EE678: Wavelets

INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

ELECTRICAL ENGINEERING

End-Semester Examination

Group 10

Authors :
Prajwal Nayak (22B4246)
Sarvadnya Purkar (22B4232)

Instructor :
Prof. Vikram Gadre

Enhancing Image Reconstruction Using Wavelet-CNNs with Self-Supervised MoCo Pretraining for Feature Extraction

CONTENTS

List of Figures

Enhancing Image Reconstruction Using Wavelet-CNNs with Self-Supervised MoCo Pretraining for Feature Extraction

***Abstract*—In this project, we begin by giving fingerprint images as input into a Multiwavelet Convolutional Neural Network (MWCNN), which reduces noise and enhances image quality, producing cleaner and more detailed representations of fingerprint features. The MWCNN leverages wavelet transforms to break down the image into multi-scale and multi-frequency components, effectively preserving fine textures and essential minutiae while removing noise and artifacts.**

These enhanced images are then fed into a ResNet model, which has been self-supervised using Momentum Contrast Version 2 (MoCoV2). MoCoV2 trains the ResNet by contrasting augmented pairs of the same image (positive pairs) against different images (negative pairs), allowing the model to learn robust, invariant features. The output of the ResNet is a condensed representation that captures the fingerprint's important features in a compact form, resilient to variations such as rotation, scaling, and noise. This feature embedding provides a rich yet compact summary of the fingerprint's unique details, making it well-suited for identification and matching tasks.

I. Introduction

In this report, we investigate the Multi-Wavelet Convolutional Neural Network (MWCNN), a cutting-edge approach to image denoising and restoration that uses multi-wavelet transforms. This model addresses the challenge of achieving high denoising performance while maintaining computational efficiency, which is crucial for various applications, including biometrics. Effective image denoising is vital for enhancing the quality of biometric images such as fingerprints, and iris scans, where noise reduction significantly affects system accuracy and reliability. The MWCNN improves feature extraction and reduces processing time by leveraging short-term residual learning and a hierarchical structure. This report examines the MWCNN's architecture and performance, and its findings are particularly relevant to our semester project focused on improving biometric systems through advanced image processing techniques.

II. Connection Between Wavelets/Filter Banks and CNN/ML/DL Structures

In this paper, discrete wavelet transform,

A. Overview of Wavelets and Filter Banks

1) Wavelets and Scaling Functions: Wavelets are mathematical functions used to decompose a signal into different frequency components, capturing both high-frequency details and low-frequency approximations. The key components are:

- **Wavelet function** $\psi(t)$: Captures high-frequency components of a signal.
- **Scaling function** $\phi(t)$: Captures low-frequency components of a signal.

The scaled and translated versions of these functions are:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$$

$$\phi_{j,k}(t) = 2^{j/2}\phi(2^j t - k)$$

where j is the scale and k is the translation (j and k are integers).

2) Multi-Resolution Analysis (MRA): MRA involves decomposing a signal into progressively finer approximations and details. The decomposition is expressed as:

$$x(t) = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} c_{j,k}\varphi_{j,k}(t) + \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} d_{j,k}\psi_{j,k}(t)$$

where $c_{j,k}$ are the approximation coefficients and $d_{j,k}$ are the detail coefficients at scale j.

3) Discrete Wavelet Transform (DWT): The DWT decomposes a signal into sub-bands at different scales. In 2D image processing, it decomposes the image into approximation and detail components.

4) 1D DWT: For a signal $x[n]$, the DWT coefficients are calculated by:

$$C_j[k] = \sum_n x[n] \cdot \phi_{j,k}(n)$$

$$D_j[k] = \sum_n x[n] \cdot \psi_{j,k}(n)$$

where $C_j[k]$ and $D_j[k]$ are the approximation and detail coefficients.

5) 1D DWT Filter Derivation: Let us consider the Haar Wavelet as it is the simplest to analyse. The Haar wavelet is one of the simplest wavelets used in the Discrete Wavelet Transform (DWT). Here's a step-by-step mathematical process to derive the filters $g[n]$ (low-pass filter) and $h[n]$ (high-pass filter) from the Haar wavelet function.

Haar Wavelet Functions

Scaling Function $\phi(t)$:

$$\phi(t) = \begin{cases} 1 & \text{if } 0 \le t < 1 \\ 0 & \text{otherwise} \end{cases}$$

Wavelet Function $\psi(t)$:

$$\psi(t) = \begin{cases} 1 & \text{if } 0 \leq t < 0.5 \\ -1 & \text{if } 0.5 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}$$

Low-Pass Filter $g[n]$:

Sampling the Scaling Function:

The low-pass filter $g[n]$ is obtained by sampling the scaling function $\phi(t)$ at discrete intervals and normalizing. The Haar scaling function $\phi(t)$ is 1 in the interval $[0, 1)$ and 0 elsewhere. To get the filter coefficients, we sample this function at $t = \frac{n}{2}$, where n is the index in the discrete domain.

$$g[n] = \frac{1}{\sqrt{2}}\phi\left(\frac{n}{2}\right)$$

For $n = 0$:

$$\phi(0) = 1$$

$$g[0] = \frac{1}{\sqrt{2}}\phi(0) = \frac{1}{\sqrt{2}}$$

For $n = 1$:

$$\phi(0.5) = 1$$

$$g[1] = \frac{1}{\sqrt{2}}\phi(0.5) = \frac{1}{\sqrt{2}}$$

For other values of n:

$$\phi\left(\frac{n}{2}\right) = 0$$

$$g[n] = 0 \text{ for } n > 1$$

Therefore, the low-pass filter coefficients are:

$$g[n] = \frac{1}{\sqrt{2}}[1, 1]$$

High-Pass Filter $h[n]$:

Sampling the Wavelet Function: The high-pass filter $h[n]$ is obtained by sampling the wavelet function $\psi(t)$ at discrete intervals and normalizing.

The Haar wavelet function $\psi(t)$ is 1 in $[0, 0.5)$, -1 in $[0.5, 1)$, and 0 elsewhere.

To get the filter coefficients, we sample this function at $t = \frac{n}{2}$, where n is the index in the discrete domain.

$$h[n] = \frac{1}{\sqrt{2}}\psi\left(\frac{n}{2}\right)$$

For $n = 0$:

$$\psi(0) = 1$$

$$h[0] = \frac{1}{\sqrt{2}}\psi(0) = \frac{1}{\sqrt{2}}$$

For $n = 1$:

$$\psi(0.5) = -1$$

$$h[1] = \frac{1}{\sqrt{2}}\psi(0.5) = -\frac{1}{\sqrt{2}}$$

For other values of n:

$$\psi\left(\frac{n}{2}\right) = 0$$

$$h[n] = 0 \text{ for } n > 1$$

Therefore, the high-pass filter coefficients are:

$$h[n] = \frac{1}{\sqrt{2}}[1, -1]$$

For a general wavelet function $\psi(t)$ and scaling function $\phi(t)$, the filters $g[n]$ and $h[n]$ are derived as follows:

Low-Pass Filter $g[n]$:

$$g[n] = \frac{1}{\sqrt{2}}\phi\left(\frac{n}{2}\right)$$

High-Pass Filter $h[n]$:

$$h[n] = \frac{1}{\sqrt{2}}\psi\left(\frac{n}{2}\right)$$

Here:

- $\frac{n}{2}$ adjusts the scale of the continuous functions to match the discrete domain. - $\frac{1}{\sqrt{2}}$ is a normalization factor to ensure that the filters have the correct energy distribution. By following these steps, we can derive the discrete filters from any continuous wavelet and scaling functions. These filters can be convolved with the functions to get a signal which has either the low frequencies or high frequencies depending on the filter used. We now use a similar technique to obtain filters for 2D DWT images as mentioned in the next topic.

6) 2D DWT for Images: For an image $x[m, n]$, the DWT results in four sub-images:

$$x_A[m, n] = \text{Conv}(x[m, n], f_A)$$

$$x_H[m, n] = \text{Conv}(x[m, n], f_H)$$

$$x_V[m, n] = \text{Conv}(x[m, n], f_V)$$

$$x_D[m, n] = \text{Conv}(x[m, n], f_D)$$

where f_A, f_H, f_V, and f_D are filters for approximation, horizontal, vertical, and diagonal details, respectively. x_A, x_H, x_V, and x_D are low frequency components, high-frequncy changes along the horizontal direction, high frequency components along the vertical direction and hig frequncy components along both directions .The filters used for DWT are:

$$f_A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad f_H = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix},$$

$$f_V = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix} \quad f_D = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

The DWT involves downsampling after convolution, effectively reducing the size of the sub-images. For a given image of size $M \times N$, the resulting sub-images after DWT will have a size of $\frac{M}{2} \times \frac{N}{2}$.

This downsampling can be thought of as taking every second pixel after convolution, which is similar to the "stride" of 2 used in CNNs.

7) Inverse Discrete Wavelet Transform (iDWT): The iDWT reconstructs the original image from the sub-band images using:

$$\tilde{x} = \sum_{i=A,H,V,D} \text{iConv}(x_i, f_i/4)$$

where x_i are the sub-band images and f_i are the filters, with $1/4$ normalizing the result.

During DWT, the image was downsampled, meaning the size of the sub-band images is smaller than the original image. iConv involves "upsampling" the sub-band images by reversing this downsampling.

If we just perform convolution using the filters and then perform inverse convolution, we get the same image without any loss of information.

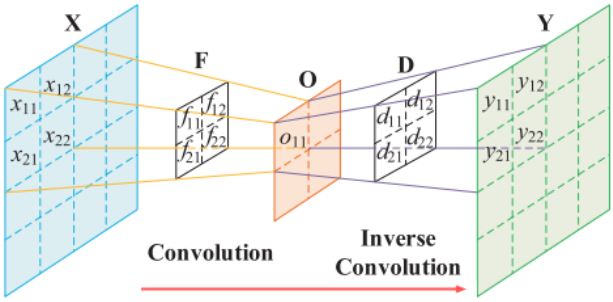

Fig. 1: Schematic diagram of the convolution and inverse convolution processes.
O_i = Conv($\mathbf{X}$,f_i) and $\mathbf{Y}$ = Sum (iConv(O_i,$\frac{f_i}{4}$)) [1]

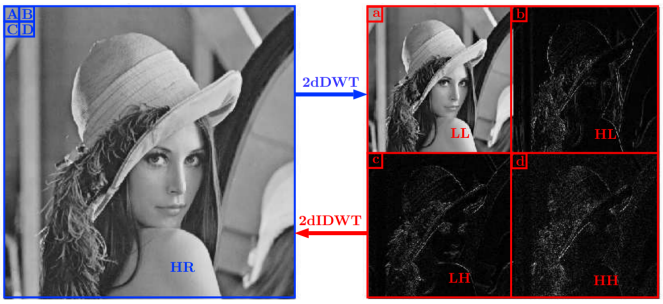

Fig. 2: The 2dDWT and 2dIDWT. A, B, C, D are four example pixels located in a 2×2 grid at the top left corner of HR image. a, b, c, d are four pixels from the top left corner of four sub-bands correspondingly [2]

B. Network Structures

The main deep learning structures used in the overall network structure are Convolutional Neural Network. The network structure we are using is MWCNN is three-layered and uses DWT and Convolutional layers. We have already seen DWTs above, now let us see the convolutional layers.

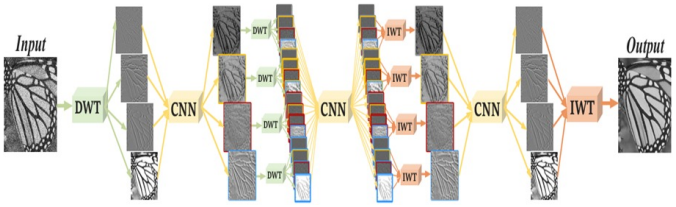

Fig. 3: Struture of Multi-Wavelet Convolutional Neural Network

1) Convolutional Layer: Let I be the grayscale image of size $H \times W$ with 1 channel. After applying DWT, we obtain 4 components: x_A, x_H, x_V, and x_D, each of size $H \times W$.

The convolutional layer has 40 filters, each of size $4 \times 3 \times 3$. Each filter operates across all 4 input channels.

The convolution operation for each filter can be represented as:

$$F_{ij} = \sum_{c=1}^{4} \sum_{m=-1}^{1} \sum_{n=-1}^{1} K_{c,m,n} \cdot x_{c,i+m,j+n}$$

where $K_{c,m,n}$ is the filter weight, $x_{c,i+m,j+n}$ is the input at position $(i+m, j+n)$ in channel c, and F_{ij} is the output feature map at position (i, j).

The output of this layer is 160 feature maps, each with non reduced spatial dimensions because of appropriate padding used.

The final output consists of 160 feature maps which capture different details of the input image but now at different levels due to the wavelet transform, each with non reduced spatial dimensions.

2) ReLU Activation Function: The ReLU (Rectified Linear Unit) activation function is applied after each convolutional layer. It is defined as:

$$\text{ReLU}(x) = \max(0, x)$$

This means for each value in the feature map, if x is negative, it is replaced with 0; otherwise, it remains unchanged. This introduces non-linearity into the model and helps in learning complex patterns.

C. The Connection

Wavelets and CNNs: Improving Results through Structural Sparsity

1. Wavelets and Structural Details: In image processing, wavelets, such as the Haar wavelet, decompose an image into four sub-bands: a low-frequency approximation and three high-frequency detail sub-bands. Each sub-band provides different types of information:

- **Low-frequency sub-band**: Captures the general structure or smooth regions of the image.
- **High-frequency sub-bands**: Capture edges, textures, and fine details (horizontal, vertical, and diagonal components).

Overlaying these sub-bands allows CNNs trained in the wavelet domain to extract structural relationships across spatial locations. By learning these sub-bands, the CNN can capture finer details and more local features compared to pixel-based representations. This improves the network's ability to retain and enhance intricate structures.

2. Sparsity in Wavelet Coefficients: Wavelet transforms produce sparse representations, meaning most wavelet coefficients are close to zero, while only a few retain significant values. Sparse activations benefit CNNs in two main ways:

- **Reduced Redundancy**: The sparsity ensures that the network focuses on important features and filters out less relevant information.
- **Efficient Learning of Fine Structures**: Sparse wavelet coefficients simplify the learning process by focusing the network's capacity on key structural details.

Mathematically, for a signal $f(x, y)$, applying a Discrete Wavelet Transform (DWT) decomposes it into a set of coefficients:

$$f(x, y) \xrightarrow{DWT} \{A(x, y), H(x, y), V(x, y), D(x, y)\}$$

where A represents the approximation (low-frequency content), and H, V, D represent the horizontal, vertical, and diagonal details (high-frequency content).

Instead of learning the dense pixel map of an image, CNNs trained in the wavelet domain learn these sparse coefficient maps:

$$f(x, y) \approx \sum_i c_i \phi_i(x, y)$$

where c_i are the sparse coefficients and ϕ_i are the wavelet basis functions. By focusing on the sparse coefficients, the network efficiently models complex patterns and textures with fewer parameters, leading to better feature extraction and preservation of fine details.

3. Structural Correlations and Learning: The DWT breaks an image into directional components (horizontal, vertical, and diagonal), introducing correlations between these sub-bands. A CNN trained in the wavelet domain can capture these correlations effectively:

- **Enhanced Edge Detection**: The sub-bands highlight edges and textures in different orientations, strengthening the network's ability to detect and enhance structural details like edges.
- **Improved Detail Emphasis**: The network processes wavelet coefficients, enabling it to emphasize fine details, textures, and small-scale structures that are difficult to capture with pixel values alone.

4. Increased Structural Similarity (SSIM): By incorporating wavelet-based structural details, CNNs better preserve the high-level structural integrity of an image, leading to higher Structural Similarity Index (SSIM) scores. SSIM measures the similarity between two images based on luminance, contrast, and structure. Since wavelets decompose images into sub-bands representing structural details, networks trained in the wavelet domain prioritize the preservation of structural information. This results in improved SSIM performance compared to pixel-based CNNs.

Conclusion: Wavelet transforms provide CNNs with sparse, multi-scale, and structurally rich features, allowing them to focus on critical details and structural correlations within images. This results in enhanced image reconstruction, sharper details, and improved preservation of the original image's structural integrity, as reflected by improved metrics such as PSNR and SSIM in super-resolution tasks.

D. Enhancements to Image Quality and Structure

Wavelets, particularly the **Daubechies wavelet**, improve the network's ability to capture fine structural details:

- Each sub-band (low-frequency and detailed sub-bands) captures different levels of information:
  - **Low-frequency sub-band** provides general image structure.
  - **High-frequency sub-bands** capture vertical, horizontal, and diagonal details, enhancing edges and fine features.
- By integrating these sub-bands, MWCNNs enhance structural correlation, leading to:
  - **Sharper images** with fewer artifacts.
  - Improved image resolution and detail preservation.

Methods	Set5	Set12	Set14	BSD68	Urban100
SRCNN [19]	0.0011	0.0011	0.0011	0.0011	0.0211
IRCNN [20]	0.0017	0.0017	0.0018	0.0017	0.0276
DnCNN [21]	0.0041	0.0041	0.0042	0.0041	0.0430
VDSR [22]	0.0056	0.0053	0.0055	0.0056	0.0488
RED30 [17]	0.0360	0.0345	0.0652	0.0364	0.2146
MWCNN [10]	0.0240	0.0235	0.0394	0.0227	0.1248
RDN [23]	0.0875	0.0827	0.1746	0.0868	0.8788
MemNet [24]	0.2127	0.2032	0.3886	0.2409	1.4648
MWRDCNN	0.0649	0.0447	0.0993	0.0478	0.3191

Fig. 4: Time Comparison with other models

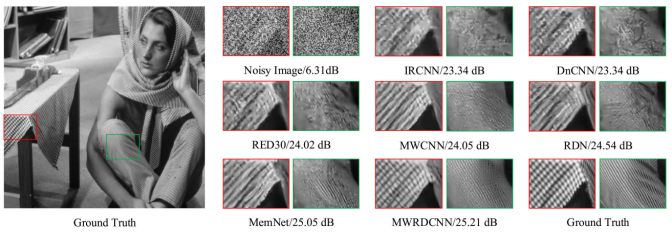

Fig. 5: Visual Results

As we see from the table and the figure that MWCNN doesn't take exceptionally lot of time to de-noise and and take as much time as any other model but with much better results than the others, in fact the closest to ground truth as compared to any other model.

Overall, MWCNNs leverage wavelet-based features to achieve efficient, interpretable, and high-quality results in image processing tasks such as **image denoising**.

III. Application of MWCNNs in Biometric Data Analysis

The application of Multi-Wavelet Convolutional Neural Networks (MWCNNs) in biometric data analysis offers several advantages due to their inherent benefits in explainability, interpretability, and economy. Here's how MWCNNs, leveraging wavelets, can contribute to your course project on biometric data analysis:

A. Enhanced Feature Extraction

Wavelet Transform Advantages:

- **Multi-Scale Analysis**: Wavelets provide a multi-resolution analysis of biometric data, such as fingerprint, iris, or facial features. This capability helps in capturing both coarse and fine details, which are crucial for accurate biometric recognition.
- **Sparse Representations**: Wavelet-based features often lead to sparse representations of biometric data. This sparsity helps in distinguishing subtle differences between biometric traits, which can be crucial for accurate identification and verification.

Application:

- Improved extraction of distinctive features from biometric data, enhancing the overall performance of recognition systems.

B. Robustness and Accuracy

- **Detail Preservation**: Wavelets ensure that fine details are preserved, which is important for high-precision biometric systems where small differences can be significant.

C. Practical Implementation

Integration:

- **Preprocessing**: Wavelets can be used for preprocessing biometric data to enhance feature extraction and reduce noise.
- **Model Training**: MWCNNs can be trained on biometric datasets to learn detailed and robust representations of biometric features, leveraging wavelet-based multi-resolution analysis.
- **Evaluation**: The interpretability of MWCNNs helps in evaluating and refining the biometric recognition models, ensuring they meet the required standards for performance and fairness.

Application:

- Effective integration of wavelet-based MWCNNs into biometric systems can lead to improvements in both accuracy and efficiency, making them suitable for practical applications in security, healthcare, and other fields requiring biometric analysis.

D. Results of running the code:

TABLE I: Results of Denoising on random fingerprints

Image Name	PSNR	SSIM	MSE	RMSE
s105_1.jpg	29.49	0.9013	73.14	8.55
s105_4.jpg	29.79	0.9268	68.29	8.26
s121_1.jpg	29.16	0.9055	78.82	8.88
s121_3.jpg	29.89	0.9201	66.73	8.17

- **PSNR (Peak Signal-to-Noise Ratio):**

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right)$$

- Measures the image's similarity to a reference. - Higher PSNR (approximately equal to or greater than 30 dB) indicates better quality. - Example: `s105 1.jpg`, PSNR = 29.49 dB (decent quality).

- **SSIM (Structural Similarity Index):**

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\text{Iorig}(i) - \text{Iprocessed}(i))^2$$

- Evaluates perceived similarity based on structure, luminance, and contrast. - Ranges from 0 (no similarity) to 1 (identical images). - Example: `s105 1.jpg`, SSIM = 0.9013 (high similarity).

- **MSE (Mean Squared Error):** - Measures pixel-wise difference between images. - Lower MSE indicates better quality. - Example: `s105 1.jpg`, MSE = 73.14 (low error).
- **RMSE (Root Mean Squared Error):** - Square root of MSE, provides error in pixel intensity units. - Example: `s105 1.jpg`, RMSE = 8.55.

(a) s105_1.jpg original

(b) s105_1.jpg denoised

(a) s105_4.jpg original

(b) s105_4.jpg denoised

(a) s121_1.jpg original

(b) s121_1.jpg denoised

(a) s121_3.jpg original

(b) s121_3.jpg denoised

We do this denoising on all the images

IV. MOCOV2

Contrastive learning is a self-supervised learning technique widely used in machine learning to learn representations by contrasting positive and negative pairs of data samples. This method enables models to learn similarity and dissimilarity among samples, which is especially useful in tasks where labeled data is scarce, such as in computer vision and natural language processing. Here's a breakdown of the key concepts and methods in contrastive learning:

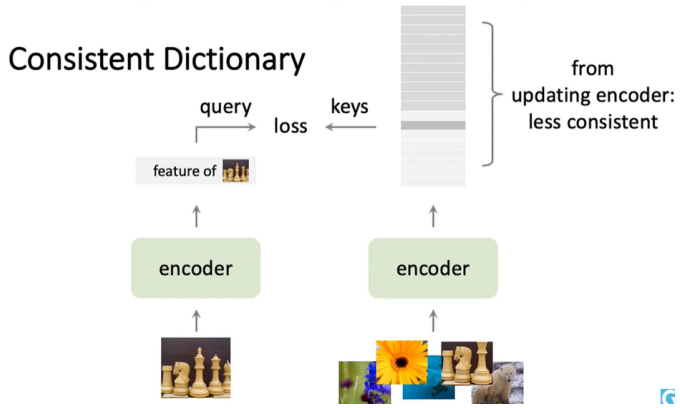

Fig. 10: Momentum-based update for the dictionary in MoCo, where the key encoder parameters θ_k are gradually updated using the query encoder parameters θ_q with a momentum coefficient m.

A. Objective of Contrastive Learning

The goal of contrastive learning is to learn an embedding space where similar samples (positives) are close together, while dissimilar samples (negatives) are far apart. Typically, the training process uses a loss function that minimizes the distance between positive pairs and maximizes the distance between negative pairs in the embedding space.

B. Dictionary Updation

In MoCo, the equation for updating the key encoder parameters is given by:

$$\theta_k := m \cdot \theta_k + (1 - m) \cdot \theta_q$$

where θ_k represents the parameters of the key encoder, θ_q denotes the parameters of the query encoder, and m is the momentum coefficient (close to 1). This momentum-based update ensures that the key encoder parameters θ_k evolve gradually over time, preventing instability from direct copying of the query encoder parameters. The gradual update helps maintain stable, consistent key representations, which is essential for robust contrastive learning and effective dictionary maintenance in MoCo.

C. Positive and Negative Samples

- **Positive pairs** are samples that are meant to be similar to each other. For example, in image processing, a positive pair could consist of two different augmentations of the same image.
- **Negative pairs** are samples that should be dissimilar to each other. Negative pairs could be composed of different images in the dataset.

D. Contrastive Loss Function

A common loss function in contrastive learning is contrastive loss or InfoNCE (Noise-Contrastive Estimation) loss. It encourages similar representations for positive pairs and penalizes similar representations for negative pairs. In its general form, the InfoNCE loss for a pair of positive samples and a set of negatives can be written as:

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k^+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)} \tag{1}$$

where:

- q is the query representation,
- k^+ is the positive key,
- k_i represents negative keys,
- τ is a temperature scaling factor.

E. Advantages of Contrastive Learning

- **Efficiency with Unlabeled Data**: It leverages self-supervision, making it valuable for training with unlabeled data, where supervised labels are limited or unavailable. Since the finger prints given to us were unlabeled hence contrastive learning suits well for training the model using the given dataset.
- **Generalization**: Models trained with contrastive learning generalize well because they learn a robust feature space that captures intrinsic similarities and differences among data.

V. Momentum Control Version 2 (MCV2)

Momentum Control Version 2 (MCV2) is an enhanced version of momentum applied in optimization algorithms, especially in machine learning, to improve convergence speed and stability. While traditional momentum reduces oscillations and accelerates convergence, MCV2 refines these advantages by adaptively modifying the momentum term.

A. Basic Momentum Control

In the basic momentum method, the parameter update is influenced by both the current gradient and previous update. The equations are:

$$v_t = \beta v_{t-1} + \eta \nabla J(\theta_t)$$
$$\theta_{t+1} = \theta_t - v_t$$

where:

- v_t: momentum term at time step t,
- β: momentum coefficient (e.g., 0.9),
- η: learning rate,
- $\nabla J(\theta_t)$: gradient of the cost function J at θ_t,
- θ_{t+1}: updated parameter.

B. Momentum Control Version 2 (MCV2) Overview

MCV2 introduces an adaptive approach for the momentum term, improving stability and performance in complex landscapes by dynamically adjusting the velocity based on gradient magnitude.

C. Mathematical Formulation of MCV2

In MCV2, the update rule adapts using an adaptive momentum scaling factor, α_t, tuned based on previous gradients:

$$v_t = \alpha_t \cdot v_{t-1} + \eta \cdot \nabla J(\theta_t)$$

where α_t adjusts based on the ratio of current to previous gradient magnitudes:

$$\alpha_t = \frac{\|\nabla J(\theta_t)\|}{\|\nabla J(\theta_{t-1})\|}$$

This scaling factor α_t controls the momentum, increasing it when gradient magnitude increases and decreasing it when it decreases.

D. Final Parameter Update in MCV2

The parameter update for MCV2 is:

$$\theta_{t+1} = \theta_t - (\alpha_t v_{t-1} + \eta \nabla J(\theta_t))$$

E. Practical Implementation of MCV2 with a CNN and Wavelet-Based Feature Extraction

To apply MCV2 within a CNN model for feature extraction, we can follow these steps:

F. Image Augmentation

Prepare images with various transformations, including:

- **Rotated Images:** Rotate images at various angles.
- **Scaled Images:** Scale images to different sizes.
- **Blurred Images:** Apply Gaussian or motion blur.
- **Normal Images:** Use original images as baseline.

In contrastive learning, augmented images are essential because they create positive pairs by providing different views or versions of the same image. These augmentations—such as cropping, rotation, or color jittering—help the model learn representations that are invariant to these transformations. By contrasting positive pairs (augmented versions of the same image) with negative pairs (different images), the model learns to focus on meaningful features, improving generalization and robustness. Augmented images also reduce the reliance on large labeled datasets, as they expand the diversity of available data without additional labeling.

Fig. 11: Examples of transformed images: normal, rotated, and blured.

G. Training the CNN Model with MCV2

To train the CNN using MCV2:

- Initialize parameters θ_0, velocity $v_0 = 0$, and adaptive momentum $\alpha_0 = 1$.
- For each training step t:
  1) Compute the gradient $\nabla J(\theta_t)$.
  2) Update $\alpha_t = \frac{\|\nabla J(\theta_t)\|}{\|\nabla J(\theta_{t-1})\|}$.
  3) Update the velocity: $v_t = \alpha_t \cdot v_{t-1} + \eta \cdot \nabla J(\theta_t)$.
  4) Update the parameters: $\theta_{t+1} = \theta_t - v_t$.
- Repeat until convergence.

VI. ResNet-50 in the im4MEC Pipeline

A. Introduction to ResNet-50

ResNet-50 is a 50-layer convolutional neural network commonly used for extracting detailed features from images. Its architecture employs residual (skip) connections, which help retain gradients during backpropagation and make training deep networks feasible.

B. Architecture

The ResNet-50 model consists of several layers of convolutional and pooling operations. Its structure includes bottleneck blocks (1x1, 3x3, 1x1 convolutions), which balance efficiency and representational power. The final output of ResNet-50 is a 2048-dimensional feature vector.

C. Feature Extraction Process

ResNet-50 extracts hierarchical features from images, capturing patterns from low-level edges to high-level textures and shapes. In the im4MEC pipeline, each tile or patch processed by ResNet-50 is summarized into a 2048-dimensional vector that represents unique characteristics of the image.

D. Self-Supervised Learning (SSL) Pre-training with MoCo

MoCo v2 (Momentum Contrast) is used to pre-train ResNet-50 in a self-supervised manner, allowing it to learn useful representations without labeled data. This pre-training enhances ResNet-50's ability to generalize to new images, capturing detailed and transferable features.

E. Advantages and Limitations

Advantages: ResNet-50 provides a strong balance of depth and computational efficiency. SSL pre-training further enhances its robustness in feature extraction.

Limitations: High memory and computational demands can be a limitation, especially when handling large image datasets.

F. Conclusion

ResNet-50 is integral to the im4MEC feature extraction process, with SSL pre-training helping it capture meaningful and robust representations of image data for downstream analysis.

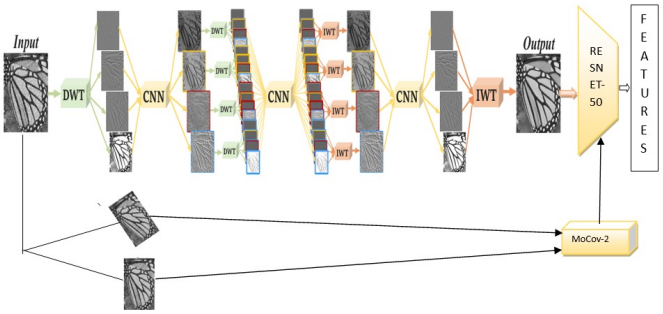

Fig. 12: Complete Network of what we are using to extract features from fingerprints

VII. Implementation of Pipeline, Steps Followed and Results

A. Denoising using MWCNN and DWT/IWT

All the images in the given dataset in the IITB_Fingerprint_Dataset Scanner (clicking on this gives the link) first processed using Discrete Wavelet Transform (DWT) using Daubechies filter(db1) (we also used haar but daubechies gave better results), which decomposes it into sub-bands capturing fine details and broad structures across different frequency levels. These sub-bands are then analyzed by convolutional neural network (CNN) layers within the MWCNN framework, extracting hierarchical features at multiple scales while simultaneously capturing spatial (e.g., edges, textures) and frequency information. Finally, the features extracted by the CNN are reconstructed into the spatial domain using Inverse Wavelet Transform (IWT), integrating the processed sub-bands into a coherent representation that preserves both spatial and frequency details of the original image.

B. ResNet Pre-trained with MoCoV2

We started by using the denoised images to train a MoCoV2 model which helps with invariance to rotation, scaling, and random blurs, running the training process for 100 epochs (image of training results and saving weights in next page). This training helped the model learn to identify and distinguish between images and their slightly altered versions (e.g., rotated, blurred, or slightly noisy). By doing this, the model became good at extracting features that are not easily affected by such variations.

Once the training was complete, we saved the learned weights (essentially, the knowledge the model gained during training) and applied these weights to a ResNet-50 model. This pre-trained ResNet-50 was then used to analyze new images during testing. With the MoCoV2 training, ResNet-50 could extract reliable and consistent features from the images, even if the images had minor imperfections or changes. This approach ensures that the features are robust and effective for further analysis or tasks.

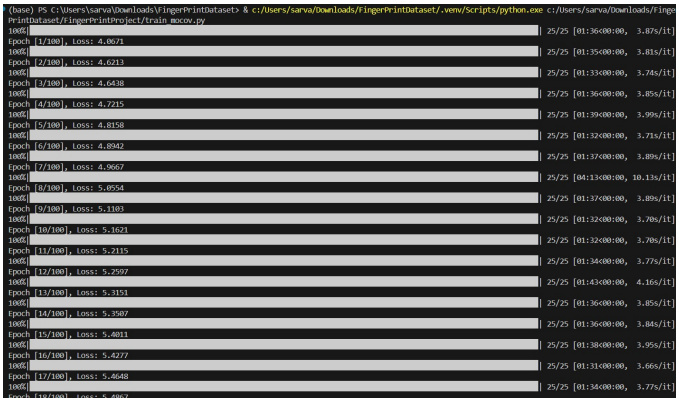

Fig. 13: Started training

Fig. 14: Completed training after 2.5 hours and saved the weights

C. Testing

We use a ResNet-50 model to extract important features from our images. ResNet-50 is a powerful tool that helps us identify and capture essential patterns and details within each image. After extracting these features, we apply Principal Component Analysis (PCA), a statistical technique that simplifies the data by reducing its complexity. Think of PCA as a way to summarize the most important information from a large set of data into just a few key components.

Specifically, we use PCA to identify the top two features that capture the most variation in our data. By plotting these two principal components, we can visualize how the images relate to each other. This visualization helps us see whether the images naturally group together into distinct categories or clusters. If the points on the plot form clear clusters, it indicates that the features are classifiable—meaning we can effectively separate different groups of images based on their characteristics. On the other hand, if the points are scattered without any obvious grouping, it suggests that the features may not be easily distinguishable.

Overall, this process allows us to assess whether the features extracted by ResNet-50 are effective for distinguishing between different classes of images, providing a visual confirmation of their separability and the potential for accurate classification.

For texting, we are using 4 images of fingerprints of the same person, and sending it to the resnet-50 and applying PCA to reduce the dimension from 2048 to 2, and plotting them.

D. Results

The images of the results are as in the attached 3 images. We see that feeatures are mostly classifiable considering a lot of people, some points are misclassified sometimes and some people have no clustering. More images are in the ”Results Folder” in the project folder sent to the TA which also has all the code used.

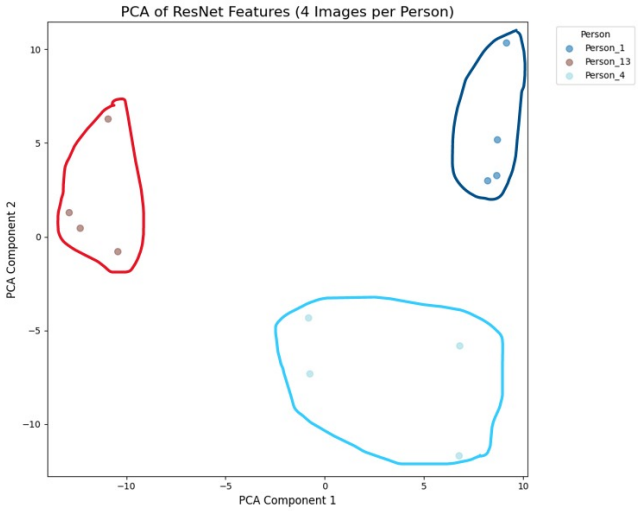

Fig. 15: Results: Clear distinction between features

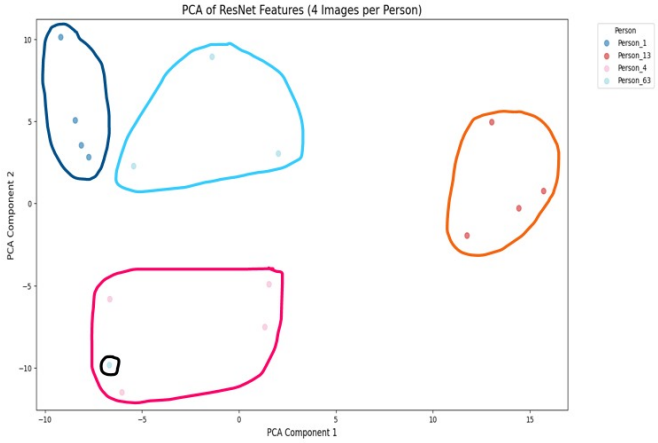

Fig. 16: Results: We see that mostly the features of the fingerprints of same people have closer points than those with different fingerprints (although there is one non-clustered point circled in black)

REFERENCES

[1] S.-F. Wang, W.-K. Yu, and Y.-X. Li, ”Multi-Wavelet Residual Dense Convolutional Neural Network for Image Denoising,” *IEEE Access*, vol. 8, pp. 185427-185436, Nov. 2020, doi: 10.1109/ACCESS.2020.3040542. [Online]. Available: https://ieeexplore.ieee.org/document/9271294.

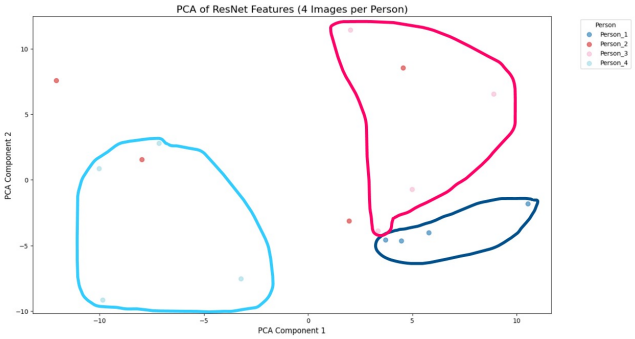

Fig. 17: Results: Here We see how person 2 doesnt have similar clustering as the other people

[2] T. Guo, H. S. Mousavi, T. H. Vu, and V. Monga, "Deep wavelet prediction for image super-resolution," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 1100–1109.

[3] L. M. P. T. et al., "Interpretable deep learning model to predict the molecular classification of endometrial cancer from haematoxylin and eosin-stained whole-slide images: a combined analysis of the PORTEC randomised trials and clinical cohorts," *The Lancet Digital Health*, vol. 2, no. 4, pp. e174–e184, Apr. 2020, doi: 10.1016/S2589-7500(20)30063-7.

[4] X. Chen et al., "Improved baselines with momentum contrastive learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2020, pp. 9729–9738, doi: 10.1109/CVPR42600.2020.00978.

[5] P. Liu, H. Zhang, W. Lian, and W. Zuo, "Multi-level wavelet convolutional neural networks," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4939-4951, 2019.

[6] AIRMEC, "im4MEC," GitHub Repository, Available: https://github.com/AIRMEC/im4MEC.

[7] Connor Shorten, "Video explaining the project," YouTube, Available: https://youtu.be/LvHwBQF14zs?si=GrO7NRNS7zDj5D5H.

[8] ComputerVisionFoundation Videos, "Another video related to the project," YouTube, Available: https://youtu.be/4VVGtYPM8JE?si=XanU793IqJH8qmiV.