Image Enhancement for Social Robots: A Comparison between Convolutional Neural Network (CNN) and Large Vision Model(LVM)

Namrata Roy
Department of Mathematics & Statistics
University of Guelph
Guelph, Canada
nroy05@uoguelph.ca

Enas Altarawneh
Department of Mathematics & Statistics
University of Guelph
Guelph, Canada
ealtaraw@uoguelph.ca

***Abstract*— Social robots are AI platforms equipped with sensors, cameras, and microphones to better interact with humans and other robots. A high-quality image enables robots to perform tasks that require perception, cognition, and action based on visual inputs. Image enhancement algorithms play a pivotal role in enhancing the interpretation of remotely sensed data by producing a refined image. This refined image is often more understandable than the original image. Enhancing images is crucial for social robots to understand human facial expressions and environments better, leading to improved interaction quality and effectiveness. A conventional neural network (CNN) can identify and differentiate between distinct aspects of an image. This study developed two different models of CNN, one using the Rectified Linear Unit (ReLU) and another using the Leaky Rectified Linear Unit (LeakyReLU) to achieve the aim of improving the quality of images. On the other hand, the large vision model (LVM) is a deep learning model specifically designed for computer vision tasks that require a large architecture, parameters, and computational resources. Here the CNN model using the ReLU technique yields an average Peak signal-to-noise ratio (PSNR) of 15.70 dB and the model with LeakyReLU performs better with an average PSNR of 19.97 dB. However, the LVM yields an average PSNR of 16.99 dB.**

Keywords*— *image enhancement, Convolutional Neural Network (CNN), Rectified Linear Unit (ReLU), Leaky Rectified Linear Unit (LeakyReLU), Large Vision Model (LVM), Peak signal-to-noise ratio (PSNR)

I. INTRODUCTION

A social robot is a robot designed specifically to interact and communicate with humans using human-like appearance, behaviour, and communication in social settings. Creating and reinforcing social cues that support interaction with humans is essential in designing social robots that interact socially with humans by referring to social behavioural patterns [1]. The process of image enhancement involves improving their appearance and quality through adjustments such as brightness and contrast, histogram equalization, noise reduction, image sharpening, and filtering for a specific application, using various techniques to make it more visually appealing or useful. A Convolutional Neural Network (CNN) is a type of artificial neural network that is designed to process and analyze visual data efficiently. They are widely used in image enhancement because they can directly learn hierarchical representations from raw pixel data [2]. A CNN is suggested to connect the input (i.e. actual image) and the output (i.e. enhanced image) [3] to improve the contrast of underwater [4] or low-light images [5].

A large language model (LLM) refers to a deep learning model used for natural language processing (NLP) tasks. It has an extensive architecture with a vast number of parameters. The rise of large language models has revolutionized traditional methods for NLP tasks [5]. The research community is now focusing on achieving vision universality following the remarkable success of language models known as large vision models (LVM). They aim to develop multi-skilled, general-purpose vision systems that can handle a wide range of vision tasks, with the help of language assistance [5].

Large vision models are advanced artificial intelligence systems built using deep learning architectures and are trained on vast datasets to acquire the ability to understand and analyze visual data. Improved visual understanding, better visual representations, and enhanced performance on downstream tasks have been observed with larger models. These improvements have been consistently demonstrated across a broad range of downstream tasks, provided that sufficient pre-training data is available [6]. These models are intended to achieve high performance on computer vision tasks like defect detection or object location with fewer labelled data.

The study involves using two CNN-based techniques: Rectified Linear Unit (ReLU) and Leaky Rectified Linear Unit (LeakyReLU) to dynamically enhance images by fine-tuning their contrast, brightness, and noise levels. Furthermore, it develops a new approach for image enhancement using an LVM technique. Finally, it compares the outcomes from the models to better understand the model efficiency. Section 2 contains a brief description of social robots and the importance of image enhancement for social robots. Furthermore, it discusses prior works done in this sector. Section 3 describes the dataset. The architecture of the proposed models has been explains in Section 4. The evaluation of the models and results are covered in Section 5. Section 6 includes an insightful discussion of the findings.

II. LITERATURE REVIEW

The four robots AIBO (Sony), iCat (Philips), BIRON (Bielefeld University) and BARTHOC (Bielefeld University) use several sensors like color camera and distance sensors, stereo microphones, acceleration sensors, webcam, loudspeaker, etc. to communicate with a human. These robots performed altogether 570 applications such as security, research, personal assistant, teaching, health care, business, toy, pet, transport etc. [8].

When robots need to perform tasks that require them to sense, think, and act, image processing can help them receive visual inputs [9]. For instance, image processing techniques such as image enhancement, restoration, and segmentation can help robots navigate, locate objects, identify faces, read signs, inspect defects, or generate maps. Robots can also use

image processing to communicate with humans or other robots through gestures, expressions, or symbols [9].

The identification of shapes in digital images has various applications, including robot localization [10], identification [11], object measurement [12], and counting [13].

When a digital image is captured under unfavourable conditions such as insufficient settings or bad natural conditions, the resulting pixel values may be of low quality, leading to a significant reduction in image quality [14]. To enhance the quality of the image, several digital image processing techniques such as gamma correction, histogram equalization, spatial filtering, and wavelet transformation are used [15]. Numerous image analysis techniques such as homomorphic filtering (HF) [16], multi-branch low-light enhancement networks (MBLLEN) [5], Convolutional Neural Networks (CNNs) [17], and Generative Adversarial Networks (GANs) [18] have been employed to extract information from an image and analyze complex images. This information provides data related to several fields such as medical science, geospatial conditions, Astro science, security surveillance, underwater life, etc. Weighted guided image filtering is a filtering algorithm that combines edge-based weighting and guided image filtering [19]. In the CNN network, a sub-network reduces the network depth needed to obtain the same features [3].

Google Research introduced a recipe to train a 22B-parameter ViT highly efficiently and stably performed excellently when training thin layers on top of frozen models to produce embeddings [20]. Another study showed that ViT models with sufficient training data roughly follow a saturating power law similar to the scaling law of LLMs in terms of the performance-compute frontier [21]. Sora is a large vision model aligned with scaling principles and the first to exhibit emergent abilities in text-to-video generation, marking a milestone in computer vision [22]. To model image pixels in visual sentences, a two-stage approach was taken in a recent study. First, a visual tokenizer was trained to convert each image into a sequence of visual tokens. Then, an autoregressive transformer model was trained on visual sentences, each represented as a sequence of tokens [23].

Another work described a framework named ViGoR which is used to improve the visual grounding capabilities of Large Vision Language Models (LVLMs). The framework combines fine-grained reward modelling of human preferences with existing open-set visual perception models to reduce errors in relational reasoning, hallucination, and counting to efficiently improve LVLMs in these aspects. [24].

III. Dataset

To fulfill our objective, a dataset [25] from Kaggle was used comprising 500 pairs of low-light and normal-light images. This dataset is further divided into 485 training pairs and 15 testing pairs. The low-light images in this dataset contain noise that was produced during the photo capture process. The majority of these images depict indoor scenes, and all of them have a resolution of 400×600.

This dataset is appropriate for evaluating image enhancement techniques for social robots. The low-light images challenge the robot's visual processing capabilities, highlighting the effectiveness of the enhancement algorithms in simulating human-like vision under poor lighting conditions, which is essential for reliable performance in diverse and dynamically changing environments.

Image processing is crucial for robots as it helps them interact with their environment, navigate, and perform various tasks autonomously and intelligently. Enhancing indoor images is particularly important for social robots to accurately perceive their surroundings, communicate well with humans, evoke emotional responses, and provide a satisfying user experience. This ultimately helps them to fulfill their social roles and objectives more effectively.

IV. Model Architecture

Convolutional neural networks (CNNs) are commonly used for image enhancement, and they require extensive datasets of paired images of input images and corresponding output images for training. These datasets will consist of input images that are intentionally degraded by factors such as low resolution, noise, blur, or a combination of these. Conversely, the corresponding output images will be clean or enhanced. Through the training process, the network will be taught to minimize the difference between the generated output and the ground truth-enhanced images.

Neural networks are structured and function similarly to biological neural networks in the human brain. They consist of interconnected nodes which are arranged in three layers: input, hidden, and output. The connections between nodes are represented by weights. Each node receives input, performs a computation based on the weights, and produces an output. Additionally, the backpropagation technique enables neural networks to learn from data by adjusting their weights in a way that minimizes the difference between the predicted outputs and the actual targets [26]. Fig 4.1 shows that within the hidden layers of CNN architecture, there are three primary components: the convolution layer, the pooling layer, and the fully connected layer [14].

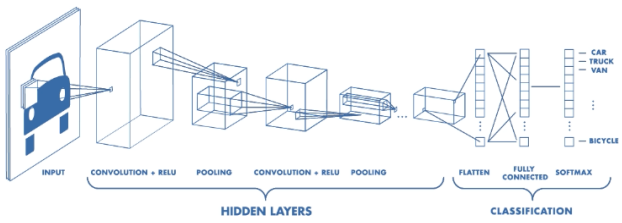

Fig 4.1: Convolutional neural networks Architecture

The convolution layer is a fundamental component designed to automatically and adaptively learn spatial hierarchies of features through backpropagation. It's responsible for carrying out the main computational load of the nodes depicted in Figure 4.2. This layer is considered the core block of the Convolutional Neural Network (CNN). Every image is comprised of pixels that are made up of three color channels: red, green, and blue (RGB). These pixel values are represented as a matrix. The layer performs a dot product between two matrices, where one matrix is the set of learnable parameters known as a kernel, and the other matrix is the restricted portion of the receptive field. The kernel is spatially smaller, but more in-depth than an image.

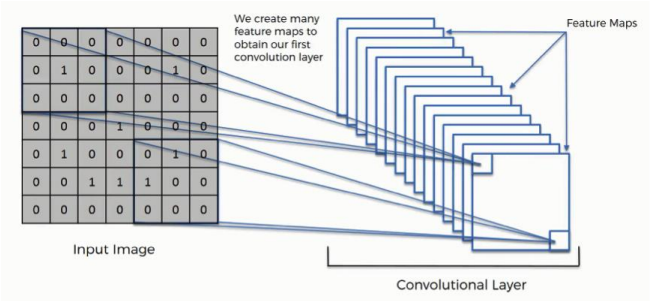

Fig 4.2: Convolution layer

The kernel slides across the height and width of the image-producing a two-dimensional representation of the image known as a feature map that gives the response of the kernel at each spatial position of the image.

The pooling layer, as shown in Figure 4.3, is applied after the Convolutional layer. Its purpose is to reduce the dimensions of the feature map, which helps to preserve the important information or features of the input image, while also reducing the computation time. A pooling layer creates a lower-resolution version of the input image that still contains the large or important elements of the image.

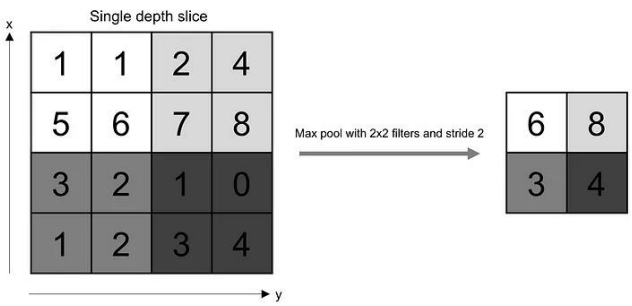

Fig 4.3: Pooling layer

In Figure 4.4, the Fully connected layer plays a crucial role in classifying the input image. Its nodes are fully connected with every node of the preceding and succeeding layers, thus enabling the transfer of information extracted from the earlier stages (such as the Convolution layer and Pooling layers). By computing the weights through matrix multiplication, this layer ultimately classifies the input image into the desired label.

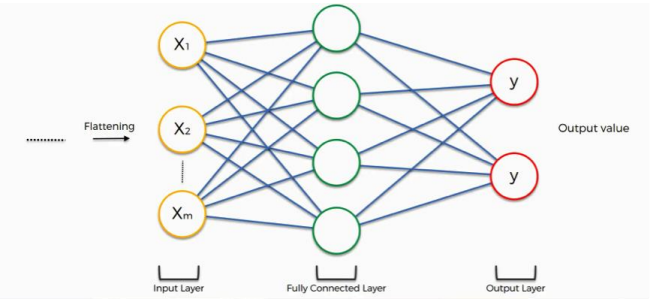

Fig 4.4: Fully connected layer

As convolution is a linear operation and images are non-linear several types of non-linear techniques such as Sigmoid, tanh, Rectified Linear Unit (ReLU) and Leaky Rectified Linear Unit (LeakyReLU) are often used for the computational part of the convolutional layer. In this project, ReLU and LeakyReLU are used as those models are more reliable and accelerate the convergence compared to the other techniques.

Rectified Linear Unit is a type of activation function commonly used in neural networks, particularly in convolutional neural networks. It is defined mathematically as:

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

This means that if the input x is positive, the output is x; if x is negative, the output is 0. The function effectively "clips" negative values to zero, which introduces non-linearity into the model while being computationally efficient.

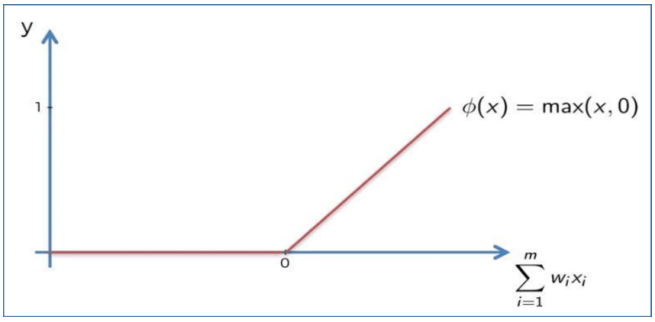

Fig 4.5: Rectified Linear Unit

Fig 4.5 visualizes a graph where the x-axis represents the input to the ReLU function and the y-axis represents the output. The graph would show a line that starts at zero (for all negative inputs) and then rises linearly with slope 1 for all positive inputs. This graph effectively shows how ReLU allows positive values to pass unchanged while blocking negative values by setting them to zero.

In this study, the model inputs an image of shape (SIZE, SIZE, 3), indicating height, width, and color channels (RGB). Fig 4.6 describes a summary of the proposed model. The input image first goes through a convolutional layer with 64 filters of size 3x3, using padding to keep the dimensions unchanged. This layer is followed by a ReLU activation for non-linearity and batch normalization to standardize inputs to the next layer, improving training stability. It concludes with a max pooling (2*2) layer to reduce spatial dimensions by half, emphasizing important features while reducing computation for deeper layers.

This pattern repeats with increasing filters (128 and then 256), each time followed by ReLU activation, batch normalization, and for the first two sets, max pooling. These layers are designed to extract increasingly abstract features from the input image.

After reaching the deepest layer, the network begins to reconstruct the resolution through upsampling layers that double the dimensions, coupled with convolutional layers (128 and then 64 filters) to refine the features, again followed by ReLU and batch normalization.

Finally, a convolutional layer with 3 filters (matching the number of color channels) applies a 3x3 kernel to form the output image. A sigmoid activation function is used here,

scaling the output to a range between 0 and 1, which is typical for image data representing pixel intensity values.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 256, 256, 3)]     0

 conv2d (Conv2D)             (None, 256, 256, 64)      1792

 activation (Activation)     (None, 256, 256, 64)      0

 batch_normalization (Batch  (None, 256, 256, 64)      256
 Normalization)

 max_pooling2d (MaxPooling2  (None, 128, 128, 64)      0
 D)

 conv2d_1 (Conv2D)           (None, 128, 128, 128)     73856

 activation_1 (Activation)   (None, 128, 128, 128)     0

 batch_normalization_1 (Bat  (None, 128, 128, 128)     512
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 128)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 64, 64, 256)       295168

 activation_2 (Activation)   (None, 64, 64, 256)       0

 batch_normalization_2 (Bat  (None, 64, 64, 256)       1024
 chNormalization)

 up_sampling2d (UpSampling2  (None, 128, 128, 256)     0
 D)

 conv2d_3 (Conv2D)           (None, 128, 128, 128)     295040

 activation_3 (Activation)   (None, 128, 128, 128)     0

 batch_normalization_3 (Bat  (None, 128, 128, 128)     512
 chNormalization)

 up_sampling2d_1 (UpSamplin  (None, 256, 256, 128)     0
 g2D)

 conv2d_4 (Conv2D)           (None, 256, 256, 64)      73792

 activation_4 (Activation)   (None, 256, 256, 64)      0

 batch_normalization_4 (Bat  (None, 256, 256, 64)      256
 chNormalization)

 conv2d_5 (Conv2D)           (None, 256, 256, 3)       1731

 activation_5 (Activation)   (None, 256, 256, 3)       0

=================================================================
Total params: 743939 (2.84 MB)
Trainable params: 742659 (2.83 MB)
Non-trainable params: 1280 (5.00 KB)
_________________________________________________________________
```

Fig 4.6: ReLU-based CNN Model Summary

Here out of a total of 743939 params 742659 params are trainable and 1280 are non-trainable params.

The model has been fitted with 100 epochs, each has steps of the length of the training dataset. A callback method was invoked where ‘checkpoint’ saved the model's weights at intervals, ‘early_stop’ monitored training to halt it if the validation performance did not improve for a set number of epochs, preventing overfitting and saving computational resources and ‘reduce_lr’ adjusted the learning rate dynamically if there was no improvement in training, which can help in fine-tuning the model when it was close to converging.

In addition, a leaky Rectified Linear Unit is a variation of the ReLU activation function, designed to address one of the potential problems of the ReLU function. This problem occurs when neurons effectively "die" during training, meaning they stop outputting anything other than zero, which in turn can halt learning in part of the neural network since gradients do not flow through completely inactive neurons. The mathematical definition is:

$$f(x) = \begin{cases} x, & x < 0 \\ ax, & x \geq 0 \end{cases}$$

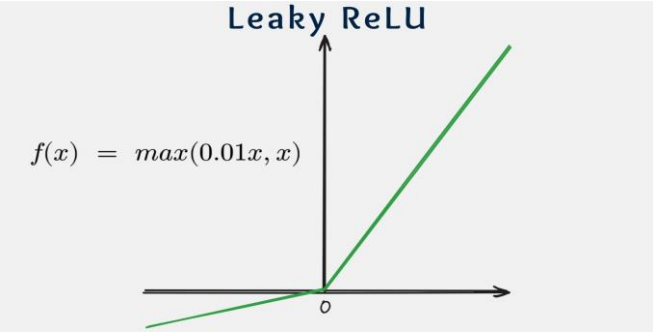

Fig 4.7: Leaky Rectified Linear Unit

Fig 4.7 shows that LeakyReLU modifies the ReLU function by allowing a small, non-zero, constant gradient alpha α (often small) when the unit is inactive and the input is less than zero. In this project, the value of the alpha α is 0.02.

In the project, to develop the LeakyReLU model, U-Net architecture has been implemented, where spatial hierarchies of features are important. Fig 4.8 explains a summary of the proposed model. This U-Net architecture consists of an encoder (downsampling path) and a decoder (upsampling path) with skip connections that help preserve spatial information throughout the network.

The function down defines the downsampling step using convolutional layers with stride 2 to reduce dimensionality, optionally applying batch normalization for more stable training, and using LeakyReLU for activation to allow a small gradient when the unit is not active (helpful in preventing dying neurons).

These downsampling layers are sequentially applied in the model function to progressively reduce the spatial dimensions while increasing the depth (number of filters), capturing increasingly abstract representations of the input. After the downsampling layers, the deepest point (bottleneck) processes the most compressed representation, which is crucial for capturing the core features of the input.

Correspondingly, the up function is used for the upsampling steps, employing Conv2DTranspose layers to increase the resolution of the feature maps. Optionally, dropout can be added to prevent overfitting by randomly dropping units during training. Each upsampling step is followed by a concatenation with the correspondingly mirrored layer from the downsampling path (skip connection), which helps in recovering the precise spatial information lost during downsampling.

The upsampling path progressively reconstructs the resolution of the output back to the input size. The final output of the model is obtained by a convolutional layer that adjusts the depth to 3 channels (suitable for RGB images), ensuring the output size matches the input size

Model: "functional_11"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 256, 256, 3)	0	-
sequential (Sequential)	(None, 128, 128, 128)	3,584	input_layer[0][0]
sequential_1 (Sequential)	(None, 64, 64, 128)	147,584	sequential[0][0]
sequential_2 (Sequential)	(None, 32, 32, 256)	296,192	sequential_1[0][…
sequential_3 (Sequential)	(None, 16, 16, 512)	1,182,208	sequential_2[0][…
sequential_4 (Sequential)	(None, 8, 8, 512)	2,361,856	sequential_3[0][…
sequential_5 (Sequential)	(None, 16, 16, 512)	2,359,808	sequential_4[0][…
concatenate (Concatenate)	(None, 16, 16, 1024)	0	sequential_5[0][… sequential_3[0][…
sequential_6 (Sequential)	(None, 32, 32, 256)	2,359,552	concatenate[0][0]
concatenate_1 (Concatenate)	(None, 32, 32, 512)	0	sequential_6[0][… sequential_2[0][…
sequential_7 (Sequential)	(None, 64, 64, 128)	589,952	concatenate_1[0]…
concatenate_2 (Concatenate)	(None, 64, 64, 256)	0	sequential_7[0][… sequential_1[0][…
sequential_8 (Sequential)	(None, 128, 128, 128)	295,040	concatenate_2[0]…
concatenate_3 (Concatenate)	(None, 128, 128, 256)	0	sequential_8[0][… sequential[0][0]
sequential_9 (Sequential)	(None, 256, 256, 3)	6,915	concatenate_3[0]…
concatenate_4 (Concatenate)	(None, 256, 256, 6)	0	sequential_9[0][… input_layer[0][0]
conv2d_5 (Conv2D)	(None, 256, 256, 3)	75	concatenate_4[0]…

Total params: 9,602,766 (36.63 MB)
Trainable params: 9,600,206 (36.62 MB)
Non-trainable params: 2,560 (10.00 KB)

Fig 4.8: LeakyReLU-based CNN Model Summary

In this model out of a total of 9602766 params 9600206 params are trainable and 2560 are non-trainable params.

This model used the Adam optimizer with a learning rate of 0.001 and 'mean_absolute_error' as its loss function. The model was trained with 100 epochs, each with a set of 60 steps.

In contrast, the model used for the LVM technique shown in Fig 4.9 defines an autoencoder neural network architecture for image data. The autoencoder is composed of an encoder and a decoder:

- Input Layer: It starts with an input layer expecting images of shape 256x256 with 3 color channels (RGB).
- Encoder: The input is passed through two convolutional layers, each followed by LeakyReLU for non-linear activation. The feature maps are downsampled twice using max pooling, reducing the dimensions to 64x64 while increasing the depth of the feature maps first to 64 and then to 128.
- Bottleneck: The encoder's output then goes through another convolutional layer without downsampling, creating a bottleneck where the model captures the most important features of the input data.
- Decoder: This bottleneck is then upsampled back to the original image size through two sets of upsample and convolutional layers, each again followed by LeakyReLU activation.
- Output Layer: The final convolutional layer has 3 filters with a sigmoid activation function to reconstruct the original image, producing the same number of color channels as the input.

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 3)]	0
conv2d (Conv2D)	(None, 256, 256, 64)	1792
leaky_re_lu (LeakyReLU)	(None, 256, 256, 64)	0
max_pooling2d (MaxPooling2D)	(None, 128, 128, 64)	0
conv2d_1 (Conv2D)	(None, 128, 128, 128)	73856
leaky_re_lu_1 (LeakyReLU)	(None, 128, 128, 128)	0
max_pooling2d_1 (MaxPooling2D)	(None, 64, 64, 128)	0
conv2d_2 (Conv2D)	(None, 64, 64, 256)	295168
leaky_re_lu_2 (LeakyReLU)	(None, 64, 64, 256)	0
up_sampling2d (UpSampling2D)	(None, 128, 128, 256)	0
conv2d_3 (Conv2D)	(None, 128, 128, 128)	295040
leaky_re_lu_3 (LeakyReLU)	(None, 128, 128, 128)	0
up_sampling2d_1 (UpSampling2D)	(None, 256, 256, 128)	0
conv2d_4 (Conv2D)	(None, 256, 256, 64)	73792
leaky_re_lu_4 (LeakyReLU)	(None, 256, 256, 64)	0
conv2d_5 (Conv2D)	(None, 256, 256, 3)	1731

Total params: 741379 (2.83 MB)
Trainable params: 741379 (2.83 MB)
Non-trainable params: 0 (0.00 Byte)

Fig 4.9: Large Vision Model Summary

The model has been compiled using the Adam optimizer and mean squared error as the loss function. This means that the model's performance will be evaluated based on how accurately it can reconstruct images from the original inputs. The model architecture is typical for autoencoders used in denoising or feature learning tasks. It's worth mentioning that there's a provision for adding skip connections, which could further assist the model during training by providing

additional paths for gradient flow, although it is currently commented out.

In this model out of a total of 741379 params 741379 params are trainable and 0 are non-trainable params.

During the training process, various functions have been employed to enhance image quality. These functions include but are not limited to colorization, deblurring, image restoration, and other enhancement filters. Such approaches have been instrumental in improving the overall quality of the images employed in the training process.

V. Result and Discussion

The models were implemented using Keras and TensorFlow. The code runs in Jupyter Notebook on the Kaggle website environment. It includes other several libraries such as matplotlib, pydot, sklearn, Sequential, pandas, numpy, opencv-python etc.

Fig 5.1 and Fig 5.2 show a visual comparison of the original low-light image, the ground truth image, and the enhanced image for the ReLU-based CNN model. Such visual comparisons are essential in the evaluation of the model's efficacy in addressing the challenges of low-light imaging.

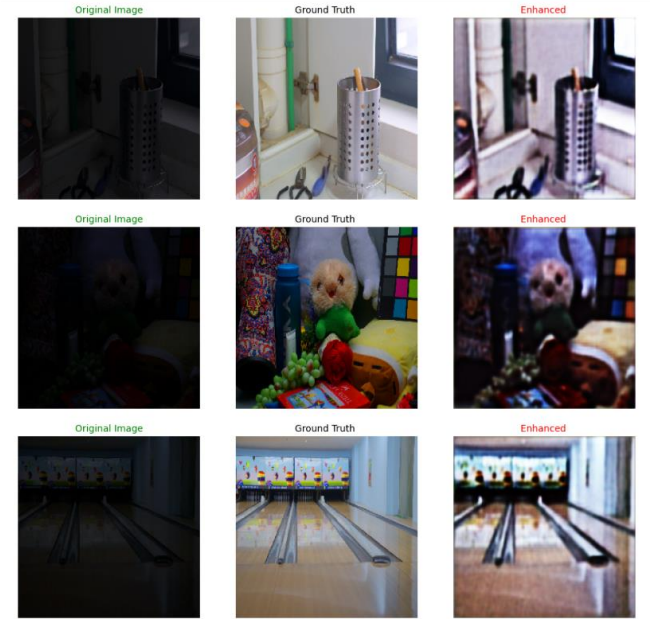

Fig 5.1: Results using ReLU-based CNN model sample 1

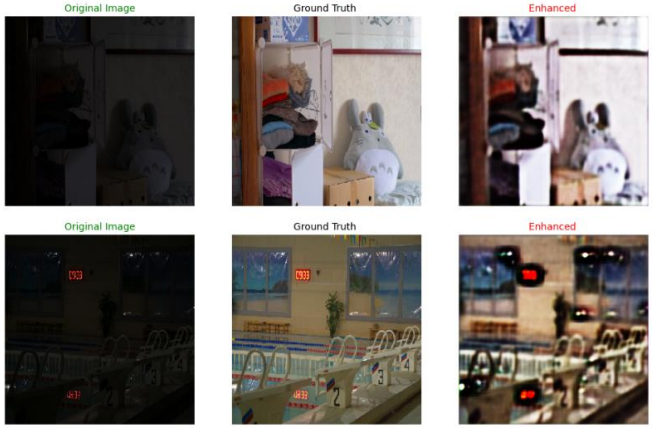

Fig 5.2: Results using ReLU-based CNN model sample 2

Visual comparisons for the LeakyReLU-based CNN model between the original low-light image, the ground truth image, and the enhanced image are shown in Fig 5.3, Fig 5.4, and Fig 5.5.

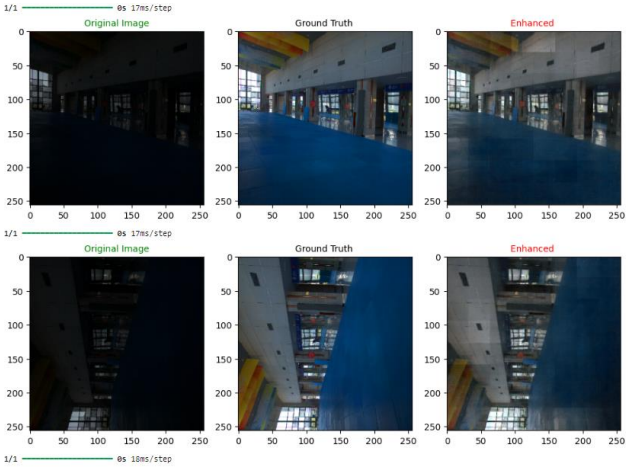

Fig 5.3: Results using LeakyReLU-based CNN model sample 1

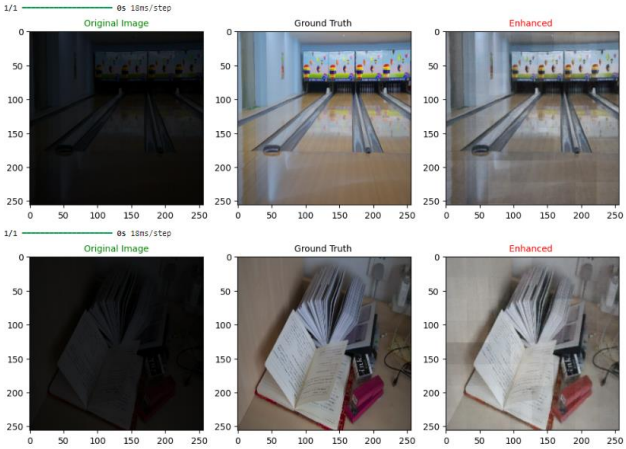

Fig 5.4: Results using LeakyReLU-based CNN model sample 2

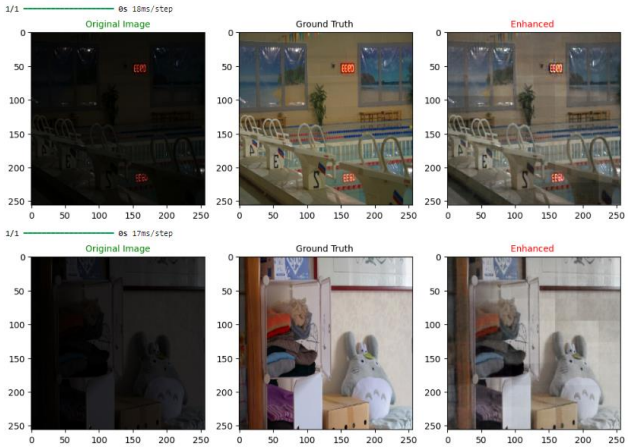

Fig 5.5: Results using LeakyReLU-based CNN model sample 3

Figures 5.6 and 5.7 show visual comparisons for the large vision model among the original low-light image, the ground truth image, and the enhanced image.

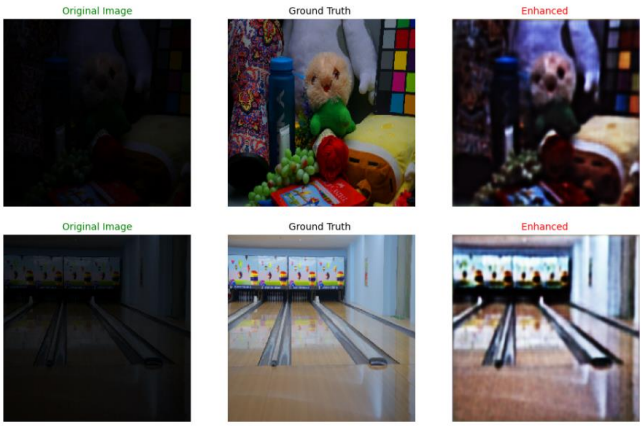

Fig 5.6: Results using LVM model sample 1

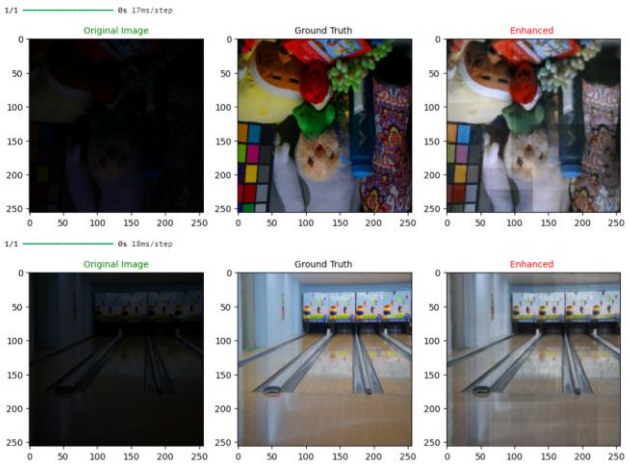

Fig 5.7: Results using LVM model sample 2

Comparing the enhanced images of the ReLU, LeakyReLU function-based CNN models and the LVM model it can be said that the enhanced images of LeakyReLU-based CNN models are visibly clearer than the other models.

In order to evaluate the effectiveness of each model, the Peak Signal-to-Noise Ratio (PSNR) has been measured. This is a ratio that compares the maximum possible power of a signal to the power of any noise that may corrupt its quality. PSNR is calculated using a logarithmic scale based on the mean squared error between the original and processed images. It is expressed in decibels (dB).

Higher PSNR Values imply that the corruption is less dominant compared to the signal's maximum power, indicating a higher quality or a closer approximation to the original image. Lower PSNR Values suggest more significant differences between the original and processed images, indicating more apparent distortions, artifacts, or loss of details.

Fig 5.8 depicts the average PSNR value for the ReLU model is 15.70 dB, whereas the LeakyReLU yields an average of 19.97 dB. On the other hand, the average PSNR value LVM model is 16.99 dB.

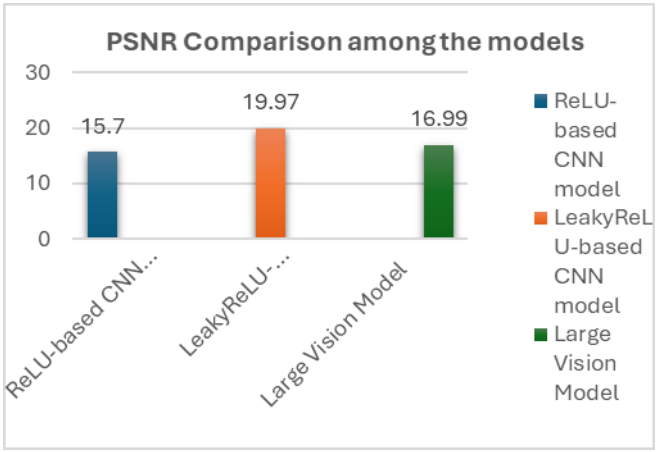

Fig 5.8: PSNR Comparison among the models

To summarize, the proposed models show a significant outcome. When deciding among the image enhancement processing models with differing PSNR values, it is typically advisable to select the option that yielded a higher value, such as 19.97 dB. This indicates that while neither process may be optimal, the one with the higher PSNR is more likely to preserve the image's original quality compared to the option with a lower PSNR of 15.70 dB and 16.99 dB.

VI. Conclusion

Convolutional neural networks (CNNs) use a training process to improve the quality of degraded images by extracting important features and relationships between input and output images. This enables effective image enhancement on new data. To enhance low-light images for social robots, this study proposed conceptual models using CNN architecture with ReLU and LeakyReLU models for computational methods in the convolutional layer.

Alternatively, large vision models trained on massive datasets require significant computational resources for training and deployment. However, they could be utilized to enhance the visual quality of images across various domains and applications.

While the LeakyReLU-based CNN model performed slightly better in terms of average PSNR value than the other models, the dataset size may have been a limiting factor. A larger dataset could improve the model's efficiency rate. Moreover, there was a limitation of time to conduct the paper.

This paper proposes conceptual models for image processing using Convolutional Neural Networks (CNNs) and Large Vision Model (LVM). In future work, a large dataset with different model tunning approaches can be used to improve the model's accuracy.

References

[1] https://medium.com/@michealomis99/image-processing-for-robotics-enabling-advanced-perception-and-control-915

[2] https://www.ibm.com/topics/convolutional-neural-networks

[3] Zheng, Meicheng & Luo, Weilin. (2022). Underwater Image Enhancement Using Improved CNN Based Defogging. Electronics. 11. 150. 10.3390/electronics11010150.

[4] Minhas, T.; Hassan, F.; Irshad, R. Underwater Image Enhancement Using Hyper-Laplacian Reflectance Priors and CNN-Based Classification. Eng. Proc. 2023, 46, 14. https://doi.org/engproc2023046014.

[5] Lv, Feifan & Lu, Feng & Wu, Jianhua & Lim, Chongsoon. (2022). MBLLEN: Low-light Image/Video Enhancement Using CNNs.

[6] Shi, B., Wu, Z., Mao, M., Wang, X., & Darrell, T. (2024). When do we not need larger vision models? arXiv (Cornell University). https://doi.org/10.48550/arxiv.2403.13043

[7] Alaguselvi, R., & Murugan, K. (2019). Image Enhancement Using Convolutional Neural Networks. 2019 IEEE International Conference on Clean Energy and Energy Efficient Electronics Circuit for Sustainable Development (INCCES). doi:10.1109/incces47820.2019.9167741.

[8] Hegel, F., Muhl, C., Wrede, B., Hielscher-Fastabend, M., & Sagerer, G. (2009). *Understanding social robots*. https://doi.org/10.1109/achi.2009.51

[9] Hegel, Frank & Lohse, Manja & Swadzba, Agnes & Wachsmuth, Sven & Rohlfing, Katharina & Wrede, Britta. (2007). Classes of Applications for Social Robots: A User Study. 938-943. 10.1109/ROMAN.2007.4415218.

[10] Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2009). Digital Image Processing, third edition. *Journal of Biomedical Optics*, *14*(2), 029901. https://doi.org/10.1117/1.3115362

[11] Y. Ma, S. Soatto, J. Košecká, S.S. Sastry, An Invitation to 3-D Vision, Springer, New York, NY, 2004, doi: 10.1007/978-0-387-21779-6.

[12] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, Cambridge, 2004, doi:10.1017/CBO9780511811685.

[13] E. Trucco, A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice Hall, Upper Saddle River, NJ, 1998.

[14] Alaguselvi, R., & Murugan, K. (2019). Image Enhancement Using Convolutional Neural Networks. 2019 IEEE International Conference on Clean Energy and Energy Efficient Electronics Circuit for Sustainable Development (INCCES). doi:10.1109/incces47820.2019.9167741.

[15] Tang, Shi & Dong, Mingjie & Ma, Jinlei & Zhou, Zhiqiang & Li, Changqing. (2017). Color image enhancement based on retinex theory with guided filter. 5676-5680. 10.1109/CCDC.2017.7978178.

[16] Chavarin, Ángel & Avalos, Omar & Gálvez, Jorge & Wario, Fernando & Perez Padilla, Nayeli. (2023). Contrast Enhancement in Infrared and Color Images by Homomorphic Filtering and Cluster-Chaotic Optimization (CCO). 10.52591/lxai202306184.

[17] Kumara, C.T.; Pushpakumari, S.C.; Udhyani, A.J.; Aashiq, M.; Rajendran, H.; Kumara, C.W. Image Enhancement CNN Approach to COVID-19 Detection Using Chest X-ray Images. Eng. Proc. 2023, 55, 45. http://doi.org/10.3390/engproc2023055045.

[18] Nagpal, Sparsh. (2022). Sketch-to-Face Image Translation and enhancement using a Multi-GAN approach. International Journal for Research in Applied Science and Engineering Technology. 10. 10.22214/ijraset.2022.48041.

[19] R. Karumuri and S. A. Kumari, "Weighted guided image filtering for image enhancement," 2017 2nd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2017, pp. 545-548, doi: 10.1109/CESYS.2017.8321137.

[20] Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Riquelme, C., Minderer, M., Puigcerver, J., Evci, U., . . . Houlsby, N. (2023). Scaling vision transformers to 22 billion parameters. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2302.05442

[21] Zhai, X., Kolesnikov, A. I., Houlsby, N., & Beyer, L. (2021). Scaling vision transformers. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2106.04560

[22] Liu, Y., Zhang, K., Liu, Y., Zhang, Y., Gao, C., Chen, R., Yuan, Z., Huang, Y., Sun, H., Gao, J., He, L., & Sun, L. (2024). SORA: a review on background, technology, limitations, and opportunities of large vision models. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2402.17177

[23] Bai, Y., Geng, X., Mangalam, K., Bar, A., Yuille, A., Darrell, T., Malik, J., & Efros, A. A. (2023). Sequential modeling enables scalable learning for large vision models. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2312.00785

[24] Shen, Y., Bai, M., Chen, W., Xiong, Z., Huang, Q., & Li, L. E. (2024). ViGoR: Improving Visual Grounding of Large Vision Language Models with Fine-Grained Reward Modeling. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2402.06118

[25] https://www.kaggle.com/datasets/soumikrakshit/lol-dataset

[26] Bray, N. W., Reilly, K. D., Villa, M. F., & Grupe, L. A. (1997). Neural network models and mechanisms of strategy development. *Developmental Review*, *17*(4), 525–566. https://doi.org/10.1006/drev.1997.04