Image Blur Detection Project Report

Student Information

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

1. Introduction

Image blur detection is an important task in digital image processing and computer vision. Blurred images can negatively affect image analysis, object recognition, and visual quality. The goal of this project is to automatically detect whether an image is **sharp** or **blurred** using classical image processing features and a machine learning model.

2. Neural Networks Background

2.1 What are Neural Networks?

Neural Networks (NN) are a class of machine learning models inspired by the structure and functioning of the human brain. They consist of interconnected units called neurons that process information by applying mathematical operations to input data. Neural networks are widely used in tasks such as image processing, speech recognition, and pattern classification because of their ability to learn complex non-linear relationships from data.

In image processing applications, neural networks are especially effective because they can model subtle variations in pixel-based features and learn decision boundaries that are difficult to define using traditional rule-based methods

2.2 Neural Network Architecture

A neural network is composed of three main types of layers:

- **Input Layer:**
  Receives the input features. In this project, the input layer consists of four neurons corresponding to the extracted features: Laplacian variance, gradient variance, Sobel magnitude, and Tenengrad.

- **Hidden Layer(s):**
  Performs intermediate computations by applying weights, biases, and activation functions. Hidden layers allow the network to learn non-linear patterns. In this project, a single hidden layer with 20 neurons was used.

- **Output Layer:**
  Produces the final prediction. Since this is a binary classification problem (sharp vs blurred), the

output layer predicts whether an image belongs to class 0 (sharp) or class 1 (blurred).

Each connection between neurons has an associated weight, and each neuron has a bias term. These parameters are adjusted during training to minimize prediction error

2.3 Activation Functions

Activation functions introduce non-linearity into neural networks, allowing them to model complex relationships. Without activation functions, a neural network would behave like a linear model regardless of the number of layers.

Common activation functions include:

- **ReLU (Rectified Linear Unit):**
  Outputs zero for negative inputs and the input value for positive inputs. ReLU is computationally efficient and helps mitigate the vanishing gradient problem.

- **Sigmoid:**
  Maps input values to a range between 0 and 1. It is often used in binary classification problems but can suffer from vanishing gradients for deep networks.

In this project, the default ReLU activation function was used in the hidden layer of the MLP classifier due to its efficiency and good empirical performance.

2.4 Optimization Algorithms

Optimization algorithms are used to update the weights and biases of a neural network during training in order to minimize the loss function.

Two common optimization algorithms are:

- **Stochastic Gradient Descent (SGD):**
  Updates model parameters using a small batch of training samples. While simple and effective, SGD can be slow to converge and sensitive to learning rate selection.

- **Adam (Adaptive Moment Estimation):**
  Combines the advantages of momentum and adaptive learning rates. Adam generally converges faster and is more robust to noisy gradients.

Although the MLP classifier used in this project relies on an internal optimization process, understanding these algorithms is essential for neural network training and performance tuning.

2.5 Loss Functions

A loss function measures how far the model's predictions are from the true labels. During training, the neural network aims to minimize this loss.

For classification problems, common loss functions include:

- **Binary Cross-Entropy Loss:**
  Measures the difference between predicted probabilities and actual class labels. It is widely used for binary classification tasks.

In this project, the loss function guides the learning process by penalizing incorrect predictions and

helping the model adjust its parameters to improve classification accuracy.

2.6 Neural Networks in This Project

In this project, a **Multilayer Perceptron (MLP)** was selected as the neural network model. MLP is a feedforward neural network that consists of fully connected layers and is suitable for structured numerical features.

The combination of handcrafted blur features and an MLP classifier allowed the model to effectively distinguish between sharp and blurred images while maintaining computational efficiency and interpretability.

3. Dataset Description

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

Dataset structure:

- sharp
- motion_blurred
- defocused_blurred

Each category contains **350 images**, resulting in a total of **1050 images**.

For classification purposes:

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

4.Preprocessing

- Images were loaded from the dataset directory.
- All images were converted to **grayscale**.
- No resizing was required since the dataset images were already standardized.

5. Feature Extraction

Four handcrafted features were extracted from each image:

1. **Laplacian Variance**

Measures image sharpness by detecting high-frequency edges.

2. **Gradient Variance**
   Measures the variance of gradient magnitudes in the image.

3. **Sobel Magnitude**
   Computes the average edge strength using Sobel operators.

4. **Tenengrad**
   Measures image focus by summing squared Sobel gradients.

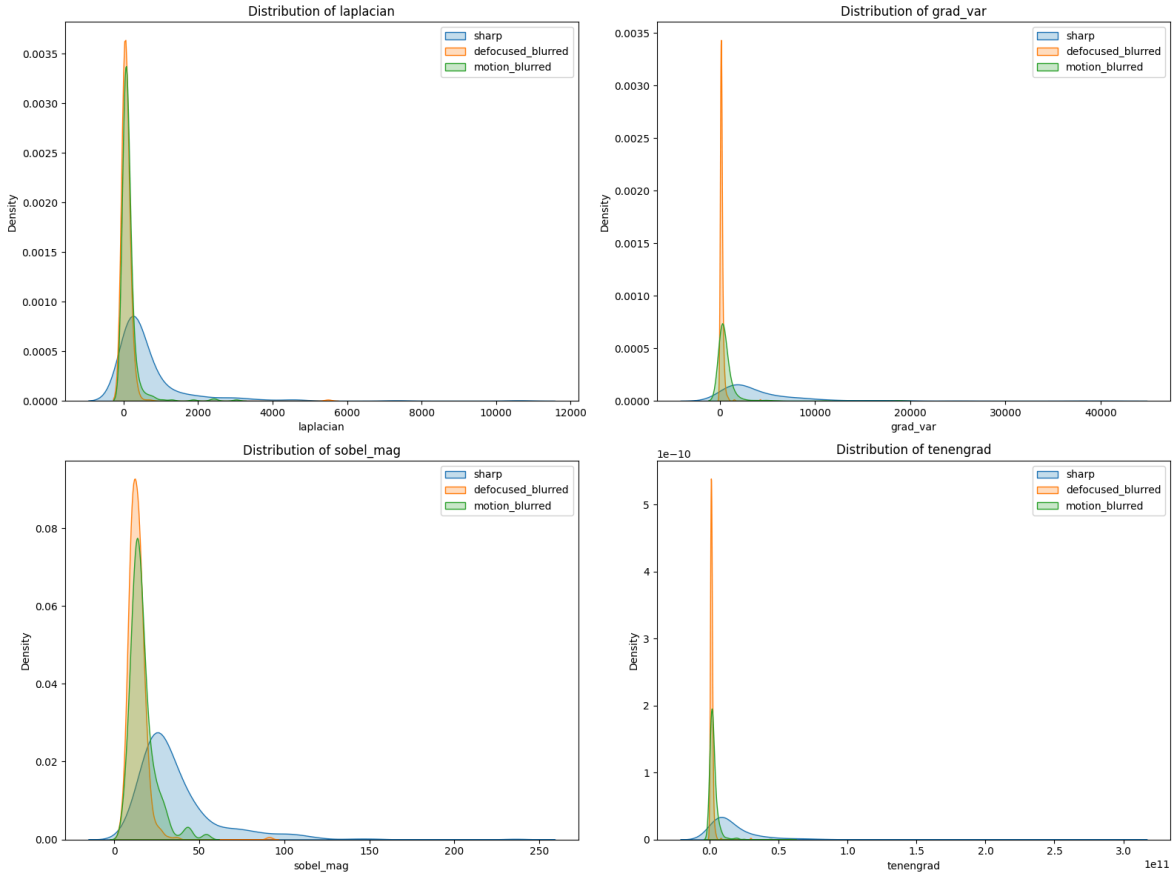
These features were stored in a DataFrame and later saved as a CSV file: **blur_features_results.csv**

6.Feature Analysis

Statistical analysis showed clear differences between sharp and blurred images:

- Sharp images had higher edge-related feature values.
- Defocused blur significantly reduced high-frequency information.
- Motion blur showed intermediate behavior.

Feature distribution plots confirmed that **Laplacian variance** and **Tenengrad** are highly discriminative features.



7. Classification Model

A **Multilayer Perceptron (MLP)** classifier was used.

Model configuration:

- Hidden layer: 20 neurons
- Activation function: default (ReLU)
- Max iterations: 500

Preprocessing:

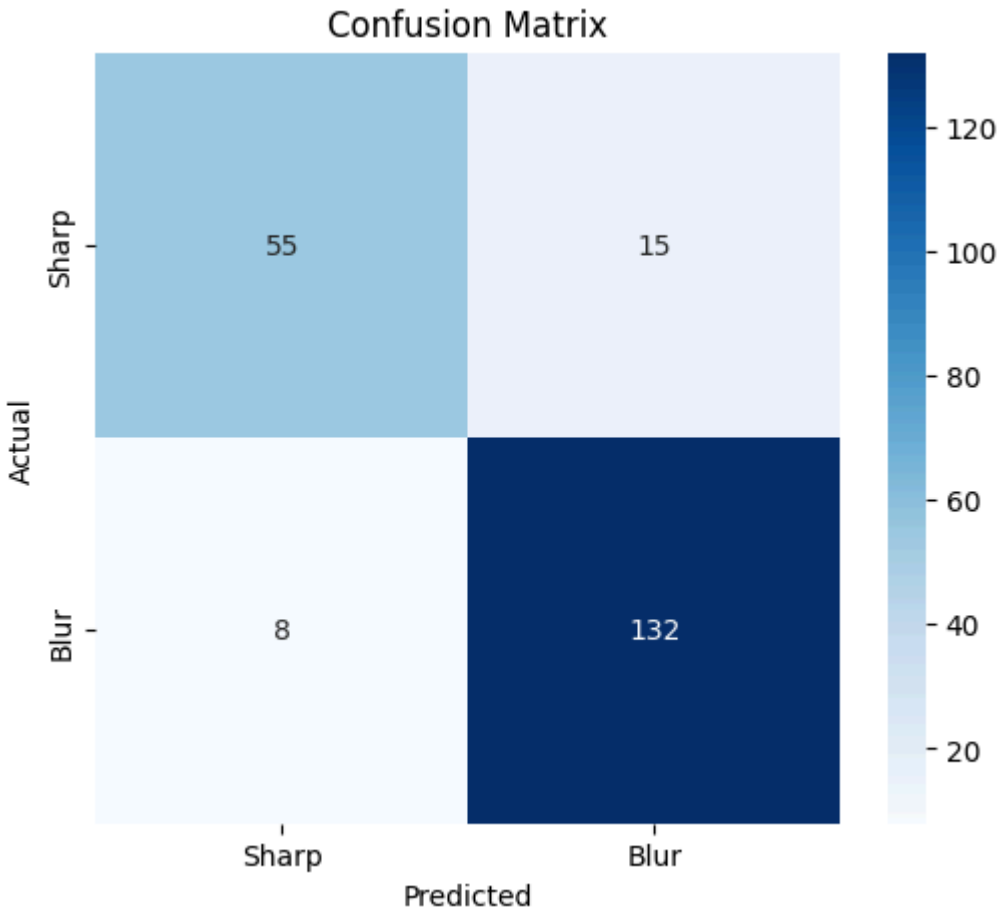
- Features were standardized using StandardScaler .
- Data split: 80% training, 20% testing.

8. Results

Overall Performance:

- **Accuracy:** 89%

Confusion Matrix:



- Sharp images: Some misclassified as blurred
- Blurred images: Detected with high accuracy

Classification Report:

- Precision (Sharp): 0.87
- Recall (Sharp): 0.79

2

- Recall (Blur): 0.94
- Precision (Blur): 0.90

Per-Blur-Type Evaluation:

- **Sharp:** Accuracy ≈ 80.9%
- **Motion Blur:** Accuracy ≈ 88.3%
- **Defocused Blur:** Accuracy ≈ 99.4%

9. Discussion

The model performed very well in detecting defocused blur, as it strongly reduces edge information. Motion blur was also detected effectively. Some sharp images were misclassified due to low contrast or lack of texture.

Handcrafted features proved effective, especially Tenengrad and Laplacian variance.

Comparison Between Laplacian Variance and Tenengrad

Both **Laplacian Variance** and **Tenengrad** are edge-based focus measures, but they capture image sharpness in slightly different ways:

- **Laplacian Variance** focuses on second-order intensity changes. It is very sensitive to fine details and high-frequency edges, which makes it effective for detecting defocused blur. However, it can be affected by image noise.

- **Tenengrad** relies on first-order gradients computed using Sobel operators. It measures the overall strength of edges in the image and is more stable in the presence of noise. Tenengrad performed consistently well across sharp, motion-blurred, and defocused images.

In this project, Tenengrad showed better robustness, while Laplacian Variance provided strong discrimination for defocused blur. Combining both features improved the overall classification performance.

Lessons Learned

- Classical image processing features can still achieve strong performance when carefully selected.
- Feature scaling is crucial for neural network models such as MLP.
- Defocused blur is easier to detect than motion blur due to the loss of high-frequency information.
- Simpler models with good features can outperform complex models on well-structured datasets.

These lessons highlight the importance of understanding both the data and the chosen features before moving to more complex deep learning approaches.

Why MLP Was Chosen?

The Multilayer Perceptron (MLP) classifier was chosen for this project due to its ability to model non

linear relationships between input features.

The extracted blur features are numerical and relatively low-dimensional, making MLP a suitable and efficient choice. Compared to more complex deep learning models, MLP requires less computational power and is easier to train and interpret, which makes it ideal for academic projects.

Additionally, MLP demonstrated strong performance when combined with feature standardization, achieving high accuracy

10. Limitations and Future Work

- Handcrafted features may fail in complex real-world images.
- Future work could involve:
- Convolutional Neural Networks (CNNs)
- Larger and more diverse datasets
- Additional blur-related features

11. Conclusion

This project demonstrated that classical image processing techniques combined with machine learning can effectively detect image blur. The achieved results confirm that handcrafted features remain a strong baseline solution for blur detection tasks.