MSIA-432 Deep Learning Project:
A Study on Celebrity Faces

Group 3: Narin Dhatwalia, Jason Summer, Naomi Zhang, Simon Zhu

Spring 2022

Table of Contents

I. Abstract

This project consists of two separate tasks—Binary Classification and Neural Style Transfer. Both tasks require the use of the CelebFaces Attributes (CelebA) dataset. For classification, the objective is to leverage deep learning to accurately determine whether a facial image is of a smiling individual or not. The ground truth labels for the response (i.e. smiling or not) are already annotated within the CelebA dataset. Two separate modeling techniques are explored—Custom CNN and Transfer Learning. The model architectures and performance metrics of both techniques are also analyzed.

For the second task, a facial image of Turkish actor Kenan Imirzalıoğlu is combined with Joe Reimer's *Waiting on Forever* landscape painting through Neural Style Transfer. The content image of the Turkish actor belongs to the CelebA dataset. A different output image is generated via each of the two modeling techniques—direct use of VGG16 and VGG19 with transfer learning. The theory behind loss functions in neural style transfer is introduced, followed by a brief analysis of output images. Convergence values of the loss function are also compared.

The accuracy for classification using a custom model is 91.60% and the accuracy using transfer learning with VGG19 is 91.29%. For the style transfer task, the out-of-box VGG16 generated an image which did not contain painterly strokes and achieved a loss of 223. The transfer learning with VGG19 generated an image with better painterly style and improved color saturation, but lost some detailing of the facial structure. The style transfer loss in this case was 195.

II. Exploratory Data Analysis

Large-scale CelebFaces Attributes dataset, commonly known as CelebA, was used as the basis for model training and prediction/identification. The dataset features more than 200K images of celebrity faces, each of which features 40 binary attribute annotations that describe the facial features and accessories present in the images. Common feature attributes include smiling, baldness, presence of glasses, gender, presence of hat, etc.

Images are uniformly shaped of size 218 x 178 and feature 3-channel pixelation for RGB-color images. Below is a depiction of average red-green-blue pixel values, from 0 to 255, across a randomly selected batch of 32 images.

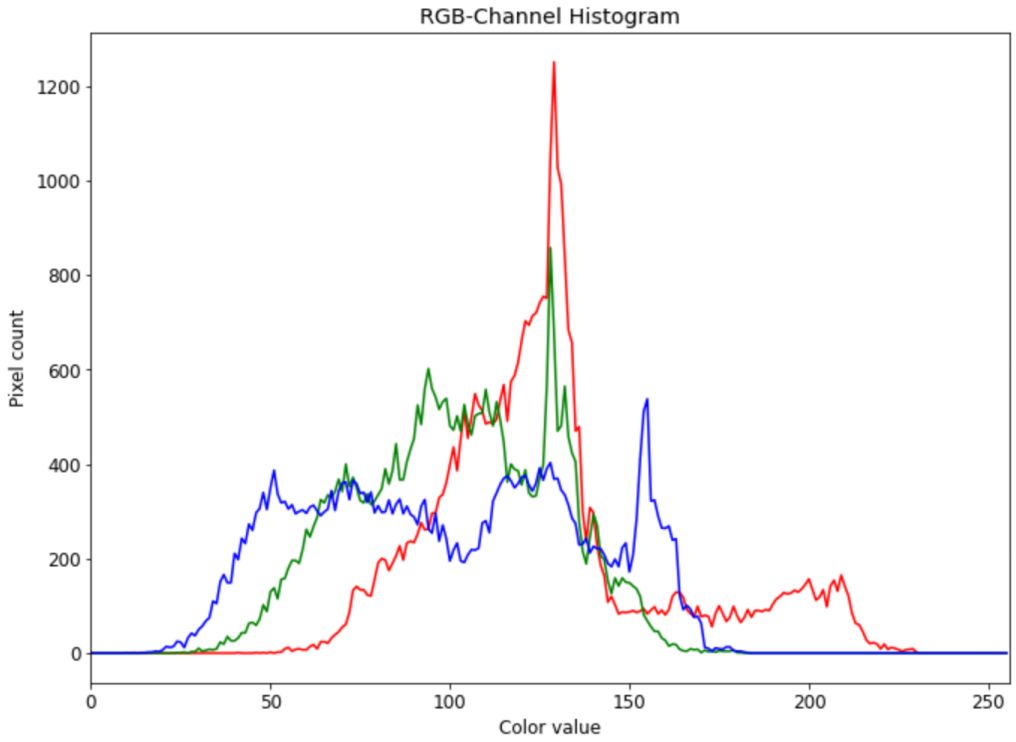

Figure 1: Distribution of red, blue, green pixel values across batch of 32 images

Typical image colors span the full spectrum, providing a diverse set of color scales. The red scale, in particular, can be mostly commonly viewed around 100 to 150 units while blue and green show less central tendency. Substantially influential outliers were not present.

As shown in the below randomly depicted images, the style, quality, pose, and setting of images are very diverse, adding to the innate variation in data samples.

Figure 2: Sample images from CelebA dataset

Of the 200K images, 10,000 images were randomly selected for modeling purposes. Of the 10,000 images, 80% was retained for model training and the remaining 20% served as a final holdout and validation test. All 10,000 sample images were segmented into batch sizes of 32 or 64 (selected based on algorithm performance).

To combat variation in images, training images were augmented by rescaling, shifting, rotating, sheering, magnifying, and flipping. Alternatively, validation images were simply rescaled to leverage learnings from training samples.

III. Model Training and Evaluation: Smiling Classification

The first cognitive problem to solve is smiling classification. The final model takes in a facial image and classifies it as either a smiling or non-smiling face. Two models were trained for this problem: One is a custom sequential model that takes advantage of convolutional layers, while the other adopts transfer learning by retraining the VGG19 model. Both models were evaluated with the accuracy metric.

Out of the 197,600 CelebA dataset images, 51.7% contains non-smiling faces, while the rest contains smiling faces. Therefore, data imbalance is of no concern.

a. Custom Model

As shown in **Figure 1A** in the Appendix, the custom model has 21 layers in total, consisting of three major blocks. Each block consists of two convolutional layers and one max pooling layer of size 2x2. Batch normalization, dropouts and L2 kernel regularizers are also included in each block to combat overfitting.

Each convolutional layer applies kernels of size 4x4, and the number of kernels used increases by a factor of two across blocks, aiming to capture more complicated patterns from the image. The padding for each layer is set so that the output shape after convolution is identical as the input image shape. The layers also utilize the activation function of "*elu*" (as defined below), as it usually converges the cost to zero faster.

$$R(z) = z\,(z > 0)$$
$$R(z) = \alpha(e^{z}-1)\,(z \leq 0)$$

The last layer of the model is a dense layer that condenses all the information from convolutions into two probabilities: one for smiling and the other for non-smiling. To ensure that the probabilities add up to one, an activation function of softmax is used. Aligning with this design, a loss function of "categorical_crossentropy" instead of "binary_crossentropy" is used.

Adam optimizer is used initially to automatically adapt the learning rates. After some iterations, however, the model performance started diverging. Therefore, a manual learning rate scheduler as shown below is set to avoid divergence.

$$
\begin{aligned}
lrate &= 0.001 \ if \ epoch \ \leq 25 \\
lrate &= 0.0005 \ if \ 25 < epoch \ \leq 50 \\
lrate &= 0.0003 \ if \ 50 < epoch \ \leq 100 \\
lrate &= 0.0001 \ if \ epoch \ > 100
\end{aligned}
$$

The final tuned model yielded an accuracy of 0.916, precision of 0.912 and recall of 0.913 on the test set. The ROC curve is displayed below in Figure 3.

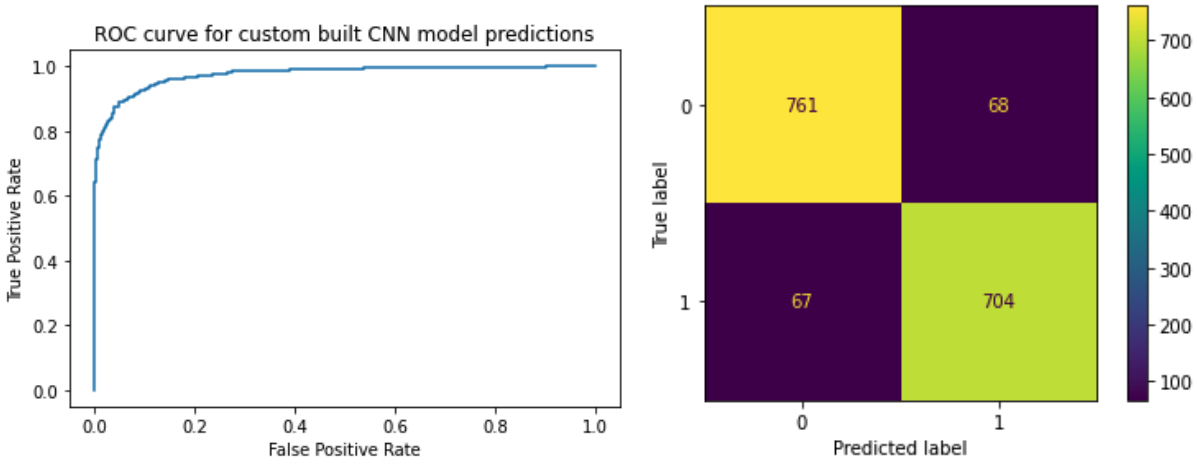

Figure 3: Evaluation results for custom built CNN model predictions

b. Transfer Learning Model VGG19

To complement and compare the custom model, a pre-trained model was constructed utilizing VGG19 as a base. VGG19 is a convolutional neural network consisting of 19 layers trained on more than one million images from the ImageNet database.

The last three layers of VGG19 model base were unfrozen to allow for training on the CelebA dataset but allowing for prior training to be leveraged as well. Additional new custom layers were added on top of the pretrained VGG19 model. Subsequently, both the VGG19 model and custom layers were combined to accurately predict whether or not the celebrity within an image is smiling (binary classification). This allows for "fine-tuning" the higher-order feature representations in the base model in order to make them more relevant for images within the CelebA dataset and for the specific task of identifying smiles in a facial image. The newly created layers are as follows:

- *Conv2D(512, (3, 3), activation='tanh')*
- *Conv2D(512, (3, 3), activation='tanh')*
- *MaxPooling2D((2, 2), strides=(2, 2))*
- *Flatten()*
- *Dense(512, activation='relu')*
- *Dense(1, activation=sigmoid)*

The first two 2D convolution layers above feature 512 filters which the convolutional layers will learn from. The kernels are of size 3x3, with which convolutions will be made into images. After convolution, a tanh activation function is used. The tanh function can be used as a nonlinear activation function between layers of a neural network. It shares a few things in common with the sigmoid activation function. They look very similar; however, while a sigmoid function will map input values to be between 0 and 1, tanh will map values to be between -1 and 1. The relu activation function, which will simply take the maximum of 0 or x and tends to converge faster than alternative activation functions, is used in the second to last (dense) layer, in which all 512 neurons receive input from all neurons in the previous layer.

The pooling layer following the convolutional layers, while without any trainable parameters, reduces the size of their input layer by taking the maximum for every pool size. Following the pooling layer, the output is reshaped in the flattening layer, which has 0 parameters. Instead, it simply unrolls the preceding output shape into a singular shape. The final layer is a densely connected layer, utilizing a sigmoid activation function with a single neuron: Doing so provides an output conducive for a binary classification problem.

Due to the size of images and number of trainable parameters, training was conducted over 20 epochs and both test and validation loss and accuracy monitored across each epoch. The subsequent figure depicts the fall in loss as the training epochs progress.

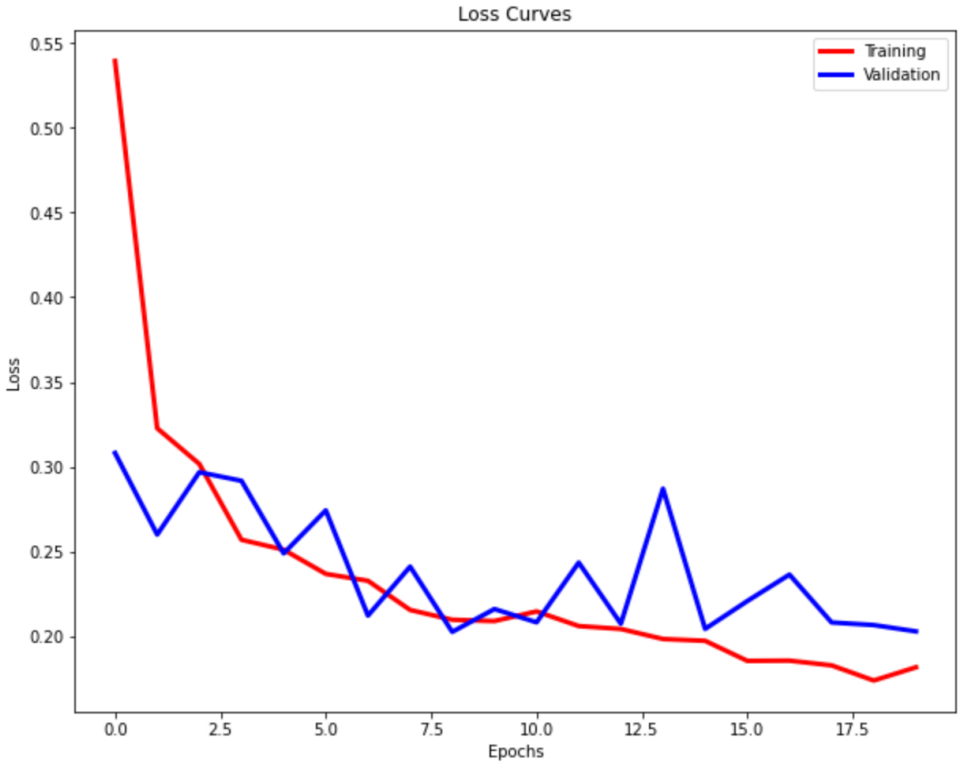

Figure 4: Loss function of training and tests sets from pre-trained VGG19 with additional layers

The 'adam' optimizer was selected, allowing for entirely automated fluctuations in the learning rate as opposed to manually scheduled learning rate changes. However, the initial learning rate was set to 0.01 to allow the model to escape an initial local minima it was often becoming stuck within. Ultimately, a prediction accuracy of 0.913 was achieved on the test set.

The ROC curve and Confusion Matrix are displayed below in Figure 4.

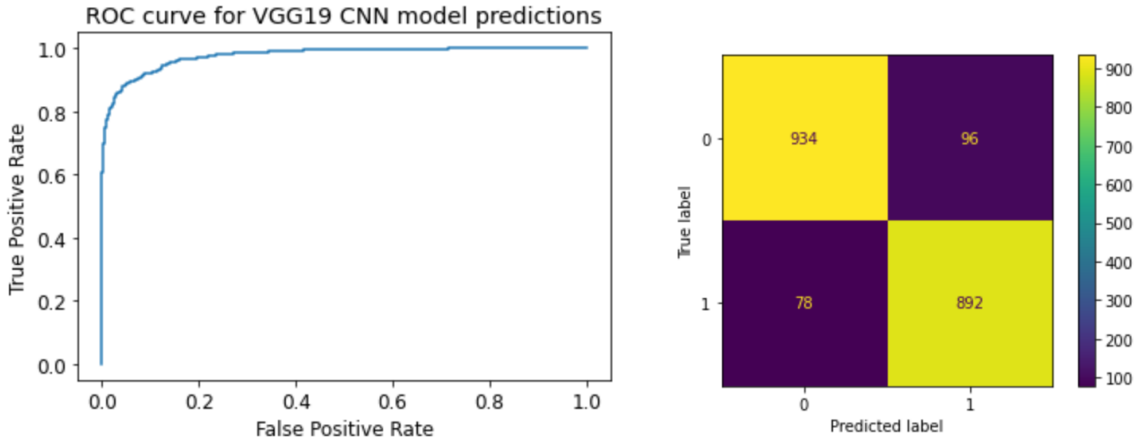

Figure 4: ROC curve and confusion matrix for VGG19 pre-trained classification

IV. Model Training and Evaluation: Style Transfer

The second problem the project delves into is style transfer, which is a technique in deep learning that merges two images. In essence, a "content" image is taken and consolidated with a "style" image, where the content image is depicted in the style of the style image. In this section of the project, we have chosen a specific image from the celebrity dataset of Kenan Imirzalıoğlu and applied style transformation to resemble Joe Reimer's *Waiting on Forever* landscape painting of a field of sunflowers.

In the TensorFlow tutorials for style transformation, the article describes two losses: Content loss and style loss. Content loss compares the loss of content features between the content image and the newly synthesized image, while style loss compares the loss of style features between the style image and the synthesized image. A weighted sum of both is used to evaluate the final image in order to balance content retention and the actual style transfer.

Within the structure of a CNN, intermediate convolutional images are used to describe the content and style of both images. These layers form the feature maps of images, a pixel-level understanding of images. From this understanding, the styles of the images can be understood via the means and correlations of the feature maps. In this case, a Gram matrix perfectly captures this information, and is subsequently used to calculate the style loss. We define the style loss as the sum of squared loss between the gram matrices of the style image and the synthesized image, divided by a product of squared channels and the squared size of the image,

$$L(c) = \sum (S - C)^2 / (4 * channels^2 * size^2),$$

where S is the gram matrix of the style image and C is the gram matrix of the synthesized (combined) image. The channels refer to the RGB channels, where it takes a constant value of 3.

The content loss, on the other hand, is simpler in that it directly compares the content image to the synthesized image, and calculates the sum of squared loss,

$$L(s) = \sum (synthesized - combination)^2,$$

The total loss, then, sums the two losses,

$$L = \sum (w_C L(c) + w_L L(s)),$$

Where the content weight loss w_C is a predefined constant and the style weight loss w_L is a predefined constant divided by the total number of style layers.

Following, in this project, we applied two different models to the previously noted style transfer problem. The first model is a pretrained VGG16 with ImageNet weights. The second is a transfer learning VGG19 model with all but the last few convolutional layers frozen, and some custom layers added and trained on the CelebA dataset.

a. Pretrained Model VGG16

The first model chosen is a pretrained VGG16 model on the widely used, general ImageNet dataset. The structure of a VGG16 model is shown in the Appendix, **Figure 3A**, made up of multiple convolutional layers and max pooling, and a final softmax activation layer.

Though the model comes pretrained, since the weights used come from ImageNet, the weights do not necessarily reflect and represent the images of our dataset composed of only facial images. The model's ability to style transfer is likely an issue when the synthesized image is created. Below is the output after 5000 iterations.

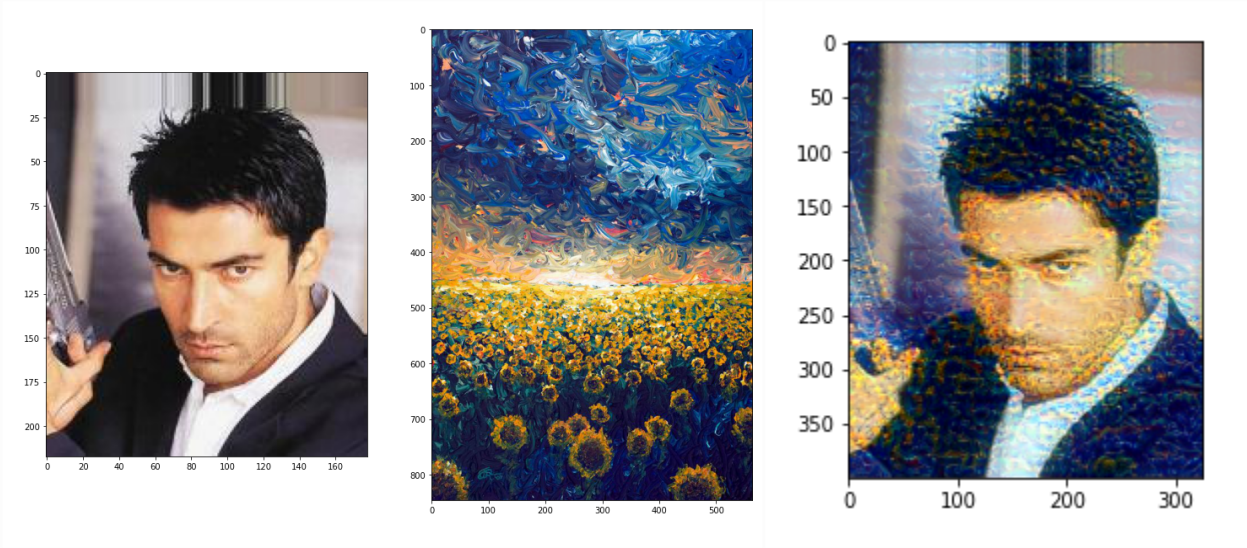

Figure 5: Content (left) and style (center) images, and the final synthesized image (right) from pretrained VGG16 model

The final loss for this content and style combination is 223.39. The synthesized image, as shown, has lost some of its coloring where the celebrity's face becomes more yellow and the darker tones have blue hues. This makes the style image's color scheme. The theme, on the other hand, added pixelation and blur to the content image, likely resulting in the higher loss value. The distinct painting strokes fail to transfer onto the synthesized image, unfortunately.

b. Transfer Learning Model VGG19

A saved neural network that was already trained on a large dataset is referred to as a "pre-trained model". Most often, the training was executed on a large-scale image classification problem. It's possible to leverage the pretrained model as-is or use transfer learning to alter the model for more specific use-cases.

The major advantage of transfer learning is that users can build upon the previously learned feature maps without having to retrain from scratch. The pre-trained model has already been trained on ImageNet, a large and general enough dataset to capture information in generic representation of the visual world.

In this project, we made use of VGG19 which is already trained on the imagenet dataset and available within the TensorFlow library. The final classification part of the pretrained model is specific to the original classification task, and subsequently specific to the set of classes on which the model was trained.

Therefore, we dropped the output layer when importing VGG19. Next, we combined two approaches to customize VGG19 for the purpose of implementing neural style transfer:

1. Fine-Tuning: In this step, we unfroze the last eight layers of the frozen VGG19 model base and added six new custom layers on top of the pretrained model. We jointly trained both the newly-added layers and the unfrozen layers of the base VGG19 to accurately predict whether or not the celebrity within an image is smiling. The newly created layers are the same as the layers outlined in Section III (b). The final accuracy obtained on the validation set is 0.91. The accuracy curves for training and validation are visualized below:

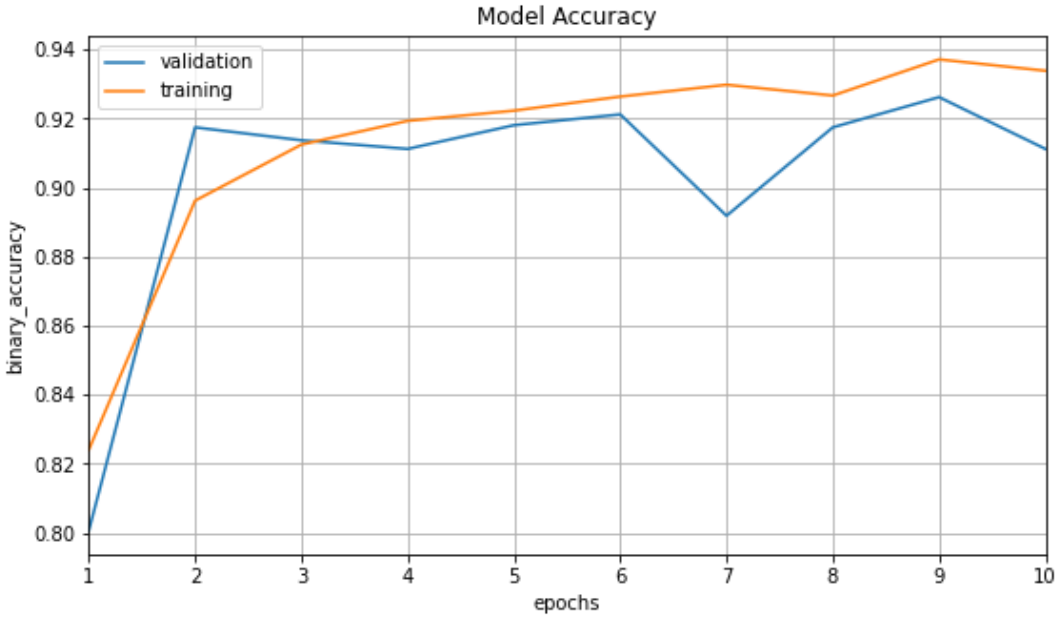

Figure 6: Model accuracy on the training and validation datasets

2. Feature Extraction: The goal here was to use the representations learned by VGG19 to extract meaningful features from new samples in the CelebA dataset. Both the content layer and the style layers (needed for calculating loss during neural style transfer) were modified to incorporate custom layers. After training the VGG19 model with custom layers for the binary classification problem, we set the style layers to be the following:

   - [*"block1_conv1", "block2_conv1", "block3_conv1", "block4_conv1", "block5_conv1", "block6_conv1"*]

   For the content layer, we selected only "*block6_conv2*" of the architecture. Note that "*block6_conv1*" and "*block6_conv2*" are custom layers that were added separately and were not present in the original VGG19 architecture.

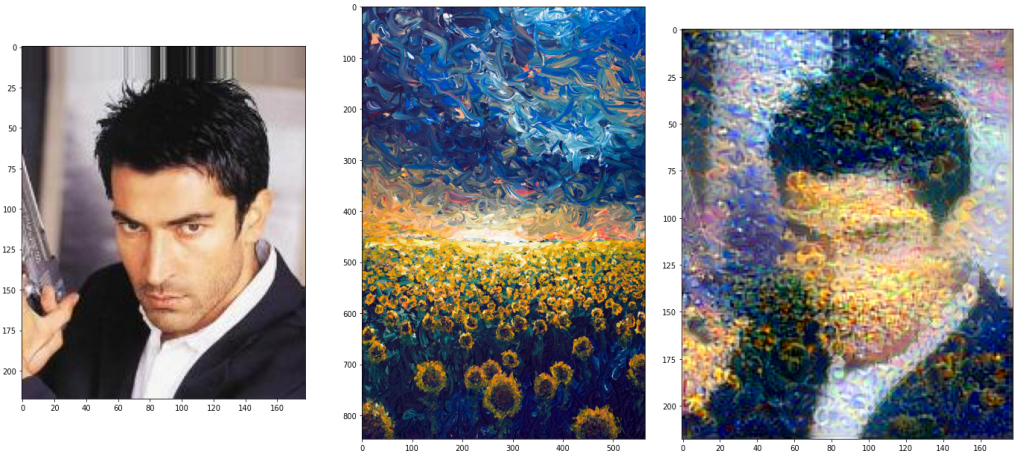

Figure 7: Content (left) and style (center) images, and the final synthesized image (right) from the transfer-learning VGG19 model

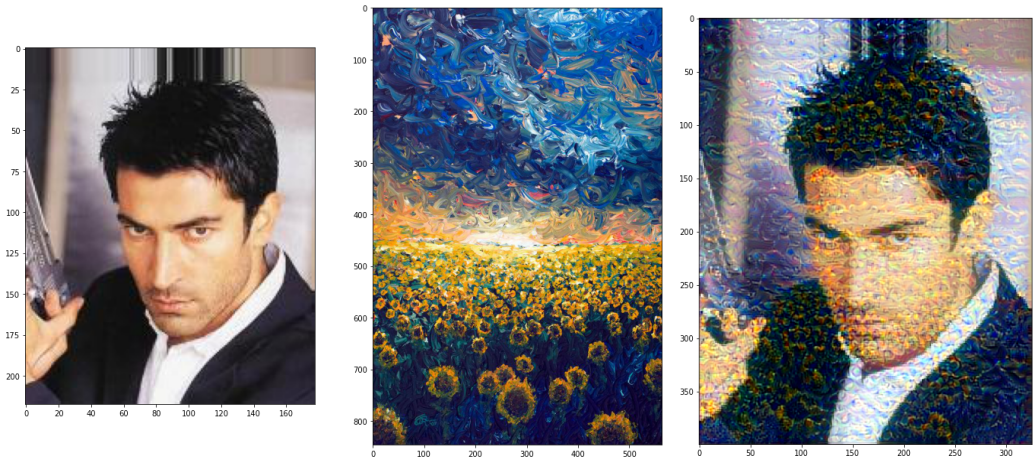

Figure 8: Content (left) and style (center) images, and the final synthesized image (right) from a pretrained VGG19 model (no transfer learning, all layers frozen)

The style transfer loss for the out-of-box VGG19 is 204.39, whereas the transfer learning model has a loss of 195.63. Both losses were calculated after 10,000 iterations. The out-of-box VGG implementations show a superior transfer of the content, however they do not capture the style image in a way that replicates the painterly brushstrokes of Joe Reimer's painting.

In contrast, the image generated by the transfer learning model is a much more "artistic" fusion of the style and content image. This can be attributed to the model's feature maps being modified to learn new parameters which are more relevant to the Celebrity Faces dataset. Since the content image also belongs to the Celebrity Faces dataset, the model can achieve superior performance on style transfer tasks as compared to out-of-box VGG models. One possible drawback of using the transfer learning model is that the output image appears slightly blurry. The model somewhat de-emphasizes the details present in the content image in exchange for better replication of style.

V. Model Operations

The trained models would be deployed via a web application. Further model iterations would also occur based on the interactions and feedback from the application. This section demonstrates how the model would be operated and maintained over time.

a. Model Deployment

Since the project mainly tackles cognitive problems of image classification and style transferring, a user interface (UI) would be essential to apply the model in real life. For smiling classification, the interface would ask the user to upload a properly-sized image, resize it to match the input shape of the model, run through the model to get an inference, and return the classified label to the user.

Similarly for the style transferring, the user would upload two images to the interface, one for transformation and the other for style reference. Then, the model would take two inputs and generate an image that's the product of style transfer.

The tuned models would be stored in an AWS S3 bucket, while the web user interface would be implemented via Flask and placed onto some platforms such as AWS ECS. The users could then access the interface from the Internet. Whenever the user uploads the images for a specific task, the web program would load the corresponding model from the S3 bucket, preprocess the images, feed it into the model and then output the results onto the interface.

A feedback loop will also be included in the deployment design. After the user's task is completed, the interface would prompt for the user's satisfaction or feedback about the model outputs. It would then store the user's uploaded images as well as the reactions or labels into the S3 bucket for future model iterations.

b. Model Maintenance

The feedback collected from users would serve as the input for the next model iteration. For smiling classification, the newly uploaded images and the labels provided by the user would be added into the current CelebA dataset. After sufficient additions, the model would be retrained on the new dataset to account for potential new arising complications in smile detection.

On the other hand, evaluating style transfer is a more subjective task. Current loss functions only consider style and content loss. Should there be enough user labels, they could also be incorporated into the loss functions. The models could then be further trained to account for user satisfaction.

VI. Conclusion

Additional resources and references can be found in the References section below. We discuss briefly our findings for this project and potential further work to expand upon this project.

a. Findings

Throughout this project, we explored a classification and style transfer problem on the widely used CelebA dataset. This dataset consisted of cropped celebrity faces with various labels added, which provided enough information for classification as well as simple images for style transfer. Images were

found to contain the default RGB-color schema and observe diversity in facial images and features (expression, facing, colorization) despite the apparent uniformity of the dataset.

The classification problem was simplified into a binary categorization of whether or not the observed facial image is smiling. The baseline model was a custom CNN model of 21 layers with a result accuracy of 0.916. The contender model chosen was a transfer learning model with VGG19 as the base. The last three layers were unfrozen to allow feature training on the facial dataset, and additional convolutional and dense layers were added. The final accuracy was found to be 0.913, slightly lower than the comparison model. The slight difference could be attributed to fluctuations in the training sample.

The style transfer problem was similarly a simple problem of altering the style of a facial image of Kenan Imirzalıoğlu to resemble Joe Reimer's *Waiting on Forever* landscape painting of a field of sunflowers. The first model was a pretrained VGG16 model with weights trained on the generic ImageNet dataset. The total loss was 223.39, with the synthesized image resembling the color scheme of Joe Reimer's painting. However, the result did not possess the painterly strokes of Joe Reimer. The contender model was, similar to the classification problem, a transfer learning model with VGG19. The total loss was 195.63, with the synthesized image having much better painterly style and gained color saturation, but lost the detailed facial structure and appeared almost blurry.

b. Further Work

As direct extensions of our work, the CelebA dataset is a very detailed and well-documented dataset with proper labels: For instance, images are labeled with masks, sunglasses, and various other accessories. In this case, the classification problem can be extended to more than a simple binary classification. With more time and computing power, as an example, a classification model can be built to identify facial expressions. Due to the complex nature of the images, a more complex pretrained model such as EfficientNet or DenseNet can be employed, as the VGG models perform better with simpler images.

This extension similarly applies to our work with style transfer: Instead of employing VGG models, more complex pretrained models can be used to compare the synthesized images. With more research into the structure of each model, a more suitable model for style transfer can be built to more accurately locate the style and content layers, thus minimizing the content loss while maximizing the amount of style preservation.

Additionally, with the appropriate amount of labels, a CNN can be trained for object detection on unlabeled data as well. Specifically, the same celebrity dataset can be used to train for a facial recognition problem (Zhong et al.). Extension of the facial recognition problem involves facial identification (Guo et al.) and verification. Sun et al. used this very dataset to train facial clustering on unlabeled data in an attempt to provide decreased image processing times. Subsets of this data exist with proper segmentation of facial images split to identify facial structures, which can help further recognition problems. Similar datasets were used to identify pose and age (Cao et al.), as well as developing models that better handle noisy or blurred inputs (Meng et al.).

One interesting avenue we explored briefly was facial image generation using model structure similar to VGG. The code and details can be found in the References section by Francois Chollet. The training step

of a Keras Sequential model was overridden, where random pixels were sampled across various images. Then, the outputs were randomly assembled with real images and labeled to be trained with a discriminator of real versus fake imagery. Following, a generator feature was similarly trained from the discriminator to create realistic synthetic images. Below are some outputs after 200 epochs.

Figure 9: Generated images with DCGAN

Though the synthesized images look like something out of our nightmares, the simple implementation of DCGAN allowed the quick generation of facial images. With more complex model structures and additional research, realistic facial images can be generated.

References

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). *VGGFace2: A dataset for recognising faces across pose and age* (arXiv:1710.08092; Version 2). arXiv. http://arxiv.org/abs/1710.08092

CelebA Dataset. (n.d.). Retrieved May 25, 2022, from https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

Chollet, F. (2019, April 29). *Keras documentation: DCGAN to generate face images*. Retrieved May 25, 2022, from https://keras.io/examples/generative/dcgan_overriding_train_step/

Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). *MS-Celeb-1M: A Dataset and Benchmark for Large-Scale Face Recognition* (arXiv:1607.08221). arXiv. https://doi.org/10.48550/arXiv.1607.08221

How to code Neural Style Transfer in Python. (2020, October 27). *Ander Fernández*. https://anderfernandez.com/en/blog/how-to-code-neural-style-transfer-in-python/

How to Develop a CNN From Scratch for CIFAR-10 Photo Classification. (n.d.). Retrieved May 25, 2022, from https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/

Joe Reimer Art—Western Canadian Landscape Paintings. (n.d.). Joereimerart. Retrieved May 25, 2022, from https://www.joereimer.com

Meng, Q., Zhao, S., Huang, Z., & Zhou, F. (2021). *MagFace: A Universal Representation for Face Recognition and Quality Assessment* (arXiv:2103.06627; Version 4). arXiv. http://arxiv.org/abs/2103.06627

Neural style transfer | TensorFlow Core. (n.d.). Retrieved May 25, 2022, from https://www.tensorflow.org/tutorials/generative/style_transfer

Sun, D., Yang, K., & Ding, Z. (2022). Confidence-Based Simple Graph Convolutional Networks for Face Clustering. *IEEE Access*, *10*, 6459–6469. https://doi.org/10.1109/ACCESS.2022.3142922

Tf.keras.preprocessing.image.ImageDataGenerator | TensorFlow Core v2.9.1. (n.d.). TensorFlow. Retrieved May 25, 2022, from https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator

VGG16 architecture. (2019, January 21). OpenGenus IQ: Computing Expertise & Legacy. https://iq.opengenus.org/vgg16/

VGG-19 convolutional neural network—MATLAB vgg19. (n.d.). Retrieved May 25, 2022, from https://www.mathworks.com/help/deeplearning/ref/vgg19.html;jsessionid=3dad7761024d4453f3c3863d0fc5

Zhong, Y., & Deng, W. (2021). *Face Transformer for Recognition* (arXiv:2103.14803). arXiv. http://arxiv.org/abs/2103.14803

Appendix

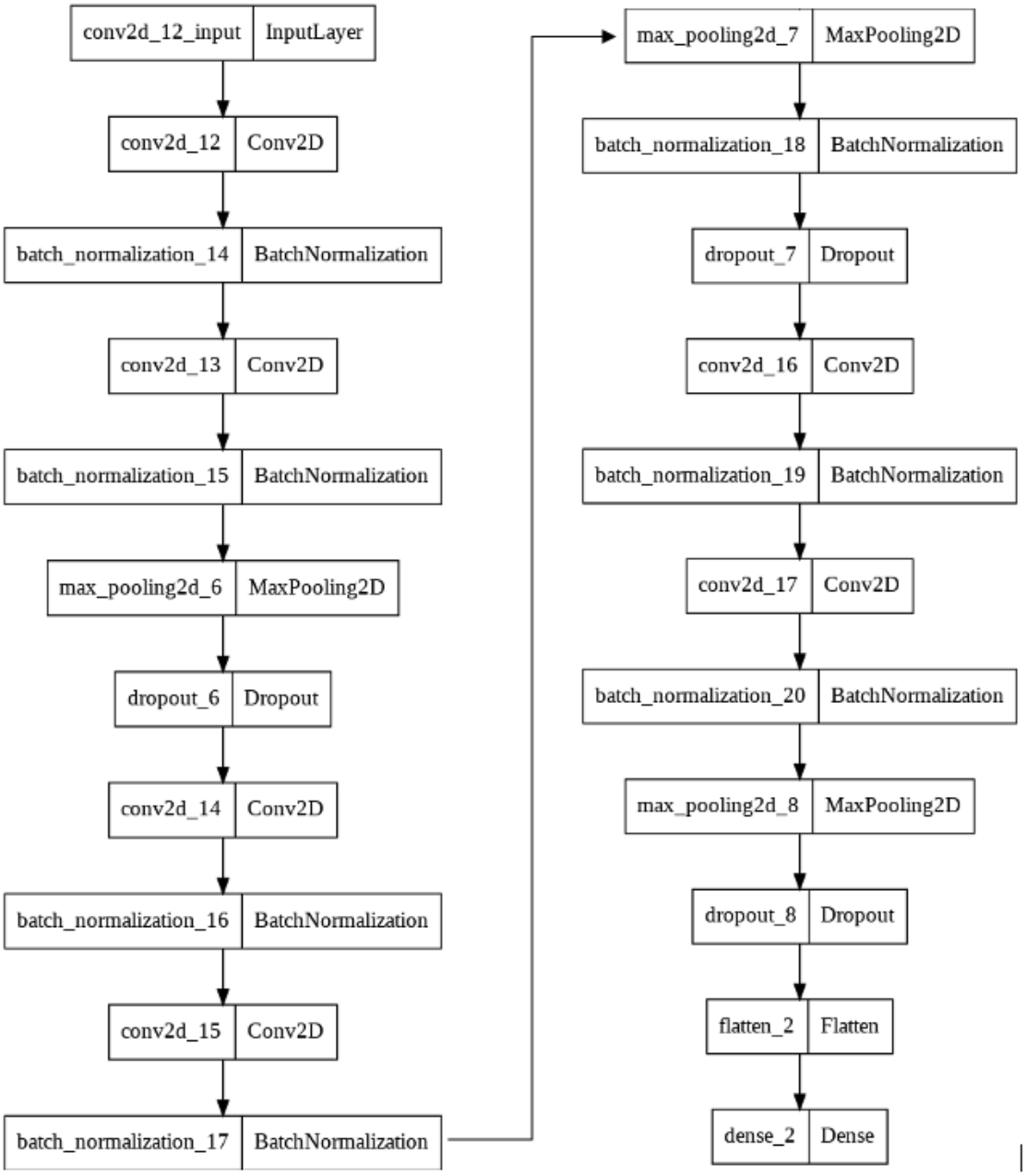

Figure 1A: Custom model architecture for smiling classification

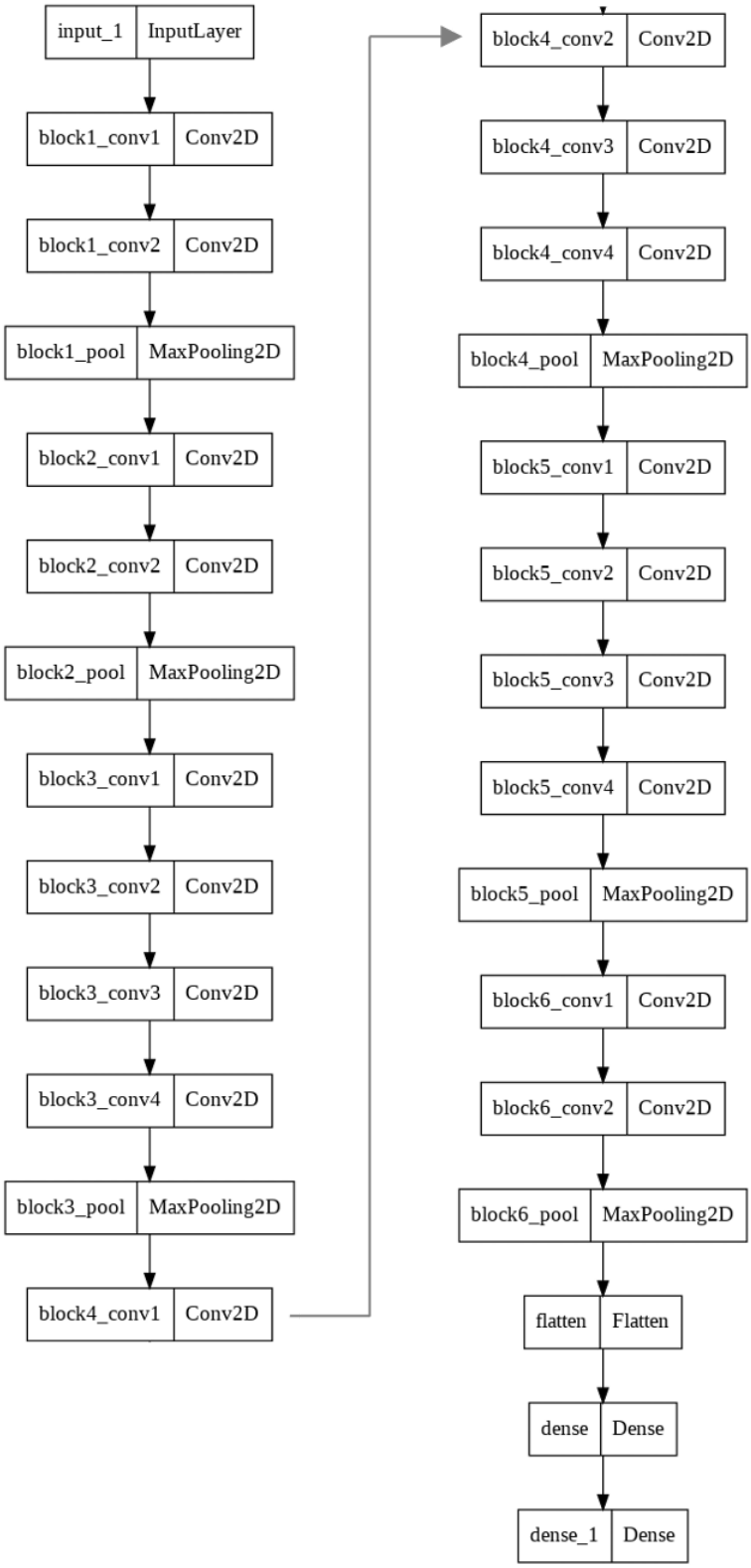

Figure 2A: Pre-Trained VGG19 with additional layers model architecture for smiling classification

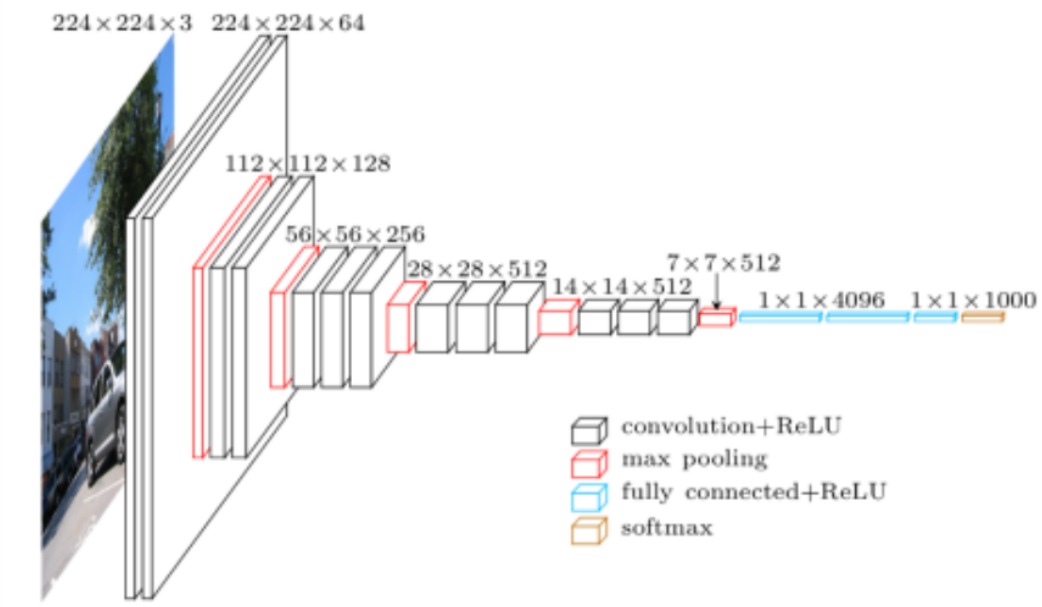

Figure 3A: VGG16 model architecture for style transfer