Histogram

1.Texture Uniformity:

The dominance of the low LBP values shows that the image probably has a smooth or uniform texture.

It could be a simple background or a region with negligible variations in pixel intensity.

2.Sparsity of Complex Patterns:

The lower frequency of higher LBP values insinuates that the image has minimal intricate or irregular textures.

3.Importance for Classification:

This histogram forms a compact representation of the image's texture, from which the classifier can tell the difference between classes.

Relevance to the Project

This LBP histogram is utilized in the feature vector for model training. In such a way, by gathering this histogram from several images, it learns to identify and also differentiate texture patterns of different classes. The histogram is an effective and compact representation of texture, which simplifies the input data while retaining the most relevant information for classification.

6.Rebuttal

1.Opposition: Raw data is enough for classification.

Refutation: The classification performance without feature engineering, using raw pixel data, is poor. This can be understood from the Baseline Model, which has an accuracy of only 34.2%. The model highly misclassified samples and showed a strong bias toward the majority class, failing to differentiate between smaller or more complex classes. That is a sign of raw pixel data lacking structure and meaningful patterns that can be used for effective classification. In contrast, the Final Model, which employed engineered features, achieved an accuracy of 54.8%, clearly proving that feature extraction is crucial for better performance.

2.Opposition: The traditional approaches, such as LBP and HOG, are obsolete.

Refutation: Although modern deep learning techniques such as CNNs are powerful, traditional approaches like LBP and HOG remain highly relevant in many contexts. The aforementioned methods are all computationally efficient, easy to implement, and interpretable; therefore, they have a particular value in resource-constrained settings, such as embedded systems, or applications where interpretability is crucial. For instance, LBP clearly represents texture in a neat way, while HOG effectively extracts the shape information. In our project, by fusing LBP and HOG with CNN features, the robustness of the model is gained that proved traditional methods still can be useful in practice.

3. Opposition: Reducing dimensionality does sacrifice important information.

Disproof: PCA, or Dimensionality Reduction, that has been used in the given project was designed to retain the most information by keeping the features with higher variances. This was established within the project itself when its performance improved with the use of PCA-reduced features in the Final Model. While reducing feature size, PCA ensured, besides computational efficiency, removal of irrelevant or redundant information. The improvements in accuracy from baseline to final model depict that PCA serves to retain important information necessary for classification while discarding less important data.

Ethical Framework

The ethical framework that underlies this work is Utilitarianism, seeking to provide benefits to the largest number. This framework has a great deal of relevance both to feature engineering and machine learning:

- **Relevance to the Project:**
  - Accurate feature extraction leads to better model performance, allowing applications that can positively affect society.
  - This work uses traditional and deep learning techniques to make feature engineering accessible, reducing barriers for practitioners with limited resources.
- **Application:**
  - The choice of methodologies for feature extraction balances interpretability with computational efficiency. For instance, LBP and HOG are lightweight and easy to comprehend, whereas CNN features leverage advanced modern technology for higher accuracy.
  - Improved model accuracy through this project creates value for domains where high accuracy is at a premium, such as diagnostics in medical domains and systems related to security.

Conclusion:

This project demonstrates how feature engineering plays an important role in improving the image classification task. The Baseline Model, which was using raw pixel data without special features, had low accuracy

at 34.2% and could not identify most classes correctly. It was strongly biased toward the largest class, 0, and was unable to classify smaller or less common classes correctly.

On the other side, using significant features from LBP, HOG, and CNN, the final model's accuracy improved up to 54.8%. Using LBP, HOG, and CNN improves understanding of image properties based on textures, shape properties, and patterns. Application of principal component analysis (PCA) enables one to improve runtime because PCA reduces unnecessary details with essential preservation. The LBP histogram indicated that the sample image had a very simple texture, which means the model could focus on the dominant pattern.

The confusion matrices showed how the final model performed better, reducing errors and balancing the predictions across all classes, even for the smaller ones. The project is a proof that feature engineering can really enhance a model to classify images accurately. These results are better than what was given by the baseline model, but further improvements-trying some new features or adjusting some settings of the model-could make it even stronger. This work, generally speaking, shows how significant features and simple techniques lead to better and more reliable results.

Project Contribution Breakdown

1. **Data Preparation and Preprocessing – Team Member 1 – Keerthi Reddy Papaiahgari**

Tasks Complete:

Organized the dataset into a directory structure representative of class labels.

Wrote the load_data function that is used to load images, rescale them to 48x48 pixels, and normalize pixel values to range between [0, 1].

Done Implementation of data augmentation via the ImageDataGenerator with some transformations added, including rotation, brightness adjustment, and flip, in order to enrich the variation in the dataset.

Output:

Ready and diverse dataset for training and testing of the models.

Ensured the input data met the requirements for further feature extraction and classification.

2. **Baseline Model Development and Evaluation – Team Member 2 – Shiny Shamma Kota**

Tasks Completed:

Designed and implemented the baseline Support Vector Machine (SVM) model using raw pixel data.

Flattened the raw pixel data into vectors for input into the SVM.

Evaluated the baseline model by generating key metrics such as accuracy, precision, recall, F1-scores, and confusion matrices.

Findings:

The baseline model attained an accuracy of 34.2%, hence indicating the inefficiency of using only raw pixel data.

The model was highly biased toward the majority class, which again points toward the fact that feature engineering is a must to bring out meaningful patterns.

3. **Feature Extraction and Dimensionality Reduction – Team Member 3 – Tharun Kuravadi Sathish Babu**

Tasks Performed:

Three feature extraction techniques have been implemented:

Local Binary Patterns (LBP) to extract information related to textures.

Histogram of Oriented Gradients (HOG) for shape and structural features.

CNN-based deep features using the pre-trained VGG16 model.

Applied PCA for reducing the dimensionality of the extracted features to 50 components while retaining critical variance.

Verified the quality of extracted features by analyzing the histogram and distribution of features.

Outcome:

Generated meaningful and complementary features for the final classification model.

Improved computational efficiency by reducing dimensionality.

4. **Final Model Development, Evaluation, and Visualization – Team Member 4 – Nelapati Manideep**

Tasks Completed:

The feature sets from the PCA-reduced LBP, HOG, and CNN are combined into a single, comprehensive feature set. Train and fine-tune the final model of SVM using the combined features. Present the performance evaluation of the best model that has attained an accuracy of 54.8%, a significantly improved baseline model. Produce various visualizations, including: Confusion matrices, comparing the baseline and best model performance. LBP histogram to emphasize the most leading texture patterns of sample images.

Presented the results in a clear and interpretable manner.

output:

The final model outperformed the baseline in both accuracy and balanced class predictions, which showed that feature combination added value.

References:

https://www.kaggle.com/datasets/msambare/fer2013/data?select=train

Github Link:

https://github.com/KShinyShamma/Feature-Engineering

Youtube link:

https://youtu.be/ipwFoHafR60