An Unsupervised Approach to Removing Noise from the Facial Expression Recognition Dataset using Convolutional Autoencoders

Muhammad Sohel
Kennesaw State University
College of Computing and Software Engineering
Marietta, GA
msohel@students.kennesaw.edu

Mohammed Ahsan
Kennesaw State University
College of Computing and Software Engineering
Marietta, GA
mahsan3@students.kennesaw.edu

***Abstract*—Image denoising is a common task in image processing and computer vision. The task of denoising includes removal of varying noise within images, this paper aims to remove Gaussian noise distribution from the FER+ dataset. Our research questions are whether convolutional autoencoders are an effective means of removing noise from images and whether the choice of noise levels have any affect on the the performance of the autoencoders in terms of reconstructing images. Our approach to tackle these questions is a novel Convolution Autoencoder (CAE). The FER+ dataset will be utilized without added noise in training, then the varying noise levels will be added to the images during the testing phase. Mean squared error (MSE) and Structural Similarity Index (SSIM) will be used to evaluate the reconstructed images against the original input. Test results showed that our CAE was a viable way of removing noise, however more corrupted images proved to be a challenge for our model. Overall, this study showed that CAE's can be a viable solution for removing noise from corrupted images, especially in cases where the noise is independent of image contents.**

***Index Terms*—Convolutional Autoencoder, Image Denoising, Facial Expression dataset, Reconstruction, Deep Learning, Machine Learning, Residuals, Convolutional Neural Networks, Unsupervised Learning**

I. Introduction

Computer vision is one of the most popular fields within Computer Science at the moment due to the rapid increase in applications ranging from autonomous vehicles to disease identification in the medical field. In this realm, denoising plays a vital role by improving the quality and reliability of visual data. By removing unwanted noise and distortions from images, denoising algorithms enhance the clarity and detail of the visual information, making it easier for computer vision systems to accurately analyze and interpret scenes. This preprocessing step is essential for applications where precision is crucial, such as in medical diagnostics, where high-quality images are needed for accurate detection and diagnoses, or in autonomous driving, where clear visual inputs are critical for safe navigation and decision-making.

A. *Denoising Models*

The computer vision task of image denoising involves removing noise from an image, whether the noise is already part of the image or processed into it. [4] Noise in images can be due to various factors, such as poor lighting conditions, pixel imperfections, or transmission errors, and these often degrade the quality of the images. Denoising models address this problem by learning to distinguish between noise and the underlying features, ultimately producing a cleaner and potentially more accurate output. Denoising models have numerous practical applications, such as in photography and video processing to enhance image quality. The development of denoising models has seen significant advancements with the advent of deep learning techniques, making it possible to achieve impressive results even in highly noisy conditions.

Some of the most popular and benchmark denoising models include CGNET, SSAMAN, BM3D, and LLD. These models are usually evaluated and scored using peak signal-to-noise ratio (PSNR) and Structural Similarity Index (SSIM) to compare a reconstructed image when compared to an original not corrupted image. CGNET consistently performs at about 40 in PSNR and 0.96x in SSIM across different datasets. SSAMAN performs at about 39 in PSNR and 0.95x in SSIM across different datasets. LLD performance varies across different datasets, with their best PSNR being a 44.95 and a SSIM of 0.977.

There are many different methods that denoising models use to approach the task of image denoising and image reconstruction. Some of the classical methods of image denoising include spatial domain filtering, variational denoising methods, total variation regularization, and non-local regularization. Some of the more contemporary approaches include CNN-based denoising, MLP models, and deep learning-based denoising methods [6]. While we are not competing with these benchmark models, our goal is to understand their denoising task and create our own novel solution. We aim to utilize a CNN based autoencoder, which we will train to reconstruct images, then feed noisy images to evaluate how well it can denoise

and reconstruct.

B. Convolutional Autoencoder (CAE)

A Convolutional Autoencoder (CAE) is a type of neural network designed usually for unsupervised learning tasks, like image denoising, feature extraction, and dimensionality reduction [1]. CAEs are an extension of traditional autoencoders, which are also a type of neural networks used to learn a compressed representation of input data. The unique aspect of CAEs is their use of convolutional layers, which are particularly effective for processing image data due to their ability to preserve spatial hierarchies in the data. This is a known advantage of using CAEs over traditional autoencoders, their ability to handle high-dimensional data by leveraging the spatial structure within the data. A typical CAE consists of an encoder and a decoder portion. The encoder compresses the input image into a lower-dimensional latent space, capturing essential features while discarding noise and redundant information. The decoder then reconstructs the image from this compressed representation, ideally producing an output that closely resembles the original input. This process of encoding and decoding helps the network to learn useful features in an unsupervised manner, which can be valuable for our task in denoising images. Their ability to maintain the spatial structure of images while reducing noise and capturing essential features makes them an invaluable tool in modern image processing and computer vision applications.

C. Research Question

RQ1: Are autoencoders an effective means of removing noise from images?

RQ2:Does the choice of noise level affect the performance of an autoencoder in terms of reconstructing images?

Understanding the effectiveness of autoencoders in removing noise from images is crucial because it directly impacts their utility in various practical applications such as image denoising, data compression, and feature extraction. By answering **RQ1**, we can determine the viability of autoencoders as a tool for improving image quality in real-world scenarios, leading to enhanced performance in tasks like medical imaging, satellite imagery, and surveillance. Furthermore, exploring **RQ2** allows us to understand the robustness and adaptability of autoencoders to different noise conditions. This knowledge is essential for optimizing autoencoder architectures and training strategies, ensuring they can handle a variety of noisy environments and still produce high-quality reconstructions. Overall, answering these questions contributes to advancing the field of image processing and enhancing the practical applicability of autoencoders in diverse domains. Autoencoders are particularly powerful for such use cases as they can run fully unsupervised, requiring little to no user defined features. To answer these questions we first look at some releated work. Following this we discuss the proposed novel model architecture, going into details about the parameters involved and the approach this group has chosen to take. Finally the results of the tests are discussed, and next steps analyzed.

II. Related Work

Xie et al. [2] in their 2012 article examine the use of denoising autoencoders in order to remove noise from an image. Their work focused on stacking sparse denoising autoencoders in order to learn noise patterns of images during a pre-training step, in order to remove noise. Their work is set as a precursor to the techniques discussed later in this paper. Xie et al analyse stacks of individual autoencoder layers in a supervised manner, where noisy sets and clean sets are demarcated and passed separately as inputs. After each layer they passed information on both the noisy and clean data onto the next autoencoder layer. Their method relied on the model learning the features of both the image and the noise, and then removing the noise features from the image. Later in the paper they highlight some of the key strengths and differences of their approach, which performs well in inpainting, or removing text from images without a predefined mask denoting the text location. Some of the downsides identified by this team were that the supervised nature of the model they developed would mean that noise that the model had previously not encountered before could not be removed effectively. The process used by Xie et al. is similar to the approach taken in this paper, where autoencoders are used to remove noise, however there are distinct differences in the application. The model proposed in this paper uses convolutional autoencoders, which use convolutional layers in comparison to the approach taken in [2], which uses a more linear style of autoencoder framework. Additionally, this paper relies on both the testing and training being purely unsupervised, a distinct difference to [2], mitigating some of the key downsides identified in their model.

In the study of "Medical image denoising using convolutional denoising autoencoders" [3] Gondara et al. focused on image denoising, specifically when preprocessing in medical image analysis, addressing the challenge of noise in images from various medical imaging techniques like X-rays, Magnetic Resonance Imaging (MRI), Computer Tomography (CT), ultrasound etc.. While they look to denoise medical images, we aim to use a similar technique on a human facial expression dataset. They discussed different denoising methods that have been used extensively, such as models based on partial differential equations, wavelets, and non-local means. They propose that convolutional autoencoders, show promise in outperforming these conventional methods, with that limitation that deep learning models usually require large datasets and high computational resources. We as well also look to use a CNN with a large dataset for our proposal. The study proposed using a CNN autoencoder for denoising medical images using small sample sizes, demonstrating that even with limited data, these models can achieve high denoising performance. Two datasets, mini-MIAS (MMM) and a dental radiography database (DX), were used, with images resized to 64x64 for computational efficiency. Varying levels of Gaussian noise were added to the images. We aim to follow a similar

approach to adding Gaussian noise to our dataset. Their CNN AE architecture contained 5 convolution layers, 2 max pooling layers, and an upsampling layer. This model was run on 100 epochs, with a a batch size of 10. Experimental results showed that CNN DAE outperformed traditional methods, achieving mean SSIM scores of 0.81 for MMM and 0.88 for DX, compared to lower scores from median filters. Further experiments with Poisson noise confirmed the superior performance of CNN DAE over median filters and non-local means, especially as noise levels increased. Their approach of using multiple noise levels and evaluation with SSIM scores can be beneficial to our proposed model. Although this model showed that it faced challenges in model convergence at higher noise levels, the research demonstrated that even simple architectures could achieve significant denoising results, which is why we would like to make a more complex architecture hoping for better results. The study concludes that CNN DAEs are effective for medical image denoising with small datasets, challenging the notion that large datasets are necessary for deep learning models. They suggest future work should explore optimal architectures for small sample sizes, higher resolution images, and integrating other denoising methods like singular value decomposition for preprocessing before applying CNN autoencoders.

In another paper on "Noise Removal from the Image Using Convolutional Neural Networks-Based Denoising Auto Encoder" [5], they proposed a model for image denoising using a denoising autoencoder based on convolutional neural networks (CNNs) as well. We noticed a trend that lots of denoising models focus on medical images, which are usually smaller datasets. Although the denoisers have a similar goal to our paper, we aim to use larger datasets and shift our focus from the medical field. Their datasets were the Covid19-radiography-database and SIIM-medical-images set. The datasets include chest X-ray images and CT medical images, with images undergoing preprocessing, denoising through autoencoders, and further enhancement using CNNs. The proposed model is trained by introducing random Gaussian noise to the input images, with the goal of producing a noise-free original image as the output. The study proposed the very common evaluation techniques of Root Mean Squared Error (RMSE) and the peak signal-to-noise ratio (PSNR), for comparing auto encoders outputs and image comparison. We aim to use similar evaluation techniques that fit our model and network. The results show that the proposed method achieves lower RMSE and higher PSNR values compared to the traditional methods, indicating better noise reduction and image preservation. The research highlights the significance of using deep learning techniques, such as autoencoders and CNNs, for effective image denoising.

In a similar study of denoising capabilities performed in 2017, Zhang et al. propose the use of a single deep neural network model to handle multiple types of image corruption [7]. They proposed using DnCNN to tackle Gaussian noise using a residual based approach, Zhang et al. further claim that such an approach would be easily expandable towards a larger class of denoising problems, such as JPEG deblocking. We have a very similar thought process behind our application, however where Zhang et al. use a single deep convolutional network, we choose to use a convolutional autoencoder, which is essentially two CNNs in sequence. Furthermore, Zhang et al.make use of residual learning or identify functions to improve the training and evaluation processes. This process is very similar to the approach taken in this paper, as residual networks or skip steps tend to work well with preventing the vanishing gradient problem and allowing for deeper networks to learn effectively. [8] In their application Zhang et al. modified the existing and well documented VGG network as the starting point of their model and tweaked it to more effectively match or exceed the performance of BM3D, which they identify as the definitive state of the art denoising algorithm. Zhang Et al.'s paper overall follows many of the same processes taken by this document, wherein a denoising problem is solved using some form of convolutional neural networks, however the approaches differ in their details. The goal of this paper is to gauge the efficacy of the autoencoder style neural networks ability to remove noise from facial image datasets, as such a novel autoencoder framework was built to tackle various noise functions, in comparison to the study performed by Zhang et al. whose main goal was to develop a new state of the art in Gaussian denoising.

III. Datasets

A. FER+

The FER+ dataset is an enhanced, updated version of the original FER2013 dataset, with tweaks and adjustments to the limitations faced by the previous version. One of the main updates of this version is to the labeling, where the images were reassessed and relabeled for accurate representations of the depicted grayscale images of human facial expressions. A new label for contempt was also added within the label categories. The dataset contains about 35,800 grayscale images of human faces, each being 48 by 48 pixels. It is labeled across eight basic emotion categories: anger, disgust, fear, happiness, sadness, surprise, neutrality, and contempt. This updated version provides a more accurate benchmark for evaluating and developing FER systems, with many studies utilizing the dataset for their proposed models. While this dataset is mainly used for the task of FER, we plan to utilize it in our denoising autoencoder model. It provides us with a large and complex dataset with denoised images, where we can apply noise and then compare our output to the original image.

B. Preprocessing

For our proposed denoising model, several preprocessing steps are applied to the dataset to prepare the images for training with the autoencoder. This is done by utilizing PyTorch's built in transforms package, which "can be used to transform or augment data for training or inference of different tasks" [11]. We start by resizing all images to 96x96 pixels, ensuring a uniform input size for the neural network. Next we applied

a horizontal flip and random rotation of 10 degrees to the images, both providing variability and helps the model become more robust to different orientations of the input images. The images are then transformed into PyTorch tensors, the format needed for feeding into a PyTorch neural network. Lastly, the image pixel values of the images are normalized. They are normalized to range between -1 and 1, with parameters of mean and standard deviation both being 0.5.

After the augmentation and normalization of our images, we then split the dataset into training and testing sets. The sets are split randomly with 80 percent allocated for training and 20 percent for testing. The sets are then loaded onto dataloaders, to create iterable data loaders for both the training and testing sets, with a batch size of 64. These data loaders enable efficient mini-batch processing, which is practical for our model's generalization performance and computational efficiency.

IV. Approach

A. Architecture

The model proposed by this group comprises of a convolutional autoencoder. This autoencoder is comprised of an encoder and decoder linked in series. The encoder network, Figure 1, takes in the image tensor as the input, while outputting a lower dimensional version of the image to capture the higher dimensional features. The decoder on the other hand, Figure 2, takes these inputs and uses a series of transposed convolutional layers to progressively increase the spatial dimensions back to the original size while reducing the depth, reconstructing the image. The encoder portion of the network contains a total of 12 convolutional layers, whereas the decoder has 6. The relatively large amount of convolutional layers in the encoder creates a problem: in some early test cases the model was not learning at all regardless of learning rate or epochs. This could be seen in the loss function being pretty much static over the course of the training. The observation of the training loss not decreasing led us to believe that the gradients were not being updated in an accurate manner; an indication of the vanishing gradient problem. To combat this, a residual term or skip connection was added to the architecture. The residual helps in training deep networks by allowing gradients to flow through the network directly, alleviating some of the issue caused by gradients getting smaller and smaller. A residual block helper class, Figure 3, was implemented in our code containing two convolutional layers, batch norms, and ReLUs.

The helper class allows the model to learn residual functions with reference to the input, which helps mitigate the vanishing gradient problem and allows for training much deeper networks. The skip (or shortcut) connection directly adds the input to the output of the convolutional layers, ensuring that the model can still learn identity mappings. This design choice has proven to significantly improve training performance and accuracy in deep networks. Figure 4 and Figure 5 provide an understanding of training loss with and without the residual block. The impacts of the residual block can easily be seen in the performance of the loss function, the case with no residual (Figure 5) has an initial, steep, reduction in loss, followed by no changes in the loss function, essentially meaning the model is no longer being optimized. The training cases where residuals are implemented however, (Figure 4), shows a much more reasonable trend, depicting the exponential decay sought after in loss functions. Both of these graphs were generated over 30 epochs with a learning rate set to 0.0001. Although it appears that the loss functions converge much earlier than 30 epochs, that particular value was chosen because it offered a good compromise between the models ability to detect and remove noise, and the training loss trends exhibited in the figures. Combining the residual blocks with the encoder and decoder, we get the core of the network architecture. The following is a detailed layout of the inputs and outputs of each layer in the network:

Encoder:

1) ResidualBlock(3, 32): Takes a 3-channel input image (e.g., RGB) and outputs 32 channels with residual connections, preserving spatial dimensions (96x96) using a stride of 1.
2) Conv2d(32, 64, 3, stride=2, padding=1): Reduces spatial dimensions to 48x48 while increasing depth to 64 channels.
3) ResidualBlock(64, 128): Outputs 128 channels, preserving 48x48 dimensions.
4) Conv2d(128, 256, 3, stride=2, padding=1): Reduces spatial dimensions to 24x24 while increasing depth to 256 channels.
5) ResidualBlock(256, 512): Outputs 512 channels, preserving 24x24 dimensions.
6) Conv2d(512, 1024, 3, stride=2, padding=1): Reduces spatial dimensions to 12x12 while increasing depth to 1024 channels.
7) ResidualBlock(1024, 2048): Outputs 2048 channels, preserving 12x12 dimensions.
8) Conv2d(2048, 4096, 3, stride=2, padding=1): Reduces spatial dimensions to 6x6 while increasing depth to 4096 channels.

Decoder:

1) ConvTranspose2d(4096, 2048, 3, stride=2, padding=1, output padding=1): Increases spatial dimensions to 12x12 and reduces depth to 2048 channels.
2) ConvTranspose2d(2048, 1024, 3, stride=2, padding=1, output padding=1): Increases spatial dimensions to 24x24 and reduces depth to 1024 channels.
3) ConvTranspose2d(1024, 512, 3, stride=2, padding=1, output padding=1): Increases spatial dimensions to 48x48 and reduces depth to 512 channels.
4) ConvTranspose2d(512, 256, 3, stride=2, padding=1, output padding=1): Increases spatial dimensions to 96x96 and reduces depth to 256 channels.
5) ConvTranspose2d(256, 64, 3, stride=1, padding=1): Keeps spatial dimensions at 96x96 and reduces depth to 64 channels.
6) ConvTranspose2d(64, 3, 3, stride=1, padding=1): Out-

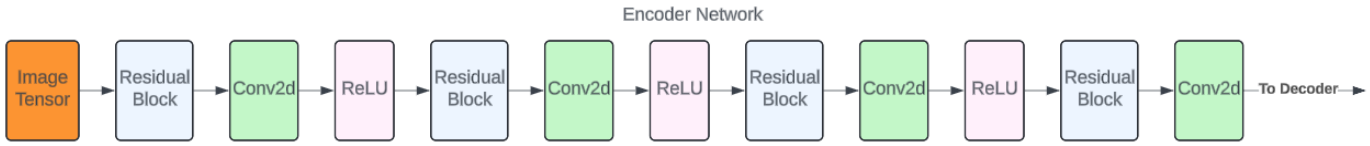

Fig. 1. Encoder Network

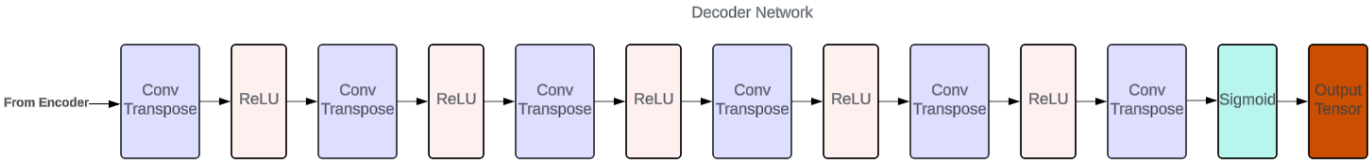

Fig. 2. Decoder Network

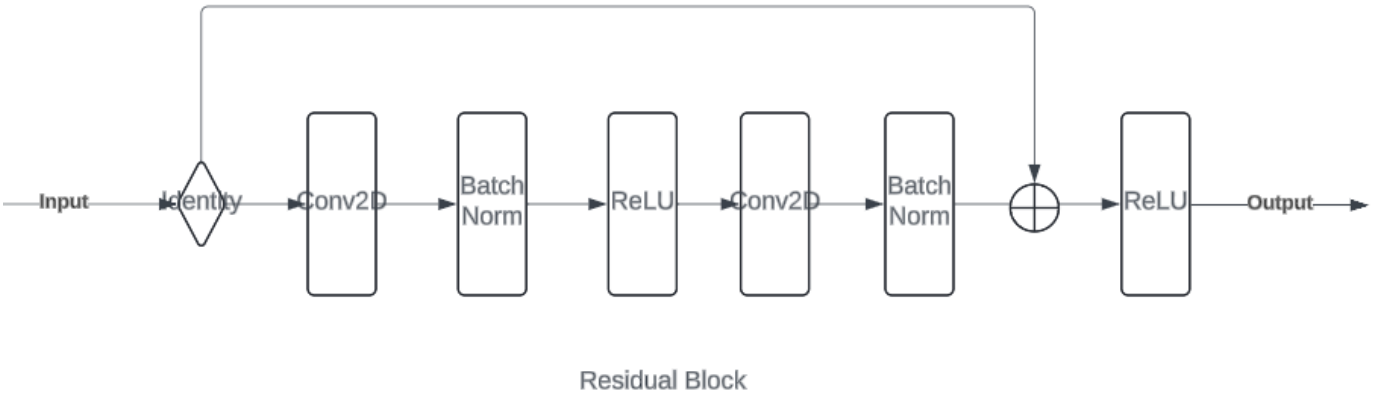

Fig. 3. Layout of a Residual Block for the Neural Network

puts a 3-channel image with the same 96x96 spatial dimensions.

B. Loss Function

Mean Squared Error (MSE) loss is a common loss function used in regression tasks and neural networks, including denoising autoencoders. It measures the average squared difference between the predicted values and the actual values, [10] as seen in Equation 1 below. In terms of autoencoders, MSE loss quantifies the difference between the input images and the reconstructed clean images produced by the autoencoder. This value assists in determining how well our autoencoder is learning and able to reconstruct images. The goal is to always minimize the MSE loss during training, which is why having effective learning parameters is necessary for reconstruction. This is why we chose to use the Adam optimizer, it combines the advantages of two other extensions of stochastic gradient descent (SGD), Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam is particularly well-suited for problems involving sparse gradients and noisy data, making it a popular choice for training deep learning models, including autoencoders. [9]

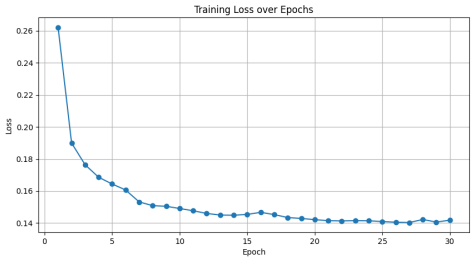

Fig. 4. Training Losses with Residuals

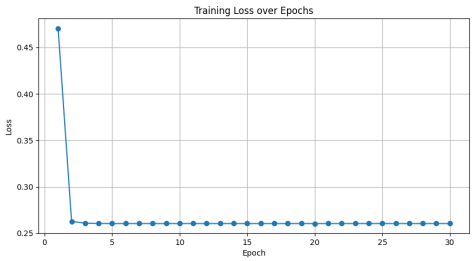

Fig. 5. Training Losses with no Residuals

$$l(x, y) = L = (l_1, \ldots, l_N)^T, l_n = (x_n - y_n)^2,$$

Equation 1: PyTorch MSE Loss Equation [10]

C. Parameters

Some of the major parameters for the proposed model are highlighted above. We will be running tests on samples of 4K images at a time, with around 3,200 used for training and 800 used for testing. The learning rate was set to 0.0001 after some testing with various other values, Values lower took too long to converge, whereas values larger had issues with the loss function updating sequentially. The best results were observed when using 30 epochs. Runs with longer epochs had diminishing returns; they took significantly more time, with not a lot of improvements in denoising accuracy. Models trained over a lower number of epochs tended to be have more grainy and poorer representations of the input datasets.The code for the system was developed in Python, using the pyTorch library. The model was optimized for CUDA based processing, and trained on a GeForce RTX 3080 Ti GPU. The ADAM optimizer was used on both the training and testing cases.

D. Training & Testing

The training of our autoencoder involves a iterative process with the goal of minimizing the reconstruction error between the input images and the corresponding outputs. Our model is trained on the training set over 30 epochs, with batch size of 64 in our dataloader. The batches are fed into the model to facilitate efficient processing and gradient computation. The Adam optimizer resets at the beginning of each iteration to clear any accumulated gradients. The model then processes the input images, generating reconstructed outputs. The reconstruction error, measured by the MSE loss function [10] quantifies how well the model has learned to reconstruct the input images. The running loss accumulates the loss values for the current epoch, weighted by the number of images in each batch, to provide a comprehensive measure of the model's performance over the entire training dataset. The running epoch loss is calculated by averaging the running loss over the total number of images. This value is recorded and used to monitor the model's progress and convergence over time 4. By iterating through this process, the autoencoder learns to efficiently encode and decode the input data, ultimately minimizing the reconstruction error and producing reconstructed output images.

With our now trained CAE model, we look to evaluate its performance on unseen noised images to assess its generalization capability and effectiveness in reconstructing denoised images. We start by iterating over our testing set and adding varying amounts of Gaussian noise to each of the images to simulate the noisy conditions. Gaussian noise, is a type of statistical additive noise that can be used to distort images by following a Gaussian (normal) distribution. Each noise is statistically independent of others because each are drawn from the same Gaussian distribution [13]. Our process of adding noise involves generating random values from a Gaussian distribution and adding them to each pixel of the original image based upon a noise factor, as seen in equation 3 below. The noisy images are generated by adding random noise scaled by Gaussian noise factor(s) (0.0, 0.15, 0.3, 0.5) to the original images, followed by clamping to ensure the pixel values remain within a valid range of 0 to 1. The noisy images are fed into the autoencoder, which processes them to produce reconstructed outputs. This theoretical approach to testing highlights the importance of evaluating the model's ability to maintain its performance outside the training environment. We will next evaluate our model, mainly by comparing the reconstructed outputs to the original images, so we can gauge the success of the autoencoder in learning the underlying clean image features and removing noise effectively, thereby validating its practical utility.

$$O(x, y) = S(x, y) + e(x, y)$$

Equation 2: Gaussian Noise [13]

E. Evaluation

The evaluation of an autoencoder involves quantitatively assessing its ability to reconstruct images from noisy inputs. The two primary metrics, Mean Squared Error (MSE) and Structural Similarity Index Measure (SSIM), are employed to capture different aspects of reconstruction quality. Similar to how MSE was used when training our model, it will be employed again here to see the loss between the reconstructed outputs and the original inputs. SSIM scores will also be calculated by comparing the reconstructed outputs with the original inputs. SSIM scores are calculated by comparing the local patterns of pixel intensities that have been normalized for luminance, contrast, and structure [12]. The overall SSIM index is a multiplicative combination of those three terms, the luminance term, the contrast term, and the structural term, all which can be seen in equation 2 below.

$$SSIM(x, y) = [l(x, y)]^a.[c(x, y)]^b.[s(x, y)]^y$$

Equation 3: SSIM Equation [12]

V. Results

The model is evaluated and 10 random images are pulled from each test case. Figures 6 to 9 are the results of the test cases evaluated by the model. Each figure depicts a raw image, the image with noise added in, and the output image from the model. Each of the figures represents a different noise factor used to corrupt the input image. The results are visually

apparent, lower noise factors yield better outputs. Figure 9, for instance exhibits distinct visual distortions on each of the images, similar to polka-dots. The reconstructed images for the 0.5 test case, Figure 9 show that the autoencoder struggled to detect the original features within the image due to the extreme noise factor. The speckled appearance is greatly reduced in Figure 8, which portrays a more realistic representation of noise as it would be encountered organically, as opposed to this synthetic test where noise is overlaid onto an image based on mathematical functions. Although the reconstructed image still displays some signs of anomalies, the output representations are drastically improved when compared to those of the 0.5 noise factor case. Output quality further improves in Figure 7, which depicts a lighter level of noise overlaid on the image. The reconstructed images for this case exhibit little to no signs of image processing when compared to their original images, with possibly the upside of revealing some of the features hidden by low illumination in the original image. These hidden features are revealed both due to the nature of the autoencoder, and the normalization techniques utilized when preparing the images for processing. Figure 6, finally, depicts the raw ability of the autoencoder to construct an image from an input. No noise is applied in this test cases, it serves as a control, proving the baseline efficacy for the developed model.

0.0 Factor	0.15 Factor	0.3 Factor	0.5 Factor
0.5491	0.3346	0.3686	0.2056
0.4162	0.2391	0.1387	0.0824
0.46466	0.51529	0.31888	-0.019
0.4817	0.27837	-0.0264	0.248
0.26833	0.19295	0.14627	0.058
0.52598	0.357389	0.24110	0.0164
0.77343	0.49135	-0.0226	0.047
0.5701	-0.0621	0.2027	-0.025
0.7800	0.0912	0.5211	0.083
0.1680	0.4597	0.2586	0.1141
AVG= 0.49975	AVG= 0.289785	AVG= 0.2147	AVG= 0.0811

TABLE I
AVERAGE SSIM OF 10 RANDOM SAMPLES WITH VARYING NOISE FACTORS

0.0 Factor	0.15 Factor	0.3 Factor	0.5 Factor
MSE= 0.0063	MSE= 0.010	MSE= 0.0096	MSE= 0.01

TABLE II
MSE TESTING LOSS OF VARYING NOISE FACTORS

Our quantitative evaluation metrics of SSIM and MSE are shown in Tables I and II. We obtained 10 random samples from the evaluation of our testing set, using SSIM and MSE to compare the original input images with the reconstructed output images. The numerical data further validate what was seen from the output images in Figures 6 to 9 . As the Gaussian noise factor increases from 0.0 to 0.5, the mean SSIM score tends to decrease from 0.49975 to 0.0811. SSIM scores range from 0 to 1, with 1 being a perfect structural similarity between two images. Although the model we created performs poorly when compared to the current state of the art models such as CGNET and SSAMAN, the goal of this paper was not to develop a new state of the art, but rather to explore a novel avenue of solving the problem of denoising. Interestingly some values of SSIM are negative, this is thought to be caused by the formulation of the SSIM scores. The output images seemed brighter and contain more distinguishable features than the input images. The negative value here is a function of luminance, contrast, and structure, in these particular cases with negative values for SSIM, it is clear that the output image has much higher luminance, and more features (structure) than the input. The MSE values also support these observations, with the lowest MSE of 0.0063 at 0.0 noise factor, indicating minimal deviation from the original. This low values is good indicator of our models ability to reconstruct images based on an input. As the noise factor rises from 0 to 0.15, 0.3, and 0.5, the MSE increases from 0.0063 to 0.01, 0.0096, and 0.01, respectively. The data contained in Table II represent the numerical differences between the raw pixel data in the images, these values are relatively easy to compute, and serve as a good indication of the models ability to match input and output images. The values in Table I however come from a much more nuanced metric, one that tries to mesh better with human perception, SSIM places greater importance in features lining up correctly rather than on raw pixel data, which may or may not be easily visualized by humans. The interplay between SSIM and MSE highlights the balance between noise introduction and the preservation of image integrity. Despite some variability among individual test samples, the overall trend demonstrates that higher noise factors leads to greater reductions in image reconstruction quality.

VI. CONCLUSION

Image denoising remains a critical preprocessing task within computer vision, significantly enhancing the accuracy and reliability of various applications such as autonomous driving and medical diagnostics. This paper highlights the importance of Convolutional Autoencoder (CAE) models and their ability to denoise images. By leveraging CAEs, our research demonstrates a powerful approach to unsupervised learning tasks, emphasizing the effectiveness of CAEs in maintaining spatial hierarchies and capturing essential features within images. Our focus on using a CAEs to reconstruct and denoise images, tested on the FER+ dataset, aligns with contemporary trends and methods in the field of computer vision, such as in security applications. The comparative analysis with related works demonstrates our unique approach of this proposed model, particularly in handling high-dimensional image data, CAE architecture, and training methodology. Our dataset selection included a large and diverse collection of images, with each image containing a much higher amount of features compared to previous studies. This is due to our dataset containing facial expressions features, which are much more nuanced by nature. In addition, we developed a novel architecture for the reconstruction of images. This involved creating an autoencoder from scratch containing multiple convolutional layers, and utilizing residuals. The addition of residuals greatly improved model accuracy as well as convergence time. Lastly,

Fig. 6. Images with 0 Gaussian noise factor

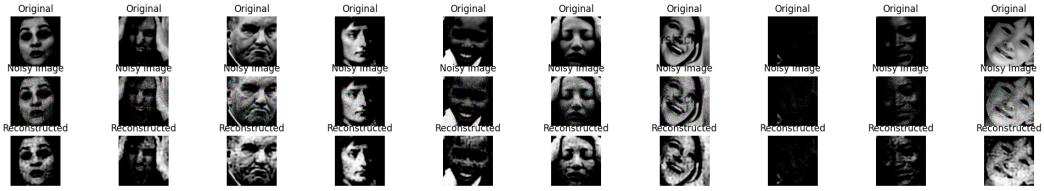

Fig. 7. Images with 0.15 Gaussian noise factor

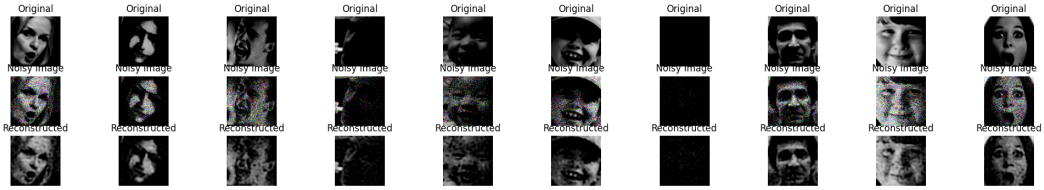

Fig. 8. Images with 0.3 Gaussian noise factor

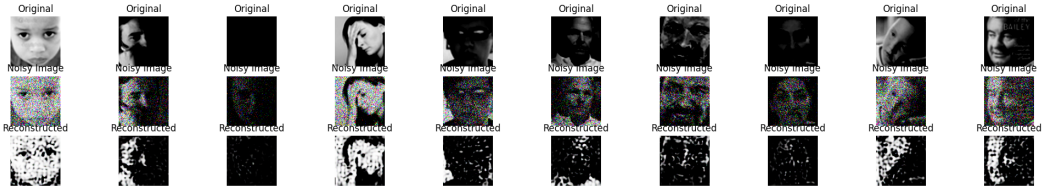

Fig. 9. Images with 0.5 Gaussian noise factor

our approach to training and testing our model differed from other denoising models, where they would train their autoencoder to learn the added noise before testing. Our approach was to have our model already trained, then introduce the noise during the testing phase to see how well our model can extract underlying features and reconstruct images. The research questions at hand included whether autoencoders are an effective means of removing noise from images. An additional question was whether varying noise levels affect the models ability to reconstruct. Our CAE model was effective in removing noise from input images contaminated with Gaussian noise distributions, with noise factors ranging from 0.15 to 0.5, proving the viability of CAEs as a denoising solution. The models ability to remove noise independent of image features is a promising sign of the models ability to react to corrupted images in practice; where noises are more predictably tied to features. The best results came from noise factors between the 0.15 and 0.3 cases, with the 0.5 case removing noise, but displaying significant visual distortions. The results showed the effectiveness of our model in recreating and denoising images with smaller noise factors, but struggling when the noise factor was over 0.5.Further model development can alleviate some of the distortions seen in the test results, and provide far better reconstructions of input images. This improvement in the model could arise from better optimization, a deeper neural network, or more available computational resources. Additional expansions on this series of tests could the introduction of other types of noise, such as Poisson or Salt and Pepper distortions. Overall, this study contributes to the ongoing advancements in image reconstruction and denoising models, showcasing the potential of convolutional autoencoders to improve image quality and support critical

applications in computer vision.

REFERENCES

[1] "Implement Convolutional Autoencoder in PyTorch with CUDA," *GeeksforGeeks*, Jul. 31, 2023. https://www.geeksforgeeks.org/implement-convolutional-autoencoder-in-pytorch-with-cuda/ (accessed Jul. 21, 2024).

[2] J. Xie, L. Xu, and E. Chen, "Image Denoising and Inpainting with Deep Neural Networks," Neural Information Processing Systems, 2012. https://proceedings.neurips.cc/paper/2012/hash/6cdd60ea0045eb7a6ec44c54d29ed402-Abstract.html

[3] L. Gondara, "Medical Image Denoising Using Convolutional Denoising Autoencoders," IEEE Xplore, 2016. https://ieeexplore.ieee.org/abstract/document/7836672/

[4] "Papers with Code - Image Denoising," paperswithcode.com. https://paperswithcode.com/task/image-denoisings (accessed Jul. 21, 2024).

[5] Y. FAROOQ and S. SAVAŞ, "Noise Removal from the Image Using Convolutional Neural Networks-Based Denoising Auto Encoder," *Journal of Emerging Computer Technologies*, vol. 3, no. 1, pp. 21–28, Mar. 2024, doi: https://doi.org/10.57020/ject.1390428.

[6] Fan, L., Zhang, F., Fan, H. et al. "Brief review of image denoising techniques". Vis. Comput. Ind. Biomed. Art 2, 7 (2019). https://doi.org/10.1186/s42492-019-0016-7.

[7] K. Zhang, W. Zuo, Y. Chen, D. Meng and L. Zhang, "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising," in IEEE Transactions on Image Processing, vol. 26, no. 7, pp. 3142-3155, July 2017, doi: 10.1109/TIP.2017.2662206.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv.org, Dec. 10, 2015. https://arxiv.org/abs/1512.03385

[9] Jason Brownlee, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," *Machine Learning Mastery*, Jul. 02, 2017. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[10] "MSELoss — PyTorch 1.7.0 documentation," *pytorch.org*. https://pytorch.org/docs/stable/generated/torch.nn.MSELoss.html

[11] "torchvision.transforms — Torchvision master documentation," *pytorch.org*. https://pytorch.org/vision/stable/transforms.html

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: https://doi.org/10.1109/tip.2003.819861.

[13] *Github.io*, 2024. https://takmanman.github.io/2020/05/23/gaussian-noise-gaussian-filter.html