Emotion-based Style Transfer

Dishant Padalia
University of Massachusetts Amherst
Amherst, MA
dpadalia@umass.edu

Rahul Saxena
University of Massachusetts Amherst
Amherst, MA
rahulsaxena@umass.edu

Abstract

We introduce an approach by integrating emotion recognition with style transfer in image processing using convolutional neural networks (CNNs). It traces the evolution of style transfer techniques and a system where styles are dynamically applied based on the emotional context of images. Utilizing a specially designed 21-layer CNN for emotion classification and a unique neural style transfer model, the study conducts experiments on the Expression in-the-Wild (ExpW) dataset. The results showcase the potential of this method in creating visually and emotionally rich images, achieving significant advancements in the field of neural style transfer and emotion recognition.

1. Introduction

The field of style transfer has undergone significant development since it began in the late 1990s when it mostly concerned fundamental image processing tasks. This journey has been marked by considerable improvements, leading up to the current condition in which the integration of emotion with style transfer represents a new frontier in image generation.

Initially, style transfer techniques were limited to signal-processing methods and filter applications, prevalent since the 1980s. These early methods focused on altering the aesthetic aspects of images using available digital manipulation tools. However, a landmark development occurred in 2015 with the pioneering work of Gatys et al. In their groundbreaking study, they introduced a novel concept of neural style transfer. This approach utilized convolutional neural networks (CNNs) to apply artistic styles to various content images, representing a significant leap in the capabilities of style transfer techniques. It allowed for the creation of visually appealing compositions by combining the content of one image with the style of another, thus expanding the aesthetic potential of images.

Building on these advancements, our research presents a novel two-fold technique for expanding CNN capabilities beyond mere style transfer. We intend to incorporate recognition of emotions into the process, giving the style transfer technique a new depth. Our solution involves recognizing emotions in input images using CNN-based image classification methods. We aim to dynamically apply styles that enhance the natural emotions in an image by identifying the emotional context of the image. This approach seeks to produce compositions that are not only visually appealing but also emotionally rich. So, our goal is to make images that are not just pretty but also capture the right emotion. This could make pictures a lot more interesting and engaging for everyone.

2. Related Work

The field of style transfer in image processing has seen remarkable contributions, with some key works significantly shaping its development. A groundbreaking paper in this area is "A Neural Algorithm of Artistic Style" by Gatys et al., published in 2015. This paper was pivotal, introducing a method for artistic style transfer that blends the content of one image with the style of another. The process involves minimizing content and style reconstruction loss using features extracted from a pre-trained convolutional neural network (CNN). This concept, building on their earlier work in texture synthesis, set the stage for future research in Neural Style Transfer (NST). This approach used a pre-trained CNN to extract content and style features from images. By optimizing a generated image to match the content features of one image and the style features of another, they succeeded in creating high-quality stylized images.

Another significant advancement was made by Johnson et al. in 2016. In their paper "Perceptual Losses for Real-Time Style Transfer and Super-Resolution," they proposed a method for real-time neural network style transfer using perceptual losses. These losses incorporate high-level features from pre-trained CNNs. By employing perceptual loss functions to train feed-forward networks for the style transfer task, they addressed computational challenges in Gatys et al.'s method and achieved substantially faster results.

Subsequently, Gatys et al. themselves expanded on their

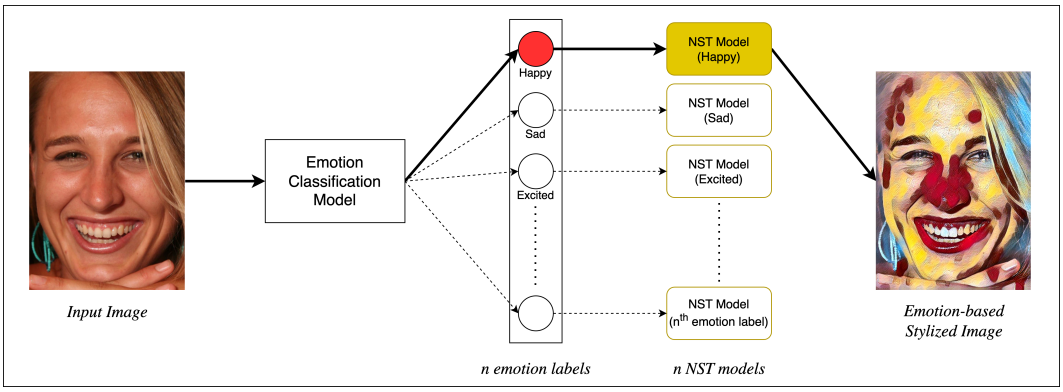

Figure 1. System Architecture

original framework, introducing a multi-scale approach that allowed for more nuanced handling of style and content at different abstraction levels, leading to more aesthetically pleasing stylized images. Huang et al. tackled the limitation of fixed styles in neural style transfer by proposing adaptive instance normalization. This method enables arbitrary style transfer, allowing for the application of diverse artistic styles to content images in real time. Li et al.'s work, "Universal Style Transfer via Feature Transforms," presented a universal style transfer technique, offering greater control over both style and content.

Convolutional Neural Networks (CNNs) have demonstrated significant prowess in tasks such as image processing and emotion recognition since their introduction in the late 1990s. However, their early applicability was constrained by limited training data and computational power. The landscape shifted markedly in the 2010s, witnessing a substantial increase in computational capabilities and the availability of expansive datasets. These developments have elevated CNNs into highly effective tools for tasks including feature extraction and emotion recognition. Numerous techniques have been devised to enhance the performance of CNNs. Recent developments and extensive research underscore CNNs as extremely favorable tools for addressing challenges in image processing, pattern recognition, and feature extraction.

The Facial Expression Recognition 2013 (FER 2013) dataset, introduced at the International Conference on Machine Learning (ICML) in 2013, has evolved into a benchmark for evaluating model performance in emotion recognition. Various CNN architectures have exhibited great performance, achieving classification accuracies ranging from 65% to 72.7%. Ensemble methods, such as the work by Liu et al., demonstrated improved performance by ensembling three CNNs, resulting in a 2.6% accuracy enhancement.

Researchers have also proposed novel architectural modifications to enhance performance. Shi et al. introduced the amend representation module (ARM), substituting the pooling layer, and achieved an accuracy of 71.38%. Tang et al. replaced the softmax layer with a support vector machine in a deep neural network, achieving a classification accuracy of 71.2%.

Pramerdorfer et al. conducted a comprehensive comparison of three prominent CNN architectures—VGG, Inception, and ResNet. Their findings reveal that VGG outperforms with an accuracy of 72.7%, followed by ResNet at 72.4%, and Inception at 71.6%. Ensembling eight CNNs boosted the performance by 2.5%.

3. Proposed Approach

3.1. System Architecture

As outlined in our system diagram (Figure 1), we first feed the input image into our emotion detection model, which labels the image with an emotion. After identifying the emotion, we send the image to a style transfer model trained specifically for that emotion. This model generates the final image with the new style, as depicted in the figure. We have trained seven style transfer models, i.e., one model for each emotion label: happy, sad, fear, neutral, surprise, disgust, and angry.

3.2. Classification Model

In the past decade, convolutional neural networks (CNNs) have shown remarkable effectiveness in image classification tasks, primarily due to their architecture, which mirrors the human visual perception system. Unlike traditional neural networks, CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images. This learning is facilitated through convolutional layers, which apply filters to capture local patterns such as edges, textures, and shapes. These filters can detect features at various scales and locations in an image, making CNNs highly efficient in dealing with the spatial and hierarchical nature of visual data. Another crucial aspect of CNNs is the pooling layers, which reduce the spatial size

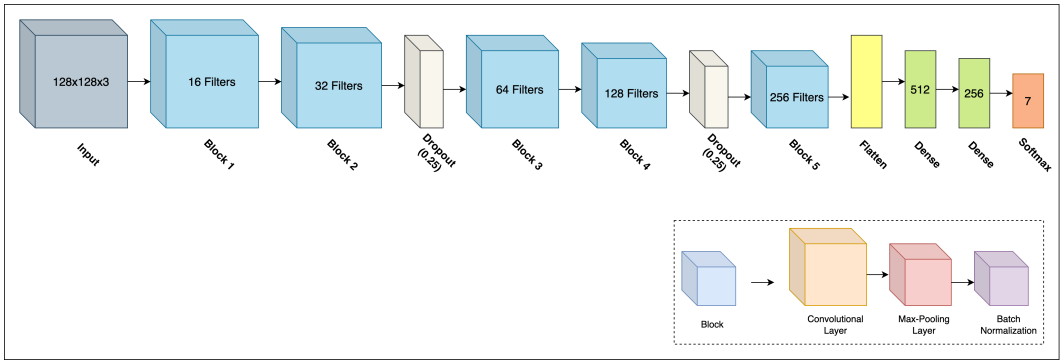

Figure 2. System Architecture

of the representation, lowering the amount of computation and weights in the network. This not only improves computational efficiency but also contributes to translation invariance. To leverage these advantages of CNNs, we have developed a ConvNet-based architecture for the multi-class emotion classification task. Our design is a 21-layer CNN network, including convolutional, pooling, batch normalization, dropout, and fully-connected layers.

Initially, we built a block of layers, reused multiple times in the model architecture, called the convolutional block. This block consists of three layers: a convolutional layer, a batch normalization layer, and a max pooling layer. The first layer is a 2D convolutional layer, followed by batch normalization that normalizes activations from the previous layer, enhancing the model's stability. A 2x2 max pooling layer follows, reducing the spatial dimensions (height and width) of the input volume for the next layer.

All the 2D convolutional layers in the model have a filter size of (3x3), a stride of 1, and no padding. These layers use the ReLU (Rectified Linear Unit) activation function. The number of filters in the first convolutional layer is 16, which doubles in subsequent convolutional layers. The model begins with a convolutional block that takes an input size of (128x128x3).

The model continues with a series of convolutional blocks, each comprising a 2D convolutional layer with progressively increasing filters—32, 64, 128, and finally 256. Like the first, these layers utilize the ReLU activation function and include batch normalization steps. The second and fourth convolutional blocks incorporate dropout layers with a rate of 0.25, to prevent overfitting by randomly deactivating a portion of neurons during training.

The network transitions from convolutional to dense layers after these blocks. A flattening layer converts the 2D feature maps into a 1D feature vector, allowing the outputs of the convolutional network to feed into fully connected layers. This is followed by two dense layers with 512 and 256 neurons, respectively, critical for synthesizing the learned features for classification. Each layer applies the ReLU activation function to maintain non-linear properties. Finally, the output of the last fully connected layer feeds into a dense layer with seven neurons, equal to the number of classes. This layer uses a softmax activation function, suitable for multi-class classification problems as it converts the outputs into a probability distribution over the predicted output classes.

For training the described model, we used the Adam optimizer for its efficiency in handling sparse gradients and adaptively adjusting learning rates. The loss function employed is sparse categorical cross-entropy, ideal for multi-class classification problems where each class is mutually exclusive.

3.3. Neural Style Transfer

Neural Style Transfer (NST) involves manipulating images to adopt the visual style of another image. This technique employs neural networks for image transformation. The core concept of NST is to blend two images: a content image and a style reference image, resulting in an output image that retains the content of the first image but is rendered in the artistic style of the second.

Neural Style Transfer is implemented using an optimization technique that adjusts the output image to match the content statistics of the content image and the style features of the style reference image. These features are extracted using a convolutional neural network. For example, the network's lower layers might capture basic features like edges and textures, while higher layers can represent more complex features. NST was first introduced in the paper "A Neural Algorithm of Artistic Style" by Leon Gatys et al. in 2015. This foundational work utilized the VGG-19 network architecture, pre-trained on image recognition tasks, to separate and recombine the content and style of images.

Traditionally, image transformation tasks such as denoising, super-resolution, and colorization were approached using feedforward convolutional neural networks trained in a supervised manner with a per-pixel loss function. This method measures the difference between the output and

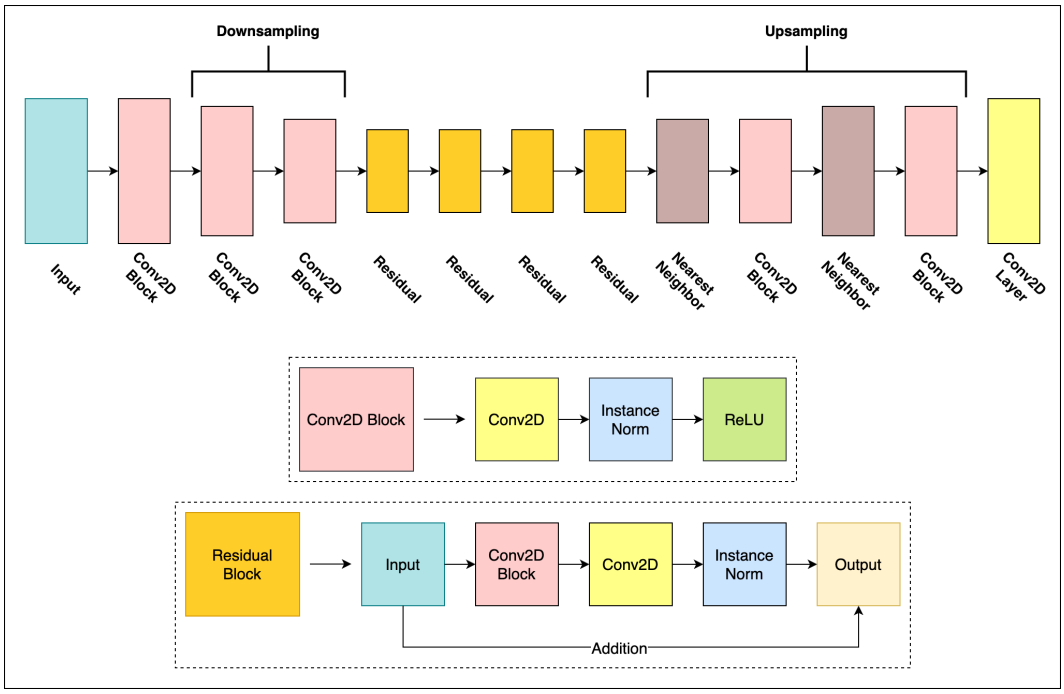

Figure 3. System Architecture

ground-truth images pixel by pixel. While efficient, this approach does not always capture perceptual differences between the output and reference images well. Style transfer by Leon Gatys et al. achieved high-quality results using perceptual loss functions. Perceptual loss uses high-level image feature representations extracted from pre-trained convolutional neural networks. The drawback of this approach is its slow inference, as it requires solving an optimization problem. During inference, the noise image is iteratively updated by calculating the gradients of the loss function and adjusting the image to minimize this loss, a time-consuming process.

J. Johnson et al. devised a method that combines the aforementioned approaches by training a feed-forward transformation network. Rather than using per-pixel loss functions that depend only on low-level pixel information, they trained the network using perceptual loss functions that rely on high-level features from a pre-trained loss network. This approach overcomes the shortcomings of per-pixel loss by using perceptual loss and also speeds up inference time, as no optimization iterations have to be run for the input image.

Our style transfer model consists of a two-stage neural network: an image transformation network and a loss network. The image transformation network is a deep residual neural network that transforms the input image into output images. The loss network is a VGG16 model pre-trained on ImageNet weights and is used for defining the perceptual loss between the content and style images.

The input and output images of the image transformation network are of the shape 3 x 256 x 256. Additionally, at test time, since this model is fully convolutional, it can be applied to images of any size. This network consists of 16 convolutional layers. No pooling layers are used in this model; instead, stride-2 and stride-1/2 convolutions are used to downsample and upsample the input, respectively. We have used instance normalization instead of batch normalization, as it yields better results and aids in faster stylization. First, we define a Conv2D block comprising three layers: a convolutional layer, instance normalization, and ReLU activation. Given the input image, a stride-1 Conv2D block is applied, followed by 2 stride-2 Conv2D blocks for downsampling. Subsequently, 5 residual blocks are applied, each containing 5 layers: 1 Conv2D block, 1 convolutional layer, and 1 instance normalization layer. The input is then added to the output of the instance normalization layer to build the residual connection. After the residual blocks, there are 2 sets of upsampling layers using nearest neighbors interpolation with convolutional layers. Finally, a stride-1 convolutional layer is applied to generate an output image of size 3 x 256 x 256.

The loss network consists of a VGG16 model pre-trained on ImageNet. For the content and style images, we take the feature maps from the output of the ReLU activation layer. Based on these feature maps for the content and output image, we calculate the content loss. Secondly, we calculate

the gram matrix for the style and output images using the feature maps, and then calculate the style loss. Our objective is to minimize the total loss, which is the sum of the content and style losses.

4. Experiments

4.1. Dataset

We conducted all our training and experiments on the Expression in-the-Wild (ExpW) dataset. The ExpW dataset contains 91,793 faces, each manually labeled with expressions. Every face image is annotated as one of the seven basic expression categories: 'angry,' 'disgust,' 'fear,' 'happy,' 'sad,' 'surprise,' or 'neutral'.

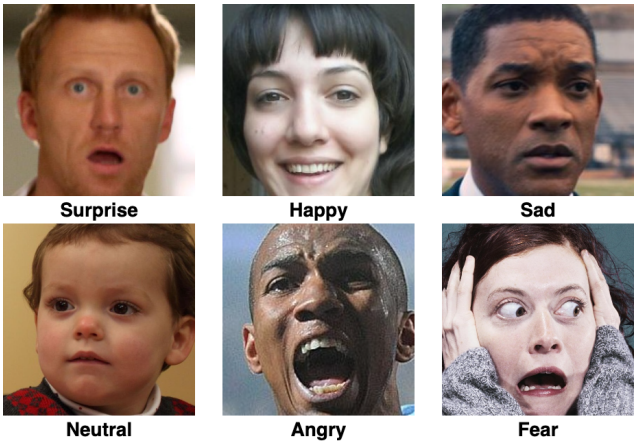

Figure 4. System Architecture

4.2. Data Preprocessing

The images in the dataset varied in size, so we first resized each image to a uniform shape of 128x128. As indicated in the table, the raw dataset exhibited a significant imbalance among the classes, leading to a bias in the classification model towards classes with more data points. To address this issue, we implemented data augmentation to equalize the number of images across classes. Specifically, we applied random rotations to the images from classes with fewer data points, aiming to bring the number of images in each class to approximately 10,000. The distribution of the dataset after augmentation is presented in Table 1. A sample of the augmented output is shown in the figure.

4.3. Implementation Details

For the emotion classification model, we divided the preprocessed dataset into train, validation, and test sets with ratios of 70:20:10, respectively. The model was trained for 50 epochs using an Adam optimizer with a learning rate of 0.001 and a batch size of 16. Accuracy and loss were the metrics used to evaluate the model's performance.

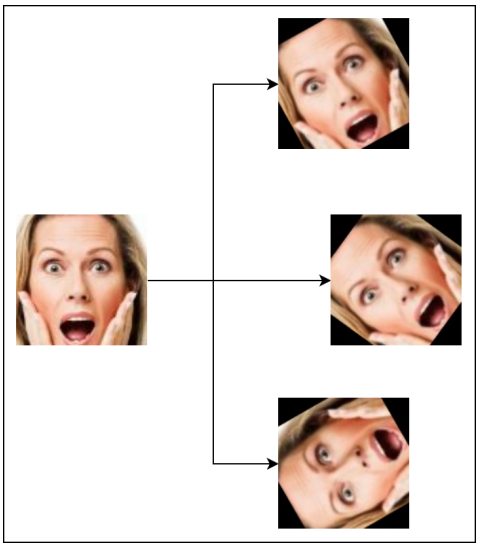

Figure 5. System Architecture

In training the neural style transfer (NST) model, we utilized the ExpW dataset. We selected 7 different artistic styles for training the 7 NST models, corresponding to each emotion class. Some of these styles were inspired by the works of painters such as Vincent van Gogh and Edvard Munch, while others were generated using DALL·E (an AI image generative model by OpenAI). For the model training, we set the content image weight to 1.0 and the style image weight to 400,000. Each of the NST models was trained for only 5 epochs due to limited computational resources and extensive training time. For training the loss network (VGG16) of the NST model, we could use the feature map of the content and style image from any of these four activation output layers: relu1_2, relu2_2, relu3_3, and relu4_3. We experimented with the NST model for the 'happy' emotion using each of these activation layer outputs separately and observed that relu2_2 produced the most stable output, effectively incorporating features of both content and style images. Therefore, for training all other NST models, we used relu2_2.

All our training and experiments were conducted using PyTorch (for the NST model) and TensorFlow (for the Emotion classification model) on an Nvidia Tesla P100 GPU. Our emotion classification model took approximately 30 minutes to complete 50 epochs of training, while each Neural Style Transfer model required about 5.5 hours for 5 epochs of training.

5. Results and Discussion

The emotion classification model achieved a training accuracy of 96.53% and a training loss of 0.1082, along with a validation accuracy of 79.34% and a validation loss of

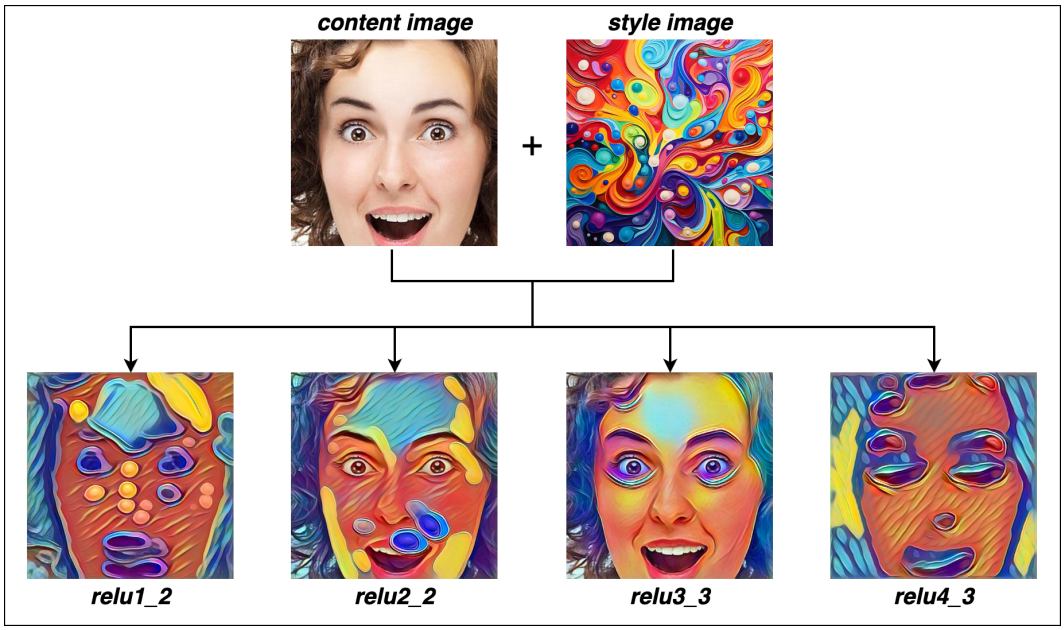

Figure 6. System Architecture

1.0734. The model reached a test accuracy of 77.51%. Several factors might explain why the model did not achieve higher accuracy. Emotions are complex and often subtle, posing a challenge for the model to differentiate between similar emotions or detect nuanced expressions. Additionally, the ExpW dataset's manual labeling introduces subjectivity, adding another layer of complexity. Variations in the interpretation and labeling of emotions in the training data can affect the model's ability to generalize accurately. Furthermore, the immense variability in how emotions are expressed through facial expressions, gestures, and context complicates the task. Initially, before augmentation, the model's performance was worse due to class imbalance in the dataset. Through validation and evaluation, we confirmed that our augmentation strategy effectively addressed this class imbalance without compromising the integrity of the facial expressions, leading to improved model performance and a more comprehensive understanding of diverse emotional states. We have illustrated the comparison of accuracy and loss for both training and validation over the 50 epochs in a figure.

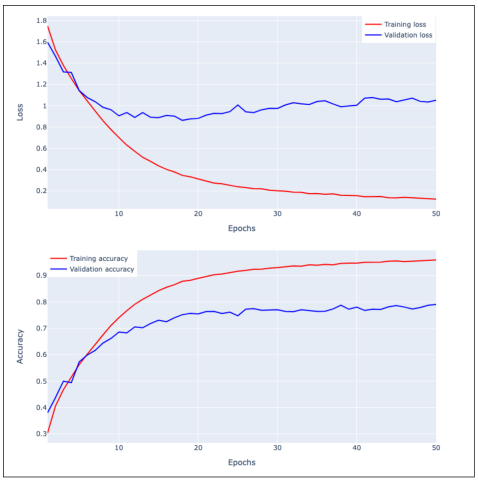

Figure 7. System Architecture

6. Conclusion

In conclusion, we have developed an approach to effectively apply artistic styles based on the emotion in the input image using convolutional neural network-based techniques. We successfully demonstrated the potential of integrating emotions into style transfer, enhancing both the aesthetic and emotional depth of images. In future work, we hope to explore applying multiple artistic styles to content images instead of just one style per image and expanding to more complex and subtle emotional states.

We encountered some limitations in the above implementation, most notably in the critical process of selecting suitable style images for effective style transfer. An incorrect choice may lead to the output image merely adopting the color palette of the style image, lacking the application of distinctive textures from the selected style image. Furthermore, the stylized image outputs could have been better for some emotions. The style loss could have been further minimized if we had sufficient compute resources (a powerful GPU) to train the models for more epochs, typically 50

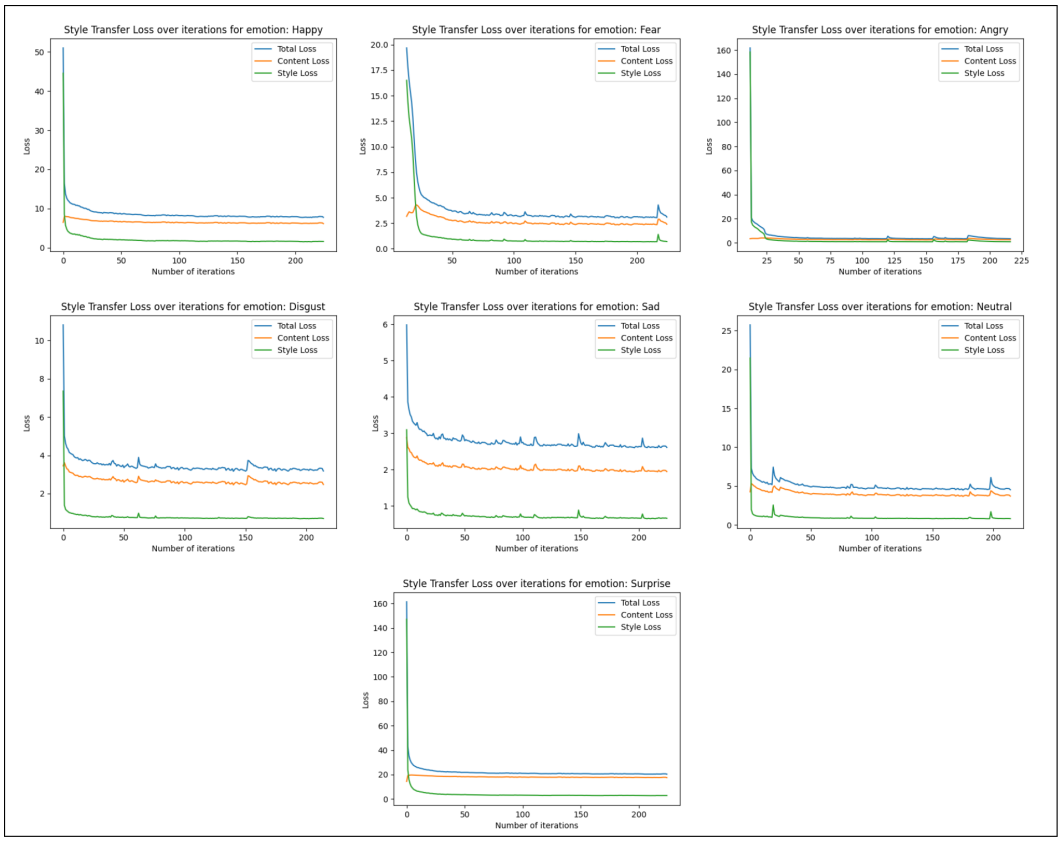

Figure 8. System Architecture

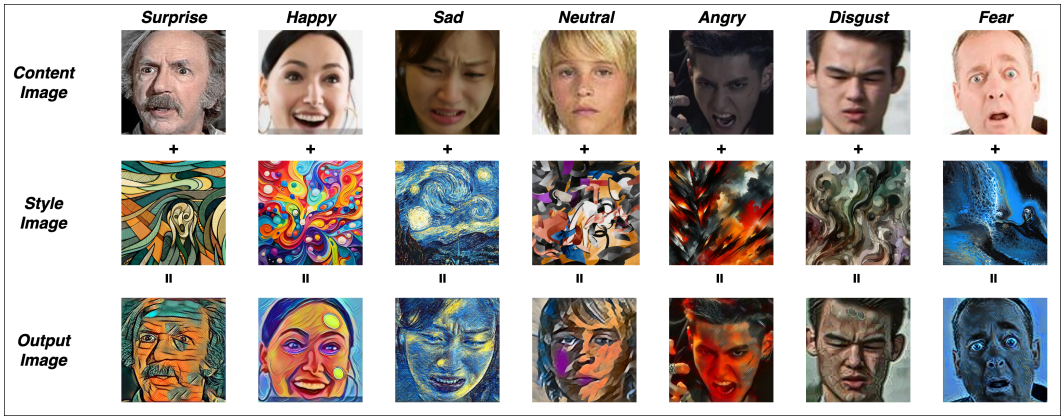

Figure 9. System Architecture

to 100 epochs. This limitation forced us to limit each style transfer model training to a mere 5 epochs.

7. Evaluation Metrics

For our emotion recognition model, we have selected accuracy, precision, and recall as the primary evaluation metrics.

For the neural style transfer operation, we plan to minimize the total loss. The total loss equation is typically represented as follows:

$$L_{\text{total}} = \alpha \cdot L_{\text{content}} + \beta \cdot L_{\text{style}}$$

Where: L_{total} is the total loss. α and β are the weighting factors for the content loss and style loss, respectively.

The content loss ensures that the activations of certain layers are similar between the content image and the gener-

ated image and is computed as follows:

$$L_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^{l}(\vec{x}) - P_{ij}^{l}(\vec{p}))^2$$

Where: $F_{ij}^{l}(\vec{x})$ is the feature map obtained from the generated image $\vec{x}$ at layer l. $P_{ij}^{l}(\vec{p})$ is the feature map obtained from the original content image $\vec{p}$ at layer l. i and j index the feature map's elements.

Style loss is computed as the mean squared error between the Gram matrices of the style image and the generated image across multiple layers:

$$L_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l \cdot E_l$$

Where: - E_l is the style loss for a single layer, computed as:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^{l}(\vec{x}) - A_{ij}^{l}(\vec{a}))^2$$

G_{ij}^{l} is the Gram matrix corresponding to the generated image $\vec{x}$ at layer l. A_{ij}^{l} is the Gram matrix corresponding to the style image $\vec{a}$ at layer l. N_l is the number of feature maps at layer l. M_l is the size of the feature map (height times width) at layer l. w_l is the weighting factor for the contribution of layer l to the style loss.

References