Image Blur Detection Project Report

Student Information

- Name: Asala Abu Grara & Sara Abu Mandil
- Major: Data Science & Artificial Intelligence
- University: University College of Applied Sciences
- Course: Digital Image Processing

1. Introduction

Image quality assessment plays a critical role in many computer vision and image processing applications such as surveillance systems, medical imaging, autonomous driving, and photography enhancement. One of the most common image quality degradations is blur, which may result from camera motion, defocus, or object movement during image capture.

Automatically detecting whether an image is blurred or sharp is an important preprocessing step before applying higher-level vision tasks such as object recognition or feature extraction. Traditional approaches often rely on deep Convolutional Neural Networks (CNNs); however, these models are computationally expensive and require large amounts of labeled data.

In this project, a lightweight and efficient alternative approach is proposed. Instead of feeding raw pixel values into deep networks, meaningful handcrafted features that capture blur characteristics are extracted from images. These features are then used as input to a traditional feed-forward neural network (Multi-Layer Perceptron – MLP) to perform binary classification (Blurred vs. Sharp).

The main objective of this project is to design, implement, and evaluate a traditional neural network model for image blur detection and analyze its performance through experiments and feature importance analysis.

2. Theoretical Background

This section presents the theoretical concepts related to neural networks and their application in image classification.

2.1 Definition of Neural Networks

Neural Networks (NNs) are computational models inspired by the biological structure of the human brain. They consist of interconnected artificial neurons organized in layers. Each neuron processes input data by multiplying it with weights, adding a bias term, and applying a non-linear activation function.

Neural networks are capable of learning complex patterns from data and are widely used in classification, regression, and computer vision applications.

In this project, a feed-forward neural network called a Multi-Layer Perceptron (MLP) is used for binary image classification

2.2 Neural Network Architecture and Components

A typical feed-forward neural network consists of:

- Input Layer
- Hidden Layer(s)
- Output Layer

Each neuron performs the following computation:

1. Multiply inputs by weights
2. Add bias
3. Apply activation function

The learning process consists of:

- Forward Propagation: Passing input data through the network to compute predictions.
- Backpropagation: Updating weights by minimizing the loss function using gradient descent.

In this project, the architecture consists of:

- Input layer: Handcrafted feature vector
- One hidden layer with 50 neurons
- Output layer with one neuron for binary classification

2.3 Activation Functions

Activation functions introduce non-linearity into the network, enabling it to learn complex decision boundaries.

Two activation functions were used:

ReLU (Rectified Linear Unit)

ReLU(x) = $max(0, x) = f(x)$

ReLU is computationally efficient and helps reduce the vanishing gradient problem. It was used in the hidden layer.

Sigmoid Function

Sigmoid(x) = $\frac{1}{x^{-}e + 1} = \sigma(x)$

The sigmoid function outputs values between 0 and 1, making it suitable for binary classification problems. It was used in the output layer.

2.4 Loss Function

Since this is a binary classification task, Binary Cross-Entropy (BCE) loss was

used. The loss function is defined as:

Loss = $[(p - y)log(1 - 1) + ylog(p)] - = L$

Where:

- y is the true label (0 or 1)
- p is the predicted probability

Binary Cross-Entropy penalizes incorrect predictions more strongly when the model is confident, which improves training performance.

2.5 Optimization Algorithms

Neural networks are trained by minimizing the loss function using optimization algorithms.
Two common optimizers are:

Stochastic Gradient Descent (SGD)

SGD updates model parameters using gradients calculated from training samples. It is simple but may converge slowly.

Adam (Adaptive Moment Estimation)
Adam combines momentum and adaptive learning rates, allowing faster and more stable convergence.

In this project, the Adam optimizer was selected due to its efficiency and strong practical performance.

2.6 Blur in Spatial and Frequency Domains

Blur can be analyzed in both spatial and frequency domains.
In the spatial domain, blur reduces edge sharpness and decreases gradient intensity, which can be measured using operators such as Laplacian and Sobel. In the frequency domain, blur suppresses high-frequency components of the image, which correspond to edges and fine details. Sharp images contain strong high-frequency information, while blurred images show attenuation in these components.

In this project, spatial-domain features were primarily used for classification.

3. Methodology

This section describes the dataset, feature extraction process, model architecture, and training procedure used in this project.

3.1 Dataset Description

The dataset used in this project is the Blur Dataset obtained from Kaggle.

The dataset is organized into three main folders:

- sharp
- motion_blurred
- defocused_blurred

Each category contains 350 images, resulting in a total of 1050 images. For binary classification purposes, the labels were assigned as follows:

- sharp → label 0
- motion_blurred and defocused_blurred → label 1

Thus, the problem was formulated as a binary classification task (Sharp vs. Blurred).

The dataset was divided into:

• Training set: 840 images (80%)
• Testing set: 210 images (20%)

3.2 Preprocessing

Before feature extraction, the following preprocessing steps were applied:

• All images were converted to grayscale
• Images were processed using OpenCV
• Feature values were extracted and stored as numerical vectors

No deep learning preprocessing techniques were applied since raw pixel input was not used.

Feature scaling using StandardScaler was applied before training the MLP model to normalize feature distributions

3.3 Feature Extraction

The extracted features include:

1. **Laplacian Variance**
   Measures second-order intensity variation.

2. **Sobel X Variance (sobel_x_var)**
   Measures the variance of horizontal gradients. This feature is particularly effective for detecting motion blur in the horizontal direction.

3. **Sobel Magnitude Mean (sobel_mag_mean)**
   Represents the average edge strength in both directions.

4. **Tenengrad (tenengrad_mean)**
   Represents the mean squared gradient magnitude and measures overall edge energy.

3.4 Model Architecture

The classifier used in this project is a Multi-Layer Perceptron (MLP) with the following structure:

• Input layer: Feature vector (4 features initially, later reduced to 3 after feature selection) • Hidden layer: 50 neurons with ReLU activation
• Output layer: 1 neuron with Sigmoid activation

The output represents the probability of the image being blurred.

3.5 Training Process

The model was trained using:

• Optimizer: Adam
• Loss Function: Binary Cross-Entropy
• Maximum iterations: 2000
• Evaluation metric: Accuracy

Additionally,5-fold cross-validation was applied on the full dataset to assess generalization capability

Performance metrics used:

• Accuracy
• Precision
• Recall
• F1-score
• Confusion Matrix
• ROC Curve and AUC

The model was implemented using Python and Scikit-learn library

3.6 Model Deployment Preparation

To ensure reproducibility and future usability, the final trained model was saved using the Joblib library as **`blur_detection_mlp.pkl`**

In addition, the selected feature list used in the final reduced model was stored in a separate text file

This guarantees that any future inference process will use the same feature configuration as the trained mode

4. Experiments

This section presents the experiments conducted to evaluate the performance of the proposed MLP classifier and analyze the impact of different design choices.

4.1 Hidden Layer Size Experiment

To determine the optimal model configuration, different numbers of hidden neurons were tested.

The following configurations were evaluated:
• 10 neurons
• 20 neurons
• 50 neurons
• 100 neurons

The obtained test accuracies were:
- 10 neurons → 0.8905
- 20 neurons → 0.8905
- 50 neurons → 0.9095
- 100 neurons → 0.9048

The best performance was achieved using 50 hidden neurons.

Therefore, the model with 50 neurons was selected as the final architecture.

4.2 Cross-Validation

To evaluate the generalization capability of the model, 5-fold cross-validation was applied on the full dataset to evaluate generalization performance.

The cross-validation scores were:

0.9381, 0.9143, 0.9143, 0.9238, 0.919

The mean cross-validation accuracy was:

0.9219

The small variation between folds indicates stable and consistent model performance.

4.3 Feature Importance Analysis

Permutation feature importance was performed to analyze the contribution of each

feature. The results showed:

- sobel_x_var → highest importance
- tenengrad → second most important
- sobel_mag_mean → low contribution
- Laplacian → minimal contribution

This indicates that gradient-based features are strong indicators of image

sharpness. The feature importance ranking is illustrated in Figure 1

.

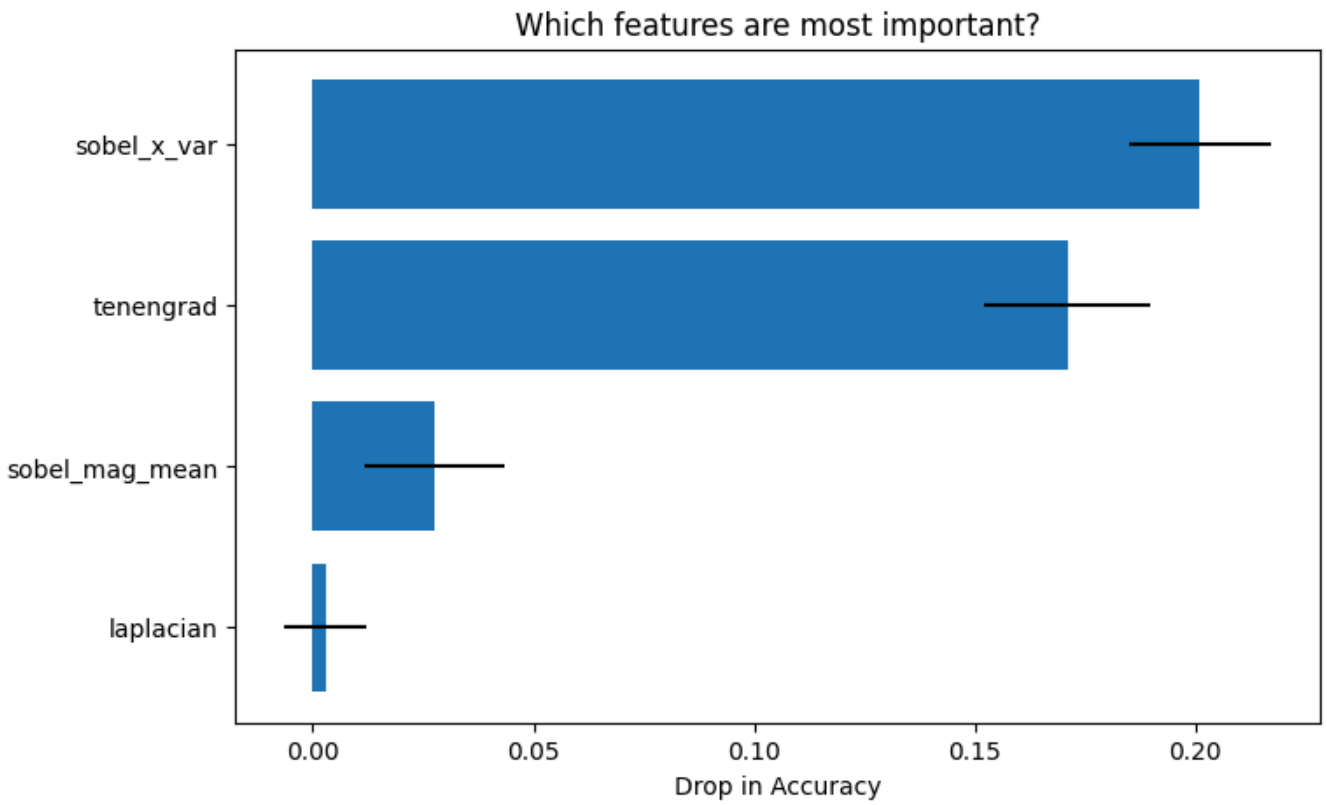

Figure 1 : Feature Importance

4.4 Feature Ablation Study

Based on the feature importance results, the Laplacian feature was removed and the model was retrained.

Before removal:
Accuracy = 0.9095

After removal:
Accuracy = 0.9095
The identical performance confirms that the Laplacian feature was redundant and did not significantly contribute to classification performance.

Removing it simplified the model without reducing accuracy.

5. Results

This section presents the final evaluation results of the proposed MLP classifier using the selected optimal configuration (50 hidden neurons).

5.1 Test Accuracy

The final model achieved:

Test Accuracy = 0.9095 (90.95%)

This indicates that approximately 91% of the test images were correctly classified as either sharp or blurred.

5.2 Classification Report

The classification performance was further evaluated using precision, recall, and

F1-score. Results summary:

Class 0 (Sharp):

- Precision = 0.92
- Recall = 0.80
- F1-score = 0.85

Class 1 (Blurred):

- Precision = 0.91
- Recall = 0.96
- F1-score = 0.93

The model demonstrates slightly higher recall for blurred images, meaning it is very effective at detecting blur.

The weighted average F1-score was approximately 0.91, indicating balanced overall performance.

5.3 Confusion Matrix

The confusion matrix obtained on the test set is shown in Figure

2. The matrix values are:

[[57, 13],
[6, 134]]

Where:

• 57 sharp images were correctly classified as sharp (True Negatives). • 134 blurred images were correctly classified as blurred (True Positives). • 13 sharp images were misclassified as blurred (False Positives). • 6 blurred images were misclassified as sharp (False Negatives).

The model performs particularly well in identifying blurred images, as indicated by the small number of false negatives.

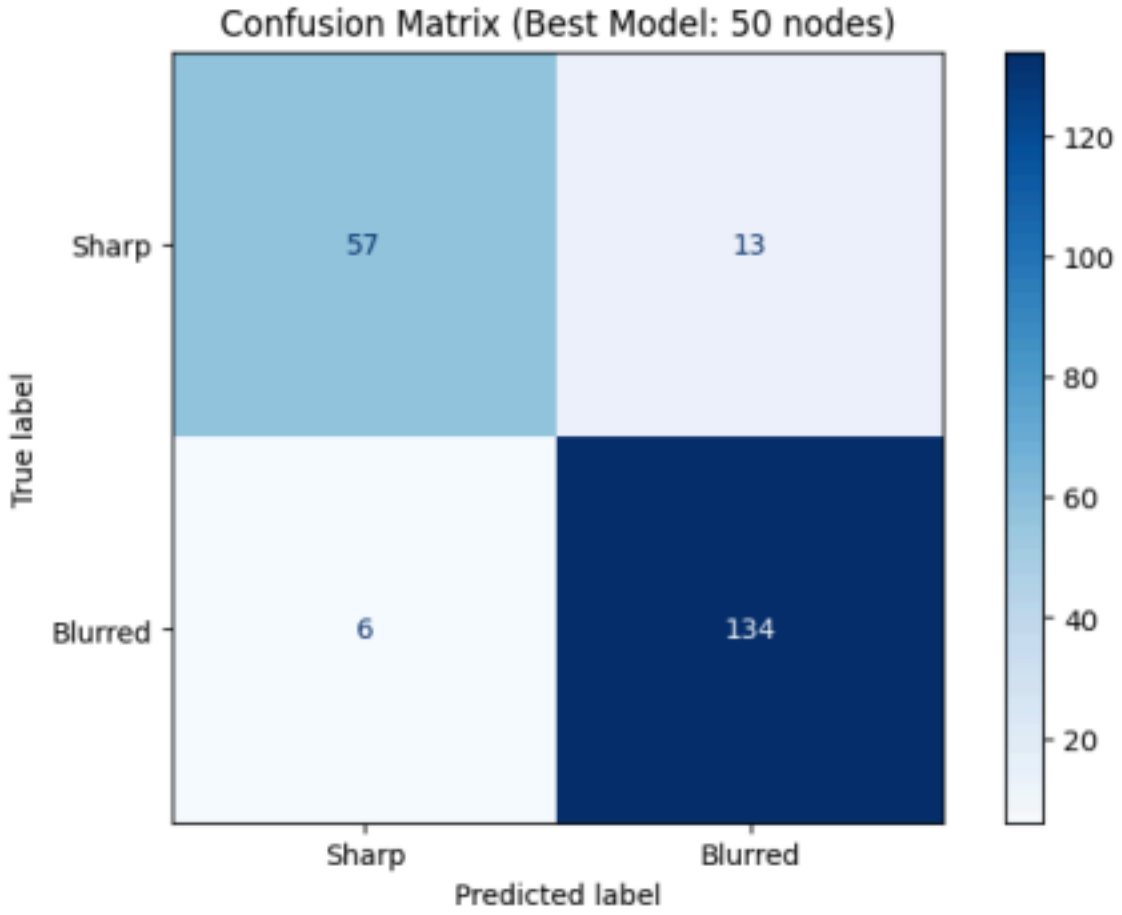

Figure 2 : Confusion Matrix

5.4 ROC Curve and AUC

To evaluate class separability, the Receiver Operating Characteristic (ROC) curve was plotted.

The ROC curve is illustrated in Figure 3.

The model achieved:

AUC ≈ 0.95

An AUC value close to 1 indicates excellent discrimination capability between sharp and blurred images. This confirms that the model is highly effective in distinguishing between the two classes across different classification thresholds.

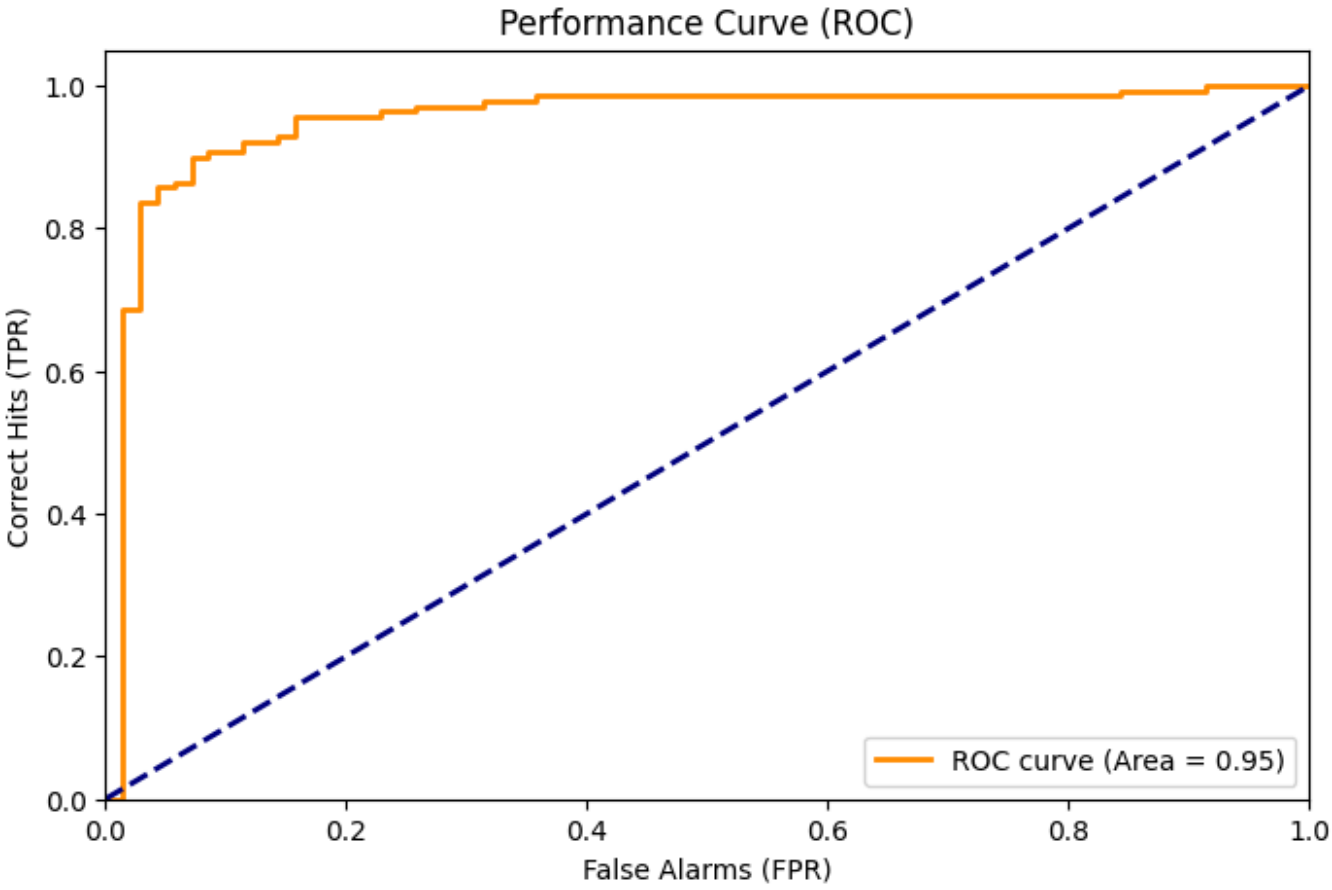

Figure 3: ROC Curve

6. Discussion

The experimental results demonstrate that handcrafted spatial features are highly effective for blur detection. Among the extracted features, Among the extracted features, Sobel X Variance and Tenengrad showed the highest contribution to classification performance to classification performance. This confirms that edge strength and gradient distribution are strong indicators of image sharpness.

The model achieved high recall for blurred images (0.96), which indicates that it rarely fails to detect blur. This is particularly important in real-world applications where detecting blur is critical before performing further image analysis.

The feature ablation study showed that removing the Laplacian feature did not reduce accuracy. This suggests that some features may contain redundant information, and proper feature selection can simplify the model without affecting performance.

Compared to deep Convolutional Neural Networks (CNNs), the proposed MLP-based approach requires significantly lower computational resources while still achieving strong classification results. This makes it suitable for real-time systems and low-resource environments.

Overall, the experiments confirm that traditional neural networks combined with well-designed handcrafted features can provide efficient and reliable blur detection.

No significant overfitting was observed, as the test accuracy (90.95%) is close to the cross-validation accuracy (92.19%)

7. Conclusion

In this project, a binary image classification system was designed to distinguish between blurred and sharp images using handcrafted features and a Multi-Layer Perceptron (MLP) classifier.

Instead of using computationally expensive deep learning models, meaningful spatial features were extracted to capture blur characteristics. The MLP classifier was trained and evaluated using multiple experiments, including hidden layer size comparison, cross-validation, feature importance analysis, and feature ablation.

The final model achieved:

- **Test Accuracy: 0.9095 (90.95%)**
- **Cross-Validation Accuracy: 0.9219**
- **AUC:≈ 0.95**
  - **Best Hidden Neurons: 50**

The results demonstrate that traditional neural networks can effectively solve blur detection problems when combined with appropriate feature engineering.

For future work, additional improvements may include:

- Exploring frequency-domain features
- Testing deeper neural network architectures
- Evaluating performance on larger and more diverse datasets

This project highlights the importance of combining theoretical understanding with practical experimentation in neural network design.